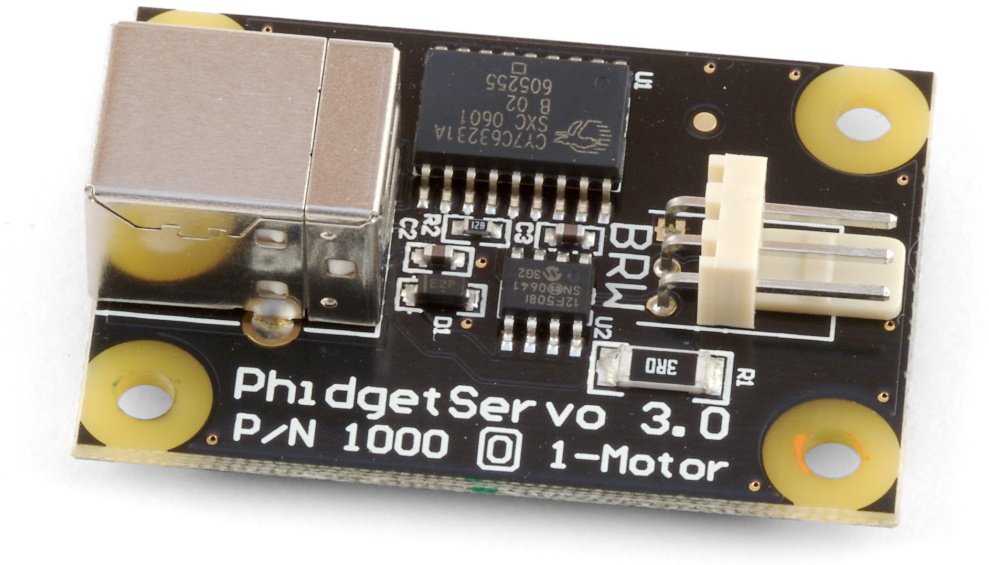
1000 - PhidgetServo 1-Motor



Product Features

- Controls one Remote Control (RC) servo motor powered directly from the USB port of a PC
- Step accuracy of 0.1 degrees
- 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.

