**인하대학교**

**컴퓨터공학 종합설계**

**상세설계보고서**

*[시각장애인을 위한 음성 안내 APP]*

컴퓨터공학과

8조

김난영 김혜윤 이영주

**목차**

1. 개요 6

1.1 목적 6

1.2 구성 7

1.3 범위 7

1.4 용어, 약어 정리 7

1. 시스템 구성 8

2.1 시스템 구성도 8

2.2 소프트웨어 9

2.3 개발 환경 10

1. 기능 10

3.1 APP 내 기능 10

1. 설계 11

4.1 데이터베이스 11

4.2 Object Detection 14

4.3 Application 21

5. 세부 추진 계획 및 일정 29

**그림 목차**

[Figure 1] 시스템 아키텍처 8

[Figure 2] 소프트웨어 구성도 9

[Figure 3] 전체 DB 스키마 설계 11

[Figure 4] 신호등 정보 저장용 12

[Figure 5] 잘못된 점자블록 정보 저장용 DB 12

[Figure 6] 위험 요소, 무조건 배제 여부를 저장하는 DB 13

[Figure 7] 신호등 관련된 텍스트를 저장하는 DB 13

[Figure 8] 위험 요소와 관련된 텍스트를 저장하는 DB 13

[Figure 9] Firebase Traffic\_Light 데이터베이스 14

[Figure 10] Firebase Text\_TrafficLight 데이터베이스 14

[Figure 11] 이미지 추출 방법 15

[Figure 12] 이미지 추출 예시 15

[Figure 13] 수정된 predefined\_classes.txt 16

[Figure 14] 수정된 my-tiny-yolo.cfg 파일 16

[Figure 15] labemImg 실행 화면 18

[Figure 16] labelImg 로 라벨 지정하기 18

[Figure 17] 모델 학습을 위한 명령어 18

[Figure 18] 학습 환경 19

[Figure 19] 학습 결과 19

[Figure 20] 테스트 명령어 20

[Figure 21] test data 를 넣은 결과 20

[Figure 22] Google MAP API meta-data 코드 21

[Figure 23] Google Map API 코드 21

[Figure 24] 어플리케이션 내에 Google Map 22

[Figure 25] Permission 확인 코드 22

[Figure 26] setDefaultLocation() 23

[Figure 27] T Map API 연동 코드 24

[Figure 28] FindPOI() 24

[Figure 29] StartSearchPath() 1 25

[Figure 30] StartSearchPath() 2 26

[Figure 31] 경로 탐색 결과 27

[Figure 32] 퍼미션 체크 28

[Figure 33] 음성 입력 후 텍스트 변환 1 28

[Figure 34] 음성 입력 후 텍스트 변환 2 28

[Figure 35] TTS 초기 설정 29

[Figure 36] TTS.speak() 29

**표 목차**

[표 1] 약어 및 풀이 8

[표 2] 개발 환경 10

[표 3] 이미지 추출 명령어 요소 설명 15

[표 4] labelImg 단축키 18

[표 5] 모델 학습 명령어 요소 20

[표 6] 테스트 명령어 요소 20

[표 7] 세부 일정 및 추진 계획 29

**1. 개요**

본 장에서는 Google MAP API, T MAP API와 Real-Time Object Detection을 이용하여 시각장애인에게 안전한 길을 음성으로 안내하는 서비스에 대한 목적, 구성, 범위 그리고 본 문서의 개요를 소개한다.

**1.1 목적**

본 프로젝트의 목적은 Google MAP API, T MAP API와 Real-Time Object Detection을 이용하여 시각장애인에게 음성으로 안전한 길을 안내하는 시스템을 구축하는 것이다.

본 프로젝트는 기존 ‘커뮤니티 맵핑’, ‘강남 지팡이’ 등 시각장애인을 위한 국내 어플리케이션에서 제공하지 않았던 신호등 정보와 위험 요소를 파악하여 인하대 후문 범위 내에서 안전한 길을 안내해주는 서비스를 제공한다.

본 문서는 Google MAP API, T MAP API와 Real-Time Object Detection을 이용하여 시각장애인에게 안전한 길을 음성으로 안내하는 시스템에 대한 기본적 요구사항의 식별과 이해를 위해 작성되었으며, 아래의 사항을 구체적으로 명시하는데 목적이 있다.

- 화면 구성을 파악하는데에 어려움이 있는 시각장애인에게 편의를 제공하기 위해 한 번 터치 시 음성 출력, 두 번 터치 시 컴포넌트가 선택되도록 한다.

- STT API를 이용하여 사용자의 음성을 텍스트로 변환한다.

- TTS API를 이용하여 텍스트를 음성으로 변환하여 사용자에게 제공한다.

- Google MAP API를 이용하여 어플리케이션 내에 지도를 생성하고 GPS와 함께 사용자의 위치를 실시간으로 확인한다.

- TMAP API를 이용하여 보행자의 도보 경로를 제공한다.

- DB에 보행자에게 안전하지 않을 수 있는 요소를 추가하여 경로 탐색 시 경로에서 배제한다.

- TTS API를 이용하여 위험요소에 해당하나 필수적으로 경로에 포함되어야 하는 요소를 음성으로 안내한다.

- Darkflow YOLO를 이용하여 실시간으로 신호등을 탐지해 사용자에게 신호등 정보를 안내한다.

**1.2 구성**

본 문서는 아래와 같이 구성되어 있다.

제 1장 개요

본 문서의 구성, 범위, 용어 설명이 기술되어 있다.

제 2장 시스템 구성

프로젝트의 시스템 구성, 소프트웨어, 개발환경이 기술되어 있다.

제 3장 기능

프로젝트의 기능이 상세하게 기술되어 있고, 이를 테이블로 명시하고 있다.

제 4장 설계

데이터베이스, Object Detection, Application 설계에 대한 자세한 내용이 기술되어 있다.

제 5장 세부 추진 계획 및 일정

프로젝트의 추진 계획 및 추후 일정에 대해 기술되어 있다.

**1.3 범위**

시각장애인에게 위험 요소를 배제한 안전한 경로를 제공하는 시스템으로, 적용 범위를 인하대 후문으로 한정하였다. 이는 위험 요소에 관한 정보를 제한된 인원 및 환경 속에서 관리하기 용이하게 하기 위해서이다.

**1.4 용어 및 약어 정리**

본 문서에서 사용된 용어 및 그에 대한 설명은 다음과 같다.

- Darknet : C와 CUDA로 작성한 Neural Network 오픈 소스 프레임워크이다.

- Darkflow : C로 작성된 YOLO를 Tensorflow를 이용하여 Python기반으로 작성한 것이다. 즉 Darknet 을 Tensorflow 에 적용한 것이다.

- YOLO : Real-Time Object Detection이 가능한 시스템이다. 이전 탐지 시스템과 다르게 하나의 신경망을 전체 이미지에 적용한다. 이 신경망은 이미지를 영역으로 분할하고 각 영역의 Bounding Box와 확률을 예측한다.

본 문서에서 사용된 약어 및 풀이는 다음과 같다.

