**2-2. 개발내용 및 범위**

1. **파이카메라 연결 – 박세훈**

* **라즈베리파이 파이카메라 연결**

전자기기이(가) 표시된 사진

자동 생성된 설명<그림 - >파이카메라 연결

영상처리를 위해 적외선 파이카메라를 선택했다. 적외선 카메라를 선택한 이유는 야간에도 사용하기 때문이다.

**1) 카메라 연결**

 우선 센서 값을 받아서 자동으로 촬영을 하기 위해서 카메라를 연결해주어야 한다.

보드에 카메라를 연결해준다.

|  |
| --- |
| $ sudo apt-get update  $ sudo apt-get upgrade |

다음으로 라즈베리파이를 최신버전으로 업데이트를 진행해준다.

|  |
| --- |
| $ sudo raspi-config |

입력하여 카메라 사용을 enable로 바꿔준다.

1. **OpenCV 컴파일 전 필요한 패키지 설치**

OpenCV를 컴파일하는데 사용하는 것들이 포함된 패키지들을 설치한다.

**build-essential** 패키지에는 C/C++ 컴파일러와 관련 라이브러리, make 같은 라이브러리가 포함되어있다.

**cmake**는 컴파일 옵션이나 빌드된 라이브러리에 포함시킬 OpenCV 모듈 설정등을 위해 필요하다.

|  |
| --- |
| $ sudo apt-get install build-essential cmake |

**pkg-config**는 프로그램 컴파일 및 링크시 필요한 라이브러리에 대한 정보를 가져오는데 사용된다.

|  |
| --- |
| $ sudo apt-get install pkg-config |

특정 포맷의 이미지 파일을 불러오거나 기록하기 위해 필요한 패키지들이다.

|  |
| --- |
| $ sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev |

특정 코덱의 비디오 파일을 읽어오거나 기록하기 위해 필요한 패키지들이다.

|  |
| --- |
| $ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libxvidcore-dev libx264-dev libxine2-dev |

Video4Linux 패키지는 리눅스에서 실시간 비디오 캡처를 지원하기 위한 디바이스 드라이버와 API를 포함하고 있다.(하단의 VideoCapture 코드에서 사용)

|  |
| --- |
| $ sudo apt-get install libv4l-dev v4l-utils |

GStreamer는 비디오 스트리밍을 위한 라이브러리다.

|  |
| --- |
| $ sudo apt-get install libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev |

OpenCV에서는 highgui 모듈을 사용하여 자체적으로 윈도우 생성하여 이미지나 비디오들을 보여줄 수 있다.

|  |
| --- |
| $ sudo apt-get install libqt4-dev |

OpenGL 지원하기 위해 필요한 라이브러리이다.

|  |
| --- |
| $ sudo apt-get install mesa-utils libgl1-mesa-dri libqt4-opengl-dev |

OpenCV 최적화를 위해 사용되는 라이브러리들이다.

|  |
| --- |
| $ sudo apt-get install libatlas-base-dev gfortran libeigen3-dev |

python2.7-dev와 python3-dev 패키지는 OpenCV-Python 바인딩을 위해 필요한 패키지들이다.

Numpy는 매트릭스 연산등을 빠르게 처리할 수 있어서 OpenCV에서 사용된다.

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

**3)OpenCV 컴파일 및 메모리 스왑**

소스 코드를 저장할 임시 디렉토리를 생성하여 이동 후 진행한다.

|  |
| --- |
| pi@raspberrypi:~$ mkdir opencv pi@raspberrypi:~$ cd opencv pi@raspberrypi:~/opencv$ |

OpenCV 4.0.1 소스코드를 다운로드 받아 압축을 풀어준다.

|  |
| --- |
| $ wget -O opencv.zip https://github.com/opencv/opencv/archive/4.5.5.zip $ unzip opencv.zip |

opencv\_contrib(extra modules) 소스코드를 다운로드 받아 압축을 풀어준다.

SIFT, SURF 등을 사용하기 위해 필요하다.

|  |
| --- |
| $ wget -O opencv\_contrib.zip https://github.com/opencv/opencv\_contrib/archive/4.5.5.zip $ unzip opencv\_contrib.zip |

다음처럼 두 개의 디렉토리가 생성된다.

|  |
| --- |
| pi@raspberrypi:~/opencv$ ls -d \*/ opencv-4.5.5/  opencv\_contrib-4.5.5/ |

opencv-4.5.5 디렉토리로 이동하여 build 디렉토리를 생성하고 build 디렉토리로 이동한다.

