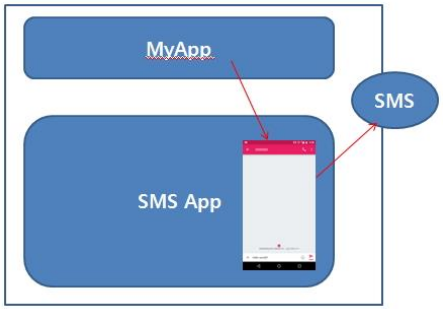
27장. 스마트폰 기능 활용

27.1 SMS 송수신

27.1.1. SMS 송신

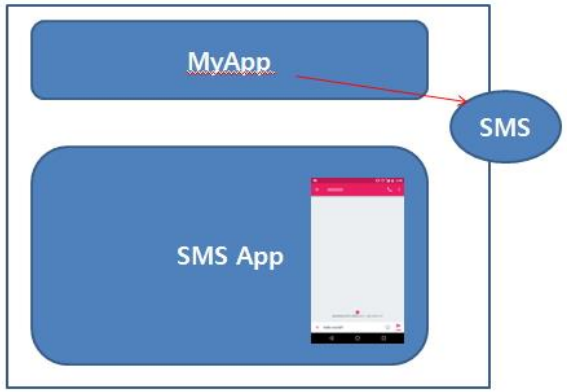
- SMS 앱 연동
- SMS 데이터를 SMS 앱에 전달하여 SMS 앱에서 데이터를 발송하는 방법




```
Uri uri = Uri.parse("smsto:0000-0000");
Intent smsIntent = new Intent(Intent.ACTION_SENDTO, uri);
smsIntent.putExtra("sms_body", "hello world!!");
startActivity(smsIntent);
```

27.1 SMS 송수신

- SmsManager 이용
- 앱에서 직접 SMS를 발송




```
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
```

- 사용자의 전화번호 추출

```
TelephonyManager telephony = (TelephonyManager) getSystemService(TELEPHONY_SERVICE);
String myNumber = telephony.getLine1Number();
```

- SMS를 발송

```
SmsManager smsManager= SmsManager.getDefault();
smsManager.sendTextMessage(phoneNumber, myNumber, message, null, null);
```

27.1 SMS 송수신

- sendTextMessage( ) 함수의 sentIntent, deliveryIntent 매개변수를 지정하여 발송 후 성공 실패를 판단
- 발송 확인 시 실행될 브로드캐스트 리시버를 준비

```
BroadcastReceiver sentReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        String msg="";
        switch (getResultCode()) {
            case Activity.RESULT_OK:
                msg="sms 전송 성공";
                break;
            case SmsManager.RESULT_ERROR_GENERIC_FAILURE:
                msg="sms 전송 실패";
                break;
            case SmsManager.RESULT_ERROR_RADIO_OFF:
                msg="무선 꺼짐";
                break;
            case SmsManager.RESULT_ERROR_NULL_PDU:
                msg="pdu 오류";
                break;
        }
        showToast(msg);
    }
};
```

27.1 SMS 송수신

27.1.2. SMS 수신

- 브로드캐스트 리시버를 등록

```
registerReceiver(sentReceiver, new IntentFilter("ACTION_SENT"));
```

  - 브로드캐스트 리시버가 실행되게 PendingIntent를 등록

```
Intent sentIntent=new Intent("ACTION_SENT");
PendingIntent sentPIntent=PendingIntent.getBroadcast(this, 0, sentIntent,
PendingIntent.FLAG_UPDATE_CURRENT);

SmsManager smsManager= SmsManager.getDefault();
smsManager.sendTextMessage(phoneNumber, myNumber, message, sentPIntent, null);
```

  - SMS 수신

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

27.1 SMS 송수신

- SMS 데이터를 수신하기 위해서는 브로드캐스트 리시버를 작성

```
<receiver
    android:name=".SMSReceiver"
    android:enabled="true"
    android:exported="true">
    <intent-filter>
        <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
    </intent-filter>
</receiver>
```

