University of Moratuwa

Faculty of Engineering

Department of Electronic and Telecommunication Engineering

PRODUCT REPORT

BUZZ LOCK

S. KOWRISAAN	200312L
H.L. KULATUNGA	200312L
V. KURRSHANTH	200331T
U.J.R. LEELANANDA	200339B

Group 12

EN1190

CONTENTS

EXECUTIVE SUMMARY

Buzz Lock is a passcode-protected cable lock for compatible appliances (Bicycle, Gate), which alarms in an instance of misconduct like cutting off the cable, external pull of the cable, and wrong passcode input while in the locked state.

This report contains all the related aspects of the product design and development from origin of the idea to initial ideas, finalized solution, specification, hardware and software used, PCB design, enclosure design, budget of the product and the final tests and conclusion.

PRODUCT INTRODUCTION

Buzz Lock

This is a passcode protected cable lock for compatible appliances (Bicycle, Gate), that alarms in an instance of misconduct like cutting off the cable, or external pull of the cable and can be unlocked with a passcode.

Origin of the project idea

Our group members Kurrshanth and Kowrisaan are from the Eastern and Northern provinces of Sri Lanka where bicycles are used by most people, especially students. Kurrshanth stated that an increase in the robbery of bicycles was recorded in his neighborhood. Kowrisaan also mentioned there were similar cases in his neighborhood also.

When we inquired the other students from areas where bicycles were used most, many confirmed that the same problem also exists in their areas. Then, with inductive reasoning, we decided this could be a common problem throughout the country since the usage of bicycles has increased due to the prevailing fuel shortage, and the economic crisis prompted desperate thieving. Further interactions with the end-users revealed the robbery of other items like water pumps in gardens and generators in the vicinity of their home.

Motivation for the product

The prevailing economic crisis in the country resulted in fuel shortage and never-ending queues in fuel stations. People are unable to get fuel for their vehicles for transportation. This led people to go for alternative transportation like bicycles and electric vehicles. Since most people can't afford electric vehicles, bicycles have become the new popular transportation and a high demand exists in the market for bicycles. Unlike before bicycles are valuable assets now, and they are prone to get robbed. Since the country's situation is not going to get back to normal anytime sooner, we can assume that the prices of bicycles will continue to grow up, and it will be difficult for the general public to buy new bicycles. And concerns about the existing security for their bicycle will also keep growing. Since most of the imports are interrupted by the ongoing situation, the prices of most of the appliances have increased and robberies of houses, garden water pumps, and generators are often reported. Even though there are existing locks for bicycles, gates, and pumps they are often being cut down by robbers with wire cutters and easily get robbed.

A need for a more secure lock that can be used for bicycles, gates, and other compatible appliances has arisen in the market due to these problems. The survey that has been conducted among the end-users revealed this need and confirmed its existence.

Initial ideas for the solutions & Constraints

We came up with two initial solutions,

➢ Creating a tracking device

➢ Creating a stronger lock

But even though there are strong bicycle locks in the market, they are being cut down no matter how strong they are. So, we concluded that whatever the tracking device or lock we create the robber always can destroy or malfunction it. The robber will always have the confidence to do it if there is a quiet environment with no one around. So, integrating a tracking device or a much stronger locking mechanism is deemed to be not the best solution.

Finalized solution

There must be a way to break the confidence of a thief by any means when he tries to cut the lock. There is a fair chance that the robber will be demotivated to steal the bicycle and the neighbors will be alerted if an alarm is triggered when trying to rob. By abductive reasoning, we assumed a lock that alerts the surrounding when a robber tries to destroy it will discourage robbers from stealing things. So, we came up with the idea of creating a lock that alarms in an instance of misconduct like cutting off the cable, or external pull of the cable and can be unlocked with a passcode.

We asked several bicycle users whether they would spend money on a lock with the above characteristics. Most of them said yes as they cannot afford to buy a new bicycle these days with their skyrocketing prices. Then to add more user-friendly aspects to the lock we asked bicycle users about other problems they faced with the existing locks. Some suggested they lose the key to the lock often, thus by deductive reasoning the definite solution was to provide a key-less lock with passcode protection.

Specifications

Earshot of the alarm	20m
Cable material	metal
Water and dust resist	Yes
Battery	9V × 3
Battery lifetime	1 year
Weight	400 grams
Dimensions (Length: Width: Height)	8cm: 7cm: 4cm
Cable length	50cm
Interfaces	Input keypad (4 keys) Output screen (OLED)
Warranty	1 year
Maintenance	An annual check-up is recommended (Free for the first 2 years)

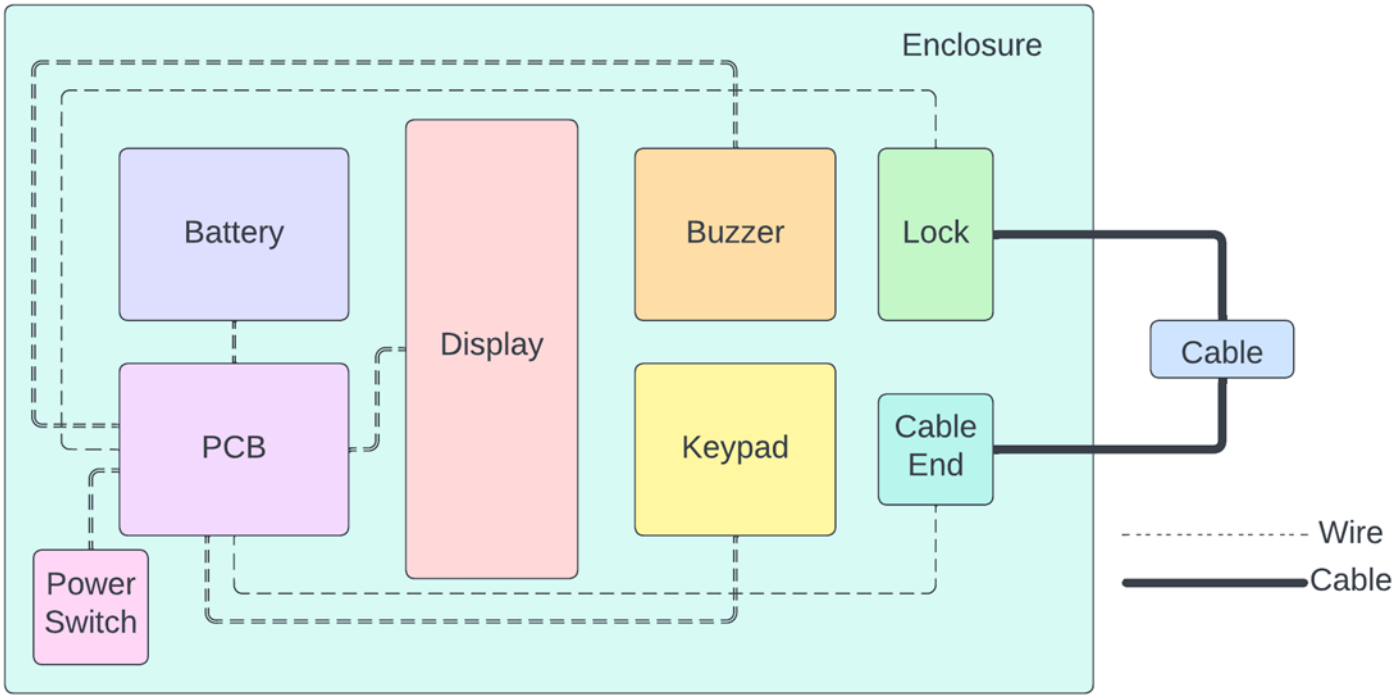
Hardware used

Atmega 328p IC	Solenoid lock
4 key membrane keypad	OLED display
Buzzer	Oscillator
22 PF capacitors	Resistors
Connectors and wires	Battery holders

