

# Getting started with Phidgets on Android

## Overview

Phidgets can run directly plugged in to Android devices with a USB port and Android 3.1 (API 12) or later. Otherwise, Android can control a Phidget remotely, over the network, via the Phidget Webservice.

In addition to Android-specific examples for Phidgets, the more general Java documentation has further examples on running Phidgets using Java.

## Libraries

First, we need to set up the proper environment and get the necessary files off the Phidgets web site. Visit the drivers section at [www.phidgets.com](http://www.phidgets.com) and get the latest:

- Android Libraries

This download contains:

- A libs/ folder
- A jar file containing the general Phidget java library (phidget21.jar)
- A jar file for directly driving USB devices from a USB port on the Android device (PhidgetsUSB.jar)

You will need these libraries for working with Phidgets on Android.

We also recommend that you download the following reference materials from the programming section:

- Programming Manual
- The Product Manual for your device
- Java Getting Started Guide
- Java API Manual
- Code Samples written for Android
- Code Samples written for Java

You can find a high level discussion about programming with Phidgets in general in the Programming Manual. The Product manual for your device also contains an API section that describes limitations, defaults, and implementation details specific to your Phidget. You may want to have these manuals open while working through these instructions.

For a list of supported classes and commands, please refer to the Java API Manual.

## Environment

Android applications can be developed under Windows, Mac OS, or Linux. Our examples were developed under Eclipse. This getting started assumes that you have Eclipse installed and setup for Android development. If not, you can learn about this here:

Step 1: <http://developer.android.com/sdk/installing.html>

Step 2: <http://developer.android.com/resources/tutorials/hello-world.html>

## Setting up a Phidgets Project

To set up an Eclipse project for using Phidgets via the **Phidget Webservice**:

- Create a new Eclipse Android project, or open an existing project.
- Copy the /libs folder and the phidget21.jar library from the Android Libraries download into your project root folder.
- Add phidget21.jar to your project libraries (Properties->Java build path->Libraries->Add JARs...)
- Add the following to your AndroidManifest.xml file:

```
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
```

To set up an Eclipse project for using Phidgets via the **USB Host** (Directly attached):

- Create a new Eclipse Android project, or open an existing project.
  - Copy the /libs folder, the phidget21.jar library and the PhidgetsUSB.jar library from the Android Libraries download into your project root folder.
  - Add phidget21.jar and PhidgetsUSB.jar to your project libraries (Properties->Java build path->Libraries->Add JARs...)
  - Add the following to your AndroidManifest.xml file:
- ```
<uses-feature android:name="android.hardware.usb.host" />
```
- Call `com.phidgets.usb.Manager.Initialize(this);` and `com.phidgets.usb.Manager.Uninitialize();` before/after any other Phidgets calls to initialize/uninitialize the USB Host subsystem.

You should also import the Phidgets libraries in your source code:

```
import com.phidgets.*;
import com.phidgets.event.*;
```

The project now has access to Phidgets and we are ready to begin coding.

## Examples

Phidgets provides two Android-specific examples. Both are for the Interface Kit; one demonstrates opening a remote Phidget via the Phidget Webservice, the other demonstrates opening a Phidget attached directly via USB Host.

To use these examples, import them into Eclipse:

- File -> Import... -> General -> Existing Projects Into Workspace -> (Next)
- Select root directory -> Browse to find and select an existing example such as the InterfaceKitExample
- Select all files, click Finish

To run the examples:

- Right-click on project in Package Explorer (To open this, use Window -> Show View -> Package Explorer)
- Select Run As... -> Android Application

## Coding For Your Phidget

Please review the Java Getting Started Guide for Java coding guidelines. Review the rest of this manual for Android-specific tips.

### open Calls

The following open calls are supported for Network attached Phidgets:

```
void open(int serial, java.lang.String ipAddress, int port)
void open(int serial, java.lang.String ipAddress, int port, java.lang.String password)
void openAny(java.lang.String ipAddress, int port)
void openAny(java.lang.String ipAddress, int port, java.lang.String password)
void openLabel(java.lang.String label, java.lang.String ipAddress, int port)
void openLabel(java.lang.String label, java.lang.String ipAddress, int port,
               java.lang.String password)
```

The following open calls are supported for USB Host attached Phidgets:

```
void open(int serial)
void openAny()
void openLabel(java.lang.String label)
```

The following open calls are unsupported:

```
void open(int serial, java.lang.String serverID)
void open(int serial, java.lang.String serverID, java.lang.String password)
void openAny(java.lang.String serverID)
void openAny(java.lang.String serverID, java.lang.String password)
void openLabel(java.lang.String label, java.lang.String serverID)
void openLabel(java.lang.String label, java.lang.String serverID,
               java.lang.String password)
```

## Phidgets Events Notes

The Phidgets API relies heavily on event-based programming. It's important to consider that Phidgets events come in from their own thread context - separate from the main thread. If you want to update a GUI element directly from an event handler, you need to do this in the context of the main thread, because on Android, the GUI can only be updated from the main thread.

This is accomplished most easily by using the `runOnUiThread(Runnable);` call. Here, special care must be taken if you plan on using any object from the event in the `Runnable run()` method - because by default this method will get run sometime in the future, after the event had returned and it's objects are no longer valid. You need to either implement a `wait()/notify()` scheme to wait for the `run()` method to run in the main thread context before allowing the event to return, or copy any data you wish to pass. Both of these methods are illustrated in the examples.

The importance of this cannot be stressed enough, because errors here will lead to segmentation faults in the native library that cannot be caught in your Java application.

## USB Host Notes

As of API Level 12 (Android 3.1), USB Host is officially supported in Android. We take advantage of this with our `PhidgetsUSB.jar` library. This library is meant to be used via JNI by our Native library (`libs/armeabi/libphidgets21.so`), and you should not try to call any of the methods, or directly use any of the classes that it exposes, with two exceptions:

In `com.phidgets.usb.Manager`:

```
static public void Initialize(Context userContext);
```

This needs to be called before you call `open` on a Phidget - otherwise, the `open` call will raise an `UNSUPPORTED` exception. This call takes a context object and retrieves the Application context from it, which is required for using the USB system. Generally you would call this in your `onCreate()`, passing in your Activity object: `com.phidgets.usb.Manager.Initialize(this);`

```
static public void Uninitialize();
```

This cleans up everything done by the `Initialize` call. This should be called after you call `close` on all Phidgets, generally in your `onDestroy()`.

Do NOT import `com.phidget.usb.*`; - this will bring in Phidget and Manager objects which will clash with the objects in `com.phidgets.*`. and just generally cause confusion.

After you call `open` on a Phidget, you will get permissions requests from the system for access to all attached Phidgets - this is required so that the native library can read in the device serial numbers, versions, and other parameters. Once you have given permission, this is remembered until the device is unplugged - even across application restarts.