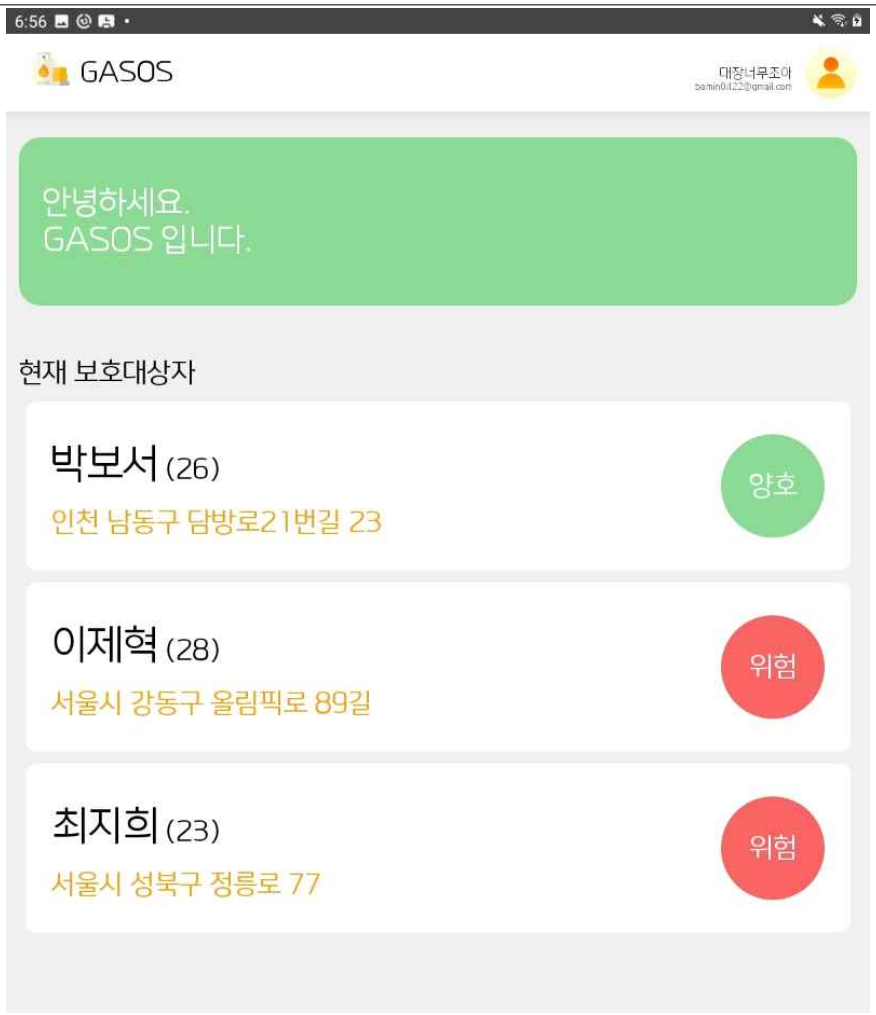
제19회 임베디드SW경진대회 개발완료보고서

[4차산업 기반 에너지]

□ 개발 요약

팀 명	KOBOTBOARD



작품명	GASOS
작품설명 (요약)	사회적 취약계층의 가정 내 일산화탄소 및 유독가스, 도시가스의 유출 여부를 보호자가 실시간으로 모바일 환경에서 확인할 수 있고, 신고까지 할 수 있는 통합 솔루션
소스코드	https://github.com/kmu-kobot/2021ESWContest_4-_6013
시연동영상	https://youtu.be/b6kvZ3J4KKo

□ 개발 개요

○ 개발 작품 개요

- 일정 거리 이상 떨어져 설치된 여러 감지 센서가 중앙 장치와 일괄적으로 통신한 후 DB 서버에 데이터를 송신하는 스타 토폴로지 형태의 무선 센서 네트워크를 구성함
- Google사의 Firebase를 통해 여러 대 중앙 장치와 하나의 모바일 이용자를 연결하는 1:N 통신을 구현하여 한 명의 관리자가 동시의 다수의 취약계층의 안전 상태를 관리·점검할 수 있도록 함
- 관리자는 모바일 앱 내에서 사용자의 실제 위치와 주소를 지도와 핀으로 쉽게 알 수 있고, 이상 상황 발생 시 스마트폰의 푸시 알림을 통해 빠르게 인지할 수 있으며 119에 신고하는 등의 즉각적인 대처가 가능함

○ 개발 배경 및 동기

- 사회적 취약계층의 가스 시설 안전 지원 서비스의 필요성 인식

[출처: https://news.seoul.go.kr/safe/archives/506425]

- 안전취약계층에 안전설비를 설치해주는 식의 사회적 움직임은 활발하지만, 설치 이후 직접 유지와 관리가 실질적으로 어려운 소외 계층이 발생할 수 있으므로 이에 대한 대처가 필요함

- 각각 적합한 설치 위치가 다른 여러 종류의 위험 물질 감지기들을 동시에 하나의 시스템 내에서 관리할 수 있는 통합 솔루션 개발
- 관리자가 원격으로 보호 대상자들의 안전 상태를 확인할 수 있는 모바일 환경 조성

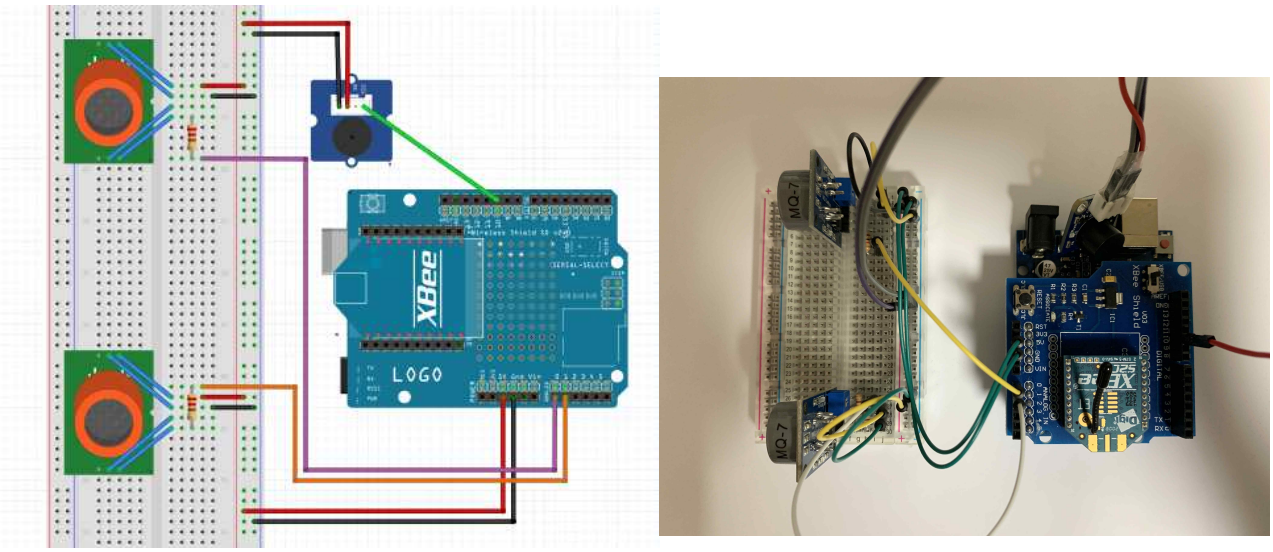
○ 개발 목표

- 가스 누출 감지기, CO 감지기와 Zigbee를 활용한 무선 센서 네트워크를 구성
- 센서의 측정, 상태 데이터를 DB에 실시간으로 저장하는 Data Logger의 구현
- 한 명의 보호자가 다수의 취약 계층을 통합 관리할 수 있는 모바일 플랫폼 제작

