Bluetooth Low Energy Door Lock with Ambient Noise Number Generation

A Design Report
Presented to the Engineering Division of the Graduate School of Cornell University
in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering, Electrical and Computer Engineering

Submitted by Danielle Regis
Advised by Dr. Bruce Land
Date: May 2016

Abstract

Master of Electrical Engineering Program
Cornell University
Design Project Report

Project Title: Bluetooth Low Energy Door Lock with Ambient Noise Number Generation
Author: Danielle Regis

Abstract:
This is a door lock that can be controlled via Bluetooth Low Energy from an iPhone. The goal of this project was to include elements that would make the device more secure and explore ambient noise as a source for random number generation. When the user opens the application to open their lock they have to enter a 4-digit code. They are then brought to a screen that asks them to enter a 10-digit code within 4 seconds. This ten-digit number is sent serially over Bluetooth LE from the Arduino. Only after this number is correctly entered can the user open their lock. The total system is comprised of an Arduino Uno, Red Bear Lab Bluetooth Low Energy shield, microphone, and motorized door lock. The microcontroller handles the random number generation from the ambient noise input, and the iPhone application handles the actions taken to avoid access to the lock via a man in the middle attack.

Executive Summary

The internet of things has been rumored to be the next big thing. The internet of things is a network of physical objects embedded with electronics that enable them to collect and exchange data. The application of these devices often make our lives much easier, but with the rapid development of these instruments we often overlook security. As we start increasing the connectivity of physical devices they often become susceptible to breaches in security. Security firms like Kaspersky have shown the vulnerabilities in systems like smart homes, baby monitors, car washes, and police surveillance systems. Whether a hacker wants to wash their car for free, or watch someone else's baby – IoT security flaws make it possible.

Today's applications of IoT devices are only the tip of the iceberg in regard to its uses. In the future the IoT has the potential to help with smarter natural disaster management, smarter healthcare, and smarter urban management. Although the direction of this project didn't explore any of these applications, a door lock was chosen for this project because it's extremely critical to the safety of one's home. The novelty of this project ultimately lied in the exploration of randomly generated numbers being sourced from ambient noise. The protection of the communication, and data of IoT devices is critical thus discovering more secure ways to encrypt this information is imperative.

Today, many random seeds for pseudo-random number generators are actually not truly random. This is a problem because it leads to the creation of encryption keys that can be predicted. The search for new means to create random numbers is not a new problem, but creating random numbers with sound is. Sound was chosen because the ambient noise in a room often has no traceable pattern so it is a source that should be explored for encryption.

The objective of this MEng project is to create a Bluetooth Low Energy (LE) door lock with ambient noise number generation. The main goal of this project is to explore key generation from ambient noise, and it's degree of randomness. A microphone will be used to collect the noise in the room, and it will then be translated into a number. The number created from the ambient noise will be the seed for the pseudo-random number generator. The numbers that are ultimately generated are then statistically tested for randomness. Bluetooth LE was chosen for the communication between the phone and the door lock because Bluetooth LE requires there to be communication within a short range from the device. This is an additional layer of potential security for the system over that of a system that communicates across the Internet.

The human interface of the whole system is an iPhone application that first presents a screen where the user has to input a simple 4 digit code. The second layer of authentication ensures that the user is a human rather than a computer automated system that has cycled through all of the possible codes for entry. The application does this by getting a 10 digit audio generated random number from the door that has to be reentered by the user within 4 seconds. Only after these two steps have been passed can the user access the lock via Bluetooth Low Energy. The combination of these steps, and the accessibility restrictions set on the system by having the lock only connect to the user's device make this lock a very secure option.

Table of Contents

Introduction 6
    Background 6
    Existing Products 7
    Cost 7
    Design Problem 8
System Design 8
    System Requirements 8
    Hardware Design Choices 9
        Microcontroller & BLE Shield 9
        Lock 9
        Microphone & Motor Driver 9
    Software Design Choices 10
        Communication 10
        Application/iOS Outline 10
        Sound Analysis 11
        Random Number 11
    High Level Design of Hardware 12
    Arduino Software Code 13
    High Level Design of iPhone Application 14
iPhone App Development 16
    Bluetooth Connection 16
    Password Screen 17
    Random Number Screen 17
Results 18
    Application Tests 18
    Pseudo Random Number Tests 18
    Interference with Other Designs 21
    Usability 22
Conclusion and Future Work 22

References 23

List of Figures

Figure 1: High-Level System Design..13
Figure 2: App Password Screen Figure 3: App Random Number Screen 15
Figure 4: App Door Access Screen Figure 5: App Door Access Screen 15
Figure 6: Picture of Final System...18
Figure 7: Microphone Output..19
Figure 8: Zoomed In Microphone Output 20
Figure 9: Arduino random() With Seed Generated by Microphone 20
Figure 10: Arduino random() Without Random Seed – Same Sequence Every Time21
Figure 11: Histogram Showing Distribution of random() Values 21

Introduction

Background

The internet of things has been rumored to be the next big thing. The internet of things is a network of physical objects embedded with electronics that enable them to collect and exchange data. The application of these devices often make our lives much easier, but with the rapid development of these instruments we often overlook security. As we start increasing the connectivity of physical devices they often become susceptible to breaches in security. With the development of new IoT devices security is often overlooked and this makes these devices especially vulnerable. Security firms like Kaspersky have shown the vulnerabilities in systems like smart homes, baby monitors, car washes, and police surveillance systems. Whether a hacker wants to wash their car free of charge, or stalk someone via their fitness tracker – IoT security flaws make it possible.

Wind River published a white paper on IoT security in January 2015 and one of their main points was that it's an unrealistic expectation that it is somehow possible to compress 25 years of security evolution into novel IoT devices. Despite glaring and gaping holes in many IoT devices they continue to be released, and the world that we are living in has continued to become more connected. For instance, as recently as May 2016 it has been released that computer scientists at University of Michigan have discovered vulnerabilities in Samsung's Smart Home automation system that allowed them to carry out a host of remote attacks, including digitally picking connected door locks from anywhere in the world. Samsung's SmartThings system is one of the leading Internet of Things platforms for smart homes and the researches discovered that the attacks were made possible due to two intrinsic design flaws that are not easily fixed. Information such as this forces us consumers to think twice before using systems such as this to connect door locks and other security-critical devices. Sadly many people don't think twice because as time goes on we are becoming more and more conditioned to "trust" technology.

