Contents

[Abstract 2](#_Toc499388325)

[Intro and Theory 2](#_Toc499388326)

[Intro 2](#_Toc499388327)

[Theory 2](#_Toc499388328)

[High level Design & Software and Hardware design (Method) 3](#_Toc499388329)

[Overall Schematic 3](#_Toc499388330)

[General Hardware 5](#_Toc499388331)

[Keypad 5](#_Toc499388332)

[Buzzer 5](#_Toc499388333)

[LCD 5](#_Toc499388334)

[Pins 6](#_Toc499388335)

[USART1 6](#_Toc499388336)

[Fingerprint Scanner (Hardware connections) 6](#_Toc499388337)

[Transmission 7](#_Toc499388338)

[Reception 7](#_Toc499388339)

[Software 7](#_Toc499388340)

[Initialization 7](#_Toc499388341)

[Actions 7](#_Toc499388342)

[Databases 7](#_Toc499388343)

[Sending Subroutines 8](#_Toc499388344)

[Mess1out 8](#_Toc499388345)

[UART\_send\_byte 9](#_Toc499388346)

[save\_data\_start352 9](#_Toc499388347)

[Compare\_basic2 10](#_Toc499388348)

[Testing the Software 10](#_Toc499388349)

[Results and Performance (results/ discussion) 11](#_Toc499388350)

[Extensions 12](#_Toc499388351)

[Technical detail 12](#_Toc499388352)

[Investigating reliability 12](#_Toc499388353)

[Errors 12](#_Toc499388354)

[Updates, Modifications and improvements 12](#_Toc499388355)

[Updates 12](#_Toc499388356)

[Modifications 13](#_Toc499388357)

[Improvements 13](#_Toc499388358)

[Conclusion 13](#_Toc499388359)

[Product Specification 13](#_Toc499388360)

[References 14](#_Toc499388361)

[Appendices (Code, data etc. ) 14](#_Toc499388362)

[Appendix A 14](#_Toc499388363)

# Abstract

The project produced a device capable of fingerprint recognition with security features to verify matching and non-matching fingerprints and it was additionally capable of storing new fingerprints. The

user output consisted of an LCD display and keyboard, which were capable of interacting with the user, where they could select the memory location of the fingerprint or determine where it was located. A final feature was the use of an alarm to alert of a possible intruder.

# Intro and Theory

## Intro

Building a driver, which could control the scanner, required extensive coding. The microprocessor had to have all the instruction codes ready to transmit to the scanner and a way of dealing with the verification signals sent back.

A major focus of the project was also user interaction, whether the user would be able to understand the capabilities of the device and know exactly how to access the different features. There were multiple options presented on the LCD display and a clear welcome screen which explained how they could access these options.

## Theory

The type of sensor used here was optical. The device worked in a similar way to a camera taking a picture with the print recorded by CCD’s.

There are a few types of optical fingerprint scanner. The one here use Frustrated Total Internal Reflection or FTIR (1). This used the reflection of light from the fingerprint to produce an image of the finger. The varying reflectivity of the light from a groove compared to a ridge provided a method to distinguish between the two. It is generally accepted that as the difference in refractive index between two media increases, the Reflectivity also increases. The difference was greater for a ridge than a groove, as demonstrated in equation 1.

The device used image analysis, which checked certain features of the finger, or minutiae, for recognition such as the distance between certain lines.

Something that was expected to reduce the accuracy of the scanner was distortion on the surface of the sensor taking the print. This included the presence of grease or dust which would create either an out of focus stored image or a poor representation of any new fingerprints. Previous studies have highlighted this as an issue that standard optical scanners do not resolve (2). This not investigated, however was suggested as a reason for limited reliability in matching.

# High level Design & Software and Hardware design (Method)

Throughout the project, signals received were examined via storage internal SRAM and checking the value through Atmel Studio on the computer, this function has since been deleted.

All comparison outputs were also displayed on the LEDs however, this became redundant when the LCD display was initialized and could be used as an output.

Constant reference to the atmega128 datasheet helped to ensure that every step could be analyzed for what was going on computationally.

## Overall Schematic

A basic overview of how the microprocessor dealt with the various hardware can be observed in figure 1. Each input or output device had to be connected to the processor via a port, which may be viewed in figure 1, on the following page.

