Contiki - Global IPv6 networks

Antonio Liñán Colina

Contiki

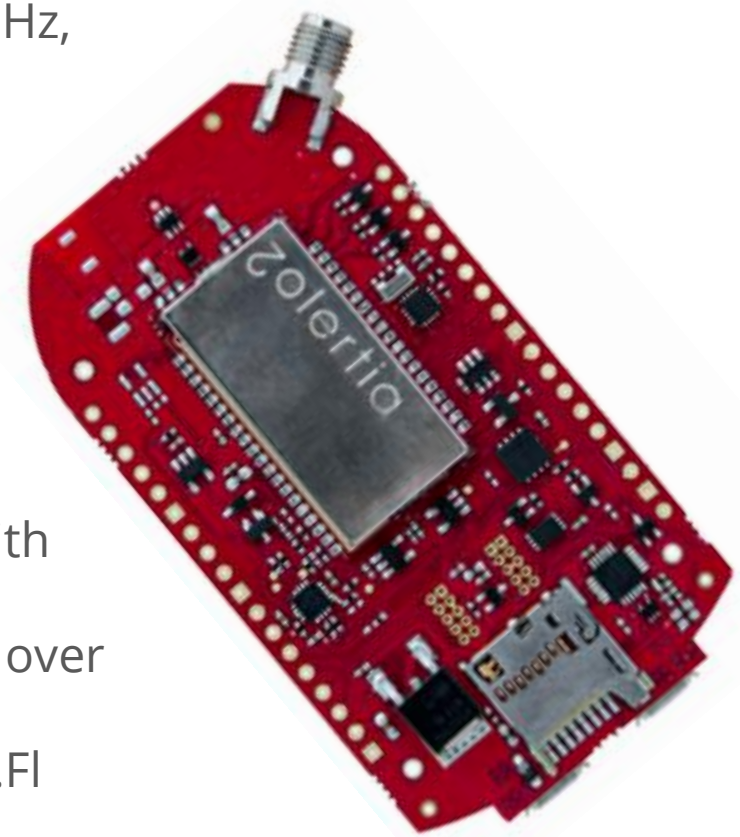
The Open Source OS for the Internet of Things

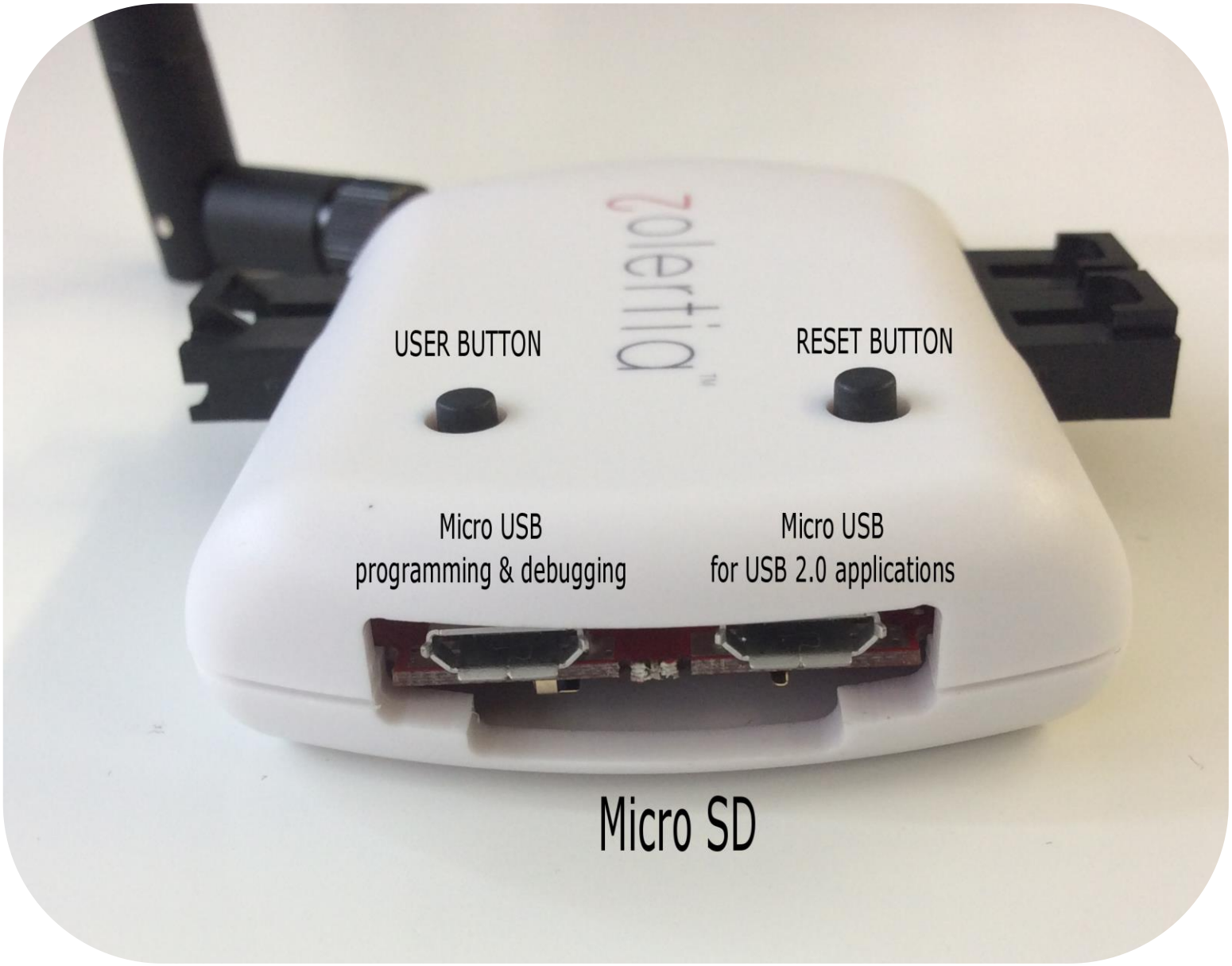
- Architectures: 8-bit, 16-bit, 32-bit
- Open Source (source code openly available)
- IPv4/IPv6/Rime networking
- Devices with < 8KB RAM
- Typical applications < 50KB Flash
- Vendor and platform independent
- C language
- Developed and contributed by Universities, Research centers and industry contributors
- +10 years development

Zolertia RE-Mote

Zolertia RE-Mote (Zoul inside)

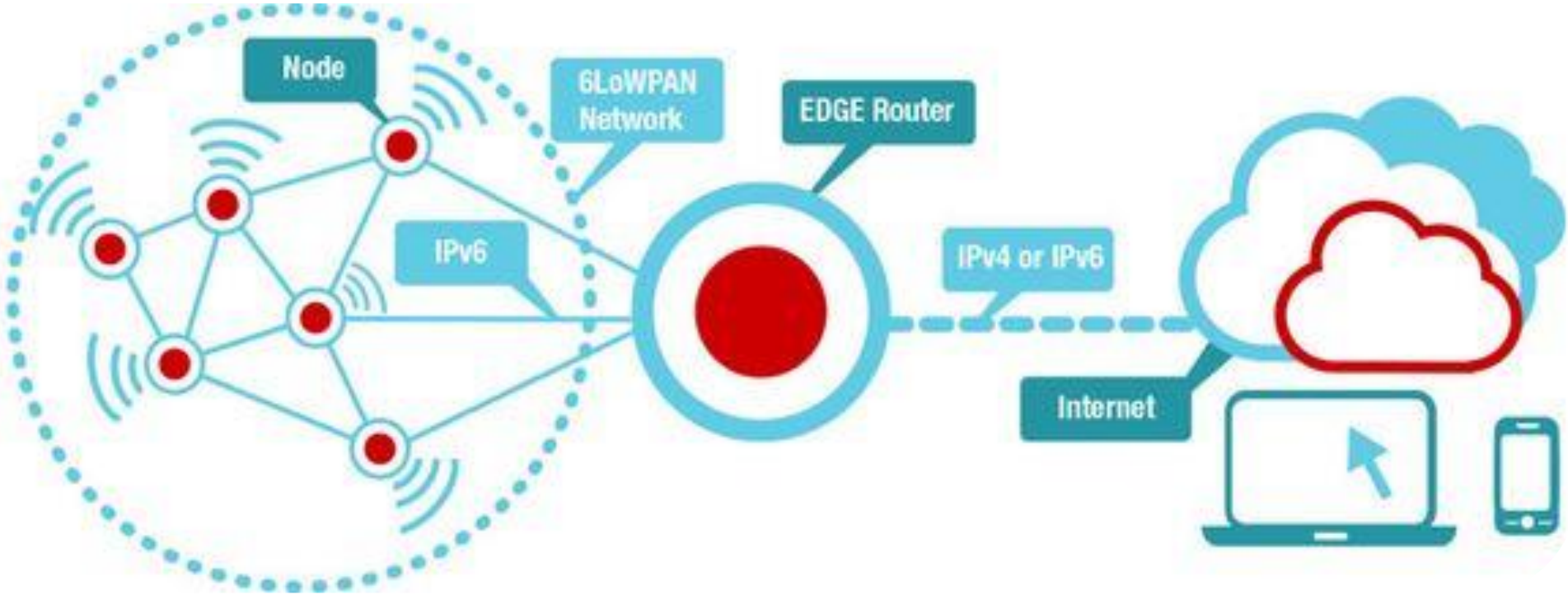
- ARM Cortex-M3, 32MHz, 32KB RAM, 512KB FLASH
- Double Radio: ISM 2.4GHz & 863-925MHz, IEEE 802.15.4-2006/e/g
- Hardware encryption engine and acceleration
- USB programing ready
- Real-Time Clock and Calendar
- Micro SD slot and RGB colors
- Shutdown mode down to 150nA
- USB 2.0 port for applications
- Built-in LiPo battery charger to work with energy harvesting and solar panels
- On-board RF switch to use both radios over the same RP-SMA connector
- Pads to use an external 2.4GHz over U.Fl connector, o solder a chip antenna





02-ipv6

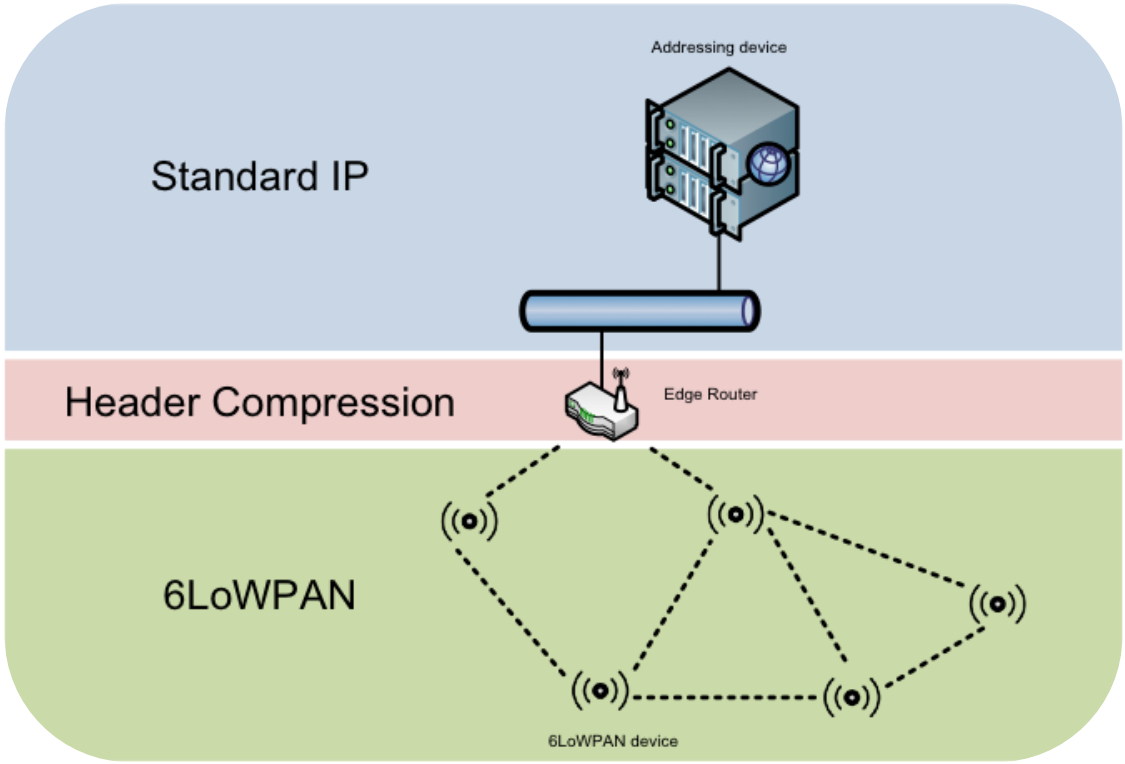
examples/zolertia/tutorial/02-ipv6



examples/zolertia/tutorial/02-ipv6/02-border-router

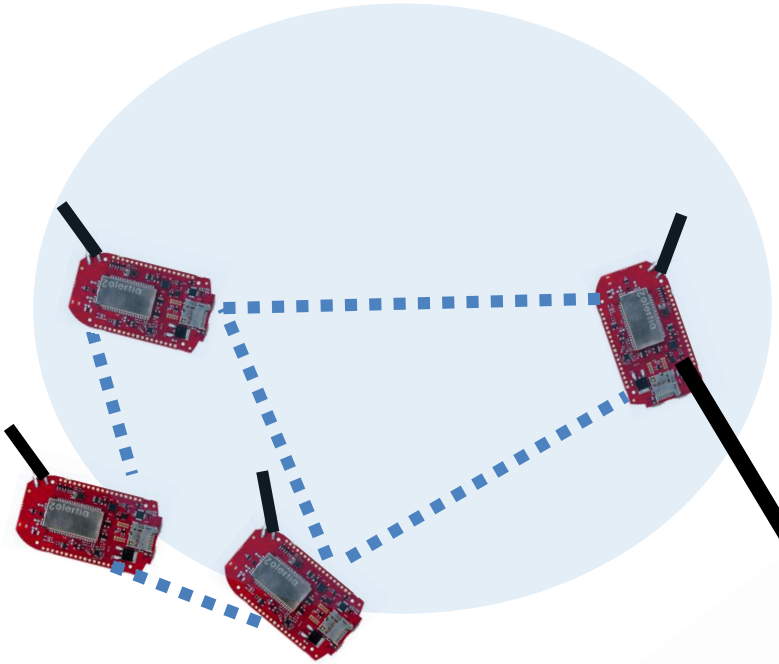
The Border Router (or Edge Router)

