CS355

Mobile Application Development
การพัฒนาโปรแกรมประยุกต์สำหรับอุปกรณ์พกพา

Pakorn Leesutthipornchai, Ph.D.
Assistant Professor
ผศ.ดร.ปกรณ์ ลี้สุทธิพรชัย
pakornl@cs.tu.ac.th

COMPUTER SCIENCE
THAMMASAT UNIVERSITY
www.cs.tu.ac.th

MA12: Get Information from Sensors
การอ่านค่าจากเซนเซอร์

Outline

- Location by using GPS: Latitude, Longitude เช่น 14.0771, 100.5935
- Accelerometer: ความเร่งเชิงเส้นตามแกน x, y, z มีหน่วยเป็น m/s^2
- Ambient Temperature --- อุณหภูมิสภาพแวดล้อมรอบเครื่อง มีหน่วยเป็น °C
- Gravity: ค่าแรง g ตามแนวแกน x, y, z มีหน่วยเป็น m/s^2
- Gyroscope: ความเร็วเชิงมุมตามแกน x, y, z มีหน่วยเป็น rad/s
- Light: มีหน่วยเป็น lux เช่น 40.0 lux
- Linear Acceleration: ความเร่งเชิงเส้นไม่รวมแรง g
- Magnetic Field: x, y, z มีหน่วยเป็น Micro Tesla
- Orientation: ทิศทาง x, y, z มีหน่วยเป็น °
- Pressure: มีหน่วยเป็น mb, mmHg, atm, Pascal
- Proximity: วัดความใกล้ มีหน่วยเป็น cm เช่น 8.0 cm
- Relative Humidity --- ความชื้นสัมพัทธ์ มีหน่วยเป็น %
- Rotation Vector --- เวกเตอร์การหมุน
- Temperature --- อุณหภูมิเครื่อง มีหน่วยเป็น °C
- Battery: Temperature, Level, Voltage

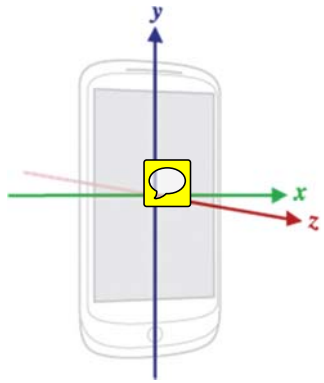
Get Information from Sensor

Table 2. Sensor availability by platform.

Sensor	Android 4.0 (API Level 14)	Android 2.3 (API Level 9)	Android 2.2 (API Level 8)	Android 1.5 (API Level 3)
TYPE_ACCELEROMETER	Yes	Yes	Yes	Yes
TYPE_AMBIENT_TEMPERATURE	Yes	n/a	n/a	n/a
TYPE_GRAVITY	Yes	Yes	n/a	n/a
TYPE_GYROSCOPE	Yes	Yes	n/a	n/a
TYPE_LIGHT	Yes	Yes	Yes	Yes
TYPE_LINEAR_ACCELERATION	Yes	Yes	n/a	n/a
TYPE_MAGNETIC_FIELD	Yes	Yes	Yes	Yes
TYPE_ORIENTATION	Yes	Yes	Yes	Yes
TYPE_PRESSURE	Yes	Yes	n/a	n/a
TYPE_PROXIMITY	Yes	Yes	Yes	Yes
TYPE_RELATIVE_HUMIDITY	Yes	n/a	n/a	n/a
TYPE_ROTATION_VECTOR	Yes	Yes	n/a	n/a
TYPE_TEMPERATURE	Yes	Yes	Yes	Yes

 http://developer.android.com/guide/topics/sensors/sensors_overview.html

Sensors on Android



This coordinate system is different from the one used in the Android 2D APIs where the origin is in the top-left corner.

 http://developer.android.com/reference/android/hardware/SensorEvent.html

Layout

GPS
- ✓LocationManager
- ✓LocationListener

Sensors
- ✓SensorManager

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-feature android:name="android.hardware.location.gps" />
```


Location by using GPS (1/3)

```
1.  public class MainActivity extends AppCompatActivity implements SensorEventListener {
2.     private static final long MINIMUM_DISTANCE_CHANGE_FOR_UPDATES = 1; // in Meters
3.     private static final long MINIMUM_TIME_BETWEEN_UPDATES = 1000; // in Milliseconds
4.     LocationManager locationManager;
5.     LocationListener locationListener = new LocationListener() {
6.        public void onLocationChanged(Location location) {
7.           String message = String.format("New Location \n Longitude: %1$s \n Latitude: %2$s",
8.                 location.getLongitude(), location.getLatitude() );
9.           TextView text=(TextView)findViewById(R.id.textView);
10.          text.setText(message);
11.       }
12.       ...
13.       public void onProviderDisabled(String s) {
14.          String message = "Provider disabled by the user. GPS turned off";
15.          TextView text=(TextView)findViewById(R.id.textView);
16.          text.setText(message);
17.       }//end onProviderDisabled

