Arduino & Android

A “How to” on interfacing these two devices

Bryant Tram

Contents

1 Overview

In this "how to" guide, I will show you the programs and steps needed to get the Arduino and the Android phone to communicate with and interface together. First, you will be introduced to the hardware and devices that will be use. Secondly, you will be shown the programs that are used after the initial set up to initialize the board and android. Lastly, I will show you an example lab showing the board and android working together.

2 Other Readings

1. Android Debug Bridge -

The Android Debug Bridge is command tool within the Android platform that allows for the Android Device to be connected to other emulators or devices. This article will inform you on how this tool works. It will also provide supported ADB commands and how to implement them.

http://developer.android.com/tools/help/adb.html

2. MicroBridge –

MicroBridge is the microcontroller's version of Android Debug Bridge and it will allow microcontrollers such as our Arduino Mega to interface with the Android via the USB shield. This article is written by the author of the MicroBridge code and will explain how it works with the Android and provide greater understanding of the code.

http://code.google.com/p/microbridge/

3. YouTube tutorial video series on how to build your own android app

http://www.youtube.com/course?list=ECB03EA9545DD188C3&feature=plcp

3 Hardware and Devices

Where to buy:
1. Arduino Mega 2560 – Figure 1
http://www.seeedstudio.com/depot/arduino-mega-2560-p-695.html?cPath=132_133

2. USB Host Shield – Figure 2
https://www.sparkfun.com/products/9947?
Or
http://www.circuitsathome.com/products-page/arduino-shields/usb-host-shield-2-0-for-arduino

3. Android 1.5+ device – ex. Archos - 28 Internet Tablet – Figure 3
http://www.newegg.com/Product/Product.aspx?Item=N82E16855501057

Figure 1

Figure 1. Arduino Mega 2560 (Arduino website)

Figure 2

Figure 2. USB Host Shield (Circuitsathome.com)

Figure 3

Figure 3. Archos -28 Internet Tablet (Amazon)

4 Programs Download

Each program is free to download and can be found at the links provided.

1. Arduino 1.0.1 software
http://arduino.cc/en/Main/Software

2. Java Eclipse
http://www.eclipse.org/downloads/

5 Other Downloads

The Arduino IDE requires these libraries to be added in order to implement the lab.

1. USB Host Shield library
http://code.google.com/p/microbridge/downloads/detail?name=MicroBridge-Arduino.zip&can=2&q=

2. MicroBridge Arduino Library
http://code.google.com/p/microbridge/downloads/detail?name=MicroBridge-Arduino.zip&can=2&q=

3. Servo Control demo App – An example of another lab that could be done after this one.
http://code.google.com/p/microbridge/downloads/detail?name=ServoControl.zip&can=2&q=

6 Setting up the Arduino Program

6.1 Step 1: Download and Extract the Arduino folder

Download and Extract the files in the arduino-1.0.1.zip files included in this folder. The extracted folder should include the following files.

FOLDERS

1. drivers
2. examples
3. hardware
4. java
5. lib
6. libraries
7. reference
8. tools

FILES

1. arduino.exe
2. cygiconv-2.dll
3. cygwin1.dll
4. libusb0.dll
5. revisions.txt
6. rxtxSerial.dll

6.2 Step 2: Download and place the Microbridge library

- Download and Extract the files in the MicroBridge-Arduino.zip files included in this folder.
- This folder should contain another folder label Adb.
- Now take this folder and drop it into the Arduino folder "6. Libraries"

6.3 Step 3: Modifying the Library (conditional)

If you are using the Arduino Mega 2560 with the USB shield from:
http://www.circuitsathome.com/products-page/arduino-shields/usb-host-shield-2-0-for-arduino
Then go to the Adb Folder and Right click "edit" on the file max3421e.h
Then on the line 67 & 80 change the following:

```
#define MAX_SS(x) digitalWrite(SS, x)
```

into

```
#define MAX_SS(x) digitalWrite(PIN_MAX_SS, x)
```

If you are using the USB host shield from sparkfun:
https://www.sparkfun.com/products/9947?

Then do not modify the Library file
We have not tested this library on this USB shield from sparkfun but Jonathon Dowdall, creator of RoFi, has got it to work on it as is.

7 Install Java / Eclipse and set up the Android SDK

7.1 Step 1: Download Eclipse & Install Eclipse

Download Eclipse IDE for Java Developers at the following website:

http://www.eclipse.org/downloads/

- **Make sure you download the correct version for whatever JAVA version you are using* Either the 64-bit or 32-bit*
- *If not sure then go to control panel and click on programs to see which one are your system is running*

7.2 Step 2: Install Android SDK

Download and install the Android SDK manager:

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

After the Manager is installed your screen should look like Figure 4.

Figure 4

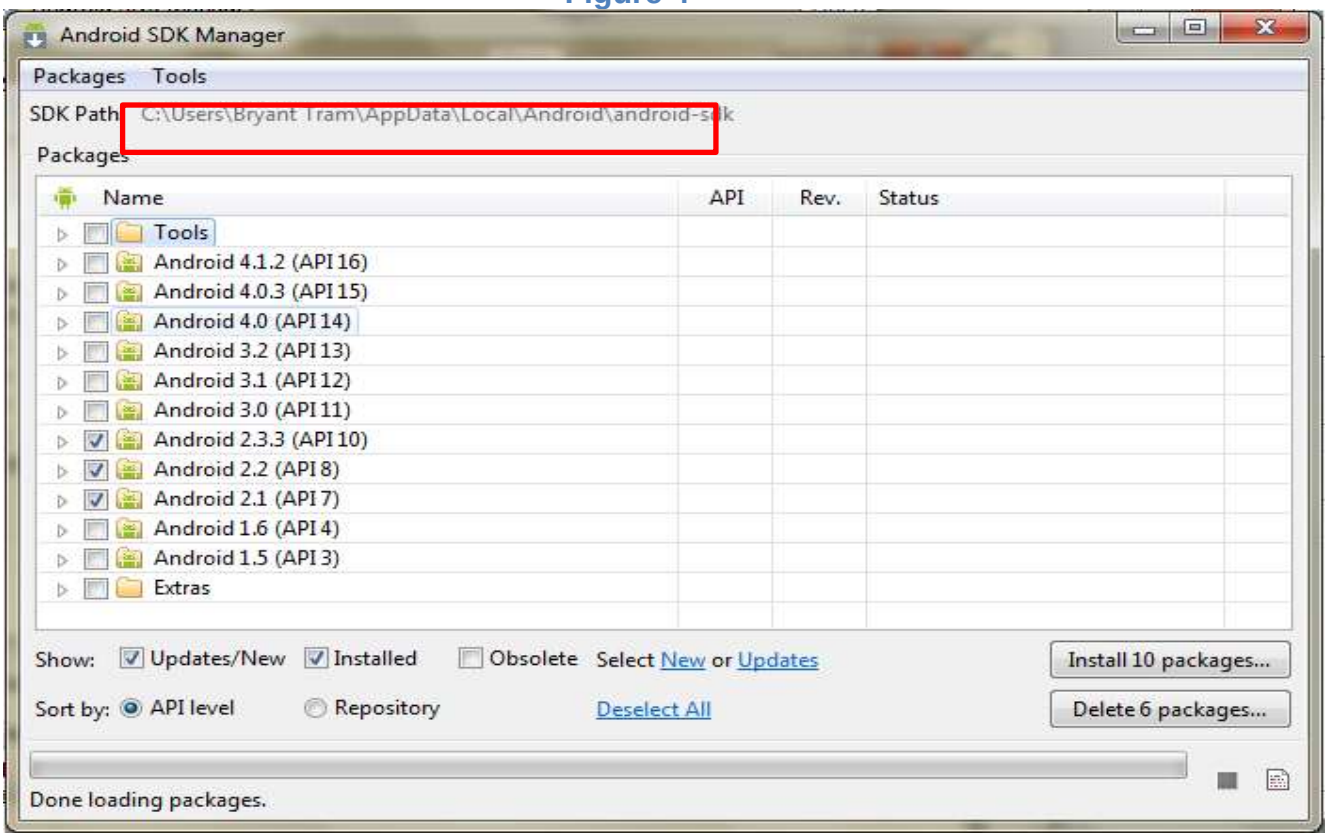

Figure 4. Screenshot of the Android SDK Manager

Now install the SDK you will be using which are the one that are currently checked.