Software used

Altium Designer (PCB Design)	Solidworks (Enclosure design)
Microchip studio (IC programming)	Proteus (Simulations)

Designing & Manufacturing process



Block diagram representation

PCB

Contains the IC and other surface-mounted and socketed components (Capacitors, Resistors, Oscillator) and makes the connection between the socketed components and external components and battery.

SOCKETED COMPONENTS

- ATMEGA328P-PU IC
- Oscillator
- 22 PF capacitors
- Resistors (330Ω x2, 220Ω, 10kΩ x2)
- LEDs
- Power regulator (5V LM7805)
- Transistor
- IN4001 Diode
- Capacitors (Polarized 100nF x3, 339nF; Ceramic 22pF x2)
- Push button
- Connectors

KEYPAD

Gets the user input for passcode. 4 key membrane keypad

DISPLAY

Displays instructions and feedbacks to the user. OLED display

BUZZER

Acts as the alarm. Triggered according to the control signals from the IC.

Lock

Solenoid lock that acts as the locking and unlocking mechanism. Remains locked when no power is given.

BATTERY

Supplies power to every component through the PCB. This block contains battery holders and 9V battery.

POWER SWITCH

Used to turn the power on or off to the display and keypad when needed to save power.

CABLE END

The cable's fixed end with the inner wire is fixed here and connected to the PCB.

CABLE

Strong cable that runs through the appliance that needs to be locked. Contains the wire that runs through the inside which makes the circuit. One end is fixed, and the other is used as the locking end and the circuit connector.

ENCLOSURE

Every component other than the cable is inside the enclosure. Enclosure is waterproof, shockproof.

ENVISIONING THE DESIGN

The product idea is to develop a cable lock to be used for bicycles and other compatible appliances that has an alarming capability when trying to cut or break the cable. Naturally this requires a cable with electrical conductivity, a circuit to detect the connection breaks and trigger the alarm and a suitable power supply. Then we planned to integrate a passcode protection for locking which required us to add a keypad and a display. Then for saving power we had to implement a power switch for the keypad and display to terminate power while not in use. Every component except the cable must be enclosed inside a protective enclosure that is robust and waterproof. The enclosure must be designed to accommodate every component inside it while minimizing the overall size and weight of the product. The cable should also be made of a strong material despite the fact it triggers the alarm when it gets cut. We came up with the initial sketch for the lock keeping these requirements in mind.

PROTEUS SIMULATION

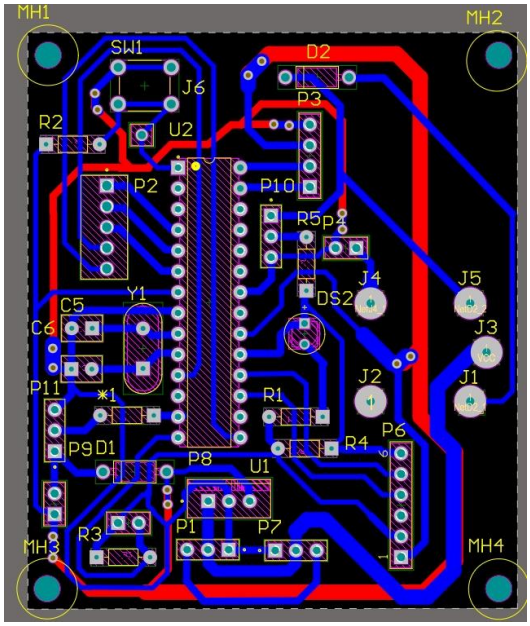
Initial circuits were implemented in proteus simulation software. All of the modules including the microcontroller, display, keypad, buzzer, and lock were tested through the software beforehand using the initial code written for the product and the code was updated and debugged during the simulation process.

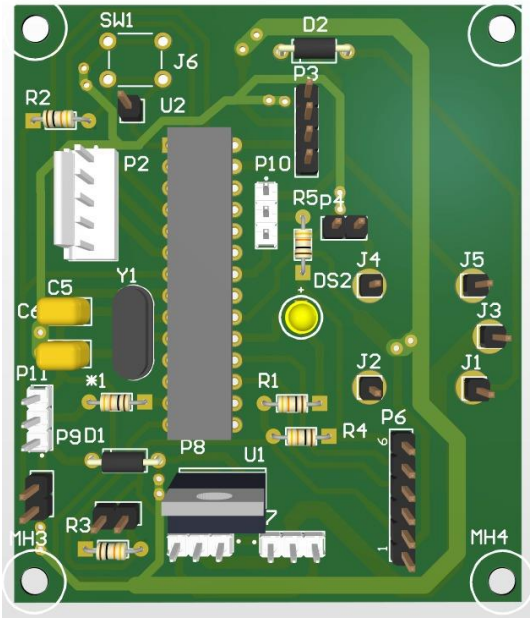
Breadboard Implementation with the Microcontroller

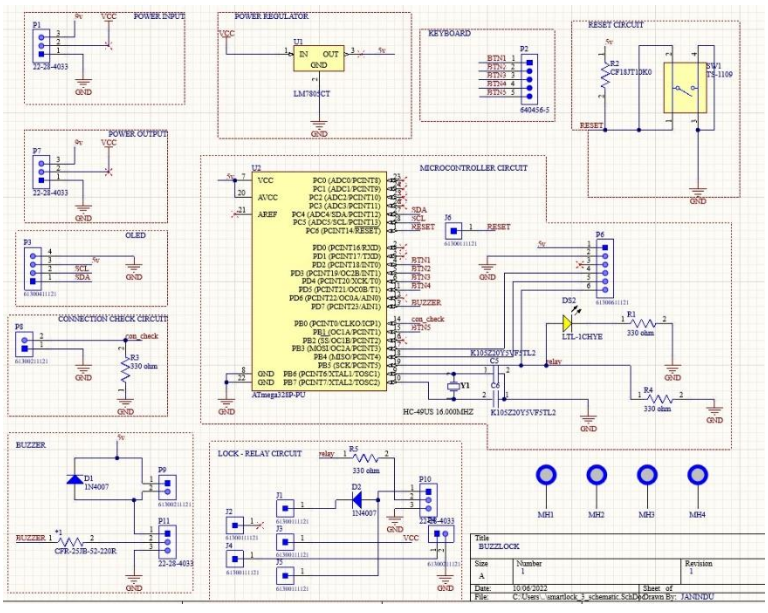
After getting successful results from the simulations, we implemented the design on breadboard using Atmega328p microcontroller and all other components. We used Arduino Uno to program the Atmega328p.

