

## 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) # 원활히 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 버튼

    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 변환
        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 nano 실행을 위한 OS인 ubuntu 탑재  ubuntu

2. 실행파일인 파이썬 파일(.py) 및 라이브러리 설치



Raspberry Pi에서 프로그램을 실행시키기 위한 OS 및 실행파일, 라이브러리

1. Raspberry Pi 실행을 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

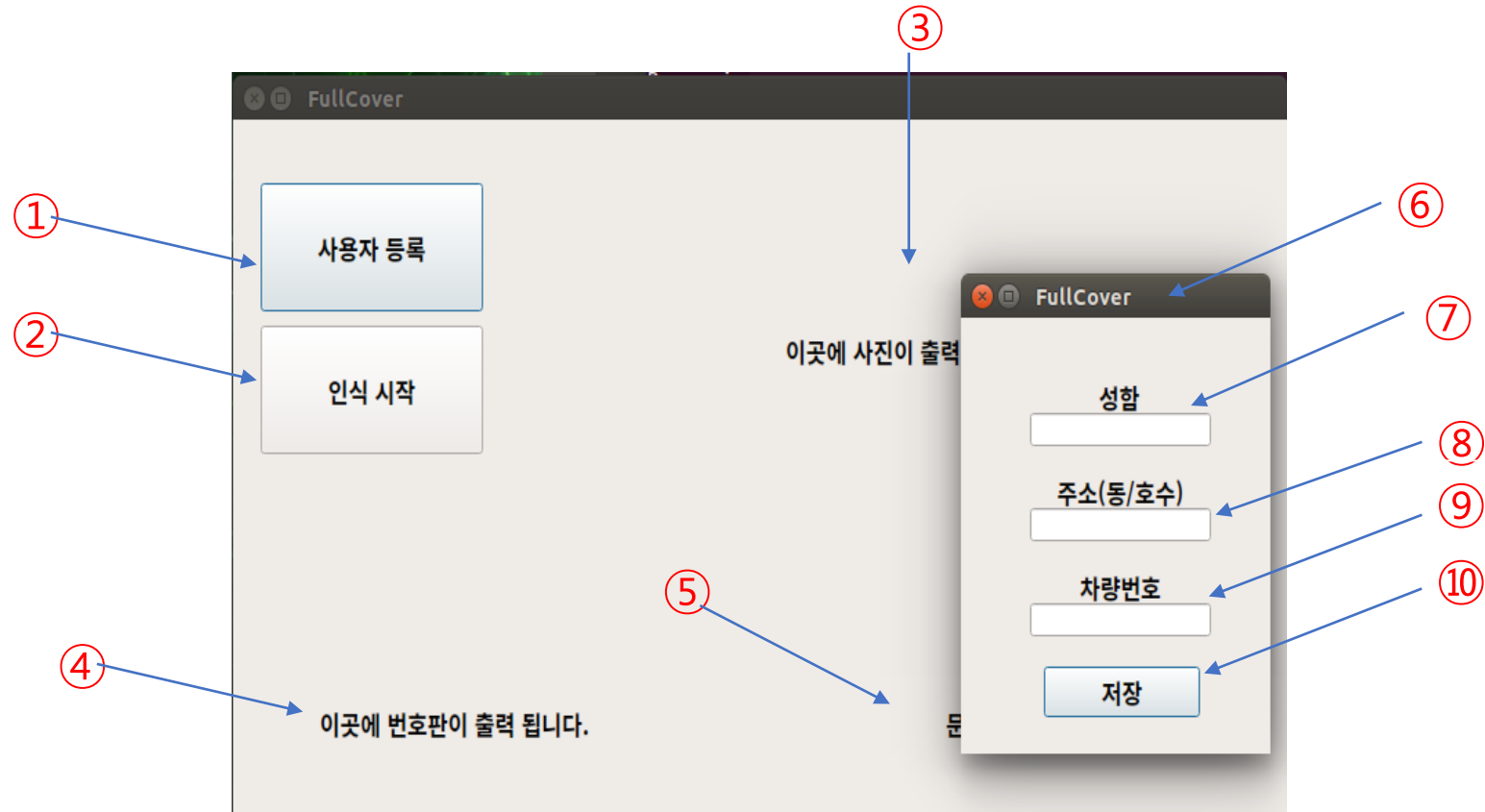
# 사용 설명



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

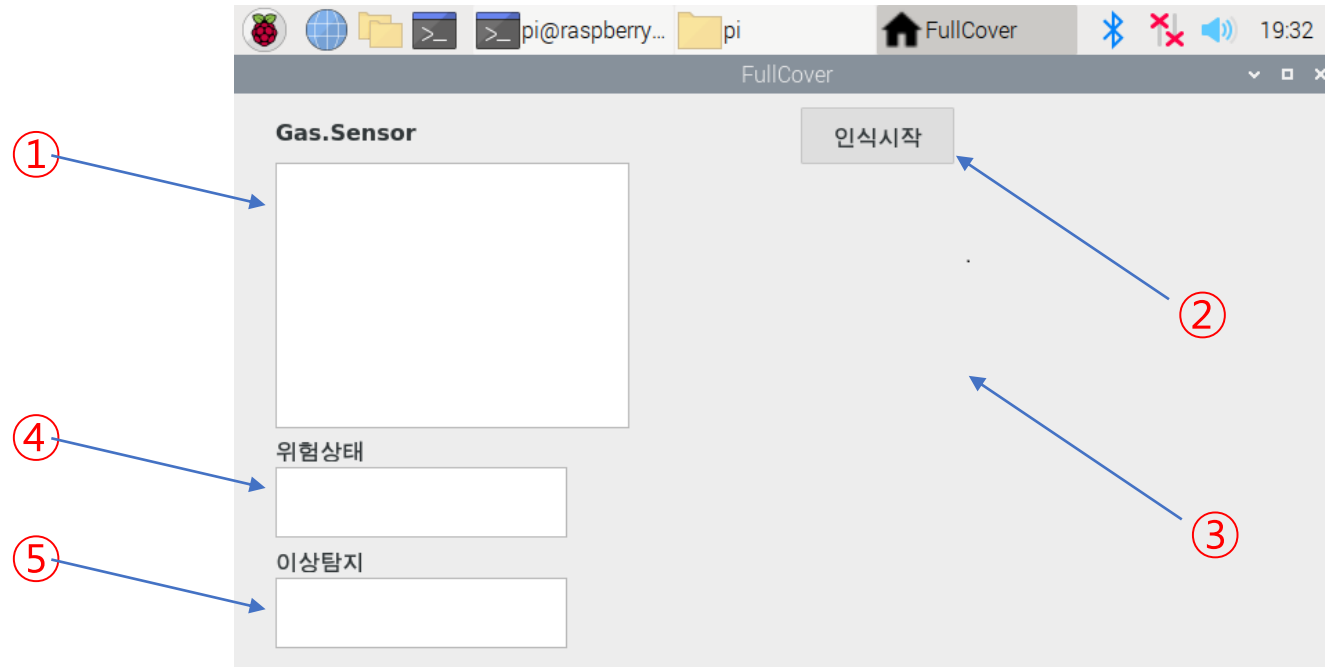
# 사용 설명



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