**MIS 284N: PROJECT MILESTONE 2**

**Team Members:** Yeggi Lee, Chetna Singhal, Vikram Gupta

**Step 1:**

* **Partner with another group and write a microbit program to send a counter back and forth (or around a circle). Be sure to resolve the problem of identifying your counter from another group’s counter. Can you do this with the same code running on the two microbits? With different code?**

No, because you cannot differentiate between whose steps are whose. In order to solve the problem of identifying our counter from another group’s counter, we added a string at the beginning of the counter that said “RT”. This way we could identify which counter was ours.

* **Do you prefer to write this in python or in javascript or in makecode?**

We preferred using makecode in this case as it was easier to visualize the solution. As of now, we aren’t too familiar with the individual code idiosyncrasies that the microbit has in python and javascript.

* **How did you solve the problem of conflicts with other groups’ counters?**

In order to solve the problem of conflicts, we did the following:

1. First, both devices were set to use the same channel
2. Then, our team would send a key/value pair via the radio. The key would be the passcode while the value held the message. The key ensured that we only received messages intended for us. Every time, we sent a message the key (‘potato’) was included.

When we received a message, an if statement checked if the string received was the correct key and if it was, the counter was updated. Otherwise, if the key wasn’t correct, the message was not received and count remained the same.

**Step 2:**

* **In your own words, what is baud rate?**

Baud rate is the rate at which information is transferred into a communication channel. So essentially, in this case, it is the measure of the number of bits per second that can be transmitted or received by the UART. It specifies how fast the data is being sent and can also show you how long it takes to transmit one bit. One of the more common rates is 9600 bits per second (bps); however, it can go much higher to usually about 115,200 bps. Going too fast though can lead to timing errors.

**Step 3:**

* **Provide the code that your (combined) group used for the microbit that was directly connected to the Raspberry Pi.**

let step = 0

radio.setGroup(21)

input.onGesture(Gesture.ThreeG, function () {

led.stopAnimation()

step += 2

})

input.onButtonPressed(Button.A, function () {

// send data to the other microbit

radio.sendValue("unique-key", step)

})

radio.onReceivedString(function (receivedString) {

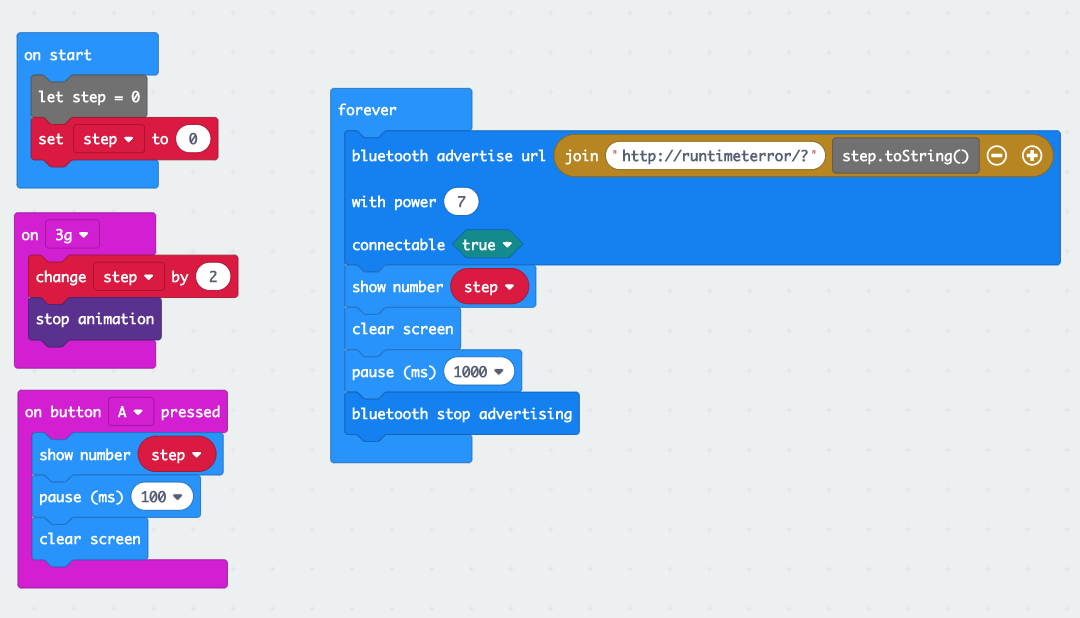
// send data to raspberry pi

serial.writeString(step.toString())

