사이버보안 캡스톤 디자인

**USB Locker**

**최종 결과 보고서**

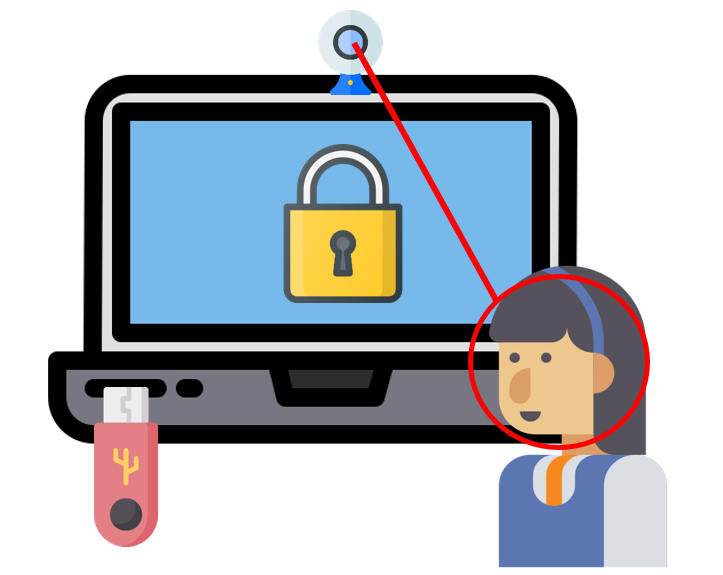
지도교수 서화정교수님

팀원 송경주(1771132)

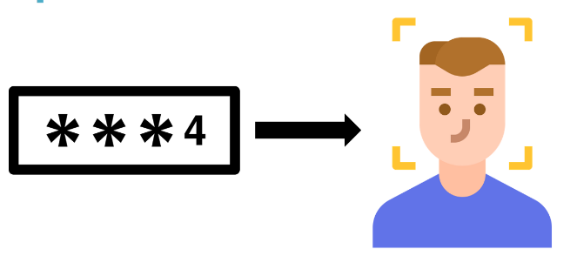
백민진(1771116)

강혜진(1791126)

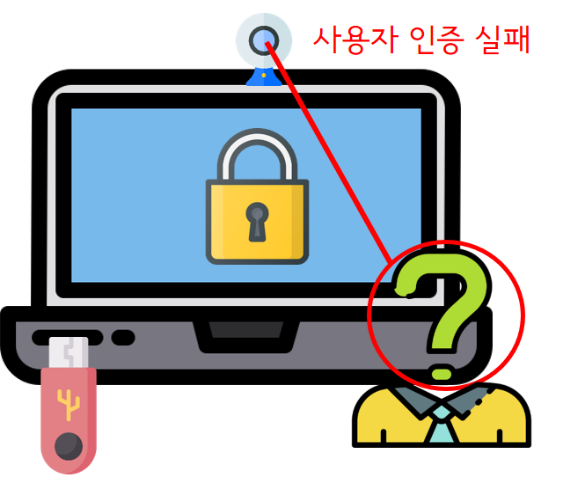
**[ 프로젝트 개요 ]**



USB를 열 때 캠을 통해 사용자 얼굴 정보를 받아와 학습되어 있는 모델과 비교하여, 사용자와 학습 모델이 동일할 시에 암호화 되어있던 USB내 정보들이 복호화 됩니다. 인증에 성공한 사용자만이 USB내 정보들에 접근할 수 있습니다.

**[강화된 보안]**

기존 비밀번호(PIN번호)로만 잠금이 가능한 USB의 보안을 강화시키기 위하여 생체 인식을 통해 본인인증이 가능한 USB 보안 프로그램입니다.



**[외부 사람 접근 불가]**

강화된 보안으로 본인이 아닌 다른 사람들은 얼굴 인증이 되지 않기 때문에 USB를 열거나 사용하는 것이 불가능합니다. 이로 인해 USB 분실 시 발생할 수 있는 개인정보 유출의 위험성도 감소할 것입니다.

**[ 주제 선정 배경 ]**

USB는 가지고 다니기도 편하고 저장용량에 따라 많은 양의 정보를 저장할 수 있어 많은 사람들이 사용하는 이동식 기억장치입니다. USB 메모리는 작고 가벼워서 휴대하기 편한 대신 잃어버리기도 아주 쉽습니다. 따라서 아주 중요한 정보를 USB에 저장해야 한다면 보안성이 가장 중요한 요소가 될 수 있습니다. 그래서 저희는 얼굴인식을 이용한 USB 잠금 해제 프로그램을 개발하고자 합니다.

**[ 주요 기능 및 시스템 구성 ]**

카메라를 통해 사용자 얼굴 데이터를 수집하고 사용자 얼굴을 특징 점을 학습합니다.

카메라를 통해 이용자 얼굴 데이터를 수집하여 학습된 모델과 비교 한 후 동일할 시 생체 인증이 성공되어 USB 파일에 접근할 수 있습니다.

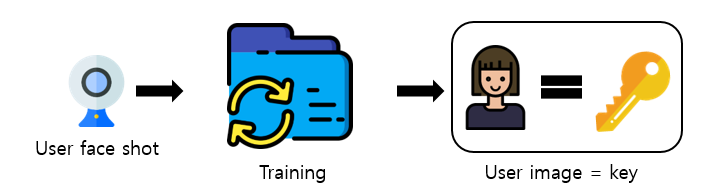
**[ 복제 불가능 ]**

완벽한 복제인간이 탄생하지 않는 한 특정 인간의 신체적 특징을 완벽하게 복제할 수 없기 때문에 보안성에서 큰 신뢰를 받을 수 있습니다.

**[ 이중 보안 ]**

도난 당하거나 잊어버릴 일이 없어 편리한 생체 인증과 핀 번호 입력을 모두 수행하기 때문에 보안이 더 강화됩니다.

**[ User face registration ]**



**[ User face recognition ]**



**[ 개발 시 요구된 기술 및 소프트웨어/하드웨어 소개 ]**

<기술>

shell : user와 linux 사이의 인터페이스로 작동하는 프로그램

python : 객체 지향 기반 인터프리터 방식의 고급 프로그래밍 언어

rules : 메모리상 규칙사항 구성, USB 메모리 장치의 연결이 발생하면 shell파일 자동실행

<소프트웨어>

raspbian : 라즈베리파이 전용 운영 체제

opencv : 실시간 컴퓨터 비전을 목적으로 한 프로그래밍 라이브러리

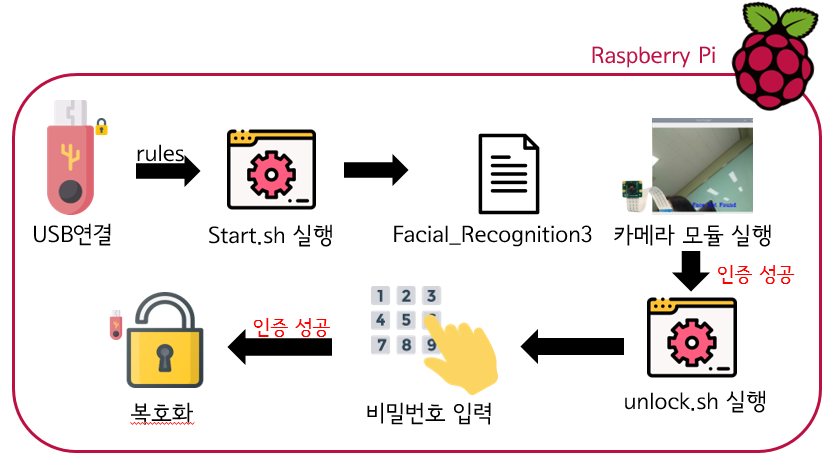
openssl : 보안 통신을 위해 사용하는 프로토콜을 구현한 라이브러리

