0.1 동작과정설명

0.1.1 Facebook 연동

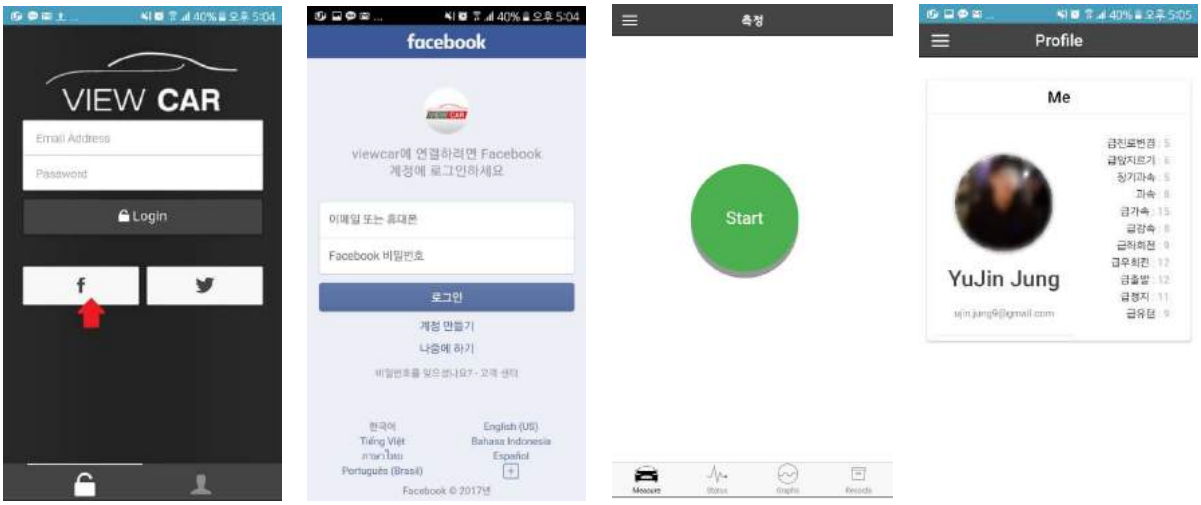


그림 26 Facebook 관련 UI

- Facebook 연동을 위해 cordova plugin 인 openFB 를 사용한다.
- openFB 는 Facebook Login 연동을 위한 Plugin 이며 inAppBrowser 기반으로 작동한다.

  Facebook SDK 없이 작동하기 때문에 Facebook에서 제공하는 모든 기능을 사용하기 어렵지만 Facebook SDK 및 Cordova 의 version 및 플랫폼에 영향을 받지 않는다.

  프로젝트에서 필요한 로그인 기능과 프로필 사진, 이메일, 이름을 제공 받을 수 있기 때문에 version 및 플랫폼에 영향을 받지 않는 openFB를 사용한다.
- Facebook Login이 성공할 경우 스마트폰에 이름, 이메일, 프로필 사진 이 저장되고 그 정보를 바탕으로 운행 정보를 저장 및 제공한다.

0.1.2 Sensing

- Sensing을 하기위해 스마트폰의 Accelerometer, Gyroscope, GPS 센서를 사용한다.
- 3축 Gyroscope 센서는 ω_x, ω_y 및 ω_z라고 각각 명명된 센서 프레임의 x, y 및 z 축에 대한 각속도를 측정한다.
- 3축 Accelerometer 센서는 ω_x, ω_y 및 ω_z라고 각각 명명된 센서 프레임의 x, y 및 z 축에 대한 중력 가속도를 측정한다.
- GPS 센서는 GPS 위성을 활용해 현재의 위치와 시간을 측정한다.

0.1.3 Sensor Fusion

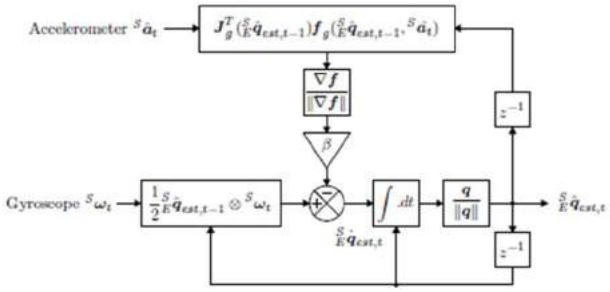


Figure 2: Block diagram representation of the complete orientation filter for an IMU implementation

그림 27 IMU Block diagram

- Sensor Fusion은 Madgwick Filter Algorithm 기반으로 한다.
- Madgwick Algorithm 은 이전에 측정된 값을 기반으로 현재 측정된 값의 노이즈를 최대한 줄여주는 Algorithm으로 Kalman Filter Algorithm의 개선된 형태이다.
- 운전할 때 차량의 속도나 차량의 회전 각도는 선형적인 데이터가 아닌 변화의 정도가 급격하게 변하기 때문에 비선형적인 데이터가 나온다. 따라서 Kalman Filter가 아닌 Extended Kalman Filter를 적용해야 한다.
- Madgwick Algorithm은 가속도 센서에 Adaptive Step인 Gradient Descent 알고리즘을 적용하여 Extended Kalman Filter에 비해 비선형 데이터를 정확하게 측정하고 계산과정에 있어서 메모리나 시간을 더 적게 차지한다는 장점을 가지고 있다.

