



Entwickeln einer eigenen App

Nicht nur auf der Hardware Seite unseres Android-zu-Seriell Wandlers sollte alles so einfach wie möglich sein, sondern auch auf der Software Seite. Da der Tunnel durch ADB unsere Daten eigentlich sehr schön wieder in einem TCP Stream verpackt, haben wir den Vorteil, dass bei der App Entwicklung alle Tutorials, Tipps und Tricks zum Thema Netzwerkprogrammierung anwendbar sind.

Mehr Details zum Andropod Interface finden Sie unter <http://www.xdevelop.at/>

1. Die Entwicklungsumgebung

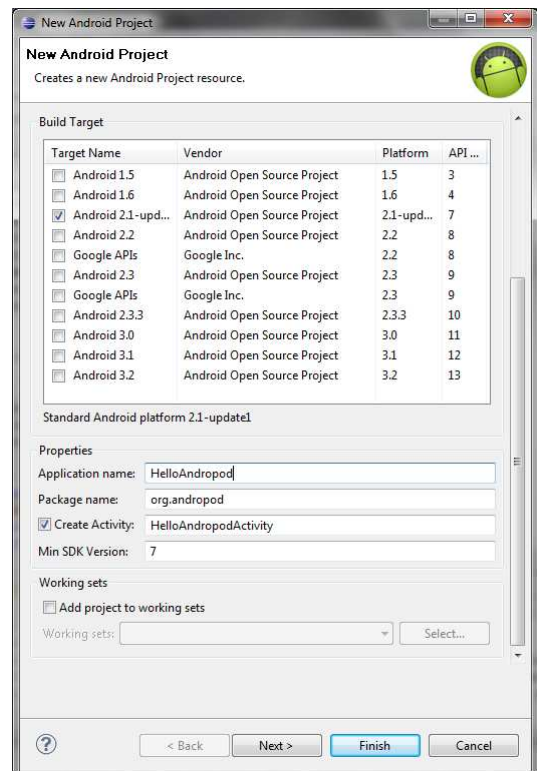
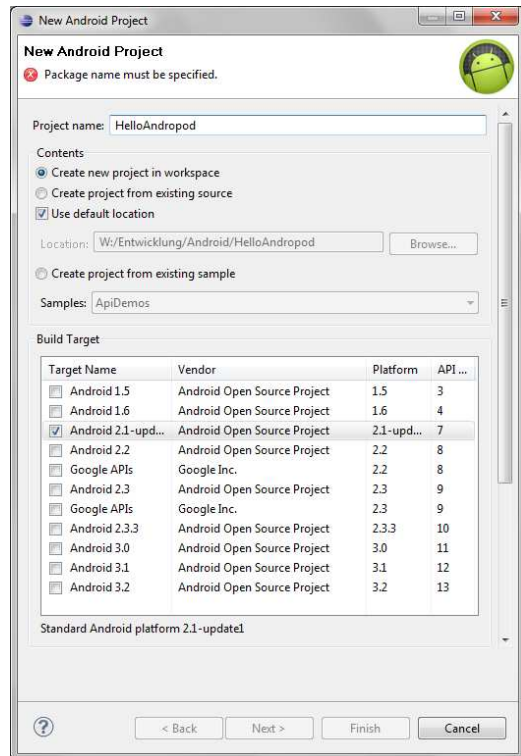
Als Entwicklungsumgebung ist der einfachste Weg auf Eclipse zu setzen, welches unter <http://www.eclipse.org/downloads/> gratis zum Download steht. Eclipse ist jedoch nur zur Entwicklung von „normalen“ Java Programmen gerüstet und nicht für Javaanwendungen für Android, daher muss es um die notwendigen Fähigkeiten erweitert werden. Dies geschieht durch die Installation des Android SDK, welches unter <http://developer.android.com/sdk/index.html> heruntergeladen werden kann. Am Ende der Installation des SDK wird automatisch der SDK Manager geöffnet, welcher nun die wirklich benötigten Pakete herunter lädt. !Gleich zu Beginn sollte ein Dialog geöffnet werden in welchem man mit „Accept All“ und einem Klick auf „Install“ den Download und die Installation startet – dies kann einige Zeit dauern. Ist diese Installation abgeschlossen, muss nun noch das passende Eclipse Plugin installiert werden, dies funktioniert aus Eclipse über den Menüpunkt „Help“ und „Install new Software...“. In diesem Dialogfenster muss Eclipse jetzt angegeben werden, dass es Plugins vom Google Server verwenden soll. Dies funktioniert in dem man unter „Add“ die URL „<https://dl-ssl.google.com/android/eclipse/>“ eingibt. Nach Bestätigen des Dialogfeldes erscheinen in der Liste einige Plugins in einem Baum aufgelistet – über „Select All“ und „Next“ können diese zur Installation ausgewählt werden. Nach dem Bestätigen des nächsten Dialogfeldes und dem Akzeptieren aller Lizenzen im darauf folgenden Fenster, startet der Download und die Installation der Plugins.