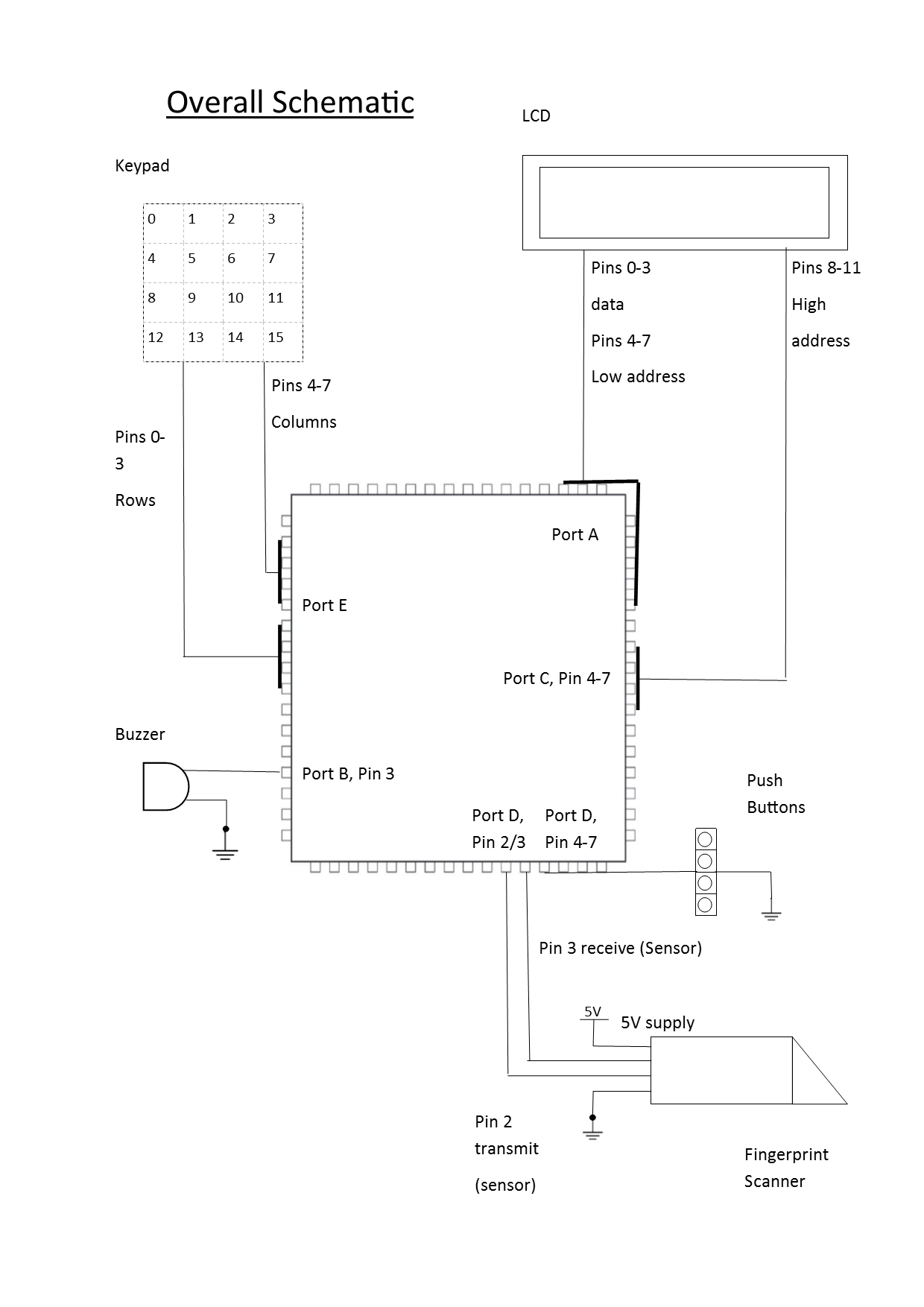


Figure 1: Diagram of overall setup. The full operation of each pin can be viewed in appendix A.

## General Hardware

### Keypad

This was solely an input device, letting the user select from 16 memory locations, listed from 0 to 15. When asked, the keyboard would provide a single byte, from which allowed the program to identify which button had been pressed. The keypad worked by running a current through a grid of wires, viewable in figure 2. If a button was pressed the voltage was pulled down on a particular wire resulting in a 0 for the pin attached to it.

