Leibniz Universität Hannover

Mobile Interaction Design

Introduction

Human-Computer Interaction Group

Prof. Dr. Michael Rohs
michael.rohs@hci.uni-hannover.de

Zeitplan

Termin	Datum	Thema
1	20.10.	Introduction to topic, related work, libGDX
2	27.10.	Brainstorming, group formation, smartwatches
3	3.11.	Interaction technique, device combination, scenario
4	10.11.	Storyboards, persona, paper prototyping
5	17.11.	Paper prototyping test
6	24.11.	Start of software prototype
7	1.12.	Software prototyping
8	8.12.	Software prototyping
9	15.12.	Heuristic evaluation
10	22.12.	Software prototyping
11	12.1.	Think-aloud user study
12	19.1.	Software prototyping
13	26.1.	Preparation of final presentation
14	2.2.	Final presentation

Wiki (← LDAP)

https://wiki.hci.uni-hannover.de

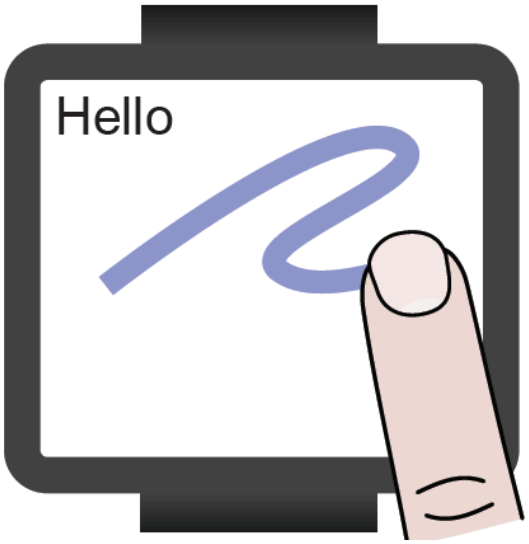
https://ldap.hci.uni-hannover.de

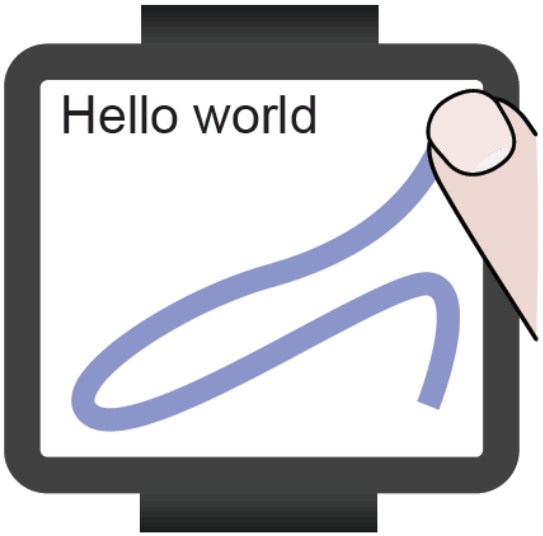
INTRODUCTION

Guo, Paek: Tilt for Wrist-Only Interactions on Smartwatches

Anhong Guo and Tim Paek. 2016. Exploring tilt for no-touch, wrist-only interactions on smartwatches. In Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '16). ACM, New York, NY, USA, 17-28. DOI: http://dx.doi.org/10.1145/2935334.2935345

Mottelson et al.: Invisiboard – Word Gestures on Smartwatches



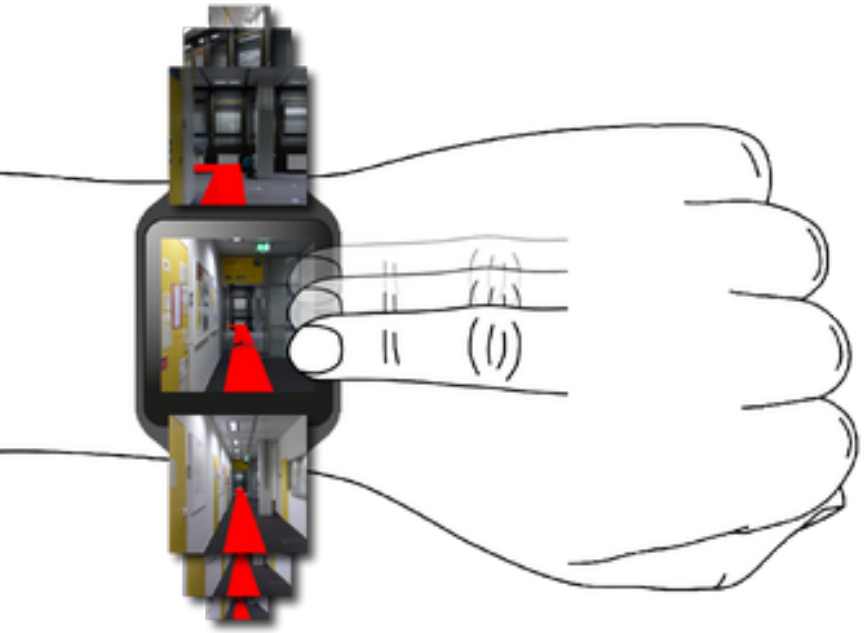


Aske Mottelson, Christoffer Larsen, Mikkel Lyderik, Paul Strohmeier, and Jarrod Knibbe. 2016. Invisiboard: Maximizing display and input space with a full screen text entry method for smartwatches. In Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '16). ACM, New York, NY, USA, 53-59. DOI: http://dx.doi.org/10.1145/2935334.2935360

Yeo et al.: WatchMI – Pressure Input on Smartwatches

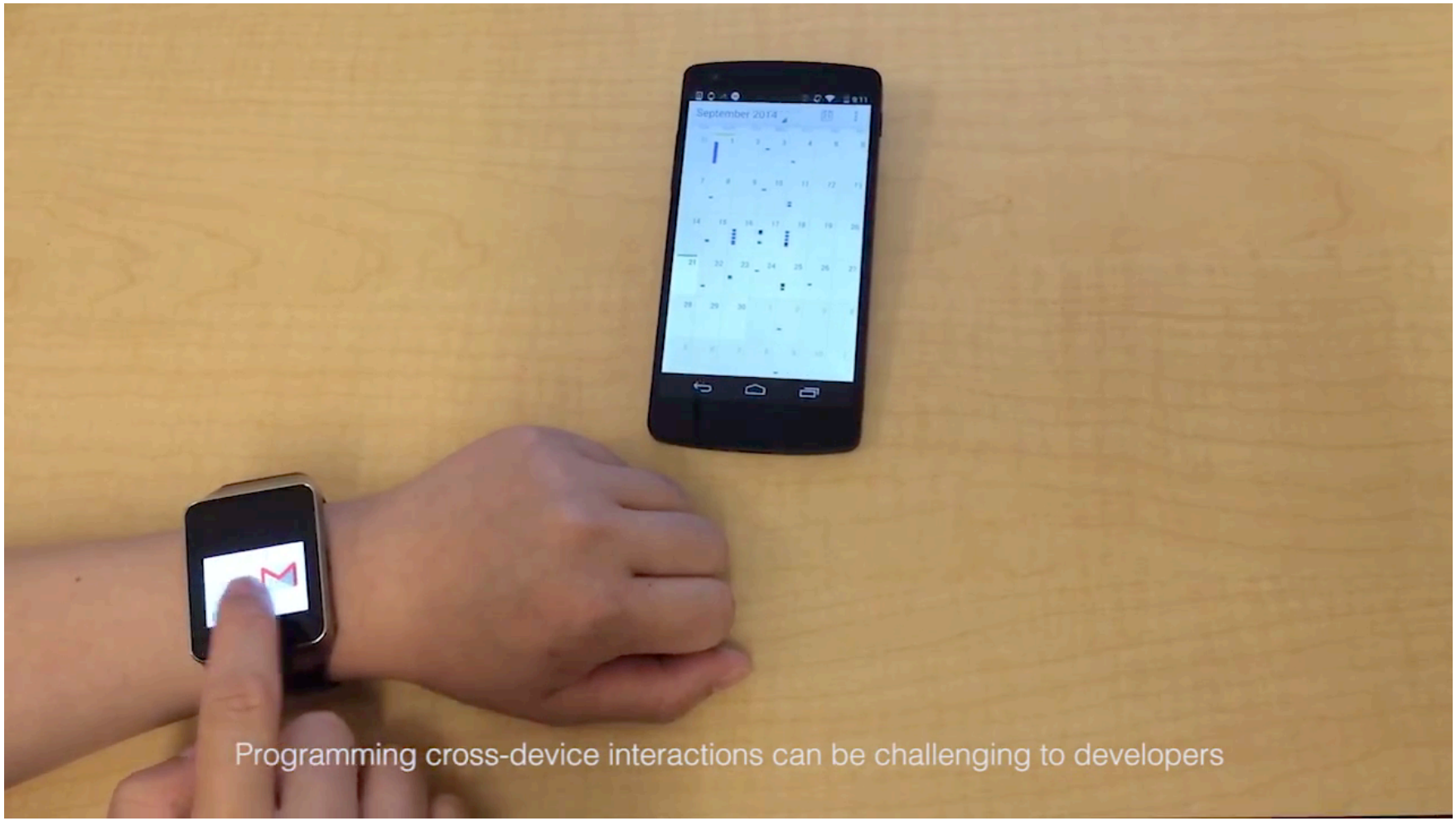
Hui-Shyong Yeo, Juyoung Lee, Andrea Bianchi, and Aaron Quigley. 2016. WatchMI: pressure touch, twist and pan gesture input on unmodified smartwatches. In Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '16). ACM, New York, NY, USA, 394-399. DOI: http://dx.doi.org/10.1145/2935334.2935375

Wenig et al.: ScrollingHome – Indoor Navigation on Smartwatches



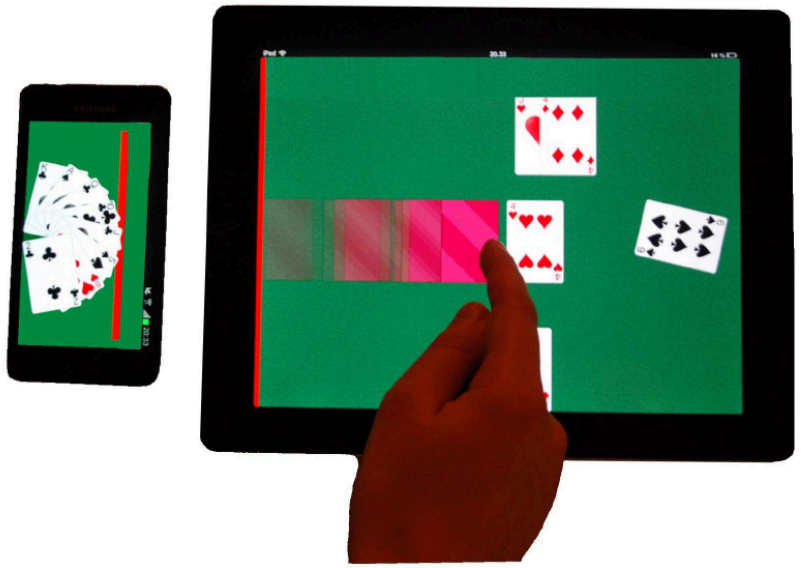
Dirk Wenig, Alexander Steenbergen, Johannes Schöning, Brent Hecht, and Rainer Malaka. 2016. ScrollingHome: bringing image-based indoor navigation to smartwatches. In Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '16). ACM, New York, NY, USA, 400-406. DOI: http://dx.doi.org/10.1145/2935334.2935373

