사이버보안 캡스톤 디자인 최종 발표

생체 인식을 활용한 기숙사 출입 시스템

- BIOMETRICS
팀장 1771105 박윤아
팀원 1771480 김예림
팀원 1871230 임지연

지도교수 이후진 교수님

기획 의도

가천대 고시관 실장이 여자 기숙사 몰래 들어가

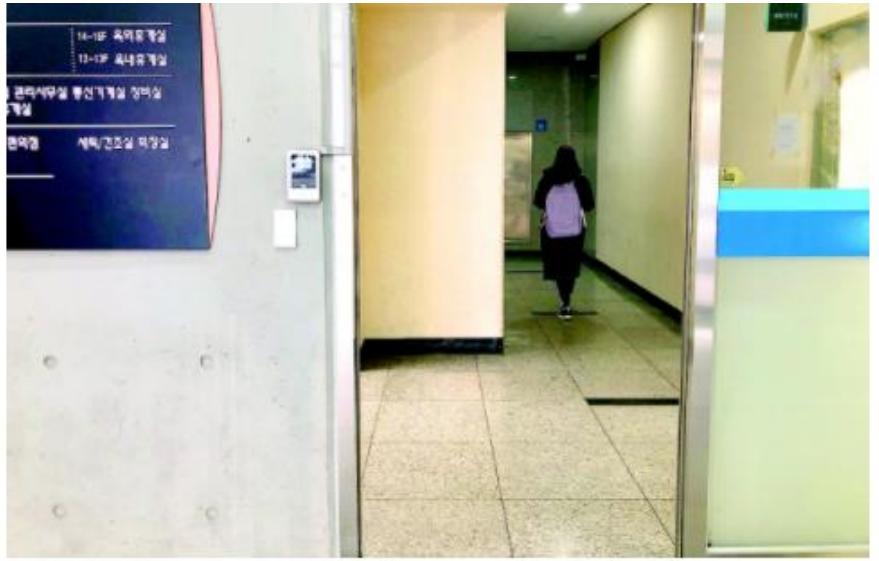
♡ 2 💬 0

입력 2021.02.19 13:10 수정 2021.02.19 14:29

대학 교직원이 여자 기숙사에 몰래 들어갔다는 신고가 접수돼 경찰이 수사에 나섰다.

19일 가천대 재학생 등에 따르면 설 연휴 경기 성남시 가천대 글로벌캠퍼스의 가천고시관 실장 A(남)씨는 고시관에 있는 여자 기숙사에 무단 침입했다. A씨는 자신이 갖고 있는 마스터키로 새벽에 몰래 들어간 것으로 파악됐다.

출처:
https://www.hankookilbo.com/News/Read/A2021021912130001505?did=NA



▲ 19일 오후 30대 남성의 무단침입 사건이 발생한 춘천 소재 대학교 여자기숙사 출입문으로 학생이 들어가고 있다.

대학교 여자기숙사에 30대 남성이 무단으로 침입하는 사건이 발생해 소동을 빚었다.지난해 12월 부산의 여자기숙사에 외부인이 침입해 성폭행을 시도한 이후 도내에서도 무단침입 사건이 발생하면서 허술한 기숙사 보안이 도마위에 올랐다.

춘천경찰서는 19일 대학교 여자기숙사에 침입한 혐의(건조물 주거침입)로 A(39)씨를 불구속 입건했다.경찰에 따르면 A씨는 지난 18일 오후 10시 40분쯤 춘천 소재의 대학교 여자기숙사에서 여학생들을 따라 출입문을 통과해 엘리베이터를 타고 기숙사 11층까지 올라간 혐의를 받고있다.기숙사에 침입한 A씨는 지나가는 여학생들에게 "라면을 사달라"며 말을 걸었던 것으로 알려졌다.

경찰 조사 결과 A씨는 지난 17일 서울에 있는 자택에서 나온 뒤 자취를 감춰 가출신고가 된 상태인 것으로 밝혀졌다.

A씨가 침입한 여자기숙사는 학생증을 보안기기에 갖다대면 출입문이 열리는 시스템으로 과거에도 외부인의 침입이 여러차례 있었던 것으로 밝혀졌다.

출처: https://www.kado.net/news/articleView.html?idxno=958057

기존 기숙사 시스템의 문제점

대부분의 기숙사에서 도입중인

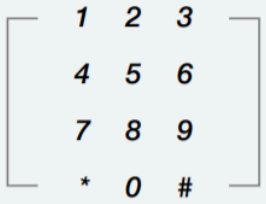
학생증을 태그하여 출입하는 방식은

학생증 분실로 인한 문제점 발생

⇨ 학생증을 습득하여 악의적으로 **악용 가능**

기획 의도

기존 출입 보안 방식의 단점 인식 및 극복



비밀 번호

비밀번호의 유출

기억력에 의존 - 망각 우려

카드 인증

카드 분실에 취약

위조가 쉬움

지문 인식

손쉬운 복제

접촉으로 인한 감염 우려

홍채 인식

짧은 인증 거리

컬러렌즈, 안경 착용 시 인증 불가

높은 가격

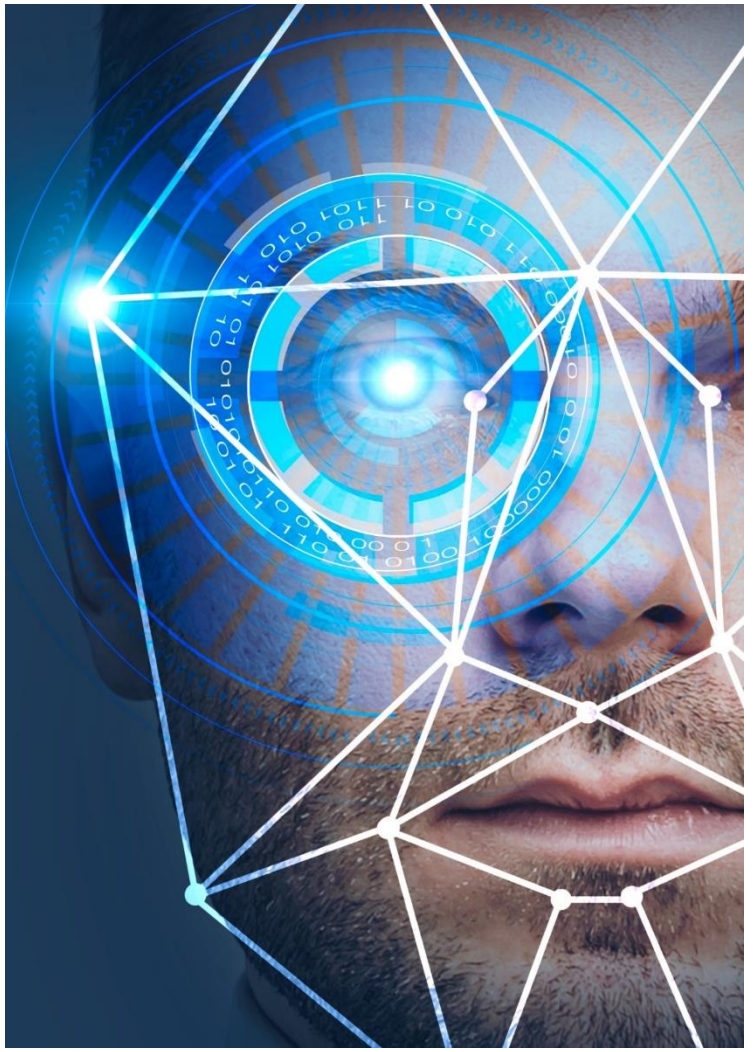
기존 출입 방식의 **문제점을 보완**하기 위해 **안면 인식**을 활용

