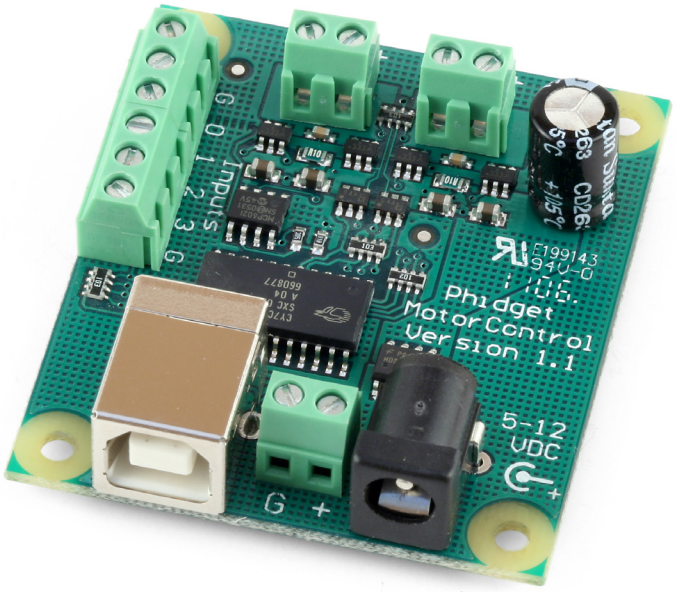
Product Manual

1060 - PhidgetMotorControl LV



Phidgets 1060 - Product Manual

For Board Revision 0

Contents

Product Features

- Control two low-voltage DC motors independently for direction, velocity and acceleration
- Motors are powered from an external power supply (5 to 12VDC with center positive polarity)
- The 1060 provides overcurrent protection
- Includes four digital inputs used to convey the state of limit switches, push buttons, relays. etc.
- Connects directly to a computer's USB port

Programming Environment

Operating Systems: Windows 2000/XP/Vista/7, 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 >> Programming.

Connection

The board connects directly to a computer's USB port.

Getting Started

Checking the Contents

You should have received:

- A PhidgetMotorControl LV
- A USB Cable

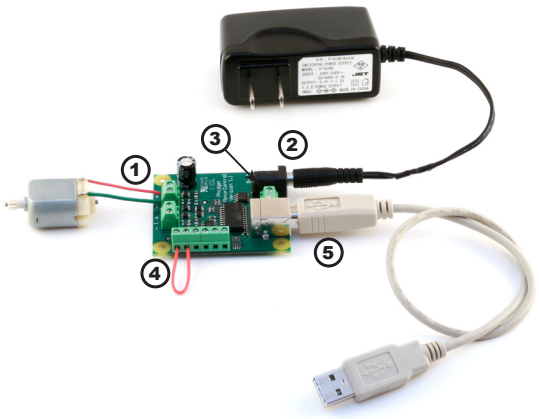
In order to test your new Phidget you will also need:

- A 5 to 12VDC power supply with center positive polarity*
- A low-voltage DC motor
- A piece of wire to test the digital inputs

* When using a power supply with a **barrel connector**, make sure that your power supply has **center positive polarity**.

Connecting all the pieces

1. Connect the low voltage Motor to the PhidgetMotorControl LV
2. Plug in a power supply using the barrel connector.
3. You can also connect a power supply to the Terminal Block. Be sure to observe correct polarity.
4. Connect one end of the wire to the Ground connector and the other end to connector 0.
5. Connect the MotorControl LV to your computer using the USB cable.



