2022 Capstone Design

최종 보고서

팀명	아리아리			
담당 교수	황기태 교수님			
팀원	이름	학번	연락처	이메일
	임슬아(팀장)	1891286	010-8605-4354	jse05150@naver.com
	김현학	1591025	010-2305-9718	07ily@naver.com
	이승진	1491006	010-2007-0358	tmdwls1134@naver.com
	이한별	1491030	010-4544-6645	byeolstar12@naver.com
Project	https://github.com/zbti84/WearOS-Navi			

목 차

<제목 차례>

1. 프로젝트 수행 목적

1.1 프로젝트 정의

'지도 API' + 'GPS' + '마이크 기능' + '진동 기능'을 이용하여 음성안내 및 시각적인 정보 없이 진동 표현으로 길을 안내할 수 있는 시각장애인용 워치 전용 길 안내 앱과 보행자의 위치변화, 입력한 목적지 등 앱에서 이루어진 모든 행동을 실시간으로 볼 수 있는 모니터링 웹 서비스

1.2 프로젝트 배경

화면을 보며 길을 따라가야 하는 길 찾기 앱은 시각장애인들이 사용하기에는 많은 사용에 어려움이 있고 만약 이어폰을 낀다면 길 안내 음성은 잘 들리지만 오히려 주변 소리가 잘 안 들리게 되는 경우가 많다. 또한, 휴대전화를 손에 들고 다니는 점에 대한 불편함도 있는데 시각장애인들은 앞에 있는 장애물을 피하고자 보조대로 짚어가며 다니곤 한다. 이럴 때 다른 손에 휴대전화 까지들 고 있다면 없을 때 보다 많이 불편하게 느끼는 경우가 많다. 이를 해소하기 위해 보행 간 두 손이 자유로울 수 있는 스마트 워치를 사용하였고 내 진동으로 길을 안내하는 방법을 선택하였다.

하지만 워치만으로는 길 안내가 잘 되고 있는지 어떻게 이루어지고 있는지 보호자 입장에선 모를 수밖에 없고 불안함에 앱을 사용하기 힘들 것이다. 이를 보완하기 위해 보호자가 보행자의 위치변화나 입력한 목적지, 안내 경로 등 앱에서 이루어지는 모든 행동을 실시간으로 볼 수 있게 하도록 모니터링 시스템을 만들었다. 보행자가 워치에 목적지를 입력하면 모니터링 시스템에는 입력된 목적지와 t map에서 받은 경로 2~4가지가 지도와 함께 보이고 그 후 알고리즘으로 선택된 안내 경로와 실시간으로 받아오는 보행자의 현재 위치가 보이며 워치에서 일어나는 모든 일들을 웹에서도 보여준다.

1.3 프로젝트 설명

가. 음성 인식

KaKao Newtone API에서 제공하는 사용자의 음성을 텍스트로 변환하는 음성 인식 기능을활용하여, 사용자의 음성 사용을 통한 목적지 선정이 가능하도록 한다.

API에서 음성 데이터는 Mono channel, 16000 Hz samplerate, 16 bit depth인 RAW PCM 포맷만 지원한다. 음성 데이터는 POST로 전송하고 요청이 성공하면 multipart/form-data 타입으로 응답을 받으며, 요청 결과에 대한 헤더들을 포함한다.

음성을 PCM 데이터로 저장하기 위해 AudioRecorder객체를 사용한다. AudioRecorder 객체의 오디오 포맷을 api에서 지원하는 포맷과 일치하게 세팅한다. 녹음 중 화면이 멈춰 보이는 것을 방지하기 위해 녹음은 thread 안에서만 이루어진다. 데이터는 byte 배열 변수 안에 저장한 후 녹음이 끝나면 api에 변수를 보내준다.

나. 진동으로 방향 표현

타겟층이 시각장애인인 만큼 UI와 음성을 최소화하고 싶어 길 안내를 시작하고 나서는 모든 것

을 진동으로 표현할 수 있게 구현하였다.

방향	진동 수	버튼 클릭	음성
직진	약하게 주기적으로 진동		
왼쪽 방향	진동 1번		
오른쪽 방향	진동 2번		
옳은 방향 도착	진동 멈춤		
분기점	각도차에 따라 진동세기 다르게 줌		
위험요소	세게 3초 진동		
목적지 입력		버튼 두번	목적지를 입력해주세요
목적지 안내			000으로 안내를 시작합니다
목적지 도착	진동 1번 후 음성안내		목적지에 도착했습니다

-진동표-

우선 직진 시 사용자가 계속 직진해야 한다는 것을 인식할 수 있게 주기적이지만 낮은 세기로 진동을 주었고

분기점을 마주쳤을 때 방향을 알려주는 것이 문제였는데 처음엔 그저 좌회전, 우회전만 알려주면 된다고 생각했지만 분기점은 온전히 90도의 좌회전, 우회전만 있는 것이 아니고 여러 각도로 나누어져 있는 경우도 있기 때문에
워치의 나침반 센서를 이용하여 사용자가 보는 방향과 목표 방향의 각도 차를 구해 각각 진동 세기를 다르게 주어올바른 방향으로 각도를 맞출 수 있게 하였다. 나침반 센서는 워치를 정확히 수평으로 들어야 한다.

사용자가 보는 방향을 0도라고 한다면 총 360도 이므로 좌측, 우측으로 180도씩 나눌 수 있는데 이를 고려해 좌측의 180도 범위 내이면 진동을 한 번씩 주고 우측의 180도 범위 내이면 진동을 두 번씩 준다.
그리고 사용자의 방향이 바뀔 때마다 맞춰야 하는 목표 방향의 각도 차를 구해 그 값이 낮아질수록 , 즉 방향이 맞춰질수록 진동의 세기를 점점 커지게 하여 맞춰졌을시 진동을 끊어 방향이 맞았음을 알려준다.

여기서 목표 방향의 각도를 구하는 식은 다음과 같다.

```
private fun getAngle(lat1: Double, lon1: Double, lat2: Double, lon2: Double): Float {
    var y1 = lat1 * Math.PI / 180
    var y2 = lat2 * Math.PI / 180
    var x1 = lon1 * Math.PI / 180
    var x2 = lon2 * Math.PI / 180

    var x = Math.sin(x2 - x1) * Math.cos(y2)
    var y = Math.cos(y1) * Math.sin(y2) - Math.sin(y1) * Math.cos(y2) * Math.cos(x2 - x1)
    var rad = Math.atan2(x, y)
    var bearing: Float = ((rad * 180 / Math.PI + 360) % 360).toFloat()
    return bearing
}
```

-목표 각도-

현재 위치의 위도, 경도를 x1, y1으로 받고 다음 위치의 위도, 경도를 x2, y2로 받은 뒤 코틀린 함수 Math를 사용하여 (x1,y1)와 (x2,y2)의 기울기에 따른 각도를 구하였다.

