Franck Dechavanne, Stéphane Lavirotte, Gérald Rocher, Jean-Yves Tigli
Polytech'Nice – Sophia
Projet SI3
2017-2018

Projet SI3
Introduction aux Systèmes Ambiants

1 Simulateur d'un environnement de dispositif connectés

1.1 Installation de Home I/O

Nous allons utiliser l'environnement Home I/O afin de simuler un habitat avec de nombreux capteurs et actionneurs connectés. Au travers ce logiciel, il sera possible d'avoir accès à des données simulées (température, luminosité, présence d'un utilisateur, …), mais aussi d'agir sur l'environnement (allumer une lumière, piloter le chauffage, fermer les stores…).

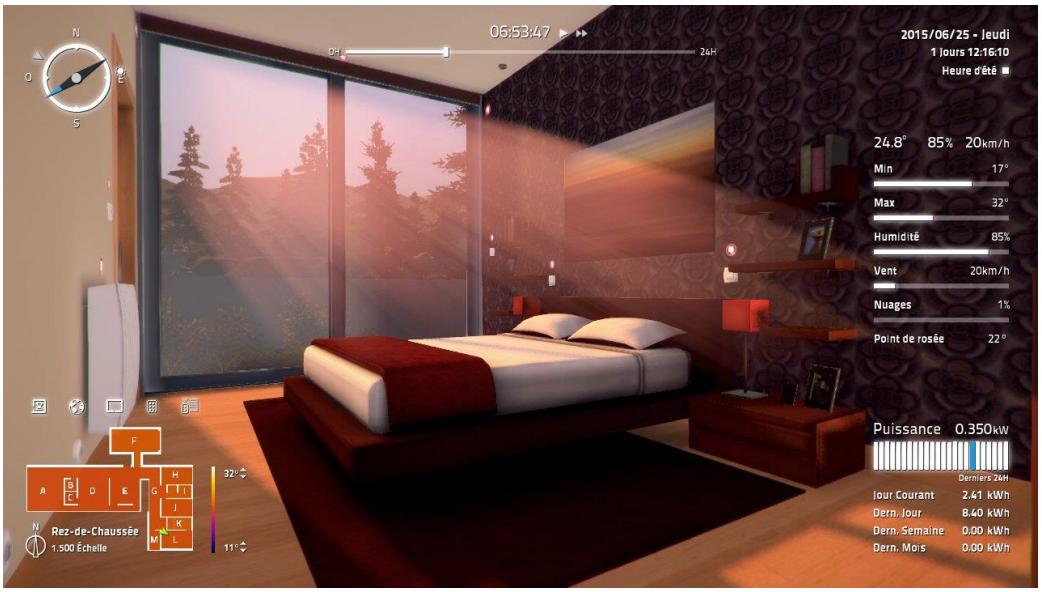
https://teachathomeio.com/

Pour installer l'environnement de simulation, vous pouvez récupérer le logiciel à l'une des adresses suivantes :

https://teachathomeio.com/telecharger-demo/

ou

http://trolen.polytech.unice.fr/cours/projet-si3/homeio-1.5.0-installer.exe



Cette version est une version de démonstration complète du logiciel qui fonctionnera durant 1 mois. Comme votre projet va dépasser cette période, nous avons fait l'acquisition d'une licence de site. Vous téléchargerez cette licence à l'adresse suivante :

http://trolen.polytech.unice.fr/cours/projet-si3/u-sciences.lic

Il vous suffit alors de copier ce fichier (et supprimer le fichier `trial.lic`) dans le dossier suivant :

```
C:\ProgramData\Real Games\Home IO
```

Vous pouvez alors lancer le simulateur. Nous vous rappelons que c'est une licence de site que nous avons achetée donc que vous avez le droit d'utiliser ce logiciel tant que vous êtes étudiants à Polytech Nice Sophia.

1.2 Utilisation de l'environnement de simulation

Pour une prise en main de l'environnement de simulation, nous vous invitons à aller consulter la documentation :

https://realgames.co/docs/homeio/fr/index.html

Vous pourrez en particulier consulter :

- La carte des capteurs et actionneurs : https://realgames.co/docs/homeio/fr/devices-map/index.html
- Le mode de pilotage des dispositifs : https://realgames.co/docs/homeio/fr/device-modes/index.html

Concernant le mode de pilotage des dispositifs, ceux-ci doivent être mis dans le mode « Externe » pour être pilotés par une application tierce (et non pas en cliquant dans le simulateur sur tous les dispositifs).

Vous pouvez cliquer sur tous les petits symboles pour les passer en bleu (mode externe)… mais cela risque de vous prendre la séance et vous risquez d'en oublier un. Le plus simple est donc de sauvegarder la simulation, de vous rendre dans le dossier « Documents/Home IO/Saves » et d'éditer le fichier `xml` contenant la configuration des capteurs et actionneurs. Vous remplacerez alors le mot « Wired » par le mot « External ». Puis, après sauvegarde, vous rechargerez la simulation dans Home I/O.