Chi et al.: DemoScript Cross-Device Interaction Scripting



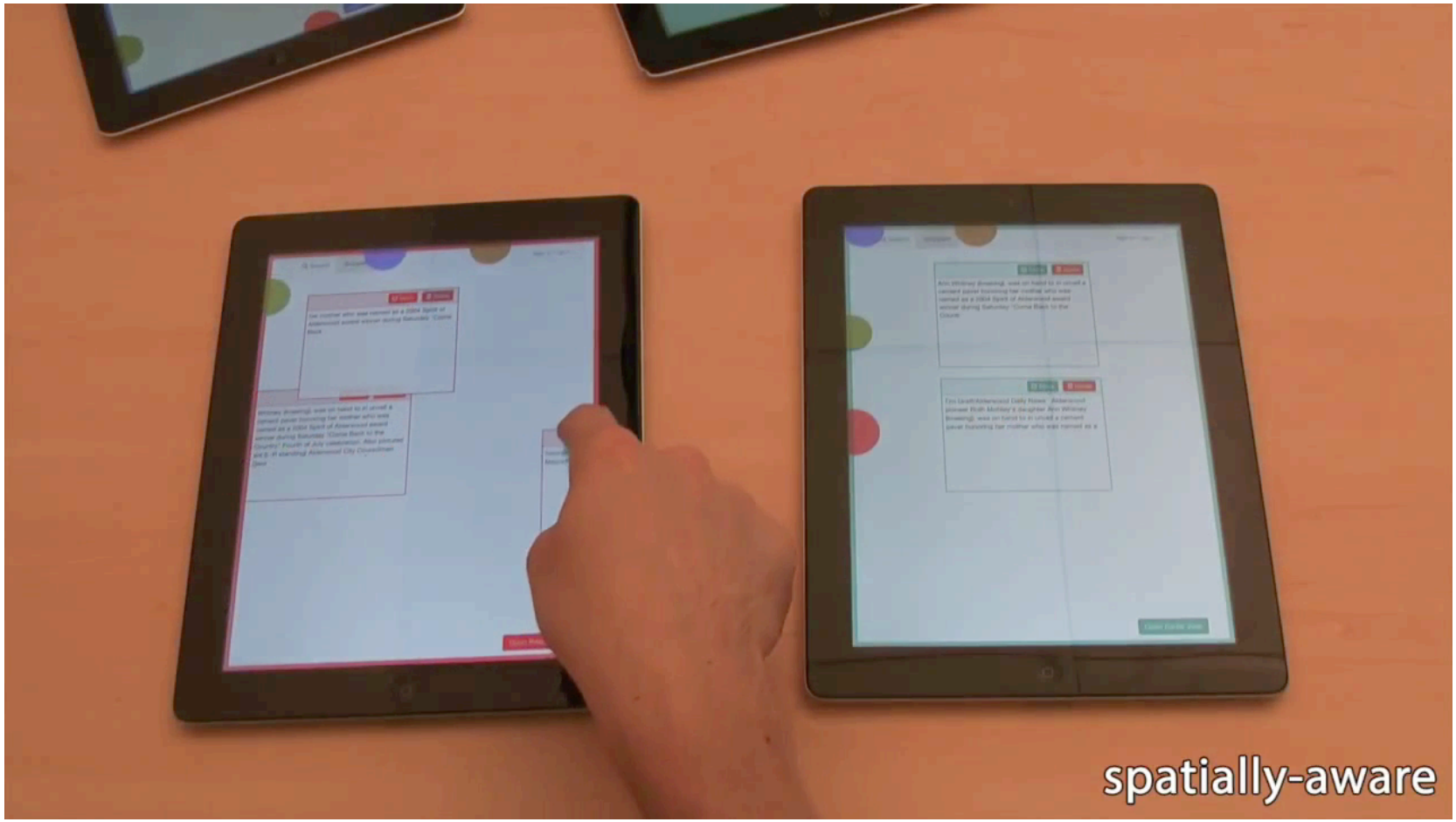
Pei-Yu (Peggy) Chi, Yang Li, and Björn Hartmann. 2016. Enhancing Cross-Device Interaction Scripting with Interactive Illustrations. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16). ACM, New York, NY, USA, 5482-5493. DOI: http://dx.doi.org/10.1145/2858036.2858382

Skov et al.: Cross-Device Interactions – Playing Cards



Mikael B. Skov, Jesper Kjeldskov, Jeni Paay, Heidi P. Jensen, and Marius P. Olsen. 2015. Investigating Cross-Device Interaction Techniques: A Case of Card Playing on Handhelds and Tablets. In Proceedings of the Annual Meeting of the Australian Special Interest Group for Computer Human Interaction (OzCHI '15),, 446-454. DOI: http://dx.doi.org/10.1145/2838739.2838763

Rädle et al: Spatially-Aware Cross-Device Interactions



Roman Rädle, Hans-Christian Jetter, Mario Schreiner, Zhihao Lu, Harald Reiterer, and Yvonne Rogers. 2015. Spatially-aware or Spatially-agnostic?: Elicitation and Evaluation of User-Defined Cross-Device Interactions. In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15). ACM, New York, NY, USA, 3913-3922. DOI: http://dx.doi.org/10.1145/2702123.2702287

Strohmeier: DisplayPointers – Cross-Device Interactions

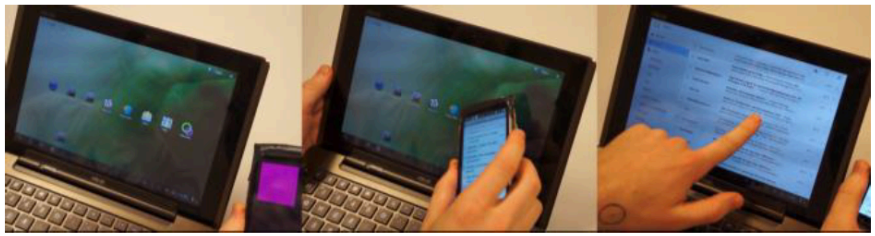


Figure 3 - Two different ways of opening an e-mail application. In the center image it is opened using a phone in the right hand image using a finger.

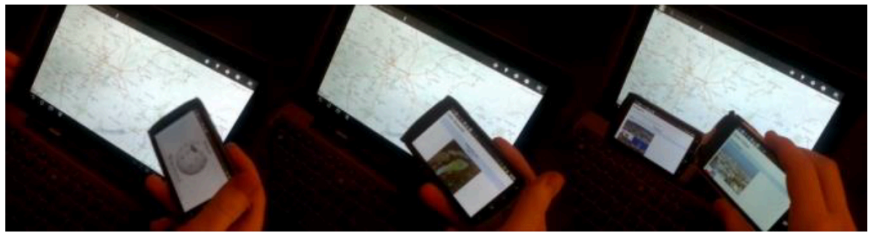


Figure 5 – A 'context' application shows Wikipedia articles which are linked to locations on a map. The user can use multiple phones to collect and arrange information.

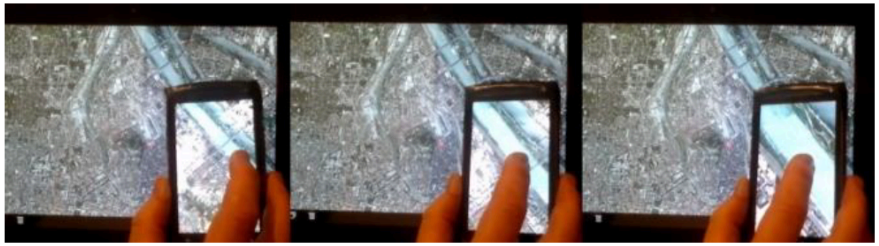
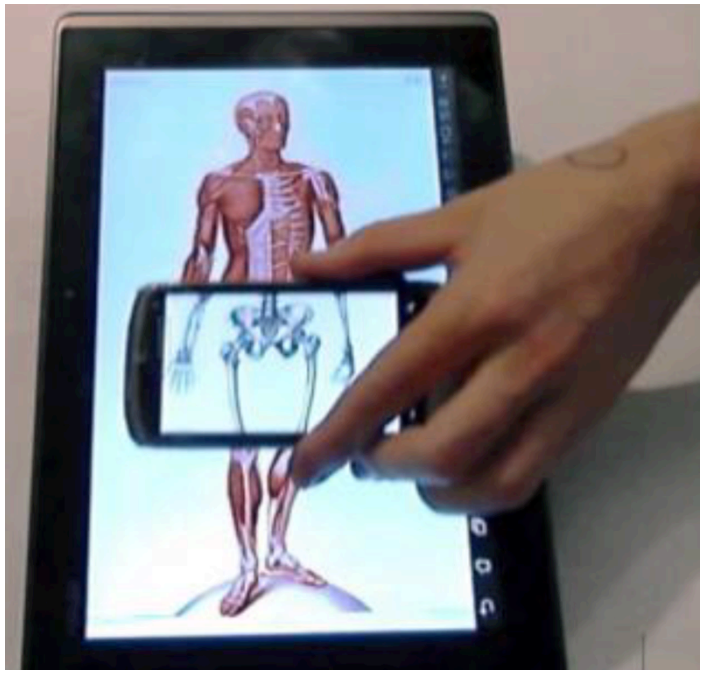


Figure 4 - Using a Touch & Tilt gesture for selecting zoom level.



Paul Strohmeier. 2015. DisplayPointers: seamless cross-device interactions. In Proceedings of the 12th International Conference on Advances in Computer Entertainment Technology (ACE '15). ACM, New York, NY, USA, , Article 4 , 8 pages. DOI: http://dx.doi.org/10.1145/2832932.2832958

Code Space: Touch + Air Interactions for Meetings

- Touchscreen
- Mobile devices
- Kinect

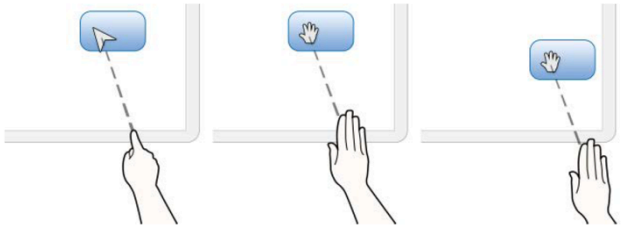


Fig. 3. Pointing + manipulating with hand postures and skeletal tracking.

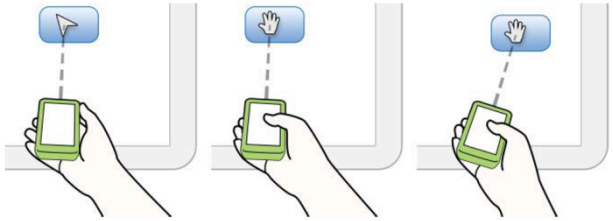
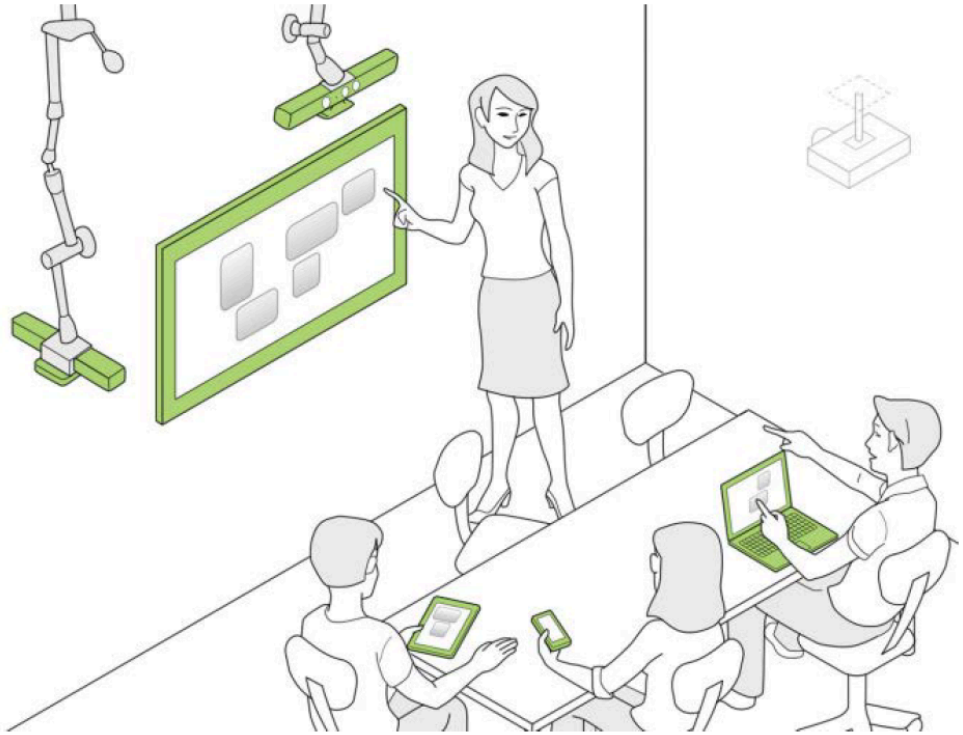


Fig. 4. Pointing and manipulating with a touch-enabled phone.



Andrew Bragdon, Rob DeLine, Ken Hinckley, and Meredith Ringel Morris. 2011. Code space: touch + air gesture hybrid interactions for supporting developer meetings. In Proceedings of Interactive Tabletops and Surfaces (ITS '11). ACM, 212-221. DOI=http://dx.doi.org/10.1145/2076354.2076393

Hamilton, Wigdor: Conductor – Cross-Device Interactions

Peter Hamilton and Daniel J. Wigdor. 2014. Conductor: enabling and understanding cross-device interaction. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14). ACM, New York, NY, USA, 2773-2782. DOI: http://dx.doi.org/10.1145/2556288.2557170

BRAINSTORMING

Brainstorm Ideas for Smartwatch Project

- Cross-Device Interactions with Smartwatches
  - Smartwatch + tablet
  - Smartwatch + phone
  - Smartwatch + interactive table
  - Smartwatch + TV / wall display
  - Smartwatch + projector
  - Smartwatch + RFID-tags, QR-codes
  - Smartwatch + Kinect
  - Smartwatch + EMS
  - Smartwatch + Coffee machine?
- (Re-)implement an interaction technique
- Develop an application scenario

Brainwriting

- Form brainstorm groups
- Repeat 5 times
  - 3 minutes: on paper, fill one row with 3 ideas
  - Pass on paper clockwise
  - Read other ideas, fill next line with 3 more ideas
- Select the 3 best ideas
  - 15 minutes
  - Present selected ideas
- Form project groups

Idee 1...	Idee 2...	Idee 3...
Idee 4...	Idee 2b...	Idee 5...
Idee 1b...	Idee 2c...	Idee 6...

