CC2650 Web Demo Tutorial

It is important to note that the cc2650-web-demo [1] includes six different demos: 1) a UDP UART, 2) a 6lbr client, 3) a CoAP server, 4) an MQTT client (IBM Quickstart), 5) a web server (HTTP), and 6) a Bluetooth Smart beacon. This paper will only discuss the 6lbr client, the MQTT client (IBM Quickstart), and the web server HTTP demo.

Figure 9 illustrates the overall system architecture. We have a WSN that collects sensor data and forwards these data to the cloud through a border router.

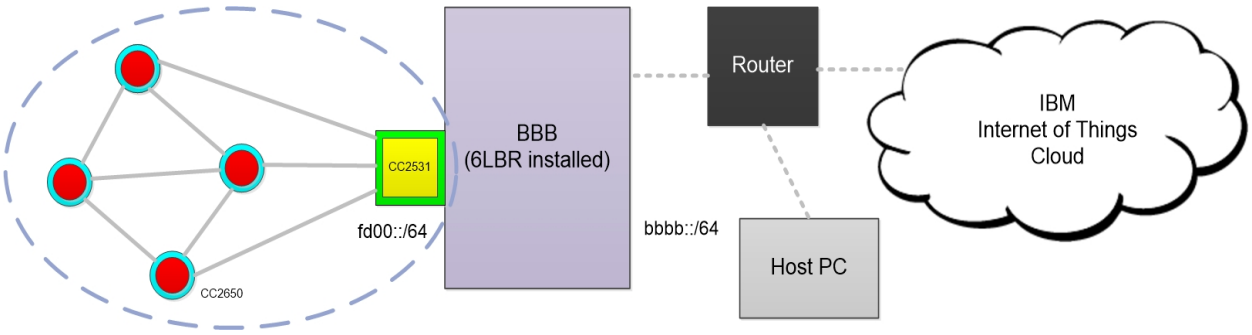

Figure 9. System architecture. The 6LoWPAN mesh network (prefixed fd00::/64) communicate to the BBB using the CC2531 SLIP radio. The BBB forwards the packets to the IBM cloud.

Our sensors, the TI SimpleLink™ CC2650STK SensorTag™, are equipped with various low-power sensors such as air pressure, accelerometer, gyroscope, magnetometer, object temperature, ambient temperature, and humidity. Its ARM® Cortex®-M3 wireless MCU is powered by a single coin cell CR2032 battery. Although the SensorTag is also enabled with ZigBee® technology, the implementation of this project used 6LoWPAN.

The border router is the BeagleBone Black with AM335x 1GHz ARM® Cortex-A8 processor, 512MB DDR3 RAM, 4GB 8-bit eMMC on-board flash storage and 2xPRU 32-bit microcontrollers. The BBB is one of the two components that form the gateway connecting the WSN to the Internet.

The second component is TI CC2531 USB enabled system-on-chip (SoC) solution for 802.15.4 applications, which is our RF transreceiver. CC2531 is used together with the BBB as the MCU-hosted gateway as described in [2] to handle the WSN side of the border router. Figure 10 depicts how CC2531 works together with the BBB to form the USB-enabled MCU-hosted gateway.

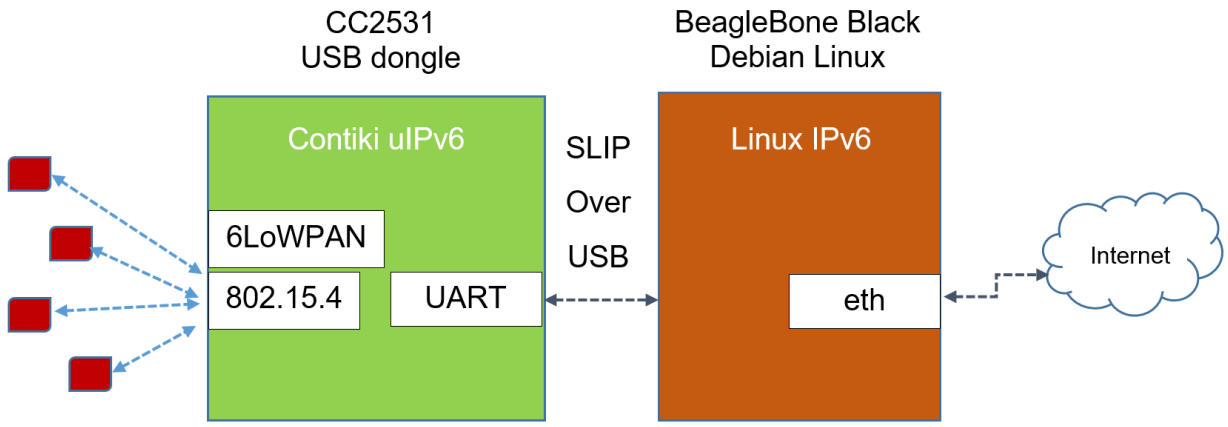

Figure 10. MCU-hosted gateway connecting the WSN (red boxes) with the Internet.

Hardware

Basically, we have a border router (BeagleBone Black), a SLIP radio (CC2531 USB dongle), and sensor(s) (CC2650 SensorTag). To flash the SLIP radio and the sensors, however, we need two debuggers: CC Debugger and SimpleLink™ SensorTag Debugger DevPack. The CC Debugger is for the SLIP radio device, while the Debugger DevPack is for the SensorTags. To prepare the development environment on the border router, we need to install the latest Debian image on the BeagleBone Black (BBB), and for that we need a microSD card. Figure 11 depicts the needed hardware for the demo.

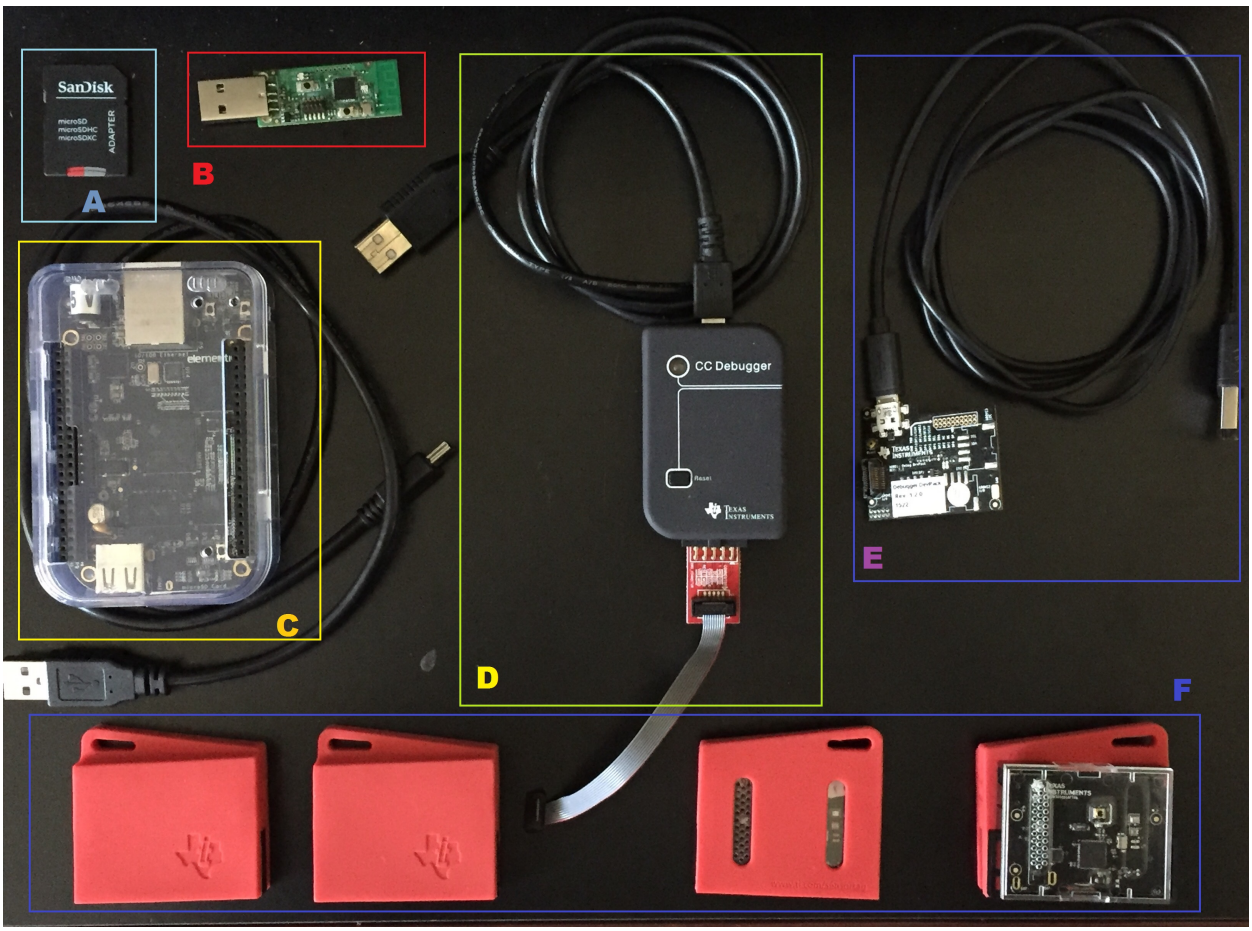

Figure 11. Needed hardware components to run cc2650-web-demo: A) microSD card, B) cc2531 USB dongle, C) BeagleBone Black, D) CC Debugger to be used with CC2531 USB dongle, E) DevPack Debugger to be used with CC2650 SensorTag, and F) CC2650 SensorTag.

Development Environment

To prepare the development environment, use the Contiki setting up sw guide [3].

Setup

The setup roughly follows TI's Cc26xx sw examples [1], although there are some changes in the order. We will walk through 1) the CC2531 SLIP radio setup, 2) the sensor node setup, and 3) the border router setup on the BBB.

