|  |
| --- |
| Result2018\_0001 |

<Where To Clean>

결과보고서

V1.7

2018년 09월 18일

[문서 정보]

|  |  |
| --- | --- |
| 문서관리고유번호 | Result2018\_0001 |
| 제품명 | Where to Clean |
| 제품유형 | 안드로이드, 응용프로그램 |
| 개발기관 | 한국교통대학교 |
| 기술책임자 | 박휘준, 황도현 |

[문서 이력관리]

|  |  |  |
| --- | --- | --- |
| **문서버전** | **내용** | **날짜** |
| V1.0 | 최초작성 | 2018.05.01 |
| V1.1 | 개요 수정 | 2018.05.03 |
| V1.2 | 유즈케이스 수정 | 2018.05.07 |
| V1.3 | 시퀀스 다이어그램 수정 | 2018.05.15 |
| V1.4 | 전반적 수정 | 2018.05.22 |
| V1.5 | 전반적 수정 | 2018.05.26 |
| V1.6 | 전반적 수정 | 2018.05.30 |
| V1.7 | 소스코드 추가 및 조사분석표 추가 | 2018.09.18 |
|  |  |  |
|  |  |  |

목 차

[1 개요 4](#_Toc515460192)

[1.1 일반 현황 4](#_Toc515460193)

[1.2 소프트웨어 기능 4](#_Toc515460194)

[1.3 유사 소프트웨어 비교 4](#_Toc515460195)

[1.4 운영 환경 5](#_Toc515460196)

[2 프로젝트 개발 방법론 및 절차 6](#_Toc515460197)

[2.1 프로젝트 방법론 6](#_Toc515460198)

[2.2 단계별 세부 절차 6](#_Toc515460199)

[2.2.1 계획 및 요구분석 6](#_Toc515460200)

[2.2.2 설계 6](#_Toc515460201)

[2.2.3 버전 개발 6](#_Toc515460202)

[3 프로젝트 범위 7](#_Toc515460203)

[3.1 물리적 범위 7](#_Toc515460204)

[3.2 논리적 범위 7](#_Toc515460205)

[4 시스템 아키텍처 8](#_Toc515460206)

[4.1 서브시스템 기반 아키텍처 8](#_Toc515460207)

[4.2 서브시스템 상세 설명 8](#_Toc515460208)

[4.2.1 Data flow 8](#_Toc515460209)

[4.2.2 User View 9](#_Toc515460210)

[4.2.3 Alert Flow 9](#_Toc515460211)

[5 기능별 명세 10](#_Toc515460212)

[5.1 Use Case 10](#_Toc515460213)

[5.2 Flow Chart 11](#_Toc515460214)

[5.2.1 관리자 11](#_Toc515460215)

[5.2.2 모바일 12](#_Toc515460216)

[5.2.3 영상 시스템 13](#_Toc515460217)

[5.2.4 데이터 처리 14](#_Toc515460218)

[5.2.5 패턴분석 및 메일링 시스템 15](#_Toc515460219)

[5.3 Sequence Diagram 16](#_Toc515460220)

[5.4 Class Diagram 19](#_Toc515460221)

[.6소프트웨어 결과 20](#_Toc515460223)

[6.1 소프트웨어 결과 캡처화면 20](#_Toc515460224)

# 개요

## 일반 현황 및 선정배경

본 프로젝트는 최근 대기오염으로 인해 대두된 가스와 미세먼지에 대한 관심에서 시작되었다. 미세먼지가 사회적인 의제가 되었지만 아직 가정이나 기업, 시설 내에서의 미세먼지와 가스에 대해선 거의 무관심하다. 2018년 최저시급 7530원, 월 최저 임금은 1,573,770(주 40시간, 월 209시간 기준)이다. CCTV를 사용하는 회사, 시설 또는 학교에서 사용하는 기존의 CCTV를 사용하고, Where To Clean 용으로 사용하는 서버 PC는 모니터까지 100만원 선에서 구매하게 된다면, 인력을 효율적으로 활용하여, 경제적으로 이익을 얻을 수 있다. ‘Where To Clean’은 실내에서의 대기 오염을 관리해주고 효율적으로 청소를 할 수 있게 도와주는 프로젝트이다.

## 소프트웨어 기능

센서를 이용해 시설의 먼지와 가스 데이터를 수집하여 어느 곳을 우선적으로 청소해야 할지 제시해준다. 이때 인공지능 영상처리 기술을 사용하여 오염도가 심하지만 청소가 급하지 않은 지역과 유동인구수가 많아 오염도와는 관계없이 청소가 요구되는 지역을 구별한다. 비정상적인 가스가 증가하거나, 오염도가 매우 심할 경우 모바일 기기에 알람을 전송해준다.

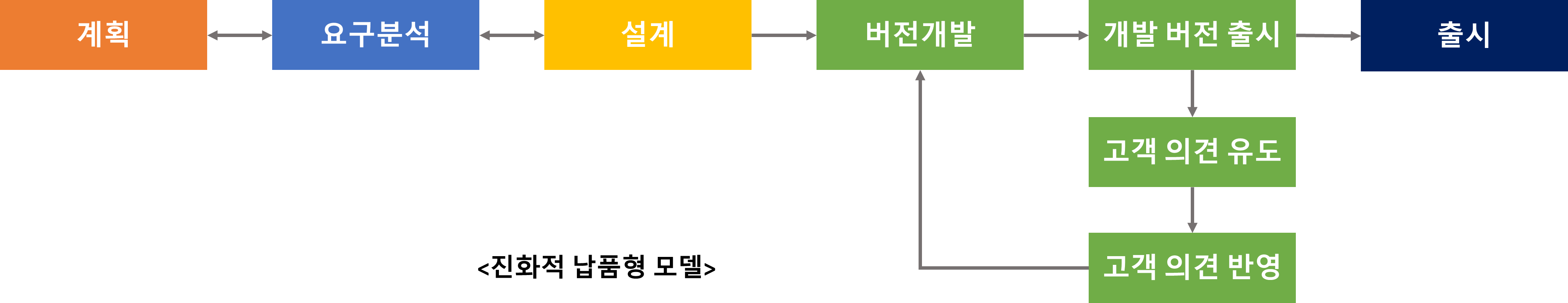
## 유사 소프트웨어 비교

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 구분 | 기능 | WhereToClean | 인천공항  smart airport | 한화테크윈  smart CCTV | 비고 |
| 기존 | 용도 | 효율적 청소관리 | 효율적 공항관리 | 방범용 |  |
| 사용대상 | 기업 및 공공기관 | 공항 |  |  |
| 적용기술 | Object Detection + 가스센서 + 모바일 실시간 모니터링 | 지능형 cctv + 가스센서 + 습도센서 + 로봇 | 지능형 cctv |  |
| 차이점 | Object Detection과 보행자 밀집도 대한 히트맵 분석 | 로봇을 활용한 청소관리 | 지능형 탐지 기술을 활용한 도난방지 |  |
| 특징 | 저비용 고효율 | 고비용 |  |  |
| WhereToClean은 타 제품에 비해, 저렴한 센서(cctv, 아두이노)를 사용하기 때문에, 비용면에서 우수하고, 우선순위 분석을 통해 사람을 효율적으로 관리 함. | | | | |
| 조사방법 | Web 검색 | | | | |
| 특허청 – 국내 특허 | | | | |
| 특허청 – 해외 특허 | | | | |

## 운영환경

# 프로젝트 개발 방법론 및 절차

## 프로젝트 방법론



## 단계별 세부 절차

### 계획 및 요구분석

기존에 있던 프로그램을 분석하여 개발 계획을 세우고 요구사항을 분석한다.

### 설계

로그와 데이터를 저장할 DB를 설계한다.

DB를 관리하고 값을 주고받을 PC서버를 설계한다.

Wifi를 이용한 센서와 PC, 모바일기기 간의 통신구조를 설계한다.

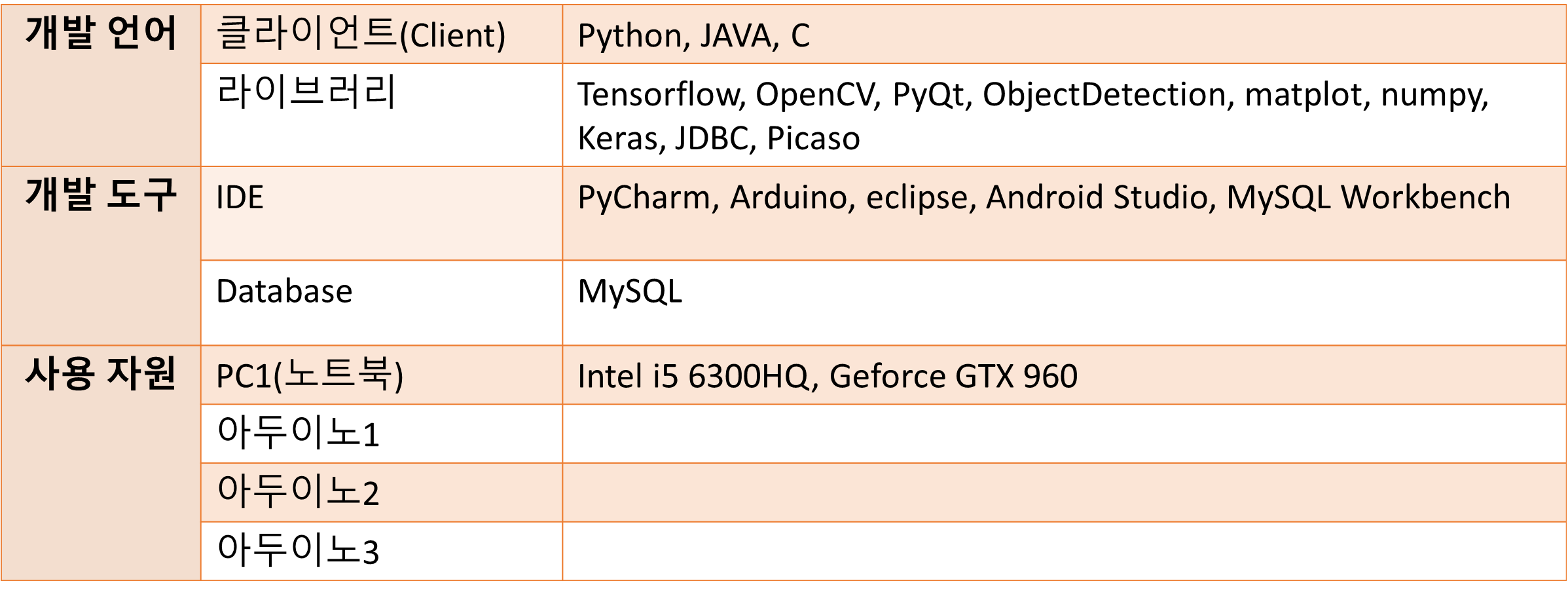
패턴분석, 영상분석 알고리즘을 설계한다.