A 6LoWPAN Border Router connects a 6LoWPAN network to the Internet, and handles traffic to and from the IPv6/IPv4 and 6LoWPAN networks



examples/zolertia/tutorial/02-ipv6/02-border-router

A node as Border-Router

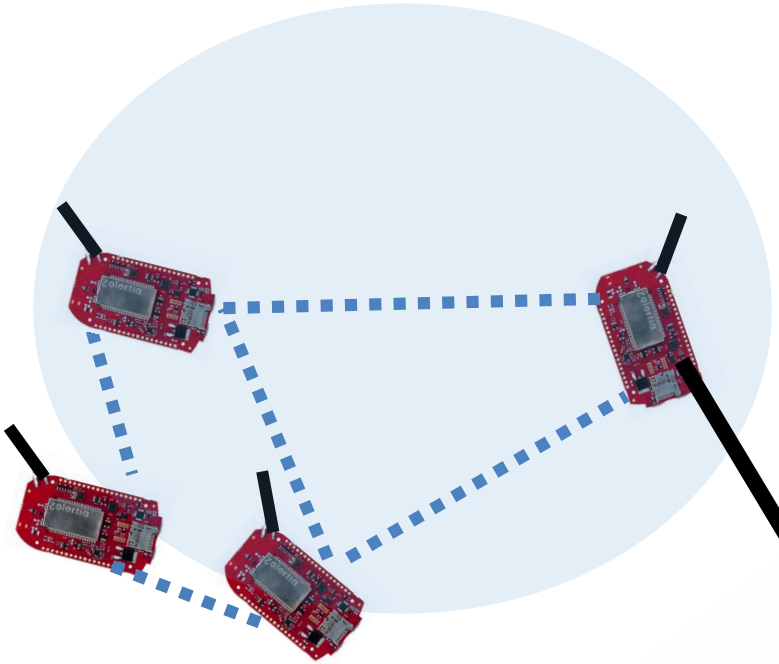


IEEE 802.15.4/6LoWPAN
wireless network

Border Router
The node talks to the host via USB, it receives a /64 prefix from tunslip6 and autoconfigures (SLAC). The node handles all the processing and it is limited by its resources

Tunnel interface – tun0
The *tunslip6* script creates a tunnel interface, forwards data from IPv6 to/from the 6LoWPAN network via USB

examples/zolertia/tutorial/02-ipv6/02-border-router



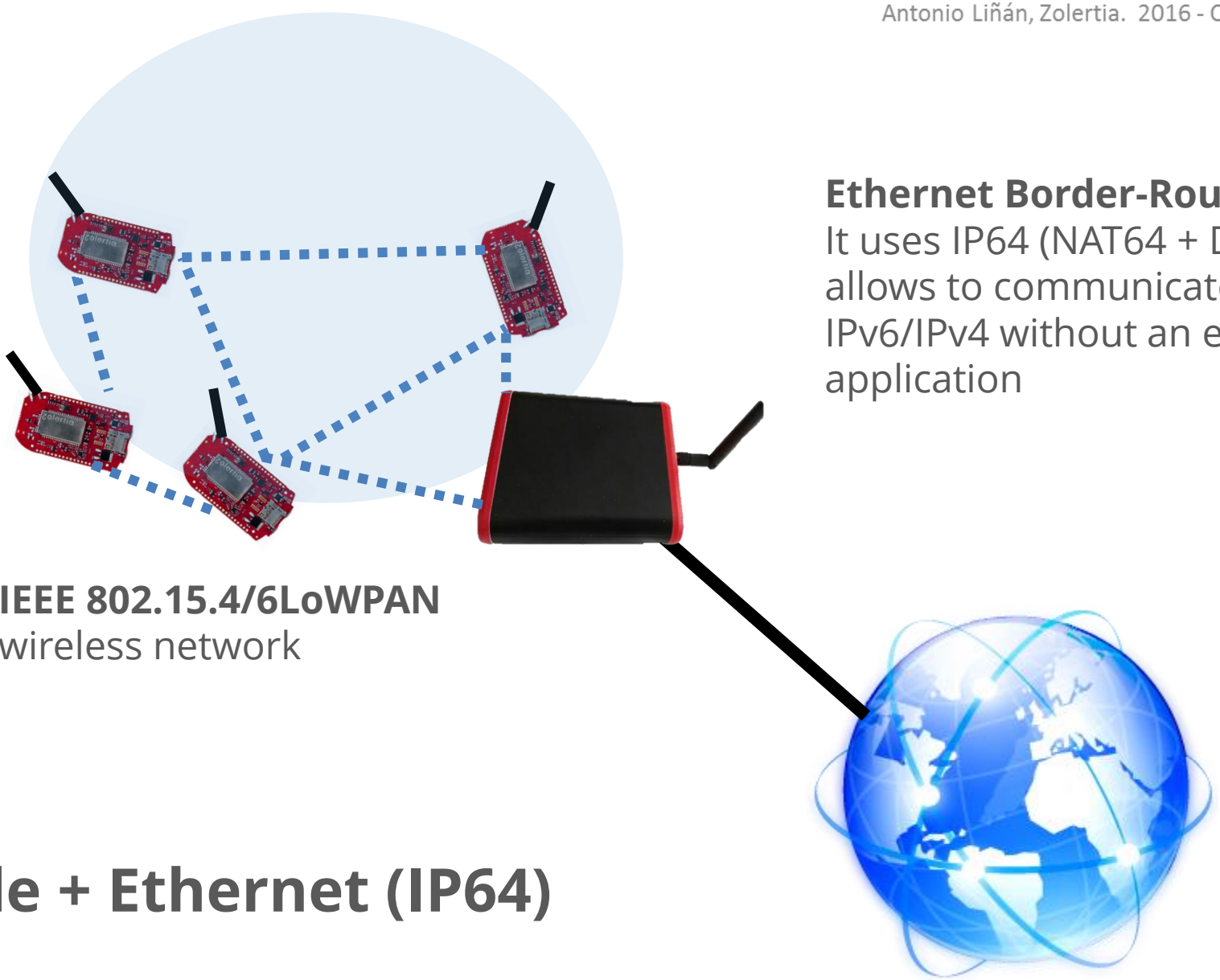
IEEE 802.15.4/6LoWPAN
wireless network

Slip Radio
The host controls the node through the USB, it receive commands to drive the radio

Native Border-Router
The host acts as the router, it has more routing and processing capabilities

A node as slip-radio, native Border-Router

examples/zolertia/tutorial/02-ipv6/02-border-router

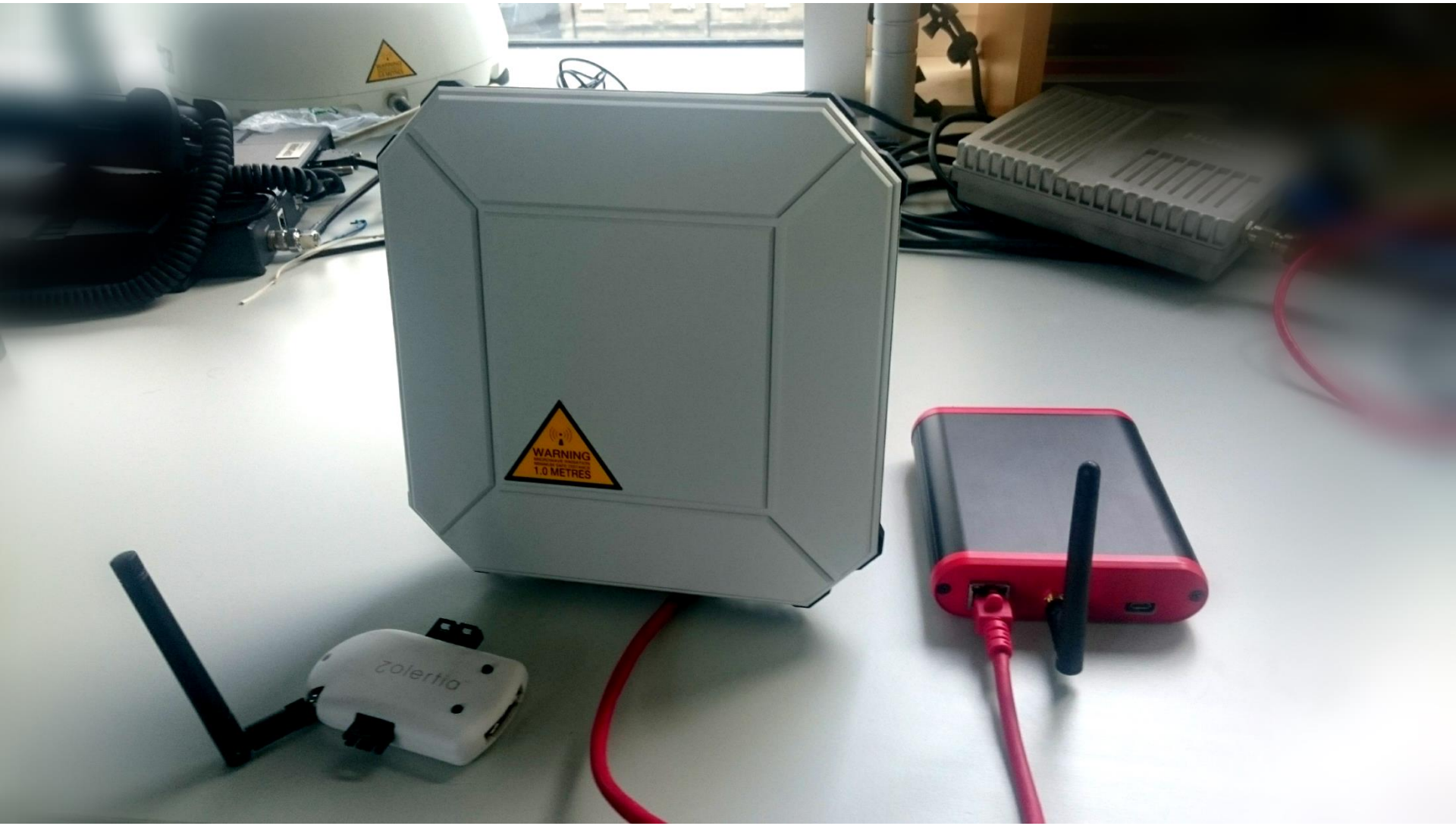


Ethernet Border-Router
It uses IP64 (NAT64 + DNS64), allows to communicate to IPv6/IPv4 without an external application

