BLUETOOTH LIST

2020년도 1학기 앱 프로그래밍 9주차 수업

목 차

I 교시

2020년도 1학기 앱 프로그래밍 – SENSOR LIST

1. 센서 목록

- 정의
  - 안드로이드 디바이스의 센서 목록화
  - TextView 위젯에 List<>로 수집한 장치 목록 적용

- 센서 목록화
  - 장치에 포함된 센서를 목록화 한다.
  - SensorManager 클래스의 getSensorList() 메서드 사용
    - 안드로이드 컵 케이크(CUPCAKE) 부터 지원

- 금주 학습 주제
  - 안드로드 장치 → 블루투스 장치 사용 및 목록화

1. 센서 목록 (계속)

- XML Code
  - View
    - LinearLayout
    - TextView

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/tvList"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

1. 센서 목록 (계속)

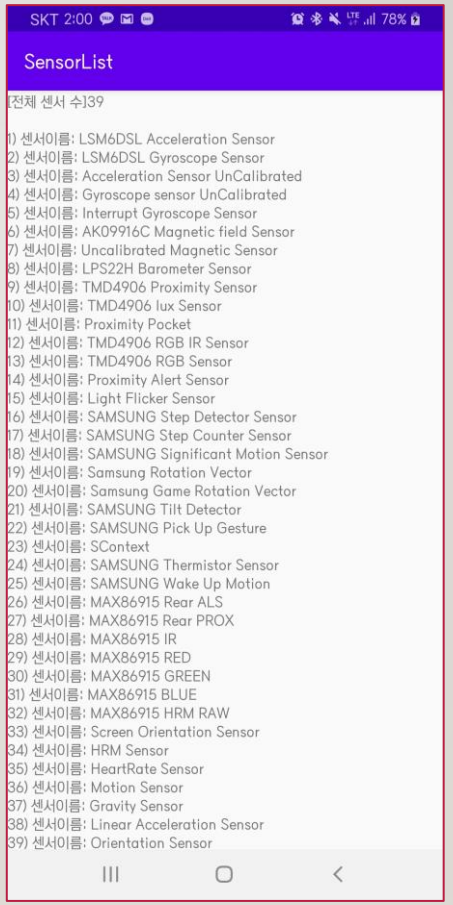
- Java Source Code
  - Core Class & Instance
    - SensorManager
    - List<>
  - Core Method
    - getSystemService()
    - getSensorList()

```java
import java.util.List;

public class MainActivity extends AppCompatActivity {
    TextView tvList;
    String result = "";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        tvList = (TextView)findViewById(R.id.tvList);

        // 센서 매니저 객체
        SensorManager manager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        // 센서 목록 획득
        List<Sensor> sensors = manager.getSensorList(Sensor.TYPE_ALL);
        // tvList에 출력
        result += "[전체 센서 수]" + sensors.size() + "\n\n";
        int iNum = 0;

        for(Sensor s : sensors)
            result += ++iNum + ") 센서이름: " + s.getName() + "\n";

        tvList.setText(result);
    }
}
```

1. 센서 목록 (계속)

- 실행 결과
  - 실행 환경
    - AVD
    - Galaxy S9+

AVD



Galaxy S9+

2교시

2020년도 1학기 앱 프로그래밍 – BLUETOOTH

3.1 BLUETOOTH 개요

- Android 플랫폼 블루투스 네트워크 스택 지원
  - 블루투스 기기와 데이터 무선 교환 기능 수행
  - 제공 프레임워크 → Android Bluetooth API를 통한 접근 권한 제공

- 수행 가능 작업
  - 주변 블루투스 장치 검색(스캔)
  - 페어링된 블루투스 장치에 대한 로컬 블루투스 어댑터 쿼리
  - 서비스 검색을 통한 다른 장치 연결
  - 기기 간 데이터 전송 및 수신
  - 다중 연결 관리

