

Rapport final de projet tuteuré

Déploiement de OM2M pour une application dans le domaine de la e-santé

Réalisé par :

Emma Dupuy

Ruby Faucher

Numa Goetschel

Marie Nourrisson

Tuteurs école

Réjane Dalce

Elyes Lamine

Tuteur entreprise

Samir Medjah

Année universitaire 2021 – 2022

Remerciements

Dans le cadre de ce projet tuteuré, nous tenons à remercier toutes les personnes ayant contribué à la réalisation et au bon déroulement de notre projet.

Tout d'abord, nous aimerions adresser nos remerciements au LAAS, laboratoire d'analyse et d'architecture des systèmes, ainsi qu'au CHU de Toulouse et plus particulièrement au Docteur Nada Drira pour avoir proposé ce projet et à Monsieur Samir Medjiah pour nous avoir accompagné tout au long de celui-ci.

Nous remercions également Monsieur Lamine, Madame Dalce et Monsieur Carayol pour l'aide et l'encadrement apporté au cours de ce projet, ainsi que pour leurs disponibilités.

Sommaire

Remerciements	1
Sommaire	2
Résumé	4
Abstract	4
Glossaire	5
Introduction	8
1. Éléments de contexte et définition du sujet	9
1.1. Projet Eclipse OM2M	10
1.2. Maladie Kératocône	14
2. Nouveaux scénarios d'utilisation	15
2.1. Montre connectée :	15
2.2. Capteurs environnementaux :	16
3. Spécifications de la solution	16
3.1. Spécifications fonctionnelles	16
3.1.1. Diagrammes de cas d'utilisation	16
3.1.2. User stories	17
Montre connectée	17
Capteurs environnementaux	17
3.1.3. Architecture du système	17
4. Environnement technique de la solution	18
4.1. Logiciels utilisés	18
4.1.1. Eclipse	18
4.1.2. PostMan	18
4.1.3. Fitbit Studio	18
4.2 Technologies utilisées	21
4.2.1. Montre connectée Fitbit Versa 2	21
4.2.2. Xiaomi Mi Band 6	22
4.2.3. Capteurs environnementaux :	24
4.3 Tests, validation et pistes d'amélioration	28
5. Gestion du projet et de l'équipe	29

5.1. Outils de communication et de partage	29
5.2. Gestion des tâches	30
Conclusion et perspectives	33
Webographie	34
OM2M	34
Kératocône	34
Fitbit	34
GadgetBridge	35
Annexes	35
Annexe 1 : Les différents code utilisé tout au long de ce projet sont intégré au sein du dépôt Github	35
Annexe 2 : CookBook	35
Table des matières	37
Installation de OM2M	38
1. Prérequis	38
2. Installation d'Eclipse	38
3. Cloner le projet	39
4. Importer le projet sur Eclipse	39
5. Installation du Plugin Tycho	40
6. Lancer un build du projet	43
6.1 Avec Eclipse	43
6.2 En ligne de commande	43
7. Tester OM2M	43
8. Ajout d'un plugin Bluetooth	44
Table des figures	48

Résumé

Ce rapport présente le déploiement du logiciel open-source OM2M, développé par le LAAS, au sein de la plateforme de recherche et d'innovation, le Connected Health Lab (CHL) situé à l'école d'ingénieurs en informatique et systèmes d'informations ISIS de Castres. Cette implémentation permet l'interconnexion de différents objets connectés afin de les faire communiquer malgré les différences de langages et de protocoles utilisés.

Cette étude est complétée par le développement de trois objets connectés permettant le suivi des patients atteints de maladie oculaire, le Kératocône. Ces objets connectés permettent la mesure des mouvements dès lors que le patient se frotte les yeux et l'analyse de l'environnement, avec deux montres connectées, FitBit et Xiaomi mi band 6 et une carte Raspberry pi 3 composée de capteurs environnementaux.

Abstract

This paper relates the advancement of the deployment of the open-source OM2M software, developed by the LAAS, within the research and innovation platform the Connected Health Lab (CHL) located in the engineering school in computer science and information systems (ISIS) in Castres. This implementation allows the interconnection of different connected objects and allows them to communicate despite their different languages or protocols.

This study is completed by the development of three connected objects allowing the monitoring of patients with eye diseases, Keratoconus. These connected objects allow the measurement of movements as soon as the patient rubs his eyes and the analysis of the environment, with respectively two connected watches Fitbit and Xiaomi mi band 6 and a Raspberry pi 3 sensor.

Glossaire

OM2M : Le logiciel Eclipse OM2M permet de déployer et de mettre en œuvre le standard oneM2M.

Android : Android est un système d'exploitation mobile fondé sur le noyau Linux et développé par Google.

API Restful : Representational State Transfer Application Program Interface

Bluetooth : Bluetooth est une norme de télécommunications permettant l'échange bidirectionnel de données à courte distance en utilisant des ondes radio UHF sur la bande de fréquence de 2,4 GHz.

CHL : Le Connected Health Lab est une plateforme unique dédiée à la recherche et l'innovation directement implantée à ISIS Castres. Ce « living lab » est mis à disposition de tous les acteurs de la e-santé afin d'y mener des projets divers autour de la notion de parcours patient.

CNRS : Centre National de la Recherche Scientifique

CookBook : Un cookbook est un document contenant essentiellement une collection de scénarios et de situations de bonnes pratiques en termes de codage et de création de structure.

Eclipse : Eclipse est un projet, décliné et organisé en un ensemble de sous-projets de développements logiciels, de la fondation Eclipse visant à développer un environnement de production de logiciels libres qui soit extensible, universel et polyvalent, en s'appuyant principalement sur Java.

Ethernet : Ethernet est un protocole de réseau local à commutation de paquets. C'est une norme internationale : ISO/IEC 802-3.

GadgetBridge : GadgetBridge est un outil utile qui permet de synchroniser les alertes, les rappels et autres données d'objets connectés. Ces transferts d'informations sans fil permettent de sauvegarder rapidement et facilement les données sur un stockage externe.

