# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>3</td>
</tr>
<tr>
<td>Hardware Setup</td>
<td>4</td>
</tr>
<tr>
<td>QuickStart with the Desktop GUI</td>
<td>6</td>
</tr>
<tr>
<td>• Step #4: Interact</td>
<td>9</td>
</tr>
<tr>
<td>Using IO</td>
<td>9</td>
</tr>
<tr>
<td>Using SPI</td>
<td>14</td>
</tr>
<tr>
<td>Using I2C</td>
<td>17</td>
</tr>
<tr>
<td>Using UART</td>
<td>23</td>
</tr>
<tr>
<td>QuickStart with Python</td>
<td>26</td>
</tr>
<tr>
<td>• Step #2: Find Connected Devices</td>
<td>27</td>
</tr>
<tr>
<td>QuickStart with CoolTerm</td>
<td>31</td>
</tr>
</tbody>
</table>
Overview

The Binho Nova is designed for ultimate flexibility, as such, there are multiple ways to control and interact with it. This guide will give you a brief introduction to each interface and show you how to get up and running quickly.

If you don't already have a Binho Nova, you can get one here:

**Binho Nova Multi-Protocol USB Host Adapter**
The Binho Nova Multi-Protocol USB Host Adapter allows you to interface your computer directly to hardware circuits. This device is powered...

[https://www.adafruit.com/product/4459](https://www.adafruit.com/product/4459)

Hooking the device up to your circuit will be easiest when using one of the available accessory boards, available for Stemma QT/Qwiic ([https://adafru.it/IDO](https://adafru.it/IDO)) and Feather form-factor products ([https://adafru.it/IDp](https://adafru.it/IDp)), as well as a breadboard adapter. The breadboard adapter is included with the Binho Nova, but it's also available for purchase separately when you need more.

**Binho Breadboard Breakout**
This board breaks a male 2x5 1.27mm connector out to breadboard-friendly 2.54mm / 0.1" pitch headers. It's an easy way to interface your

[https://www.adafruit.com/product/4460](https://www.adafruit.com/product/4460)

This guide will begin with the hardware setup, which includes getting up and running on Windows, MacOS, or Ubuntu. Then we'll show you how to connect the Binho Nova to your circuit. Once everything is connected, follow the guides below to learn how to interact with the Binho Nova using Mission Control - the name of our GUI desktop software, Python libraries and even using a serial console.
Level #1 - GUI Guide

The easiest way to get familiar with your Binho Nova is by taking it for a test drive with the cross-platform desktop GUI software - Mission Control. Even if you intend to use it in automated environments and applications, Mission Control is a great starting point before diving in to writing scripts.

Level #2 - Python Guide

When you’re ready to start writing your own automated scripts, take a look at the guide for using open-source Python libraries.

Level #3 - ASCII Guide

And when you're ready to go to the deepest level of device interaction to learn how to control it from any language/platform capable of opening a serial port and sending ASCII text, have a look at the final guide. This guide is presented using CoolTerm due to it's elegant GUI and relatively identical performance across all the platforms supported by the Binho Nova.

Let's Get Started!

Hardware Setup

The Binho Nova Multi-Protocol USB Host Adapter works on Windows, MacOS, and Linux, and getting up and running with it is pretty quick.

Driver Installation

Great News! Most modern operating systems already have the device drivers needed for your Binho Nova Multi-Protocol USB Host Adapter to work properly installed. The Binho Nova utilizes the standardized USB Communications Device Class driver in order to achieve maximum compatibility with as many systems as possible. As such, there's no driver to download and install.

Windows 7 does not have the standard USB CDC driver mentioned above installed by default, however a driver is available on Binho's support website. You can download it [here](https://adafruit.it/IDq).
Device Permissions

If this is the first time connecting to a hardware device from your account on your Mac, you'll need to grant permission to your user account. On MacOS, you'll need to ensure that your user account is added to the `wheel` group in order to grant it the permissions needed to access the hardware. This is only necessary once per user. Simply run the following command:

```
sudo dscl . -append /groups/wheel GroupMembership
```

If you're on Ubuntu or another Linux distribution and running into issues with your device, please perform the procedure shown on this support page (https://adafru.it/1Dr).