PCB DESIGN

PCB design was done using Altium Designer software. All the connections and components were first simulated in proteus software and was checked with the program. Then the whole circuit was designed on a breadboard to verify the functionality. A 2-layer PCB layout was designed to make the PCB more compact so that it will fit in to the physical compartment appropriately. Board shape was redesigned to be a 7cm x 8cm rectangular board. 0.7 mm was selected as the minimum trace width and for the power lines 1 mm trace width was used. Width of VCC was selected to be 1.5 mm to cater high current ratings. Minimum clearance was set to be 0.75 mm and pad width was selected to be 1.8 mm in most components to ensure better soldering. 3D layout was checked to find out whether components clash with each other and all the components were placed on a paper model to ensure that components exactly match with the footprints on the PCB.



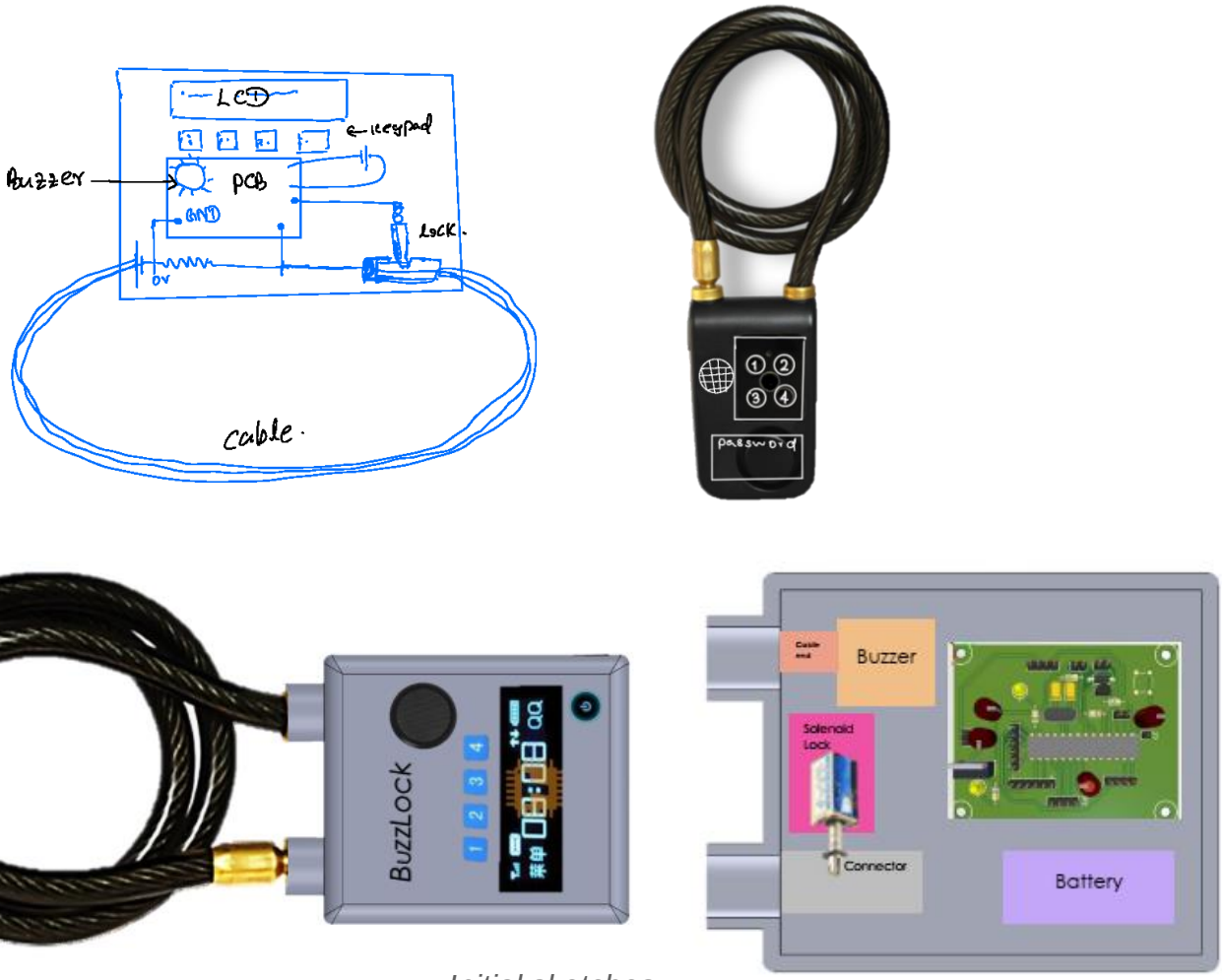




Final PCB design and Schematic

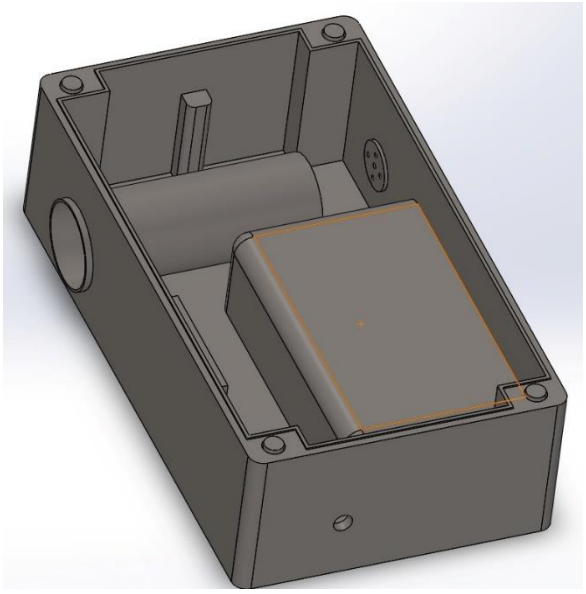
ENCLOSURE DESIGN

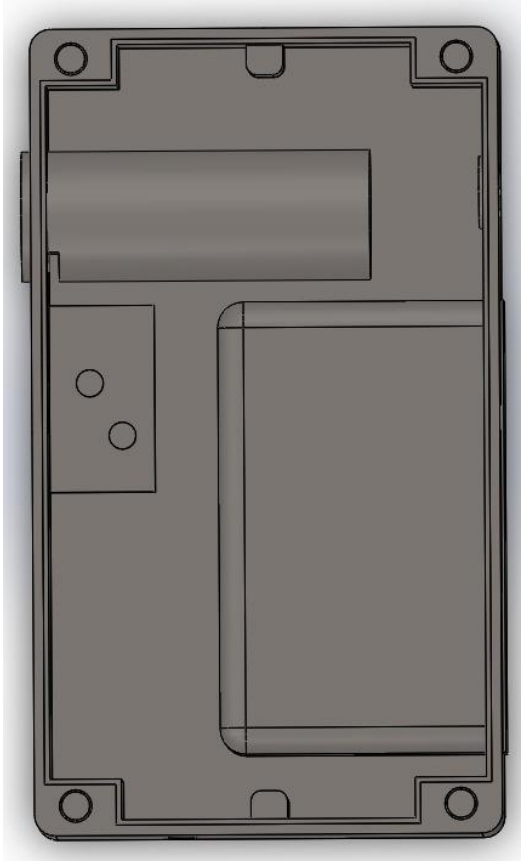
As for the enclosure we came up with the initial sketches shown below, considering the requirements of the product.

