



User Guide

Version 2.1

05 Mac 2020



BY TENSORFACTORY

DISCLAIMER OF WARRANTIES

* **TENSORFACTORY** MEANS TENSORFACTORY ENTERPRISE AND ITS OWNER.

** **PFCS** MEANS PIFACECAM-EDGE AND PIFACECAM-APP

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTY OR WHATSOEVER, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

YOU EXPRESSLY UNDERSTAND AND AGREE THAT YOUR USE OF THE PFCS** IS AT YOUR SOLE RISK AND THAT THE APPLICATION IS PROVIDED "AS IS" AND "AS AVAILABLE" WITHOUT WARRANTY OF ANY KIND FROM TENSORFACTORY*.

YOUR USE OF THE PFCS IS AT YOUR OWN DISCRETION AND RISK AND YOU ARE SOLELY RESPONSIBLE FOR ANY DAMAGE TO YOUR COMPUTER SYSTEM OR OTHER DEVICE OR LOSS OF DATA THAT RESULTS FROM SUCH USE.

TENSORFACTORY FURTHER EXPRESSLY DISCLAIMS ALL WARRANTIES AND CONDITIONS OF ANY KIND, WHETHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES AND CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT.

YOU EXPRESSLY UNDERSTAND AND AGREE THAT TENSORFACTORY SHALL NOT BE LIABLE TO YOU UNDER ANY THEORY OF LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES THAT MAY BE INCURRED BY YOU, INCLUDING ANY LOSS OF DATA, LOSS OF PROFITS, LOSS OF BUSINESS, INTERRUPTION OF BUSINESS, WHETHER OR NOT TENSORFACTORY HAVE BEEN ADVISED OF OR SHOULD HAVE BEEN AWARE OF THE POSSIBILITY OF ANY SUCH LOSSES ARISING.

SPECIFICATIONS AND INFORMATION CONTAINED IN THIS DOCUMENT ARE PROVIDED FOR INFORMATION USE ONLY, AND ARE SUBJECT TO CHANGE AT ANY TIME WITHOUT NOTICE, AND SHOULD NOT BE CONSTRUED AS COMMITMENT BY TENSORFACTORY. TENSORFACTORY ASSUMES NO RESPONSIBILITY OR LIABILITY FOR ANY ERRORS OR INACCURACIES THAT MAY APPEAR IN THIS DOCUMENT.

Copyright © 2020 TensorFactory Ent. All rights reserved.



BY TENSORFACTORY

Contents:

1 Overview

- 1.1 PiFaceCam System
- 1.2 Features

2 PiFaceCam-Edge setup

- 2.1 Hardware requirement
- 2.2 Software
 - 2.2.1 Getting system ready of PiFaceCam_Edge (Client/Verification server)
 - 2.2.2 Installing PiFaceCam_Edge (Client/Verification server)
 - 2.2.3 Naming your device
 - 2.2.4 Main program

3 PiFaceCam-App

- 3.1 Requirements
- 3.2 Versions
- 3.3 Overview
 - 3.3.1 Network Devices
 - 3.3.1.1 Search for devices
 - 3.3.1.2 Setting device id
 - 3.3.1.3 Status LED and device shutdown pin
 - 3.3.1.4 Setting passkey
 - 3.3.1.4.1 Missing passkey
 - 3.3.1.5 Camera type
 - 3.3.1.5.1 Stereo camera
 - 3.3.1.6 Face bounding box and constrains.
 - 3.3.1.7 Camera settings
 - 3.3.1.7.1 Camera index
 - 3.3.1.7.2 Use of custom frame size
 - 3.3.1.7.2.1 No frame size or width-height ratio setting for stereo camera setup.
 - 3.3.1.7.2.2 When custom frame size is not supported.
 - 3.3.1.7.3 Items to display in the live video feeds
 - 3.3.1.8 Detection settings
 - 3.3.1.8.1 Detection confidence level
 - 3.3.1.8.2 Bounding box constrains setting
 - 3.3.1.8.3 Delta between left and right cameras (for stereo cameras only)
 - 3.3.1.9 Verification mode (only available in Profession version of PiFaceCam_App)
 - 3.3.1.9.1 Information to server
 - 3.3.1.9.2 Information returned from server
 - 3.3.1.9.3 Guidelines for preparing reference image.



BY TENSORFACTORY

3.3.1.10 Video streaming

3.3.1.11 Program

3.3.1.12 View log

3.3.1.13 Setting upload

3.3.2 All Scripts

3.3.2.1 Creating a new script

3.3.3 All Programs

3.3.4 Create Face ID

3.3.5 All Face IDs

3.3.6 All Groups

4 Implementation Examples

- 4.1 Case 1: Simple auto door unlocking and alerting (buzzer and email).
- 4.2 Case 2: Remote controlling and displaying of ids through JSON server.



