

SELOCO,Inc.

# 사무실 출입 관제 시스템

aGates/aDoors

인턴 이우림 송진호

2021-2-26

## 목차

<b>1. 목표</b>	<b>3</b>
<b>2. 시나리오 및 구성도</b>	<b>3</b>
2.1. 시스템 시나리오	3
2.2. 시스템 구성도	3
2.2.1. 소프트웨어	4
2.2.2. 하드웨어	5
2.3. flow chart	5
<b>3. 개발 환경</b>	<b>4</b>
3.1. Jetson Nano 환경 설정	6
3.1.1. 환경설정 과정	6
3.1.2. 전원 연결	7
3.2. 메모리 환경 늘리기	9
3.3. OpenCV 설치	9
3.4. Python path 변경	12
<b>4. 시스템 개발</b>	<b>12</b>
4.1 Jetson Nano 와 Raspberry Pi Cam 연결	12
4.1.1. CSI-Camera 설치	13
4.2 센서 연결	14
4.2.1. PIR 센서	14
4.2.1. 초음파 센서	15
4.3 얼굴 인식 프로그램 개발	17
4.3.1. Harr-cascade algorithm 소개	17
4.3.2. 얼굴 인식 프로그램 기능	17
4.4 flask 를 이용한 관리자 페이지 구축	19

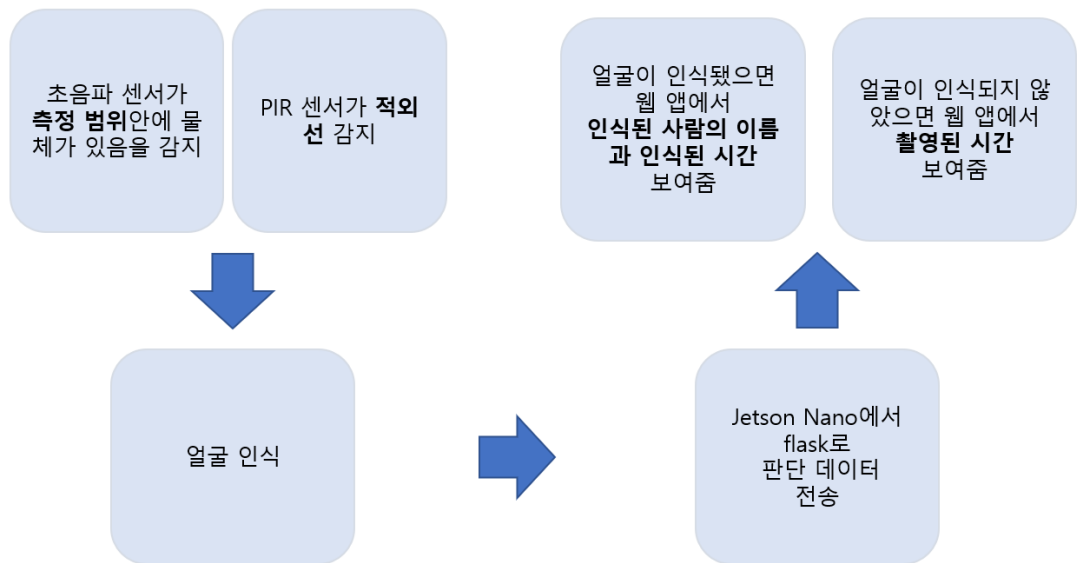
4.4.1. 관리자 페이지 소개 .....	19
<b>5. 향후 계획 .....</b>	<b>20</b>
5.1 문제점 및 개선사항 .....	20
<b>6. 팀원 담당 업무 .....</b>	<b>20</b>
<b>7. Source Code.....</b>	<b>21</b>

## 1. 목표

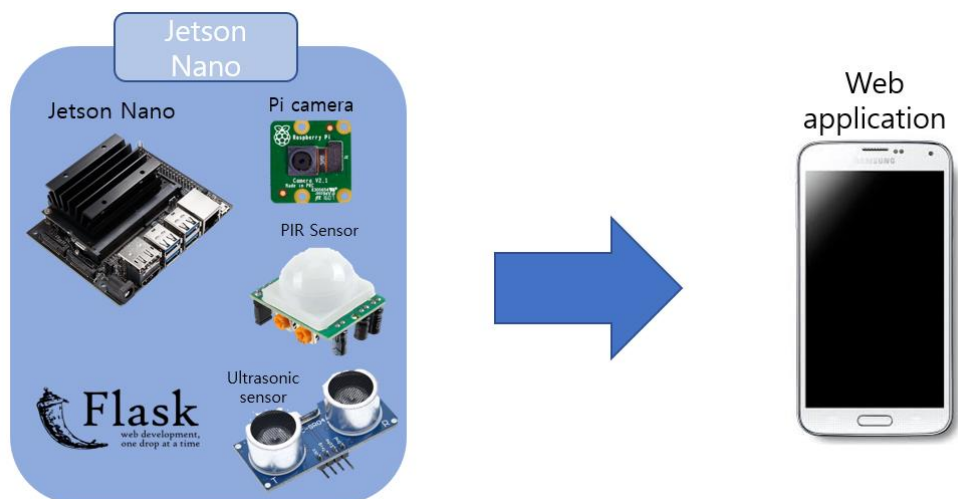
PIR 센서와 초음파 센서를 통해 물체를 감지하면 사진을 촬영하여 flask 서버로 전송하게 되고, 등록된 사람과 그렇지 않은 사람을 구별해서 테이블로 저장한다. 사용자는 웹 애플리케이션 UI를 통해서 결과를 확인할 수 있다.

## 2. 시나리오 및 구성도

### 2.1 시스템 시나리오



### 2.2 시스템 구성도



### 2.2.1 소프트웨어

- Python



파이썬은 플랫폼에 독립적이며 인터프리터식, 객체지향적, 동적 타이핑 대화형 언어이다. 비영리의 파이썬 소프트웨어 재단이 관리하는 개방형, 공동체 기반 개발 모델을 가지고 있다. C언어로 구현된 사이썬 구현이 사실상의 표준이다.

- Flask



플라스크는 파이썬으로 작성된 마이크로 웹 프레임워크의 하나로, Werkzeug 툴킷과 jinja2 템플릿 엔진에 기반을 둔다. 플라스크는 특별한 도구나 라이브러리가 필요 없기 때문에 마이크로 프레임워크라 부른다. 데이터 베이스 추상화 계층, 양식 유효성 확인, 기타 기존의 서드파티 라이브러리가 공통 기능을 제공하는 구성 요소가 없다. 그러나 플라스크는 애플리케이션 기능을 추가할 수 있는 확장 기능을 지원한다.

- OpenCV



OpenCV(Open Source Computer Vision)은 실시간 컴퓨터 비전을 목적으로 한 프로

그래밍 라이브러리이고 실시간 이미지 프로세싱에 중점을 둔 라이브러리이다. 이 라이브러리는 윈도우, 리눅스 등에서 사용 가능한 크로스 플랫폼이며 오픈소스 BSD 허가하에서 무료로 사용할 수 있다. OpenCV는 TensorFlow, Torch/PyTorch 및 Caffe의 딥러닝 프레임워크를 지원한다.

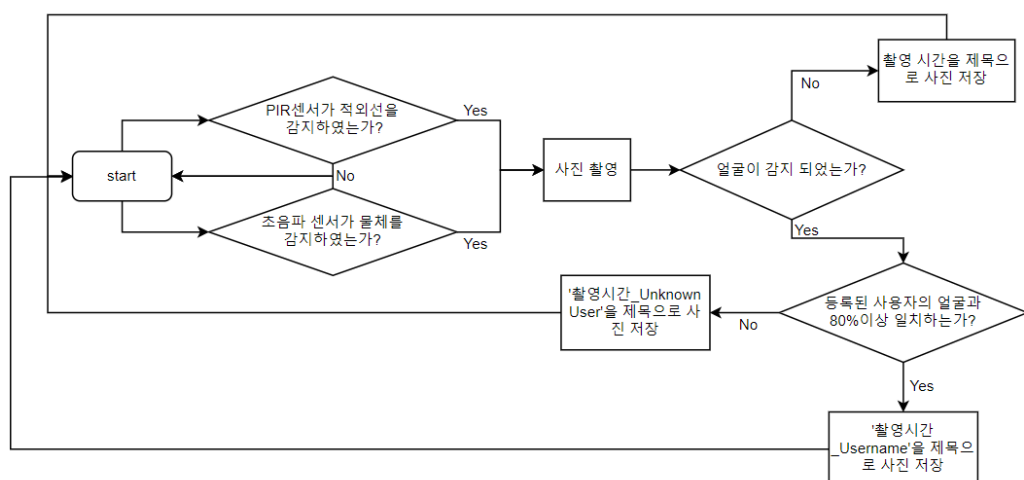
## 2.2.2 하드웨어

- Jetson Nano



Jetson Nano는 엔비디아의 임베디드 컴퓨팅 보드의 시리즈로 젯슨 TK1, TX1, TX2 모델 모두 ARM 아키텍처 중앙처리장치(CPU)를 포함한 엔비디아의 테그라 프로세서를 장착하고 있다. 젯슨은 저전력 시스템이며 기계 학습 애플리케이션을 가속하도록 설계되어 있다.