18.       public void onProviderEnabled(String s) {
19.          String message = "Provider enabled by the user. GPS turned on";
20.          TextView text=(TextView)findViewById(R.id.textView);
21.          text.setText(message);
22.       }//end onProviderEnabled
23.    };
```


Location by using GPS (2/3)

```
1.  protected void onCreate(Bundle savedInstanceState) {
2.
3.     locationManager = (LocationManager) getSystemService( Context.LOCATION_SERVICE );
4.     if (locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)){
5.        Toast.makeText(this, "GPS is Enabled in your device", Toast.LENGTH_SHORT).show();
6.     }else{
7.        showGPSDisabledAlertToUser();
8.     }
9.     if (locationManager != null) {
10.       if ( Build.VERSION.SDK_INT >= 23 &&
11.          checkSelfPermission(ACCESS_FINE_LOCATION) != PERMISSION_GRANTED &&
12.          checkSelfPermission(ACCESS_COARSE_LOCATION) != PERMISSION_GRANTED ) {
13.             locationManager.removeUpdates( locationListener );
14.       }
15.       locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
16.                                              MINIMUM_TIME_BETWEEN_UPDATES,
17.                                              MINIMUM_DISTANCE_CHANGE_FOR_UPDATES,
18.                                              locationListener);
19.    }
20.    ...
21.    button1.setOnClickListener(new View.OnClickListener() {
22.       public void onClick(View v) {
23.          showCurrentLocation();
24.       }
25.    });
26.    ...
27.    button2.setOnClickListener(new View.OnClickListener() {
28.       public void onClick(View v) {
29.          stopLocation();
30.       }
31.    });
```

Handle Uncertainty Task

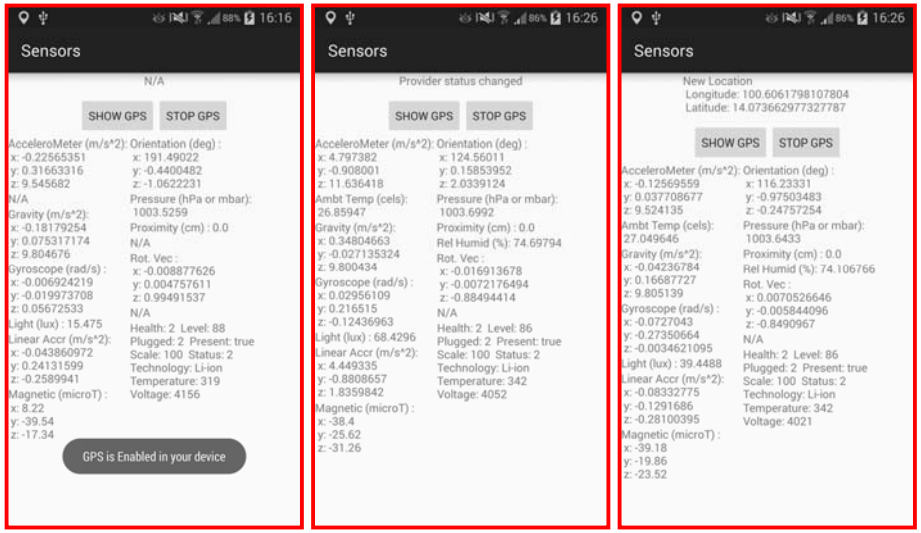
Location by using GPS (3/3)

```
1.  protected void showCurrentLocation() {
2.     if (locationManager != null) {
3.        if ( Build.VERSION.SDK_INT >= 23 &&
4.           checkSelfPermission(ACCESS_FINE_LOCATION) != PERMISSION_GRANTED &&
5.           checkSelfPermission(ACCESS_COARSE_LOCATION) != PERMISSION_GRANTED ) {
6.              locationManager.removeUpdates(locationListener);
7.        }
8.        Location location =
9.                 locationManager.getLastKnownLocation( LocationManager.GPS_PROVIDER );
10.       if (location != null) {
11.          String message = String.format("Current Location \n Longitude: %1$s \n
12.                                          Latitude: 2$s", location.getLongitude(), location.getLatitude());
13.          TextView text=(TextView)findViewById(R.id.textView);
14.          text.setText(message);
15.       }
16.    }
17. }

