EE 459 Capstone Project
Spring 2024

Team 6: ProLock

Riad Alasadi, Arda Caliskan, Tyler Chen

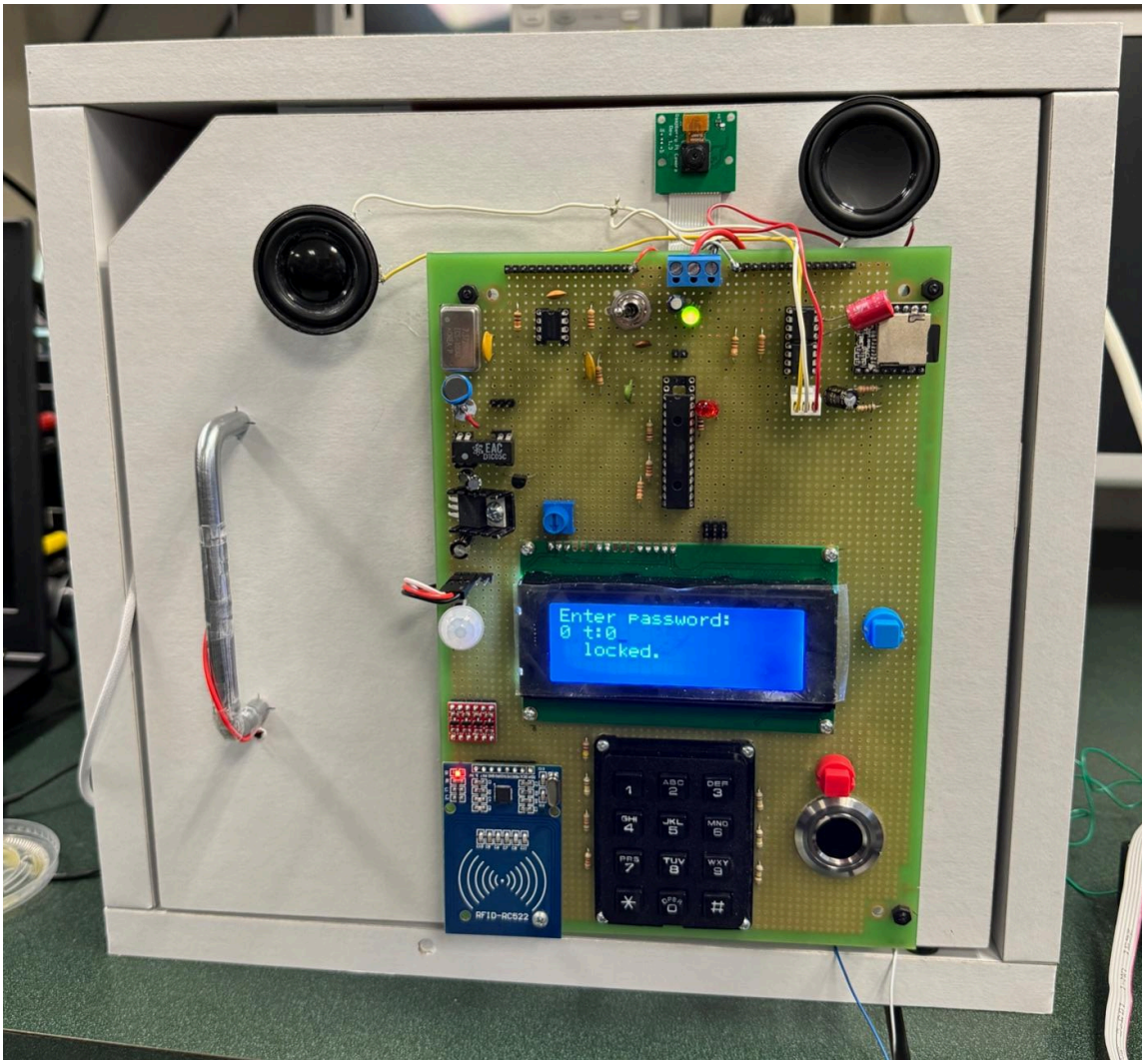


Table of Contents

1. Introduction

1.1 Problem Formulation and Approach:

The design of a lock is always a delicate balance between security and convenience. Most door locks today use pin tumbler locks and keys, which do not offer great security as they can easily be picked even by an amateur lockpick. Unfortunately they are not very convenient either, as keys need to be carried along and can be easily lost, and if multiple people need access the keys must be physically cloned. ProLock aims to solve these problems by creating a highly secure, configurable, and convenient to use lock by combining multiple high-tech methods of authentication. Many of these methods can run in parallel to get you in your home quickly without sacrificing your security. In addition, the lock can detect and record attempted break-ins and alert the user of such instances, which provides valuable peace of mind in these trying times.

2. Overview

2.1 Product Functionality:

Four different authentication methods are available: facial recognition via the camera, 4 digit passcode via the keypad, fingerprint via the fingerprint scanner, and RFID via the RFID scanner. The ProLock also includes a website where users can set their password, add their face and fingerprint, and update the number of valid authentication methods required to be granted access. For guests attempting to gain access to the door, the ProLock provides a doorbell feature where the guest can press the doorbell push button. This will allow the system to capture an image and send it to the website for the product owner to identify the guest and decide whether to grant or deny access. If the door handle is touched while the system is locked or if the authentication methods are unsuccessful several times in a row, then the system identifies an intruder and triggers an image capture and a loud sound track to be played on the speakers to deter the intruder. The image is then sent to the website to allow the product owner to view the intrusion attempt. For the prototype of the ProLock, a small cabinet with a door and door handle was used to convey the functionality of the product.

2.2 System Design:

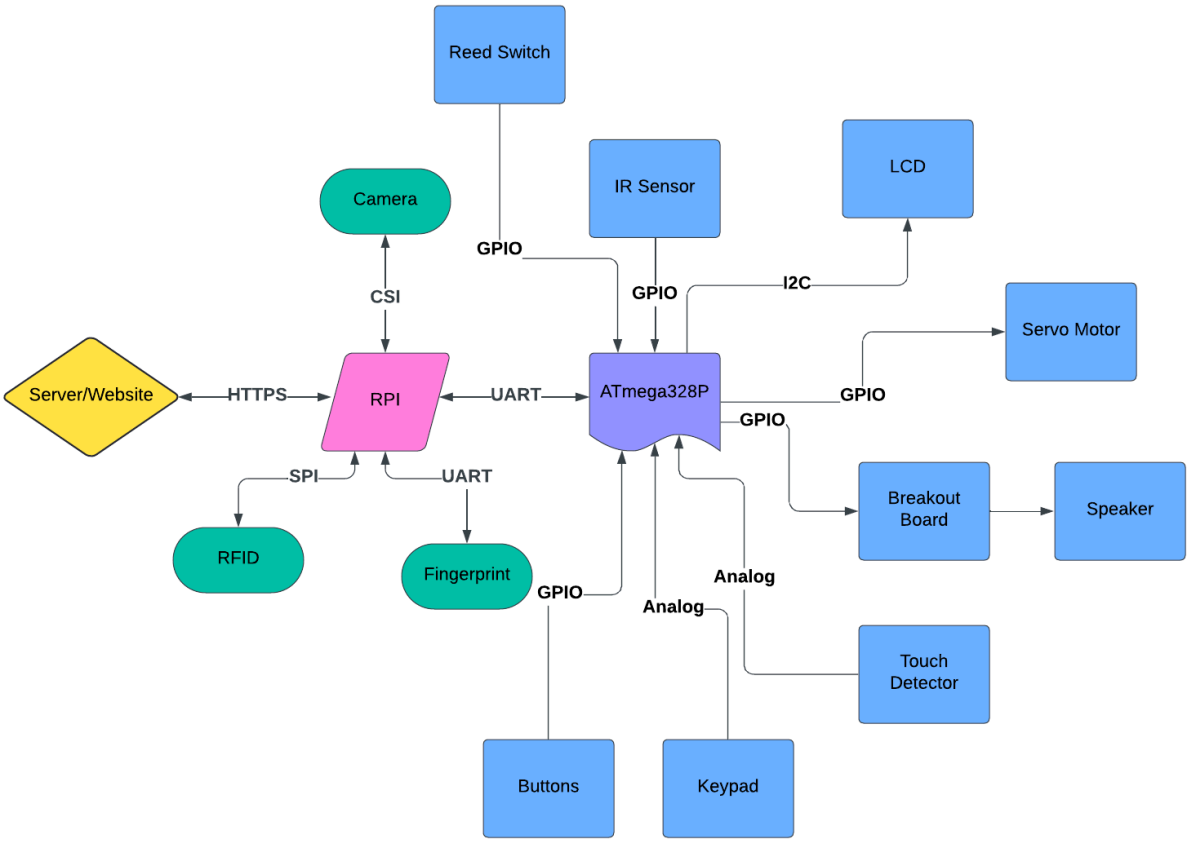


Figure 2.1

In Figure 2.1, the system design of ProLock is provided as an overview of how the system is integrated together. The ATmega328P is the main component of the system as it controls most of the I/O between the sensors, which include push buttons, an MP3 module, a servo motor, an IR Sensor, a Reed switch, a keypad, a touch detection system, and an LCD. The ProLock also contains a Raspberry Pi (RPi), which provides powerful computing capabilities that allow for low latency and parallelized interaction with the camera, RFID, and fingerprint sensor. The RPi is critical to the system as its multicore architecture allows for significant speedup in authentication in comparison to running everything off of the ATmega, which is one of the main selling points of ProLock.

