University of Moratuwa

Department of Electronic and Telecommunication Engineering

EN2160 – Electronic Design Realization

Project Report – Package Delivery Safe Box

Name : A. C. Pasqual
Index No. : 200445V

25th July 2023

Table of Contents

1. Product Description

Getting packages delivered to the home is a frequent practice in modern life, and its popularity is increasing with services such as online shopping. A common issue faced by customers using delivery services is receiving a package when no one is at home to accept it. In this scenario the delivery person might leave the package outside, which will lead to it getting stolen.

The product designed to address this issue is a password-locked box to safekeep packages delivered to a house when no one is at home. The box should be fixed outside the house similar to a letter box. A one-time password can be set and communicated to the delivery person to open the box and leave the package.

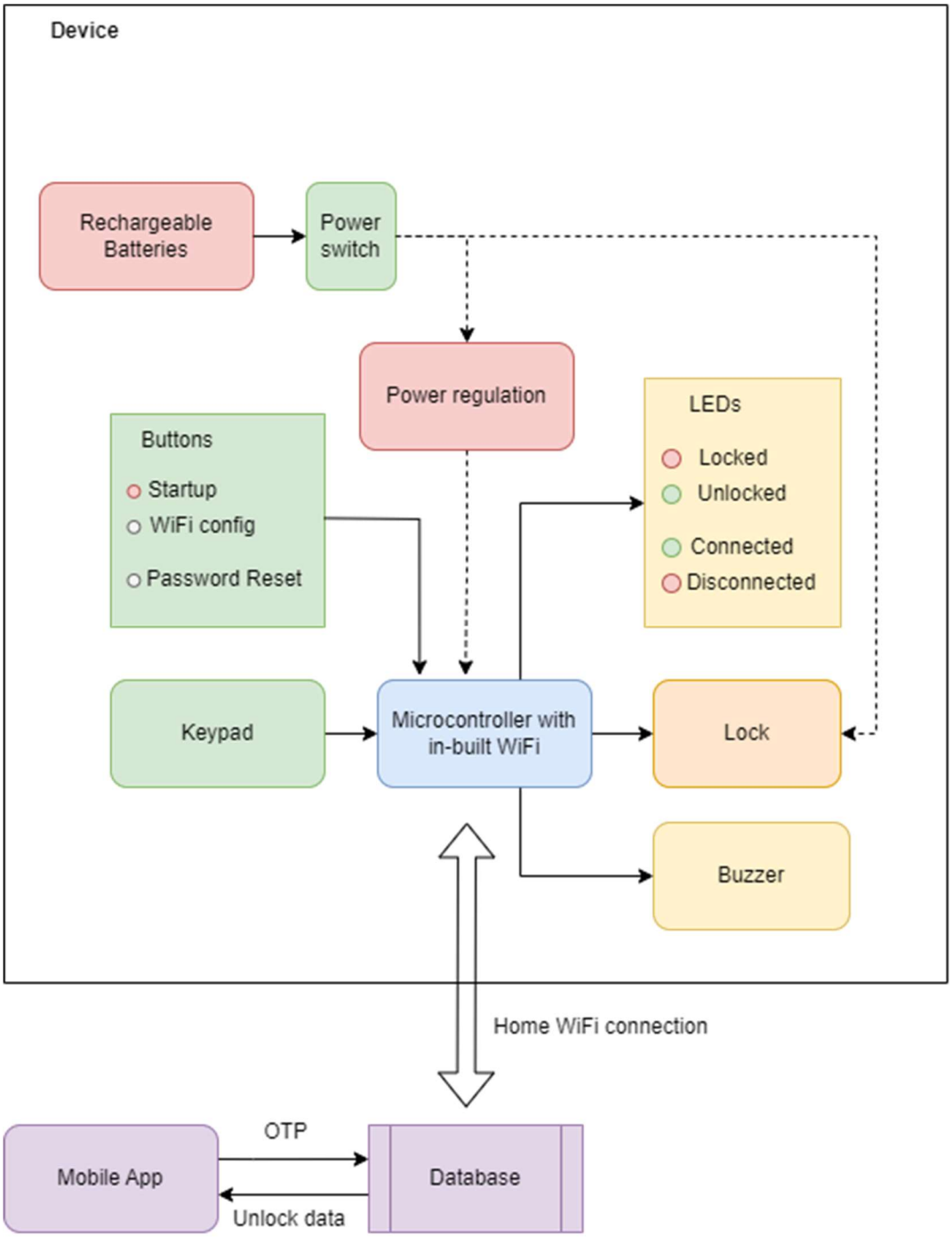
1.1 Features

- The box contains a password lock, which has a master password for the owner and a one-time password (OTP) if enabled.
- The OTP can be set remotely from a mobile app when a third party needs to unlock the box.
- The owner can open the box using the master password.
- The history of unlocking the box can be viewed in the mobile app, and a notification will be sent if the box is unlocked using the OTP.
- For security, the master password can only be changed from the device.
- The device will obtain the OTP from the mobile app using a Wi-Fi connection. It can be connected to the home Wi-Fi network through instructions provided in the app.
- The box can be used as a password-locked safe box even when Wi-Fi is disconnected.

1.2 Specifications

- External dimensions : 44 x 31 x 49 cm (L x W x H)
- Internal dimensions (storage area) : 43 x 30 x 39 cm (L x W x H)
- Power supply : 3 x 3.7 V rechargeable batteries

2. Block Diagram of the Device Architecture



3. Component Selection and Circuit Design

3.1 Microcontroller

The device required Wi-Fi connectivity to communicate with the mobile app from any location. Therefore, using a microcontroller with built-in Wi-Fi functionality is convenient for the design. An ESP32 microcontroller (ESP32-WROOM-32D chip) was chosen for this purpose as it is commonly available and has the following features:

- Faster Wi-Fi connectivity
- Several different sleep modes that can be used for power saving
- Has a sufficient number of input/output pins

3.2 Lock

Two main electronically controlled lock designs were considered:

- Solenoid lock
- Servo motor attached to a locking mechanism

The servo motor based design was eliminated as it has a low strength, and designing a proper locking mechanism for it would increase the complexity and cost of the design. The solenoid lock is stronger and is available for a lower cost.

Figure: Solenoid Lock

As the solenoid lock operates at 12 V, a switching circuit was designed using an N-channel MOSFET in order to control the lock from the microcontroller. The IRL520N MOSFET was selected for this based on the current draw of the lock and the threshold switching voltage.

3.3 Power Supply

As the box must be placed outside the house, powering it with AC power will be inconvenient for some users. Therefore 3 3.7 V rechargeable lithium-ion batteries (to provide 12V for the solenoid lock) are used as the power supply. Since the batteries cannot be charged in place, the batteries can be removed for charging. The voltage is regulated to 3.3V with an LD1117V33 voltage regulator and provided to the microcontroller.

3.4 Keypad

A 3x4 matrix keypad was selected. This will allow users to enter numbers and also provide “ENTER” and “CLEAR” functions.

3.5 LEDs

Four LED indicators are used to communicate the state of the lock and the Wi-Fi connection status. This will indicate responsiveness to user actions as well as being useful for troubleshooting.

Lock:

- Red LED on – Locked
- Green LED on – Unlocked
- Green LED blinking – in password reset mode

Wi-Fi connection:

- Red LED on – Trying to connect
- Red LED blinking – Not connected
- Green LED on – Connected

3.6 Buzzer

The buzzer is used to give feedback to the user's key presses and indicate whether the password is correct or incorrect. A 5 V passive buzzer was selected for this purpose.

4. Schematic and PCB Layout Design

The schematic and layout design for the PCB was done using the Altium Designer software.