Testing Using Windows 2000/XP/Vista

Downloading the Phidgets drivers

Make sure that you have the current version of the Phidget library installed on your PC. If you don't, do the following:

Go to www.phidgets.com >> Drivers

Download and run Phidget21 Installer (32-bit, or 64-bit, depending on your PC)

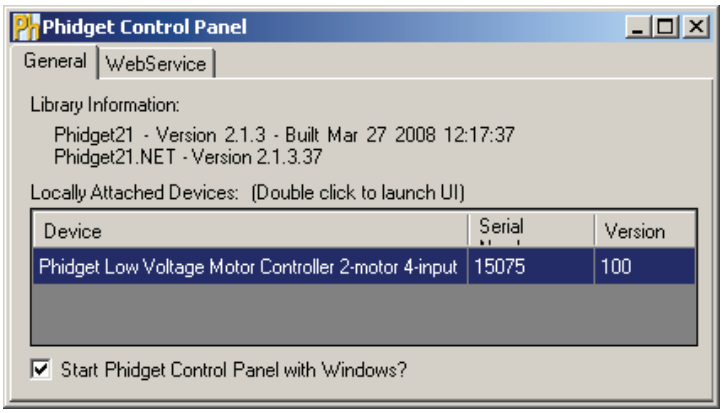
You should see the icon on the right hand corner of the Task Bar.

Running Phidgets Sample Program

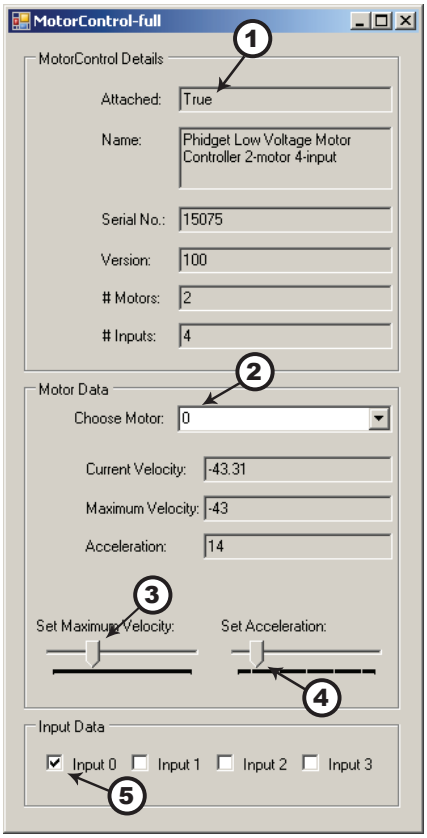
Double clicking on the icon loads the Phidget Control Panel; we will use this program to make sure that your new Phidget works properly.

The source code for the MotorControl-full sample program can be found under C# by clicking on www.phidgets.com >> Programming.

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



1. Double Click on **Phidget Low Voltage Controller** in the Phidget Control Panel to bring up MotorControl-full and check that the box labelled Attached contains the word True.
2. Select the connected motor. If you have connected your motor at the same place as the one in the picture on page 3, it should be at position 0.
3. Move the slider pointer to set the maximum velocity. The maximum velocity is shown in the Maximum Velocity box, and the current motor velocity in the box above.
4. Change the acceleration by moving the slider. The value is displayed in the Acceleration box.
5. Test the digital input by disconnecting the wire end connected to the digital input connector. The tick mark in the box will go away.



Testing Using Mac OS X

- Click on System Preferences >> Phidgets (under Other) to activate the Preference Pane
- Make sure that the **Phidget Low Voltage Controller** is properly attached.
- Double Click on Phidget **Phidget Low Voltage Controller** in the Phidget Preference Pane to bring up the MotorControl-full Sample program. This program will function in a similar way as the Windows version.

If you are using Linux

There are no sample programs written for Linux.

Go to www.phidgets.com >> Drivers

Download Linux Source

- Have a look at the readme file
- Build Phidget21

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

Notes:

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.

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 >> Drivers

Download x86, ARMV4I or MIPSII, 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.

Documentation

Programming Manual

The Phidget Programming Manual documents the Phidgets software programming model in a language and device unspecific way, providing a general overview of the Phidgets API as a whole. You can find the manual at www.phidgets.com >> Programming.

Getting Started Guides

We have written Getting Started Guides for most of the languages that we support. If the manual exists for the language you want to use, this is the first manual you want to read. The Guides can be found at www.phidgets.com >> Programming, and are listed under the appropriate language.

API Guides

We maintain API references for COM (Windows), C (Windows/Mac OSX/Linux), Action Script, .Net and Java. These references document the API calls that are common to all Phidgets. These API References can be found under www.phidgets.com >> Programming and are listed under the appropriate language. To look at the API calls for a specific Phidget, check its Product Manual.

