

Biometric Based Door Lock

A PROJECT REPORT

Submitted By

Aditi Srivastava

2000290140009

Ritesh Kumar Bharti

2000290140103

**Submitted in partial fulfillment of the
Requirements for the Degree of**

MASTER OF COMPUTER APPLICATIONS

Under the Supervision of

Mr. Ankit Verma

Assistant Professor



Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS

KIET Group of Institutions, Ghaziabad

Uttar Pradesh-201206

CERTIFICATE

Certified that **Aditi Srivastava(2000290140009), Ritesh Kumar Bharti (2000290140103)** have carried out the project work having “Biometric Based Door Lock” for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself / herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Aditi Srivastava(2000290140009)

Ritesh Kumar Bharti(2000290140103)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Mr. Ankit verma

**Assistant Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad**

Signature of Internal Examiner

Signature of External Examiner

**Dr. Ajay Shrivastava
Head, Department of Computer Applications KIET
Group of Institutions, Ghaziabad**

ABSTRACT

This paper presents the development and implementation of a Fingerprint Sensor Based Door Lock System that will automatically unlock a door when a registered fingerprint is sensed. The method employed in accomplishing this involves the use of a fingerprint scanner R305 interfaced with ATMEGA 328 Arduino microcontroller to actuate the locking and unlocking process of a door. Once a registered finger print is placed on the sensor, access is granted to the user, the door slides open and it closes after five seconds. During this process, the 16x2 Liquid Crystal Display (LCD) displays the name of the individual with the registered fingerprint. If an unregistered fingerprint is sensed, access is denied. The developed Biometric Based Door Lock System was tested and it functioned in line with the desired objectives.

ACKNOWLEDGEMENTS

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, **Mr. Ankit Verma** his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Ajay Kumar Shrivastava, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Aditi Srivastava

Ritesh Kumar Bharti

List of Chapters

Chapter 1: Introduction

- 1.1 Overall description
- 1.2 Project Scope
- 1.3 Hardware / Software requirements

Chapter 2: Feasibility Study

- 2.1 Technical feasibility
- 2.2 Operational Feasibility
- 2.3 Behavioral Feasibility
- 2.4 Operational Feasibility

Chapter 3: Database Design

- 3.1 Flow Chart
- 3.2 Block Diagram
- 3.3 Circuit Diagram

Chapter 4: Working

Chapter 5: Testing

- 5.1 Unit Testing
- 5.2 Integration Testing
- 5.3 Software Verification and Validation
- 5.4 Black Box Testing
- 5.5 System Testing
- 5.6 Test Cases

Chapter 6: Design Issues

Chapter 7: Design Requirements

Chapter 8: Advantages & Disadvantages

Chapter 9: Application

Chapter 10: Future Improvements

Chapter 11: Problem Formulation

Chapter 12: Conclusion

Chapter 13: Output

Chapter 14: Coding

Chapter 15: Appendix

Chapter 16: Bibliography

Chapter 1 Introduction

1.1 Overall Description

1.1.1 Product Perspective

Biometric systems have overtime served as robust security mechanisms in various domains. Fingerprints are the oldest and most widely used form of biometric identification. The use of fingerprint for identification has been employed in law enforcement for about a century. A much broader application of fingerprint is for personal authentication, for instance to access a computer, a network, an ATM machine, a car or a home.

Electronic lock using fingerprint recognition system is a process of verifying the fingerprint image to open the electronic lock. This project highlights the development of fingerprint verification. Verification is completed by comparing the data of authorized fingerprint image with incoming fingerprint image. Then the information of incoming fingerprint image will undergo the comparison process to compare with authorized fingerprint image.

Fingerprint door lock incorporates the proven technology. Fingerprint reader scanning is the most mature and tested type of biometric technology. Recent studies on biometrics have shown that compared to the hand method, fingerprint is more accurate and cost-effective. The duplication of biometric fingerprint technology is virtually impossible, only one in one billionth of a chance. Biometric security guarantees a positive method of user identification with something that cannot be lost, replicated or stolen.

1.1.2 Product Features

The software described in this SRS will be used to detect people's emotions. This project can be used in several areas that like to measure customer satisfaction in a marketing platform, helping advertisers to sell products more effectively.

1.3 HARDWARE/SOFTWARE USED IN PROJECT

1.3.1 Hardware Requirement

ARDUINO MEGA (2560):

Arduino is an open-source electronic platform based on easy-to-use hardware and software. Arduino boards are able to read inputs – light on sensor, a finger on a button, or a Twitter message – and turn it into an output – activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on writing), and the arduino software (IDE), based on processing.

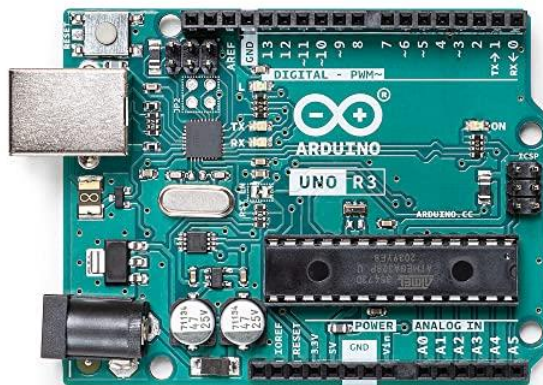
The power pins description

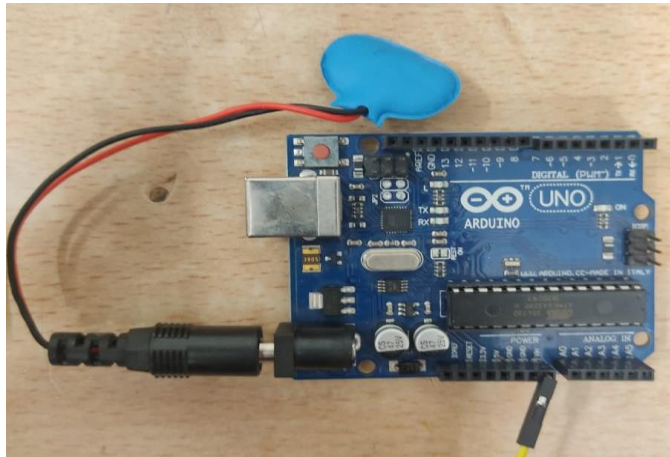
+VIN:- The input voltage to the Arduino board when it's using an external power source (as opposed to volt from the USB connector or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access through this pin.

+5V:- The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN an on-board regulator, or be supplied but USB or another 5V supply.

+3V3:- A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50mA.

GND:- Ground pins.

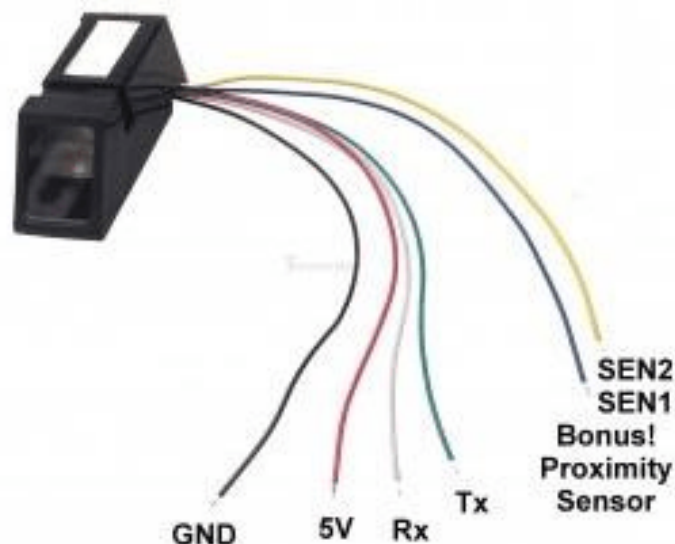




FINGERPRINT SENSOR

A fingerprint is an impression left by the friction ridges of a human finger. The recovery of partial fingerprints from a crime scene is an important method of forensic science. Moisture and grease on a finger result in fingerprints on surfaces such as glass or metal. Deliberate impressions of entire fingerprints can be obtained by ink or other substances transferred from the peaks of friction ridges on the skin to a smooth surface such as paper. Fingerprint records normally contain impressions from the pad on the last joint of fingers and thumbs, although fingerprint cards also typically record portions of lower joint areas of the fingers.

Human fingerprints are detailed, nearly unique, difficult to alter, and durable over the life of an individual, making them suitable as long-term markers of human identity. They may be employed by police or other authorities to identify individuals who wish to conceal their identity, or to identify people who are incapacitated or deceased and thus unable to identify themselves, as in the aftermath of a natural disaster.



BATTERY

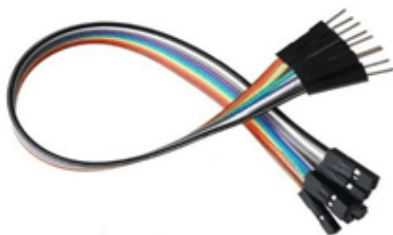
The nine-volt battery, or 9-volt battery, is a common size of battery that was introduced for the early transistor radios. It has a rectangular prism shape with rounded edges and a polarized snap connector at the top. This type is commonly used in walkie-talkies, clocks and smoke detectors.

The nine-volt battery format is commonly available in primary carbon-zinc and alkaline chemistry, in primary lithium iron disulfide, and in rechargeable form in nickel-cadmium, nickel-metal hydride and lithium-ion. Mercury-oxide batteries of this format, once common, have not been manufactured in many years due to their mercury content. Designations for this format include NEDA 1604 and IEC 6F22 (for zinc-carbon) or MN1604 6LR61 (for alkaline). The size, regardless of chemistry, is commonly designated PP3—a designation originally reserved solely for carbon-zinc, or in some countries, E or E-block.



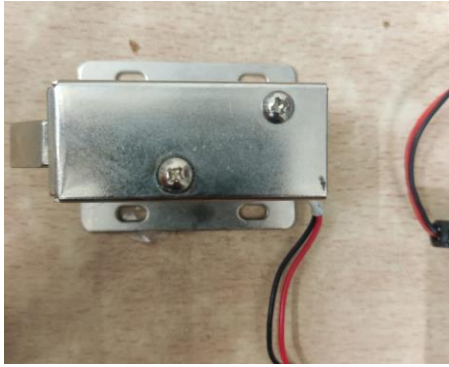
JUMPER WIRE

A jump wire is an electrical wire, or group of them in a cable, with a connector or pin at each end, which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering.