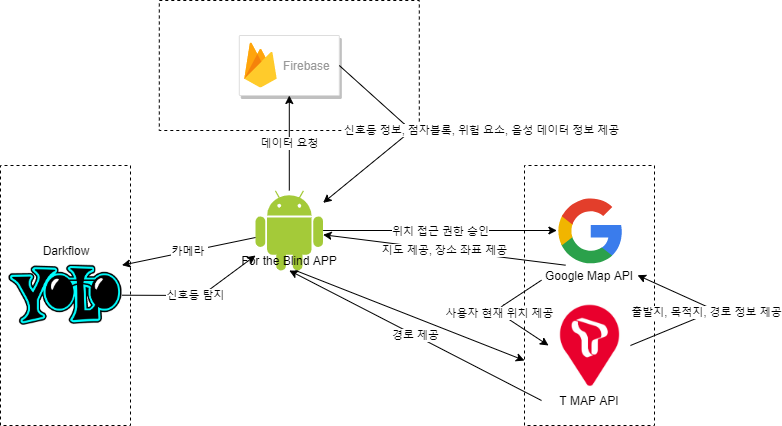
|  |  |  |
| --- | --- | --- |
| **약어** | **풀이** | **비고** |
| STT | Speech-To-Text |  |
| TTS | Text-To-Speech |  |
| YOLO | You Only One Look |  |
| DB | DataBase |  |

**[표 1] 약어 및 풀이**

**2. 시스템 구성**

본 장에서는 Google MAP API, TMAP API와 Object Detection을 활용한 모바일 안드로이드 어플리케이션을 개발하여 인하대 후문에서 시각장애인들이 안전하게 이동할 수 있도록 돕는 ‘For the Blind’의 배경과 목표, 기술, 개념 사항에 대해 기술한다.

**2.1 시스템 구성도**

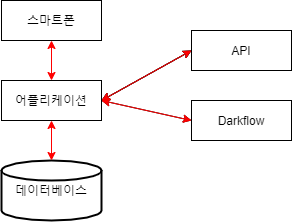
****

**[Figure 1] 시스템 아키텍처**

시스템 기능 설명

* 어플리케이션은 Android application을 통해 API를 요청 후 받아오고 경로 안내 서비스를 제공한다.
* 데이터베이스는 Firebase 서버를 이용하여 통신한다.
* 사용자가 입력한 목적지를 STT & TTS API로 가져오고 사용자의 위치를 Google MAP API로 얻어와 출발지와 목적지를 설정한다.

**2.2 소프트웨어**

****

**[Figure 2] 소프트웨어 구성도**

본 프로젝트에서는 안드로이드 어플리케이션을 통해 API를 요청 및 받아오고, 사용자에게 경로 안내를 하며 Darkflow를 활용한 Object Detection을 통해 신호등 탐지 서비스를 제공한다. 본 절에서는 소프트웨어를 기술한다.

2.2.1 Android SDK

Android SDK를 이용하여 안드로이드 소프트웨어에서 제공하는 라이브러리들을 사용한다. 이는 View와 각종 Activity를 포함하며 각종 Java 라이브러리를 포함한다.

2.2.2 Firebase 실시간 데이터베이스 SDK

안드로이드 어플리케이션에 필요한 데이터 저장하기 위한 데이터베이스로 사용된다. 신호등, 잘못된 점자블록, 위험요소에 관한 정보 및 음성 안내 텍스트를 저장해서 관리한다.

**2.3 개발 환경**

|  |  |  |  |
| --- | --- | --- | --- |
| **API** | **Android Application** | **Database** | **Object Detection** |
| Google Map API | Android Studio | Firebase | Darkflow |
| T MAP API | JAVA |  | YOLO |
| STT API |  |  |  |
| TTS API |  |  |  |

**[표 2] 개발 환경**

**3. 기능**

본 장에서는 프로젝트 진행을 위한 기술에 대하여 어플리케이션 사용에 대한 UX 순서를 기준으로 작성하였다.

**3.1 APP 내 기능**

본 어플리케이션에서는 시각장애인의 편의를 위해 한번 터치하면 해당 컴포넌트 내용이 음성으로 출력되고, 두번 터치하면 컴포넌트가 선택된다. 이 기능이 어플리케이션 전체에 적용되도록 구현하였다.

3.1.1 사용 방법 팝업

화면 구성을 파악하는 데에 어려움이 있는 시각장애인을 위하여 어플리케이션 시작 시 팝업창을 띄워 사용 방법을 안내한다. 한 번 터치시 안내사항이 음성으로 출력되고, 두 번 터치 시 팝업창이 닫히도록 구현하였다.

3.1.2 목적지 검색

목적지를 타이핑하여 검색하는 것이 어려운 시각장애인을 위해 목적지를 음성으로 검색하는 기능을 구현하였다. 음성으로 입력한 값을 텍스트로 변환하여 이를 다시 음성 변환을 통해 사용자에게 목적지가 맞는지 확인시킨다. 사용자가 음성으로 입력한 출력값과 원하는 목적지의 일치 여부에 따라 실행 결과가 달라진다.

3.1.2.1 일치할 경우

백 버튼을 터치하여 사용자 음성의 변환 결과 값이 목적지로 지정된다.

3.1.2.2 불일치할 경우

화면 두 번 터치를 하여 이전화면으로 돌아간 뒤 목적지를 다시 검색한다.

3.1.3 안전한 경로 탐색

목적지가 입력되면 Google MAP API와 T MAP API를 이용하여 안전한 경로를 사용자에게 안내한다.

3.1.3.1 사용자 주변 위험요소 발견

지정된 경로 상에 위험요소가 있는 경우, 사용자가 DB에 저장된 위험요소의 위치에 근접하면 경고 음성이 출력된다.

3.1.4 실시간 신호등 탐지

지정된 경로 상에 횡단하는 경우가 있을 경우, 사용자가 횡단하기 위해 DB에 저장된 신호등 위치에 근접하면 카메라로 전면을 비추라는 음성 안내가 출력된다. 카메라에 입력된 영상에 신호등의 정보를 검출하여 사용자에게 음성으로 안내함으로써 안전한 횡단을 돕는다.

**4. 설계**

**4.1 데이터베이스**

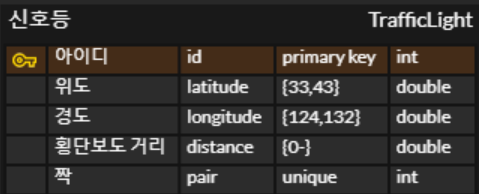
4.1.1 데이터 베이스 구조

****

**[Figure 3] 전체 DB 스키마 설계**

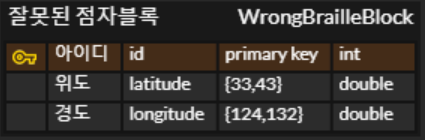
전체 DB 스키마 설계이다. 총 4 개의 DB 테이블을 만들었고 이를 Firebase에 저장하여 활용할 계획이다.

4.1.2 데이터 베이스 테이블

****

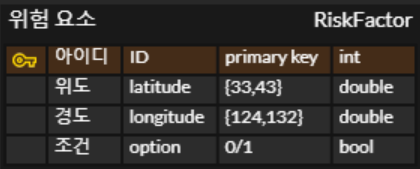
**[Figure 4] 신호등 정보 저장용 DB**

인하대 후문 범위 내에 있는 신호등에 관한 정보를 저장한 데이터베이스이다. Primary key인 신호등의 아이디(id)와 위도(latitude), 경도(longitude), 횡단보도 거리(distance), 신호등의 짝(pair)으로 구성되어 있다. GPS로 받은 사용자의 위치가 저장된 신호등의 일정범위 안에 포함되면 음성 안내가 출력된다. 짝을 지정하지 않았을 경우, 횡단을 했음에도 불구하고 또 다시 건너편 신호등 범위 안에 포함되어 안내가 중복되고 이는 경로 진행에 혼선을 유발한다. 이를 방지하기 위하여 신호등의 짝을 지정하였다.

