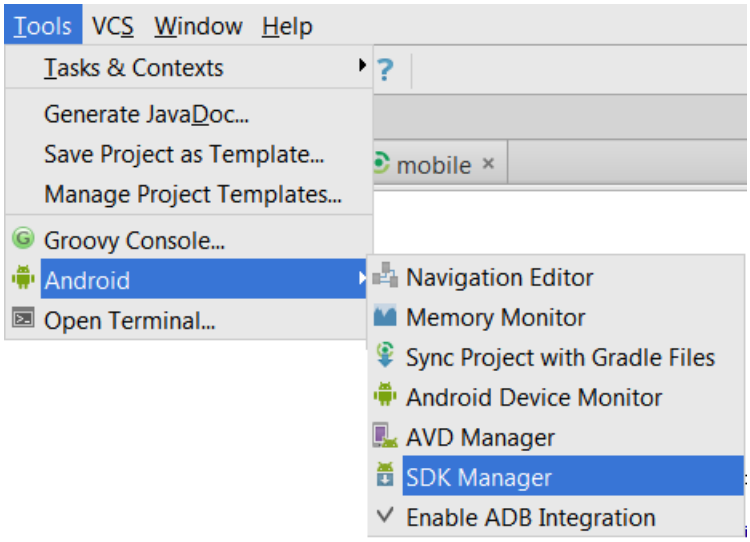
Android Wear Codelab

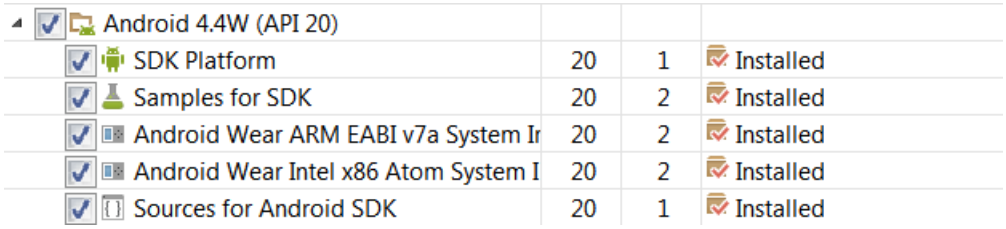
This document provides brief instructions on setting up your machine and getting things in place for the Android Wear Codelab.

Dependencies and Prerequisites

1. You have installed Android Studio 0.8.x . You can install the latest from here: http://tools.android.com/download/studio/canary/0-8-14
2. Ensure that you have the Android 4.4W - API Level 20 SDK downloaded and available on your machine. If you are not sure of how to do that, follow these steps:
    a. Launch Android Studio.
    b. Go to Tools -> Android -> SDK Manager from the main menu as shown below:



    c. Ensure that Android 4.4W – API Level 20 SDK is installed on your machine. Be patient while you download since it can take a while.



3. For the code lab, we plan to use a real Android phone and pair it with an Android Wear Emulator. What this means is that you need to ensure that your Android phone can be connected to your development machine and has the right drivers in place. Once you connect your Android phone to your machine, just ensure that you can see it via the **adb devices** command.

4. Next thing is to visit the Play Store on your Android phone and download the official Android Wear App. We need that to pair the phone to the Android Wear Device (real or emulator). Vist the link and download the App from here.
5. Create an Android Wear AVD. The instructions are clearly mentioned here. **Follow the instructions carefully. It is very important that you wait till step 5 completes and you see a screen that looks like this:**

The disconnected cloud indicates that the Wear Device (i.e. Emulator) is not yet paired with the Android phone. You can follow the link above after Step 5 but I am just highlighting the areas here that give you more detail.

6. Next, go to the command prompt/console and fire the **adb devices** command. You should see at least 2 devices listed i.e. your phone and your emulator (Wear) as shown below:

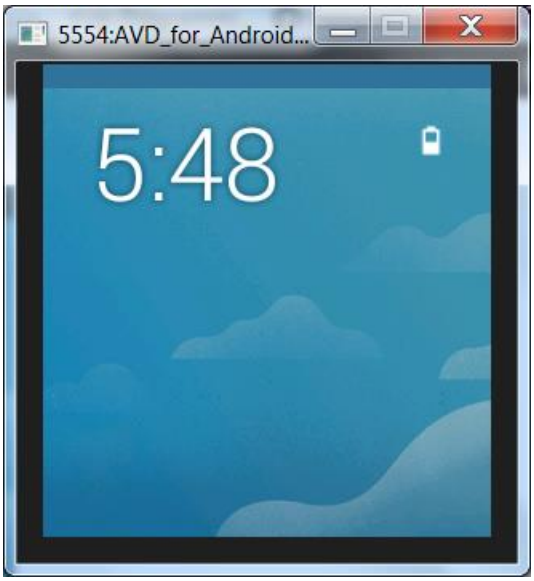
```
C:\>adb devices
List of devices attached
emulator-5554   device
TA93305YW5      device
```

7. Next, fire the following command as shown below:

```
adb -d forward tcp:5601 tcp:5601
```

This will forward the AVD's communication port to the connected handheld device (you must do this every time the handheld is connected):

8. Now launch Android Wear on your Android Device and connect it to the Emulator. This can get tricky and I have written a blog post for it here.
9. If all is well, you should see the disconnected icon go away and you can try showing some demo cards from the Android Wear app on the phone. They should appear on your Wear Emulator.



You are all set now ! Nothing can stop you from taking over mankind now !

Sample Codelab Projects

We have 3 Android Studio Projects that you can use to explore Android Wear Notifications. The Code Labs here will focus only on bridged notifications. These notifications are automatically bridged from your phone to the paired wearable. We will see a variety of these notifications ranging from:

1) Simple Notification – just text, icon, title
2) Simple Notification with Background
3) Stacked Notifications
4) Notifications with Multiple pages
5) Notifications with actions

The 3 Code Labs are described below, each of them is an Android Studio Project that you can import into the IDE and get going:

AndroidNotifications2 and AWN

Both of these projects are similar in nature and they demonstrate the various kinds of Notifications that you can raise, as mentioned above. You can look at either of them.

Once you get them working successfully, play around with the notifications. Here are things that you can do:

1) Create your own notifications : change the text, background. See them work.
2) Start thinking of Stacked Notifications and other examples of where it can useful. Go ahead and create some dummy Stacked Notifications.
3) Create a Notification with your own additional pages.
4) Add your own Actions that fire the Intent on the phone. It can be a simple Activity that gets launched once it is launched on phone from the Wear device action.

Note : The AWN2 Project has been inspired by the excellent series present at : http://code.tutsplus.com/tutorials/signals-and-microinteractions-for-smartwatches-hands-on--cms-22260

Mumbai Traffic Alerts

This one is a bit interesting and shows a real world app. It hits a backend service hosted at Mumbai Monsoon Alerts. We are going to look at a live Web Service (App Engine Cloud Endpoints powered) that gives you Traffic Alerts for Road , Rail and Air traffic in Mumbai. If there are any disruptions or alerts, the Web Service response contains that information.

The Endpoint is available at :

https://mumbaimonsoonalerts.appspot.com/_ah/api/bmctrafficalertsendpoint/v1/bmctrafficalerts

If you hit it (Please be nice and considerate – which means hit it a few times only), you will get the following response:


```
{
items: [
{
id: "1",
data_air: "Normal",
data_road: "Normal",
data_rail: "Normal",
kind: "bmctrafficalertsendpoint#resourcesItem"
}
],
kind: "bmctrafficalertsendpoint#resources",
etag: ""FiHrXd0AkUYA7kEIfICBhFMJeqk/2ir9VF8_GEcChNV00y4YxTUnd-s""
}
```


You need to monitor the data_air, data_road and data_rail elements in the response. If they are Normal, all is good. If they are not, there is an Alert and **this is a perfect candidate to raise a Notification that can be shown on your Wear Device.**

The Android App source code is straightforward to follow. You will find a Service that makes regular calls to the Endpoint , checks for changes and raises Notifications as appropriate.

Think of this project as a template for any public services that you want to monitor and then raise notifications for.

Note: The source code in the projects is kept simple to demonstrate the Wear Notifications. This should not be considered as best practices or production ready code.