기획 의도

안면 인식 장점

BIOMETRICS

- 분실 및 망각에 대한 걱정이 없음.
- 특정 인물의 얼굴을 완벽하게 복제할 수 없기 때문에 보안성이 뛰어남.
- 카드 태그나 지문·홍채 등 생체 인식 방식은 사용자가 카드를 꺼내는 동작과 손가락이나 눈을 센서에 가까이 가져가는 동작이 필요함.
- '안면 인식 출입 시스템'은 사용자, 관리자 모두의 편의성을 높여 줌.



기획 의도

안면 인식률이 떨어질 경우,

사진으로도 출입이 가능해지는 문제점 발생

부가적으로 **지문 인식**을 추가하여 이를 해결

기획 의도

지문 인식 장점

BIOMETRICS

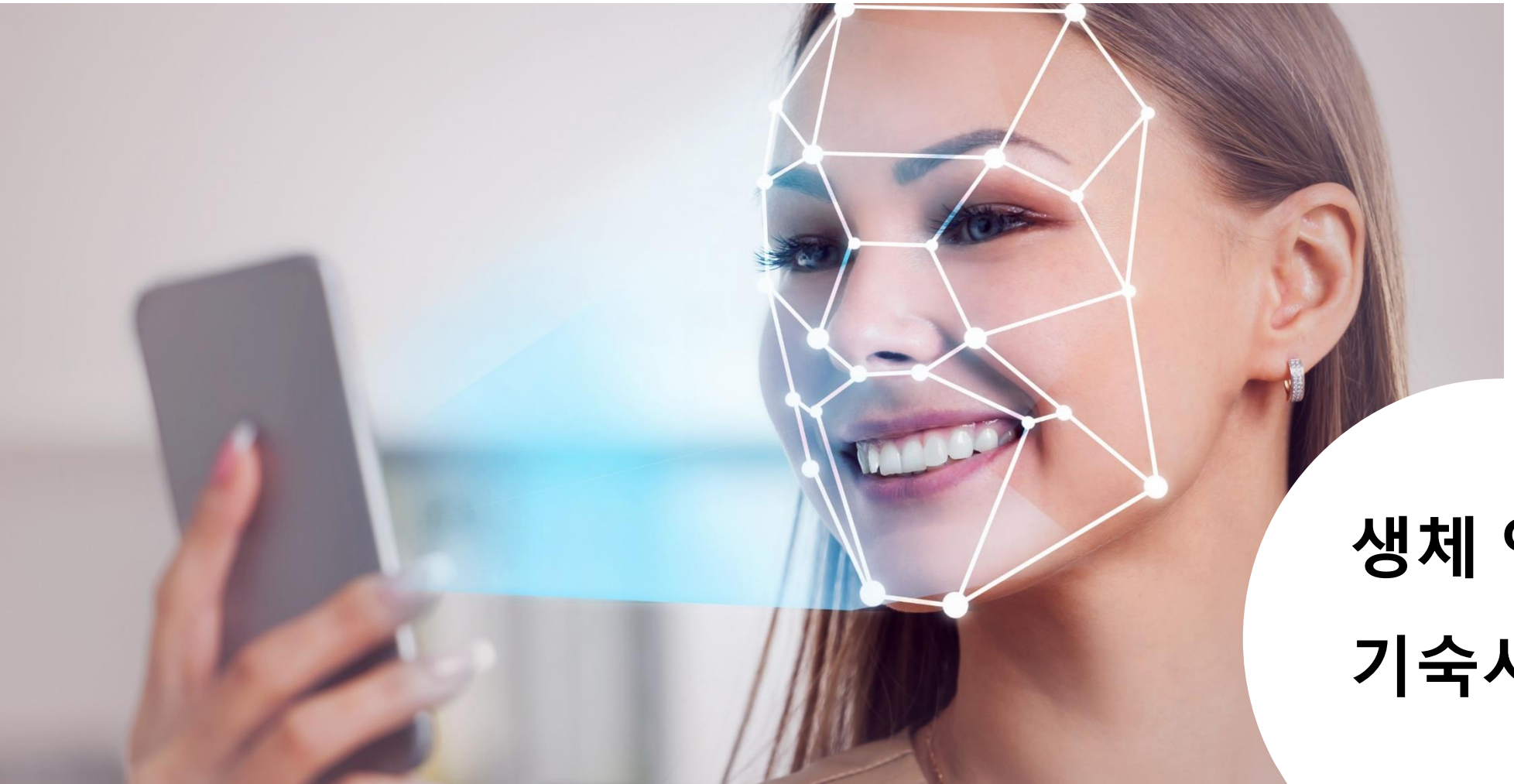
- 지문은 땀샘이 융기되어 일정한 흐름으로 형성된 각 개인의 고유한 신체적 특성으로, 그 모양이 평생 변하지 않는 것으로 알려져 실생활에서 본인 여부를 판별하는 방법으로 사용됨.

- 분실 및 망각에 대한 걱정이 없음.

- 고유한 신체 특성을 완벽하게 복제할 수 없으므로 보안성이 뛰어남.