SLIP Radio Setup

To flash the CC2531 USB dongle with the SLIP radio program, we need the CC Debugger, TI's SmartRF® Flash Programmer, and the slip-radio-contikimac hex file as the flash image (will be provided in the submission for convenience).

- Download the cc2531-sip-radio_contikimac.hex from [1].

- Plug the CC2531 to the PC-connected BBB to power it and then connect the CC Debugger to the plugged CC2531.
- Connect the CC Debugger to the host PC. A green light on the CC Debugger marks proper connectivity (Figure 12).

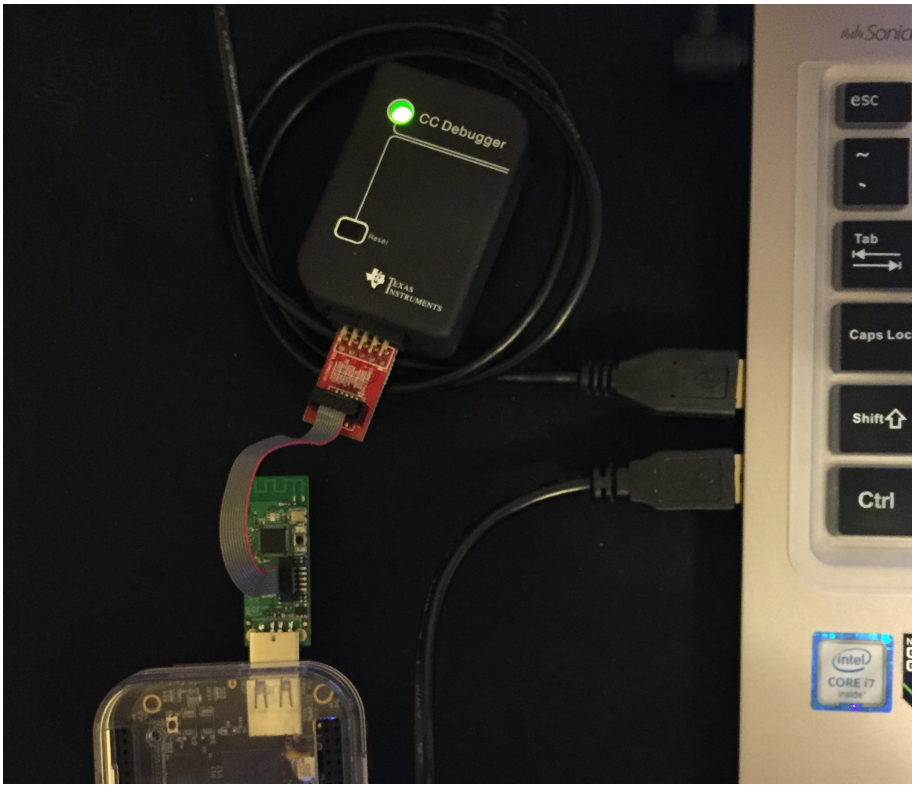

Figure 12. Connecting the CC Debugger to cc2531

- Open the Flash Programmer, and on the "What do you want to program?" dropdown menu, select "Program CCxxxx SoC or MSP430."
- Click on the "System-on-Chip" tab to see the following display (Figure 13).

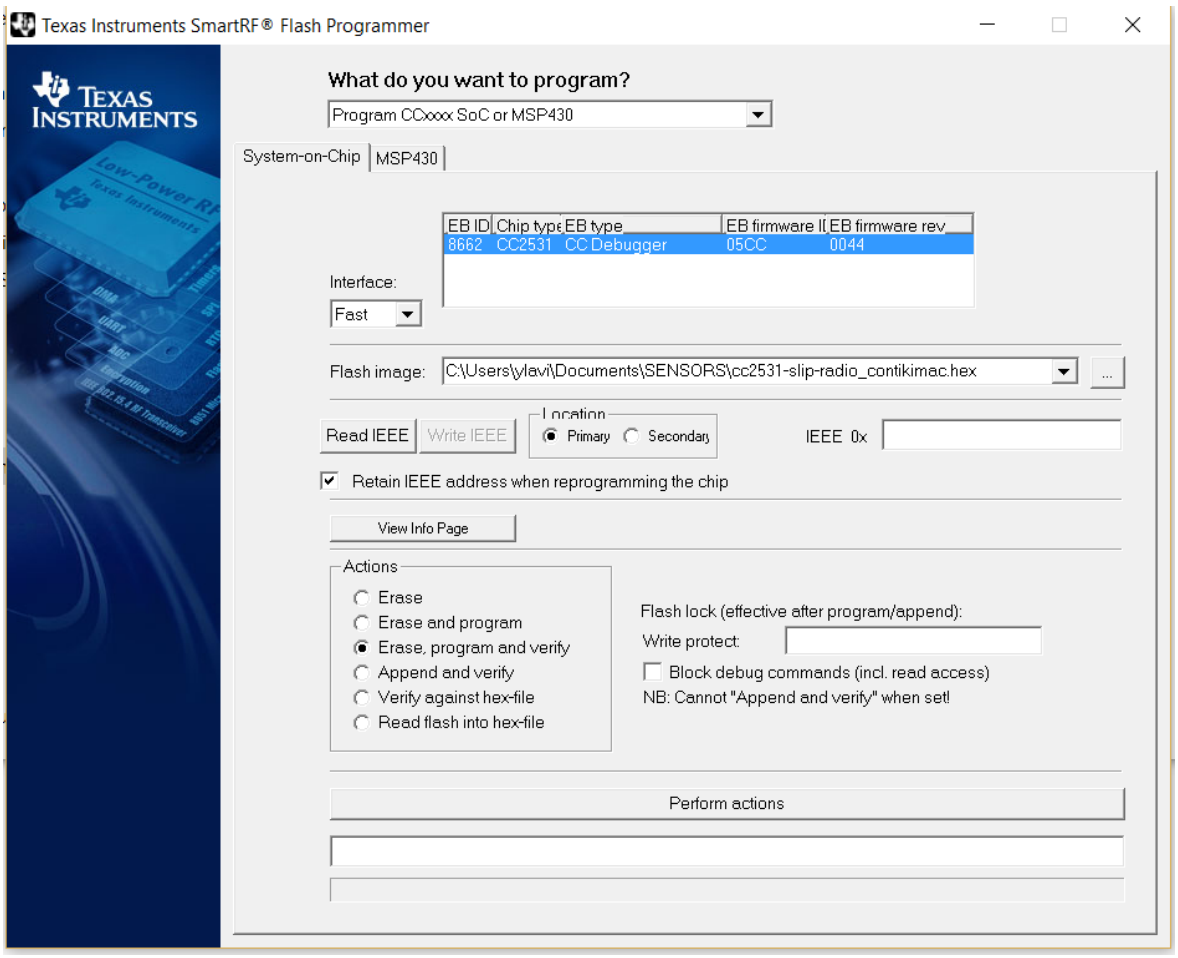

Figure 13. Flashing the CC2531 USB dongle with the slip-radio-contikimac hex program using CC Debugger and Flash Programmer.

- Select the cc2531-slip-radio_contikimac.hex
- Select “Erase, program and verify”
- Click “Perform actions” to flash

Now the CC2531 USB dongle is equipped with the slip-radio_contikimac program at 2.4 GHz.

Sensor Node Setup

The default RF band is sub-1GHz; thus, since we are using 2.4 GHz, we need to manually change the RF band by adding the following command in ~/contiki/platform/srf06-cc6xx/contiki-conf.h (modification shown in Figure 14):

```
#define DOT_15_4G_CONF_FREQUENCY_BAND_ID DOT_15_4G_FREQUENCY_BAND_2450
```

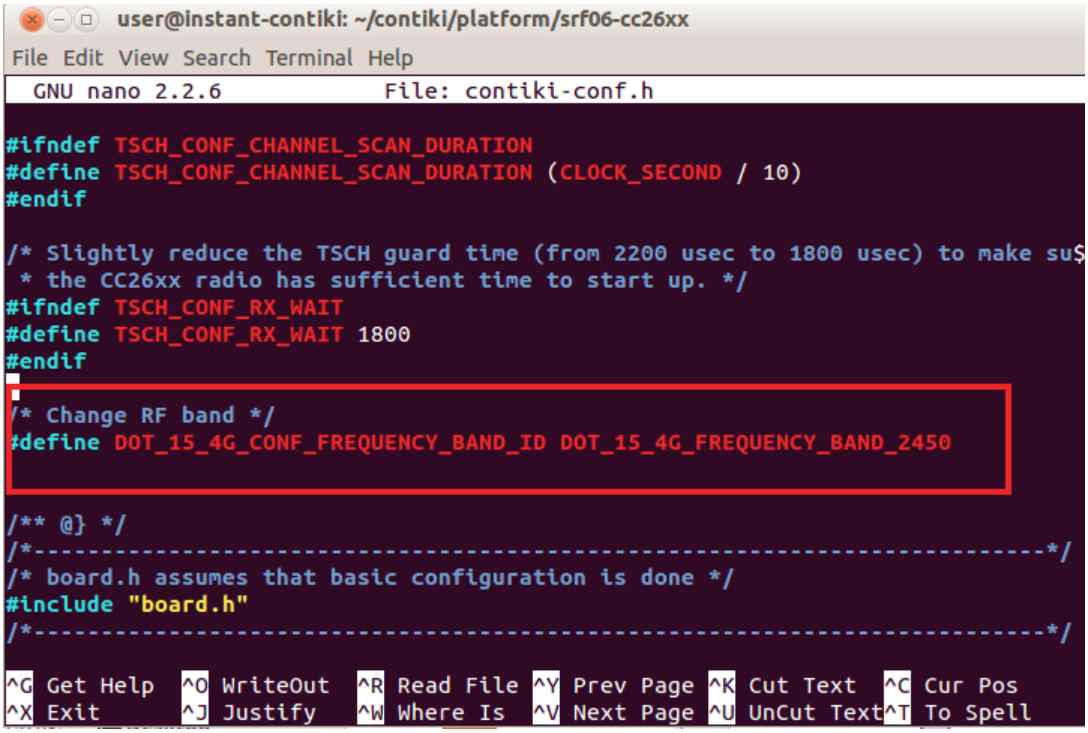

Figure 14. Changing the RF band to 2.4GHz in ~/contiki/platform/srf06-cc26xx/contiki-conf.h

Next, we are changing the IBM MQTT broker IP address in ~/contiki/examples/cc26xx/cc26xx-web-demo/mqtt-client.c.

