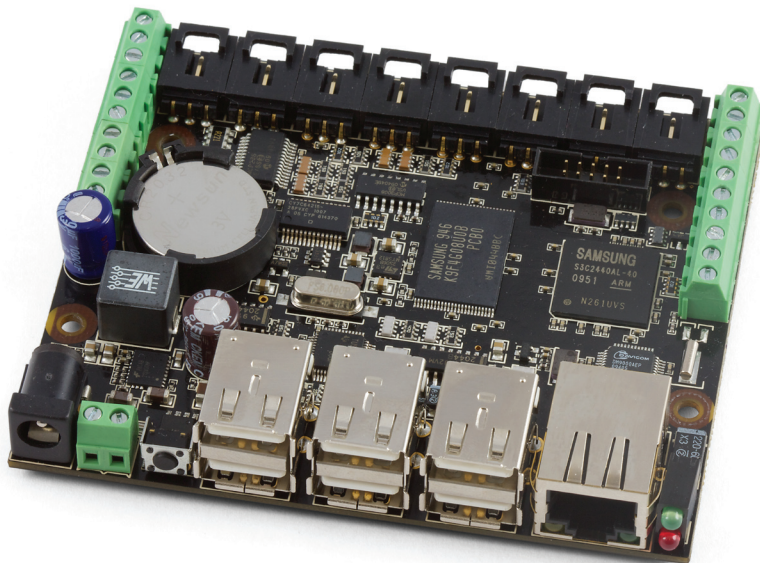


# Product Manual

**1072 - PhidgetSBC2**



**Phidgets 1072 - Product Manual**  
**For Board Revision 0**  
**© Phidgets Inc. 2011**

# Contents

## 6 Introduction

- 6 Overview
- 6 Product Features
  - 6 Computer
  - 6 Connections
  - 7 Integrated InterfaceKit 8/8/8
  - 7 Programming Environment

## 8 Getting Started Guide

- 8 Checking the Contents
- 8 Connecting all the pieces
- 9 Testing Using Windows 2000/XP/Vista
  - 9 Downloading the Phidgets drivers
  - 9 Running Phidgets Sample Program
  - 10 Setting the Password
  - 10 Updating the SBC
  - 11 Testing the Phidget InterfaceKit 8/8/8 Over the Webservice
  - 12 Viewing the Webcam
  - 12 Rebooting/Resetting the PhidgetSBC
- 13 Ports

## 14 User's Guide

- 14 Basic Use
- 14 Phidget Webservice
  - 14 Reliability
  - 15 Finding Phidgets on the Network
- 16 Configuration (Web Interface)
  - 16 The Configuration Pages
  - 17 Status: System
  - 17 Status: Network
  - 17 Status: Processes
  - 17 Status: USB
  - 18 Status: Phidgets
  - 18 Network: Status
  - 18 Network: Settings
  - 18 Network: Wireless
  - 19 Phidgets: Status

19	Phidgets: Webservice
20	Projects: Projects
21	Webcam: Webcam
21	System: General
21	System: Logs
22	System: Password Change
22	System: File Editor
22	System: Backup & Restore
23	System: Packages
23	System: Reboot
23	Main: About

## **24 Networking Guide**

24	Initial Setup
24	Zero Configuration Networking
24	Setting up an Ethernet Connection
24	Setting up a Wireless connection
25	No DHCP Server?

## **26 Advanced User's Guide**

26	Custom Applications
26	Java
27	C/C++
27	Other Languages
27	Debian/Packages
27	Phidget Dictionary
27	SSH/SFTP
28	Custom Kernel and Filesystem
29	Recovery / Upgrade System
29	Entering the Recovery System
29	Upgrades
29	Factory Reset
29	Recovery

## **30 Technical**

30	Power Over Ethernet
30	Hardware Layout
30	Software Layout
30	Date and Time
30	Wireless Networking System

30	Configuration System
31	Nand Layout
31	Boot Process
31	Drivers for USB to Serial adapters
32	U-Boot
32	SBC2 Device Specifications

## **33 PhidgetInterfaceKit 8/8/8**

33	General
33	Product Features
33	Programming Environment
33	Programming a Phidget
33	Architecture
33	Libraries
34	Programming Hints
34	Networking Phidgets
34	Documentation
34	Code Samples
35	API for the InterfaceKit 8/8/8
37	Technical
37	Analog Inputs
40	Digital Inputs
43	Digital Outputs
45	Using the 6-Port USB Hub
46	InterfaceKit 8/8/8 Device Specifications

## **47 Product History**

## **47 Support**

## **47 Legal Information**

# Introduction

---

## Overview

The PhidgetSBC2 is a Single Board Computer with an integrated PhidgetInterfaceKit 8/8/8. At its most basic, it can be thought of as a Phidget that you connect using a network cable instead of USB. The PhidgetSBC2 also provides six full-speed ports that allow you to use normal USB Phidgets over its network connection. This can extend the effective range of a Phidget from USB's maximum of 15 feet, to anywhere that your network reaches.

The PhidgetSBC2 exposes an easy to use interface for setting up and running custom applications on-board. This allows the PhidgetSBC2 to operate autonomously, without the need for a graphical interface or a remote connection at all times.

For more advanced users, the PhidgetSBC is an embedded computer that runs Debian GNU/Linux. We provide full shell access via a built-in SSH server, access to the full Debian package repository, and all of the standard command line tools expected on a modern Linux system.

An integrated PhidgetInterfaceKit 8/8/8 allows you to connect devices to any of 8 analog inputs, 8 digital inputs and 8 digital outputs. It provides a generic, convenient way to interface your PC and PhidgetSBC with a wide variety of devices and it operates exactly the same way as an external PhidgetInterfaceKit.



## Product Features

### Computer

- Single board computer running Debian with access to the full Debian package repository.
- Web based configuration and monitoring interface.
- Abundant on-board storage.
- Run custom programs on-board for autonomous operation.
- Realtime clock with battery backup.

### Connections

- 6 full-speed USB ports with built in support for webcams, Phidget USB devices, and other USB devices.
- Ethernet port. Remotely use attached Phidgets from another computer anywhere over the network.

# Integrated InterfaceKit 8/8/8

The PhidgetInterfaceKit 8/8/8 allows you to connect devices to any of 8 analog inputs, 8 digital inputs and 8 digital outputs.

## Analog inputs

They are used to measure continuous quantities, such as temperature, humidity, position, pressure, etc. Phidgets offers a wide variety of sensors that can be plugged directly into the board using the cable included with the sensor. Here is a partial list of sensors currently available:

IR Distance Sensor	IR Reflective Sensor	Vibration Sensor	Light Sensor
Force Sensor	Humidity Sensor	Temperature Sensor	Magnetic Sensor
Rotation Sensor	Voltage Divider	Touch Sensor	Motion Sensor
Mini Joy-Stick	Pressure Sensor	Voltage Sensor	Current Sensor
Slide Sensor			

## Digital Inputs

Digital Inputs can be used to convey the state of push buttons, limit switches, relays, logic levels, etc...

## Digital Outputs

Digital Outputs can be used to drive LEDs, solid state relays (have a look at our SSR board), transistors; in fact, anything that will accept a CMOS signal.

Digital outputs can be used to control devices that accept a +5V control signal.

With transistors and some electronics experience, other devices can be controlled, such as buzzers, lights, larger LEDs, relays.

# Programming Environment

**Operating System:** Custom Linux Distro, built using Buildroot

**Programming Languages (APIs):** C/C++, Java

**Examples:** Many example applications for all the operating systems and development environments above are available for download at [www.phidgets.com](http://www.phidgets.com).

**Note:** An internet browser is required to use the configuration GUI.

When controlling the PhidgetSBC remotely, you can use any Phidgets supported operating systems and languages:

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

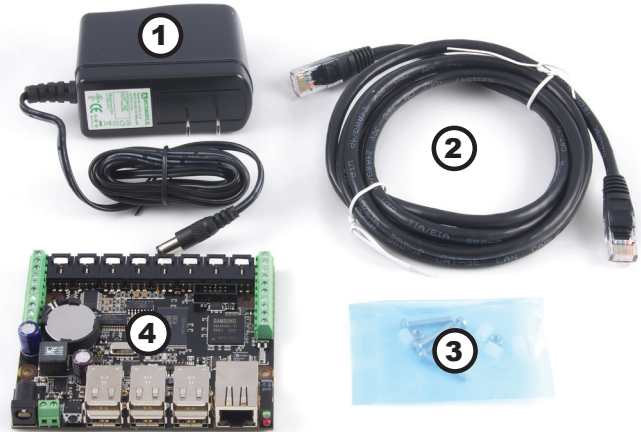
**Programming Languages (APIs):** VB6, VB.NET, C#.NET, C++, Flash 9, Flex, Java, LabVIEW, Python, Max/MSP, and Cocoa.

# Getting Started Guide

## Checking the Contents

### You should have received:

1. A Power Supply
2. A Cat-5e network cable
3. A Mounting kit (4 nuts & bolts, 4 plastic spacers)
4. A PhidgetSBC2 Board

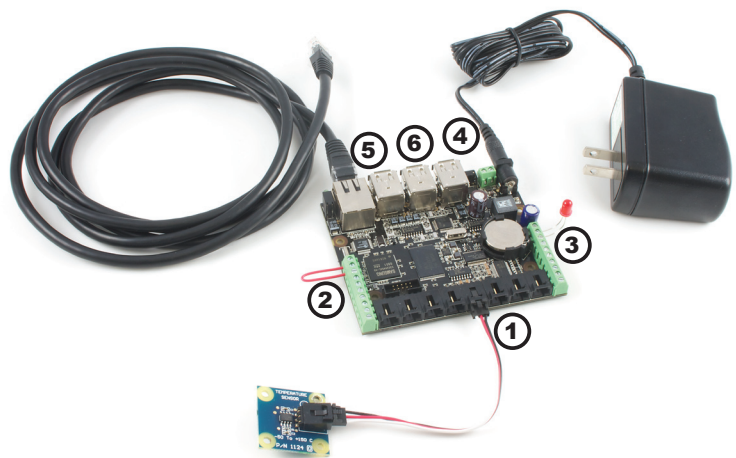


### To test your new PhidgetSBC, you will also need:

- A short length of wire to test the digital inputs
- An LED to test the digital outputs
- An Analog Sensor to test the analog inputs
- A UVC compatible Webcam

## Connecting all the pieces

1. Connect the analog sensor to the analog input port 4 using a Phidgets sensor cable. The analog input ports are numbered from 0 to 7 starting from the left.
2. Connect one end of a wire to digital input port 0 and the other end to ground (labelled 'G' on the underside of the board).
3. Connect the LED by inserting the long LED wire into the digital output 7 and the shorter wire into Ground.
4. Connect the power supply to the PhidgetSBC2 using the barrel connector.
5. Connect the PhidgetSBC2 to your network with an Ethernet cable.



Plug the wall adapter into an appropriate outlet. The red status indicator light located near the USB ports should be lit if the unit is receiving power. The green LED located above the red LED indicates boot status. The green LED will turn on and off once during boot and then turn back on when everything is running.

6. Other Phidgets can also be connected to the 1072 using a 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](http://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

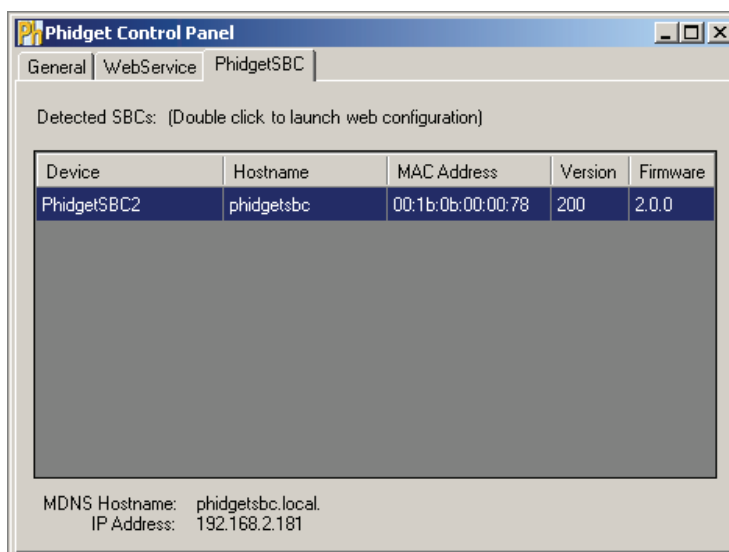
Double clicking on the  icon loads the Phidget Control Panel; we will use this program to make sure that your new Phidget works properly.

Make sure that the **PhidgetSBC2** is powered and properly connected to your network.

Click the PhidgetSBC2 tab in the Phidget Control Panel.

Double click on the PhidgetSBC2 device to bring up the PhidgetSBC2 configuration panel in your default web browser

You can differentiate multiple PhidgetSBC2s by their MAC address, which is printed on the sticker on the back of your board.