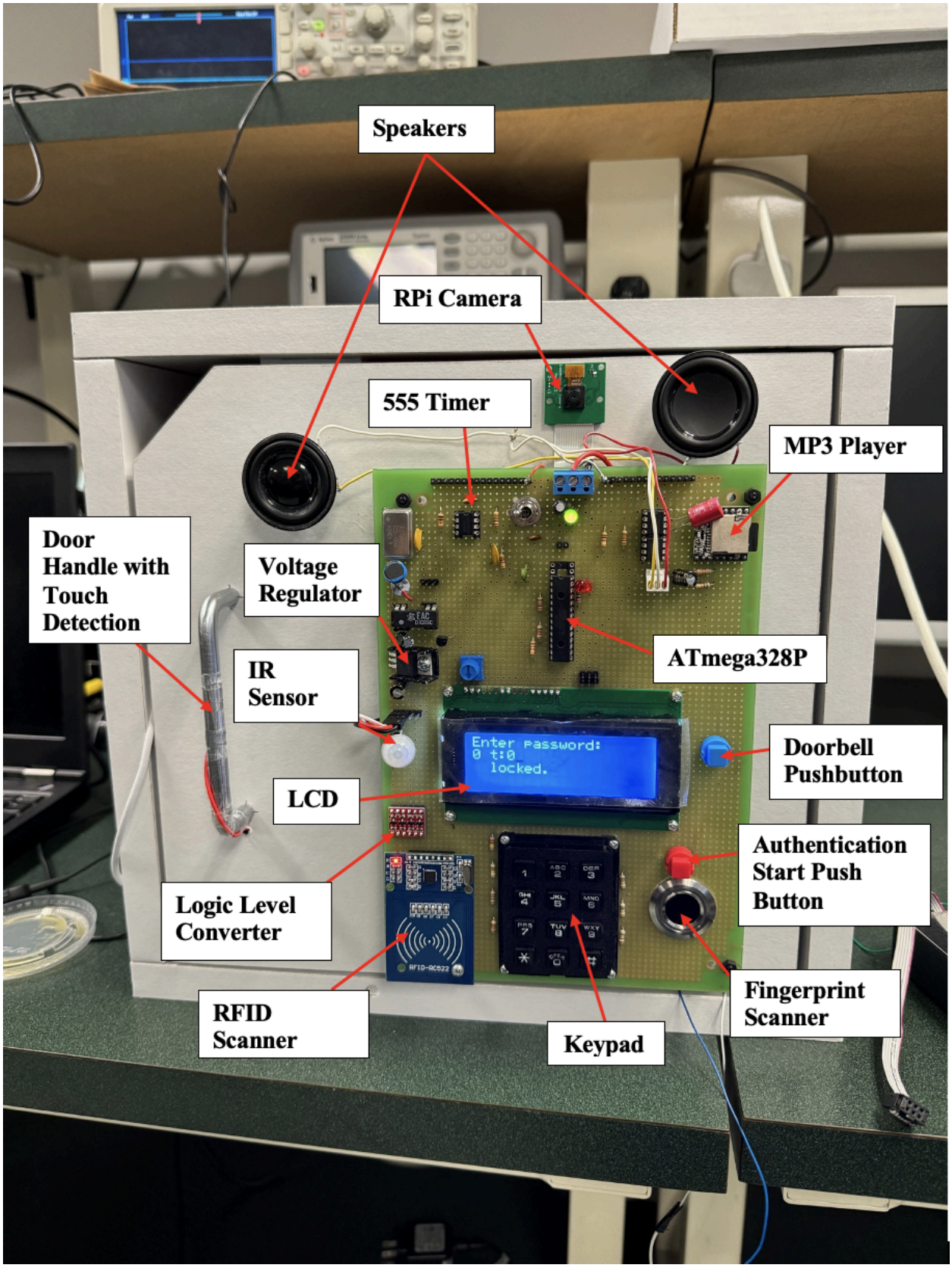

Speakers
RPi Camera
555 Timer
MP3 Player
Door Handle with Touch Detection
Voltage Regulator
ATmega328P
IR Sensor
Enter password:
0 t:0
locked.
Doorbell Pushbutton
LCD
Authentication Start Push Button
Logic Level Converter
Fingerprint Scanner
RFID Scanner
Keypad

Figure 2.2

3. Sub Systems

3.1 ATmega328P:

In Figure 3.1, the pinout of the Atmega328P can be seen with labels on each of the pins denoting their purpose. We are running the chip at 7.3728 MHz.

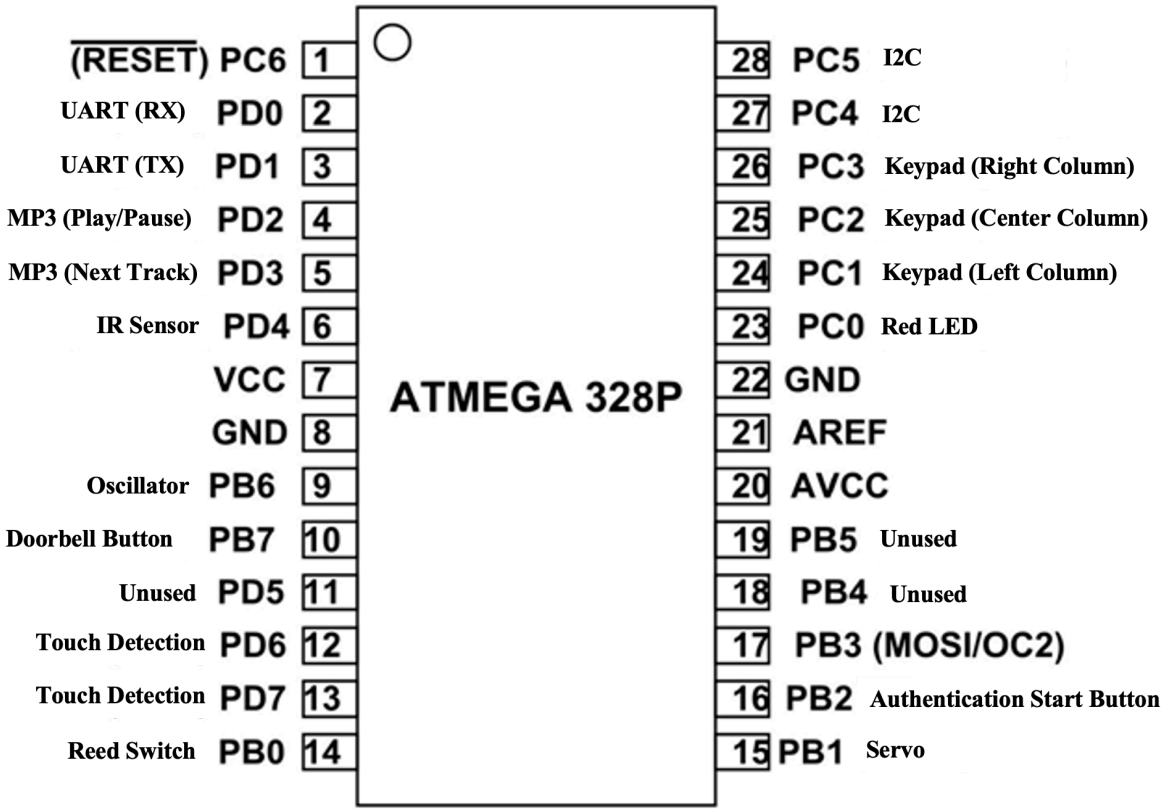


Figure 3.1

3.2 Shocker (No longer part of product):

The initial design of the ProLock included a shocker which would shock the intruder if the handle was held while the system was locked. This was initially included to deter any intruder from attempting to break into the system. However, due to safety concerns, this feature is no longer included in the design and the final product. The schematic and explanation of the shocker are included to provide information on the engineering behind this feature.

