모바일 시스템 프로그래밍

09 Mobile Sensing Pipeline 1

2017 1학기

모바일 센싱 파이프 라인..
파이프 라인?
어떤... 센싱하는 통로들?

강승우

Computer Science and Engineering, KOREATECH

Mobile Sensing System review

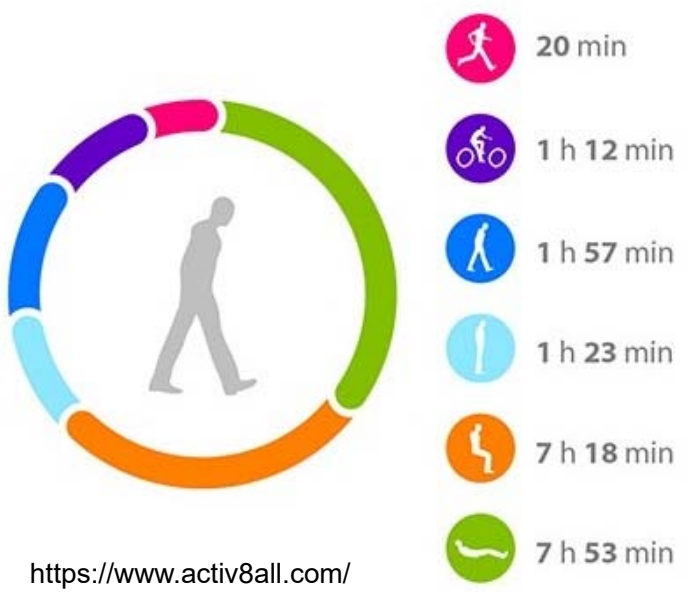
Mobile Phone Sensing

- 스마트폰에 내장된 다양한 센서를 이용하여 각종 센싱 애플리케이션을 구동할 수 있음
  - fitness, pedometer (활동량, 만보계): Google Fit, Accupedo, S Health
  - activity/location tracking (활동량, 장소): Moves
  - transportation mode (버스/자전거/지하철/도보) transportation mode? 무슨 모드지?
  - environment/place (주변 환경/장소 정보)
  - conversation (대화 상대/대화 상태)
  - emotion (감정 상태)

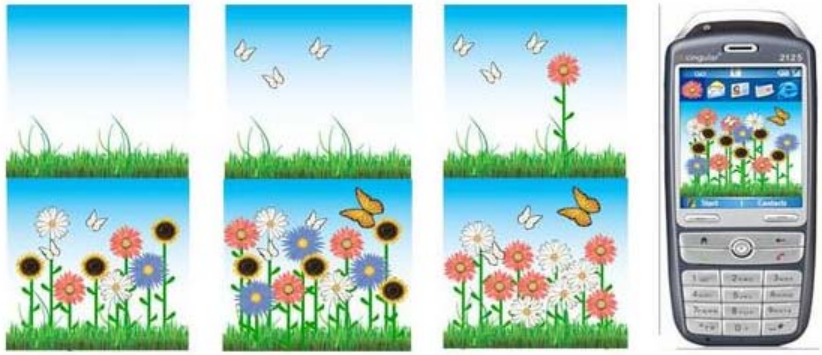
- 사람들의 다양한 활동, 상태, 상황 정보를 인지/추론/예측하여 서비스를 제공할 수 있음

Physical Activity

- activity inference 예제
  - 걷기, 달리기, 싸이클링, 계단 오르내리기, …
- 사용 센서 예
  - accelerometer, gyroscope, compass
- 애플리케이션
  - 활동량 로그
  - Health/behavior intervention: 운동 장려



https://www.activ8all.com/



Consolvo, Sunny, et al. "Flowers or a robot army?: encouraging awareness & activity with personal, mobile displays." *Proceedings of the 10th international conference on Ubiquitous computing*. ACM, 2008.

