

Smart Lock Security System

Group 22



Group Members:

Ava Raymond

Colin Slunecka

Nick Trammell-Jamison

Roman Yoder

13 December 2019

Table of Contents

Design Summary.....	2
System Details.....	4
Functional Diagram.....	7
Logic Diagram.....	10
Design Evaluation.....	11
Partial Parts List.....	16
Lessons Learned.....	20
Appendix A.....	24
Appendix B.....	25
References.....	40

Design Summary

The Smart Lock Security System secures a standard walk-through hinged door that would ordinarily be located at the front of the house. The system is designed to lock the door when specific parameters are met. It is comprised of three major subassemblies: a deadbolt installed in a door frame, a control box installed in the wall next to the door, and a sensor board placed in a garage. These components can be seen below in Figures 1 and 2. There are three main parameters that will determine when the door is locked and only one parameter for unlocking the door. The three locking parameters include a numeric keypad, a photoresistor, and an ultrasonic sensor. The door will unlock when the correct numeric combination (eg. 1234) is pressed on the 12-digit keypad and the pound key (#) is then pressed. In addition, the system will lock the deadbolt if it goes from light to dark outside or if a vehicle departs from the scaled-down garage stall. Similarly, the door can be locked directly from the keypad by pressing the star (*) key.

A speaker plays tones when numbers are pressed on the keypad as well as distinct sounds for correct and incorrect numeric sequence entry and when the lock actuates. The LCD display shows the current status of the system and the keypad numbers that have been pressed. Additionally, the LED strip surrounding the control box will initiate as various colors depending on the status of the system.

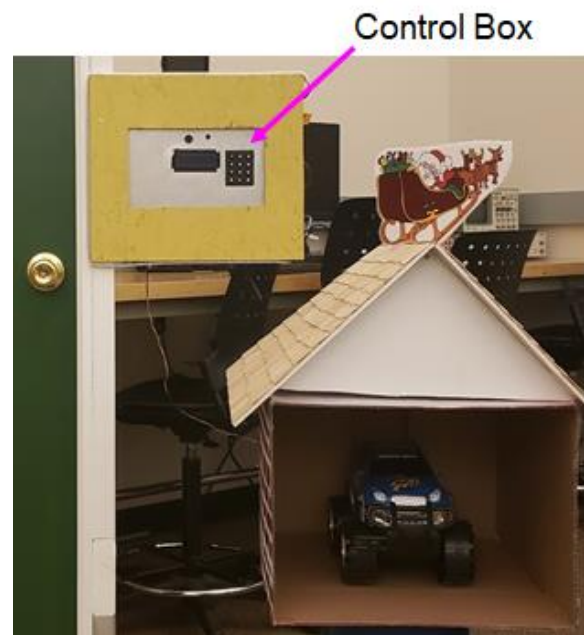


Figure 1: System Overview (Front View)

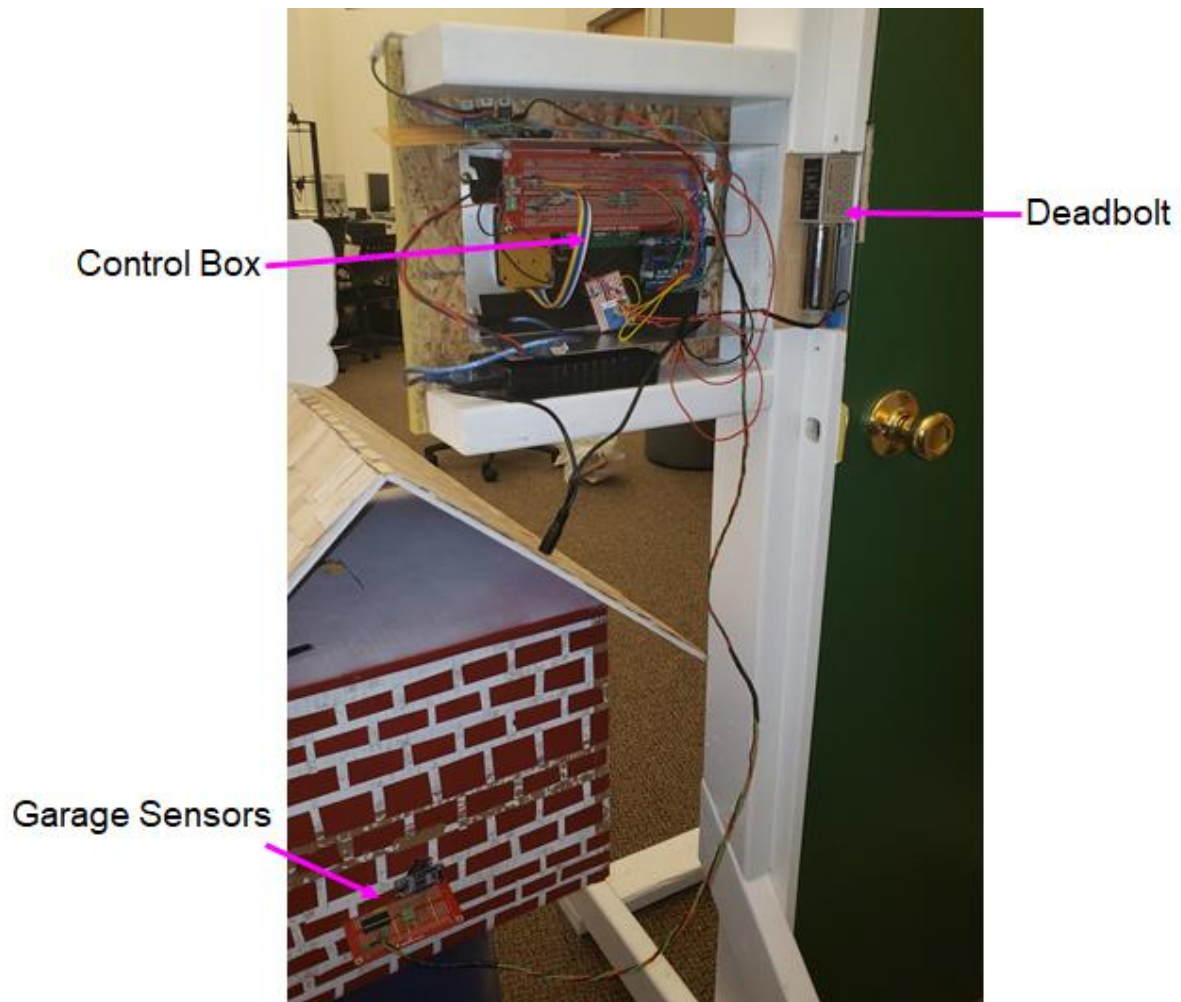


Figure 2: System Overview (Rear View)

System Details

The main components of the Smart Lock Security System's control box can be seen below in Figures 3, 4, and 5. The back cover of the aluminum box has been removed and some components have been moved outward for clarity. Nylon screws and spacers were used to ensure that there is no electrical pathway between the components or to the box itself. Screw terminals and pin headers on the circuit boards ensure that all wires will have a solid connection.