<하드웨어>

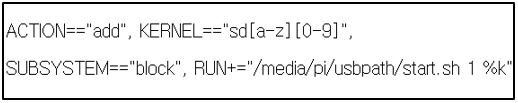
Raspberry Pi 3 Model B

Raspberry Pi Camera V2

**[ 구현 결과 상세 소개 ]**

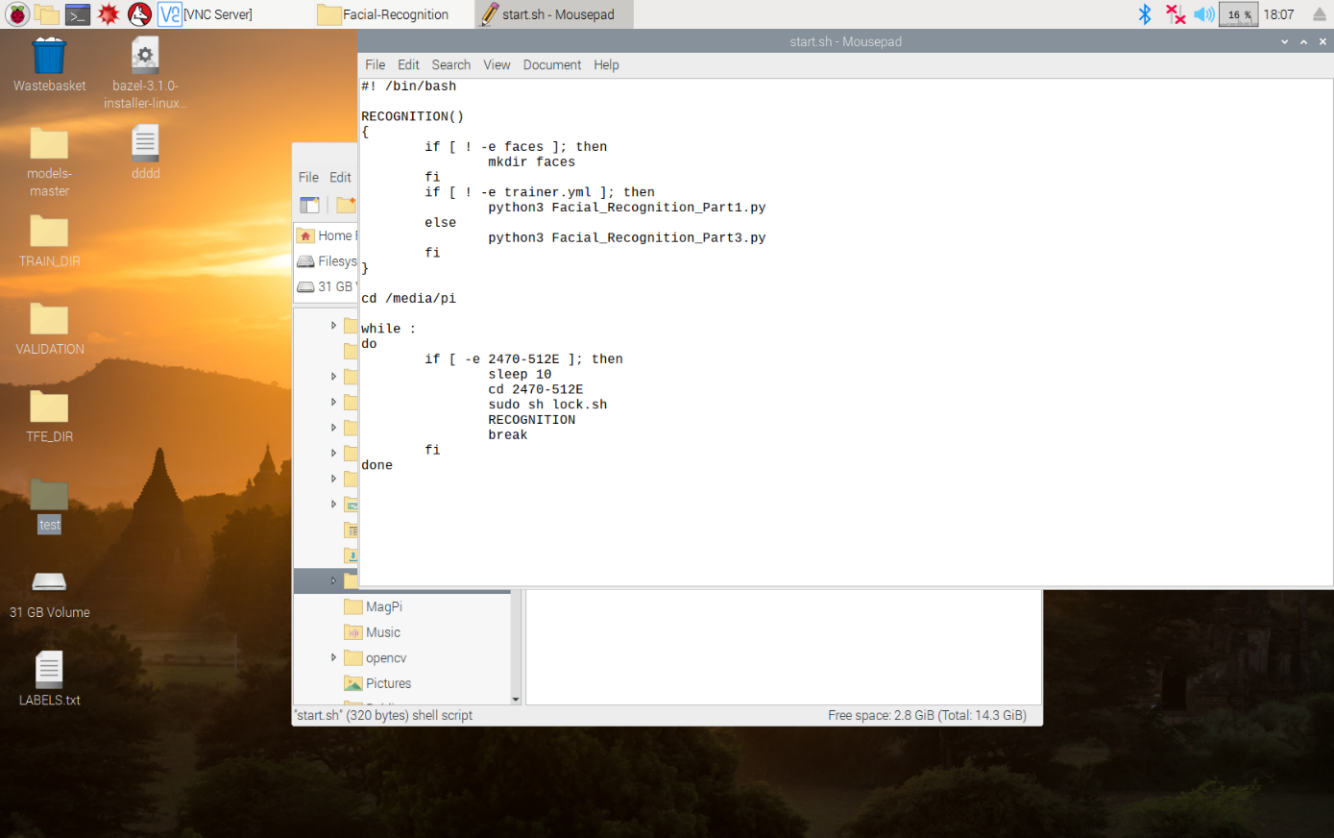


**[rules 실행코드]**

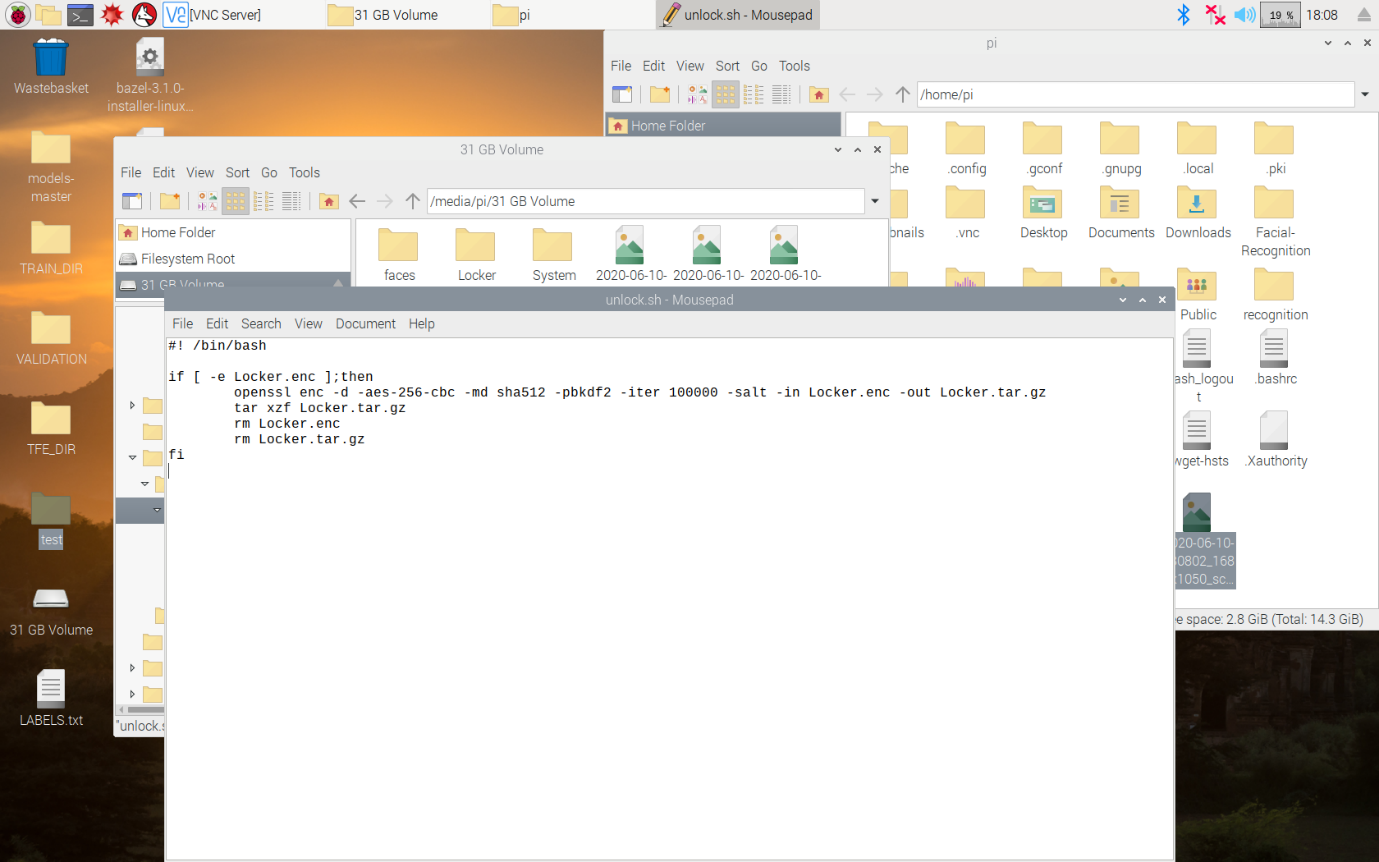


|  |  |
| --- | --- |
| **rules** | |
| ACTION | detect changes in registered devices |
| KERNEL | sending events from the kernel |
| RUN | execute specified file |

