

**감염병 확산 방지를 위한**

**Deep running 얼굴 인식 기반의   
출입관리 및 전자명부**



**팀 이름: 비욘드 마스크**

**안형수**

**E-MAIL :** [**crystalhs22@naver.com**](mailto:crystalhs22@naver.com)

**☏ : 010-2827-8792**

**심은용**

**E-MAIL : hbyseyhi**[**@gmail.com**](mailto:ktkzzabg@naver.com)

**☏ : 010-9780-5386**

**조하나**

**E-MAIL :** [**hana8951@gmail.com**](mailto:hana8951@gmail.com)

**☏: 010-8598-88951**

**최규범**

**E-MAIL :** [**rbqjachl95@gmail.com**](mailto:rbqjachl95@gmail.com)

**☏: 010-3997-8450**

**통합 코드**: <https://github.com/GGuiGui/electronic_list_project.git>

**시연 영상**: <https://www.youtube.com/watch?v=NI0HpAQ9JUc>

**한국직업능력교육원 안산**

내용

[**1.** **개요** 1](#_Toc104547061)

[**1.1 프로젝트 명** 1](#_Toc104547062)

[**1.2 팀 명** 1](#_Toc104547063)

[**1.3 프로젝트 기간** 1](#_Toc104547064)

[**1.4 프로젝트 추진 배경** 1](#_Toc104547065)

[**1.5 프로젝트 목적** 1](#_Toc104547066)

[**1.6 프로젝트 기대 효과** 1](#_Toc104547067)

[**2. 프로젝트 역할 분배** 2](#_Toc104547068)

[**2.1 프로젝트 역할 조직도** 2](#_Toc104547069)

[**3. 프로젝트 작업 일정** 4](#_Toc104547070)

[**3.1 작업 계획** 4](#_Toc104547071)

[**3.2 온도 이상 상태의 사람 출입 통제 방법** 4](#_Toc104547072)

[**4. 프로젝트 범위** 5](#_Toc104547073)

[**4.1 작업명세서** 5](#_Toc104547074)

[**5. 시스템 구성** 6](#_Toc104547075)

[**5.1 하드웨어 구축** 6](#_Toc104547076)

[**5.2 소프트웨어 구축** 7](#_Toc104547077)

[**6. 프로젝트 관리** 9](#_Toc104547078)

[**6.1 프로젝트 세부 내용** 9](#_Toc104547079)

[**6.2 프로젝트 세부 내용** 10](#_Toc104547080)

[**6.3 Troubleshooting** 11](#_Toc104547081)

[**6.4** **프로젝트 성능평가 결과** 13](#_Toc104547082)

[**6.5 프로젝트 후기** 14](#_Toc104547083)

[**7. 프로젝트 핵심 코드 및 참고 자료** 15](#_Toc104547084)

[**7.1 Raspberry 관련** 15](#_Toc104547085)

[**7.2 Detecting-Computer 관련** 17](#_Toc104547089)

[**7.3 Cloud-Server 관련** 21](#_Toc104547095)

# **개요**

## **1.1 프로젝트 명:** 감염병 확산 방지를 위한 딥러닝 얼굴인식 기반 출입 관리 및 전자 명부

- “감염병 확산 방지를 위한 딥러닝 얼굴인식 기반 출입 관리 및 전자 명부”의 선정 이유는 프로젝트의 목적 및 배경을 직관적으로 파악이 가능하고 사용법을 떠올리기에 명확하다고 판단하여 채택하게 되었다.

## **1.2 팀 명:** 비욘드 마스크

- 비욘드 마스크의 의미는 “마스크를 초월한”이라는 뜻으로 감염병을 극복한다는 의미와 마스크를 쓰고 있어도 누구인지 파악이 가능하다는 의미를 담아 선정하였다.

## **1.3 프로젝트 기간**

- 2022.02.28 ~ 2022.06.10

## **1.4 프로젝트 추진 배경**

- WHO는 2020년 3월 11일 코로나19에 대한 세계적인 대유행, 팬더믹을 선언하였다. 경제 및 사회적 활동으로 건물 출입을 해야 하는 상황이 발생하고, 대면 접촉으로 인해 바이러스가 확산되었다. 이에 사회적 거리두기 및 QR코드 출입으로 대응하였으나, 위치 추적 등 현실적인 문제로 고무적인 효과를 얻기 어려웠다. 이에 대한 원인이 기존 출입 시스템의 문제라 판단하였고, 이 문제점을 해결하기 위해서 “감염병 확산 방지를 위한 출입 통제 및 전자 명부” 프로젝트를 추진하게 되었다.

## **1.5 프로젝트 목적**

- 감염병의 주요 증상의 공통점은 정상 체온보다 높은 발열이라고 판단했다. 이에 대한 대응으로 열화상 카메라와 초음파 센서를 사용, 감염자의 방문 위치 추적을 위해서 학습 모델을 통해 인물을 인식하고 자동으로 전자 명부를 작성, 웹서버의 검색 기능으로 위치 추적을 간편하게 수행하여 최종적으로 기존 QR코드 출입 시스템을 대체할 수 있는 출입 시스템을 만드는 것이 목표이다.

## **1.6 프로젝트 기대 효과**

- 위의 프로젝트는 기존 QR코드 방식과 달리 방문자가 스마트폰을 조작할 필요가 없기 때문에 구세대를 포함한 스마트폰에 익숙하지 않은 사람들도 문제없이 사용할 수 있고, 기존 방식(온도측정 -> 정보 수기 작성)을 자동화했기 때문에 불필요한 인력 소모 감소 및 시간 단축의 효과를 볼 수 있다.

또한, 클라우드 서버의 웹서버를 통한 검색기능으로 공간의 제약 없이 관리자가 감염자 출입 여부 파악이 용이하여 위치추적에 효과적일 것으로 판단된다.

