Capstone Design 보고서

작품명 : 안드로이드 원격 컨트롤러

과 목 명 : Capstone Design
작 품 명 : 안드로이드 원격 컨트롤러
성 명 : 김 현 정

1. 개요

1-1. 개발 목적

스마트 기기의 대중화와 더불어 블루투스, 와이파이, GPS 송수신과 같은 무선 통신이 활발해진 가운데 스마트 기기 사이의 통신뿐만 아니라 스마트 기기 자체를 컨트롤러로 사용할 수 있도록 하는 움직임이 활발해지고 있다. 이에 따라 블루투스 모듈을 장착한 임베디드 및 하드웨어 시스템을 GPS 기능 및 블루투스로 제어할 수 있는 컨트롤러를 스마트폰으로 제작하였다. 세부적으로는 스마트폰의 GPS 기능을 이용하여 통신하는 기기의 현재 위치를 확인할 수 있고 기기와 스마트 기기 사이의 통신 또한 가능하므로 그 응용 범위가 광범위하다고 할 수 있다.

1-2. 기대 효과

스마트 기기를 이용하여 임베디드 및 하드웨어 시스템을 제어할 수 있으므로 컨트롤러를 따로 제작할 필요가 없어 비용절감을 기대할 수 있으며, 임베디드 및 하드웨어 각자의 컨트롤러를 따로 휴대할 필요가 없어 편리성을 제공한다.

2. 시스템

2-1. 프로그램 흐름도

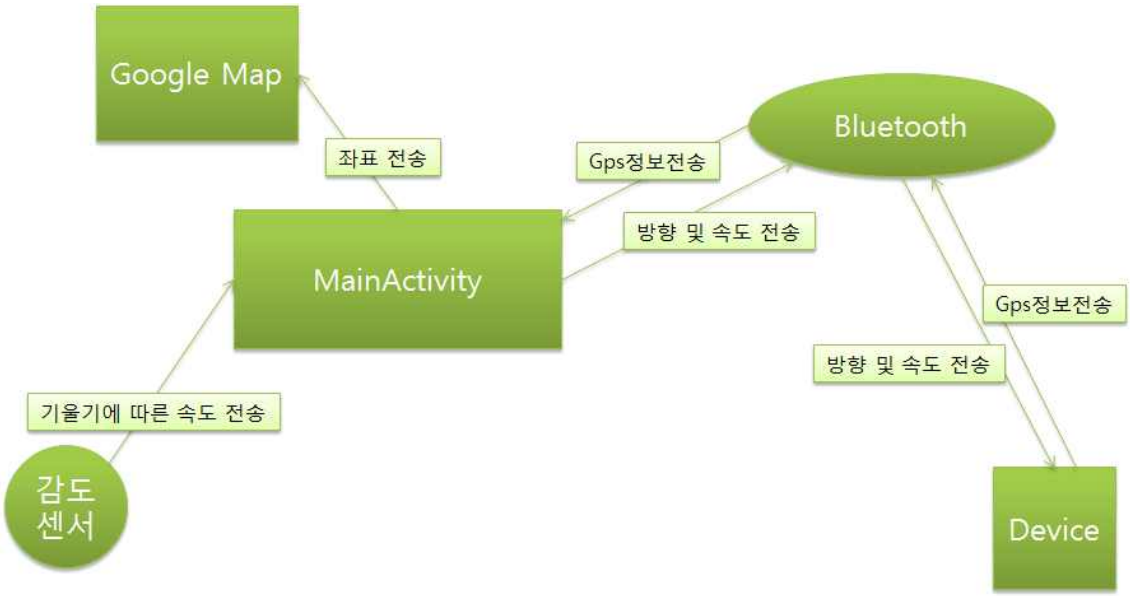


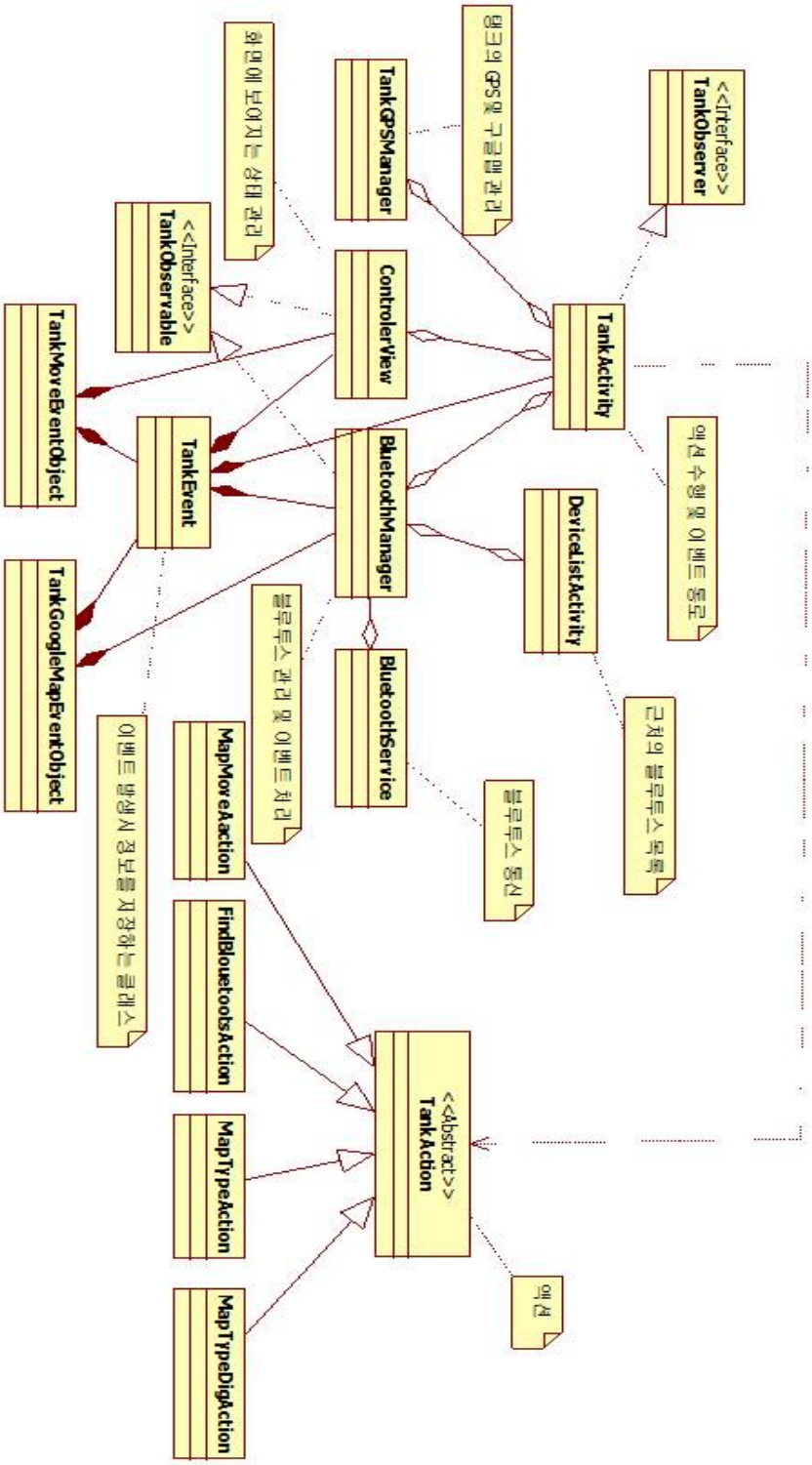
그림1. 전체 흐름도

1) 프로그램 흐름도에 대한 설명

프로그램 흐름도는 프로그램의 전체적인 흐름을 설명하기 위한 다이어그램이므로 여기에서는 간략하게 프로그램의 전체 흐름을 소개한다.

- Main Activity는 감도, 구글맵, 블루투스 모두를 관리한다.
- 감도 센서 즉 자이로 센서는 기울기 및 속도가 감지되면 Main Activity에게 알려준다.
- Google Map은 Main Activity에서 좌표값을 받아 화면에 그려주고 현재위치를 찍는다.
- Bluetooth는 Main Activity와 Device중간에서 통신을 할 수 있게 하는 주요 기능이다.

2-1-1. Class Diagram



다이어그램1. 클래스 다이어그램

2-1-2. Class Diagram 설명

본 다이어그램은 실제 작품의 소스를 클래스 다이어그램으로 표현한 것이며, 각각의 화살표는 액티비티 서로 간의 연관성을 표현한 것이다. 클래스명의 'Tank'라는 표현은 컨트롤러의 타겟이 될 디바이스의 상징적 표현이므로 참고한다. 상세 내용은 다음과 같다.