Code Samples

We have written sample programs to illustrate how the APIs are used.

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.

Go to www.phidgets.com >> Programming to see if there are code samples written for your device. Find the language you want to use and click on the magnifying glass besides "Code Sample". You will get a list of all the devices for which we wrote code samples in that language.

API for the MotorControl LV

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. For exact values, refer to the device specifications.

Functions

int MotorCount() [get] : Constant = 2

Returns the number of Motors that can be controlled by this PhidgetMotorControl.

double Velocity (int MotorIndex) [get,set]

Velocity is the percentage of time the motor is being powered for. The PhidgetMotorControl rapidly switches power to the motor on/off. Velocity can be set between –100 and +100. –100 corresponds to the motor being driven 100% of the time in reverse, +100 driven 100% of the time forward. 0 is off.

double Acceleration (int MotorIndex) [get,set]

Returns how fast a motor will be accelerated between given velocities. The valid range is between AccelerationMax and AccelerationMin. This parameter is currently measured in percent, where 100% is the fastest velocity ramping available.

double AccelerationMax (int MotorIndex) [get] : Constant

Returns the maximum acceleration that a motor will accept, or return.

double AccelerationMin (int MotorIndex) [get] : Constant

Returns the minimum acceleration that a motor will accept, or return.

int InputCount() [get] : Constant = 4

Returns the number of digital inputs.

bool InputState(int InputIndex) [get]

Returns the state of a digital input. True means that the input is activated, and False indicated the default state.

Events

OnMotorChange(int MotorIndex, double Velocity) [event]

An event issued when the velocity a motor is being driven at changes.

OnInputChange(int InputIndex, bool State) [event]

An event issued when the state of a digital input changes.

Technical Section

Brushed DC Motors

A brushed DC motor is a device that converts electricity into mechanical rotation through electromagnetism. Many variations of brushed DC motors exist: permanent magnet motors, electromagnet motors, coreless motors, linear motors... the PhidgetMotorController can be used with any of these, as well as other devices like small solenoids, incandescent light bulbs, and hydraulic or pneumatic devices like small pumps and valves.

Pulse Width Modulation (PWM)

In pulse width modulation, the HIGH and LOW time of a select time period (~400µs in this case) is varied to produce a 0 to 100% adjustment in velocity. For example, at 20% the PWM signal is at a HIGH voltage level for 80µs and at a LOW voltage level (0 volts) for 320µs.

When a PWM signal is passed through a low-pass filter, a constant voltage results. The DC motor, in this case, acts like a low-pass filter and smoothes the PWM signal into an analog voltage. This voltage dictates the speed of the motor: a 10V control signal set to 50% PWM will be filtered at the motor to 5V, and will allow the motor half the total power it would normally see at 100% PWM.

Using the PhidgetMotorControl with a DC Motor

The PhidgetMotorControl has been designed to be used with a variety of DC motors independant of the motor-specific velocity and torque limits. Select a motor that suits your application and falls within the MotorControl device specifications (see page 10). To use a DC motor, first select (in software) which attached motor the PhidgetMotorControl should affect. Velocity (as a percentage out of 100) and acceleration can be controlled for each motor in both directions of rotation. The software will fire an event if a motor attempts to draw too much current (overcurrent condition).

DC Motors

The PhidgetMotorControl LV will work with a variety of low-voltage, low-current brushed DC motors. A few motors are listed below. Note: the PhidgetMotorControl LV will not work with brushless DC motors.

Manufacturer	Part Number	Description
GWS	RC-04	12V 15500RPM 0.15oz-in Brushed DC Motor
GWS	CN12-B2C	4.5V 23700RPM 0.27oz-in Brushed DC Motor
Mabuchi	FF-050SK-1490	6V 15025RPM 0.93oz-in Brushed DC Motor

These and many other brushed DC motors are available directly from the manufacturer or at local distributors.