})

**Step 4:**

* **Include a screenshot of the makecode solution for the microbit.**

****

*Figure 1. Makecode Microbit Code for Using Bluetooth*

**Step 5**:

* **In your own words, explain what you think is happening in the provided code and 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.**

1. `asyncio.get\_event\_loop() creates a new event loop at the start (as there is no running loop)

2. Create a new bluetooth socket (0) for I/O operations to commence on

3. To the event loop, add a scan for communications made on that socket.

4. Have this loop run until an interrupt.

5. Whenever a URL is advertised, we process the packet. The steps involved are:

a) picking up the event

b) decoding it using the Eddystone protocol

c) pick up the URL, if one exists, and check if it has our signature

d) parse the URL and display the number of steps

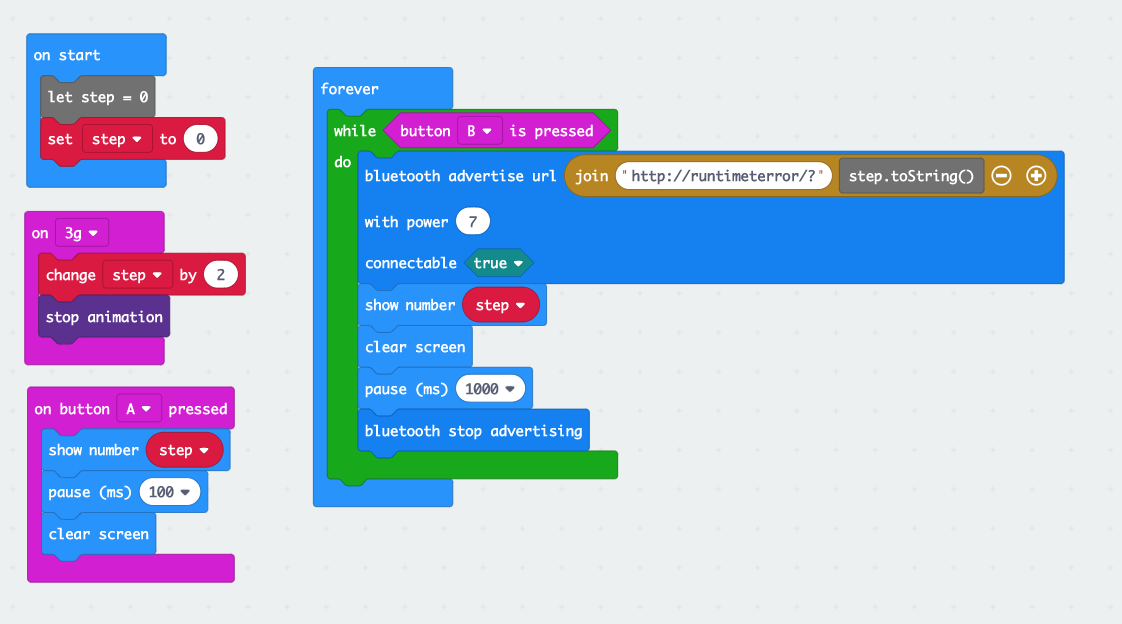
**Step 6:**

* **How did you solve the problem (not more than one paragraph)? Include a screenshot of the microbit program and a listing of the python receiver.**

Our solution to the problem included:

1. The eddystone.py file in the aioblescan package had to be corrected. It threw an error at `Itself` in line 351.
2. We chose to advertise the URL only during button press. If the button was let go, we stopped advertising (with a buffer of 100ms).
3. The url had our group name ('runtimeterror') within it. We only decoded messages advertised with that string in the URL.

