A. Equipment

- Hardware: Redbear, BLE Beacons, Ubertooth-One (for bluetooth sniffing)
- Software: Particle.io, (optional) Arduino IDE, wireshark

  Particle.io: https://www.particle.io

B. Background Information

Arduino

- The main programming language used for arduino device (aka RedBear) is C++
- Useful Arduino libraries for this project

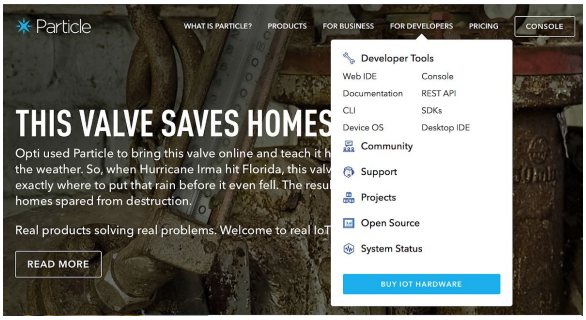
  Redbear Duo: Arduino/C/C++ Programming Reference Manual

Particle

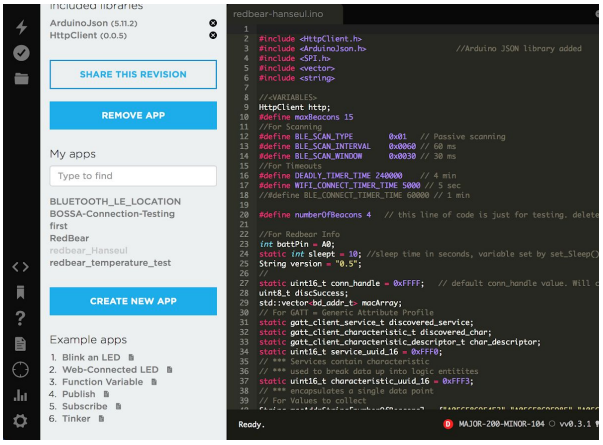
- Credential:

  iit.xrdsdaemon@gmail.com

- Particle.io was used as a server for redbears to be able to interface with BOSSA api. Complete reference manual can be found here: Particle Reference Manual.
- Basics to know:
  1. *Where to find a code on Particle.io*



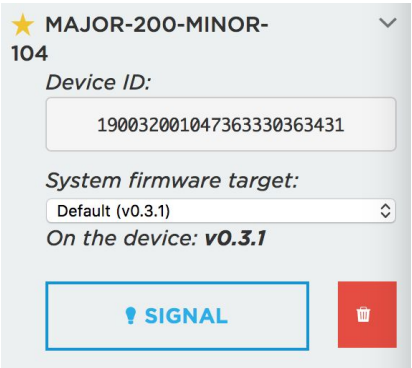
Under *FOR DEVELOPERS*, click Web IDE.



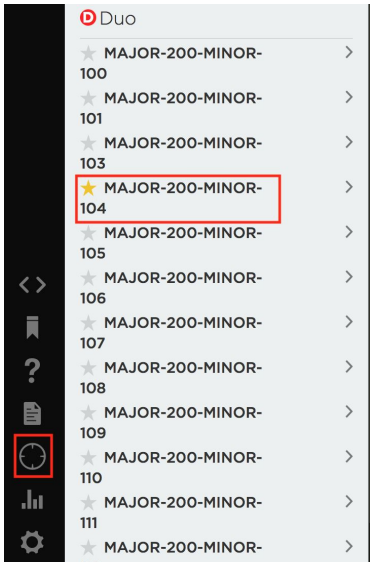
Click '<>' symbol to see all the codes under *My apps*.

2. *How to register a device (= RedBear)*
   Check *RedBear_Registration_Guide.md* in github (Click Here)

3. *How to flash a code to RedBear*



   Click Device, and double check whether device is on. To check whether the device is on or not, click the name of the device and click signal. If the device is on and registered on Particle, a RedBear will flash various random colors.



   To flash a code to specific device, make sure to click a star button next to the name of the device.

4. *Console*
   Real-time published data can be checked here. Click the graph shape to see the console.

