

실무중심 산학협력 프로젝트 최종 결과보고서

IoT 기술을 이용한 스마트 주거환경 서비스

(Smart Residential Environment Service Using
IoT Technology)

2020 년 06 월 17 일

과제 수행팀 : 32151648, 박동학
32152057, 방승환
32150781, 김승준
32155068, 홍승기

목차

1. 개발 과제 개요	3
1.1 개발 과제 목표	3
1.2 개발 과제 필요성	3
1.3 개발 제한조건	6
1.4 개발 시스템	6
1.5 개발 과제의 기대효과	8
1.6 참고 문헌	9
2. 과제 수행 내용	10
2.1 개발 환경 및 요구 조건	10
2.2 시스템 블록 다이어그램	11
2.3 알고리즘	14
2.4 구현 방법	24
2.5 실험 결과 및 성능 분석	27
3. 과제 수행 결과 내역	30
3.1 추진 계획 및 실적	30
3.2 참여 인원별 역할 및 수행 소감	31
3.3 과제 결과물	33
3.4 활용 방안	35
4. 후기	36
[부록1]Software Source Code	별첨
[부록2]사용설명서	별첨

1. 과제 개요

(1) 개발 과제 목표

현재 딥 러닝, IoT 기술의 발전과 함께 많은 응용 분야가 개발되어 실생활에서 사용되고 있다. 대표적인 예로 실생활에서 많이 접할 수 있는 스마트 빌딩, 스마트 오피스, 최신행 아파트들은 IoT 장비와 얼굴 인식, 위치 기반 사용자 친화적 환경 제공, 인공지능 비서 등의 서비스를 편리함을 위해서 제공되고 있다. 하지만 이러한 서비스를 제공 받기 위해서는 아파트 건설 단계에서부터 센서들과 기기들이 함께 설치되어야 한다. 새로운 기능이 등장하여 이를 적용하기 위해서는 기존 장비들의 교체가 필요하다. 하지만 비용적 측면에서 단기간에 적용하기는 쉽지 않고 IoT 기술 등장 이전에 지어진 주거 공간의 경우에는 특히 이러한 서비스를 제공하는 것은 큰 공사를 동반하게 된다.

본 프로젝트는 딥 러닝을 지원하는 모듈형 IoT 시스템으로 이러한 문제점을 해결하고자 중앙관리형 모듈과 이를 관리하는 라즈베리파이와 센서를 설치하여 최신행기술에 유연하게 대처할 수 있으며 적은 비용으로 이를 활용할 수 있는 시스템을 개발하고자 한다.

(2) 개발 과제 필요성

－ 시장규모 추정

국내 스마트홈 시장 규모 추이(단위: 억 원, %)¹⁾

구분	2014년	2015년	2016년	2017년	2018년	CAGR
스마트 융합가전	26,260	32,825	42,672	57,608	83,532	33.5%
홈오트메이션	3,200	3,520	4,048	4,655	5,586	14.9%
스마트 홈시큐리티	5,794	6,953	8,691	11,298	15,253	27.4%
스마트 그린홈	969	1,114	1,337	1,672	2,173	22.4%
스마트TV& 홈엔터테인먼트	49,454	59,345	68,247	75,071	82,578	13.7%
전체	85,677	103,757	124,995	150,304	189,122	21.9%

<그림 1 . 국내 스마트홈 시장 규모 추이>



- SK텔레콤의 스마트홈은 조명, 난방, 대기전력차단 등의 세대별 기본 서비스는 물론 각 아파트 단지별 공지사항, 주민투표, 엘리베이터 호출, 관리비 등을 공용 서비스로 제공한다.
- 9월 초 분양 예정인 성남시 최대 아파트 단지인 '산성역 포레스티아'에도 SK텔레콤의 스마트홈 서비스가 기본으로 제공된다. 이 단지에는 SK텔레콤의 스마트홈 기본 기능은 물론, 세대별 거실과 안방, 침실, 부엌 등에 조도 조절이 가능한 LED감성 조명이 기본 제공돼 스마트홈 앱을 통해 영화모드, 절전모드, 수면모드 등으로 조절이 가능하다.

<그림 2. 스마트홈 적용 누적 세대수>

위의 표는 IoT 스마트홈 서비스 시장규모가 점차 증가하고 있다는 것을 보여준다. SK, 두산 등 대기업 건설회사도 물론 IoT전문 기업들도 적극적으로 시장 진입 하고 있다. 미래의 주거환경은 IoT 테크놀로지가 탑재 되어 신 건축물은 물론, 노후화되고 낙후된 시설의 건축물도 스마트 IoT 홈 서비스로 새 단장 할 수 있을 것이다.

● '서신 아이파크' IoT(사물인터넷), 미세먼지 측정 시설 구비



- '최신식 IoT시설은 미세먼지 걱정도 덜 수 있다. 공기질 측정 센서를 통해 세대 내 공기 상황을 실시간 감지·분석해 입주인에게 알려준다. 세대 내 공기질 측정 센서가 천장에 설치돼 관련 정보를 확인할 수 있다.
- 필요한 경우 IoT(사물인터넷)가 연동된 에어컨, 공기청정기 등이 자동으로 작동하게 된다. 특히 초미세먼지가 많은 날처럼 공기 질 상태가 좋지 않을 경우 가로등 미세먼지 상태 표시등 색깔이 변해 미세먼지 정보를 제공한다.

< 그림 3 . IoT 시스템 - 서신 아이파크 >

최근 '서신 아이파크'에서는 첨단 IoT 시스템으로 무장하여 등장했다. 눈 여겨 볼 사항은 각종 센서들을 통해 거주민에게 실시간 실내 상태 정보를 디스플레이 할 수 있게 되었다.



<그림 4 용도별 스마트홈 장치에 대한 소비자 선호도>

현재, 많은 실내 IoT 기능이 등장하고 있다. 최신의 기능 중 소비자가 선호하는 기능들을 추려 선호도를 보여주는 그림이다.

우리나라처럼 제한된 토지에 높은 인구 밀도를 가지는 나라의 특성상 많은 인구를 수용하기 위해서는 고층 건물, 공동 주거 형태가 필요하다. 최근 많은 기술의 발전으로 건물에 여러 기능들을 설치한 스마트 빌딩에 대한 연구 및 활용이 활발하게 이루어지고 있다. 스마트 빌딩은 기존의 건물들에 비해서 사람의 노동력은 줄이고 정확도는 높일 수 있는 위험 감지 시스템, 자동 건물 관리 시스템 등이 도입되어 사람들의 편리성과 시간적 효율성을 극대화하고 있다.

하지만 현재의 시스템은 건물의 건설 단계부터 이를 매립 하거나 후에 많은 비용을 지출하여 건물에 설치 해야 하기 때문에 최근에 지어진 건물이외에는 이러한 시스템을 도입하기에는 무리가 있다고 볼 수 있다. 빠르게 변하는 기술에 대응하여 건물 관리 시스템을 구축하는 것은 인건비를 절감하고 안전성을 높이는 등 필요성이 대두 되고 있지만 현재의 IoT 장비들은 확장성에 있어서 제한 되는 것이 현실이다. 건물에 적용할 수 있는 수 많은 모듈들이 개발되고 시장에 나와 있지만 이를 통합적으로 사용하기에는 제조사마다 호환성이 달라 원하는 기능만을 선택하여 사용하는 것은 쉽지 않다.

본 프로젝트에서 목표로 하는 저 비용 고 호환성의 시스템을 통하여 환기 시스템, 화재 시스템, 주차 시스템 등을 통합적으로 관리하는 것은 수 많은 시설에서 생활하고 있는 현대에서 필요하다고 생각된다.

(3) 개발 제한조건

1. 제한된 개발환경

본 프로젝트에서는 많은 모듈형 기기들로 데이터를 입력받고 중앙에서 처리하여 결과를 알려주는 시스템 구축을 목표로 하고 있다. 하지만 이러한 시스템은 완전히 개발되어 시장에 판매되고 있는 제품에 비해서 크기나 성능 측면에서 최적화되지 못한다는 문제가 있다. 프로젝트를 개발하고 테스트하는 시점에서는 기존의 시스템에 비해서 모듈의 크기나 성능면에서 뒤떨어질 수 있을 것이라고 생각한다. 또한 현재 시장에서 구할 수 있는 Jetson Nano, 라즈베리 파이, 각종 모듈들은 실험, 교육을 목표로 소량 생산하여 유통되고 있어 제한적인 비용으로 모든 서비스를 테스트하고 연동하는데 한계가 있을 것으로 보인다.

2. 보안성 문제

얼굴 인식을 통한 보안 절차를 핵심 기능으로 삼고 있는 만큼 높은 정확도 만큼이나 보안이 중요하다 하지만 현재 보유하고 있는 카메라는 심도를 측정하지 않고 이미지의 픽셀 정보만을 가지고 학습을 진행하기 때문에 사용자의 실제 모습과 사진속의 사용자 모습을 구별하지 못한다. 예를 들어 애플의 안면인식의 경우 수 만개의 지점의 깊이를 측정하여 사진과 실제 사람을 구별하지만 현재 우리 팀에서 개발하고 있는 시스템은 모든 장비를 구비하기에 한계가 있다.

(4) 개발 시스템 specification

IoT, Deep-Learning Kit를 이용한 IoT 시스템 구축을 목표로 하며 프로젝트에서 사용되는 장비는 다음과 같다.

Jetson Nano - Deep learning kit	얼굴 인식, 번호판 인식을 처리하며 가장 많은 데이터를 처리하게 될 중앙 처리 모듈로 높은 하드웨어 성능을 바탕으로 하고 있다.
라즈베리 파이	각 센서들에 연결되어 입력되는 신호들을 전처리하여 Jetson Nano로 전송한다.
카메라 모듈	안면인식, 번호판 인식을 진행하기 위한 이미지 정보를 읽어오게 도와주는 모듈
가스 센서	수행하고자 하는 기능에 필요한 대기중 가스농도 정보를 읽어온다.
LAN카드, 안테나	무선 네트워킹을 위한 부품

본 프로젝트에서 제공하고자 하는 주요 기능은 아래와 같다.

1. 아파트 공동 현관 얼굴 인식

: 카메라 모듈을 통해서 사용자의 이미지를 입력 받아 이를 Jetson Nano로 전송하다. 전송된 사진은 Deep Learning Model에 의해서 반환되는 Label을 통해서 사용자 본인 여부를 판정한다. 이를 통해서 사용자 본인 인증이 완료되면 아파트 공동 현관 출입문을 열면서 이와 동시에 엘리베이터를 불러오고 사용자의 거주하고 있는 층으로 안내한다.

2. 주차장 예약 및 배치 서비스

: 사용자가 건물 주차장에 진입 할 때 카메라, 번호판을 인식하여 사용자 인증을 통해 사용자의 주거 위치를 파악한다. 이를 통해서 현재 주차 가능한 가장 가까운 위치를 알려주고 이를 기록하여 후에 자동차의 위치를 쉽게 파악 할 수 있게 한다.

3. 가스 센서를 통한 스마트 홈 지원

: 라즈베리를 활용하여 센서를 연결, 주거 공간이나 공용 공간에 발생하는 이상을 탐지한다. 가장 대표적인 기능으로는 가스 센서를 이용하여 가스 유출 유무를 판단하여 사용자에게 알려주고 또한 이상 탐지 기능을 탑재하여 시스템 해킹 유무 및 시스템 이상을 사용자에게 제공 한다.

(5) 개발 과제의 기대효과

가. 기존 기술의 현황, 문제점 및 개선 방안

기존 기술 현황 1. 삼성 래미안 아파트에서 이용되는 서비스로 스마트 홈키 등 미리 설치된 기기로 NFC 기술을 이용하여 자신이 주차한 차량의 위치를 알려주는 서비스가 존재한다.

문제점 : 스마트 홈키(Home Key)를 이용하기 때문에 특정 기기인 스마트 홈키를 소유하지 않으면 이용이 제한된다. 또한 키를 분실할 경우 서비스 사용과 사용자 보안에 문제가 발생 된다.

개선 방안 : 주차장 입구에 잭슨 나노를 설치하여 안면인식을 통한 보안 절차를 통해서 입주민에게 효율적인 주차공간을 제공한다.

기존 기술 현황 2. 이마트, 홈플러스 등 대형 마트나 공영 주차장등에서 사용되는 주차 여석 확인 시스템

문제점 : 주차장 입구에서 확인한 여석과 실제로 주차장에 진입하였을 때 남은 여석의 차이가 존재 할 경우가 있음 또한 남은 여석이 있어도 그 주차공간의 정확한 위치를 제공하지 않기 때문에 혼란이 올 수 있다.

개선 방안 : 주차장 입구에서 미리 주차공간의 위치를 확인할뿐더러 미리 주차공간을 예약하여 효율적인 주차가 가능해진다.

기존 기술 현황 3. 아파트 공동현관문 출입 시 비밀번호 입력 또는 NFC카드를 통해 출입

문제점 : 보안성이 취약하고 NFC카드를 분실하였을 경우와 소지하지 않았을 경우 출입이 제한된다.

개선 방안 : 안면인식을 통해 높은 보안성과 NFC카드를 소지하는 번거로움이 해소된다.

나. 과제 개발에 따른 기대효과

- 최근 신축되는 아파트나 건물에 적용되고 있는 스마트 IoT 서비스를 적은 비용으로 제공 받을 수 있다.
- 아파트 주민의 얼굴을 인식하여 별도의 절차 없이 편하게 공용 현관을 통과할 수 있으며 동시에 외부인의 출입을 차단해 보안 기능을 증대 할 수 있다.
- 지속적으로 발생하는 주차공간 부족 문제와 주차를 위해 탐색하는 시간을 줄여 보다 효율적인 주차공간 관리를 할 수 있다.
- 여러 가지 센서를 통해서 부재중에도 집이나 사무공간의 안전을 모니터링하고 조치할 수 있다.

