Lab 10
WiFi connected Adafruit Feather Bluefruit Sense
using Adafruit IO

Part 1 - Prepare your AdafruitIO account then connect with HTTP

Login to your AIO account and confirm you still have the three feeds created in Lab 9. There should be a feed called random_value, a feed called tx and a feed called output.

Create a new dashboard - Lab 10. On the dashboard create two new blocks. The first block will be a Stream block to show the random_value feed. The second block will be a Line Chart. It will also visualize the random_value feed.

Copy the following code into a new Mu editor tab and save it to the Feather Sense as code.py.

```
import time
import board
import busio
import random
from digitalio import DigitalInOut
import neopixel
from adafruit_esp32spi import adafruit_esp32spi
from adafruit_esp32spi import adafruit_esp32spi_wifimanager

# Get wifi details and more from a secrets.py file
try:
    from secrets import secrets
except ImportError:
    print("WiFi secrets are kept in secrets.py, please add them there!")
    raise

# AirLift Featherwing
esp32_cs = DigitalInOut(board.D13)
esp32_ready = DigitalInOut(board.D11)
esp32_reset = DigitalInOut(board.D12)

spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
esp = adafruit_esp32spi.ESP_SPIcontrol(spi, esp32_cs, esp32_ready,
esp32_reset)

status_light = neopixel.NeoPixel(
    board.NEOPIXEL, 1, brightness=0.2
)

wifi = adafruit_esp32spi_wifimanager.ESPSPI_WiFiManager(esp, secrets,
status_light)

print("*"*60)
print("\tFeather Airlift HTTP to AIO test")
print("*"*60)

while True:
    try:
        print("Posting data...", end="")
        data = random.randint(25,50)
```

```
        feed = "random-value"
        payload = {"value": data}
        response = wifi.post(
            "https://io.adafruit.com/api/v2/"
            + secrets["aio_username"]
            + "/feeds/"
            + feed
            + "/data",
            json=payload,
            headers={"X-AIO-KEY": secrets["aio_key"]},
        )
        print(response.json())
        response.close()
        print("OK")
    except (ValueError, RuntimeError) as e:
        print("Failed to get data, retrying\n", e)
        wifi.reset()
        continue
    response = None
    time.sleep(15)
```

The code does not run, right? You should have gotten a “KeyError”. This is because the program is trying to communicate with AIO and we need to provide a little more information. Remember when you used the Bluefruit App to act as a gateway to AIO, you had to set up the Bluefruit app with your AIO username and your AIO key. For this program to work you will add those values to your secrets.py file.

Add the following 2 lines to the dictionary in the secrets.py file and save the file.

```
'aio_username' : 'your_AIO_username',

'aio_key' : 'your_AIO_KEY',
```

The code on the Feather Sense should run now. Check the Lab 10 dashboard in AIO to confirm AIO is receiving the random values being sent.

What feed does the CP program try to use? (Check line 42 of code.py)

AIO has specific naming conventions. Even though your feed is called random_value on AIO, when communicating with AIO the feed is called random-value instead. If you select the random_value feed from the feeds menu on AIO and then select the Feed Info settings in the upper right hand corner, you will see the “Key” that AIO expects to use for your feed.

Show your instructor your data being collected on AIO.

Save your code as lastname_lab10_part1.py. Modify header information and inline comments of your code.

Part 2 - HTTP GET AND POST

In part 1, wifi.post was used to send information to AIO. In part 2 you will modify your code from part 1 to include a GET from AIO.

The wifi.post and wifi.get methods are almost identical. Add the following to your code after the try/except for the wifi.post:

```
try:
    print("Getting data...", end="")
    feed = "output"
    response = wifi.get(
        "https://io.adafruit.com/api/v2/"
        + secrets["aio_username"]
        + "/feeds/"
        + feed,
        headers={"X-AIO-KEY": secrets["aio_key"]},
    )
    print(response.json())
    response.close()
    print("OK")
except (ValueError, RuntimeError) as e:
    print("Failed to get data, retrying\n", e)
    wifi.reset()
    continue
response = None
```

Your code should now use a wifi.get from the output feed. Printing the response as a JSON shows a large dictionary-like data structure. The output feed's value is in there. Here is how to get the last value saved to the feed.

Modify the line:

```
print(response.json())
```

to now say:

```
print(response.json().get('last_value'))
```

Go to the AIO output feed and manually enter a new value. Check to make sure the new value is being read by the Feather Sense.

NOTE: we could also print the last_value using the following:

```
print(response.json()['last_value'])
```

Show your instructor your code GETting the output feed from AIO.

Save your code as lastname_lab10_part2.py. Modify header information and inline comments of your code.

Part 3 - Duplicate Lab 8 Part 4

In Lab 8 Part 4 the Feather Sense was programmed to recognize a “PAUSE” command from the AIO output feed. The “PAUSE” output was triggered by an Action in AIO. This Action should still be working. In this part of lab 10, use your knowledge of HTTP Get and Post to recreate the functionality of Lab 8 Part 4.