IEEE 802.15.4/6LoWPAN
wireless network

Node + Ethernet (IP64)

examples/zolertia/tutorial/02-ipv6/02-border-router



examples/zolertia/tutorial/02-ipv6/02-border-router

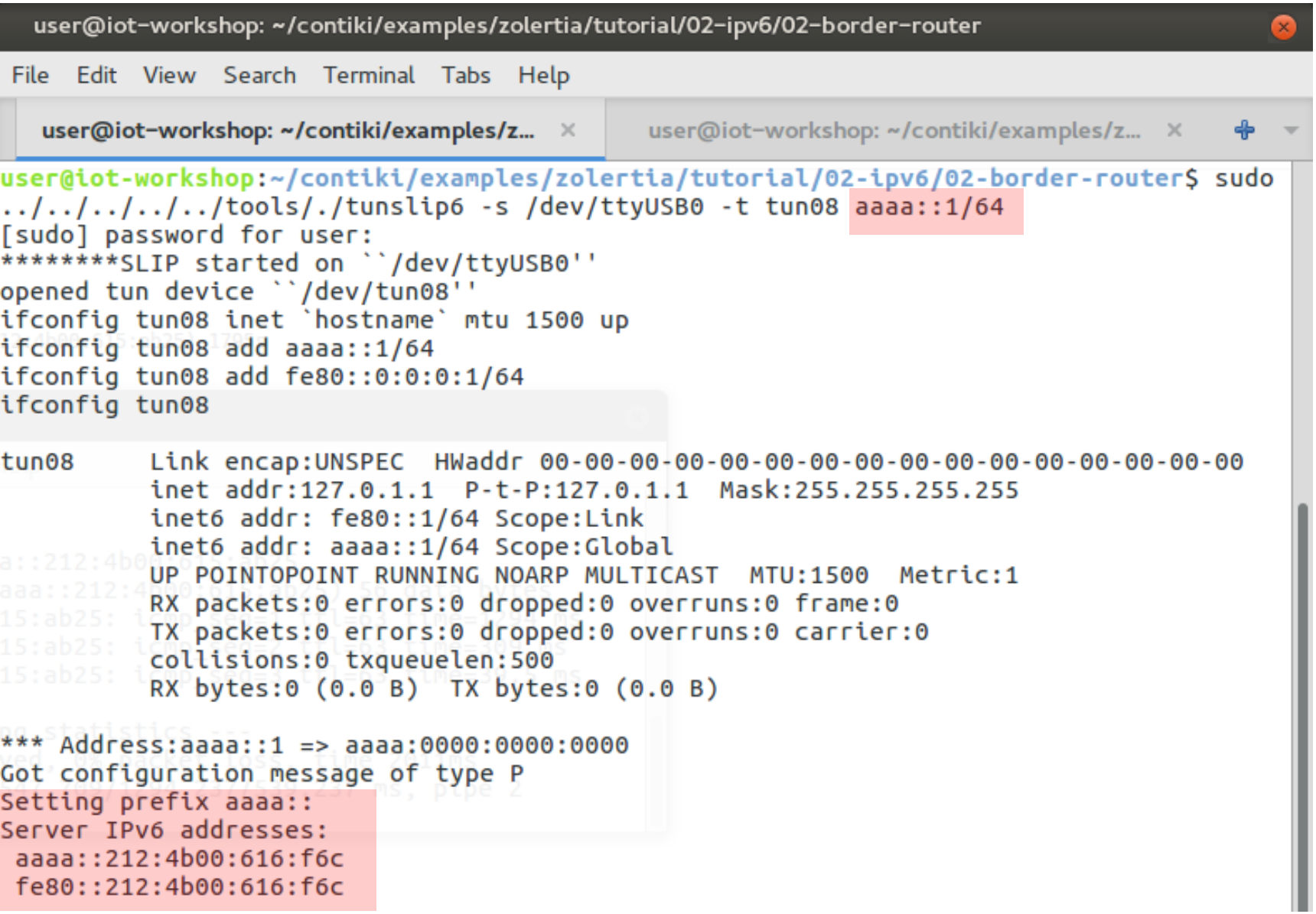
Connect a RE-Mote and run:

```
make border-router.upload && make login
```

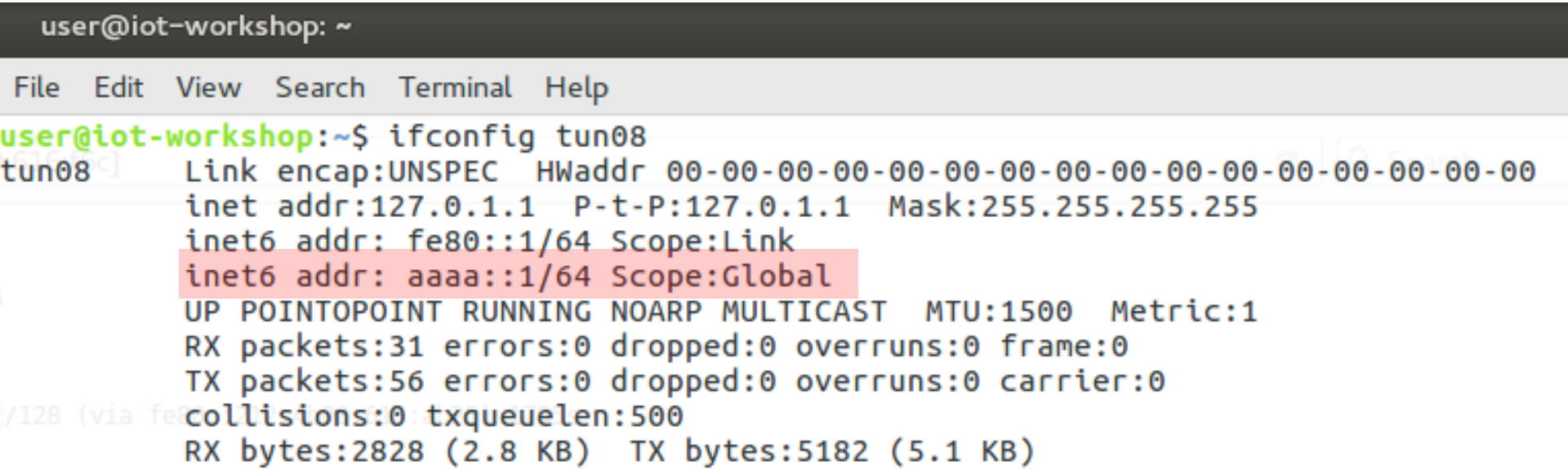
Then connect the Border Router to tunslip6 (don't close the terminal afterwards!):

```
cd ../../../../../tools
make tunslip6
sudo ./tunslip6 –s /dev/ttyUSB0 –t tun01 aaaa::1/64
```

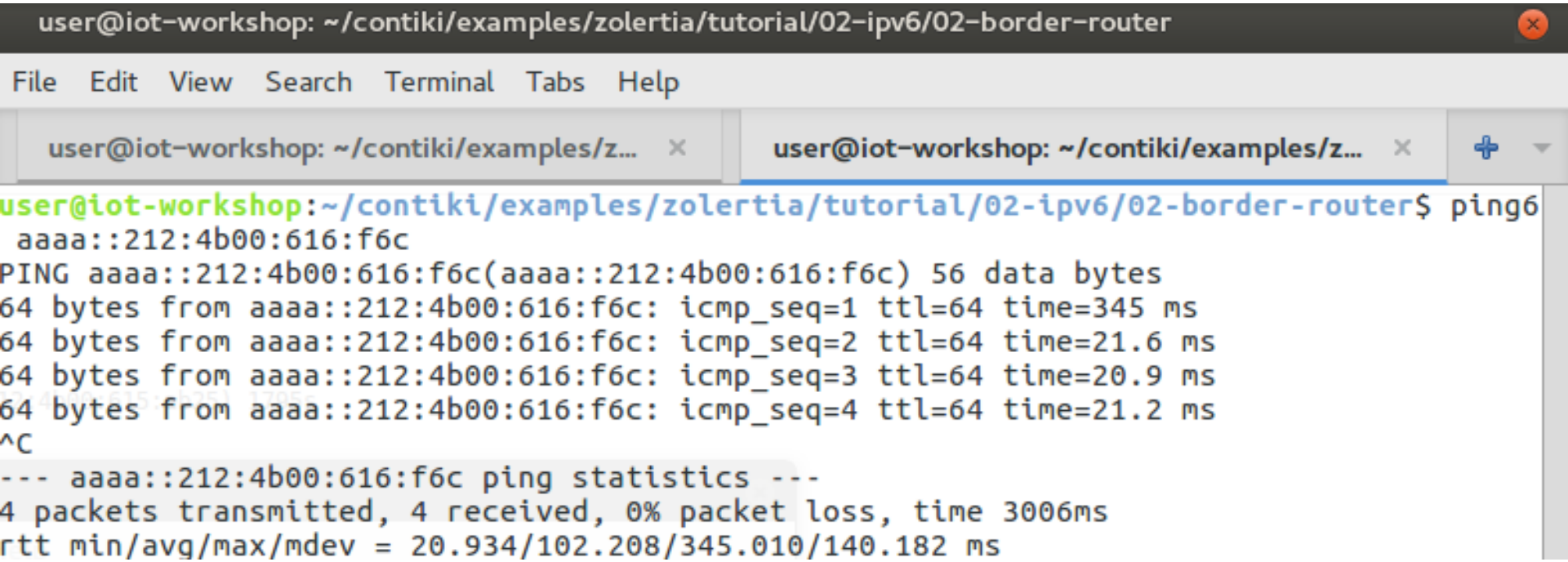
examples/zolertia/tutorial/02-ipv6/02-border-router



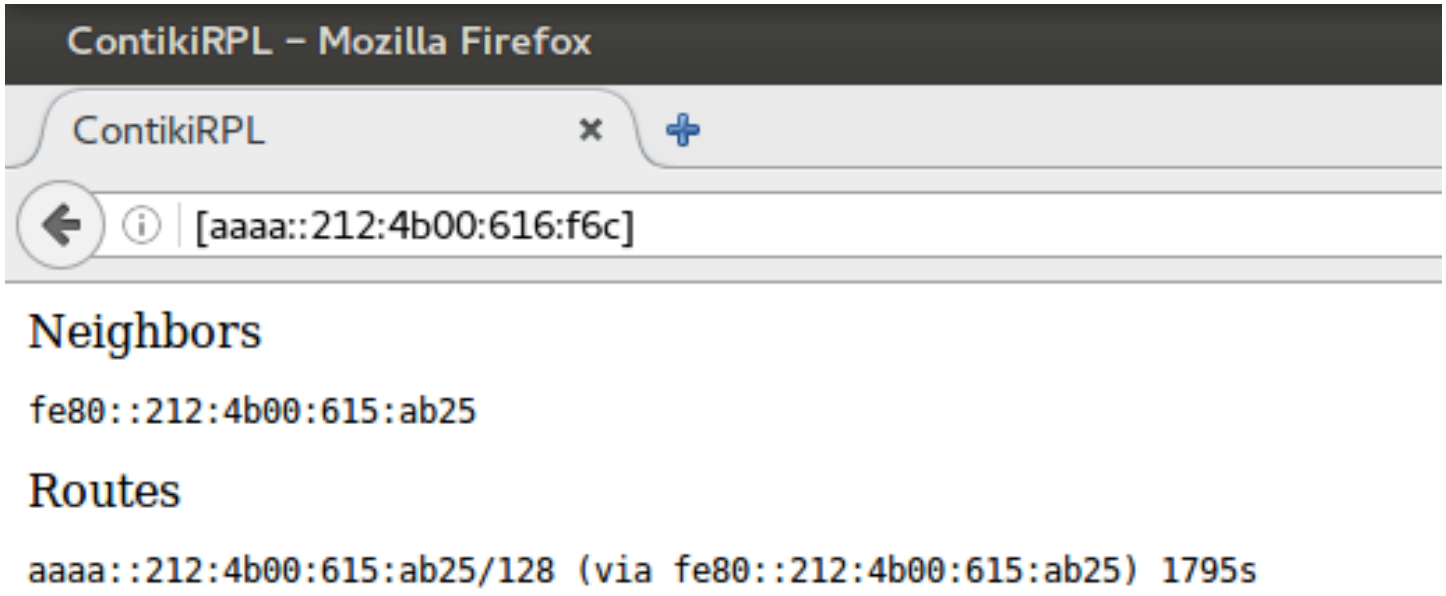
examples/zolertia/tutorial/02-ipv6/02-border-router



