CircuitPython Libraries with the Binho Nova Multi-Protocol USB Host Adapter

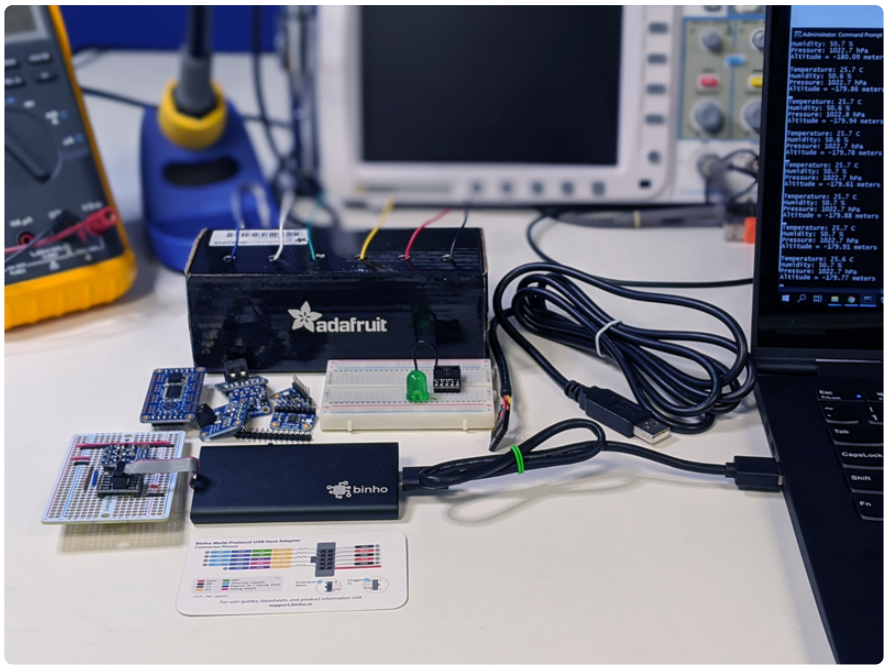
Created by Francis Guevarra

https://learn.adafruit.com/circuitpython-with-binho-nova-multi-protocol-usb-host-adapter

Last updated on 2024-03-08 03:39:45 PM EST

Table of Contents

Overview

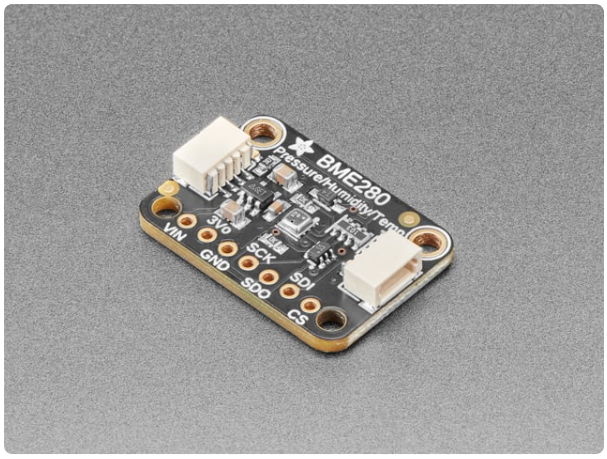


The **Binho Nova** Multi-Protocol USB Host Adapter allows you to interface your computer directly to hardware circuits. This device is powered by the USB connection to the host PC and is also able to provide downstream power to test circuits. The following guide details how to use a **Binho Nova** with CircuitPython libraries to interface with sensors and components via I2C (Inter-Integrated Circuit), SPI (Serial Peripheral Interface), GPIO's (General Purpose Input/Output), or UART (Universal Asynchronous Receiver/Transmitter).

Parts

Binho Nova Multi-Protocol USB Host Adapter
Discontinued - you can grab our Adafruit Trinkey QT2040 - RP2040 USB...
https://www.adafruit.com/product/4459

And things we'll be connecting to!