****

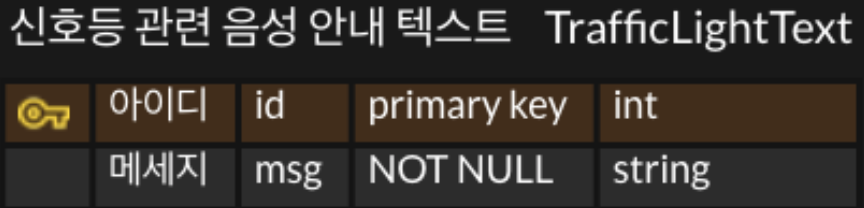
**[Figure 5] 잘못된 점자블록 정보 저장용 DB**

지정된 범위 내의 잘못 표기된 점자블록의 정보를 저장하는 데이터베이스이다. Primary key인 아이디(id), 점자블록의 위도(latitude)와 경도(longitude) 로 구성되어 있다.

****

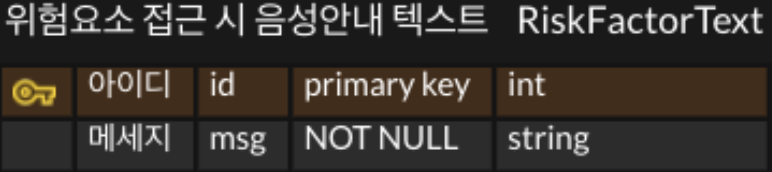
**[Figure 6] 위험 요소를 저장하는 DB**

지정된 범위 내의 위험요소들의 정보를 저장하는 데이터베이스이다. 요소들의 위도(latitude), 경도(longitude), 조건(option) 그리고 primary key인 아이디(id)로 구성되어 있다. 여기서 조건은 경로 탐색 시 무조건 배제 여부이다. 공사현장과 같이 경로상에 무조건적으로 포함하지 않을 곳은 0으로 지정하고 도착 지점이 포함될 수 있는 위험요소 같은 경우에는 1로 지정하였다.

****

**[Figure 7] 신호등 관련된 텍스트를 저장하는 DB**

신호등 탐지를 위해 진행과정 순서에 따라 사용자에게 안내할 텍스트를 저장한 데이터베이스이다. 텍스트의 아이디(id), 메시지(msg) 그리고 primary key인 아이디(id)로 구성되어 있다.

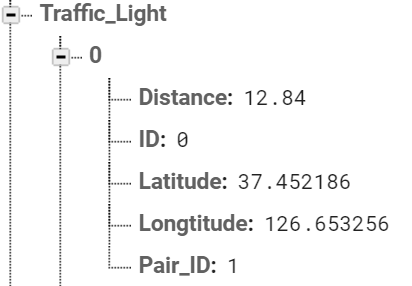
****

**[Figure 8] 위험 요소와 관련된 텍스트를 저장하는 DB**

위험요소에 접근 시 음성으로 안내될 텍스트가 저장 되어있는 데이터베이스이다. Primary key인 아이디(id)와 출력될 메시지(msg)로 구성되어 있다.

4.1.3 Firebase 구조

생성한 데이터베이스 테이블을 기반으로 Firebase에 데이터베이스를 추가하였다.



**[Figure 9] Firebase Traffic\_Light 데이터베이스**



**[Figure 10] Firebase Text\_TrafficLight 데이터베이스**

**4.2 Object Detection**

Object Detection이란 이미지에서 특정 Object의 위치를 감지하여 해당 영역 안의 Object를 분류하는 작업이다.

4.2.1 darkflow 클론

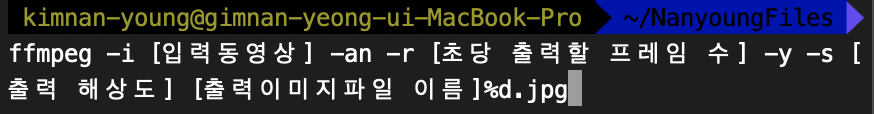
<https://github.com/thtrieu/darkflow> 에서 darkflow repository를 클론한다.

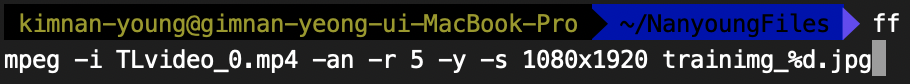
4.2.2 동영상 촬영

구글에서 이미지를 크롤링 하는 것 대신 본 어플리케이션의 적용 범위가 ‘인하대학교 후문’ 이라는 것을 고려하여 직접 후문의 신호등을 영상으로 촬영함으로써 훈련 데이터를 수집한다.

4.2.3 이미지 추출

촬영한 영상에서 이미지 파일을 추출함으로써 훈련을 위한 데이터 셋을 구축하였다.

**[Figure 11] 이미지 추출 방법**

**[Figure 12] 이미지 추출 예시**

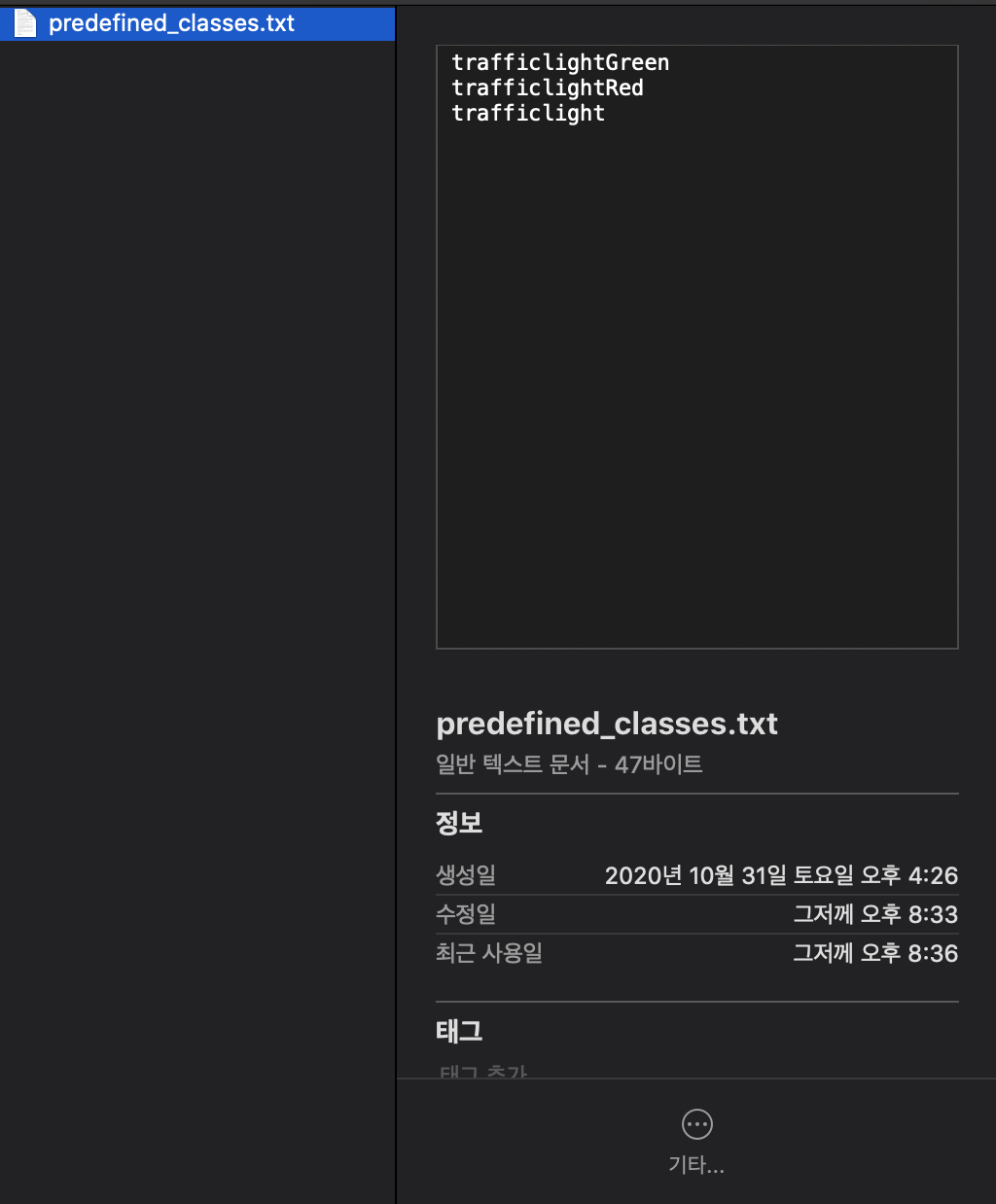
영상 스트림 프로그램인 ffmpeg을 활용하여 위와 같이 이미지를 추출하였으며, 아래의 명령어를 실행하면 영상이 저장된 경로에 이미지가 추출되어 저장된다.

