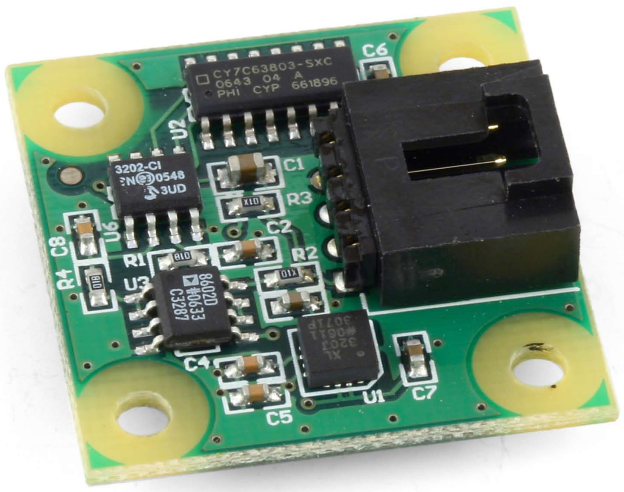
1053 - PhidgetAccelerometer



Product Features

- Dual axis accelerometer
- Measures ± 5 gravities (± 49.0 m/s2) change per axis.
- Measure both dynamic acceleration (vibration) and static acceleration (gravity or tilt).
- Internally calibrated at the factory to be very accurate
- Connects directly to a computer's USB port

Programming Environment

Operating Systems: Windows 2000/XP/Vista, Windows CE, Linux, and Mac OS X

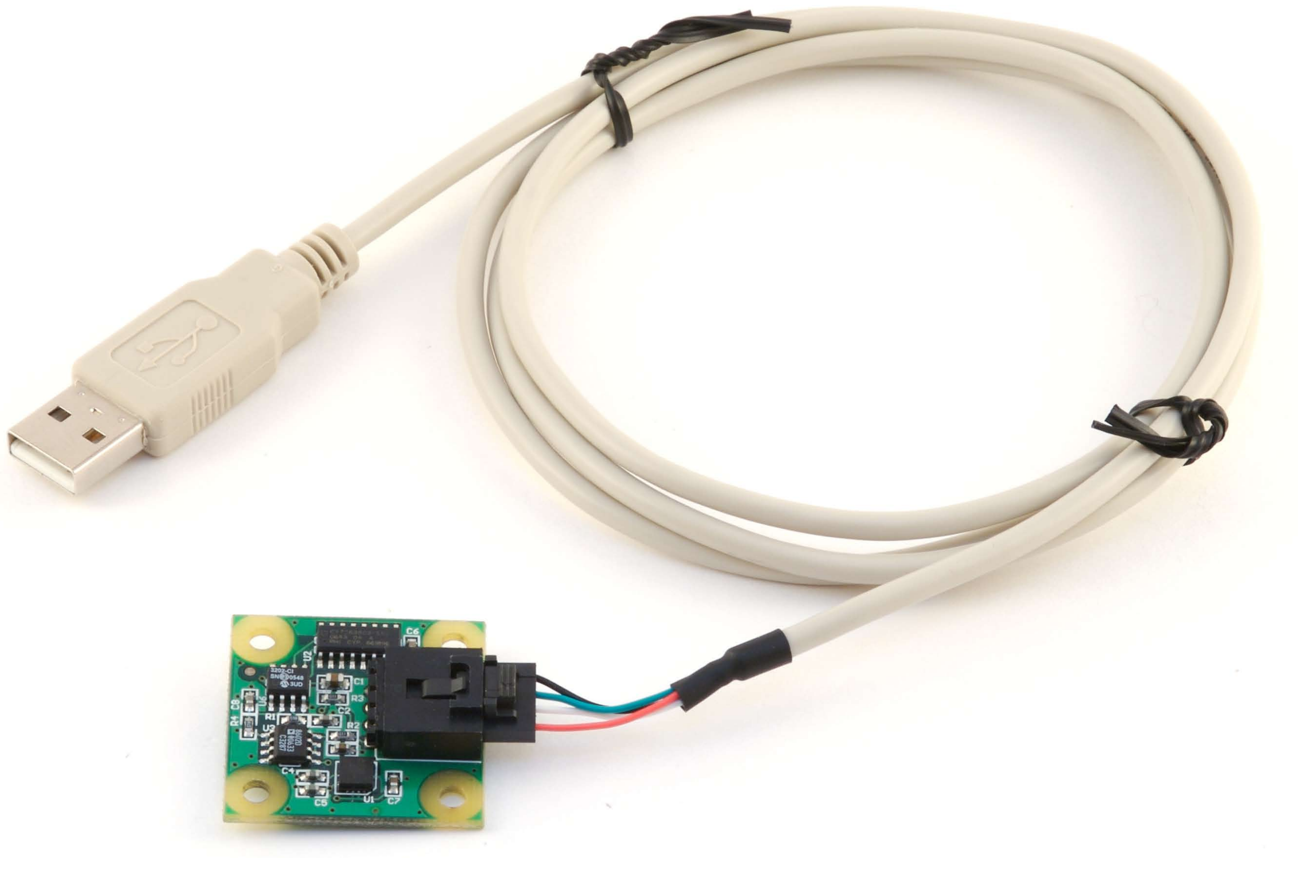
Programming Languages (APIs): VB6, VB.NET, C#.NET, C++, Flash 9, Flex, Java, LabVIEW, Python, Max/MSP, and Cocoa.