18. protected void stopLocation() {
19.    if (locationManager != null) {
20.       if ( Build.VERSION.SDK_INT >= 23 &&
21.          checkSelfPermission(ACCESS_FINE_LOCATION) != PERMISSION_GRANTED &&
22.          checkSelfPermission(ACCESS_COARSE_LOCATION) != PERMISSION_GRANTED ) {
23.             locationManager.removeUpdates(locationListener);
24.       }
25.       locationManager.removeUpdates(locationListener);
26.    }
27. }
```

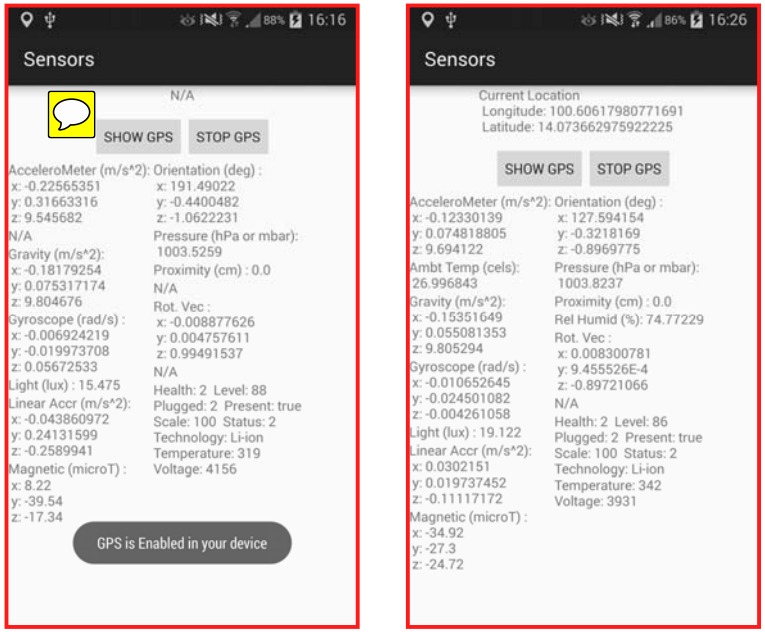
Location Listener

- **Location Listener** ประกอบด้วย 4 methods สำคัญ คือ
  - **onLocationChanged()**
  - **onStatusChanged()**
    - OUT_OF_SERVICE
      if the provider is out of service, and this is not expected to change in the near future
    - TEMPORARILY_UNAVAILABLE
      if the provider is temporarily unavailable but is expected to be available shortly
    - AVAILABLE
      if the provider is currently available
  - **onProviderDisabled()**
  - **onProviderEnabled()**

Location by using GPS : Run on Mobile Device (1/2)



Location by using GPS : Run on Mobile Device (2/2)



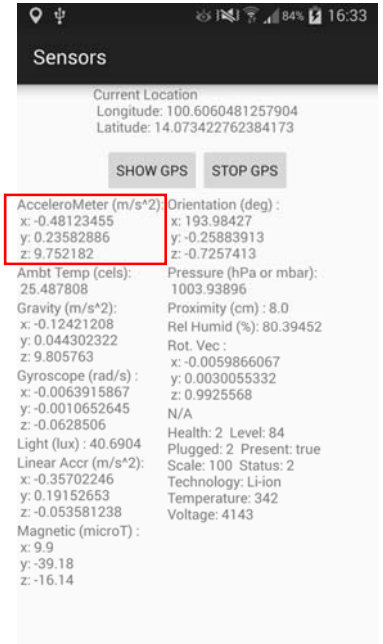
Sensors

```
protected void onCreate(Bundle savedInstanceState) {
        sensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
        sensorManager.registerListener(this,
                        sensorManager.getDefaultSensor( Sensor.TYPE_* ),
                        SensorManager.SENSOR_DELAY_NORMAL);
}

public void onSensorChanged(SensorEvent event){
   int type = event.sensor.getType();
   float x, y, z;

   if(type == Sensor.TYPE_* ){
      x=event.values[0];
      y=event.values[1];
      z=event.values[2];
      String message = String.format("AcceleroMeter (m/s^2): \n
                                    x: %1$s \n y: %2$s \n z: %3$s", x, y, z);
      TextView text=(TextView)findViewById(R.id.textView1);
      text.setText(message);
   }

}
```

Delay

Delay

- SENSOR_DELAY_FASTEST
  get sensor data as fast as possible
- SENSOR_DELAY_GAME
  rate suitable for games
- SENSOR_DELAY_NORMAL
  rate (default) suitable for screen orientation changes



Accelerometer

- Sensor.TYPE_ACCELEROMETER
- มีหน่วยเป็น เมตร/วินาที^{2}
  - values[0] : Acceleration force along the x axis
  - values[1] : Acceleration force along the y axis
  - values[2] : Acceleration force along the z axis

```
public class MainActivity extends Activity implements SensorEventListener {
    SensorManager sensorManager;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sensorManager=(SensorManager)getSystemService(SENSOR_SERVICE);
        sensorManager.registerListener(this,
            sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
            SensorManager.SENSOR_DELAY_NORMAL);
    }
```


Accelerometer (2)

