
Google™
11
I/O

Introducing Android Open Accessories and ADK

Jeff Brown, Erik Gilling, Mike Lockwood
May 10, 2011

Session Feedback

- http://goo.gl/leddF
- Twitter: #io2011 #Android

USB on Android Yesterday

- Android USB device built-in functions
  - USB mass storage
  - adb
  - USB tethering
- Limitations
  - Most Android devices have no USB host support
  - USB Host in Android 3.0 is very limited
  - No USB APIs

Input Devices

USB Basics

USB is an asymmetric protocol

- Host
  - Controls the entire bus
  - Keeps track of all attached devices and hubs (enumeration)
  - Initiates communication with the devices
  - Is a power source
- Devices
  - Communicates only with the host through endpoints
  - Describes its capabilities to the host during enumeration
  - Implements standard or vendor specific functions
  - Can draw power from the host

USB Descriptors

Device describes its capabilities to host during enumeration via USB descriptors

- Device Descriptor
- Configuration Descriptor
- Interface Descriptor
- Endpoint Descriptor
- (and others)

Device Descriptor

Provides a top level description of the device

- Vendor ID (assigned by USB.org)
- Product ID (assigned by vendor)
- Device class, subclass and protocol IDs
- Manufacturer, Product and Serial Number strings

Configuration Descriptor

- Device may present multiple configurations
- Android devices have only one
  - Configuration number
  - List of interfaces
  - Max power usage

Interface Descriptor

- An interface represents a specific function implemented by the device
  - Interface number
  - Class, Subclass and Protocol IDs
  - List of endpoints
- Android examples:
  - USB mass storage
  - adb
  - RNDIS (USB ethernet)
  - MTP

Endpoint Descriptor

Endpoints are the channels for sending and receiving data

- Address
- Type (control, bulk, interrupt or isochronous)
- Direction (OUT: host to device, IN: device to host)
- Max packet size

Endpoint types

- Control
  - Endpoint zero used for enumeration
  - Vendor- and class-specific requests
  - Host-initiated, bi-directional
- Bulk: For general-purpose I/O; uni-directional
- Interrupt: For small asynchronous messages/events
- Isochronous: For time critical, low latency messages

USB Host API

USB Host API

- New in Android 3.1
- Only supported on hardware with USB host
- Can support existing USB peripherals as well as devices designed for Android

USB Host classes

Used to describe the capabilities of a device

- UsbDevice
  - Vendor and product ID
  - Device class, subclass and protocol
  - List of interfaces
- UsbInterface
  - Interface class, subclass and protocol
  - List of endpoints
- UsbEndpoint
  - Type (bulk, interrupt)
  - Direction
  - Max packet size

USB Host classes (continued)

Used for communicating with the device

- UsbDeviceConnection
  - Encapsulates an open connection to the device
  - Claim and release interfaces
  - Initiate transfers, wait for results
- UsbRequest
  - Encapsulates data to be sent to or received from a device

Example: Finding device endpoints

```
private void openDevice(UsbDevice device) {
  int vid = device.getVendorId();
  int pid = device.getProductId();
  if (vid == 0x22B8 && pid == 0x70A8 && device.getInterfaceCount() > 0) {
    UsbInterface intf = device.getInterface(0);
    if (intf.getEndpointCount() == 2) {
      UsbEndpoint ep1 = intf.getEndpoint(0);
      UsbEndpoint ep2 = intf.getEndpoint(1);
      if (ep1.getDirection() == UsbConstants.USB_DIR_IN) {
        epIn = ep1;
      } else {
        epOut = ep1;
      }
      if (ep2.getDirection() == UsbConstants.USB_DIR_IN) {
        epIn = ep2;
      } else {
        epOut = ep2;
      }
      if (epIn == null || epOut == null) {
        Log.e(TAG, "endpoints in both directions not found");
        return;
      }
```

Example: Communicating With a Device

```
UsbDeviceConnection connection = mUsbManager.openDevice(device);
if (connection != null && connection.claimInterface(intf, false)) {
  // we are connected
}

// send a control request
int count = connection.controlTransfer(
            UsbConstants.USB_TYPE_VENDOR | UsbConstants.USB_DIR_OUT,
            request, value, index, message, message.length, timeout);

// bulk transfer
int count = mConnection.bulkTransfer(epIn, buffer, buffer.length, timeout);

// queue asynchronous request
UsbRequest request = new UsbRequest();
request.initialize(connection, epOut);
request.queue(buffer, bufferLength);

// wait for result
UsbRequest request = connection.requestWait();
```