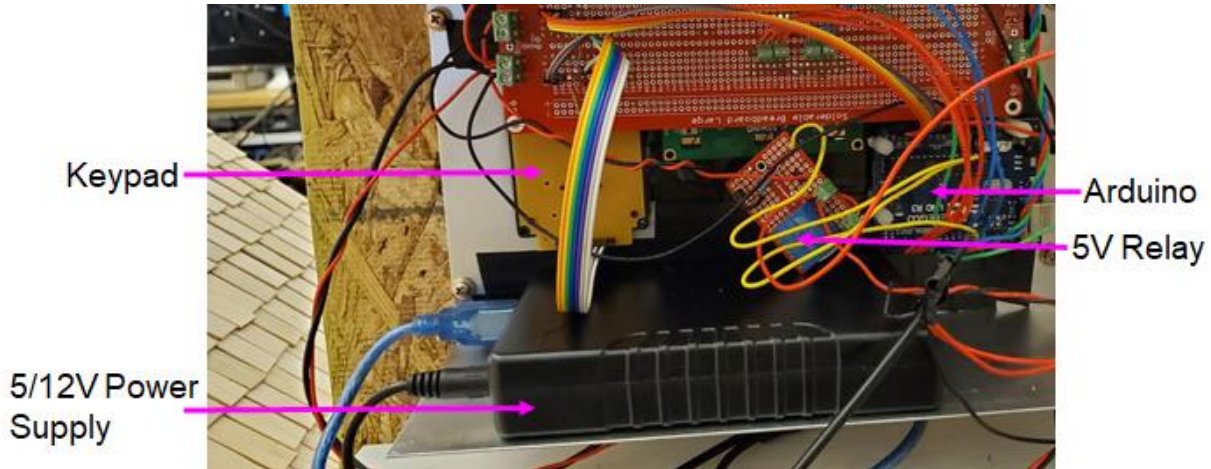


Figure 3: Internal Components

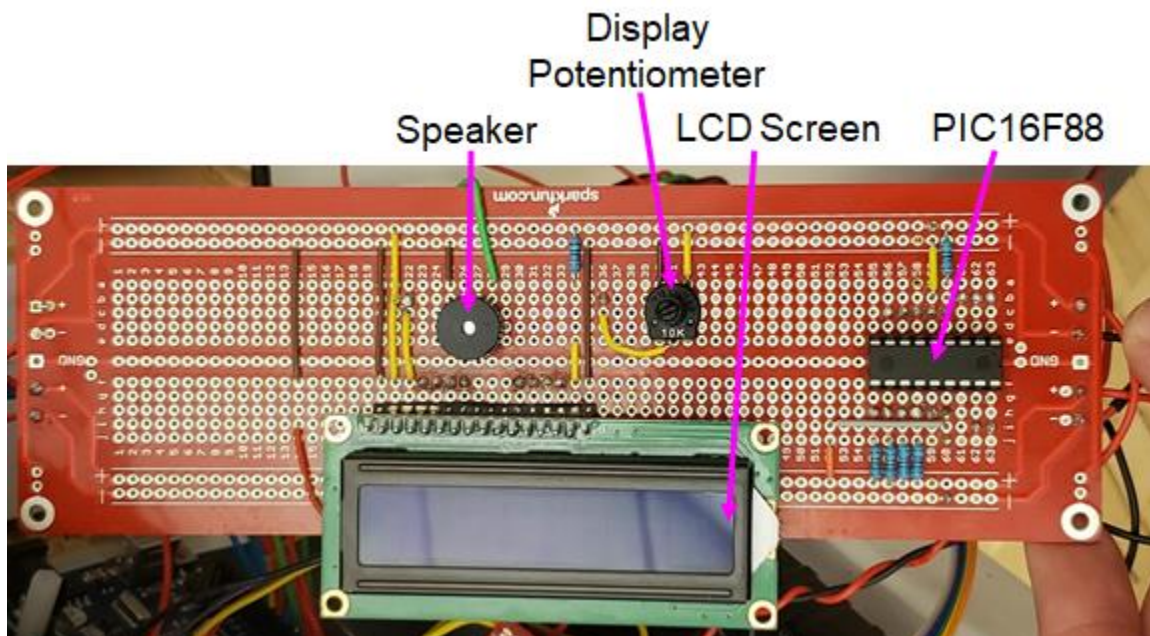


Figure 4: Main Display Board

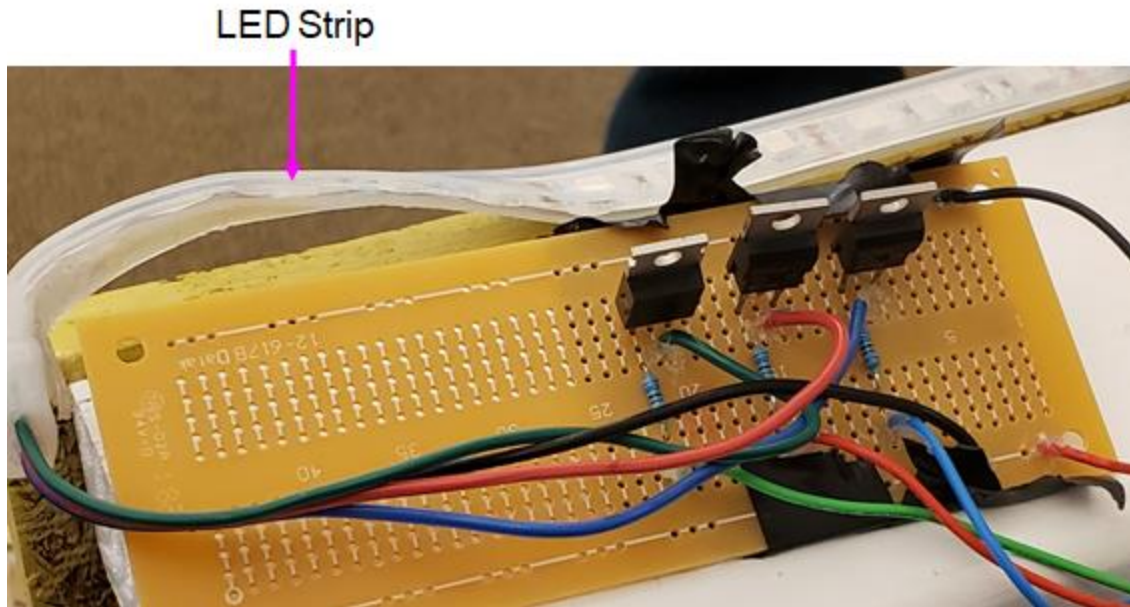


Figure 5: Transistor Bank Controlling LED Strip

The solenoid deadbolt is installed into the frame of the door while it's corresponding magnetic faceplate is mounted to a notch in the edge of the door shown below in Figure 6. The deadbolt extends just over 0.5 in., ensuring ample overlap between the lock and the door.

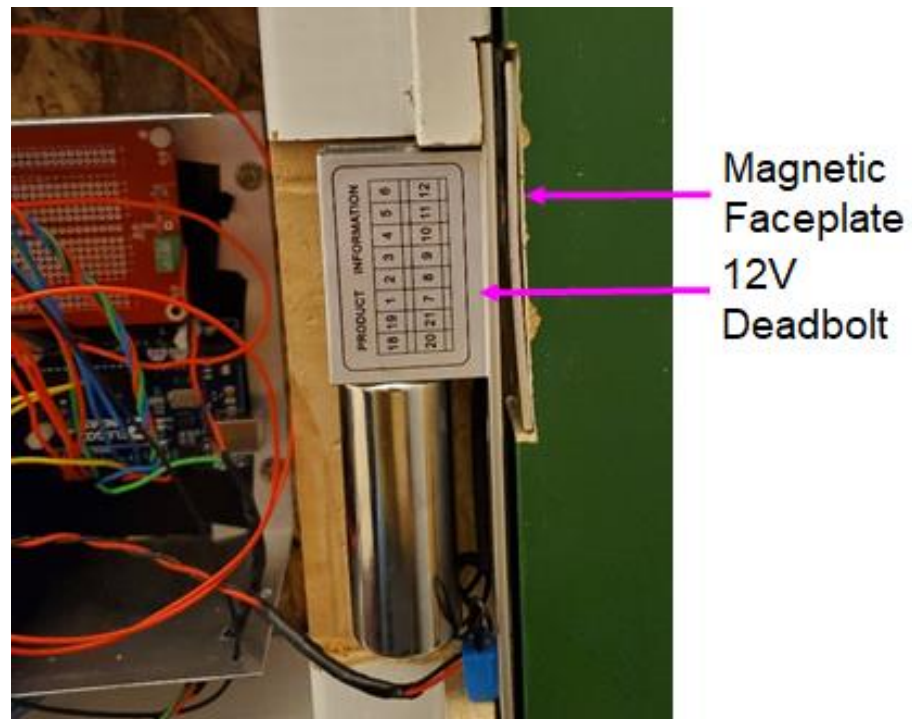


Figure 6: Deadbolt Installed in Door Frame

The garage sensor board was installed on the exterior wall of the model garage with the ultrasonic sensor protruding through the wall into the garage as shown in Figure 7. In a full scale application this board would be housed in some kind of container and either be mounted in the ceiling or within the wall of a real garage with long enough wires for the photoresistor to be exposed to the outdoors. A wiring diagram for every component of the system can be found in Appendix A.