The shocking circuit was taken from a simple prank gum toy which we reverse engineered. It turns out that the entire circuit (Figure 3.1) has only three components: a switch for activation, an unmarked integrated circuit with three terminals (Figure 3.2), and a transformer. From the output waveforms (Figure 3.3) we suspect that the IC is just a simple coil driver which continuously switches the transformer on and off with a relatively low duty cycle. This, in turn, produces a large voltage pulse on its secondary coil which is then directly applied to the finger. For unknown reasons the return path of the high voltage was power, not ground. In the toy they used three 1.5V coin cell batteries for their power supply for a total DC voltage of 4.5V. However, we found that when powered from a DC power supply 4.5V produced much too strong of a shock, even for a thief. Therefore we settled on 3.3V–strong enough to deter, but not enough to burn. The switch in the toy circuit was the pulling action of the gum, which we replaced with a simple through-hole relay.

Figure 3.1

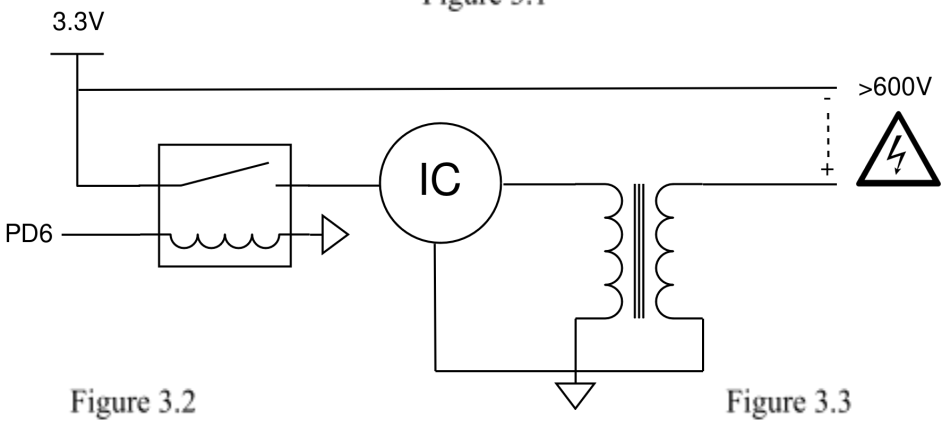


Figure 3.2

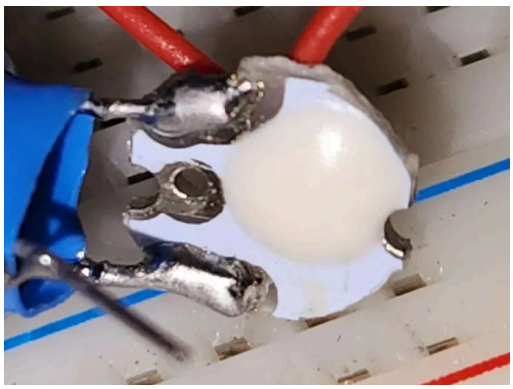


Figure 3.3

3.3 Touch Detection and Handle:

We found it necessary to be able to detect when the handle was being touched. Doing so, we would know exactly when to administer the shock and also take a picture anytime an unauthorized user tries to open the door.

We implemented this touch detection feature via capacitive sensing. We produce a 95% duty cycle square wave from a 555 timer circuit (see Figure 3.3). This square wave then continuously charges and discharges a small capacitance through a large resistor. The positive node of the capacitor is connected to the handle of the door, such that when the handle is held the voltage discharges slower due to the added capacitance of the hand (see Figure 3.4). This analog signal is connected to AIN1 (PD7) of the Atmega, which is the non-inverting input to the Atmega's internal op-amp. AIN0 (PD6), which is the inverting input, is connected to a constant reference voltage which is produced via a voltage divider. The Atmega has a feature which allows it to generate interrupts each time the internal op-amp's output toggles, simply by enabling the Analog Comparator Interrupt Enable (ACIE) pin in the Analog Comparator Control and Status Register (ACSR). Setting the Analog Comparator Interrupt mode Select (ACIS) bits further allows us to trigger interrupts on the falling edge of the op-amp output.

```
ACSR |= (1<<ACIE)|(1<<ACIS1); // trigger interrupts on falling edge
```

This greatly simplifies our code as we do not have to continuously poll at a high frequency to figure out if the analog output is going low or not, we can simply set a flag in the ISR, and then periodically poll the flag to see if it has been set. If the flag **has** been set, that means that the handle **is not** being touched. If the flag **has not** been set, that means that the handle **is** being touched. However, we found that we could achieve a much higher sensitivity if, instead of setting a flag, we increment a counter. This allows us to detect touches that are not strong enough to completely stop interrupts, but strong enough to stop some of them.

```
volatile unsigned int analogcount = 0;
ISR(ANALOG_COMP_vect){    // Called when not touched
    analogcount++;    // i'm not being touched
}
```

The polling of the count variable is done using timer 0 with prescaler 256 in CTC mode, therefore generating interrupts at 112.5 Hz. In the ISR we check if the count is greater than 300, which we empirically found to be a good threshold for the handle not being touched. There is some extra code which turns on the interrupts for only 1/12th of the time, which we will get into

in the debugging section. The touched variable is the clean flag that is consumed in the main loop.

```
ISR(TIMER0_COMPA_vect){    // Have I been touched?
    if((isrcount % 12)==0){
        ACSR |= (1 << ACIE);
    }
    else if(((isrcount-1) % 12) == 0){
        ACSR &= ~(1 << ACIE);
        if(analogcount > 300){ // no, i have not been touched a lot
            touched = 0;
        }
        else{  // yes, i have been touched quite a lot
            touched = 1;
        }
    }
    analogcount = 0;
    isrcount++;
}
```

The main loop code is shown below. The UART packet is being sent to the Rpi to take a picture, which we will get into in the software section. The lock_flag variable is 1 if the door is locked, therefore the if statement executes if the handle has just been touched and the door is locked.

```
if(touched && !touch_handled && lock_flag){
    uart_transmit(0x04);    // take picture of intruder
    PORTC |= 1 << 0;    // indicate on red LED
    PORTD |= 1 << 5;    // administer shock. Disable after demo.
    play_track(2);     // get off my door!
    touch_handled = 1;    // sound should only play once
}
else if(!touched){
    PORTC &= ~(1 << 0); // Turn off the LED when no longer touching
    PORTD &= ~(1 << 5);    // relieve shock
    touch_handled = 0;
}
```

In order to apply this sensing and shocking technique to the handle we required a handle with three separate conducting terminals: one for the high voltage, one for the return path of the high voltage, and one for the touch detection. In order to create these three terminals we took a metal handle and applied tape to the back of it to create an insulating surface, which we then taped small squares of tin foil onto in order to create the two extra terminals. The end result can be seen in figure 3.6.