# **2. 프로젝트 역할 분배**

## **2.1 프로젝트 역할 조직도**

- 프로젝트 조직

|  |
| --- |
| C:\Users\IOT\Desktop\조직도.jpg |
| 조직도 |

**2.2 조직별 역할**

- 각 역할

|  |  |
| --- | --- |
| **성명** | **직무** |
| 안형수 | - 프로젝트 아이디어 제시  - 케이스 모델 아이디어 제시  - mjpg-streamer 환경 구축  - [Pyqt5 GUI 구성](#_GUI_QThread_및)  - 라즈베리파이와 검출 컴퓨터 연결 코드 작성  - 모델 학습 자료 수집 및 학습 모델 생성 |
| 심은용 | - 케이스 모델 아이디어 제시  - 간트차트 작성  - mjpg-streamer 환경 구축  - 열화상, [초음파 센서](#_거리_측정) 코드 작성  - [적절한 온도 보정값 측정](#_온도_측정_및)  - 하드웨어 및 케이스 설계  - 라즈베리파이와 검출 컴퓨터 연결 코드 작성(socket 통신)  - 열화상 이미지 프로세싱  - TTS 프로그래밍 |
| 최규범 | - 프로젝트 구현 소프트웨어 구성 방식 및 구현 기술 제시  - flowchart 작성, 포트폴리오 작성  - 모델 [학습 자료 수집](#_학습_데이터_수집) 및 [학습 모델 생성](#_학습_시키기)  **-** 검출컴퓨터의 [전체 코드](#_다중_검증을_통한) 작성([DB연동](#_DB에_사진_자료), cvlib, 검출, 소켓 등)  **-** [카카오톡](#_카카오_메시지_전송) 나에게 메시지 보내기 전후 작업 및 코드 작성  - 라즈베리파이와 검출 컴퓨터 연결 코드 작성  - 통신 환경 구성(포트포워딩)  - 모델 케이스 3D 프린트 출력  - Azure 관련  - Azure 환경 구축 서포트 |
| 조하나 | - web 페이지 구성 틀 제시  - 대시보드 웹 페이지 반응형 제작  - 대시보드 웹 페이지 로그인 기능 구현  - 대시보드 [mysql 연동](#_Mysql_연동_및) 및 [테이블 검색](#_Mysql_데이터_가져오기), 페이징 구현  - 그라파나 차트 구현 및 [대시보드 삽입](#_차트_구현)  - Azure 서버 관리  - Azure 관련  - mysql 서버 구축 및 테이블 작성, 스케줄러를 통한 [자동화](#_자동삭제_기능)  - Azure 환경 구축(apache2, php, phpmyadmin, grafana) |

# **3. 프로젝트 작업 일정**

## **3.1 작업 계획**

- Gantt chart

|  |
| --- |
| C:\Users\IOT\Desktop\간트차트.png |
| 간트 차트 |

## **3.2 온도 이상 상태의 사람 출입 통제 방법**

- 온도 이상이 발생하면 디스플레이에 출입 거부 문구가 뜨며, 추가로 관리자 핸드폰 카카오톡에 “~가 ~도 로 출입 통제되었습니다.” 라는 메시지가 전송되고 카카오 메시지의 자세히 보기를 터치하면 Azure의 WEB 서버에 접속되어 로그인 후 전자 명부 확인이 가능하다. (카카오 메시지를 통해 여러 방향성 제시 가능)

# **4. 프로젝트 범위**

## **4.1 작업명세서**

- 자료 수집(raspberry-pi)

|  |  |
| --- | --- |
| **업무** | **업무 범위** |
| 자료 수집 | - LS-2718(일반 카메라) mjpg-server [영상 자료]  - HC-SR04(초음파 센서) [온도의 거리에 따른 보정값]  - MLX90641(열화상 카메라) [온도 측정] |
| 통신 환경 | - socket 클라이언트 [검출 컴퓨터 통신 환경] |
| 방문자 안내 | - 7 inch LCD HDMI Display-c(디스플레이) [사용자 안내용]  - Pyqt5 [인터페이스 화면 구성] |
| 하드웨어 | - 부품 모델링, 케이스 설계 및 3D 프린트 제작, 부품 조립 및 배선 |
| 문서 | - Gantt chart, 자재명세서 |

- 자료 분석(Detecting-computer)

|  |  |
| --- | --- |
| **업무** | **업무 범위** |
| Tensorflow 모델 학습 | - 모델 학습에 필요한 인물 자료 수집 및 데이터 처리  - 검출 모델 학습 |
| 자료 분석 | - mjpg-client [영상을 받음]  - cvlib [영상에서 사람의 얼굴 검출]  - tensorflow.keras [모델을 통한 사람 판별] |
| 통신 환경 | - socket 서버 [라즈베리 파이 통신 환경]  - pymysql [Cloud-server의 DB와 연동]  - 카카오톡 API [온도 이상에 따른 관리자 메시지 전송] |
| 테스트 및 안정화 | - 라즈베리파이와 연동하여 문제점 파악  - 검출결과 Cloud DB 확인 |

- 자료 활용(Cloud-server[Azure])

|  |  |
| --- | --- |
| **업무** | **업무 범위** |
| 데이터 베이스 | - table 구성 및 자동화 기능  - phpmyadmin [web과 연동]  - 외부 접근 허용 등 환경 구축 |
| 그라파나 | - 데이터 베이스와 연동, 환경 구축  - 시계열에 대한 그래프 및 시각화 그래프 작성 [web과 연동] |
| 웹 서버 | - mjpg-client [영상정보 띄우기]  - DB 정보 가져오기  - 검색 기능 및 자료 시각화 |