Connection to the Computer

Use the provided USB Type-C (male) to USB Type-A (male) cable to connect the Binho Nova Multi-Protocol USB Host Adapter to your host PC. The device can be plugged into any available USB port - there is no requirement to plug it into a USB 3.0 port.

Upon power up, the Status LED on the Binho Nova will shine yellow while it is waiting for the COM port to be opened. When connected properly, the device should immediately enumerate and become available for use as a new COM port. Once a serial connection has been established between the device and the host computer, the Status LED will shine blue.

Connection to the Test Circuit

The Binho Nova Multi-Protocol USB Host Adapter features an integrated wire harness terminated with a 1.27mm pitch 2x5 IDC Connector. This harness contains the 5 signal pins, 1 x 3V3 power signal, 1 x VBUS power signal, and 3 x GND signals.

The figure below shows the connector pinout and channel functions:
QuickStart with the Desktop GUI

Step #1: Download and Install the Binho Mission Control Software

The Binho cross-platform software is available for Windows, Mac, and Linux, and provides an easy-to-use GUI interface for your Binho Nova. You can download the appropriate release for your operating system here (https://adafru.it/IDs).

Once the download completes, unzip the directory. The Windows 32-bit and Windows 64-bit releases are distributed with .exe installers, whereas Mac and Linux distributions can simply be unzipped and run.
The Windows Defender SmartScreen prompt is displayed when running the installer. Click "More Info" and then the "Run Anyway" button to continue. The software releases are codesigned, however the reputation is still building within Microsoft SmartScreen so that eventually this warning will go away.

Here's what the process looks like on Mac:

Here's what the process looks like on Ubuntu:
Step #2: Launch the Software & Connect to the Device

Now it's time to launch the software. If you haven't done so already, connect the Binho Nova to your computer using the provided USB Cable. Once you've launched the application, the available devices will be displayed in a drop-down box at the top. Simply click the "Connect" button to connect to the selected host adapter. If no devices are shown in the list, go ahead and click the "Refresh" button to check again, perhaps it was still enumerating after plug-in when the software initially checked for devices.
Step #3: Set the Operation Mode

The device mode of operation is displayed in the bottom left corner of the main application window. The default mode of operation is IO, where all 5 pins can be used for any function from the IO tab of the software. Activating SPI or I2C modes of operation will assign the pins associated with those protocols to their protocol-specific function, and the remaining unused IO pins will still be available for use from the IO tab. Setting SPI or I2C mode of operation is done on the respective protocol tab as shown below.

Step #4: Interact

At this point, you're all set to start interacting with your device. For final step, please take a look at the next few pages in this guide.

Using IO

The five IO pins of the Binho Nova Multi-Protocol USB Host Adapter can be used in a variety of modes, and unassigned pins can be used for their IO functions even when other pins are used for SPI or I2C operation. The image below shows the pinout and functionality of each of the pins:
Digital Input

All 5 of the IO pins can be used as digital inputs. This can be achieved by selecting "DIN" from the pin assignment dropdown box. In this mode, the value of the input signal can be read by clicking the "Update" button. The value is displayed as either a logical 0 or 1.

Digital Output

All 5 of the IO pins can be used as digital outputs. This can be achieved by selecting "DOUT" from the pin assignment dropdown box. In this mode, the value of the output signal can be set by clicking either the "Set Low" or "Set High" buttons.
Analog Input

All 5 of the IO pins can be used as analog inputs. This can be achieved by selecting “AIN” from the pin assignment dropdown box. In this mode, the value of the input signal can be read by clicking the “Update” button. Note that the voltage reference is 3.3 Volts. The measured value is displayed in Volts.
Analog Output

Only IO1 is capable of functioning as an analog output. This pin can be configured to use the 10-bit DAC to output a voltage between 0 and 3.3 Volts. IO1 can be set as an analog output by setting "AOUT" from the pin assignment dropdown box. In this mode, the value of the output signal can be set by dragging the slider from 0 to 1024. Note that the output signal is updated only after releasing the slider.
PWM