Class명	설명
TankActivity	MVC패턴의 Controller의 역할을 한다. 각 기능을 하는 클래스들에게 블루투스 통신, 구글맵 이동, 기기의 현재 위치 표시 등의 명령을 내리면 클래스들의 변경이 이뤄지게 되는데 그를 감지해서 그에 따른 블루투스 통신, 맵 마커, 디바이스 연결 등의 액션을 취하는 기능을 하도록 하는 클래스이며 모든 기능을 여기에서 컨트롤하도록 만들었다.
TankGPSManager	디바이스의 현재 위치가 변경될 때마다 TankActivity에게 알리는 역할을 하며, TankActivity가 이 요청을 받으면 구글맵에 디바이스의 현재 위치를 찍어주는 기능을 하는 매니저를 구현하였다.
ControlerView	User Interface 및 각종 화면 구성을 여기에서 구현하였다.
BluetoothManager	디바이스에서 컨트롤러에게 데이터 송수신을 하거나, 컨트롤러가 디바이스에게 데이터를 즉 가속도, 각도 등의 데이터를 송수신할 때 통신에 필요한 것이 여기에 구현되어 있다.
BluetoothService	실제로 가속도, 각도, GPS 정보 등의 데이터가 블루투스 서비스를 통하여 오가게 된다.
TankObserver, TankObserbale	클래스끼리 통신을 하는데 그 방식을 옵저버 패턴으로 구현하였다. Obserbaled에 구현된 클래스는 Observer가 모두 감지하고 있으며 그에 따라 일 처리를 하게하는 패턴으로 구현하였다.
TankEvent	클래스끼리 통신할 때 쓰이는 메시지 랩퍼를 구현.
TankAction	액션의 단위 해당 클래스를 상속받아 액션을 하나하나 구현.

2-2. 작품에 적용된 블루투스 기능

2-2-1. Android Bluetooth Architecture 소개

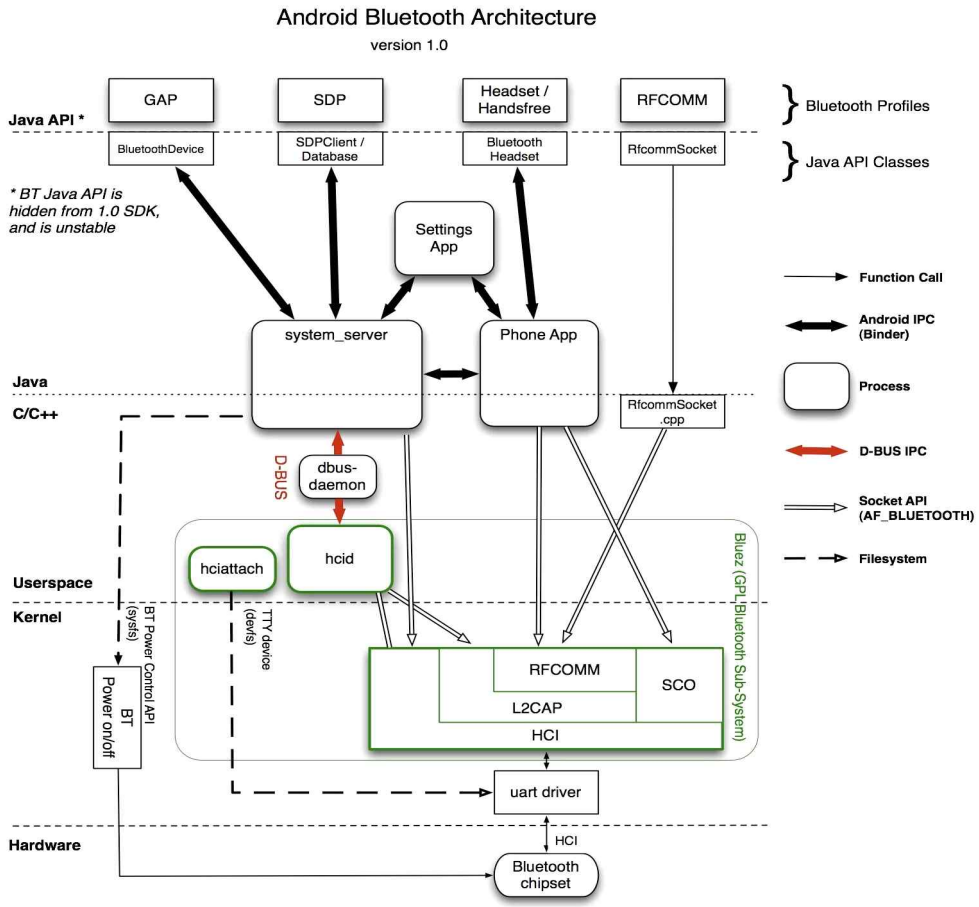


그림2. Android Bluetooth Architecture

안드로이드는 블루투스 프로토콜 스택을 포함하고 있어 블루투스 디바이스들과 무선통신이 가능하다. 블루투스 프레임웍을 사용하여 블루투스에 엑세스할 수 있으며 다음과 같은 작업을 수행한다.

- **주변 블루투스 디바이스 검색**
- **페어링 된 블루투스 디바이스를 위한 블루투스 어탭터 쿼리**
- **RFCOMM 채널 설정**
- **SDP(Service Disovery Protocol)을 통한 다른 디바이스와의 커넥션**
- **양방향 데이터 전송**
- **복수 커넥션 관리**

실제 하드웨어를 컨트롤하는 부분은 보이지 않으며 함수 Call에 따라 내부 설정 등을 변경해 놓는다. 하드웨어 컨트롤 관련된 부분은 2.01에서는 별도로 추가된 것으로 보인다.

2-2-2. Bluetooth와 관련된 라이브러리

BT Power on/off 를 담당하는 소스 경로	Make파일 공유 라이브러리 libbluedroid.so를 JAVA에서 사용 가능 하도록 하는 라이브러리	BlueTooth와 관련된 JNI 관련 파일
-system/bluetooth/bluedroid/bluetooth.c	-frameworks/base/core/jni/Android.mk	-android_server_BlueToothA2dpService.cpp -android_server_BlueToothDeviceService.cpp -android_server_BlueToothEventLoop.cpp

2-2-3. 블루투스 통신 주요 클래스 요약

BluetoothAdapter	로컬 블루투스 어탭터 하드웨어를 나타낸다. 다른 블루투스 디바이스 찾기, 페어링 된 디바이스 퀘리, 다른 디바이스에서 부터의 통신 요구를 기다리기 위한 BluetoothServerSocket 만들기를 할 수 있다.
BluetoothDevice	상대방의 블루투스 디바이스를 나타낸다. BluetoothSocket을 통해 상대방 디바이스와 커넥션을 요구하거나 이름, 주소, 클래스, 페어링 상태등의 정보를 퀘리할 수 있다.
BluetoothSocket	블루투스 소켓을 위한 인터페이스를 나타낸다. 어플리케이션 InputStream과 OutputStream을 사용해서 다른 블루투스 디바이스와 데이터 교환을 할 수 있는 연결 포인트이다.
BluetoothServerSocket	두대의 디바이스를 연결하기 위해 한쪽의 디바이스는 이 클래스를 사용해서 서버소켓을 오픈해야만 한다.
BluetoothClass	블루투스 디바이스의 일반적 특성과 기능을 나타낸다. 이 클래스는 디바이스의 디바이스 클래스와 서비스를 정의하는 읽기 전용 속성의 집합이다.

2-2-4. 작품에 적용된 안드로이드 블루투스 커넥션 과정에 대하여

2-2-4-1. 블루투스 Permission 선언

우선 블루투스를 사용하기 위하여 BLUETOOTH, BLUETOOTH_ADMIN 둘 중 하나의 퍼미션을 선언하였다. 각 퍼미션의 기능은 다음과 같다.