□ 개발 환경 설명

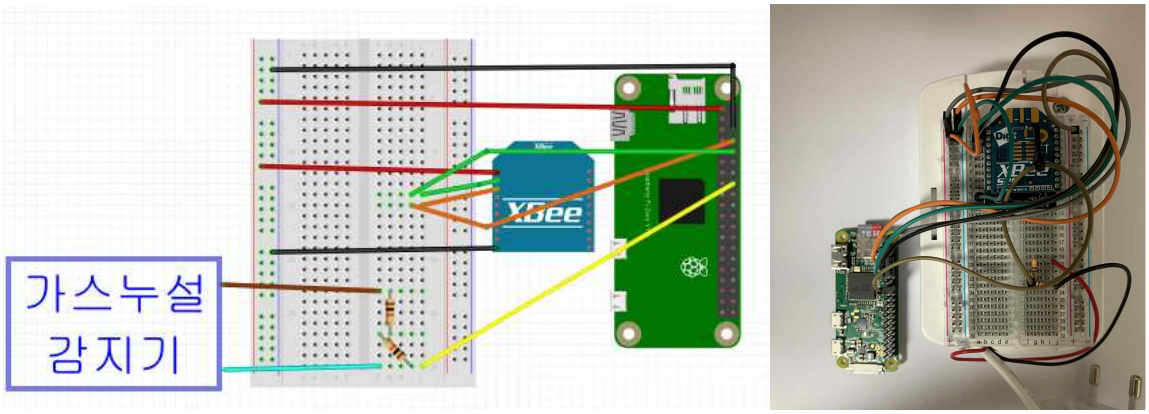
○ Hardware 구성

- CO 감지 모듈 구성: Arduino Uno, XBee Shield, XBee S2C, MQ-7(센서), Active Buzzer
- 2개의 가스 센서를 사용하여 정확도를 향상
- 아두이노의 ADC (Analog to Digital Converter) 기능을 활용하여 설계
- 센서값이 기준치를 넘을 경우 부저로 경보음을 내고, 중앙 장치로 경보 송신



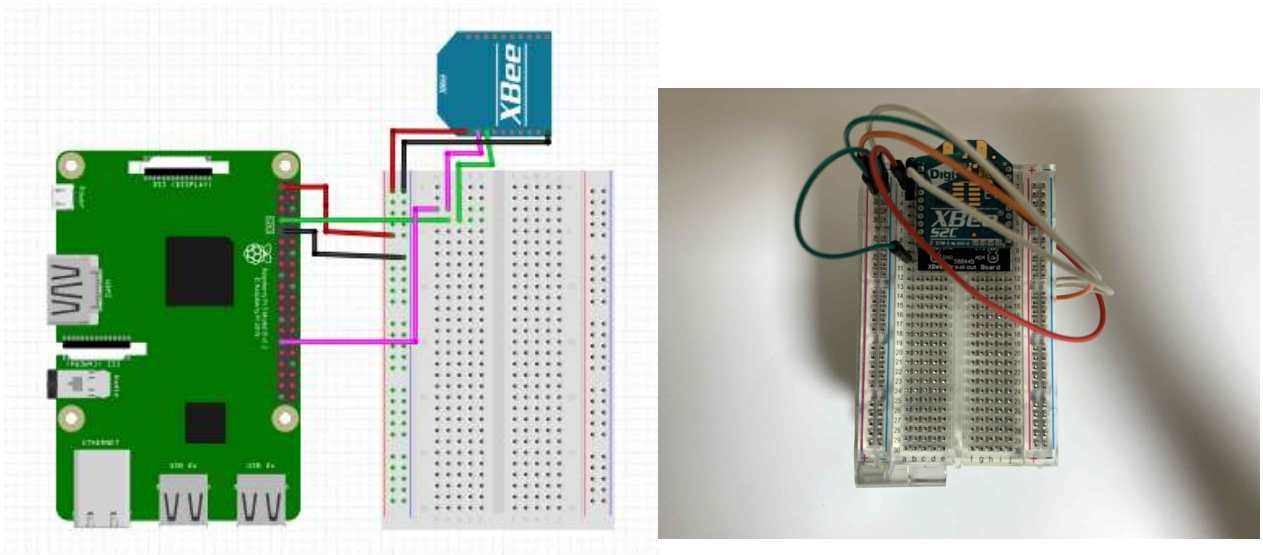
[회로도와 실제 회로 이미지 – CO 감지기]

- 가스누설 경보기 모듈: RaspberryPi Zero W, XBee S2C, 시제품 가스누설 경보기로 구성
- 시제품 가스누설 경보기가 경보 발생시 10V 아날로그 신호를 발생시키는 것에 비해, RaspberryPi는 ADC가 존재하지 않고 3.3V 기준전압 디지털 신호만을 입력받기에 보드 내부저항을 고려하여, 10KΩ 저항 두 개를 사용해 신호의 세기를 맞춰 입력받음
- RaspberryPi Zero와 XBee간 UART 통신 (GPIO 14, 15번 핀 사용, Serial Port=ttyAMA0)
- 누설 경보 발생 시 중앙 장치로 이상 상태 전달



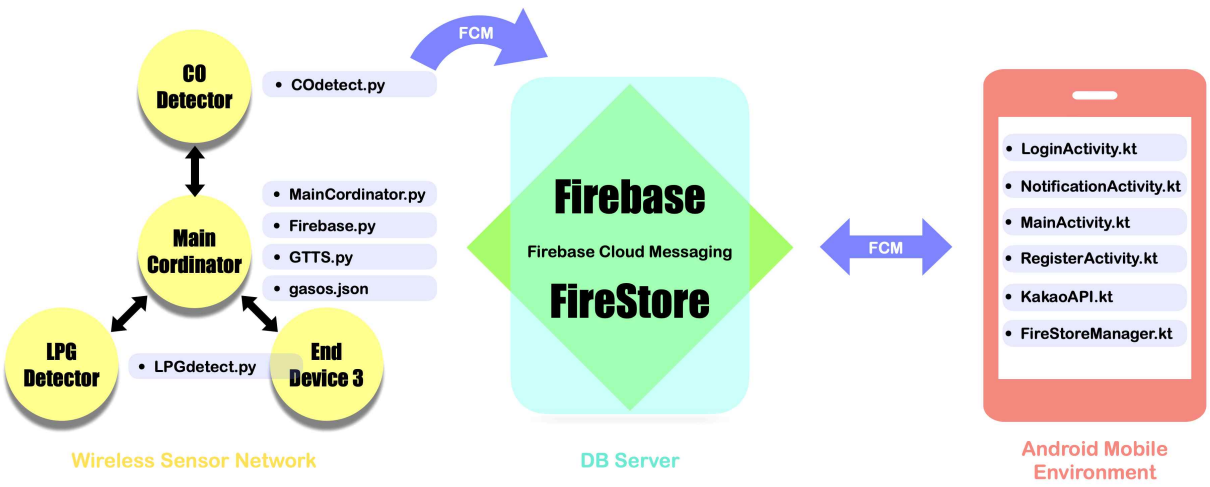
[회로도와 실제 회로 이미지 – 가스누설 경보기]

