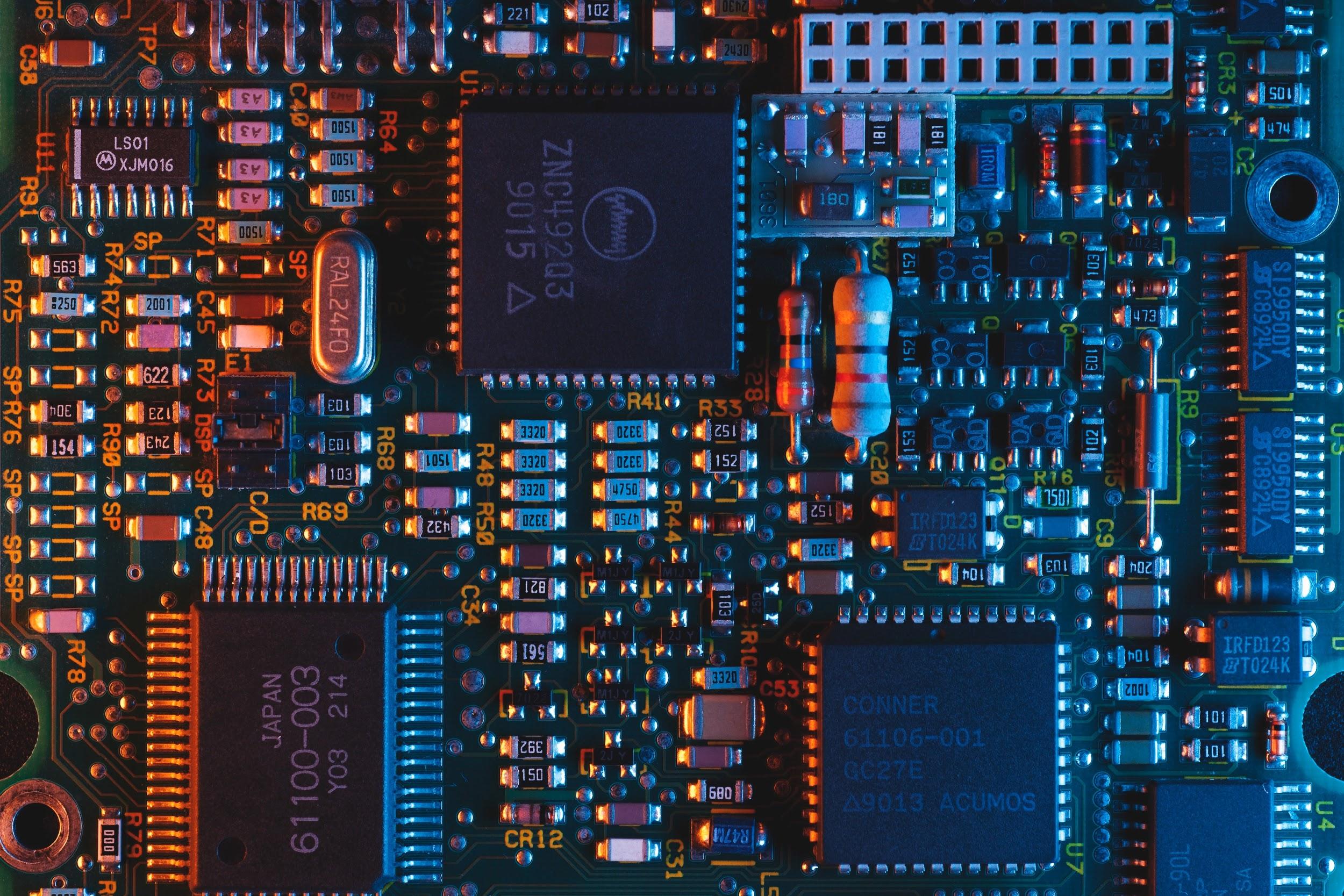
## horizontal line



FACE RECOGNITION SECURITY CAMERA SYSTEM

09.04.2020

**─**

Ly Huynh Chuong - N01250219

Vu Dang Khoa Khuong - N01228325

# **Table of Contents**

# I. Abstract …….……………………………………………………………………………………………………………..….1

# II. Introduction……...………………………………………………………………………………………………………....1

# III. Core content ………...……………………………………………………………………………………………………..2

1. Group Members
2. Bill of Material
3. Derivation & Explanation
4. Source Code
5. Schematics & Layouts
6. Block Diagrams

IV. Result, Reference & Supplementary

1. Result & Analysis
2. Project Management
3. References

**Figures**

Figure 1 -

Figure 2 -

Figure 3 -

# 

# 

# 

# 

# 

# 

# 

# I. Abstract

- This report presents the details of the project. This project is about the camera for the security system. The hardware of the system is built by a raspberry pi board and a night camera. The project is run by the python code which is built on the AWS platform to send the messages to the user. The system uses the AWS Rekognition API to detect the face of the user, after it is recognized, the LED will turn on and the image’s face will be captured, finally the image, messages will be sent to email by AWS SNS along with the captured image.

