CircuitPython BLE Libraries on Any Computer

Created by Dan Halbert

https://learn.adafruit.com/circuitpython-ble-libraries-on-any-computer

Last updated on 2024-06-03 03:12:48 PM EDT

Table of Contents

Overview

Adafruit CircuitPython supports using Bluetooth Low Energy (BLE) to communicate wirelessly with BLE devices, phones, tablets, and with other CircuitPython boards. Adafruit provides many libraries to make this easy and to support specific devices.

Now you can use those same libraries (or write your own) on any host computer--Windows, Mac, or Linux--that has BLE hardware. Most modern computers, especially laptops, already have Bluetooth hardware built in. If not, you can plug in a USB adapter such as Adafruit's Bluetooth 4.0 USB Module (http://adafru.it/1327).

The Adafruit Blinka bleio library (https://adafru.it/Od0) makes this possible. It is a regular Python library that runs on desktop Python, not on CircuitPython boards. It re-implements the `_bleio` module that is part of CircuitPython: all our BLE libraries are ultimately based on `_bleio`.

The Blinka bleio library only supports acting in a BLE central role. You can connect to peripheral devices, such as heart rate monitors, pulse oximeters, bicycle sensors, etc., but you cannot act as a peripheral yourself with this code.

The Blinka bleio library is part of the family of "Blinka" libraries that run under regular Python and implement CircuitPython functionality, including Blinka and Blinka displayio.

This guide will explain how to get Python set up on your host computer, how to install the Blinka bleio library, and then give some examples of how to use it.

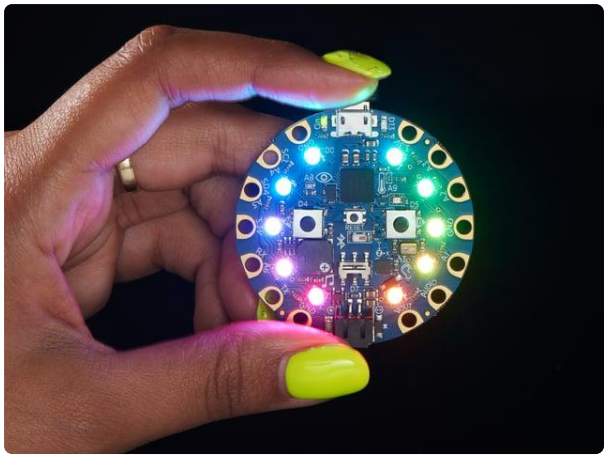
Bluetooth Capable Devices from Adafruit

Finger Pulse Oximeter with Bluetooth LE
Discontinued - you can grab the Finger Pulse Oximeter - BM1000 instead! This Finger Pulse...
https://www.adafruit.com/product/4582

Bluetooth 4.0 USB Module (v2.1 Back-Compatible)
Add Bluetooth capability to your computer super fast with a USB BT 4.0 adapter. This adapter is backwards compatible with v2.1 and earlier, but also supports the latest v4.0/Bluetooth...
https://www.adafruit.com/product/1327

Some Adafruit Microcontrollers with Bluetooth

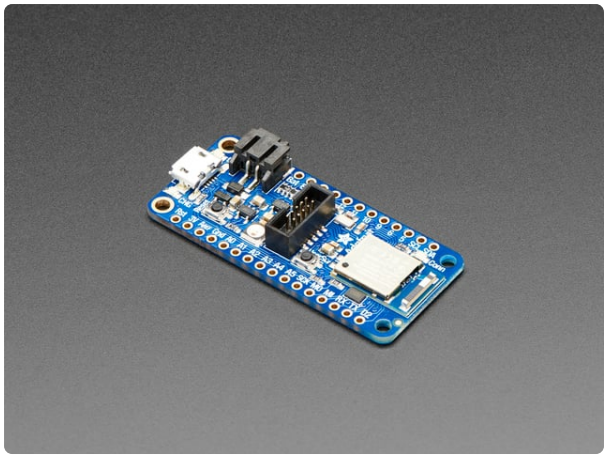
Adafruit CLUE - nRF52840 Express with Bluetooth LE
Do you feel like you just don't have a CLUE? Well, we can help with that - get a CLUE here at Adafruit by picking up this sensor-packed development board. We wanted to build some...
https://www.adafruit.com/product/4500



Circuit Playground Bluefruit - Bluetooth Low Energy

Circuit Playground Bluefruit is our third board in the Circuit Playground series, another step towards a perfect introduction to electronics and programming. We've...