0.1.4 Calibration

- Calibration은 기준 프레임을 바꾸는 것이다.
- 측정하려는 물체가 지평면과 평행하게 있지 않고 틀어져 있다면 이는 Calibration이 필요하다고 할 수 있다.
- Calibration 이 되면 이 물체는 틀어져 있는 상태가 지평면과 평행한 상태라고 인식하면서 기준 프레임을 물체가 원래 있던 상태를 기반으로 Yaw값의 측정이 이루어진다.
- Gyroscope 센서와 Accelerometer 센서로부터 100msec 주기로 각속도와 가속도 값을 AD변환하여 읽어온다. 이 데이터는 내부 버퍼에 누적되어 기록된다.
- Madgwick 필터 역시 100ms로 동작하면서 가속도와 각속도 값을 융합하여 Quaternion을 지속적으로 업데이트한다. 6초 이후 Rotation matrix의 직교성을 유지하는 알고리즘이 수행된다.

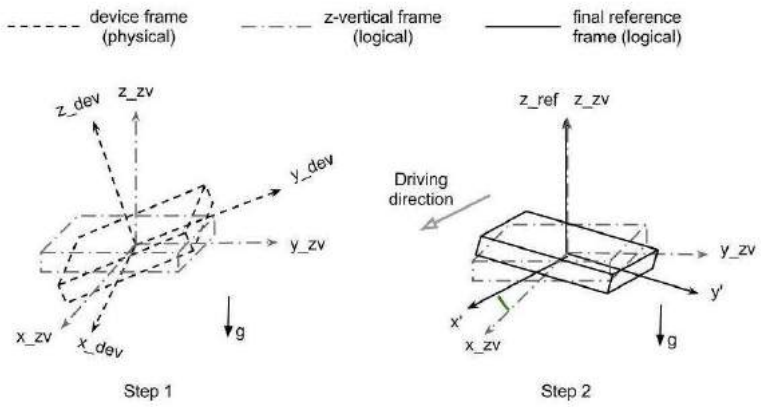


그림 28 Calibration

- 메커니즘은 위 그림과 같다. Beta[1]는 0에서 1사이의 값을 갖는데 beta가 클수록 빠르게 자세를 찾을 수 있지만 정확성은 떨어진다. 반면, beta가 작을수록 정확성은 증가하며 자세를 찾는데 드는 시간이 오래 걸린다. 따라서 2초간 beta를 1로 setting하여 현재 자세를 빠르고 부정확하게 찾은 후 그 후 IMU의 최적 beta값인 0.033로 setting하여 느리고 정확하게 찾는다.

0.1.5 GPS 데이터 수집 및 속도 측정

- GPS의 좌표를 이용한 속도API를 이용하여 speed와 timestamp을 수집한다.
- Time interval을 1초간격으로 준다.

0.1.6 속도를 가속도로 변환

- GPS의 speed를 이용하여 가속도로 변환하기 위해 Queue를 이용한다.
- 이전 speedQueue와 현재 speedQueue의 차이를 이용하여 가속도를 구한 후 이전 Queue를 제거 한 후 1초 후 Queue를 추가하는 작업을 실행한다.

0.1.7 각도 측정

- 100ms당 가속도센서와 자이로센서의 출력되는 값들을 Madgwick filter를 통해 보완된 쿼터니언 값을 추출한다.
- 이 쿼터니언은 지구 프레임에 대한 센서 프레임의 회전이므로 Calibration을 통해 틀어진 센서 프레임에 대한 측정 센서 프레임의 회전을 측정할 수 있게 한다

[1] Beta는 가속도계에서 얻은 데이터를 그래디언트 디센트 알고리즘과 Jacobian을 통해 계산하여 자이로스코프에서 측정되는 데이터의 오차를 줄여주기 위해 반복적으로 빼주기 위한 값이다.

- 측정된 쿼터니언을 오일러각으로 변환시켜 yaw를 산출하는데 이 yaw 값은 100ms당 회전한 각도이다.
- 이를 두 개의 큐에 넣어서 비교를 하고 비교한 값을 임의의 큐에 넣는다.
- 큐의 개수에 따라 누적 시간 동안의 회전각을 구할 수 있다.

