TP3 : Javascript, MongoDB, ...

Gilles Menez - UNS - UFR Sciences - Dépt. Informatique

March 25, 2020

Les codes sont donnés ... sur le site ... et dans l'annexe de ce document.

1 De nouvelles ambitions

On est toujours dans une architecture de type : "objets (ESP32) <-> Station de monitoring (PC)". Mais on aimerait réaliser une application de gestion d'objets de l'Internet qui soit **plus dynamique et plus attractive**.

1.1 ESP : MQTT

L'ESP utilise MQTT pour publier ses valeurs.

- ➢ Normalement, cette première phase est maîtrisée (par vous) puisqu'on l'a déjà fait.
- ➢ Le code initial se trouve dans le répertoire "SurlESP"

1.1.1 ESP : Analyse du code

① L13-16 : On va manipuler les capteurs habituels

② L18-19 : L'ESP va être un client MQTT

③ L21 : L'ESP va ce choisir un "identifiant" de flotte.

④ L25-28 : IP du Serveur MQTT et topics

⑤ L55 : L'ESP est subscriber du topic LED => définition de la fonction callback invoquée par la bibliothèque "PubSubClient" dès réception d'un message MQTT.

Il faut alors analyser le topic et le message reçu.

On peut ainsi lui demander d'allumer/éteindre une LED.

⑥ L89 : Fonction de subscribing à un topic particulier.

Il faut être connecté au préalable.

⑦ L107-130 : Des accesseurs ... pour être propre.

⑧ A partir ligne 130 :

L'ESP souscrit au topic LED et il publie sur les topics TEMP et LIGHT.

Le message (payload) respecte la notation JSON

1.1.2 Snapshot ESP

On lance tout ça et on vérifie quand même au niveau du PC que l'information arrive :

1.2 Phase 1 : Gestion des messages MQTT

L'objectif de cette première phase est de pouvoir mémoriser au niveau de la stations (PC) les échantillons produits par l'ESP :

- ➢ Pour les traiter et les analyser
- ➢ ou pour les afficher (plot).

On va utiliser pour cela une base de données NoSQL (=> MongoDb) et un Node JS serveur dont le rôle sera de récupérer chaque message MQTT et après l'avoir analyser, le placer si il le faut dans la base.

1.2.1 MongoDB : Base de données

- ➢ Je vous laisse installer la base de données sur votre station :

  `https://docs.mongodb.com/manual/administration/install-community/`

  Sous Debian . . . les packages sont disponibles dans le repository !

  Ensuite .. il y a de la documentation !

  - ✔ `https://docs.mongodb.com/`

1.2.2 Node.js : Javascript côté serveur

- ➢ Je vous laisse installer l'outil sur votre station :

  `https://nodejs.org/en/download/`

  plus quelques modules JS :

  - ✔ npm install express
  - ✔ npm install mqtt
  - ✔ npm install mongodb

1.2.3 ESP : Analyse du code

① L15 : identifiants du serveur et de la base de données

② L17 : connection au serveur de la base.

③ L25 : connection à la base.

④ L30 : connection au broker (le même que celui utilisé par l'ESP)

⑤ L35 : installation du callback de l'event connect du client_mqtt
=> Souscription immédiate aux topics qui nous intéressent : lumière et température.