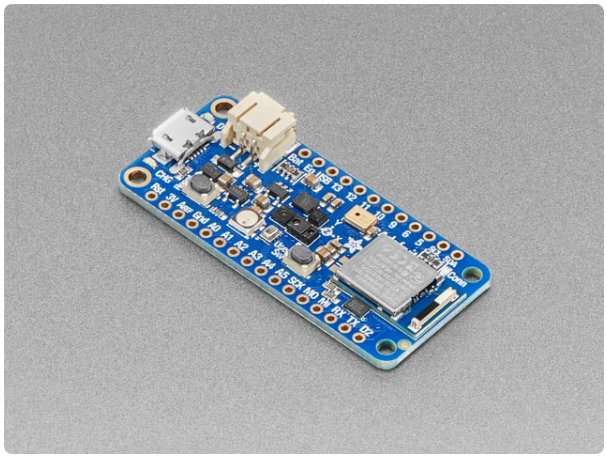
https://www.adafruit.com/product/4333



Adafruit Feather nRF52840 Express

The Adafruit Feather nRF52840 Express is the new Feather family member with Bluetooth Low Energy and native USB support featuring the nRF52840! It's...

https://www.adafruit.com/product/4062



Adafruit Feather nRF52840 Sense

The Adafruit Feather Bluefruit Sense takes our popular Feather nRF52840 Express and adds a smorgasbord of sensors...

https://www.adafruit.com/product/4516

Install Python on Your Host Computer

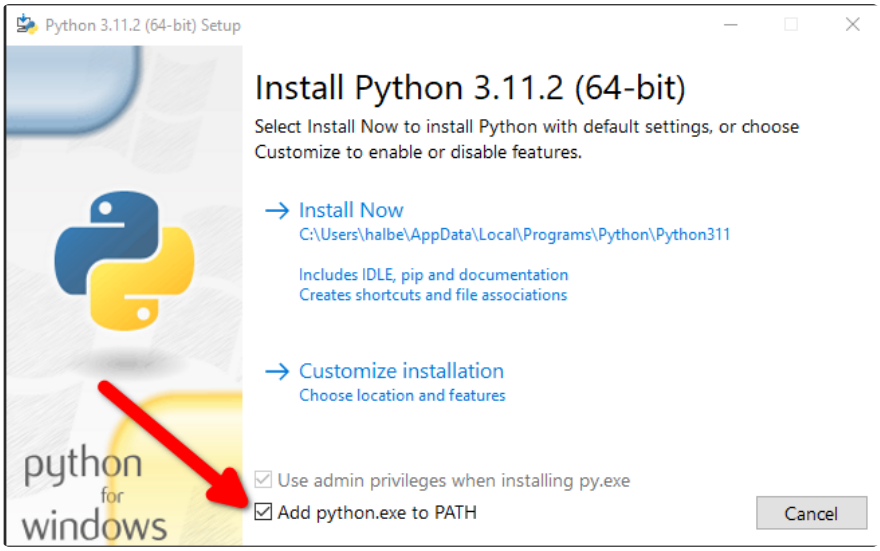
You Need Python 3 and `pip3`

To run CircuitPython BLE libraries on a host computer, you'll need at least Python 3.9, and you'll need the `pip3` program to install the libraries. You may have `pip` already, but often the `pip` (not pip3) command installs software for Python 2, so make sure you are using `pip3`.

Windows 10

You can install Python from https://python.org, (https://adafru.it/Od1)or from the Windows Store. See our guide Using Python on WIndows 10 (https://adafru.it/Od2) for an easy way to get Python installed.

If you don't use the Windows Store version, make sure you check the box to add Python to your PATH when you run the installer. See the screenshot below.



Starting Python 3 on Windows

Depending on how you install Python 3, the command to start it in a command shell in Windows can be different.

- Installed from the https://python.org (https://adafru.it/Od3) download:
  - Works: `python` (`python.exe`)
  - Works: `py` (`py.exe`, the Python launcher).
  - Does not work: `python3` (`python3.exe`). You'll get directed to the Windows Store to install Python 3 yet again.
- Installed from the Windows Store:
  - Works: `python` (`python.exe`)
  - Does not work: `py` (the Python launcher is not installed).
  - Works: `python3` (`python3.exe`).

