Designing a basic Mobile Appcesssory – HEllo WOrld

# Overview

We will create and build a simple hello world application for your android device. We will walk through the process of configuring the environment and installing drivers for debugging a Nexus 5. Basic android debugging concepts will be introduced as well as basic user interface concepts.

# Required Materials

## Hardware

* Android device for debugging. The device recommended for this purpose is the Google Nexus 5 as it is currently the best supported for android development. If this device is not is used, ensure the device supports Bluetooth low energy and Android 4.3 (API Level 18) or above, as these features will be used by later labs. Can be purchased on the Google play store.

<https://play.google.com/store/devices/details/Nexus_5_16GB_Black?id=nexus_5_black_16gb>

* Micro USB cable type B to type A for attaching the mobile to the PC. Comes included with the Nexus 5.

## Software

* A 64-bit version of JDK (Java SE Development Kit) is required by android studio for java compilation. This can be downloaded free of charge from Oracle’s website.

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

* Android Studio for app development. This is the Android development environment which includes the Android SDK Tools and an emulator. This can be downloaded for free on the Android Developer Tools page.

<https://developer.android.com/sdk/installing/studio.html>

* Google USB drivers. These are required for Android Debug Bridge (adb) to detect your device. We will install these through Android Studio later in these lab notes. They can also be found online on the Android Developer Tools page.

<http://developer.android.com/sdk/win-usb.html>

# Creating a new project

Creating a “Hello World!” app in android studio is a very simple process. Follow the *Create New Project* wizard, targeting API version 18 (or android 4.3). This API version is required as this marks the introduction of the Bluetooth Low Energy (BLE) API. Pick the *Blank Activity* option for the activity to add. Follow the wizard to the end, and you should be presented with our app as in Figure 1.



Figure . "Hello World!" app after finishing the *Create New Project* wizard.

# USB Setup

Setting up USB debugging with your Nexus 5 can be very finicky so it’s advised to follow these instructions to the letter.

## Enabling USB Debugging

In order to enable USB debugging on your device, we must first enable the *Developer options* section of the settings app. This is done by opening the settings app, picking the *About phone* section and pressing the *Build number* entry seven times. This should popup a toast notification confirming the developer options have been enabled. Browse back to the root of the settings app and open the newly created *Developer options*. Scroll down and tick *USB debugging*, and accept the confirmation dialog.

You may also run into issues with driver installation unless you switch the USB computer connection to PTP mode. This can be achieved by opening the settings app again, opening *Storage*, pressing the three dots in the top right of the screen then clicking *USB computer connection*. Select the *Camera (PTP)* option. We should now be ready to install the drivers.

## USB Drivers

The correct drivers must be installed for your device to be recognized by Android Debug Bridge (adb). These can be downloaded from *Android SDK Manager*  which is found on the toolbar inside Android Studio. Scroll down to *Extras* and tick *Google USB Driver*. Press the *Install* button. If the installation fails you may find Android Studio is installed inside Program Files, in which case you are required to reopen the program with the *Run as administrator* option.

Once the installation is complete, open up the Control Panel and pick Device Manager. You should see a Nexus 5 under the *Other devices* section as in Figure 2. Right click on Nexus 5 and select *Update Driver Software…*. Browse to the SDK path as shown in the *Android SDK Manager* and select extras/google/usb\_driver. Ensure *Include subfolder* is ticked and click *Next*. Accept any confirmation dialog which may appear and wait for the process to complete. You should be greeted with a message that Windows has successfully updated your driver software!

