Leveraging the Android Accessory Protocol

Gary Bisson

Adeneo Embedded

gbisson@adeneo-embedded.com

Android Builders Summit 2013

Session Overview

- Introduction to Android Open Accessory
- Protocol specifications
- Accessory Development Kit
- Software implementation
- Demonstrations

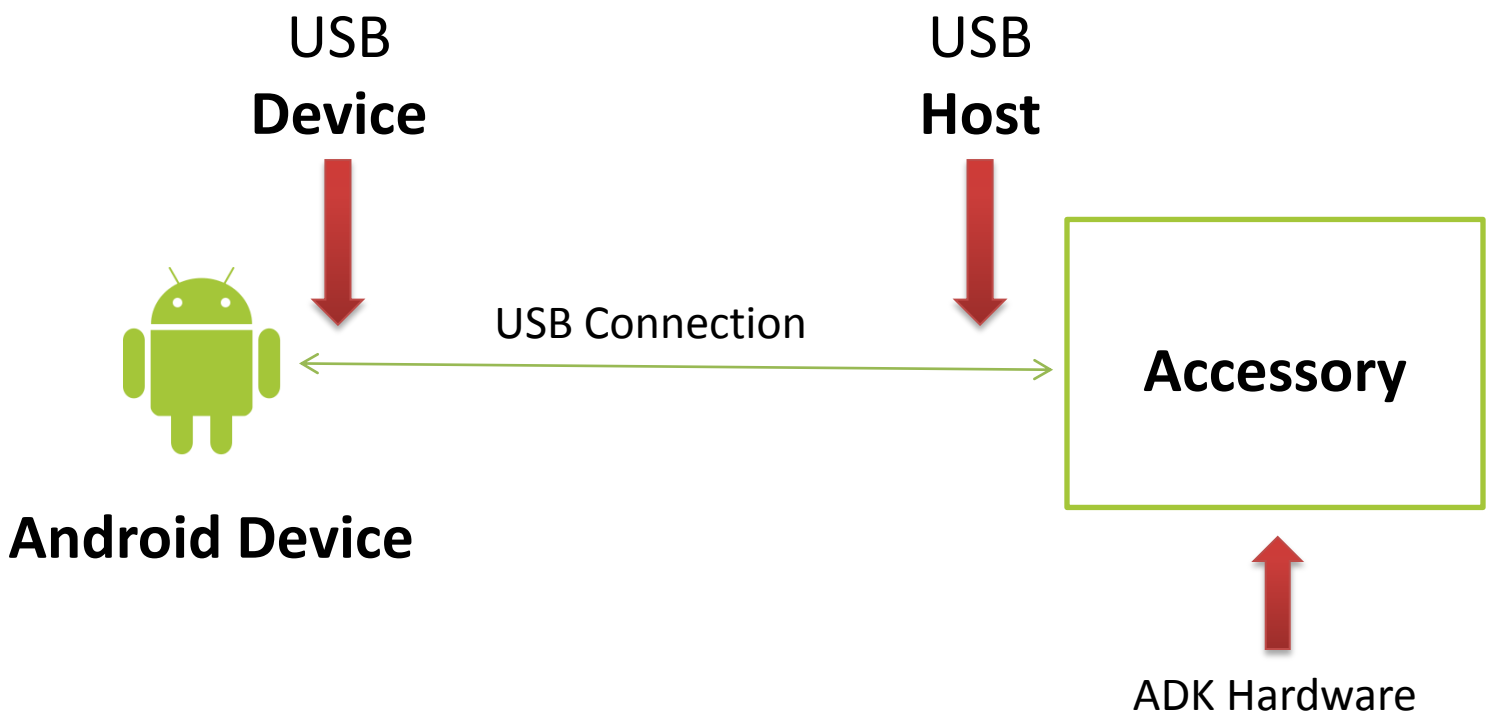
Who am I?

- Software engineer at Adeneo Embedded (Bellevue, WA)
  - Linux, Android
  - Main activities:
    - BSP adaptation
    - Driver development
    - System integration

Introduction to AOA

- Allows USB hardware to interact with an Android-powered device
  - No need for the Android device to act as **USB Host**
  - **Standard API**
- Introduced in **Android 3.1** (API level 12)
  - Backported to Android 2.3.4 (API level 10)
- Version **2.0** released with **Android 4.1**

