

2020년도 한국산업기술대학교 컴퓨터공학부


# 종합설계 프로젝트 수행 보고서

Department of Computer Engineering, Korea Polytechnic Univ.

프로젝트명	모바일 앱을 이용한 버스 승하차 시스템 'BUS PICK'
팀번호	S2-1
문서제목	수행계획서( 0 ) 2차발표 중간보고서( 0 ) 3차발표 중간보고서( 0 ) 4차발표 중간보고서( 0 ) 최종결과보고서( 0 )

2020. 12. 01.

팀원 : 이재근 (팀장)  
나세영  
전유미  
최재원

지도교수 : 배유석 교수 

## 문서 수정 내역

작성일	대표작성자	버전(Revision)	수정내용	
2020.01.12	최재원	1.0	수행계획서	최초작성
2020.02.19	나세영	2.0	2차 발표자료	설계서추가
2020.04.24	전유미	3.0	3차 수행보고서	프로토타입 추가
2020.06.26	이재근	4.0	4차 수행보고서	Test 결과 추가
2020.11.19	최재원	5.0	최종 보고서	최초작성
2020.11.21	나세영	5.1	본론 내용 수정	추가작성
2020.11.25	전유미	5.2	본론 내용 수정	추가작성
2020.11.30	이재근	5.3	최종 보고서	추가작성

## 문서 구성

진행단계	프로젝트 계획서 발표	중간발표1 (2월)	중간발표2 (4월)	학기말발표 (6월)	최종발표 (10월)
기본양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식
포함되는 내용	I. 서론 (1~6)	I. 서론 (1~6)	I. 서론 (1~6)	I. 서론 (1~6)	I
	II. 본론 (1~3)	II. 본론 (1~4)	II. 본론 (1~5)	II. 본론 (1~7)	II
	참고자료	참고자료	참고자료	참고자료	III

이 문서는 한국산업기술대학교 컴퓨터공학부의 “종합설계”교과목에서  
프로젝트“모바일 앱을 이용한 버스 승하차 시스템 BUS PICK”을  
수행하는

(S2-1, 이재근,나세영,전유미,최재원)들이 작성한 것으로 사용하기 위해서는  
팀원들의 허락이 필요합니다.

# 종합설계기획 보고서 목차

## I. 서론

1. 작품선정 배경 및 필요성 .....	1
2. 기존 연구/기술동향 분석 .....	2
3. 개발 목표 .....	2
4. 팀 역할 분담 .....	3
5. 개발 일정 .....	3
6. 개발 환경 .....	4

## II. 본론

1. 개발 내용 .....	5
2. 문제 및 해결방안 .....	7
3. 시험 시나리오 .....	8
4. 상세 설계 .....	9
5. Prototype 구현 .....	18
6. 시험/테스트 결과 .....	29
7. Coding & DEMO .....	32

## III. 결론

1. 연구 결과 .....	50
2. 작품제작 소요재료 목록 .....	52

참고자료 .....	53
------------	----

# I. 서론

## 1. 작품선정 배경 및 필요성

필요성	내 용
수요배경	<ul style="list-style-type: none"><li>국가 통계포털 KOSIS의 국토교통부-「대중교통현황조사」(2018)에 따르면 현재 시내버스는 대중교통 이용객들의 57.3%가 이용할 정도로 많은 사람들이 이용하고 있음. (<a href="http://kosis.kr/statHtml/statHtml.do?orgId=116&amp;tblId=DT_MLTM_5719&amp;conn_path=I2">http://kosis.kr/statHtml/statHtml.do?orgId=116&amp;tblId=DT_MLTM_5719&amp;conn_path=I2</a>)</li></ul>
필요성 1	<ul style="list-style-type: none"><li>경기도 버스(GBUS)의 민원 상담 현황을 보면 많은 사람들이 버스 무정차로 인해 불편함을 겪고 있는 것을 확인할 수 있음. (<a href="http://www.gbuse.or.kr/2006/program/board/report_lst.php?gotopage=1&amp;sm=5_1&amp;inja=unkind">http://www.gbuse.or.kr/2006/program/board/report_lst.php?gotopage=1&amp;sm=5_1&amp;inja=unkind</a>)</li><li>많은 승객들이 불편함을 겪고 있는 버스 무정차 문제가 개선되지 않는 이유는 승객이 기사에게 명확한 승차 의사전달을 할 수 없기 때문. 현재 승객이 승차 의사를 표현할 수 있는 방법은 버스 기사를 향해 손을 흔드는 등의 원시적인 방법밖에 없으므로 이러한 문제를 승차 예약이라는 개념과 서비스를 도입하여 개선할 수 있음.</li></ul>
필요성 2	<ul style="list-style-type: none"><li>승객이 버스에서 하차하기 위해서는 본인이 내려야 하는 정류장을 확인하기 위해 창밖을 주시하거나 도착알람방송을 듣고 현재 위치를 가늠해야 한다는 불편함 존재</li><li>많은 탑승객들에 의해 하차벨을 누르지 못하는 경우, 버스 내 안내방송을 듣지 못하거나 잠이 들 경우 내려야 할 정류장을 지나치는 경우가 종종 발생</li><li>노선 안내 서비스와 하차 알람, 모바일 하차벨 서비스를 도입한다면 이러한 문제를 개선할 수 있음.</li></ul>
필요성 3	<ul style="list-style-type: none"><li>승객이 효율적으로 버스를 이용하기 위해 최단 경로를 탐색하려면 다른 지도 앱이나 웹에 검색해야 한다는 불편함 존재</li></ul>

## 2. 기존 연구/기술동향 분석

개발내용	내 용
국내 관련연구 소개, 장점, 단점	<p>■ 카카오버스</p> <ul style="list-style-type: none"> <li>• (소개) 전국 버스 정보를 확인 할 수 있는 모바일 어플리케이션</li> <li>• (장점) 버스 도착 정보 확인 가능, 승차알람과 하차알람으로 놓칠 걱정 없이 버스 이용 가능</li> <li>• (단점) 알람만 가능하기 때문에 하차벨은 개인이 따로 눌러야 함.</li> </ul> <p>■ 버스 도착 알림 어플리케이션</p> <ul style="list-style-type: none"> <li>• (소개) 버스가 정류장에 도착하기 전 알람이 오는 서비스 시스템을 소개함.</li> <li>• (장점) 자주 탑승하는 버스를 ‘즐거찾기’기능을 통해 이용할 수 있음. GPS 장치를 버스에 따로 부착하지 않고 기사 휴대폰의 위치기반 서비스를 이용함.</li> <li>• (단점) 도착 알림만 가능함. 기사 휴대폰의 위치기반 서비스를 이용하면 비용 절감, 확장성 등의 장점은 있지만, 기사가 휴대폰을 가져오지 않았거나 방전되었을 때 서비스를 이용하지 못할 수 있음.</li> </ul>

## 3. 개발 목표

### 최종목표

- 버스 이용 시 가장 불편함을 느끼는 버스 무정차, 하차 정류장 놓침 문제를 해소하기 위해서 현재 상용화 되어있지 않은 모바일 **승차벨/하차벨 기능**의 버스 모바일 어플리케이션을 제작해 불편함을 감소시키고, 분실물 게시판 및 버스 관리자 웹페이지(관리자 기능)를 활용해 이용자의 편의성을 증진시키고자 한다.

#### 4. 팀 역할 분담

		이재근	나세영	전유미	최재원
자료수집		서버 및 DB 구성요소 간 통신 기법	GBUS API 사용법 구성요소 간 통신 기법	구현 플랫폼	블루투스, 아두이노
설계		서버 및 DB 설계 구축	하차벨 모듈 설계 및 구현, 앱 UI 디자인	분실물 웹페이지, 운전자용 앱 설계	사용자용 앱 설계
구현	단계별 구현	서버, DB 설계 데이터 분석	앱 로고 제작, 하드웨어 설계	앱 분석 설계 및 구성요소 간 통신	앱 분석 설계 및 구성요소 간 통신
		관리자 웹페이지 구현, 서버, DB 연동 및 앱 API 파싱	사용자 앱 통신 및 하드웨어 구현	분실물 웹페이지, 운전자용 앱 개발, 서버, DB 연동 및 앱 API 파싱	사용자용 앱 개발, 서버, DB 연동 및 앱 API 파싱

#### 5. 개발 일정

##### ■ 활동 내용

항목		12월	1월	2월	3월	4월	5월	6월	7월	8월	9월	10월
시스템 설계	애플리케이션 설계	●	●	●								
	DB 설계	●	●	●								
	서버 설계	●	●	●								
애플리케이션 및 하드웨어 제작			●	●	●	●						
시연 및 데모	모듈간 연동			●	●	●	●	●				
	대외 실험(버스 탑승)				●	●	●	●	●			
	유지 보수						●	●	●	●	●	
문서화 및 보고서 작성							●	●	●	●	●	●
최종 검토 및 발표								●	●	●		

## 6. 개발 환경

### ■ SW

– Android Studio

개발사 : 구글

사용 개발 언어 : JAVA

라이선스 : Apache 2.0

사용 버전 : 3.5.3

– 아두이노 IDE

개발사 : 아두이노 소프트웨어

사용 개발 언어 : C

라이선스 : GPL

사용 버전 : 1.8.10

– MySQL

개발사 : MySQL AB

종류 : DBMS

라이선스 : GPL

사용 버전 : 8.0.18

– Apache Tomcat

개발사 : 아파치소프트웨어재단

라이선스 : 아파치라이선스 2.0

사용 버전 : 9.0.26

### ■ HW

– 아두이노 우노

마이크로 컨트롤러 : ATmega328

입력 전압(권장) : 7~12V

시스템 전압: 5V

디지털 입출력 핀 : 20개

아날로그 입력 핀 : 6개

플래시 메모리 : 32KB

SRAM : 2KB EEPROM : 1KB

무게 : 약25g

– HC-06

블루투스 무선 시리얼 통신

블루투스 V2.0 프로토콜 지원

범위 : ~10M

동작 전원 : 3.6 ~ 6V

사용 전력 : ~30mA

크기 : 36 x 15 mm

아두이노, 라즈베리 파이에서 사용 가능

– LED 램프

10파이(10mm)LED

동작전압 :2.2V

전류 : 20~30mA

– 피에조 부저

DM616 아두이노

능동 부저

동작 전압 : DC 3~24V

– 스위치 택트 버튼

원형 지름12mm

## II. 본론

### 1. 개발 내용

#### 개발 방법

##### 1. Application

###### ■ 사용자 APP

- 서버에서 받아온 버스정보로 버스/정류장 상세정보(노선, 방향, 시간, 위치 등) 검색 및 실시간 버스정보 확인
- 실시간 버스 정보와 자신의 위치 GPS를 기반으로 승차 벨을 요청해 운전자APP으로 카운트를 전달
- GPS와 Network 위치정보를 이용해 예약한 목적지 정류장 500m 이내 하차 알람 동작
- 블루투스를 이용하여 아두이노와 연동해 모바일 하차 벨 작동
- Google Map Api를 사용해 현재 위치 지도상에서 확인

###### ■ 운전자 APP

- DB와 연결된 서버를 통해 버스 번호 입력 후 운행버스 선택
- 서버로부터 받은 사용자용앱의 승차신호를 받아 정류장별 승객수 확인
- API와 GPS를 활용해 실시간 버스 운행 위치 확인
- Google Map Api와 지오코드, GPS를 활용해 운전자 현재 위치 지도상 표현

##### 2. Hardware

- 아두이노 우노에 블루투스 모듈, LED램프, 피에조 부저, 스위치 택트 버튼 연결
- 전체 동작의 이해도를 높이고 스위치 버튼을 문이 열리며 누를 수 있는 버스 모형 제작

##### 3. Server 및 DB

- Apache Tomcat을 이용한 서버 구축
- 경기버스 API를 활용해 공공데이터(실시간 버스위치 정보, 노선정보)파싱 및 DB 연동
- MySQL을 이용해 시흥/안산 버스 노선정보를 저장할 DB 구축
- 서버와 사용자/운전자용 어플리케이션 연동

##### 4. Web

###### ■ 관리자 웹 페이지

- MySql과 Apache tomcat을 이용해 노선-정류장 데이터 관리자 기능 구현(수정 및 갱신 기능)
- 노선-정류장 지도맵 상에 표현
- 사용자용&운전자용 앱 이용률에 대한 통계자료 확인

###### ■ 분실물 웹 게시판

- MySql과 Apache tomcat을 이용해 서버와 DB간 통신으로 로그인, 회원가입, 분실물 목록 확인, 분실물 습득시 웹게시판 글 게시 기능을 구현