1) BLUETOOTH - 커넥션 요구, 커넥션 accept, 데이터 전송 등의 기능
2) BLUETOOTH_ADMIN - 디바이스 discovery를 시작 또는 블루투스 설정을 조작

주요 소스 1 :

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

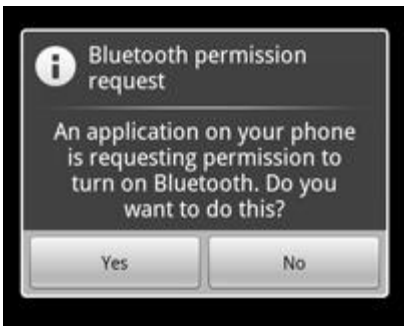
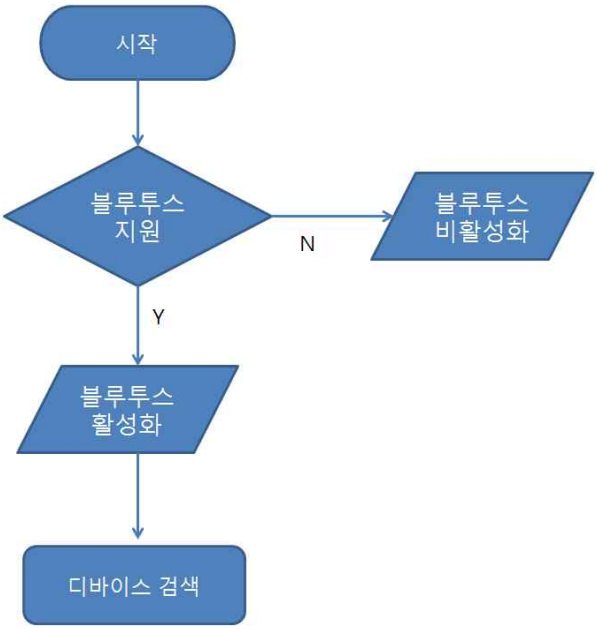


그림3. 블루투스 퍼미션 대화상자 확인

2-2-4-2. 블루투스 셋업 과정

1) 블루투스 셋업



다이어그램2. 블루투스 지원 확인

블루투스 통신을 하기에 앞서 디바이스가 블루투스를 지원하는지에 대하여 확인하는 과정이 필요하다. 블루투스를 지원하지 않는 경우 블루투스를 비활성화 해야한다. 블루투스를 지원하지만 활성화 되어있지 않은 경우 사용자가 어플리케이션을 떠나지 않고 블루투스를 활성화하도록 요구할 수 있다. 이 작업은 블루투스 어탭터를 사용하여 두 단계로 나눈다.

1단계. BluetoothAdapter 얻기

블루투스 액티비티에서 BluetoothAdapter가 요구되고 getDefaultAdapter()를 호출한다.
그러면 디바이스의 블루투스 어탭터를 나타내는 BluetoothAdapter 인스턴스를 리턴한다.

주요 소스 2 :

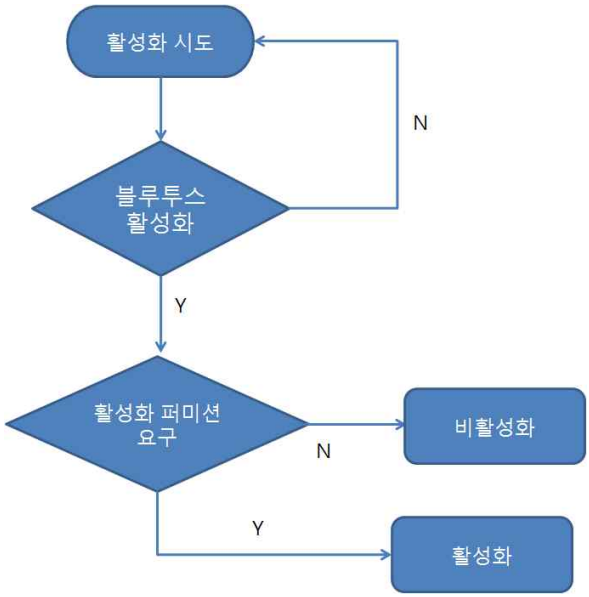
```
BluetoothAdapter mBTAdapter = BluetoothAdapter.getDefaultAdapter();
        if (mBTAdapter == null) {
        // device does not support Bluetooth
        }
```

2단계. 블루투스 활성화

블루투스가 활성화 되어있는지 확인하는 메소드(isEnabled())를 호출하고 메소드가 리턴하는 값(boolean)에 따라서 활성화 유무를 알 수 있다. 블루투스를 활성화 시키기 위하여 ACTION_REQUEST_ENABLE 인텐트로 startActivityForResult()를 호출하였다.

주요 소스 3 :

```
If (!mBTAdapter.isEnabled()) {
   Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
   startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
}
```

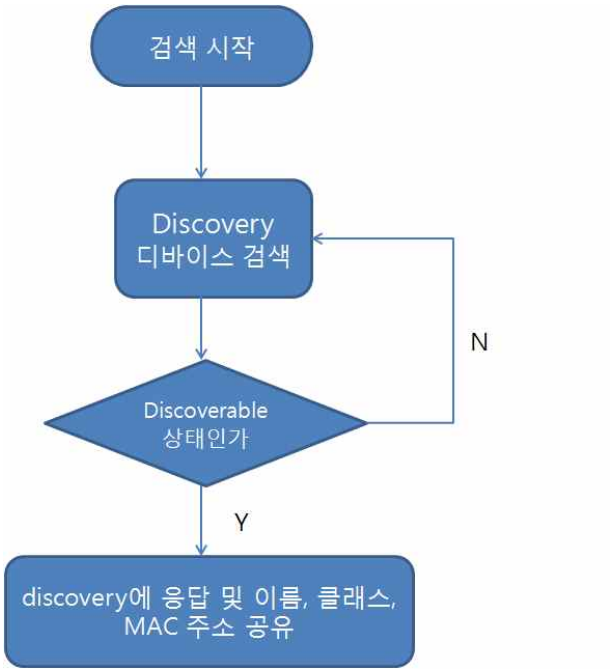


다이어그램3. 블루투스 활성화

그림2와 같이 블루투스 활성화 퍼미션을 요구하는 대화창에서 Yes를 선택하면 블루투스는 활성화 되고 어플리케이션으로 돌아온다.

2) 디바이스 검색

블루투스 어탭터를 사용하면 디바이스 discovery 또는 페어링 된 원격 블루투스 디바이스 목록을 찾을 수 있다. 디바이스 discovery는 주변의 활성화된 블루투스 목록을 찾고 정보를 요구하며 통신가능 범위에 들어있는 블루투스 기기라 하더라도 현재 discoverable 상태 즉 활성화 상태인 경우에만 응답한다. 이에 대한 다이어그램은 다음과 같다.

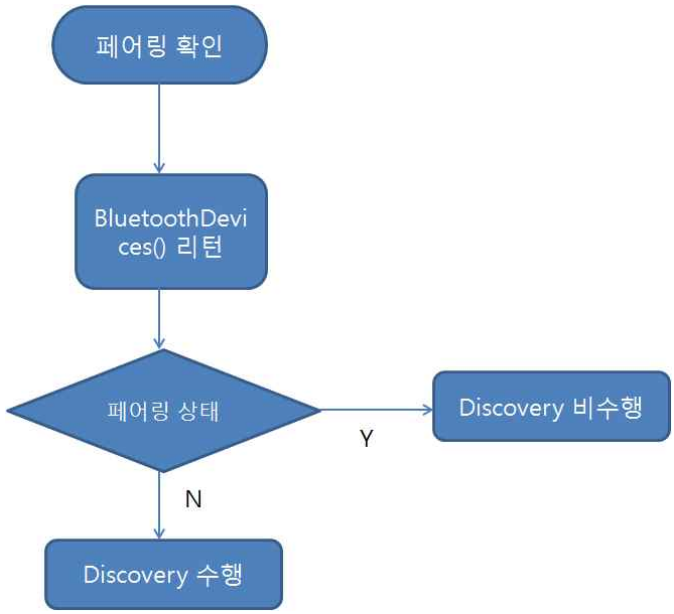


다이어그램4. Discoverable상태의 디바이스 또는 페어링된 디바이스 검색

디바이스와 연결에 성공하면 페어링을 할 것인가 물어본다. 디바이스 페어링이 이루어지면 상대 디바이스에 대한 기본정보가 저장되고 이미 알고있는 원격 디바이스의 경우 MAC주소를 사용하게 되면 바로 커넥션 과정으로 들어간다.

3) 페어링된 디바이스를 쿼리

“그렇다면 이미 페어링된 디바이스에 대해서는?”
디바이스가 discovery를 하기 앞서 원하는 디바이스가 이미 페어링 되어있는지 확인해 볼 필요가 있다. getBondedDevices()함수를 호출하면 이 작업이 가능하다. 이 함수는 페어링 된 디바이스들의 집합이 있는 함수를 리턴하게 해준다. 이에 대한 다이어그램은 다음과 같다.