주요 기능

BIOMETRICS



생체 인식을 활용한
기숙사 출입 시스템

1. 얼굴 인식 기능

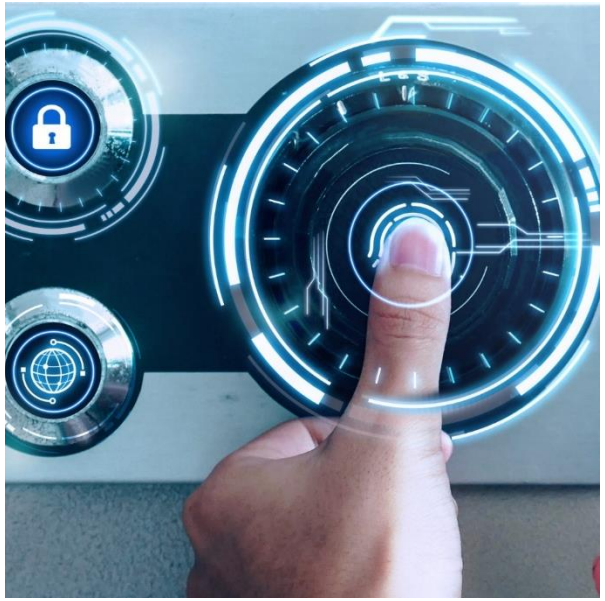
사용자 얼굴의 다양한 모습을 사전에

여러 번 학습시켜 인식률을 높임.

캠을 통해 사용자 얼굴 정보를 받아 와

학습되어 있는 모델과 비교하여

동일 인물로 판단되면 출입이 가능.



2. 지문 인식 기능

안면 인식률이 떨어질 경우를 대비하여

지문 인식 센서에서 사용자 지문 정보를 받아 와

등록되어 있는 지문과 비교하여

동일 지문으로 판단되면 출입이 가능.

3. 이중 보안

안면 인식과 지문 인식을 결합하여

이중 보안을 실행함에 따라 **보안성 강화**.

