Smart Device System

Smart Box

학번	이름	학번	이름
201939089	An Eui Hyun	201939090	Alabdulwahab Abrar Sami S
학번	이름		
201939892	Aye Thiri		

Table of Contents

Abstract

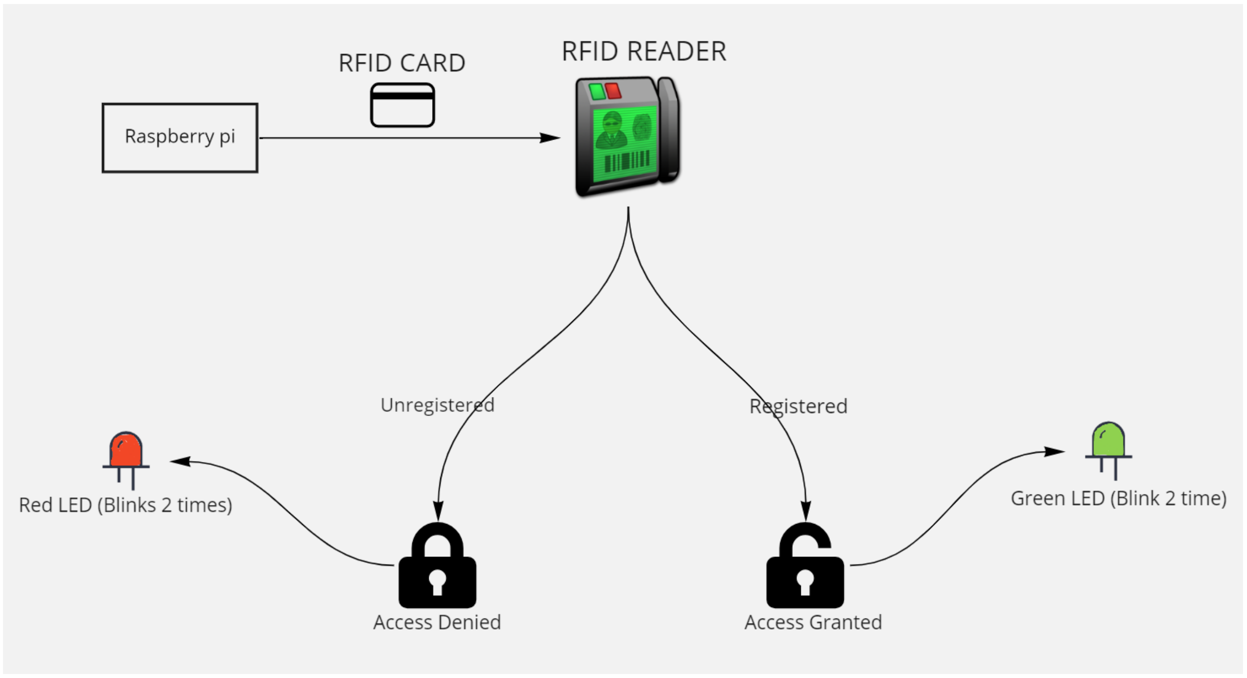
The Internet of Things (IoT) refers to objects with unique identities that are linked to the internet. It is not about something simply about things of the devices and etc. They have distinct identities and those connected to the internet with a focus of making people's lives more convenient. IoT refers to the controls, configurations, networking of the devices, and objects/things over the Internet. Traditionally sensors such as thermostat, motion sensors, and etc. were connected only via Bluetooth connection. IoT filters, processes, categorizes and context to extract the data and creates information from those lower-level data. Therefore, allowing progression to move forward to the main objective goal and allows it to perform intelligently. In this paper, we will focus on a smart home security that allows residence to remotely access or deny someone using smart automated gadgets. By using his or her mobile device they can easily keep intruders at bay. Using the security methods, we have created, it can keep the owners and their belongings safe. In other words, it will provide a slew of additional advantages allowing them to investigate the intruders. We will be exploring smart home and security in this paper, as well as the tools for smart home security.

Introduction

A smart home is called an Automated or Intelligent home, and it refers to the automation of regular tasks in home using electrical appliances. This includes the control of lights, air-conditioning, house surveillance with cameras for interiors, and etc. After evolving significantly over the previous century our technology has revolutionize our smart devices that allows our human society to be more secure and convenient. For example, our traditional security was based on real human personal to watch the CCTV or walk around the premises to secure a specified area. However, simple technology and the internet allowed us to eliminate the hassle of having a real human personal watching the CCTV all day and it keeps the specified area more secure 24/7.

This paper is about a security Smart box which can be converted to any other locks such as doors and entrances. Our technology can lock and unlock the box through an RFID technology and capture images as long as you scan the RFID. It will alert the owner/user who try to access the box or who did access the box by sending them the picture and the NFC tag identification tag number through their email. If the owner/users spouse needs access to your box when you are at work you can simply access the box through our application, we have made to control the box mechanism.