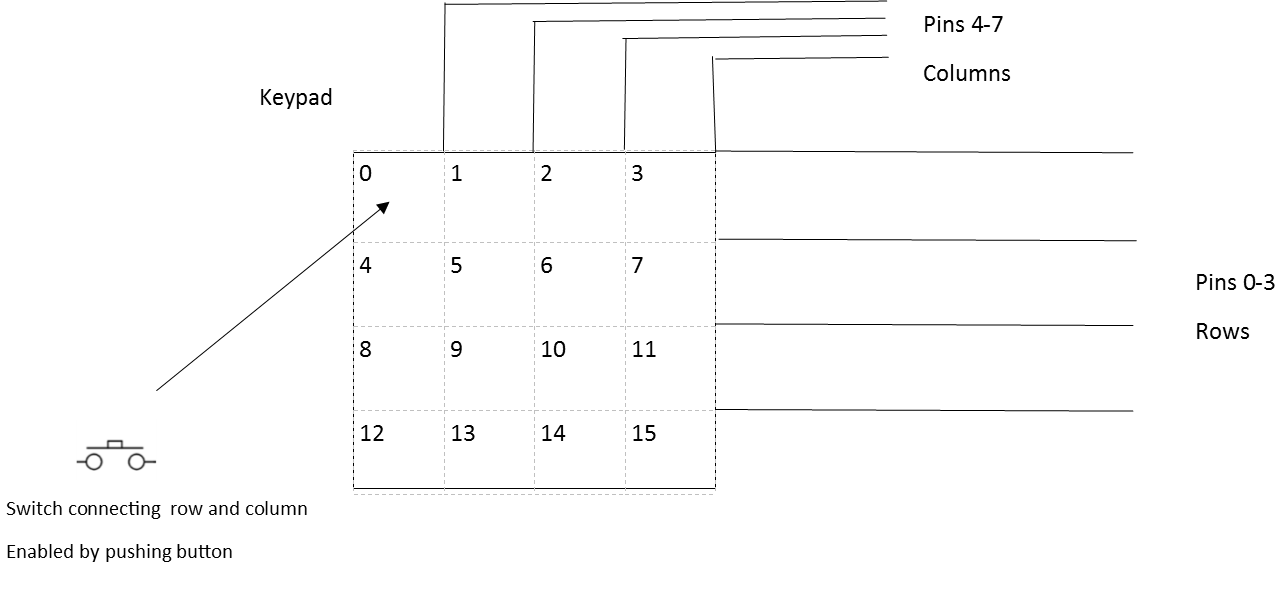


Figure 2: Diagram of the Keypad used.

### Buzzer

The buzzer was connected to Port B, where a specific pin controlled whether it received a voltage signal enabling it to transmit an audible signal. The buzzer was used as an additional output device to improve the security aspect where the device could immediately signal an unrecognized person.

### LCD

The LCD presented a more user-friendly experience where a person could be presented with options on how to interact with the device as shown in figure 3. It also meant a more varied way to display credible and fraudulent fingerprints. It allowed the specific ID of the person to be presented to the user.

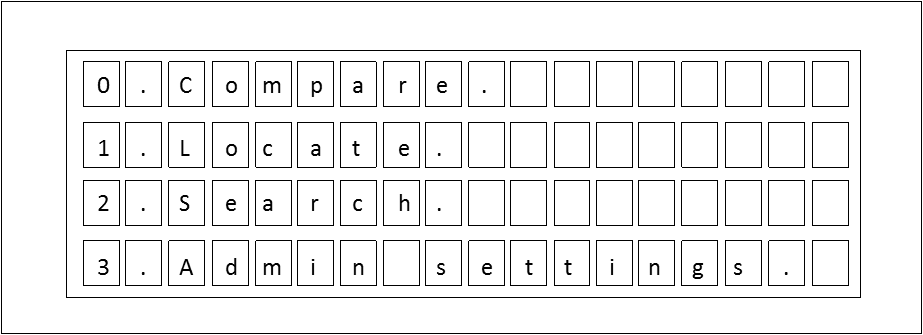


Figure 3: The layout of the options on the LCD.

### Pins

These were an input that allowed the user to select which action the User wanted to access. Pin D was sent up as an input port for this purpose. The ones used here were pull down pins, as can bee seen from the overall schematic. This meant they redirected power from the pins when they were pressed resulting in a different value being read from the pins. An index of four different values could be looked up to determine which pin had been pressed hence which option the User had selected.