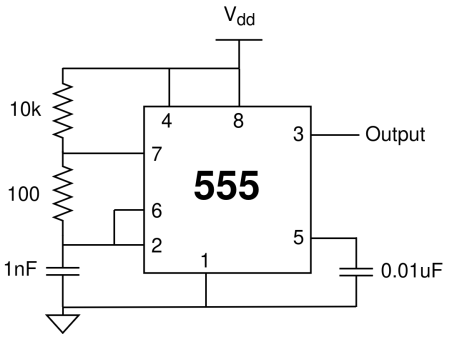


Figure 3.4

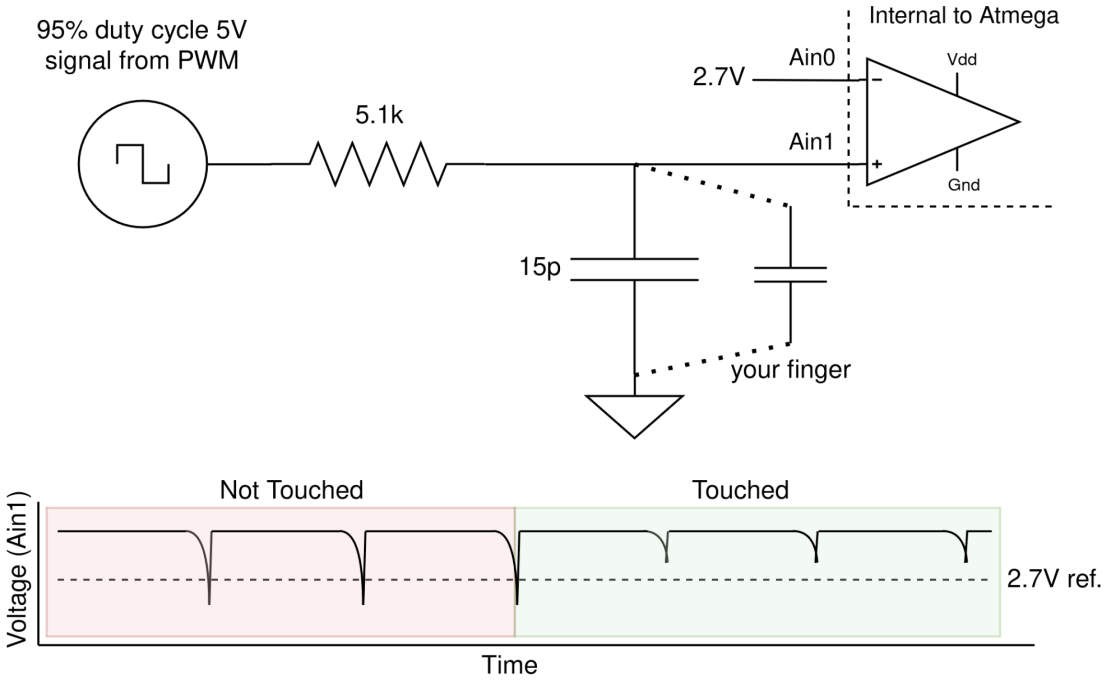


Figure 3.5

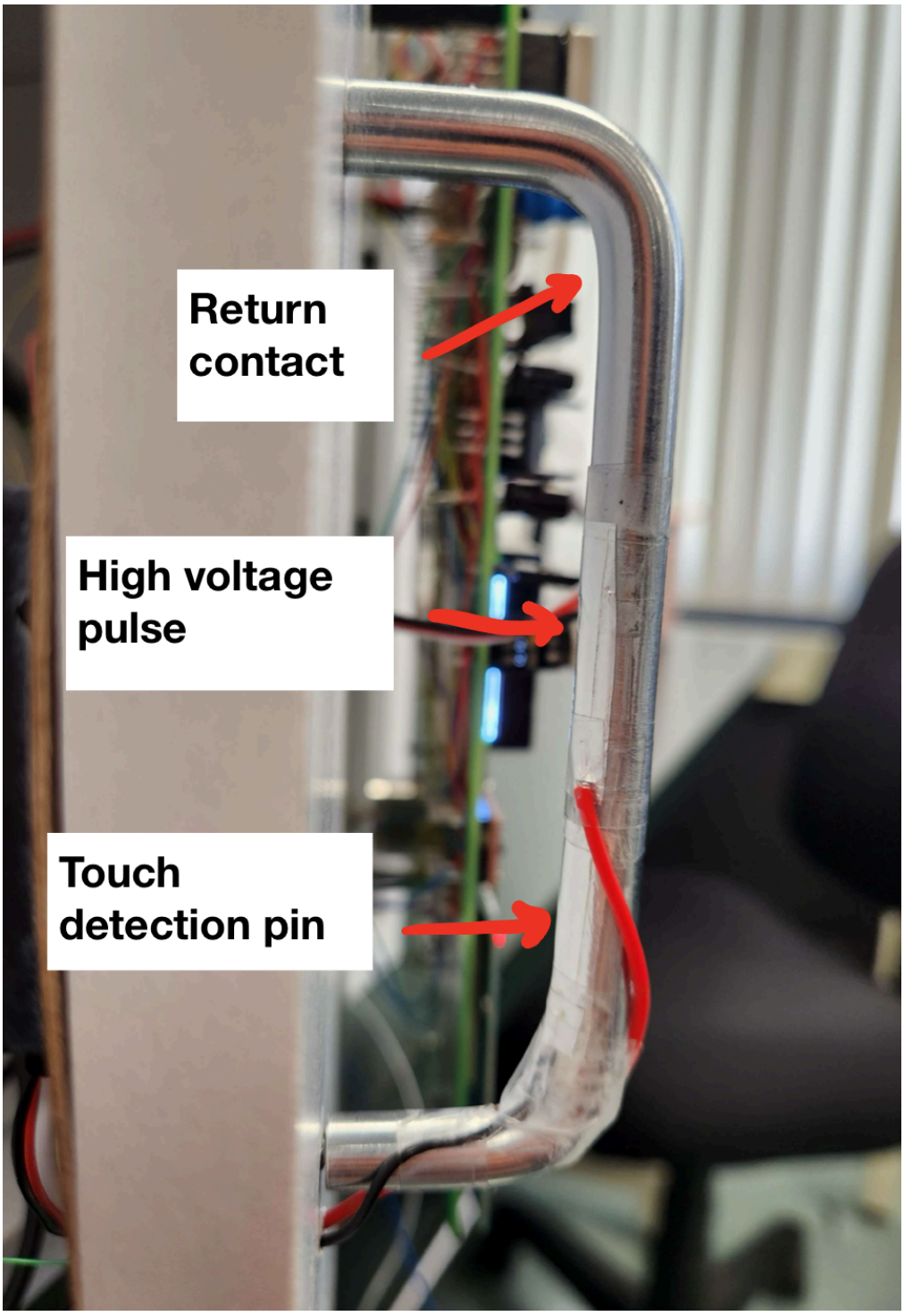


Figure 3.6

3.4 Voltage Regulator:

We needed a 3.3V supply to power the MP3 player and shocker circuit. We chose the ‘LM3940’ for its simplicity and high current output (1A). The schematic is shown in Figure 3.7.

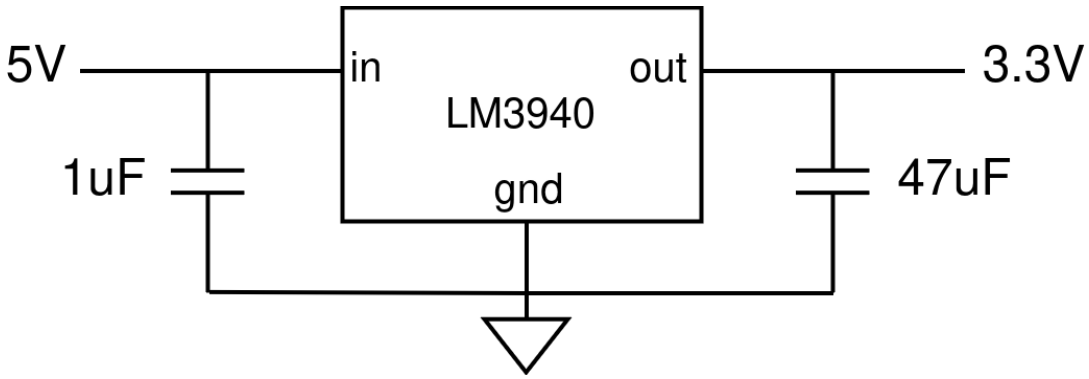


Figure 3.7

3.5 Audio System:

We wanted the door to provide audio feedback as to when access was granted or denied, so we looked for a simple audio module that we could trigger using the ATmega. We settled on the ‘DFPlayer Mini MP3 Player’ due to its small size, high compatibility with various SD cards, and simplicity in interfacing. The module has two analog input pins which you can pull down with different resistances to do different things. We only needed two of these resistances on one of the pins, 33k and 15k on ADKEY1, which provide functionality to play/pause and skip to the next track, respectively. The two DAC outputs, right and left, are connected to LM386 amplifiers which we chose for their wide applicability and cheap cost. The outputs are then fed to a capacitor network which removes high frequency noise and blocks the DC biasing. The circuit is shown in Figure 3.8.