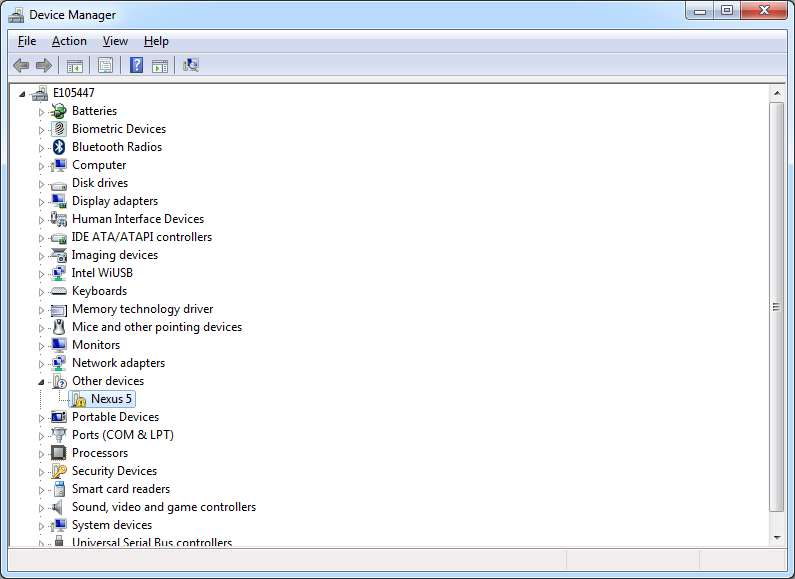


Figure . Nexus 5 selected in Device Manager with no drivers installed.

If you run into any problems with this process, the Android developer website has further information on installing USB drivers at <http://developer.android.com/sdk/win-usb.html>.

## Confirmation

Back in Android studio, press the *Debug*  button. A dialog should popup on the device asking if you want to authorize the USB connection, as shown in Figure 3. It may be useful to tick *Always allow from this computer*. Click OK to accept and dismiss the dialog. If everything has been successful you should see our Hello World app begin to run on the device!

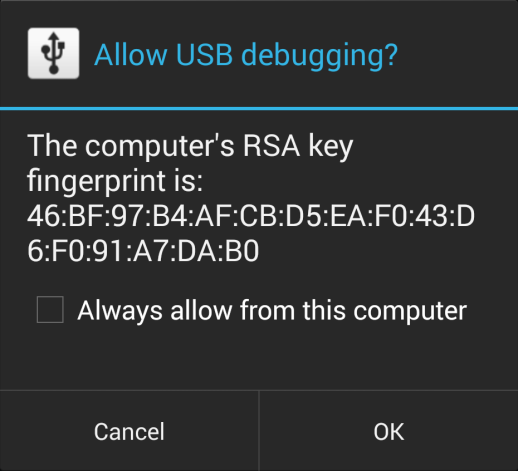


Figure . USB debugging authorization prompt.

# Debugging

## Debugging From Android Studio

Press the *Debug*  button to begin the build and debugging step. When the build is finished, a dialog should popup with device selection (Figure 4). If the device does not appear or the state is *Offline* / *Unauthorized*, try unplugging then plugging the USB cable back into the computer. Pick the target device, optionally tick *Use same device for future launches*, then press OK. Our app should now appear on the mobile device!

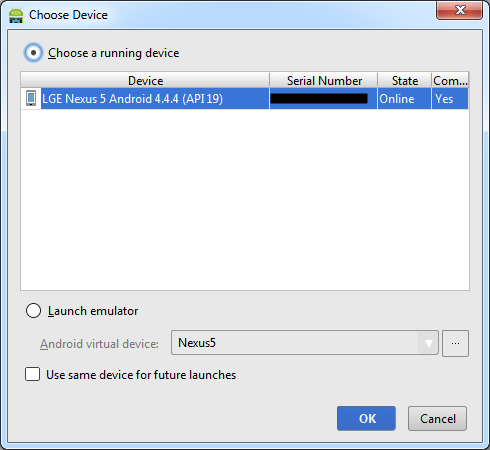


Figure . Device selection dialog for debugging.

All of this lab will be run in debug mode as it allows for observation of the status of our running app, however if you only wish to run the app on your phone without attaching a debugger it is possible to press the *Run* button  which is next to the *Debug* button. You may always attach a debugger later if the app is already running with the *Attach debugger to Android process* button .

## Logcat

Change the visible tool window to Android  and view the logcat output for your device. You will probably see a lot of messages if you have many apps running in the background. The debug line Log.d("MainActivity", "Hello from onCreate!") to the *onCreate* method and restart the debug step. See if you can find the message “com.arm.university.helloworld D/MainActivity﹕ Hello from onCreate!” appear in logcat. It may be worth filtering the output to only display messages from the hello world app, as in Figure 5. Enter the package name as configured when the app was created. The package name can also be found at the top of the Activity source code after the *package* keyword.

