# Multipurpose Key Chain with Lock Device

By

Junting Lou

Yaming Tang

Lida Zhu

Final Report

ECE 445, Senior Design
(Fall 2012)

TA: Rajarshi Roy

08 December 2012

Project No. 11

# Abstract

This project involves building two separating circuits implementing the design of a multipurpose keychain with its associated lock device. It aims to solve a numbers of daily problems incurred from mechanical locks.

The keychain in the completed design carries 8 keys with 8 LEDs corresponding to each of them. There is a separate device to be installed on a mechanical lock. Both devices have an Atmel ATmega328 microcontroller and a Bluetooth transceiver. With wireless communication enables between them, up-to-date status about whether a lock is locked can be updated and stored in the keychain. The keychain is also able to indicate the correct key to a certain lock based on a unique lock ID. The keychain is also equipped with a buzzer which can be triggered by the lock device when one cannot find the keys. In addition, remotely locking or unlocking a door is also implemented.

The project was completed with a few minor variations from the initial design and we are able to have all predefined functionalities working. Although the actual products have a few drawbacks at this moment, with the current design, a few modifications will absolutely make it worth produced and marketed in the future.

# Contents


Abstract……1
Contents……..2
1. Introduction……..3
1.1. General Introduction……3
1.2. Blocks and Modules……4
1.2.1. Microcontroller Unit……..4
1.2.2. Power Supply……4
1.2.3. Bluetooth……4
1.2.4. Sensor……4
1.2.5. Buzzer and LEDs……...4
1.2.6. Servo Motor……...4
1.3. Functionality……5
2. Design……..6
2.1. Power Supply……6
2.2. Microcontroller Unit……6
2.2.1. ATmega328P-PU……...6
2.2.2. ATmega328 on Breadboard……...7
2.2.3. Addressable Latches……..7
2.3. Bluetooth Transceivers……8
2.3.1. Configuration……8
2.3.2. Connection and Logic Level Converter……9
2.4. Sensor……..10
2.5. PCB……10
3. Requirements and Verifications……..11
3.1. Power Supply……..11
3.2. Microcontroller Unit……...12
3.3. Bluetooth……13
3.4. Sensor……..14
3.5. Buzzer……14
3.6. Failed Requirements……...15
4. Cost……16
4.1. Parts……16
4.2. Labor……..16
5. Conclusion……17
5.1. Accomplishments……17
5.2. Uncertainties……...17
5.3. Future Work……18
5.4. Ethical Considerations……18
5.5. Acknowledgements……19
Reference……...20
Appendix A – Figures and Pictures……...21
Appendix B – Requirements and Verifications Table……...27
Appendix C – Microcontroller Code (Lock Device #3)……35
Appendix D – Microcontroller Code (Keychain)……..38

# 1. Introduction

## 1.1 General Introduction

Despite the improvement in the variety of locking devices available in the market, including electronic keycards and fingerprint recognition, mechanical locks still play a very significant role in our everyday life. Our project in general is targeted to solve problems associated with mechanical locks. By building a keychain and corresponding lock devices, one is able to access information about whether a certain lock is locked at anytime. Also, it is not anymore a problem identifying the correct key out of a number of similar or identical ones. The key chain is equipped with a buzzer which can be triggered when one cannot find it. Last but not least, similar to any car key nowadays, one could remotely lock or unlock a door with the keychain. Both the key chain and the lock devices are controlled by an Atmel ATmega328 microcontroller and they communicate with each other via Bluetooth. Other parts such as LEDs, buzzer and servo motor are installed for various purposes. Current development aims to build a prototype with a key chain that holds 8 keys. In the future, other functionalities and extensions can be added to improve the overall marketability of the product.

Figure 1.1 below is the basic block diagram illustrating how this project is designed and constructed. The listed blocks and modules will be explained in greater depth in the next section.

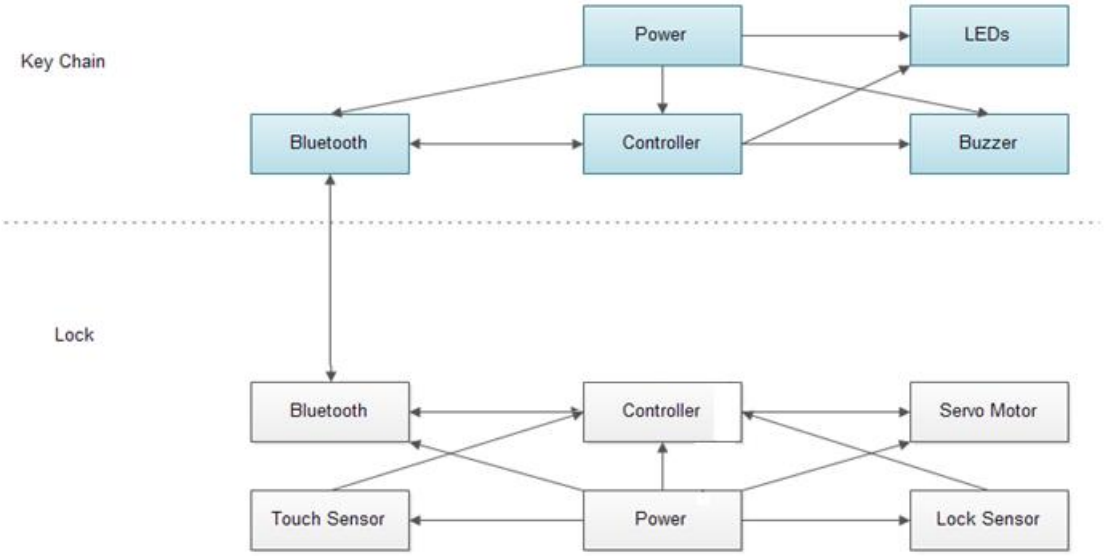


Fig. 1.1 Block Diagram

## 1.2 Blocks and Modules

### *1.2.1 Microcontroller Unit*

On both the keychain and the lock device, an Atmel ATmega328 chip will be used as the microcontroller. They are preloaded with the Arduino UNO bootloader and will be programmed using the Arduino programming environment with the help of the FT232RL USB to serial breakout board. It is responsible for processing all information received via Bluetooth or button presses and generating correct output.

### *1.2.2 Power Supply*

On both the keychain and the lock device, a 9V alkaline battery will act as the power source. In addition, a 5V voltage regulator and a 3.3V voltage regulator are going to be used to produce the correct voltage. The entire circuit will be operating at 5V except for the logic level converter that will need both 5V and 3.3V input voltages.

### *1.2.3 Bluetooth*

The wireless communication between two devices is achieved over Bluetooth. The Bluetooth module this project makes use of is the HC-05 Bluetooth transceiver. How these modules are configured and paired will be explained in greater details in Chapter 2.

### *1.2.4 Sensor*

The built-in sensor in the lock device is simply designed using a voltage divider circuit. It is able to keep track of whether the lock is locked or unlocked at the moment by sensing whether the latch is inside the socket.

### *1.2.5 Buzzer and LEDs*

The display interface on the keychain makes use of a buzzer and 8 bi-color (red and green) LEDs with common cathode. The LEDs are responsible for displaying the status of all locks as well as indicating the correct lock upon a button press. The buzzer will be triggered in order for the user to locate where the keychain is.

### *1.2.5 Servo Motor*

The servo motor we chose is the Parallax continuous rotation servo motor. Through the microcontroller, we could tell the servo to rotate at a specific speed for a certain time period. This makes it possible to tell it to rotate for a specific angle so as to perform the actions of lock and unlock.

## 1.3 Functionality

The objectives of the project up from the design stage are all met except that the Ethernet module is removed from the design as it is not a function that people will use frequently and some issues such as security are involved in the design. Instead, remotely lock and unlock via Bluetooth communication is added into the project. There are a total of 8 bi-color LEDs on the keychain, each corresponding to a key-lock pair. Whenever a door is locked or unlocked with the keychain, the built-in sensor on the lock side will update the status in the keychain. Therefore, one is able to access such information from anywhere anytime. The lock device is equipped with two buttons, one of them being able to trigger the buzzer on the keychain and the other one will ask the keychain to light up the LED corresponding to this particular lock. Figure 1.2 below shows the completed keychain PCB with 8 LEDs.

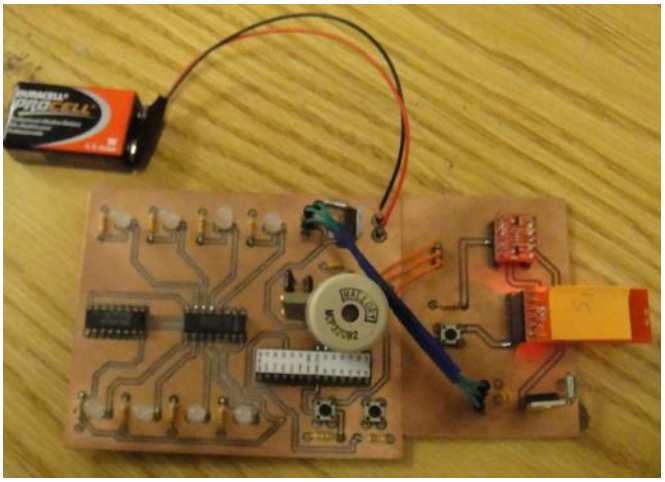

Fig. 1.2 Keychain

# 2. Design

## 2.1 Power Supply

The power source on both devices comes from a 9V alkaline battery each. As the circuit mainly requires 5V voltage with a few exceptions that require 3.3V, 9V battery seems to be the best choice considering its size and feasibility to step the voltage to a variety of what we need. Another possible choice is to use 4 AAA batteries but they will occupy a lot more space than a single 9V battery does.