BY TENSORFACTORY

1.0 Overview

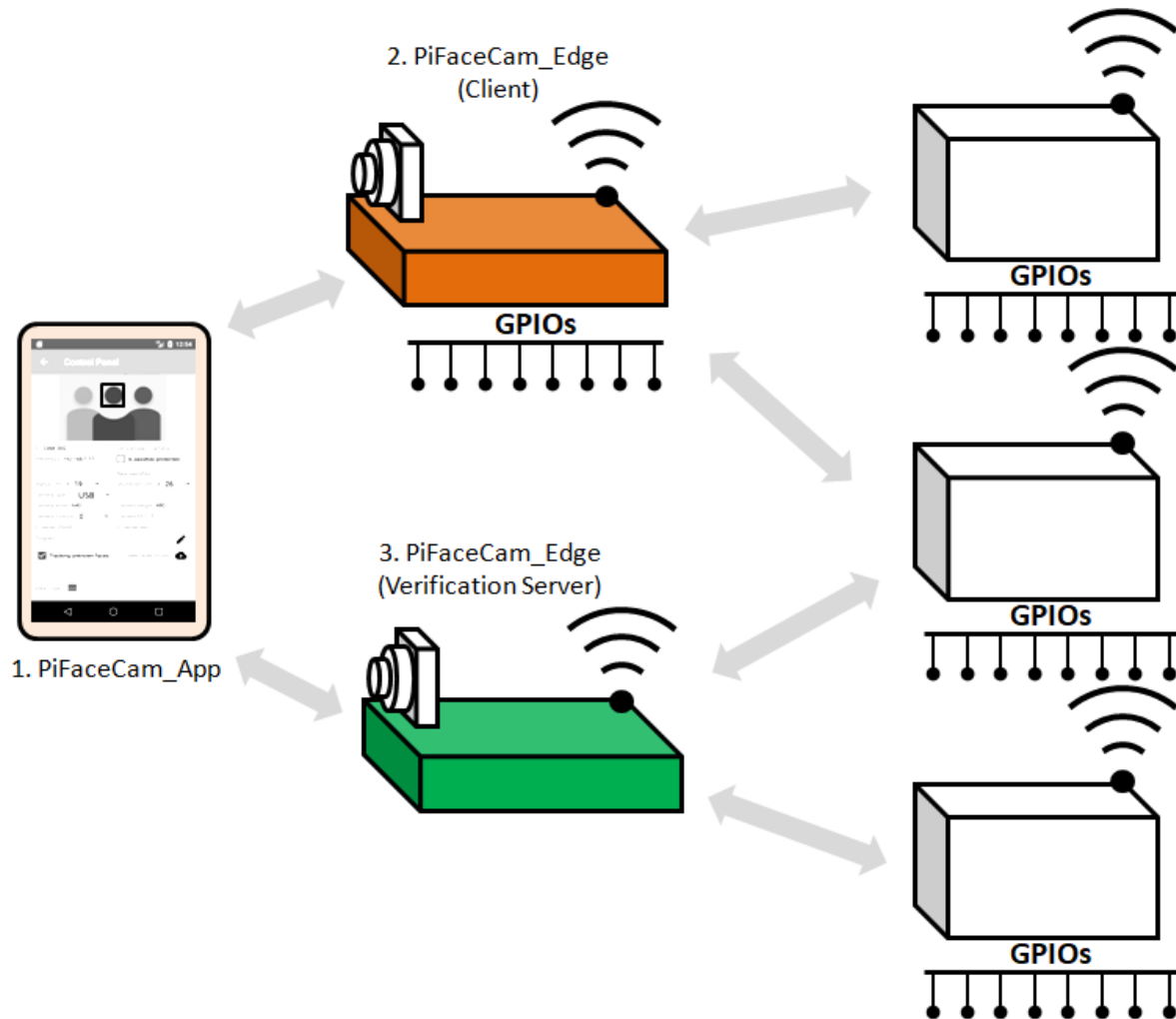


Figure 1.0

1.1 PiFaceCam system

PiFaceCam system uses deep neural network for accurate facial recognition. It is designed from ground-up to run efficiently in Raspberry Pi*.

PiFaceCam currently consists of 3 components (which collectively known as PiFaceCam system).

1. **PiFaceCam_Edge (Client)**: Runs in Raspberry Pi* with camera(s) attached. It performs all facial recognition processes and can be programmed to carry out instructions using simple scripts. It can also be programmed to send facial data to other devices.



BY TENSORFACTORY

2. **PiFaceCam_Edge (Verification server):** Runs in Raspberry Pi* with camera(s) attached. It acts as verification server where client send images for verifications. It compares the face in received image with the one detected in front of the camera(s) and return the verification results to clients.
3. **PiFaceCam_App:** Runs in Android devices (currently only support android devices). It is for managing and controls all PiFaceCam devices connected through local network.

1.2 Features

- Facial recognition based on assigned face ids and carry instruction per scripts.
- Support stereo camera setting where 2 cameras are used to capture images of a person at different angles as an added level of security against simply attack using photo.
- Comprehensive tools for collecting, editing and organizing face ids.
- Simple scripting concept for automation.
- User-friendly tools to create, edit and manage scripts/programs.
- Remote monitor, manage and control multiple devices.
- Built-in functions to control GPIOs, communicate with JSON-server and sending emails.



BY TENSORFACTORY

2.0 PiFaceCam_Edge setup

2.1 Hardware requirement

PiFaceCam_Edge requires Raspberry Pi 3 / 4 (Tested on Pi3 Model B+ and Pi4 Model B 4GB) to run. As facial-recognition is computationally heavy, proper heat management is required. Standard cooling fan with heat sink on CPU was tested to be sufficient in both Pi3 Model B+ and Pi4 Model B.

By default, GPIO 19 and 26 are reserved for status and shutdown purpose. GPIO 19 should be connected to a LED via a resistor. The LED will blink during system loading and on continuously when system is ready. LED blinks indefinitely signify error has occurred. GPIO 26 will trigger system shutdown when connect to high.

2.2 Software

2.2.1 Getting system ready of PiFaceCam Edge (Client/Verification server)

This PiFaceCam_Edge was developed for Raspbian Buster with desktop (Kernel version: 4.19). You will also need to install the following supporting libraries.

1. Python3 (develop in version 3.7.3)
2. Tensorflow 1.X (develop in version 1.15)
3. OpenCV for python (develop in version 4.2.0)
4. Scikit-learn (develop in version 0.22)
5. Pycryptodomex (develop in version 3.9.6)

2.2.2 Installing PiFaceCam Edge (Client/Verification server)

After you have installed all the necessary libraries, copy the whole "pifacecam_edge" folder from github (<https://github.com/tensorfactory/PiFaceCam>) into the home directory.

If you don't want to go through all the trouble of installing everything yourself, we are sharing out SD Card's image file at github as well. Please take note that we are using 16GB SD Card, therefore you will need a large card to flash the image.

After flashing your SD Card, remember to reclaim the missing disk space by running raspi-config -> Advanced Options -> Expand Filesystem Ensures that all of the SD card storage is available.

PiFaceCam_Edge's files are placed in "home/pi" folder.



BY TENSORFACTORY

2.2.3 Naming your device

The default device id is "CAM001". The first thing you need to do is to give your device a unique name (Maximum 15 characters) by placing it in the "id.txt" file or setting it from PiFaceCam_App (Note: Value from PiFaceCam_App supersedes value in id.txt).

2.2.4 Main program

In the folder, you will see "PiFaceCam_Edge_X_X_X.so". This is the PiFaceCam_Edge program. To use it, first import "PiFaceCam_Edge_X_X_X" to your project and call the "run()" method.

Once started, "run()" will not return until user terminates it by setting the "shutdown" GPIO pin to high. We can make use of this blocking characteristic as shutdown mechanism for raspberry pi when " PiFaceCam_Edge " exit, as in below example.

```
import PiFaceCam_Edge_2_0_0
import os

if __name__ == "__main__":
    PiFaceCam_Edge_2_0_0.run()
    os.system("sudo shutdown -h now")
```

(wrapper.py)

The default mode of PiFaceCam_Edge is client mode. It can be switch to run in verification-server mode using PiFaceCam_App (Professional Edition).



BY TENSORFACTORY



3.0 PiFaceCam-App

3.1 Requirements

Requires Android OS Marshmallow and above. The device will also require network connection to communicate with PiFaceCam_Edge and camera if you want to create face ids directly from image captured. It allows you to control and manage all connected PiFaceCam_Edge devices.