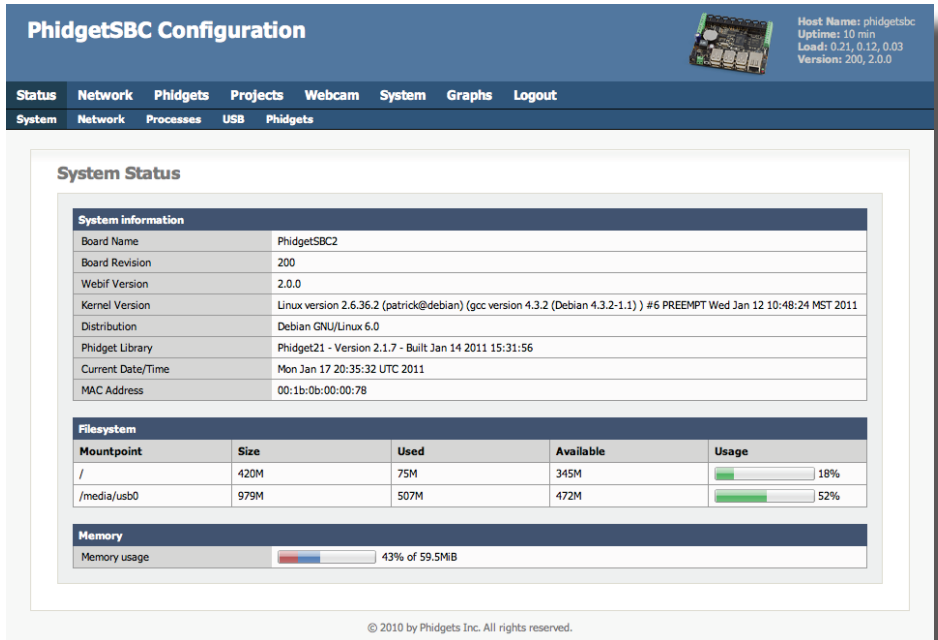
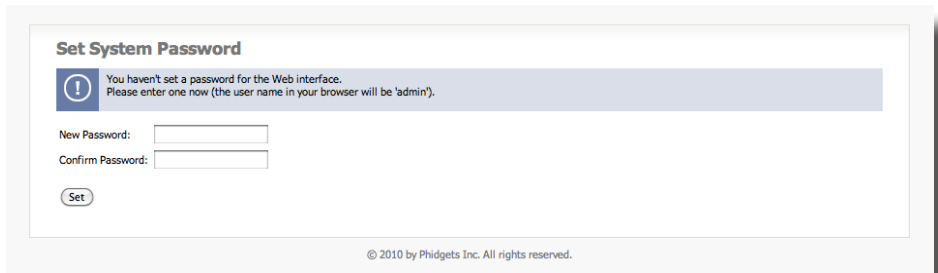
## Setting the Password

On your first visit to the PhidgetSBC web configuration, you will need to set a password.

Type in your password and click on Set.

Subsequent visits will use the username 'admin' and the password you chose.

The PhidgetSBC Info is displayed.

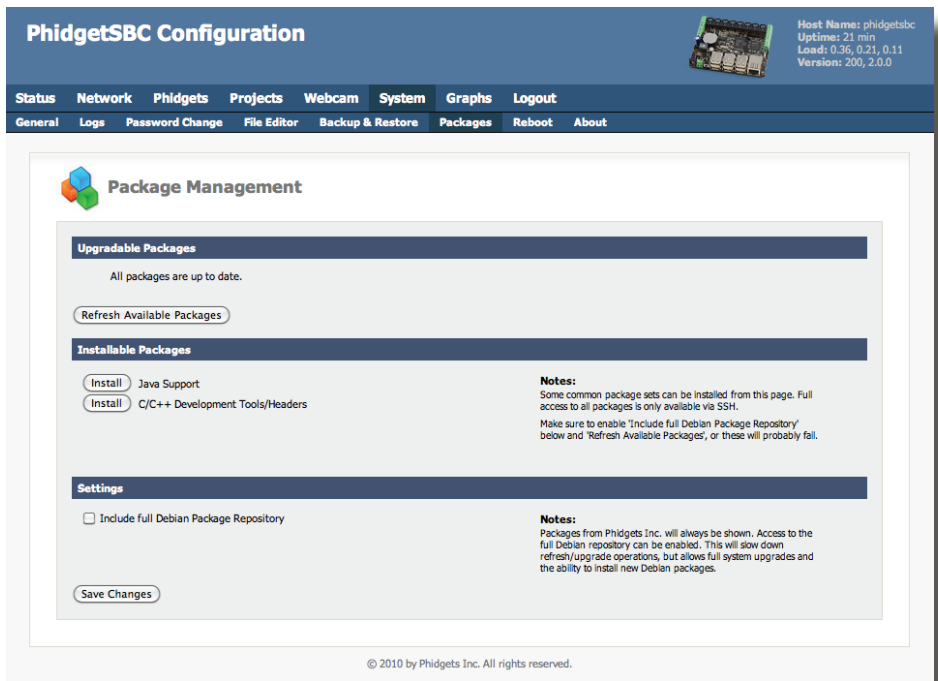


## Updating the SBC

Click on System >> Packages.

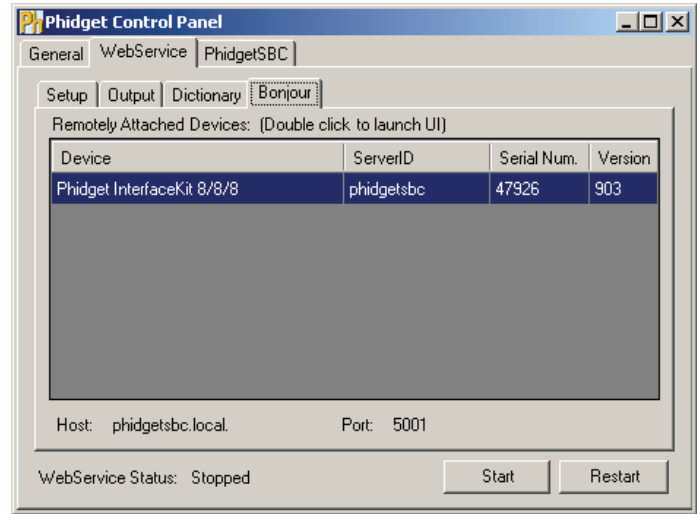
Click "Refresh Available Packages"

Any available updates will be shown. If there are updates available, click "Upgrade All Packages" to download and install them.

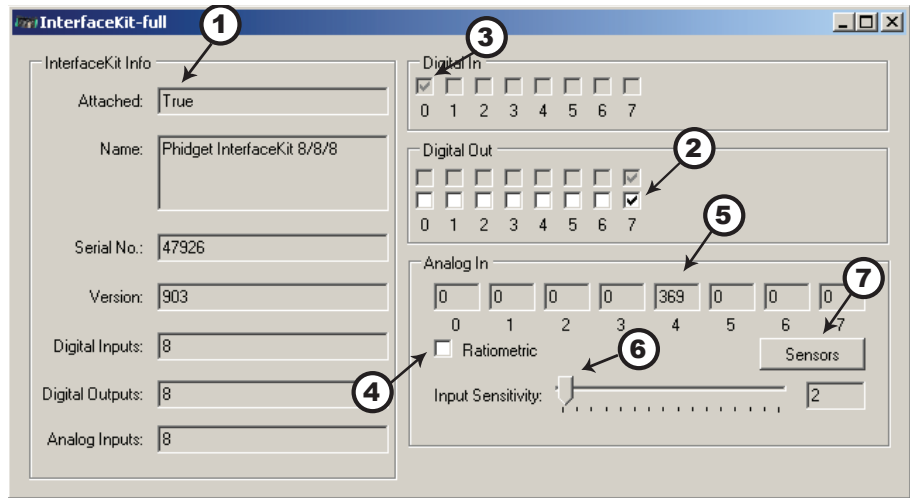


# Testing the Phidget InterfaceKit 8/8/8 Over the Webservice

1. Open the Phidget control panel.
2. Click on the WebService->Bonjour tab.
3. Double click the Phidget InterfaceKit 8/8/8 with the 'phidgetsbc' Server ID to bring up InterfaceKit-full.



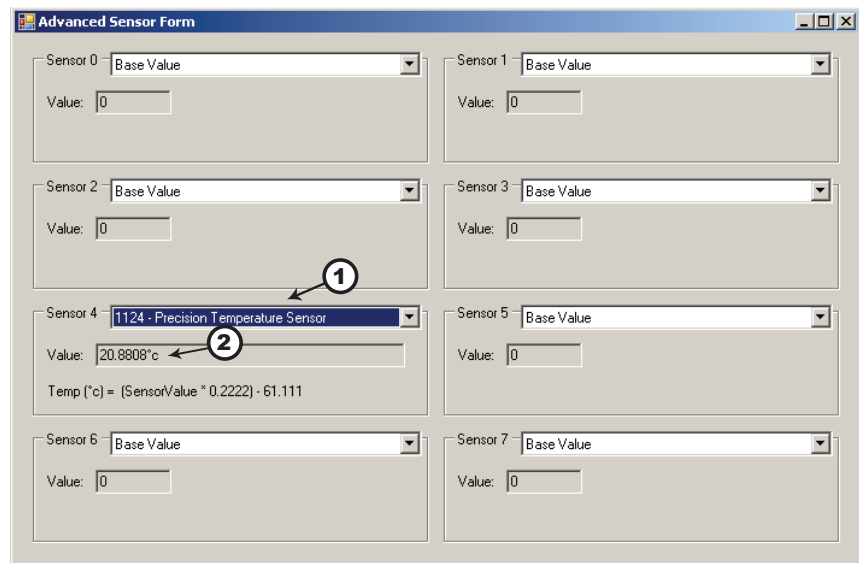
1. Check that the box labelled Attached contains the word True.
2. Test the digital output by clicking on the white box to turn on the LED. Clicking again will turn the LED off. The bottom row shows the status of the request, while the top row displays the status of the digital output as reported by the device.
3. Test the digital input by disconnecting the wire end connected to the digital input connector. The tick mark in the box will go away.



4. Click on the Ratiometric Box if your sensor is ratiometric. Check the sensor product manual if you are not sure.
5. Test the analog input sensor by observing the sensor value as you activate the Phidget sensor.
6. You can adjust the input sensitivity by moving the slider pointer.
7. Click on Sensors to launch the Advanced Sensor Form.

1. In the drop down menu, select the Sensor you have attached to the analog input port 4 of the 1018. In our case we select the 1124 - Precision Temperature Sensor.
2. The ambient temperature sensed by the 1124.
3. Formula used to convert the analog input sensorval into temperature.

**Note:** Value and formula information will vary from sensor to sensor.



## Viewing the Webcam

1. Connect a UVC compatible webcam to your PhidgetSBC.
2. Launch the configuration interface.
3. Click the Webcam tab.
4. Under the settings, select 'Enabled' for the webcam, choose your resolution and frame rate, and then click 'Save Changes'.
5. The webcam stream should now be visible.

### PhidgetSBC Configuration

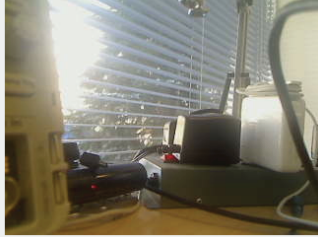
Host Name: phidgetsbc  
Uptime: 19 min  
Load: 0.31, 0.18, 0.09  
Version: 200, 2.0.0

Status Network Phidgets Projects Webcam System Graphs Logout

Webcam

#### Webcam

##### Live Webcam Stream



**Notes:**  
Live streaming should work properly in Firefox and Safari. Internet Explorer users will need to have Java installed. Other browsers may or may not be able to interpret the stream.  
The live stream address is: <http://phidgetsbc.local:81/?action=stream>  
This is an M-JPEG stream that can be viewed/saved by programs such as VLC.  
Webcam video and control is exposed over the specified port (). Various stream formats are available at the webcam webpage at [http://\[phidget\\_sbc\\_ip\]:\[port\]](http://[phidget_sbc_ip]:[port]) (<http://phidgetsbc.local:81>).  
If you are viewing this page through a NAT router (ie. over the internet), the embedded video stream will not work. You will need to make sure that the webcam port is open, and that the router is forwarding it.

**Webcam Control:**  
Click [here](#) to bring up the webcam control dialog.  
This allows you to control pan and tilt on supported cameras, as well as various picture settings such as brightness and contrast.  
If a command fails, this means that it is either not supported by your webcam, or that the setting has reached its minimum or maximum.

##### Webcam Settings

Enabled  Disabled

Resolution: 320x240

Framerate: 25

Port: 81

Password:

**Resolution:**  
All resolutions supported by your Webcam are listed. Because the PhidgetSBC does not have high-speed USB ports, some higher resolutions supported by your webcam may not be shown.

**Framerate:**  
Framerates of up to 30fps can be used with good results, depending on resolution and network bandwidth. Available framerates will depend on the selected resolution.

**Port:**  
The port that the video stream is sent to.

**Password:**  
Protect the webcam stream with a password. This will add a simple username/password prompt whenever you view the webcam stream - including on this page. The username is 'webcam'. Set to nothing to disable passwords.

© 2010 by Phidgets Inc. All rights reserved.