### 버전 개발

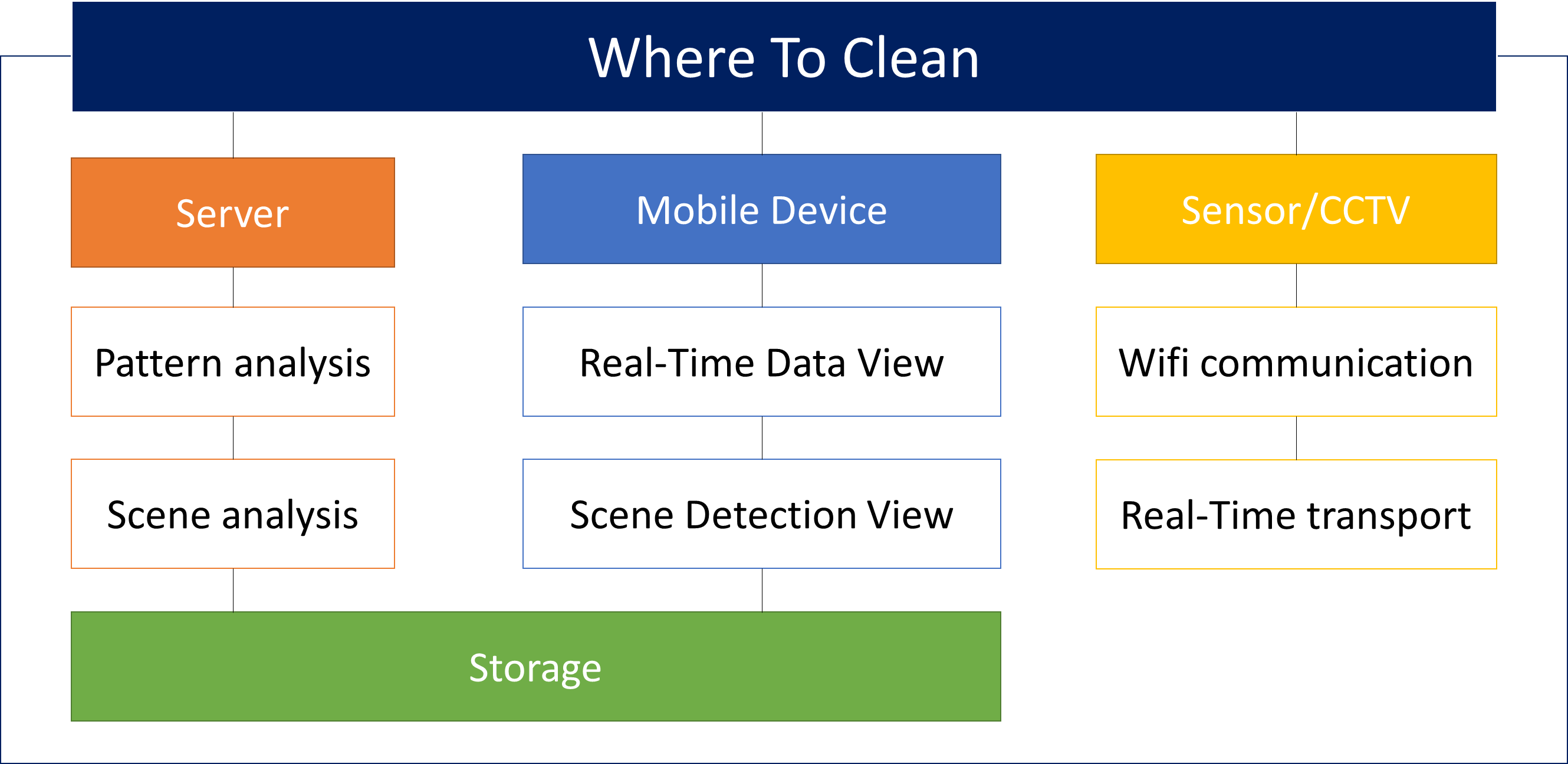
빠르게 첫번째 프로토타입을 완성시킨 후 피드백을 받아 계속 버전을 갱신한다.

# 프로젝트 범위

## 물리적 범위

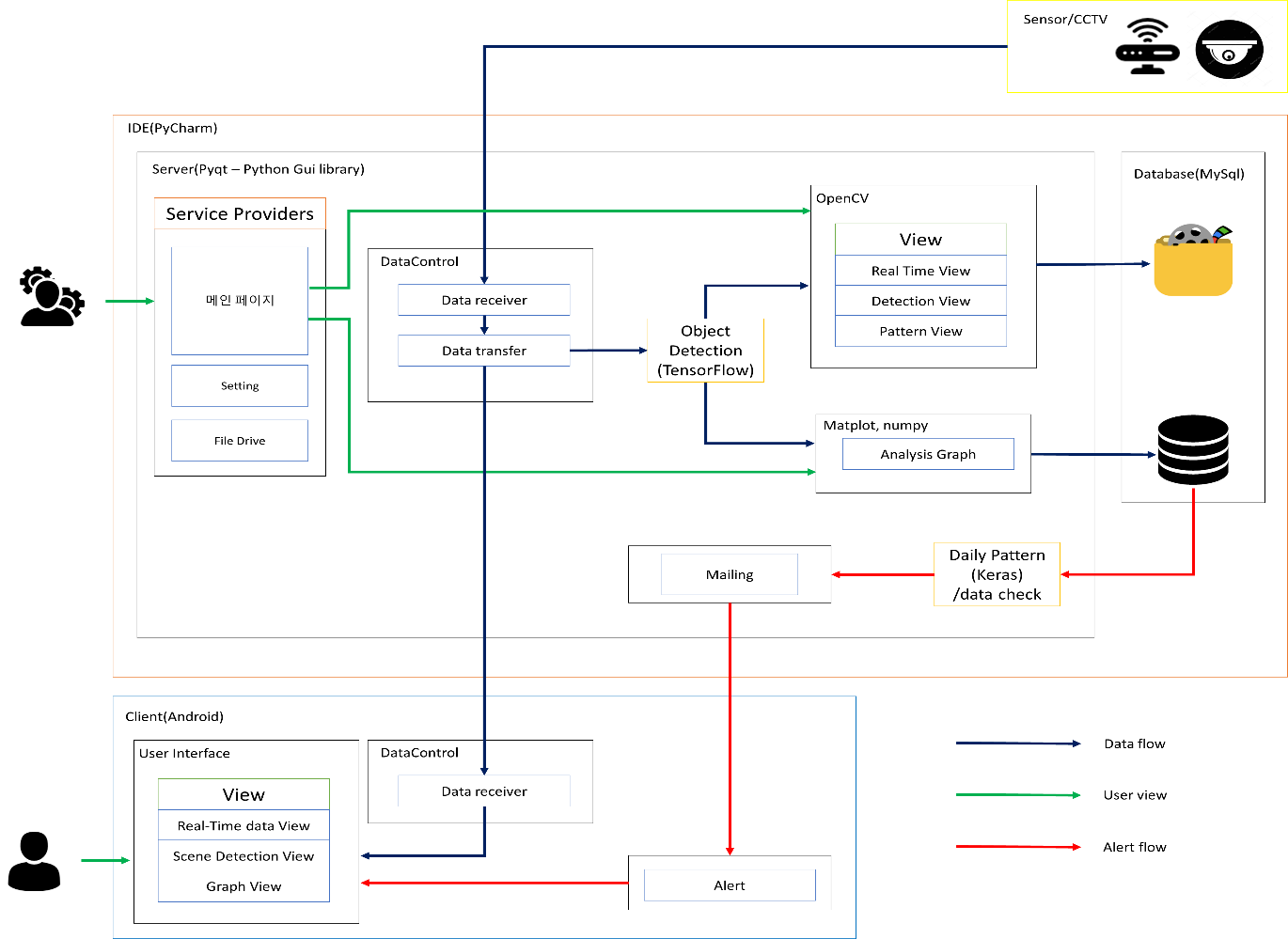


## 논리적 범위



# 시스템 아키텍처

## 서브시스템 기반 아키텍처



## 서브시스템 상세 설명

### Data flow

각 센서(cctv, 아두이노 센서)로부터 받아온 데이터의 값을 DataControl의 data receiver를 이용하여 값을 받아온다.

cctv영상 정보는 object detection으로 전달 후 처리되어 View를 보여준다. 센서 값과 영상 처리 값으로 Graph를 그린다. 멀티미디어 파일은 지정 폴더에 저장되고, 분석된 값은 데이터베이스에 저장합니다.

각 사용자(청소부)는 필요한 값을 받아와 모바일 기기에 각종 View 보여준다.

### User View

1) 관리자

서버 PC에서 프로그램을 실행하게 되면, 메인 페이지, 세팅, File Drive 선택이 가능하다.

메인 페이지에서는 현재 프로그램의 상황(view, graph)을 실시간으로 확인 할 수 있다.

세팅에서는 폴더 지정, 탐지 종류, 인터페이스 수정을 할 수 있다. File Drive에서는 저장된 멀티미디어파일(이미지, 동영상) 등을 확인 가능하다.

2) 사용자

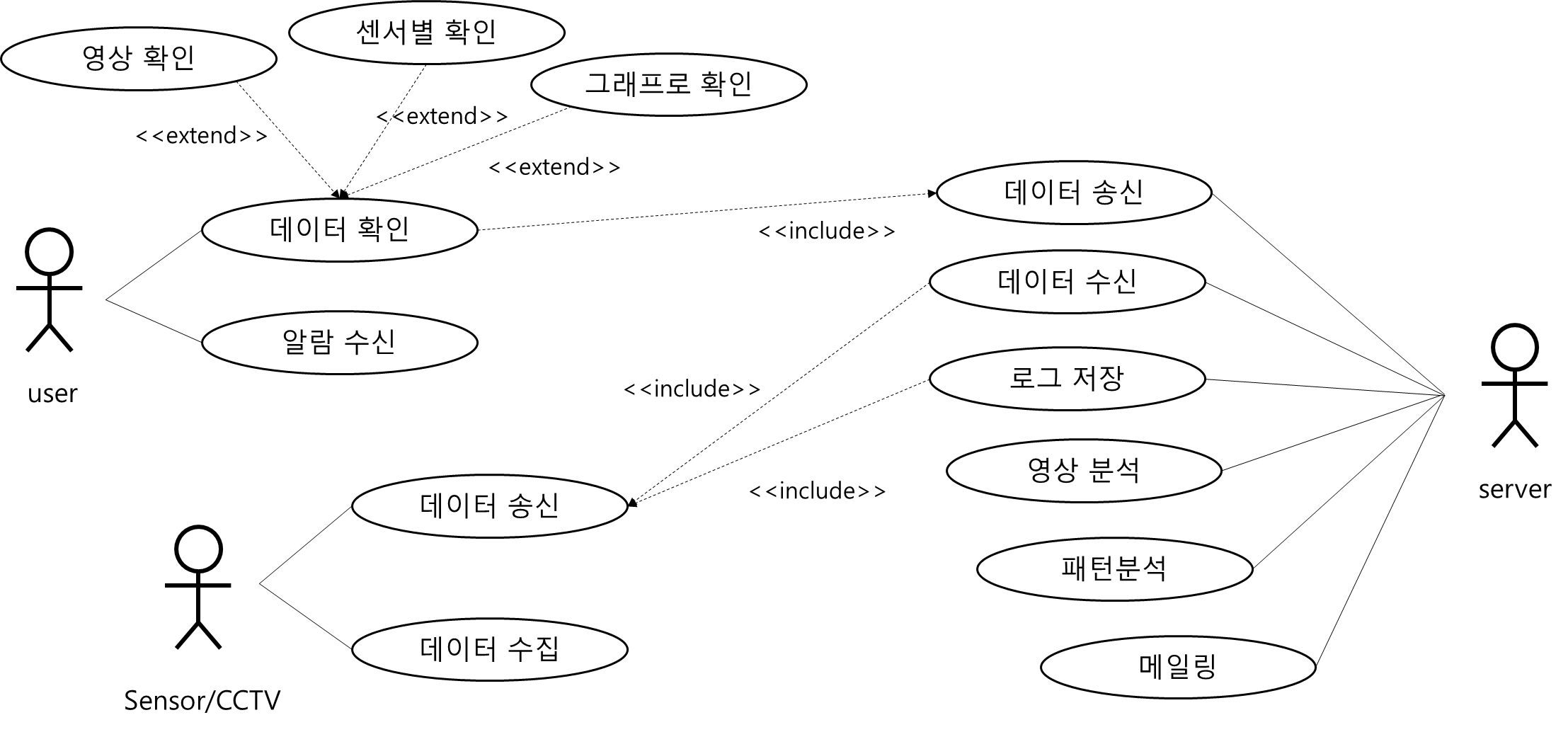
개인이 휴대한 스마트폰 어플리케이션에서 실시간 상황(view)을 확인 가능하며, 알림 여부를 선택하여 효율적인 경로를 선택 할 수 있다.

