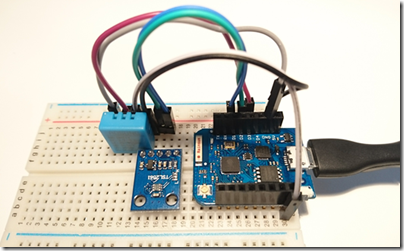
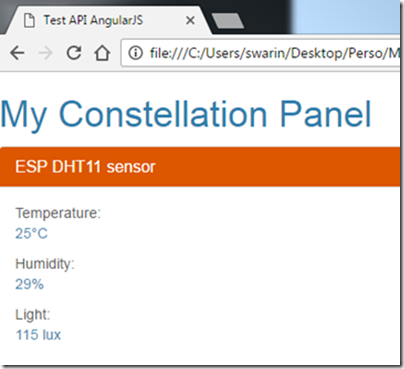
Dans ce tutoriel créons un capteur de température, d’humidité et de luminosité connecté avec une page Web de visualisation des données en temps réel.

[](https://developer.myconstellation.io/wp-content/uploads/2017/05/image-52.png)[](https://developer.myconstellation.io/wp-content/uploads/2017/05/image-55.png)

Prérequis

* Un Arduino connecté ou un ESP8266
* Des capteurs (DHT11 pour l’humidité et la température, un TSL2561 pour la luminosité par exemple)
* Un serveur Constellation

Etape 1 : le montage

Ici nous utiliserons un D1 Mini Pro (ESP8266) avec un capteur DHT11 combinant la mesure de la température et de l’humidité ainsi qu’un capteur TSL2561 pour la luminosité.

Les deux capteurs seront alimentés en 3,3v par la D1 Mini.

Le TSL2561 dispose d’une interface I²C (SDA et SDL qu’on connectera sur D1 et D2) et le DHT11 n’a besoin que d’une entrée digitale (qu’on connectera sur D3).

Etape 2 : la programmation

Pour démarrer vous devez dans Constellation déclarer une sentinelle associée à une clé d’accès et un package virtuel. Ici notre ESP8266 est représenté par la sentinelle nommée “ESP8266” et le package virtuel se nomme “MySensor”.

Dans l’Arduino IDE, nous créons un nouveau projet à partir du Starter Kit Constellation pour ESP8266 dans lequel nous allons définir le nom de notre réseau Wifi (SSID) ainsi que sa clé d’accès  puis nous configurerons le client Constellation en spécifiant l’identité de notre package virtuel, sa clé d’accès et l’adresse/port de notre serveur Constellation :



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | // ESP8266 Wifi  #include <ESP8266WiFi.h>  char ssid[] = "MON SSID";  char password[] = "macléWifi!!!!";    // Constellation client  Constellation<WiFiClient> constellation("X.X.X.X", 8088, "ESP8266", "MySensor", "xxxxxxxxxxxxxxxxx") |

Encore une fois si tout cela est nouveau pour vous, je vous recommande [de suivre ce guide d’introduction à l’API Constellation pour Arduino/ESP8266](https://developer.myconstellation.io/getting-started/connecter-un-arduino-ou-un-esp8266-constellation/).

Lancez maintenant le « gestionnaire de bibliothèque » (menu *Croquis > Inclure une bibliothèque > Gérer les bibliothèques*) et installez les bibliothèques suivantes :

* Adafruit Unified Sensor (par Adafruit)
* DHT Sensor library (par Adafruit)
* Adafruit TSL2561 (par Adafruit)

Une fois les librairies installées, ajoutez en entête de votre code :



|  |  |
| --- | --- |
| 1  2  3 | #include <Adafruit\_TSL2561\_U.h>  #include <DHT\_U.h>  #include <Wire.h> |

Déclarez ensuite, juste après la déclaration du client Constellation, le capteur TSL2561 et le DHT11 par les lignes :

C++



|  |  |
| --- | --- |
| 1  2 | DHT\_Unified dht(D3, DHT11);  Adafruit\_TSL2561\_Unified tsl(TSL2561\_ADDR\_FLOAT); |

Déclarez également un entier pour définir l’intervalle de temps (en ms) entre deux mesures que nous fixons à 10 secondes par défaut :

C++



|  |  |
| --- | --- |
| 1 | int interval = 10000; //ms |

Dans la méthode « *setup()* » configurez les I/O utilisés par l’I²C (D1 et D2 dans notre cas) :

C++



|  |  |
| --- | --- |
| 1 | Wire.begin(D1, D2); |

Initialisez ensuite le capteur DHT11 par la ligne :

