



Android 로케일과 센서 설정

모바일 응용

남 광 우

안드로이드 에뮬레이터 한글화

❖ 에뮬레이터의 로케일 설정

- ✓ 에뮬레이터를 실행 후, 메뉴에서 Settings 를 실행하고,
- ✓ Locale & text 를 선택하고,
- ✓ Select locale에서 Korean을 선택
- ✓ 화면을 스크롤할때에는, 터치폰을 사용하듯 마우스 클릭한로 드래깅(dragging)함

안드로이드 에뮬레이터 한글화

❖ 안드로이드 한글 자판 설치

- ✓ 박성서씨가 개발한 접촉식 한글자판을 설치
- ✓ 최신 자판의 apk 파일을 받음
 - <http://www.androidpub.com/keyboard>
- ✓ 받은파일 HanguKeyboard.apk 을 C:\Android\android-sdk-windows\tools 디렉토리로 옮기고
 - cmd 창을 띄우고 tools 디렉토리로 이동
 - android.bat 파일을 실행한 후 에뮬레이터 실행
 - apk 파일을 adb를 이용하여 설치
- ✓ 에뮬레이터 설정에서 로케일 키보드 설정
 - “한글 접촉식 키보드”만 남기고 나머지 키보드는 체크를 지움
- ✓ 마켓을 설치했을 경우에는 마켓에서 직접 다운로드 및 설치 가능
 - “Hangul”로 검색 =>Korean Hangul Keyboard

에뮬레이터에서 마켓 실행

❖ 1.5, 1.6에서 안드로이드 마켓 실행하기

- ✓ 기본적으로 안드로이드 SDK에는 안드로이드 마켓이 빠져있지만 Dev폰의 system 이미지 (system.img) 파일을 포팅함으로써 AVD에서도 안드로이드 마켓을 이용할 수 있음
- ✓ 수정 방식은 android SDK의 platforms 폴더에 각 버전별 android 데이터가 있는데...
이중 images 폴더의 system.img 를 교체한 후 avd를 생성함으로써 가능
 - dev 폰의 초기 설정 과정에서 슬라이드를 올리라고 나오는 부분이 있는데...
Ctrl+F11을 누르면 Landscape 모드로 바뀌며 슬라이드를 올린 것으로 인식이 됨
- ✓ 이미지
 - 1.5 (cupcake) : <http://www.kandroid.org/download/system.img>
 - 1.6 (donut) : <http://www.4shared.com/file/165624746/fc72c3ed/system.html>
 - Droid 이미지를 설치할 경우 korean 로케일 설정 안됨
- ✓ 마켓 이용해 보기
 - 네이버 및 게임 설치해보기

DDMS - 에뮬레이터에서 GPS와 통화/SMS 사용하기

❖ 안드로이드 DDMS

- ✓ DDMS = Dalvik Debug Monitor Server
- ✓ 에뮬레이터/단말기 내의 로그나 디버깅, 실행중인 프로세스 확인이나 화면 캡처 등의 작업을 수행할 수 있도록 해주기 위한 Eclipse용 툴

❖ DDMS의 지원 기능

- ✓ Device의 모니터링
- ✓ Device Thread 모니터링
- ✓ Application Tracker
- ✓ File Explorer
- ✓ 가상 Telephony/SMS 전송
- ✓ 가상 GPS 가상 전송
- ✓ 스크린 캡처