3.2 BLUETOOTH 기본

- 블루투스 지원 기기 데이터 전송
  - 페어링 프로세스를 사용하여 통신 채널을 형성
  - 검색 가능한 장치가 수신되는 연결 요청 용도로 사용
  - 서비스 검색 프로세스를 사용해 검색 가능한 기기 찾기
  - 검색 가능한 기기가 페어링 요청 수락
    - 두 장치 간 보안 키 교환 후 연결프로세스 완료
    - 보안 키는 차후 사용을 위해 기기에 보관(캐싱)
  - 두 장치는 페어링과 연결 프로세스를 완료한 뒤 정보 교환
  - 세션이 완료되면
    - 페어링 요청을 시작한 기기가 자신을 검색 가능한 기기와 연결해준 채널 해제
    - 두 기기는 연결된 상태로 유지
      - 서로 범위 내에 있는 상태 / 두 기기 중 어느 쪽도 연결을 삭제하지만 않는 경우
      - 향후 세션에서도 자동으로 다시 서로 연결할 수 있음

3.2 블루투스 권한

- 블루투스 기능 사용 권한 2가지
  - BLUETOOTH
    - 연결 요청 / 연결 수락 / 데이터 전송 의 블루투스 통신 수행에 필요
  - ACCESS_FINE_LOCATION
    - 블루투스 스캔을 사용하여 사용자 위치에 대한 정보 수집에 사용
    - 위치 정보 → 사용자 본인 기기의 위치 정보 / 공공 정보의 블루투스 비콘 위치 정보
- BLUETOOTH_ADMIN
  - 앱에서 기기 검색을 시작 및 블루투스 설정 조작에 사용
  - 일반적으로 로컬 블루투스 기기 검색 기능에 사용
    - "파워 관리자" 이외에 블루투스 설정 및 수정 등의 다른 기능 사용 지양

AndroidManifest.xml

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

3.3 블루투스 프로필 작업

- 장치 간 블루투스 기반 통신에 대한 무선 인터페이스 사양
  - Android Bluetooth API 제공 프로필
    - 헤드셋
      - 헤드셋 지원
    - A2DP
      - 고품질 오디오
    - 의료 기기
      - 심박 측정 등

```
BluetoothHeadset bluetoothHeadset;

// Get the default adapter
BluetoothAdapter bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

private BluetoothProfile.ServiceListener profileListener = new BluetoothProfile.ServiceListener() {
    public void onServiceConnected(int profile, BluetoothProfile proxy) {
        if (profile == BluetoothProfile.HEADSET) {
            bluetoothHeadset = (BluetoothHeadset) proxy;
        }
    }
    public void onServiceDisconnected(int profile) {
        if (profile == BluetoothProfile.HEADSET) {
            bluetoothHeadset = null;
        }
    }
};

// Establish connection to the proxy.
bluetoothAdapter.getProfileProxy(context, profileListener, BluetoothProfile.HEADSET);

// ... call functions on bluetoothHeadset

// Close proxy connection after use.
bluetoothAdapter.closeProfileProxy(bluetoothHeadset);
```

3.4 블루투스 설정

- 블루투스 장치 지원 확인 및 활성화
  - 지원 기기의 블루투스 활성화 전환
    - 어플리케이션에서 장치 확인과 비활성화 상태에서 활성화 상태로 설정 변경

- 구현 : BluetoothAdapter 클래스 사용 2단계 설정
  1. BluetoothAdapter 인스턴스 화 → getDefaultAdapter() 메서드 사용
     - null 반환 시 블루투스 미지원 장치

```
// 로컬 Bluetooth 어뎁터 가져오기...
mBtAdapter = BluetoothAdapter.getDefaultAdapter();
if (mBtAdapter == null) {
    // 블루투스를 지원하지 않는 장치의 경우 처리 코드 삽입
    Toast.makeText(getApplicationContext(), text: "미지원 기기입니다. 앱을 종료합니다.", Toast.LENGTH_SHORT).show();
    finish();
}
```

3.4 블루투스 설정 (계속)

- 구현 : BluetoothAdapter 클래스 사용 2단계 설정
  2. 블루투스 활성화
     - isEnabled() 메서드를 통한 활성화 상태 확인
       - false 반환 시 비 활성화 상태 / true 반환 시 활성화 상태
       - 블루투스 활성화 요청
         - startActivityForResult() 메서드 호출 → Intent와 ACTION_REQUEST_ENABLE 매개변수 사용
       - 요청 결과 처리
         - onActivityResult() 메서드의 resultCode 확인 및 처리 → RESULT_OK 또는 RESULT_CANCELED

```
// 블루투스가 꺼져 있는 경우 활성화하도록 요청한다.
if (!mBtAdapter.isEnabled()) {
    Intent enableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableIntent, ACTION_REQUEST_ENABLE); // 블루투스 활성화 응답에 대한 처리 반영
}
```