### Alert Flow

데이터베이스로부터 일정 시간마다 값을 가져와, 분석한 후 우선순위를 정하고, 사용자에게 메일링을 한다. 메일링에서는 이미지, 시간, 위치 등 정보를 사용자에게 제공한다.

# 기능별 명세

## Use Case



**1) user**

- 유저는 가스/영상 데이터를 확인하고 알람 메시지를 수신할 수 있다.

- 데이터를 확인하는 방법에는 각 센서별로 확인하는 방법과 그래프로 확인하는 방법이 있다.

- 확인하고자 하는 호실을 선택하면 영상데이터를 확인할 수 있다.

**2) server**

- 서버는 센서로부터 데이터를 수신하고 수신 받은 데이터의 로그를 저장한다.

- 수신 받은 데이터를 분석한 후 분석된 데이터를 DB에 저장한다.

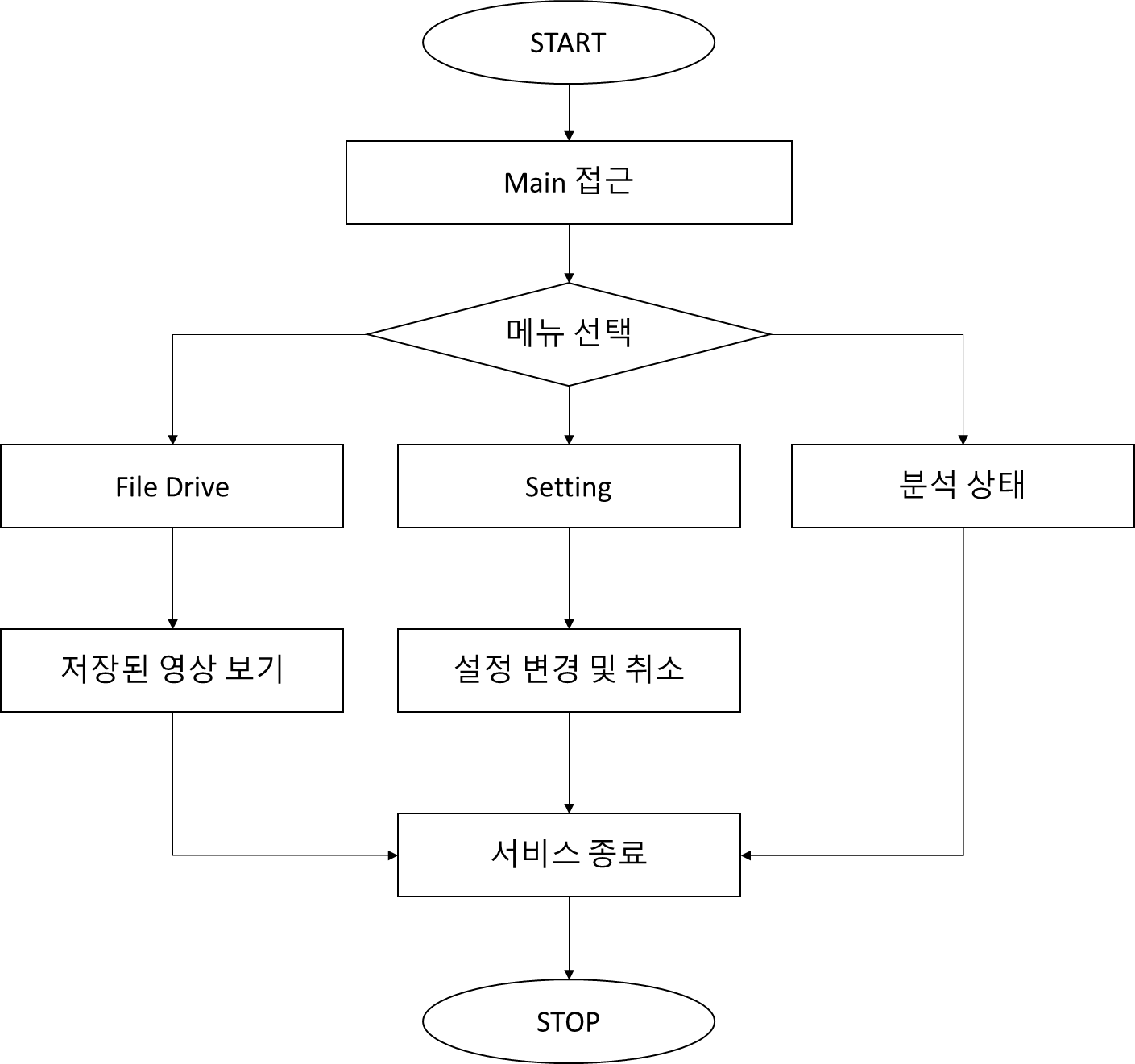
- 저장된 데이터를 모바일 기기에 송신한다.

**3) sensor/CCTV**

- 센서는 데이터를 수집하여 와이파이를 이용해 서버로 송신한다.