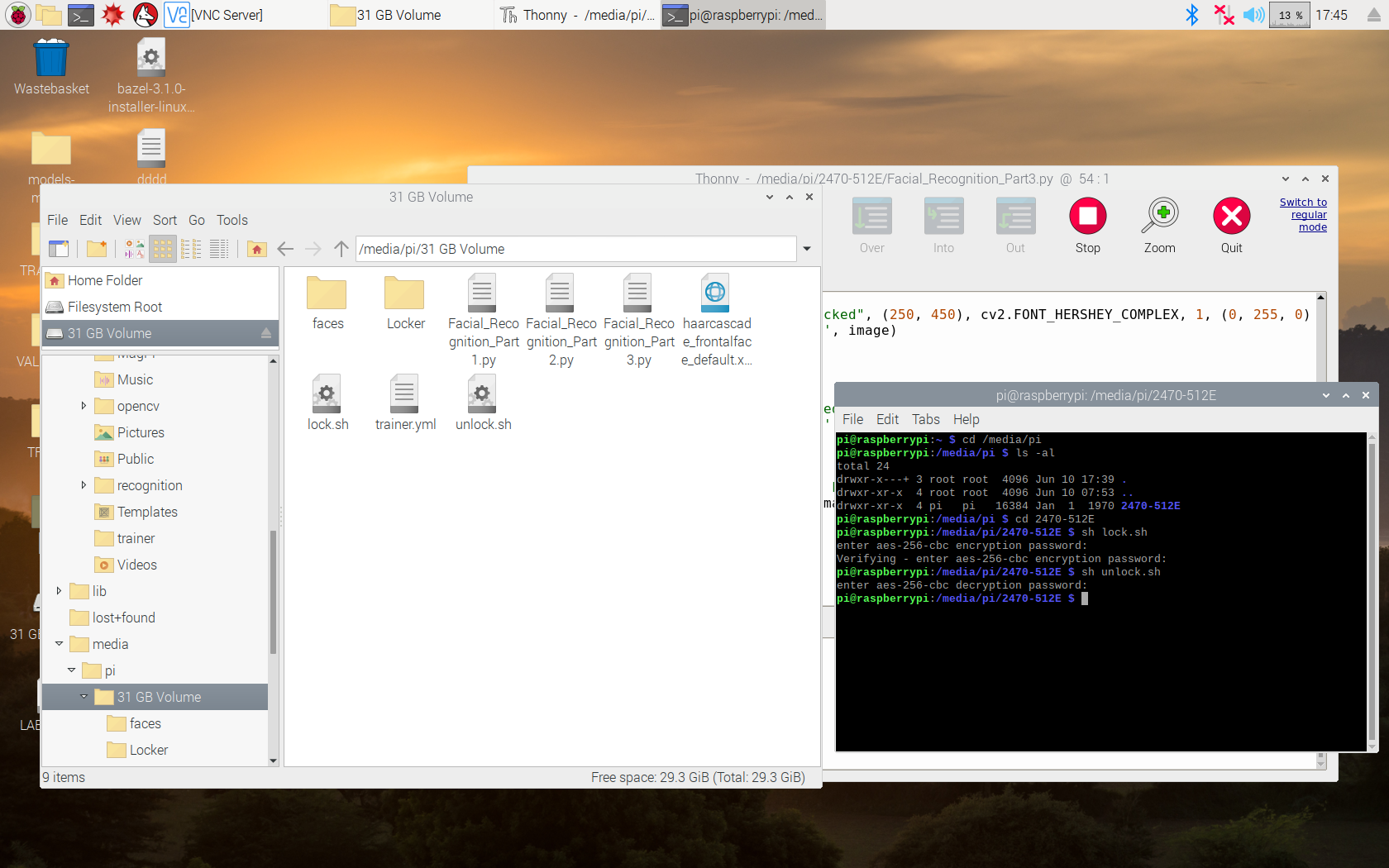
**[start.sh 실행 코드]**



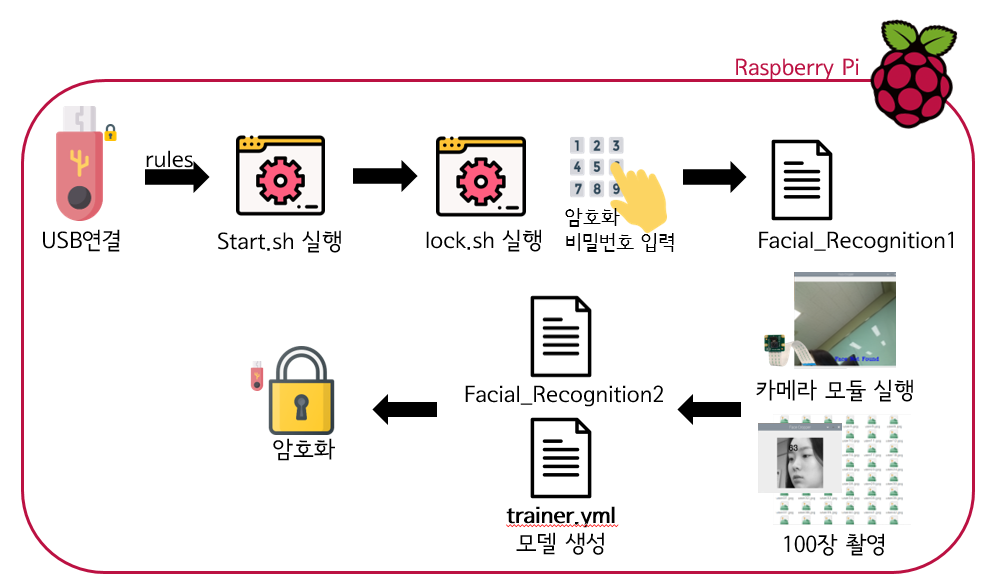
**[Unlock.sh 실행 코드]**



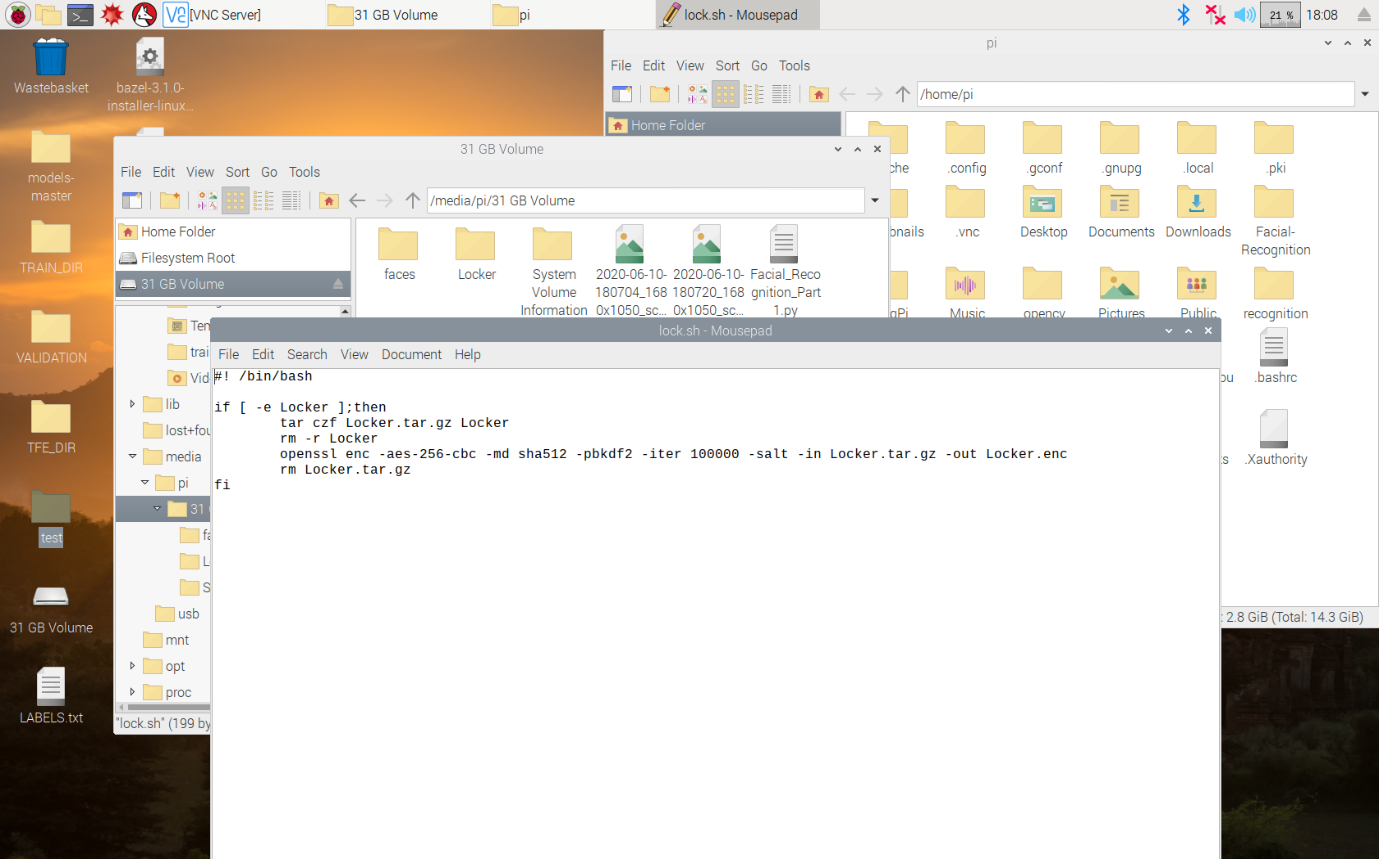