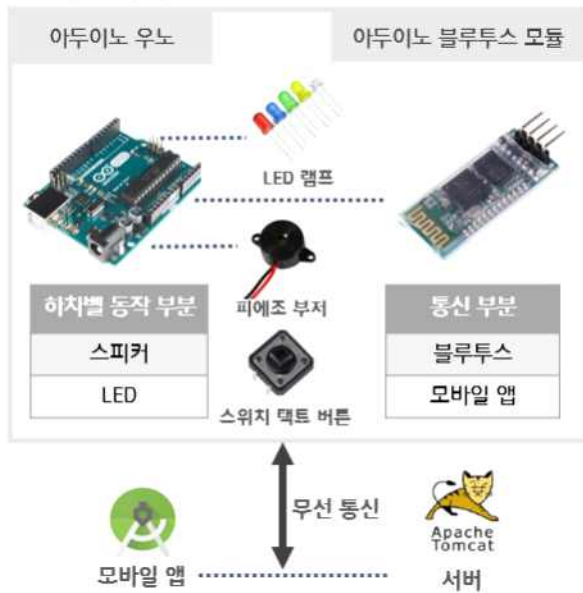
##### 5. GitHub

- <https://github.com/seyoung622/bus>

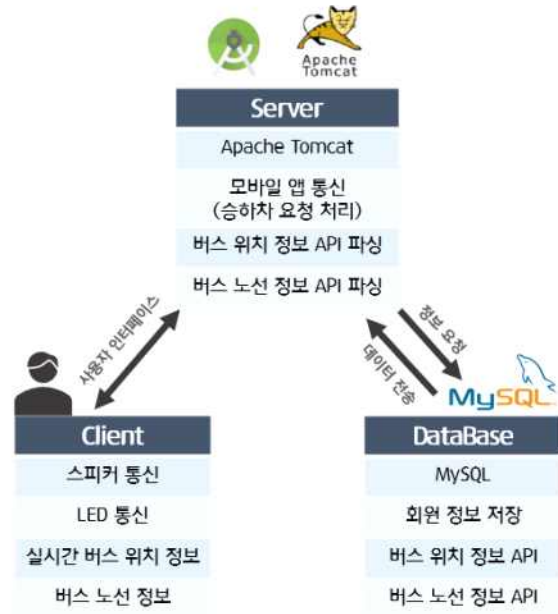


## 시스템 구성도

### ■ 하드웨어



### ■ 소프트웨어



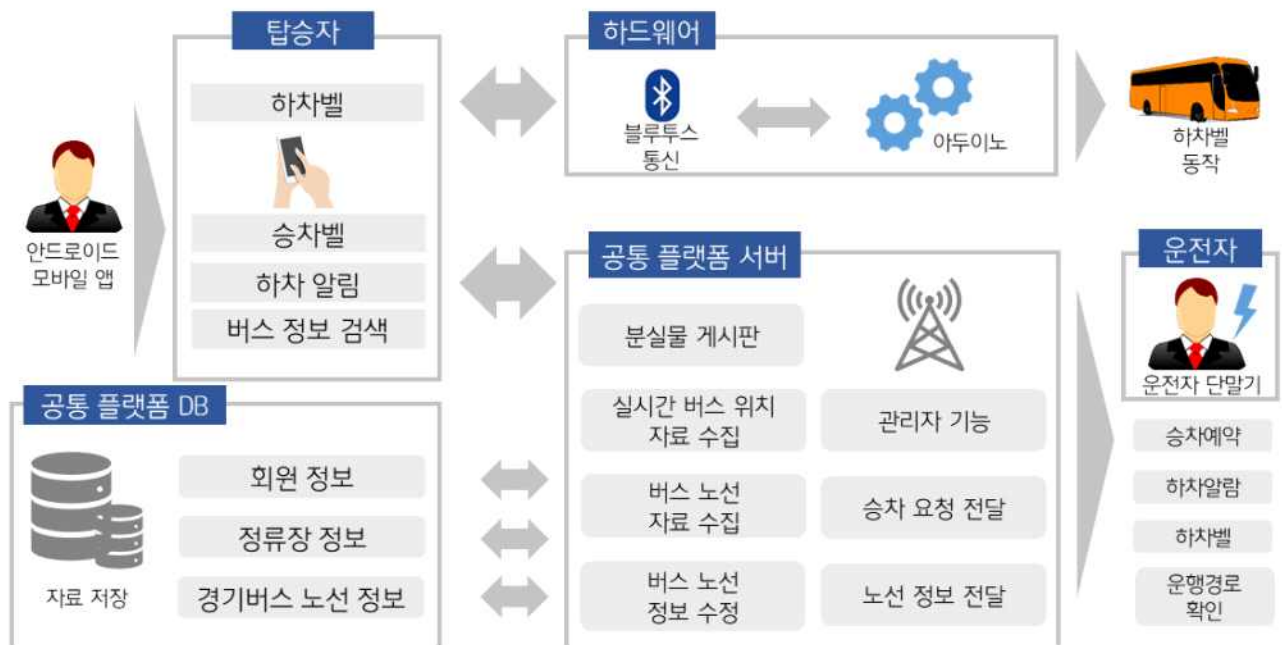
## 2. 문제 및 해결방안

문제점	
1	■ 기존 불변의 데이터가 아닌 실시간 데이터 통한 환경 구축
2	■ 승차 요청한 정류장의 승객 수 표시
3	■ 목적지까지 거리에 따른 하차 알람 동작
4	■ 모바일 앱을 통해 하차벨 동작
5	■ 데이터 관리에 대한 기능 부재
해 결 방 안	
1	■ 환경에 맞는 API 활용 - 실시간 버스위치 정보, 노선정보 API 파싱 및 DB연동
2	■ 계정 통해 승차 요청 시 그 수를 카운트 하여 표기
3	■ 승객 GPS 위치와 실시간 버스 위치를 이용해 거리 계산
4	■ 하드웨어(아두이노, 블루투스 모듈, 부저, 센서) 활용해 하차벨 기능 수행
5	■ 정류장 및 노선 데이터 관리를 위한 관리자페이지 구현

### 3. 시험 시나리오

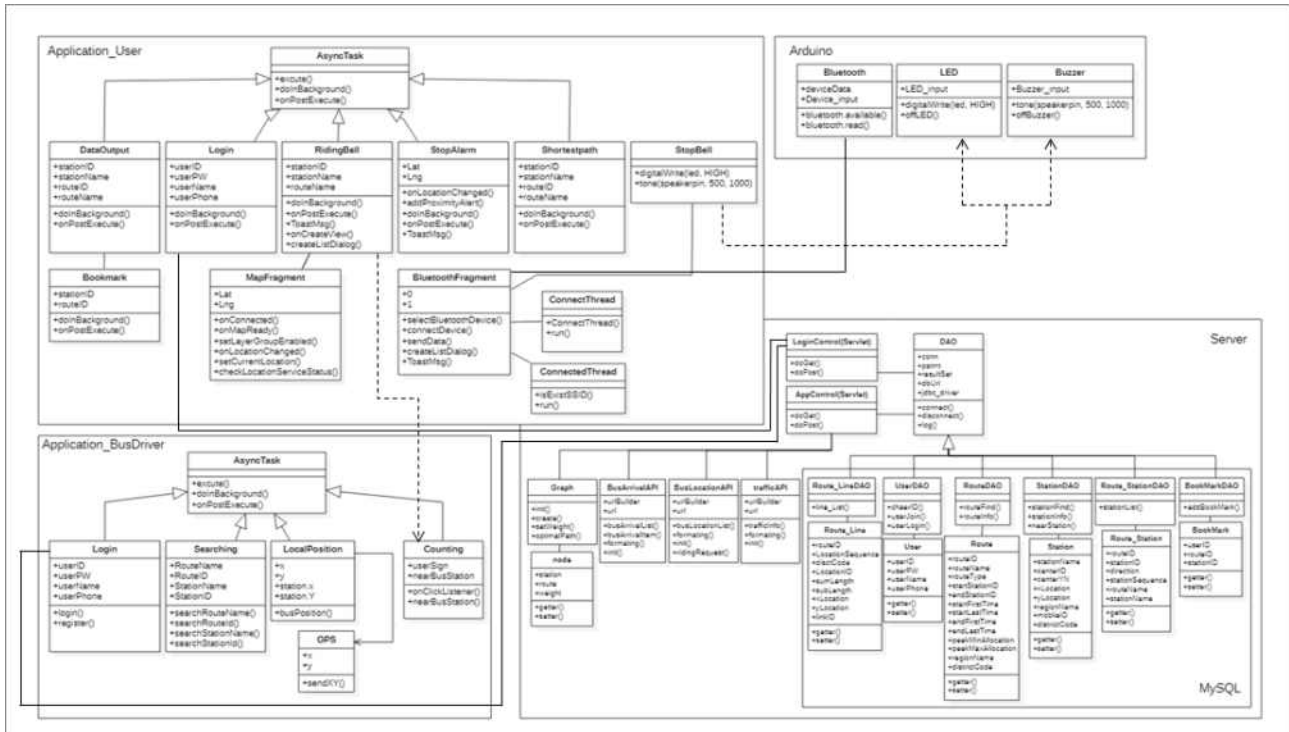
시나리오명	내 용
시나리오 1	<ul style="list-style-type: none"> <li>• (로그인/회원가입)</li> </ul> 운전자 및 사용자 계정으로 로그인, 회원가입
시나리오 2	<ul style="list-style-type: none"> <li>• (승차 예약)</li> </ul> 탑승객이 정류장에서 근처 버스 검색을 통해 모바일 승차 알림을 운전자유 앱에 전송 시 운전자 확인 후 해당 정류장 정차
시나리오 3	<ul style="list-style-type: none"> <li>• (하차 알림)</li> </ul> 하차 정류장 선택 시 사용자 스마트폰 GPS와 버스 노선 정보를 통해 목적지에 가까워지면 알람 수신
시나리오 4	<ul style="list-style-type: none"> <li>• (하차벨)</li> </ul> 램프와 스피커 부착된 아두이노와 블루투스 연결하여 하차 요청을 알리기 위해 모바일 앱을 통해 하차벨 눌러 하차
시나리오 5	<ul style="list-style-type: none"> <li>• (노선검색)</li> </ul> 버스 정보 입력 시 해당 버스의 노선 출력되어 확인 가능
시나리오 6	<ul style="list-style-type: none"> <li>• (분실물 게시)</li> </ul> 버스 이용 시 분실한 물품 찾기 및 습득한 분실물 정보 게시 가능
시나리오 7	<ul style="list-style-type: none"> <li>• (관리자 기능)</li> </ul> 버스 노선 및 정류장 데이터 관리에 대한 기능 수행

시나리오 (그림)

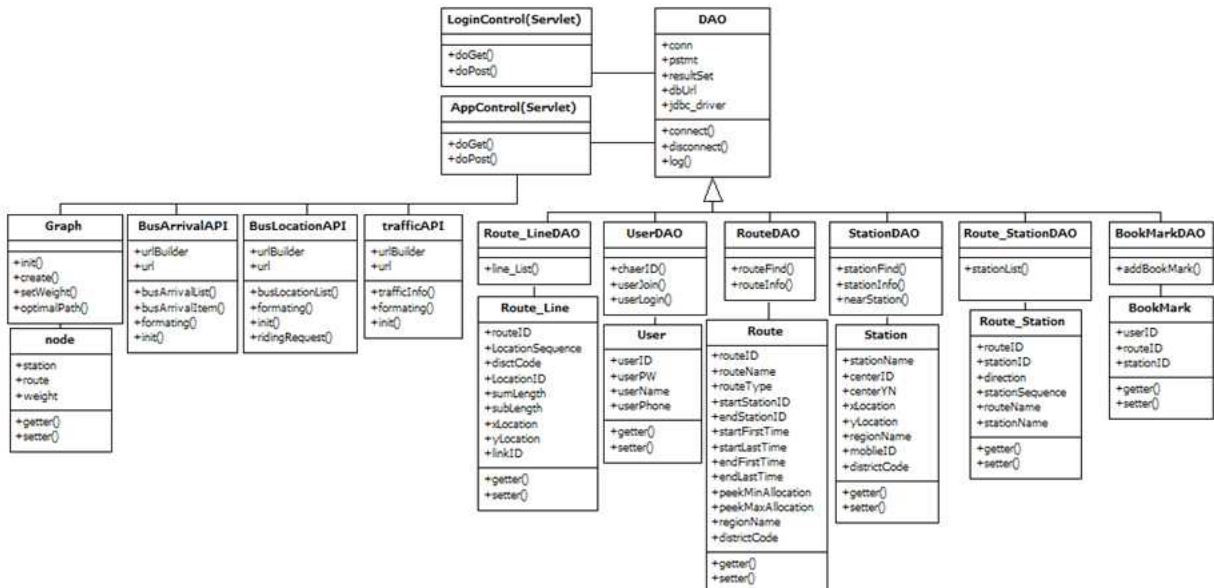


## 4. 상세설계

### 전체 Class Diagram



### Server Class Diagram



## ■ Server 대표 세부기능

### <userJoin>

- ◆ 기능
  - 사용자 app에서 넘어온 정보를 DB에 추가
- ◆ 다루는 값
  - userID, userPW, username, userPhone
  - 반환 : 0 (성공), 1 (실패)
- ◆ 로깅에 기록할 정보
  - 요청시간, ID, 성공실패 여부, 실패 시 실패 사유
- ◆ 반영하는 시점
  - 상시

### <userLogin>

- ◆ 기능
  - 사용자의 로그인 요청 처리, ID와 비밀번호가 DB와 일치하는지 확인 후 결과 전달
- ◆ 다루는 값
  - 입력 : userID, userPW
  - 반환 : 0 (성공), 1 (ID없음), 2 (PW 불일치)
- ◆ 로깅에 기록할 정보
  - 요청시간, ID, 성공실패 여부, 실패 시 실패 사유
- ◆ 반영하는 시점
  - 상시