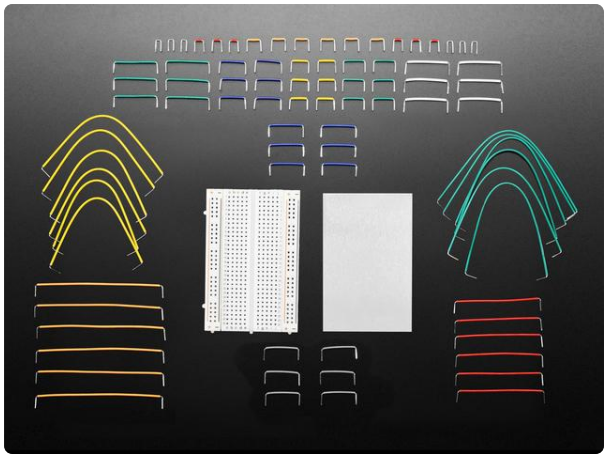


Adafruit BME280 I2C or SPI Temperature Humidity Pressure Sensor

Bosch has stepped up their game with their new BME280 sensor, an environmental sensor with temperature, barometric pressure and humidity! This sensor is great for all sorts...
https://www.adafruit.com/product/2652

Diffused Green 10mm LED (25 pack)

Need some big indicators? We are big fans of these huge 10mm diffused green LEDs. They are fairly bright so they can be seen in daytime, and from any angle. They go easily into a...
https://www.adafruit.com/product/844



Half Size Breadboard + 78 Piece 22AWG Jumper Wire Bundle

This is a cute half-size breadboard with an assortment of small jumper wires, great for prototyping. The breadboard is 2.2" x 3.4" (5.5 cm x 8.5 cm) with a standard...
https://www.adafruit.com/product/3314

FTDI Serial TTL-232 USB Cable

Just about all electronics use TTL serial for debugging, bootloading, programming, serial output, etc. But it's rare for a computer to have a serial port anymore. This is a USB to...
https://www.adafruit.com/product/70

Running CircuitPython Code without CircuitPython

There are two parts to the CircuitPython ecosystem:

- **CircuitPython firmware**, written in C and built to run on various microcontroller boards (not PCs). The firmware includes the CircuitPython interpreter, which reads and executes CircuitPython programs, and chip-specific code that controls the hardware peripherals on the microcontroller, including things like USB, I2C, SPI, GPIO pins, and all the rest of the hardware features the chip provides.
- **CircuitPython libraries,** written in Python to use the native (built into the firmware) modules provided by CircuitPython to control the microcontroller peripherals and interact with various breakout boards.

But suppose you'd like to use CircuitPython **libraries** on a board or computer that does not have a native CircuitPython **firmware** build. For example, on a PC running Windows or macOS. Can that be done? The answer is yes, via a separate piece of software called **Blinka**. Details about Blinka follow, however it is important to realize that the **CircuitPython firmware is never used**.

CircuitPython firmware is NOT used when using Blinka.

Adafruit Blinka: a CircuitPython Compatibility Library

Enter **Adafruit Blinka**. Blinka is a software library that emulates the parts of CircuitPython that control hardware. Blinka provides non-CircuitPython implementations for `board`, `busio`, `digitalio`, and other native CircuitPython modules. You can then write Python code that looks like CircuitPython and uses CircuitPython libraries, without having CircuitPython underneath.

There are multiple ways to use Blinka:

- Linux based Single Board Computers, for example a Raspberry Pi
- Desktop Computers + specialized USB adapters
- Boards running MicroPython

More details on these options follow.

Raspberry Pi and Other Single-Board Linux Computers

On a Raspberry Pi or other single-board Linux computer, you can use Blinka with the regular version of Python supplied with the Linux distribution. Blinka can control the hardware pins these boards provide.

Desktop Computers

On Windows, macOS, or Linux desktop or laptop ("host") computers, you can use special USB adapter boards that that provide hardware pins you can control. These boards include MCP221A (https://adafru.it/IfV) and FT232H (https://adafru.it/xia) breakout boards, and Raspberry Pi Pico boards running the **u2if** software (https://adafru.it/Sje). These boards connect via regular USB to your host computer, and let you do GPIO, I2C, SPI, and other hardware operations.

MicroPython

You can also use Blinka with MicroPython, on MicroPython-supported boards (https://adafru.it/SBi). Blinka will allow you to import and use CircuitPython libraries in your MicroPython program, so you don't have to rewrite libraries into native MicroPython code. Fun fact - this is actually the original use case for Blinka.

Installing Blinka

Installing Blinka on your particular platform is covered elsewhere in this guide. The process is different for each platform. Follow the guide section specific to your platform and make sure Blinka is properly installed before attempting to install any libraries.

Be sure to install Blinka before proceeding.

Installing CircuitPython Libraries

Once Blinka is installed the next step is to install the CircuitPython libraries of interest. How this is down is different for each platform. Here are the details.

Linux Single-Board Computers

On Linux single-board computers, such as Raspberry Pi, you'll use the Python `pip3` program (sometimes named just `pip`) to install a library. The library will be downloaded from pypi.org (https://adafru.it/19ff) automatically by `pip3`.