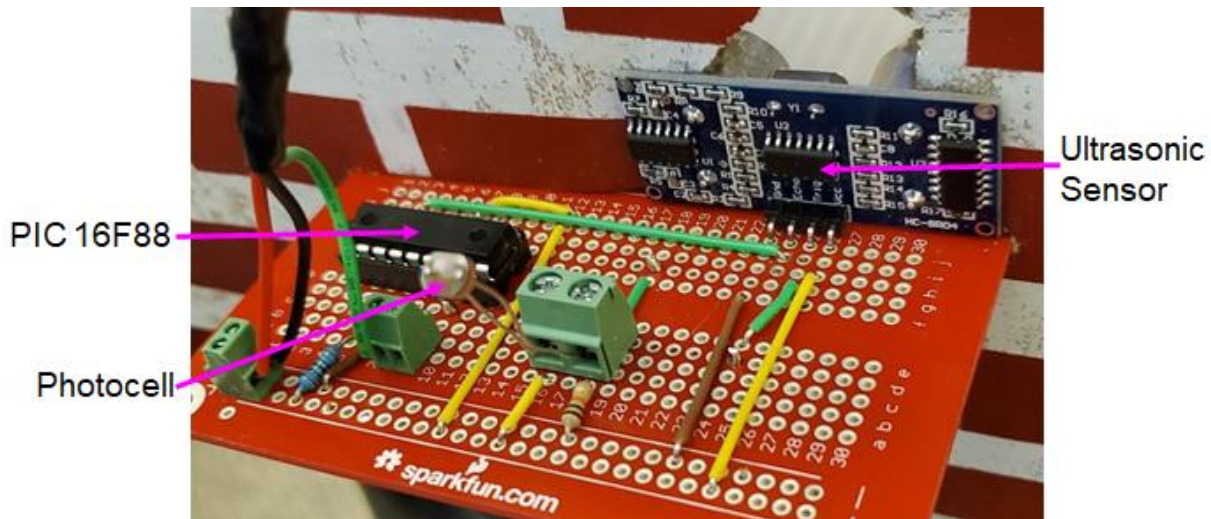


Figure 7: Sensor Board on Exterior Garage Wall

Operation of the Smart Lock Security System is very simple. The overall flow of information through the system can be seen below in Figure 8.

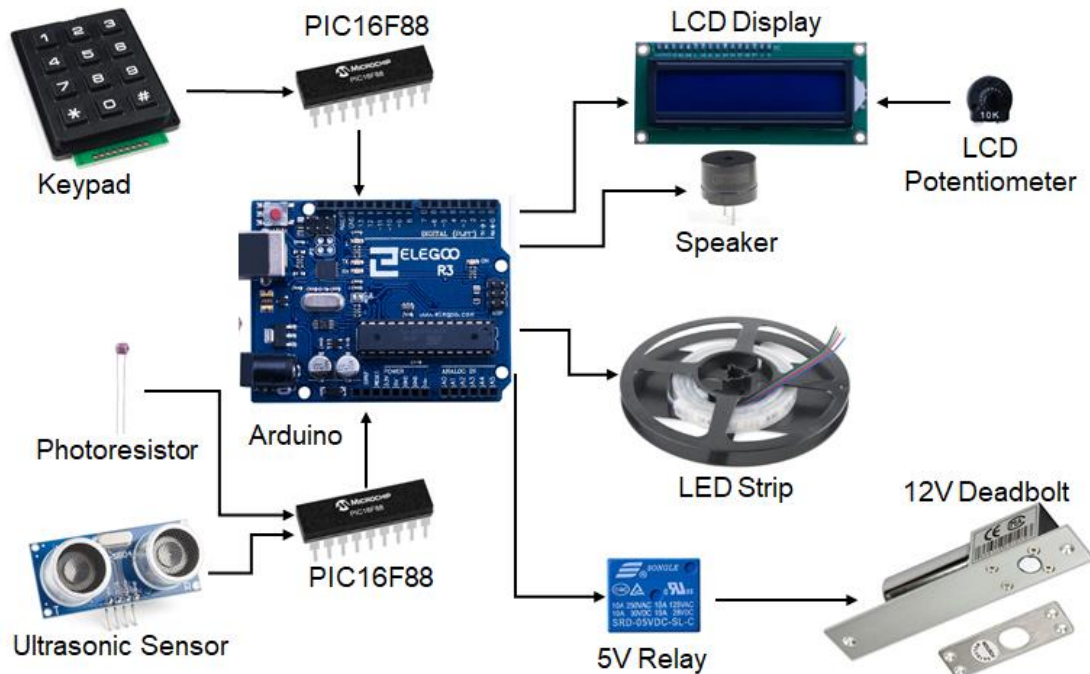


Figure 8: Functional Diagram

On startup, the door will automatically lock. At any point during operation, the star (*) button can be pressed to lock the door. When this happens, the LCD will display “LOCKING”, the speaker will sound, the LED strip will turn blue, and the deadbolt will actuate the lock. The screen will appear as it does in Figure 9. The screen will then clear itself and wait for further inputs.



Figure 9: Example Status Display on LCD Screen

The user interface for the system is shown below in Figure 10. The potentiometer controls the contrast of the LCD screen, and all audio and visual indicators are easily noticeable to the user.

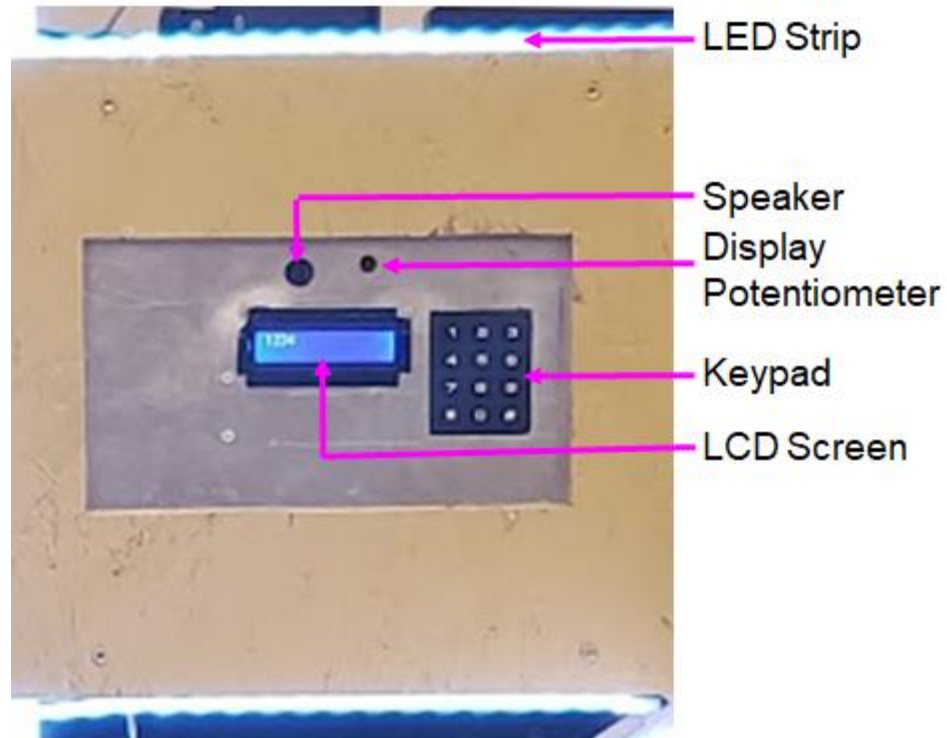


Figure 10: User Interface

The keypad is the only way to unlock the door in this system. Inputs to the keypad are sent to the first PIC16F88 and converted to a binary signal and trigger, through the use of a polling system, via five outputs to the Arduino. The PIC code to interpret the keypad inputs can be found in Appendix B. As buttons are pressed, the trigger is detected and they appear on the LCD with a tone sounding each time. After four buttons have been pressed, the pound button (#) must be pressed to enter the code. If the passcode is correct the LCD will display “UNLOCKING”, the speaker will sound a unique chime, the LED strip will turn green, and the door will unlock. If the passcode is incorrect the LCD will display “INCORRECT CODE”, the speaker will play a different chime, the LED strip will turn red, and the door will remain locked. The screen will then clear after a moment and the LED strip will return to white.