## Rebooting/Resetting the PhidgetSBC

To reboot the device, quickly press the black reset button found between the USB connectors and the power terminals. Both Ethernet Port LEDs (yellow-connectivity, green-activity), and the green status LED will turn off. The reboot is done when all LEDs come back on in about 25 seconds.

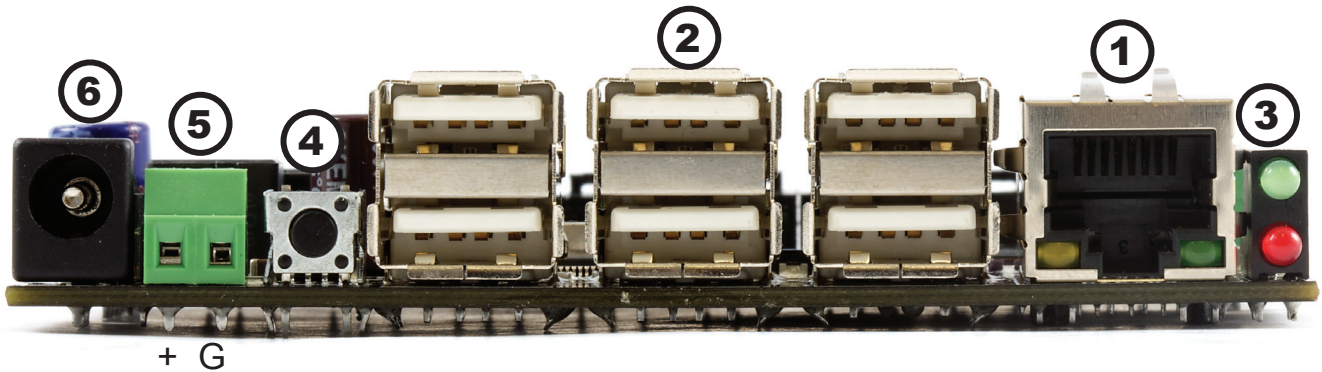
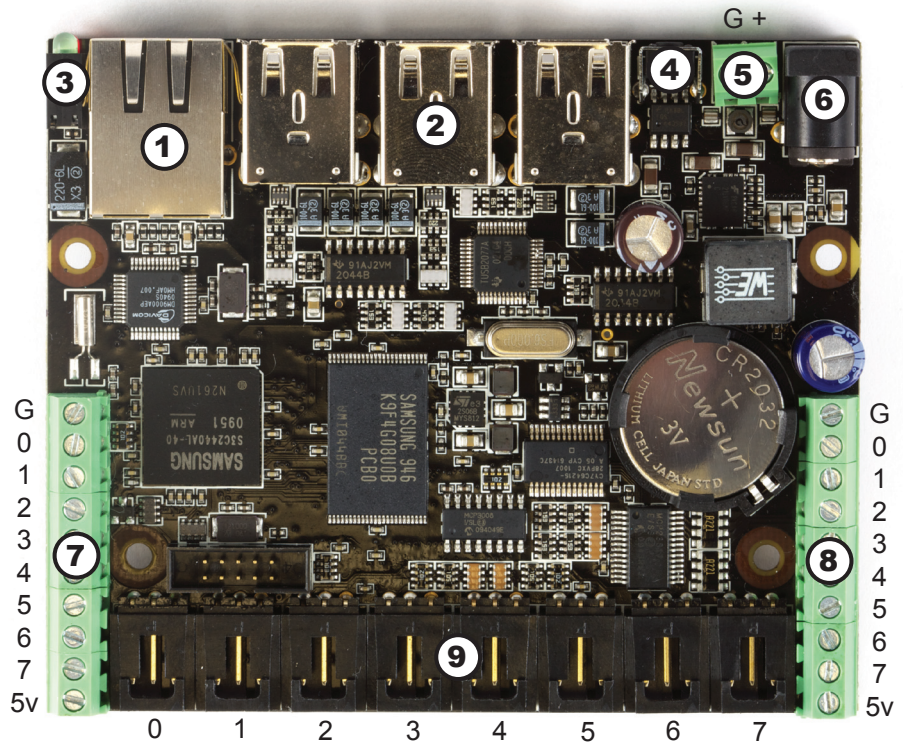
To reset the firmware, press and hold the button for 10 seconds until the green status LED begins to blink, then release. Both Ethernet Port LEDs will turn off (yellow-connectivity, green-activity) for 80 seconds; the green status LED will then turn off; then all LEDs will come back on in 20 seconds. All data will be lost and the operating system will be reset to a factory state.

To boot into the Recovery / Upgrade system, hold the button for 20 seconds until the green status LED switches from blinking slowly to blinking quickly, then release. The recovery system allows for factory reset, full system upgrades and recovery of the main system.

# Ports

Numbered in the circles on the diagram:

1. 10/100baseT Ethernet
2. Six USB Full-Speed Ports
3. Indicator LEDs
4. Reboot / Reset Button
5. Power input terminal
6. Power input jack
7. Eight Interface Kit Digital Inputs (Indexed 0 to 7)
8. Eight Interface Kit Digital Outputs (Indexed 0 to 7)
9. Eight Interface Kit Analog Inputs (Indexed 0 to 7)



1. This Ethernet port is used for network connectivity to the PhidgetSBC. This enables access to the PhidgetSBC as well as any connected Phidgets through the webservice. Alternatively, the USB Wireless adapter can be used for network connectivity.
2. These USB ports can be used for connecting Phidgets, Wi-Fi adapters, flash drives, webcams, USB hubs, etc.
3. These LEDs indicate the status of the PhidgetSBC. The Red LED indicates that the power supply is on and running properly. The green LED indicates boot status. The green LED will turn on and off once during boot and then turn back on when everything is running.
4. This will reboot the board if pressed once. Note that this is a forced reboot. Any user programs that were running may leave their data in a inconsistent state, but this is safe for the base system. A soft reboot can be performed remotely from the configuration interface.

If held for more then 10 seconds, the red LED will start to blink and enter emergency Reset mode. Once the button is released, the onboard memory will revert to a factory-fresh state. This includes overwriting the kernel and root file system, and erasing all configuration, user data, and applications.

If held for more then 20 seconds, the Recovery/Upgrade system will be booted, from which a Factory Reset/Full filesystem upgrade can be performed.

5,6. The PhidgetSBC can be powered from either the terminals or the barrel connector. The polarity of the terminals is also labeled on the underside of the board.

7,8,9. The Interface Kit I/O is explained in the Interface Kit section of the manual.

# User's Guide

---

This guide is intended to provide a look into the basic functionality and configurations that the PhidgetSBC provides. Before continuing, make sure the Phidget21 Libraries are installed as outlined in the Quick Start Guide. Refer to the networking guide in the next section for details on configuring the PhidgetSBC to run on a network. Advanced topics such as working directly with the onboard operating system can be found in the Advanced User's Guide.

The PhidgetSBC consists of an embedded computer combined with an Interface Kit 8/8/8. These elements are essentially separate entities as the Interface kit is connected to the embedded computer using an on-board USB link. The Interface Kit's use will be described in its own section, while the embedded computer will be detailed here.

## Basic Use

Basic use of the PhidgetSBC allows the opening of connected Phidgets over the network. Using another Phidget with the PhidgetSBC in this way is almost exactly like using Phidgets over USB, in respect to the API calls and behavior.

However, some extra considerations need to be made when working with the PhidgetWebservice.

## Phidget Webservice

Support for opening Phidgets over the network is made possible via the Phidget Webservice. This allows a user to write an application in a system and language of their choosing and then operate Phidgets connected to the PhidgetSBC. It is a socket based server that runs on the PhidgetSBC at all times (unless disabled), and allows any attached Phidgets to be seen and opened directly over the network.

Opening and controlling a Phidget over the network is nearly the same as opening one locally. The main differences are:

- Different open calls that include server information. New calls OpenRemote and openRemoteIP (naming depends on language).
- Access to Webservice based properties: Server hostname, port and ID.
- Access to server connect and disconnect events, and network error events.
- Phidgets can be opened by more than one separate application at the same time.
- Reliability is more of a issue because network connections are easily broken.

Opening a Phidget over the network is asynchronous and pervasive, just like opening locally. This means that if a connection to the remote server cannot be established right away, it will keep trying indefinitely, and even survive the server being stopped and started, etc.

Instances of the Phidget Webservice can be referred to either using hostname (IP Address) and port number, or by Server ID. The advantage of using a Server ID is that it stays consistent compared to IP addresses, and you don't need to know the Port number. A Webservice Server ID is assigned when the Webservice is run - which on the PhidgetSBC defaults to 'phidgetsbc'. In order to use a Server ID, the Bonjour utility also needs to be installed.

Refer to the Programming Manual and the API manual for your language for more information about using the Phidget Webservice.

## Reliability

Determining reliability needs can become important while opening Phidgets over the network, because the network connection can potentially be interrupted at any time. This can leave the network attached Phidget in an undesirable state. For example - if a motor controller is driving a motor and the connection is lost, there is no way to stop the motor until the connection is re-established. These issues are less important if you are just receiving sensor data from an Interface Kit.

It's generally a good idea to catch server connect and disconnect and Phidget attach and detach events in order to know the state of the connections. It's also a good idea to catch error events - this is where network errors will be reported.

If reliability is important, you should consider writing a program to run locally on the PhidgetSBC, and communicate with it through the Dictionary interface. This way, if the connection is broken, the local application will notice and be able to take any appropriate actions. See the advanced chapter for more information.

## **Finding Phidgets on the Network**

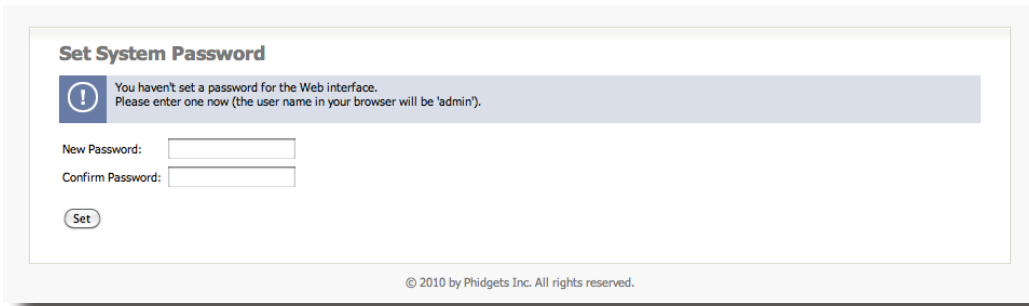
Any Phidgets attached to the PhidgetSBC can be identified using the Status >> Phidgets page in the configuration interface, and should be seen on the network through the Webservice.

The Phidget Control Panel has a Bonjour tab (under Webservice >> Bonjour) that lists all detected network attached Phidgets. The Phidgets connected to the PhidgetSBC should be seen here and can be opened by double clicking its name in the menu.

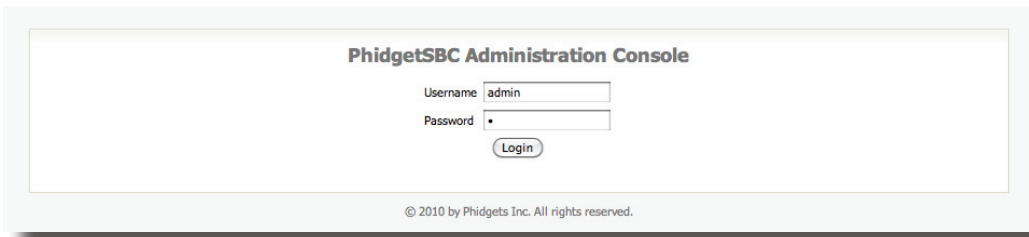
Network attached Phidgets can also be located programmatically with the Phidget Manager. The Phidget Manager is used with either hostname and port, or server ID, just like with 'Open'. The manager can also be used to find all Phidgets on any Webservice through Bonjour, by specifying a NULL Server ID. See your specific language's guide for more information about coding with the Phidget Manager.

# Configuration (Web Interface)

The PhidgetSBC is configured through a built-in configuration interface, through an internet browser much like your wireless access point or router. You can double click on the device under the PhidgetSBC tab in the Phidget Control Panel to bring up its configuration interface. Alternatively, if you have Bonjour installed, you can access the PhidgetSBC by name. For example, by default you can use 'http://phidgetsbc.local.'. Once you know the IP address of your PhidgetSBC, you can also just type it into your web browser of choice.



The first time you access the configuration page, the system will prompt you to set a password before you can continue. This is to maintain security, and cannot be left blank. After setting a password, you can log onto the configuration page with user name 'admin' and the password you chose.



From here you have the option to view the system information and status, configure network settings, start the ssh server, set up custom applications and manage files, or view connected webcams. A breakdown of each function is provided in this manual.

At this point you may also want to upgrade the system and then configure the network settings if the default settings are not appropriate.

## The Configuration Pages

On loading the interface, you will see a tool bar along the top of the page. It holds some information across all the configuration pages. The information is as follows:

**Host Name** - The host name given to the PhidgetSBC on the network.

**Uptime** - Total time elapsed since the last reboot.

**Load** - The average CPU utilization in the last minute, 5 minute, and 10 minute durations.

**Version** - The current board and web interface version.



## Status: System

This is the first page you should see after loading the configuration Interface. It contains general information about the SBC.