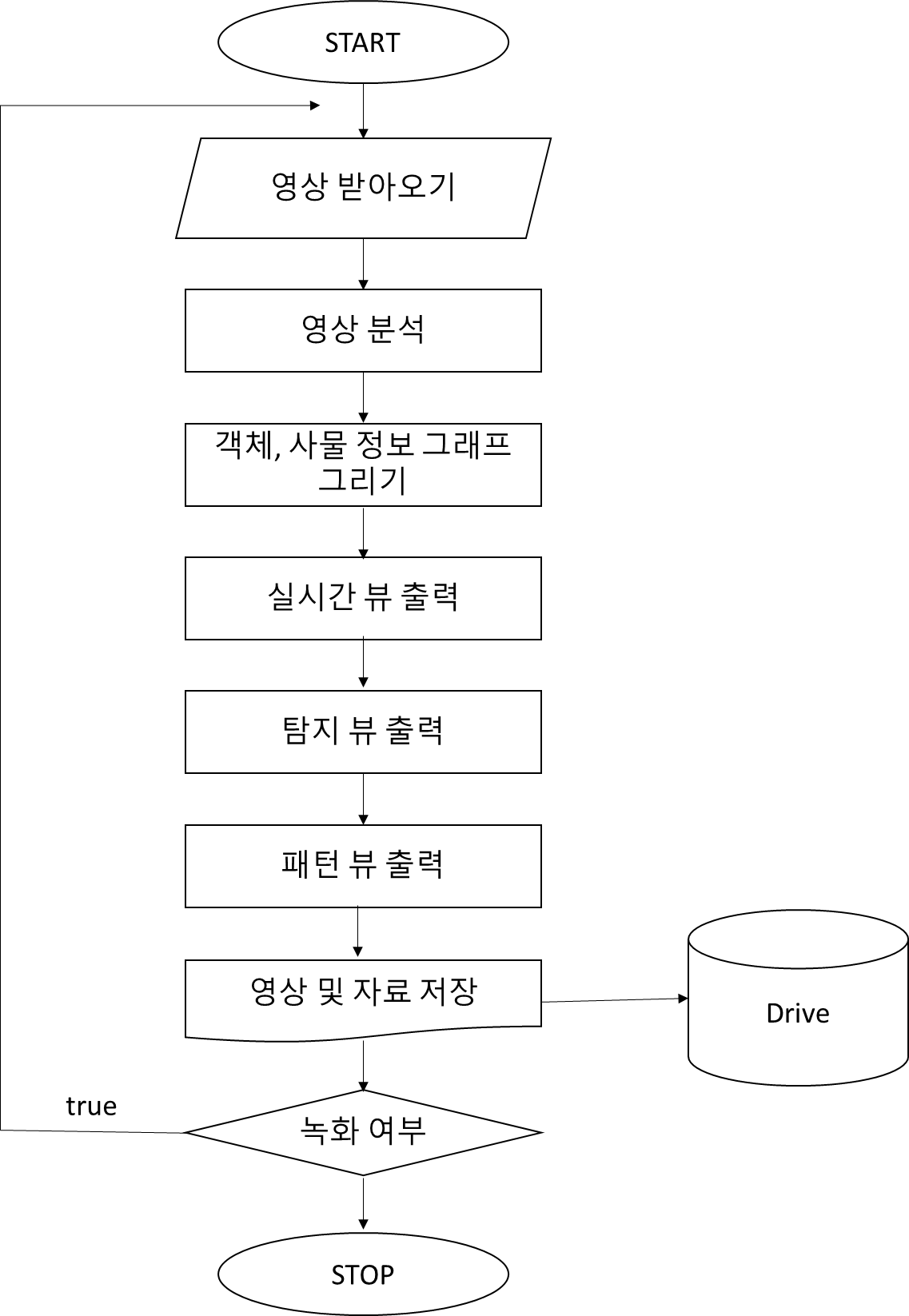
## Flow Chart

### 관리자

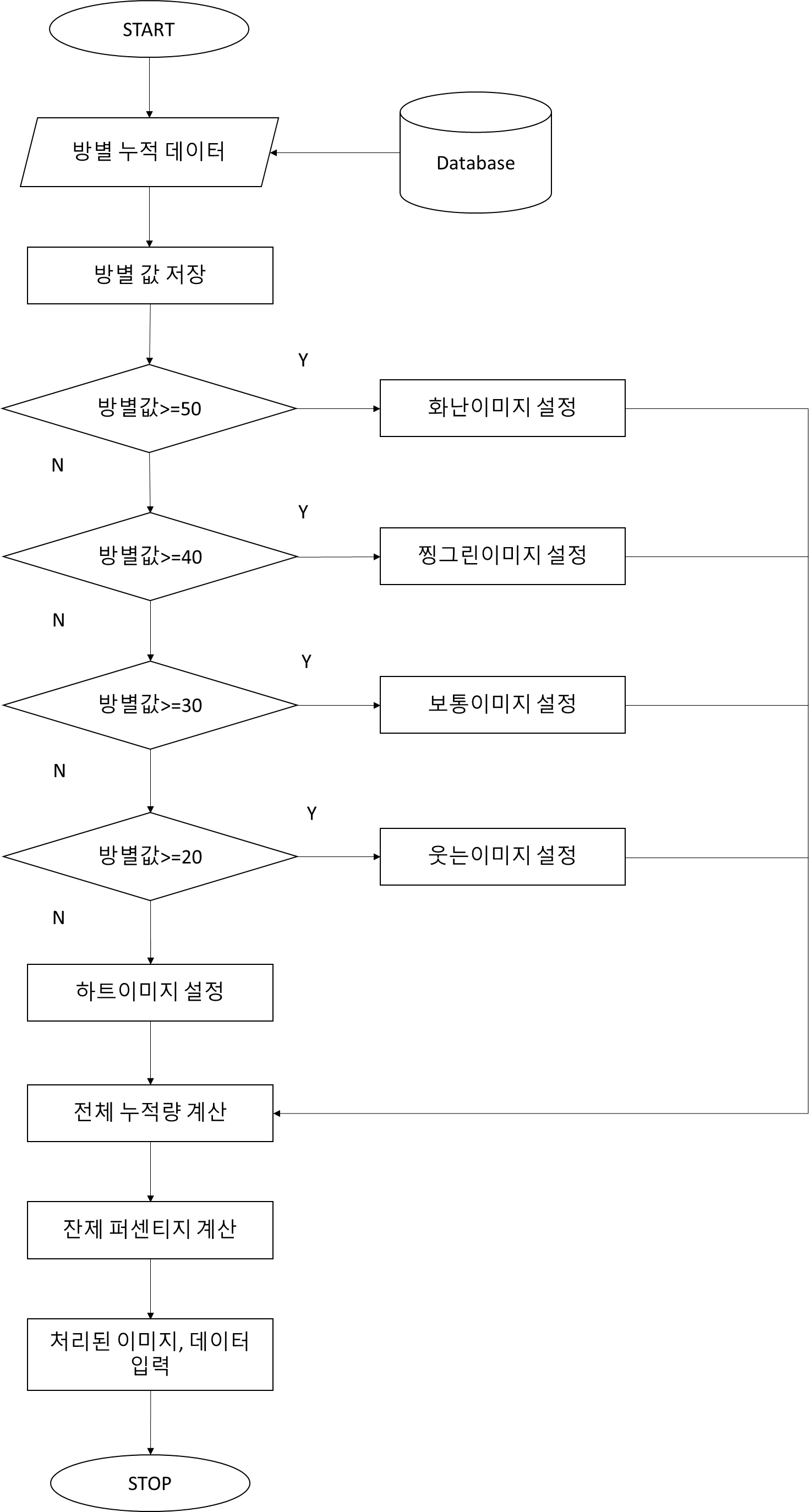


### 모바일

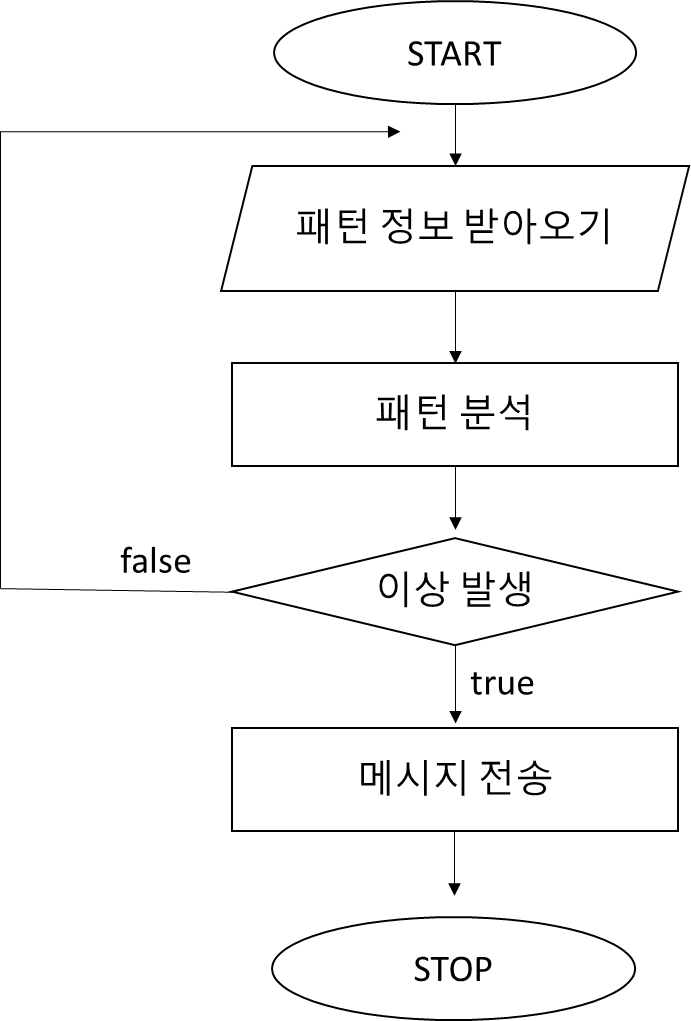
### 영상 시스템



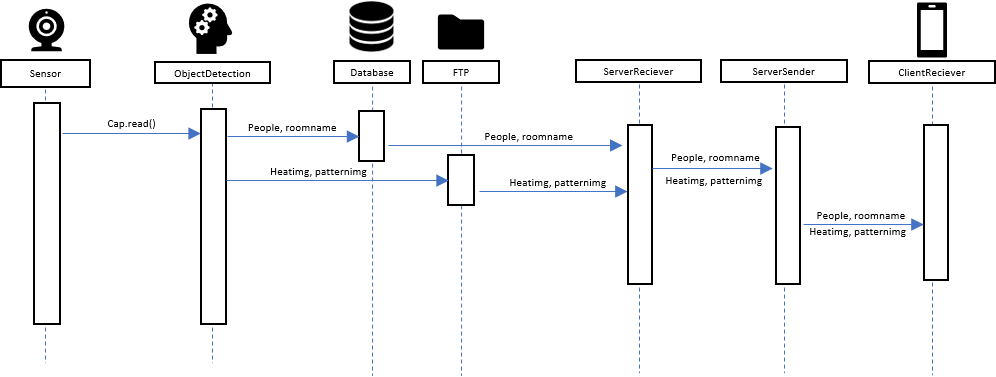
### 데이터 처리



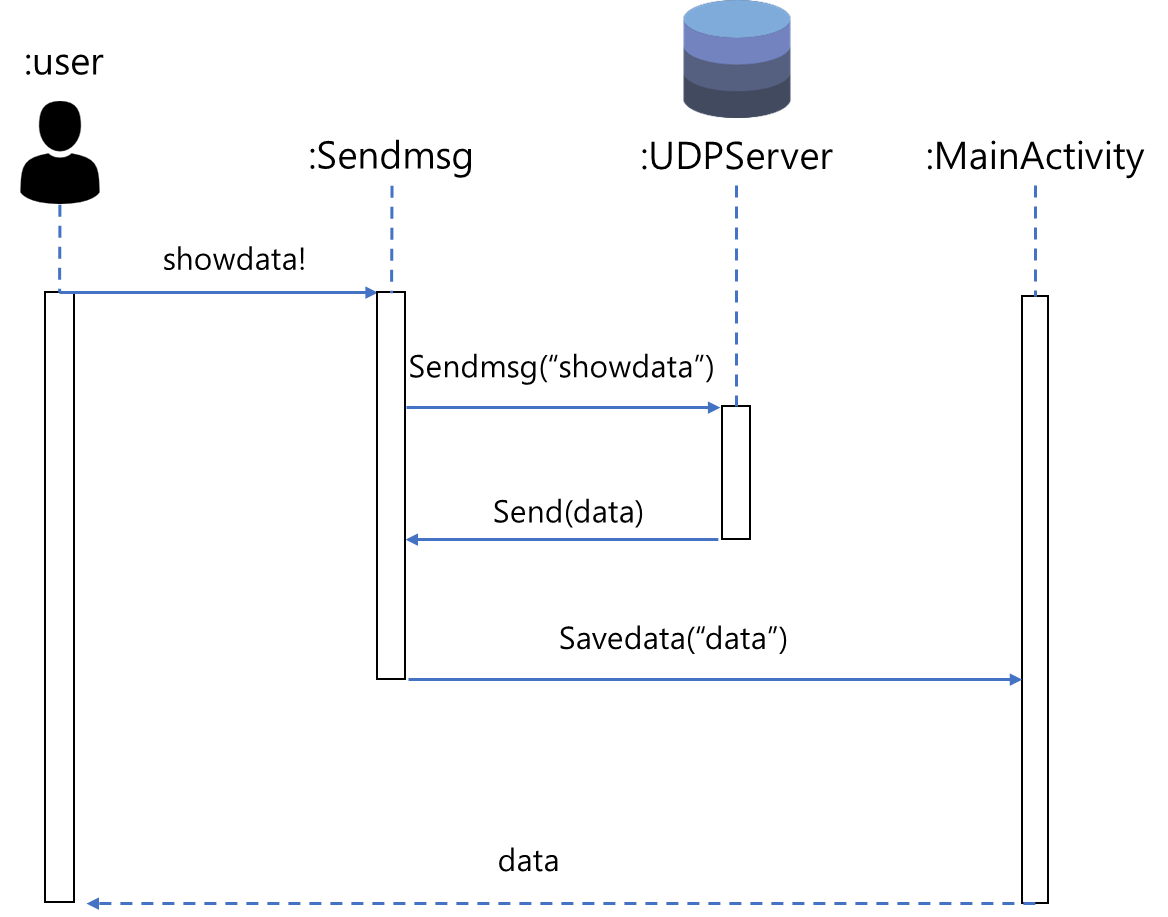
### 패턴분석 및 메일링 시스템



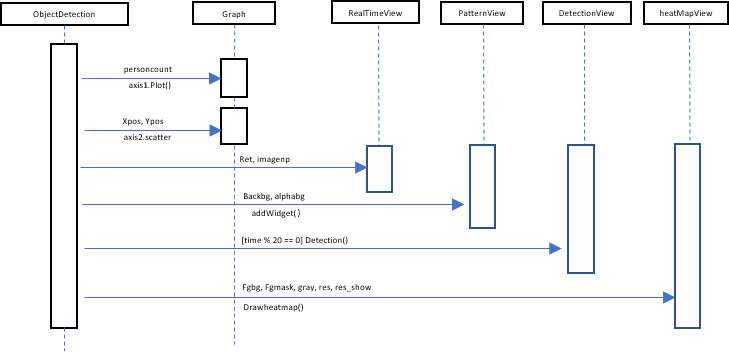
## Sequence Diagram



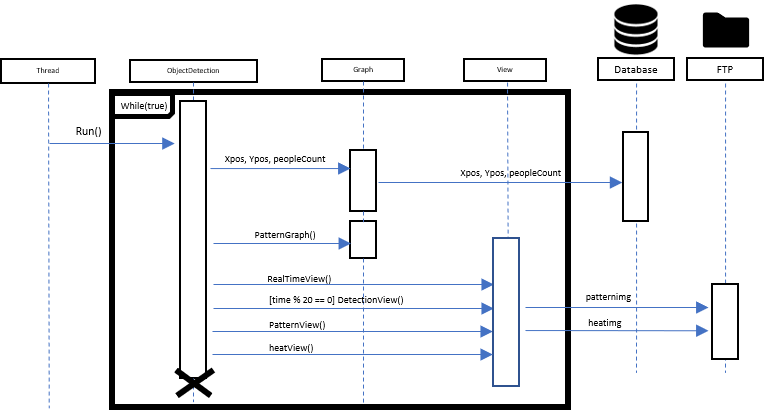
Data flow diagram



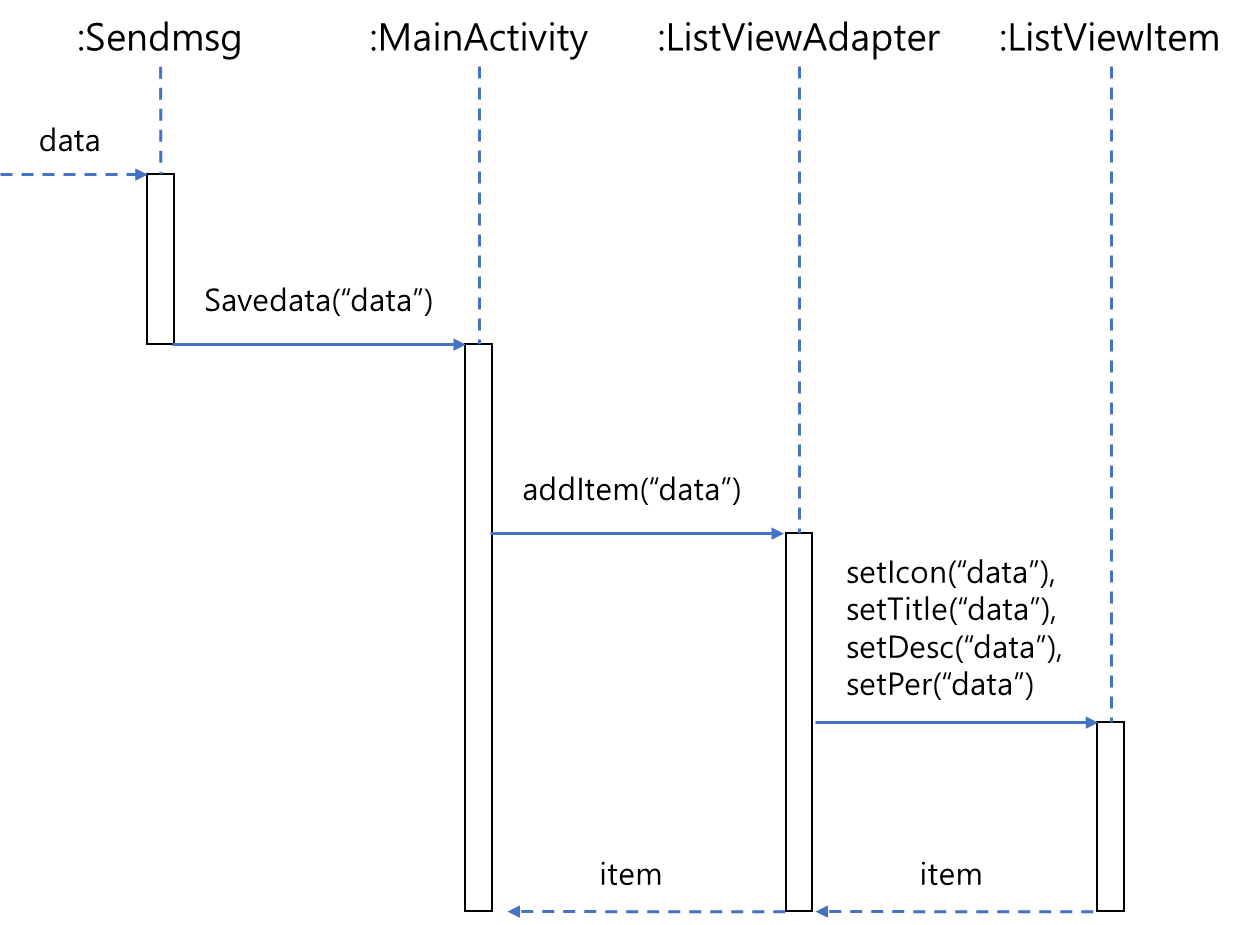
data request diagram



Scene analysis diagram

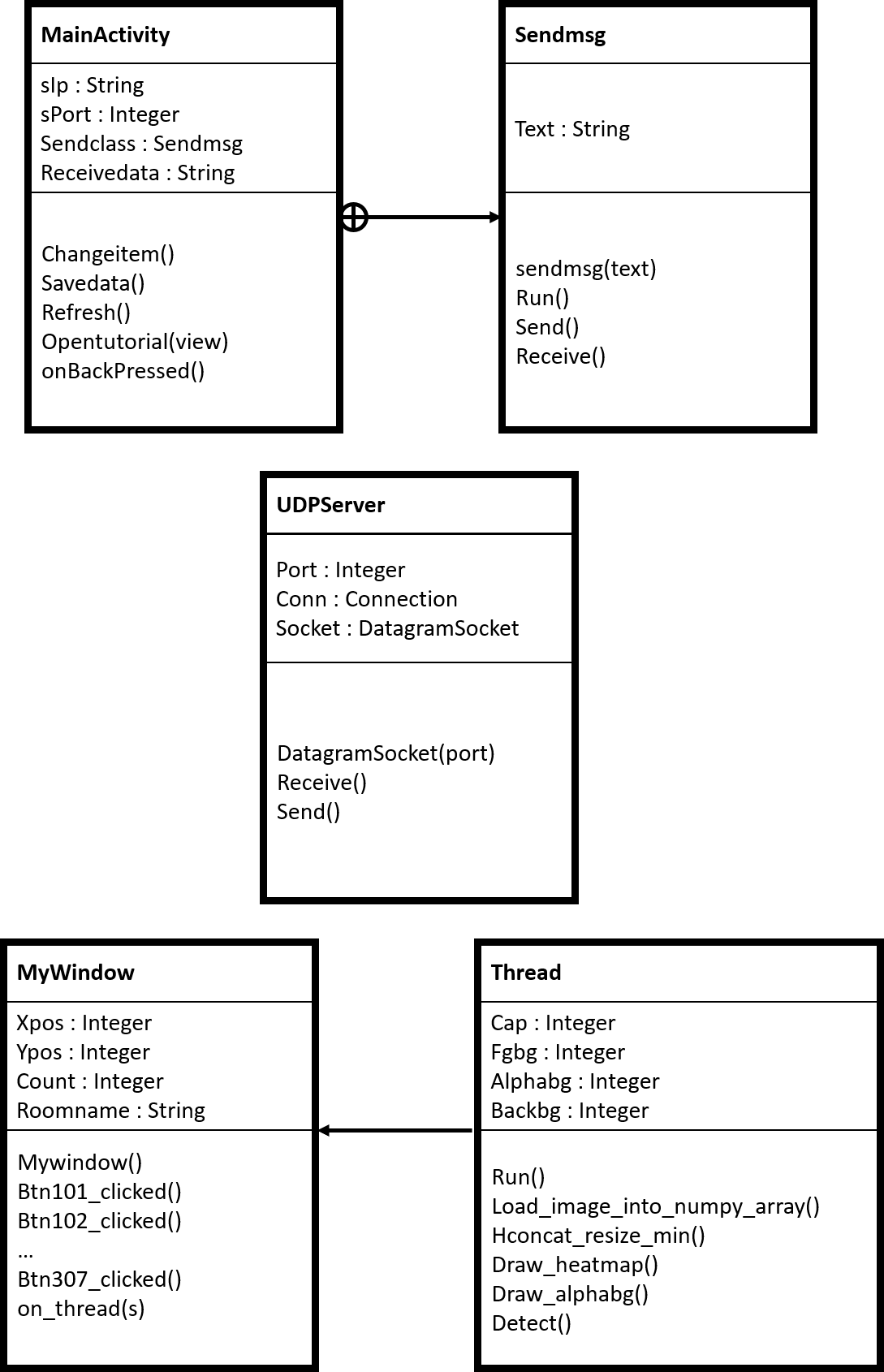


Server System diagram



Mobile data setting diagram

## Class Diagram

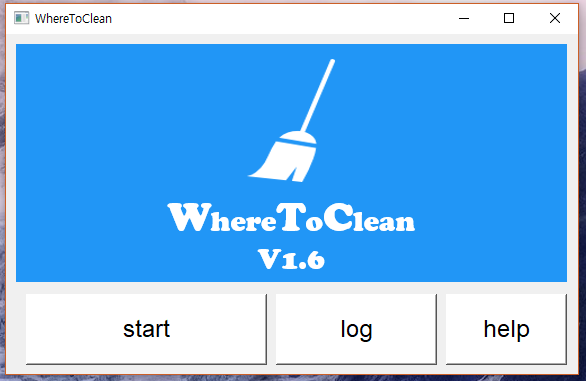


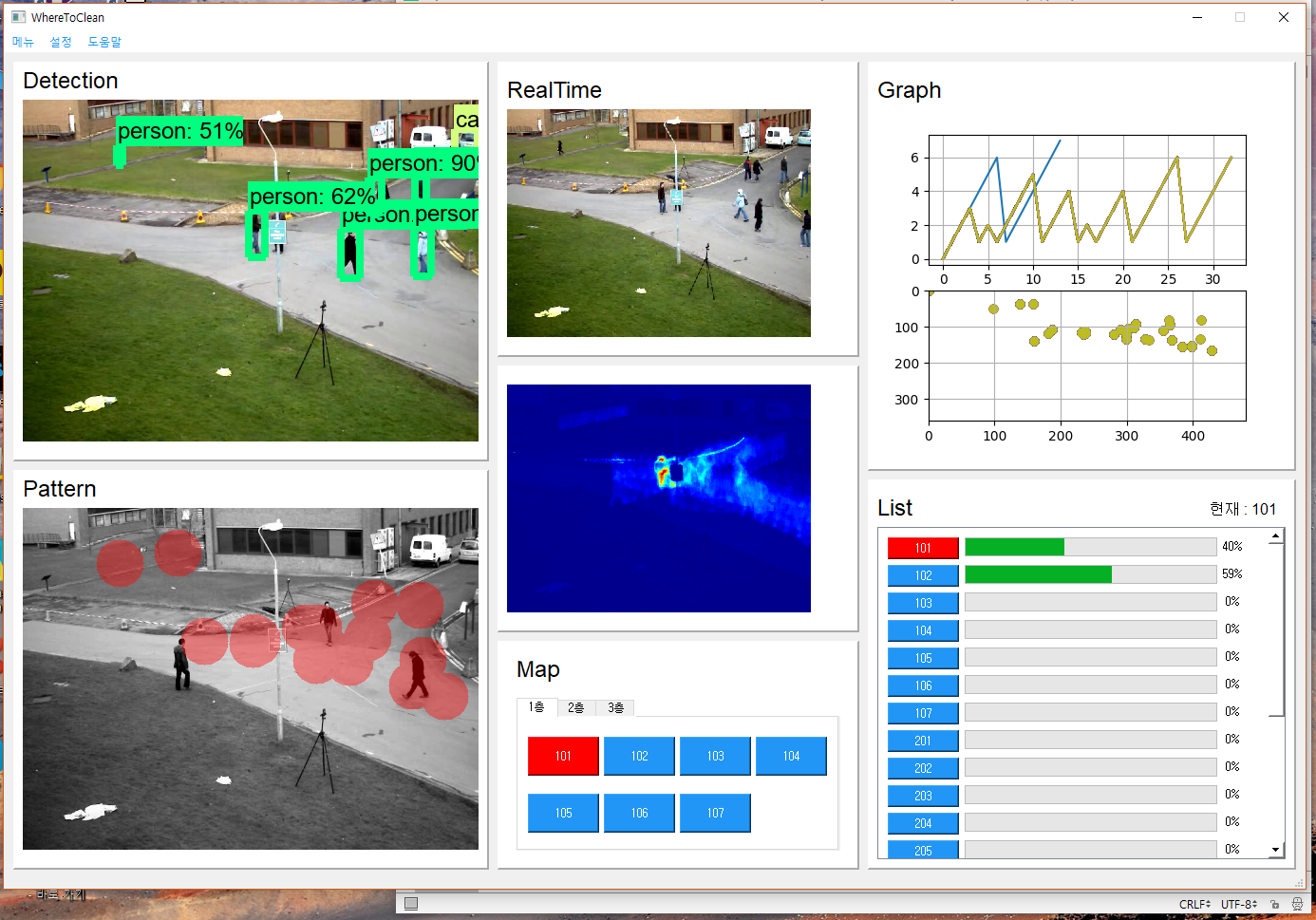


## .6소프트웨어 결과

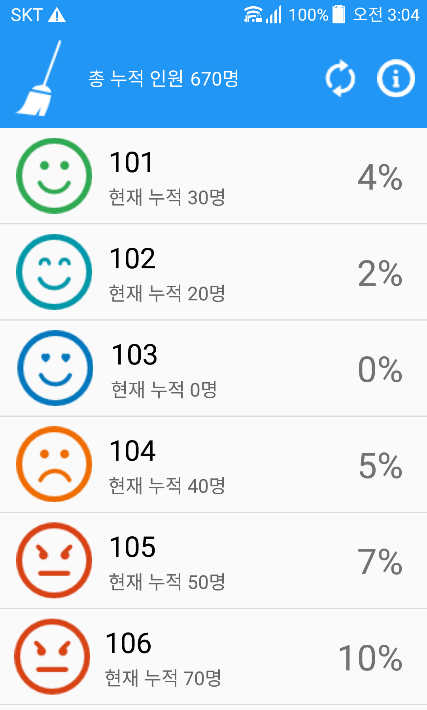
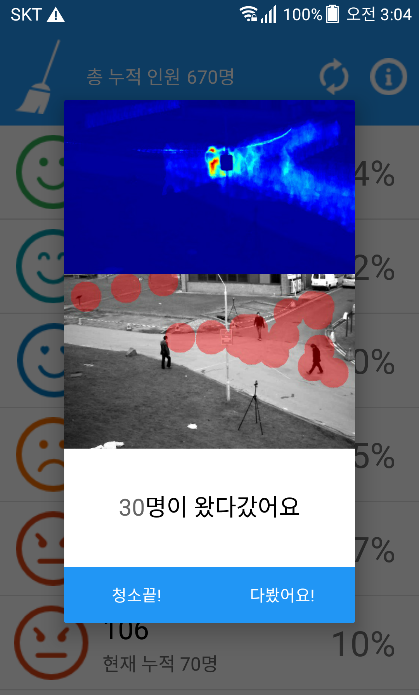
## 소프트웨어 결과 캡처화면

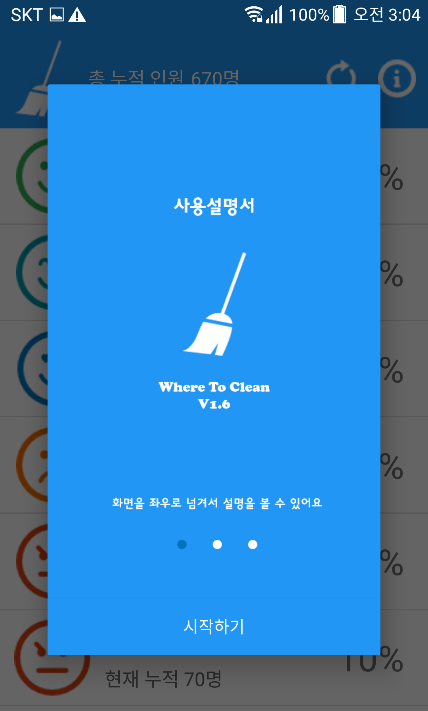
**서버**





**모바일**

**  **

** **

**Program Source Code**

1. **Server**

|  |  |  |
| --- | --- | --- |
|  | import java.net.DatagramPacket;  import java.net.DatagramSocket;  import java.sql.Connection;  import java.sql.Date;  import java.sql.DriverManager;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.text.SimpleDateFormat;    public class GasServer  {        public static void main(String[] args)      {            // DB관련객체          Connection conn = null;          PreparedStatement pstmt = null;          ResultSet rs = null;          // 포트번호          int