The Feather Sense should be sending random data values between 25 and 50 to AIO every 10 seconds. If the Feather GETs a new “PAUSE” command on the output feed, it should stop sending data points and instead blink the neopixel for 1 minute.

HINT: Monitor the “updated_at” value while also checking the output feeds “last_value”.

Show your instructor your working code. Make sure to manually add a PAUSE to trigger the delay in the Feather if necessary.. ☐

Save your code as lastname_lab10_part3.py. Modify header information and inline comments of your code.

Part 4 - Using the Adafruit IO_HTTP module

In part 4 you will use code from Adafruit’s module written specifically to communicate with AIO. Time to add another module to the lib folder on the Feather Sense. Your instructor will provide you with the following module:

- **adafruit_io**

Copy the module into the lib folder on the Feather Sense and then copy the following code into a new Mu editor tab. Save it to the Feather Sense as code.py.

```
import time
import board
import busio
from digitalio import DigitalInOut, Direction
import adafruit_esp32spi.adafruit_esp32spi_socket as socket
from adafruit_esp32spi import adafruit_esp32spi
import adafruit_requests as requests
from adafruit_io.adafruit_io import IO_HTTP, AdafruitIO_RequestError

try:
    from secrets import secrets
except ImportError:
    print("WiFi secrets are kept in secrets.py, please add them there!")
    raise

# AirLift Featherwing
esp32_cs = DigitalInOut(board.D13)
esp32_ready = DigitalInOut(board.D11)
esp32_reset = DigitalInOut(board.D12)
```

```
spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
esp = adafruit_esp32spi.ESP_SPIcontrol(spi, esp32_cs, esp32_ready,
esp32_reset)

print("Connecting to AP...")
while not esp.is_connected:
    try:
        esp.connect_AP(secrets["ssid"], secrets["password"])
    except RuntimeError as e:
        print("could not connect to AP, retrying: ", e)
        continue
print("Connected to", str(esp.ssid, "utf-8"), "\tRSSI:", esp.rssi)

socket.set_interface(esp)
requests.set_socket(socket, esp)

# Set your Adafruit IO Username and Key in secrets.py
# (visit io.adafruit.com if you need to create an account,
# or if you need your Adafruit IO key.)
aio_username = secrets["aio_username"]
aio_key = secrets["aio_key"]

# Initialize an Adafruit IO HTTP API object
io = IO_HTTP(aio_username, aio_key, requests)

try:
    # Get the 'digital' feed from Adafruit IO
    digital_feed = io.get_feed("digital")
except AdafruitIO_RequestError:
    # If no 'digital' feed exists, create one
    digital_feed = io.create_new_feed("digital")

# Set up LED
LED = DigitalInOut(board.BLUE_LED)
LED.direction = Direction.OUTPUT

while True:
    # Get data from 'digital' feed
    print("getting data from IO...")
    feed_data = io.receive_data(digital_feed["key"])

    # Check if data is ON or OFF
    if int(feed_data["value"]) == 1:
        print("received <- ON\n")
    elif int(feed_data["value"]) == 0:
        print("received <= OFF\n")

    # Set the LED to the feed value
    LED.value = int(feed_data["value"])

    time.sleep(10)
```

Take some time and study the code. One really nice thing this code will do is create a new feed for you if it does not already exist. Until now you did not have a feed called “digital’ on AIO. Go to the AIO feeds page now. You should see the feed “digital” has been created for you. Manually enter the new value “1” to the digital feed. What happens on the Feather Sense?

You may be wondering why this program uses the “BLUE_LED” and not the “RED_LED” that you have used in the past. There is a lot happening on the Feather Sense board and sometimes the processor pins are connected to more than one thing. The RED_LED is connected to the board.D13 pin which is already being used by the ESP32 Airlift. If you tried to use it you would get an error message.

Add a new block to your Lab 10 dashboard. Add a toggle button and link it to the digital feed. Set the on value to 1 and the off value to 0. Now you should be able to “toggle” the BLUE_LED on/off from your AIO Lab 10 dashboard.

Show your instructor your working code and dashboard.

Save your code as lastname_lab10_part4.py. Modify header information and inline comments of your code

Part 5 - Duplicate Lab 10 Part 3 Using the Adafruit_IO module

Once again you will duplicate Lab 10 Part 3. Use your combined experiences to program the Feather Sense to talk to AIO using the adafruit_io module’s commands.

To send data to AIO you can use something like this:

```
response = io.send_data(random_val_feed["key"],rand_data)
```

HINT: The “response” data returned from io.receive_data() is already a python dictionary. No need to use the .json() method on the response before using the .get() method. Monitor the “updated_at” value while also checking the output feeds “last_value”.

Modify the toggle button on the Lab 10 Dashboard. Link the toggle to the output feed and have it change values between “RUN” (on) and “PAUSE” (off). You should be able to use this button to manually trigger the one minute pause in the Feather Sense. If set up correctly, the toggle button should automatically switch to off when a PAUSE action is triggered.

Show your instructor your working code and working dashboard.

Save your code as lastname_lab10_part5.py. Modify header information and inline comments of your code.