Project Report

on

Smart Greetings and Security System

Due: 2022-05-15

CS 437: Internet of Things

Spring 2022

by

Matthew Chapman Caesar
Professor
Department of Computer Science
University of Illinois at Urbana-Champaign

Submitted By

Ayush Khanna
akhanna6@illinois.edu

Gazi Muhammad Samiul Hoque
ghoque2@illinois.edu

Jennifer Tsai
jtsai21@illinois.edu

Yunhan Guo
yunhang2@illinois.edu

GitHub Repo: https://github.com/samhq/cs437iot-final

Video: https://uofi.box.com/s/qq5l0aw8imhrtn1ng0cqr4zyp5753q4w

Contents

List of Figures

1 Motivation

If you're planning to make anything "smart" about your house or even when you are planning to go on a vacation, then home security is first. In 2020, the reported burglary rate in the U.S. is 314.2 per 100,000 of the population[1]. Even this already decreased by 75% compared with that in 1990, it's still frightening given its unpredictable outcome. If your family members are at home, their lives might be in danger. In any case, all your most precious valuables are there. Protecting human lives and the private property should be a top priority. A smart home security system helps you to identify suspicious strangers, so that you can be cautious and report to the police in advance. Also it helps to keep tabs on your house while you're away, from the convenience of your laptop or smartphone. With known and unknown faces captured in a database, the security cameras will identify your friends and family members and won't give an alarm. Otherwise, you'll get an instant alert, so you know that someone unidentified has hang around or even intruded into your property.

Traditionally, when you think of securing your house, you think of calling a home security provider who would send a technician to install a wired system and cameras in your home and enroll you in a professional monitoring service with bunch of wiring all over the place. It has several downsides. Firstly, it will not smartly detect danger. You will expect you are alerted only when suspicious people are detected (but of course, with high sensitivity), rather than watch the camera feed yourself for the whole day. Besides, the maintenance cost of such devices is also high, and you always have to rely on security providers to take care of it. With the advent of smart technology as IoT, you can have a professional monitor, getting real-time updates and notifications on your smart device. In terms of cost and feasibility, you can now even build and install the smart system on your own, fix the bugs and even further extend its functionality for customized needs.

We are looking to build a smarter and more advanced home security system. Besides providing live video, it has two important extensions. First, with the facial recognition part, it can smartly detect whether there is a known face or a totally new visitor. Instead of passively monitoring there and doing nothing more, it can send greetings to our friends and families, and more importantly, capture a photo of an unknown person, and notify you via an email, so you can quickly know in case there's potential danger. Many people would argue why not have a biometric recognition instead of a facial recognition, which is based on unique behavioral and physiological traits, which includes fingerprints, keystrokes dynamics, etc. Facial recognition appears to be a favorite due to its accuracy, ease of use, and no-contact sensing. Second, our system provides a user-friendly and powerful management console, where you can watch the live video, re-train the facial recognition model, and list the visitor history, so that you can easily manage this system and even further extend features as you want.

2 Technical Approach

2.1 Overall Architecture and Data Flow

Our project mainly consists of two components: the Pi that captures images and greets people, and the management console that supports multiple features like re-training models, managing images, watching live video and viewing history of visitors.

The Pi, attached with a motion sensor, a camera and a speaker, will be installed outside of our front door. The motion sensor keeps monitoring human motions. Whenever it **detects** anyone

[1] https://www.statista.com/statistics/191243/reported-burglary-rate-in-the-us-since-1990/

