Smart and Secure Mailbox

With a rise of online stores such as Amazon and eBay it has become a lot easier to shop online. Online shoppers not only save time by eliminating a trip to a local store but also save money by buying merchandise for less. Such convenience comes with certain risks. Those risks include losing your payment information online, receiving bad or damaged product, or maybe not receiving it at all. But if you are not home when delivery shows up your package can be stolen. That is why I have decided to build a smart mailbox to keep packages safe. It would protect a package from the moment it is delivered until you or someone who has access to it retrieves the package.

My mailbox works as follows: when mailman delivers the package, he or she scans the tracking number on the package with a barcode scanner which is built into a mailbox. If the tracing number matches the number stored in the database, box will unlock for a few seconds and package can be placed inside. After that it will create a delivery log and store it in the database as well as send an alert text massage saying that mailbox has been accessed. When retrieving a package, a user has to scan RFID card to gain access to a mailbox. Once again, an access log will be created and stored in the database as well as text massage will be sent alerting user that mailbox has been accessed. Supposedly, someone tries to gain access to a mailbox with a different RFID card that doesn't match users information in the database, mailbox will remain locked and access attempt log will be created and stored in the database as well as text massage alerting of failed access attempt will be sent to a user.

For this project you will need:

Hardware:

1. A box of a desired size which will store mail packages.
2. A Raspberry Pi 4 with Micro SD and power adapter (could also use Raspberry Pi 3). (Amazon link)
3. Electric Lock DC 12V. (Amazon link)
4. 12V DC adapter. (Amazon link)
5. 5V Single Channel Relay. (Amazon link)
6. USB RFID Reader which emulate keyboard input. (Amazon link)
7. RFID Cards. (Cards are included with RFID Reader)
8. USB Barcode Scanner which emulate keyboard input. (Amazon link)
9. Breadboard jumper wires. (Amazon link)
10. Also, a conventional mechanical lock to open one of the sides incase of a power outage or system failure.

For the purpose of this project I used a small button as a bypass the whole system and unlock the lock in case something goes wrong.

Hardware part.

Step 1: Preparing the Box:

Figure 1

A box should be built out of sturdy materials and water resistant. For a demonstration purpose I have used a small carton box which was purchased at Home Goods (Figure 1). To prepare a box for hardware installation you need to plan and mark out location where equipment will be mounted. First thing that I did was to cut out 3 holes. The first small hole was cut out for a 12v DC power adapter which powers up the lock and Raspberry Pi power adapter (Figure 2). The second hole is needed to place a "back up button" incase something would not work (Figure 3).

Finally, the third cut is needed to install a barcode reader which can be seen on a Figure 4. This barcode reader activates every time it detects motion therefore its front has to be completely unobstructed.

Figure 4

Figure 2

Figure 3

Step 2: Attaching a lock

A lock comes with a set of screws and can be mounted anywhere depending on your set up. For this demonstration purpose I have glued it to my box with a glue gun. As shown on a (Figure 5). Before it is glued it has to be aliened perfectly so that when lid is closed a little latch is engaged in the lock.

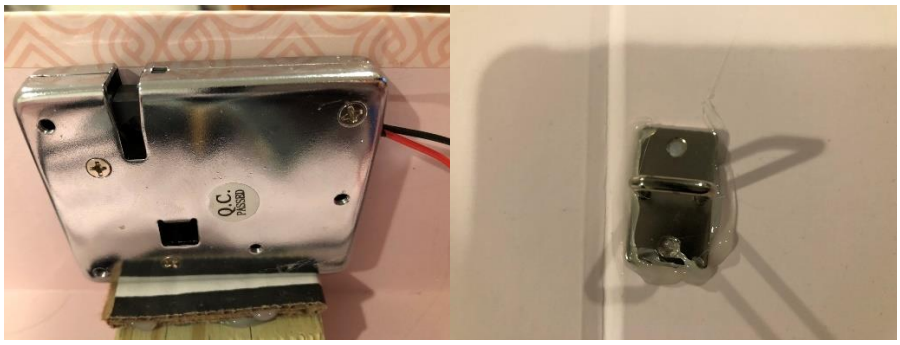


Figure 5

Step 3: Adding hardware:

Figure 6

At this time, we can add the rest of the hardware. First, I installed a “back up button” into a hole that was predrilled before. Then I glued relay at the spot where it would be easy to wire it in. Next, I placed and glued Raspberry Pi and added RFID reader alone with Barcode Scanner. Proximity RFID card will scan through thin wall so in my case I placed reader inside the box. I used “good old” duct tape to keep reader and scanner in place just in case if I have to move it. If placement is permanent, it can also be glued. Everything can be seen on (Figure 6).