4. 높은 보안성과 편의성

얼굴과 지문은 고유한 신체 특성으로,

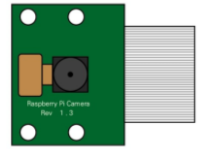
완벽하게 **복제 불가능**

분실에 대한 걱정이 없음

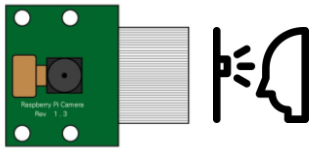
⇨ 뛰어난 보안성과 편의성

프로젝트 구조도

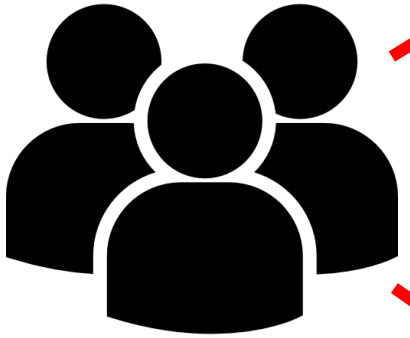
캠을 통해 사용자 얼굴의 다양한
모습을 사전에 여러 번 학습시킴



캠을 통해 사용자 얼굴 정보를 받아 옴



학습되어 있는 모델과 비교하여
동일 인물로 판단되면 출입 가능



User

Data Base

안면 인식률이 떨어질 경우, 사진으로
출입이 가능해지는 문제를 해결하기 위한 방안

등록되어 있는 지문과 비교하여
동일 지문으로 판단되면 출입 가능

지문인식 센서를 통해
사용자 지문 정보를 받아 옴

지문인식 센서를 통해 사용자
지문 정보를 사전에 등록시킴

개발 시 요구된 기술 및 소프트웨어

Python : 객체 지향 기반 인터프리터 방식의 동적 타이핑 고급 프로그래밍 언어

Raspbian : 라즈베리파이 전용 운영체제

OpenCV : 실시간 컴퓨터 비전을 목적으로 한 프로그래밍 라이브러리,
Python에 바인딩되어 사용자에게 개발 환경을 지원

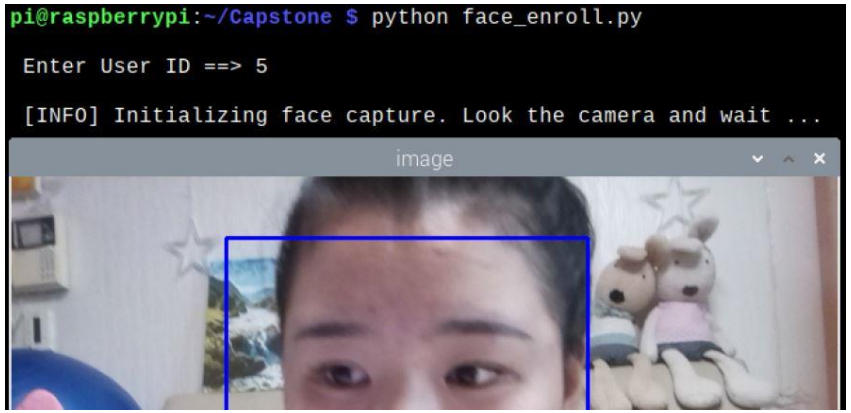
PuTTY : SSH를 위한 클라이언트로 동작하는 응용 프로그램

VNC Viewer : 라즈베리파이 제어에 사용되는 원격 액세스 소프트웨어

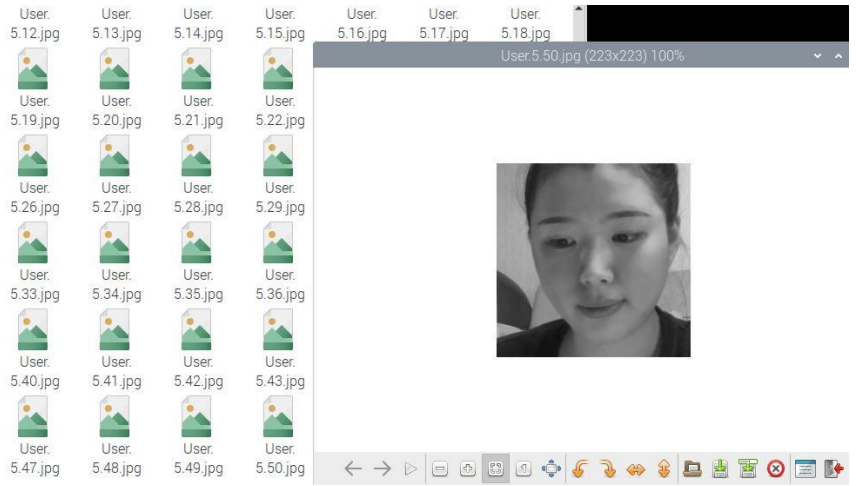
구현 결과

1) 사용자 얼굴 등록

- 등록할 사용자의 ID를 입력하면 카메라 창이 실행되고 100장의 사진 촬영 시작



- dataset 디렉토리에 사용자의 얼굴이 등록 됨



구현 결과

2) 사용자 얼굴 학습

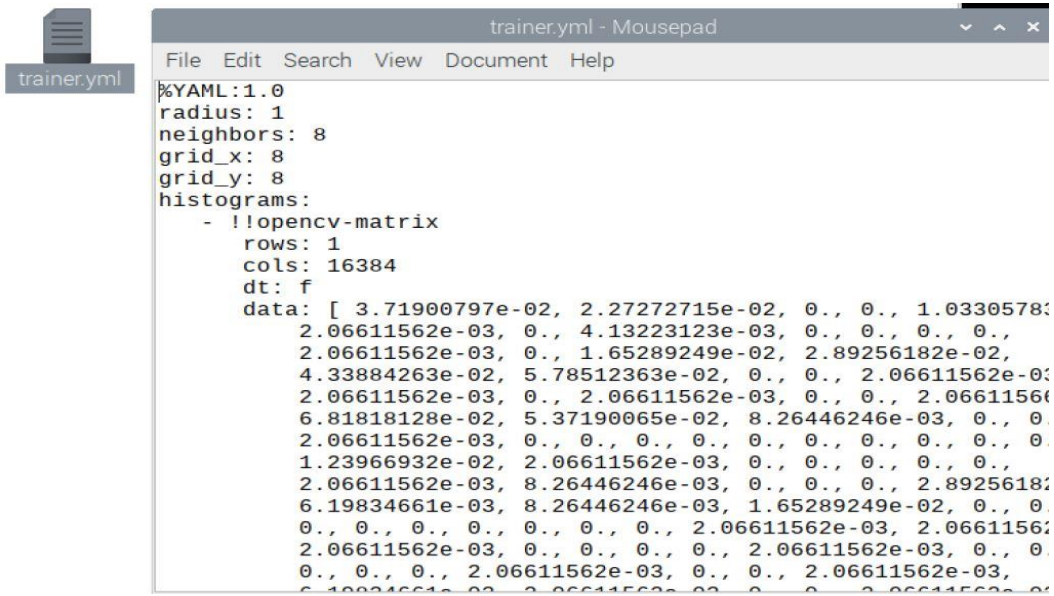
- dataset 디렉토리에 저장되어 있는 사용자들의 얼굴을 학습

```
pi@raspberrypi:~/Capstone $ python face_training.py

 [INFO] Training faces. Wait ...

 [INFO] 4 faces trained. Exiting Program
```

- 각 사용자의 얼굴 별 특징이 추출된 결과가 ‘trainer.yml’에 저장 됨



구현 결과

3) 사용자 지문 등록

- 현재 등록된 지문의 수가 출력되고 지문 인식 센서는 지문이 감지될 때까지 대기

```
pi@raspberrypi:~/Capstone $ python finger_enroll.py
Currently used templates: 2/300
Waiting for finger...
```

- 지문 인식 센서에 지문이 감지되면 지문 등록 시작. 해당 지문의 position Number 할당

```
pi@raspberrypi:~/Capstone $ python finger_enroll.py
Currently used templates: 2/300
Waiting for finger...
Remove finger...
Waiting for same finger again...
Finger enrolled successfully!
New template position #2
```

- 지문 인식 센서에 감지된 지문이 이미 등록된 지문일 경우, 해당 지문의 position Number 출력

```
pi@raspberrypi:~/Capstone $ python finger_enroll.py
Currently used templates: 2/300
Waiting for finger...
Template already exists at position #1
```

구현 결과

4) 사용자 지문 확인

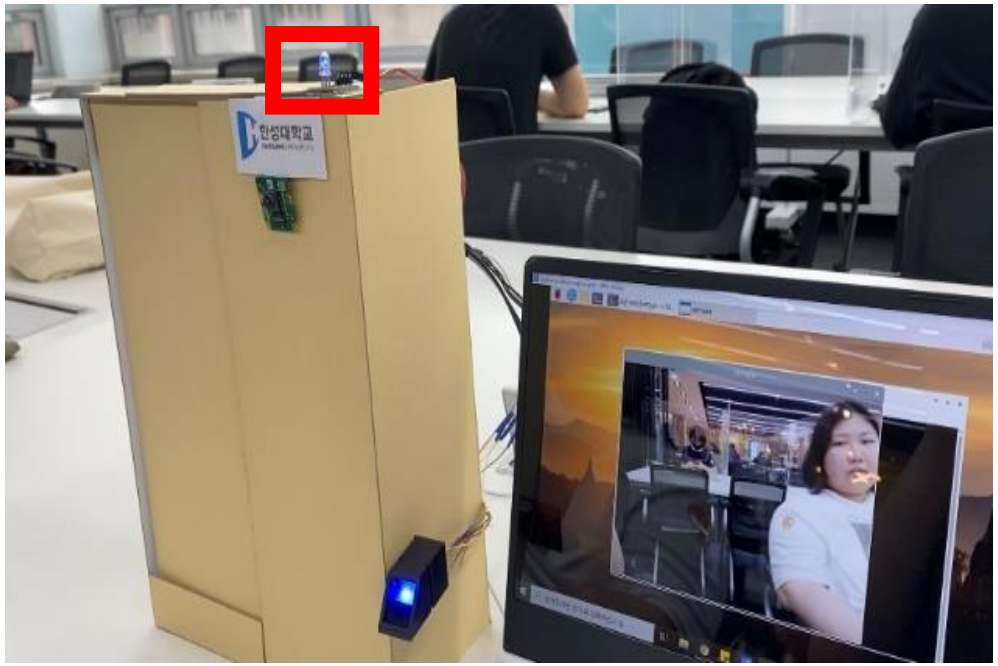
- 지문의 등록 여부를 확인. 지문이 감지되면 해당 지문의 position Number와 정확도를 출력

```
pi@raspberrypi:~/Capstone $ python finger_search.py
Currently used templates: 3/300
Waiting for finger...
Found template at position #1
The accuracy score is: 138
```