마지막으로 위험 요소를 마주쳤을 때는 강한 세기로 3초 정도 진동을 주어 위험을 알려주고
목적지에 도착했을 때는 진동으로만 알려준다면 사용자가 방향 안내로 잘 못 알 수도 있기 때문에 음성을 함께 내보내며 도착을 알려준다.

다. GPS를 이용한 길 안내

목적지 검색의 경우 사용자가 목적지에 대한 정확한 명칭이나 주소를 알고 있는 경우에는 정상적으로 사용할 수 있었으나, 정확한 지점 명을 모르거나 같은 이름의 건물이 다른 지역에도 있는 경우 사용에 어려움이 있었다. 따라서 현재 위치를 기준으로 2km 안에 있는 목적지만 검색 결과에 나올 수 있게 조정했고, 항상 나와 가장 가까운 목적지를 검색할 수 있게 하였다. 예를 들어 맥도날드 매장을 찾고 싶을 때 근처에 있는 맥도날드 매장의 이름을 모를 때 그저 '맥도날드'로 검색하는 것으로 내 근처에 있는 맥도날드 매장으로 안내할 수 있게 했다.

다음은 좌표에 관한 것이다. t map API에서 받은 원본 데이터에서 좌표와 좌표 사이의 거리가 너무 멀기 때문에 경로 안내에 사용할 수가 없었다. 또한, 경로에 대한 좌표만 제공하는 것이 아니라 좌표에 대한 설명, 예를 들어 '횡단보도 앞', '좌회전' 등의 정보도 같이 제공하기 때문에 필요한 정보만 따로 추출할 필요가 있었다.

경로 세분화에 대해서 말하자면 만약 좌표와 좌표 사이의 거리가 60m라면 양쪽을 30m로 나눌 수 있는 중간좌표를 얻고 다시 30m를 15m로 쪼개는 방식으로 t map에서 얻는 좌표를 세분화하여 저장했다. 하지만 이렇게 좌표를 세분화하게 되면 기존에 좌표가 가지고 있는 분기점 정보, 신호등 정보 등의 의미가 완전히 사라지기 때문에 분기점 정보만 가지고 있는 새로운 배열도 만들어 사용했다.

다음으로 경로 안내 부분이다.

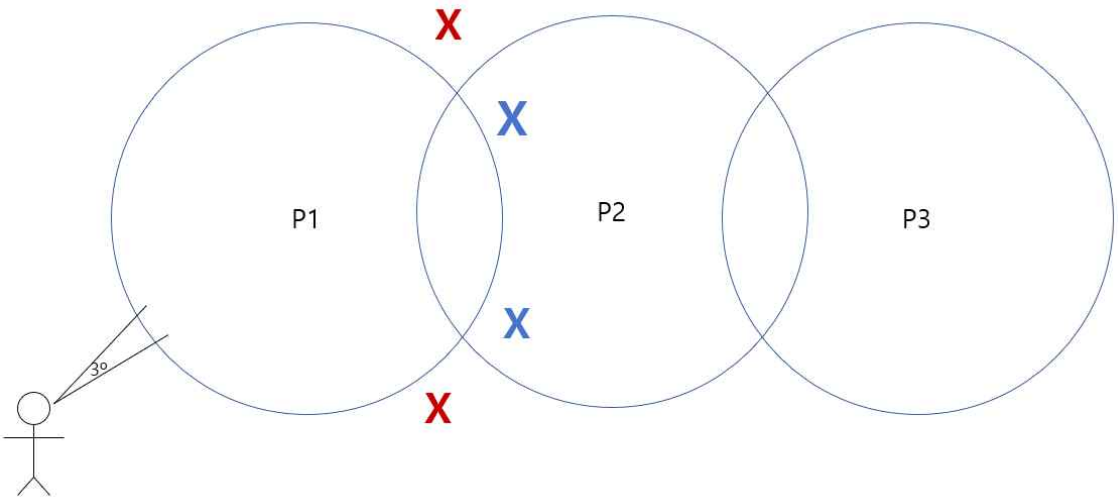


그림 1 - 경로 안내

지금까지 현재 위치에서 목적지까지의 세분화 된 좌표를 얻었다. 하지만 시각장애인에게 진행 방향을 알려주지 않으면 엉뚱한 방향으로 움직일 가능성이 있다. 따라서 현재 위치에서 첫 번째 좌표(P1)로 가는 방향을 먼저 알려주었다. 방향을 알려주는 방식은 '나. 진동으로 방향 표현'에서 설명한 방식을 이용했다.

이후 경로 이탈은 거리 계산을 통해 구현했다. GPS 정보는 0.5초마다 갱신된다. 갱신할 때마다 현재 위치와 첫 번째 좌표(P1), 현재 위치와 두 번째 좌표(P2) 사이의 거리를 계산한다. 만약 현재 위치와 첫 번째 좌표(P1)의 거리가 미리 설정한 오차를 벗어났을 때, 현재 위치와 두 번째 좌표(P2) 사이의 거리 또한 오차를 벗어났다면 경로 이탈로 판단했다(빨간색 X표시). 하지만 벗어나는 즉시 경로 이탈로 판단하는 것은 실제 사용에 있어 매우 불편했다. 그래서 약 10초 이상 지속해서 벗어났을 때 경로 이탈로 판단했다. 실제로 경로 이탈이 발생했다면, 사용자에게 '멈춤' 신호를 주고 그 현재 위치로부터 목적지까지의 경로를 재탐색했다. 따라서 경로 이탈 지점은 새로운 출발지가 되는 것이다.

하지만 현재 위치와 첫 번째 좌표(P1)의 거리가 미리 설정한 오차를 벗어났을 때, 현재 위치와 두 번째 좌표(P2) 사이의 거리가 오차를 벗어나지 않았다면(파란색 X표시), 경로를 잘 따라가고 있다고 판단했다. 이후 두 번째 좌표(P2)와 세 번째 좌표(P3)를 통해 경로 이탈을 판단한다.

