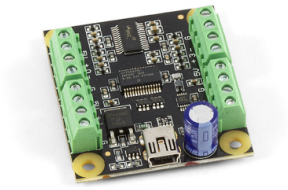
User Guide



Go to this device's product page [1]

Getting Started

Checking the Contents

You should have received:

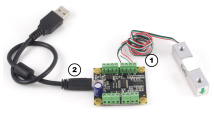
- A PhidgetBridge 4-Inputs
- A Mini-USB cable

In order to test your new Phidget you will also need:

- A wheatstone bridge based sensor

Connecting the Pieces

1. Connect the load cell to the PhidgetBridge - use bridge 0. We are using a 3133 - Micro Load Cell (0-5kg) - CZL635. Connect the red wire to 5V, the green wire to +, the white wire to -, and the black wire to G. If your Load Cell is not documented, refer to the technical section of this page for instructions on how to connect it.
2. Connect the PhidgetBridge to your computer using the Mini-USB cable.



Testing Using Windows 2000 / XP / Vista / 7

Make sure you have the current version of the Phidget library installed on your PC. If you don't, follow these steps:

1. Go to the Quick Downloads section on the Windows page
2. Download and run the Phidget21 Installer (32-bit, or 64-bit, depending on your system)
3. You should see the icon on the right hand corner of the Task Bar.

Running Phidgets Sample Program

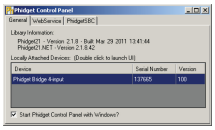
Double clicking on the icon loads the Phidget Control Panel; we will use this program to ensure that your new Phidget works properly.

The source code for the **Bridge-full** sample program can be found in the quick downloads section on the C# Language Page. If you'd like to see examples in other languages, you can visit our Languages page.

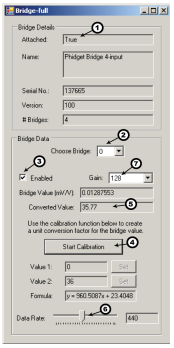
Updating Device Firmware

If an entry in this list is red, it means the firmware for that device is out of date. Double click on the entry to be given the option of updating the firmware. If you choose not to update the firmware, you can still run the example for that device after refusing.

Double Click on the icon to activate the Phidget Control Panel and make sure that the **PhidgetBridge 4-input** is properly attached to your PC.



1. Double Click on **PhidgetBridge 4-input** in the Phidget Control Panel to bring up Bridge-full and check that the box labelled Attached contains the word True.
2. If you have connected your device to the same bridge as we did, select bridge 0.
3. Click to enable the bridge.
4. Click on the Start Calibration Button. Enter 0 for Value 1 (no weight on the load cell). Put a known weight on the load cell and enter the number in Value 2 box. The calibration formula is displayed in the formula box. If your sensor is not a load cell, you can still use the Calibration functionality - you just need two known values for your sensor.
5. Put a different weight on the load cell and the bridge value gets converted using the calibration formula.
6. You can use the slider to adjust the data rate from 8ms to 1000ms in increments of 8ms.
7. You can set the gain to 1, 8, 16, 32, 64, 128; in general a higher gain value gives you lower noise and higher resolution.



Testing Using Mac OS X

1. Go to the Quick Downloads section on the Mac OS X page
2. Download and run the Phidget OS X Installer
3. Click on System Preferences >> Phidgets (under Other) to activate the Preference Pane
4. Make sure that the PhidgetBridge 4-input is properly attached.
5. Double Click on PhidgetBridge 4-input in the Phidget Preference Pane to bring up the Bridge-full Sample program. This program will function in a similar way as the Windows version.

Using Linux

For a step-by-step guide on getting Phidgets running on Linux, check the Linux page.

Using Windows Mobile / CE 5.0 / CE 6.0

For a step-by-step guide on getting Phidgets running on Windows CE, check the Windows CE page.