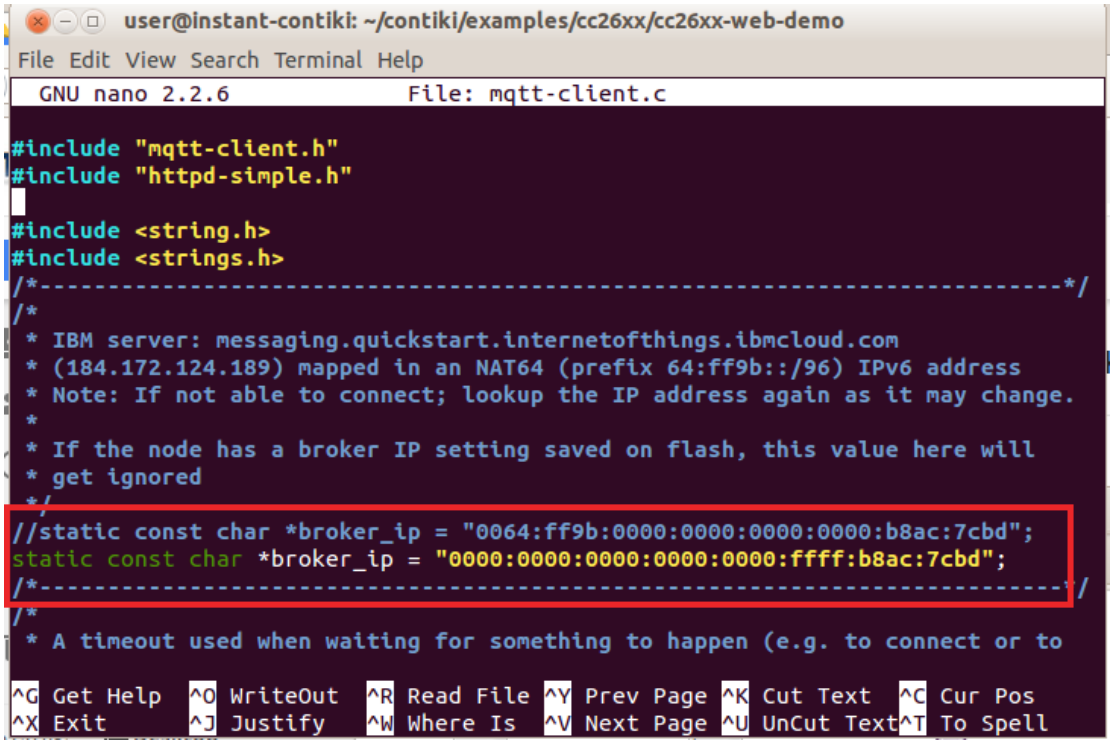

Figure 15. Changing the MQTT broker IP to 0000:0000:0000:0000:0000:ffff:b8ac:7cbd.

Note that we did not change the last 4 bytes (b8ac7cbd) but merely the prefix. The 6lbr gives options whether we wanted to use the 64:ff9b/96 prefix (called the "Well-Known Prefix" as described in RFC6052 [4]) or not. This project chose to use the IPv6-mapped IPv4 notation (prefix ::ffff/96) instead of the RFC6052 prefix (64:ff9b/96). The last four bytes b8ac7cbd are hex for 184.172.124.189, the IBM MQTT broker IPv4 address.

Now we are ready to build the cc26xx-web-demo and flash the cc2650 with the bin file. We need the SensorTag Debugger DevPack and the SmartRF™ Flash Programmer 2 to do this.

- Make sure we are in the ~/contiki/examples/cc26xx/cc26xx-web-demo/ directory
- Then build it using:
  ```
  make TARGET=srf06-cc26xx BOARD=sensortag/cc2650 cc26xx-web-demo.bin
  ```
- Copy the cc26xx-web-demo.bin file from Instant Contiki to the host PC
- Take off the SensorTag from its casings; don't connect it to the Debugger DevPack yet
- Connect the SensorTag Debugger DevPack to the host PC
- Connect the SensorTag to the Debugger DevPack (Figure 16)

- Open SmartRF™ Flash Programmer 2

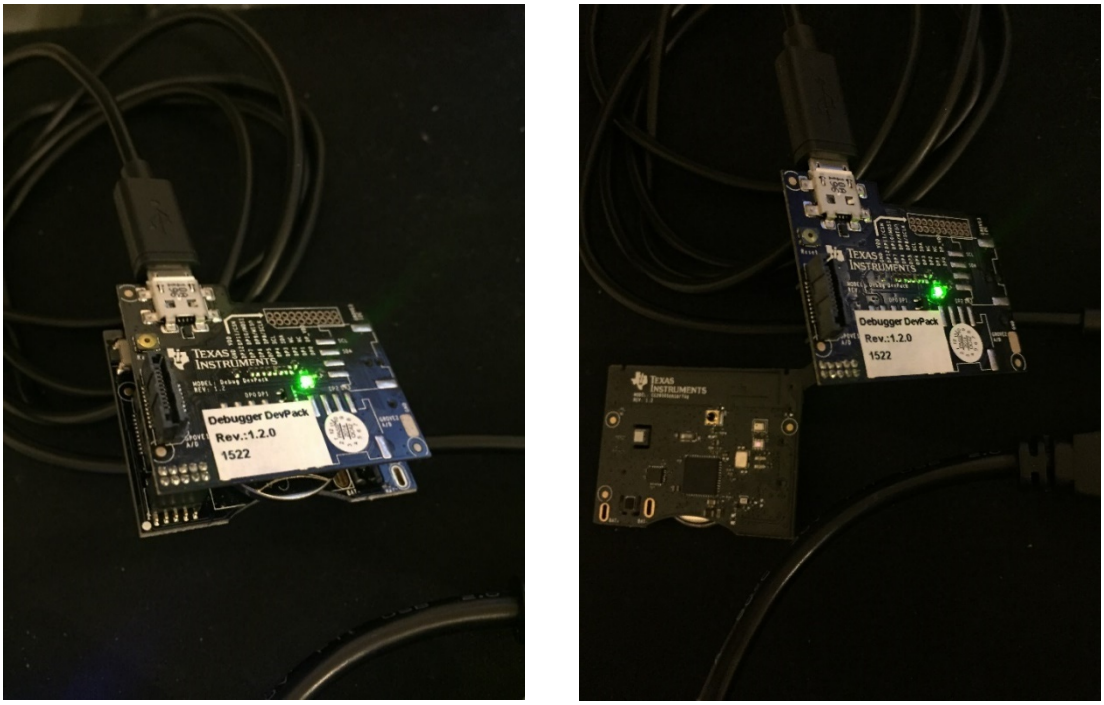

Figure 16. The out-of-casing SensorTag and the Debugger DevPack not connected yet (left) and when they are connected (right).

- Make sure that the Flash Programmer 2 detected the CC2650
- Select the cc2650-web-demo.bin file
- Check “Erase,” “Program,” “Verify” boxes for Actions
- Click on the play icon 
- Figure 17 depicts how the actions look like
- Flash the cc26xx-web-demo.bin file to as many CC2650 SensorTags as needed
- Now all our SensorTags have become mesh devices

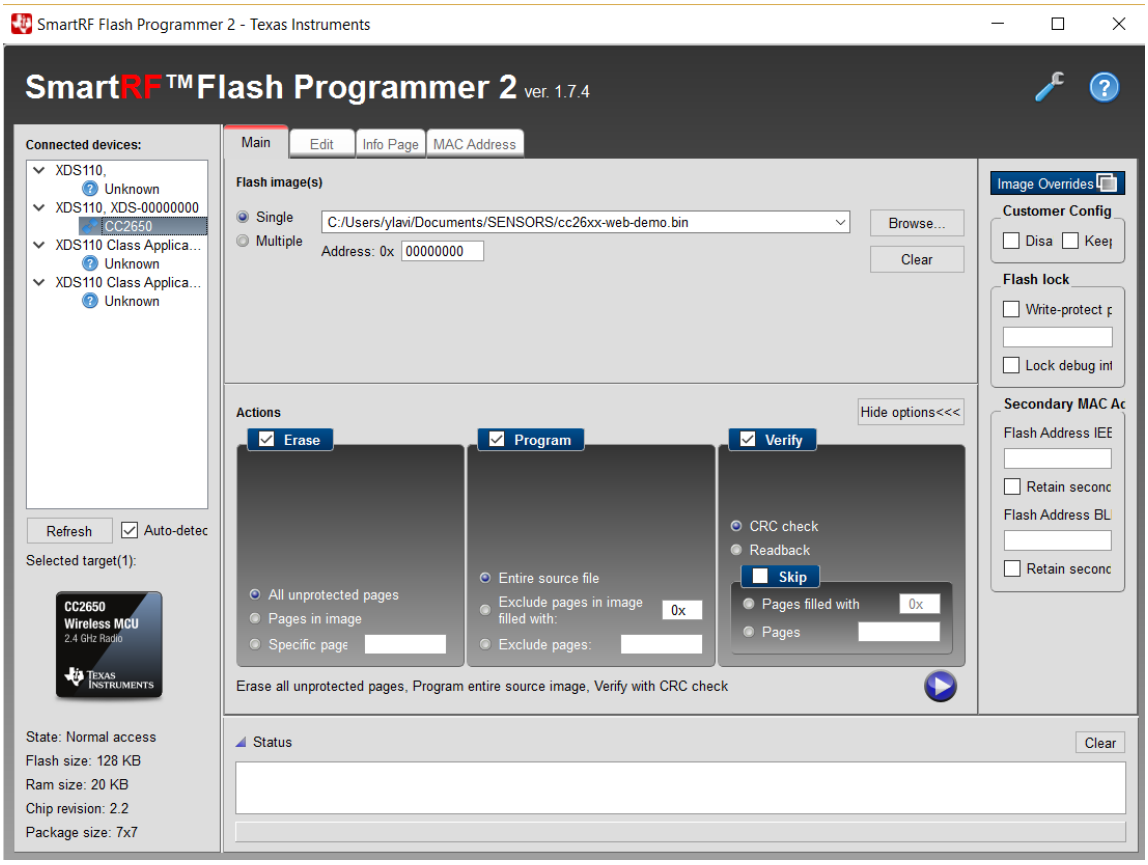

Figure 17. Flashing the CC2650 SensorTag with the cc26xx-web-demo.bin using Debugger DevPack and Flash Programmer 2.

