Contiki - CoAP and MQTT

Antonio Liñán Colina

Contiki

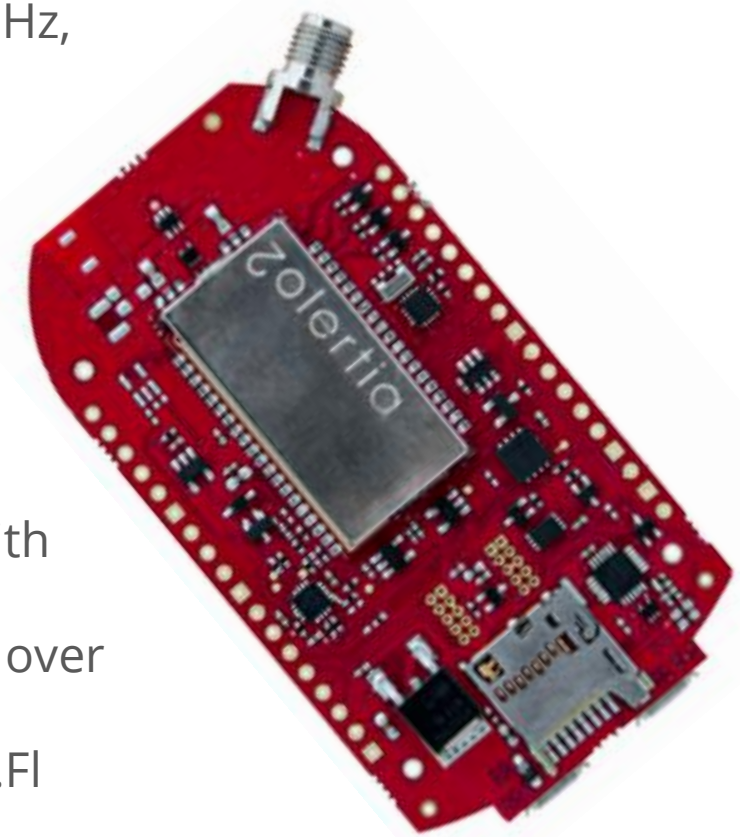
The Open Source OS for the Internet of Things

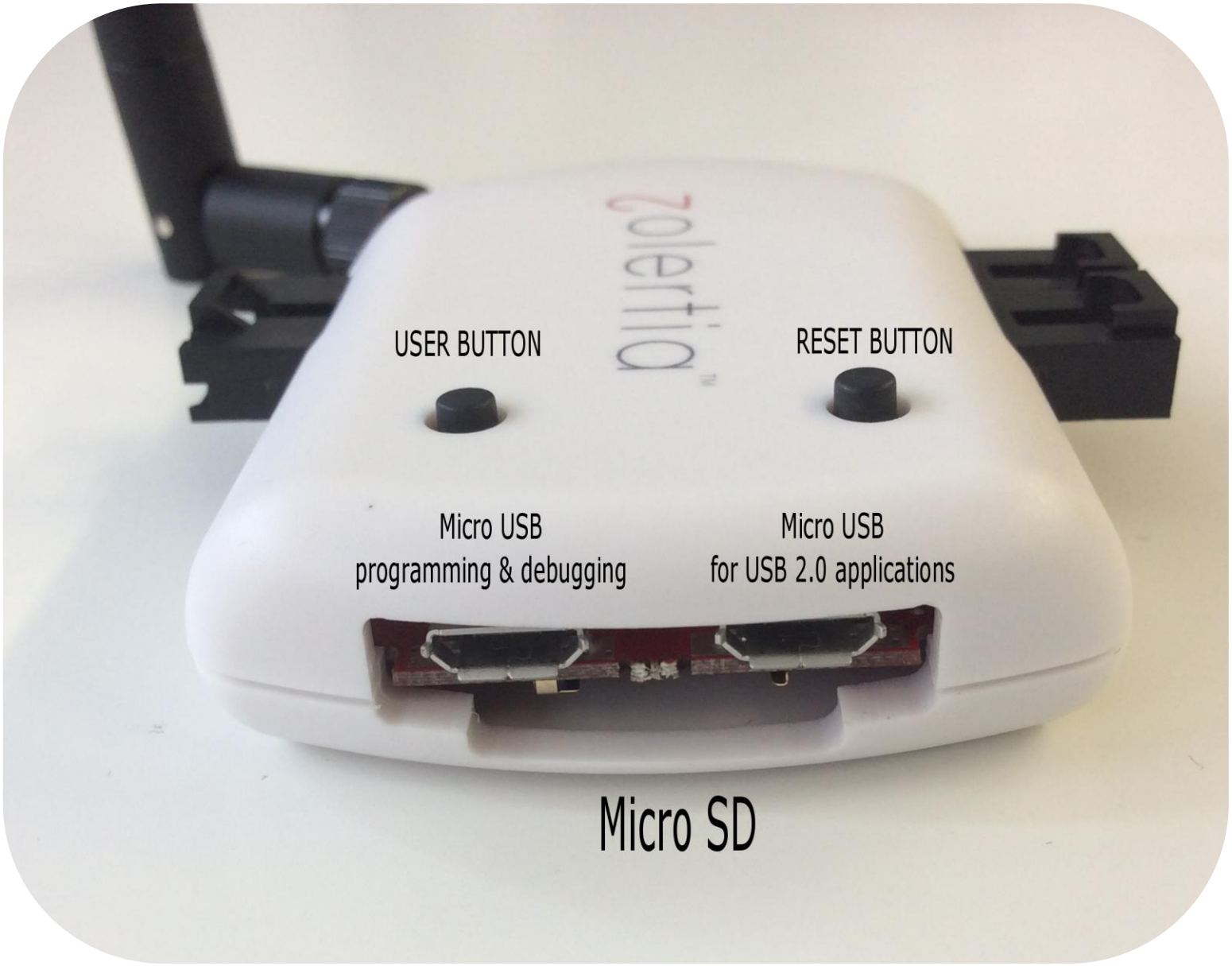
- Architectures: 8-bit, 16-bit, 32-bit
- Open Source (source code openly available)
- IPv4/IPv6/Rime networking
- Devices with < 8KB RAM
- Typical applications < 50KB Flash
- Vendor and platform independent
- C language
- Developed and contributed by Universities, Research centers and industry contributors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

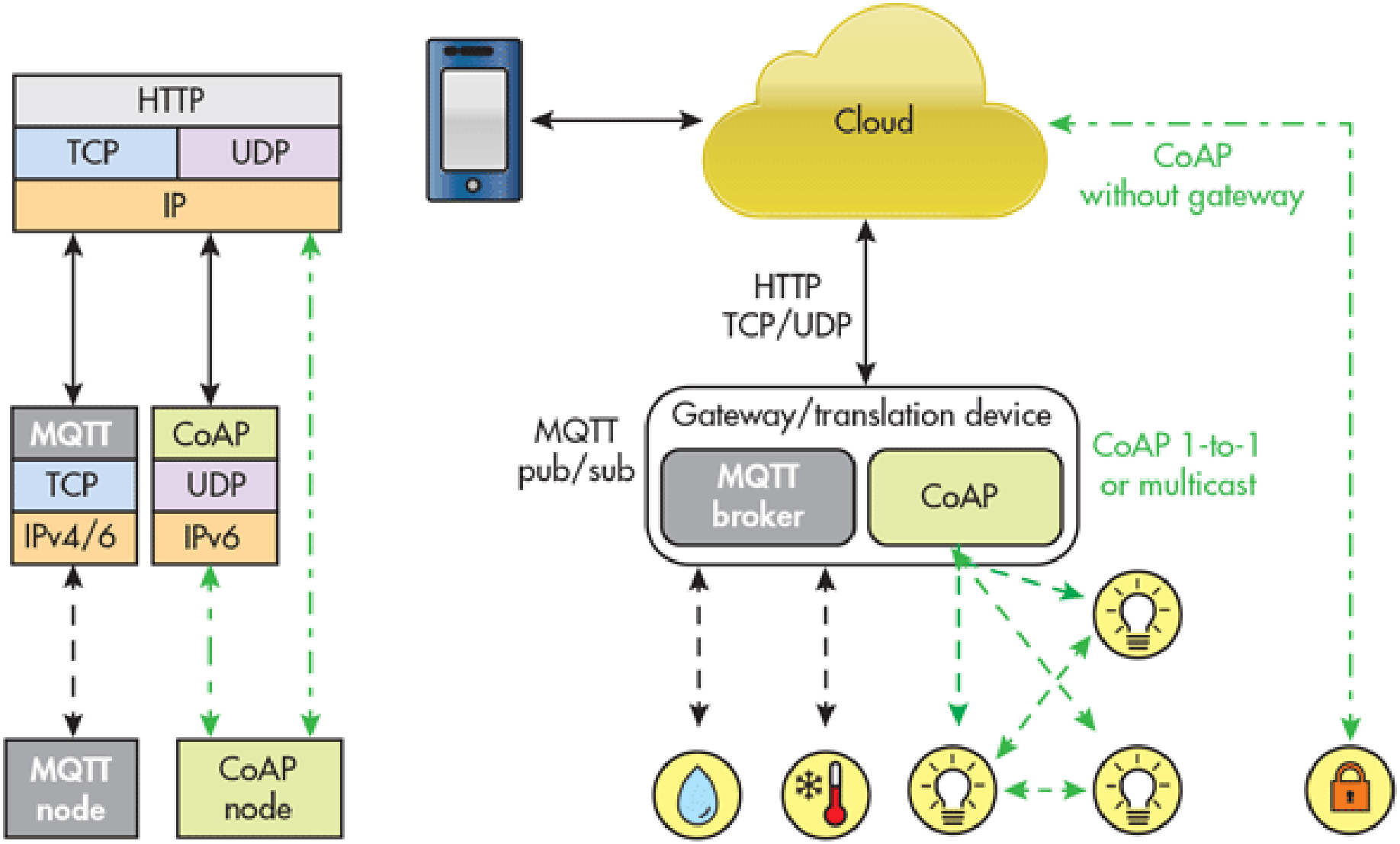
Zolertia RE-Mote

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







http://electronicdesign.com/iot/mqtt-and-coap-underlying-protocols-iot

03-coap

examples/zolertia/tutorial/03-coap

CoAP

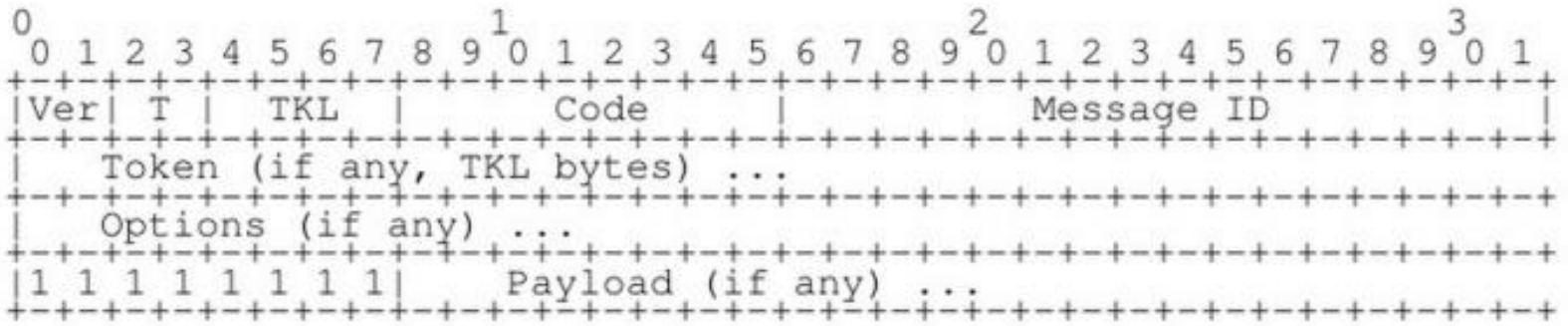
RFC 7252 Constrained Application Protocol

"The Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained networks in the **Internet of Things.**
The protocol is designed for machine-to-machine (M2M) applications such as smart energy and building automation."