The automatic sensors for the system are connected to the second PIC16F88 that acts as an analog to digital converter and pulse width reader to send a high signal to the Arduino if it becomes dark outside or if a car is detected leaving the garage. The PIC code for the sensor PIC can also be found in Appendix B. Additionally, these sensor inputs will only lock the door provided that the door is already unlocked, otherwise, the sensors will remain inactive. Once either of the sensors has triggered the door lock, they will not be read by the Arduino until it has received an unlocking keypad code while the sensor pic is outputting a low signal. The logic

diagram showing how the entire system works together can be seen in Figure 11. Ultimately, all inputs and outputs flow through the Arduino Uno and the Arduino code can be found in Appendix B. Several of the output wires from the Arduino are connected to the gates of transistors since the Arduino does not output a high enough current to activate the necessary components.

The power source for the system is a rechargeable 5/12 V dual output battery. Unplugging the power supply from the wall does not cause any sort of system interruption as it seamlessly switches to battery mode. The battery is rated for 6000 mAh. Since the largest current draw in the system is the deadbolt at 450 mA, it is estimated that the system should run for approximately twelve hours in the event of a power outage. Similarly, the LED strip and the deadbolt both operate off of the 12V battery output. A 5V relay is used to switch on and off the deadbolt while also isolating the 12V power supply from other components. The LEDs use a row of MOSFETs with each MOSFET triggering the 12V supply necessary for each color. If all three transistors are active, white light will appear at the LED. The red, green, and blue MOSFETs can be initialized in any combination to show a variety of colors.

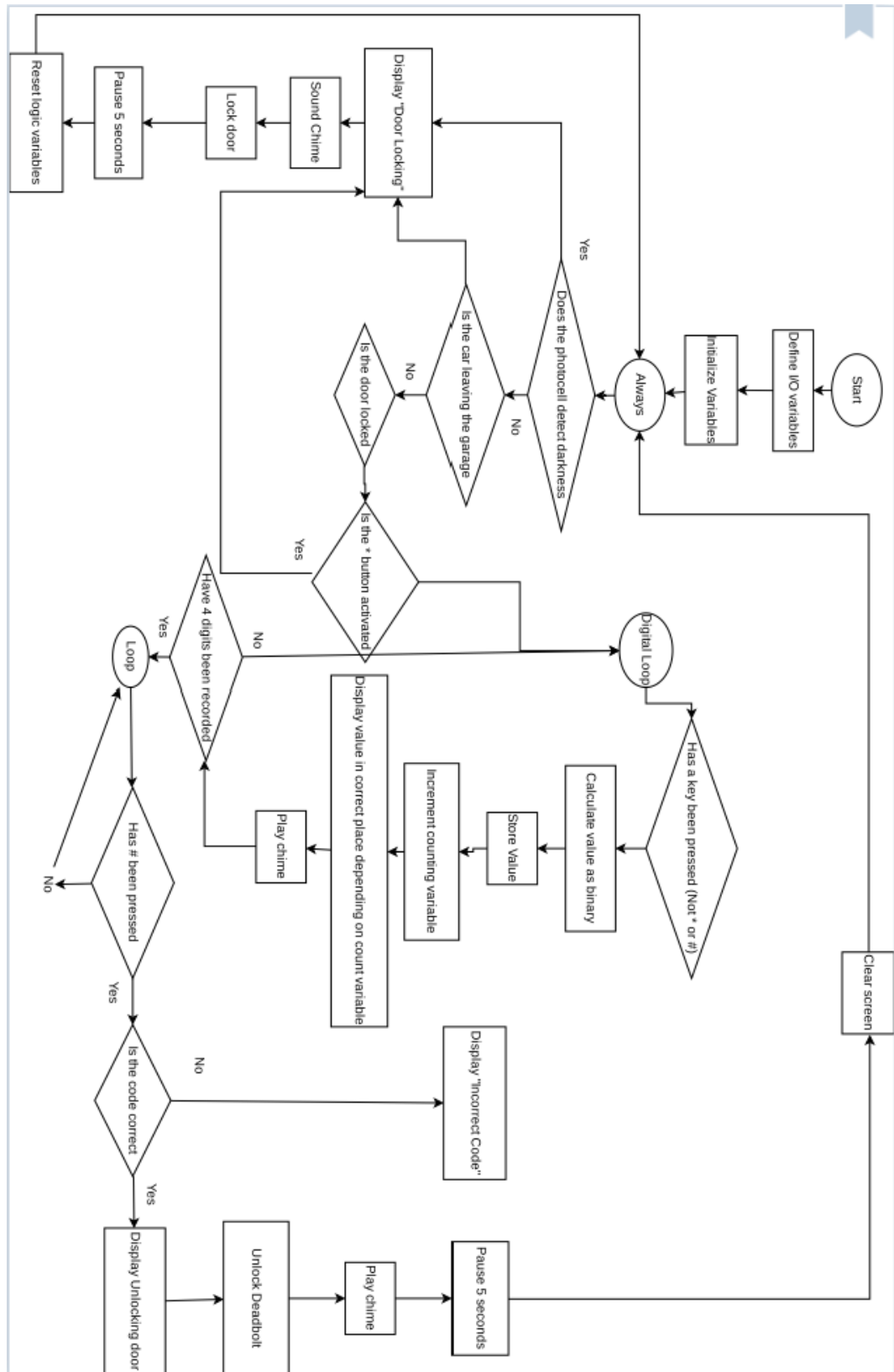


Figure 11: Logic Diagram

Design Evaluation

The following sections outlines the performance of the Smart Lock Security System in terms of the functional categories specified in the grading criteria. The components mentioned within each respective category required extensive research and effort when integrating the overall Smart Lock design. In addition, key features of each component are discussed as well as any additional qualitative characteristics present in the system. Many of the components went through several iterations in terms of software and circuitry until they each functioned reliably; then they were compiled into one fully functioning system.

Category A: Output Display

1. LCD Screen

- Custom screen sequences that indicate overall system status
- Interfaced with an Arduino Uno
- Direct means of user feedback
- Backlit display controlled by a potentiometer

For the output display, an LCD is interfaced with a numeric keypad. The LCD primarily serves as direct feedback between the user and the security system; a convenience for the user to verify the entered code. The LCD's purpose is to display the numbers being entered on the keypad and then display either "LOCKING", "UNLOCKING", or "INCORRECT CODE" depending on whether the entered code is correct or the star button is pressed to lock the door.

2. LED Strip

- Interfaced with an Arduino Uno
- Powered by 12V battery pack
- Seamlessly integrated with LDC readout
- Appealing use of colors for enhanced user experience

An RGB LED strip borders the control box. The LED strip is an additional component to improve overall user experience; acting as an extension of the LCD display. When the system is turned on, white light will always be present so that the user can see the display and keypad. The LED strip then turns blue when locking, green when unlocking, red with an incorrect code, and then returns to white when the LCD screen clears.

Category B: Audio Output Device

1. Speaker with software controlled frequency tones

- Distinct tones for each type of input (single key press, Unlocking, Locking, Incorrect Code)
- Serves as a backup feedback system for the user in the event that the LCD display and LED strip are damaged
- Seamlessly integrated with LCD readout

To further improve user experience, a small speaker is implemented in the system. This speaker is integrated with the keypad so that it makes a distinct noise after each key press. This simple sound will provide the user with an auditory experience as it ensures when a key is pressed. In addition, the speaker will emit unique sounds when the code is entered correctly, incorrectly, or when the door locks.

Category C: Manual User Input

1. 12-Digit Keypad

- Provides intuitive operation
- Effective interfacing between controlling PIC and the Arduino
- Provides tactile feedback for the user

One of the main components of the security system is the tactile numeric keypad. This keypad has four rows and three columns consisting of numbers 0-9 as well as '*' and '#.' In order to unlock the door, a series of predetermined numbers must be entered into the keypad in the correct order. The keypad is the primary means of user input as the door cannot be unlocked without the keypad.

Category D: Automatic Sensor

1. Photoresistor

- Reliable use of photoresistor to detect varying levels of light sensitivity
- Controlled with a PIC and interfaced with an Arduino Uno
- Low cost and unique means of improving Smart Lock functionality

- Ideally, A/D conversion would have been used, though, due to time constraints this function was limited

When considering user needs, it was apparent that implementing a feature that would automatically lock the door at night would be beneficial. Thus, a photoresistor is used to detect when it becomes dark outside, which then triggers the locking mechanism. This photoresistor is placed on the exterior of the model garage. Ideally, the sensitivity would be determined from testing various darkness levels outside (different times in the evening), therefore, as an improvement of the system, this feature would be implemented. Given the complexity of integrating the entire Smart Lock system, there was not enough time to implement this feature.