Nun sollte die Entwicklungsumgebung soweit fertig konfiguriert sein und der ersten „Hello Andropod“ Anwendung, die auf das Andropod Interface zugreift, steht nichts mehr im Weg.

2. Erstellen des Android Projektes

Zuerst muss ein neues Android Projekt in Eclipse angelegt werden, was, wenn Eclipse richtig installiert wurde, ganz einfach über den Menüpunkt „New“, „Android Project“ geschieht. Daraufhin öffnet sich ein Dialogfeld indem die Projekteinstellungen vorgenommen werden können.

Als erste verbindliche Angabe muss hier der Name des anzulegenden Projektes angegeben werden, für diesen Test verwenden wir „HelloAndropod“. Das Projekt soll ein komplett neues Projekt sein, das heißt „Create new project in workspace“ muss ausgewählt werden. Um den Projektordner festlegen zu können, kann das Häkchen bei „Use default location“ deaktiviert werden. Im nächsten Schritt muss das „Build Target“, das heißt die minimale Android Version für die entwickelt wird, festgelegt werden. Hier ist es generell nicht von großem belang welche Version gewählt wird, da Netzwerkverbindungen welche für die App benötigt werden von allen Android Versionen unterstützt werden (für die Demo wird Version 2.1 als minimal Version gewählt). Die weiteren Einstellungen für das Projekt sind nach dem Herunterscrollen unter der Auswahl des „Build Target“ zu finden. Bei diesen Einstellungen sollte alles bereits nach Eingabe des Projektnamens voreingestellt sein, nur der „Package name“ muss noch angegeben werden, für die Test App wird hier „org.andropod“ gewählt. Nun kann bereits mit „Finish“ bestätigt werden, da die weiteren Dialogfelder nicht weiter relevant sind.





3. Die Projekteinstellungen

Das Dateisystem eines Android Projektes beinhaltet bereits vom Start weg viele verschiedene Dateien und Ordner, welche notwendig sind und nicht gelöscht oder umbenannt werden sollten. Der Ordner „src“ beinhaltet im Unterordner „org.andropod“ nun alle Java Dateien wie auch die Haupt-„Activity“, die beim Starten der App aufgerufen wird. Bei Android Apps sind viele Konfigurationseinstellungen sowie grafische Oberflächen in XML Dateien verpackt, welche im „res“ Verzeichnis zu finden sind. Die Haupt-Konfigurationsdatei, mit dem Namen „AndroidManifest.xml“, findet man direkt im Stammverzeichnis des Projektes.

