

# Getting started with Phidgets in VBA

## 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](http://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 the following reference materials from the programming section:

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

In the VBA examples you can find more robust examples for some Phidgets. 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

The Phidget examples were written using Microsoft Excel 2000 and this tutorial assumes its use. Other environments such as Microsoft Office should work provided they support VBA, and each would be set up in a similar manner.

To begin, launch Excel with a new workbook for our project. Launch the VBA editor (Tools | Macro | Visual Basic Editor) and open "ThisWorkbook" in the navigator.

## Coding For Your Phidget

Before you can use the Phidget, you must include the library in your project. This can be accomplished from the references screen (Tools | References...) by checking the box beside "Phidget Library 2.1", or by browsing to the location the framework was installed and choosing the Phidget21COM.dll.

Afterwards, the Phidget object will need to be declared and then initialized inside the workbook item. For example, we can declare and create a PhidgetInterfaceKit at the top of the code with:

```
Public WithEvents phid As PhidgetInterfaceKit
Private Sub Workbook_Open()
    Set phid = New PhidgetInterfaceKit
End Sub
```

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 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.

```
phid.Open  
phid.WaitForAttachment(3000)
```

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.

At the end of your program, don't forget to call close to free any locks on the Phidget.

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)  
    phid.Close  
End Sub
```

## Event Driven Programming

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

```
Private Sub phid_OnSensorChange(ByVal Index As Long, ByVal SensorValue As Long)  
    Range("A2").Select  
    ActiveCell.Offset(Index, 0).Value = SensorValue  
End Sub
```

With this method, the code inside onSensorChange will get executed every time the InterfaceKit reports a change on one of its analog inputs. You can let the editor generate the procedure declaration for you through the drop down menu at the top of the code window.

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 `InterfaceKits`.

```
phid.OutputState(4) = 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` 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.

## 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.