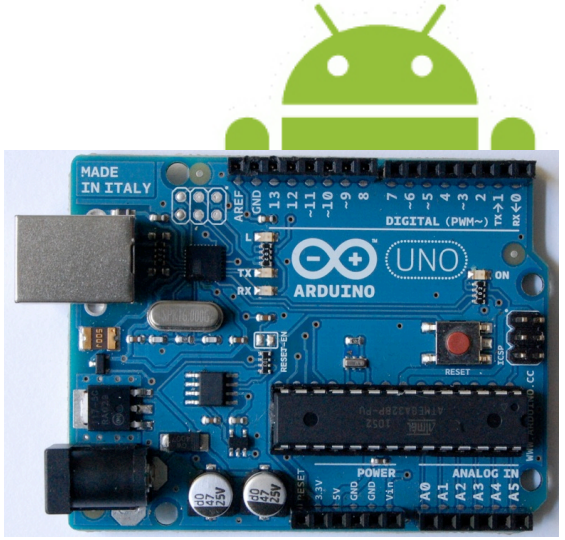
Building Android Accesories

... using the Open Accessory Development Kit and Arduino

Simon Monk.

Agenda

- Introduction
- Demonstrations
- Setting up
- A Simple Example
  - The Arduino library
  - Android
- Resources



Introduction

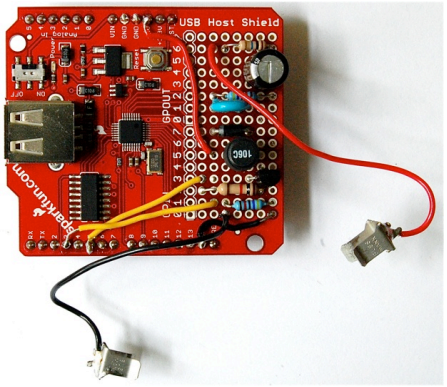
- Arduino
  - USB-enabled prototyping board
  - Simple, low power 8 bit microcontroller
  - Electronics enthusiasts and artists
  - IDE - Windows, Mac, Linux
  - Open source hardware

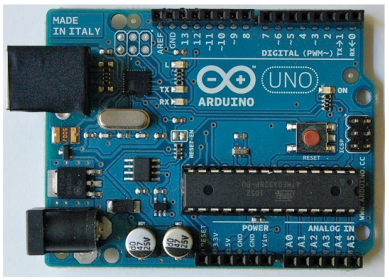
- Open Accessory
  - Google Standard and APIs for USB communication to Accessories for Android phones
  - Uses the Arduino firmware (bootloader) and the Arduino IDE

- ADK
  - Google reference design hardware, similar to Arduino board

Geiger Counter Accessory

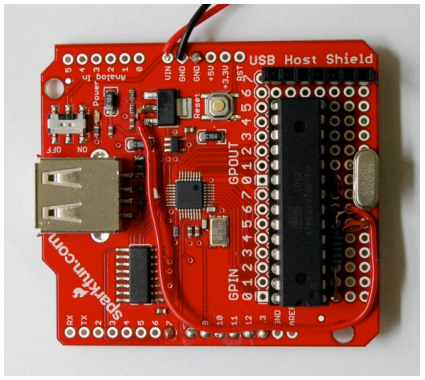
- Arduino Uno
- Sparkfun USB Host Shield
- Prototyping area