### System Information

**Board Name** - Name of the device. It should always read "PhidgetSBC2".

**Board Revision** - Board revision number. This tracks the hardware design.

**Webif Version** - The version of the web interface currently being used. This will change with updates to the web interface/configuration system.

**Kernel Version** - The type and version of the loaded Linux kernel.

**Distribution** - The running Linux distribution name/version. This should read "Debian GNU/Linux 6.0"

**Phidget Library** - The version of the installed Phidget21 library. These libraries are included with the firmware, and may need to be updated to use newly released Phidgets.

**Current Date/Time** - Current date and time.

**MAC Address** - A PhidgetSBC is uniquely identified by its MAC address shown here. This address is also printed on the label of the underside of the PhidgetSBC. Other Phidgets, including the integrated InterfaceKit, use a serial number to identify themselves.

### Filesystem

All mounted filesystems are listed, along with their size and usage.

### Memory

Memory usage is shown. Wired/active memory is shown in red and cached/inactive memory is shown in blue.

## Status: Network

See: Network: Status.

## Status: Processes

This lists all running processes, along with their Process ID (PID), User, State, CPU usage and memory usage. Advanced users can use this to tell if any application is using too much memory, or has crashed.

## Status: USB

This lists all USB devices. The "S3C24XX OHCI" Host Controller, the "General Purpose USB Hub" and the built in Interface Kit 8/8/8 should always be listed, along with any connected devices. Also listed are any mounted USB drives.

### All connected devices

A list of all the USB devices present in the system. This includes the main USB, the built in 6 port hub, and all Phidget and non-Phidget devices.

### Mounted USB / SCSI devices

This area lists of all the USB based drives connected to the PhidgetSBC, and their mount points. USB drives are automatically mounted at /media/usb(0-9) when attached.

**Unmount** - Use this button before removing the device to safely disconnect it.

## Status: Phidgets

see: Phidgets: Status.

## Network: Status

General network status can be viewed on this page. Modifying these values are done on other pages.

**Adapter** - Abbreviated name and number of the network interface.

**Type** - Wired or wireless connection.

**Mode** - Network protocol used.

**IP Address** - The IP address of the network interface.

**Subnet Mask** - The subnet mask of the network interface.

**Gateway** - The IP address of your gateway.

**MAC Address** - The MAC address of the interface.

**Wireless State** - Connection status information for the Wireless link. This could be 'CONNECTED', 'INACTIVE', 'FAILED', etc.

**Wireless SSID** - The plaintext name of the wireless connection access point.

**Wireless Security** - Security protocol used for a wireless link.

**DNS Server(s)** - List of nameservers being used.

## Network: Settings

This is where TCP/IP settings for the wired ethernet are configured. This is also where SSH is enabled.

### Network Settings

**TCP/IP settings** - DHCP will set the system IP Address, Subnet Mask, and Gateway automatically. In the absence of a DHCP server, Static should be used and filled in manually. Note that the same TCP/IP settings will be used at all access points.

**DNS settings** - DNS can be set up automatically if DHCP is enabled. Under manual settings, up to two DNS servers can be specified. Note that DNS settings are system-wide and will apply to all interfaces.

**SSH Server** - This is where the SSH server can be enabled or disabled. Enabling SSH for the first time can take several minutes as the keys are generated.

## Network: Wireless

Wireless networking is supported via a USB wifi adapter. When an adapter is plugged in, this wireless configuration page will be available.

Wireless networks are joined based on a list of saved networks. You can join, manually enable and disable, as well as delete these saved networks. To add a wireless network to this list, either choose from the list of detected networks, or enter the details manually. Supported security includes WEP, WPA(2) Personal and WPA(2) Enterprise. Saved networks will be joined first based on security and secondly based on best signal strength.

### Add a Wireless Network

**SSID** - The SSID of the access point that you wish to add. This is the plaintext name of the access point.

**Security** - The security system used by this access point.

**Remember this network** - If enabled, this network will be added to the list of saved networks permanently, and will be available to be automatically joined in the future. Otherwise, this network will remain in the list of saved networks until the board is reset, or another network is added.

## Manage saved networks

[Join This Network](#) - Joining a specific network will temporarily disable all other saved networks, so that the specific network will be joined, if available. The other networks will remain disabled until the board is reset, or another network is added.

[Delete This Network](#) - Delete a saved network. There is no confirmation and this cannot be undone.

[Enable / Disable](#) - Selected networks that are enabled will be joined automatically. Disabled networks will never be joined.

## Wireless Network Settings

[TCP/IP settings](#) - DHCP will set the system IP Address, Subnet Mask, and Gateway automatically. In the absence of a DHCP server, Static should be used and filled in manually. Note that the same TCP/IP settings will be used at all access points.

[DNS settings](#) - Switch the DNS settings between Automatic and Manual. DNS can be set up automatically if DHCP is enabled. Otherwise, up to two DNS servers can be specified. Note that DNS settings are system-wide and will apply to all interfaces.

## Phidgets: Status

### Library Version

The version of the installed Phidget21 library. These libraries are included and are updated along with the firmware.

### List of attached Phidgets

A list of all detected Phidgets connected to the PhidgetSBC. This includes the integrated PhidgetInterfaceKit and displays both the serial number and version.

## Phidgets: Webservice

The Phidget webservice is a simple server that allows Phidgets connected to the PhidgetSBC board to be opened over the network. This is enabled by default and starts with the SBC. The webservice also exposes a key-value dictionary which can optionally be used for communication between applications written for the PhidgetSBC, and applications running on your local computer. This page lets you view and modify the Phidget webservice settings, as well as stop and start the server. Please see the Advanced User's Guide for more information on programming with the dictionary.

### Phidget Webservice

[Enabled/Disabled](#) - Enables or disables the Phidget Webservice.

[Server ID](#) - Server ID is used when opening a connection to the PhidgetSBC using the mDNS based openRemote calls. This is by default the same as the PhidgetSBC hostname (phidgetsbc), but can be set to anything (up to 63 characters).

[Port](#) - Port is the port that the webservice runs on - default is 5001.

[Password](#) - The password is used for securing the webservice. By default, this option is disabled with a blank password. Note that while the authentication protocol and password is encrypted during authentication, all following data is sent in the clear.

[Start/Stop](#) - Use this button to start/stop the webservice.

## Projects: Projects

This is where user projects are set up. Custom applications can be written in either C or Java, and then set up to run on the PhidgetSBC at system startup. On the main page, there is a list of installed applications as well as the controls for creating a new application space. Application names should not contain spaces.

On a specific application space page, there are controls to start and stop the program, as well as view the stdout and stderr from the most recent (or current) run.

There is a filesystem browser, which allows viewing, editing and removal of application files, as well as the ability to upload new files. File upload size is limited to 5MiB per file.

The application settings section configures the application. When an application is enabled, it will start at the end of the system boot process (after things like bringing up the network, starting the Phidget Webservice, etc.). The startup order field specifies a start order among the custom applications, with lower numbers being started first. The Run as Daemon check box ensures that the application is run as a daemon. This should only be unchecked if the application daemonizes itself, or quits right away - otherwise it will stall the boot process. Executable name is the name of the file to execute. If this filename ends in `.class` or `.jar`, the program will be run as a Java program, otherwise it is run as an ARM Binary.

See the chapter on Advanced use for more information about setting up custom applications in Java and C.

## User Applications

[Currently installed applications](#) - This is a list of all created applications and their current status. Enabled applications will try to run on system boot, and the stopped/running status indicates if the program is currently executing. Delete applications using the red 'X' near their name. You can click on the application name to launch the application page.

[Create new app](#) - This button creates a new application space using the input field for its name.

[Free space remaining on userspace partition](#) - The amount of free space remaining on the user partition in bytes.

## Application page

[Start/Stop](#) - This button is for starting or stopping the execution of the program specified under application settings. Starting a program will generate stdout and stderr logs.

[View stdout](#) - You can view the standard console output of your program through this link.

[View stderr](#) - In the event of an error that halts program execution, its corresponding error message is printed here.

## Filesystem Browser

[Upload a file](#) - Use Browse to select the file from your PC, or type the file path of the item to be uploaded. The Upload button will then copy the selected file to the current open folder in the Filesystem Browser.

[Create a directory](#) - Type the directory name you wish to create in the field. Click Create to make the directory in the current open folder.

[Free space remaining on userspace partition](#) - The amount of free space remaining on the user partition in bytes.

## Application Settings

[Enabled/Disabled](#) - Enabled applications will start automatically when the PhidgetSBC is booted. Disabled applications can be started manually by the user.

[Startup order](#) - Use to set the startup order when multiple applications are defined. Lower numbers get started first.

[Run as daemon](#) - Runs the application as a daemon. This should only be disabled if the application daemonizes itself - otherwise the PhidgetSBC startup process will hang.

[Executable/Class name](#) - Name of the program file to execute. This file must exist (have been uploaded) before it gets defined here. Files ending in `.class` and `.jar` will be run through the Java VM, otherwise they are executed directly.

[Arguments](#) - Command line argument list to pass to the program on execution.

## Webcam: Webcam

This is where you can view and control a connected UVC (USB Video Class) webcam. For a list of UVC compliant webcams see here: <http://linux-uvc.berlios.de/#devices>

Video is streamed from these devices in M-JPEG format, and can be viewed through the web interface, or any compatible M-JPEG viewer (such as VLC). Webcams that support pan/tilt can be controlled via the web interface. Multiple frame rates and resolutions are supported. To get started, plug in the webcam and the interface will be initialized.

If the video stream is to be exported over the internet, it is recommended that the password be enabled. However, it must be noted that this is a simple HTTP authentication, which is sent unencrypted, and is thus not highly secure.

If prompted for the webcam password, the username is 'webcam'.

If multiple webcams are attached, they will all start up using the same settings, except that each additional webcam will run on the next higher port number. Multiple webcams will generally only work when the resolution and framerate are set to low levels.

### Webcam Settings

**Enabled/Disabled** - Enable or disable the webcam streaming video. Video streaming can consume a lot of bandwidth depending on the settings used.

**Resolution** - The resolution of the capture in pixels. Only resolutions supported by the webcam are listed.

**Framerate** - The transmission frame rate of the capture. Available frame rates will depend on the selected resolution.

**Port** - The port that the video stream is sent to.

**Password** - Protect the webcam stream with a password. This will add a simple username and password prompt whenever you view the webcam stream - including on this page. The username is 'webcam'. Set to nothing to disable passwords.

## System: General

This is where general system settings are set up.

### System Settings

**Host name** - The system hostname. This is used for the system's mDNS hostname, as well as the Phidget Webservice default Server ID. All PhidgetSBCs have a default hostname of 'phidgetsbc'.

### Time Settings

**Timezone** - Set up your time zone according to the nearest city of your region from the predefined list.

**Zoneinfo String** - Standard zoneinfo names are defined for different areas of the world. You can supply any zone from the official list as an alternative to selecting a predefined zone from the list.

## System: Logs

This is where the kernel and system logs can be viewed. This also includes the ability to filter the text.

### Text Filter



[Text to Filter](#) - Insert a string that covers what you would like to see or exclude. In fact you can use the regular expression constants like: 00:[[:digit:]]{2}:[[:digit:]]{2} or .debug|.err.

[Filter Mode](#) - You will see only messages containing the text in the Include mode while you will not see them in the Exclude mode.

[Remove Filter](#) - Clears the filter being used

[Filter Messages](#) - Change the filter being used. Including a blank Text to Filter effectively removes the filter.

## System: Password Change

This is where the system password can be changed. The system password is the 'root' user password, used for logging into this web interface, as well as logging in as root via SSH. Changes here made will take effect immediately after being saved, without asking for confirmation. The new password must consist of alphanumeric characters and be at least 1 character long.

### Password Change

[New Password](#) - The first field for a new password.

[Confirm Password](#) - The second field for a new password. This must match the first field.

[Set Password](#) - This button will commit the changes to your password.

## System: File Editor

This grants access to the full filesystem. Files can be uploaded, edited and deleted, and directories can be created. Some important system directories and files are protected from deletion, but it is still easy to break the system by editing/deleting files.

File upload is limited to a filesize of 5MiB.

File/Directory permission, owner/group and creation time information is available by hovering over the file/directory icon.

## System: Backup & Restore

This is where the system can be backed up to disk and restored. This also allows access to the recovery/upgrade system.

Note that this only backs up configuration files directly related to the web interface - such as network configuration, hostname, time settings, and user project setting (but not including project files).

When restoring from a backup file, the system will check that it is a valid backup before asking the user to continue.

### Backup Configuration

[Name this configuration](#) - You can give the backup file an arbitrary name. The name is only shown during restore.

[Backup](#) - This button creates the backup. A download link to the backup will be provided and you will be prompted to save the file to a location. The download link is not valid indefinitely.

### Restore Configuration

[Saved backup file](#) - Choose a backup file for this machine type. The restore system will check the file and ask for confirmation before running the restore.

[Restore](#) - This button applies the backup. The backup file will then be verified. Clicking restore again will commit the changes and take effect immediately.

### Reset Configuration

[Reset](#) - This will reset the web interface configuration files to their default states. This will not reset the system as a whole.