- UDP-*reliable* (confirmable), SMS supported
- CoRE *Link-format* (GET */.well known/core*)
- Client/Server
- IANA Registered (error codes, content format)
- Resource Discovery and asynchronous subscription
- Four-bytes compact header
- Multicast and one-to-one supported
- HTTP verbs GET, PUT, POST, DELETE
- HTTP-like header (*Options*)
- URI (Uniform Resource Identifier)

http://coap.technology/
https://tools.ietf.org/html/rfc7252

examples/zolertia/tutorial/03-coap



Ver - Version (1)

T - Message Type (Confirmable, Non-Confirmable, Acknowledgement, Reset)

TKL- Token Length, if any, the number of Token bytes after this header

Code - Request Method (1-10) or Response Code (40-255)

Message ID - 16-bit identifier for matching responses

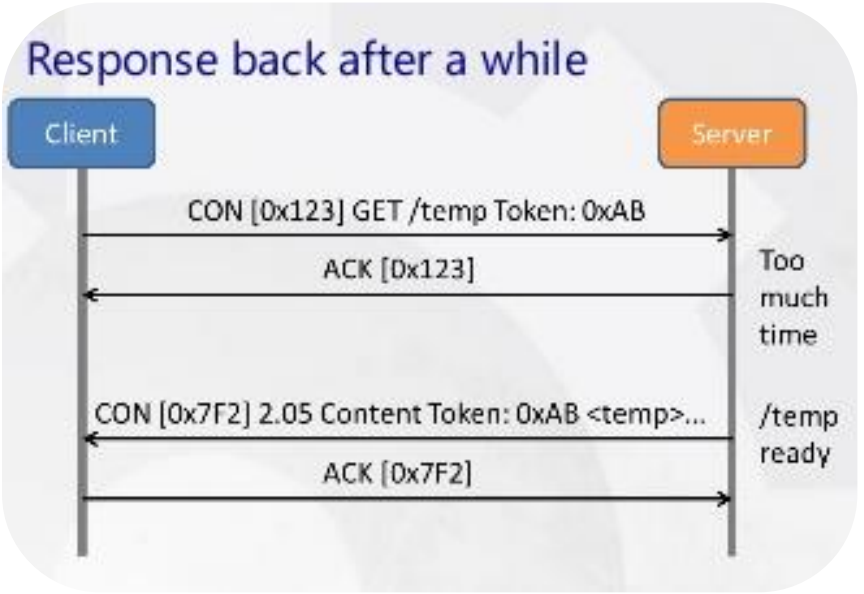
Token - Optional response matching token

http://www.slideshare.net/zdshelby/coap-tutorial

examples/zolertia/tutorial/03-coap

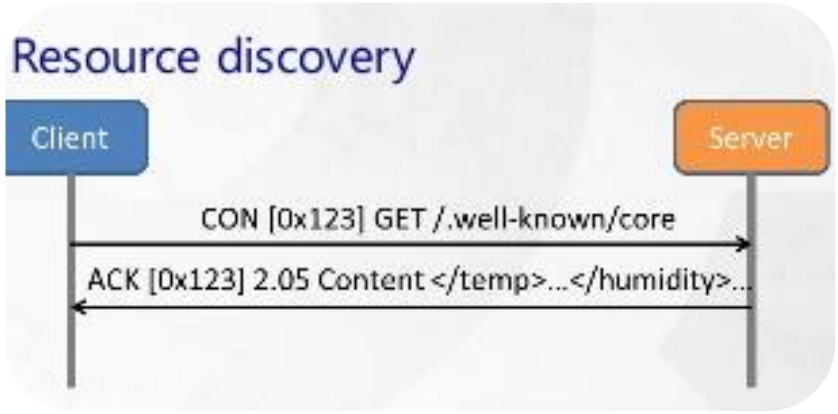
CoAP URI

coap://[aaaa::c30c:0:0:1234]:5683/actuators/leds?color=b



http://www.slideshare.net/paolopat/mqtt-iot-protocols-comparison

examples/zolertia/tutorial/03-coap



http://www.slideshare.net/paolopat/mqtt-iot-protocols-comparison

examples/zolertia/tutorial/03-coap

A **normal resource** is defined by a static Uri-Path that is associated with a resource handler function. This is the basis for all other resource types.

```
#define RESOURCE(name, attributes, get_handler, post_handler, put_handler,
 delete_handler) \
  resource_t name = { NULL, NULL, NO_FLAGS, attributes, get_handler, post_handler,
 put_handler, delete_handler, { NULL } }
```

A **parent resource** manages several sub-resources by evaluating the Uri-Path, which may be longer than the parent resource.

```
#define PARENT_RESOURCE(name, attributes, get_handler, post_handler, put_handler,
 delete_handler) \
  resource_t name = { NULL, NULL, HAS_SUB_RESOURCES, attributes, get_handler,
 post_handler, put_handler, delete_handler, { NULL } }
```

apps/er-coap
apps/rest-engine

If the server is not able to respond immediately to a `CON` request, it simply responds with an empty ACK message so that the client can stop re-transmitting the request. After a while, when the server is ready with the response, it sends the response as a `CON` message. The following macro allows to create a CoAP resource with **separate response**:

```
#define SEPARATE_RESOURCE(name, attributes, get_handler, post_handler, put_handler,
 delete_handler, resume_handler) \
  resource_t name = { NULL, NULL, IS_SEPARATE, attributes, get_handler,
 post_handler, put_handler, delete_handler, { .resume = resume_handler } }
```

An **event resource** is similar to a periodic resource, but the second handler is called by a non periodic event such as the pressing of a button.

```
#define EVENT_RESOURCE(name, attributes, get_handler, post_handler, put_handler,
 delete_handler, event_handler) \
  resource_t name = { NULL, NULL, IS_OBSERVABLE, attributes, get_handler,
post_handler, put_handler, delete_handler, { .trigger = event_handler } }
```

apps/er-coap
apps/rest-engine

If we need to declare a **periodic resource**, for example to poll a sensor and publish a changed value to subscribed clients, then we should use:

```
#define PERIODIC_RESOURCE(name, attributes, get_handler, post_handler, put_handler,
 delete_handler, period, periodic_handler) \
  periodic_resource_t periodic_##name; \
  resource_t name = { NULL, NULL, IS_OBSERVABLE | IS_PERIODIC, attributes,
 get_handler, post_handler, put_handler, delete_handler, { .periodic =
 &periodic_##name } }; \
  periodic_resource_t periodic_##name = { NULL, &name, period, { { 0 } },
 periodic_handler };
```

Notice that the `PERIODIC_RESOURCE` and `EVENT_RESOURCE` can be observable, meaning a client can be notified of any change in a given resource.

apps/er-coap
apps/rest-engine

```
/* GET method */
RESOURCE(res_adxl345,
         "title=\"Accelerometer 3-axis\";rt=\"adxl345\"",
         res_get_handler,
         NULL,
         NULL,
         NULL);
```

URI Query

Function to invoke whenever there's a GET request

```
static void
res_get_handler(void *request, void *response, uint8_t *buffer,
                uint16_t preferred_size, int32_t *offset)
{
  uint8_t x_axis = adxl345.value(X_AXIS);
  uint8_t y_axis= adxl345.value(Y_AXIS);
  uint8_t z_axis= adxl345.value(Z_AXIS);
  unsigned int accept = -1;

  REST.get_header_accept(request, &accept);

  if(accept == REST.type.APPLICATION_JSON) {
    REST.set_header_content_type(response, REST.type.APPLICATION_JSON);
    snprintf((char *)buffer, REST_MAX_CHUNK_SIZE,
    "{'adxl345':{'X':%d, 'Y':%d, 'Z':%d}}", x_axis, y_axis, z_axis);
    REST.set_response_payload(response, buffer, strlen((char *)buffer));
```

The CoAP Accept option can be used to indicate which Content-Format is acceptable to the client.

examples/zolertia/tutorial/03-coap/resources/res-adxl345.c

Importing the Resource

```
/*
 * Resources to be activated need to be imported through the extern keyword.
 * The build system automatically compiles the resources in the corresponding
 * sub-directory.
 */
extern resource_t
  res_hello,
  res_leds,
  res_toggle,
  res_adxl345,
  res_event,
  res_separate;
```

Resource activation

```
/*
 * Bind the resources to their Uri-Path.
 * WARNING: Activating twice only means alternate path, not two instances!
 * All static variables are the same for each URI path.
 */
rest_activate_resource(&res_hello, "test/hello");
rest_activate_resource(&res_leds, "actuators/leds");
rest_activate_resource(&res_toggle, "actuators/toggle");
rest_activate_resource(&res_adxl345, "sensors/adxl345");
rest_activate_resource(&res_event, "sensors/button");
rest_activate_resource(&res_separate, "test/separate");
```

examples/zolertia/tutorial/03-coap/resources/er-example-server.c

```
all: er-example-server

CONTIKI = ../../../..

CFLAGS += -DPROJECT_CONF_H=\"project-conf.h\"

# Automatically build RESTful resources
REST_RESOURCES_DIR = ./resources

REST_RESOURCES_FILES = $(notdir $(shell find $(REST_RESOURCES_DIR) -name '*.c'))
PROJECTDIRS += $(REST_RESOURCES_DIR)
PROJECT_SOURCEFILES += $(REST_RESOURCES_FILES)

# linker optimizations
SMALL = 1

# REST Engine shall use Erbium CoAP implementation
APPS += er-coap
APPS += rest-engine

CONTIKI_WITH_IPV6 = 1
include $(CONTIKI)/Makefile.include
```

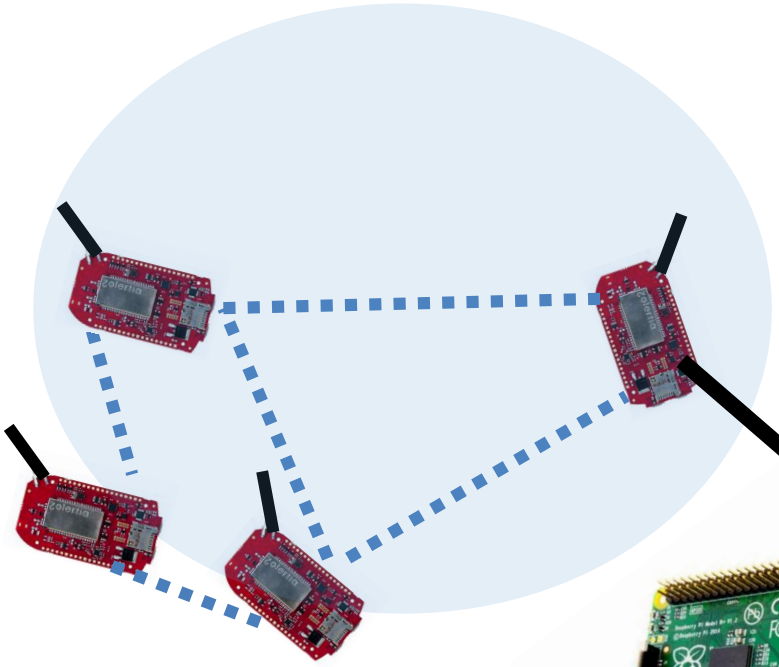
Search resources to include in the “resources” directory

REST engine and CoAP libraries

examples/zolertia/tutorial/03-coap/resources/Makefile

Only on Firefox: install the following plug-in:

https://addons.mozilla.org/en-US/firefox/addon/copper-270430/
http://people.inf.ethz.ch/mkovatsc/copper.php



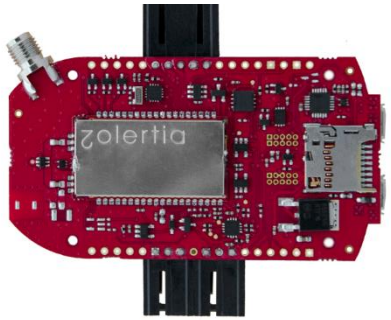
Border Router
IPv6/6LoWPAN

er-example-server
CoAP server in Contiki OS

Copper CoAP user-agent
On Firefox browse the CoAP Server, discover and learn its resources

