|  |
| --- |
|  |
| **제18회 임베디드SW경진대회 개발완료보고서**  **[자유공모]** |
|  |

**□ 개발 요약**

|  |  |
| --- | --- |
| **팀 명** | THE FACO |
|  | |
| **작품명** | 대중교통(광역버스) 대기인원을 판별, 계산 후 더 정확한 최소시간 경로 지원 시스템 |
| **작품설명**  **(요약)** | 대중교통의 대기인원을 판별하기 위해 Open CV 를 이용하여 평균적인 대기인원의 모습을 정리하여 인원수를 판별하고, Open API 를 통해 알아낸 대중교통 정보를 대중교통 환승 장소의 대기 인원수와 함께 활용하여 경로를 추천한다. |
| **소스코드** | https://github.com/woojin1027/THE-FACO |
| **시연동영상** | < YouTube 링크 / 1개의 동영상 링크만 삽입 > |

**□ 개발 개요**

※ 프로젝트 개발 배경, 동기, 목표, 필요성 등 / 2page 이내로 작성

○ 개발 작품 개요 / 제안하는 작품에 대한 개요를 자세히 기술한다.

- 현재의 대중교통 길 찾기 앱들 중, 한국인의 대표적인 앱 두가지를 살펴보면, 약간의 경로추천 차이가 존재하기는 하지만, 최단 경로라는 것은 단지 대중교통을 이용할 때의 예상시간을 이용하여, 길 찾기 결과를 보여준다.



- 이러한 경로 계산은 평상시에는 문제가 되지 않지만, 사람들이 많이 몰리는 출 퇴근시간 특히, 신도시에서 서울로 출근하는 인원들이 광역버스를 이용할 때, 문제가 발생한다. 여러 정거장에서 이미 많은 대기 인원으로 인하여 버스들이 만원인 상태이므로 해당버스를 탑승하기 어려운 상황을 맞이하지만, 이를 길 찾기 앱에서는 확인해주지 않는다.

○ 개발 목표 / 개발 목표를 명확하게 제시한다.

- 기존의 대중교통 어플에서 제공하는 노선 안내와 대중교통 길 찾기 서비스를 넘어, 광역버스를 대기하는 인원수를 안내하고 이에 따른 경로 추천을 제공하는 것이 목표이다. 등하교 및 출퇴근 시간과 같은 탑승인원이 몰리는 시간에 발생하는 길 찾기 예상 소요시간의 오차를 줄일 수 있다.

- 코로나로 인해 사회적 거리두기 캠페인이 진행되고 있으나 출퇴근을 위한 좁은 정류장에서 사회적 거리두기를 실천하기 어렵다. 이 경우 사용자는 어플을 사용하여 대기 인원을 파악 후 대체 경로를 선택함으로써, 줄을 분산하여 설 수 있어 사람들 사이의 밀접 접촉을 줄이고자 한다.

○ 개발 작품의 필요성 / 개발하는 작품이 왜 필요한지에 대해 상세히 서술한다.

- 많은 직장인들과 학생들이, 각각 출퇴근 시간과 등하교 시간에 한 경로의 대중교통(광역버스)를 타기위해 긴 줄을 서며, 많은 시간을 줄을 서면서 보내게 된다. 실제로 대중교통을 이용하는 사용자들은 막연하게 긴 시간이 걸릴 것이라는 것만을 예상할 뿐 정확히 얼마나 기다려야 하는지, 몇 대의 버스를 대기 인원으로 인해 보내야 하는지 알지 못한다.

- 자신이 탑승할 환승 센터(정류장)에 직접 가지 않고서는 대기 인원이 얼마나 있는지도 알지 못하기에 한 정류장에서 대기할 때 감에만 의존해서, 지금 오는 버스들에 탑승 가능 한지를 판단한다. 어플을 사용한다면 직접 환승 센터(정류장)에 가지 않고서 정류장의 대기 인원을 알 수 있기 때문에 현재와 같은 코로나 사태로 인해 진행하는 사회적 거리 두기에도 긍정적 영향을 미칠 수 있다는 장점과 필요성을 설명할 수가 있다.