Border Router Setup

The BBB acts as the border router that forwards the sensor data packets to the IBM cloud. To do so, we need a software called 6LBR (6LoWPAN Border Router).

- Connect the Ethernet cable to the BBB (note: failure to do this before connecting the BBB to the host PC led to the BBB acting very slowly to detect Internet connection)
- Connect the BBB to the host PC
- After the host PC detects the presence of BBB, connect to the BBB through ssh or PuTTY (if using Windows)
- Login with username `debian` and password `temppwd`
- Verify current date using `date`
- If incorrect, update by typing `sudo ntpdate pool.ntp.org`
- Verify date again
- Verify Internet connection using `ping 8.8.8.8`
- After successful ping, build and install 6lbr:

```
sudo apt-get update
git clone https://github.com/cetic/6lbr
cd 6lbr
git submodule update –init –recursive
cd examples/6lbr
make all
make plugins
make tools
sudo make install
make plugins-install
update-rc.d 6lbr defaults
```

- Create the 6lbr.conf file. The example can be found in /etc/6lbr/6lbr.conf.example.

```
cd /etc/6lbr
touch 6lbr.conf
```

- The newly created 6lbr.conf should look like the following:

```
MODE=ROUTER
#MODE=SMART-BRIDGE
#MODE=RPL-RELAY
#MODE=FULL-TRANSPARENT-BRIDGE
#MODE=NDP-ROUTER
#MODE=6LR
#MODE=RPL-ROOT

RAW_ETH=1
BRIDGE=0
DEV_BRIDGE=br0
DEV_TAP=tap0
DEV_ETH=eth0
#RAW_ETH_FCS=1
RAW_ETH_FCS=0

DEV_RADIO=/dev/ttyACM0 #2.4GHz USB Dongle will enumerate like ACM0
```

```
#DEV_RADIO=/dev/ttyUSB1 #SmartRF06EB+CC13xxEM will enumerate like USB1
```

```
BAUDRATE=115200

LOG_LEVEL=3
```

- Start 6lbr:
  `sudo /etc/init.d/6lbr start`
- After started, we should find an nvm.dat file in /etc/6lbr. If it is not there, we need to restart the BBB, start the 6lbr, and check again if the nvm.dat file is created. The nvm.dat file is needed for the next step.
- Still in the /etc/6lbr directory, check 6lbr configuration using nvm-tool:
  `/usr/lib/6lbr/bin/nvm_tool --print /etc/6lbr/nvm.dat`
- As stated on the 6lbr web page, in March 2016, Contiki replaced the default WSN context from aaaa::/64 to fd00::/64 [5]. This requires the following modifications in the 6lbr configuration:

`/usr/lib/6lbr/bin/nvm_tool –update --wsn_prefix fd00:: /etc/6lbr/nvm.dat`
`/usr/lib/6lbr/bin/nvm_tool –update –wsn_ip fd00::100 /etc/6lbr/nvm.dat`

- Channel is supposed to be 25. Recent experience shows that channel 25 is already the default in the latest 6lbr. However, if the channel is not 25, type the following:
  `/usr/lib/6lbr/bin/nvm_tool --update --channel 25 /etc/6lbr/nvm.dat`
- It is important to note, however, that all the modifications using nvm-tool can be done later too on the “Configuration” page of the 6lbr client website.
- Connect the CC2531 USB dongle to the BBB
- Verify that the BBB detected the CC2531:
  `sudo lsusb -v`
- We should be able to find the following (among other lines):

```
Bus 001 Device 002: ID 0451:16a8 Texas Instruments, Inc.
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB               2.00
```

```
  bDeviceClass            2 Communications
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0         8
  idVendor           0x0451 Texas Instruments, Inc.
  idProduct          0x16a8
  bcdDevice            0.00
  iManufacturer           1 Texas Instruments
  iProduct                2 CC2531 USB Dongle
  iSerial                 3 00124B00015A755C
```

- Now we have all the necessary components to start the demo

6LBR Web Client Demo

The 6lbr web client is a demo displaying the webpages powered by 6lbr. Besides viewing the detected sensor devices, the 6lbr webpage also displays the network topology, the IP64 mapping, and the current configuration, which we can directly tweak without having to use terminal and nvm-tool.

- On the BBB, to activate 6lbr, type:
  `sudo /etc/init.d/6lbr start`
- Open [bbbb::100] on a web browser (note: bbbb::100 is the address of the 6lbr webpage)
- The 6lbr page is displayed
- Now we are routing the data (should be done after opening the bbbb::100 page). On the BBB, type the following:
  `sudo ip -6 route add bbbb::100 dev eth0`
  `sudo route -A inet6 add fd00::/64 gw bbbb::100`
- The BBB routing table should look like the following:

```
debian@beaglebone:~/6lbr/examples/6lbr$ route -A inet6
Kernel IPv6 routing table
Destination                    Next Hop                   Flag Met Ref Use If
::1/128                        ::                         U    256 0     0 lo
bbbb::100/128                  ::                         U    1024 1   339 eth0
fd00::/64                      bbbb::100                  UG   1    1   124 eth0
fe80::/64                      ::                         U    256 0     0 usb0
fe80::/64                      ::                         U    256 1    39 eth0
::/0                           ::                         !n   -1  1   643 lo
::1/128                        ::                         Un   0   2    11 lo
fe80::7aa5:4ff:fec8:a0b0/128   ::                         Un   0   1     0 lo
fe80::7aa5:4ff:fec8:a0b1/128   ::                         Un   0   1     0 lo
ff00::/8                       ::                         U    256 1   115 usb0
ff00::/8                       ::                         U    256 1   508 eth0
::/0                           ::                         !n   -1  1   643 lo
```

Figure 18. The BBB routing table. Notice that packets from any address with fd00::/64 would be forwarded to bbbb::100.

- On the 6lbr main page, click “Configuration”
- Figure 19 shows a part of the Configuration:

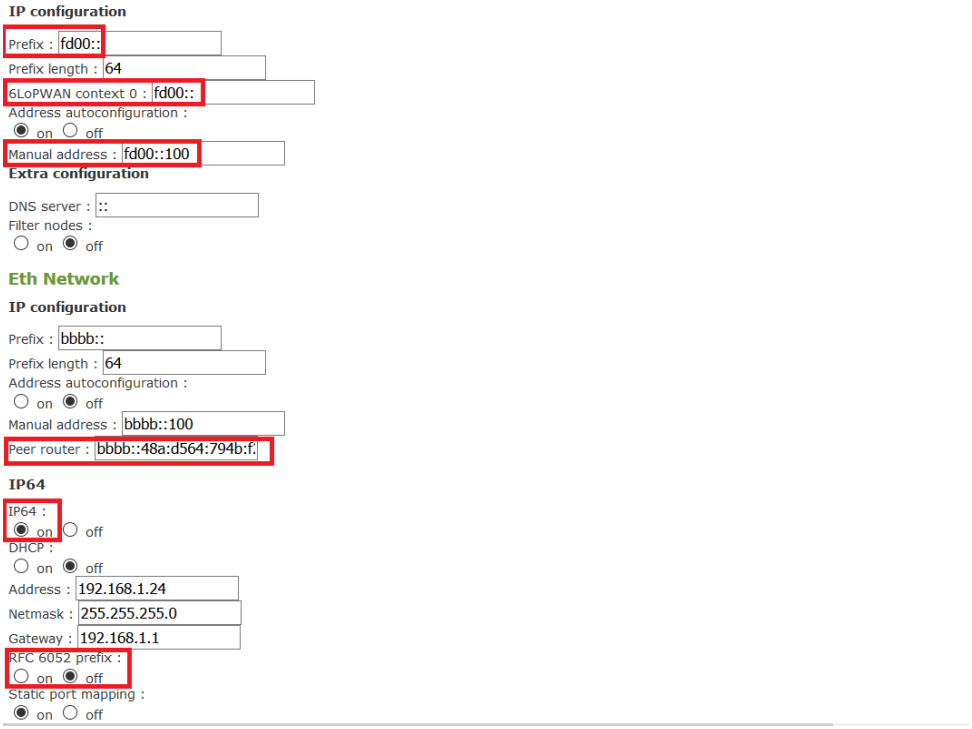

Figure 19. Parts of the Configuration page. The red boxes denote the items that require careful attention. Note that “Prefix,” “6LoWPAN context 0,” and “Manual address” are all started with fd00 and not the old prefix aaaa. “Peer router” should be a PC in the same subnet as BBB with address prefix bbbb. “IP64” should be turned on as this would activate NAT64 in the 6lbr. Finally, “RFC 6052 prefix” should be turned off as we are not using the 64:ff9b::/64 prefix.