- Keyword: Raspberry pi, Night-vision camera, AWS Rekognition, AWS SNS.

# II. Introduction

- The idea of the project is from the face recognition system on smartphones, so we decided to build a device to recognize the face of the user. The device will be used for the security system of a house, room, etc. The tasks of the device will recognize the guest assigned to the system, and it will send the message with the image's face of the guest to the email of the owner. By the application of the device, we can develop it to work with smart equipment (smart lock, smart door, smart speaker, etc).

III. Core content

1. **Group Members**

|  |  |
| --- | --- |
| **- Khoa Khuong Vu Dang** | **- Face recognition** |
| **- Ly Huynh Chuong** | **- Sending the message to user** |

1. **Bill of Material**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Reference Designator** | **Description** | **Manufacturer** | **Source** | **Quantity** | **ship** | **Price** | **Total** |
| Raspberry Pi 3 Model B Board | Model B Board | Raspberry Pi | amazon.ca | **1** | **0** | 66.50 | 66.50 |
| Smraza Raspberry Pi 4 Cooling Fan, Pi 4 Fan | 4PCS Aluminum Heatsinks, DC 5V Fan for Raspberry Pi 4, Fan Compatible with Raspberry Pi 4 Model B, Pi 3 B+, Pi 3, Pi 2, Size: 30x30mm | Smraza | amazon.ca | **1** | **0** | 9.99 | 9.99 |
| Smraza Camera Module for Raspberry Pi | 5MP 1080p OV5647 Video Webcam Night Vision, Camera Compatible with Raspberry 4 Model B/Pi 3 2 Model B | Smraza | amazon.ca | **1** | **0** | 30.99 | 30.99 |
| Gigastone 32GB 2-Pack Micro SD Card | Adapter, U1 C10 Class 10, Full HD Available, Micro SDHC UHS-I Memory Card | Gigastone | amazon.ca | **1** | **0** | 16.98 | 16.98 |
| AWS Access (Free tier options available) |  |  |  |  |  | 0 | 0 |

Total = $124.46 cad

1. **Derivation & Explanation**

The project comprises 2 main parts: The hardware and software

1. *Hardware*

The Raspberry Pi board is connected with a NoIR camera module, which is used to interact with the “door lock relay” or LED in this case. Figure 12 shows the connection of the hardware in use.

1. *Software*

The software consists of 2 python scripts:

* ***index\_faces.py*** script is to register customer’s new faces by AWS Rekognition from my-faces-rpi bucket, in which contains the images of customers separated by folder’s name in figure 1. When the script is run, each image is indexed to a different ID but the index names correspond to the person to which the images relate. The Rekognition API uses artificial intelligence to “learn” which faces to compare with the data set indexed before.