4.1 Schematic Design

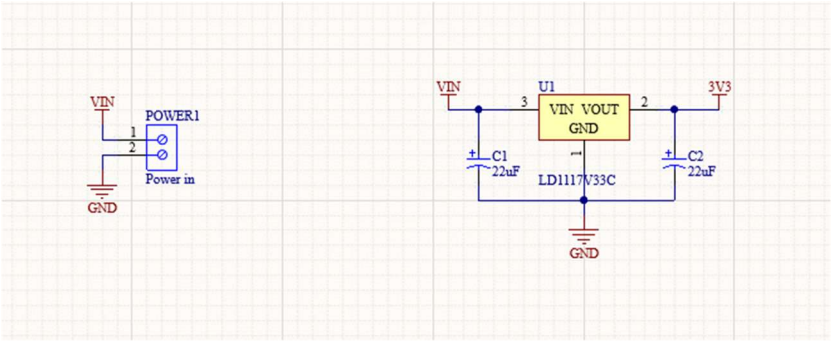


Figure: Power Circuit

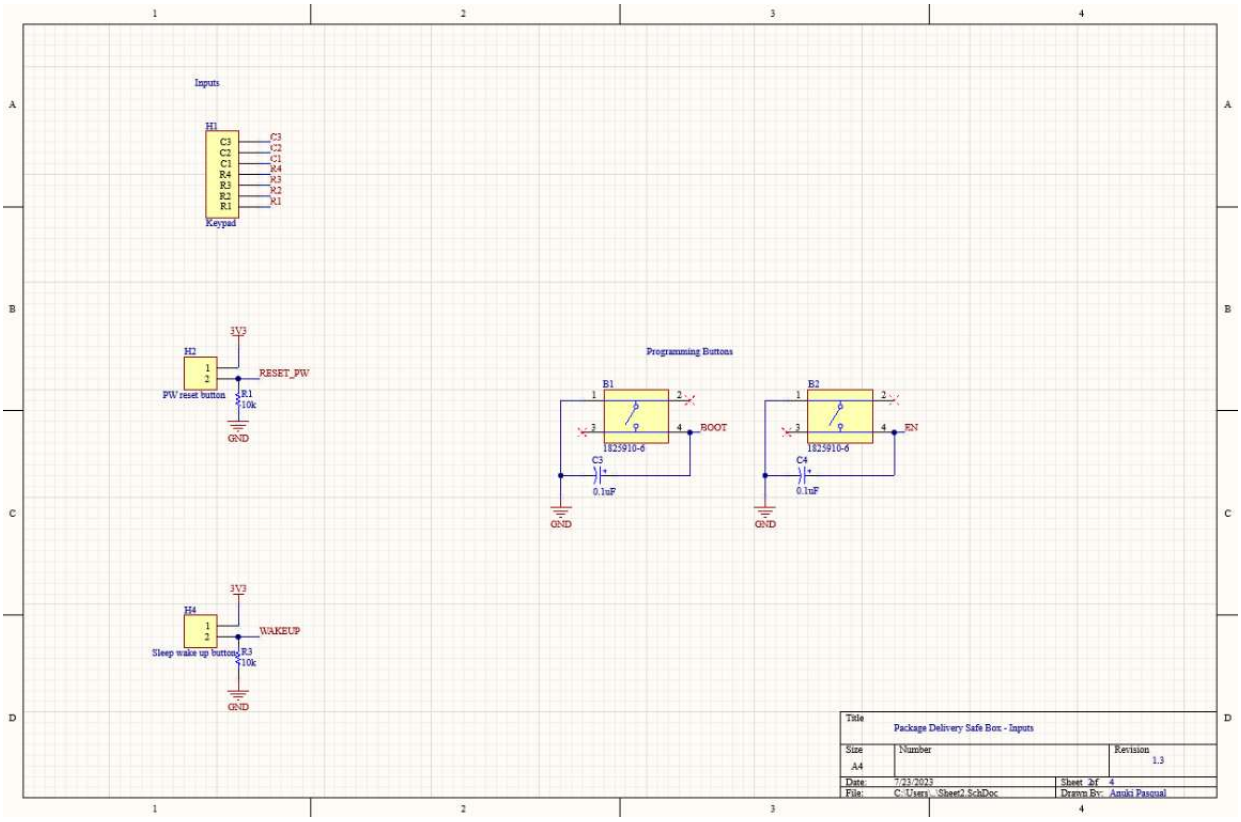


Figure: Inputs

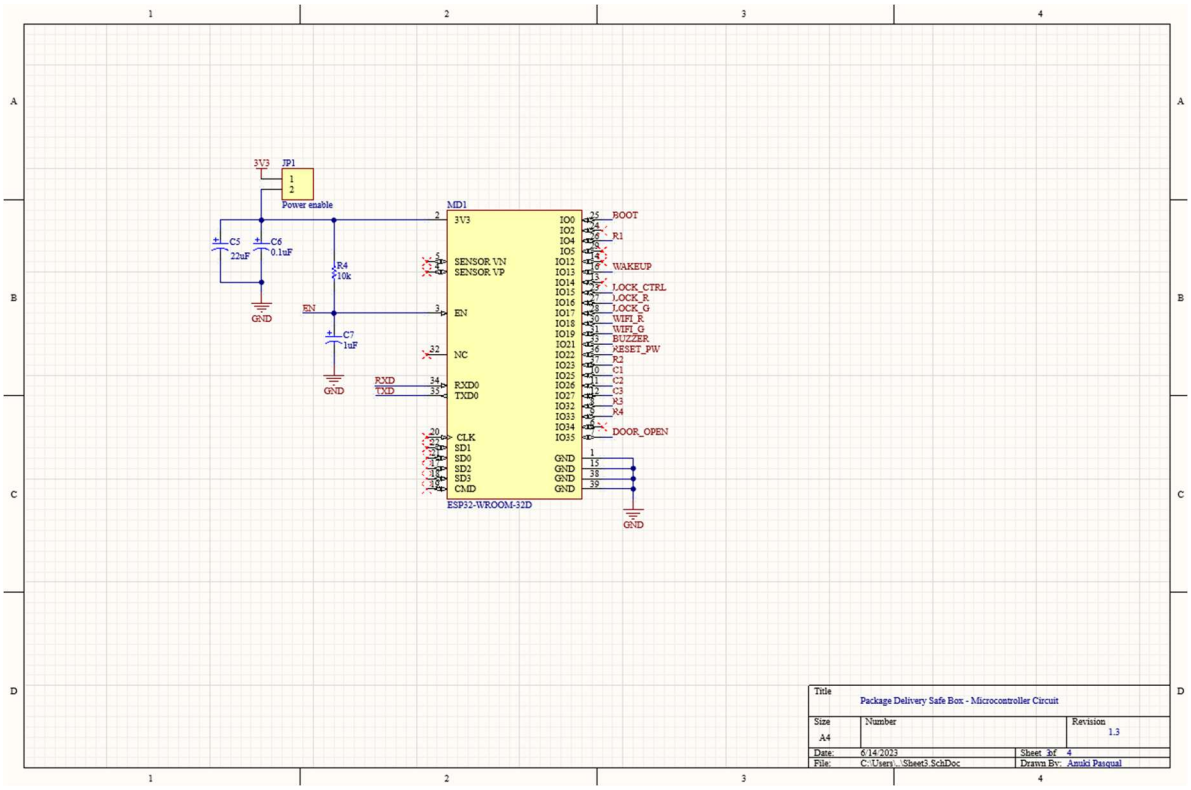


Figure: Microcontroller Circuit

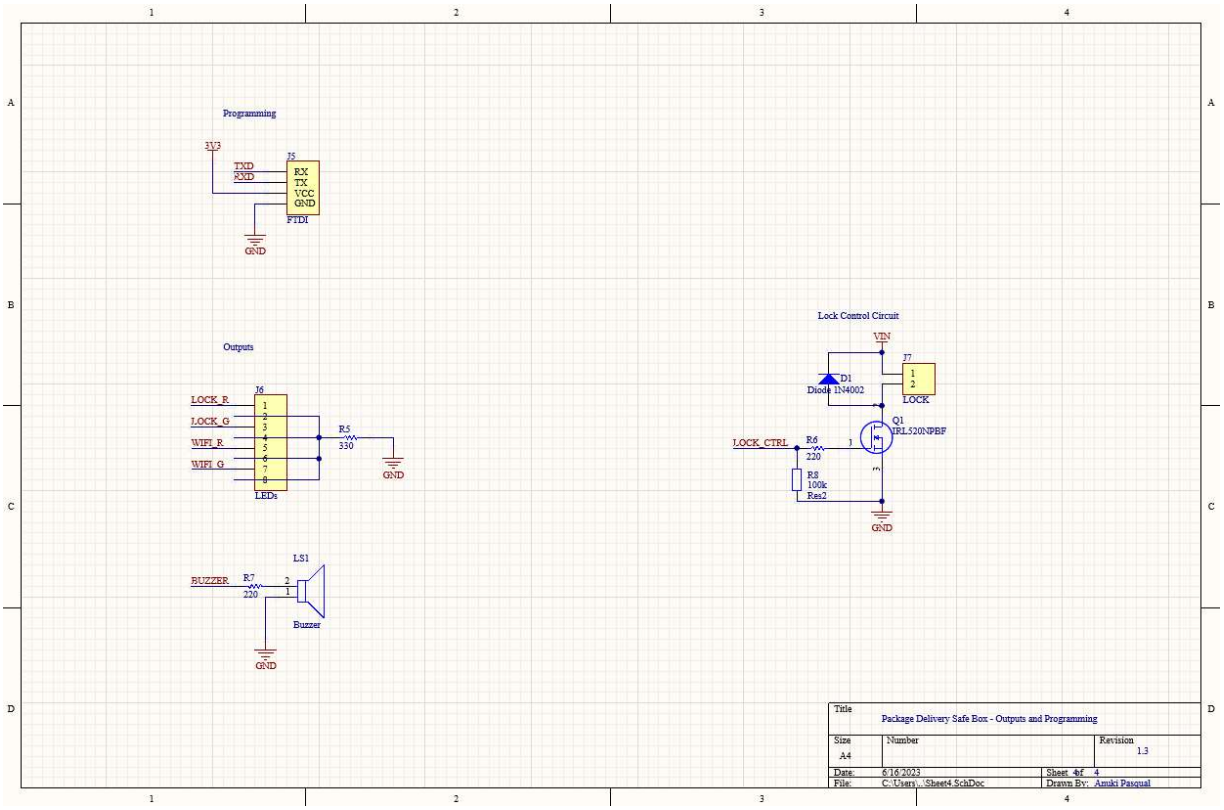


Figure: Outputs and Programming pins

4.2 PCB Layout Design

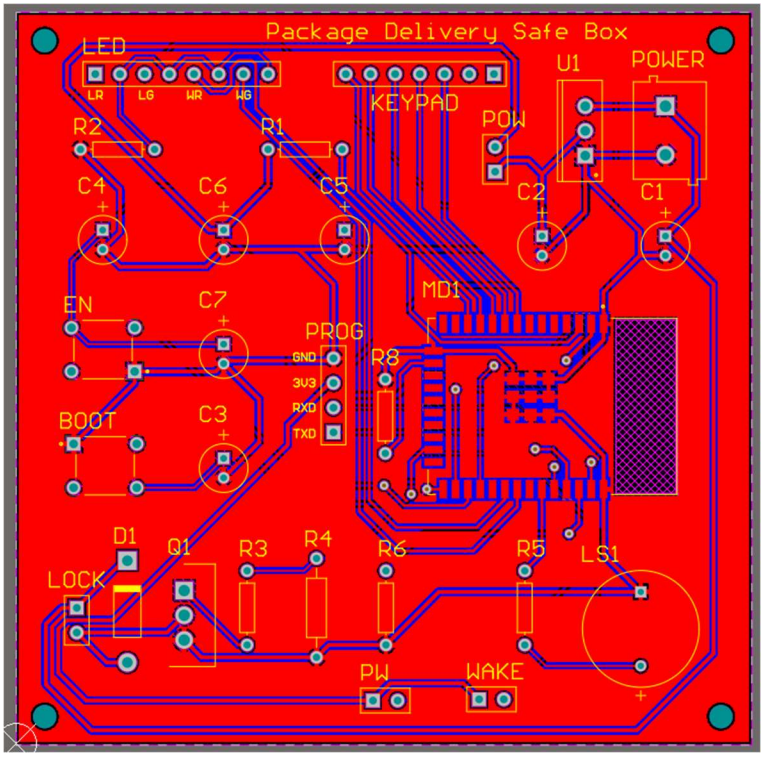


Figure: Top Layer

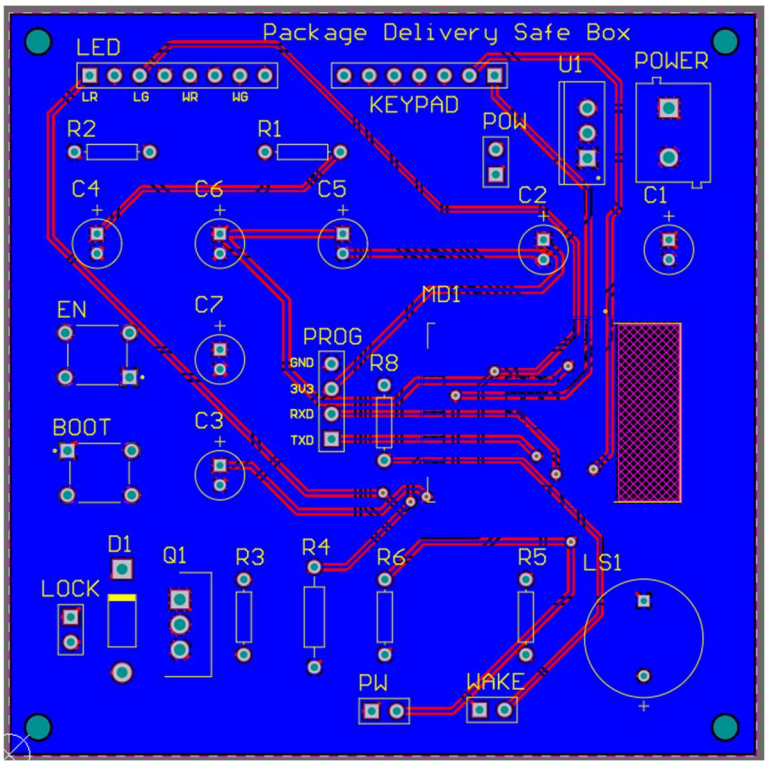


Figure: Bottom Layer

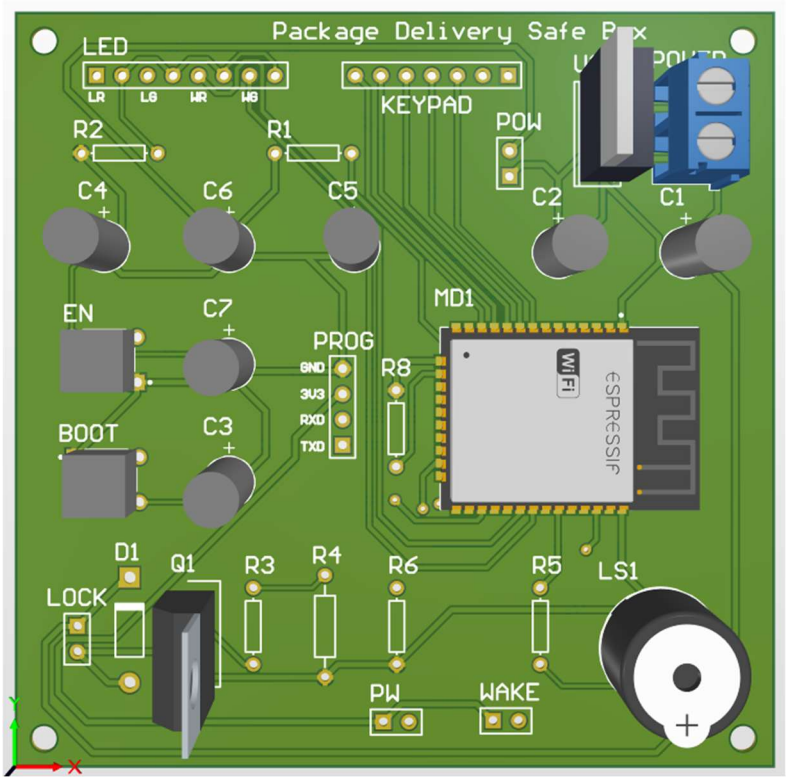


Figure: 3D view

4.3 Gerber Files and Drill Files

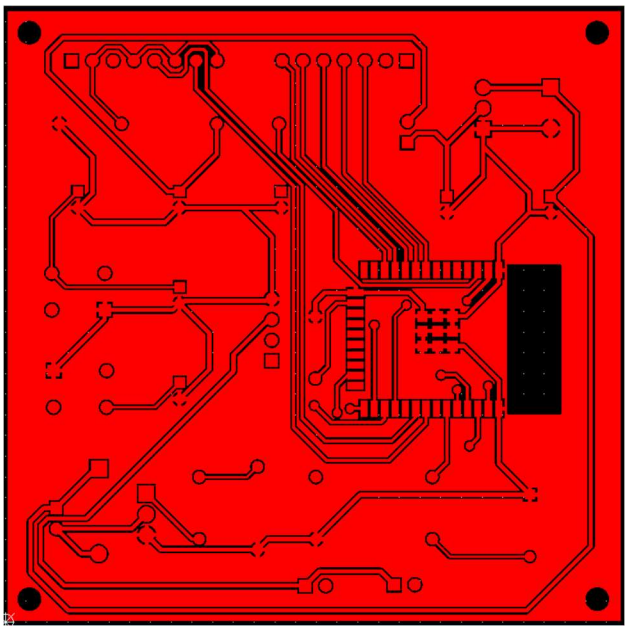


Figure: Top Layer

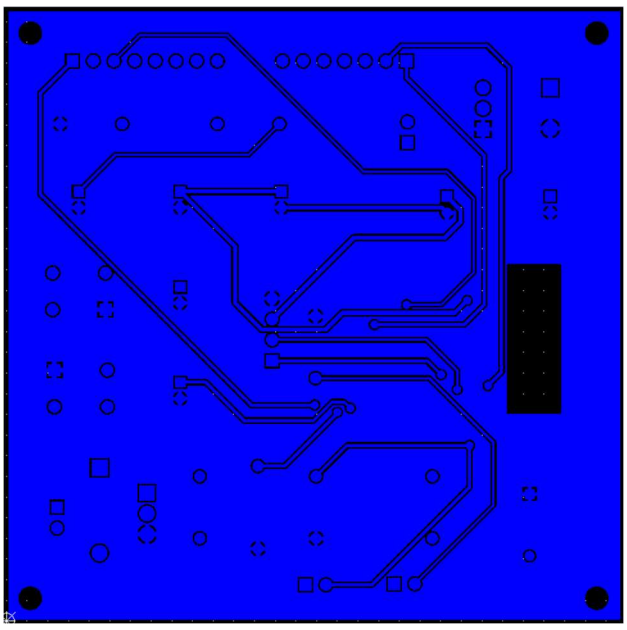


Figure: Bottom Layer

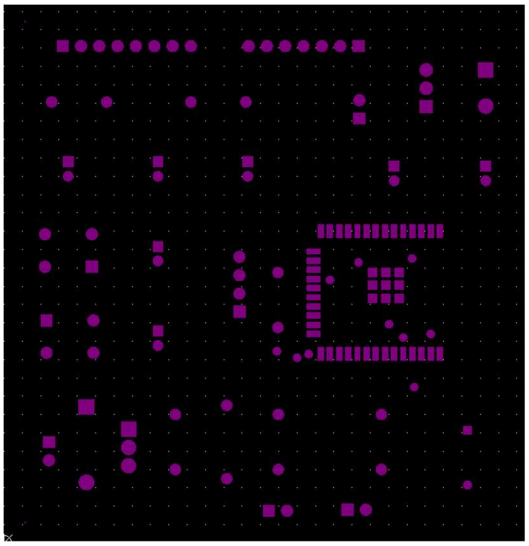


Figure: Top Soldermask

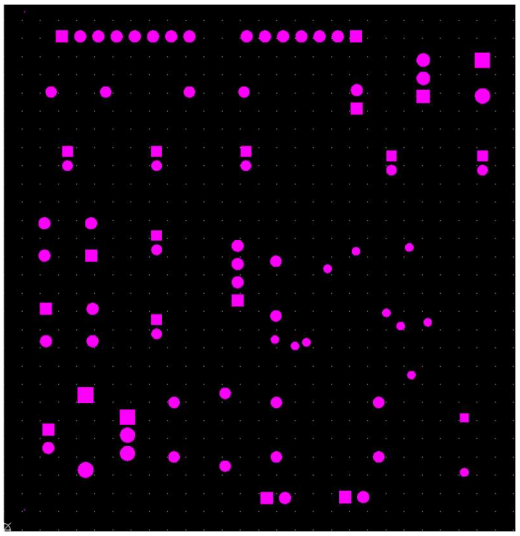


Figure: Bottom Soldermask

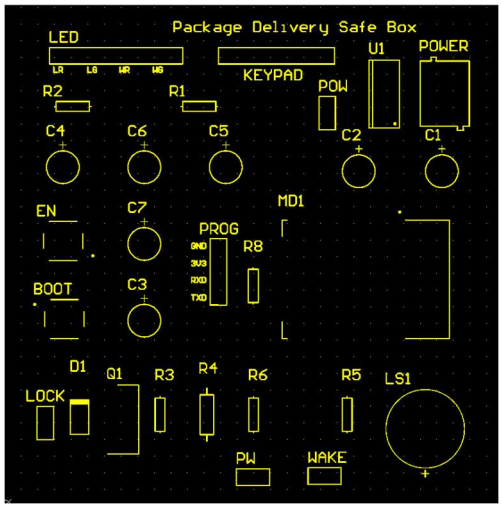


Figure: Top Overlay

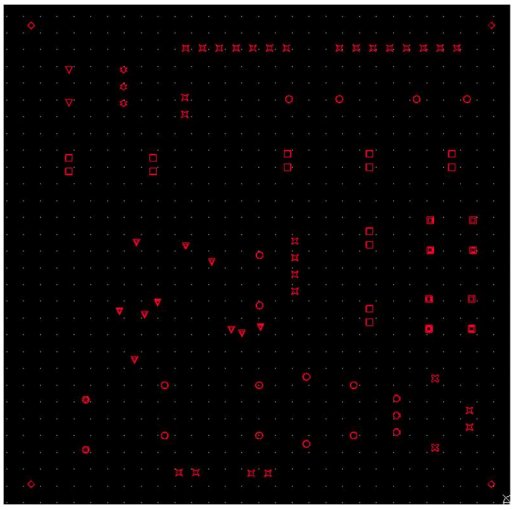


Figure: Drill Drawing Layer

NCDrill File Report For: pcb.PcbDoc 6/3/2023 3:10:40 PM

Layer Pair : Top Layer to Bottom Layer
ASCII RoundHoles File : pcb.TXT

Tool	Hole Size	Hole Tolerance	Hole Type	Hole Count	Plated	Tool Travel
T1	20mil (0.5mm)		Round	10	PTH	2.56inch (65.00mm)
T2	30mil (0.749mm)		Round	2	PTH	0.30inch (7.60mm)
T3	32mil (0.8mm)		Round	14	PTH	4.65inch (118.19mm)
T4	34mil (0.851mm)		Round	14	PTH	6.28inch (159.53mm)
T5	35mil (0.899mm)		Round	27	PTH	7.06inch (179.33mm)
T6	39mil (0.991mm)		Round	8	PTH	1.69inch (42.94mm)
T7	43mil (1.1mm)		Round	3	PTH	0.20inch (5.08mm)
T8	47mil (1.199mm)		Round	2	PTH	0.41inch (10.46mm)
T9	50mil (1.28mm)		Round	3	PTH	0.20inch (5.08mm)
T10	51mil (1.3mm)		Round	2	PTH	0.20inch (5.00mm)
T11	98mil (2.499mm)		Round	4	PTH	9.35inch (237.53mm)
Totals				89		

Total Processing Time (hh:mm:ss) : 00:00:00

Figure: Drilling Details

4.4 PCB Manufacturing

PCB Fabrication was done through the Chinese PCB manufacturer JLCPCB.