IOS : iOS, anciennement iPhone OS, le « i » d'iOS étant pour iPhone d'où la minuscule, est le système d'exploitation mobile développé par Apple pour plusieurs de ses appareils.

Javascript : JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives et à ce titre est une partie essentielle des applications web. Avec les technologies HTML et CSS, JavaScript est parfois considéré comme l'une des technologies cœur du World Wide Web.

JSON : JavaScript Object Notation est un format de données textuelles dérivé de la notation des objets du langage JavaScript.

Kératocône : Vient du grec “kerato” qui veut dire « cornée », et “konos” pour « cône », est une maladie oculaire provoquant une déformation de la cornée.

LAAS : Laboratoire d’analyses et d’architecture des systèmes.

LoRaWAN : LoRaWAN est un protocole de télécommunication permettant la communication à bas débit, par radio, d’objets à faible consommation électrique.

LPWAN : Un réseau étendu à basse consommation (Low Power Wide Area Network ou LPWAN) est un type de réseau employé dans l’Internet des objets (Internet of Things ou IoT) et dans la communication inter machine (Machine to Machine ou M2M).

LTE-M : LTE-M (Long-Term Evolution for Machines) est une technologie qui a été conçue spécifiquement pour l’Internet des objets. Elle s’est depuis imposée comme l’une des solutions LPWAN les plus adaptées pour déployer des systèmes IoT nécessitant une basse consommation tout en couvrant de larges zones géographiques.

NB-IoT : Le Narrowband IoT désigne un standard de communication normalisé en 2016 et dédié à l’Internet des objets. Il s’agit plus concrètement d’un standard de communication LPWAN dont la principale mission consiste à faciliter la communication d’importants volumes de données sur une très grande distance.

oneM2M : oneM2M est un projet de partenariat mondial fondé en 2012 et constitué de 8 des principales organisations mondiales de développement de normes TIC (technologies de l’information et de la communication). L’objectif de l’organisation est de créer une norme technique mondiale d’interopérabilité concernant l’architecture, les spécifications d’API, les solutions de sécurité et d’inscription pour les technologies Machine-to-Machine et IoT sur la base des exigences fournies par ses membres.

Protocole HTTP : Hypertext Transfer Protocol (HTTP, littéralement « protocole de transfert hypertexte ») est un protocole de communication client-serveur.

Rollup.js : Rollup est un bundler JavaScript. C’est-à-dire qu’il lit votre code et *bundle* l’ensemble des modules importés (via import ou require) en un fichier unique. Il est capable d’exporter ce module dans les principaux formats de modules (CJS, ESM, AMD, IIFE) correspondant à tous les cas d’usage.

SDK : Un kit de développement logiciel, *Software Development Kit* (SDK) ou devkit en anglais, aussi appelé trousse de développement logiciel, est un ensemble d’outils logiciels destinés aux développeurs, facilitant le développement d’un logiciel sur une plateforme donnée (par exemple, iOS, Android, BlackBerry 10, Symbian, Bada, Linux, OS X, Microsoft Windows).

Sigfox : Sigfox est un opérateur de télécommunications français créé en 2009 et implanté à Labège. C’est un opérateur télécom de l’Internet des objets.

Typescript : TypeScript est un langage de programmation libre et open source développé par Microsoft qui a pour but d’améliorer et de sécuriser la production de code JavaScript.

XML : Extensible Markup Language, « langage de balisage extensible » en français, est un méta langage informatique de balisage générique.

Risque de balkanisation : En l'absence protocoles et de standards l'Internet des objets se réduirait à un patchwork de réseaux propriétaires et incompatibles, chacun dédié soit à une application particulière, soit à un groupe d'utilisateurs donnés.

Cornéoplastie : Le terme « cornéoplastie » regroupe l'ensemble des techniques chirurgicales destinées à améliorer le profil de la cornée.

Type MIME : Le type Multipurpose Internet Mail Extensions est un standard permettant d'indiquer la nature et le format d'un document. Il est défini au sein de la RFC 6838.

Origine : L'origine d'une application web est définie par le schéma (protocole), l'hôte (domaine) et le port de l'URL utilisée pour y accéder. Deux objets ont la même origine seulement quand le schéma, l'hôte et le port correspondent.

Carte SD : Une carte SD (« SD » étant le sigle de l'expression anglaise « Secure Digital ») est une carte mémoire amovible de stockage de données numérique.

Introduction

Ce projet tuteuré a été proposé par le LAAS, laboratoire d'analyse et d'architecture des systèmes.

Créé en 1968, le LAAS est une unité propre de recherche du CNRS (centre national de la recherche scientifique). Il regroupe 650 personnes, dont 200 chercheurs et enseignants-chercheurs, 275 doctorants, 45 post-doctorants, et plus de 110 ingénieurs, techniciens et personnels administratifs. Il mène des recherches en informatique, robotique, automatique et micro et nano systèmes.

Le projet vise à étendre les domaines d'application de la plateforme de services pour les objets connectés Eclipse OM2M développée par l'équipe SARA au sein du LAAS.

La plateforme Eclipse OM2M assure l'interconnexion entre des objets connectés hétérogènes par leurs technologie/protocole de communication mais aussi par leur format et modèle de données.

L'objectif de ce projet est de déployer la plateforme dans un scénario de e-santé en connectant de nouveaux capteurs/actionneurs propres à la e-Santé ainsi que le développement d'une application « exemple ».

Pour ce projet tuteuré, les attendus sont doubles. Il faut d'un côté, en fonction des capteurs/actionneurs identifiés, augmenter la plateforme Eclipse OM2M en développant des modules logiciels (en Java) « Interworking Proxy » correspondants aux technologies de communication propres à ces capteurs/actionneurs (Bluetooth, ANT+, ...).