(6) 참고 문헌

1. Practical Python and OpenCV - Adrian Rosebrock
2. Air Quality Monitoring System Based on IoT Using Raspberry Pi - Kumar, Somansh, and Ashish Jasuja. 2017 International Conference on Computing, Communication and Automation (ICCCA). IEEE, 2017.
3. All eyes on you: Distributed Multi-Dimensional IoT microservice anomaly detection - Pahl, Marc-Oliver, and François-Xavier Aubet. 2018 14th International Conference on Network and Service Management (CNSM). IEEE, 2018.
4. OpenCV 얼굴 인식하여 잠금해제 예제-파이썬(Python)
-<https://m.blog.naver.com/PostView.nhn?blogId=chandong83&logNo=221436424539&proxyReferer=https:%2F%2Fwww.google.com%2F>
5. 파이썬 자동차 번호판 인식 -<https://opentutorials.org/module/3811/25288>
6. Raspberry Pi Developer Forums - <https://www.raspberrypi.org/forums>
7. Jetson Nano Developer Forums - <https://forums.developer.nvidia.com>

2. 과제 수행 내용

(1) 개발 환경 및 요구조건

개발 환경

개발 언어 : Python 3.6, Anacaconda Virtual environment

GUI tool : PyQt5, PyQt Designer

운영 체제 : Ubuntu 18.04 LTS(Jetson Nano), Raspbian (raspberry pi 4)

주요 소프트웨어

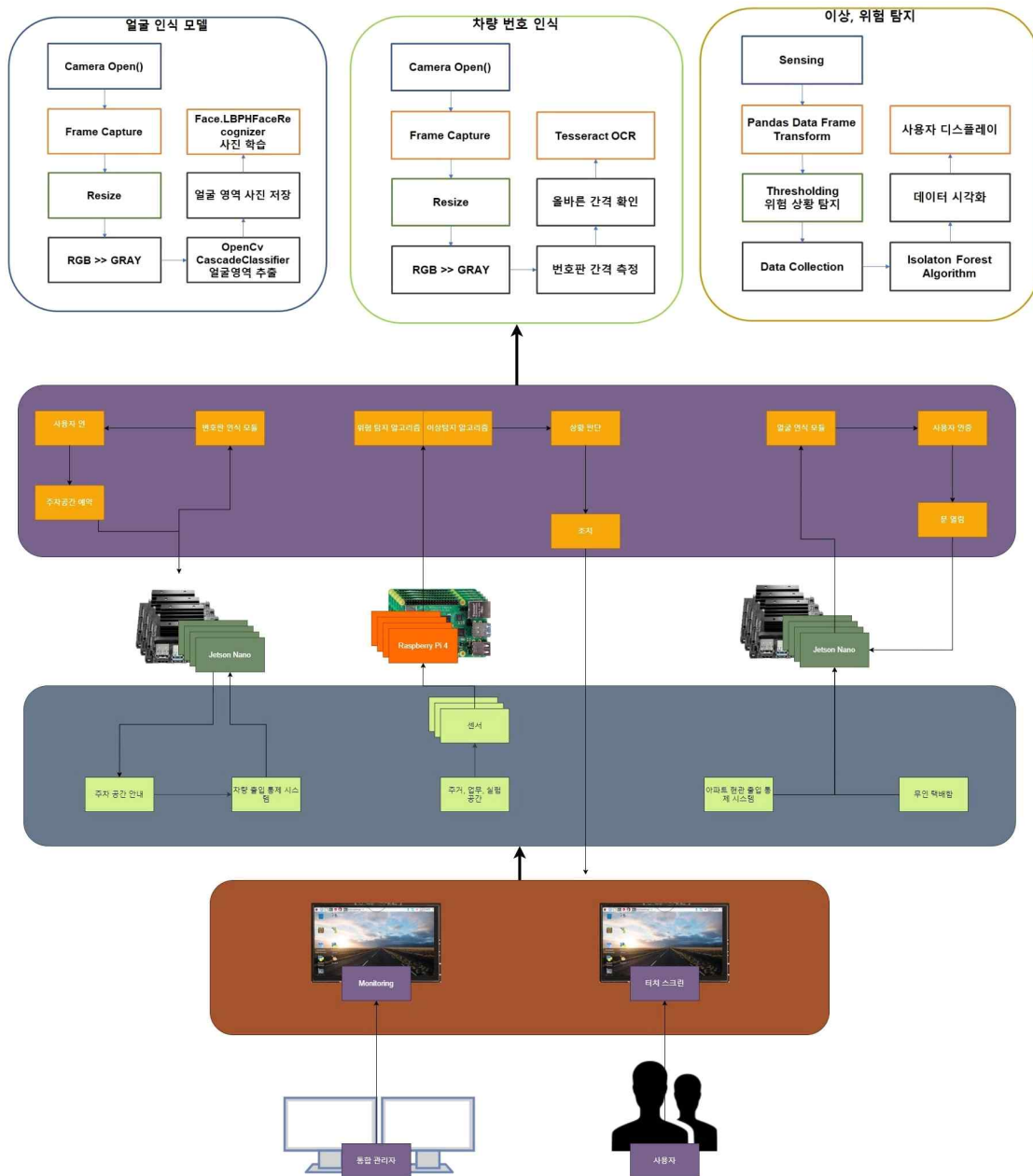
1. OpenCV(Python) - 얼굴인식, 번호판 인식
2. Tesseract - 문자 인식
3. Sciki-learn - isolation Forest, anomaly Detection
4. Numpy - 행렬 계산 및 딥러닝에 필요한 도구 제공
5. Matplotlib - 데이터 시각화
6. pandas - DataFrame, to_Csv 기능 제공

주요 하드웨어

1. Jetson Nano - Deep Learning 및 중앙 데이터 처리용(안면인식, 번호판인식)
2. 터치스크린 디스플레이 - 사용자 인터페이스 및 입력
3. Raspberry Pi - 센서 컨트롤, 데이터 입출력 용
4. 카메라 모듈 - 번호판, 안면인식
5. 가스 센서 - 가스 이상 탐지용(Raspberry Pi에 연동된 가스 센서 비정상 수치 - 가스 유출 및 이상탐지 사용자에게 보고)
6. 무선 LAN 카드, 안테나 - 무선 네트워킹을 위해 사용

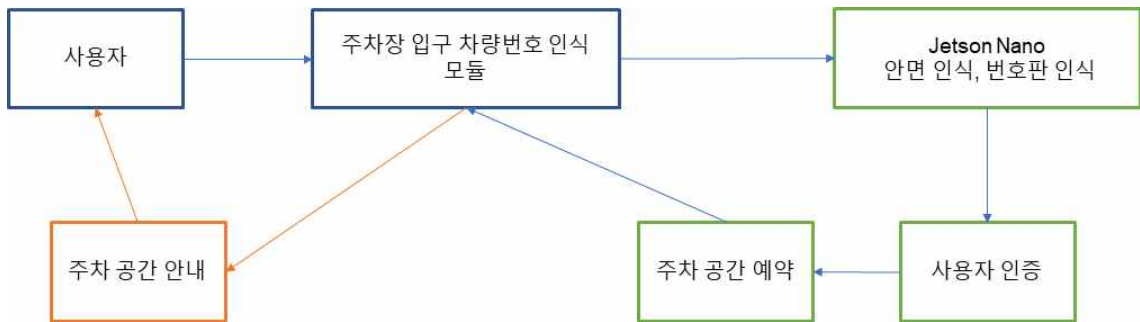
(2) 시스템 블록 다이어그램

전체 시스템 다이어그램



< 그림 5 . 시스템 블록 다이어그램 >

1) 스마트 주차 시스템



< 그림 6 . 차량 인식 모듈 다이어그램 >

- 사용자가 시설의 입구로 차량을 주행한다.
- 주차장 입구에 설치된 카메라를 통해서 사진을 촬영한다.
- 촬영된 사진을 Jetson nano에 탑재된 번호판 인식 모듈을 통해 인식한다.
- 식별된 번호가 사용자 데이터베이스에 저장된 번호인지 구별한다.
- 등록된 사용자라면 거주하는 공간에서 가장 가까운 주차공간을 시스템에서 예약한다.
- 예약된 공간을 사용자에게 안내한다.

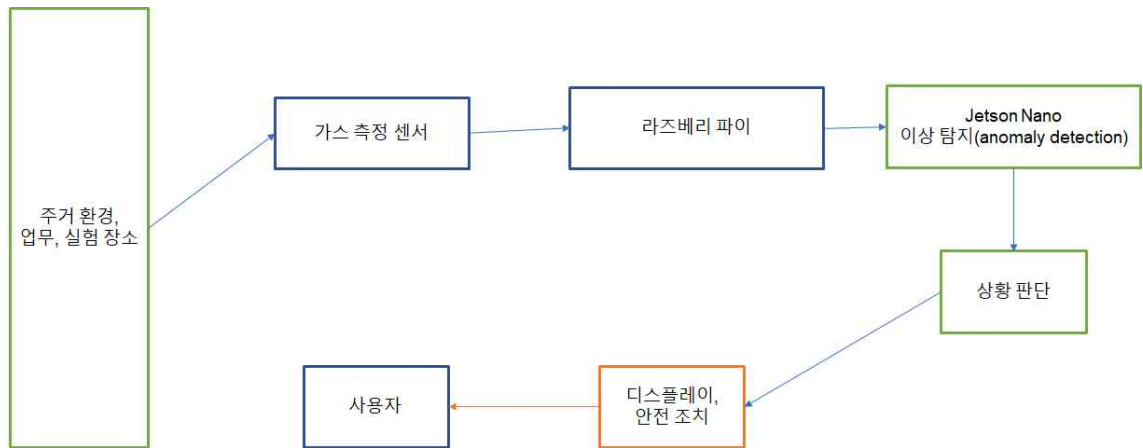
2. 얼굴 인식을 통한 서비스 제공



< 그림 7. 얼굴 인식 서비스 다이어그램 >

- 사용자가 시설의 입구로 진입 시도한다.
- 공동 현관 월패드에 설치된 카메라를 통해서 사진을 촬영한다.
- 촬영된 사진을 Jetson nano에 탑재된 안면 인식 모듈을 통해 인식한다.
- 식별된 번호가 사용자 데이터베이스에 저장된 주민인지 구별한다.
- 등록된 사용자라면 출입문을 개방하고 등록된 거주지에 맞는 층으로 엘리베이터를 호출한다.

3. 안전을 위한 홈 iot 서비스



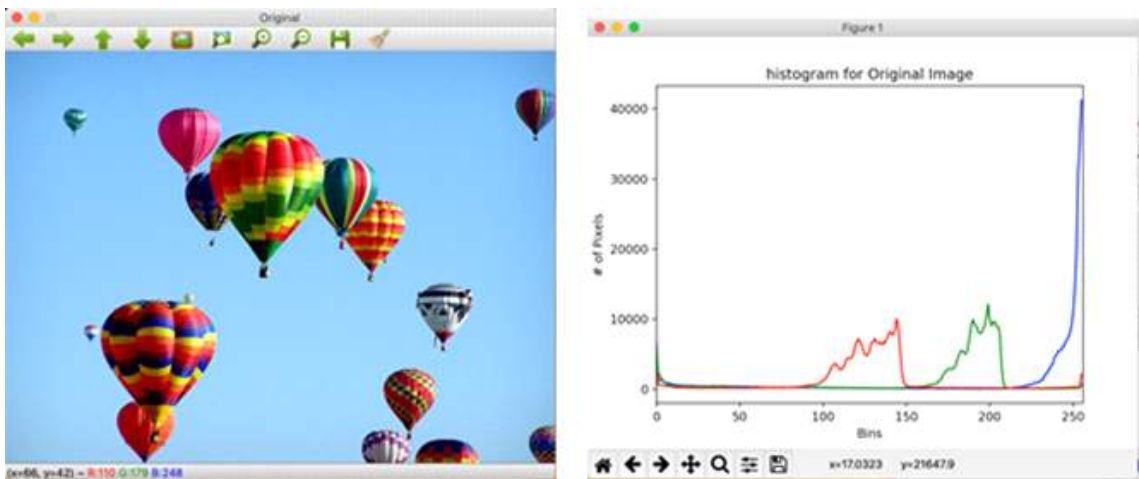
< 그림 8. 이상, 위험 탐지 모듈 다이어그램 >

- a) 센서를 통해서 주기적으로 데이터를 수집한다.
- b) 수집된 데이터는 CSV 파일 형식으로 라즈베리파이에서 저장된다.
- c) 위험 수치를 넘은 데이터는 사용자에게 알림을 준다.
- d) 수집된 데이터를 Jetson Nano로 전송한다.
- e) 전송된 데이터를 Isolation Forest를 통해서 이상탐지를 실시한다.
- f) 이상 데이터로 판단되면 사용자에게 알림을 준다.

(3) 알고리즘

1. 데이터 전처리

Histogram : 어떤한 변수에 대해서 구간별 빈도수를 나타낸 그래프, 즉 각각의 변수에 대해서 어떤 분포를 보여주는 그래프이다. 본 과제에서는 히스토그램을 통해서 색 영역의 분포를 계산할 수 있다. 계산된 분포를 정규화함으로 사진의 윤곽을 다르게 나타낼 수 있다.



<그림 9.Histogram >

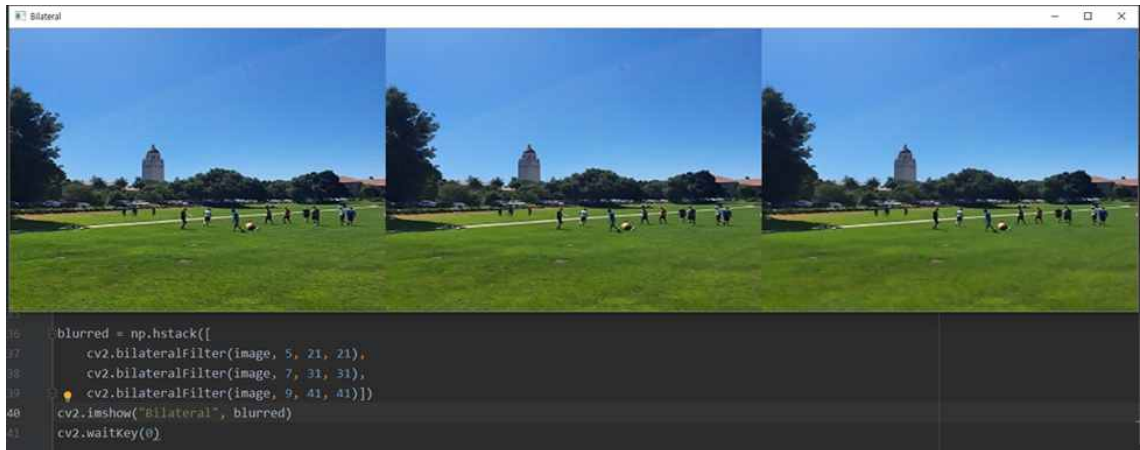
Blur and Smoothing : 사진의 윤곽선의 뭉개짐, 매끄러운 정도를 조절하여 인식률을 높이는 과정으로 Low_pass Filter를 통해 Convolution을 진행함으로 이를 구현할 수 있다.

img_input							
(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)
(0,0)	(0,1)	(0,2)					
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)
(1,0)	(1,1)	(1,2)					
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)
(2,0)	(2,1)	(2,2)					
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)
(5,0)	(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)
(6,0)	(6,1)	(6,2)	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)
(7,0)	(7,1)	(7,2)	(7,3)	(7,4)	(7,5)	(7,6)	(7,7)
(8,0)	(8,1)	(8,2)	(8,3)	(8,4)	(8,5)	(8,6)	(8,7)

< 그림 10. Convolution >

위 그림에서 보이는 노란색 부분은 3X3 필터를 적용하는 것을 시각화 한 것으로 적용되는 필터에 따라 다양한 효과를 구현할 수 있다. OpenCV에서는 다양

한 필터를 제공하고 있으며 본 과제에서 적용한 필터는 가우시안 필터, BilateralFilter 이다.



< 그림 11. Blur and Smoothing >

Edge Detection : 본 과제에서 주요하게 사용한 Edge Detection Algorithm 은 Canny Edge Detection으로 1986년 John F. Canny가 개발한 알고리즘이다. 알고리즘은 다단계로 구성된 알고리즘이다.

1단계 : 노이즈 제거 (Noise Reduction)

– 이미지에서 노이즈는 올바른 경계값을 인식하는데 방해요소가 될 수 있기 때문에 5x5 가우시안 필터를 이용해 이미지의 노이즈를 줄인다.

2단계 : Gradient 값이 높은 부분 탐색

– 경계값은 이미지에서 고주파수를 가지는 것을 의미하며 빠르게 변화한다는 뜻이다. Sobel 커널을 수평, 수직 방향으로 적용함으로 기울기를 구할 수 있다.