마지막으로 분기점 및 위험 요소에 대한 설명이다.

번호	정보	번호	정보	번호	정보	번호	정보
11	직진	18	2시 방향 우회전	129	계단+경사로 진입	214	8시 방향 횡단보도
12	좌회전	19	4시 방향 우회전	200	출발지	215	10시 방향 횡단보도
13	우회전	125	육교	201	목적지	216	2시 방향 횡단보도
14	U-turn	126	지하보도	211	횡단보도	217	4시 방향 횡단보도
16	8시 방향 좌회전	127	계단 진입	212	좌측 횡단보도		
17	10시 방향	128	경사로 진입	213	우측		

	좌회전				횡단보도		

표 1 - tmap 제공 회전 정보

t map에서는 표 1과 같은 정보를 좌표와 함께 제공한다. 이 중에서 125~129번의 정보는 시각장애인에게 위험 요소라고 판단했다. 안전한 길 알고리즘을 통해 이런 위험 요소가 적은 길을 선택하겠지만 그래도 이러한 길이 있다면 위험 요소 앞에서 '위험 요소' 신호를 주어 조심할 수 있도록 유도했다.

그 외 방향 전환과 관련된 정보는 모두 하나로 묶어 '분기점' 신호로 통합하여 '나. 진동으로 방향 표현'에서 설명한 방식으로 방향 전환이 이루어질 수 있도록 안내했다.

라. 안전한 길 알고리즘

1. SK open API 회원가입

2. < T-Map API - Web service - Guide - 경로안내 >
   순서에 맞춰 진행하고 AppKey 발급

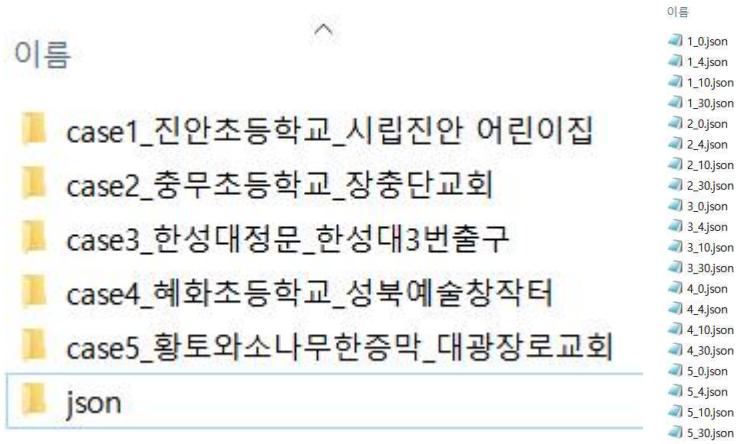
3. < Web service - Docs - 경로안내 - 보행자 경로안내 >
   REST API에 대한 내용이 명시되어 있음.

4. Postman을 통한 API 활용
   a. [Postman 다운로드]
   b. POST
      i. Params: Query Params
         1. Version:1
         2. Appkey: *(발급받은 AppKey)*
      ii. Body:
         1. startX: (출발지 경도)
         2. startY: (출발지 위도)
         3. endX: (도착지 경도)
         4. endY: (도착지 위도)
         5. startName: (출발지 이름)
         6. endName: (도착지 이름)
         7. searchOption: (경로 탐색 옵션, 0:추천(기본값), 4:추천+대로우선, 10:최단, 30:최단거리+계단제외)

   ** 이 밖의 내용은API Documentation 링크 참고.

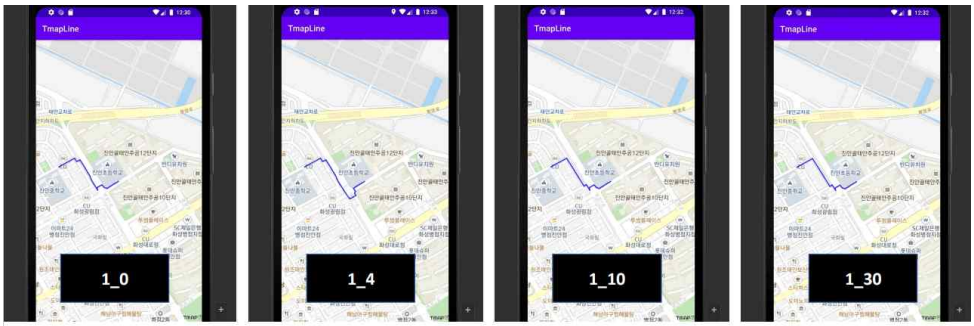
5. 원하는 위치의 위도/경도 정보는 Google 지도 앱 등을 활용

6. Python을 활용한API 호출 결과 저장



7. Python을 활용한 JSON 파일로 저장된 경로 정보 조정

결과 예시



From: 진안초등학교
To: 시립진안 어린이집

- 0: 1.815 (육교 포함, 도로변 보행로 선택)
- 4: 43.294 (2개의 횡단보도 포함)
- 10, 30: 1.714 (육교 포함, 블록 내부 보행로 선택)

Python을 활용한 가중치 조정 과정

1. 경로 데이터는 크게 Point와 LineString으로 구분되며,
   각 자료형 별로 내포한 데이터 종류가 다르다.

2. Point는 실제 경로 안내동안 회전(방향 전환)의 기점에 대한 자료이고,
   LineString은 사용자가 실제로 이동해야 하는 경로를 두 점으로 정의되는 직선의 집합에 대한 자료이다.

3. 처음에 나오는 Point는 특별히 시작점으로서 totalDistance(단위: m) 정보를 가진다.