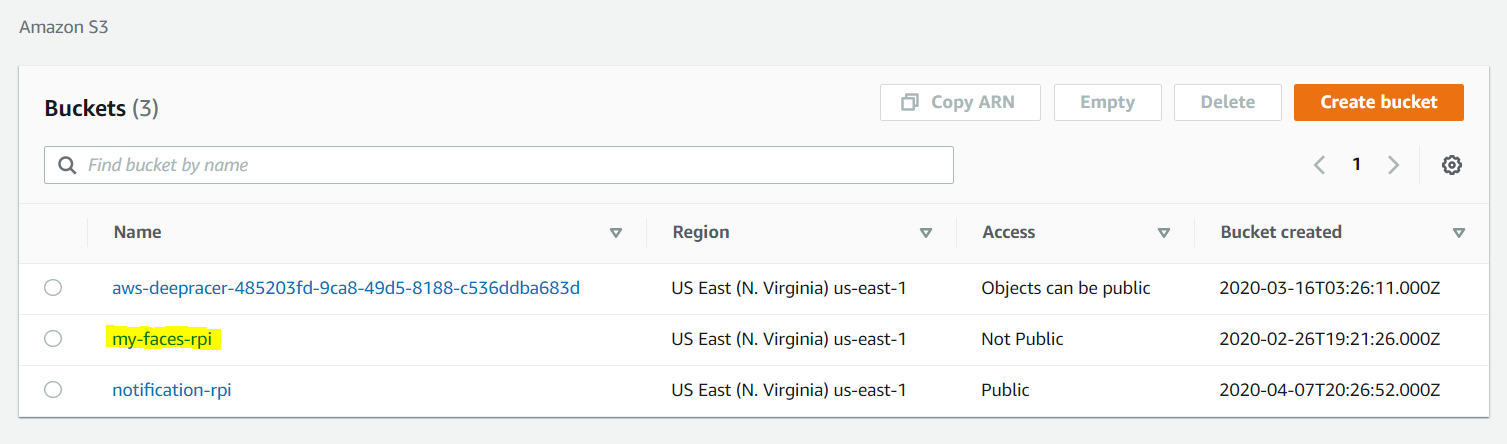


Figure 1 – AWS S3 Buckets

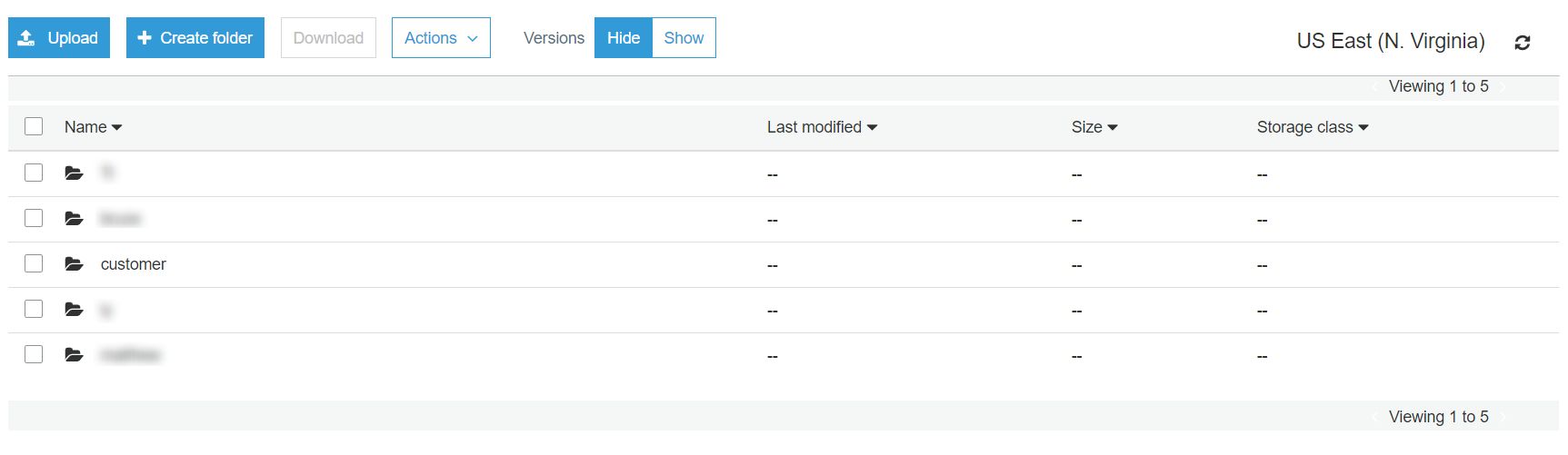


Figure 2 – Bucket my-faces-rpi

* To initiate the face detection sequence on the Raspberry Pi, the ***capstone.py*** script would be executed on start-up. This script would automate the process of face detection and email notification.

1. **Source Code**

**Index\_faces.py**

***Index\_faces.py*** is only used when we need to register any new faces for the system.



Figure 3 – Rekognition collection initializing

The script uses one of the S3 buckets for storing the faces’ images. The list\_response variable contains an array of collections taken from the ***list\_collection()*** method. Next, we delete any existing collection if it exists except for our collection by ***delete\_collection()*** method. Finally, a new collection is created by ***create\_collection()***.

We start indexing each face based on their users. Each image will have a different indexing label but the ID you want to assign to all the faces detected in the image depends on the name of the folder containing them, which results in matching different images to one person just by folder name.



Figure 4 – Rekognition indexing

**capstone.py**

The ***capstone.py*** script would automate the process of face recognition, storing snapshot to local storage and notification via email, of which will be run by the Raspberry Pi.

Figure 5 – Creating AWS Service Client

The 3 lines above create the all the AWS services clients that we are going to use



Figure 6 – Image to Bytes Conversion

In order for AWS Rekognition to detect faces that have been indexed before, the frame recorded is being converted into PIL image and then converted again to bytes, which is the appropriate argument for the ***search\_faces\_by\_image()*** method below.