- Change the “Peer Router” to either the host PC IPv6 address (prefixed with bbbb::), or any PC in the same subnet

- Note that 6lbr supports NAT64; thus, we do not have to use Wrapsix as instructed in [1]
- The IPv6-IPv4 address mapping can be seen in the "Status" page and look like this:

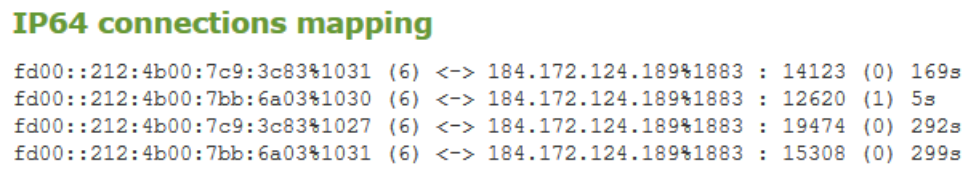

Figure 20. 6lbr's NAT64. It shows that the IPv6 nodes (fd00 addresses) are mapped to the IPv4 destination address of 184.172.124.189, which is the IPv4 broker IP address of IBM Quickstart.

- Now we want to activate RA reception. Windows users do not need to do anything since RA reception is activated by default. Linux or Mac users should consult [1]
- To ensure that the components are working, we want to ping them
- On the host PC:

  `ping bbbb::100`

  `ping -6 <your_ipv6_node_address e.g. fd00::212:4b00:7c9:3c83>`
- Pinging the bbbb::100 should get the following response:

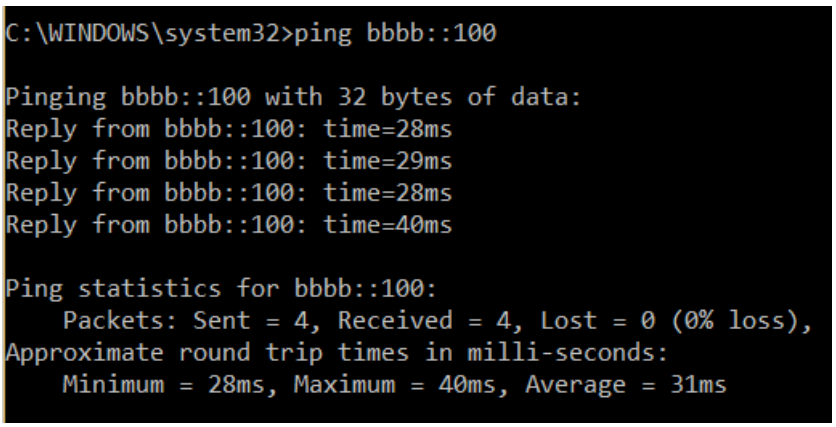

Figure 21. bbbb::100 ping response.

- Pinging the sensor node should give somewhat of the following response:

```
C:\WINDOWS\system32>ping -6 fd00::212:4b00:7c9:3c83

Pinging fd00::212:4b00:7c9:3c83 with 32 bytes of data:
Reply from fd00::212:4b00:7c9:3c83: time=499ms
Reply from fd00::212:4b00:7c9:3c83: time=2289ms
Reply from fd00::212:4b00:7c9:3c83: time=2913ms
Request timed out.

Ping statistics for fd00::212:4b00:7c9:3c83:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 499ms, Maximum = 2913ms, Average = 1900ms
```

Figure 22. fd00::212:4b00:7c9:3c83 ping -6 response.

- The sensors can be seen in "Sensors," like so:

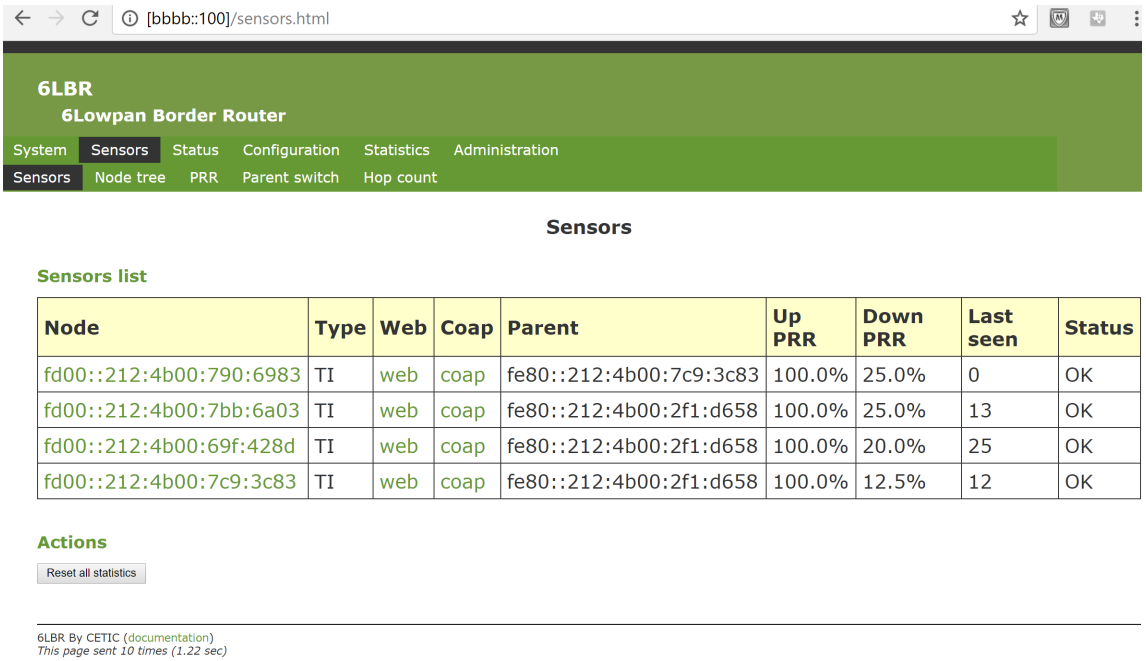

Figure 23. The sensors list as detected by 6lbr. The list also includes the sensor's Parent, that is, the next hop node in the shortest upward path to the border router. For example, the node fd00::212:4b00:790:6983 has fe80::212:4b00:7c9:3c83 as Parent, which is the same as the fd00::212:4b00:7c9:3c83 (only prefix difference) that has fe80::212:4b00:2f1:d658 (the border router) as Parent. This indicates that node 6983 (last four digits of the node) hops to 3c83 to get to the border router d658.

- Also, note that Parent d658 is the DODAG root as it acts as the border router.
- If we look at Status > RPL (Figure 24), we can see the DODAG configuration. In this case the DODAGID is fd00::212:4b00:2f1:d658, which is the DODAG root that is recognized as the border router (destination) in Figure 23.

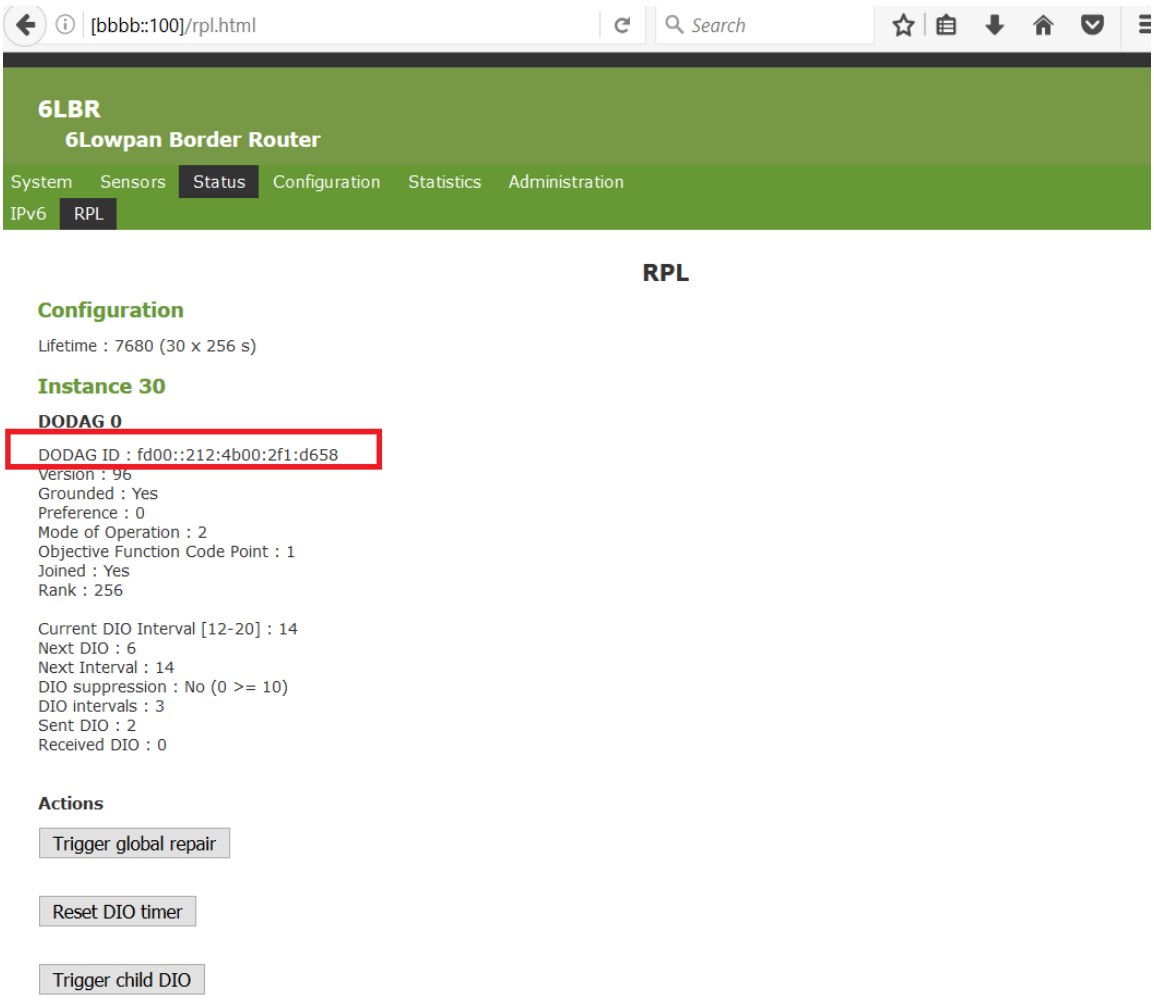

Figure 24. RPL configuration.