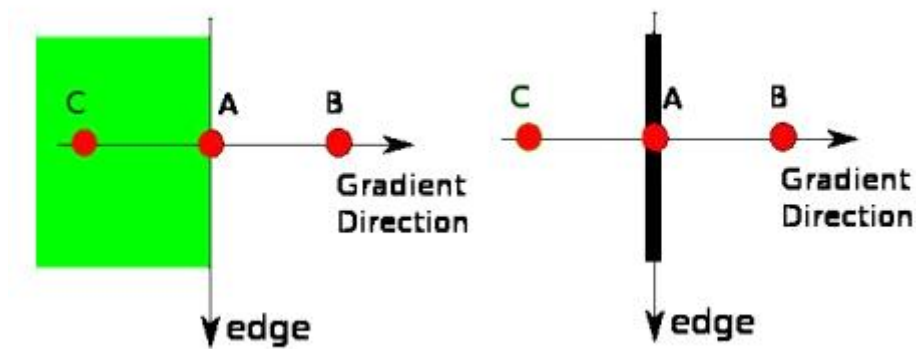
$$Edge_Gradient (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle (\theta) = \tan^{-1} \left(\frac{G_y}{G_x} \right)$$

< 그림 12. Edge Detection >

3단계 : 최대값이 아닌 픽셀의 값을 0으로 변환

– 엣지에 기여하지 않는 픽셀을 제거하기 위해 이미지 전체를 탐색하고 해당되지 않는 부분을 0으로 변환한다.



< 그림 13. Edge Detection 2 >

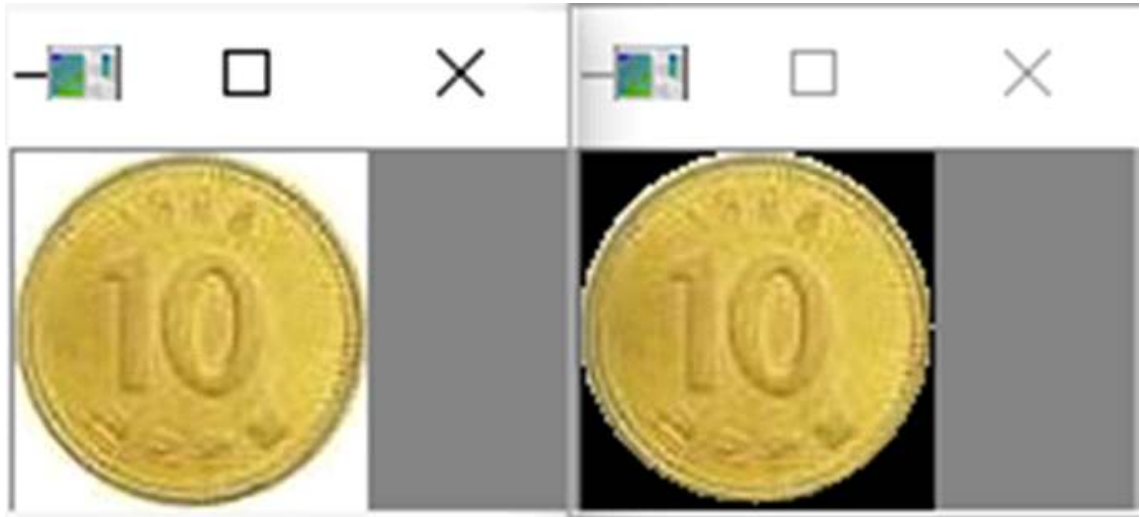
4단계 : Hyteresis Thresholding

- 이전 단계까지 처리된 픽셀에 대해서 Thresholding 기법을 적용하여 엣지를 탐색한다. 이를 조절함으로 확실한 엣지를 탐색, 애매한 엣지까지도 탐색할 것인지를 결정할 수 있다,



< 그림 14. Edge Detection Result >

Find Contours : 이미지 Contour란 같은 값을 가진 픽셀에 대한 연결선이라고 할 수 있다. 이미지에서 동일한 값을 가진 픽셀값들은 주로 경계값을 의미하는 경우가 많기 때문에 물체의 경계를 인식하는데 주로 사용되는 기법이다. 올바른 결과 값을 찾기 위해서는 여러 가지 전처리가 필요한 경우가 많다.



< 그림 15. Find Contours >

2. 번호판 인식 기능에서의 적용 : 위의 알고리즘들은 얼굴인식에서도 유용하게 사용되지만 번호판인식에서 뚜렷하게 효과를 볼 수 있기 때문에 예시로 번호판 인식 모듈을 설명한다.

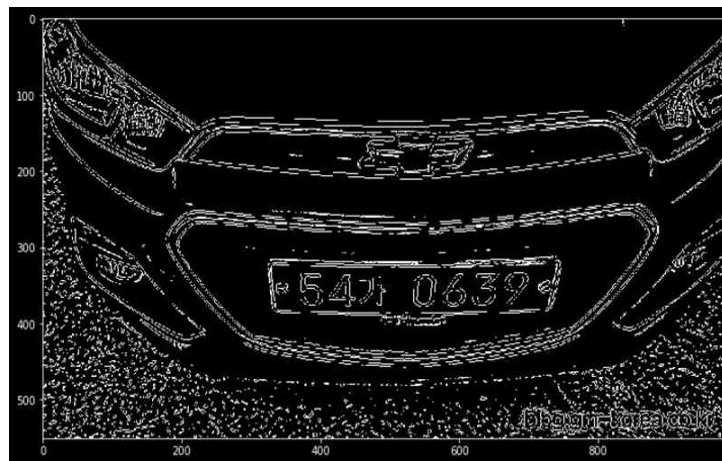
Convert Image to Grayscale and Contrast adjust
: 사진에 대비를 강조해서 사진의 인식률을 올리는 방식



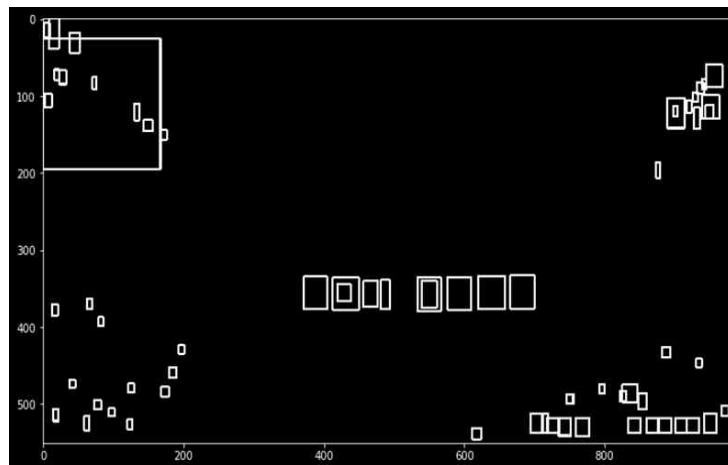
Adaptive Thresholding
: 사진의 노이즈를 제거하여 인식률을 높이는 과정



Find Contours
: 경계값을 찾는 과정



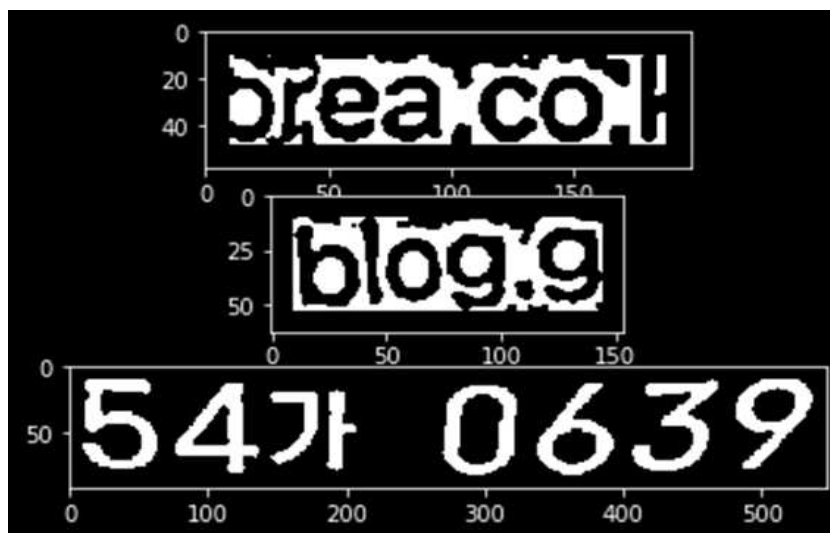
Find Candidates by Char Size
: 글자인 것 같은 대상을 찾는 과정



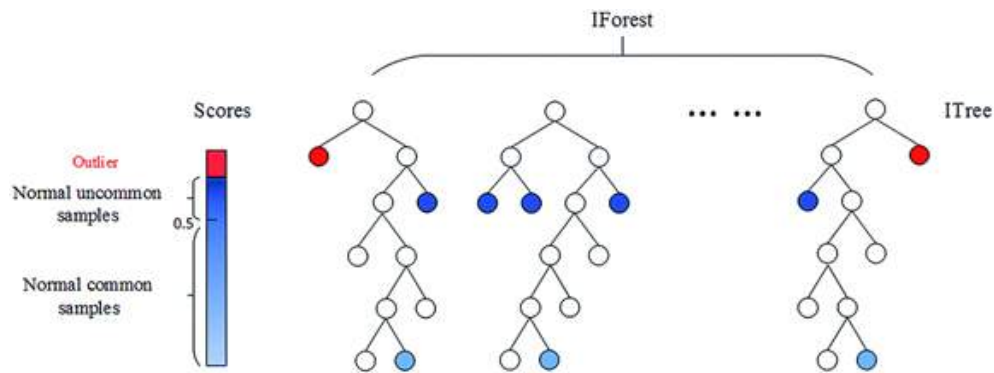
Select Best Candidates
: 자료들중 베스트 대상을 찾는 과정



Select Char Candidate
: 대상을 찾아 인식하는 과정



3. 이상, 위험 탐지



< 그림 22. Isolation Forest example >

이상 탐지(Anomaly Detection) : 이상한 것을 찾는 것라는 의미로 시계열 데이터에서 과거 또는 비슷한 시점의 다른 데이터의 보편적인 패턴에서 벗어나거나 그러한 징후가 보이는 패턴이나 객체를 탐색하는 분야를 의미한다. 본 과제에서는 시계열 데이터에 대해서 해킹이나 장비 오작동을 탐지하고자 한다.

이상 감지(Anomaly Detection) 와 아웃라이어 감지(Outlier Detection) : 이상 탐지는 이상 감지와 아웃라이어 감지로 나눌 수 있다. 아웃라이어 감지는 시간데이터에 관계없이 표현된 데이터에 대해서 보편적인 대상과 차이를 보이는 것을 탐색하는 반면 이상 감지는 시간 또는 순서에 따른 데이터의 패턴을 보편적인 다른 데이터의 패턴과 비교한다.

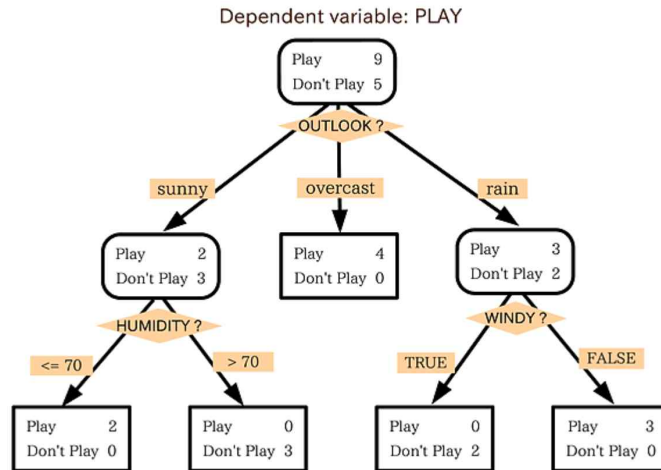
Isolation Forest

: Isolation Forest는 다차원 데이터셋에 대해서 효율적이며 의사결정 트리기반의 이상탐지 기법이기 때문에 결과를 직관적으로 볼 수 있으며 이해도가 높다.

: Random한 칼럼을 선택하고 선택된 칼럼의 최대값과 최소값을 분리하는 값 또한 랜덤으로 선택하여 분류를 진행한다.

Decision Tree(의사 결정 나무)

: 데이터를 분석하여 이들 사이에 존재하는 패턴을 예측 가능한 조합으로 나타내며, 모양이 나무와 비슷하여 의사결정 트리라고 불린다. 각 데이터에 대해서 특징에 해당하는 요소에 대해서 분류를 진행한다.



< 그림 23. Decision tree >

Pseudo Code : Making Decision_tree

Algorithm Build_DT(D)

Input : dataset D

```

1: Tree = {}
2: if D is "Belong to One Class" OR other stopping criteria met then
3:   terminate
4: end if
5: for all attribute a ∈ D do
6:   Predict_DT(Attr)
7: end for
8: atest = Best attribute according to above computed criteria
9: Tree = Create a decision node that tests atest in the root
10: Dv = Induced sub-datasets from D based on atest
11: for all Dv do
12:   Treev = Decision_Tree(D)
13:   Attach Treev to the corresponding branch of Tree
14: end for
15: return Tree
  
```

Algorithm Predict_DT(Attr)

Predict Decision Tree Node's class of x with (a_{test})

1: **for each** test **do**

2: compute predict =

$$Entropy(A) = - \sum_{k=1}^m p_k \log_2(p_k)$$

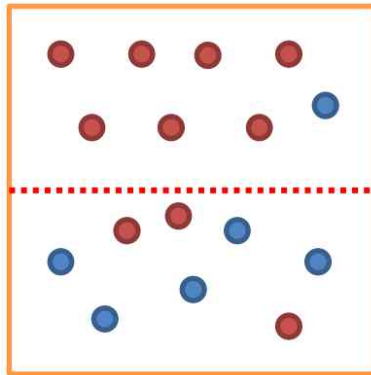
3: best = in best diminishing direction

4: **return** Attr

불순도 / 불확실성

Decision Tree는 구분을 거쳐 각 영역의 순도(homogeneity가 증가, 불순도(impurity) 혹은 불확실성(uncertainty)이 최대한 감소하도록 하는 방향으로 학습을 진행하며 이를 정보획득(information gain)이라고 한다.