|  |  |
| --- | --- |
| -an | 오디오 무시 |
| -r 5 | 초당 5 프레임 추출 |
| -s | 결과 이미지 크기 지정 |
| -1080x1920 | 결과 이미지 크기 |

**[표 3] 이미지 추출 명령어 요소 설명**

4.2.4 predefined\_classes.txt 수정

신호등 검출을 위해 총 세개의 클래스 라벨을 지정하였다.



**[Figure 13] 수정된 predefined\_classes.txt**

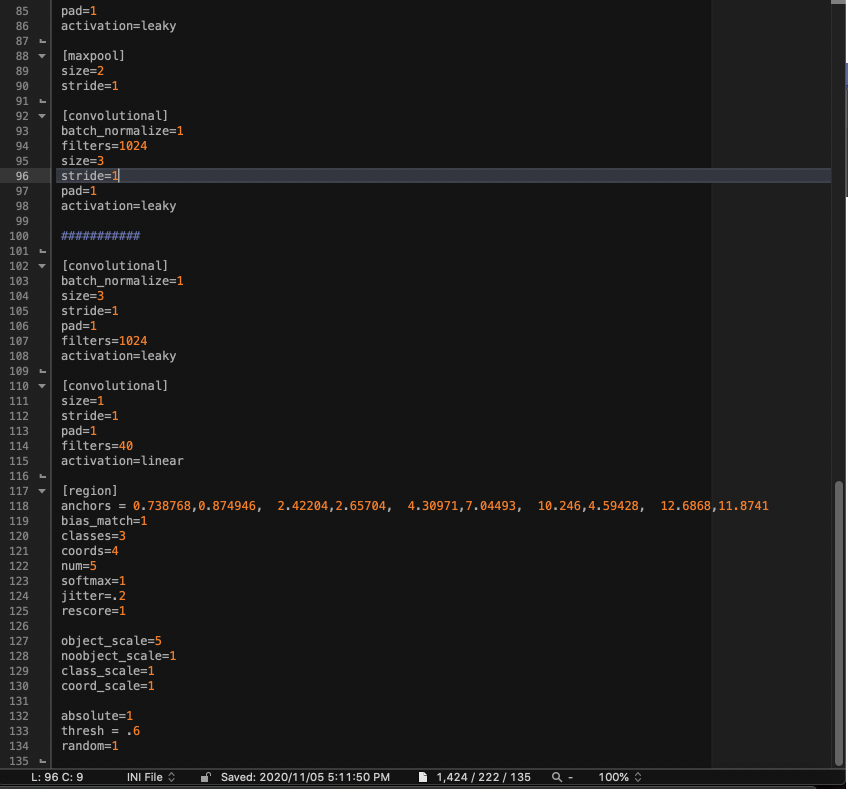
labelImg/data/predefined\_classes.txt 을 위와 같이 수정하였다. ‘trafficlightGreen’ 은 보행자 신호가 초록불인 상태의 이미지를 의미하고, ‘trafficlightRed’는 보행자 신호가 빨간불인 상태의 이미지를 의미하며, ‘trafficlight’는 신호에 불이 들어오지 않은 상태의 이미지를 의미한다.

4.2.5 label.txt 수정

darkflow/label.txt 도 [Figure 13]와 동일하게 수정한다.

4.2.6 cfg 파일 수정

darkflow/cfg 디렉토리 내의 cfg 중 사용할 모델을 수정한다.



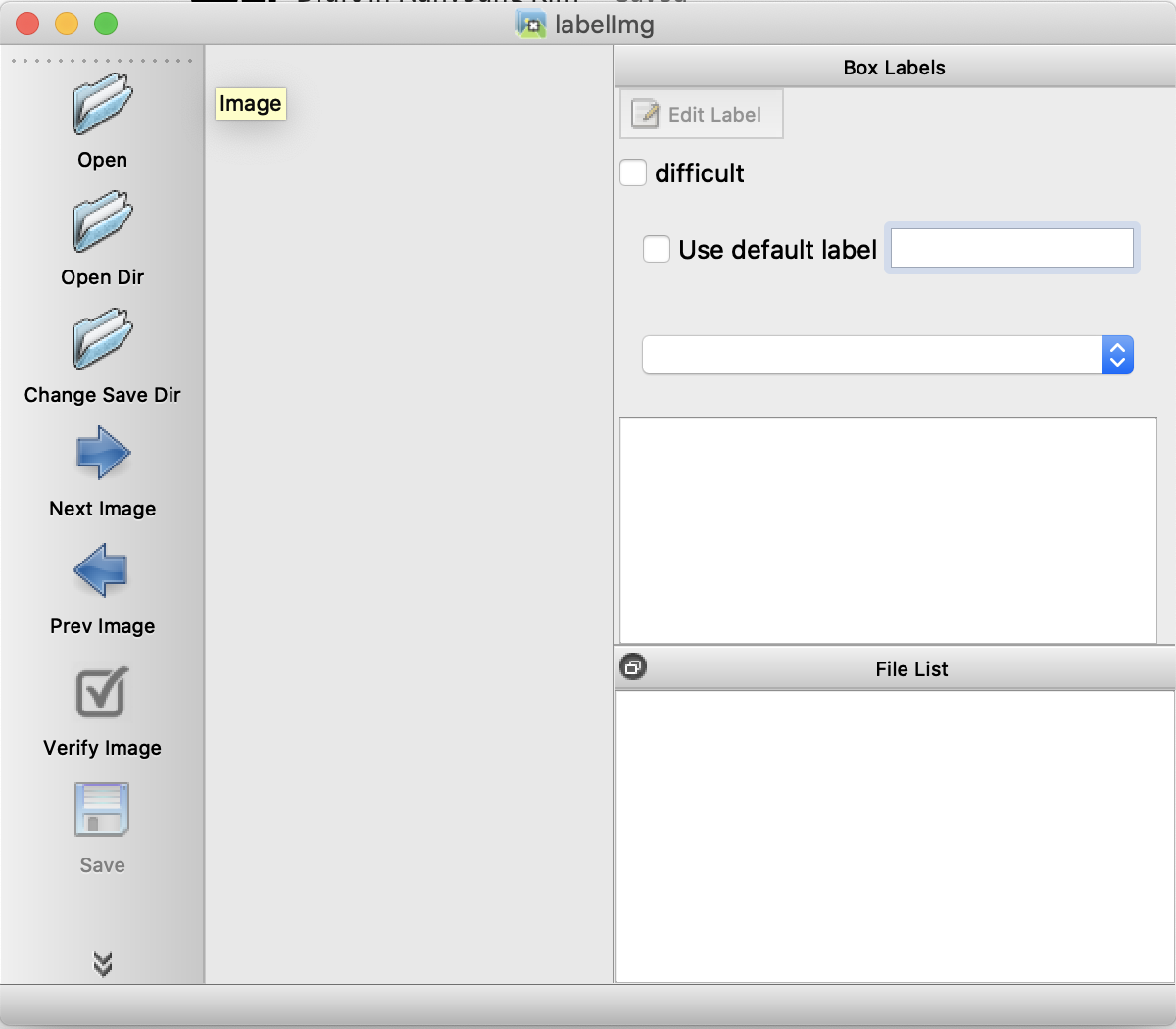
**[Figure 14] 수정된 my-tiny-yolo.cfg 파일**

모바일 애플리케이션에 사용될 모델이기 때문에 tiny-yolo.cfg 를 이용하였고 해당 파일을 my-tiny-yolo.cfg 로 파일명을 바꾼 뒤 위와 같이 Line 114,120을 수정하였다.