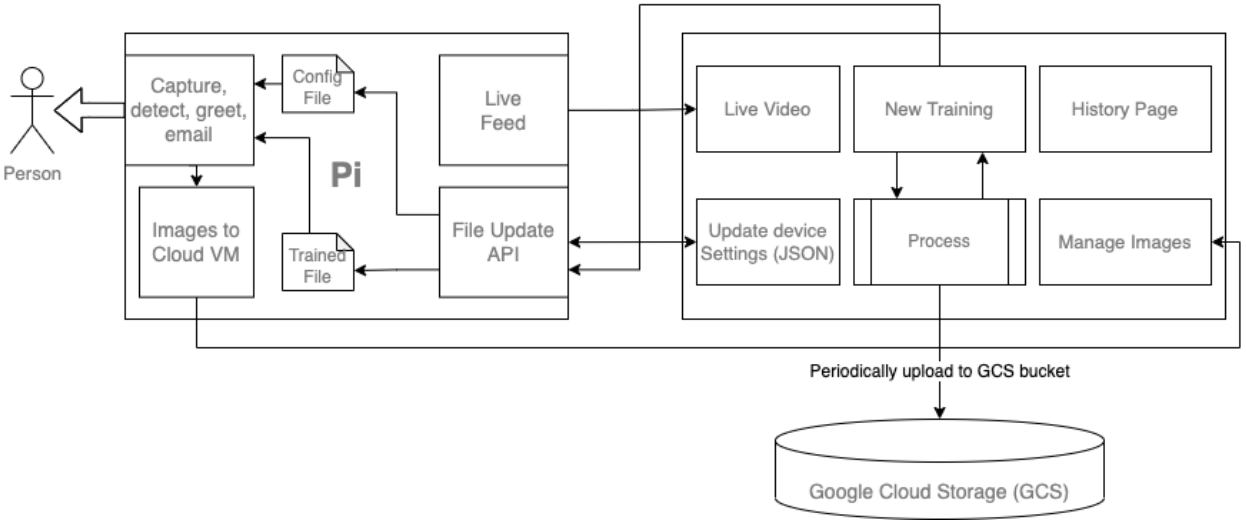

Figure 1: *Overall Architecture and Data Flow*

comes to the door (in a certain distance of the door), the camera will be triggered to **capture** a couple of images of that person. Then it turns to its **trained file** to see this person can be matched by any stored images. If the person's face is recognized, then a **greeting** message will be spoken to the person. If the person is not recognized, then there will be no greetings, but the captured images will be stored and uploaded to a virtual machine. In both cases, the Pi will send an **email** to us with the person's photo. Another feature the Pi supports is the **live feed** depending on our needs.

The management console supports several features:

Live Feed Instead of waiting for the image capturing to be triggered, you may want to watch a live feed, just like a home-security camera. We can send requests to the Pi and it will respond with the live feed to be displayed on the front-end.

Re-train If we want to add faces of our families or friends to our model, We can re-train the model. Specifically, we extract images from the VM, rename these images to the person's name, and re-train the model locally. When the whole training process is completed, we delete the images locally and synchronize the new version of the model to the Pi which will be used for future recognition.

History Page To view the known and unknown visitors, we shall be able to see a history by extracting them from the VM, with timestamps and other properties.

Update Device Settings We can perform basic maintenance on the Pi if required, by creating a json file to update the original settings in the Pi.

Manage images We can extract the images from the VM, sort and annotate the images for different persons that are unrecognized previously, and then send the them back to the VM.

The VM provides storage for all of the images. Both the Pi and our management console interact with the VM to do operations with the images. The Pi is mainly sending images to the VM, and the management console can access images for viewing, training, managing, etc.

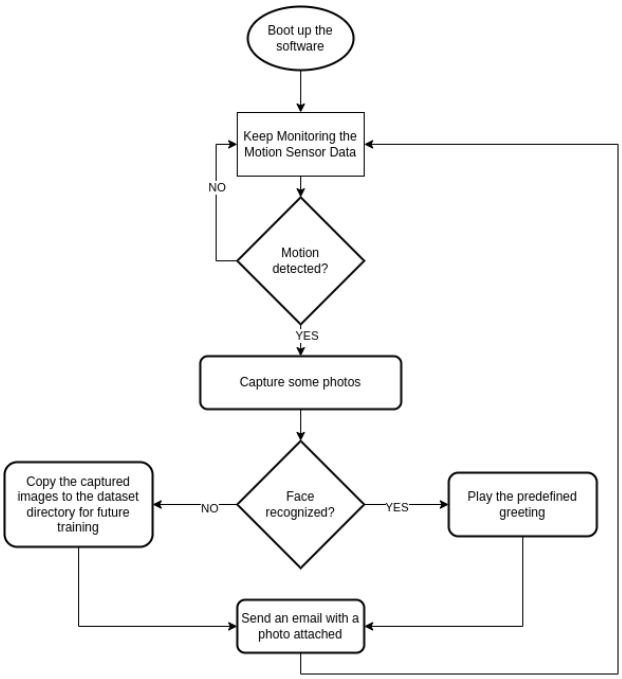

Figure 2: *Sample Flow Chart*

2.2 Physical Device Creation and Circuits

We integrated the Motion sensor and the camera with the raspberry pi, something like the following:

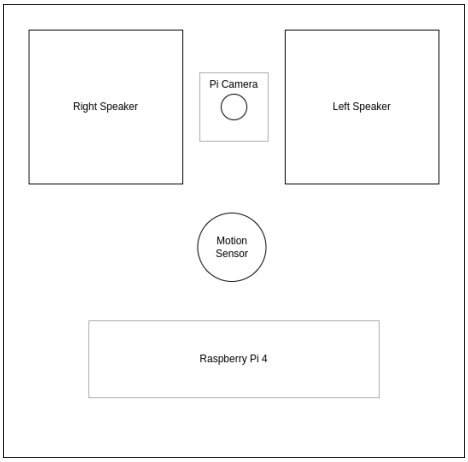

Figure 3: *Potential Structure of the Final Device*

The circuit

To assembly it, we configure the circuits as follows:

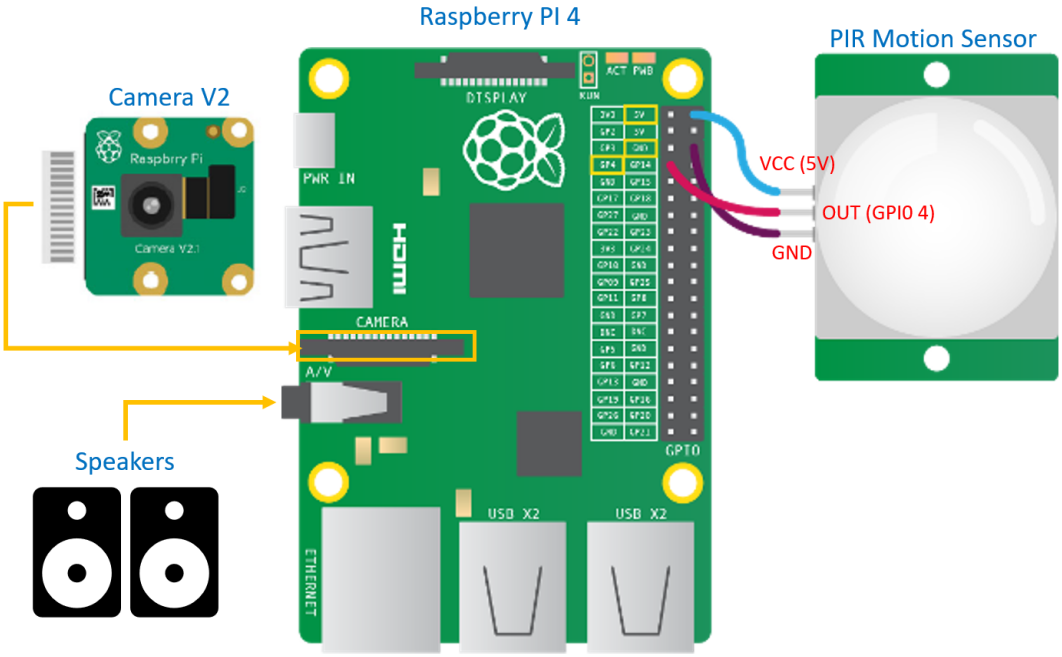

Figure 4: *Circuits of the Pi*

For the PIR motion sensor, the VCC port is connected to a 5V power port; the Out port is connected to GPIO 4; the GND port is connected to Ground. The speakers are connected to the Audio port. Here we use a wired connection instead of a wireless one for the sake of network stability. The camera is connected to the Camera port.

3 Implementation Details

3.1 The Pi Part

3.1.1 Physical Components

We will need the following components to develop the device:

- Raspberry Pi 4 with Raspbian Buster
- Raspberry Pi Camera Module V2 - 8 Megapixel, 1080p for capturing images and providing live video
- PIR Motion Sensor for detecting human motions [2]
- Raspberry Pi Power Adapter
- Small Speakers for the greeting feature
- PVC Boards for making the device structure

[2]For example, see https://store.roboticsbd.com/sensors/23-pir-motion-sensor.html

3.1.2 Motion Monitoring

The motion sensor is the trigger of all of our following steps. For this step, we use the package **`import MotionSensor from gpiozero`** to call its functions. We use the function **`wait_for_motion()`** to monitor motions - it keeps monitoring motions and will proceed to the next step if any motion is detected.