Happy Brainstorming

Slide: Max Maurer

Brainstorming Results

Slide: Max Maurer

Team 1

Team 2

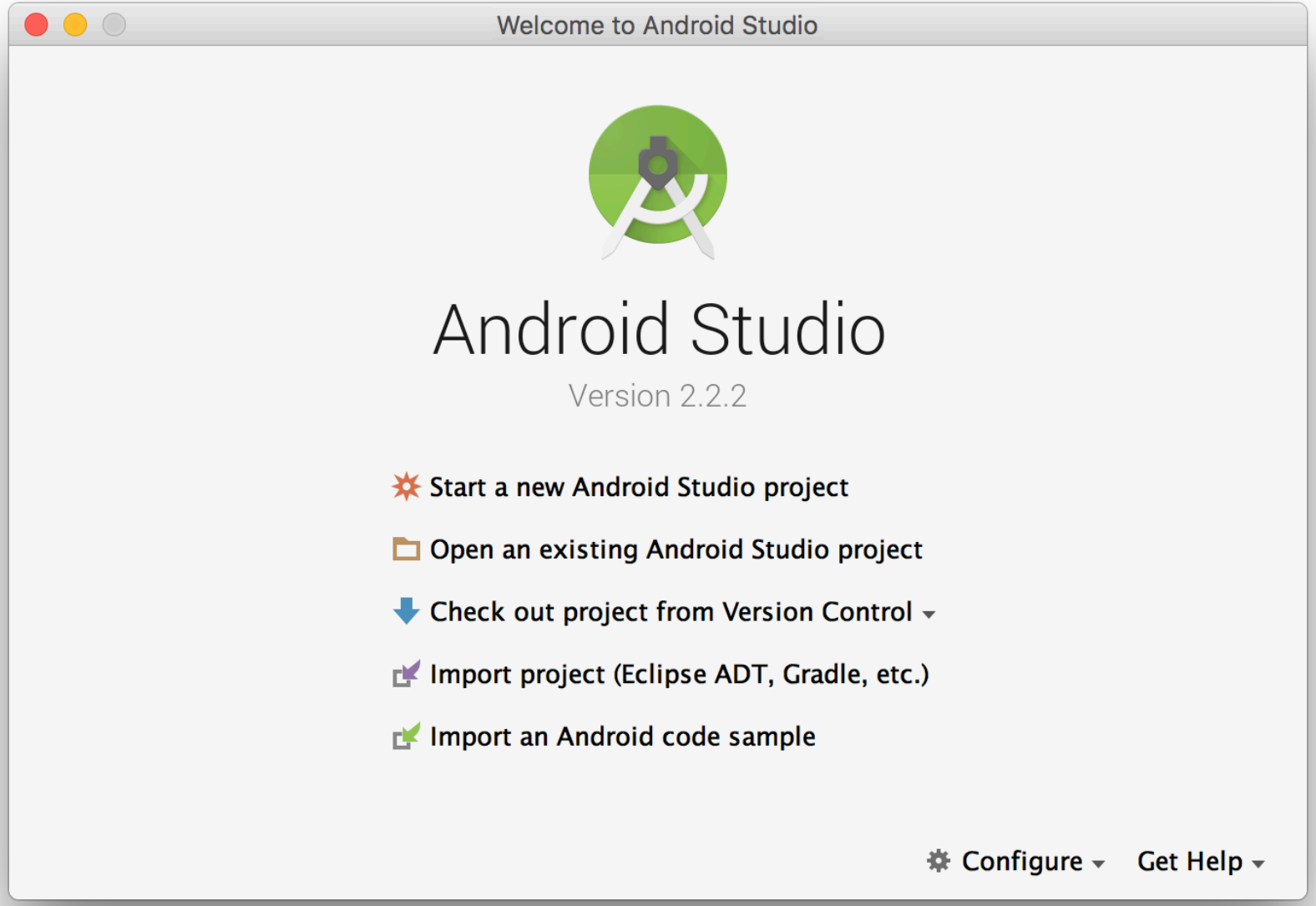
Team 3

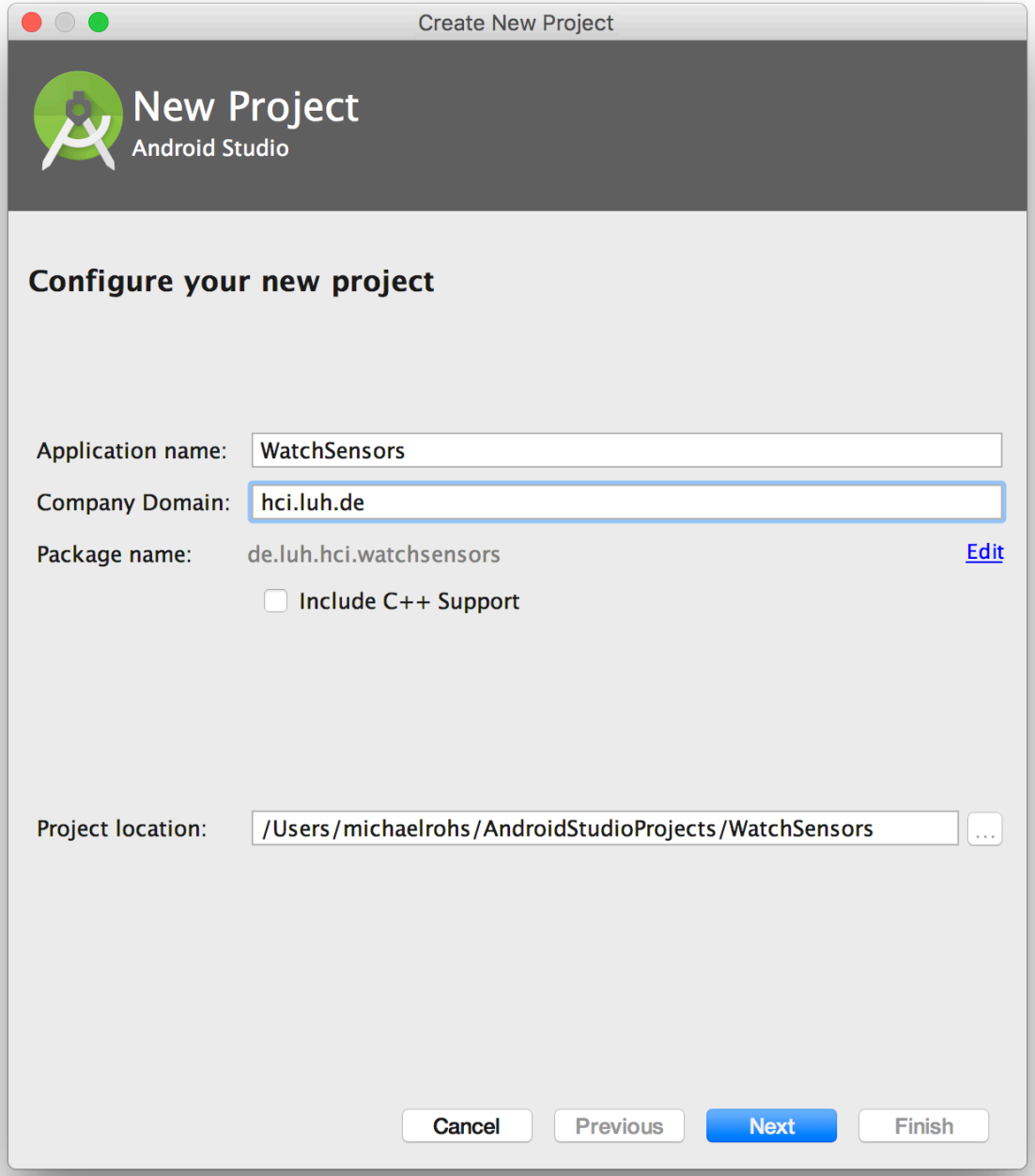
Team 4

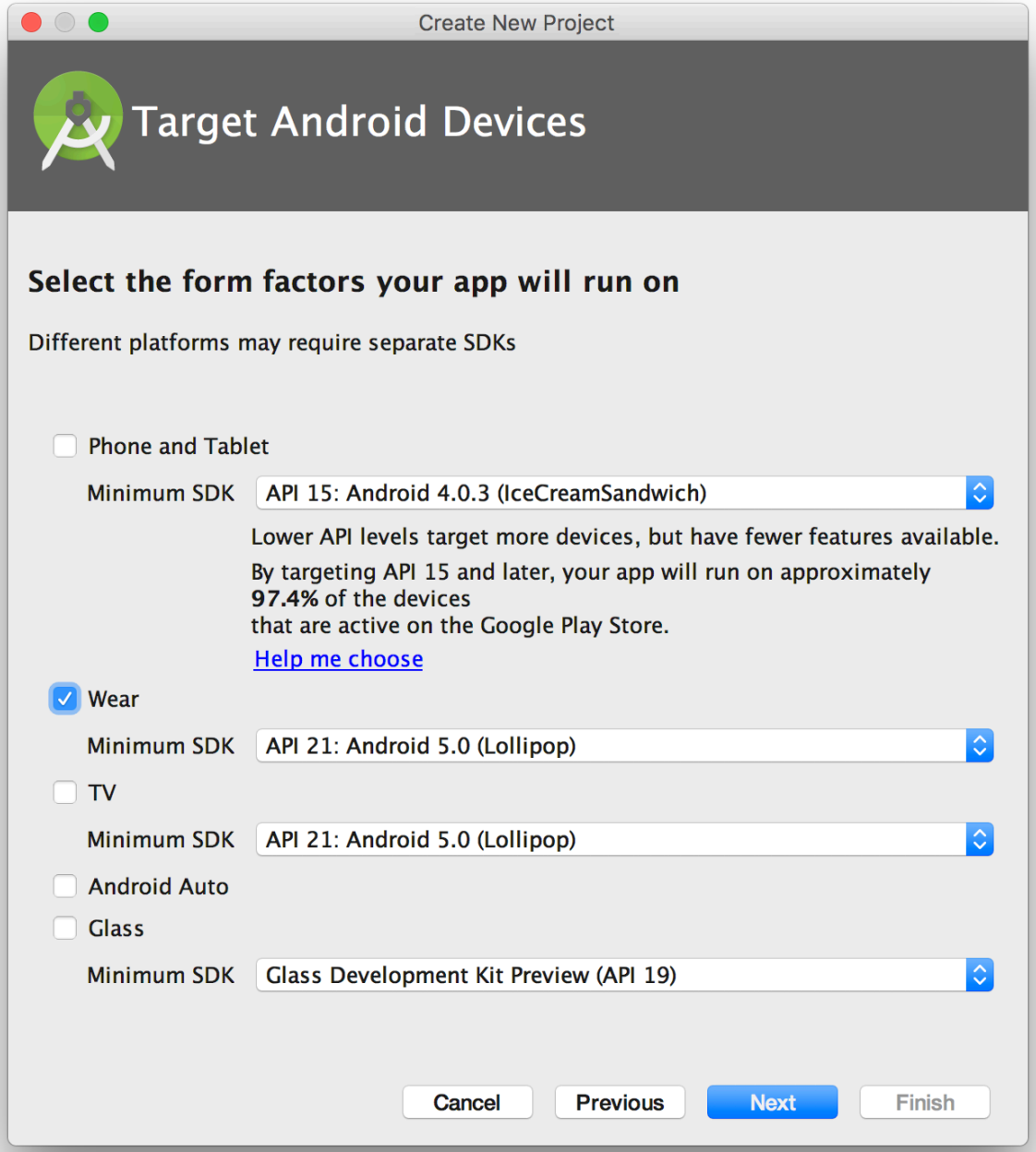
Team 5

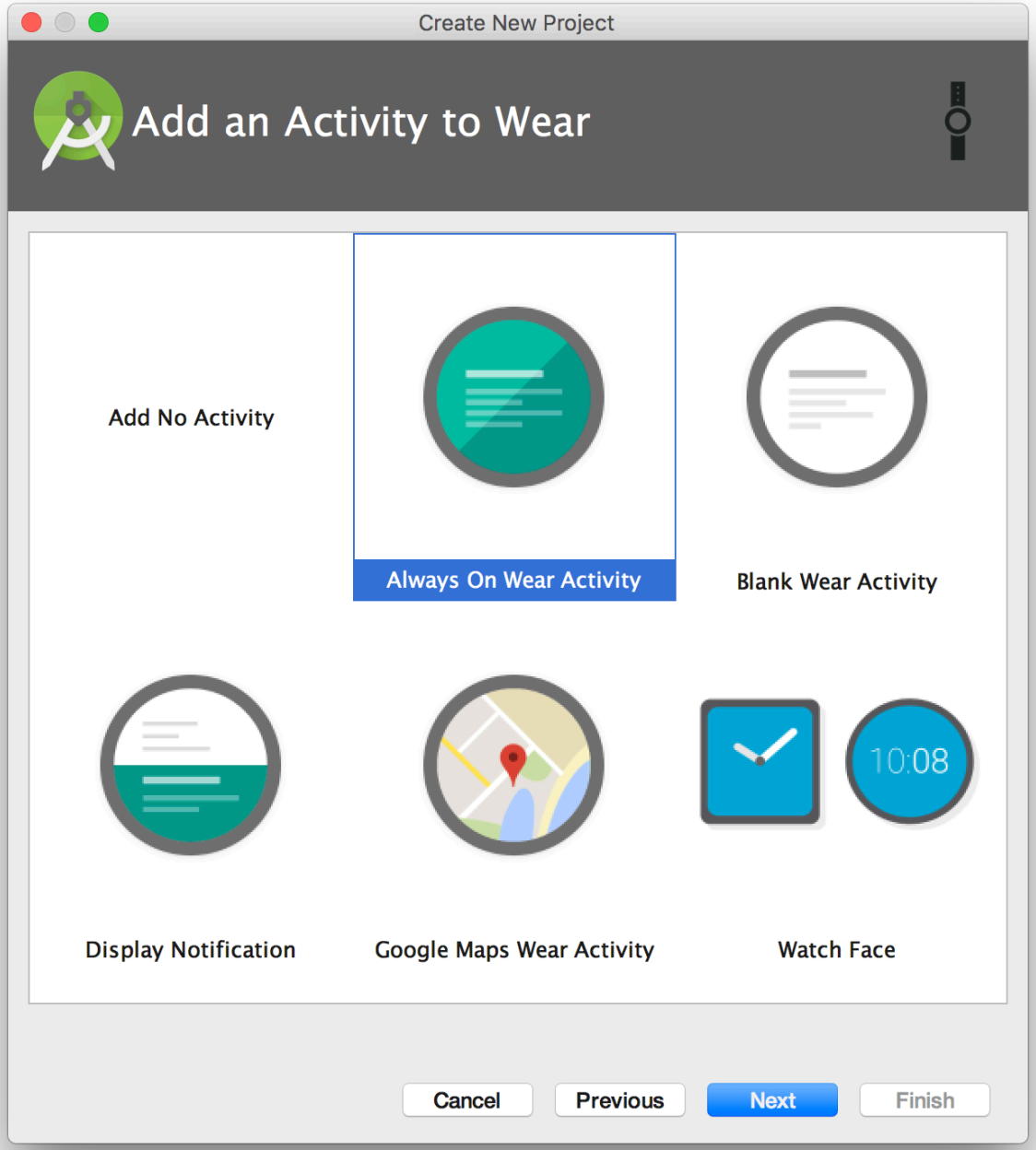
SMARTWATCH SENSORS

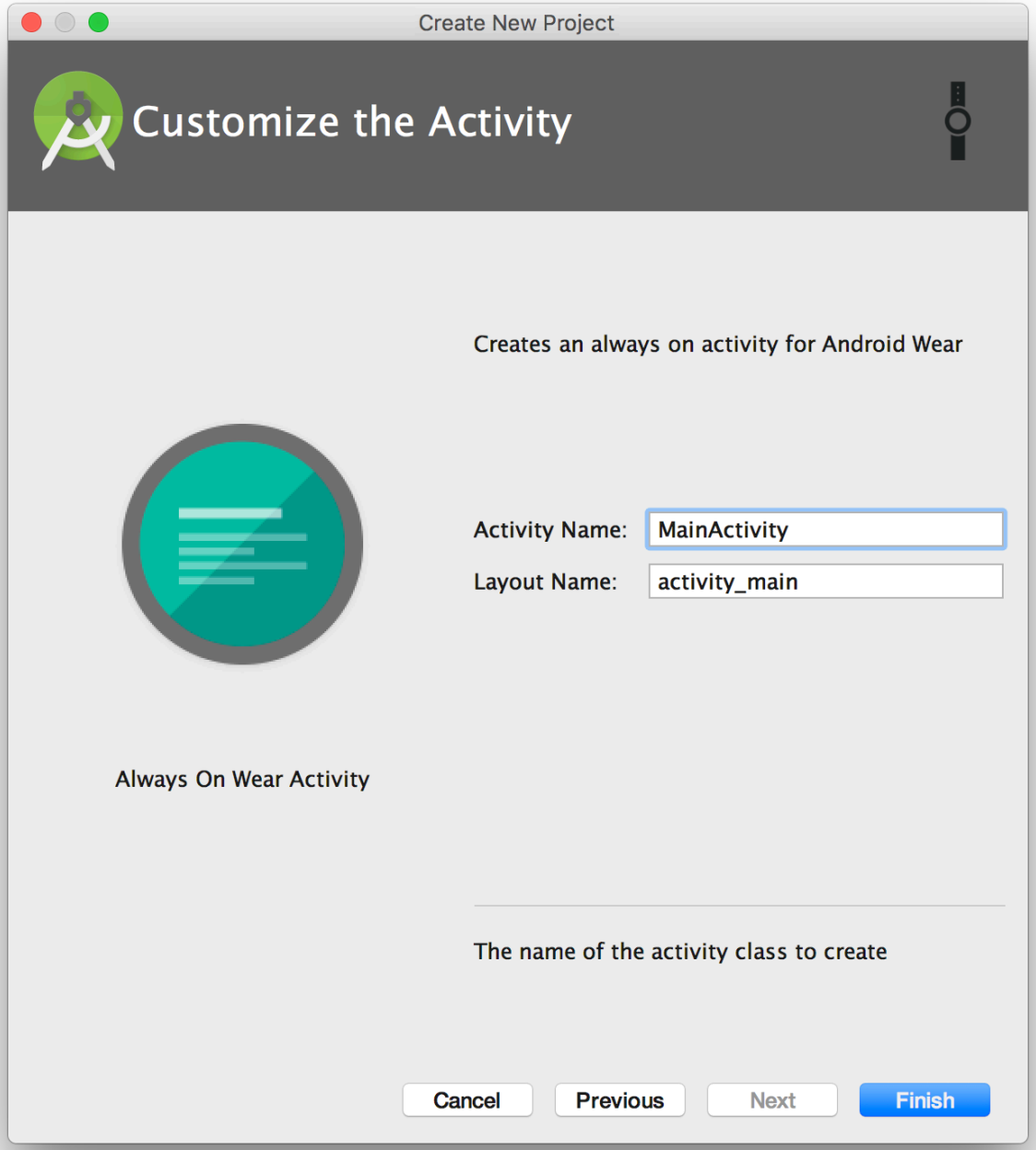
Create a new Project



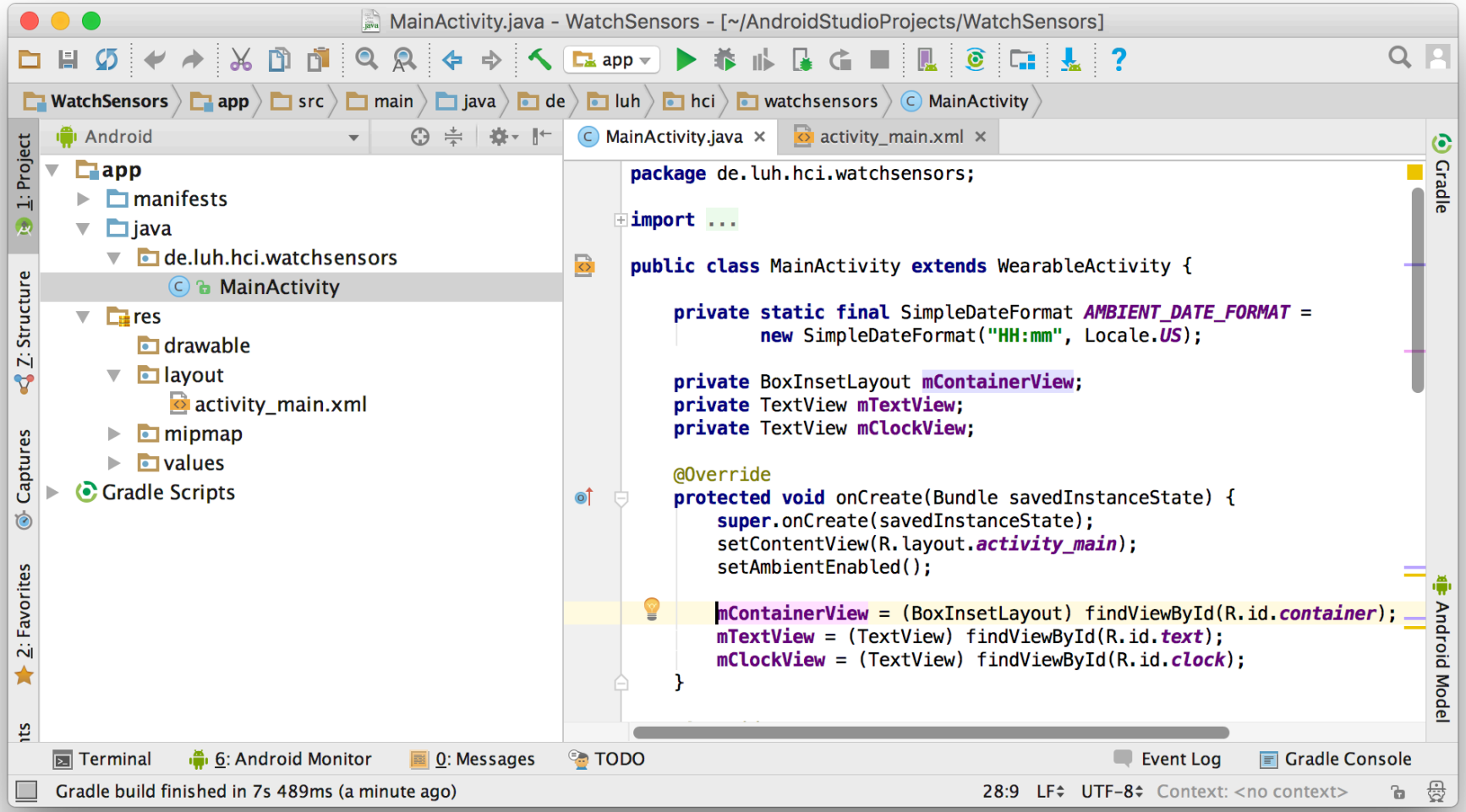

Create New Project
New Project
Android Studio
Configure your new project
Application name:
WatchSensors
Company Domain:
hci.luh.de
Package name:
de.luh.hci.watchsensors
Edit
Include C++ Support
Project location:
/Users/michaelrohs/AndroidStudioProjects/WatchSensors
...
Cancel
Previous
Next
Finish


Create New Project
Target Android Devices
Select the form factors your app will run on