3.2 Versions

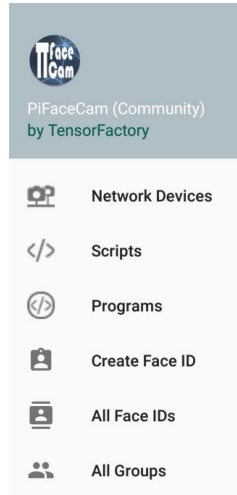
There are 2 versions of PiFaceCam_App, PiFaceCam (Community) and PiFaceCam (Professional).

Features	Community 	Professional 
Create and manage scripts / programs.	Yes	Yes
Create and manage face ids.	Limited to 20	Not limited
Setting and manage of attached cameras.	Yes	Yes
Setting facial recognition criteria.	Yes	Yes
Enabling face verification server mode.	No	Yes
Price	Free Download from Google Play Store.	Paid Please check Google Play Store for more info.



BY TENSORFACTORY

3.3 Overview



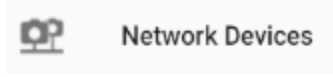
From the main page, you will be able to navigate to all the features.

- 1) Network Devices:
 - a) Scan/access all connected PiFaceCam_Edge devices.
 - b) Change settings of connected PiFaceCam_Edge devices.
 - c) Load face ids and program to PiFaceCam_Edge devices.
- 2) Scripts:
 - a) List and manage of all scripts.
 - b) Create and edit scripts.
 - c) Export/import scripts to file.
- 3) Programs:
 - a) List and manage of all programs (Which are combinations of one or more scripts).
 - b) Create and edit programs.
 - c) Export/import programs to file.
- 4) Create Face ID:
 - a) Create face ids from camera or image gallery.
 - b) Edit face ids.
- 5) All Face IDs:
 - a) Shown all available face ids.
 - b) Export / import face ids to file.
- 6) All Groups:
 - a) List and manage of all groups (Which are combinations of one or more face ids).
 - b) Export / import groups to file.



BY TENSORFACTORY

3.3.1 Network Devices



3.3.1.1 Search for devices

Key-in single or range of IP addresses to search.
Example: 192.168.1.6, 192.168.1.1-254.

 **CAM_002** IP: 192.168.1.4(Connected)
28-02-20 09:50:44
Ver:2.1
Key: Not Available. 

You can search all PiFaceCam_Edge devices that are connected to the local network. Or, if you already know the ipaddress of the device, you can key-in the exact address.

All connected PiFaceCam_Edge devices will be listed here. You can access the device by clicking on the list. The padlock symbol will indicate if the connected device is passkey locked which will require passkey to access. Once clicked, you will reach the control panel.

3.3.1.2 Setting device id

ID: Device type: Camera

IPAddress:192.168.1.4

From here, you can change the device ID. The maximum character for device ID is 15 (extra characters will be trimmed).

3.3.1.3 Status LED and device shutdown pin

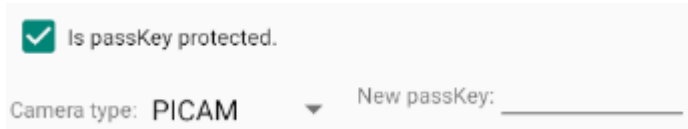
Status GPIO #: Shutdown GPIO #:

You can set the GPIO pins number for device status and device shutdown. The device status pin will be connected to a LED via a resistor. Blinking indicates loading and continuous on indicates device is ready. The shutdown pin will trigger device to shutdown when connected to high.



BY TENSORFACTORY

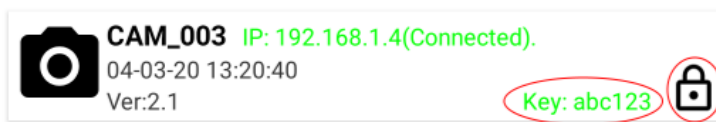
3.3.1.4 Setting passkey



By default, the attached PiFaceCam_Edge device is not passkey protected. To enable passkey protection, check the "Is passkey protected checkbox and provide a passkey and click the settings upload button.



(Setting upload button)



Once a passkey was set, you will notice the lock icon in the network devices list. The passkey will also be saved in this PiFaceCam_App.

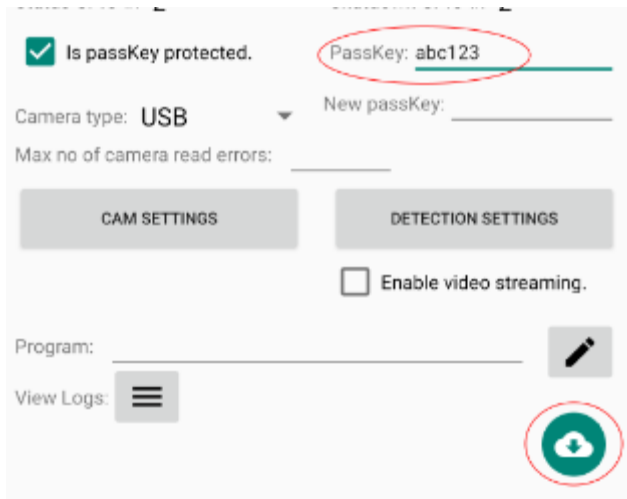
3.3.1.4.1 Missing passkey

If for some reasons, you have deleted the device from the list and try to add back by searching for it again, you will notice PiFaceCam_App no longer has the passkey.



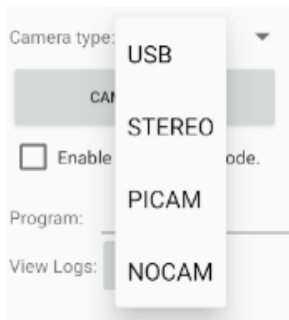
Can't access device's setting without passKey.

In this event, if you try to connect to the device by clicking on the list, you will get a "Can't access device's setting without passkey" message.



In order to resolve this, you need to key in the passkey and click on the setting download button (above).