```
    public void onSensorChanged(SensorEvent event){
        int type = event.sensor.getType();
        float x, y, z;

        if(type==Sensor.TYPE_ACCELEROMETER){
            x=event.values[0];
            y=event.values[1];
            z=event.values[2];
            String message = String.format("AcceleroMeter (m/s^2): \n
                x: %1$s \n y: %2$s \n z: %3$s \n", x, y, z);
            TextView text=(TextView)findViewById(R.id.textView1);
            text.setText(message);
        }
    }
}
```

acceleration = gravity + linear-acceleration

$g = 9.81\ m/s^2$

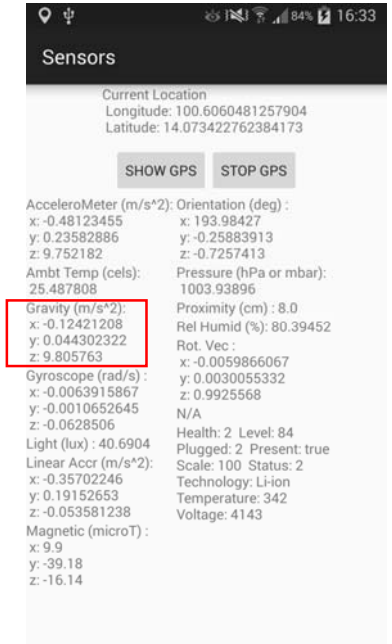
Ambient Temperature

- Sensor.TYPE_AMBIENT_TEMPERATURE
- มีหน่วยเป็น เซลเซียส (Celsius)
  - values[0] ค่าอุณหภูมิ

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    sensorManager=(SensorManager)getSystemService(SENSOR_SERVICE);
    sensorManager.registerListener(this,
        sensorManager.getDefaultSensor(Sensor.TYPE_AMBIENT_TEMPERATURE),
        SensorManager.SENSOR_DELAY_NORMAL);
}
public void onSensorChanged(SensorEvent event){
    int type = event.sensor.getType();
    float x, y, z;
    if(type==Sensor.TYPE_AMBIENT_TEMPERATURE){
        x=event.values[0];
        String message = String.format("Ambt Temp (cels): \n %1$s \n", x);
        TextView text=(TextView)findViewById(R.id.textView2);
        text.setText(message);
    }
}
```

Gravity

- Sensor.TYPE_GRAVITY
- มีหน่วยเป็น เมตร/วินาที^{2}
  - SensorEvent.values[0]
    Force of gravity along the x axis.
  - SensorEvent.values[1]
    Force of gravity along the y axis.
  - SensorEvent.values[2]
    Force of gravity along the z axis.

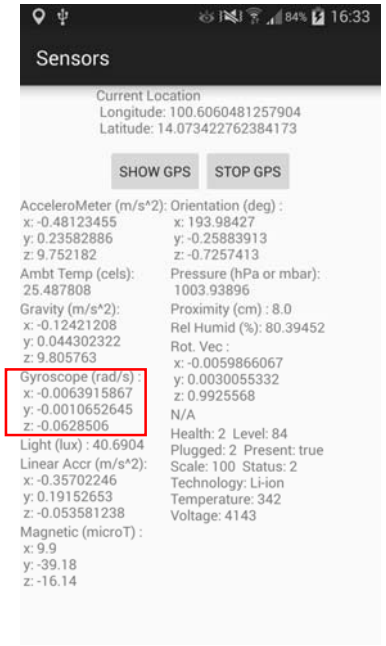


Gravity (2)

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sensorManager=(SensorManager)getSystemService(SENSOR_SERVICE);
        sensorManager.registerListener(this,
                        sensorManager.getDefaultSensor(Sensor.TYPE_GRAVITY),
                        SensorManager.SENSOR_DELAY_NORMAL);
}

public void onSensorChanged(SensorEvent event){
        int type = event.sensor.getType();
        float x, y, z;

        if(type==Sensor.TYPE_GRAVITY){
                x=event.values[0];
                y=event.values[1];
                z=event.values[2];
                String message = String.format("Gravity (m/s^2): \n
                                x: %1$s \n y: %2$s \n z: %3$s", x, y, z);
                TextView text=(TextView)findViewById(R.id.textView3);
                text.setText(message);
        }
}
```


Gyroscope

- Sensor.TYPE_GYROSCOPE
- มีหน่วยเป็น radians/second
- Rotation is **positive in the counter-clockwise** direction
  - values[0]: Angular speed around the x-axis
  - values[1]: Angular speed around the y-axis
  - values[2]: Angular speed around the z-axis
- Typically the output of the gyroscope is integrated over time to calculate a rotation.



Gyroscope (2)

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sensorManager=(SensorManager)getSystemService(SENSOR_SERVICE);
        sensorManager.registerListener(this,
                        sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE),
                        SensorManager.SENSOR_DELAY_NORMAL);
}

public void onSensorChanged(SensorEvent event){
        int type = event.sensor.getType();
        float x, y, z;

        if(type==Sensor.TYPE_GYROSCOPE){
                x=event.values[0];
                y=event.values[1];
                z=event.values[2];
                String message = String.format("Gyroscope (rad/s) : \n
                                x: %1$s \n y: %2$s \n z: %3$s", x, y, z);
                TextView text=(TextView)findViewById(R.id.textView4);
                text.setText(message);
        }
}
```