2. Ultrasonic Sensor

- Low cost sensor that provides unique functionality to the system
- Controlled with a PIC and interfaced with an Arduino Uno
- Used to determine vehicle presence

In addition to a photoresistor, having a system that will detect the presence of the user's automobile was desired to increase the security systems versatility and usability. To accommodate this desired feature, an ultrasonic sensor was placed in the user's garage. This ultrasonic sensor is placed on the garage front wall and is able to communicate when the user's automobile has left the garage and, therefore, is not home. If the user's car is not in the garage, the system automatically locks the main door.

Category E: Actuators, Mechanisms and Hardware

1. Solenoid driven deadbolt

- Utilizes a relay to isolate 12V required to operate the deadbolt
- Magnetic faceplate prevents the door from locking when opened
- Efficient design and implementation reduced complexity and cost

To effectively implement the security system, a locking mechanism was essential. This locking mechanism not only had to be strong enough to withstand forced entry, but it also needed to be able to integrate with every other fundamental component of the system. The lock is a deadbolt that is actuated with a solenoid. When the logic conditions are met, a current runs through the solenoid, thereby causing the deadbolt to move into position. Once the correct code

is entered on the keypad, the lock will turn “OFF” and a spring will retract the deadbolt. A magnetic faceplate on the door aligns with a normally open magnetic switch inside the deadbolt mechanism. This prevents the deadbolt from extending if the door is not fully closed. Furthermore, the deadbolt is active high so that if power is lost it will automatically unlock itself.

2. Aluminum Project Box

To effectively implement all the elements of this security system, an aluminum project box was fitted within a wall section next to the door. The box was cut and drilled to mount the LCD, keypad, main board, and Arduino with nylon screws as well as having openings for the speaker and potentiometer. The goal was to seamlessly incorporate all circuitry, except the deadbolt installed in the door frame and the photoresistor and ultrasonic sensor on the garage, with no protruding wires or components.

Category F: Logic, Processing, Control and Miscellaneous

1. 2x PIC16F88

- A/D converter utilized for the sensors
- Keypad entry interpreted with polling
- Seamless integration of PIC’s and Arduino Uno
- Effective use of programmed logic

Two PIC16F88’s were chosen due to the number of inputs and outputs necessary in addition to the location of components. The first PIC interprets the signals from the keypad, through the use of polling, and sends a binary output as well as a trigger to the Arduino. The purpose of the trigger is to let the Arduino know if one digit is pressed twice. The second PIC is on the garage sensor board and acts as an A/D converter for the photoresistor and pulse length reader for the ultrasonic sensor. This allowed for using a minimal number of wires to connect the garage sensors to the Arduino as one output was used for both sensors and power simply had to be provided to the sensor board.

2. Arduino Uno R3

- Effectively performs logic
- Handles the systems control and data processing

The Arduino Uno is the master controller for the system. It receives inputs via both PICs and sends output signals to the speaker, LCD, LED strip, and the lock. Almost every available pin is used on the Arduino. As the Arduino was the central controller it had the most complicated logic as it had to control all inputs and outputs; this is evident in the lines of code represented in Appendix B.

Overall Performance

Each unique component of the system operates as intended. The system as a whole is effectively integrated, with no redundancies, or other faulty components. Due to a simplified wiring diagram, each circuit consisted of short wires and electrical components, all soldered into place. These soldered circuits created a reliable system, with little to no errors. In addition, the software is efficiently dispersed among the various microcontrollers, thereby reducing hardware and complexity.

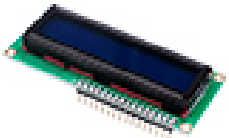



Though some aspects of the security system were demonstrated in lecture and lab, a lot of research and troubleshooting was required to achieve the desired result. Additionally, a custom door and scaled garage were created to better present the functionality of the security system. Overall, the Smart Lock Security System employs simplified user functionality while maintaining a clean design at half the cost of market competitors.





Partial Parts List




Listed below in Table 1 is a partial list of components which is organized by functional group. This partial list consists of unique and prominent components used in the Smart Lock Security System and does not list common electrical components such as resistors and capacitors.

* Indicates the price before shipping and handling costs.

Table 1: Partial Parts List

Partial Parts List					
Category	Part	Manufacturer Part Number	#	Description	*Price for one
Output Display	LCD Screen 	Elegoo LCD 1602 (from UNO R3 Starter Kit)	1	Displays keypad inputs in terms of numeric values (1234) as well as status of door lock (LOCKING, UNLOCKING, etc.).	\$5.99
	LED Strip 	Yetda Industry Ltd. 5060BRG4	1	Displays white light constantly when power is on. Displays blue when deadbolt is being locked, green when door is being unlocked, and red when an incorrect code is inputted.	14.95
Audio Output Device	Mini Speaker 	Sparkfun COM-07950 RoHS	1	Indicates user input in keypad by producing a single tone. Also outputs different tones based on entered keypad code or locking state.	\$1.95
Manual User Input	12 Button Keypad 	Sparkfun COM-14662	1	User inputs a predetermined code (4 successive numbers) and then presses the “#” key to unlock the door. If the code is incorrect, the door remains locked. The “*” key locks	\$4.50

				the door and resets the inputted code.	
Automatic Sensor	Mini Photocell 	Sparkfun SEN-09088	1	If unlocked, deadbolt is activated when it becomes dark outside.	\$1.50
	Ultrasonic Distance Sensor 	Sparkfun SEN-15569	1	Indicates when a car is not in the garage which then activates the deadbolt so that the door is locked.	\$3.95
Actuators, Mechanisms, Hardware	NC DC 12V, Electric Drop Bolt Door Lock Deadbolt Strike Fail-Safe Mode 	Zoter EDL-DB200	1	The main locking mechanism that is controlled with a built in solenoid. It is attached to the door frame and includes a magnetic faceplate to prevent the door from locking when not aligned.	\$24.99
	Aluminum Mini-box 10 x 6 x 3.5 in. 	Bud Industries Inc. CU-3010-A	1	Houses a majority of the components	21.68
Logic Processing, Control, and Misc	Arduino Uno	Arduino 8058333490137	1	Handles logic, control and data processing for entire system. Integrates with seperate PIC microcontrollers.	\$20.90

					
	 PIC16F88	Microchip PIC16F88	2	One of the PIC's controls the inputs from the keypad. The other PIC controls the photoresistor by converting analog inputs to digital and reads the pulse width of the proximity sensor. Both PIC's are integrated with the Arduino Uno.	\$2.60
	 12V/5V Battery	TalentCell YB1206000-USB	1	Supplies backup power to the entire system in the event of a power outage. The 12V output will be used to power the deadbolt and the 5V output will power the rest of the system.	\$33.99
	Total Parts Cost				\$137.00

The actual total cost of the Smart Lock Security System was less than listed due to many free components. Components such as the ultrasonic distance sensor, photocell, Arduino Uno, and LCD screen were all taken from provided lab materials. Thus, the out of pocket cost of manufacturing and all purchased parts (including soldering boards, fasteners, and other miscellaneous items) was **\$98.73**.

The overall price for the Smart Lock Security System is very competitive in relation to other automatic locking doors that are on the market. The generic locking door on the market does not include any sort of display screen or any additional features besides a keypad such as sensors. A lower end door lock costs roughly \$90.00 while a higher end model costs around \$300.00. Examples of higher and lower quality locks are shown below in Figures 12 and 13.