To step the voltage down, we made use of two voltage regulators, LM7805AC (5V, 1A) and UA78M33C (3.3V, 500mA) on both devices. Our circuits need 5V and 3.3V voltage supplies and lower than 200mA current under full operation mode. Both these voltage regulators are 3-pin devices with an input, an output and a ground pin. The basic connection is illustrated in Fig. 2.1 below using example of the 5V regulator. It is noted that a 0.33uF capacitor is connected from input to ground while a 0.1uF capacitor is connected from output to ground.

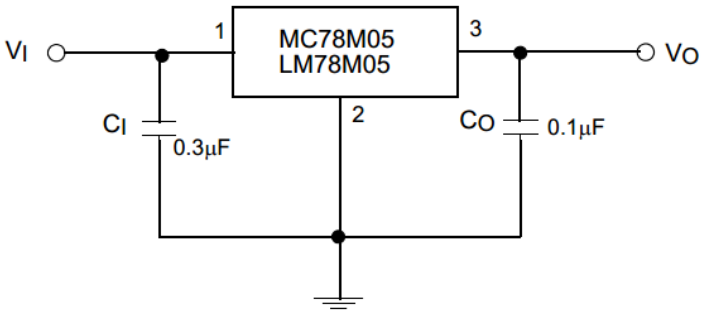


Fig.2.1 Voltage Regulator circuit

## 2.2 Microcontroller Unit

### *2.21 ATmega328P-PU*

Although a variety of microcontrollers that can accomplish the tasks, Arduino microcontrollers seem to be the most widely used ones for lots of project nowadays. Using an integrated Arduino microcontroller board is going to take up a lot of space. Also, an Arduino board such as Arduino UNO costs a lot more than just a single microcontroller IC chip. Bearing these considerations in mind, we decided to use the Atmel ATmega328P-PU chip which is the exactly same chip that is used on an Arduino UNO. The chips that we purchased are preloaded with the UNO Optiboot bootloader. This enables us to program it through the Arduino software at treating it as an Arduino UNO. The microcontroller has 14 digital I/O pins and 6 analog input pins which will fulfill our needs.

## *2.22 ATmega328 on Breadboard*

In order to have this microcontroller working on a breadboard, some external connections are necessary. This is further illustrated in Fig. 2.2 below.

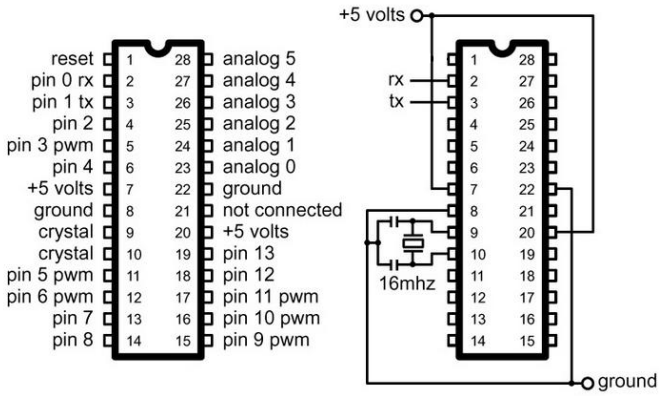


Fig. 2.2 Basic ATmega328 Setup on Breadboard

Firstly, the Vcc and ground pins are connected to 5V and ground respectively. Then, a 16MHz crystal oscillator is connected between pin 9 and pin 10 with a 22pF capacitors connected from each of these pins to ground. Not included in this figure is the connection on the reset pin (Pin 1). A push button is used to enable the reset feature of the Breadboad Arduino. The top left pin of the button is connected to ground and the bottom left one is connected to pin 1 of the microcontroller. Whenever the button is pressed, pin 1 is connected to ground and causes the controller to reset. Also necessary is a pull up resistor connected from pin 1 to Vcc. The resistor I used has a resistance value of 10kΩ. This makes sure that when the button is not pressed, pin 1 is always pulled up to Vcc, preventing the chip from resetting itself. When all the above steps are performed, the microcontroller is successfully set up. It came in with a preloaded LED blink program. By connected an LED and a resistor from pin 13 to ground, we are able to see the effect of the blink program. The LED starts blinking slowly and speeds up gradually and finally stops.

## *2.23 Addressable Latches*

The reason for using addressable latches (74LS259) is that we want to update all LEDs at the same time instead of one by one. We need some storing units to store the information about the color of a specific LED while updating the rest. To display information about all locks, bi-color LEDs (red and green) are used. The addressable latch on the top right corner accounts for the red colors and the other accounts for the green colors. Pin 13 on the addressable latches are the data inputs and pin 14 and pin 15 are used to switch among different modes of the latches. When pin 14 and pin 15 are both high, the outputs from the addressable latches will not change and they are in memory mode. When pin 14 is low and pin 15 is high, the data from pin 13 will be updated to the output specified by the address pins without changing the values of the other outputs. In general, status of each lock will be obtained from the corresponding location in a predefined array. If it is true, the data denoting red will be high and that denoting green will be low. If it is false, it is the other way. Then the address bits will be determined based on the lock

number, i.e. the array index. Then color will be updated piecewise to each output of the addressable latches. The schematic involving just the microcontroller and the addressable latches is included in Fig.2.3 below.

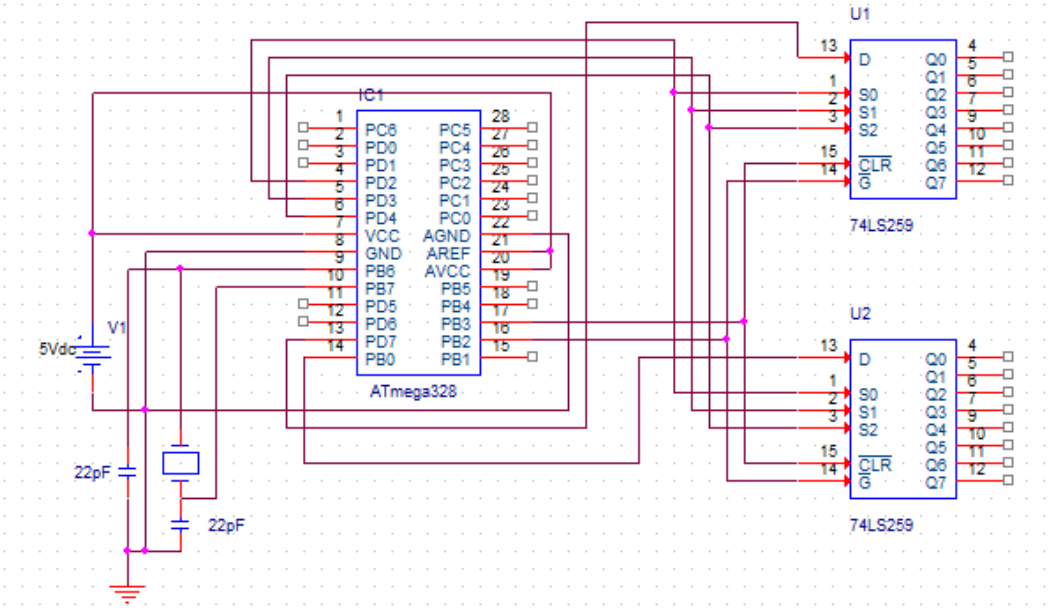


Fig. 2.3 Schematic of Keychain Display Module

## 2.3 Bluetooth Transceivers

The Bluetooth transceivers in this project are responsible for sending and receiving data at both devices. It is the most crucial component of the design since all functionalities are based on successful transmitting of data quickly and reliably. The Bluetooth module chosen for this project is the HC-05 Bluetooth transceiver modules. Bluetooth connection can only be set up between a master and a slave module, meaning that neither two masters nor two slaves could establish connections. In addition, to have two modules paired, they need to have the same paring code. The HC-05 modules are master/slave modules, meaning that each of them could be configured to be a master or a slave whenever the user wants.

### *2.3.1 Configuration*

The Bluetooth modules can be configured using AT command though any serial communication software. In this case, we chose to use Putty. To enter the AT mode of the module, Pin 34 has to be grounded first when the chip is powered. Then it automatically enters AT mode when pin 34 is connected to 5V. To have the Bluetooth chip communicating with the computer, the FTDI breakout board that is used to program the microcontroller is used again here. Figure 2.4 shows the connection for configuration of the Bluetooth module. AT command is used to change the

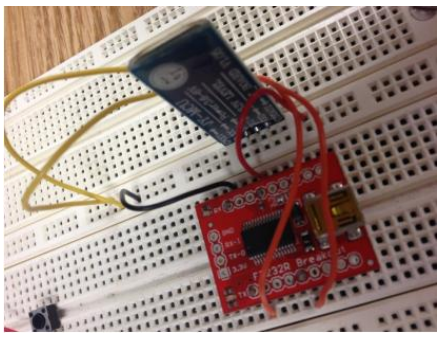

Fig.2.4 Configuration Circuit

name, the role (master/slave), the pairing code, the communication baud rate and a lot more of the Bluetooth module. Table 2.1 below includes some of the most important AT commands that are used in this project.

Table 2.1 Important AT Command

| Purpose | Command |
|---|---|
| Verify Connection | AT |
| Set Name | AT+NAME=<name> |
| Set Role | AT+ROLE=1(Master)/0(Slave) |
| Set Pairing Code | AT+PSWD=<password> |
| Pairing Options | AT+CMODE=1(Auto)/0(Last Address) |

### *2.3.2 Connection and Logic Level Converter*

The original HC-05 Bluetooth chip requires 3.1V to 4.2V voltage input and has a data (RX/TX) level voltage of around 3.3V. We purchased the breakout board because they do not require soldering. These chips are able to operate at a voltage of 5V. However, the data level voltage is still 3.3V. This can be seen from Fig 2.5 below.


Fig. 2.5 Bluetooth Transceiver Test Result

Fig 2.5 illustrates the result of a simple test involving two Bluetooth modules and two microcontrollers. One of the controllers is made to send some data repeatedly over Bluetooth to the other controller. The top waveform captures the data that comes out from the first microcontroller and the bottom waveform captures that being received by the Bluetooth module connected to the second controller. As seen above, the top waveform has a data level of 5V while the bottom one being 3.3V. This means that some sort of converting process is required if we want to connect data I/O from the Bluetooth module into the microcontroller. This is while we made use of the logic level converter. The logic level converter that we purchases is able to convert a 3.3V signal to 5V and vice versa. Details about how logic level converters are connected can be found in the Appendix.