4. 고려한 변수는 크게 다음 4가지로 구분 가능하다.
   a. totalDistance: 경로 총 길이 (단위: m)
   b. distance: 구간 거리(단위 : m)
   c. roadType: 도로 타입 정보
   d. facilityType: 구간의 시설물 정보

5. 주어진 각 구간의 정보를 토대로 부분합(partialScore)을 내고,
그에 대한 총합(totalScore)으로 벌점을 매겨서 최종적으로 벌점이 가장 낮은 경로를 선택하는 방식을 채택했다.
    a. (각 구간의 distance) / totalDistance의 계수를 적용하여, 전체 경로 길이에 대한 비중을 고려하여 매 구간의 부분합을 적용한다.
    b. facilityType
        i. IF (facilityType == 11)
        일반보행자도로를 의미하고, 이 경우 roadType만 고려한다.
        ii. IF (facilityType == 15)
        횡단보도를 의미하고, 각 항목에 대한 가중치를 부여할 때, 반드시 보다 적은 수의 횡단보도를 통한 경로를 선택하도록 접근했다. ('시각 장애인에게 안전한 길'에 대한 정의 참조)
    c. roadType
        i. IF (roadType == 21 || 23)
        차도와 인도가 구분되어 있거나, 차량 통행이 불가한 보행자 도로 나머지 두 경우에 비해 안전하다고 평가
        ii. IF (roadType == 22 || 24)
        차도와 인도가 구분되어 있지 않은 보행자 도로 또는 보행자 도로가 아닌 일반 도로

Python Code

```
179 for n in range(0, len(jsonList)):
180     cur = jsonList[n]
181     features = cur["features"]
182 
183     nTurnPoints = nLineString = totalScore = 0
184 
185     for i in range(0, len(features)):
186 
187         objName = "properties"
188         obj = features[i][objName]
189 
190         if i == 0:
191             totalDistance = obj["totalDistance"]
192 
193         if features[i]["geometry"]["type"] == "Point":
194             turnType = obj["turnType"]
195             if 211 <= turnType and turnType <= 217:
196                 nTurnPoints += 1
197 
198         else:
199             nLineString += 1
200             roadType = obj["roadType"]
201             distance = obj["distance"]
202             try:
203                 facilityType = int(obj["facilityType"])
204             except ValueError:
205                 facilityType = 0
206 
207             totalScore = totalScore + road2score(totalDistance, distance, facilityType, roadType)
208 
209     print(n,"totalScore",nTurnPoints, totalScore)
210 
```

** line 193 ~ 196은 횡단보도 분기점에 대한 정보를 식별하는 것인데,
구간의 facilityType 정보로 식별 가능하므로 실제 고려되지는 않는 정보이다.

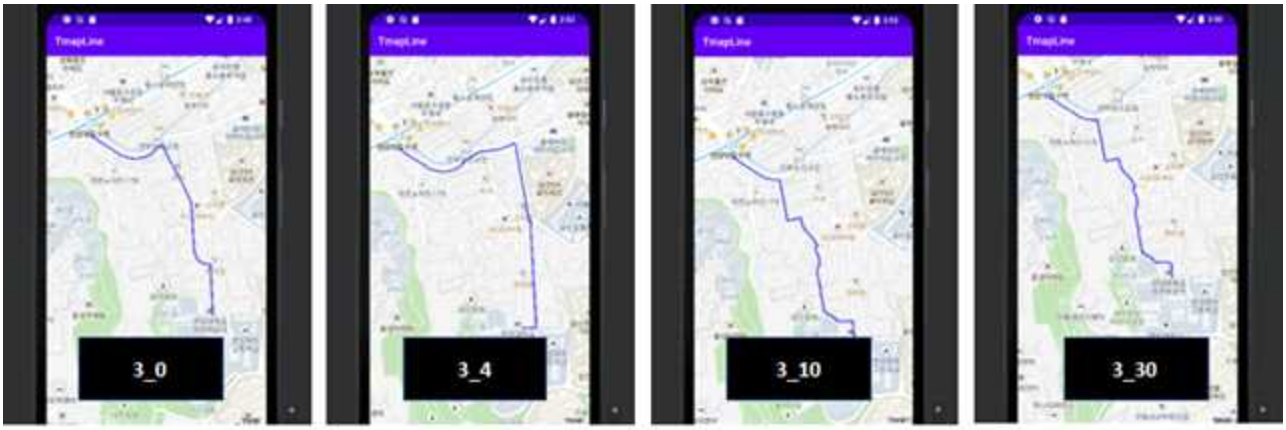
< 결과 >

From: 진안초등학교
To: 시립진안 어린이집

- 0: 1.815 (육교 포함, 도로변 보행로 선택)
- 4: 43.294 (2개의 횡단보도 포함)
- 10, 30: 1.714 (육교 포함, 블록 내부 보행로 선택)

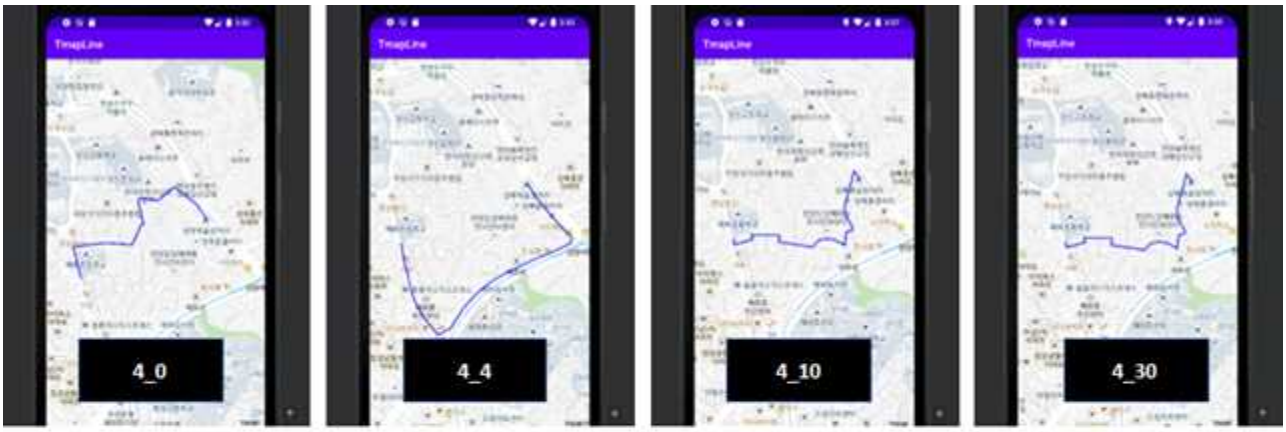
From: 충무초등학교
To: 장충단교회

- 0: 4.177 (육교 포함, 도로변 보행로 선택)
- 4: 110.932 (3개의 횡단보도 포함)
- 10, 30: 3.732 (육교 포함, 블록 내부 보행로 선택)