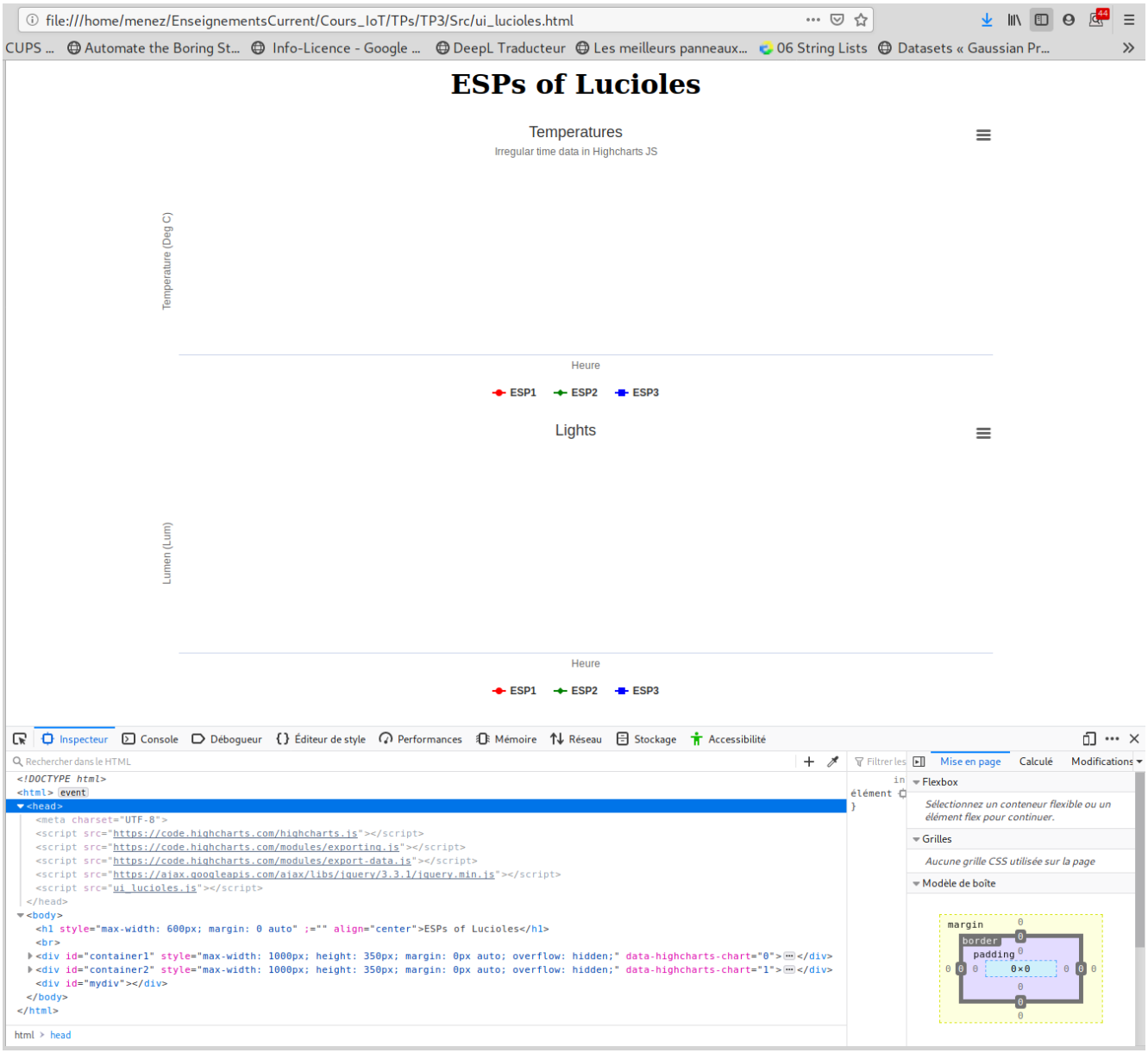
⑥ L50 : callback des messages MQTT

- ➢ On récupère un message JSON dont on extrait l'identifiant de flotte et la valeur.
- ➢ On fabrique un dictionnaire que l'on va stocker dans la base de données. On intègre un timestamp.
- ➢ Le nom du topic sert de clé dans la base. Est-ce la meilleure organisation pour la base ? c'est vous les spécialistes . . . à vous de dire !
- ➢ Insertion dans la collection sélectionnée par la key de cette nouvelle entrée.

1.2.4 Snapshot Phase1

1.3 Phase 2 : User Interface

Cette deuxième phase d'évolution de l'outil vise à présenter les données récoltées et stockées dans la database.



1.3.1 Page Web de l'UI

Dans le but de l'ouvrir avec un navigateur, on crée une page Web `ui_lucioles.html` qui va exploiter le package javascript `code.highcharts.com` pour dessiner des courbes dans deux containers :

- ✔ "container1"
- ✔ "container2"

Cette page utilise un Javascript `ui_lucioles.js` pour faire le lien avec le Node serveur que l'on a débuté dans la phase 1.

La phase 2 doit donc compléter le code du Node serveur pour gérer les requêtes en provenance du client/-navigateur Web : `node_lucioles_v2.js`

1.4 Phase 3 : A cause du coronavirus

Compte tenu du confinement que nous respectons tous, j'ai fait évoluer la localisation des serveurs pour les placer hors de UCA.