- 중앙 통제 장치: RaspberryPi 4, XBee S2C로 구성
- RaspberryPi와 XBee간 UART 통신 (GPIO 4, 5번 핀 사용, Serial Port=ttyAMA1)
- 스타 토폴로지 네트워크의 중심 역할을 하는 장치
- 각 엔드 디바이스들과 XBee를 통한 정보 송수신 가능
- 가스밸브 차단기(퓨즈콕 차단기)를 연결하려 하였으나, 신호처리에 문제가 발생하여 제외
- USB 스피커를 연결하는 것으로, TTS (Text To Speech)기능을 활용하여 경보 음성 출력

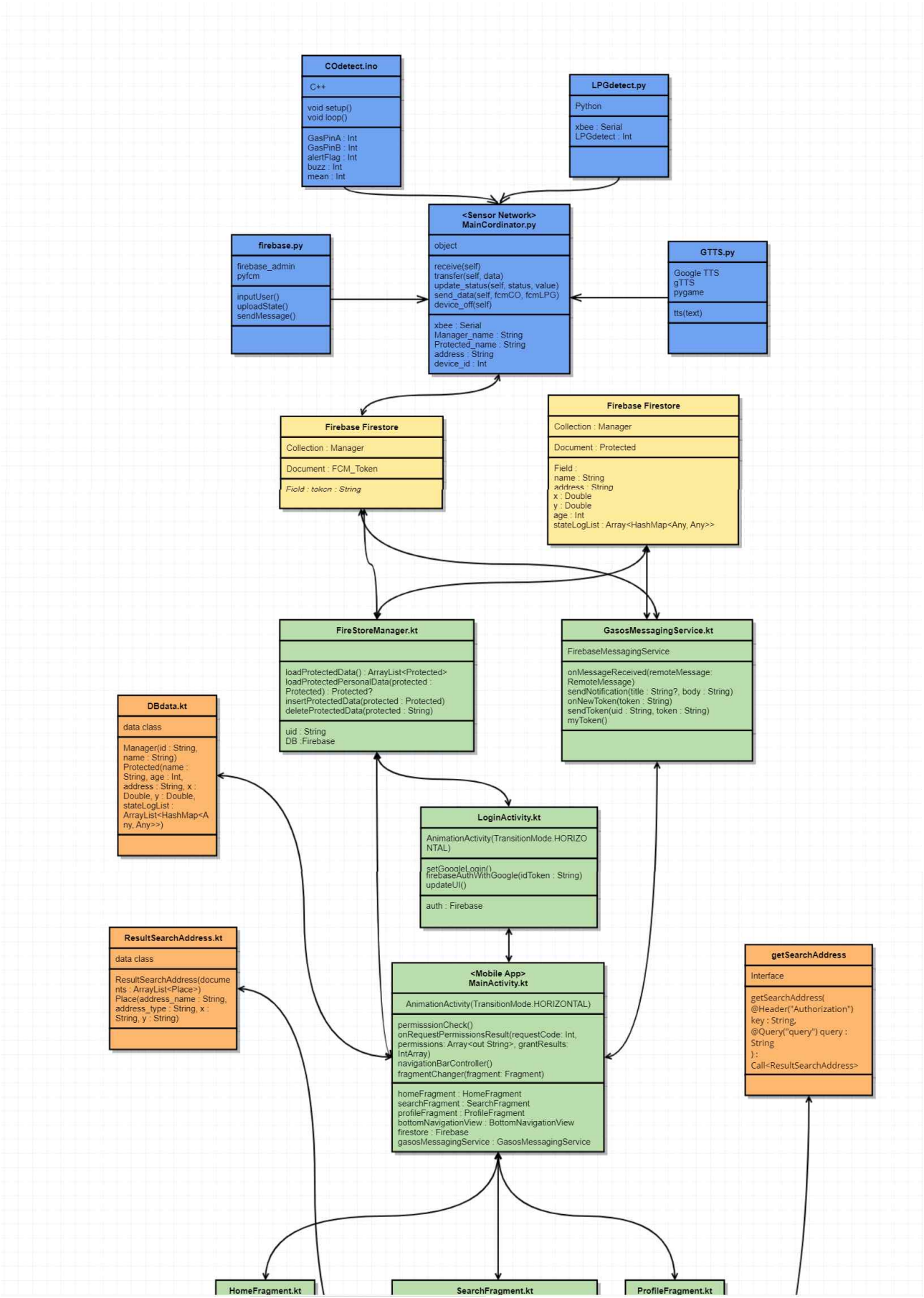


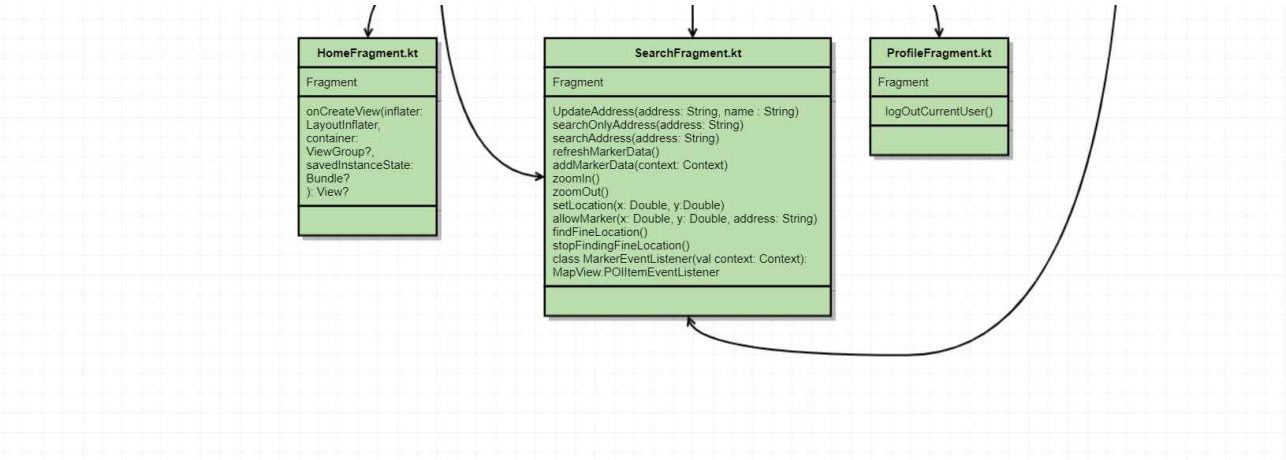
[회로도와 실제 회로 이미지 – 중앙 통제 장치]

○ Software 구성



○ Software 설계도



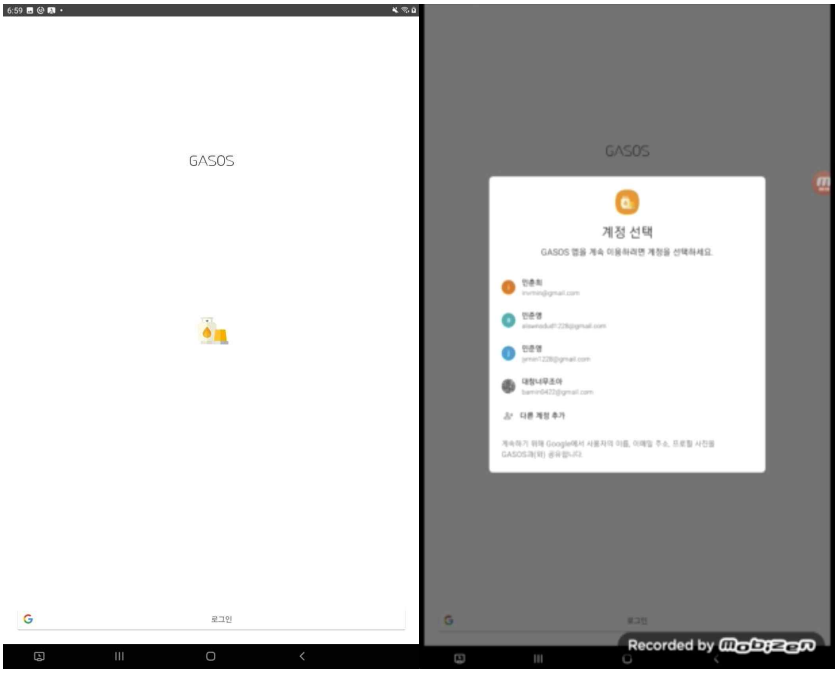


○ Software 기능 (필요 시 알고리즘 설명 포함)

