How to develop your own app

It’s important that everything on the hardware side and also on the software side of our Android-to-serial converter should be as simple as possible. We have the advantage that it is possible to use all tutorials, tips and tricks which are applicable for network programming in java, because the tunnel through ADB is packing our data in a very nice way in a TCP stream.

More details to the Andropod Interface can be found at http://www.xdevelop.at/

1. The Development Environment

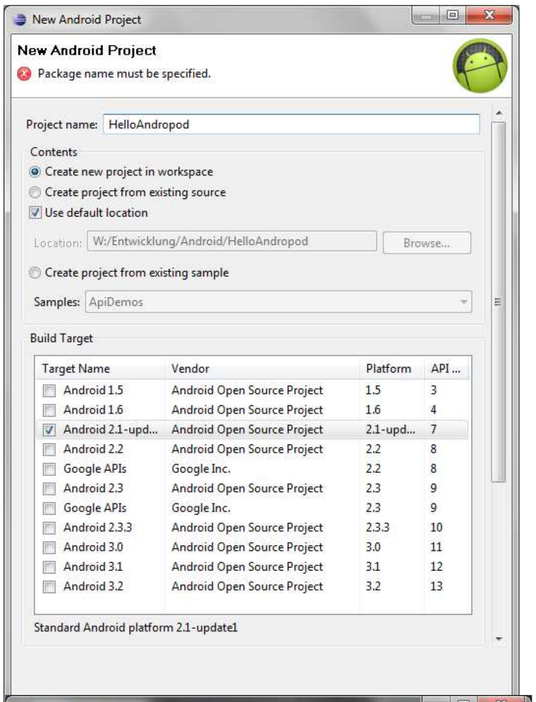
The easiest way to develop an app is to use Eclipse, which can be downloaded from http://www.eclipse.org/downloads/ for free. Eclipse is only prepared for the development of “normal” Java programs, but not for Java applications for Android. So it’s necessary to improve the program with the necessary features. This is done by installing the Android SDK, which can be downloaded from http://developer.android.com/sdk/index.html. At the end of the installation of the SDK the SDK manager will open automatically, which loads only the necessary packages. At the beginning of the installation a window should pop up, to start the installation you have to click “Accept All” and then “Install” - this may take some time. When the installation process is finished, an suitable Eclipse plugin must be installed. This works out of Eclipse via the menu item “Help” and “Install new software...”. The next step is to tell Eclipse to use plugins from the Google server. This happens when you enter the URL” https://dl-ssl.google.com/android/eclipse/“ under the item “Add”. After accepting the dialog some plugins will be listed in the tree - via “Select All” and “Next” they can be selected for the installation. After confirming the next dialog window and accepting all the licenses in the next window, the download and the installation of the plugins will start.

Now the development environment should be ready to create the first “Hello Andropod” application, which accesses the Andropod Interface.

2. Create the Android project

First it's necessary to create a new Android project in Eclipse. That should work easily via the menu item "New", "Android Project", if Eclipse was installed correct. As a result of this, a dialog opens, where the project settings can be found.

As the first necessary property you have to name the project you want to create, in this case with "HelloAndropod". This project will be a completely new project, so "Create new project in workspace" must be selected. To be able to select the project folder, deactivate the tick at "Use default location". The next step is to set the "Build target". That's the minimal Android version you are developing for. Here it's not important which version is chosen, because network connections which are needed for the app, are supported by all versions of Android (for the demo version 2.1 is chosen as minimal version). The other settings for the project can be found by scrolling down below the "Build target" section. In these settings everything should be left by the default values, only the "Package name" must be given. "org.andropod" is chosen here for the test app. Now you can already chose "Finish", because the other dialog windows aren't important.


New Android Project

New Android Project

Package name must be specified.

