

Getting started with Phidgets in Flex 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 and get the latest:

- Phidget Framework
- Flex AS3 examples

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

- Flash AS3 examples
- 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. The Programming Manual gives a high level discussion about programming with Phidgets in general. The Product manuals also contain an API section that describes limitations, defaults, and implementation details specific to your Phidget. You may want to have these manuals and examples open while working through these instructions.

Setting up a Phidgets Project

The Phidget examples were written using Actionscript 3.0 under Adobe Flex Builder 3 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 Flex and generate a new project with a descriptive name such as PhidgetTest. You will then need to add the Phidget Library to your project library path. This can be done under Project | Properties | Flex Build Path on the Library tab, by clicking the “Add SWC” button and selecting the Phidget21Library.swc from the Flex examples. As an alternative, you can copy the “com” folder that is located in the Flash AS3 examples into your project source directory and it should work as well.

Coding For Your Phidget

Before you can use the Phidget, you must import references from the library to the device in the main body of code. In MXML and Actionscript 3.0:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">
```

```

<mx:Script>
    <![CDATA[
        import com.phidgets.PhidgetInterfaceKit;
        import com.phidgets.events.*;
        //More code goes here
    ]]>
</mx:Script>
</mx:Application>

```

Afterwards, the Phidget object will need to be declared and then initialized. For example, a start function is set to execute on initialization, and then an instance of a PhidgetInterfaceKit is declared and then set inside with:

```

public var phid:com.phidgets.PhidgetInterfaceKit;
public function start():void{
    phid = new PhidgetInterfaceKit();
    //More code goes here
}

```

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

Next, the program needs to try and 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);
```

The onSensorChange method 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.

```
private function onSensorChange (evt:PhidgetDataEvent):void{
    //Insert your code here
}
```

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, 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.