examples/zolertia/tutorial/02-ipv6/02-border-router



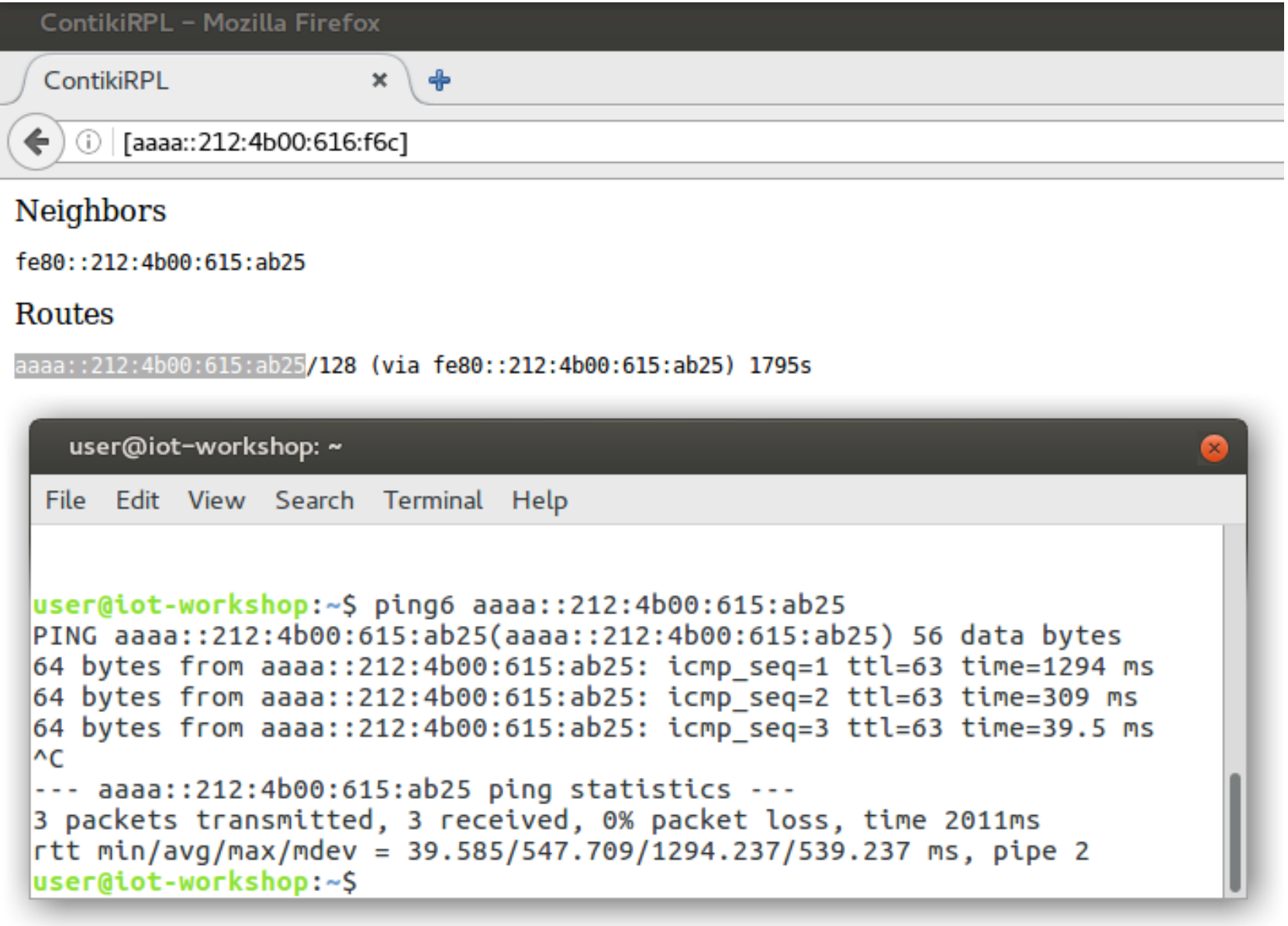
examples/zolertia/tutorial/02-ipv6/02-border-router



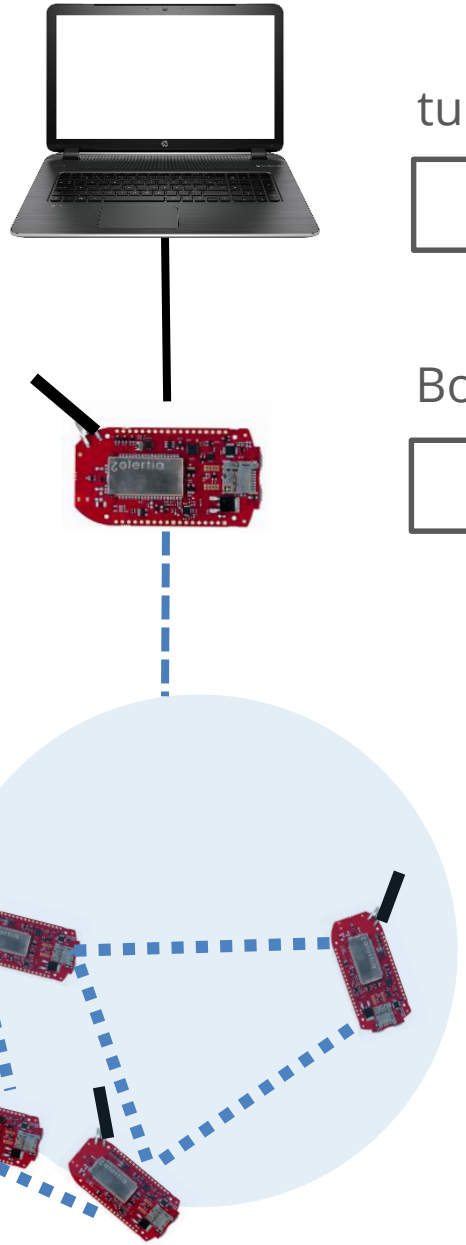
The Border Router's webserver shows its routing table, useful to verify what devices are in the 6LoWPAN network, its uptime, route to the device, etc.

Take another RE-Mote and program the 01-udp-local-multicast example, now the device will join a PAN (DODAG) and will be accesable from outside networks... remember to verify the channel and PAN ID matches the Border Router!

examples/zolertia/tutorial/02-ipv6/02-border-router



examples/zolertia/tutorial/02-ipv6/02-border-router

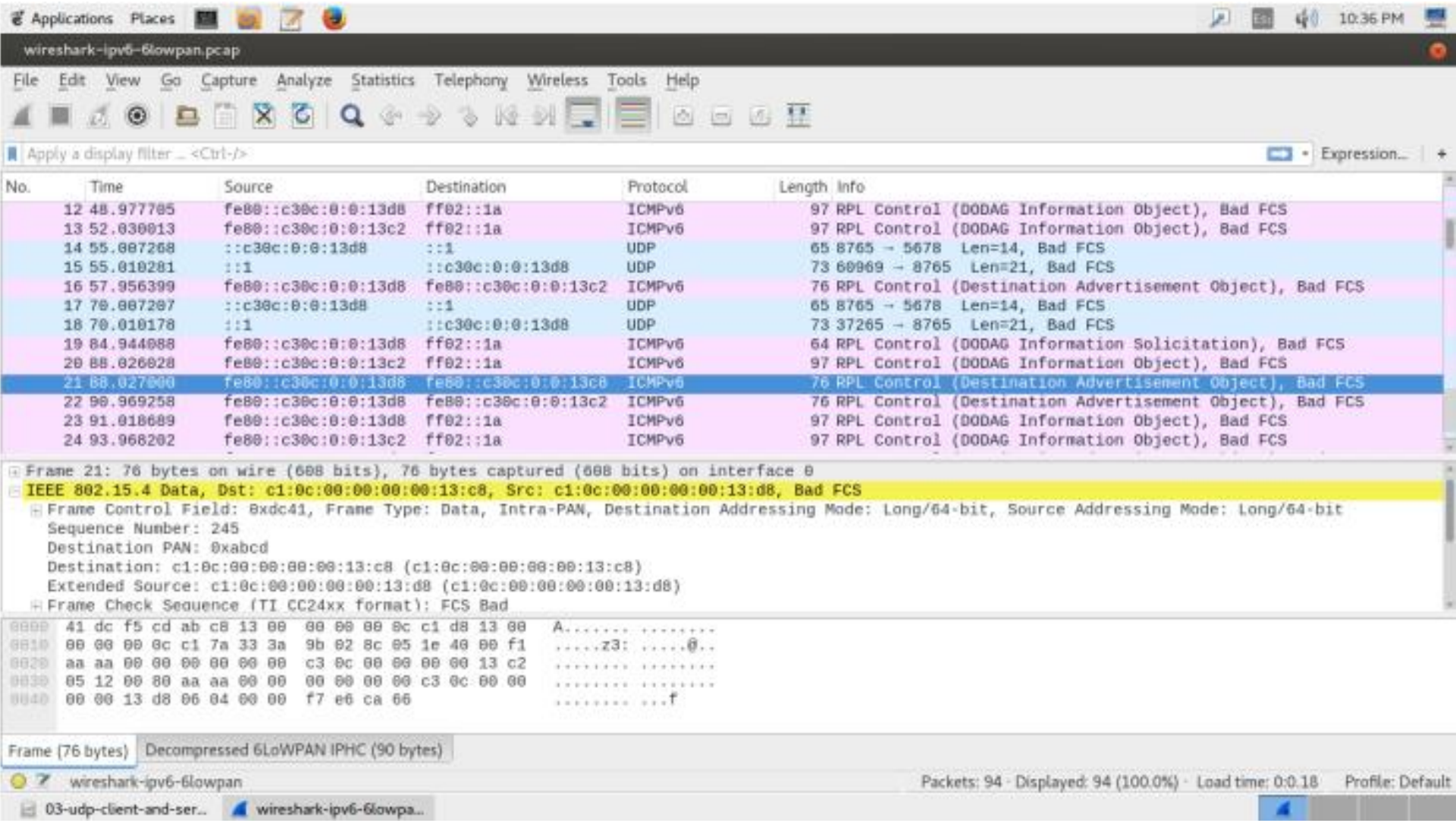


tunslip6 IPv6 global interface address

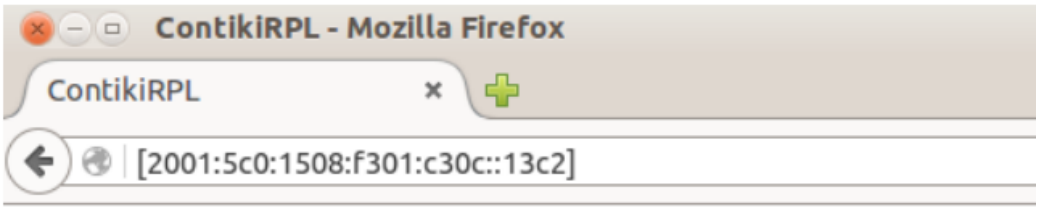
Border Router global IPv6 address

UDP node IPv6 global address

examples/zolertia/tutorial/02-ipv6/02-border-router



examples/zolertia/tutorial/02-ipv6/02-border-router



Neighbors

```
fe80::c30c:0:0:13c8
fe80::212:4b00:616:fd7
```

Routes

```
2001:5c0:1508:f301:c30c::13c8/128 (via fe80::c30c:0:0:13c8) 16711425s
```

ONLINE PING IPV6

PING = **P**acket **I**nter**N**et **G**rouper

This online IPv6 ping webtool is a computer network tool used to test whether a particular host is reachable across an IP network. It works by sending ICMP "echo request" packets to the target host and listening for ICMP "echo response" replies. ping estimates the round-trip time, generally in milliseconds, and records any packet loss, and prints a statistical summary when finished.

Source: **WikiPedia**

An IPv4 version of this webtool is **available here**!

IPv6 Ping Output:

```
PING 2001:5c0:1508:f301:c30c::13c2(2001:5c0:1508:f301:c30c::13c2) 32 data bytes
40 bytes from 2001:5c0:1508:f301:c30c::13c2: icmp_seq=0 ttl=54 time=65.8 ms
40 bytes from 2001:5c0:1508:f301:c30c::13c2: icmp_seq=1 ttl=54 time=63.7 ms
40 bytes from 2001:5c0:1508:f301:c30c::13c2: icmp_seq=2 ttl=54 time=64.3 ms
40 bytes from 2001:5c0:1508:f301:c30c::13c2: icmp_seq=3 ttl=54 time=64.7 ms

--- 2001:5c0:1508:f301:c30c::13c2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3011ms
rtt min/avg/max/mdev = 63.719/64.677/65.848/0.796 ms, pipe 2

---- Finished ------
```

examples/zolertia/tutorial/02-ipv6/02-border-router

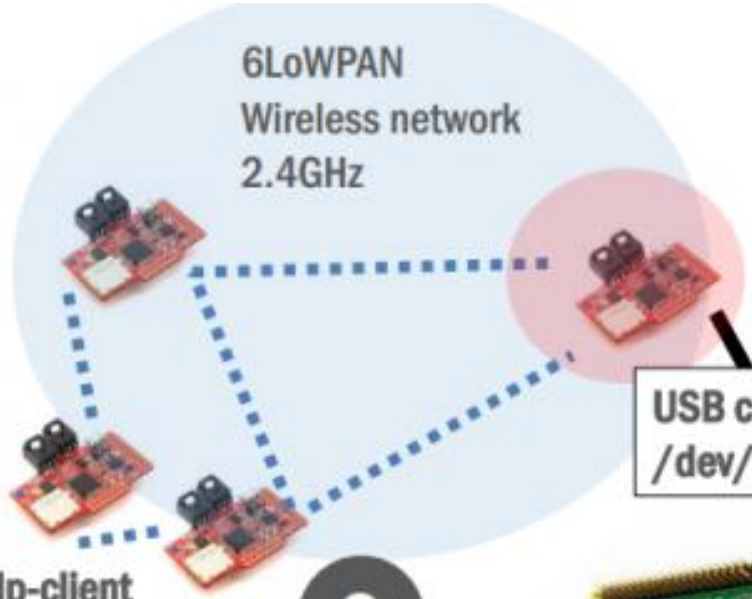
UDP client and server example

The next example will show how to create an UDP6 6LoWPAN wireless network, with devices acting as UDP clients connecting to an UDP server running in a host... locally or remotely!