port = 8888;          // 데이터그램소켓객체          DatagramSocket socket = null;          // //현재날짜          // Date date = new Date(System.currentTimeMillis());          // SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");          // String date2 = sdf.format(date);          // String realdate = date2+"%";            try          {              String jdbcUrl = "jdbc:mysql://localhost:3306/testdb";              String jdbcId = "root";              String jdbcPw = "rootpass";              Class.forName("com.mysql.jdbc.Driver");              conn = DriverManager.getConnection(jdbcUrl, jdbcId, jdbcPw);                System.out.println("접속 대기상태입니다.");              socket = new DatagramSocket(port);                while (true)              {                  // 데이터 받을 버퍼                  byte[] buffer = new byte[1024];                  // 받을 패킷, 수신대기                  DatagramPacket receivePacket = new DatagramPacket(buffer, buffer.length);                  socket.receive(receivePacket);                  // 수신확인                  String msg = new String(receivePacket.getData(), 0, receivePacket.getLength());                  System.out.println("받은 데이터 : " + msg);                    if (msg.equals("showdata"))                  {                      String sendmsg = "";                      String Query1 = "SELECT roomname,sum(people) FROM testtable group by roomname";                      pstmt = conn.prepareStatement(Query1);                      rs = pstmt.executeQuery();                        while (rs.next())                      {                          sendmsg += rs.getString(2) + ",";                      }                        byte[] sendbuffer = sendmsg.getBytes();                      // System.out.println(sendmsg);                      // System.out.println(sendbuffer.length);                      DatagramPacket sendall = new DatagramPacket(sendbuffer, sendbuffer.length,                              receivePacket.getAddress(), receivePacket.getPort());                      socket.send(sendall);                      pstmt.close();                      rs.close();                  } else                  {                      String Query3 = "DELETE FROM testtable WHERE roomname=" + msg;                      pstmt = conn.prepareStatement(Query3);                      pstmt.executeUpdate();                      String Query4 = "INSERT INTO testtable (roomname, people) VALUES (" + msg + ",0)";                      pstmt = conn.prepareStatement(Query4);                      pstmt.executeUpdate();                      pstmt.close();                  }              }          } catch (Exception e)          {              System.out.println(e);          } finally          {              if (socket != null)                  socket.close();          }        }    }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