## 2.4 Sensor

The built-in sensor on the lock device needs to produce different output when the lock is in different positions (locked/unlocked). The sensor makes use of a simple voltage divider circuit as shown in Fig 2.6 below. Resistor R1 is 340kΩ each and R2 is 220Ω. The two black dots in the circuit symbolize two open wires that are mounted inside the socket of the lock. When the lock is locked, the conducting latch will make the circuit connected. When the lock is unlocked, the circuit is open and therefore the output from the sensor will simply be 5V. When it is locked, the output voltage can be derived from equation (2.1) below.

$$V_{out} = \frac{0.22}{0.22+340\times2} \times 5V \approx 0V \quad (2.1)$$

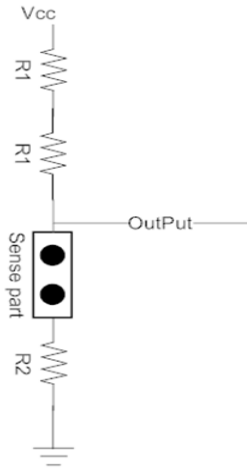


Fig. 2.6 Sensor Circuit

## 2.5 PCB

The PCB design involves a few considerations. It is relatively easier for design the PCB on the lock side as size is less considered to be a factor. On the keychain side, we want to make it as small as possible. However, with the number of components that we have in our design and the consideration of have wider traces for easier fabrication and soldering, we decided to make two PCBs for our keychain. One of them has the main circuit with all components except the Bluetooth module and the logic level converter which will be placed onto the second smaller board. We placed some headers on them for later connection. Detailed information about PCB designs can be found in the Appendix.

# 3. Requirements and Verifications

The detailed table of requirements and verifications from original design review can be found in the Appendix. Since the Ethernet module is removed from the circuit, corresponding requirements and verifications are no longer applicable. Most of the verifications met requirements except the battery lifetime that is going to be explained subsequently.

## 3.1 Power Supply

Since all of the components in the circuit require either 5V or 3.3V voltage supplies, it is important to have steady output from the voltage regulators.

The test that is performed can be illustrated in Fig 3.1 below.

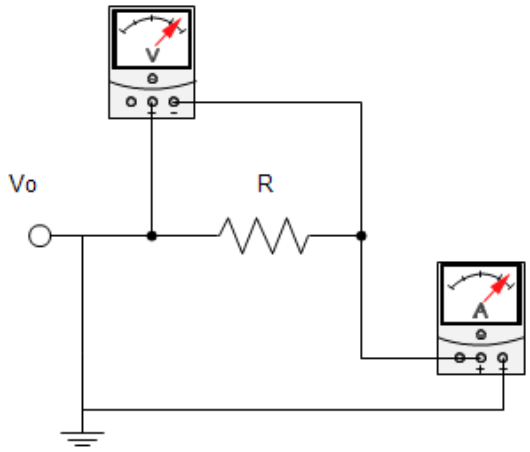


Fig. 3.1 Test Circuit for Power Supply

The purpose of this test is to verify that the output from the voltage regulator stays at a value around 5V with a tolerance of ±0.2V at a variety of current values. The current requirements for both devices are estimated to be between 20mA to 200mA in the best and worst cases, respectively. In Fig.3.1 above, $V_o$ is the output from the voltage regulator circuit as described in Fig.2.1 and the resistor value is varied to achieve desired current. The test is performed at current values from 20mA to 200mA with 20mA intervals and the corresponding resistance values are calculated based on Ohm's Law in equation (3.1) below. Table.1 below illustrates results from this test.

$$R = \frac{V}{I} = \frac{5}{I} \tag{3.1}$$

From Table 3.1 below, at all current values with respective resistance, the voltage meet our requirements of staying in the range of 5±0.2V. This confirms that the voltage regulator works up to required standard. Exact same test is performed on 3.3V circuit and results also fall in the accepted range.

Table.3.1 Test Results for Power Supply

| R, Resistance (Ω) | I, Current Reading (mA) | V, Voltage Reading (V) |
|---|---|---|
| 250 | 19.88 | 5.03 |
| 125 | 39.96 | 5.02 |
| 83 | 60.31 | 5.02 |
| 63 | 79.75 | 5.02 |
| 50 | 101.03 | 5.02 |
| 42 | 119.62 | 5.04 |
| 36 | 141.10 | 5.01 |
| 31 | 160.84 | 5.06 |
| 28 | 179.23 | 5.02 |
| 25 | 201.50 | 5.02 |

## 3.2 Microcontroller Unit

Regarding the microcontroller, there are two major issues. The first issue is that we want to make sure that buttons are debounced if necessary. The second is that we want it to be able to display correct information based on pre-stored data regarding the status of locks. This is also partly a test for the operation of addressable latches.

As for the buttons, we verified that we do not need a special debouncing function for it. This is mainly because that after each button press, the system needs to wait for about 4 to 5 seconds for the Bluetooth modules to establish connection. During this time period, the bouncing of the switches is already over.

In order to verify correct display of pre-stored data, we connected the circuit on the keychain side with only the microcontroller, the addressable latches and the LEDs. The array holding data in the microcontroller program is written with different combination of values and we observed the color on the LEDs to see whether they match. Table 3.2 below illustrates the test results and all tests pass by having the exactly correct display.

Table 3.2 Test Results for Keychain Display Interface

| Stored Combination | Verified |
|---|---|
| LLLLUUUU | Yes |
| UUUULLLL | Yes |
| LLUULLUU | Yes |
| UULLUULL | Yes |
| LULULULU | Yes |
| ULULULUL | Yes |
| LLUUUULL | Yes |
| UULLLLUU | Yes |
| LUUUUUUL | Yes |
| ULLLLLLU | Yes |

## 3.3 Bluetooth

The primary reason we chose Bluetooth to be our protocol for wireless communication is that it is one of the most reliable protocols which possibly fulfill our requirements for distance. The features of the product require Bluetooth communication to be able to send and receive data within 20 meters reliably. In addition, it should also work without any error when data is sent as fast as once every 0.5 second.

The test we performed here is similar to that is mentioned in Section 2.3.2 where we made use of two microcontrollers, each connected to a Bluetooth chip. On one of the microcontroller, an integer variable will act as a counter and will increment from 0 over a set time period until 300.. The number then will be sent via Bluetooth to the other microcontroller. The other one will display the information onto the computer screen via Putty. If all data is sent and received successfully, we should see something like Fig 3.2 below.

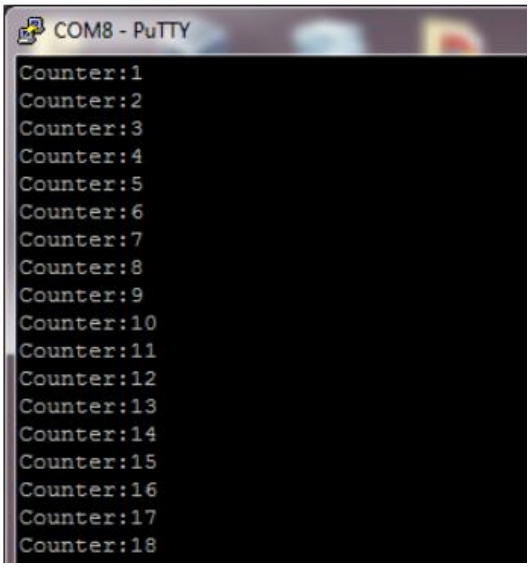


Fig 3.2 Screen Output in Bluetooth Test

We performed this test with a number of distance between modules as well as different time intervals for sending data and as illustrated in Table 3.3 below, all tests are passed, verifying that the communication is reliable and accurate.

Table 3.3 Test Results of Bluetooth Test

| Time Interval / Distance | 0.5s | 1s | 2s |
|---|---|---|---|
| 0.1 m | ✓ | ✓ | ✓ |
| 1 m | ✓ | ✓ | ✓ |
| 5 m | ✓ | ✓ | ✓ |
| 10 m | ✓ | ✓ | ✓ |
| 20 m | ✓ | ✓ | ✓ |

## 3.4 Sensor

The purpose of the sensor is to detect the change of status of a lock. Therefore, we need to make sure that the output from the sensor can be accurately and quickly updated to the microcontroller whenever one is locking or unlocking the door.

We verified this by attaching the sensor on the lock and repeatedly lock and unlock it for 20 times with a frequency of once every second. The output from the sensor is observed on the oscilloscope. The captured waveform from the oscilloscope is included in Figure 3.3 below. From the graph we can see that every action of locking and unlocking can be seen clearly and updated very quickly.

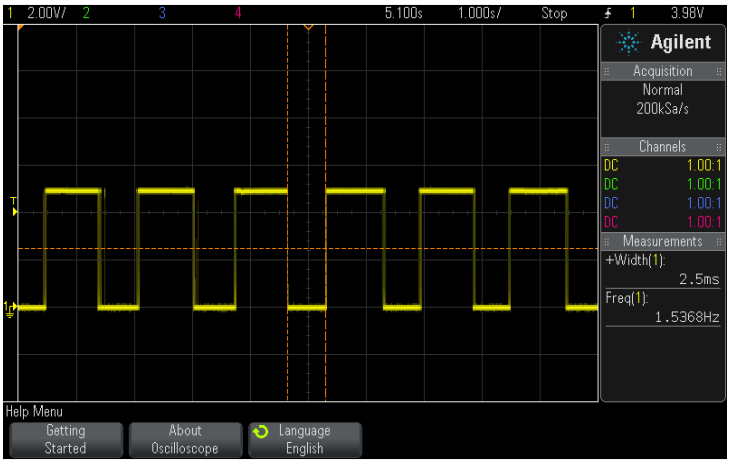


Fig 3.3 Oscilloscope Output from Sensor Test

## 3.5 Buzzer

In order to make sure that the buzzer noise is loud enough to be heard, we created a few situations for the test. The buzzer needs to be heard within 15 meters in an open space, within a box or drawer or in an adjacent room. Table 3.4 shows the test results. It is confirmed that the buzzer creates noise that is loud enough to be heard under a variety of situations.