IO0, IO2, IO3, and IO4 pins have PWM output capabilities. The duty cycle and frequency are both controllable from the software. This can be achieved by selecting "PWM" from the pin assignment dropdown box. In this mode of operation, the frequency can be set from 750Hz to 80kHz by dragging the top slider, and the duty cycle can be set by dragging the bottom slider from 0 to 1024. Please note that IO0 and IO2 share a frequency, and IO3 and IO4 share a separate frequency. As such, updating the frequency slider for one of these channels will also update the corresponding channel's frequency setting.
Using SPI

The diagram below shows the pin assignments (in yellow) for the SCK, MOSI, and MISO pins on the Binho Nova Multi-Protocol USB Host Adapter when it is in SPI mode.
Configuring the SPI Bus Settings

Once the SPI mode of operation has been activated by clicking the "Activate SPI Mode" button on the SPI tab, the SPI settings will be unlocked and ready for configuration.

The Binho Nova supports SPI bus clock frequencies from 500kHz to 12MHz, and all four SPI modes are supported. Data can be configured for MSB or LSB first bit orders and for 8 or 16 bits per transfer.

Finally, configure the Chip Select signal, which can be active high or active low, to be either on the IO0 or IO1 signals.

Sending & Receiving Data

Since SPI is a full-duplex communication protocol, the process of sending and receiving data is identical, and is simply referred to as "transfer" from within our
software. To transfer data from the host adapter to a SPI secondary device, simply enter in the data into the textbox and click the "Transfer" button. Data can be typed in binary, decimal, or hex formats. In the case of binary, the 8-bit value should be preceded by a prefix of "0b", likewise a hex value should be preceded by "0x". Numbers without a prefix will be evaluated as a decimal number.

For clarity, here is an example of a valid data transfer displayed in each of the three supported bases:

Decimal: 222 173 190 239

Binary: 0b11011110 0b10101101 0b10111110 0b11101111

Hex: 0xDE 0xAD 0xBE 0xEF

You can find the results of the transfer displayed in the transaction list at the bottom of the window.
Note that reading data from a SPI secondary device can be achieved by transfer bytes of value 0 for the desired number of bytes to be read back. This is how the main clocks out data from the secondary device.

Using I2C

The diagram below shows the pin assignments (in purple) for the SCL and SDA pins on the Binho Nova Multi-Protocol USB Host Adapter when it is in I2C mode.

![I2C Diagram]

Configuring the I2C Bus Settings

Once the I2C mode of operation has been activated by clicking the "Activate I2C Mode" button on the I2C tab, the I2C settings will be unlocked and ready for configuration.

The Binho Nova supports I2C bus clock frequencies from 100kHz up to 3.4MHz, which covers all common operating modes (standard, full, fast, and high speed modes). Clock stretching and repeated starts are also supported. There are internal pull-up resistors which can be engaged or disabled as necessary.
Scanning for Devices

If you already know the address of your target I2C secondary device, you can type it directly into the Address textbox (in either decimal or hex, preceded by "0x"). However, Binho makes it easy to discover devices on the I2C bus -- simply leave the "Address" textbox empty and click the "Scan Entire Bus..." button and the host adapter will check for devices on the bus. Any devices that are found will then be displayed in a listbox. Simply select the address of the device that you'd like to interact with from the list.
The remainder of the commands in this guide require that the "Target Address" has already been set, either by scanning the bus or via directly typing the secondary address.

Adafruit typically refers to I2C device addresses in 7-bit format, but some manufacturer’s datasheets give device addresses in 8bit format. You can use the "Format" field to choose your desired setting.

Reading Data

Reading data from the secondary device is as simple as providing the number of bytes to read (in either decimal or hex, preceded by "0x") and clicking the "Read [n] Byte(s)" button.
Note that the results of this action are displayed in the transaction list at the bottom of the window.

Writing Data

Writing data to the secondary device can be done by entering the data into the "Write" textbox and clicking the "Write [n] Byte(s)" button. An option checkbox can be selected to immediately send a repeated start bit after writing the data. Data can be typed in binary, decimal, or hex formats. In the case of binary, the 8bit value should be preceded by a prefix of "0b", likewise a hex value should be preceded by "0x". Numbers without a prefix will be evaluated as a decimal number.