## Upgrade / Factory Reset

[Go](#) - This will boot the SBC into the Recovery/Upgrade system. See the Recovery/Upgrade section below for more information.

## System: Packages

This is where package updates are performed. From time to time, updates will be made available. These will address bugs and security issues, add new features, and update the phidget21 library and webservice.

There are also some other package management options available.

Note that this page can take quite a long time to load when "Include full Debian Package Repository" is enabled.

### Upgradable Packages

This lists all available package updates, if any.

[Upgrade All Packages](#) - This installs all available package updates. Selective updating can only be done via SSH.

[Refresh Available Packages](#) - This runs 'apt-get update' to refresh the package list and check for new updates.

### Installable Packages

This lists some package sets that be installed to add common features to the SBC. Before installing these package sets, enable "Include full Debian Package Repository", and "Refresh available packages" - otherwise the needed packages won't be available and the install will fail.

[Java Support](#) - This installs 'libphidget21-java' and 'default-jre-headless' to support Java based projects.

[C/C++ Development Tools/Headers](#) - This installs 'build-essential' which allows for on-board development in C/C++.

### Settings

[Include full Debian Package Repository](#) - This enables the full Debian repository in the apt sources list. By default, only the Phidgets repository is included. When enabled, all system packages will show up in the updates list, rather than just Phidgets packages, allowing for full system updates. This also allows the user to install any packages they like, from the SSH interface.

## System: Reboot

The board can be rebooted remotely from this page. The reboot should take 45-60 seconds depending on network conditions. It tries to shut down all running programs before restarting as opposed to the reset button.

## Main: About

The license information and credits for the configuration interface is displayed here. A link is provided to the original sources and the Phidgets web site.

# Networking Guide

---

## Initial Setup

1. Make sure that Phidget21 is installed on your computer.
2. Install Bonjour (recommended)
3. Plug the PhidgetSBC into your network and power it up.

The PhidgetSBC will try to get an IP address from DHCP, which should work on most networks. If there isn't a DHCP server running, then it will fall back on Link Local addressing. In order to find the PhidgetSBC on your network, you either need to look at the DHCP server log, or use Bonjour. DHCP servers are provided by most home routers, and by your ISP.

It is typical for the PhidgetSBC to take as long as a minute to appear in the Phidget Control Panel when connected, and longer to be removed from the list after being disconnected.

## Zero Configuration Networking

We recommend that you install the Bonjour utility for simpler networking with the PhidgetSBC through Zeroconfig.

- For Macs, Bonjour is already installed.
- Windows users can get it from: <http://www.apple.com/support/downloads/bonjourforwindows.html>. After installing, a reboot may be required for the Phidget Control Panel to function.
- Linux users should use Avahi, which if not installed will be available as a package.

Bonjour lets you access your PhidgetSBC without knowing its IP address. Multicast DNS (part of Bonjour) will expose each of your PhidgetSBCs using a .local hostname, which you can use anywhere an IP address would be used. The default name for the PhidgetSBC is 'phidgetsbc' - so 'phidgetsbc.local' would be its mDNS hostname.

If you have more than one PhidgetSBC on the network, their host name will appear like phidgetsbc.local., phidgetsbc-1.local., phidgetsbc-2.local., etc. They will also be given the server ID usually similar to the host name as phidgetsbc, phidgetsbc (1), phidgetsbc (2)... etc.

If you use Bonjour, any PhidgetSBCs on the network will show up in the Phidget control panel (windows) or Phidget preference pane (Mac). For Linux users, there is a tool in the examples folder of the library download, which will list PhidgetSBCs. The PhidgetSBC will also show up in the Bonjour tab in the Safari web browser.

You can identify a specific PhidgetSBC by matching the MAC address reported in the Phidget control panel with that printed on the label on the underside of your PhidgetSBC.

## Setting up an Ethernet Connection

Simply plug an Ethernet cable from your network into the Ethernet port of the PhidgetSBC. By default, the PhidgetSBC will set the system IP address, subnet mask, and gateway automatically. If Bonjour or Avahi is installed, then the PhidgetSBC will automatically register itself with mDNS when connected.

## Setting up a Wireless connection

The PhidgetSBC supports wireless networking via the available USB Wireless adapter. The Wireless antenna should be kept in clear space as performance can be dramatically reduced if it is handled, or if it comes into contact with objects. To initially configure the wireless, you must first connect to the PhidgetSBC configuration interface through Ethernet.



If the wireless adapter is plugged in and detected, then you can configure it under the Network : Wireless menu. Use the Re-Scan button to search for networks and then select your network from the list. Next input any required user names or passwords and click add this network. The newly saved network should appear under Manage Saved Networks. Now under Wireless Network Settings, make sure TCP/IP settings are set to DHCP and the DNS settings are on Automatic. Once these steps are completed, then you can switch to the wireless connection by disconnecting the Ethernet cable.

If you have both a Wireless connection and an Ethernet connection configured, you can freely switch between the two by disconnecting one or the other.

## No DHCP Server?

If you don't have a DHCP server on your network, you will need to initially communicate with the PhidgetSBC using link local addressing. It is highly recommended that Bonjour be installed in this case, otherwise it will be very difficult to obtain the PhidgetSBC's IP address.

Mac: Bonjour is installed by default, and link local addresses should work.

Windows: With Bonjour installed, link local addresses should work.

Linux: These routes need to be added (where eth0 is your primary interface):

- `route add -net 169.254.0.0 netmask 255.255.0.0 dev eth0 metric 99`
- `route add default dev eth0 metric 99`

For more information, visit <http://developer.apple.com/qa/qa2004/qa1357.html>

Even with Bonjour installed, you may have to do some special set-up to talk to the device. The main TCP/IP networking options are DHCP and Static. By default, the board operates in DHCP mode, in which case your network needs to be running a DHCP server. Static addressing can be used if necessary, in which case you need to specify IP address, subnet mask and internet gateway addresses appropriate for your network environment.

DNS servers can be selected automatically if DHCP is used for TCP/IP. Otherwise, you need to manually enter your DNS server(s). Note that if the board does not have DNS properly set up, it will not be able to resolve internet hostnames. This will disable the NTP daemon which runs at startup to set the correct date and time.

TCP/IP and DNS settings are system wide, for all network interfaces. The PhidgetSBC does not support different DNS settings for the wireless and the wired interface, or per-access point TCP/IP settings.

# Advanced User's Guide

---

This section describes use of the PhidgetSBC outside of the web configuration, and basic opening of Phidgets over the network. This includes custom applications, SSH, customizing the system, etc. It is recommended that you have some experience with Linux before trying some of these tasks.

## Custom Applications

The PhidgetSBC supports custom user application written in either Java or C. Custom applications are set up and managed using the configuration interface.

Custom applications are created under the `/usr/userapps/(application name)` directory and contain all application files. This must include at least the executable file.

Note that custom applications should not try to get user input, as `stdin` is closed before the application gets run.

If your application writes important information to disk, make sure to call `sync` - otherwise data may be lost in an unexpected reboot.

### Java

First, install 'libphidget21-java' via 'apt-get install', or Install 'Java Support' via the 'System: Packages' web interface page.

Java applications can be compiled on a separate development machine, where they can also be tested before deployment. When coding, make sure to include the correct version of `phidget21.jar` file as part of the environment. You can use the link under Projects: Projects page in the configuration interface to ensure that your Java program is synchronized with the version of `phidget21` on the PhidgetSBC. When using Java packages, make sure to create the appropriate directory for them.

The PhidgetSBC also supports `.jar` files and you may find it easier to compile an upload a `.jar` instead of the all the necessary `.class` files. When your project is completed we recommend to compile the project as a `.jar`. This reduces the number of extra files created into a single package that is easier to manage and can be executed from the command line, or even by double clicking the file if your operating system environment is configured properly.

Under the command line, you can use the `jar` utility from the Java SDK to package the `.class` and `.java` files. The process starts by going to the directory where your program is located and creating a manifest file. This manifest file tells the java jar compiler the version of the program and the name of the Main-Class which acts as the entry point for the application. The entry point is the class that contains the main method that is run when the program is started. You also want to add a line to specify the class-path which will point to the `phidget21.jar` file that contains the library for including into the jar.

Let's assume we want to distribute our program `MyProgram.java` as an executable jar file. First, compile the program to generate the `.class` files. Now, create the manifest file `MyProgram.mf` which contains the following lines:

```
Manifest-Version: 1.0
Class-Path: phidget21.jar
Main-Class: MyProgram
```

**NOTE:** The Manifest file (`MyProgram.mf`) MUST have a blank line at the end.

Save the file and close it. Create the jar by running the following command from the command line:

```
jar -cmf MyProgram.mf MyProgram.jar *.class
```

You should now have an executable jar file called `MyProgram.jar` that you can distribute easily as one package and run from anywhere very easily. To run the executable jar file you can either type the following into the command line:

```
java -jar MyProgram.jar
```

Or, you can double click the file in a visual operating system if your environment is configured properly.

If your program uses phidget21.jar, you must include it on the Classpath. On the SBC this looks like:

```
java -cp ./usr/share/java/phidget21.jar -jar MyProgram.jar
```

**Note:** Some IDEs, such as Eclipse and Netbeans, automatically create jar files when you build your project. Simply look in your build output folders for your .jar file.

## C/C++

First, install 'libphidget21-dev' and 'build-essential' via 'apt-get install', or Install 'C/C++ Development Tools/Headers' via the 'System: Packages' web interface page.

C programs can be compiled on the PhidgetSBC itself, via the SSH interface, or off-board using a cross compiler. Use of a cross compiler is not strictly documented here.

If you need to log data from a custom application, you can either log directly to the application directory with the size limits of the userspace in mind, or to /tmp if the data should be erased on reboot. Alternatively, you can use a flash drive, which are mounted automatically at /media/usb(0-9) when plugged in.

## Other Languages

Other languages, such as Python, Mono, etc. are easy to support by installing the correct packages. These are not however explicitly supported/documented by Phidgets currently.

## Debian/Packages

The PhidgetSBC runs Debian. We have included packages from the Emdebian project where available in order to reduce space requirements. This is hidden from the end user.

Full access to the Debian package repository is available. In order to enable access to all packages, enable 'Include full Debian Package Repository' on the 'System: Packages' page. Editing the Apt sources list files by hand is not recommended, as these may be overwritten during boot.

For an overview of apt-get and dpkg, see the Debian documentation.

## Phidget Dictionary

Communication between a custom application on the PhidgetSBC and the outside world can be facilitated by using the dictionary interface of the Phidget Webservice. The dictionary lets you set and listen for key/value pairs over the network, and take action accordingly. This could be used to post data or listen for commands over the network, while maintaining reliability and ultimate control on the PhidgetSBC itself in case of network failure.

See the Phidget Programming manual for its use in your language of choice.

## SSH/SFTP

The built-in SSH Server can be enabled to allow console access to the PhidgetSBC. By default, this server is disabled. SSH access to the PhidgetSBC is enabled in the Network: Settings configuration page on the PhidgetSBC. Enabling the server for the first time can take several minutes as the encryption keys are generated.

Once SSH is enabled, connect to the PhidgetSBC using its hostname or IP address (e.g. 'ssh root@phidgetsbc.local'). You must login using the 'root' account using same password set for the web interface. You may wish to set up a restricted user account for general use rather than running under the root account full-time.

Files can be sent to the board using scp or by uploading through the web interface in Userspace: Userspace Browser, or via sftp.

Text files (source code, etc.) can be edited using vi or with the web interface.

On Windows, we recommend Putty for an SSH client. You can get this at <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>.

# Custom Kernel and Filesystem

You can compile your own Kernel and flash it to the board. Phidgets Inc. provides our kernel patches on phidgets.com. It is left up to the user to configure an appropriate cross-compiler for kernel development. You may also be able to compile a new kernel on-board.

Installing a new kernel involves erasing the old kernel, flashing the new kernel, installing the new kernel modules, and rebooting:

1. `make uImage; make modules` in the kernel directory
2. `flash-eraseall /dev/mtd3`
3. `nandwrite -p /dev/mtd3 arch/arm/boot/uImage`
4. `make modules-install`
5. `reboot`

Custom kernels can also be flashed from the Recovery/Upgrade system.

Custom filesystem can be flashed from the Recovery/Upgrade system. The creation of custom filesystems is not documented, as the Debian package repository should provide enough flexibility for most users. However, if you need to create a root filesystem image, use the following settings:

Filesystem type is: UBIFS

Commands to create it:

1. `mkfs.ubifs -m 2KiB -e 126KiB -c 4050 -r \$ROOTFS/ system\_ubifs.img`
2. `ubinize -o system\_ubi.img -m 2KiB -p 128KiB -s 512 ubinize.cfg`

Where `ubinize.cfg` contains:

```
# Section header
[rootfs]
# Volume mode (other option is static)
mode=ubi
# Source image
image=system_ubifs.img
# Volume ID in UBI image
vol_id=0
# Volume size
vol_size=64128KiB
# Allow for dynamic resize
vol_type=dynamic
# Volume name
vol_name=rootfs
# Autoresize volume at first mount
vol_flags=autoresize
```

You then flash `system\_ubi.img` (not `system\_ubifs.img`) from the recovery system.