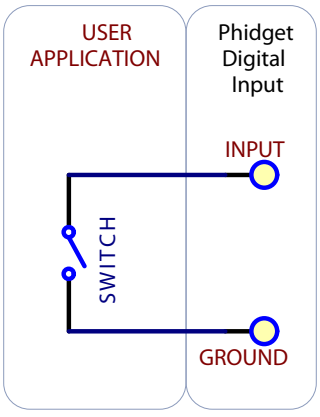
Digital Inputs

Using the Digital Inputs

Here are some circuit diagrams that illustrate how to connect various devices to the digital inputs on your Phidget.

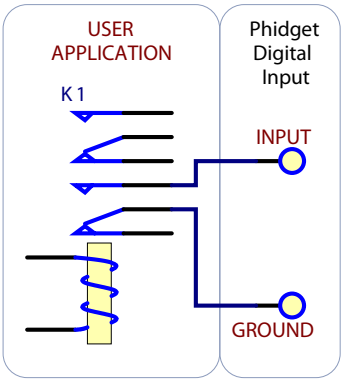
Wiring a switch to a Digital Input

Closing the switch causes the digital input to report TRUE.



Monitoring the position of a relay

The relay contact can be treated as a switch, and wired up similarly. When the relay contact is closed, the Digital Input will report TRUE.



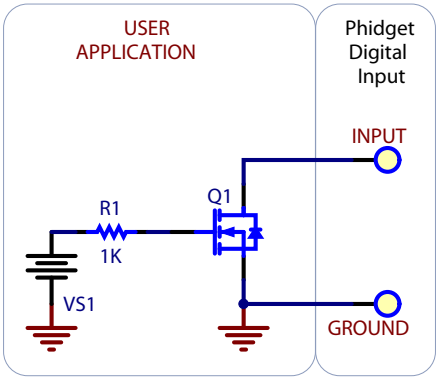
Detecting an external Voltage with an N-Channel MOSFET

A MOSFET can be used to detect the presence of an external voltage. The external voltage will turn on the MOSFET, causing it to short the Digital Input to Ground.

If the MOSFET is conducting > 360µA, the Digital Input is guaranteed to report TRUE.

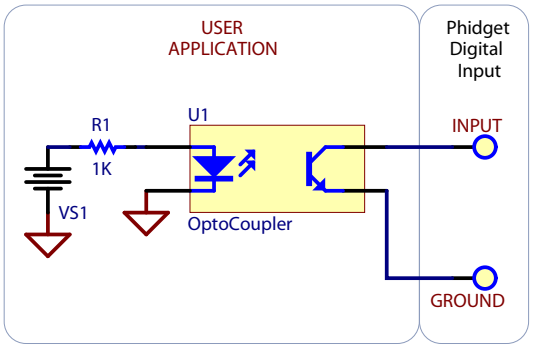
If the MOSFET is conducting < 160µA, the Digital Input is guaranteed to report FALSE.

The voltage level required to turn on the MOSFET depends on the make of of MOSFET you are using. Typical values are 2V-6V.



Isolating a Digital Input with an Optocoupler

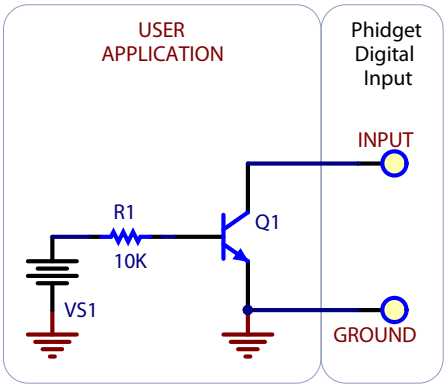
When driving current through the LED, the Digital Input will report TRUE. The amount of current required will depend on the optocoupler used. Design to sink at least 360µA to cause the digital input to report TRUE, and less than 160µA to report FALSE.



Detecting an external Voltage with an NPN Transistor

This circuit can be used to measure if a battery is connected, or if 12V (for example) is on a wire.