Ce n'est pas plus mal, je trouve qu'on se rapproche d'un déploiement réaliste.

① La base de données sur `https://www.mongodb.com/cloud/atlas`

② Le broker MQTT sur `hivemq.com`

③ Et le Node JS sur `https://medium.com/swlh/how-to-deploy-your-node-js-app-to-heroku-cf3b9cc31`
Celui là je ne l'ai pas finalisé, mais M.Buffa me dit que c'est du tout cuit ;-)

Les codes ont un tout petit peu évolué.

2 Travail à fournir

Cette application ne doit être qu'un point de départ.

Il faut donc :

① Comprendre,

② Faire marcher,

③ Puis faire évoluer !

2.1 Evolution

Je vais préciser un peu le cahier des charges de l'évolution :

✔ On va typer cela "e-santé" . . . pour être utile !

➢ Vous allez former des groupes de 5 au minimum au moins le temps de choisir des topics et des payloads communs au groupe.

. . . ensuite vous pouvez travailler de votre côté, ou pas (comme vous voulez) ?

. . . mais attention sur la User Interface, je dois tous vous voir !

➢ Vous devez faire évoluer la partie UI (User Interface) pour que l'on puisse visualiser les capteurs de chacun des objets des membres du groupes. Essayer d'avoir une approche ergonomique !

Si il faut modifier autre chose que l'UI . . . faîtes donc !

➢ Sur cette UI, vous devez faire apparaître un bouton qui permette de "pinger" chaque objet/personne du groupe.

Imaginez que l'objet permette la surveillance d'un patient, lors du ping, la Led de l'objet va s'allumer. Si le patient va bien alors il va "éteindre" le capteur de lumière en mettant son doigt dessus.

L'objet va le détecter (edge computing) et transmettre une réponse au ping.

➢ Cette réponse doit avoir un effet sur l'UI !

Il faut aussi que je puisse envoyer les secours et donc il me faut un moyen d'accéder à l'identification de chaque capteur. Vous décrirez le protocole que vous mettez en place pour obtenir cette information. Essayez de d'être "économes".

Quand cela fonctionne vous m'envoyez les codes nécessaires qui me permettront de voir et de faire fonctionner le groupe.

Et le top serait que je puisse y adhérer . . . peut être grâce à un topic "adhesion" ?