Demo

Android Open Accessories

USB for the rest of the the robots

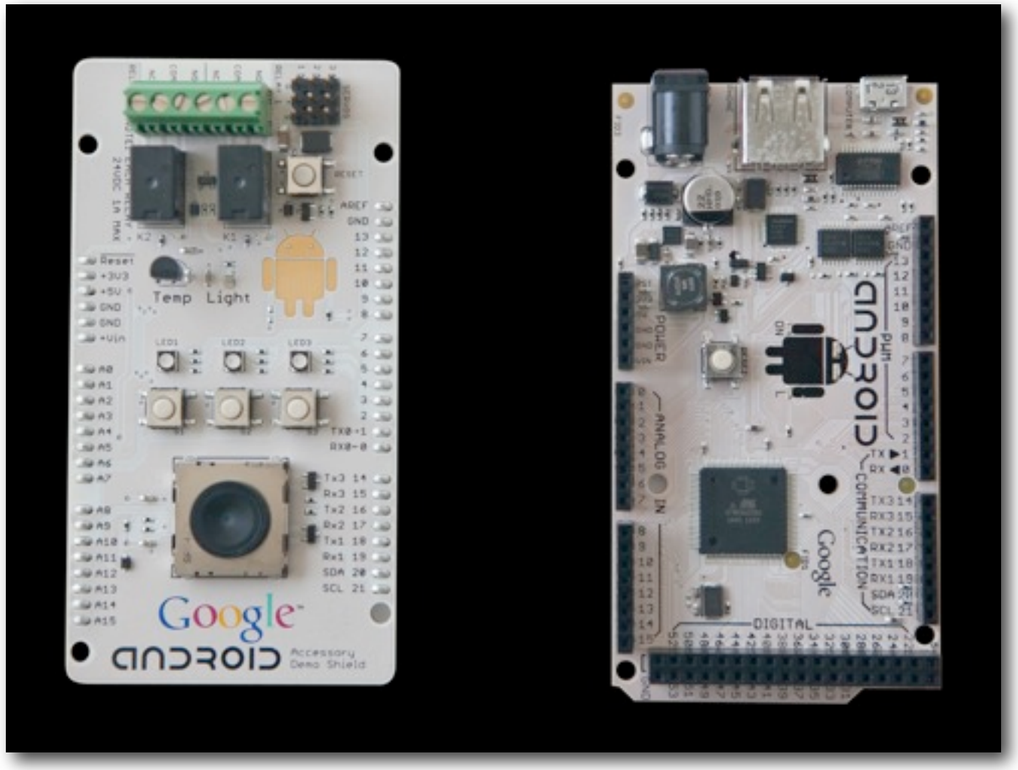
- Most Android devices do not support USB host mode
- Every compliant Android device supports USB device mode
- Accessory plays the role of the host

What is an Open Accessory?

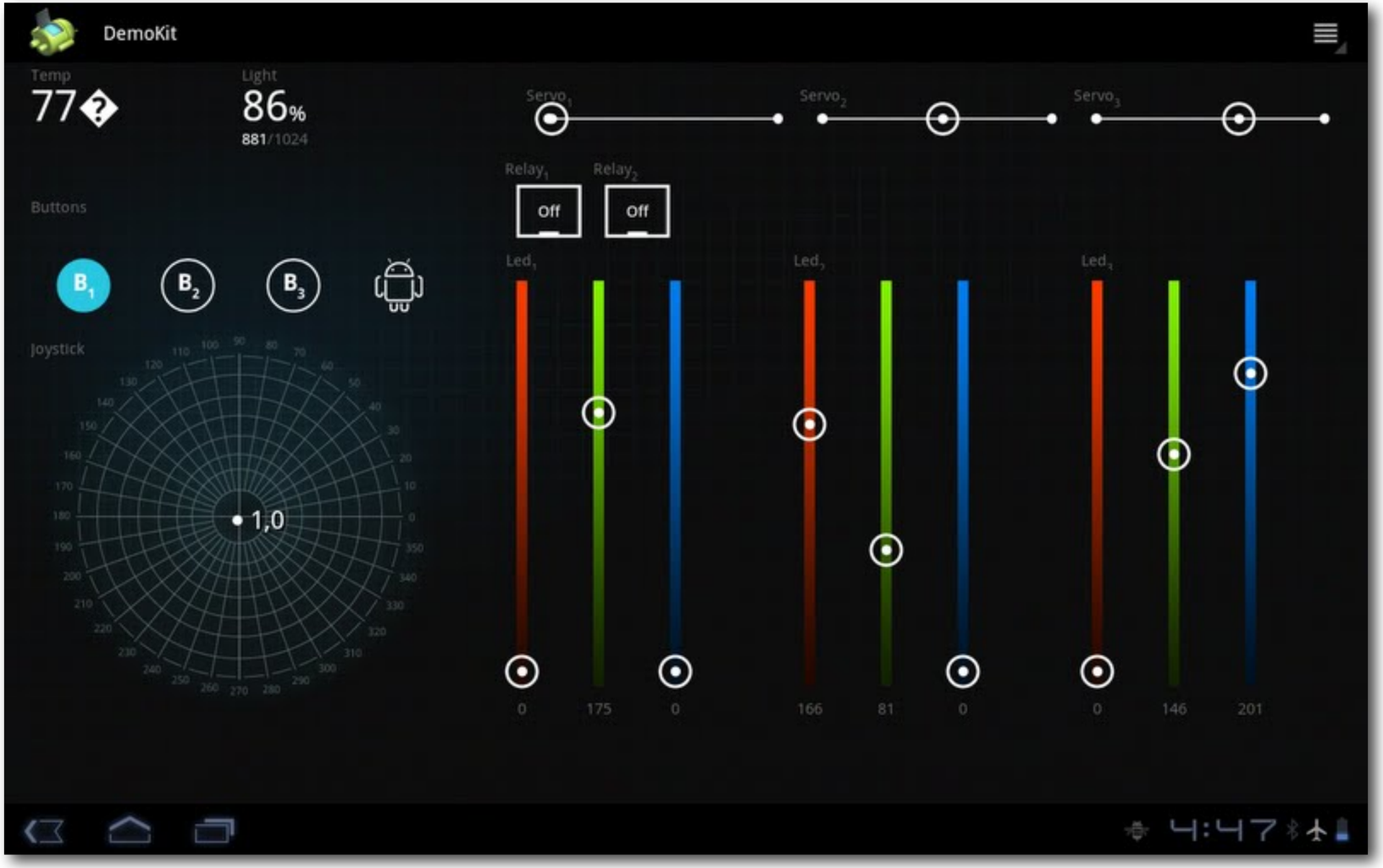
- Simple USB protocol for communication between peripherals and Android devices
- Accessory is the host, Android is the device
- Bi-directional communication over two bulk endpoints
- Protocol for associating Android applications with the hardware they support