# Recovery / Upgrade System

The recovery / upgrade system is a small system from which the main system can be reset/upgraded, or recovered.

## Entering the Recovery System

The recovery system can be entered in two ways.

1. From the 'System: Backup & Restore' web interface page.
2. By holding down the reset button for 20+ seconds - until the green light has switched from flashing slowly to flashing quickly.

## Upgrades

Generally, you should not need to do a full system upgrade. Upgrades are meant to be performed via the package system, and Phidgets maintains it's own package repository from which to push out updates.

Occasionally, you may wish to go back to a clean install and upgrade to the latest rootfs/kernel from Phidgets. Phidgets will not be creating these images with every release of phidget21 as we did with PhidgetSBC1, rather they will be released several times a year, as needed for major changes not easy to push out via packages.

You can also flash your own custom kernel or root filesystem image.

## Factory Reset

This restores the kernel and root filesystem from backup, overwriting any changes that may have been made. This is equivalent to holding down the reset button for 10 seconds.

## Recovery

If the main filesystem has been damaged/misconfigured in such a way that it won't boot, you may be able to fix the issue / recovery important files before running a reset. The recovery system runs an SSH server that you can login to for console access. Username/password is: root/root.

You can mount the main root filesystem with the following commands (assuming it's not damaged):

1. `ubiattach /dev/ubi_ctrl -m 6`
2. `mount -t ubifs /dev/ubi0_0 /mnt`

# Technical

---

## Power Over Ethernet

Power over Ethernet can be used to provide both a network connection and power to a device when a power outlet is not available. This means that with the proper adapters, you can run the PhidgetSBC entirely off an Ethernet source. The PhidgetSBC does not draw power directly from a powered Ethernet line, but instead can use a setup where the power is split to a separate line again near the PhidgetSBC. The board has been tested and will work with Power Sourcing Equipment that can output 6-12VDC.

## Hardware Layout

The PhidgetSBC is based around the SC32440 processor. This is an ARM-920T based microprocessor from Samsung, which runs at 400MHz. Connected to this is 64 MB of SDRAM, 512 MB of large page NAND and a 10/100baseT Ethernet controller. The microprocessor brings USB Host port is connected to a 7-port USB 1.1 Hub chip. The integrated PhidgetInterfaceKit 8/8/8 is connected to one of the hub ports, with the other 6 ports being brought out to the user.

## Software Layout

The PhidgetSBC runs Debian/GNU Linux as its operating system and gets booted with U-Boot. The kernel is 2.6.x and generally kept up to date with the latest releases. The root filesystem is created using Buildroot and is mounted in a ~460MB nand partition using the UBIFS filesystem, in Read-Write mode.

Configuration data is located at `/etc/webif/`. This is where all configuration that can be set through the website is located.

User applications are stored in `/usr/userapps/`, each is their own directory.

The kernel is stored on bare Nand in its own 3MiB partition, in the uImage format.

## Date and Time

The date and time are set using ntp (network time protocol) at boot. The ntp daemon continues to run in the background and will periodically update the clock, keeping it very close to real time.

There is a real-time clock with battery backup which will preserve date/time across reboots, power removal. The real-time clock is synced to system time during reboot/shutdown. If power is unplugged suddenly, the real-time clock may not have the correct time.

## Wireless Networking System

Wireless networking is supported using the available adapter and is configured through the configuration interface. The back end of the system will be explained here.

## Configuration System

The configuration system used by the website is stored in `/etc/webif/`. These files should generally not be changed manually, but there is no reason why they could not be. It's very easy to enter invalid data that could cause the system to behave unexpectedly or not boot.

# Nand Layout

The board contains 64MB on Nand. This nand is split into 7 partitions as follows:

0: u-boot	size: 256K	Read Only
1: u-boot_env	size: 128K	Read Only
2: recovery_kernel	size: 2M	Read Only
3: kernel	size: 3M	Writable
4: flashfs	size: ~3.625M	Read Only
5: recovery_fs	size: ~ 43M	Read Only
6: rootfs	size: ~ 460M	Writable

The final size of flashfs/recovery\_fs/rootfs depends on the image size at production, and on the number/location of bad blocks in the NAND.

U-Boot and recovery kernel and filesystem cannot be written from Linux - this is a safety measure.

## Boot Process

This describes the boot process from power on.

1. Processor loads first 4 bytes from NAND into Steppingstone and runs it.
2. Steppingstone sets up RAM, copies u-boot from NAND into RAM and runs U-Boot.
3. U-Boot initializes the processor, sets GPIO state, etc., copies the linux kernel into RAM, sets up the kernel command line arguments, checks that the kernel image is valid, and boots it.
4. Linux boots, bringing up USB, Networking, NAND, etc. and then mounts the rootfs NAND partition on /.
5. init gets run as the parents of all processes, as uses the /etc/inittab script to bring up the system. This includes mounting other filesystems, settings the hostname, and running the scripts in /etc/init.d, among other things.
6. inittab then turns the green LED on.
7. inittab then sets up a getty on the first serial port, ready for interfacing using the debug board.

## Drivers for USB to Serial adapters

The SBC kernel contains driver support for the following USB to Serial Adapters.

Please consult the kernel documentation for details into the driver support for the USB to Serial adapters.

Company	Product
ConnectTech	WhiteHEAT
Keyspan	USA-18X, USA-28, USA-28X, USA-28XA, USA-28XB, USA-19, USA-19W, USA-19QW, USA-19QI, USA-49W, USA-49WLC
FTDI	Single Port Serial Adapter
Cypress	M8 CY4601 Family
Digi International	AccelePort USB Serial
Belkin	USB Serial Adapter F5U103
MCT	USB Single Port Serial Adapter U232
Inside Out Networks	Edgeport Serial Adapter
Prolific	PL2303

# U-Boot

U-Boot is used for setting up the processor and booting Linux, and is only accessible by the serial port. Normal users will not need to use it. If you are connected to the serial port, you will see the U-Boot prompt shortly after power up. You can view the environment variables for information on how to properly boot Linux on the PhidgetSBC.

Be very careful when modifying the u-boot partition. If it is damaged or overwritten, it is difficult to fix.

Refer to U-Boot documentation here: <http://www.denx.de/wiki/DULG/Manual> for more information on using U-Boot.

## SBC2 Device Specifications

Characteristic	Value
CPU	Samsung S3C2440
Core	ARM920T
CPU Speed	400MHz
Nand size	512MB
SDRAM	64MB
Boot time	30 Seconds
Ethernet	10/100baseT
USB	6-Port Full Speed
Operating Temperature	0 - 70°C
Power Input	6-15VDC
Power Consumption	1.2 watt base /w Ethernet
Per additional USB device <sup>1</sup>	2.5 watt Max

<sup>1</sup> including the PhidgetInterfaceKit



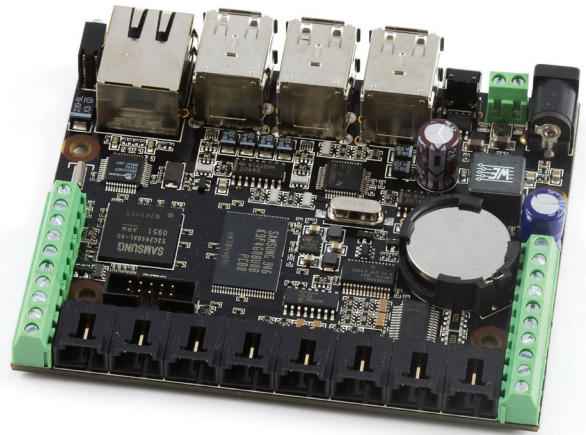
# PhidgetInterfaceKit 8/8/8

---

## General

### Product Features

- 8 analog inputs used to measure temperature, humidity, position, pressure, etc. Each analog input can be adjusted to sample at a data rate ranging from 1 sample to up to 1000 samples per second
- 8 digital inputs — with on-board noise filtering — used to convey the state of push buttons, limit switches, relays, etc.
- 8 digital outputs used to drive LEDs, solid state relays, transistors.



### 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](http://www.phidgets.com) >> Programming.

## 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.
- For full performance, the Phidget APIs are designed to be used in an event driven architecture. Applications that require receiving all the data streaming from the device will have to use event handlers, instead of polling.

## 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](http://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](http://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](http://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](http://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 InterfaceKit 8/8/8

## Functions

### **int InputCount() [get] : Constant = 8**

Returns the number of digital inputs supported by this PhidgetInterfaceKit.

### **bool InputState(int InputIndex) [get]**

Returns the state of a particular digital input. Digital inputs read True where they are activated and false when they are in their default state.

### **int OutputCount() [get] : Constant = 8**

Returns the number of digital outputs supported by this PhidgetInterfaceKit.

### **bool OutputState (int OutputIndex) [get,set]**

Sets/returns the state of a digital output. Setting this to true will activate the output, False is the default state. Reading the OutputState immediately after setting it will not return the value set - it will return the last state reported by the Phidget.

### **int SensorCount() [get] : Constant = 8**

Returns the number of sensors (Analog Inputs) supported by this PhidgetInterfaceKit. Note that there is no way of determining is a sensor is attached, and what sensor is attached.

### **int SensorValue(int SensorIndex) [get]**

Returns the sensed value of a particular Analog Input. SensorValue varies between 0-1000, corresponding to the 0-5V input range of the Analog Input.

If you are using an Analog Sensor from Phidgets Inc., it's manual will specify the formula used to convert SensorValue into the measured property.

### **int SensorRawValue (int SensorIndex) [get]**

Returns the full resolution of the Analog Input. This is a more accurate version of SensorValue. The valid range is 0-4095. Note however that the analog outputs on the Interface Kit 8/8/8 are only 10-bit values and this value represents an oversampling to 12-bit.

### **double SensorChangeTrigger (int SensorIndex) [get,set]**

Returns the change trigger for an analog input. This is the amount that an inputs must change between successive SensorChangeEvents. This is based on the 0-1000 range provided by getSensorValue. This value is by default set to 10 for most Interface Kits with analog inputs. SensorChangeTrigger is sometimes referred to as sensitivity.

### **int DataRate (int SensorIndex) [get,set]**

Gets/sets the data rate for an analog input. This corresponds to the fastest rate at which SensorChange events will be fired. The data rate is superseded by SensorChangeTrigger, which can be set to 0 if a constant data rate is required. Data Rate is in milliseconds and corresponds to the amount of time between events. Data Rate is bounded by DataRateMax and DataRateMin. The analog inputs cannot all be set to the fastest data rate at the same time - if this is attempted, an exception will be thrown when the data bandwidth has been exceeded. For data rates less than the maximum, data is still sampled at the maximum speed, and averaged between events for the user. Supported data rates are: 1, 2, 4, 8, and every multiple of 8 until DataRateMin. Setting an unsupported data rate (ie. 3, 9, 17) will result in a thrown exception. Note that data rate is limited to 16ms when opening over the Phidget Webservice.

### **int DataRateMax (int SensorIndex) [get]**

The maximum data rate that can be set for an analog input, in milliseconds.

### **int DataRateMin (int SensorIndex) [get]**

The minimum data rate that can be set for an analog input, in milliseconds. This is usually 1000.

### **bool Ratiometric() [get,set]**

Sets/returns the state of Ratiometric. Ratiometric = true configures the Analog Inputs to measure w.r.t VCC (nominal 5V). Ratiometric = false configures the Analog Inputs to measure w.r.t an internal precision 5V reference. Ratiometric is not updated from the Phidget. It is recommended to explicitly set Ratiometric when the Interfacekit is opened. After changing the ratiometric state, wait until the ratiometric property matches what was set before reading analog data.

## **Events**

### **OnInputChange(int InputIndex, bool State) [event]**

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

### **OnOutputChange(int OutputIndex, bool State), [event]**

An event that is issued when the state of a digital output changes.

### **OnSensorChange(int SensorIndex, int SensorValue), [event]**

An event that is issued when the returned value from a sensor (Analog Input) varies by more than the SensorChangeTrigger property.

# Technical

## Analog Inputs

### Using the Analog Inputs with Sensors provided by Phidgets

Analog Inputs are used to interface many different types of sensors. Each Analog Input provides power (Nominal +5VDC), ground, and an analog voltage return wire driven by the sensor to some voltage. The PhidgetInterfaceKit continuously measures this return voltage and reports it to the application.

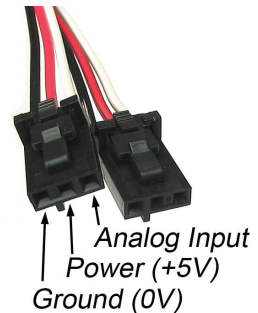
Analog Inputs are used to measure continuous quantities, such as temperature, humidity, position, pressure, etc. Phidgets offers a wide variety of sensors that can be plugged directly into the board using the cable included with the sensor.

### Using the Analog Inputs with your own sensors

For users who wish to interface their own sensors, we describe the Analog Inputs here.

### Mechanical

Each Analog Input uses a 3-pin, 0.100 inch pitch locking connector. Pictured here is a plug with the connections labeled. The connectors are commonly available - refer to the Table below for manufacturer part numbers.