At least two RE-Motes will be required (one acting as Border Router, and the other(s) as UDP clients). **A single Border Router will be hosted by the lecturer**

If you don't have an IPv6 network, this example will allow you to run the UDP server in your laptop, and forward information to servers and applications on Internet

examples/zolertia/tutorial/02-ipv6/udp-client-and-server



Border Router 2
Node ID: 0x1234
aaaa::c30c:0:0:1234
Receives the prefix from tunslip6 (over the USB) when the tunnel "tun0" is created with tunslip6

USB connection to /dev/ttyUSB0

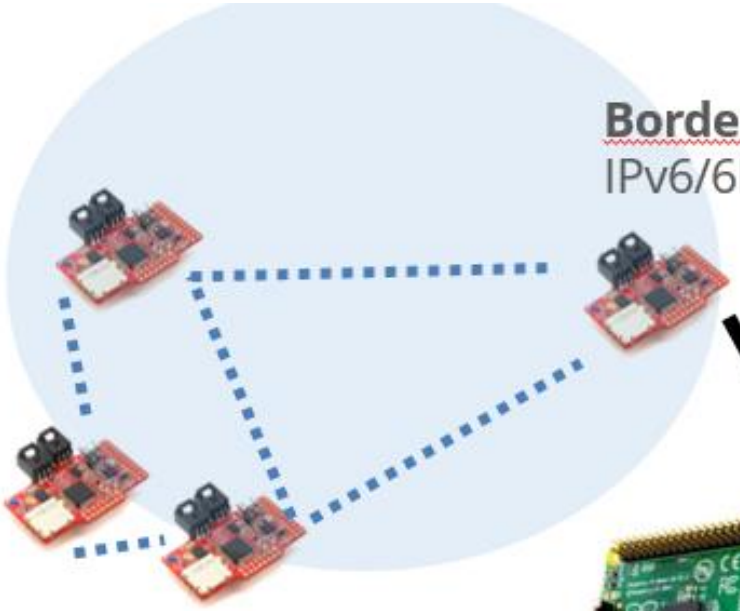
Tunnel interface "tun0" 1
aaaa::1/64
Created when running the tunslip6 script in the host. Is a virtual tunnel interface, it sends the aaaa::/64 prefix to the Border Router. The Raspberry Pi will have a "tun0" interface with an aaaa::1/64 address

03-udp-client 3
Node ID: 0x4567
aaaa::c30c:0:0:4567
Receives the aaaa::/64 prefix from the Border Router when joining the DAG.
Sends an UDP packet to aaaa::1

When the UDP Server runs in the host, it will use the same address as the host, in this case the aaaa::1/64. 4

In the "contiki/tools" location, to create a tunnel type:
sudo ./tunslip6 -s /dev/ttyUSB0 –t tun0 aaaa::1/64

examples/zolertia/tutorial/02-ipv6/udp-client-and-server



Border Router
IPv6/6LoWPAN

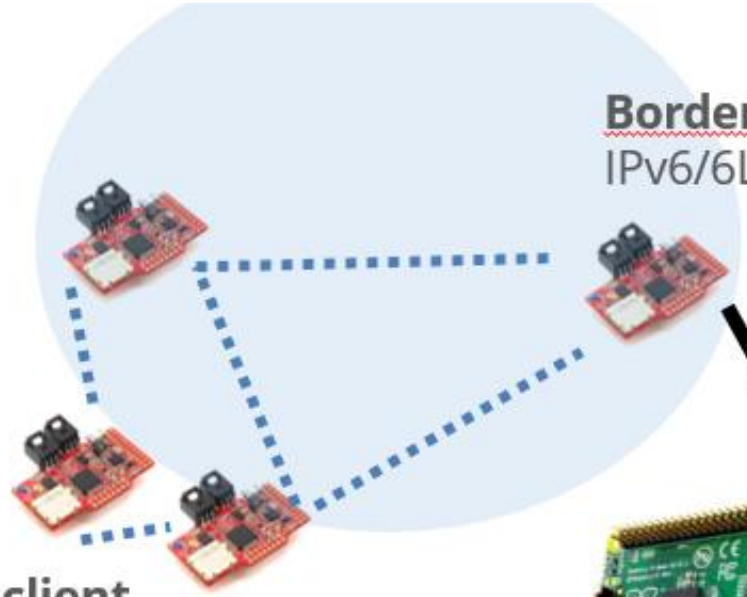
UDP-MQTT-server.py
Publish the received data to a topic at the MQTT broker

03-udp-client
Sends temperature, acceleration and battery data to the UDP server

mqtt-client.py
Subscribed to the topic, when the UDPServer publishes something we received the message

MQTT broker
iot.eclipse.org

examples/zolertia/tutorial/02-ipv6/udp-client-and-server



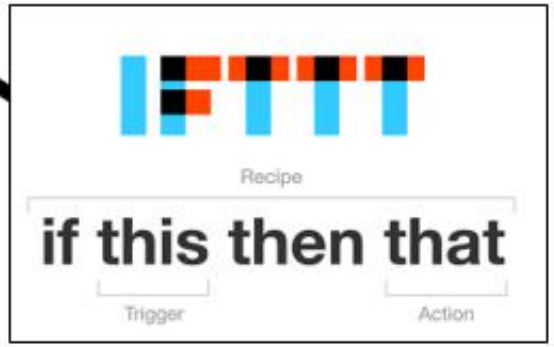
Border Router
IPv6/6LoWPAN

03-udp-client
Sends temperature, acceleration and battery data to the UDP server

UDP-IFTTT-server.py
Publish the received data to IFTTT's Maker channel

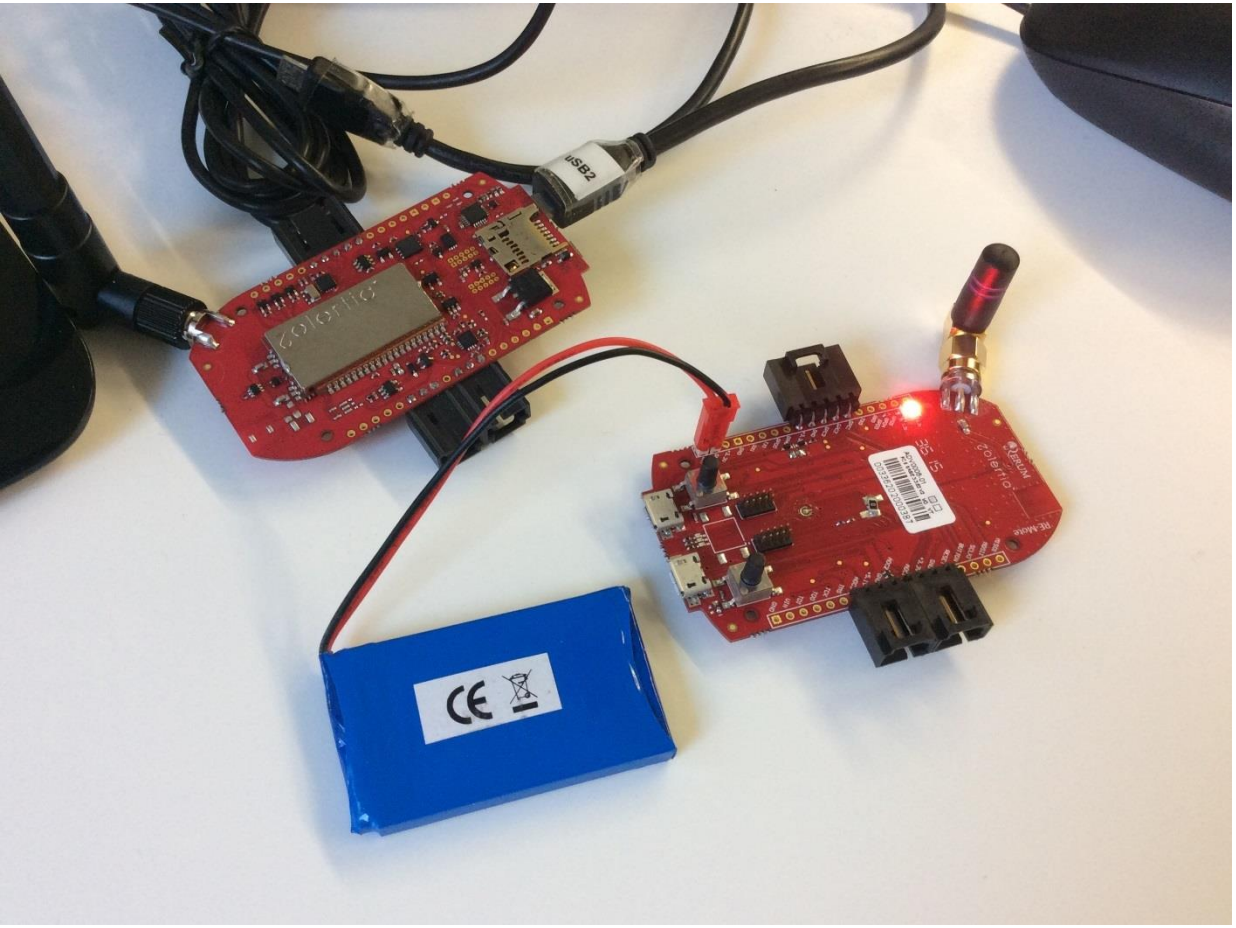
if  then 

- Press the Button
- Battery is below 3V
- Tampered
- Signal strenght is below optimal



IFTTT http://ifttt.com

examples/zolertia/tutorial/02-ipv6/udp-client-and-server



Connect a RE-Mote and program the example:

```
make 03-udp-client.upload && make login
```

examples/zolertia/tutorial/02-ipv6/03-udp-client-and-server

user@iot-workshop: ~/contiki/examples/zolertia/tutorial/02-ipv6/03-udp-client-and-server

File Edit View Search Terminal Help

```
Contiki-3.x-2614-g19f3dc5
Zolertia RE-Mote platform
CC2538: ID: 0xb964, rev.: PG2.0, Flash: 512 KiB, SRAM: 32 KiB, AES/SHA: 1, ECC/RSA:
System clock: 16000000 Hz
I/O clock: 16000000 Hz
Reset cause: External reset
Rime configured with address 00:12:4b:00:06:16:0f:6c
 Net: sicslowpan
 MAC: CSMA
 RDC: nullrdc
UDP client process started
Server address: fd00::1
Client IPv6 addresses:
fe80::212:4b00:616:f6c
Created a connection with the server :: local/remote port 8765/5678
ID: 171, core temp: 24.762, ADC1: 2308, ADC2: 0, ADC3: 1472, batt: 3272, counter: 1
Send readings to 1'
ID: 171, core temp: 24.762, ADC1: 2304, ADC2: 0, ADC3: 1468, batt: 3272, counter: 2
Send readings to 1'
ID: 171, core temp: 24.762, ADC1: 2300, ADC2: 0, ADC3: 1472, batt: 3270, counter: 3
Send readings to 1'
ID: 171, core temp: 24.762, ADC1: 2296, ADC2: 0, ADC3: 1472, batt: 3270, counter: 4
Send readings to 1'
```

examples/zolertia/tutorial/02-ipv6/udp-client-and-server

user@iot-workshop: ~/contiki/examples/zolertia/tutorial/02-ipv6/03-udp-client-and-server

File Edit View Search Terminal Help

```
Contiki-3.x-2614-g19f3dc5
/* The structure used in the Simple UDP library to create an UDP connection */
static struct uip_udp_conn *client_conn;

/* This is the server IPv6 address */
static uip_ipaddr_t server_ipaddr;
```

```
 Net: sicslowpan
 MAC: CSMA
 RDC: nullrdc
UDP client process started
Server address: fd00::1
Client IPv6 addresses:
fe80::212:4b00:616:f6c
Created a co
ID: 171, cor
Send reading
ID: 171, cor
Send reading
ID: 171, cor
Send reading
ID: 171, cor
Send readings to 1
```

Make sure this address matches the address of the tunslip6 interface

```
/* Set the server address here */
uip_ip6addr(&server_ipaddr, 0xfd00, 0, 0, 0, 0, 0, 0, 1);

printf("Server address: ");
PRINT6ADDR(&server_ipaddr);
printf("\n");
```

examples/zolertia/tutorial/02-ipv6/udp-client-and-server

user@iot-workshop: ~/contiki/examples/zolertia/tutorial/02-ipv6/03-udp-client-and-server

File Edit View Search Terminal Help

```
Contiki-3.x-2614-g19f3dc5
Zolertia RE-Mote platform
CC2538: ID: 0xb964, rev.: PG2.0, Flash: 512 KiB, SRAM: 32 KiB, AES/SHA: 1, ECC/RSA:
System clock: 16000000 Hz
I/O clock: 16000000 Hz
Reset cause: External reset
Rime configured with address
 Net: sicslowpan
 MAC: CSMA
 RDC: nullrdc
UDP client process started
Server address: fd00::1
Client IPv6 addresses:
fe80::212:4b00:616:f6c
Created a connection with the
ID: 171, core temp: 24.762, A
Send readings to 1'
ID: 171, core temp: 24.762, A
Send readings to 1'
ID: 171, core temp: 24.762, A
Send readings to 1'
ID: 171, core temp: 24.762, A
Send readings to 1'
```