- The network topology can be seen by clicking “Node tree” on the “Sensors” page.
- The following shows three of many different arrangements of four sensors, depending on their location.
- Figure 25 shows node 6983 losing access to the border router (BR) and thus needs to hop to 3c83 to access BR. The 6a03 and 428d nodes can directly access BR.

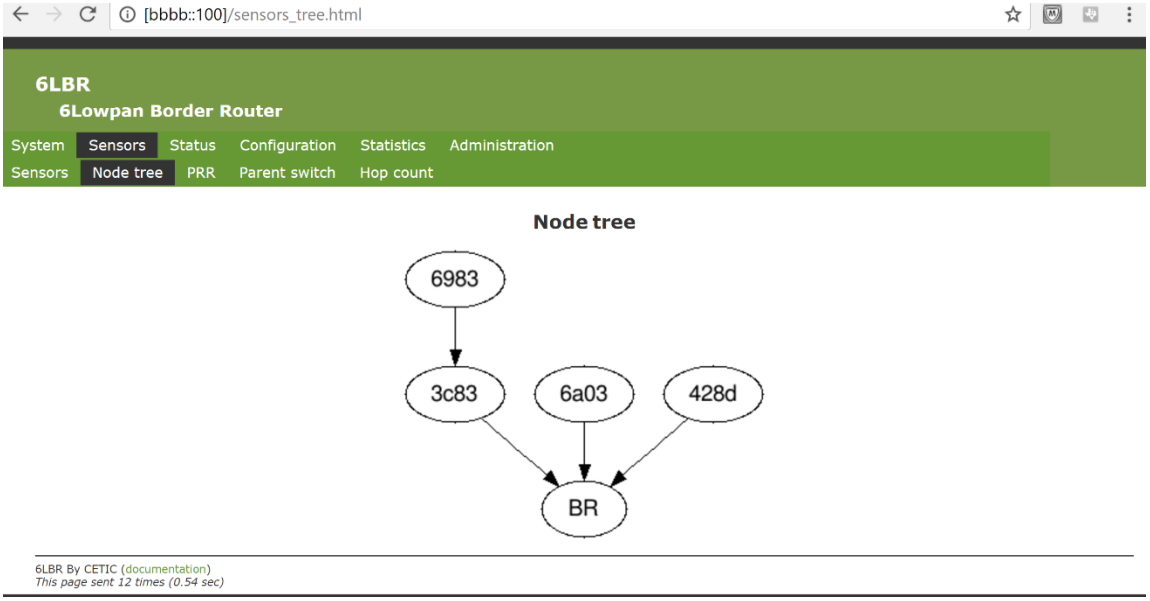

Figure 25. Node 6983 hops to 3c83 to get to BR.

- Figure 26 shows nodes 6983 and 428d using node 3c83 as relay to get to BR. Node 6a03 can directly access BR.

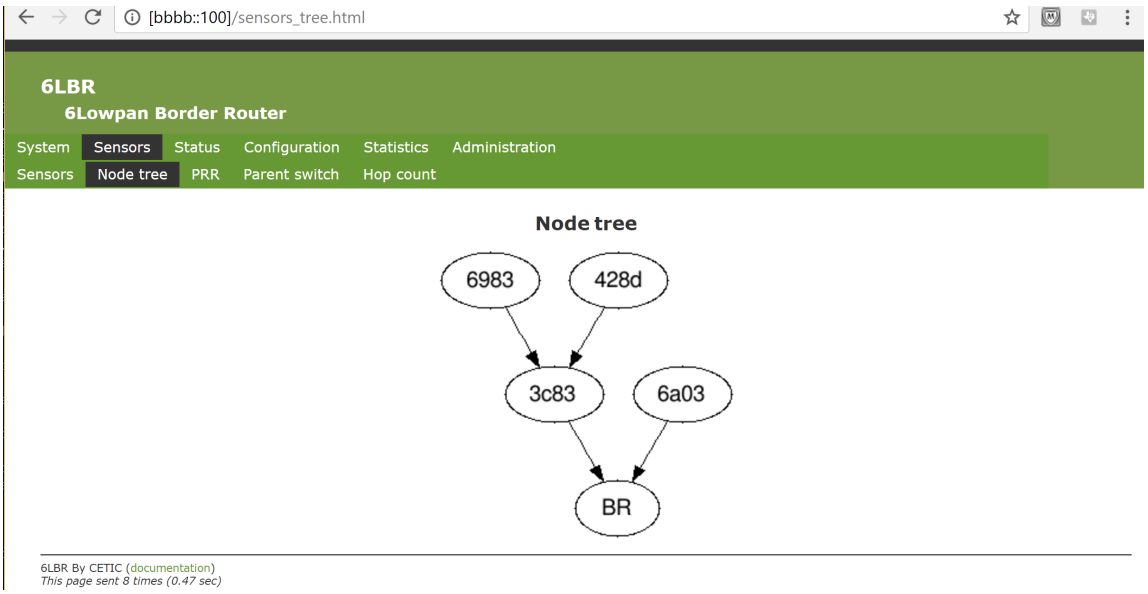

Figure 26. Nodes 6983 and 428d hop to 3c83 to get to BR.

- Figure 27 shows node 6983 using node 6a03 as relay, while node 6a03 itself is using node 3c83 as relay to get to BR. Similarly, node 428d is shown to be unable to reach BR, and thus it uses node 3c83 as relay to access BR.

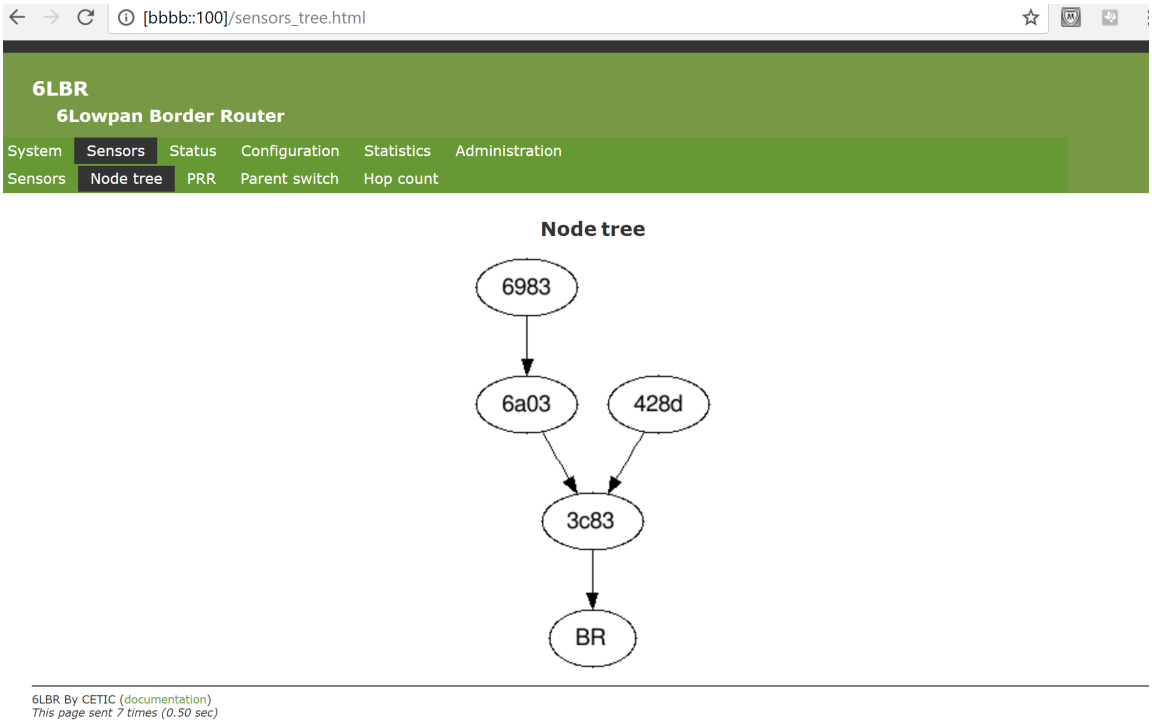

Figure 27. Node 6983 hops to 6a03, which in turn hops to 3c83 to get to BR. Node 428d hops to 3c83 to access BR.

Sensor Web Server Demo

This demo builds on top of the 6LBR client demo. As seen in Figure 28, 6LBR displays a list of the detected nodes in a table. Clicking the “web” on any node row would open a web page generated by the selected node.