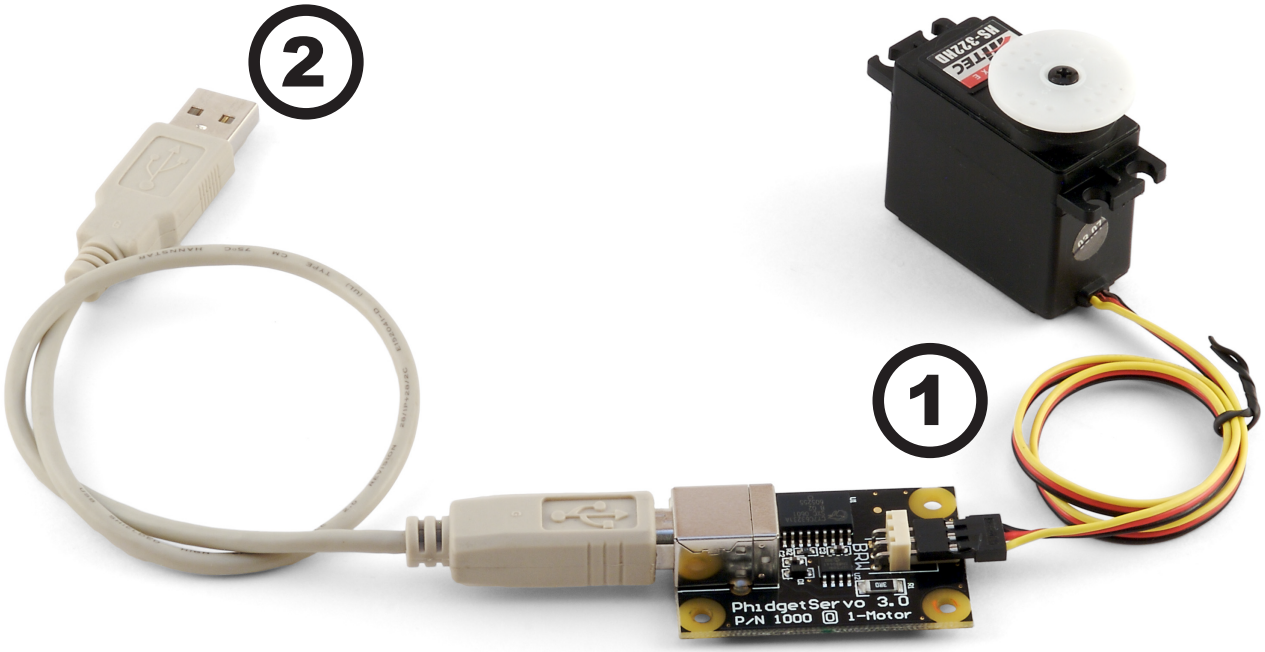
Installing the hardware

The kit contains:

- A PhidgetServo 1-Motor Board
- A USB Cable

You will also need:

- A Servo Motor



1. Attach the connector from the servo motor onto the PhidgetServo board. The board is labeled with B R W (Black Red White) to match the wire colors from servo motors. If you connect it backwards, it will not work! Many servo motors have a yellow wire instead of a white wire.
2. Connect the PhidgetServo board to the computer using the USB cable.

Downloading and Installing the software

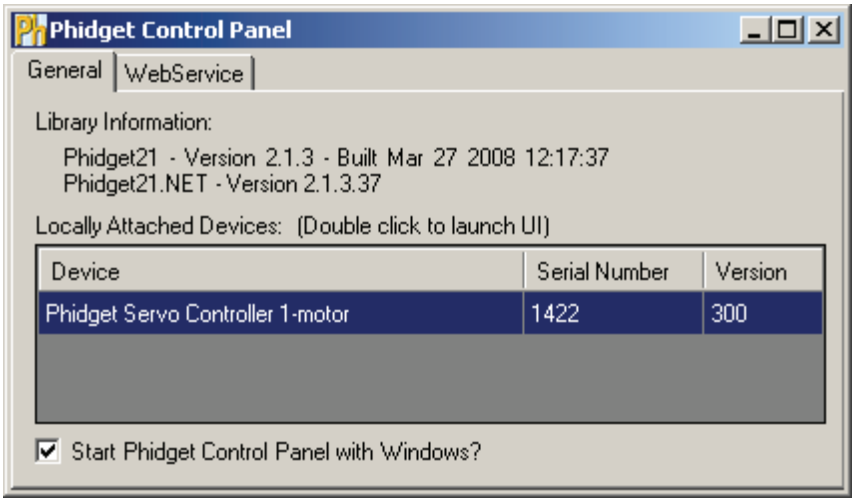
If you are using Windows 2000/XP/Vista

Go to www.phidgets.com >> Drivers

Download and run Phidget21.MSI

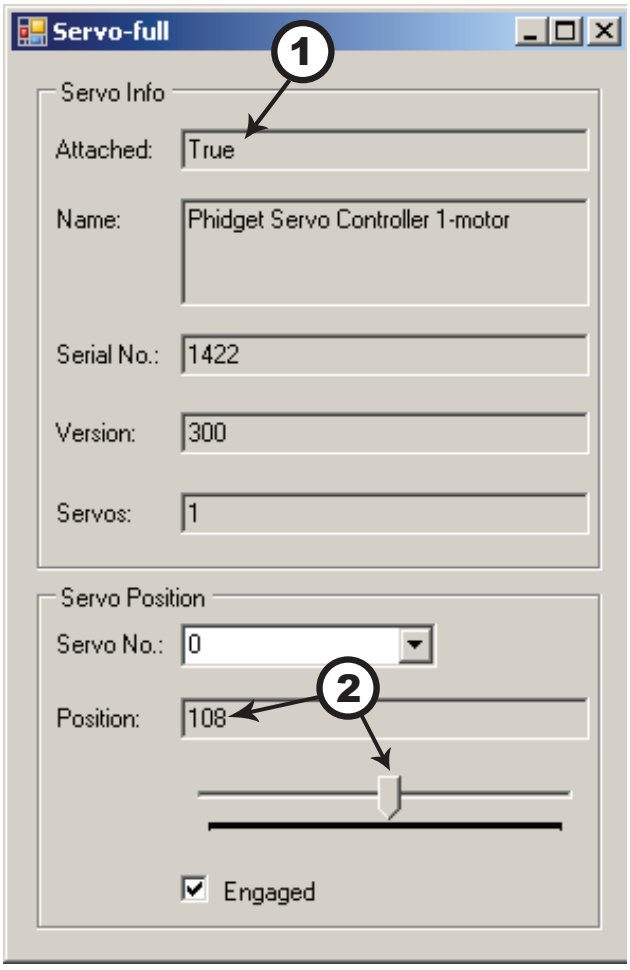
You should see the icon on the right hand corner of the Task Bar.