Prints the device's addresses, only link-local at the moment as it haven't joined a DODAG yet

```
static void
print_local_addresses(void)
{
  int i;
  uint8_t state;

  PRINTF("Client IPv6 addresses:\n");
  for(i = 0; i < UIP_DS6_ADDR_NB; i++) {
    state = uip_ds6_if.addr_list[i].state;
    if(uip_ds6_if.addr_list[i].isused &&
       (state == ADDR_TENTATIVE || state == ADDR_PREFERRED)) {
      PRINT6ADDR(&uip_ds6_if.addr_list[i].ipaddr);
      PRINTF("\n");
      /* hack to make address "final" */
      if (state == ADDR_TENTATIVE) {
        uip_ds6_if.addr_list[i].state = ADDR_PREFERRED;
      }
    }
  }
}
```

```
/* Create a new connection with remote host.  When a connection is created
 * with udp_new(), it gets a local port number assigned automatically.
 * The "UIP_HTONS()" macro converts to network byte order.
 * The IP address of the remote host and the pointer to the data are not used
 * so those are set to NULL
 */
client_conn = udp_new(NULL, UIP_HTONS(UDP_SERVER_PORT), NULL);

if(client_conn == NULL) {
  PRINTF("No UDP connection available, exiting the process!\n");
  PROCESS_EXIT();
}

/* This function binds a UDP connection to a specified local port */
udp_bind(client_conn, UIP_HTONS(UDP_CLIENT_PORT));

PRINTF("Created a connection with the server ");
PRINT6ADDR(&client_conn->ripaddr);
PRINTF(" local/remote port %u/%u\n", UIP_HTONS(client_conn->lport),
                                     UIP_HTONS(client_conn->rport));
```

```
fe80::212:4b00:616:f6c
Created a connection with the server :: local/remote port 8765/5678
ID: 171, core temp: 24.762, ADC1: 2308, ADC2: 0, ADC3: 1472, batt: 3272, counter: 1
Send readings to 1'
ID: 171, core temp: 24.762, ADC1: 2304, ADC2: 0, ADC3: 1468, batt: 3272, counter: 2
Send readings to 1'
ID: 171, core temp: 24.762, ADC1: 2300, ADC2: 0, ADC3: 1472, batt: 3270, counter: 3
Send readings to 1'
ID: 171, core temp: 24.762, ADC1: 2296, ADC2: 0, ADC3: 1472, batt: 3270, counter: 4
Send readings to 1'
```

examples/zolertia/tutorial/02-ipv6/udp-client-and-server

```
while(1) {
  PROCESS_YIELD();

  /* Incoming events from the TCP/IP module */
  if(ev == tcpip_event) {
    tcpip_handler();
  }

  /* Send data to the server */
  if((ev == sensors_event && data == &button_sensor) ||
     (ev == PROCESS_EVENT_TIMER)) {

    send_packet();

    if(etimer_expired(&periodic)) {
      etimer_reset(&periodic);
    }
  }
}
```

```
Created a connection with the server :: local/remote port 8765/5678
ID: 171, core temp: 24.762, ADC1: 2308, ADC2: 0, ADC3: 1472, batt: 3272, counter: 1
Send readings to 1'
ID: 171, core temp: 24.762, ADC1: 2304, ADC2: 0, ADC3: 1468, batt: 3272, counter: 2
Send readings to 1'
ID: 171, core temp: 24.762, ADC1: 2300, ADC2: 0, ADC3: 1472, batt: 3270, counter: 3
Send readings to 1'
ID: 171, core temp: 24.762, ADC1: 2296, ADC2: 0, ADC3: 1472, batt: 3270, counter: 4
Send readings to 1'
```

examples/zolertia/tutorial/02-ipv6/udp-client-and-server

```c
static void
send_packet(void)
{
  uint32_t aux;
  counter++;

  msg.id      = 0xAB;
  msg.counter = counter;
  msg.value1  = cc2538_temp_sensor.value(CC2538_SENSORS_VALUE_TYPE_CONVERTED);
  msg.value2  = adc_zoul.value(ZOUL_SENSORS_ADC1);
  msg.value3  = adc_zoul.value(ZOUL_SENSORS_ADC2);
  msg.value4  = adc_zoul.value(ZOUL_SENSORS_ADC3);

  aux = vdd3_sensor.value(CC2538_SENSORS_VALUE_TYPE_CONVERTED);
  msg.battery = (uint16_t) aux;

  /* Print the sensor data */
  printf("ID: %u, core temp: %u.%u, ADC1: %d, ADC2: %d, ADC3: %d, batt: %u, counter: %u\n",
         msg.id, msg.value1 / 1000, msg.value1 % 1000, msg.value2, msg.value3,
         msg.value4, msg.battery, msg.counter);

  /* Convert to network byte order as expected by the UDPServer application */
  msg.counter = UIP_HTONS(msg.counter);
  msg.value1  = UIP_HTONS(msg.value1);
  msg.value2  = UIP_HTONS(msg.value2);
  msg.value3  = UIP_HTONS(msg.value3);
  msg.value4  = UIP_HTONS(msg.value4);
  msg.battery = UIP_HTONS(msg.battery);

  PRINTF("Send readings to %u'\n", server_ipaddr.u8[sizeof(server_ipaddr.u8) - 1]);

  uip_udp_packet_sendto(client_conn, msgPtr, sizeof(msg),
                        &server_ipaddr, UIP_HTONS(UDP_SERVER_PORT));
}
```

examples/zolertia/tutorial/02-ipv6/udp-client-and-server

On another terminal run the Border Router

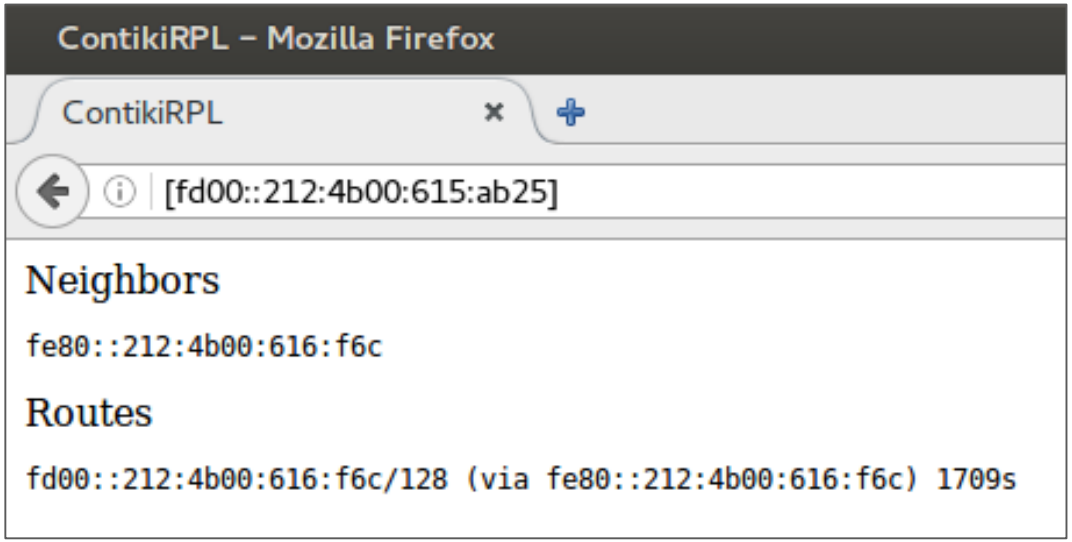
```
user@iot-workshop:~/contiki/examples/zolertia/tutorial/02-ipv6/02-border-router$ sudo ../
../../../../tools/./tunslip6 -s /dev/ttyUSB1 fd00::1/64
[sudo] password for user:
********SLIP started on ``/dev/ttyUSB1''
opened tun device ``/dev/tun0''
ifconfig tun0 inet `hostname` mtu 1500 up
ifconfig tun0 add fd00::1/64
ifconfig tun0 add fe80::0:0:0:1/64
ifconfig tun0

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:127.0.1.1  P-t-P:127.0.1.1  Mask:255.255.255.255
          inet6 addr: fd00::1/64 Scope:Global
          inet6 addr: fe80::1/64 Scope:Link
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

*** Address:fd00::1 => fd00:0000:0000:0000
Got configuration message of type P
Setting prefix fd00::
Server IPv6 addresses:
 fd00::212:4b00:615:ab25
 fe80::212:4b00:615:ab25
```

examples/zolertia/tutorial/02-ipv6/udp-client-and-server

Verify the UDP client is in our network and responsive

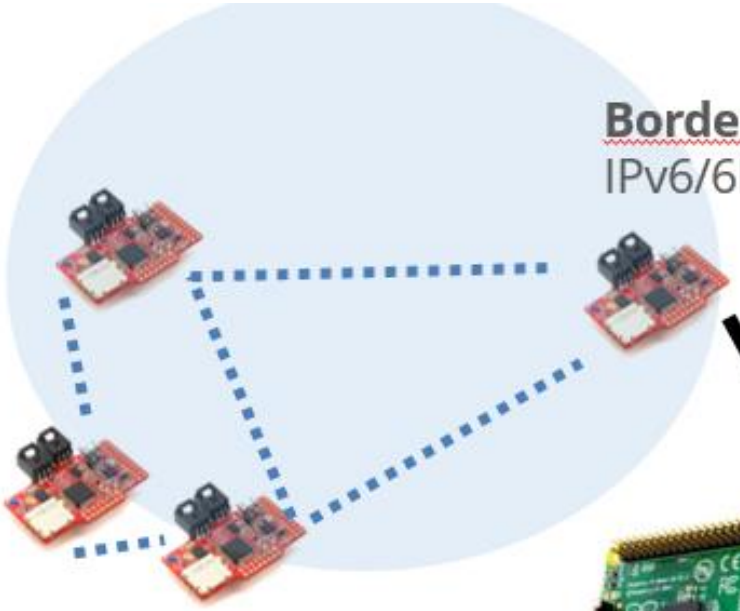



```
user@iot-workshop:~/contiki/examples/zolertia/tutorial/02-ipv6/03-udp-client-and-server$
ping6 fd00::212:4b00:616:f6c
PING fd00::212:4b00:616:f6c(fd00::212:4b00:616:f6c) 56 data bytes
64 bytes from fd00::212:4b00:616:f6c: icmp_seq=1 ttl=63 time=292 ms
64 bytes from fd00::212:4b00:616:f6c: icmp_seq=2 ttl=63 time=32.6 ms
64 bytes from fd00::212:4b00:616:f6c: icmp_seq=3 ttl=63 time=32.5 ms
64 bytes from fd00::212:4b00:616:f6c: icmp_seq=4 ttl=63 time=31.7 ms
64 bytes from fd00::212:4b00:616:f6c: icmp_seq=5 ttl=63 time=32.5 ms
^C
--- fd00::212:4b00:616:f6c ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4010ms
rtt min/avg/max/mdev = 31.791/84.430/292.627/104.099 ms
```

examples/zolertia/tutorial/02-ipv6/udp-client-and-server

We can verify using **netcat** UDP6 packets being received, but content is not decoded. Press the user button and a packet will be sent each time

```
user@iot-workshop:~/contiki/examples/zolertia/tutorial/02-ipv6/03-udp-client-and-server$
nc -ul6 5678
◆pg<
      ϙqg< ◆
           ţrg◆◆
                ϙsh*      ◆
                          ◆^C
```



Border Router
IPv6/6LoWPAN

UDP-MQTT-server.py
Publish the received data to a topic at the MQTT broker

03-udp-client
Sends temperature, acceleration and battery data to the UDP server

mqtt-client.py
Subscribed to the topic, when the UDPServer publishes something we received the message

MQTT broker
iot.eclipse.org

examples/zolertia/tutorial/02-ipv6/udp-client-and-server

```
UDP6-MQTT server side application V0.1
Started 2016-07-12 05:06:04.018618
UDP6-MQTT server ready: 5678
msg structure size:  13

MQTT: Connected (0)
2016-07-12 05:06:08 -> fd00::212:4b00:616:f6c:8765 14
{
  "values": [
    {
      "value": 171,
      "key": "id"
    },
    {
      "value": 0,
      "key": "counter"
    },
    {
      "value": 26493,
      "key": "core_temp"
    },
    {
      "value": 2364,
      "key": "ADC1"
    },
    {
      "value": 0,
      "key": "ADC2"
    },
    {
      "value": 1280,
      "key": "ADC3"
    },
    {
      "value": 3264,
      "key": "battery"
    }
  ]
}
MQTT: Publishing to {0}... 0 (171)
```

```
python UDP-MQTT-server.py
```

examples/zolertia/tutorial/02-ipv6/udp-client-and-server/UDP-MQTT-server.py

```
PORT                = 5678
CMD_PORT            = 8765
BUFSIZE             = 1024
# If using a client based on the Z1 mote, then enable by equal to 1, else if
# using the RE-Mote equal to 0
EXAMPLE_WITH_Z1     = 0
#---------------------------------------------------------#
MQTT_URL            = "iot.eclipse.org"
MQTT_PORT           = 1883
MQTT_KEEPALIVE      = 60
MQTT_URL_PUB        = "v2/zolertia/tutorialthings/"
MQTT_URL_TOPIC      = "/cmd"
```

```
user@iot-workshop:~/contiki/examples/zolertia/tutorial/02-ipv6/03-udp-client-and-server$
 python mqtt-client.py
connecting to iot.eclipse.org
Connected with result code 0
Subscribed to v2/zolertia/tutorialthings/#
v2/zolertia/tutorialthings this is a test
v2/zolertia/tutorialthings/171 {"values":[{"key": "id", "value": 171},{"key": "counter",
 "value": 0},{"key": "core_temp", "value": 25738},{"key": "ADC1", "value": 2418},{"key":
 "ADC2", "value": 4},{"key": "ADC3", "value": 1280},{"key": "battery", "value": 3264}]}
```

