INF2C – groep 3

**Projectleden**

Tedjan Hulshof

Dennis Hartmann

Justin Bakker

Roy Heidotting

Melvin Bos

Tim Wennekes



**Hoe komt de verbinding tussen de hoofdunit en tablet tot stand?**

Inhoud

[Methode 3](#_Toc346197330)

[Hoe komt de verbinding tussen de hoofdunit en de tablet tot stand? 4](#_Toc346197331)

[1. ADK Connection over USB 4](#_Toc346197332)

[2. Accessory USB Code 4](#_Toc346197333)

[3. USB via serial 9](#_Toc346197334)

[Bibliografie 9](#_Toc346197335)

# Methode

Voor dit onderzoek is gebruik gemaakt van literatuuronderzoek. Er is gekeken naar bestaande code dat hergebruikt kan worden. Tevens zijn er praktijktests gedaan met een arduino en een Android tablet.

# Hoe komt de verbinding tussen de hoofdunit en de tablet tot stand?

## 1. ADK Connection over USB

The ADK 2012 app and hardware accessory can also use a USB connection to communicate, similar to the original ADK.

## 2. Accessory USB Code

The ADK library takes care of most of the implementation details for a USB connection, the accessory code must make a few calls to initialize USB connectivity, including setting the accessory identification strings:

ADK L;

void setup() {

L.adkInit();  
 L.usbSetAccessoryStringVendor(...);  
 L.usbSetAccessoryStringName(...);  
 L.usbSetAccessoryStringLongname(...);  
 L.usbSetAccessoryStringVersion(...);  
 L.usbSetAccessoryStringUrl(...);  
 L.usbSetAccessoryStringSerial(...);

L.usbStart();  
  
}

Note: The identification strings must match the USB accessory filter settings specified in the connecting Android application,otherwise the application cannot connect with the accessory.

Once USB is enabled with code shown above, the accessory listens for connection requests. The ADK library handles listening and connection details, so the accessory calls ADK::adkEventProcess() once during each loop execution:

void loop(void)  
{  
 ...  
 L.adkEventProcess(); //let the adk framework do its thing  
 ...  
}

The accessory must then check for a live USB connection to process commands and send messages. Here is a summary of the relevant code:

void loop() {  
  
 if (L.accessoryConnected()) {  
  
 int recvLen = L.accessoryReceive(msg, sizeof(msg));  
  
 if (recvLen > 0) {  
 ... // process message  
 }  
  
 L.accessorySend(outmsg, outmsgLen);

}  
  
 L.adkEventProcess();  
  
}

For more details, review the implementations of these methods in the clock.ino sketch.

Android App USB Code

In the ADK 2012 Android app, the code for handling USB connections is encapsulated in a UsbConnection class. This class sets up a [BroadcastReceiver](http://developer.android.com/reference/android/content/BroadcastReceiver.html) to listen for USB events and then attempts to connect when a matching connection event is received. Here is a summary of the relevant code:

import com.android.future.usb.UsbAccessory;  
import com.android.future.usb.UsbManager;  
  
mUSBManager = UsbManager.getInstance(this);  
UsbAccessory acc = mUSBManager.getAccessoryList()[0];

if (!mUSBManager.hasPermission(acc)) return;

The ADK 2012 app uses the support library to implement the USB accessory connections, in order to support devices running Android 2.3.4 (API Level 10). If you only need to support Android 3.1 (API Level 12) and higher devices, you can replace the first 4 lines the following code:

import android.hardware.usb.UsbAccessory  
import android.hardware.usb.UsbManager

mUSBManager = (UsbManager) getSystemService(Context.USB\_SERVICE);  
  
UsbAccessory acc = (UsbAccessory)  
 intent.getParcelableExtra(UsbManager.EXTRA\_ACCESSORY);

Note that the app only receives events when the USB accessory identification information matches the information in theres/xml/usb\_accessory\_filter.xml file, referenced by the application’s manifest statement:

<meta-data   
 android:name="android.hardware.usb.action.USB\_ACCESSORY\_ATTACHED"   
 android:resource="@xml/usb\_accessory\_filter" />

Connections from other USB devices are not received by the ADK 2012 accessory.

Once the connection is established, the app can communicate with the accessory through file input and output streams, as shown in the following example code:

ParcelFileDescriptor mFD = mUSBManager.openAccessory(acc);

if (mFD != null) {

FileDescripter fd = mFD.getFileDescriptor();  
 mIS = new FileInputStream(fd); // use this to receive messages  
 mOS = new FileOutputStream(fd); // use this to send commands

}

Review the UsbConnection.java file provided in the ADK 2012 source code for more implementation details.

