ENG5220

Real-Time Embedded Programming

Project Report

Team 16: Fun Security System

Bin Liu	2522070L
Shuaiqi Liu	2629952L
Genyuan Su	2534443S

1. Introduction

People often have the need of installing an effective security system to protect their private spaces and to grant access to themselves and their family or friends. Following this clue, our team thought about developing a raspberry-pi-based real-time security system and using face recognition to control the opening of the door lock. Face recognition is a quick and accurate way to distinguish between authorized hosts and unwelcomed people. This report will first demonstrate few points should be considered before we start our program, including problems may encountered, how our system is supposed to work and what hardware we should prepare.

2. Problems May Encountered

- How to ensure the performance of camera in abnormal circumstance (e.g., too light or too dark, camera angle);
- How to upload and save face data to database easily and quickly. The form of face data saved;
- What methods could be applied to matching the scanned face and face data.

3. How This System Works

When the whole system is activated, all the events start, which include face recognition, motor, LED strip, sound detection, and LED bulb. The camera for face recognition is always activated while the LED bulb is on only when a sound is detected (somebody approaching the door). The purpose of this function is to make sure the system can work properly in a dark environment. If the face captured by the camera can match any faces in the pre-created facial recognition library, a motor connected to the door lock will unlock the door and the LED strip installed inside the house will turn green. Otherwise, the motor will not operate, and the door will stay locked. If someone comes and tries to unlock the door but fails, the LED strip will turn red, which alarms the host in the house that an unauthorised visitor is outside.

4. System Flow Chart

There are two modes, which are automatic mode and manual mode, respectively. The flow charts of the system are shown in Figure 1.