# **5. 시스템 구성**

## **5.1 하드웨어 구축**

- 하드웨어 구성

|  |
| --- |
| C:\Users\IOT\Desktop\hard-ware.jpg |
| 하드웨어 구상도 |

|  |  |
| --- | --- |
| C:\Users\IOT\AppData\Local\Microsoft\Windows\INetCache\Content.Word\20220520_135717.jpg | C:\Users\IOT\AppData\Local\Microsoft\Windows\INetCache\Content.Word\20220520_135723.jpg |
| 하드웨어 실물 사진 | |

- 하드웨어 구성 현황

|  |  |  |
| --- | --- | --- |
| **하드웨어** | **모델명** | **상세 설명** |
| 일반 카메라 | LS-2718 | 일반 영상 카메라 |
| 열화상 카메라 | MLX90641 | 온도 측정 카메라 |
| 디스플레이 | 7 inch LCD HDMI | 검출컴퓨터에서 보낸 sign에 따른 방문자 안내 |
| 초음파 센서 | HC-SR04 | 사람이 떨어진 거리를 측정 |
| RaspberryPi4 | RaspberryPi4 | Mjpg-stream 서버 개설, 안내 및 소켓 코드 작성 |
| 모델 케이스 | - | 3D 프린트로 출력한 하드웨어 케이스 |
| Detecting-com | ASUS-notebook | 자료 분석을 담당할 개인 노트북 |
| Cloud-server | Azure | 데이터베이스 및 WEB서버를 서비스하는 Cloud-server |

## **5.2 소프트웨어 구축**

- 소프트웨어 구성

|  |
| --- |
| C:\Users\IOT\Downloads\SW.jpg |
| 소프트웨어 구상도 |

- 소프트웨어 구성 현황

|  |  |  |
| --- | --- | --- |
| **하드웨어 구분** | **소프트웨어** | **상세 설명** |
| RaspberryPi4 | Mjpg-streamer | - 영상 데이터를 Detecting-com과 Cloud-server에서 사용하기 위해 Mjpg-streamer server 개설  - 부팅과 동시에 실행 설정 |
| Python3 | - 방문자 안내를 위해 Pyqt5모듈기반 GUI 작성  - Detecting-computer와 소켓 통신을 통해 온도 요청을 받음  - 초음파 센서와 열화상 카메라를 통해 온도 측정값 송신  - Detecting-computer의 송신 값에 따른 안내 문구 띄우기 |
| Detecting-com | Python3 | - mjpg-server를 통해 영상 정보 가져오기  - 영상을 cvlib모듈을 이용 사람 얼굴 부분만 가져오기  - cvlib의 결과를 학습 모델에 넣어서 사람 판별 |
| Detecting-com | Python3 | - raspberry에 온도값을 받아서 상태 여부 판별  - 각 상태에 따른 sign raspberry에 송신  - 온도 이상의 경우 kakao 메시지 보내기  - 정리된 자료 Cloud-server의 DB에 추가시키기 |
| Kakao-API | - 나에게 메시지 보내기 기능 구현  (code 발급받기 => original Token 발급 => 사용자 정보 조회 => access Token 발급) |
| Cloud-server | Mysql | - 외부 접근 허용 등 환경 구축  - 등록된 사용자 신상정보 table과 전자 명부 table 생성 및 작성  - 전자 명부 만료 기간에 따른 데이터 삭제 및 index 재정렬[자동화] |
| Apache2, php | - phpmyadmin [web과 mysql 연동]  - mjpg-server를 통해 영상 정보 가져오기  - 로그인 페이지 구성(영상 띄우기)  - 로그인 후 메인 페이지 구성(테이블 검색 기능, 시각화 그래프) |
| Grapana | - mysql 연동  - 시계열 그래프 및 시각화 그래프 작성 |

# **6. 프로젝트 관리**

## **6.1 프로젝트 세부 내용**

- 순서도

|  |
| --- |
| C:\Users\IOT\Downloads\dd.jpg |
| Flow chart |

- 장비 작동 시연

a. 각 유형에 따른 디스플레이 화면

|  |  |  |
| --- | --- | --- |
| **사례** | **온도 정상, 등록자** | **온도 정상, 미 등록자** |
| **Display** | C:\Users\IOT\AppData\Local\Microsoft\Windows\INetCache\Content.Word\20220527_112415.jpg | C:\Users\IOT\AppData\Local\Microsoft\Windows\INetCache\Content.Word\20220527_111026.jpg |
| **Kakao**  **[온도**  **이상]** | C:\Users\IOT\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20220511-162438_KakaoTalk.jpg | C:\Users\IOT\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20220511-162438_KakaoTalk.jpg |

b. web서버 접속 시연

|  |  |
| --- | --- |
| **로그인 화면** | **메인 화면** |
| C:\Users\IOT\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20220520-151411_KakaoTalk.jpg |  |

## **6.2 프로젝트 세부 내용**

- 주의사항

1) 작동순서의 주의 사항

- raspberry 부팅 -> Azure 부팅 -> Detecting-computer 부팅 및 코드 실행 -> raspberry 코드 실행

- raspberry가 부팅되면 자동으로 mjpg-server가 오픈되기 때문에 먼저 부팅되어야 하며, 코드를 맨 마지막에 실행하는 이유는 Detecting-computer가 소켓 통신의 서버가 되기 때문이다.