Technical Details

How to Calibrate the Bridge

We have observed a 1.5% difference between gain=1 and gain=8. This may require that each system (PhidgetBridge + sensors) are calibrated as a whole. For maximum accuracy, decide on and keep with a chosen gain before calibrating the system.

Expensive sensors will ship with a certificate of calibration specifying, often in mv/V, how the sensor responds to stimulus. Less expensive will have to be calibrated, which requires having at least two points where you know accurately what is being measured. In the case of weight measurement, this would be a known force or weight.

Record the output from the PhidgetBridge at one known point, and at a second known point. It helps if the two values are reasonably far apart. Use the values to make a linear equation to convert the PhidgetBridge output in mV/V (called X) to the appropriate unit you are measuring (called Y). Two calibration coefficients (a,b) set the slope

and offset for the calibration: (Y = aX + b). It's possible to use more than two points, if available.

The C# Bridge-full example shows how to do a 2-point calibration and apply the coefficients programmatically.

Gain Setting vs. Resolution

We report the measured voltage in a ratiometric unit known as mv/V. This is how the maximum range of sensors that use strain gauges is usually specified. mV/V is the output value in mV of the measured sensor, scaled for a 1V sensor supply voltage. This value will correspond to the physical quantity that the sensor is measuring, regardless of the actual voltage supplied to the sensor.

Gain	Resolution	Range
1	119 nV/V	± 1000 mV/V
8	14.9 nV/V	± 125 mV/V
16	7.45 nV/V	± 62.5 mV/V
32	3.72 nV/V	± 31.25 mV/V
64	1.86 nV/V	± 15.625 mV/V
128	0.93 nV/V	± 7.8125 mV/V

When choosing the Gain setting, it's best to use the highest gain possible that can still measure the full range of your sensor. For an individual unit, you can apply the maximum stimulus to the sensor, and ensure the BridgeValue reported is well within the range for the Gain setting you have chosen. If many units are being deployed, it's best to consult the data sheet for the strain gauge and look for maximum offset.

Some wheatstone bridges - most often those produced from silicon and used in pressure sensors, will have a very wide offset, and large manufacturing variation in the offset. This will restrict the gain to lower settings, particularly if the application must support a number of deployed systems with the expected variation. Fortunately, the very high precision electronics used in the PhidgetBridge means that in many application, higher gain is not necessary to get adequate accuracy and resolution.

Connecting your Strain Gauge/Load Cell

Load cells are pressure sensors that can be used with the 1046 - PhidgetBridge.

For more information, see our Load Cell Primer.

If no documentation is available for your strain gauge, it's possible to use a multimeter to determine how to connect it, provided there are no electronics in the sensor. First, measure resistance between the 4 wires. There are 6 combinations - two combinations will have a resistance 20-40% higher than the other four. Choose one of these high-resistance combinations, and wire it into 5V and G on the PhidgetBridge. Connect the other two wires into +/-. Apply a load - if the mV/V responds in the opposite way to your expectations, flip the +/- wires.

Measurement Considerations

The PhidgetBridge is designed to measure voltages as a ratio of the supply voltage - it's not practical to make measurements of absolute voltages with this product.

For maximum accuracy, all wires from the PhidgetBridge to the sensor should be the same length and thickness. Changes in temperature will change the resistance of the wires - if they are all the same, the errors will cancel out.

Each bridge input can be powered down, reducing power consumption with Bridge-Sensors, and useful for reducing heating of sensors, which can introduce errors.

