Simplifying IoT with Firebase

Mike McDonald

Engineer @ Google/Firebase

@asciimike, @firebase

IoT development requires system level thought

Hardware
Interacting with the physical world

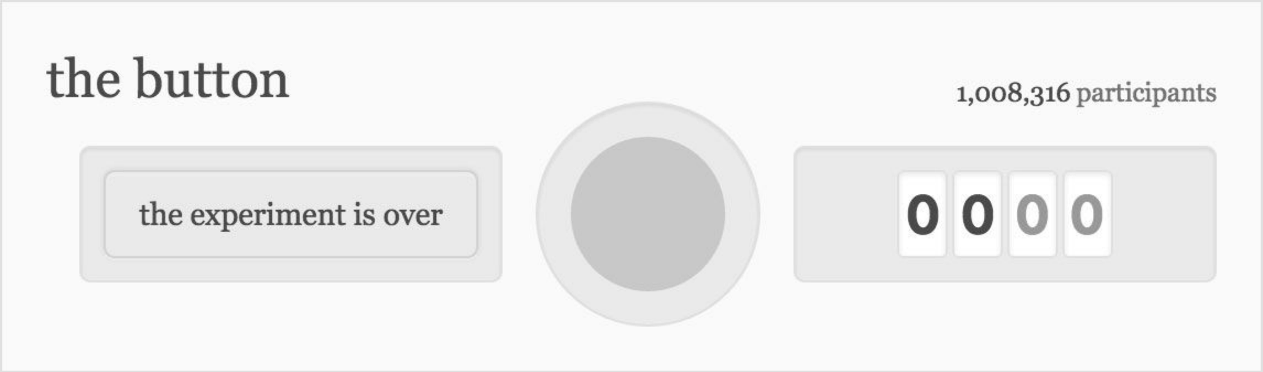
Mobile
Building compelling user experiences

Cloud
Bridging the gap between devices

It's like trying to build a stool

How can we make hardware development as fast and easy as software development?

Let’s give ourselves something to work towards...

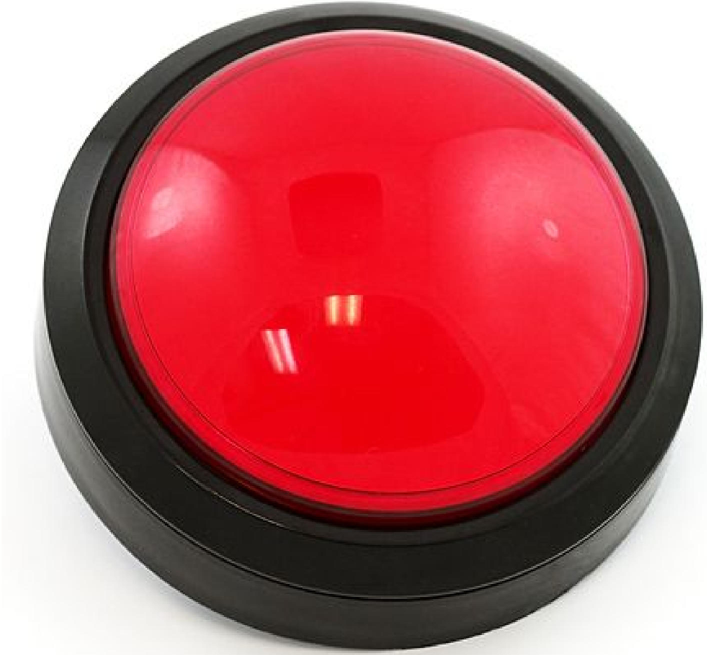


the button

1,008,316 participants

the experiment is over

Our button

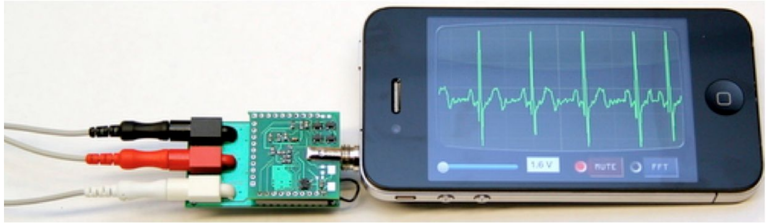


Hardware

Talking to hardware: USB Host/Accessory

- Allows developers to create accessories to plug in via USB: game controllers, robots, etc.
- Started in 2011 w/ Android Open Accessory and the Arduino ADK Board
- Now, any* Android 3.1+ device can use USB devices w/ a USB OTG cable (*host mode*)
- USB Serial for Android implements a nice driver for common serial devices