How to install a particular library using `pip3` is covered in the guide page for that library. For example, here is the `pip3` installation information (https://adafru.it/OkF) for the library for the LIS3DH accelerometer.

The library name you give to `pip3` is usually of the form `adafruit-circuitpython-libraryname`. This is not the name you use with `import`. For example, the LIS3DH sensor library is known by several names:

- The GitHub library repository is Adafruit_CircuitPython_LIS3DH (https://adafru.it/uBs).
- When you import the library, you write `import adafruit_lis3dh`.
- The name you use with `pip3` is `adafruit-circuitpython-lis3dh`. This the name used on pypi.org (https://adafru.it/19ff).

Libraries often depend on other libraries. When you install a library with `pip3`, it will automatically install other needed libraries.

Desktop Computers using a USB Adapter

When you use a desktop computer with a USB adapter, like the MCP2221A, FT232H, or u2if firmware on an RP2040, you will also use pip3. However, **do not install the library with `sudo pip3`**, as mentioned in some guides. Instead, just install with `pip3`.

MicroPython

For MicroPython, you will not use `pip3`. Instead you can get the library from the CircuitPython bundles. See this guide page (https://adafru.it/ABU) for more information about the bundles, and also see the Libraries page on circuitPython.org (https://adafru.it/ENC).

Setup

Prerequisites:

Python 3 Installed with pip. If you do not have Python 3 installed, you can get it for free for major platforms on python.org (https://adafru.it/fa7).

Verify pip:

```
C:\Binho\adafruit>pip --version
pip 19.3.1 from c:\program files
(x86)\python38-32\lib\site-packages\pip (python
3.8)
```

Step 1: Setup Binho Nova Host Adapter hardware

The Binho Nova Multi-Protocol USB Host Adapter 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 for most modern operating systems (OS).

Certain operating systems, like Mac and Ubuntu, may require additional permissions to start using Binho Nova. In addition, Windows 7 does **not** have the standard USB CDC driver included as default.

Please check the following guide to setup permissions on Mac\Ubuntu and Windows 7 driver setup:

https://support.binho.io/user-guide/using-the-device/software-installation (https://adafru.it/HDX)

Step 2: Install the Binho Host Adapter Libraries

The following command will install the **binhoHostAdapter** Python library.

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

Verify Nova can communicate with **binhoHostAdapter** Python library:

```
C:\Binho\adafruit>python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:
23) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for
more information.
>>> from binhoHostAdapter import binhoUtilities
>>> devices =
binhoUtilities.binhoUtilities().listAvailableDevices()
>>> print(devices)

['COM8']
```

Step 3: Install Adafruit Blinka

```
pip install adafruit-blinka
```

Step 4: Set BLINKA_NOVA environment variable

In order for Adafruit blinka libraries to use Binho Nova, set the **BLINKA_NOVA** environment variable with the following command.

Windows Command line:

```
set BLINKA_NOVA=1
```

Windows Powershell:

```
$Env:BLINKA_NOVA = "1"
```

Mac/Ubuntu:

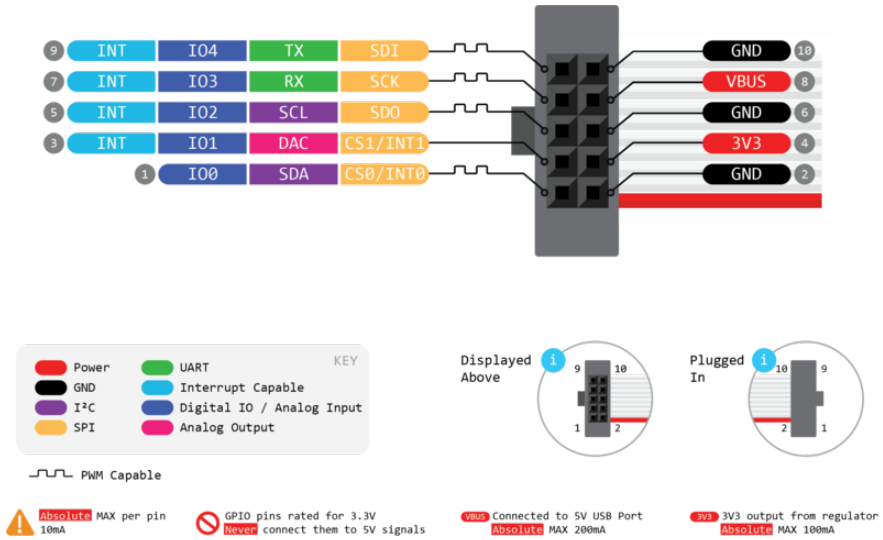
```
export BLINKA_NOVA=1
```

Verify Binho Nova's environment variable is set and the Adafruit Blinka libraries can recognize and communicate with the adapter:

```
C:\Binho\adafruit>set BLINKA_NOVA=1

C:\Binho\adafruit>python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit (I
Type "help", "copyright", "credits" or "license" for more information.
>>> import board
>>> dir(board)
['I2C', 'IO0', 'IO1', 'IO2', 'IO3', 'IO4', 'MISO', 'MOSI', 'RX', 'SCK', 'SCL',
d', 'detector', 'pin', 'sys']
>>>
```