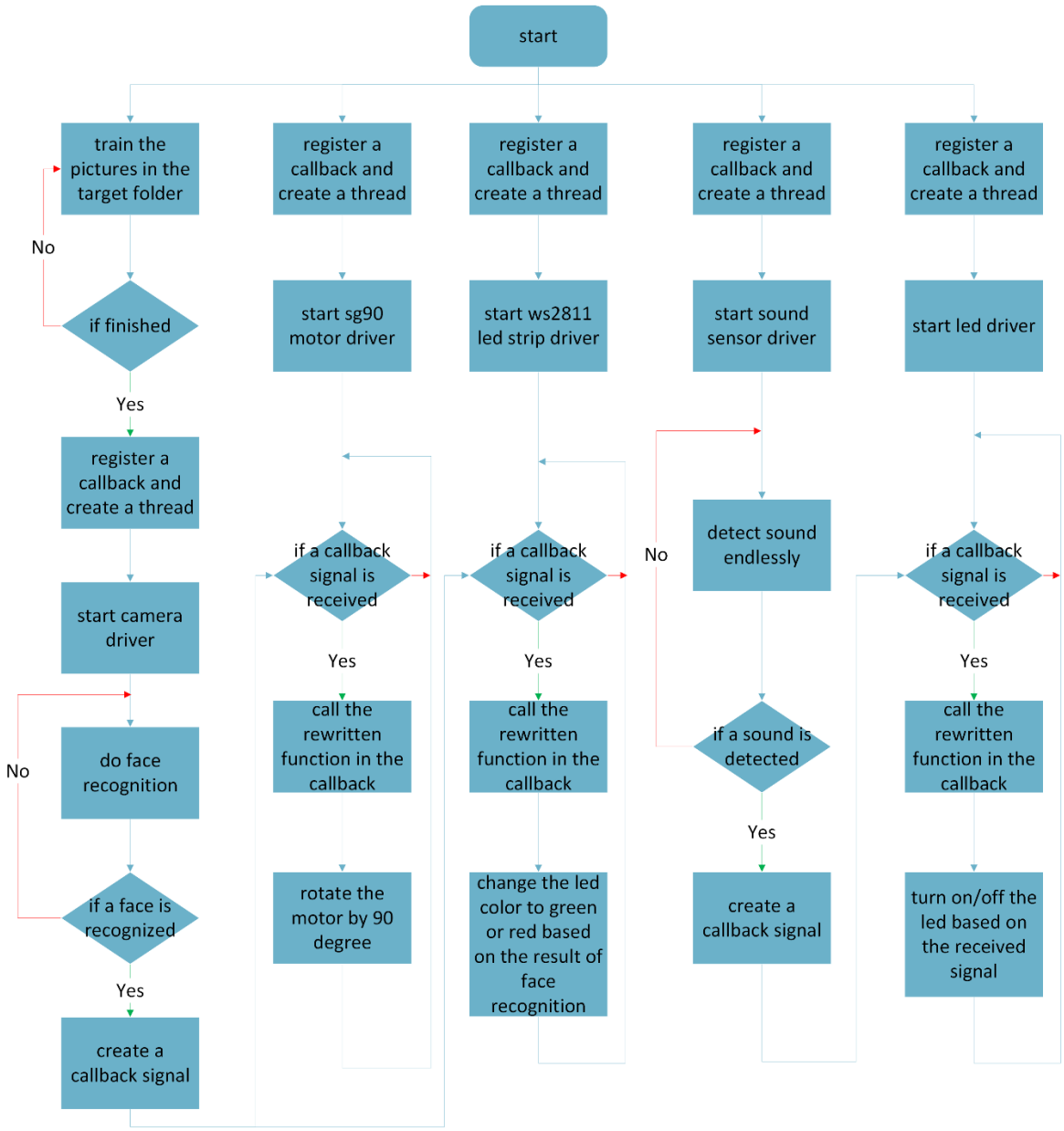


Figure 1 (a). The flow chart of the Automatic Mode

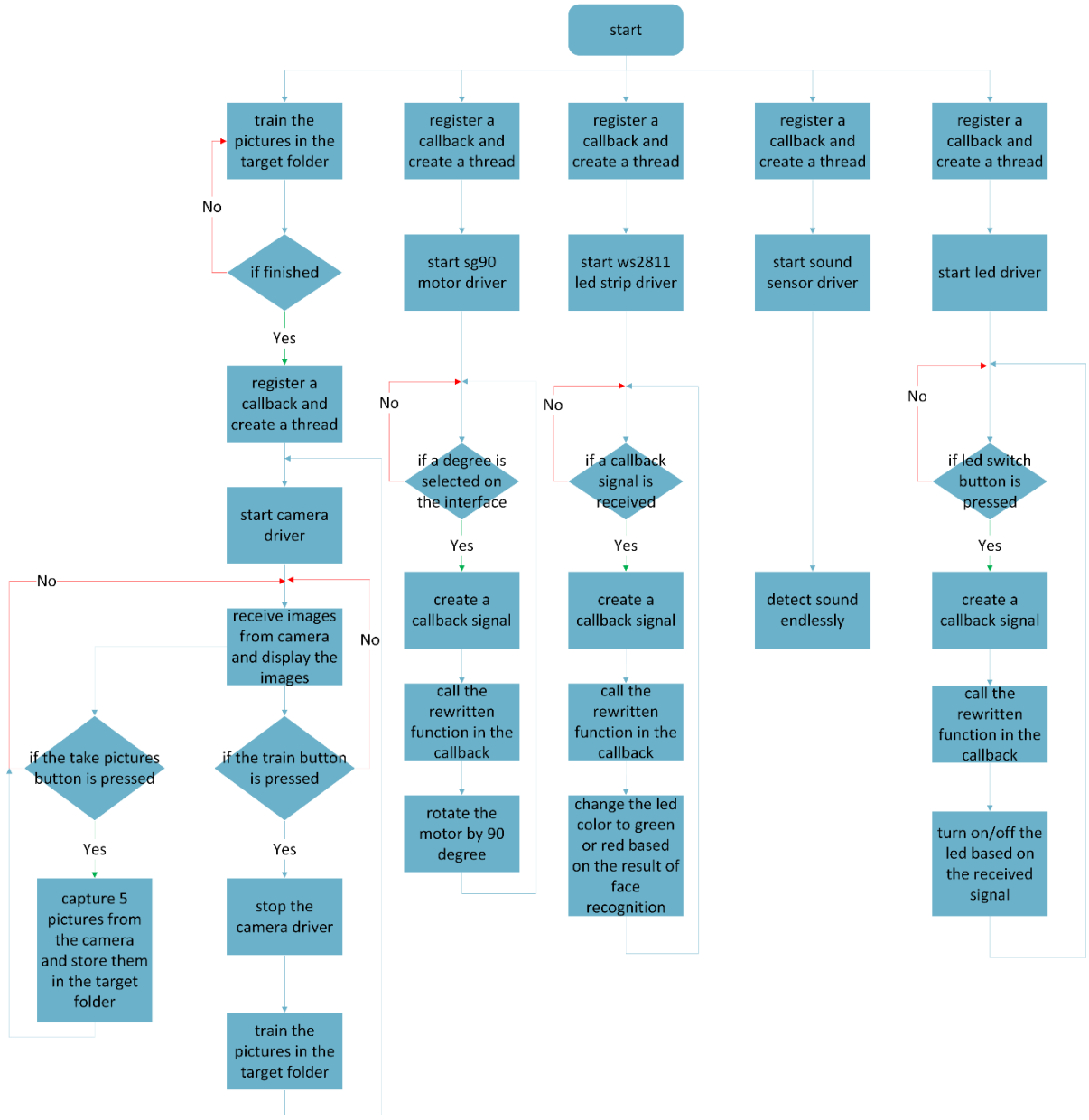


Figure 1 (b). The flow chart of the Manual Mode

5. Hardware

- Raspberry Pi 3B
- Sound Sensor
- 8MP Auto Focus Camera Module
- Digital Programmable LED Strip
- SG90 Micro Motor
- Breadboard, Dupont Lines, etc.

6. GUI Design

For better experience of using this system, a graphic operation interface is provided. So that, users can control the firmware individually by clicking some buttons when it's in