**[복호화 비밀번호 입력창]**



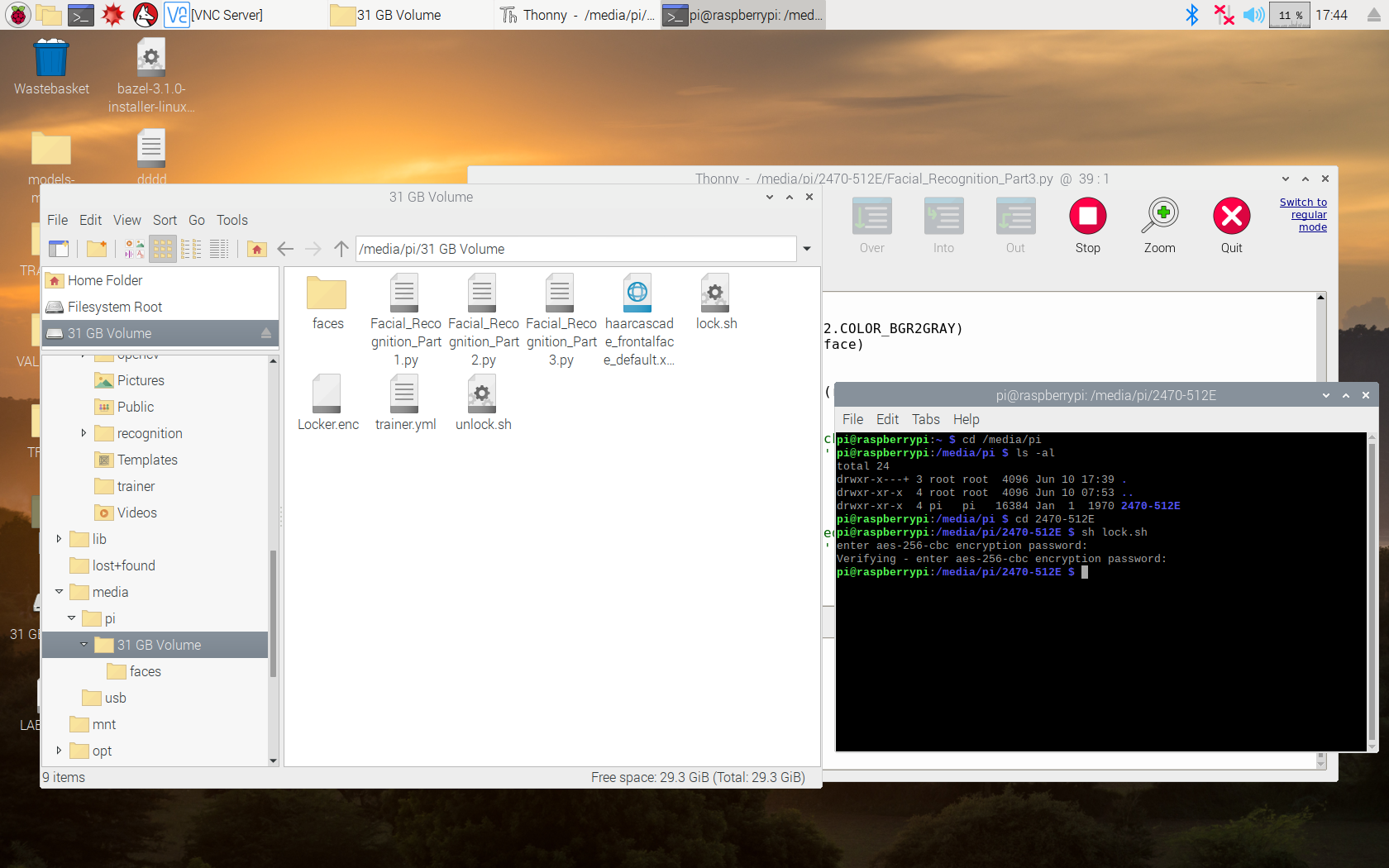
**[ 사전 작업 ]**



Lock.sh 실행 코드 / 암호화할 파일 및 폴더는 Locker라는 이름을 가진 폴더 안에 있어야 합니다.