Table 3.4 Results from Buzzer Test

| Conditions / Distance | Open Space | Inside Locker | Adjacent Room |
|---|---|---|---|
| 1 m | ✓ | ✓ | ✓ |
| 5 m | ✓ | ✓ | ✓ |
| 10 m | ✓ | ✓ | ✓ |
| 15 m | ✓ | ✓ | ✓ |
| 20 m | ✓ | ✓ | Not So Clear |

## 3.6 Failed Requirements

The only requirement that is failed to be verified in this project is the battery lifetime. Based on the design review, the battery needs to have a battery lifetime of more than 45 days provided with reasonable assumptions. These assumptions include the number of operations the keychain performs daily. However, under current circumstance, this requirement could not be fulfilled and current battery can only last up to about 20 days under the same assumptions. The situation is even worse on the lock device side because the servo motor is consuming a lot of power continuously. This problems is not able to be solved with the current design, but is easily solvable with a few modifications and improvements. It will be explained later in Section 5.3.

# 4. Cost

## 4.1 Parts

Table 4.1 Total Cost for Parts

| Module | Part | Unit Cost | Number | Subtotal |
|---|---|---|---|---|
| Power Supply (Key Chain) | Energizer 522BP-2 9V Alkaline Battery | 3.99 | 1 | 3.99 |
| | UA78M33C Voltage Regulator | 1.18 | 1 | 1.18 |
| | LM7805AC Voltage Regulator | 1.05 | 1 | 1.05 |
| Power Supply (Lock) | Energizer 522BP-2 9V Alkaline Battery | 3.99 | 1 | 3.99 |
| | UA78M33C Voltage Regulator | 1.18 | 1 | 1.18 |
| | LM7805AC Voltage Regulator | 1.05 | 1 | 1.05 |
| Bluetooth (Key Chain) | HC-05 Bluetooth Transceiver Module | 5.30 | 1 | 5.30 |
| Bluetooth (Lock) | HC-05 Bluetooth Transceiver Module | 5.30 | 1 | 5.30 |
| LEDs | Bi-Color LED – Common Cathode, Pack of 10 | 2.00 | 1 | 2.00 |
| Buzzer | PCB Tone Alarm Buzzer | 0.15 | 1 | 0.15 |
| Motor | Parallax Continuous Rotation Servo Motor | 11.99 | 1 | 11.99 |
| Controller (Key Chain) | Atmel ATmega328P-PU Microcontroller with Bootloader | 5.50 | 1 | 5.50 |
| | **USB to UART Bridge – FT232RL*** | **10.95** | **1** | **10.95** |
| Controller (Lock) | Atmel ATmega328P-PU Microcontroller with Bootloader | 5.50 | 1 | 5.50 |
| Misc. | Resistors, Capacitors, Wires, Tactile Buttons, PCB and so on | 10.00 (Esti-mate) | 1 | 10.00 |
| | | | Total | 69.13 **(58.18)** |

## 4.2 Labor

Table 4.2 Total Cost for Labor

| Members | $/Hour | Hours/Week | Number of Weeks | Total/Person | *Multiplier (2.5) |
|---|---|---|---|---|---|
| Junting Lou | 30 | 12 | 12 | 4320 | 10800 |
| Yaming Tang | 30 | 12 | 12 | 4320 | 10800 |
| Lida Zhu | 30 | 12 | 12 | 4320 | 10800 |
| | | | | Total | 32400 |

# 5. Conclusion

## 5.1 Accomplishments

We created a keychain on PCB with a corresponding lock device to be mounted on any mechanical lock. The devices are able to demonstrate all four functionalities as previously intended. Figure 5.1 below shows a picture of our final product.

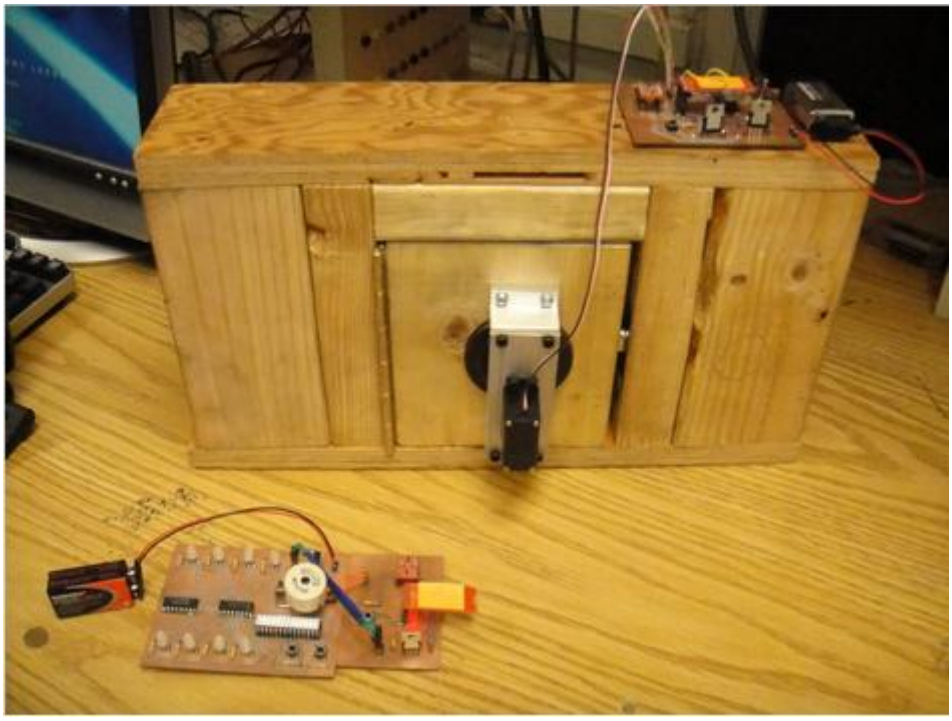

Fig. 5.1 Final Product

## 5.2 Uncertainties

Although we are able to achieve all four functionalities working, there are a few uncertainties involved in the final produce.

Firstly, as suggested in previous sections, the battery lifetime is relatively shorter than what we intended. Secondly, after each button press, about 4 seconds are needed for the Bluetooth modules to setup connection and this is not very efficient. In addition, the PCBs that we are having right now are very big because of the limitations that we have in PCB designs and fabrication. In the next section, we are going to suggest a few possible solutions to these problems. Lastly, since there is no external memory unit in our design, all the information is stored in the microcontroller as part of the program. This prevents users from turning off the microcontroller, because the information will be lost whenever power is cut off.

## 5.3 Future Work

To solve the issue with battery lifetime, we can add a memory unit such as an SD card to the design so that information about locks is no longer stored inside the microcontroller. Whenever the keychain is not being used, we can power it off to save power. Since more concern regarding power consumption comes from the lock device, we could possibly get rid of battery on the lock side and instead use wall power since the lock device is always mounted stationary.

Another possible future work is to replace Bluetooth protocol with some other wireless communication methods such as radio frequency or Xbee modules that can broadcast to others. This could potentially eliminate the time needed for Bluetooth modules to pair up with each other every time a button is pressed or a lock status changes.

Lastly, with more efficient PCB design and fabrication, the product can be a lot smaller than the current one.

## 5.4 Ethical Considerations

Table 5.1 Relevant Ethical Considerations

| Relevant IEEE Code of Ethics | Relevance in Project |
|---|---|
| 1. to accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment; | The keychain that we developed is going to integrate a battery as the power supply. We made sure that the battery makes stable performance at all times so that it does not create potential harm to others. As we are dealing with doors and locks, we will make sure that the device to be develop does not pose any safety or security issues on society. |
| 3. to be honest and realistic in stating claims or estimates based on available data; | We declare that all experimental data and calculations illustrated in related documents in the process of this project are honest and real. |
| 6. to maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations; | We will not embark on producing or marketing devices made in this course unless after qualified training and approval. |
| 7. to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others; | Our group accepted all constructive advice and suggestions from people around us and will acknowledge any contribution towards the eventual result of this project. |

## 5.5 Acknowledgements

We would like to thank Professor Andrew Singer and our TA Rajarshi Roy for their valuable support and guidance throughout the process of the project. We would like to thank Mr. Skee Aldrich and Mr. Scott McDonald from the ECE machine shop for making a nice door mounted with our motor and sensor for our demo. We would like to thank Mr. Skot Wiedmann and Mr. Mark Smart for their help with our PCB design and fabrication. Without all of the people mentioned above and the others who have helped us in one way or another, this project could never be completed.