Figure 7 – search\_faces\_by\_image() method

Based on the AWS document [3], we will scan faces through the indexed faces, if true then print out the face’s name, its confidence score and increment counter. The counter only increments every 100 frames if face is detected.



Figure 8 – Door lock Mechanism

If the counter counts to 10, which translates to 100 frames or 2.5 seconds in real life if frame per second is 40fps, call ***capture()*** function to take a snapshot and set variable pause to 1.

If pause is 1, we send a high signal to the Raspberry pi’s BCM pin 18 to turn the lock open. At the same time, pause\_counter increments to hold the signal on and the counter is reset to 0.

If the pause\_counter counts to 20 or 5 seconds, we send a low signal to the Raspberry pi’s BCM pin 18 to close the lock and reset pause and pause\_counter to 0.



Figure 9 – capture() function Source code

***capture()*** function starts with modifying i variable to global so i doesn’t get referenced before assignment. The snapshots are captured when ***capture()*** is called by ***imwrite()*** method and call ***upload()*** and ***email()*** functions after 1 second delay to wait for the upload to complete. Finally, i increments.

The snapshot is named after i so there is no duplicate is made both in local storage and on the server.



Figure 10 – upload() function Source code

***upload()*** function upload the snapshot by using ***upload\_file()*** method.

Based on the document (“SES.”), the ***email()*** function sends HTML-style email with the snapshot URL link from the S3 server to the customer’s email