Different platforms may require separate SDKs
Phone and Tablet
Minimum SDK
API 15: Android 4.0.3 (IceCreamSandwich)
Lower API levels target more devices, but have fewer features available.
By targeting API 15 and later, your app will run on approximately
97.4% of the devices
that are active on the Google Play Store.
Help me choose
Wear
Minimum SDK
API 21: Android 5.0 (Lollipop)
TV
Minimum SDK
API 21: Android 5.0 (Lollipop)
Android Auto
Glass
Minimum SDK
Glass Development Kit Preview (API 19)
Cancel
Previous
Next
Finish

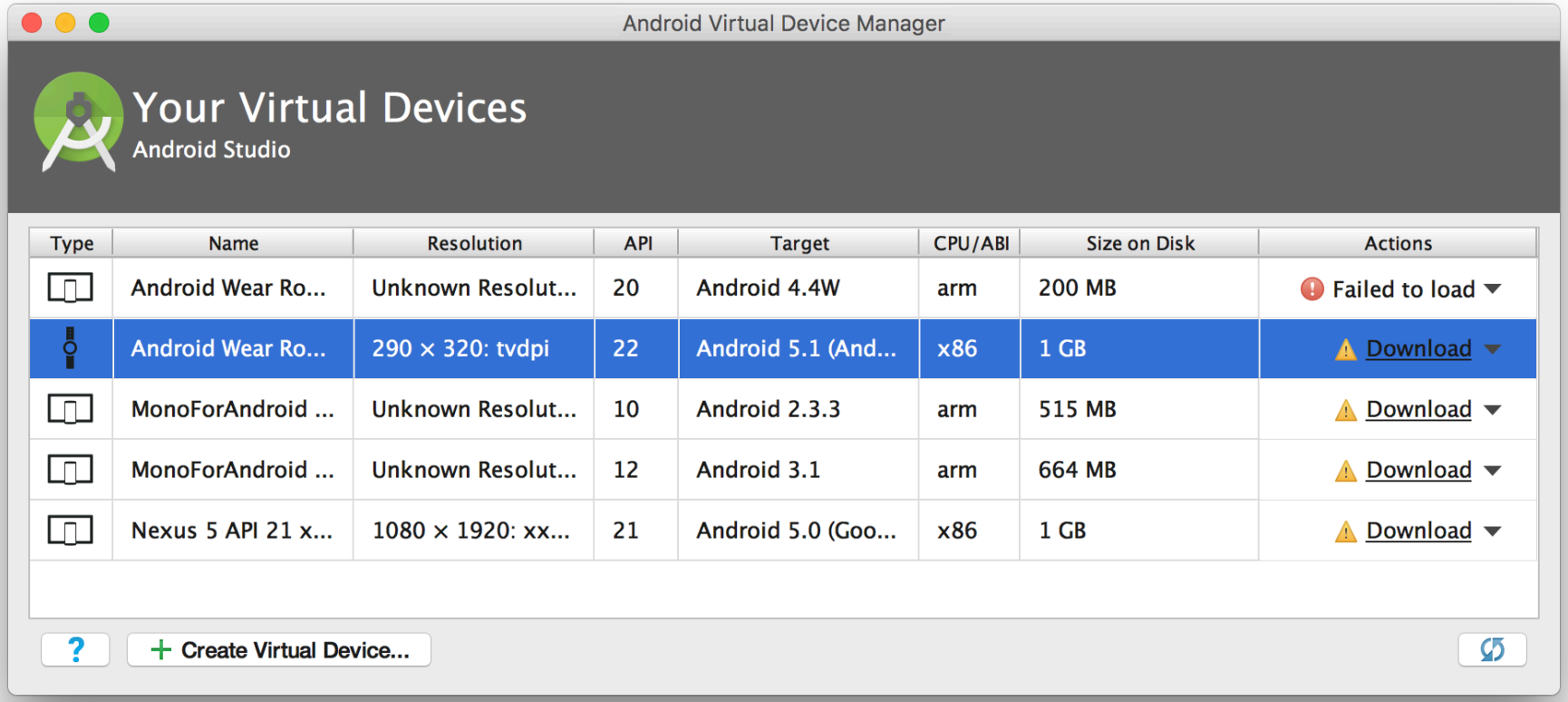

Create New Project
Add an Activity to Wear
Add No Activity
Always On Wear Activity
Blank Wear Activity
Display Notification
Google Maps Wear Activity
Watch Face
10:08
Cancel
Previous
Next
Finish