L'intérêt serait de pouvoir déployer la plateforme au sein du Connected Health Lab (CHL), un laboratoire dédié à la recherche et à l'innovation implanté au sein de l'école d'ingénieurs en informatique et système d'information pour la santé ISIS de Castres. Ce dernier est mis à disposition des acteurs de la e-santé afin d'y mener des projets divers autour de la notion du parcours patient. L'apport de cette plateforme serait ainsi de pouvoir permettre l'interopérabilité entre tous les types d'objets connectés permettant de faciliter le suivi du parcours de santé du patient et ainsi d'interconnecter les différents moyens de communication de ces objets.

D'un autre côté, pour mettre en application ce mode de communication, le développement d'un prototype d'une application métier simple pour valider le déploiement et l'intégration effective des nouveaux capteurs/actionneurs est attendu. Le docteur Drira, médecin interne en ophtalmologie au CHU de Toulouse prend ainsi part à ce projet en recherchant une solution qui permettrait d'amener un suivi patient dans le traitement d'une maladie oculaire, la Kératocône.

1. Éléments de contexte et définition du sujet

Une ressource web est un contenu que l'on peut retrouver en ligne et que nous pouvons regarder, lire, écouter ou avec lequel on peut interagir, tels que des vidéos ou des fichiers en ligne.

Pour pouvoir récupérer, publier ou manipuler ce type de données entre un client périphérique et un serveur nous pouvons utiliser des services Web tels que les API RestFul utilisant des méthodes HTTP (figure 1).

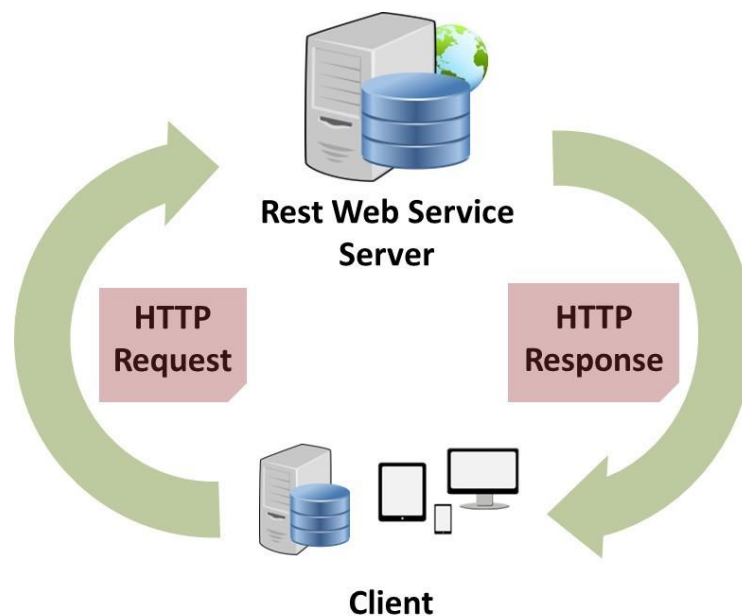


Figure 1 : Architecture d'une API Rest

En utilisant le protocole HTTP, les API Rest permettent aux logiciels d'un appareil de communiquer avec le même appareil ou avec un autre même si ces derniers utilisent des systèmes d'exploitation et des architectures différents. Le client va ainsi pouvoir demander des ressources avec un langage que le serveur comprend et le serveur va lui répondre avec un langage que le client accepte. La réponse est très souvent renvoyée au format JSON, XML ou tout simplement texte.

Cette API est basée sur le principe d'architecture client-serveur. Les clients vont utiliser des appels HTTP pour demander, envoyer ou bien modifier une ressource.

Les principales méthodes que l'on retrouve sont :

- GET : Permet d'appeler une ressource
- POST : Permet d'envoyer une ressource
- PUT : Permet de modifier une ressource existante

- DELETE : Permet de supprimer une ressource
- HEAD : Permet d'interroger l'en tête de la réponse
- OPTION : Permet de décrire les options de communications pour la ressource cible

Les ressources Web sont constamment utilisées et de plus en plus développées dans le domaine de la santé permettant d'améliorer le suivi du parcours de soin et de santé des patients. La prise en main des API de type Rest sont donc très utilisées pour faire le lien entre les objets connectés qui permettent de suivre ce dernier.

De plus, l'écosystème de l'Internet des Objets est vaste et peuplé de beaucoup de types de standards et protocoles différents, ils sont donc souvent difficile de faire interagir tous les types de langage et de communication entre eux. En effet, les objets peuvent autant s'appuyer sur des réseaux locaux tels que les réseaux Wi-Fi, LoRaWAN privé, Ethernet ou encore Bluetooth, que sur des réseaux publics opérés à grande échelle tels que les réseaux LPWAN basés sur les technologies Sigfox ou LoRaWAN, ou les réseaux basés sur des bandes de fréquences licenciées (2/3/4/5G), avec des déclinaisons LPWAN cellulaires comme LTE-M et NB-IoT. Par ailleurs, contrairement à Internet dont le succès repose sur l'adoption généralisée de protocoles de communication clairement définis (TCP/IP, SMTP, HTTP, etc.), l'Internet des objets présente un risque de balkanisation en l'absence de protocoles et standards universels définis au préalable. En effet, l'ensemble de ces différents moyens de communications et de ces différents standards et protocoles implique que les objets connectés ne sont pas toujours tous compatibles et ne peuvent parfois pas échanger directement des informations entre eux.

1.1. Projet Eclipse OM2M

Le projet Eclipse OM2M, initié par le LAAS-CNRS, est une implémentation open source du standard oneM2M. Il fournit une API RESTful pour la création et la gestion des ressources M2M.

OM2M fournit une plateforme de services open source pour l'interopérabilité M2M.

Le M2M ou machine to machine permet des communications entre machines différentes sans intervention humaine directe. Le M2M permet l'automatisation des tâches et de déléster la charge de travail humaine d'un nombre d'actions et de vérifications redondantes et répétitives, tout en diminuant le nombre d'erreurs.