Examples: Many example applications for all the operating systems and development environments above are available for download at www.phidgets.com.

Installing the hardware

The kit contains:

- A PhidgetAccelerometer
- A custom USB cable



Connect the PhidgetAccelerometer to your PC using the custom USB cable.

Downloading and Installing the software

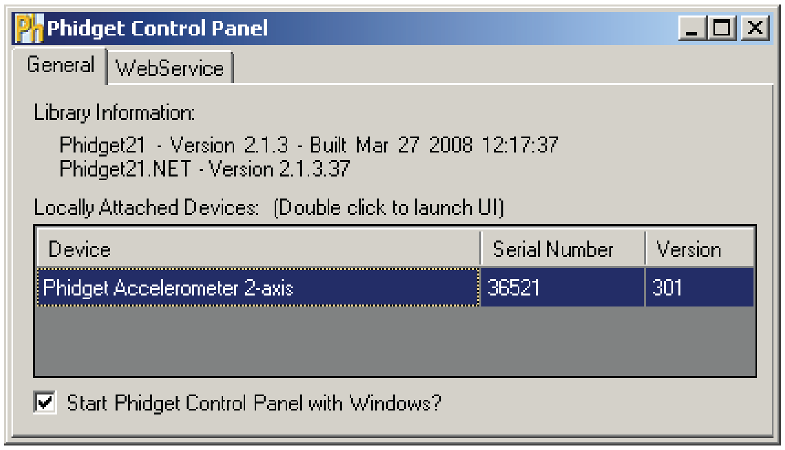
If you are using Windows 2000/XP/Vista

Go to www.phidgets.com >> Downloads >> Windows

Download and run Phidget21.MSI

You should see the Ph icon on the right hand corner of the Task Bar.

Testing the PhidgetAccelerometer Functionality



Library Information:

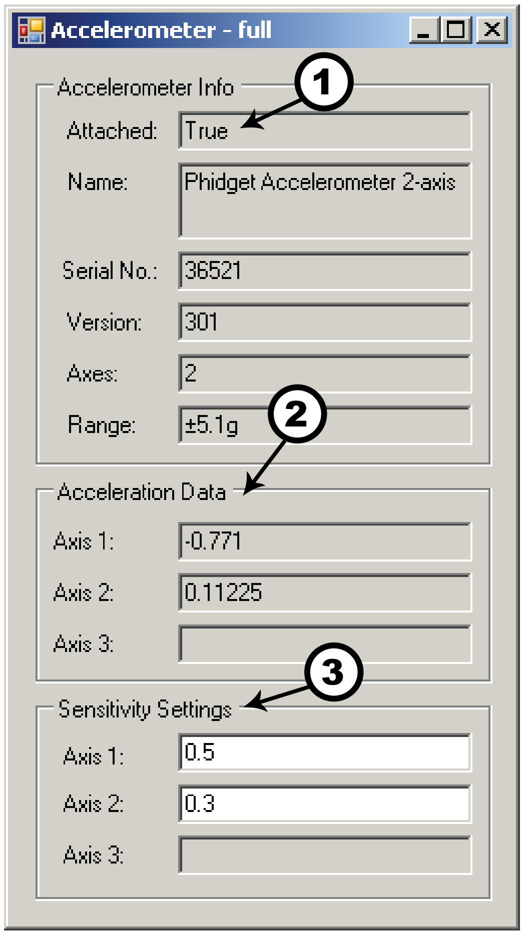
Phidget21 - Version 2.1.3 - Built Mar 27 2008 12:17:37
Phidget21.NET - Version 2.1.3.37