컴파일은 build 디렉토리에서 이루어진다.

|  |
| --- |
| pi@raspberrypi:~/opencv$ cd opencv-4.5.5/ pi@raspberrypi:~/opencv/opencv-4.5.5$ mkdir build pi@raspberrypi:~/opencv/opencv-4.5.5$ cd build pi@raspberrypi:~/opencv/opencv-4.5.5/build$ |

컴파일 진행하기 전에 스왑(swap) 공간을 늘려줘야 멀티코어를 사용하여 컴파일시 메모리 부족으로 에러가 나지 않음.

( 참고 <https://www.pyimagesearch.com/2017/10/09/optimizing-opencv-on-the-raspberry-pi/> )

|  |
| --- |
| pi@raspberrypi:$ sudo nano /etc/dphys-swapfile |

/etc/dphys-swapfile 파일을 열어서

$ sudo nano /etc/dphys-swapfile

CONF\_SWAPSIZE  변수값을 100에서 2048로 수정한다.

Ctrl + O를 눌러 저장하고 Ctrl + X를 눌러 빠져나온다

# set size to absolute value, leaving empty (default) then uses computed value

#   you most likely don't want this, unless you have an special disk situation

# CONF\_SWAPSIZE=100

CONF\_SWAPSIZE=2048

스왑 서비스 재시작하여 변경된 설정을 반영시켜주면 스왑 크기가 대략 20배가 된다.

현재는 Swap이 100 메가.

pi@raspberrypi:~/opencv/opencv-4.5.5/build $ free

               total        used        free      shared  buff/cache   available

Mem:         3930860      179928     1751316       36172     1999616     3554424

Swap:         102396           0      102396

스왑 관련 서비스를 재시작한다.

pi@raspberrypi:~/opencv/opencv-4.5.5/build $ sudo /etc/init.d/dphys-swapfile restart

[ ok ] Restarting dphys-swapfile (via systemctl): dphys-swapfile.service.

Swap이 2048메가로 바뀐다.

pi@raspberrypi:~/opencv/opencv-4.5.5/build $ free

               total        used        free      shared  buff/cache   available

Mem:         3930860      181864      109168       36872     3639828     3551788

Swap:        2097148           0     2097148

|  |
| --- |
| pi@raspberrypi:~/opencv/opencv-4.5.5/build $ make |

컴파일 진행.

약 60분 정도 걸린다.

이제 컴파일 결과물을 설치한다.

|  |
| --- |
| pi@raspberrypi:~/opencv/opencv-4.5.5/build $ sudo make install |

/etc/dphys-swapfile 파일을 열어서

|  |
| --- |
| $ sudo nano /etc/dphys-swapfile |

$ sudo nano /etc/dphys-swapfile

CONF\_SWAPSIZE  변수값을 다시 100으로 수정한다.

# set size to absolute value, leaving empty (default) then uses computed value

#   you most likely don't want this, unless you have an special disk situation

CONF\_SWAPSIZE=100

스왑 서비스 재시작하여 변경된 설정을 반영시켜주면 스왑 크기가 원래대로 돌아온다.

pi@raspberrypi:~/opencv/opencv-4.5.5/build $ sudo /etc/init.d/dphys-swapfile restart

[ ok ] Restarting dphys-swapfile (via systemctl): dphys-swapfile.service.

pi@raspberrypi:~/opencv/opencv-4.5.5/build $ free

               total        used        free      shared  buff/cache   available

Mem:         3930860      181020     2235192       36256     1514648     3556824

Swap:         102396           0      102396

|  |
| --- |
| pi@raspberrypi:~/opencv/opencv-4.5.5/build $ python3 |
| Python 3.9.2 (default, May  23 2022, 03:53:54)  [GCC 7.3.0] on linux  Type "help", "copyright", "credits" or "license" for more information.  >>> import cv2  >>> cv2.\_\_version\_\_  '4.5.5'  >>> |

OpenCV가 잘 설치된 것을 확인 할 수 있다.

1. **세부코드**
   1. 눈동자 트래킹 및 깜빡임 횟수 검출

눈동자 트래킹을 하기 위해서 딥러닝을 통해 학습을 시켜야하지만 OpenCV 깃헙에서 잘 학습된 데이터셋을 오픈소스로 배포하기 때문에 OpenCV 깃헙에서 가져온다.

깃헙 주소 : <https://github.com/opencv/opencv/tree/4.x/data/haarcascades>

텍스트이(가) 표시된 사진

자동 생성된 설명

<그림 – OpenCV 오픈소스>

사람, 실내, 안경, 보안경이(가) 표시된 사진

자동 생성된 설명

<그림 – 눈동자를 트래킹한 모습 >

또한 눈동자만 인식하는 것이 아닌 이 눈동자에서 랜드마크를 찍어 눈동자의 깜빡임을 감지 해야한다.