Figure 11 – email() function Source code

1. **Schematics & Layouts**

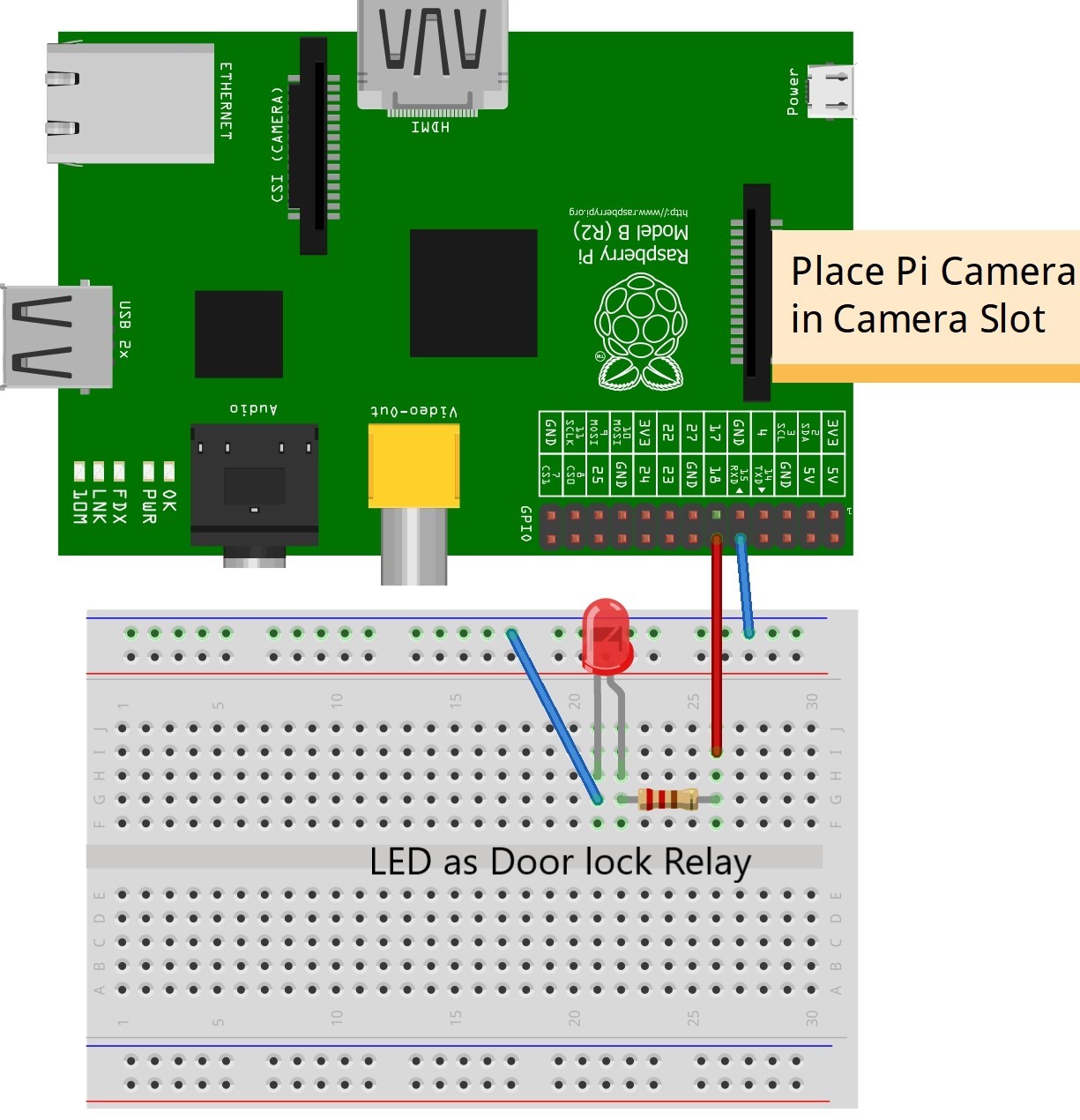
****

Figure 12 – Hardware Layout

1. **Block Diagrams**

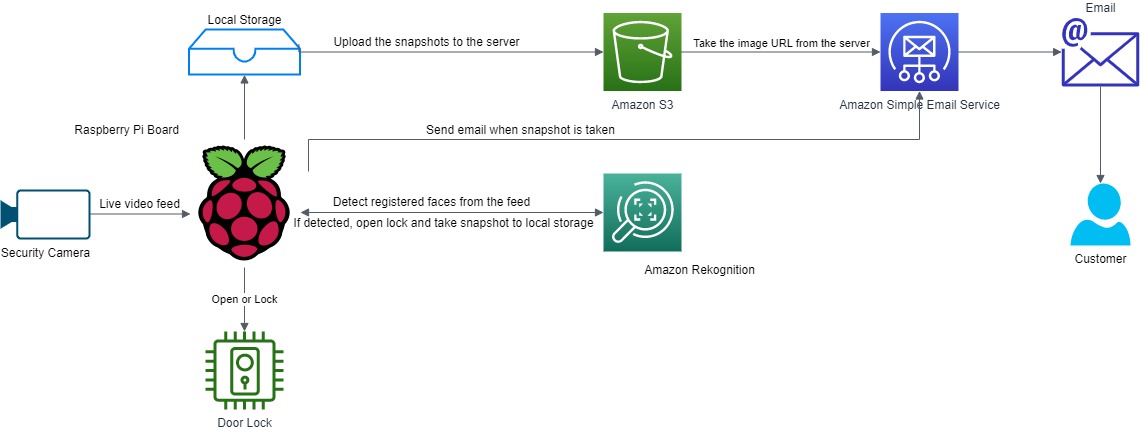
****

Figure 13 – System Architecture

# IV. Result, References & Supplementary

1. **Result and Analysis**

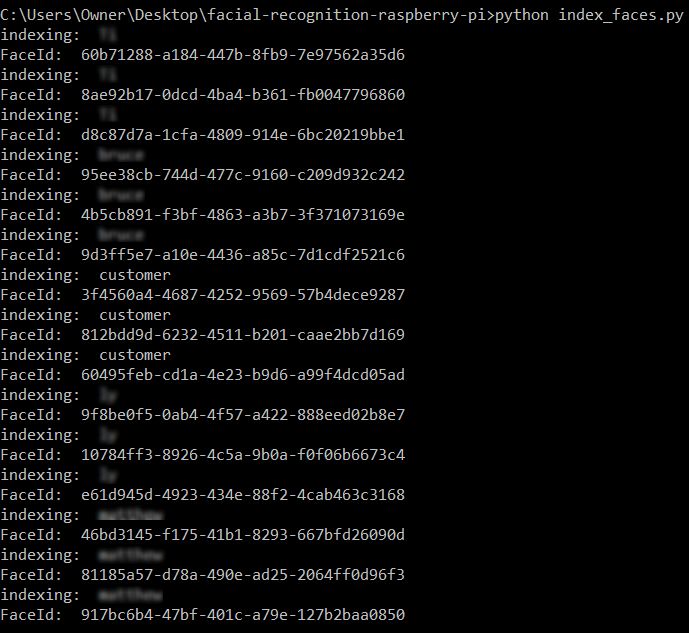


Figure 14 – index.py terminal screen

As expected, after running ***index\_faces.py,*** all the images in my-faces-rpi bucket have been registered with unique ID. The index values match up with different person, which correspond to the folder’s name.