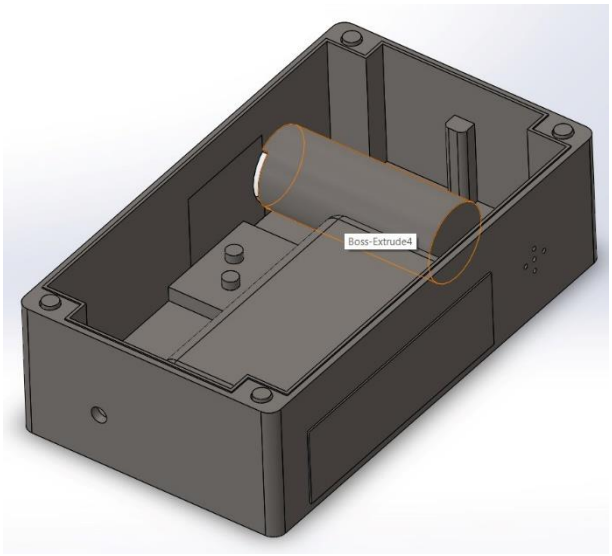


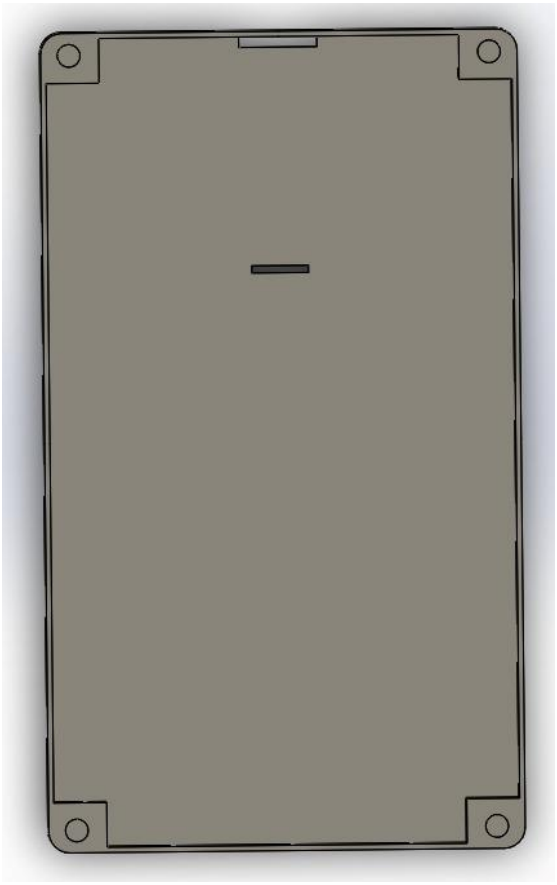
Initial sketches

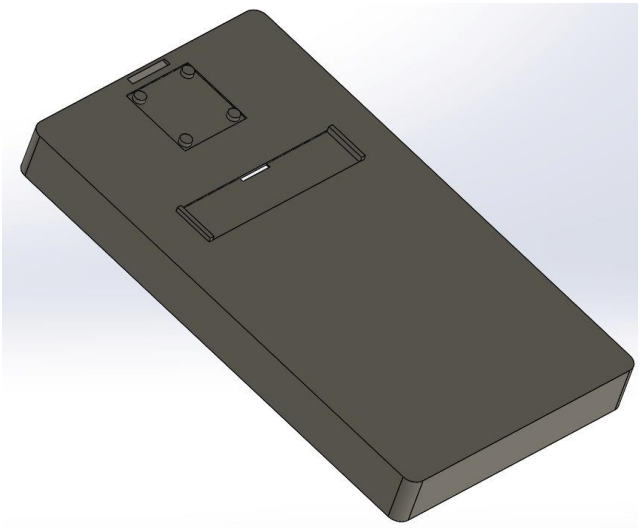
After coming up with initial sketches for our design, we modeled it using solid works. The design consisted of two layers, one for the PCB (the upper compartment) and the lower compartment for the battery chamber, solenoid and a cylindrical structure which allows the lock to be positioned. Two layers were separated by a plastic sheet, which was cut using laser cutting technology. The PCB was mounted on it. Wire holes were positioned to get wires from the lower compartment to the upper compartment. The face of the enclosure was installed with an OLED panel and a four-digit keypad. A wire was taken out from the lower compartment and the end of it was fixed with the pin of the lock. The enclosure is made of strong PVC plastic, which can withstand shock and sudden impulses.



