**ELEC3542 Final Report**

Name: Lam Ho Kwan

UID: 3035377452

Project Title: My Dream House

# Introduction

This project is about a smart home device that can make peoples’ life more convenient. Its main function is to detect the face in front of TV and recognise faces to turn on or process the user-defined actions. It consists of three main features: face recognition, Siri control system and the temperature monitoring system.

# System Architecture

Raspberry Pi Number 1 (attached with Camera): Take pictures and recognise faces

Raspberry Pi Number 2 (Main device): Receive output from other devices and initiate the system

Raspberry Pi Number 3 (Command device): Receive command from Siri or application

Raspberry Pi Number 4 (weather monitoring device): Monitor temperature and humidity and send it to Main device

All the connection will use MQTT.

Username of recognised face

Eg. A

Captured pictures

camera

RPi2

RPi1

Temperature and humidity

Turn to A’s favourite channel

RPi4

Voice command

RPi3

Instruction

Phone

TV

**Orange arrows represents turn on device when system is on**

# Features

## Face Recognition

At the system bootup, the camera is turned on and automatically capture pictures in every 5 seconds. The reason why I choose pictures in JPG format rather than a video is because a video would consume more space and the frame rate would affect the accuracy of the algorithm. Also, video is much more difficult to apply into the face recognition part and it takes longer time to process. Therefore, with the concern of space and time, image in JPG would be more preferable. The reason why I choose JPG to be the file format is because JPG file size is relative small as its compression ratio can be very large. I use 40% by default and it can still recognise peoples’ faces. In my project, it is around 50KB per each. For saving space, when the program is running, it would automatically remove some useless pic like the pictures without faces and the processed one. For 10 loops of my program (10 pictures are taken), 5 would be deleted after processing so as to prevent overloading the file system.

After taken the pictures, it would save in Raspberry Pi number 1 (Camera device) and process the pictures to check whether there are faces or not. If face(s) is/are detected, it would further process the image to find whether it has match in the database which stores the users’ input pictures. If user(s) is/are detected, it would send the username(s) to Raspberry Pi number 2 (Main device) and Main device would fetch data from its datafile then send instructions to command device. MQTT is used in communication as it has a secured port for transferring message and it can send to all other Pi/other devices at a time so the main device won’t need to send the instruction one by one to make the system more responsive and faster.

The programming language I used is mainly python as it has a library called “face\_recognition” (credit to Adam). It claims to have an accuracy of 99.38% on the Labeled Faces in the Wild benchmark. The functions are mainly listed in the PDF. I had used the following functions (full code detail would in github):

Python commands (reference to PDF):

* face\_recognition.load\_image\_file(image,mode=”RGB”) to load image file from system into a numpy array(almost all file format are compatible like TIFF,PNG,JPG etc)
  + image – picture path or filename to be loaded
  + mode – format of picture (RGB=8-bit RGB, 3 channels / L=Black and White)
  + return a numpy array represents the image
* face\_recognition.face\_locations(image, n=1,model=’hog’) to locate face location in image
  + image – image in numpy array
  + number\_of\_times\_to\_upsample – higher number would find smaller face. I use default (1) in my program
  + model – can be “hog” or “cnn”. “hog” is less accurate but faster on CPU while cnn is a deep learning model with higher accuracy. I use the default one (hog) in my system as I want to reduce the processing time
  + return a list of tuples of faces in order of (top, right, bottom, left)
* face\_recognition.face\_encodings(known\_img, known\_face\_locations=None, num\_jitters=1) to encode image to bytes
  + image – pictures with face(s) in numpy array format
  + known\_face\_location – face(s) location in the image
  + num\_jitters – how many times face(s) is/are re-sample to encoding. I use the default value (1). Larger number would be slower but more accurate
  + return list of 128-dimentional face encodings
* face\_recognition.compare\_faces(known\_faces, unknown\_face\_encoding, tolerance =0.6) compare two image encoding and return list of true or false.
  + Known\_face – list of known face encodings
  + unknown\_face\_encoding – the coding of unknown image
  + tolerance – match if “image distance” is smaller than tolerance. I use 0.2 in my program as it would be the most accurate in my system. Default is 0.6.