[ 암호화 비밀번호 입력창 ]



**[ 개발 일정 및 팀원 별 담당업무 ]**

|  |  |  |
| --- | --- | --- |
| **일정** | **담당 업무** | |
| 1주차 | 사이버보안 캡스톤 주제선정 | |
| 2주차 | 주제 구체화 및 일정 정리 | |
| 3주차 | 최종 제안서 작성, 발표 동영상 촬영 | |
| 4주차 | 송경주 | opencv-python facial recognition 사진 학습 부분 코드 분석을 수행함 |
| 백민진 | opencv-python facial recognition 얼굴 추출 부분 코드 분석을 수행함 |
| 강혜진 | aws rekognition api python version 분석을 수행함 |
| 5주차 | 송경주 | opencv-python facial recognition 얼굴 비교 부분 인식률 수정을 수행 |
| 백민진 | opencv-python facial recognition 얼굴 비교 부분 코드 분석을 수행 |
| 강혜진 | 자동 실행 프로그램 만들기를 수행 |
| 6주차 | 송경주 | 자동 실행 시 필요한 exe파일 만들기 및 D드라이브 인식 확인 프로그램코드 분석. |
| 백민진 | USB 연결시 window상의 자동실행 프로그램 autorun을 수행하여 동작 확인. |
| 강혜진 | 우분투 상에서 안면인식을 위한 카메라 연결 방법 모색 및 코드 분석. |
| 7주차 | 송경주 | 라즈베리 파이 상에서 자동 실행할 수 있는 프로그램 코딩 |
| 백민진 | 라즈베리 파이 카메라 모듈 연결 및 동작확인 |
| 강혜진 | 라즈베리 파이 상에서 OpenCV이용하여 사용자 얼굴 받아 오기 |
| 8주차 | 송경주 | 라즈베리 상에서 자동실행 프로그램 동작확인 및 코드수정 |
| 백민진 | 중간 발표 보고서 작성 및 각 코드 오류 수정 |
| 강혜진 | 라즈베리 파이 상에서 OpenCV이용하여 사용자 얼굴 학습시키고 실시간으로 얼굴 구별하는 코드 작성하기 |
| 9주차  (중간발표) | 중간 발표 자료 준비, 발표 동영상 촬영 | |
| 10주차 | 송경주 | 리눅스 상에서 실행할 수 있는 파일 암호화 프로그램 실행 및 확인 |
| 백민진 | 얼굴 인식 하기위해 필요한 디렉토리 자동 생성 쉘 스크립트 작성 |
| 강혜진 | 파일을 zip파일로 변경하여 폴더를 잠그고 푸는 쉘 스크립트 작성 |
| 11주차 | 송경주 | 라즈베리 파이 상에서 자동 실행 여러 코드 같이 실행 |
| 백민진 | 라즈베리 파이 상에서 학습된 얼굴 인식률을 높임 |
| 강혜진 | 얼굴 인식 코드와 zip파일 복호화 스크립트 연동 |
| 12주차 | 송경주 | 자동실행파일 경로를 USB내부 파일로 설정 시도 |
| 백민진 | 다른 사람의 얼굴을 같은 얼굴로 인식하는 문제 해결 |
| 강혜진 | 얼굴인식 후 비밀번호를 이용한 폴더 복호화 |
| 13주차 | 송경주 | 얼굴 학습하는 파일 USB 인식 시 자동 실행하기 |
| 백민진 | 학습할 얼굴 사진 100장 찍는 과정을 한 번만 실행하도록 하기 |
| 강혜진 | 사용자가 파일을 암호화할 수 있는 쉘 스크립트 만들기 |
| 14주차 | 송경주 | USB 인식 시 실행할 모든 프로그램 자동 실행하기 |
| 백민진 | 최종 결과 보고서 개요, 배경, 구성 정리하기 |
| 강혜진 | 얼굴 인증 화면 디자인 정리하기 |

**[ 지원금 집행 내역 ]**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 지원항목 | 세부항목 | 산출내역 | 금액(원) | 비고 |
| 실험실습비 | 재료 구입비 | 카메라 모듈 | 29,700 | 부가세 2700 |
| 회의비 | 회의비 | 스낵(한성대학교 매점) | 10,000 |  |
| 실험실습비 | 재료 구입비 | ELECOM랜케이블 2개, HDMI-HDMI 케이블 2개 | 16,050 | 봉투(소) 50 |
| 회의비 | 회의비 | 한성대학교 CU | 9,520 |  |
| 회의비 | 회의비 | 그라찌에 | 9,000 |  |
| 실험실습비 | 재료 구입비 | 우드락 | 5,000 |  |
| 실험실습비 | 재료 구입비 | 투명양면테이프 | 4,200 |  |

**[ 참고문헌 ]**

라즈베리파이 opencv 설치

<http://www.3demp.com/community/boardDetails.php?cbID=235>

opencv 활용한 얼굴인식

<https://m.blog.naver.com/PostView.nhn?blogId=ljy9378&logNo=221440034735&targetKeyword=&targetRecommendationCode=1>

openssl 대칭 키<https://zetawiki.com/wiki/%EB%A6%AC%EB%88%85%EC%8A%A4_openssl_%EB%8C%80%EC%B9%AD%ED%82%A4_%EC%95%94%EB%B3%B5%ED%98%B8%ED%99%94_%EC%8B%A4%EC%8A%B5>

pckdf2

<https://d2.naver.com/helloworld/318732>

shell 문법

<http://calab.hanyang.ac.kr/courses/SP_taesoo/prct_04_shell_programming.pdf>

rules 규칙

<https://mokga.tistory.com/54>

<https://kldp.org/node/154396>

<http://kjjeon.blogspot.com/2015/05/udev-usb-mount.html>

**[ 프로젝트 수행 후기 ]**

**백 민 진**

프로젝트를 진행하면서 os환경에 따라 수행되지 않거나 구현하기에 어려움이 있는 기능들로 인해 여러 환경에서 시도해보았으며, 많은 시행착오 끝에 라즈베리파이-라즈비안 환경에서 원하는 기능들을 구현할 수 있었습니다. 이 덕분에 윈도우의 bat파일, 리눅스의 shell파일 같이 각 환경간의 차이점과 그 특징에 대해서 알 수 있었습니다. 강의를 통해 배웠던 내용들을 다시 한 번 정리하고 직접 프로젝트 개발을 위해 적용해보면서 온전한 저의 것으로 만들 수 있었습니다. 팀원들과 서로 소통하며 프로젝트를 진행하여 커뮤니케이션 능력도 키울 수 있었습니다.