2) 제품 배선 관련 주의 사항

- raspberry의 전원공급 방식이 유선으로 되어있어 전원 공급이 끊기면 소켓 통신을 다시 연결해야 하므로 Detecting-computer의 코드 또한 다시 실행해야 한다. (Detecting-computer의 나머지 코드가 실행되지 않음)

## **6.3 Troubleshooting**

- 해결한 문제

1) Raspberry와 Detecting-computer간 동기화

(1) 검출이 끝난 대상을 바로 재검출하는 현상 [검출과 다음 검출의 term 부재]

(2) 해결 방법

* DB에 내용 추가 후 interrupt switch 변수 생성 및 시간 측정, 사람 얼굴 인식 시의 시간을 측정하여 일정 시간 이후 switch를 True로 전환하여 검출 실행

2) 초음파 센서에 과도한 접근 시 기기가 멈추는 현상

(1) 초음파 센서에 과도한 접근 시 무한 루프에 빠지는 현상

(2) 해결 방법

* 일정 시간 무한 루프 발생 시 루프를 빠져나가게 if문 설정

3) 대시보드의 시각화 그래프 생성의 어려움

(1) mysql과 json을 통한 데이터 연동이 익숙하지 않아 원하는 그래프 구현하는데 많은 시간이 소요됨

(2) 해결 방법

* Grafana를 이용하여 실시간 그래프 구현

4) Grafana와 Flask 호환성 문제

(1) Grafana 그래프가 Flask 서버에서 iframe태그로 구현이 되지 않음

(2) 해결 방법

* 기존 사용 예정이었던 Flask 대신 apache를 사용하여 web-server 구축

5) Raspberry 사양의 한계

(1) Raspberry에서 사람의 얼굴 부분을 인식하려 했으나 사양의 문제로 속도 저하

(2) 해결 방법

* Detecting-computer에서 사람의 얼굴 부분 인식 코드를 구현하여 Raspberry의 부담 감소 및 mjpg fps 조절, overclock 활용

6) 학습 모델의 결과 신뢰도 문제

(1) 학습 모델 자체의 검출 정확도 문제

(2) 해결 방법

* 코딩으로 총 5번을 검출하여 개수와 확률의 합이 가장 높고, 일정 확률 이상일 경우에만 검출된 것으로 알고리즘 구현(교차 검증과 비슷한 원리) 그 외, 미검출 대상
* 학습데이터 장소와 검출 장소의 조감도 영향이 많은 것으로 판단, 학습과 검출 장소 통일
* 그러나 근본적인 문제인 학습 class 데이터 부족 문제로 인한 검출 정확도는 높지 않음

7) 데이터베이스에 사진 저장 문제

(1) 기존 cv2의 frame을 byte로 변환하여 DB에 저장했으나 사진이 깨지는 현상 발생

(2) 해결 방법

* Frame을 파일로 저장하여 byte형으로 불러오고 DB에 저장하여 정상 출력됨

8) Kakao Token 1~2일 후 사용 불가 현상

(1) 카카오 나에게 메시지 보내기의 사용 토큰이 refresh 기간 전에 만료되는 현상

(2) 해결 방법

* 카카오 데브톡에 문의하여 원인 파악 후 3단계 사용자 정보 조회를 통해 카카오 서버에 세션을 생성하여 정상 작동

9) Pyqt5의 자동 문구 변경 문제

(1) GUI 구조상 사용자 반응형 동작으로 자동 문구 변경이 어려움

(2) 해결 방법

* Threading을 통한 label을 변경하여 자동 문구 변경 구현

10) 데이터베이스 자동화

(1) mysql 자동화 지식 부족

(2) 해결 방법

* 검색을 통해 프로시저와 스케줄의 예제 수집 후 적용하여 자동화 구현

11) 온도 측정값과 실제 값의 오차

(1) 열화상 카메라의 온도 값과 실제 온도 값의 오차 확인

(2) 해결 방법

* 두 온도 차의 원인이 거리의 요소가 큰 영향을 끼치는 것으로 판단, 초음파 센서 추가 배치 및 적절한 보정 값 적용

12) 모델 설계 시 3D프린터의 사양 및 성능을 간과한 설계

(1) pcb 부품 서포트 파손, 최대 제작 사이즈를 고려하지 않아 다시 설계

(2) 해결 방법

* pcb 부품 서포트 하부 보강, 사이즈를 고려한 분할 제작 후 결합

13) 하드코딩

(1) IP, PATH 등의 환경설정 부분이 코드 내부에 존재하여 관리의 어려움

(2) 해결 방법

* 환경설정에 대한 대부분의 요소들을 JSON파일에 dic형태로 저장하고 불러오는 함수를 생성하여 json파일로 환경을 설정함

14) Raspberry의 GUI 멈춤 튕김 현상

(1) 파악하기 힘든 여러 가지 요인으로 GUI가 먹통이 되는 현상

(2) 해결 방법

* 기존 Thread 활용 및 setfont 지속적인 변경과 time.sleep 궁합 문제 등의 여러 요인이 원인이 된다고 판단하여 GUI와 궁합이 좋은 QThread 사용, setfont 한번만 선언, time.sleep 대신 Pyqt5의 sleep 기능을 활용, Global 변수 삭제 등 적용
* swap 메모리 100MB => 2048로 확장
* 결국 BUS 에러는 확인하지 못함

- 미해결 문제점 및 개선 방안

a. 문제점

1) socket 통신 재연결 시 검출 코드 미실행

2) 부팅 순서의 복잡도

3) raspberry의 다중 기기에 대한 제어

4) 검출 대상 추가 시 모델 재학습 필요 (처음부터 학습)

b. 개선방안

1) Detecting-computer의 코드 재배열 (검출 코드가 실행되지 않는 현상)

- 해결을 위해 소켓 통신 연결이 끊기는 오류 파악 필요

- 해당 오류에 if문 추가 시 개선될 것으로 보임

2) 부팅 순서의 복잡도

- Detecting-computer의 코드를 Cloud-server에 이전시켜 Detecting-computer의 존재 생략

3) raspberry의 다중 기기에 대한 제어(Threading 활용)