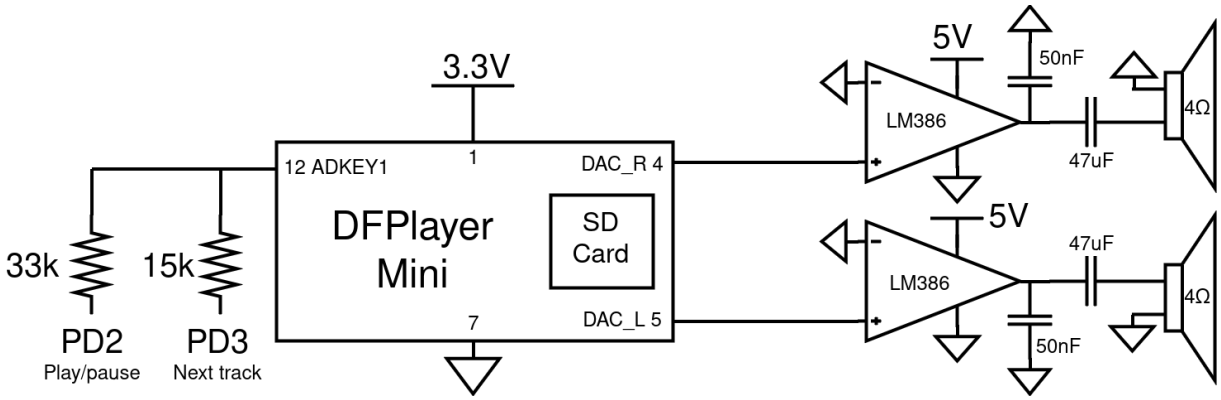


Figure 3.8

When we want to trigger the functionality of the ADKEY1 pin we simply enable PD2/PD3 as an output with value 0, which pulls down the ADKEY1 pin with the specified resistance, and then we set it back to an input to let it float again.

```
#define PLAY_PAUSE 2
#define NEXT 3

void play_pause(void){
    DDRD |= (1<<PLAY_PAUSE);    // Pull ADKEY1 down
    PORTD &= ~(1<<PLAY_PAUSE);
    _delay_ms(40);          //Short press for play/pause
    DDRD &= ~(1<<PLAY_PAUSE); //Switch to input again for not pressed
}
void skip_track(void){
    DDRD |= (1<<NEXT);    // Pull ADKEY1 down
    PORTD &= ~(1<<NEXT);
    _delay_ms(60);     //Short press for skipping track
    DDRD &= ~(1<<NEXT);    //Switch to input again for not pressed
}
```

In order to automatically remember which track the module is on, we also implemented a function for playing a specific track:

```
unsigned char num_tracks = 5; // current # of tracks on SD card
unsigned char current_track = 5; // starts with last track

void play_track(unsigned char track){
    if(track == current_track){
      play_pause();
    }
    else{
        while(track != current_track){
            skip_track();
            _delay_ms(60);   // min delay for skip to register
            current_track--; // the pointer moves backwards
            if(current_track < 1){
               current_track = num_tracks;
```

```
            }
        }
    }
}
```

3.6 Locking Mechanism:

The locking mechanism works through the use of a servo motor. To ensure that the locking mechanism is robust, the “Miuzei 20KG Servo Motor High Torque RC Servo” was selected as the component for this sub-system. This is a very strong and sturdy servo and succeeded in its functionality in preventing the door from being opened when locked.

We mounted it by sticking the back surface to the inside of the door and making an inlet in the inner side of the left surface of the cabinet. The purpose of this is that when the system is unlocked, the servo motor hand is in the 90° position when unlocked so the hand is not in the inlet, and in the 0° position when locked so the hand is in the inlet and stops the door from being opened.

Figure 3.9

Figure 3.10

For the control signal we chose to use a 23 Hz PWM signal ranging from 2.5% duty cycle to 4%, which represents 0° and 90°, respectively. The code which initializes this is shown below.

```
void servo_init(void) {
    // Clear on compare match, set at BOTTOM (non-inverting mode)
    // Fast PWM mode with ICR1 as top
    TCCR1A |= (1 << WGM11) | (1 << COM1A1);
    // Prescaler = 8
    TCCR1B |= (1 << WGM12) | (1 << WGM13) | (1 << CS11);
    // Set ICR1 so that PWM frequency = 50Hz (Period = 20ms)
    ICR1 = 39999;
    // Set PB1 (OC1A, Digital Pin 1) as output
    DDRB |= (1 << PB1);
}
```

3.7 Reed Switch:

An important feature of the ProLock is that it can sense when the door is closed or open. This is important because after the door has been unlocked, the door should automatically lock after some arbitrary time. However, it must be ensured that the servo motor does not move to the lock position while the door is still open. Therefore, a reed switch was included in this design. The reed switch is placed at the bottom of the inside of the door and it is connected to PB0, initialized as an input with the internal pull up resistor enabled. The reed switch is normally closed and opens when it interacts with a strong magnetic field. So a small magnet is placed on the base of the cabinet which causes the switch to open when the door is closed. Therefore, when the door is closed PB0 reads high, and when the door is open PB0 reads low.

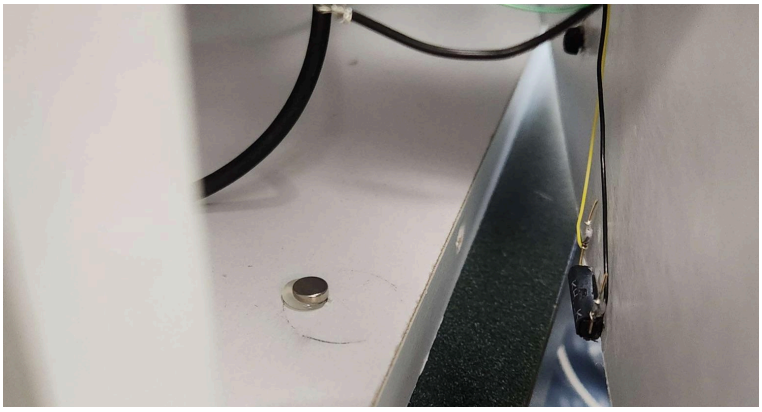


Figure 3.11

3.8 Keypad:

For the password input we chose to use the Sparkfun - “COM-14662” bare metal keypad with 7 pins, 3 for each column and 4 for each row. When a button is pressed, the pin of the pressed row is connected to the pin of the pressed column. A typical application of a keypad such as this would be to wire all the pins to the microcontroller and set the columns to an output in the high state, and the rows to an input with pull-up resistors enabled. You could then selectively set a column pin to low, and then read all of the row pins to see if one of them was pressed. The main drawback to this method is that it requires 7 pins, which is not ideal for our design. So we decided to instead use analog encoding, the schematic is shown in Figure 3.12. This method uses only 3 pins on the Atmega, PC1, PC2, and PC3. We continuously run analog-to-digital conversions in the main loop to determine if a button has been pressed, and can tell which column is pressed by observing which pin's voltage has changed from Vdd, and which row is pressed by comparing the ADC result to the known encodings. Since the pull-up resistances are different, each row will produce a different voltage. If multiple buttons are pressed at the same time, an incorrect voltage will be produced and the keypad will not register the press.

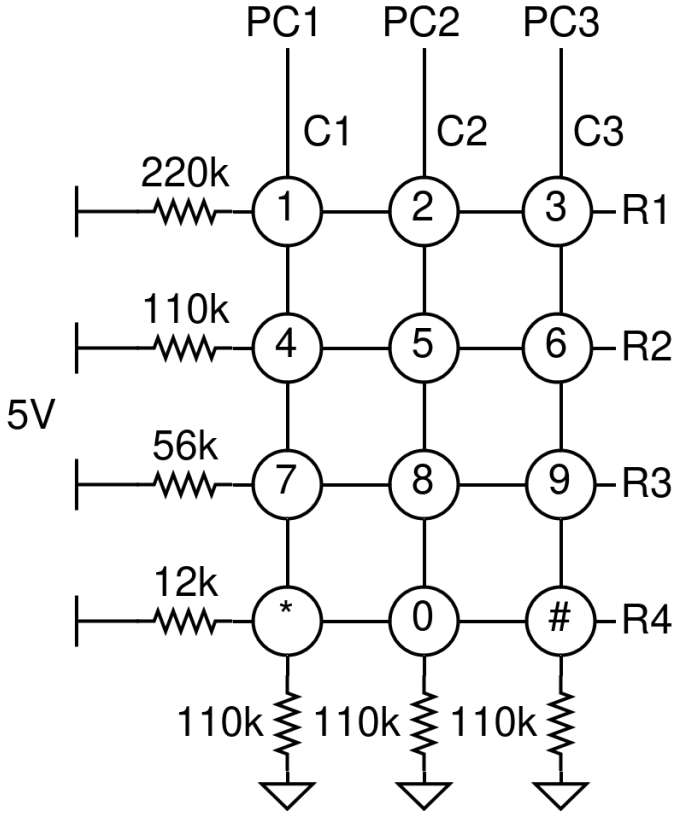


Figure 3.12

3.9 RPi:

The ProLock’s functionalities include several different options for authentication, and the use of an RPi provides the performance for the product to be able to offer these without too much additional latency. The facial recognition algorithm, fingerprint scanning, and RFID verification