examples/zolertia/tutorial/03-coap/er-example-server.c

CoAP Server



Erbium for Contiki

- REST Engine
- For embedded devices
- Enables thin server architecture

Core temperature

Voltage level

RGB LED

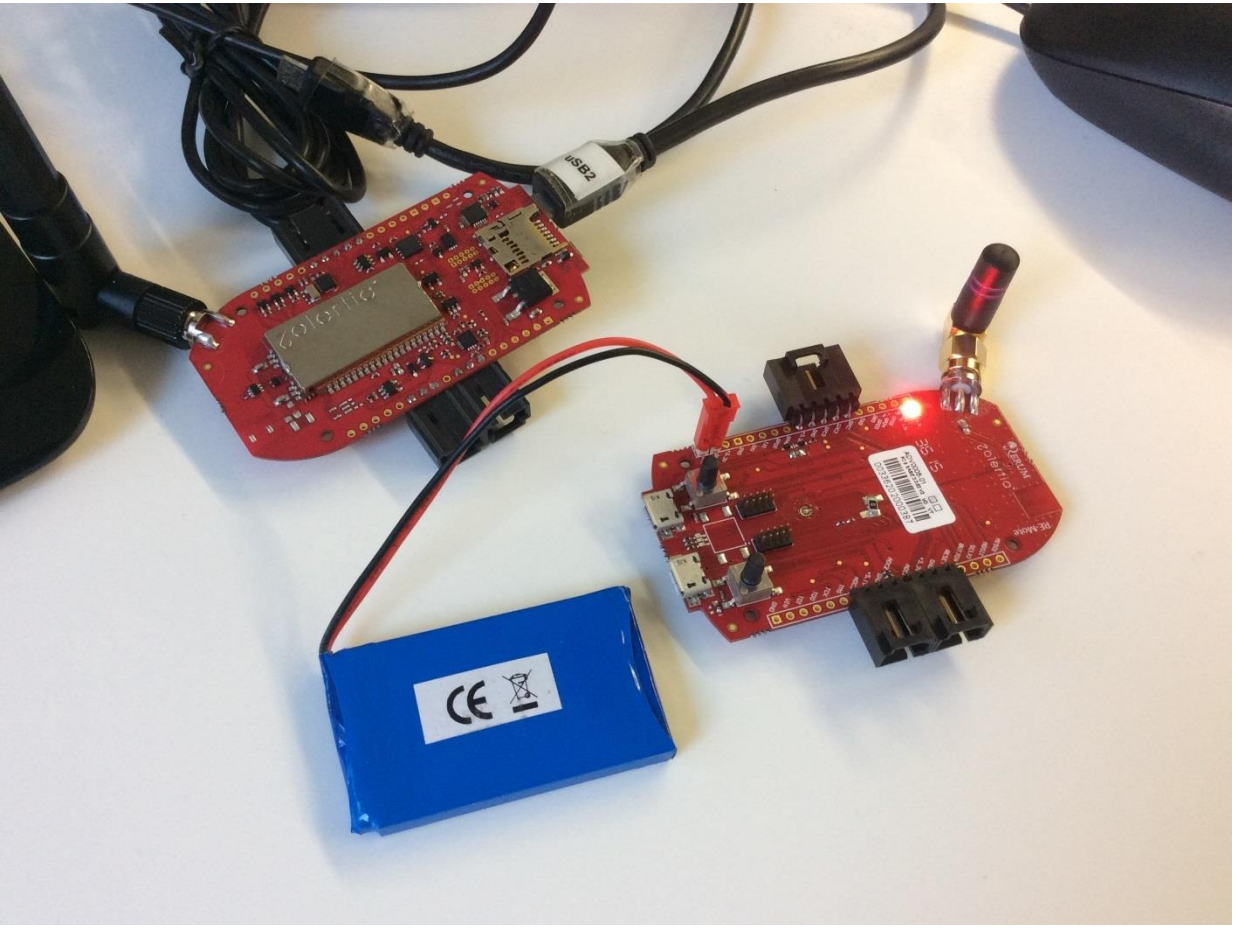
Button events

Copper for Firefox

- CoAP user-agent
- For user interaction
- Integration and testing tool

CoAP Client

examples/zolertia/tutorial/03-coap/er-example-server.c



Connect a RE-Mote and program the example:

```
make er-example-server.upload
```

examples/zolertia/tutorial/03-coap/er-example-server.c

On another terminal run the Border Router

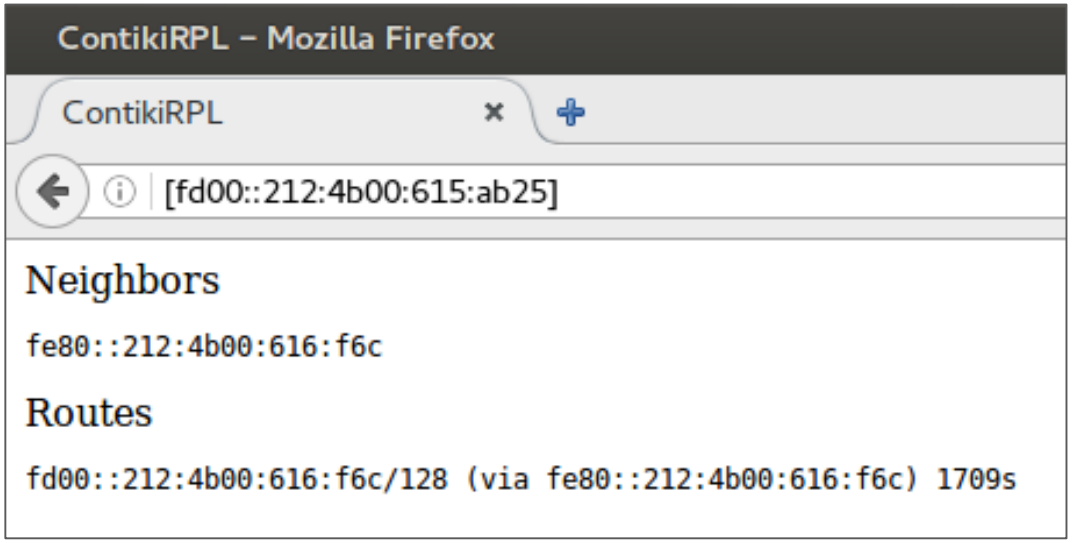
```
user@iot-workshop:~/contiki/examples/zolertia/tutorial/02-ipv6/02-border-router$ sudo ../
../../../../tools/./tunslip6 -s /dev/ttyUSB1 fd00::1/64
[sudo] password for user:
********SLIP started on ``/dev/ttyUSB1''
opened tun device ``/dev/tun0''
ifconfig tun0 inet `hostname` mtu 1500 up
ifconfig tun0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

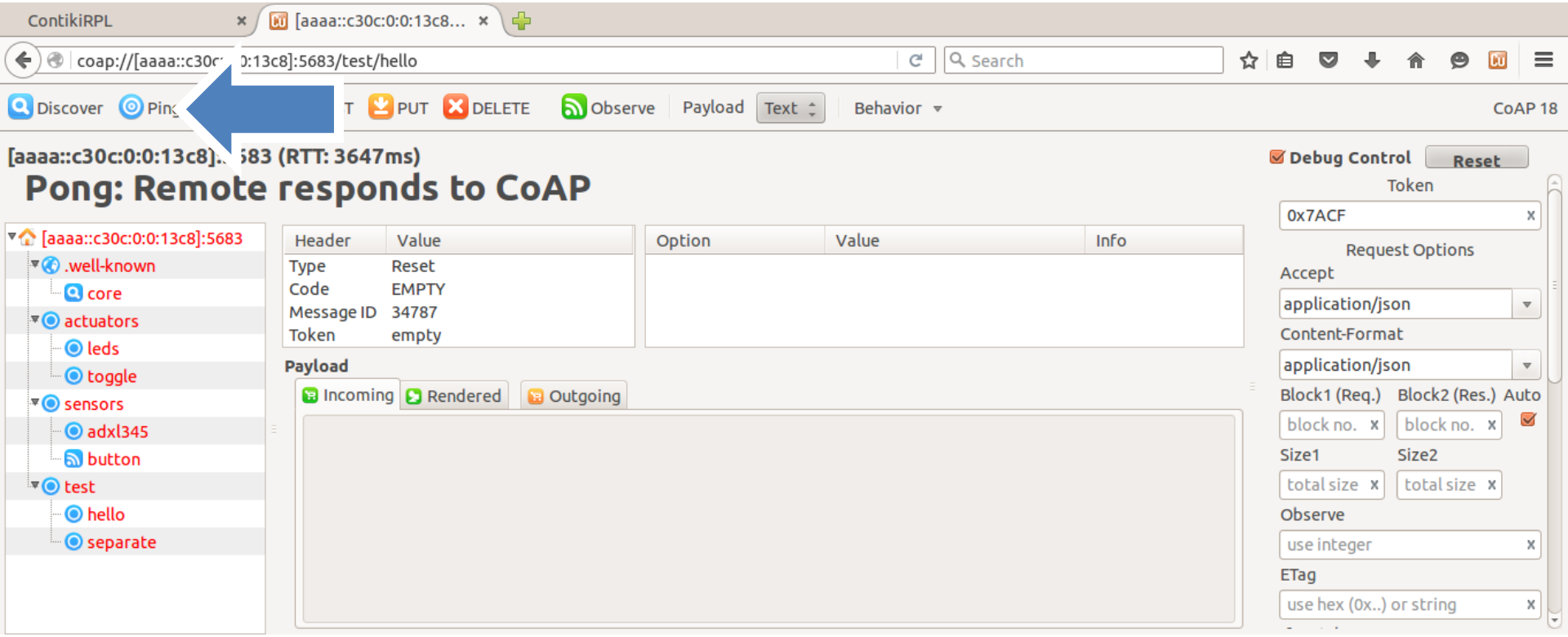
Verify the UDP client is in our network and responsive




```
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
```

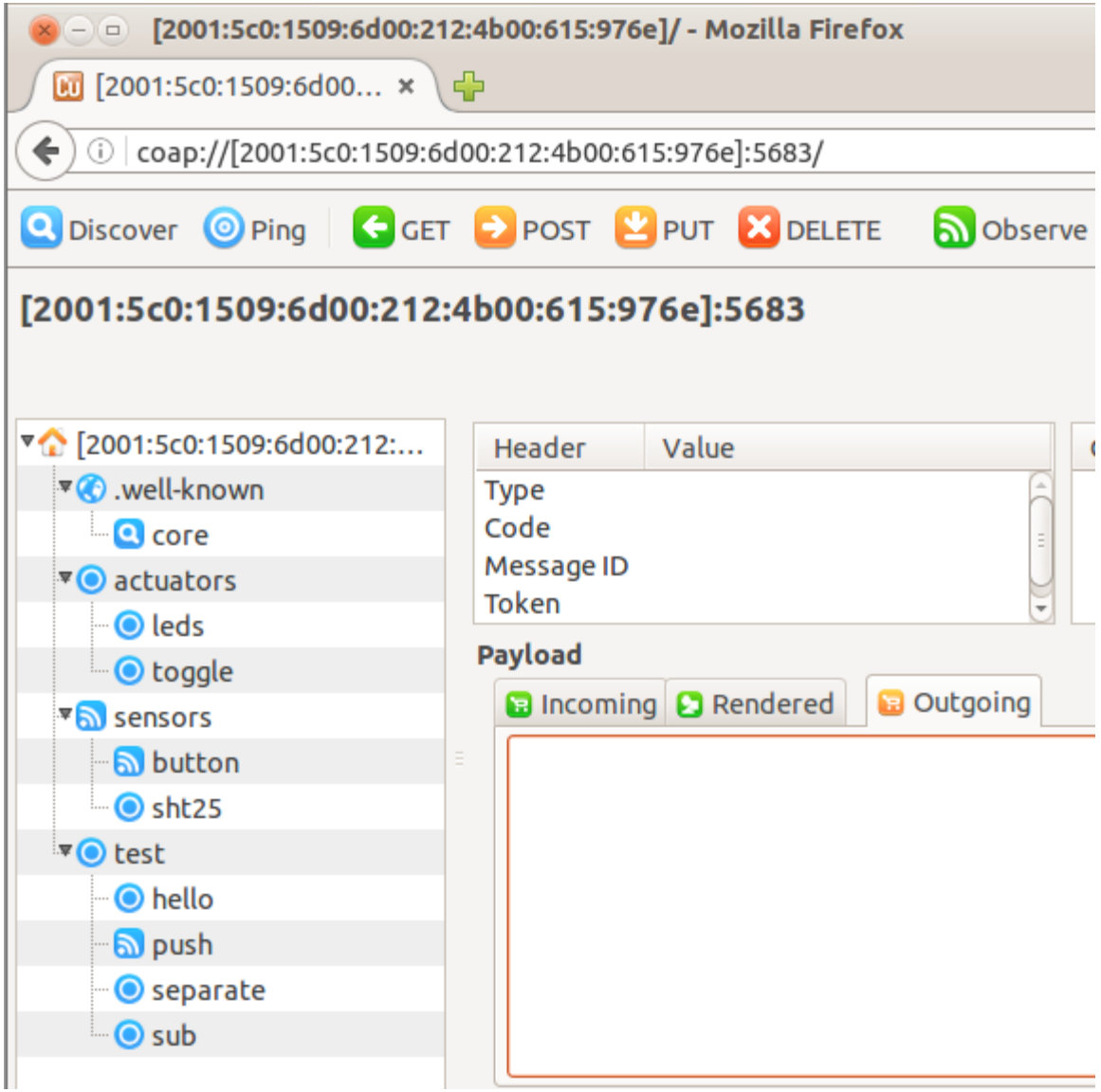
examples/zolertia/tutorial/02-ipv6/udp-client-and-server

Pong! – check the CoAP server is online

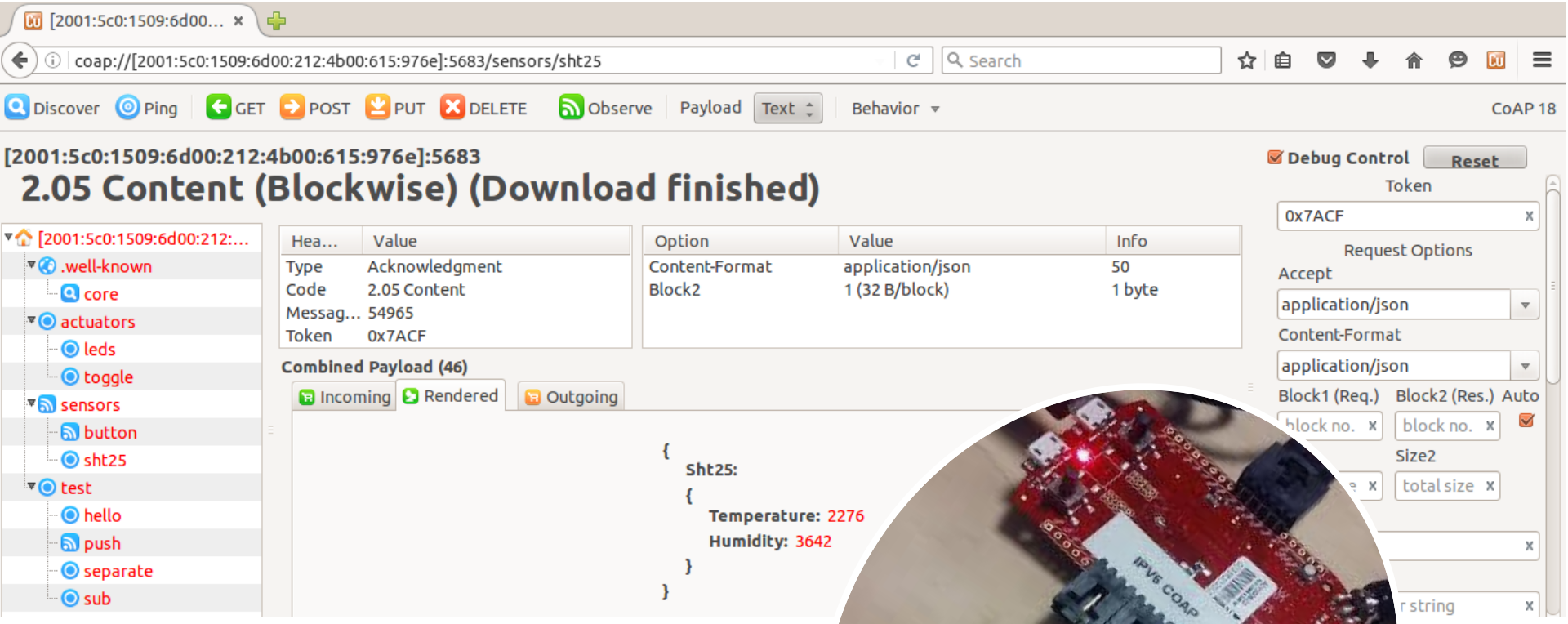


examples/zolertia/tutorial/03-coap/resources/er-example-server.c

Discover the Resources of the devices, and how to use them



examples/zolertia/tutorial/03-coap/er-example-server.c



Retrieve data on different formats

examples/zolertia/tutorial/03-coap/er-example-server.c

Control devices and actuators

examples/zolertia/tutorial/03-coap/er-example-server.c

[2001:5c0:1509:6d00:212:4b00:615:976e]/sensors/button - Mozilla Firefox

[2001:5c0:1509:6d00...

coap://[2001:5c0:1509:6d00:212:4b00:615:976e]:5683/sensors/button

Search

Discover Ping GET POST PUT DELETE Cancel Payload Text Behavior

[2001:5c0:1509:6d00:212:4b00:615:976e]:5683

2.05 Content (Observing)

- [2001:5c0:1509:6d00:212:...
  - .well-known
    - core
  - actuators
    - leds
    - toggle
  - sensors
    - button
    - sht25

Hea...	Value
Type	Non-Confirmable
Code	2.05 Content
Messag...	53459
Token	0x7ACF

Option	Value	Info
Observe	11	1 byte
Content-Format	text/plain	0

Payload (8)

Incoming Rendered Outgoing