My design project is exploring a simple way to make an IoT door lock more secure. The human interface of the whole system is an iPhone application that first presents a screen where the user has to input a simple 4 digit code. The second layer of authentication ensures that the user is a human rather than a computer automated system that has cycled through all of the possible codes for entry. The application does this by getting a 10 digit random number from the door that has to be reentered by the user within 4 seconds. Only after these two steps have been passed can the user access the lock via Bluetooth Low Energy.

Existing Products

There are many wireless door locks out on the market today. Most of them come with one or more means of entry which include RFID, WiFi, keypad, biometric (fingerprint) or Bluetooth. Most of these locks open via app and can integrate with the user's home automation system. According to SafeWise the leading source for home security and safety advice for the average consumer the electronic door locks found below are suggested.

1. Kwikset Kevo
2. Yale Real Living Electronic Touch Screen Deadbolt
3. SoHoMiLL YL 99 Keyless Electronic Keypad Lock
4. August Smart Lock
5. Samsung Digital Door Lock
6. Schlage Camelot Touchscreen Deadbolt

Despite security concerns that have come to light with systems like Samsung's SmartThings more IoT locks continue to roll out onto the market every month. The appeal of these devices is that they never require you to copy a key again. Many offer you the ability to create virtual keys for family and guests and set windows of access for particular users. The additional functionality that these locks provide make them a great addition of many people's lives despite their questionable security.

The MEng project explored was a proof of concept of random number generation from ambient noise in the form of a relatively secure IoT door lock. Although the system may appear to not be as complex as the locks that are available on the market, by not being connected to the internet, and not having any cloud connectivity, this alternative is a very viable safe option.

Cost

One constraint for an average user is cost. The average wireless door lock on the market today costs around $200. The lock created for this project is not only more secure, but cheper.

Name	Number	Unit Cost/$
Arduino Uno Board	1	24.95
Red Bear Labs Bluetooth Shield	1	19.99
Bi-Directional DC Motor Driver Board	1	4.95
Lockitron Motorized Door Lock Body	1	44.95
Electret Microphone	1	7.95

Wire	several	N/A
AA Battery	4	1.50

The cost of the system is $108.79, but if this were to be mass-produced the price would likely be less than $75.

Design Problem

The goal of this project is to design a proof of concept system that is more secure than wireless locks on the market today. Many of the systems that are on the market are susceptible to man in the middle attacks. For instance many of the systems today can be hacked by simply scanning for open ports on the Web that are commonly used by known control systems.

The Z wave wireless protocol is particularly popular for home automation. There are millions of Z-Wave products worldwide and they can be found in thousands of hotels, cruise ships, and vacation rentals. Yet ad Def Con 19 it was shown that some of these devices were easily hacked by man in the middle attacks. Breaches like this show that security of home systems needs to be reevaluated. For the scope of this project the fundamental design problem faced was discovering where the weakest points were for wireless locks and working to eliminate them.

This deliverable will show you a system that is resistant to man in the middle and brute force attacks. The complete system will be of great interest to consumers who are interested in freedom associated with wireless locks, but are weary of the apparent security risks of many of the locks that are available on the market.

System Design

System Requirements

The main priority is to create a device that can only be accessed by the owner of the device. To do this a two-step authentication process was chosen.
First, an iPhone application presents a screen where the user has to input a simple 4 digit code. Second, a 10 digit random number from the door that has to be reentered by the user within 4 seconds is sent to the device. Only after these two steps have been passed can the user access the lock via Bluetooth Low Energy.

The following requirements for the project were outlined:
- The system shall not be connected to the Web
- The system shall be compact
- The system shall be low cost (<$200)

- The system shall not be susceptible to man in the middle and brute force attacks
- The system shall have a manual override
- The system shall be able to be powered by a battery pack
- The system shall be able to be controlled by an iPhone application
- The system shall only allow the lock to interact with the user's device

Hardware Design Choices

Microcontroller & BLE Shield

The microcontroller board being used for this project is an Arduino Uno, which is based on the ATmega328P. It has 14 digital input/output pins, 6 analog inputs, and a 16MHz quartz crystal. It was primarily chosen for this project because of its vast open source documentation, which makes it a great prototyping platform. Additionally when looking for devices that could easily interface with a compatible iOS device through Bluetooth LowEnergy (BLE), the Arduino Uno arose as the board with the most options. Ultimately, the shield that was chosen was BedBearLab's BLE Shield Version 2.1 because of its compatibility with the Arduino Uno, and Arduino Yun. Compatibility with the Arduino Yun was considered due to it being a more robust board that is great for web connected or networked projects. Possible web connectivity would be a great way to expand this project in the future.

Lock

The body of the lock chosen was that of the first generation wireless lock by Lockitron. Lockitron was one of the first wireless locks to come to the market, but after crowd funding in 2012, they had significant manufacturing, and software issues. The mountain of issues that they faced, and years of delayed shipments led them to scrapping their first generation design, and they are currently focusing all development, and sales on their second-generation lock. With the scrapping of their initial design, the body of their first generation design is available for sale through websites like Adafruit. Lockitron has stripped down the initial device, and leaves the guts of the device to be manipulated. The housing contains a geared down DC motor, a 4xAA battery holder, and the mechanical rotation mechanism and detection switches. Using this body is incredibly convenient due to the fact that it is easily fits over a thumb turn deadbolt lock, and the lack of their control board, motor driver or app, is easily replaced with alternatives more fitting for the direction of this project.

Microphone & Motor Driver

The last two components chosen for the device being built were an Adafruit TB6612 1.2A DC/stepper motor driver breakout board, and an Electret Microphone Amplier – MAX 9814 with auto gain control. The driver breakout board was chosen due to it being the driver used by those at Adafruit when they created an SMS-controlled door lock using the same housing. Knowing that they used this breakout board eliminates the question of if the driver is compatible. The microphone being used for this project was chosen because unlike most other available microphones it is not linear so the gain is not fixed. It samples audio levels over five to ten milliseconds and adjusts the gain accordingly so the sound is never too loud or too quiet. This microphone amplifies noise that is far away, and doesn't allow loud nearby noises to saturate the microphone. These qualities make this microphone ideal for capturing even subtle noises that might be present in a room.

Software Design Choices

When deciding the specifications for this project there were a few choices that had to be made ranging from the method of communication to the design of the software.

Communication

There were a few factors that went into deciding on Bluetooth Low Energy as the method of communication between the mobile device, and the Arduino Uno. Some of the alternatives considered were Classic Bluetooth, and WiFi. Bluetooth Low Energy was selected over Classic Bluetooth due to the reduced power consumption of Bluetooth Low Energy in comparison to some of the high power alternatives like WiFi. Battery life is an incredibly important factor for the scope of this project because for a device as critical as a door lock, continuous and reliable battery is absolutely necessary.