Le standard oneM2M est un projet de partenariat mondial fondé en 2012 et constitué par 8 des principales organisations mondiales de développement de normes TIC : ARIB (Japon), ATIS (États-Unis), CCSA (Chine), ETSI (Europe), TTA (Corée), TTC (Japon), TSDSI (Inde), TTA (Corée) et TTC (Japon). L'objectif de l'organisation est de créer des exigences, une architecture, des spécifications API, des solutions de sécurité et une interopérabilité pour les technologies Machine-to-Machine et IoT.

OneM2M est un standard horizontal, c'est -à -dire qu'il se positionne de manière transverse peu importe le domaine d'application. Il est capable d'avoir une vision homogène du système et ce, quel que soit le domaine d'application (Maison connectée, e-santé, gestion de la ville...). Et cela grâce à une couche service accessible à partir de l'API (figure 2).

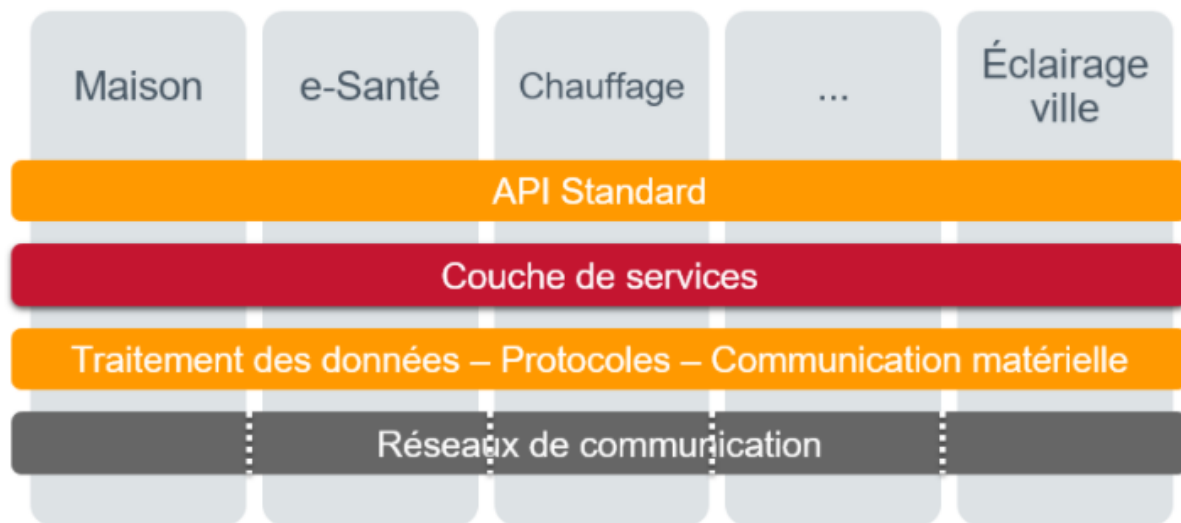


Figure 2 : Standard horizontal

La structure du système est modélisée par différentes entités. On retrouve, au niveau applicatif, les AE (Application Entities) qui permettent de représenter les différentes applications connectées au système. On retrouve ensuite la couche service représentée par les CSE (Common Services Entities) qui permettent aux applications de s'enregistrer dans leur système et de fournir les différents services liés au standard. Finalement, on retrouve la dernière couche, la plus basse, qui est la couche liée au transport des données sur les réseaux de communications qui va échanger avec la couche service (figure 3).

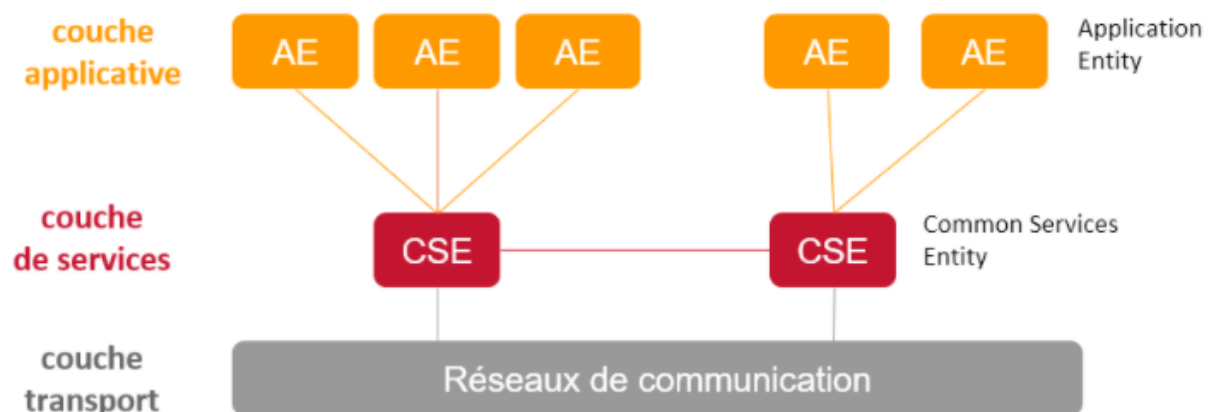


Figure 3 : Structure système OM2M

En pratique, dans le cas du domaine de la e-santé pour le suivi d'un patient, on va retrouver différents objets connectés qui vont être connectés à une passerelle qui va permettre l'échange dans les deux sens avec le serveur. Les protocoles utilisés vont dépendre des technologies et la passerelle va pouvoir permettre de communiquer avec le serveur selon le protocole HTTP. Sur le serveur va s'exécuter une application qui va permettre au patient de contrôler les états des objets connectés et d'agir dessus. Grâce à OM2M, on va retrouver un ensemble d'applications qui vont représenter les objets connectés, la passerelle et la partie applicative sur le serveur. Les entités s'enregistrent sur le CSE, précédemment présenté, et vont pouvoir communiquer entre elles. Les applications vont s'appeler ADN (Application Dedicated Node), la passerelle MN (Middle Node) et enfin, le serveur IN (Infrastructure NODE) (figure 4).