Transportation Mode

- mode inference 예제
  - 버스, 자동차, 지하철, 자전거, …
- 사용 센서 예
  - accelerometer, gyroscope
  - GPS, WiFi, Cell
- 애플리케이션
  - 지능형 대중 교통 시스템

http://www.123rf.com/clipart-vector/mode_of_transportation.html

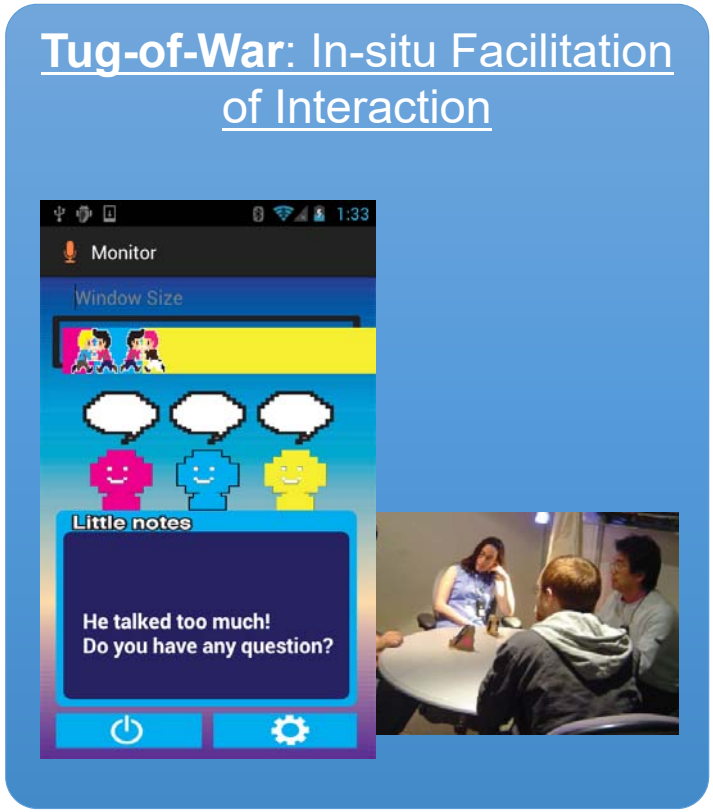
Place/Environment Context

- context inference 예제
  - 대화, 음악, 파티 장소, 운전 중, …
- 사용 센서 예
  - microphone, camera
  - accelerometer, GPS, WiFi, Cell
- 애플리케이션
  - automated diary
  - place tagging
  - health/wellness

https://www.technologyreview.com/s/413958/cell-phones-that-listen-and-learn/

Conversation, Human Voice

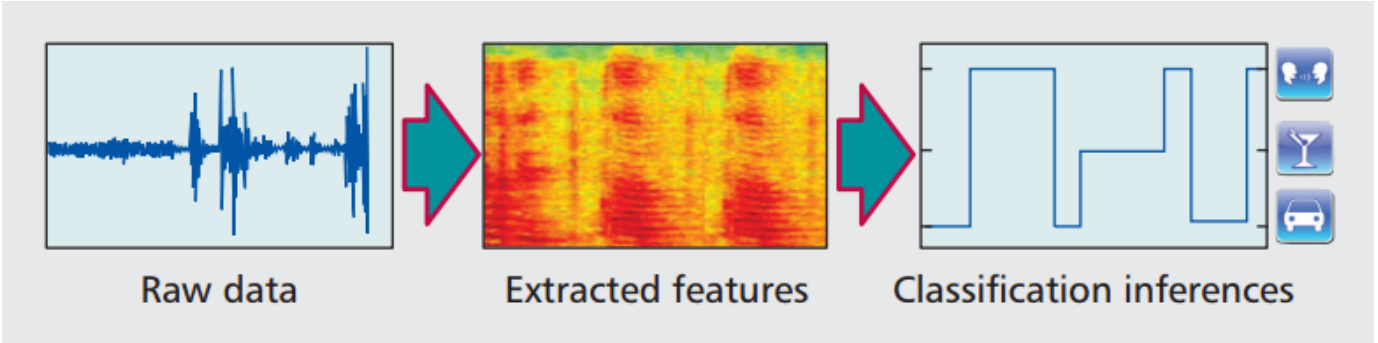
- 예제
  - turn-taking, stress, speaker Id.
- 사용 센서 예
  - microphone
- 애플리케이션
  - 소셜 네트워크 분석
  - 대화 상대, 대화 참여도
  - 스트레스 정도