## 2.3 FLOW CHART



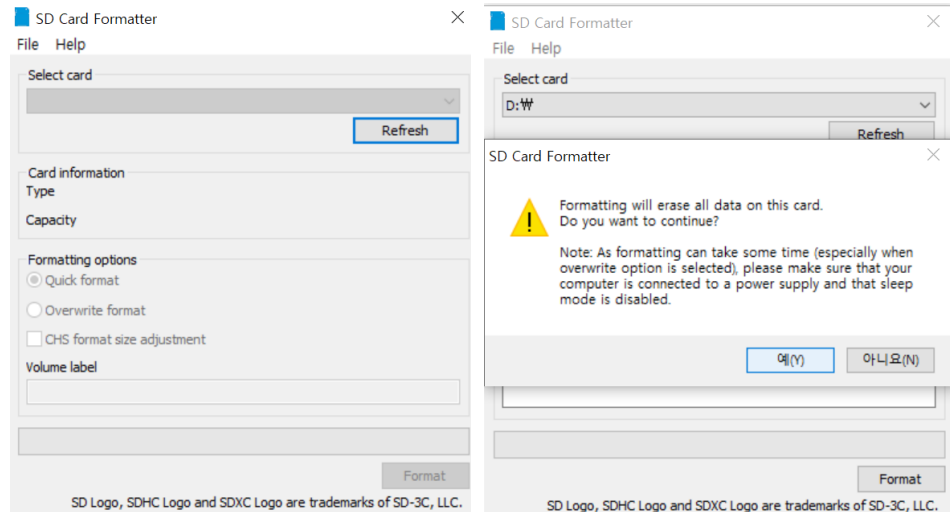
### 3. 개발 환경

개발환경	구분	상세내용
S/W 개발환경	OS	Ubuntu 18.04
	개발환경(IDE)	VS Code, Jetson Nano
	개발도구	OpenCV 3.4
	개발언어	Python3.6
H/W 개발환경	디바이스	Jetson Nano
	언어	Python3

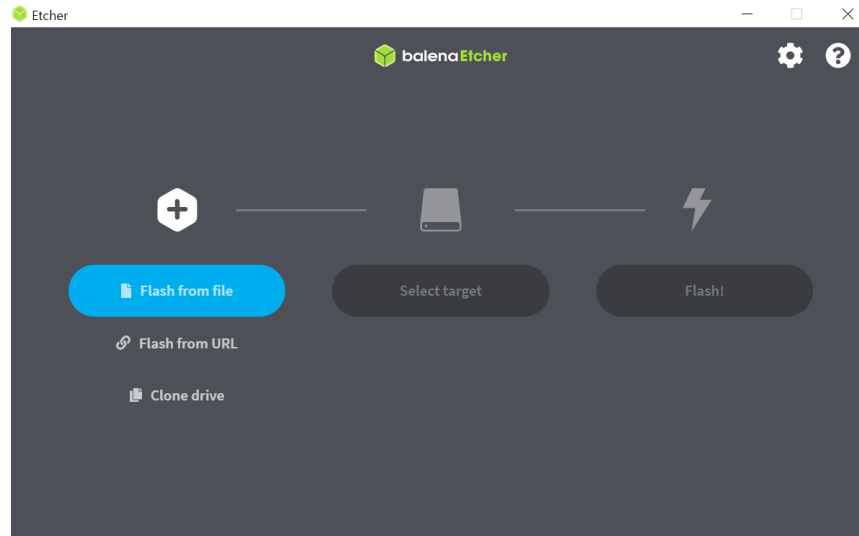
#### 3.1 JETSON NANO 환경설정

##### 3.1.1 환경설정 과정

- SD card format
  - SD card formatter 다운로드 : : [SD Memory Card Formatter for Windows Download - SD Association \(sdcard.org\)](https://www.sdcard.org/downloads/formatter_for_windows/)



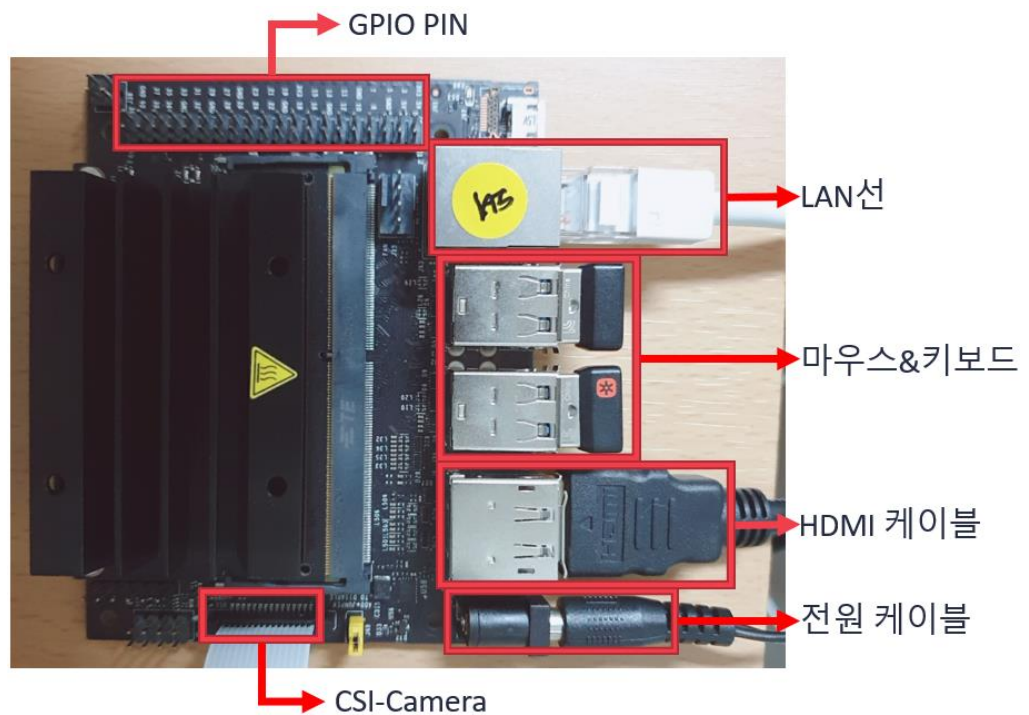
1. SD card 드라이브를 선택한 후 Quick format을 선택
  2. Volume label을 비워두고 format을 클릭한 후 '예' 클릭
- 이미지(JetPack)/Etcher 다운로드 : : <https://developer.nvidia.com/jetson-nano-sd-card-image-r3231> / <https://www.balena.io/etcher>