Mechanical Keyless Door Lever Set

Product #: MPBL15 | Finish: Satin Nickel

\$89.00 ★★★★★ [1 review](#)

QTY:

1

In Stock: 78

ADD TO CART

Figure 12: Low Cost Market Competitor



Click image to open expanded view

by Yale

Yale BAU-NTB620-NR x 626 nexTouch Keypad Lock, Capacitive Touchscreen, No Radio, Aluminum

[See all 2 in this Product Family](#)

[3 answered questions](#)

Save \$15.27

Price: **\$302.49** ✓prime

New (3) from **\$299.00** + FREE Shipping

Product Specifications

Part Number	BAU-NTB620-NR x 626
Number of Items	1
UPC	049306846899
Brand Name	Yale
EAN	0049306846899
Item Weight	7.08 pounds
Material	Aluminum
Model Number	BAU-NTB620-NR x 626
Style	Capacitive Touchscreen

Figure 13: High Cost Market Competitor

Lessons Learned

Group availability/scheduling

Initially when working on our project and the deliverables, we ran into difficulties with aligning our schedules. We found ourselves meeting two or three people at a time and utilizing a group text to communicate updates and send pictures. This worked very well for the majority of the project. In the last month of the semester, we all sacrificed a lot toward working as much as we could on this project. Even if one lab partner had only a small window of time, they came to the lab to help troubleshoot and work on whatever needed to get done. We utilized everyone's strengths within the project tasks to get things done well and efficiently.

For future students, we would recommend communicating as much as possible within the group. Knowing where the project is at and what needs to get done next makes time in the lab much more efficient as less time is wasted figuring out the status of the system.

LCD Issues

At many points while working on the project, our LCD screen would stop working. One of the most common issues was that our screen would completely stop displaying. The rest of the system would work with no issues while our LCD would not display a single command. It was usually a connection issue so we would look over the wiring diagram to ensure everything was connected correctly. In addition, wiggling and pushing on the connection between the female ports on breadboard and the pins soldered onto the LCD fixed this issue. Another issue we faced was the LCD showing incorrect letters and symbols instead of the correct message. This issue usually happened when one of the inputs into the Arduino from the PIC was loose or incorrect. Efficiently arranging components and circuit boards to reduce ground loops also helped resolve the problem.

For future students, the main lesson learned is to not get defeated when one component stops working and instead start troubleshooting. Our project got to a point where everything was set up perfectly, but it randomly would all stop working. It was usually a simple fix that took some time and messing around, but if you just keep troubleshooting you will eventually figure out the issue. One of the main keys to

troubleshooting is to immediately verify voltage values at all areas of the problematic circuit and compare them with the desired value. Often, one component had gone bad or a small wire had lost contact. These issues can be difficult to spot but are easy to fix once identified. Additionally, be conscious of electromagnetic interference (EMI) fields when integrating all components, as such fields can interfere with the systems performance.

Analog vs Digital Sensor Issues

The last components that were integrated into the system were the ultrasonic sensor and the photoresistor. The ultrasonic sensor needed to be calibrated with the height of the scaled down garage with and without a toy car in it, so it was put off until the garage model was made. We had lots of success with the proximity sensor during tests using a clipboard and holding it at different distances from the sensor, using the measured pulse width. When it was installed into the ceiling of the garage and calibrated, it stopped working correctly because the signal was not bouncing off the soft cardboard correctly. It was then installed into the back wall of the garage to try and resolve this issue. There were still issues after the change in location that we assumed to be from the sensor code, but we ran out of time to pinpoint where the error was coming from. Fortunately, the proximity code ended up working during the class presentation with no change to it's code. Issues with the photoresistor also developed two hours before the lab presentation. Although the analog code for the photoresistor had performed well in the past with high accuracy, issues with its accuracy had been identified due to it not being compared to a value with enough bits and being left justified. The code was then updated but it was never tested as the team opted to switch to a digital control for simplicity and performance reasons. Following this, it was identified that a burnt out photoresistor was creating issues in the system. The photoresistor was replaced and then used with the digital version of the code.

Some advice for future students is to use the simplest code possible. Even if you know a more intricate code that could work, it will usually cause more issues than a simplified code would. Our analog code was initially working, and began failing when it was integrated, while the simpler digital signal worked with the integrated system.

Integrating

Most of the issues with the project happened while integrating all of the components together. The relay, speaker, and deadbolt integrated well with the Arduino. The majority of the issues were with the wiring and connections. We continuously tested each circuit on the solderless breadboards, which had many connection issues, until the wiring was perfect for the soldered breadboards. The computer program Fritzing was used to create the wiring diagram. The only connection issues we ran into after we soldered the breadboards were poor connections in the LCD socket and a few misplaced resistors. In the installation process there were some spontaneous errors that were potentially due to the metal box used for our housing unit. Covering the inside with electrical tape seemed to fix most of the random errors. Further, issues from the relays on the main soldered breadboard arose. To solve this a flyback diode and pull down resistor were added to the control.

Advice for future students is to figure out how one component works at a time and then integrate it into the system. It is smart to start with the hardest part such as the LCD and the keypad. If they stop working while integrating them, or if an already integrated component stops working, it is always smart to remake it on the solderless breadboard by itself and test it with some indicator such as a single LED. As more components were added to our system, it became obvious that transistors were needed as the Arduino could not send enough power through all of its outputs.

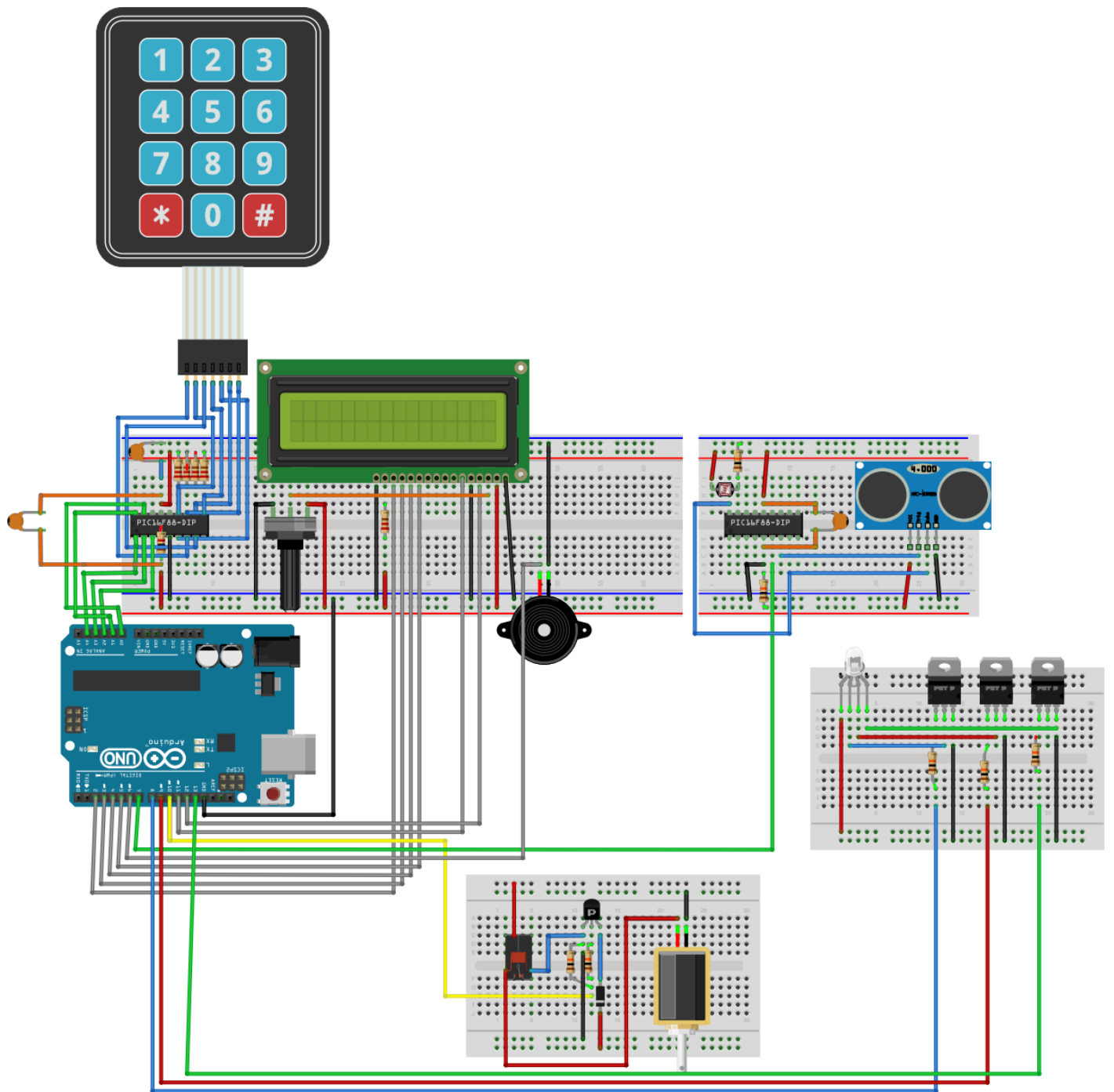
A note on Perseverance and Patience:

At the end of the day it came down to the group's willingness to succeed. At times, components and circuits would work for no apparent reason. Individuals frequently spent late nights in the lab trying to get circuits to work. When something does not work, it can be incredibly frustrating but you have two choices: to let that frustration get to you and cloud your judgement or to take a deep breath and approach the problem with a positive attitude.