### USART1

The USART1 was set by the Atmega128 to be controllable through port D. Information received was placed in the USART buffer. The microprocessor was programmed to collect the data and to store it in various SRAM locations. This can be seen in figure 4.

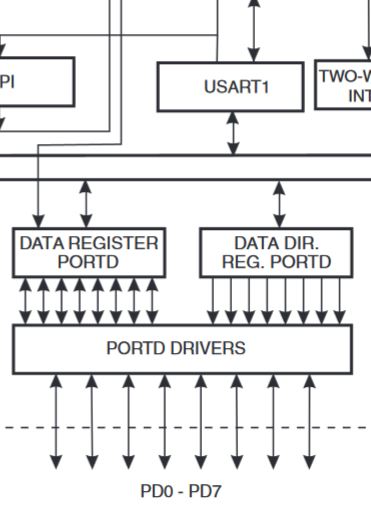


Figure 4: USART1 communication with Port D.

## Fingerprint Scanner (Hardware connections)

The fingerprint scanner itself interacted with microprocessor via serial communication. This meant transmission and reception had to be a single bit at a time. Pin 2 was set up for reception and Pin 3 for transmission. For transmission the bits were sent manually via software, using the UART\\_send\\_byte subroutine, which enabled a serial signal to be sent through port D. For reception, the USART1 hardware was employed.

### Transmission

Software was used to manually send serial signals. Subroutines were called that sent each individual bit of every byte. Loops sent the bits at carefully calculated increments so that they could be sent with the correct frequency.

### Reception

Originally, a similar method was employed to receive data where it was collected as individual bits and pin 2 was constantly polled for data. This however gave an issue were the Atmega128 board had to be prepared all the time and was not always quick enough to switch to a state where it could receive data. Later the USART1 interrupt allowed any signal received into Pin 2 of Port D to be saved in the memory, whenever data was available.

The previous error meant there were occasions where confirmation data on whether a fingerprint had indeed been recorded was not picked up by the Microprocessor. The hardware USART had to be set up to enable interrupts and included the use of the subroutine save\\_data\\_start352. Every Byte of data that was received in Port D was transferred to the USART1 buffer where it could be collected by the microprocessor before another byte was received into the buffer.

## Software

### Initialization

### Actions

The program was split into various modules, called actions. These were responsible for the features the User could access. The actions were kept in a separate include file. They would call shared subroutines from the main file to operate. An example is action5, given in figure 6. This action was responsible for counting the number of stored Fingerprints.

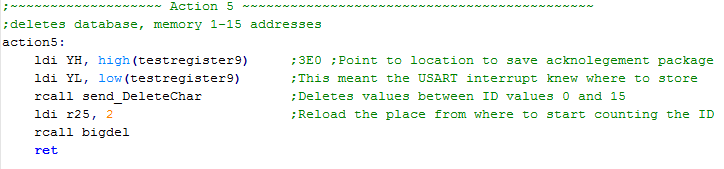


Figure 5: Action5, Actions were the largest blocks the program could be split into.

### Databases

Various databases with the different Command Packages split up into bytes had to be stored in the Atmega128. This meant various commands could be sent at various stages in the program to the Fingerprint Scanner. Figure 6 below shows the database required for telling the Fingerprint Scanner to count the number of Fingerprint currently stored on the device.



Figure 6: A database with a Command Package split into individual bytes.

The form of the command packages is provided in the datasheet for the Fingerprint Scanner.

### Sending Subroutines

A number of subroutines were used, each of which sent a different Command Package to the Fingerprint Scanner. Figure 7 below shows the subroutine which sent the Command for counting the number of stored templates, or fingerprints.

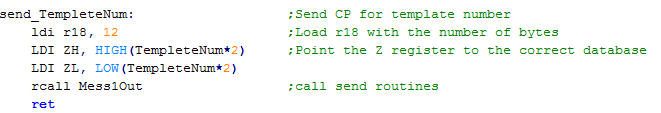


Figure 7: Snippet of the sending routine responsible for outputting the data from the Atmega128.