Project name: HelloAndropod

Contents

Create new project in workspace

Create project from existing source

Use default location

Location: W:/Entwicklung/Android/HelloAndropod

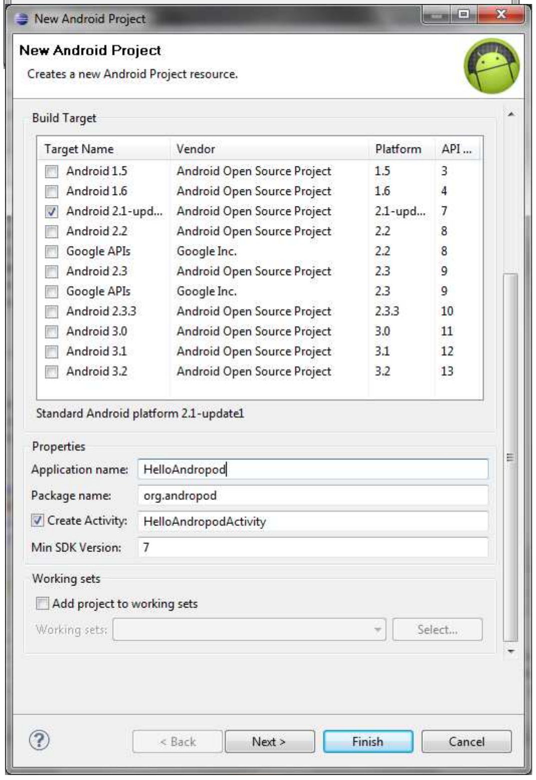
Create project from existing sample

Samples: ApiDemos

Build Target

	Target Name	Vendor	Platform	API ...
☐	Android 1.5	Android Open Source Project	1.5	3
☐	Android 1.6	Android Open Source Project	1.6	4
✓	Android 2.1-upd...	Android Open Source Project	2.1-upd...	7
☐	Android 2.2	Android Open Source Project	2.2	8
☐	Google APIs	Google Inc.	2.2	8
☐	Android 2.3	Android Open Source Project	2.3	9
☐	Google APIs	Google Inc.	2.3	9
☐	Android 2.3.3	Android Open Source Project	2.3.3	10
☐	Android 3.0	Android Open Source Project	3.0	11
☐	Android 3.1	Android Open Source Project	3.1	12
☐	Android 3.2	Android Open Source Project	3.2	13

Standard Android platform 2.1-update1


New Android Project

New Android Project

Creates a new Android Project resource.

Build Target

	Target Name	Vendor	Platform	API ...
☐	Android 1.5	Android Open Source Project	1.5	3
☐	Android 1.6	Android Open Source Project	1.6	4
✓	Android 2.1-upd...	Android Open Source Project	2.1-upd...	7
☐	Android 2.2	Android Open Source Project	2.2	8
☐	Google APIs	Google Inc.	2.2	8
☐	Android 2.3	Android Open Source Project	2.3	9
☐	Google APIs	Google Inc.	2.3	9
☐	Android 2.3.3	Android Open Source Project	2.3.3	10
☐	Android 3.0	Android Open Source Project	3.0	11
☐	Android 3.1	Android Open Source Project	3.1	12
☐	Android 3.2	Android Open Source Project	3.2	13

Standard Android platform 2.1-update1

Properties

Application name: HelloAndropod

Package name: org.andropod

✓ Create Activity: HelloAndropodActivity

Min SDK Version: 7

Working sets

☐ Add project to working sets

Working sets:

3. The project settings

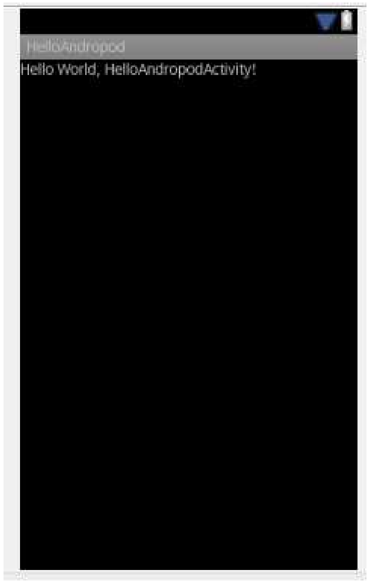
The file system of an Android project already includes many different files and folders right from the beginning, which are necessary and shouldn't be erased or renamed. The "src" folder contains in the subfolder "org.andropod" all Java files as well as the main-"activity", which will be shown when the app is started. In the "res" directory of the Android apps you can find many configuration settings as well as graphical user interface, which is packaged in XML files. The main configuration file, named "AndroidManifest.xml", can be found directly in the root directory of the project.