Advice for future students would be to not give up. When things do not work and frustration builds, take a step back, take a deep breath, and try to approach the problem from a different angle or ask other students for advice. You can get the project to work

you just have to be willing to make the sacrifice to put in the hours and the research. At several points our group thought we were not going to make it or demonstrate a working project, but we learned to collect ourselves and pull through.

Appendix A: Detailed Wiring Diagrams



fritzing

Appendix B: Code

Arduino Code

/*

LiquidCrystal Library - Hello World

Demonstrates the use a 16x2 LCD display. The LiquidCrystal library works with all LCD displays that are compatible with the Hitachi HD44780 driver. There are many of them out there, and you can usually tell them by the 16-pin interface.

This sketch prints "Hello World!" to the LCD and shows the time.

The circuit:

- * LCD RS pin to digital pin 12
- * LCD Enable pin to digital pin 11
- * LCD D4 pin to digital pin 5
- * LCD D5 pin to digital pin 4
- * LCD D6 pin to digital pin 3
- * LCD D7 pin to digital pin 2
- * LCD R/W pin to ground
- * LCD VSS pin to ground
- * LCD VCC pin to 5V
- * 10K resistor:
- * ends to +5V and ground
- * wiper to LCD VO pin (pin 3)

// include the library code:

#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD interface pin

// with the arduino pin number it is connected to

const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;

const int blue=8, red=9, green=13;

const int sense=7;

const int KeyCode = 1234;

// set the input code

const int locking = 10;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

// lcd set up taken from Hello World

void setup() {

pinMode(14, INPUT);

//setting pin 14 as an input, the trigger

```

pinMode(15, INPUT);           //setting pin 15 as an input, the LSD
pinMode(16, INPUT);           //setting pin 16 as an input
pinMode(17, INPUT);           //setting pin 17 as an input
pinMode(18, INPUT);           //setting pin 18 as an input,the MSB
pinMode(7, INPUT);             //setting pin 7 as an input the sensors
pinMode(10, OUTPUT);           //setting pin 10 as the lock output
pinMode(8, OUTPUT);//BLUE      //setting pin 8 as an LED output
pinMode(9, OUTPUT);//RED       //setting pin 9 as an LED output
pinMode(13, OUTPUT);//GREEN    //setting pin 13 as an LED output

}

void loop() {
  lcd.clear();
  int CodeCount = 0;           // the number of numbers entered starts as zero
  int CodeStore = 0;           // variable to store the current code inputted
  int inputVariable=2396       //the current number being outputted by the
                                keypad PIC, it starts as a number it will never be

  int J = -1;                  // determine positive edge triggering so the photo
                                sensor will not constantly actuate in the dark

  digitalWrite(locking, HIGH); //initialize lock in locking position
  digitalWrite(blue, HIGH);     //set LED to white
  digitalWrite(red, HIGH);      //set LED to white
  digitalWrite(green, HIGH);    //set LED to white
                                // set loop so it is continuously running

  while (1== 1){
    if (digitalRead(sense)== HIGH && J==1 )
//Reports that the sensor input has gone high; J can only be reset if the sensor is low when it is
//unlocked
{
  lcd.setCursor(0, 0);         // set the cursor to the top left
  lcd.print("LOCKING");        //Display locking on the LCD
  digitalWrite(locking, HIGH); //lock deadbolt
  digitalWrite(green, LOW);    //display blue on LED
  digitalWrite(red, LOW);      //display blue on LED
  tone(6, 329, 250);           //play locking tone
  delay(250);
  tone(6, 415, 750);
  delay(100);
  tone(6, 261, 250);
  delay(1000);
  CodeCount = 0;               //Reset logic variables
  CodeStore = 0;
  lcd.clear();                 //Clear LCD screen
  J=-1;                        //set J to negative so it can not continually actuate
}
}

```



```

    digitalWrite(green, HIGH);           //set LED back to white
    digitalWrite(red, HIGH);             //set LED back to white
}

if (digitalRead(14) == HIGH){           // if the trigger goes high enter this if statement
    delay(100);                          // allow the numeric inputs to set
    inputVariable= digitalRead(15)+digitalRead(16)*2+digitalRead(17)*4+digitalRead(18)*8;
                                         //set the input variable to correspond to the input values
    delay(100);
    if (inputVariable == 10){            //star has been pressed and is being reported
        lcd.setCursor(0, 0);            // set the cursor to the top left
        lcd.print("LOCKING");           //display locking on the lcd display
        digitalWrite(green, LOW);       //display blue
        digitalWrite(red, LOW);         //display blue
        digitalWrite(locking, HIGH);    //lock the deadbolt
        // lock the lock set the lights
        tone(6, 329, 250);              //play chime for locking
        delay(250);
        tone(6, 415, 750);
        delay(100);
        tone(6, 261, 250);
        delay(1000);                   // pause for 1 second to show the screen print out
        CodeCount = 0;                 //reset the logic variables
        CodeStore = 0;
        inputVariable=2396;
        lcd.clear();                  //clear LCD
        J=-1;                         //set J to negative so it can not continually actuate
        digitalWrite(green, HIGH);     //set LED back to white
        digitalWrite(red, HIGH);       //set LED back to white
        // clear lcd for future use
    }else if (CodeCount == 0 && inputVariable!=10 && inputVariable!=12){
        // enter this when the first digit has been pressed
        //and the digit actuated is not # or *
        CodeStore= 1000 * inputVariable; // Stores the variable in the thousands place of
        //code store.
        CodeCount = CodeCount+1         //increments code store by one for the next number
        //to be pressed

        lcd.setCursor(0, 0);
        lcd.print(inputVariable, DEC);  //displays first number pressed in the first place
        tone(6, 400, 250);              //have speaker beep once

    }else if (CodeCount== 1 && inputVariable!=10 && inputVariable!=12){
        // enter this when the second digit has been
        // pressed and the digit actuated is not # or *

```

```

CodeStore= CodeStore+ (100* inputVariable);          // store the pressed number in the
                                                    //hundreds place
CodeCount= CodeCount+1;          // increment code count to 2 for the 3rd number
lcd.setCursor(1, 0);          //place cursor in position for the second number pressed
lcd.print(inputVariable, DEC);          //display second number pressed
tone(6, 400, 250);          //play tone

}else if (CodeCount== 2 && inputVariable!=10 && inputVariable!=12){
                                                    // if the third number is being pressed and it is not # or *
CodeStore= CodeStore+ (10* inputVariable);          // Store the number in the tens place
CodeCount= CodeCount+1;          // increment for the next number
lcd.setCursor(2, 0);
lcd.print(inputVariable, DEC);          //display third number pressed
tone(6, 400, 250);          //play tune

}else if (CodeCount== 3 && inputVariable!=10 && inputVariable!=12){
                                                    // if the fourth number is being pressed and it is not # or *
CodeStore= CodeStore+ (1* inputVariable);          //store the number in the ones place
CodeCount= CodeCount+1;          // increment for the enter pound symbol place
lcd.setCursor(3, 0);
lcd.print(inputVariable, DEC);
tone(6, 400, 250);          //play tone

}else if (CodeCount== 4 && inputVariable==12){
// if four numbers have been entered and pound is pressed enter the unlocking and
//incorrect code statements
if (CodeStore == KeyCode){          // if the code is correct
lcd.setCursor(0, 0);
lcd.print("Unlocking");
digitalWrite(blue, LOW);          //display green
digitalWrite(red, LOW);
digitalWrite(locking, LOW);          //unlock deadbolt
tone(6, 261, 200);          //play tone
delay(200);
tone(6, 329, 300);
delay(300);
tone(6, 261, 750);
if (digitalRead(7)==LOW){
J=1;          //Reset J for the positive edge trigger on the photo sensor
}

}else {          // if the code does not match then it is incorrect
lcd.setCursor(0, 0);
lcd.print("INCORRECT CODE");