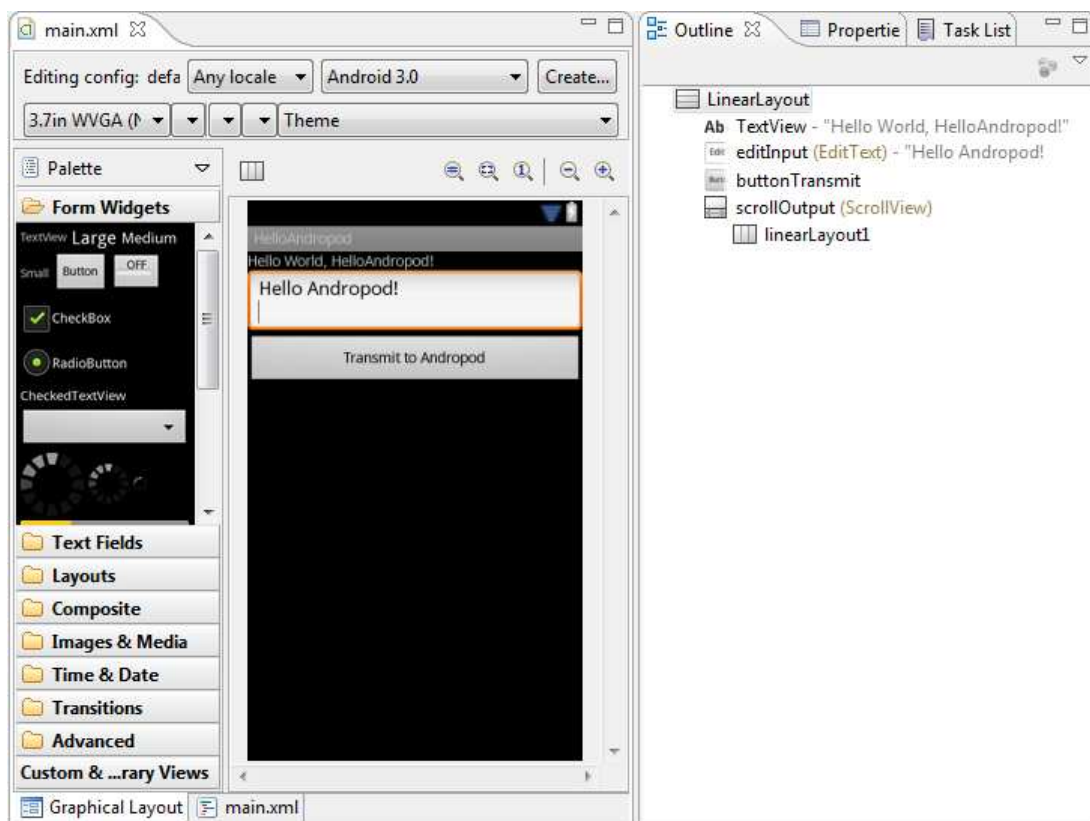
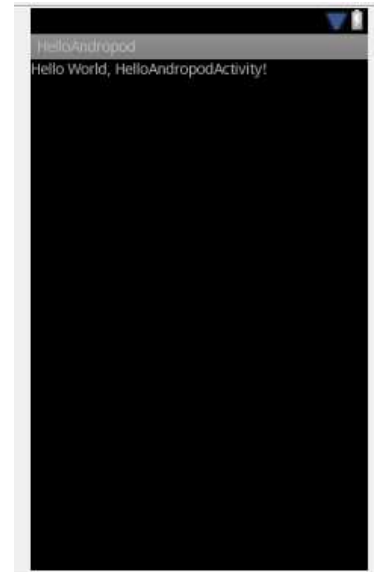
Das Rechte System in Android ist sehr strikt, sodass jede App wirklich nur das darf, in was der Benutzer bei der Installation auch wirklich eingewilligt hat. Da das Andropod Interface eine TCP Verbindung zum Datenaustausch benutzt, muss dem App nun im ersten Schritt das Privileg gegeben werden diese aufbauen zu dürfen. Dies geschieht durch Öffnen der Konfigurationsdatei „AndroidManifest.xml“ und durch Registrieren der Berechtigung „android.permission.INTERNET“. Im Reiter „Permission“ mit „Add..“, „Uses Permission“ kann diese hinzugefügt werden.

Nach dem Erlangen der Rechte für den Internet- bzw. Netzwerkzugriff muss nun noch eine bereits vorgefertigte Verbindungsklasse für das Andropod Interface eingefügt werden, welche von <http://www.xdevelop.at/files/AndropodConnection.zip> heruntergeladen werden kann. Die Datei „AndropodConnection.java“ sollte in den Ordner „src/org.andropod“ im Projektverzeichnis entpackt werden. Anschließend muss Eclipse noch dazu gebracht werden, die Projektdateien neu einzulesen, was durch einen Rechtsklick auf das Projekt im „Package Explorer“ und dem anschließenden Klick auf „Refresh“ geschieht. Nun sollte die neu eingefügte Datei in der Übersicht erscheinen.