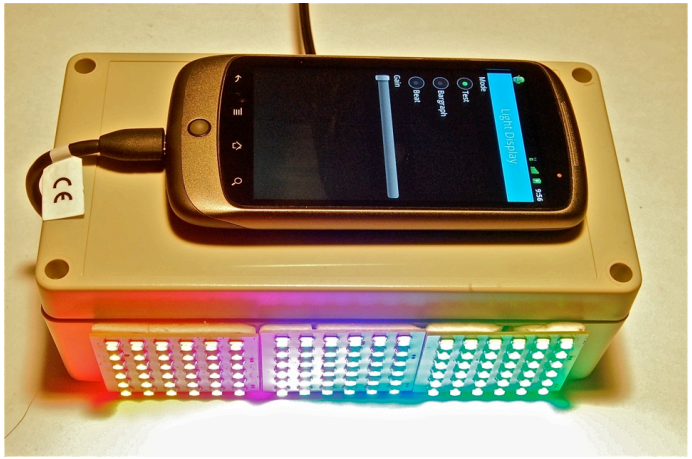




Light Show Charger

- Duinodroid Base
- Off board Arduino in prototying area of USB host shield



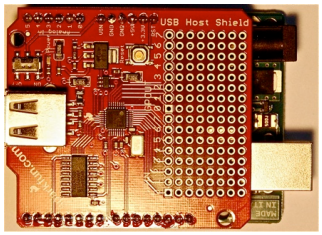


Setting Up - Arduino

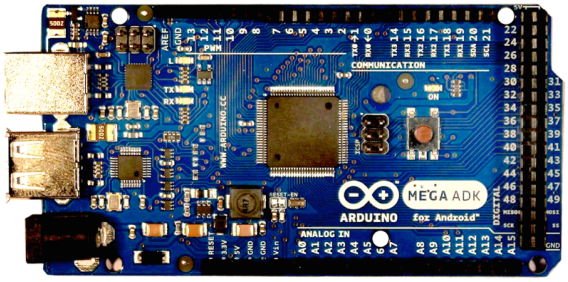
- Arduino Libraries
  - Copy into arduino/libraries
  - From **microbridge.googlecode.com/files/usb_host_patched.zip**
    - Modified for USB Host Shield and Arduino Uno
    - USB_Host_Shield
    - #include <Max3421e.h>
    - #include <Usb.h>
  - From **developer.android.com/guide/topics/usb/adk.html**
    - #include <AndroidAccessory.h>
    - Do NOT install USB_Host_Shield from here
  - Example Arduino Sketch - **www.duinodroid.com**
    - apA_open_accessory_test.zip

Arduino Options

- Arduino Uno + USB Host Shield

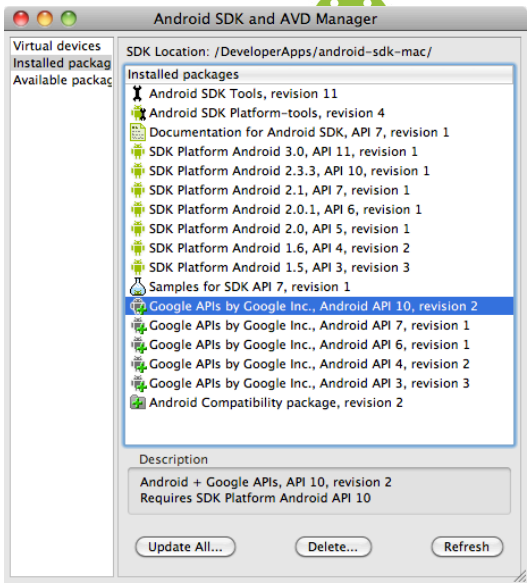


- Arduino Mega ADK
  - Arduino Mega with USB Host



Setting Up - Android

- Android
  - Android Version
    - Android 2.3.4+ (but not all)
    - Nexus One
    - Nexus S
    - Some HTC models ?
  - ADK Eclipse Project
    - **developer.android.com/guide/topics/usb/adk.html**
    - Google APIs level 10 (Android 2.3.3)
    - This example project - **www.duinodroid.com**
      - OpenAccessoryTest.zip - source project
      - OpenAccessoryTest.apk - binary



Simple Example

- Increment
  - Enter a number in a field on the phone and click ‘send’
  - The Arduino Increments it and sends it back
- Trace
  - Log area displays the execution path through the App
- My attempt to get a handle on a complex process
- A template for you to use.

The Arduino Code

- Arduino has its own IDE
- C / C++
- Wiring library
- Connect Arduino by USB and upload a ‘Sketch’ to the board
- Compiles and sends executable code to Arduino board’s Flash memory

The Arduino Code

```
#include <Max3421e.h>
#include <Usb.h>
#include <AndroidAccessory.h>

AndroidAccessory acc("Simon Monk",
                     "OpenAccessoryTest",
                     "DemoKit Arduino Board",
                     "1.0",
                     "http://www.duinodroid.com",
                     "0000000012345678");
void setup()
{
  acc.powerOn();
}
```

The Arduino Code

```
void loop()
{
  byte msg[1];
  if (acc.isConnected())
  {
    int len = acc.read(msg, sizeof(msg), 1);
    if (len >= 1)
    {
      byte value = msg[0];
      sendMessage(value + 1);
    }
  }
}
```

nakLimit

The Arduino Code

```
void sendMessage(int value)
{
  if (acc.isConnected())
  {
    byte msg[2];
    msg[0] = value >> 8;
    msg[1] = value & 0xff;
    acc.write(msg, 2);
  }
}
```