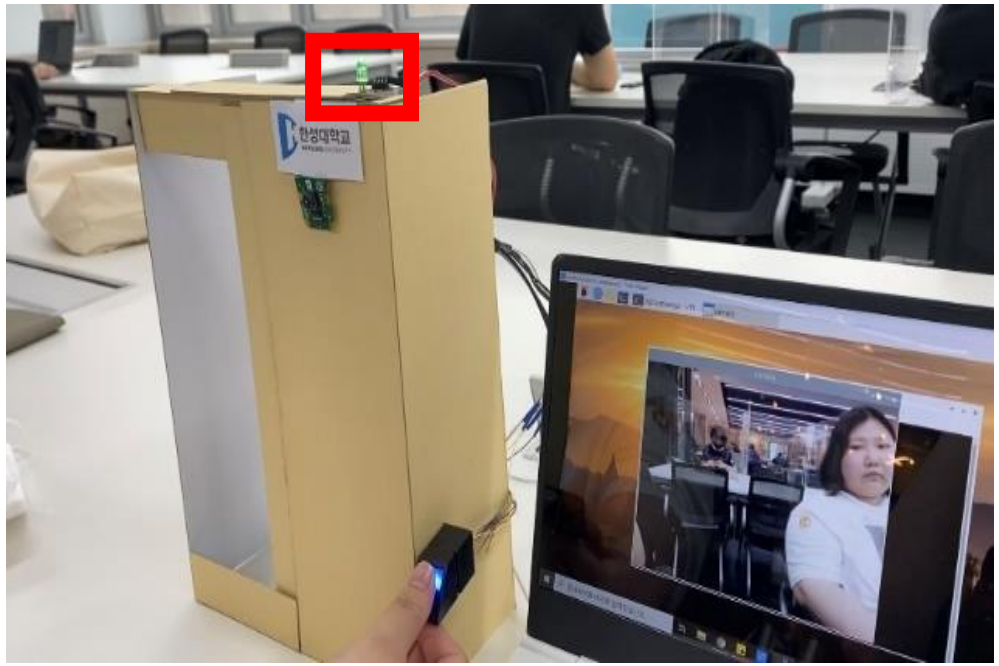
구현 결과

5) 사용자 출입

- 등록된 사용자의 경우, 1단계 안면인식 성공 시 LED 파란색 출력
  2단계 지문인식 성공 시 LED 초록색 출력되며 문 열림



- 등록된 사용자 1단계 안면인식 성공 화면

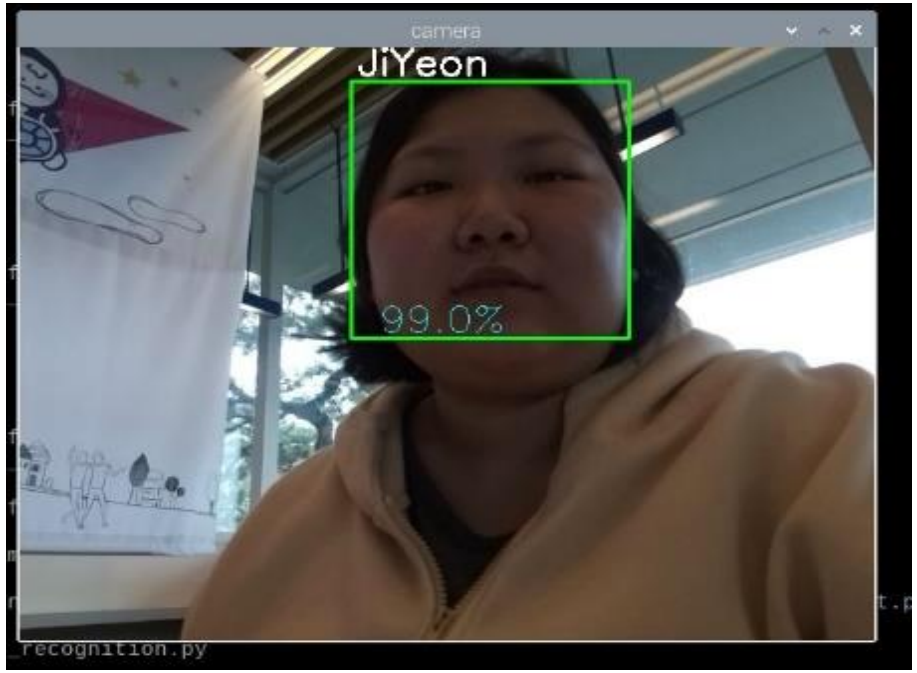


- 등록된 사용자 2단계 지문인식 성공 화면

구현 결과

5) 사용자 출입

- 출입 성공 시, 해당 사용자의 ID와 안면 인식률, 지문 position Number와 정확도 출력

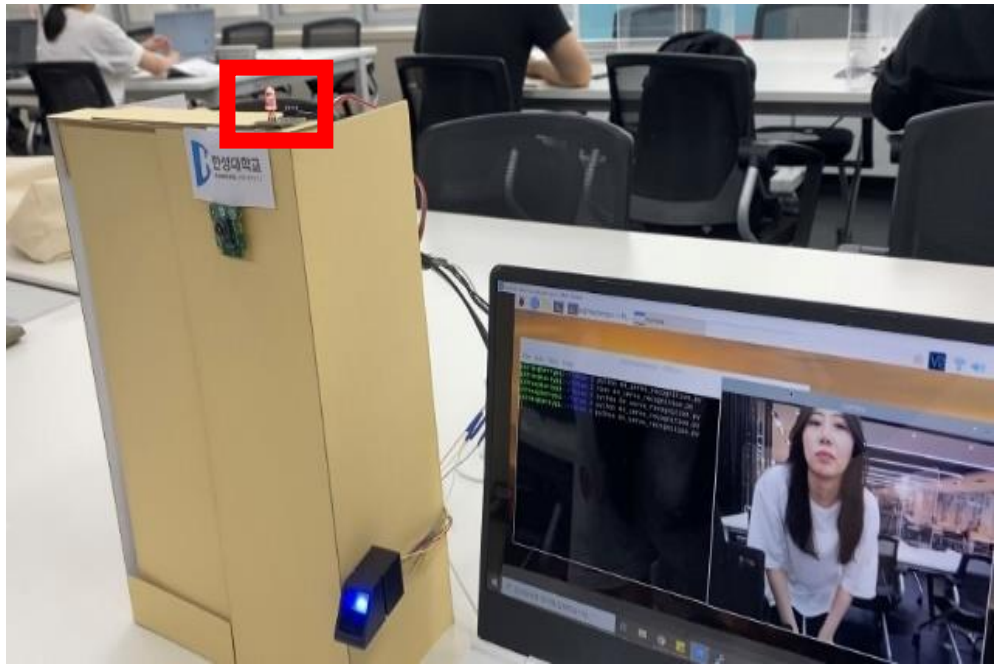



```
pi@raspberrypi:~/Capstone $ python recognition.py
Found template at position #1
The accuracy score is : 127
```

