

# WEARIT™



## WEARIT DEVELOPER DOCUMENTATION 0.2 preliminary release July 20<sup>th</sup>, 2013

## ■ Index

Terms of Use	<a href="#">page 3</a>
1 WEARIT SDK	<a href="#">page 4</a>
2 Platform	<a href="#">page 5</a>
3 Configure the Environment	<a href="#">page 7</a>
4 Setup the Emulator	<a href="#">page 9</a>
5 Code the Application	<a href="#">page 13</a>
6 Sensors	<a href="#">page 14</a>
6.1 Accelerometer	<a href="#">page 14</a>
6.2 Magnetometer	<a href="#">page 14</a>
6.3 ANT+	<a href="#">page 14</a>
6.4 Pedometer	<a href="#">page 14</a>
6.5 GPS	<a href="#">page 18</a>
6.6 Buzzer	<a href="#">page 19</a>
6.7 Buttons	<a href="#">page 20</a>

## ■ Terms of Use

### Copyright Information and Usage Notice

The information disclosed herein is property of Si14 S.p.A. No part of this publication may be reproduced or transmitted in any form or by any means including electronic storage, reproduction, execution or transmission without the prior written consent of Si14 S.p.A. The recipient of this document by its retention and use agrees to respect the copyright of the information contained herein. The information contained in this document is subject to change without notice and should not be construed as a commitment by Si14 S.p.A.

“Android” and the “Android logo” are trademarks or registered trademarks of Google Inc.

“ANT+” is trademarks or registered trademarks of Dynastream Innovations Inc.

“Bluetooth” is trademarks or registered trademarks of Bluetooth SIG.

“WiFi” is trademarks or registered trademarks of Wi-Fi Alliance

*If you have received this preliminary document it means that you have been selected to join the Alpha WearIT Developer Program.*

*As the WearIT Developer Team, we want to thank you since we really rely on your co-operation and your feedback to improve the product and to ensure the best experience for the users and the developers.*

*Since this is a work-in-progress version we ask you to not disclose this document nor its content to third-parties.*

©2013 Si14 S.p.A. All Rights Reserved.

## ■ WEARIT SDK

Hello and thank you for your interest about the WEARIT platform.

WEARIT is a watch designed for sport, health & wellness monitoring applications. WEARIT OS is based on Android™ and its APIs are compliant with Android 4.1.2.

Within this guide you will find instructions on how to start developing your first WEARIT application starting from the Android SDK.

Knowledge of the Android framework and the Android application lifecycle is essential before you start writing WEARIT Android applications. For more informations about Android development refer to the official documentation.

A good starting place is the “Android Developers Guide”:

<http://developer.android.com/guide>



## ■ The Platform

The WEARIT Operating System is based on Android 4.1.2. When possible, the standard Android APIs are used.

The following is a list of the main WEARIT Platform Features:

<b>OS</b>	Android Version 4.1.2
<b>CPU</b>	TI Cortex A8
<b>RAM</b>	256 MB
<b>Screen Resolution</b>	240x240 MDPI
<b>Screen Size</b>	1.5"
<b>Touchscreen</b>	Multitouch capacitive sensor
<b>Connectivity</b>	WiFi™ with “WiFi Direct” support.
	Bluetooth™ with support of A2DP AVRCP TEHTERING
	Bluetooth™ LE, not currently available, software support will be available in future OS releases
	ANT+™
<b>Sensors</b>	Accelerometer
	GPS
	Magnetometer
	Hardware Pedometer
<b>Storage</b>	4GB eMMC
	<ul style="list-style-type: none"> <li>• 700 MB available for the applications</li> <li>• 2.7 GB available as SDCard storage</li> </ul>
<b>Buttons</b>	3 Hardware Buttons
<b>Power Supply</b>	Charging clip
<b>USB</b>	USB connection through charging clip
<b>Audio notifications</b>	Buzzer

In order to access the non-standard features of the device, two java libraries are available:

- The PedometerSystemService: a library to access the hardware pedometer informations;
- The BuzzerSystemService: a library to command the integrated buzzer.