예시) 빨간공과 파란공의 구분



1) 엔트로피(Entropy)

– 무질서도를 정량화해서 표현한 값으로 엔트로피가 높을수록 집단의 특징을 찾는 것이 어렵다는 것을 의미

– m개의 레코드가 속하는 A영역에 대한 엔트로피는 아래와 같은 식으로 정의

$$Entropy(A) = - \sum_{k=1}^m p_k \log_2(p_k)$$

– 위의 예시에 대한 엔트로피를 계산해 보면

$$Entropy(A) = -\frac{10}{16} \log_2\left(\frac{10}{16}\right) - \frac{6}{16} \log_2\left(\frac{6}{16}\right) \approx 0.95$$

빨간선을 기준으로 두 개의 집합으로 분할한다고 가정해보면

$$Entropy(A) = \sum_{i=1}^d R_i \left(- \sum_{k=1}^m p_k \log_2(p_k) \right)$$

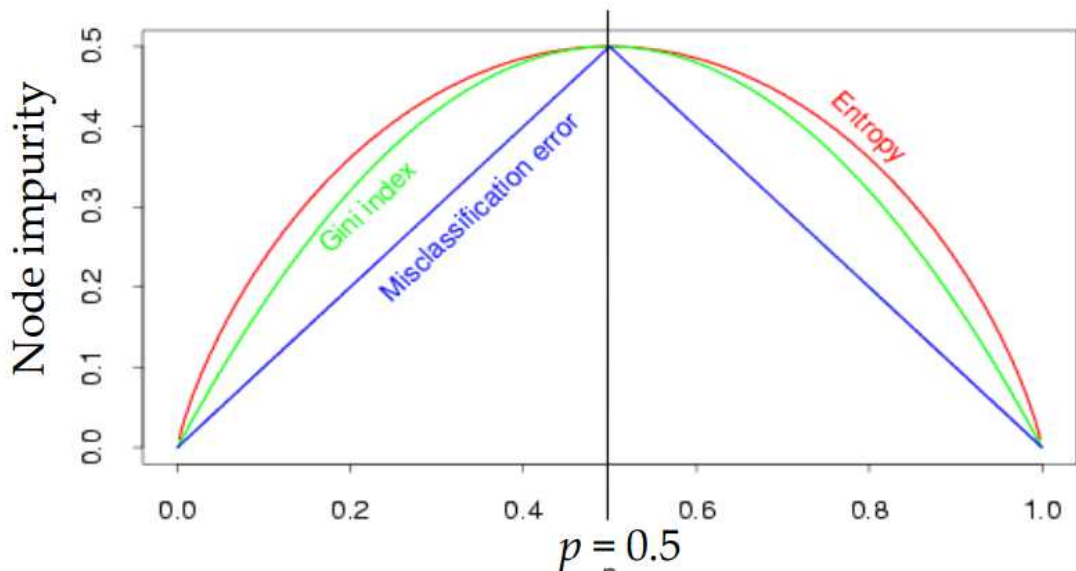
$$Entropy(A) = 0.5 \times \left(-\frac{7}{8} \log_2 \left(\frac{7}{8} \right) - \frac{1}{8} \log_2 \left(\frac{1}{8} \right) \right) + 0.5 \times \left(-\frac{3}{8} \log_2 \left(\frac{3}{8} \right) - \frac{5}{8} \log_2 \left(\frac{5}{8} \right) \right) \approx 0.75$$

다음과 같이 불확실성이 감소하는 방향으로 학습을 진행

2) 지니계수(Gini Index)

- 지니지수를 가장 감소시켜주는 예측변수와 그 때의 최적분리에 의해서 자식노드를 선택
- 지니 불순도 지수는 $0 - (m-1)/m$ 의 사이의 값을 가짐

$$G.I(A) = \sum_{i=1}^d \left(R_i \left(1 - \sum_{k=1}^m p_{ik}^2 \right) \right)$$



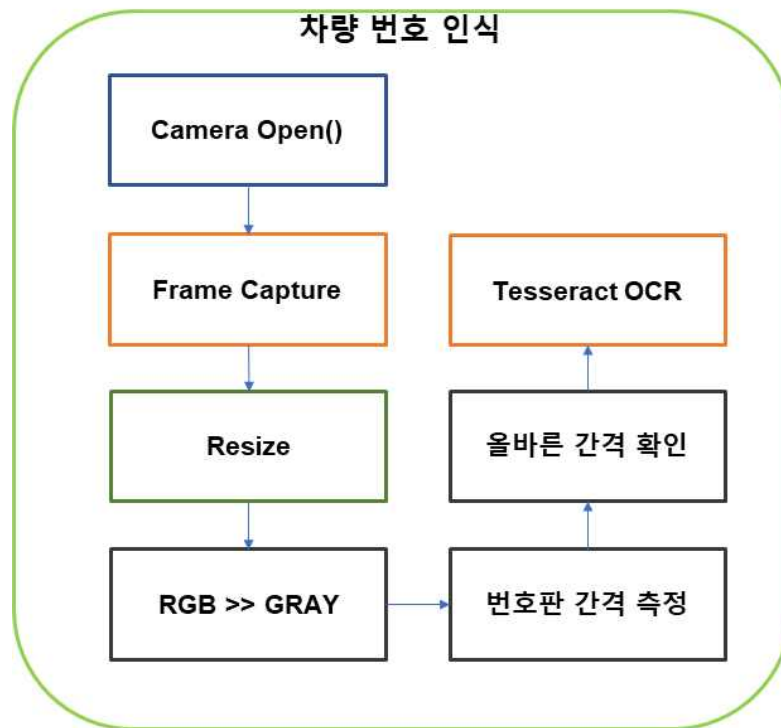
■ scikit-learn에서의 criterion

`criterion { "gini" , "entropy" }, default=" gini"`

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

(4) 구현 방법

1. 스마트 주차 시스템



- a) Camera Open()을 통해서 카메라 모듈을 연다
- b) Camera로부터 Frame을 불러온다
- c) 적절한 Size로 Resize 진행한다.
- d) 3channel RGB에서 GRAY로 색상을 변경한다.
- e) Edge Detection, Find Contour를 통해서 각 객체에대한 사각형을 그리고 간격을 측정한다.
- f) 올바른 간격을 지니는 후보군을 확인한다,
- g) Tesseract OCR을 통해서 글자를 인식한다.

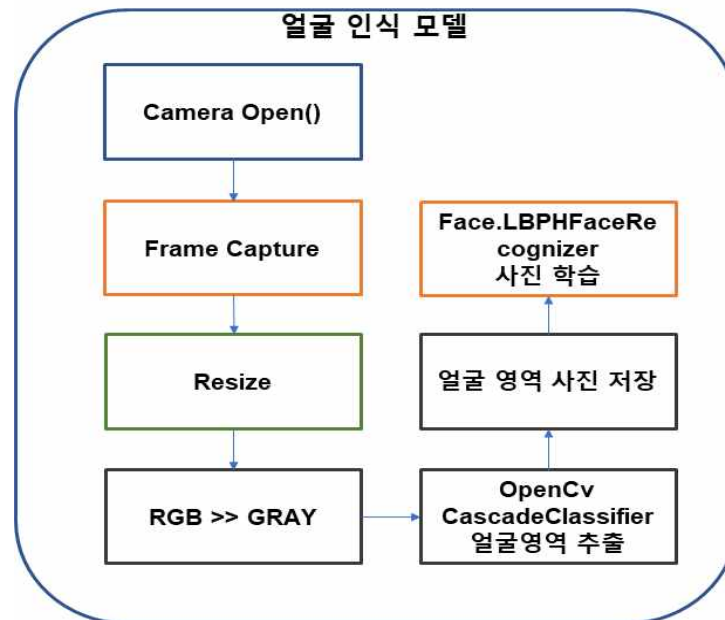
OpenCV, Tesseract OCR 활용 번호판 인식 알고리즘 사용

- 글자 영역 탐색 : OpenCV
- Image Processing : OpenCV
- 번호판 영역의 문자 인식 : Tesseract - OCR

위치 기반 주차공간 안내

- 거주, 이용 공간 기준 최소거리를 이용한 Matching Algorithm 사용

2. 얼굴 인식을 통한 서비스 제공



- a) Camera Open()을 통해서 카메라 모듈을 연다
- b) Camera로부터 Frame을 불러온다
- c) 적절한 Size로 Resize 진행한다.
- d) 3channel RGB에서 GRAY로 색상을 변경한다.
- e) Edge Detection, Find Contour를 통해서 각 객체에대한 사각형을 그리고 간격을 측정한다.
- f) CascadeClassifier를 통해서 얼굴영역의 특징을 가진 후보를 특정한다.
- g) 특징지어진 얼굴 영역을 사진으로 저장한다.
- h) 저장된 사진을 Face.LBPHFaceRecognizer를 통해서 학습한다.

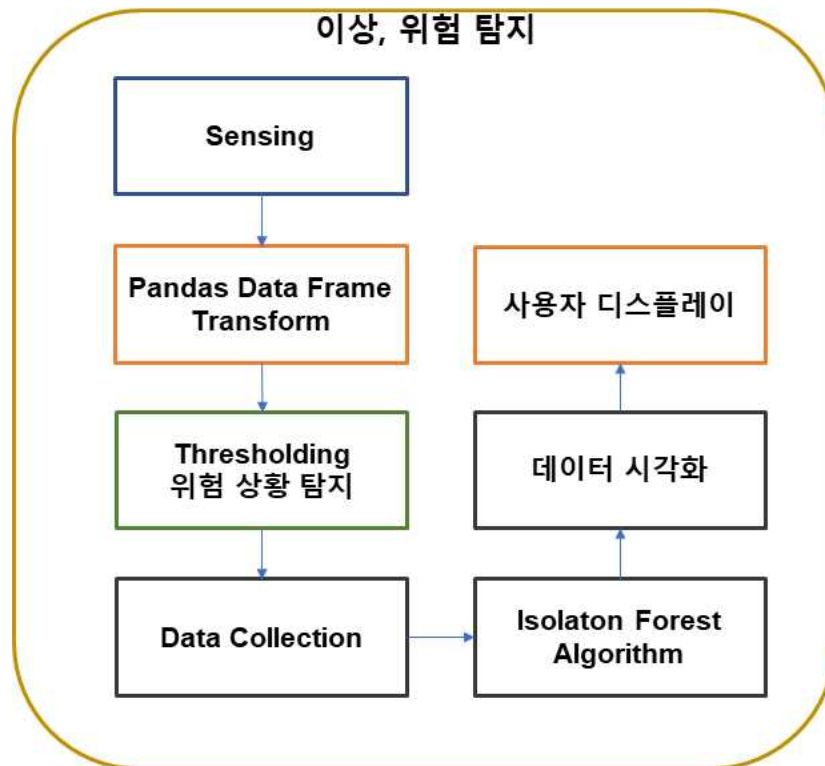
OpenCV를 통한 안면 인식

- 사용자의 얼굴 영역 인식 : OpenCV
- 이미지 학습 진행 : OpenCV
- 안면 인식 - OpenCV, dlib

얼굴 인식 기반의 출입문, 택배 보관함 개폐 기능

- 실질적인 출입문 개폐는 현실적으로 어려워 특정 신호를 주는 것으로 대체 예정

3. 안전을 위한 홈 IoT 서비스



- a) Sensor를 통해서 실제 상황에서의 Data를 Sensing 한다.
- b) Sensing한 데이터를 처리하기 위해서 Pandas Data Frame으로 변환한다.
- c) 1차적으로 변환된 데이터에 대해서 Thresholding 기법을 통해 기준치를 초과한 데이터에 대해서 위험 상황을 탐지한다.
- d) 위험 상황 데이터를 포함한 전체 데이터를 수집한다.
- e) isolation Forest Algorithm을 통해서 이상 데이터를 탐지한다. (지속적인 학습 진행이 필요)
- f) 수집되는 데이터에 대해서 사용자에게 시각화를 제공한다.
- g) 그려진 그래프와 데이터 값, 상황에 대해서 사용자 디스플레이를 통해서 알린다.

Isolation Forest를 활용한 Anomaly Detection

- 센서를 통한 데이터 입력, 수집
- 이상 탐지 알고리즘 수행

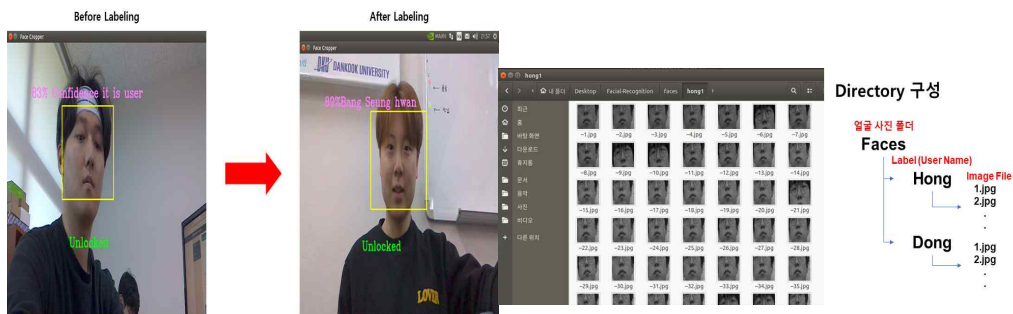
위험 상황 발생 탐지 및 알림

- 위험 상황 발생 시 알림 및 소방서, 경찰서 도움 요청

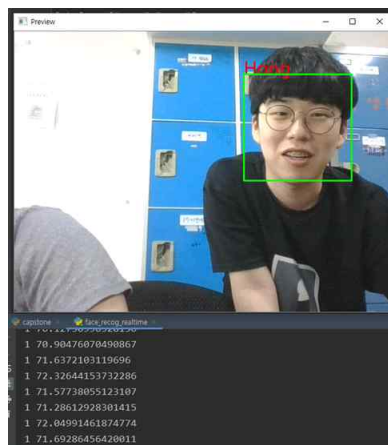
(5) 실험 결과 및 성능 분석

1. 얼굴 인식 기능

얼굴 인식 기능 부분에 대한 초기 정확도가 미미한 수준으로 측정된 원인을 학습 데이터 부족과 불규칙적인 학습 데이터의 위치라고 판단하여 학습 디렉터리를 재구성하여 학습을 진행하고, 라벨링 작업 및 학습 데이터의 양을 증가 시키고 배경에 따른 영향을 고려하여 여러 배경의 사진을 이용하여 정확도를 향상



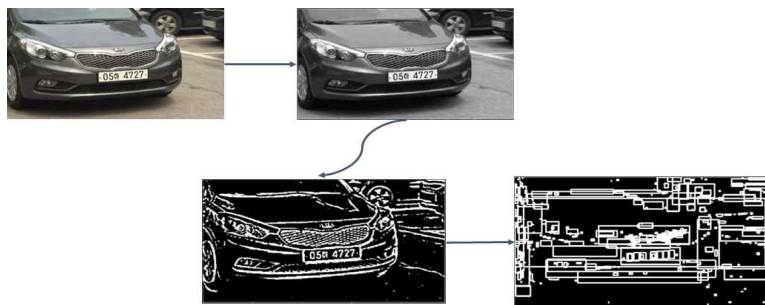
위와 같은 작업을 진행 하고 다시 테스트 해본 결과



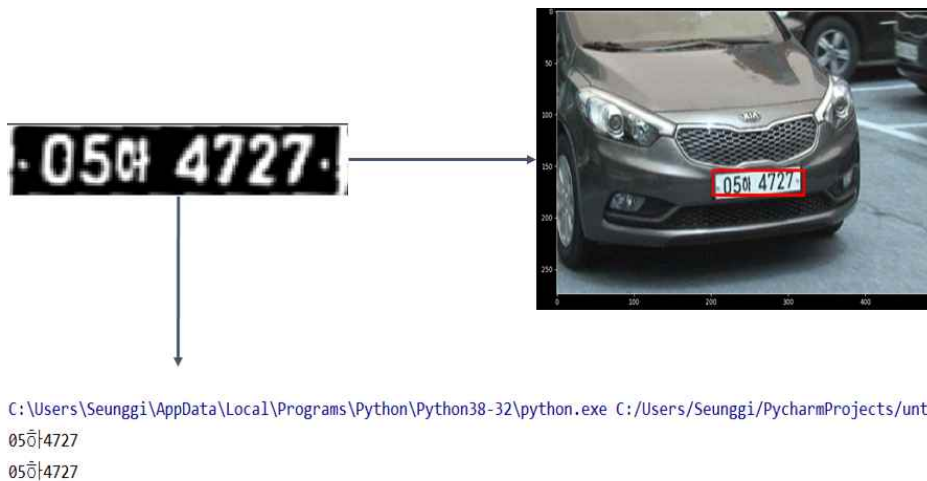
기존 50~60% 정확도를 보였지만 개선 작업 후 60~70%의 정확도를 보여주게 되었다. 비록 바라던 수준의 성능을 보여주지는 못했지만, 개선 작업 후 적정선의 성능을 보여주게 된 것 같다.

2. 차량 번호판 인식 기능

차량 번호판 인식 기능 같은 경우 OpenCV와 TesseractOCR을 활용하여 프로그램 작성하였고,



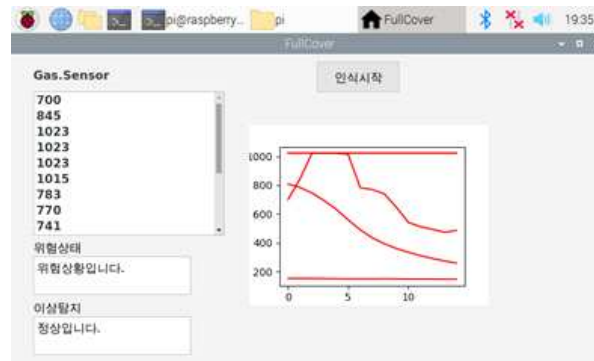
위 그림의 흐름도와 같이 알고리즘이 진행되어 번호판 영역을 추출해 내고, TesseractOCR을 이용하여 번호판의 문자를 추출하는 기능을 수행한다.



