Lora Fabian Introduction Workshop

Purpose of this document.

This document aims at giving reader(s) a progressive way to enter into the Lora Fabian Stack.

About Lora Fabian

Lora Fabian is a Network Protocols Stack for the Internet of Things needs. Althought being mostly designed for Lora(tm) and the associated constraints (low bandwidth..) it can be reused on top of every Layer-1 technology.

Lora Fabian uses CoAP for signaling (node registration etc...) and offer nodes a CoAP channel that can be mapped to HTTP(s) using a simple one to one translation mechanism. That way, objects can be accessed using classical Internet protocols, with their own DNS name. It aims at abstracting IoT network complexity while providing a complete Web-like REST schemantic. Simple HTTP(s) URLs can be used to interact with the nodes, like http://myobject.example.org/helloworld.

Part I: Discovering Lora Fabian

A) Node Side

Download the files available in http://lora-serv.fablabnet.fr/tp/ and open them with the Arduino Editor.

Take a few 5,10, 15 minutes to read the code and understand:

- What this code do.
- The role of the different files.
- The process and the messages flow.

What is this code doing ?

.....
.....
.....

What are the role of the different files ?

LoraTP1.ino: .....
.....
.....

loraShield.cpp : .....
.....
.....
.....

Coap.c : .....
.....
.....

Endpoints.h: .....
.....
.....

What are the process and message flow:

When receiving a message ?

.....
.....
.....
.....
.....

When seeding a message ?

.....
.....
.....
.....
.....

B) Internet Side

Try to use the unix command “dig AAAA +short nodeName.s.ackl.io”.

What do you see ?

.....
.....
.....

What does that mean ?

.....
.....
.....

Try to use the same unix command to find the IPv6 address of another node

What do you see ?

.....
.....
.....

What does that mean ?

.....
.....

.....

Try to access http://nodeName.s.ackl.io/helloWorld

What do you see ?

.....
.....
.....

What does that mean ?

.....
.....
.....

Descrive the flow followed by your HTTP request and the response you received:

.....
.....
.....
.....
.....
.....

Part II: Going Further into Lora Fabian

Internet of Things can have better purpose than sending hello worlds :)

A) Simple LED Application

If you give a detailled look on the files you can see some code definition for a LED.

By putting “digitalWrite(led, HIGH);” or “digitalWrite(led, LOW);” you can power on/off a LED.

Try to do it and to have the led blinking at a regular interval using the loop(){}

How does the code looks like:

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

We have some code to make a LED to blinks and some code to control Arduino using CoAP and a simple HTTP translation. We will now make them working together:

In the endpoint.c file:

- Add an handler handle_get_led_on() make it power on the LED and return a simple CoAP response.
- Same for a handle_get_led_off().
- Do not forget to add them in the endPoints list
- Try to launch your code usingTry to access
  - http://nodeName.s.ackl.io/ledon
  - http://nodeName.s.ackl.io/ledoff

How does the code looks like:

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

B) A bit more of engeenering of this LED Application.

The simple CoAP-controled LED application we wrote do not really follow the REST Schemantic. It can be interesting to have a real and properly engeenered application.

Step 1) Add a context (boolean), reflecting the state of the LED into the application and update the context in the led_on/led_off handler

Step 2): Add a new led/ handler that once receiving a GET request ouputs the state of the LED.

Step 3) Remove the digitalWrite from the ledon ledoff handlers and find a good place to move them

How does the code looks like:

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Step 4) add a new led/ handler that will now receive PUT requests and update the context.

How does the code looks like:

.....
.....
.....
.....
.....
.....
.....

Part III (optional): Back to the Internet world

Now that we have a real RESTfull CoAP application we can control it using common internet protocols.

Use your favorise language, can be Javascript, Python, Shell, ASM68000.. to interact with our Application and compare your implentation with another group.

Part IV (optional): Let's communicate

Now that you got your hand on a stable and simple implentation on the node side and on the client side you can made objets communicate with each others. Mix with one other group, define scenarios where you not only control one object but consider the state of the other one, and implement them.