

# Getting started with Phidgets in C#

## 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 use and program with Phidgets. We also recommend that you download the following reference materials from the programming section:

- .NET API Manual
- Programming Manual
- The Product Manual for your device
- Example Programs written in C#

The .NET API manual lists 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 Visual C# 2005 and this tutorial assumes its use. Newer versions of Visual Studio Express are freely available for download from Microsoft. Older versions of Visual Studio work as well and would be set up in a similar manner (Note: you would have to recreate the user interface in the examples for Visual Studio versions earlier than 2005). In Visual Studio:

- Generate a new C# Windows Application with a descriptive name such as PhidgetTest.
- Launch the Add Reference window (Project | Add Reference).
- Under the .NET tab, select the most recent Phidget21.NET library. If it does not appear in this list, then you can Browse to the Phidget Framework installation directory and add the Phidget21.NET.dll. For earlier versions of Visual Studio, you will want to use the Phidget21.NET1.1.dll instead.
- Place a TextBox on your main form for the purpose of capturing output.
- Hook the form's Load and FormClosing events. Phidget initialization and shutdown will take place there.

The project now has access to Phidgets and we are ready to begin coding.

## Coding For Your Phidget

Before you can use the Phidget, you must include a reference in the code to the libraries. Launch the code editor for your form and add this to your using statements:

```
using Phidgets;  
using Phidgets.Events;
```

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

```
namespace PhidgetTest  
{  
    public partial class Form1 : Form  
    {  
        //The Phidget object declaration  
        private InterfaceKit ifKit;  
  
        public Form1()  
        {  
            InitializeComponent();  
        }  
        //... Form1_Load and Form1_OnClosing 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 a call to `open()`. 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 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 by calling `waitForAttachment`. `waitForAttachment` will block indefinitely until a connection is made to the Phidget, or an optional timeout is exceeded.

```
private void Form1_Load(object sender, EventArgs e)  
{  
    ifKit = new InterfaceKit();  
    ifKit.open();  
    ifKit.waitForAttachment(3000);  
}
```

The different types of `open` can be used with parameters to try and get the first device 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.

## Event Driven Programming

We recommend the use of event driven programming when working with Phidgets. In C# we can hook an event handler at loading with the following code:

```
ifKit.SensorChange += new SensorChangeEventHandler(ifKit_SensorChange);
```

The ifKit\_SensorChange method is defined as follows:

```
void ifKit_SensorChange(object sender, SensorChangeEventArgs e)
{
    textBox1.Text = "Index " + e.Index + " Value: " + e.Value;
}
```

With this function, the code inside ifKit\_SensorChange 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.

At the end of your program, unhook any events and call Application.DoEvents(). This will make sure there are no outstanding events being processed before calling close.

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    ifKit.SensorChange -= new SensorChangeEventHandler(ifKit_SensorChange);
    //run any events in the message queue
    Application.DoEvents();
    ifKit.close();
}
```

## Working directly with the Phidget

Some values can be read and sent directly to the Phidget, simply use the instance members and properties. This is also how you would set properties on the Phidget such as the output state or sensor sensitivity. These functions can be used inside a polling loop as an alternative to event driven programming.

```
int val = ifKit.sensors[0].Value;
```

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