all run on separate threads, enabling the user to perform these authentications in parallel. The RPi communicates with the ATmega328P over UART to determine when the RPi processing needs to take place and to inform the ATmega328P of the results of these authentication attempts. Since the RPi's logic level is 3.3V while the ATmega's is 5V, a logic level converter had to be used. We chose the Sparkfun BOB-12009 - "Logic Level Converter" for its simplicity and reliability. In addition to its communication with the ATmega, the RPi client also communicates with our main server over HTTPS to send and receive data between each other. The breakdown of the RPi pinout is shown in figure 3.13.

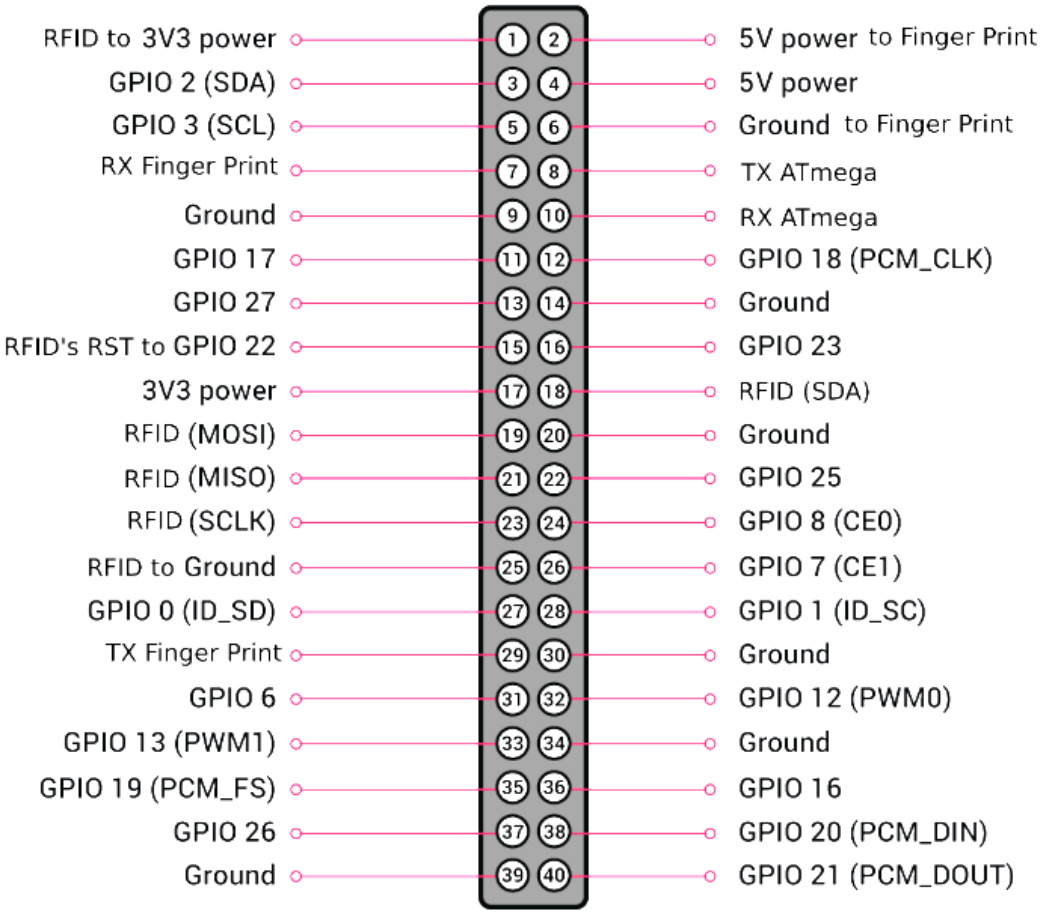


Figure 3.13

3.10 Facial Recognition:

The flagship function of the ProLock is that it allows you to use your face as a method of authentication, powered by the python face-recognition library and the 'RPi Camera Module V2'. When the user is in position to attempt facial recognition, they simply push the red push button which sends a UART packet to the RPi telling it to run the algorithm. The RPi then uses the mounted camera to take an image and run it through the facial recognition model to produce

the embedded face information, and then compares it with the existing authenticated embeddings. The RPi handles this processing locally and sends a UART packet with the authentication result back to the ATmega. This entire process happens within 5 seconds, and is threaded so that the user can also authenticate their finger and RFID in parallel. Our fast response time is attributed to caching the embeddings of the authenticated faces, and reductions in the image resolution. Additionally, the doorbell feature of the product also results in the camera capturing an image of the individual when the doorbell push button is pressed or when an intruder is detected.

3.11 Fingerprint Scanner:

The fingerprint module allows users to authenticate with their fingerprint. We chose to use the Adafruit 4651 - “Rugged Panel Mount Fingerprint Sensor” for its ease of mounting. The module communicates directly with the RPi using UART and a software serial library, since the main UART pins of the RPi were taken up by the ATmega. For the code we took reference from the python library adafruit-circuitpython-fingerprint, just tweaked it so that the scanning was threaded. Using the website the user can activate the fingerprint adding mode, and then after two short presses of the scanner their fingerprint will be saved in the device forever. When they are ready to authenticate with their fingerprint they can click the red push button which begins an authentication attempt, and after pressing their finger once onto the scanner the result (valid or invalid) is then sent back to the ATmega.

3.12 RFID Scanner:

The RFID module allows users to authenticate with the corresponding RFID card. We chose to use the ISOKEE - “Card Reader Module for RFID RC522” for its ease of mounting and high security. The module communicates directly with the RPi using SPI. For the code we took reference from the python library mfrc522. Again, we simply tweaked it so that the scanning was threaded with the main python loop. At any time the user can tap the RFID card onto the scanner, and the module will authenticate the card virtually instantly.

3.13 Push Buttons:

There are two push buttons used in this system - one for the doorbell (PB7) and one for starting facial recognition and fingerprint authentication (PB2). Their corresponding pins on the ATmega are initialized as inputs and the internal pull up resistors are enabled. The buttons are debounced using simple delays, shown below.

```
void handle_button_press(char bit){    // Function to check for
button press
    _delay_ms(5);
    while((PINB & (1 << bit)) == 0){
      _delay_ms(5);
```

```
    }
}
```

3.14 LCD:

The LCD serves as an interface for the user. Important information like the number of required authentication methods, the feedback for the various forms of authentication, and the updating of settings are all displayed on the LCD. We chose the Crystalfontz - “CFAH2004AC-TMI-EW” 20x4 LCD for its high reliability and option for I2C, as we were short on pins. We found the LCD to be very reliable and well-made, and recommend it for future projects.

3.15 IR Sensor:

The IR sensor is included in ProLock to identify if there is any movement in front of the door. We chose the Parallax 28033 - “PIR Mini”, as we were impressed with its highly compact form factor. However after testing the device we were disappointed to discover that it has an internal timeout functionality. When movement is detected the output signal goes high, but does not go low again until movement has subsided for a few seconds, which limits its utility within our project. Nonetheless, we still used it to provide “greeting” functionality. If an individual moves in front of the door locking system, the speakers say “hello there!”. After 20 seconds, if there is no interaction with the door and there has been no movement, the sound will be played once again if there is movement in front of the door again.

3.16 Website:

The website is a complimentary interface for the ProLock that allows you to change passwords as well as adding your face and/or fingerprint to the system. Additionally, any pictures captured of intrusion attempts will have their face shown on the website. The doorbell functionality is also powered by the website–when the doorbell is rung the RPi captures an image and sends it to the website, which is then displayed to you along with an option to either grant or deny access to them.