WiFi was eliminated as the means of communication due internet connectivity allowing the device to be accessed from anywhere. For all of the smart door locks currently on the market their focus is solely on providing the greatest amount of access and flexibility to their device so WiFi provides them with the freedom to do this. Considering that the focus of this smart door lock isn't flexibility, but security, the long range is actually a downfall. The device's long range and the only barrier to entry to control the device remotely being a password, leaves room for improvement if security is the main focus for the device. Bluetooth has a finite max range of about 10 meters, but in practice this range is not this large. With a shorter range it ensures that the user trying to access the lock is actually within the range. With this, an attacker trying to gain access to the lock can't do it from very far away. Also, with Bluetooth Low Energy only one device can be connected and interfacing at a time with the slave device, which in this case will be the Arduino Uno. Being that it can only connect with one device at a time, once the Bluetooth connection is established, a third party can't interfere. This is beneficial because it forces man in the middle attacks to be increasingly difficult.

Application/iOS

Rather than just using the application as a means to access the lock, there are a few layers of authentication added to boost the difficulty of falsifying credentials to the lock. The first step to being able to access the lock is the user entering the iOS application and entering their unique 4 digit passcode.

Once access has been granted to the application, the user will request access to the lock, and at this time, they will be sent a number from the lock. This number will be randomly generated every time that the user requests access to the lock. Within 4 seconds the user has to retype the number which ensures that a human user is interfacing with the lock.

The development of an iPhone application was chosen over developing an Android application for two reasons. First, I have an iPhone already so the process of finding a device enabled with BLE could be skipped. Secondly BLE was only introduced with API 18 (Android 4.3) so the ability to connect via BLE has not been around as long as on iPhone.

Sound Analysis

The decision to source the random number from a microphone, rather than some other source of randomness that could be measured by one of the Arduino's analog inputs came from thinking about what is truly random. The noises in a room, along with the introduction of gain to create distortion of the frequencies provide an arena that is unpredictable. To get the random number that will be used, a microphone will be used to send the sound to the Arduino Uno. The audio will be sampled and processed to ultimately provide frequencies that will be placed into a 16 bit number. This number will serve as the seed to the pseudo random number generator.

Random Number

For encryption there were many different choices as there has been research on the creation of pseudo random numbers, and encryption for decades. The pseudo random number generator chosen is Arduino's random() function. The issue with Arduino's random function is that it generates the exact same list of pseudo random numbers every time. An example of how the random function works is that if a slot machine were built, and the first try was a winner – then you can be sure that if you were to reset the Arduino board, and pull the handle again – you would be a winner the first time again. The way that Arduino handles this is by using the randomSeed() function. This function takes in a value and uses that number to alter the random list generated by the random() function. Due to the seed varying over time with the noise in the room, the random list created by the random() function will also vary. These functions were chosen over many other pseudo random number generators because for the purpose of this project they provide adequate randomness. The key to making the random() function provide truly random numbers is the manipulation of the random list due to the seed, and as long as the seed that is fed into randomSeed() is of adequate randomness, random() will work for this project. An example of how it is used with an alternate random source can be found below. With this example, an unconnected floating analog pin is used as the random seed.

```
long randNumber;
```

```
void setup(){
        Serial.begin(9600);
        randomSeed(analogRead(0));
}

void loop(){
        randNumber = random(300);
        Serial.println(randNumber);

        delay(50);
}
```

High Level Design of Hardware

For the embedded hardware design of the lock I chose an Arduino Uno microcontroller. The Arduino Uno is a single-board microcontroller; the hardware overall consists of a 16MHz ceramic resonator, a USB connection, an open-source hardware board based on the ATmega328, a power jack, an ICSP header, a rest button and several LEDs. Overall due to the Arduino board being a highly developed open source hardware board, there are several libraries that make programming on this board ideal for beginners. Another benefit of this board is that it does not need an additional programmer to download the code to the microcontroller, alternatively the USB and on-board ATmega16U2, which is programmed as a USB to serial converter, can be used along with the IDE to download the code to the microcontroller. For this project the microcontroller is responsible for processing the input from the microphone, and driving the Bluetooth shield that talks to the iPhone.

For the Bluetooth hardware, the Red Bear Labs BLE Shield version 2.1 is used. The chip used on the shield is the Nordic nRF8001. This shield can only support the peripheral (slave) role. The nRF8001 does not behave as a pure SPI slave device as the nRF8001 can receive new data over the air at any time or be busy processing a connection event or new data. The chip has peak currents as low at 12.5mA and average currents down to 9μA (for a 1s connection interval). This enables battery lifetimes of months to years from a single coin cell running the device. This chip's low power consumption, ADC for battery level monitoring, low tolerance 32kHz RC oscillator, 16MHz crystal oscillator, two voltage regulators, linear voltage regulator, and DC/DC voltage regulator that when enabled can further cut current consumption by up to 20% make this Bluetooth chip ideal. Even with such low power consumption, the device can still be discoverable and connectable within 10 meters, which is sufficient for this project.

A microphone is used to collect the audio from the room that the system is in. In order to maintain the integrity of the audio in the room the Electret Microphone Amplifier – MAX9814 with Auto Gain Control was acquired from Adafruit. With this board the nearby loud sounds are quieted so they don't overwhelm and clip the amplifier, and even quiet far away sounds are amplified. This makes it a great system for when audio levels can intermittently change. This allows the audio in the room to be captured in totality, which is essential to this project. This board is attached to the Arduino UNO.

The TB6612 motor controller breakout board from Adafruit is attached to both the Lockitron body and Arduino UNO. The Lockitron body completes the prototype for the system as it can actually open and close a deadbolt.