Step 4: Wiring:

Now it is time to wire everything. You can follow the diagram shown on (Figure 7). It is a good idea to solder all wire splices.

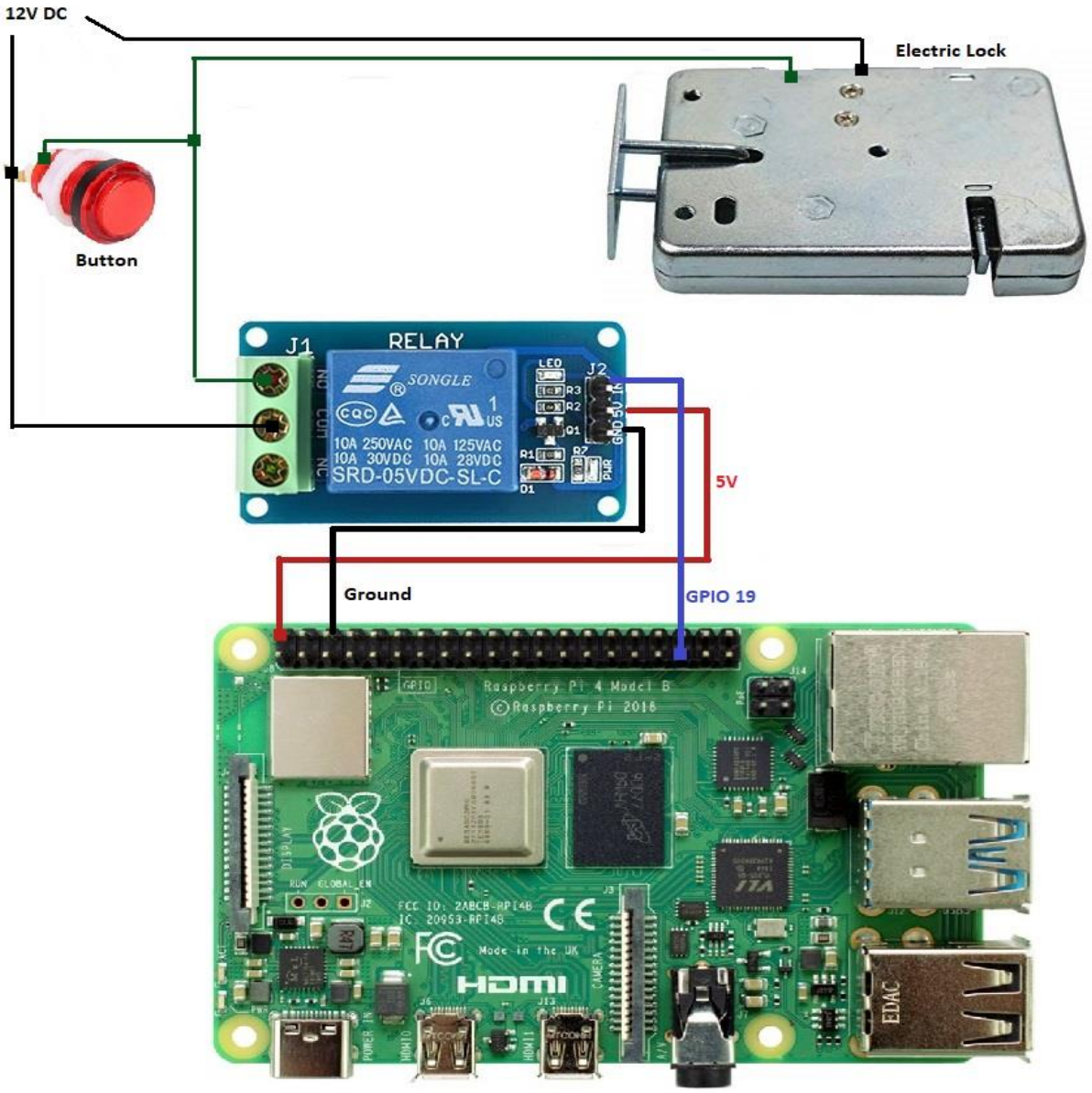


Figure 7

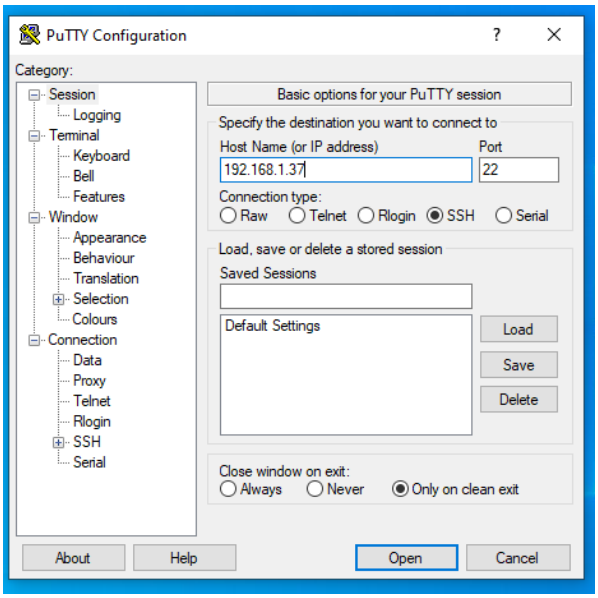
Software part:

Step 1: Preparing Raspberry Pi:

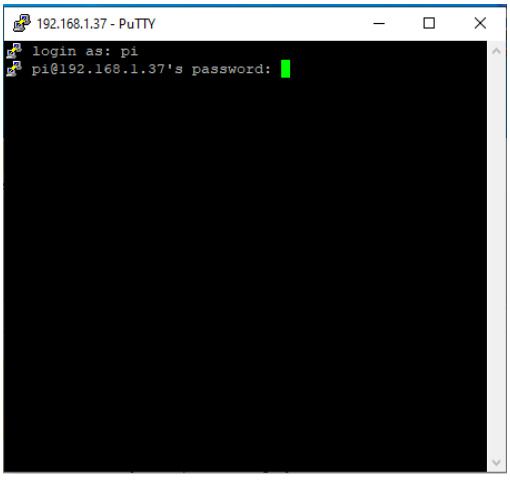
For a fresh Raspbian install it is always a good idea to start with formatting micro SD card. Installation guide can be found on official Raspberry Pi web site. After it is done we can start working on Raspberry Pi using a few different methods. The easiest one would be to connect it to a monitor or a TV via HDMI. Then connect keyboard with a mouse and start loading everything that we need for this project using

command line. Second way is to use SSH which is a great way to get terminal (command line) access to a Raspberry Pi without having to hook up a monitor, a keyboard, or a mouse. For this step we will only need an internet connection, power to power up our Pi, and a computer.

Since the IP address assigned to our Raspberry Pi may change after we reboot it, we would have to either go through fixed IP set up or use network scanner to obtain an IP preassigned to our Pi. In this demo we will use Advanced IP Scanner which could be downloaded by clicking here. Alternative way to find out Raspberry Pi's IP would be to go to your router settings which I did in my case. It was easier since I have Netgear Orbi which comes with easy to use app. After we have located our IP address we can connect to our Pi and that is done with a program called Putty. This is a free program and it can be downloaded here. After you run Putty the only thing you will need to do is to enter an IP address you obtained earlier into a space where it says, "Specify the destination you want to connect to: Host Name". Everything else should be unchanged. (As shown in a picture bellow.)



Now if prior steps were done correctly you should get a pop-up massage "PuTTY Security Alert". By clicking YES, you should be able to see terminal screen. Now it will ask you for a username which is: "pi" and a password which is: "raspberry". (This can bee seen on a picture below).



The username and a password can be changed in the settings later.

