EMERGING TECHNOLOGIES I

MIS 284N | Fall 2021

Project Milestone 2

Overview: For our second milestone, we'll (1) set up a Raspberry Pi device and (2) connect the microbit step counter to a python program on the Raspberry Pi. We'll use both a serial connection via USB and a Bluetooth connection to get data from the microbit to the Raspberry Pi. The Raspberry Pi will serve a secondary purpose: it will actually be a WiFi access point that the Android device can eventually connect to. You will also use this WiFi connection to run VNC from your laptop computer to program the Raspberry Pi without any need for peripherals (e.g., keyboard, mouse, monitor) for the Raspberry Pi.

Step 1: Get your Raspberry Pi, download the VNC viewer, and follow the instructions given in class.

Step 2: Write a microbit program to send a counter back and forth between two microbit devices. Be sure to identify and resolve the problem that we encounter because everyone in the classroom is trying to do this at the same time. Do you prefer to write this in python or in javascript or in makecode? Why?

Step 3: Write a microbit program to send some dummy data from the microbit to the Raspberry Pi using a serial connection across USB. For instance, you could Extend your Milestone 1 solution so that you send your step count via USB upon a button press, which will then printed to the terminal on the Raspberry Pi.

On the Raspberry Pi side, start by following the Linux instructions provided by microbit (https://makecode.microbit.org/device/serial). You can extend this to Python using the following template:

```
import serial
s = serial.Serial(timeout=0.1)
s.baudrate = 115200
s.port = "/dev/ttyACM0"
s.open()
while True:
    data = s.readline().decode('utf-8')
    if data:
        print(data)
s.close()
```

You might have the change the above code, depending on what device node your microbit is assigned when you plug it into your Raspberry Pi. To check, try ls /dev/ttyACM* from the command line. Once you run the program on the Raspberry Pi, you should print out whatever is sent by the microbit on the screen.

Step 4: In this step, we'll get a bluetooth connection working from the microbit to the Raspberry Pi. Send an Eddystone beacon from the microbit to the Raspberry Pi. This will be easy on the microbit side but was (for us) non-trivial on the Raspberry Pi python side. We will use the aioblescan library to implement this.

Step 4a: Create the microbit program to send an Eddystone beacon. Then invoke the following command in the terminal on the Raspberry Pi to ensure that the Raspberry Pi is receiving the Eddystone beacon:

```
sudo python3 -m aioblescan -e
```

Step 4b: Write the python code to run on the Raspberry Pi. Here's some code you can start with:

```
import aioblescan as aiobs
from aioblescan.plugins import EddyStone
import asyncio

def _process_packet(data):
  ev = aiobs.HCI_Event()
  xx = ev.decode(data)
  xx = EddyStone().decode(ev)
  if xx:
    print("Google beacon: {}".format(xx))

if __name__ == '__main__':
  mydev = 0
  event_loop = asyncio.get_event_loop()
  mysocket = aiobs.create_bt_socket(mydev)
  fac = event_loop._create_connection_transport(mysocket,aiobs.BLEScanRequester,None,None)
  conn, btctrl = event_loop.run_until_complete(fac)
  btctrl.process = _process_packet
  btctrl.send_scan_request()
  try:
    event_loop.run_forever()
  except KeyboardInterrupt:
    print('keyboard interrupt')
  finally:
    print('closing event loop')
    btctrl.stop_scan_request()
    conn.close()
    event_loop.close()
```

Note: when you send an Eddystone beacon, you have to send a well-formatted URL, but you can send sensor values or other data as part of that URL. For instance, you could send a URL that looks like "http://steps?12345."

Step 5: Create a makecode program that (a) reliably counts steps and gives the user feedback on the microbit about his or her step count (this is just Milestone 1); (b) sends the step count to a Raspberry Pi program when instructed to (e.g., by a button press); and (c) prints any received step count to the terminal. Some requirements and hints:

- The microbit should *not* continuously beacon. You need to decide when to start and stop beaconing.
- You want to create a URL that is somewhat unique to your program (that is, you don't want your Raspberry Pi program to read and parse *everyone's* step counts. Just yours.

- You can create interesting string variables on the microbit. You'll want to check out the Variables section of the makecode page and the Text section.
- In the python program, you should print out some nicely formatted information. For instance, "Step count received at Saturday, October 31 at 7pm is 8935."

What to submit

Via Canvas, submit a brief writeup that includes the following information for each step:

- **Step 2**: Answer the questions embedded in Step 2. How did you solve the problem of conflicts with other groups' counters?
- **Step 3**: In your own words, what is *baud rate*?
- **Step 4**: Include the code that is your solution for the microbit. If it's makecode, a screenshot is fine.
- **Step 5**: In your own words, explain what you think is happening in the provided code. You do not need to go line by line, but you should be able to explain the overall flow of activity. The documentation for both asynchio (https://docs.python.org/3/library/asyncio.html) and for aioblescan (https://github.com/frawau/aioblescan) may be helpful. Note that we're using the _create_connection_transport function in the asynchio event_loop. It's not part of the public interface. It's a "fix" I stole from the aioblescan source code. For the purposes of this response, you can assume it behaves basically like the event_loop's create_connection function.
- **Step 6**: How did you solve the problem (not more than one paragraph)? Include a screenshot of the microbit program or the python code for the microbit and a listing of the python receiver on the Raspberry Pi.

Thursday, November 4, in office hours, demonstrate Step 6. Be prepared to explain what the code does on each side (both the makecode or python on the microbit and the python on the Raspberry Pi).