### Mess1out

This function allowed the data to manually be sent to the device, byte by byte. It was called as a subroutines by the send\_TempleteNum in figure 7, that pointed to the correct database. Mess1out then sent the bytes. Figure 8 shows how it looped through whole Command Package, sending the data byte by byte.

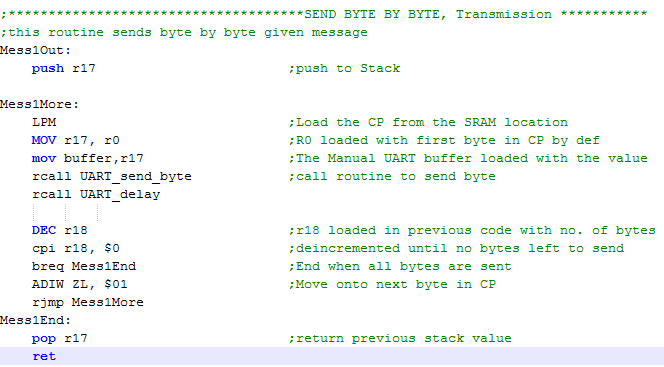


Figure 8:

### UART\_send\_byte

This Subroutine was set up to manually send bits of data. A snippet of the code may be viewed in figure 9.

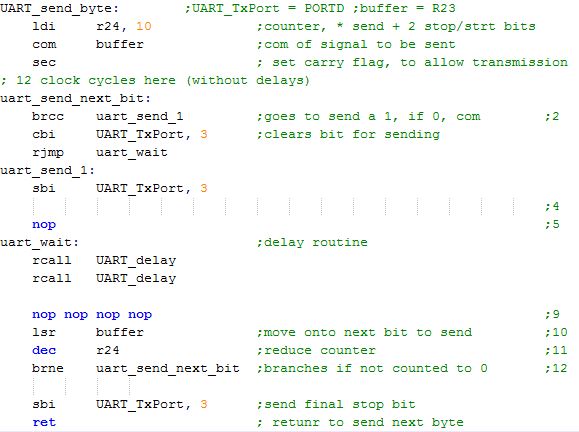


Figure 9: Snippet of Transmission code.

### save\_data\_start352

This interrupt sequence allowed any data that was captured in the USART buffer to be stored in the previously pointed to location in SRAM. It required USART1 to be set up in a way that allowed it to store data to the USART1 buffer whenever data was received into pin 3 of Port D which can be viewed in figure 10.

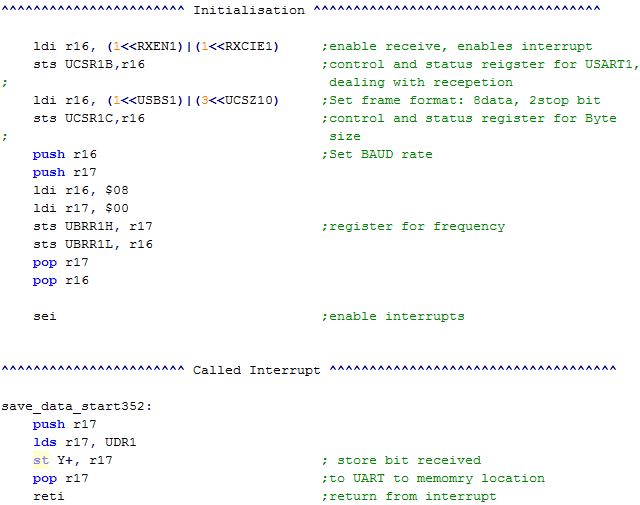


Figure 10: Code snippets of the set-up and routine for the USART1 interrupt subroutine.

### Compare\_basic2

Any input recorded indicating the success of a matching had to be compared to a database. This database had a unique sequence that determined whether the input indicated a correct Match. Comparing the input sequence to that of a sequence of a correct Match allowed the program to fully recognize whether the scanner had found a Match for the fingerprint. This was a relatively long section of code and may be viewed in appendix B.

## Testing the Software

Each sent Command Package would generate an acknowledgement package which would state what the Fingerprint Scanner had done. The acknowledgement package always included a byte of confirmation code that would verify a successful task or an inability to read the Command Package. These codes were given in the datasheet and throughout the production of the overall product, these were saved to SRAM. The various locations in SRAM where these were saved were 20 locations apart, the various locations can be viewed in figure 11. Several locations were required as several commands were sent so each step needed to be verified with a separate confirmation, each of which had to be stored in SRAM independently.