the manual mode, which offers more interactions between users and the system. There are mainly four parts of operations provided: face entering/training, led control, motor control, led strip control. In addition, users are free to turn on the automatic mode and also stop it, which is more interactive compared with the version 1.0 in which the system is in automatic mode by default after the system is started. A picture of the interface is shown below:

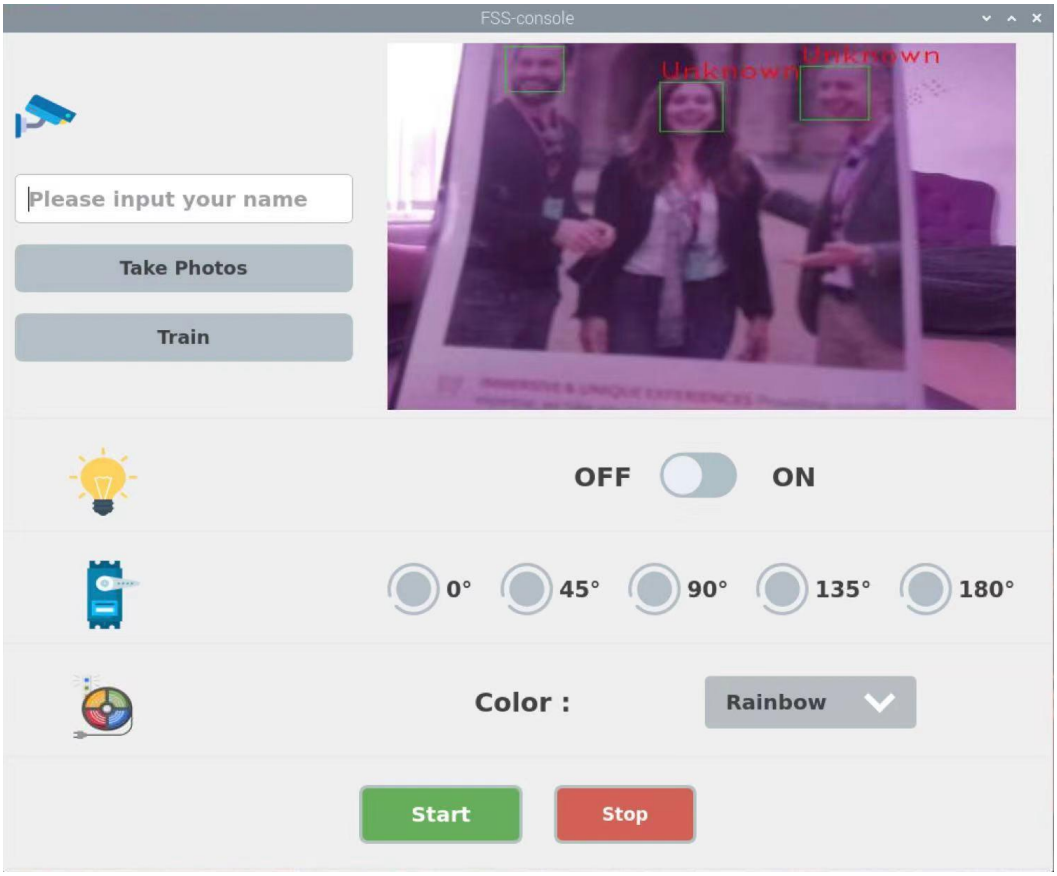
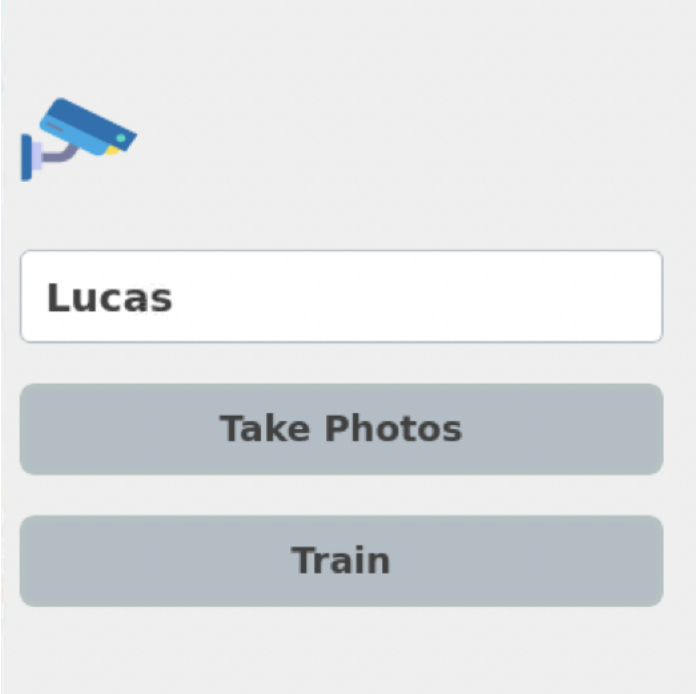