1.3 SDK

Vous consulterez ensuite la partie interfaçage et programmation avec cet environnement. L'ensemble de la documentation est accessible à l'adresse suivante :

https://realgames.co/docs/homeio/fr/sdk-getting-started/index.html

Vous téléchargerez le SDK fournit avec l'application :

https://realgames.co/downloads/tutorials/homeio/sdk.zip

Le SDK est pour un environnement de programmation C#. Vous avez un exemple simple de code dans la documentation permettant d'accéder aux dispositifs simulés (capteurs et actionneurs). Vous disposez aussi d'exemples dans le SDK, n'hésitez pas à les consulter pour voir comment accéder aux informations ou piloter l'environnement.

De plus, avec le SDK, vous avez le logiciel Engine I/O Explorer qui vous permet de visualiser l'état de la simulation et de contrôler celle-ci. Cette application a été réalisée avec la bibliothèque fournie dans le SDK. Vous noterez que vous avez trois catégories d'informations : les INPUTS (les informations issues de capteurs de la simulation), les OUTPUTS (les actionneurs dont vous pouvez lire l'état et modifier celui-ci), les MEMORIES (contenant les informations globales à la simulation). Pour chacune de ces catégories, vous avez accès aux informations suivant différents types (bit, octet, short, int, long, float, …).

Vous avez donc accès à un simulateur d'espace ambiant et vous disposez d'un SDK qui vous permettra de vous interfacer avec celui-ci. Nous allons maintenant nous intéresser à l'orchestration des flots de données qui vont permettre de mettre en place des logiques applicatives entre ces capteurs et actionneurs.

2 Orchestration des flots de l'Internet des Objets

2.1 Installation de l'environnement Node-RED

Maintenant que vous disposez des capteurs et actionneurs du simulateur, vous allez vous intéresser à l'environnement Node-RED qui nous permettra d'exploiter ces informations.

https://nodered.org/

Node-RED est basé sur Node.js (que vous devez déjà utiliser ou que vous utiliserez pour votre projet). Si ce n'est pas encore le cas, vous pouvez vous rendre à la page suivante :

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

Une fois que vous disposez de l'environnement Node.js sur votre machine, vous pourrez installer Node-RED simplement comme un paquetage supplémentaire. Vous suivrez la documentation disponible à l'adresse suivante :

https://nodered.org/docs/getting-started/installation

Vous ouvrirez alors un interprète de commandes (PowerShell si vous êtes sous Windows) et lancerez la commande `node-red`. L'interface de contrôle de l'environnement est alors accessible sur votre machine à l'adresse suivante :

http://127.0.0.1:1880

2.2 Communications entre les différents Objets

MQTT est un protocole de messagerie utilisé dans l'Internet des Objets. Celui-est est basé sur un modèle *Publisher-Broker-Subscriber*. Un *Publisher* envoie les informations à un *Broker*. Un *Subscriber* s'abonnera à un topic auprès d'un Broker.

2.3 *Publisher-Broker-Subscriber* MQTT en Node-RED

2.3.1 Broker MQTT dans Node-RED

Afin de faciliter la mise en œuvre, vous pouvez lancer un *Broker* dans Node-RED. Mais pour cela vous devez installer un paquetage supplémentaire. Vous pouvez le faire directement en ligne de commande (mais vous devrez relancer votre instance Node-RED)

```
npm install node-red-contrib-mqtt-broker
```

ou bien à partir de l'interface de Node-RED En vous rendant dans le menu de droite puis « Manage Palette / Install ». Vous taperez alors :

```
node-red-contrib-mqtt-broker
```

En instanciant un node « *mosca* », pour en cliquant sur « Déploy » vous lancerez un Broker MQTT qui sera disponible à l'adresse `127.0.0.1:1883`.