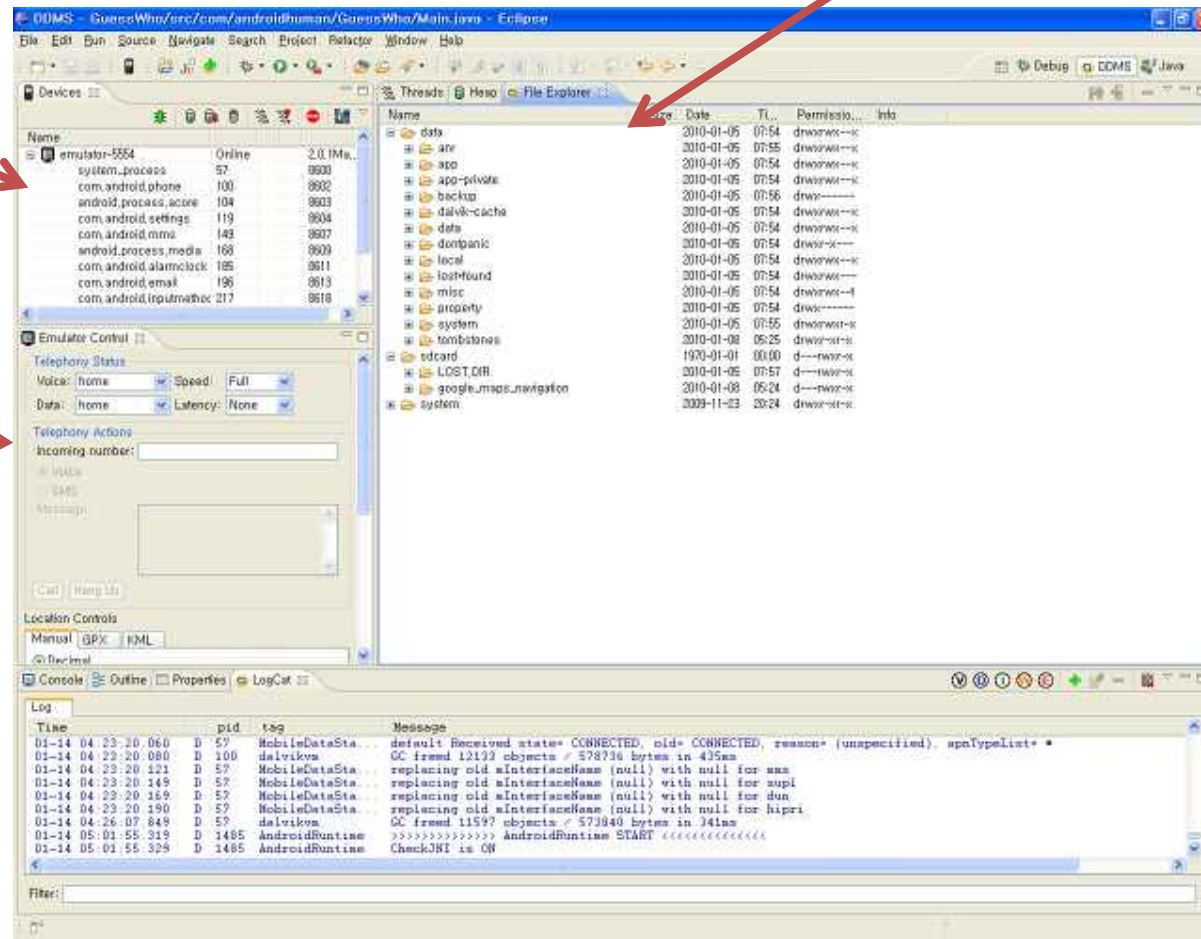
❖ Eclipse에서 DDMS Perspective 실행하기



❖ DDMS 화면을 통한 기능 개요

Device들 상태보기

Emulator
Control :
GPS/SMS
/Telephone



DDMS - 전화걸기

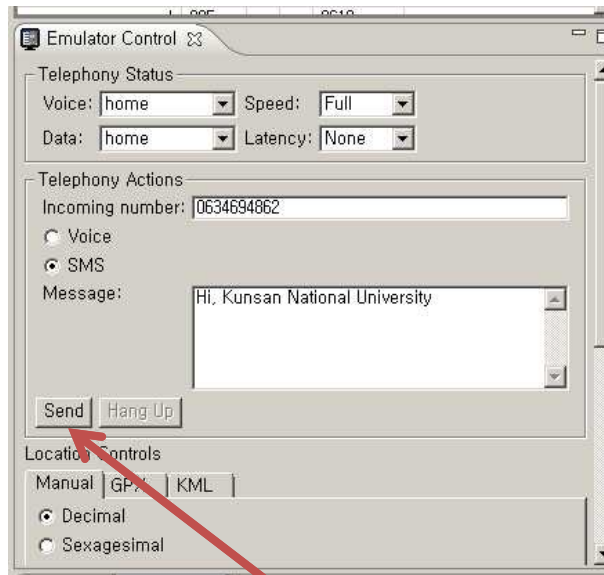
❖ DDMS Emulator Control을 통한 전화걸기

- ✓ 에뮬레이터의 통신 상태 조절 및 가상의 위치 정보를 설정해주는 것이 가능
- ✓ 에뮬레이터에 가상으로 전화를 걸거나 SMS를 전송 가능
 - 실제 기기를 선택한 상태에서는 이 메뉴를 사용할 수 없음
 - Contact에 전화번호가 있으면 이름이 보임

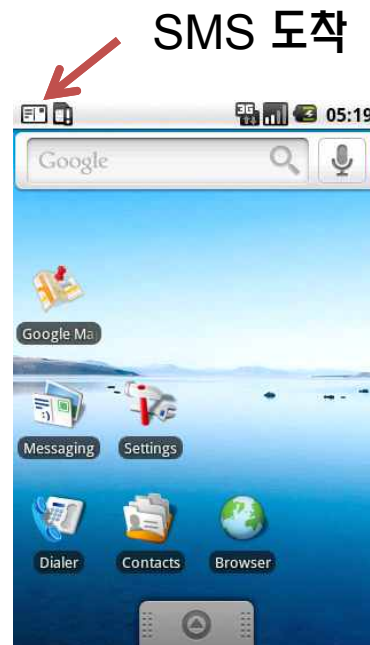


DDMS – SMS 보내기

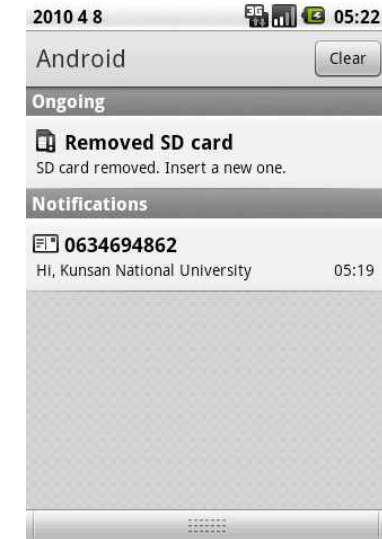
- ❖ DDMS Emulator Control을 통한 SMS 보내기
 - ✓ 한글 SMS는 화면에 깨져서 보이므로 영문으로 전송