3.3.1.5 Camera type

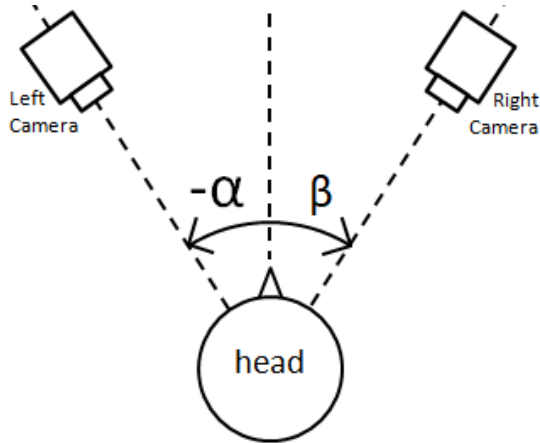


For camera types, you have options of USB, STEREO and PICAM. If you use a single generic USB camera, choose "USB" option. If you are using single Pi-Camera, choose the "PICAM" option. Note, you can choose "USB" option when using Pi-Camera and it will still works but you will not be able to access to Pi-camera specific features (camera rotation and FPS setting). For Pi-Camera, we recommend using 30FPS, camera width 640 and height 480. (Note: Although the image processing rate is around 8~12FPS (much lower than 30FPS), using 30FPS camera setting will ensure there is always image ready for image processing.)



BY TENSORFACTORY

3.3.1.5.1 Stereo camera



Cameras layout (top view)

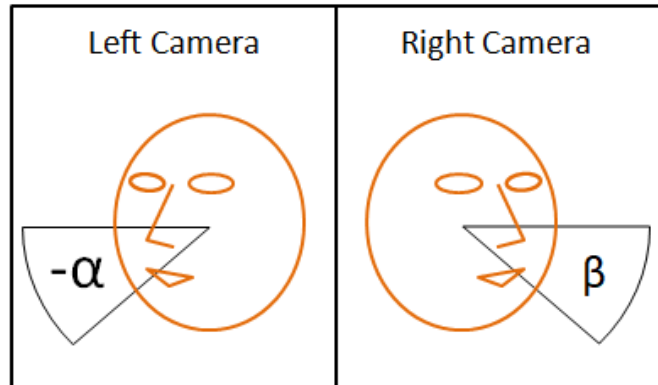


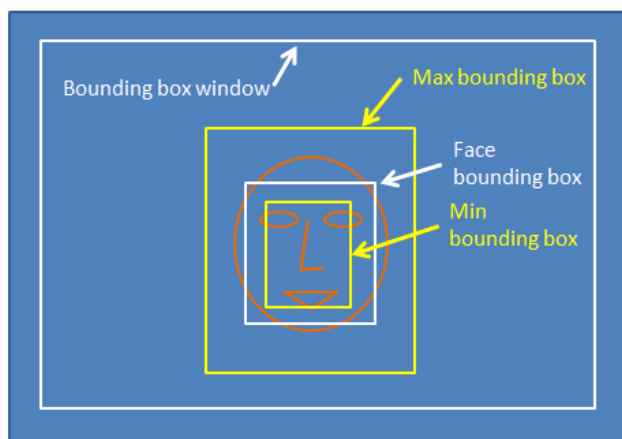
Image from left and right cameras

To defense against attack using photo, you can use stereo cameras setup. It works by detecting the face angles from the left and right camera, $-\alpha$ & β . The difference of these angles ($\beta - (-\alpha) = \beta + \alpha$) should be about the same as the angle between the left and right camera.

For example, the left and right cameras were placed 40° apart, $\beta - (-\alpha)$ will be close to 40° . However, if a photo was placed in front of both cameras, both cameras will measure the same face angle and $\beta - (-\alpha)$ will be close to 0° .

More about setting an acceptable range for $\beta - (-\alpha)$, named as "delta angle", later in detection setting section.

3.3.1.6 Face bounding box and constrains



Bounding box constrains



BY TENSORFACTORY

The face bounding box is area tracked and used by PiFaceCam for facial recognition. This bbox will appear in the live-feed video if you enable "Show bounding boxes" in the camera setting page.

This face bbox will be in white color. However, if it failed any of the dimensional criteria (smaller than the min bounding box, larger than the max bounding box, touching or go outside the bounding box window, or if it is stereo camera setup, failed any of the delta dimension ranges), it will change to red color.

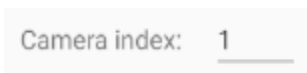
If it is in client mode (not operating as verification server), it will turn green when the face match any of the registered face ids.

3.3.1.7 Camera settings



You can change the camera settings by clicking on the "CAM SETTINGS" button.

3.3.1.7.1 Camera index



If you have more than one camera attached to PiFaceCam, you can choose which camera by index number.



If you are using stereo camera setup, make sure you choose the correct camera for left and right as per shown in 3.3.1.5.1 as swapping the cameras will gives negative delta angle value.



BY TENSORFACTORY

3.3.1.7.2 Use of custom frame size

☒ Use custom frame size.

Frame width (px):

Frame height (px):

We do not recommend the use custom frame size. If you choose custom frame size, PiFaceCam will try to set the camera as per your requirement. However, whatever the size of the output image of your camera, PiFaceCam will crop and resize the image to meet the facial recognition neural network requirement. Therefore having a very high resolution will not improve the accuracy but may have an adverse effect of slowing down the process.

Frame width/height ratio: 1.00

Display:

- ☒ Show bounding boxes.
- ☒ Show face ids.
- ☒ Show cam id.
- ☒ Show FPS.

If custom frame size is not used, for USB camera, PiFaceCam will use the default camera output size. It will then crop and resize the output image to satisfy both the frame width/height ratio settings (above) and facial recognition neural network requirement. For Pi-camera, PiFaceCam will choose a camera output size that meets both the frame width/height ratio setting and facial recognition neural network requirement so that no post cropping or resizing is required.

3.3.1.7.2.1 No frame size or width-height ratio setting for stereo camera setup.

In stereo camera setup, neither frame size not width/height ratio setting is available. PiFaceCam use the default camera output size. Images from the left and right will be cropped into width/height ratio of 0.93 which combine to width/height ratio of 1.86 for live streaming.



BY TENSORFACTORY

3.3.1.7.2.2 When custom frame size is not supported.

```
30) 04/03/2020 21:26:45: USB camera (1) not supporting  
size 320x480. Reset to 640x480
```

(From attached device's event logs)

In some instances, the attached camera does not support the frame size that we set. In this example, we try to set the frame size of USB camera to 320x480 (WxH), the camera was not able to support this resolution and switch to its default setting of 640x480. This can be found in the device's event logs (above). When this happened, the output image will be cropped to the nearest ratio of our initial setting. In this example, our initial setting was 320x480 and the nearest W/H ratio was 1.0.

3.3.1.7.3 Items to display in the live video feeds

- ☒ Show bounding boxes. If selected, will display face bounding boxes.
- ☒ Show face ids. If selected, will display recognized face ids (only in client mode).
- ☒ Show cam id. If selected, will display the device's id.
- ☒ Show FPS. If selected, will display the average frame per seconds.
- ☒ Show measurements. If selected, will display the delta values between left and right cameras (only for stereo camera setup).
- ☒ Show positioning guides. If selected, will display the min/max bounding boxes and window bounding box.