PCB Specifications

- Base material: FR-4
- Dimensions: 75.7 mm x 75.3 mm
- Surface finish : HASL(with lead)
- PCB thickness: 1.6 mm
- Copper weight: 1 oz
- PCB color: Green
- Silkscreen color: White

PCB manufacturing cost: Rs 3020.00 (For 5 PCBs and shipping)

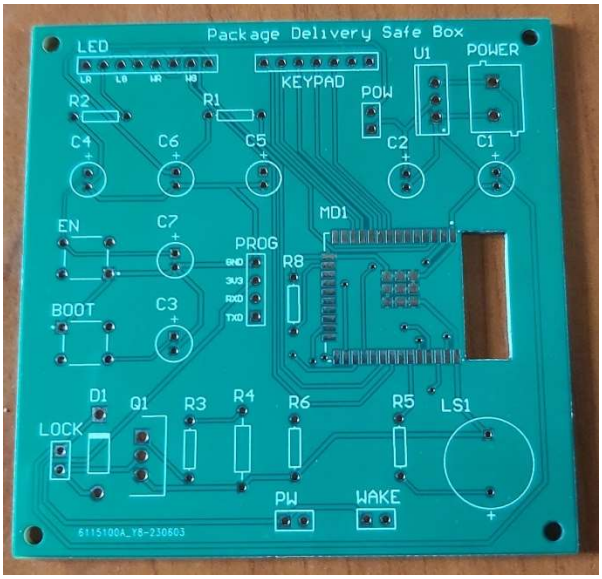
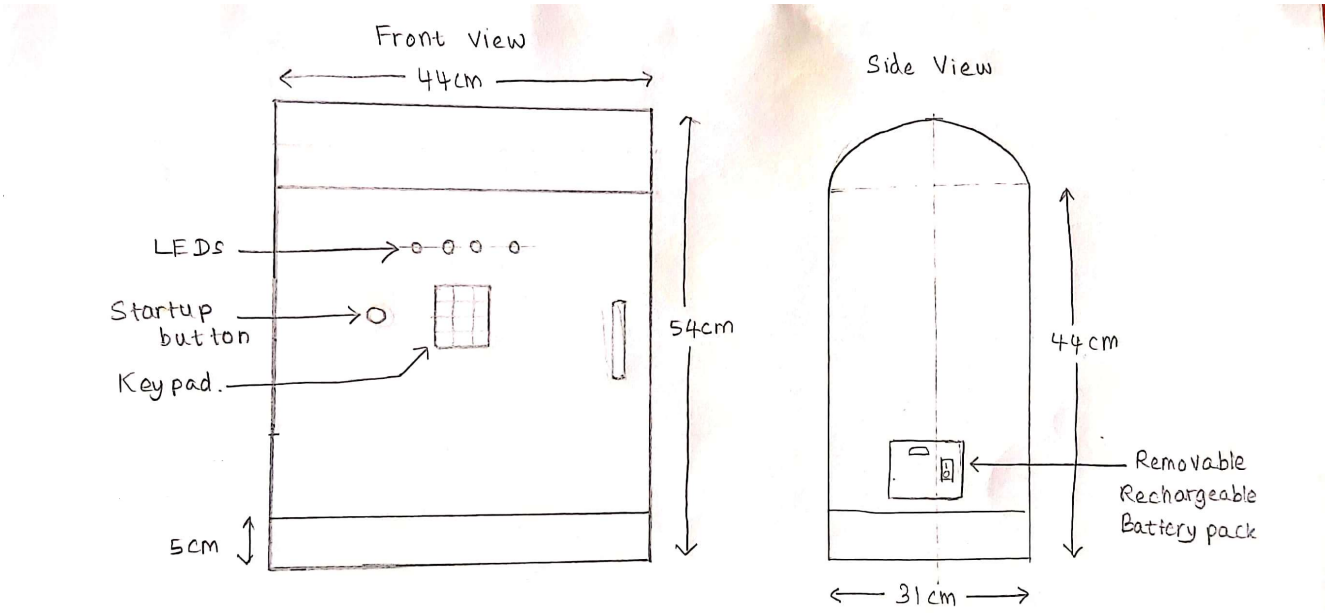
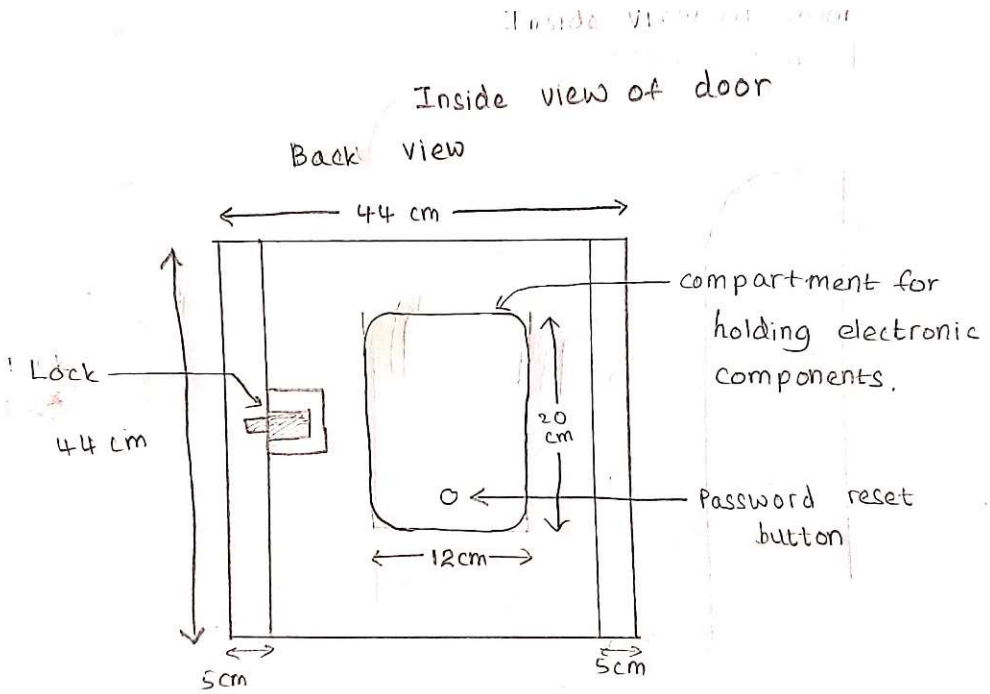


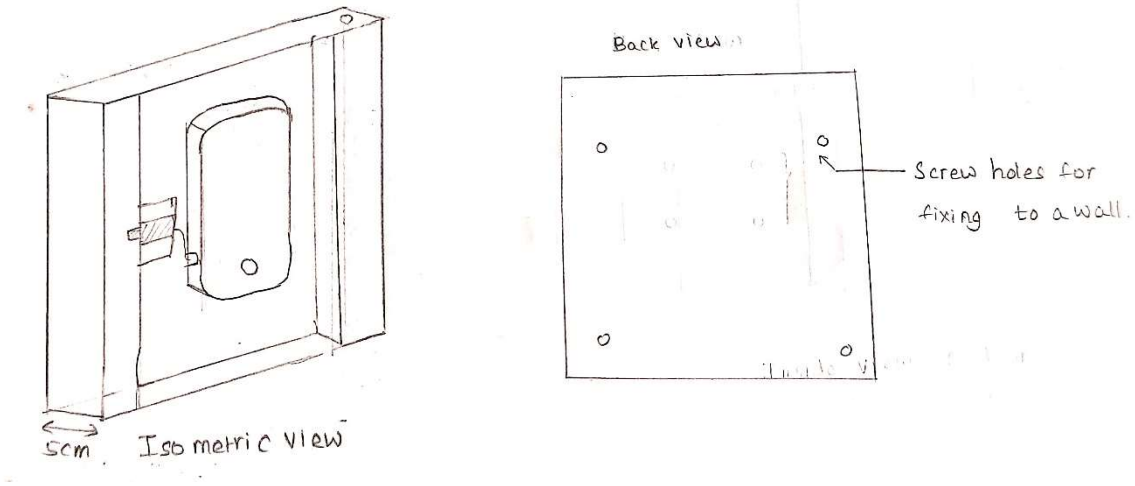
Figure: Fabricated PCB

5. Enclosure Design

5.1 Hand Sketches







5.2 User Interface

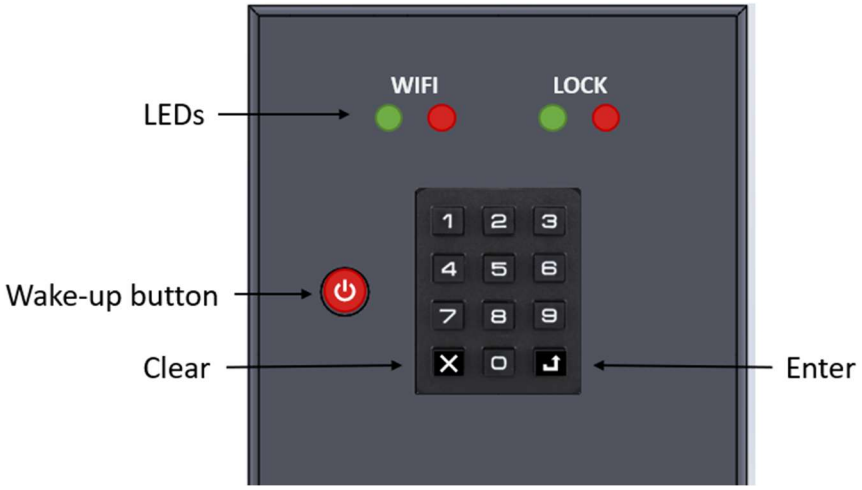


Figure: Outside view

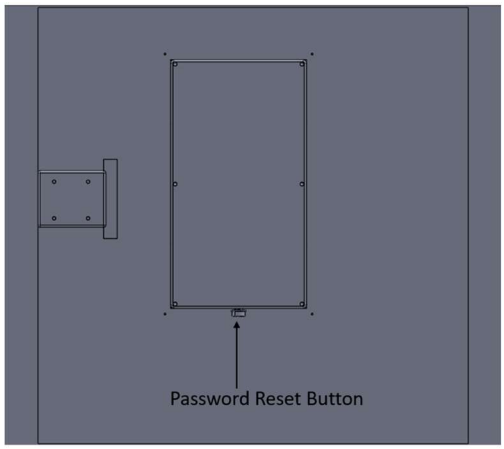
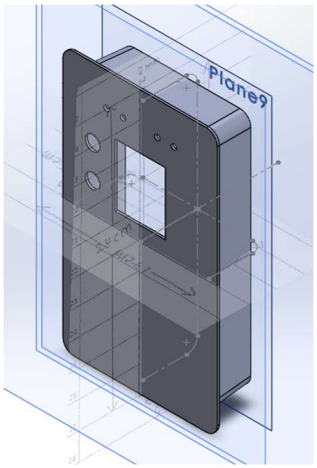


Figure: Inside view

5.3 Solidworks Design

Internal component box



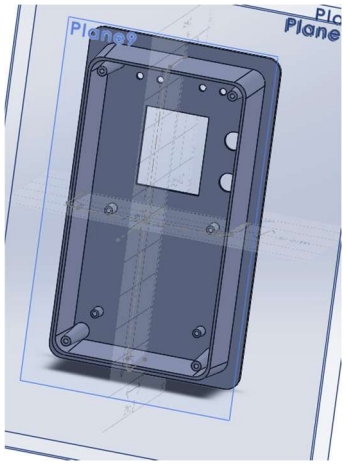


Figure: Main Body

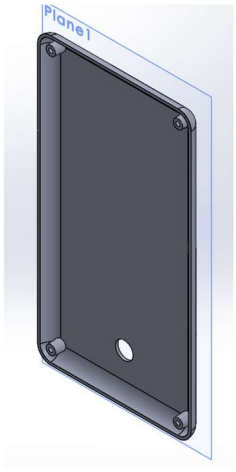
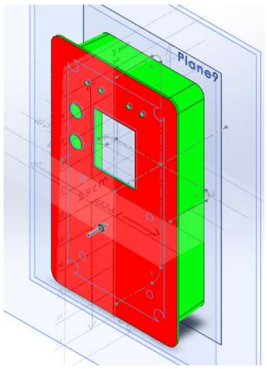


Figure: Lid



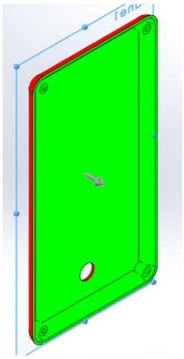
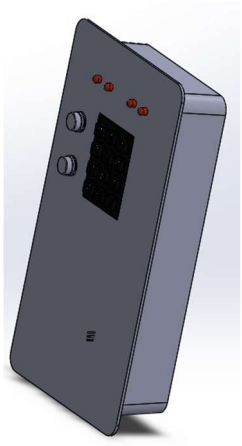


Figure: Draft Analysis



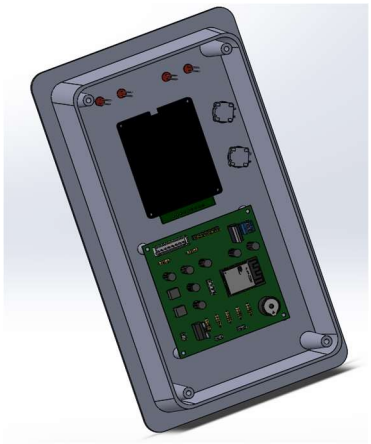


Figure: Assembly with components

Main Box

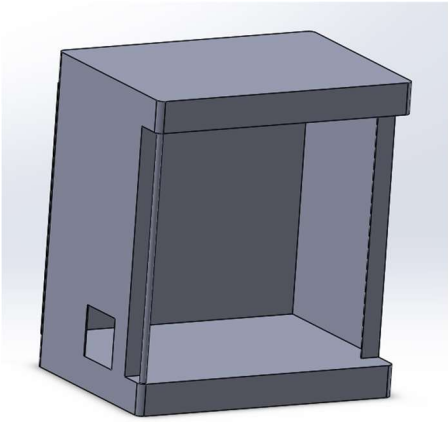


Figure: Main Body

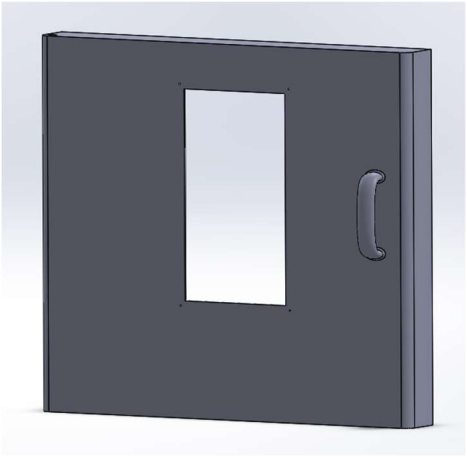


Figure: Door

Other Accessories

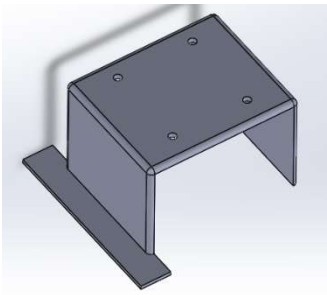


Figure: Lock Mounting Piece

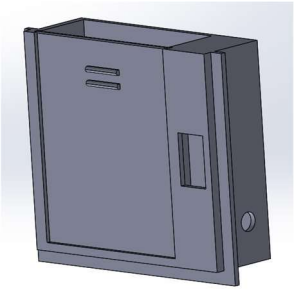
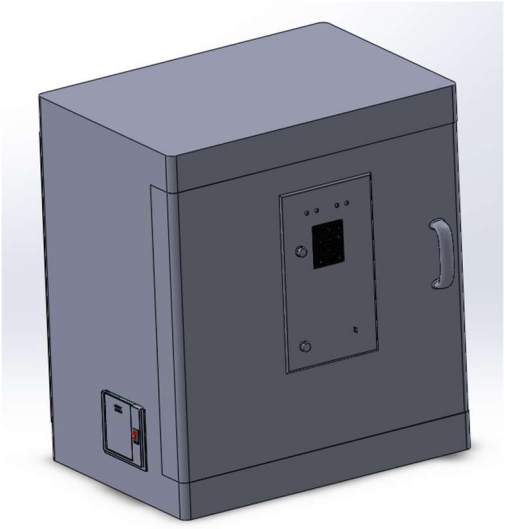
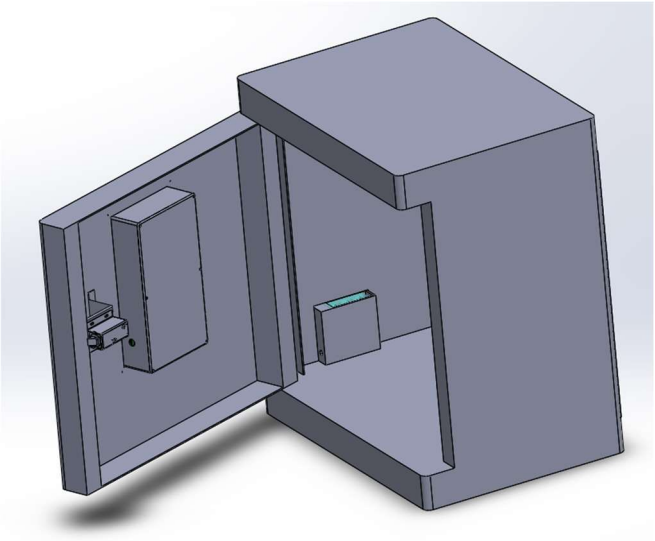


Figure: Battery Holder

Complete Assembly





5.4 Enclosure Manufacturing Methods

- The design of the box is based on an enclosure available in the market. This allows us to use this pre-made enclosure when manufacturing on the small scale in order to reduce costs.