SMS 보내기



SMS 도착

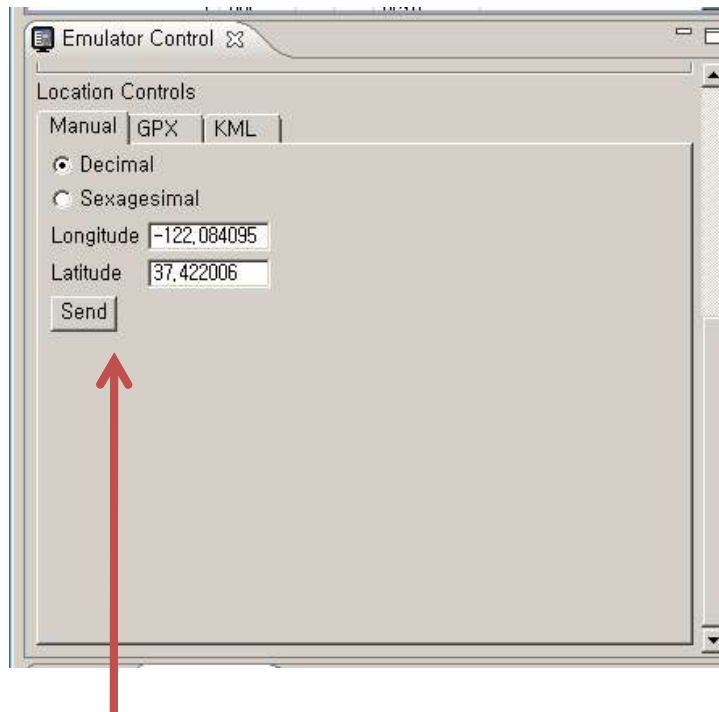


0634694862: Hi, Kunsan National University
Sent: 05:18

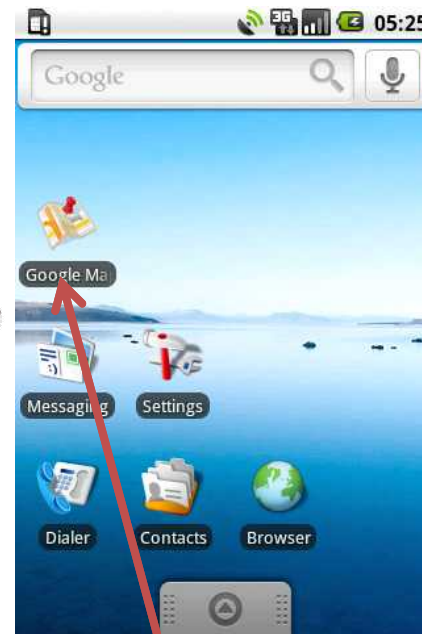
Type to compose Send

DDMS – GPS 값 보내기

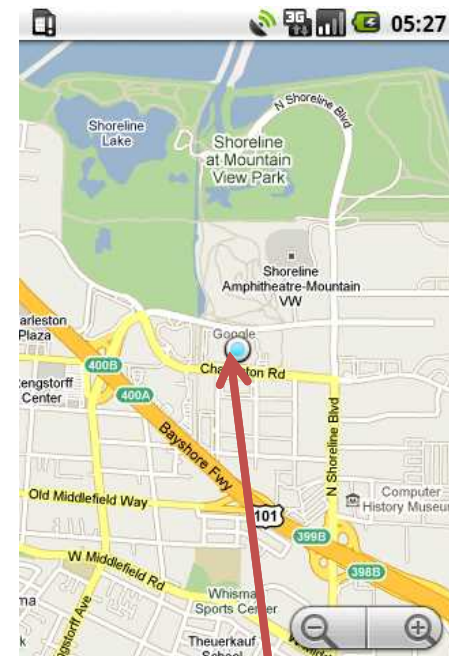
- ❖ DDMS Emulator Control에서 GPS값 보내기
 - ✓ 단일 GPS값 보내기