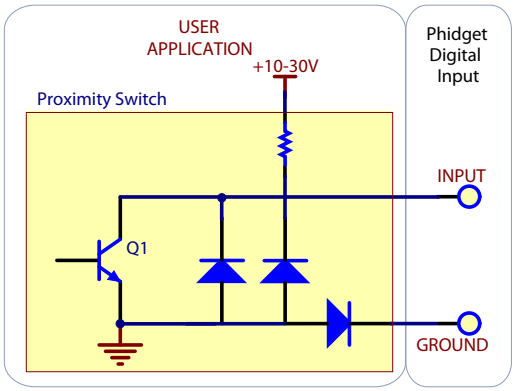
By designing to have Collector-Emitter current > 360µA, the digital input will report TRUE.



Using a Capacitive or Inductive Proximity Switch

Capacitive proximity switches can detect the presence of nearby non-metallic objects, whereas inductive proximity switches can detect only the presence of metallic objects. To properly interface one of these proximity switches to the digital inputs, a 3-wire proximity switch is required, as well as an external power supply.

We have checked the following switches from Automation Direct to verify that they work with the Digital Inputs. Similar capacitive or inductive proximity switches from other manufacturers should work just as well.



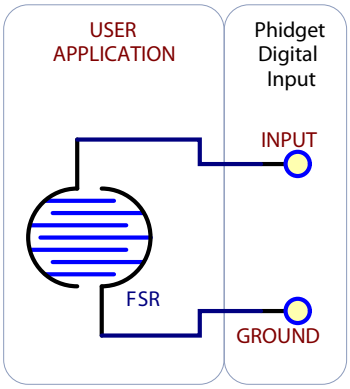
Manufacturer	Web Page	Capacitive Part No	Inductive Part No
Automation Direct	www.automationdirect.com	CT1 Series	AM1 Series

Using an FSR or other variable resistor as a switch

The digital inputs can be easily wired to use many variable resistors as switches.

If the resistance falls below 3.9k Ohms, the Digital Input will go TRUE.

If the resistance rises above 21k Ohms, the Digital Input will go FALSE.



Functional Description

The digital inputs have a built in 10K pull-up resistor. By connecting external circuitry, and forcing the input to Ground, the Digital Input in software will read as TRUE. The default state is FALSE - when you have nothing connected, or your circuitry (switch, etc) is not pulling the input to ground.

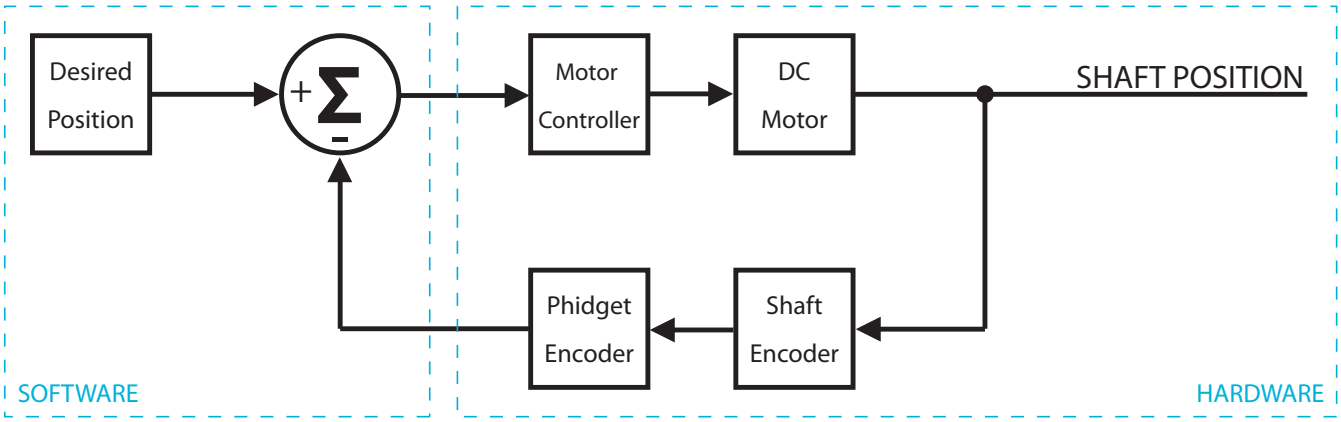
Digital Input Sampling Characteristics

The state of the digital inputs are reported back to the PC periodically. During this sampling period, if a digital input was true for greater than 16ms, the digital input will be reported as true in software.