- Android 2.3.3
- Android 2.2
- Android 2.1

After clicking install, it will take some time for your computer to download and install.
Also note where the PATH of the SDK is installed

7.3 Step 3: Install ADT plugin for Eclipse

1. Start Eclipse, then select **Help** > **Install New Software**.

2. Click **Add**, in the top-right corner. *Refer to Figure 5*

3. Type in "ADT Plugin" for name and https://dl-ssl.google.com/android/eclipse/ for location

4. Then check on Developer Tools and click on next and install: *Refer to Figure 7*
If you get a security warning saying that the authenticity or validity of the software can't be established, click OK

5. Restart Eclipse

6. Once Eclipse restarts, in the "Welcome to Android Development" window that appears, select Use existing SDKs

7. Now you will be asked the location of the SDK you recently downloaded in STEP 7.2. So copy and paste the location given in 7.2 and paste. Then click Next.

At this point if you have ran into any errors then you have successfully install and configure Eclipse to develop Android Apps.

Figure 5

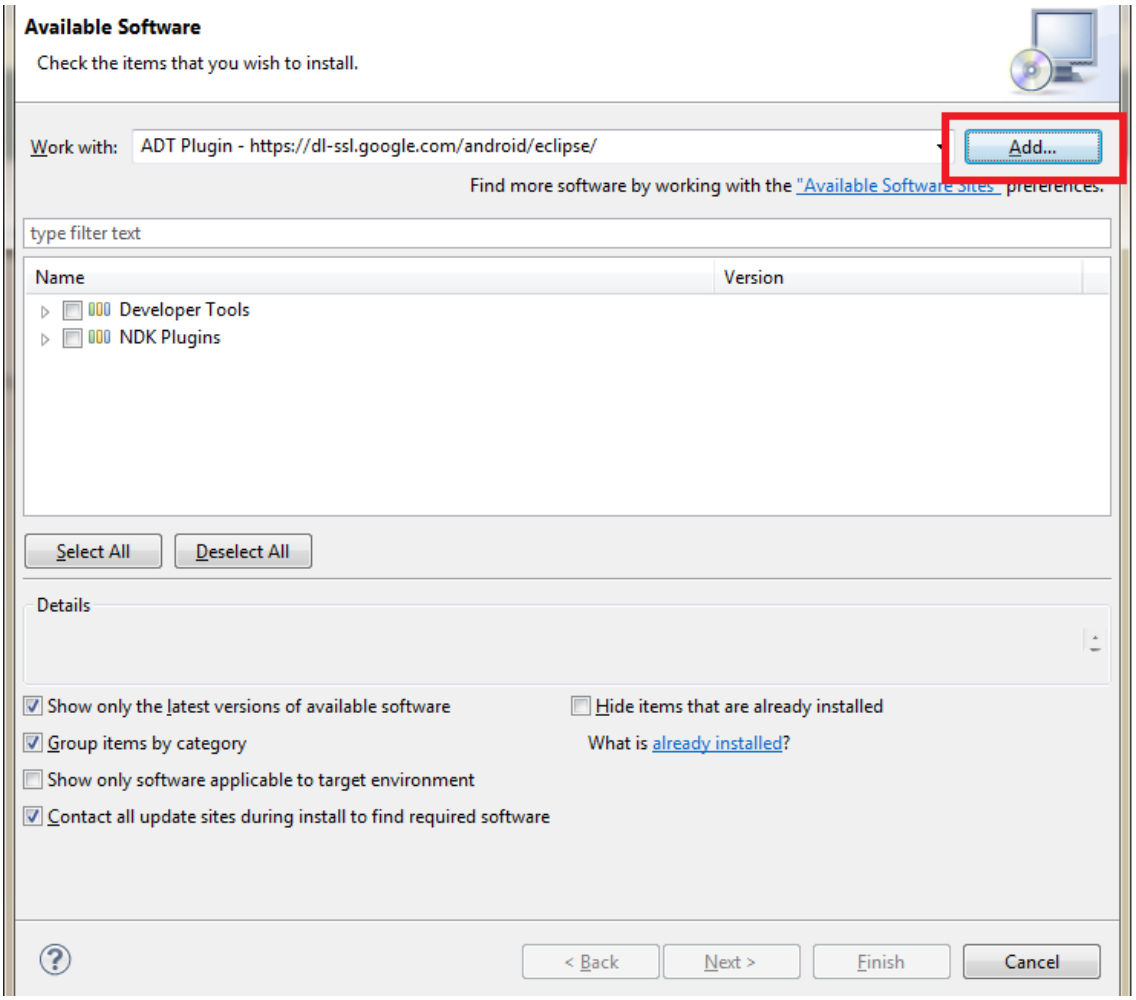

Figure 5. The window that pops up after clicking **Install New Software**

Figure 6

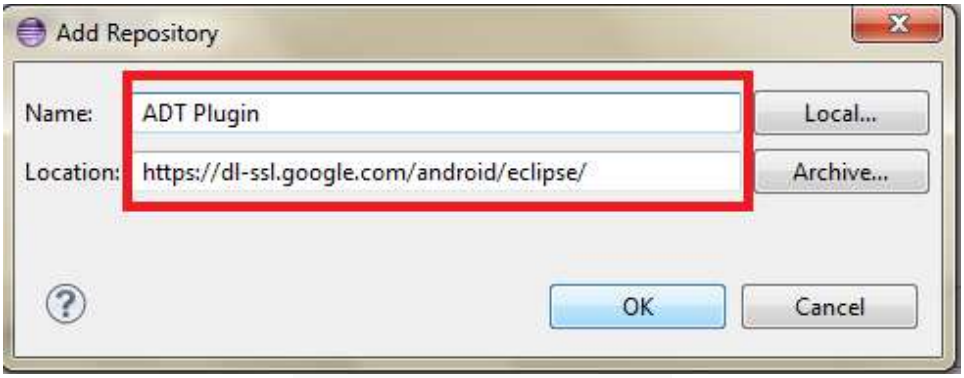

Figure 6. Screen for step 3

Figure 7

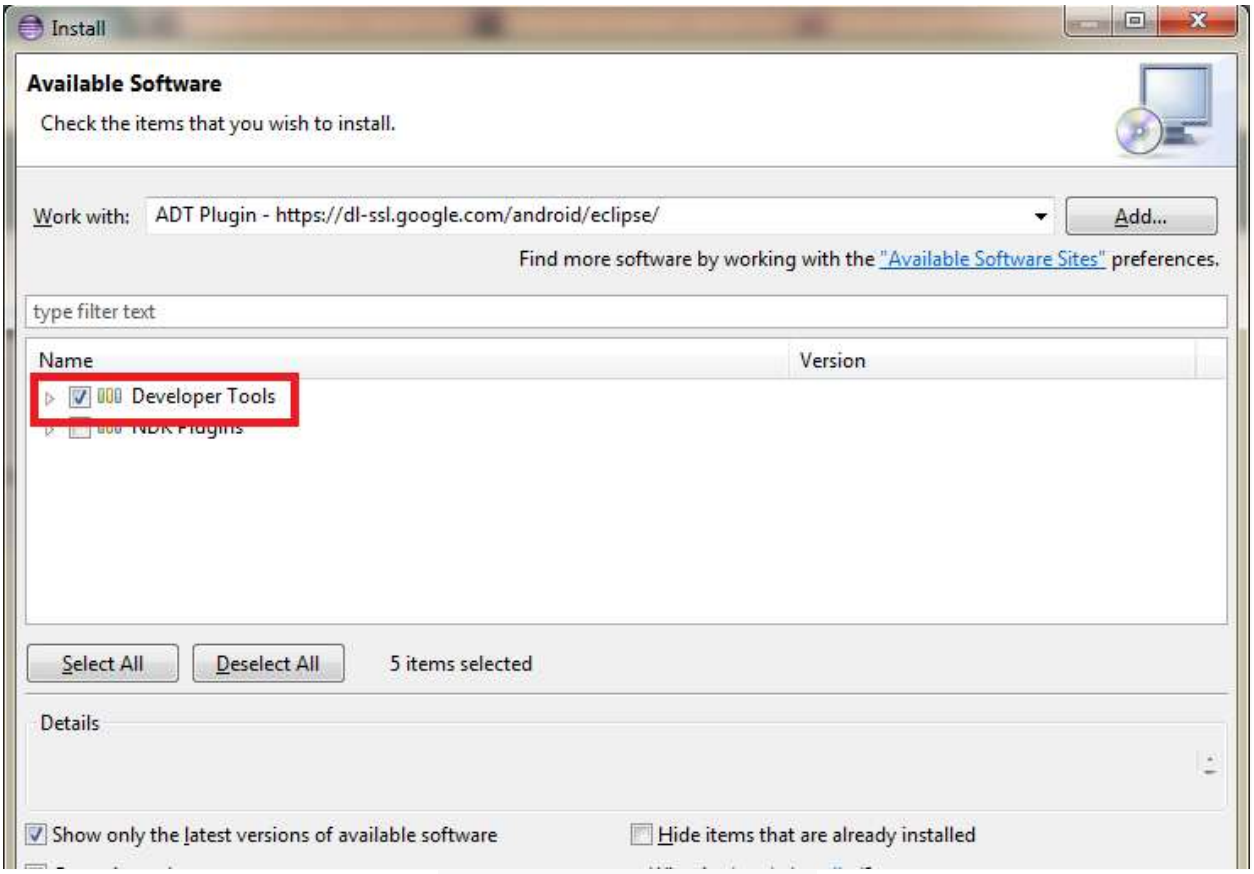

Figure 7. Screen for step 4

8 Compiling the apk and uploading the Adb code

8.1 Step 1: Starting a new eclipse project

1. Start Eclipse, then select **File** > **New** > **Project**
2. Select Android Application Project then click Next

Figure 8

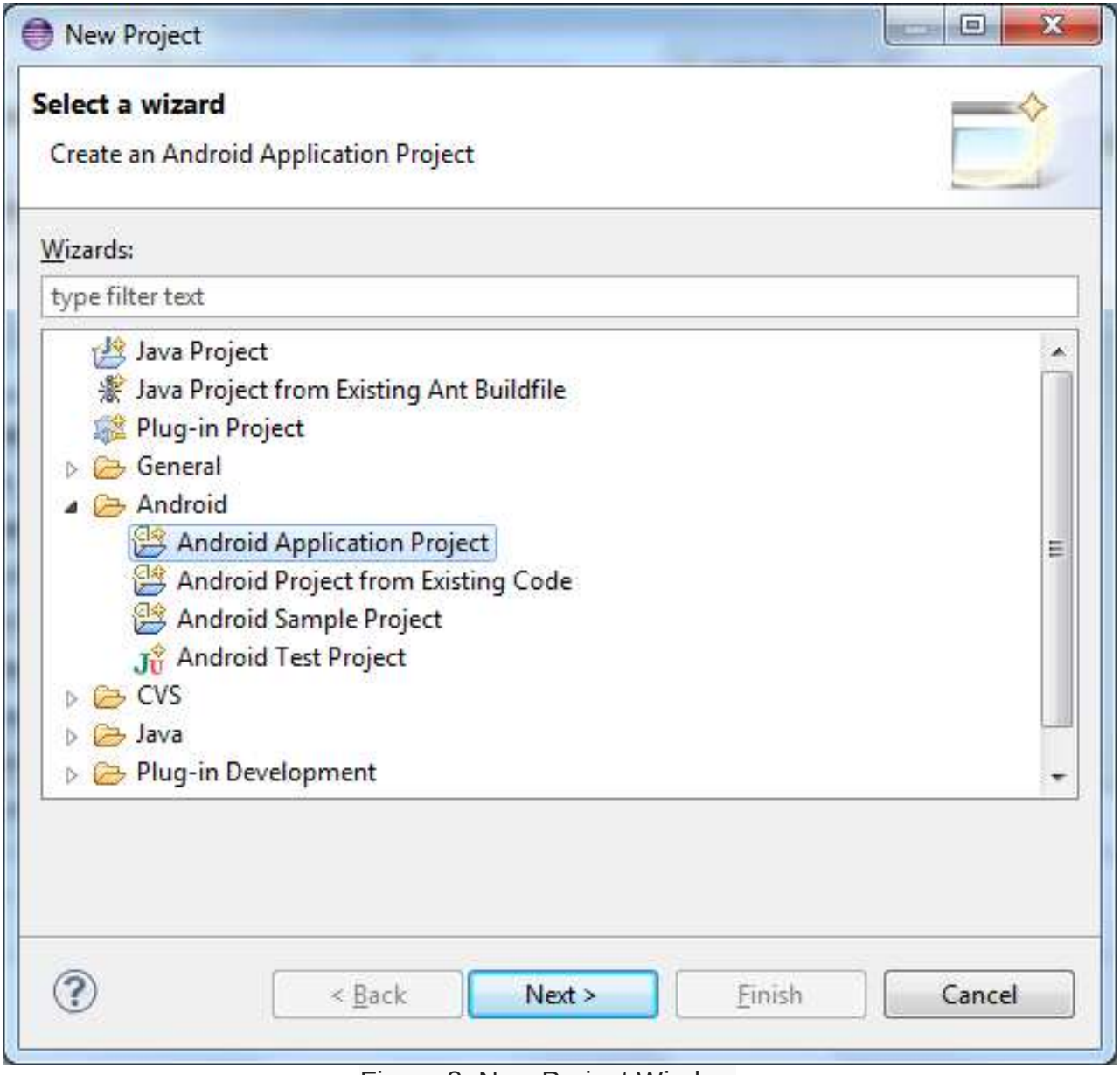

Figure 8. New Project Window

3. Now we name our project/app name

(Typing in the application name box will adjust the rest of the names)

4. Change the Build SDK to whichever android platform your device is working on. For the Archos, use API 7. *Refer to Figure 9*