For clarity, here is an example of a valid data transfer displayed in each of the three supported bases:

Decimal: **222 173 190 239**

Binary: **0b11011110 0b10101101 0b10111110 0b11101111**
Hex: 0xDE 0xAD 0xBE 0xEF

Read Register

While not an official part of the I2C specification, a very common implementation across various I2C devices makes it convenient to have a Read Register function. Read Register is effectively a 1-byte Write command which contains the desired register address, followed immediately by a read command.
The Read Register command supports both 8-bit and 16-bit register addresses. When targeting a 16-bit register on a secondary device, simply use the hex address padded with leading zeros as necessary.

For example, to address register 32 on a device which is expecting a 16-bit address, enter **0x0020** in the Register Address textbox.
Note that the results of this action are displayed in the transaction list at the bottom of the window.

Using UART

The diagram below shows the pin assignments (in green) for the TX and RX pins on the Binho Nova Multi-Protocol USB Host Adapter when it is in UART mode.
Configuring the UART Settings

Once the UART mode of operation has been activated by clicking the "Activate UART Mode" button on the UART tab, the UART settings will be unlocked and ready for configuration.

The Binho Nova supports UART baud rates from 300bps up to 1000000bps. The UART bus can be configured for 5, 6, 7, or 8 bit data packets; Even, Odd, or no Parity bit; and to use 1 or 2 stopbits.

Starting & Stopping the UART Bridge

Once the settings for the UART bus have been configured, click the "Start UART Bridge" button to begin transacting on the bus. Note that when the UART Bridge is started, the caption of the button will change to "Stop UART Bridge". The UART Bridge must be stopped before performing any other interactions with the Binho Nova.
Transmitting Data

Data can be transmitted on the UART bus by typing it into the text box and clicking the Send button. By default, no line endings are used when sending the entered text. However, it's very common for lines to be terminated by \n, \r, or \r\n so any of these options can be configured using the radio buttons below the Send button.
Receiving Data

Data received on the UART bus will be displayed in the console as it arrives.

QuickStart with Python

Binho provides Python libraries to make it lighting fast to start automating test and development tasks with a Binho Nova. The packaged releases contains two libraries:

binhoHostAdapter

This library is essentially a wrapper for all of the commands presented in the ASCII Command Set (https://adafru.it/IDt) documentation. The library is written in such a way to support multiple devices as well as properly handling INTERRUPTS (https://adafru.it/IDu) by making use of threads.
binhoUtilities

This library provides a handful of functions which aid in device management, as in identifying COM ports and Binho devices attached to the host computer.

Step #1: Download and Install the Binho Host Adapter Libraries

The officially-supported Python library can easily be installed using pip:

```bash
pip install binho-host-adapter
```

This library is cross-platform and is intended for use with Python 3.x. Source code can be found here. (https://adafruit.it/IDv)

Step #2: Find Connected Devices

Let's use the `binhoUtilities` (https://adafruit.it/IDw) class to find devices attached to this computer. Start by creating a new Python script, call it `binhoDemo.py`, and enter the following code:

```python
from binhoHostAdapter import binhoUtilities
from binhoHostAdapter import binhoHostAdapter

utilities = binhoUtilities.binhoUtilities()
devices = utilities.listAvailableDevices()
```
print('Binho Host Adapters Found On The Following Ports:')
print(devices)

Make sure your device is plugged in and then run the script. A list of ports where Binho devices have been found will be printed to the screen.

Step #3: Connect to a Device

Now that you know how to use the `binhoUtilities` class to discover devices, let's extend the script to connect with the first device it discovers. We'll leave a comment where we'll implement our desired functionality later in Step #4, and then just immediately close the connection and exit.

```python
from binhoHostAdapter import binhoUtilities
from binhoHostAdapter import binhoHostAdapter

utilities = binhoUtilities.binhoUtilities()
devices = utilities.listAvailableDevices()

print('Binho Host Adapters Found On The Following Ports:')
print(devices)
print()

# Make sure at least one Binho device was found before proceeding
if len(devices) &lt; 1:
    print('No Binho Host Adapter found...Quitting script')
    exit(1)

# Target the port of the first device in the list
targetPort = devices[0]
print('Connecting to host adapter on ' + targetPort)
print()
```
# Connect to the binho device on the target port
binho = binhoHostAdapter.binhoHostAdapter(targetPort)
print('Connected!')
print()