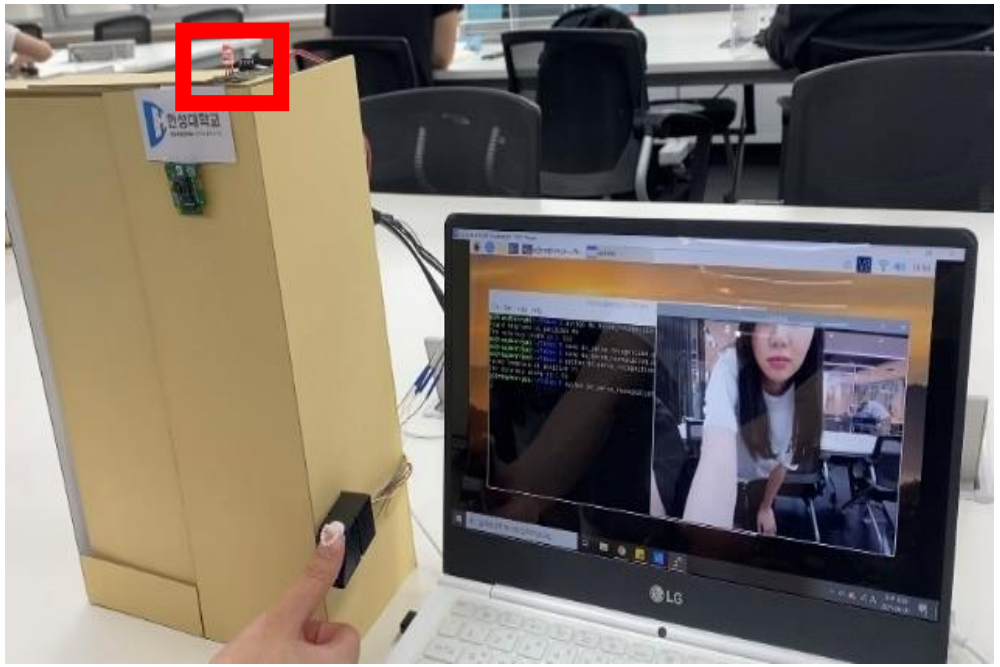
구현 결과

5) 사용자 출입

- 등록되지 않은 사용자의 경우, LED 빨간색 출력. 지문인식 시도 불가



- 등록되지 않은 사용자 안면인식 실패 화면



- 등록되지 않은 사용자 지문인식 실패 화면

기대효과 및 활용방안

- 시스템에서 상황마다 랜덤하게 특정한 표정을 요구하게 함

: 사진이나 가면을 통한 불법 침입 방지. 안면 인식 보안성 강화

- 다양한 생체 인식 기법 추가

: 정맥, 음성, 손금 등을 활용하여 시스템 보안성 강화

⇨ 각각의 **단점을 보완**하여 정확도를 높임으로써 기숙사 침입 문제 예방

프로그램 코드

[face_enroll.py] : 사용자의 얼굴을 등록

```
import cv2
import os

cam = cv2.VideoCapture(0) # VideoCapture 객체 생성
cam.set(3,640) # width 설정
cam.set(4,480) # height 설정

# haarcascade_frontalface_default.xml을 Classifier로 사용
face_detector = cv2.CascadeClassifier('/home/pi/opencv/data/haarcascades/haarcascade_frontalface_default.xml')

# 등록하는 사용자의 id 입력
face_id = input('\n Enter User ID ==> ')
print("\n [INFO] Initializing face capture. Look the camera and wait ...")

count = 0 # count 초기화

while(True):
        ret, img = cam.read()
        img = cv2.flip(img,-1) # 상하반전
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = face_detector.detectMultiScale(gray, 1.3, 5)

        for (x,y,w,h) in faces:
                cv2.rectangle(img, (x,y), (x+w, y+h), (255,0,0), 2)
                count += 1
                # 촬영한 사진 dataset 폴더에 저장
                cv2.imwrite("dataset/User." + str(face_id) + '.' +str(count) + ".jpg", gray[y:y+h, x:x+w])
                cv2.imshow('image', img) # image라는 이름으로 출력

        k = cv2.waitKey(100) & 0xff
        if k==27: # ESC를 누르면 종료
                break
        elif count >= 100: # 100장을 촬영하고 종료
                break

print("\n [INFO] Exiting Program and cleanup stuff")
cam.release()
cv2.destroyAllWindows()
```

프로그램 코드

[face_training.py] : 사용자의 얼굴을 학습

```
import cv2
import numpy as np
from PIL import Image
import os

path = 'dataset' # 사용자 얼굴 DB 경로
recognizer = cv2.face.createLBPHFaceRecognizer() # Recognizer 생성
# haarcascade_frontalface_dafault.xml을 Classifier로 사용
detector = cv2.CascadeClassifier("/home/pi/opencv/data/haarcascades/haarcascade_frontalface_default.xml");

# 사용자 얼굴 DB를 불러와 np.array 생성
def getImagesAndLabels(path):
        imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
        faceSamples=[]
        ids=[]
        for imagePath in imagePaths:
                PIL_img = Image.open(imagePath).convert('L')
                img_numpy = np.array(PIL_img, 'uint8')
                id = int(os.path.split(imagePath)[-1].split(".")[1])
                faces = detector.detectMultiScale(img_numpy)

                for (x,y,w,h) in faces:
                        faceSamples.append(img_numpy[y:y+h, x:x+w])
                        ids.append(id)
        return faceSamples, ids

print("\n [INFO] Training faces. Wait ...")
faces, ids = getImagesAndLabels(path)
# np.array를 학습시킴. DB의 양이 많을수록 더 많은 시간이 소요
recognizer.train(faces, np.array(ids))

# 학습시킨 결과를 trainer.yml로 저장
recognizer.save('trainer/trainer.yml')
# 학습된 얼굴의 수 출력
print("\n [INFO] {0} faces trained. Exiting Program".format(len(np.unique(ids))))
```

프로그램 코드

[finger_enroll.py] : 사용자의 지문을 등록

```
import time
from pyfingerprint.pyfingerprint import PyFingerprint

# 지문인식 센서 초기화
try:
    f = PyFingerprint('/dev/ttyUSB0', 57600, 0xFFFFFFFF, 0x00000000)

    if ( f.verifyPassword() == False ):
        raise ValueError('The given fingerprint sensor password is wrong!')

except Exception as e:
    print('The fingerprint sensor could not be initialized!')
    print('Exception message: ' + str(e))
    exit(1)

# 지문 정보 불러오기
print('Currently used templates: ' + str(f.getTemplateCount()) +'/'+ str(f.getStorageCapacity()))

# 새 지문 등록
try:
    print('Waiting for finger...')

    while ( f.readImage() == False ): # 지문 인식 대기
        pass

    f.convertImage(0x01) # 인식한 지문 이미지를 변환하여 저장_1

    result = f.searchTemplate()
    positionNumber = result[0]

    if ( positionNumber >= 0 ): # 이미 등록된 지문인지 확인
        print('Template already exists at position #' + str(positionNumber))
        exit(0)
```

```
    print('Remove finger...')
    time.sleep(2)

    print('Waiting for same finger again...')

    while ( f.readImage() == False ): # 지문 인식 재대기
        pass

    f.convertImage(0x02) # 인식한 지문 이미지를 변환하여 저장_2

    if ( f.compareCharacteristics() == 0 ): # _1과 _2를 비교
        raise Exception('Fingers do not match')

    f.createTemplate() # Template 생성

    # 새 postionNumber에 Template 저장
    positionNumber = f.storeTemplate()
    print('Finger enrolled successfully!')
    print('New template position #' + str(positionNumber))

except Exception as e:
    print('Operation failed!')
    print('Exception message: ' + str(e))
    exit(1)
```