```
EVENT 10
```

Observe a variable and receive notifications if any change

examples/zolertia/tutorial/03-coap/er-example-server.c

	Destination	Protocol	Length	Info
.508:f301:c30c::13d8	2001:5c0:1508:f300:143f:bd0e:4c65:b1a6	CoAP	89	ACK, MID:18376, 2.05 Content, Block #8
.508:f300:143f:bd0e:4c65:…	2001:5c0:1508:f301:c30c::13d8	CoAP	71	CON, MID:18377, GET, End of Block #9, /.well-known/core
.508:f301:c30c::13d8	2001:5c0:1508:f300:143f:bd0e:4c65:b1a6	CoAP	89	ACK, MID:18377, 2.05 Content, Block #9
.508:f300:143f:bd0e:4c65:…	2001:5c0:1508:f301:c30c::13d8	CoAP	71	CON, MID:18378, GET, End of Block #10, /.well-known/core
.508:f301:c30c::13d8	2001:5c0:1508:f300:143f:bd0e:4c65:b1a6	CoAP	86	ACK, MID:18378, 2.05 Content, End of Block #10
.508:f300:143f:bd0e:4c65:…	2001:5c0:1508:f301:c30c::13d8	CoAP	52	CON, MID:18379, Empty Message
.508:f301:c30c::13d8	2001:5c0:1508:f300:143f:bd0e:4c65:b1a6	CoAP	52	RST, MID:18379, Empty Message
.508:f300:143f:bd0e:4c65:…	2001:5c0:1508:f301:c30c::13d8	CoAP	87	CON, MID:65201, POST, /actuators/leds?color=g (application/json)
.508:f301:c30c::13d8	2001:5c0:1508:f300:143f:bd0e:4c65:b1a6	CoAP	52	ACK, MID:65201, 2.05 Content
.508:f300:143f:bd0e:4c65:…	2001:5c0:1508:f301:c30c::13d8	CoAP	80	CON, MID:18975, POST, /actuators/leds (application/json)
.508:f301:c30c::13d8	2001:5c0:1508:f300:143f:bd0e:4c65:b1a6	CoAP	52	ACK, MID:18975, 4.00 Bad Request
.508:f300:143f:bd0e:4c65:…	2001:5c0:1508:f301:c30c::13d8	CoAP	80	CON, MID:18976, PUT, /actuators/leds (application/json)
.508:f301:c30c::13d8	2001:5c0:1508:f300:143f:bd0e:4c65:b1a6	CoAP	52	ACK, MID:18976, 4.00 Bad Request
.508:f300:143f:bd0e:4c65:…	2001:5c0:1508:f301:c30c::13d8	CoAP	88	CON, MID:51297, POST, /actuators/leds?color=g (application/json)
.508:f301:c30c::13d8	2001:5c0:1508:f300:143f:bd0e:4c65:b1a6	CoAP	52	ACK, MID:51297, 2.05 Content
.508:f300:143f:bd0e:4c65:…	2001:5c0:1508:f301:c30c::13d8	CoAP	87	CON, MID:51298, POST, /actuators/leds?color=g (application/json)
.508:f301:c30c::13d8	2001:5c0:1508:f300:143f:bd0e:4c65:b1a6	CoAP	52	ACK, MID:51298, 2.05 Content
.508:f300:143f:bd0e:4c65:…	2001:5c0:1508:f301:c30c::13d8	CoAP	81	CON, MID:38081, POST, /actuators/toggle (application/json)
.508:f301:c30c::13d8	2001:5c0:1508:f300:143f:bd0e:4c65:b1a6	CoAP	52	ACK, MID:38081, 2.05 Content
.508:f300:143f:bd0e:4c65:…	2001:5c0:1508:f301:c30c::13d8	CoAP	81	CON, MID:38082, POST, /actuators/toggle (application/json)
.508:f301:c30c::13d8	2001:5c0:1508:f300:143f:bd0e:4c65:b1a6	CoAP	52	ACK, MID:38082, 2.05 Content
.508:f300:143f:bd0e:4c65:…	2001:5c0:1508:f301:c30c::13d8	CoAP	74	CON, MID:51474, GET, End of Block #0, /sensors/adxl345
.508:f301:c30c::13d8	2001:5c0:1508:f300:143f:bd0e:4c65:b1a6	CoAP	89	ACK, MID:51474, 2.05 Content, Block #0 (application/json)
.508:f300:143f:bd0e:4c65:…	2001:5c0:1508:f301:c30c::13d8	CoAP	74	CON, MID:51475, GET, End of Block #1, /sensors/adxl345
.508:f301:c30c::13d8	2001:5c0:1508:f300:143f:bd0e:4c65:b1a6	CoAP	62	ACK, MID:51475, 2.05 Content, End of Block #1 (application/json)
.508:f300:143f:bd0e:4c65:…	2001:5c0:1508:f301:c30c::13d8	CoAP	72	CON, MID:51476, GET, End of Block #0, /sensors/adxl345
.508:f301:c30c::13d8	2001:5c0:1508:f300:143f:bd0e:4c65:b1a6	CoAP	67	ACK, MID:51476, 2.05 Content, End of Block #0 (text/plain)
.508:f300:143f:bd0e:4c65:…	2001:5c0:1508:f301:c30c::13d8	CoAP	67	CON, MID:55727, GET, End of Block #0, /test/hello
.508:f301:c30c::13d8	2001:5c0:1508:f300:143f:bd0e:4c65:b1a6	CoAP	70	ACK, MID:55727, 2.05 Content, End of Block #0 (text/plain)

```
on wire (552 bits), 69 bytes captured (552 bits)