Locally Attached Devices: (Double click to launch UI)

Device	Serial Number	Version
Phidget Accelerometer 2-axis	36521	301

☑ Start Phidget Control Panel with Windows?

Double Click on the Ph icon to activate the Phidget Control Panel and make sure that ***PhidgetAccelerometer*** is properly attached to your PC.



Accelerometer Info

Attached:	True
Name:	Phidget Accelerometer 2-axis
Serial No.:	36521
Version:	301
Axes:	2
Range:	±5.1g

Acceleration Data

Axis 1:	-0.771
Axis 2:	0.11225
Axis 3:	

Sensitivity Settings

Axis 1:	0.5
Axis 2:	0.3
Axis 3:	

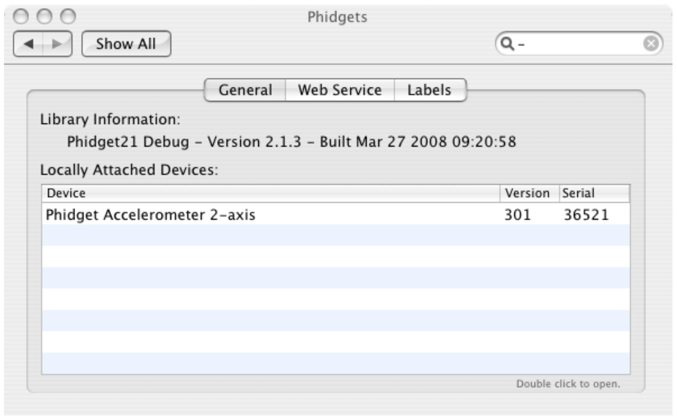
1. Double Click on ***Phidget Accelerometer*** in the Phidget Control Panel to bring up Accelerometer-full and check that the box labelled Attached contains the word True.
2. Move the Accelerometer board and you should see the data change to reflect the change of position along the 2 axes.
3. You can adjust the sensivity by typing a number for any or all of the 2 axes.

If you are using Mac OS X

Go to www.phidgets.com >> downloads >> Mac

Download Mac OS X Framework

Testing the PhidgetAccelerometer functionality



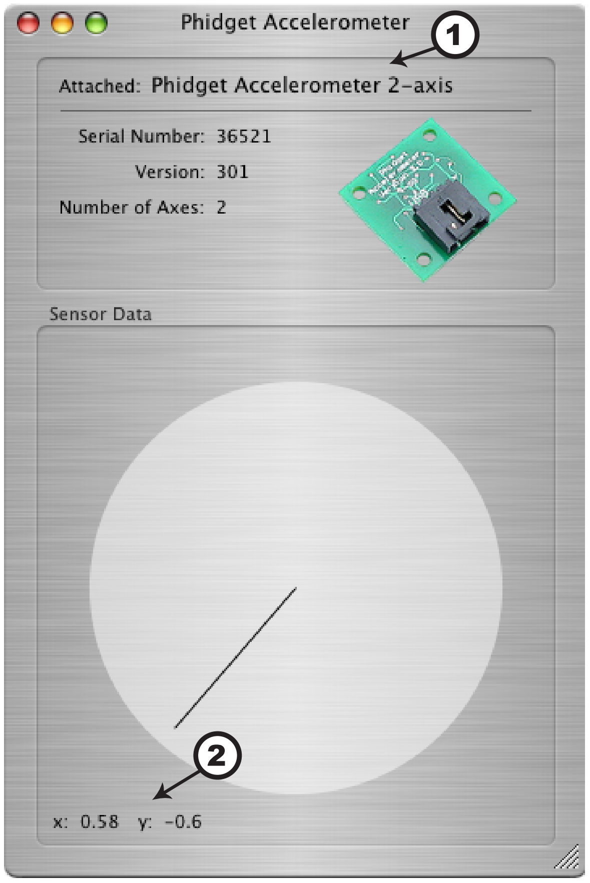
Library Information:
Phidget21 Debug - Version 2.1.3 - Built Mar 27 2008 09:20:58