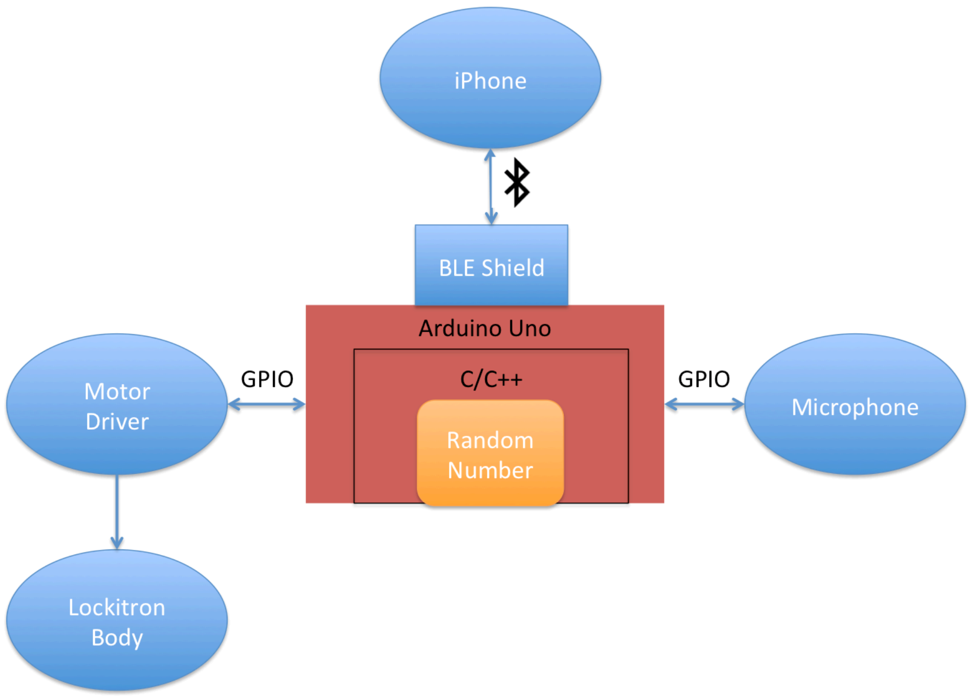

Figure 1: High-Level System Design

Arduino Software Code

By choosing the Arduino Uno and Red Bear Labs BLE Shield there were a lot of resources available that made development much easier. After installing the libraries that are provide by Red Bear Labs' Github (see reference), connecting the Arduino and application showed to be simple as the protocol to set up the connection was outlined in all of the sample applications. With the additional installed libraries the functions ble_connected() and ble_available() were available. These functions when called return true if the BLE is connected and BLE is available respectively.

When the ble_connected() is true the led on the board turns on to indicate that there has been a successful connection. While the BLE connection is available the random number is generated from reading analog pin 0 which is connected to the microphone. This value is then used in Arduino's randomSeed function. The random number is then generated using Arduino's random(). The value of ble_read() is then used within if statements to indicate when certain blocks of code should be executed. The value of ble_read is received from the iPhone application. When the iPhone application sends data0 == 0x01 the random number is sent with the support of the serial library as the Arduino treats the transmission of data via Bluetooth as a serial data connection. When the iPhone application sends data1 == 0x01 it indicates that the user should have access to the lock so the moveMotor function is called which then turns the motorized door lock to the open position.

High Level Design of iPhone Application

An iPhone application is the primary user interface for this project. The development of this application was done incrementally.

1. The fist step of app development was handling the Bluetooth connection and wireless data transmission. The Red Bear Labs examples where utilized heavily to do this development. The connection to Bluetooth is done automatically as the application is opened.

2. The second step was creating the initial screen that appears once the user opens the application. On this screen the user is prompted to enter their passcode as shown in Figure 2. If the password is correct the user is sent to the next screen.

3. The third step was creating the screen that receives the random number from the Arduino via BLE and contains a timer. On this screen the number that is transmitted, the screen for you to retype the number and timer that is counting down can be seen as shown in Figure 3. Here the numbers that are being sent serially are concatenated into one number that is displayed to be retyped. If the number is correctly typed prior to the timer running out, access to the next screen that allows the user to open the lock is granted. The final screen that allows the user to open the lock can be seen in Figure 4. If the user does not type the number correctly or the timer runs out, the screen shown in Figure 5 is displayed.