1. PC에 micro SD card 삽입
2. 이미지 압축 파일 선택 후 Select target에서 sd card 선택

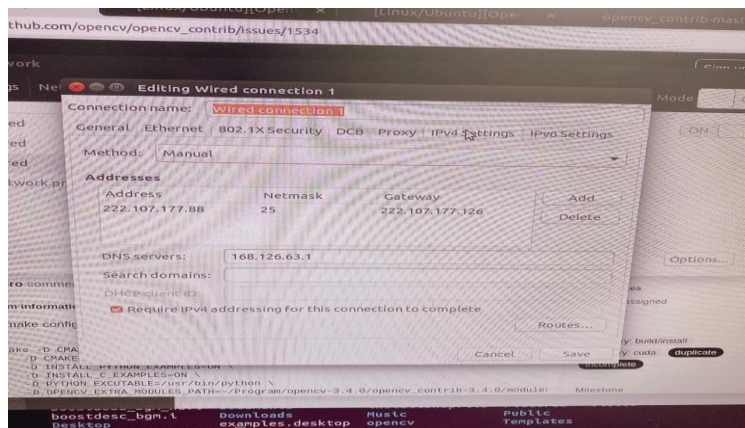
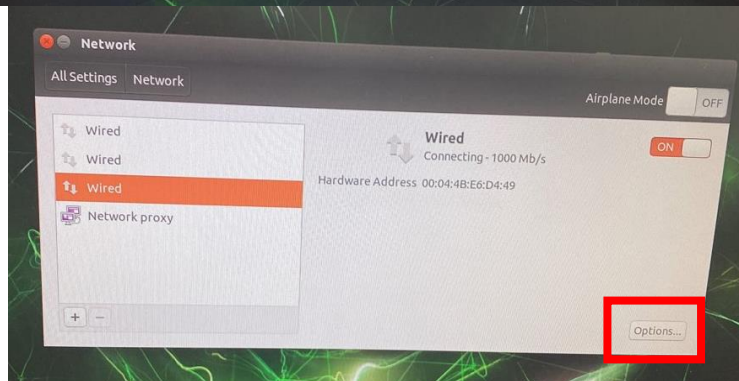
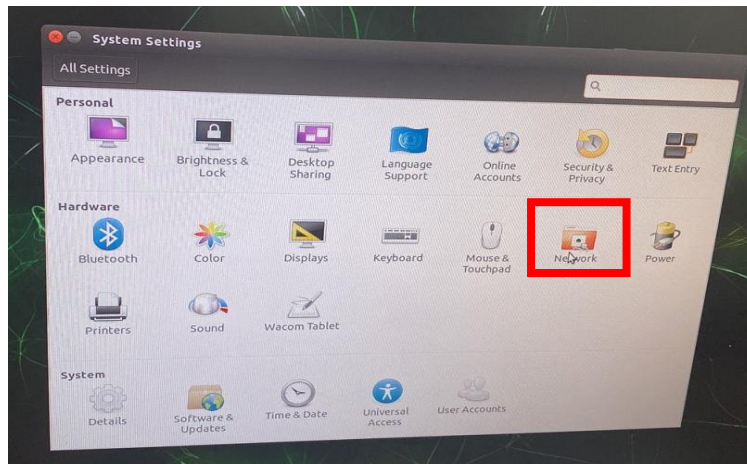
### 3.1.2 전원 연결

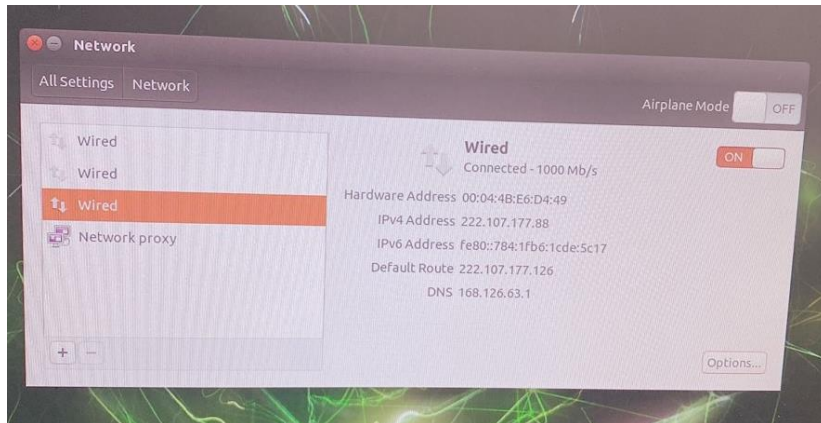
- Jetson Nano 세팅





- 네트워크 세팅





## 3.2 메모리 환경 늘리기

- OpenCV 설치 과정 중 Jetson Nano가 작동되지 않을 수 있어 가상메모리 (swapfile)를 통해 용량을 늘려준다.

```
$ git clone https://github.com/JetsonHacksNano/installSwapfile
$ cd installSwapfile
$ ./installSwapfile.sh
```

## 3.3 OPEN CV 설치

- Ubuntu 18.04에 기본(Default) OpenCV 제거

```
$ sudo apt-get remove libopencv*
$ sudo apt-get autoremove
$ sudo find /usr/local/ -name "*opencv*" -exec rm {} \;
```

- Prerequisites 설치

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install build-essential cmake unzip pkg-config
```

- Libraries 설치

```
$ sudo apt-get install libjpeg-dev libpng-dev libtiff-dev
$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-
dev libv4l-dev v4lutils libxvidcore-dev libx264-dev libxine2-dev
$ sudo apt-get install libgstreamer1.0-dev libgstreamer-plugins-
base1.0-dev
$ sudo apt-get install libgtk-3-dev
$ sudo apt-get install mesa-utils libgl1-mesa-dri libgtkgl2.0-dev
libgtkglext1-dev
$ sudo apt-get install libatlas-base-dev gfortran libeigen3-dev
cmake unzip pkg-config
```

- Python 설치

```
$ sudo apt-get install python2.7-dev python3-dev python-numpy
python3-numpy
```

- OpenCV 3.4.0 다운로드

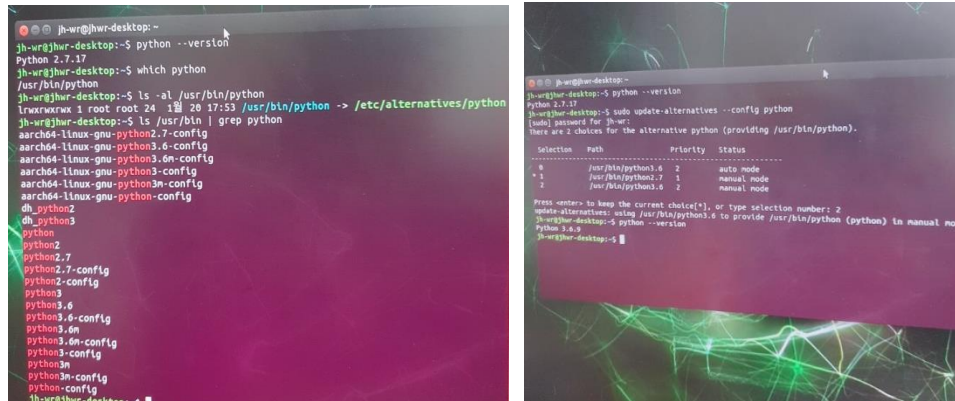
```
$ mkdir opencv
$ cd opencv
$ wget -O opencv.zip
https://github.com/opencv/opencv/archive/3.4.0.zip
$ wget -O opencv_contrib.zip
https://github.com/opencv/opencv\_contrib/archive/3.4.0.zip
$ unzip opencv.zip
$ unzip opencv_contrib.zip
```

- Build & install OpenCV

```
$ cd opencv-3.4.0
$ mkdir build
$ cd build
$ cmake -D CMAKE_BUILD_TYPE=RELEASE ₩
-D CMAKE_INSTALL_PREFIX=/usr/local ₩
-D WITH_TBB=OFF ₩ -D WITH_IPP=OFF ₩
-D WITH_1394=OFF ₩
-D BUILD_WITH_DEBUG_INFO=OFF ₩
-D BUILD_DOCS=OFF ₩
-D INSTALL_C_EXAMPLES=ON ₩
-D INSTALL_PYTHON_EXAMPLES=ON ₩
-D BUILD_EXAMPLES=OFF ₩
-D BUILD_TESTS=OFF ₩
-D BUILD_PERF_TESTS=OFF ₩
-D WITH_QT=OFF ₩ -D WITH_GTK=ON ₩
-D WITH_OPENGL=ON ₩
-D OPENCV_EXTRA_MODULES_PATH=../opencv_contrib-
3.4.0/modules ₩
-D WITH_V4L=ON ₩
-D WITH_FFMPEG=ON ₩
-D WITH_XINE=ON ₩
-D BUILD_NEW_PYTHON_SUPPORT=ON ₩
-D PYTHON2_INCLUDE_DIR=/usr/include/python2.7 ₩
-D
PYTHON2_NUMPY_INCLUDE_DIRS=/usr/lib/python2.7/distpackag
es/numpy/core/include/ ₩
-D PYTHON2_PACKAGES_PATH=/usr/lib/python2.7/dist-packages
₩
-D PYTHON2_LIBRARY=/usr/lib/x86_64-linux-gnu/libpython2.7.so
₩
-D PYTHON3_INCLUDE_DIR=/usr/include/python3.6m ₩
-D
PYTHON3_NUMPY_INCLUDE_DIRS=/usr/lib/python3/distpackages
/numpy/core/include/ ₩
-D PYTHON3_PACKAGES_PATH=/usr/lib/python3/dist-packages ₩
-D PYTHON3_LIBRARY=/usr/lib/x86_64-linux-
gnu/libpython3.6m.so ₩
../
$ make -j4
```

## 3.4 PYTHON PATH 변경

- 기본적으로 Jetson Nano에 Python이 2.7버전으로 설치되어 있으나 얼굴 인식 과정에서 Python 3.6 버전이 필요하므로 path를 변경해 주어야 한다.



The first screenshot shows a terminal window with the following commands and output:

```
jh-wr@jhwr-desktop:~$ python --version
Python 2.7.17
jh-wr@jhwr-desktop:~$ which python
/usr/bin/python
jh-wr@jhwr-desktop:~$ ls -al /usr/bin/python
lrwxrwxrwx 1 root root 24 1월 20 17:53 /usr/bin/python -> /etc/alternatives/python
jh-wr@jhwr-desktop:~$ ls /usr/bin | grep python
aarch64-linux-gnu-python2.7-config
aarch64-linux-gnu-python3.6-config
aarch64-linux-gnu-python3-config
aarch64-linux-gnu-python3m-config
dh_python2
python
python2
python2.7
python2.7-config
python2-config
python3
python3.6
python3.6-config
python3.6m-config
python3-config
python3m
python3m-config
python-config
jh-wr@jhwr-desktop:~$
```

The second screenshot shows the output of the command `sudo update-alternatives --config python`:

```
jh-wr@jhwr-desktop:~$ sudo update-alternatives --config python
[sudo] password for jh-wr:
There are 2 choices for the alternative python (providing /usr/bin/python).

  Selection Path Priority Status
  -----
* 0          /usr/bin/python3.6 2      auto mode
  1          /usr/bin/python2.7 1      manual mode
  2          /usr/bin/python3.6 2      manual mode