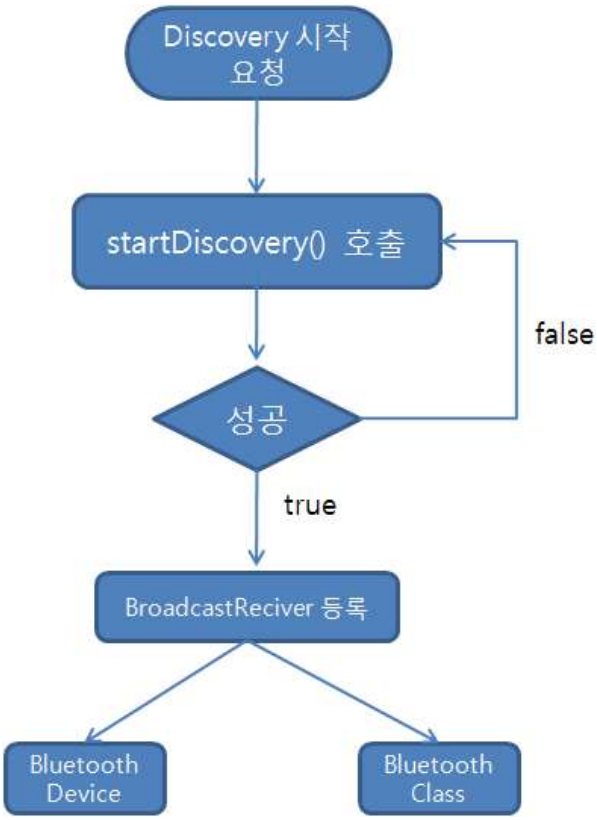
다이어그램5. 페어링 상태의 디바이스 확인

주요 소스 4 :

```
Set<BluetoothDevice> pairedDevices = mBluetoothAdapter.getBondedDevices();
if (pairedDevices.size() <> 0) {
    for (BluetoothDevice device : pairedDevices) {
        mArrayAdapter.add(device.getName() + “₩n” + device.getAddress());
    }
}
```

4) 디바이스 Discovery

디바이스 discovery를 시작하려면 startDiscovery()를 호출하면 된다. 이 과정은 비동기식이라 메소드를 호출하면 discovery가 성공적으로 시작되었나 결과를 알려주는 boolean값을 곧바로 돌려준다. Discovery과정은 보통 12초간의 inquiry scan후 발견된 각 디바이스에 대해 이름을 가져오기 위한 page scan으로 이루어진다. 어플리케이션은 각 발견된 디바이스에 대한 정보를 받기 위해 ACTION_FOUND 인텐트를 위한 BroadcastReceiver를 등록해야만 한다. 각 디바이스마다 시스템이 ACTION_FOUND 인텐트를 브로드캐스트 한다. 이에대한 다이어그램은 다음과 같다.



다이어그램6. 디바이스 Discovery 과정

주요 소스 5 :

```
 Final BroadcastReceiver mReceiver = new BroadcastReceiver() {
 public void onReceive(Context context, Intent intent) {
 String action = intent.getAction();
 if (BluetoothDevice.ACTION_FOUND.equals(action)) {
BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
            mArrayAdapter.add(device.getName() + "₩n" + device.getAddress());
        }
    }
```

```
};
BroadcastReceiverIntentFilter
filter = new IntentFilter(BluetoothDevice.ACTION_FOUND;
registerReceiver(mReceiver, filter);
```

주의 할 점 : 디바이스 discovery를 수행하는건 블루투스 어탭터에게 매우 부담이 큰 작업으로 매우 많은 리소스를 요구한다. 커넥션 할 디바이스를 찾았다면 커넥션을 시작하려고 시도하기 전에 cancelDiscovery()를 호출해서 discovery를 멈춰야 한다. 또한 이미 다른 디바이스와 커넥션 되어 있으면 discovery과정동안에 대역폭이 떨어질 수도 있기 때문에 커넥션 된 상태에서는 discovery를 하지 않아야 한다.

5) Discoverable 활성화

다른 디바이스가 내 디바이스를 검색 가능하게 하는 기능은 startActivityForResult(Intent, int)에 ACTION_REQUEST_DISCOVERABLE 액션 인텐트를 넣어 호출한다. 이 메소드를 호출하면 어플리케이션을 멈추지 않고 discoverable 모드를 활성화 하도록 요청한다.
기본적으로는 디바이스는 120초동안 discoverable 모드로 있는다.

주요 소스 6 :

```
Intent discoverableIntent =
new Intent(BluetoothAdpater.ACTION_REQUEST_DISCOVERABLE);
discoverableIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION,
300); startActivity(discoverableIntent);
```

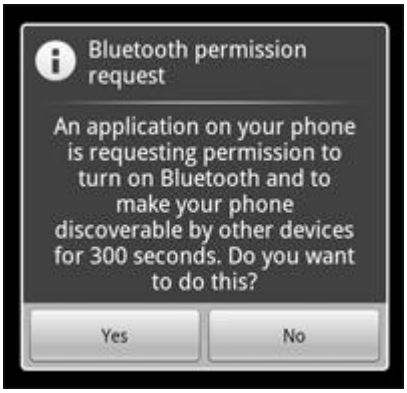


그림4. 디바이스 상태 변경 다이얼로그

그러면 그림4와 같은 다이얼로그가 떠서 디바이스를 discoverable 상태로 만들도록 할 것인지 묻는다. Yes를 선택하면 디바이스는 정해진 시간동안 discoverable상태가 된다.
No를 선택하거나 에러가 발생하면 Activity.RESULT_CANCELLED가 된다. 디바이스는 discoverable 시간동안 아무 반응이 없이 조용히 있는다. 만일 discoverable모드가 변경되는 순간 변경되었는지의 통보를 받고 싶으면 ACTION_SCAN_MODE_CHANGED 인텐트에 대한 BroadcastReceiver를 등록할 수도 있다. 이에 대해선 아래 표에 정리하겠다.

EXTRA_PREVIOUS_SCAN_MODE EXTRA_SCAN_MODE	각각 이전 스캔모드와 변경된 새 스캔모드가 들어있다.
SCAN_MODE_CONNECTABLE_DISCOVERABLE SCAN_MODE_CONNECTABLE SCAN_MODE_NONE	**각 필드에 들어갈 수 있는 값** - discoverable 모드 - discoverable은 아니지만 커넥션을 받아들일 수는 있는 모드 - discoverable도 아니고 커넥션도 받아들일 수 없는 모드

2-2-5. 실제 작품에서 블루투스 설정 부분

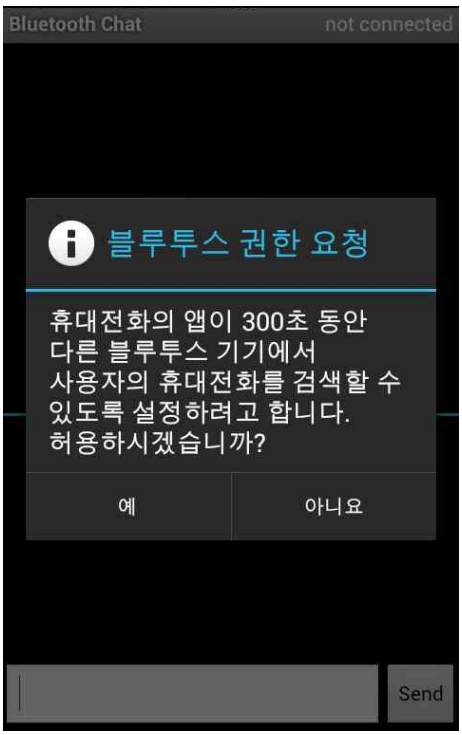


그림5. 실제 작품에서 클라이언트 기기가 될 부분의 블루투스 권한 요청하는 모습

- 실제 작품에서 클라이언트 즉 컨트롤러가 제어하게 될 블루투스 모듈을 장착한 기기가 될 부분이며 현재 본 작품에서는 '안드로이드 모바일 기기'를 이용하고 있음을 참고한다. 화면에서 앱이 300초 동안 다른 블루투스 기기에서 검색할 수 있도록 설정되어 있는데 이 대기 시간은 자유롭게 코드상에서 설정 가능하다.