1. **ObjectDetectionSys.py**

|  |  |
| --- | --- |
| import random    from matplotlib.backends.backend\_qt5agg import FigureCanvasQTAgg  from matplotlib.figure import Figure    import sys  from os import path  # import matplotlib; matplotlib.use('Agg')  # pylint: disable=multiple-statements    import cv2  import numpy as np  from matplotlib.backends.backend\_qt5agg import FigureCanvasQTAgg as FigureCanvas  from matplotlib.figure import Figure  from PyQt5.QtCore import \*  from PyQt5.QtWidgets import \*  from PyQt5.QtGui import \*  import numpy as np  import os  import six.moves.urllib as urllib  import sys  import tarfile  import tensorflow as tf  import zipfile  import time  import copy  import random  import pymysql  from PyQt5 import uic  from collections import defaultdict  from io import StringIO  from matplotlib import pyplot as plt  from PIL import Image  import math  from matplotlib.backends.backend\_qt5agg import FigureCanvasQTAgg  from matplotlib.figure import Figure    from PyQt5 import QtCore #conda install pyqt  from PyQt5 import QtWidgets  import ftplib  import time      sys.path.append("..")    from utils import label\_map\_util    from utils import visualization\_utils as vis\_util    imgwidth = 480  imgheight = 360  yvalue = 0  xvalue = 0  frame = [0]  xpos = [[0]\*1 for i in range(2)]  ypos = [[0]\*1 for i in range(2)]  count = [[0]\*1 for i in range(2)]  countsum = [0,0,0,0,0,0,0,0,0,0,0,0,0]  currentroomnum = 0 #현재방번호 0: 101 1 : 102  roomname = '101' #방번호  fsize = 400  line1 = 0  sf2 = 0  startflag = 0 #시작변수    cap = 0  f = 0  res =0  fgbg =0  kernel = 0  \_ = 0  alphabg = 0  backbg =0  bg =0    class Login(QWidget):      def \_\_init\_\_(self, parent):          QWidget.\_\_init\_\_(self, parent)          self.ui = uic.loadUi("WTC-Login.ui")          self.ui.show()          self.ui.btn1.clicked.connect(self.btn1\_clicked)        def btn1\_clicked(self):          QMessageBox.about(self, "잠시만 기다려주세요.", "로그인 되었습니다.")          global startflag          startflag = 1          self.start = MyWindow(self)          self.start.show()          self.start.move(50,50)    form\_class = uic.loadUiType("WTC-Main.ui")[0]    class Thread(QtCore.QThread):      personcount = QtCore.pyqtSignal(list)      changePixmap1 = pyqtSignal(QPixmap)      changePixmap2 = pyqtSignal(QPixmap)      changePixmap3 = pyqtSignal(QPixmap)      changePixmap4 = pyqtSignal(QPixmap)        def \_\_init\_\_(self, parent=None):          super(Thread, self).\_\_init\_\_(parent)        def run(self):          # What model to download.          MODEL\_NAME = 'ssd\_mobilenet\_v1\_ppn\_shared\_box\_predictor\_300x300\_coco14\_sync\_2018\_07\_03'          MODEL\_FILE = MODEL\_NAME + '.tar.gz'          DOWNLOAD\_BASE = 'http://download.tensorflow.org/models/object\_detection/'            # Path to frozen detection graph. This is the actual model that is used for the object detection.          PATH\_TO\_CKPT = MODEL\_NAME + '/frozen\_inference\_graph.pb'            # List of the strings that is used to add correct label for each box.          PATH\_TO\_LABELS = os.path.join('data', 'mscoco\_label\_map.pbtxt')            NUM\_CLASSES = 90            # ## Download Model            # In[ ]:            #ftp서버          ftp = ftplib.FTP("112.175.184.69")          ftp.login("surk0130", "rootpass123123")  # login (ID, Password)          ftp.cwd("/html/image/")  # 파일 전송할 Ftp 주소 (받을 주소)          os.chdir(r"C:\Users\hwi\Anaconda3\envs\tensorflow\models\object\_detection")  # 파일 전송 대상의 주소(보내는 주소)            opener = urllib.request.URLopener()          opener.retrieve(DOWNLOAD\_BASE + MODEL\_FILE, MODEL\_FILE)          tar\_file = tarfile.open(MODEL\_FILE)          for file in tar\_file.getmembers():              file\_name = os.path.basename(file.name)              if 'frozen\_inference\_graph.pb' in file\_name:                  tar\_file.extract(file, os.getcwd())            # ## Load a (frozen) Tensorflow model into memory.            # In[ ]:              detection\_graph = tf.Graph()          with detection\_graph.as\_default():              od\_graph\_def = tf.GraphDef()              with tf.gfile.GFile(PATH\_TO\_CKPT, 'rb') as fid:                  serialized\_graph = fid.read()                  od\_graph\_def.ParseFromString(serialized\_graph)                  tf.import\_graph\_def(od\_graph\_def, name='')            # ## Loading label map          # Label maps map indices to category names, so that when our convolution network predicts `5`, we know that this corresponds to `airplane`.  Here we use internal utility functions, but anything that returns a dictionary mapping integers to appropriate string labels would be fine            # In[ ]:              label\_map = label\_map\_util.load\_labelmap(PATH\_TO\_LABELS)          categories = label\_map\_util.convert\_label\_map\_to\_categories(label\_map, max\_num\_classes=NUM\_CLASSES,                                                                      use\_display\_name=True)          category\_index = label\_map\_util.create\_category\_index(categories)            # ## Helper code            # In[ ]:              def load\_image\_into\_numpy\_array(image):              (im\_width, im\_height) = image.size              return np.array(image.getdata()).reshape(                  (im\_height, im\_width, 3)).astype(np.uint8)            def hconcat\_resize\_min(im\_list, interpolation=cv2.INTER\_CUBIC):              h\_min = min(im.shape[0] for im in im\_list)              im\_list\_resize = [                  cv2.resize(im, (int(im.shape[1] \* h\_min / im.shape[0]), h\_min), interpolation=interpolation)                  for im in im\_list]              return cv2.hconcat(im\_list\_resize)            # For the sake of simplicity we will use only 2 images:          # image1.jpg          # image2.jpg          # If you want to test the code with your images, just add path to the images to the TEST\_IMAGE\_PATHS.          PATH\_TO\_TEST\_IMAGES\_DIR = 'test\_images'          TEST\_IMAGE\_PATHS = [os.path.join(PATH\_TO\_TEST\_IMAGES\_DIR, 'img{}.jpg'.format(i)) for i in range(1, 3)]            # Size, in inches, of the output images.          IMAGE\_SIZE = (12, 8)            # ---------------------------------------------- matplot          # 그래프 그리기          # 그래프 값          framesize = 0            # ---------------------- excel,mysql          #MySQL Connection 연결          conn = pymysql.connect(host='localhost', user='root', password='rootpass',                                db='testdb', charset='utf8')            #Connection 으로부터 Cursor 생성          curs = conn.cursor()          sql = "insert into testtable(roomname,people,x,y) values (%s,%s,%s,%s)"            # ----------------------- opencv          filename = '101'          global roomname          global cap          cap = cv2.VideoCapture(roomname+'.avi')  # 'experiment/'+filename+'.avi'          fourcc = cv2.VideoWriter\_fourcc(\*'XVID')          out = cv2.VideoWriter(filename + '\_out\_rcnninception.avi', fourcc, 20.0, (imgwidth, imgheight))            countsecond = 1  # 초세기          personcount = 0  # 사람수            # detection          Detectflag = True  # 디텍션 이프문 처음에만 들어가기            # 영상 블러처리 -히트맵          start = 1          duration = 10          fps = '30'            global f          global res          global fgbg          global kernel          global \_          global bg          #hitmap          \_, f = cap.read()          bg = copy.deepcopy(f)          f = cv2.cvtColor(f, cv2.COLOR\_BGR2GRAY)          f = cv2.GaussianBlur(f, (11, 11), 2, 2)          res = 0.05 \* f          res = res.astype(np.float64)            fgbg = cv2.createBackgroundSubtractorMOG2(history=1, varThreshold=100,                                                    detectShadows=True)            kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (13, 13))            global backbg          bg = cv2.resize(bg, (imgwidth, imgheight))          bg = cv2.cvtColor(bg, cv2.COLOR\_BGR2GRAY)          bg = copy.deepcopy(cv2.cvtColor(bg, cv2.COLOR\_GRAY2RGB))  # 흑백으로 바꿨다 컬러로          backbg = copy.deepcopy(bg)            threadtime = 0.5 #스레드시간          threadtimesum = 0 #스레드 시간 합          with detection\_graph.as\_default():              with tf.Session(graph=detection\_graph) as sess:                  while True:                      self.sleep(threadtime)                      threadtimesum += threadtime                      print(threadtimesum)                      ret, image\_np = cap.read()                      image\_np3 = copy.deepcopy(image\_np)  # 1번화면 히트맵에 복사                        # 히트맵 그리기                      fgmask = fgbg.apply(image\_np3, None, 0.01)                      gray = cv2.cvtColor(image\_np3, cv2.COLOR\_BGR2GRAY)                      gray = cv2.GaussianBlur(gray, (11, 11), 2, 2)                      gray = gray.astype(np.float64)                      fgmask = cv2.morphologyEx(fgmask, cv2.MORPH\_CLOSE, kernel)                      fgmask = fgmask.astype(np.float64)                      res += (40 \* fgmask + gray) \* 0.01                      res\_show = res / res.max()                      res\_show = np.floor(res\_show \* 255)                      res\_show = res\_show.astype(np.uint8)                      res\_show = cv2.applyColorMap(res\_show, cv2.COLORMAP\_JET)                        image\_np = cv2.resize(image\_np, (imgwidth, imgheight))  # 사이즈 수정                        image\_np2 = copy.deepcopy(image\_np)  # 1번화면 2번에복사                        # 20초마다 디텍션하기                      if (Detectflag == True or (threadtimesum) % 30 == 0):                          Detectflag = False                          threadtimesum = 0                          print(threadtimesum)                          personcount = 0  # 사람카운트 초기화                            # the array based representation of the image will be used later in order to prepare the                          # result image with boxes and labels on it.                          # Expand dimensions since the model expects images to have shape: [1, None, None, 3]                          image\_np\_expanded = np.expand\_dims(image\_np2, axis=0)                          image\_tensor = detection\_graph.get\_tensor\_by\_name('image\_tensor:0')                          # Each box represents a part of the image where a particular object was detected.                          boxes = detection\_graph.get\_tensor\_by\_name('detection\_boxes:0')                          # Each score represent how level of confidence for each of the objects.                          # Score is shown on the result image, together with the class label.                          scores = detection\_graph.get\_tensor\_by\_name('detection\_scores:0')                          classes = detection\_graph.get\_tensor\_by\_name('detection\_classes:0')                          num\_detections = detection\_graph.get\_tensor\_by\_name('num\_detections:0')                          # Actual detection.                          (boxes, scores, classes, num\_detections) = sess.run(                              [boxes, scores, classes, num\_detections],                              feed\_dict={image\_tensor: image\_np\_expanded})                          # Visualization of the results of a detection.                          vis\_util.visualize\_boxes\_and\_labels\_on\_image\_array(                              image\_np2,                              np.squeeze(boxes),                              np.squeeze(classes).astype(np.int32),                              np.squeeze(scores),                              category\_index,                              use\_normalized\_coordinates=True,                              line\_thickness=8)                            threshold = 0.5  # 0.5이상만 보여주기                          global currentroomnum                          for index, value in enumerate(classes[0]):                              ymin = boxes[0][index][0] \* imgheight                              xmin = boxes[0][index][1] \* imgwidth                              ymax = boxes[0][index][2] \* imgheight                              xmax = boxes[0][index][3] \* imgwidth                              personclassname = (category\_index.get(value)).get('name')                              widthvalue = int((xmax - xmin) / 2)  # width 길이                              heightvalue = int((ymax - ymin) / 2)  # height 길이                              if scores[0, index] > threshold and personclassname == 'person':                                  personcount = personcount + 1                                  countsum[currentroomnum] += personcount                                  curs.execute(sql, (roomname,personcount,int(xmin),int(ymin)))  # 사람 수 저장함.                                  count[currentroomnum].append(personcount)                                  xpos[currentroomnum].append(xmin)                                  ypos[currentroomnum].append(ymin)                                  conn.commit()                                  bg = cv2.circle(bg, (int((xmax + xmin) / 2), int((ymax + ymin) / 2)), 25, (0, 0, 255),                                                  -1)  # 사람 좌표 표시                            convertimg = copy.deepcopy(image\_np2)  # objectdetection 적용된이미지                          #ftp 서버에 저장                          # now = time.localtime()                          # filename = "%02d-%02d-%02d-%02d-%02d.jpg" % (now.tm\_mon, now.tm\_mday, now.tm\_hour, now.tm\_min, now.tm\_sec)                            global alphabg                          alphabg = copy.deepcopy(bg)                          cv2.addWeighted(backbg, 0.6, alphabg, 1 - 0.6, 0, alphabg)  # 그린 것 반투명처리                            filename1 = roomname+"-1.jpg"                          filename2 = roomname+"-2.jpg"                          