Talking to hardware: Headphone Jack



- Outputs audio signal through the headphone jack to a microcontroller, which can then interact with sensors
- Easily ported cross platform
- HiJack: project from U of M
- Thermodo: commercial temp sensor

Talking to hardware: Bluetooth

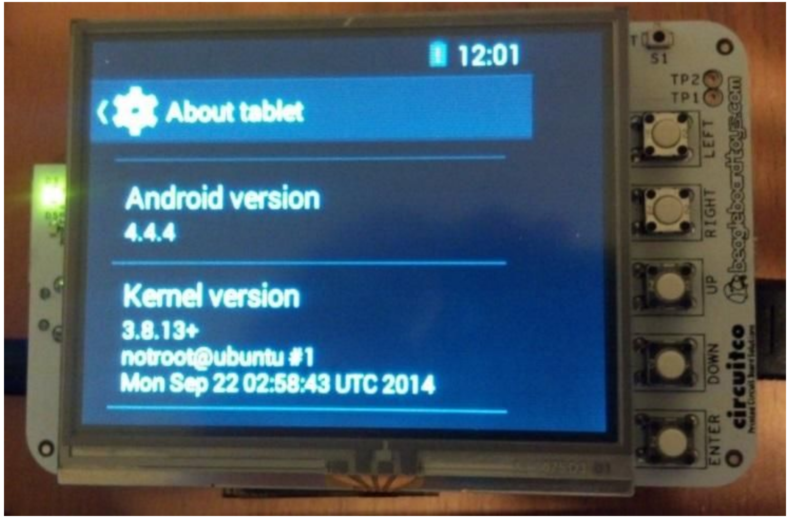
- android.bluetooth is your new best friend
- Great for low power wireless, audio transmission
- There are *lots* of different bluetooth protocols: A2DP, HID, etc. which can be somewhat confusing
- Beacons are a great offshoot of this technology (iBeacon, Eddystone)

Talking to hardware: Internet (WiFi, ethernet)

- Easy for mobile: **it's just another API**
- Robust, high throughput, secure*
- But, if your Internet goes out...

*with HTTPS...