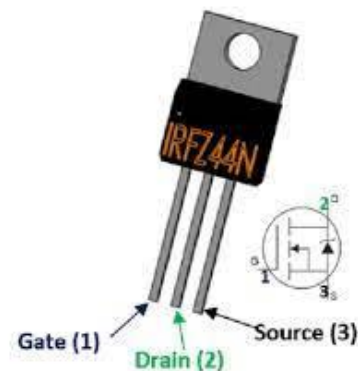
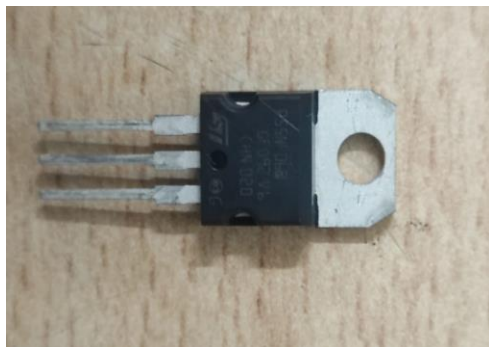
SOLINOID DOOR LOCK

This 12V Electromagnetic Solenoid Lock can be used for locking storage shelves, file cabinets, etc. You can control the opening and closing of the lock with the help of programming. But for the lock to work, it must be connected to a power supply as this is an electromagnetic lock. The lock works as the current disconnects, and it will unlock as the instant power is on.



IRFZ44N MOSFET

IRFZ44N is an advanced HEXFET Power MOSFET that utilizes advanced processing techniques to achieve extremely low on-resistance per silicon area. This benefit, combined with the fast switching speed and ruggedized device design that ex FAT power MOSFETs are well known for, provides the designer with an extremely efficient and reliable device for use in a wide variety of applications.



LED

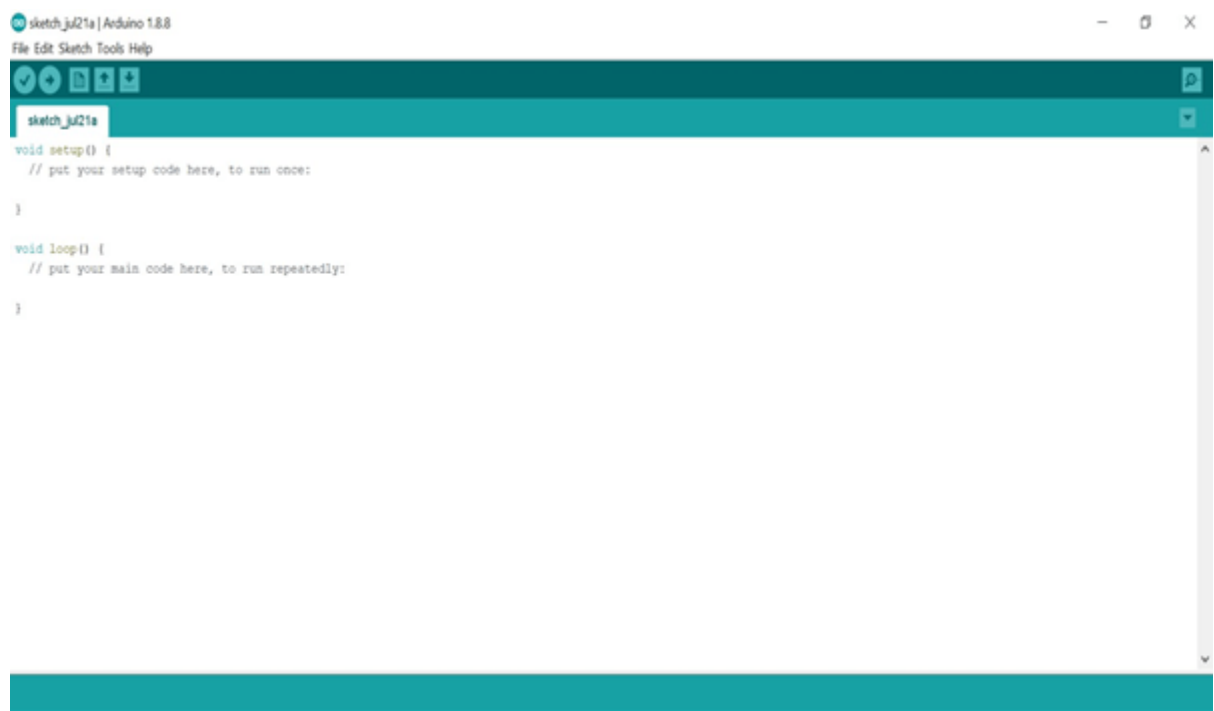
A light-emitting diode (LED) is a semiconductor light source that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons. This effect is called electroluminescence. The color of the light (corresponding to the energy of the photons) is determined by the energy required for electrons to cross the band gap of the semiconductor. White light is obtained by using multiple semiconductors or a layer of light-emitting phosphor on the semiconductor device.



1.3.2 Software Requirement

ARDUINO IDE (R3):

The open-source Arduino software (IDE) makes it easy to write code and upload it to the board. It runs on windows, MAX OS X, and Linux. The environment is written in java and based on processing and other open-source software. The Arduino IDE (integrated Development Environment or Arduino software (IDE)) contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common function and a series of menus. It connects to the Arduino and Genuine hardware to upload the program and communicate with them.



Chapter 2

2. FEASIBILITY STUDY

A feasibility study is a high-level capsule version of the entire System analysis and design Process. The study begins by classifying the problem definition. Feasibility is to determine if it's worth doing. Once an acceptance problem definition has been generated, the analyst develops a logical model of the system. A search for alternatives is analyzed carefully. There are 3 parts in feasibility study.

2.1 TECHNICAL STUDY

This involves questions such as whether the technology needed for the system exists, how difficult it will be to build, and whether the firm has enough experience using that technology. The assessment is based on outline design of system requirements in terms of input, processes, output, fields, programs, and procedures. This can be qualified in terms of volume of data, trends, frequency of updating in order to introduce the technical system. The application is the fact that it has been developed on windows 10 platform and a high configuration of 8GB RAM on Intel Pentium Dual core processor. This is technically feasible. The technical feasibility assessment is focused on gaining an understanding of the present technical resources of the organization and their applicability to the expected needs of the proposed system. It is an evaluation of the hardware and software and how it meets the need of the proposed system.

2.2 OPERATIONAL STUDY

Operational feasibility is the measure of how well a proposed system solves the problems and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. The operational feasibility assessment focuses on the degree to which the proposed development projects fits in with the existing business environment and objectives with regard to development schedule, delivery date, corporate culture and existing business processes. To ensure success, desired operational outcomes must be imparted during design and development. These include such design-dependent parameters as reliability, maintainability, supportability, usability, producibility, disposability, sustainability, affordability, and others. These parameters are required to be considered at the early stages of design if desired operational behaviors are to be realized. A system design and development require appropriate and timely application of engineering and management efforts to meet the previously mentioned parameters. A system may serve its intended purpose most effectively when its technical and operating characteristics are engineered into the design. Therefore, operational feasibility is a critical aspect of systems engineering that needs to be an integral part of the early design phases.

2.3 BEHAVIORAL STUDY

Establishing the cost-effectiveness of the proposed system i.e., if the benefits do not outweigh the costs, then it is not worth going ahead. In the fast-paced world today there is a great need of online social networking facilities. Thus, the benefits of this project in the current scenario make it economically feasible. The purpose of the economic feasibility assessment is to determine the positive economic benefits to the organization that the proposed system will provide. It includes quantification and identification of all the benefits expected. This assessment typically involves a cost/benefits analysis.

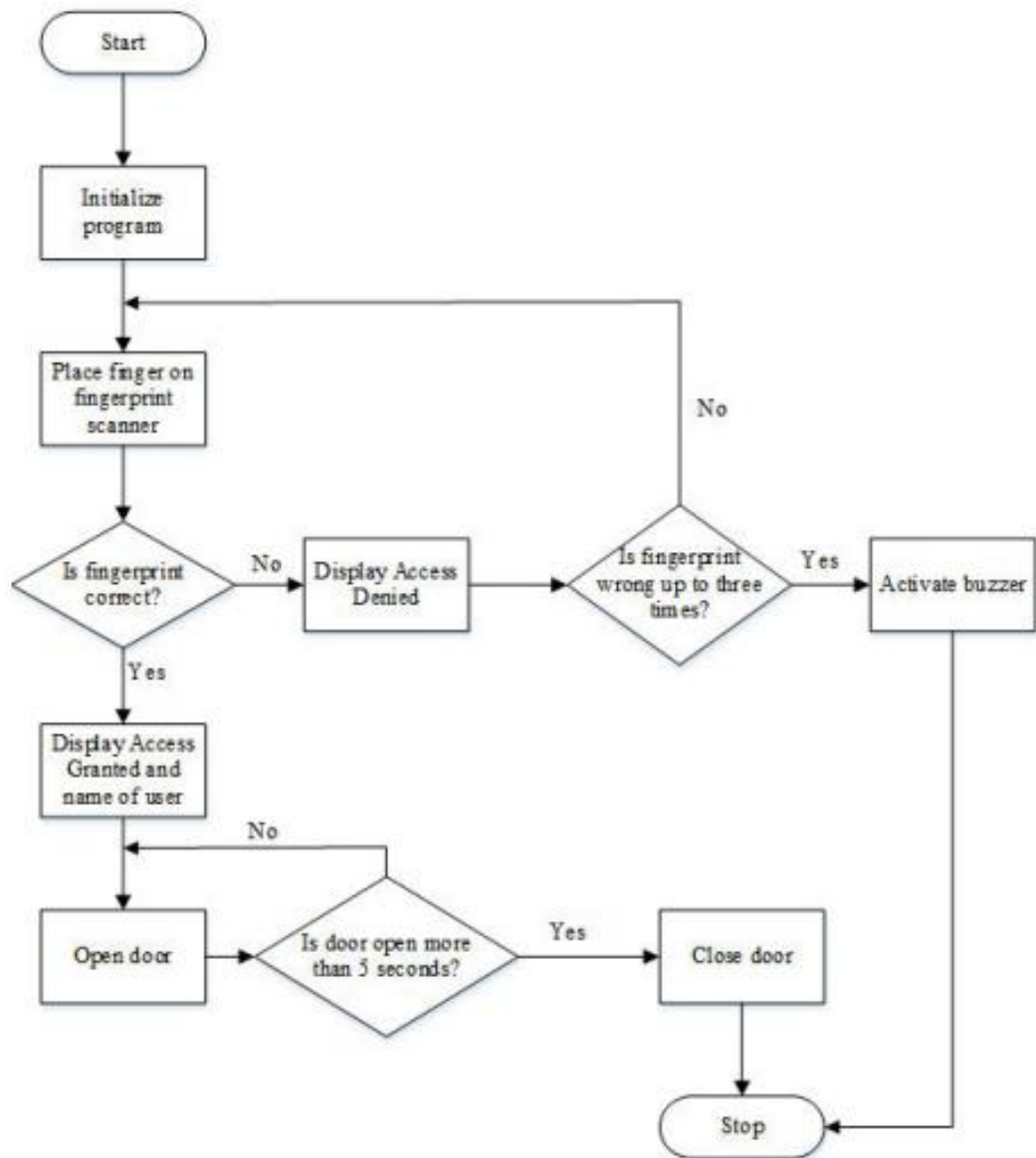
Chapter 3 Database Design

3.1. Flowchart:

Flowchart is a graphical representation of an algorithm. Programmers often use it as a program-planning tool to solve a problem. It makes use of symbols which are connected among them to indicate the flow of information and processing. The process of drawing a flowchart for an algorithm is known as “flowcharting”.