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if(resultCode == RESULT_OK){
        Toast.makeText(getApplicationContext(), text: "블루투스 활성화!", Toast.LENGTH_SHORT).show();
    }
    else{
        Toast.makeText(getApplicationContext(), text: "블루투스 활성화 실패!\n앱을 종료합니다.", Toast.LENGTH_SHORT).show();
        finish();
    }
}
```

3.5 블루투스 기기 찾기

- BluetoothAdapter 사용
  - 원격 블루투스 기기 찾기
    - 기기 검색 및 페어링 된 기기 목록 쿼리
- 페어링과 연결의 차이점
  - 페어링
    - 두 기기가 서로의 존재를 알고있는 상태
    - 인증에 사용할 수 있는 공유 링크 키 소유
    - 서로 암호화된 연결 설정 가능
  - 연결
    - 기기가 현재 RFCOMM 채널 공유 상태
    - 데이터를 서로 전송할 수 있음
- Android Bluetooth API
  - RFCOMM 연결을 설정 전에 기기 페어링 하도록 요청
  - Bluetooth API와 암호화된 연결을 시작하면 페어링 자동 실행

3.6 블루투스 페어링 기기 쿼리

- 기기 검색 전 페어링된 기기 집합 쿼리 → 연결을 위한 장치 확인
  - getBondedDevices() 메서드
    - BluetoothDevice 객체 세트 반환
      - 페어링 된 장치들의 이름과 MAC 주소 확인 가능

```
// 현재 페어링된 장치 세트 가져오기...
Set<BluetoothDevice> pairedDevices = mBtAdapter.getBondedDevices();
```

```
// 페어링 된 장치가 있는 경우 각 장치를 ArrayAdapter에 추가한다.
if (pairedDevices.size() > 0) {
    findViewById(R.id.title_paired_devices).setVisibility(View.VISIBLE);
    for (BluetoothDevice device : pairedDevices) {
        pairedDevicesArrayAdapter.add(device.getName() + "\n" + device.getAddress());
    }
} else {
    String noDevices = "페어링 된 장치가 없습니다.".toString();
    pairedDevicesArrayAdapter.add(noDevices);
}
```

3교시

2020년도 1학기 앱 프로그래밍 – BLUETOOTH 검색 & 앱 구현 실습 및 과제

3.6 블루투스 기기 검색

- startDiscovery() 메서드
  - 비동기 검색 시작 → 검색 시작 상태 반환(bool)
  - 12초 가량의 조회 스캔
  - 검색된 각 기기의 페이지 스캔을 통한 블루투스 이름 획득
  - 애플리케이션이 검색된 각 기기에 대한 정보 수신
    - ACTION_FOUND 인텐트에 대한 BroadcastReceiver 등록 필요
      - 안드로이드 시스템의 브로드캐스트 인텐트 수신

```
// 장치 발견시 브로드 캐스트 리시버를 등록한다.
IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
registerReceiver(mReceiver, filter);
```

기기 검색 광고 수신기 등록

광고 수신기 해제

```
@Override
protected void onDestroy() {
    super.onDestroy();
    // 장치 검색을 수행하고 있지 않는지 확인 검색 중단
    if (mBtAdapter != null) {
        mBtAdapter.cancelDiscovery();
    }