**□ 개발 환경 설명**

※ 최대한 자세하게 기술 / 15page 이내로 작성

○ Hardware 구성

- 라즈베리파이 4

- 카메라 센서 모듈

- paho mqtt

○ Hardware 기능 (제어 방법 등 서술)

- Raspberry PI 4 : 카메라 모듈을 통해 실시간으로 영상을 얻고 나면, 다음으로는 얻을 영상을 바탕으로 객체와 객체가 형성하는 줄을 인식하여 이를 판별해내는 과정을 라즈베리파이에서 진행한다.

앞서 설명한 바와 같이, 영상처리 과정은 PI의 CPU의 연산속도가 중요하기에 이를 해결할 수 있는 정도의 CPU 성능을 지닌 H/W 를 사용한다.

- Camera Module : 실시간 영상처리 과정을 위하여 영상을 확인할 카메라의 필요성을 얻게 되었다.

이를 위해 어느 정도의 화질을 보장하는 카메라 모듈을 이용하여 영상을 촬영해 영상에서의 객체를 판별하도록 한다.

- paho mqtt : 객체인식을 통해 얻은 데이터를 서버로 전송하기 위해 mqtt 를 이용하여 AWS 데이터베이스에 데이터를 제공함으로써 앱으로의 데이터 전송을 효율적으로 할 수 있게 한다.

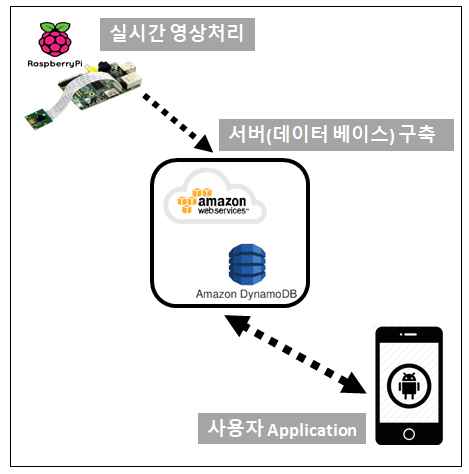
○ Software 구성

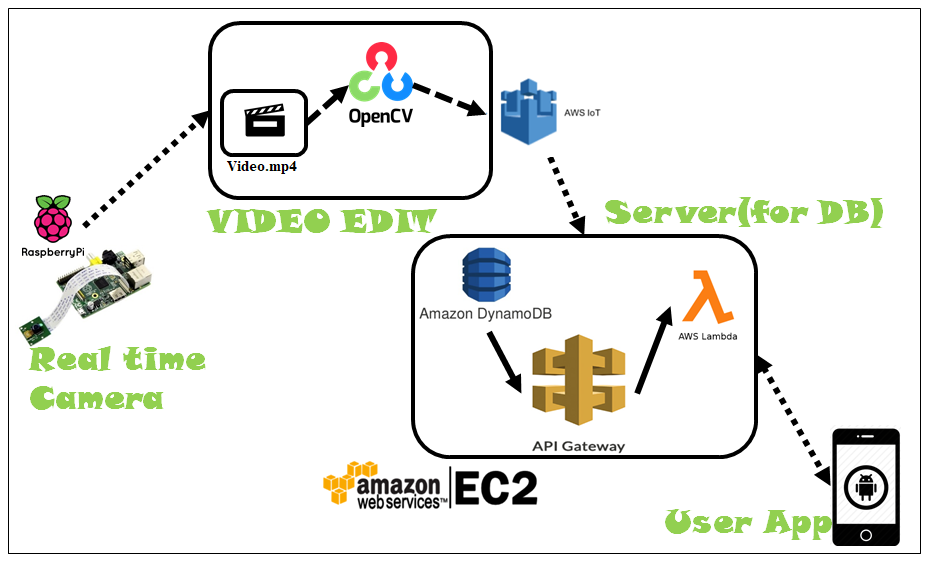
- Open CV

- Open API

- Smart Phone APP(Android APP)

○ Software 설계도





○ Software 기능 (필요 시 알고리즘 설명 포함)

- Open CV : 카메라 모듈을 통해 얻은 실시간 영상을 바탕으로 영상 내에서 사람 객체의 인식과 동시에 줄을 서고 있는 인원을 파악하여 데이터화를 진행한다.