프로그램 수행 시 다음과 같이 경우 정상적으로 실행되는 것을 확인 할 수 있었다. 한글의 특성상 제대로 인식이 안되는 몇몇 경우도 나타났지만, 경우의 수가 매우 적어서 만족할 만한 성능을 끌어낸 것 같다.

3. 이상, 위험 탐지 기능

이상, 위험 탐지 기능은 라즈베리파이에 가스센서를 연결하여 데이터를 모으고, 수집된 데이터를 바탕으로 Isolation Forest Algorithm을 사용하여 사용자에게 위험 상황과 이상 상황 알리며, 해당 데이터를 시각화하여 제공해 주는 프로그램을 작성하였다.



프로그램 수행결과는 다음 그림과 같으며 가스센서를 통해 수집된 데이터를 수치적으로 리스트를 통해 보여주며, 수집된 데이터를 기반으로 위험 상태와 이상 상태를 나타내주고, 또한 그래프를 통해서 시각적으로 표현하였다.

처음에 기획했던 가스센서 이외의 다른 센서들을 사용하지 못하여 비교적 단순해졌고, 한 가지 부분에 집중 할 수 있어서 비교적 좋은 성능을 얻어냈다.

3. 과제 수행 결과 내역

(1) 추진계획 및 실적

추진 계획

작업	담당	Start	End	Week														
				1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
준비	자료 조사	팀 전체	1	3														
	아이디어 조율 및 확정		1	2														
	아이디어 수정		1	3														
			2	4														
상세 설계	시나리오 구상	박동학	2	13														
	개발 환경 설정, 필요 모듈 설계	박동학	2	5														
	기능 및 데이터 베이스 설계	홍승기, 방승환	2	6														
	안전인식, 변조판 인식 설계	박동학, 김승준	2	7														
구현			3	15														
	GUI 구현	김승준	3	12														
	Python Code 작성 (모듈화)	방승환																
	입력 및 모듈 구현	박동학	3	13														
	데이터 베이스 구현	홍승기	5	13														
	Project 통합 구현	박동학	13	15														
중간 점검			6	9														
	중간 점검	팀 전체	6	9														
분석			2	14														
	시장 분석	방승환	2	8														
	최적화 분석	홍승기	8	14														
Test			14	15														
	오류 검사 및 수정	김승준	14	15														
	최종 테스트 / 보고서	박동학	14	15														

세부 목표	세부 계획	실 적	달성도(%)	사 유	참여자 성명
자료 조사, 시장 분석	과제의 필요성, 구현 가 능성에 대한 조사와 구현 방법에 대한 자료 수집	1. 참고 문헌 2. 시장 조사 3. 과제 필요성	80%	현재 필요한 시스템인지에 대한 조사와 구현 가능성, 이론 등을 깊이 조사	박동학 홍승기
시나리오 구상	실제 발생 가능한 시나리 오를 구상하고 이를 기록	1. 테스트 케이스 선정 2. 사용자 입장에서 구성 3. 에러 및 버그 예상	95%	가상의 환경에서 다양한 시나리오를 구상하고 테스트 진행	박동학 홍승기 김승준 방승환
시스템 설계	하나의 통합적인 시스템 구성	1. 모니터링 시스템 2. 모듈화 된 프로그램 3. 재사용성 높은 코드	70%	시스템을 유기적으로 구성하는데 한계가 있음	박동학 홍승기 김승준 방승환
GUI 구현	사용자가 편리한 GUI 구 성	1. 사용자 편의를 추구한 GUI 2. 추상화를 통해 작동 원리에 대한 이해 없는 사용 가능성	90%	통합 GUI를 구성 했지만 디자인 요소가 부족함	방승환 김승준
인식 모델 구현	높은 정확도를 보이는 모 델 구축	1. 사용자 개개인 인식 가능 2. 차량 번호판에 대한 한글 인 식 3. 이상 탐지 기능	75%	다양한 모델을 실험하지 못함	박동학
데이터 베이스 설계 및 구축	일관성이 유지되는 데이 터베이스 구축, 사용자 정보 수집	1. 높은 속도를 보이는 접근성 2. 수정 가능함 3. 파이썬을 통한 데이터 접근 가능	75%	직접적인 DataBase를 구축하기에는 절대적인 Data양이 부족	홍승기
최종 테스트 및 문서화	진행한 프로젝트에 대한 체계적인 정리	1. 알고리즘 분석 2. 최종 시스템 분석	80%	다양한 사례조사가 부족	박동학 홍승기 김승준 방승환

(2) 참여 인원별 역할 및 수행 소감

성명	구현부	비고
박동학	딥 러닝 모듈 구축 1. 얼굴 인식 모델 탐색(OpenCV) 2. Anormally Detection 모델 개발 (Isolation Forest)	팀 리더 - 프로젝트 방향성 조율 및 행정 업무 담당
김승준	IoT 장비 설계 및 테스트 1. Jetson Nano, Raspberry, Arduino 장비의 세부적인 스펙 조사 2. 스마트 주차장(공석 예약) 시스템 구현을 위한 센서 구동 환경 구축 및 테스트 3. IoT 홈 서비스 (미세먼지 센서) 시스템 구현을 위한 센서 구동 환경 구현 및 테스트	팀원 - 물품 구입 신청 - 하드웨어 관리 및 보관 - Jetson Nano, Raspberry Pi 연동 - 미세먼지 센서, 초음파 센서 관리 및 연동
방승환	Python GUI 1. GUI 구현시 사용자 편의성을 중심으로 두고, 사용자 친화적인 GUI 환경 구현 센서 동작 알고리즘 구현 1. 번호판 인식을 위한 이미지 프로세싱 개발 (Python, OpenCV) 2. Jetson Nano와 센서부에서 처리할 알고리즘 구현	팀원 - 사용자 편의성 테스트 - 예외 처리 테스트 - 사용자 친화적인 GUI 환경 구성을 위해 기존에 나와있는 GUI 구조 탐색
홍승기	데이터 베이스, 데이터 셋 구축 1. Jetson Nano 와 연결할 적합한 DB환경을 찾아 DB 구축 (CSV-Local) 데이터 시각화 1. 가스센서 등 가상의 입주자에게 제공 될 데이터들을 시각적 자료로 구현	팀원 - 데이터를 가상의 입주자에게 시각적 자료로 보여주는 UI 구성 작업

박동학 : OpenCV를 통해서 이미지를 전처리하고 인식할 수 있는 모델을 탐색, 적용하는데 있어서 평소에는 잘 신경 쓰지 않는 정확도를 고려한다는 점이 낯설었지만 최근 많은 서비스에서 사용하는 머신러닝 알고리즘에 있어서 빠질 수 없는 소용이기 때문에 큰 공부가 된 것 같다.

Anormally Detection 알고리즘은 데이터 중에 비정상적인 데이터를 식별하는 알고리즘으로 특징에 의존해서 실행된다. 우리가 시도한 Isolation Forest의 경우에는 Decision Tree와 유사한 알고리즘을 통해서 다른 데이터들과는 다르게 사전에 빠르게 분류가 완료되는 즉 타 데이터에 비해서 특징이 도드라 지는 것을 탐색한다. 최근 이 알고리즘을 통해서 설비이상 탐지, 어뷰징 방지등 다양한 프로젝트를 기업에서 수행할 만큼 의미 있는 알고리즘을 공부할 수 있었다.

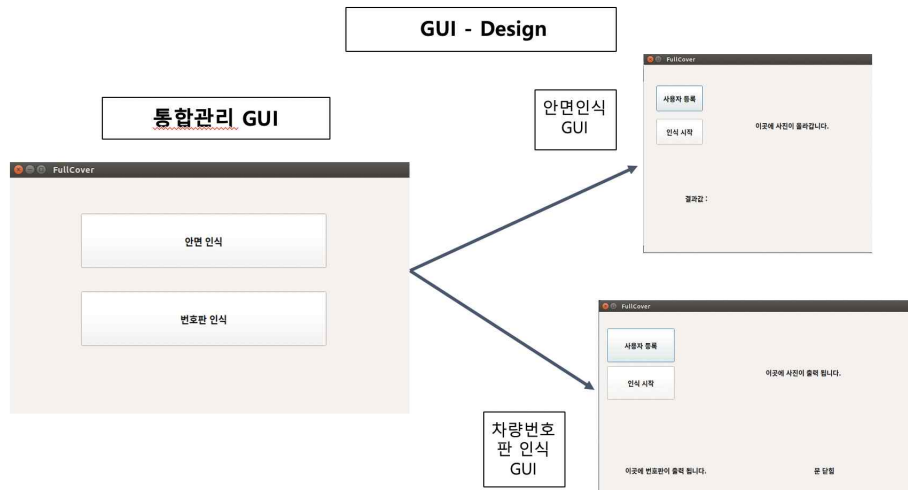
김승준 : Jetson Nano Kit 조립 및 구동을 맡아 낯선 부품들이지만, 철저한 조사를 통해 조립을 성공하였다. 구동 시 부팅이 안되는 문제가 있었는데 LAN카드 문제라는 것을 알고 9번, 10번 회로 절연해 구동 성공시켰다. IoT 홈 서비스중 하나인 가스 센서를 사용 했다. 라즈베리 파이에 가스 센서를 연결해 구동시켜 이상탐지, 위험탐지 기능을 구축했다. GUI 부분은 가스 센서에서 읽어온 정보를 띄우기 위해 QStandardItemModel() 함수를 사용해 띄울 수 있었다. 이번 프로젝트를 하면서 파이썬, OpenCV, 아나콘다, PyQt - QtDesigner 등 여러 프로그램을 사용할 기회가 있었는데 학업에 좀 더 흥미를 붙일 계기가 되어서 좋았다.

방승환 : Python GUI와 번호판인식을 위한 이미지 프로세싱, 센서 부 동작 알고리즘을 구현하며 처음 접하는 부분이 많았지만 Python GUI는 인터넷 검색을 통해 필요한 부분들을 찾아 하나하나 배워가며 구현하는데 많은 도움이 되었습니다. 또한 이미지 프로세싱은 “Practical Python and OpenCV” 책이 도움이 많이 되어 많은 바 역할을 수행할 수 있었습니다.

홍승기 : 이번 프로젝트에서 내가 맡은 역할은 프로그램에서 사용할 적당한 DB를 찾고 DB를 구축하는 것이었다. 여러 DB들을 탐색하고 적당한 DB들을 찾아봤지만 마땅한 DB를 찾지 못하여 프로그램 작성에서 사용하는 Python 언어에서 지원하는 pandas 모듈을 이용하여 수집한 데이터들을 데이터 프레임화 시키고 만들어지는 데이터 프레임을 csv 파일로 저장하여 프로젝트에서 사용하는 하드웨어 상에서 데이터를 수집 및 관리 가능하게 구현을 하였다. 맡은 역할을 수행하면서 새로운 언어(Python)에 대한 스킬도 얻게 되고 여러 경험을 하게 되어 만족스러운 결과를 얻은 것 같다.

(3) 과제 결과물

통합 GUI 디자인



- 같은 장비를 사용하는 안면 인식 기능과 번호판 인식 기능을 선택할 수 있도록 통합적인 GUI 작성
- 사용자가 사용감에 있어서 불편함이 없게 디자인



안면 인식, 사용자 등록 정보를 통해 거주민인 것을 확인후 문을 열어주도록 설계

GUI - Design

실행 모습



텍스트를 성공적으로 인식하여 거주민 확인 후 주차장 문 열림과 동시에 자리
배정하도록 설계

(4) 활용 방안

- 현재 우리나라에서 건설되고 있는 최신 아파트들은 내, 외부 구조뿐만 아니라 실제로 주거하고 생활하는데 더욱 편리한 주거 서비스 또한 아파트 거주자들에게 제공이 된다. 이러한 서비스는 딥 러닝 및 IoT등 여러 가지 기술의 발전이 이루어지고 그 기술을 응용하기 때문에 거주자들에게 제공이 되는 것인데 대표적인 예로 얼굴 인식, 위치 기반 사용자 친화적 환경제공, 인공 지능 비서, 스마트 홈 IoT 기술들이 있다. 하지만 이러한 기술과 서비스는 기술이 발달된 최근부터 제공되고 사용 되어 왔다. 그렇기 때문에 건설된지 오래된 노후 아파트에서는 이러한 서비스를 제공 받기 어려운 것이 현실이다. 이 프로젝트는 이러한 노후 아파트에 서비스를 제공하여 노후 아파트 주민들에게 보다 스마트한 주거 환경을 조성하기 위하여 보급하여 활용하는 방식을 생각하고 있다.

1. 오래된 아파트 및 빌라 등 노후된 다세대 주택들은 공동현관문 출입하는데 있어서 요즘 아파트나 신식 빌라들과는 다르게 안전성이 떨어진다. 누구나 출입 가능한 경우도 있고, 비밀번호를 벽에 써놓는 경우도 있어서 비밀번호의 의미가 없어진 경우도 존재한다. 이러한 시설들의 공동현관문에 우리 프로젝트의 얼굴 인식 기능을 사용해서 건물 출입에 대해서 등록된 주민들에게 서비스를 제공하여 거주하는 사람들에게 안전함을 제공할 수 있다.

2. 아파트단지 내의 주차공간을 이용시 아파트 거주자가 아닌 사람들의 차량으로 인해서 주차공간이 부족해지고 자기가 살고있는 단지과 떨어져서 주차해야하는 경우가 있다. 이러한 경우에 우리 프로젝트에서 제공하는 차량 번호판 인식 기능을 아파트 출입이나 단지내 주차장 출입하는 곳에 적용하여서 실제 주민의 차량과 외부 차량을 구분하고 주민 차량에 대해 통과와 주차공간 배정 기능을 제공하고 외부차량을 통제하는 시스템을 간단히 제공할수 있다.

3. 국가가스안전공사가 국가통계포털사이트에 올린 가스사고 통계에 따르면 지난 2016년부터 지난해까지 가스사고발생 건수는 총 504건으로 집계되었는데, 가스사고발생으로 인한 인명피해는 446건으로 이중 45명이 사망하고 401명이 부상한 것으로 조사됐다.

사고원인으로는 사용자 취급부주의와 시설미비, 제품노후 또는 고장 등으로 인한 사고발생 잦았는데 이러한 경우 우리의 프로젝트 기능 중 이상탐지 기능을 활용하여 보다 손 쉽고 안전하게 사용자가 가스 누출 사고를 예방할 수 있다.

4. 후기

박동학

: 저희 팀은 3가지 주제를 선정하여 이를 통합적인 서비스로 완성하고자 노력했습니다. 노후화된 시설에 스마트 IoT를 보급할 수 있다는 아이디어에서 시작해 가장 실질적인 출입, 주차, 안전이라는 주제로 접근했으며 OpenCV, Tesseract 등 오픈소스를 적극 활용해 서비스를 제작하려고 계획했습니다.

프로젝트의 대부분이 이미지를 처리하는 것이었기 때문에 “Practical Python and OpenCV” 라는 교재를 선정하여 학습하면서 진행했습니다.

이 교재를 통해서 이미지를 전처리하는 방법에 대해서 학습했습니다. 얼굴 인식과 자동차 번호판 인식은 전처리 방법에 있어 과정이 비슷하게 작동하기 때문에 올바른 전처리를 위한 학습은 필수적이었습니다.