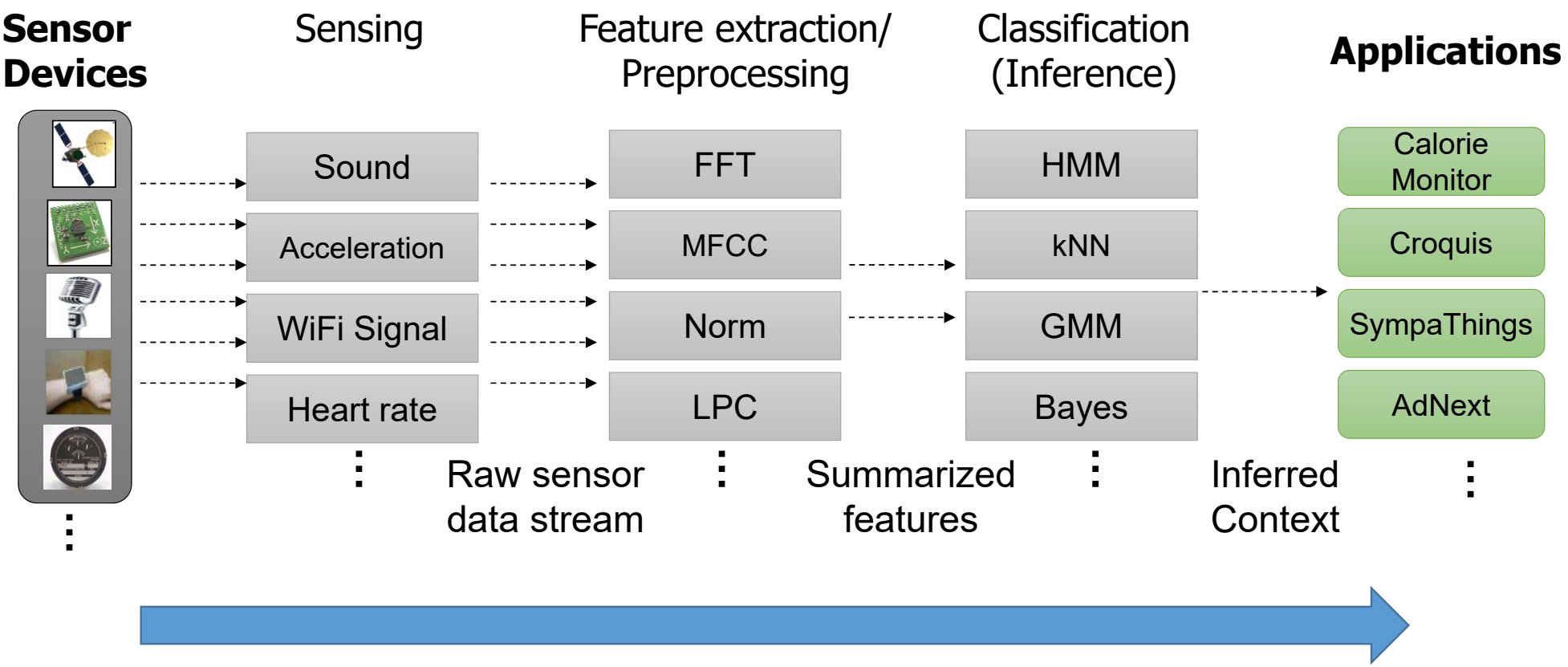
Lee, Youngki, et al. "Sociophone: Everyday face-to-face interaction monitoring platform using multi-phone sensor fusion." *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. ACM, 2013.

Common Mobile Sensing Design Pattern

- 일반적으로 다음과 같은 작업을 공통으로 수행
  - Collect raw data using the sensors (mobile phone, other wearable sensors)
  - Infer a context (state, situation) of a user using the raw sensor data
    - physical activity: running, waking, driving?
    - conversation
  - Provide the inferred result to the user or use the result to adapt the service/information provided by the application