- The internal box containing electronic components will be embedded into the door of the safe box, and the battery holder will be embedded into the side of the box. These are designed as separate parts for easier manufacturing. For small scale manufacturing, this internal box can also be manufactured by CNC machining a pre-made enclosure box and adding mounting spacers.

- For large scale manufacturing, both enclosures can be manufactured using injection molding. The design is made to meet injection molding requirements.

- For the prototype, a pre-made enclosure was used for the safe box and 3D printing was used for the remaining parts.

6. Firmware

6.1 Functional Behaviour

The firmware of the device is written using the Arduino platform due to the availability of helper libraries for peripherals, Wi-Fi and database connections. The microcontroller communicates with the mobile application through a Firebase database.

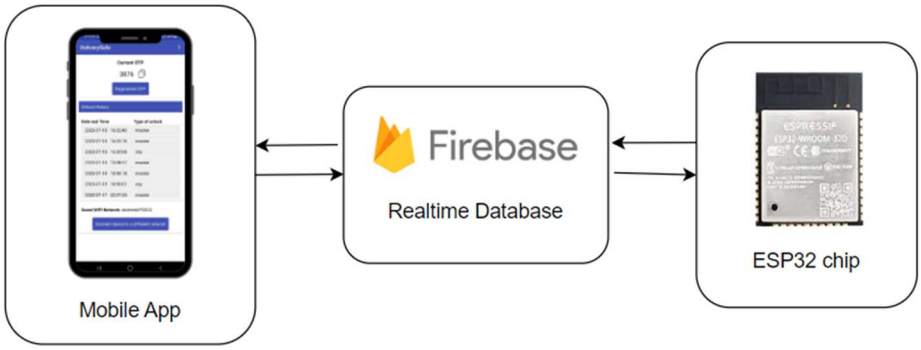


Figure: Connectivity

The program performs the following tasks:

- Connect to a Wi-Fi network
- Connect to the Firebase database using the API key
- Periodically update the OTP by checking the database
- Identify key presses, verify whether the password is correct and unlock the lock
- Send unlock information to the database (timestamp and whether the unlock was using the master password or OTP)
- Remove the OTP from the database and memory after it is used for unlocking
- Enable master password resetting when “password reset” button is pressed after entering the master password. The master password will be saved in the flash memory of the ESP32 for persistent storage.
- Change LED states to represent the state of the Wi-Fi connection and the lock

In addition, it has the following features for convenience.

Connect to a custom Wi-Fi network

When the device is not connected to Wi-Fi, pressing “0” and enter will send the device into Wi-Fi configuration mode and the ESP32 will act as a Wi-Fi access point. Then a mobile phone or any other device can connect to this network, go to the address 192.168.4.1 and enter the SSID and password of the home Wi-Fi network. Then the device will be connected to that Wi-Fi network.

Auto-reconnect
The Wi-Fi network details are stored in the flash memory of the ESP32 and can be used to auto-reconnect if the device is powered off or connection is lost.

Deep sleep mode
If the device is idle for 1 minute (no key presses or unlocking), the device will go into deep sleep mode which will reduce the current consumption to 10 µA. It can be woken up by pressing the wake-up button. It will also periodically wake up to receive any OTP updates and go back to sleep.

6.2 Microcontroller Code

- WiFi.h Header file

```
#include <WiFi.h>
#include <FirebaseESP32.h>
#include <WiFiManager.h> // https://github.com/tzapu/WiFiManager

// Provide the token generation process info.
#include <addons/TokenHelper.h>

// Provide the RTDB payload printing info and other helper functions.
#include <addons/RTDBHelper.h>

FirebaseData fbdo;
WiFiManager wm;

FirebaseAuth auth;
FirebaseConfig config;

const char* WIFI_SSID = "AndroidAP20CC";//HomeNet2";
const char* WIFI_PASSWORD = "wkbp2113";//"AjP160#1218";
const char* API_KEY = "AIzaSyAxP8GeimLKZwvs7xAu0psmbB3bMY2afY4";
const char* DATABASE_URL = "https://deliverysafe-92d96-default-rtdb.firebaseio.com/";
const char* DATABASE_SECRET = "TKWAIOOJzE2r8wvALFN8lJEcGMpsKQbwGbyKx7mh";

// NTP server to request epoch time
const char* ntpServer = "pool.ntp.org";

bool signupOK = false;
int config_portal_timeout = 120;
int connect_timeout = 30;

void setup_firebase() {
  Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);

  /* Assign the RTDB URL (required) */
  config.database_url = DATABASE_URL;
  config.signer.tokens.legacy_token = DATABASE_SECRET;
  Firebase.begin(&config, &auth);
  Firebase.setDoubleDigits(5);

  if (WiFi.status() == WL_CONNECTED) {
    Firebase.setStringAsync(fbdo, "DeliverySafe/user1/wifi", wm.getWiFiSSID());
  }
  Serial.println("Firebase started");
}

```

```
45. void setup_wifi() {
46.   WiFi.mode(WIFI_STA);
47.   wm.setRestorePersistent(true);
48.   wm.setEnableConfigPortal(false);
49.   wm.setConnectTimeout(connect_timeout);
50.   configTime(0, 0, ntpServer);
51.
52.   if (wm.autoConnect()) {
53.     setup_firebase();
54.   }
55. }
56.
57. void autoconnect_wifi() {
58.   // set configportal timeout
59.     wm.setConfigPortalTimeout(config_portal_timeout);
60.     wm.setConnectTimeout(connect_timeout);
61.     if (!wm.startConfigPortal("DeliverySafe")) {
62.       Serial.println("failed to connect and hit timeout");
63.       delay(100);
64.       return;
65.     }
66.     //if you get here you have connected to the WiFi
67.     Serial.print("Connected with IP: ");
68.     Serial.println(WiFi.localIP());
69.     Serial.println();
70.
71.     setup_firebase();
72. }
73.
74. unsigned long get_time() {
75.   time_t now;
76.   struct tm timeinfo;
77.   if (!getLocalTime(&timeinfo)) {
79.     return(0);
80.   }
81.   time(&now);
82.   return now;
83. }
84.
85. void update_unlock_info(String type, unsigned long timestamp) {
86.   Firebase.setStringAsync(fbdo, "DeliverySafe/user1/history/" + String(timestamp), type);
87.   if (type == "otp") {
88.     Firebase.setStringAsync(fbdo, "DeliverySafe/user1/pwhash/", "xxx");
89.   }
90. }
```

- Main Program

```
 1. #include <Keypad.h>
 2. #include <Preferences.h>  // for permanent storage
 3. #include <WiFi.h>
 4. #include <FirebaseESP32.h>
 5. #include "wifi.h"
 6.
 7. #define ROW_NUM 4
 8. #define COLUMN_NUM 3
 9. #define RESET '*'
10. #define ENTER '#'
11.
12. #define K1 4
13. #define K2 23
14. #define K3 32
```

```
#define K4 33
#define K5 25
#define K6 26
#define K7 27

#define NORMAL 0
#define CHANGE 1

#define RESET_PIN 22
#define RED_LED_PIN 16
#define GREEN_LED_PIN 17
#define BUZZER_PIN 21
#define LOCK_PIN 15
#define WIFI_RED_LED 18//LED_BUILTIN  //18
#define WIFI_GREEN_LED 19

#define KEY_PRESS 440
#define CORRECT 880
#define WRONG 220

const unsigned long idle_time = 60 * 1000;
const unsigned long sleep_time = 30;

RTC_DATA_ATTR unsigned long checkDataPrevMillis = 0;
RTC_DATA_ATTR unsigned long last_action_time = 0;
RTC_DATA_ATTR unsigned long offset_time = 0;
RTC_DATA_ATTR unsigned long last_unlock_time = 0;

RTC_DATA_ATTR String last_unlock_type = "";
RTC_DATA_ATTR bool is_last_unlock_master = false;

char keys[ROW_NUM][COLUMN_NUM] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};

byte pin_row[ROW_NUM] = {K2, K7, K6, K4}; //connect to the row pinouts of the keypad
byte pin_column[COLUMN_NUM] = {K3, K1, K5}; //connect to the column pinouts of the keypad

Keypad keypad = Keypad(makeKeymap(keys), pin_row, pin_column, ROW_NUM, COLUMN_NUM);
Preferences preferences;

RTC_DATA_ATTR String entered = "";
RTC_DATA_ATTR String master_password;
RTC_DATA_ATTR String otp_password = "xxx";

RTC_DATA_ATTR int mode = NORMAL;

unsigned long wifi_blink_millis = 0;
int wifi_blink_status = HIGH;
unsigned long lock_blink_millis = 0;
int lock_blink_status = HIGH;

void light_LED(int state) {
  digitalWrite(RED_LED_PIN, !state);
  digitalWrite(GREEN_LED_PIN, state);
}

void light_WIFI_LED(int state) {
  digitalWrite(WIFI_RED_LED, !state);
  digitalWrite(WIFI_GREEN_LED, state);
}

```

```
void lock_state_outputs(int state) {
  digitalWrite(LOCK_PIN, state);  // to confirm that it is not floating
  light_LED(state);
}

void update_otp() {
  if (Firebase.getString(fbdo, "/DeliverySafe/user1/pwhash")) {
    otp_password = fbdo.to<String>();
    otp_password = otp_password.substring(2, otp_password.length()-2);
    Serial.println("Set OTP password: " + otp_password);
  } else {
    Serial.println("Password not updated");
  }
}

void enter_action() {
  if (entered == "0") { // wifi config mode
    if (WiFi.status() != WL_CONNECTED) {
      light_WIFI_LED(LOW);
      Serial.println("Entering WiFi config mode");
      Serial.println("Go to 192.168.4.1");
      autoconnect_wifi();
    }

  } else if (mode == NORMAL) {
    if (entered != master_password && WiFi.status() == WL_CONNECTED) update_otp();
    if (entered == master_password || entered == otp_password) {
      Serial.println("Correct password");
      lock_state_outputs(HIGH);

      tone(BUZZER_PIN, CORRECT);
      delay(400);
      noTone(BUZZER_PIN);

      delay(2000);
      lock_state_outputs(LOW);

      last_unlock_time = get_time();

      if (entered == otp_password) {
        last_unlock_type = "otp";
        otp_password = "xxx";
        is_last_unlock_master = false;
      } else {
        last_unlock_type = "master";
        is_last_unlock_master = true;
      }

    } else {
      Serial.println("Wrong password");
      tone(BUZZER_PIN, WRONG);
      delay(400);
      noTone(BUZZER_PIN);
    }
  } else if (mode == CHANGE) {
    master_password = entered;
    preferences.begin("password", false);
    preferences.putString("pass",master_password);
    preferences.end();

    Serial.println("Changed master password to: " + master_password);
    mode = NORMAL;
    light_LED(LOW);
  }
}
```

```
145.
146. void key_action(char key) {
147.   if (key == RESET) {
148.     entered = "";
149.   } else if (key == ENTER) {
150.     enter_action();
151.     entered = "";
152.   } else {
153.     entered = entered + String(key);
154.   }
155.   Serial.println(entered);
156. }
157.
158. void setup() {
159.   // put your setup code here, to run once:
160.   pinMode(RESET_PIN, INPUT_PULLDOWN);
161.   pinMode(RED_LED_PIN, OUTPUT);
162.   pinMode(GREEN_LED_PIN, OUTPUT);
163.   pinMode(WIFI_RED_LED, OUTPUT);
164.   pinMode(WIFI_GREEN_LED, OUTPUT);
165.   pinMode(BUZZER_PIN, OUTPUT);
166.   pinMode(LOCK_PIN, OUTPUT);
167.   esp_sleep_enable_ext0_wakeup(GPIO_NUM_13,1);
168.   esp_sleep_enable_timer_wakeup(sleep_time * 1000000);
169.
170.   Serial.begin(115200);
171.   lock_state_outputs(LOW);
172.   light_WIFI_LED(LOW);
173.
174.   if (esp_sleep_get_wakeup_cause() == ESP_SLEEP_WAKEUP_UNDEFINED) {
175.     offset_time = millis();
176.     last_action_time = millis();
177.   } else if (esp_sleep_get_wakeup_cause() == ESP_SLEEP_WAKEUP_EXT0) {
178.     last_action_time = offset_time + millis();
179.   }
180.
181.   preferences.begin("password", true);
182.   master_password = preferences.getString("pass", "!!");
183.   preferences.end();
184.
185.   if (master_password == "!!") {  // No password set
186.     master_password = "1234";
187.     preferences.begin("password", false);
188.     preferences.putString("pass",master_password);
189.     preferences.end();
190.   }
191.
192.   Serial.println("Password is: "+master_password);
193.
194.   setup_wifi();
195.
196. }
197.
198. void loop() {
199.   // put your main code here, to run repeatedly:
200.   char key = keypad.getKey();
201.   if (key) {
202.     tone(BUZZER_PIN, KEY_PRESS);
203.     delay(200);
204.     noTone(BUZZER_PIN);
205.     key_action(key);
206.   }
207.
208.   if (key || digitalRead(RESET_PIN)) {
209.     last_action_time = offset_time + millis();
```