- 다중 실행을 이용한 코드 구조 재정리

- raspberry의 기기당 다중 실행으로 제어가 가능할 것으로 판단됨

4) 검출 대상 추가 시 모델 재학습

- 현재의 모델 학습 방식으로는 구조적 한계가 있음

- 파이썬 face\_recognition의 패키지를 활용하여 학습 및 검출이 필요할 것으로 판단됨

## **프로젝트 성능평가 결과**

- 관련 하드웨어 사양

|  |  |
| --- | --- |
| **역할에 따른 HW** | **상세 내역** |
| 1. Raspberry | Soc: Broadcom BCM2711 SoC, CPU: ARM 1.5GHz, RAM: 4GB |
| 2. Detecting-Com | CPU: i5-8250U 1.6GHz(쿼드코어), GPU: GTX1050, RAM: 8GB |
| 3. 네트워크 기기 | 모델명: iptime A604R, 5GHz: 867Mbps, DRAM: 64MB, 안테나: 5dBi |

- 평가 결과

|  |  |  |
| --- | --- | --- |
| **평가 항목** | **평가 방법 [총 20번 측정]** | **결과** |
| 1. 얼굴인식 정확도 | 대시보드 테이블에 저장된 Data 분석  (정답 횟수/측정 횟수)\*100 | 70(%) |
| 2. 인식 시간 | 얼굴 검출 안내문부터 통과 안내문 출력까지의 시간  (안내 시간/측정 횟수) | 8.267(sec) |
| 3. 측정 온도 편차 | 체온계 측정값 - 보정값을 적용한 측정 값의 평균 편차  (실제값-측정값)/측정 횟수 | +0.30/-0.68(℃) |

## 

## **6.5 프로젝트 후기**

- 잘했다고 생각하는 부분

1) 아이디어 제시 후 소프트웨어 구성 방법과 구현 기술을 구체적으로 계획하여 프로젝트 진행 중의 혼란이 최소화되었다.

2) 문제점이 발생되었을 때 적절한 대체품을 파악하였다.

3) 팀원들이 각각 맡은 역할을 최선을 다해 수행하였다.

4) 각 문제 발생 시 서로 해결책을 제시하여 문제를 해결하였다.

5) 정기적인 회의를 통한 서로의 진행 여부 확인 및 의사소통이 원활했다.

6) 정해진 시간 계획에 따라 프로젝트가 진행되었다.

7) Azure를 경험하기 위해 학원에서 결제하기 전 개인 계정으로 Azure를 구축하여 Test를 통한 기존 소프트웨어 구성도의 문제점을 미리 파악하였다.

- 아쉬웠던 부분

1) 상용 모듈만으로 제작한 하드웨어 부분이 아쉽다. (모듈을 직접 제작했으면 좋았을 것 같다)

2) 확장성을 고려하여 함수를 더욱 세분화해서 코딩해야 했다. (다중 제어를 고려했어야 한다)

3) 제품화를 고려하여 하드웨어 및 소프트웨어 구성을 설계하지 않은 것이 아쉽다. (비용 문제)

4) 프로젝트 주제 특성상 시각화에 필요한 자료의 다양화가 부족했다.

5) 3D 프린팅 기계가 멀리 있어 시간이 많이 소모되었다.

6) 전체적인 시스템의 안정성이 빈약한 것이 아쉽다.

7) DB를 더 활용하여 json 환경설정 파일의 항목을 감소시키지 못한 부분이 아쉽다.

# **7. 프로젝트 핵심 코드 및 참고 자료**

## **7.1 Raspberry 관련**

<자료 수집>

파일명: temper.py

### 거리 측정

def distance(self): #거리 측정 함수

    while(True):

        GPIO.output(self.TrigPin, False)

        time.sleep(0.00001)

        GPIO.output(self.TrigPin, True)

        time.sleep(0.00001)

        GPIO.output(self.TrigPin, False)

        time\_first = time.time()

#트리거를 on/off하며 그 사이의 텀을 이용해 거리를 측정하는 원리

        while GPIO.input(self.EchoPin) == 0 :

            start\_time = time.time()

            if start\_time-time\_first >= 10.0: break

        if start\_time-time\_first >= 10.0: continue

#트러블이 생겻던 가까워 질 때 신호가 꼬이며 start\_time loof에 나오지 못하는 현상

#(loof에 들어가기전의 시간을 측정하여 일정 시간 이상이 되면 빠져나오고 외부 loof문으로 돌림)

        while GPIO.input(self.EchoPin) == 1: end\_time = time.time()

        duration = end\_time - start\_time

        distanceCm = duration \* 17000

        distanceCm = round(distanceCm, 2)

        if distanceCm <= 150.0:

            return distanceCm

        else: continue

#측정 거리가 너무 길면 온도 보정이 맞지 않아 150cm 이하로 반환 값이 나오게 함

### 온도 측정 및 거리 보정값 적용

def get\_temp(self): #온도 측정 함수(보정값 적용)

    while True:

        self.mlx.getFrame(self.frame)

        if self.frame[0] != 'nan': #온도 측정시 맨 처음 nan값을 반환하여 방지

            max\_temp = max(self.frame)

            result = [num for num in self.frame if num <= 36.0] #온도 측정시 이상한 값 필터

            result.sort(reverse=True)

            mean\_temp = sum(result[0:3])/3.0 #높은 온도 3개의 평균냄

            distanceCm = self.distance()

            compen\_temp = mean\_temp + (1.2 + distanceCm / 25) #온도에 거리에 따른 보정값 적용

            return compen\_temp

<인터페이스>

파일명: ras\_run.py

### GUI QThread 및 화면 구성 및 소켓 통신

class Thread1(QThread): #GUI에 영상을 띄워주기 위한 QThread

    def \_\_init\_\_(self, parent, label\_1, mjpg\_PATH):

        super().\_\_init\_\_(parent)

        self.label\_1 = label\_1

        self.mjpg\_PATH = mjpg\_PATH

    def run(self):

        self.show\_video()

    def show\_video(self):

        cap = cv2.VideoCapture(self.mjpg\_PATH)

        self.label\_1.resize(width, height)

        while True:

            ret, img = cap.read()

            if ret:

                img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

                h,w,c = img.shape

                qImg = QtGui.QImage(img.data, w, h, w\*c, QtGui.QImage.Format\_RGB888)

                pixmap = QtGui.QPixmap.fromImage(qImg)

                resizeImage = pixmap.scaled(600, 480, QtCore.Qt.KeepAspectRatio)