# Here's where we'll implement our desired functionality, see Step #4
#
# Close the connection to the device before exiting
binho.close()
print('Connection closed!')
print()
# Exit gracefully
exit(0)

Step #4: Interact

Now that we have a basic script which handles device discovery, connection, and disconnection, all that's left to do is implement our desired functionality. The simple example below shows how to generate a PWM signal on IO0.

```python
from binhoHostAdapter import binhoUtilities
from binhoHostAdapter import binhoHostAdapter

utilities = binhoUtilities.binhoUtilities()
devices = utilities.listAvailableDevices()

print('Binho Host Adapters Found On The Following Ports:')
print(devices)
print()
# Make sure at least one binho device was found before proceeding
```
if len(devices) < 1:
    print('No Binho Host Adapter found... Quitting script')
    exit(1)

# Target the port of the first device in the list
targetPort = devices[0]
print('Connecting to host adapter on ' + targetPort)
print()

# Connect to the binho device on the target port
binho = binhoHostAdapter.binhoHostAdapter(targetPort)
print('Connected!')
print()

# Set the LED color to GREEN
binho.setLEDColor('GREEN')
print('Set LED to Green')
print()

# Set the operation mode to IO
binho.setOperationMode(0, 'IO')

# Set I00 to PWM mode
binho.setIOPinMode(0, 'PWM')

# Set I00 PWM Frequency to 75kHz
binho.setIOPinPWMFreq(0, 75000)

# Set I00 PWM duty cycle to 512/1024
binho.setIOPinValue(0, 512)
print('I00 set to PWM with 50% duty cycle / 75kHz frequency')
print()

# Close the connection to the device before exiting
binho.close()
print('Connection closed!')
print()

# Exit gracefully
exit(0)
Going Further

We've just covered the most basic case of using the Python libraries to automate your Binho Nova. The full documentation of the python libraries can be found on Binho's support website [here](https://adafru.it/IDx).

There's also a seamless integration with CircuitPython ([link](https://adafru.it/IDy)) so you can easily use your Binho Nova to interact with all of your favorite Adafruit boards directly from your desktop computer.

---

QuickStart with CoolTerm

Step #1: Download and Install CoolTerm

CoolTerm is a very popular cross-platform serial console application developed by Roger Meier. It's got all the necessary features for communicating with hardware devices and an elegant user interface.

You can [download the latest release](https://adafru.it/IDz) directly from his website. Installation simply entails extracting the folder from it's compressed .zip archive and placing in a known location on your drive.
Step #2: Connect the Binho Multi-Protocol USB Host Adapter

Connect the Binho Nova to your PC using the provided USB cable. The status LED on the host adapter will shine yellow when it is powered and waiting for the host PC to open the connection to the device.

Step #3: Configure the Serial Connection

Configure the serial connection by clicking the "Options" button on the toolbar.

The connection parameters should be configured to the following:

- Baudrate: 1000000
- Databits: 8
- Parity: None
- Stop Bits: 1
- No Flow Control
Now let's adjust some CoolTerm settings to make it even easier to use.

1. Click the "Options" button again and select the "Terminal" Category from the list on the left.

2. Change the "Terminal Mode" setting from "Raw Mode" to "Line Mode"

3. Select the checkbox to enable "Local Echo" functionality.

Step #4: Open the Serial Connection

Now that the connection parameters have been configured correctly, it's time to connect to the Binho Nova. Click the "Connect" button on the toolbar to open the serial connection.
Once the serial connection has been established, the Status LED on the Binho Nova will shine blue. You can now begin sending commands to the host adapter.

Step #5: Send Commands

Sending commands to the device is as simple as typing a command and pressing Enter. Let’s get the unique device ID of this host adapter by sending the device command `+ID`.

![Serial connection interface with command line showing `+ID` command entered.](image)
Note that the host adapter will respond to every command. If you do not receive a response from the command, then the serial connection may not be configured properly.

You're ready to hit the ground running. Check out the full User Guide (https://adafruit.it/IDA) on Binho's support website to learn how to take advantage of the full set of supported protocols and features of your host adapter.