- AWS DynamoDB : 영상처리한 데이터를 앱으로 보내기전에 데이터를 담아두는 역할을 한다.

- AWS API Gateway : AWS DynamoDB에 담겨진 대기인원에 대한 데이터를 앱에서 rest api 방식으로 호출하기 위해 사용한다.

- AWS Lambda : AWS API Gateway 를 사용하기 위한 간단한 rest api 함수를 get방식으로 앱에서 호출할 때 사용한다.

- Open API : 현재 대중교통의 실시간 파악을 위해 공공데이터포털의 Open API 를 이용하여 현재 해당 노선의 정보를 받아와 H/W를 통해 얻는 자료들과 함께 연산을 진행하는데 이용한다.

- Smart Phone APP : App 사용에 있어서 사용자가 편리하게 이용할 수 있는 서비스를 제공하기 위해 UI/UX 디자인을 고려하여 App을 제작한다.

○ 프로그램 사용법 (Interface)

1)

2)

○ 개발환경 (언어, Tool, 사용시스템 등)

- OS : 라즈비안, 안드로이드

- 개발환경(IDE) : MS Visual Studio Code, Android Studio

- 개발도구 : Android SDK, Java SDK, Maps SDK for Android

- 개발언어 : Python, Java, Node JS

**□ 개발 프로그램 설명**

※ 최대한 자세하게 기술 / 8page 이내로 작성

○ 파일 구성

- BackPressCloseHandler.java : '뒤로' 버튼을 두 번 연속 클릭 시 어플 종료

- BaseActivity.java , BaseActivity2.java : 여러 액티비티를 한번에 종료

- Bus\_items.java : M4102와 8100의 버스 아이템 셋팅

- BusitemAdapter.java : M4102의 버스 정류장 어댑터

- BusitemAdapter2.java : 8100의 버스 정류장 어댑터

- Frag1.java , Frag2.java , Frag3.java : bottomNavigation의 세 항목에 따른 Fragment 설정

- FgFirst.java , FgSecond.java , FgThird.java : 버스 선정이유 설명

- FgFirst2.java , FgSecond2.java , FgThird2.java, FgFourth2.java : 자세한 사용법 설명

- MainActivity.java : 메인 액티비티 설정

- map\_around\_busstop.java , map\_around\_busstop2.java : 현 위치 주변 정류장 안내

- Path\_items.java : 추천경로 아이템 셋팅

- PathitemAdapter.java : 추천경로 아이템 어댑터

- pathset\_mapshow\_start.java , pathset\_mapshow\_end.java : 선택한 정류장 위치 안내

- pathSetting\_start.java , pathSetting\_end.java : 출발, 도착 정류장 선택

- Reason.java , Reason2.java : 스와이프로 창 변환

- result.java : 추천 경로 안내

- showActivity.java : M4102의 노선, 버스 위치, 잔여좌석, 정류장 대기인원 안내

- subActivity.java : 8100의 노선, 버스 위치, 잔여좌석, 정류장 대기인원 안내

- splashActivity.java : 어플의 첫 화면

○ 함수별 기능

- setFrag(int n) : 하단 네비게이션을 선택하면 프래그먼트가 교체되도록 함. app시작 시 디폴트 프래그먼트는 첫 번째인 '버스'이며, 케이스는 총 3개로 각각 Frag1, Frag2, Frag3로 교체된다.

- DataCalculate, TextSet : api데이터, aws서버로부터 받은 대기인원수와 발생 가능한 경우의 수를 고려하여 조건문을 돌려 버스의 탑승 가능 여부를 파악하여 "00명 탑승 가능", "탑승불가", "출발대기"로 분류하여 안내함.

- getCurrentAddress(LatLng latlng) : 현재 사용자의 위치를 경위도 값으로 받은 뒤 지번 주소로 변환한다.

- busStationAroundList : '경기버스정보\_정류소조회 api'를 통해 주변 정류소 데이터를 파싱한다.

- getStationLocationList : '경기버스정보\_정류소조회 api'를 통해 해당 정류소의 위치 데이터를 파싱한다.