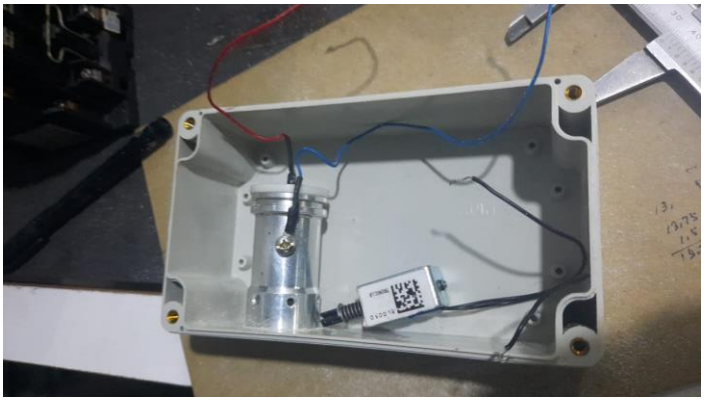


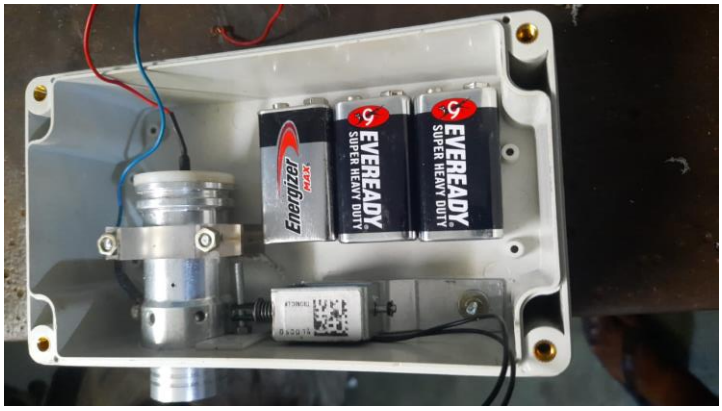


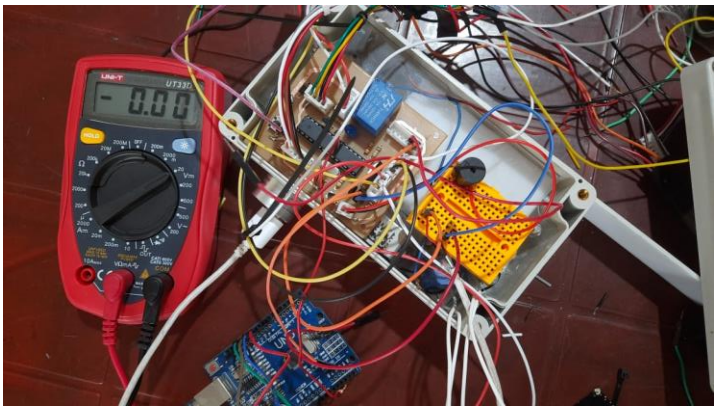




Final Solidworks design







Final Prototype Making

UI DESIGN

User will be interacting with the product for locking and unlocking the lock. User interface comes into play when unlocking the lock. User must be able to get instructions and feedback for his action. The built-in display shows the instructions to input the password once the power button is turned on. User's input numbers will be displayed and if it is the correct passcode it displays 'unlocked' while unlocking the lock and displays 'wrong passcode' otherwise. The alarm will be triggered after 15s of wrong passcode input if not the lock is unlocked within 15s with the correct passcode.

The user can input the passcode by the 4 key membrane keypad. Keys in the keypad are clearly visible and responsive. User must turn on the power button for the display and keypad to turn on. This instruction is printed above the power button to assist the user.

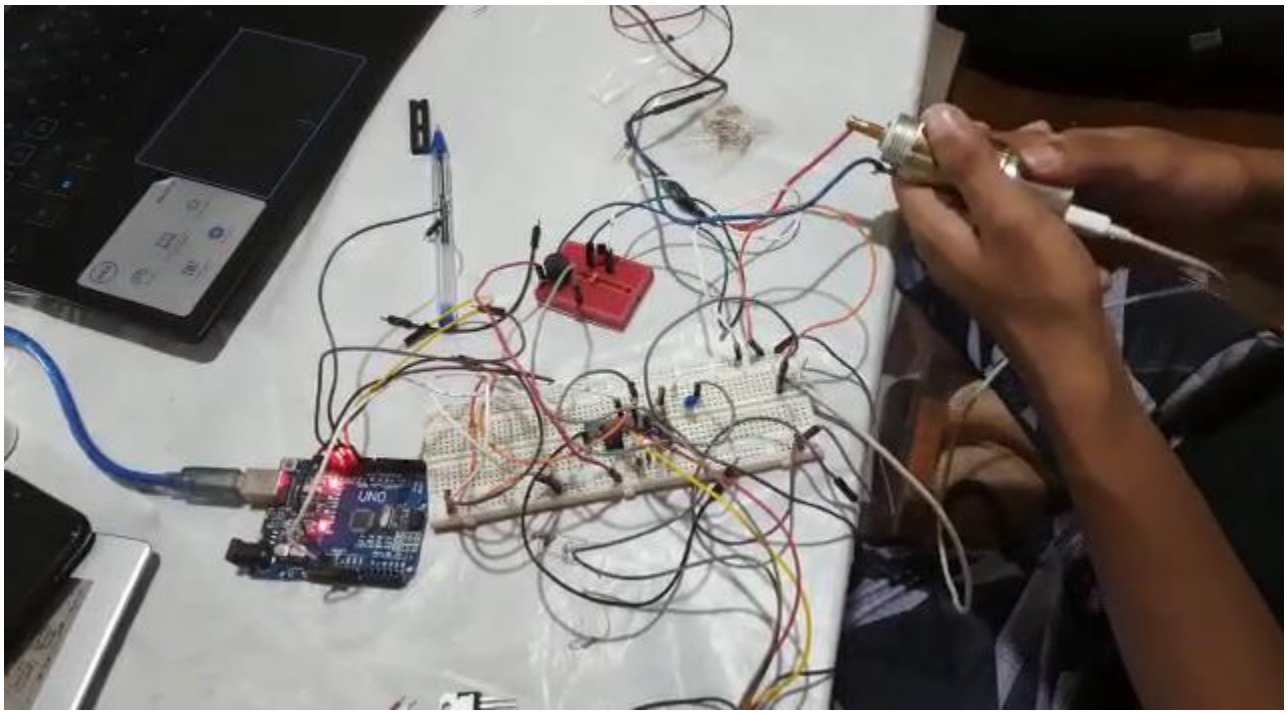
The user is given with an initial password, and he is instructed by the user manual to change is as soon as possible. Explaining briefly how the code is implemented, it uses the memory available in the Atmega328P chip to enable the device to update its password and to remember it even after a power loss. The user has two options to select at the home screen.

1. Enter password
2. Update password.

If the user press number 1 then he will be taken to a new window where he can type his password. If the user press 2 then he will be asked to type his old password. If he enters a wrong password then, he will be thrown back to the home window. But if he enters his old password correctly then a new window will pop up asking him to enter his new password. Then after 4 digits gets completed, he will be informed that his password has been changed.

FUNCTIONAL PROTOTYPE

The initial implementation of the product was done using Arduino Uno.



COST OF MODULES AND COMPONENTS AND PRODUCT PRICE

Technical Components & Others	**Cost**
Atmega 328p IC	1700
4 Keys membrane keypad	250
Buzzer	50
22 pF capacitors	20
Solenoid lock	450
OLED display	600
Oscillator	30
Battery	200
Battery holders	20
Connectors and wires	20
PCB	80
Enclosure (Material)	300
Packages (Sealing & packaging material)	40
Total cost for product at factory level (T1)	**3760**

The above chart is based on the actual cost for the product to make BUZZLOCK as a device. But beyond that, the following costs are to be considered.

Service Costs	**Cost**
Components assembling cost	20
Packing cost for machine/human	30
Transport	30
Total cost (assembling, packing, transport) (T2)	**80**

Therefore, Finalized Project Budget **= T1 + T2 = 3840 Rs**
And, Finalized price of BUZZLOCK Device **= T1 + T2 + Profit = 4840 Rs**
We are decided to provide **5% discounts** at the market for Sri Lankan people considering the economic crisis.
So, **Final market price of BUZZLOCK Device = 4600 Rs**
Comparing to the current bicycle prices (50 000 – 60 000 Rs) the product price is quite reasonable and affordable.

MARKETING, SALES, AND BEYOND

Manufacturing

We have decided to limit manufacturing quantity to 50 devices (BUZZLOCK) at a time. Continuing the success of the initial batch we plan to increase the production for the next batch and distribute the product island-wide. It will take 48 hours to finish the whole manufacturing process. After that 50 products will be shipped to the Testing Unit to check whether it gives the expected outputs while test cases (inputs) are given as inputs. In the testing unit, passed devices are shipped to the packaging unit and other devices will be sent for reassembling the components for reuse.

Product packaging

PROTECTION

Since our product is an electronic-based product, the constraints are related to the endurable temperature, level of sunlight exposure, and vibrations that our product can withstand.