    // 브로드캐스트 수신기 등록 해제
    this.unregisterReceiver(mReceiver);
}
```

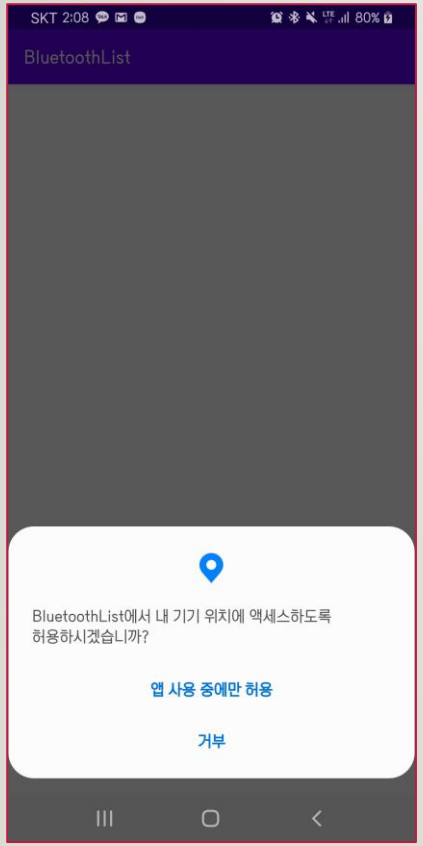
3.6 블루투스 기기 검색 (계속)

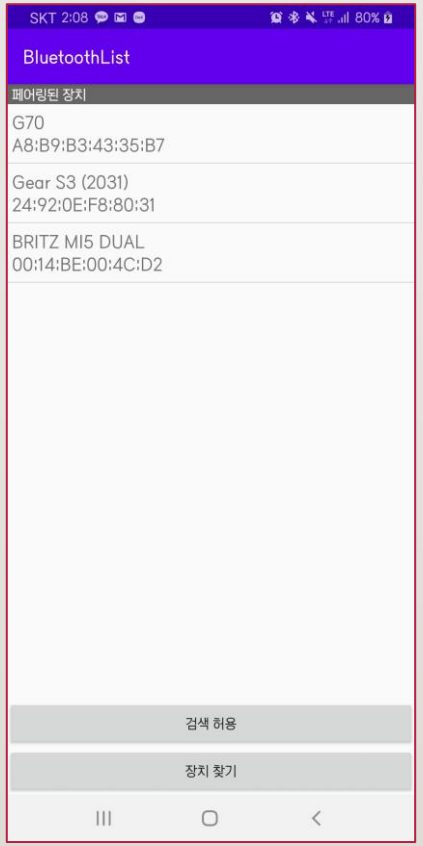
- BroadcastReceiver 인스턴스 생성
  - BluetoothDevice
    - intent 매개변수 정보 → getParcelableExtra() 메서드
      - 블루투스 장치 객체 획득
  - 메서드
    - getName() 메서드
      - String 타입 기기 이름 수집
    - getAddress() 메서드
      - String 타입 기기 MAC 주소 수집
  - action 상태 정보 확인
    - ACTION_FOUND
    - ACTION_DISCOVERY_FINISHED

```
// 장치 검색을 통해 장치 검색 광고를 수신하고 발견시 처리를 위한 BR 수신기
private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();

        // 기기 검색 상태 브로드캐스트 수신 시 처리
        if (BluetoothDevice.ACTION_FOUND.equals(action)) {
            // Intent에서 블루투스 장치 객체 가져오기
            BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
            // 검색 기기의 목록 추가 작업
            // 기기 검색 목록에 없는 경우 목록에 추가
            if (device.getBondState() != BluetoothDevice.BOND_BONDED) {
                mNewDevicesArrayAdapter.add(device.getName() + "\n" + device.getAddress());
            }
        }
        // 기기 검색이 완료 상태 브로드캐스트 수신 시 처리
        else if (BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(action)) {...}
    }
};
```

3. 구현 실습

- BluetoothList 실행화면





4. 앱 프로그래밍 학습을 위한 실습 (과제)

- 실습 및 예습 문제
  - 수업 시간에 설명한 블루투스 목록 표시 프로젝트 구현
    - 리스트뷰와 목록 표현 부분까지만 작성, 즉 프로젝트의 모형 부분까지 구성
    - 추가 설명 이후 기능 과제 부여 예정

- 제출 방법
  - http://ctl.gtec.ac.kr 의 과제 제출란에 제출
  - Java 파일과 xml 파일을 제출한다.
  - 실행 화면을 캡쳐하여 위 파일들과 함께 압축 파일하여 제출한다.
  - 압축파일은 자신의 "반_학번_이름.zip" 로 한다.
  - 제출 일자와 시간을 엄수하여 제출하세요!