- toastshow(String string) : string에 텍스트를 지정하면 app디자인과 어울리도록 커스터마이징한 토스트를 텍스트와 함께 화면 하단에 띄운다.

- onBackPressed : '뒤로' 버튼을 한 번 누르면 안내 토스트가 띄워지고 2초 이내에 한 번 더 누르면 app이 종료된다.

- actFinish : 길 찾기 경로를 설정하는 과정으로 인해 쌓인 액티비티를 한꺼번에 종료한다.

-

○ 주요 함수의 흐름도

- 버스의 현 위치, 잔여좌석, 정류장 대기인원수, 탑승 가능 여부 안내 : 경기도에서 제공하는 api 중 버스도착정보목록(getBusArrivalItem), 버스위치정보조회(getBusLocationList), 버스도착정보조회(getArrInfoByRouteAllList)를 통해 데이터를 얻고 aws 서버로부터 각 정류장에 대기하고 있는 인원수를 받아온다. DataCalculate함수로 연산을 진행하고 TextSet으로 화면에 표시한다.

- TextWatcher : 정류장명과 정류소번호가 담긴 hashmap 데이터에서 정류장 검색 기능을 사용하기 위해 beforeTextChanged(검색창에 입력 데이터가 없을 때 기본 세팅), onTextChanged(검색창에 데이터를 입력하고 있는 도중에 실행될 함수), afterTextChanged(검색창에 입력을 마쳤을 때 이전 문자열에 비해 데이터가 얼마나 바뀌었는지 알려줌)를 사용한다.

- 주변 정류소 안내 : 내 위치를 파악하고 주변 정류장의 정류장명과 정류소번호를 파악하여 지도에 표시한다. 내 위치는 LocationCallback으로 확인한 경도, 위도 값에 따라 초록색 마커로 표시하며 반경 500m를 원으로 표시한다. 주변 정류장은 '경기버스정보\_정류소조회 api'의 'getBusStationAroundList'를 통해 파악하고 빨간색 버스형태의 마커로 모두 표시하며 클릭 시 정류장명과 정류소번호를 안내한다.

- 내 위치 주소 안내 : 주소 좌표 변환 툴인 Geocoder로 GPS 좌표로 취득한 경위도 좌표를 지번 주소로 변환한다.

- 정류소의 위치 지도에 표시 : 사용자가 hashmap에서 출발 및 도착 정류장을 선택하면 '경기버스정보\_정류소조회 api'의 'getBusStationList'를 호출하여 응답 메시지와 정류장명과 정류소번호를 비교한다. 만약 일치한다면 지도에 분홍색 마커로 표시하고 마커 클릭 시 위 정류장을 출발 및 도착 정류장으로 선택할지 여부를 묻는 alert 창을 띄운다.

- 대중교통 길 찾기 기능 : 사용자가 출발 정류장과 도착 정류장을 설정하면 최단 시간 순으로 대중교통 경로를 안내한다. 출발지 경위도 정보와 도착지 경위도 정보를 인텐트로 보내어 경기도에서 제공하는 api 중 대중교통환승경로조회(getPathInfoByBusNSubList)를 사용하여 데이터를 파싱한 후 각각의 데이터를 카드뷰에 담아 화면에 표시한다.

- gif파일 실행 : 안드로이드 스튜디오의 Palette만으로는 gif를 실행할 수 없기 때문에 'Glide' 라이브러리를 다운받아 Glide.with().load()를 호출하여 gif를 추가한다.

○ 기술적 차별성

1) 기존 길 찾기 정보에서 대기 인원을 통한 연산으로 더 정확한 교통 이용시간 판단

- 기존의 대중교통 길 찾기 앱에 비교하여, 광역버스의 이용 시 대기 인원을 판별함으로써 다음 버스를 이용할 수 있는지를 알 수 있다는 장점이 존재한다. 기존에 앱에서는 대기인원 파악이 없어 광역버스의 예상시간이 출퇴근 시간에는 부정확해지는 문제점을 해결할 수가 있다.

