Onboard Android Sensor Access

Agenda

6:00 Install Android Studio, your Android device driver, connect phone

6:15 Install your device SDK, download/unzip file linked in meetup event comments

6:30-8:00 Primary Workshop

IEEE DFW Sensor & IoT Forum

dallas-sensor.com

Speakers

Stacy Devino

Leroy Levin

John Lindsay

Wifi

SSID:

Password:

Onboard Android Sensor Access

Sensors

A device that measures something in the real world and provides an output value proportional to the magnitude of that “something”

Sensors in Apps

- Games (“Shake” apps, Pokemon and (Augmented Reality) – device orientation)
- Fishing (pressure)
- Driving (speed, position, orientation)

Onboard Android Sensor Access: Overview

6:50

What sensors are necessary for the application?

What sensors are available on the device? Saturation?

How to retrieve the sensor data?

How to display the sensor data?

Post-retrieval sensor data processing?

Miscellaneous concerns

Battery, CPU, UI,

Our Best Friend:
https://developer.android.com/guide/topics/sensors/sensors_overview.html

Leroy Levin

- Contact
  - Big On Mobile (Texas dba)
  - www.bigonmobile.com
  - bigonmobile@gmail.com
  - Twitter: @leroy2l

- Play Store app: “RV Expenses”

Was/Is

- Was
  - Hardware Tech
  - Unix Software Contractor
  - TI/Sterling/CA Contractor
  - CA Employee (10+ years)
  - Time off just because
- Now: into the IOT world
  - Microprocessors, sensors, MEAN stack dev, Node Red, M2X, Flow Designer, BlueMix, meetups, seminars, expos
  - Looking for work tomorrow

STACY DEVINO

- **Senior Android Innovator at The Home Depot Dallas Technology Center**
- **Works on Consumer Mobile App and Internal Product Innovation**
- **Six Sigma BlackBelt, Intel Innovator, DMS Member, Vintage game collector/restorer**
- **Women Techmakers Lead for Dallas/ Ft. Worth**

WEBSITES
www.stacydevino.com
www.ledgoes.com
www.openbrite.com

EMAIL
childofthehorn@gmail.com

G+
https://plus.google.com/+StacyDevino

TWITTER
@DoesitPew

What Are Sensors

- Sensors are devices that measure a real world physical property and return a data representation of that property.
  - Could be voltage/resistance/digital value
- Sensor are used for
  - Environmental/health monitoring
  - Location tracking/direction assist
  - Gaming, Weather prediction
  - Agriculture/crop management
  - Industrial processing, home/personal security

Android Sensor Categories

- Category by sensor implementation
  - Hardware (Base)
  - Software, Virtual (Composite)
- Category by type of data returned
  - Motion
  - Positional
  - Environmental
- Wake-up vs. non wake-up

Hardware/Software Sensors

- Hardware (Base) Sensors
  - Single sensor data but not the raw output of a physical sensor, bias/compensation may be applied
  - Acceleration, geomagnetic, angular change
- Software/Virtual/Synthetic (Composite) Sensors
  - Derive their data from one or more base sensors
  - Linear acceleration, gravity sensor

Note: Android does not require device manufacturers to build any particular types of sensors into their Android-powered devices, so devices can have a wide range of sensor configurations.

Motion Sensors

- Accelerometer
- Gravity
- Gyroscope
- Linear Acceleration
- Rotation Vector
- Step Counter
- Step Detector

Position Sensors

- Game Rotational Vector
- Geomagnetic Rotation Vector
- Magnetic Field
- Orientation
- Proximity

Environmental Sensors

- Ambient Temperature
- Light
- Pressure
- Relative Humidity

Wakeup vs. non Wakeup

- Wake-up sensors
  - Can wake up an app to deliver the sensor event
- Non wake-up sensors
  - App is not woken from suspend mode
  - App must keep a partial wake lock if event needed
  - Otherwise events stored in hardware FIFO
  - Events lost if FIFO overflow

Sensor Apps In Play Store

- Show sensor power consumption and sensor return data
  - AndroSensor
  - Physics Toolbox Sensor Suite
  - Sensor Box for Android

App Considerations

- When to enable the sensor
- Device resource consumption
- Data privacy/security

Manual vs. Automatic Sensing

- Manual/user enabled
  - user must provide interaction, start/ stop sensing.
  - user can on demand control duration of data gathered and stored
  - user must have incentive to continue monitoring
- Automatic enabled
  - user input is not required to start/stop data collection
  - lots of data gathered, but how much and what part of the data is really useful
  - data filtering more imperative to reduce data quantity