팀원 전체가 알고리즘에 대한 이해가 바탕이 되어 추후 프로젝트를 진행함에 있어 각자 맡은 역할을 잘 수행할 수 있었습니다.

제가 맡은 역할인 얼굴 인식, 차량 번호판 인식 알고리즘 부분을 수행하면서 높은 정확도를 보이기 위해서 노력했습니다.

얼굴 인식의 경우 여러 배경에서 찍은 사진을 학습시키기도 하고 스마트폰 카메라로 찍은 사진을 Resize 하여 학습시키기도 했습니다. OpenCV의 이미 설계된 모델을 위주로 프로젝트를 진행했기 때문에 이를 프로젝트 후반에 Tensorflow 기반의 새로운 모델로 교체하려고 시도했지만 부족한 시간과 부족한 하드웨어 성능으로 인해 교체하지 못한 것이 아쉬웠습니다.

차량 번호판 인식의 경우 전처리 과정을 어떻게 하느냐에 따라 인식률이 큰 차이를 보였기 때문에 전처리 과정에 최대한 집중을 했습니다. 글자 사이의 간격을 기준으로 번호판이라고 판단했기 때문에 외국의 번호판의 경우에는 인식하지 못한 것과 실제 차량에 대해서 실험하기에 전기를 공급하거나 차량을 섭외하는 것이 힘들어 수행하지 못한 부분이 아쉬웠습니다.

같은 학번의 동기들과 마지막 학년에 종합설계 프로젝트를 하면서 단국대학교에서 4년 동안 전공 분야 지식이 각기 다른 방향이지만 성장했음을 느낄 수 있었습니다. 저는 소프트웨어 분야에서 협업이 가장 중요하다고 생각합니다. 각자의 장점을 살려 프로젝트를 수행하는 방법을 학습 할 수 있는 좋은 시간이었다고

생각합니다. 코로나19 (COVID19)로 인해 강의실에서 만나 진행하지 못했지만 줌, 스카이프 등 다양한 도구를 이용해 의사소통에 큰 어려움 없이 잘 해결할 수 있었던 것 같습니다.

김승준

: 이번 프로젝트에서 주로 다룬 내용은 스마트 IoT 주거환경 설계였다. 내가 할 당받은 부분이 하드웨어가 관련되어 있었다.

하드웨어를 제대로 사용해본 적이 없어 처음엔 사용하는데 무척 애를 먹었다. 특히 무선 네트워크 사용하기 위해 렌 카드를 조립할 때 Jetson Nano kit가 부팅이 안되는 오류가 발생했는데 구글 자료를 찾아보고 공부를 해보면서 렌 카드 9, 10번 회로가 오류 원인이라는 것을 알고 테이프로 절연을 해서 부팅을 성공시켰다. 조원과 합심하여 하니 금방 해결할 수 있었던 것 같다.

첫 번째로 스마트 IOT 주거환경의 핵심 기술은 ‘인식’이다. 파이썬 코딩을 이용해 카메라 모듈과 연동하여 인식을 진행하였다. 그래서 별도의 GUI가 필요했는데, 이를 위해 PyQt5 - Qt Design 프로그램을 사용하기로 하였다. 처음 써보는 프로그램이라 기본 예제부터 만들어 보면서 감을 잡아갔다. 조원 중 혼자 3학년이라 미비한 부분이 있어서 좀 더 오래 걸렸지만 포기하지 않고 성공적인 GUI를 만들었다. 기억에 남는 부분은 Main window UI를 만들고 파이썬에 연동하면서 얼굴 인식 모듈을 실행하여 얼굴 인식을 성공했을 때다. 두 번째 설계는 라즈베리 파이 디스플레이에 준비한 가스센서를 이용해 이상 탐지, 위험 탐지를 해서 디스플레이 하는 것이다. 실시간 가스센서 정보를 띄우기 위해 GUI에 Qlistview가 필요했다. QTextbrowser 과는 쓰임이 많이 달라서 구현하는데 어려움이 있었는데, QStandardItemModel() 함수를 사용해 구현할 수 있었다.

이번 학기는 코로나-19로 인해 팀원과 합동하는데 제한이 있었지만 이렇게 한 학기 무사히 캡스톤디자인을 하게 해준 팀원과 교수님께 다시 한번 감사를 드립니다.

방승환

: 이번 프로젝트는 OpenCV, Tesseract OCR, Python, Jetson nano, Raspberry Pi, 기타 센서들을 통한 차량 번호판 인식, 사용자 안면인식, 이상탐지를 구현하였습니다.

이번 프로젝트를 진행하면서 처음으로 Jetson nano, Raspberry Pi를 구매하여 사용하였는데 직전학기 실무중심 산학 협력 프로젝트에서 소프트웨어만 사용하

여 진행했을 때 보다 이번프로젝트에는 하드웨어까지 같이 사용해보니 처음에는 진행이 다소 더디어지는 부분도 많았습니다. 하지만 처음부터 차근차근 하드웨어와 센서들의 사용법 찾아 배우는 과정을 통해 프로젝트를 진행할 수 있게 되었습니다. 그리고 하드웨어를 이용하다 보니 프로젝트를 진행하며 점점 완성도가 올라갈수록 결과물이 직접적으로 눈에 보여 흥미로웠던 것 같습니다.

또한 저는 PyQt를 통하여 Python GUI부분을 중점으로 구현하였는데 PyQt에서 구현한 UI(버튼과 결과를 출력하는 라벨)와 실질적으로 실행이 되어야하는 소스 코드, 저장된 외부 데이터를 연결하고 불러오는 과정에서 다소 난관이 있었습니다. 하지만 인터넷 검색을 통해 필요한 부분을 하나하나 찾아가면서 해결할 수 있었습니다.

비록 이번 코로나 때문에 팀원들과 직접적으로 자주 만나 프로젝트를 진행하지는 못했지만 화상회의를 통해 서로 진행방향을 잡아주고 부족한 부분을 채워주며 프로젝트를 성공적으로 마칠 수 있었습니다. 마지막으로 이번 프로젝트를 통해 소프트웨어영역 뿐만 아니라 하드웨어영역까지 다뤄보며 프로젝트 이전보다 더 넓은 분야의 스펙트럼을 넓힐 수 있는 뜻 깊은 계기가 된 것 같습니다.

홍승기 : 이번 학기는 코로나의 여파로 인해서 팀원들과의 의사소통과 계획을 실행하는데 많은 어려움을 겪었다. 하지만 팀원들과 화상회의나 자주는 아니지만 몇 번 정도 만나서 프로젝트를 진행할 수 있어서 생각보다는 어려움 없이 진행된 것 같아 다행이었다고 생각한다.

우리 팀이 진행했던 프로젝트는 IoT 기술을 이용하여 스마트 주거환경 서비스를 제공하는 플랫폼을 만들어 내는 것이었다. 처음에 구상했던 최종 결과물은 실제 환경(주차장, 건물 등)에서 우리가 만든 프로그램을 탑재한 하드웨어를 사용하여 시연을 해보는 것까지 가 최종 목표였는데 코로나 상황이 심각해짐에 따라서 결과물 테스트 방식의 방향을 전환하게 되었다.

프로젝트의 진행 방향은 안면인식 기능을 통해 공동 현관문 출입에 대한 안전성을 지원하는 것과, 차량 번호판 인식 기능을 통해서 세대원들에게 효율적이고 쾌적한 환경의 주차환경을 제공하면 어떨까에서 개발 방향을 잡고 프로젝트를 진행해 나갔다. 프로젝트를 진행 하면서, 이상 탐지, 위험 탐지 기능을 추가하여 더욱 실용적이고 안전함을 제공해 주는 기능 부분을 제공하여 프로젝트의 질을 더욱 높이는 방향을 찾아갔다. 학부 생활을 하면서 하드웨어적인 부분을 접해본 적이 없었는데, 이번 프로젝트를 하면서 젓슨나노, 라즈베리 파이 등 여러 하드웨어를 다루는 계기가 되어서 프로젝트에 굉장한 흥미를 가지고 진행할 수 있었

다. 처음 사용하는 하드웨어들이라 처음에는 사용에 미숙함이 많았지만, 프로젝트를 진행하면서 여러 사용법에 익숙해져서 다행이었던 것 같다. 지금 생각해봐도 정말 좋은 경험이 된 것 같다.

프로젝트에서 내가 맡은 부분은 하드웨어와 연동시킬 적당한 DB를 찾고 DB를 구축하는 것이었다. 여러 DB들을 탐색하면서 적당한 방법을 찾지 못하여 최종적으로 사용한 방법은 프로젝트에 사용하는 Python 언어에서 지원하는 pandas 모듈을 이용하여 작성된 프로그램에서 얻는 정보를 데이터 프레임을 이용하여 csv 파일로 저장하는 방식을 사용해서 프로그램들에서 수집되는 데이터를 수집하는 방향으로 진행하게 되었다. 위와 같은 방법을 사용하면서 데이터를 효과적으로 실제 사용되는 하드웨어 위에서 관리할 수 있어서 괜찮은 결과를 얻어냈다고 생각이 들었다.

최종 결과물이 팀원들과 자주 못 만나는 환경에서도 나름 괜찮게 나온 것 같아서 한편으로는 다행이라고 생각이 들었지만, 정상적인 학기 중에 팀원들과 더 자주 만나며 프로젝트를 진행했었다면, 더 좋은 결과물을 보여줄 수 있었을 것 같다는 생각 때문에 약간의 아쉬움이 남는 프로젝트였다.

Car_Number_Recog.py

```
##*-coding: utf-8*-
#한글인식을 위한 utf-8
import cv2
import matplotlib.pyplot as plt
import numpy as np
import pytesseract
#필요한 라이브러리 import

#번호판 인식을 위한 Python 모듈
def Recog():

    plt.style.use('dark_background')
    img_ori = cv2.imread('temp.jpg')
    # 폴더에 있는 temp.jpg 사진을 로드
    height, width, channel = img_ori.shape
    #높이, 너비, 채널 설정 ( 이미지 원래 모양대로 )

    #그레이색상으로 변경
    gray = cv2.cvtColor(img_ori, cv2.COLOR_BGR2GRAY)
    structuringElement = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
    imgTopHat = cv2.morphologyEx(gray, cv2.MORPH_TOPHAT,
    structuringElement)
    imgBlackHat = cv2.morphologyEx(gray, cv2.MORPH_BLACKHAT,
    structuringElement)
    imgGrayscalePlusTopHat = cv2.add(gray, imgTopHat)
    gray = cv2.subtract(imgGrayscalePlusTopHat, imgBlackHat)

    img_blurred = cv2.GaussianBlur(gray, ksize=(5, 5), sigmaX=0)
    #GaussianBlur를 통해서 잡음 제거

    img_thresh = cv2.adaptiveThreshold(
        img_blurred,
        maxValue=255.0,
        adaptiveMethod=cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
        thresholdType=cv2.THRESH_BINARY_INV,
        blockSize=19,
        C=9
    )
    #Threshold 과정을 통해 잡음 제거
```



```

contours, _ = cv2.findContours(
    img_thresh,
    mode=cv2.RETR_LIST,
    method=cv2.CHAIN_APPROX_SIMPLE
)

temp_result = np.zeros((height, width, channel), dtype=np.uint8)
cv2.drawContours(temp_result, contours=contours, contourIdx=-1, color=(255,
255, 255))
temp_result = np.zeros((height, width, channel), dtype=np.uint8)
contours_dict = []
#경계값 인식

for contour in contours:
    x, y, w, h = cv2.boundingRect(contour)
    cv2.rectangle(temp_result, pt1=(x, y), pt2=(x + w, y + h), color=(255, 255,
255), thickness=2)

    contours_dict.append({
        'contour': contour,
        'x': x,
        'y': y,
        'w': w,
        'h': h,
        'cx': x + (w / 2),
        'cy': y + (h / 2)
    })
#경계값 영역을 씌움 (사각형으로 영역 표시)

MIN_AREA = 80
MIN_WIDTH, MIN_HEIGHT = 2, 8
MIN_RATIO, MAX_RATIO = 0.25, 1.0
#영역간 간격 정의

possible_contours = []

cnt = 0
for d in contours_dict:
    area = d['w'] * d['h']

```

```

ratio = d['w'] / d['h']

if area > MIN_AREA \
    and d['w'] > MIN_WIDTH and d['h'] > MIN_HEIGHT \
    and MIN_RATIO < ratio < MAX_RATIO:
    d['idx'] = cnt
    cnt += 1
    possible_contours.append(d)

temp_result = np.zeros((height, width, channel), dtype=np.uint8)
#번호판이 될수 있는 후보 저장

for d in possible_contours:
    cv2.rectangle(temp_result, pt1=(d['x'], d['y']), pt2=(d['x'] + d['w'], d['y'] +
d['h']), color=(255, 255, 255),
                    thickness=2)

MAX_DIAG_MULTIPLYER = 5
MAX_ANGLE_DIFF = 12.0
MAX_AREA_DIFF = 0.5
MAX_WIDTH_DIFF = 0.8
MAX_HEIGHT_DIFF = 0.2
MIN_N_MATCHED = 3

def find_chars(contour_list):
    matched_result_idx = []

    for d1 in contour_list:
        matched_contours_idx = []
        for d2 in contour_list:
            if d1['idx'] == d2['idx']:
                continue

            dx = abs(d1['cx'] - d2['cx'])
            dy = abs(d1['cy'] - d2['cy'])

            diagonal_length1 = np.sqrt(d1['w'] ** 2 + d1['h'] ** 2)

            distance = np.linalg.norm(np.array([d1['cx'], d1['cy']]) -
np.array([d2['cx'], d2['cy']]))

```

```

        if dx == 0:
            angle_diff = 90
        else:
            angle_diff = np.degrees(np.arctan(dy / dx))
        area_diff = abs(d1['w'] * d1['h'] - d2['w'] * d2['h']) / (d1['w'] *
d1['h'])

        width_diff = abs(d1['w'] - d2['w']) / d1['w']
        height_diff = abs(d1['h'] - d2['h']) / d1['h']

        if distance < diagonal_length1 * MAX_DIAG_MULTIPLYER \
            and angle_diff < MAX_ANGLE_DIFF and area_diff <
MAX_AREA_DIFF \
            and width_diff < MAX_WIDTH_DIFF and height_diff <
MAX_HEIGHT_DIFF:
            matched_contours_idx.append(d2['idx'])

        matched_contours_idx.append(d1['idx'])

        if len(matched_contours_idx) < MIN_N_MATCHED:
            continue

        matched_result_idx.append(matched_contours_idx)

        unmatched_contour_idx = []
        for d4 in contour_list:
            if d4['idx'] not in matched_contours_idx:
                unmatched_contour_idx.append(d4['idx'])

        unmatched_contour = np.take(possible_contours,
unmatched_contour_idx)

        recursive_contour_list = find_chars(unmatched_contour)

        for idx in recursive_contour_list:
            matched_result_idx.append(idx)

        break

    return matched_result_idx
#올바른 간격을 가지는 사각형에 대해서 Tesseract를 이용해서 번호판의 글자를 추