3.3.1.8 Detection settings



You can change the detection settings by clicking on the "DETECTION SETTINGS" button.



BY TENSORFACTORY

3.3.1.8.1 Detection confidence level

Detection confidence level (%): 99.9

You can adjust the confidence level of facial recognition. Setting it at 99.9% will mean PiFaceCam will only return positive if it is 99.9% certain the recognition is correct. This value was calibrated using “Labeled Faces in the Wild” dataset.

3.3.1.8.2 Bounding box constrains setting

Min bbox width: <u>0.14</u>	Min bbox window_x: <u>0.05</u>
Max bbox width: <u>1.0</u>	Max bbox window_x: <u>0.95</u>
Min bbox height: <u>0.18</u>	Min bbox window_y: <u>0.05</u>
Max bbox height: <u>1.0</u>	Max bbox window_y: <u>0.95</u>

You can limit the size of face bounding box and also constrain it to within a window in the camera’s frame. The value is set in ratio of the image width or height.

3.3.1.8.3 Delta between left and right cameras (for stereo cameras only)

Max delta bbox width (%): <u>20.0</u>
Max delta bbox height (%): <u>20.0</u>
Min delta face angle (deg): <u>30.0</u>
Max delta face angle (deg): <u>60.0</u>

If it is a stereo camera setting, you can constrain the difference between the left and right cameras.



BY TENSORFACTORY

3.3.1.9 Verification mode (only available in Profession version of PiFaceCam App)

☒ Enable verification mode. ☐ Enable video streaming.

Verification Server Port: Verification Token:

In verification mode, you need to provide the port number and verification token. The verification token will be used by the client during request.

On setup, the PiFaceCam_Edge will act as a verification server. Any client can send to it a reference image for verification. The verification server will return if the person currently in front of the camera matches the person in the reference image.

3.3.1.9.1 Information to server

To request for verification, the client needs to send both the reference image and token in JSON format to server.

Element	Key	Value
Token	"token"	string
Image	"image"	Reference image (base64 string)
If need to return image	"need_return_image"	boolean

The image bytes have to be converted to base64 string (see below example). The length of this JSON bytes needs to be included at the beginning of the packet when send to verification server. The length value is encoded in a big-endian ordering 4 bytes.

```
# -----Load image bytes from file-----
img_filestream = open(image_file_pathname, "rb")
ori_image_bytes = img_filestream.read()
img_filestream.close()
# -----Convert to base64 string-----
ori_image_base64 = base64.b64encode(ori_image_bytes)
ori_image_base64_str = ori_image_base64.decode('utf-8')

# -----Compose JSON packet-----
JSON_dict = {"token": server_token,
            "image": ori_image_base64_str,
            "need_return_image": need_return_image}

JSON_str = json.dumps(JSON_dict)

# -----Convert to binary string and add length of packet-----
JSON_bytes = JSON_str.encode("utf-8")
len_of_packet = len(JSON_bytes)

len_of_packet_bytes = pack('>I', len_of_packet)
total_bytes = len_of_packet_bytes + JSON_bytes
```



BY TENSORFACTORY

```
# -----Send packet to verification server-----  
server_address = (server_ipaddress, server_port)  
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
  
sock.connect(server_address)  
sock.sendall(total_bytes)
```

3.3.1.9.2 Information returned from server

Return JSON object.

Key	Value
"isSuccessful"	True / False
"errMessage"	Error message if isSuccessful is False
"noOfFaces"	Number of faces detected. Only faces that passed the dimensional constrain check.
"imageWidth"	Width of the image from PiFaceCam_Edge.
"imageHeight"	
"faces"	List of faces detected. Only faces that passed the dimensional constrain check.
"returnImage"	Returned image (base64 string)

Faces list object.

Key	Value
"top"	Top y-coordinate of face bounding box.
"left"	Left x-coordinate of face bounding box.
"width"	Width of face bounding box.
"height"	Height of face bounding box.
"confPercentage"	Confidence level (%) of face matches image sent.

Similarly, the first 4 bytes of the returned packet indicates the length of packet.

```
# -----Receive reply from server-----  
len_of_packet_bytes = sock.recv(4)  
len_of_packet_int = unpack('>I', len_of_packet_bytes)[0]  
  
received_data = b''  
while len(received_data) < len_of_packet_int:  
    packet = sock.recv(len_of_packet_int - len(received_data))  
    if not packet:  
        break  
    received_data += packet  
  
sock.close()  
  
#-----Decode received binary string to JSON object-----  
received_data_string = received_data.decode("utf-8")
```



BY TENSORFACTORY

```
received_data_JSON_obj = json.loads(received_data_string)

# -----Retrieve verification results-----
isSuccessful = received_data_JSON_obj.get("isSuccessful")
if isSuccessful:
    noOfFaces = received_data_JSON_obj.get("noOfFaces")
    imageWidth = received_data_JSON_obj.get("imageWidth")
    imageHeight = received_data_JSON_obj.get("imageHeight")

    if noOfFaces > 0:
        faces = received_data_JSON_obj.get("faces")
        for face_idx, face in enumerate(faces):
            face_top = face["top"]
            face_left = face["left"]
            face_width = face["width"]
            face_height = face["height"]
            confPercentage = face["confPercentage"]

            print("Face {}: confident {:.2f}%".format(face_idx, confPercentage))
    else:
        print("No face detected.")

    if need_return_image:
        returned_image_base64_str = received_data_JSON_obj.get("returnImage")
        returned_image_base64 = returned_image_base64_str.encode("utf-8")
        returned_image_bytes = base64.b64decode(returned_image_base64)

        returned_image_array = np.frombuffer(returned_image_bytes, np.uint8)
        returned_image_np = cv2.imdecode(returned_image_array, cv2.IMREAD_COLOR)

        cv2.imshow("Returned image", returned_image_np)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
else:
    print("Server return error message :" + received_data_JSON_obj.get("errMessage"))
```