Version 6, Src: 2001:5c0:1508:f300:143f:bd0e:4c65:b1a6 (2001:5c0:1508:f300:143f:bd0e:4c65:b1a6), Dst: 2001:5c0:1508:f301:c30c::13d8 (2001:5c0:1508:f301:c30
ocol, Src Port: 63264 (63264), Dst Port: 5683 (5683)
ation Protocol, Confirmable, GET, MID:18368
```

examples/zolertia/tutorial/03-coap/wireshark-coap-server-example.pcap

04-mqtt

examples/zolertia/tutorial/04-mqtt

- On top of TCP/IP
- Publish/Subscribe messaging pattern
- Message Broker distributes topics to clients
- Topics are UTF-8 string-based with hierarchical structure
- No direct connection between clients
- Quality of Service
- Retain-Flag: new subscribed clients will received last value
- Last Will: notify other clients when disconnected ungracefully
- KeepAlive: ping request messages to the bróker
- Clients have to know beforehand the structure of the data published to a topic
- MQTT is data-agnostic

http://mqtt.org

examples/zolertia/tutorial/04-mqtt

CONNECT

Waits for a connection to be established with the server

DISCONNECT

Waits for the MQTT client to finish any pending task and closes the TCP session

SUBSCRIBE

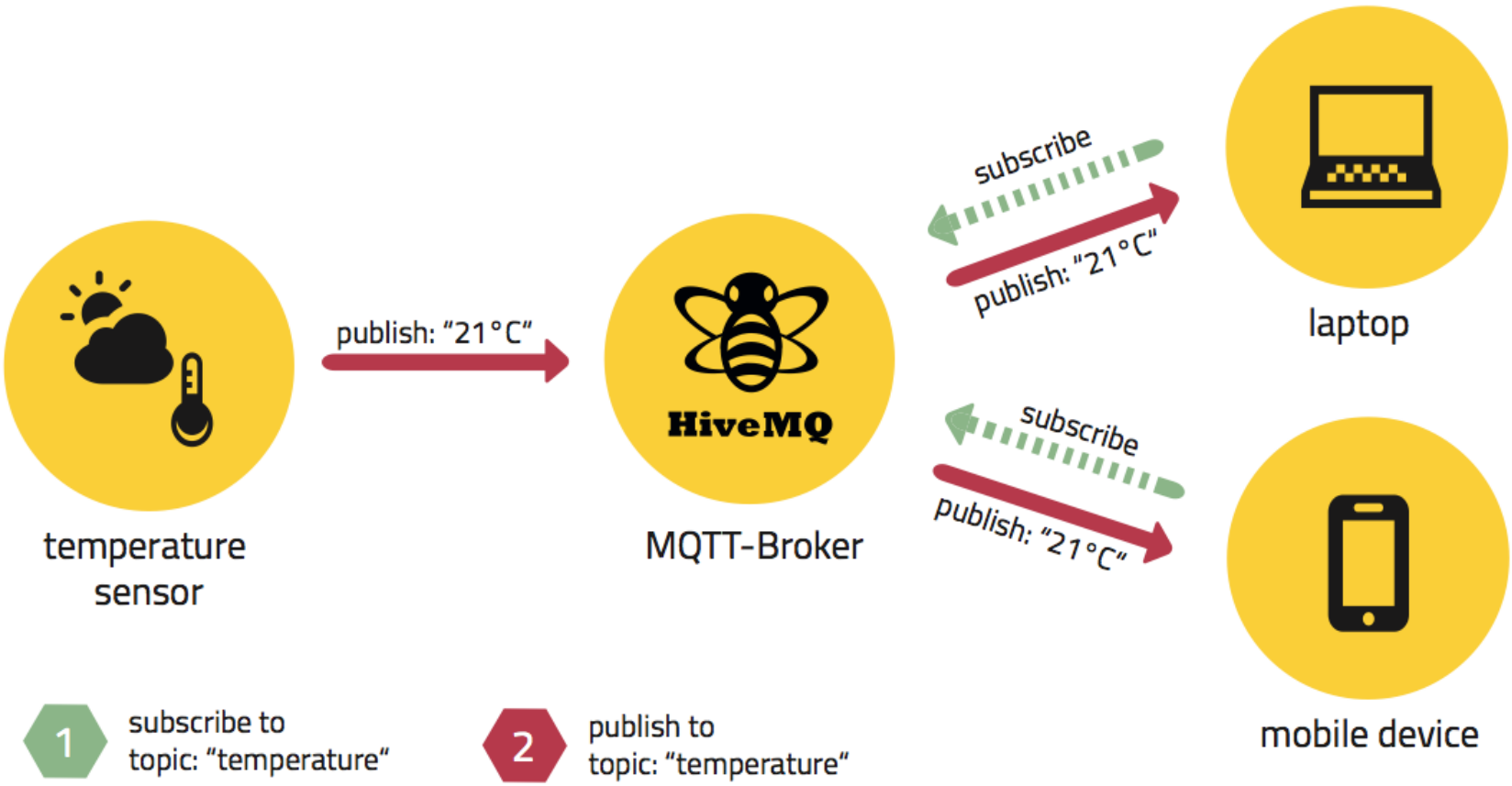
Request the server to subscribe the client to one or more topics

UNSUBSCRIBE

Request the server to subscribe the client to one or more topics

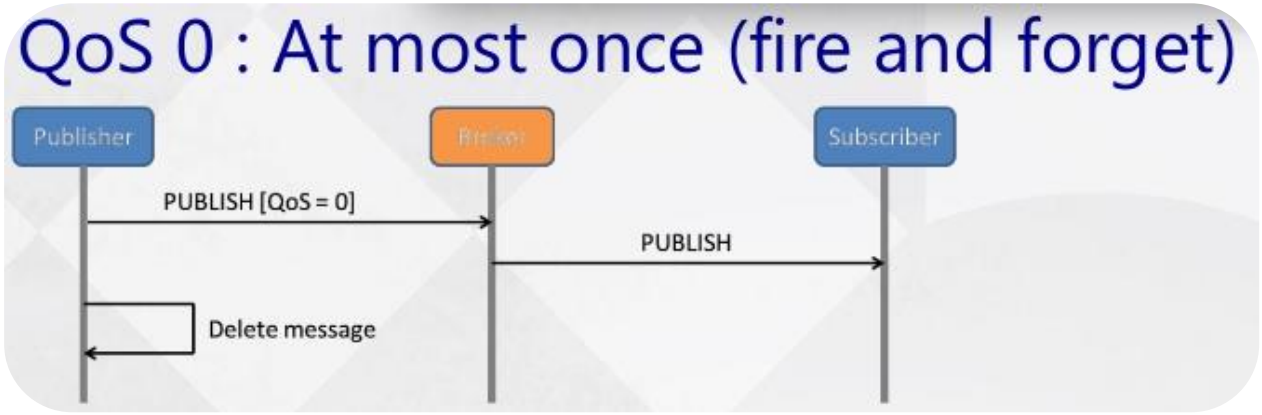
PUBLISH

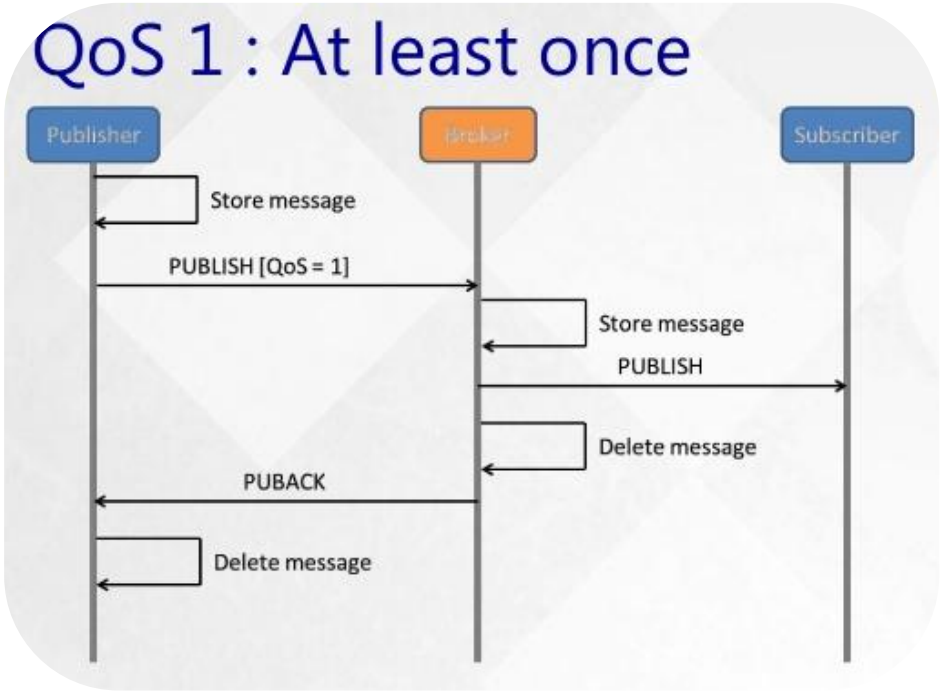
Updates a topic with data

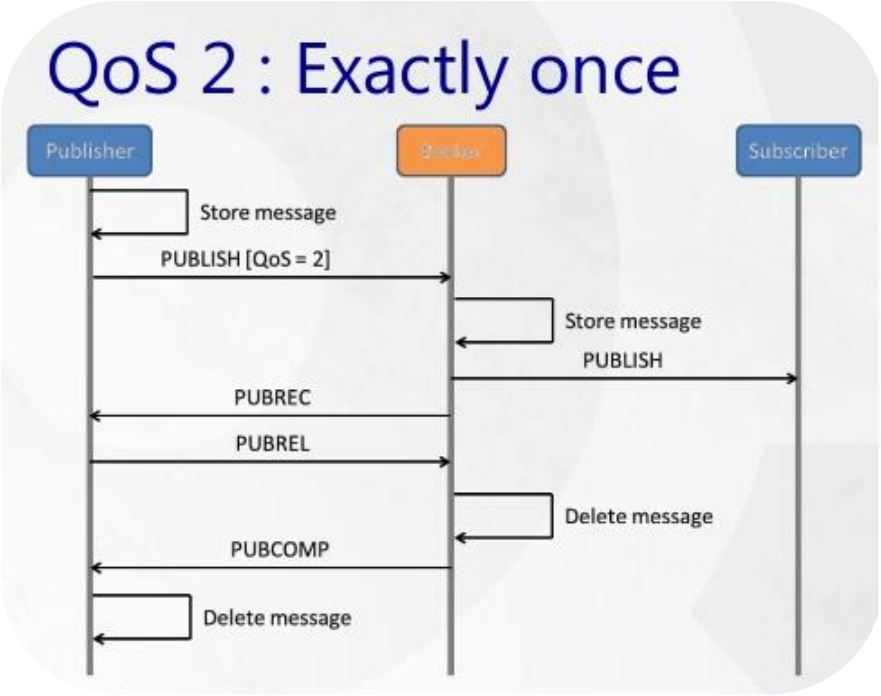


http://www.hivemq.com/blog/how-to-get-started-with-mqtt

examples/zolertia/tutorial/04-mqtt

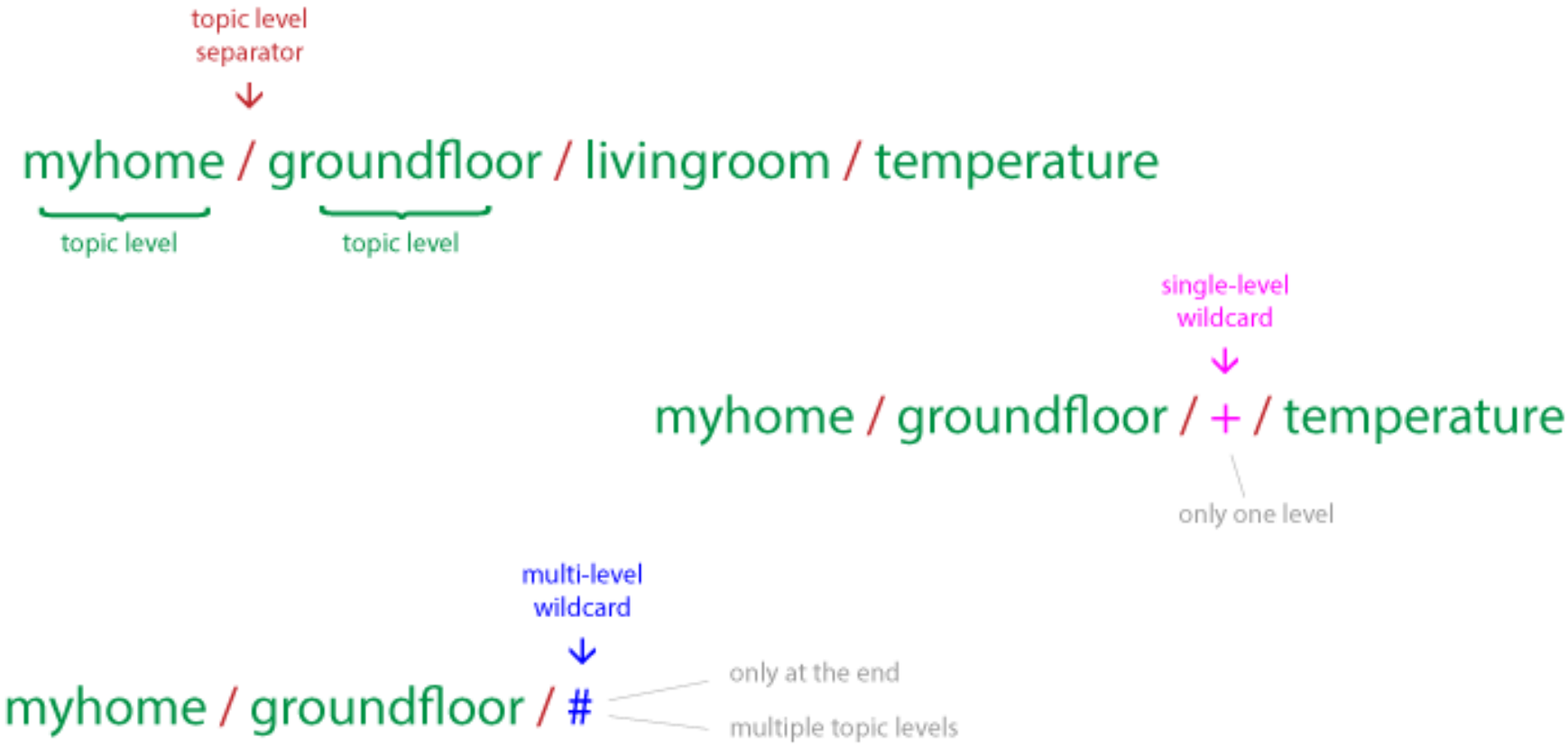






http://www.slideshare.net/paolopat/mqtt-iot-protocols-comparison

examples/zolertia/tutorial/04-mqtt



Topics starting with $ are special
These are reserved for the broker statistics

$SYS/broker/clients/connected
$SYS/broker/clients/disconnected
$SYS/broker/clients/total
$SYS/broker/messages/sent
$SYS/broker/uptime

http://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices

examples/zolertia/tutorial/04-mqtt

Required to be included in the process using MQTT

```
/*---------------------------------------------------------------------------*/
/**
 * \brief           MQTT event callback function
 * \param m         A pointer to a MQTT connection
 * \param event     The event number
 * \param data      A user-defined pointer
 *
 * The MQTT socket event callback function gets called whenever there is an
 * event on a MQTT connection, such as the connection getting connected
 * or closed.
 */
typedef void (*mqtt_event_callback_t)(struct mqtt_connection *m,
                                      mqtt_event_t event,
                                      void *data);
```

To start the MQTT client this function should be called first
The max_segment_size is the TCP chunk of data to be sent (default 32 bytes)
The client_id is a string identifying the client