### <routeFind>

- ◆ 기능
  - 사용자가 검색창에 입력한 버스 노선 이름을 포함하는 버스 노선 정보 전달
  - 노선 이름만으로는 정보를 알 수 없기 때문에 정보를 추가 제공, 원하는 노선을 선택
- ◆ 다루는 값
  - 입력 : 노선 이름
  - 반환 : 노선 이름, 노선 지역, 노선 아이디, 노선 타입(성공), 1 (실패)
- ◆ 로깅에 기록할 정보
  - 요청시간, ID, 성공실패 여부, 실패 시 실패 사유
- ◆ 반영하는 시점
  - 상시

#### <routeInfo>

##### ◆ 기능

– 사용자가 요청한 버스 노선 정보 전달, 버스 경유 정류장, 기점, 종점, 배차간격, 첫 차, 막차 정보 전달

##### ◆ 다루는 값

- 입력 : 노선 이름, 노선 아이디
- 반환 : 노선 정보, 노선-정류장 정보, 실시간 버스 위치 (성공), 1 (실패)

##### ◆ 로깅에 기록할 정보

- 요청시간, ID, 성공실패 여부, 실패 시 실패 사유

##### ◆ 반영하는 시점

- 상시

#### <ridingRequest>

##### ◆ 기능

- 사용자의 승차요청을 해당 버스에 전달

##### ◆ 다루는 값

- 입력 : routeID, stationID, palteNO
- 반환 : 0 (성공), 1 (실패)

##### ◆ 로깅에 기록할 정보

- 요청시간, ID, 버스노선, 정류장, 성공 실패 여부 , 실패 시 실패 사유

##### ◆ 반영하는 시점

- 상시

#### <stationFind>

##### ◆ 기능

- 사용자가 검색창에 입력한 정류장 이름을 포함하는 정류장 정보 전달
- 정류장 이름만으로는 정보를 알 수 없기 때문에 정보를 추가 제공, 원하는 정류장을 선택

##### ◆ 다루는 값

- 입력 : 정류장 이름
- 반환 : 정류장 이름, 정류장 번호(성공), 1 (실패)

##### ◆ 로깅에 기록할 정보

- 요청시간, ID, 성공실패 여부, 실패 시 실패 사유

##### ◆ 반영하는 시점

- 상시

#### <stationInfo>

- ◆ 기능
  - 사용자가 요청한 버스 정류장 정보 전달
  - 버스 정류장 모바일 번호, 버스정류장 이름, 방향, 지도내 위치, 버스 도착정보 전달
- ◆ 다루는 값
  - stationID
  - 반환 : 노선 정보, 노선-정류장 정보, 정류장 도착정보 (성공), 1 (실패)
- ◆ 로깅에 기록할 정보
  - 요청시간, ID, 성공실패 여부, 실패 시 실패 사유
- ◆ 반영하는 시점
  - 상시

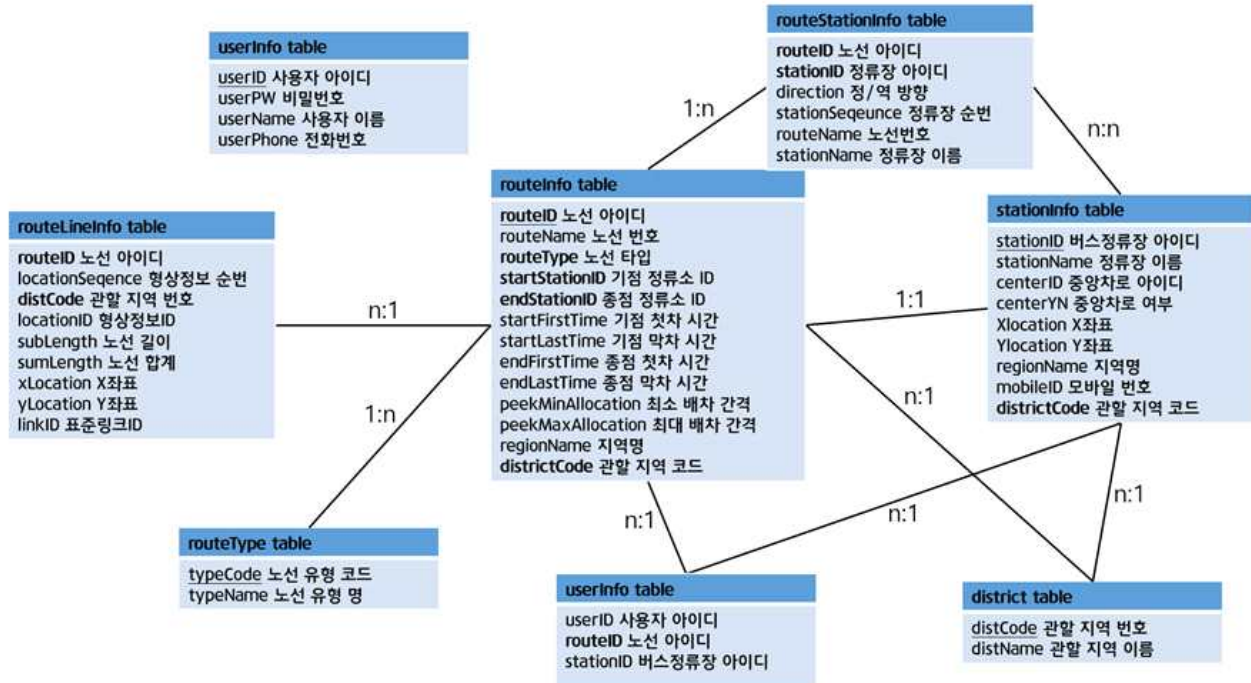
#### <nearStation>

- ◆ 기능
  - 사용자 주변 버스 정류장 정보 반환
  - 버스 정류장 모바일 번호, 버스정류장 이름, 방향, 지도내 위치, 버스 도착정보 전달
- ◆ 다루는 값
  - station 객체, 사용자 좌표
  - 반환 : 노선 정보, 노선-정류장 정보, 정류장 도착정보 (성공), 1 (실패)
- ◆ 로깅에 기록할 정보
  - 요청시간, ID, 성공실패 여부, 실패 시 실패 사유
- ◆ 반영하는 시점
  - 상시

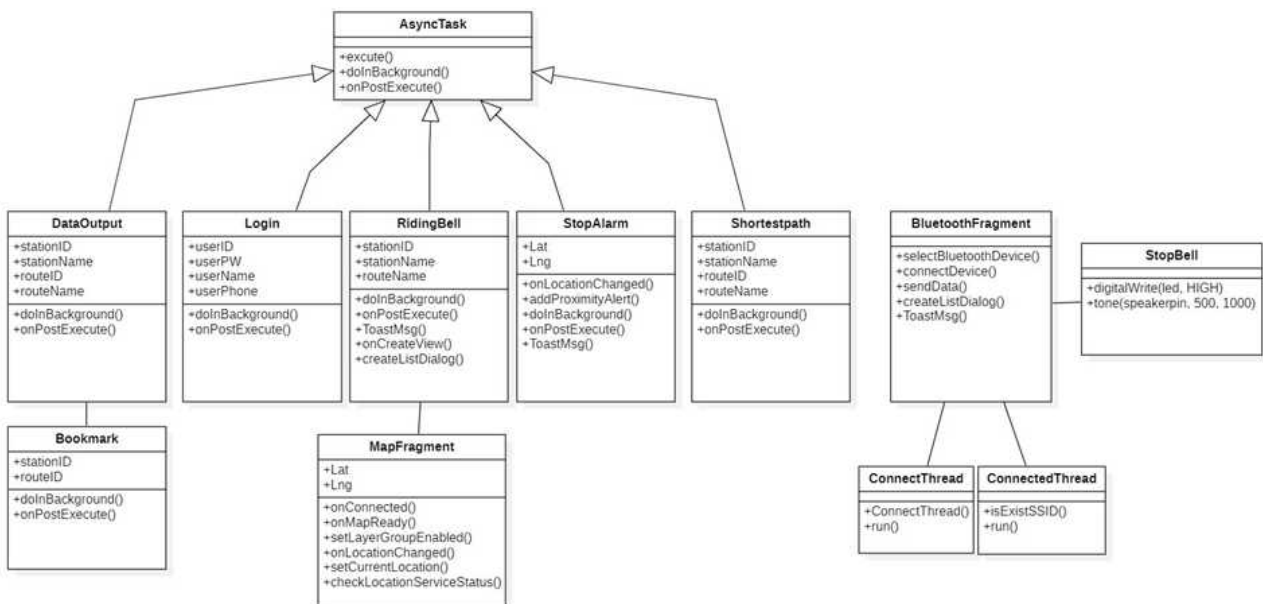
#### <optimalPath>

- ◆ 기능
  - 출발 정류소와 목적지 정류소를 이용해 경로 탐색
  - 실시간 교통정보를 수집해 간선의 가중치를 설정
  - 경유 정류장, 노선정보 전달
- ◆ 다루는 값
  - 입력 : 출발 정류장, 목적지 정류장
  - 반환 : 경유 정류장, 버스노선 (성공), 1 (실패)
- ◆ 로깅에 기록할 정보
  - 요청 시간, ID, 성공, 실패 여부, 실패 시 사유
- ◆ 반영하는 시점
  - 상시

## ■ DB 관계



## ■ 사용자 APP Class Diagram





## ■ 사용자 APP 대표 세부기능

### <DataOutput>

#### ◆ 기능

- 서버를 통해 DB에 저장된 정류장 정보, 버스 노선 정보를 읽고 출력해 줌.
- 버스번호 또는 정류장명을 입력하면 해당 정보를 검색 함.
- 자주 검색하는 정류장 또는 버스를 즐겨찾기에 추가

#### ◆ 다루는 값

- 전송 : 찾을 버스 노선 번호, 정류장 이름
- 반환 : stationID : 정류장ID, stationName: 정류장이름, routeID : 버스노선ID routeName : 버스번호
- 즐겨찾기 기능 (stationID, routeID)

#### ◆ 로깅에 기록할 정보

- 검색 기록

#### ◆ 필요한 정보

- DB의 경기버스 API 정류장 정보, 버스 노선 정보 데이터

### <Login>

#### ◆ 기능

- 로그인 및 회원가입에 필요한 정보 관리

#### ◆ 다루는 값

- userID, userPW, userName, userPhone
- id 없을 시 : 회원가입 / password 불일치 : 재로그인 / id+password 일치 : 로그인 성공

#### ◆ DB 반영 시점

- 새로운 id, password로 회원가입 시
- 로그인 시도 시

### <RidingBell>

#### ◆ 기능

- 탑승객의 GPS정보를 받아와 지도상에 가까운 정류장 위치를 보여줌
- 정류장을 선택하면 탑승 가능한 버스 번호 LIST를 보여줌
- 선택한 정류장, 버스 번호 정보를 확인 한 뒤 서버와 HTTP통신을 통해 운전자 어플로 탑승 요청을 보냄

#### ◆ 다루는 값

- MapFragment 사용 (onMapReady() , onLocationChanged() , setCurrentLocation() 등)
- Lat : 탑승객의 위도, Lng : 탑승객의 경도
- stationID : 정류장ID, stationName: 정류장이름, routeName : 버스번호
- ToastMsg(): 선택사항 확인 창, onCreateView(): 프래그먼트 호출
- createListDialog(): 버스 리스트 생성

#### ◆ 필요한 정보

- Android GPS 을 통해 사용자의 위치 파악
- 카카오맵API 를 통해 지도상의 위치 표현
- 경기버스 API의 정류장 데이터

### <Bluetooth>

#### ◆ 기능

- 사용자의 하차벨 요청 유무를 전송하기 위해 사전에 장치를 검색하고 연결함

#### ◆ 다루는 값

- selectBluetoothDevice() : 장치 선택
- connectDevice() : 장치와 연결
- sendData() : 값 전송
- createListDialog(): 리스트 다이얼로그 생성
- ToastMsg(): 연결 확인 창
- 연결을 위한 ConnectThread, ConnectedThread 사용

#### ◆ 반영하는 시점

- 사용자가 APP에서 하차벨 버튼을 눌렀을 때 블루투스로 값 전송

### <StopAlarm>

#### ◆ 기능

- 사용자가 설정 해 놓은 하차할 정류장이 가까워지면 알람 전송

#### ◆ 다루는 값

- onLocationChanged () : 최신 위치 갱신
- addProximityAlert() : 지정한 위도, 경도에 가까워지면 근접알림 발생
- ToastMsg(): 알람 메시지 창

#### ◆ 반영하는 시점

- 하차할 정류장의 전 정류장을 출발하고 10초 뒤 알람

### <StopBell>

#### ◆ 기능

- 사용자가 하차벨 버튼을 누르면 버스에 있는 기존 하차벨 LED가 켜지고 부저에서 소리가 남

#### ◆ 다루는 값

- digitalWrite(led, HIGH) : LED가 켜짐
- tone(speakerpin, 500, 1000) : 부저에서 BEEP 음이 1초동안 지속됨