Nova Pinout

Below you will find the pinout for the **Binho Nova** Multi-protocol host adapter. We have labelled the pinout on the Breadboard Breakout accessory that comes with each **Nova** as this is most likely how you would connect your **Nova** to any hardware project you might be working on.



Examples

The examples following this page show just how easy it is to use a Binho Nova with many of CircuitPython's readily available sensors and their Python libraries along with example code.

We will be providing examples on how to use the following CircuitPython packages:

- busio SPI to talk to a temperature/barometric pressure/humidity sensor
- busio I2C to talk to a temperature/barometric pressure/humidity sensor
- digitalio GPIO to toggle an LED
- pulseio PWM (Pulse Width Modulation) to change the brightness of an LED
- busio UART to send and receive messages from a terminal application

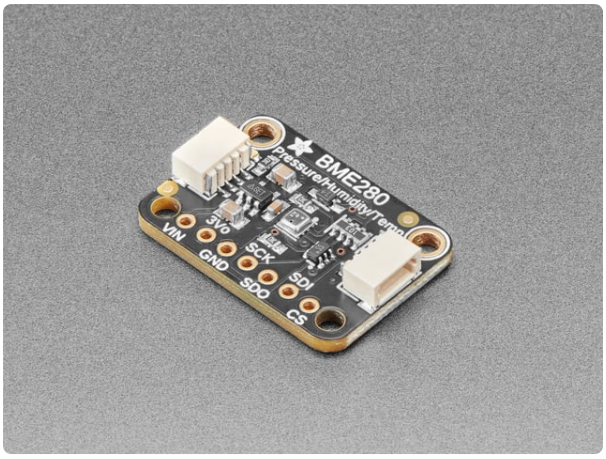
BME280 sensor

Setup

In this example, we will be using the Bosch BME280 temperature, barometric pressure, and humidity sensor which has an Adafruit CircuitPython library.

Install circuitpython bme280 python library:

```
pip install adafruit-circuitpython-bme280
```