# References

Mellis, D. (2008, OCTOBER 23). Building an arduino on a breadboard. Retrieved from http://arduino.cc/en/Main/Standalone

tigoe. (2009, MAY 28). Controlling lots of outputs from a microcontroller. Retrieved from http://www.tigoe.com/pcomp/code/arduinowiring/486/

Amedee, C. (2011, DECEMBER 2). *Easy bluetooth enabled door lock with arduino android*. Retrieved from http://www.instructables.com/id/Easy-Bluetooth-Enabled-Door-Lock-With-Arduino-An/?ALLSTEPS

Microchip. (2012, July 30). *enc28j60 stand-alone ethernet controller with spi interface*. Retrieved from http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en022889

# Appendix A - Figures and Pictures

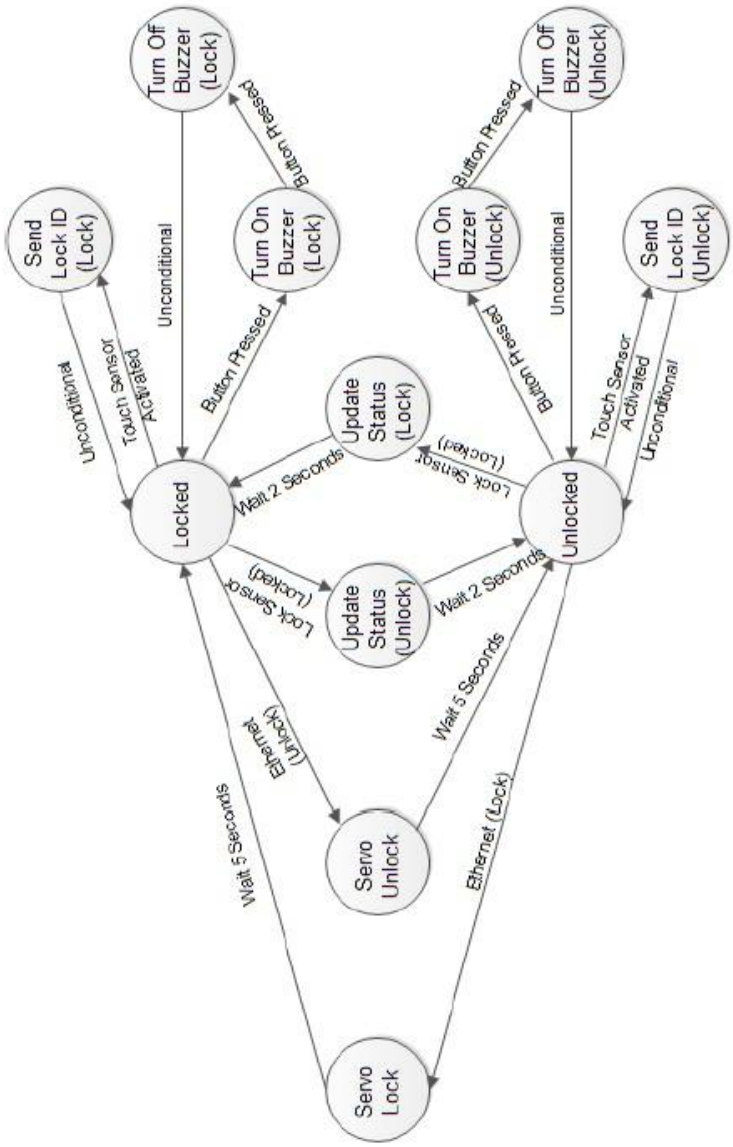


Fig. A.1 Control Flow (Lock Device)

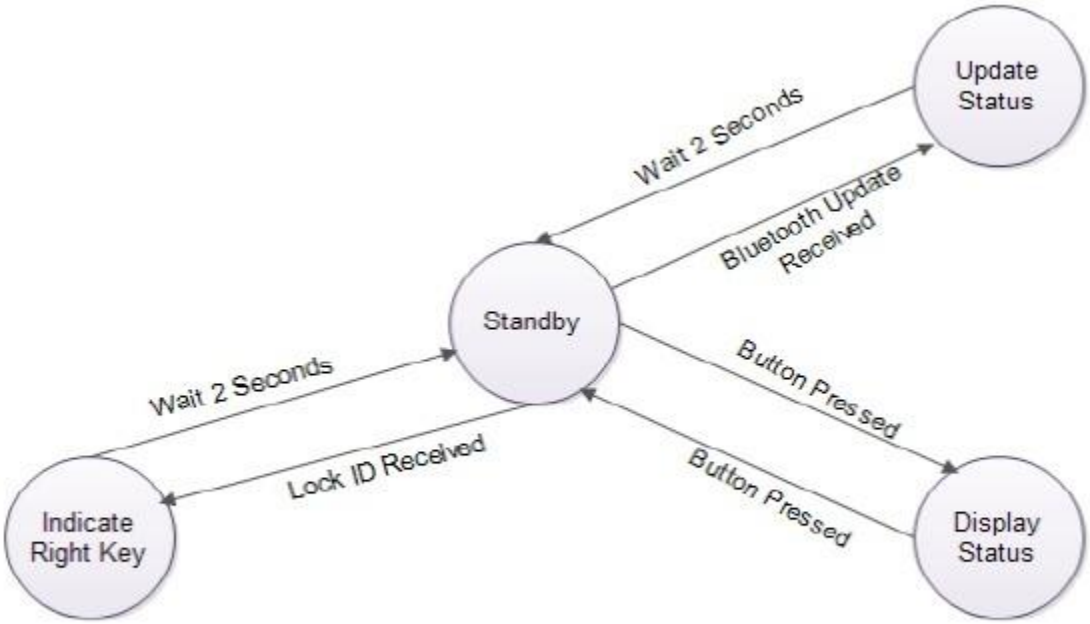


Fig. A.2 Control Flow (Keychain)

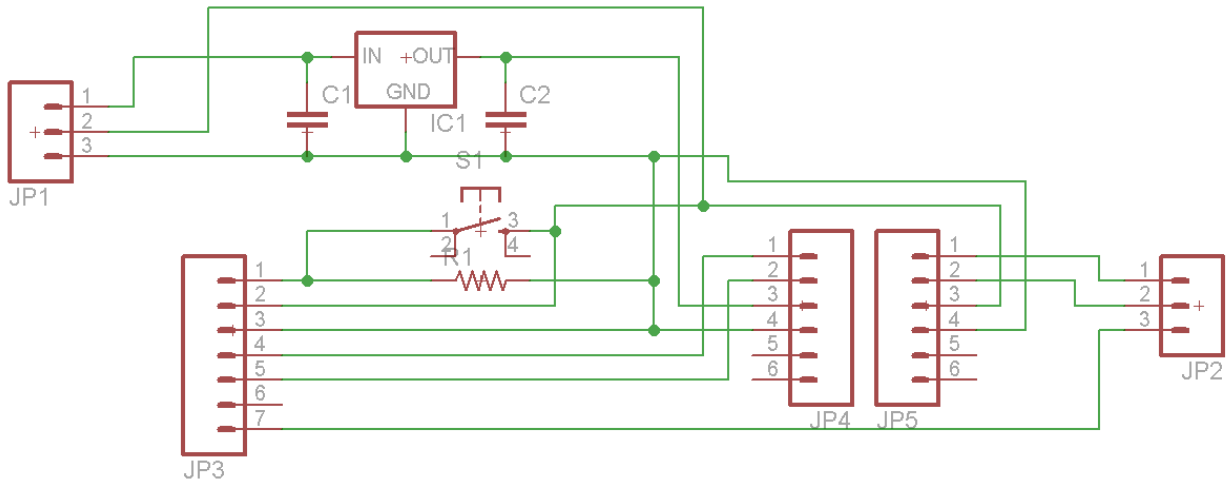


Fig. A.3 Schematics (Keychain Bluetooth)

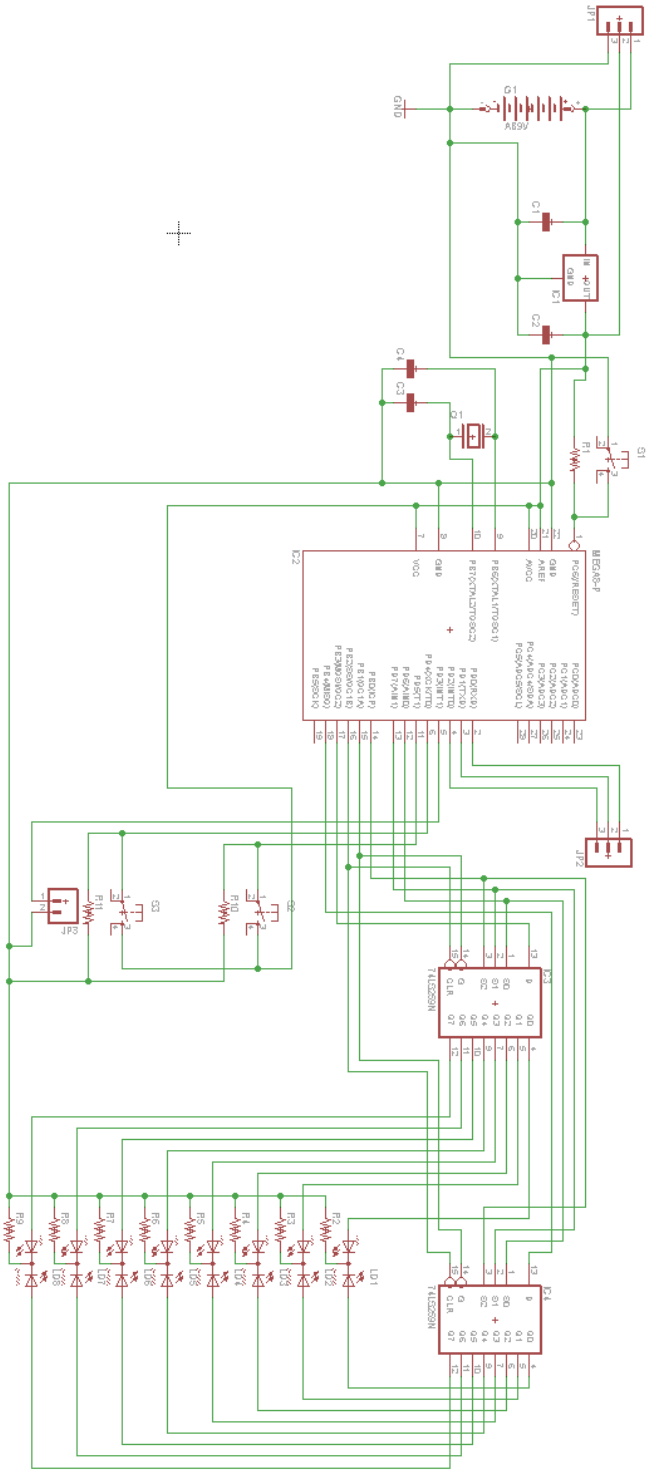

Fig. A.4 Schematics (Keychain Without Bluetooth)