#### ◆ 반영하는 시점

- 사용자가 하차벨 버튼을 눌렀을 때 LED ON/SOUND ON

### <Shortestpath>

#### ◆ 기능

- 출발지/목적지 버스정류장 선택
- 사용자가 선택한 버스 정류장 간의 최단경로를 찾은 결과를 화면에 보여 줌

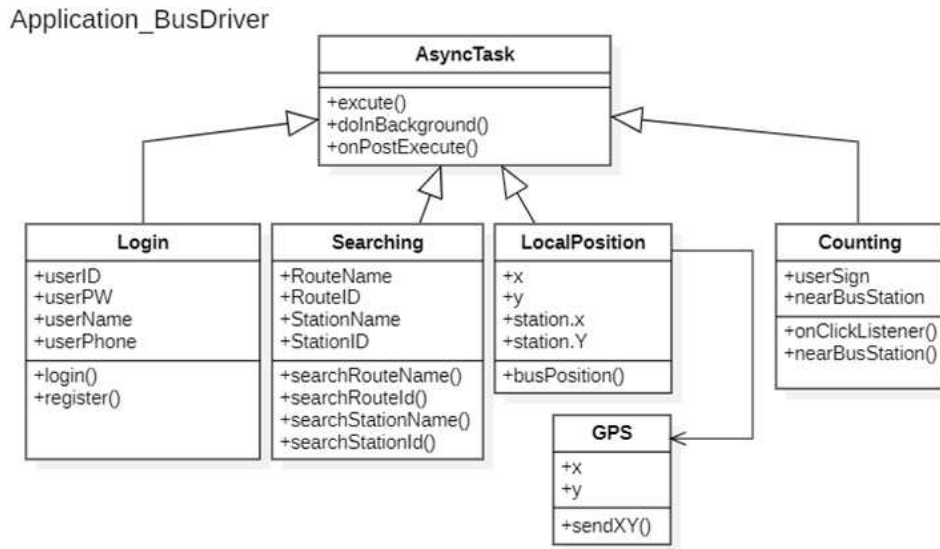
#### ◆ 다루는 값

- stationID, stationName (출발 정류장, 도착 정류장)
- mapType 의 setLayerGroupEnabled()메서드의 LAYER\_GROUP\_TRAFFIC 그룹 사용

#### ◆ 로깅에 기록할 정보

- 이전 검색 이력

## ■ 운전자 APP Class Diagram



## ■ 운전자 APP 대표 세부기능

### <Login>

#### ◆ 기능

- 로그인 및 회원가입에 필요한 정보 관리

#### ◆ 다루는 값

- Id, Password, UserName, PhoneNum
- id 없을 시 : 회원가입 / password 불일치 : 재로그인 / id+password 일치 : 로그인 성공

#### ◆ DB 반영 시점

- 새로운 id, password로 회원가입 시
- 로그인 시도 시

### <BusData>

#### ◆ 기능

- 버스 번호, 버스 노선, 버스 정류장 등 정보 관리

#### ◆ 다루는 값

- RouteName, RouteID, StationName, StationID 정보

#### ◆ DB 반영 시점

- (앱 구현 전 버스 데이터 DB설계 완료)
- 버스 번호 검색 시
- 버스 정류장 검색 시
- 정류장 별 승차 신호 확인 시

### <LocalPosition>

#### ◆ 기능

- GPS 통해 현재 버스 위치 실시간 관리

#### ◆ 다루는 값

- station.x : GPS상의 x 좌표 station.y : GPS 상의 y좌표

- ◆ DB 반영 시점
  - 버스 노선 및 정류장 좌표와 현재 위치 좌표 비교 시
  - 인접 정류장의 승차 예약 신호 확인 시

### <Counting>

- ◆ 기능
  - RidingBell Module로부터 받은 정류장 승차 신호 관리
- ◆ 다루는 값
  - Sign : 승차 예약 신호
  - (사용자 앱)승차 버튼 OnClick 동작 시 +1
- ◆ DB 반영 시점
  - 정류장 별 승차 신호 카운팅 시
  - 해당 버스 정류장 지나가면 0으로 초기화

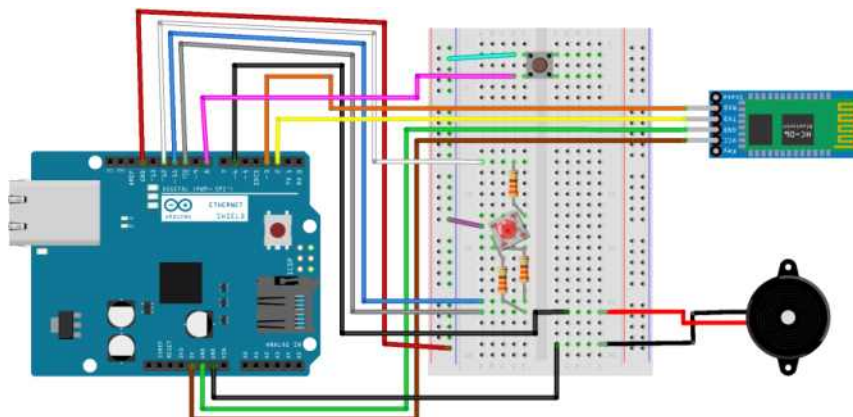
## ■ 하차벨 설계

Modules

Bluetooth	LED	Buzzer
+deviceData +Device_input	+LED_input	+Buzzer_input
+connectDevice() +selectBluetoothDevice() +sendData()	+onLED() +offLED()	+onBuzzer() +offBuzzer()

- 블루투스 모듈 - 어플리케이션과 블루투스로 통신
- LED 모듈 - 하차벨 동작 시 LED ON
- 피에조 부저 모듈 - 하차벨 동작 시 SOUND ON
- 스위치 택트 버튼 모듈 - 하차벨 작동 후 문이 열릴 시 LED OFF


## ■ 하차벨 아두이노 회로도





## 5. Prototype 구현

### ■ 사용자 앱


#### 1. Intro

화면	설명
	<ol style="list-style-type: none"> <li>1. Application 이름 표시</li> <li>2. 서비스를 나타내는 intro 로고 표시</li> <li>3. 2초 후 자동으로 메인화면으로 이동</li> <li>4. SplashActivity를 생성하여 실행 후 메인 액티비티로 넘어가도록 설정함.</li> </ol>

#### 2. 메인화면

화면	화면 상세보기	설명
		<ol style="list-style-type: none"> <li>1. 메인화면에 구글 지도로 사용자가 지도를 확인할 수 있게 함.</li> <li>2. 오른쪽 상단 BUSPICK 로고를 클릭하면 메인화면으로 이동함.</li> <li>3. 오른쪽 상단 오버플로우 메뉴 (⋮) 클릭 시 즐겨찾기, 검색, 승차벨, 하차알람, 하차벨, 최단경로가 포함된 메뉴를 열 수 있음.</li> </ol>


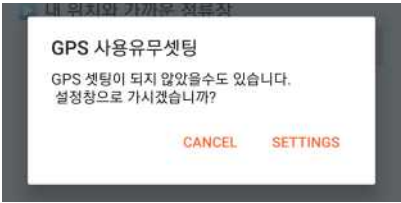


### 3. 메뉴 구성

화면	화면 상세보기	설명
	<div style="border: 1px solid black; padding: 10px; margin: 10px;"> <p><b>검색</b></p> <p><b>승차벨</b></p> <p><b>하차알람</b></p> <p><b>하차벨</b></p> <p><b>최단경로</b></p> <p><b>분실물 게시판</b></p> </div>	<ol style="list-style-type: none"> <li>1. 검색 : 클릭 &gt; 검색 화면 이동</li> <li>2. 승차벨 : 클릭 &gt; 승차벨 화면 이동</li> <li>3. 하차알람 : 클릭 &gt; 하차알람 화면 이동</li> <li>4. 하차벨 : 클릭 &gt; 하차벨 화면 이동</li> <li>5. 최단경로 : 클릭 &gt; 최단경로 화면 이동</li> <li>6. 분실물 게시판 : 클릭 &gt; 분실물 게시판 화면 이동</li> </ol>

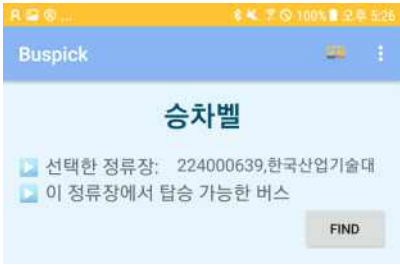
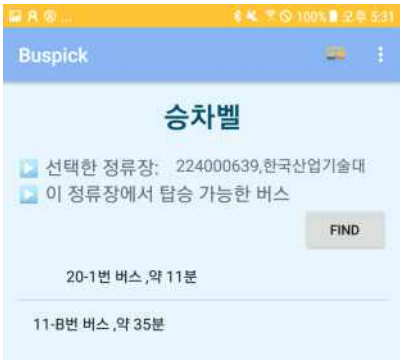
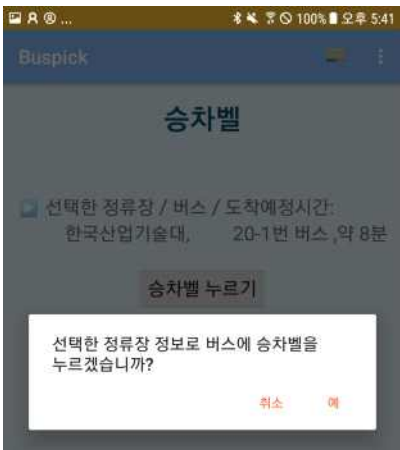
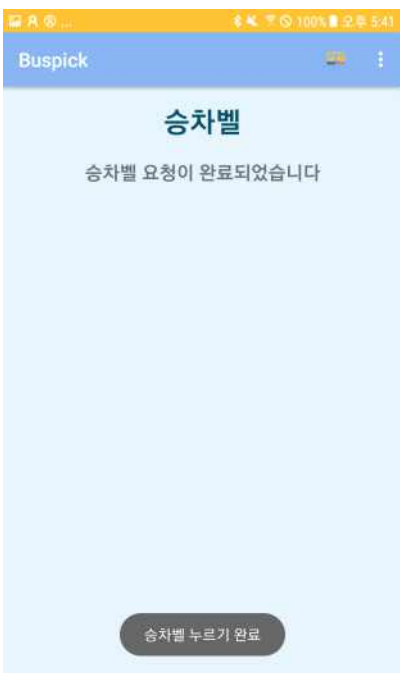
#### 4. 버스/정류장 검색

화면	화면 상세보기	설명
	 	<p>1. 버스 [검색]          버스 번호 입력 &gt; 클릭 &gt; 버스 정보가 서버를 통해 DB에 저장되어 있는 routeID, routeName 값을 받아와 원하는 버스 정보를 출력함.</p> <p>2. 정류장 [검색]          정류장 입력 &gt; 클릭 &gt; 정류장 정보가 서버를 통해 DB에 저장되어 있는 stationID, stationName 값을 받아와 원하는 정류장 정보를 출력함.</p>

## 5. 승차벨

화면	화면 상세보기	설명
	<div>  </div> <div>  </div> <div>  </div>	<p>1. [CHECK] 클릭 &gt; GPS 사용유무 다이얼로그의 SETTINGS 클릭 &gt; GPS 사용 권한 설정 &gt; 내 위치의 위도, 경도 정보를 출력하고 Toast 메시지를 띄웁니다.</p> <p>2. [FIND] 클릭 &gt; 위도, 경도 정보를 서버에 전송하여 내 위치에서 300m 내 있는 가까운 정류장을 리스트로 출력함.</p> <p>3. 내 위치와 가까운 정류장 리스트 중 원하는 정류장 클릭 &gt; 다음 화면으로 이동</p>




	   	<p>4. intent를 이용해 이전 액티비티에서 선택한 stationID와 stationName을 가져옴.</p> <p>5. [FIND] 클릭 &gt; 해당 정류장에서 탑승 가능한 버스 리스트 출력</p> <p>6. [승차벨 누르기] 클릭 &gt; 정보를 확인하는 다이얼로그 메시지를 출력함. 예 &gt; 서버를 통해 운전자 앱으로 승차벨 정보 전달.</p> <p>7. “승차벨 누르기 완료” Toast 메시지와 완료창 출력.</p>
--	--	---

## 6. 하차벨



화면	화면 상세보기	설명
		<ol style="list-style-type: none"> <li>1. 액티비티 접속 시 블루투스 허용여부와 페어링 되어있는 블루투스 디바이스 목록이 출력되고 원하는 하차벨 모듈과 페어링함.</li> <li>2. [하차] 클릭 &gt; LED 꺼짐 &gt; 1초간 부저 울림</li> <li>3. 하차할 때 열리는 버스 문이 문 끝에 있던 스위치 버튼을 누르면 LED 꺼짐.</li> </ol>
<div style="display: flex; justify-content: space-around;">   </div>		

## ■ 운전자 앱

### 1. INTRO 시작화면

화면	설명
	<ol style="list-style-type: none"> <li>1. 운전자 Application INTRO Logo - 4초</li> <li>2. 4초 후 메인화면[버스 번호로 검색하기] 전환</li> <li>3. Handler와 Intent통해 첫 인트로 화면 구성</li> </ol>


### 2. 메인화면

화면	화면 상세보기	설명
		<ol style="list-style-type: none"> <li>1. Main [버스 번호로 검색하기] : 클릭 &gt; 검색 화면으로 이동</li> <li>2. [지도 보기] : 클릭 &gt; 검색 화면 이동</li> <li>3. [노선 검색] : 클릭 &gt; 버스 검색 화면 이동</li> <li>4. [분실물 게시판] : 클릭 &gt; 분실물 게시판 이동</li> <li>5. [오른쪽 상단 BUSPICK 로고] : 메인화면으로 이동</li> </ol>