Locally Attached Devices:

Device	Version	Serial
Phidget Accelerometer 2-axis	301	36521

Double click to open.

Click on System Preferences >> Phidgets (under Other) to activate the Preference Pane. Make sure that the ***Phidget Accelerometer*** is properly attached.



Attached: Phidget Accelerometer 2-axis

Serial Number: 36521
Version: 301
Number of Axes: 2

x: 0.58 y: -0.6

1. Double Click on ***Phidget Accelerometer*** in the Phidgets Preference Pane to bring up Phidget Accelerometer Example and check that the PhidgetAccelerometer is properly connected.
2. Move the Accelerometer board and you should see the data change to reflect the change of position along the 2 axes.

If you are using Linux

Go to www.phidgets.com >> Downloads >> Linux

- Download Linux Source
- Have a look at the readme file
- Build Phidget21

The most popular programming languages in Linux are C/C++ and Java.

Note: Many Linux systems are now built with unsupported third party drivers. It may be necessary to uninstall these drivers for our libraries to work properly.

Note: Phidget21 for Linux is a user-space library. Applications typically have to be run as root, or udev/hotplug must be configured to give permissions when the Phidget is plugged in.

If you are using Windows Mobile/CE 5.0 or 6.0

Go to www.phidgets.com >> Downloads >> Windows Mobile/CE

Download x86 or ARMV4I, depending on the platform you are using. Mini-itx and ICOP systems will be x86, and most mobile devices, including XScale based systems will run the ARMV4I.

The CE libraries are distributed in .CAB format. Windows Mobile/CE is able to directly install .CAB files.

The most popular languages are C/C++, .NET Compact Framework (VB.NET and C#). A desktop version of Visual Studio can usually be configured to target your Windows Mobile Platform, whether you are compiling to machine code or the .NET Compact Framework.

Programming a Phidget

Phidgets' philosophy is that you do not have to be an electrical engineer in order to do projects that use devices like sensors, motors, motor controllers, and interface boards. All you need to know is how to program. We have developed a complete set of Application Programming Interfaces (API) that are supported for Windows, Mac OS X, and Linux. When it comes to languages, we support VB6, VB.NET, C#.NET, C, C++, Flash 9, Flex, Java, LabVIEW, Python, Max/MSP, and Cocoa.

Architecture

We have designed our libraries to give you the maximum amount of freedom. We do not impose our own programming model on you.

To achieve this goal we have implemented the libraries as a series of layers with the C API at the core surrounded by other language wrappers.

Libraries

The lowest level library is the C API. The C API can be programmed against on Windows, CE, OS X and Linux. With the C API, C/C++, you can write cross-platform code. For systems with minimal resources (small computers), the C API may be the only choice.

The Java API is built into the C API Library. Java, by default is cross-platform - but your particular platform may not support it (CE).

The .NET API also relies on the C API. Our default .NET API is for .NET 2.0 Framework, but we also have .NET libraries for .NET 1.1 and .NET Compact Framework (CE).

The COM API relies on the C API. The COM API is programmed against when coding in VB6, VBScript, Excel (VBA), Delphi and Labview.

The ActionScript 3.0 Library relies on a communication link with a PhidgetWebService (see below). ActionScript 3.0 is used in Flex and Flash 9.