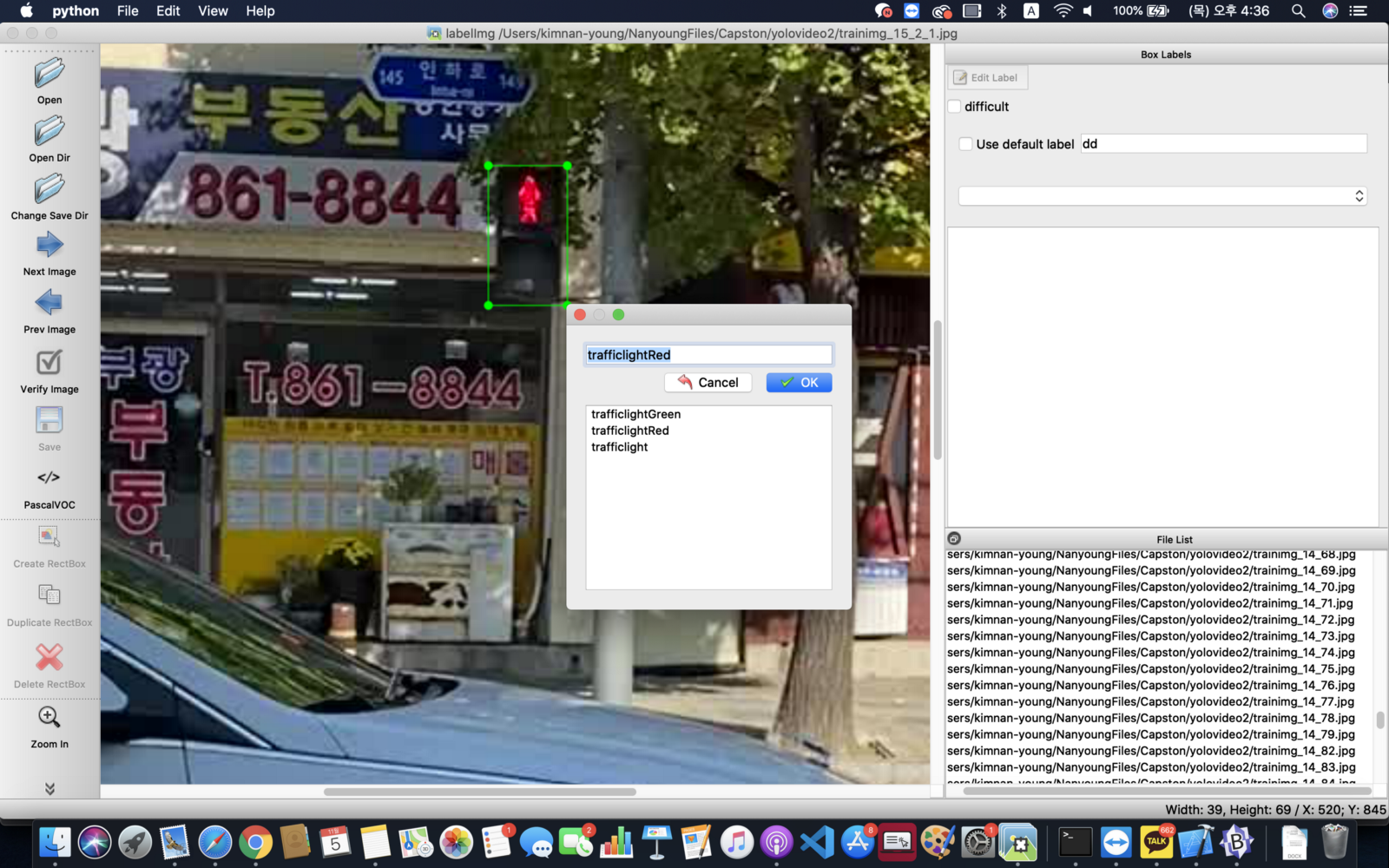
Line114의 Filters와 Line 120의 classes를 수정해준다. Supervised Learning 을 위한 총 라벨 수가 3개(‘trrafficlightGreen’, ‘trafficlightRed’, ‘trafficlight’) 이므로 ‘classes = 3’으로 지정한다. filters = (classes수 + 5) \* 5 로 지정해줘야 하며 본 어플리케이션에서는 (3+5)\*5 = 40으로 지정하였다.

4.2.7 이미지 라벨링

YOLO 모델은 Supervised Learning 이므로 정답인 클래스 라벨을 지정한 뒤 학습시켜야 한다. Object Detection의 정답 라벨은 각 객체의 Class Label과 Boundary-box의 쌍으로 구성되며 이를 Annotation이라 부른다. LabelImg를 이용해 트레이닝 이미지의 라벨을 지정함으로써 추출한 이미지에 Annotation을 할당하였다.



**[Figure 15] labemImg 실행 화면**

****

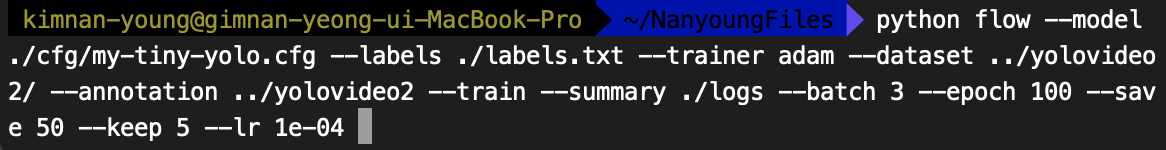
**[Figure 16] labelImg 로 라벨 지정하기**

이미지를 불러와서 신호등에 라벨을 지정한 상태이다.

|  |  |
| --- | --- |
| w | Create rectangle box |
| d | Next image |
| a | Previous image |

**[표 4] labelImg 단축키**

4.2.8 모델 학습

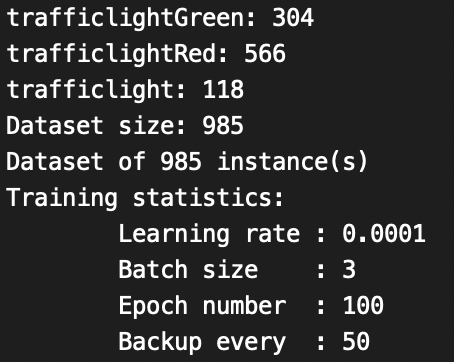


**[Figure 17] 모델 학습을 위한 명령어**

위의 명령어를 터미널에 입력하여 라벨링 한 데이터로 모델 학습을 시작한다.