The access right management in Android is very strict, so each app is only allowed to do that, in what the user has consented by installing the app. The Andropod Interfaces uses a TCP connection for the data exchange, so it must be given a privilege for it to build up this connection. This can be done by opening the configuration file, which is named "AndroidManifest.xml" and by registering the permission "android.permission.INTERNET". This can be added in the tab "Permission" with "Add..", "Uses Permission".

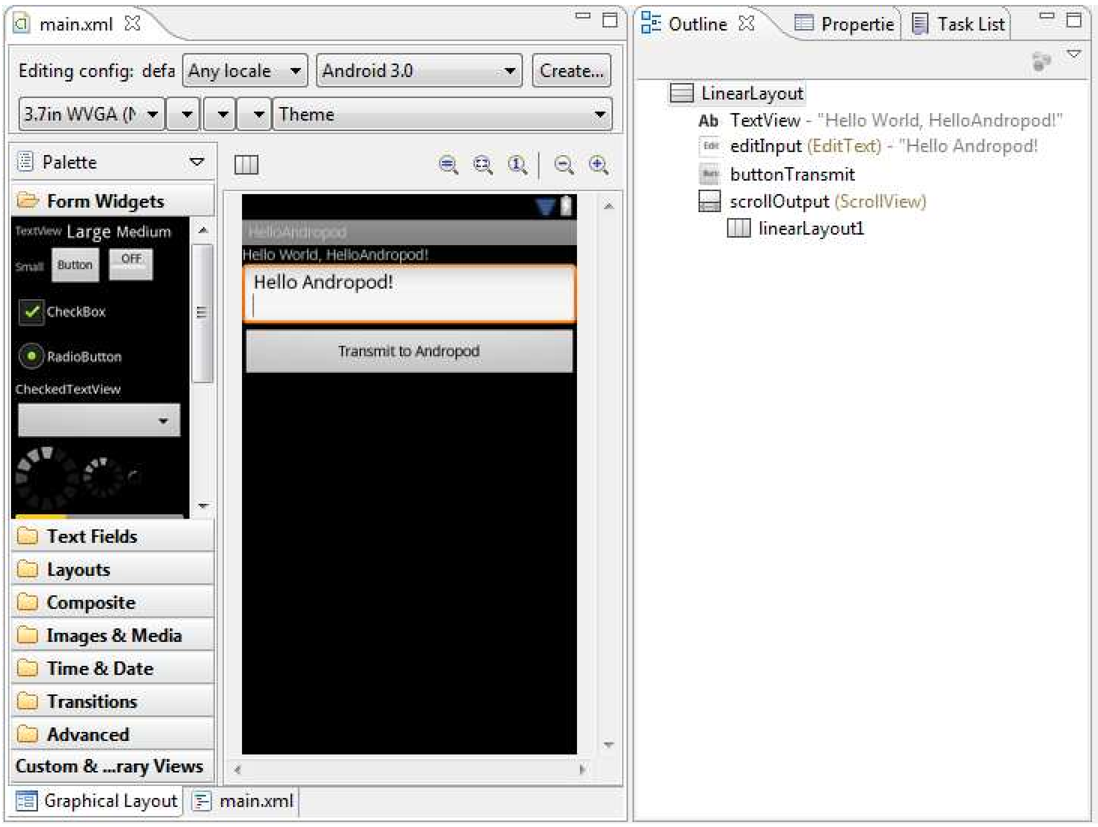
A pre-implemented connection class for the Andropod interface must be added after obtaining the rights for the internet or network access. It can be downloaded from http://www.xdevelop.at/files/AndropodConnection.zip. The file "AndropodConnection.java" should be unpacked in the "src/org.andropod" folder in the project directory. Afterwards Eclipse must be forced to refresh the project files. This happens by right clicking the project in "Package Explorer" and clicking "Refresh". Now you should see the newly added file in the package explorer now.