These differences can be confusing; see this page (https://adafru.it/Od4) for detailed documentation.

macOS

Modern macOS comes with Python 3, but it may be an older version. Instead of using the system-supplied version, we recommend that you use the Homebrew (https://adafru.it/wPC) system to install a more recent version of Python 3 and keep it up to date. The Homebrew installation will not interfere with the system-installed Python, and does not replace it.

This article (https://adafru.it/Od5) describes in detail various ways of managing Python on macOS. It's worth reading to understand the issues, and to see various ways of managing multiple versions of Python.

Bluetooth Permissions

A user has reported that, at least as of macOS Big Sur, you must add your terminal application to the Bluetooth Privacy Settings in **System Preferences** > **Security & Privacy** > **Privacy** > **Bluetooth**.

Linux

Modern versions of Linux always come with Python. They may include both Python 2 and Python 3. See if the version supplied with your Linux distribution is at least 3.9. If not, your distribution may allow you to install additional versions that do not interfere with the original system-supplied version. If you can upgrade your distribution, considering doing so. For instance, Ubuntu 22.04 comes with Python 3.10.

Make sure you have `pip3` installed as well, by trying to run it. If it's not installed, install it in the appropriate way for your Linux distribution. For instance, for Ubuntu and Debian, do:

```
sudo apt install python3-pip
```

Raspberry Pi OS

Raspberry Pi OS (the new name for Raspian), which is based on Debian Linux, also comes with Python. The latest version as of this writing (April 2023), bullseye, comes with Python 3.9, invoked with the `python3` command.

If you are using Raspberry Pi OS Lite, it may not come with `pip3`. You'll need to install it by doing:

```
sudo apt install python3-pip
```

Do not use sudo when running pip3.

Do Not Use `sudo` When Running `pip3`

You may see Internet advice to install libraries using `sudo pip3` on Linux or MacOS. This is in general a bad idea, because you can damage the libraries that the underlying system depends on, and in general end up trashing your system in mysterious ways. Always install using `pip3` without `sudo`. If your `pip3` is old, you may need to specify `pip3 --user`, but these days `--user` is often the default.

Here's a detailed discussion (https://adafru.it/Od6) of why `sudo pip3` is a bad idea.

Install BLE Libraries

Once you have installed Python 3 and `pip3`, you are ready to install the Blinka bleio library and the base CircuitPython BLE library. In your shell, enter this command to install both:

```
pip3 install --upgrade adafruit-blinka-bleio adafruit-circuitpython-ble
```

The `--upgrade` will ensure that you get the latest versions, even if either library was previously installed. The `adafruit-blinka-bleio` library depends on a number of other libraries, which will be installed automatically, and upgraded if necessary.

Linux and Raspberry Pi: Add User to `bluetooth` Group

On Linux, including on Raspberry Pi, you must also add your user to the `bluetooth` group. Reboot after doing this to ensure your user is added to the group and also to ensure that `~/.local/bin` is added to your path (after doing `pip3` above). To add yourself to the group, do this:

```
sudo usermod -a -G bluetooth $USER
sudo reboot
```

Raspberry Pi 3B+ and 4B Firmware Fix

Raspberry Pi **3B**+ and **4B** boards use different hardware for BLE and WiFi than Raspberry Pi **3B** and Pi **Zero W** boards. There was a bug (https://adafru.it/Oda) in the firmware for the 3B+ and 4B boards, in the `bluez-firmware` package. `bluez-firmware-1.2-4+rpt2` worked, but versions `rpt3`, `rpt4`, `rpt5`, and `rpt6` did not. Make sure your version is at least `rpt8`, as `rpt8` includes a security fix not present in `rpt7`.

```
$ apt list bluez-firmware
Listing... Done
bluez-firmware/testing,now 1.2-4+rpt8 all [installed,automatic]
N: There is 1 additional version. Please use the '-a' switch to see it
```

If you need to upgrade bluez-firmware, do:

```
$ sudo apt upgrade bluez-firmware
```

Pulse Oximeter, Heart Rate Monitor, and BBQ Thermometer

Once the base BLE libraries are installed, you can install helper libraries for various third-party BLE peripherals, and access them directly. We'll show examples of peripherals accessed from several different host computers.