                QApplication.processEvents()

                self.label\_1.setPixmap(pixmap)

            QtTest.QTest.qWait(5)

        cap.release()

class Thread2(QThread): #GUI에 안내 글자를 띄우기 위한 QThread

    def \_\_init\_\_(self, parent, cli, label\_2, temp\_move):

        super().\_\_init\_\_(parent)

        self.cli = cli

        self.label\_2 = label\_2

        self.temp\_move = temp\_move

    def run(self):

        self.check\_temp()

    def check\_temp(self):

        self.label\_2.setText("대기중입니다(진행이 안될경우 고개를 움직이세요)")

        self.label\_2.setFont(QtGui.QFont("궁서",38))

        self.label\_2.setStyleSheet("color: black;"

                         "border-style: solid;"

                         "border-width: 5px;"

                         "border-color: #7FFFD4;"

                         "border-radius: 5px")

        while True:

            QApplication.processEvents()

            sign = self.cli.recmessage() #Detecting의 보낸 값을 통해 작동(소켓)

            if '0' in sign:

                self.label\_2.setText("체온을 측정합니다(진행이 안될경우 가까이 와주세요)")

                self.label\_2.setStyleSheet("color: green;"

                             "border-color: #7FFFD4;")

...

                self.time\_term()

                self.label\_2.setText("대기중입니다(진행이 안될경우 고개를 움직이세요)")

            QtTest.QTest.qWait(300)

def start(qwid, label\_1, label\_2, cv2\_PATH, client, temp\_mo):

#QThread를 한번에 실행시키는 함수

    th1 = Thread1(qwid, label\_1, cv2\_PATH)

    th2 = Thread2(qwid, client, label\_2, temp\_mo)

    th1.start()

    th2.start()

    print("started..")

app = QtWidgets.QApplication([])

win = QtWidgets.QWidget()

vbox = QtWidgets.QVBoxLayout()

label1 = QtWidgets.QLabel()

label2 = QtWidgets.QLabel()

vbox.addWidget(label1)

vbox.addWidget(label2)

win.setLayout(vbox)

start(win, label1, label2, mjpg\_PATH, cli, temp\_move)

win.setWindowTitle('출입 안내 시스템')

win.showMaximized()

win.show()

sys.exit(app.exec\_())

#Pyqt5의 화면 구성에 대한 코드

## **7.2 Detecting-Computer 관련**

<딥러닝 관련>

파일명: face\_collection.py

### 학습 데이터 수집

def capture(webcam):

    state = 2#0: non\_mask, 1: white, 2: black

    captured\_num = state

    person = 3 #0: 규범, 1: 형수, 2: 하나, 3: 은용

    while webcam.isOpened():

        li = ['./01039978450\_Choi/01039978450\_'+str(captured\_num)+'.jpg',

            './01080082021\_An/01080082021\_'+str(captured\_num)+'.jpg',

            './01085988951\_Jo/01085988951\_'+str(captured\_num)+'.jpg',

            './01097805386\_Sim/01097805386\_'+str(captured\_num)+'.jpg']

        # read frame from webcam

        status, frame = webcam.read()

        if not status:

            break

        # cvlib패키지를 활용 사람의 얼굴 영역 검출

        face, confidence = cv.detect\_face(frame)

        print(confidence)

        if confidence:

            if confidence[0] > 0.9:

                for idx, f in enumerate(face):

                    (startX, startY) = f[0], f[1]

                    (endX, endY) = f[2], f[3]

                    face\_in\_img = frame[startY:endY, startX:endX, :]

                    cv2.imwrite(li[person], face\_in\_img)

                    captured\_num = captured\_num + 3

                    print('================================')

                    print(captured\_num)

        if cv2.waitKey(33) == 27: exit()

        if captured\_num >= 900: exit()

        cv2.imshow("captured frames", frame)

def main():

    try:

        webcam = cv2.VideoCapture("http://192.168.0.16:9090/?action=stream")

        if not webcam.isOpened():

            print("Could not open webcam")

        while True:

            capture(webcam)

            webcam.release()

            cv2.destroyAllWindows()

    except cv2.error:

        webcam.release()

        main()

if \_\_name\_\_ == "\_\_main\_\_":  main()

파일명: create\_model.py

### 학습 시키기

#학습 모델을 가저오고 커스터마이징하는 함수

def create\_model(model\_type='xception', in\_shape=(224, 224, 3), n\_classes=4):

    input\_tensor = Input(shape=in\_shape)

    if model\_type == 'resnet50v2':

        base\_model = tf.keras.applications.ResNet50V2(include\_top=False, weights='imagenet', input\_tensor=input\_tensor)

    elif model\_type == 'xception':

        base\_model = tf.keras.applications.Xception(include\_top=False, weights='imagenet', input\_tensor=input\_tensor)

    x = base\_model.output

    x = GlobalAveragePooling2D()(x)

    x = Dense(1024, activation='relu')(x)

    x = Dropout(0.3)(x)

    preds = Dense(units=n\_classes, activation='softmax')(x)

    model = Model(inputs=input\_tensor, outputs=preds)

    with open("model\_layer.txt", "w") as fh:

        model.summary(print\_fn=lambda x: fh.write(x + '\n'))