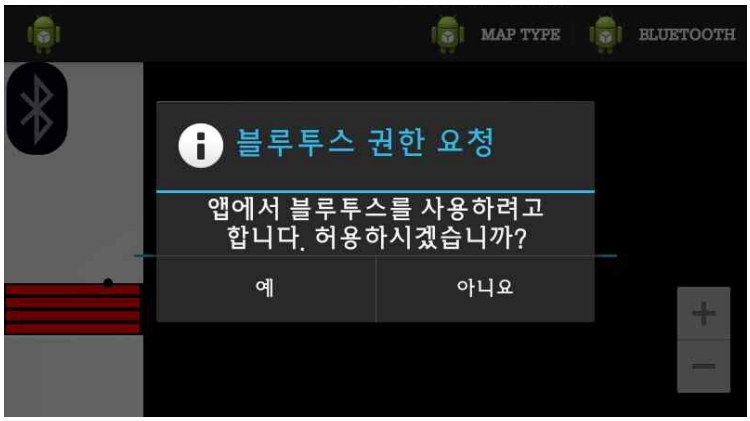


그림6. 실제 작품에서 컨트롤러의 블루투스 권한 요청

- 실제 작품에서 컨트롤러가 될 부분이다. 블루투스 통신을 하기에 앞서 블루투스 권한 요청 다이얼로그를 실행하도록 코딩하였고, 이 과정이 끝나면 블루투스 기기를 수동으로 검색할 수 있는 우측 상단 액션바를 두어 편리한 인터페이스를 제공하도록 노력하였다.

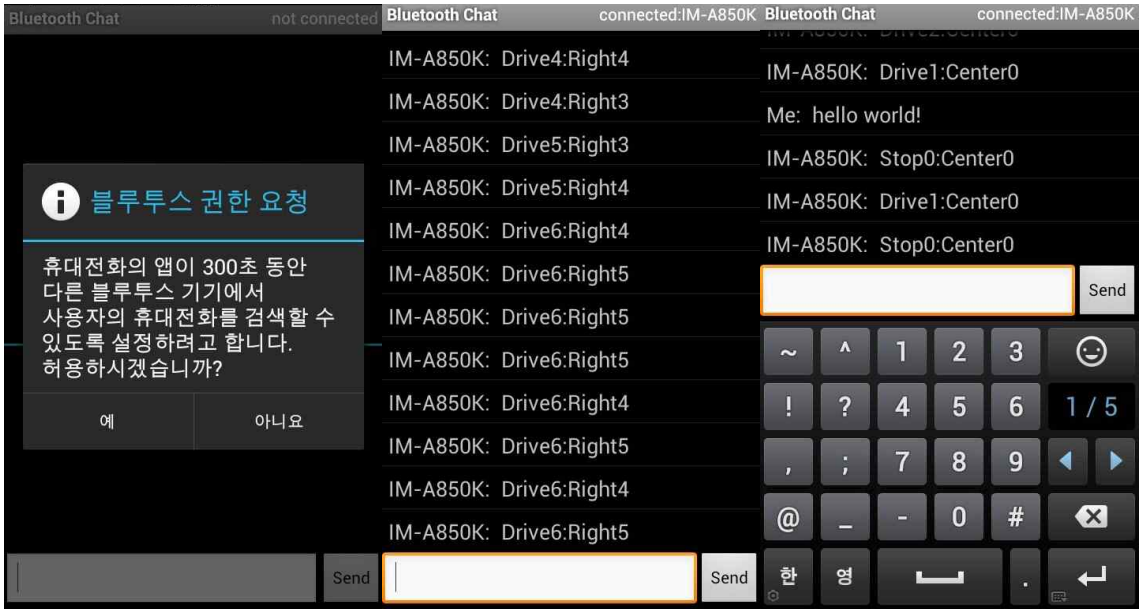


그림7. 블루투스 설정 및 클라이언트 단말 부분 (왼쪽부터 가,나,다 순서)

가. 블루투스 권한 요청이 되는 부분
나. 컨트롤러 측에서 데이터를 보내고 그 데이터 내용을 화면에 뿌려주는 부분
다. 컨트롤러가 클라이언트 기기에 수동적으로 입력한 데이터를 보내는 부분
다-(1). 클라이언트 단말쪽에서 키보드 입력을 할 수 있게 한 이유는 양방향 블루투스 통신이 되는지를 알아보기 위한 실험을 위해서이다.

2-3. 작품에 적용된 안드로이드 각도 및 가속도(자이로) 센서

2-3-1. 각도 제어 부분

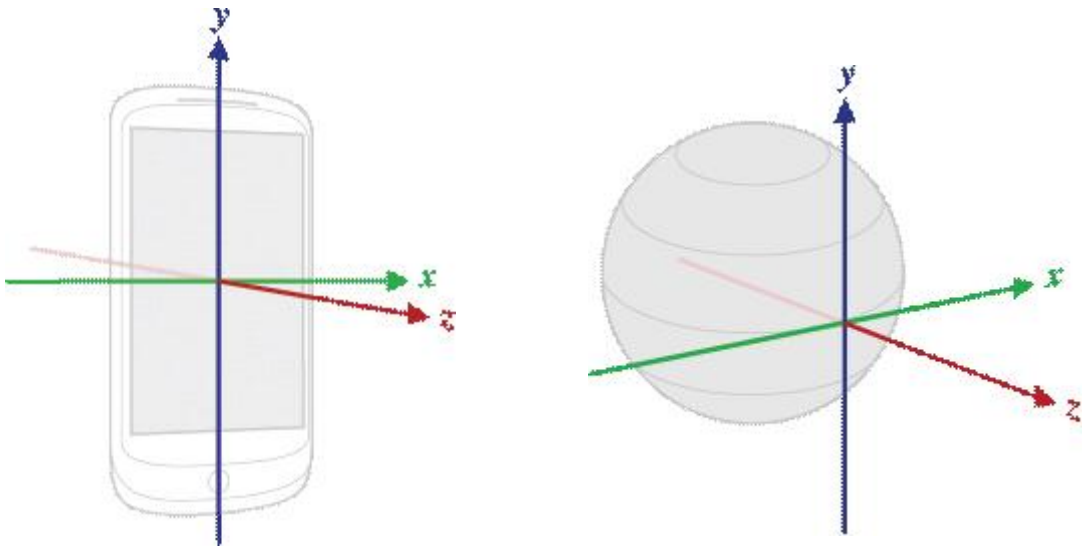


그림8. 단말에 적용된 자이로 센서 및 자이로센서 기본 축

1) SENSOR_ORIENTATION (방향 센서)

각도(자이로) 센서가 측정하는 데이터는 움직임에 대한 값(혹은 기기에 가해지는 중력 값)이므로 가속도를 측정하기 위해 움직인 시간동안의 데이터를 저장한 후 별도로 연산해주었다. 기본적인 가속도를 구하는 방법은 직전 움직임 값에서 현재 움직임 값을 빼는 것이며 각 데이터에 대한 설명은 다음과 같다.

X축(Lateral)	-측면 움직임 측정 -움직임 없을 때의 값 : 0 -왼쪽으로 기울기 : X축 증가(가속도 감소) -오른쪽으로 기울기 :X축 감소(가속도 증가)
Y축(Longitudinal)	-전방 후방 움직임 측정 -움직임 없을 때의 값 : 0 -전방 기울기 : Y축 감소 -후방 기울기 : Y축 증가
Z축(Vertical)	-위 아래 움직임 측정 -수평을 유지 : 9.81(중력 값) -Z축 움직임 측정법 : Z축 데이터 - 9.81 -하늘 쪽 움직임 : Z축 감소 -땅 쪽 움직임 : Z축 증가