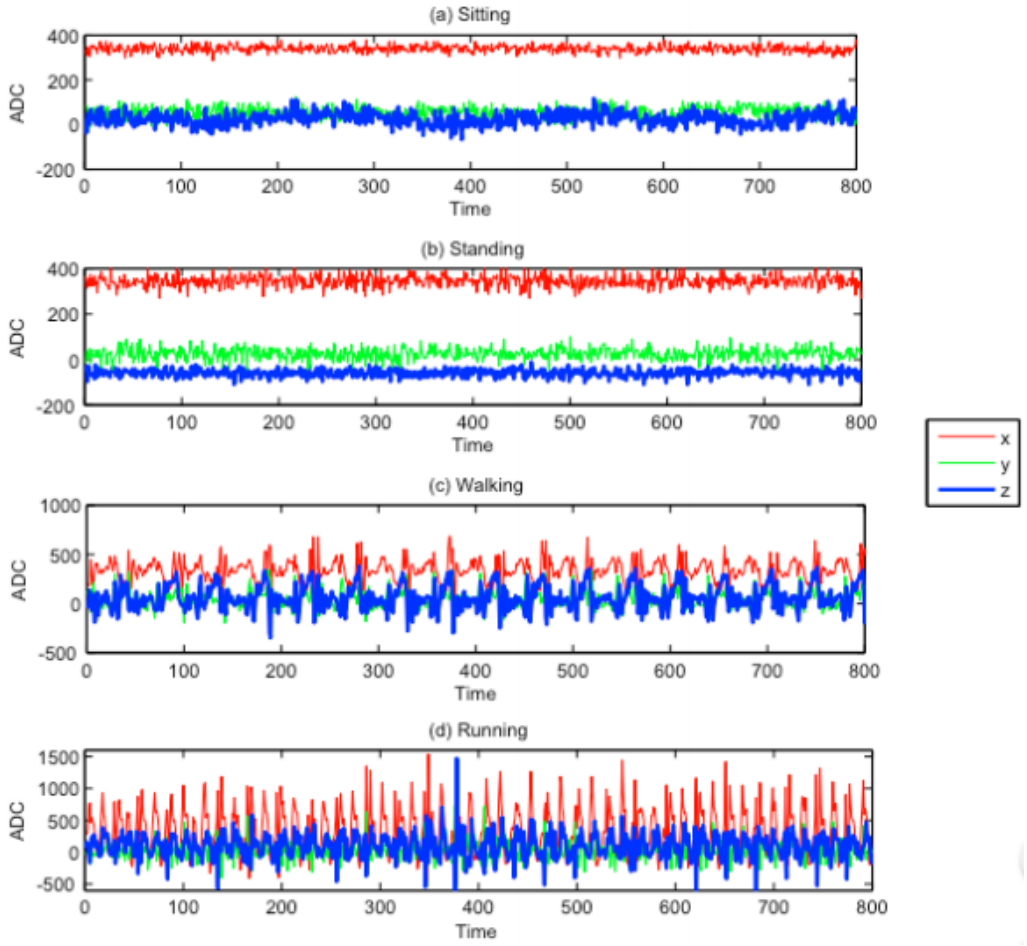


Mobile Sensing Pipeline



Physical Activity Example

- Activities
  - sitting, standing, walking, running
- Sensor
  - accelerometer
- Features
  - Mean
  - Standard deviation
  - Number of peaks
  - FFT-derived features (spectrum peak, sub-band energy, spectral entropy, …)
- Classification
  - Decision Trees
  - Hidden Markov Models
  - Naïve Bayes



Activity에 따른 가속도 센서 데이터의 변화

What to learn

- 모바일 센싱 기능 구현을 위한 안드로이드 Application Framework의 각종 시스템 서비스 API 활용
  - SensorManager
  - LocationManager
  - WiFiManager
  - BluetoothManager
  - PowerManager

지금까지 주로 학습한 내용

- 모바일 센싱 애플리케이션 구현

- 효율적인 모바일 센싱 시스템 구현을 위한 디자인 방법

Example cases for Mobile Sensing Pipeline

- 두 가지 간단한 사례를 가지고 살펴보자
  - Accelerometer 기반 활동량 모니터링 (Step Monitor)
  - GPS, WiFi를 이용한 위치 모니터링 (Location Tracker)

SensorManager review

안드로이드 센서 사용 및 관리

- 관련 주요 class 및 interface
  - SensorManager
  - Sensor
  - SensorEvent
  - SensorEventListener