0.1.8 10대 위험 운전 행동 판정

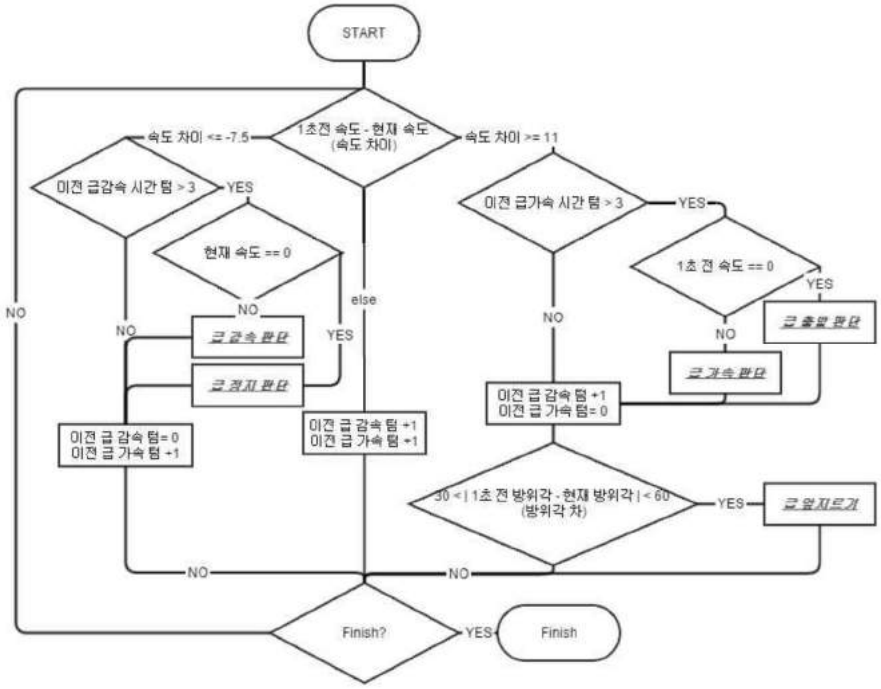


그림 29 판정 알고리즘 Flow Chart(1)

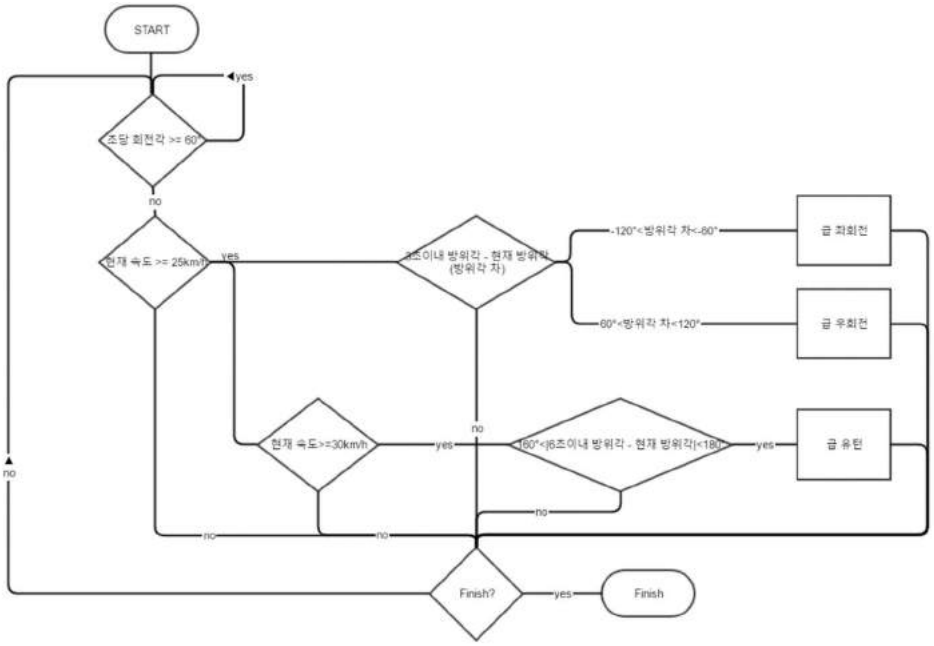


그림 30 판정 알고리즘 Flow Chart(2)

그림 31 판정 알고리즘 Flow Chart(3)