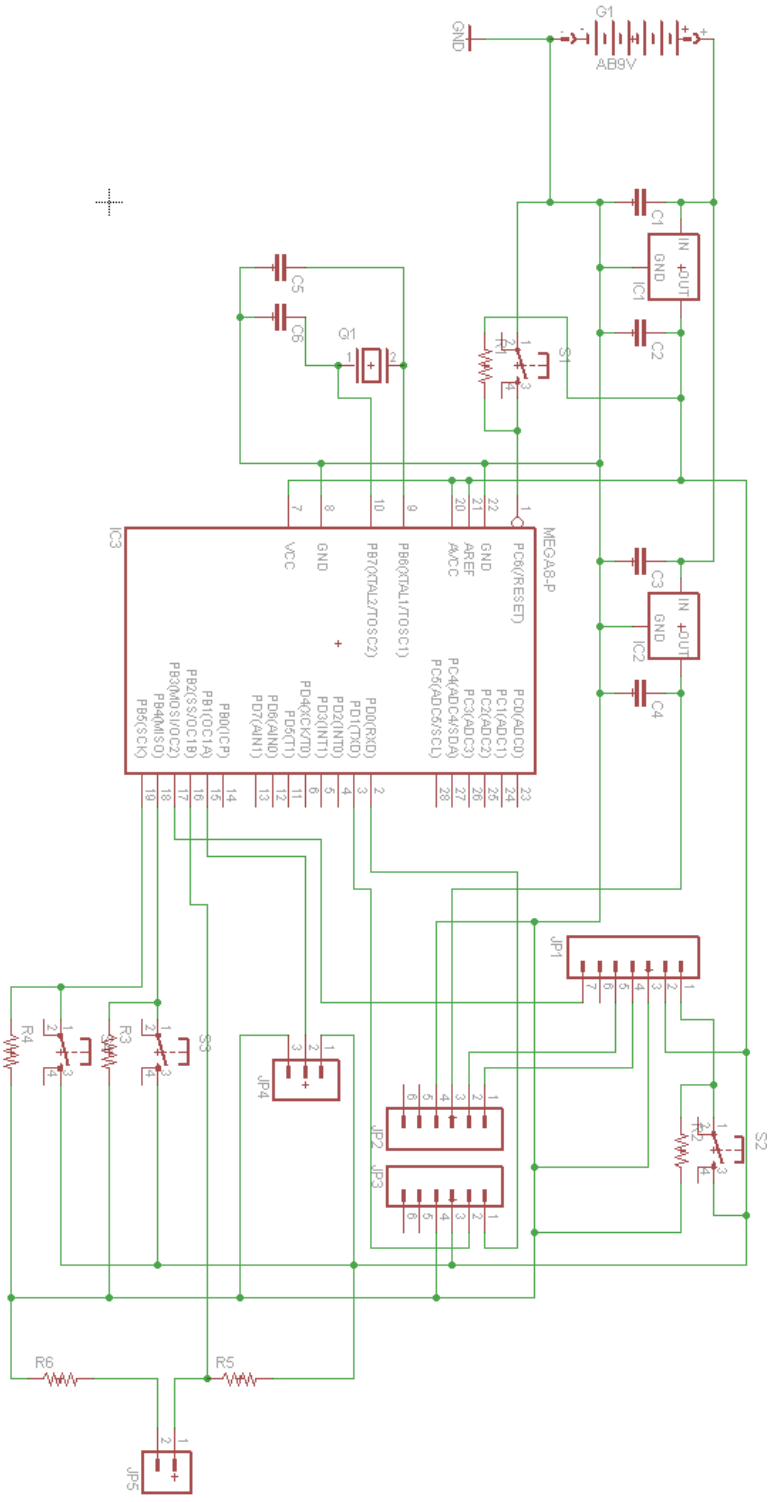


Fig. A.5 Schematics (Lock Device)

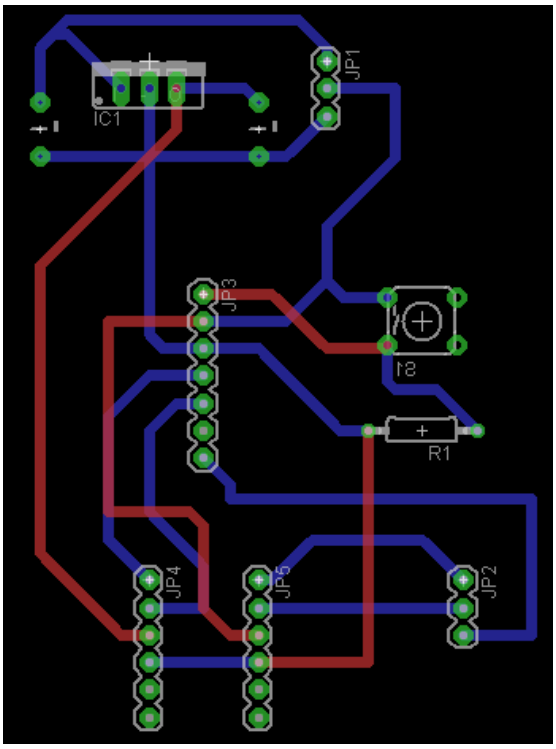


Fig. A.6 PCB (Keychain Bluetooth)

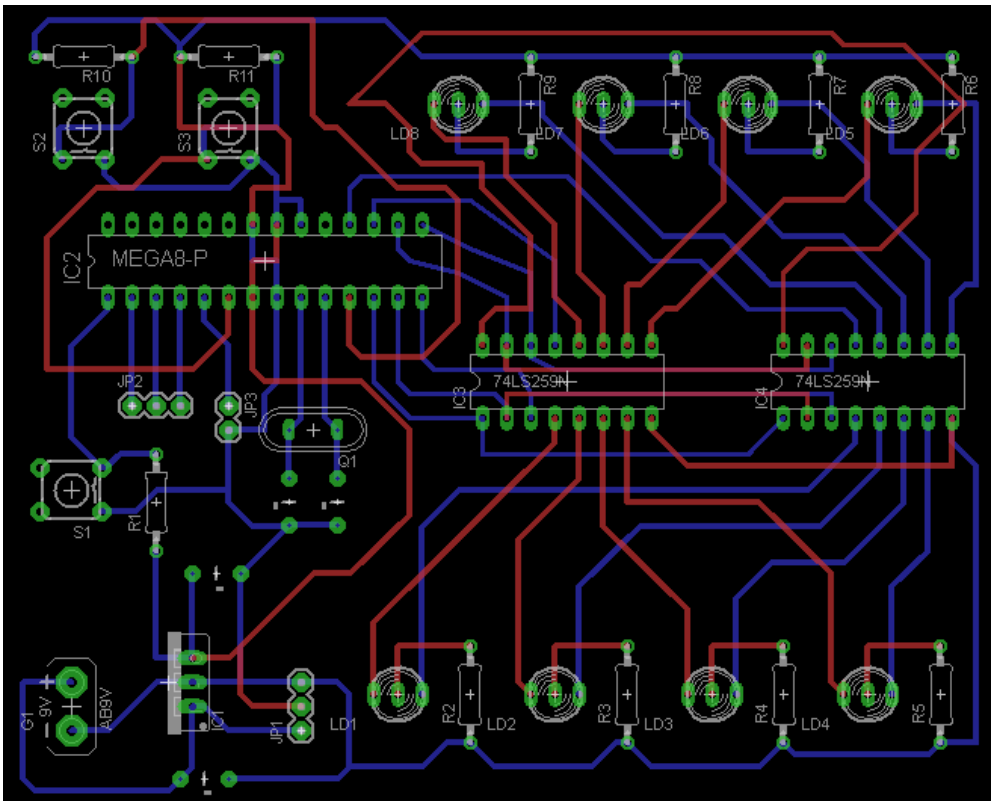


Fig. A.7 PCB (Keychain Without Bluetooth)

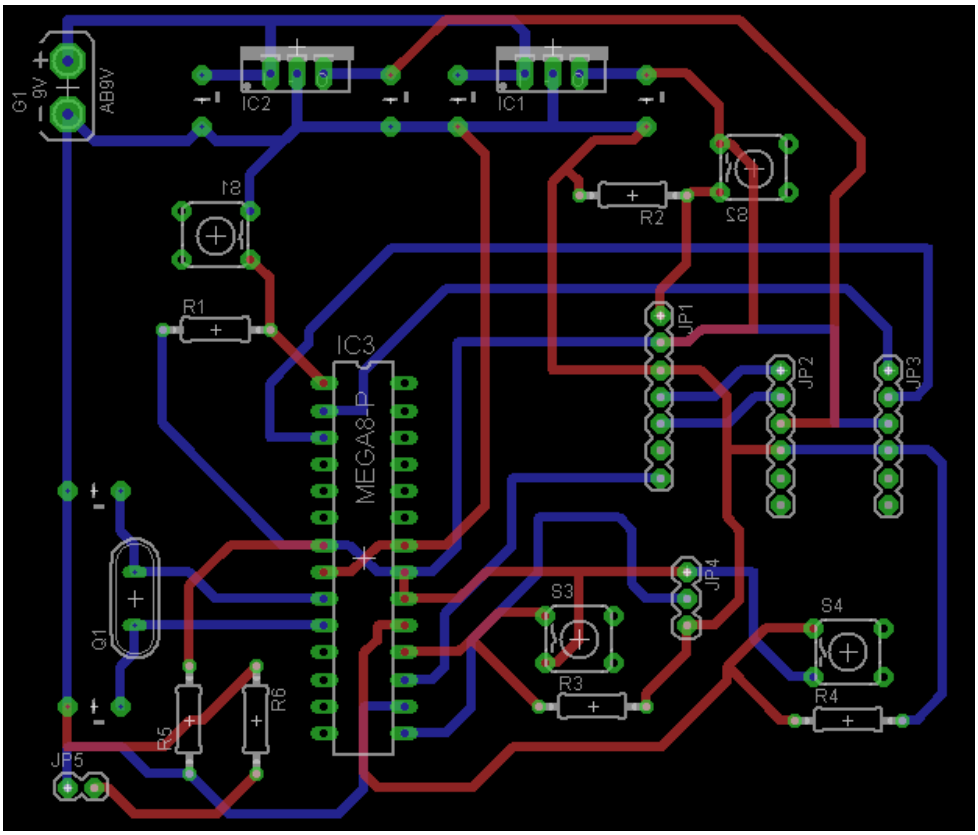


Fig. A.8 PCB (Lock Device)