2) 기존에 길 찾기 앱에서는 존재하지 않는 차별성으로 인한 새로운 특징

- 특정 정류장의 대기인원을 알아내 제공함으로써 이용자가 직접 정류장에 가지 않아도 대기인원을 알 수가 있게 됨으로써 현재와 같이 사람들의 붐비는 장소가 부담스러운 상황에 이를 미리 살펴보게 됨으로써 이용자에게 다른 방안을 생각해보는 기회가 주어진다는 장점이 있다.

**□ 개발 중 발생한 장애요인과 해결방안**

※ 개발 과정에서 나타났던 모든 장애 요인(Risk)들을 나열하고, 이러한 장애요인들이 발생했던 경우 어떻게 해결했는지 구체적으로 제시 / 1page 이내로 작성

○ 프로젝트 설계 측면

- 최초의 프로젝트를 진행할 때, 모았던 인원들 중에서 중간에 하차하는 인원들이 발생함으로써 발생하는 인원 공백과 이로 인한 프로젝트 분량의 재분배를 확인하게 되었다. 이러한 문제는 최초의 팀 구성부터 프로젝트의 분량대비 인원의 부족이 문제점으로 꼽혔던 우리 팀의 입장에서는 상당한 문제점으로 작용하였다. 하지만 이러한 문제점을 결국에는 팀원들과 분배에 대해 서로 의논하며, 최종적으로는 성공적으로 마무리 짓게 되면서 문제점을 해결할 수 있었다.

○ 작품 개발 측면

- 영상처리를 이용하기 위해, OPEN CV를 이용하였는데 이론적으로 생각하였던 알고리즘의 이용이 어려움 상황을 맞이하였다. 이유는, 하드웨어의 연산을 라즈베리파이를 이용하여 알고리즘의 연산을 진행하려 하였는데 이를 이용하기 위해서는 상당한 연산처리 속도가 필요한데, 이를 라즈베리파이로는 감당하기 어려운 상황을 맞이하였다. 이는, 실제 CCTV에서도 먼저 연산을 진행하여 검출한 인원파악만을 데이터화하여 전송하려는 프로젝트의 입장에서 문제점으로 다가오게 되었다. 이때, 연산처리가 가능한 정도의 성능의 알고리즘을 알아내어 이를 이용함으로써 프로젝트의 이론적 계획과 현실에서의 문제점을 확인, 해결할 수가 있었다.

- 공공 API의 경우, 버스의 API가 종종 업데이트되면서 해당 프로젝트로 구현화 하려던 해당 정류장의 API ID가 변화하거나, 없어지고, 특히나, 최근 들어 코로나 바이러스(C19)로 인하여 극단적으로는 노선의 변화가 생김으로써 발생하는 문제점이 발생하였다. 이러한 문제점은 프로젝트의 진행에서 구현한 노선의 변경을 해야 할 정도의 큰 문제점이었는데, 이러한 문제점들을 해결하기 위해, 공공데이터 포털에 여러 번 문의를 하고, 최대한 많은 이용객이 이용하는 노선을 구현하여 변경의 가능성을 최소화하는 노선으로 프로젝트의 구현화를 진행하는 등 문제점을 해결하기 위해 노력하였다.

**□ 개발결과물의 차별성**

※ 개발한 결과물과 기존 발표된 유사작품(제품) 간 차별성 및 우수성 설명 / 1page 이내로 작성

○ 일반인이 사용할 시 이점

- 출퇴근(등하교) 시간에 많은 인원들이 하나의 노선에 몰리게 되면서, 대중교통 길 찾기 서비스에서 보여지는 것처럼, 바로 오는 버스를 탑승하지 못하고 다음버스를 타게 되는 상황을 실제로 알려주는 서비스를 제공할 수가 있다.

- 특히 신도시의 경우, 아직 지하철이 개통되지 않아 거의 2030년까지 대중교통 중 광역버스만을 이용하여, 출퇴근(등하교)을 해야 한다. 특정 시간대에는 신도시의 대다수 인원들이 직장(학교)를 위하여 버스 정류장에서 광역버스를 타기 위하여 줄을 서기 때문에 위 프로젝트의 아이디어가 큰 효과를 발휘하리라 생각한다.