Related Work

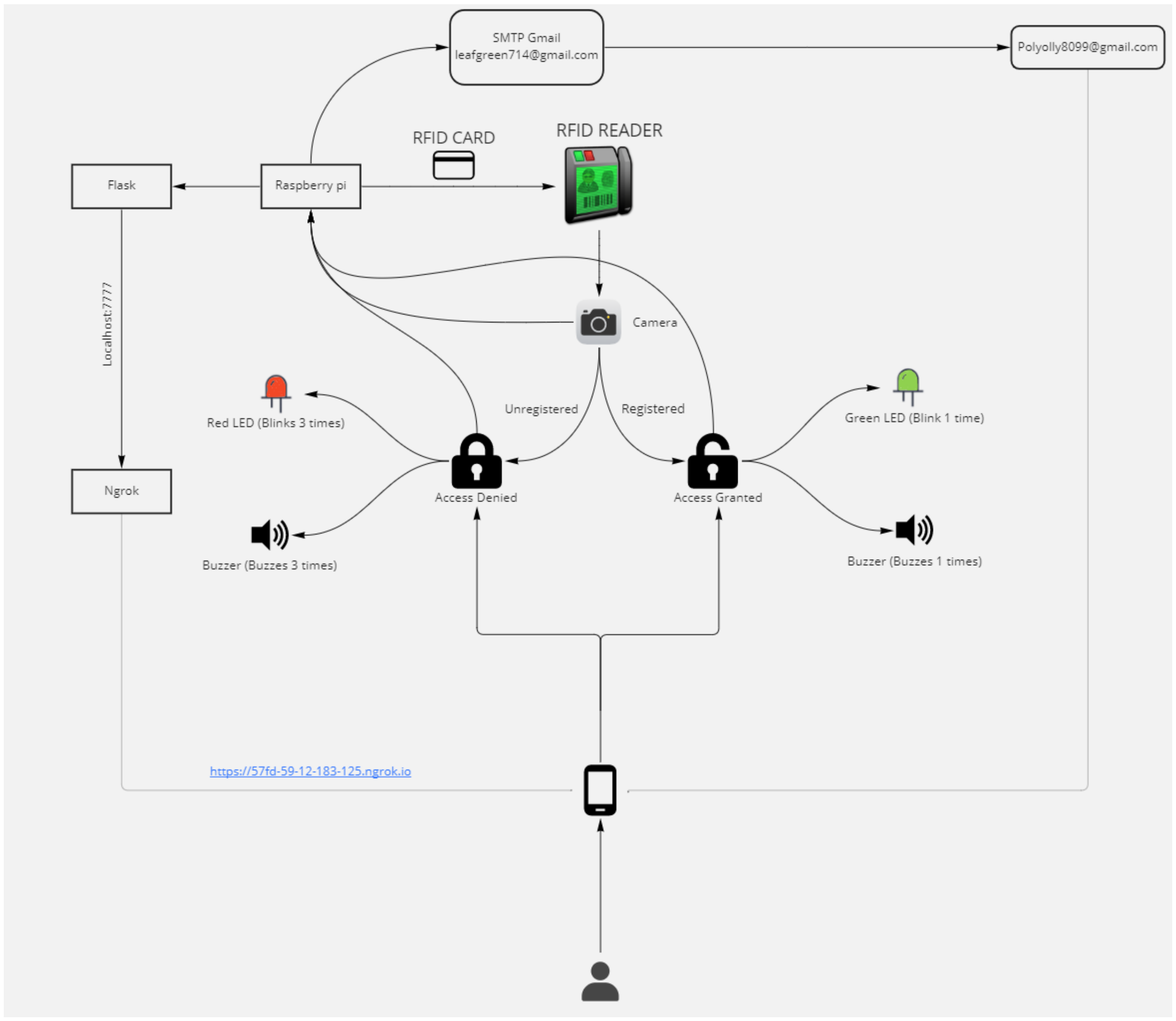


Previously, the project was built with own RFID door using an Arduino, an RFID sensor, and a few other components. As a result, a list of accepted tags will be loaded, allowing for limited access through the door.

The Raspberry Pi is used to link the various components. To begin, set up the circuit on a breadboard to ensure that all components are operational and to register the two RFID tags that will be used to lock and unlock the door. The green LED will serve as the servo mechanism's authorization and trigger. The red LED indicates that the RFID has been read but not validated, and hence the mechanism remains locked.

After that, they designed a servo bracket and placed the servo below it, using the simple 3d printed lock mechanism. Because the servo movement limitations had to be placed in the code so that the servo didn't go too far in either direction. The accepted RFID tags were then saved in an array at the start of the function. It was then run through the setup code, which connects to the RFID sensor and creates a start-up LED sequence, before sending the tag number to the function, which checks if it is accepted or not. The green or red LEDs flashed to signify and lock and unlock the lock, respectively, as the result of the test. After completing all of the tests to ensure that RFID was functioning properly, the mechanism was installed on the door.

Methodology



This project is composed of

- Raspberry pi as our main system.
- RFID (Radio-frequency identification) to scan the registered cards and identify who is accessing the house/box.
- Raspberry pi camera for taking pictures of those intruders/users.
- SMTP (Simple Mail Transfer Protocol) for sending emails.
- Led, we used Red LED means access denied, and Green LED means access granted.

- Buzzer has a long buzz to represent access granted and three buzzes for an access denied.
- Servo for locking and unlocking.
- MIT Application to control the lock from another place in the world.
- HTML, CSS, and JavaScript for creating the website of the door lock through the internet.
- Ngrok for exposing the local server ports to the Internet, to control the door lock.

Raspberry pi

Raspberry pi can be considered as a small, low-cost computer with the size of the credit card that can be connected to any monitor or television. It utilizes a conventional keyboard and mouse. Even though this component is a small computer it has nearly the same functionality as a desktop maybe even more. It can access the internet and etc. Furthermore, raspberry pi can access the GPIO ports to allow creative users and developers to innovate better technology for the future. Therefore, we choose raspberry pi as our main system for our project as it is very convenient and easy to utilize.

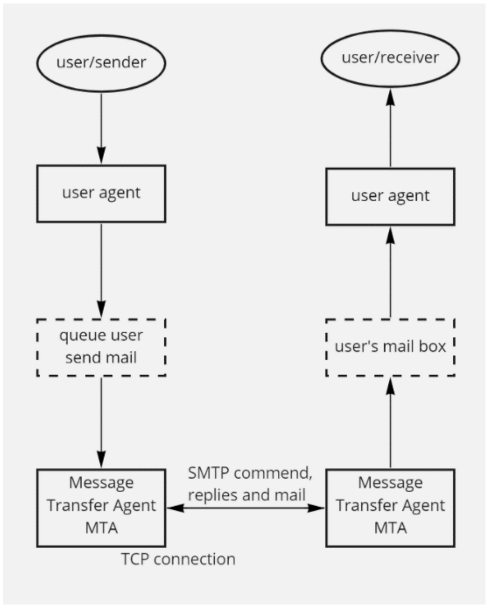
RFID (Radio-Frequency Identification)

The figure above shows the RFID of our initial door lock. It was coded with python to read and write the owner's key fob and identify and record who is accessing the house/box and time in detail. Due to the key fobs/ NFC tags having their own unique identification code and on top of that we can add additional data such as someone's name to those NFC tags. These NFC tags identify the data by using the electromagnetic field. In order to use a key fob to open the door, simply place the fob in front of the RFID reader. Each key fob has its own microchip, which the RFID reader will recognize. The door will automatically be open if you present the correct key fob to the reader. In this project, we will utilize the key fobs for the initial security.

Raspberry Pi camera

We installed the raspberry pi camera to take pictures of the people who tries to access the house/box. After scanning the key fob/ card the camera will take a picture, whether the scanning card is correct or not to notify the owner/user who is accessing their house/box.