Shell commands:

* face\_recognition --cpus -1 --tolerance 0.2 --show-distance true ./known\_face ./capture\_pic/image to encode all the image in ./known\_face and compare one by one with ./capture\_pic/image.
  + Each of comparison will generate a “image distance” (can be seen as accuracy, more accurate if it closes to 0).
  + Flag *tolerance* is to define “match” of two images like in Python. In this case, image distance with fewer than 0.2 is considered to be match.
  + Flag *cpus* define the use of CPU cores in the device. In above case, all cores are used to process the operation.
  + Flag *show-distance* true if the “difference” of two images are shown. Default is false (not show)

## Siri Control

It runs in Raspberry Pi number 3 (command device) which would be turned on when system is on.

My main algorithm take reference to Sanjeet (2017). It first requires user to configure his/her iPhone. Users should create a gmail account for public use and enable the IMAP and for the iOS devices, link Notes to that account and set it to upload the Notes in real-time. So, when users are going to instruct the system, he/she should say “take note” before the command and it would be saved on both gmail accounts and the user devices.

For the main program (siricontrol.py), it fetches latest note from gmail account using IMAP and process command using pre-defined modules to handle different command (detail in github).

I use this method because it is easy to manage. Python enables the IMAP transfer given a username and password. It could be risky if the gmail account is known. Therefore, the password can be changed in a period of time automatically. The main body of my siricontrol takes reference to Sanjeet with some further modifications like the handling of the command. The modules are all defined by myself to fit in this project.

For each modules, it would have the followings:

* moduleName – the name you want to call the module
* commandWords – list of command words. Command would be executed if all the voice-recognised command matches *all* the command words
* execution part – what exactly I want to do in the modules

I have created several modules for normal daily use of TV like turn on/off, volume up/down and changing channel/input. One extra function is that the weather (temperature and humidity) would be known and give response to user when user ask for it.

## Weather Monitoring

It is using sense hat to collect temperature and humidity data from the environment and it would send to Main device (if there is weather checking command) through MQTT. Also, the temperature would be shown on the LED output of sense hat. It would publish the data all the time when it starts.

# Challenges

It is a challenging project as the algorithm of the programs are quite new to me. The preparation work is tough too as before I use the face\_recognition library in python, I would need to download a lot of packages like dlib or opencv. These packages setup is time consuming as it takes long time to download and the memory cost is high. I would need to expand file system and allocate the GPU memory first. The camera would be malfunction sometimes as the resource is not well allocated.

Also, face recognition part is quite time-consuming as the data processing is complex and the Pi has limited resource. The solution is I can use up all the cores of the Camera device and it greatly reduce the process time by 20%.

However, this project helps me to learn something new and some useful that I can apply to my home and can actually make my life more convenient or contributes to the society. I am quite satisfied with myself in this course (although I am not the best performing student)

# Future Development

For the future development, I would expand my system to other IoT devices like lighting system and security devices. Also, my prototype would expand to try the real TV control part and an application can be developed for users to instruct the command directly to the command device and to upload photos, provides more system configurations (eg. Notifications, scheduled bootup, etc) Also, with higher computation power of latest Raspberry Pi model, the system can be more responsive and faster to process the command and fasten the communication process.

# Reference

Adam Geitgey (2018-). The world's simplest facial recognition api for Python and the command line. Link as <https://github.com/ageitgey/face_recognition> and <https://media.readthedocs.org/pdf/face-recognition/latest/face-recognition.pdf>

Sanjeet-theraspberryguy (2017) Control anything with Siri voice commands. Link as <https://github.com/theraspberryguy/SiriControl-System>

SANJEET (2017). SiriControl: A Voice Control System <https://thereallycoolstuff.wordpress.com/2017/04/29/siricontrol-a-voice-control-system/>