```
/**
 * \brief Initializes the MQTT engine.
 * \param conn A pointer to the MQTT connection.
 * \param app_process A pointer to the application process handling the MQTT
 *        connection.
 * \param client_id A pointer to the MQTT client ID.
 * \param event_callback Callback function responsible for handling the
 *        callback from MQTT engine.
 * \param max_segment_size The TCP segment size to use for this MQTT/TCP
 *        connection.
 * \return MQTT_STATUS_OK or MQTT_STATUS_INVALID_ARGS_ERROR
 *
 * This function initializes the MQTT engine and shall be called before any
 * other MQTT function.
 */
mqtt_status_t mqtt_register(struct mqtt_connection *conn,
                            struct process *app_process,
                            char *client_id,
                            mqtt_event_callback_t event_callback,
                            uint16_t max_segment_size);
```

The keep_alive value is used by a timer waiting a PINGRES from the broken, if expired and no response is obtained, it triggers a disconnection

```
/**
 * \brief Connects to a MQTT broker.
 * \param conn A pointer to the MQTT connection.
 * \param host IP address of the broker to connect to.
 * \param port Port of the broker to connect to, default is MQTT port is 1883.
 * \param keep_alive Keep alive timer in seconds. Used by broker to handle
 *        client disc. Defines the maximum time interval between two messages
 *        from the client. Shall be min 1.5 x report interval.
 * \return MQTT_STATUS_OK or an error status
 *
 * This function connects to a MQTT broker.
 */
mqtt_status_t mqtt_connect(struct mqtt_connection *conn,
                           char *host,
                           uint16_t port,
                           uint16_t keep_alive);
```

```
/**
 * \brief Disconnects from a MQTT broker.
 * \param conn A pointer to the MQTT connection.
 *
 * This function disconnects from a MQTT broker.
 */
void mqtt_disconnect(struct mqtt_connection *conn);
```

```
/**
 * \brief Subscribes to a MQTT topic.
 * \param conn A pointer to the MQTT connection.
 * \param mid A pointer to message ID.
 * \param topic A pointer to the topic to subscribe to.
 * \param qos_level Quality Of Service level to use. Currently supports 0, 1.
 * \return MQTT_STATUS_OK or some error status
 *
 * This function subscribes to a topic on a MQTT broker.
 */
mqtt_status_t mqtt_subscribe(struct mqtt_connection *conn,
                             uint16_t *mid,
                             char *topic,
                             mqtt_qos_level_t qos_level);
```

Message ID (mid) is zero for QoS=0

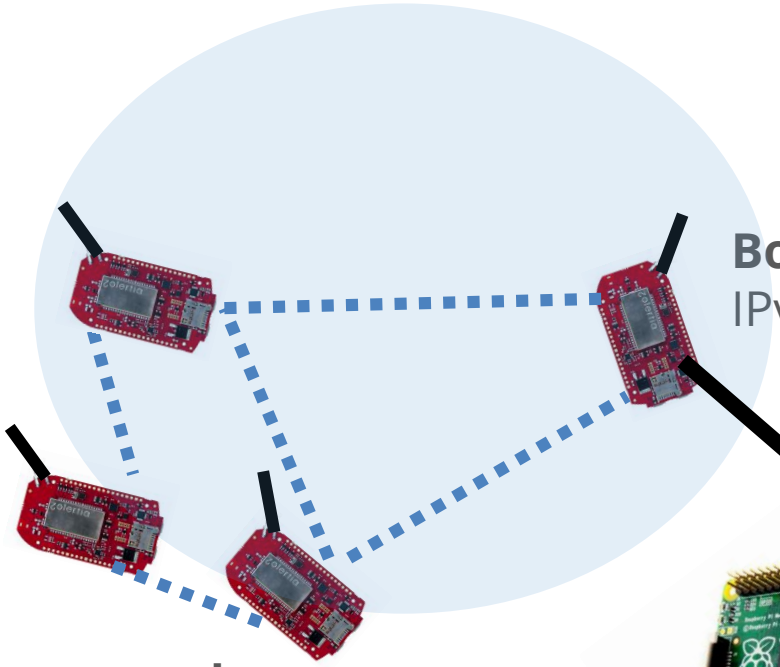
```
/**
 * \brief Unsubscribes from a MQTT topic.
 * \param conn A pointer to the MQTT connection.
 * \param mid A pointer to message ID.
 * \param topic A pointer to the topic to unsubscribe from.
 * \return MQTT_STATUS_OK or some error status
 *
 * This function unsubscribes from a topic on a MQTT broker.
 */
mqtt_status_t mqtt_unsubscribe(struct mqtt_connection *conn,
                               uint16_t *mid,
                               char *topic);
```

```
/**
 * \brief Publish to a MQTT topic.
 * \param conn A pointer to the MQTT connection.
 * \param mid A pointer to message ID.
 * \param topic A pointer to the topic to subscribe to.
 * \param payload A pointer to the topic payload.
 * \param payload_size Payload size.
 * \param qos_level Quality Of Service level to use. Currently supports 0, 1.
 * \param retain If the RETAIN flag is set to 1, in a PUBLISH Packet sent by a
 *        Client to a Server, the Server MUST store the Application Message
 *        and its QoS, so that it can be delivered to future subscribers whose
 *        subscriptions match its topic name
 * \return MQTT_STATUS_OK or some error status
 *
 * This function publishes to a topic on a MQTT broker.
 */
mqtt_status_t mqtt_publish(struct mqtt_connection *conn,
                           uint16_t *mid,
                           char *topic,
                           uint8_t *payload,
                           uint32_t payload_size,
                           mqtt_qos_level_t qos_level,
                           mqtt_retain_t retain);
```

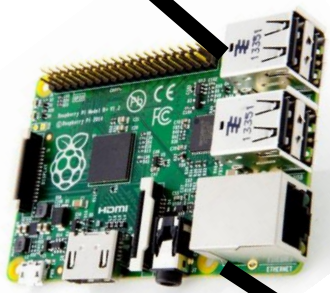
```
/**
 * \brief Set the user name and password for a MQTT client.
 * \param conn A pointer to the MQTT connection.
 * \param username A pointer to the user name.
 * \param password A pointer to the password.
 *
 * This function sets clients user name and password to use when connecting t
 * a MQTT broker.
 */
void mqtt_set_username_password(struct mqtt_connection *conn,
                                char *username,
                                char *password);

    /**
     * \brief Set the last will topic and message for a MQTT client.
     * \param conn A pointer to the MQTT connection.
     * \param topic A pointer to the Last Will topic.
     * \param message A pointer to the Last Will message (payload).
     * \param qos The desired QoS level.
     *
     * This function sets clients Last Will topic and message (payload).
     * If the Will Flag is set to 1 (using the function) this indicates that,
     * if the Connect request is accepted, a Will Message MUST be stored on the
     * Server and associated with the Network Connection. The Will Message MUST
     * be published when the Network Connection is subsequently closed.
     *
     * This functionality can be used to get notified that a device has
     * disconnected from the broker.
     *
     */
    void mqtt_set_last_will(struct mqtt_connection *conn,
                            char *topic,
                            char *message,
                            mqtt_qos_level_t qos);
```

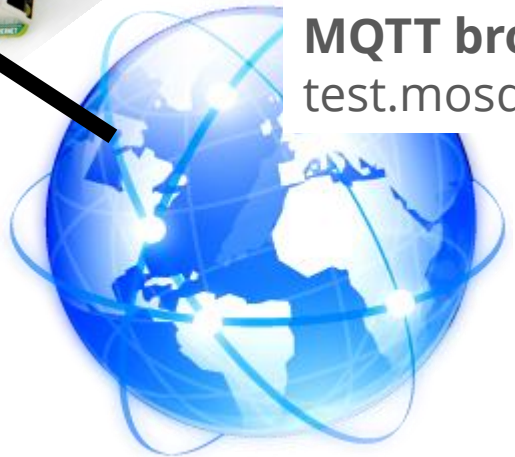
apps/mqtt



Border Router
IPv6/6LoWPAN



MQTT bróker
test.mosquitto.org



mqtt-example
The MQTT client publishing to the topic “zolertia/evt/status” and subscribed to “zolertia/cmd/leds”

mqtt-client.py
Paho MQTT client
Subscribed to the “zolertia/evt/status” topic, used to check the example

examples/zolertia/tutorial/04-mqtt

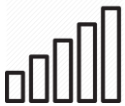
MQTT Broker

PUBLISH

zolertia/evt/status

Uptime

Core temperature



Signal strength

Voltage level

ID/Name

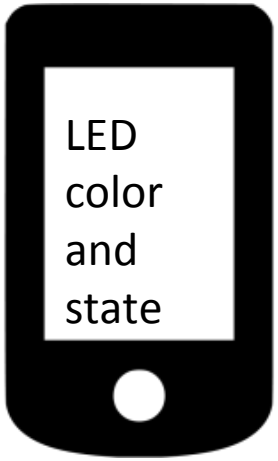
Default Route

examples/zolertia/tutorial/04-mqtt

MQTT Broker

SUBSCRIBE

zolertia/cmd/leds



PUBLISH

zolertia/cmd/leds

examples/zolertia/tutorial/04-mqtt

MQTT Mosquitto

You can run your own MQTT bróker ocally on your machine, and also use the mosquitto-cients to test your application

```
sudo apt-add-repository ppa:mosquitto-dev/mosquitto-ppa
sudo apt-get update
sudo apt-get install mosquitto mosquitto-clients
```

Mosquitto

An Open Source MQTT v3.1/v3.1.1 Broker

Mosquitto

An Open Source MQTT v3.1/v3.1.1 Broker

If you don't have DNS resolv, always double-check the broker addresses!

Address lookup

canonical name	test.mosquitto.org.
aliases	
addresses	2001:41d0:a:3a10::1 37.187.106.16

```
/*
 * If you have an IPv6 network or a NAT64-capable border-router:
 * test.mosquitto.org
 * IPv6 "2001:41d0:a:3a10::1"
 * NAT64 address "::ffff:5577:53c2" (85.119.83.194)
 *
 * To test locally you can use a mosquitto local broker in your host and use
 * i.e the fd00::1/64 address the Border router defaults to
 */
#define MQTT_DEMO_BROKER_IP_ADDR  "fd00::1"
```

examples/zolertia/tutorial/04-mqtt/project-conf.h

- Use the IPv6 address of the Mosquitto MQTT broker
- Use the IPv4 address of the Mosquitto MQTT broker, if we have a NAT64-capable Border Router, or a NAT64 software like wrapsix[9] (out of the scope of this book).
- Use a mosquitto broker running in your host, in this case use the virtual interface address created with `tunslip6`, as shown in previous chapters.

```
/*
 * If you have an IPv6 network or a NAT64-capable border-router:
 * test.mosquitto.org
 * IPv6 "2001:41d0:a:3a10::1"
 * NAT64 address "::ffff:5577:53c2" (85.119.83.194)
 *
 * To test locally you can use a mosquitto local broker in your host and use
 * i.e the fd00::1/64 address the Border router defaults to
 */
#define MQTT_DEMO_BROKER_IP_ADDR  "fd00::1"
```

examples/zolertia/tutorial/04-mqtt/project-conf.h

```
/*---------------------------------------------------------------------------*/
/* Default configuration values */
#define DEFAULT_EVENT_TYPE_ID       "status"
#define DEFAULT_SUBSCRIBE_CMD_TYPE  "leds"
#define DEFAULT_BROKER_PORT         1883
#define DEFAULT_PUBLISH_INTERVAL    (45 * CLOCK_SECOND)
#define DEFAULT_KEEP_ALIVE_TIMER    60