SMTP (Simple Mail transfer protocol)



Simple Mail Transfer Protocol (SMTP) is when the client attempts to send mail establishes a TCP connection with the SMTP server and sends mail through it. The receiving mode of the SMTP server is always on. The SMTP process starts the connection on port as soon as the client receives the TCP connection. The client process immediately sends mail after successfully establishing a TCP connection. In this project we use Gmail along with the port 587. We used SMTP in order to send email to notify the owner of the intruder/user by sending capture photos of who tried or did access the house/box.

LEDs

In our project we also used LEDs, Red LED and Green LED. The usage of LEDs is for notifying the person if the process of accessing was granted or denied. The Green LED is for access granted and the Red LED is for access denied.

Buzzer

The Buzzer will work after the scanning process is done. When the scanning key fob/card is correct it is going to buzz a long buzz one time and when the scanning card is not registered it is going to buzz three times.

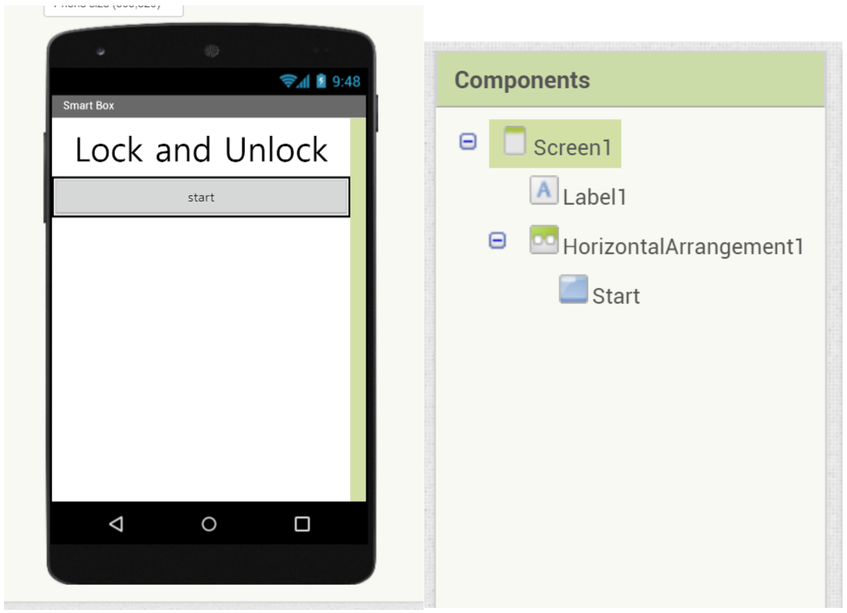
Servo

The micro servo sg90 was used in our project for locking and unlocking the door as it works just like the normal kind but it's smaller. When it is access is granted, it moves 90 degrees (unlock) and then it is going to sleep for 0.5 seconds and then it is going to the original position which is 0 degrees (lock). If access is denied then the servo will stay at 0 degrees (lock) and if it is at 90 degrees (unlock) already the servo will go to 0 degrees (lock).

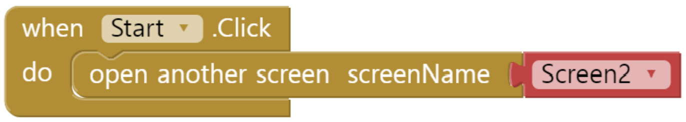
MIT app inventor

MIT app inventor is a web application integrated development environment that was created by Google. It is a drag-and-drop programming method. There is a Designer tab and a Blocks tab. The Designer tab is to design the application and in the Blocks tab, we can program the items we used. We wanted to use another way of locking and unlocking the door, which is using our phones. Therefore, we used MIT app inventor to create an application that gives the house owner the accessibility to control the servo.

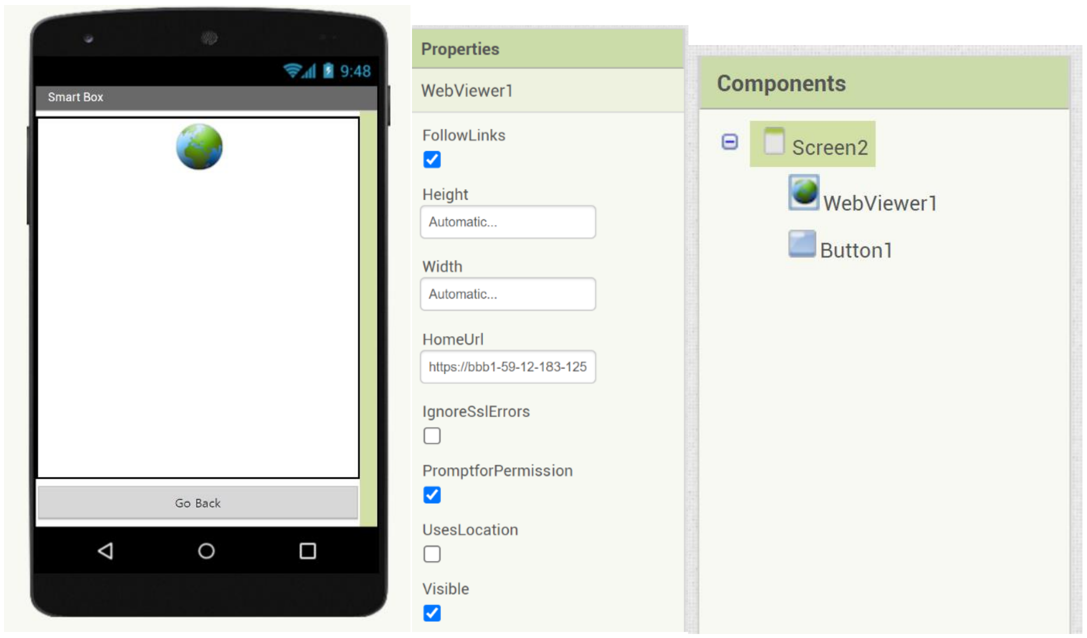
To create the application, we add two screens. One is the main screen and the other one is where the owner can control the servo. On screen 1 we added a label to write a heading for the application, and then we added a Button to be able to go to the second screen.



Then we add a block that can do an action when we click on the start button. So when it starts, Click open another screen (screen 2).