Introduction to AOA



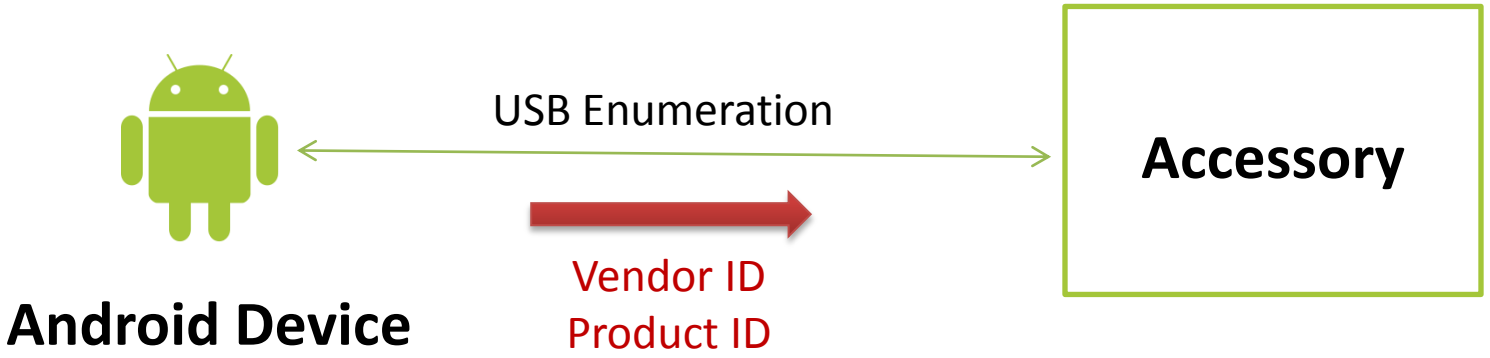
AOA 1.0 Protocol

- Android Open Accessory 1.0 is a protocol that allows an **Android device** to interact with an **Android USB accessory** in a special accessory mode.

- Basically there are **four steps** to initiate the communication.

AOA 1.0 Protocol

1. Wait for and detect connected devices



AOA 1.0 Protocol

2. Determine the device's accessory mode support

AOA 1.0 Protocol

- The accessory must check the Vendor & Product ID's of the connected device
  - Possibility to target some devices
  - Detection of devices already in Accessory mode

- If it is already in accessory mode then:
  - Vendor ID = **0x18D1** (**Google**)
  - Product ID = 0x2D00 | | 0x2D01

AOA 1.0 Protocol

- Accessory mode product IDs
  - **0x2D00** – Supports Accessory Mode
    - 1 interface with 2 bulk endpoints
  - **0x2D01** – Supports Accessory Mode + ADB
    - 2 interfaces with 2 bulk endpoints each

- If it is not in accessory mode then Accessory mode support can be asked:
  - Send a "**Get Protocol**" request (**51**) on endpoint 0
    - AOA Version (1.0 or 2.0) is returned by Android device

AOA 1.0 Protocol

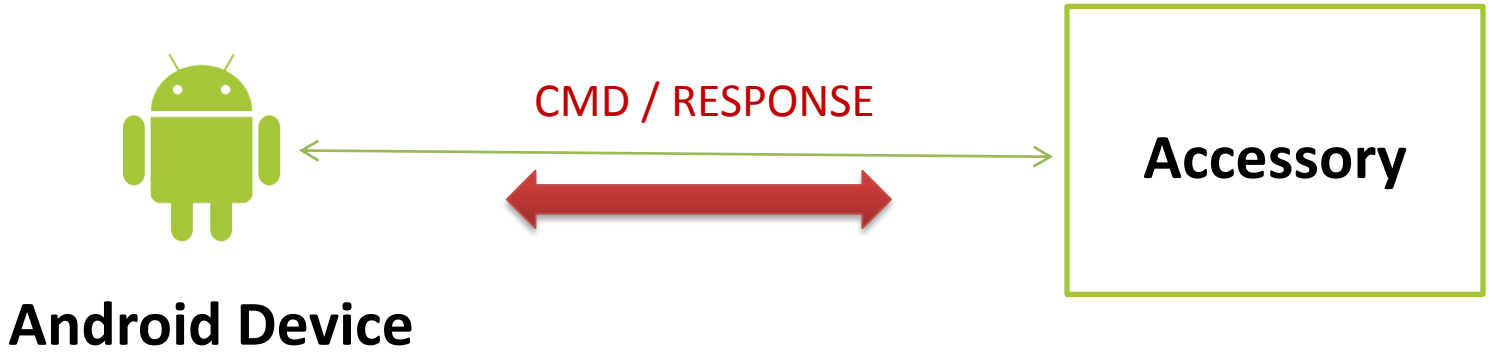
3. Attempt to start the device in accessory mode

AOA 1.0 Protocol

- If the AOA Version # is okay, then send string's identifying our ADK to the device
  - Send "**Identity**" request (**52**) for each identifier:
    - Manufacturer
    - Model Name
    - Description
    - Version
    - URL
    - Serial Number
  - Send "**Start Accessory**" request (**53**) to request the Android device re-introduce itself on the bus in accessory mode

AOA 1.0 Protocol

4. Establish communications



AOA 1.0 Protocol

- The accessory must obtain the Bulk endpoints and be prepared to initiate communication with the device.