Create New Project
Customize the Activity
Creates an always on activity for Android Wear
Activity Name:
MainActivity
Layout Name:
activity_main
Always On Wear Activity
The name of the activity class to create
Cancel
Previous
Next
Finish

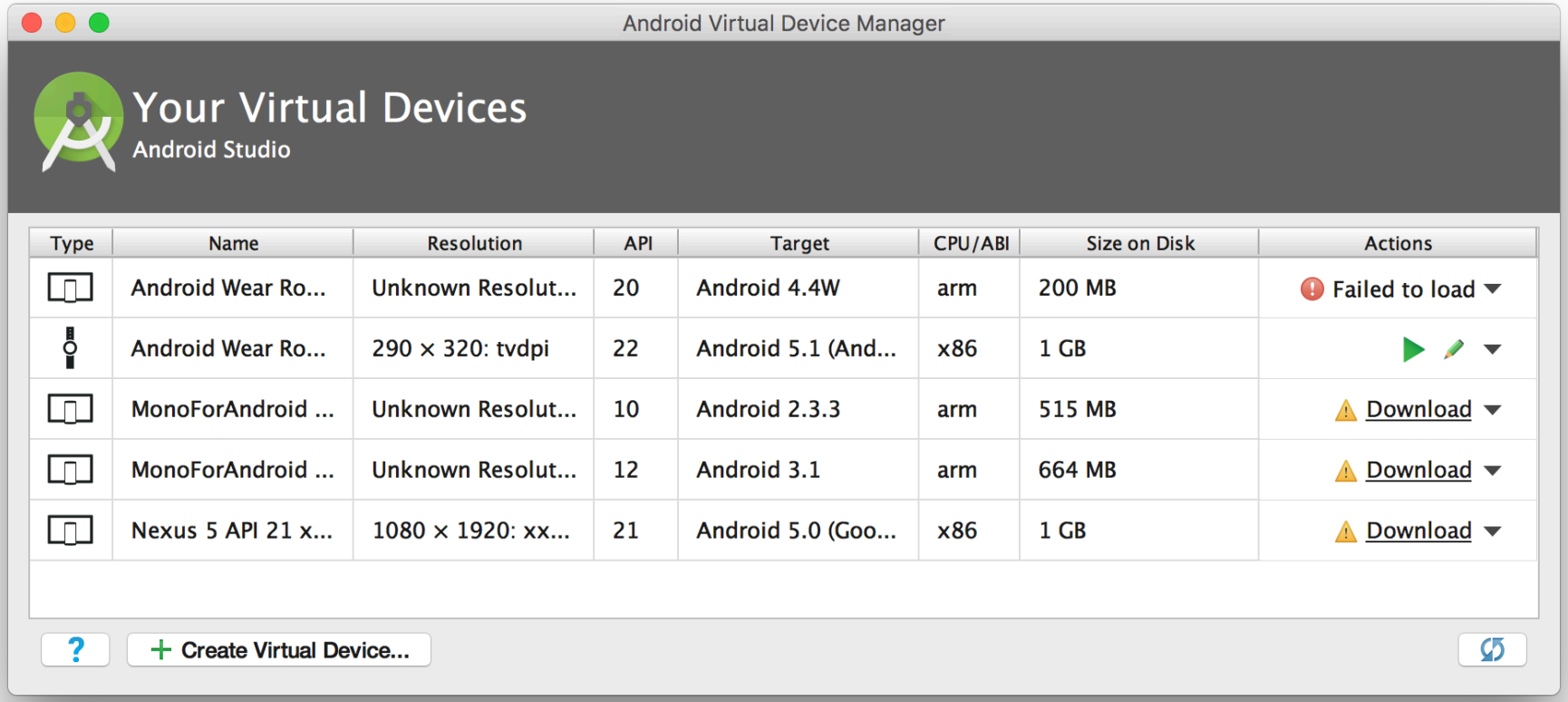
Generated Files



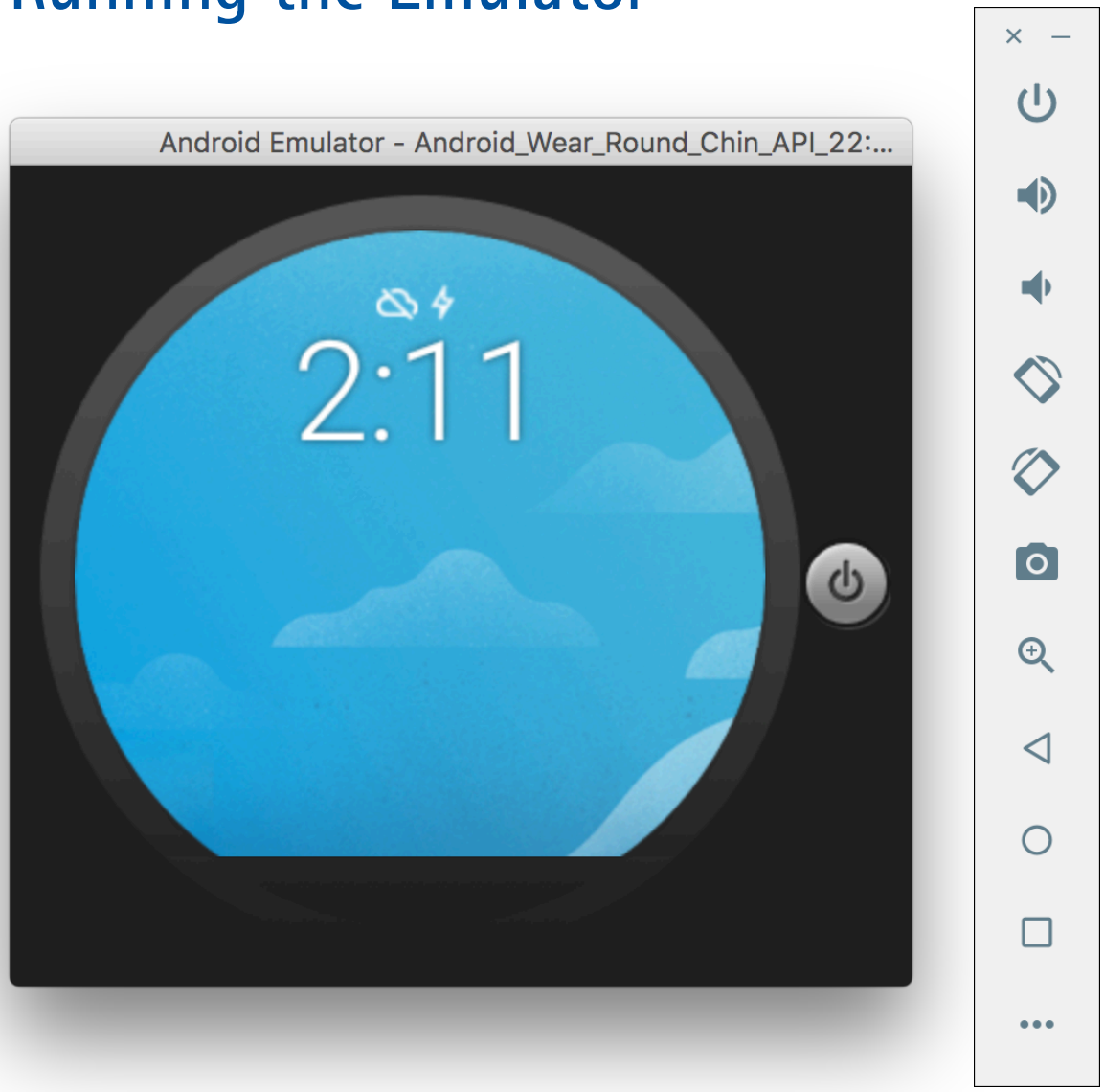
Download System Image for Watch

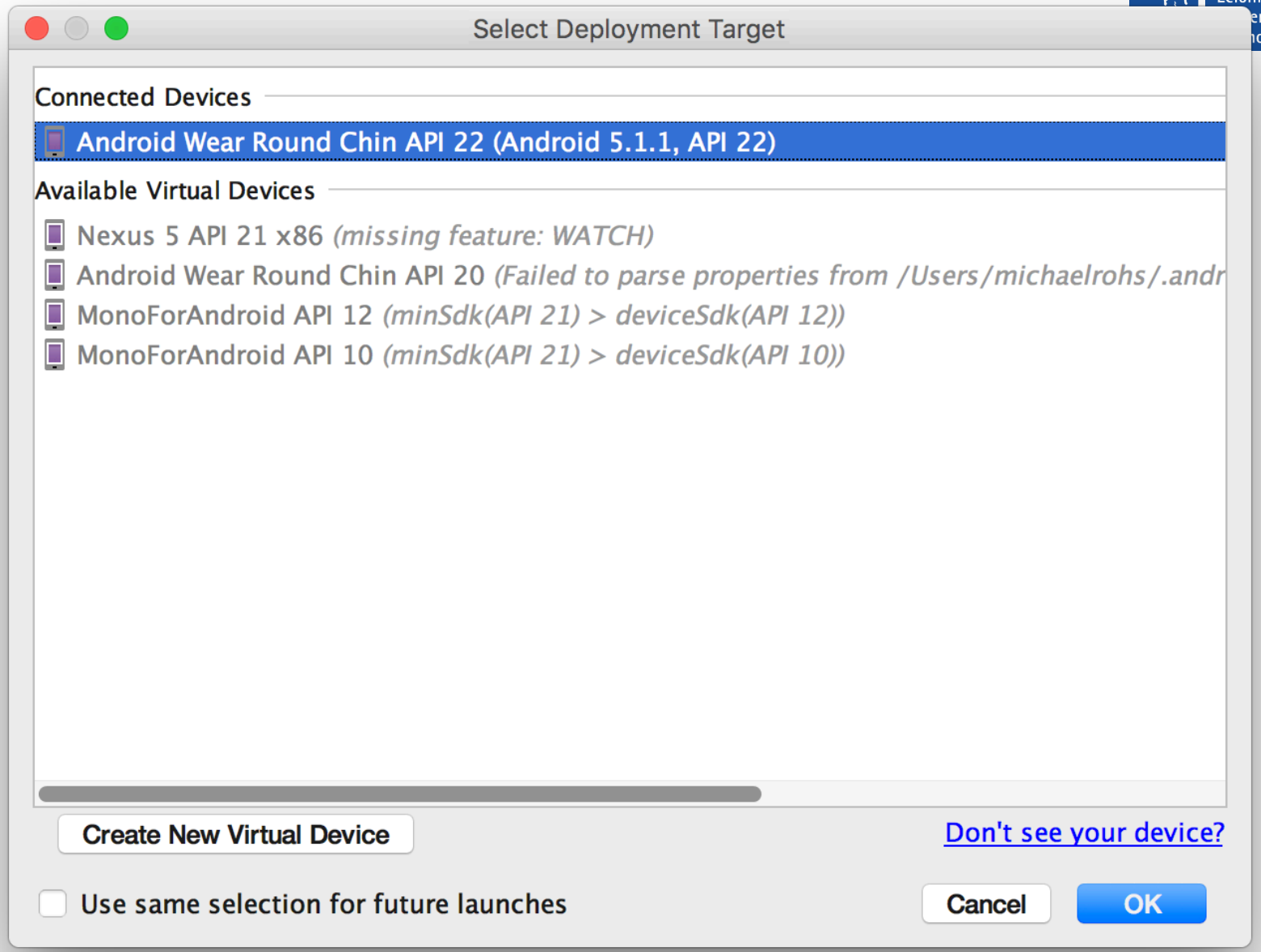


Download System Image for Watch

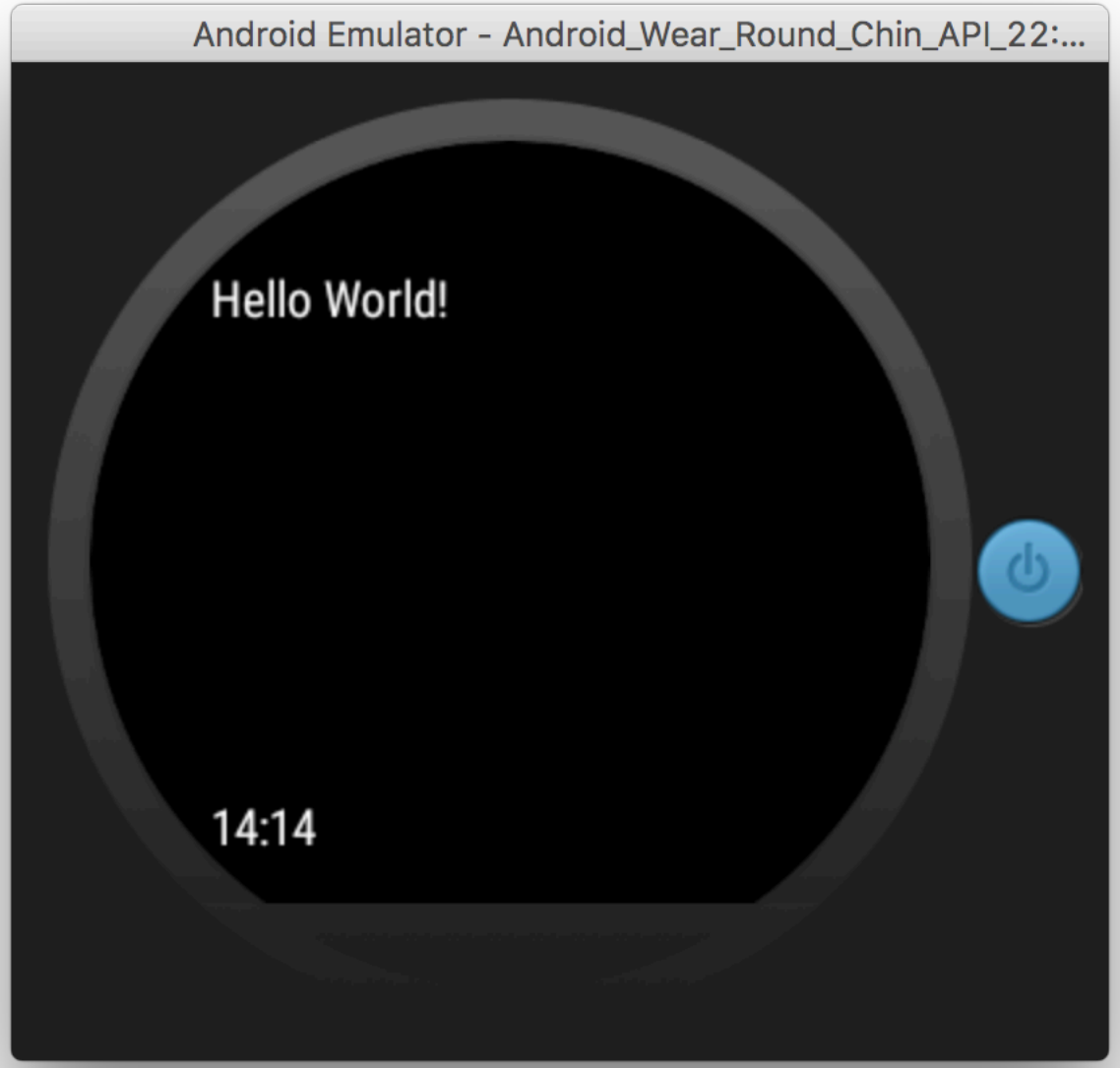


Running the Emulator




Select Deployment Target
Connected Devices
Android Wear Round Chin API 22 (Android 5.1.1, API 22)
Available Virtual Devices
Nexus 5 API 21 x86 (missing feature: WATCH)
Android Wear Round Chin API 20 (Failed to parse properties from /Users/michaelrohs/.andr
MonoForAndroid API 12 (minSdk(API 21) > deviceSdk(API 12))
MonoForAndroid API 10 (minSdk(API 21) > deviceSdk(API 10))
Create New Virtual Device
Don't see your device?
Use same selection for future launches
Cancel
OK


Android Emulator - Android_Wear_Round_Chin_API_22:...
Hello World!


Android Emulator - Android_Wear_Round_Chin_API_22:...
Hello World!
14:14

Layout Definition (activity_main.xml)

MainActivity.java | activity_main.xml

```
android.support.wearable.view.BoxInsetLayout
<?xml version="1.0" encoding="utf-8"?>
<android.support.wearable.view.BoxInsetLayout xmlns:android="http://
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="de.luh.hci.watchsensors.MainActivity"
    tools:deviceIds="wear">

    <TextView
        android:id="@+id/text"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Hello World!"
        app:layout_box="all" />

    <TextView
        android:id="@+id/clock"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|start"
        android:textColor="@android:color/white"
        app:layout_box="all" />

</android.support.wearable.view.BoxInsetLayout>
```