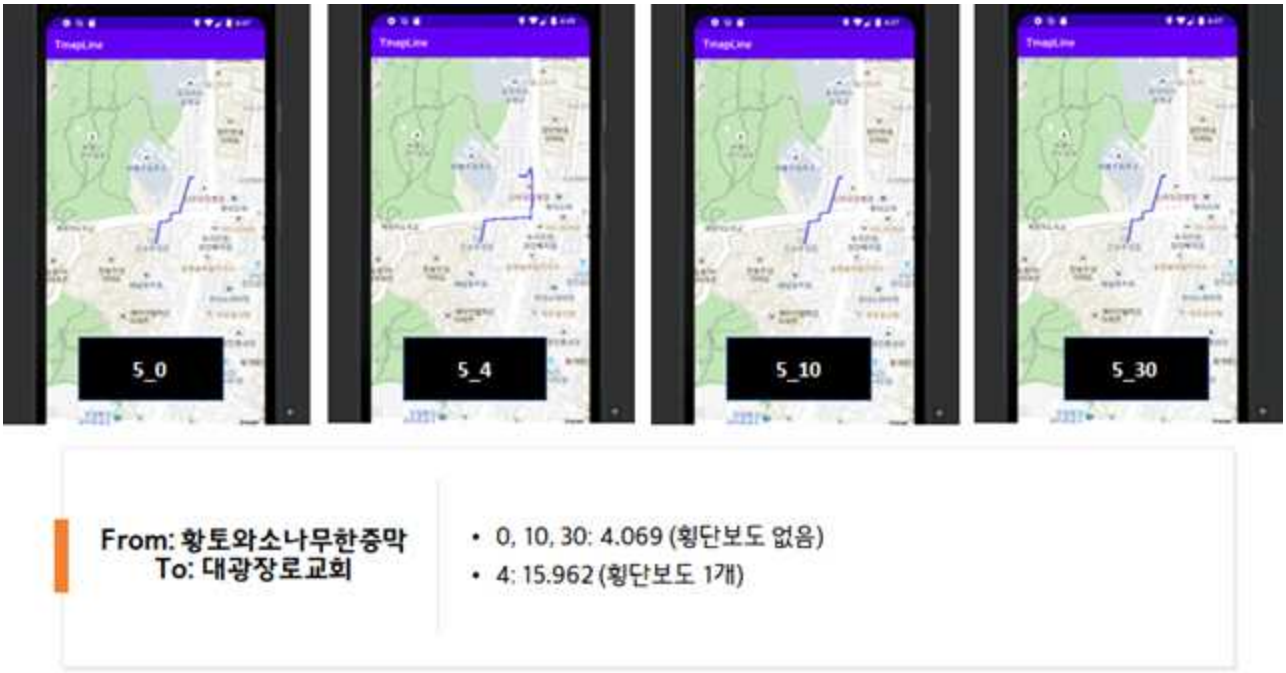
From: 한성대정문
To: 한성대3번출구

- 0: 2.374
- 4: 0.558
- 10,30: 6.835



From: 혜화초등학교
To: 성북예술창작터

- 0: 2.265
- 4: 1.370
- 10,30: 6.805



마. MQTT

워치에서 어떠한 일이 일어날 때 자동으로 웹페이지에 반응이 일어나도록 발행-구독 기반의 메시징 프로토콜을 사용하였다.

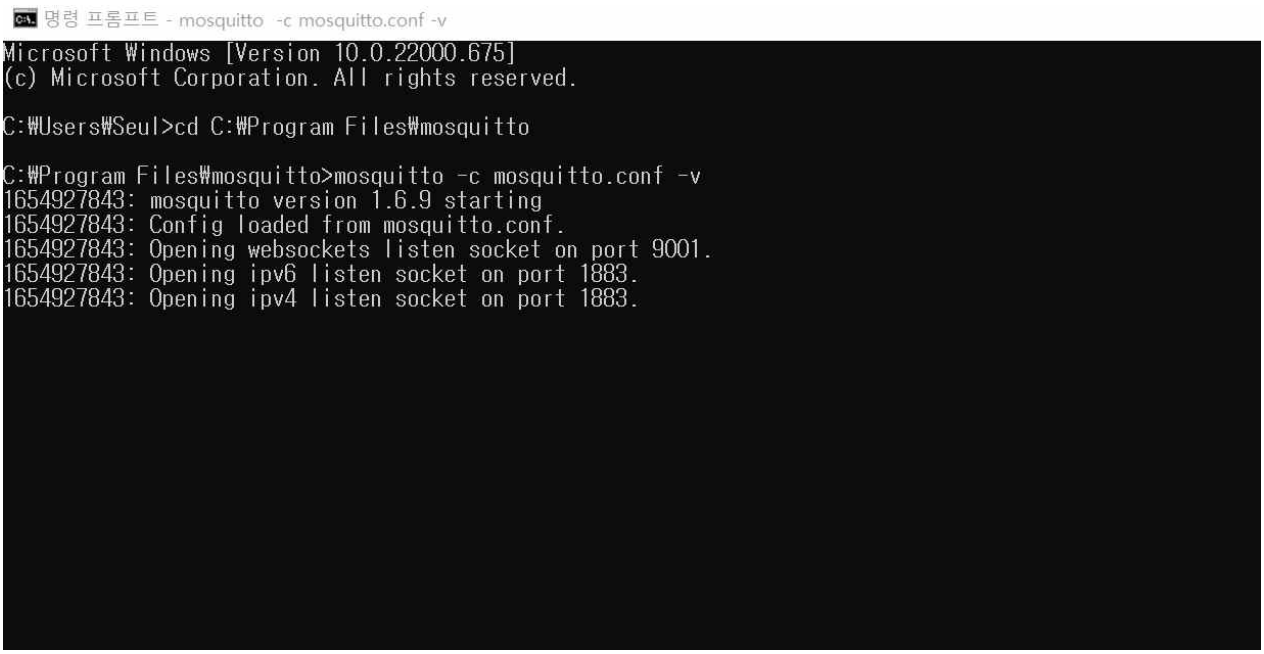
워치 - 워치에서 일어나는 일을 웹 화면에 띄우기 위해 메시지를 발행하는 publisher가 된다.

워치에서 publish하기 위해서는 먼저 mqttConnection을 수행해야 하는데 publish가 먼저 실행되는 것을 방지하기 위해 handler 함수를 사용하였다.

아래는 워치에서 발행하는 메시지

Topic 명	“Message”	
topic	“음성 녹음 중입니다.”, “목적지 : 창신역”, “목적지를 재입력 중입니다”,“분기점을 마주했습니다”, “방향 조정 중입니다” …	
vibe	“시작”, “끝”	진동의 시작과 끝을 알림
route	“위도1,위도1,../경도1,경도1,경도1!위도2,위도2,.../경도2,경도2,...!위도4,위도4,위도4...,경도4,경도4,경도4”	tmap에서 경도4가지를 받으면 하나의 문자열로 만들어 발행
route_res	“위도, 위도, 위도,.../경도, 경도, 경도...”	4가지 경로 중 안전하 길 알고리즘으로 선택된 경로를 문자열로 만들어 발행
current	“위도, 경도”	현재 위치가 갱신될 때마다 보낸다.