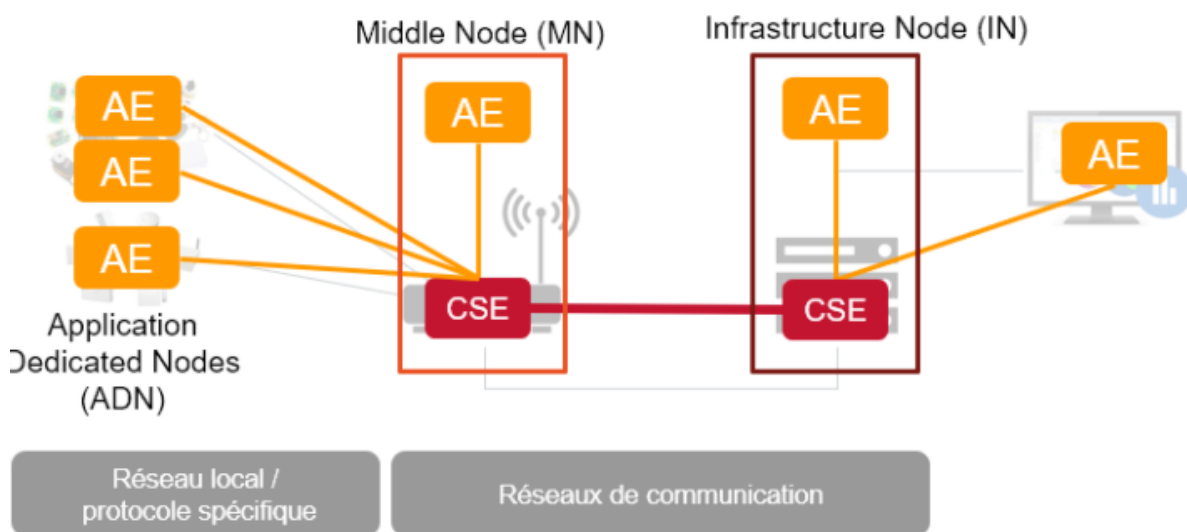


Figure 4 : Architecture globale OM2M

Pour conclure, OM2M permet, qu'importe le cas d'utilisation, une vision identique. Les AE seront enregistrées sur les CSE et le mode de communication restera toujours identique.

Maintenant que nous avons décrit la structure générale du système, nous allons étudier l'architecture en profondeur des différents nœuds, aussi bien de la passerelle (MN) que du serveur (IN) où se stockent les données. La passerelle et le serveur sont constitués de couches service, comme vu précédemment, qui sont-elles mêmes constituées d'applications et de terminal service, ce terminal permet à un utilisateur d'accéder à des applications et des données sur un ordinateur distant, via n'importe quel type de réseau. Concernant les applications, elles sont, pour le cas de la passerelle, constituées de deux types de conteneurs. Un conteneur est une structure de données, autrement dit, les conteneurs sont utilisés pour stocker des objets sous une forme organisée qui suit des règles d'accès spécifiques. On retrouve un conteneur descripteur qui permet de spécifier la structure des données et un conteneur data qui stockera la donnée.

On se retrouve ainsi avec une hiérarchie présentée ci-contre (figure 5) :

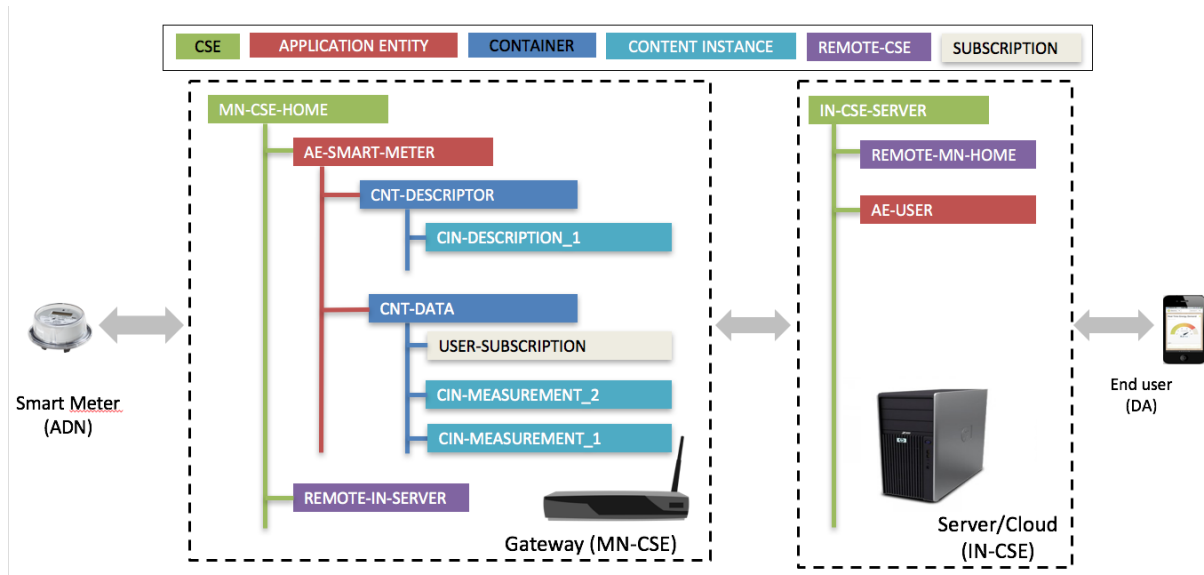


Figure 5 : Architecture détaillée OM2M

Pour ensuite requêter ce contenu, le protocole HTTP est utilisé. Il suffira d'appeler le localhost du serveur sur lequel OM2M se trouve. Le contenu de la requête sera accepté aussi bien en JSON qu'en XML.

La requête doit avoir un en-tête composé de deux variables, qui doivent être définies. Le premier est le content-type, permettant de spécifier le type de contenu défini par la requête et le deuxième, le accept permettant de définir quel type de format le client accepte en retour de la part du serveur (figure 6).