Press <enter> to keep the current choice[*], or type selection number: 2
update-alternatives: using /usr/bin/python3.6 to provide /usr/bin/python (python) in manual mode
jh-wr@jhwr-desktop:~$ python --version
Python 3.6.9
jh-wr@jhwr-desktop:~$
```

- ✓ `python -version` : 파이썬 버전 확인
- ✓ `which python` : 파이썬이 설치된 위치 확인
- ✓ `ls -a` : 숨겨진 파일 및 디렉토리까지 표시
- ✓ `ls -l` : 권한, 포함된 파일 수, 소유자, 그룹, 파일 크기 등 파일의 자세한 내용 표시
- ✓ `ls '위치' | grep python` : '위치'에 존재하면서, 파일명에 python이 들어가는 파일 목록 전체 출력
- ✓ `sudo update-alternatives -config python` : 사용할 파이썬의 버전을 설정

## 4. 시스템 개발

### 4.1 JETSON NANO와 라즈베리 파이 캠 연결

- Raspberry pi cam v2(800만 화소)만 Jetson Nano와 호환이 가능하므로 Raspberry pi cam의 버전을 확인해야 한다.

- 

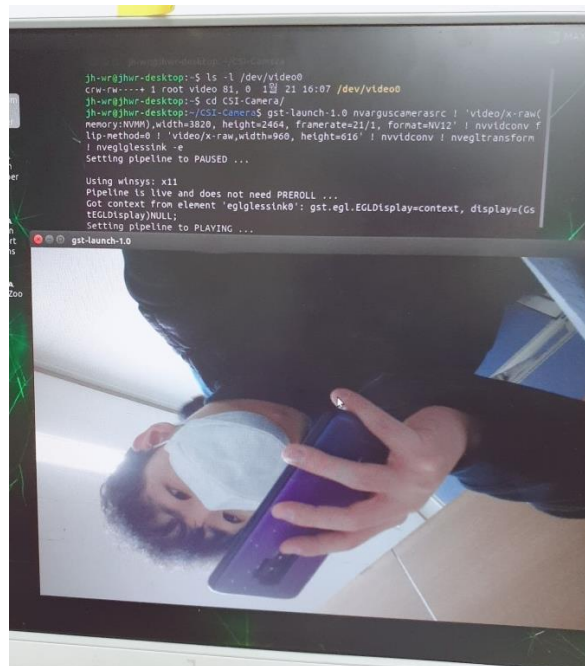
- CSI-Camera는 Gstreamer라는 함수를 사용해 Jetson Nano에서 pi cam을 사용해 볼 수 있도록 하는 오픈소스이다.(참조 : [GitHub - JetsonHacksNano/CSI-Camera: Simple example of using a CSI-Camera \(like the Raspberry Pi Version 2 camera\) with the NVIDIA Jetson Nano Developer Kit](https://github.com/JetsonHacksNano/CSI-Camera:Simple-example-of-using-a-CSI-Camera-(like-the-Raspberry-Pi-Version-2-camera)-with-the-NVIDIA-Jetson-Nano-Developer-Kit))
- 터미널을 실행시켜 명령어를 입력한다.

페이지 13 | 33

- ls -l /dev/video0 명령어를 입력하면 Jetson Nano가 Raspberry Pi Cam을 잘 인식 했는지 알 수 있다.

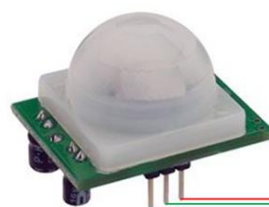
```
jh-wr@jhwr-desktop: ~/CSI-Camera
jh-wr@jhwr-desktop:~$ ls -l /dev/video0
crw-rw----+ 1 root video 81, 0 1월 21 16:07 /dev/video0
jh-wr@jhwr-desktop:~$ cd CSI-Camera/
jh-wr@jhwr-desktop:~/CSI-Camera$ gst-launch-1.0 nvarguscamerasrc ! 'video/x-raw(memory:NVMM),width=3820,height=2464,framerate=21/1,format=NV12' ! nvvidconv f
lip-method=0 ! 'video/x-raw,width=960,height=616' ! nvvidconv ! nvegltransform
! nvegllessink -e
```

- Raspberry Pi Cam이 정상적으로 출력된 모습



## 4.2 센서 연결

### 4.2.1 PIR 센서



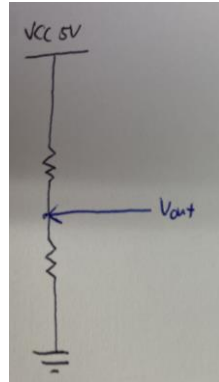
검은 선 : GND  
 녹색 선 : OUTPUT  
 붉은 선 : VCC(3.3V)

Sysfs	Name	Pin	Pin	Name	Sysfs
	3.3V DC	1	2	5V DC	
	IO_2_SDA	3	4	5V DC	
	IO_2_SCL	5	6	GND	
gpio216	AUDIO_MCLK	7	8	UART_2_TX	
	GND	9	10	UART_2_RX	
gpio0	UART_2_RTS	11	12	IO_4_CLK	gpio79
gpio14	SPI_1_SCK	13	14	GND	
gpio194	LCD_TE	15	16	SPI_2_CS1	gpio232
	3.3V DC	17	18	SPI_2_CS0	gpio15
gpio16	SPI_1_MOSI	19	20	GND	
gpio17	SPI_1_MISO	21	22	SPI_2_MISO	gpio11
gpio18	SPI_1_SCK	23	24	SPI_2_CS0	gpio19
	GND	25	26	SPI_2_CS1	gpio20
	IO_2_SDA	27	28	IO_1_SCL	
gpio149	CAM_AF_EN	29	30	GND	
gpio200	GPIO_P20	31	32	LCD_BL_PWM	gpio168
gpio38	GPIO_P66	33	34	GND	
gpio76	IO_4_LRCK	35	36	UART_2_CTS	gpio51
gpio12	SPI_2_MOSI	37	38	IO_4_SDOIN	gpio77
	GND	39	40	IO_4_SDOOUT	gpio78



#### 4.2.2 초음파 센서

- 초음파 센서의 Echo핀의 경우 5V의 전압을 사용하는데, 이를 GPIO핀에 직접적으로 연결할 경우 Jetson Nano 보드가 망가질 수 있어 3.3V로 전압을 낮춰주어야 한다.



$$V_{out} = VCC * \frac{R_2}{R_1 + R_2}$$

$$3.3 = 5 * \frac{R_2}{R_1 + R_2}$$

$$\frac{3.3}{5} = \frac{R_2}{R_1 + R_2}$$

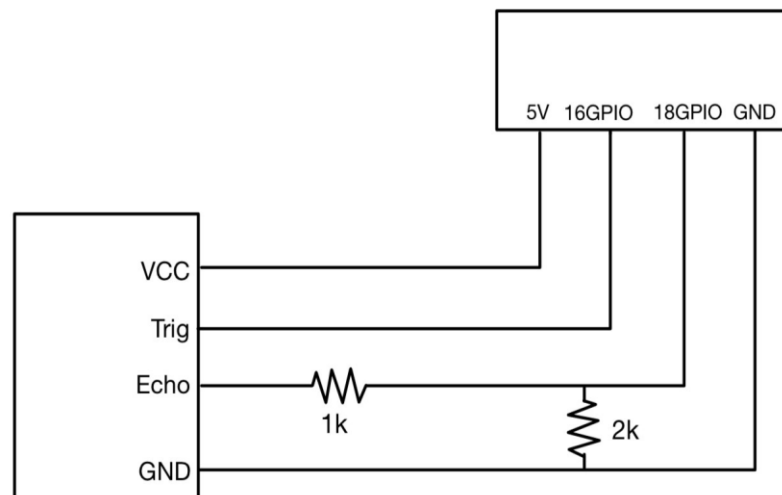
$$5R_2 = 3.3R_1 + 3.3R_2$$

$$3.3R_1 = 1.7R_2$$

$$R_2 \cong 2R_1$$

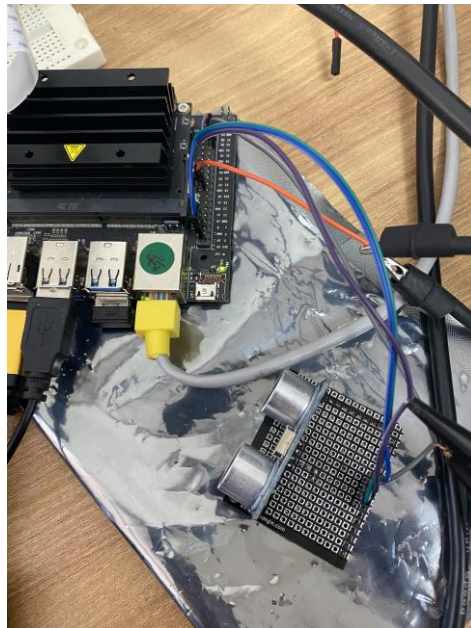
$$R_1 = 1k\Omega$$

$$R_2 = 2k\Omega$$

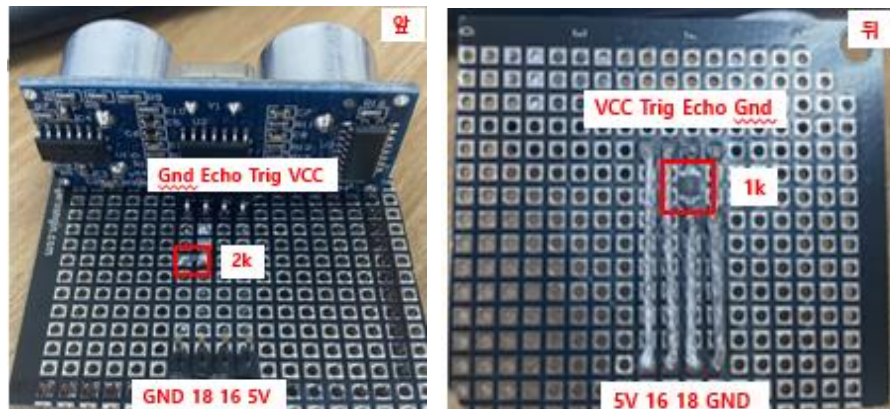




- 초음파 센서와 Jetson Nano를 연결한 모습



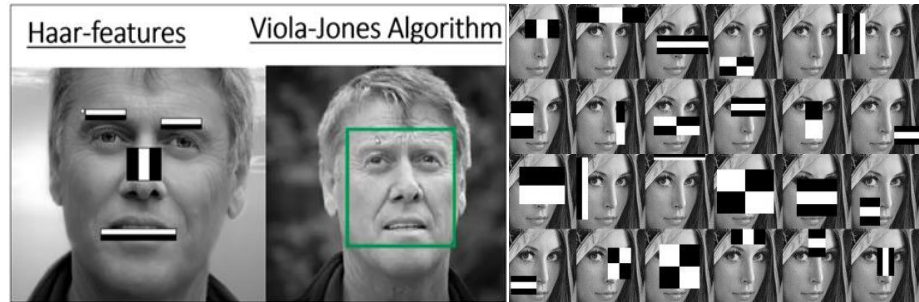
- Echo 핀의 출력전압을 5V에서 3.3V로 낮춰주기 위해 1k $\Omega$ 과 2k $\Omega$ 의 저항을 달아 납땜해준 모습