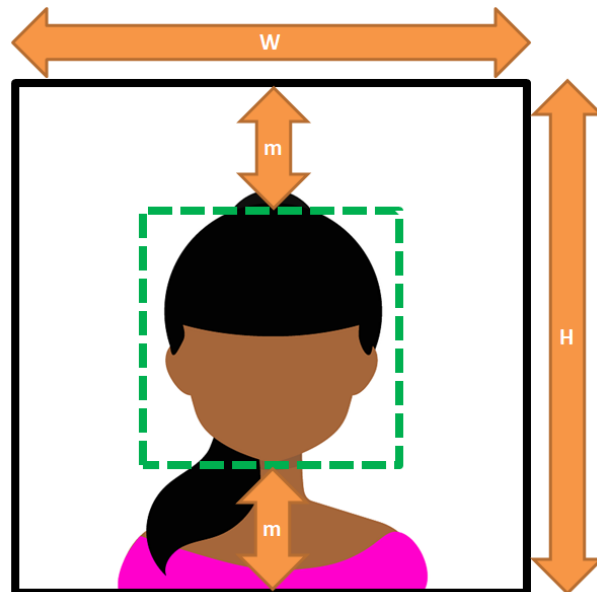
For complete code example, go to the following link.

https://github.com/tensorfactory/PiFaceCam/example_verification_client/example_verification_client.py



BY TENSORFACTORY

3.3.1.9.3 Guidelines for preparing reference image.



The reference image used for verification has to meet the following guidelines, failing which may affect the accuracy or getting rejected by the verification server.

- 1) The face (green box) has to be at the center of the image.
- 2) The image has to be square ($W = H$).
- 3) The margin (m) between face and border of image has to be within 10~25 % of the image size (H). (Example, if the image size is 500pixels x 500pixels, then m has to be within the range of 50 to 125pixels.)
- 4) Image size (W and H) has to be within the range of 320 to 512 pixels.

3.3.1.10 Video streaming



Once enabled, it will start to stream live video from port 9090. The video can be access using from URL such as <http://169.254.121.52:9090/video> where in this case the "169.254.121.52" is the ipaddress of the PiFaceCam_Edge device.



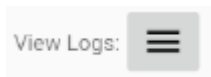
BY TENSORFACTORY

3.3.1.11 Program



Click on the “Pencil” icon to select a program you want to run on the connected device.
(Note: Delete the program's name at the edit box and click upload button will remove any program at the connected device.)

3.3.1.12 View log



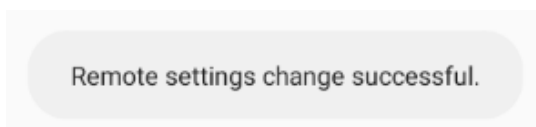
You can access the error/warning logging of remote device here.

3.3.1.13 Setting upload



(Setting upload button)

After you have made all changes, you need to click the setting upload button to upload everything to the connected device before those changes will take effect.



If successful, you will see a "Remote settings change successful." message.

Note: During setting upload, all face ids from your face ids list will also be uploaded to the connected device.

3.3.2 All Scripts





BY TENSORFACTORY

You will see all your created scripts (if available) listed here. Click on “+” button to add scripts. You have option to import from a “.srp” file or create from scratch.

You can export any of the scripts to file by clicking on the share button at the top.

3.3.2.1 Creating a new script

Script Editor

Name: Open Home Door

Description: Open door when detect owner

Faces: Huayi; KC Yee;

To create a new script, first give you script a meaningful name and description.

Next Click on “pencil” icon to select face-ids that you will be using in this script.

Faces:

Trigger type:

Action type:

GPIO #:

Trigger duration(s):

Min duration btw trig(s):

When detect

When not detect

When detect others

When detect any

When not detect any

Always OFF

Select a trigger type.

- 1) Action is triggered when any of the selected faces are detected,
- 2) action is triggered when the none of selected faces are detected,
- 3) action is triggered when any faces other than those that were selected are detected,
- 4) action will triggered when detect any faces and
- 5) action will be triggered when no face is detected.



BY TENSORFACTORY

Action type: **Trigger GPIOs** ▼

GPIO #:

Trigger duration(s):

Min duration btw trig(s):

☒ Always OFF

Next is to select the action type. You have options of triggering a GPIO, sending an email or upload face id list and a key value to a JSON server.

GPIO #: ☒ Always OFF

Trigger duration(s):

Min duration btw trig(s):

For GPIO, you can select any GPIO pin to trigger but it has to be different from the status and shutdown pin. You can also select always OFF or always ON mode. In always OFF mode, the pin will be always at low state until it is trigger and vice versa.

You can set the trigger duration and the minimum time interval between triggers "Min duration btw trig(s)". However, as with other temporal settings, their accuracies are not very reliable and shall not be depended on for any application.

Action type: **Send Emails** ▼

Min trigger interval(s):

Server address:

Server port:

Email password: ☒ Use SSL (not TLS)

Sender address:

Receiver addresses:

Subject:

Body:

☒ Attach Image

Max send per period: Period duration(s):



BY TENSORFACTORY

For action type of sending email through a smtp server. You have to provide the necessary smtp server information.

For security protocol, you can choose between SSL and TLS. Uncheck the "Use SSL" checkbox will select TLS protocol.

You have the option to provide a short email subject and body, and to include an attachment of the image that triggers this action.

In this example (left), to prevent multiple emails send when this script is triggered continuously, we set the min trigger interval to 5 seconds, meaning after the first email sent, it will wait for 5 seconds before the next triggering can happen. Also we set the max send per period to 20 and the period to 600 seconds (10 minutes) which means, in every 10 minutes, the script can only send a maximum of 20 emails. The counter will reset after each 10 minutes period.

Action type: Upload JSON

Min trigger interval(s): 0.1

Server address: 192.168.1.15

Server port: 9000

Max send per period: 100

Period duration(s): 60

Key value: 101

For action type, a JSON packet will be send to the JSON server with the following format.

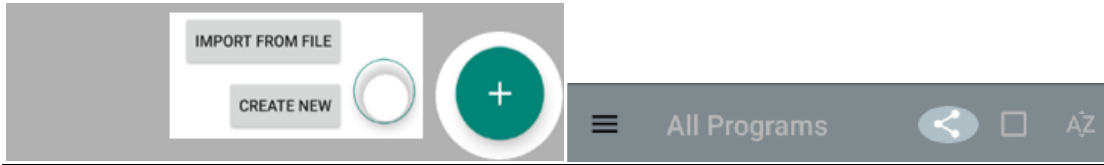
```
{  "device_id": device_id,
  "num_of_faces": num_of_faces,
  "faceID_list": faceID_list,
  "key_value": key_value,
  "time_stamp": time_stamp }
```

"faceID_list" is list of face ids in the image that trigger this action. "key_value" is an arbitrary integer value to pass over to JSON server for identification. "time_stamp" is the number of seconds that have elapsed since January 1, 1970, 00:00:00 (UTC). Please see an example of implementation in our github folder.