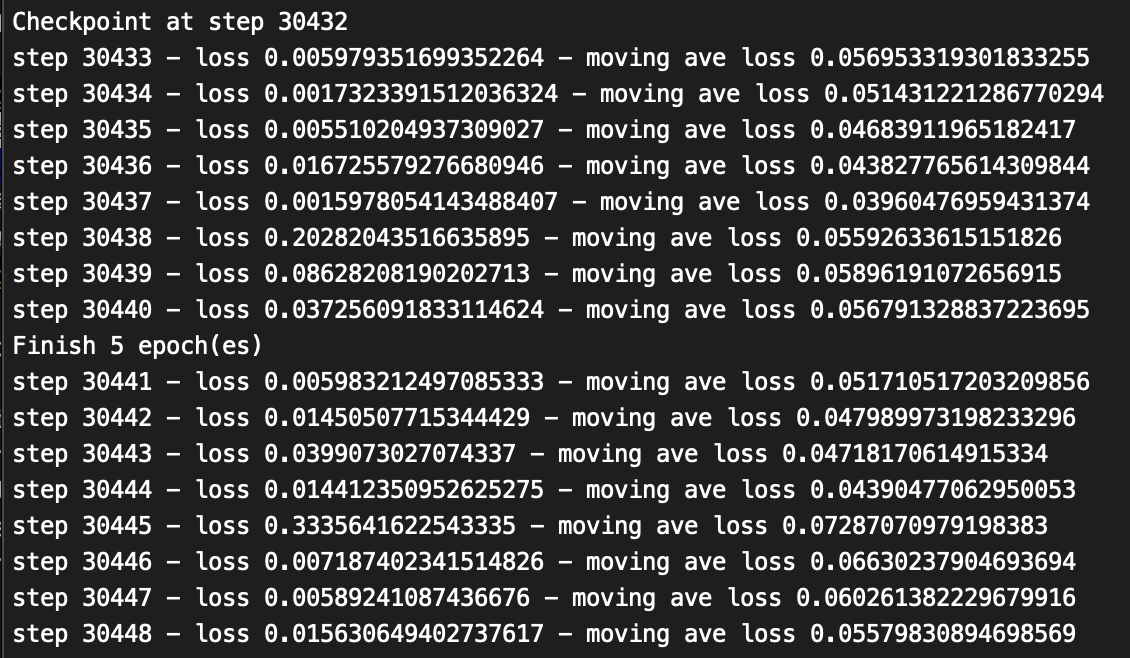
|  |  |
| --- | --- |
| -trainer | Optimizer 설정 |
| -lr | Learning rate (1e-04 : 0.0001) |
| -gpu | Gpu 사용 여부 |
| -load | 이전 학습 가중치 사용 여부. -1 설정 시 마지막 save 를 불러오며 특정 step 으로 설정 시 해당 step 부터 학습 이어서 시작. |

**[표 5] 모델 학습 명령어 요소**

****

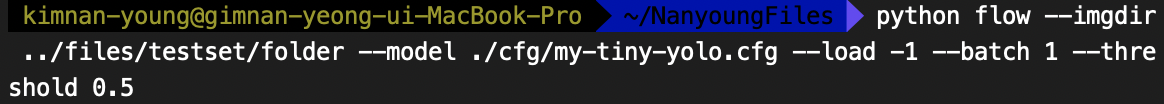
**[Figure 18] 학습 환경**

4.2.7 학습 결과

****

**[Figure 19] 학습 결과**

30,000번 이상의 step을 훈련한 모델의 loss 값을 확인할 수 있다. 초반의 loss 값은 100초반 대였으나 점점 0에 가까워진 모습이다.

****

**[Figure 20] 테스트 명령어**

|  |  |
| --- | --- |
| --imgdir | 테스트할 이미지가 위치한 디렉토리 경로 |
| --model | Cfg 모델이 위치한 디렉토리 경로 |
| --threshold | Confidence 가 0.5보다 클 경우 boundary-box 를 수용한다는 의미 |

**[표 6] 테스트 명령어 요소**



**[Figure 21] test data 를 넣은 결과**

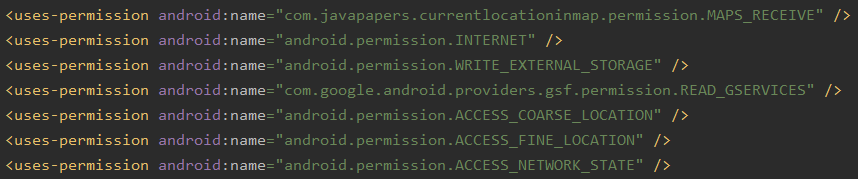
훈련한 모델에 훈련할 때 쓴 이미지가 아닌 테스트 이미지를 넣어 테스트 한 결과이다. 신호가 초록불인 상태를 올바르게 인식한 것을 확인할 수 있다.

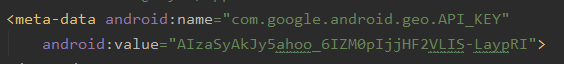
**4.3 Application**

4.3.1 Google MAP API

Google MAP API는 지도를 생성하고, GPS를 이용하기 위하여 사용된다.

4.3.1.1 Google MAP API 등록

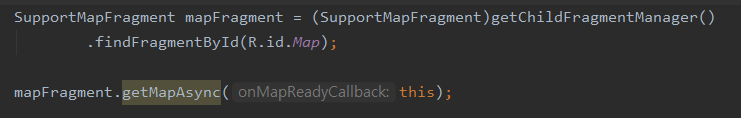




**[Figure 22] Google MAP API meta-data 코드**

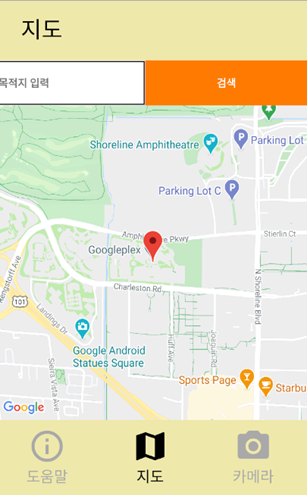
MAP 기능을 이용하기 위해서는 구글 클라우드 플랫폼에 안드로이드 프로젝트를 등록해야 한다. 위의 그림과 같이 안드로이드 내에서 인터넷 사용을 허가해야 하고, 구글 클라우드 플랫폼에서 발급받은 App-Key를 입력해야 한다.

4.3.1.2 Google Map 생성



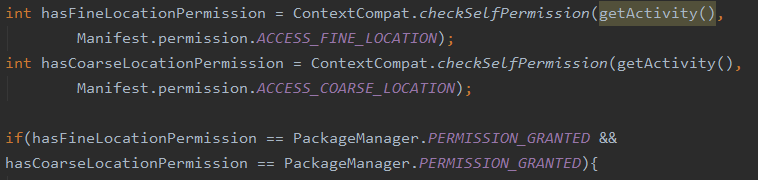
**[Figure 23] Google Map API 코드**

어플리케이션 내에 지도를 생성하기 전에 Activity가 아닌 Fragment에서 지도를 생성하기 때문에 mapFragment 변수와 지도를 띄울 layout(R.id.Map)을 연결한다. 그 후, getMapAsync()와 onMapReady()를 호출하여 지도를 생성한다.