On the second screen, we added a web viewer, and in the Home URL we linked it to the URL in order to forward the Ngrok link. It is encrypted to the local host port number. It causes the bonding between the internet and the local host. Therefore, you can access the localhost from anywhere throughout the world. Also, we used a button to go back to the first page.



Properties

WebViewer1

FollowLinks ✓

Height: Automatic...

Width: Automatic...

HomeUrl: https://bbb1-59-12-183-125

IgnoreSslErrors ☐

PromptforPermission ✓

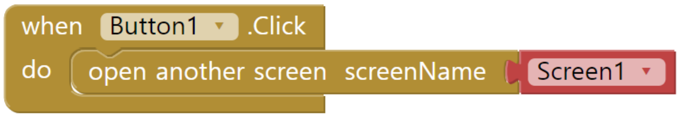
UsesLocation ☐

Visible ✓

Components

- Screen2
  - WebViewer1
  - Button1

This Block is to go to the first screen.

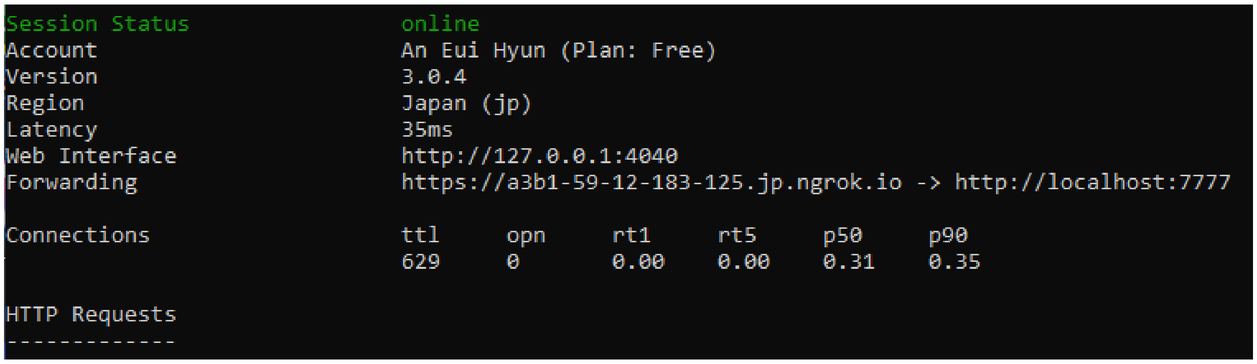


```
when Button1 .Click
do   open another screen  screenName  Screen1
```

HTML, CSS, JavaScript

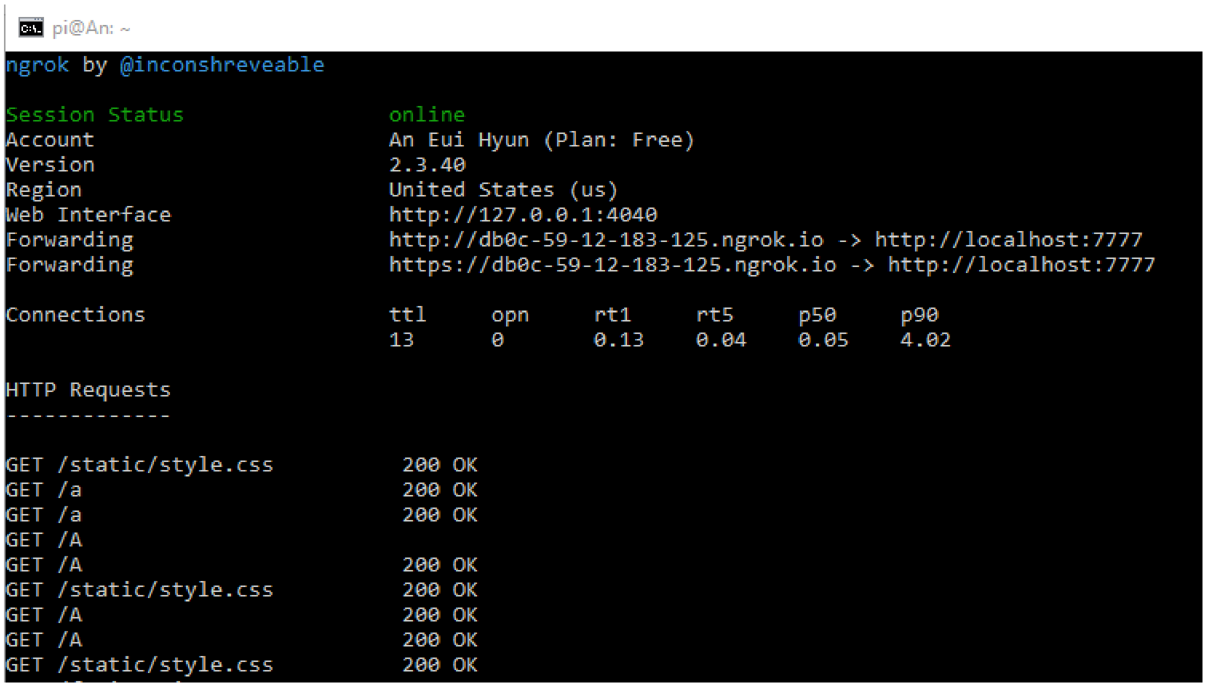
As we need the webpage to remotely access the door, we needed to create an interface for the user to control it. We made a simple html page along with the CSS and JavaScript to get access anywhere around the world.