Pulse Oximeter

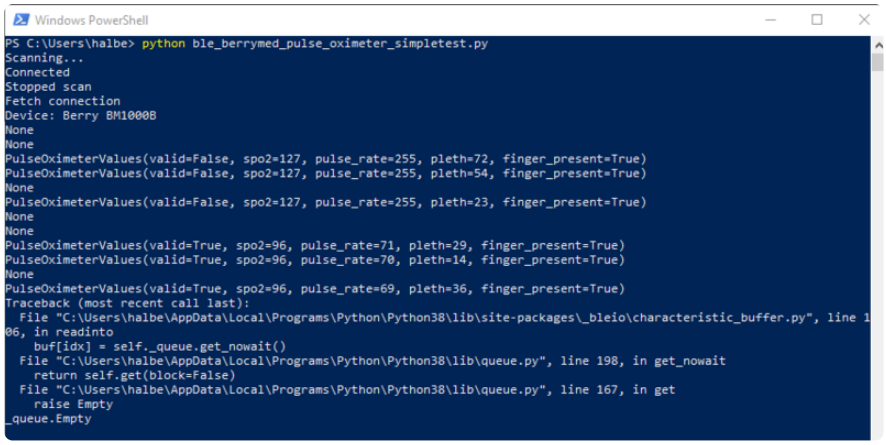
This example uses the BerryMed Pulse Oximeter (http://adafru.it/4582) sold by Adafruit. Install the pulse oximeter library by typing the `pip3` command below into your shell. Then download the example program below.

```
pip3 install adafruit-circuitpython-ble-berrymed-pulse-oximeter
```

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
Read data from a BerryMed pulse oximeter, model BM1000C, BM1000E, etc.
"""

# Protocol defined here:
#     https://github.com/zh2x/BCI_Protocol
# Thanks as well to:
#     https://github.com/ehborisov/BerryMed-Pulse-Oximeter-tool
#     https://github.com/ScheindorfHyenetics/berrymedBluetoothOxymeter
#
# The sensor updates the readings at 100Hz.

import _bleio
import adafruit_ble
from adafruit_ble.advertising.standard import Advertisement
from adafruit_ble.services.standard.device_info import DeviceInfoService
from adafruit_ble_berrymed_pulse_oximeter import BerryMedPulseOximeterService

# CircuitPython <6 uses its own ConnectionError type. So, is it if available.
Otherwise,
# the built in ConnectionError is used.
connection_error = ConnectionError
if hasattr(_bleio, "ConnectionError"):
    connection_error = _bleio.ConnectionError

# PyLint can't find BLERadio for some reason so special case it here.
ble = adafruit_ble.BLERadio()  # pylint: disable=no-member

pulse_ox_connection = None

while True:
    print("Scanning...")
    for adv in ble.start_scan(Advertisement, timeout=5):
        name = adv.complete_name
        if not name:
            continue
        # "BerryMed" devices may have trailing nulls on their name.
        if name.strip("\x00") == "BerryMed":
```

```
            pulse_ox_connection = ble.connect(adv)
            print("Connected")
            break

    # Stop scanning whether or not we are connected.
    ble.stop_scan()
    print("Stopped scan")

    try:
        if pulse_ox_connection and pulse_ox_connection.connected:
            print("Fetch connection")
            if DeviceInfoService in pulse_ox_connection:
                dis = pulse_ox_connection[DeviceInfoService]
                try:
                    manufacturer = dis.manufacturer
                except AttributeError:
                    manufacturer = "(Manufacturer Not specified)"
                try:
                    model_number = dis.model_number
                except AttributeError:
                    model_number = "(Model number not specified)"
                print("Device:", manufacturer, model_number)
            else:
                print("No device information")
            pulse_ox_service = pulse_ox_connection[BerryMedPulseOximeterService]
            while pulse_ox_connection.connected:
                print(pulse_ox_service.values)
    except connection_error:
        try:
            pulse_ox_connection.disconnect()
        except connection_error:
            pass
        pulse_ox_connection = None
```

Run the example program, and turn on the pulse oximeter, reading from your finger. Here's a screenshot from running it on Windows, in PowerShell. In this example, I typed ctrl-C after getting a few readings.



Heart Rate Monitor

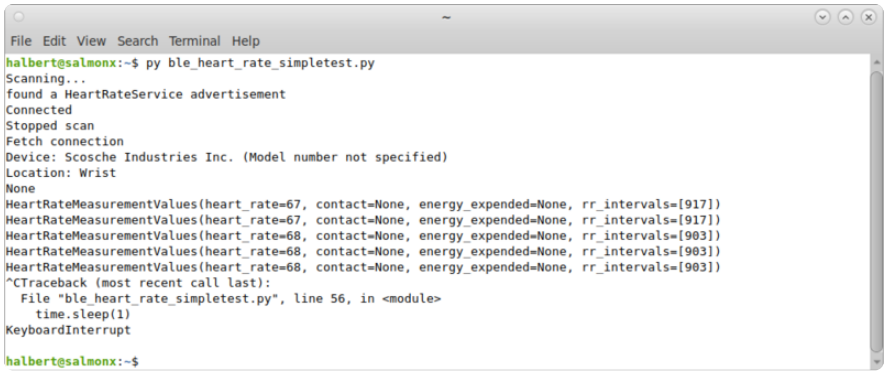
This is a Schoshe Heart Rate Monitor, which transmits data using the standard BLE Heart Rate Monitor service. Install the heart rate library by typing the `pip3` command below into your shell. Then download the example program below.