Basic Symbols used in Flowchart Designs

1. **Terminal:** The oval symbol indicates Start, Stop and Halt in a program’s logic flow. A pause/halt is generally used in a program logic under some error conditions. Terminal is the first and last symbols in the flowchart.
2. **Input/output:** A parallelogram denotes any function of input/output type. Program instructions that take input from input devices and display output on output devices are indicated with parallelogram in a flowchart.
3. **Processing:** A box represents arithmetic instructions. All arithmetic processes such as adding, subtracting, multiplication and division are indicated by action or process symbol.
4. **Decision:** Diamond symbol represents a decision point. Decision based operations such as yes/no question or true/false are indicated by diamond in flowchart.
5. **Connectors:** Whenever flowchart becomes complex or it spreads over more than one page, it is useful to use connectors to avoid any confusions. It is represented by a circle.
6. **Flow lines:** Flow lines indicate the exact sequence in which instructions are executed. Arrows represent the direction of flow of control and relationship among different symbols of flowchart.



3.2. Block Diagram:

A block diagram is a visual representation of a system that uses simple, labeled blocks that represent single or multiple items, entities or concepts, connected by lines to show relationships between them. An entity relationship diagram (ERD), one example of a block diagram, represents an information system by showing the relationships between people, objects, places, concepts or events within that system.

Block diagrams are used heavily in engineering and design of diagrams for electronics, hardware, software and processes. Most commonly, they represent concepts and systems in a higher level, less detailed overview. The diagrams are useful for troubleshooting technical issues.

Block diagrams are a generalized representation of a concept and are not intended to display complete information in regards to design or manufacture. Unlike schematics, blueprints and layout diagrams, block diagrams do not portray the necessary detail for physical construction. Block diagrams are made simple so as not to cloud concepts.

The simplification in block diagrams can also be useful when demonstrating an idea, but concealing the inner workings of potentially secret intellectual property (IP). Top-down design in electrical engineering often progresses through increasingly detailed block diagrams. After enough detail is added through iterations, the block diagram becomes a schematic. Block diagrams in process control show the functions of operations but not the components that perform them. The functions of block diagrams may then be implemented with programmable logic controllers (PLC).

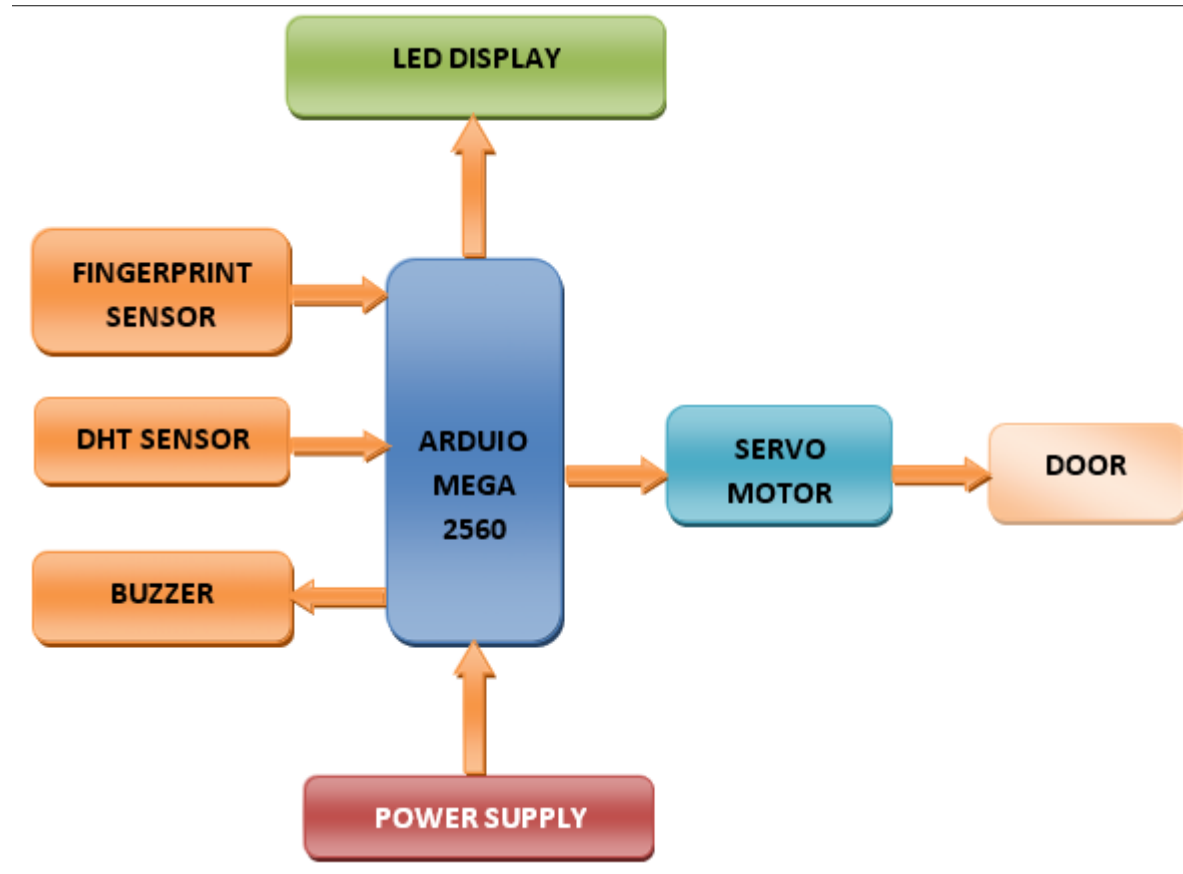
Basic Components of Block Diagram

Drawing a system block diagram is relatively easy. You just have to be familiar with the components of what makes up this diagram. It only consists of basic geometric shapes and symbols. Below are their functions and uses.

Box- represents the major part or function in a system. Every block in a system has only one input and output.

Line- this symbol connects the boxes in a system to illustrate relationships.

Arrow Line- this figure is specifically used to indicate the flow of the signal or data through the electrical block diagram and software design.



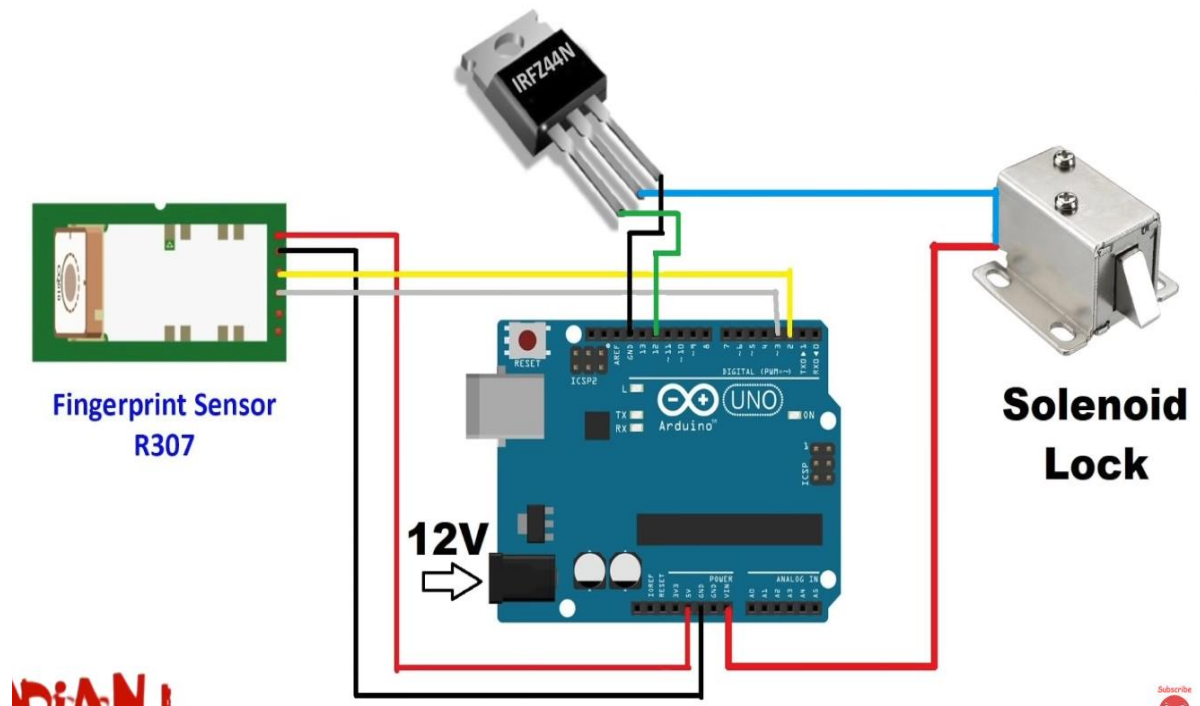
3.3. Circuit Diagram

The circuit diagram for the Fingerprint locker is mentioned below.

The gate and the source pins on MOSFET are connected to the Arduino Uno on GND and digital pin 12 respectively. The drain pin is connected to the solenoid lock.

The Tx and Rx pins on the fingerprint sensor are connected to the digital pins 2 and 3 respectively. 5V is connected to 5V and GND to GND.