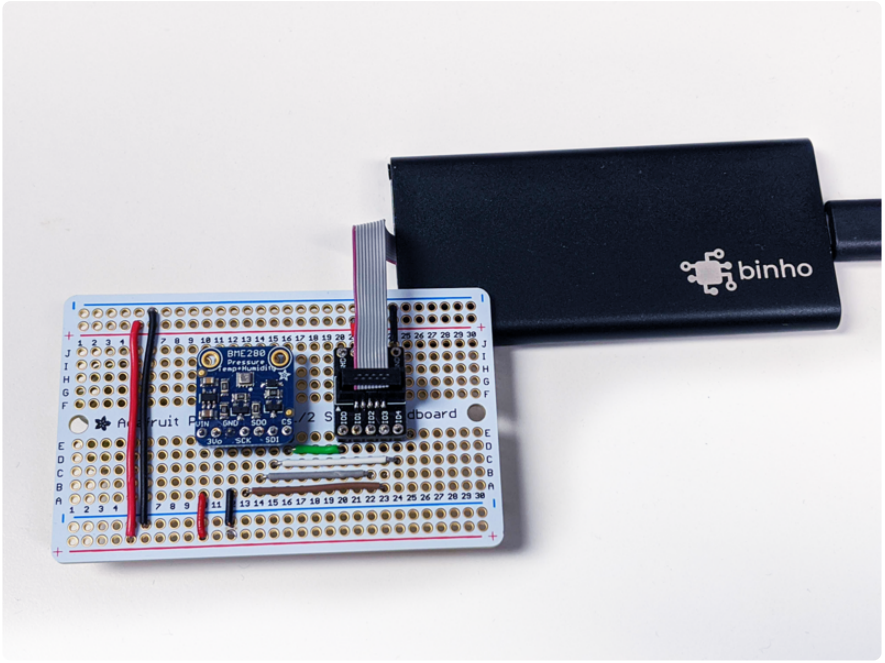


Adafruit BME280 I2C or SPI Temperature Humidity Pressure Sensor

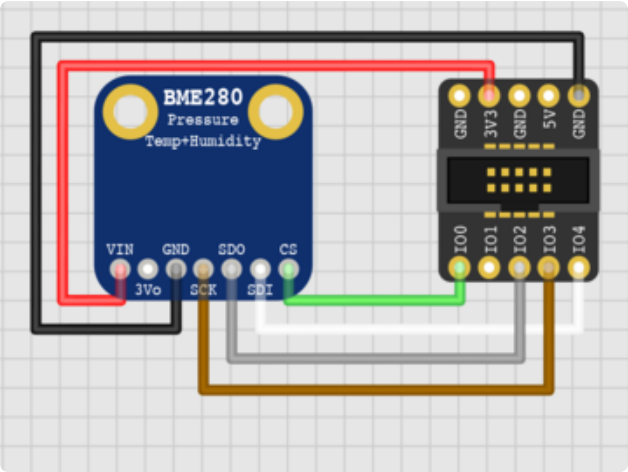
Bosch has stepped up their game with their new BME280 sensor, an environmental sensor with temperature, barometric pressure and humidity! This sensor is great for all sorts...

https://www.adafruit.com/product/2652

SPI



Setup



Pin connections from Nova to BME280

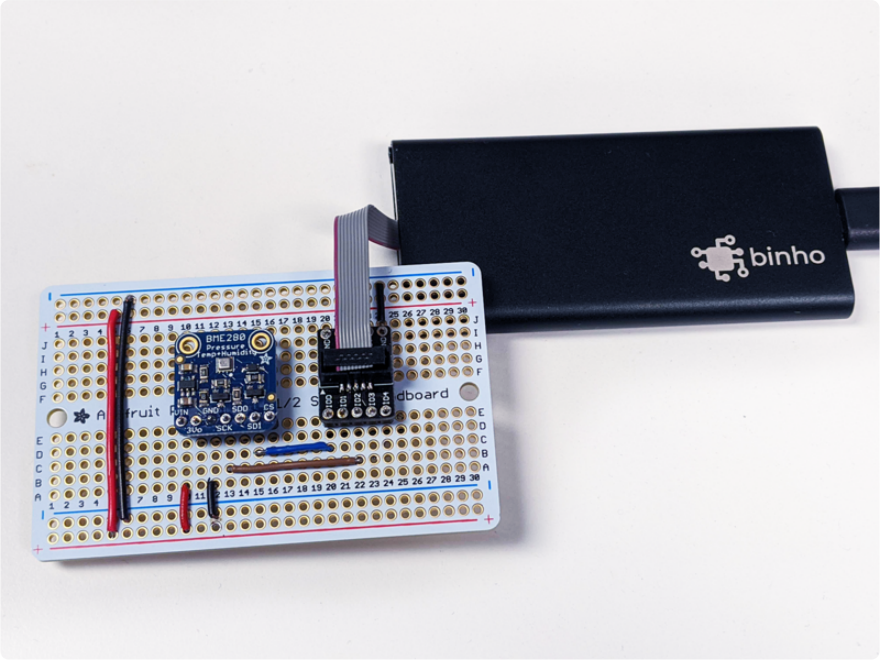
Nova **IO0** to **CS**
Nova **IO2** to **SDO**
Nova **IO3** to **SCK**
Nova **IO4** to **SDI**
Nova **3V3** to **VIN**
Nova **GND** to **GND**

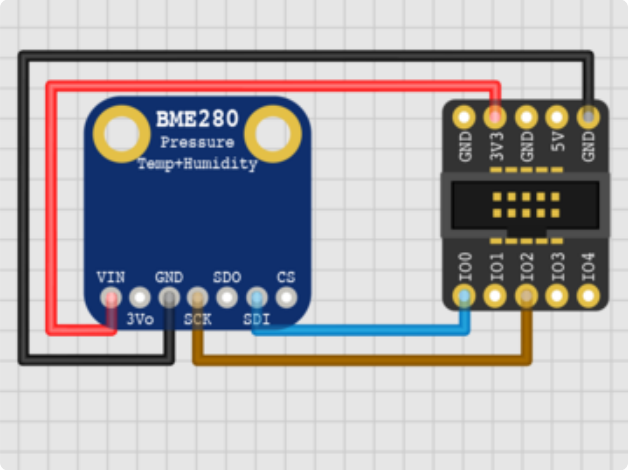
Example code:

This example uses Adafruit's **digitalio** package to create a **DigitalInOut** object for the Chip Select Pin and the **busio** package to create an **SPI** object.

```
import time
import board
import digitalio
import busio
import adafruit_bme280

# Create library object using our Bus SPI port
spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
bme_cs = digitalio.DigitalInOut(board.IO0)
bme280 = adafruit_bme280.Adafruit_BME280_SPI(spi, bme_cs)

# change this to match the location's pressure (hPa) at sea level
bme280.sea_level_pressure = 1013.25

while True:
  print("\nTemperature: %0.1f C" % bme280.temperature)
  print("Humidity: %0.1f %%" % bme280.humidity)
  print("Pressure: %0.1f hPa" % bme280.pressure)
  print("Altitude = %0.2f meters" % bme280.altitude)
  time.sleep(2)
```

I2C





Pin connections from Nova to a BME280:

Nova **IO0** to **SDI**
Nova **IO2** to **SCK**
Nova **3V3** to **VIN**
Nova **GND** to **GND**

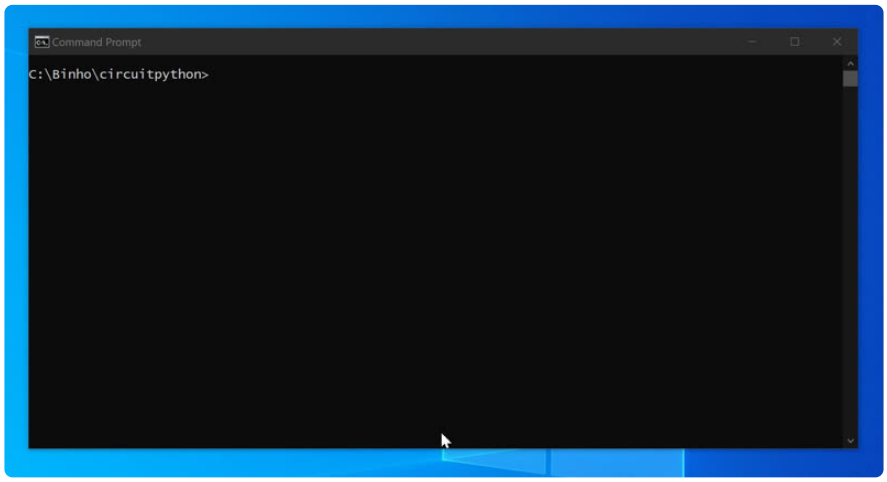
Example code

This example uses Adafruit's **busio** package to create an **I2C** object.

```
import time
import board
import busio
import adafruit_bme280

# Create library object using our Bus I2C port
i2c = busio.I2C(board.SCL, board.SDA)
bme280 = adafruit_bme280.Adafruit_BME280_I2C(i2c)

# change this to match the location's pressure (hPa) at sea level
bme280.sea_level_pressure = 1013.25
```

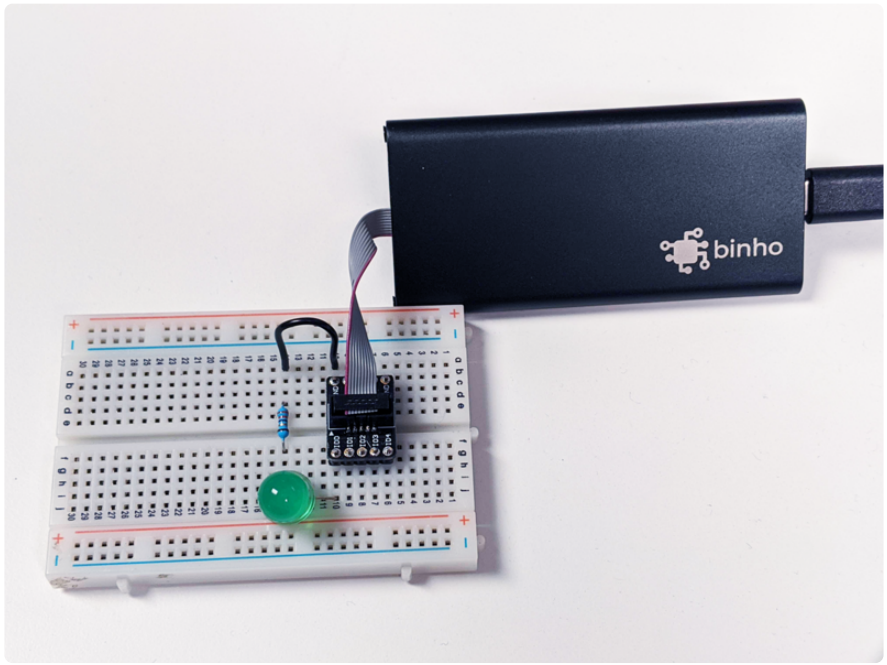
```
while True:
  print("\nTemperature: %0.1f C" % bme280.temperature)
  print("Humidity: %0.1f %%" % bme280.humidity)
  print("Pressure: %0.1f hPa" % bme280.pressure)
  print("Altitude = %0.2f meters" % bme280.altitude)
  time.sleep(2)
```