```
pip3 install adafruit-circuitpython-ble-heart-rate
```

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
Read heart rate data from a heart rate peripheral using the standard BLE
Heart Rate service.
"""

import time

import adafruit_ble
from adafruit_ble.advertising.standard import ProvideServicesAdvertisement
from adafruit_ble.services.standard.device_info import DeviceInfoService
from adafruit_ble_heart_rate import HeartRateService

# PyLint can't find BLERadio for some reason so special case it here.
ble = adafruit_ble.BLERadio()  # pylint: disable=no-member

hr_connection = None

while True:
    print("Scanning...")
    for adv in ble.start_scan(ProvideServicesAdvertisement, timeout=5):
        if HeartRateService in adv.services:
            print("found a HeartRateService advertisement")
            hr_connection = ble.connect(adv)
            print("Connected")
            break

    # Stop scanning whether or not we are connected.
    ble.stop_scan()
    print("Stopped scan")

    if hr_connection and hr_connection.connected:
        print("Fetch connection")
        if DeviceInfoService in hr_connection:
            dis = hr_connection[DeviceInfoService]
            try:
                manufacturer = dis.manufacturer
            except AttributeError:
                manufacturer = "(Manufacturer Not specified)"
```

```
            try:
                model_number = dis.model_number
            except AttributeError:
                model_number = "(Model number not specified)"
            print("Device:", manufacturer, model_number)
        else:
            print("No device information")
        hr_service = hr_connection[HeartRateService]
        print("Location:", hr_service.location)
        while hr_connection.connected:
            print(hr_service.measurement_values)
            time.sleep(1)
```

Turn on your heart rate monitor, and run the example program. reading from your finger. Here's a screenshot from running it on Ubuntu Linux. In this example, I typed ctrl-C after getting a few readings.



iBBQ Thermometer

Here's a BLE-enabled food thermometer, which can take two probes. It and similar models are readily available under the "nutrichef" brand name, and identify as "iBBQ" in their BLE advertisements.

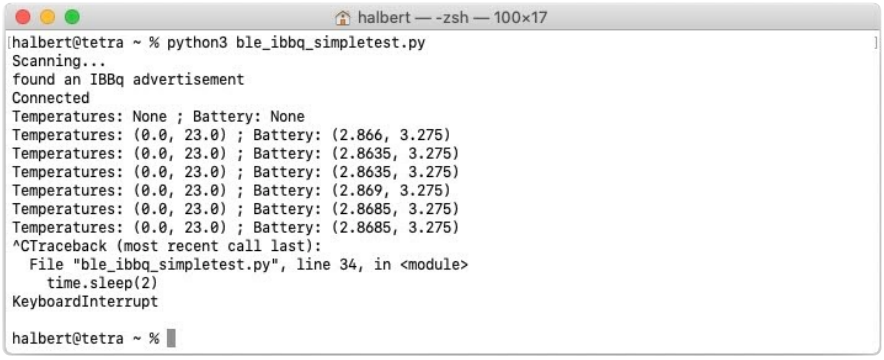
Install the iBBQ library by typing the pip3 command below into your shell. Then download the example program below.

```
pip3 install adafruit-circuitpython-ble-ibbq
```

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import time

import adafruit_ble
from adafruit_ble.advertising.standard import ProvideServicesAdvertisement
```