랜드마크 참고 :<https://pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/>

**텍스트, 벽, 스크린샷이(가) 표시된 사진

자동 생성된 설명**

<그림 -> 라이브러리

텍스트이(가) 표시된 사진

자동 생성된 설명

<그림 -> 검출기 및 랜드마크

텍스트이(가) 표시된 사진

자동 생성된 설명

<그림 – 68개 랜드마크 리스트>

총 68개의 랜드마크를 찾는다. (턱, 눈, 눈썹, 얼굴 형태, 입 등)

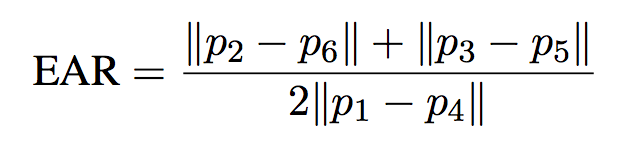
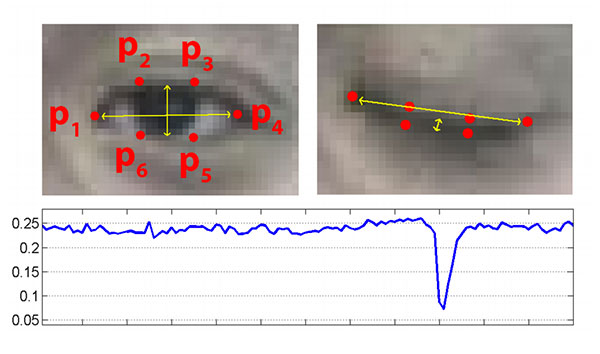
왼쪽 눈과 오른쪽 눈만 사용할 것 이기 때문에 36번부터 48번 까지 사용한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

<그림 – 세부코드1>

설정한 랜드마크를 찍고, 유클리드 거리 계산을 통해 눈의 가로와 세로의 종횡비를 계산한다.



[Soukupová와 Čech 의 논문 Real-Time Eye Blink Detection using Facial Landmarks]

<그림 – 종횡비 계산>

영상이 아닌 실시간으로 눈 깜빡임을 확인하기 위하여 OpenCV의 비디오 캡쳐를 사용한다.



텍스트이(가) 표시된 사진

자동 생성된 설명

<그림 – 세부코드2>

영상을 흑백으로 바꾸어 처리하는 이유는 노이즈를 없애고 연산처리 속도를 빠르게 하기 위함이다. RGB값을 함께 계산하기 되면 연산처리속도가 느려질 수 밖에 없다.

그 예시로 고양이는 적록색약인데 밤중에도 사냥을 잘한다. 그래서 흑백영상을 이용하는것이 정확도면에서도 뒤쳐지지않고 연산처리에도 유리하기때문이다.

앞서 계산한 종횡비 값(EAR)이 0.45미만이면 눈동자를 감은것으로 판단하였고, 2초이상 감고있다면 졸고있는 것으로 기준을 잡아 위험알림을 firebase를 통해 전송한다.

또한 1분마다 firebase를 통해 눈깜빡임 횟수 및 CO2농도를 전송한다.