```

출

```
result_idx = find_chars(possible_contours)

matched_result = []
for idx_list in result_idx:
    matched_result.append(np.take(possible_contours, idx_list))

temp_result = np.zeros((height, width, channel), dtype=np.uint8)

for r in matched_result:
    for d in r:
        cv2.rectangle(temp_result, pt1=(d['x'], d['y']), pt2=(d['x'] + d['w'], d['y']
+ d['h']),
                        color=(255, 255, 255),
                        thickness=2)

PLATE_WIDTH_PADDING = 1.3
PLATE_HEIGHT_PADDING = 1.5
MIN_PLATE_RATIO = 3
MAX_PLATE_RATIO = 10

plate_imgs = []
plate_infos = []

for i, matched_chars in enumerate(matched_result):
    sorted_chars = sorted(matched_chars, key=lambda x: x['cx'])

    plate_cx = (sorted_chars[0]['cx'] + sorted_chars[-1]['cx']) / 2
    plate_cy = (sorted_chars[0]['cy'] + sorted_chars[-1]['cy']) / 2

    plate_width = (sorted_chars[-1]['x'] + sorted_chars[-1]['w'] -
sorted_chars[0]['x']) * PLATE_WIDTH_PADDING

    sum_height = 0
    for d in sorted_chars:
        sum_height += d['h']

    plate_height = int(sum_height / len(sorted_chars) *
PLATE_HEIGHT_PADDING)
```

```

triangle_height = sorted_chars[-1]['cy'] - sorted_chars[0]['cy']
triangle_hypotenuse = np.linalg.norm(
    np.array([sorted_chars[0]['cx'], sorted_chars[0]['cy']]) -
    np.array([sorted_chars[-1]['cx'], sorted_chars[-1]['cy']])
)

angle = np.degrees(np.arcsin(triangle_height / triangle_hypotenuse))

rotation_matrix = cv2.getRotationMatrix2D(center=(plate_cx, plate_cy),
angle=angle, scale=1.0)

img_rotated = cv2.warpAffine(img_thresh, M=rotation_matrix,
dsiz=(width, height))

img_cropped = cv2.getRectSubPix(
    img_rotated,
    patchSize=(int(plate_width), int(plate_height)),
    center=(int(plate_cx), int(plate_cy))
)

if img_cropped.shape[1] / img_cropped.shape[0] < MIN_PLATE_RATIO or
img_cropped.shape[1] / img_cropped.shape[
0] < MIN_PLATE_RATIO > MAX_PLATE_RATIO:
    continue

plate_imgs.append(img_cropped)
plate_infos.append({
    'x': int(plate_cx - plate_width / 2),
    'y': int(plate_cy - plate_height / 2),
    'w': int(plate_width),
    'h': int(plate_height)
})

longest_idx, longest_text = -1, 0
plate_chars = []

for i, plate_img in enumerate(plate_imgs):
    plate_img = cv2.resize(plate_img, dsiz=(0, 0), fx=1.6, fy=1.6)
    _, plate_img = cv2.threshold(plate_img, thresh=0.0, maxval=255.0,

```

```

type=cv2.THRESH_BINARY | cv2.THRESH_OTSU)

    contours, _ = cv2.findContours(plate_img, mode=cv2.RETR_LIST,
method=cv2.CHAIN_APPROX_SIMPLE)

    plate_min_x, plate_min_y = plate_img.shape[1], plate_img.shape[0]
    plate_max_x, plate_max_y = 0, 0

    for contour in contours:
        x, y, w, h = cv2.boundingRect(contour)

        area = w * h
        ratio = w / h

        if area > MIN_AREA \
            and w > MIN_WIDTH and h > MIN_HEIGHT \
            and MIN_RATIO < ratio < MAX_RATIO:
            if x < plate_min_x:
                plate_min_x = x
            if y < plate_min_y:
                plate_min_y = y
            if x + w > plate_max_x:
                plate_max_x = x + w
            if y + h > plate_max_y:
                plate_max_y = y + h

    img_result = plate_img[plate_min_y:plate_max_y,
plate_min_x:plate_max_x]

    img_result = cv2.GaussianBlur(img_result, ksize=(3, 3), sigmaX=0)
    _, img_result = cv2.threshold(img_result, thresh=0.0, maxval=255.0,
                                type=cv2.THRESH_BINARY |
cv2.THRESH_OTSU)
    img_result = cv2.copyMakeBorder(img_result, top=10, bottom=10,
left=10, right=10,
                                borderType=cv2.BORDER_CONSTANT,
value=(0, 0, 0))

    chars = pytesseract.image_to_string(img_result, lang='kor',
config='--psm 7 --oem 0')

```

```

result_chars = ''
has_digit = False
for c in chars:
    if ord('가') <= ord(c) <= ord('힉') or c.isdigit():
        if c.isdigit():
            has_digit = True
            result_chars += c

plate_chars.append(result_chars)

if has_digit and len(result_chars) > longest_text:
    longest_idx = i

info = plate_infos[longest_idx]
chars = plate_chars[longest_idx]

img_out = img_ori.copy()

cv2.rectangle(img_out, pt1=(info['x'], info['y']), pt2=(info['x'] +
info['w'], info['y'] + info['h']),
              color=(255, 0, 0), thickness=2)
cv2.imwrite("result.jpg",img_out)
return chars

```

Car_start.py

```
import cv2
import time
import Car_Number_Recog

def gstreamer_pipeline(
    capture_width=400,
    capture_height=400,
    display_width=400,
    display_height=400,
    framerate=30,
    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,
        )
    )
    #Jetson Nano 카메라 사용을 위한 파이프라인

def start():
    camera = cv2.VideoCapture(gstreamer_pipeline(flip_method=0),
cv2.CAP_GSTREAMER)
    #Jetson Nano 카메라 오픈
    count = 0

    while True:
```



```
(grabbed, frame) = camera.read()
count += 1
if not grabbed:
    break

cv2.imshow("Tracking", frame)
time.sleep(0.025)

print(count)
if count == 40:
    cv2.imwrite('temp.jpg', frame)
    break
camera.release()
cv2.destroyAllWindows()
result = Car_Number_Recog.Recog()
# 카메라를 통해서 사진을 촬영 후 반환
return result
```

Face.py

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

def Face_reg():
    labels = ["Dong","Hong"] #사용자 라벨

    def gstreamer_pipeline(
        capture_width=400,
        capture_height=400,
        display_width=400,
        display_height=400,
        framerate=30,
        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,
            )
        )

    #Jetson Nano에서 사용할 카메라 파이프라인

    face_cascade = cv2.CascadeClassifier('./haarcascade_frontalface_default.xml')
    #OpenCV 모델 로드
    recognizer = cv2.face.LBPHFaceRecognizer_create()
```

```

recognizer.read("face-trainner.yml")
#recognizer 설정

cap          =          cv2.VideoCapture(gstreamer_pipeline(flip_method=0),
cv2.CAP_GSTREAMER)
#카메라 오픈

if cap.isOpened() == False :
    exit()

count =0
name = ''
while True :
    ret, img = cap.read()
    gray  = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    #그레이 색상으로 변경
    faces  =      face_cascade.detectMultiScale(gray,      scaleFactor=1.5,
minNeighbors=5)

    for (x, y, w, h) in faces :
        roi_gray = gray[y:y+h, x:x+w]

        id_, conf = recognizer.predict(roi_gray)
        #카메라로 부터 오는 frame을 recognizer로 전송
        print(id_, conf)
        #정확도 출력 (cmd창에)
        if conf>=50:
            font = cv2.FONT_HERSHEY_SIMPLEX
            name = labels[id_]
            cv2.putText(img, name, (x,y), font, 1, (0,0,255), 2)
            cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
            if conf>=67:
                count =1
            #정확도와 라벨명을 이미지에 찍운다.

        print("Recognize")

    if count ==1:
        cv2.imwrite('Face_result.jpg', img)
        break

```

```
cap.release()  
cv2.destroyAllWindows()  
#열어둔 카메라 닫기  
return name, conf
```

Total_GUI.py

```
# -*-coding utf-8-*-
import sys
import urllib.request
import cv2
import time
import numpy as np
import threading
import pytesseract
import matplotlib.pyplot as plt
from concurrent.futures import ThreadPoolExecutor
import PIL
import pandas as pd
import datetime

# import pdb

import Face
import start
import Car_Number_Recog

from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5 import uic

form_class = uic.loadUiType("./ui/main.ui")[0]

class MainWindow(QMainWindow, form_class):
    def __init__(self):
        super().__init__()
        self.setupUi(self) # 초기 화면을 띄우는 코드

        self.facebtn.clicked.connect(self.facebtnFunction) # 초기 화면의 1번 버튼 (
안면 인식 )
        self.carbtn.clicked.connect(self.carbtnFunction) # 초기 화면의 2번 버튼 (
차량 번호판 인식)

    def facebtnFunction(self):
```

```

        FaceMainClass(self) # 버튼 클릭시 안면 인식 기능 GUI를 띄운다.

def carbtnFunction(self): # 버튼 클릭시 차량 번호판 인식 기능 GUI를 띄운다.
    CarMainClass(self)

class FaceMainClass(QDialog): # 안면 인식 기능 GUI
    def __init__(self, parent):
        super(FaceMainClass, self).__init__(parent)
        option_ui = "./ui/FaceMain.ui"
        uic.loadUi(option_ui, self) # 지정된 UI 파일을 불러옴
        self.show() # UI 파일을 받아오고 출력하는 함수

        # 버튼에 기능을 연결하는 코드
        self.btn_1.clicked.connect(self.button1Function)
        self.btn_2.clicked.connect(self.button2Function)

    def button1Function(self): # btn_1이 눌리면 작동할 함수
        face1class(self)

    def button2Function(self): # btn_2가 눌리면 작동할 함수
        t_executor = ThreadPoolExecutor(1)
        t = t_executor.submit(Face.Face_reg) # 원할히 GUI를 구동하기 위해 스레드
        사용

        while True:
            if t.done(): # 스레드가 종료 되었을 때
                name, conf = t.result() # 결과값을 받아옴 Loop 탈출
                break

        self.qPixmapFileVar = QPixmap()
        self.qPixmapFileVar.load("Face_result.jpg")
        self.qPixmapFileVar = self.qPixmapFileVar.scaledToWidth(350)
        self.lbl_picture.setPixmap(self.qPixmapFileVar) # 안면인식을 통해 얻어온 결
        과 사진 출력
        self.lbl2_picture.setText(name) # 해당 사용자 이름 출력

        check = pd.read_csv('facedata.csv') # 저장된 csv 파일을 읽어옴
        temp = np.array(pd.DataFrame(check, columns=['name']))
        if name in temp: # 사용자 이름이 저장된 csv 파일에 존재한다면

```

```

self.lbl2_picture_2.setText("문 열림\n %.2f Accur" % conf) # 문열림
else:
    self.lbl2_picture_2.setText("미 등록 사용자")

f_name = []
f_home = []

class facelclass(QDialog): # 사용자 정보 등록 UI
    def __init__(self, parent):
        super(facelclass, self).__init__(parent)
        option_ui = "./ui/facel.ui"
        uic.loadUi(option_ui, self)
        self.show()
        self.btn_Text.clicked.connect(self.btn_printTextFunction) # 이줄은 저장 버
튼

    def btn_printTextFunction(self): # 저장버튼 함수

        # Lineedit의 글자를 배열에 저장
        f_name.append(self.lineedit_Test1.text()) # UI에 쓰여진 정보를 해당 변수에
저장

        f_home.append(self.lineedit_Test2.text())
        data = [[f_name[-1], f_home[-1]]]
        submission = pd.DataFrame(data) # 저장된 데이터를 데이터 프레임으로 변
환

        submission.to_csv('./facedata.csv', header=False, mode='a', index=False)
# 변환된 데이터 프레임을 csv 파일로 저장
        self.close()

c_name = []
c_home = []
c_num = []

class carlClass(QDialog): # 차량 등록 UI
    def __init__(self, parent):
        super(carlClass, self).__init__(parent)

```

```

option_ui = "./ui/car1.ui"
uic.loadUi(option_ui, self)
self.show()

self.btn_Text.clicked.connect(self.btn_printTextFunction) # 이줄은 저장 버
튼

def btn_printTextFunction(self): # 저장버튼 함수
    # Lineedit의 글자를 배열에 저장
    c_name.append(self.lineedit_Test1.text())
    c_home.append(self.lineedit_Test2.text())
    c_num.append(self.lineedit_Test3.text())
    data = [[c_name[-1], c_home[-1], c_num[-1]]]
    submission = pd.DataFrame(data) # 입력된 데이터를 데이터 프레임으로 변
환
    submission.to_csv('./cardata.csv', header=False, mode='a', index=False)
# 변환된 데이터 프레임을 csv 파일로 변환
    self.close()

class CarMainClass(QDialog): # 차량 번호판 인식 기능 GUI
    def __init__(self, parent):
        super(CarMainClass, self).__init__(parent)
        option_ui = "./ui/carmainTest.ui"
        uic.loadUi(option_ui, self)
        self.show()

        # 버튼에 기능을 연결하는 코드
        self.btn_1.clicked.connect(self.button1Function)
        self.btn_2.clicked.connect(self.button2Function)

    def button1Function(self): # 1번 버튼 클릭시

        car1Class(self)

    def button2Function(self): # 2번 버튼 클릭시

        t_executor1 = ThreadPoolExecutor(1)
        t1 = t_executor1.submit(start.start) # 원활한 GUI 구동을 위해 스레드 사용
        show = ""

```



```

while True:
    if t1.done(): # 스레드 종료 시
        print("Recognition complete")
        show = t1.result() # 결과를 저장

        break
    self.qPixmapFileVar = QPixmap()
    self.qPixmapFileVar.load("result.jpg")
    self.qPixmapFileVar = self.qPixmapFileVar.scaledToWidth(350)
    self.lbl_picture.setPixmap(self.qPixmapFileVar) # 결과 이미지 표시
    self.lbl2_picture.setText(show)

    check = pd.read_csv('cardata.csv') # csv파일을 읽어옴
    temp = np.array(pd.DataFrame(check, columns=['car']))

    if show in temp: # csv파일안에 등록된 번호와 동일하면
        self.lbl2_picture_2.setText("문 열림" + Location) # 문 열림 과 주차위치
지정
    else:
        self.lbl2_picture_2.setText("미 등록 사용자")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    myWindow = MainWindow()
    myWindow.show()
    app.exec_()

```

danger_detect.py

```
import os
import csv
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

data = pd.read_csv("./GAS_DATASET.csv")
X = data["TIMESTAMP"]
Y = data["GAS_DATA"]

# line_fitter = LinearRegression()
plt.plot(Y,'o')
plt.xlabel("time(ms)")
plt.ylabel("Gas_data")

plt.axhline(y = 250, color = "r", linewidth = 2)
plt.show()
```

gasLeakage.py

```
import sys
import spidev
import wiringpi as w

CS_MCP3208 = 8
SPI_CHANNEL = 0
SPI_SPEED = 1000000
ADC_CHANNEL = 0

FAN_MT_P_PIN = 4
FAN_MT_N_PIN = 17
WARNING_LEVEL = 700

class gasLeakage:

    def __init__(self):
        self.spi = spidev.SpiDev()
        self.spi.open(0, 0)
        self.spi.max_speed_hz = 1000000
        self.setupWiringPiGpio()
        self.initMQ5()
        self.initFan()

    def setupWiringPiGpio(self):
        if w.wiringPiSetupGpio() == -1:
            print("[ERROR] Error in wiringSetupGpio")
            sys.exit(1)

    def getSensorData(self):
        SensingVal = self.readMQ5(ADC_CHANNEL)

        return SensingVal

    def controlFan(self, sensorValue):
        if sensorValue >= WARNING_LEVEL:
            self.FanOn()
        else:
            self.FanOff()
```