4. The GUI

After all preparations the main work starts now, so the first step is to create a suitable GUI. The “main.xml” file in the “res/layout” directory contains the description of the graphical user interface of the main activity. This file can be easily edited with a simple graphical editor and you don’t have to edit it manually.



You need an “EditText” input field with the ID „@+id/editInput“ for the test application, which receives the text that should be sent later. Setting the ID works by selecting the ID elements in the graphical editor in the “Properties” window. Additionally a “Button” with the ID„@+id/buttonTransmit“ is necessary, which is intended to send the data. To be able to scroll you should add a “ScrollView” with the ID „@+id/scrollOutput“ and the options “Layout height” and “Layout width”. Each of the options should be set to “fill_parent”. Now a “TextView” needs to be added to the “ScrollView” with the ID „@+id/textOutput“. The finished GUI should then look like this:



LinearLayout
- TextView - "Hello World, HelloAndropod!"
- editInput (EditText) - "Hello Andropod!
- buttonTransmit
- scrollOutput (ScrollView)
  - linearLayout1

5. The Sourcecode

Now at least minimal Java programming skills are very useful. To edit the main “Activity” the file “Hello.Andropod.java” must be opened. For developers who have only worked with desktop applications so far, it’s a bit difficult to understand on the beginning, how Android applications are started and stopped. That would go beyond the scope of this article to explain this, so we have to refer you to the documentation of Android: http://developer.android.com/reference/android/app/Activity.html.

The one thing to be mentioned is that every time you start the app, first the “onCreate” function is called, which normally set up the GUI. Subsequently the “onResume” function is called, which, for example, can be used for connecting the Andropod Interface. The function “onPause”, that should be used to close the connection, is called up when the app is paused, that means closing or putting the app in the background.

At the beginning of the main activity some variables need to be declared:

```
private AndropodConnection    andropod          = new AndropodConnection();
private Handler               handlerGui        = new Handler();
private Button                buttonTransmit    = null;
private EditText              editInput         = null;
private TextView              textOutput        = null;
private ScrollView            scrollOutput      = null;
private Thread                threadReader      = null;
```

These are, among other things, an object of the AndropodConnection class (andropod), a handler object for the GUI (handlergGui), a thread which is necessary for the reading the data of the interface (threadReader) and the elements of the GUI of the app (buttonTransmit, editInput, textOutput and scrollOutput).

The method “onCreate” is implemented with the following calls, after the new creating of the project:

```
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);
```

After these calls, which are responsible for the correct initialisation and set up of the GUI, the following code has to be added.

```
editInput                = (EditText)findViewById(R.id.editInput);
textOutput               = (TextView)findViewById(R.id.textOutput);
scrollOutput             = (ScrollView)findViewById(R.id.scrollOutput);
buttonTransmit           = (Button)findViewById(R.id.buttonTransmit);
```

Initially, you have to load references of the GUI elements to the member variables of the class.

```
buttonTransmit.setOnClickListener(new View.OnClickListener()
{
@Override
      public void onClick(View v) {
            andropod.write(editInput.getText().toString().getBytes());
      }
});
```

After that a “listener” has to be implemented for the “send” button, so that the button does something on a click. It’s easy to explain what the button does: it takes the text of the input field “editInput” and converts it into an array of bytes and sends it to the Andropod interface.

```
threadReader = new Thread(new Runnable() {
@Override
      public void run() {
            boolean wasConnected = false;
            try
            {
//Loop as long as the interface is open
                  while(andropod.isOpen())
                  {
                        boolean isConnected = andropod.isConnected();

                        //Check connection status
                        if(isConnected != wasConnected)
                        {
                              appendText(
isConnected?
"-- Device connected --\r\n":
"-- Device disconnected --\r\n"
);
                              wasConnected = isConnected;
                        }

                        if(isConnected)   //Receive data if connected
                        {
                              byte []buffer = new byte[1];
//if data has been read
                              if(andropod.read(buffer) > 0)
                              {
                                    appendText(new String(buffer));
//append read data
                              }
                        }
                        else              //Not connected
                        {
//Sleep some time to minimize CPU usage        Thread.sleep(100);
                        }
                  }
            }catch (Exception e){}
      }
});
```

```
threadReader.start();
```