0.1.9 클러스터

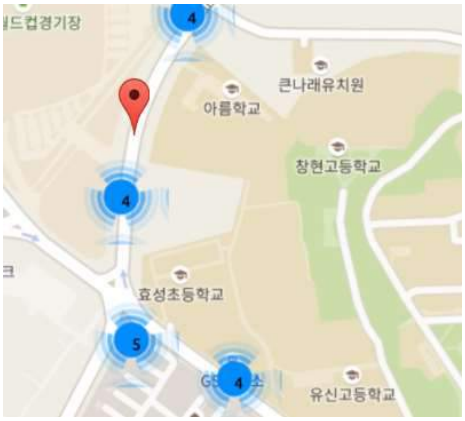


그림 32 Cluster

특정 구간에서 같은 위험 행동이 여러 번 발생 할 경우 경고 메시지를 TTS로 알려주는 기능을 구현하기 위해 데이터를 분석하는 도구로써 클러스터링을 사용하였다. 좌표 기반으로 주변의 위험 행동 들을 묶어주어야 하기 때문에 다른 데이터 분석 방법보다 효율적이라고 생각했다.

Google에서 제공하는 Map 기반 Clustering 도구를 사용하였고 Google에서는 클러스터링을 진행하여 마커를 표현해주는 데에 목적이 있지만 이번 프로젝트에서 요구하는 기능은 클러스터링을 통해 생성된 클러스터 주변으로 진입하게 될 경우 경고 메시지를 TTS로 알려주는 것이기 때문에 파일을 받아 소스코드를 추가 및 삭제하였다.

- Database에 있는 위험 행동 리스트를 스마트폰에 저장
- 현재 좌표를 기준으로 주변에 위험 행동을 한 곳의 좌표의 마커 생성
- 각 마커들 간 클러스터링 진행
- 각 클러스터 반경 70m 안으로 진입할 경우 위험 행동 다발 구간임을 TTS로 알린다.
- TTS로 알렸다는 것을 표시하고 일정 거리 이상 멀어 지기 전까지 해당 클러스터에 접근하더라도 TTS로 알리지않는다.
- 일정 구간 멀어 졌을 경우 표시를 제거하고 다시 재진입 할 경우에는 TTS로 음성 안내를 실시한다.

0.1.10 트래커를 통한 측정 정확성 확인

그림 33 트래커 관련 UI

사용자에게 운행 정보 및 위험 행동이 발생한 구간, 속도, 가속도, 각속도의 측정 그래프를 제공하기 위하여 측정 시작 시 해당 데이터를 데이터베이스에 저장한다. 사용자는 운행이 끝난 뒤 운행정보를 확인 할 수 있고 트래커의 하단에 존재하는 그래프를 통해 속도, 가속도, 각속도 측정 정확성을 확인 할 수 있다. 그래프는 Highchart 의 spline chart 를 이용하였다.

- 운행 중 운행 좌표, 속도, 가속도, 각속도 Database 에 저장
- Records Tab 으로 이동
- Records Tab 새로고침 혹은 처음 로드 시 Database 에 저장되어 있는 운행 기록을 스마트폰에 저장
- 운행 기록 중 정확성을 확인하기 위한 기록 클릭
- Google Map, Marker, Polyline 및 Highchart Spline Chart 호출 및 출력
- 지도 및 속도/가속도/각속도 그래프를 통한 정확성 확인