/* Specific platform values */
#if CONTIKI_TARGET_ZOUL
#define BUFFER_SIZE                 64
#define APP_BUFFER_SIZE             512
```

Subscribed to the topic:

zolertia/cmd/leds

Publishes to the topic:

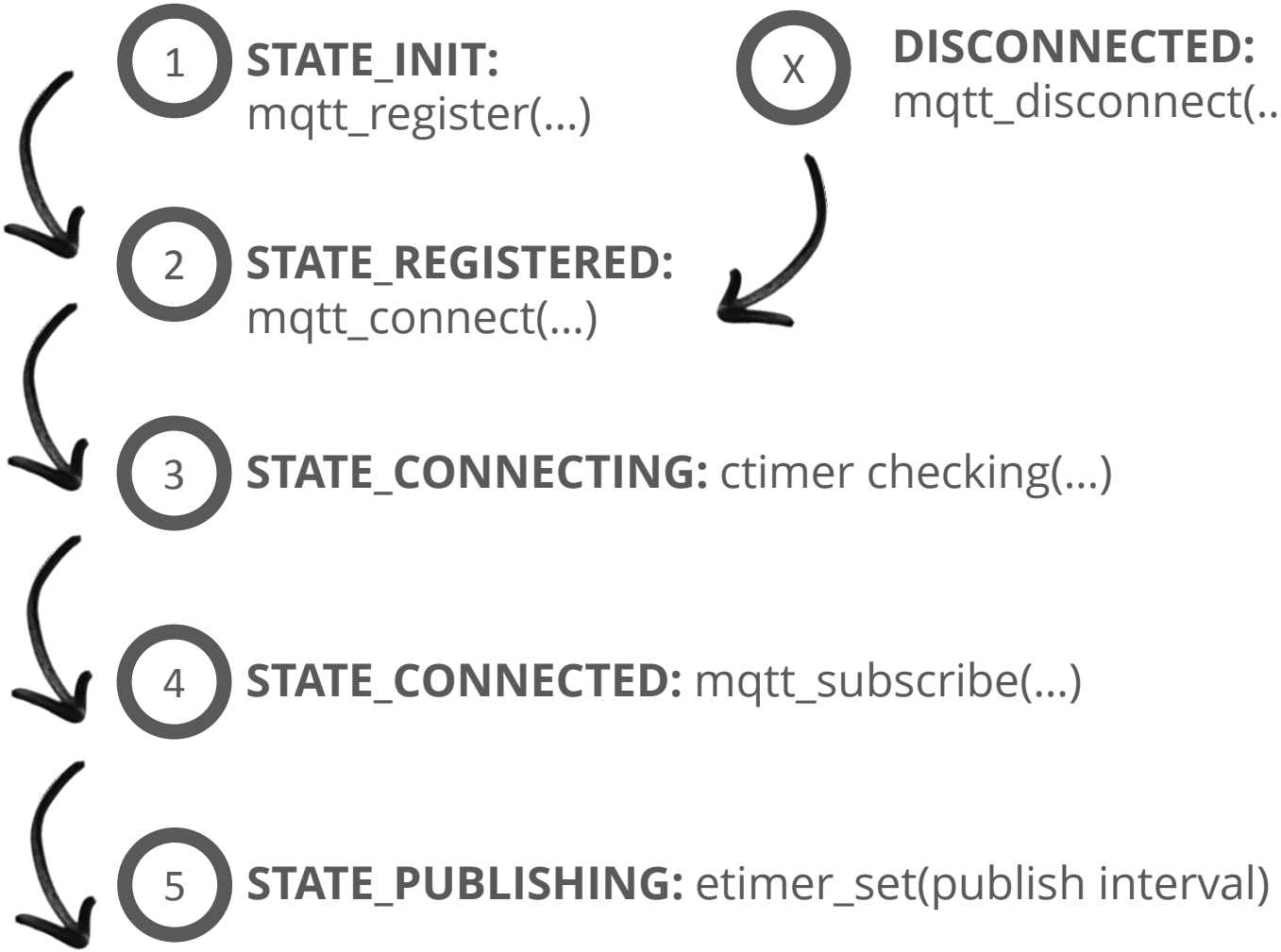
zolertia/evt/status

The publication period (how often the sensor data is published) is given by `DEFAULT_PUBLISH_INTERVAL`. Note the `DEFAULT_KEEP_ALIVE_TIMER` should be always larger than `DEFAULT_KEEP_ALIVE_TIMER`, at least 1.5 times.

examples/zolertia/tutorial/04-mqtt/project-conf.h

```
APPS += mqtt
CONTIKI_WITH_IPV6 = 1
```

examples/zolertia/tutorial/04-mqtt/Makefile



examples/zolertia/tutorial/04-mqtt

```c
/*---------------------------------------------------------------------------*/
/**
 * \brief Data structure declaration for the MQTT client configuration
 */
typedef struct mqtt_client_config {
  char event_type_id[CONFIG_EVENT_TYPE_ID_LEN];
  char broker_ip[CONFIG_IP_ADDR_STR_LEN];
  char cmd_type[CONFIG_CMD_TYPE_LEN];
  clock_time_t pub_interval;
  uint16_t broker_port;
} mqtt_client_config_t;

/*---------------------------------------------------------------------------*/
static int
init_config()
{
  /* Populate configuration with default values */
  memset(&conf, 0, sizeof(mqtt_client_config_t));
  memcpy(conf.event_type_id, DEFAULT_EVENT_TYPE_ID,
         strlen(DEFAULT_EVENT_TYPE_ID));
  memcpy(conf.broker_ip, broker_ip, strlen(broker_ip));
  memcpy(conf.cmd_type, DEFAULT_SUBSCRIBE_CMD_TYPE, 4);

  conf.broker_port = DEFAULT_BROKER_PORT;
  conf.pub_interval = DEFAULT_PUBLISH_INTERVAL;

  return 1;
}
```

examples/zolertia/tutorial/04-mqtt/mqtt-example.c

```
/*
 * Buffers for Client ID and Topic.
 * Make sure they are large enough to hold the entire respective string
 */
static char client_id[BUFFER_SIZE];
static char pub_topic[BUFFER_SIZE];
static char sub_topic[BUFFER_SIZE];
```

```
/*---------------------------------------------------------------------------*/
static int
construct_pub_topic(void)
{
  int len = snprintf(pub_topic, BUFFER_SIZE, "zolertia/evt/%s",
                     conf.event_type_id);
  if(len < 0 || len >= BUFFER_SIZE) {
    printf("Pub Topic too large: %d, Buffer %d\n", len, BUFFER_SIZE);
    return 0;
  }

  return 1;
}
```

examples/zolertia/tutorial/04-mqtt/mqtt-example.c

```
PROCESS_THREAD(mqtt_demo_process, ev, data)
{
  PROCESS_BEGIN();
  if(init_config() != 1) {                                          ❶
    PROCESS_EXIT();
  }
  update_config();                                                  ❷
  while(1) {
    PROCESS_YIELD();
    if((ev == PROCESS_EVENT_TIMER && data == &publish_periodic_timer) ||
       ev == PROCESS_EVENT_POLL) {                                  ❸
      state_machine();
    }
  }
  PROCESS_END();
}
```

❶ Initial configuration values, as described earlier

❷ Creates the client ID, publish and subscribe topics. The initial state `STATE_INIT` is set and the `publish_periodic_timer` event is scheduled

❸ Handles the `publish_periodic_timer`, this is where the application actually starts

examples/zolertia/tutorial/04-mqtt/mqtt-example.c

```
static void
state_machine(void)
{
  switch(state) {
  case STATE_INIT:                                    ❶
    mqtt_register(&conn, &mqtt_demo_process, cli
                  MAX_TCP_SEGMENT_SIZE);
    state = STATE_REGISTERED;

  case STATE_REGISTERED:                              ❷
    if(uip_ds6_get_global(ADDR_PREFERRED) != NUL
      /* Registered and with a public IP. Connec
      connect_to_broker();
    }
    etimer_set(&publish_periodic_timer, NET_CONN
    return;
    break;
```

```
case STATE_PUBLISHING:                                ❺
  if(mqtt_ready(&conn) && conn.out_buffer_sent) {
    /* Connected. Publish */
    if(state == STATE_CONNECTED) {
      subscribe();
      state = STATE_PUBLISHING;
    } else {
      publish();
    }
    etimer_set(&publish_periodic_timer, conf.pub_interval);
    return;
  }
  break;

case STATE_DISCONNECTED:                              ❻
  if(connect_attempt < RECONNECT_ATTEMPTS ||
     RECONNECT_ATTEMPTS == RETRY_FOREVER) {
    mqtt_disconnect(&conn);
    etimer_set(&publish_periodic_timer, interval);
    state = STATE_REGISTERED;
    return;
  }
  break;
}
```

examples/zolertia/tutorial/04-mqtt/mqtt-example.c

```
static void
mqtt_event(struct mqtt_connection *m, mqtt_event_t event, void *data)
{
  switch(event) {
  case MQTT_EVENT_CONNECTED: {                    ❶
    state = STATE_CONNECTED;
    break;
  }
  case MQTT_EVENT_DISCONNECTED: {                 ❷
    state = STATE_DISCONNECTED;
    process_poll(&mqtt_demo_process);
    break;
  }
  case MQTT_EVENT_PUBLISH: {                      ❸
    pub_handler(msg_ptr->topic, strlen(msg_ptr->topic), msg_ptr->payload_chunk,
                msg_ptr->payload_length);
    break;
  }
  case MQTT_EVENT_SUBACK: {                       ❹
    break;
  }
  case MQTT_EVENT_UNSUBACK: {                     ❺
    break;
  }
```

examples/zolertia/tutorial/04-mqtt/mqtt-example.c

```
case MQTT_EVENT_PUBLISH: {                    ❸
  pub_handler(msg_ptr->topic, strlen(msg_ptr->topic), msg_ptr->payload_chunk,
              msg_ptr->payload_length);
  break;
}
```

```
static void
pub_handler(const char *topic, uint16_t topic_len, const uint8_t *chunk,
            uint16_t chunk_len)
{
  /* If we don't like the length, ignore */
  if(topic_len != 17 || chunk_len != 1) {
    printf("Incorrect topic or chunk len. Ignored\n");
    return;
  }

  if(strncmp(&topic[13], "leds", 4) == 0) {
    if(chunk[0] == '1') {
      leds_on(LEDS_RED);
      printf("Turning LED RED on!\n");
    } else if(chunk[0] == '0') {
      leds_off(LEDS_RED);
      printf("Turning LED RED off!\n");
    }
    return;
  }
}
```

examples/zolertia/tutorial/04-mqtt/mqtt-example.c

MQTT demo client running on the device

```
Starting 'MQTT Demo'
MQTT Demo Process
Subscription topic zolertia/cmd/leds
Init
Registered. Connect attempt 1
Connecting (1)
APP - Application has a MQTT connection
APP - Subscribing!
APP - Application is subscribed to topic successfully
Publishing
APP - Publish to zolertia/evt/status
```

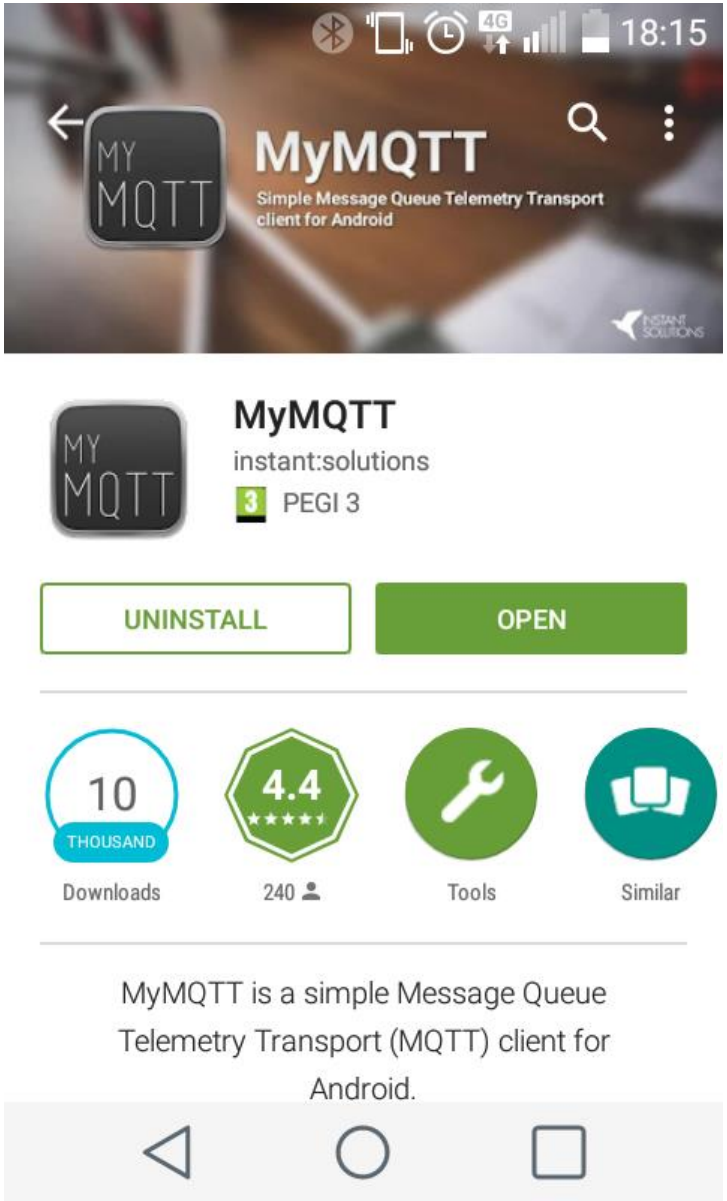
Paho MQTT client in Python subscribed

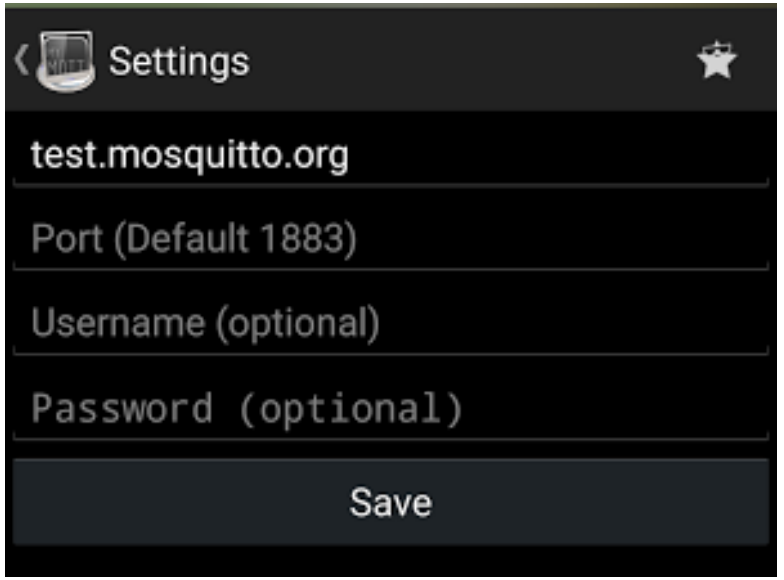
```
$ python mqtt_client.py
connecting to test.mosquitto.org
Connected with result code 0
Subscribed to zolertia/evt/status
Subscribed to zolertia/cmd/leds
zolertia/evt/status {"d":{"myName":"Zolertia Z1 Node","Seq #":3,"Uptime (sec)":141,"Def
zolertia/evt/status {"d":{"myName":"Zolertia Z1 Node","Seq #":4,"Uptime (sec)":186,"Def
```

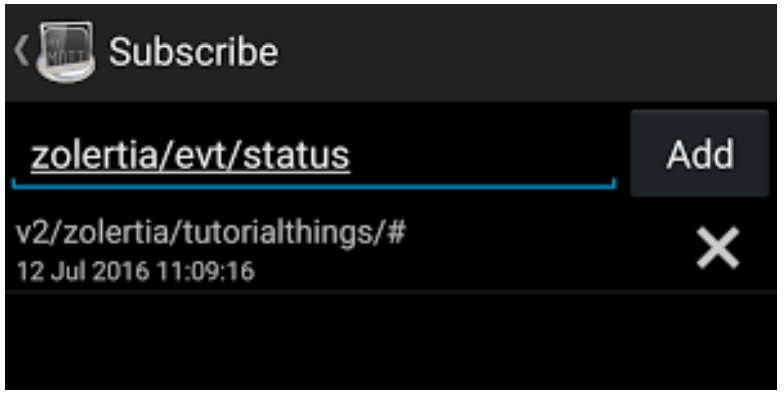
Mosquitto publishing to turn a LED