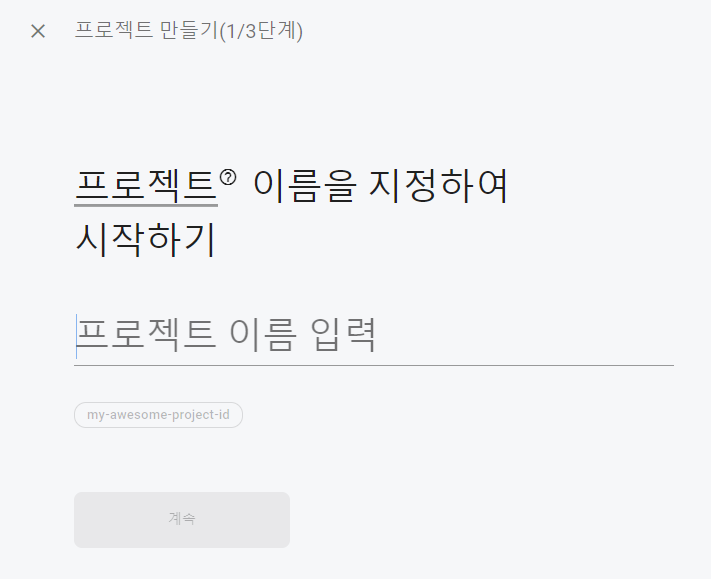
텍스트, 사람, 스크린샷이(가) 표시된 사진

자동 생성된 설명

<그림 – 랜드마크 및 EAR출력>

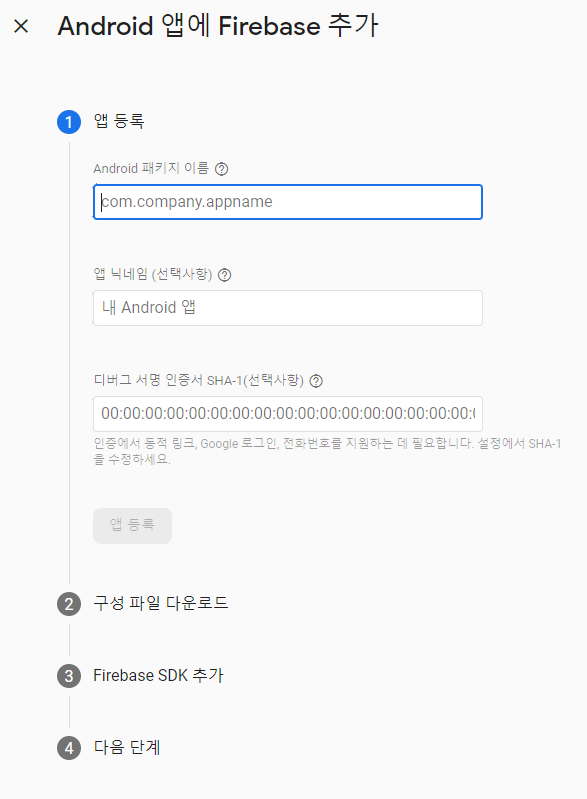
코드를 실행하여 랜드마크를 찍고 EAR값을 출력하는 것을 확인.

* 1. Realtimebase로 DB값 전달



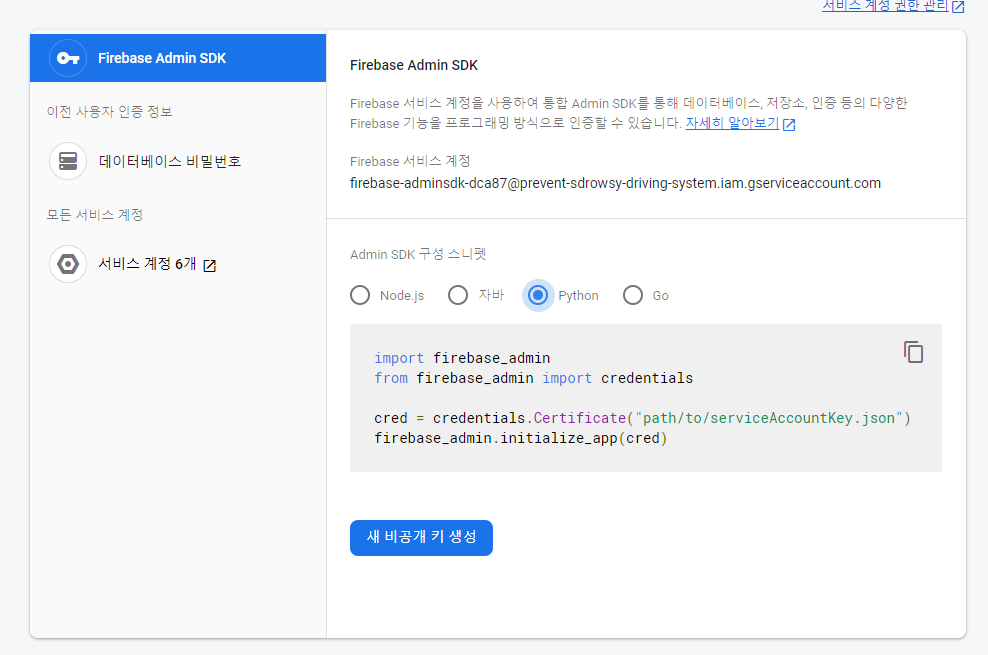
<그림 – firebase-1>

Firebase에서 프로젝트 생성을 하고, 플랫폼을 안드로이드로 선택.



<그림 – firebase-2 >

앱 등록 및 구성파일 다운로드, SDK추가 등 단계를 거친다.



<그림 – firebase-3>

.json파일인 비공개 키를 생성하여 라즈베리파이의 /home/pi에 저장.

텍스트이(가) 표시된 사진

자동 생성된 설명

<그림 - >자격증명

고유한 키값을 가진 json파일로 자격증명 변수 cred를 선언한다.

DB경로를 세팅해주고, 값을 업데이트한다.