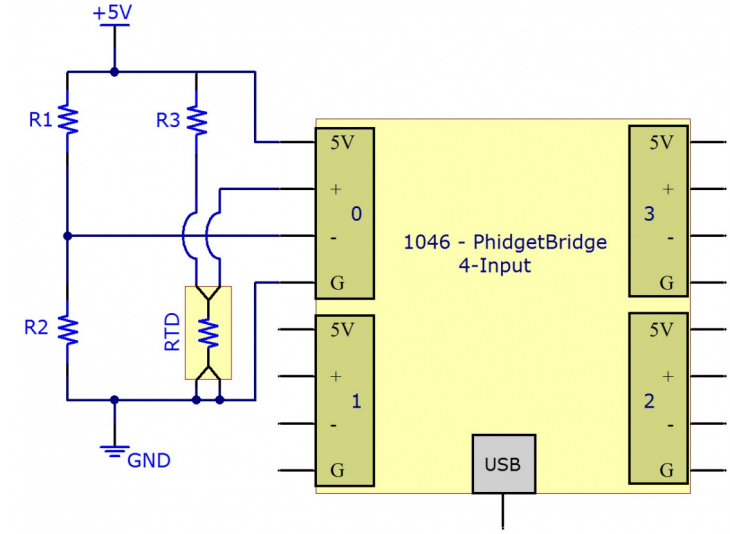
Changing the Data Rate

Using a slower sampling rate will reduce the noise in the measurements dramatically. The noise figures are specific to individual applications and sensors. The lowest noise level achievable is 5nV/V RMS.

Measuring Resistive Thermal Devices (RTD)

Using a Wheatstone Bridge

A Wheatstone bridge is the classic method of measuring unknown resistances, and requires three resistors of known values. It uses the current in each leg of the bridge to create a voltage differential between both voltage dividers. Using the voltage differential and the three known resistors, the resistance of the fourth resistor can be determined.



This diagram shows how to connect the RTD to a Wheatstone bridge, and then to a PhidgetBridge 4-Input.

To determine the resistance of the RTD, the following formula can be used:

$$R_{RTD} = \frac{R_3 \times [R_2 + V_B \times (R_1 + R_2)]}{R_1 - (R_1 + R_2) \times V_B}$$

Where V_B is the Bridge Value given by the PhidgetBridge (in mV/V) , and R_1, R_2 and R_3 are the resistances of the known resistors.

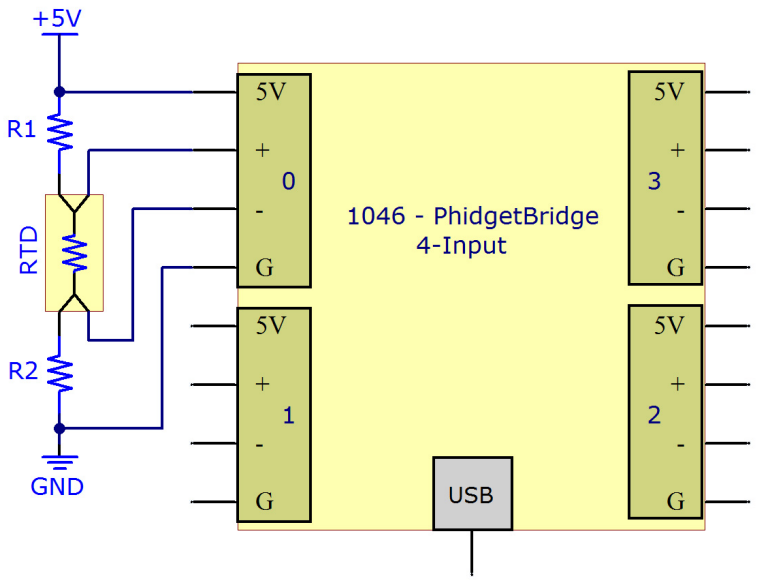
Using a Voltage Divider

The alternate method requires only two resistors. This reduces the amount of error that can be introduced into the system due to resistor tolerances. A voltage is applied to the two resistors and the RTD in series. The voltage drop across the RTD is measured. Using the voltage drop and the values of the two resistors, the resistance of the RTD can be determined.

To determine the resistance of the RTD, the following formula can be used:

$$R_{RTD} = (R_1 + R_2) \times \frac{V_B}{1 - V_B}$$

Where V_B is the Bridge Value given by the PhidgetBridge (in mV/V) , and R_1 and R_2 are the resistances of the known resistors.