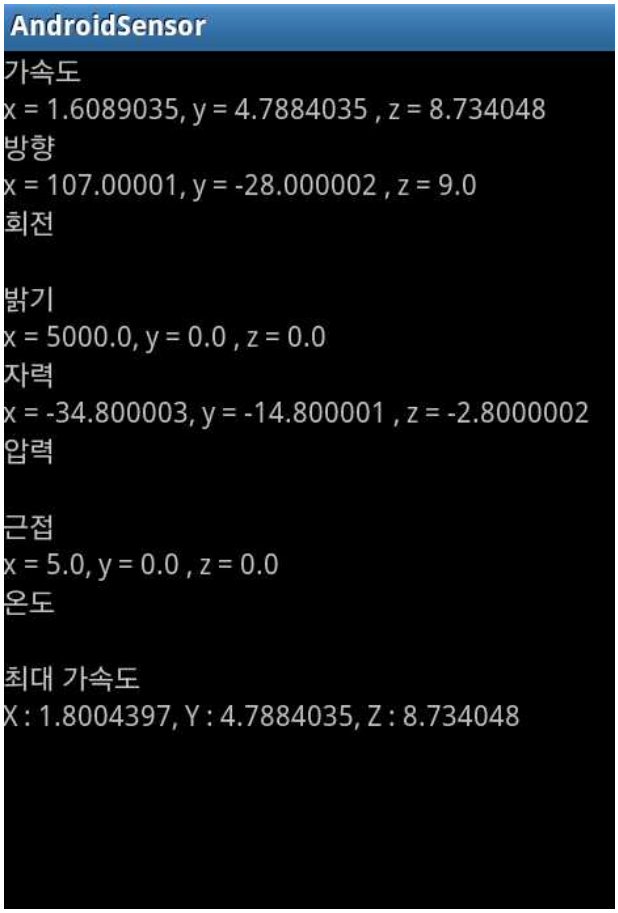


그림9. 자이로센서가 측정할 수 있는 값 샘플 코드 작성 결과 화면

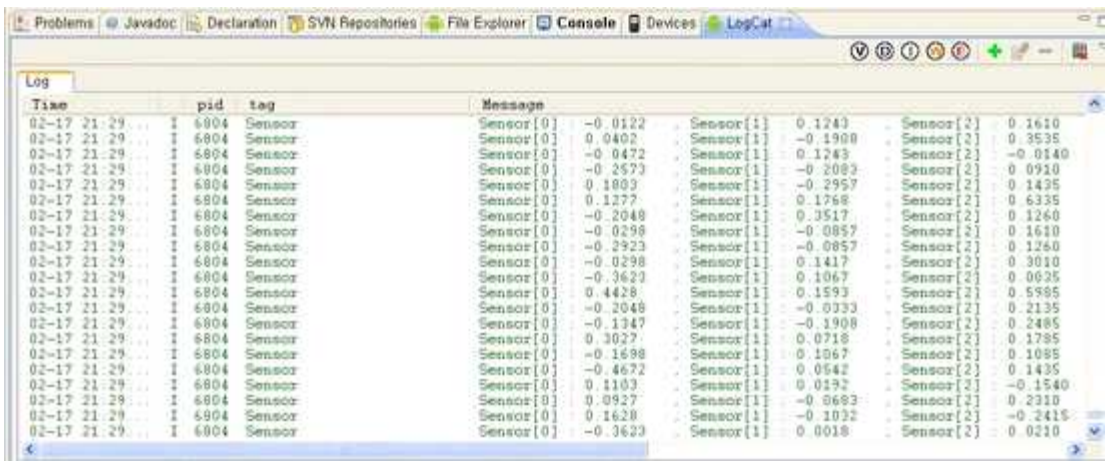


그림10. 각 축이 받아오는 데이터를 보여주는 콘솔창

위와 같이 실제 자이로 센서를 이용하여 작품을 코딩하기 전에 자이로 센서 값을 측정하는 방법 및 실제 코딩을 하여 다음과 같은 샘플 코드를 작성하였다. 자이로 센서는 가속도, 방향, 각도 뿐만 아니라 다양한 안드로이드 센서가 측정할 수 있는 값들을 측정 할 수 있었다.

주요 소스 7 :

```
SensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
(중략)
x_axis.setText(""+ Float.toString(event.values[0]));
y_axis.setText(""+ Float.toString(event.values[1]));
z_axis.setText(""+ Float.toString(event.values[2]));
 break;
(중략)
```

values[0]	**z축을 기준으로 한 회전값** 0 ~ 359의 범위 **z축이 가르키는 방향** 북쪽일 경우 0 동쪽일 경우 90 남쪽일 경우 180 서쪽일 경우 270
values[1]	**x축을 기준으로한 기울기 값** -180 ~ 180의 범위 **x축을 기준으로 구르기 값** 위로 구르면 +, 아래로 구르면 -
values[2]	**y축을 기준으로한 기울기 값** -90 ~ 90의 범위 **y축을 기준으로 구르기 값** 왼쪽으로 구르면 +, 오른쪽으로 구르면 -

▲ 각 values값에 대한 해석

2-3-2. 가속도 제어 부분

2) SENSOR_ACCELOROMETER (가속도 센서)

가속도 부분에서의 각 배열의 값은 (m/s2) 단위로 되어있으며, 접촉힘을 측정한다.
각 values 값의 의미는 다음과 같다.

values[0]	X축에 적용되는 힘
values[1]	Y축에 적용되는 힘
values[2]	Z축에 적용되는 힘

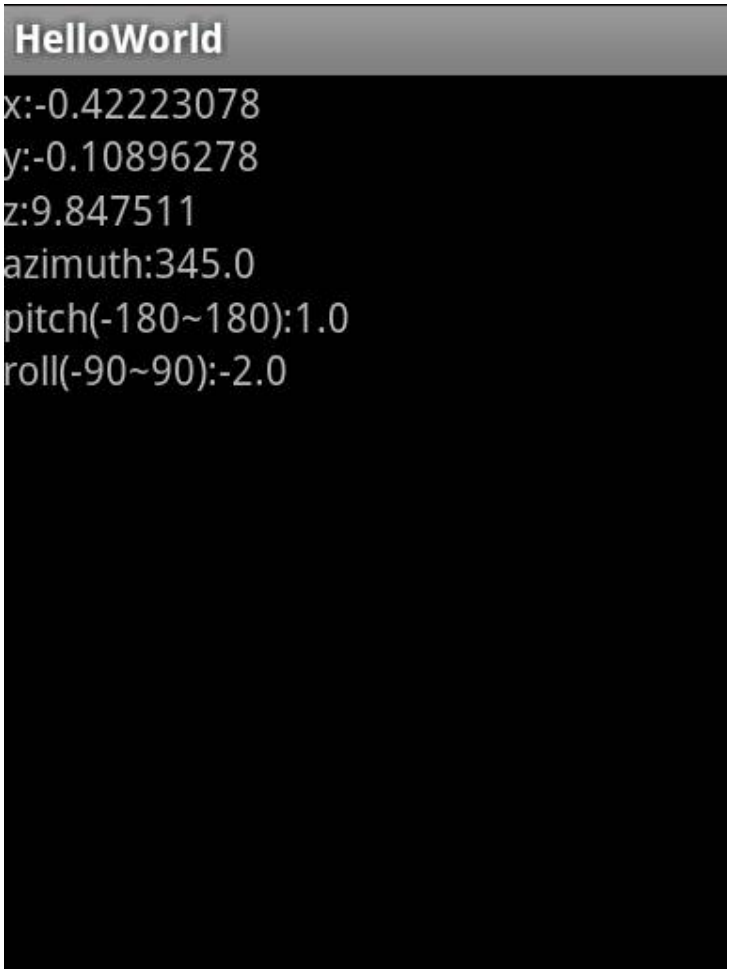


그림11. 가속도 측정에 대한 샘플 코드 작성 결과 화면

단말의 왼쪽 측면을 오른쪽 방향으로 눌렀을때 x 가속도 값은 음수를 나타낸다. 단말이 테이블위에 평평하게 놓여있을때 -STANDARD_GRAVITY 값, 즉 -9.8 (m/s2) 을 가진다. 이 것은 즉 단말이 테이블에서 중력에 대한 반작용으로 적용되는 힘을 나타낸다.

2-3-3. 실제 작품에서 컨트롤러가 각도를 송수신하는 부분

그림12. 컨트롤러 측 각도센서 제어부 (좌측 제어바)

1) 컨트롤러가 단말기의 가속도 및 좌,우,앞,뒤 데이터를 클라이언트 단말기 쪽으로 송신한다. 좌측 제어바가 빨간 색일 경우 전진 가속도를 나타내며, 제어바가 검정색일 경우 후진 가속도를 나타내도록 구현하였고, 가운데 점은 좌, 우 방향을 표시하도록 구현하였다. 인터페이스 구현에 대한 설명은 다음 장을 참고한다.

그림13. 클라이언트 단말 측 수신 데이터 화면에 출력

2) 컨트롤러가 송신한 가속도 및 방향에 대한 정보를 받아와 유저가 확인하기 쉽도록 뿌려주게 구현하였으며, Drive는 가속도의 힘을 나타내며, Right, Left는 방향을 나타내도록 구현하였다.

2-4. 작품에 적용된 GPS 시스템 부분

안드로이드 단말 기기에 내장된 GPS 로부터 위도, 경도 정보를 받아서 화면에 출력하려면? GPS로부터 위치정보를 수신하려면 LocationManager가 필요하고 LocationManager가 수신한 위치정보에 변화가 있는 경우에는 LocationListener 인터페이스의 onLocationChanged()를 이용하여 처리할 작업을 정의할 수 있다.