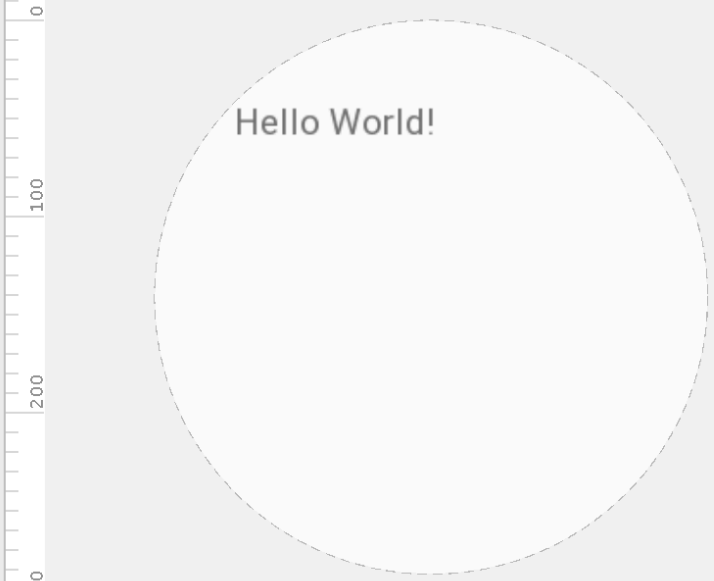
Design | Text

Preview

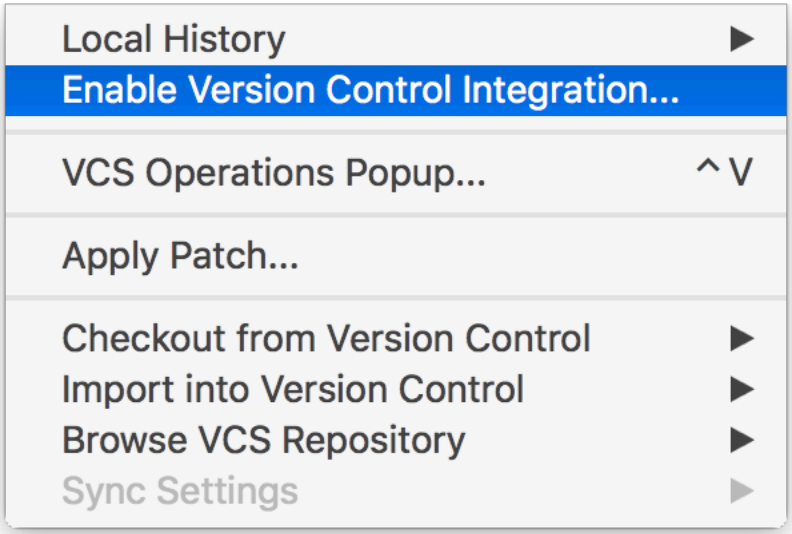
Palette

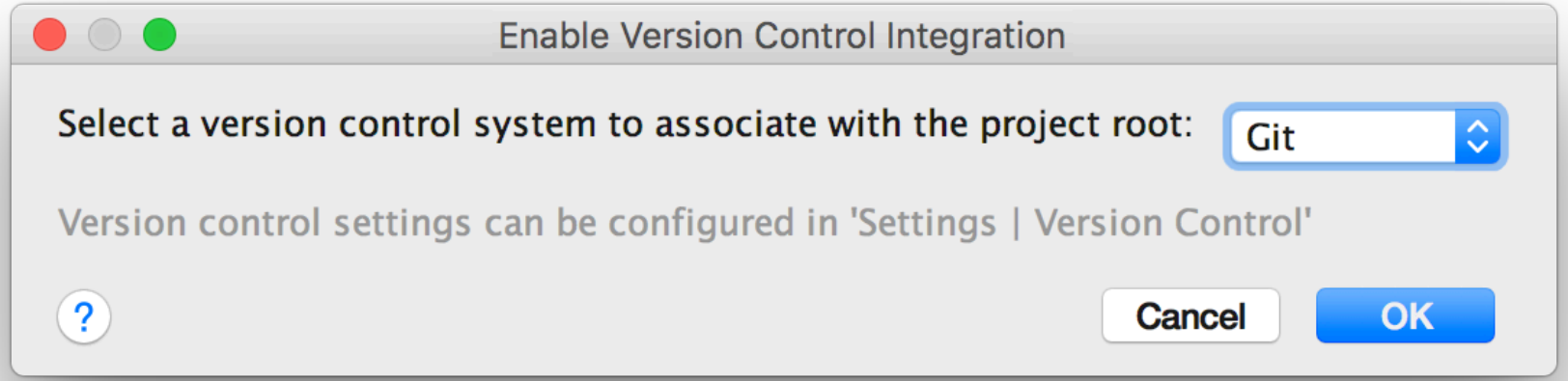
Wear Round | 25 | Light | Language

176%

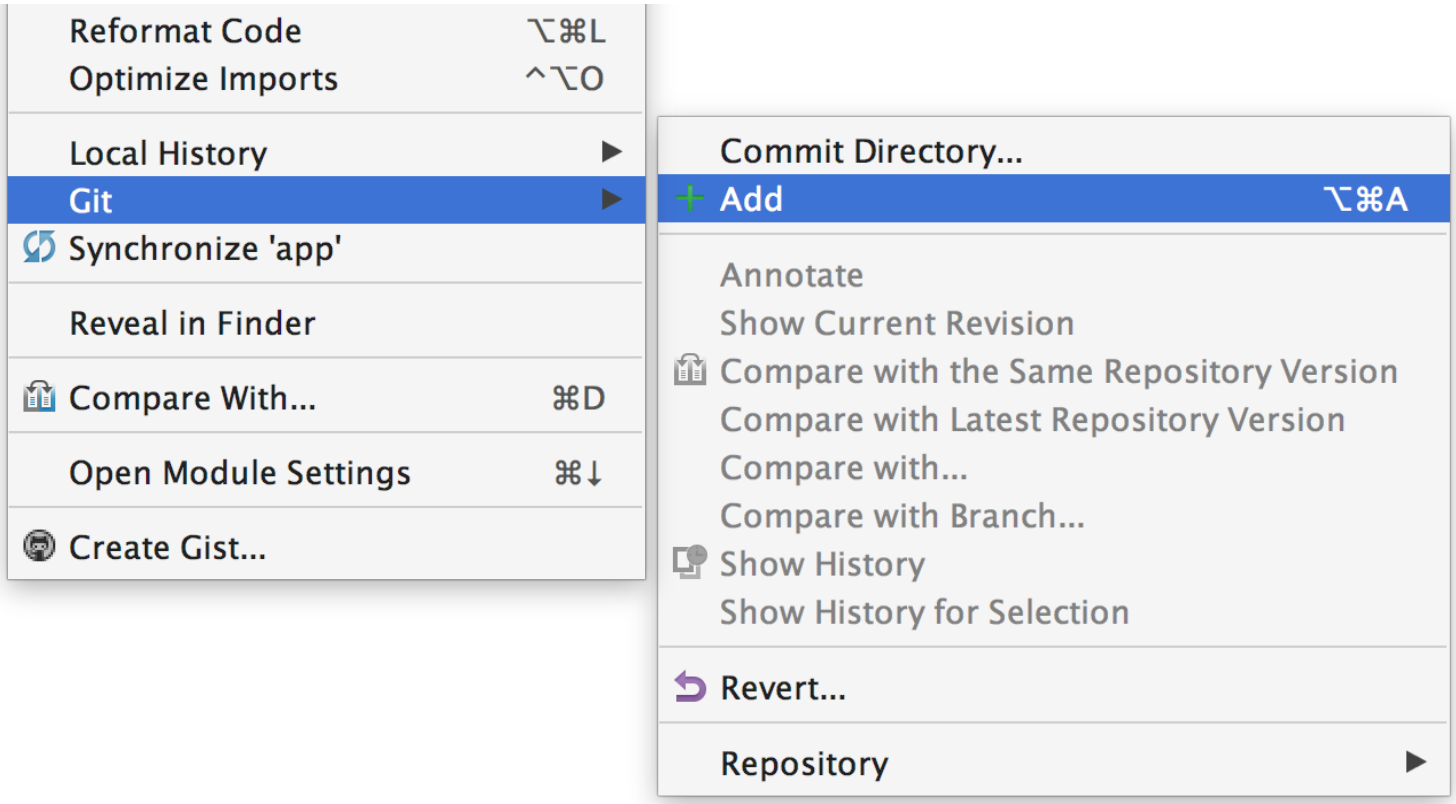


Use a Version Control System

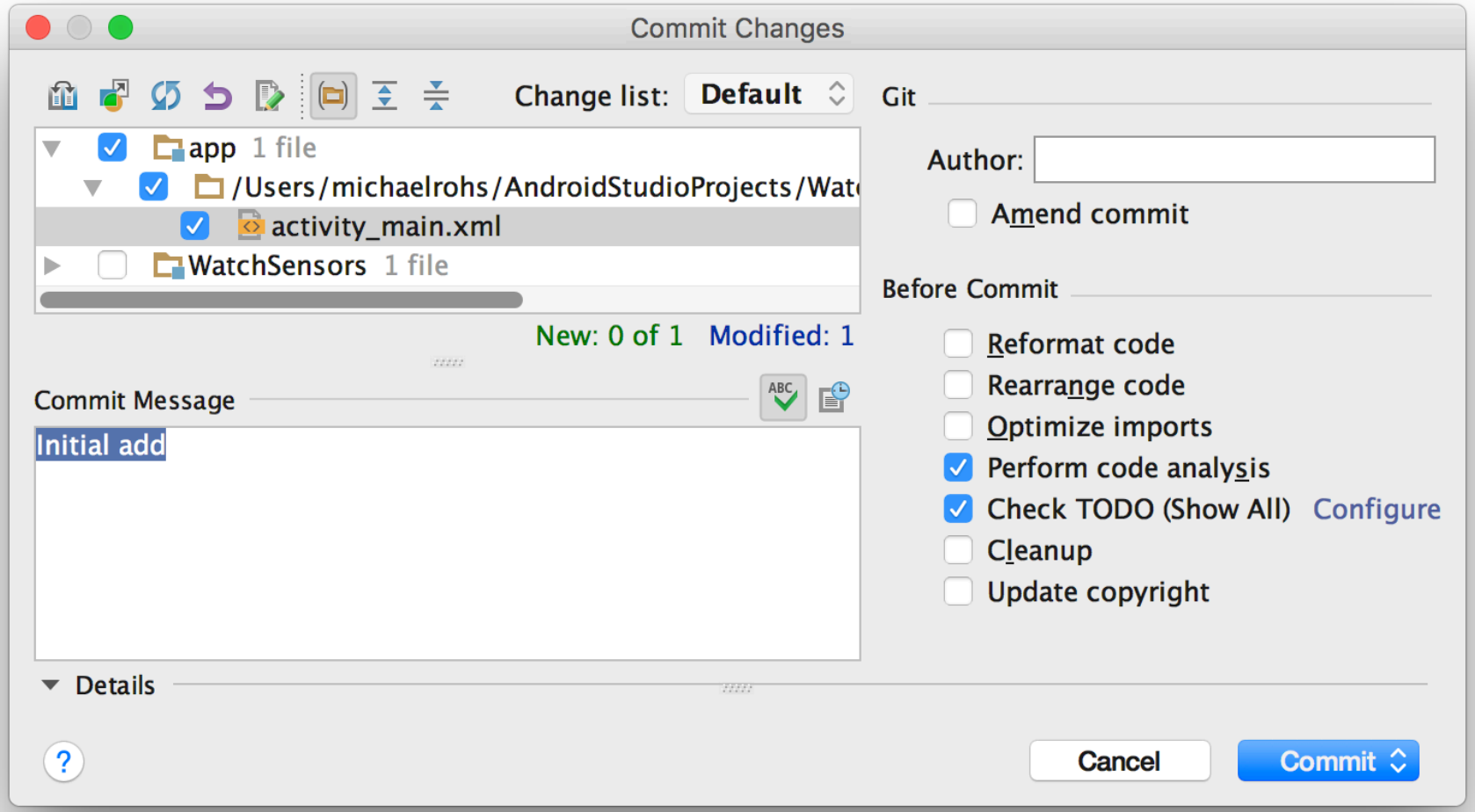




Use a Version Control System



Use a Version Control System

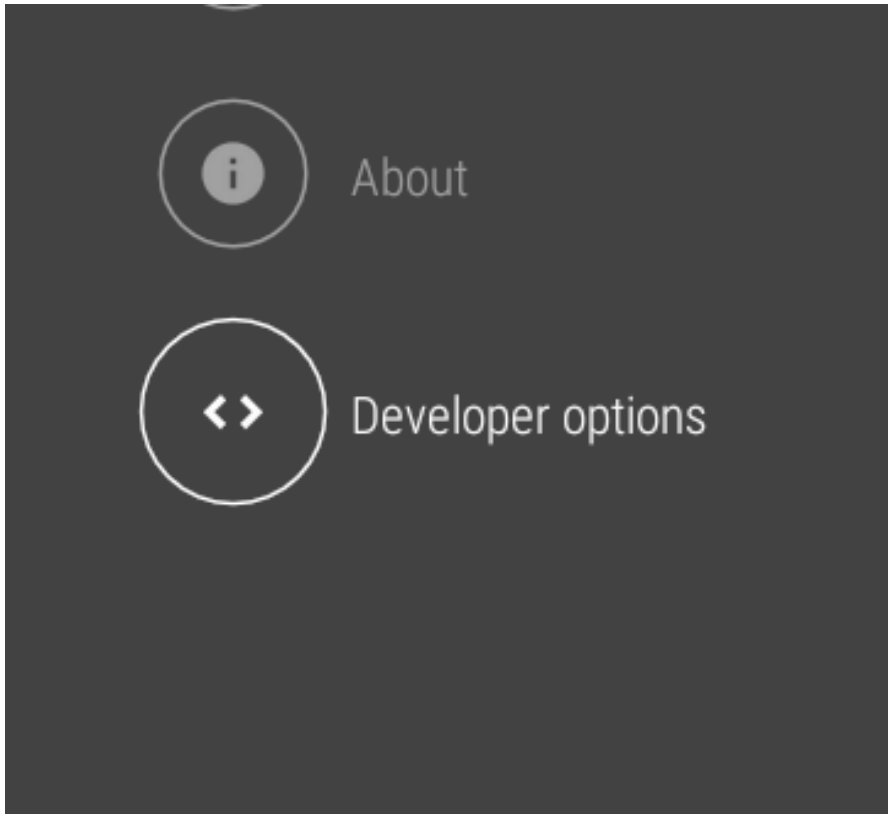


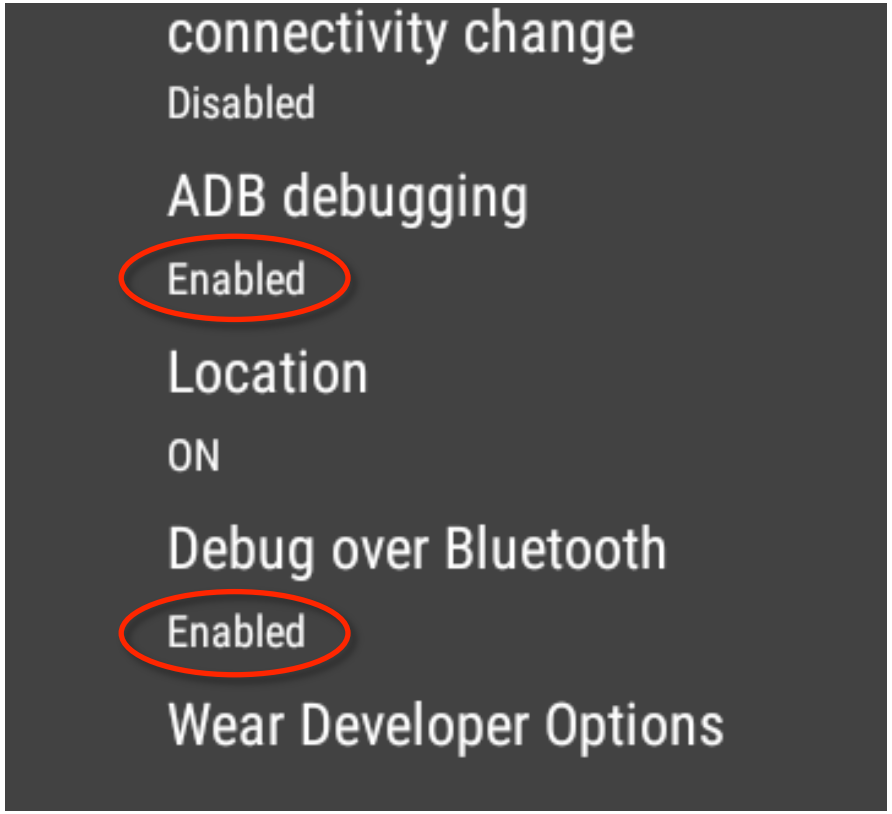
Installing on a Real Smartwatch

- Install “Android Wear” app on Android phone
- Enable Bluetooth, pair phone and watch
- Enable Bluetooth debugging on the watch
  - Open the Settings menu on the watch
  - Scroll to the bottom of the menu and tap About
  - Tap the build number 7 times
  - From the Settings menu, tap Developer Options
  - Enable ADB debugging
  - Enable Debug over Bluetooth