Figure 9

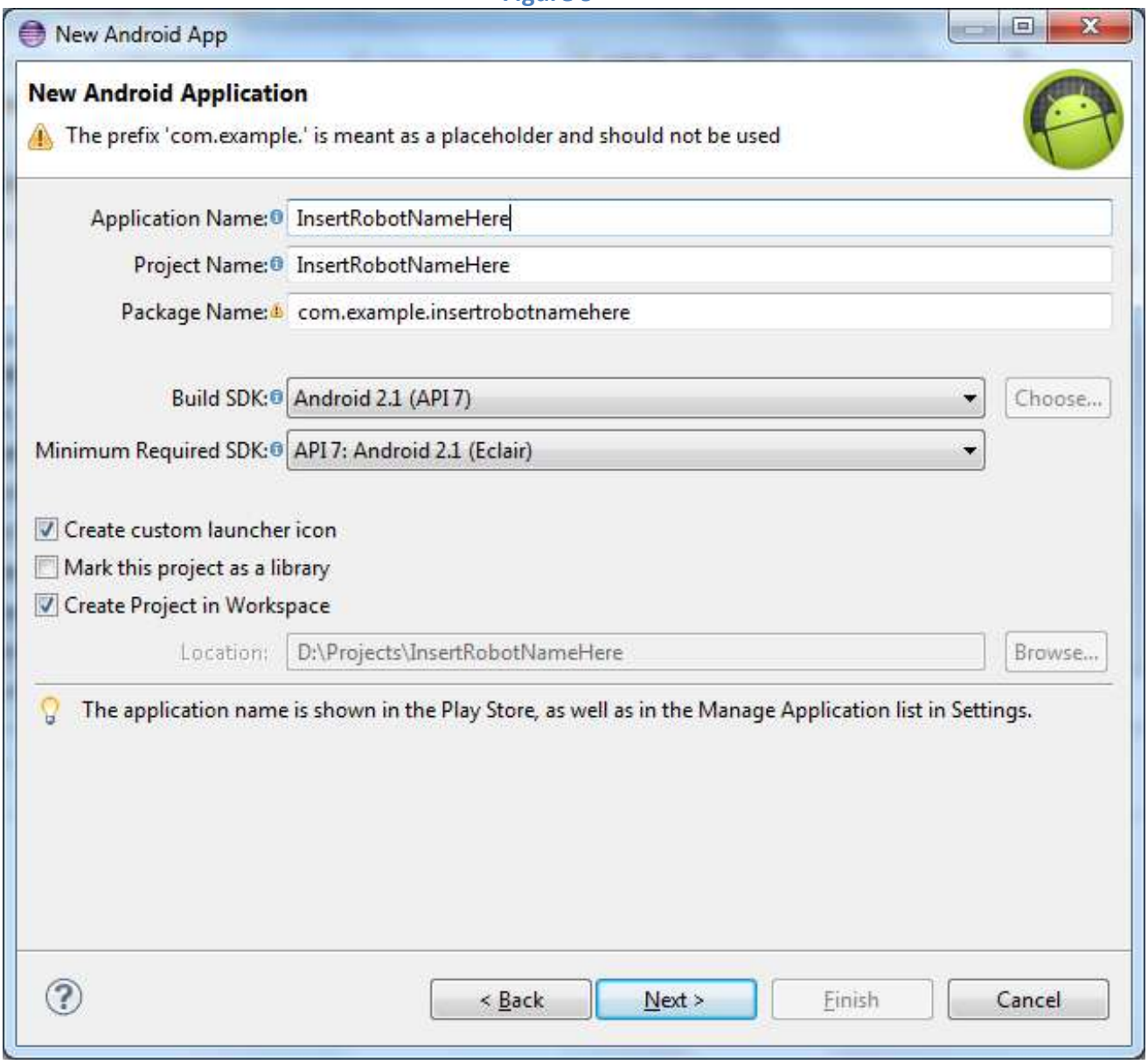

Figure 9. Screen of New Android Application

5. This step is where you can edit the icon of the app (optional)
For now we will just skip this step and just click next