ADK

- Android Accessory Board:
  - Based on the Arduino Mega 2560
  - Maxim MAX3421E host controller
  - Works with Arduino tool chain (http://arduino.cc)
- Google Shield
  - 3 RGB LEDs
  - 4 buttons (3 mechanical, 1 capacitive)
  - 3 servo channels, 2 relays
  - joystick
  - light & temperature sensors



Reference Android ADK App



Requirements for Open Accessory Hardware

- USB host
- Must supply 500mA @5V charging power

Open Accessory Handshake

When a new device is connected, the accessory will perform these steps:

- Send “Get Protocol” command to get Accessory protocol version. If this fails, the device does not support accessories
- Send manufacturer, model, description, version, serial number, and URI strings to identify the accessory to the Android Device
- Send “Start” command
- The Android device should re-enumerate in accessory mode and launch an app

USB Accessory Handshake, part 2

- You're in Accessory Mode if:
  - Vendor ID is 0x18D1 (Google)
  - Product ID is 0x2D00 or 0x2D01
- Read configuration descriptors
- Look for first bulk IN and first bulk OUT endpoints
- Set configuration to 1
- Endpoints are now ready for communication

Open Accessory APIs

- New USB APIs in Android 3.1
  - com.android.hardware.usb.*
  - Use this if your app will require Android 3.1 (API 12) or later
  - Supported on Motorola Xoom with Android 3.1 update
- Compatibility Library for Gingerbread
  - com.android.future.usb.*
  - Link against com.android.future.usb.accessory.jar
  - Very similar to Android 3.1 API
  - Use this if you want to support Gingerbread and later
  - Supported on Nexus One and Nexus S with 2.3.4 update

Connecting to an Accessory

- Application describes compatible accessories in manifest meta-data
- USB Manager matches accessory to compatible application(s)
- Asks user if it is OK to use your app with the accessory or to choose among multiple applications
- Application's Activity is started with USB_ACCESSORY_ATTACHED Intent
- Association made permanent if the user selects "always use this app for this accessory" in the dialog
- USB_ACCESSORY_DETACHED Intent sent when accessory disconnected

Example: AndroidManifest.xml

```
<manifest ...>
<application android:label="Accessory Sample">
<uses-library android:name="com.android.future.usb.accessory" />

<activity android:name="UsbReceiver">
<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>
```

Example: accessory_filter.xml

```
<resources>
<usb-accessory manufacturer="Acme Corporation"
               model="USB Anvil"
               version="1.0"
               />
</resources>
```

UsbAccessory class

- Describes the USB accessory based on the strings it provides in the handshake
  - Manufacturer Name
  - Model Name
  - Description (user-visible string)
  - Version
  - URL (web page to visit if no installed apps support the accessory)
  - Serial Number (optional)
- Manufacturer, Model and Version are used for associating accessories with applications

USB Manager class

- **`getAccessoryList()`** returns currently attached accessories (currently there can only be one)
- **`openAccessory()`** returns a ParcelFileDescriptor
- **`hasPermission()`** to see if you have access to the accessory
- **`requestPermission()`** to request permission from user

Example: Opening an Accessory for IO

```
// Get the accessory from the USB_ACCESSORY_ATTACHED Intent
Intent intent = getIntent();
UsbAccessory accessory = mUsbManager.getAccessory(intent);

// Open the accessory
ParcelFileDescriptor pfd = mUsbManager.openAccessory(accessory);
if (pfd != null) {
    FileDescriptor fd = pfd.getFileDescriptor();
    InputStream input = new FileInputStream(fd);
    OutputStream output = new FileOutputStream(fd);
    // now read and write data to the accessory
}
```

Q&A

Session feedback: http://goo.gl/leddF
Twitter: #io2011 #Android


Google™
11
I/O