The positive terminal on the solenoid lock is connected to the Vin on the Arduino Uno. The circuit of this Arduino Fingerprint Security System is very simple which contains Arduino Mega2560 which controls whole the process of the project, push button, buzzer, and LCD. Arduino Mega2560 controls the complete processes. The push button is directly connected to pin A9 (ENROL), A10 (OK/DEL), A11 (UP), A12 (DOWN) and A8 (CLOSED) of Arduino Mega2560 with respect to ground and Red LED is connected at Digital pin D4 of Arduino Mega2560 with respect to ground through a 10 ohms resistor and Green LED is connected to D3 of Arduino Mega2560 with the same method. Finger Print Module's Rx and Tx directly connected at Software Serial or Digital pin D11 and D10 of Arduino Mega2560. 5v supply is used for powering finger print module taken from Arduino Mega pin and Servo motor is also connected to PWM pin D5 of Arduino mega2560. A 16×2 LCD is configured in 4-bit mode and its RS, EN, D4, D5, D6, and D7 are directly connected at Digital pin D13, D12, D6, D7, D9, and D8 of Arduino Mega2560. Buzzer is connected at the Digital pin D14 of Arduino Mega2560 and with respect to the ground. DHT is connected at the Digital pin D2 of Arduino Mega2560 and with respect to the ground and Vcc. Potentiometer of middle pin is connected LCD (Vo). Firstly, "Enroll" button is pressed to enroll the finger print of the authenticate user. Finger print is stored by pressing "OK/Del" button. Gate is closed by pressing the "Close" button of the module. When any user tries to open the gate, module checks the authenticity of the user by comparing his/her fingerprint with the database, if the users fingerprint matches with the one that is stored in the database then arduino sends the signal to run the motor which then opens the gate



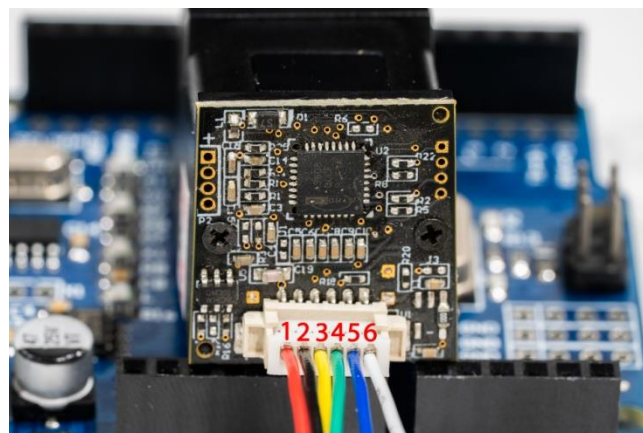
Chapter 4 Working

Working principle of the Fingerprint sensor

- Generally, the fingerprint scanner works in two ways. First is the optical one, another is the capacitive one. The optical fingerprint has a light inside it. This light will be flashed to the finger and it will capture the picture digitally.
- Then it converts this fingerprint pattern into a source code of 0 and 1. This code is unique as each fingerprint is unique. And this code can be used to identify the fingerprint. If the fingerprint source code matches the authorized one, then and only then access is granted by the sensor to the system.
- The fingerprint sensor which we are going to use in this project is an optical sensor. R307 optical fingerprint scanner.
- As we now know, the fingerprint scanner works in two processes. Fingerprint enrollment and matching. Similarly, it also works in the same way in this project.
- At first, we have to store the fingerprint in the sensor via fingerprint enrollment. Once the fingerprint is stored, we can start with fingerprint matching.
- Whenever the enrolled print matches the current fingerprint, the scanner sends those signals to the Arduino Uno. The Arduino in turn sends the signal to the solenoid door lock to unlock if the fingerprint matches.
- An IRFZ44N MOSFET has been connected between the solenoid lock and the Arduino Uno. A MOSFET is a semiconductor device that is generally used to amplify electronic signals in devices.

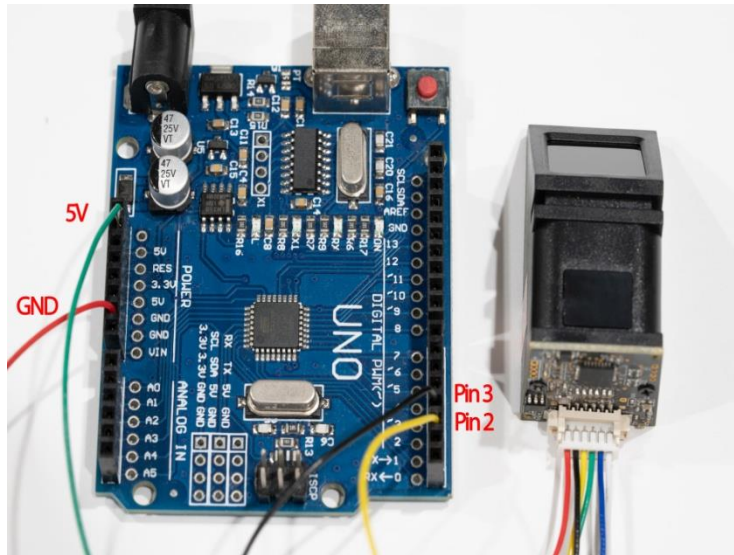
The fingerprint sensor has a TTL Serial interface, connecting only 2 data cables (TX, RX) and VCC, GND cables for a total of 4 wires.

1) Pin-out details of the fingerprint sensor



- 1 => GND (red)
- 2 => RX (Black)
- 3 => TX (Yellow)
- 4 => VCC 3.6-6VDC (Green)
- 5 => Finger detection signal (active high) (blue)
- 6 => Finger detection power supply 3.6-5VDC (white)

2) Connecting the Fingerprint Sensor (Fingerprint Sensor) to Arduino Uno R3



Arduino Uno R3 <---> Fingerprint Sensor

5V <---> VCC 3.6-6VDC (Green)

GND <---> GND (Red)

Pin 2 <---> TX (Yellow)

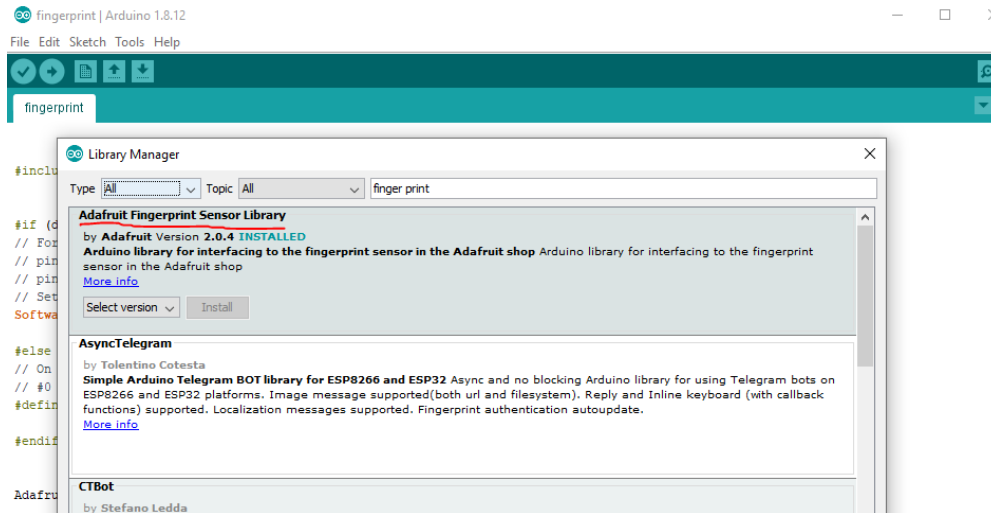
Pin 3 <---> RX (Black)

3) Installing libraries and testing programs

The first step is to install the Adafruit-Fingerprint-Sensor-Library Library . .

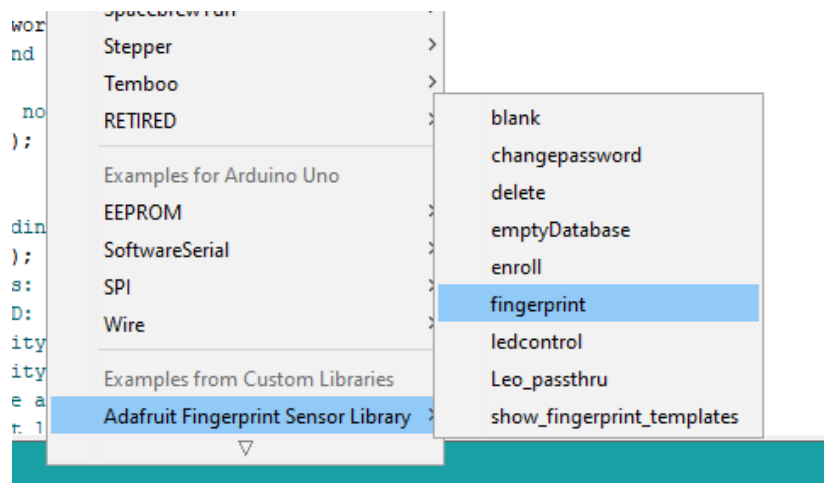
The recommended method is to install via Library Manager of Arduino IDE

by selecting from menu Sketch >> Include Library >> Manage Libraries, then search for Library named "Adafruit Fingerprint Sensor Library", then press Install to install.

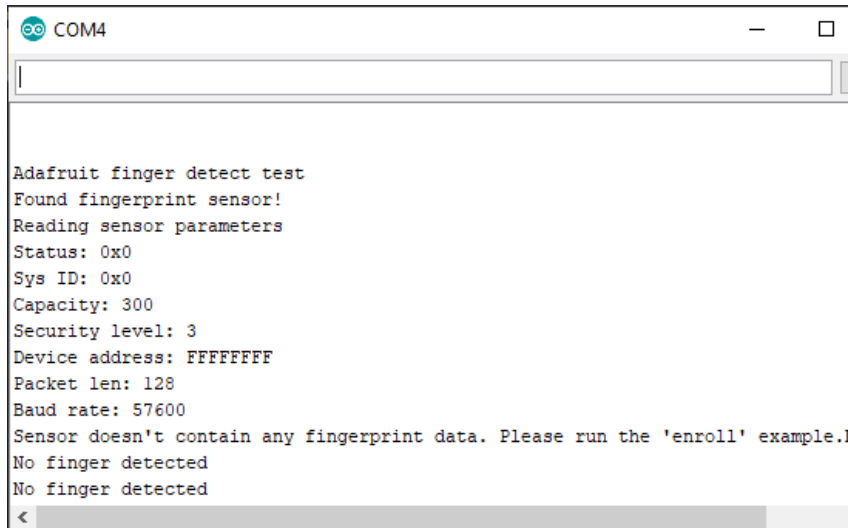


4) Upload the program and Test the functionality of the fingerprint sensor.