The device is packaged inside a rectangular box (wood fiber or paper pulp) with custom colors and our printed logo on it. A bubble polyethylene cover (or bubble wrap) is used to isolate it from the package. This will provide protection from vibrations while shipping and will protect from excess heat from the environment and static electricity.

When our product is shipped, there will be another layer of protection by a similar wrap over the bubble wrap. Doubling the packaging will help to improve the safety while shipping the product.

APPEARANCE

We have planned to separate the cable and the other enclosure part by packing them in separate bubble wraps and to put both with the user manual guide inside a package box that is made of wood fiber or paper pulp (Black and White). The dimensions of the box are length:8.5 cm, width:7.5 cm, and height:6 cm. The box will have a 2-part lift-offs hard-lidded box. To print the product name, we use the “ARIAL BLACK” font with white color. On the package box, there will be a seal with transparent polyethylene.

INFORMATION

The front face of the box will contain

- Attractive images of final product
- Name of the product
- Name & address of manufacturer

- Durability details

The back and sides of the box will contain

- All the factory details (battery capacity, power consumption, microchips used in the product, power consumption, etc.)
- Universal store code
- Assembly instructions
- Instructions for storing the product

EASE OF USE

There will be a user guide with the packaging to guide the users with the installation operation of the product. Providing a good-looking user guide with understandable pictures will be much easier for the user to get an understanding of the product on their own.

STORAGE

We are decided to go with the rectangular box with size (0.5 cm more than the size of the enclosure in width and length and 2 cm more in length). There will be no empty space inside the packaging so we can avoid issues like shaking, which may occur at shipping or transporting. And, by making package box enough small without any space inside the box, we will be able to transport more packages. And having a rectangular box (rather than having a round or any shapes) is better to transport many products to market. It's also very easy for distributors.

PROMOTION

On the product package box, the product name (as "BUZZLOCK"), the company logo, and the name are mentioned. So, we are planning to do a promotion using the name and logo.

Marketing

THE TARGET CUSTOMERS

- Students and other people of all ages who use bicycles and who want protection for Gates, and water pumps from robbery.

  In the current situation of Sri Lanka, there are many people who use a bicycle for daily transport due to the fuel crisis and electrical failure. So, people use a bicycle more frequently than earlier and the price of bicycles also is hiking day by day. Therefore, it should be protect against robbery.
- More specifically, people who are interested in smart ways of protection
- People who are interested in simplicity and multipurpose of products.

  It can be used to provide safety on gates, water pumps, door locks and other physical appliances.

COMPETITORS AND COMPETITIVE PRODUCTS

- Omni – Wireless Waterproof Bike Wheel Electric Smart Bicycle alibaba.com/Bike-Smart-Lock

- Cycle zone - Bike, Motorcycle, Electric Bike alibaba.com/Lock-Smart-Bike-Fingerprint-U-shaped

- MYPIN – Fingerprint Bike Lock Cable fingerprint-bike-lock-cable

THE DEVICE BUZZ LOCK WILL BE LAUNCHED AND PROMOTED THROUGH

Social media

- Making introduction video clip about our product mentioning the details, purpose of uses, etc. and post it on Instagram, Facebook, YouTube, Twitter, etc.

Making public attention: -

- Having a cycling race and giving our device to the participators in public places like markets, schools and providing other applications that can be used.

Using influencers
Word of mouth
Company website

- Creating a webpage with a whole details of product

SALES

Considering the current increase in bicycle usage, this is the best time to market the product.
The selling and distribution methods will be

- Sold through company website and other online shopping pages (Daraz, Amazon, Alibaba)
- Customers will be able to order the product on web pages and that will be delivered at door. Until the product is shipped to the delivery address, customers can track where it is and which stage it's in.
- Sold in local market

  It will be available at hardware shops, Electronics products store like SINGER, LG, ABANS.

While buying the product, customization options will be available (colors) at online platforms and local market.

Maintenance

There will be electronic and electrical device service centers authorized by our production company in the selected areas. There, users will be able to clarify anything about the product. And maintenance of the device is recommended annually (It will be provided free for the first 2 years).

Product Life cycle

REPAIRS

If the product is not working properly, customers can inquire with the employers in the service center that is authorized, and it will be monitored by the BUZZLOCK Management panel. They will fully check the device component-wise and find where the problem is. And they will fix those problems. Unless they could repair it, they will check the warranty period (while buying a device, the company provides a 1-year Warranty) and they will replace the device with a new one or reimburse the amount if the warranty period is not exceeded.

REUSE/RECYCLE

When the device is given as replaced model, the employer will check the device thoroughly and identify the working components. The components that are working will be separated for recycling and reusing purposes. Therefore, the separated usable components will be used to replace the damaged components in other devices. From that, we will be able to reuse/recycle the usable components again and again.

DISPOSAL

If the damaged devices and components are found by the service center, they will be donated to schools and church for education purposes or given to the Certified E-Waste Recycler.

PRODUCT TEST AND RESULTS

We conducted tests on the product during the manufacturing process, separately on each module to ensure the if the modules are working, and on the final product, after the completion of the product.

Quality of the result

The results were the same as anticipated during the design process. Mainly, the locking mechanism, the keypad inputs, display outputs and the buzzer were functional with the expected outcome. All the components of the product we secured inside the enclosure and the enclosure was quite sturdy. Consumer needs identified during the market study were met by the final product sufficiently.

CONCLUSION

We conclude that our final product has provided the required solution to the initial problem we hoped to address. But there is more room for improvement of the product according to the market study and consumer reviews after the initial distribution of the product into the market.

APPENDICES

References

ATMEGA328P-PU Datasheet
Guide for I2C OLED Display with Arduino
Solenoid Valves – Rotork
bc548 Data Sheet