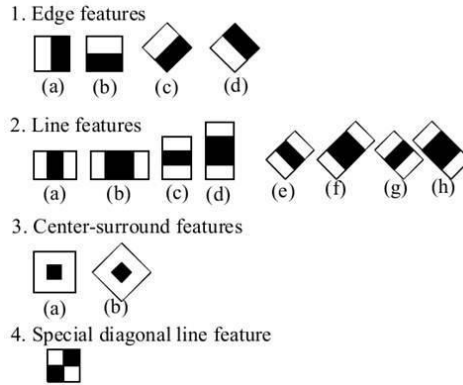
## 4.3 얼굴 인식 프로그램 개발

### 4.3.1 HAAR CASCADE ALGORITHM 소개

- Harr-cascade algorithm : 머신러닝 기반의 오브젝트 검출 알고리즘



- 2001년 논문 'Rapid Object Detection using a Boosted Cascade of Simple Features'에서 Paul Viola와 Michael Jones가 제안한 특징을 기반으로 비디오 또는 이미지에서 오브젝트를 검출하기 위해서 사용되는 알고리즘입니다.
  - 직사각형 영역으로 구성되는 특징을 사용하기 때문에 픽셀을 직접 사용할 때 보다 동작속도가 빠릅니다.
  - Haar Cascade Algorithm은 '**Haar Feature Selection**', '**Creating Integral Images**', '**Adaboost Training**', '**Cascading Classifiers**'의 4단계로 구성됩니다.
- Haar Feature Selection
    - 첫번째 단계는 이미지 전체를 스캔하여 Haar의 특징을 계산합니다.
    - Haar의 특징은 이미지를 스캔하면서 위치를 이동시키는 인접한 직사각형들의 영역 내에 있는 픽셀의 합의 차이를 이용하는 방법으로, 사각 영역 내부의 픽셀들을 빠르게 더하기 위해 적분 이미지를 사용합니다.
    - 가로 방향으로 검은색 사각 영역과 흰색 사각영역이 있는 특징의 경우에는 코와 뺨보다 눈 부분이 더 어둡다는 특성을 사용합니다.
    - 세로 방향으로 흰색 사각영역이 있고 좌우에 검은색 사각 영역이 있는 특징의 경우에는 중앙에 있는 코보다 양쪽에 있는 눈 부분이 더 어둡다는 특성을 사용합니다.



- Creating Integral Images

- Haar의 특징을 계산하려면 검은색 사각형과 흰색 사각형 아래에 있는 픽셀의 합을 구해야 하는데, 이 과정을 빠르게 처리하기 위해서 적분 이미지를 사용합니다. 따라서 이미지의 크더라도 지정한 영역의 픽셀의 합을 빠르게 구할 수 있습니다.

- Adaboost Training(Adaptive Boosting Training)

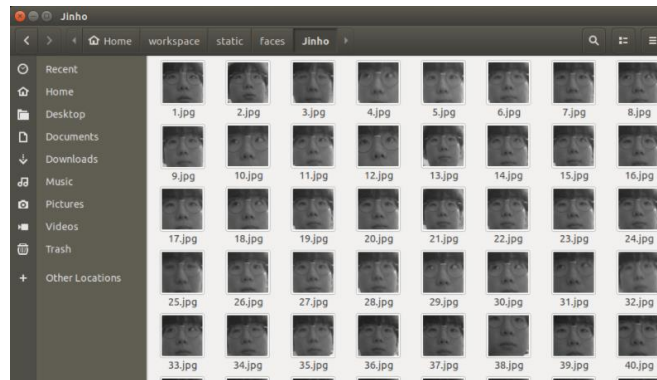
- 전 단계에서 선택한 Haar 특징을 사용하여 계산하면 160000개 이상의 특징이 검출됩니다. 얼굴 검출에 도움이 되는 특징을 추출하는 과정에서 처리 성능을 향상시키기 위한 방법으로 Adaboost를 사용합니다.
- 따라서 Adaboost를 통해서 Classifier가 생성되면 160000개 이상의 특징은 6000개의 특징으로 좁혀지게 됩니다.

- Cascading Classifiers

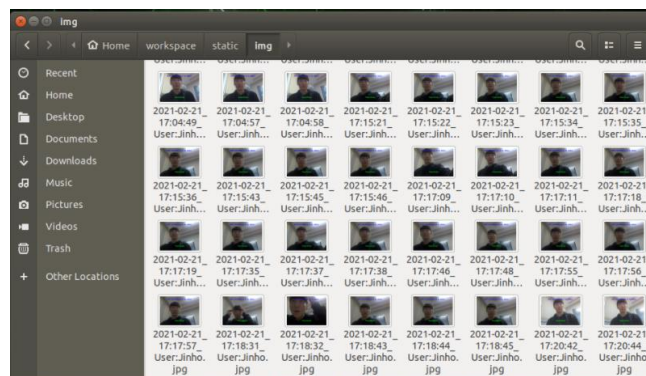
- 최종적으로 얼굴을 검출하기 위해서는 이미지를 준비하여 24x24 크기의 윈도우를 사용하여 6000개의 Haar 특징을 적용하는 과정을 거치게 되는데 이미지가 차지하는 대부분의 영역은 얼굴이 없기 때문에 계산량이 많아져서 비효율적입니다.
- 따라서 현재 윈도우가 있는 영역이 얼굴 영역인지를 단계별로 체크하여 계산량을 줄이는 과정을 거치게 되는데, 이를 Cascade Classifier라고 합니다.
- 윈도우가 이미지의 특정 영역으로 이동할 때마다 6000개의 특징을 모두 적용하지 않고 여러 단계의 그룹으로 묶어서 사용하는 방법입니다.
- 낮은 단계에서는 짧은 시간 안에 얼굴 영역인지를 판단하게 되며, 상위 단계로 올라갈수록 시간이 오래 걸리는 연산을 수행하게 됩니다.

### 4.3.2 얼굴인식 프로그램 기능

- Raspberry Pi Cam을 통해 촬영된 사진을 얼굴 부분만 crop하여 얼굴을 학습한다.(학습 데이터 사진 첨부)



- PIR 센서 및 초음파 센서로 가까이 있는 물체를 감지하면 Raspberry Pi Cam으로 얼굴을 촬영해서 기존 사용자의 데이터와 일치하는지 판단한다.
- 얼굴이 인식된 경우에는 카메라를 통해서 보여지는 화면을 촬영하게 되며, '날짜\_시간\_사용자(혹은 Unknown).jpg' 형식으로 이미지를 저장합니다.



- 기존 사용자의 데이터와 80% 이상의 일치율을 보이면, 얼굴 하단에 일치하는 사용자의 이름을 화면에 녹색 글씨로 출력한다



- 일치율이 80% 미만이면 외부인으로 판단하여 얼굴 하단에 'Unknown'을 붉은 글씨로 출력한다.



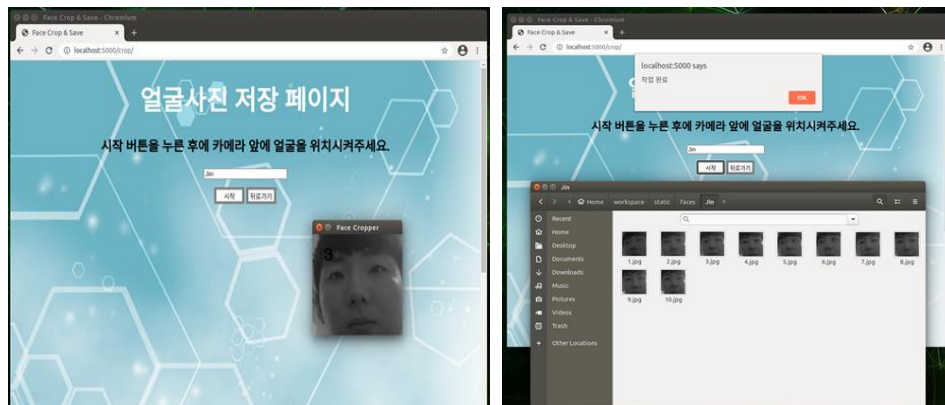
## 4.4 FLASK를 이용한 관리자 페이지 구축

### 4.4.1 관리자 페이지 소개

- 메인 페이지

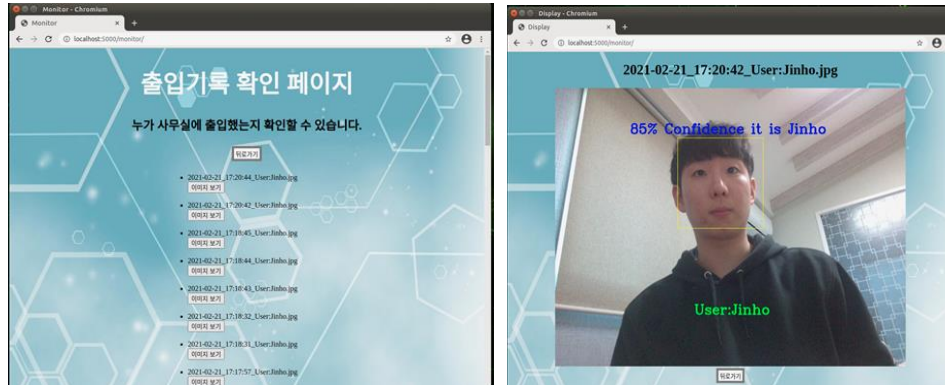


- 얼굴 사진 저장 페이지: 얼굴인식 프로그램에 사용되는 얼굴 이미지 데이터를 만들어주는 페이지





- 촬영 이미지 확인 페이지: 얼굴인식 프로그램을 통해서 촬영된 이미지는 ‘날짜\_시간\_이름.jpg’ 형식으로 저장되며, 위에서부터 최근 날짜 순으로 파일 이름이 표시된다. 파일 이름 하단에 위치한 각각의 ‘이미지 보기’ 버튼을 누르면 촬영된 이미지를 볼 수 있다.



## 5. 향후 계획

### 5.1 문제점 및 개선사항

- 현재 얼굴인식 프로그램은 harr-cascade 알고리즘을 사용해 만들었는데 좀 더 나은 정확도를 위해 다른 알고리즘이나 얼굴인식 모듈을 사용해야 한다.
- 초음파 센서의 Echo 핀에서 출력되는 전압을 5V에서 3.3V로 낮추는 것에 성공하였으나 소스코드 상의 문제로 작동되지 않아 조금 더 연구하여 수정이 필요하다.
- 관리자 페이지에 로그인/회원가입 기능을 추가하여 권한을 부여할 필요가 있다.
- 현재 개발된 프로그램은 harr-cascade 알고리즘을 통해 얼굴이 감지되면 촬영을 하도록 설계되어 있으므로, 사람이 아닌 사물이나 동물이 있을 때에도 센서를 통해 감지되면 촬영이 가능하도록 수정이 필요하다.

## 6. 팀원 담당 업무

이우림	송진호
1. 얼굴인식 프로그램 개발	1. 얼굴인식 프로그램 개발
2. 웹 애플리케이션 제작	2. Flask 서버 구축
3. 초음파 센서와 Jetson Nano 연결 및	3. PIR 센서와 Jetson Nano 연결 및 소스

소스코드 작성 4. 문서 작성	코드 작성 4. 문서 작성
---------------------	-------------------

## 7. SOURCE CODE

- flaskServer.py