Programming Hints

- Every Phidget has a unique serial number - this allows you to sort out which device is which at runtime. Unlike USB devices which model themselves as a COM port, you don't have to worry about where in the USB bus you plug your Phidget in. If you have more than one Phidget, even of the same type, their serial numbers enable you to sort them out at runtime.
- Each Phidget you have plugged in is controlled from your application using an object/handle specific to that phidget. This link between the Phidget and the software object is created when you call the .OPEN group of commands. This association will stay, even if the Phidget is disconnected/reattached, until .CLOSE is called.
- The Phidget APIs are designed to be used in an event-driven architecture. While it is possible to poll them, we don't recommend it. Please familiarize yourself with event programming.

Networking Phidgets

The PhidgetWebService is an application written by Phidgets Inc. which acts as a network proxy on a computer. The PhidgetWebService will allow other computers on the network to communicate with the Phidgets connected to that computer. ALL of our APIs have the capability to communicate with Phidgets on another computer that has the PhidgetWebService running.

The PhidgetWebService also makes it possible to communicate with other applications that you wrote and that are connected to the PhidgetWebService, through the PhidgetDictionary object.

API documentation

We maintain API manuals for COM (Windows), C (Windows/Mac OSX/Linux), Action Script, .Net and Java. Look at the section that corresponds to the Phidget you are using. These manuals can be accessed in different ways:

Using Downloads on main menu

Click on Downloads >> Operating System (i.e. Windows) >> Platform (i.e. C#) >> API Document (i.e. Net API Manual)

Using Products on Home Page

Click on InterfaceKits (under Products) >> 1018 PhidgetInterfaceKit 8/8/8 >> API Manual (Under Software Information)

Using Information on Home Page

Click on Information (under Main Menu) >> Your API Manual (under Phidgets API Manuals)

Examples

We have written examples to illustrate how the APIs are used. Examples for the C#.NET programming language, including .exe files for each of the examples in the directory root.

Due to the large number of languages and devices we support, we cannot provide examples in every language for every Phidget. Some of the examples are very minimal, and other examples will have a full-featured GUI allowing all the functionality of the device to be explored. Most developers start by modifying existing examples until they have an understanding of the architecture.

To get the examples, go to www.phidgets.com and click on Downloads. Under Step 2: click on your platform of choice and click on the File Name beside Examples.

Support

- Click on Live Support on www.phidgets.com to chat with our support desk experts
- Call the support desk at 1.403.282.7335 8:00 AM to 5:00 PM Mountain Time (US & Canada) - GMT-07:00
- E-mail sales@phidgets.com