https://developer.android.com/training/wearables/apps/bt-debugging.html

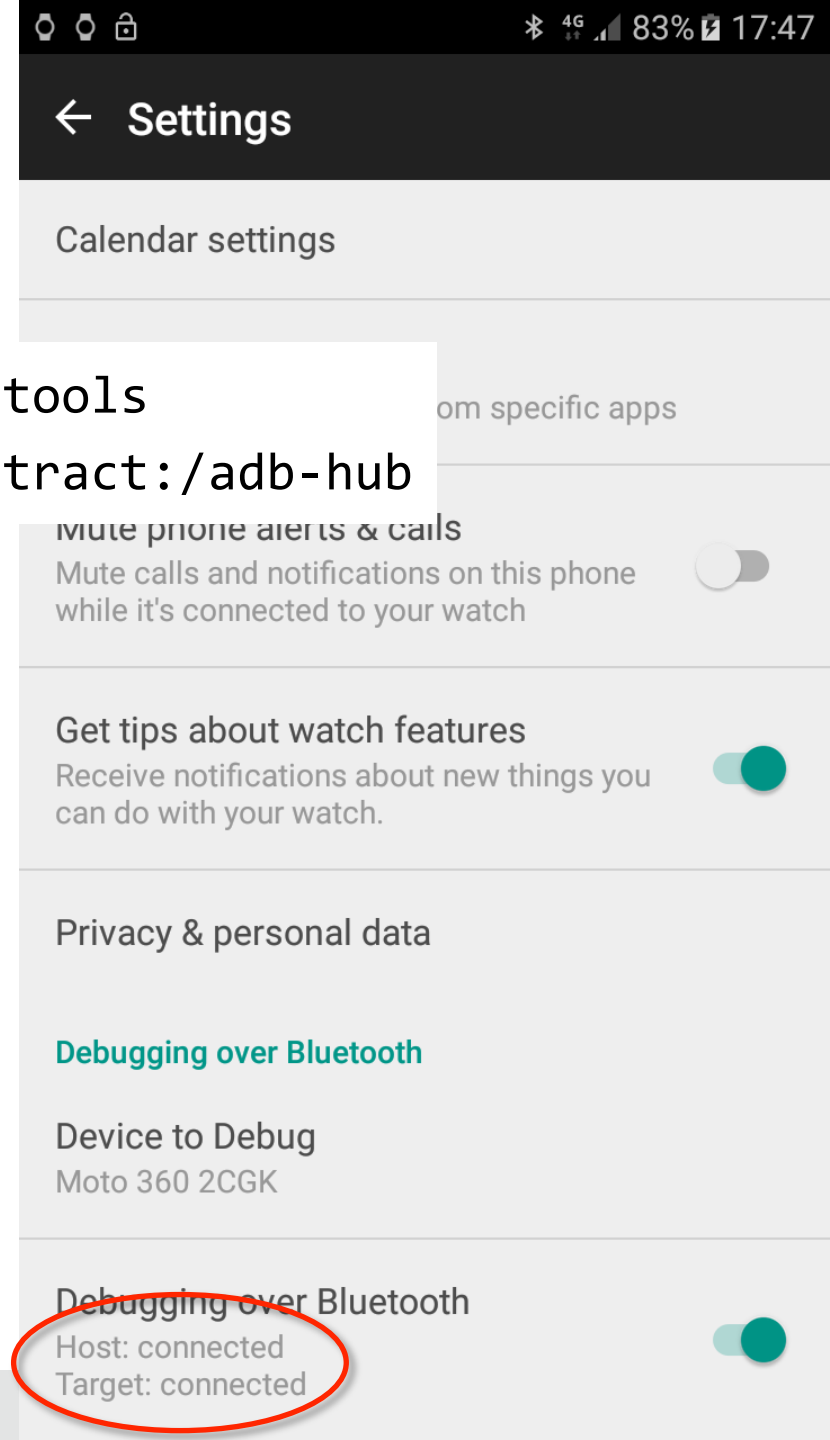
Enable Developer Options on Smartwatch



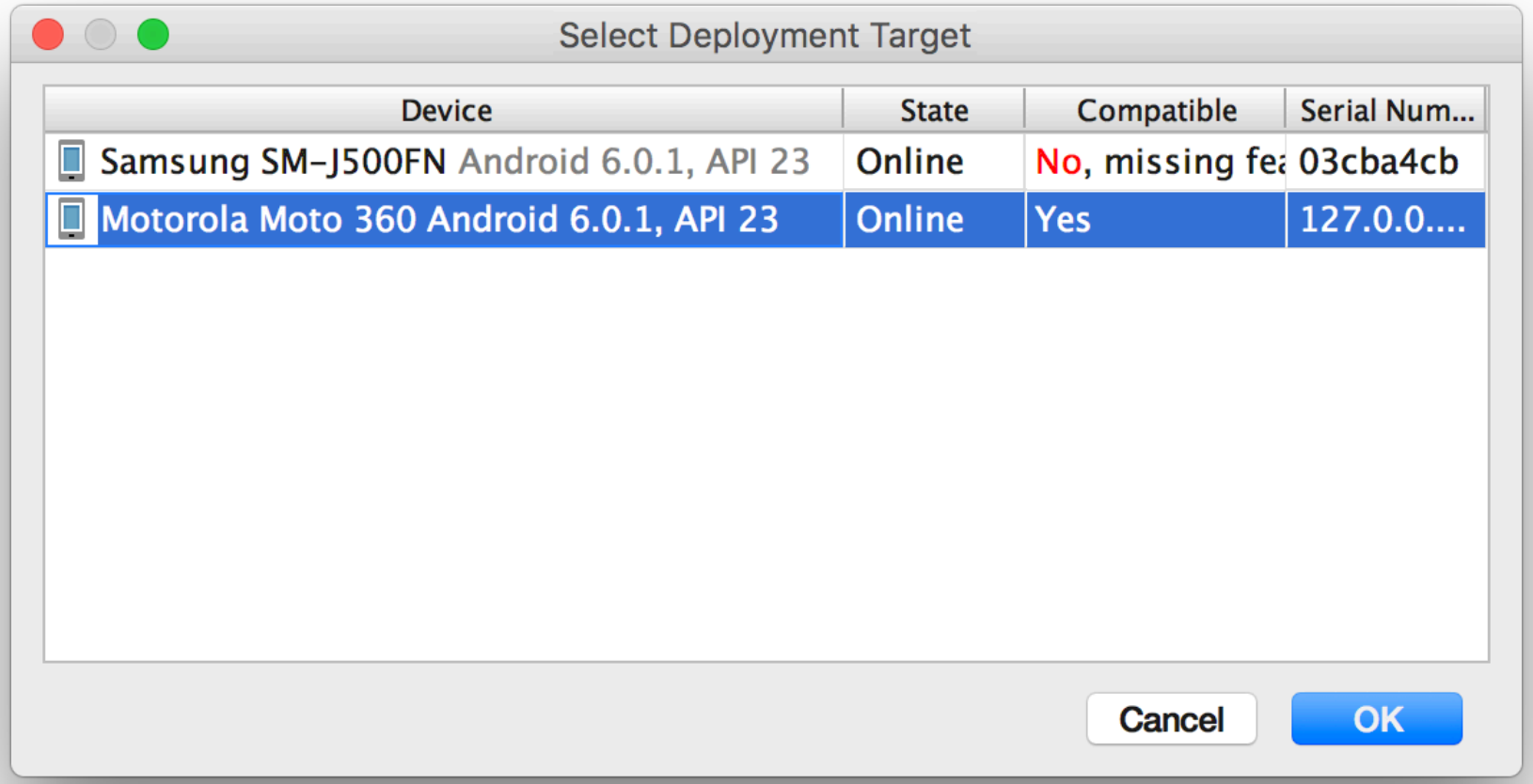


Android Debug Bridge

```
$ cd android-sdk-macosx/platform-tools
$ ./adb forward tcp:4444 localabstract:/adb-hub
$ ./adb connect 127.0.0.1:4444
connected to 127.0.0.1:4444
$ ./adb devices
List of devices attached
127.0.0.1:4444  device
03cba4cb        device
```



Installing on a Real Smartwatch



“Sensors” on Moto360

Name	Type	Range	Resolution	Power	Delay
Accelerometer Sensor	1	78.5	0.0098	0.45	20000
Step Counter Sensor	19	65535.0	1.0000	0.45	0
Wrist Tilt Sensor	26	1.0	1.0000	0.45	0
Gyro Sensor	4	34.9	0.0100	0.45	20000
Light Sensor	5	65535.0	1.0000	0.45	0
Game Rotation Vector Sensor	15	78.5	0.0098	0.45	20000
Wellness Passive Sensor	65538	65535.0	1.0000	0.45	0
Significant Motion Sensor	17	1.0	1.0000	0.45	-1
Detailed Step Counter Sensor	65537	65535.0	1.0000	0.45	0
User Profile Sensor	65539	0.0	0.0000	0.45	0
User Stride Factor Sensor	65546	0.0	0.0000	0.45	0
Gravity Sensor	9	78.5	0.0098	0.45	20000
Linear Acceleration Sensor	10	78.5	0.0098	0.45	20000

Enumerating the "Sensors" on Moto360

```
import android.hardware.Sensor;
import android.hardware.SensorManager;
import android.util.Log;
import java.util.List;
...
private SensorManager sensorManager;
...
sensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);

List<Sensor> sensorList = sensorManager.getSensorList(Sensor.TYPE_ALL);
for (Sensor sensor : sensorList) {
    Log.d("MainActivity", sensor.toString());
}
```

Registering for Sensor Events

```
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.widget.TextView;
...
public class MainActivity extends WearableActivity
                                implements SensorEventListener {
  private Sensor accSensor;
  private TextView accView;
  ...
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);
     setAmbientEnabled();
    accSensor = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
     sensorManager.registerListener(this, accSensor, 100 * 1000);
     accView = (TextView) findViewById(R.id.line1);
  }
```

Registering for Sensor Events

```
@Override
public void onSensorChanged(SensorEvent event) {
    Log.d("MainActivity", "onSensorChanged: " + event);
    if (event.sensor == accSensor) {
        float[] vs = event.values;
        accView.setText(String.format("a: %.3f, %.3f, %.3f", vs[0], vs[1], vs[2]));
    }
}

@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {
    Log.d("MainActivity", "onAccuracyChanged: " + sensor + ", " + accuracy);
}
```

Layout File (activity_main.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.wearable.view.BoxInsetLayout xmlns:android=...">
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_gravity="center">
        <TextView
            android:id="@+id/line1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="placeholder 1" />
        <TextView
            android:id="@+id/line2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="placeholder 2" />
    </LinearLayout>
</android.support.wearable.view.BoxInsetLayout>
```

BLUETOOTH LE COMMUNICATION

Bluetooth Low Energy (BLE)

- Ultra low power consumption (button cell batteries)
  - 1% to 50% of Bluetooth classic
  - Small, low-cost transceivers
- History
  - Originally Wibree (Nokia 2006)
  - Now part of Bluetooth 4.0 (2010)
- Applications / profiles
  - Smartwatches
  - Human-interface devices, e.g., wireless keyboards
  - Health care: blood pressure, thermometer, blood glucose
  - Sports and fitness: heart rate, running/cycling speed, weight scale
  - Environmental sensing, proximity sensing, beacons

http://www.bluetooth.com/Bluetooth/Products/low_energy.htm

Bluetooth Low Energy (BLE)

- Technology
  - 2.4 GHz ISM band (like classic Bluetooth)
  - 10–500 mW power consumption, 100+ meters range
  - 270 kbps application bit rate
  - 6 ms latency (from non-connected state)
- Generic Attribute Profile (GATT)
  - Specification for exchanging data attributes over a BLE link
    - Attributes identified by a UUID (128 bits)
  - Characteristic: A single value and "descriptors" that describe the value
    - Example characteristic: "heart rate measurement"
  - Service: A collection of "characteristics"
    - Example: "heart rate service"

https://www.bluetooth.com/specifications/adopted-specifications

BLE Device Roles

- Roles before connection
  - Central: scan, look for advertisements
    - Example: mobile phone
  - Peripheral: make advertisements
    - Example: heart rate monitor
- Roles after connection
  - GATT server (provides data)
    - Example: heart rate monitor
  - GATT client (receives data)
    - Example: mobile phone

https://developer.android.com/guide/topics/connectivity/bluetooth-le.html

Android BLE API

- since since Android 4.3, API Level 18
- BLE API changed with API Level 21