All the other features are available using their corresponding standard Android APIs.

## ■ Configure the Environment

To build your first WEARIT application you need to get the Android SDK from the Android developers website.

Download the ADT Bundle with Eclipse and the ADT Plugin from:

<http://developer.android.com/sdk/index.html>

unzip the package and follow the Android SDK documentation.

### Get the Android SDK

The Android SDK provides you the API libraries and developer tools necessary to build, test, and debug apps for Android.

If you're a new Android developer, we recommend you download the ADT Bundle to quickly start developing apps. It includes the essential Android SDK components and a version of the Eclipse IDE with built-in **ADT (Android Developer Tools)** to streamline your Android app development.

With a single download, the ADT Bundle includes everything you need to begin developing apps:

- Eclipse + ADT plugin
- Android SDK Tools
- Android Platform-tools
- The latest Android platform
- The latest Android system image for the emulator

If you prefer to use an existing version of Eclipse or another IDE, you can instead take a more customized approach to installing the Android SDK. See the following instructions.

▼ **USE AN EXISTING IDE**

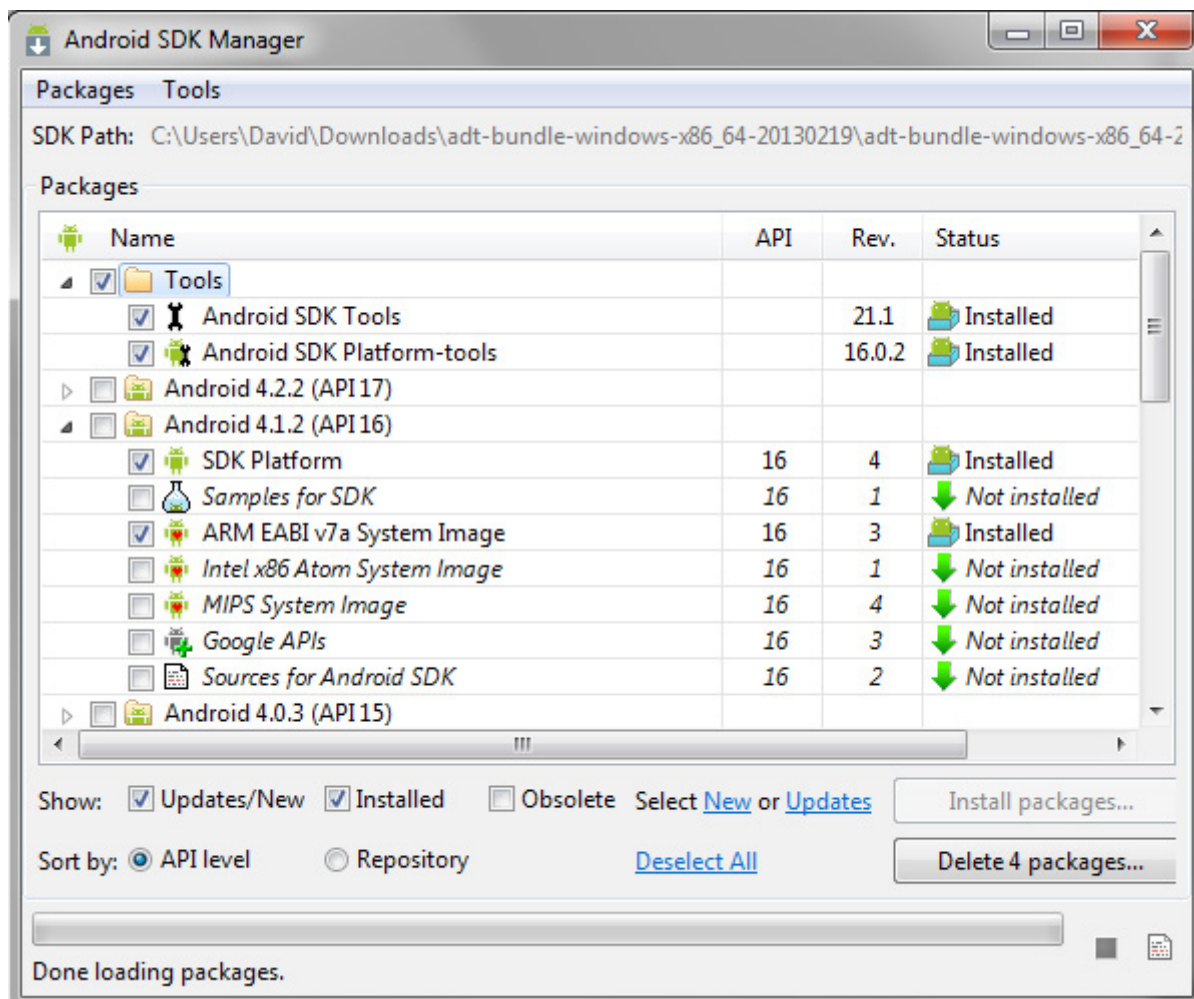
▼ **SYSTEM REQUIREMENTS**

▼ **DOWNLOAD FOR OTHER PLATFORMS**



**Download the SDK**  
ADT Bundle for Windows

Launch the “**SDK Manager**” application and, inside the “**Packages**” section, select “**SDK Platform for Android 4.1.2**” and “**ARM EABI v7a System Image for Android 4.1**” to be installed.

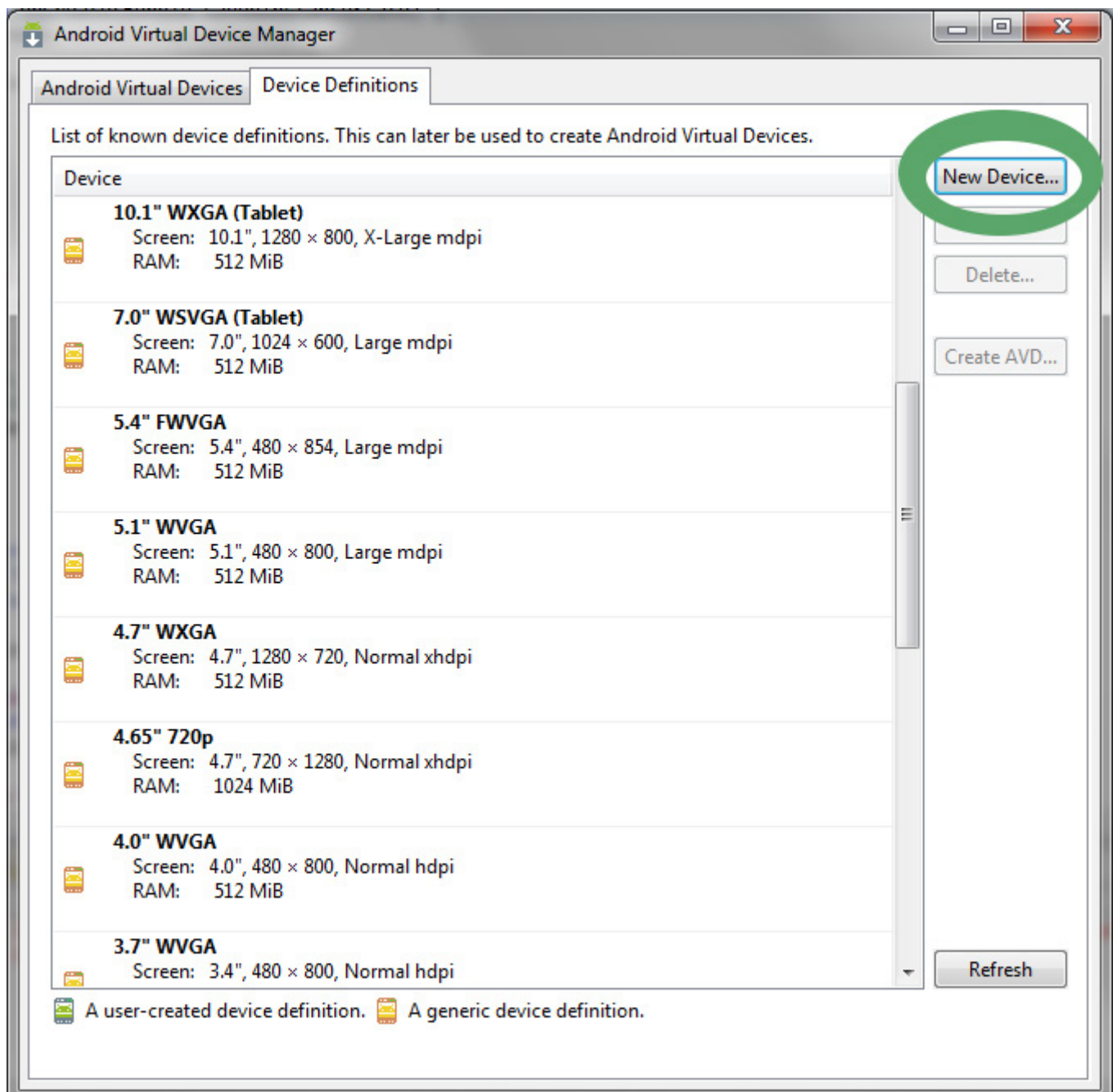




## ■ Setup the Emulator

To test your application for **WEARIT** you need to setup the emulator to emulate the **WEARIT OS**. **WEARIT** screen resolution is “**240x240 MDPI**”. Android does not support square screen so setup your application for a screen sized 240x241 and remember that the last line will be not visible on the device screen.

Create a new device definition in Android Virtual Manager as depicted in the following screenshots.



Under the “**Device Definitions**” section of the “**Android Virtual Device Manager**” application, press the “**New Device**” button.

**Edit Device**

Name:

Screen Size (in):

Resolution (px):  x

Sensors: ☒ Accelerometer ☐ Gyroscope  
☒ GPS ☐ Proximity Sensor

Cameras: ☐ Front ☐ Rear

Input: ☐ Keyboard  
☒ No Nav ☐ DPad ☐ Trackball

RAM:

Size:

Screen Ratio:

Density:

Buttons:

Device States:

Portrait: ☒ Enabled ☐ Navigation

Landscape: ☒ Enabled ☐ Navigation

Portrait with keyboard: ☒ Enabled ☒ Navigation

Landscape with keyboard: ☒ Enabled ☒ Navigation

☒ Override the existing device with the same name

Setup the Virtual Device entering the following configuration values:

**Name:** WearITSimulator

**Screen Size:** 1.5 in

**Resolution:** 240x241 px

Under the “**Sensors**” section, ensure that only Accelerometer and **GPS** boxes are checked

Under the “**Cameras**” section, ensure that there is no box checked

Under the “**Input**” section, select “**No Nav**”. The Keyboard box should be unchecked.

**RAM:** 256 Mib

**Size:** Normal


**Screen Ratio:** notlong

**Density:** mdpi

**Buttons:** Hardware

**Device States:** “Portrait” and “Landscape” boxes should be checked.

Once you have entered these values, press the “**Edit Device**” button.

 **Edit Android Virtual Device (AVD)**

AVD Name:

Device:

wearit (240 × 241: mdpi)

Target:

Android 4.1.2 - API Level 16

CPU/ABI:

ARM (armeabi-v7a)

Keyboard:

☐ Hardware keyboard present

Skin:

☒ Display a skin with hardware controls

Front Camera:

None

Back Camera:

None

Memory Options:

RAM:

VM Heap:

Internal Storage:

MiB

SD Card:

☒ Size:

MiB

☐ File:

Browse...

Emulation Options:

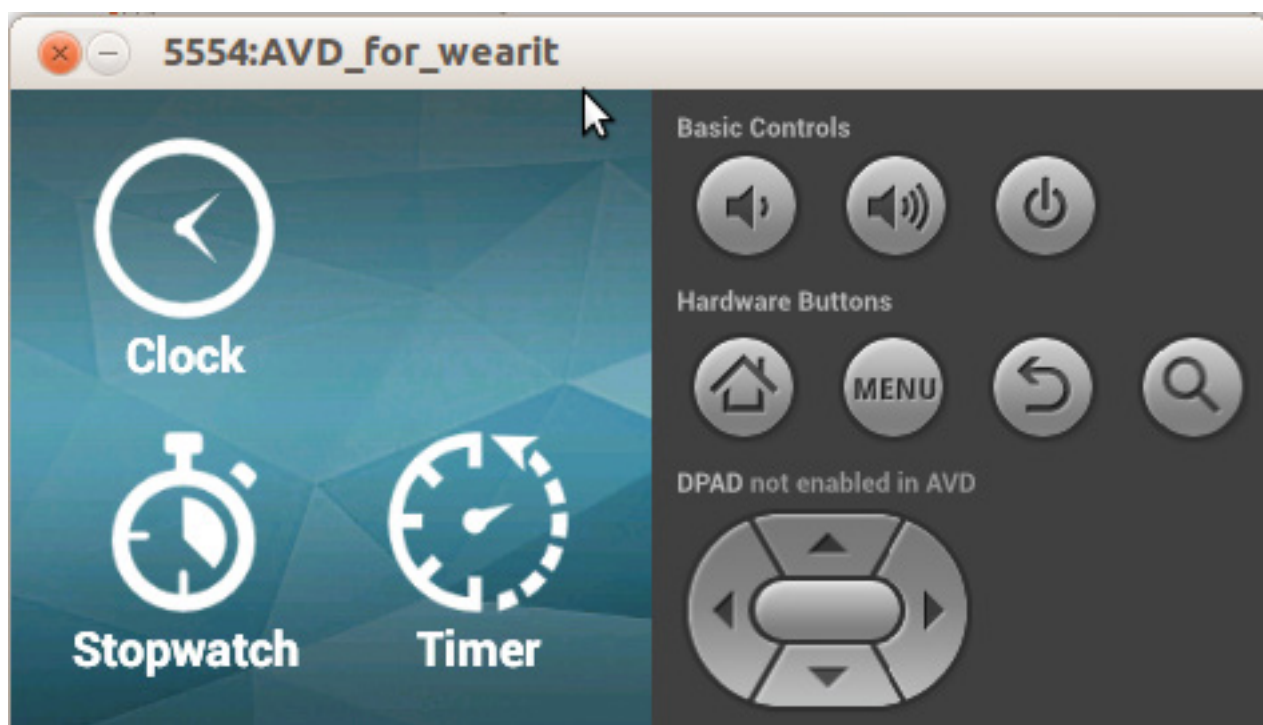
☐ Snapshot

☐ Use Host GPU

☐ Override the existing AVD with the same name

Cancel

OK

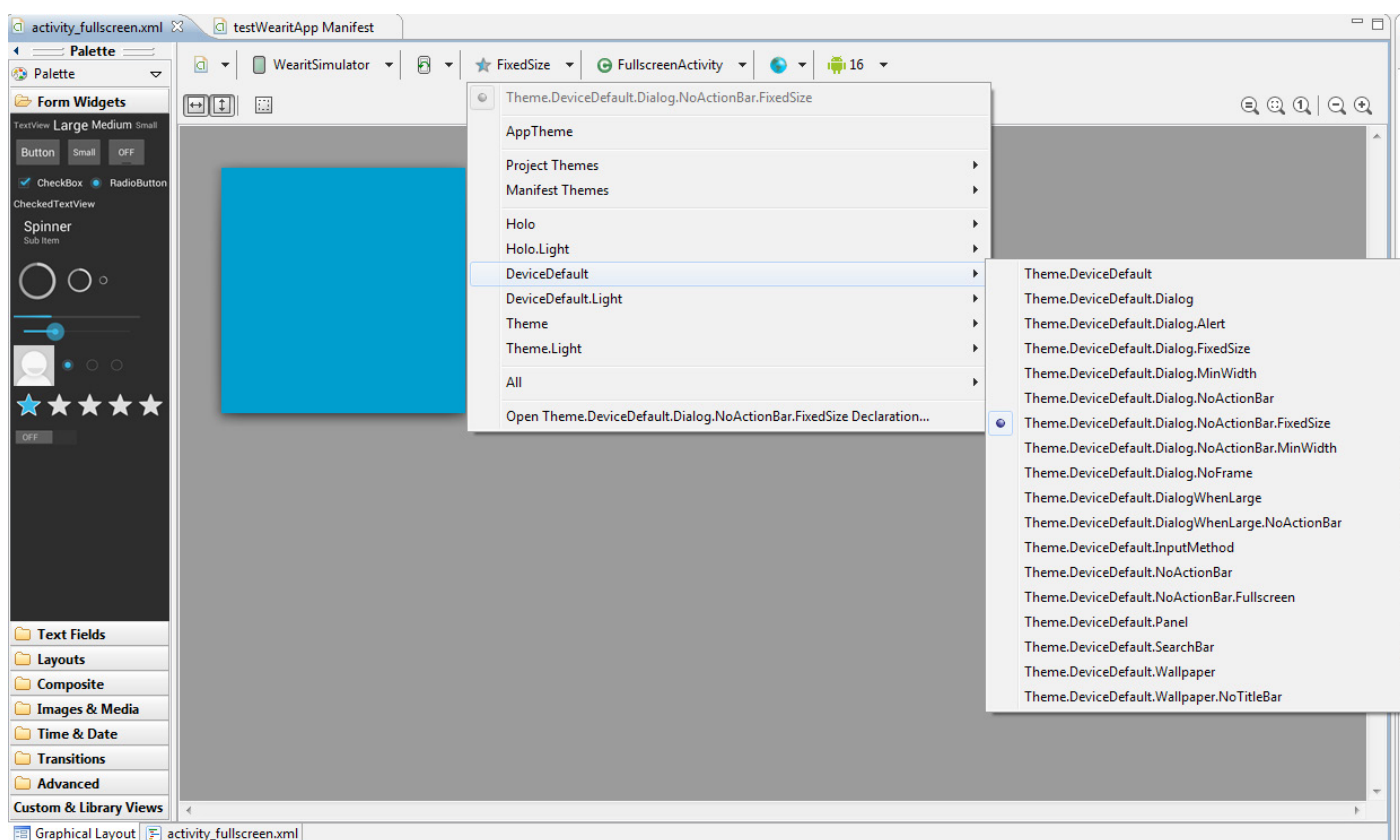


## ■ Code the Application

Create your application like a standard Android App.

While developing, take into consideration the **following notes**:

- By default, activities are launched in full-screen mode
- The Android **status bar** is only showed when the Launcher is in the resumed state, i.e. visible to the user. For all the “standard” activity launched by the user the status bar is hidden by the system. This behavior differs from the standard Android behavior. To simulate it in the emulator you can set the following attribute to the <activity> or <application> element in your Android manifest:  
`android:theme= "@android:style/Theme.NoTitleBar.Fullscreen"`
- Screen resolution is 240x240, ensure that your application layouts are optimized for this resolution



## ■ Sensors

### Accelerometer

Accelerometer measures the acceleration force (in m/s<sup>2</sup>) that is applied to the device on all three physical axes (x, y, and z), including the gravitational force.

To use the accelerometer refer to Android Sensor Manager API:

<http://developer.android.com/reference/android/hardware/SensorManager.html>

### Magnetometer

A magnetometer is a measuring instrument used to measure the strength and the direction of the magnetic field, for all three physical axes (x, y, z) in  $\mu\text{T}$ .

To use magnetometer refer to Android Sensor Manager API:

<http://developer.android.com/reference/android/hardware/SensorManager.html>

### ANT+

ANT+ is a wireless communications used to connect WEARIT with ANT+ compatible accessories like heart rate monitors, speed sensors, weight measuring devices etc.

To use the ANT+ integrated radio see documentation at <http://www.thisisant.com/developer/>

Download the ANT SDK Package and follow the documentation and the examples.

To use the emulator in combination with an ant USB key you can use the “ANT Android Emulator Bridge Tool” and follow the documentation as you would do with any Android device.

### Pedometer

WEARIT includes a hardware pedometer integrated in the accelerometer device.

This sensor is able to measure steps, speed and calories independently from the OS, this allows it to collect data also when the system is in stand-by mode.

Use of Pedometer is supported on WEARIT SDK through a system service and a dedicated library.

To download the library and the documentation please refer to the WEARIT developer section on the website (still under development at the moment of writing).

The Pedometer system service is an Android remote service implemented through .aidl files.

The interface is described in two files:

### IONPedometerChangedListener.aidl

```

/**
 * Interface used to receive notification from the remote
 * pedometer Service.
 */
interface IONPedometerChangedListener
{
    /**
     * Called whenever one of the values read from the pedometer's value
     changes.
     * @param steps      Absolute step count.
     * @param speed       Actual estimated speed      [m/h].
     * @param calories    Estimated calories consumption [cal].
     * @param distance    Estimated distance traveled  [m].
     * @param runStyle    current kind of running, can be one of 0: un-
     known, 1: rest, 2: walking, 3: jogging
     */
    void onPedometerChanged(float steps, float speed/*[m/h]*/, float
    calories/*[cal]*/, float distance/*[m]*/, int runStyle);
}

```

### IPedometerService.aidl

```

/**
 * IPedometerService is the remote interface to the WearIT pedometer
 * system service.
 */
interface IPedometerService
{
    /**
     * Use this function to read the pedometer gender.
     * Returns the gender, 0 female, 1 male
     */
    int getGender();
    /**
     * Use this function to configure the pedometer gender.
     * @param gender      The sex, 0 female, 1 male
     */
}

```



```

void setGender(int gender);
/**
 * Use this function to read the pedometer height.
 * @param height The person height [cm] (from 100 to 225)
 */
int getHeight();
/**
 * Use this function to configure the pedometer height.
 * Returns the person height in [cm] (from 100 to 225)
 */
void setHeight(int height);
/**
 * Use this function to read the pedometer weight.
 * Returns the weight in [Kg] (from 35 to 150)
 */
int getWeight();
/**
 * Use this function to configure the pedometer weight.
 * @param weight The person weight [Kg] (from 35 to 150)
 */
void setWeight(int weight);
/**
 * Use this function to configure the pedometer service.
 * @param sex    The sex, 0 female, 1 male
 * @param height The person height [cm] (from 100 to 225)
 * @param weight The person weight [Kg] (from 35 to 150)
 */
void setConfiguration(int sex, int height, int weight);
/**
 * Use this function to reset the pedometer values.
 */
void reset();
/**
 * Call this function to register your local  {@link IOnPedometer-
ChangedListener}
 * to the remote pedometer service.
 *
 * @param onPedometerChangeListener The {@link IOnPedometer-
ChangedListener}
 * you want to register
 */

```



```

    void registerPedometerChangedListener(in IOnPedometerChangedListen-
er onPedometerChangedListener);
    /**
     * Call this function to unregister your local {@link IOnPedometer-
    ChangedListener}
     * to the remote pedometer service.
     *
     * @param onPedometerChangedListener The {@link IOnPedometer-
    ChangedListener}
     * you want to unregister
     */
    void unregisterPedometerChangedListener(in IOnPedometerChangedLis-
tener onPedometerChangedListener);
}

```

For informations on how to use an Android remote service, please refer to the services section of the Android official documentation:

<http://developer.android.com/guide/components/services.html>

`IOnPedometerChangedListener.Stub` can be extended to receive value change notifications and the Android `ServiceConnection` is implemented to connect with the remote pedometer service. The following is a code extract (see the download section for a complete example):

```

private static class PedometerCallback extends IOnPedometerChangedLis-
tener.Stub
{
    public PedometerCallback()
    {}
    @Override
    public void onPedometerChanged(float steps, float speed, float calories,
float distance, int runStyle) throws RemoteException
    {
        // Use the values ...
    }

    private class PedometerConnection implements ServiceConnection
    {
        @Override
        public void onServiceConnected(ComponentName name, IBinder service)
        {
            mPedometer = IPedometerService.Stub.asInterface(service);
            try
            {
                sendConfig();
            }
        }
    }
}

```

```

        mPedometer.registerPedometerChangeListener(mPedometerListener);
    }
    catch (RemoteException e)
    {
    }

    @Override
    public void onServiceDisconnected(ComponentName name)
    {
        if (mPedometer != null)
        {
            try
            {
                mPedometer.unregisterPedometerChangeListener(mPedometerListener);
            }
            catch (RemoteException e)
            {
            }
            mPedometer = null;
        }
    }
}

```

## GPS

**WEARIT** integrate a High Performance GPS location sensor.

To use GPS refer to **Android LocationManager API**:

<http://developer.android.com/reference/android/location/LocationManager.html>

Use of particular GPS settings to enable low power consumption and fine tune the GPS behavior will be updated soon on SDK.

In order to test the GPS with the emulator you could find helpful the following open source project:

<https://code.google.com/p/android-gps-emulator/>

## Buzzer

Use of Buzzer is supported on **WEARIT** through a system service and a dedicated library, called BuzzerSystemLib. We also released a client library, the BuzzerSystemClient, that allows you to directly use the BuzzerSystemLib writing a few lines of code.

The client exposes only two functions:

```
/**
 * Call this function to play a wave at the given frequency.
 * @param frequency The frequency to play, in Hz.
 * @param durationMillis For how long this frequency must be hold,
in
 *                      milli seconds.
 */
public void ring(int frequency, int durationMillis)

/**
 * Use this function to configure the buzzer volume.
 * @param volume      The volume. Can be one of:
 *  {@link BuzzerService.VOLUME_MUTE}
 *  {@link BuzzerService.VOLUME_MIN}
 *  {@link BuzzerService.VOLUME_MEDIUM}
 *  {@link BuzzerService.VOLUME_MAX}
 */
public void setVolume(int volume)
```

In order to use the client in you Activity or Service use:

```
...
BuzzerClient mBuzzer;
...
@Override
public void onCreate(Bundle savedInstanceState)
{
    ...
    mBuzzer = new BuzzerClient(this);
    mBuzzer.onCreate();
    ...
}
```

and

```
@Override
public void onDestroy()
{
```

```
...
mBuzzer = new BuzzerClient(this);
mBuzzer.onDestroy();
...
}
```

to play the Buzzer simply call the ring method:

```
mBuzzer.ring(myFrequencyHz, myDurationMillis);
```

and to change the volume use the setVolume method (note that this changes the duty cycle of the buzzer, and then the volume control has a coarse granularity):

```
mBuzzer.setVolume(BuzzerService.VOLUME_MIN);
```

## Buttons

**WEARIT** has 3 hardware buttons:

- The central one is used as power button and is mapped to the “Home key” signal (KEYCODE\_HOME)
- The button on the top sends “Back key” signal (KEYCODE\_BACK)
- The button on the bottom sends the “Menu key” signal (KEYCODE\_MENU)

You can override the KEYCODE\_MENU and KEYCODE\_BACK in your application to enhance its usability.

Pressing for 8 seconds the central button allows to force the device reboot.