```

```

digitalWrite(blue, LOW);
digitalWrite(green, LOW);
tone(6, 1500, 200);
delay(300);
tone(6, 1500, 200);
}
delay(1000); // show the screen for five seconds then clear and reset variables
CodeCount = 0;
CodeStore = 0;
inputVariable=2396;
lcd.clear();
digitalWrite(green, HIGH); //set LED back to white
digitalWrite(red, HIGH);
digitalWrite(blue, HIGH);
}
}
}
}

```

Keypad PIC Code

'Set the oscillation speed to 4 MHz

DEFINE OSC 8

OSCCON.4 = 1

OSCCON.5 = 1

OSCCON.6 = 1

'Turn off the A/D converter

ANSEL = 0

Row1 Var PORTB.7

Row2 Var PORTB.6

Row3 Var PORTB.5

Row4 Var PORTB.4

Col1 Var PORTB.2

Col2 Var PORTB.1

Col3 Var PORTB.0

'Define inputs and output variables

EightPlace Var PORTA.2

FourPlace Var PORTA.3

TwoPlace Var PORTA.4

OnePlace Var PORTA.1

' Disable PORTB pull-ups

OPTION_REG = \$FF

'Initialize the I/O (RB7: RB4 and RB3 as inputs and RB2: RB0 as outputs)

TRISB = %11111000

TRISA = %00000000

Define what's an input and output

' Keypad polling loop

Low EightPlace: Low FourPlace: Low TwoPlace: low OnePlace

Mainloop:

 ' Check column 1

 Low Col1: High Col2 : High Col3

 If (Row1 == 0) Then

 'Key 1 is down

 Low EightPlace: Low FourPlace: Low TwoPlace: High OnePlace

 EndIf

 If (Row2 == 0) Then

 'Key 4 is down

 Low EightPlace: High FourPlace: Low TwoPlace: Low OnePlace

 EndIf

 If (Row3 ==0) Then

```

        ' key 7 is down
        Low EightPlace: High FourPlace: High TwoPlace: High OnePlace

    Endif
    If (Row4 == 0) Then
        ' Key * is down
        High EightPlace: Low FourPlace: High TwoPlace: Low OnePlace

    EndIf
    ' check column 2

    High Col1: Low Col2: High Col3
    If (Row1 == 0) Then
        ' Key 2 is down
        Low EightPlace: Low FourPlace: High TwoPlace: Low OnePlace

    EndIf
    If (Row2 == 0) Then
        ' Key 5 is down
        Low EightPlace: High FourPlace: Low TwoPlace: High OnePlace

    EndIf
    If (Row3 ==0) Then
        ' key 8 is down
        High EightPlace: Low FourPlace: Low TwoPlace: Low OnePlace

    Endif
    If (Row4 == 0) Then
        'Key 0 is down
        High EightPlace: Low FourPlace: High TwoPlace: High OnePlace

    EndIf
    ' Check column 3

    High Col1: High Col2: Low Col3
    If (Row1 == 0) Then
        'Key 3 is down
        Low EightPlace: Low FourPlace: High TwoPlace: High OnePlace
    EndIf
    If (Row2 == 0) Then
        'Key 6 is down
        Low EightPlace: High FourPlace: High TwoPlace: Low OnePlace
    EndIf
    If (Row3 ==0) Then

```

```
        ' key 9 is down
        High EightPlace: Low FourPlace: Low TwoPlace: High OnePlace
    Endif
    If (Row4 == 0) Then
        'Key # is down
        High EightPlace: High FourPlace: Low TwoPlace: Low OnePlace
    EndIf
    Goto Mainloop
End
```

Digital Sensor PIC Code

'Set the oscillating speed to 4MHz

DEFINE OSC 8

OSCCON.4 = 1

OSCCON.5 = 1

OSCCON.6 = 1

'Turn off the A/D converter

ANSEL = 0

BRIGHTOUT Var PORTB.0

LIGHTIN var PORTA.0

pulseout var PORTB.1 'trig

echoin var PORTA.1 'eco

a VAR WORD

'Define the output and input variables

' Disable PORTB pull-ups

OPTION_REG = \$FF

'Initialize the I/O

TRISB = %00000000

TRISA = %11111111

'Define Inputs and Outputs

Mainloop:

low BRIGHTOUT

'Set the Led low to begin to not trigger the lock

if (LIGHTIN == 0) Then

' if the photoresistor goes low trigger the output

 HIGH BRIGHTOUT

 pause 200

 low BRIGHTOUT

endif

high pulseout

pauseus 10

low pulseout

'Send a pulse out to operate the proximity sensor

PULSIN echoin, 1, a

'Measure the incoming pulse from the proximity sensor

a = a/580

'Convert the incoming pulse width to cm

if (a > 17) then

'If the distance is greater than 17 cm trigger the output


```
high BRIGHTOUT  
pause 200  
low BRIGHTOUT  
endif
```

```
Goto Mainloop  
End
```

Original Analog Sensor PIC Code

```
'Set the oscillation speed to 4 MHz
DEFINE OSC 4
OSCCON.4 = 1
OSCCON.5 = 1
OSCCON.6 = 1

DEFINE ADC_BITS 8
DEFINE ADC_SAMPLEUS 50
define ADC_CLOCK 3
ANSEL = %00000001
ADCON0= %11000101
ADCON1= %01000000
'Turn on and define the digital to analog converter and set to left justified

'ANSELH = %00000000
TRISA = %11111111
TRISB = %00000000
'Set the inputs and outputs
lockout Var PORTB.0
adval VAR BYTE
pulseout var PORTB.1 'trig
echoin var PORTA.1 'eco
low lockout
a VAR WORD
'Define the input and output variables as well as the storage variables
Mainloop:

ADCIN 000, adval
'Store the analog input in adval
if (adval <= %00111111 ) Then
'Compare the analog input so when it goes low the output is turned high
HIGH lockout
pause 100
low lockout
endIF
high pulseout
pauseus 10
low pulseout
'Send a pulse to run the proximity sensor
PULSIN echoin, 1, a
'Measure the return pulse from the proximity sensor
```

```
a = a/580
'Convert the pulse width to centimeters
if (a > 17) then
  'If the value is greater then 17 cm turn on the trigger
    high lockout
    pause 100
    low lockout
  endif

Goto Mainloop
End
```

New, Untested Analog Sensor PIC Code

'This code is exactly the same as the above code except the analog input was changed to be
'Right biased, and the value was compared with a more accurate bit size unfortunately there
'was not time to test this code

```
DEFINE OSC 4
OSCCON.4 = 1
OSCCON.5 = 1
OSCCON.6 = 1
```

```
'Turn off the A/D converter
DEFINE ADC_BITS 8
DEFINE ADC_SAMPLEUS 50
define ADC_CLOCK 3
ANSEL = %00000001
ADCON0= %11000101
ADCON1= %11000000
```

```
'ANSELH = %00000000
TRISA = %11111111
TRISB = %00000000
```

```
lockout Var PORTB.0
adval VAR word
pulseout var PORTB.1 'trig
echoin var PORTA.1 'eco
low lockout
a VAR WORD
Mainloop:
```

```
ADCIN 000, adval
if (adval <= %00000000000111111 ) Then
    HIGH lockout
    pause 100
    low lockout
endIF
high pulseout
pauseus 10
low pulseout
```

```
PULSIN echoin, 1, a
```

```
a = a/580
```

```
if (a > 17) then
  high lockout
  pause 100
  low lockout
endif
```

```
Goto Mainloop
End
```

References

Alciatore, D., 2019, *Introduction to Mechatronics and Measurement Systems*, McGraw Hill Education, New York, NY, Chap. 7

Iovine, John., 2004, *PIC Microcontroller Project Book: for PICBasic and PICBasic Pro Compilers*, McGraw Hill Education.

- “Hello World”
 - Arduino Liquid Crystal Library
 - <https://www.arduino.cc/en/Tutorial/HelloWorld>