```
from adafruit_ble_ibbq import IBBQService

# PyLint can't find BLERadio for some reason so special case it here.
ble = adafruit_ble.BLERadio()  # pylint: disable=no-member

ibbq_connection = None

while True:
    print("Scanning...")
    for adv in ble.start_scan(ProvideServicesAdvertisement, timeout=5):
        if IBBQService in adv.services:
            print("found an IBBq advertisement")
            ibbq_connection = ble.connect(adv)
            print("Connected")
            break

    # Stop scanning whether or not we are connected.
    ble.stop_scan()

    if ibbq_connection and ibbq_connection.connected:
        ibbq_service = ibbq_connection[IBBQService]
        ibbq_service.init()
        while ibbq_connection.connected:
            print(
                "Temperatures:",
                ibbq_service.temperatures,
                "; Battery:",
                ibbq_service.battery_level,
            )
            time.sleep(2)
```

Turn on your thermometer, and run the example program. Here's a screenshot from running it on MacOS. In this example, I typed ctrl-C after getting a few readings.



BLE UART

One simple way of communicating between two BLE devices is to use a simulated "UART". A UART provides a bi-directional byte stream, so that both ends of a connection can transmit and receive bytes with each other.

There are standard BLE UART services, such as the Nordic UART Service (NUS). The Adafruit Bluefruit Connect (https://adafru.it/F-x) app uses NUS to talk to BLE boards.

Once you get the UART service working, it's easy to invent your own ad hoc protocol that sends and receives commands and data over the serial stream.

BLE UART Python `eval()` Example

Here's a simple example that uses BLE UART to send a text string from a host computer to a CircuitPython board over BLE. The board calls the Python function `eval()` on the string, to evaluate it as a Python expression, and sends the result back as a string to the host computer. For instance, the host might send `2+2`, and the board will send back `4`.

To try this example, first install this library from the latest library bundle (https://adafru.it/ABU) on your BLE-capable CircuitPython board, such as a Feather nRF52840 or a Circuit Playground Bluefruit:

- **adafruit_ble**

Then copy the program below to **CIRCUITPY** on your CircuitPython board as **code.py**:

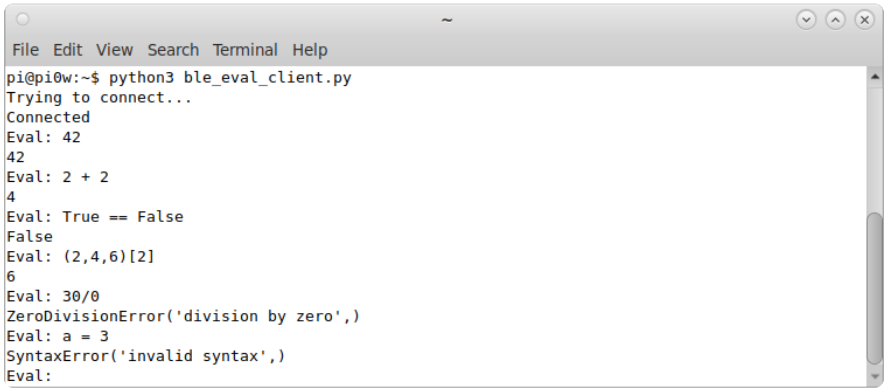
```
# SPDX-FileCopyrightText: 2020 Dan Halbert for Adafruit Industries
#
# SPDX-License-Identifier: MIT

# Provide an "eval()" service over BLE UART.

from adafruit_ble import BLERadio
from adafruit_ble.advertising.standard import ProvideServicesAdvertisement
from adafruit_ble.services.nordic import UARTService

ble = BLERadio()
uart = UARTService()
advertisement = ProvideServicesAdvertisement(uart)

while True:
    ble.start_advertising(advertisement)
    print("Waiting to connect")
    while not ble.connected:
        pass
    print("Connected")
    while ble.connected:
        s = uart.readline()
        if s:
            try:
                result = str(eval(s))
            except Exception as e:
                result = repr(e)
            uart.write(result.encode("utf-8"))
```

Now copy the second program to your host computer, and run it. Wait for it to connect to your board, and then type some Python expressions at the **Eval:** prompt.