- 센서 관련 API의 이용
  - 사용 가능한 센서 식별, 센서의 특성/기능 파악
  - 센서 이벤트 모니터링
    - Sensor name, timestamp (이벤트 발생 시간), accuracy, raw sensor data

https://developer.android.com/guide/topics/sensors/sensors_overview.html?hl=ko

SensorManager 클래스

- 안드로이드 플랫폼에서 센서를 관리하고 이용할 수 있도록 제공되는 시스템 서비스
  - 센서에 대한 정보를 제공하고, 특정 타입의 센서를 액세스 할 수 있게 해줌
  - 센서 이벤트 리스너 등록을 통해 센서 데이터를 받을 수 있음

  사용 예:

  ```
  SensorManager sm =
  (SensorManager)getSystemService(Context.SENSOR_SERVICE);
  ```

https://developer.android.com/reference/android/hardware/SensorManager.html?hl=ko

Sensor 클래스

- SensorManager 클래스의 메소드를 이용하여 사용하고자 하는 센서 객체를 얻을 수 있음
  - getDefaultSensor(int type) 메소드 이용

  사용 예: Sensor accel = sm.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);

- Sensor 클래스 메소드

  float getMaximumRange()
  센서 값의 최대 범위

  int getMinDelay()
  두 개의 센서 이벤트 사이의 최소 딜레이 (ms 단위) 혹은 0 (측정 데이터에 변화가 있을 때만 값을 주는 경우

  String getName()

  float getPower()
  사용 중에 소모한 파워 (mA 단위)

  String getStringType()

  int getType()

  String getVendor()

  int getVersion()

https://developer.android.com/reference/android/hardware/Sensor.html?hl=ko

SensorEventListener 인터페이스

- 센서 값이 변경됐을 때 SensorManager로부터 이벤트 형태로 전달 받을 수 있도록 해줌
- SensorEventListener 등록
  - SensorManager 클래스에 정의된 메소드 이용

  boolean registerListener(SensorEventListener listener, Sensor sensor, int samplingPeriodUs)
    - listener: SensorEventListener 객체
    - sensor: 데이터를 받고 싶은 센서 객체 (앞에서 언급한 것처럼 SensorManager의 getDefaultSensor 메소드를 통해 얻음)
    - samplingPeriodUs: 데이터를 전달 받고자 하는 속도. SensorManager 클래스에 다음 상수가 정의되어 있음. 일정한 속도를 보장하지는 않음. 기기마다 다를 수 있음

      SENSOR_DELAY_NORMAL

      SENSOR_DELAY_UI

      SENSOR_DELAY_GAME

      SENSOR_DELAY_FASTEST

SensorEventListener 인터페이스

- SensorEventListener 해제

  public void unregisterListener(SensorEventListener listener, Sensor sensor)

  public void unregisterListener(SensorEventListener listener)

  - 센서 데이터 업데이트가 더 이상 필요하지 않을 때는 반드시 해제를 해주어야 함

- SensorEventListener 인터페이스 구현

  - 아래 두 개의 메소드를 구현해야 함

  public void onAccuracyChanged(Sensor sensor, int accuracy)

  ✓등록된 센서의 정확도가 변경됐을 때 호출

  public void onSensorChanged(SensorEvent event)

  ✓센서 값이 변했을 때 호출. SensorEvent 객체로 데이터를 전달 받게 됨

https://developer.android.com/reference/android/hardware/SensorEventListener.html?hl=ko

SensorEvent 클래스

- 센서로부터 센서 데이터가 업데이트 되었다는 이벤트를 나타냄
- 센서 타입, 시간(timestamp), 정확도, 센서 데이터와 같은 정보를 담고 있음