Autostart and Download

- Arduino

```
AndroidAccessory acc("Simon Monk",
        "OpenAccessoryTest",
        "DemoKit Arduino Board",
        "1.0",
        "http://www.duinodroid.com",
        "0000000012345678");
```

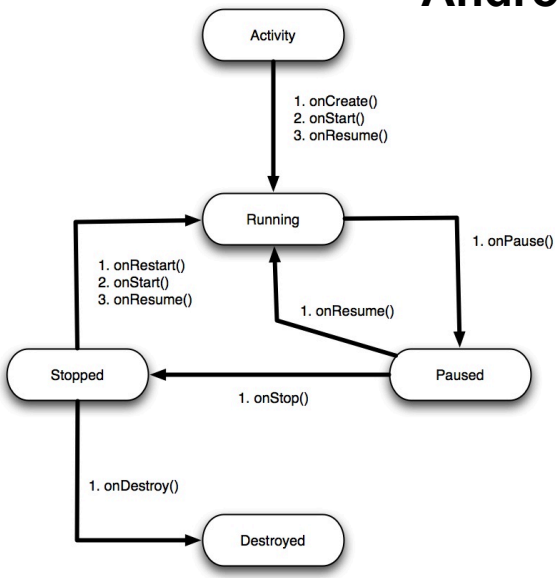
- Android

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

<meta-data android:name="android.hardware.usb.action.USB_ACCESSORY_ATTACHED"
  android:resource="@xml/accessory_filter"/>
```

- xml/accessory_filter.xml

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

<resources>
    <usb-accessory manufacturer="Simon Monk"
                   model="OpenAccessoryTest" version="1.0" />
</resources>
```

Android Lifecycle



Opening the Accessory

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    mByteField = (EditText) findViewById(R.id.messagebyte);
    mResponseField = (EditText) findViewById(R.id.arduinoresponse);
    mSendButton = (Button) findViewById(R.id.sendButton);
    mSendButton.setOnClickListener(new OnClickListener() {
      public void onClick(View v) {
        sendMessageToArduino();
      }
    });
    setupAccessory();
  }
```

- Create the controls
- Setup an onClick listener for the button
- Call setupAccessory

setupAccessory()

```
private void setupAccessory() {
    log("In setupAccessory");
    mUsbManager = UsbManager.getInstance(this);
    mPermissionIntent = PendingIntent.getBroadcast(this, 0, new Intent(ACTION_USB), 0);
    IntentFilter filter = new IntentFilter(ACTION_USB);
    filter.addAction(UsbManager.ACTION_USB_ACCESSORY_DETACHED);
    registerReceiver(mUsbReceiver, filter);
    if (getLastNonConfigurationInstance() != null) {
      mAccessory = (UsbAccessory) getLastNonConfigurationInstance();
      openAccessory(mAccessory);
    }
  }
```

- Link Broadcast receiver
- Open accessory from stored configuration instance
- Or don’t

openAccessory()

```
private void openAccessory(UsbAccessory accessory) {
    mFileDescriptor = mUsbManager.openAccessory(accessory);
    if (mFileDescriptor != null) {
      mAccessory = accessory;
      FileDescriptor fd = mFileDescriptor.getFileDescriptor();
      mInputStream = new FileInputStream(fd);
      mOutputStream = new FileOutputStream(fd);
      Thread thread = new Thread(null, this, "OpenAccessoryTest");
      thread.start();
      alert("openAccessory: Accessory opened");
    } else {
      log("openAccessory: accessory open failed");
    }
  }
```

- Create input and output streams
- Start a thread listening for incoming messages

onResume()

```
public void onResume() {
    log("Resuming");
    super.onResume();

    if (mInputStream != null && mOutputStream != null) {
      log("Resuming: streams were not null");
    } else {
      log("Resuming: streams were null");
      establishPermissionsAndOpenAccessory();
    }
  }
```

- If we still have streams, do nothing
- otherwise, establish permissions and open the accessory

establishPermissionsAndOpenAccessory()