- 브로드캐스트 리시버에서 수신한 SMS 데이터를 획득

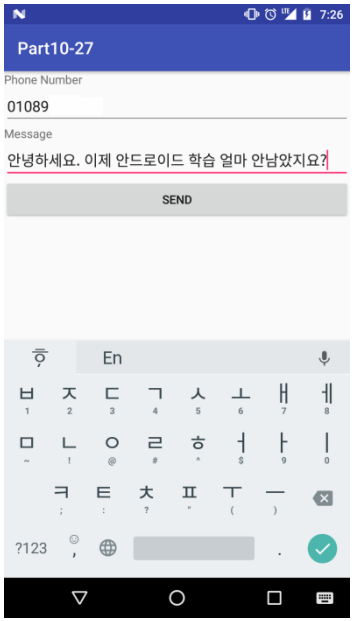
```
Bundle bundle = intent.getExtras();
Object[] pdus = (Object[]) bundle.get("pdus");
SmsMessage[] messages = new SmsMessage[pdus.length];
for (int i = 0; i < pdus.length; i++) {
    messages[i] = SmsMessage.createFromPdu((byte[]) pdus[i]);
    try {
        String message = new String(messages[i].getMessageBody());
        String phoneNumber = messages[i].getOriginatingAddress();
} catch (Exception e) {
        e.printStackTrace();
    }

}
```

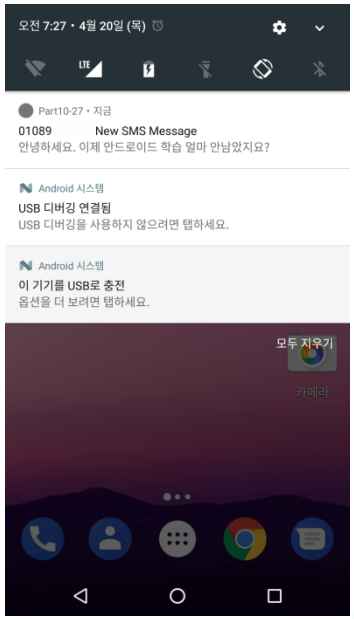
Step by Step 27-1 – SMS 송수신

SMS 송수신을 테스트

•하나의 앱에서 송신과 수신을 모두 테스트
•유저가 입력한 전화번호와 문자열로 SMS 송신을 하고 SMS 수신은 Notification으로 유저에게 알리는 테스트

1. Module 생성
2. 파일 복사
3. AndroidManifest.xml 작업
4. MainActivity.java 작성
5. BroadcastReceiver 생성
6. SMSReceiver 작성
7. 실행





27.2 센서 다루기

- 센서의 목록을 가져오는 방법

```
SensorManager sm = (SensorManager) this.getSystemService(SENSOR_SERVICE);
List<Sensor> sensors = sm.getSensorList(Sensor.TYPE_ALL);
```

- 센서 하나에 대한 정보가 Sensor 객체로 전달

  - sensor.getName(): 센서 이름
  - sensor.getType(): 센서 타입
  - sensor.getVendor(): 센서 제조사
  - sensor.getVersion(): 센서 버전
  - sensor.getPower(): 전력 사용량

27.2 센서 다루기

- Sensor.TYPE_ACCELEROMETER: 가속도 센서
- Sensor.TYPE_MAGNETIC_FIELD: 자기장 센서
- Sensor.TYPE_GYROSCOPE: 자이로스코프 센서
- Sensor.TYPE_LIGHT: 조도 센서
- Sensor.TYPE_PRESSURE: 압력센서
- Sensor.TYPE_TEMPERATURE: 온도센서
- Sensor.TYPE_PROXIMITY: 근접센서
- Sensor.TYPE_GRAVITY: 중력센서
- Sensor.TYPE_LINEAR_ACCELERATION: 선형 가속도 센서
- Sensor.TYPE_ROTATION_VECTOR: 회전센서
- Sensor.TYPE_AMBIENT_TEMPERATURE: 주변온도센서
- Sensor.TYPE_STEP_DETECTOR: 걸음 감지 센서
- Sensor.TYPE_SIGNIFICANT_MOTION: 특정 모션 트리거 센서
- Sensor.TYPE_GEOMAGNETIC_ROTATION_VECTOR: 자석 회전 벡터 센서

27.2 센서 다루기

27.2.1. 근접 센서 다루기

- 근접 센서의 사용 예

```
SensorManager manager = (SensorManager) this.getSystemService(SENSOR_SERVICE);
Sensor proximity = manager.getDefaultSensor(Sensor.TYPE_PROXIMITY);
if(proximity != null){
    proximityMaximumRange=proximity.getMaximumRange();
}
```

- 센서의 값을 얻는 방법은 SensorEventListener를 구현한 객체를 등록

```
manager.registerListener(listener, proximity, SensorManager.SENSOR_DELAY_UI);
```

- SENSOR_DELAY_FASTEST: 가능한 한 가장 빠른 속도
- SENSOR_DELAY_GAME: 게임에 적합한 속도
- SENSOR_DELAY_UI: UI 수정에 적합한 속도
- SENSOR_DELAY_NORMAL: 화면 방향 변화를 모니터링하기에 적합한 속도