```
# SPDX-FileCopyrightText: 2020 Dan Halbert for Adafruit Industries
#
# SPDX-License-Identifier: MIT

# Connect to an "eval()" service over BLE UART.

from adafruit_ble import BLERadio
from adafruit_ble.advertising.standard import ProvideServicesAdvertisement
from adafruit_ble.services.nordic import UARTService

ble = BLERadio()

uart_connection = None

while True:
    if not uart_connection:
        print("Trying to connect...")
        for adv in ble.start_scan(ProvideServicesAdvertisement):
            if UARTService in adv.services:
                uart_connection = ble.connect(adv)
                print("Connected")
                break
        ble.stop_scan()

    if uart_connection and uart_connection.connected:
        uart_service = uart_connection[UARTService]
        while uart_connection.connected:
            s = input("Eval: ")
            uart_service.write(s.encode("utf-8"))
            uart_service.write(b'\n')
            print(uart_service.readline().decode("utf-8"))
```

Here's an example of running the **ble_eval_client.py** program on a Raspberry Pi Zero W, talking to a Circuit Playground Bluefruit.

Note that errors, like division by zero, are caught and reported. Also note you can only type Python expressions, not statements. So `a = 3` doesn't work.



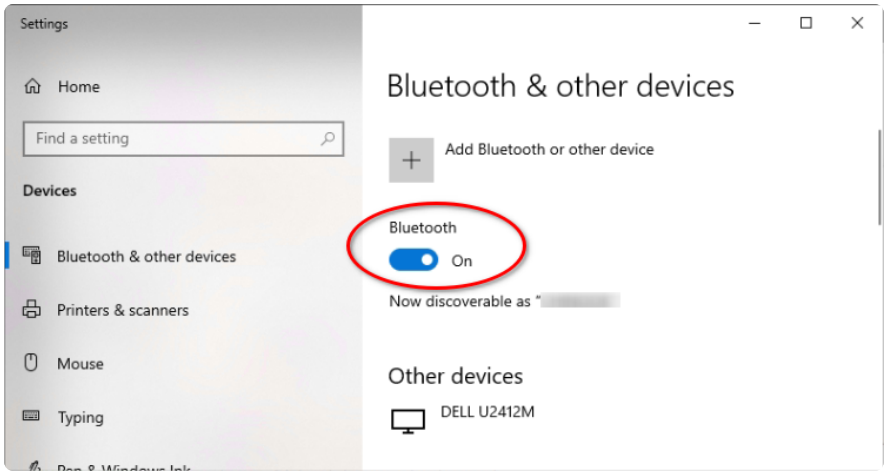
Troubleshooting

Reset Bluetooth

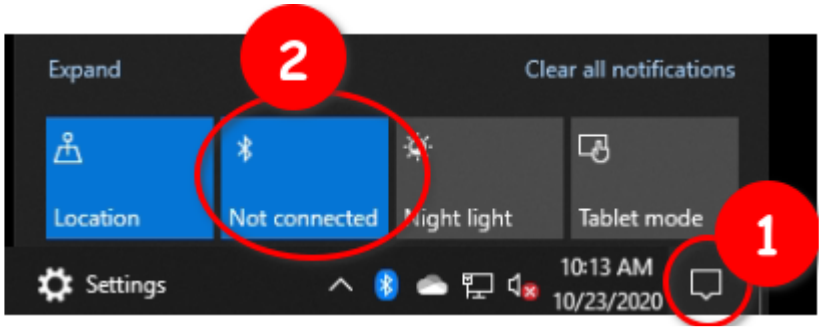
Sometimes the Bluetooth system software on the host computer becomes stuck, confused, or in a bad state. This can cause Blinka bleio programs to stop working. Often a simple fix is just to turn Bluetooth off and then back on, on the host computer.