3 Annexe : Codes

3.1 esp32_lucioles/esp32_lucioles.ino

```
/*
 * Auteur : G.Menez
 */
#include <WiFi.h>
#include <PubSubClient.h>
#include <ArduinoJson.h> // by Benoit Blanchon
#include <Wire.h>
#include "OneWire.h"
#include "DallasTemperature.h"
#include "net_misc.h"

/*============== GPIO ========================*/
const int ledPin = 19; // LED Pin
const int photo_resistor_pin = A0;
OneWire oneWire(23);
DallasTemperature tempSensor(&oneWire);

WiFiClient espClient; // Wifi
PubSubClient client(espClient) ; // MQTT client

String whoami; // Identification de CET ESP au sein de la flotte

//StaticJsonBuffer<200> jsonBuffer;

/*====== MQTT broker/server and TOPICS =========*/
//const char* mqtt_server = "192.168.1.100";
const char* mqtt_server = "broker.hivemq.com";
#define TOPIC_TEMP "sensors/temp"
#define TOPIC_LED "sensors/led"
#define TOPIC_LIGHT "sensors/light"

/*============== SETUP ======================*/

void setup () {
  // Gpio
  pinMode (ledPin , OUTPUT);
  // Serial
  Serial.begin (9600);

  /* Wifi */
  connect_wifi();

  /*  L'ESP est un client du mqtt_server */
  client.setServer(mqtt_server, 1883);
  // set callback when publishes arrive for the subscribed topic
  // methode a effet local => on n'a pas a initier/verifier la connection.
  client.setCallback(mqtt_pubcallback) ;

  /* Choix d'une identification pour cet ESP ---*/
  // whoami = "esp1";
  whoami =  String(WiFi.macAddress());
}

/*============== MQTT CALLBACK ===============*/

void mqtt_pubcallback(char* topic, byte* message, unsigned int length) {
  /*
```

```
58     *  Callback if a message is published on this topic.
59     */
60
61   // Byte list to String ... plus facile a traiter ensuite !
62   // Mais sans doute pas optimal en performance => heap ?
63   String messageTemp ;
64   for(int i = 0 ; i < length ; i++) {
65     messageTemp += (char) message[i];
66   }
67
68   Serial.print("Message : ");
69   Serial.println(messageTemp);
70   Serial.print("arrived on topic : ");
71   Serial.println(topic) ;
72
73   // Analyse du message et Action
74   if(String (topic) == TOPIC_LED) {
75      // Par exemple : Changes the LED output state according to the message
76     Serial.print("Action : Changing output to ");
77     if(messageTemp == "on") {
78       Serial.println("on");
79       set_pin(ledPin ,HIGH);
80
81     } else if (messageTemp == "off") {
82       Serial.println("off");
83       set_pin(ledPin ,LOW);
84     }
85   }
86 }
87
88 /*============== MQTT SUBSCRIBE ======================*/
89
90 void mqtt_mysubscribe(char* topic) {
91   /*
92    * ESP souscrit a ce topic. Il faut qu'il soit connecte.
93    */
94   while(!client.connected()) { // Loop until we are reconnected
95     Serial.print("Attempting MQTT connection...");
96     if(client.connect("esp32", "try", "try")) { // Attempt to connect
97       Serial.println("connected");
98       client.subscribe(topic); // and then Subscribe
99     } else { // Connection failed
100      Serial.print("failed , rc=");
101      Serial.print(client.state());
102      Serial.println("try again in 5 seconds");
103      // Wait 5 seconds before retrying
104      delay(5*1000);
105    }
106  }
107 }
108
109 /*============== ACCESSEURS ======================*/
110
111 float get_temperature() {
112   float temperature;
113   tempSensor.requestTemperaturesByIndex(0);
114   delay (750);
115   temperature = tempSensor.getTempCByIndex(0);
116   return temperature;
117 }
118
119 float get_light(){
```

```
120   return analogRead(photo_resistor_pin);
121 }
122
123 void set_pin(int pin, int val){
124  digitalWrite(pin, val) ;
125 }
126
127 int get_pin(int pin){
128   return digitalRead(pin);
129 }
130
131 /*================== LOOP ======================*/
132 void loop () {
133   char data[80];
134   String payload; // Payload : "JSON ready"
135   int32_t period = 60 * 10001; // Publication period
136
137   /* Subscribe to TOPIC_LED if not yet ! */
138   if (!client.connected()) {
139     mqtt_mysubscribe((char*) (TOPIC_LED));
140   }
141
142   /* Publish Temperature & Light periodically */
143   payload = "{\"who\":␣\"";
144   payload += whoami;
145   payload += "\",␣\"value\":␣" ;
146   payload += get_temperature();
147   payload += "}";
148
149   payload.toCharArray(data, (payload.length() + 1)); // Convert String payload to a char
        array
150   Serial.println(data);
151   client.publish(TOPIC_TEMP, data);  // publish it
152
153   /* char tempString[8];
154      dtostrf(temperature, 1, 2, tempString);
155      client.publish(TOPIC_TEMP, tempString); */
156
157   payload = "{\"who\":␣\"" + whoami + "\",␣\"value\":␣" + get_light() + "}";
158   payload.toCharArray(data, (payload.length() + 1));
159   Serial.println(data);
160   client.publish(TOPIC_LIGHT, data);
161
162   delay(period);
163   client.loop(); // Process MQTT ... obligatoire une fois par loop()
164 }
```

