Sending and Receiving data to and from a device with Bluetooth Low Energy (BLE) via usb bluetooth adapter using the bleak module for python.

By: David Krljanovic, Shabbir Yusufali, Shawn Han, Jeffrey Zhu
Last Updated: April 13, 2025

This file will guide the user through:

1. Choosing and testing hardware for BLE (and bluetooth) communications
2. Installing python3 and the required libraries
3. Writing programs for Sending and receiving Data
4. Easily copying python files to your NFS directory.

Table of Contents:

Formatting

1. Commands for the host's linux console are shown as:
   `(host)$ echo "Hello PC world!"`
2. Commands for the target's linux console are shown as:
   `(byai)$ echo "Hello embedded world!"`

1 – Hardware

1.1 – Recommended Hardware

The Beagle Y-AI is advertised to have included a bluetooth and BLE module on-board. This module is known to have widespread issues, and we were unable to find a way to drive it properly. Instead, we recommend that you purchase a separate bluetooth adapter capable of BLE communication. We used the Asus BT-500 bluetooth adapter, and this tutorial will assume you are using this adapter or a similar adapter (they *should* all work the same).

1.2 - Testing bluetooth module functionality.

a) First install the required tech stack onto the target:

```
(byai)$ sudo apt update
(byai)$ sudo apt install bluez libglib2.0-0 dbus
```

b) Check whether the adapter is able to scan for devices using bluetoothctl:

```
(byai)$ bluetoothctl scan
```

i) If this command hangs with no output, then your bluetoothctl is not detecting any usable bluetooth module.

ii) if no output, try plugging and unplugging the adapter, rebooting the board, or running bluetoothctl enable and bluetoothctl power on

c) Attempt to pair to a bluetooth or BLE device by entering the mac address of the device you wish to pair to:

```
(byai)$ bluetoothctl pair 12:34:56:78:90:AB
```

d) Troubleshooting:

i) If you're seeing something like this:

```
(byai)[bluetooth]#
```

Not to worry! You've simply entered the bluetoothctl shell. You can gracefully exit by entering `exit`. You can also continue to enter bluetoothctl commands here without including the `bluetoothctl` before them. Try running `help` in this shell for a list of commands.

2 – Software

2.1 – Installing required libraries

a) If you haven't already, install python3:

```
(byai)$ sudo apt install python3
```

i) Validate the installation by running

```
(byai)$ python3 --version
```

You should see "python 3.x.x"

b) Install bleak using sudo apt (pip/pip3 install probably won't work!)

```
(byai)$ sudo apt install python3-bleak
```

2.2 – Program for Sending Data

Your main codebase is most likely written in C, or C++. In order to allow this python file to accept input from the rest of your project, you need to use pipes. This file receives commands from a pipe, and sends them over BLE to its destination. Some variables have placeholder data as examples. **TIP**: do not send data too rapidly across this pipe; most pipes use a buffer, don't drown it! A 10 ms delay between pipe sends should be enough.

```
import asyncio
import os
import sys
from bleak import BleakClient
# REPLACE WITH YOUR DESTINATION ADDRESS
DESTINATION_MAC_ADDRESS = str(sys.argv[1])

# EXAMPLE SERVICE AND CHARACTERISTIC_UUIDS (think of these like they're "sockets"!)
SERVICE_UUID = "180C"
CHARACTERISTIC_UUID = "00002a56-0000-1000-8000-00805f9b34fb"
PIPE_PATH = "/tmp/pipeName"

# create/replace pipe.
if os.path.exists(PIPE_PATH):
   os.remove(PIPE_PATH)
os.mkfifo(PIPE_PATH)
```

```
async def send_ble_commands():
     async with BleakClient(DESTINATION_MAC_ADDRESS) as client:
       if not await client.is_connected():
           print(f"BLE.py: Failed to connect to {DESTINATION_MAC_ADDRESS}")
           return

       print(f"BLE.py: Connected to {DESTINATION_MAC_ADDRESS}")

       with open(PIPE_PATH, "r") as pipe:
           while True:

               command = pipe.readline().strip()  # Read from named pipe

               if not command:
                   continue

               if command.lower() == "exit":
                   print("Exiting...")
                   break

               # Write gatt char is an asynchronus function! You need the await!
               # Or else the function won't run to completion (incomplete send!).
               await client.write_gatt_char(CHARACTERISTIC_UUID, command.encode())

asyncio.run(send_ble_commands())
```

2.3 – Program for receiving Data

This is a separate .py file that receives commands. It again uses pipes to communicate with the rest of your C/C++ program.

```
import asyncio
import os
import sys
from bleak import BleakClient

# REPLACE WITH YOUR SOURCE'S ADDRESS
SOURCE_MAC_ADDRESS = str(sys.argv[1])


# EXAMPLE CHARACTERISTICS (think of these like they're "sockets"!)
SERVICE_UUID = "180C"
READING_CHARACTERISTIC_UUID = "00002a57-0000-1000-8000-00805f9b34fb"
```

```
PIPE_PATH = "/tmp/your_pipe"

# create/replace pipe.
if os.path.exists(PIPE_PATH):
   os.remove(PIPE_PATH)
os.mkfifo(PIPE_PATH)

async def send_ble_commands():
   async with BleakClient(SOURCE_MAC_ADDRESS) as client:
       if not await client.is_connected():
           print(f"BLE.py: Failed to connect to {SOURCE_MAC_ADDRESS}")
           return

       print(f"BLE.py: Connected to {SOURCE_MAC_ADDRESS}")

       with open(PIPE_PATH, "w") as pipe:
           while True:

               data = await client.read_gatt_char(READING_CHARACTERISTIC_UUID)
               decoded = data.decode().strip()
               if decoded:
                   pipe.write(decoded)
                   pipe.flush()

asyncio.run(send_ble_commands())
```

2.4 – Quality of life file copying

You may notice that, when you compile the bulk of your project using cmake, that the .py file *does not* transfer to your cmpt433/public or /mnt/remote/myApps (or equivlent directory). You can automate copying to your NFS directory by allowing your project to copy the python file every time you build the project code. Assuming that your python file is **in the root directory of your project folder**, add this command to the end of CMakeLists.txt **within the /app directory** of your project:

```
# Copy the BLE.py folder to NFS
add_custom_command(TARGET your_project POST_BUILD
 COMMAND "${CMAKE_COMMAND}" -E copy
    "${CMAKE_SOURCE_DIR}/BLE.py"
    "~/cmpt433/public/myApps/BLE.py"
 COMMENT "Copying BLE.py to public NFS directory")
```