Code

```
// Include Arduino Wire library for I2C
#include <Wire.h>
#include <string.h>

#include <EEPROM.h>
using namespace std;



#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

Adafruit_SSD1306 buzzlock = Adafruit_SSD1306(128,64,&Wire);

// Include LCD display library for I2C

//#include <LiquidCrystal_I2C.h>
// Include Keypad library
#include <Keypad.h>

//Getting line input
int LineValue = 1;
int Line = 8;
int Buzzer = 7;

// Length of password + 1 for null character
#define Password_Length 5
// Character to hold password input
char Data[Password_Length];
// Password
char Master[Password_Length] = "1234";
```

```
//The new password that is read from the readMemmory function
char loaded_password[Password_Length];









// Pin connected to lock relay input
int lockOutput = 13;

// Counter for character entries
byte data_count = 0;
byte space_count = 0;
byte data_count_2 = 0;
byte space_count_2 = 0;


// Character to hold key input
char customKey;
char customKey_menu;

// Constants for row and column sizes
const byte ROWS = 1;
const byte COLS = 4;

// Array to represent keys on keypad
char hexaKeys[ROWS][COLS] = {
  {'1', '2', '3', '4'}
};

// Connections to Arduino
byte rowPins[ROWS] = {9};
byte colPins[COLS] = {5, 4, 3, 2};

// Create keypad object
Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

// Create LCD object
//LiquidCrystal_I2C lcd(0x3F, 16, 2);
//LiquidCrystal_I2C lcd(0x27,20,4);
void setup() {
  pinMode(Line, INPUT);
  pinMode(Buzzer, OUTPUT);



  buzzlock.begin(SSD1306_SWITCHCAPVCC, 0x3C);

  // Setup LCD with backlight and initialize
```

```
  //lcd.backlight();
  //lcd.init();

  // Set lockOutput as an OUTPUT pin
  pinMode(lockOutput, OUTPUT);
}


void setDisplay()
{
  buzzlock.clearDisplay();
  buzzlock.setTextColor(WHITE);
  buzzlock.setTextSize(1.5);
  buzzlock.setCursor(0,5 );

}

//The function that takes the password from EEPROM and assigns it to the loaded_password
variable.

void readFromMemmory(){

  for (int i =0; i <4; i++){
   loaded_password[i] = EEPROM.read( i);

  }

}

void updateMemmory(char new_password[5]){
  for (int i =0; i <4; i++){
   EEPROM.update( i, new_password[i]);

  }
}




void loop() {

  LineValue = digitalRead(Line);


  if(LineValue){
            EEPROM.write(5, false);
            digitalWrite(Buzzer, LOW);

            // Initialize LCD and print
```

```
        //  lcd.setCursor(0, 0);
        //  lcd.print("Enter Password:");
          setDisplay();
          buzzlock.print("Press,");
          buzzlock.setTextSize(1);
          buzzlock.setCursor(0,20);
          buzzlock.print("1 to Enter password");

          buzzlock.setTextSize(1);
          buzzlock.setCursor(0,35);
          buzzlock.print("2 to Update password");

          // Look for keypress
          customKey_menu = customKeypad.getKey();
          //customKey = customKeypad.getKey();

          // Work space for Selection Menu


        if (customKey_menu == '1'){
           setDisplay();
           buzzlock.setCursor(0, 1);
           buzzlock.print("Enter  Password");
           buzzlock.display();
           while (true){
             if(digitalRead(Line) == 0){
               clearData();
               break;
             }
             customKey = customKeypad.getKey();
             if (customKey) {
               // Enter keypress into array and increment counter
               Data[data_count] = customKey;
               buzzlock.setTextColor(WHITE);
               buzzlock.setTextSize(1.5);
               buzzlock.setCursor(space_count, 15);
               buzzlock.print("*");
               buzzlock.display();
               delay(300);
               data_count++;
               space_count = space_count + 20;
             }

             if (data_count == Password_Length - 1){

                 readFromMemmory();
                 if (!strcmp(Data, loaded_password)){
                   EEPROM.write(5,true);
```

```
      digitalWrite(lockOutput, HIGH);
      setDisplay();
      buzzlock.print("Correct");
      buzzlock.display();
      delay(500);

      for (int i=10; i>0; i--)
      {
        setDisplay();
        buzzlock.print("Locking in " );
        buzzlock.setCursor(0,20 );
        buzzlock.print(i);
        buzzlock.setCursor(10, 20);
        buzzlock.print(" s" );
        buzzlock.display();

        delay(1000);
      }

      digitalWrite(lockOutput, LOW);

      if ( digitalRead(Line)){
          EEPROM.write(5,false);
       }



      // Turn on relay for 5 seconds


      setDisplay();
      buzzlock.print("Locked");
      buzzlock.display();
      delay(500);


    }
    else{
      setDisplay();
      buzzlock.print("Incorrect password");
      buzzlock.display();
      delay(1000);

    }

  buzzlock.clearDisplay();
  while (data_count != 0) {
    Data[data_count--] = 0;
  }
```

```
            while (space_count != 0) {
              Data[space_count--] = 0;
            }
            break;


        }

      }
```

```
}
else if (customKey_menu == '2')
{
  setDisplay();
  buzzlock.setCursor(0, 1);
  buzzlock.print("Enter Old Password");
  buzzlock.display();
  while (true){
    if(digitalRead(Line) == 0){
      clearData();
      break;
    }
    customKey = customKeypad.getKey();
    if (customKey) {
      // Enter keypress into array and increment counter
      Data[data_count] = customKey;
      buzzlock.setTextColor(WHITE);
      buzzlock.setTextSize(1.5);

      buzzlock.setCursor(space_count, 15);
      buzzlock.print("*");
      buzzlock.display();
      delay(300);
      data_count++;
      space_count = space_count + 20;
    }

    if (data_count == Password_Length - 1){
```

```
                readFromMemmory();
                if (!strcmp(Data, loaded_password)){
                  setDisplay();
                  buzzlock.setCursor(0, 1);
                  buzzlock.print("Enter New Password");
                  buzzlock.display();
                  while (true){
                    if(digitalRead(Line) == 0){
                      clearData();
                      break;
                    }
                    char customKey_update = customKeypad.getKey();
                    if (customKey_update){
                      Data[data_count_2] = customKey_update;
                      buzzlock.setTextColor(WHITE);
                      buzzlock.setTextSize(1.5);
                      buzzlock.setCursor(space_count_2, 15);
                      buzzlock.print(customKey_update);
                      buzzlock.display();
                      delay(300);
                      data_count_2++;
                      space_count_2 = space_count_2 + 20;


                    }

                    if (data_count_2 == Password_Length - 1){
                      updateMemmory(Data);

                      setDisplay();
                      buzzlock.print("Changes saved!");
                      buzzlock.display();
                      delay(300);

                      while (data_count_2 != 0) {
                        Data[data_count_2--] = 0;
                      }
                      while (space_count_2 != 0) {
                        Data[space_count_2--] = 0;
                      }
                      break;

                    }



                  }

                }else{
```

```
                    setDisplay();
                    buzzlock.print("Incorrect Password");
                    buzzlock.display();
                    delay(600);
                  }

                  while (data_count != 0) {
                    Data[data_count--] = 0;
                  }
                  while (space_count != 0) {
                    Data[space_count--] = 0;
                  }
                  break;

                }


              }

            }









            delay(100);
            buzzlock.display();

    }else{