The next and probably the most difficult part of the application is the reading thread. Its purpose is to read the data of the Andropod interface and to display them on the TextView. First you have to create a new thread and implement the method run that starts when the thread starts. Now an infinite loop runs in this run method or more precisely a loop that runs as long as the AndropodConnector is open. It continuously checkes, if a connection could be made to the Andropod interface. Should this be not the case, the thread sleeps for 100ms to avoid burdening the CPU. If the connection is open and a byte could be read (andropod.read), it will be appended with “AppendText” function to the GUI (the “AppendText” method will be discussed later in more detail). As an additional feature the thread checks if the connection status has changed, that means if a connection could be made this is shown with the AppendText method.

Actually most of the work is already done, only the “onResume” and “onPause” methods have to be built in. So we need a quick sidestep into object oriented programming.

```
@Override
public void onResume()
{
      super.onResume();       //Resume activity
      andropod.onResume();    //Resume andropod
}

@Override
public void onPause()
{
      super.onPause();        //Pause activity
      andropod.onPause();     //Pause andropod
}
```

“onPause” and “onResume” are already implemented on any activity because they are inherited and they will be called when you pause or when you resume the app. When you want to use them, the inherited methods aren’t enough and the functions have to be overwritten. So the keyword @Override has to stand before the function declaration. Now it has to be insured that the equivalent function of the super class has to be called (super.***). As the connection to the Andropod interface should be established or closed with these functions, the matching functions of the “AndropodConnection” have to be called up.

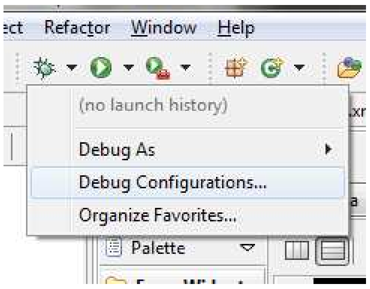
```
public void appendText(String text)
{
final String newText = text;

      handlerGui.post(new Runnable() {
            @Override
            public void run() {
                  textOutput.append(newText);
                  scrollOutput.fullScroll(ScrollView.FOCUS_DOWN);
```