Testing the PhidgetServo 1-Motor Functionality



Double Click on the icon to activate the Phidget Control Panel and make sure that the ***Phidget Servo Controller 1-Motor*** is properly attached to your PC.

1. Double Click on ***Phidget Servo Controller 1-motor*** in the Phidget Control Panel to bring up Servo-full and check that the box labelled Attached contains the word True.
2. Move the slider to make the motor turn. The motor position is displayed in the box above the Slider.



If you are using Mac OS X

Go to www.phidgets.com >> Drivers

- Download Mac OS X Framework
- Click on System Preferences >> Phidgets (under Other) to activate the Preference Pane.
- Make sure that your Phidget is properly attached.
- Double click on the attached Phidget to launch the Example.

If you are using 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 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.

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.

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 under Programming and are listed under the appropriate language.

API documentation

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 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 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.

Support

- 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 support@phidgets.com

Technical Section

Servo Motors

Servos are motors that are typically used when shaft position needs to be controlled. Internally, a servo motor's shaft is mechanically connected to a potentiometer; this tells the motor's integrated electronics the present position of the shaft. A pulse-code-modulated signal sent from the PhidgetServo on the control wire tells the motor the desired position of the shaft, which is set in software. The motor is then powered and controlled by the integrated electronics until the current and desired positions match.

Pulse Code Modulation (PCM)

A PCM signal has a defined period (typically 20ms in servo applications) and a specified ON- and OFF-time. The ON-time is the amount of time during the period that the signal is at 5V; the rest of the time the signal is at 0V (the OFF-time). When using PCM with servo motors, one specific duration of ON-time will represent the minimum shaft position, and a different and longer duration ON-time will represent the maximum shaft position. The ON-time half-way between these two bounds is the setting where the shaft-position is centered. These values are defined by the motor manufacturer and vary between servo motors.

Using the PhidgetServo with a Servo Motor

The PhidgetServo has been designed to be used with a variety of RC servo motors independant of the motor-specific position, velocity and torque limits. Select a motor that suits your application and falls within the PhidgetServo device specifications (see page 9). To use a servo motor, simply set the desired shaft position in software. It should be noted that the PhidgetServo can not sense the actual position of a servo on its own.

Degree Abstraction

The PhidgetAdvancedServo software component uses degrees to specify position, velocity, and acceleration. The degree unit is translated into a pulse sent to the servo, but it's up to the servo to translate this signal into a particular position. This translation varies between servo models and manufacturers, and it is up to the user to set up their particular servo so that the degree abstraction matches up with reality.

Phidgets Inc. has quantified a number of common servo motors (see API section), which can be used with the ServoType function for to set these parameters. For servos not in this list, or to quantify a specific servo, the setServoParameters function can be used.

Defining a Custom Servo

Servos are driven with a PCM (Pulse Code Modulation) signal. To define a custom servo, you need to find the minimum and maximum PCM values that the servo supports.