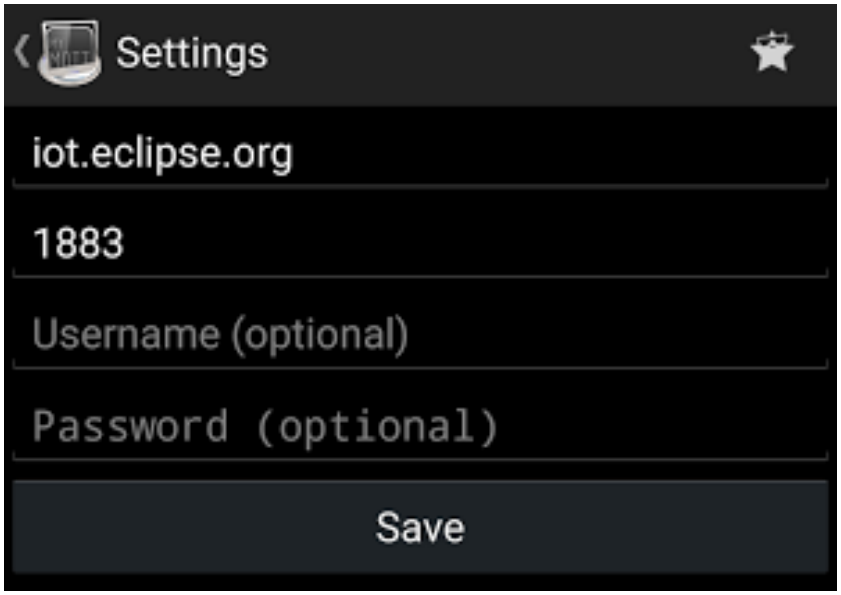
examples/zolertia/tutorial/02-ipv6/udp-client-and-server/UDP-MQTT-server.py

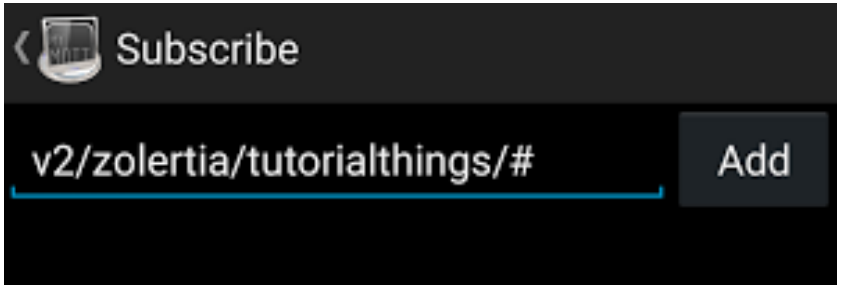
MyMQTT
instant:solutions
PEGI 3

UNINSTALL OPEN

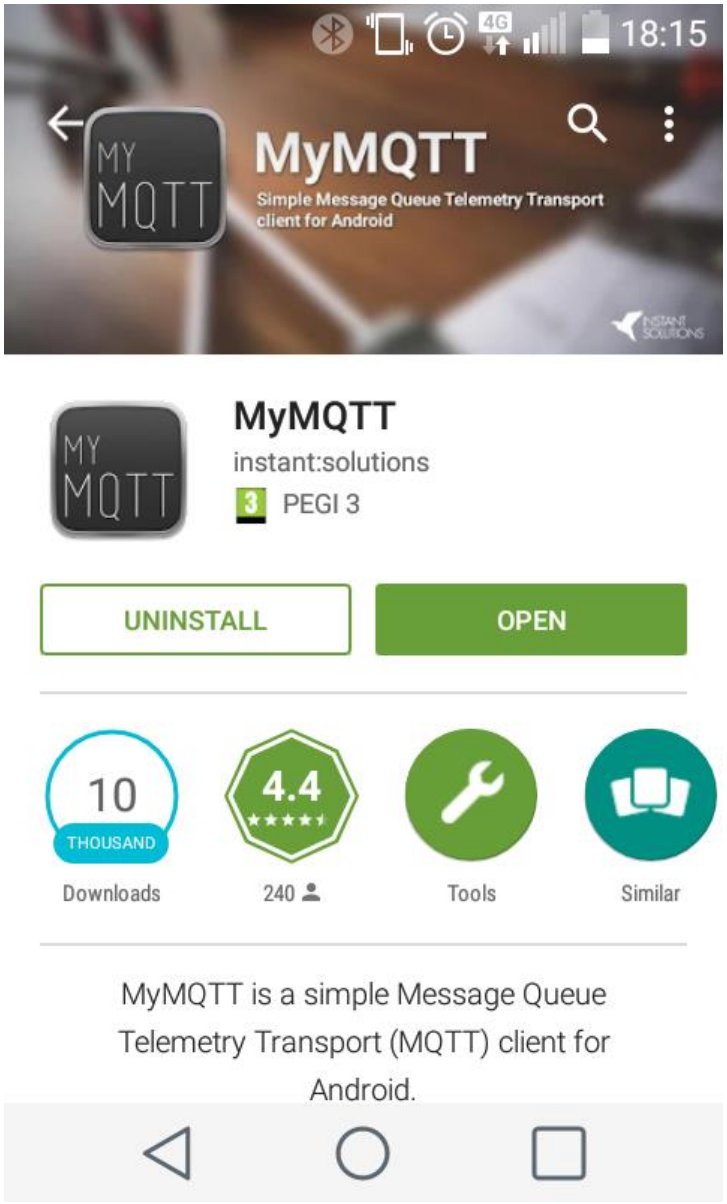
10 THOUSAND Downloads | 4.4 240 | Tools | Similar

MyMQTT is a simple Message Queue Telemetry Transport (MQTT) client for Android.

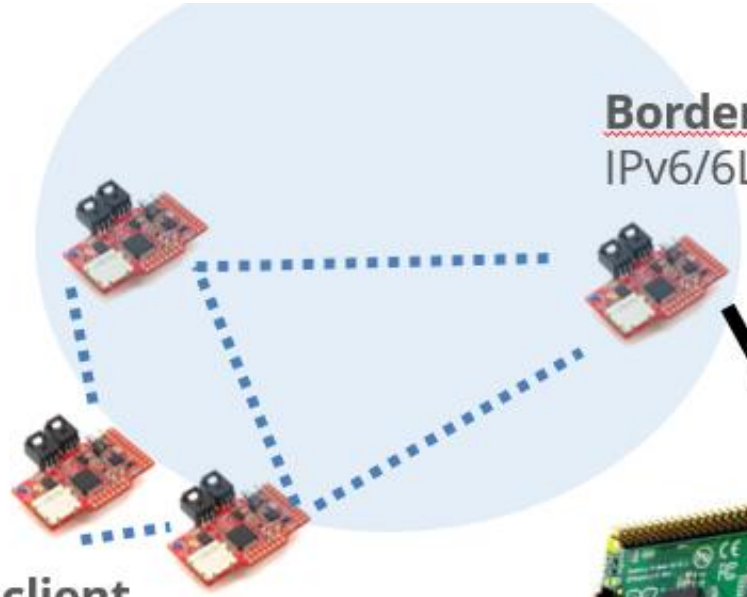




examples/zolertia/tutorial/02-ipv6/udp-client-and-server/UDP-MQTT-server.py



examples/zolertia/tutorial/02-ipv6/udp-client-and-server/UDP-MQTT-server.py



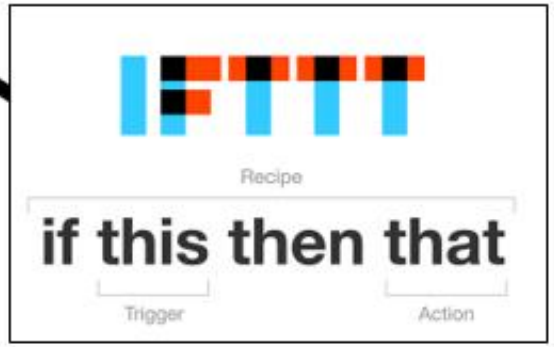
Border Router
IPv6/6LoWPAN

03-udp-client
Sends temperature, acceleration and battery data to the UDP server

UDP-IFTTT-server.py
Publish the received data to IFTTT's Maker channel

if  then 

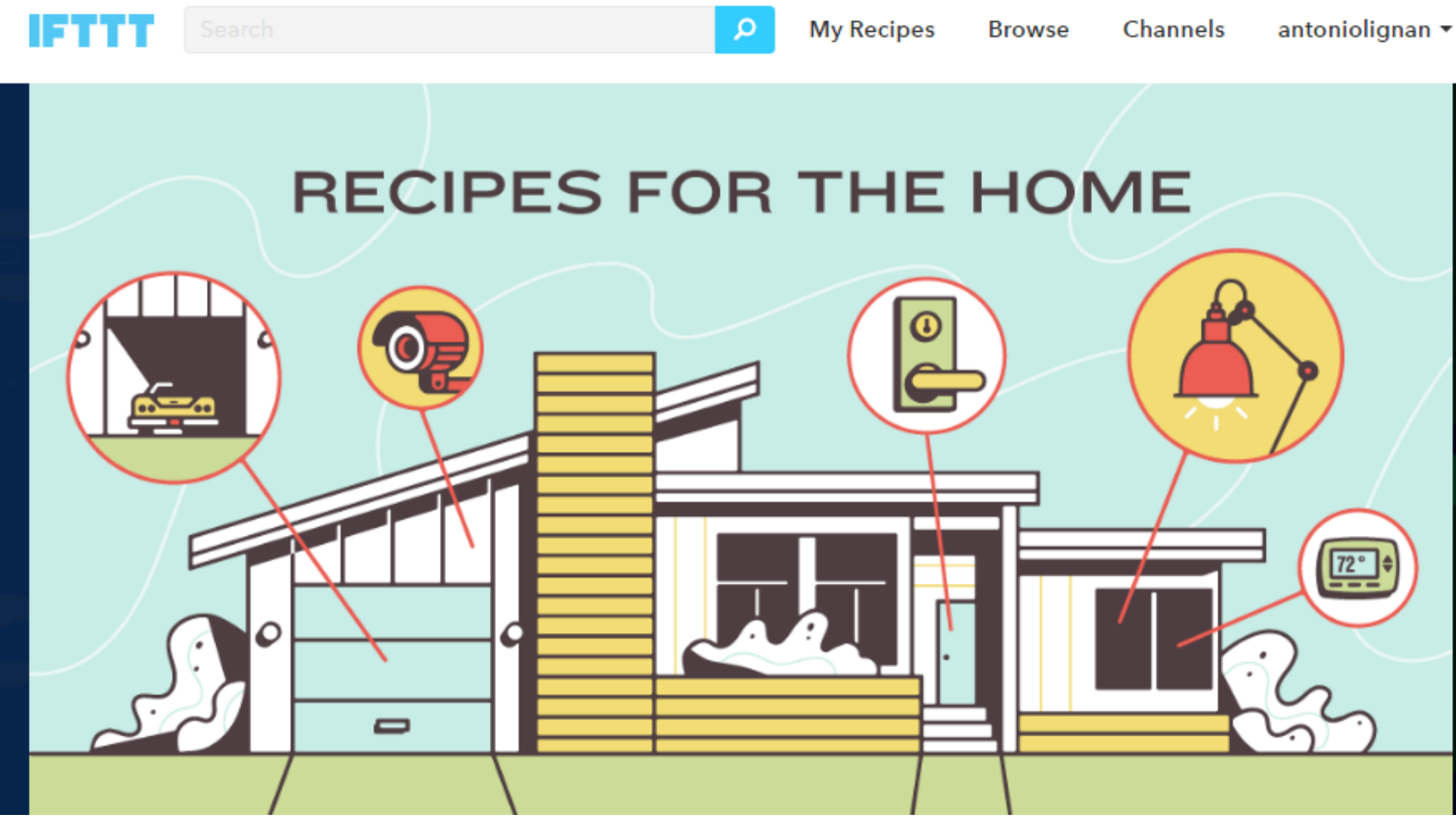
- Press the Button
- Battery is below 3V
- Tampered
- Signal strenght is below optimal