3.1.3 Photos Capturing and Faces Recognization

With motion detected, we capture a photo with our camera. Here we use the package **`from imutils.video import VideoStream`** to grab a frame from the video stream. Then we try to recognize the face with the package **`face_recognition`**. If it matches any person(stored encoding) that we already know, then we proceed to the greeting step. It none of our storage matches it, we will proceed to send an e-mail.

3.1.4 Greeting

This greeting function is implemented with the **`espeak`** package. When a known face is identified, the speaker will greet this person with his/her name. If a stranger is coming, it will speak out a warning.

3.1.5 Email Sending

When a stranger is detected, an email will be sent with the image as an attachment. We use **mailgun's python API** to achieve this. After signing up an account, mailgun will provide a sandbox url and keys, so that we can programmatically send e-mails to our target.

```python
import requests

key = 'Your Key'
sandbox = 'Your Sandbox Url'
recipient = 'Your Target E-mail Address'

request_url = 'https://api.mailgun.net/v2/{0}/messages'.format(sandbox)
request = requests.post(request_url, auth=('api', key), data={
    'from': 'hello@example.com',
    'to': recipient,
    'subject': 'Hello',
    'text': 'Hello from Mailgun'
})

print('Status: {0}'.format(request.status_code))
print('Body:   {0}'.format(request.text))
```

Figure 5: *Mailgun Python API*

3.2 The Management Console

3.2.1 Front End

The front end design has four tabs **Home**, **History**, **Settings** and **Live Feed**. The **Home** page lists all the devices. We can also add new devices here. At the **History** page, we can see all the images with an identified name(or "unknown") and the timestamp. Also, we have an edit option to change the name. The **Settings** is where we can update the configuration file in json format. The **Live Feed** page can support live feed.

3.2.2 Back End

We use Rest for the back end, to achieve the functions to configure the device settings and manipulate images. One file to work on is **`devices.json`** for some useful information of all the devices that use this system, including the url to post the live feed, and the url for API. The other is **`settings.json`** for settings of all the devices including the name, the target email address, the mailgun API and the sender email. We also provide sample files **`sample_devices.json`** and **`sample_settings.json`** in case there are no existing versions.

We provide APIs to configure the settings(by configuring these two json files) as well as to manipulate images:

Get Devices We created a new device into **`devices.json`** and use **`sample_settings.json`** as the default setting.

Delete Devices We Delete the device from **`devices.json`**.

Get Settings We access and return the **`sample_devices.json`** of a particular device.

Post Settings We update a device's **`sample_devices.json`**.

Get Image We dive into the image folder to access and return the specified image.

Train Image We download all images for a device in a local folder, perform training, saving the **`encoding.pickle`** to the VM storage, and then download the the new **`encodings.pickle`** file.

Rename Image When we want to add a new person into our system, we will need to rename some images that were marked as "unknown". We just find the image in the image folder and rename it.

Get History We access and return all images in the specified decive's folder.

4 Results

Finally, our Pi is successfully equipped. The components, the camera, the speaker, and especially the PIR Motion Sensor are correctly connected to the Pi. Correspondingly, they are all able to function properly. The motion sensor can detect human moves and then trigger the cameras to capture images, then greetings are sent through the speaker.

Figure 6: *Image of the full system*

Our notification function is also working well. When the visitor detected is not matched with known faces in our storage, it immediately sends an email to our target, attached with a photo of this visitor, so we can get that warning in time to prevent danger.

Figure 7: *Emails for Unknown Visitors*

Our management console provides better user experience. We provide a **home** page to list all of the devices, where we can also add new devices. The **history** page lists all the visitors, where we can easily change the name of the image for the training purpose. The **Settings** Page support the features of re-training models and updating the configuration. The **Live Feed** page supports live monitoring through the camera.