The easiest way to do this is by bringing up the example and choosing RAW_us_MODE. This will display all positions in microseconds instead of degrees. Move the servo to both of its extremes, stopping when it hits the stops, then easing up a little (leave a few degrees of leaway), and record these values. You could also choose a specific range in degrees that you require and find the PCM values that correspond. Most servos operate within 500us - 2500us.

Record the degrees of rotation that this PCM range represents (using a protractor, for example).

Feed these three values into the serServoParameters function to complete the set up. This should be done before any other function are called (in the attach event ideally).

Note that many servos can operate quite a bit outside of their rated ranges.

Degree Abstraction (historical model)

Historically, our degree abstraction has been based on the Futaba FP-S148 servo. This is the default abstraction used for the PhidgetAdvancedServo, to maintain backwards compatibility when the new model was added.

Pulse Width (in microseconds) = (Software MotorPosition + 23) * 10.6

Servo Motor Gear Slop

Although the PhidgetServo can position to an accuracy of 0.1 degrees, the repeatability of positioning the shaft of a servo motor is affected by the servo motor's gear slop. Gear slop is the amount of play between the interlocking teeth of gears within the servo motor. More expensive servo motors, built with precision gears, will have a smaller amount of error, while cheaper RC servo motors constructed with plastic gears may have an error of one degree or more.

Using the PhidgetServo with Continuous Rotation Servos

A continuous rotation servo is a servo motor that has had its headgear-stop removed and potentiometer replaced by two matched-value resistors. This has the effect of allowing the motor to rotate freely through a full range of motion, but disables the motor's ability to control shaft position.

When using the PhidgetServo with a servo motor modified in this way, position control in software becomes the motor's speed control. Because the two resistors that replace the motor's potentiometer are matched in value, the motor will always think its shaft is at center position. If the target position in software is set to center, the motor will believe it has achieved the target and will therefore not rotate. The further away from center the target position is set to, the faster the motor will rotate (trying to reach that position, but never doing so). Changing the value above or below center changes the direction of rotation.

Using the PhidgetServo with PCM-to-DC Motor Controllers

Some DC motor controllers accept a servo motor PCM signal as valid input, and use the signal to control the speed of a DC motor. Examples of these include Victor and Thor series motor controllers from IFI Robotics. Operation of these are similar to the way the PhidgetServo is used to control continuous rotation servos, however DC motors with much higher voltage/current ratings can be driven. Note: a buffer on the control line is sometimes required when interfacing to these types of motor controllers, and can typically be purchased from the motor controller manufacturer.

RC Servo Motors

The PhidgetServo 1-Motor will work with a variety of small to medium sized 3-wire servo motors. A few motors are listed below.

Manufacturer	Part Number	Description
Hitec	HS-55	Feather Series RC Servo Motor
Hitec	HS-322HD	Deluxe Series RC Servo Motor (shown)
Hitec	HS-805BB	Mega Quarter Scale RC Servo Motor

The Hitech HS-322HD is available for purchase at www.phidgets.com. Many RC servo motors are available directly from manufacturers like Hitec or at local distributors.

API (Software Technical)

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, please refer to the device specifications.

Structures

enum Phidget_ServoType {

PHIDGET_SERVO_DEFAULT = 1,

PHIDGET_SERVO_RAW_us_MODE,

PHIDGET_SERVO_HITEC_HS322HD,

PHIDGET_SERVO_HITEC_HS5245MG,

PHIDGET_SERVO_HITEC_805BB,

PHIDGET_SERVO_HITEC_HS422,

PHIDGET_SERVO_TOWERPRO_MG90,

PHIDGET_SERVO_HITEC_HS1425CR,

PHIDGET_SERVO_HITEC_HS785HB,

PHIDGET_SERVO_HITEC_HS485HB,

PHIDGET_SERVO_HITEC_HS645MG,

PHIDGET_SERVO_HITEC_HS815BB,

PHIDGET_SERVO_USER_DEFINED

}

Used with the ServoType [get,set] functions. These are servos that have been quantified by Phidget Inc. for your convenience. The Default setting is included for historical reasons, so that the API will be backwards compatible by default. RAW_us_MODE is used for quantifying new servos, or simply when a microsecond based interface makes more sense then a degree based abstraction. USER_DEFINED should never be set directly with ServoType - this is returned when a custom servo type has been defined with setServoParameters.