When ***capstone.py*** is executed in the Raspberry Pi, there will two components appearing: The Object Detector window showed as a camera review screen and the terminal displaying the output. The output contains face detected and its confidence score; number of frame and the lock’s state.

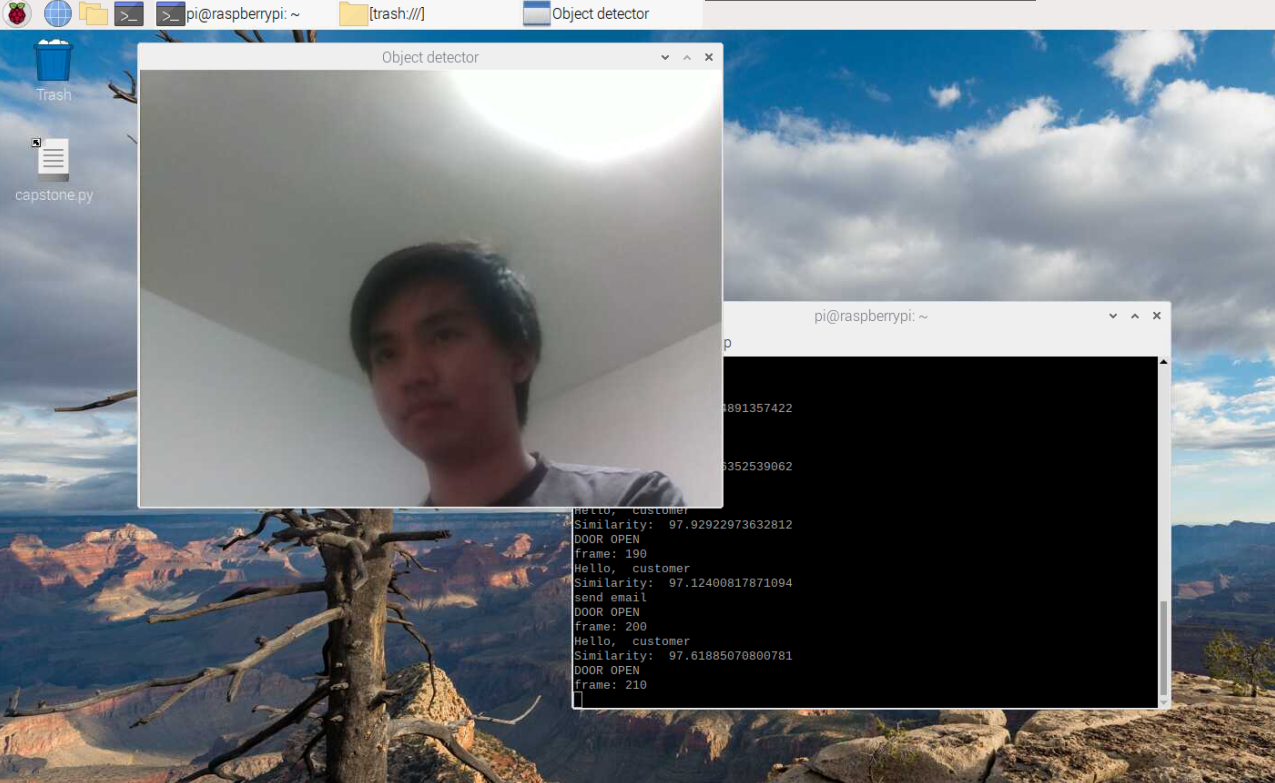


Figure 15 – Raspberry Pi

**Stage 1**

Because AWS Rekognition API is only called every certain frame defined in figure 6 as a way to reduce API call, the number of frame show the frame that is being analyzed. And when a registered face is detected, a string line “Hello + FaceName” is printed along with the confidence score of how much the API believe the face belong to one of our data. As the process continues as long as the face is still in frame, a counter is counting for the lock open. This procedure is made to assure the identity of the face. After this, the system will be going to stage 2.