Open the sample code named fingerprint from menu File >> Examples >> Adafruit Fingerprint Sensor Library , then upload the code and open Serial Monitor, you will get the result as shown below. If you see a message saying " Did not find fingerprint sensor ":(



" Try to check the wiring connection. The most common problem is wrong connection of RX TX. Correct connection, RX, TX must be crossed, that is, RX of Arduino goes into TX of Sensor and RX of Sensor goes into TX of Arduino

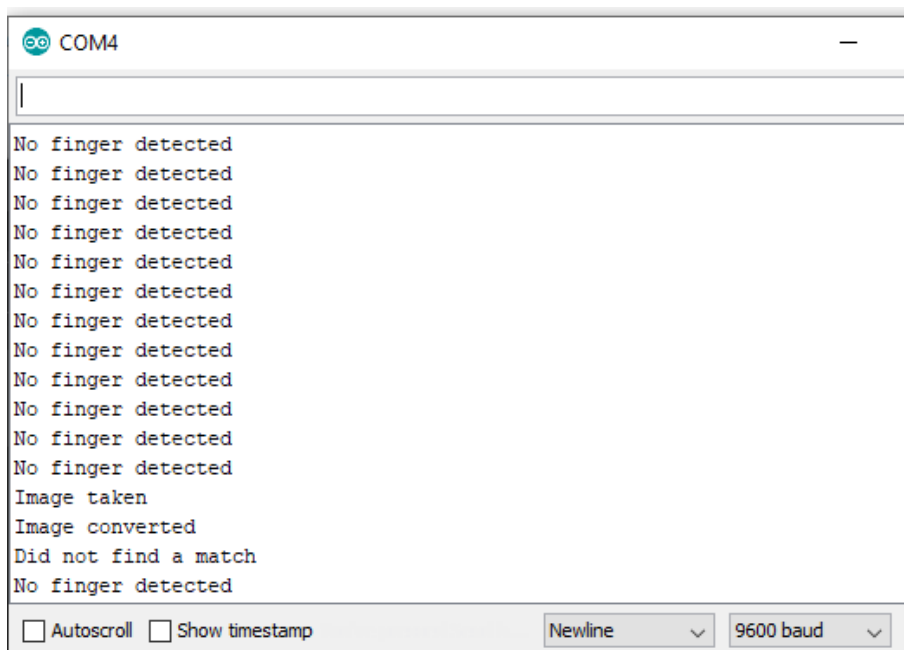


```
COM4

Adafruit finger detect test
Found fingerprint sensor!
Reading sensor parameters
Status: 0x0
Sys ID: 0x0
Capacity: 300
Security level: 3
Device address: FFFFFFFF
Packet len: 128
Baud rate: 57600
Sensor doesn't contain any fingerprint data. Please run the 'enroll' example.
No finger detected
No finger detected
<
```

as shown in the picture Indicates that the fingerprint sensor connection is correct.

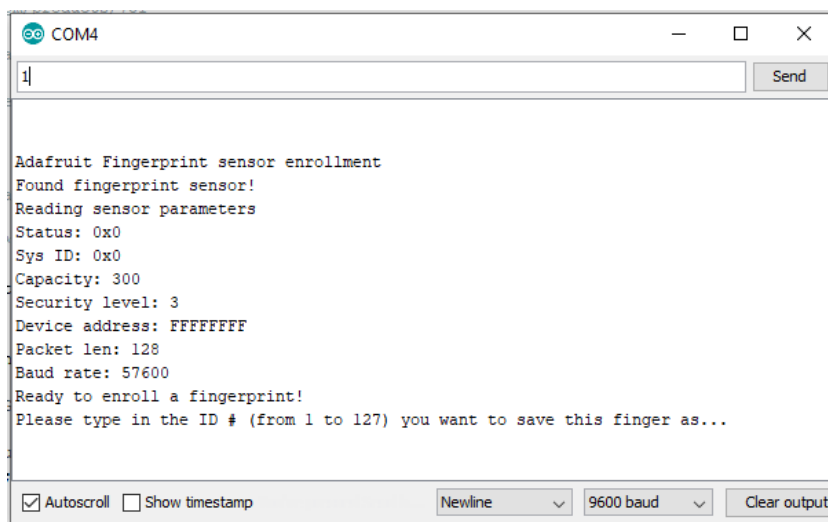
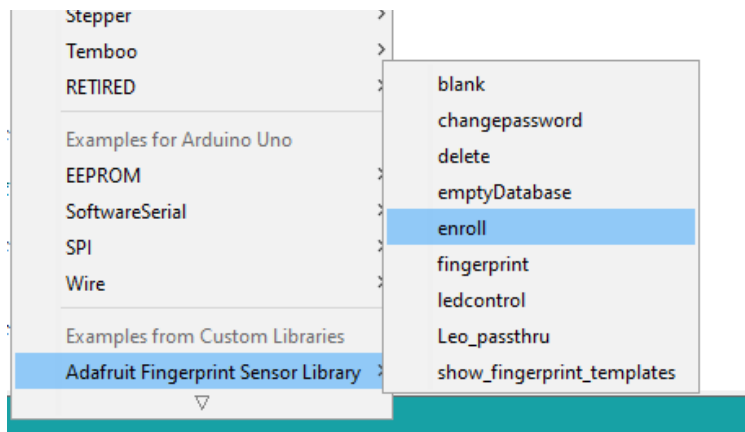
5) The fingerprint is saved on the sensor. Open the sample code named enroll from menu File >> Examples >> Adafruit Fingerprint Sensor Library , then upload the code and open Serial Monitor, you will get the result as shown below. This step determines the ID (numerical value 1 to 127) of the fingerprint to be recorded. Here, it will be assigned as the 1st fingerprint, type 1 and press the Send button.



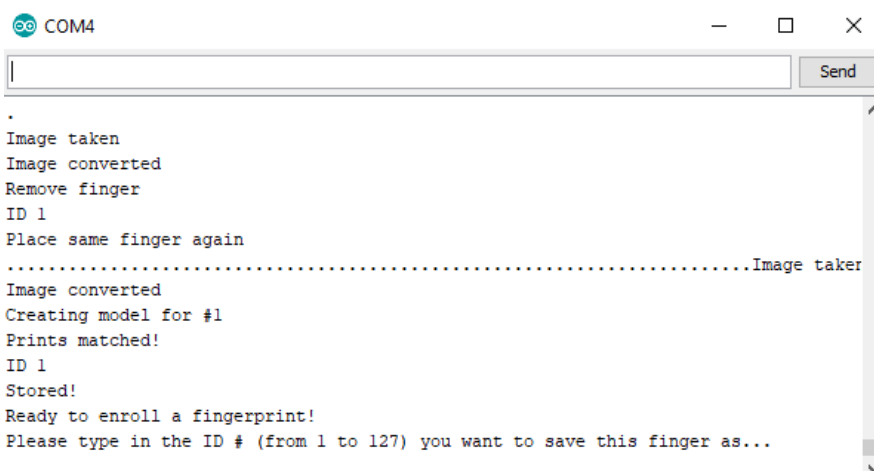
```
COM4

No finger detected
No finger detected
No finger detected
No finger detected
No finger detected
No finger detected
No finger detected
No finger detected
No finger detected
No finger detected
No finger detected
No finger detected
No finger detected
Image taken
Image converted
Did not find a match
No finger detected

☐ Autoscroll ☐ Show timestamp Newline 9600 baud
```

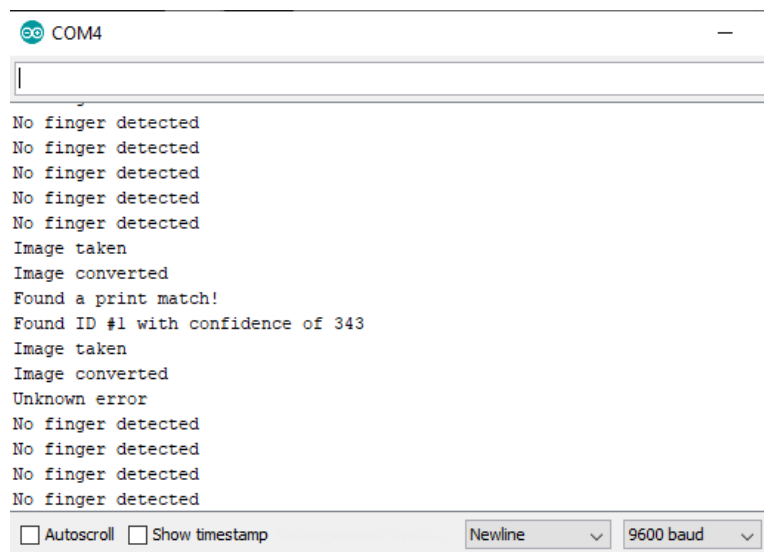


Then the program will show "Waiting for valid finger to enroll as #1", place the finger you want to register on the sensor twice, when the program says "Stored!", it will notify you that the fingerprint has been saved successfully . **Detect saved fingerprints** Open the fingerprint test program in step 4) again, then try to place the registered fingerprint on the sensor and observe the result in the Serial Monitor. The result is the fingerprint ID. Scannable hand, and confidence value of match is shown in the picture.



Found a print match!

Found ID #1 with confidence of 343



Chapter 5 Testing

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 28 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:

Participant ID	Identification Trial Number									
	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

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 29 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.

5.1 Unit Testing

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. It forms the basis for component testing. Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

5.1.2 Benefits

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. A unit test provides a strict, written contract that the piece of code must satisfy. As a result, it affords several benefits.

- 1. Find problems early:** Unit testing finds problems early in the development cycle. In test-driven development (TDD), which is frequently used in both extreme programming and scrum, unit tests are created before the code itself is written. When the tests pass, that code is considered complete. The same unit tests are run against that function frequently as the larger code base is developed either as the code is changed or via an automated process with the build. If the unit tests fail, it is a bug either in the changed code or the tests themselves. The unit tests then allow the location of the fault or failure to be easily traced. Since the unit tests alert the development team of the problem before

handing the code off to testers or clients, it is still early in the development process.

2. Facilitates Change: Unit testing allows the programmer to refactor code or upgrade system libraries later, and make sure the module still works correctly (e.g., in regression testing). The procedure is to write test cases for all functions and methods so that whenever a change causes a fault, it can be quickly identified. Unit tests detect changes which may break a design contract.

3. Simplifies Integration: Unit testing may reduce uncertainty in the units themselves and can be used in a bottom-up testing style approach. By testing the parts of a program first and then testing the sum of its parts, integration testing becomes much easier.

4. Documentation: Unit testing provides a sort of living documentation of the system. Developers looking to learn what functionality is provided by a unit, and how to use it, can look at the unit tests to gain a basic understanding of the unit's interface (API). Unit test cases embody characteristics that are critical to the success of the unit. These characteristics can indicate appropriate/inappropriate use of a unit as well as negative behaviors that are to be trapped by the unit. A unit test case, in and of itself, documents these critical characteristics, although many software development environments do not rely solely upon code to document the product in development.

5.2: INTEGRATION TESTING

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

5.2.1 Purpose

The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items. These "design items", i.e., assemblages (or groups of units), are exercised through their interfaces using black-box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test whether all the components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e., unit testing. The overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration

testing of further assemblages. Software integration testing is performed according to the software development life cycle (SDLC) after module and functional tests. The cross dependencies for software integration testing are: schedule for integration testing, strategy and selection of the tools used for integration, define the cyclomatic complexity of the software and software architecture, reusability of modules and life-cycle and versioning management. Some different types of integration testing are big-bang, top-down, and bottom-up, mixed (sandwich) and risky-hardest. Other Integration Patterns [2] are collaboration integration, backbone integration, layer integration, client-server integration, distributed services integration and high frequency integration.