0.1.11 운행 종료 후 운행 정보 확인

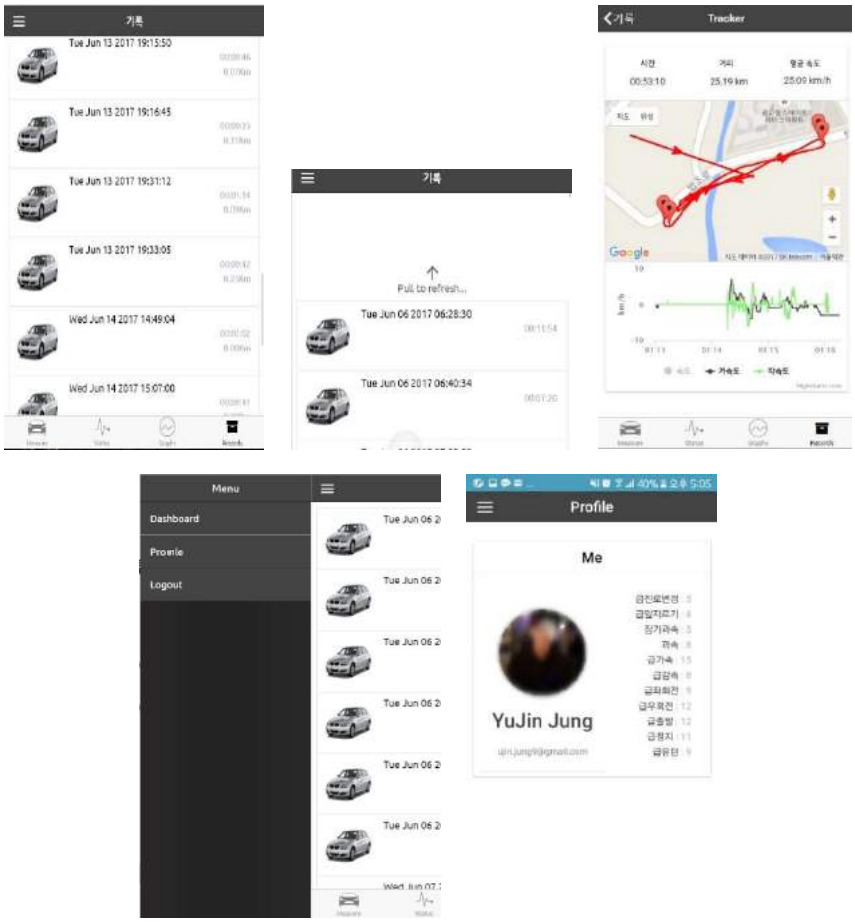


그림 34 운행 기록 관련 UI

운행이 종료 된 후 Records Tab으로 이동하면 운행 기록이 시간대 별로 정렬 되어있다. Records Tab에서 화면을 아래로 당기면 운행 기록을 새로 고침 할 수 있다. 확인하고자 하는 운행 기록을 클릭하면 트래커 화면이 나타나고 운행 시간, 운행 거리, 운행 평균 속도를 제공 받을 수 있고 트래커를 통해 운행 기록을 분석해 볼 수 있다. 그리고 왼쪽 상단의 ≡를 클릭하면 나타나는 프로필 탭을 선택하면 현재 모든 운행 구간에서 총 발생한 위험 운전의 숫자를 제공 받을 수 있다.

- Records Tab으로 이동
  - ■ 필요 시 화면을 아래로 당기면 새로고침
  - ■ 새로고침 혹은 처음 로드 시 Database에 저장되어 있는 운행 기록을 스마트폰에 저장
- 운행 기록을 시간 역순으로 출력
- 운행 정보를 제공 받고자 하는 운행 기록 클릭

- 운행 정보에 해당하는 트래커 화면 생성
- 왼쪽 상단의 ≡ 를 클릭하여 나타나는 프로필 탭 선택
- 프로필 상에서 총 일어난 위험 운전 행동의 수 확인

1 구현 결과

1.1 구현 환경: 서버 환경

- Google Cloud SaaS platform
- Database : Firebase
- Login Module : Facebook(https://developers.facebook.com/)

1.2 구현 환경: 클라이언트 환경

- Text editor : VS code
- Device : Android, iOS Smartphone
- Language : JavaScript, CSS, HTML
- Vehicle : K5

1.3 테스트 시나리오

1) 앱을 통해 로그인 할 시 회원 DB 로부터 데이터를 받아와 사용자 입력과 비교한다.
2) 운전 시작 버튼을 누를 경우 스마트폰의 가속도센서와 자이로센서의 값을 읽어와 센싱된 데이터를 통해 Calibration 이 이루어진다.
3) 운전자의 위치, 회전각, 속도 등의 데이터를 운행 중 실시간으로 스마트폰 센서를 이용해 얻는다.
   - 얻어낸 데이터를 사용하여 위험 운전 행동 여부를 판단하고 했을 경우 위험 운전 행동 기록을 서버 내 회원 DB 와 도로 DB 로 전송한다.
   - 운행 중 GPS 를 통해 측정한 data 를 기반으로 회원 DB 에 저장된 해당 도로의 위험 상황에 대한 Data 를 전송받는다.
   - 운행 중 GPS 를 통해 측정한 data 를 기반으로 도로 DB 에 저장된 해당 도로의 위험 상황에 대한 Data 를 전송받는다.
4) 서버 내의 회원 DB 는 각 회원들의 위험 운전 행동 패턴을 도로에 따라 분석한다.
5) 서버 내의 도로 DB 는 모든 회원들의 위험 운전 행동 패턴을 도로에 따라 분석한다.
6) 분석 된 회원 DB 데이터 및 도로 DB 데이터는 App 상에서 회원의 요청이 있을 경우 전송한다.