- specify Permissions and features AndroidManifest.xml

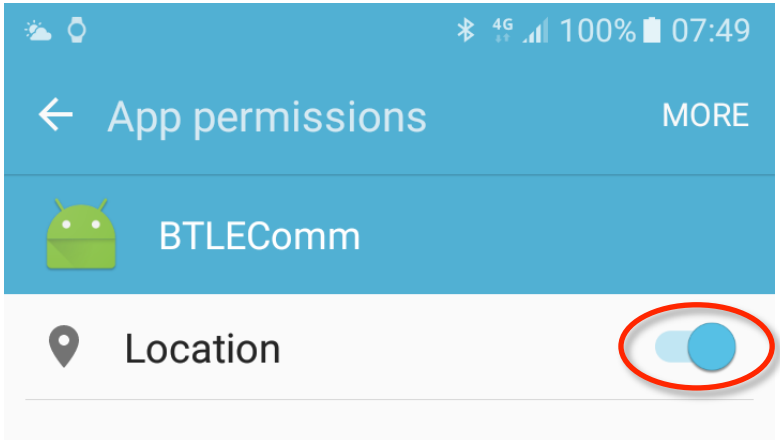
- Finding BLE Devices
- Communicating

https://developer.android.com/guide/topics/connectivity/bluetooth-le.html

Android BLE API – Permissions

- AndroidManifest.xml

```
<uses-permission android:name=
  "android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name=
  "android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name=
  "android.permission.BLUETOOTH"/>
<uses-permission android:name=
  "android.permission.BLUETOOTH_ADMIN"/>
<uses-feature android:name=
  "android.hardware.bluetooth_le"
   android:required="true"/>
```



https://developer.android.com/guide/topics/connectivity/bluetooth-le.html

Android BLE API – Finding Devices (new version with API 21)

```
bm = (BluetoothManager) getSystemService(Context.BLUETOOTH_SERVICE);
ba = bm.getAdapter();
bs = ba.getBluetoothLeScanner(); // API 21
...
Log.d("MainActivity", "version: " + android.os.Build.VERSION.SDK_INT);

handler = new Handler();
handler.postDelayed(new Runnable() {
        public void run() { ... }  }, 10000); // turn scanning off after 10 seconds

if (android.os.Build.VERSION.SDK_INT < 21) {
    ba.startLeScan(bleStartScan);
} else {
     bs.startScan(sc);
}
```

https://developer.android.com/guide/topics/connectivity/bluetooth-le.html

Android BLE API – Finding Devices (new version with API 21)

```
handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            Log.d("MainActivity", "run");
            if (android.os.Build.VERSION.SDK_INT < 21) {
                ba.stopLeScan(bleStopScan);
            } else {
                bs.stopScan(sc);
            }
        }
    }, 10000);
```

https://developer.android.com/guide/topics/connectivity/bluetooth-le.html

Android BLE API – Finding Devices (new version with API 21)

Old API:

```
private BluetoothAdapter.LeScanCallback bleStartScan =
        new BluetoothAdapter.LeScanCallback() {
            public void onLeScan(BluetoothDevice device, int rssi, byte[] scanRecord) {
                // process received device information
                Log.d("MainActivity", "bleStartScan: " + device);
            }
        };
private BluetoothAdapter.LeScanCallback bleStopScan =
        new BluetoothAdapter.LeScanCallback() {
            public void onLeScan(BluetoothDevice device, int rssi, byte[] scanRecord) {
                // process received device information
                Log.d("MainActivity", "bleStopScan: " + device);
            }
        };
```

https://developer.android.com/guide/topics/connectivity/bluetooth-le.html

Android BLE API – Finding Devices (new version with API 21)

New API:

```
private ScanCallback sc = new ScanCallback() {
    public void onScanResult(int callbackType, ScanResult result) {
        super.onScanResult(callbackType, result);
        Log.d("MainActivity", "onScanResult: " + result);
    }
    public void onBatchScanResults(List<ScanResult> results) {
        super.onBatchScanResults(results);
        Log.d("MainActivity", "onBatchScanResults: " + results.size());
    }
    public void onScanFailed(int errorCode) {
        super.onScanFailed(errorCode);
        Log.d("MainActivity", "onScanFailed: " + errorCode);
    }
};
```

https://developer.android.com/guide/topics/connectivity/bluetooth-le.html

Android BLE API – Connecting to Services

- Connect to a server
  - gatt = device.connectGatt(this, false/*autoConnect*/, gattCallback);
- GATT callback

```
BluetoothGattCallback gattCallback = new BluetoothGattCallback() {
    public void onConnectionStateChange(BluetoothGatt gatt,
            int status,
            int newState) { ... }
    public void onServicesDiscovered(BluetoothGatt gatt, int status) {...}
    public void onCharacteristicRead(BluetoothGatt gatt,
            BluetoothGattCharacteristic characteristic,
            int status) { ... }
};
```

getUUID(), getIntValue(...)

https://developer.android.com/guide/topics/connectivity/bluetooth-le.html

Android BLE API – Connecting to Services

- Reading characteristics

```
List<BluetoothGattCharacteristic> gattCharacteristics =
        gattService.getCharacteristics();
```

- Receiving notifications
  - Notification when a characteristic changes

```
public void onCharacteristicChanged(BluetoothGatt gatt,
        BluetoothGattCharacteristic characteristic) {
    ...
}
```

https://developer.android.com/guide/topics/connectivity/bluetooth-le.html

To-Dos: Wiki, Development Environment

- Create group page (wiki.hci..., ldap.hci...)
- Describe the idea (1-2 paragraphs)
- Describe the planned scenario
- Describe the planned interaction technique
- What mobile / device features does it use?

- Install Android and libGDX
- Get familiar with Android and libGDX

- Develop a test app that shows sensor data (on the watch)
- Develop an app that sends sensor data to another device