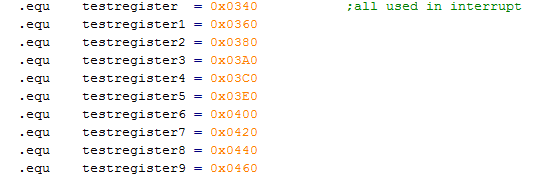


Figure 11: SRAM locations for storing various Acknowledgement packages.

# Results and Performance (results/ discussion)

The device performs all the actions mentioned in the plan. This includes the ability to recognize a fingerprint and to alert of a possible fraudulent identity via an alarm.

The majority of times the Fingerprint scanner did exactly what the User told it to do.

Action 1 always attempted took two fingerprint and the programme was not forced to stop if they were not presented n the time. in this case, it simply went back to the option screen.

Action 2, was called 'locate'. It performed the function of getting the fingerprint before asking the user for a location against which to compare the fingerprint. The only issue in this case being if the User did not press the keypad for a sufficient length of time, around 0.1s.

Action 3 performed the general database search. The on screen option for it was 'Search Database'. The function was only limited by the speed of searching. According to the datasheet, the fingerprint scanner could take up to a second to search the database. This caused a significant delay in receiving a result.

Actions 4, 5 and 6 were set as administrative options as they could alter the memory of the Scanner or check it. They required the User already be registered as an administrator. Currently, only Experimenter 1 and 2 were set as administrators.

Action 4 could store a fingerprint if it was presented and wouldn't take one if it was not presented. The only issue here being that the memory location for the next fingerprint wold be incremented, regardless of whether a fingerprint was taken or not.

Action 5 was able to check how many Fingerprints were already stored in the memory. This was a useful feature which allowed the User to know if the database had been cleared recently of non-permanent identities.

Action 6 was a simple delete function. Capable of clearing memory locations. Although he user could enter more fingerprints not accessible by the Keyboard i.e. beyond location 15. This function allowed these also to be removed.

There was also a possibility to return to the first option screen from the second, displayed on the second screen as 'return'.

## Extensions

The project went beyond the plan set out by including extra capabilities. This included administrative setting where the device did not just recognize the specified user; it could add new fingerprints, remove old ones and check the memory usage. A final adaption was a function to check the memory manually.

## Technical detail

### Investigating reliability

#### General reliability

The reliability of the fingerprint scanner for was measured for different fingers to check the variation of different sizes and type of finger. This concluded in the discovery the scanner recognized certain fingers more easily than others. It was very capable of measuring the thumb, index and middle fingers at around 80% accuracy. It struggled more with the fourth and fifth fingers with less than 70% accuracy, compromising the reliability of the scanner.

The quoted failure rate from the datasheet of the fingerprint scanner was 1% however, it was not mentioned if this was just for a particular person or a particular finger and whether they were instructed to present their finger in a certain way. The variation in size of the fingerprint appeared to be factor from experiment.

Testing for false positives resulted in the conclusion that it was highly unlikely to get a false positive with none returned in all forms of testing, this however was not recorded. This was also provided as 0.001% or 1 out of 100,000. Given that it was unlikely more than 1000 prints would be taken over the course of the experiment, this was concluded to be a futile to experiment formally.

### Errors

Multiple factors could have contributed to the discrepancy between the stated failure rate and the observed failure rate. This included the position of the fingerprint over the sensor and distortion from dirt and moisture on the surface of the scanner. These could warrant future investigation.

#### Effect of Moisture and dust

The most productive investigation would likely be a check the effect of a dry and wet print to determine if there was a noticeable difference. Previous study (2) has indicated that it may be a significant detractor form the reliability of the Scanner.

## Updates, Modifications and improvements

### Updates

Additional functions that can be added to the Product could involve a third screen to get a few more options.

One that would be relatively easy to implement would be toggling the alarm. This would be the equivalent of a feature that disabled the security. This would just require some coding that would only call the setup of Port B as an output port in certain situations.

Another feature would be to add a passcode override for the admin settings when a fingerprint is not recognized. This would be relatively easy to implement as the keypad is already set up and a simple routine could be called from the main screen which allows the admin settings to be accessed via a password.