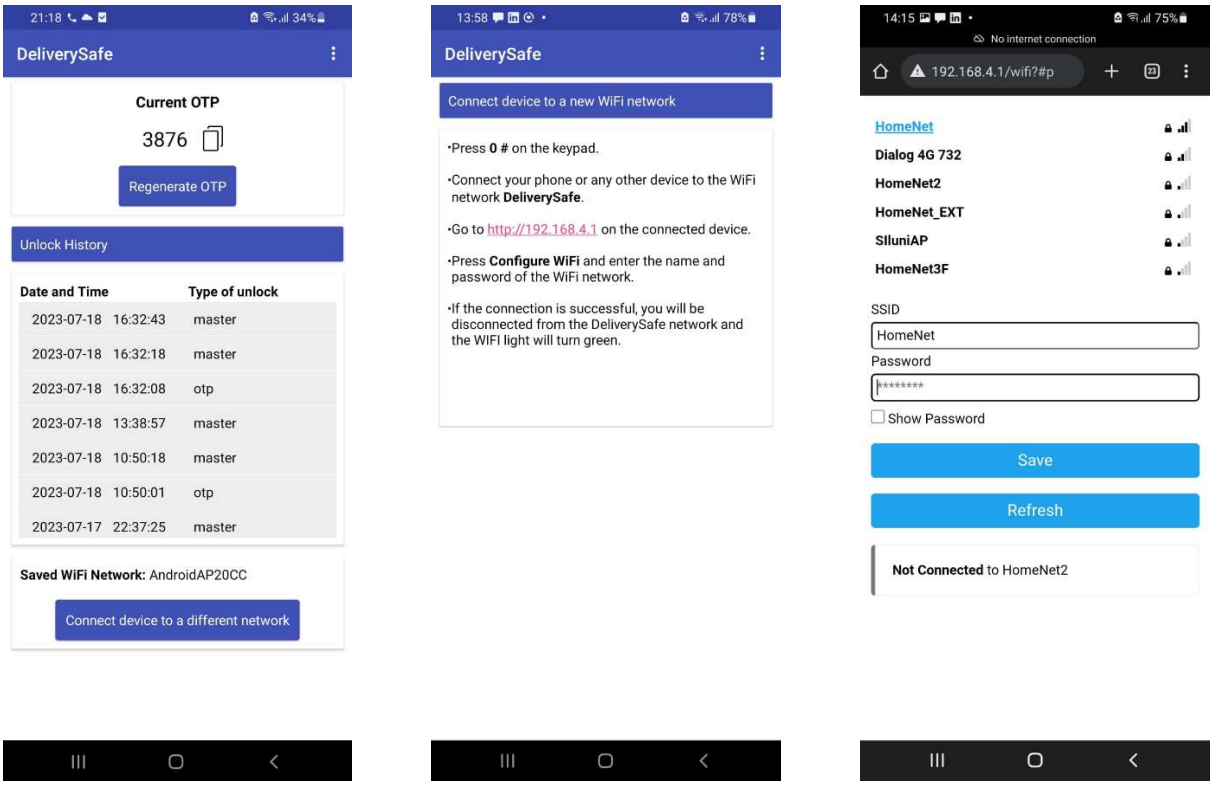
```
  }

  if (digitalRead(RESET_PIN) && is_last_unlock_master) {
    Serial.println("In password changing mode");
    mode = CHANGE;
    light_LED(HIGH);
    tone(BUZZER_PIN, CORRECT);
    delay(200);
    noTone(BUZZER_PIN);
  }

  if ((millis() - checkDataPrevMillis > 15000 || checkDataPrevMillis == 0) && WiFi.status() == WL_CONNECTED) {
    checkDataPrevMillis = millis();
    Serial.println("Checking for updates...");
    update_otp();
  }

  if (last_unlock_type != "" && WiFi.status() == WL_CONNECTED) {
    update_unlock_info(last_unlock_type, last_unlock_time);
    last_unlock_type = "";
  }

  if (WiFi.status() == WL_CONNECTED) {
    light_WIFI_LED(HIGH);
  } else if (millis() - wifi_blink_millis > 200) {
    digitalWrite(WIFI_GREEN_LED, LOW);
    wifi_blink_status = !wifi_blink_status;
    digitalWrite(WIFI_RED_LED, wifi_blink_status);
    wifi_blink_millis = millis();
  }

  if (mode == CHANGE && (millis() - lock_blink_millis > 200)) {
    lock_blink_status = !lock_blink_status;
    digitalWrite(GREEN_LED_PIN, lock_blink_status);
    lock_blink_millis = millis();
  }

  if (offset_time + millis() - last_action_time > idle_time) {
    Serial.println(offset_time + millis() - last_action_time);
    offset_time = offset_time + millis();
    Serial.println("Going to sleep now");
    esp_deep_sleep_start();
  }
}

```

7. Mobile Application and Database

7.1 Mobile App

The mobile app performs the following functionalities:

- Set a random OTP at the user's request and save it to the Firebase database
- Read unlock data from the database and show the history of recent unlocks
- Provide instructions for connecting the device to Wi-Fi, and display the currently connected Wi-Fi network of the device
- Send a notification when the box is unlocked using the OTP



Figures: Mobile app interface and Wi-Fi settings interface

7.2 Database

A Firebase Realtime Database was used as the cloud database for this system as it is capable of synchronizing data across all client devices in real time. The following data is stored in the database for a single user:

- The hash of the one-time password (the plaintext passcode is not stored due to security concerns)
- Unlocked timestamp and unlock type (OTP or master password)
- Currently connected Wi-Fi network

8. Assembly

8.1 Soldering process

The PCB was soldered using hand soldering. First the ESP32-WROOM-32D chip was hand soldered using flux and then cleaned using isopropyl alcohol. Then the through-hole components were soldered in the order of component height for ease of soldering. Finally, the connector wires were soldered to the respective peripherals and insulated with heat shrink tubing.

JST connecters were used to connect the peripherals as they provide a stable connection, as well as being removable for repairs and impossible to insert the wrong way.

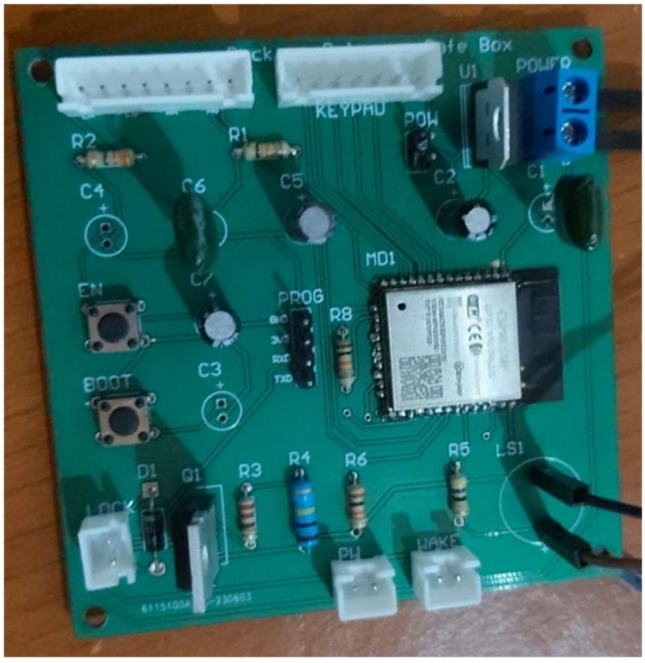


Figure: Soldered PCB

8.2 Program Uploading Process

The firmware is uploaded to the microcontroller using the header pins marked as PROG, and the EN and BOOT buttons of the PCB. Programming can be done using an FTDI or USBasp programmer. The TX, RX, GND and 3V3 pins should be connected to the corresponding pins of the programmer. Then the EN button must be pressed and released while the BOOT button is being pressed in order to enter the boot mode of the ESP32. Afterwards the program can be uploaded using the Arduino IDE or a suitable other software.

8.3 Component and Enclosure Assembly

- The PCB and the lock were mounted using M3 screws.
- The peripherals such as LEDs, keypad, buzzer and switches were secured in place using some laser-cut spacers.
- The lock mounter, component box and battery holder were fixed to the enclosure using super glue. These parts are permanent and not meant to be removable.
- The connectors from the lock and the battery holder are connected to the PCB and the component box is closed using M3 screws.

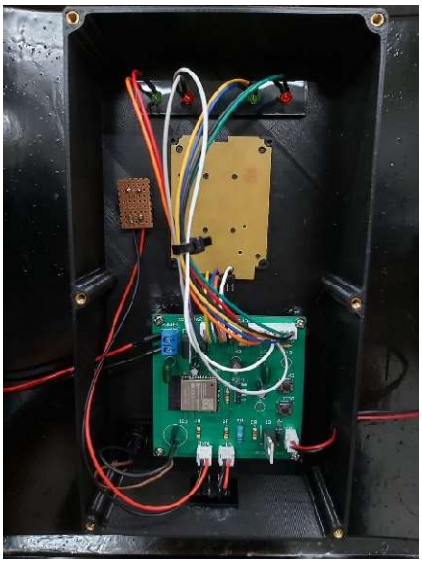
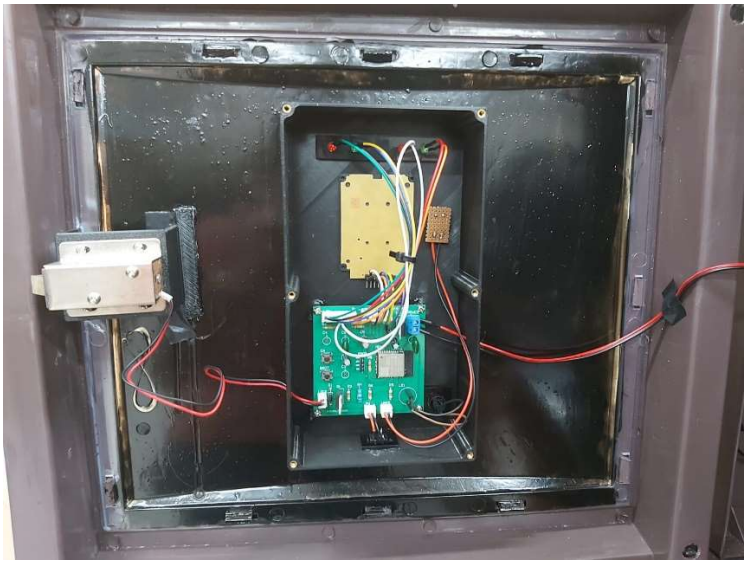



Figure: Inside of device after assembly

Figure: Final Product

9. Testing

Thorough testing of the product must be conducted in order to ensure proper functionality under all possible conditions. The following steps were taken in the circuit design stage to aid with testing:

- A jumper clip was added between the power circuit output and the microcontroller power input, allowing us to test the power circuit before giving power to the microcontroller. This will prevent any damage to the microcontroller from faults in the power circuit.
- Header pins for TX and RX pins of the microcontroller as well as EN and BOOT buttons were added to allow updating the firmware of the chip as many times as needed even after fully assembling the device. This will also allow us to view the serial output while the program is running, which is helpful for debugging.

Then, the following testing steps were conducted before and during assembly.

- Check the PCB for connectivity and short circuit faults
- Test each through-hole component in a suitable way before soldering to the PCB
- Check all soldered connections for proper continuity
- Check the output voltage of the power circuit with the jumper clip removed
- Upload a basic program to the microcontroller to ensure its functionality
- Upload test programs to test the functionality of all functional blocks of the circuit, and the Wi-Fi connectivity
- Upload the actual firmware (which is tested beforehand using a development board) and test the functionality for different user actions
- Test the integration with the mobile application

In addition to circuit testing, it was tested that the lock will always close properly and will not come apart under physical force.

10. Cost and Pricing

10.1 Bill of Materials

Component	Quantity	Manufacturer	Store	Unit Price (Rs.)
ESP32-WROOM-32D	1	Espressif Systems	Mouser	1356
Diode 1N4007	1	Diotec Semiconductor	Mouser	66
IRL520NPBF MOSFET	1	Infineon Technologies	Mouser	335
LD1117V33C 3.3V regulator	1	STMicroelectronics	Mouser	295
Keypad	1	Adafruit	Mouser	1590
12V Solenoid Lock	1	Adafruit	Mouser	2320
5V Passive Buzzer	1	CUI Devices	Mouser	150
2-pin Terminal Block	1		Local Store	20
5mm LEDs	4		Local Store	4
Small push buttons	2		Local Store	10
Large tactile push buttons	3		Local Store	40
10uF capacitor	1	Samwha Capacitor	LCSC	20
0.1uF capacitor	2	Samwha Capacitor	LCSC	33
22uF capacitor	1	Samwha Capacitor	LCSC	20
1uF capacitor	1	Samwha Capacitor	LCSC	33
0.33uF capacitor	1	Samwha Capacitor	LCSC	33
330ohm resistor	1	Resistor.Today	LCSC	11
100k ohm resistor	1	Resistor.Today	LCSC	11
220 ohm resistor	1	Resistor.Today	LCSC	11
47ohm resistor	1	Resistor.Today	LCSC	11
10k ohm resistor	3	Resistor.Today	LCSC	11
Male header pins	6		Local Store	6
2-pin JST connector	4		Local Store	45
7-pin JST connector	1		Local Store	45
8-pin JST connector	1		Local Store	45
Power toggle switch	1		Local Store	20
18650 3.7V Rechargeable Battery	3		Local Store	600
Jumper clip	1		Local Store	10

10.2 Rough Cost of Prototyping

Item	Cost (Rs.)
Electronic components (including shipping cost)	10000
PCB (5 PCBs including shipping cost)	3000
3D printed enclosure parts	12000
Pre-made enclosure box	3500
Total	**28500**

10.3 Proposed Product Price

When manufacturing in bulk at the small business level, the enclosure cost can be reduced significantly by avoiding 3D printing and using pre-made enclosures with laser cutting or CNC machining.

The effect of shipping cost on one unit will be reduced when components and PCBs are ordered in bulk. The unit price may also be reduced.

There will be no cloud database cost at the initial stages as Firebase databases are free for up to 50000 monthly users.

Manufacturing cost considering the above factors

Item	Cost (Rs.)
Electronic components	7500
PCB	350
Enclosure parts	6000
Total	**13850**

Keeping a profit margin of Rs. 2000, the product can be sold at a price of around Rs. 16000. This is significantly cheaper than similar products currently available in the market.

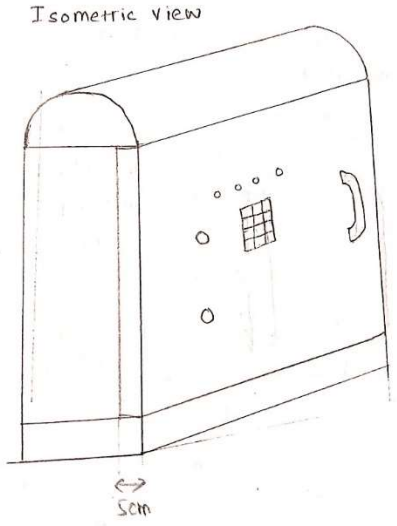
11. Further Improvements

Another design iteration was conducted for this product to identify improvements, flaws and get user feedback. A conceptual design and preliminary design stage was used to develop this.

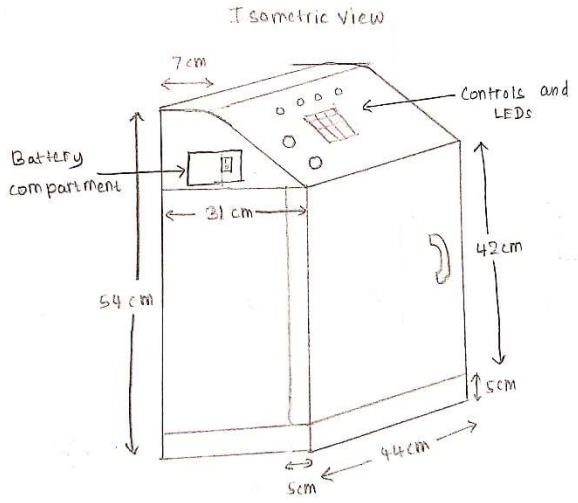
11.1 Conceptual Design Stage

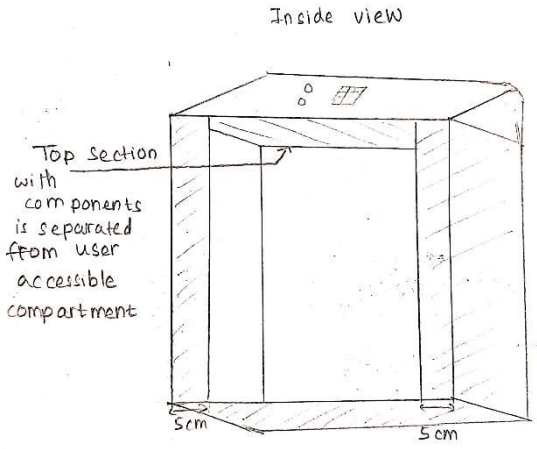
Alternate circuit and enclosure designs were proposed during this stage.