BY TENSORFACTORY

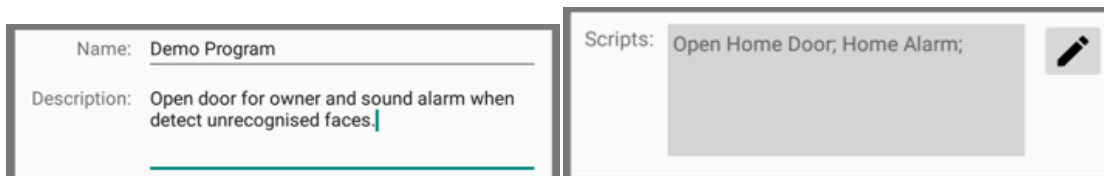
3.3.3 All Programs



You will see all your created programs (if available) listed here. Click on “+” button to add programs. You have option to import from a “.prg” file or create from scratch.

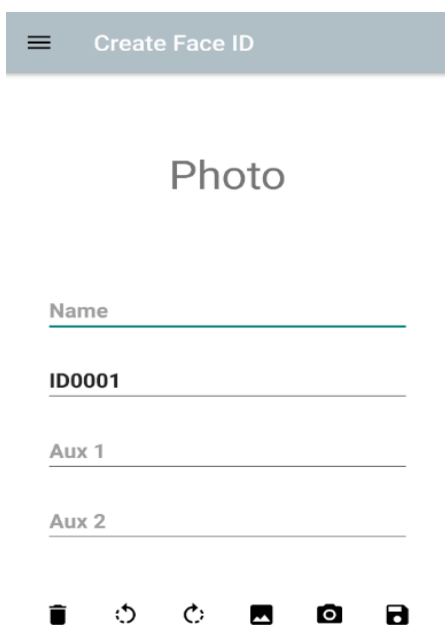
You can export any of the programs to file by clicking on the share button at the top.

A program is a group of scripts. As each script is a unit of detection-action pair, using program will allow us to implement multiple scripts at once.



To create a new program, first give you program a meaningful name and description. Next click on “pencil” icon to select scripts that you want to include in this program.

3.3.4 Create Face ID





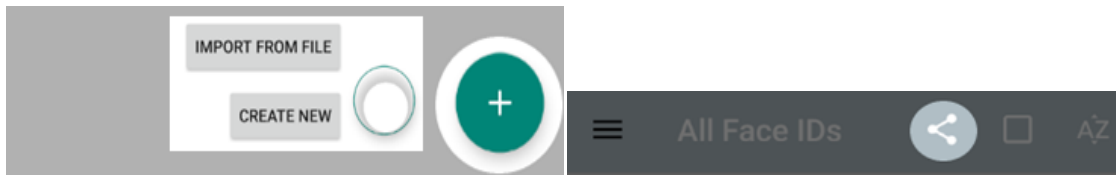
BY TENSORFACTORY

Click on “Photo” or the gallery or camera button to start import face photos of this person. Name is compulsory and ID is auto-generated (can be altered). All imported face photos will be shown at the top and rejected photos will have a red cross.

You can delete a selected photo by clicking on the dustbin. (Long click will delete all photos). You can also rotate the selected photo by clicking at the rotation button at the bottom.

Note: If you import more than 5 photos, PiFaceCam will choose the best 5 photos to used (saved).

3.3.5 All Face IDs

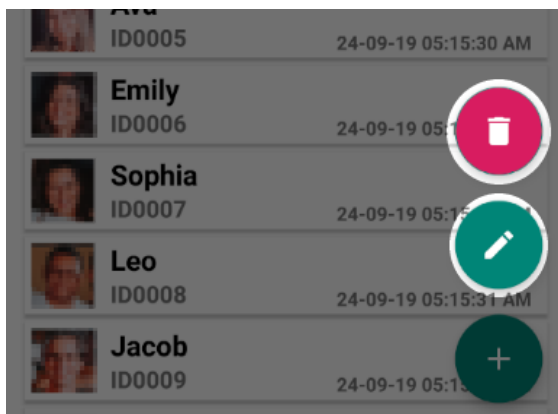


You will see all your created scripts (if available) listed here. Click on “+” button to add face ids. You have option to import from a “.fig” file or create from scratch.

You can export any of the face-ids to file by clicking on the share button at the top.

Long click on any face id to start selection.

Click on any item will open edit dialog.

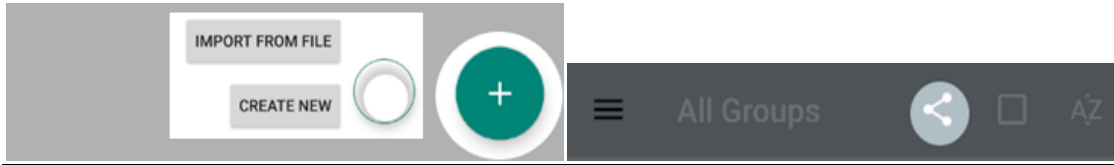


You can also delete or edit selected face id by clicking on the “dustbin” or “pencil” button.



BY TENSORFACTORY

3.3.6 All Groups



Group is a collection of face ids. This is to help us organize our face-ids.

You will see all your created groups (if available) listed here. Click on “+” button to create new group. You have option to import from a “.grp” file or create from scratch.

You can export any of the groups to file by clicking on the share button at the top.

If you choose to create new one, you will be prompted for a name.

Start adding face IDs as member of this group by clicking on the "+" button. Long click on any face id to start selection. You can delete selected IDs or continue to add more.

Remember to save changes before leaving.





BY TENSORFACTORY

4.0 Implementation Examples

4.1 Case 1: Simple auto door unlocking and alerting (buzzer and email).

In this example, we would like to automate our office's door locking so that only authorized personnels can enter and would like to receive an email if detect unauthorized person trying to enter. We will be using a Raspberry Pi (3B+ or 4B) with PiCamera attached.

Requirements:

- Keeping list of face ids of authorised personnels.
- When detect any of the authorised personnels, trigger door unlock for 3 seconds by setting GPIO 4 to high for 3 second.
- At the same time we would like to have a buzz of 1 second whenever door unlock is triggered. This is done by setting GPIO 5 to high for 1 second.
- When there an unauthorised person trying to enter, we would like our HR department to be alerted through email (hr_manager@company.com). The email should include an image of the person face.

Note: It is assumed PiFaceCam-Client is properly installed as per instruction in section (2.0) and the user is "pi". Also, the Raspberry Pi has and will auto connect to the local network.