#학습 레이어를 텍스트 파일로 저장

    return model

N\_EPOCHS = 200

BATCH\_SIZE = 8

IMAGE\_SIZE = 224

fig, loss\_ax = plt.subplots()

loss\_ax.plot(history.history['loss'], 'y', label='train loss')

loss\_ax.plot(history.history['val\_loss'], 'r', label='val loss')

loss\_ax.set\_xlabel('epoch')

loss\_ax.set\_ylabel('loss')

loss\_ax.legend(loc='upper right')

plt.show() #학습 과정을 그래프로 표기함

|  |
| --- |
| 학습 과정 그래프 |
| C:\Users\IOT\Desktop\electronic_list_project\Detect_code\model\xception\xception.png |

파일명: integ.py

### 다중 검증을 통한 확률 올리기

def detecting(self):

    roof\_cnt = 0

    detect\_num = 0

    detect\_cnt = [0]\*(len(self.person\_num)-1) #검출 개수를 담을 리스트

    detect\_acc = [0]\*(len(self.person\_num)-1) #확률을 담을 리스트

    while True:

...

        if roof\_cnt < 5: #roof\_cnt가 5번 돌면 다음으로 넘어감

            print(result)

            roof\_cnt += 1

            if np.max(result) > 0.9: #검출 확률이 0.9이상일때

                detect\_cnt[np.argmax(result)] += 1 #검출되는 인덱스 갯수 세기

                detect\_acc[np.argmax(result)] += np.max(result) #검출되는 확률 합 저장

            continue #roof\_cnt가 5가 될때까지 반복

<카카오 API 관련>

파일명: kakao\_send.py

### 카카오 메시지 전송 관련

def update\_tokens(self):

    now = time.time()#만료기간 판별을 위해 현재시간을 구함

    tokens\_make = os.path.getmtime(self.KAKAO\_TOKEN)#토큰의 생성 시간을 구함

    tokens = self.load\_tokens(self.KAKAO\_TOKEN)#kakao\_access토큰을 불러옴

    if 'error' in list(tokens.keys()) or now-tokens\_make > tokens['expires\_in']:

    #access토큰을 다시 받아야 되는 경우를 표현

        print('token을 새로 받습니다')

        kakao\_ori = self.load\_tokens(self.KAKAO\_Ori)

        url = "https://kauth.kakao.com/oauth/token"

        #data 꾸리기

        data = {

            "grant\_type" : "refresh\_token",

            "client\_id" : self.KAKAO\_APP\_KEY,

            "refresh\_token" : kakao\_ori['refresh\_token']

        }

        #access키 받기

        response = requests.post(url, data=data)

        tokens = response.json()

        if 'error' in list(tokens.keys()):

            print('OriginKey 만료되었습니다. 교체 필요!')

            return 0

    if 'refresh\_token' in tokens:   self.save\_tokens(self.KAKAO\_Ori, tokens)

#original 토큰이 갱신된 경우 original토큰 교체

    else:   self.save\_tokens(self.KAKAO\_TOKEN, tokens)

    #갱신 토큰 저장

    return tokens

<DB 연동>

파일명: mysql\_cli.py

### DB에 사진 자료 올리기

def mysql\_add(db\_info, temp, person\_num, accuracy, frame):

    db = pymysql.connect(host = db\_info[0], port=db\_info[1], user=db\_info[2], passwd=db\_info[3], db=db\_info[4], charset=db\_info[5])

    cursor = db.cursor(pymysql.cursors.DictCursor)

    if person\_num != '-':

        sql = "INSERT INTO electronic\_list (Date, Time, phone\_num, person\_name, addr, sex, temperature, accuracy, img) select now(), now(), phone\_num, person\_name, addr, sex, %s, %s, %s from person\_info where phone\_num=%s"

        val = (temp, accuracy, frame, person\_num)

    else:

        sql = "INSERT INTO electronic\_list (Date, Time, phone\_num, person\_name, addr, sex, temperature, accuracy, img) values (now(), now(), '-', '-', '-', '-', %s, %s, %s)"

        val = (temp, accuracy, frame)

    cursor.execute(sql, val)

    db.commit()

    db.close()

# person\_info 테이블의 정보를 조회, 전자명부 테이블에 내용 추가시킴

## **7.3 Cloud-Server 관련**

<대시보드>

파일명: index.php

### 차트 구현

<!-- chart -->

<!-- Grafana에서 차트 생성 후 iframe태그로 웹 페이지에 보여줌 -->

<div class="bs-docs-section" style="margin-top: 30px">

<div class="row">

<!-- 시간대별 체온 (시계열 꺾은선) -->

<div class="col-12">

<iframe src="http://20.200.184.152:3000/d-solo/g1gFgyw7k/dashboard-img?orgId=1&refresh=5m&theme=light&panelId=2" width="100%" height="300" frameborder="0"></iframe>

</div>

</div>

</div>

...