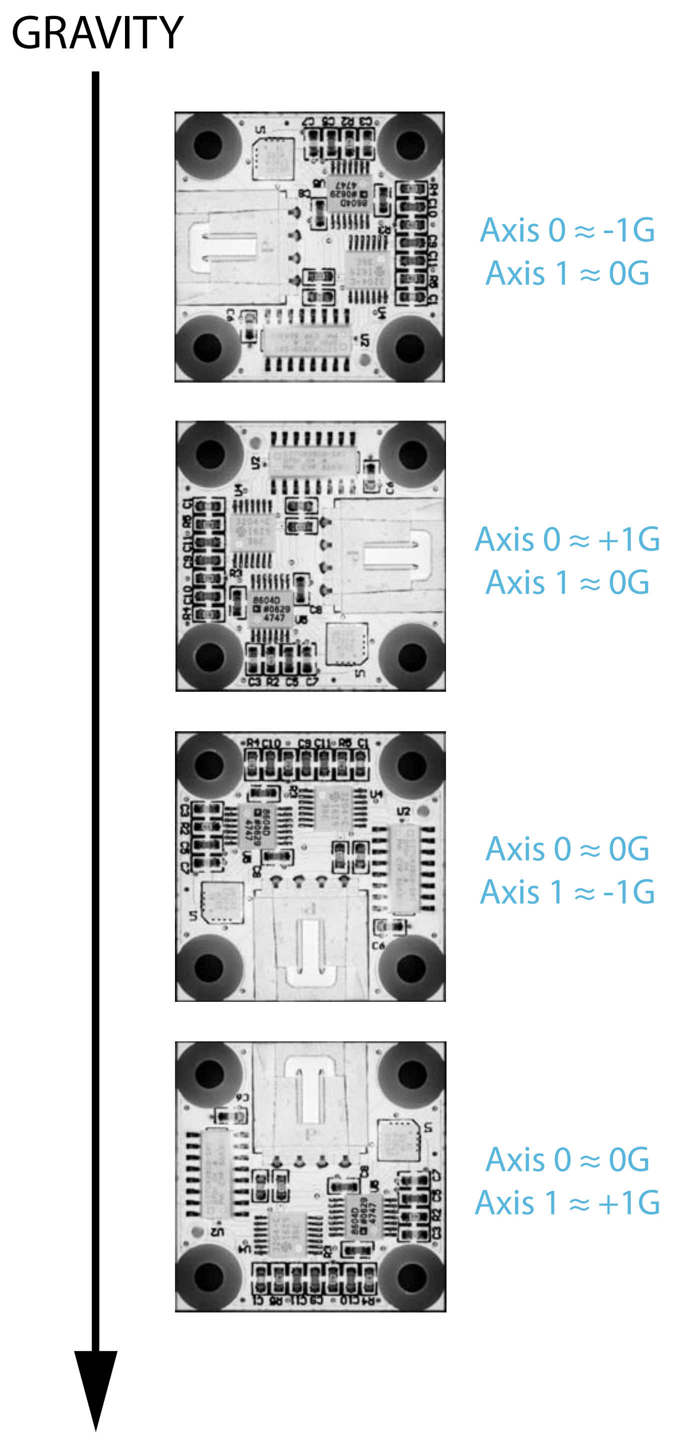
Simple example written in C#

```
/* - Accelerometer simple -
 ****************************************************************************************
 * This simple example simply creates an accelerometer object, initializes it, hooks the
 * event handlers and opens it.  It then waits for an accelerometer to be attached and
 * waits for events to be fired. We preset the sensitivity of each axis to 1.0 to make it
 * easier to see the event data.  For a more detailed example with the ability to see and
 * manipulate all of the accelerometer Phidget’s properties, see the Accelerometer-full
 * example.
 *
 * Please note that this example was designed to work with only one Phidget Accelerometer
 * connected. For an example using multiple Phidget Accelerometers, please see a
 * “multiple” example in the Accelerometer Examples folder.
 *
 * Copyright 2007 Phidgets Inc.
 * This work is licensed under the Creative Commons Attribution 2.5 Canada License.
 * To view a copy of this license, visit http://creativecommons.org/licenses/by/2.5/ca/
 */
using System;
using System.Collections.Generic;
using System.Text;
using Phidgets; //needed for the accelerometer class and the phidgets exception class
using Phidgets.Events; //needed for the phidget event handling

namespace Accelerometer_simple
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                //Declare an accelerometer object
                Accelerometer accel = new Accelerometer();

                //Hook the basic event handlers
                accel.Attach += new AttachEventHandler(accel_Attach);
                accel.Detach += new DetachEventHandler(accel_Detach);
                accel.Error += new ErrorEventHandler(accel_Error);

                //hook the phidget specific event handlers
                accel.AccelerationChange += new AccelerationChangeEventHandler
                                                      (accel_AccelerationChange);

                //open the acclerometer object for device connections
                accel.open();

                //get the program to wait for an accelerometer device to be attached
                Console.WriteLine(“Waiting for accelerometer to be attached....”);
                accel.waitForAttachment();

                //Set the sensitivity of each of the available axes on the accelerometer
                //to 1.0 so that we can actually see the changes instead a flurry of
                //text on the screen
                for (int i = 0; i < accel.axes.Count; i++)
                {
                    accel.axes[i].Sensitivity = 1.0;
                }

                //Get the program to wait for user input before moving on so that we can
                //watch for some events
                Console.WriteLine("Press any key to end");
                Console.Read();

                //If user input has been read, we can now terminate the program, so
                //close the phidget object
                accel.close();

                //set the object to null to clear it from memory
                accel = null;

                //if no exceptions have been trhown at this point, the program can
                //terminate safely
                Console.WriteLine("ok");
            }
            catch (PhidgetException ex)
            {
                Console.WriteLine(ex.Description);
            }
        }

        //Attach event handler...Display the serial number of the attached
        //accelerometer to the console
        static void accel_Attach(object sender, AttachEventArgs e)
        {
            Console.WriteLine("Accelerometer {0} attached!",
                                    e.Device.SerialNumber.ToString());
        }

        //Detach event handler...Display the serial number of the detached accelerometer
        //to the console
        static void accel_Detach(object sender, DetachEventArgs e)
        {
            Console.WriteLine("Accelerometer {0} detached!",
                                    e.Device.SerialNumber.ToString());
        }

        //Error event handler...Display the description of the error to the console
        static void accel_Error(object sender, ErrorEventArgs e)
        {
            Console.WriteLine(e.Description);
        }

        //Acceleration change event handler...Display which axis the device is
        //accelerating in as well as the measured acceleration value
        static void accel_AccelerationChange(object sender,
                                            AccelerationChangeEventArgs e)
        {
            Console.WriteLine("Axes {0} Acceleration {1}", e.Index, e.Acceleration);
        }
    }
}
```