From this point the communication is defined by the ADK Developer

AOA 2.0 Protocol

- AOA 2.0 was released at Google I/O end of June 2012 alongside **Jelly Bean**

- Two new features:
  - **Audio Output**
  - **Accessory as HID**

AOA Protocol version 2.0

- **Audio output**:
  - From an Android device to an accessory
  - Standard USB audio class interface (ISO)
  - Only supports 2 Channel, 16-bit PCM @ 44100KHz

- To enable the audio support, the accessory must send a new USB control request:
  - "**Audio Support**" request (**58**)

AOA Protocol version 2.0

- **HID support**:
  - Registers one or more USB HID with to the Android device.
  - Reverses the direction of communication for typical USB HID (Host <-> Device).
  - Uses USB control requests:
    - ACCESSORY_REGISTER_HID
    - ACCESSORY_UNREGISTER_HID
    - ACCESSORY_SET_HID_REPORT_DESC
    - ACCESSORY_SEND_HID_EVENT

AOA Protocol version 2.0

- Compatible with original AOA 1.0 protocol

- Also adds **Bluetooth** support
  - Not the focus of this presentation…
  - Example available in ADK source code

AOA 2.0 Protocol

- New Product ID's
  - **0x2D02** – Supports Audio
    - 2 Audio interfaces (control + streaming)
  - **0x2D03** – Supports Audio + ADB
    - 3 interfaces: 2 Audio + 1 Bulk
  - **0x2D04** – Supports AOA 1.0 + Audio
    - 3 interfaces: 2 Audio + 1 Bulk
  - **0x2D05** – Supports AOA 1.0 + Audio + ADB
    - 4 interfaces: 2 Audio + 2 Bulk

Accessory Development Kit

- Hardware for ADK 2011 is based on a Arduino Mega2560
- Hardware for the ADK 2012 is based on an ARM Cortex M3
- ADK 2012 Guide

What part of AOA do you Control?

- ADK SW
  - Developer controls what runs on the ADK HW and what runs on the Android device
- ADK HW?
  - Yes! Just requires a USB Host capable device that can provide power to the Android device
  - This is what makes it easy to port the ADK SW to any ARM-based board

ADK Software

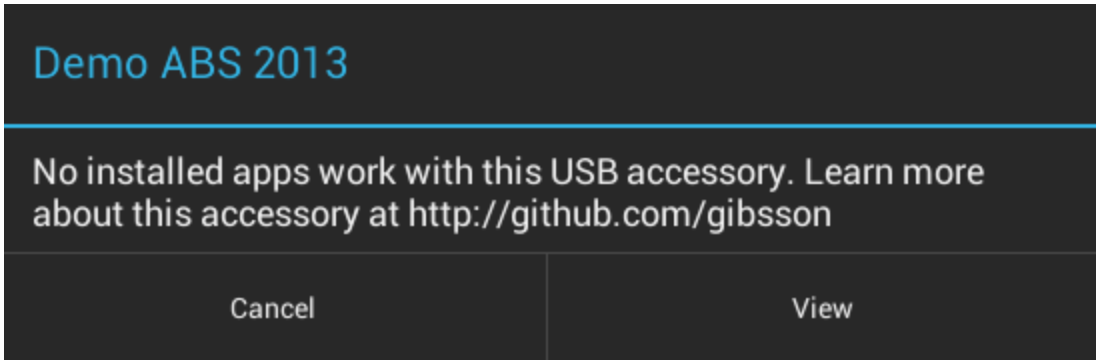
- Google provides sample software for both sides:
  - Source code repo:
    - http://android.googlesource.com/accessories/manifest
  - Android device application can apply on any device with API level > 10
    - Android Application presents UI for control, and communication
  - Accessory code is Arduino-specific
    - Need for a libusb-based Accessory sample code

Software implementation

1. Software on the ADK HW
   - Handles communication with Android device
   - Controls sensors, displays, etc. on the ADK HW
2. Software on the Android device
   - Can be installed automatically via pop-up URL on connection
   - Handles communication with ADK HW
   - Presents a GUI for control, data input/output from ADK