5.2.1.1 Big Bang

In the big-bang approach, most of the developed modules are coupled together to form a complete software system or major part of the system and then used for integration testing. This method is very effective for saving time in the integration testing process. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of integration testing. A type of big-bang integration testing is called "usage model testing" which can be used in both software and hardware integration testing. The basis behind this type of integration testing is to run user-like workloads in integrated user-like environments. In doing the testing in this manner, the environment is proofed, while the individual components are proofed indirectly through their use. Usage Model testing takes an optimistic approach to testing because it expects to have few problems with the individual components. The strategy relies heavily on the component developers to do the isolated unit testing for their product. The goal of the strategy is to avoid redoing the testing done by the developers, and instead flesh-out problems caused by the interaction of the components in the environment. For integration testing, Usage Model testing can be more efficient and provides better test coverage than traditional focused functional integration testing. To be more efficient and accurate, care must be used in defining the user-like workloads for creating realistic scenarios in exercising the environment. This gives confidence that the integrated environment will work as expected for the target customers.

5.2.1.2 Top-down and Bottom-up

Bottom-up testing is an approach to integrated testing where the lowest level components are tested first, then used to facilitate the testing of higher-level components. The process is repeated until the component at the top of the hierarchy is tested. All the bottom or low-level modules, procedures or functions are integrated and then tested. After the integration testing of lower-level integrated modules, the next level of modules will be formed and can be used for integration testing. This approach is helpful only when all or most of the modules of the same development level are ready. This method also helps to determine the levels of software developed and makes it easier to report testing progress in the form of a percentage. Top-down testing is an approach to integrated testing where the top integrated modules are tested and the branch of the module is tested step by step until the

end of the related module. Sandwich testing is an approach to combine top down testing with bottom up testing.

5.3: SOFTWARE VERIFICATION AND VALIDATION

5.3.1 Introduction

In software project management, software testing, and software engineering, verification and validation (V&V) is the process of checking that a software system meets specifications and that it fulfills its intended purpose. It may also be referred to as software quality control. It is normally the responsibility of software testers as part of the software development lifecycle. Validation checks that the product design satisfies or fits the intended use (high-level checking), i.e., the software meets the user requirements. This is done through dynamic testing and other forms of review. Verification and validation are not the same thing, although they are often confused. Boehm succinctly expressed the difference between

- Validation: Are we building the right product?
- Verification: Are we building the product right?

According to the Capability Maturity Model (CMMI-SW v1.1)

Software Verification: The process of evaluating software to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.

Software Validation: The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements.

In other words, software verification is ensuring that the product has been built according to the requirements and design specifications, while software validation ensures that the product meets the user's needs, and that the specifications were correct in the first place. Software verification ensures that "you built it right". Software validation ensures that "you built the right thing". Software validation confirms that the product, as provided, will fulfill its intended use.

From Testing Perspective

- Fault – wrong or missing function in the code.
- Failure – the manifestation of a fault during execution.
- Malfunction – according to its specification the system does not meet its specified functionality

Both verification and validation are related to the concepts of quality and of software quality assurance. By themselves, verification and validation do not guarantee software quality; planning, traceability, configuration management and other aspects of software engineering are required. Within the modeling and simulation (M&S) community, the definitions of verification, validation and accreditation are similar:

- M&S Verification is the process of determining that a computer model, simulation, or federation of models and simulations implementations and their associated data accurately represent the developer's conceptual description and specifications.
- M&S Validation is the process of determining the degree to which a model, simulation, or federation of models and simulations, and their associated data are accurate representations of the real world from the perspective of the intended use(s).

5.3.2 Classification of Methods

In mission-critical software systems, where flawless performance is absolutely necessary, formal methods may be used to ensure the correct operation of a system. However, often for non- mission-critical software systems, formal methods prove to be very costly and an alternative method of software V&V must be sought out. In such cases, syntactic methods are often used.

5.3.3 Test Cases

A test case is a tool used in the process. Test cases may be prepared for software verification and software validation to determine if the product was built according to the requirements of the user. Other methods, such as reviews, may be used early in the life cycle to provide for software validation.

5.4: Black-Box Testing

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied virtually to every level of software testing: unit, integration, system and acceptance. It typically comprises most if not all higher-level testing but can also dominate unit testing as well.

5.4.1 Test Procedures

Specific knowledge of the application's code/internal structure and programming knowledge in general is not required. The tester is aware of what the software is supposed to do but is not aware of how it does it. For instance, the tester is aware that a particular input returns a certain, invariable output but is not aware of how the software produces the output in the first place.

5.4.2 Test Cases

Test cases are built around specifications and requirements, i.e., what the application is supposed to do. Test cases are generally derived from external descriptions of the software, including specifications, requirements and design parameters. Although the tests used are primarily functional in nature, non-functional tests may also be used. The test designer selects both valid and invalid inputs and determines the correct output, often with the help of an oracle or a previous result that is known to be good, without any knowledge of the test object's internal structure.

5.5: White-Box Testing

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT). White-box testing can be applied at the unit, integration and system levels of the software testing process. Although traditional testers tended to think of white-box testing as being done at the unit level, it is used for integration and system testing more frequently today. It can test paths within a unit, paths between units during integration, and between subsystems during a system-level test. Though this method of test design can uncover many errors or problems, it has the potential to miss unimplemented parts of the specification or missing requirements.

5.5.1. Levels

1) Unit testing: White-box testing is done during unit testing to ensure that the code is working as intended, before any integration happens with previously tested code. White box testing during unit testing catches any defects early on and aids in any defects that happen later on after the code is integrated with the rest of the application and therefore prevents any type of errors later on.

2) Integration testing: White-box testing at this level are written to test the interactions of each interface with each other. The Unit level testing made sure that each code was tested and working accordingly in an isolated environment and integration examines the correctness of the behavior in an open environment through the use of white-box testing for any interactions of interfaces that are known to the programmer.

3) Regression testing: White-box testing during regression testing is the use of recycled white-box test cases at the unit and integration testing levels.

5.5.2 Procedures

White-box testing's basic procedures involves the tester having a deep level of understanding of the source code being tested. The programmer must have a deep understanding of the application to know what kinds of test cases to create so that every visible path is exercised for testing. Once the source code is understood then the source code can be analyzed for test cases to be created.

These are the three basic steps that white-box testing takes in order to create test cases:

- Input involves different types of requirements, functional specifications, detailed designing of documents, proper source code, security specifications. This is the preparation stage of white box testing to layout all of the basic information.
- Processing involves performing risk analysis to guide whole testing process, proper test plan, execute test cases and communicate results. This is the phase of building test cases to make sure they thoroughly test the application the given results are recorded accordingly.
- Output involves preparing final report that encompasses all of the above preparations and results.

5.5.3 Advantages

White-box testing is one of the two biggest testing methodologies used today. It has several major advantages:

- A side effect of having the knowledge of the source code is beneficial to thorough testing.
- Optimization of code by revealing hidden errors and being able to remove these possible defects.
- Gives the programmer introspection because developers carefully describe any new implementation.
- Provides traceability of tests from the source, allowing future changes to the software to be easily captured in changes to the tests.
- White box testing give clear, engineering-based, rules for when to stop testing.

5.5.5 Disadvantages

Although white-box testing has great advantages, it is not perfect and contains some disadvantages:

- White-box testing brings complexity to testing because the tester must have knowledge of the program, including being a programmer. White-box testing

requires a programmer with a high level of knowledge due to the complexity of the level of testing that needs to be done.

- On some occasions, it is not realistic to be able to test every single existing condition of the application and some conditions will be untested.
- The tests focus on the software as it exists, and missing functionality may not be discovered.

5.6: SYSTEM TESTING

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic. As a rule, system testing takes, as its input, all of the "integrated" software components that have passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing tests not only the design, but also the behavior and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification.

Chapter 6 Design Issues

- **Economic:** The goal of this project was to have the overall cost of the system below Rs.3000. 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.
- **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.
- **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.
- **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. 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.
- **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.

Chapter 7 Design Requirements

- **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.
- **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.
- **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!
- **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.

- **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.
- **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.

Chapter 8 Advantages & Disadvantages

ADVANTAGES

- This project provides security.
- Power consumption is less.
- Used commonly available component.
- Circuit diagram is simple and easy.
- Easy to use and setup.
- Storage of up to 200 fingerprints.
- Generally it is used in ATM, fingerprint car and home door lock etc for security.

DISADVANTAGES

- Different biometric technologies need the use of different devices that have a range of cost.
- Entry and delete fingerprints need to operate multiple steps, the program is too much trouble, convenience is not enough.
- Performance can be fluctuating to dry, wet, dirty fingers.
- Population coverage may be a problem with old age people or people with skin disease.

Chapter 9 Application

- Used in Banks and Offices to secure the vaults door or simply for residential houses door lock system.
- Fingerprint security system can be used in ATM, fingerprint operated Vehicles.
- Can be used for voter ID registration.

Chapter 10 Future Improvement

Advancements in biometric identification management technology are moving so fast, In future we will make advancement and multi functions like sms alert if authorized person try to lock the door. Image recognizing process system and password system based. Also eyes retina for password which helps authorized persons for authentication for entrance so biometric technology makes individual convenient in real life.

Chapter 11 Problem Formulation

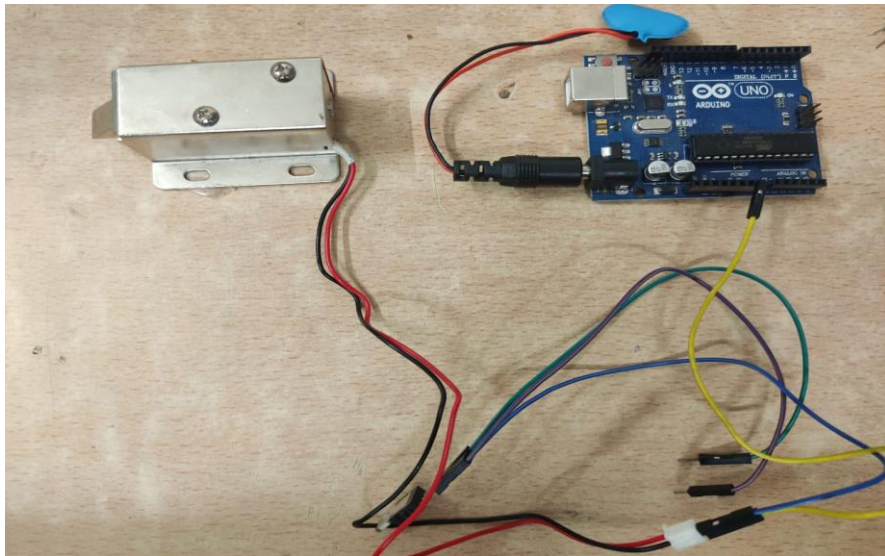
- The fingerprint will not work when it is dirty or wait finger.
- “Module not detected” problem may occur which can be easily solved by resetting using reset button.