Ngrok

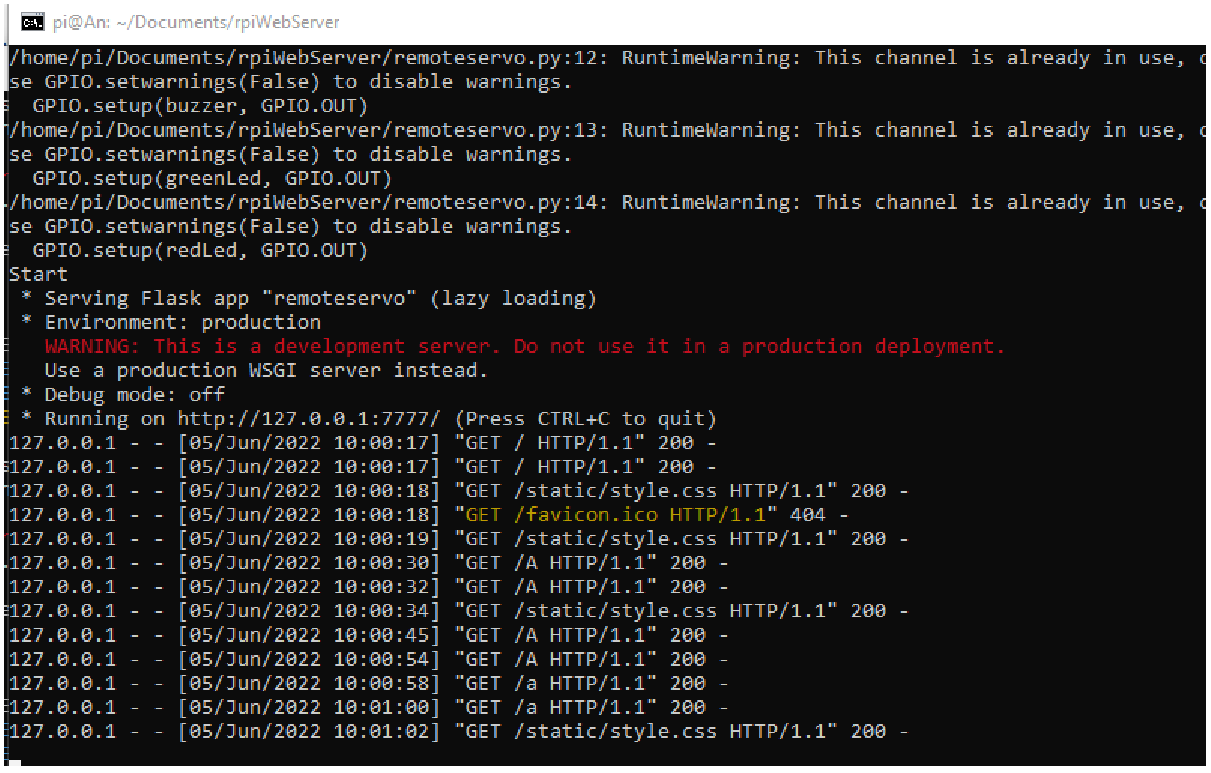


```
Session Status                online
Account                       An Eui Hyun (Plan: Free)
Version                       3.0.4
Region                        Japan (jp)
Latency                       35ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://a3b1-59-12-183-125.jp.ngrok.io -> http://localhost:7777

Connections                   ttl     opn     rt1     rt5     p50     p90
                              629     0       0.00    0.00    0.31    0.35

HTTP Requests
-------------
```

Ngrok is a cross-platform application that uses the internet to expose local server ports. By building a long-lived TCP tunnel from a randomly created subdomain on ngrok.com to the local machine, ngrok is able to overcome NAT mapping and firewall constraints. After defining the port on which our web server listens, the ngrok client application establishes a secure connection with the ngrok server, allowing the client with sole ngrok tunnel to make requests to our local host. Here, we forwarded the port and tunnel set up from the localhost port 7777 to the public at https://a3b1-59-12-183-125.jp.ngrok.io. This URL may change whenever you start over the session or when your session has crashed for safety reasons. Furthermore, the user can access our html anywhere around the world and unlock the door in case the client is not at the house/box. As a result, your HTTP Requests should constantly change when there is any interaction between ngrok and local server.

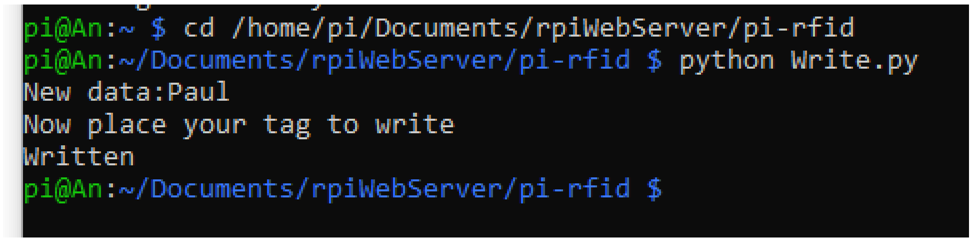


```
pi@An: ~
ngrok by @inconshreveable

Session Status                online
Account                       An Eui Hyun (Plan: Free)
Version                       2.3.40
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://db0c-59-12-183-125.ngrok.io -> http://localhost:7777
Forwarding                    https://db0c-59-12-183-125.ngrok.io -> http://localhost:7777

Connections                   ttl     opn     rt1     rt5     p50     p90
                              13      0       0.13    0.04    0.05    4.02

HTTP Requests
-------------

GET /static/style.css          200 OK
GET /a                         200 OK
GET /a                         200 OK
GET /A
GET /A                         200 OK
GET /static/style.css          200 OK
GET /A                         200 OK
GET /A                         200 OK
GET /static/style.css          200 OK
```