2-4-1. 맵을 띄우기 위한 셋업 과정

1) 맵뷰와 맵액티비티

지도를 넣기 위해서는 화면 영역을 할당 받을 수 있도록 맵뷰(MapView)를 사용한다. 맵뷰를 사용할 때는 맵 액티비티(MapActivity)를 같이 사용하여야 한다.

2) 개발자 키 발급 받기

맵을 사용할 때는 개발자 키를 발급받아 등록해야 구글 서버에서 지도 데이터를 받을 수 있다. 이 과정을 생략하면 맵 자체를 띄울 수 없다.

3) 라이브러리의 사용

맵을 사용하기 위하여 구글 맵에서 제공하는 라이브러리를 사용하여야 하며, 맵을 단순히 띄우기 위한 목적으로만 사용하였으며 프로그램 안의 세부 코드 및 표현은 직접 코딩하고 수정하였다. 라이브러리를 추가하기 위해서는 android sdk manager의 extras의 google play services항목을 설치해야하며, 이를 사용하기 위한 프로젝트에 import시켜 주면 된다.

4) 매니페스트 수정

지도를 추가하게 되면 크게 두 가지 권한(인터넷 접속 권한, GPS 위치정보 확인 권한)이 필요하다. 이 모든 작업은 통합개발환경(본인은 이클립스 사용)에서 manifest.xml파일에 구글 맵을 사용하기 위해 꼭 추가해 주어야 할 옵션같은 개념이다. 따라서 다음과 같은 퍼미션을 추가하는 작업을 한다. 그리고 구글 맵 라이브러리를 사용하기 위해서는 google에서 key값을 받아서 추가해야 한다.

주요 소스 8 :

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
(구글 라이브러리를 사용할때는 application 태그 안에 아래와같은 태그를 넣어주어야한다.)
<uses-library android:name="com.google.android.maps"/>
<meta-data
        android:name="com.google.android.maps.v2.API_KEY"
        android:value="AIzaSyAWy7cECGgjLL2iX8xGgBhI4MisjGvsCUk"/>
```

5) 레이아웃에 맵뷰 추가하기

안드로이드에서는 View라는 개념이 있는데 이 것은 사용자가 직접 눈으로 볼 수 있는 화면을 나타내는 단위이며, 구글맵도 하나의 뷰에 속하므로 뷰를 보여주려면 레이아웃 파일에 맵뷰 형식의 뷰를 추가해 주어야 하는 작업을 거쳐야만 한다.

2-4-2. 맵 제어 과정

1) 맵 타입 변경

맵 타입을 변경하기 위해서는 맵 객체를 레이아웃에 설정된 id값을 통해서 객체를 얻은 다음, 그 객체에 맵 타입을 변경하는 명령을 해주면 된다.

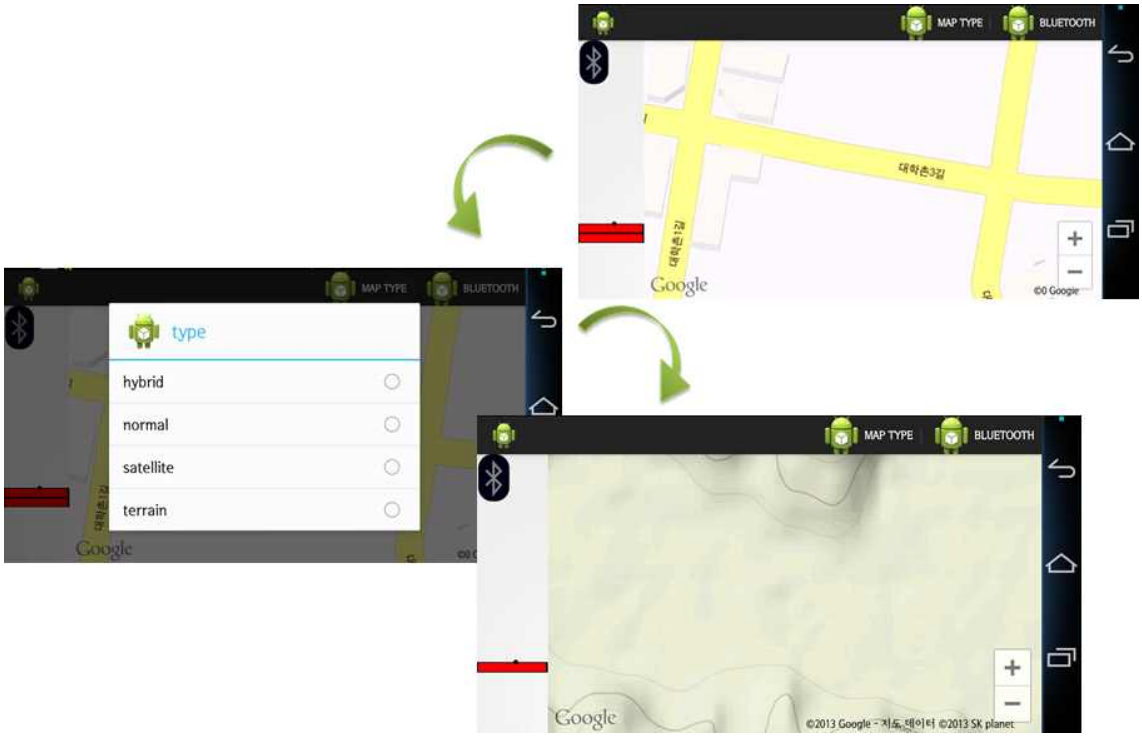


그림14. 맵타입이 변경되는 상황

주요 소스 9 :

```
private GoogleMap googleMap
googleMap = ((MapFragment)
getFragmentManager().findFragmentById(R.id.my_container)).getMap();
googleMap.setMapType(type);
```

2) 맵 위치 변경

맵 위치 변경을 위해 위와 동일하게 맵 객체에 CameraUpdate객체를 생성해서 해당 메소드를 실행하면 그 위치로 이동하게 된다. CameraUpdate객체는 CameraUpdateFactory의 statc메소드를 이용해서 얻을 수 있다. 필요한 값으로는 위치정보와 zoom정보가 있다.

주요 소스 10 :

```
googleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(new
LatLng(obj.latitude, obj.longitude), 19));
```

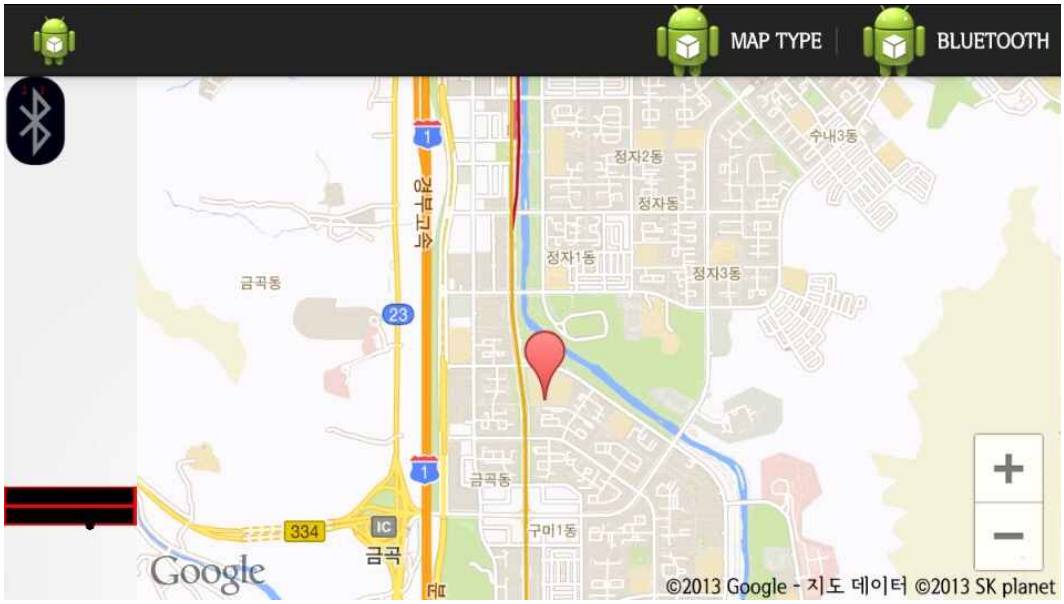


그림15. 맵에 marker가 추가된 화면

3) 구글 맵 위치(marker) 표시

-구글 맵 위치 위치를 표시하기 위해서는 구글 맵 객체에 MarkerOptions객체를 생성해서 해당 메소드를 실행하면 marker가 추가된다. MarkerOptions를 생성하기 위해서는 위치 값이 필요하다.

2-5. UserInterface 구현 방법 소개

1) 속도 및 방향 바 그리기

그림16. 좌측 제어바 및 블루투스 on/off 인터페이스를 그리는 과정

속도와 방향에 대한 인터페이스는 '2-3'절에서 구한 각도를 바탕으로 그림을 그려주는 작업으로 시작한다. 그림을 그리기 위해서는 Android의 View를 상속받아 onDraw메소드를 Overriding하여 그림이 그려지게 한다.

주요 소스 11 :