Talking to hardware: Running Android on Hardware



12:01

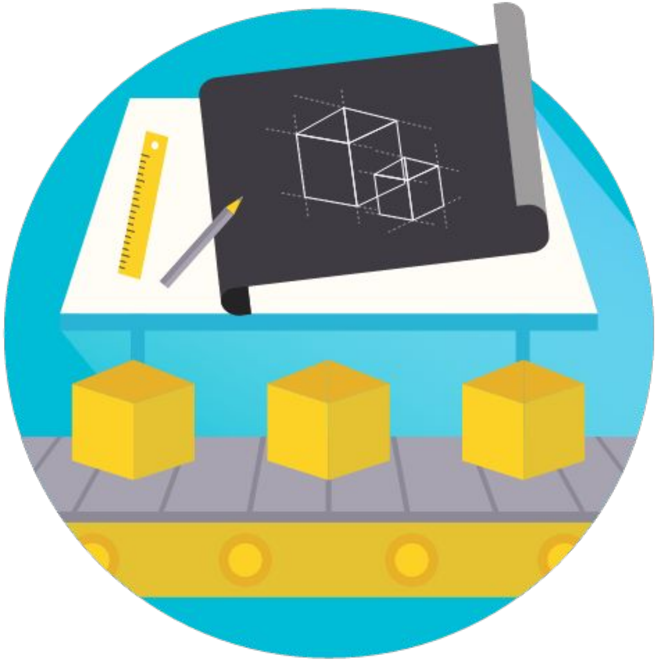
About tablet

Android version
4.4.4

Kernel version
3.8.13+
notroot@ubuntu #1
Mon Sep 22 02:58:43 UTC 2014

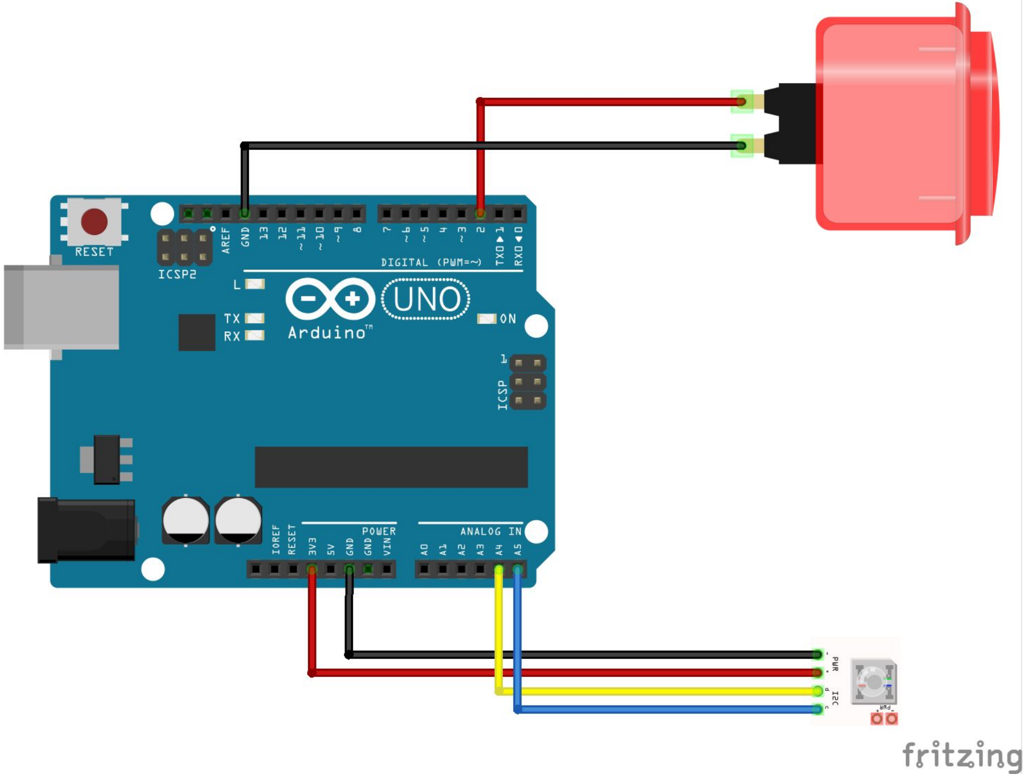
- Android is just Linux, so of course you can run it on an RPi, Beaglebone, etc.
- Most versions use 4.x, no push to go higher
- Compile using the NDK, calls through the JNI if you want to build an app
- This solution solves distribution, OTA updates, etc.

Talking to hardware: Brillo!

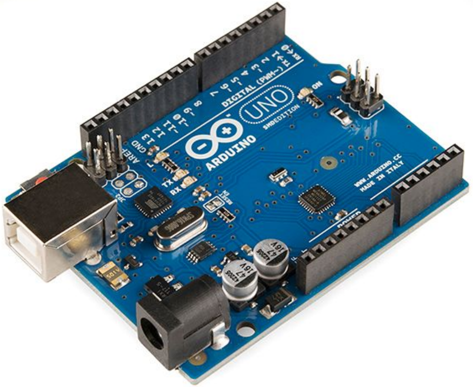


- An OS (based on AOSP)
- Access to developer tools like metrics, OTA updates, Weave
- Device to device and device to cloud connectivity
- Development toolchain with access to common tools like adb

Hardware schematic



Our version of “the button”



Mobile

```
// Step 1: Add an Intent Filter in Manifest.xml
<intent-filter>
   <action android:name=”android.hardware.usb.action.USB_DEVICE_ATTACHED” />
</intent-filter>

<meta-data android:name=”android.hardware.usb.action.USB_DEVICE_ATTACHED”
   android:resource=”@xml/device_filter” />
```

```
// Step 2: Create your device_filter.xml
<?xml version=”1.0” encoding=”utf-8”?>
<resources>
   <!-- Arduino Uno R3; Product ID: 0x0043 Vendor ID: 0x2a03 -->
   <usb-device vendor-id=”10755” product-id=”67” />
</resources>
```

```java
// Step 3: Handle the intent in MainActivity
Intent intent = getIntent();
String action = intent.getAction();

UsbDevice device = (UsbDevice)intent.getParcelableExtra(UsbManager.EXTRA_DEVICE);
if (UsbManager.ACTION_USB_DEVICE_ATTACHED.equals(action)) {
    setDevice(device);
} else if (UsbManager.ACTION_USB_DEVICE_DETACHED.equals(action)) {
    if (mDevice != null && mDevice.equals(device)) {
        setDevice(null);
    }
}
```