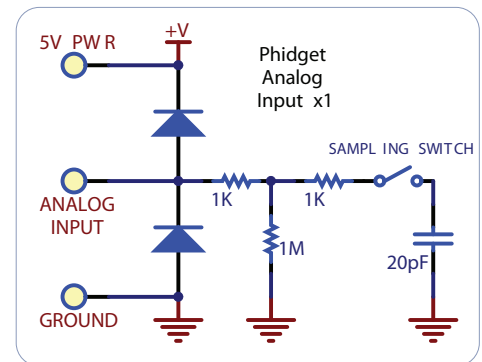
Cable Connectors		
Manufacturer	Part Number	Description
Molex	50-57-9403	3 Position Cable Connector
Molex	16-02-0102	Wire Crimp Insert for Cable Connector
Molex	70543-0002	3 Position Vertical PCB Connector
Molex	70553-0002	3 Position Right-Angle PCB Connector (Gold)
Molex	70553-0037	3 Position Right-Angle PCB Connector (Tin)
Molex	15-91-2035	3 Position Right-Angle PCB Connector - Surface Mount

Note: Most of the above components can be bought at [www.digikey.com](http://www.digikey.com)

### Electrical

The maximum total current consumed by all Analog Inputs should be limited to 400mA.

The analog measurement is represented in the software through the SensorValue as a value between 0 and 1000. A sensor value of 1 unit represents a voltage of approximately 5 millivolts. The RawSensorValue property brings out a 12-bit value (0-4095) for users who require maximum accuracy. Please note that the sampling is actually done with an oversampled 10-bit ADC, but reported as a 12-bit value to allow future expansion.



### Ratiometric Configuration

The group of Analog Inputs can be collectively set to Ratiometric mode from software using the Ratiometric property. If you are using a sensor whose output changes linearly with variations in the sensor's supply voltage level, it is said to be ratiometric. Most of the sensors sold by Phidgets are ratiometric (this is specified on the web product page and in the sensor's product manual).

Setting Ratiometric causes the reference to the internal Analog to Digital Converter to be set to the power supply voltage level. When Ratiometric is enabled, the maximum voltage returned on the Analog Input should be the +5V nominal power provided by the PhidgetInterfaceKit.

## Non-Ratiometric Configuration

If Ratiometric is false, the ADC reference is set to a 5.0V 0.5% stable voltage reference. The maximum voltage returned on the Analog Input should be maximum 5.0V. Note that the Analog Input power supply voltage is not affected by the setting of the Ratiometric property.

## Factors that can affect Accuracy

**High Output Impedance** - Sensors that have a high output impedance will be distorted by the 900K input impedance of the Analog Input. If your output impedance is high, it is possible to correct for this distortion to some extent in your software application.

**Power Consumption** - Sensor cables have some resistance, and the power consumption of the sensor will cause the sensor to have a slightly different ground from the Analog Input on the PhidgetInterfaceKit. The more power consumed by the sensor, and the longer the sensor cable, the more pronounced this effect will be.

**Intrinsic Error In Sensors** - For many sensors, the error is quite predictable over the life of the sensor, and it can be measured and calibrated out in software.

**Non-Ratiometric Configuration** - Voltage Reference error. The 5.0VDC voltage reference is accurate to 0.5%. This can be a significant source of error in some applications, but can be easily measured and compensated for.

## Changing the Data Rate

You can change the data rate for each Analog Input from 1 millisecond to 1 second. By default, the analog input data set is sent to the PC every 8ms. If, for example, you set the data rate to 1ms, you will receive a packet containing 8 milliseconds worth of 1 ms samples every 8ms. For values less than 8 ms, the data rate sets the sampling rate, not the transmission rate. When the Data rate is set at a multiple of 8 ms, the data rate sets both the sampling rate and the transmission rate.

There is also a limit as to how many channels can be set at a high sampling rate, since you will, at one point run out of bandwidth. We estimate that you can set up to 4 channels to 1ms or you could set all channels to 2ms. You will get an error when you exceed the available bandwidth, warning you of lost data samples.

Setting the data rate at 1, 2, or 4ms will not allow you to react to received sensor data any faster than every 8ms. You will simply get more sample data. This feature is useful if you need to log sensor data at less than 8 ms resolution.

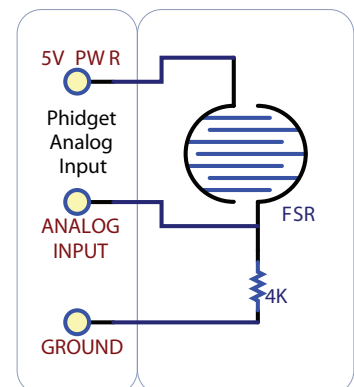
Note that data rate is limited to at most 16ms when opening over the Phidget Webservice. Actual data rate will depend on network latency.

## Connecting non-Phidget devices to the Analog Inputs

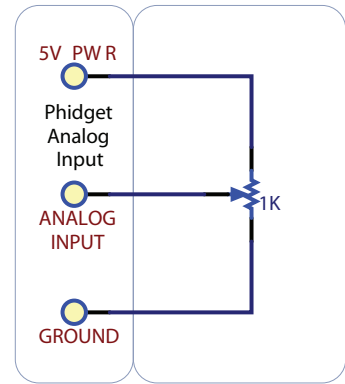
Here are some circuit diagrams that illustrate how to connect various non Phidgets devices to the analog inputs on your Phidget.

### Sensing the value of a variable resistance sensor

In this diagram, an FSR (Force Sensitive Resistor) is shown.

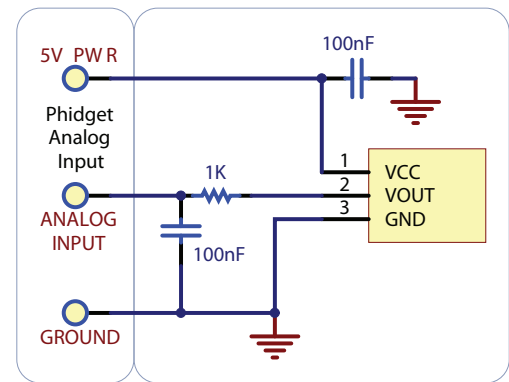


## Sensing the position of a potentiometer



## Interfacing to an arbitrary sensor

Note the use of power supply decoupling and the RC Filter on the output. The RC filter also prevents VOUT from oscillating on many sensors



## Non Phidgets Sensors

In addition to Phidgets sensors, any sensor that returns a signal between 0 and 5 volts can be easily interfaced. Here is a list of interesting sensors that can be used with the PhidgetInterfaceKit 8/8/8. Note: these sensors are not "plug & play" like the sensors manufactured by Phidgets.

Analog Sensors		
Manufacturer	Part Number	Description
MSI Sensors	FC21/FC22	Load cells - measure up to 100lbs of force
Humirel	HTM2500VB	Humidity sensors
Measurement Specialties	MSP-300	Pressure sensors - ranges up to 10,000 PSI
Freescale Semiconductor	MPXA/MPXH	Gas Pressure Sensors
Allegro	ACS7 series	Current Sensors - ranges up to 200 Amps
Allegro	A1300 series	Linear Hall Effect Sensors - to detect magnetic fields
Analog	TMP35 TMP36 TMP37	Temperature Sensor
Panasonic	AMN series	Motion Sensors
Honeywell	FS01, FS03	Small, accurate Piezo-resistive load cells
AllSensors-Europe	BARO-A-4V	Barometric Pressure Sensor - 600 to 1,100 mbar

Note: Most of the above components can be bought at [www.digikey.com](http://www.digikey.com)

## Digital Inputs

### Digital Input Hardware Filter

There is built-in filtering on the digital input, to eliminate false triggering from electrical noise. The digital input is first RC filtered by a 15K/100nF node, which will reject noise of higher frequency than 1KHz. This filter generally eliminates the need to shield the digital input from inductive and capacitive coupling likely to occur in wiring harnesses.

### Digital Input Hysteresis

The digital input has hysteresis - that is, it will hold it's current state (false or true), unless a large change occurs. To guarantee FALSE, the digital input must be at least 3.75V, and to guarantee TRUE, the digital input must be less than 1.25V.

### Digital Input Sampling Characteristics

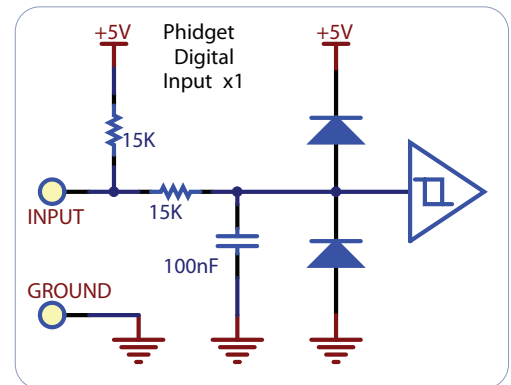
The state of the digital inputs are reported back to the PC periodically. During this sampling period, if a digital input was true for greater than 4.0ms, the digital input is guaranteed to be reported as true in software. This makes the digital input much more sensitive to reporting TRUE state, and makes it useful to watch for short events. Any Digital Input True events of less than 1.5ms are never reported.

### 5Volt Terminal Block

For users who need it, we provide 5V on the terminal block next to Digital Input 7.

### Functional Block Diagram

The digital inputs have a built in 15K 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.

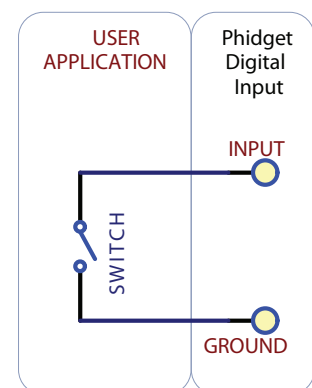


### Using the Digital Inputs

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

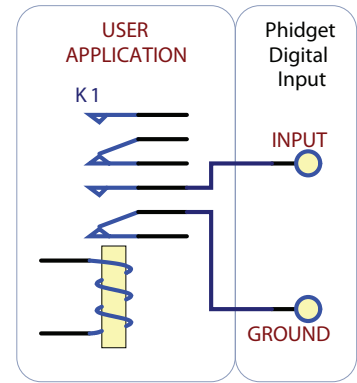
Closing the switch causes the digital input to report TRUE.





## 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.



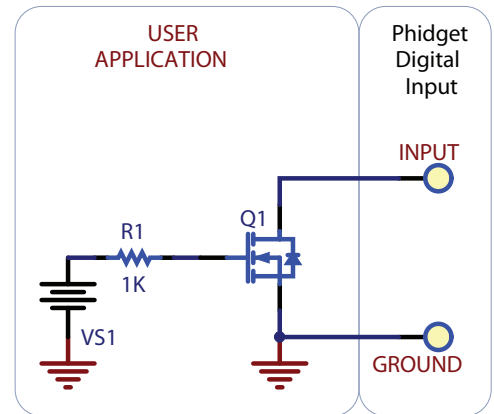
## Detecting an external Voltage with an N-Channel MOSFET

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  $> 270\mu\text{A}$ , the Digital Input is guaranteed to report TRUE.

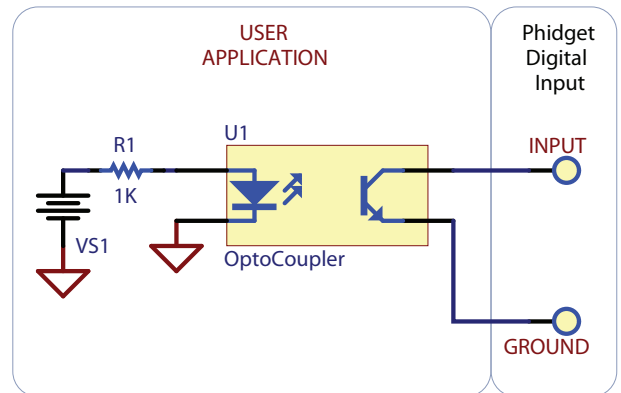
If the MOSFET is conducting  $< 67\mu\text{A}$ , the Digital Input is guaranteed to report FALSE.

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



## Isolating a Digital Input with an Optocoupler

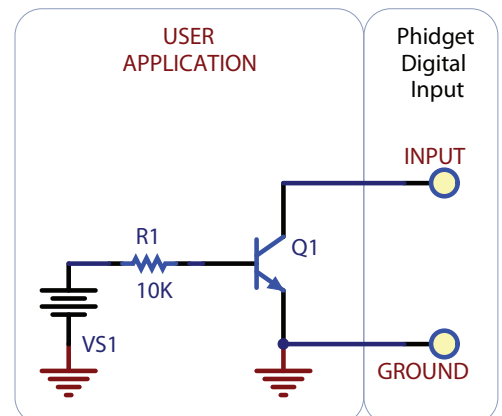
When driving current through the LED, the Digital Input will report TRUE. The amount of current required will depend on the optocoupler used. Design to sink at least  $270\mu\text{A}$  to cause the digital input to report TRUE, and less than  $67\mu\text{A}$  to report FALSE.