4. Die GUI

Nach allen Einstellungen im Vorfeld beginnt nun die eigentliche Arbeit, zunächst das Erstellen einer passenden GUI. Die Datei „main.xml“ im „res/layout“ Verzeichnis beinhaltet die Beschreibung der grafischen Oberfläche der Hauptaktivität. Diese Datei muss nicht von Hand bearbeitet werden, sondern kann durch einen einfachen grafischen Editor leicht editiert werden.

Für die Test Anwendung wird ein „EditText“ Eingabefeld mit der ID „@+id/editInput“ benötigt, welches später den Text aufnimmt der gesendet werden soll. Das Einstellen der ID funktioniert nach dem Auswählen des GUI Elementes im Grafischen Editor im „Properties“ Fenster. Außerdem wird ein „Button“, der zum Abschicken der Daten gedacht ist, mit der ID „@+id/buttonTransmit“ benötigt. Um scrollen zu können, sollte eine „ScrollView“ mit der ID „@+id/scrollOutput“ und den Optionen „Layout height“ und „Layout width“ jeweils auf „fill_parent“ gesetzt, eingefügt werden. In diese ScrollView muss jetzt noch eine „TextView“ mit der ID „@+id/textOutput“ gezogen werden. Die fertige GUI sollte anschließend folgendermaßen aussehen:





5. Der Sourcecode

Nun geht es noch einen Schritt tiefer auf Java Ebene, das heißt zumindest minimale Java-Programmierkenntnisse sind jetzt von großem Nutzen. Zum Editieren der Haupt „Activity“ muss die Datei „HelloAndropod.java“ geöffnet werden. Für Entwickler die bis jetzt nur mit Desktop Anwendungen gearbeitet haben, ist es zu Beginn oft sehr schwer zu begreifen, wie Android Applikationen gestartet und beendet bzw. angehalten werden. Da dies den Rahmen dieses Artikels sprengen würde, müssen wir hier auf die Dokumentation von Android verweisen: <http://developer.android.com/reference/android/app/Activity.html>.

Soviel sei gesagt, bei jedem Starten der App wird zunächst die „onCreate“ Funktion aufgerufen, welche im Normalfall die GUI aufbaut. Anschließend wird die „onResume“ Funktion aufgerufen, welche zum Beispiel zum Verbinden des Andropod Interface s genutzt werden kann. Beim Pausieren, das heißt dem Schließen oder in den Hintergrund stellen der App, wird die Funktion „onPause“ aufgerufen, die zum Schließen der Verbindung verwendet werden sollte.

Im Kopf der Hauptaktivität müssen nun einige Variablen deklariert werden:

```
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;
```

Das sind unter anderem ein Object der AndropodConnection Klasse (andropod), ein Handler Objekt für die GUI (handlerGui), ein Thread der zum Lesen der Daten vom Interface gedacht ist (threadReader), sowie die Elemente der GUI der App (buttonTransmit, editInput, textOutput und scrollOutput).

Die Methode „onCreate“ ist nach dem neuen Erstellen des Projektes bereits implementiert, und zwar mit folgenden Aufrufen:

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

Nach diesen Aufrufen, welche für das richtige Intialisieren und Setzen der richtigen GUI verantwortlich sind, muss nun der Code der Test App eingefügt werden.

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



Zunächst müssen Referenzen zu den GUI Elementen auf Membervariablen der Klasse geladen werden.