public int accuracy
- ✓이 이벤트의 정확도

public Sensor sensor
- ✓이 이벤트를 발생한 센서

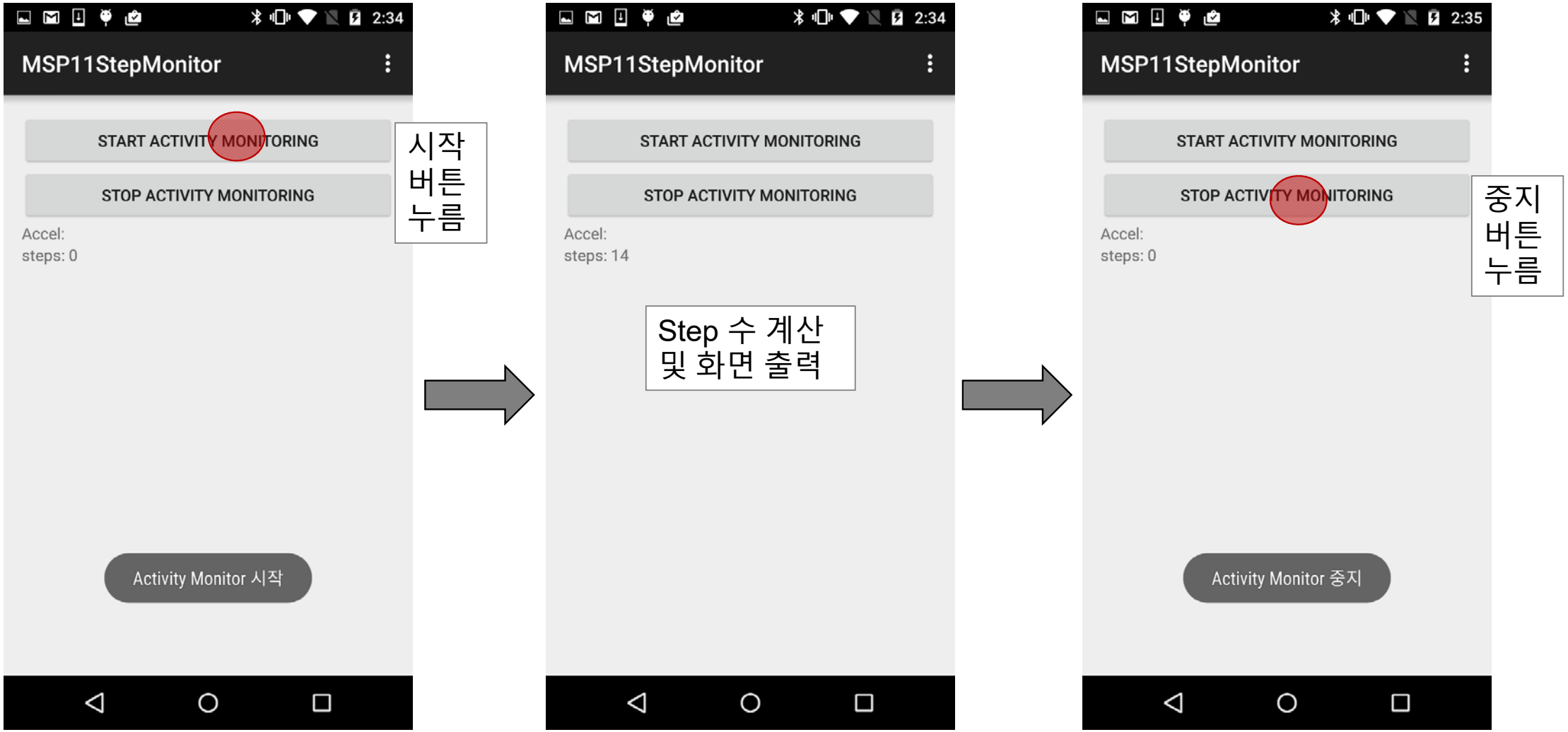
public long timestamp
- ✓이벤트가 발생한 시각 (나노초 단위)