```
private void establishPermissionsAndOpenAccessory() {
    UsbAccessory[] accessories = mUsbManager.getAccessoryList();
    UsbAccessory accessory = (accessories == null ? null : accessories[0]);
    if (accessory != null) {
      if (mUsbManager.hasPermission(accessory)) {
        openAccessory(accessory);
      } else {
       synchronized (mUsbReceiver) {
          if (!mPermissionRequestPending) {
            mUsbManager.requestPermission(accessory, mPermissionIntent);
            mPermissionRequestPending = true;
          }
        }
      }
    } else {
      log("establishPermissionsAndOpenAccessory:mAccessory is null");
    }
  }
```

- If we have an accessory and permissions, open the streams
- Otherwise request permission to use USB

Broadcast Receiver

```
private final BroadcastReceiver mUsbReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
      String action = intent.getAction();
      if (UsbManager.ACTION_USB_ACCESSORY_DETACHED.equals(action)) {
        UsbAccessory accessory = UsbManager.getAccessory(intent);
        if (accessory != null && accessory.equals(mAccessory)) {
          log("Detached");
          closeAccessory();
        }
      }
    }
  };
```

- Recieves system messages such as:

USB_ACCESORY_DETACHED

closeAccessory()

```
private void closeAccessory() {
    log("In closeAccessory");
    try {
      if (mFileDescriptor != null) {
        mFileDescriptor.close();
      }
    } catch (IOException e) {
    } finally {
      mFileDescriptor = null;
      mAccessory = null;
      mInputStream = null;
      mOutputStream = null;
    }
}
```

- Close and null everything
- When we reconnect we will start agan

Sending Data

- sendMessageToArduino
  - read a byte value from the text field
  - call sendCommand(value)
    - construct a byte array
    - write it on the output stream

sendCommand()

```
public void sendCommand(byte value) {
    byte[] buffer = new byte[1];
    buffer[0] = (byte) value;
    if (mOutputStream != null) {
      try {
        mOutputStream.write(buffer);
      } catch (IOException e) {
        log("Send failed: " + e.getMessage());
      }
    } else {
      log("Send failed: mOutStream was null");
    }
  }
```

- More than we need for this example (we could just send the byte)
- Generally, pack all the data to send into a byte array

Back on the Arduino

```
void loop()
{
  byte msg[1];
  if (acc.isConnected())
  {
    int len = acc.read(msg, sizeof(msg), 1);
    if (len >= 1)
    {
      byte value = msg[0];
      sendMessage(value + 1);
    }
  }
}
```

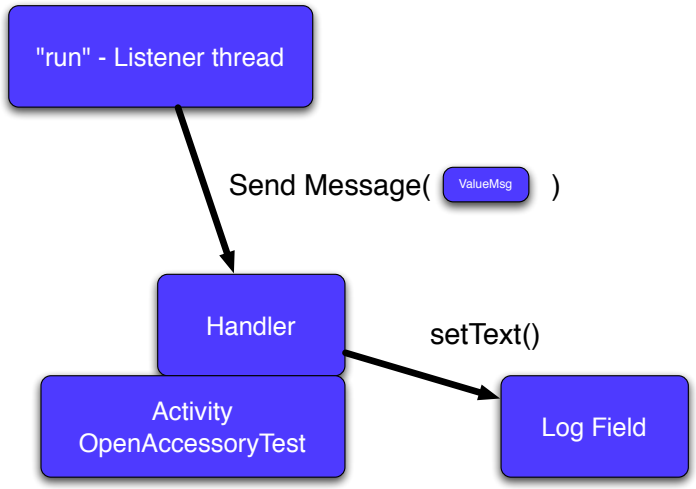
- read the byte
- add 1 to it
- send it back

Back on the Arduino

```
void sendMessage(int value)
{
  if (acc.isConnected())
  {
    byte msg[2];
    msg[0] = value >> 8;
    msg[1] = value & 0xff;
    acc.write(msg, 2);
  }
}
```

- if not connected, then connect
- pack the int into a byte array
- send the byte array to Android

Receiving Data (Android)



Handler

```
Handler mHandler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
      ValueMsg t = (ValueMsg) msg.obj;
      log("Arduino sent: " + t.getFlag() + " " + t.getReading());
    }
  };
```

- Direct interaction with Activity thread is not allowed
- A ‘Handler’ is allowed to act on the Activity
- The handler passes a message object

ValueMsg

```
public class ValueMsg {
	private char flag;
	private int reading;

	public ValueMsg(char flag, int reading) {
		this.flag = flag;
		this.reading = reading;
	}

	public int getReading() {
		return reading;
	}

	public char getFlag() {
		return flag;
	}
}
```

- For more complex messages from the Arduino then add more properties.

Receiving data