Advanced Background Information

Using the PhidgetMotorControl to create a simple Position Control System

Unlike a servo motor, a DC motor does not contain the internal electronics necessary to control its shaft position.



However, a control system built from simple hardware and software components can perform this task.

One of the key components that allows an output variable (such as position, velocity, acceleration) to be controlled is a device used to sense its current value. In a position control system, we require an encoder mechanically mounted to the motor shaft to tell us the current shaft position. Once this value has been quantized and is available in software, the rest of the work is in the programming (try using a PhidgetEncoder High Speed for a simple USB interface).

The block diagram above depicts the abstracted operation of a DC motor position control system. The shaft position is **fed back** to the PC through the shaft encoder and PhidgetEncoder. There it is compared with the Desired Position of the shaft (set in software) to produce an error signal which represents the difference between the desired position and the actual shaft position in the real world. The error signal is then applied as the Velocity of the PhidgetMotorController which causes the DC motor to rotate towards the target position. As the shaft nears the desired angle, the error signal moves closer to zero and less power is applied to the motor, causing the shaft to slow and stop at the desired position.

Proportional, Integral and Derivative (PID) Control

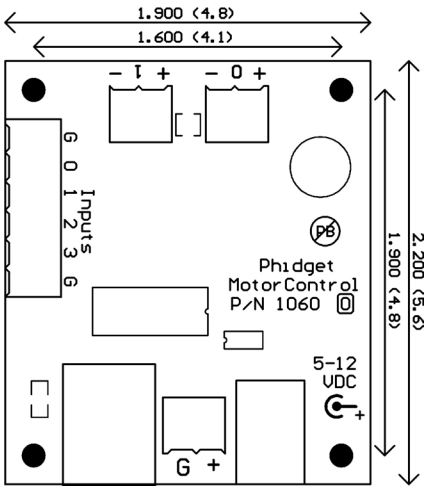
Depending upon the motor and the load it is driving, the shaft may slow too much before reaching the target position and subsequently not arrive there at all. In this case, some amplification or proportional control of the error signal may be necessary. This is accomplished by simply multiplying the error signal by a static value before applying it to the PhidgetMotorController. This will serve to speed up the motor response, but too much amplification may cause the shaft to overshoot the target position or oscillate around it.

If the motor shaft approaches the desired angle at an acceptable speed but there still remains an error above or below the wanted position, what is required is known as integral control. The integral value in software is simply the running sum of all the error signals calculated over time. This value should also be multiplied by a static number to scale it properly, then added to the proportional control before being applied to the PhidgetMotorController.

If the shaft reaches the desired position but overshoots and oscillates around it before returning to the correct angle, use some **derivative control**. The derivative is the difference in value between the current and last calculated error signals (the rate of change), and is used to predict when the shaft position nears its target. Like integral, this value should be scaled then added to the proportional control value before being applied to the PhidgetMotorController.

Mechanical Drawing

1:1 scale



Note: When printing the mechanical drawing, "**Page Scaling**" in the Print panel must be set to "**None**" to avoid re-sizing the image.

Device Specifications

Characteristic	Value
Output Controller Update Rate	30 updates/second
Response Time	30ms
Velocity Resolution	1.5%
Acceleration Resolution	0.1%
Acceleration Limit (-100% to +100% velocity)	250ms
PWM Frequency	2.5kHz
Minimum Power Supply Voltage	5VDC
Maximum Power Supply Voltage	12VDC
Continuous Motor Current	1.5A
Motor Overcurrent Trigger	1.5A
USB-Power Current Specification	100mA max
Device Quiescent Current Consumption	20mA

Note: Current from USB supply is not available for motors.

Product History

Date	Board Revision	Device Version	Comment
November 2005	n/a	100	Product Release
June 2006	0	n/a	Voltage range upgraded to 5 - 12VDC

Support

Call the support desk at 1.403.282.7335 9:00 AM to 5:00 PM Mountain Time (US & Canada) - GMT-07:00

or

E-mail us at: support@phidgets.com