3.2 node_lucioles_v0.js

```
// Importation des modules
var path = require('path');

// var, const, let :
// https://medium.com/@vincent.bocquet/var-let-const-en-js-quelles-diff%C3%A9rences-b0f14caa2049

const mqtt = require('mqtt')
// Topics MQTT
const TOPIC_LIGHT = 'sensors/light'
const TOPIC_TEMP  = 'sensors/temp'

// MongoDB
var mongodb = require('mongodb');
const mongoName = "lucioles"                    // Nom de la base
const mongoURL = 'mongodb://localhost:27017/'; //URL de connection

// Connection a la DB MongoDB
mongodb.MongoClient.connect(mongoURL, {useNewUrlParser: true}, function(err,  mongodbClient){
    if(err) throw err; // If connection to DB failed ...
                       // else we get a "db" engine reference

    //================================================
    // Get a connection to the DB "lucioles" or create
    //
    var dbo = mongodbClient.db(mongoName);

    //================================================
    // Connection au broker MQTT distant
    //
    const mqtt_url = 'http://192.168.1.100:1883'
    var client_mqtt = mqtt.connect(mqtt_url);

    //================================================
    // Des la connection, le serveur NodeJS s'abonne aux topics MQTT
    //
    client_mqtt.on('connect', function () {
  client_mqtt.subscribe(TOPIC_LIGHT, function (err) {
      if (!err) {
    //client_mqtt.publish(TOPIC_LIGHT, 'Hello mqtt')
    console.log('Server has subscribed to ', TOPIC_LIGHT);
      }
  })
  client_mqtt.subscribe(TOPIC_TEMP, function (err) {
      if (!err) {
    //client_mqtt.publish(TOPIC_TEMP, 'Hello mqtt')
    console.log('Server has subscribed to ', TOPIC_TEMP);
      }
  })
    })

    //================================================================
    // Callback de la reception des messages MQTT pour les topics sur
    // lesquels on s'est inscrit.
    // C'est cette fonction qui alimente la BD.
    //
    client_mqtt.on('message', function (topic, message) {
  console.log("MQTT msg on topic : ", topic.toString());
  console.log("Msg payload : ", message.toString());

  // Parsing du message supposé recu au format JSON
  message = JSON.parse(message);
  wh = message.who
  val = message.value

  // Debug : Gerer une liste de who pour savoir qui utilise le node server
  let wholist = []
  var index = wholist.findIndex(x => x.who==wh)
  if (index === -1){
      wholist.push({who:wh});
  }
  console.log("wholist using the node server :", wholist);
```

```
72
73    // Mise en forme de la donnee à stocker => dictionnaire
74    var frTime = new Date().toLocaleString("fr-FR", {timeZone: "Europe/Paris"});
75    var new_entry = { date: frTime, // timestamp the value
76          who: wh,       // identify ESP who provide
77          value: val     // this value
78        };
79
80    // On recupere le nom du topic du message
81    var topicname = path.parse(topic.toString()).base;
82
83    // Stocker la donnee/value contenue dans le message en
84    // utilisant le nom du topic comme key dans la BD
85    key = topicname
86    dbo.collection(key).insertOne(new_entry, function(err, res) {
87        if (err) throw err;
88        console.log("Item inserted in db in collection :", key);
89        console.log(new_entry);
90    });
91
92    // Debug : voir les collections de la DB
93    dbo.listCollections().toArray(function(err, collInfos) {
94        // collInfos is an array of collection info objects that look like:
95        // { name: 'test', options: {} }
96        console.log("List of collections currently in DB: ", collInfos);
97    });
98      }) // end of 'message' callback installation
99
100     //=============================================================
101     // Fermeture de la connexion avec la DB lorsque le NodeJS se termine.
102     //
103     process.on('exit', (code) => {
104   if (mongodbClient && mongodbClient.isConnected()) {
105       console.log('mongodb connection is going to be closed ! ');
106             mongodbClient.close();
107   }
108     })
109
110 });// end of MongoClient.connect
```

3.3 ui_lucioles.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <script src="https://code.highcharts.com/highcharts.js"></script>
    <script src="https://code.highcharts.com/modules/exporting.js"></script>
    <script src="https://code.highcharts.com/modules/export-data.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <script src="ui_lucioles.js"></script>
  </head>
  <body>
    <h1 style="max-width: 600px; margin: 0 auto" ; align="center">ESPs of Lucioles</h1>

    <br>
    <div id="container1" style="max-width: 1000px; height: 350px; margin: 0 auto"></div>
    <div id="container2" style="max-width: 1000px; height: 350px; margin: 0 auto"></div>

    <div id="mydiv"></div>
  </body>
</html>
```

3.4 ui_lucioles.js

```
//
// Cote UI de l'application "lucioles"
//
// Auteur : G.MENEZ
// RMQ : Manipulation naive (debutant) de Javascript
//
window.onload = function init() {