C++



|  |  |
| --- | --- |
| 1 | dht.begin(); |

Puis le capteur TSL2561 :

C++



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | if(!tsl.begin())  {    Serial.print("No TSL2561 detected!");    while(1);  }  tsl.enableAutoRange(true);  tsl.setIntegrationTime(TSL2561\_INTEGRATIONTIME\_13MS); |

Toujours dans la méthode «*setup* », récupérons les settings de notre package sur Constellation et affectons le setting « Interval » si défini sur Constellation (autrement c’est la valeur de 10 secondes déclarée ci-dessus qui sera utilisée) :

C++



|  |  |
| --- | --- |
| 1  2  3  4 | JsonObject& settings = constellation.getSettings();  if(settings.containsKey("Interval")) {    interval = settings["Interval"].as<int>();  } |

Pour finir dans la boucle principale, nous allons prendre les mesures de nos capteurs et publier trois StateObjects avec une durée de vie de deux fois l’intervalle de temps :

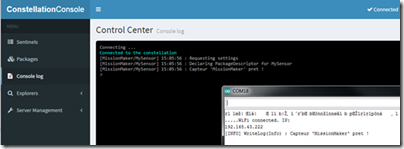
C++



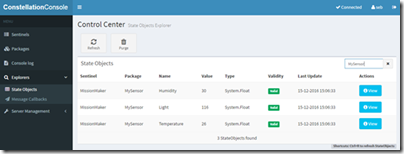
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | void loop(void) {    constellation.loop();    static int lastBeat = 0;    if(millis() - lastBeat > interval) {      lastBeat = millis();      sensors\_event\_t event;      // Luminosité      tsl.getEvent(&event);      if (event.light) {        constellation.pushStateObject("Light", event.light, interval \* 2);      }      // Temperature      dht.temperature().getEvent(&event);      if (!isnan(event.temperature)) {        constellation.pushStateObject("Temperature", event.temperature, interval \* 2);      }      // Humidité      dht.humidity().getEvent(&event);      if (!isnan(event.relative\_humidity)) {        constellation.pushStateObject("Humidity", event.relative\_humidity, interval \* 2);      }    }  } |

**Et voilà votre capteur connecté est prêt !**

Il ne reste plus qu’à téléverser votre programme et ouvrir votre Console Constellation pour constater qu’il est bien opérationnel :

[](https://developer.myconstellation.io/wp-content/uploads/2017/05/image-53.png)

Ouvrez maintenant le [StateObject Explorer](https://developer.myconstellation.io/constellation-platform/constellation-console/stateobjects-explorer/) et rechercher le terme “MySensor” :

[](https://developer.myconstellation.io/wp-content/uploads/2017/05/image-54.png)

On retrouve bien, en temps réel, les trois mesures réalisées par notre ESP8266. N’hésitez pas à ouvrir ces StateObjects et cliquez sur « Subscribe » pour voir les mesures en temps réel dans la Console Constellation.

Etape 3 : Une page Web pour afficher votre capteur en temps réel

Créons maintenant une page Web pour afficher en temps réel les mesures de notre capteur connecté. Pour cela créez un fichier HTML sur votre poste avec une structure HTML de base.

Dans les entêtes ajoutez les scripts suivants :



|  |  |
| --- | --- |
| 1  2  3  4  5 | <script type="text/javascript" src="https://code.jquery.com/jquery-2.2.4.min.js"></script>  <script type="text/javascript" src="https://ajax.aspnetcdn.com/ajax/signalr/jquery.signalr-2.2.1.min.js"></script>  <script type="text/javascript" src="https:/cdn.myconstellation.io/js/Constellation-1.8.1.min.js"></script>  <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.7/angular.min.js"></script>  <script type="text/javascript" src="https://cdn.myconstellation.io/js/ngConstellation-1.8.1.min.js"></script> |

Sur la balise <html> ajoutez l’attribut :



|  |  |
| --- | --- |
| 1 | ng-app="MyDemoApp" |

Et sur la balise <body> ajoutez l’attribut :



|  |  |
| --- | --- |
| 1 | ng-controller="MyController" |

Dans le corps de votre page, placez les valeurs vos StateObjects où bon vous semble. Par exemple :



|  |  |
| --- | --- |
| 1  2  3  4  5 | <div>      Temperature: <p>{{Temperature}}°C</p>      Humidity: <p>{{Humidity}}%</p>      Light: <p>{{Light}} lux</p>  </div> |

Pour finir ajoutez le script suivant dans votre page (dans une balise <script>):