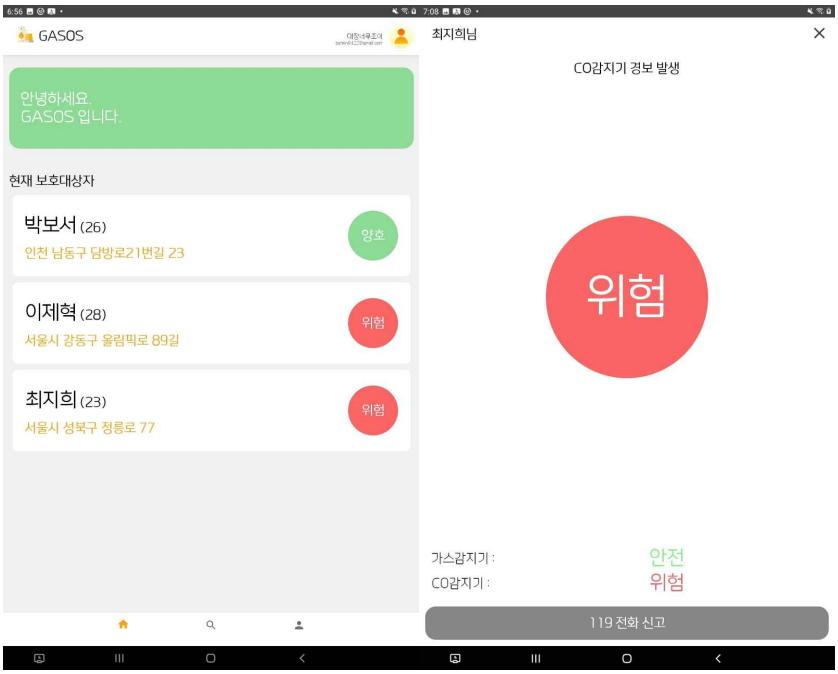
- CO 감지 모듈(COdetect.ino): 폴링 방식으로 두 개의 CO 감지 센서값을 0.15초마다 확인하여 평균을 내고, 수신 버퍼의 내용을 확인한다. 수신 버퍼의 내용이 CO 감지 모듈을 호출하는 내용이라면 자신의 현재 상태와 센서값의 평균 데이터를 중앙 장치로 송신한다.
- 가스누설 감지 모듈(LPGdetect.py): 폴링 방식으로 0.1초 간격을 두며 수신 버퍼와 가스누설 감지기의 출력을 확인한다. 수신 버퍼의 내용이 가스누설 감지 모듈을 호출하고 있다면 현재 가스누설 여부를 중앙 장치로 송신한다.
- 중앙 장치(MainCordinator.py): 폴링 방식으로 0.1초 간격을 두고 번갈아 가며 무선 네트워크상에서 CO 감지 모듈, 가스누설 감지 모듈을 호출하여 각각의 상태를 확인한다. 상태를 확인하고 수신한 데이터를 Firebase DB 서버에 송신하여 저장하고 이상 상황 발생 시 FCM을 사용해 DB서버가 모바일로 푸시 알림 명령을 보낼 수 있도록 하며, GTTS.py에서 import된 tts 함수를 통해 경보 음성을 스피커로 출력하는 기능을 한다.
- GTTS.py: python gTTS 라이브러리를 활용하여 음성으로 출력하고자 하는 경보 문구를 음성 데이터 파일로 작성하여 지정 경로상에 저장하는 역할을 한다.
- firebase.py: LPG, CO가 한번씩 loop를 돌아 log를 생성할 때마다 map형식으로 묶어서 firebase의 firestore 데이터베이스에 저장한다. 만약 log가 warning이라면 fcm(firebase cloud messaging)을 통해 모바일로 바로 푸쉬 알림을 보내도록 한다.

○ UI 사용법 (구성 및 설명)

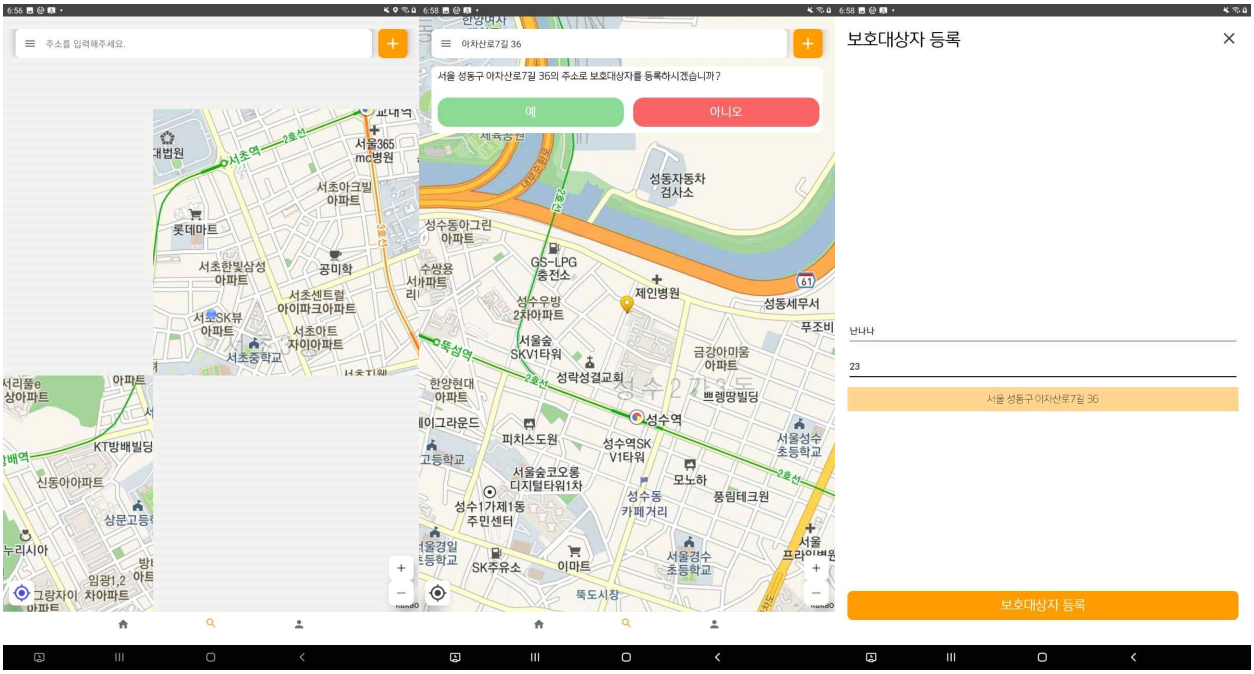
- 앱 설치 후 첫 실행 시 로그인 페이지
- 화면 하단의 로그인 버튼을 누르고 구글 계정을 연동하여 로그인을 진행함



