

# Getting started with Phidgets in Flash AS3

## Environment and Libraries

Phidgets supports development in Flash for all types of Phidgets using ActionScript 3.0. First, we need to set up the proper environment and get the necessary files off the Phidgets website. Visit the drivers and programming section at [www.phidgets.com](http://www.phidgets.com) and get the latest:

- Phidget Framework
- Flash AS3 examples

You will need the ActionScript libraries from the Flash AS3 examples to program with Phidgets, and the Phidget Framework to use them. After the Framework is installed, you must also enable the WebService in the Phidget Control Panel for any programs using the AS3 libraries. We also recommend that you download from the programming section the following reference materials:

- AS3 API Manual
- Programming Manual
- The Product Manual for your device

The AS3 API manual contains calls and events for every type of Phidget and can be used as a reference. 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.

## Setting up a Phidgets Project

The Phidget examples were written using ActionScript 3.0 under Flash CS3 and this tutorial assumes its use. Other development environments should work provided they support ActionScript 3.0, and each would be set up in a similar manner.

First launch Flash and generate a new Flash File with a descriptive name such as PhidgetTest. You will also need to copy the “com” folder that is located in the Flash AS3 examples into your project root to use the Phidget Library.

## Coding For Your Phidget

Before you can use the Phidget, you must include a reference to the library in the action frame (Window | Actions). In Actionscript 3.0:

```
import com.phidgets.PhidgetInterfaceKit;  
import com.phidgets.events.*;
```

Afterwards, the Phidget object will need to be declared and then initialized. For example, we can declare a PhidgetInterfaceKit with:

```
public var phid:com.phidgets.PhidgetInterfaceKit;  
phid = new PhidgetInterfaceKit();
```

The object name for any type of Phidget is listed in the API manual. Every type of Phidget also inherits functionality from the Phidget base class.

## Connecting to the Phidget

The program can try to connect to the Phidget through an open call. Open will continuously try to connect to a Phidget, based on the parameters given, even trying to reconnect if it gets disconnected. This means that simply calling open does not guarantee you can use the Phidget immediately. We can handle this by using event driven programming and tracking the AttachEvents and DetachEvents, or checking the isAttached property.

```
phid.open("localhost", 5001);
```

The API manual lists all of the available modes that open provides. In Flash, the parameters can be used to open the first Phidget of a type it can find or based on its serial number. One important thing to remember is that when working with Phidgets, a local connection will reserve the device until closed. This means other programs that open the Phidget may end up preventing other instances from retrieving data. The one connection per device limit does not apply when exclusively using the Phidget WebService, and the Actionscript libraries by extension.

## Event Driven Programming

We recommend the use of event driven programming when working with Phidgets. In Actionscript 3.0, we hook an event handler with the following code:

```
phid.addEventListener(PhidgetDataEvent.SENSOR_CHANGE, onSensorChange);  
  
function onSensorChange(evt:PhidgetDataEvent):void{  
    trace (evt.Data); //Echo  
  
    //Insert your code here  
}
```

With this method, the code inside onSensorChange will get executed every time the InterfaceKit reports a change on one of its analog inputs. The values from the report can be accessed from the PhidgetDataEvent object properties.

Some events such as Attach and Detach belong to the base Phidget object and thus are common to all types of Phidgets. Please refer to the API manual for a full list of events and their usage.

## **Working directly with the Phidget**

Some values can be directly read and set on the Phidget and used as an alternative to event driven programming. Simply use the instance properties or call member functions such as `getSensorValue(index: int)` or `setOutputState(index: int, val: Boolean)` for `InterfaceKits`.

## **Working with multiple Phidgets**

Multiple Phidgets of the same type can easily be run inside the same program. In our case, it requires another `PhidgetInterfaceKit` instance to be defined and initialized. The new instance can then be set up, opened and used in the same process as the previous one.

If the application needs to distinguish between the devices, `open` can be called with the serial number of a specific Phidget.

## **Flash Security Settings**

During debugging or after publishing the project, you may encounter some difficulties with Flash network security settings either inside or outside of the development environment with Phidgets. Permissions for your project folder can be added through the settings manager at [http://www.macromedia.com/support/documentation/en/flashplayer/help/settings\\_manager04a.html](http://www.macromedia.com/support/documentation/en/flashplayer/help/settings_manager04a.html), under "Always trust files in these locations" | "Edit locations..." | "Add location...".

## **Other Phidgets**

The design given in this document can also be followed for almost all Phidgets. For example, if you were using a `PhidgetRFID` instead of an `InterfaceKit`, you would declare a `PhidgetRFID` instead of a `PhidgetInterfaceKit`. The methods and events available would change but they can be accessed in a similar manner.