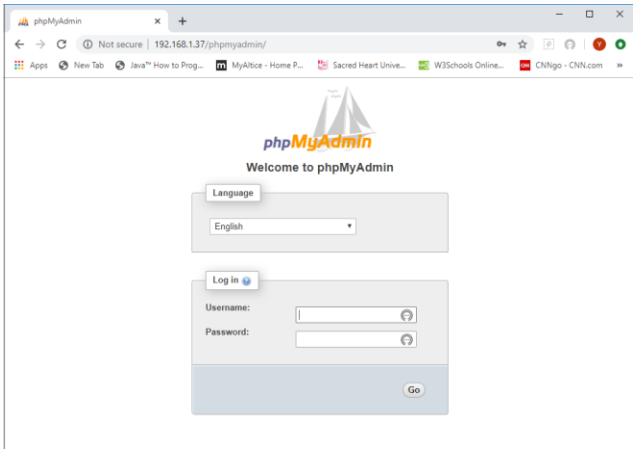
Step 2: Installing Apache2, PHP, MySQL (MariaDB Server), and phpMyAdmin on a Raspberry Pi:

After we have our terminal open, we can begin installing everything that we need for this project on to our Pi. But before any installation it is always good idea to check for any updates available for you Pi and its done with commands: "**sudo apt update**" and then after update is done run "**sudo apt upgrade**". When it is completed, we can install our web server software which is Apache2. This is done by typing "**sudo apt install apache2**" command.

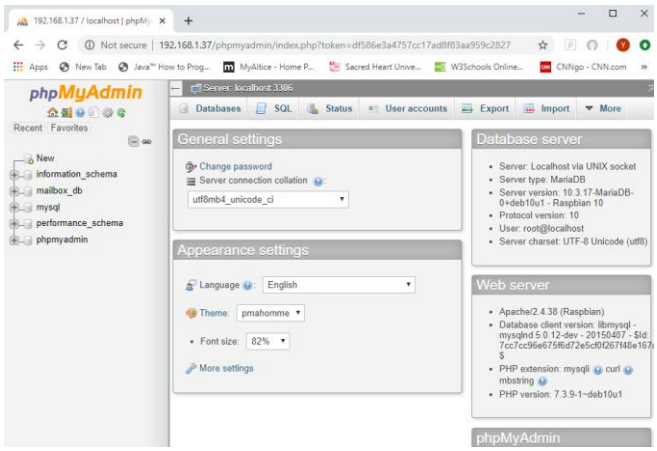
Now we need to install PHP which is the scripting language used for dynamic web applications. To do that we need to run another command "**sudo apt install php**". Finally, we are ready to install our relational database MySQL which also requires PHP-MySQL extension. For installation run this command: "**sudo apt install mariadb-server php-mysql**". After installation is complete, we need to restart our apache service to be able to proceed. So, we need to run another command which is: "**sudo service apache2 restart**". Now we can secure our MySQL Installation with this command: "**sudo mysql_secure_installation**". A massage "Enter current password for root (enter for none)" will appear in the command line or a terminal. That is where we set a password which we will need later to set-up phpMyAdmin. Reenter new password and answer "Y" per every prompt. If everything was set up properly you should see a massage "**Thanks for using MariaDB!".**

At this point we are ready to install phpMyAdmin which is a free software that make it a lot easier to use MySQL trough the web interface. To install it run this command: "**sudo apt install phpmyadmin**". During installation you will be prompted with a few questions. Choose "Yes" when asked this question: "Configure database for phpMyAdmin with dbconfig-common?". Then choose Apache2 when prompted and keep confirming following props. When prompted with a password please enter password we have created earlier. Restart Apache2 again with: "**sudo service apache2 restart**".

Finally, at this point we can login to phpMyAdmin using root as a username and a password we have created. To do that go to your browser and type your Pi's IP address and followed by /phpMyAdmin (xxx.xxx.x.xx/phpMyAdmin). In my case it was 192.168.1.37/phpmyadmin. If everything is set up properly you should see a page shown below.