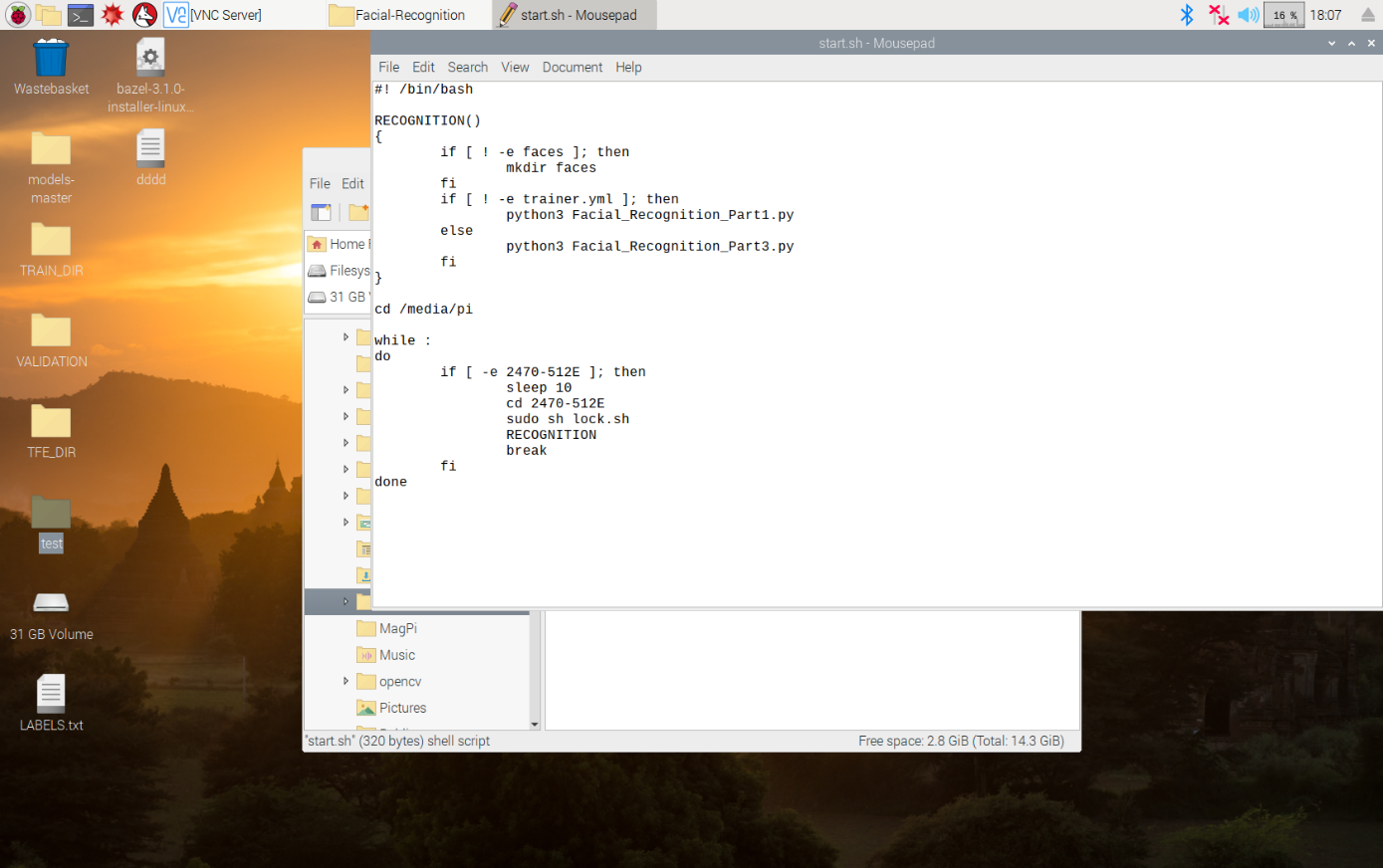
**송 경 주**

프로젝트를 진행하며 수업에서 배운 내용을 응용하고 오류 해결을 통해 더 깊은 이해를 할 수 있었습니다. 리눅스 환경에서의 코딩은 수업시간에서 많이 배워 왔지만 간단한 코드 이해와 주어진 프로그램 실행정도로만 실습했기 때문에 응용하는 방법에 대해서는 잘 알지 못하였습니다. 하지만 프로젝트를 통해 수업 때 배웠던 명령어(chmod, sh, ls -al 등)를 직접 적용해보고 단순히 외우기만 했던 명령어나 오류를 겪으며 적용해 보니 외우기도 더 쉽고 실행 방식에 대한 이해도 높아졌습니다. 프로젝트 진행 전에는 리눅스 권한변경이 왜 필요한 것인지 알지 못했고 쓸 일도 별로 없을 것이라 생각했지만 리눅스 환경에서 무언가를 할 때마다 권한변경이 굉장히 중요하다는 것도 깨달았습니다. 또한 시스템 내부에서 메모리 규칙사항을 사용자 임의로 rules 작성하여 정할 수 있다는 것도 알게 되었습니다. 팀원들과 함께 오류를 해결하고 프로그램을 구현하면서 팀워크의 중요성과 일 분배의 효율성도 배울 수 있었습니다.

**강 혜 진**

라즈베리파이-라즈비안 환경에서 원하는 프로그램을 실행시키기 위해 user와 linux 사이의 인터페이스로 작동하는 프로그램인 shell을 사용하고 파일 또는 폴더를 암호화하기 위해 openssl을 사용했습니다. 적합한 암호화를 하기 위해 openssl에 줄 수 있는 옵션을(aes-256, pbkdf2, sha512, salt) 찾아보고 공부하였습니다. 캡스톤 디자인을 진행하면서 강의에서 이론으로 다루었던 내용을(암호화 및 복호화) 직접 코드로 적용하며 배울 수 있었고, shell 문법에 대해 여러 자료를 찾아보고 script를 작성하며 공부하는 시간이 됐습니다. 또한 주차마다 개발 일정을 정리하면서 해당 주의 목표를 뚜렷하게 하고 프로젝트를 진행에 집중할 수 있었습니다.