위경도 좌표설정 후 Send



안드로이드 마켓에서
구글맵설치후 실행

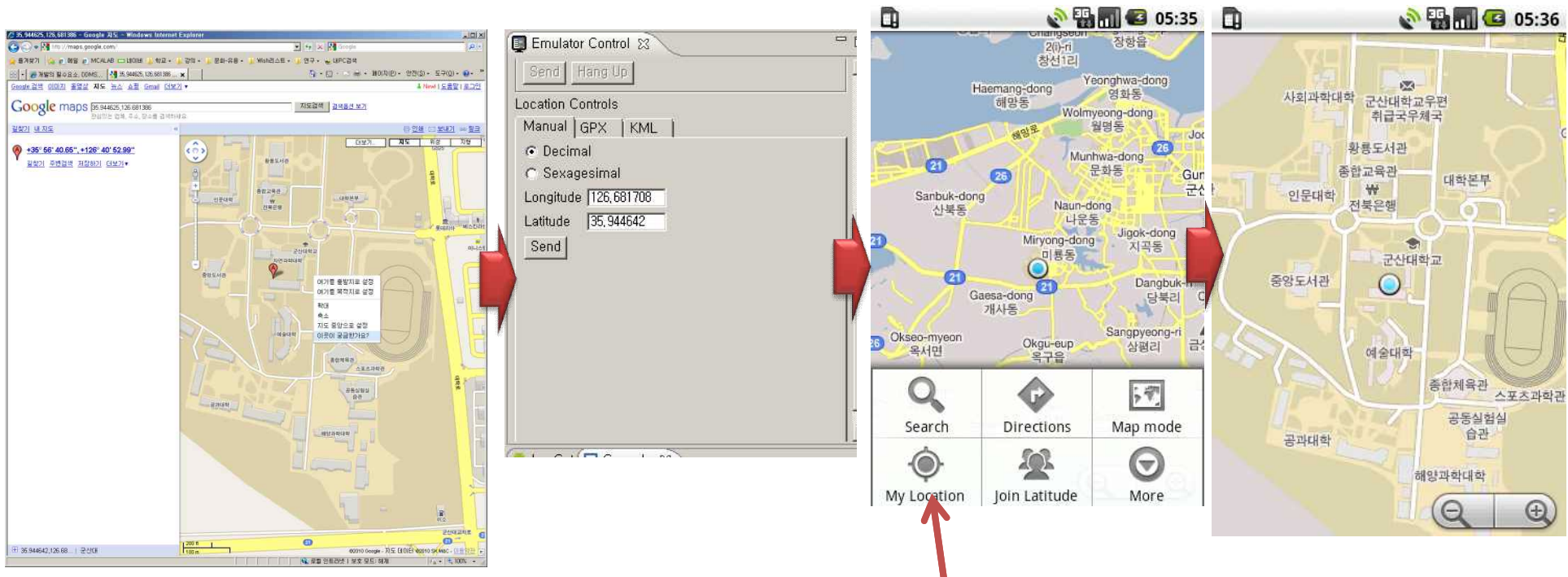


현재위치

DDMS – GPS 값 보내기

❖ DDMS Emulator Control에서 GPS값 보내기

✓ 위치 아는 방법: maps.google.com => 마우스 오른쪽버튼



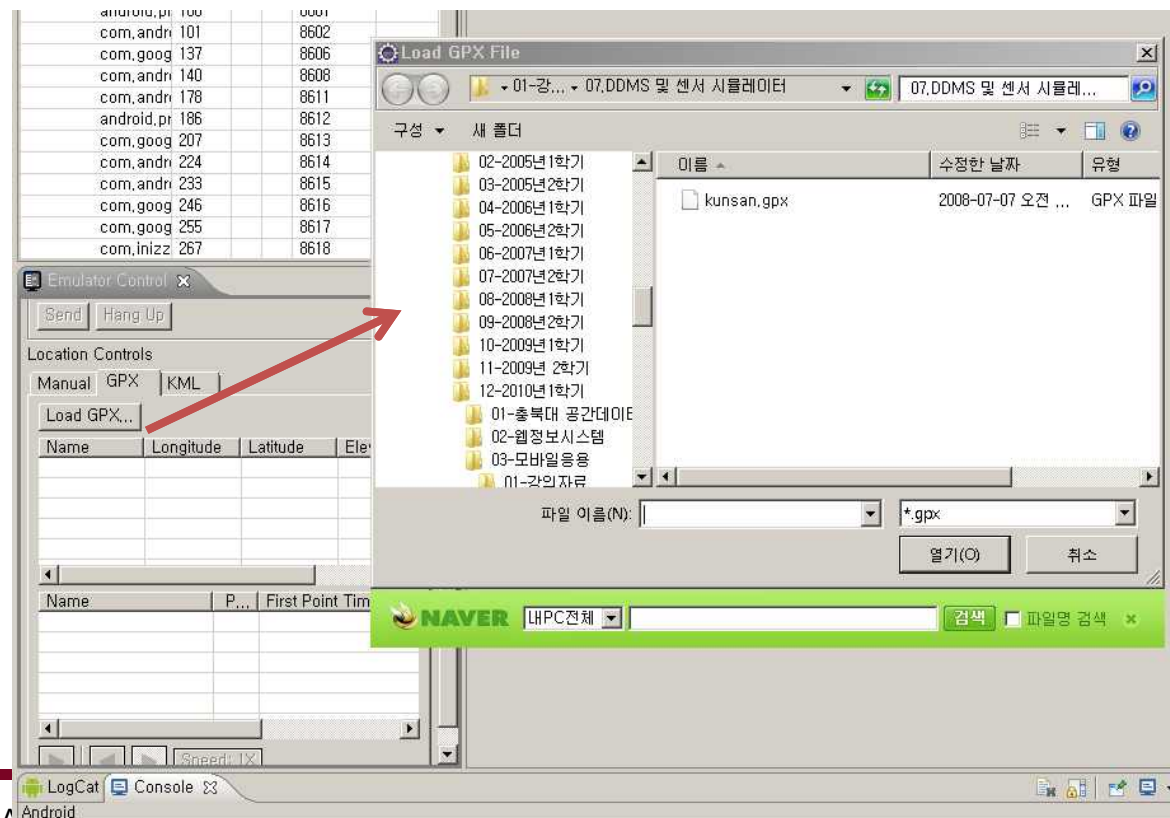
메뉴 버튼후 MyLocation

DDMS – GPS Tracking 정보 보내기

❖ DDMS Emulator Control에서 GPS값 보내기

✓ GPS Tracking 정보 보내기(GPX 데이터)

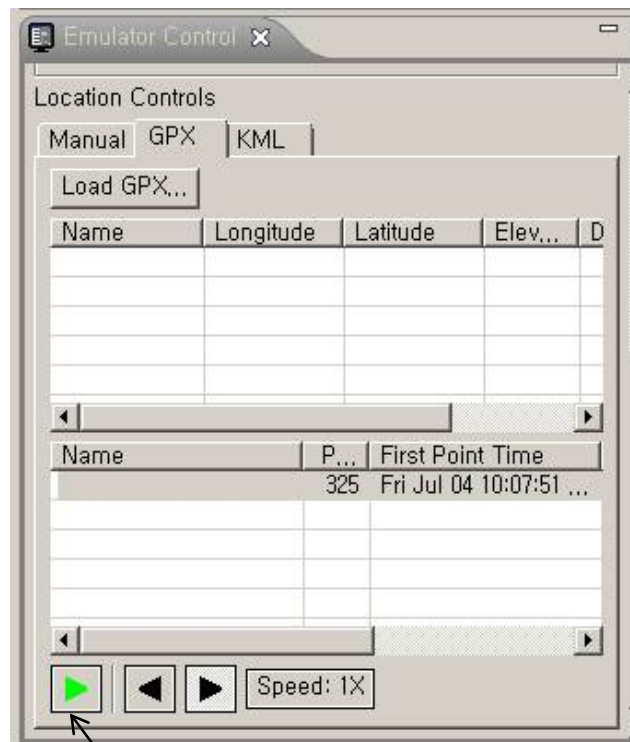
- mcalab.kunan.ac.kr => 강의게시판=> 강의자료05번 첨부파일
- GPX 1.1 데이터 => 인터넷 참고
 - KML 사용가능



DDMS – GPS Tracking 정보 보내기

❖ DDMS Emulator Control에서 GPS값 보내기

- ✓ GPS Tracking 정보 보내기(GPX 데이터)