웹 - 워치에서 일어나는 일을 화면에 보여준다. 윈도우 환경 웹에서 mqttt프로토콜을 사용하기 위해 브로커가 1883포트 번호와 9001포트 번호를 둘 다 인식하게 한다.



topic들을 구독하여 워치에서 메시지를 보낼 때마다 웹페이지가 자동으로 바뀌게 한다. 앱 프로그램 속도와 웹 페이지가 갱신되는 속도가 일치하지 않아 워치에서 메시지를 발행하는 순서에 맞게 웹페이지가 바뀌지 않는 경우가 있다. 이를 방지하기 위해 topic을 최소화하고 워치내에서 handler 함수를 적절히 배치하였다.

웹페이지에 지도를 띄우기 위해 t map api를 사용하였다. (위도, 경도)쌍을 리스트로 만들어 지도를 그린다. 지도는 총 두 가지다.

1. 경로 4가지를 띄우는 지도
2. 안전한 길 알고리즘으로 선택된 경로와 현재 위치 마커를 찍는 지도

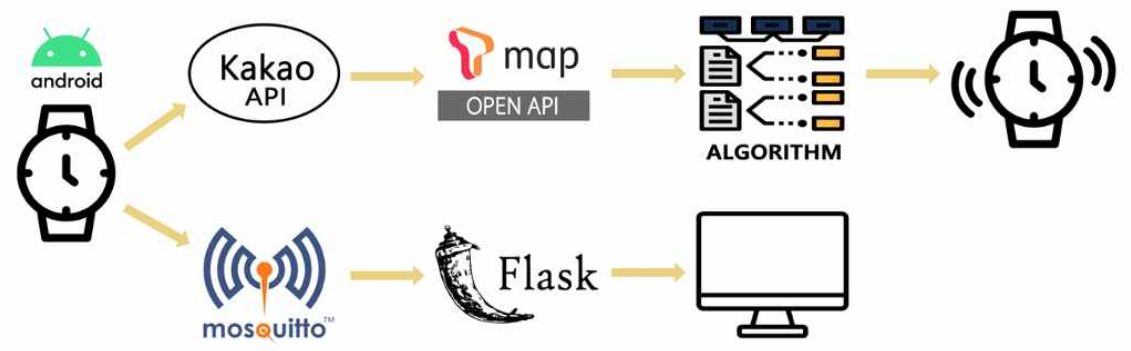
1번 지도에서 경로 4가지는 모두 다른 색으로 표현한다. 앱 특성상 가까운 목적지를 주로 사용하다 보니 경로는 평균적으로 2~3가지 경로가 나온다.

2번 지도에서는 안전한 길 알고리즘으로 선택된 경로를 띄우고 출발지점과 목적지점에 마커를 찍는다. 또한, 사용자의 현재 위치를 보여주는 마커를 찍는다. 마커는 사용자의 위치가 변할 때마다 바뀐다.

사용자가 방향조절을 할 때는 워치에서 진동이 울리는데 이때 웹페이지 내 워치 아이콘도 떨리는 애니메이션을 넣었다.