1.4 매뉴얼 / 예제

1.4.1 로그인 / 회원가입

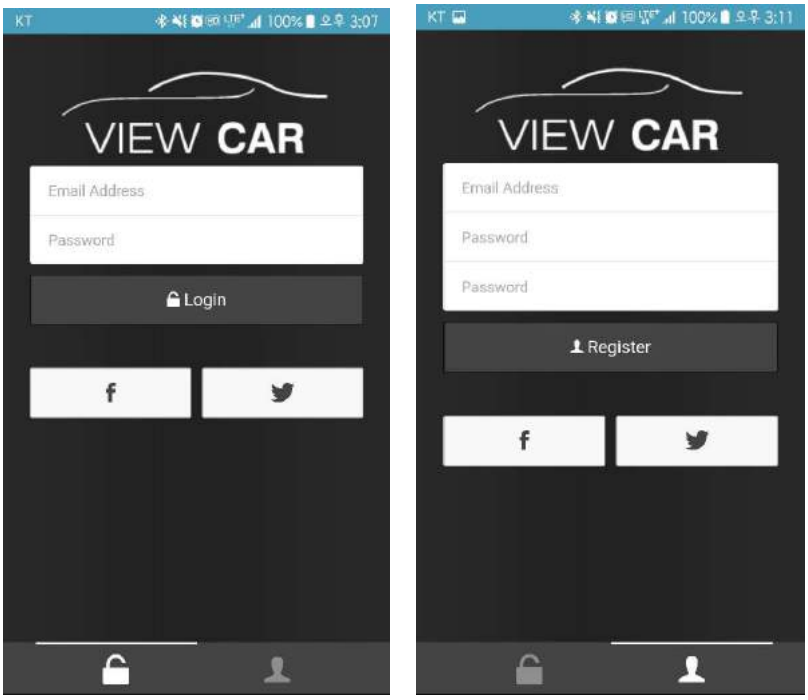


그림 35 로그인 / 회원가입 UI

1.4.2 운행 시작

그림 36 운행 시작 UI

1.4.3 운행 중

그림 37 운행 중 UI

1.4.4 운행 종료 후

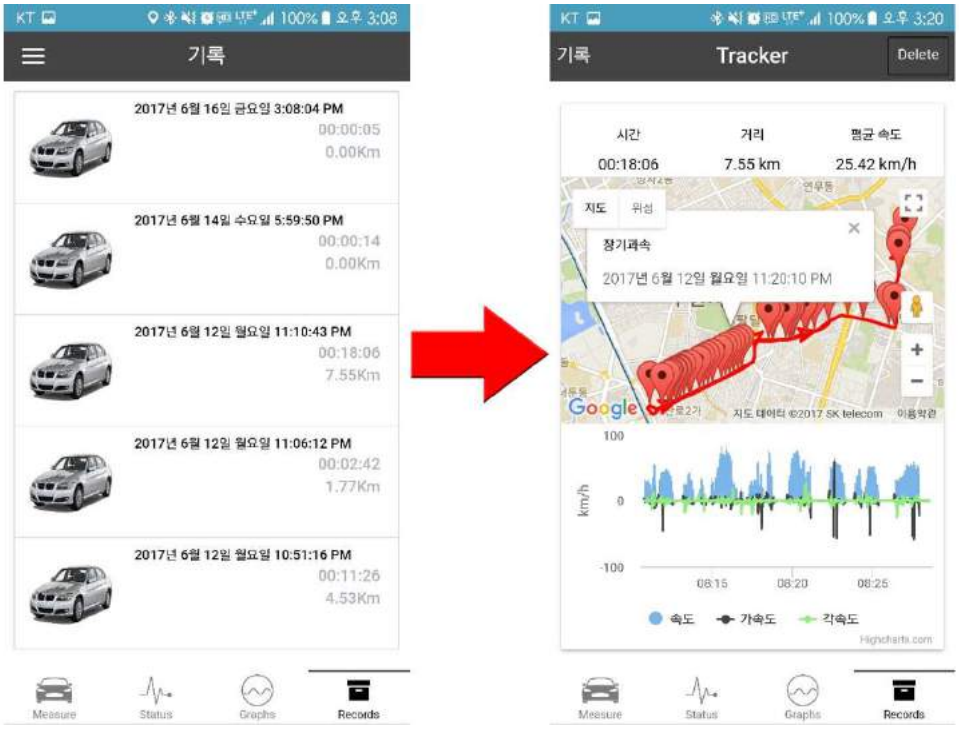


그림 38 운행 종료 후

2 과제 관리

2.1 Test Plan

2.1.1 Ionic View / Smart Phone / Chrome 개발자 도구

UI 확인 및 디자인 확인에는 Ionic View를 활용하였고 여러 센서 및 알고리즘 테스트를 위해서는 스마트폰을 노트북에 USB로 연결하여 빌드 및 실행을 통해 테스트 하였다.

여러 API를 적용하는 과정에서 디버깅이 필요한 경우 혹은 UI UX를 적용하는 과정에서는 Chrome에 있는 개발자 도구를 활용하여 문제점 및 console창을 확인해 테스트를 진행하였다.