Play 누르기



GPS 값 자동 전송됨

Sensor의 종류

- 모바일 디바이스에서 주로 사용되는 센서는 아래와 같고 대부분은 안드로이드에서는 대부분 지원한다.
- 가속도 : 기울기 정보
- 지자기 : 자기 세기 정보
- Orientation Sensor : 가속도와 지자기를 같이 감지할 수 있다. autocalibration 기능을 가지고 있다.
- 조도 ALS(Ambient Light Sensor) : 주변 밝기 정보 : lux 단위
- 근접 PS(Proximate Sensor) : 조도 센서와 같은 칩으로 사용되는 것이 대부분이며 물체와의 거리 정보를 알 수 있다. cm단위
- 자이로센서 : 좌우로 반복적 이동시 중간 데이터 처리 없이? 줄여 반응 속도를 개선한 것임

Android Sensor Class Type

- TYPE_ACCELEROMETER : 3차원 중력 가속도를 측정하는 가속도 센서
- TYPE_ALL : 모든 센서
- TYPE_GYROSCOPE : 회전운동 센서
- TYPE_LIGHT : 빛의 밝기를 측정하는 조도 센서
- TYPE_MAGNETIC_FIELD : 3차원 자기장 세기를 측정하는 지자기 센서(나침반)
- TYPE_ORIENTATION : 다른 장치들과 결합해 기기의 현재 방향결정
- TYPE_PRESSURE : 압력 센서
- TYPE_PROXIMITY : 어떤 물체와의 거리를 측정하는 근접 센서
- TYPE_TEMPERATURE : 온도를 측정하는 온도 센서
-

[출처] [Sensor 처리 과정](#) | 작성자 [즐겁게](#)

Sensor

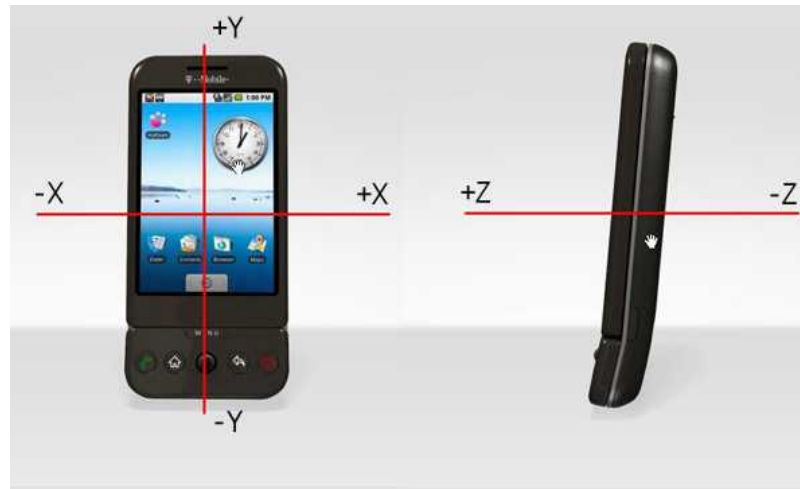
❖ 좌표 시스템

✓ 좌표 축 정의

X축 : 화면에 수평축(portrait 모드에서 짧은 에지를 landscape 모드에서 긴 에지)을 나타내며 오른쪽을 가리킨다.

Y축 : 화면에 수직축을 나타내며 화면 위쪽을 가리킨다.(원점은 왼쪽 밑 코너)

Z축 : 단말이 화면을 위로 해서 테이블에 올려져 있다고 생각했을 때 하늘을 가리킨다.



Sensor - Orientation

❖ Orientation

✓ SENSOR_ORIENTATION (방향 센서)

각 배열의 값은 각도를 나타냅니다.

단말기 화면이 하늘을 향한 상태로 테이블 위에 수평으로 놓여있는 상태를 기준으로 각 축을 중심으로 회전시키는 것을 생각하면 됩니다.

values[0] : Z 축을 중심으로 회전 ($0 \leq \text{azimuth} < 360$)

0 = 북, 90 = 동, 180 = 남, 270 = 서

values[1] : X축을 중심으로 회전 ($-180 \leq \text{pitch} \leq 180$)

Z축이 Y축 방향으로 향하면 0보다 큰값

화면이 하늘을 향하고 테이블위에 수평으로 놓여있는 상태 0, 화면이 아래를 향하면 -180 or 180,

똑바로 세우면 -90, 거꾸로 세우면 +90

values[2] : Y축을 중심으로 회전 ($-90 \leq \text{roll} \leq 90$)

Z축이 X축 방향으로 향하면 0보다 큰값

Sensro - Accelerometer

❖ 가속도계

✓ Accelerometer

SENSOR_ACCELOROMETER (가속도 센서)

각 배열의 값은 (m/s^2) 단위로 되어있으며, 접촉힘(Contact Force)을 측정합니다.

values[0] : X축에 적용되는 힘

values[1] : Y축에 적용되는 힘

values[2] : Z축에 적용되는 힘

❖ 지자기 센서

✓ Mgnetic field

SENSOR_MAGNETIC_FIELD (자기장 센서)

모든 값은 micro-Tesla (μT) 단위로 되어있으며, X, Y, -Z 축 주변 자기장을 측정합니다.

* Z축의 값이 바뀐 것 주의

Sensor Programming

❖ SensorEventListener(2.2)

✓ 예제 : 핸드폰 흔들림 감지 <http://pulsebeat.tistory.com/44>

```
public class ShakeActivity extends Activity implements SensorEventListener {  
  
    ....  
  
    private SensorManager sensorManager;  
    private Sensor accelerometerSensor;  
  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);  
        accelerometerSensor = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);  
    }  
  
    public void onStart() {  
        super.onStart();  
        if (accelerometerSensor != null)  
            sensorManager.registerListener(this, accelerometerSensor, SensorManager.SENSOR_DELAY_GAME);  
    }  
  
    public void onStop() {  
        super.onStop();  
        if (sensorManager != null)  
            sensorManager.unregisterListener(this);  
    }  
}
```