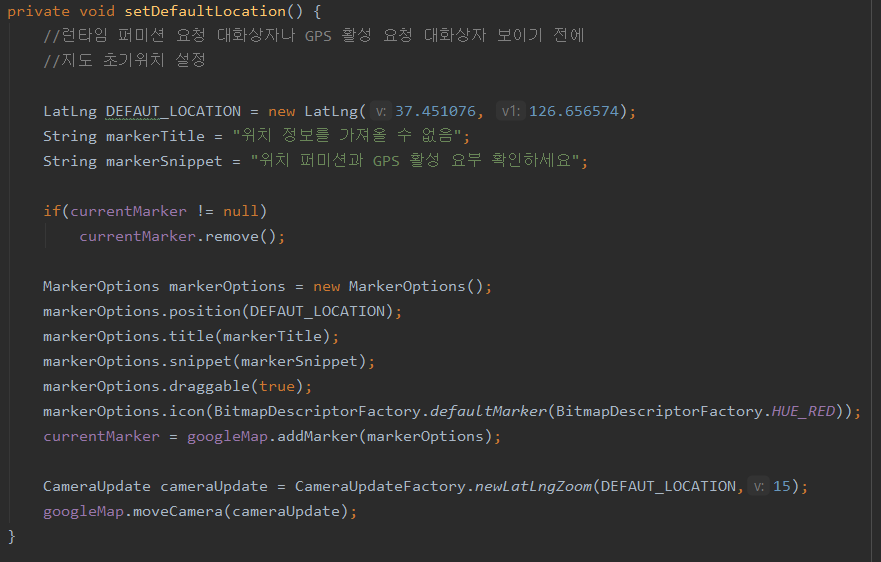
**[Figure 24] 어플리케이션 내에 Google Map**

4.3.1.3 GPS



**[Figure 25] Permission 확인 코드**

사용자의 위치정보를 가져오기 위해서는 위치 퍼미션이 필요하다. 그를 확인하는 코드이다. 퍼미션이 있다면 현재위치를 업데이트 하는 startLocationUpdates()가 호출된다. 퍼미션이 없다면 퍼미션을 요청한다.

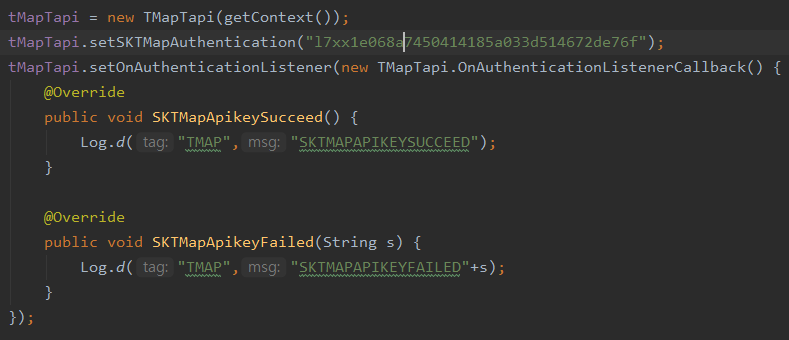
 **[Figure 26] setDefaultLocation()**

위의 코드는 퍼미션이 없거나 GPS 기능이 활성화되지 않은 경우의 위치를 DEFAULT\_LOCATION으로 설정한다. 여기서 DEFAULT\_LOCATION은 인하대학교 후문으로 설정하였다. MarketOptions를 이용하여 Marker를 지도에 추가해준다. 퍼미션이 있고 GPS 기능이 활성화되어 있다면 setDefaultLocation() 가 아닌 setCurrentLocation()가 호출되어 현재 위치를 나타내는 Marker가 지도에 추가된다.

4.3.2 T map API

T map API는 목적지 검색과 경로를 탐색하기 위해서 사용된다.

4.3.2.1 T map API 등록



**[Figure 27] T Map API 연동 코드**

T Map API를 이용하기 위해서는 SK open API 플랫폼에서 프로젝트를 등록해야 한다. 그 후 발급받은 App-Key를 위의 코드처럼 등록해준다.

4.3.2.2 장소 찾기

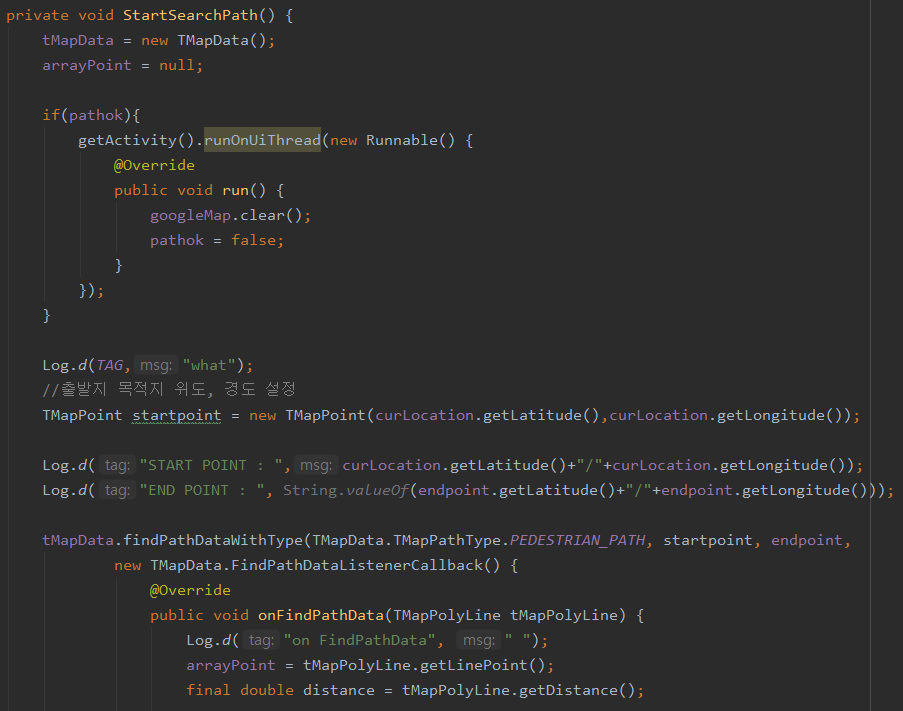
입력된 데이터를 받아 장소의 정보를 가져온다.

**[Figure 28] FindPOI()**

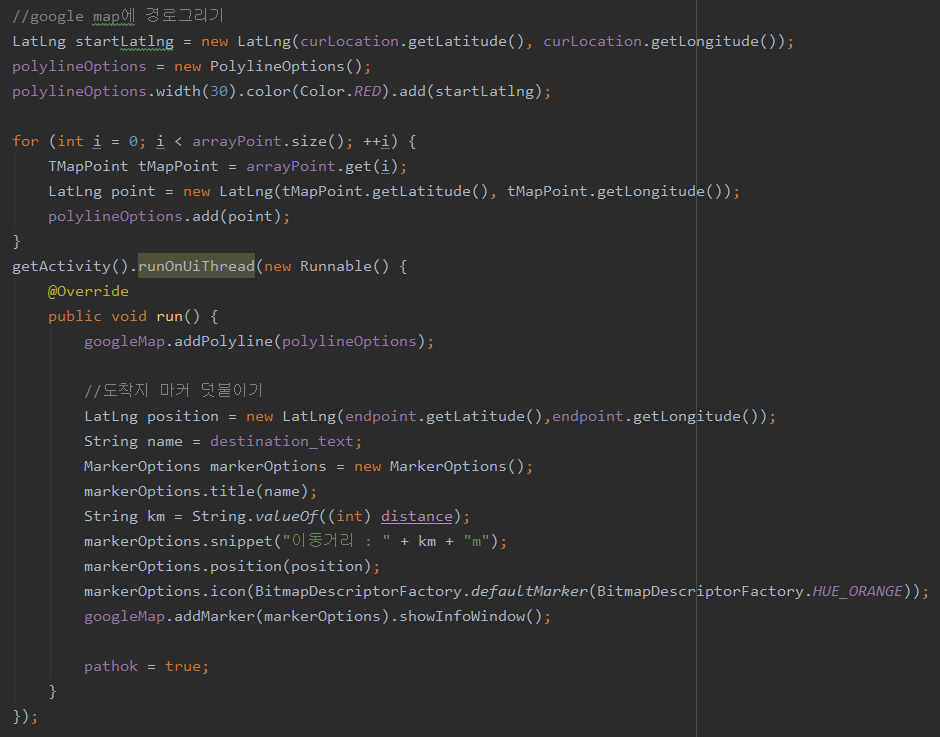
위의 코드는 POI 이용을 위한 것이다. 음성으로 입력 받은 목적지의 명칭과 일치하는 장소의 위도와 경도를 endpoint에 저장한다. 그리고 경로를 찾기 위한 StartSearchPath()를 호출한다.