IFTTT http://ifttt.com

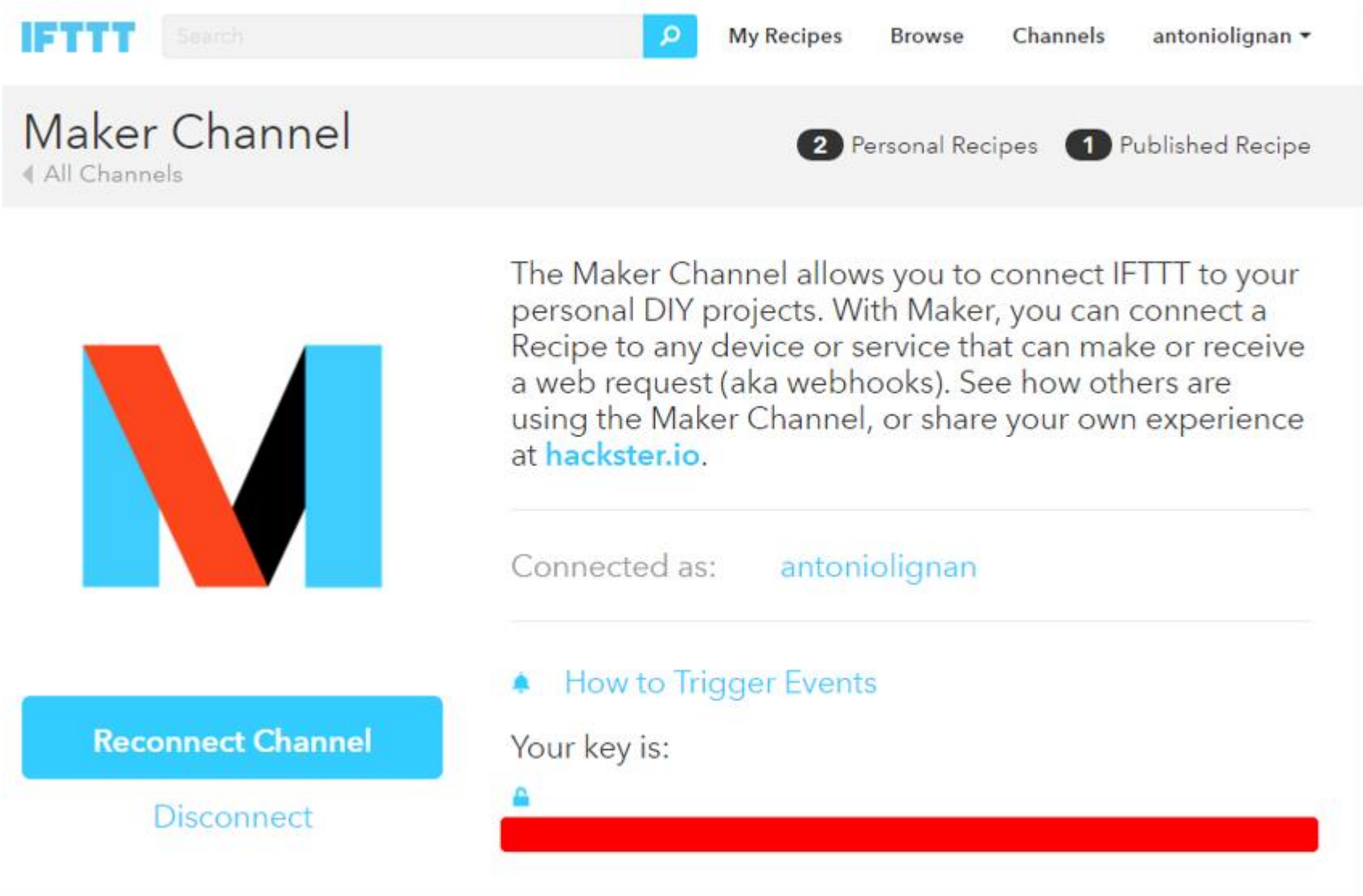
examples/zolertia/tutorial/02-ipv6/udp-client-and-server

examples/zolertia/tutorial/02-ipv6/udp-client-and-server/UDP-IFTTT-server.py



http://ifttt.com

examples/zolertia/tutorial/02-ipv6/udp-client-and-server/UDP-IFTTT-server.py



https://ifttt.com/maker

examples/zolertia/tutorial/02-ipv6/udp-client-and-server/UDP-IFTTT-server.py

Your key is: [redacted]

Back to Channel

To trigger an Event

Make a POST or GET web request to: **The name of your event**

```
https://maker.ifttt.com/trigger/event/with/key/[redacted]
```

With an optional JSON body of:

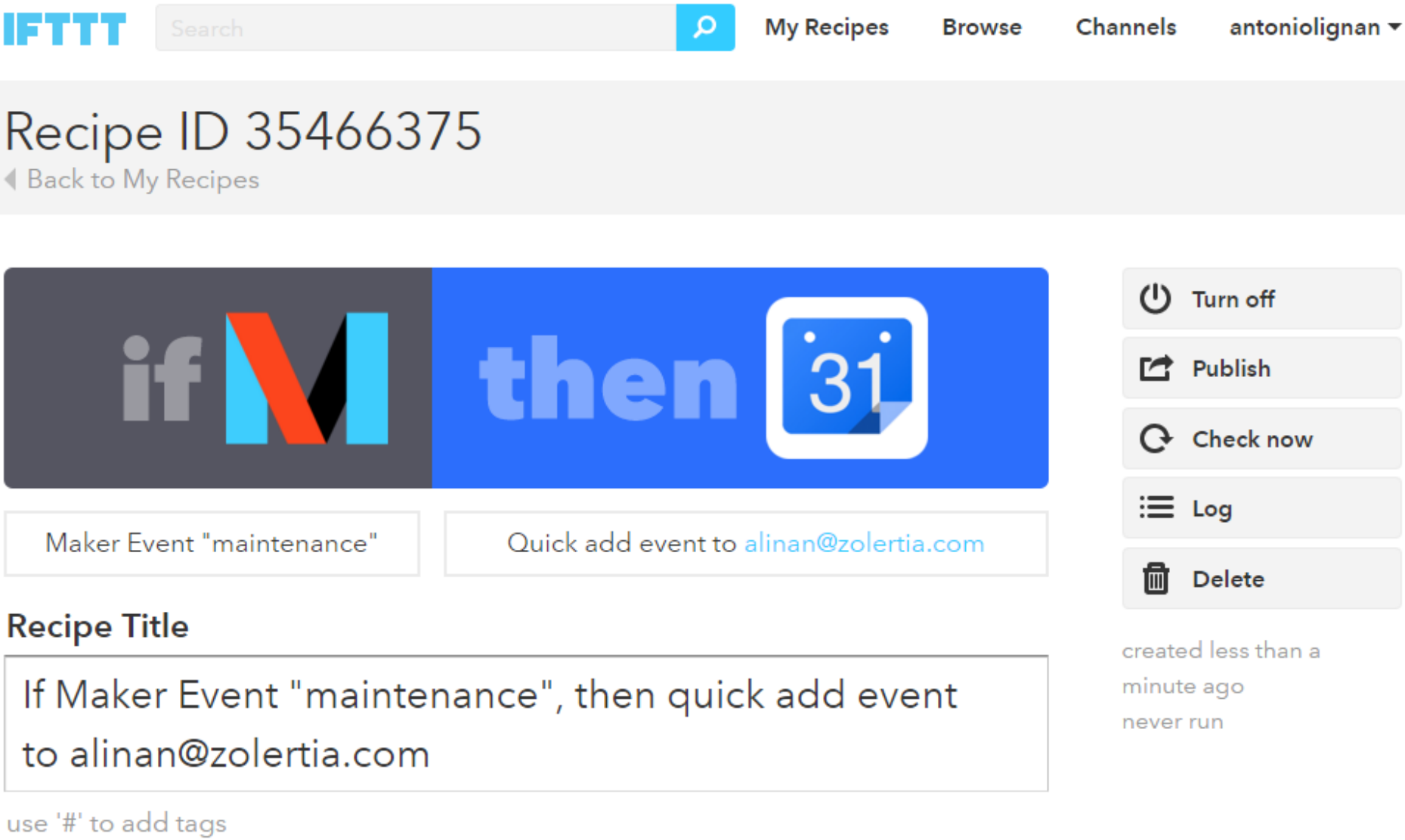
```
{ "value1" : "", "value2" : "", "value3" : "" }
```

The data is completely optional, and you can also pass `value1`, `value2`, and `value3` as query parameters or form variables. This content will be passed on to the Action in your Recipe.

You can also try it with `curl` from a command line.

```
curl -X POST https://maker.ifttt.com/trigger/event/with/key/[redacted]
```

examples/zolertia/tutorial/02-ipv6/udp-client-and-server/UDP-IFTTT-server.py



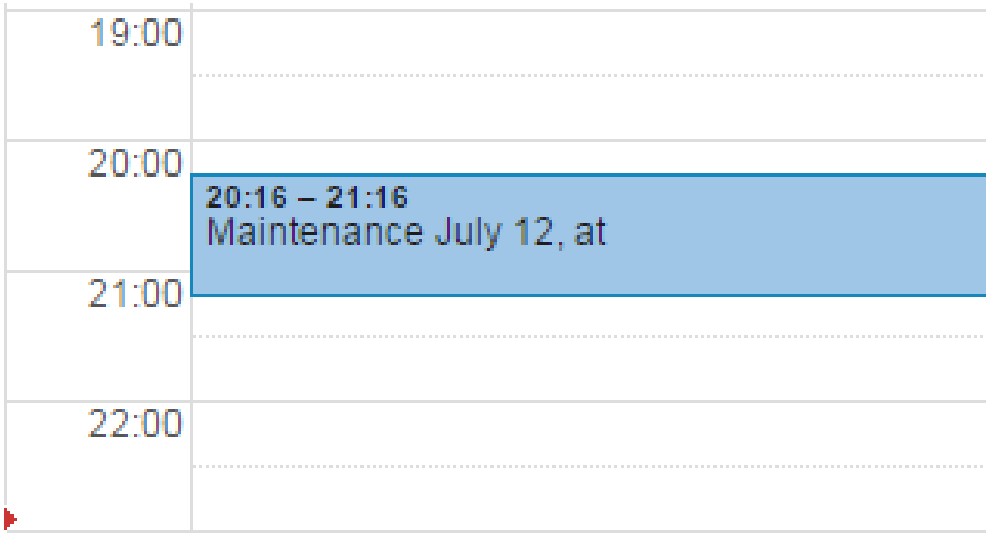
examples/zolertia/tutorial/02-ipv6/udp-client-and-server/UDP-IFTTT-server.py

```
PORT                = 5678
CMD_PORT            = 8765
BUFSIZE             = 1024
#---------------------------------------------------------------#
# If using a client based on the Z1 mote, then enable by equal to 1, else if
# using the RE-Mote equal to 0
EXAMPLE_WITH_Z1     = 0
#---------------------------------------------------------------#
IFTTT_URL           = "https://maker.ifttt.com/trigger/"
IFTTT_EVENT         = "maintenance"
IFTTT_KEY           = ""
```

```
user@iot-workshop:~/contiki/examples/zolertia/tutorial/02-ipv6/03-udp-client-and-server$
 python UDP-IFTTT-server.py
UDP6-IFTTT server side application V0.1
Started 2016-07-12 05:48:15.460651
UDP6-IFTTT server ready: 5678
msg structure size:  13

2016-07-12 05:48:18 -> fd00::212:4b00:616:f6c:8765 14
***
id:171  counter:0  core_temp:26800  ADC1:2346  ADC2:0  ADC3:1280  battery:3264
***
```

examples/zolertia/tutorial/02-ipv6/udp-client-and-server/UDP-IFTTT-server.py




```
user@iot-workshop:~/contiki/examples/zolertia/tutorial/02-ipv6/03-udp-client-and-server$
 python UDP-IFTTT-server.py
UDP6-IFTTT server side application V0.1
Started 2016-07-12 05:48:15.460651
UDP6-IFTTT server ready: 5678
msg structure size:  13

2016-07-12 05:48:18 -> fd00::212:4b00:616:f6c:8765 14
***
id:171  counter:0  core_temp:26800  ADC1:2346  ADC2:0  ADC3:1280  battery:3264
***
```

examples/zolertia/tutorial/02-ipv6/udp-client-and-server/UDP-IFTTT-server.py

Conclusions

You should be able to:

- Connect 6LoWPAN Wireless networks
- Understand how a Border Router Works
- Understand how networks are created using RPL
- Create UDP applications
- Use ping6 to assert the device's connectivity
- Use the Border Router's webservice to check routing table
- Forward data from UDP applications to other services and protocols such as MQTT and IFTTT

Antonio Liñán Colina

alinan@zolertia.com
antonio.lignan@gmail.com

Twitter: @4Li6NaN
LinkedIn: Antonio Liñan Colina
github.com/alignan
hackster.io/alinan