2. 프로젝트 개요

2.1 프로젝트 구조

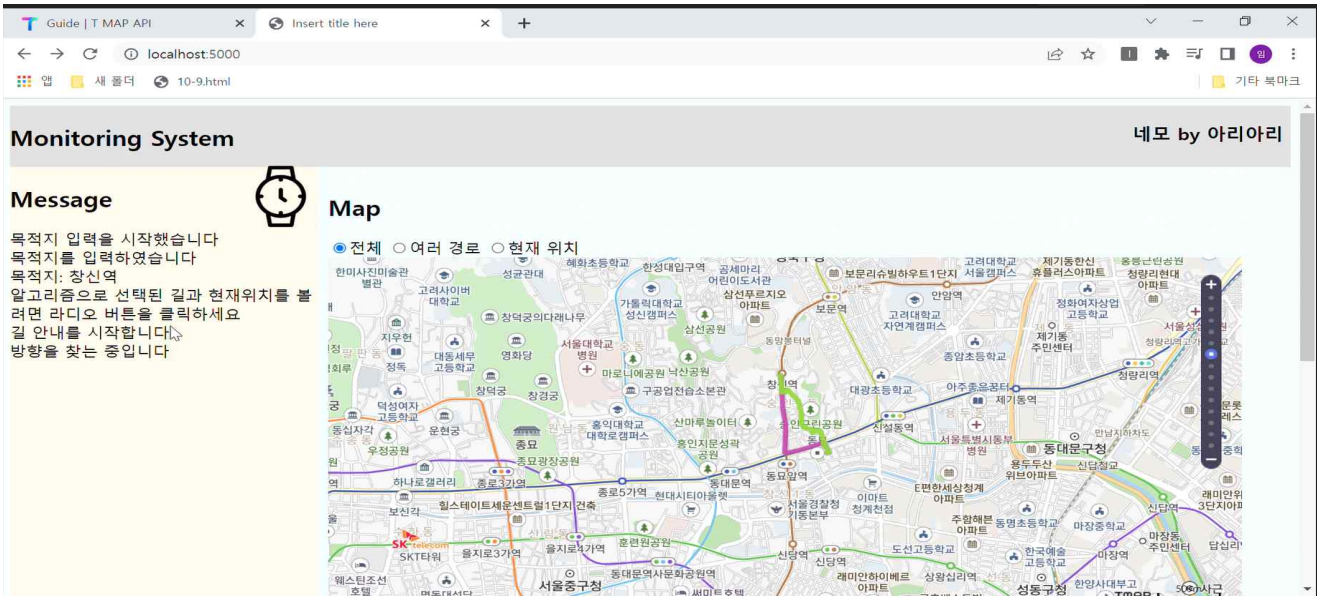


2.2 프로젝트 결과물

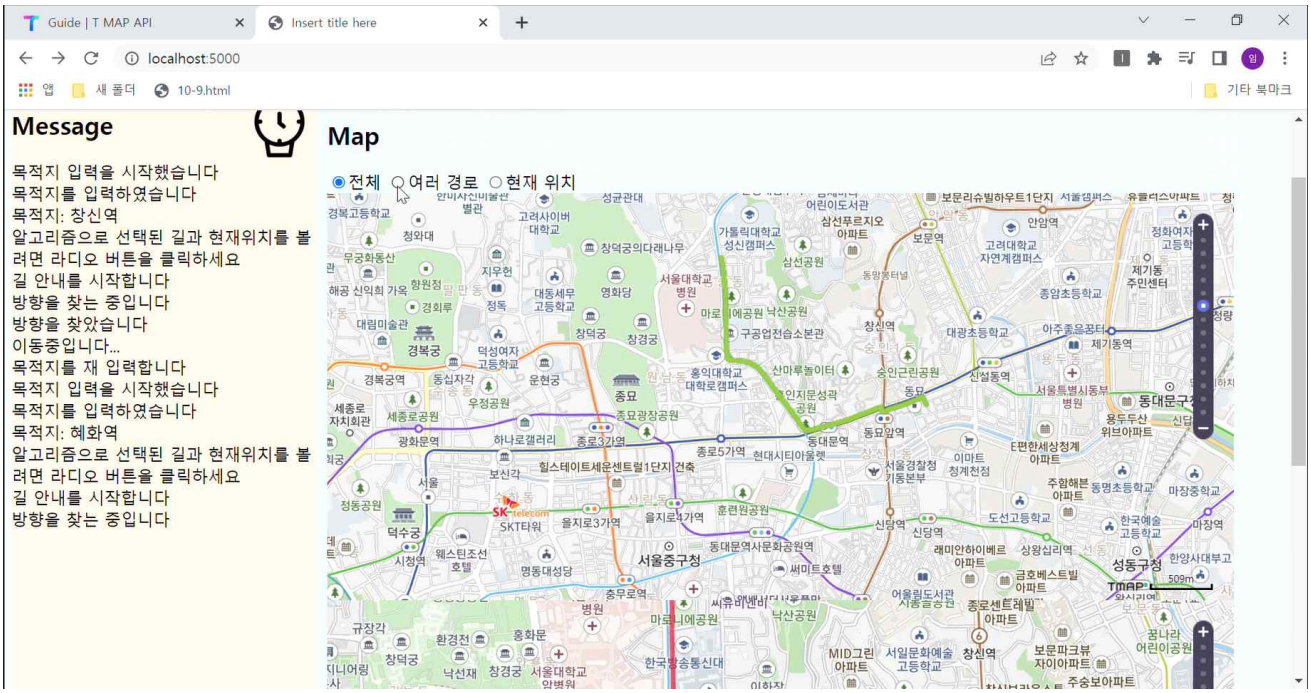
1. 사용자가 목적지를 입력

2. 경로 4가지를 가져오고 안전한 길을 선택

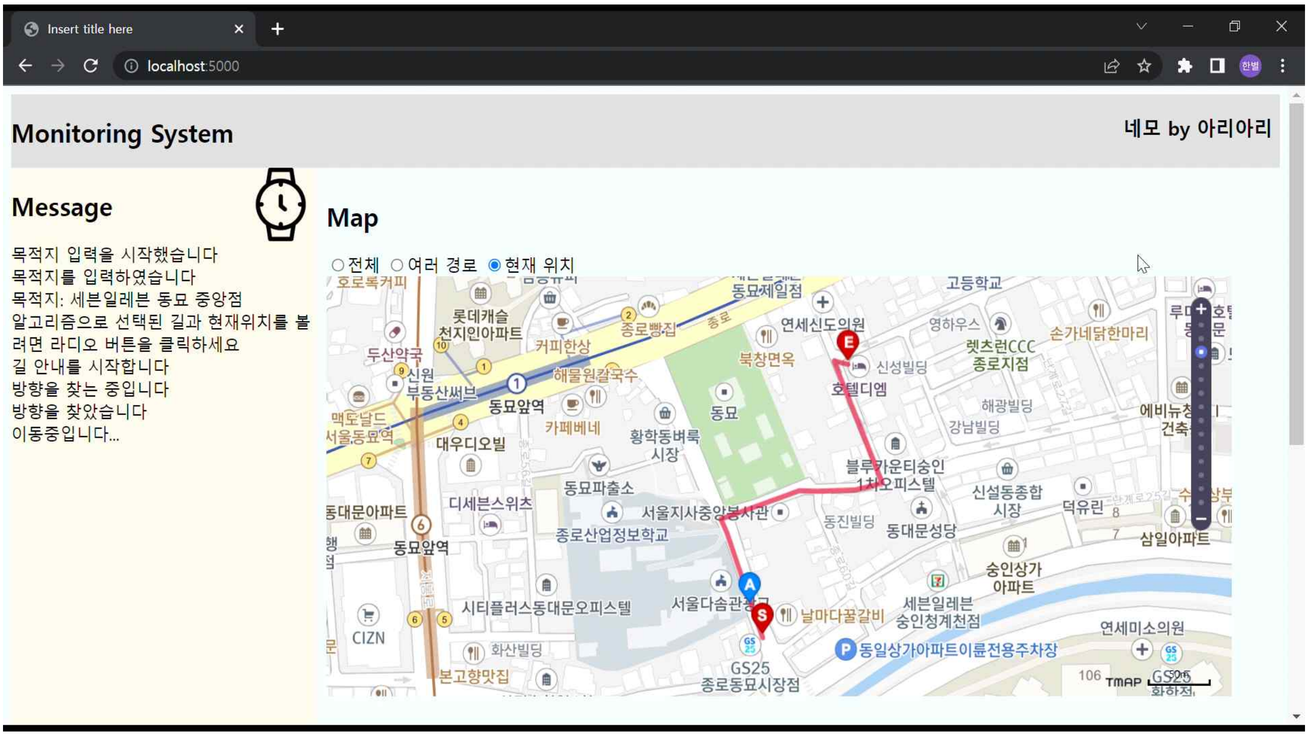
3. 길 안내중

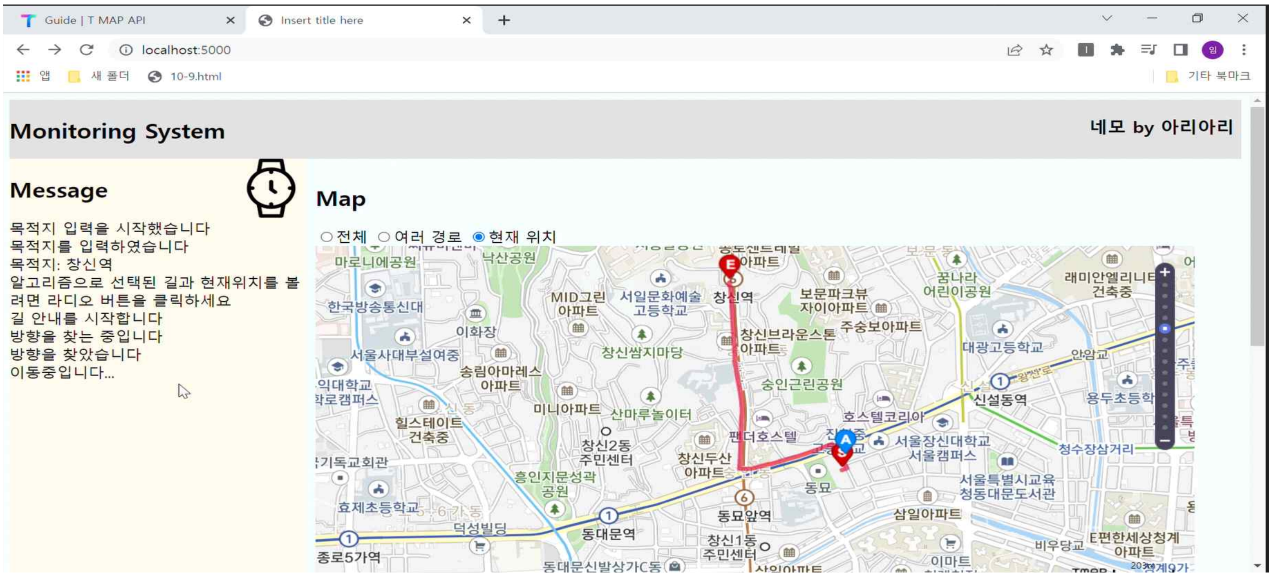


1. 목적지 입력 , 경로 2가지



2. 목적지 재 입력 , 지도 새로 띄움

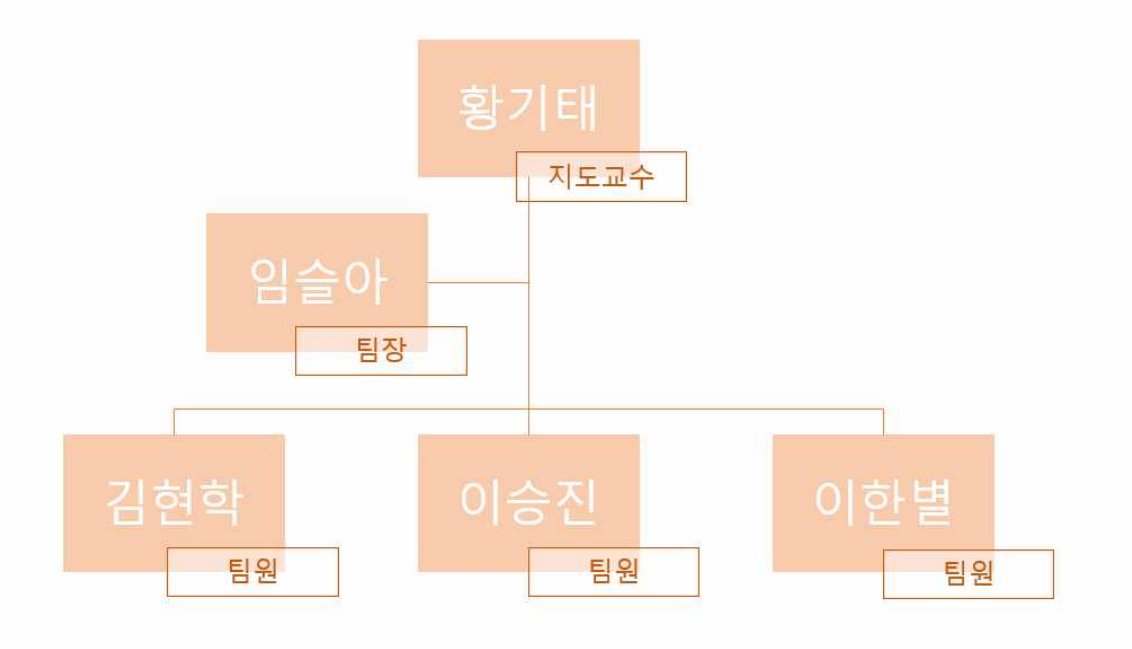

Insert title here
localhost:5000
Monitoring System
네모 by 아리아리
Message
목적지 입력을 시작했습니다
목적지를 입력하였습니다
목적지: 세븐일레븐 동묘 중앙점
알고리즘으로 선택된 길과 현재위치를 볼려면 라디오 버튼을 클릭하세요
길 안내를 시작합니다
방향을 찾는 중입니다
방향을 찾았습니다
이동중입니다...
Map
전체
여러 경로
현재 위치


Guide | T MAP API
Insert title here
localhost:5000
Monitoring System