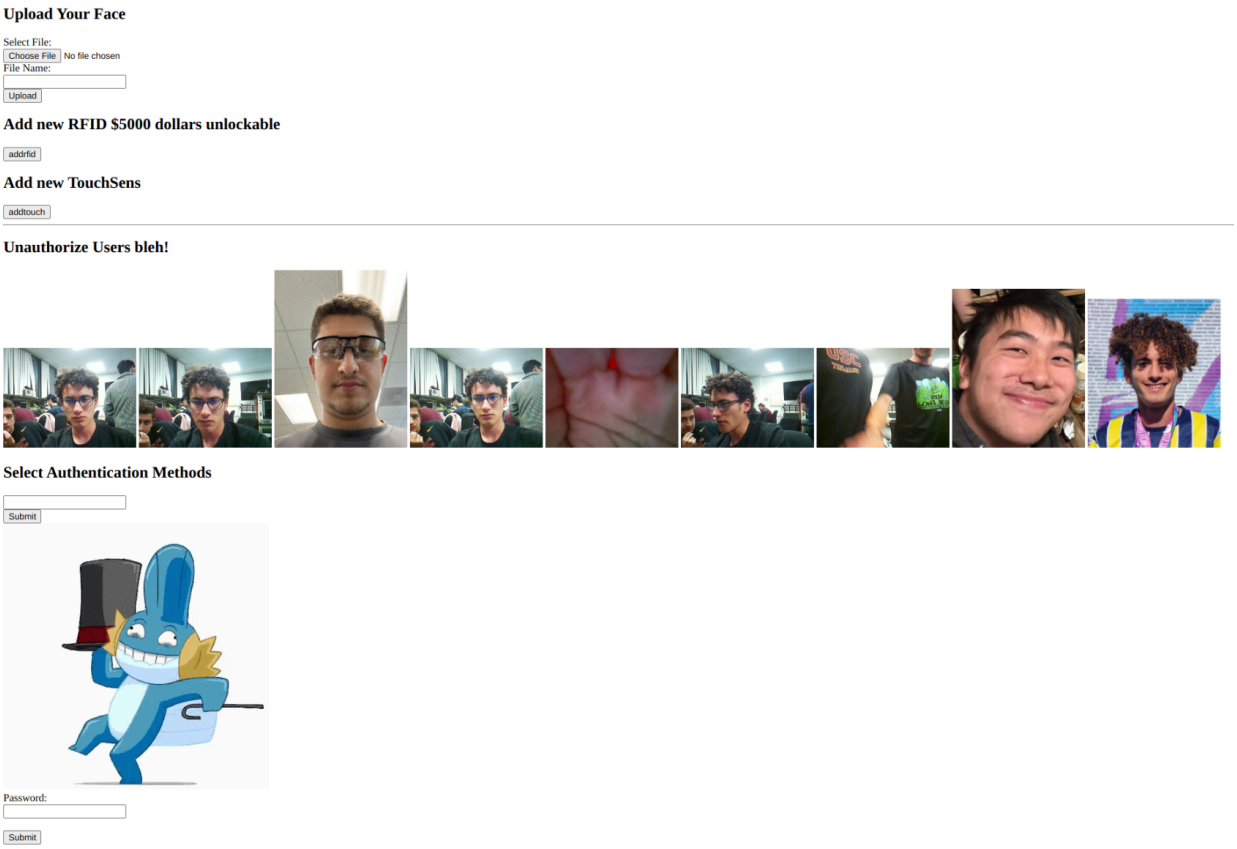

Upload Your Face
Select File:
Choose File
No file chosen
File Name:
Upload
Add new RFID $5000 dollars unlockable
addrfid
Add new TouchSens
addtouch
Unauthorize Users bleh!
Select Authentication Methods
Submit
Password:
Submit

Figure 3.14

4. Software

4.1 Atmega328p Software:

Specific implementation of functions are elaborated in part 3. The code base is segmented into different c files that define and implement functions relating to protocols or features of the ProLock. The main code loops through conditions in a never ending while loop checking whether to execute a particular part of the code.

4.2 RaspberryPi Software to Server:

Our RPi serves as a client that handles incoming requests from the server and generates requests to the server based on data from the microcontroller. The RPi contains a Python script that runs in a never ending while loop which constantly polls the server to determine if it has any tasks to be done (add fingerprint or change password). This is necessary as the device has to be able to operate behind even the most stringent of firewalls, therefore we coded it under the assumption that the server cannot talk directly to the client *unless* it is spoken to first. This rule does not apply in reverse, as our server is open to the public internet. If the RPi has data it needs to send to the server (image of intruder/guest) it does so directly, by using curl post commands to various endpoints of the server, and is identified by its IP address.

4.3 RaspberryPi Software to ATmega:

The RaspberryPi communicates a significant amount of data with the ATmega over UART, therefore it was necessary to create a simple communication protocol. The following paragraphs detail this protocol.

Every packet begins with a Packet ID, which is used to determine the meaning of the data being sent. The Packet ID is then followed by some amount of bytes, which are the payload. For operations that have a variable amount of bytes (the name associated with a face), a terminate byte of 0xFF is appended to the payload.

Packets from RPi to ATmega		
Packet ID	Payload structure	Action
0x01	{0x00} = invalid face {‘n’, ‘a’, ‘m’, ‘e’, 0xFF} = valid face	Receiving face result
0x02	{0x00} = invalid fingerprint {0x01} = valid fingerprint	Receiving fingerprint result
0x03	{0x00} = invalid RFID {0x01} = valid RFID	Receiving RFID result
0x04	{MSB, LSB} = new password encoded in 16 bit payload (0000-9999)	Receiving updated password
0x05	{auth_byte} = one-hot encoded byte of selected authentication methods	Receiving message of selected auths
0x06	{0x00} = deny access {0x01} = grant access	Receiving message to grant/deny access
0x07	No payload	Receiving message that touch has been added

Packets from ATmega to RPi		
Packet ID	Payload Structure	Action
0x01	No payload	Request face recognition
0x02	No payload	Request fingerprint authentication
0x03	No payload	Take picture of guest
0x04	No payload	Take picture of intruder

4.4 Server Software:

The website is hosted using NGINX, which acts as the primary web server. For specific sections of the website, NGINX is configured to forward requests to a Flask application based on predefined URL paths or patterns.

On the website, the user can update the password, change levels of authentication, and add a new fingerprint. Upon hitting the request button for each corresponding request, the Flask application will convert the request into a JSON format. The JSON is formatted in a key value pair where the key signifies the type of request (password change, fingerprint change, etc.) and value is the payload (new password, etc.). Sometimes the corresponding value is left blank as it might not be needed (new fingerprint). These requests are placed into a global queue. Upon request by an appropriate device, the IP address of the requester is checked against a hardcoded database, and the buffer's contents is sent to the RPi to handle, and the buffer is emptied.

5. Debugging

Fortunately, we did not face many debugging issues in the final design of the ProLock. We ensured that the components selected did not have any known difficulties or defects. Therefore, the only significant issue that we faced was a bug in the audio module where after a variable amount of time the play_track function would lose track of the current track and play the wrong sound file.

5.1 Skip Track Function Failure:

To play the various sounds that the ProLock MP3 module contains, we use the play_track function detailed in section 3.5, which calls the skip_track function in a for loop to skip to whichever track you want to play. It is critical that all the skips in the loop register all the time, as if even one skip fails to register then you will always be playing incorrect tracks until the unit is power cycled, which is exactly the problem that we faced. At some point in the program some of the skips were not registering, and as a result the unit would play a random track instead of what it was supposed to be playing.

After a while of narrowing down the bug, we were able to notice that it was related to the touch detection of the handle. If you "flicked" the handle (touch it very briefly), all skips would register and it would play the signature intruder-spooking sound of "get out my door". But if you held the handle down and then released it, it would not play the correct track and the sounds would be all wrong from then on. After this realization we scrutinized the touch detection code down to the assembly level, and were able to find the problem. It turned out that our analog ISR that we thought was very fast was actually not so fast, taking up 22 clock cycles. When you consider that the interrupts are being generated at 117 kHz this becomes problematic, as 2.6 million clock cycles of this ISR are being inserted into the pipeline every second, and our processor is only running at 7.3728 MHz. This explains why flicking the handle worked to play the correct track but holding it down did not–when you are holding the handle interrupts are not being triggered, which means the delays are significantly faster than normal, which causes them to fail to register. This is why we added code to only turn on interrupts 1/12th of the time shown in section 3.3, as we needed to reduce the number of clock cycles being inserted into the pipeline

in order to keep the duration of the delays as consistent as possible, which was necessary to minimize the time taken to seek a specific track. If we had more time we would have rewired the 555 timer to a lower frequency instead of relying on a software solution.