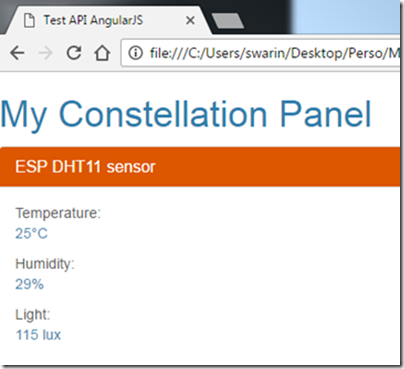
JavaScript



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | var myDemoApp = angular.module('MyDemoApp', ['ngConstellation']);  myDemoApp.controller('MyController', ['$scope', 'constellationConsumer', function ($scope, constellation) {      constellation.initializeClient("http://localhost:8088", "clé d'accès constellation", "MySensor WebPage");        constellation.onConnectionStateChanged(function (change) {        if (change.newState === $.signalR.connectionState.connected) {            constellation.registerStateObjectLink("\*", "MySensor", "\*", "\*", function (so) {            $scope.$apply(function () {             $scope[so.Name] = so.Value;            });          });        }      });        constellation.connect();    }]); |

N’oubliez pas de définir l’adresse de votre Constellation ainsi que la clé d’accès pour vous connecter à votre Constellation lors de l’appel de la méthode « constellation.initializeClient »

**Vous obtenez en quelques minutes une page Web permettant d’afficher en temps réel les mesures de température, d’humidité et de luminosité réalisées par votre ESP8266 !**

[](https://developer.myconstellation.io/wp-content/uploads/2017/05/image-55.png)

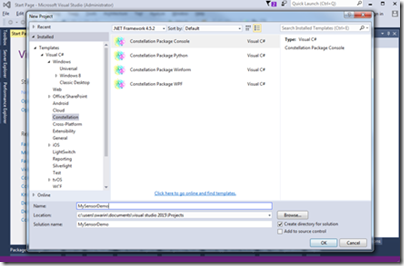
Etape 4 (optionnelle) : créer un package .NET pour exploiter les données du capteur

La page Web réalisée ci-dessus permet d’afficher en temps réel les valeurs de vos StateObjects.

Si maintenant vous souhaitez récupérer la valeur de ces StateObject dans un programme pour réagir en fonction de certaines valeurs (par exemple alerter si T° trop froide ou allumer des lumières si la luminosité est trop faible) ou bien tout simplement enregistrer ces valeurs dans un fichier, une base de données ou un service de cloud, suivez la suite.

Pour cet exemple, réalisons un programme en C#. On part du principe que vous avez installé Visual Studio et [le SDK Constellation depuis le « Web Platform Installer »](https://developer.myconstellation.io/getting-started/installer-constellation/).

Créez alors un nouveau projet de type « Constellation Package Console » :

[](https://developer.myconstellation.io/wp-content/uploads/2017/05/image-56.png)

Dans votre classe, ajoutez simplement des propriétés pour chaque StateObject avec l’attribut « StateObjectLink » pour lier ces propriétés .NET aux StateObjects de votre capteur (servez-vous du snippet « stateobjectlink »).

Par exemple pour inclure le StateObject « Light » dans notre classe C# :

C#



|  |  |
| --- | --- |
| 1  2 | [StateObjectLink("MySensor", "Light")]  public StateObjectNotifier Light { get; set; } |

Vous pouvez maintenant afficher sa valeur à tout moment, par exemple :

C#



|  |  |
| --- | --- |
| 1 | PackageHost.WriteInfo($"Luminosité {this.Light.Value.LastUpdate} = {this.Light.DynamicValue} lux"); |

La propriété « Light » de votre classe contiendra toujours la dernière valeur du StateObject connu dans votre Constellation.

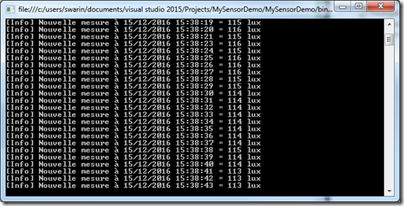
Vous pouvez donc manipuler la valeur mesurée par votre ESP8266 dans votre code C# comme une simple propriété .NET de votre code.

Vous pouvez également vous abonner à chaque mise à jour du StateObject pour déclencher une action grâce à l’évènement « ValueChanged » :

C#



|  |  |
| --- | --- |
| 1  2  3  4 | this.Light.ValueChanged += (s, e) =>  {      PackageHost.WriteInfo($"Nouvelle mesure à {this.Light.Value.LastUpdate} = {this.Light.DynamicValue} lux");  }; |

[](https://developer.myconstellation.io/wp-content/uploads/2017/05/image-57.png)