Network Topology

A term topology refers to a way that a network is laid out. It can represents how devices are connected to each other. There are four basic topologies: mesh,

start, bus and ring. For our project, beacons and the Redbear are in *star topology*. In a star topology, each node has a dedicated point-to-point link to a central controller. All beacons are nodes (devices) and the redbear is the central controller of the beacons. In a star topology, the nodes are not directly connected to each other. Beacons, also, do not communicate with each other. A main advantage of this topology is robustness. If one link fails, that link is only affected not rest of the active links. A disadvantage of this topology is that the entire system is dependent on the central controller. If a redbear goes down, the entire system can possibly go down, or the data collected can be useless.

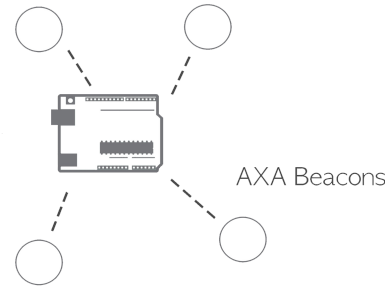


Fig. Star topology with RedBear and Beacons

Bluetooth

Bluetooth is the main technology to connect RedBear and ble beacons in this project. Bluetooth is a protocol defined by the IEEE 802.15 standard. It is considered as a wireless personal-area network (PAN). Bluetooth sniffing was done with Ubertooth and Wireshark to check the communication between RedBear and ble beacons.

C. Code & Algorithm

Algorithm Workflow

1) Redbear wakes up
2) Synchronize time of the device with particle cloud
3) deadly_timer starts
    a) (After every 10 minutes, the device is forced to sleep)
4) Receive list from BOSSA, via Particle, of MAC addresses of the beacons for which the RedBear is responsible
5) Check retained memory and print any error message/notice (By what means is retained memory populated?)
6) Connect to Wifi
    a) WifiConnectTimer starts
    b) Stop WifiConnectTimer.
    c) If the device could not connect to Wifi in 5 seconds:
        (1) *r_got_wifi* is set to false, increment *r_wifi_timer_count* by 1
        (2) Disconnect from Particle server

(3) the device is forced to sleep for 30 min.

d) If the device is connected to wifi,

(1) Retained flag *r_got_wifi* is set to true, *r_wifi_timer_count* is set to 0

7) Send RedBear config {Version, Sleeptime, Voltage} to Particle -- `RedbearUpdate`
8) Record address type of each beacon and initiate average rssi
9) Set the bluetooth scan parameters
10) Start bluetooth scanning for 30 sec
    a) Beacons are constantly broadcasting
    b) Redbear sends a request to beacons
    c) When a beacon is connected with the RedBear, execute `bleScanCallback` function
    d) Sum up rssi values and increment number of times the beacon was connected with the Redbear
    e) Record the scanned address type
11) Use the mac address from the BOSSA platform ***
    a) Convert string values to a mac address type
12) Connect to each beacon with the mac address and address type, recorded from the scan
    a) Send temp and humidity data {beacon id, temperature, humidity} when the beacons are connected and the read function is complete -- `gattReadCallback`
13) Filter rssi and send average RSSI {mac, rssi} to Particle-- `particleRssiUpdate`
14) Stop deadly timer
15) Sleep for 4 min -- `deepSleep` ***