Figure 6 : Requête HTTP

1.2. Maladie Kératocône

Le kératocône est une maladie dégénérative de l'œil qui se caractérise par un amincissement progressif de la cornée ainsi qu'une perte de la forme sphérique de celle-ci pour prendre la forme d'un cône irrégulier.

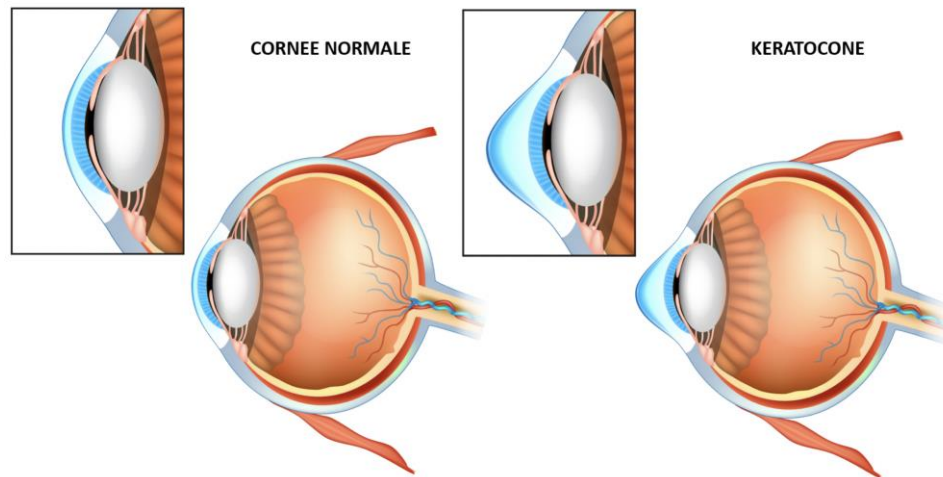


Figure 7 : Différence entre une cornée normale et une cornée malade

Cette maladie se déclare le plus souvent à la puberté et évolue le plus souvent jusqu'à l'âge de 30-40ans, mais peut également survenir à un âge plus précoce ou plus avancé. Le diagnostic se fait entre 10 et 39 ans pour 90% des patients. Elle touche indifféremment les femmes et les hommes et peut être très asymétrique au niveau de la sévérité et de la progression entre les deux yeux.

Le kératocône peut être classé selon 4 stades de sévérités différents :

- Stade 1, stade fruste (infraclinique) : le patient ne se plaint de rien, la découverte d'une anomalie morphologique de la cornée est fortuite, lors d'un bilan de chirurgie réfractive par exemple (topographie cornéenne).
- Stade 2, stade minime : la vision est troublée (surtout la nuit) mais les lunettes permettent encore une qualité de vision et de vie normale.
- Stade 3, stade modéré : les lunettes sont peu efficaces en raison de l'excès d'aberrations optiques, l'adaptation en lentilles rigides est indiquée. En cas d'intolérance, différentes stratégies de cornéoplasties dites « mini-invasives et séquentielles » sont proposées.
- Stade 4, stade sévère : la cornée est très amincie et trop bombée, les lentilles sont difficilement tolérées, la chirurgie mini invasive reste limitée, la greffe de cornée est souvent la seule issue.

L'évolution au stade supérieur n'est pas systématique et n'est pas linéaire dans le temps. De ce fait, elle est difficile à prévoir.

La prévalence de cette maladie est comprise entre une personne sur 2000 et une personne sur 500. Ces variations de calculs sont dues aux différentes méthodes de détection et différents critères de diagnostic car certaines formes légères du kératocône peuvent passer inaperçues ou être associées à d'autres maladies plus complexes.

La déformation provoquée par la maladie entraîne souvent plusieurs troubles visuels tels qu'une myopie ou un astigmatisme évolutif, un flou visuel, une baisse de l'acuité visuelle particulièrement de loin, une photophobie ou encore des difficultés lors de la conduite nocturne. Ainsi la qualité de vie est souvent détériorée mais grâce à une prise en charge adaptée le patient peut mener une vie normale.

2. Nouveaux scénarios d'utilisation

Dans l'objectif de mettre en application OM2M au sein du laboratoire Connected Health Lab différents scénarios ont été envisagés. Ainsi, le développement des prototypes d'application métier simples permet d'étudier et de mettre en application le mode de communication pour valider le déploiement et l'intégration effective de nouveaux capteurs et actionneurs.

2.1. Montre connectée :

Le premier scénario envisagé est celui permettant de suivre l'évolution du Kératocône chez un patient. L'objectif à partir de ce scénario serait de quantifier le nombre de frottements à partir d'une montre connectée possédant un altimètre et un gyroscope. La maladie s'aggrave dès lors que le patient réalise un frottement trop prononcé sur ses yeux, de plus le frottement étant un mouvement naturel, souvent réalisé involontairement, il est nécessaire de réaliser un suivi qui permettrait de quantifier et notifier le patient dès lors que ce mouvement apparaît. Le patient porterait une montre connectée, de type Xiaomi Mi Band 6 ou Fitbit Versa 2, permettant une mesure de l'altitude et de la position en continue. Les données seront transmises à une application que le patient aura sur son smartphone qui s'occupera de les envoyer sur la plateforme OM2M pour les stocker et les étudier afin d'analyser le mouvement qui est réalisé. Le patient pourra ainsi être notifié en recevant un message sur l'application qui échangerait avec la montre ou bien par vibration de la montre.

Cet objectif amènerait un gros effort sur le calibrage pour reconnaître avec précision que le patient se frotte bien les yeux. L'objectif pour le projet tuteuré se limitera à réaliser un prototype basique qui va identifier le mouvement du bras du patient qui se lève vers la tête à partir de la hauteur de la main plutôt que d'aller dans le détail du type de frottement.

2.2. Capteurs environnementaux :

L'intérêt d'OM2M étant l'interconnexion entre objets connectés, il a été envisagé d'également réfléchir à un scénario supplémentaire qui nous permettrait d'échanger plusieurs types de données.