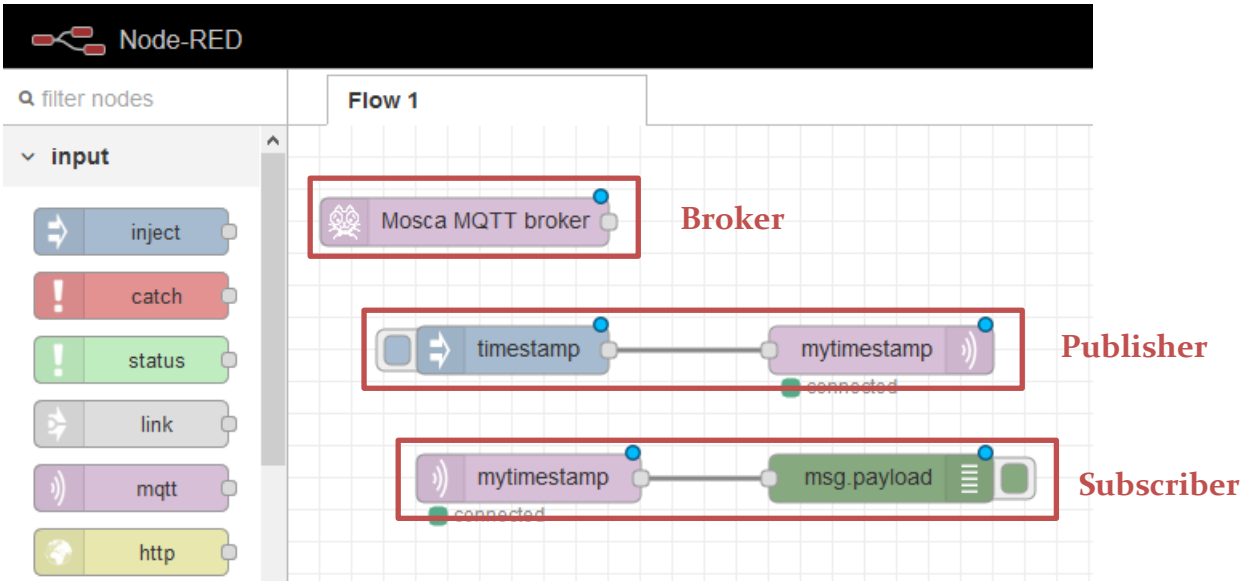
2.3.2 Publisher MQTT dans Node-RED

Publier une information sur un topic MQTT est trivial. Tout est question de nommage du topic (nommage basé sur la notion de chemin comme pour trouver un fichier).

Vous commencerez par mettre un node « *inject* » (qui enverra un timestamp par défaut) dans la catégorie INPUT et un node « *mqtt* » dans la catégorie OUTPUT. Vous relierez les deux nodes en question pour que les informations fournies par le timestamp soient publiées sur un topic de nom « mytimestamp ». La configuration d'un node est réalisé en double cliquant dessus. Vous venez de mettre en place un Publisher MQTT.

2.3.3 Subscriber MQTT dans Node-RED

Pour souscrire à un topic MQTT, il vous suffit d'ajouter un node « *mqtt* » de type INPUT et un node « *debug* » (pour consulter que l'information a bien été envoyée (vous pourrez la visualiser dans la console de debug à droite). Après avoir relié les deux nodes et avoir configuré le topic du node mqtt à la valeur « mytimestamp », à chaque fois que vous cliquerez sur l'injecteur, vous recevrez l'information via MQTT. Vous devriez avoir le schéma suivant.



Pour vous convaincre que ces communications passent bien par le réseau, amusez-vous à lancer uniquement le *Broker* sur un ordinateur, avoir un *Publisher* sur une autre machine, et le *Subscriber* sur une troisième. Répartissez-vous cela dans le groupe de travail.

3 MQTT

3.1 Bibliothèque MQTT

Le dernier problème à résoudre est de mettre en place le protocole MQTT au-dessus des capteurs et actionneurs de l'environnement virtuel. Cela permettra de faciliter l'intégration de ce simulateur avec le reste des développements et en plus cela sera cohérent avec les technologies de l'IoT.

Pour faciliter ce travail, vous utiliserez la bibliothèque MQTT disponible en C# :

https://m2mqtt.wordpress.com/

3.2 Exposition des données du simulateur via le protocole MQTT

Votre travail va donc consister à réaliser un programme en C# qui permette de faire la passerelle entre le simulateur Home I/O et le protocole MQTT. Pour cela, vous analyserez la manière dont les informations sont accessibles dans le simulateur en utilisant la bibliothèque EngineIO.dll (accès par mémoire partagée entre les deux applications). Vous rendez alors ces informations disponibles suivant le protocole MQTT via la mise en place d'un *publisher MQTT*.

Vous allez devoir penser à l'arborescence des topics pour les publications des informations. N'oubliez pas que vous allez avoir à la fois une publication des informations en lecture, mais aussi en écriture (pour modifier les valeurs des actionneurs). A vous de penser et mettre en place les développements adéquats.

Vous pourrez utiliser le Broker MQTT que nous avons mis en place dans Node-RED pour la publication des données pour débugger vos développements. Si vous connectez un node « *debug* » au node « *Mosca MQTT Broker* », vous pouvez voir tous les messages publiés et leurs valeurs.

4 Conclusion et perspectives pour la suite de projet

Vous avez mis en place un simulateur pour un environnement « Smart-Home », et vous pouvez maintenant facilement développer des choses plus complexes pour la mise en œuvre d'un Home Automation. Vous pourrez alors connecter la partie Front-End et Back-End de votre système de gestion des incidents.

Enfin, lors des séances prochaines concernant les systèmes ambiants, nous pourrons remplacer le système de simulation par des capteurs et actionneurs réels en utilisant toujours Node-RED et MQTT.