프로그램 코드

[finger_search.py] : 해당 지문이 등록되어 있는지 확인

```
import hashlib
from pyfingerprint.pyfingerprint import PyFingerprint

# 지문인식 센서 초기화
try:
    f = PyFingerprint('/dev/ttyUSB0', 57600, 0xFFFFFFFF, 0x00000000)

    if ( f.verifyPassword() == False ):
        raise ValueError('The given fingerprint sensor password is wrong!')

except Exception as e:
    print('The fingerprint sensor could not be initialized!')
    print('Exception message: ' + str(e))
    exit(1)

# 지문 정보 불러오기
print('Currently used templates: ' + str(f.getTemplateCount()) +'/'+ str(f.getStorageCapacity()))

# 지문 검색
try:
    print('Waiting for finger...')

    while ( f.readImage() == False ): # 지문 인식 대기
        pass

    f.convertImage(0x01) # 인식한 지문 이미지를 변환하여 저장_1

    # template 검색
    result = f.searchTemplate()

    positionNumber = result[0]
    accuracyScore = result[1]

    if ( positionNumber == -1 ):
        print('No match found!')
        exit(0)
    else: # positionNumber와 정확도 출력
        print('Found template at position #' + str(positionNumber))
        print('The accuracy score is: ' + str(accuracyScore))

except Exception as e:
    print('Operation failed!')
    print('Exception message: ' + str(e))
    exit(1)
```

프로그램 코드

[recognition.py] : 안면 인식 + 지문 인식 실행

```
import cv2
import numpy as np
import os
import RPi.GPIO as GPIO
import time
import hashlib
from pyfingerprint.pyfingerprint import PyFingerprint

# 지문인식 센서 초기화
try:
        f=PyFingerprint('/dev/ttyUSB0', 57600, 0XFFFFFFFF, 0x00000000)

        if(f.verifyPassword() == False):
                raise ValueError('The fingerprint sensor is wrong!')

except Exception as e:
        print('The fingerprint sensor could not be initialized!')
        print('Exception message: ' + str(e))
        exit(1)

# RGB LED 모듈 핀번호
R=17
G=27
B=22

GPIO.setwarnings(False) # 경고 메세지 해제
GPIO.setmode(GPIO.BCM) # GPIO 모드 설정

GPIO.setup(18, GPIO.OUT) # 서보모터 핀번호 18
# RGB LED 모듈 OUTPUT으로 설정
GPIO.setup(R, GPIO.OUT)
GPIO.setup(G, GPIO.OUT)
GPIO.setup(B, GPIO.OUT)
```

```
GPIO.output(18, 1)
p = GPIO.PWM(18, 50) # 서보모터를 50Hz로 지정
p.start(1) # 서보모터 실행(duty 1%부터)

recognizer = cv2.face.createLBPHFaceRecognizer() # Recognizer 생성
recognizer.load('trainer/trainer.yml')
cascadePath = "/home/pi/opencv/data/haarcascades/haarcascade_frontalface_default.xml"
# haarcascades_frontalface_default.xml을 Classifier로 사용
faceCascade = cv2.CascadeClassifier(cascadePath);
font = cv2.FONT_HERSHEY_SIMPLEX

id = 0 # id 초기화
names = ['None', 'JiYeon', 'YeLim', 'YoonA'] # 사용자 이름

cam = cv2.VideoCapture(0) # VideoCapture 객체 생성
cam.set(3,640) # width 설정
cam.set(4,480) # height 설정

# 최소 창 크기 정의
minW = 0.1*cam.get(3)
minH = 0.1*cam.get(4)

while True:
        ret, img = cam.read()
        img = cv2.flip(img, -1) # 상하반전
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

        faces = faceCascade.detectMultiScale(
                gray,
                scaleFactor = 1.2,
                minNeighbors = 5,
                minSize = (int(minW), int(minH)),
        )

        for(x,y,w,h) in iter(faces):
                cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
                id = recognizer.predict(gray[y:y+h,x:x+w])
                confidence = recognizer.predict(gray[y:y+h,x:x+w])

                # confidence의 값이 작을수록 인식률 높은 것
                if(confidence < 4): # 얼굴인식 성공시
                        id = names[id]
                        confidence = " {0}%".format(round(100 - confidence))
```

프로그램 코드

[recognition.py] : 안면 인식 + 지문 인식 실행

```
# LED 파란색 출력
GPIO.output(R,False)
GPIO.output(G,False)
GPIO.output(B,True)

try:
        while(f.readImage() == False): # 지문 인식 대기
                pass

        f.convertImage(0x01) # 인식한 지문 이미지를 변환하여 저장_1

        result = f.searchTemplate() # template 검색

        positionNumber = result[0]
        accuracyScore = result[1]

        if(positionNumber == -1): # 등록된 지문이 아닌 경우
                print('No match found!')
                # LED OFF
                GPIO.output(R,False)
                GPIO.output(G,False)
                GPIO.output(B,False)
                exit(0)

        else: # 지문인식 성공시
                p.ChangeDutyCycle(9) # 서보모터 작동(가상 현관 개방)
                # LED 초록색 출력
                GPIO.output(R,False)
                GPIO.output(G,True)
                GPIO.output(B,False)
                time.sleep(2)
                p.stop() # 서보모터 OFF
                # LED OFF
                GPIO.output(R,False)
                GPIO.output(G,False)
                GPIO.output(B,False)

                # positionNumber와 정확도 출력
                print('Found template at position #' + str(positionNumber))
                print('The accuracy score is : ' + str(accuracyScore))
```

```
                exit(1)

        except Exception as e:
                print('Operation failed!')
                print('Exception message: ' + str(e))
                exit(1)

    else: # 얼굴인식 실패시
        id = "unknown"
        confidence = " {0}%".format(round(100 - confidence))
        # LED 빨간색 출력
        GPIO.output(R,True)
        GPIO.output(G,False)
        GPIO.output(B,False)
        time.sleep(6)
        # LED OFF
        GPIO.output(R,False)
        GPIO.output(G,False)
        GPIO.output(B,False)
        exit(1)

      # 해당 사용자의 id와 정확도 출력
      cv2.putText(img, str(id), (x+5,y-5), font, 1, (255,255,255), 2)
      cv2.putText(img, str(confidence), (x+5,y+h-5), font, 1, (255,255,0), 1)

    cv2.imshow('camera', img)
    k = cv2.waitKey(10) & 0xff
    if k==27: # ESC 누르면 종료
        break

print("\n [INFO] Exiting Program and cleanup stuff")
cam.release()
cv2.destroyAllWindows()
GPIO.cleanup() # GPIO 리셋
```