- 등록 해제

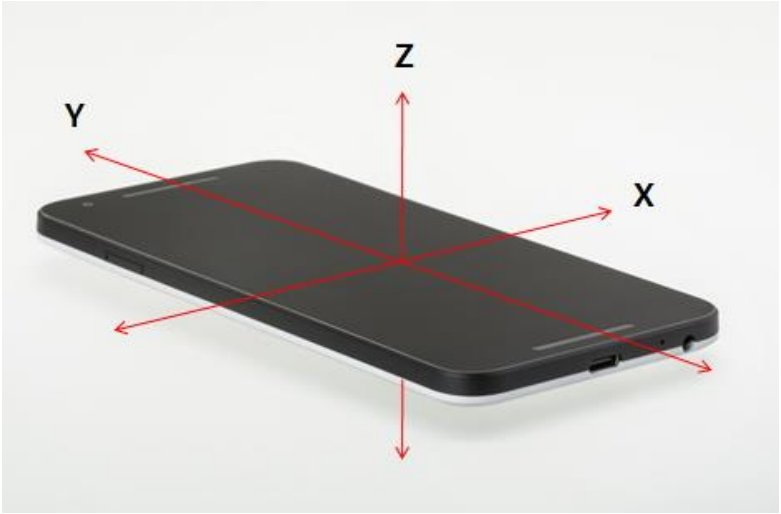
```
manager.unregisterListener(listener);
```

27.2 센서 다루기

- 센서값을 받을 객체를 SensorEventListener로 구현

```
SensorEventListener listener=new SensorEventListener() {
    @Override
    public void onSensorChanged(SensorEvent event) {
        float distance = event.values[0];
        //...
    }
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {

    }
};
```

27.2.2. 가속도 센서 다루기



27.2 센서 다루기

```
SensorManager manager = (SensorManager) this.getSystemService(SENSOR_SERVICE);
Sensor accelerometer=manager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
```

- SensorManager에 등록하여 센서값을 획득

```
manager.registerListener(listener, accelerometer, SensorManager.SENSOR_DELAY_NORMAL);
```

- 센서값은 등록된 SensorEventLister 객체의 함수가 호출되면서 전달

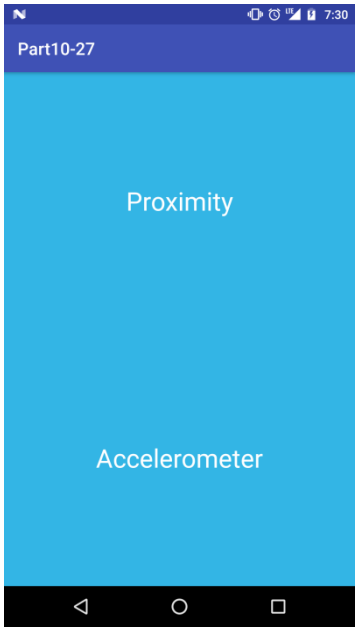
```
SensorEventListener listener=new SensorEventListener() {
    @Override
    public void onSensorChanged(SensorEvent event) {
                if(event.sensor.getType()==Sensor.TYPE_ACCELEROMETER){
            float xValue=Math.abs(event.values[0]-lastX);
            float yValue=Math.abs(event.values[1]-lastY);
            float zValue=Math.abs(event.values[2]-lastZ);
            //.........
        }
    }
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }
};
```

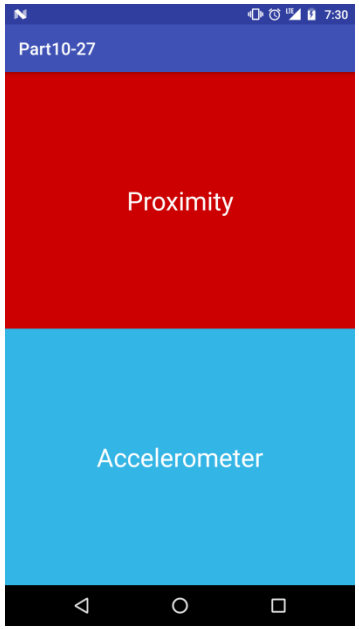
Step by Step 27-2- 센서

Sensor 값을 획득하는 테스트

•근접 센서와 가속도 센서값을 획득하여 값이 변경되는 순간 Activity의 색상을 변경하는 간단한 테스트

1. Activity 생성
2. 파일 복사
3. Lab27_2Activity 작성
4. Lab27_2Activity.java 실행





27.3 블루투스

27.3.1. 블루투스 제어

- 스마트폰과 스마트폰 혹은 스마트폰과 마우스 등 블루투스를 지원하는 다양한 기기와의 근접 통신에 이용

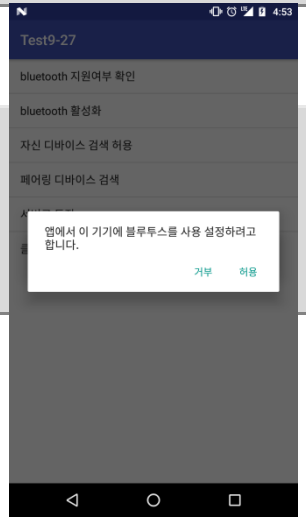
```
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
```