      if (EEPROM.read(5)){
        setDisplay();
        buzzlock.display();
      }
      else{
        digitalWrite(Buzzer, HIGH);

      }
    }
}

void clearData() {
  // Go through array and clear data
  while (space_count != 0) {
    Data[space_count--] = 0;
  }
  while (data_count != 0) {
    Data[data_count--] = 0;
  }
```

```
  return;

}
// Include Arduino Wire library for I2C
#include <Wire.h>
#include <string.h>

#include <EEPROM.h>
using namespace std;



#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

Adafruit_SSD1306 buzzlock = Adafruit_SSD1306(128,64,&Wire);

// Include LCD display library for I2C

//#include <LiquidCrystal_I2C.h>
// Include Keypad library
#include <Keypad.h>

//Getting line input
int LineValue = 1;
int Line = 8;
int Buzzer = 7;

// Length of password + 1 for null character
#define Password_Length 5
// Character to hold password input
char Data[Password_Length];
// Password
char Master[Password_Length] = "1234";


//The new password that is read from the readMemmory function
char loaded_password[Password_Length];







// Pin connected to lock relay input
int lockOutput = 13;

// Counter for character entries
byte data_count = 0;
byte space_count = 0;
```

```
byte data_count_2 = 0;
byte space_count_2 = 0;


// Character to hold key input
char customKey;
char customKey_menu;

// Constants for row and column sizes
const byte ROWS = 1;
const byte COLS = 4;

// Array to represent keys on keypad
char hexaKeys[ROWS][COLS] = {
  {'1', '2', '3', '4'}
};

// Connections to Arduino
byte rowPins[ROWS] = {9};
byte colPins[COLS] = {5, 4, 3, 2};

// Create keypad object
Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

// Create LCD object
//LiquidCrystal_I2C lcd(0x3F, 16, 2);
//LiquidCrystal_I2C lcd(0x27,20,4);
void setup() {
  pinMode(Line, INPUT);
  pinMode(Buzzer, OUTPUT);



  buzzlock.begin(SSD1306_SWITCHCAPVCC, 0x3C);

  // Setup LCD with backlight and initialize
  //lcd.backlight();
  //lcd.init();

  // Set lockOutput as an OUTPUT pin
  pinMode(lockOutput, OUTPUT);
}


void setDisplay()
{
  buzzlock.clearDisplay();
  buzzlock.setTextColor(WHITE);
  buzzlock.setTextSize(1.5);
  buzzlock.setCursor(0,5 );
```

```
}

//The function that takes the password from EEPROM and assigns it to the loaded_password
variable.

void readFromMemmory(){

  for (int i =0; i <4; i++){
   loaded_password[i] = EEPROM.read( i);

  }

}

void updateMemmory(char new_password[5]){
  for (int i =0; i <4; i++){
   EEPROM.update( i, new_password[i]);

  }
}




void loop() {

  LineValue = digitalRead(Line);


  if(LineValue){
            EEPROM.write(5, false);
            digitalWrite(Buzzer, LOW);

            // Initialize LCD and print
          //  lcd.setCursor(0, 0);
          //  lcd.print("Enter Password:");
            setDisplay();
            buzzlock.print("Press,");
            buzzlock.setTextSize(1);
            buzzlock.setCursor(0,20);
            buzzlock.print("1 to Enter password");

            buzzlock.setTextSize(1);
            buzzlock.setCursor(0,35);
            buzzlock.print("2 to Update password");

            // Look for keypress
            customKey_menu = customKeypad.getKey();
```

```
 //customKey = customKeypad.getKey();

 // Work space for Selection Menu


if (customKey_menu == '1'){
   setDisplay();
   buzzlock.setCursor(0, 1);
   buzzlock.print("Enter  Password");
   buzzlock.display();
   while (true){
     if(digitalRead(Line) == 0){
       clearData();
       break;
     }
     customKey = customKeypad.getKey();
     if (customKey) {
       // Enter keypress into array and increment counter
       Data[data_count] = customKey;
       buzzlock.setTextColor(WHITE);
       buzzlock.setTextSize(1.5);
       buzzlock.setCursor(space_count, 15);
       buzzlock.print("*");
       buzzlock.display();
       delay(300);
       data_count++;
       space_count = space_count + 20;
     }

     if (data_count == Password_Length - 1){

         readFromMemmory();
         if (!strcmp(Data, loaded_password)){
           EEPROM.write(5,true);

           digitalWrite(lockOutput, HIGH);
           setDisplay();
           buzzlock.print("Correct");
           buzzlock.display();
           delay(500);

           for (int i=10; i>0; i--)
           {
             setDisplay();
             buzzlock.print("Locking in " );
             buzzlock.setCursor(0,20 );
             buzzlock.print(i);
             buzzlock.setCursor(10, 20);
             buzzlock.print(" s" );
```

```
                buzzlock.display();

                delay(1000);
              }

              digitalWrite(lockOutput, LOW);

              if ( digitalRead(Line)){
                  EEPROM.write(5,false);
               }



              // Turn on relay for 5 seconds


              setDisplay();
              buzzlock.print("Locked");
              buzzlock.display();
              delay(500);


            }
            else{
              setDisplay();
              buzzlock.print("Incorrect password");
              buzzlock.display();
              delay(1000);

            }

          buzzlock.clearDisplay();
          while (data_count != 0) {
            Data[data_count--] = 0;
          }
          while (space_count != 0) {
            Data[space_count--] = 0;
          }
          break;


        }

      }
```

```
}
else if (customKey_menu == '2')
{
  setDisplay();
  buzzlock.setCursor(0, 1);
  buzzlock.print("Enter Old Password");
  buzzlock.display();
  while (true){
    if(digitalRead(Line) == 0){
      clearData();
      break;
    }
    customKey = customKeypad.getKey();
    if (customKey) {
      // Enter keypress into array and increment counter
      Data[data_count] = customKey;
      buzzlock.setTextColor(WHITE);
      buzzlock.setTextSize(1.5);

      buzzlock.setCursor(space_count, 15);
      buzzlock.print("*");
      buzzlock.display();
      delay(300);
      data_count++;
      space_count = space_count + 20;
    }

    if (data_count == Password_Length - 1){
      readFromMemmory();
      if (!strcmp(Data, loaded_password)){
        setDisplay();
        buzzlock.setCursor(0, 1);
        buzzlock.print("Enter New Password");
        buzzlock.display();
        while (true){
          if(digitalRead(Line) == 0){
            clearData();
            break;
          }
          char customKey_update = customKeypad.getKey();
          if (customKey_update){
            Data[data_count_2] = customKey_update;
```

```
        buzzlock.setTextColor(WHITE);
        buzzlock.setTextSize(1.5);
        buzzlock.setCursor(space_count_2, 15);
        buzzlock.print(customKey_update);
        buzzlock.display();
        delay(300);
        data_count_2++;
        space_count_2 = space_count_2 + 20;


      }

      if (data_count_2 == Password_Length - 1){
        updateMemmory(Data);

        setDisplay();
        buzzlock.print("Changes saved!");
        buzzlock.display();
        delay(300);

        while (data_count_2 != 0) {
          Data[data_count_2--] = 0;
        }
        while (space_count_2 != 0) {
          Data[space_count_2--] = 0;
        }
        break;

      }



    }

  }else{
    setDisplay();
    buzzlock.print("Incorrect Password");
    buzzlock.display();
    delay(600);
  }

  while (data_count != 0) {
    Data[data_count--] = 0;
  }
  while (space_count != 0) {
    Data[space_count--] = 0;
  }
  break;
```

```
                }


              }

            }









            delay(100);
            buzzlock.display();

    }else{
      if (EEPROM.read(5)){
        setDisplay();
        buzzlock.display();
      }
      else{
        digitalWrite(Buzzer, HIGH);

      }
    }
}

void clearData() {
  // Go through array and clear data
  while (space_count != 0) {
    Data[space_count--] = 0;
  }
  while (data_count != 0) {
    Data[data_count--] = 0;
  }
  return;

}
```