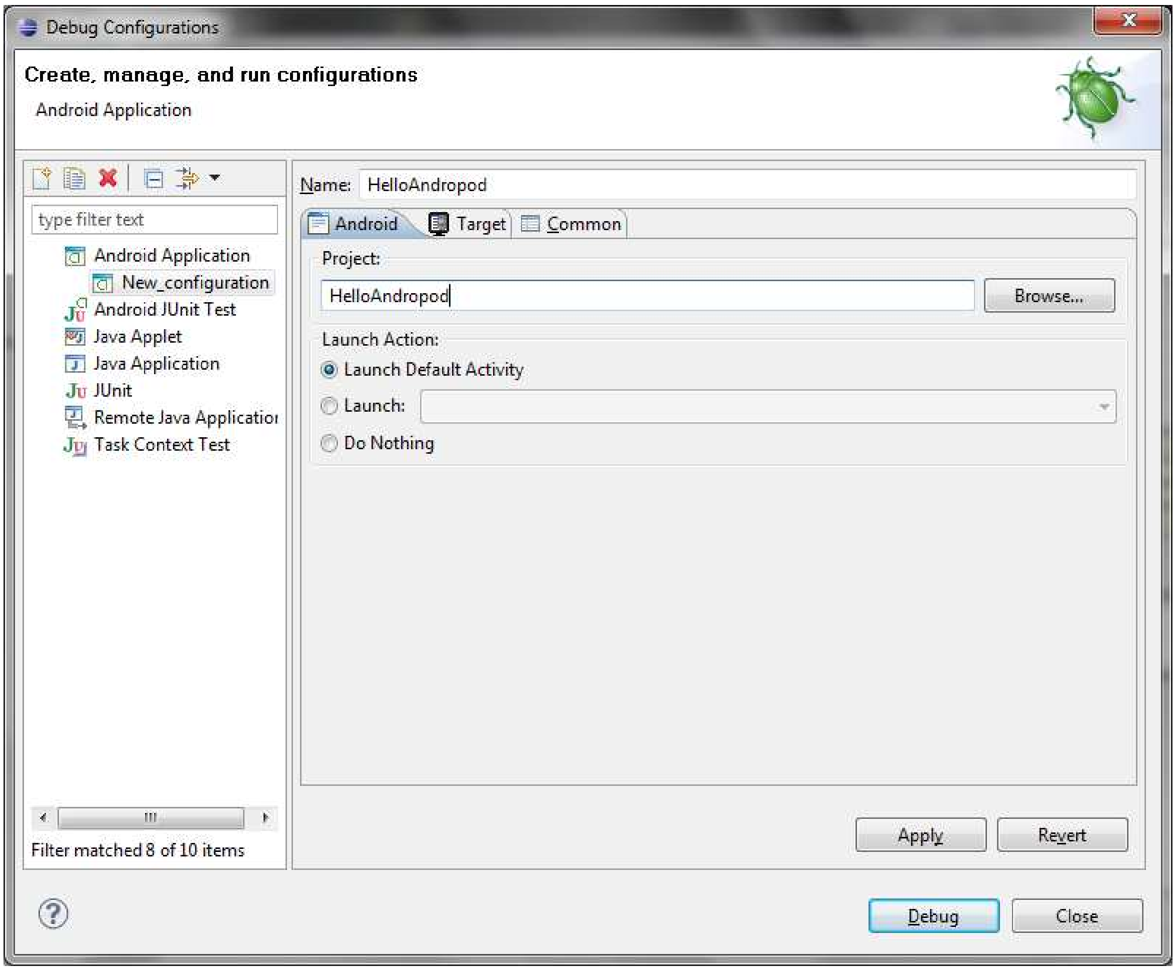
```
			}
		});
}
```

Finally you need the “AppendText” method, which writes the text on the GUI. It’s not possible to work directly on the GUI, because the data is read into a separate tread, so a so called “Handler” has to do this in the GUI thread.

6. Upload and Debug

To push the app to the phone, a suitable debug configuration has to be created in Eclipse. It works with a click on the arrow next to the "Debug"-button and selecting the sub item "DebugConfigurations...".




Debug Configurations

Create, manage, and run configurations

Android Application

type filter text

- Android Application
  - New_configuration
- Android JUnit Test
- Java Applet
- Java Application
- JUnit
- Remote Java Applicatio
- Task Context Test

Filter matched 8 of 10 items

Name: HelloAndropod

Android | Target | Common

Project: HelloAndropod

Launch Action:
[✓ Launch Default Activity]
Launch:
Do Nothing

You can create the debug configuration for the current project by double clicking on "AndroidApplication", which is now set as the standard configuration after closing the dialog.

Now the development of the first small sample application is finished and you only have to put the finished app on the phone. For that you have to plug the Andropod interface between the computer and the phone, so that the micro USB cable goes from the computer to the Andropod interface and a second cable from the Andropod interface to the Smartphone. The configuration jumpers have to be removed, because the Andropod interface has to be in the right mode, the debug mode. If everything works correctly, the Andropod interface (which is connected with the phone) will be detected as "Android ADB Interface" on the computer. By clicking on the "Debug"-button in Eclipse, the application is now transferred to the phone and should work now. The slow blinking Andropod interface should now light up permanently and the app should be visible on the phone.

Now the app sends all data that is entered in the textfield to the Andropod Interface when the "Send" button is clicked. All the data, that is received, is shown on the screen.

This project should show how easy it is to use the Andropod interface (the serial port for Android) for own apps. The entire test application can be downloaded from http://www.xdevelop.at/files/HelloAndropod.zip. All the source code may be used freely.