Flowchart

D. Helpful Concepts to Understand the Code

GATT

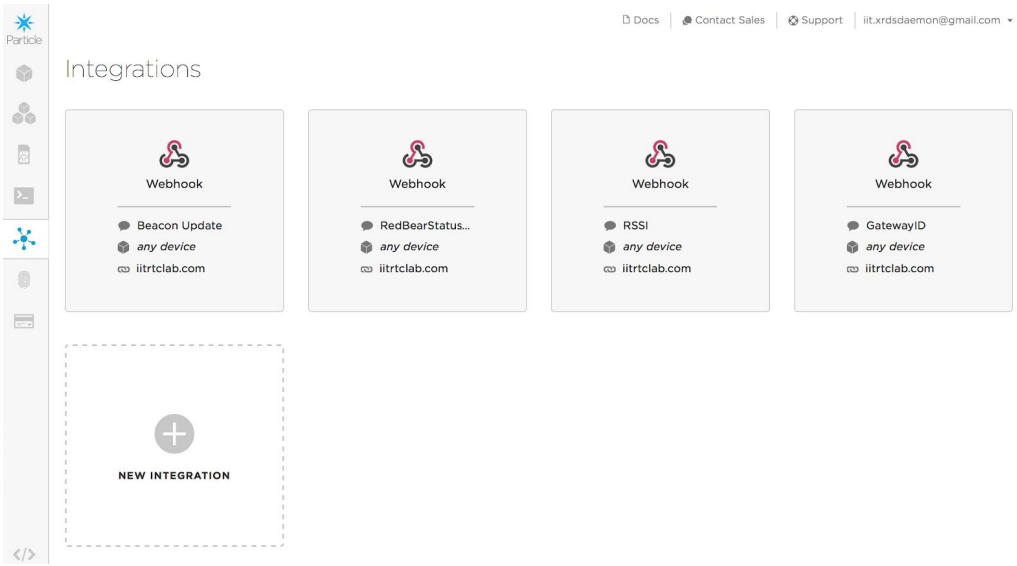
Introduction to Bluetooth Low Energy

Callback Function

A callback function pass another function as an argument. Whenever the inside function is triggered, the callback function is completed. For example, `bleScanCallback` will be triggered after `ble.startScanning`. When `bleScanCallback` is done, ble.`onScanReportCallback` will be done.

Webhook

Webhook is an HTTP callback. Webhook connects Particle cloud to the BOSSA platform.
To add a webhook on Particle.io:



Click integration to add a webhook. The event name and the variable names(fields' name) should match the one with the api. To send the data to a specific url, choose the request type to be POST. When Particle publishes specific event, it will send the data to the POST. This won't succeed if the field names do not match. To receive data through BOSSA, choose request type to be GET.

E. Bluetooth Sniffing

Ubertooth

Bluetooth sniffing was done with Ubertooth to capture bluetooth communication between RedBear and Beacons. Ubertooth can be installed in preferably Linux device.

Installation Guide
UbertoothOne bluetooth sniffing guide

There are three advertising channels for bluetooth low energy (BLE or BTLE): channel 37, 38 and 39. These three channels are used simultaneously. Which means that three computers and ubertoothes are needed to sniff the bluetooth communication.

How to capture traces:

1. Make /tmp/pipe directory (`mkfifo /tmp/pipe`) and move to the specific directory
2. Open wireshark and a terminal
    a. Wireshark setting
        i. Go to Capture-manage Interface and Pipe. Click on the + sign and add /tmp/pipe. Press enter and then save.
        ii. Make sure /tmp/pipe is highlighted.
3. On terminal type a command to start sniffing

    ```
    ubertooth-btle -f -A 37 -r library.pcapng
    ```

    37 refers to a advertising channel. Change this number to change which advertising channel to sniff. Here is a guide with more detailed explanation to write a code to sniff: Ubertooth-btle commands

4. Start to capture on wireshark
5. Check on Particle.io whether redbear receives data from beacons and push them to the particle server properly. Make sure to capture more than one full routine of beacons and redbear connection, as Ubertooth-One does not capture everything 100%.
6. Stop sniffing and wireshark capture.
7. To see only PPI protocol packets, go to Preference-> Protocols -> DLT_USER and enter btle as a protocol.
8. It is possible to filter traces captured by wireshark. Display filter reference for bluetooth low energy link layer can be found here. One of the most useful filter is:

    ```
    btle.advertising_address == (MAC address)
    ```

Crackle:
It is possible to decrypt the STK and LTK to crack data between 2 devices.

Ubertooth-One traces reveal following information:

1. Scan Request send by REDBEAR
2. Scan Response answer by AXA-Beacon
3. Connection Request initiated by the Redbear
    a. It would contain the Hopping Map
4. Connection Response by the AXA-Beacon
5. When ran in follow mode -f the Ubertooth-one should lock onto the connection, and reveal all communication exchange between the two devices.
6. ATT protocol packets should be captured which will contain the following:
    a. GATT data which reveals the properties of how the data exchange will occur.

b. GATT packet that contains the actual value for temperature and humidity readings.