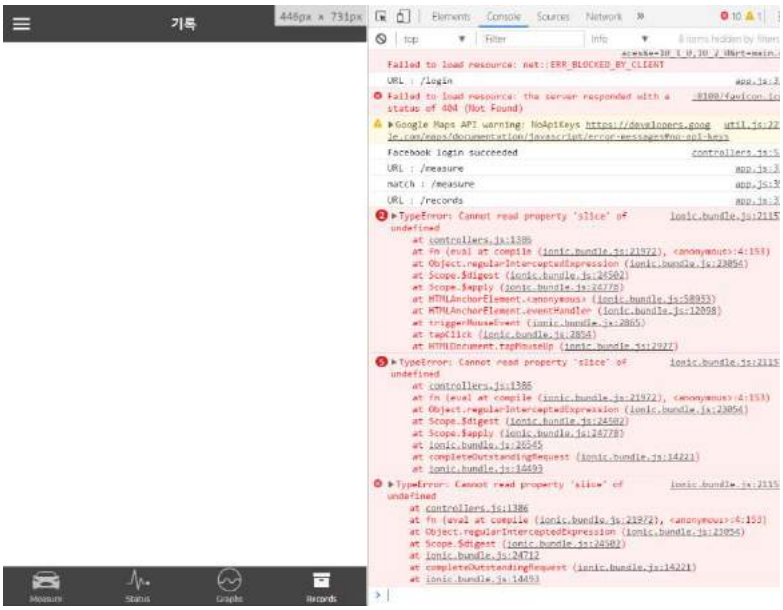


그림 43 개발자도구

2.1.2 RC Car Test

그림 44 RC카 테스트 영상

알고리즘 Test를 위하여 렌트가 필요한 실제 차량을 빌릴 수 없어 우선적으로 RC car를 이용하여 Test를 진행하였다. RC car의 경우 이동 반경이 좁고 순간적인 속도가 빨라 GPS를 적용하기에 무리가 있었고 차량 기준으로 작성된 위험행동 판정 기준으로 테스트하기 어려웠다. 그래서 가속도 센서 및 자이로센서를 이용하여 속도 및 가속도를 측정하였고 판정 기준을 RC car에 맞게 수정하여 테스트를 진행하였다.

2.1.3 실제 차량 테스트

RC car 를 활용하여 알고리즘 Test를 진행한 후에 실제 차량으로 테스트하기 위하여 위험행동 판정 기준을 원래의 기준으로 되돌리고 속도 및 가속도를 GPS 기반으로 측정하였다.

테스트를 위해 학교 앞에 있는 쏘카 서비스나 렌터카를 이용하였고, 주로 차량이 없는 새벽 시간을 이용하여 테스트를 진행하였다.

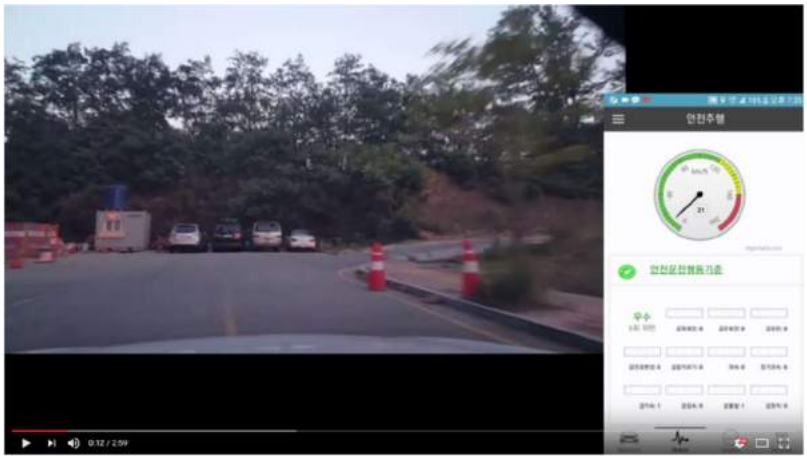


그림 45 실제 차량 테스트 영상

2.2 Code Quality 관리 방안

2.2.1 Coding convention

팀원들이 사용해온 언어 및 코드 작성 스타일이 달라 프로젝트를 진행함에 있어서 어려움이 있을 것으로 예상되어 Coding convention을 정하고 프로젝트를 진행하였다.

NHN, Naver의 Coding convention 에 따라 진행하였다.