Step 1) Setting up the PiFaceCam-Client to auto-start everytime we power ON the Raspberry Pi.

a) Create a script file "launcher.sh" with below lines

```
#!/bin/sh
sleep 10
cd /
cd home/pi/pifacecam_edge
sudo python3 wrapper.py
```

b) Make the launcher script an executable, with command "chmod 755 launcher.sh"

c) Make a directory for the any errors in crontab to go

```
Navigate back to your home directory:
cd
Create a logs directory:
mkdir logs
```

d) Add to Your Crontab. To brings up a crontab window, run

```
sudo crontab -e
```

e) Now, enter the line:

```
@reboot sh /home/pi/pifacecam_edge /launcher.sh >/home/pi/logs/cronlog 2>&1
```



BY TENSORFACTORY

Step 2) Setting up the face-ids of authorised personnels.

- a) Following instructions from section 4.3.4 , create a face id for each of the authorised persons.
- b) Create a group for these ids (name it something like "department A") following section 4.3.6.

Step 3) Creating scripts.

There are three detection-actions in this implementation. 1) Detect authorised personnels, unlock door, 2) detect authorised personnels sound buzz and 3) detect un-authorised persons, send email. We will create 3 scripts, one for each of the detection-action as per section 4.3.2.

i) Script 1:

Name: Unlock door (Department A)
Description: ON GPIO 4 for 3 seconds when detect authorised personnels.
Faces: Select face ids from group "department A"
Trigger type: Select "When detect"
Action type: Select "Trigger GPIOs"
GPIO#: Select number 4
Check "Always OFF"
Trigger duration(s): 3
Min duration btw trig(s): 0.01

ii) Script 2:

Name: Sound buzzer (Department A)
Description: ON GPIO 5 for 1 second when detect authorised personnels.
Faces: Select face ids from group "department A"
Trigger type: Select "When detect"
Action type: Select "Trigger GPIOs"
GPIO#: Select number 5
Check "Always OFF"
Trigger duration(s): 1
Min duration btw trig(s): 0.01

iii) Script 3:

Name: Send email (Department A)
Description: Send email to HR manager when detect unauthorised persons.
Faces: Select face ids from group "department A"
Trigger type: Select "When detect others" (To detect unauthorised persons)
Action type: Select "Send Emails"
Min duration btw trig(s): 5 (Allow some gaps for smtp server to response)
Server address: smtp.gmail.com (Using google smtp service)
Server port: 587
Email password: abcd1234
Sender address: departmentA@gmail.com
Receiver addresses: hr_manager@company.com
Subject: Unauthorised person detection alert.
Body: Unauthorised person detection at department A. Please see attachment.



BY TENSORFACTORY

Check "Attach Image"

Max send per period: 20

Period duration(s): 600 (Every 10 minutes only allow maximum 20 emails.

Step 4) Creating program (following instructions from section 4.3.3)

Name: Door Security System (Department A)

Description: Manage door unlock and unauthorised persons detection.

Scripts: Select scripts

"Unlock door (Department A)",
" Sound buzzer (Department A)" and
" Send email (Department A)"

Step 5) Upload program to PiFaceCam-Client device.

- a) Search the device in all network devices list, click on the device to open the control panel.
- b) The default name is "CAM001" and we would like to rename it to "CAM_DEPT_A".
- c) Since this is for door entrance and we would like to reduce the chances of wrong identification as much as possible and since the camera is mount in front of the door, the camera will have good angle of the person's face we can reduce the sensitivity level from 4 to 3.
- d) By default, the PiFaceCam-Client device is not passkey protected, we will on the passkey protection and provide a "New passkey".
(Note: Once ON, the passkey will be automatically stored in this PiFaceCam-App. In event if the passkey was erased, you can key-in the password manually at the "PassKey" edit box.)
- e) We are using Pi Camera, so we choose PICAM for camera type and use the recommended 640x480 camera size and 30 FPS.
- f) Click on the pencil icon next to program and select "Door Security System (Department A)".
- g) Click the green upload button at the bottom right to upload the setting and program to the connected PiFaceCam-Client. The PiFaceCam-Client will start executing the instructions in the program.

4.2 Case 2: Remote controlling and displaying of ids through JSON server.

Expanding from case 1, we would like to have our security office to be alerted whenever the camera detects a person. If it is an authorized person, green LED should light up and red LED if it is an unauthorized person. We also want the id of all the persons in the camera to be displayed.

ESP8266 JSON server

We will use a ESP8266 module (ESP07) as a JSON server which will receive information from PiFaceCam-Client. It will control 2 LEDs and display ids on a 20x4 hd44780 LCD (with I2C). The Arduino sketch for this can be found at github "ESP8266_JSON_server_V02/ESP8266_JSON_server_V02.ino".



BY TENSORFACTORY

This sketch uses

- "ArduinoJson" library by Benoit Blanchon for handling of JSON.
- "hd44780" library by Bill Perry to control LCD via I2C.
- "WiFiManager" library by Tzapu to manage network credentials (SSID and password).
- "TimeLib" library by Paul Stoffregen to convert Unix epoch time to standard format.

PiFaceCam-Client

As in case 1, we will create 2 scripts for this. One to send a key-value 1 to JSON server when detect authorized personnels and the other to send a key-value 2 when detect unauthorized personnels. ESP8266 server will receive a JSON packet with all these information and turn ON/OFF the corresponding LED as per the key_value. It will also display all (max 4) ids on the 20 x 4 matrix LCD.

```
{  "device_id": device_id,
  "num_of_faces": num_of_faces,
  "faceID_list": faceID_list,
  "key_value": key_value,
  "time_stamp": time_stamp }
```

i) Script 1:

Name: JSON key-value 1 (Department A)

Description: Send key-value 1 to JSON server when detect authorised personnels.

Faces: Select face ids from group "department A"

Trigger type: Select "When detect"

Action type: Select "Upload JSON"

Min duration btw trig(s): 0.1 (A small value as response of ESP8266 server is relatively fast).

Server address: IP address of the ESP8266 server.

Server port: Port number of the ESP8266 server.

Max send per period: 100 (A large value as response of ESP8266 server is relatively fast).

Period duration(s):1

Key value: 1

ii) Script 2:

Name: JSON key-value 2 (Department A)

Description: Send key-value 2 to JSON server when detect unauthorised personnels.

Faces: Select face ids from group "department A"

Trigger type: Select "When detect others" (To detect unauthorised persons)

Action type: Select "Upload JSON"

Min duration btw trig(s): 0.1

Server address: IP address of the ESP8266 server.

Server port: Port number of the ESP8266 server.

Max send per period: 100



BY TENSORFACTORY

Period duration(s):1

Key value: 2

Append these 2 new scripts to the program in case 1, upload the program to PiFaceCam-client, it will start sending JSON packets to the ESP8266 server.