주차별 개발 일정

주차	목표
1~3	주제 선정 및 프로젝트 제안서 발표
4	OpenCV 설치 코드 분석
5	안면인식 및 지문인식 코드 분석
6	Raspbian 설치 및 원격 접속
7	SSH, Wi-Fi 파일 오류 해결
8	중간 발표
9	‘Assertion failed’ 오류 해결
10	안면 인식 환경 구축 (1)
11	카메라 강제 종료 오류 해결
12	안면 인식 환경 구축 (2)
13	가상 현관 제작 및 동작 테스트
14	지문 인식 환경 구축 및 최종 테스트
15	최종 발표

팀원 소개

1771105

박윤아 (팀장)

pya7106@naver.com

1771480

김예림

ddr2038@naver.com

1871230

임지연

dlab0104@naver.com

팀원 별 담당 업무

박윤아 (팀장)

- 안면인식 및 지문인식 환경 구축
- 가상현관 제작
- 주차별 보고서 및 최종 보고서 작성
- ppt제작, 발표자료 최종 검토 및 전체 발표자

김예림

- 안면인식 및 지문인식 환경 구축
- 가상현관 제작
- 자료수집 및 PPT 제작
- 최종 보고서 작성

임지연

- 안면인식 및 지문인식 환경 구축
- 가상현관 제작
- 자료수집 및 PPT 제작

참고문헌

- 참고 기사 출처

(1) https://www.kado.net/news/articleView.html?idxno=958057

(2) https://www.hankookilbo.com/News/Read/A20210219121300015O5?did=NA

- 라즈베리파이 원격 접속

: https://sergeswin.com/1268/

- OpenCV 설치

: https://jow1025.tistory.com/276?category=924125

: https://towardsdatascience.com/installing-opencv-3-4-3-on-raspberry-pi-model-b-e9af08a9f1d9

-안면인식 관련 논문

: https://scienceon.kisti.re.kr/commons/util/originalView.do?cn=CFKO201714956117030&oCn=NPAP12898491&dbt=CFKO&journal=NPRO00379494&keyword=%EC%95%88%EB%A9%B4%EC%9D%B8%EC%8B%9D

- 안면인식 예제

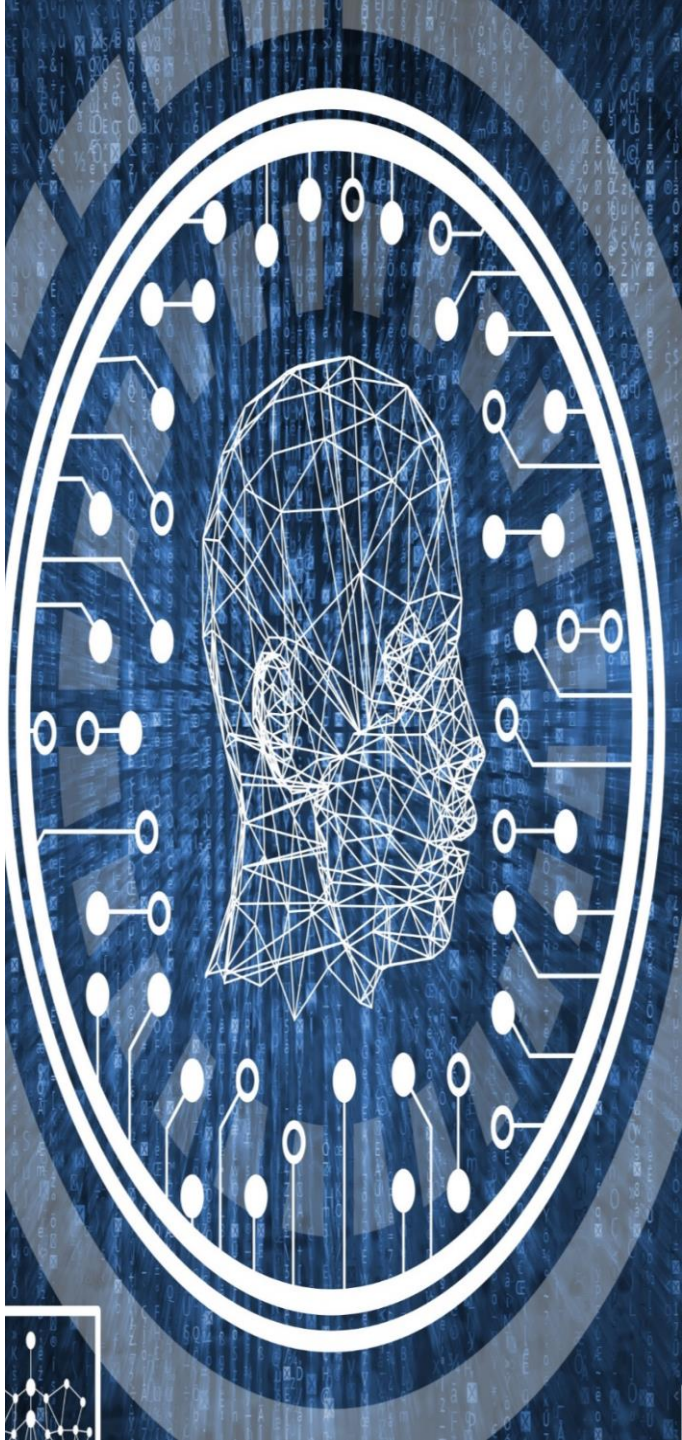
: https://www.hackster.io/mjrobot/real-time-face-recognition-an-end-to-end-project-a10826

- 지문인식 관련 논문

: https://scienceon.kisti.re.kr/commons/util/originalView.do?cn=JAKO201708034060398&oCn=JAKO201708034060398&dbt=JAKO&journal=NJOU00292091

- 지문인식 예제

: https://circuitdigest.com/microcontroller-projects/raspberry-pi-fingerprint-sensor-interfacing



THANK YOU

감사합니다

프로젝트 시연 영상 및 최종 발표 영상

프로젝트 시연 영상

: https://youtu.be/cURZHSUOnnc

최종 발표 영상

: https://youtu.be/3efb9waEcPA