```
def initFan(self):
    print("[DEBUG] Fan Initialize")
    w.pinMode(FAN_MT_P_PIN, 1)
    w.pinMode(FAN_MT_N_PIN, 1)
    self.FanOff()

def FanOn(self):
    w.digitalWrite(FAN_MT_P_PIN, 1)
    w.digitalWrite(FAN_MT_N_PIN, 0)

def FanOff(self):
    w.digitalWrite(FAN_MT_P_PIN, 0)
    w.digitalWrite(FAN_MT_N_PIN, 0)

def readMQ5(self, nAdcChannel):
    nAdcValue = self.ReadMcp3208ADC(nAdcChannel)

    return nAdcValue

def ReadMcp3208ADC(self, adcChannel):

    w.digitalWrite(CS_MCP3208, 0)
    nAdcValue = self.spi.xfer2([1, (8 + adcChannel) << 4, 0])
    nAdcValue = ((nAdcValue[1] & 3) << 8) + nAdcValue[2]

    w.digitalWrite(CS_MCP3208, 1)

    return nAdcValue

def initMQ5(self):
    if w.wiringPiSPISetup(SPI_CHANNEL, SPI_SPEED) == -1:
        print("[ERROR] Error in wiringPiSPISetup")
        sys.exit(1)

    print("[DEBUG] MQ-5 Initialization")
    w.pinMode(CS_MCP3208, 1)
```

isolation_forest.py

```
from sklearn.ensemble import IsolationForest
import pandas as pd
import csv
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from mpl_toolkits.mplot3d import Axes3D

clf=IsolationForest(n_estimators=50, max_samples=50, contamination=float(0.004),
                    max_features=1.0, bootstrap=False, n_jobs=-1,
random_state=None, verbose=0,behaviour="new")

GAS_DATA = pd.read_csv("./GAS_DATASET.csv")

# 50개의 노드 수, 최대 50개의 샘플
# 0.04%의 outlier 색출.
clf.fit(GAS_DATA)
pred = clf.predict(GAS_DATA)
GAS_DATA['Class']=pred
outliers=GAS_DATA.loc[GAS_DATA['Class']==-1]
outlier_index=list(outliers.index)
print(GAS_DATA['Class'].value_counts())

#####

pca = PCA(n_components=3)
scaler = StandardScaler()

X = scaler.fit_transform(GAS_DATA)
X_reduce = pca.fit_transform(X)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.set_zlabel("x_composite_3")
ax.scatter(X_reduce[:, 0], X_reduce[:, 1], zs=X_reduce[:, 2], s=4, lw=1,
label="inliers",c="green")
ax.scatter(X_reduce[outlier_index,0],X_reduce[outlier_index,1],
X_reduce[outlier_index,2],
lw=2, s=60, marker="x", c="red", label="outliers")
ax.legend()
```

```
plt.show()
#####
import numpy as np
from sklearn.decomposition import PCA
pca = PCA(2)
pca.fit(GAS_DATA)
res=pd.DataFrame(pca.transform(GAS_DATA))
Z = np.array(res)
plt.title("IsolationForest")
b1 = plt.scatter(res[0], res[1], c='green',
                 s=20,label="normal points")
b1 =plt.scatter(res.iloc[outlier_index,0],res.iloc[outlier_index,1], c='green',s=20,
edgecolor="red",label="predicted outliers")
plt.legend(loc="upper right")
plt.show()
```

sensing.py

```
import sys
from gasLeakage import *
import pandas as pd
import datetime
import time

def sen():
    count = 0
    temp = []

    gas = gasLeakage()

    while count <=20:
        getData = gas.getSensorData()
        gas.controlFan(getData)

        print("[DEBUG] Smoke Sensor Value = %u"%(getData))

        data = [[datetime.datetime.now(), getData]]
        submission = pd.DataFrame(data)
        submission.to_csv('./Gas_DataSet.csv', header = False, mode = 'a', index
= False)
        time.sleep(0.5)
        temp.append(getData)
        temp2.append(str(getData))
        count +=1
    return temp, temp2

if __name__ == "__main__":
    sen()
```

sensor.py

```
import sys
import urllib.request
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5 import uic
import sensing
import time
import matplotlib.pyplot as plt
from gasLeakage import *
import pandas as pd
import datetime

form_class = uic.loadUiType("sensor.ui")[0]

class WindowClass(QMainWindow, form_class) :
    def __init__(self) :
        super().__init__()
        self.setupUi(self)

        self.btn_1.clicked.connect(self.button1Function)

    def button1Function(self) :

        count =0
        temp = []
        temp2 = []
        gas = gasLeakage()

        while True:
            getData = gas.getSensorData()
            gas.controlFan(getData)
            print("[DEBUG] Smoke Sensor Value = %u"%(getData))
            data = [[datetime.datetime.now(), getData]]
            submission = pd.DataFrame(data)
            submission.to_csv('./Gas_DataSet.csv', header = False, mode = 'a',
index = False)
            time.sleep(0.5)
            temp.append(getData)
```



```

        temp2.append(str(getData))
        count +=1
        if count >= 15:
            break

    auth = 0
    for element in temp:
        if element >= 200:
            auth = 1

    if auth == 1:
        self.textBrowser_2.setPlainText("위험상황입니다.")
    else:
        self.textBrowser_2.setPlainText("정상입니다.")
    self.textBrowser_3.setPlainText("정상입니다.")

    view = self.listView
    model = QStandardItemModel()
    for f in temp2:
        model.appendRow(QStandardItem(f))
    view.setModel(model)

    plt.plot(temp, color='red')
    plt.xlabel("Time")
    plt.ylabel("Gas_data")
    plt.draw()
    fig = plt.gcf()
    fig.set_figwidth(320/fig.dpi)
    fig.set_figheight(240/fig.dpi)
    fig.savefig("myfile.png")
    self.qPixmapFileVar = QPixmap()
    self.qPixmapFileVar.load("myfile.png")
    self.qPixmapFileVar = self.qPixmapFileVar.scaled(320,240)
    self.lbl_picture.setPixmap(self.qPixmapFileVar)

if __name__ == "__main__" :
    app = QApplication(sys.argv)
    myWindow = WindowClass()
    myWindow.show()
    app.exec_()

```


“IoT 기술을 이용한 스마트 주거환경 서비스”


통합 프로그램 사용 설명서

사용 설명


실행을 위해 필요한 S/W


Jetson nano에서 프로그램을 실행시키기 위한 OS 및 실행파일,라이브러리

1. Jetson nano실행을 위한 OS인 ubuntu 탑재  ubuntu

2. 실행파일인 파이썬 파일(.py)및 라이브러리 설치 

Raspberry Pi에서 프로그램을 실행시키기 위한 OS 및 실행파일,라이브러리

1. Raspberry Pi실행을 위한 OS인 Raspbian 탑재 

2. 실행파일인 파이썬 파일(.py) 및 라이브러리 설치 



```
apt-get install python3
pip3 install cv2
pip3 install matplotlib
pip3 install numpy
pip3 install pytesseract
pip3 install scikitlearn
pip3 install mpl_toolkits
pip3 install PyQt5
```



- 다음 보시는 화면은 Jetson nano에서 시스템을 실행했을 때 띄워지는 시스템의 총 메인 화면입니다.
- ①은 눌렀을 때 안면인식의 메인 화면이 띄워지는 버튼
- ②는 눌렀을 때 번호판인식의 메인 화면이 띄워지는 버튼

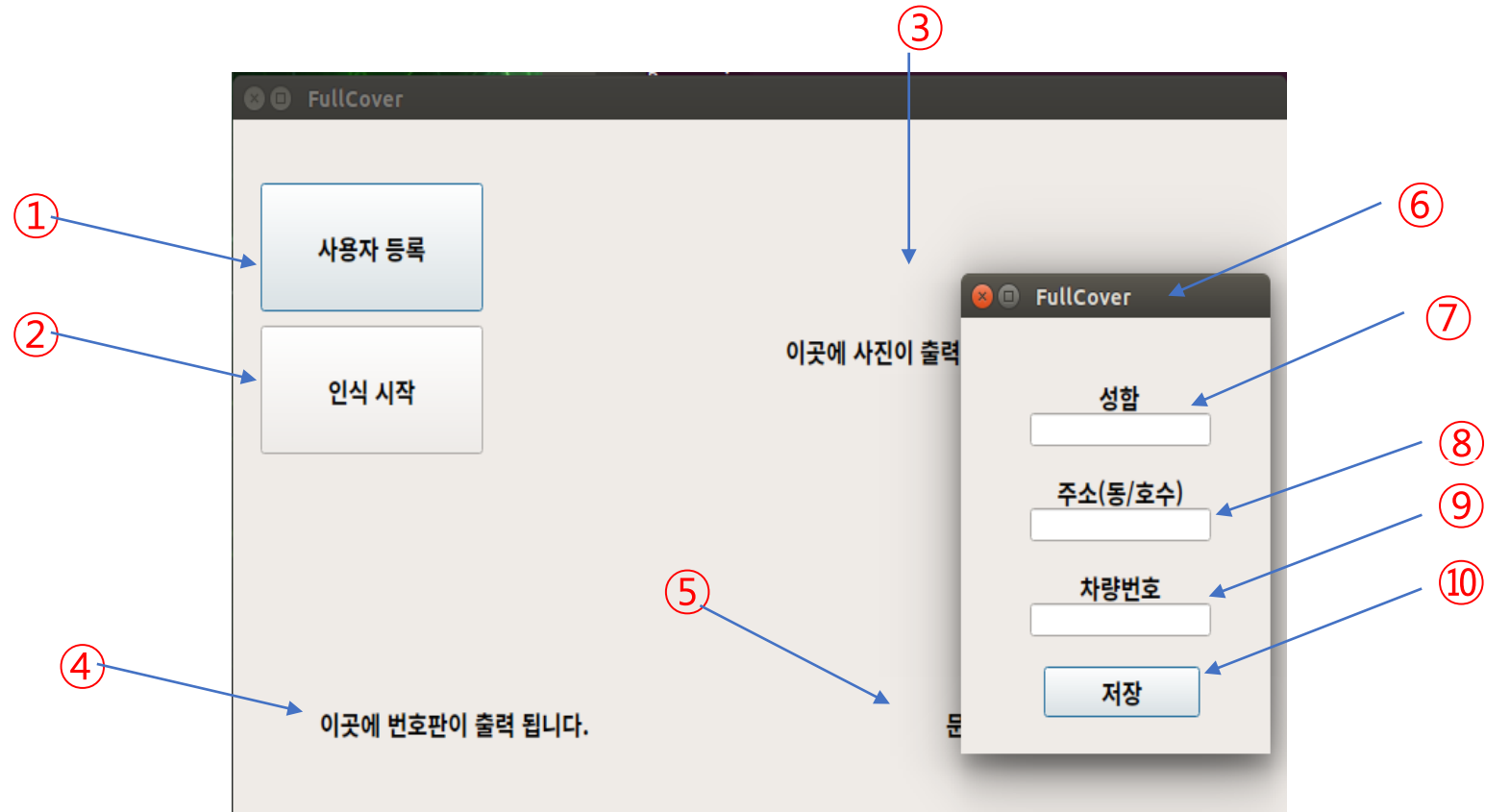
사용 설명



다음 화면은 안면인식의 메인화면과 사용자 등록 화면 입니다.

- ①은 사용자의 정보를 등록할 수 있는 ⑤화면을 띄워 주는 버튼
- ②는 안면인식을 실행시켜주는 버튼
- ③은 ②로 실행한 안면인식 프로그램을 실행 후 인식이 완료된 사용자의 사진이 출력
- ④는 ②로 실행한 안면인식 프로그램을 실행 후 인식이 완료되어 리턴되는 결과의 값을 출력
- ⑤사용자의 정보를 등록하는 화면
- ⑥,⑦사용자의 성함과 주소를 입력하는 칸
- ⑧ ⑥,⑦에 입력한 사용자의 정보를 저장하는 버튼

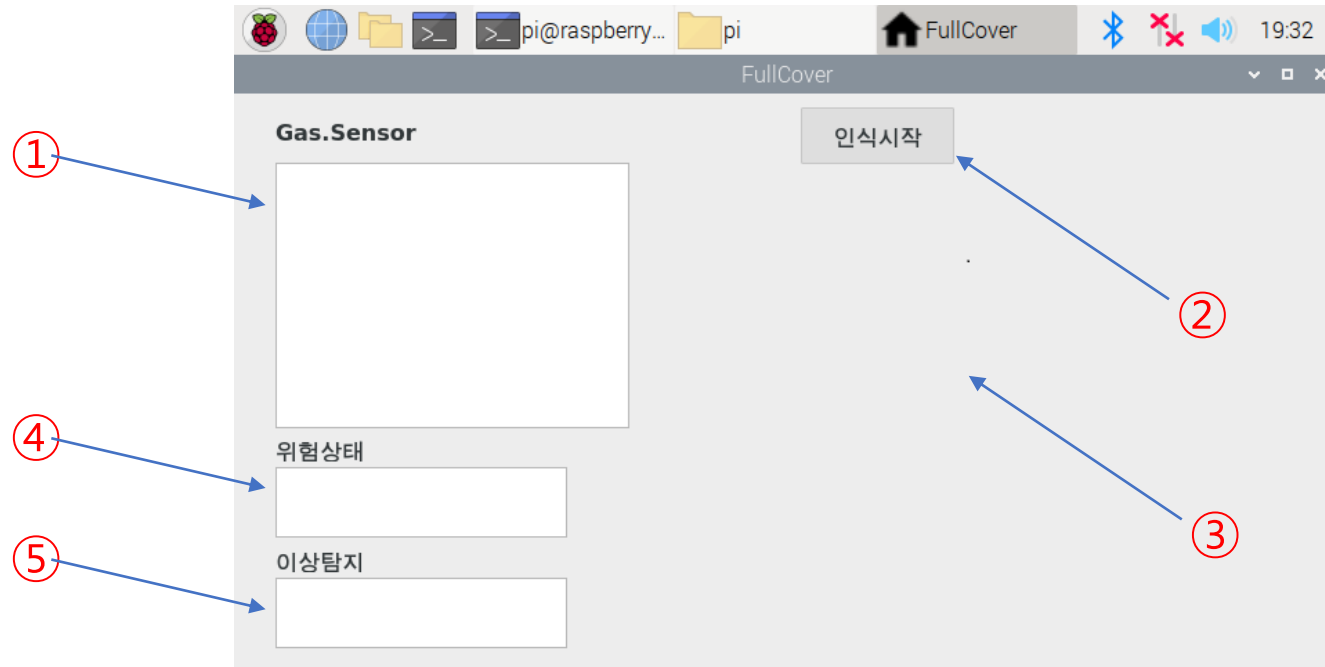
사용 설명



다음 화면은 번호판 인식의 메인화면 입니다.

- ①은 사용자의 정보를 등록할 수 있는 ⑥화면을 띄워 주는 버튼
- ②는 번호판인식을 실행시켜주는 버튼
- ③은 ②로 실행한 번호판인식 프로그램을 실행 후 인식이 완료된 번호판 사진이 출력
- ④는 ②로 실행한 번호판인식 프로그램을 실행 후 인식이 완료되어 리턴되는 결과의 값을 출력
- ⑤ ④에 리턴되는 결과값을 바탕으로 문열림 여부와 주차 공간 배정되어 출력
- ⑥ 사용자의 정보를 등록하는 화면
- ⑦,⑧,⑨ 사용자의 성함,주소,차량번호를 입력하는 칸
- ⑩ ⑦,⑧,⑨에 입력한 사용자의 정보를 저장하는 버튼

사용 설명



다음 화면은 라즈베리 파이에서 실행되는 이상탐지 화면 입니다.

- ① 가스데이터 실시간 출력 화면
- ② 가스데이터 인식 시작 버튼
- ③ 인식된 가스데이터가 그래프로 그려지는 곳
- ④ 가스데이터로 부터 이상이 있는지 없는지 나타내는 화면
- ⑤ 외부에서 비정상적인 접근이 시도되었는지 나타내는 화면