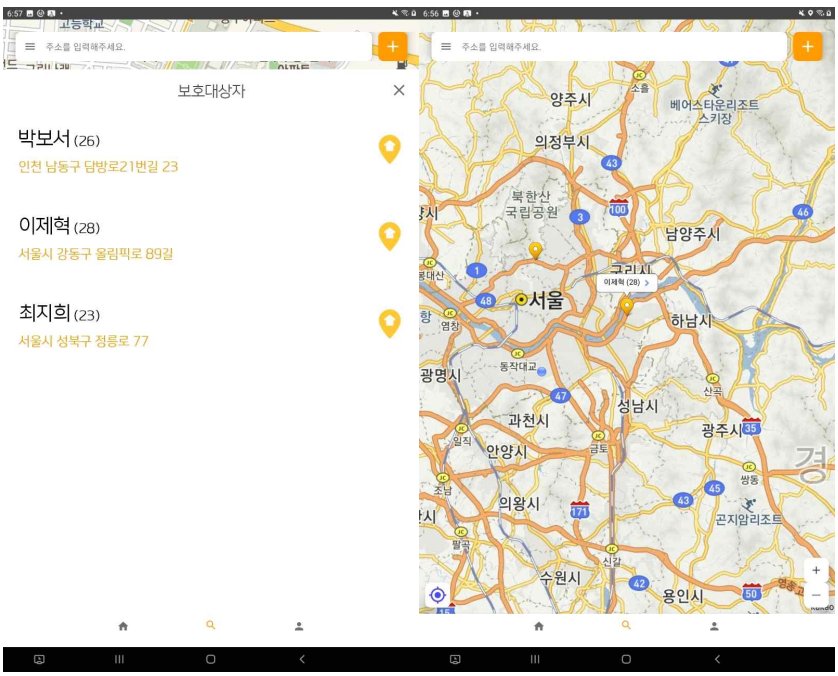
- 메인 페이지 (화면 하단의 좌측 홈 버튼)
- 현재 나의 보호 대상자 전체 목록을 볼 수 있으며 각 대상자 버튼을 누르면 우측 사진과 같이 상세 정보를 확인할 수 있음



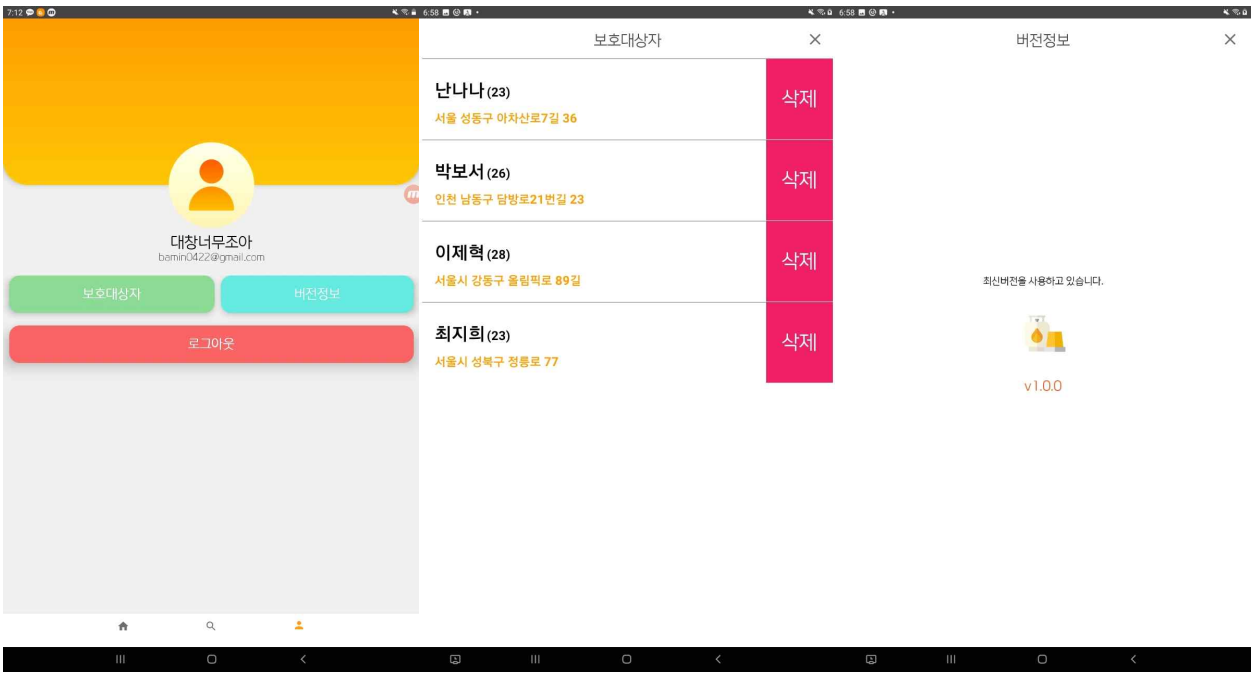
- 지도 페이지 (화면 하단의 중앙 돋보기 모양 버튼)
- 상단 주소창에 추가하고자 하는 보호 대상자의 주소를 입력하고 화면 상단 우측의 +버튼을 누르면 세 번째 사진과 같이 이름과 나이를 입력하는 팝업 윈도우가 나타나며, 입력을 마치고 보호대상자 등록 버튼을 누르면 등록이 완료됨



- 지도 페이지 주소창 좌측의 Hamburger Menu를 클릭하면 현재 등록된 나의 보호 대상자 전체가 팝업 윈도우로 나타나며, 각 대상자 우측의 노란색 핀 버튼을 누르면 해당 보호 대상자의 위치가 지도의 중심으로 설정됨



- 마이페이지 (화면 하단 우측의 사람 모양 버튼)
- 첫 번째 사진의 녹색 버튼을 누르면 현재 등록되어 있는 보호 대상자를 삭제할 수 있는 페이지로 이동할 수 있으며, 빨간색 삭제 버튼을 누르면 삭제할 수 있음. 또한 첫 번째 사진의 하늘색 버튼을 누르면 현재 사용중인 애플리케이션의 버전 정보를 확인할 수 있음



○ 개발환경 (언어, Tool, 사용시스템 등)

사용 요소	분류		
-	Sensor-Network	Mobile App	DB
운영 체제	Raspbian	Windows 10	Windows 10
사용 언어	C++ / Python3.6.9	Kotlin, Java	-
프레임워크/ 라이브러리	- Google TTS (Text To Speech) - RPI.GPIO - serial / PySerial - firebase_admin - pyfcm - pygame	- kotlinx-coroutines - firebase-firestore - circleImageView - retrofit2 - firebase-auth - kakao sdk - gson - firebase-mssaging	Firebase FireStore
DBMS	Firebase FireStore	Firebase FireStore	Firebase FireStore
데이터 통신	FCM (Firebase Cloud Messaging)	FCM (Firebase Cloud Messaging)	FCM (Firebase Cloud Messaging)
개발 도구	VS code Arduino IDE XCTU	Android Studio	Firebase Console
협업도구	Git Hub / Slack		