Functions

int MotorCount() [get] : Constant = 1

Returns the number of servos that can be controlled by this PhidgetServo. Note that there is no way of determining the number of servos actually attached.

double Position(int ServoIndex) [get,set] : Degrees

Sets/returns the desired servo motor position for a particular servo motor.

Note that reading Position will not tell you where the servo really is. RC Servos are open loop – the PhidgetServo can command them to travel to a position, but there is no feedback available for if they arrived, or their position.

If the servo is not engaged, the position is unknown and calling this function will throw an exception.

The range is between PositionMin and PositionMax, and corresponds aproximately to an angle in degrees. Note that most servos will not be able to operate accross this entire range. Typically, the range might be 25 - 180 degrees, but this depends on the servo

double PositionMax(int ServoIndex) [get] : Constant

Returns the maximum position that the PhidgetServo will accept, or return.

double PositionMin(int ServoIndex) [get] : Constant

Returns the minimum position that a PhidgetServo will accept, or return.

bool Engaged(int ServoIndex) [get,set]

If Engaged is set to false, no PWM signals will be sent to the servo. This engages or disengages the servo. The motor is engaged whenever you set a position, or use this function to disengage and reengage without setting a position.

Phidget_ServoType ServoType(int ServoIndex) [get,set]

Gets / Sets the servo type for an index. There is a list of some common servos that have been predefined by Phidgets Inc. This sets the PCM range (range of motion), the PCM to degrees ratios used internally and the maximum velocity. This allows the degree based functions to be accurate for a specific type of servo.

Note that servos are generally not very precise, so two servos of the same type may not behave exactly the same. Specific servo motors, as well as servos not in the list, can be independently quantified by the user and set up with the setServoParameters funtion. This is detailed in the technical section.

void setServoParameters(int ServoIndex, double MinUs, double MaxUs, double Degrees)

Sets the parameters for a custom servo motor. MinUs is the minimum PCM in microseconds, MaxUs is the maximum PCM in microseconds and Degrees is the degrees of rotation represented by the given PCM range.

Quantifying a custom servo motor is detailed in the technical section.

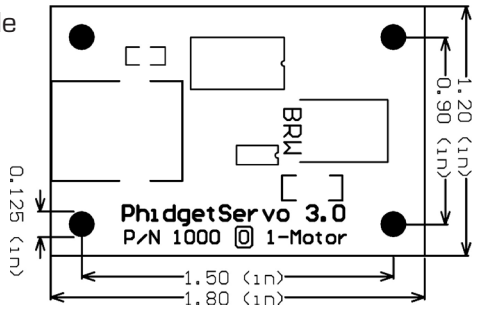
Events

OnPositionChange(int ServoIndex, double MotorPosition) [event]

An event that is issued whenever the position of a PhidgetServo changes.

Mechanical Drawing

1:1 scale



Device Specifications

Pulse Code Period	20 ms
Minimum Pulse Width	10 µs
Maximum Pulse Width	2.55 ms
Time Resolution	1 µs
Output Controller Update Rate	50 updates / second
Output Impedance (control)	600 ohms
Lower Position Limit	- 23.00°
Upper Position Limit	232.99°
Operating Motor Voltage	5.0 V
USB-Power Current Specification	500 mA max
Device Quiescent Current Consumption	13 mA
Device Active Current Consumption	500 mA max

Hitech HS-322HD RC Servo Specifications

Torque @ 4.8V	41.66 oz.in
Speed @ 4.8V	190ms/60°
Size L x W x H	1.57" x 0.78" x 1.43"
Weight	1.51 oz.

Product History

Date	Product Revision	Comment
June 2001	DeviceVersion 200	1 Degree Position Resolution
January 2002	DeviceVersion 300	0.1 Degree Position Resolution
January 2004	DeviceVersion 313	Added State Echoing