Example use



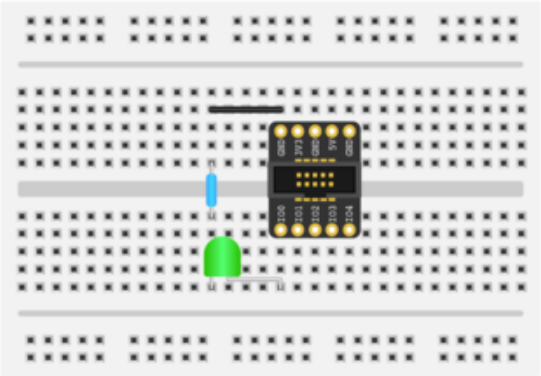
Blinking and Pulsing an LED

In this example, we show you how to complete the "Hello World" of hardware: blinking an LED. We will also show you how to PWM the LED.



Setup

Use a small resistor, about 150 ohms (give or take) and a green LED on a breadboard.



Pin connections to LED:

Nova **IO0** to LED **Anode (+)**
LED **Cathode (-)** to **Resistor**
Resistor to GND

GPIO

This example uses the Adafruit's **digitalio** package to create a **DigitalInOut** object.

The LED should blink with on and off times of a half second.

```
import time
import board
import digitalio

led = digitalio.DigitalInOut(board.IO0)
led.direction = digitalio.Direction.OUTPUT

while True:
    led.value = True
    time.sleep(0.5)
    led.value = False
    time.sleep(0.5)
```

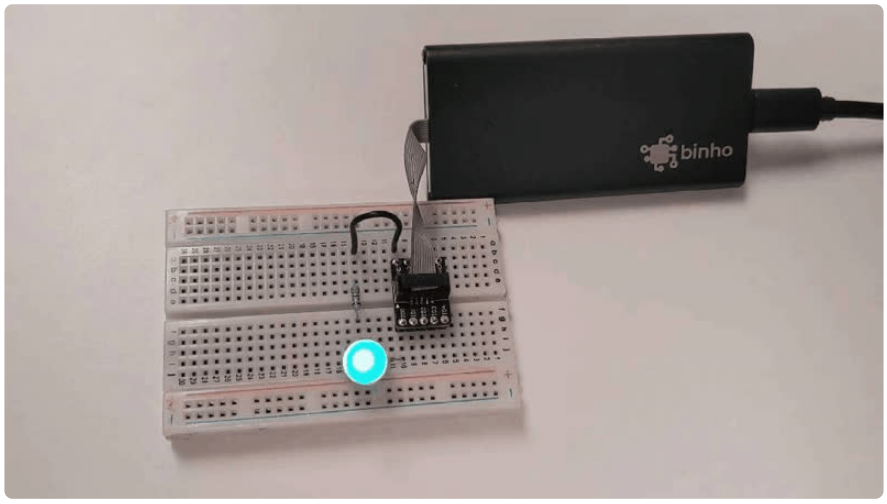
PWM

This example uses the Adafruit's **pulseio** package to create a **PWMOut** object.

```
import time
import board
import pulseio

led = pulseio.PWMOut(board.IO0, frequency=5000, duty_cycle=0)

while True:
    for i in range(100):
        # PWM LED up and down
        if i &lt; 50:
```