Figure 10

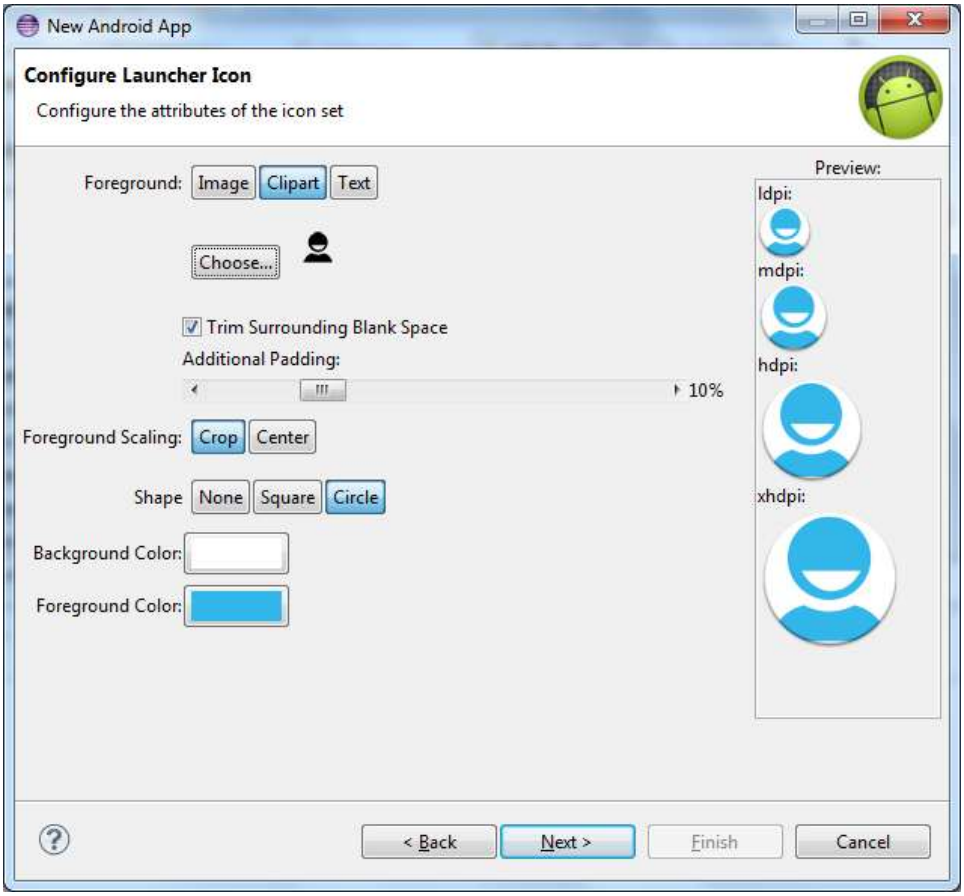

Figure 10. Screenshot of the Icon edit window

6. In the Create Activity screen, just click next since we want to customize our app from scratch.

7. Now we click finish and it will take you to the app editor / workspace. *Refer to figure 11*

Figure 11

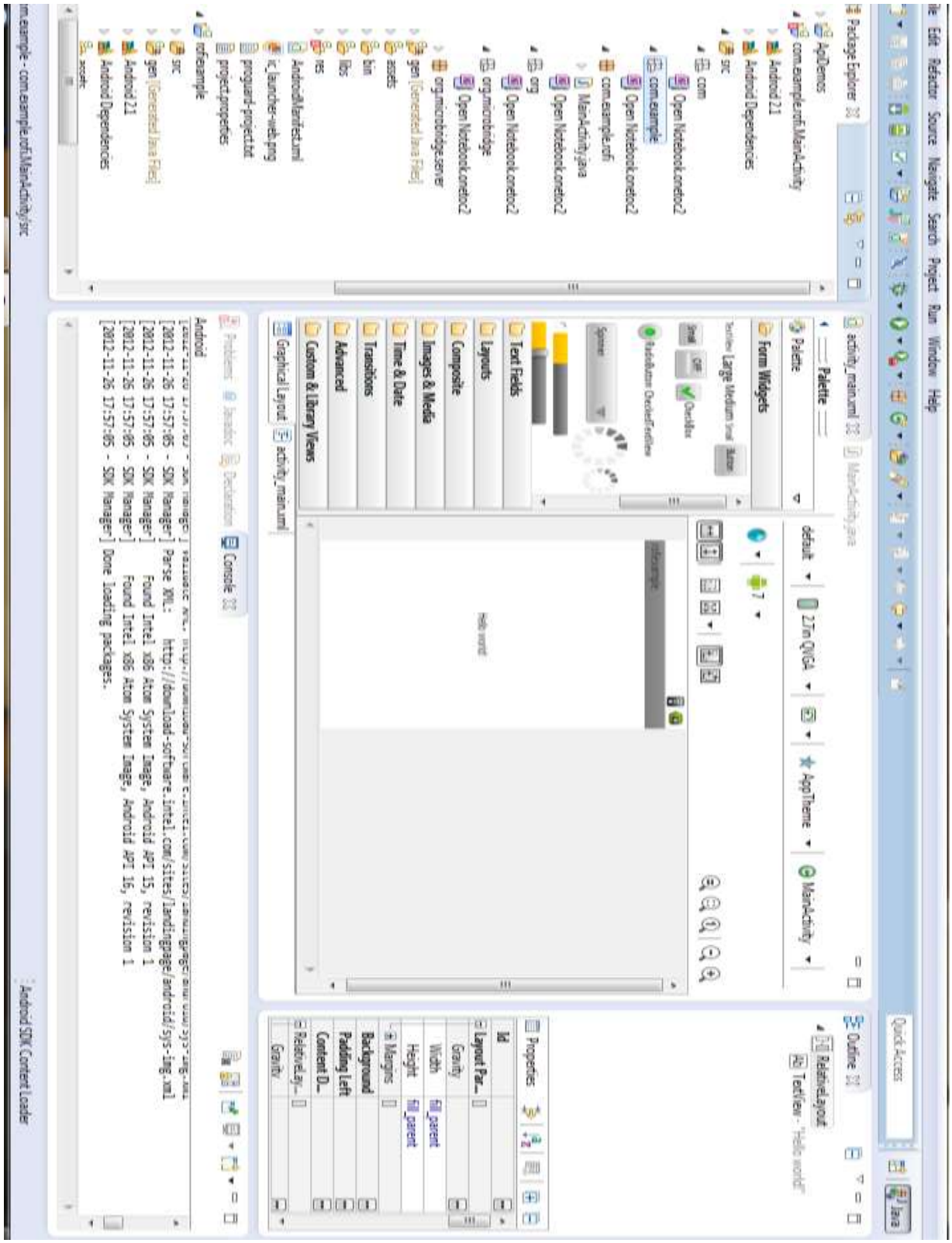

Figure 11. This is what your workspace should look like

8.2 Step 2A: Setting up the application screen

1. This part is where you add buttons and such to the screen and it will be the layout in the android device. First you should make sure the screen size is correct for your device. In our case it is *2.7 QVG*

Figure 12

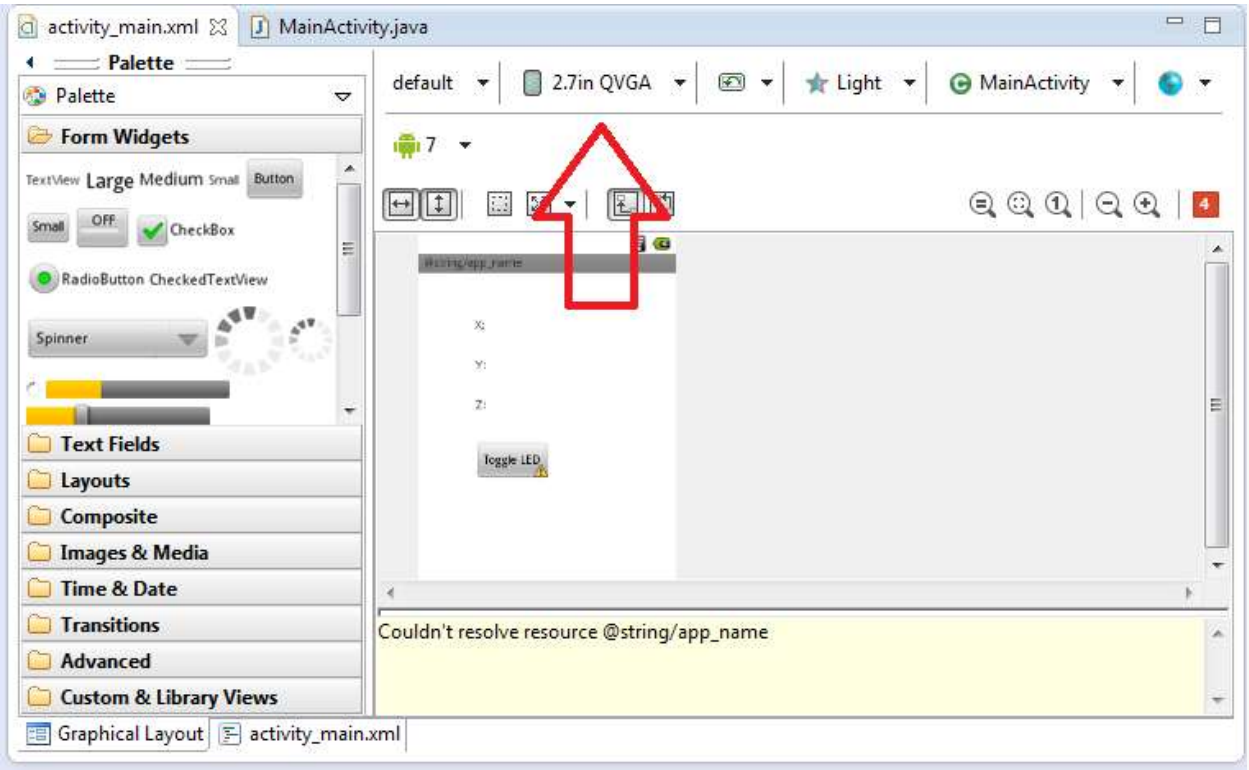

Figure 12. Change the layout of the screen workspace into the format your Android device is in.

2. Now we place a button on the main screen by dragging and dropping it there. Refer to the red arrows if you don't know where. *Refer to Figure 13*

3. Now we add the other values by clicking on the activity_main.xml tab at the bottom of the screen and just copy and paste this code right in. *Refer to Figure 14*

4. Now go to your MainActivity.java file and double click it to open up the file window. And copy and paste the code provided underneath your package file. *Refer to Figure 15*

5. Now just run and compile the project and it should produce an APK file inside the bin folder. We now just connect the android to your computer and drop the apk file into the device. Then just install it by clicking on the apk file while using the device.