- 스마트폰이 블루투스를 지원하는지를 알아보는 작업

```
BluetoothAdapter ap = BluetoothAdapter.getDefaultAdapter();
if (ap == null)
    showToast("bluetooth 를 지원하지 않습니다.");
else
    showToast("bluetooth 를 지원합니다.");
```

- 블루투스 활성 상태 변경

```
BluetoothAdapter ap1 = BluetoothAdapter.getDefaultAdapter();
if (!ap1.isEnabled()) {
    Intent bIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(bIntent, 5);
}
```



- 위의 다이얼로그에서 사용자가 거부한 건지 허용한
- 건지를 판단 필요

27.3 블루투스

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    switch (requestCode) {
        case 5:
            // bluetooth 활성화
            if (resultCode == RESULT_OK)
                showToast("bluetooth 가 활성화 되었습니다.");
            else if (resultCode == RESULT_CANCELED)
                showToast("bluetooth 가 활성화 되지 못하였습니다.");
    }
}
```

- 블루투스를 활성화하는 다른 방법

```
BluetoothAdapter ap1 = BluetoothAdapter.getDefaultAdapter();

if (!ap1.isEnabled()) {
    ap1.enable();
}else {
    ap1.disable();
}
```

27.3 블루투스

- 기기 검색 허용

```
Intent dIntent = new Intent(
        BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
dIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION, 100);
startActivity(dIntent);
```

- 페이링 기기 검색

```
BluetoothAdapter ap2 = BluetoothAdapter.getDefaultAdapter();
Set<BluetoothDevice> devices = ap2.getBondedDevices();

if (devices.size() > 0) {
    Iterator<BluetoothDevice> iter = devices.iterator();
    while (iter.hasNext()) {

        BluetoothDevice d = iter.next();
        deviceArray[i] = d.getName();
        Log.d("kkang", "device info:" + d.getName() + "-"
                + d.getAddress());
    }
}
```

27.3 블루투스

- 페어링 되는 순간을 감지하려면 브로드캐스트 리시버를 이용

```
<receiver
   android:name=".BluetoothReceiver">
   <intent-filter>
      <action android:name="android.bluetooth.device.action.FOUND"></action>
   </intent-filter>
</receiver>
```

- 브로드캐스트 리시버에서 현재 페어링 된 기기 정보를 추출

```
@Override
public void onReceive(Context context, Intent intent) {
   BluetoothDevice device=intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
}
```

27.3 블루투스

27.3.2. 블루투스 통신

- 서버로 동작
- 블루투스 서버로 동작하게 하는 방법은 BluetoothServerSocket 클래스를 이용

```
UUID MY_UUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");BluetoothServerSocket
serverSocket = adapter.listenUsingRfcommWithServiceRecord("kkang_test",MY_UUID);
```

- BluetoothServerSocket의 accept( ) 함수가 실행되면 클라이언트에서 연결 요청이 오기까지 대기 상태

```
BluetoothSocket  socket = serverSocket.accept();
```

27.3 블루투스

- 클라이언트로 동작
- 서버에 연결 요청을 하기 전에 BluetoothAdapter의 cancelDiscovery( ) 함수를 호출

```
BluetoothAdapter adapter = BluetoothAdapter.getDefaultAdapter();
adapter.cancelDiscovery();
```

- UUID로 식별되는 서버와 연결을 시도

```
BluetoothSocket clientSocket = device.createRfcommSocketToServiceRecord(MY_UUID);
clientSocket.connect();
```

- 데이터 전송

```
out=clientSocket.getOutputStream();
out.write("kkang".getBytes());
out.flush();
```

- 데이터 수신

```
byte[] buffer = new byte[1024];
int bytes;
in=new BufferedInputStream(clientSocket.getInputStream());
bytes=in.read(buffer);
```

Step by Step 27-3 – 블루투스 채팅

블루투스를 테스트

•간단한 블루투스 채팅으로 테스트

1. Activity 생성
2. 파일 복사
3. AndroidManifest.xml 작업
4. Lab27_3Activity 작성
5. Lab27_2Activity.java 실행