    //=== Initialisation des traces/charts de la page html

    // Apply time settings globally
    Highcharts.setOptions({
  global: { // https://stackoverflow.com/questions/13077518/highstock-chart-offsets-dates-for-no-reason
        useUTC: false,
        type: 'spline'
  },
  time: {
      timezone: 'Europe/Paris'
  }
    });

    // cf https://jsfiddle.net/gh/get/library/pure/highcharts/highcharts/tree/master/samples/highcharts/demo/spline-irregular-time/
    var chart1 = new Highcharts.Chart({
        title: {
            text: 'Temperatures'
        },
  subtitle: {
            text: 'Irregular time data in Highcharts JS'
  },
        legend: {
            //title: {
            //    text: 'Temperatures'
            //},
            enabled: true
        },
        credits: false,
        chart: {renderTo: 'container1'},
        xAxis: {
            title: {
                text: 'Heure'
            },
            type: 'datetime'
        },
        yAxis: {
            title: {
                text: 'Temperature (Deg C)'
            }
        },
        series: [{name: 'ESP1', data: []},
     {name: 'ESP2', data: []},
     {name: 'ESP3', data: []},
    ],

  //colors: ['#6CF', '#39F', '#06C', '#036', '#000'],
  colors: ['red', 'green', 'blue'],

        plotOptions: {
            line: {
                dataLabels: {
                    enabled: true
                },
                //color: "red",
                enableMouseTracking: true
            }
        }
    });

    var chart2 = new Highcharts.Chart({
        title: { text: 'Lights' },
```

```
70          legend: {
71              //title: {
72              //    text: 'Lights'
73              //},
74              enabled: true
75          },
76          credits: false,
77          chart: {renderTo: 'container2'},
78          xAxis: {
79              title: {
80                  text: 'Heure'
81              },
82              type: 'datetime'
83          },
84          yAxis: {
85              title: {
86                  text: 'Lumen (Lum)'
87              }
88          },
89      series: [{name: 'ESP1', data: []},
90         {name: 'ESP2', data: []},
91         {name: 'ESP3', data: []}],
92
93      //colors: ['#6CF', '#39F', '#06C', '#036', '#000'],
94      colors: ['red', 'green', 'blue'],
95
96          plotOptions: {
97              line: {
98                  dataLabels: {
99                      enabled: true
100                 },
101                 enableMouseTracking: true
102             }
103         }
104     });
105
106
107     //=== Recuperation dans le Node JS server des samples de l'ESP et
108     //=== Alimentation des charts ====================================
109
110     function get_samples(path_on_node, serie, wh){
111   // path_on_node => help to compose url to get on Js node
112   // serie => for choosing chart/serie on the page
113   // wh => which esp do we want to query data
114
115   //node_url = 'http://localhost:3000'
116   //node_url = 'http://10.9.128.189:3000'
117   node_url = 'http://192.168.1.102:3000'
118
119   //https://openclassrooms.com/fr/courses/1567926-un-site-web-dynamique-avec-jquery/1569648-le-
          fonctionnement-de-ajax
120         $.ajax({
121             url: node_url.concat(path_on_node), // URL to "GET" : /esp/temp ou /esp/light
122             type: 'GET',
123             headers: { Accept: "application/json", },
124         data: {"who": wh}, // parameter of the GET request
125             success: function (resultat, statut) { // Anonymous function on success
126                 let listeData = [];
127                 resultat.forEach(function (element) {
128         listeData.push([Date.parse(element.date),element.value]);
129         //listeData.push([Date.now(),element.value]);
130                 });
131                 serie.setData(listeData); //serie.redraw();
132             },
133             error: function (resultat, statut, erreur) {
134             },
135             complete: function (resultat, statut) {
136             }
137         });
138     }
139
140     //=== Installation de la periodicite des requetes GET=============
141
```

```
142     function process_esp(which_esps,i){
143   const refreshT = 100000 // Refresh period for chart
144   esp = which_esps[i];    // L'ESP "a dessiner"
145   //console.log(esp) // cf console du navigateur
146
147   // Gestion de la temperature
148   // premier appel pour eviter de devoir attendre RefreshT
149   get_samples('/esp/temp', chart1.series[i], esp);
150   //calls a function or evaluates an expression at specified
151   //intervals (in milliseconds).
152   window.setInterval(get_samples,
153           refreshT,
154           '/esp/temp',     // param 1 for get_samples()
155           chart1.series[i],// param 2 for get_samples()
156           esp);            // param 3 for get_samples()
157
158   // Gestion de la lumiere
159   get_samples('/esp/light', chart2.series[i], esp);
160   window.setInterval(get_samples,
161           refreshT,
162           '/esp/light',     // URL to GET
163           chart2.series[i], // Serie to fill
164           esp);             // ESP targeted
165     }
166
167
168     //=== Gestion de la flotte d'ESP ================================
169
170     var which_esps = ["1761716416",
171           "80:7D:3A:FD:C9:44",
172           "80:7D:3A:FD:E8:E8"]
173     for (var i = 0; i < which_esps.length; i++) {
174   process_esp(which_esps, i)
175     }
176 };
```

3.5 node_lucioles_v2.js

```
// Importation des modules
var path = require('path');

// var, const, let :
// https://medium.com/@vincent.bocquet/var-let-const-en-js-quelles-diff%C3%A9rences-b0f14caa2049

const mqtt = require('mqtt')
// Topics MQTT
const TOPIC_LIGHT = 'sensors/light'
const TOPIC_TEMP  = 'sensors/temp'

// express
const express = require('express');
const bodyParser = require('body-parser');

const app = express();
//Pour permettre de parcourir les body des requetes
app.use(bodyParser.urlencoded({ extended: true }))
app.use(bodyParser.json())
app.use(express.static(path.join(__dirname, '/')));
app.use(function(request, response, next) { //Pour eviter les problemes de CORS/REST
    response.header("Access-Control-Allow-Origin", "*");
    response.header("Access-Control-Allow-Headers", "*");
    response.header("Access-Control-Allow-Methods", "POST, GET, OPTIONS, PUT, DELETE");
    next();
});

// MongoDB
var mongodb = require('mongodb');
const mongoBaseName = "lucioles"                    // Nom de la base
//const uri = 'mongodb://localhost:27017/'; //URL de connection
//const uri = 'mongodb://10.9.128.189:27017/'; //URL de connection
const uri = "mongodb+srv://menez:mettrelevotre@cluster0-x0zyf.mongodb.net/test?retryWrites=true&w=
    majority";

const MongoClient = require('mongodb').MongoClient;
const client = new MongoClient(uri, { useNewUrlParser: true });

// Connection a la DB MongoDB
client.connect(function(err,  mongodbClient){
    if(err) throw err; // If connection to DB failed ...
                       // else we get a "db" engine reference

    //=============================================
    // Get a connection to the DB "lucioles" or create
    //
    var dbo = client.db(mongoBaseName);

    dbo.dropCollection("temp", function(err, delOK) {
  if (err) throw err;
  if (delOK) console.log("Collection deleted");
    });

    dbo.dropCollection("light", function(err, delOK) {
  if (err) throw err;
  if (delOK) console.log("Collection deleted");
    });

    //=============================================
    // Connection au broker MQTT distant
    //
    //const mqtt_url = 'http://192.168.1.100:1883' ///134.59.131.45:1883'
    const mqtt_url = 'http://broker.hivemq.com'
    var client_mqtt = mqtt.connect(mqtt_url);

    //=============================================
    // Des la connection, le serveur NodeJS s'abonne aux topics MQTT
    //
    client_mqtt.on('connect', function () {
  client_mqtt.subscribe(TOPIC_LIGHT, function (err) {
      if (!err) {
```

```
71      //client_mqtt.publish(TOPIC_LIGHT, 'Hello mqtt')
72      console.log('Node Server has subscribed to ', TOPIC_LIGHT);
73        }
74    })
75    client_mqtt.subscribe(TOPIC_TEMP, function (err) {
76        if (!err) {
77      //client_mqtt.publish(TOPIC_TEMP, 'Hello mqtt')
78      console.log('Node Server has subscribed to ', TOPIC_TEMP);
79        }
80    })
81      })
82
83      //================================================================
84      // Callback de la reception des messages MQTT pour les topics sur
85      // lesquels on s'est inscrit.
86      // C'est cette fonction qui alimente la BD.
87      //
88      client_mqtt.on('message', function (topic, message) {
89    console.log("MQTT msg on topic : ", topic.toString());
90    console.log("Msg payload : ", message.toString());
91
92    // Parsing du message supposé recu au format JSON
93    message = JSON.parse(message);
94    wh = message.who
95    val = message.value
96
97    // Debug : Gerer une liste de who pour savoir qui utilise le node server
98    let wholist = []
99    var index = wholist.findIndex(x => x.who==wh)
100   if (index === -1){
101       wholist.push({who:wh});
102   }
103   console.log("wholist using the node server :", wholist);
104
105   // Mise en forme de la donnee à stocker => dictionnaire
106   // Le format de la date est iomportant => copatible avec le
107   // parsing qui sera realise par hightcharts dans l'UI
108   // cf https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_tolocalestring_date_all
109   // vs https://jsfiddle.net/BlackLabel/tgahn7yv
110   //var frTime = new Date().toLocaleString("fr-FR", {timeZone: "Europe/Paris"});
111   var frTime = new Date().toLocaleString("sv-SE", {timeZone: "Europe/Paris"});
112   var new_entry = { date: frTime, // timestamp the value
113         who: wh,       // identify ESP who provide
114         value: val     // this value
115       };
116
117   // On recupere le nom du topic du message
118   var topicname = path.parse(topic.toString()).base;
119
120   // Stocker la donnee/value contenue dans le message en
121   // utilisant le nom du topic comme key dans la BD
122   key = topicname
123   dbo.collection(key).insertOne(new_entry, function(err, res) {
124       if (err) throw err;
125       console.log("Item inserted in db in collection :", key);
126       console.log(new_entry);
127   });
128
129   // Debug : voir les collections de la DB
130   dbo.listCollections().toArray(function(err, collInfos) {
131       // collInfos is an array of collection info objects that look like:
132       // { name: 'test', options: {} }
133       console.log("\nList of collections currently in DB: ", collInfos);
134   });
135     }) // end of 'message' callback installation
136
137     //================================================================
138     // Fermeture de la connexion avec la DB lorsque le NodeJS se termine.
139     //
140     process.on('exit', (code) => {
141   if (mongodbClient && mongodbClient.isConnected()) {
142       console.log('mongodb connection is going to be closed ! ');
143             mongodbClient.close();
```

```
144   }
145     })
146
147     //=============================================================
148     //==== REQUETES HTTP reconnues par le Node ====================
149     //=============================================================
150
151     // Accés par le Node a la page HTML affichant les charts
152     app.get('/', function (req, res) {
153   res.sendFile(path.join(__dirname + '/ui_lucioles.html'));
154     });
155
156     // Function for answering GET request on this node server ...
157     // probably from navigator.
158     // The request contains the name of the targeted ESP !
159     //      /esp/temp?who=80%3A7D%3A3A%3AFD%3AC9%3A44
160     // Utilisation de routes dynamiques => meme fonction pour
161     // /esp/temp et /esp/light
162     app.get('/esp/:what', function (req, res) {
163   // cf https://stackabuse.com/get-query-strings-and-parameters-in-express-js/
164   console.log(req.originalUrl);
165
166   wh = req.query.who // get the "who" param from GET request
167                      // => gives the Id of the ESP we look for in the db
168   wa = req.params.what // get the "what" from the GET request : temp or light ?
169
170   console.log("\n------------------------------");
171   console.log("A client/navigator ", req.ip);
172         console.log("sending URL ",  req.originalUrl);
173   console.log("wants to GET ", wa);
174   console.log("values from object ", wh);
175
176   const nb = 200; // Récupération des nb derniers samples
177                   // stockés dans la collection associée a ce
178                   // topic (wa) et a cet ESP (wh)
179   key = wa
180   //dbo.collection(key).find({who:wh}).toArray(function(err,result) {
181   dbo.collection(key).find({who:wh}).sort({_id:-1}).limit(nb).toArray(function(err, result) {
182       if (err) throw err;
183       console.log('get on ', key);
184       console.log(result);
185       res.json(result.reverse()); // This is the response.
186       console.log('end find');
187   });
188   console.log('end app.get');
189     });
190
191 });// end of MongoClient.connect
192
193
194 // L'application est accessible sur le port 3000
195 app.listen(3000, () => {
196     console.log('Server listening on port 3000');
197 });
```