### 3. 검색 및 승차 신호 확인




화면	화면 상세보기	설명
		<p>1. [버스 검색] 운전할 버스의 routeID 입력 후 검색 클릭</p> <p>-&gt; 서버를 통해 전달된 차량번호(plateNo)와 정류장순서(stationSeq) 출력</p> <p>-&gt; 운전자의 차량 번호와 일치하는 버스 선택</p> <p>-&gt; 다음 Activity로 이동</p>
		<p>1. [정류장별 승차요청 받기] 앞 액티비티에서 운전자의 차량 번호와 일치하는 버스 클릭 후 전환되는 화면. Station별 승객의 승차요청을 서버로부터 수신.</p> <p>2. stationName에 해당되는 카운팅 숫자를 화면에 출력.</p> <p>3. [실시간 운행 위치] 실시간 위치 API를 받아 현재 이동하고 있는 위치 표시. (SequenceNo 활용)</p> <p>4. CountDownTimer() 메소드를 활용해 30초마다 요청 및 ViewHolder갱신</p>



#### 4. 지도

화면	설명
	<p>▶ Google Map API와 Device GPS 권한 활용해 운전자의 현재 위치 확인 가능</p> <ol style="list-style-type: none"> <li>1. Map API 활용 위해 Manifest 파일 내 해시값 및 권한 설정 (API KEY, INTERNET, ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION)</li> <li>2. 위도, 경도 좌표값을 받기 위해 Device Permission 허용</li> <li>3. Marker 이용해 위치를 정확히 표시</li> <li>4. 우측 상단 아이콘 클릭해 현재 위치 찾아 중앙 정렬</li> <li>5. 예외처리 및 동작 오류 알림 <ul style="list-style-type: none"> <li>- 지오코더 서비스 사용불가</li> <li>- 잘못된 GPS좌표</li> <li>- 주소 미발견</li> <li>- 위치 권한과 GPS 활성화 여부 확인 등</li> </ul> </li> </ol>


#### ■ 앱 공통 기능

##### 1. 분실물 게시판

화면	화면 상세보기	설명
	 	<ol style="list-style-type: none"> <li>1. [앱 내 연결] WebView를 이용해 분실물 게시판 웹페이지 연결</li> <li>2. [로그인] MySQL DB user_table 연동하여 등록된 회원만 이용. (userid, passwd)</li> <li>3. [회원가입] MySQL DB user_table 통해 회원가입 후 게시판 이용. (userid, passwd, username, tel, email, sex)</li> </ol>

화면	화면 상세보기	설명
		<p>4. [분실물 목록] MySQL DB Product_table 연동하여 올라온 글 활용해 찾고자 하는 물건이 등록돼있는지 확인 가능</p> <p>5. [분실물 게시물 올리기] MySQL DB Product_table 연동하여 습득한 게시물 글 작성 (pno, userid, pname, category, busnum, image, info)</p>

## 2. 관리자 페이지 (웹)

화면	설명
	<p>1. [메인페이지] Tomcat, JSP를 이용해 관리자 페이지 구현</p> <ul style="list-style-type: none"> <li>- 정류장 추가 정류장 정보 추가 페이지 이동</li> <li>- 전체 버스 정류장 데이터 보기 정류장 데이터 보기 페이지 이동</li> <li>- 전체 버스 노선 데이터 보기 버스 노선 보기 페이지 이동</li> </ul>

화면

Update Bus

정류장 아이디

공백없이 입력하세요

정류장 이름

공백없이 입력하세요

차로 아이디

공백없이 입력하세요

중앙차로 여부

공백없이 입력하세요

정류장 경도

공백없이 입력하세요

정류장 위도

공백없이 입력하세요

정류소 관리 지역

공백없이 입력하세요

모바일 코드

공백없이 입력하세요

지역 코드

공백없이 입력하세요

보내기

다시 작성

메인 페이지 이동

정류장 아이디

검색

정류장 아이디	정류장 이름	차로 아이디	중앙차로 여부	정류장 경도	정류장 위도	정류소 관리 지역	모바일 코드	지역 코드
1	1	1	1	1	1	1	1	1
115000067	발효문화교	10000000	N	126.8059667	37.5645833	서울	16164	1
115000083	김포공항역내선(3번출)	10000000	N	126.8010833	37.56505	서울	16392	1
115000098	한국공화문사	10000000	N	126.7987667	37.5625333	서울	16411	1
115000068	대한항공빌딩	10000000	N	126.81485	37.5507667	서울	16995	1
115000071	김포새마을아시아나점남교	10000000	N	126.8080833	37.5557	서울	16993	1
115000072	김포공항화물터미널	10000000	N	126.8114	37.5534	서울	16992	1
115000073	대한항공문화센터	10000000	N	126.81315	37.5463167	서울	16988	1
115000074	대한항공화물터미널	10000000	N	126.81335	37.55155	서울	16987	1
115000075	대한항공문화센터	10000000	N	126.8123333	37.5458667	서울	19992	1
115000080	아시아나점남교	10000000	N	126.8075833	37.5563167	서울	16983	1
115000081	김포공항역내선(양방향)	10000000	N	126.8032833	37.5595	서울	16605	1
115000086	조선총독부단지	10000000	N	126.7974667	37.53815	서울	16726	1
115000087	조선총독부단지	10000000	N	126.79785	37.5381833	서울	16725	1

전체 버스 정류장 데이터

메인 페이지 이동

정류장 아이디

검색

정류장 아이디	정류장 이름	차로 아이디	중앙차로 여부	정류장 경도	정류장 위도	정류소 관리 지역	모바일 코드	지역 코드
1	1	1	1	1	1	1	1	1
115000067	발효문화교	10000000	N	126.8059667	37.5645833	서울	16164	1
115000083	김포공항역내선(3번출)	10000000	N	126.8010833	37.56505	서울	16392	1
115000098	한국공화문사	10000000	N	126.7987667	37.5625333	서울	16411	1
115000068	대한항공빌딩	10000000	N	126.81485	37.5507667	서울	16995	1
115000071	김포새마을아시아나점남교	10000000	N	126.8080833	37.5557	서울	16993	1
115000072	김포공항화물터미널	10000000	N	126.8114	37.5534	서울	16992	1
115000073	대한항공문화센터	10000000	N	126.81315	37.5463167	서울	16988	1
115000074	대한항공화물터미널	10000000	N	126.81335	37.55155	서울	16987	1
115000075	대한항공문화센터	10000000	N	126.8123333	37.5458667	서울	19992	1
115000080	아시아나점남교	10000000	N	126.8075833	37.5563167	서울	16983	1
115000081	김포공항역내선(양방향)	10000000	N	126.8032833	37.5595	서울	16605	1
115000086	조선총독부단지	10000000	N	126.7974667	37.53815	서울	16726	1
115000087	조선총독부단지	10000000	N	126.79785	37.5381833	서울	16725	1

전체 버스 노선 데이터

메인 페이지 이동

노선 아이디

검색

노선 아이디	노선 번호	노선 유형	기점 정류장 아이디	종점 정류장 아이디	기점 주차 출발 시간	기점 주차 출발 시간	종점 주차 출발 시간	종점 주차 출발 시간	배차 간격(분)	지역 코드
210000002	12	13	234000945	115000601	04:30	22:30	05:50	23:40	10 ~ 20	2
216000001	202	12	217000293	202000106	04:50	22:50	05:40	23:50	10 ~ 15	2
216000002	66	12	216000190	217000739	04:40	04:50	06:00	06:10	10 ~ 9	2
216000003	71	13	217000293	216000260	05:20	22:40	05:20	22:40	10 ~ 15	2
216000004	62	13	234000501	216000532	05:20	22:30	05:00	22:30	10 ~ 12	2
216000006	122	13	234000620	216000098	05:20	22:30	05:00	22:30	8 ~ 10	2
216000007	123	13	216000222	217000576	05:20	22:00	05:20	22:00	20 ~ 30	2
216000008	71-2	13	216000222	216000267	05:45	21:30	06:30	22:20	180 ~ 180	2
216000009	722-1	13	217000665	217000666	07:00	19:30	정보 없음	정보 없음	60 ~ 120	2
216000010	66-1	13	216000053	233001901	05:05	23:00	05:10	23:00	10 ~ 15	2
216000011	55	13	216000053	234000743	04:45	22:50	04:50	22:50	8 ~ 10	2
216000016	62	13	216000053	234001001	04:50	23:10	04:50	23:10	6 ~ 10	2
216000017	33	13	216000053	216000672	07:50	18:40	06:10	20:15	180 ~ 180	2
216000019	31-2	13	225000140	211000232	04:50	22:40	05:30	23:40	10 ~ 20	2

설명

2. [정류장 추가]

정류장 아이디, 이름, 위치 등을 이용해 정류장 데이터 추가

정류장 아이디를 primary키로 사용

형식은 경기버스 api형식 사용

3. [전체 버스 정류장 데이터]

안산, 시흥 내 모든 정류장 정보 열람

항목 별 검색 가능

정류장 아이디를 클릭할 경우 정류장 정보 수정 페이지로 이동

정류장 이름을 클릭할 경우 정류장 위치 페이지로 이동

4. [정류장 정보 수정]

정류장 아이디를 제외한 정보 수정 가능

5. [정류장 위치]

정류장 위치 정보를 이용해 구글 지도에서 정확한 위치 확인 가능

6. [노선 데이터 보기]

안산, 시흥 내 운행하는 모든 버스 번호 열람

항목별 검색 가능

노선 아이디를 클릭할 경우 노선 정보 수정 페이지 이동

노선 번호를 클릭할 경우 노선 정보 상세 페이지 이동

7. [노선 정보 수정]

노선 아이디를 제외한 정보 수정 가능

8. [노선 정보 상세 보기]

구글 지도에 버스 운행 경로 표시 및 노선 내 정류장 정보 확인

정류장 이름을 클릭할 경우 정류장 위치 페이지로 이동

9. [통계 기능 열람]



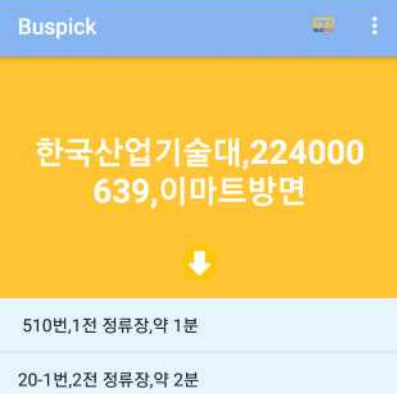
일자별 정류장, 노선 검색 결과 및 승차벨 요청 노선 정류장 횡수 열람 가능



## 6. 시험/테스트 결과





### Prototype 이후 추가 및 보완 기능

#### 1. (사용자용 앱) 검색 상세 페이지

화면	화면 상세보기	설명
		<p>1. [버스 검색] 기존 버스 검색 기능에서 버스 선택 시 해당 버스의 노선과 버스 정보를 출력함. 노선의 정류장을 클릭하면 정류장 세부페이지로 이동함. 상세정보 버튼을 누르면 운행지역, 시간, 배차간격을 확인이 가능함.</p> <p>2. [정류장 검색] 기존 정류장 검색 기능에서 해당 정류장의 방면정보를 추가. 정류장 선택 시 해당 정류장의 다음 정류장 정보, 도착할 예정인 버스 번호, 도착 예정 정보(정류장, 시간)가 리스트로 출력됨.</p>
		



## 2. (사용자용 앱) 하차알람

화면	화면 상세보기	설명
	  	<p>1. [검색] 검색한 정류장을 리스트로 출력.</p> <p>2. [경도, 위도] 정류장의 경도, 위도의 정보를 서버를 통해 받아와 출력.</p> <p>3. [내 위치정보 가져오기] 버튼을 누르면 현재 내 위치의 경도, 위도를 GPS 또는 NETWORK를 사용해 10초마다 갱신하며 출력. 정류장까지의 거리도 같이 갱신되어 계산되며 미터 단위로 출력됨.</p> <p>4. [하차알람] 정류장까지의 거리가 500m이하이면 다이얼로그로 하차알람 메시지가 출력되며 소리/진동이 울림. 상단바 푸시알람과 아이콘에도 알람이 표시됨. 다이얼로그의 OK 버튼을 클릭하면 모든 알람이 종료됨.</p>

### 3. (운전자용 앱) 실시간 운행 경로

화면	화면 상세보기	설명
		<p>1. [실시간 운행 경로] 실시간 위치 API를 받아 현재 이동하고 있는 위치 표시. (SequenceNum 활용)</p> <ul style="list-style-type: none"> <li>- 현재 운행 중인 버스위치 시퀀스와 승차예약번호 수신</li> <li>- 현재 위치 RecyclerView 내 버스 아이콘으로 출력</li> <li>- CountdownTimer() 메소드를 활용해 30초마다 요청 및 ViewHolder갱신</li> </ul>

## 7. Coding & DEMO

### Coding

#### 1. 서버

##### 1-1 API

클래스 목적 및 기능 : API 요청시 로그 기록

클래스 기능 정의 :

구분	메소드 명	기능
로그 출력	print_log	사용자용 어플에서 호출한 메소드명, 시간을 출력
데이터 추출	extractData	가공한 문자열에서 데이터값만 뽑아 반환

상세 :