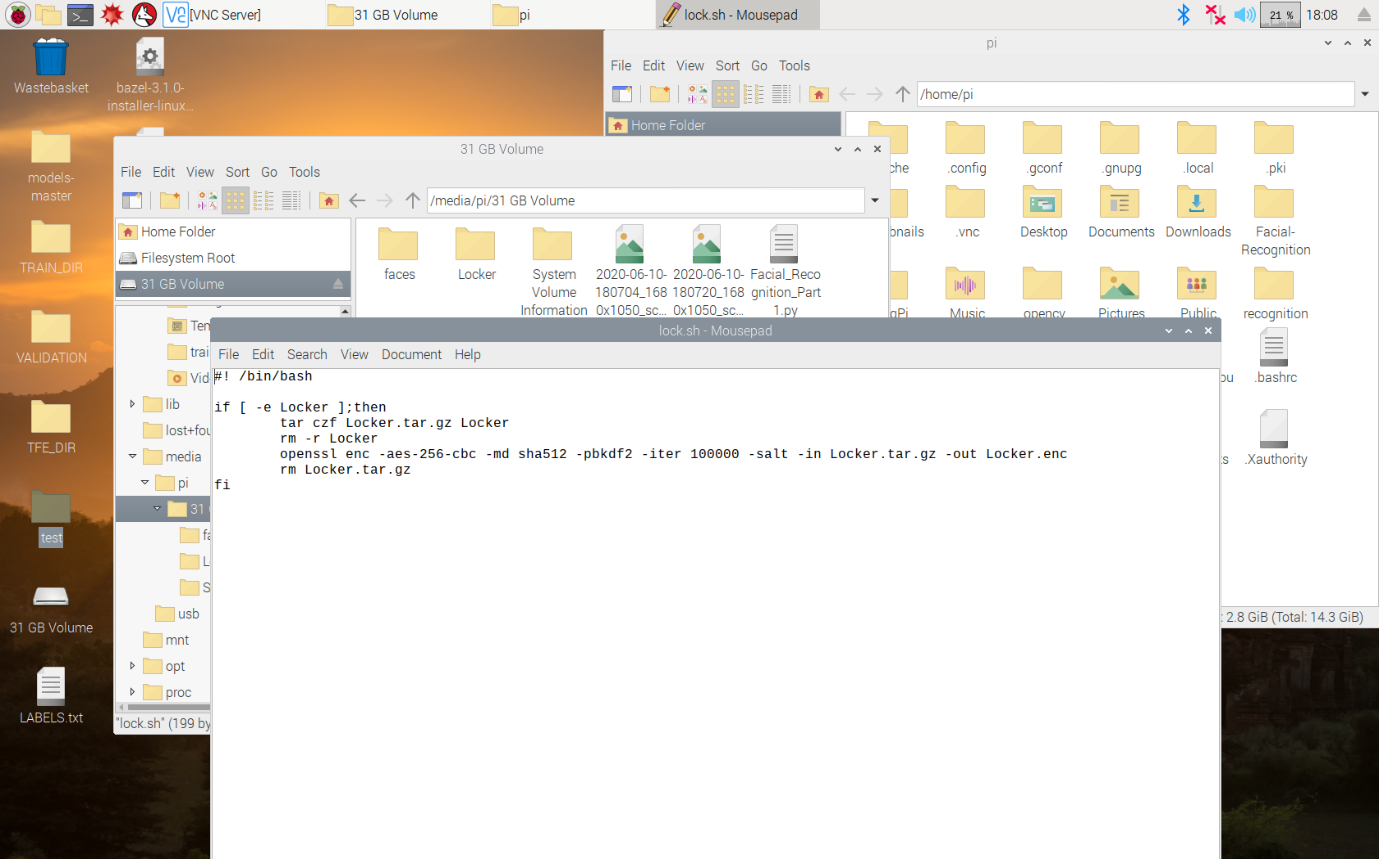
**[ 부록 ]**



usb 경로

<start.sh>

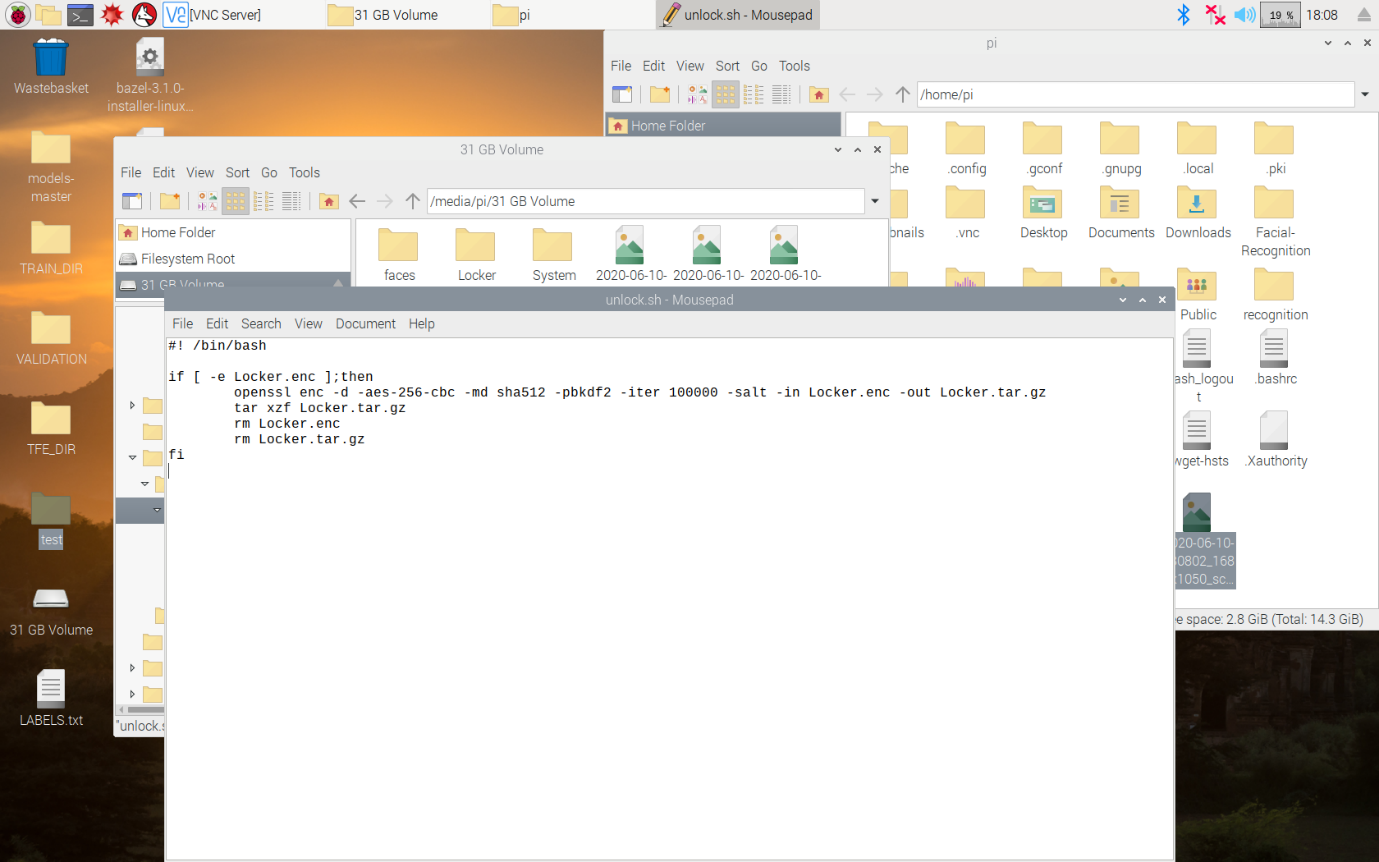
usb 경로가 존재하면 usb 경로로 이동하고 lock.sh를 실행하고 RECOGNITION함수를 실행합니다. RECOGNITION 함수는 사용자의 얼굴을 저장할 faces 폴더가 없으면 만들어주고 얼굴 학습한 결과인 trainer.yml이 없으면 Facial\_Recognition\_Part1.py을 실행, 그렇지 않으면 Facial\_Recogntion\_Part3.py를 실행합니다.



<lock.sh>

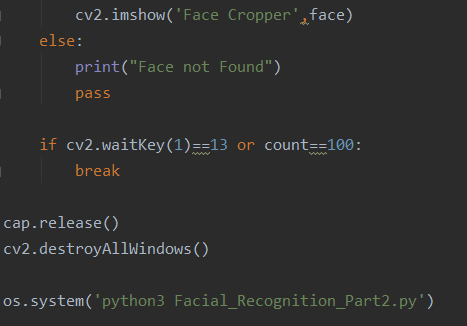
Locker 폴더가 존재하면 Locker 폴더를 Locker.tar.gz 파일로 만들어주고 남아있는 Locker 폴더를 지워줍니다. openssl을 이용하여 Locker.tar.gz를 암호화하여 결과를 Locker.enc로 저장합니다. 남아있는 Locker.tar.gz 파일을 지워줍니다. 아래는 openssl의 옵션 구조이고 굵게 표시된 옵션을 사용하였습니다. -aes-256 cbc 모드로 암호화하고 sha512 해쉬 함수를 사용합니다. 솔트를 적용한 후 해쉬 함수의 반복 횟수를 100000번으로 선택합니다.