```
import os, sys, json

from flask import Flask, request, redirect, render_template, jsonify, make_response
from face_save import gstreamer_pipeline, face_detector, face_save

app = Flask(__name__) # Flask Class 의 인스턴스 생성

@app.route('/')
@app.route('/index/')
def index():
    return render_template('index.html')

@app.route('/crop/', methods=['GET', 'POST'])
def crop():
    if request.method == 'GET':
        return render_template('crop.html')
    elif request.method == 'POST':
        name = list(request.json.values())[0]
        face_save(name)
        return jsonify("SUCCESS") # JSON Parse 해서 클라이언트에 응답

@app.route('/monitor/', methods=['GET', 'POST'])
def monitor():
    if request.method == 'GET':
        path = 'static/img/'
        files = os.listdir(path)
        files.sort() # 파일을 오름차순으로 정렬
        files.reverse() # 파일을 내림차순으로 정렬
        return render_template('monitor.html', path='../'+path, files=files)
    elif request.method == 'POST':
        img = list(request.form.keys())[0] # submit 의 name 이 지칭하는 파일명 가져오기
        return render_template('display.html', img=img)

if __name__ == "__main__":
    app.run(host='0.0.0.0', debug=True) # 로컬서버로 실행, app.run(host='0.0.0.0')으로 모든 OS 에게 접근가능하도록 설정
```

- face\_save.py(얼굴 사진 저장 프로그램)

```
import numpy as np
import cv2

from os import makedirs
from os.path import isdir

# 사진을 200x200 사이즈로 crop 후에 회색 바탕으로 바꿔서 저장
face_dirs = 'static/faces/'

face_cascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")

# Jetson Nano 에서 Pi Cam 을 사용하기 위해서 필요한 메소드(건들지 말것)
def gstreamer_pipeline(
    capture_width=3280,
    capture_height=2464,
    display_width=820,
    display_height=616,
    framerate=21,
    flip_method=0,
):
    return (
        "nvarguscamerasrc ! "
        "video/x-raw(memory:NVMM), "
        "width=(int)%d, height=(int)%d, "
        "format=(string)NV12, framerate=(fraction)%d/1 ! "
        "nvvidconv flip-method=%d ! "
        "video/x-raw, width=(int)%d, height=(int)%d, format=(string)BGRx ! "
        "videoconvert ! "
        "video/x-raw, format=(string)BGR ! appsink"
        % (
            capture_width,
            capture_height,
            framerate,
            flip_method,
            display_width,
            display_height,
        )
    )

# 얼굴을 인식했는지 판단하는 메소드
```



```

def face_detector(img):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (5,5), 0)
    faces = face_cascade.detectMultiScale(blur, 1.3, 5)

    # 얼굴을 인식하지 못하면 None 리턴(안 써주면 오류 발생)
    if faces is ():
        return None

    for(x, y, w, h) in faces:
        roi = img[y:y+h, x:x+w]
    return roi

# 얼굴을 인식하면 jpg 파일로 저장하는 메소드
def face_save(name):
    user_dirs = face_dirs+name+'/'

    # 해당 디렉토리가 존재하지 않으면 생성
    if not isdir(user_dirs):
        makedirs(user_dirs)

    cap = cv2.VideoCapture(gstreamer_pipeline(), cv2.CAP_GSTREAMER)
    if cap.isOpened():
        cv2.namedWindow("Face Cropper", cv2.WINDOW_AUTOSIZE)

        count = 1
        while cv2.getWindowProperty("Face Cropper", 0) >= 0:
            ret, img = cap.read()
            # gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            # blur = cv2.GaussianBlur(gray, (5,5), 0)
            # faces = face_cascade.detectMultiScale(blur, 1.3, 5)

            # for(x, y, w, h) in faces:
            #     cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
            #     roi_gray = gray[y : y+h, x : x+w]
            #     roi_color = img[y : y+h, x : x+w]

            # 카메라를 통해서 읽어온 이미지를 얼굴로 인식할 경우
            face = face_detector(img)
            if face is not None:
                crop = cv2.resize(face, (200, 200)) # 얼굴 사이즈에 맞게 200x200
                # 0 사이즈로 이미지 축소
                gray = cv2.cvtColor(crop, cv2.COLOR_BGR2GRAY) # 축소된 이미지를
                # 회색 바탕으로 변경

```

```

        cv2.imwrite(user_dirs+str(count)+'.jpg', gray) # 축소되고 회색
        으로 변경된 이미지를 저장

        cv2.putText(gray, str(count), (25, 50), cv2.FONT_HERSHEY_COMPL
EX, 1, (0, 0, 0), 2) # 현재 몇번째 이미지를 촬영하고 있는지 좌측 상단에 숫자로 표
시

        cv2.imshow("Face Cropper", gray)
        count += 1

        keyCode = cv2.waitKey(1) & 0xFF
        if keyCode == 27 or count > 10:
            break

        cap.release()
        cv2.destroyAllWindows()
    else:
        print("Unable to open camera")

if __name__ == "__main__":
    name = input("Name: ")
    face_save(name)

```

- face\_identification.py(얼굴 감지 및 인식을 통한 신원확인 프로그램)

```

import numpy as np
import cv2
import datetime
import Jetson.GPIO as GPIO

from os import listdir, makedirs
from os.path import isdir, isfile, join

# PIR 센서 인식을 위한 PIN 번호 설정
PIR = 7

# Jetson Nano 표면에 쓰여있는 PIN 번호를 기반으로 인식할 것임을 정의
GPIO.setmode(GPIO.BOARD)
# PIR 센서를 Input Sensor 로 정의
GPIO.setup(PIR, GPIO.IN)

# 사진을 200x200 사이즈로 crop 후에 회색 바탕으로 바꿔서 저장
face_dirs = 'static/faces/'
# 회원과 외부인을 가리지 않고 방문한 사람을 모두 촬영해서 웹으로 전송
image_dirs = 'static/img/'

```

```

face_cascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")

# Jetson Nano 에서 Pi Cam 을 사용하기 위해서 필요한 메소드(건들지 말것)
def gstreamer_pipeline(
    capture_width=3280,
    capture_height=2464,
    display_width=820,
    display_height=616,
    framerate=21,
    flip_method=0,
):
    return (
        "nvarguscamerasrc ! "
        "video/x-raw(memory:NVMM), "
        "width=(int)%d, height=(int)%d, "
        "format=(string)NV12, framerate=(fraction)%d/1 ! "
        "nvvidconv flip-method=%d ! "
        "video/x-raw, width=(int)%d, height=(int)%d, format=(string)BGRx ! "
        "videoconvert ! "
        "video/x-raw, format=(string)BGR ! appsink"
        % (
            capture_width,
            capture_height,
            framerate,
            flip_method,
            display_width,
            display_height,
        )
    )

# 각각의 모델을 트레이닝하는 메소드
def train_model(name):
    path = face_dirs+name+'/'
    onlyfiles = [f for f in listdir(path) if isfile(join(path, f))]

    training_data, labels = [], []
    for i, files in enumerate(onlyfiles):
        img = path + onlyfiles[i]
        gray = cv2.imread(img, cv2.IMREAD_GRAYSCALE)
        if gray is None:
            continue
        training_data.append(np.asarray(gray, dtype=np.uint8))
        labels.append(i)
    if len(labels) == 0:
        print("There is no data to train.")

```

```

        exit()

    labels = np.asarray(labels, dtype=np.int32)
    model = cv2.face.LBPHFaceRecognizer_create()
    model.train(np.asarray(training_data), np.asarray(labels))
    print("Model: '"+name+"' Training Complete!!!")
    return model

# 트레이닝을 위한 모든 모델을 불러오는 메소드
def train_models():
    path = face_dirs
    model_dirs = [f for f in listdir(path) if isdir(join(path, f))]

    models = {}
    for model in model_dirs:
        result = train_model(model)
        if result is None:
            continue
        models[model] = result
    return models

# 얼굴을 인식했는지 판단하는 메소드
def face_detector(img):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (5,5), 0)
    faces = face_cascade.detectMultiScale(blur, 1.3, 5)

    # 얼굴을 인식하지 못하면 None 리턴(안 써주면 오류 발생)
    if faces is ():
        return None

    # current = datetime.datetime.now()
    # current_datetime = current.strftime('%Y-%m-%d %H:%M:%S')
    # cv2.imwrite(image_dirs+current_datetime+'.jpg', img) # 얼굴을 인식하면 jpg
    # 파일로 저장

    for(x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 255), 1) # 얼굴을 감지
        # 하면 얼굴 주변에 정사각형이 그려짐
        roi = img[y:y+h, x:x+w]
        roi = cv2.resize(roi, (200, 200)) # 학습한 데이터와 비교하기 위해서 얼굴
        # 부분만 200x200 사이즈로 Crop
    return roi

# 얼굴을 인식하면 사진촬영해서 저장하고 사용자인지 외부인인지 판별하는 메소드

```

```

def face_identification(models):
    cap = cv2.VideoCapture(gstreamer_pipeline(), cv2.CAP_GSTREAMER)
    if cap.isOpened():
        cv2.namedWindow("Face Identification", cv2.WINDOW_AUTOSIZE)

        while cv2.getWindowProperty("Face Identification", 0) >= 0:
            ret, img = cap.read()
            # gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            # blur = cv2.GaussianBlur(gray, (5,5), 0)
            # faces = face_cascade.detectMultiScale(blur, 1.3, 5)