Le second scénario implique que le patient soit porteur de lunettes. Toujours afin de quantifier les frottements oculaires et leur durée, nous pouvons imaginer calculer le nombre de fois où le patient retire ses lunettes et se frotte les yeux grâce à un système de lunettes connectées. Cependant, les patients ne possédant pas tous des lunettes, l'étude d'un élément environnemental déclenchant un frottement pourrait être étudiée (luminosité trop forte ou trop faible, air pollué, etc.).

Ainsi, les données récoltées grâce au bracelet pourraient être croisées avec un capteur de pression atmosphérique, un capteur de luminosité ou bien un détecteur de poussière dans l'environnement.

3. Spécifications de la solution

3.1. Spécifications fonctionnelles

3.1.1. Diagrammes de cas d'utilisation

Diagramme de cas d'utilisation de la montre connectée :

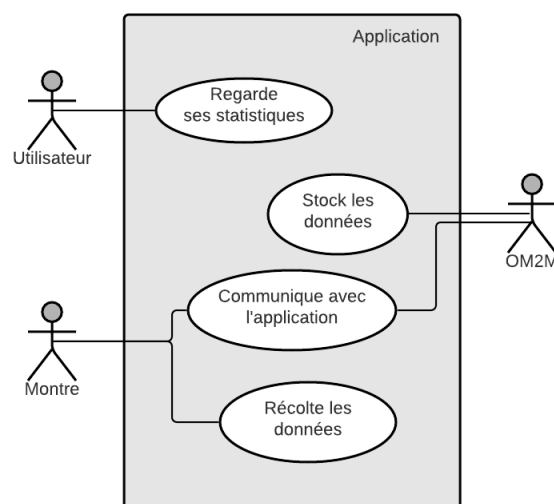


Figure 8 : Diagramme de cas d'utilisation de la montre connectée

Diagramme de cas d'utilisation des capteurs environnementaux :

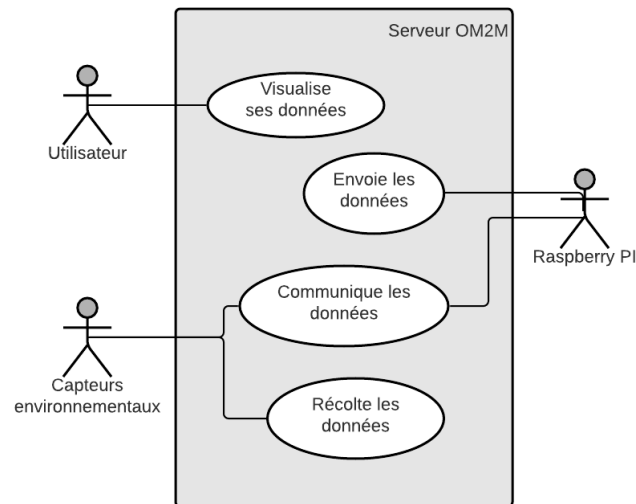


Figure 9 : Diagramme de cas d'utilisation des capteurs environnementaux

3.1.2. User stories

Montre connectée

En tant que patient atteint de Kératocône, je veux pouvoir détecter les mouvements que je réalise et être notifié à partir de la montre connectée afin de comptabiliser le nombre de fois où je me frotte et les yeux et être notifié pour pouvoir arrêter.

Capteurs environnementaux

En tant que patient atteint de Kératocône, je veux pouvoir analyser l'environnement dans lequel je me trouve afin d'être notifié quand un environnement est susceptible de m'amener à me frotter les yeux et pouvoir ainsi appréhender cette situation.

3.1.3. Architecture du système

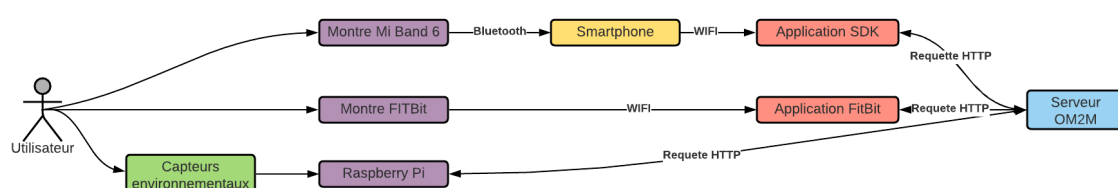


Figure 10 : Architecture du système

4. Environnement technique de la solution

Dans cette partie nous présenterons les différents langages, outils et environnements de développement que nous avons utilisés durant ce projet.

4.1. Logiciels utilisés

4.1.1. Eclipse

Nous avons utilisé le logiciel Eclipse comme environnement de travail étant donné que l'application OM2M demande à être lancée sur Eclipse.

C'est au CHL (Connected Health Lab) que nous avons implémenté le logiciel OM2M. Cette mise en place d'OM2M au CHL nous a cependant posé quelques difficultés techniques, notamment en ce qui concerne les différents prérequis à la bonne mise en place du logiciel, tels que les différentes versions de java ou Maven nécessaires. C'est pourquoi, nous avons également réalisé un « CookBook » répertoriant la bonne démarche à effectuer et les problèmes rencontrés lors du lancement d'OM2M sur Eclipse. Ce CookBook se trouve en annexe 2 de ce rapport et devrait permettre aux prochains utilisateurs d'éviter certaines complications.

4.1.2. PostMan

Pour tester les différentes requêtes envoyées au serveur OM2M, nous avons utilisé le logiciel Postman. Postman sert à exécuter des appels HTTP directement depuis une interface graphique, et à visualiser le résultat de nos requêtes sans avoir à dépendre du frontend.

Afin de récupérer les données de notre bracelet connecté Xiaomi, nous avons également utilisé une application mobile GadgetBridge permettant de faire le lien entre le bracelet et le serveur OM2M.

Enfin, l'application web Fitbit Studio nous a permis de créer une application directement sur la montre connectée afin d'en récupérer les données.