Enclosure Design 1

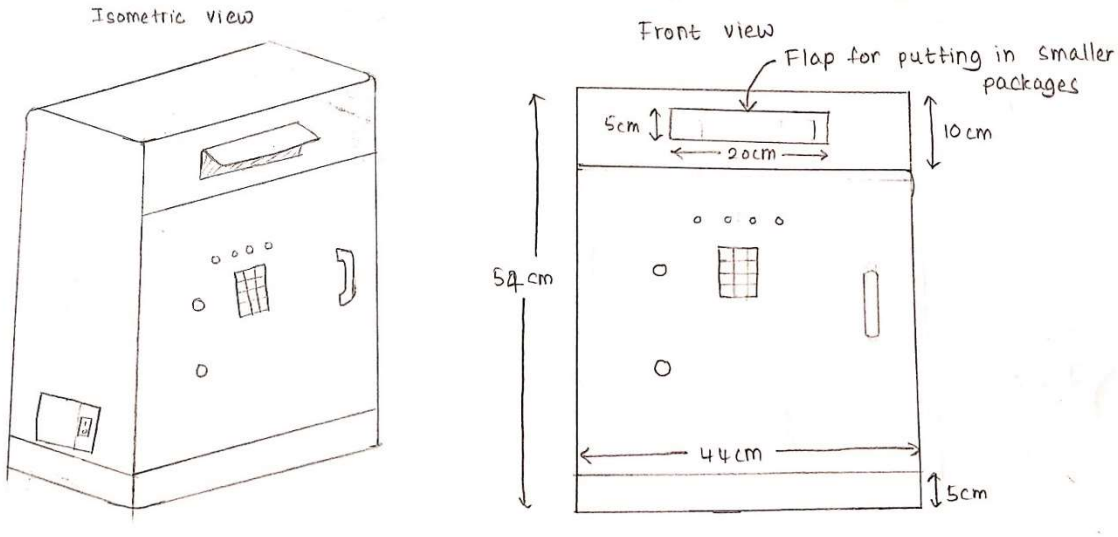


Enclosure Design 2

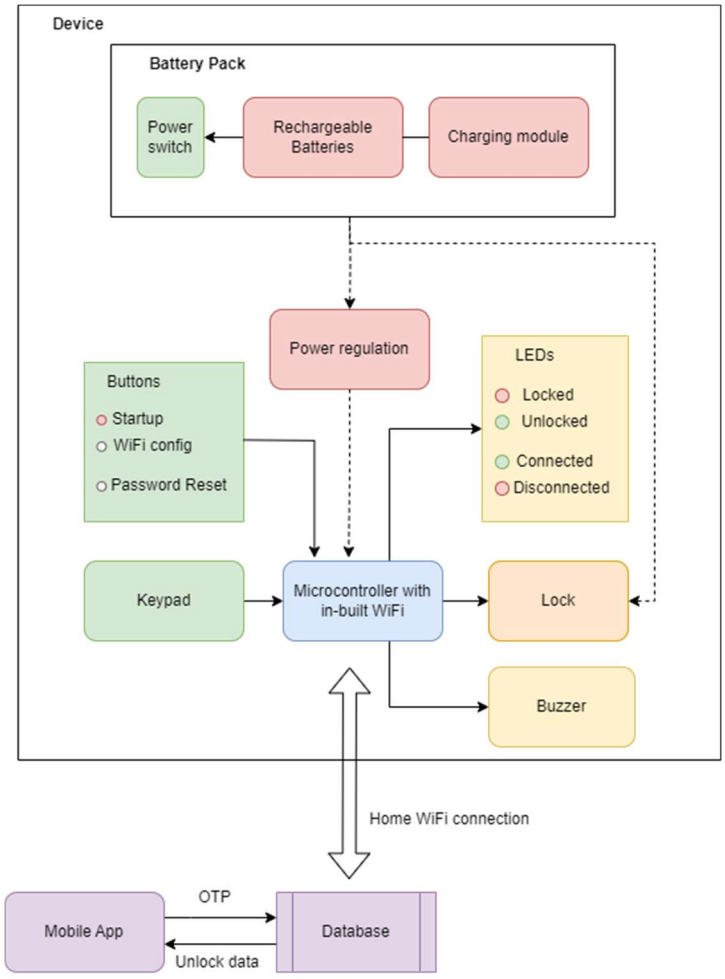




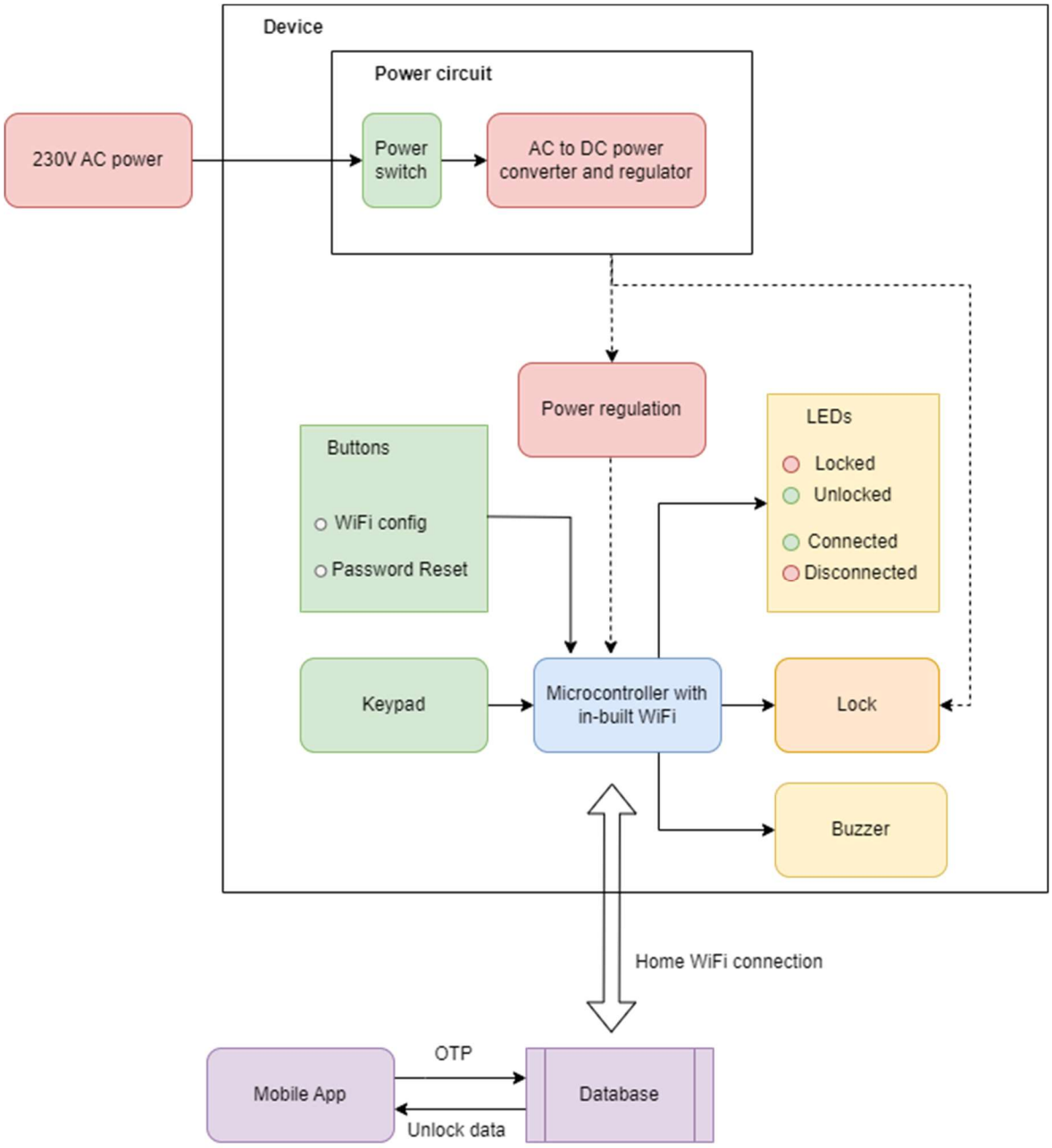
Enclosure Design 3



Block Diagram Design 1



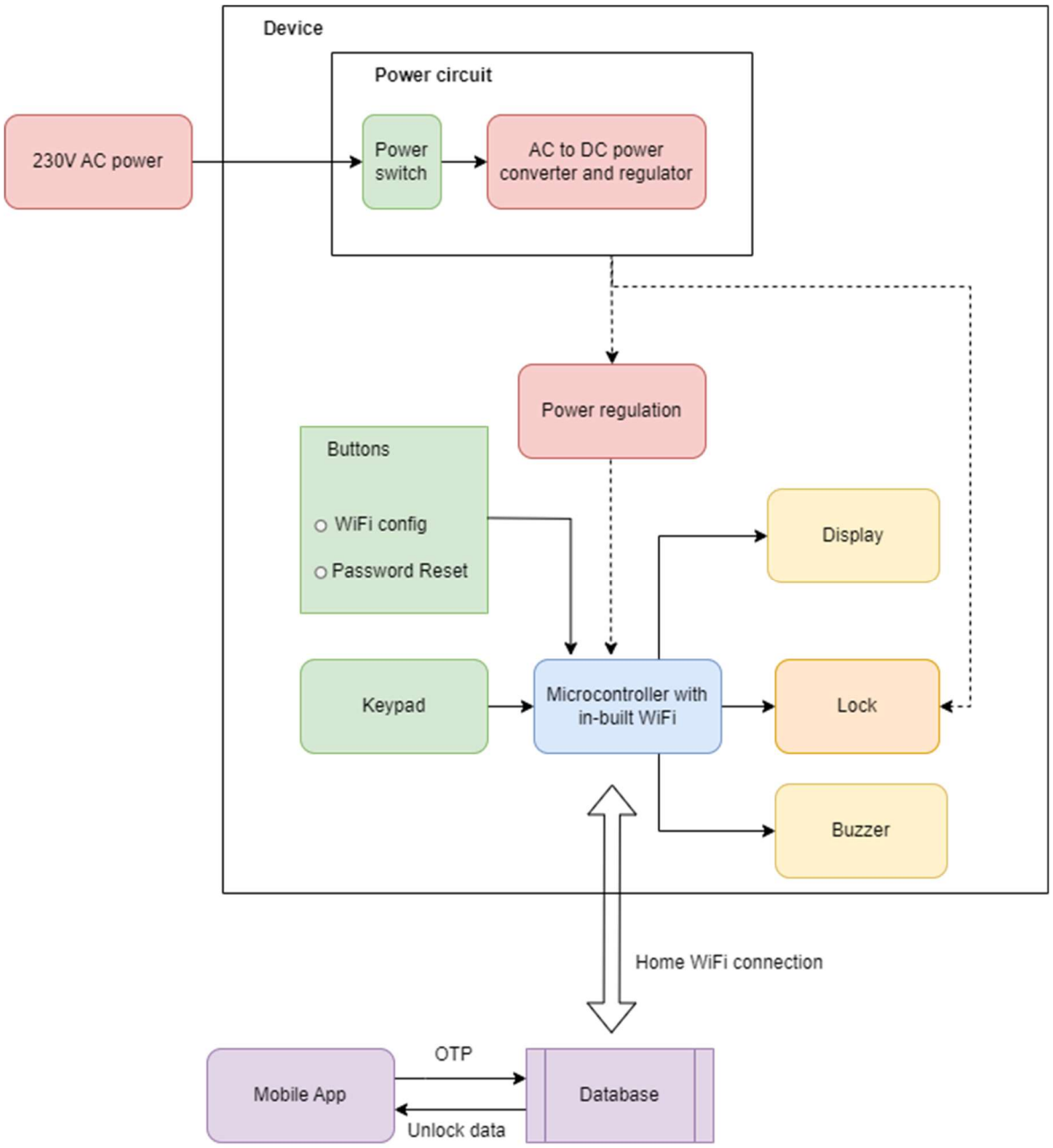
Block Diagram Design 2



Differences from original design

- Provide 230V AC power instead of using batteries
- Remove the startup button as it is only required to save power when batteries are used

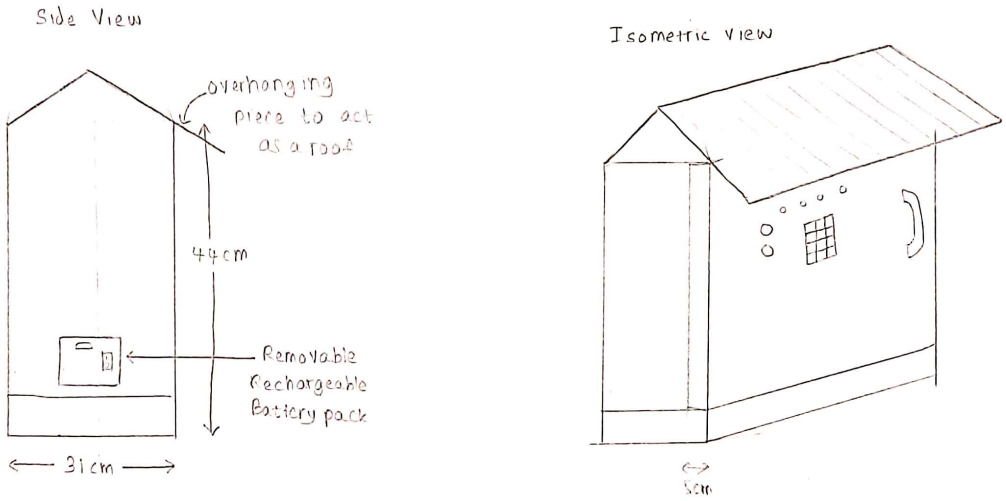
Block Diagram Design 3



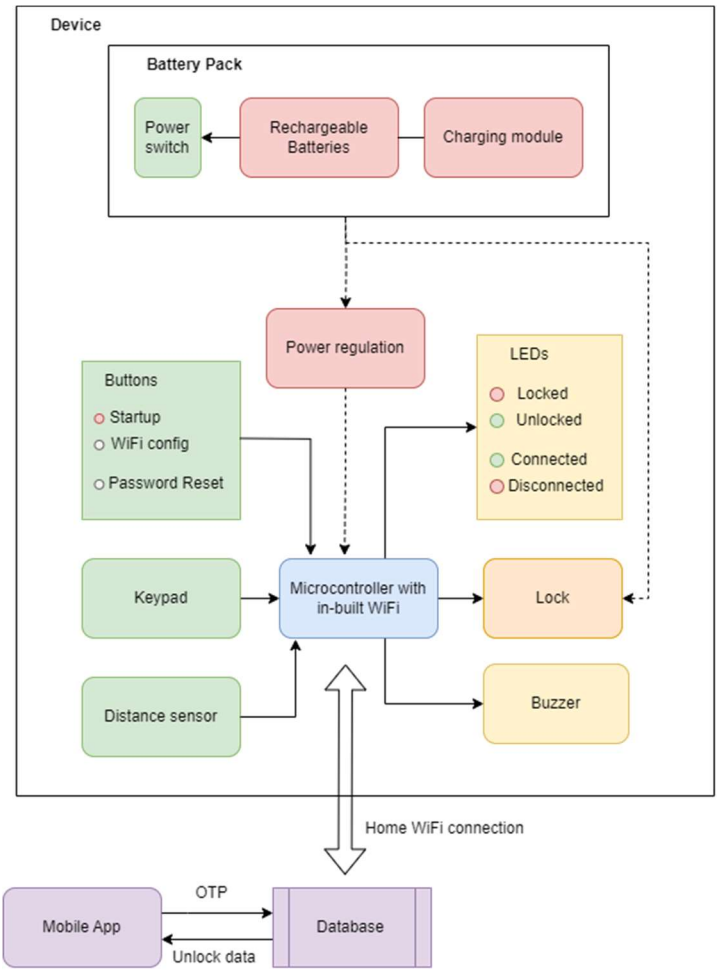
Differences from original design

- A small display instead of LEDs to show information
- 230V AC power instead of batteries as using the display will increase the power consumption

Enclosure Design with User Feedback



Block Diagram with User Feedback



Difference from original design: Adding a sensor to identify when the door is opened, to detect whether it was forcibly opened without using the password

Evaluation of enclosure designs

Each of the above designs will be evaluated based on the following criteria by giving marks between 1-5. (5 – most favourable, 1 – least favourable)

Since the outer dimensions of all 4 designs can be adjusted based on the requirements, the size of the enclosure is kept constant in order to accurately compare other factors.

Criteria	Marks (1-5)			
	Design 1	**Design 2**	**Design 3**	**Design with user feedback**
Security	4	4	1	4
Capacity for placing packages	4	3	5	4
Attractiveness	5	5	3	3
User-friendliness	4	2	5	4
Ease of repair	5	2	5	5
Ease of wiring between lock, PCB and battery	4	3	4	4
Lifespan of the device	4	4	4	5
Total	30	23	27	29

Detailed remarks about the evaluation

- In Design 3, the opening at the top can allow a thief to reach through it with some long object and manipulate the lock. If the top part was separated from the rest of the box to avoid this problem, it would be easy for the thief to retrieve packages put into the top part.
- In Design 2, having the controls at the top makes it difficult to use if the box has to be mounted at a higher level.
- In Design 3, having a separate opening for small packages removes the need to use an OTP for those packages.

