Getting started with Phidgets in AutoIt

Environment and Libraries

First, we need to set up the proper environment and get the necessary files off the Phidgets website. Visit the drivers section at www.phidgets.com and get the latest:

- Phidget Framework

You will need the Phidget Framework to program with or use Phidgets. We also recommend that you download from the programming section the following reference materials:

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

The COM 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 the manuals and examples open while working through these instructions.

Setting up a Phidgets Project

Applications using Phidgets can be developed in AutoIt v3 through the COM API and this tutorial assumes its use. The Phidget framework will register the necessary .dlls for you. To begin, create a new AutoIt script file and open it with your preferred editor. Then, create a simple user interface to work with and then place an input control for the purpose of capturing some simple output.

```
#include <GUIConstantsEx.au3>

GUICreate("Phidget Test")
$InputControl = GUICtrlCreateInput("", 20, 40, 160)
GUISetState(@SW_SHOW)

;Phidget initialization code goes here

Do
     $msg = GUIGetMsg()
Until $msg = $GUI_EVENT_CLOSE

;Phidget Event Handlers go here
```

Coding For Your Phidget

Before you can use the Phidget, you must create the object from the Phidget21COM library at initialization. This can be accomplished through a call to ObjCreate. For example, we can declare and create a PhidgetInterfaceKit with:

```
$oPhid1 = ObjCreate("Phidget21COM.PhidgetInterfacekit")
```

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

Connecting to the Phidget

The program can try to connect to the Phidget through a call to open. 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 account for a connection by using event driven programming and tracking the AttachEvents and DetachEvents, or by calling WaitForAttachment. WaitForAttachment will block indefinitely until a connection is made to the Phidget, or an optional timeout is exceeded.

```
$oPhid1.Open()
$oPhid1.WaitForAttachment(3000)
If NOT $oPhid1.IsAttached Then
      MsgBox(0, "Error", "Phidget Device not attached")
      Exit
EndIf
```

The different parameters and open calls can be used to open the first Phidget of a type it can find, open based on a serial number, or even open across the network. The API manual lists all of the available modes that open provides. One important thing to remember is that when working with Phidgets, a local connection will reserve the device until closed. This prevents any other instances from retrieving data from the Phidget, including other programs. The one connection per device limit does not apply when exclusively using the Phidget Webservice.

Event Driven Programming

We recommend the use of event driven programming when working with Phidgets. This allows the program to execute other tasks until the Phidget generates a new event. In AutoIt, you enable event handlers for a Phidget at initialization with the following line:

```
$oPhidEvents = ObjEvent($oPhid1, "phid1_")
```

AutoIt will use the phidl1_ prefix we declared when looking for the handler to execute. You can define the handler functions for events such as OnSensorChange or Attached elsewhere in your code:

```
Func phid1_OnSensorChange($Index, $SensorValue)
      GUICtrlSetData($InputControl, $SensorValue)
EndFunc
```

With this method, the code inside phid1_OnSensorChange will get executed every time the PhidgetInterfaceKit reports a change on one of its analog inputs.

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 inside polling loops used as an alternative to event driven programming. Simply use the instance properties such as SensorValue(Index as Long) or OutputState(Index as Long) for PhidgetInterfaceKits.

```
$oPhid1.OutputState(0) = "true"
```

Note that when setting booleans on the Phidget, it must be encapsulated in quotes. Alternatively, you can usually use the integer value 0 for false and 1 for true.

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 object to be defined and initialized. The new object 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.

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 PhidgetInterfacekit, you would declare a PhidgetRFID object instead of a PhidgetInterfaceKit. The methods and events available would change but they can be accessed in a similar manner.