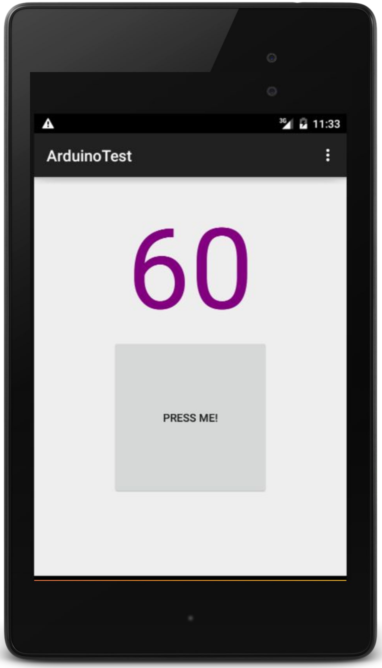
```
// Step 4: Set up our USB Device Properly
private void setDevice(UsbDevice device) {
 // Step 1: Open the interface
 // Step 2: Get the correct endpoint
 // Step 3: Set the device (assuming everything is correct above)
 // Step 4: Open the connection
 // Step 5: Claim the interface
 // Step 6: Set up USB to Serial converter on the Arduino
}
```

```
  // Step 4 (code): Set up our USB Device Properly
  private void setDevice(UsbDevice device) {
   UsbInterface intf = device.getInterface(1); // Yes, this number matters
   mEndpointOut = intf.getEndpoint(0); // As does this one!
   mDevice = device;
   mConnection connection = mUsbManager.openDevice(device);
   connection.claimInterface(intf, true) {
   new Thread(new Runnable() {
     public void run() {
       mConnection.controlTransfer(/* Set up USB correctly */);
     }
   }).start();
}
```

```java
// Step 5: Send Commands!
private void sendCommand(int r, int g, int b) {
   synchronized (this) {
       if (mConnection != null) {
           String colors = "Color:" + r + ";" + g + ";" + b + "\n";
           final byte[] message = colors.getBytes();
           new Thread(new Runnable() {
               public void run() {
                   mConnection.bulkTransfer(mEndpointOut, message, message.length, 0);
               }
           }).start();
       }
   }
}
```

```
// Step 6: Receive Commands
public void run() {
  ByteBuffer buffer = ByteBuffer.allocate(/* Size of buffer*/);
  UsbRequest request = new UsbRequest();
  request.initialize(mConnection, mEndpointIntr);
  while (true) {
    request.queue(buffer, /* Size of buffer */);
    if (mConnection.requestWait() == request) {
      // Examine the buffer
      try { Thread.sleep(100); } catch (InterruptedException e) {}
    } else {
      break;
    }
  }
}
```

Let’s see the app!



Cloud

Firebase glues devices together

```
// Firebase is JSON
"your-firebase": {
  "devices": {
    "color": {
      "b": 255,
      "g": 255,
      "r": 255
    },
    "button": {
      "pressed": true | false
    }
  },
  "timer": 60
}
```

```java
// Access your data by going directly to that path
mRef = new Firebase("https://iot-test.firebaseio-demo.com");

mRef.child("devices/color").addValueEventListener(new ValueEventListener() {
   @Override
   public void onDataChange(DataSnapshot dataSnapshot) {
       LEDColor c = dataSnapshot.getValue(LEDColor.class);
       sendCommand("Color:" + c.getR() + ";" + c.getG() + ";" + c.getB() + "\n");
   }

   @Override
   public void onCancelled(FirebaseError firebaseError) {
   }
});
```

```java
// Set data by writing directly to a path
mRef = new Firebase("https://iot-test.firebaseio-demo.com");

final int c = getColorForCount(mCount); // Convenience method to get the right color
HashMap<String, Integer> colors = new HashMap<String, Integer>() {{
    put("r", Color.red(c));
    put("g", Color.green(c));
    put("b", Color.blue(c));
}};
mRef.child("devices/color").setValue(colors);
```

How did we make hardware development as fast and easy as software development?

Abstraction!

Arduino: HAL over an 8 bit uC

Android: Makes mobile development simple

Firebase: Device to device communication and data model

Debugging is (still) your greatest challenge

- Helpful to have hardware: voltmeters, oscilloscope, bus pirates
- Your toolchain is the biggest time sink
  - Program firmware
  - Program tablet
  - Switch connectors, connect device to tablet
  - How do you view exceptions, how do you view bus traffic?
- Android has a few nice tools (ADB over WiFi)
- Arduino has next to none--this is an advantage for Brillo (and others)
- IoT forces you to get creative!

Questions?

Appetize: https://goo.gl/nJtOIY
Twitter: @asciimike, @firebase, @googledevs
Github: mcdonamp/firebutton (code & slides)