Windows

There are two ways to cycle Bluetooth off and on in Windows. In Settings, find the Bluetooth setting page, and turn Bluetooth off and then back on:

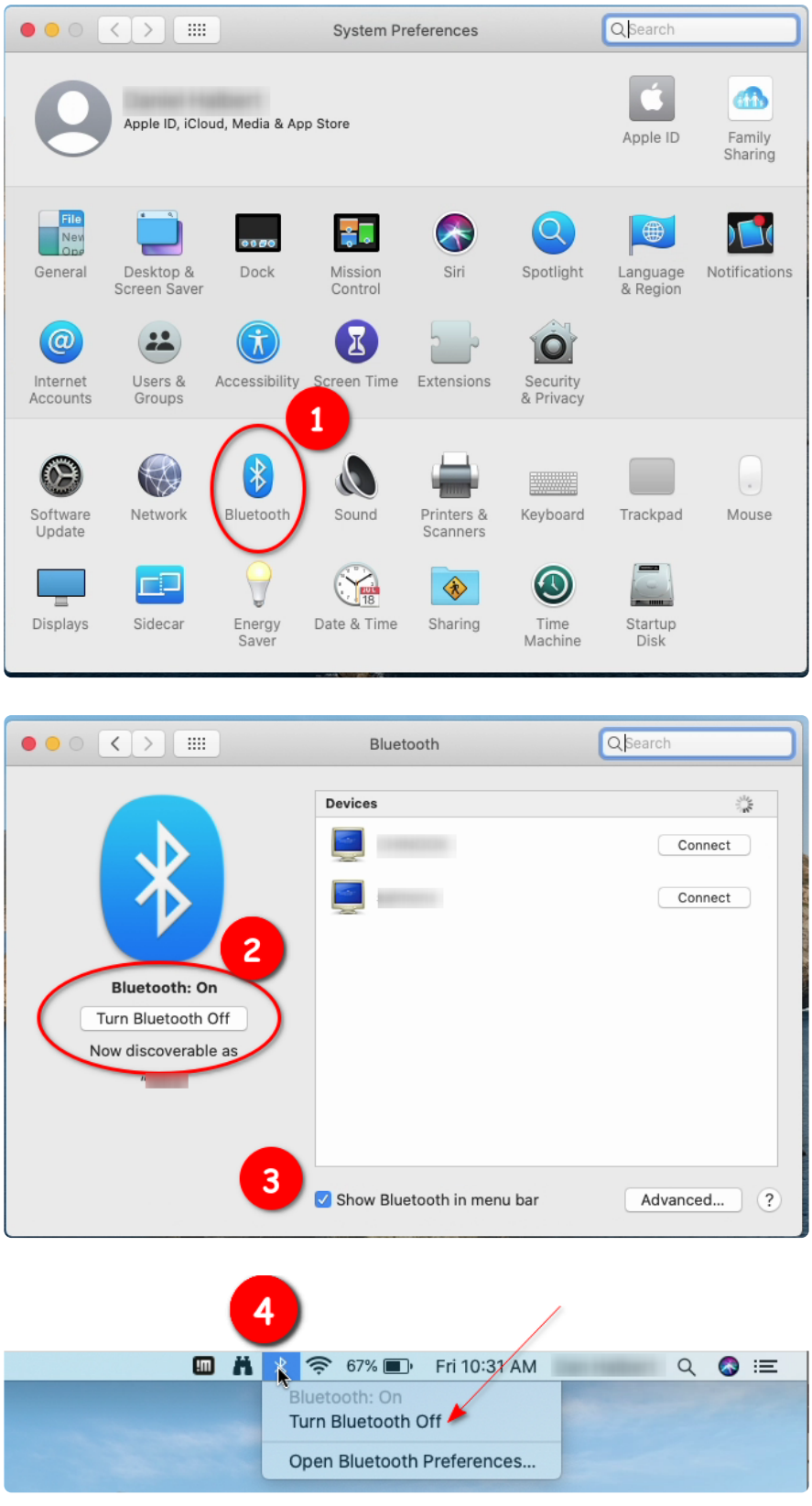


Or, select the Notification area by clicking the Notification icon in the taskbar (1), and then toggle Bluetooth off and then back on (2):



macOS

Go to System Preferences, and choose Bluetooth (1). Then toggle Bluetooth off and on (2). If you check the box that says "Show Bluetooth in menu bar" (3), you can toggle Bluetooth more quickly (4).



Linux and Raspberry Pi

Depending on which Linux distribution and desktop software you're using, there might be a Bluetooth item in the task bar or menu bar for your desktop that will allow to you cycle Bluetooth off and back on easily. For instance, the default Raspberry Pi

desktop has a Bluetooth icon in the top menu bar; you can turn Bluetooth off and then back on from there.

If you don't have a desktop icon to use, you can cycle Bluetooth from the command line. For Raspberry Pi OS, Debian, and Ubuntu, type these commands into your shell:

```
rfkill block bluetooth
rfkill unblock bluetooth
```

You may find it convenient to create a script to do this. For instance, you can put these commands into a file called `btcycle`, make it executable, and put it in some directory on your PATH:

```
#!/bin/bash
# Cycle Bluetooth on and off to reset it.
rfkill block bluetooth
rfkill unblock bluetooth
 # Wait for bluetooth to come back on. You may need to change this delay.
sleep 1.5
```