□ 개발 프로그램 설명

○ 파일 구성

Module	역할, 설명
센서 네트워크	
MainCordinator.py	센서 네트워크 전반을 관리하며 DB 서버와 통신한다.
firebase.py	firebase firestore에서 Manager ID와 Protected ID를 불러오고 데이터를 읽기, 쓰기한다.
GTTS.py	구글 TTS(Text To Speech) API를 사용해, 메인 모듈에 장치되어 있는 스피커에서 'OOO님 댁에서 CO/가스 누출이 감지되었습니다.'라는 음성을 출력한다.
HW_Device/COdetect.ino	CO감지기의 수치를 확인하고, MainCordinator와 Zigbee 통신으로 정보를 교환한다.
HW_Device/LPGdetect.py	가스 누출 감지를 확인하고, MainCordinator와 Zigbee 통신으로 정보를 교환한다.
모바일 앱	
activity/MainActivity.kt	Bottom Navigation Bar를 통해, Fragment(HomeFragment / SearchFragment / ProfileFragment)간 전환을 할 수 있도록 하는 메인 액티비티이다. 사용자에게 권한을 얻는 부분이다.
activity/LoginActivity.kt	사용자가 구글 로그인했을 때, Firestore의 Collection 값에 Manager_이름 즉 사용자 정보를 저장하는 부분이다.
activity/NotificationActivity.kt	RecyclerView에서 아이템을 클릭했을 때, 보호대상자의 위험 수치와 가스 및 CO 누출 여부를 확인하고 119 신고를 할 수 있도록 한다.
activity/AnimationActivity.kt	프래그먼트 및 액티비티 간 전환이 있을 때, 각각의 상황에 맞는 알맞은 애니메이션과 전환 효과를 주는 역할을 한다.
activity/MenuActivity.kt	ProfileFragment에서 보호 대상자를 삭제할 수 있도록 도와주는 Activity이다.
activity/RegisterAcitivity.kt	SearchFragment에서 보호 대상자를 추가할 때, 보호대상자의 정보를 입력하는 Form을 만들어 준다.
activity/SplashActivity.kt	GASOS 모바일 앱이 실행되었을 때, 처음으로 생성되는 인트로 화면이다.
activity/VersionActivity.kt	Version 정보가 기록되어 있고, 이를 알려주는 파일이다.
fragment/HomeFragment.kt	보호대상자들의 RecylerView인 MainRecyclerView를 보여주고, CO 및 가스 누출 여부를 한눈에 파악할 수 있도록 한다.
fragment/ProfileFragment.kt	사용자의 로그인 정보와 버전 정보, 보호 대상자 삭제, 로그아웃 등의 기능을 할 수 있는 Fragment이다.
fragment/Search	카카오맵 API를 사용해, 현 위치를 한눈에 볼 수 있고, 보호 대상자의 위

Fragment.kt	치를 확인할 수 있다. 또한, 주소 입력을 통해 새로운 보호 대상자를 등록할 수 있는 기능을 한다.
util/FireStoreManager.kt	클라이언트에서 FireStore로 접근할 때, 사용하는 파일이다. 데이터 읽기, 쓰기 등 Firestore에 관한 여러 가지 기능을 가지고 있다.
util/GasosMessagingService.kt	FCM(Firebase Cloud Messaging)을 사용하여, 센서 네트워크에서 송신한 push 알림을 수신하고 이에 맞는 event(앱 실행 및 MainActivity로 이동)를 수행한다.
util/GlobalApplication.kt	Kakao SDK를 사용하기 위한 App Key를 포함하고 있는 파일이다.
util/KakaoAPI.kt	Retrofit2를 사용해, Kakao Map API와 통신을 할 수 있는 인터페이스이다.
adaptor/MainRvAdapter.kt	MainRecyclerView의 아이템들을 적용하고, item click 이벤트를 지정하는 Adapter이다.
adaptor/MapRvAdapter.kt	SearchFragment에서 햄버거 메뉴 버튼을 클릭했을 때, 생성되는 RecyclerView의 아이템과 click이벤트를 지정하는 Adapter이다.
adaptor/SettingRvAdapter.kt	ProfileFragment에서 보호대상자 관리를 눌렀을 때, 생서되는 RecyclerView의 아이템과 Click 이벤트(보호대상자 삭제)를 하는 Adapter이다.
data/DBdata.kt	Firestore에 저장되는 Data Class를 정의하는 파일이다.
data/ResultSearchAddress.kt	Kakao Map에서 데이터 읽기를 할 때, 사용되는 Data Class를 정의하는 파일이다.

○ 함수별 기능

1. 센서 네트워크

1. HW_Device

COdetect.ino

수신 버퍼를 확인하여 본 모듈이 호출되었을 때, 일산화탄소 감지 센서에서 측정한 농도 값이 일정 수치 이상이면 경보를 포함한 내용을, 그렇지 않으면 농도 데이터만을 중앙 장치로 송신한다. (폴링 방식)

Class	Method	역할
-	setup()	Initialize Device
	loop()	센서를 통해 일산화탄소 농도를 측정하고 상태에 따라 네트워크 통신을 수행하는 반복문

LPGdetect.py

수신 버퍼를 확인하여 본 모듈이 호출되었을 때, 가스 누출 감지기에서 상태 값을 받아와 현재 상태의 위험 여부를 중앙 장치로 송신한다. (폴링 방식)

2. 메인 모듈

MainCordinator.py		
LPG, CO 모듈의 상태 및 측정값을 번갈아 가며 확인한 뒤, 데이터베이스 서버(Firebase Firestore)와 통신하여 데이터를 저장하고, Google TTS(Text To Speech)를 통한 음성 경보 알림을 수행		
Class	**Method**	**역할**
main_cordinator (object)	receive()	현재 수신 버퍼를 클리어한 뒤 센서 네트워크에서 스트링 한 줄을 수신한다.
	transfer(data)	입력 파라미터로 받은 데이터가 바이트 타입이라면 네트워크로 transmit하고, 스트링 형태라면 줄바꿈 문자를 정규표현식 라이브러리를 활용해 제거한 뒤 바이트로 변환하여 네트워크로 transmit한다.
	update_status(status, value)	해당 main device의 id, 안전 상태, 센서값을 업데이트한다.
	send_data(fcmCO, fcmLPG)	이상 상황이 발생하여 경보를 발생시켜야 하는 경우, FCM을 통해 모바일에서 푸시 알림을 발생시킬 수 있도록 명령하며 TTS 기능을 통해 스피커로 경보 음성을 재생한다.
	device_off()	네트워크를 종료한다.

firebase.py		
Firebase Firestore와 통신하여 데이터를 DB에 저장하고, FCM(Firebase Cloud Messaging) notification를 통해 모바일 앱으로 Push 알림을 송신한다.		
Class	**Method**	**역할**
-	uploadState(path, COstate, LPGstate)	CO와 LPG의 로그를 받아와 데이터베이스에 리스트 형태로 저장한다.
	sendMessage(Manager_name, errorMessage)	CO나 LPG의 로그에서 위험 상황이 감지되면 즉시 FCM을 통해 warning 메시지를 모바일에 Push 알림 형태로 전송한다.

GTTS.py		
음성 경고 메시지가 담긴 mp3 파일을 지정 경로에 생성.		
Class	**Method**	**역할**
-	tts(text)	text를 mp3 파일로 변환한 뒤 같은 경로상에 저장한 후 재생한다.