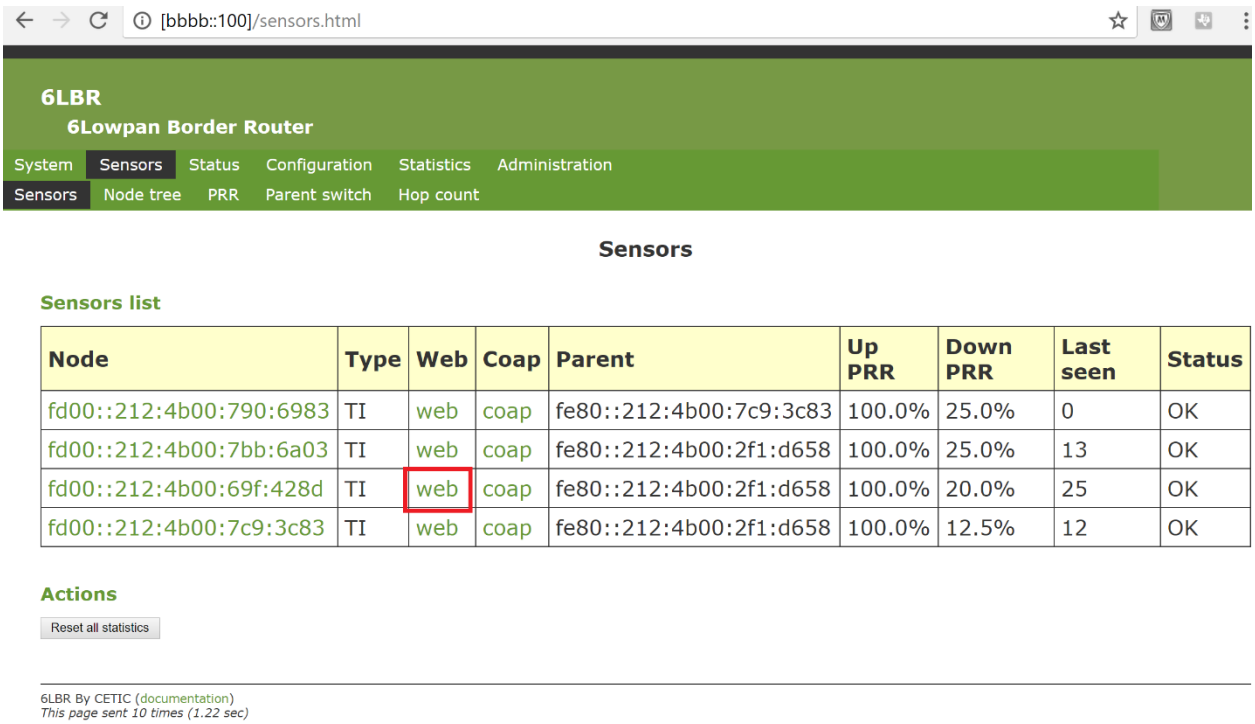

Figure 28. 6LBR Sensor Table. Clicking "web" would open a web page that is generated by the selected node.

In this example, clicking "web" on the row fd00::212:4b00:69f:428d would open a web page. Note that the URL is fd00::212:4b00:69f:428d, the node's IPv6 address (Figure 29). Here we see all the data from all sensors (air pressure, battery temperature, humidity, light, accelerometer x, y, z, gyroscope x, y, z, etc.) alongside the node's neighbors and default route, which is its next hop towards the router or the router itself.

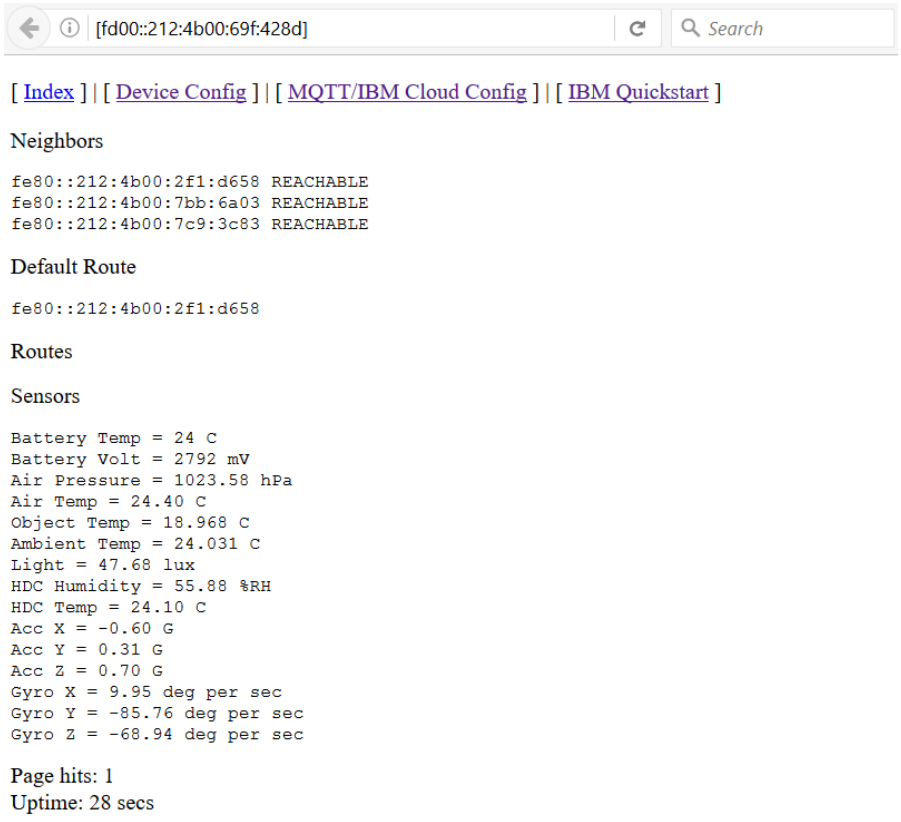

Figure 29. The index page of node fd00::212:4b00:69f:428d.

Figure 30 shows the configuration page (also generated by the node) where we could select which sensors we want to activate or deactivate.

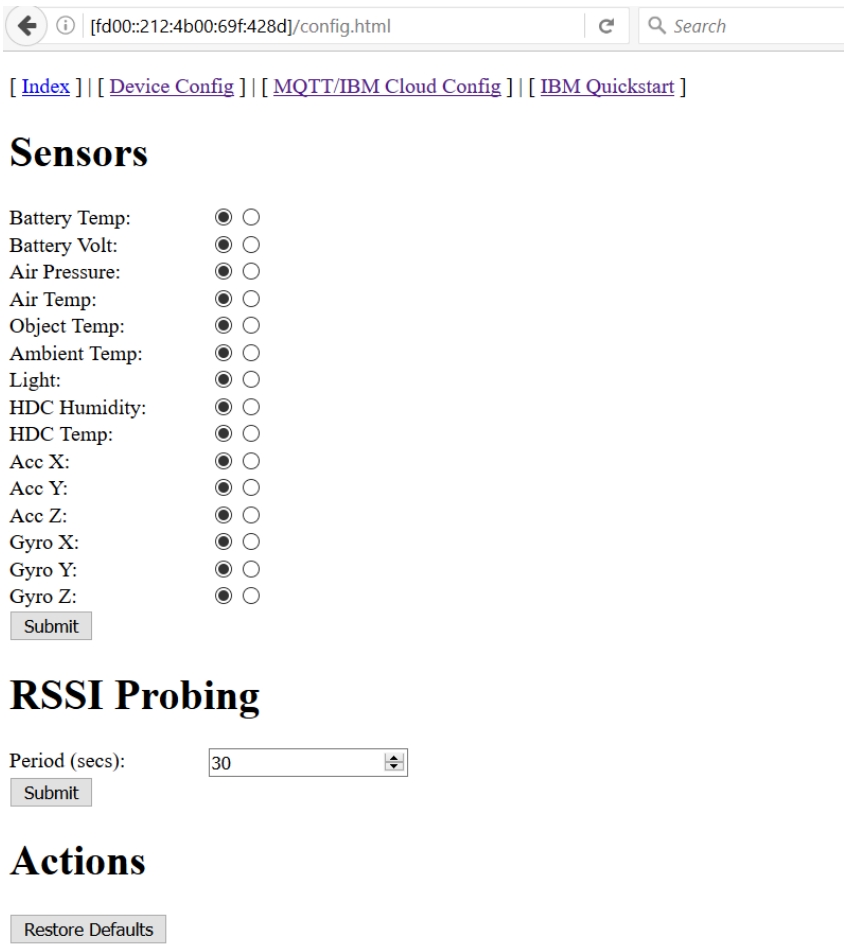

Figure 30. The sensor configuration page. We can activate or deactivate certain sensors by filling the right (activate) or left (deactivate) circle.

Figure 31 shows the MQTT/IBM Cloud configuration where we specify or modify the MQTT broker IP address.

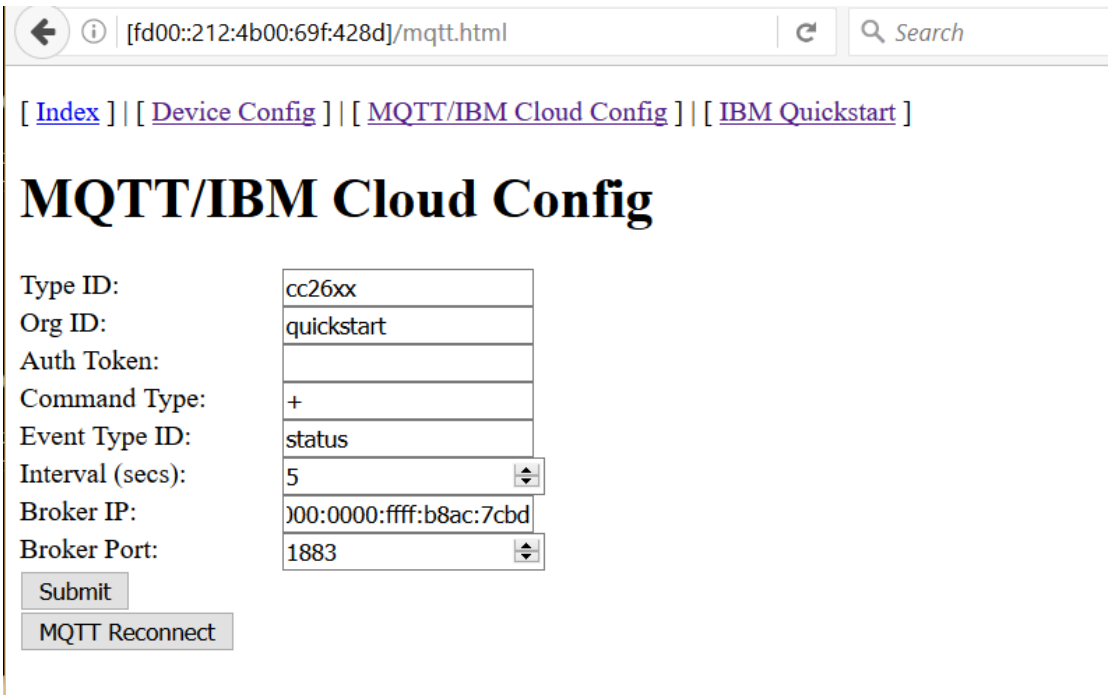

Figure 31. The MQTT/IBM Cloud configuration page. Following the MQTT protocol, we can modify the type ID, organization ID, event type ID, interval, broker IP address, and broker IP port. Note that the broker IP address is 0000:0000:0000:0000:0000:ffff:b8ac:7cbd.