cv2.imwrite(filename1, res\_show)                          cv2.imwrite(filename2, alphabg)                          myfile = open(filename1, 'rb')  # Open( ~ ,'r') <= Text파일은 됨, Open( ~ ,'rb') <= 이미지파일 됨                          ftp.storbinary('STOR ' + filename1, myfile)                          myfile = open(filename2, 'rb')  # Open( ~ ,'r') <= Text파일은 됨, Open( ~ ,'rb') <= 이미지파일 됨                          ftp.storbinary('STOR ' + filename2, myfile)                        # cv2.imshow('히트맵그리기', res\_show)    #                    horizontalimg = hconcat\_resize\_min([image\_np, convertimg, res\_show])                        # fig.savefig("fall1\_out\_rcnninception.png") #matplot 이미지 저장                      # wb.save(filename+'\_out\_rcnninception.xlsx')  # 엑셀에 저장                      # cv2.imshow('object\_detection', numpy\_horizontal)                            cv2.resize(image\_np,(360,240))                      cv2.resize(res\_show,(360,240))                      rgbImage1 = cv2.cvtColor(image\_np, cv2.COLOR\_BGR2RGB)                      convertToQtFormat = QImage(rgbImage1.data, rgbImage1.shape[1], rgbImage1.shape[0],                                                 QImage.Format\_RGB888)                      convertToQtFormat = QPixmap.fromImage(convertToQtFormat)                      p1 = convertToQtFormat.scaled(360, 240, Qt.KeepAspectRatio)                        rgbImage2 = cv2.cvtColor(convertimg, cv2.COLOR\_BGR2RGB)                      convertToQtFormat = QImage(rgbImage2.data, rgbImage2.shape[1], rgbImage2.shape[0],                                                 QImage.Format\_RGB888)                      convertToQtFormat = QPixmap.fromImage(convertToQtFormat)                      p2 = convertToQtFormat.scaled(imgwidth, imgheight, Qt.KeepAspectRatio)                        rgbImage3 = cv2.cvtColor(res\_show, cv2.COLOR\_BGR2RGB)                      convertToQtFormat = QImage(rgbImage3.data, rgbImage3.shape[1], rgbImage3.shape[0],                                                 QImage.Format\_RGB888)                      convertToQtFormat = QPixmap.fromImage(convertToQtFormat)                      p3 = convertToQtFormat.scaled(360, 240, Qt.KeepAspectRatio)                        rgbImage4 = cv2.cvtColor(alphabg, cv2.COLOR\_BGR2RGB)                      convertToQtFormat = QImage(rgbImage4.data, rgbImage4.shape[1], rgbImage4.shape[0],                                                 QImage.Format\_RGB888)                      convertToQtFormat = QPixmap.fromImage(convertToQtFormat)                      p4 = convertToQtFormat.scaled(imgwidth, imgheight, Qt.KeepAspectRatio)                        self.changePixmap1.emit(p1)                      self.changePixmap2.emit(p2)                      self.changePixmap3.emit(p3)                      self.changePixmap4.emit(p4)                      self.personcount.emit(count[currentroomnum])    class MyWindow(QMainWindow,form\_class):      def \_\_init\_\_(self, parent=None):          super(MyWindow, self).\_\_init\_\_(parent)            if startflag == 0:              self.login = Login(self)          else:              self.mywindow()        def mywindow(self):          self.setupUi(self)            self.label1.resize(imgwidth, imgheight)            self.figure = Figure()          self.canvas = FigureCanvasQTAgg(self.figure)            self.axis1 = self.figure.add\_subplot(211)          self.axis1.grid(True)          self.axis1.set\_ylabel("count")            self.sf2 = self.figure.add\_subplot(212)          self.sf2.set\_ylabel("location")          self.sf2.set\_xlim(0, imgwidth)  # 그래프 처음끝값          self.sf2.set\_ylim(imgheight, 0)          self.sf2.grid(True)            self.plotlayout.addWidget(self.canvas)            # 디텍션 스레드          self.th = Thread(parent=self)          self.th.changePixmap1.connect(self.label1.setPixmap)          self.th.changePixmap2.connect(self.label2.setPixmap)          self.th.changePixmap3.connect(self.label3.setPixmap)          self.th.changePixmap4.connect(self.label4.setPixmap)          self.th.personcount.connect(self.on\_threadSample\_newSample)          self.th.start()            self.btn101.clicked.connect(self.btn101\_clicked)          self.btn102.clicked.connect(self.btn102\_clicked)          self.btn103.clicked.connect(self.btn101\_clicked)          self.btn104.clicked.connect(self.btn101\_clicked)          self.btn105.clicked.connect(self.btn101\_clicked)          self.btn106.clicked.connect(self.btn101\_clicked)          self.btn107.clicked.connect(self.btn101\_clicked)          self.btn201.clicked.connect(self.btn101\_clicked)          self.btn202.clicked.connect(self.btn101\_clicked)          self.btn203.clicked.connect(self.btn101\_clicked)          self.btn204.clicked.connect(self.btn101\_clicked)          self.btn205.clicked.connect(self.btn101\_clicked)          self.btn206.clicked.connect(self.btn101\_clicked)        def btn101\_clicked(self):          global cap          global roomname          roomname = '101'          cap =  cv2.VideoCapture('101.avi')          print("버튼101")            # hitmap          global f          global res          global fgbg          global kernel          global \_          global bg          \_, f = cap.read()          bg = copy.deepcopy(f)          f = cv2.cvtColor(f, cv2.COLOR\_BGR2GRAY)          f = cv2.GaussianBlur(f, (11, 11), 2, 2)          res = 0.05 \* f          res = res.astype(np.float64)            fgbg = cv2.createBackgroundSubtractorMOG2(history=1, varThreshold=100,                                                    detectShadows=True)            kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (13, 13))            # alpha          global backbg          bg = cv2.resize(bg, (imgwidth, imgheight))          bg = cv2.cvtColor(bg, cv2.COLOR\_BGR2GRAY)          bg = copy.deepcopy(cv2.cvtColor(bg, cv2.COLOR\_GRAY2RGB))  # 흑백으로 바꿨다 컬러로          backbg = copy.deepcopy(bg)            self.btn101.setStyleSheet("background-color:rgb(255,0,0); color:rgb(255,255,255)")          self.btn102.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.btn103.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.btn104.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.btn105.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.btn106.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.btn107.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.btn201.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.btn202.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.btn203.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.btn204.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.btn205.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.btn206.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.listbtn101.setStyleSheet("background-color:rgb(255,0,0); color:rgb(255,255,255)")          self.listbtn102.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.listbtn103.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.listbtn104.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.listbtn105.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.listbtn106.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.listbtn107.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.listbtn201.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.listbtn202.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.listbtn203.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.listbtn204.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.listbtn205.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.listbtn206.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.currentvalue.setText("현재 : 101")            global currentroomnum          currentroomnum = 0          self.axis1.cla()          self.axis1.grid(True)          self.sf2.cla()  # 그래프리셋          self.sf2.grid(True)          self.sf2.set\_xlim(0, imgwidth)  # 그래프 처음끝값          self.sf2.set\_ylim(imgheight, 0)            totalsum = 0          global countsum          for i in range(len(countsum)):              totalsum += countsum[i]          self.probar101.setValue(int(countsum[0] / totalsum \* 100))          self.probar102.setValue(int(countsum[1] / totalsum \* 100))          self.probar103.setValue(int(countsum[2] / totalsum \* 100))          self.probar104.setValue(int(countsum[3] / totalsum \* 100))          self.probar105.setValue(int(countsum[4] / totalsum \* 100))          self.probar106.setValue(int(countsum[5] / totalsum \* 100))          self.probar107.setValue(int(countsum[6] / totalsum \* 100))          self.probar201.setValue(int(countsum[7] / totalsum \* 100))          self.probar202.setValue(int(countsum[8] / totalsum \* 100))          self.probar203.setValue(int(countsum[9] / totalsum \* 100))          self.probar204.setValue(int(countsum[10] / totalsum \* 100))          self.probar205.setValue(int(countsum[11] / totalsum \* 100))          self.probar206.setValue(int(countsum[12] / totalsum \* 100))        def btn102\_clicked(self):          global cap          global roomname          roomname = '102'          cap =  cv2.VideoCapture('102.avi')          print("버튼102")            #hitmap          global f          global res          global fgbg          global kernel          global \_          global bg          \_, f = cap.read()          bg=copy.deepcopy(f)          f = cv2.cvtColor(f, cv2.COLOR\_BGR2GRAY)          f = cv2.GaussianBlur(f, (11, 11), 2, 2)          res = 0.05 \* f          res = res.astype(np.float64)            fgbg = cv2.createBackgroundSubtractorMOG2(history=1, varThreshold=100,                                                    detectShadows=True)            kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (13, 13))            #alpha          global backbg          bg = cv2.resize(bg, (imgwidth, imgheight))          bg = cv2.cvtColor(bg, cv2.COLOR\_BGR2GRAY)          bg = copy.deepcopy(cv2.cvtColor(bg, cv2.COLOR\_GRAY2RGB))  # 흑백으로 바꿨다 컬러로          backbg = copy.deepcopy(bg)            self.btn101.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.btn102.setStyleSheet("background-color:rgb(255,0,0); color:rgb(255,255,255)")          # self.btn103.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          # self.btn104.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          # self.btn105.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          # self.btn106.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          # self.btn107.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          # self.btn201.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          # self.btn202.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          # self.btn203.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          # self.btn204.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          # self.btn205.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          # self.btn206.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.listbtn101.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.listbtn102.setStyleSheet("background-color:rgb(255,0,0); color:rgb(255,255,255)")          # self.listbtn103.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          # self.listbtn104.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          # self.listbtn105.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          # self.listbtn106.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          # self.listbtn107.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          # self.listbtn201.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          # self.listbtn202.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          # self.listbtn203.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          # self.listbtn204.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          # self.listbtn205.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          # self.listbtn206.setStyleSheet("background-color:rgb(33,150,246); color:rgb(255,255,255)")          self.currentvalue.setText("현재 : 102")            global currentroomnum          currentroomnum = 1          self.axis1.cla()          self.axis1.grid(True)          self.sf2.cla()  # 그래프리셋          self.sf2.grid(True)          self.sf2.set\_xlim(0, imgwidth)  # 그래프 처음끝값          self.sf2.set\_ylim(imgheight, 0)            totalsum = 0          global countsum          for i in range(len(countsum)):              totalsum += countsum[i]          self.probar101.setValue(int(countsum[0] / totalsum\*100))          self.probar102.setValue(int(countsum[1] / totalsum\*100))          self.probar103.setValue(int(countsum[2] / totalsum \* 100))          self.probar104.setValue(int(countsum[3] / totalsum \* 100))          self.probar105.setValue(int(countsum[4] / totalsum \* 100))          self.probar106.setValue(int(countsum[5] / totalsum \* 100))          self.probar107.setValue(int(countsum[6] / totalsum \* 100))          self.probar201.setValue(int(countsum[7] / totalsum \* 100))          self.probar202.setValue(int(countsum[8] / totalsum \* 100))          self.probar203.setValue(int(countsum[9] / totalsum \* 100))          self.probar204.setValue(int(countsum[10] / totalsum \* 100))          self.probar205.setValue(int(countsum[11] / totalsum \* 100))          self.probar206.setValue(int(countsum[12] / totalsum \* 100))          @QtCore.pyqtSlot(list)      def on\_threadSample\_newSample(self, sample):      #     self.sf.plot(sample)      #     #self.line1.set\_xdata(frame)             self.sf2.scatter(xpos[currentroomnum], ypos[currentroomnum])  # 위치 표시           self.axis1.plot(sample)           self.canvas.draw()    if \_\_name\_\_ == "\_\_main\_\_":      app = QApplication(sys.argv)      tt = MyWindow()      #tt.show()      app.exec\_()    [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |