

Course Project

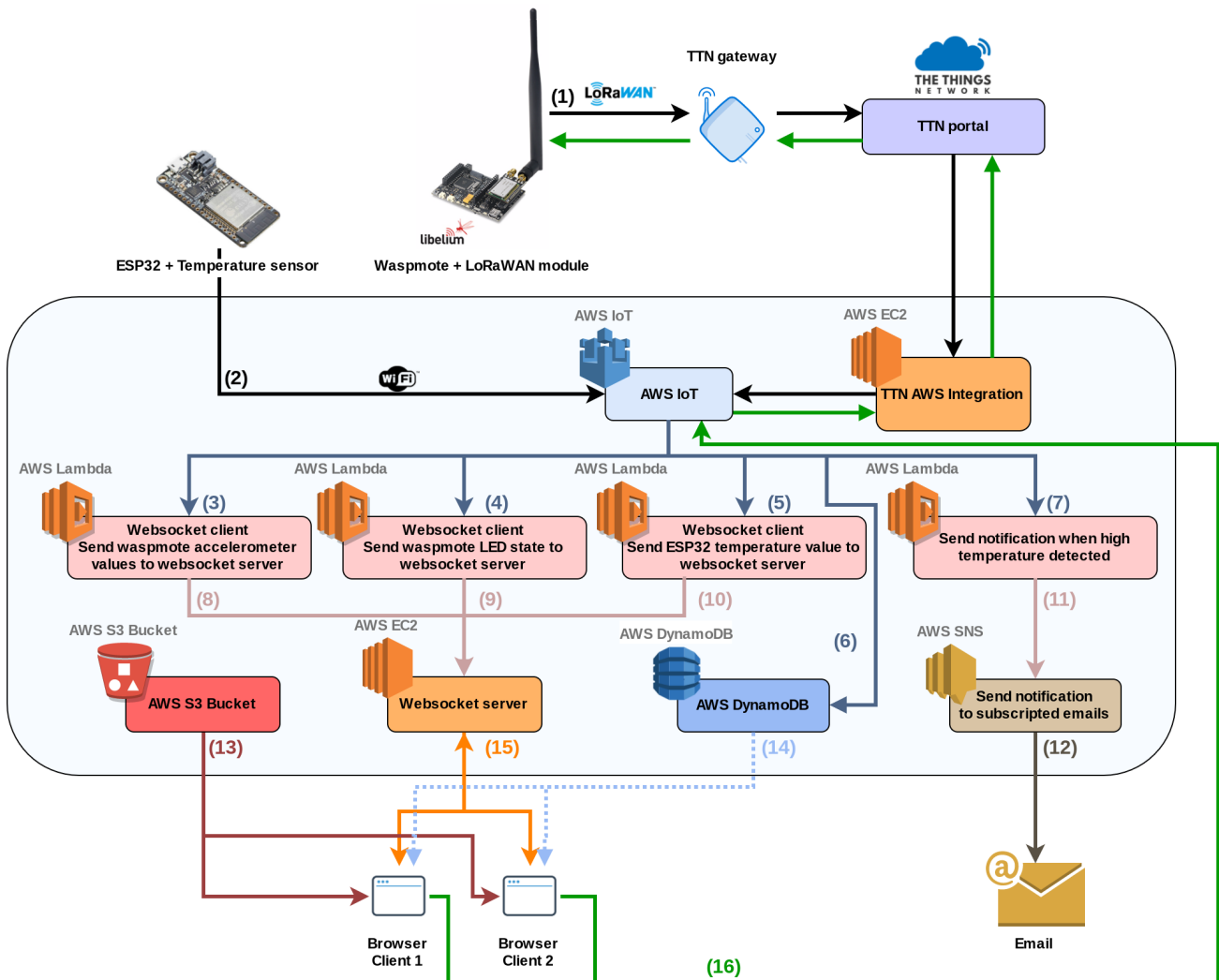
IoT - HES-SO Master

2018 - 2019 Winter Semester

The complete IoT project

Raed Abdennadher & Ludovic Gindre

1. Architecture globale



2. Description

(1) : Message LoRaWAN envoyé par le waspmote à TTN (1 par 30 secondes) contenant 3 valeurs de l'accéléromètre (x, y et z) et l'état de la LED1 en hexadécimal.

- Message en hexadécimal :
x : 16 bits, y 16 bits, z 16 bits et led state 8 bits.
- Exemple : `04 82 FF 97 00 5F 00 01`
- En arrivant dans le portail de TTN, ce message sera décodé (Payload Format decoder) et retourné en format JSON :

```
{
  "led_state": 1,
  "x_acc": 1154,
  "y_acc": -105,
  "z_acc": 95
}
```

(2) : Message MQTT envoyé par ESP32 (1 pat minute) contenant la valeur du capteur de température en degré Celsius. Exemple JSON :

```
{
  "Name": "Inside",
  "temp": 24.2
}
```

(3) : Lors de la réception de données du waspmote, une action est lancée avec la *Rule query statement* suivante (concerne l'accéléromètre) :

```
SELECT dev_id, payload_fields.x_acc AS x_acc, payload_fields.y_acc AS y_acc,
payload_fields.z_acc AS z_acc, timestamp() AS Time FROM
'iot_2018_19_abdennadher_gindre/devices/waspmote_0/up'
```

Cette action va lancer un script d'une fonction *Lambda* `send_accelero_state` qui recevra les données sous forme JSON. Exemple :

```
{
  "dev_id": "waspote_0",
  "x_acc": 1154,
  "y_acc": -105,
  "z_acc": 95,
  "Time": 1546533762406
}
```

RULE

accelerometer_rule

DISABLED

Actions ▾

Overview

Description

The accelerometer of waspmote

Edit

Rule query statement


The source of the messages you want to process with this rule.

```
SELECT dev_id, payload_fields.x_acc AS x_acc, payload_fields.y_acc AS y_acc,
payload_fields.z_acc AS z_acc, timestamp() AS Time FROM
'iot_2018_19_abdennadher_gindre/devices/waspmote_0/up'
```

Using SQL version 2016-03-23

Actions

Actions are what happens when a rule is triggered. [Learn more](#)



Send a message to a Lambda function

send_accelero_state

Remove Edit ▶

(4) : Lors de la réception de données du waspmote, une autre action est lancée avec la *Rule query statement* suivante (concerne l'état de la LED) :

```
SELECT payload_fields.led_state AS led_state FROM
'iot_2018_19_abdennadher_gindre/devices/waspmote_0/up'
```

Cette action va lancer un script d'une fonction *Lambda* `send_led_state` qui recevra les données sous forme JSON. Exemple :

```
{
  "led_state": "1"
}
```

RULE

led_rule

DISABLED

Actions ▾

Overview

Description

The led1 state of waspmote

Edit

Rule query statement

The source of the messages you want to process with this rule.


```
SELECT payload_fields.led_state AS led_state FROM
'iot_2018_19_abdennadher_gindre/devices/waspmote_0/up'
```

Using SQL version 2016-03-23

Edit

Actions

Actions are what happens when a rule is triggered. [Learn more](#)



Send a message to a Lambda function

send_led_state

Remove Edit ▶

(5) et (6) : Lors de la réception de données du ESP32, une action est lancée avec la *Rule query statement* suivante (concerne la température) :

```
SELECT Name AS Name, temp AS temp, timestamp() AS Time FROM 'sensor/temp'
```

Cette action va lancer un script d'une fonction *Lambda* `send_last_temperature` et va enregistrer les données dans une table `temp` DynamoDB qui recevront les données sous forme JSON. Exemple :

```
{
  "Name": "Inside",
  "temp": 24.2
}
```

Overview

Description

Get last temperature value from the topic "sensor/temp" (esp32)

Edit

Rule query statement

The source of the messages you want to process with this rule.


SELECT Name AS Name, temp AS temp, timestamp() AS Time FROM 'sensor/temp'

Using SQL version 2016-03-23


Edit

Actions

Actions are what happens when a rule is triggered. [Learn more](#)

 Split message into multiple columns of a Dyna...
temp

Remove Edit ▸

 Send a message to a Lambda function
send_last_temperature

Remove Edit ▸

Create table Delete table

Filter by table name X

Name

temp

temp Close

Overview Items Metrics Alarms Capacity Indexes Global T

Create item Actions ▾

Scan: [Table] temp: Name, Time ▾

	Name ⓘ	Time	temp
<input type="checkbox"/>	Inside	1546221857559	18.269043
<input type="checkbox"/>	Inside	1546221917676	17.983398
<input type="checkbox"/>	Inside	1546221977783	18.554688
<input type="checkbox"/>	Inside	1546222037879	18.364258
<input type="checkbox"/>	Inside	1546222097963	18.459473
<input type="checkbox"/>	Inside	1546222158086	18.173828
<input type="checkbox"/>	Inside	1546222218183	18.459473
<input type="checkbox"/>	Inside	1546222278316	18.554688

(7) : Lors de la réception d'une température du ESP3 supérieure à 30°, une action est lancée avec la *Rule query statement* suivante (concerne la température) :

```
SELECT * FROM 'sensor/temp' where temp > 30
```

Cette action va lancer un script d'une fonction *Lambda* `high_temperature` qui recevra les données sous forme JSON. Exemple :

```
{
  "Name": "Inside",
  "temp": 31.5
}
```

RULE

high_temperature_rule

DISABLED

Actions ▾

Overview

Description

Rule for high temperature detected (temp > 30°)

Rule query statement


The source of the messages you want to process with this rule.

SELECT * FROM 'sensor/temp' where temp > 30

Using SQL version 2016-03-23

Actions

Actions are what happens when a rule is triggered. [Learn more](#)

 Send a message to a Lambda function

high_temperature

Remove Edit ▶

(8) : La fonction *Lambda* `send_accelero_state` envoie un message sous forme JSON via websocket sous la forme suivante :

```
{
  "action": "accelero",
  "x_acc": "x_acc value",
  "y_acc": "y_acc value",
  "z_acc": "z_acc value",
  "time": "time value"
}
```

(9) : La fonction *Lambda* `send_led_state` envoie un message sous forme JSON via websocket sous la forme suivante :

```
{
  "action": "led",
  "state": "led state"
}
```

(10) : La fonction *Lambda* `send_last_temperature` envoie un message sous forme JSON via websocket sous la forme suivante :

```
{
  "action": "temp",
  "temp": "temperature value",
  "time": "time value"
}
```

(11) : La fonction *Lambda* `high_temperature` envoie un message JSON sous la forme du point 6 au service de notification *SNS* à la rubrique (*Topic*) `esp32_alarm_notification`

(12) : Dans la rubrique (*Topic*) on a une souscription (*Subscription*) via le protocole *email* pour envoyer un email à une adresse personnelle

Topic details: esp32_alarm_notification

Publish to topic

Other topic actions ▾

Topic ARN arn:aws:sns:eu-west-1:166010466283:esp32_alarm_notification
Topic owner 166010466283
Region eu-west-1
Display name
Encryption at rest Disabled ⓘ

Subscriptions

Create subscription

Request confirmations

Confirm subscription

Other subscription actions ▾

Filter

<input type="checkbox"/>	Subscription ID	Protocol	Endpoint	Subscriber
<input type="checkbox"/>	arn:aws:sns:eu-west-1:166010466283:...	email	abdennadher.raed@gmail.com	166010466283

(13) : On a configuré un site web statique dans le service *S3 Bucket*. Le lien vers ce site est le suivant: <http://website-iot.s3-website-eu-west-1.amazonaws.com/>. En effectuant cette requête http, l'utilisateur recevra les fichiers html, js, css et images.

Amazon S3 > website-iot

Overview

Properties

Permissions
Public

Management

Upload

Create folder

Download

Actions

EU (Ireland)

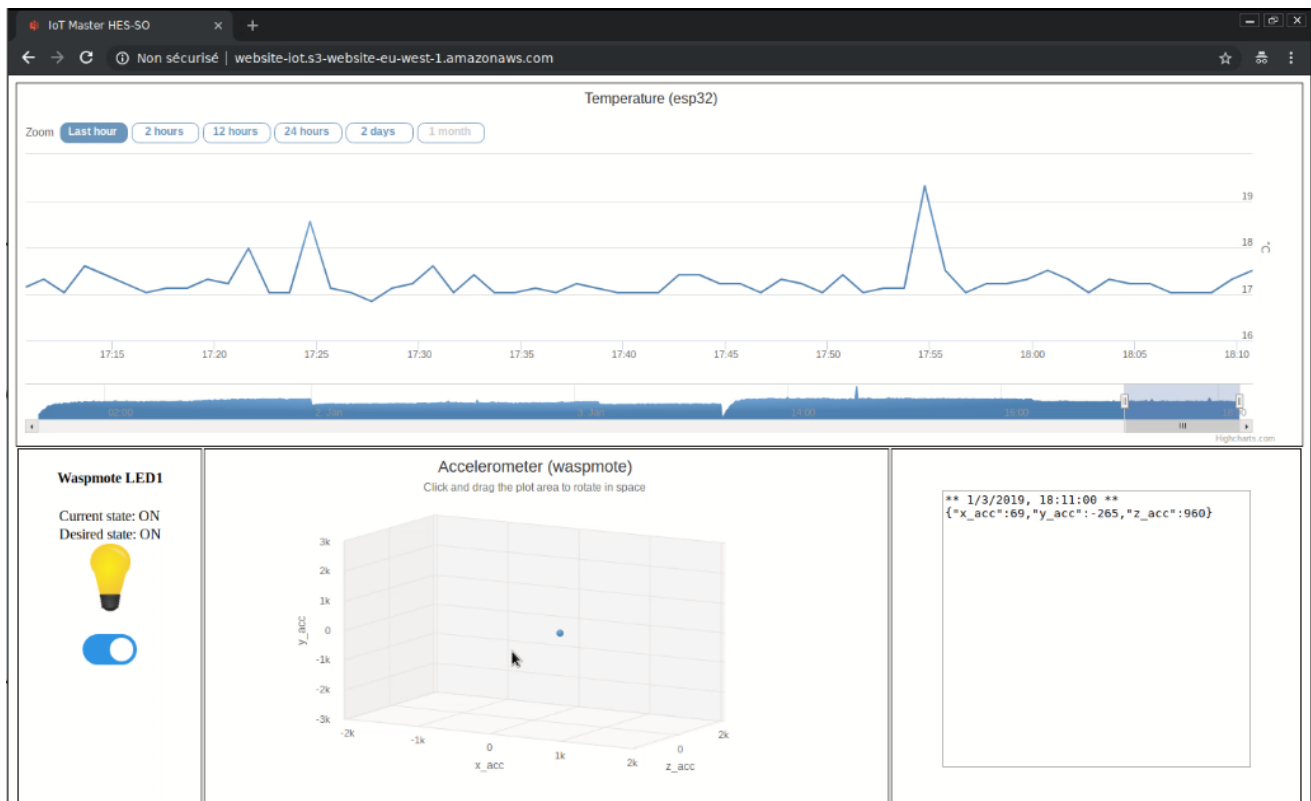
Viewing 1 to 4

<input type="checkbox"/>	Name	Last modified	Size	Storage class
<input type="checkbox"/>	css	--	--	--
<input type="checkbox"/>	img	--	--	--
<input type="checkbox"/>	js	--	--	--
<input type="checkbox"/>	index.html	Jan 3, 2019 5:55:19 PM GMT+0100	2.4 KB	Standard

Viewing 1 to 4

(14): En chargeant la page `index` du site, un script js va solliciter la table `temp` de DynamoDB et récupérer toutes les températures enregistrées dans un diagramme (*Chart*).

```
var dynamodb = new AWS.DynamoDB();
...
dynamodb.query(params, function(err, data) {
    if (err) {
        logging(err);
        return reject(err);
    }
    let dataTemp = [];
    for (var i in data['Items']) {
        TemperatureRead = parseFloat(data['Items'][i]['temp']['N']);
        TimeRead = parseFloat(data['Items'][i]['Time']['N']);
        dataTemp.push({
            x: TimeRead,
            y: TemperatureRead
        })
    }
    ...
});
...
```

(15): Le serveur Websocket (écrit Python dans une instance EC2 avec une adresse ip élastique) diffuse les données reçus par les fonctions *Lambda* ainsi que l'action de changement de l'état de la LED, pour actualiser l'affichage des page html ouvertes de tous les clients :

```
according to the value of the key 'action' of the received message:
'led'           => broadcast {'type': 'led', 'state': state}
'temp'          => broadcast {'type': 'temp', 'temp': temp, 'time': time}
'accelero'      => broadcast {'type': 'accelero', 'x_acc': x_acc, 'y_acc': y_acc,
                             'z_acc': z_acc, 'time': time}
'requestState'  => broadcast {'type': 'pending', 'desiredState': state}
```

(16): Quand l'utilisateur demande le changement de l'état de la LED (en cliquant sur le check button), deux actions vont se produire :

1. Envoyer un message JSON au serveur websocket sous la forme suivante :

```
{"action": "requestState", "state": desiredLedState}
```

2. Publier un message à *AWS IoT (DownLink)*

```
var iotdata = new AWS.IotData({endpoint: 'a72x50k0riqjj-ats.iot.eu-west-1.amazonaws.com'});
...
// Event handler for changing LED state request by checking/unchecking the checkbox
function changeLedStateRequest(checkbox) {
    desiredLedState = "0";
    if (checkbox.checked) {
        desiredLedState = "1";
    }
}
```

```

    }
    websocket.send(JSON.stringify({action: "requestState", state: desiredLedState}));
    publish();
}

// Publish a downlink
function publish() {
    let payloadJson = '{"port": 1,"confirmed": false,"payload_raw": "' +
    btoa(desiredLedState) + '"}', /* btoa convert string to base64 */
    waspmoteTtnTopic = 'iot_2018_19_abdennadher_gindre/devices/waspmote_0/down';
    let params = {
        topic: waspmoteTtnTopic,
        payload: payloadJson,
        qos: 0
    };
    iotdata.publish(params, function(err, data) {
        if (err) {
            // an error occurred
            logging(err.stack);
        } else {
            // successful response
            logging("publish: " + payloadJson + " to topic: " + waspmoteTtnTopic);
        }
    });
}

```