public final float[] values
- ✓센서 데이터를 담고 있는 배열. 센서 타입에 따라 길이나 그 내용은 달라짐

https://developer.android.com/reference/android/hardware/SensorEvent.html?hl=ko

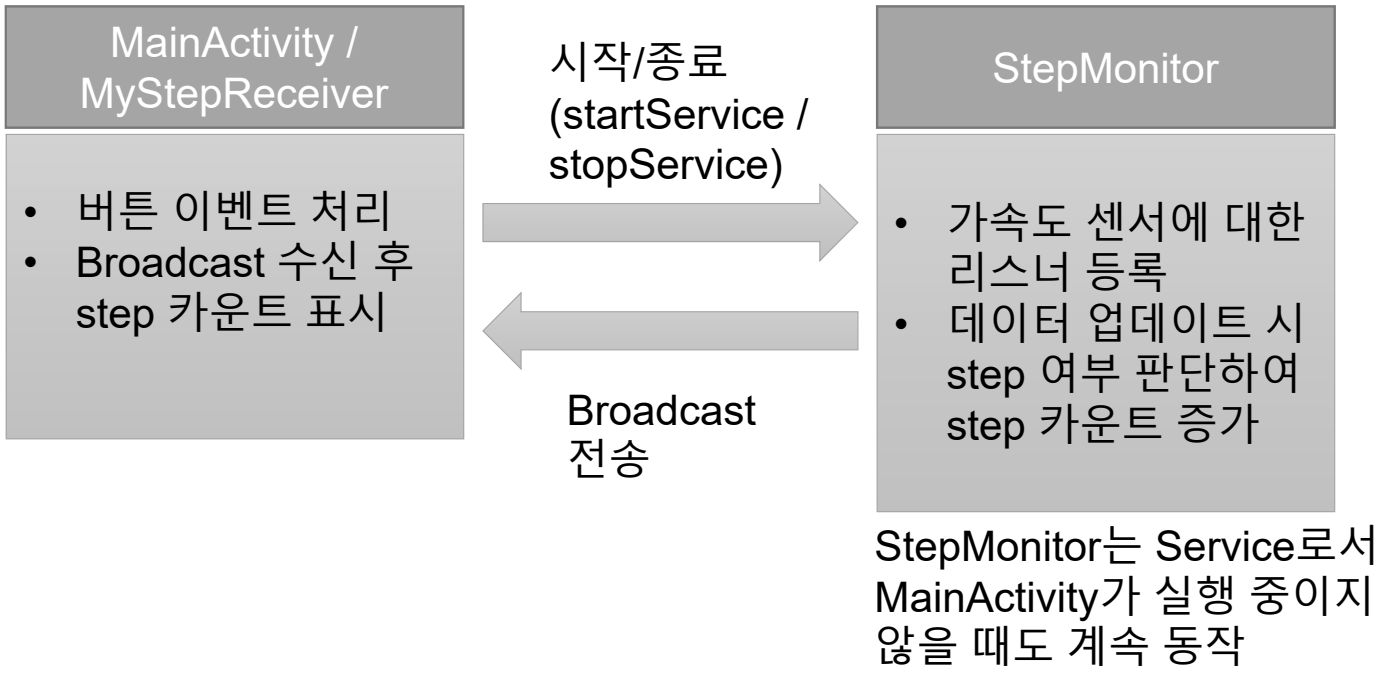
Accelerometer를 이용한 활동량 모니터링 (Step Monitor)

Step Monitor (예제 프로젝트 이름: MSP11StepMonitor)



예제: Step Monitor

- 코드 구조



Simple sensing pipeline

- StepMonitor에서 사용된 가속도 센서 데이터 처리 과정
  - Sensor data collection
    - 3축 가속도 센서 데이터 중 y축 값을 수집
  - Feature extraction
    - 시간에 따른 센서 값의 변화를 계산
      - 이전 가속도 센서 데이터 y축 값과 현재 y축 값의 차이 절대값
  - Classification
    - Step 유무 판단
      - 센서 값의 변화가 일정 threshold보다 크면 step이 있었다고 판단
      - 작으면 없었다고 판단
    - Step이 있었다고 판단되면 step count 증가

Code snippet (MainActivity.java)

```
public class MainActivity extends AppCompatActivity {
    private TextView stepsText;
    private int steps;

    private BroadcastReceiver MyStepReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            if(intent.getAction().equals("kr.ac.koreatech.msp.stepmonitor")) {
                steps = intent.getIntExtra("steps", 0);
                stepsText.setText("steps: " + steps);
            }
        }
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        stepsText = (TextView)findViewById(R.id.count);
    }
```

StepMonitor Service로부터 step 값을 담은 Broadcast를 수신하기 위한 Broadcast Receiver

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

    IntentFilter intentFilter = new IntentFilter("kr.ac.koreatech.msp.stepmonitor");
    registerReceiver(MyStepReceiver, intentFilter);

}

@Override
protected void onPause() {
    super.onPause();
    unregisterReceiver(MyStepReceiver);
}
```

Broadcast를 수신하기 위한 Broadcast Receiver 등록