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

Im Anschluss daran muss ein „Listener“ auf den „Senden“ Button gelegt werden, damit der Button bei einem Klick auch wirklich etwas macht. Was er macht ist einfach zu erklären, er nimmt den Text des Eingabefeldes „editInput“ und verwandelt ihn in ein Array aus einzelnen Bytes und sendet dieses an das 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();
```

Der nächste und wohl auch schwierigste Teil der Anwendung ist der Lese Thread, welcher grob gesagt die Aufgabe hat auf Daten vom Andropod Interface zu warten und diese an die TextView in der GUI anzuhängen. Zunächst wird ein neuer Thread angelegt und die Methode run, welche beim Start des Threads ausgeführt wird, implementiert. In dieser run-Methode läuft nun eine Endlosschleife oder genauer gesagt eine Schleife die solange läuft, wie der AndropodConnector geöffnet ist. In dieser Schleife wird nun kontinuierlich überprüft ob eine Verbindung zum Andropod Interface hergestellt werden konnte. Sollte dies nicht der Fall gewesen sein, so schläft der Thread für 100ms um die CPU nicht zu belasten. Sollte jedoch eine Verbindung erfolgreich zu Stande gekommen sein, so wird versucht ein Byte vom Andropod Interface zu lesen (andropod.read). Konnte dieses Byte gelesen werden, wird es an die GUI mit „AppendText“ angehängt (auf die „AppendText“ Methode wir später noch genauer eingegangen). Als Zusatzfeature überprüft dieser Thread auch gleich noch ob sich der Verbindungsstatus geändert hat, das heißt ob eine Verbindung hergestellt werden konnte oder nicht und gibt dies ebenfalls mittels AppendText aus.

Nun ist eigentlich die Hauptarbeit schon fast getan und es müssen, wie zuvor bereits kurz erwähnt, nur noch die „onResume“ und „onPause“ Methoden eingebaut werden, was zu einem kleinen Abstecher in objektorientierte Programmierung führt.

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

Diese ominösen Funktionen „onPause“ und „onResume“ werden, wie erwähnt, beim Pausieren bzw. beim Wiederaufnehmen der App aufgerufen und sind standardgemäß bereits in jeder Activity implementiert, da sie mit vererbt werden. Sollen sie jedoch bewusst benutzt werden reichen die vererbten Methoden nicht aus, das heißt sie müssen überschrieben werden, daher auch das Schlüsselwort @Override vor der Funktionsdeklaration. In den überschriebenen Funktionen muss nun sichergestellt werden, dass auch das entsprechende Pendant in der Überklasse wieder aufgerufen wird (super.***). Da die Verbindung zum Andropod Interface ja mittels dieser Funktionen getrennt beziehungsweise aufgebaut werden



soll, müssen die passenden Funktionen der „AndropodConnection“ Klasse aufgerufen werden.

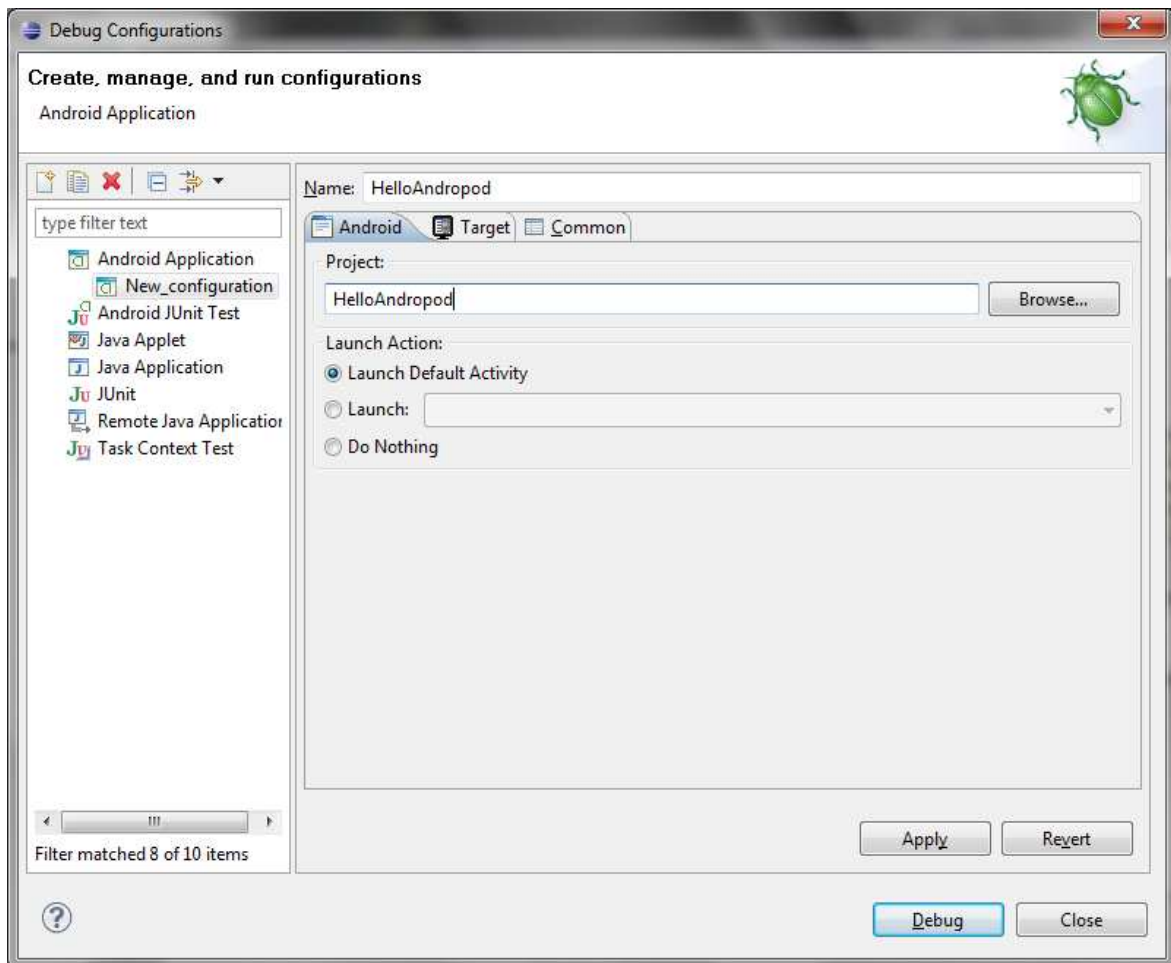
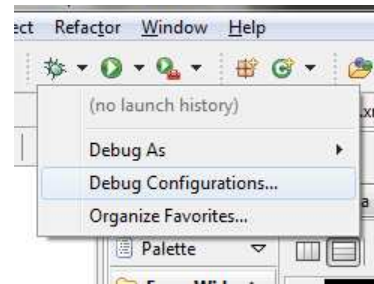
```
public void appendText(String text)
{
    final String newText = text;