○ 코로나에 미치는 긍정적인 영향

- 현재 C19(코로나 바이러스)로 인하여, 일상 속에서 사회적 거리두기를 권장하고 있다. 그러나, 대중교통을 이용하여, 직장(학교)로 매일같이 출퇴근을 해야 하는 사람들의 경우, 앞서 권장하는 거리두기를 지키기가 다소 어려운 상황임을 알고 있다. 이러한 환경에서 위 프로젝트 아이디어가, 사회적 거리두기의 실행을 돕는데 큰 역할을 할 수 있으리라 생각한다.

- 실제 버스정류장에 가지 않더라도, 현재 가려는 버스정류장의 인원이 어느 정도인지 알수 있다. 또한 이러한 서비스를 통해, 한 노선으로만 사람이 몰릴 수가 있는 상황에서 가장 적절하게 여러 노선을 추천함으로써 대중교통의 대기 인원들의 분산을 촉진할 수 있다.

- 만약 목적지까지 가는데 노선의 대기인원이 많아서 많은 시간을 줄을 서면서 기다려야 한다면, 이를 실제로 어느정도 시간이 걸릴지를 알려줌으로써 그 사이 시간에 다른 경제적 활동을 하는 것을 추천해 줄 수가 있다. 이러한 추천을 통해, C19로 인하여 많은 경제적 활동이 급감한 현재, 조금이나마 현상황을 변화시키는데 영향을 줄 수 있으리라 생각한다.

○ 버스 회사의 참고 자료로 활용

- 현재 광역버스를 운영하는 회사 입장에서, 그동안 놓치면서 보냈던 해당 버스를 기다리다가 다른 경로를 이용하는 인원들을 확인할 수가 있기에 이를 활용하면 기존의 정보보다 더 정확하게 해당 버스를 이용할 수 있는 인원들의 수를 파악하면서 버스 배차 간격의 조절을 더 정확하게 할 수가 있으리라 생각한다.

**□ 개발 일정**

※ 실제 프로젝트 개발 일정 작성 / 1page 이내로 작성

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **내용** | **2020年** | | | | | | | |
| **2月** | **3月** | **4月** | **5月** | **6月** | **7月** | **8月** | **9月** |
| 1 | 문제점 인식 및 해결방안 고안 |  |  |  |  |  |  |  |  |
| 2 | 필요한 기술 및 라이브러리 검색 |  |  |  |  |  |  |  |  |
| 3 | Device에 따른 SDK 구분 |  |  |  |  |  |  |  |  |
| 4 | 알고리즘 순서도 작성 |  |  |  |  |  |  |  |  |
| 5 | 영상처리 및 대기 인원 산출 |  |  |  |  |  |  |  |  |
| 6 | 라즈베리파이, app, 서버의 연동, wifi 모듈과 app간 통신 시스템 구축 |  |  |  |  |  |  |  |  |
| 7 | Smart Phone UI설계, Open Api 파싱 |  |  |  |  |  |  |  |  |
| 8 | 각 Device에 빌드하여 성능, UI피드백 |  |  |  |  |  |  |  |  |
| 9 | 완성, 시제품 테스트 |  |  |  |  |  |  |  |  |

**□ 팀 업무 분장**

※ 프로젝트 개발 관련 팀원의 업무 분장 위주로 작성 / 1page 이내로 작성

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **구분** | **성명** | **참여인원의 업무 분장** |
| 1 | 팀장 | 김우진 | 프로젝트 진행 관련 일정을 관리하고 회의를 진행하였다. 프로젝트에서 하드웨어 분야, 특히 영상처리를 통해 객체를 인식하는 알고리즘을 사용하여 광역버스 대기 인원파악에 알맞도록 제작, 수정하였다. |
| 2 | 팀원 | 이유진 | App의 전반적인 UI/UX 디자인 분야를 맡았다. Open API를 다루어 현재 사용자의 위치 기반의 주변 정류장 파악 및 선택한 정류장 정보 안내를 담당하였다. |
| 3 | 팀원 | 이채민 | AWS 서버와 App이 통신할 수 있도록 설정하였다. Open API를 다루어 버스의 현 위치에 따른 탑승 가능 여부 및 대중교통 경로 안내를 담당하였다. |