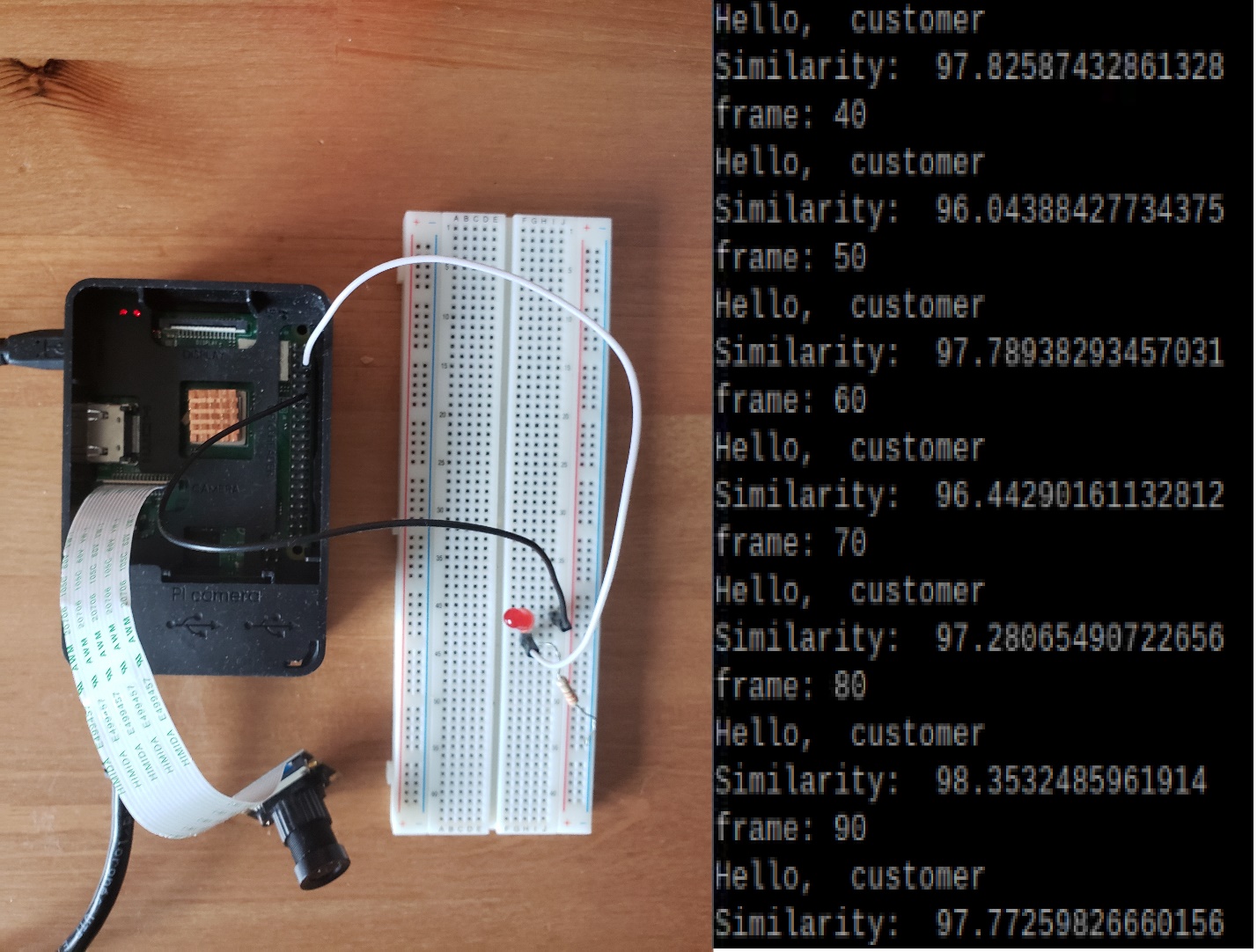


Figure 16 – Stage 1

**Stage 2**

After the internal counter’s limit is reached, the Raspberry Pi will send a notification to a registered email included in the script – figure 11 with a confirmation with the message ID printed in the output. At the same time, the LED will turn on by supplying a high signal to a GPIO pin, a string of “Door open” is printed to confirm the action. And at every number of frame this signal will be kept on while another counter is incrementing. The time period for which the door is unlocked is usually shorter than the time needed for opening but it is configurable. It is made to give the person a window of time to open the door.



Figure 17 – Stage 2

**Stage 3**

Even if no face is present, the door still remains open until the second counter reaches limit. After that, the Raspberry Pi will send a low signal to the GPIO pin to turn off the LED locking the door.