**openssl enc -cipher** [-help] [-ciphers] **[-in filename] [-out filename]** [-pass arg] [-e] [-d] [-a] [-base64] [-A] [-k password] [-kfile filename] [-K key] [-iv IV] [-S salt] **[-salt]** [-nosalt] [-z] **[-md digest] [-iter count] [-pbkdf2]** [-p] [-P] [-bufsize number] [-nopad] [-debug] [-none] [-rand file...] [-writerand file] [-engine id]

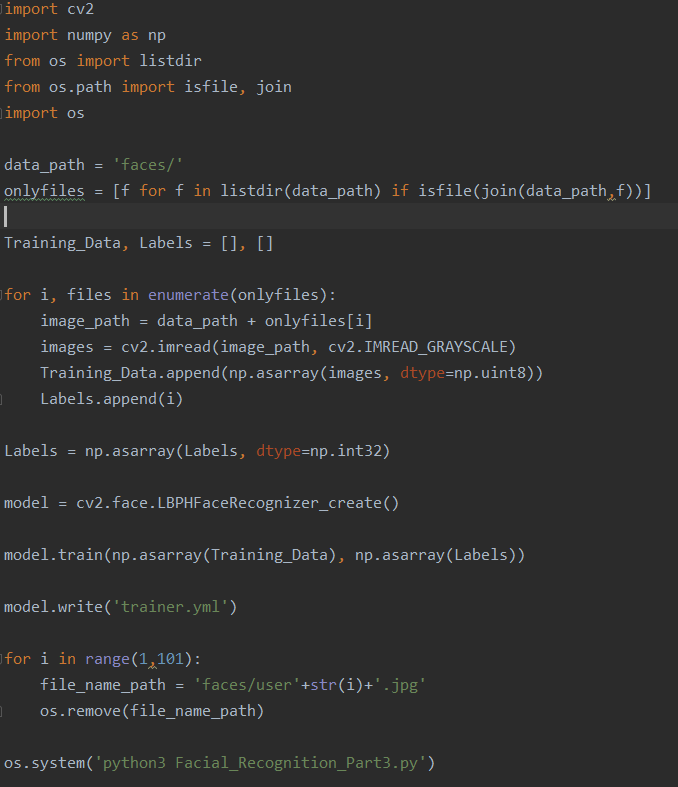


<unlock.sh>

Locker 폴더를 암호화한 Locker.enc 파일이 있으면 openssl을 이용하여 복호화하고 복호화한 결과를 Locker.tar.gz 파일로 저장합니다. Locker.tar.gz 파일을 Locker 폴더로 만들어줍니다. 남아 있는 Locker.enc와 Locker.tar.gz 파일을 지워줍니다.

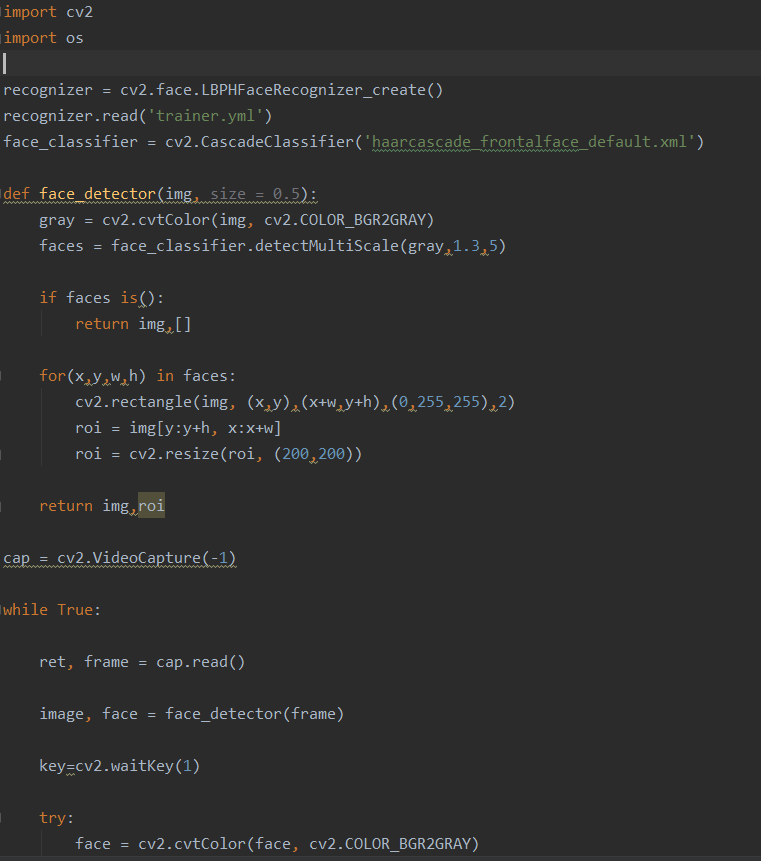
<Facial\_Recognition\_Part1.py>

얼굴 검출을 위해 hararcascade 트레이닝 데이터를 읽어 CascadeClassifier 객체를 생성합니다. CascadeClassifier의 detectMultiScale 함수에 grayscale 이미지를 입력하여 얼굴을 검출합니다. 얼굴이 검출되면 얼굴의 위치를 리스트로 리턴합니다. (x , y)는 검출된 얼굴의 좌 상단, w, h는 가로와 세로의 크기입니다. 영상에서 사람의 얼굴을 인식하면 얼굴 부분을 캡처하여 faces 디렉토리에 usercount.jpg로 저장합니다. 100장을(count) 찍으면 카메라를 종료하고 시스템 명령어 호출 os.system(‘명령어”)를 이용하여 Facial\_Recognition\_Part2.py를 불러옵니다.



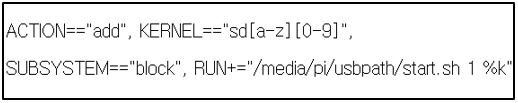
<Facial\_Recognition\_Part2.py>

faces 디렉토리에 존재하는 이미지 파일을 불러옵니다. 이를 model.train()을 통해 학습시키고 그 결과를 model.write()를 통해 trainer.yml 파일로 저장합니다. 시스템 명령어 호출 os.system(‘명령어”)를 이용하여 Facial\_Recognition\_Part3.py를 불러옵니다.



<Facial\_Recognition\_Part3.py>

recognizer.predict()를 통해 trainer.yml에 저장된 데이터와 현재 이미지 얼마나 유사한지 확인합니다. 유사한 정도(confidence)가 82 초과일 경우 반복문을 종료합니다. 시스템 명령어 호출 os.system(‘명령어”)를 이용하여 복호화 코드가 있는 unlock.sh를 불러옵니다.



<10-local.rule>

ACTION 을 사용해 USB 메모리 장치의 연결이 발생하면 KERNEL을 통해 kernel에 event를 보내 RUN에 지정한 파일을 실행하도록 작성한 rules 파일입니다. USB 메모리가 연결된 것이 확인되면 start.sh을 동작시킵니다.