Sensor Programming

❖ onAccuracyChanged(), onSensorChanged()

```
public void onAccuracyChanged(Sensor sensor, int accuracy) {  
}  
  
@Override  
public void onSensorChanged(SensorEvent event) {  
    if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {  
        long currentTime = System.currentTimeMillis();  
        long gabOfTime = (currentTime - lastTime);  
        if (gabOfTime > 100) {  
            lastTime = currentTime;  
            x = event.values[SensorManager.DATA_X];  
            y = event.values[SensorManager.DATA_Y];  
            z = event.values[SensorManager.DATA_Z];  
            speed = Math.abs(x + y + z - lastX - lastY - lastZ) /  
                gabOfTime * 10000;  
            if (speed > SHAKE_THRESHOLD) {  
                // 이벤트 발생!!  
            }  
            lastX = event.values[DATA_X];  
            lastY = event.values[DATA_Y];  
            lastZ = event.values[DATA_Z];  
        }  
    }  
}
```

에뮬레이터에서 센서 시뮬레이터 이용하기

❖ SensorSimulator 개요

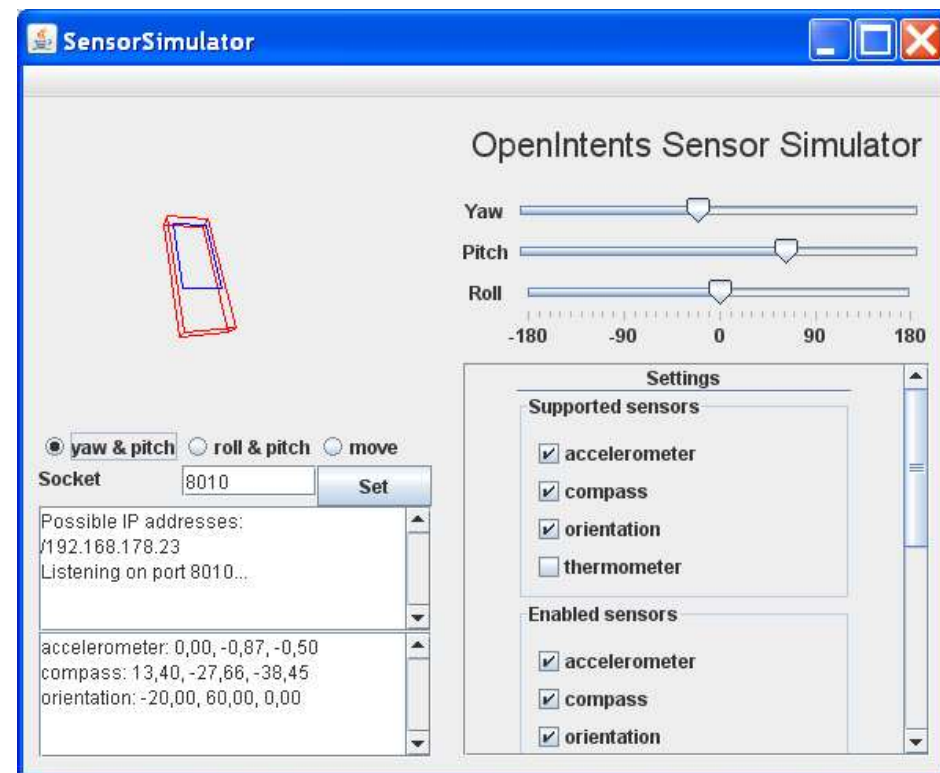
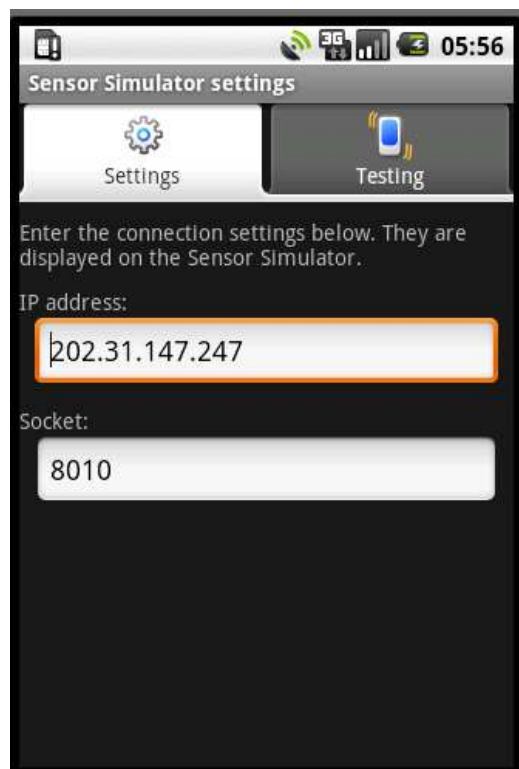
- ✓ 프로그래밍 용으로 센서데이터를 시뮬레이션해서 보낼 수 있는 오픈소스 프로그램
 - 기존의 프로그램에서 센서가 있는 것처럼 동작하게 하는 것이 아니라,
 - 이클립스에서 간단히 프로그램을 개발하면서 센서값을 테스트하고,
 - 실제 기기에 적용할때는 실제 클래스로 대체
 - `import org.openintents.sensorsimulator.hardware.SensorManagerSimulator;`