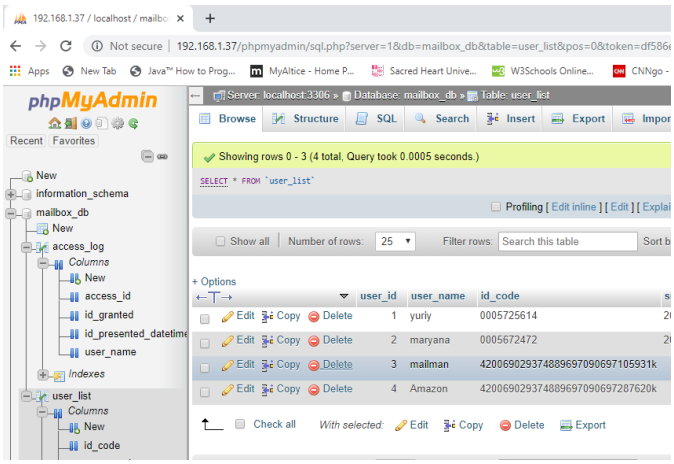


Like I mentioned earlier you should use "root" as username and password you have created to login. Once you have logged in you can start building database which will store data needed for this project.

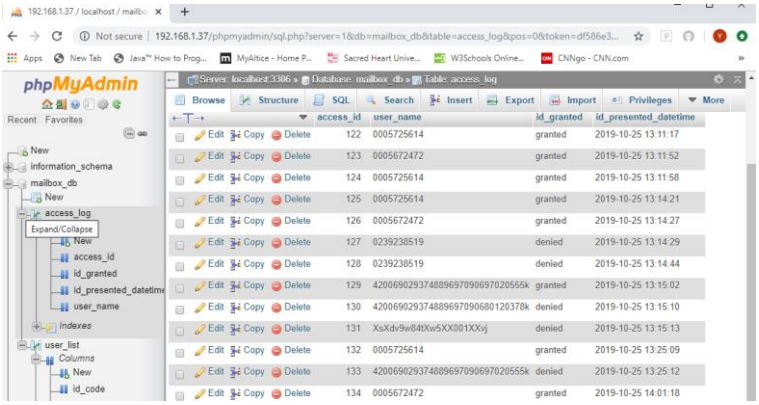


Step 3: Creating Database.

For this project we will create very simple database with 2 tables. One is to store user information and the other to store access information. In a user table I only have user_id as a primary key, user_name, and id_code. User_id autoincremented number for every user added to this project. User_name is obviously a user's name for easier distinction and id_code contains unique token for RFID card or tracking number of the package.