Device Resource Usage

- Battery
  - sensors themselves consume power. **Sensor.getPower**()
  - Power for CPU processing of software sensor data
  - App processing requirements
  - radio power, if data is uploaded to the cloud
- Data usage/storage
  - Data plan usage for uploading data to the cloud
  - batch data uploads to reduce radio startup/power down power consumption
  - Local data store
  - any in app advertising will increase data usage

Sensor Data and Privacy

- Personal space
  - Only a concern is device is lost/stolen
- Group sharing
  - limited to authorized access, may be full or partial data sharing
- Community sharing
  - user are anonymous, limit type of data shared, ensure privacy is respected
- *May require user to explicitly grant permission*

Know your design

- Which sensors are required, which are optional?
  - Can you support reduced functionality or should you deny support for reduced device
- Is default API sufficient or do you need to roll your own sensor data algorithms
- How pervasive is the support for your target sensors?
  - Android does not require a specific sensor support on a target device
- Will you Seamlessly support device upgrade
  - Easy transfer of any on device data store

Sensor Best Practices

- Unregister sensor listeners, else they continue to consume power
- Test on real devices
- Verify sensors exist before you try to use them
- Only sensor that are absolutely necessary
- Choose sensor deliver rate carefully
- Consider contextual enable/disable of sensors
- Be aware of startup time required for newly activated sensor to become stable (GPS satellite detection for instance)

Sensor Best Practices

- Unregister sensor listeners, else they continue to consume power
- Test on real devices
- Verify sensors exist before you try to use them
- Only sensor that are absolutely necessary
- Choose sensor deliver rate carefully
- Consider contextual enable/disable of sensors
- Be aware of startup time required for newly activated sensor to become stable (GPS satellite detection for instance)

Onboard Android Sensor Access

Recall from 7/14: Only ~4.5 Lines of “Necessary” Code

.5 – Tell Android Studio that you'll implement sensor functionality

- implements SensorEventListener

1.5 – Get instance of SensorManager (system service that manages sensors)

- getSystemService(SENSOR_SERVICE)

2.5 – Get instance of a Sensor (an individual sensor)

- SensorManager.getDefaultType(int typeOfSensor)

3.5 – Register SensorEventListener (for callback functions on sensor events)

mSensorManager.registerListener(SensorEventListener sensorEventListener, Sensor sensor,

int rate);

4.5 – Retrieve Values on a SensorEvent (something happens with the sensor)

event.values[0]

Onboard Android Sensor Access

7:00

Default Android Class Declaration Statement for An Activity

public class MainActivity extends Activity

.5 Telling Android Studio that you'll implement sensor functionality

public class SensorRawAccelerometerActivity extends Activity *implements SensorEventListener*

Exercise 1 (Group, ~10 minutes)

- Download the project from Meetup event comments and unzip
- In Android Studio: File, Open
- Tell Android Studio that you'll implement sensor functionality
  - Also Uncomment “@override ” by onAccuracyChanged and onSensorChanged (// = comments in Android)
- Launch the app on your device (“Play” button)
  - Choose your phone
  - You'll only see “SensorRawValues” title now and labels

Onboard Android Sensor Access

Android Activity lifecycle – onCreate(), onResume(), onPause()

1.5 Get SensorManager instance

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

2.5 Attempt to get sensor instance

Supported sensor types in Android

SensorManager.getDefaultType(int typeOfSensor)

Today - we're using accelerometer

```
if (!null == (mAccelerometer =
    mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)))

    // do stuff;
```

Onboard Android Sensor Access

3.5 Tell Android to listen for sensor events

New values, accuracy change

Register SensorEventListener (onResume())

mSensorManager.registerListener(SensorEventListener sensorEventListener, Sensor sensor,

int rate); (rate in microseconds)

(Be Kind) Unregister SensorEventListener (onPause())

mSensorManager.unregisterListener(this);

Onboard Android Sensor Access

4.5 Retrieve Values on a SensorEvent

```
public void onSensorChanged(SensorEvent event)
        if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER)
            float x = event.values[0]
```

Data is sensor specific (Table 1)

Accelerometer – float values for x, y, and z

```
float x = event.values[0];
float y = event.values[1];
float z = event.values[2];
```

Exercise 2 (~20-30)

get SensorManager instance(onCreate()), accelerometer instance(onCreate()), register SensorEventListener (onResume()), retrieve accelerometer values (onSensorChanged)

- Set breakpoints, Run in Debug mode to confirm values

Code Walkthrough: Onboard Android Sensor Access

8:05

How to display sensor data ?

Normal Android Activity layout fields

Retrieved Sensor Data :

```
float x = event.values[0];
float y = event.values[1];
float z = event.values[2];
```

Output Display Fields

```
mXValueView.setText(String.valueOf(x));
mYValueView.setText(String.valueOf(y));
mZValueView.setText(String.valueOf(z));
```

Exercise 3 (~10)

- Setup display fields
- Run App