# Appendix B – Requirements and Verifications Table

Table B.1 Requirements and Verifications Table

| Module | Requirements | Verification Procedures |
|---|---|---|
| Power Supply (Key Chain) | 1. Supply steady 3.3V output voltage.<br>a) Able to provide 3.3V output voltage at various output current.<br><br>2. Able to last for a considerable amount of time (45 days).<br>a) The worst case scenario is tested assuming all components are consuming maximum amount of power.<br>b) The average case is examined with reasonable estimates and calculations. | 1. a)<br>- A test circuit will be built using the voltage output from the voltage regulator with different values of resistors<br>- Based on preliminary calculations, the current output for standby mode is around 100mA and will not exceed 300mA with all components active.<br>- Tests will be performed from 50mA to 300mA with 10mA intervals.<br>- Values of resistors are controlled to obtain the current values.<br>- The voltage across resistor will be measured using a multi-meter and recorded.<br>**Expected Results:** At all current values, the voltages measured should fall within the range of 3.25V to 3.35V.<br><br>2. (This step is calculated, instead of tested using bench equipment because it is not possible to monitor the battery for a long time over months. However, with reasonable assumptions, the calculated results should give a brief idea of the power consumption of the devices.)<br>a) Assuming all modules are consuming maximum power, calculate the time it is going to take before the battery drains. This should give us a brief idea of how long the device could last at least.<br>b) Estimate that an average user uses the key chain 6 times a day and each time the LEDs light up for 5 seconds. |

| | | |
|---|---|---|
| | | Also assume that the buzzer is used once every 3 days with duration of 15 seconds each time. Under these circumstances, calculate the operating current together with devices that continuously draws power. The result needs to be more than 45 days. |
| Power Supply (Lock) | 1. Supply steady 3.3V output voltage for entire circuit except the servo motor.<br>a) Able to provide 3.3V output voltage at various output current.<br>b) Supply required voltage for servo motor.<br><br>2. Same as "2." from Power Supply (Key Chain) section. | 1. a) Same as "1. a)" in Power Supply (Key Chain) section.<br>b)<br>- The servo motor operates best between 4.8V and 6V.<br>- The test will be set up using power supply from the 3.3V voltage output.<br>- The current through the servo motor will be measured using a multi-meter.<br>- The operation behavior will be monitored too.<br>**Expected Results:** If the current falls within the normal range suggested by the data sheet and the operation behavior of the servo motor is not abnormal, 3.3V will be used. If not, a different circuit will be used to provide power to the servo motor.<br><br>2. Same as "2." from Power Supply (Key Chain) section except with different estimations. The servo motor together with the Ethernet module is estimated to operate once every 6 days. The corresponding result should be more than 45 days. |
| LEDs | 1. The LEDs should be able to light up two distinguishably different colors. | 1.<br>- The test circuit will be set up using the bi-color LEDs with each input connected to a 65 Ohm resistor.<br>- 3.3V voltage will be applied to each pin of each LED.<br>**Expected Results:** The LED should |

| | | |
|---|---|---|
| | | light up red with input to the red pin and green with input to the green pin. These colors should match the intended color and should be easily distinguishable. |
| Buzzer | 1. Able to produce sound to be heard under variety of circumstances.<br>a) Able to hear sound within 15 meters in clear distance.<br>b) Able to hear sound within 10 meters in a separate room.<br>c) Able to hear sound within 10 meters with buzzer places in closed objects, such as drawers and boxes. | 1. a)<br>- The buzzer will be placed on a bread board with 3.3V input voltage.<br>- It will be triggered by one of the group members at 15 meters away from the rest.<br>**Expected Results:** The other two people have to be able to clearly hear the buzzer and determine its location.<br><br>b)<br>- The same buzzer circuit from part (a) will be triggered inside one of the rooms in Everitt Lab 10 meters away from where the other two members stand.<br>**Expected Results:** Both members have to be able to hear the sound clearly and determine which room it is in.<br><br>c)<br>- The same buzzer circuit will be triggered inside a drawer at home with two members standing 10 meters away.<br>**Expected Results:** Both members have to be able to hear the sound clearly and locate the buzzer. |
| Bluetooth Transceiver (Key Chain) | 1. The Bluetooth module on the key chain should be able to receive data reliably and quickly from the sender.<br>a) The Bluetooth module should receive data from sender without errors at distance between 0.1m to 15m.<br>b) The Bluetooth module should receive data without errors within closed objects or rooms.<br>c) No data is lost when the time | 1. a) b)<br>- The Bluetooth transceiver will be connected to an Arduino Uno board.<br>- The Arduino will be programmed to send 300 sets of data with 2 second intervals.<br>- The computer enabled with Bluetooth will be the receiver and all data received will be monitored on the screen.<br>**Expected Results:** All data should |

| | | |
|---|---|---|
| | interval between sending is short. | arrive and be displayed on the screen with no single error.<br><br>c)<br>- This is the same setup as described above.<br>- The Arduino will be sending 100 sets of data with 0.5 second interval.<br>**Expected Results:** All data should arrive and be displayed on the screen without losing any single one. |
| Bluetooth Transceiver (Lock) | 1. The Bluetooth module on the lock should be able to send data reliably and quickly from the sender.<br>a) The Bluetooth module should send data that is passed from the controller without errors.<br>b) No data is lost when the time interval between sending is short. | 1. a)<br>- The sender will be the same circuit above with the receiver being another Bluetooth module connected to an ATmega328 chip.<br>- 300 sets of data will be sent and the received data will be monitored on the screen.<br>**Expected Results:** All data should arrive and be displayed on the screen with no single error.<br><br>c)<br>- This is the same setup as described above.<br>- The Arduino will be sending 100 sets of data with 0.5 second interval.<br>**Expected Results:** All data should arrive and be displayed on the screen without losing any single one. |
| Servo Motor | 1. Able to rotate a specified angle correctly.<br><br>2. Able to turn lock's knob to perform lock and unlock. | 1.<br>- The servo motor will be connected with PWM output from the ATmega328 chip.<br>- The ATmega is programmed to send out a series of control command involving roting 10 to 180 degrees clockwise with 10 degree interval.<br>- The servo needs to return to its original |

| | | |
|---|---|---|
| | | position 5 seconds after each rotation.<br>- A protractor will be used to measure the angle turned and all measurements will be recorded.<br>- The same test is then performed with the servo motor rotating counter-clockwise.<br>**Expected Results:** All recorded data should be within 1degree tolerance range of the intended rotation.<br><br>2.<br>- The servo will be connected to a mechanical lock.<br>- The angle to be turned in order to lock and unlock the lock is recorded.<br>- The servo will receive command from the ATmega328 chip to turn the predetermined angle in order to lock and unlock the lock.<br>- 10 locks and 10 unlocks are performed and results are recorded.<br>**Expected Results:** All 20 operations should be completed with no failures meaning that each time the lock needs to be locked or unlocked matching the intentions. |
| Lock Sensor | 1. Should output correctly when the status of lock changes.<br><br>2. Should keep track of the status of lock with fast change of state, meaning when lock and unlock are performed with very short time interval. | 1<br>- The sensor will be installed on a mechanical lock.<br>- The lock will be manually locked and unlocked 100 times each.<br>- The output is observed on an oscilloscope.<br>**Expected Results:** The waveform on the oscilloscope should match correctly to the operations without a single error.<br><br>2.<br>- The same setup from part 1 will be |

| | | |
|---|---|---|
| | | used.<br>- The door will be manually locked but at a very fast speed by a group member.<br>- One other member will count the number of times locks and unlocks occurred.<br>**Expected Results:** The waveform on the oscilloscope should not miss any of the operations. |
| Ethernet Module | 1. Correctly receive information based on controls sent via computers or smart phones. | 1.<br>- The Ethernet module will be connected to the ATmega328 chip.<br>- 10 Sets of commands will be sent from an iPhone, an Android phone, an iPad and a computer each.<br>- The received command is decoded and displayed on screen.<br>**Expected Results:** All data on screen should match that sent from all devices without any error. |
| Controller (Key Chain) | 1. Detects button presses correctly.<br>a) Button needs to be debounced and respond correctly to fast pressings.<br><br>2. Able to display correct information by lighting up correct LEDs.<br>a) Correctly indicates the right key given the lock ID.<br>b) Correctly light up all LEDs based on stored information of lock status.<br><br>3. Able to store or output correct information | 1. a)<br>- The button will have one side being connected to a 3.3V voltage source and the other to an ATmega328 chip.<br>- The ATmega controller is preprogrammed with a debounce function and the output to an oscilloscope is high when button is pressed.<br>- The button will be manually pressed 100 times as fast as possible.<br>**Expected Results:** The output on the oscilloscope should indicate 100 presses.<br>2. a)<br>- The controller will be connected to a Bluetooth module.<br>- A total of 100 signals indicating lock IDs will be sent.<br>- The controller will generate signals light up the correct LED based on the |

| | | |
|---|---|---|
| | | ID.<br>- The generated signal will be monitored using an oscilloscope.<br>**Expected Results:** The output signal should match the intended output to the correct LED.<br><br>b)<br>- 10 sets of different combinations of the locks' status will be written into the microcontroller.<br>- The LEDs are going to be attached to the microcontroller as indicated in schematics 2 in page 8.<br>- The behavior of all 8 LEDs will be monitored and recorded.<br>**Expected Results:** The LEDs should exactly match the pre-written information about the locks for all 10 times.<br><br>3.<br>- Without the Bluetooth module attached, an input pin will be used to simulate the data input from the Bluetooth module.<br>- 20 different Commands including triggering buzzer and update information will passed to the controller.<br>- The controller will generate different output based on different information received.<br>**Expected Results:** The output should match the commands without single error. |
| Controller (Lock) | 1. Same as "1." from Controller (Key Chain) section.<br><br>2. Correctly generates output signals to be sent via Bluetooth and Ethernet. | 1. Same as "1." from Controller (Key Chain) section.<br><br>2. a) b) c)<br>- A function generator will be used to |

| | | |
|---|---|---|
| | a) Correctly generate output for updating status.<br>b) Correctly generate lock ID.<br>c) Correctly generate signal to trigger buzzer.<br>d) Lock or unlock. | generate square wave simulating the lock sensor and touch sensor.<br>- The output from the microcontroller will be monitored on a computer.<br>**Expected Results:** The output needs to show the exact command to be generated based on the input from different sensors without a single error. |