One possible area of improvement could be the time between the User selecting an action and an output being displayed in the screen. Some of this can’t be completely avoided as it takes roughly one second, according to the datasheet, to carry out full search. Information on the screen whilst a search occurs could help the user to know how long they will be waiting for a result. This would be in the form of a load screen where a set of blocks would incrementally be displayed on the screen according to an estimate of progress.

### Modifications

[Things to remove]

### Improvements

The main issue was the accuracy meaning that someone would either have to repeatedly present their fingerprint to reduce the chance of false rejection or they would have to only present certain fingers. A method could be that someone would have to take multiple prints for a verification. This would involve a loop that kept taking prints and comparing them. If a successful print was found it would exit the loop. A timer could be implemented to end the cycle once 5 seconds had elapsed without a successful attempt. This would reduce the chance of false rejection to 16\% from 40\% for the little finger, assuming 2 cycles could be implemented in the time.

# Conclusion

We managed to complete the project to the point where the device could alert of a possible intruder via a buzzer as was set out. A slight variation was that we made it more of a user experience where they could interact with the device in several ways however it could also strictly be used as a security device.

# Product Specification

1. Number of Fingerprints stored simultaneously for direct keypad location: 15.
2. Maximum fingerprint memory capacity 162 Fingerprints.
3. Times

|  |  |
| --- | --- |
| Action 1 | 1 |
| 2 | <2 from Keypad input |
| 3 | <4 to Display outcome, from pin selection |
| 4 | <2 |
| 5 | 1 |
| 6 | 0.5 |

1. 5V DC
2. FRR 10% for Thumb Index and Third finger
3. FFR 35% for 4th and 5th finger
4. Scanner requires the User to be set up as an administrator by a technician to access admin features.
5. Dpi 86. Dots per inch.

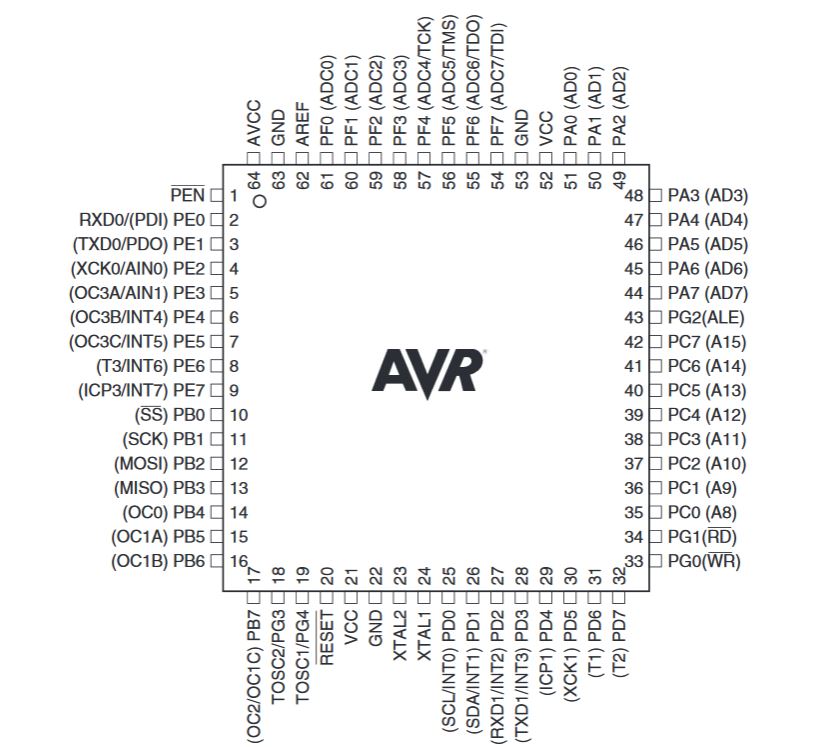
# References

1. Handbook : 978-1-84882-253-5
2. Seon-Woo Back, Yong-Geon Lee, Sang-Shin Lee, and Geun-Sik Son, "Moisture-insensitive optical fingerprint scanner based on polarization resolved in-finger scattered light," Opt. Express **24**, 19195-19202 (2016)

# Appendices (Code, data etc. )

13. Back S, Lee Y, Lee S, Son G. Moisture-insensitive optical fingerprint scanner based on polarization resolved in-finger scattered light. Optics Express. 2016;24(17):19195.

## Appendix A



Viewable