```
// Start/Stop 버튼을 눌렀을 때 호출되는 콜백 메소드
// Activity monitoring을 수행하는 service 시작/종료
public void onClick(View v) {
    if(v.getId() == R.id.startMonitor) {
        Intent intent = new Intent(this, StepMonitor.class);
        startService(intent);
    } else if(v.getId() == R.id.stopMonitor) {
        stopService(new Intent(this, StepMonitor.class));
        stepsText.setText("steps: " + 0);
    }
}
```

StepMonitor Service를 시작/종료하기 위한 버튼 이벤트 처리 메소드

Code snippet (StepMonitor.java)

```
public class StepMonitor extends Service implements SensorEventListener {
    private SensorManager mSensorManger;
    private Sensor mAccel;
    private float previousY, currentY;
    private int threshold, steps;

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public void onCreate() {
        // step 여부를 판단하기 위한 y축 가속도 값의 차이의 문턱값
        threshold = 10;
        previousY = currentY = steps = 0;

        mSensorManger = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        mAccel = mSensorManger.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
        // SensorEventListener 등록
        mSensorManger.registerListener(this, mAccel, SensorManager.SENSOR_DELAY_NORMAL);
    }
```

가속도 센서를 받기 위해
SensorManager 객체를 이용하여
SensorEventListener 등록

```
public void onDestroy() {
    Toast.makeText(this, "Activity Monitor 중지", Toast.LENGTH_SHORT).show();
    Log.d(TAG, "onDestroy()");

    // SensorEventListener 해제
      mSensorManger.unregisterListener(this);

}
```

SensorEventListener 구현을 위한 필수 메소드

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

// 센서 데이터가 업데이트 되면 호출
public void onSensorChanged(SensorEvent event) {
    if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
        //***** sensor data collection *****//
        // 현재 y축 가속도 값
        currentY = event.values[1];

        // simple step calculation
        computeSteps(previousY, currentY);

        // 현재 y축 가속도 값을 이전 y축 가속도 값으로 기억해 둠
        previousY = event.values[1];
    }
}
```

가속도 센서 데이터를 받고 처리하는 코드

가속도 센서 데이터를 이용하여 step 수를 계산하는 함수
(y축 가속도 값의 차이를 이용. Step 수가 증가하는 경우 Broadcast 전송)

```
// a simple inference for step count
// 이전 y축 가속도 값과 현재 y축 가속도 값의 차가 threshold보다 크면 걸음 수 증가
private void computeSteps(float prev, float curr) {
    //***** feature extraction *****//
    // calculate feature data:
    // 여기서는 이전 y축 가속도 값과 현재 y축 가속도 값의 차이를 이용
    float feature = Math.abs(curr - prev);

    //***** classification *****//
    // check if there is a step or not:
    // 여기서는 y축 가속도 값 차이가 일정 문턱값 이상이면 step으로 판단
    if(feature > threshold) {
        steps++;

        // if steps increased, send steps data to MainActivity
        Intent intent = new Intent("kr.ac.koreatech.msp.stepmonitor");
        intent.putExtra("steps", steps);
        sendBroadcast(intent);
    }
}
```

생각해볼 문제

- 센서 데이터 중 y축 값만 보는 것으로 충분한가?
  - 현재의 문제점:
- 현재 값과 바로 이전 값의 차이를 보는 것으로 step을 구분하는 것이 적당한가?
  - 현재의 문제점:
- Threshold를 어떻게 정하는가?
  - 현재의 문제점:

실습

- 예제 프로그램을 테스트 해보면서 step 수 계산이 잘 이루어지는지 확인해보자
  - 사용자로부터 threshold 값을 입력 받아서 다양한 값을 가지고 테스트 해 볼 수 있도록 만들어보자
    - MainActivity의 UI layout에 사용자로부터 데이터를 입력 받을 수 있는 EditText 추가
    - 값이 입력된 상태에서 Start activity monitoring 버튼을 누르면 그 값을 threshold 값으로 StepMonitor Service로 전달
    - StepMonitor Service에서는 전달 받은 값을 threshold 값으로 이용하여 step 수 계산
      - 현재는 소스 코드 상에 값이 하드 코딩 되어 있음
  - 앞의 생각해 볼 문제의 질문에 대해서 생각해보자