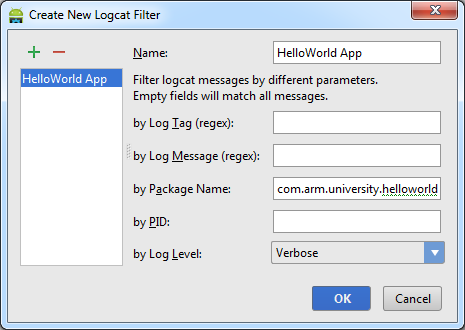


Figure . Logcat filtering to com.arm.university.helloworld package.

## Breakpoints

Try adding a breakpoint to the line in onOptionsItemSelected as in Figure 6, and entering debug mode. What actions do you have to perform in the app to cause the breakpoint to be hit? What happens if you try to interact with the app when the breakpoint has been hit?

Press the options menu at the top left of the screen (the three vertical dots), then press the *Settings* option. The app freezes and any touches on the app are ignored.



Figure . Breakpoint added to onOptionsItemSelected method.

# App Programming

Currently all our app contains is a TextView with the text “Hello World!”. If we wish to provide the user with any further interaction we have to add other controls to the user interface. The page we see when we open our app is known as an *Activity* in the Android system. The activity can be found by navigating to app/src/main/res/layout in the *Project* view, or under layout in the *Packages* view. Open the xml file for the blank activity which we created during the *Create New Project* wizard.

Add a button to the center of the screen and change the *onClick* property to *displayMessage*. We can find further information about the *onClick* property on the Android API guide. Optionally change the text to display a message to the user. The display should resemble Figure 7.

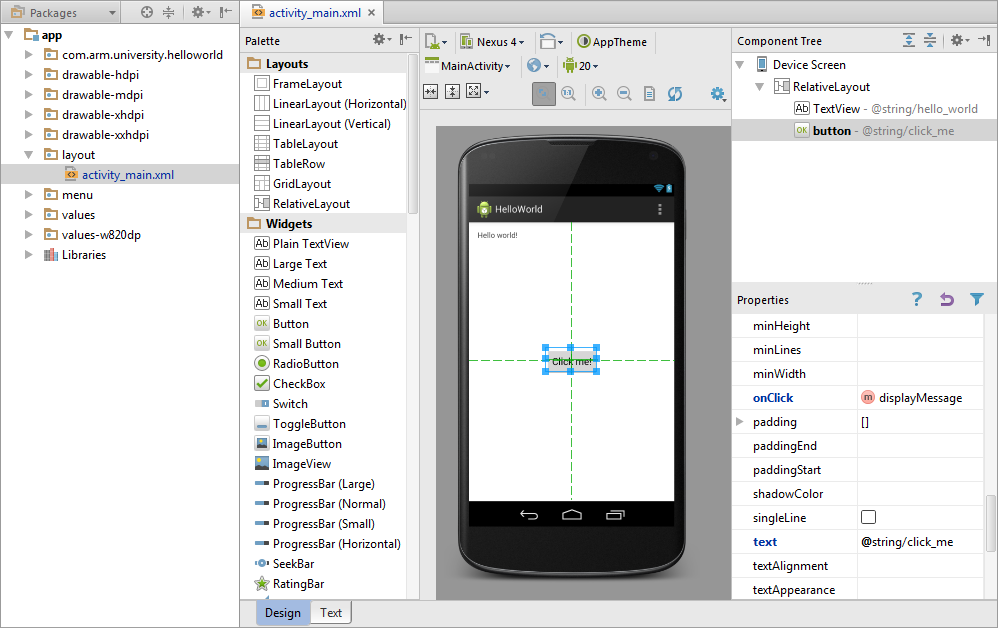


Figure . User interface with added button.

Go back to the java source file for the activity, and add code to display a message “Hello from button!” when the button is pressed. The snippet should be inside of a method called *displayMessage*. For reference, refer to the API guide for the *onClick* property and for the Toast.*makeText* function.

public void displayMessage(View view) {

Toast.makeText(this, "Hello from button!", Toast.LENGTH\_SHORT).show();

}

Add a breakpoint to the Toast.*makeText* function call and press the button on your device. Look inside the *Variables* window for the *View* object. What is its type? What is the type and value of the *mText* variable inside the *View* object?

The *View* object is the Button that we defined. Its type is android.widget.Button. The *mText* variable is a java.lang.String with value “Click me!”.