IBM MQTT Client

The IBM MQTT client demo can be accessed either through the sensor web page (notice the "IBM Quickstart" link at the top right on Figure 31) or directly by going to https://quickstart.internetofthings.ibmcloud.com/#/device and putting the device ID (00124b followed by the last 3 bytes of the node IPv6 address). To get a better visualization of the MQTT packets exchange between the sensors and the IBM Quickstart, we will sniff them using tcpdump, simply because the MQTT is built on top of the TCP.

- On the BBB, type:
  ```
  sudo tcpdump -i eth0 -X -v port 1883
  ```
- If the MQTT packets are being sent and received properly, we should see an exchange between the BBB (appearing as beaglebone.home.XXXX, although in this example it displays beaglebone.home.10614) and the MQTT broker (appearing as msproxy-quickstart-0.da106.internetofthings.ibmcloud.com.1883) in the terminal window (Figure 32). In other words, we should see beaglebone.home.10614 > msproxy-quickstart-0.da106.internetofthings.ibmcloud.com.1883 on one packet and msproxy-quickstart-0.da106.internetofthings.ibmcloud.com.1883 > beaglebone.home.10614 on the next one.

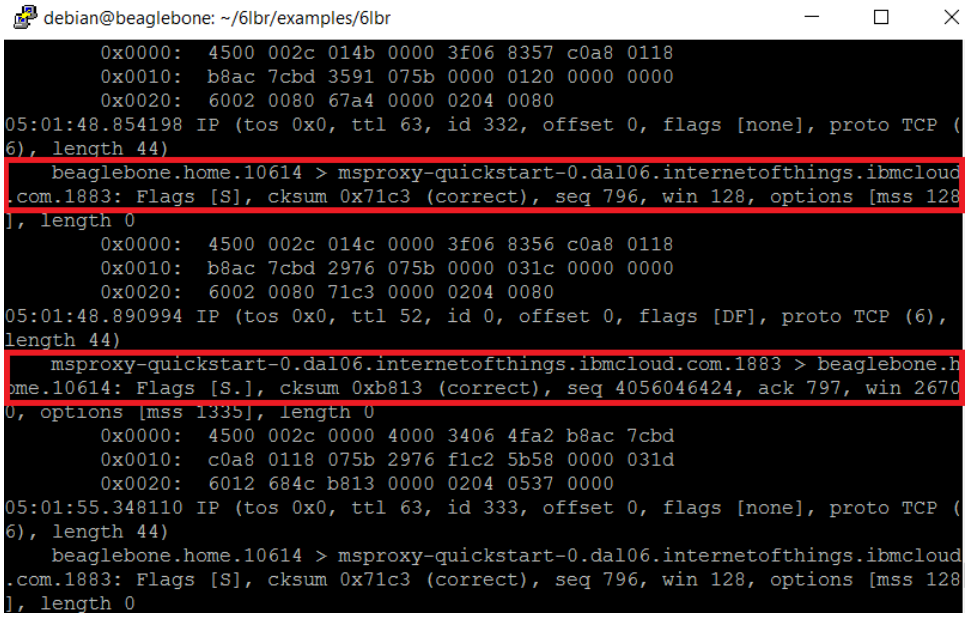

Figure 32. Exchange between the BBB and IBM MQTT broker as indication of proper MQTT packet sending and receiving.

- Once we see the exchange, click the "IBM Quickstart" link on the sensor node web page. It should open the IBM Quickstart page where we could see the sensor data visualization (Figure 33). The IBM Quickstart automatically visualizes the sensor node data without requiring us to input the sensor node ID anymore. If we access the IBM Quickstart page without going through the sensor node web page, we need to input the sensor node ID (00124b followed by the last 3 bytes of the IPv6 address node).
- Figure 33 shows the visualization of the gyroscope Y.

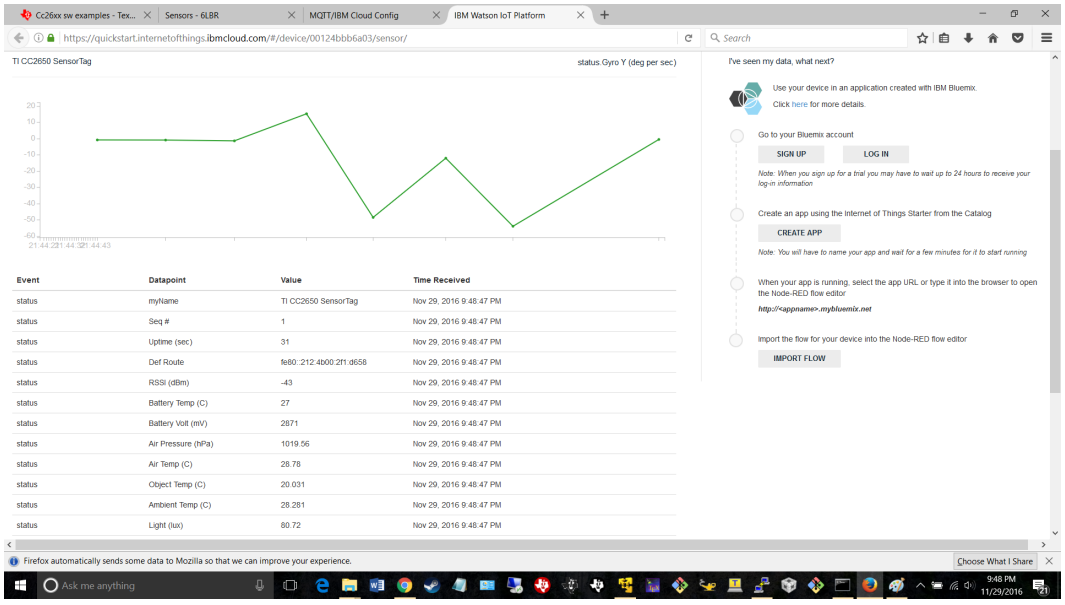

Figure 33. IBM Quickstart visualization of the gyroscope Y data.

- Figure 34 shows visualization of the object temperature data

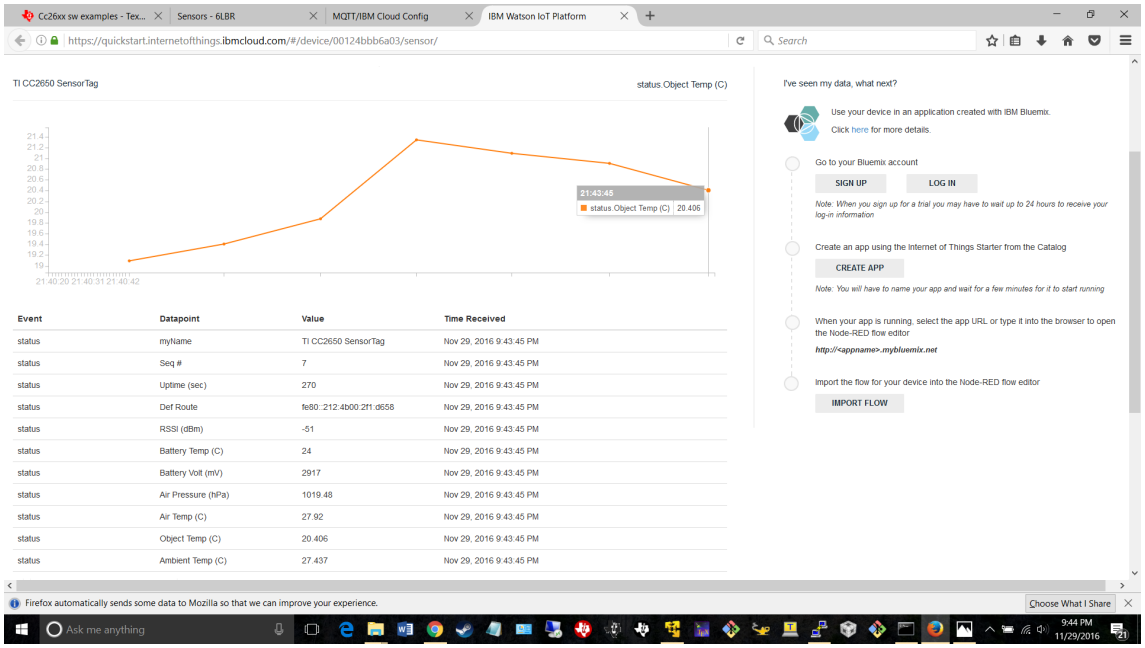

Figure 34. IBM Quickstart visualization of the object temperature data.

References

[1] "Cc26xx sw examples," Texas Instruments, 7 December 2015. [Online]. Available: processors.wiki.ti.com/index.php/Cc26xx_sw_examples. [Accessed 12 November 2016].

[2] G. Oikonomou and I. Phillips, "Experiences from porting the Contiki operating system to a popular hardware platform," in *Proceedings of the 2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS'11)*, Barcelona, 2011.

[3] "Contiki setting up sw," Texas Instruments, 23 September 2015. [Online]. Available: http://processors.wiki.ti.com/index.php/Contiki_setting_up_sw.

[4] C. Bao et al., "IPv6 Addressing of IPv4/IPv6 Translators," IETF RFC 6052, 2010.

[5] L. Deru, "WSN," CETIC 6LBR, 4 May 2016. [Online]. Available: https://github.com/cetic/6lbr/wiki/WSN. [Accessed 23 November 2016].