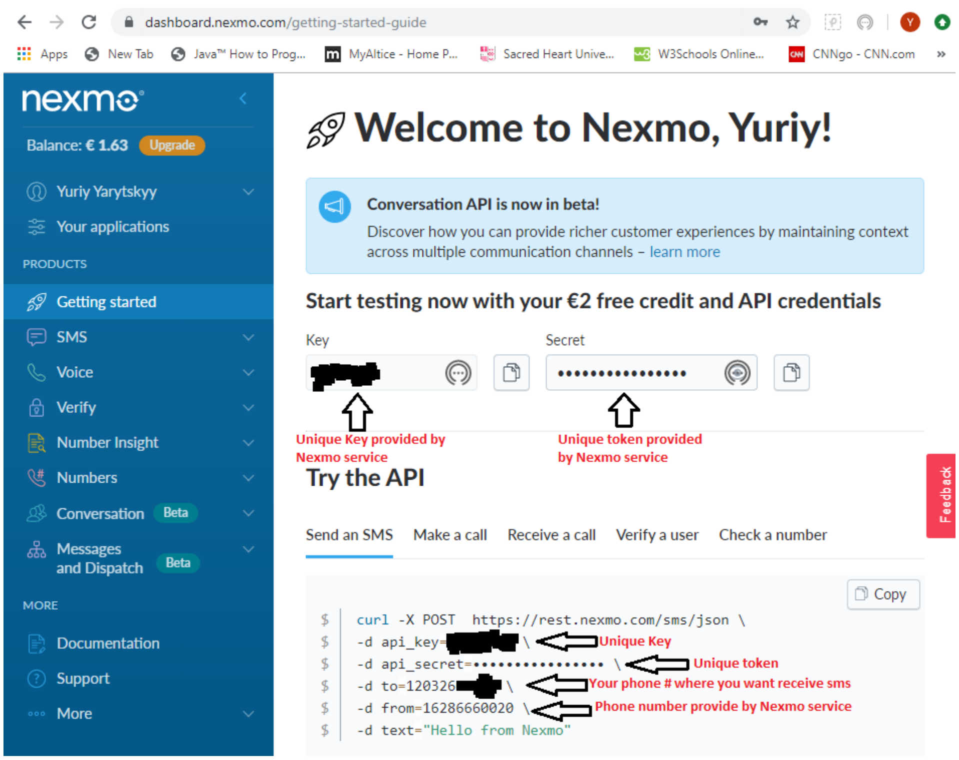


Second table contains access_id as a primary key, user_name, id_granted, and id_presented_datetime. Access_id is autoincremented number for every access transaction to a mailbox. User_name is different form user_name in first table. This user_name is to store every RFID code or Barcode scanned even the once that are not allowed to access mailbox. We use id_granted to note if access was granted or denied. Finally, id_presented_datetime is used to store a timestamp of mailbox access or access attempt.



Step 4: Setting up Nexmo SMS service

Nexmo is a service that allows you to send and received taxed massages. Go to https://dashboard.nexmo.com/getting-started-guide and set up free account. They provide free trial credit which is more then enough for few small projects like this one. I have been testing a lot and burned less the a quarter of trial credit balance. Once you have signed up you will receive a special key and a secret token which you will need to use in the code. Nexmo also provides python code to use their service.



Step 5: Python Script

Now we can start working on Python scrip which will bond together everything we have done so far.

mailbox2Fiinal.py - H:\last\mailbox2Fiinal.py (3.6.6)

File Edit Format Run Options Window Help

```python
#!/usr/bin/env python3
import mysql.connector
from evdev import InputDevice
from select import select
import RPi.GPIO as GPIO
import time
import nexmo


#Nexmo service login and text massage defenition
client = nexmo.Client(key='[illegible]', secret='[illegible]')
receiver = '1203[illegible]5'
message = '''Alet!!! Mailbox Has been accessed!!!'''
message2 = '''Warning!!! Unauthorized access attempt to the Mailbox!!!'''
```

Code 1

As shown on the screenshot above (Code 1) first we need to define all the modals that are needed for this project. If some of them are not automatically included in the Python we have to make sure that we download them to the same location on our Raspberry Pi where main script will reside.

Then we start with defining Nexmo authentication where we will need to use the “key” and the “secret” provided to us by Nexmo during registration process. Also, we define “receiver” which is the phone number where you want to receive your SMS notifications as well as text of notifications itself. Most of this code is provided by Nexmo service on their webpage.

```
#Raspberry Pi Pin utilization / i have chossen pin #19
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(19,GPIO.OUT)
```

Code 2

The second screenshot (Code 2) shows how to define a pin use on Raspberry Pi. When you set it to “high” pin is active and when it is switched back to “low” it becomes inactive. Like on/off switch. I chose pin number 19 because it gave me plenty of space to connect a small touchscreen. I only connected it for additional convenience, but it is not needed for this project.

```
#Input of USB devices interpreted as keyboard input / in this can USB-RFID Reader and USB-Barcoad Scanner
#bouth devices support keyboard input
keys = "X^1234567890XXXXqwertzuiopXXXXasdfghjklXXXXXyxcvbnmXXXXXXXXXXXXXXXXXXXXXXX"
devices = map(InputDevice, ('/dev/input/event0','/dev/input/event1'))
devices = {dev.fd: dev for dev in devices}
id_presented = "" #device defined
```

Code 3

This next section of the code shown on the screenshot above is a part of python-evdev library which allows us to use RFID SCANNER and BARCODE READER as a generic keyboard input. I have found a very good tutorial about this online which you can see here.

```
# Function to send sms when access was granted
def send_sms():
    response = client.send_message({'from' : '16286660020', 'to' : receiver, 'text' : message})
    response = response['messages'][0]

    if response['status'] == '0':
        print ('sending SMS')
    else:
        print ('Error')


# Function to send sms when access was denied
def send_sms_access_denied():
    response = client.send_message({'from' : '16286660020', 'to' : receiver, 'text' : message2})
    response = response['messages'][0]

    if response['status'] == '0':
        print ('sending SMS')
    else:
        print ('Error')
```