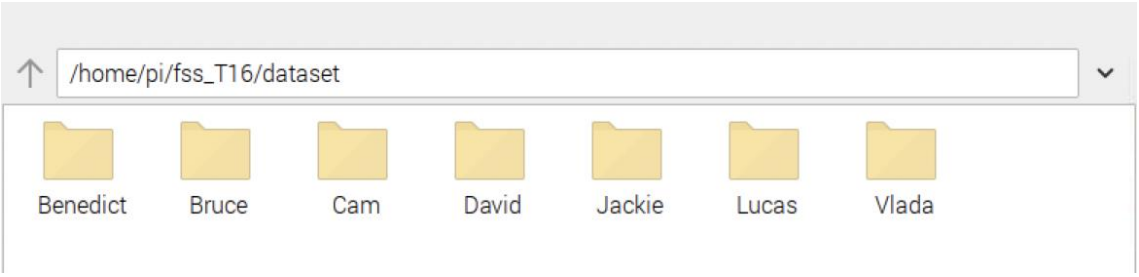
Figure 2. The GUI Designed for the System

6.1 Face Entering/Training

In the first part, there are three operations provided, input a name, take pictures and train the pictures. The interface is shown as follows:

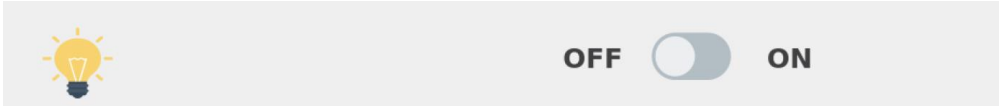


One thing should be noted when operating this interface: a name has to be input before clicking the take pictures button. Because a folder with its name offered by users will be created to store the pictures. After that, users should be able to do training with the new pictures. Here is a picture showing the path of the folder saved at:



6.2 LED Control

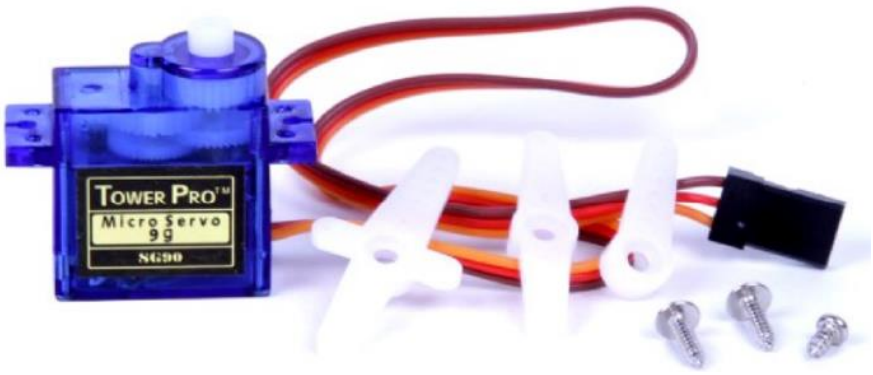
This interface allows users to turn on/off the LED bulb when the the system is in the manual mode. Users can not control the led bulb when it's in the automatic mode as it should be turned on/off based on the signal received from the sound sensor. This function gives users multiple choices when using this system, they can choose to keep the light on/off instead of letting it be a voice-controlled light. Here is a picture showing this interface:



6.3 Motor Control

In this interface, users can rotate the motor by choosing a degree of rotation meaning that users can open the door or lock the door remotely by a simple click, which is very convenient. Here is a picture showing this interface:

In this project, we use a type "SG90" servo motor. The position of this motor is controlled by the PWM signal. According to data sheet of this servo motor. One standard PWM period is 20ms(50Hz), and its signal amplitude on operation is 4.8v~6v.



The functions of three ports in different color on the servo are:

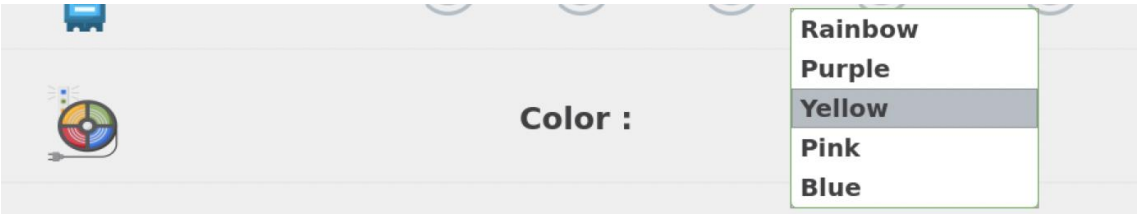
PWM	Orange port
Vcc	Red port (+)
Ground	Brown port (-)

Relationship beween position and PWM duty cycle:

0.5ms (2.5%)	0 degree
1.0ms (5.0%)	45 degrees
1.5ms (7.5%)	90 degrees
2.0ms (10.0%)	135 degrees
2.5ms (12.5%)	180 degrees

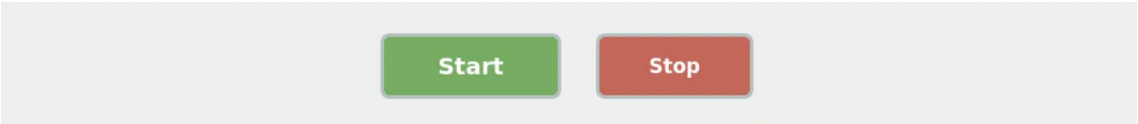
6.4 LED strip Control

Users can also choose the color of the LED strip, there are multiple choices provided. This allows users to select a color they would like to see in the manual mode, and in the automatic mode when no faces are detected. Noted that the LED strip will turn to green when a person is recognized and red when the person is not recognized by the system. The interface is shown as follows:



6.5 Start/Stop

As introduced above, users are free to switch the working mode in the system, the automatic mode and the manual mode. The system is in the manual mode by default. After the system is started, users can control each part of the firmware. Also, users can switch to the automatic mode by clicking the start button and switch back to the manual mode by clicking the stop button. Note that once the automatic mode is on, no operations are allowed to control the firmware, so all the buttons will be disable until the stop button is clicked. A picture of this interface is shown below:



7. Computer Vision

The computer vision (CV) part mainly concentrates on the face detection and face recognition, which is the core module of the whole system. The former determines whether a face is present and returns its pixel position, while the latter validates the input face against known information to determine whether there is a matched face in the database. Detection is an a priori condition for recognition. Therefore, OpenCV built-in function are used to detect the face first, and then performing face recognition. The face detection code is shown below.

```
CascadeClassifier classifier;
classifier.load("/home/pi/raspi_project_16/cascades/haarcasc
a de_frontalface_alt.xml");
classifier.detectMultiScale(frame, faces, 1.2, 5);
```

There are three methods given by OpenCV regards face recognition, which are Eigen Faces, Fisher Faces and Local Binary Patterns Histogram (LBPH), respectively. The project needs of CV are:

1. Superior accuracy;
2. High robustness and versatility;
3. Low hardware performance requirements;
4. Easy to update models;
5. Low cost.

Thus, the LBPH is used here. The advantage of this algorithm is that it is not affected by lighting, scaling, rotation and panning, and can also run on the not very high-performance platforms smoothly, making it feasible on Raspberry Pi.

The LBPH algorithm is to compare the grey value of the pixels on a circle of radius R with that of the current pixel , and if the surrounding pixel has a greater grey value, the position of the pixel is marked as 1, otherwise it is 0. By arranging the value of each point, the current pixel is given a binary LBP value. The relationship between (x_p, y_p) and (x_c, y_c) is:

$$\begin{cases} x_p = x_c + R\cos\left(\dfrac{2\pi i}{P}\right) \\ y_p = y_c - R\sin\left(\dfrac{2\pi i}{P}\right) \end{cases}, \ i \in P$$

where P is the number of sampling points. However, as the number of sampling points grows, the number of LBP patterns increases exponentially, to 2^P in total, which is not conducive to expressing graphic information. The LBP patterns, therefore, are integrated through a certain rule called

Equivalent Model, reducing the dimensionality of LBP patterns, so that the information of the image can be best represented with a reduced amount of data. The content about means of reducing pattern is not represented here.

With the method mentioned above, each pixel is given an LBP value based on surround information. The next step is LBP feature matching and the process is as follows:

1. Dividing images into non-overlapping areas;
2. Construct a grey scale histogram within each area;
3. Stitching the grey-scale histogram features of the whole image in a certain order to construct the overall features;
4. The similarity between the face to be recognized and the features from the database is calculated, and the highest similarity which is greater than the threshold is considered as the same person.

To train the model, the folders are named after the registrants' names and the photos corresponding to the registrants are put under the folders. During training, the folder name is used as the label and the vector obtained from the training is stored in .xml extension files, which is the model for face recognition. When validation, the confidence level returned by the function is compared to a set threshold to determine if it is a known individual.

8. Conclusion

Even though we have this project worked properly, there are still many improvements that can be done to have a better performance. The face recognition event does not work 100 percent precisely, sometimes, it could not recognize the same face it has identified before. Efforts on this are needed to improve the practicability. In addition, the function of our system is not hundred percent compatible with a real-life scenario, and it could be more fun. We were planning to build a dancing light that can dance with music in the environment. It requires extra sensors and efforts, but we believe we can work this out if we have enough time. Also, we need more

explorations on event-driven programming approach. This was like an unknown continent to most of us, which makes it hard for us to embark on coding before studying the concepts of it. Even though we finally got the system worked fine, we believe more efforts need to be invested on this to benefit our future career.

With the continuous hardworking of all of our team members, we managed to get this project done. We learnt a lot from our lecturers, Dr. Bernd Porr and Dr. Nicholas Bailey. Thanks to their guides and all the materials they offered, we got a better knowing of real-time embedding programming; we gained knowledge of many sensors and how to apply them in coding. We now have a clear understanding of what skills we need to enhance for the career we are pursuing.

9. Reference

[1] Event-driven methodology: https://berndporr.github.io/realtime_cpp_coding/

[2] GUI development: https://www.qt.io

[3] WS2811 led strip: https://github.com/jgarff/rpi_ws281x

[4] OpenCV official docs:
https://docs.opencv.org/3.4/da/d60/tutorial_face_main.html
https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html