C code:

```
volatile unsigned int analogcount = 0;
ISR(ANALOG_COMP_vect){    // Called when not touched
    analogcount++;    // i'm not being touched
}
```

Compiled into assembly:

```
// 4 clocks for jump to ISR
push r25               // 2 clocks
lds r24,analogcount    // 2 clocks
lds r25,analogcount+1  // 2 clocks
adiw r24,1             // 2 clocks
sts analogcount+1,r25  // 2 clocks
sts analogcount,r24    // 2 clocks
pop r25                // 2 clocks
reti                   // 4 clocks
// Total: 22 clocks
```

6. Future Improvements

6.1 iOS/Android Application:

Future improvements involve creating an iOS and Android application. Although the website works well with the ProLock, developing a standalone application allows for more functionality to control the Prolock. It's simpler to pull out a phone and open an application to control the device than to open a web browser and visit the website.

6.2 Adding RFID:

Our system works with authentication of pre-verified RFID cards. We'd like to add functionality to allow users to add new RFID cards and to be able to delete pre-existing cards from the website.

6.3 Use Internal Timer for Touch Detection:

The initial design of the ProLock had all the timers occupied, and so an external PWM signal using a 555 timer was required for the touch detection. This led to some issues as seen in the Debugging section. After finalizing the design, a timer became available, and this could be used instead of a hard-wired PWM signal.

6.4 Alternative to RPi:

One of the key reasons for using an RPi was for the image processing to be able to successfully achieve the facial recognition functionality of the ProLock. However, RPis are relatively expensive and power hungry. Therefore, carrying out the processing on a cheaper or more efficient multi-core single-board computer would be very beneficial to ProLock's commercial success, so long as the latency of authentication is not greatly affected.

6.5 Improved Locking Mechanism:

Even though the servo we used worked well, manufacturing our own servo designed by a mechanical engineer, for example, would help significantly increase the physical security of ProLock.

6.6 PCB Design and Fabrication:

ProLock was manufactured on a protoboard and wire wrapped by hand, which was a very labor-intensive and error-prone process. Therefore moving this design over to a PCB would help improve it significantly, as it would make it smaller, lighter, and cheaper, allowing us to reflect these cost savings onto consumers.

7. Cost Analysis

Cost Analysis of ProLock		
Item	Quantity	Cost ($)
Raspberry Pi 4	1	80.00
Adafruit 4651 - “Rugged Panel Mount Fingerprint Sensor”	1	22.00
‘LM3940’- Voltage Regulator	1	2.00
‘COM-14662’ bare metal keypad	1	5.00
Miuzei 20KG Servo Motor High Torque RC Servo	1	14.00
Parallax 28033 - “PIR Mini”	1	11.00
Adafruit 3968 - Speakers	2	10.00
‘DFPlayer Mini MP3 Player’	1	2.00
RPi Camera Module V2	1	30.00
Crystalfontz - “CFAH2004AC-TMI-EW” 20x4 LCD	1	10.00
Gebildet Plastic Reed Switch	1	1.00
Push buttons	2	1.00
Total		188.00

From the cost breakdown, it can be seen that the cost can be considered reasonable for a secure and smart security system. Additionally, the RPi makes up about 40% of the total cost, which provides an insight into where future price reductions could be made, such as finding an alternative for the use of an RPi.

8. Engineering Standards

8.1 SPI (ISO) Standard:

Our product's adherence to the ISO/IEC standard for Small Computer System Interface (SCSI) Part 115: Parallel Interface-5 (SPI-5), outlined in ISO/IEC 14776-115:2004, underscores our commitment to industry-recognized protocols. By leveraging the SPI protocol as the communication backbone between our Raspberry Pi (RPI) and RFID scanner, we ensure seamless data exchange and compatibility within our system architecture. This standardization not only guarantees robustness and reliability but also facilitates interoperability, enabling our solution to integrate smoothly into diverse hardware environments.

8.2 HTTPS IETF (Internet Engineering Task Force) Standard:

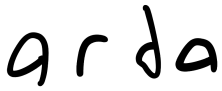
TLS 1.3, as specified in RFC 8446, represents a significant advancement in secure communication protocols, offering improved security, performance, and privacy features compared to its predecessors. Within our system architecture, we leverage HTTPS, which relies on TLS 1.3 for establishing secure connections between users and our server. This ensures that sensitive data exchanged during user interactions remains encrypted and protected from unauthorized access. Through HTTPS, users can seamlessly interact with our server, initiating requests and receiving responses, which are then relayed to the Raspberry Pi (RPI) for processing.

9. Signature Sheet

Project Portion	Riad Alasadi	Arda Caliskan	Tyler Chen
System Design	33.33%	33.33%	33.33%
Component Selection	33.33%	33.33%	33.33%
Hardware Design	40%	40%	20%
Software Design	20%	40%	40%
Documentation	33.33%	33.33%	33.33%
Project Report (oral)	33.33%	33.33%	33.33%
Project Report (written)	33.33%	33.33%	33.33%

Initials:

Riad Alasadi: 

Arda Caliskan: 

Tyler Chen: 

10. User Manual

10.1 Getting Started:
First, make sure the device is powered by 5V and that both the Atmega328P and RPi are on. Next, walk up to the ProLock - you should have the device greet you with a “Hello There!”. This greeting will be activated anytime motion is detected, with a 20 second cooldown period.

10.2 Authentication Validation:

There are four types of authentication validations: passcode, facial recognition, fingerprint, and RFID. Depending on the validation threshold selected on the website, you can complete any one of these validation options to unlock the device. The threshold number is displayed on the LCD in the following format: “#auth4”. Press the red button to attempt to verify your face and/or fingerprint. After pressing this button, an image of you will be captured, and the fingerprint scanner will begin flashing blue to indicate that the fingerprint scanner is live and waiting to read. Both of these authentications are activated with the red button, but it is not necessary for both to be completed for validation. The RFID and passkey can be attempted at any time. Upon successful validation where the number of correct authentications completed meets the minimum threshold set, the door will unlock. If nothing is attempted for 5 seconds then the device will lock itself and all authentications will go back to being invalid.

10.3 Failed Authentication Validation:
If any authentication method is failed 3 times, then it is considered an intrusion attempt and the device will play the sound “access denied” three times. Additionally, an image of the intruder will be taken and stored in the website for your viewing.

10.4 Locking door
To lock the door, simply close the door and the door should automatically shut. Make sure that the door, when closed, is near the small magnet on the bottom of the cabinet floor.

10.5 Doorbell:
The doorbell is the blue button. When it is pushed the ProLock initiates a doorbell sound and will take a picture of the guest who pushed the button. The website will show the picture along with an option to accept or deny the guest. If you accept the request, the door will unlock. If not, then the door will still stay closed.

10.6 Website Interactions:

The website allows you to change the password, change levels of authentication, and add your fingerprint and face. To change the password, simply fill in the textbox titled “password” and click submit. In order to add your fingerprint, press the button titled “Add new TouchSens”. To change the levels of authentication, fill in the text box titled “Select Authentication Methods” and press the submit button.

10.7 Touch Detection:

If at any time anybody tries to open the door by placing their hand on the door handle while the door is still locked, then this will result in the ProLock identifying them as an intruder. Hence, the “get off my door” sound will be played, and an image of the person will be taken and stored in the website.

11. References

Adafruit, “GitHub - adafruit/Adafruit-Fingerprint-Sensor-Library: Arduino library for interfacing to the fingerprint sensor in the Adafruit shop,” *GitHub*. https://github.com/adafruit/Adafruit-Fingerprint-Sensor-Library

AdityaX, “GitHub - 1AdityaX/mfrc522-python: The mfrc522-python library is used to interact with RFID readers that use the MFRC522 chip interfaced with a Raspberry Pi.,” *GitHub*. https://github.com/1AdityaX/mfrc522-python

Weber, A. (n.d.). *EE 459Lx - Embedded Systems Design Laboratory*. USC EE 459LX. https://ece-classes.usc.edu/ee459/

Face-recognition. PyPI. (n.d.). https://pypi.org/project/face-recognition/