Lab 6

Intro to BLE on the Adafruit Feather Bluefruit Sense

**Part 1 - Use feather as BLE scanner**

First we need to move a couple more modules into our \lib folder for use by CP. You need to check your \lib folder on the Feather sense for these modules. If you do not have them, see your instructor to get copies.

- adafruit_ble
- adafruit_ble_adafruit
- adafruit_bluefruit_connect

Once you are sure you have the modules, open the Mu editor in CP mode and copy the following code. Install it onto the Feather Sense as code.py and see what it does.

```
# ble_scanner.py
# scans for nearby BLE devices and display advertisements

from adafruit_ble import BLERadio

ble = BLERadio()
print("scanning")
found = set()
scan_responses = set()
for advertisement in ble.start_scan():
    addr = advertisement.address
    if advertisement.scan_response and addr not in scan_responses:
        scan_responses.add(addr)
    elif not advertisement.scan_response and addr not in found:
        found.add(addr)
    else:
        continue
    print(addr, advertisement)
    print("\t" + repr(advertisement))
    print()

print("scan done")
```

List one of the advertisements here:

Is the Feather sense acting as a BLE peripheral or central here?

***Show your code working on the Feather to your instructor.***

Save your ble_scanner code as lastname_lab6_part1.py. Add header information and add inline comments as you study the code.

**Part 2 - Send commands to the Feather Sense with Bluefruit LE Connect app**

Adafruit has provided a free phone app that we can use to connect to their BLE devices. Bluefruit LE Connect is available for both Ios and Android devices. At least one of your lab partners should download the app to their phone for use in this lab.

Bluefruit LE Connect will act as the software on one side. We need to program some CP code for the Feather Sense side.

Open a new programming tab in the Mu editor and copy the following code:

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

from adafruit_bluefruit_connect.packet import Packet
from adafruit_bluefruit_connect.button_packet import ButtonPacket

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

while True:
   ble.start_advertising(advertisement)
   while not ble.connected:
       pass

   # Now we're connected

   while ble.connected:
       if uart.in_waiting:
           packet = Packet.from_stream(uart)
           if isinstance(packet, ButtonPacket):
               if packet.pressed:
                   if packet.button == ButtonPacket.BUTTON_1:
                       # The 1 button was pressed.
                       print("1 button pressed!")
                   elif packet.button == ButtonPacket.UP:
                       # The UP button was pressed.
                       print("UP button pressed!")

   # If we got here, we lost the connection. Go up to the top and start
   # advertising again and waiting for a connection.
```

Save the code to the Feather Sense as code.py and then try to connect to the Feather with the Bluefruit app. Ask your instructor for help! You want to navigate to the control pad of the Bluefruit connect app and to hit the buttons.

What happens when you are connected and hitting the buttons? Which buttons cause a response?

Is the Feather sense acting as a BLE peripheral or central here?

**MODIFY YOUR CODE** - now you are going to add to your CP code so the Feather Sense will respond to all the buttons on the app control pad. Add a corresponding button pressed response for each of the buttons. Use your computational thinking skills to figure out how! **Show your code working on the Feather to your instructor.**

Add inline comments to your code and save it as lastname_lab6_part2.py.

**Part 3 - control the Feather's RED_LED and NEOPIXEL with phone control pad**

a. Modify your CP code to turn the RED_LED on and off. **Show your instructor your working application.**

b. Modify your CP code to turn the NEOPIXEL red, green, blue or white. **Show your instructor your working application.**

Add inline comments and save your modified code as lastname_lab6_part3.py

**Part 4 - control Feather's NEOPIXEL with the Color Picker**

Open a new Mu editor window and copy the following sample code:

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

# CircuitPython NeoPixel Color Picker Example

import board
import neopixel
```

```
from adafruit_bluefruit_connect.packet import Packet
from adafruit_bluefruit_connect.color_packet import ColorPacket

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

ble = BLERadio()
# enter a name for your ble device so you find it easily
ble.name = "TEST_FEATHER"
uart_service = UARTService()
advertisement = ProvideServicesAdvertisement(uart_service)

pixels = neopixel.NeoPixel(board.NEOPIXEL, 1, brightness=0.1)
current_color = (0,0,0)
pixels.fill(current_color)

while True:
   # Advertise when not connected.
   ble.start_advertising(advertisement)
   while not ble.connected:
       pass
   ble.stop_advertising()

   while ble.connected:
       if uart_service.in_waiting:
           packet = Packet.from_stream(uart_service)
           if isinstance(packet, ColorPacket):
               print(packet.color)
               pixels.fill(packet.color)
```

First thing you should notice is the name change for the Feather in the Bluefruit Connect app. Change the name on line 13 to something unique for your lab group. Connect to the feather with the Bluefruit connect app and go to the color picker. Use it to change the color of the NeoPixel.

Now combine some ideas and code from Part 2,3 and 4. Use the up arrow to turn the NeoPixel on and the down arrow to turn the NeoPixel off. The Color Picker should still change the color of the NeoPixel whenever it is on and it should always turn on with the previous color.

**Part 5 - control external NEOPIXELs with the Color Picker**

In your code from Part 4, create another NeoPixel object "pixels_external". This time instead of board.NEOPIXEL, use board.D5 when creating the object. Also create pixels_external to be 25 pixels long instead of 1.

Your code from Part 4 should still work as before but you should now also be controlling pixels_external at the same time as pixels. Ask your instructor to bring the external pixels to you for testing.