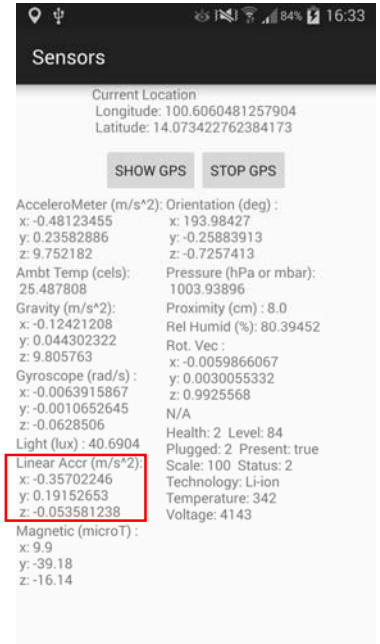
Light

- Sensor.TYPE_LIGHT
- มีหน่วยเป็น lux
- values[0]: Ambient light level

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sensorManager=(SensorManager)getSystemService(SENSOR_SERVICE);
        sensorManager.registerListener(this,
                                sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT),
                                SensorManager.SENSOR_DELAY_NORMAL);
}
public void onSensorChanged(SensorEvent event){
        int type = event.sensor.getType();
        float x, y, z;
        if(type==Sensor.TYPE_LIGHT){
                x=event.values[0];
                String message = String.format("Light (lux) : %1$s", x);
                TextView text=(TextView)findViewById(R.id.textView5);
                text.setText(message);
        }
}
```


Linear Acceleration

- Sensor.TYPE_LINEAR_ACCELERATION
- มีหน่วยเป็น เมตร/วินาที^{2}
- SensorEvent.values[0]
  Acceleration force along the x axis (excluding gravity)
- SensorEvent.values[1]
  Acceleration force along the y axis (excluding gravity)
- SensorEvent.values[2]
  Acceleration force along the z axis (excluding gravity)



Linear Acceleration (2)

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sensorManager=(SensorManager)getSystemService(SENSOR_SERVICE);
        sensorManager.registerListener(this,
                        sensorManager.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION),
                        SensorManager.SENSOR_DELAY_NORMAL);
}

public void onSensorChanged(SensorEvent event){
        int type = event.sensor.getType();
        float x, y, z;

        if(type==Sensor.TYPE_LINEAR_ACCELERATION){
                x=event.values[0];
                y=event.values[1];
                z=event.values[2];
                String message = String.format("Linear Accr (m/s^2): \n
                                x: %1$s \n y: %2$s \n z: %3$s", x, y, z);
                TextView text=(TextView)findViewById(R.id.textView6);
                text.setText(message);
        }
}
```


Magnetic Field

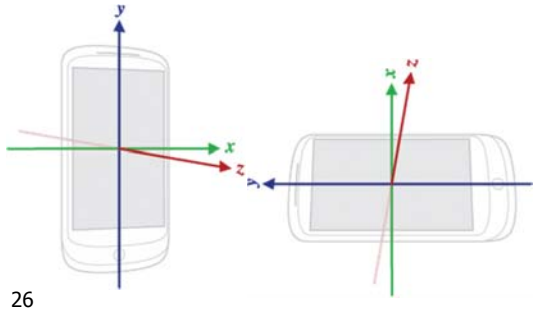
- Sensor.TYPE_MAGNETIC_FIELD
- Measure the ambient magnetic field in the X, Y and Z axis
- มีหน่วยเป็น micro-Tesla (uT)
- SensorEvent.values[0]
  Geomagnetic field strength along the x axis.
- SensorEvent.values[1]
  Geomagnetic field strength along the y axis.
- SensorEvent.values[2]
  Geomagnetic field strength along the z axis.

Magnetic Field (2)

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sensorManager=(SensorManager)getSystemService(SENSOR_SERVICE);
        sensorManager.registerListener(this,
                        sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD),
                        SensorManager.SENSOR_DELAY_NORMAL);
}

public void onSensorChanged(SensorEvent event){
        int type = event.sensor.getType();
        float x, y, z;

        if(type==Sensor.TYPE_MAGNETIC_FIELD){
                x=event.values[0];
                y=event.values[1];
                z=event.values[2];
                String message = String.format("Magnetic (microT) : \n
                                x: %1$s \n y: %2$s \n z: %3$s", x, y, z);
                TextView text=(TextView)findViewById(R.id.textView7);
                text.setText(message);
        }
}
```


Orientation

- Sensor.TYPE_ORIENTATION
- มีหน่วยมุมเป็น องศา (degree)
- วนตามเข็มเป็นบวก
  - values[0]: around the z-axis (0 to 359)
    0=North, 90=East, 180=South, 270=West
  - values[1]: around x-axis (-180 to 180)
  - values[2]: around y-axis (-90 to 90)



Orientation (2)

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sensorManager=(SensorManager)getSystemService(SENSOR_SERVICE);
        sensorManager.registerListener(this,
                        sensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION),
                        SensorManager.SENSOR_DELAY_NORMAL);
}

public void onSensorChanged(SensorEvent event){
        int type = event.sensor.getType();
        float x, y, z;

        if(type==Sensor.TYPE_ORIENTATION){
                x=event.values[0];
                y=event.values[1];
                z=event.values[2];
                String message = String.format("Orientation (deg) : \n
                                x: %1$s \n y: %2$s \n z: %3$s", x, y, z);
                TextView text=(TextView)findViewById(R.id.textView8);
                text.setText(message);
        }
}
```