메소드 원형	static void print_log(String methodName)	
리턴값	void	
함수명	print_log	
파라미터	methodName	호출한 메소드 명
정의		
로그 기록(어떤 API를 언제 사용했는지) 출력		

메소드 원형	static String extractData(String KeyValueString)	
리턴값	splitData	추출한 데이터
함수명	extractData	
파라미터	KeyValueString	추출할 데이터를 가진 문자열
정의		
Key=Value 형식의 데이터에서 Value 반환		

##### 1-2 busArrivalAPI extends API

클래스 목적 및 기능 : 버스 도착 정보를 얻고 가공, 원하는 데이터 반환

클래스 기능 정의 :

구분	메소드 명	기능
초기화	__init__	API 사용을 위한 값들 초기화
버스 도착 정보	busArrivalList	해당 정류장에 정차하는 모든 노선에 대한 정보 반환
	busArrivalItem	해당 정류장에 정차하는 특정 노선에 대한 정보 반환
문자열 처리	formating	HTML 태그 제거
정류장 정보 반환	extractSpecific	남은 정류장, 도착 예정시간, 차량 번호, 노선 아이디 반환
	extractPredict	도착 예정시간, 노선 이름 반환

상세 :

메소드 원형	void __init__(void)	
리턴값	void	
함수명	__init__	
파라미터	void	
정의		
API 정보를 얻기위한 url, 인증키 초기화		

메소드 원형	String busArrivalList(String stationID)	
리턴값	data	해당 정류장에 정차하는 모든 노선정보
함수명	busArrivalList	
파라미터	StationID	정보를 얻을 정류장 아이디
정의		
해당 정류장에 정차하는 모든 노선에 대한 정보(위치, 도착 예정시간, 빈자리, 저상버스 여부 등) 반환		

메소드 원형	String busArrivalList(String stationID, String routeID)	
리턴값	data	해당 정류장에 정차하는 특정 노선정보
함수명	busArrivalList	
파라미터	stationID	정보를 얻을 정류장 아이디
	routeID	정보를 얻을 노선 아이디
정의		
해당 정류장에 정차하는 특정 노선에 대한 정보(위치, 도착 예정시간, 빈자리, 저상버스 여부 등) 반환		

메소드 원형	String[] formating(String data)	
리턴값	format[]	Key=Value 형식으로 포매팅된 문자열 배열
함수명	formating	
파라미터	data	API를 통해 얻은 데이터
정의		
API로 받아온 데이터에서 HTML 태그 제거 후, Key=Value 형태로 가공		

메소드 원형	ArrayList<String> extractSpecific(String[] dataArr)	
리턴값	dataList	위치, 도착 예정시간, 차량번호, 남은 좌석 개수, 노선 아이디
함수명	extractSpecific	
파라미터	dataArr	Key=Value 형식으로 포매팅된 문자열 배열
정의		
위치, 도착 예정시간, 차량번호, 남은 좌석 개수, 노선 아이디를 담은 리스트 반환		

메소드 원형	ArrayList<String> extractRoute(String[] dataArr)	
리턴값	dataList	도착 예정시간, 노선 번호를 담은 리스트
함수명	extractSpecific	
파라미터	dataArr	Key=Value 형식으로 포매팅된 문자열 배열
정의		
도착 예정시간, 노선 번호를 담은 리스트 반환		

### 1-3 BusLocationAPI extends API

클래스 목적 및 기능 : 버스 위치 정보를 얻고 가공, 원하는 데이터 반환

클래스 기능 정의 :

구분	메소드 명	기능
초기화	__init__	API 사용을 위한 값들 초기화
버스 위치 정보 API	busLocationList	현재 운행중인 해당 노선에 대한 정보 반환
문자열 처리	formating	HTML 태그 제거
버스 위치정보 반환	extractStationSeq PlateNo	버스의 위치(정류장 순서), 차량 번호 반환
	extractSeq	특정 차량 번호의 현재 위치(정류장 순서) 반환
	getNearPlateNo	사용자가 요청한 정류장에서 가장 가까운 특정 버스의 차량 번호 반환

상세 :

메소드 원형	void __init__(void)	
리턴값	void	
함수명	__init__	
파라미터	void	
정의		
API 정보를 얻기위한 url, 인증키 초기화		

메소드 원형	String busLocationList(String routeID)	
리턴값	data	노선에 대한 정보
함수명	busLocationList	
파라미터	routeID	버스 노선 아이디
정의		
해당 노선에 대한 막차여부, 저장버스여부, 차량 번호, 차종, 차량 빈자리수 등 정보를 제공		

메소드 원형	String[] formating(String data)	
리턴값	format[]	Key=Value 형식으로 포매팅된 문자열 배열
함수명	formating	
파라미터	data	API를 통해 얻은 데이터
정의		
API로 받아온 데이터에서 HTML 태그 제거 후, Key=Value 형태로 가공		

메소드 원형	ArrayList<String> extractStationSeqPlateNo(String[] dataArr)	
리턴값	dataList	버스 위치, 차량 번호를 담은 리스트
함수명	extractStationSeqPlateNo	
파라미터	dataArr	Key=Value 형태로 가공된 데이터
정의		
API 가공 데이터를 이용해 운행중인 버스의 위치, 차량 번호를 반환		

메소드 원형	String extractSeq(String[] dataArr, String plateNo)	
리턴값	sequence	특정 차량의 현재 위치
함수명	extractSeq	
파라미터	dataArr	Key=Value 형태로 가공된 데이터
	plateNo	차량 번호 ex) 경기xx바xxxx
정의		
특정 차량 번호의 현재 위치(정류장 순서) 반환		

#### 1-4 DAO

클래스 목적 및 기능 : DB접근, 요청시 로그 기록용

클래스 기능 정의 :

구분	메소드 명	기능
DB 연결	connect	DB에 연결
DB 연결 해제	disconnect	DB 접근이 끝난 후, DB연결 해제
로그 출력	print_log	DB에 접근한 메소드명, 시간을 출력

상세 :

메소드 원형	void connect(void)	
리턴값	void	
함수명	connect	
파라미터	void	
정의		
DB 접속		

메소드 원형	void disconnect(void)	
리턴값	void	
함수명	disconnect	
파라미터	void	
정의		
DB 접속 해제		

### 1-5 Route\_StationDAO extends DAO

클래스 목적 및 기능 : 노선이 경유하는 정류장 관련 정보

클래스 기능 정의 :

구분	메소드 명	기능
DB 접근	getRouteStationList	특정 노선이 정차하는 노선 객체 리스트 반환
	stationCount	특정 노선이 정차하는 정류장 수 반환
	SeqToStationInfo	특정 노선이 정차하는 순번에 해당하는 정류장 정보 반환
	stationToSeq	특정 노선, 정류장 아이디에 해당하는 정차 순번 반환
	allRouteStation	모든 노선의 정차 정류장 리스트 반환
	findLine	특정 노선이 정차하는 정류장 아이디 반환

상세 :

메소드 원형	ArrayList<Route_Station> getRouteStationList(String routeID)	
리턴값	stationInfoList	특정 노선이 정차하는 노선 객체 리스트
함수명	getRouteStationList	
파라미터	routeID	노선 아이디
정의		
해당 버스 노선이 정차하는 노선 객체 리스트 반환		

메소드 원형	Integer stationCount(String routeID)	
리턴값	stationSeq	노선이 정차하는 정류장 개수
함수명	stationCount	
파라미터	routeID	노선 아이디
정의		
특정 노선이 정차하는 정류장 수 반환		

메소드 원형	Station seqToStationInfo(String routeID, int seq)	
리턴값	stationInfo	정류장 정보 객체
함수명	SeqToStationInfo	
파라미터	routeID	노선 아이디
	seq	정차 정류장 순번
정의		
특정 노선이 정차하는 순번에 해당하는 정류장 정보 반환		

메소드 원형	Integer stationToSeq(String routeID, int seq)	
리턴값	stationSeq	정차 정류장 순번
함수명	stationToSeq	
파라미터	routeID	노선 아이디
	stationID	정류장 아이디
정의		
특정 노선, 정류장 아이디에 해당하는 정차 순번 반환		

메소드 원형	ArrayList<String> allRouteStation()	
리턴값	R_SInfo	모든 노선의 정차 정류장이 담긴 리스트
함수명	allRouteStation	
파라미터	void	
정의		
모든 노선의 정차 정류장 리스트 반환		

메소드 원형	ArrayList<String> findLine(String routeID)	
리턴값	lineList	특정 노선이 정차하는 정류장 정보 리스트
함수명	findLine	
파라미터	routeID	노선 아이디
정의		
특정 노선이 정차하는 정류장 아이디 반환		

### 1-6 RouteDAO extends DAO

클래스 목적 및 기능 : 노선 관련 상세 정보

클래스 기능 정의 :

구분	메소드 명	기능
DB 접근	routeFind	특정 문자열이 포함된 모든 노선 아이디 반환
	routeInfo	특정 노선의 정보 반환
	getRouteID	노선 이름과 정확히 일치하는 노선 아이디 반환
	routeName	노선 아이디의 노선 이름 반환
	allRouteID	모든 노선 아이디 반환

상세 :

메소드 원형	ArrayList<String> routeFind(String routeName)	
리턴값	routeNameList	특정 문자열이 포함된 노선 아이디를 담은 리스트
함수명	routeFind	
파라미터	routeName	특정 노선의 문자열
정의		
특정 문자열이 포함된 모든 노선 아이디 반환		

메소드 원형	Route routeInfo(String routeID)	
리턴값	rInfo	노선 정보를 담은 Route 객체
함수명	routeInfo	
파라미터	routeID	노선 아이디
정의		
특정 노선의 정보 반환		

메소드 원형	String getRouteID(String routeName)	
리턴값	routeID	노선 아이디
함수명	getRouteID	
파라미터	routeName	노선 문자열
정의		
노선 이름과 정확히 일치하는 노선 아이디 반환		

메소드 원형	String routeName(String RouteID)	
리턴값	rName	노선 이름
함수명	routeName	
파라미터	RouteID	노선 아이디
정의		
노선 아이디의 노선 이름 반환		

메소드 원형	ArrayList<String> allRouteID(void)	
리턴값	routeIDList	모든 노선 아이디를 담은 리스트
함수명	allRouteID	
파라미터	void	
정의		
모든 노선 아이디 반환		

### 1-7 StationDAO extends DAO

클래스 목적 및 기능 : 정류장 관련 상세 정보

클래스 기능 정의 :

구분	메소드 명	기능
DB 접근	stationFind	특정 문자열이 포함된 모든 정류장 아이디 반환
	stationInfo	특정 정류장 정보 반환
	nearStationList	입력받은 좌표 주변 300m 내 모든 정류장 객체 반환
	allStationID	모든 정류장 아이디 반환
	getStationName	아이디를 통해 특정 정류장 이름 반환

상세 :

메소드 원형	ArrayList<String> stationInfo(String stationName	
리턴값	stationNameList	특정 문자열이 포함된 정류장의 아이디 리스트
함수명	stationFind	
파라미터	stationName	문자열 ex) 안산 or 산업
정의		
특정 문자열이 포함된 모든 정류장 아이디 반환		

메소드 원형	Station stationInfo(String stationID)	
리턴값	stationInfo	정류장 정보를 담은 Station 객체
함수명	stationInfo	
파라미터	stationID	정류장 아이디
정의		
특정 정류장 정보 반환		



메소드 원형	ArrayList<Station> nearStationList(String user_Lat, String User_Lon)	
리턴값	stationInfoList	300m 이내 모든 정류장 정보를 담은 Station 객체 리스트
함수명	nearStationList	
파라미터	user_Lat	사용자의 현재 위도
	user_Log	사용자의 현재 경도
정의		
입력받은 좌표 주변 300m 내 모든 정류장 객체 반환		

메소드 원형	ArrayList<String> allStationID(void)	
리턴값	allStationIDList	모든 정류장의 아이디를 담은 문자열 리스트
함수명	allStationID	
파라미터	void	
정의		
모든 정류장 아이디 반환		

메소드 원형	String getStationName(String stationID)	
리턴값	stationName	정류장 이름
함수명	getStation	
파라미터	stationID	정류장 아이디
정의		
아이디를 통해 특정 정류장 이름 반환		

## 2. 사용자/운전자 APP 공통

클래스 기능 정의 :

구분	메소드 명	기능
데이터	NetworkTask	서버로 필요한 데이터 url로 요청
	doInBackground	가공한 문자열에서 데이터 값만 뽑아 반환
	onPostExecute	받은 데이터를 필요에 따라 사용
분실물 웹게시판	loadUrl	분실물 게시판 웹 주소 호출

상세 :

메소드 원형	String doInBackground(Void.. params)
리턴값	result
함수명	doInBackground
파라미터	url, values
정의	
가공한 문자열에서 데이터 값만 뽑아 반환	

메소드 원형	void onPostExecute(String s)
리턴값	list
함수명	onPostExecute
파라미터	result
정의	
받은 데이터를 필요에 따라 사용	

### 3. 사용자 APP

클래스 기능 정의 :

구분	메소드 명	기능
위치	getLocation	사용자의 위도 경도를 받아옴
	onLocationChanged	사용자의 위치를 갱신함
	distance	사용자 위치와 정류장까지의 거리를 계산함
알람	createNotification	푸시 알림 생성
	removeNotification	푸시 알림 제거
권한	callPermission	앱에 필요한 권한 요청
블루투스	connectDevice	블루투스 페어링된 디바이스 탐색
	selectBluetoothDevice	블루투스 디바이스 연결