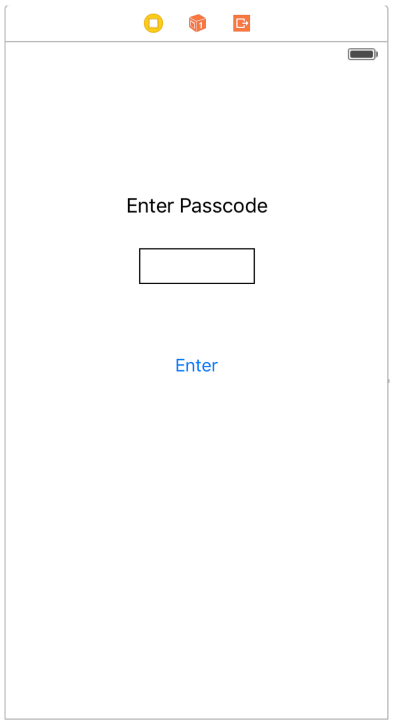

Figure 2: App Password Screen

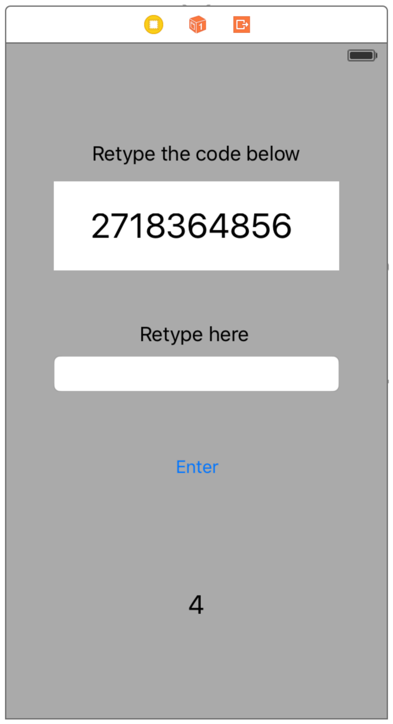

Figure 3: App Random Number Screen

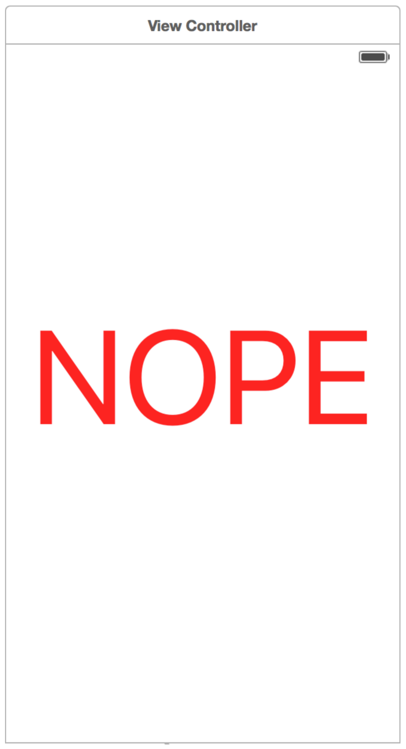

Figure 4: App Door Access Screen

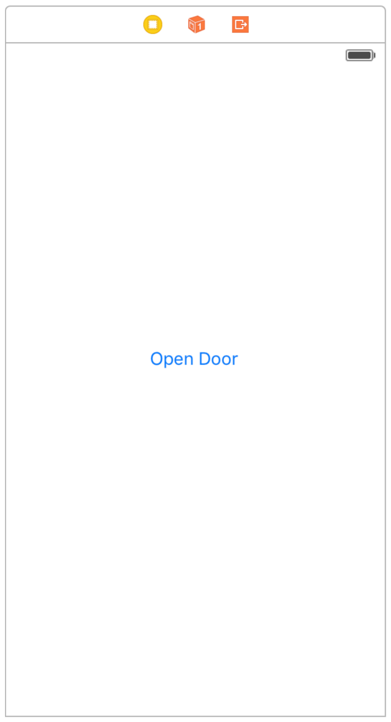

Figure 5: App Door Access Screen

iPhone App Development

Writing the iPhone application showed to be the most difficult part of this project, but with the support of the Red Bear Lab examples many difficult portions were simplified. To explain the development of the application I will go through the development of the three main components: Bluetooth Connection, Password Screen, and Random Number Screen.

Bluetooth Connection

When the view controller is loaded, the connection to the BLE board is attempted. Many of the examples used a button to indicate the actions of connecting and disconnecting the device, but to connect the device once the application is opened the code below is used.

```
self.ble = [[BLE alloc] init];
[self.ble controlSetup:1];
self.ble.delegate = self;

[self tryToConnectToBLESheild];

- (void) tryToConnectToBLEShield {
//Check core bluetooth state
if (self.ble.CM.state != CBCentralManagerStatePoweredOn)
[self waitAndTryConnectingToBLE];

//Check if any periphrals
if (self.ble.peripherals.count == 0)
[self.ble findBLEPeripherals:2.0];
else
if (! self.ble.activePeripheral)
[self.ble connectPeripheral:[self.ble.peripherals objectAtIndex:0]];

[self waitAndTryConnectingToBLE];
}
- (void) waitAndTryConnectingToBLE {
if (self.ble.CM.state != CBCentralManagerStatePoweredOn)
[self performSelector:@selector(tryToConnectToBLESheild) withObject:nil
afterDelay:0.25];
else
[self performSelector:@selector(tryToConnectToBLESheild) withObject:nil
afterDelay:2.0];
}
```

Here it can be seen that once the connection to the BLE shield is made, the application uses connectPeripheral to get the BLE shield to connect and be ready to accept commands. This portion of the development was simplified greatly due to the BLE Framework provided by Red Bear Lab.

Password Screen

With the password screen I decided to have the correct password hardcoded into the program. In ViewController.m when the view loads the password for the user is set as “1234” with self.password for the simplicity of this this prototype. The passwordTextField which is the empty box allows the user to type in their password for the system. By using NSLog to log into the console the user’s entry into the text field can be viewed. Once the user’s password is entered the enter button is pressed. The enter button’s actions are done with an (IBAction) function. A Boolean is created for the password. If the string entered into the text field is equal to the password the application moves to the next screen. If the password is incorrect, a notification label is output onto the console that says “Incorrect Password” in red.

Random Number Screen

When this view loads the application sends data to the Arduino. This is done as the view loads by putting in the viewDidLoad function. The data is in the form of a two bit buffer, and it is sent using [self.ble write:data]. The framework provided by Red Bear Labs makes the process of transmitting data very easy in iOS. Once this is transmitted it waits for data to be received. Red Bear Lab’s chat application helped greatly with learning how to handle and receive the data. The data is handled in the function bleDidReceiveData and the random number string that is surrounded by “< >” and the length of the string are the inputs. The string that is input is then redefined to remove the “< >” with stringByReplacingOccurencesOfString. This string is then saved as randomNumber and the length is kept track of. The process of taking the random number sent from the Arduino, removing the “< >” and concatenating the strings is done until the length of randomNumber is 10. This manipulation of the NSString is done until the length of the string is 10. The string data is then displayed in the text box (UITextView) with randNumTextView.text = randomNumber.

With the display of the 10 digit random number the timer starts to count down. The count down timer was implemented by using NSTimer scheduledTimerWithTimeInterval. If the counter reaches zero the screen is switched to that with a red “NOPE.” From this screen the user does not have any option to go backwards. They have to cancel out of the application and open it again if they want to try again to access the lock. If the user is able to retype the 10 digit string, and press the enter button within four seconds, the contents of the text field will be compared to randomNumber with isEqualToString. This process is similar to that done with the user’s initial password. If the contents of the text field do not equal randomNumber the user is sent to the “NOPE” screen. If the contents of the text field equals to the randomNumber, the next screen with the open door button is presented to the user. When the button is open door pressed the (IBAction) function sends the NSData with the UInt8 buf value with [self.ble write:data]. This action signals for the door to open.

Results

For this system, beyond the application functioning correctly by only allowing the user to gain entry to the lock when all tests were passed, the analysis of the pseudo random numbers used was necessary.