The makecode portion code was done as follows:

****

*Figure 2. Makecode Microbit Program for Bluetooth Sending*

Microbit Program Javascript Code

input.onGesture(Gesture.ThreeG, function () {

step += 2

led.stopAnimation()

})

input.onButtonPressed(Button.A, function () {

basic.showNumber(step)

basic.pause(100)

basic.clearScreen()

})

let step = 0

step = 0

basic.forever(function () {

while (input.buttonIsPressed(Button.B)) {

bluetooth.advertiseUrl(

"[http://runtimeterror/?](https://slack-redir.net/link?url=http%3A%2F%2Fruntimeterror%2F%3F)" + step.toString(),

7,

true

)

basic.showNumber(step)

basic.clearScreen()

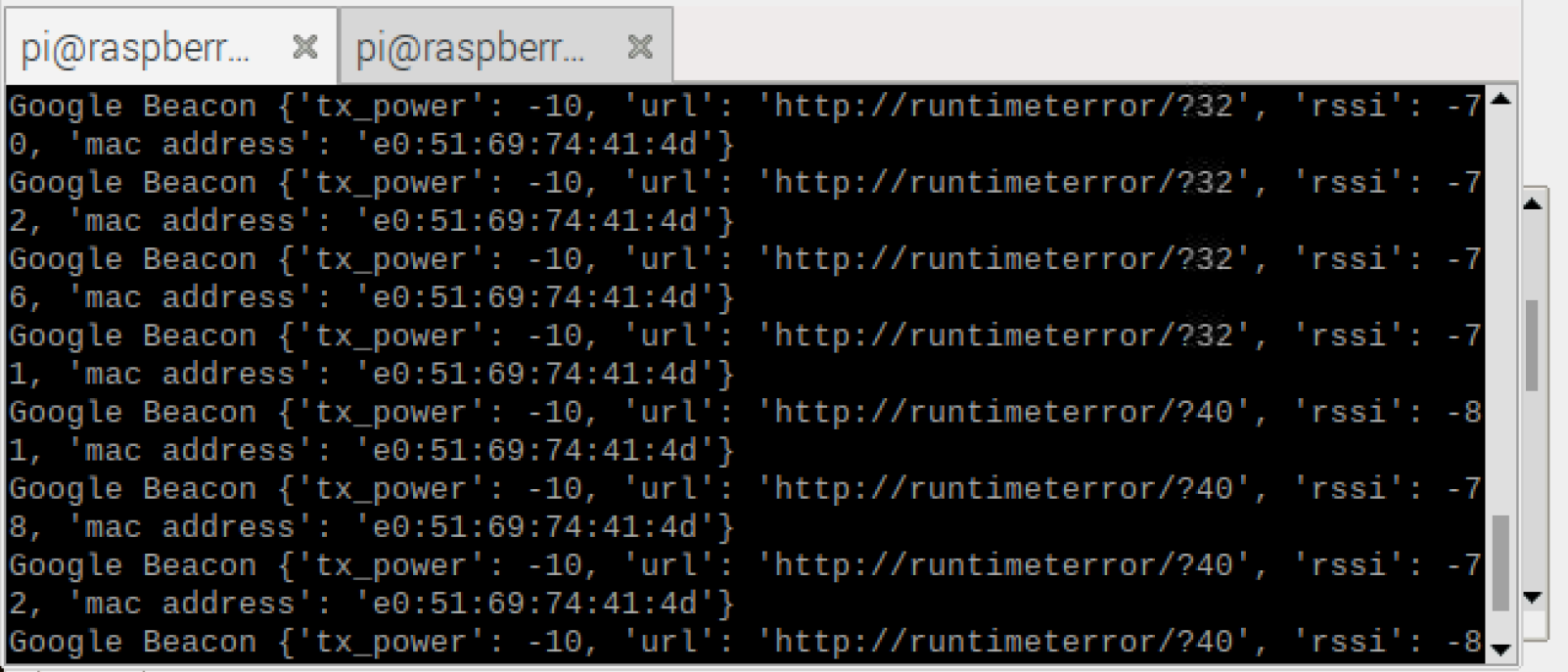
basic.pause(1000)