Atmospheric Pressure

- Sensor.TYPE_PRESSURE
- มีหน่วยเป็น millibar
- event.values[0]: Atmospheric pressure

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sensorManager=(SensorManager)getSystemService(SENSOR_SERVICE);
        sensorManager.registerListener(this,
                        sensorManager.getDefaultSensor(Sensor.TYPE_PRESSURE),
                        SensorManager.SENSOR_DELAY_NORMAL);
}
public void onSensorChanged(SensorEvent event){
        int type = event.sensor.getType();
        float x, y, z;
        if(type==Sensor.TYPE_PRESSURE){
                x=event.values[0];
                String message = String.format("Pressure (hPa or mbar): \n %1$s", x);
                TextView text=(TextView)findViewById(R.id.textView9);
                text.setText(message);
        }
}
```

Proximity

- Sensor.TYPE_PROXIMITY
- มีหน่วยเป็น เซนติเมตร
- event.values[0]: Proximity sensor distance
- Some proximity sensors only support a binary **near** or **far** measurement.

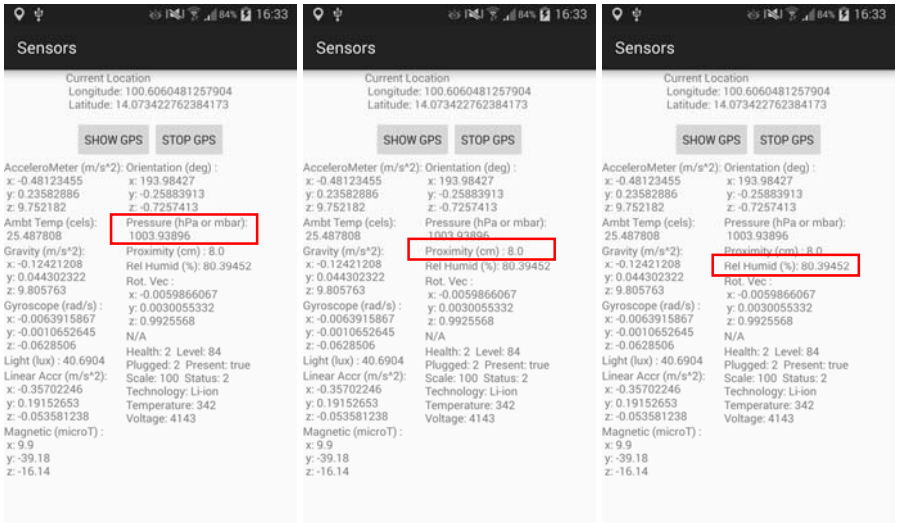
```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sensorManager=(SensorManager)getSystemService(SENSOR_SERVICE);
        sensorManager.registerListener(this,
                                sensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY),
                                SensorManager.SENSOR_DELAY_NORMAL);
}
public void onSensorChanged(SensorEvent event){
        int type = event.sensor.getType();
        float x, y, z;
        if(type==Sensor.TYPE_PROXIMITY){
                x=event.values[0];
                String message = String.format("Proximity (cm) : %1$s", x);
                TextView text=(TextView)findViewById(R.id.textView10);
                text.setText(message);
        }
}
```


Relative Humidity

- Sensor.TYPE_RELATIVE_HUMIDITY
- มีหน่วยเป็น %
- event.values[0] ค่าความชื้นสัมพันธ์

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sensorManager=(SensorManager)getSystemService(SENSOR_SERVICE);
        sensorManager.registerListener(this,
                sensorManager.getDefaultSensor(Sensor.TYPE_RELATIVE_HUMIDITY),
                SensorManager.SENSOR_DELAY_NORMAL);
}
public void onSensorChanged(SensorEvent event){
        int type = event.sensor.getType();
        float x, y, z;
        if(type==Sensor.TYPE_RELATIVE_HUMIDITY){
                x=event.values[0];
                String message = String.format("Rel Humid (%): \n %1$s", x);
                TextView text=(TextView)findViewById(R.id.textView11);
                text.setText(message);
        }
}
```


Atmospheric Pressure, Proximity and Relative Humidity



Rotation Vector

- Sensor.TYPE_ROTATION_VECTOR
- ไม่มีหน่วย
- SensorEvent.values[0]
  Rotation vector component along the x axis.
- SensorEvent.values[1]
  Rotation vector component along the y axis.
- SensorEvent.values[2]
  Rotation vector component along the z axis.