            # for(x, y, w, h) in faces:
            #     cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 255), 1)
            #     roi_gray = gray[y : y+h, x : x+w]
            #     roi_color = img[y : y+h, x : x+w]

            # PIR 센서를 통해서 생물체를 인식 및 카메라를 통해서 읽어온 이미지를 얼굴로 인식할 경우
            face = face_detector(img)
            if GPIO.input(PIR) == GPIO.HIGH and face is not None:
                face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY) # crop 한 이미지를 회색 배경으로 바꿔서 저장

                # 학습한 모델로 예측 시도
                min_score = 999
                min_score_name = ""
                for key, model in models.items():
                    result = model.predict(face)
                    if min_score > result[1]:
                        min_score = result[1]
                        min_score_name = key

                # result[1]는 신뢰도를 뜻하며, 0에 가까울수록 본인과 일치함을 의미
                if min_score < 500:
                    confidence = int(100*(1-(result[1])/300))
                    display_string = str(confidence)+'% Confidence it is '+min_score_name
                    cv2.putText(img, display_string, (175, 100), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 0, 0), 2)

                # confidence에 따라서 등록된 사용자인지 외부인인지 판별하여 이미지를 저장
                current = datetime.datetime.now()
                current_datetime = current.strftime('%Y-%m-%d_%H:%M:%S_')

```

```

        if confidence >= 80:
            cv2.putText(img, 'User:'+min_score_name, (325, 500), cv2.F
ONT_HERSHEY_COMPLEX, 1, (0, 255, 0), 2)
            cv2.imwrite(image_dirs+current_datetime+'User:'+min_score_
name+'.jpg', img) # 등록된 사용자로 인식될 경우에 대한 이미지 파일 저장
            print("User:"+min_score_name)
        else:
            cv2.putText(img, 'Unknown', (325, 500), cv2.FONT_HERSHEY_C
OMPLEX, 1, (0, 0, 255), 2)
            cv2.imwrite(image_dirs+current_datetime+'Unknown.jpg', img
) # 외부인으로 인식될 경우에 대한 이미지 파일 저장
            print("Unknown User")

    cv2.imshow("Face Identification", img)
    keyCode = cv2.waitKey(1) & 0xFF
    if keyCode == 27:
        break

    cap.release()
    cv2.destroyAllWindows()
else:
    print("Unable to open camera")

if __name__ == "__main__":
    print("Wait for training models . . .")
    models = train_models()
    face_identification(models)

```

- index.html(관리자 페이지 메인)

```

<!DOCTYPE html>
<html>
<head>
    <title>Index</title>
    <meta charset="utf-8">
</head>
<body style="background-
image: url( {{ url_for('static', filename='asset/blue.jpg')}} ); background-
position: center; background-repeat: no-repeat; height: 1100px; background-
size: 1200px 1200px">
    <center>
        <h1 style = "font-
size: 40pt; color :white;">출입관제 시스템<br>관리자 페이지</h1>
    <p>

```

```

        <input type='button' name='crop' value='얼굴 사진 저장' onclick="window.location.href='/crop/'" style = "border: 7px solid gray; padding: 5px; color: black; font-size:20pt; width:150pt; height:45pt; background-color: white;">
    </p>
    <p>
        <input type='button' class="btn" name='monitor' value='출입기록 확인' onclick="window.location.href='/monitor/'" style = "border: 7px solid gray; padding: 5px; color: black; font-size:20pt; width:150pt; height:45pt; background-color: white;">
    </p>
</center>
<!--
<a href="/test">웹사이트 test 로 이동</a>
a href 는 GET 방식만 지원하기 때문에 접근하려는 사이트가 POST 방식이면 Error
-->
</body>
</html>

```

- crop.html(얼굴 사진 저장 페이지)

```

<!DOCTYPE html>
<html>
<head>
    <title>Face Crop & Save</title>
    <meta charset="utf-8">
</head>
<body style="background-image: url( {{ url_for('static', filename='asset/blue.jpg')}} ); background-position: center; background-repeat: no-repeat; height: 1100px; background-size: 1200px 1200px">
    <center>
        <h1 style = "font-size: 40pt; color : white;">얼굴 사진 저장 페이지</h1>
        <h3 style = "font-size: 20pt">시작 버튼을 누른 후에 카메라 앞에 얼굴을 위치시켜주세요.</h3>
        <!-- <form action='/crop/' method='POST'>
            <input type='text' name='name'>
            <input type='submit' id='crop' value='시작'>
        </form> -->
        <input type="text" id="name" name="name">
    <p>
        <input type="submit" id="crop" name="crop" value="시작" style = "border: 5px solid gray; padding: 3px; color: black; font-size:10pt; width: 50pt; height: 25pt; background-color: white;">
    </p>

```

```

        <input type="button" id="back" name="back" value="뒤로가기" onclick=
k="window.location.href='/index/'" style = "border: 5px solid gray; padding: 3
px; color: black; font-size:10pt; width: 50pt; height: 25pt; background-
color: white;">
    </p>
</center>
</body>

<!-- jQuery-latest -->
<script src="https://code.jquery.com/jquery-latest.min.js"></script>

<!-- Javascript -->
<script>
    $("#crop").on("click", function() {
        var userName = $('#name').val()
        // var userName = $('#name').get(0).value;
        // var userName = document.getElementById('name').value
        // var userName = document.getElementsByName("name")[0].value
        var sendData = {
            name : userName
        }
        $.ajax({
            url : "{ url_for('crop') }",
            type : "POST",
            dataType : "json", // "json", "JSON"은 되지만 json 만 불가능
            contentType : "application/json", // 이거 안 써주면 오류
            // data : {
            //     name : userName // 이렇게 보내주면 400 오류
            // },
            data : JSON.stringify(sendData), // json 객체를 String 객체로 변환
            success:function(response) {
                alert('작업 완료')
                window.location.href = '../index'
            },
            error:function(jqXHR, exception) {
                var msg = '';
                if (jqXHR.status === 0) {
                    msg = 'Not connected. Please Verify Network.';
                } else if (jqXHR.status == 404) {
                    msg = '[404] Requested page not found.';
                } else if (jqXHR.status == 500) {
                    msg = '[500] Internal Server Error.';
                } else if (exception === 'parsererror') {
                    msg = 'Requested JSON parse failed.';
                } else if (exception === 'timeout') {

```



```

        msg = 'Time out error.';
    } else if (exception === 'abort') {
        msg = 'Ajax request aborted.';
    } else {
        msg = 'Uncaught Error.\n' + jqXHR.responseText;
    } alert(msg)
    }
    })
})
</script>
</html>

```

- monitor.html(촬영 이미지 목록 페이지)

```

<!DOCTYPE html>
<html>
<head>
    <title>Monitor</title>
    <meta charset="utf-8">
</head>
<body style="background-
image: url( {{ url_for('static', filename='asset/blue.jpg')}} ); background-
position: center; background-repeat: no-repeat; height: 1100px; background-
size: 1200px 1200px">
    <center>
        <h1 style = "font-size: 40pt; color : white;">출입기록 확인 페이지</h1>
        <h3 style = "font-
size: 20pt;">누가 사무실에 출입했는지 확인할 수 있습니다.</p>
        <input type="button" id="back" name="back" value="뒤로가기" onclick="w
indow.location.href='/index/'" style = "border: 5px solid gray; padding: 3px;
color: black; font-size:10pt; width: 50pt; height: 25pt; background-
color: white;">
    </center>
    <ul style = "width: 300px; margin-left : auto; margin-right: auto;">
        {% for file in files %}
            <li>
                {{ file }}
                <form action="/monitor/" method='POST'>
                    <input type="submit" name="{{ path }}"{{ file }}" value
="이미지 보기">
                </form>
            </li>
            <br>
        {% endfor %}
    </ul>

```

```
</body>
</html>
```

- display.html(촬영 이미지 확인 페이지)

```
<!DOCTYPE html>
<html>
<head>
  <title>Display</title>
  <meta charset="utf-8">
</head>
<body style="background-
image: url( {{ url_for('static', filename='asset/blue.jpg')}} ); background-
position: center; background-repeat: no-repeat; height: 1100px; background-
size: 1200px 1200px">
  <center>
    <h1>{{ "".join(img.split('../static/img/')) }}</h1>
    <br>
    <input type="button" id="back" name="back" value="뒤로가기" onclick="w
indow.location.href='/monitor/'" style = "border: 5px solid gray; padding: 3px
; color: black; font-size:10pt; width: 50pt; height: 25pt; background-
color: white;">
  </center>
</body>
</html>
```