### 4. 운전자 APP

클래스 기능 정의 :

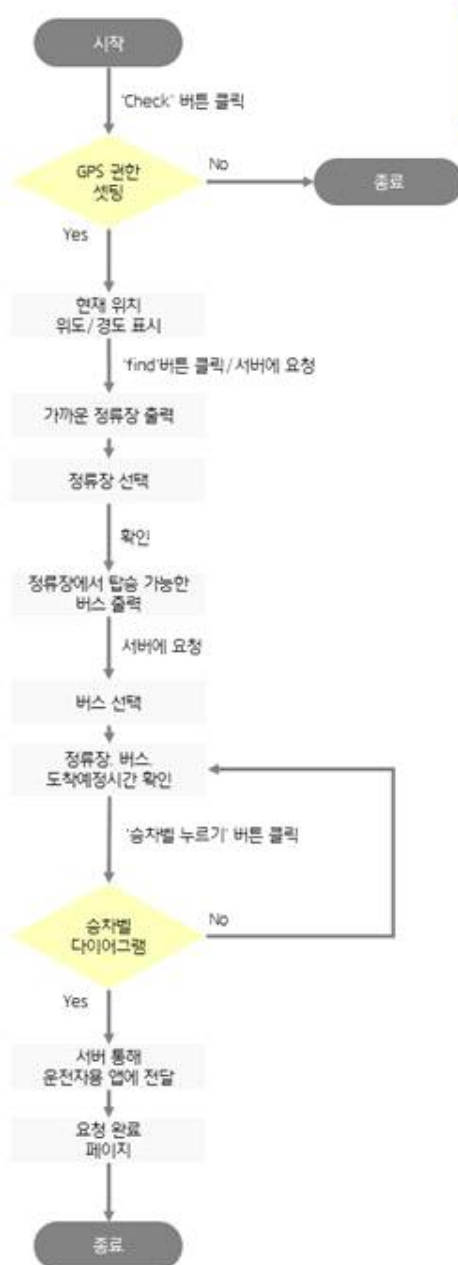
구분	메소드 명	기능
운전 경로 출력	CustomAdapter	버스 시퀀스, 현재 위치 시퀀스 전달
	getView	이미지, 리스트 데이터 구조화
	CountDownTimer	데이터 업데이트 간격 및 시간 지정
	ViewHolder	itemlist를 활용하기 위한 recyclerList사용
지도	onMapReady	GPS 위치 권한 확인
	startLocationUpdate	위치 권한 업데이트
	getCurrentAddress	운전자 현재 위치 좌표 업데이트
	setDefaultLocation	디폴트 위치 처리

### 5. 웹 게시판

구분	메소드 명	기능
Controller	MemberController	로그인&회원가입 동작 구현
	ProductController	분실물 업로드 및 목록 동작 구현
Domain	MemberVO	회원 멤버변수 및 생성자
	ProductVO	분실물 변수 및 생성자
Persistence	MemberDAO	회원 사용자 인터페이스와 DB연결 동작
	ProductDAO	분실물 사용자 인터페이스와 DB연결 동작
Service	Action	request/response forward
	ActionForward	
	MemberjoinAction	회원가입 동작 구현
	MemberListAction	회원 리스트 목록 확인
	MemberLoginAction	로그인 동작 구현
	ProductInfoAction	분실물 게시물 목록 확인
	ProductListAction	회원의 등록된 게시물 목록 확인
WebContent	ProductUpleadAction	분실물 게시물 등록
	*.jsp	사용자 결과화면 출력(ui)

## 승차벨 시나리오

### • FLOW CHART

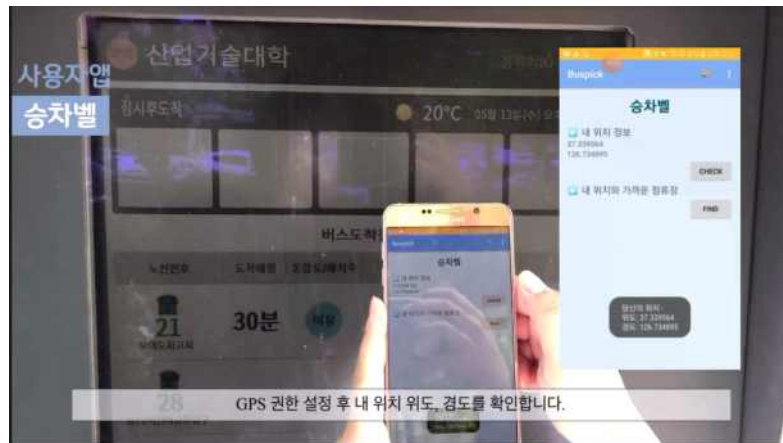


1. GPS권한 여부를 설정한 뒤 Check버튼을 클릭해 gps로 현재 위치 정보를 받아온다.
2. find버튼을 눌러 가까운 정류장List를 서버에 요청한다.
3. 수신한 List에서 탑승할 정류장을 선택하고 확인한다.
4. 해당 정류장에서 탑승가능한 버스 List를 서버로 요청한다.
5. 탑승할 버스를 선택하고 정류장 이름과 버스도착예정시간을 확인한다.
6. '승차벨 누르기' 버튼을 클릭하고 Dialog의 'Yes' 버튼을 누르면 서버로 전송되고, 운전자앱에 카운트되어 전달된다.
7. 요청 완료 페이지 확인 후 종료

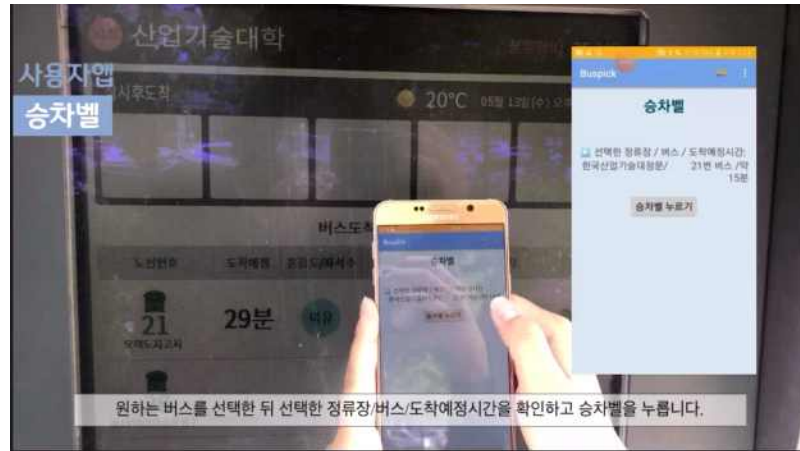
## 승차벨 DEMO

### 1. 사용자 APP

- 한국산업기술대 정문 정류장에서 승차벨 테스트 진행

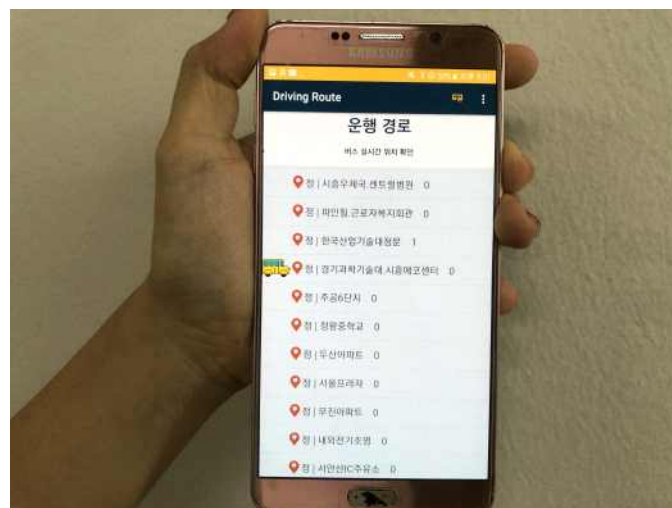


- GPS를 통해 사용자의 위도 경도 값 (37.339564/126.734895)을 출력하고 내 위치에서 300m 이내 정류장의 정류장 id와 정류장 이름이 출력되며 현재 위치인 '한국산업기술대정문' 정류장을 선택.
- 사용자가 선택한 정류장에서 탑승 가능한 버스의 번호와 예상 도착시간이 출력됨.



- 버스를 선택하고 정보들을 확인한 뒤 다이얼로그 '예' 버튼을 누르면 승차벨 신호가 전달됨.

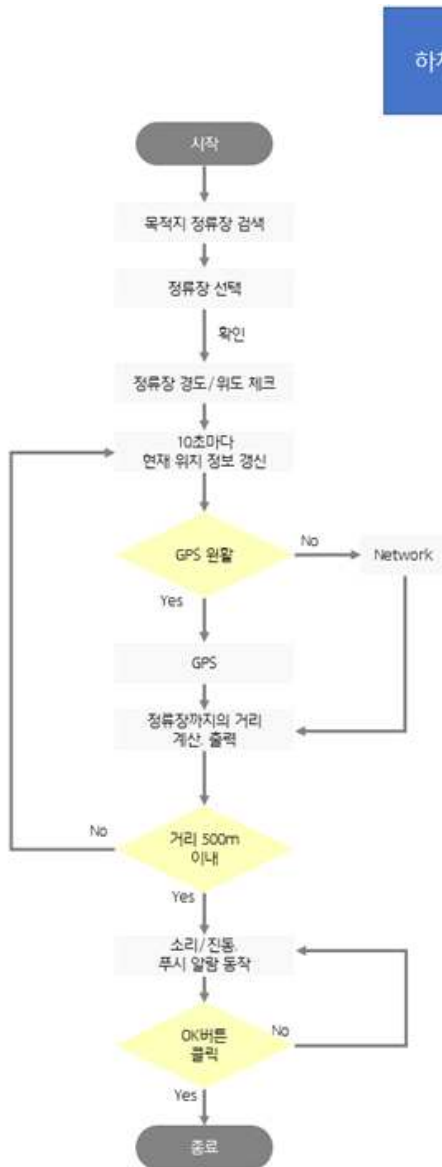
## 2. 운전자 APP



- 운전자 APP의 운행경로 화면에 정류장 이름 옆 승차 요청신호가 카운트 된 것을 확인할 수 있음.

## 하차알람 시나리오

### • FLOW CHART



1. 탑승자가 하차할 정류장의 이름을 검색한다.
2. 서버로부터 받은 정류장List에서 정류장을 선택한다.
3. 선택한 정류장 확인 후 해당 정류장의 경도/위도를 체크한다.
4. 10초마다 사용자의 현재 위치를 갱신한다.
5. GPS가 원활하지 않은 경우 Network정보를 이용해 실시간 위치를 얻어온다.
6. 정류장과 사용자의 거리를 계산해 출력한다.
7. 거리가 500m보다 먼 경우 위치정보를 계속 갱신한다.
8. 거리가 500m보다 가까워지면 소리/진동, 푸시알람이 동작한다.
9. OK버튼을 클릭하면 알람이 종료되고, 누르지 않을 시 푸시알람이 유지된다.



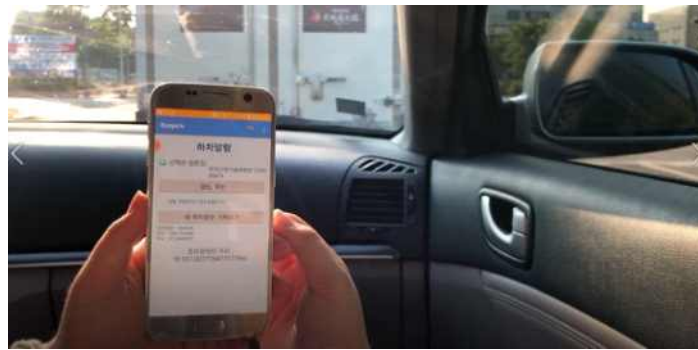
## 하차알람 DEMO

### 1. 사용자 APP

- 한국산업기술대 정문 정류장을 기준으로 하차 알람 테스트를 진행함.
- 정류장으로부터 600m 떨어진 위치에서 테스트 시작.



- 알람을 설정할 '한국산업기술대정문'정류장을 검색 후 선택.