Figure 13

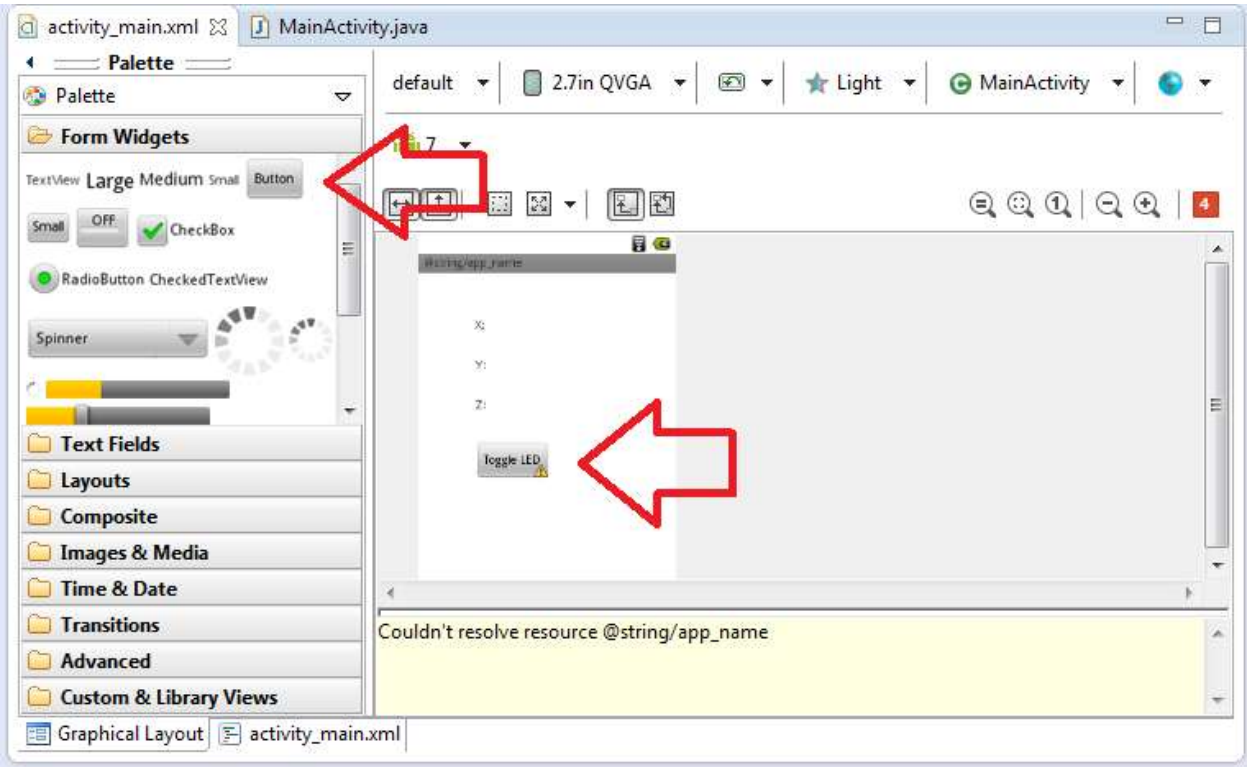

Figure 13. Click and drag the button onto the screen

Figure 14

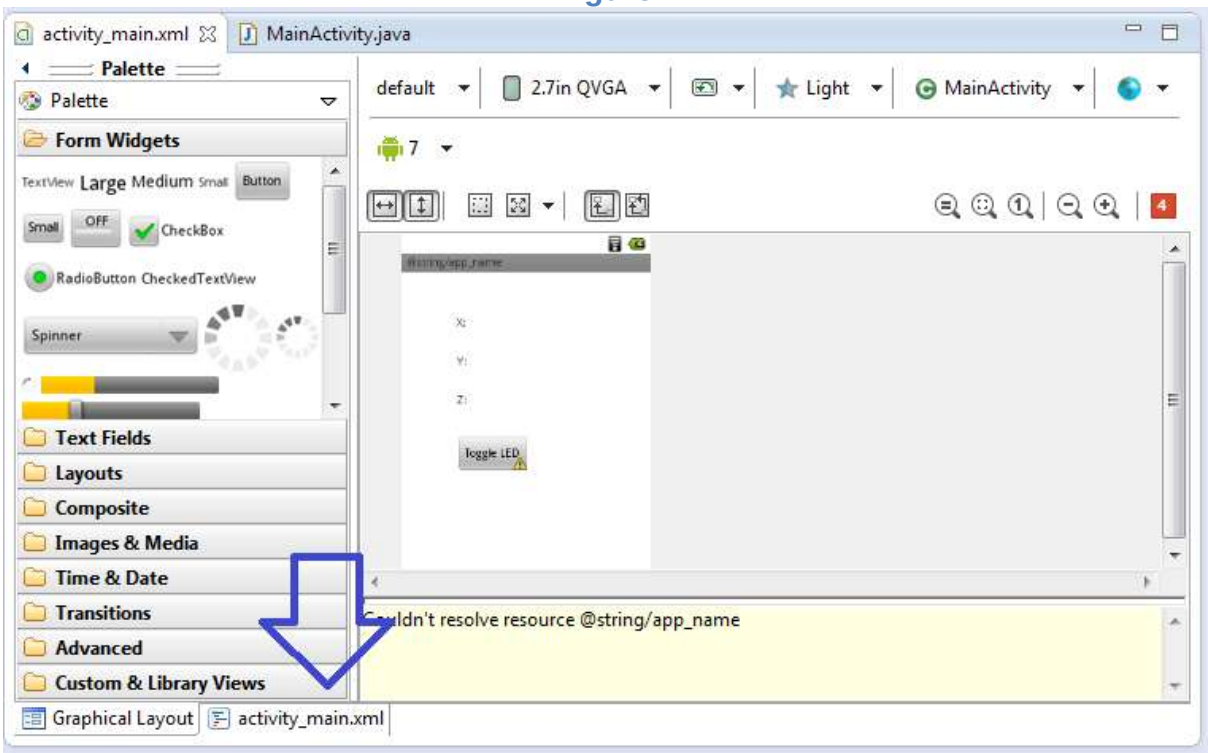

Figure 14. The tab for the screen containing the code for the layout of the screen

Note that this code is the java code for the layout of app.

CODE for activity_main.xml:

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <TextView
        android:id="@+id/accelerometerX"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="70dp"
        android:layout_marginTop="54dp"
        android:text="X:" />

    <TextView
        android:id="@+id/accelerometerY"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/accelerometerX"
        android:layout_below="@+id/accelerometerX"
        android:layout_marginTop="30dp"
        android:text="Y:" />

    <TextView
        android:id="@+id/accelerometerZ"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/accelerometerY"
        android:layout_below="@+id/accelerometerY"
        android:layout_marginTop="30dp"
        android:text="Z:" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/accelerometerZ"
        android:layout_below="@+id/accelerometerZ"
        android:layout_marginTop="37dp"
        android:onClick="toggleLed"
        android:text="Toggle LED" />