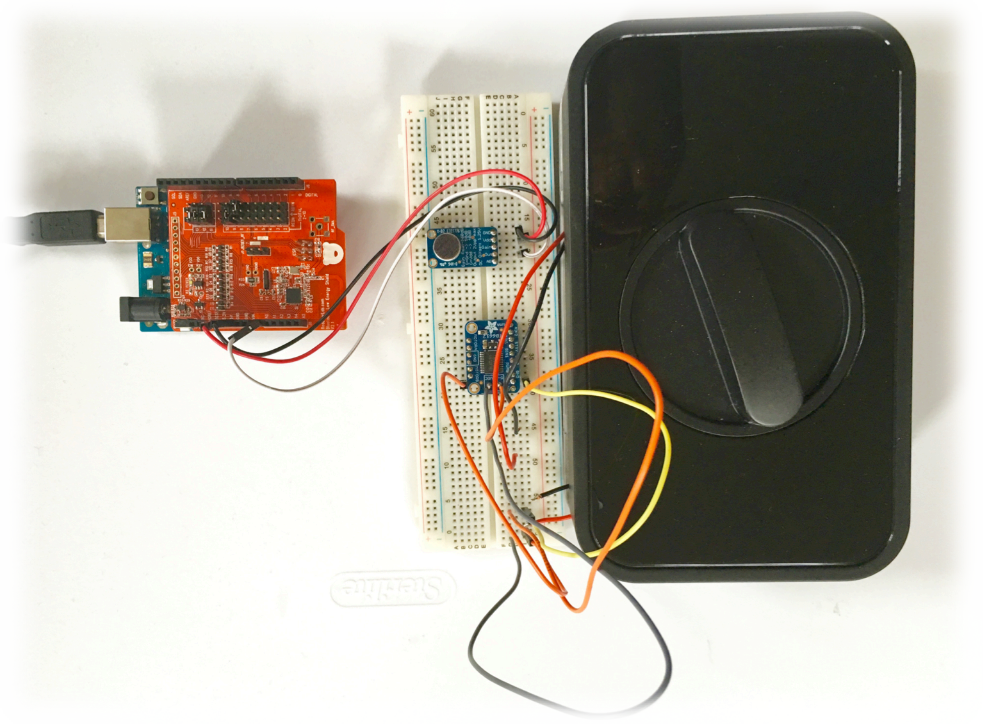

Figure 6: Picture of Final System

Application Tests

To test if the application runs correctly the actions below were done:

1. Incorrect password -> notification label "Incorrect Password"
2. Correct password -> advancement to next screen
3. Timer times out -> NOPE screen
4. Random number entered incorrectly -> NOPE screen
5. Random number entered correctly -> advancement to next screen
6. Open Door button pressed -> door opens
7. Blaring frequency into microphone -> random number differs

Pseudo Random Number Tests

A key part to this project is creating a pseudo random number from the ambient noise in the room that the door lock is in. In this section, the output of the microphone, and the random numbers generated are explored. In Figure 7 the output of the microphone can be seen. The values that are output are put into randomSeed() which is responsible for initializing the pseudo-random number generator, causing it to start at an arbitrary point in its random sequence. This sequence is always the same. In order for the sequences of numbers that are generated to differ with every execution of the sketch the input into randomSeed() needs to be fairly random. With observation of Figure 7 it may appear that there is not much randomness due to the large concentration of the values that fall between 200 and 300, but while observing the values that fall between 200 and 300 in Figure 7 the values appear to have a relatively uniform distribution.

When looking at the pseudo-random numbers generated by the system in Figure 9 it appears to have the same degree of uniform distribution as Figure 10. Despite both exhibiting uniform distribution, with each sketch the sequence of numbers for Figure 8 varies, while the sequence remains the same for Figure 10. Finally, Figure 11 displays that the numbers generated by the system show uniform distribution. If the sample of random numbers was greater than the 1000 used, the histogram would have likely shown less variation. Regardless, it is indisputable that there is a nice distribution within the bins.