Technical Section

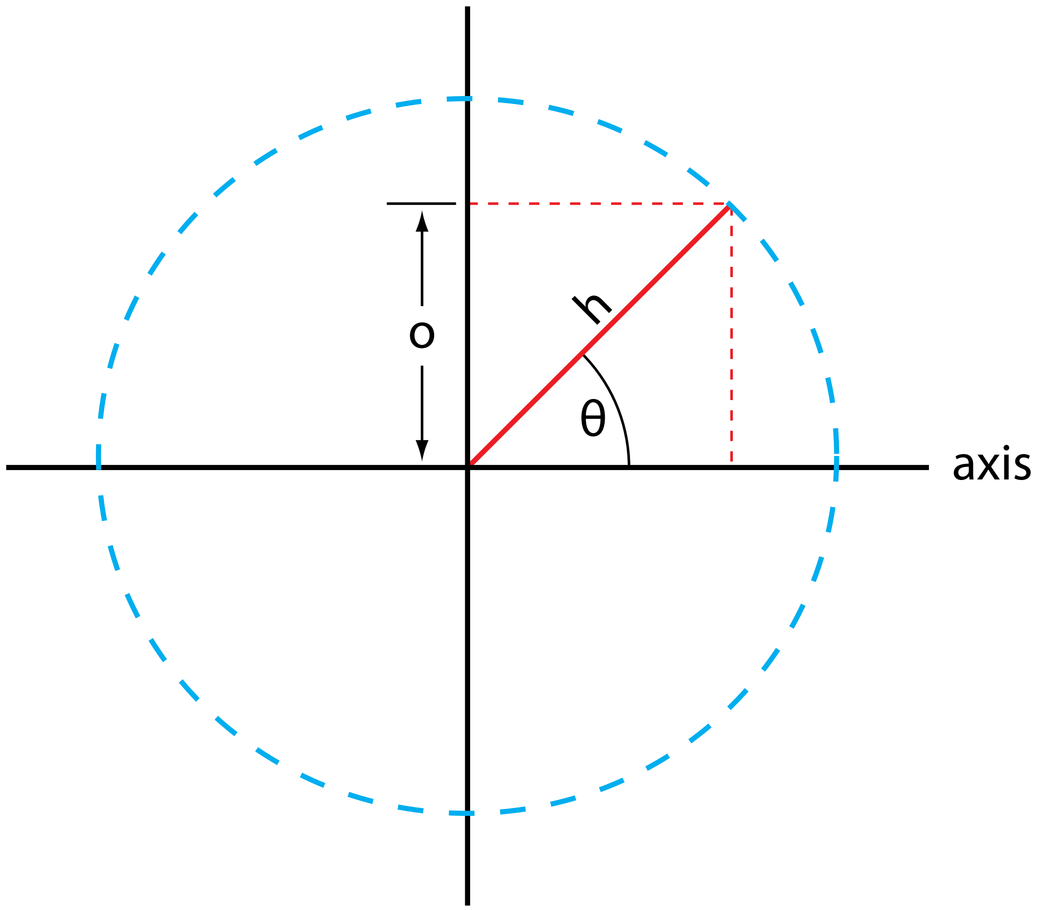
The PhidgetAccelerometer is a dual axis accelerometer that can measure ± 5 gravities (± 49.0 m/s2) change per axis. It can measure both dynamic acceleration (vibration) and static acceleration (gravity or tilt). It is internally calibrated to be very accurate. The sensor used is an Analog Devices ADXL320.