</RelativeLayout>
```

Figure 15

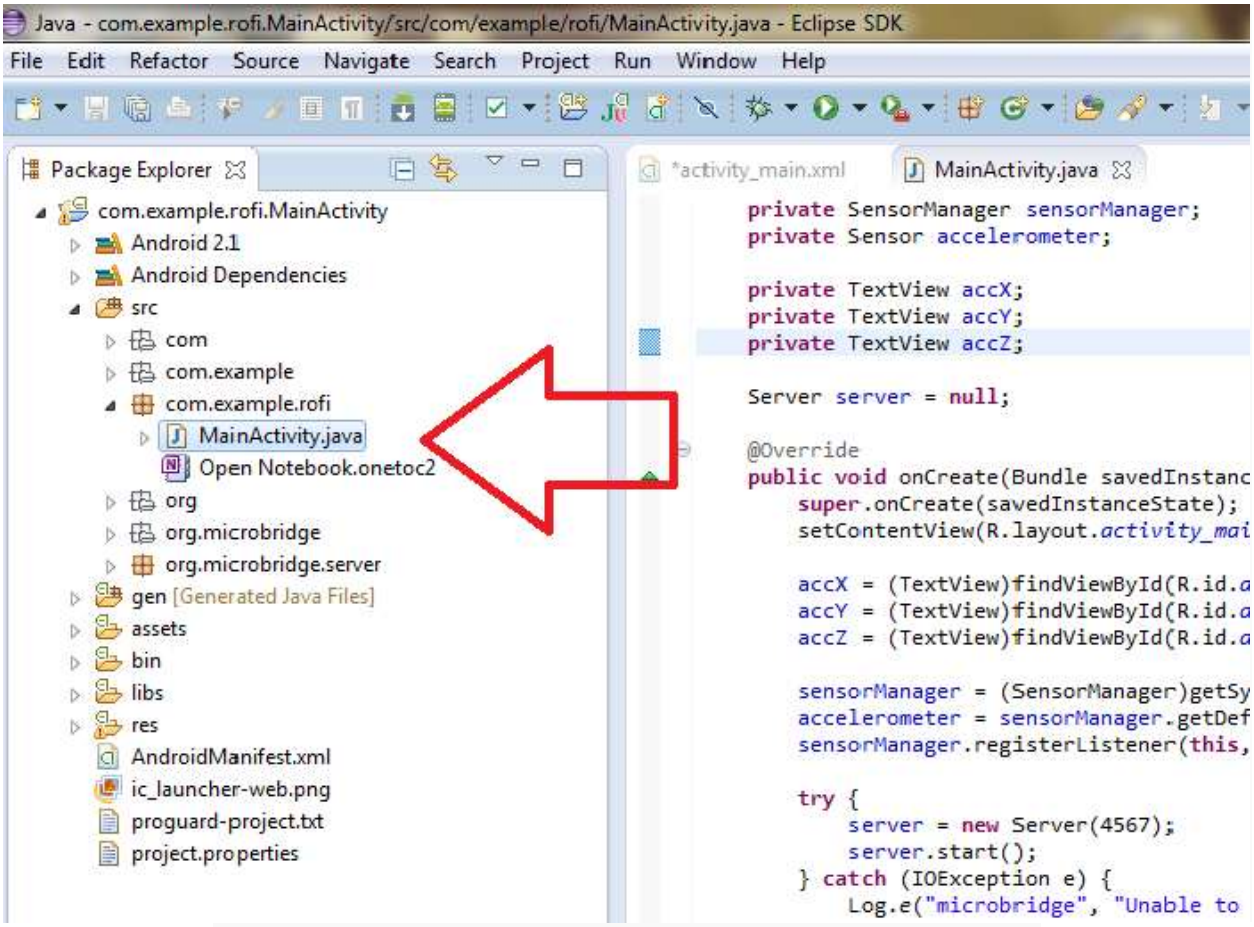

Figure 15. Where the main code of the program is located

Code for the Main Activity section:

```
import java.io.IOException;

import org.microbridge.server.Server;

import android.app.Activity;
import android.content.Context;
import android.content.res.Configuration;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.View;
import android.widget.TextView;

public class MainActivity extends Activity implements
SensorEventListener {
```

```
      private SensorManager sensorManager;
      private Sensor accelerometer;

      private TextView accX;
      private TextView accY;
      private TextView accZ;

      Server server = null;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        accX = (TextView)findViewById(R.id.accelerometerX);
        accY = (TextView)findViewById(R.id.accelerometerY);
        accZ = (TextView)findViewById(R.id.accelerometerZ);

        sensorManager =
(SensorManager)getSystemService(Context.SENSOR_SERVICE);
        accelerometer =
sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
        sensorManager.registerListener(this, accelerometer,
SensorManager.SENSOR_DELAY_NORMAL);

        try {
                server = new Server(4567);
                server.start();
           } catch (IOException e) {
                Log.e("microbridge", "Unable to start TCP server", e);
                System.exit(-1);
           }

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }

    @Override
      public void onConfigurationChanged(Configuration newConfig) {
             super.onConfigurationChanged(newConfig);
      }

    @Override
    protected void onResume() {
      super.onResume();
      sensorManager.registerListener(this, accelerometer,
SensorManager.SENSOR_DELAY_NORMAL);
```

```
    }

    @Override
    protected void onPause() {
      super.onPause();
      sensorManager.unregisterListener(this);
    }

      public void onAccuracyChanged(Sensor arg0, int arg1) {
            // TODO Auto-generated method stub

      }

      public void onSensorChanged(SensorEvent event) {
            // TODO Auto-generated method stub
            float x = event.values[0];
            float y = event.values[1];
            float z = event.values[2];

            accX.setText("X: " + String.format("%.2f", x));
            accY.setText("Y: " + String.format("%.2f", y));
            accZ.setText("Z: " + String.format("%.2f", z));

            try {
                  server.send(new byte[] { (byte)0, (byte)x, (byte)y,
(byte)z });
            } catch (IOException e) {
                  Log.e("microbridge", "problem sending TCP message",
e);
            }