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


Zuletzt wird noch die „AppendText“ Methode benötigt, mit der Text auf die GUI geschrieben wird. Da die Daten in einem separaten Thread eingelesen werden, kann hier die GUI nicht direkt bearbeitet werden, sondern ein sogenannter „Handler“ muss dies im GUI Thread erledigen.

6. Hochladen und Debuggen

Um die App nun auf das Telefon übertragen zu können, muss in Eclipse noch eine passende Debug Konfiguration erstellt werden. Die Funktioniert durch einen Klick auf den Pfeil neben dem „Debug“-Button, und auswählen des „Debug Configurations...“ Unterpunktes.



Im Dialogfeld, das sich daraufhin öffnet, wird die Debug Konfiguration für das aktuelle Projekt einfach durch einen Doppelklick auf „Android Application“ erstellt und ist nach dem schließen nun als Standardkonfiguration eingestellt.

Nun ist das Ende der Entwicklung der ersten kleinen Beispielanwendung eigentlich schon erreicht und die fertige Test-App muss nur noch auf das Telefon übertragen werden. Hierzu wird das Andropod Interface zwischen PC und Telefon gesteckt, sodass ein Micro-USB Kabel vom PC zum Andropod Interface und ein zweites USB Kabel vom Andropod Interface zum Smartphone geht. Damit das Andropod Interface im richtigen Modus, dem Debug Modus ist, müssen alle Konfigurations-Jumper entfernt werden. Funktioniert alles richtig, sollte das Andropod Interface (mit dem Handy angeschlossen) am PC als „Android ADB Interface“ erkannt werden. Mittels Klick auf den „Debug“-Button  in Eclipse wird die Applikation nun auf das Telefon überspielt und ausgeführt. Nun sollte das zuvor langsam



blinkende Andropod Interface dauerhaft leuchten und die App sollte am Telefon sichtbar sein.

Nun kann in der App mit einem Klick auf den „Senden“ Button die eingegebene Nachricht über das Andropod Interface gesendet werden. Alle Daten die empfangen werden, werden darunter angezeigt.

Dieses Projekt soll veranschaulichen wie einfach es ist, dass Andropod Interface, den seriellen Port für Android, in eigenen Apps zu benutzen. Die gesamte hier behandelte Test-Applikation kann unter <http://www.xdevelop.at/files/HelloAndropod.zip> heruntergeladen werden und alle Quellcodeteile dürfen frei verwendet werden.