AndroidAccessory(const char \*manufacturer,  
 const char \*model,  
 const char \*description,  
 const char \*version,  
 const char \*uri,  
 const char \*serial);

For example, the DemoKit app that comes with the ADK identifies itself as version 1.0 of the Google DemoKit accessory:

AndroidAccessory acc("Google, Inc.",  
 "DemoKit",  
 "DemoKit Arduino Board",  
 "1.0",  
 "http://www.android.com",  
 "0000000012345678");

Your own app might looks something like this:

AndroidAccessory acc("My App Company",  
 "CoolAccessory",  
 "CoolAccessory Arduino Board",  
 "1.0",  
 "http://www.example.com/CoolAccessory",  
 "0000000012345678");

The three key ones you’ll need to remember are manufacturer, model and version as you’ll need to define these in you’re Android app so that the device can notify your app when your accessory is connected.

When you create your Android app, there’s a few steps you need to take. First, not every Android-powered device supports the USB accessory APIs. In fact, the Android Open Accessory ADK is only supported by Android OS 3.1 and 2.3.4 platforms. To identify that your app has USB features, you need to include a <uses-feature> element in your manifest to declare that your application uses the android.hardware.usb.accessory feature.

<uses-feature android:name="android.hardware.usb.accessory" />

Next, if you’re using the add-on library, add a <uses-library> element with com.android.future.usb.accessory for the USB accessory library.

<uses-library android:name="com.android.future.usb.accessory" />

and set the minimum SDK of the application to API Level 10. If you’re using theandroid.hardware.usb package then the <uses-library> isn’t necessary and you only need to set the minimum SDK to 12 .

Lastly, you’ll want your application to be notified when your accessory is attached. Specify an <intent-filter> and <meta-data> element pair for the android.hardware.usb.action.USB\_ACCESSORY\_ATTACHED intent in your main activity. Overall your manifest should look something like this:

<manifest ...>

    <uses-feature android:name="android.hardware.usb.accessory" />

    <uses-sdk android:minSdkVersion="<version>" />

    ...

    <application>

      <uses-library android:name="com.android.future.usb.accessory" />

        <activity ...>

            ...

            <intent-filter>  
                <action  
android:name="android.hardware.usb.action.USB\_ACCESSORY\_ATTACHED" />  
            </intent-filter>  
  
            <meta-data android:name="android.hardware.usb.action.USB\_ACCESSORY\_ATTACHED"  
                android:resource="@xml/accessory\_filter" />  
  
        </activity>

    </application>

</manifest>

The <meta-data> element references an XML resource file that includes the manufacturer, model and version you defined earlier in your Arduino sketch. Create an accessory\_filter.xml resource file in the res/xml/ directory with a <usb-accessory> element to identify your accessory.

<resources>  
 <usb-accessory model="CoolAccessory" manufacturer="My App Company" version="1.0"/>  
</resources>

Now, when you connect your accessory to the device android will send an intent to open an appropriate application. The best part is that more than one application can respond to a given intent so multiple apps could optionally respond for the same accessory.

## 3. USB via serial

Het is ook mogelijk om een verbinding tussen de tablet en hoofdunit te maken op serieel niveau. Dit houdt in dat alle data via een seriële poort verstuurd en ontvangen kan worden. Het is mogelijk om in plaats van de ADK een seriële verbinding te maken tussen de tablet PC en de interface van de Arduino. Hierdoor is het ook mogelijk af te zien van de Arduino Mega.

Data dat op serieel niveau verstuurd wordt word dan in series met bits verstuurd achter elkaar. Dit is een goede manier om op korte afstand data te verplaatsen met snelheden tot 115200 baud.

De tablet is voorzien van een USB host waardoor het mogelijk is om een kabel tussen de Arduino en de tablet te leggen. Softwarematig kan er gebruik gemaakt worden van de FTDriver. Deze library is vrij verkrijgbaar en makkelijk te implementeren in een applicatie.

# Bibliografie

*Github*. (sd). Opgeroepen op 01 16, 2013, van https://github.com/ksksue/FTDriver/blob/master/README.md

Google Inc. (2012). *Accessory Development Kit 2012 Guide*. Opgeroepen op 11 28, 2012, van http://developer.android.com/tools/adk/adk2.html

Sambells, J. (2011, 15 5). *Identifying Your Android USB Accessory*. Opgeroepen op 12 14, 2012, van http://jeffreysambells.com/2011/05/15/identifying-your-android-usb-accessory