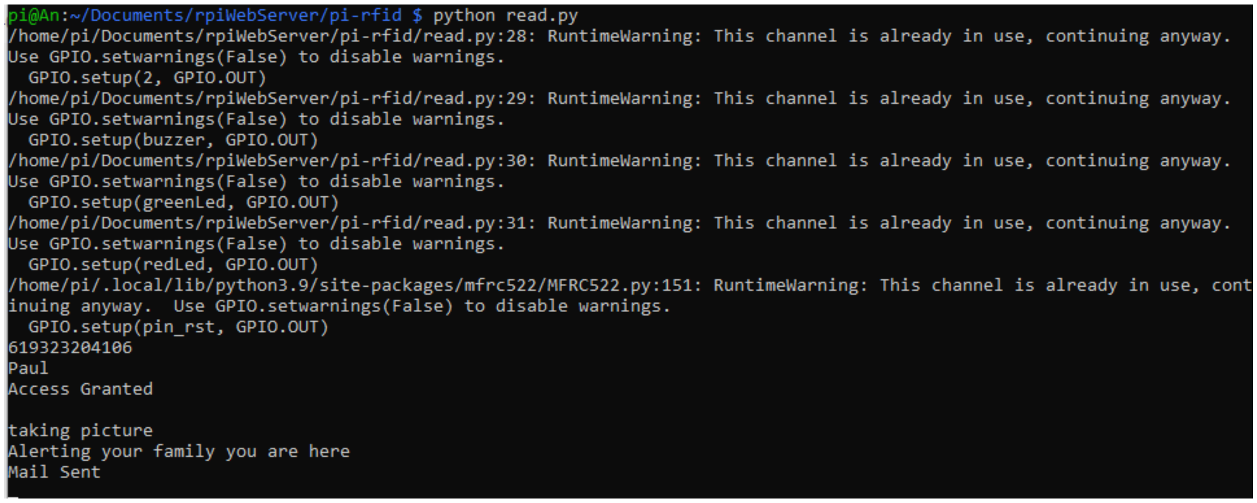
```
pi@An: ~/Documents/rpiWebServer
/home/pi/Documents/rpiWebServer/remoteservo.py:12: RuntimeWarning: This channel is already in use, 
se GPIO.setwarnings(False) to disable warnings.
  GPIO.setup(buzzer, GPIO.OUT)
/home/pi/Documents/rpiWebServer/remoteservo.py:13: RuntimeWarning: This channel is already in use, 
se GPIO.setwarnings(False) to disable warnings.
  GPIO.setup(greenLed, GPIO.OUT)
/home/pi/Documents/rpiWebServer/remoteservo.py:14: RuntimeWarning: This channel is already in use, 
se GPIO.setwarnings(False) to disable warnings.
  GPIO.setup(redLed, GPIO.OUT)
Start
 * Serving Flask app "remoteservo" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:7777/ (Press CTRL+C to quit)
127.0.0.1 - - [05/Jun/2022 10:00:17] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [05/Jun/2022 10:00:17] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [05/Jun/2022 10:00:18] "GET /static/style.css HTTP/1.1" 200 -
127.0.0.1 - - [05/Jun/2022 10:00:18] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [05/Jun/2022 10:00:19] "GET /static/style.css HTTP/1.1" 200 -
127.0.0.1 - - [05/Jun/2022 10:00:30] "GET /A HTTP/1.1" 200 -
127.0.0.1 - - [05/Jun/2022 10:00:32] "GET /A HTTP/1.1" 200 -
127.0.0.1 - - [05/Jun/2022 10:00:34] "GET /static/style.css HTTP/1.1" 200 -
127.0.0.1 - - [05/Jun/2022 10:00:45] "GET /A HTTP/1.1" 200 -
127.0.0.1 - - [05/Jun/2022 10:00:54] "GET /A HTTP/1.1" 200 -
127.0.0.1 - - [05/Jun/2022 10:00:58] "GET /a HTTP/1.1" 200 -
127.0.0.1 - - [05/Jun/2022 10:01:00] "GET /a HTTP/1.1" 200 -
127.0.0.1 - - [05/Jun/2022 10:01:02] "GET /static/style.css HTTP/1.1" 200 -
```

Results

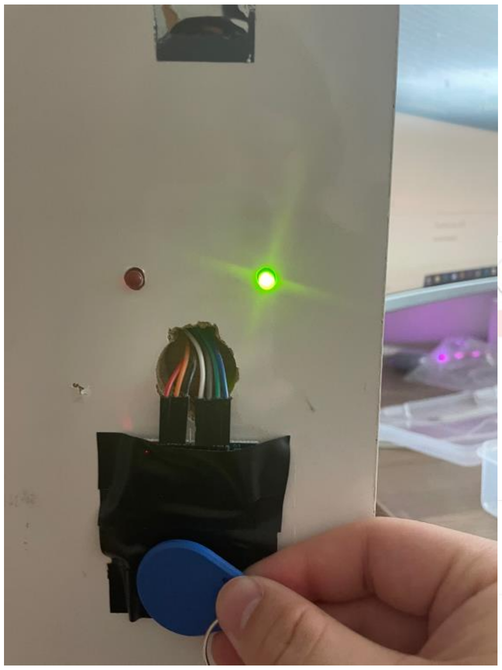


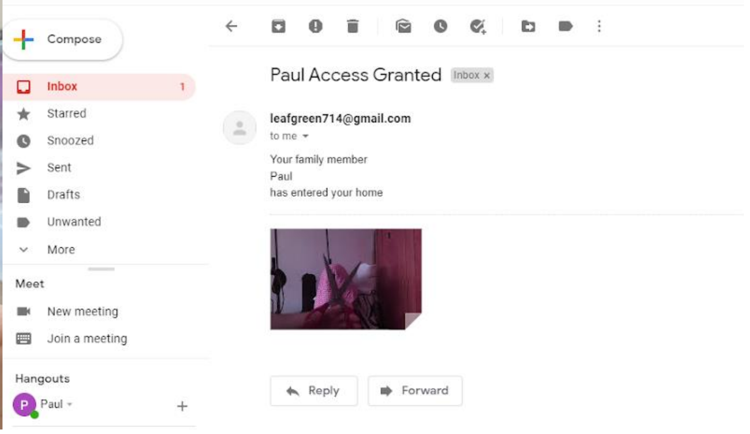
```
pi@An:~ $ cd /home/pi/Documents/rpiWebServer/pi-rfid
pi@An:~/Documents/rpiWebServer/pi-rfid $ python Write.py
New data:Paul
Now place your tag to write
Written
pi@An:~/Documents/rpiWebServer/pi-rfid $
```

Firstly, we write the data into the RFID fob. Therefore, it gets stored and listed in another id list in the python.



```
pi@An:~/Documents/rpiWebServer/pi-rfid $ python read.py
/home/pi/Documents/rpiWebServer/pi-rfid/read.py:28: RuntimeWarning: This channel is already in use, continuing anyway.
Use GPIO.setwarnings(False) to disable warnings.
  GPIO.setup(2, GPIO.OUT)
/home/pi/Documents/rpiWebServer/pi-rfid/read.py:29: RuntimeWarning: This channel is already in use, continuing anyway.
Use GPIO.setwarnings(False) to disable warnings.
  GPIO.setup(buzzer, GPIO.OUT)
/home/pi/Documents/rpiWebServer/pi-rfid/read.py:30: RuntimeWarning: This channel is already in use, continuing anyway.
Use GPIO.setwarnings(False) to disable warnings.
  GPIO.setup(greenLed, GPIO.OUT)
/home/pi/Documents/rpiWebServer/pi-rfid/read.py:31: RuntimeWarning: This channel is already in use, continuing anyway.
Use GPIO.setwarnings(False) to disable warnings.
  GPIO.setup(redLed, GPIO.OUT)
/home/pi/.local/lib/python3.9/site-packages/mfrc522/MFRC522.py:151: RuntimeWarning: This channel is already in use, cont
inuing anyway.  Use GPIO.setwarnings(False) to disable warnings.
  GPIO.setup(pin_rst, GPIO.OUT)
619323204106
Paul
Access Granted

taking picture
Alerting your family you are here
Mail Sent
```