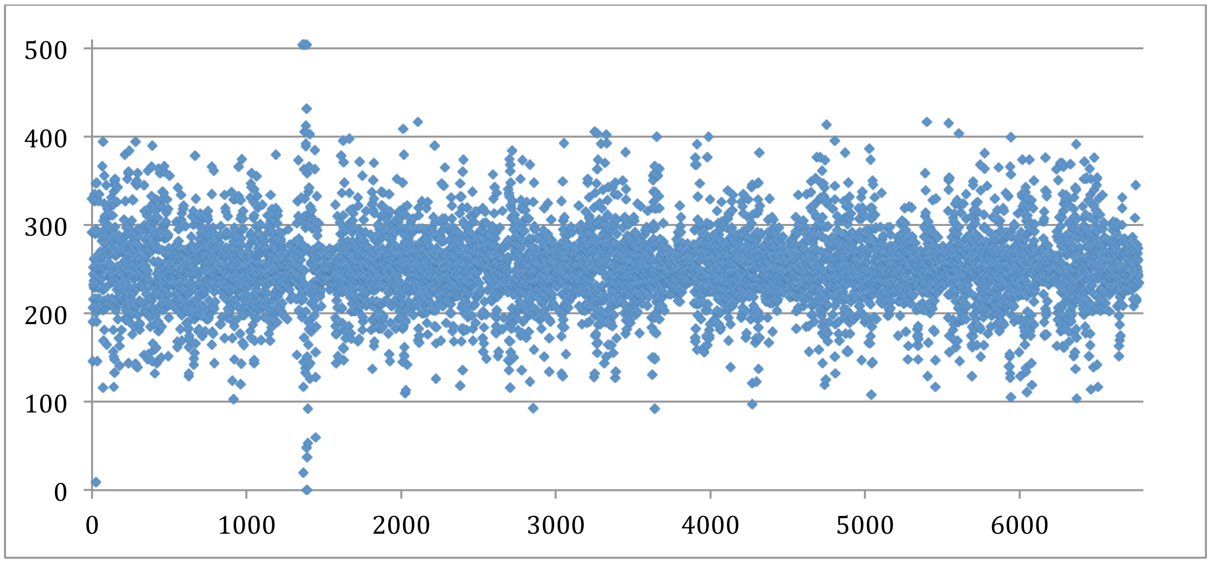

Figure 7: Microphone Output

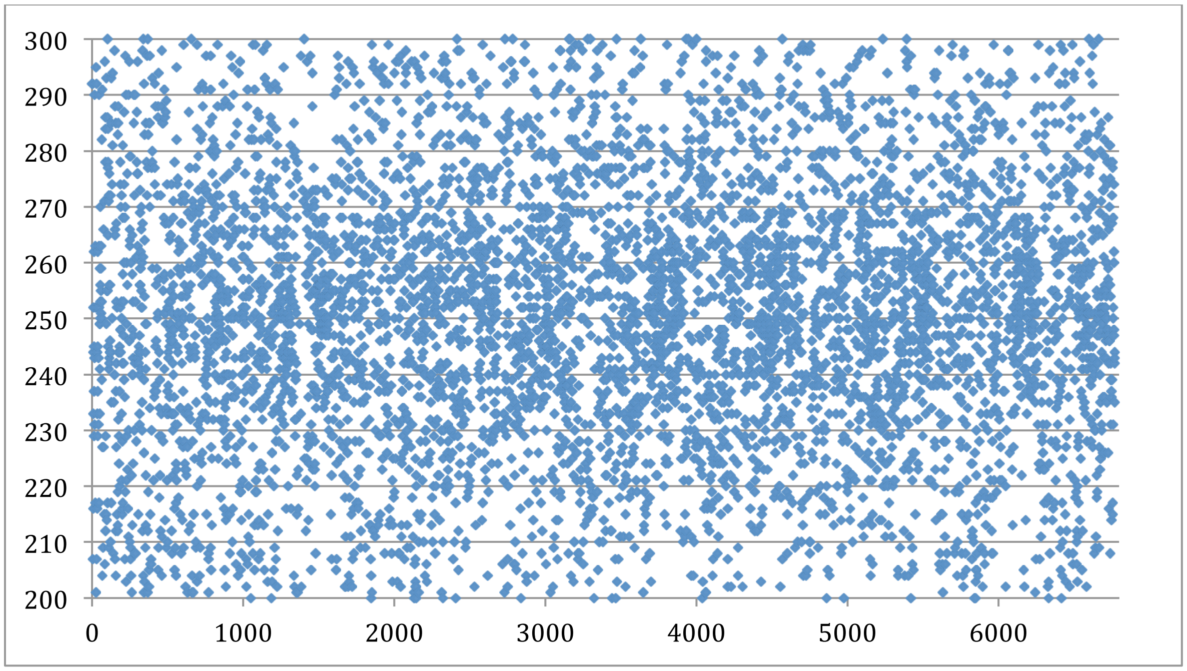

Figure 8: Zoomed In Microphone Output

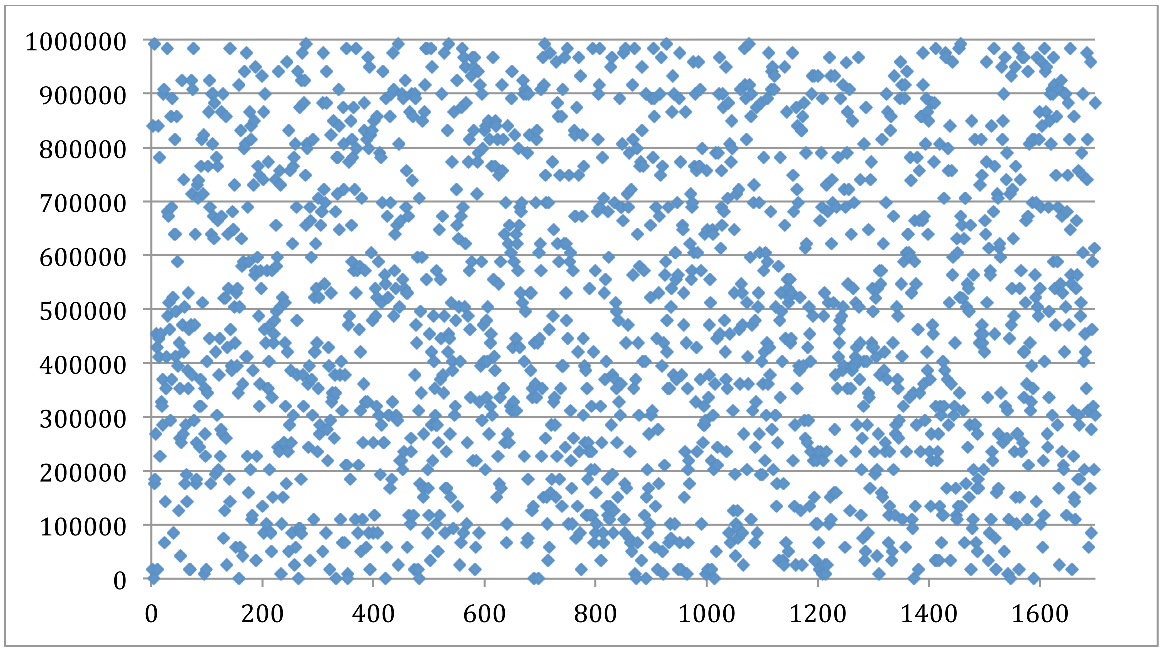

Figure 9: Arduino random() With Seed Generated by Microphone

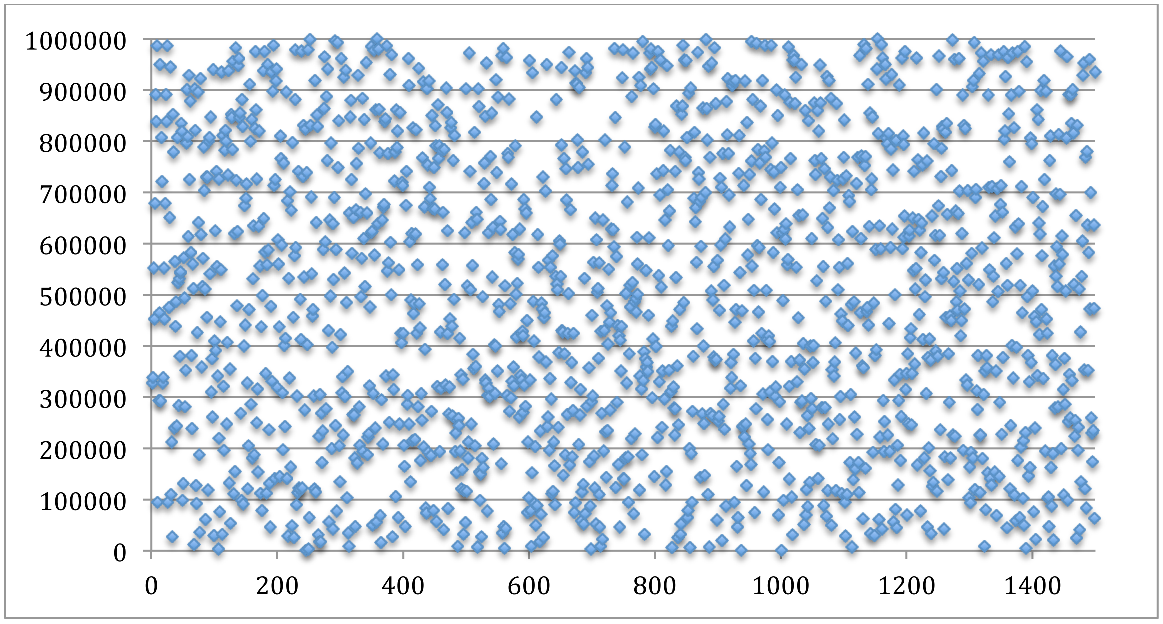

Figure 10: Arduino random() Without Random Seed – Same Sequence Every Time

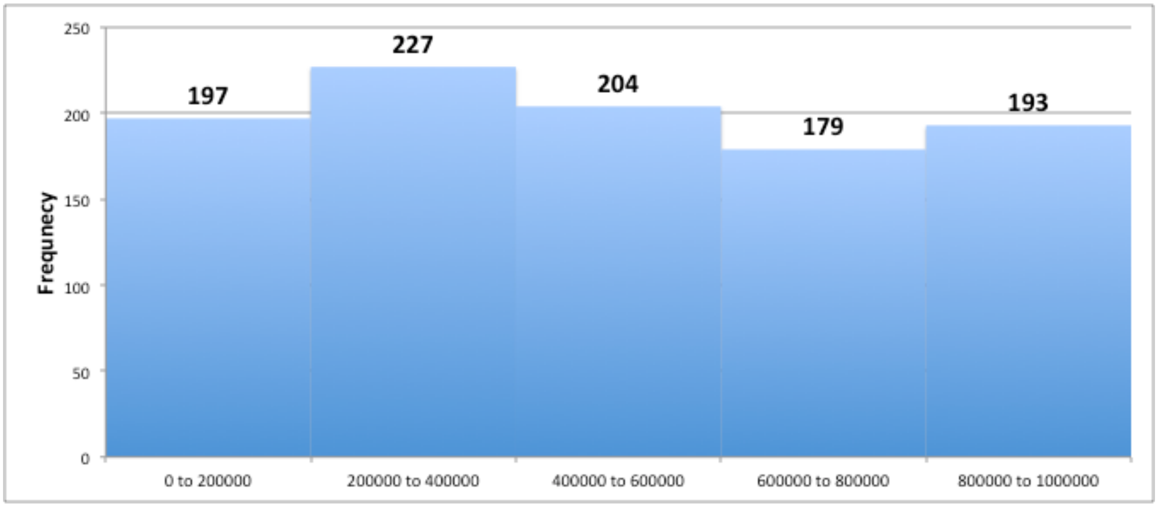

Figure 11: Histogram Showing Distribution of random() Values

Interference with Other Designs

The Bluetooth module used to connect to the mobile device has the potential to interfere with other devices although it is unlikely. It would be unlikely because the Bluetooth module conforms to the standards outlined by the Bluetooth Special Interest Group.

Usability

In reality this device is supposed to be able to be used on a front door. The prototype in its current form on the breadboard would not be stable enough to use in this capacity. However, the Lockitron housing that contains the motor to turn the door lock contains enough room inside of it to house a consolidated form of the prototype system. Systems in the Adafruit Bluefruit product line could be utilized to consolidate the current system as there are compact Arduinos with Bluetooth LE compatibility that are available.

The concerns for the device usability come with the fact that this system is still relatively unsecure. If a hacker wanted to breach the system they could cycle through all of the possible codes for the first screen, and manually enter the random number themselves. By doing this they could gain entry despite not being the owner of the lock. Although the bar of security is still very high due to the fact that the hacker would have to have the device of the owner of the lock. This is due to the fact that the hacker would not have the unique application or necessary Bluetooth ID.

Conclusion and Future Work

This lock was created as a means to explore random number generation from ambient noise and improving the security of wireless door locks. This project was successful and the random number generation should realistically be explored for commercial devices. Additionally, the design of the door lock showed to be very secure. With the restriction on the dissemination of the iPhone application, and customization for every user's Bluetooth ID the lock could actually be a valid option for a door lock. With doing this project it was discovered that researchers in the Department of Computer Engineering at the National Technical University of Ukraine (see reference) have published research on generating true random numbers based on environmental noise for military applications. Finding and reading this research supported my findings in this project