Evaluation of block diagrams

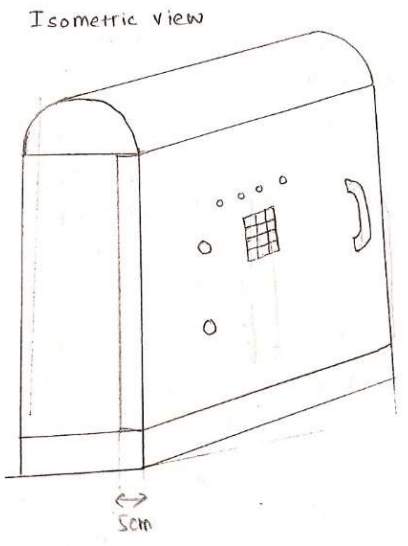
Each of the above designs will be evaluated based on the following criteria by giving marks between 1-5. (5 – most favourable, 1 – least favourable)

Criteria	Marks (1-5)			
	Design 1	**Design 2**	**Design 3**	**Design with user feedback**
Security	4	4	4	5
Ease of providing power	5	3	3	5
Reliability	4	3	3	5
Ease of use for the owner	3	4	5	3
Ease of use for the delivery person	4	4	5	4
Power consumption	5	5	3	4
Design simplicity	4	5	4	4
Cost effectiveness	3	4	3	3
Total	32	32	30	33

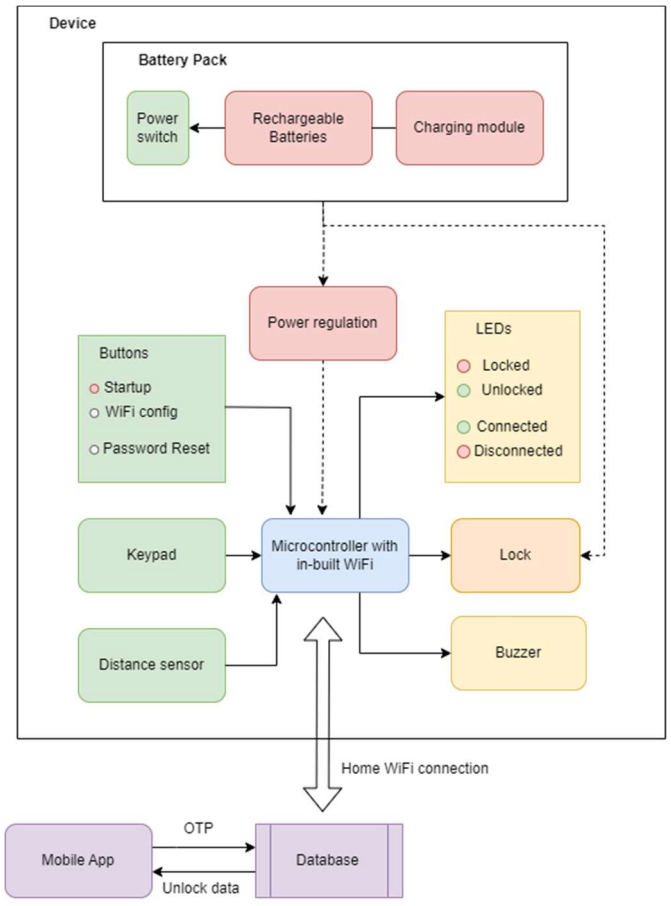
Detailed remarks about the evaluation

- In the design with user feedback, the added distance sensor provides a theft detection feature which makes the system more secure and reliable.
- In Design 2 and 3, providing main AC power to the device might be difficult because the device has to be located outside the house.
- In Design 1, maintenance of the device is comparatively difficult as the battery pack has to be removed and recharged from time to time.

Selected enclosure design – Design 1



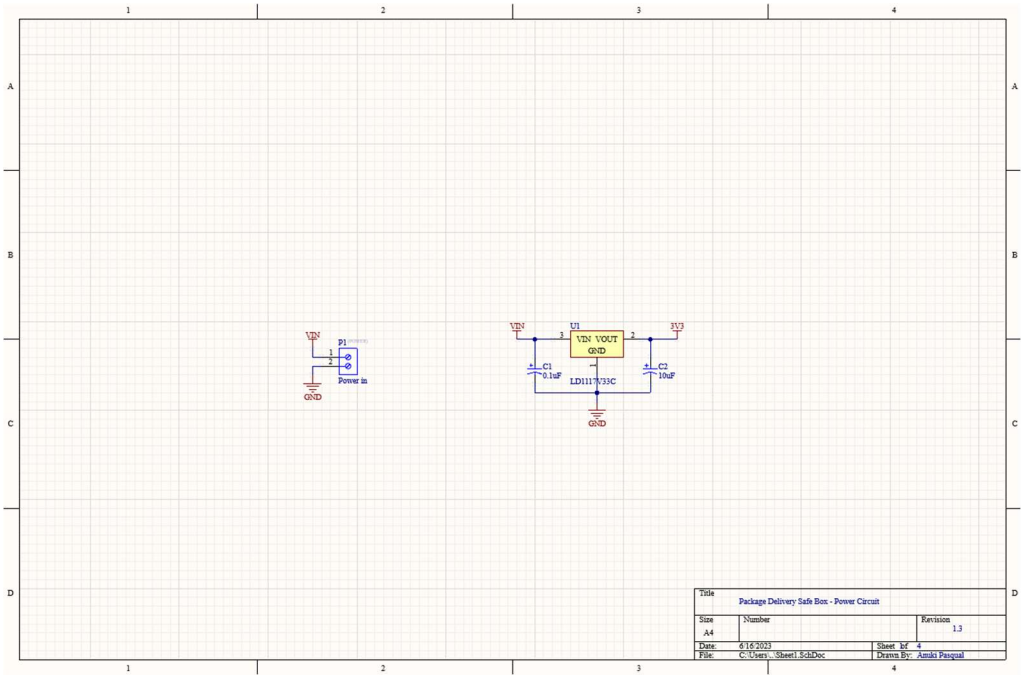
Selected block diagram – Design with user feedback

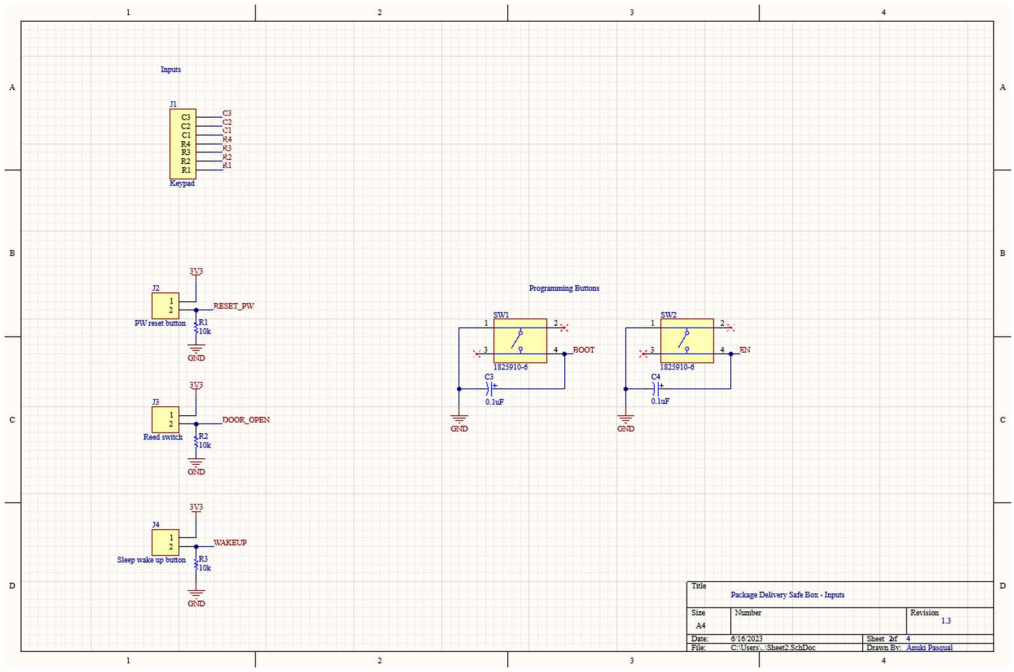


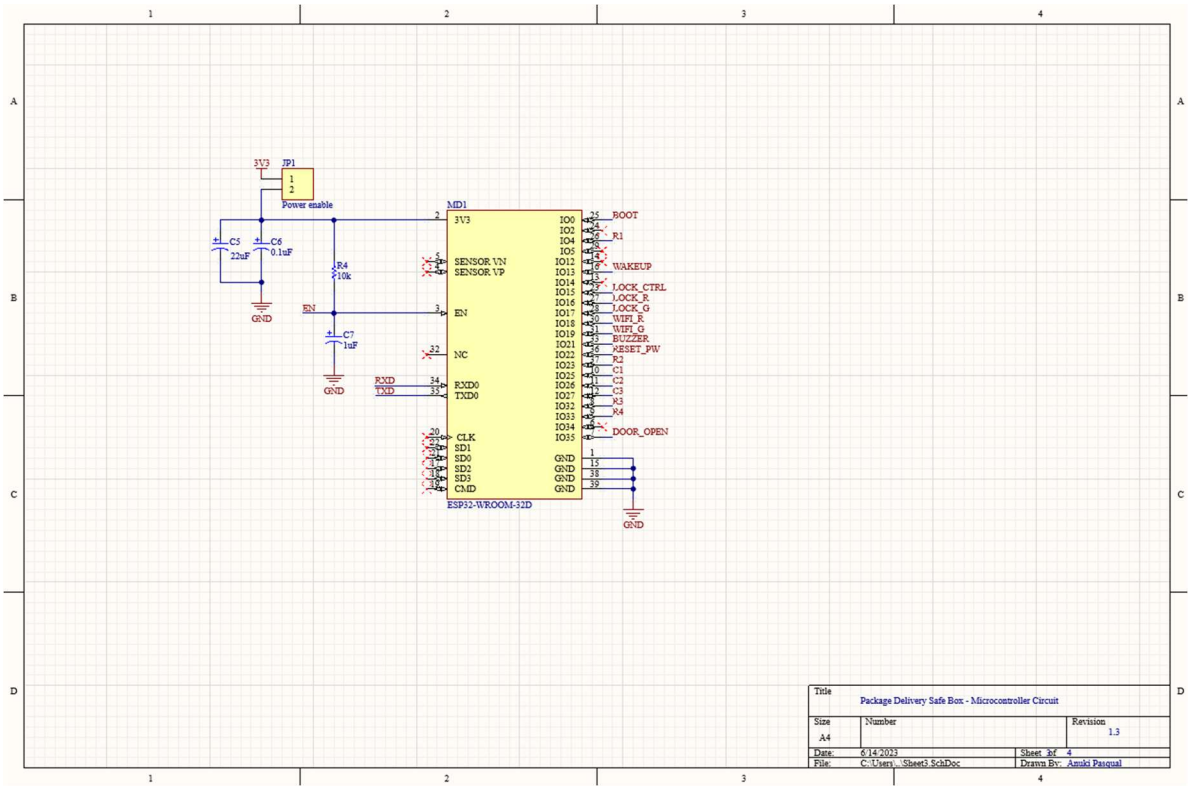
11.2 Preliminary Design Stage

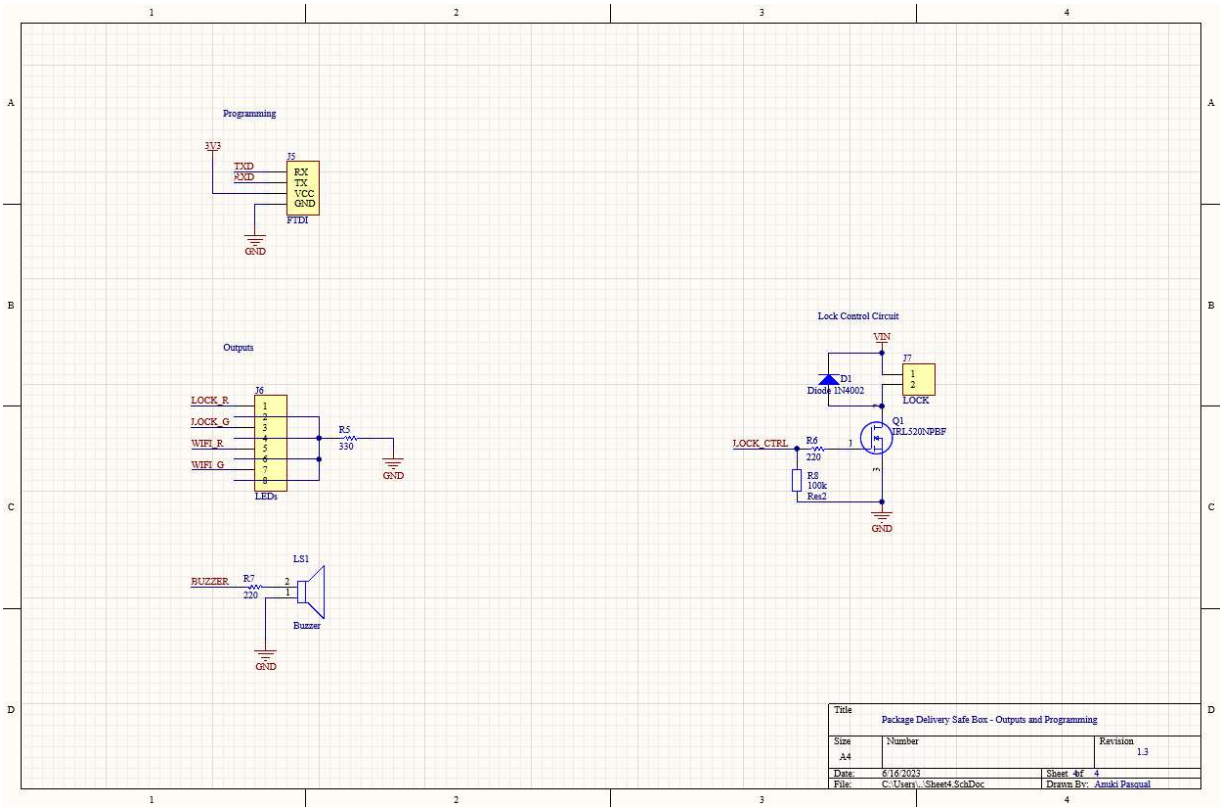
In this stage, the schematic design and Solidworks design was done for the previously selected conceptual designs.