Chapter 12 Conclusion

Fingerprint door locks are great investment for home or business. It provides great security by providing restrictions to unwanted access. This device increases level of security by adding unique biological features of authorized person. For anyone who wants more security to their homes, fingerprint door locks are best choice.

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.

Chapter 13 Outputs



Chapter 14 Coding

```
#include "dht.h"
#define dht_apin 2 // Analog Pin sensor is connected to
dht DHT;
#include<LiquidCrystal.h>
LiquidCrystal lcd(13,12,6,7,9,8);
#include <SoftwareSerial.h>
SoftwareSerial fingerPrint(10, 11);
#include<Servo.h>
Servo myServo;

#include <Adafruit_Fingerprint.h>
uint8_t id;
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&fingerPrint);
#define rabin A8
#define enroll A0
#define del A1
#define up A2
#define down A3
#define openLight 3
#define closeLight 4
#define servoPin 5

void setup()
{
    delay(1000);
    myServo.attach(servoPin);
    myServo.write(180);
    pinMode(enroll, INPUT_PULLUP);
    pinMode(up, INPUT_PULLUP);
    pinMode(down, INPUT_PULLUP);
    pinMode(del, INPUT_PULLUP);
    pinMode(rabin, INPUT_PULLUP);
    pinMode(openLight, OUTPUT);
    pinMode(closeLight, OUTPUT);
    lcd.begin(16,2);
    DHT.read11(dht_apin);

    lcd.setCursor(0,0);
    lcd.print("  MANMOHAN ");
    lcd.setCursor(0,1);
    lcd.print("POLYTECHNIC COLLAGE");
    lcd.scrollDisplayLeft();
    delay(1000);
    lcd.scrollDisplayLeft();
    delay(1000);
```

```

    lcd.scrollDisplayLeft();
    delay(1000);
    lcd.scrollDisplayLeft();
    delay(5000);
    lcd.clear();
    finger.begin(57600);
    Serial.begin(9600);
    lcd.clear();
    lcd.print(" Finding Module ");
    lcd.setCursor(0,1);
    delay(2000);
    if (finger.verifyPassword())
    {
        Serial.println("Found fingerprint sensor!");
        lcd.clear();
        lcd.print(" Module Found ");
        delay(2000);
    }
    else
    {
        Serial.println("Did not find fingerprint sensor :(");
        lcd.clear();
        lcd.print("Module Not Found");
        lcd.setCursor(0,1);
        lcd.print("Check Connections");
        while (1);
    }
}

void loop()
{
    lcd.setCursor(0,0);
    lcd.print("HUMI  =");
    lcd.print(DHT.humidity);
    lcd.print("% ");
    lcd.setCursor(0,1);
    lcd.print("TEMP  =");
    lcd.print( DHT.temperature);
    lcd.println("C ");
    delay(5000);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print(" Press Match ");
    lcd.setCursor(0,1);
    lcd.print("To Start System");
    digitalWrite(closeLight, HIGH);
    if(digitalRead(up)==0 || digitalRead(down)==0)
    {
        for(int i=0;i<5;i++) { lcd.clear(); lcd.print("Place Finger"); delay(2000); int
        result=getFingerprintIDez(); if(result>=0)

```

```

    {
        digitalWrite(openLight, HIGH);
        digitalWrite(closeLight, LOW);
        lcd.clear();
        lcd.print("  Allowed  ");
        lcd.setCursor(0,1);
        lcd.print(" Gate Opened  ");
        myServo.write(0);
        delay(5000);
        myServo.write(180);
        digitalWrite(closeLight, HIGH);
        digitalWrite(openLight, LOW);
        lcd.setCursor(0,1);
        lcd.print(" Gate Closed  ");
        return;
    }
}
}
checkKeys();
delay(2000);
if( currentposition==0 && digitalRead(rabin) == 0)
{
    displayscreen();

}
int l ;
char code=keypad.getKey();
if(code!=NO_KEY)
{
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("PASSWORD:");
    lcd.setCursor(7,1);
    lcd.print(" ");
    lcd.setCursor(7,1);
    for(l=0;l<=currentposition;++l)
    {

        lcd.print("*");
        keypress();
    }

    if (code==password[currentposition])
    {
        ++currentposition;
        if(currentposition==4)
        {

            unlockdoor();
            currentposition=0;

```

```

    }

    }

    else
    {
        ++invalidcount;
        incorrect();
        currentposition=0;

    }
    if(invalidcount==5)
    {

        ++invalidcount;
        torture1();

    }
    if(invalidcount==8)
    {
        torture2();
    }
    }

    void checkKeys()
    {
        if(digitalRead(enroll) == 0)
        {
            lcd.clear();
            lcd.print("Please Wait");
            delay(2000);
            while(digitalRead(enroll) == 0);
            Enroll();
        }

        else if(digitalRead(del) == 0)
        {
            lcd.clear();
            lcd.print("Please Wait");
            delay(2000);
            delet();
        }
    }

    void Enroll()
    {
        int count=0;
        lcd.clear();
        lcd.print("Enroll Finger  ");
    }

```

```

lcd.setCursor(0,1);
lcd.print("Location:");
while(1)
{
  lcd.setCursor(9,1);
  lcd.print(count);
  if(digitalRead(up) == 0)
  {
    count++;
    if(count>25)
    count=0;
    delay(500);
  }

  else if(digitalRead(down) == 0)
  {
    count--;
    if(count<0) count=25; delay(500); } else if(digitalRead(del) == 0) { id=count;
getFingerprintEnroll(); return; } else if(digitalRead(enroll) == 0) { return; } } } void delet() {
int count=0; lcd.clear(); lcd.print("Delete Finger "); lcd.setCursor(0,1); lcd.print("Location:");
while(1) { lcd.setCursor(9,1); lcd.print(count); if(digitalRead(up) == 0) { count++;
if(count>25)
  count=0;
  delay(500);
}

  else if(digitalRead(down) == 0)
  {
    count--;
    if(count<0)
    count=25;
    delay(500);
  }
  else if(digitalRead(del) == 0)
  {
    id=count;
    deleteFingerprint(id);
    return;
  }

  else if(digitalRead(enroll) == 0)
  {
    return;
  }
}
}

uint8_t getFingerprintEnroll()
{
  int p = -1;

```

```

lcd.clear();
lcd.print("Finger ID:");
lcd.print(id);
lcd.setCursor(0,1);
lcd.print("Place Finger");
delay(2000);
while (p != FINGERPRINT_OK)
{
  p = finger.getImage();
  switch (p)
  {
    case FINGERPRINT_OK:
      Serial.println("Image taken");
      lcd.clear();
      lcd.print("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      Serial.println("No Finger");
      lcd.clear();
      lcd.print("No Finger");
      break;
    case FINGERPRINT_PACKETRECEIVEERR:
      Serial.println("Communication error");
      lcd.clear();
      lcd.print("Comm Error");
      break;
    case FINGERPRINT_IMAGEFAIL:
      Serial.println("Imaging error");
      lcd.clear();
      lcd.print("Imaging Error");
      break;
    default:
      Serial.println("Unknown error");
      lcd.clear();
      lcd.print("Unknown Error");
      break;
  }
}

// OK success!

p = finger.image2Tz(1);
switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image converted");
    lcd.clear();
    lcd.print("Image converted");
    break;
  case FINGERPRINT_IMAGEMESS:
    Serial.println("Image too messy");

```

```

    lcd.clear();
    lcd.print("Image too messy");
    return p;
case FINGERPRINT_PACKETRECEIVEERR:
    Serial.println("Communication error");
    lcd.clear();
    lcd.print("Comm Error");
    return p;
case FINGERPRINT_FEATUREFAIL:
    Serial.println("Could not find fingerprint features");
    lcd.clear();
    lcd.print("Feature Not Found");
    return p;
case FINGERPRINT_INVALIDIMAGE:
    Serial.println("Could not find fingerprint features");
    lcd.clear();
    lcd.print("Feature Not Found");
    return p;
default:
    Serial.println("Unknown error");
    lcd.clear();
    lcd.print("Unknown Error");
    return p;
}

Serial.println("Remove finger");
lcd.clear();
lcd.print(" Remove Finger ");
delay(2000);
p = 0;
while (p != FINGERPRINT_NOFINGER) {
    p = finger.getImage();
}
Serial.print("ID "); Serial.println(id);
p = -1;
Serial.println("Place same finger again");
lcd.clear();
    lcd.print(" Place Finger ");
    lcd.setCursor(0,1);
    lcd.print(" Again ");
while (p != FINGERPRINT_OK) {
    p = finger.getImage();
    switch (p) {
    case FINGERPRINT_OK:
        Serial.println("Image taken");
        break;
    case FINGERPRINT_NOFINGER:
        Serial.print(".");
        break;
    case FINGERPRINT_PACKETRECEIVEERR:

```



```

        Serial.println("Communication error");
        break;
    case FINGERPRINT_IMAGEFAIL:
        Serial.println("Imaging error");
        break;
    default:
        Serial.println("Unknown error");
        break;
    }
}

// OK success!

p = finger.image2Tz(2);
switch (p) {
    case FINGERPRINT_OK:
        Serial.println("Image converted");
        break;
    case FINGERPRINT_IMAGEMESS:
        Serial.println("Image too messy");
        return p;
    case FINGERPRINT_PACKETRECIEVEERR:
        Serial.println("Communication error");
        return p;
    case FINGERPRINT_FEATUREFAIL:
        Serial.println("Could not find fingerprint features");
        return p;
    case FINGERPRINT_INVALIDIMAGE:
        Serial.println("Could not find fingerprint features");
        return p;
    default:
        Serial.println("Unknown error");
        return p;
}

// OK converted!
Serial.print("Creating model for #"); Serial.println(id);

p = finger.createModel();
if (p == FINGERPRINT_OK) {
    Serial.println("Prints matched!");
} else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    Serial.println("Communication error");
    return p;
} else if (p == FINGERPRINT_ENROLLMISMATCH) {
    Serial.println("Fingerprints did not match");
    return p;
} else {
    Serial.println("Unknown error");
    return p;
}

