

# Getting started with Phidgets in Microsoft Visual Studio C\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 and programming section at [www.phidgets.com](http://www.phidgets.com) and get the latest:

- Phidget Framework
- Visual Studio C/C++ Examples

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

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

The C 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 Visual Studio C++ 2005 and this tutorial assumes its use. Other environments such as other versions of Visual Studio work as well and would be set up in a similar manner. If you are using a different compiler, please consult your compiler documentation for specific details on how to link to external libraries. In Visual C++ 2005:

- Generate a new C/C++ console project with a descriptive name such as PhidgetTest.
- Open the project properties window.
- Navigate to Configuration Properties | C/C++.
- Add "C:\Program Files\Phidgets" to the additional include directories field.
- Navigate to Configuration Properties | Linker | Input.
- Add "C:\Program Files\Phidgets\phidget21.lib" to the additional dependencies field.

The project now has access to the Phidget21 function calls and we are ready to begin coding.

**Note:** it is assumed that the Phidgets Framework is installed into the C:\Program Files\Phidgets directory.

## Coding For Your Phidget

Before you can use the Phidget, you must include a reference to the library header.

```
#include <phidget21.h>
```

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

```
int _tmain(int argc, _TCHAR* argv[])
{
    CPhidgetInterfaceKitHandle ifKit = 0; //Declare an InterfaceKit handle
    CPhidgetInterfaceKit_create(&ifKit); //Create the InterfaceKit object

    //More code goes here
    return 0;
}
```

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. 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 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.

```
CPhidget_open((CPhidgetHandle)ifKit, -1);
CPhidget_waitForAttachment((CPhidgetHandle)ifKit, 2500);
```

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.

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

```
CPhidget_close((CPhidgetHandle)ifKit);
CPhidget_delete((CPhidgetHandle)ifKit);
```

## Event Driven Programming

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

```
CPhidgetInterfaceKit_set_OnSensorChange_Handler (ifKit, SensorChangeHandler, NULL);

int __stdcall SensorChangeHandler(CPhidgetInterfaceKitHandle IFK, void *usrptr,
                                  int Index, int Value)
{
    printf("Sensor: %d > Value: %d\n", Index, Value);
    //Insert your code here
    return 0;
}
```

With this function, the code inside SensorChangedHandler 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 read and sent directly to the Phidget. Simply use the C API functions such as CPhidgetInterfaceKit\_getSensorValue() for PhidgetInterfaceKits.

```
int val, i;
for (i = 0; i < 10; i++) {
    CPhidgetInterfaceKit_getSensorValue(phid, 0, &val);
    printf("Value: %d\n", val);
}
```

The functions can be used inside a polling loop as an alternative to event driven programming. Above, a sensor attached to the PhidgetInterfaceKit is read and its value is displayed to screen.

## 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 call CPhidgetRFID\_create instead of CPhidgetInterfaceKit\_create. The functions and events available would change but they can be accessed in a similar manner.