- ✓ 홈페이지 및 다운로드
 - <http://code.google.com/p/openintents/wiki/SensorSimulator>

에뮬레이터에서 센서 시뮬레이터 이용하기

❖ SensorSimulator의 화면 구성

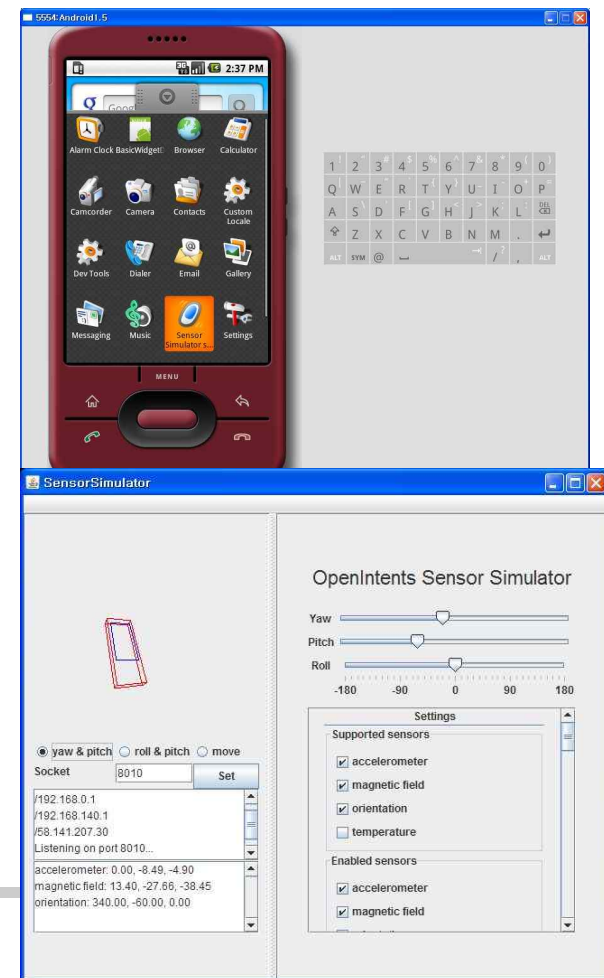
- ✓ 에뮬레이터용 SensorSimulatorSetting과 Java 프로그램 SensorSimulator로 구성



에뮬레이터에서 센서 시뮬레이터 이용하기

❖ SensorSimulator 설치 방법

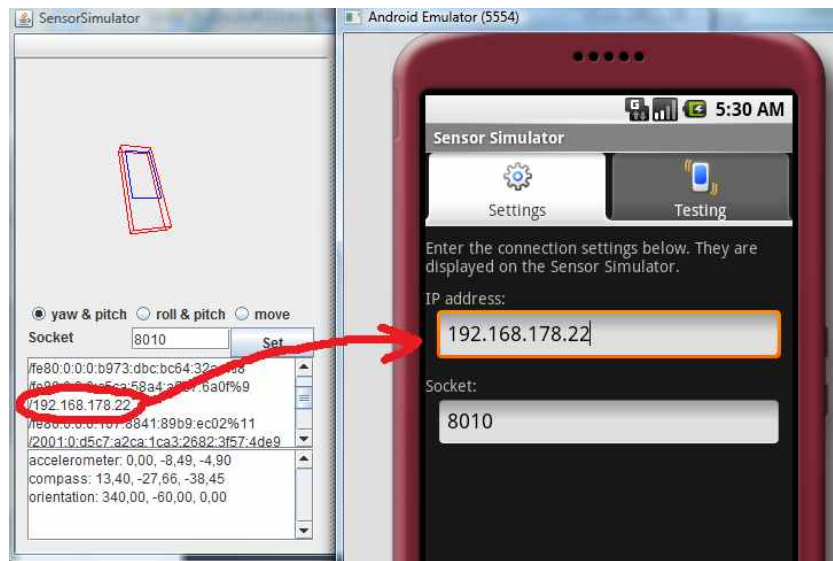
- ✓ 다운로드 후 zip 해제
- ✓ bin 디렉토리의 SensorSimulatorSetting.apk를 에뮬에 설치
 - apk 파일을 “sdk설치폴더\tools” 폴더로 옮김
 - 에뮬레이터 실행
 - sdk설치폴더\tools\android.bat 실행
 - android 버전 선택 및 에뮬 로딩
 - apk 인스톨
 - C:>adb install SensorSimulatorSetting.apk
 - 에뮬에서 설치 확인
- ✓ bin 디렉토리의 SensorSimulator.jar 실행
- ✓ 에뮬의 SensorSimulatorSetting 실행후 설정
- ✓ 테스트