Looking forward there is more work to be done to refine, and advance this project. The first thing that can be done to advance this project is adding the capability to allow additional users to have access to the lock, and remove them. This is a necessary addition because it will allow more than one user to open the door, and allow temporary accesses. To refine this project the Arduino Uno and Bluetooth Low Energy shield can be replaced with an all in one board. Additionally the microphone and motor driver can be consolidated. With these modifications these parts can be housed in the Lockitron body. This would allow the lock to fit and function independently on the door as an actual product.

References

Blaszczyk, Marta, and Richard A. Guinee. "A Novel Modelled True Random Binary Number Generator for Key Stream Generation in Cryptographic Applications." *MILCOM 2008 - 2008 IEEE Military Communications Conference* (2008): n. pag. Web. <http://www.wseas.us/e-library/conferences/2009/cambridge/ISPRA/ISPRA09.pdf>.

"BLE Shield." *Red Bear Labs*. N.p., n.d. Web. <http://redbearlab.com/bleshield/>.

Drozhzhin, Alex. "Internet of Crappy Things - IoT." *Https://blog.kaspersky.com/internet-of-crappy-things/7667/*. Kaspersky, n.d. Web. <https://blog.kaspersky.com/internet-of-crappy-things/7667/>.

Goodin, Dan. "Samsung Smart Home Flaws Let Hackers Make Keys to Front Door." *Ars Technica*. N.p., 02 May 2016. Web. 2016. <http://arstechnica.com/security/2016/05/samsung-smart-home-flaws-lets-hackers-make-keys-to-front-door/>.

Goodin, Dan. "Samsung Smart Home Flaws Let Hackers Make Keys to Front Door." *Ars Technica*. N.p., 02 May 2016. Web. <http://arstechnica.com/security/2016/05/samsung-smart-home-flaws-lets-hackers-make-keys-to-front-door/>.

"NRF8001." */ Bluetooth Smart/Bluetooth Low Energy / Products / Home*. Nordic Semiconductor, n.d. Web. <http://www.nordicsemi.com/eng/Products/Bluetooth-R-low-energy/nRF8001>.

"Open Sesame! A SMS-controlled Door Lock." *Overview*. Adafuit, n.d. Web. <https://learn.adafruit.com/open-sesame-a-sms-controlled-door-lock?view=all>.

"7 Best Electronic Door Locks for Your Home." *SafeWise RSS*. N.p., 16 Feb. 2016. Web. <http://www.safewise.com/blog/finding-the-perfect-electronic-door-lock-for-your-home/>.

Smith, Ms. "Hacking and Attacking Automated Homes." *Network World*. N.p., 25 June 2013. Web. <http://www.networkworld.com/article/2224849/microsoft-subnet/hacking-and-attacking-automated-homes.html>.

WIND. *SECURITY IN THE INTERNET OF THINGS* (n.d.): n. pag. Web. <http://www.windriver.com/whitepapers/security-in-the-internet-of-things/wr_security-in-the-internet-of-things.pdf>.