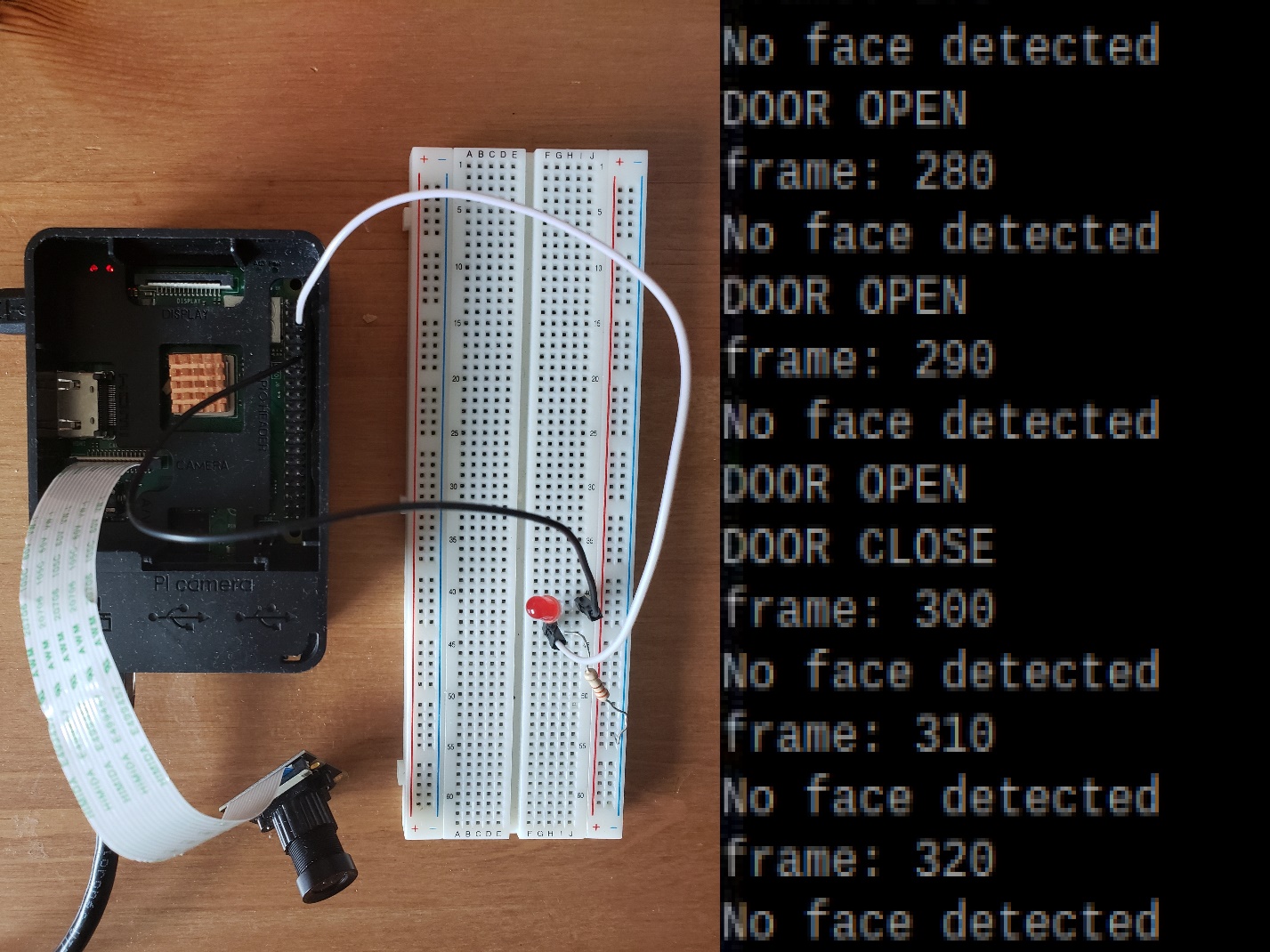


Figure 18 – Stage 3

1. **Project Management & Progress Update**

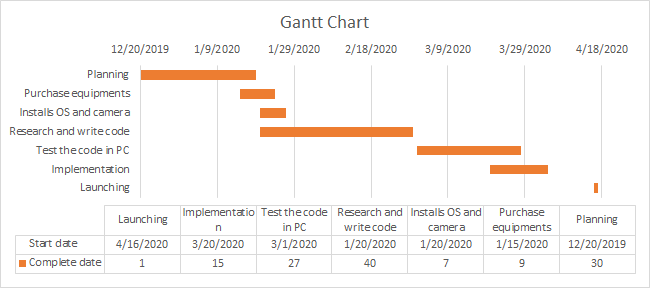
****

Figure 19 – Gantt Chart

1. **References**

“Rekognition.” *Rekognition - Boto 3 Docs 1.12.44 Documentation*, boto3.amazonaws.com/v1/documentation/api/latest/reference/services/rekognition.html.

“SES.” *SES - Boto 3 Docs 1.12.44 Documentation*, boto3.amazonaws.com/v1/documentation/api/latest/reference/services/ses.html.