The theoretical static response of the PhidgetAccelerometer to gravitational acceleration (1g) in space is diagrammed below. This diagram is intended for use in sensor orientation.



Calculating Tilt Angle from Acceleration

A simple trigonometric calculation based on the unit circle is used when trying to determine the tilt angle of the device. Although gravitational acceleration is limited to 1G by definition, the PhidgetAccelerometer measures both static and dynamic acceleration up to ±5G, and can easily output values greater than 1G during a tilt measurement if the device is in motion. Accelerations in excess of 1G are not valid during tilt measurements; values should be limited to this maximum to ensure that the formula below remains appropriate.



The unit circle represents the maximum gravitational acceleration of 1G. A line drawn from the origin out to the unit circle then represents the present acceleration vector of the device in a given axis. For that axis, the angle θ (which represents the tilt angle of the device in that axis) can be calculated using the length of the hypotenuse (labelled h; this value is always 1 since it touches the unit circle) and the distance of the endpoint of the hypotenuse from the axis in question (labelled o, which represents the value of acceleration reported by the device). The formula for doing so becomes:

$$\theta = \textit{arcsin}\ (o/h)$$

In an example, the device reports the acceleration of axis 0 to be 0.7071G. The calculation of *arcsin* (0.7071G/1G) yields the result of 45°, the current tilt angle of the device in axis 0.

API (Software Technical)

We document API Calls specific to the 1053. Functions common to all Phidgets are not covered here. This section is deliberately generic - for calling conventions in a specific language, refer to that languages' API manual.

Functions

int AxisCount() [get] : Constant = 2

Returns the number of axis this PhidgetAccelerometer can measure acceleration on.

double Acceleration (int AxisIndex) [get] : Units = G (9.8m/s^2)

Returns the acceleration of a particular axis. At a standstill each axis will measure between -1.0 and 1.0 g's depending on orientation.

This value will always be between getAccelerationMin and getAccelerationMax.

Index 0 is the x-axis, and 1 is the y-axis.

double AccelerationMax (int AxisIndex) [get] : Constant

Returns the maximum acceleration value that this axis will report.

double AccelerationMin (int AxisIndex) [get] : Constant

Returns the minimum acceleration value that this axis will report.

double AccelerationChangeTrigger (int AxisIndex) [get,set]

An event that is issued when the Acceleration varies by more than the AccelerationChangeTrigger property.

Events

OnAccelerationChange (int AxisIndex, double Acceleration) [event]

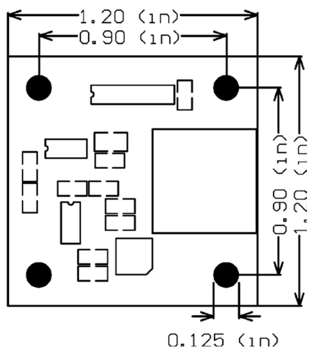
An event issued when a change in acceleration occurs that is greater than the AccelerationChangeTrigger.

Device Specifications

Update Rate	60 samples/second
Measurement Range (XYZ Axis)	±5G (49.05 m/s^2)
Bandwidth (XYZ Axis)	30Hz
Axis 0 Noise Level (X Axis)	1.9mG standard deviation (σ)
Axis 1 Noise Level (Y Axis)	1.9mG standard deviation (σ)
USB Power Current Specification	500mA max
Device Quiescent Current Consumption	16mA
Device Active Current Consumption	16mA max

Mechanical Drawing

1:1 scale



Product History

Date	Product Revision	Comment
October 2003	DeviceVersion 100	Based on ADXL311 (2g)
April 2004	DeviceVersion 200	Based on ADXL210 (10g)
September 2004	DeviceVersion 300	Based on ADXL320 (5g)