```
mosquitto_pub -h "test.mosquitto.org"  -t "zolertia/cmd/led" -m "1" -q 1 -r
```

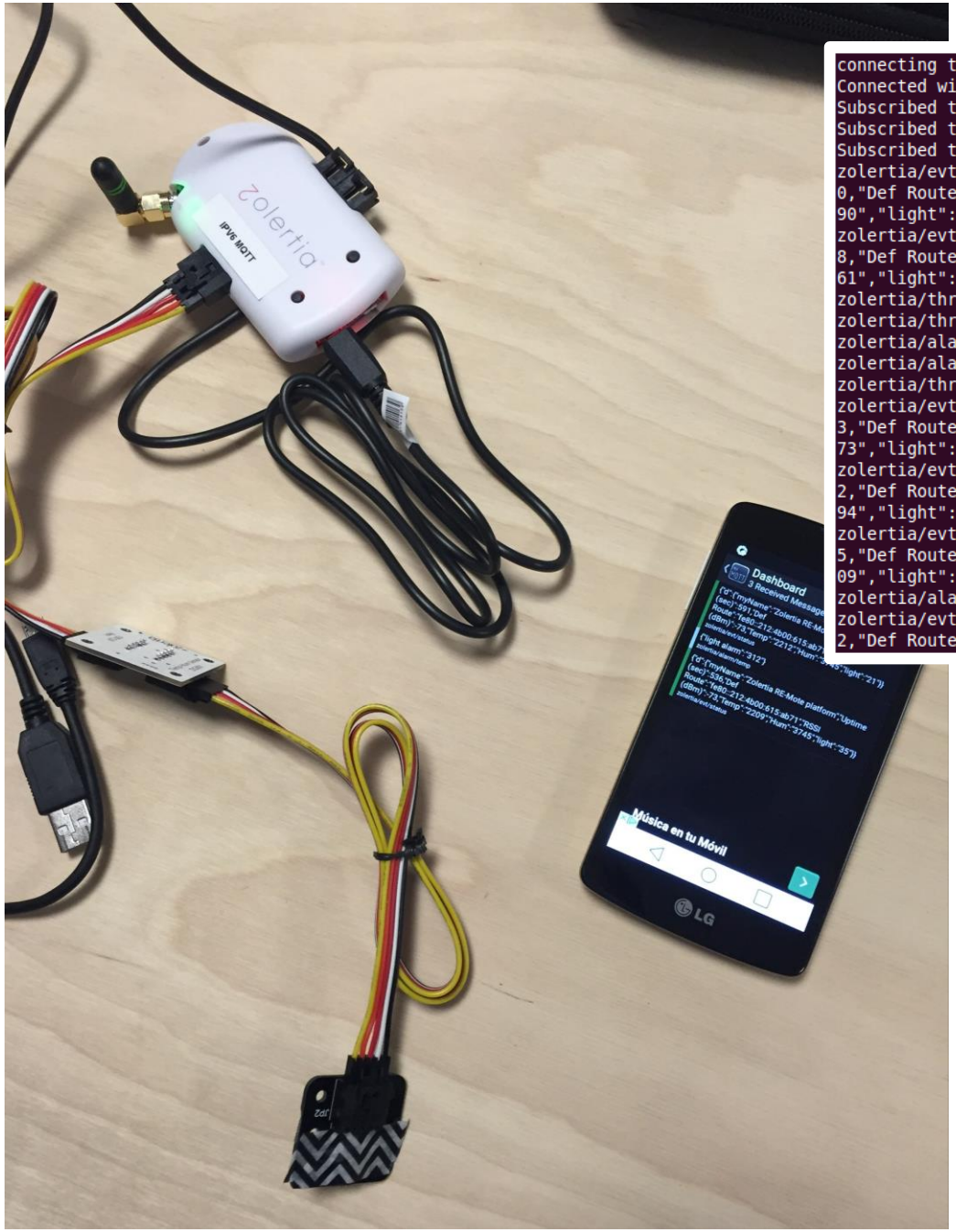
examples/zolertia/tutorial/04-mqtt/mqtt-example.c





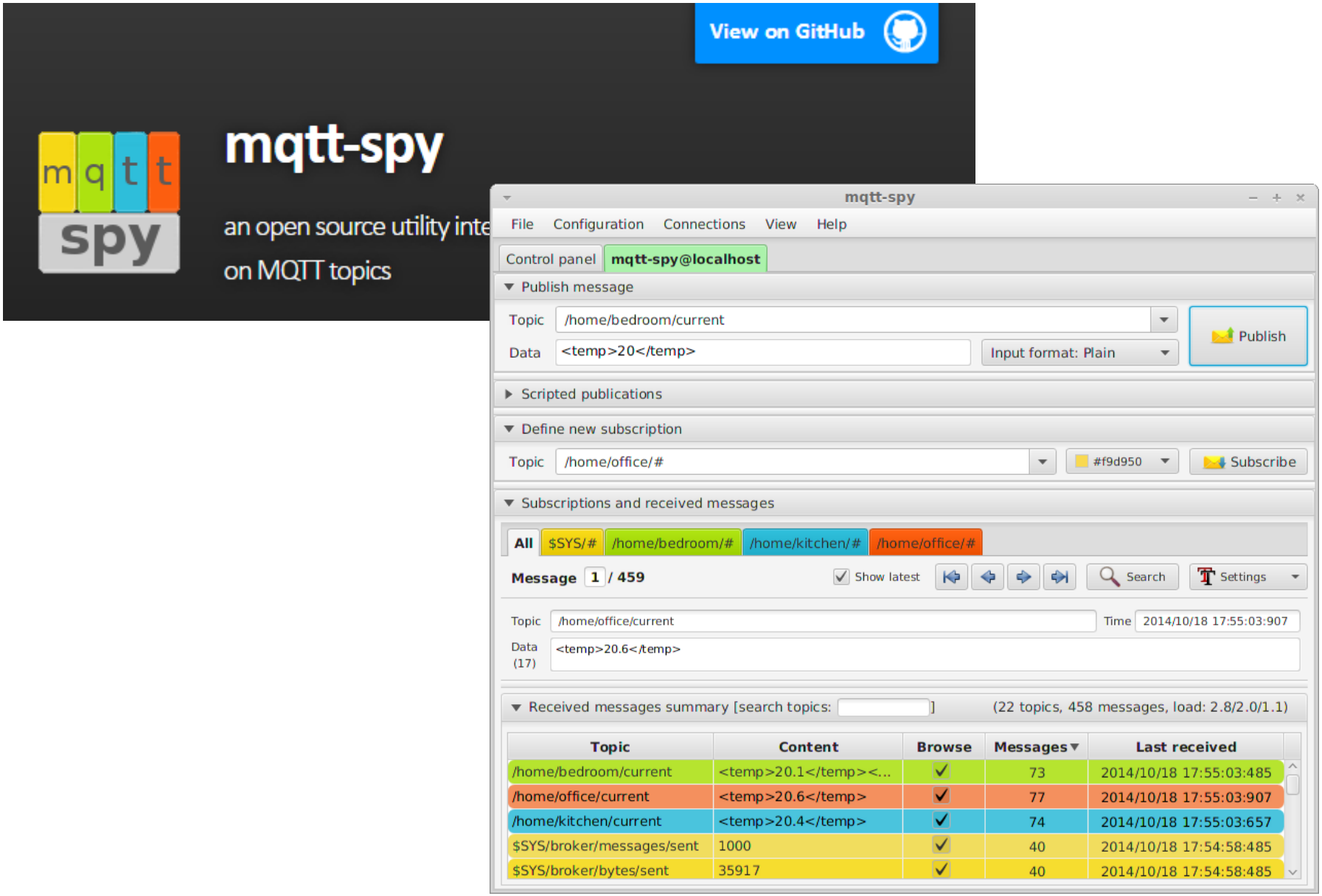


examples/zolertia/tutorial/04-mqtt




```
connecting to test.mosquitto.org
Connected with result code 0
Subscribed to zolertia/evt/status
Subscribed to zolertia/thres/temp
Subscribed to zolertia/alarm/temp
zolertia/evt/status {"d":{"myName":"Zolertia RE-Mote platform","Uptime (sec)":10
0,"Def Route":"fe80::212:4b00:615:ab71","RSSI (dBm)":-73,"Temp":"2302","Hum":"35
90","light":"17"}}
zolertia/evt/status {"d":{"myName":"Zolertia RE-Mote platform","Uptime (sec)":14
8,"Def Route":"fe80::212:4b00:615:ab71","RSSI (dBm)":-73,"Temp":"2277","Hum":"36
61","light":"17"}}
zolertia/thres/temp 4500
zolertia/thres/temp 1500
zolertia/alarm/temp {"temperature alarm":"2265"}
zolertia/alarm/temp {"temperature alarm":"2259"}
zolertia/thres/temp 4500
zolertia/evt/status {"d":{"myName":"Zolertia RE-Mote platform","Uptime (sec)":24
3,"Def Route":"fe80::212:4b00:615:ab71","RSSI (dBm)":-73,"Temp":"2246","Hum":"36
73","light":"18"}}
zolertia/evt/status {"d":{"myName":"Zolertia RE-Mote platform","Uptime (sec)":29
2,"Def Route":"fe80::212:4b00:615:ab71","RSSI (dBm)":-73,"Temp":"2235","Hum":"36
94","light":"18"}}
zolertia/evt/status {"d":{"myName":"Zolertia RE-Mote platform","Uptime (sec)":34
5,"Def Route":"fe80::212:4b00:615:ab71","RSSI (dBm)":-73,"Temp":"2224","Hum":"37
09","light":"18"}}
zolertia/alarm/temp {"light alarm":"302"}
zolertia/evt/status {"d":{"myName":"Zolertia RE-Mote platform","Uptime (sec)":39
2,"Def Route":"fe80::212:4b00:615:ab71","RSSI (dBm)":-73,"Temp":"2219","Hum":"37
```

examples/zolertia/tutorial/04-mqtt



https://kamilfb.github.io/mqtt-spy/

examples/zolertia/tutorial/04-mqtt

Conclusions

You should be able to:

- Create CoAP server applications
- Implement CoAP resources with different response format types
- Use Copper plug-in CoAP client
- Understand CoAP features and functionalities
- Create MQTT client applications
- Create publish/subscribe topics
- Install a local MQTT broker using mosquitto
- Use desktop and mobile applications to debug MQTT applications

Antonio Liñán Colina

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

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