네모 by 아리아리
Message
목적지 입력을 시작했습니다
목적지를 입력하였습니다
목적지: 창신역
알고리즘으로 선택된 길과 현재위치를 볼려면 라디오 버튼을 클릭하세요
길 안내를 시작합니다
방향을 찾는 중입니다
방향을 찾았습니다
이동중입니다...
Map
전체
여러 경로
현재 위치

3. 최종 경로 + 현재 위치

3. 프로젝트 수행 추진 체계 및 일정

3.1 각 조원의 조직도



3.2 역할 분담

임슬아 : STT, 모니터링/웹
이한별 : 클라이언트 , 모니터링/워치
김현학 : 안전한 길 알고리즘
이승진 : 클라이언트

3.3 주 단위의 프로젝트 수행 일정

4. 참고 자료

- https://developer.android.com/jetpack?gclid=CjwKCAjw7vuUBhBUEiwAEdu2pF3W836qii7bEks6kfTe9Tdgzq3_VuPj_Oo5lf3A7O4dLwgL7IWP0hoCPpgQAvD_BwE&gclsrc=aw.ds
- https://mqtt.org/
- https://developers.kakao.com/
- https://tmapapi.sktelecom.com/
- https://support.google.com/maps/answer/18539?hl=ko&co=GENIE.Platform%3DDesktop