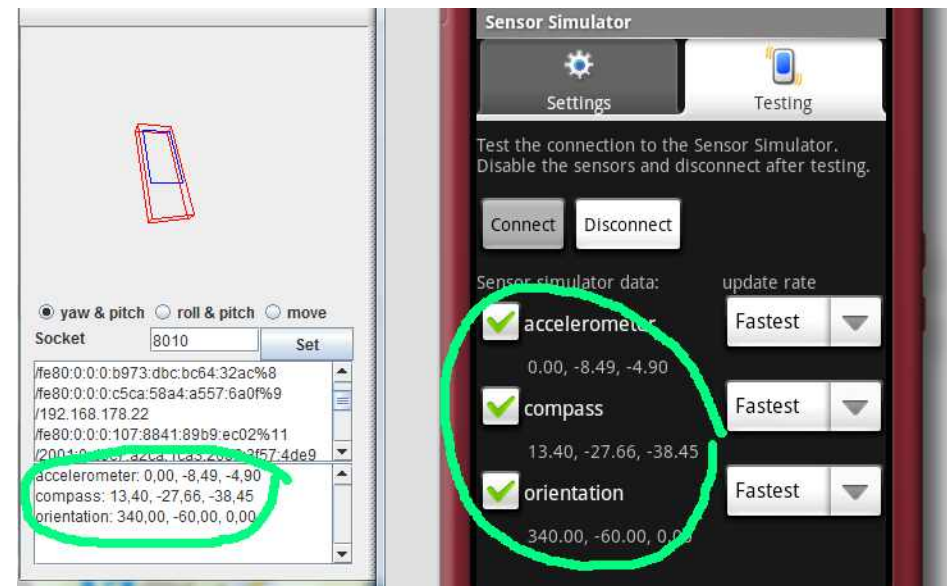
에뮬레이터에서 센서 시뮬레이터 이용하기

❖ SensorSimulator 설정 및 이용 방법

SensorSimulator의 IP 설정



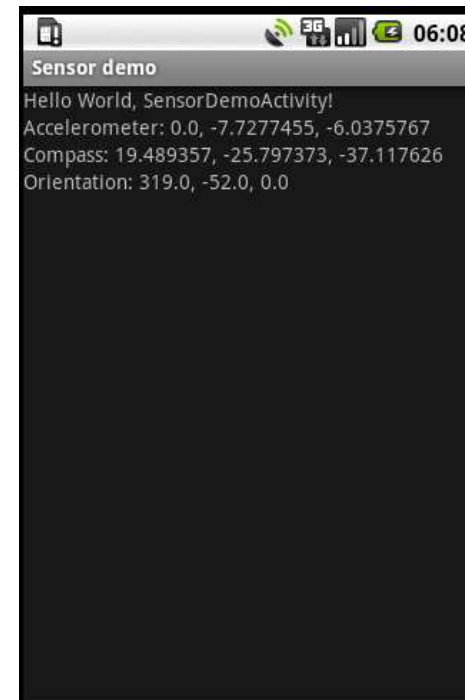
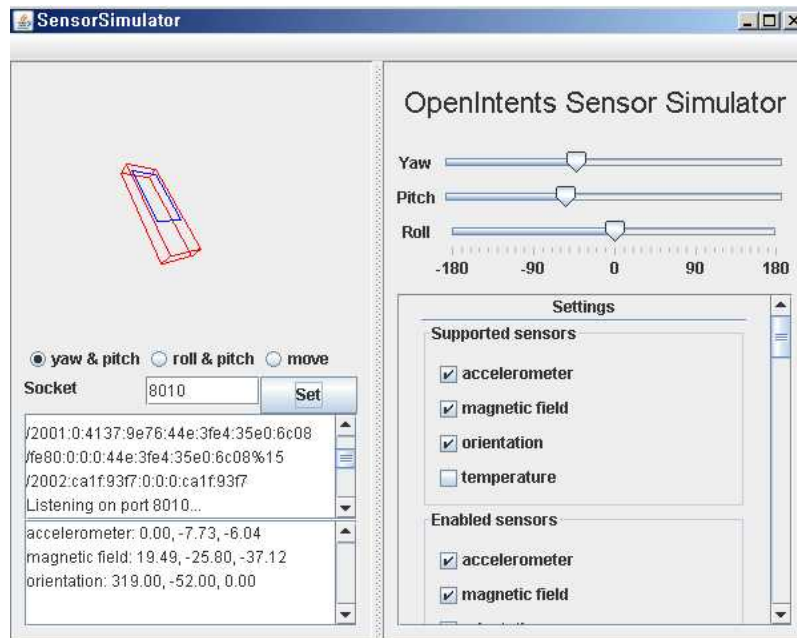
사용할 센서 설정



에뮬레이터에서 센서 시뮬레이터 이용하기

❖ SensorSimulator 센서 데모 프로그램 실행

- ✓ Sensorsimulator의 데모 프로그램 import 및 실행
 - \sensorsimulator-1.0.0-beta1\samples\SensorDemo



에뮬레이터에서 센서 시뮬레이터 이용하기

❖ SensorSimulator 센서 프로그래밍 예제

```
import org.openintents.sensorsimulator.hardware.Sensor;
import org.openintents.sensorsimulator.hardware.SensorEvent;
import org.openintents.sensorsimulator.hardware.SensorEventListener;
import org.openintents.sensorsimulator.hardware.SensorManagerSimulator;
```

```
public class SensorDemoActivity extends Activity implements SensorEventListener{
```

```
    private SensorManagerSimulator mSensorManager;
```

```
    TextView mTextView1;
    TextView mTextView2;
    TextView mTextView3;
```

```
    /** Called when the activity is first created. */
```

```
    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
```

```
        mTextView1 = (TextView) findViewById(R.id.text1);
        mTextView2 = (TextView) findViewById(R.id.text2);
        mTextView3 = (TextView) findViewById(R.id.text3);
        mSensorManager = SensorManagerSimulator.getSystemService(this, SENSOR_SERVICE);
```

```
        mSensorManager.connectSimulator();
```

```
    }
```

실제 기기에서 작동시
실제 SensorManager
클래스로 대체

실제 기기에서 작동시 다음과 같이 대체
mSensorManager =
(SensorManager) getSystemService(SENSOR_SERVICE);

실제 기기에서 작동시 필요없으므로 삭제

에뮬레이터에서 센서 시뮬레이터 이용하기

```
@Override
protected void onResume() {
    super.onResume();

    sensorMgr = SensorManagerSimulator.getSystemService(this, SENSOR_SERVICE);
    sensorMgr.connectSimulator();
    accelSupported = sensorMgr.registerListener(this, sensorMgr.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
        SensorManager.SENSOR_DELAY_FASTEST);
}

@Override
protected void onStop() {
    mSensorManager.unregisterListener(this);
    super.onStop();
}

public void onAccuracyChanged( Sensor sensor, int accuracy ) {
}

public void onSensorChanged( SensorEvent event ) {
    int sensorType = event.type;
    float[] values = event.values;
    switch(sensorType) {
        case Sensor.TYPE_ACCELEROMETER:
            mTextView1.setText("Accelerometer: " + values[0] + ", " + values[1] + ", " + values[2]);
            break;
        case Sensor.TYPE_MAGNETIC_FIELD:
            mTextView2.setText("Compass: " + values[0] + ", " + values[1] + ", " + values[2]);
            break;
        case Sensor.TYPE_ORIENTATION:
            mTextView3.setText("Orientation: " + values[0] + ", " + values[1] + ", " + values[2]);
            break;
    }
}
```

다른 부분은 실제 기기에서 동작시
변경할 필요 전혀 없음