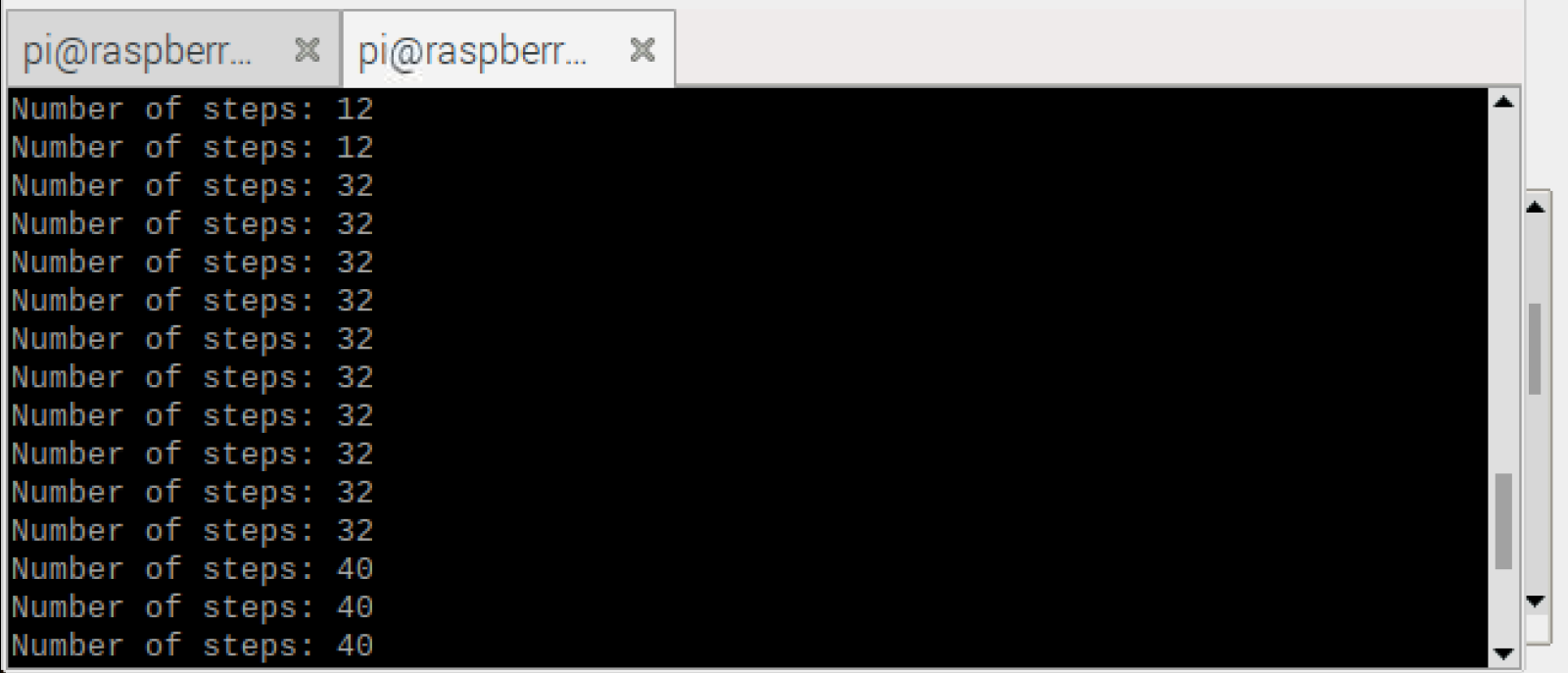
bluetooth.stopAdvertising()

}

})

Listing of the Python Receiver:





*Figure 3. Raspberry Pi - Receiver listing*

Modified Python code for Receiver (Raspberry Pi):

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 'url' in xx:

url = xx['url']

if 'runtimeterror' in url:

steps = int(url.split('?')[1])

print('Number of steps: {}'.format(steps))

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()