4.3.2.3 경로 탐색 및 그리기

FindPOI()에서 찾은 endpoint의 마커, 현재 위치의 위도와 경도를 저장한 startpoint의 마커 그리고 탐색한 경로를 Google Map에 그린다.



**[Figure 29] StartSearchPath() 1**



**[Figure 30] StartSearchPath() 2**

TMapData 객체의 findPathDataWithType()를 이용하여 보행자를 위한 경로를 탐색한다. 1번 코드에서 탐색한 경로를 2번 코드 중 googleMap.addPolyline()으로 Google Map에 그려준다. 그리고 startpoint와 endpoint에 저장된 위도와 경도를 googleMap.addMarker()을 이용하여 출발지와 목적지 마커를 생성한다. 구분을 위해 출발지 마커는 빨간색으로, 도착지 마커는 주황색으로 설정하였다.

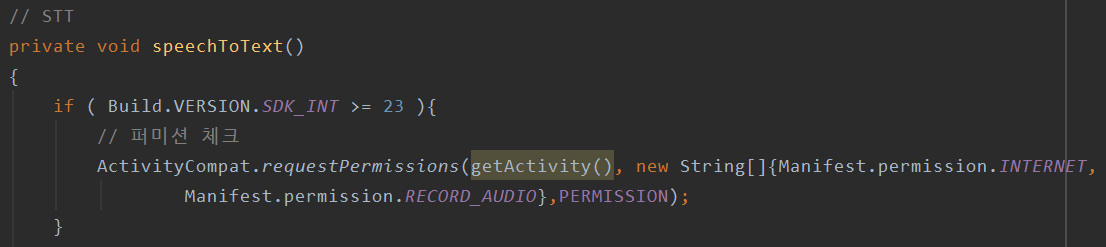
.



**[Figure 31] 경로 탐색 결과**

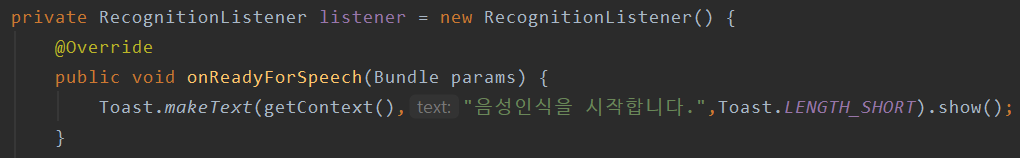
4.3.3 STT API

사용자의 음성 입력을 텍스트로 변환하기 위하여 사용된다. 해당 API는 안드로이드에서 제공한다.

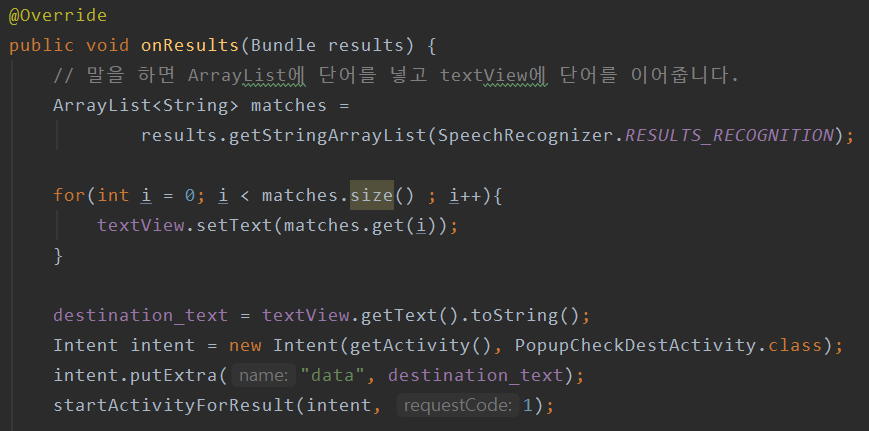


**[Figure 32] 퍼미션 체크**

우선, 사용자의 음성을 입력 받기 위해 퍼미션을 확인한다.



**[Figure 33] 음성 입력 후 텍스트 변환 1**



**[Figure 34] 음성 입력 후 텍스트 변환 2**

RecognitionLister()를 통해 사용자의 음성을 입력 받는다. 이후, 이를 텍스트로 변환한다. 해당 기능을 이용하여 사용자가 원하는 목적지를 음성으로 입력하여 설정할 수 있도록 한다.

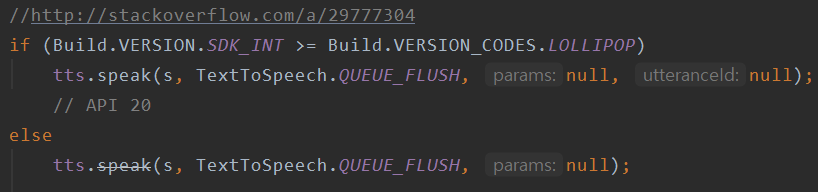
4.3.4 TTS API

텍스트로 입력된 값을 사용자에게 음성으로 제공하기 위하여 사용한다. 해당 API 또한 안드로이드에서 제공한다.



**[Figure 35] TTS 초기 설정**

TTS를 시작하기 위해 초기 설정을 한다.



**[Figure 36] TTS.speak()**

출력하고자 하는 텍스트 값을 tts.speak() 에 파라미터값으로 지정하여 해당 텍스트를 음성으로 출력한다. 해당 기능을 이용하여 사용자가 입력한 목적지 값이 맞는지 확인해주거나, 컴포넌트 한 번 선택 시 해당 컴포넌트 텍스트를 음성으로 출력한다.

**5. 세부 추진 계획 및 일정**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **구분** | **세부 구분** | **추진 내용** | **추진일정(월)** | | | |
| **9** | **10** | **11** | **12** |
| 설계 | 개발 계획 & 환경 | 주변 탐색, 시스템 구조 설계 및 구성도 작성 |  |  |  |  |
| 개발 | 앱 UX&UI 구현 | 제스처 인식 기능 및 어플 디자인 |  |  |  |  |
| DB 구현 | DB 생성 및 데이터 저장 |  |  |  |  |
| API 적용 | MAP API와 SST & TTS API 적용 및 기능 구현 |  |  |  |  |
| YOLO 모델 적용 | Darkflow 이용하여 Object Detection |  |  |  |  |
| 테스트 | 통합 테스트 | 시나리오 기반으로 개발자의 의도 및 프로토 타입 일치 여부 확인 |  |  |  |  |
| 디바이스 테스트 | 길거리에서 시뮬레이션으로 기능 작동 확인 |  |  |  |  |
| 유지보수 | 최적화 & 디버깅 | 오류 여부 지속적인 확인 & 최적화 시도 |  |  |  |  |
| 종료 | 발표 | 최종 발표 |  |  |  |  |

**[표 7] 세부 일정 및 추진 계획**