Schematic Design in Altium



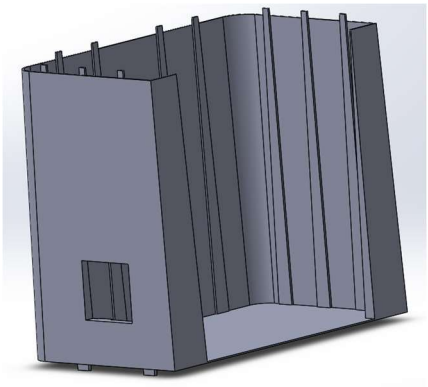


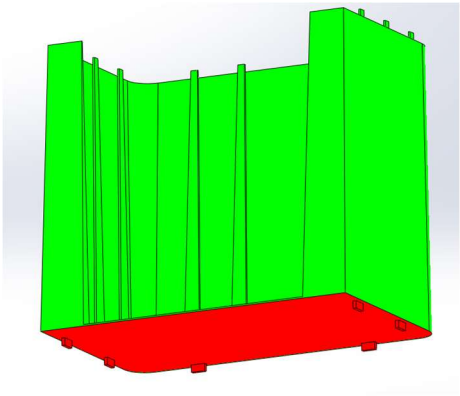

3V3
JP1
Power enable
MD1
3V3
SENSOR VN
SENSOR VP
EN
NC
RXD0
TXD0
CLK
SD1
SD0
SD2
SD3
CMD
IO0
IO2
IO4
IO5
IO12
IO13
IO14
IO15
IO16
IO17
IO18
IO19
IO21
IO22
IO23
IO25
IO26
IO27
IO32
IO33
IO34
IO35
GND
BOOT
R1
WAKEUP
LOCK_CTRL
LOCK_R
LOCK_G
WIFI_R
WIFI_G
BUZZER
RESET_PW
R2
C1
C2
C3
R3
R4
DOOR_OPEN
ESP32-WROOM-32D
C5
22uF
C6
0.1uF
R4
10k
C7
1uF
RXD
TXD
Title
Package Delivery Safe Box - Microcontroller Circuit
Size
A4
Number
Revision
1.3
Date:
6/14/2023
Sheet 3of 4
File:
C:\Users\...Sheet3.SchDoc
Drawn By:
Amuki Pasqual

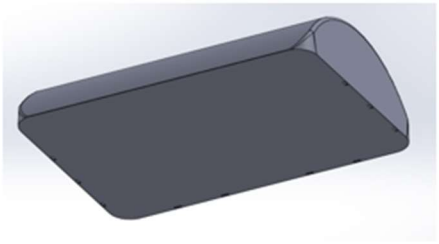

Programming
3V3
TXD
RXD
J5
RX
TX
VCC
GND
FTDI
Outputs
J6
LOCK_R
LOCK_G
WIFI_R
WIFI_G
LEDs
R5
330
LS1
BUZZER
R7
220
Buzzer
Lock Control Circuit
VIN
D1
Diode 1N4002
J7
LOCK
Q1
IRL520NPBF
LOCK_CTRL
R6
220
R8
100k
Res2
Title
Package Delivery Safe Box - Outputs and Programming
Size
A4
Number
Revision
1.3
Date:
6/15/2023
Sheet 4of 4
File:
C:\Users\...\Sheet4.SchDoc
Drawn By:
Amuki Pasqual

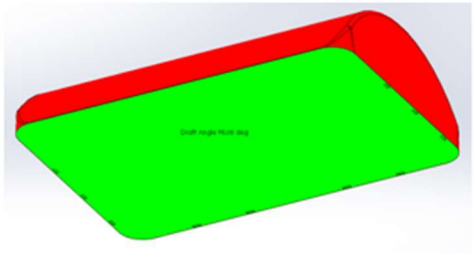
Enclosure Design in Solidworks

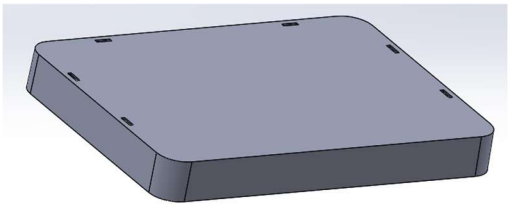
Main box (Designed as several separate parts to make it moldable)

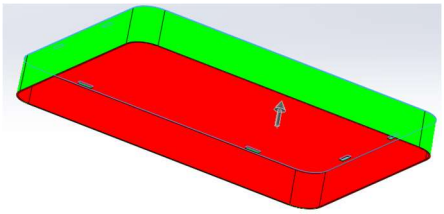


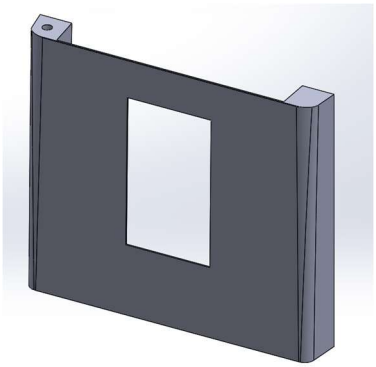


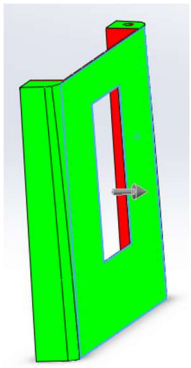




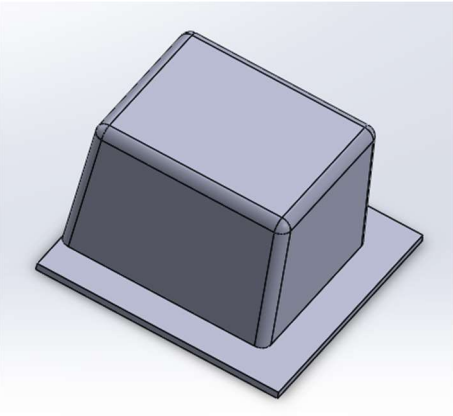


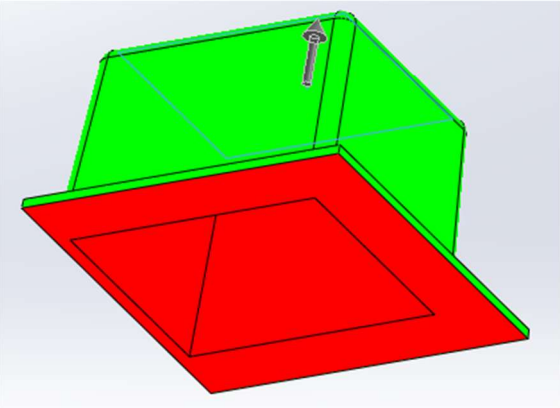




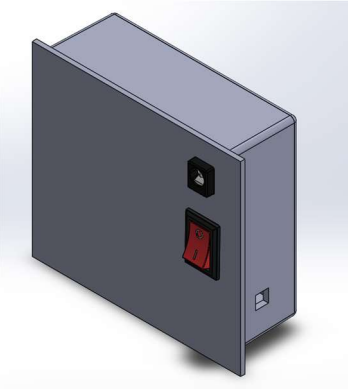


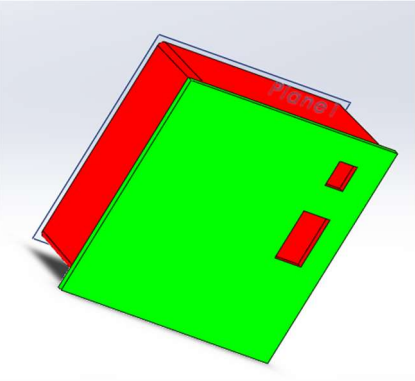
Lock Mounter (Designed to be more stable and strong)



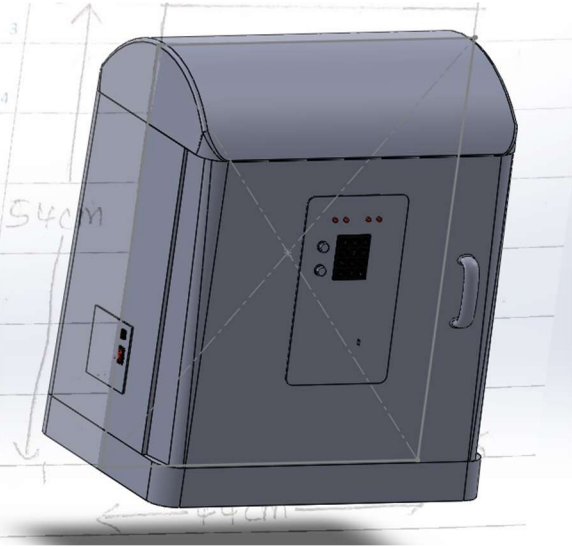


Battery Pack (Designed to be removed fully for charging)





Complete Assembly



11.3 Other improvements that can be implemented in future versions

- Currently it is only possible to unlock the box electronically. This could be a problem if some fault occurs in the circuit as there will be no way to open the door. Therefore, the locking mechanism can be improved to allow the box to be unlocked either with the passcode or a physical key.

- Some of the wiring between the lock, PCB and battery case is exposed when the door is opened. The enclosure can be designed to allow wiring through a hidden path to eliminate exposed wires.

- Making the device work with both main AC power and batteries can make it more convenient for the user depending on their individual requirements.

- More protection features can be added to deter theft.

12.User Manual

Introduction

The DeliveryBox provides a convenient way for you to keep your packages safe if no one is at home when it is delivered. You can remotely set a one-time password (OTP) for the delivery driver to unlock the box and leave the package. There is a different master password for the owner to use.

Package Contents

- DeliveryBox device
- Battery charger
- User manual
- Warranty card

Specifications

- External dimensions : 44 x 31 x 49 cm (L x W x H)
- Internal dimensions (storage area) : 43 x 30 x 39 cm (L x W x H)
- Power supply : 3 x 3.7 V rechargeable batteries
- Net weight : 8.2 kg

User Interface

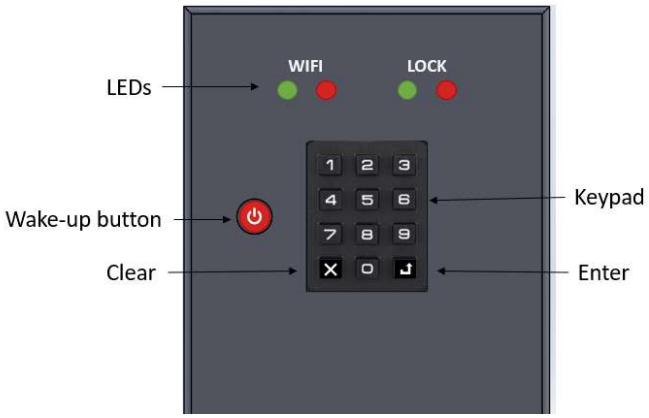


Figure 1: Outside interface

LED Indicators

WIFI Green: Wi-Fi connected
WIFI Red blinking: Wi-Fi not connected
WIFI Red: Trying to connect
LOCK Green: Unlocked
LOCK Red: Locked
LOCK Green blinking: In password changing mode

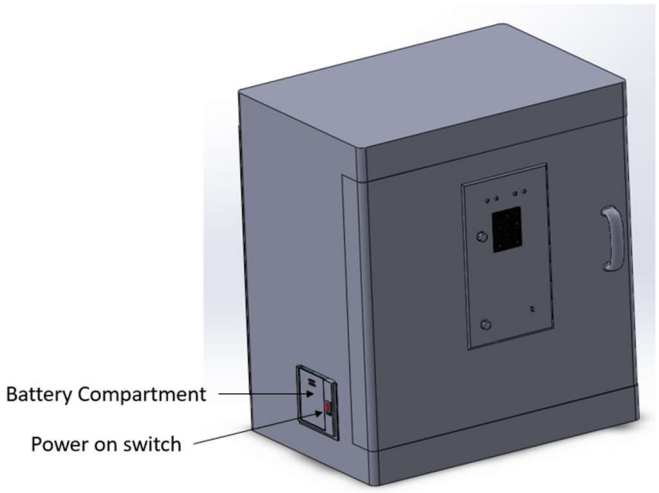


Figure 2: Side View

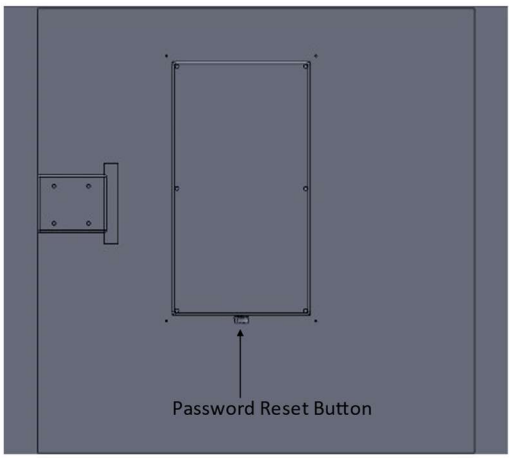


Figure 3: Back side of the door

Operating the device for the first time

IMPORTANT NOTICE: Please make sure that you change the master password after opening the box for the first time, for security reasons.

DEFAULT FACTORY MASTER PASSWORD: 1234

- Turn on the power on switch (see Figure 2).
- Wait for the red LOCK light to turn on. Ignore the WIFI lights.
- Enter the password 1234 on the keypad and press the enter key (see Figure 1). The green LOCK light will turn on and the door will be unlocked. Open the door within 10 seconds.

Resetting the master password

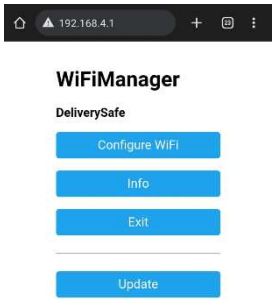
- Open the box using the master password. The password reset mode will not be available if the one-time password is used instead.
- Locate and press the password reset button on the back of the door (see Figure 3). The green LOCK light will start blinking.
- Enter the new password and press the enter key. The password can be between **3-10** digits long. Do not use personal information like birthdays, telephone numbers etc. for your password.
- After pressing enter, the green LOCK light will turn off and the red LOCK light will turn on. The device is in normal operating mode again.

Opening the door using a password

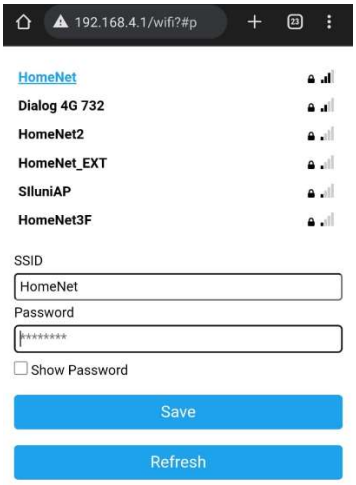
- If the device is powered off, turn on the power switch.
- If the device is powered off but all the LED lights are off, press the wake-up button (see Figure 1) to bring the device out of sleep mode.
- Wait for the red LOCK light to turn on.
- Enter the master password or the OTP if it has been set, and press enter.
- Each time a key is pressed, a beep will be heard. The key press has not been registered if you do not hear the beep.
- If the password is correct, the green LOCK light will turn on and the door will be unlocked. Open the door within 10 seconds.
- If an incorrect password is entered, the door will not unlock and a error beep sound will be heard. Entering an incorrect PIN three consecutive times will freeze the locking mechanism and keypad for 20 seconds.
- If you started entering the wrong digits, press the clear button (see Figure 1) to clear all the digits you entered during this attempt.

Connecting to a Wi-Fi network

- If the device is powered off, turn on the power switch.
- If the device is powered off but all the LED lights are off, press the wake-up button (see Figure 1) to bring the device out of sleep mode.
- Wait for the red LOCK light to turn on.
- If the device is not connected to Wi-Fi, the red WIFI light will start blinking.
- Press 0 on the keypad and press enter. The red WIFI light will light without blinking and. The device will go into Wi-Fi configuration mode and act as a Wi-Fi access point.
- From a mobile phone or any other suitable device, connect to the Wi-Fi network named DeliverySafe which should be visible in the Wi-Fi settings.
- You will be redirected to a webpage with the address 192.168.4.1. Manually go to this address if you are not redirected.



- Click on “Configure WiFi”



- Ensure that the network you plan to connect to is in the displayed list. Otherwise it means that the Wi-Fi signal is not reaching the device.
- Click on the required Wi-Fi network name and enter its password.
- Click save. If the connection is successful, the green WIFI light will turn on. Otherwise you can refresh the web page and try connecting to a different network, or go back and click “Exit” to go back to normal operating mode without Wi-Fi.

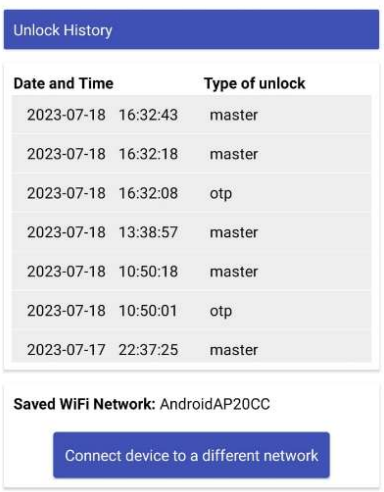
Setting and using a one-time password (OTP)

- Download the DeliverySafe mobile app from App store or Google Play store.

- When the app is used for the first time, the home screen should display “No OTP set”. Otherwise it will display the currently set OTP.

- Press “Regenerate OTP” to randomly set a new OTP.
- Press the copy icon next to the OTP to copy it to the clipboard if needed.
- Communicate the OTP to the person opening the door through a message or a call.
- Instruct them to press the wake-up button and wait for the red LOCK light to turn on before entering the password.
- After they have successfully opened the door using the password, you will receive a notification in the mobile app and the OTP will be removed.
- If they need to open the door again for some reason, a new OTP can be generated.

- The history of unlocking the box and the currently saved Wi-Fi network can be viewed in the app. It will also contain instructions for connecting the device to a Wi-Fi network similar to this manual.



Charging the batteries

- Empty the box before charging the batteries.
- Power off the device.
- Slide open the cover of the battery compartment (see Figure 2).
- Remove the 3 Lithium-ion batteries from the case.
- Insert them into the provided battery charger and charge for around 5 hours or until the light turns green.
- Insert the batteries back into the battery compartment.