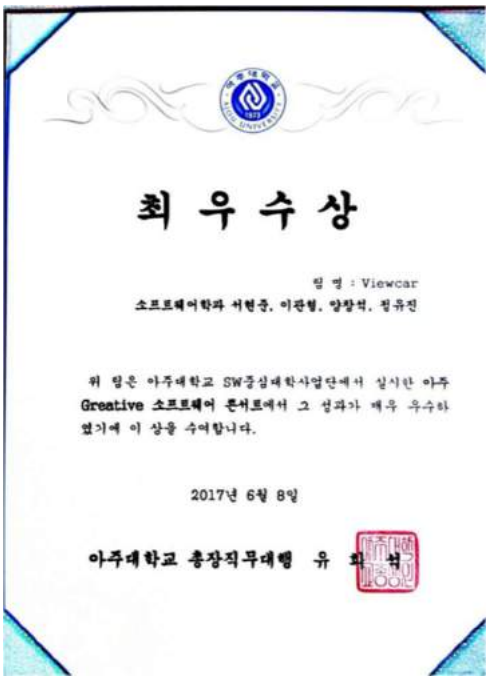
2.2.2 Peer Review

서로의 진행 상황 공유 없이 프로젝트를 진행할 경우 일정관리에 문제가 있다고 판단하여 주말이나 실습 시간에 모여 진행 상황을 공유하고 각자의 task에 대해 피드백 하는 시간을 가졌다. 또한 틈틈이 Trello / 카카오톡에 현재 진행 상황을 공유하면서 프로젝트를 진행하였다.

3 성과

3.1 아주 Greative 소프트웨어 페스티벌

정보통신대학 소프트웨어학과에서 6월 8일 팔달관에서 2017 '아주 Greative 소프트웨어 콘서트'를 개최했다. 소프트웨어 콘서트는 소프트웨어학과에서 매 학기 개최하는 교과목 프로젝트 발표회로, 이를 통해 학생들의 학습 참여에 대한 동기 부여 및 SW 가치 확산을 추구하는 행사이다. 금번 행사에는 2017-1학기 소프트웨어학과 교과목 프로젝트 결과물을 비롯하여 사이버보안학과 BK21+ 사업팀이 참여하였고, SW중심대학지원사업의 가치확산 사업의 일종인 초등학교 방문형 SW 교육 참여팀까지 다양한 팀들이 참가하여 총 99팀이 전시에 참가했다. 그 중 개발 시연 부문(팀)에서 최우수상을 받음으로써 프로젝트의 성과를 높였다.



최 우 수 상

팀 명 : Viewcar

소프트웨어학과 서현준, 이관형, 양창석, 정유진

위 팀은 아주대학교 SW중심대학사업단에서 실시한 아주 Greative 소프트웨어 콘서트에서 그 성과가 매우 우수하였기에 이 상을 수여합니다.

2017년 6월 8일

아주대학교 총장직무대행 유 [illegible]

그림 51 소프트웨어 콘서트 최우수상

그림 52 소프트웨어 콘서트 수상

4 참고 문헌

1) Sebastian O.H. Madgwick. An efficient orientation filter for inertial and inertial/magnetic sensor arrays. April 30, 2010

2) Sebastian O.H. Madgwick, Andrew J.L. Harrison, Ravi Vaidyanathan. Estimation of IMU and MARG orientation using a gradient descent algorithm. June 29 – July 1, 2011.

3) Johannes Paefgen, Flavius Kehr, Yudan Zhai, Florian Michahelles. Driving Behavior Analysis with Smartphones: Insights from a Controlled Field Study.

4) Li Wang, Zheng Zhang and Ping Sun. Quaternion-based Kalman Filter for AHRS Using an Adaptive-step Gradient Descent Algorithm. March 12, 2015.

5) Ali Baharev, PhD. Gyroscope calibration. September 2-3, 2010.

6) Jack B. Kuipers. Quaternions and Rotation Sequences. September 1-10, 1999.

7) Sara Stancin and Saso Tmazic. Time- and Computation-Efficient Calibration of MEMS 3D Accelerometers and Gyroscopes. August 13, 2014.

8) Roberto G. Valenti, Ivan Dryanovski and Jizhong Xiao. Keeping a Good Attitude: A Quaternion-Based Orientation Filter for IMUs and MARGs. August 6, 2015.

9) David Tedaldi, Alberto Pretto and Emanuele Menegatti. A Robust and Easy to Implement Method for IMU Calibration without External Equipments

10) Mark Pedley. High-Precision Calibration of a Three-Axis Accelerometer

11) NTREX. G.INS NT-ARSv2 User's Manual.

12) KITECH 양광웅. Angular Velocity

13) KITECH 양광웅. Euler Angles and Rotation Matrix.

14) KITECH 양광웅. Quaternion.

15) KITECH 양광웅. 1 차 상보 필터와 쿼터니언으로 ARS 설계

16) KITECH 양광웅. Kalman Filter 와 쿼터니언으로 ARS 설계

17) KITECH 양광웅. Extended Kalman Filter로 ARS 설계

5 해당 프로젝트 GitHub 링크

https://github.com/sphilee/eTAS_GPS