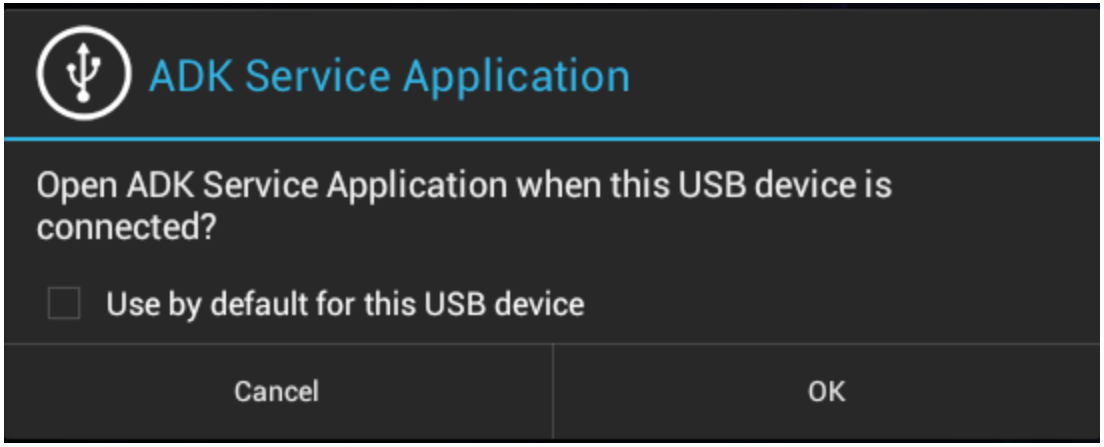
Software implementation

In case the software is not already installed:



Software implementation

Otherwise the user must authorize the execution of the Accessory app:



Application on the Android Device

- Must discover the accessory when it is attached
- Communicate with the accessory via ADK Developer defined commands over USB
- Utilizes the USB API in Android

Application on the Android Device

- Android contains two different packages to support AOA protocol:
  - android.hardware.usb
    - Works with no add-on library for Android 3.1 or higher
  - com.android.future.usb
    - From Google API add-on library for Android 2.3.4 or higher
    - Wrapper around android.hardware.usb
    - Better choice to support the widest range of devices

Discovering the Accessory

- First add an intent-filter to the application's manifest

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
    <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" />
```

(**AndroidManifest.xml**)

Discovering the Accessory

- Then define a resource file that details which USB Accessory this Application communicates with

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <usb-accessory
        manufacturer="Google, Inc."
        model="DemoKit"
        version="2.0" />

</resources>
```

(**accessory_filter.xml**)

Communication with the Accessory

- Grab the UsbAccessory handle from the Intent received
- Open up input/output file streams with the Accessory

```
if(getIntent().getAction().equals("android.hardware.usb.action.USB_ACCESSORY_ATTACHED")){
    UsbAccessory accessory = (UsbAccessory) getIntent().getParcelableExtra(UsbManager.EXTRA_ACCESSORY);
    UsbManager manager = ((UsbManager) getSystemService(Context.USB_SERVICE));
    FileDescriptor fd = manager.openAccessory(accessory).getFileDescriptor();
    mOutputStream = new FileOutputStream(fd);
    mInputStream = new FileInputStream(fd);
}
```

(**myactivity.java**)

Demonstrations

- Android Device:
  - Stock Nexus 7 Tablet running Android Jellybean (v4.2.1)
- Custom ARM-based accessory
  - Raspberry Pi model B
  - Running Linux

Demonstrations

- Why were these particular platforms chosen?
  - Raspberry Pi is a very cheap & popular board nowadays
  - Wanted to show ease of using libusb-based code
  - Nexus 7 is a recent stock device that needs no modification
    - Any Jellybean device would work the same

Demonstration #1

- Create an accessory device that:
  - Displays pictures
  - Allows control of the slideshow from Android
- Shows AOA v1.0 capabilities
  - Application on both sides

Demonstration #2

- Create an accessory device that:
  - Plays audio
  - Adds 1 Human Interface Device
- Shows AOA v2.0 capabilities
  - No application required on Android device

Conclusion

- Interesting standard
  - Great flexibility
  - Compatibility v2.0 / v1.0
- Limitations
  - Audio format
- Full code sources soon available at:

  http://github.com/gibsson

Questions?

References

- Based on Jesse Dannenbring's presentation:
  - https://speakerdeck.com/jdannenbring/ad-2002slides-dannenbring
  - https://github.com/jdannenbring/android-arm-accessory
  - https://github.com/jdannenbring/android-arm-accessory-app

- "Android Developers". http://developer.android.com/index.html. 1 February 2013.

References

- Di Cerbo, Manuel. “Turn your Linux computer into a huge Android USB Accessory” Using Android in Industrial Automation. 5-13-2011. http://android.serverbox.ch/?p=262

- “libusb”. http://www.libusb.org/. 1 February 2013.

- “monaka / libusb-android / overview”. https://bitbucket.org/monaka/libusb-android/. 1 February 2013.


deneo
Embedded