```
public void run() {
    int ret = 0;
    byte[] buffer = new byte[16384];
    int i;
    while (true) {
      try {
        ret = mInputStream.read(buffer);
      } catch (IOException e) {
        break;
      }
      i = 0;
      while (i < ret) {
        int len = ret - i;
        if (len >= 2) {
          Message m = Message.obtain(mHandler);
          int value = composeInt(buffer[i],
                                 buffer[i + 1]);
          m.obj = new ValueMsg('a', value);
          mHandler.sendMessage(m);
        }
        i += 2;
      }
    }
}
```

- read the message
- construct a Message
- send the Message to the Handler for the Activity

Open Accessory Alternatives (USB)

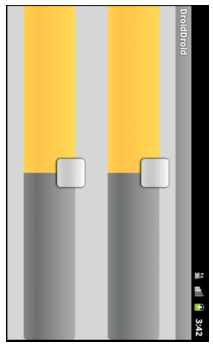
- ADK - Googles reference hardware - given away at another Android conference!

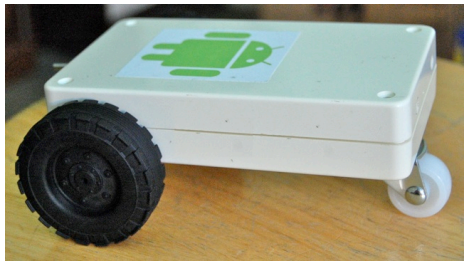
- Microbridge
  - ADB to standard Arduino over USB - using USB Host board.
  - http://romfont.com/2011/05/15/microbridge-adb-support-for-arduino/

- IOIO (yoyo)
  - With or without Open Accessory USB, non standard Arduino board

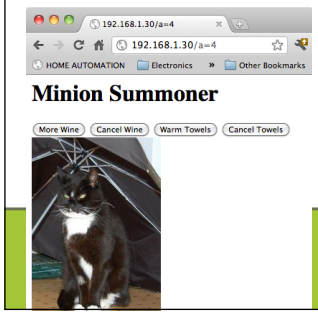
Open Accessory Alternatives (Non-USB)

- Amarino
  - Arduino and Android Bluetooth project
  - http://www.amarino-toolkit.net/



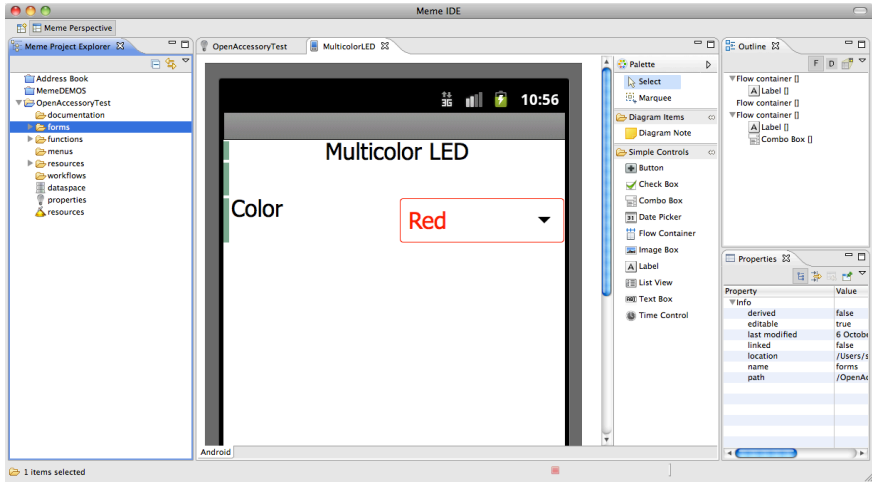


- Ethernet / WiFi Shield
  - Web Interface



Android SDK Alternatives

- Native Interfaces from Cross-platform tools
  - Phone Gap (next door)
  - Meme IDE
- Meme IDE Presentation
  - NOT part of Android Open
  - Today, this hotel, Board Room C
  - 6pm - 7pm



Resources

- Books
  - Open Accessory and Arduino
    - Arduino + Android Projects for the Evil Genius. Simon Monk. (Dec 2011)
  - Android Programming
    - Hello Android. Ed Burnette

- Web Resources
  - Official Google Page
    - http://developer.android.com/guide/topics/usb/adk.html
  - Useful tips on using standard Arduino kit
    - http://letsmakerobots.com/node/26839

- My Blog: http://srmonk.blogspot.com/