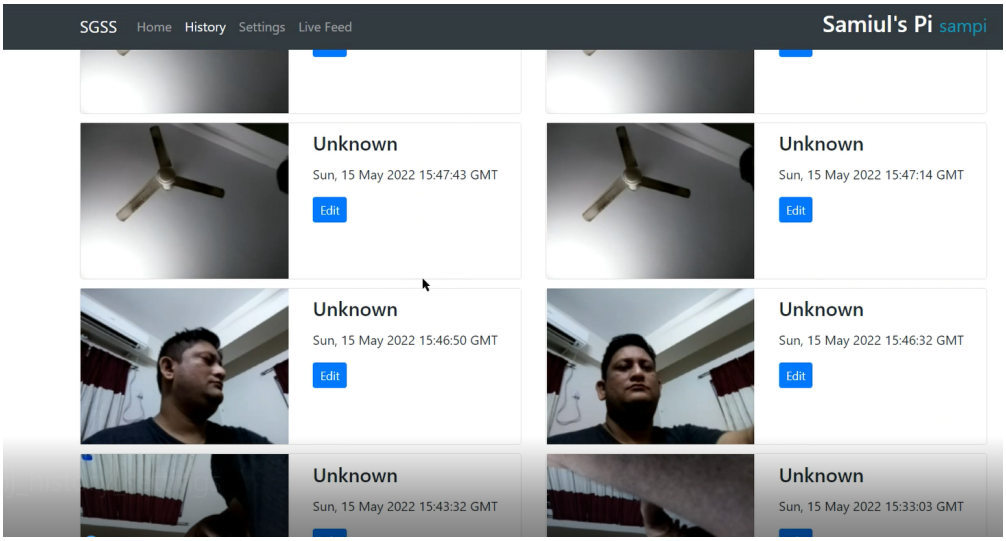

Figure 8: *Management Console*

The key takeaway is that, our project designs a home security system that's further extended

on the traditional security camera. We leverage the facial recognition algorithm to smartly detect unknown people and notify us immediately. Also, we provide a user-friendly management console that helps us manage all the devices, all the images, the facial recognition model, the video live feed and supports customized configuration.

5 Challenges

The biggest challenge is the interactions among the Pi, the management console and the cloud VM. We use the Pi for photos capturing, motion detection and greeting, but it doesn't provide enough storage, and is not capable of doing large-scale model training. To handle this, we have to turn to the Cloud VM and Storage. Then we design a management console for users. That makes it much more complicated compared to what we did in Lab2. Instead of simply getting the video feed and display on the webpage, we designed the data flow as shown in Figure 1, and developed different APIs for easier interactions.

Another challenge we faced was with Raspberry Pi Camera, we needed to take picture when a motion is detected and also needed to keep the Live Video Feed running using python Threading. However, using one single Camera module you cannot do both tasks in parallel even with the use of threading. You will get resource overflow error on camera module. To overcome this problem, we created trigger based threading which was invoked inside the face detector program to stop the video feed for a sec and quickly take the picture of a visitor when a motion is detected, then restart the Live video feed with the use of threading. Similarly, in the Video feed program, we called motion detector program using threading to enable the program when a motion is detected.

6 Citations

- Tutorial on PIR Motion Sensor: https://projects.raspberrypi.org/en/projects/physical-computing/11
- Tutorial on programming with speakers: https://www.dexterindustries.com/howto/make-your-raspberry-pi-speak/
- Tutorial on facial recognition with Pi: https://www.tomshardware.com/how-to/raspberry-pi-facial-recognition
- Mailgun API Guide: https://documentation.mailgun.com/en/latest/quickstart-sending.html

7 Group Members

- Ayush Khanna (Net ID: **akhanna6**)
- Gazi Muhammad Samiul Hoque (Net ID: **ghoque2**)
- Jennifer Tsai (Net ID: **jtsai21**)
- Yunhan Guo (Net ID: **yunhang2**)

A Demonstration Video

A.1 Video Link

You can find the demonstration video here:
https://uofi.box.com/s/qq5l0aw8imhrtn1ng0cqr4zyp5753q4w

A.2 Video Table of Contents

- `0:05`: Part 1 - Building the device
- `1:32`: Part 2 - Application Process
- `3:29`: Part 3 - Demo on Unknown People
- `3:50`: Part 3 - Demo on Known People
- `4:10`: Part 4 - Management Console Demo
- `7:15`: Part 5 - Code walk-through