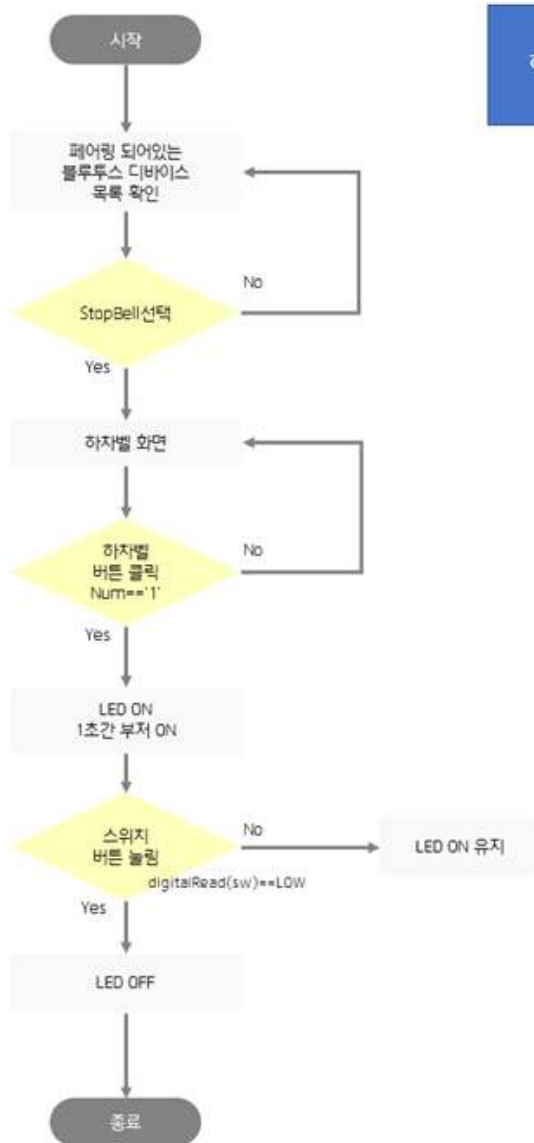
- 선택 정류장 이름을 확인한 뒤 버튼을 누르면 정류장의 경도/위도 (126.7335167/37.3387167)가 출력됨.
- '내 위치정보 가져오기' 버튼을 누르면 위치정보가 GPS 또는 Network를 통해 10초마다 갱신되고 해당 정류장과의 거리도 갱신되어 출력됨.



- 정류장으로부터 500m 이내로 접근하면 소리/진동과 함께 푸시 알림이 울리는 것을 확인.

## 하차벨 시나리오

### • FLOW CHART



1. 메뉴에서 하차벨을 클릭해 StopBellActivity로 이동한다.

2. '페어링 되어있는 블루투스 디바이스 목록'에서 'StopBell'을 선택해 아두이노 블루투스 모듈과 연결한다.

3. 하차벨 버튼을 클릭하면 아두이노에서 입력 신호 1을 받는다.

4. LED가 켜지고, 부저가 1초간 울린다.

5. 하차할 때 문이 열리며 스위치 버튼이 눌리면(`digitalRead(sw)== LOW`) LED가 꺼진다.



## 하차벨 DEMO

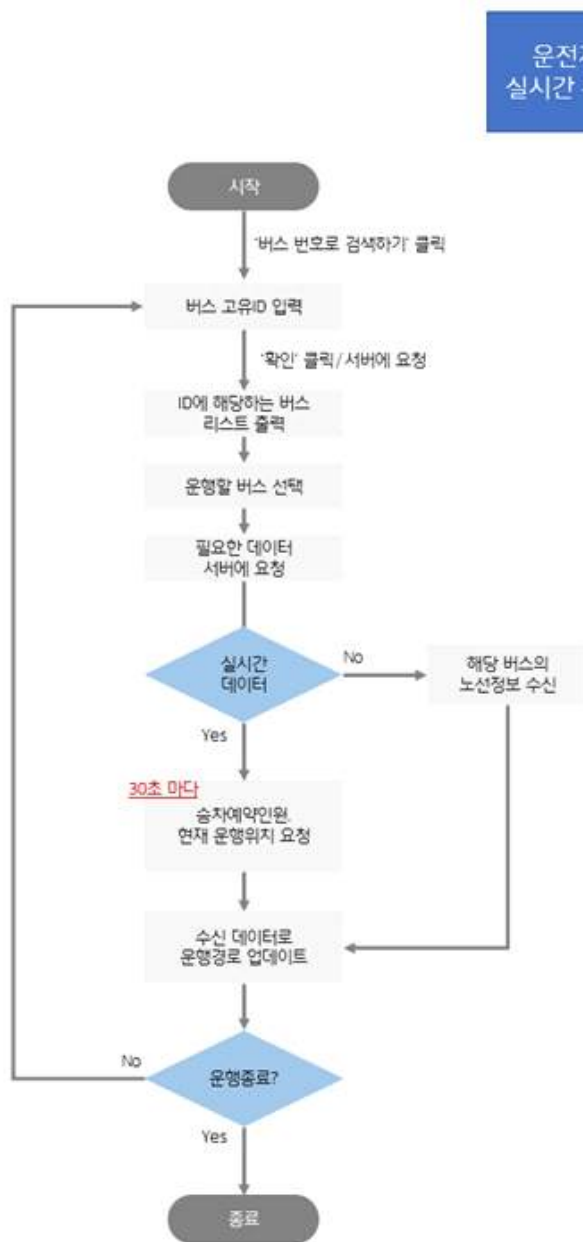
### 1. 하차벨 하드웨어



- 블루투스 페어링 후 사용자 APP의 하차벨 버튼을 누르면 LED가 켜지고 1초간 부저가 울림.
- 기존 하드웨어 외관을 보완하고 버튼을 누르기 위한 버스 모형 제작.
- 버스 문이 열릴 때 버튼이 눌리며 LED가 꺼지는 것을 확인 할 수 있음.

## 실시간 주행 시나리오

### • FLOW CHART



1. 버스 고유ID를 입력하여 서버로 송신한다.
2. 서버로부터 받은 ID에 해당하는 버스 정보를 Listview에 담고, 운행할 버스를 선택한다.
3. 선택한 버스의 필요한 데이터(노선정보, 승차예약인원, 현재 운행위치)를 요청해 서버로부터 받는다.
4. 데이터들을 RecyclerView에 담아 실시간 운행경로를 확인한다.
5. CountdownTimer를 통해 30초마다 승차예약인원 및 현재 운행위치 정보를 요청한다.
6. CountdownTimer를 통해 35초마다 데이터를 담은 ViewHolder를 갱신한다.
7. 운행이 종료되면, 다른 차량 운행을 시작하거나 앱을 종료한다.

## 실시간 주행 DEMO

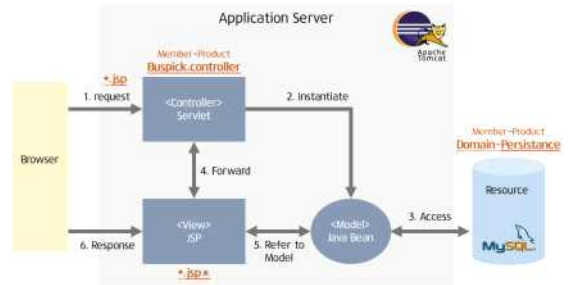
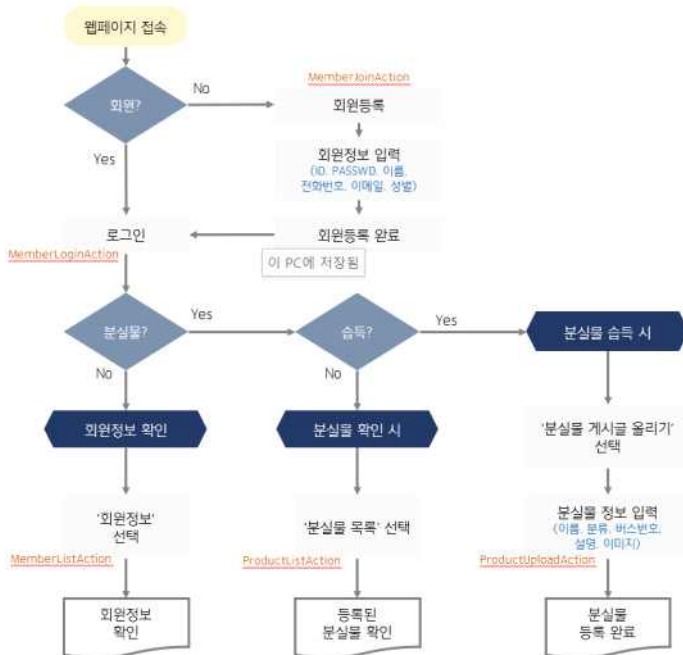
### 1. 운전자 APP



- 운전자 실시간 위치는 정문 정류장에서 테스트를 진행함.
- 운전자가 선택한 버스가 정류장 전광판에 표시된 위치정보와 앱의 정보가 일치함을 확인.
- 30초마다 갱신되어 버스가 정류장을 떠나면 다음 정류장으로 아이콘이 이동.

## 분실물 웹게시판 시나리오

### • FLOW CHART



1. PC/App을 통해 웹페이지에 접속한다.

2. 기존에 등록된 회원이면 로그인하고, 등록되어 있지 않다면 회원 가입한다.

(회원이름은 ID,P Passwd, 이름, 전화번호, 이메일, 성별을 입력하여 수행한다.)

3. (분실물 확인) '분실물 목록'을 선택하여 게시된 분실물들의 정보(이름, 분류, 버스번호, 설명, 이미지)를 확인한다.

4. (분실물 등록) 등록한 분실물을 등록하기 위해 '분실물 게시글 올리기'를 선택하여, 정보(이름, 분류, 버스번호, 설명, 이미지)를 입력하고 '등록'을 누른다.

5. (회원정보 확인) 회원들의 정보를 확인하기 위해 '회원정보'를 선택한다.

### III. 결론

#### 1. 연구 결과

##### 과제의 최종목표

■ 많은 승객이 불편함을 겪고 있는 버스 무정차 문제가 개선되지 않는 이유는 승객이 기사에게 명확한 승차 의사전달을 할 수 없기 때문임. 현재 승객이 승차 의사를 표현할 방법은 버스 기사를 향해 손을 흔드는 등의 원시적인 방법밖에 없으므로 이러한 문제를 승차 예약이라는 개념과 서비스를 도입하여 개선하려 함.

■ 승객이 버스에서 하차하기 위해서는 본인이 내려야 하는 정류장을 확인하기 위해 창밖을 주시하거나 도착 알람 방송을 듣고 현재 위치를 가늠해야 한다는 불편함이 존재하고, 많은 탑승객에 의해 하차 벨을 누르지 못하는 경우 버스 내 안내방송을 듣지 못하거나 잠이 들 경우 내려야 할 정류장을 지나치는 경우가 종종 발생함. 따라서 실시간 노선 안내 서비스와 하차 알람, 모바일 하차 벨 서비스를 도입해 이러한 문제를 극복하려 함. 덧붙여 분실물 게시판 및 버스 관리자 웹페이지(관리자 기능)를 활용해 이용자의 편의성을 증진하고자 함.

##### 최종연구결과 요약

###### ■ 기존 프로그램과의 차이점

·기존의 ‘카카오버스’에 검색기능, 실시간 버스 도착정보, 승차시간/하차시간 알람 등의 기능이 존재함. 그러나 BUSPICK은 ‘카카오버스’의 기능 외에도 운전자에게 승차 신호를 전달하는 승차벨, 앱을 통해 하차의사를 알리는 하차벨, 사용자의 편의를 위한 분실물게시판 등의 차별화된 기능이 탑재되어있음.

###### ■ 구현상의 문제점

·본 작품의 서버는 개인 PC에서 구현되었기 때문에 내부 망에서의 접속만 가능하고, 외부 망에서의 접속이 불가능 함. 향후 AWS나 클라우드 서비스를 이용한다면 다양한 사용자에게 서비스를 제공하여 상용화가 가능할 것으로 보임.

·BUSPICK의 운전자용 모바일 앱은 스마트폰을 통해 동작하기 때문에, 버스 운전자가 실제로 운행하며 사용하기에 적합하지 않음. 따라서 버스 운전자들이 사용할 수 있도록 다른 플랫폼(헤드업 디스플레이 등)을 이용해 개선하면 운행 중 실시간 서비스 이용이 원활할 것으로 예상됨.

###### ■ 향후 기술향상 방안

·현재 본 작품은 안산/시흥으로 지역이 제한되어 있지만, 추후 데이터를 보완하여 전국적 이용이 가능하게 될 것임.

·추후 사용자가 많아지게 되면 데이터가 축적되어 관리자 페이지의 통계기능을 통해 사람들의 BUSPICK 기능 이용 현황 정보를 얻을 수 있음.

## 주요 성과

"2020 컴퓨터공학부 종합설계 온라인 발표회" 학부생들이 참여한 투표에서 우수 작품 선정

## 핵심 단어

대중교통, 승하차벨, 편의성증진

## 2. 작품제작 소요재료 목록

### 상세 목록

- 아두이노 40핀 점퍼와이어 케이블 M-M타입 10cm
- 330옴 막대저항 20개
- 아두이노 우노 USB 전원, 통신 케이블
- 아두이노 우노 R3
- 아두이노 40핀 점퍼와이어 케이블, 점퍼선, 점퍼 MM,MF,FF,10cm,20cm,30cm
- 아두이노 캡 푸쉬, 푸시 탭트버튼 12mm
- 아두이노 블루투스 HC-06
- 30mm 피에조 부저
- 400핀 브레드보드
- 5mm 슈퍼플렉스 LED 빨간색
- 한글보드 아두이노 디지털 탭트 푸쉬 버튼

## 참고자료

순번	참고문헌 정보
1	- 버스 도착 알림 어플리케이션 (2018.06, 한국정보기술학회, 542-544, 맹지은, 권민하, 조예원, 최은서, 이만희, 강승석 )
2	- 비콘 기반의 버스 자동 승하차 시스템 구현 (2015.06, 한국통신학회, 1390-1391, 김준호, 이성원)
3	- 스마트폰 또는 이동통신 단말기를 이용한 하차벨 시스템 및 방법 (2013.11, Google Patents, 최병삼)
4	- 블루투스4.0 기반의 학원통학버스 승하차 관리 시스템 (2018.06, 한국정보과학회, 1769-1771, 김민혁 외 6명 )