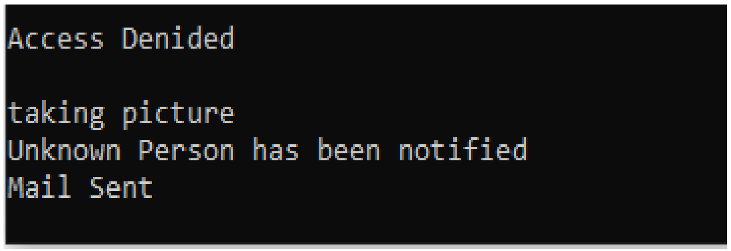
Paul Access Granted Inbox

leafgreen714@gmail.com
to me

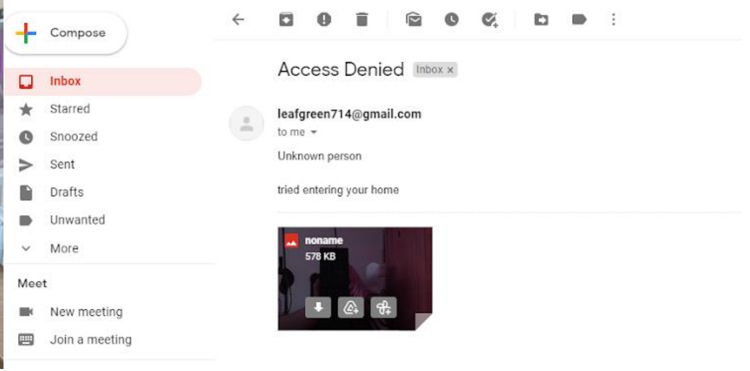
Your family member
Paul
has entered your home

If the key fob/card is registered, it will flash the green light. It also will give the output of “Access Granted” and send the mail to the owner through SMTP.

In case of non-registered key fob/card, the red light will blink three times, and it will alert the owner by taking the picture of the person trying to access the house/box and it will send the email to the owner/user.



```
Access Denided

taking picture
Unknown Person has been notified
Mail Sent
```



Access Denied Inbox ×

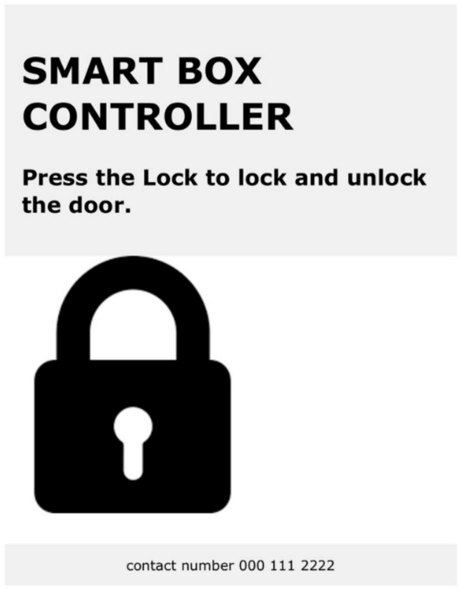
leafgreen714@gmail.com
to me

Unknown person

tried entering your home

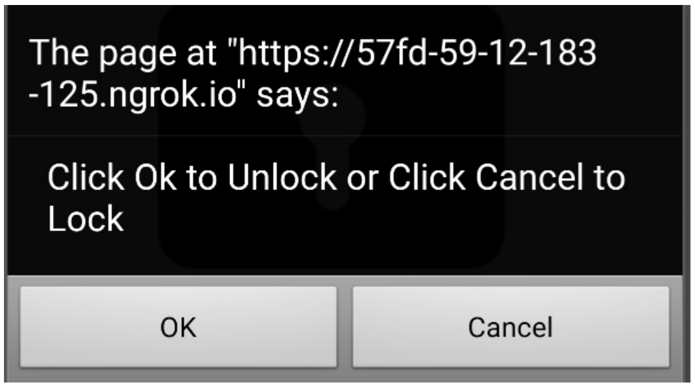
noname
578 KB

For remotely unlocking the door, we are using the MIT application.


SMART BOX CONTROLLER

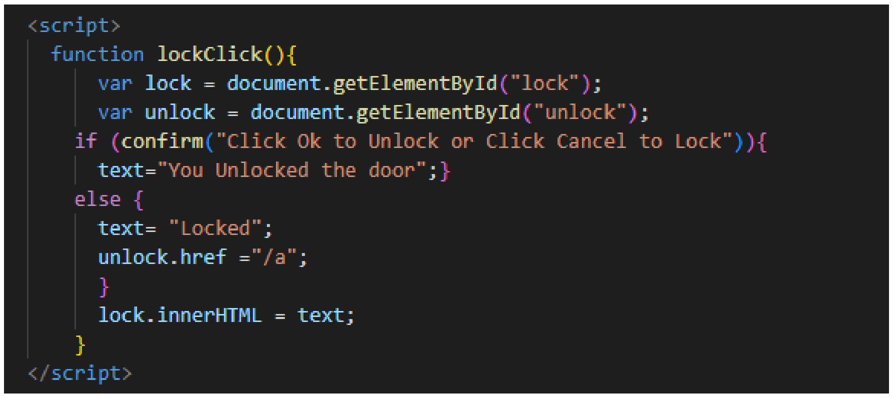
Press the Lock to lock and unlock the door.