### jQuery datatable을 이용해 테이블 구현

<!-- Script -->

<script>

$(document).ready(function(){

// DataTable

var dataTable = $('#dbtable').DataTable({ // #dbtable에 datatable 적용

'processing': true,   // 서버와 통신할 때 응답을 받기 전이라는 ui를 띄울 것인지 여부

'serverSide': true,   // 서버와의 통신 여부, ajax로 서버 데이터 처리

'serverMethod': 'post',   // 데이터는 post로 넘겨받음

'order': [[0, 'desc']],   // 인덱스가 0번인 열에 해당하는 데이터 desc(내림차순) 정렬

'searching': true,  // 검색 기능

'scrollX': true,    // 가로 스크롤

'pagingType': 'simple\_numbers',   // 페이지네이션 타입 설정

'ajax': {

'url': 'filter.php',    // filter.php로부터 데이터를 넘겨받음

'data': function(data){

var from\_date = $('#from\_date').val();    // 날짜 검색을 위한 변수 설정

var to\_date = $('#to\_date').val();

data.searchByFromdate = from\_date;    // 위에서 저장한 날짜를 data에 추가

data.searchByTodate = to\_date;

}

},

'columns': [  // tbody에 들어갈 리스트. filter.php $data[]로 넘겨받은 값

{ data: 'No'},

{ data: 'Date'},

{ data: 'Time'},

{ data: 'person\_name'},

{ data: 'phone\_num'},

{ data: 'addr'},

{ data: 'sex'},

{ data: 'temperature'},

{ data: 'accuracy'},

{ data: 'img'}

],

'dom': "<'row'<'col-5'l><'col-7'f>>" + "<'row'<'col-12'i>t<'col-12'p>>",

// l : length, f: filter, i: info, t: table, p: paging 순서 및 자리 배치

});

});

</script>

파일명: filter.php

### datatable과 mysql 연동

<?php

include 'config.php';   // 데이터베이스 연결

## 변수 할당

$draw = $\_POST['draw'];

$row = $\_POST['start'];

$rowperpage = $\_POST['length'];   // 한 페이지에 보여줄 데이터 개수

$columnIndex = $\_POST['order'][0]['column'];

$columnName = $\_POST['columns'][$columnIndex]['data'];

$columnSortOrder = $\_POST['order'][0]['dir'];

$searchValue = mysqli\_real\_escape\_string($con, $\_POST['search']['value']);

// mysqli\_real\_escape\_string: 입력창에 delete나 drop등의 명령어를 입력했을 경우 DB 데이터가 손실될 가능성이 있기 때문에 escape상태로 만들어주는 함수

## Date search value --> post로 넘겨받은 Date를 변수에 저장

$searchByFromdate = mysqli\_real\_escape\_string($con, $\_POST['searchByFromdate']);

$searchByTodate = mysqli\_real\_escape\_string($con, $\_POST['searchByTodate']);

## Search --> 키워드로 검색을 하는 sql문

$searchQuery = "";

if($searchValue != ''){

$searchQuery = " and (Date like '%".$searchValue."%' or person\_name like '%".$searchValue."%' or phone\_num like'%".$searchValue."%' or state like '%".$searchValue."%' ) ";

}

## Date filter --> 날짜 검색을 하는 sql문

if($searchByFromdate != '' && $searchByTodate != ''){

$searchQuery .= " and (Date between '".$searchByFromdate."' and '".$searchByTodate."') ";

}

## Total number of records without filtering --> 검색기능을 사용하지 않을 때는 DB에 저장된 모든 데이터의 행 개수를 보여줌

$sel = mysqli\_query($con,"select count(\*) as allcount from electronic\_list");

$records = mysqli\_fetch\_assoc($sel);

$totalRecords = $records['allcount'];

## Total number of records with filtering --> 검색기능을 사용할 때는 검색 조건에 맞는 데이터의 행 개수를 보여줌

$sel = mysqli\_query($con,"select count(\*) as allcount from electronic\_list WHERE 1 ".$searchQuery);

$records = mysqli\_fetch\_assoc($sel);

$totalRecordwithFilter = $records['allcount'];

## Fetch records --> 최종 sql실행문

$tbQuery = "select \* from electronic\_list WHERE 1 ".$searchQuery." order by ".$columnName." ".$columnSortOrder." limit ".$row.",".$rowperpage;

$tbRecords = mysqli\_query($con, $tbQuery);

$data = array();

while ($row = mysqli\_fetch\_assoc($tbRecords)) { // 쿼리 실행 결과를 $row변수에 저장

  $data[] = array(

    "No" => $row['No'],     // key값=>value값 (value값을 key값에 저장)

    "Date" => $row['Date'], // key: [index.php] 파일 - <script> - dataTable - column - data에서 사용됨

    "Time" => $row['Time'],

    "person\_name" => $row['person\_name'],

    "phone\_num" => $row['phone\_num'],

    "addr" => $row['addr'],

    "sex" => $row['sex'],

    "temperature" => $row['temperature'],

    "accuracy" => $row['accuracy'],

    "img" => '<img src="data:image/jpeg;base64,'.base64\_encode($row['img']).'" />',

  );

}

## Response --> 결과 전송

$response = array(

  "draw" => intval($draw),

  "iTotalRecords" => $totalRecords,

  "iTotalDisplayRecords" => $totalRecordwithFilter,

  "aaData" => $data

);

echo json\_encode($response);

die;

?>

<mysql 자동화>

### 자동삭제 기능

[프로시저 생성]

DELIMITER $$ #끝나는 특수문자 지정함

CREATE PROCEDURE autoDel()

BEGIN

DELETE FROM electronic\_list WHERE date < DATE\_FORMAT(DATE\_ADD(NOW(), INTERVAL -14 DAY), '%Y-%m-%d');

END $$

DELIMITER ;

[스케줄러 생성]

CREATE EVENT autoDelEvent ON SCHEDULE EVERY '0 0:0:14' DAY\_SECOND STARTS NOW() DO CALL autoDel();

### 자동 번호 매기기 기능

[프로시저 생성]

DELIMITER $$ #끝나는 특수문자 지정함

CREATE PROCEDURE sortNum()

BEGIN

alter table `electronic\_list` auto\_INCREMENT = 1;

set @CNT = 0;

update `electronic\_list` set No = @CNT:=@CNT+1;

END $$

DELIMITER ;

[스케줄러 생성]

CREATE EVENT sortNumEvent ON SCHEDULE EVERY 1 SECOND STARTS NOW() DO CALL sortNum();