Rotation Vector (2)

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sensorManager=(SensorManager)getSystemService(SENSOR_SERVICE);
        sensorManager.registerListener(this,
                sensorManager.getDefaultSensor(Sensor.TYPE_ROTATION_VECTOR),
                SensorManager.SENSOR_DELAY_NORMAL);
}

public void onSensorChanged(SensorEvent event){
        int type = event.sensor.getType();
        float x, y, z;

        if(type==Sensor.TYPE_ROTATION_VECTOR){
                x=event.values[0];
                y=event.values[1];
                z=event.values[2];
                String message = String.format("Rot. Vec : \n
                                x: %1$s \n y: %2$s \n z: %3$s", x, y, z);
                TextView text=(TextView)findViewById(R.id.textView12);
                text.setText(message);
        }
}
```


Device Temperature

- Sensor.TYPE_TEMPERATURE
- มีหน่วยเป็น องศาเซลเซียส (°C)
- event.values[0] ค่าอุณหภูมิของเครื่อง (Device temperature)

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sensorManager=(SensorManager)getSystemService(SENSOR_SERVICE);
        sensorManager.registerListener(this,
                        sensorManager.getDefaultSensor(Sensor.TYPE_TEMPERATURE),
                        SensorManager.SENSOR_DELAY_NORMAL);
}
public void onSensorChanged(SensorEvent event){
        int type = event.sensor.getType();
        float x, y, z;
        if(type==Sensor.TYPE_TEMPERATURE){
                x=event.values[0];
                String message = String.format("Temp (cels): \n %1$s", x);
                TextView text=(TextView)findViewById(R.id.textView13);
                text.setText(message);
        }
}
```


Battery

- EXTRA_HEALTH
  - BATTERY_HEALTH_COLD
  - BATTERY_HEALTH_DEAD
  - BATTERY_HEALTH_GOOD *
  - BATTERY_HEALTH_OVERHEAT
  - BATTERY_HEALTH_OVER_VOLTAGE
- EXTRA_LEVEL
- EXTRA_PLUGGED
  - BATTERY_PLUGGED_AC Power source is an AC charger
  - BATTERY_PLUGGED_USB Power source is a USB port
- EXTRA_PRESENT : boolean indicating whether a battery is present
- EXTRA_SCALE : integer containing the maximum battery level
- EXTRA_STATUS : integer containing the current status constant
  BATTERY_STATUS_CHARGING, BATTERY_STATUS_DISCHARGING*, BATTERY_STATUS_FULL
- EXTRA_TECHNOLOGY : String describing the technology of the current battery
- EXTRA_TEMPERATURE : integer containing the current battery temperature
- EXTRA_VOLTAGE : integer containing the current battery voltage level

Health: 2 Level: 47
Plugged: 0 Present: true
Scale: 100 Status: 3
Technology: Li-ion
Temperature: 378
Voltage: 3752

Battery (2)

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        this.registerReceiver(this.batteryInfoReceiver,
                        new IntentFilter(Intent.ACTION_BATTERY_CHANGED));
}
```

Battery (3)

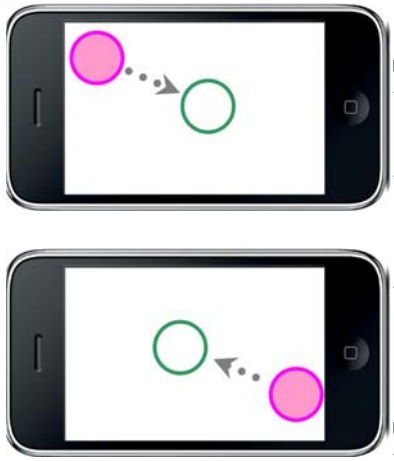
```
private BroadcastReceiver batteryInfoReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
        int health= intent.getIntExtra(BatteryManager.EXTRA_HEALTH,0);
        int level= intent.getIntExtra(BatteryManager.EXTRA_LEVEL,0);
        int plugged= intent.getIntExtra(BatteryManager.EXTRA_PLUGGED,0);
        boolean present= intent.getExtras().getBoolean(BatteryManager.EXTRA_PRESENT);
        int scale= intent.getIntExtra(BatteryManager.EXTRA_SCALE,0);
        int status= intent.getIntExtra(BatteryManager.EXTRA_STATUS,0);
        String technology= intent.getExtras().getString(BatteryManager.EXTRA_TECHNOLOGY);
        int temperature= intent.getIntExtra(BatteryManager.EXTRA_TEMPERATURE,0);
        int voltage= intent.getIntExtra(BatteryManager.EXTRA_VOLTAGE,0);
        TextView text=(TextView)findViewById(R.id.textView14);
        text.setText("Health: "+health+"   "+"Level: "+level+"\n"+
                        "Plugged: "+plugged+"   "+"Present: "+present+"\n"+
                        "Scale: "+scale+"   "+"Status: "+status+"\n"+
                        "Technology: "+technology+"\n"+
                        "Temperature: "+temperature+"\n"+
                        "Voltage: "+voltage+"\n");
        }
};
```


Battery (4)

References

- http://developer.android.com
- http://www.javacodegeeks.com/2010/09/android-location-based-services.html
- http://mobiledevtuts.com/android/ android-sdk-get-device-battery-information/
- http://developer.android.com/reference/ android/os/BatteryManager.html

Assignment 2



<u>ตัวเลือกที่ 1</u>

เกมส์เลื่อน ลูกบอล หรือ วงกลม ลงหลุม

- ✓ เขย่าเครื่องเพื่อสุ่มตำแหน่งของลูกบอลให้อยู่บริเวณใดก็ได้ของหน้าจอ (ใช้ Accelerometer ตรวจจับการเขย่าเครื่อง)
- ✓ อ่านค่าจาก Sensor เพื่อควบคุม ลูกบอล ให้ไหลไปในทิศทางที่ระดับต่ำกว่า โดยมีหลุมเป็นเป้าหมาย ต้องเอียงเครื่องไปในมุมต่าง ๆ เพื่อเลื่อนลูกบอลให้ลงหลุม
- ✓ หากมีการกลับหน้าจอจาก Portrait เป็น Landscape ตัว Application จะต้องไม่ถูก Destroy แล้ว Create ใหม่ (มีการทำ Handling Runtime Change)

<u>ตัวเลือกที่ 2</u>

เกมส์ทำนายชะตาชีวิต

- ✓ เขย่าโทรศัพท์ เหมือน เซียมซี เพื่อสุ่มเลือกเบอร์
- ✓ แสดงผลการทำนายจากเบอร์ที่ผู้เล่นได้รับ

Hint