2. Firestore – DataBase 구조

Collection	Document	Field	
Manager_name	FCM_Token	token : String	-
	Protected_name	address : String	
		x : Double	
		y : Double	
		age : Int	
		name : String	
		stateLogList : ArrayList<HashMap <Any, Any.>	COstate : Int
			LPGstate : String
			time : Timestamp

3. Mobile App (Android)

1. activity

MainActivity.kt		
Bottom Navigation View를 통해, Frament간 이동을 유연하게 해주는 Main Activity 입니다.		
Class	**Method**	**역할**
MainActivity	permissionCheck()	사용자로부터 위치 권한을 얻어온다.
	onRequestPermissionsResult(requestCode: Int, permissions: Array<out String>, grantResults: IntArray)	permission을 받은 후, 이벤트를 처리한다.
	navigationBarController()	Bottom Navigation View의 Item이 클릭되었을 때 프래그먼트 간 이동을 처리한다.
	fragmentChanger(fragment: Fragment)	Fragment간 이동을 한다.

LoginActivity.kt		
구글 로그인을 하고, 로그인 후 Firebase Firestore에 Manager 데이터를 저장.		
Class	**Method**	**역할**
LoginActivity	setGoogleLogin()	구글 로그인을 실행하는 method
	onActivityResult(requestCode: Int, resultCode: Int, data: Intent?)	구글 로그인 requestcode를 받았을 때, 구글 로그인
	firebaseAuthWithGoogle(idToken: String)	로그인 성공했을 때, updateUI() 실행
	updateUI()	MainActivity 시작

SplashActivity.kt
인트로 화면 3초간 보여준 뒤, LoginActivity 시작

AnimationActivity.kt		
액티비티 간 전환 시 전환 효과를 주는 액티비티		
Class	**Method**	**역할**
Animation Activity	onCreate onBackPressed finish	각 상황에 맞는 애니메이션을 실행한다.
Transition Mode	enum class	애니메이션 방향 정의를 해 놓은 enum class

MenuActivity.kt		
일산화탄소 감지 센서에서 일산화탄소 값을 받아와서 일정 수치이상이면 Warning을 리턴하는 파일이다.		
Class	**Method**	**역할**
MenuActivity	onCreate	레이아웃 적용 후, Firestore에서 Protected 데이터를 가져와 삭제할 수 있다.

RegisterActivity.kt		
일산화탄소 감지 센서에서 일산화탄소 값을 받아와서 일정 수치이상이면 Warning을 리턴하는 파일이다.		
Class	**Method**	**역할**
Register Activity	postMainServer(managerName: String, name: String, age: Int, address: String)	Firestore에 보호대상자 등록 및 데이터 추가

VersionActivity.kt
버전 정보를 보여주는 Activity

2. fragment

HomeFragment.kt		
보호대상자들을 보여주는 MainRecyclerView로 구성되어 있는 액티비티		
Class	**Method**	**역할**
HomeFragment	onCreateView	layout 적용

SearchFragment.kt		
카카오맵 API를 호출하여, 현위치 표시하고 보호대상자를 Firestore에 추가할 수 있는 프래그먼트		
Class	**Method**	**역할**
SearchFragment	UpdateAddress(address: String, name : String)	카카오맵 API에 Retrofit2를 통해 연결하여, 데이터를 마커로 표시한다.
	searchOnlyAddress(address: String)	주소를 String으로 받아 위치를 찾는다.
	searchAddress(address: String)	주소를 String으로 받아 위치를 찾고 마커를 표시한다.
	refreshMarkerData()	마커를 업데이트한다.
	addMarkerData(context: Context)	마커를 추가한다.
	zoomIn()	지도의 줌레벨을 확대한다.
	zoomOut()	지도의 줌레벨을 축소한다.
	setLocation(x: Double, y:Double)	위도, 경도 데이터를 받아 해당 위치에 위치시킨다.
	allowMarker(x: Double, y: Double, address: String)	보호대상자를 등록시키는지 여부를 묻는 다이얼로그 생성
	findFineLocation()	현위치 조회
	stopFindingFineLocation()	현위치 조회 취소
MarkerEventListener	onPOIItemSelected(p0: MapView?, p1: MapPOIItem?)	마커 선택 시, 카카오맵 중앙으로 이동
	onCalloutBalloonOfPOIItemTouched(p0: MapView?, p1: MapPOIItem?)	말풍선 클릭 시, 현재 CO 및 가스 누출 상태 알려주는 NotifiactionActivity 시작

ProfileFragment.kt		
보호대상자들을 보여주는 MainRecyclerView로 구성되어 있는 액티비티		
Class	**Method**	**역할**
ProfileFragment	onCreateView	layout 적용

3. adaptor

MainRvAdapter.kt		
HomeFragment에 있는 MainRecyclerView에 아이템과 아이템 클릭 이벤트를 적용		
Class	**Method**	**역할**
MainRvAdapter	getItemCount() : Int	item 개수 count
	onCreateViewHolder(parent: ViewGroup, viewType: Int): Holder	view 생성
	onBindViewHolder(holder: Holder, position: Int)	view를 holder와 연결
Holder	bind(protected: Protected, context : Context)	item click 이벤트 처리 및 item layout 구성

MapRvAdapter.kt		
SearchFragment에 있는 MapRecyclerView에 아이템과 아이템 클릭 이벤트를 적용		
Class	**Method**	**역할**
MainRvAdapter	getItemCount() : Int	item 개수 count
	onCreateViewHolder(parent: ViewGroup, viewType: Int): Holder	view 생성
	onBindViewHolder(holder: Holder, position: Int)	view를 holder와 연결
	setItemClickListener()	item이 클릭 되었을 때의 이벤트 처리를 설정
Holder	bind(protected: Protected, context : Context)	item click 이벤트 처리 및 item layout 구성
ItemClickListener ※interface	onClick(view: View, position: Int)	아이템 클릭 시 이벤트

SettingRvAdapter.kt		
ProfileFragment의 보호대상자 관리에서 아이템과 아이템 클릭 이벤트를 적용		
Class	**Method**	**역할**
MainRvAdapter	getItemCount() : Int	item 개수 count
	onCreateViewHolder(parent: ViewGroup, viewType: Int): Holder	view 생성
	onBindViewHolder(holder: Holder, position: Int)	view를 holder와 연결
Holder	bind(protected: Protected, context : Context)	item click 이벤트 처리 및 item layout 구성

4. data

DBdata.kt		
Firestore에 저장하는 DB 데이터 클래스		
Class	**Data**	**역할**
Manager	id : String	관리자 id
	name : String	관리자 이름
Protected	name : String	보호대상자 이름
	age : Int	보호대상자 나이
	address : String	보호대상자 주소
	x : Double	보호대상자 주소 x좌표
	y : Double	보호대상자 주소 y좌표
	stateLogList :ArrayList<HashMap<Any, Any>>?	CO 및 가스 누출 상태 로그

SettingRvAdapter.kt		
Kakao map에 Request 보내는 데이터 클래스		
Class	**Data**	**역할**
ResultSearchAddress	documents : ArrayList<Place>	주소 리스트
Place	address_name : String	주소 이름
	adress_type : String	주소 유형
	x : String	x좌표
	y : String	y좌표

5. util

FireStoreManager.kt		
Firestore 데이터 읽기, 쓰기, 삭제, 추가를 용이하게 해주는 class이다.		
Class	**Method**	**역할**
FireStore Manager	loadProtectedData(): ArrayList<Protected>	Firestore에서 보호대상자 목록을 리턴하는 메서드이다.
	loadProtectedPersonalData(protected : String)	Firestore에서 특정 보호대상자 한 사람을 리턴한다.
	insertProtectedData(protected: Protected)	Firestore에 특정 보호대상자를 추가한다.
	deleteProtectedData(protected : String)	Firestore에 특정 보호대상자를 삭제한다.