      }

      public void toggleLed(View view) {
            try {
                  server.send(new byte[] { (byte)1 });
            } catch (IOException e) {
                  Log.e("microbridge", "problem sending TCP message",
e);
            }
      }
}
```

8.3 Step 2B: Compiling code if you have the Source code

1. If you have the source code folder which I included in the folder with this tutorial, then take these steps to compile the code. First Start Eclipse, then select **File** > **New** > **Project**

2. Now this time select Android Application Project from existing code then click Next

Figure 16

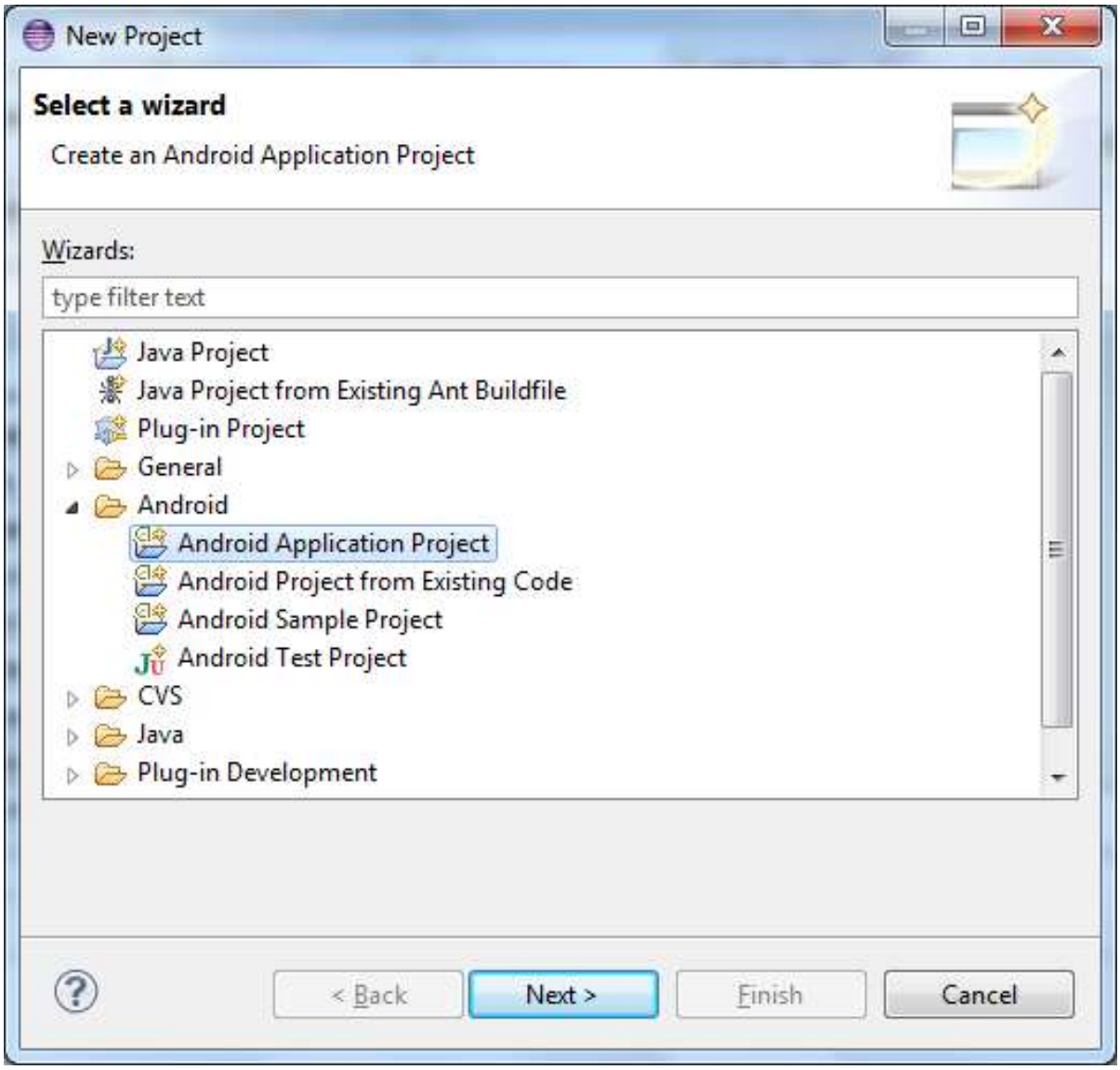

Figure 16. Screenshot of what to click when source code is provided

3. Now browse for where ever you extracted the Rofi.zip folder to and to check copy project to workspace as well. Then click finish.

Figure 17

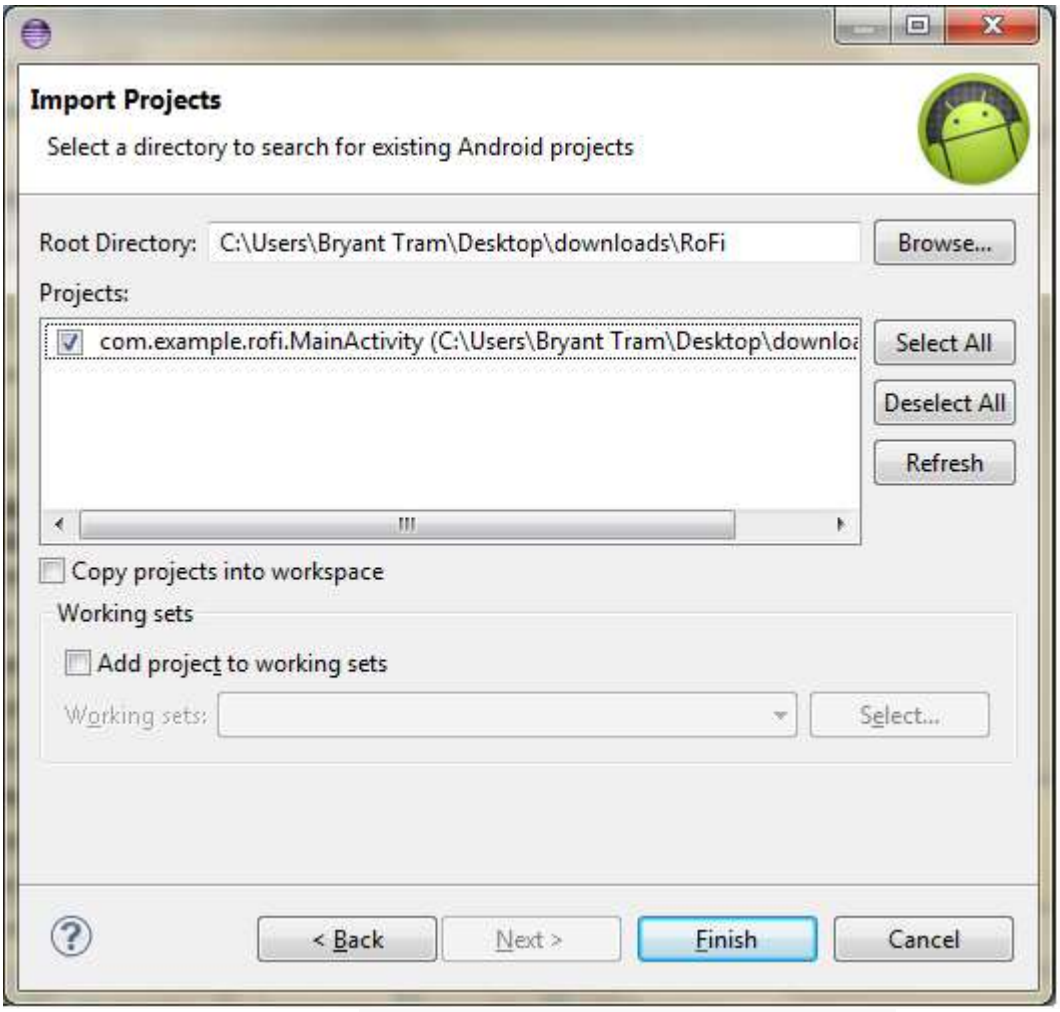

Figure 17. Select the source code location

4. Everything should be ready to be compiled so from there just hit run and the file will be compiled.

8.4 Compile and upload the Arduino code

1. Now we will compile the Arduino code and upload it onto our board. Open up the arduino.exe and open the Android.ino file included in the folder.

2. Click verify / compile under the sketch tab.

3. Now just upload your code to your board under files.

9 Connecting the wires/ Expected results

1. At this point your Arduino should still be connected to your computer, you should leave it connected.
2. Now connect the micro USB cable from the Android to the USB shield.
3. Run the App which in this case is Rofi
4. Now you should see three values which is constantly changing according the way you hold the device and everytime you click the toggleLED button, you should see the led at pin 13 turn on and off.

Figure 18

Figure 18. What my setup looks like after everything is installed and uploaded.

Appendix A Tips & Troubleshooting

- If an error pops up during runtime that MicroBridge server cannot start. Then check to make sure that android.permission.INTERNET is in the AndroidManifest.xml file. This is because MicroBridge requires internet APIs.
- If your Android application crashes when the orientation changes, then add android:configChanges="keyboardHidden|orientation" to the activity node in the AndroidManifest.xml file and add this method to your main activity file.

```
@Override
public void onConfigurationChanged(Configuration newConfig) {
  super.onConfigurationChanged(newConfig);
}
```

This mainly a bug in older Android versions 2.3 and below, but has been fixed in Android versions 3.0+.

References and Websites

Great guide on installing the SDK
http://www.talkandroid.com/guides/developer/android-sdk-install-guide/

Android Debug Bridge
http://developer.android.com/tools/help/adb.html

MicroBridge
http://code.google.com/p/microbridge/

Great YouTube tutorial videos on how to build your own android app
http://www.youtube.com/course?list=ECB03EA9545DD188C3&feature=plcp