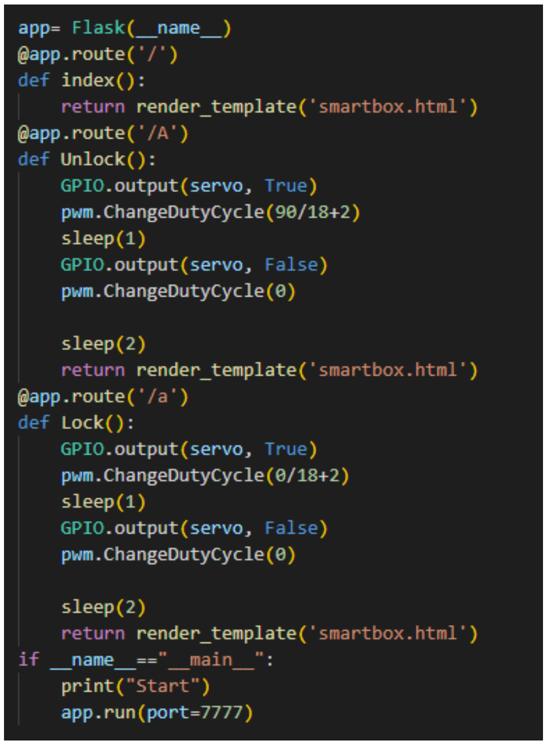
contact number 000 111 2222


The page at "https://57fd-59-12-183-125.ngrok.io" says:

Click Ok to Unlock or Click Cancel to Lock

When the client clicks on the lock, it will pop up a confirm window to let the user choose to unlock the door or lock the door. It will pass the value "/A" to unlock the door and "/a" to lock the door to the server.


```
<script>
  function lockClick(){
      var lock = document.getElementById("lock");
      var unlock = document.getElementById("unlock");
    if (confirm("Click Ok to Unlock or Click Cancel to Lock")){
      text="You Unlocked the door";}
    else {
      text= "Locked";
      unlock.href ="/a";
      }
      lock.innerHTML = text;
    }
</script>
```



```
app= Flask(__name__)
@app.route('/')
def index():
    return render_template('smartbox.html')
@app.route('/A')
def Unlock():
    GPIO.output(servo, True)
    pwm.ChangeDutyCycle(90/18+2)
    sleep(1)
    GPIO.output(servo, False)
    pwm.ChangeDutyCycle(0)

    sleep(2)
    return render_template('smartbox.html')
@app.route('/a')
def Lock():
    GPIO.output(servo, True)
    pwm.ChangeDutyCycle(0/18+2)
    sleep(1)
    GPIO.output(servo, False)
    pwm.ChangeDutyCycle(0)

    sleep(2)
    return render_template('smartbox.html')
if __name__=="__main__":
    print("Start")
    app.run(port=7777)
```

frequency = 50Hz

$$\text{Time Period} = \frac{1}{50} = 0.02\text{s}$$

$$= 20\text{ms}$$

duty cycle = 60%

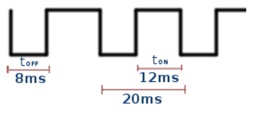
$$\frac{t_{ON}}{t_{ON} + t_{OFF}} = 0.6$$

$$\frac{t_{ON}}{20\text{ms}} = 0.6$$

$$t_{ON} = 0.6\text{x}20 = 12\text{ms}$$

$$t_{OFF} = 20\text{-}12 = 8\text{ms}$$

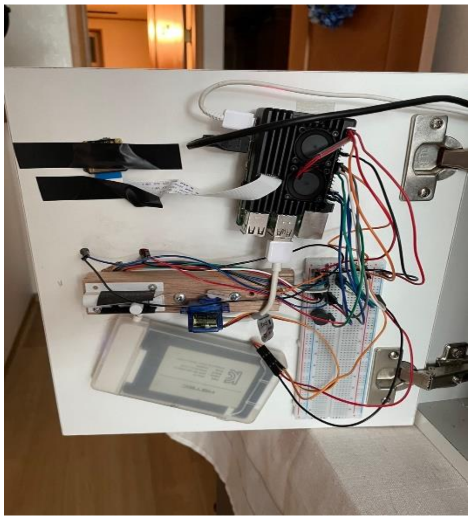
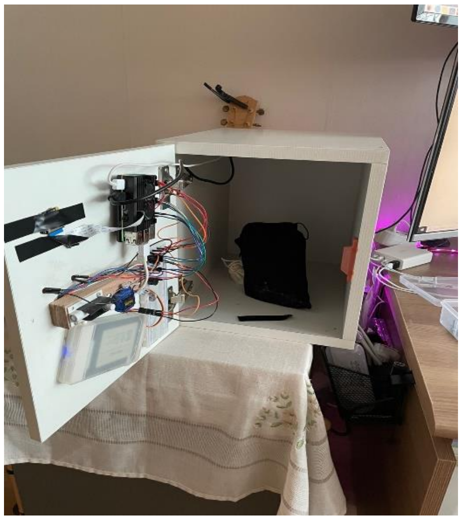
The resulting PWM wave will be :



app.route('/') is our default and the index will render the "smartbox.html". GPIO gives the servo pin number 2 to activate. As the servo needs pulses to activate, we set Pulse Width Module (PWM) to ChangeDutyCycle with a certain pulse in order to get a certain duty cycle which is the Time on over Time on + Time off Basically changing a duty cycle to a certain percentage to unlock the door.

Conclusion

Smart Box project initial results were presented. Which was about security Smart box that can be converted to any other locks such as doors and entrances. Our technology can lock and unlock the box through an RFID technology and capture images as long as you scan the RFID. It will alert the owner/user who try to access the box or who did access the box by sending them the picture and the NFC tag identification tag number through their email. If the owner/users spouse needs access to your box when you are at work you can simply access the box through our application, we have made to control the box mechanism.

References:

- Bahga, A. (2014, August 9). Internet of things: A hands-on approach. Google Books. Retrieved May 28, 2022, from https://books.google.com/books/about/Internet_of_Things_A_Hands_On_Approach.html?id=JPKGBAAAQBAJ

- Robles, R. J., Kim, T. H., Cook, D., & Das, S. (2010). A review on security in smart home development. International Journal of Advanced Science and Technology, 15.

- Bangali, J., & Shaligram, A. (2013). Design and Implementation of Security Systems for Smart Home based on GSM technology. International Journal of Smart Home, 7(6), 201-208.

- A. C. Jose and R. Malekian, "Improving Smart Home Security: Integrating Logical Sensing Into Smart Home," in IEEE Sensors Journal, vol. 17, no. 13, pp. 4269-4286, 1 July1, 2017, doi: 10.1109/JSEN.2017.2705045.

- Hayashi, V. T., Arakaki, R., & Ruggiero, W. V. (2020, December). OKIoT: Trade off analysis of smart speaker architecture on open knowledge IoT project. *Internet of Things*, 100310. https://doi.org/10.1016/j.iot.2020.100310

- Klements, M. (2020, February 7). Arduino Based RFID Door Lock – Make Your Own. The DIY Life. https://www.the-diy-life.com/arduino-based-rfid-door-lock-make-your-own/