```
x = (leftOrRight * (controlerViewWidth / 2) / 10) + (controlerViewWidth / 2);
y = zeroSpeedBar
(중략)
for(int i = 0; i > speed i--) {
        speedPaint.setColor(Color.RED);
        Rect r = new Rect()
        r.top = y;
        r.bottom = y - seedOneBlockHeight
        r.left = 0;
        r.right = controlerViewWidth

        linePaint.setColor(Color.BLACK);
        speeBlockDraw(c, r, speedPaint, linePaint, 2);
        y -= seedOneBlockHeight
}
```

2) 블루투스 on/off 그리기

블루투스 이미지 on/off 두 개를 준비하여(저작권 문제를 우려하여 본인이 직접 이미지를 제작하였다.) 블루투스가 연결되거나 연결이 끊어질 때마다 해당 이미지로 교체해준다. 이미지가 교체가 되면 자동으로 화면이 갱신이 되는 것이 아니기 때문에 View의 invalidate메소드를 실행시켜 화면이 갱신되게 한다. 한마디로 on/off가 될 때마다 그림이 다시 그려지게 되는 것이다.

주요 소스 11 :

```
public void blueToothOn(){
        Resources res = getContext().getResources();
        img = BitmapFactory.decodeResource(res, R.drawable.bluetooth_on);
        img = Bitmap.createScaledBitmap(img, 72, 100, true);
        invalidate();
```

```
}

public void blueToothOff(){
        Resources res = getContext().getResources();
        img = BitmapFactory.decodeResource(res, R.drawable.bluetooth_off);
        img = Bitmap.createScaledBitmap(img, 72, 100, true);
        invalidate();
}
```

3) 액션바 설정

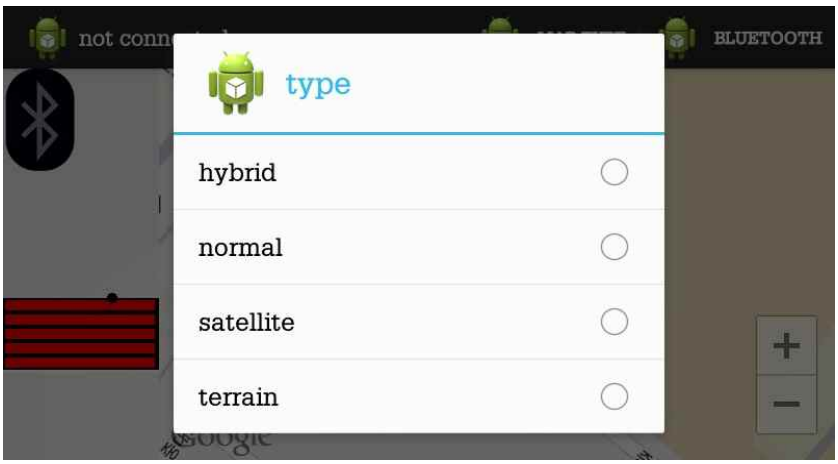


그림17. 상단 액션바 'MapType'을 누르면 나타나는 다이얼로그

그림18. 상단 액션바 'BLUETOOTH'를 누르면 나타나는 다이얼로그

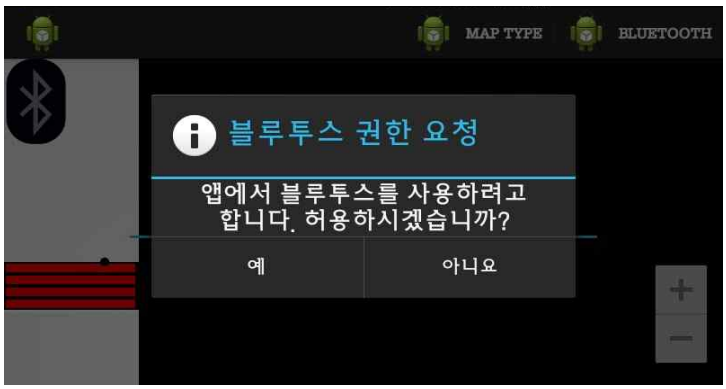
본 작품에서는 옵션이 되는 두가지 메뉴를 액션바에 넣어 인터페이스를 구현하였다. 액션바에 메뉴를 넣기 위해서는 menu.xml파일에 android:showAsAction속성을 설정해주면 된다. 그 후 Activity의 onCreateOptionsMenu메소드를 구현해주면 된다.

주요 소스 12 :

```
public boolean onCreateOptionsMenu(Menu menu) {
getMenuInflater().inflate(R.menu.main, menu);
return true
}
```

3. 부록

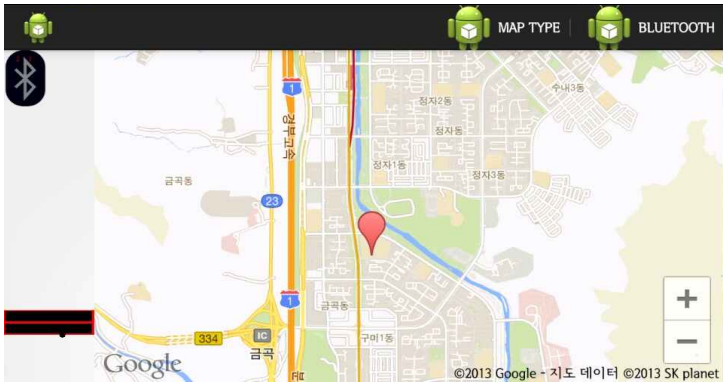
3-1. 작품 시연



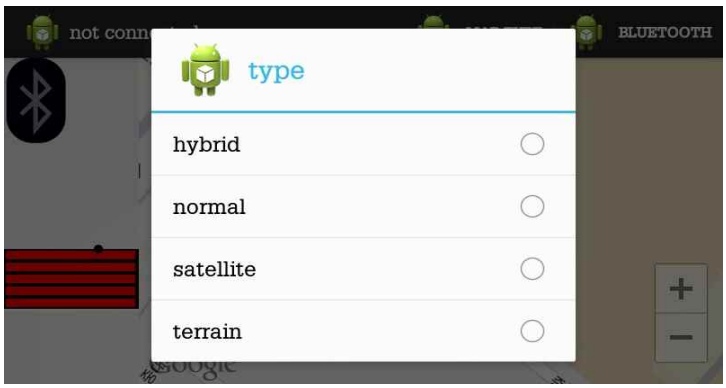
1) 블루투스 권한 요청에 대한 화면

2) 가속도 및 방향이 기준점 아래로 향하는 화면

3) 가속도 및 방향이 기준점 위로 향하는 화면



4) 블루투스로 연결된 디바이스의 현재 위치를 Maker로 표시하는 화면



5) 상단 액션바 중 맵 타입을 변경할 수 있는 옵션 다이얼로그 화면

6) 상단 액션바 중 주변 블루투스 기기를 연결 할 수 있는 옵션 다이얼로그 화면

3-2. 부록

1) Reference

- **주요 소스 :** 직접 구현한 소스 첨부 (사용 Tool : eclipse)
- **다이얼로그 및 순서도 :** 직접 제작 (사용 Tool : MS Office PowerPoint)
- **그림 :** 직접 구현하고 실행시킨 화면
- **클래스 다이어그램 :** 직접 제작 (StarUML)
- **작품 제작 참고 싸이트 :**
  안드로이드 개발자 공식 웹 : http://developer.android.com/
  안드로이드 폽 : http://www.androidpub.com/
  구글맵 참고 : https://developers.google.com/maps/?hl=ko
- **작품 제작 참고 서적 :**
  안드로이드 프로그래밍 (김상형 저, 한빛미디어) 외 각종 논문 및 문서

2) 소스 첨부

- 각 단원의 설명 상단 및 하단에서 '주요 소스'부분 참고