Code 4

Next step is to create functions to send SMS notifications. The first one is to send notification for whenever access to mailbox was granted and the second one is for whenever it was denied. Both are the same accept they use different massages which we have defined earlier. Once again, the phone

number “16286660020” shown in the code was provided by Nexmo service. Also, because I had a touchscreen attached to my Raspberry Pi, I have these functions print statement weather there was an error or SMS is being sent for my own convenience.

```
#Main
while True:
        r,w,x = select(devices, [], [])
        for fd in r:
            for event in devices[fd].read():
                if event.type==1 and event.value==1:
                        if event.code==28:

                                #DATABASE login authentification and user authentification
                                mydb = mysql.connector.connect(host="localhost", user="root", password="[illegible]", database="mailbox_db")
                                mycursor=mydb.cursor(dictionary=True)
                                mycursor.execute("SELECT * FROM user_list WHERE id_code = '%s'" % (id_presented))
```

Code 5

Finally, we can start working on main section of our script. As you can see on the screenshot above (Code 5) first section of the while loop checks “evdev” for an input. Once we receive input from an RFID Reader or a Barcode Scanner the connection to our database is made where it checks user table record to validated user’s access. Since the database is residing on Raspberry Pi host is localhost. To connect to a database, you have to use same username and password you used when you logged into phpMyAdmin. Also name of your database has to be defined.

```
        #Loop to check access permision
        for (id_code) in mycursor:
                #Mailbox access granted/mailbox unlocked for 4 second
                #sending sms
                print("Mailbox is unlocked for 4-sec")
                GPIO.output(19,GPIO.HIGH)
                send_sms()
                time.sleep(4)

                # Mailbox locked
                print("Mailbox is locked")
                GPIO.output(19,GPIO.LOW)
                #Access log created and inserted into a database
                mycursor.execute("INSERT INTO access_log SET user_name = '%s', id_granted = 'granted', id_presented_datetime = NOW()" % (id_presented))
                mydb.commit()
                break
        else:
                #Unathorized acces attemped/access denied/access log created and inserted into a database
                #sending sms
                print("Access Denied.")
                mycursor.execute("INSERT INTO access_log SET user_name = '%s', id_granted = 'denied', id_presented_datetime = NOW()" % (id_presented))
                mydb.commit()
                send_sms_access_denied()

        id_presented = ""
else:
        id_presented += keys[ event.code ]
```

Code 6

Last part of the code is shown on screenshot above (Code 6). Here we can see that if device input matches data from the database, Raspberry Pi’s pin number 19 is engaged and a lock opens for 4 seconds. Then SMS containing “Alert!!! Mailbox Has been accessed!!!” is sent to a user. After 4 seconds passes Raspberry Pi’s pin is set to low which locks the mailbox. Then access log is created and stored in the database. Incase if device input doesn’t match user records in the database, “else” statement is triggered, and mailbox remains locked. But It also sends SMS “Warning!!! Unauthorized access attempt to the Mailbox!!!” to notify user and creates another access log which is also stored in the database.

References:

1) **Three factor authentication using RFID Reader**
   https://www.switchedonnetwork.com/2017/11/10/build-the-ultimate-door-security-system-with-three-factor-authentication/

2) **RaspberryPi Raspbian installation guide**
   https://www.raspberrypi.org/documentation/installation/installing-images/README.md

3) **Python-evdev tutorial** https://python-evdev.readthedocs.io/en/latest/tutorial.html#accessing-evdev-constants

4) **Raspberry Pi: Install Apache + MySQL + PHP (LAMP Server)**
   https://randomnerdtutorials.com/raspberry-pi-apache-mysql-php-lamp-server/

5) **Nexmo Service** https://dashboard.nexmo.com/getting-started-guide

6) **Advanced IP Scanner download** https://www.advanced-ip-scanner.com/

7) **PuTTy download** https://www.putty.org/