This diagram illustrates how to connect the RTD to the PhidgetBridge with a voltage divider cirtuit.

Getting Higher Accuracy

In order to get the highest accuracy from the RTD, consider the following:

- Use resistors with a high degree of tolerance. There will be less variability in the manufacturing of 0.1% resistors when compared to 1% resistors.
- Measure the known resistors with an ohmmeter. By obtaining the most accurate measurements for the known resistances, the formula will result in a more accurate measurement of the RTD.
- Use a moving average when obtaining the Bridge Value to reduce the amount of noise in the measured signal.
- Estimate or Measure the resistance of the +5V and GND wires between the RTD and the 1046 PhidgetBridge. Add this resistance to the two resistors.
- Turn off the power to the RTD (by disabling the channel on the PhidgetBridge) to reduce self-heating of the RTD.
- By using higher resistor values (> 1 Kilo ohm), there will be less self-heating of the RTD, but the resolution of the measurement will be reduced somewhat. We recommend 1 Kilo Ohm resistors as a reasonable trade off.

API

We document API Calls specific to this product in this section. Functions common to all Phidgets and functions not applicable to this device are not covered here. This section is deliberately generic. For calling conventions under a specific language, refer to the associated API manual in the Quick Downloads section for that language. For exact values, refer to the device specifications.

Functions

int InputCount [get] : Constant = 4

Returns the number of bridges supported by this PhidgetBridge.

double BridgeValue(int index) [get]

Returns the value of the selected input, in mV/V. If the input is not enabled, this will throw an EPHIDGET_UNKNOWNVAL exception. If the bridge is saturated, this will be equal to BridgeMax or BridgeMin and an error event will be fired - in this case, gain should be reduced.

double BridgeMax(int index) [get]

Returns the maximum value that the selected Bridge can measure, in mV/V. This value will depend on the selected gain. At a gain of 1, BridgeMax == 1000mV/V.

double BridgeMin(int index) [get]

Returns the minimum value that the selected Bridge can measure, in mV/V. This value will depend on the selected gain. At a gain of 1, BridgeMin == -1000mV/V.

boolean Enabled(int index) [get,set]

Gets / Sets the enabled state of a Bridge. This applies power between +5v and Ground and starts measuring the differential on the +/- pins. By default, all Bridges are disabled, and need to be explicitly enabled on startup.

Gains Gain(int index) [get,set]

Gets / Sets the gain for a selected bridge. Supported gains are 1, 8, 16, 32, 64 and 128. Note that increasing the gains will reduce the measurable voltage difference by the gain factor, with +-1000mV/V being the maximum, with no gain.

int DataRate [get,set]

Gets / Sets the data rate, in ms. Data rate applies to all 4 bridges simultaneously. Setting a slower data rate will reduce noise at the cost of sample time. Also note that each bridge is being sampled only 1/4 of the time - this is probably ok for very stable signals, but for changing signals, it's may be best to set a higher sampling rate and do averaging in software.Data rate must be a multiple of 8ms. Trying to set something between multiples of 8 will cause an EPHIDGET_INVALIDARG exception to be thrown.

int DataRateMax [get] : Constant = 8

Gets the maximum supported data rate, in ms.

int DataRateMin [get] : Constant = 1000

Gets the minimum supported data rate, in ms.

Events

OnBridgeData(int index, double value) [event]

An event that is issued at the specified DataRate, for each enabled bridge. Value is the bridgeValue, in mV/V.

OnError(int ErrorCode, String ErrorDescription)

The PhidgetBridge will throw error events under certain circumstances:

ErrorCode = EEPHIDGET_OUTOFRANGE

A bridge input has gone out of range. This indicates either an overrange or underrange condition. If possible, gain should be reduced.

See the ErrorDescription string for specific error details.

Product History

Date	Board Revision	Device Version	Comment
May 2011	0	100	Product Release
May 2011	0	101	getLabelString fixed for labels longer than 7 characters