```
class SensorInfo{
    float accX, accY, accZ;
    float ambTemp;
    float graX, graY, graZ;
    float gyrX, gyrY, gyrZ;
    float light;
    float laccX, laccY, laccZ;
    float magX, magY, magZ;
    float orX, orY, orZ;
    float pressure;
    float proximity;
    float humid;
    float rotX, rotY, rotZ;
    float temp;
}
```

สร้างคลาส SenserInfo
ไว้เก็บค่าจากเซนเซอร์ของเครื่อง

สร้างเทรด ไว้ตรวจจับการเขย่า

```
private Runnable pollTask = new Runnable() {
    public void run() {
        showDialog();
        hdr.postDelayed(pollTask, POLL_INTERVAL);
    }
};
```



```
public void showDialog() {
    String message1 = String.format("AcceleroMeter (m/s^2): \n
                        x: %1$s \n y: %2$s \n z: %3$s",
                        sensor_info.accX, sensor_info.accY, sensor_info.accZ);
    TextView text1 = (TextView) findViewById(R.id.textView1);
    text1.setText(message1);
    ...
    if(  (Math.abs(sensor_info.accX)>shake_threshold) ||
         (Math.abs(sensor_info.accY)>shake_threshold) ||
         (Math.abs(sensor_info.accZ)>shake_threshold) ) {
      if(!shown_dialog) {
        shown_dialog = true;
        final AlertDialog.Builder viewDialog = new AlertDialog.Builder(this);
        viewDialog.setIcon(android.R.drawable.btn_star_big_on);
        viewDialog.setTitle("ข้อความ");  viewDialog.setMessage("โทรศัพท์มีการเขย่า");
        viewDialog.setPositiveButton("OK",
            new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int which) {
                    dialog.dismiss();  shown_dialog = false;
                }
            });
        viewDialog.show();
      }//end if
    }//end if
}//end method
```


เมื่อ onResume ให้เรียกอ่านค่าจากเซนเซอร์
สถานะแบตเตอรี่ ทำให้หน้าจอสว่าง และปลุกเทรด

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

    sensorManager.registerListener(this, sensorManager.getDefaultSensor(
                                    Sensor.TYPE_ACCELEROMETER),
                                SensorManager.SENSOR_DELAY_NORMAL);
    sensorManager.registerListener(this, sensorManager.getDefaultSensor(
                                    Sensor.TYPE_AMBIENT_TEMPERATURE),
                                SensorManager.SENSOR_DELAY_NORMAL);
    ...

    this.registerReceiver(this.batteryInfoReceiver,
                    new IntentFilter(Intent.ACTION_BATTERY_CHANGED));
    if (!wl.isHeld()) {
        wl.acquire();
    }
    hdr.postDelayed(pollTask, POLL_INTERVAL);
}
```


เมื่อ onPause ให้หยุดการอ่านค่า ปลดการให้หน้าจอสว่าง และหยุดเทรด

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

    sensorManager.unregisterListener(this);

    this.unregisterReceiver(this.batteryInfoReceiver);

    if (wl.isHeld()) {
        wl.release();
    }

    hdr.removeCallbacks(pollTask);
}
```