# Appendix C – Microcontroller Code (Lock Device #3)

```
#include <Servo.h>

int motorPin = 9;
int sensorPin = 10;
int btResetPin = 11;
int switchPin = 12;
int switchPin2 = 13;

Servo door3;

int lockID = 3;

boolean locked;
boolean currentLocked;

void setup()
{
  Serial.begin(9600);
  door3.attach(motorPin);
  pinMode(motorPin, OUTPUT);
  pinMode(sensorPin, INPUT);
  pinMode(btResetPin, OUTPUT);
  pinMode(switchPin, INPUT);
  pinMode(switchPin, INPUT);
  digitalWrite(btResetPin, LOW);
}

void loop()
{
  if(digitalRead(switchPin))
  {
    digitalWrite(btResetPin, HIGH);
    delay(5000);
    Serial.write(lockID);
    for(int i = 0; i <500; i++)
    {
      delay(10);
      if(Serial.available())
      {
        int val = Serial.read();
        if(val == 11)
        {
          if(locked)
```

```
        {
          door3.writeMicroseconds(1700);
          delay(500);
        }
        else
        {
          door3.writeMicroseconds(1300);
          delay(500);
        }
        door3.writeMicroseconds(1500);
      }
    }
  }
  digitalWrite(btResetPin, LOW);
}

else if(digitalRead(switchPin2))
{
  digitalWrite(btResetPin, HIGH);
  delay(5000);
  Serial.write(9);
  delay(1000);
  digitalWrite(btResetPin, LOW);
}

/*else if(Serial.available()>0)
{
  int val = Serial.read();
  if(val == 11)
  {
    if(locked)
    {
      door3.writeMicroseconds(1700);
      delay(500);
    }
    else
    {
      door3.writeMicroseconds(1300);
      delay(500);
    }
  }
}*/
door3.writeMicroseconds(1500);
currentLocked = digitalRead(sensorPin);
if(currentLocked == !locked)
{
```

```
    digitalWrite(btResetPin, HIGH);
    delay(5000);
    if(currentLocked)
      Serial.write(30);
    else
      Serial.write(31);
    locked = currentLocked;
    delay(1000);
    digitalWrite(btResetPin,LOW);
  }
}
```

# Appendix D – Microcontroller Code (Keychain)

```
boolean statusArray[8] = {false, true, true, true, false, false, true, true};
int switchPin = 5;
int switchPin2 = 4;
int redPin = 11;
int greenPin = 12;
int cPin = 10;
int ePin = 9;
int addrPin_2 = 8;
int addrPin_1 = 7;
int addrPin_0 = 6;
int lockID = 2;
int buzzerPin = 3;

int state = 0;

void setup()
{
  Serial.begin(9600);
  pinMode(switchPin, INPUT);
  pinMode(switchPin2, INPUT);
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(cPin, OUTPUT);
  pinMode(ePin, OUTPUT);
  pinMode(addrPin_2, OUTPUT);
  pinMode(addrPin_1, OUTPUT);
  pinMode(addrPin_0, OUTPUT);
  pinMode(buzzerPin, OUTPUT);

}

void loop()
{
  switch(state)
  {
    case 0: //standby
    if(digitalRead(switchPin))
    {
      display();
      delay(500);
      state = 1;

    }
```

```
else if(Serial.available()>0)
{
  int val = Serial.read();
  if(val == 9)
  {
    state = 3;
  }
  else if(val < 9 && val > 0)
  {
    indicate(val-1);
    delay(500);
    state = 2;

  }
  else if(val == 10)
  {
    update(0, false);
    delay(20);

  }
  else if(val == 11)
  {
    update(0, true);
    delay(20);

  }
  else if(val == 20)
  {
    update(1, false);
    delay(20);

  }
  else if(val == 21)
  {
    update(1, true);
    delay(20);

  }
  else if(val == 30)
  {
    update(2, false);
    delay(20);

  }
  else if(val == 31)
```

```
{
  update(2, true);
  delay(20);

}
else if(val == 40)
{
  update(3, false);
  delay(20);

}
else if(val == 41)
{
  update(3, true);
  delay(20);

}
else if(val == 50)
{
  update(4, false);
  delay(20);

}
else if(val == 51)
{
  update(4, true);
  delay(20);

}
else if(val == 60)
{
  update(5, false);
  delay(20);

}
else if(val == 61)
{
  update(5, true);
  delay(20);

}
else if(val == 70)
{
  update(6, false);
  delay(20);
```

```
  }
  else if(val == 71)
  {
    update(6, true);
    delay(20);

  }
  else if(val == 80)
  {
    update(7, false);
    delay(20);

  }
  else if(val == 81)
  {
    update(7, true);
    delay(20);

  }
  Serial.flush();
}
break;

case 1://display
  delay(4000);
  displayOff();
  state = 0;
break;

case 2://indicate
  for(int i = 0; i < 400; i++)
  {
    delay(10);
    if(digitalRead(switchPin2))
    {
      Serial.write(11);
      delay(20);
      break;
    }
  }
  displayOff();
  state = 0;
break;

case 3://buzzer
buzzer();
```

```
    delay(20);
    state = 0;
    break;
  }
}

void update(int id, boolean lock)
{
  statusArray[id] = lock;
}

void buzzer()
{
  int buzzerCount = 0;
  boolean buzzerOn = true;
  while(buzzerOn && buzzerCount<30)
  {
    digitalWrite(buzzerPin, HIGH);
    delay(1000);
    buzzerCount++;
    if(digitalRead(switchPin))
      buzzerOn = false;
  }
  digitalWrite(buzzerPin, LOW);
  delay(500);
}

void displayOff()
{
  for(int i = 0; i < 8; i++)
  {
    boolean a2;
    boolean a1;
    boolean a0;

    if((i>>2)&1 == 1)
      a2 = true;
    else a2 = false;
    if((i>>1)&1 == 1)
      a1 = true;
    else a1 = false;
    if(i&1 == 1)
      a0 = true;
    else a0 = false;

    boolean red = false;
```

```
    boolean green = false;

    digitalWrite(cPin, HIGH);
    digitalWrite(ePin, HIGH);
    digitalWrite(redPin, red);
    digitalWrite(greenPin, green);
    digitalWrite(addrPin_2, a2);
    digitalWrite(addrPin_1, a1);
    digitalWrite(addrPin_0, a0);
    digitalWrite(cPin, HIGH);
    digitalWrite(ePin, LOW);
    delay(20);
    digitalWrite(cPin, HIGH);
    digitalWrite(ePin, HIGH);
  }
}

void indicate(int id)
{
  for(int i = 0; i < 8; i++)
  {
    boolean s = statusArray[i];
    boolean a2;
    boolean a1;
    boolean a0;

    if((i>>2)&1 == 1)
      a2 = true;
    else a2 = false;
    if((i>>1)&1 == 1)
      a1 = true;
    else a1 = false;
    if(i&1 == 1)
      a0 = true;
    else a0 = false;

    boolean red;
    boolean green;

    if(id == i)
    {
      if(s)
      {
        red = true;
        green = false;
      }
```

```
      else
      {
        red = false;
        green = true;
      }
    }

    else
    {
      red = false;
      green = false;
    }


    digitalWrite(cPin, HIGH);
    digitalWrite(ePin, HIGH);
    digitalWrite(redPin, red);
    digitalWrite(greenPin, green);
    digitalWrite(addrPin_2, a2);
    digitalWrite(addrPin_1, a1);
    digitalWrite(addrPin_0, a0);
    digitalWrite(cPin, HIGH);
    digitalWrite(ePin, LOW);
    delay(20);
    digitalWrite(cPin, HIGH);
    digitalWrite(ePin, HIGH);
  }
}

void display()
{
  for(int i = 0; i < 8; i++)
  {
    boolean s = statusArray[i];
    boolean a2;
    boolean a1;
    boolean a0;

    if((i>>2)&1 == 1)
      a2 = true;
    else a2 = false;
    if((i>>1)&1 == 1)
      a1 = true;
    else a1 = false;
    if(i&1 == 1)
      a0 = true;
```

```
    else a0 = false;

    boolean red;
    boolean green;

    if(s)
    {
      red = true;
      green = false;
    }
    else
    {
      red = false;
      green = true;
    }

    digitalWrite(cPin, HIGH);
    digitalWrite(ePin, HIGH);
    digitalWrite(redPin, red);
    digitalWrite(greenPin, green);
    digitalWrite(addrPin_2, a2);
    digitalWrite(addrPin_1, a1);
    digitalWrite(addrPin_0, a0);
    digitalWrite(cPin, HIGH);
    digitalWrite(ePin, LOW);
    delay(20);
    digitalWrite(cPin, HIGH);
    digitalWrite(ePin, HIGH);
  }
}
```