GasosMessagingService.kt		
FCM을 용이하게 사용할 수 있도록 해주는 class이다.		
Class	**Method**	**역할**
GasosMessagingService	onMessageReceived(remoteMessage: RemoteMessage)	FCM 메시지를 받았을 때 호출된다.
	sendNotification(title: String?, body: String)	Push 알림을 생성하고, 클릭 시 이벤트를 호출한다.
	onNewToken(token: String)	새로운 FCM 토큰을 발급하는 메서드이다.
	sendToken(uid : String, token : String)	FCM 토큰을 Firestore에 저장한다.
	myToken()	현재 FCM 토큰을 가져온다.

GlobalApplication.kt
Kakao SDK 앱키가 있는 Application을 상속받은 class이다.

ProfileFragment.kt		
Retrofit2를 활용해, 카카오맵 api에서 검색하는 interface이다.		
Interface	**Method**	**역할**
KakaoAPI	getSearchAddress(@Header("Authorization") key : String, @Query("query") query : String) : Call<ResultSearchAddress>	주소 검색하는 메서드이다.

○ 주요 함수의 흐름도

- MainCordinator.py

  ◦ Device & DB Initialize -> Repeat [LPG 모듈 상태 확인 (transfer, receive) -> CO 모듈 상태 확인 (transfer, receive) -> 이상 상황이 발생했다면 서버로 FCM 명령 전송 및 TTS 음성 출력 (send_data) -> 현재 장치들의 상태값 서버로 전송 (uploadState) -> 반복]

○ 기술적 차별성

- 비교적 가벼운 DB 서버를 사용하였기에 속도가 빠르고 유지와 관리가 용이함
- 확장형 물리적 스타 토폴로지 센서 네트워크 구조를 채택하여 하나의 엔드 디바이스의 네트워크 연결 문제가 생겼을 시 상황을 빠르게 인식할 수 있으며 다른 디바이스에 영

향을 주지 않음. 또한 향후 추가적인 감지 센서나 제어 장치의 필요성이 발생했을 경우 네트워크의 확장이 용이함

- 소형의 RaspberryPi Zero와 아두이노를 엔드 디바이스의 프로세서로 사용하여 설치와 이동이 간편하고, RaspberryPi의 경우 LPGdetect.py를 시작프로그램으로 등록하여 별도의 조작 없이 단순히 전원을 공급하는 것만으로 사용 준비가 완료됨
- FCM 메시지를 사용하여 데이터베이스 서버를 거치지 않고 바로 모바일 기기에 알람을 보내므로 긴박한 상황에서 빠른 데이터 송수신이 이루어질 수 있다.

□ 개발 중 장애요인과 해결방안

○ ZigBee 설정 장애

- XBee 칩을 초기 설정하기 위해서 필수적인 XCTU 프로그램을 활용하기 위해서는 XBee 칩을 USB로 XCTU 프로그램이 설치된 데스크탑 또는 노트북에 연결할 필요가 있는데, USB 인식이 정상적으로 되지 않아 개발이 일시적으로 중단됨
- XBIB-U (XBee PRO Interface Board)를 추가로 구매하여 해결

○ RaspberryPi와 ZigBee의 호환성 문제

- Arduino XBee Shield 등의 보조 장치를 통해 사용자가 직접 직렬 통신을 설정하지 않아도 XBee 칩을 쉽게 활용할 수 있는 Arduino와 달리, RaspberryPi의 Linux기반 라즈비안 OS 환경에서는 다양한 사전 설정이 필요했음
- UART를 지원하는 핀이 1쌍(Tx,Rx)뿐인 RaspberryPi Zero W는 하드웨어적 UART(PL011, UART0)과 소프트웨어적으로 구현된 miniUART(UART1)가 존재하나, miniUART는 많은 제약과 단점이 존재하여 하드웨어적 UART를 사용하려 하였음. 그러나 해당 UART가 디폴트로 Bluetooth 기능에 할당되어 있는 관계로, Bluetooth 기능을 비활성화하고 시리얼 통신에 UART0을 할당해서 문제를 해결함
- UART를 지원하는 핀이 6개인 RaspberryPi 4도 /boot/config.txt 경로의 파일을 수정해 GPIO 핀을 UART 기능에 할당해야 XBee 칩과 통신이 가능

○ Node.js 기반 서버의 기능 장애

- 처음 개발하고자 하였던 Node.js와 express를 사용한 REST API 서버에서 Retrofit2를 통한 통신 구현에 문제가 있었고 DB 서버 연결 역시 어려움이 있었음
- 기존의 방식으로는 불필요하고 무거운 서버가 구성될 것으로 예상하여 DB 서버로 전환하여 유연한 NoSQL 방식인 Firebase Firestore를 사용하였고 모바일과 센서 네트워크 간 통신은 FCM(Firebase Cloud Messaging)을 사용하여 문제점을 개선함

□ 개발결과물의 차별성

○ 여러 감지 기능을 하나의 시스템 내에 갖춘 통합 솔루션
- 현재 일산화탄소와 도시가스 감지 기능을 하나의 앱 내에서 이용할 수 있으며, 향후 네트워크의 확장으로 다양한 센서를 추가한 통합 솔루션으로 발전 가능
- 다수의 감지 장치를 일일이 확인할 필요 없이, 단일 모바일 화면에서 복수의 장치 안전 상태를 체크할 수 있으므로 기존의 단일 솔루션들을 다수 활용하는 것에 비해 편리함

○ Xbee 무선통신 사용
- 전력효율이 뛰어난 Xbee를 통해 네트워크를 구성하여 배터리 수명 부분에서 우수함

○ 모바일 App 채택으로 접근성 용이
- 관리자의 핸드폰으로 24시간 내내 보호 대상자의 안전 상태를 확인할 수 있으므로 안전 사각지대가 줄어듦
- 눈으로 확인하기 어려운 센서 값들을 애플리케이션에서 직관적인 확인이 가능하고 더욱 유연한 대처가 가능함

○ User-Friendly Application
- Google 로그인, Kakao map API 등 대중에게 친숙한 기존 서비스를 적극적으로 활용하여 제작하였기에 사용자가 더욱 빠르게 애플리케이션에 적응할 수 있음
- 긴급상황 시 순간적인 패닉에 빠질 수 있으므로 자동으로 119 다이얼링을 수행하여 보다 빠르고 정확한 대처를 할 수 있도록 함

□ 개발 일정

No	내용	2020年			
		6月	7月	8月	9月
1	아이디어 기획, 구체화				
2	HW 설계 및 구현				
3	DB서버 설계 및 구현				
4	모바일 앱 개발				
5	모바일, HW, 서버간 통신 구현				
6	종합 디버그				
7	시험 평가 및 테스트				

□ 팀 업무 분장

No	구분	성명	참여인원의 업무 분장
1	팀장	최지희	- 프로젝트 아이디어 제안 및 총괄 - DB(Firebase firestore) 구조설계 - FCM(Firebase Cloud Messaging)을 통한 모바일/센서간 통신 구현 - 서버 구조설계
2	팀원	민대인	- GASOS 안드로이드 애플리케이션 제작 - Firebase와 모바일 간 네트워크 구성 - GASOS 모바일 앱 UI / UX 디자인 - TTS (Text To Speech) 기능 구현
3	팀원	박보서	- Python 코드 정리 및 최적화 - Node.js 기반의 서버 통신 설계
4	팀원	이제혁	- 무선 센서 네트워크 설계 - 하드웨어 설계 및 구현 - Arduino, RaspberryPi 소프트웨어 프로그래밍 - 개발완료보고서 작성 및 영상 제작