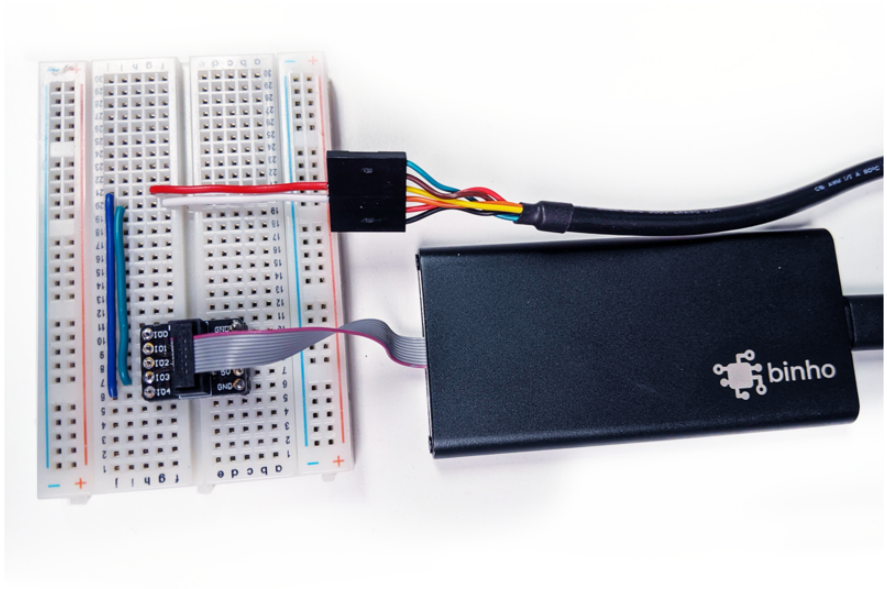
```
        # Up
        led.duty_cycle = int(i * 2 * 65535 / 100)
    else:
        # Down
        led.duty_cycle = 65535 - int((i - 50) * 2 * 65535 / 100)
    time.sleep(0.01)
```

Example demo

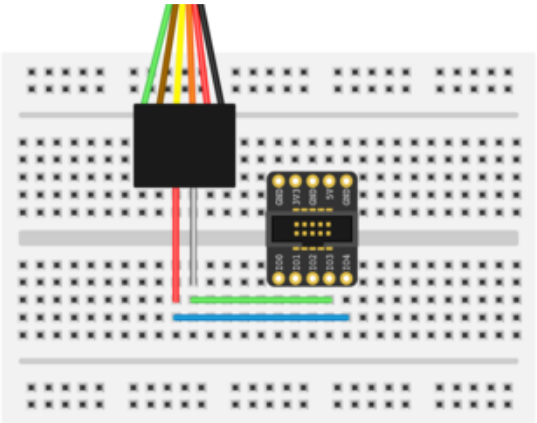


UART

The following UART example uses a TTL-234X-3V3 FTDI cable (compatible with Adafruit product FTDI Serial TTL-232 USB Cable (http://adafru.it/70)).



Setup



Pin Connections to FTDI cable:

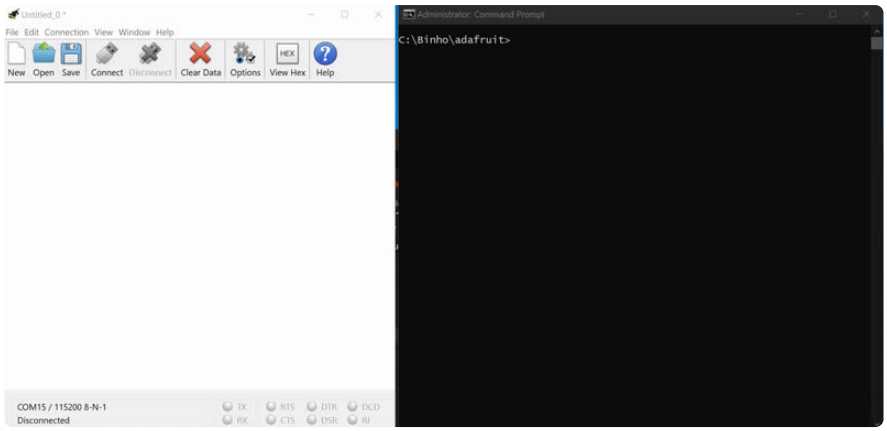
Nova **IO4** (TX) to FTDI **RX (Yellow)**
Nova **IO3** (RX) to FTDI **TX (Orange)**

FTDI Serial TTL-232 USB Cable
Just about all electronics use TTL serial for debugging, bootloading, programming, serial output, etc. But it's rare for a computer to have a serial port anymore. This is a USB to...
https://www.adafruit.com/product/70

Example Code

This example uses Adafruit's **busio** package to create a **UART** object. It will read 3 characters from the FTDI cable which CoolTerm (Windows) or other terminal emulator is connected to. The script then sends 'hello world' to the FTDI cable which will display in the terminal.

```
import board
import busio

uart = busio.UART(board.TX, board.RX, 115200, 8, None, 1, 1000)
data = uart.read(3)
# convert bytearray to string
data_string = ''.join([chr(b) for b in data])
print(data_string, end="")
uart.write('hello world')
uart.deinit()
```


Untitled_0 *
File Edit Connection View Window Help
New Open Save Connect Disconnect Clear Data Options HEX View Hex Help
COM15 / 115200 8-N-1
Disconnected
TX RTS DTR DCD
RX CTS DSR RI
Administrator: Command Prompt
C:\Binho\adafruit>