## 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.

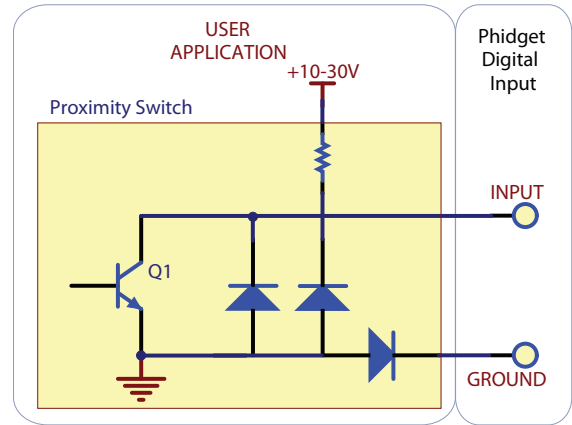
By designing to have Collector-Emitter current  $> 270\mu\text{A}$ , the digital input will report TRUE.



## 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 switch from Automation Direct to verify that it works with the Digital Inputs. Similar capacitive or inductive proximity switches from other manufacturers should work just as well.



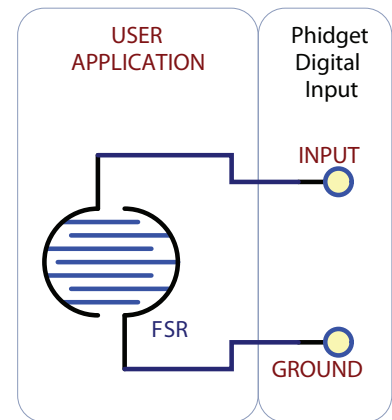
Manufacturer	Web Page	Capacitive Part No	Inductive Part No
Automation Direct	<a href="http://www.automationdirect.com">www.automationdirect.com</a>	CT1 Series	AM1 Series

## 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 3.75k Ohms, the Digital Input will go TRUE.

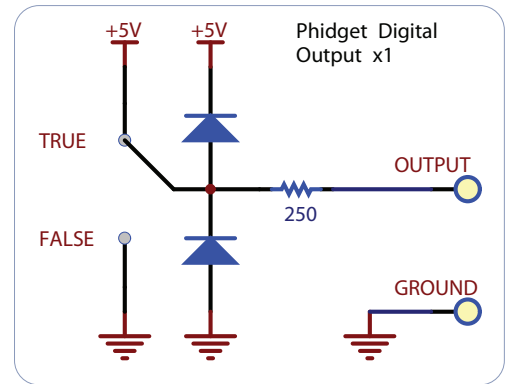
If the resistance rises above 75k Ohms, the Digital Input will go FALSE.



# Digital Outputs

## Functional Block Diagram

The 250 ohm resistance is internal to the PhidgetInterfaceKit 8/8/8, and limits the current that can flow through the output. This is intended to protect the device from being damaged if there is a short to ground or if an LED is used. The output is intended to drive TTL or CMOS inputs; it is not designed to provide power to an external circuit.



## Ground Protection

Ground terminals on the InterfaceKit share a common ground with USB ground. Because they are not internally isolated, these terminals will expose the USB ground potential of the PC to which they are connected. Be sure you are completely familiar with any circuit you intend to connect to the InterfaceKit before it is connected. If a reverse voltage or dangerously high voltage is applied to the input or output terminals, damage to the Phidget or the PC may result.

## 5Volt Terminal Block

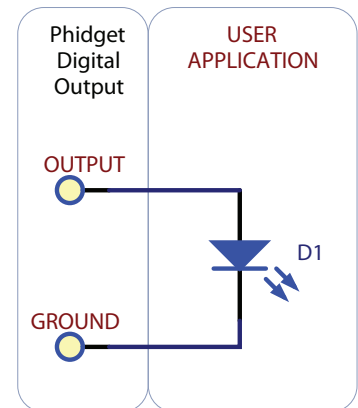
For users who need it, we provide 5V on the terminal block next to Digital Output 7.

## Using the Digital Outputs

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

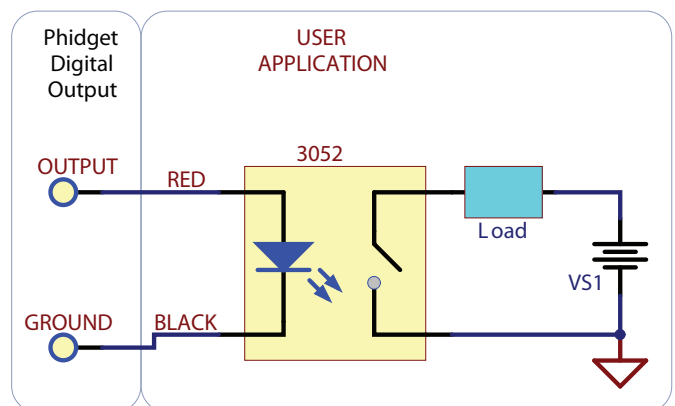
### Driving an LED with the Digital Output

Connecting an LED to a digital output is simple. Wire the anode to a digital output labeled 0 to 7 on the Interface Kit, and the cathode to a supplied ground, labeled G.



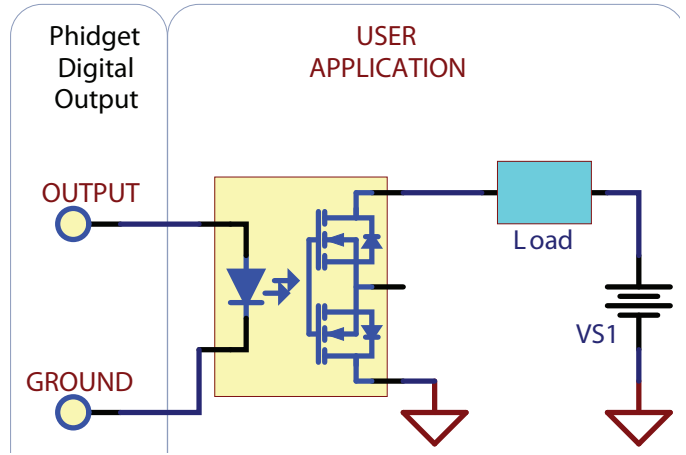
### Using a 3052 SSR Board with a Digital Output

Setting the digital output to true causes the output of the 3052 to turn on. This can be used to control AC or DC devices. The load can also be switched with the 3052 on the high side. High side switching is helpful for powering more complicated circuitry that cannot tolerate having multiple grounds.



### Isolating a Digital Output with a MOSFET based SSR

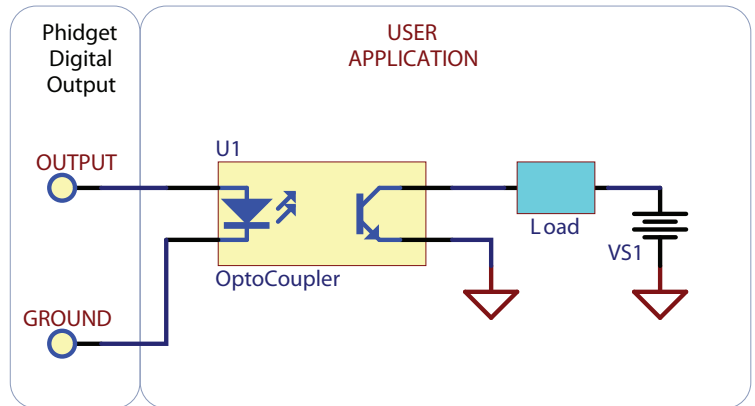
It's possible to wire up your own Solid State Relay to the digital output. MOSFET based SSRs have the advantage that they can be understood as being a simple switch. There are many other types of SSRs that are more suitable for controlling higher power, higher voltage AC devices that can also be controlled in the same fashion.



### Isolating a Digital Output with an Optocoupler

In some applications, particularly where there is a lot of electrical noise (automotive), or where you want maximum protection of the circuitry (interactive installations, kiosks), electrical isolation buys you a huge margin of protection.

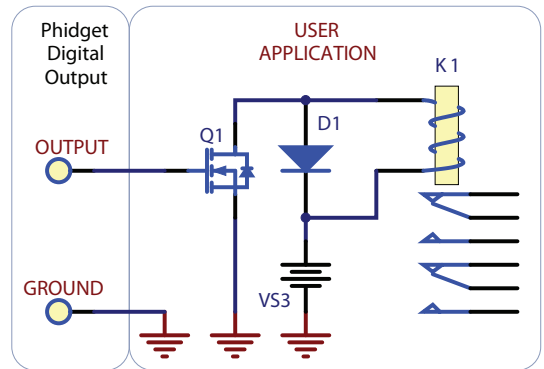
Driving the LED causes the output transistor to sink current. The maximum current through the transistor will depend in part on the characteristics of the optocoupler.



### Controlling a relay with a N-Channel MOSFET

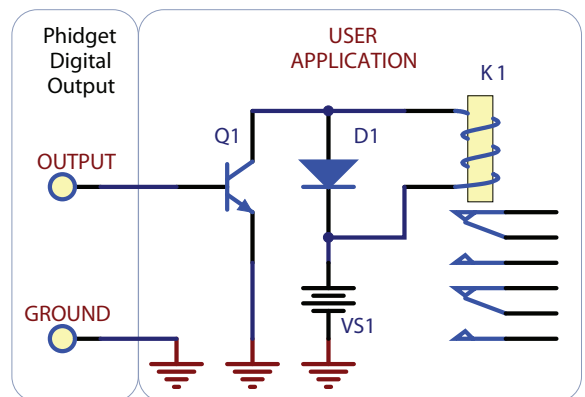
A inexpensive mosfet and flyback diode can be used to control larger loads - relays for example - directly from the digital output.

Be sure to use a Logic-Level MOSFET so that the +5V Digital Output is able to turn it on.



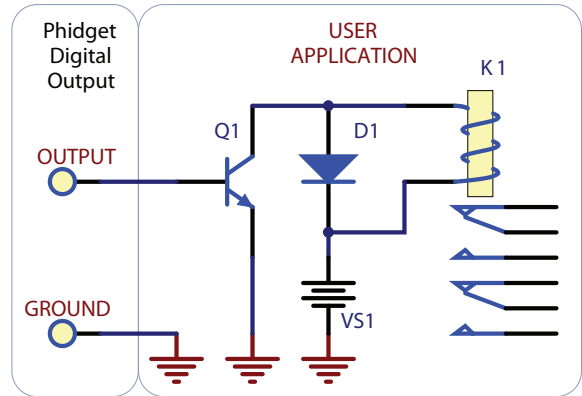
### Controlling a relay with a NPN transistor

This circuit is very similar to the N-channel mosfet - but you may already have NPN transistors on hand.



## Using a 3051 Dual Relay Board with one or two Digital Outputs

The 3051 Dual Relay Board is designed to be used with the PhidgetInterfaceKit 8/8/8. An Analog Input can be used to supply power to the relays, and one or two digital outputs used to control the relays. The 3051 is a good option if you need a couple relays in your project.



## Using the 6-Port USB Hub

### Powering the PhidgetSBC

An external 6 - 15V supply must be used to power the PhidgetSBC and any attached USB devices.

Connecting additional USB devices to the PhidgetSBC is as easy as plugging them into the on-board 4-port hub. Each USB port on the hub has a maximum current supply of 500mA. Ensure the power supply selected has a high enough current output to supply the required current to all external USB devices as well as the PhidgetSBC and any sensors or devices connected to it. The worst case requirement is 3 Watts input power per USB device. A 24 Watt 12VDC / 2 Amp power supply is provided with the 1072 - more than sufficient.

The USB Hub is a full-speed hub with a transfer rate of 12Mbits/second. We chose to go with a full speed implementation since it is fast enough to handle traffic from Phidgets; an added benefit is lower power consumption.

### Chaining the USB Hubs

The 1072 follows USB specifications and can be daisy chained to the maximum hub depth of 4.

## InterfaceKit 8/8/8 Device Specifications

Characteristic	Value
<b>Board</b>	
USB Voltage	4.75 to 5.25 V
USB-Power Current Specification	Max 500 mA
Quiescent Current Consumption	13 mA
Available External Current (source)	487 mA
Operating Temperature	0 - 70°C
<b>Analog Inputs</b>	
Impedance	900K ohms
5V Reference Error	Max 0.5%
Update Rate	1000 samples/second max for 4 channels 500 samples/second max for all 8 channels 62.5 samples/second max over webservice
<b>Digital Inputs</b>	
Pull-Up Resistance	15K ohms
Low Voltage (True)	1.25V Max
High Voltage (False)	3.75V Min
Maximum Voltage	±15V
Update Rate	~125 samples/second
Recommended Wire Size	16 - 26 AWG
Wire Stripping	5-6mm strip
<b>Digital Outputs</b>	
Series Resistance	300 ohms
Update Rate	~125 samples/second
Recommended Wire Size	16 - 26 AWG

# Product History

Date	Board Revision	Device Version	Comment
February 2011	0	200	Product Release

## 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](mailto:support@phidgets.com)

## Legal Information

For licensing and warranty information, please see the Phidgets End User License. It is available on our website at: [http://www.phidgets.com/documentation/Licenses/Phidgets\\_End\\_User\\_License\\_Agreement.pdf](http://www.phidgets.com/documentation/Licenses/Phidgets_End_User_License_Agreement.pdf).

By using this product, you are agreeing to be bound by the terms and conditions set forth by this licensing agreement.