

Getting started with Phidgets in Delphi

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 files installed with the Phidget Framework to program and use Phidgets. We also recommend that you download the following reference materials from the programming section:

- Examples in Delphi
- 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 these manuals open while working through these instructions.

Setting up a Phidgets Project

The Phidget examples were written using Borland Delphi 2005 and 7, and this tutorial assumes its use. The ActiveX objects installed with the Phidget libraries are not automatically imported into Delphi, so to begin you will need to manually import them:

1. Go to Component >> Import ActiveX Control...
2. Select the Phidget Library 2.1 and click install
3. In the next popup select the "install into new package" tab
4. Give the package a suitable name and description such as "Phidgets", and then click OK
5. Compile and Install the newly created package if not done so automatically.

In newer versions of Delphi:

1. Go to Component >> Import Component...
2. Choose "Import ActiveX Control" and click Next
3. Pick Phidget Library 2.1 from the list and click Next
4. Set the palette Page to ActiveX and check the Generate Component Wrappers option
5. Choose Install into a new package and click next
6. Give the package a name and description such as "Phidgets", and then click Finish

Once installed, all future projects will be able to use the Phidget ActiveX objects.

Coding For Your Phidget

Before you can use the Phidget, you must declare and initialize its ActiveX object. The simplest method is to place the control from the ActiveX component tab on to your form. For this tutorial, create a PhidgetInterfaceKit control (PhidgetInterfaceKit1) and then add a text edit field to the form for the purpose of capturing simple output.

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, we need to tell the program to try and connect to the Phidget through a call to `open()`. The `open` will tell the program to continuously try to connect to a Phidget, based on the parameters given, even trying to reconnect if it gets disconnected. This means that calling `open` does not guarantee you can use the Phidget immediately. We can handle it by either 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.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    PhidgetInterfaceKit1.Open(-1);
    PhidgetInterfaceKit1.WaitForAttachment(3000);
end;
```

The parameters can also be used to open the first Phidget of a type it can find, open based on its 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.

You can call `Close` any time outside of the Phidget's own event handlers to end the connection.

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 fires a new event. In Delphi, you can hook an event handler by selecting the Phidget object and then double clicking its event in the object inspector. For a PhidgetInterfaceKit, an event handler looks like the following:

In Delphi, when the event library was installed it added Phidget ActiveX components for each type of Phidget. These ActiveX components will show up in the toolbox under the ActiveX tab. The ActiveX object for your Phidget can be added to the form and then hooked to the event handlers you wish to use like any other control. For a Phidget21COMIPhidgetInterfaceKitEvents control, the event handler looks like the following:

```
procedure TForm1.PhidgetInterfaceKit1SensorChange(ASender: TObject; Index,
    SensorValue: Integer);
begin
```

```
Edit1.Text := IntToStr(Index) + ': ' + IntToStr(SensorValue);  
end;
```

With this method, the code inside `PhidgetInterfaceKit1SensorChange` will get executed every time the `InterfaceKit` 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 COM API manual for a full list of events and their general 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 `CallString` such as `SensorValue[Index]` or `OutputState[Index]` for `InterfaceKits`.

```
PhidgetInterfaceKit1.OutputState[0] := 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` `ActiveX` object to be added to the project. The new instance can then be set up, opened and used in the same fashion 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 place a `PhidgetRFID` `ActiveX` object instead of a `PhidgetInterfaceKit`. The methods and events available would change but they can be accessed in a similar manner.

Enabling Logging

Often it's a good idea to enable logging during development for debugging purposes. This log, depending on the level set, will record certain events and errors from `Phidgets`. In Delphi, this is accomplished by calling `EnableLogging(logLevel, filename)` on the `Phidget` object.

```
PhidgetInterfaceKit1.EnableLogging(6, 'testlog.txt');
```

Similarly, you can call `DisableLogging()` at the end of your program to shut it off. Should you need to contact `Phidgets` for support, including this log is very helpful for revealing the cause of the problem. Please see the `Programming Manual` for a general discussion on `Phidget` logging and the levels provided.