```

```

    }

    Serial.print("ID "); Serial.println(id);
    p = finger.storeModel(id);
    if (p == FINGERPRINT_OK) {
        Serial.println("Stored!");
        lcd.clear();
        lcd.print("Stored!");
        delay(2000);
    } else if (p == FINGERPRINT_PACKETRECEIVEERR) {
        Serial.println("Communication error");
        return p;
    } else if (p == FINGERPRINT_BADLOCATION) {
        Serial.println("Could not store in that location");
        return p;
    } else if (p == FINGERPRINT_FLASHERR) {
        Serial.println("Error writing to flash");
        return p;
    }
    else {
        Serial.println("Unknown error");
        return p;
    }
}

int getFingerprintIDez()
{
    uint8_t p = finger.getImage();

    if (p != FINGERPRINT_OK)
        return -1;

    p = finger.image2Tz();
    if (p != FINGERPRINT_OK)
        return -1;

    p = finger.fingerFastSearch();
    if (p != FINGERPRINT_OK)
    {
        lcd.clear();
        lcd.print("Finger Not Found");
        lcd.setCursor(0,1);
        lcd.print("Try Later");
        delay(2000);
        return -1;
    }
    // found a match!
    Serial.print("Found ID #");
    Serial.print(finger.fingerID);
    return finger.fingerID;
}

```

```

}

uint8_t deleteFingerprint(uint8_t id)
{
    uint8_t p = -1;
    lcd.clear();
    lcd.print("Please wait");
    p = finger.deleteModel(id);
    if (p == FINGERPRINT_OK)
    {
        Serial.println("Deleted!");
        lcd.clear();
        lcd.print("Figer Deleted");
        lcd.setCursor(0,1);
        lcd.print("Successfully");
        delay(2000);
    }

    else
    {
        Serial.print("Something Wrong");
        lcd.clear();
        lcd.print("Something Wrong");
        lcd.setCursor(0,1);
        lcd.print("Try Again Later");
        delay(2000);
        return p;
    }
}

void unlockdoor()
{
    delay(900);

    lcd.setCursor(0,0);
    lcd.println(" ");
    lcd.setCursor(1,0);
    lcd.print("Access Granted");
    lcd.setCursor(4,1);
    lcd.println("WELCOME!!");
    lcd.setCursor(15,1);
    lcd.println(" ");
    lcd.setCursor(16,1);
    lcd.println(" ");
    lcd.setCursor(14,1);
    lcd.println(" ");
    lcd.setCursor(13,1);
    lcd.println(" ");
    unlockbuzz();

    for(pos = 180; pos>=0; pos-=5) // goes from 180 degrees to 0 degrees

```

```

{
myservo.write(pos); // tell servo to go to position in variable 'pos'
delay(5); // waits 15ms for the servo to reach the position
}
delay(2000);

delay(1000);
counterbeep();

delay(1000);

for(pos = 0; pos <= 180; pos +=5) // goes from 0 degrees to 180 degrees
{ // in steps of 1 degree
myservo.write(pos); // tell servo to go to position in variable 'pos'
delay(15);

int currentposition=0;

lcd.clear();
displayscreen();

}
}

//*****WRONG CODE
FUNCTION*****//

void incorrect()
{
delay(500);
lcd.clear();
lcd.setCursor(1,0);
lcd.print("CODE");
lcd.setCursor(6,0);
lcd.print("INCORRECT");
lcd.setCursor(15,1);
lcd.println(" ");
lcd.setCursor(4,1);
lcd.println("GET AWAY!!!");

lcd.setCursor(13,1);
lcd.println(" ");
Serial.println("CODE INCORRECT YOU ARE UNAUTHORIZED");
digitalWrite(redled, HIGH);
digitalWrite(buzz, HIGH);
delay(3000);
lcd.clear();

```

```

digitalWrite(redled, LOW);
digitalWrite(buzz,LOW);
displayScreen();
}
//***** CLEAR THE
SCREEN!!!*****//
void clearScreen()
{
  lcd.setCursor(0,0);
  lcd.println(" ");
  lcd.setCursor(0,1);
  lcd.println(" ");
  lcd.setCursor(0,2);
  lcd.println(" ");
  lcd.setCursor(0,3);
  lcd.println(" ");
}
//*****KEYPRESS*****
*//
void keypress()
{

digitalWrite(buzz, HIGH);
delay(50);
digitalWrite(buzz, LOW);
}
//*****DISPALAY
FUNCTION!!!*****//
void displayScreen()
{

  lcd.setCursor(0,0);
  lcd.println("*ENTER THE CODE*");
  lcd.setCursor(1,1);

  lcd.println("TO _/ _ (OPEN)!!");
}
//***** ARM
SERVO*****
*****//
void armServo()
{

  for (pos=180;pos<=180;pos+=50)
  {
    myservo.write(pos);
    delay(5);
  }
}

```

```

delay(5000);

for(pos=180;pos>=0;pos-=50)
{
myservo.write(pos);
}

}
//*****UNLOCK
BUZZ*****//
void unlockbuzz()
{

digitalWrite(buzz, HIGH);
delay(80);
digitalWrite(buzz, LOW);
delay(80);
digitalWrite(buzz, HIGH);
delay(80);
digitalWrite(buzz, LOW);
delay(200);
digitalWrite(buzz, HIGH);
delay(80);
digitalWrite(buzz, LOW);
delay(80);
digitalWrite(buzz, HIGH);
delay(80);
digitalWrite(buzz, LOW);
delay(80);
}

//*****COUNTER
BEEP*****//
void counterbeep()
{
delay(1200);

lcd.clear();
digitalWrite(buzz, HIGH);

lcd.setCursor(2,15);
lcd.println(" ");
lcd.setCursor(2,14);
lcd.println(" ");
lcd.setCursor(2,0);
delay(200);
lcd.println("GET IN WITHIN::");

lcd.setCursor(4,1);

```

```

lcd.print("5");
delay(200);
lcd.clear();
lcd.setCursor(2,0);
lcd.println("GET IN WITHIN:");
digitalWrite(buzz,LOW);
delay(1000);
//2
digitalWrite(buzz, HIGH);
lcd.setCursor(2,0);
lcd.println("GET IN WITHIN:");
lcd.setCursor(4,1); //2
lcd.print("4");
delay(100);
lcd.clear();
lcd.setCursor(2,0);
lcd.println("GET IN WITHIN:");
digitalWrite(buzz,LOW);
delay(1000);
//3
digitalWrite(buzz, HIGH);
lcd.setCursor(2,0);
lcd.println("GET IN WITHIN:");
lcd.setCursor(4,1); //3
lcd.print("3");
delay(100);
lcd.clear();
lcd.setCursor(2,0);
lcd.println("GET IN WITHIN:");
digitalWrite(buzz,LOW);
delay(1000);
//4
digitalWrite(buzz, HIGH);
lcd.setCursor(2,0);
lcd.println("GET IN WITHIN:");
lcd.setCursor(4,1); //4
lcd.print("2");
delay(100);
lcd.clear();
lcd.setCursor(2,0);
lcd.println("GET IN WITHIN:");
digitalWrite(buzz,LOW);
delay(1000);
//
digitalWrite(buzz, HIGH);
lcd.setCursor(4,1);
lcd.print("1");
delay(100);
lcd.clear();
lcd.setCursor(2,0);

```

```

lcd.println("GET IN WITHIN::");
digitalWrite(buzz,LOW);
delay(1000);
//5
digitalWrite(buzz, HIGH);
delay(40);
digitalWrite(buzz,LOW);
delay(40);
digitalWrite(buzz, HIGH);
delay(40);
digitalWrite(buzz,LOW);
delay(40);
digitalWrite(buzz, HIGH);
delay(40);
digitalWrite(buzz,LOW);
delay(40);
digitalWrite(buzz, HIGH);
delay(40);
digitalWrite(buzz,LOW);
lcd.clear();
lcd.setCursor(2,0);
lcd.print("RE-LOCKING");
delay(500);
lcd.setCursor(12,0);
lcd.print(".");
delay(500);
lcd.setCursor(13,0);
lcd.print(".");
delay(500);
lcd.setCursor(14,0);
lcd.print(".");
delay(400);
lcd.clear();
lcd.setCursor(4,0);
lcd.print("LOCKED!");
delay(440);
}
//*****TORTURE1*****
*****/
void torture1()
{
delay(1000);
lcd.clear();
lcd.setCursor(2,0);
lcd.print("WAIT FOR ");
lcd.setCursor(5,1);
lcd.print("15 SECONDS");
digitalWrite(buzz, HIGH);
delay(15000);
digitalWrite(buzz, LOW);

```



```

lcd.clear();
lcd.setCursor(2,0);
lcd.print("LOL..");
lcd.setCursor(1,1);
lcd.print(" HOW WAS THAT??");
delay(3500);
lcd.clear();

}
//*****TORTURE2*****
*****//
void torture2()
{
delay(1000);
lcd.setCursor(1,0);
lcd.print(" ");
lcd.setCursor(2,0);
lcd.print("EAR DRUMS ARE");
lcd.setCursor(0,1);
lcd.print(" PRECIOUS!! ");
delay(1500);
lcd.clear();
lcd.setCursor(1,0);
lcd.print(" WAIT FOR");
lcd.setCursor(4,1);
lcd.print(" 1 MINUTE");
digitalWrite(buzz, HIGH);
delay(55000);
counterbeep();
lcd.clear();
digitalWrite(buzz, LOW);
lcd.setCursor(2,0);
lcd.print("WANT ME TO");
lcd.setCursor(1,1);
lcd.print("REDICULE MORE??");
delay(2500);
lcd.clear();
lcd.setCursor(2,0);
lcd.print("Ha Ha Ha Ha");
delay(1700);
lcd.clear();
}

```

Chapter 15 Appendix 1: Work 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 the 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 use 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 "software serial" 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

61 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.

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, spark fun 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=GoogleBase62&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.eleccircuit.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.

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 63 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> array 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 7Eabcefg hijklmnopqrstuvwxyz 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 squagre) 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! 64

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 master mode. 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 mater mode 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 65 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?

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 "Indian" nature and that the data might essentially be back words, 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

66 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 cmd_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 67 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).

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 push buttons. If its high enroll a user, if a different one was high identify. At first I had some trouble with what was returned from digital Read 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 68 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 of black squares on the 1st line. This means according to someone [here](http://www.avrfreaks.net/index.php?name=PNphpBB2&file=printview&t=25912&start=0) <http://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 spark fun 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 hyper terminal 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 hyper terminal 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 be best. Its also easier to debug using hyper terminal 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 master mode 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 69 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.

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: didn't 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 71 ENROLL 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 reenter. 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 it's 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 it fits 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 72 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.

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 fact for the, 2. Change the chek sum and in sum statements to reflect the new variable H 3. Change the ID_2 if it is a success to have H, don't 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 73 Working through the testing, also noticed that there is a < on the First Draft 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.

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 won't 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 74 positive and negative leads to power the MOSFET wrong, woops. The problem was easily 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 re sit 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.

Chapter 16 Bibliography

- <https://pypi.org/project/keras/>
- https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml
- <https://numpy.org/>
- <https://www.wisegeek.com/what-is-emotionalintelligence.htm>