4.1.3. Fitbit Studio

Il est possible de réaliser des applications et des cadrans d'horloge pour les appareils Fitbit OS, à l'aide du kit de développement logiciel (SDK) de Fitbit, Fitbit Studio. L'architecture d'une application Fitbit Studio se présente comme suit :



Figure 11 : Architecture type d'une application Fitbit Studio

Les dossiers `/common/`, `/companion/` et `/settings/` sont des dossiers facultatifs.

Comme cet SDK sert à créer des applications pour montres, la taille d'une application ne peut excéder 10 mégaoctets à l'installation et une fois installée, l'espace total du système de fichiers utilisés ne peut excéder 15 mégaoctets, incluant les ressources et les fichiers écrits à l'aide de l'API du système de fichiers.

Les fichiers peuvent être écrits en JavaScript ou Typescript. Pendant le build, les scripts sont compilés automatiquement par le compilateur TypeScript et rollup.js.

À l'aide de Fitbit Studio nous avons réalisé une application pour montre qui nous permet de récupérer l'altitude de la montre en mètres et de l'afficher sur l'écran de notre Versa 2. Voici figure 12 un extrait de code qui nous permet de récupérer l'altitude.

```

1
2 import { display } from "display";
3 import * as document from "document";
4
5 import { geolocation } from "geolocation";
6
7
8 const altitudeData= document.getElementById("altitude-data");
9 const altitudeLabel=document.getElementById("orientation-label");
10
11 var watchID = geolocation.watchPosition(locationSuccess, locationError, { timeout: 60 * 1000 });
12
13
14 function locationSuccess(position) {
15   altitudeData.text=position.coords.altitude;
16   console.log("Altitude: " + position.coords.altitude+ " mètres");
17 }
18
19 function locationError(error) {
20   console.log("Error: " + error.code, "Message: " + error.message);
21 }
22

```

Figure 12 : Code récupérant l'altitude

Ensuite nous avons essayé d'ajouter un « companion » à notre application afin d'envoyer vers OM2M les données récupérées. Un companion permet d'améliorer une application Fitbit en lui fournissant un environnement d'exécution JavaScript supplémentaire qui existe dans l'application mobile Fitbit. L'ajout d'un companion n'est pas obligatoire mais est essentiel si nous voulons pouvoir effectuer des tâches telles que par exemple communiquer avec des API Web, télécharger des images ou d'autres ressources sur Internet ou encore utiliser le capteur GPS de l'appareil mobile.

Dans notre cas, nous cherchons à effectuer une requête POST pour envoyer nos données à OM2M. Voici figure 13 le code de notre requête.

```

1 fetch('http://127.0.0.1:8080/~in-cse',
2   {
3     method: "POST",
4
5     headers: {
6       "X-M2M-Origin": 'admin:admin',
7       "Content-Type": 'application/json;ty=2',
8       "Access-Control-Allow-Origin": '*'
9     },
10
11     body: {
12       "m2m:ae": {
13         "api": "app-sensor",
14         "rr": "false",
15         "lbl": ["Type/sensor", "Category/temperature", "Location/home"],
16         "rn": "MY_SENSOR_test3"
17       }
18     }
19
20 }).then(function(response) {
21   return response;
22 }).catch(function(error) {
23   console.log('Il y a eu un problème avec l\'opération fetch: ' + error.stack);
24 });
  
```

Figure 13 : Requête POST

La requête est envoyée au format JSON à l'url de OM2M. Nous utilisons l'API Fetch qui nous permet d'envoyer et de récupérer des ressources à travers le réseau de manière asynchrone. Tout d'abord nous spécifions l'url vers laquelle sera envoyée notre requête. Ensuite nous spécifions notre méthode d'envoi dans *method*. Dans les *headers* nous spécifions l'identifiant et le mot de passe pour permettre à fetch d'écrire dans OM2M ainsi que le *Content-Type* qui sert à indiquer le type MIME de la ressource et l'entête *Access-Control-Allow-Origin* qui renvoie une réponse indiquant si les ressources peuvent être partagées avec une origine donnée. Enfin, dans *body*, nous spécifions l'ensemble des données que nous voulons envoyer vers OM2M.

Malheureusement, la requête ne fonctionne pas et nous n'arrivons pas à communiquer avec OM2M depuis la Versa 2.

4.2 Technologies utilisées

4.2.1. Montre connectée Fitbit Versa 2

Afin de récolter des données et réaliser plusieurs tests, nous avons eu accès à la montre connectée Fitbit Versa 2. Cette montre fonctionne avec un système d'exploitation propriétaire, Fitbit OS, et est compatible avec les systèmes Android 5.0 et supérieur et iOS 5 et supérieur. La Versa 2 a une mémoire de 4GB (avec 2.5GB disponibles pour stocker de la musique). Elle sauvegarde sept jours de données de mouvement et sauvegarde les totaux quotidiens des 30 derniers jours. De plus, elle enregistre les données de fréquence cardiaque à intervalles d'une seconde pendant l'exercice et à intervalles de 5 secondes le reste du temps. Elle est compatible NFC, Bluetooth 4.0 et Wi-Fi.

Au niveau des capteurs nous avons accès à :

- Accéléromètre 3-axes
- Cardiofréquencemètre optique
- Altimètre
- Capteur de lumière ambiante
- Moteur de vibration
- Antenne Wi-Fi (802.11 b/g/n)
- Capteur de SpO2 relative
- NFC
- Microphone
- Bluetooth 4.0



Figure 14 : Fitbit Versa 2

La montre se connecte et partage ses données avec l'application Fitbit, disponible sur Android et iOS. Sur cette application nous pouvons retrouver les informations recueillies par la montre. Nous avons par exemple accès au nombre de pas effectués au cours des jours de la semaine, à la qualité du sommeil et aux différentes phases de celui-ci, ou encore à une courbe de la saturation en oxygène sanguine estimée pendant la nuit.

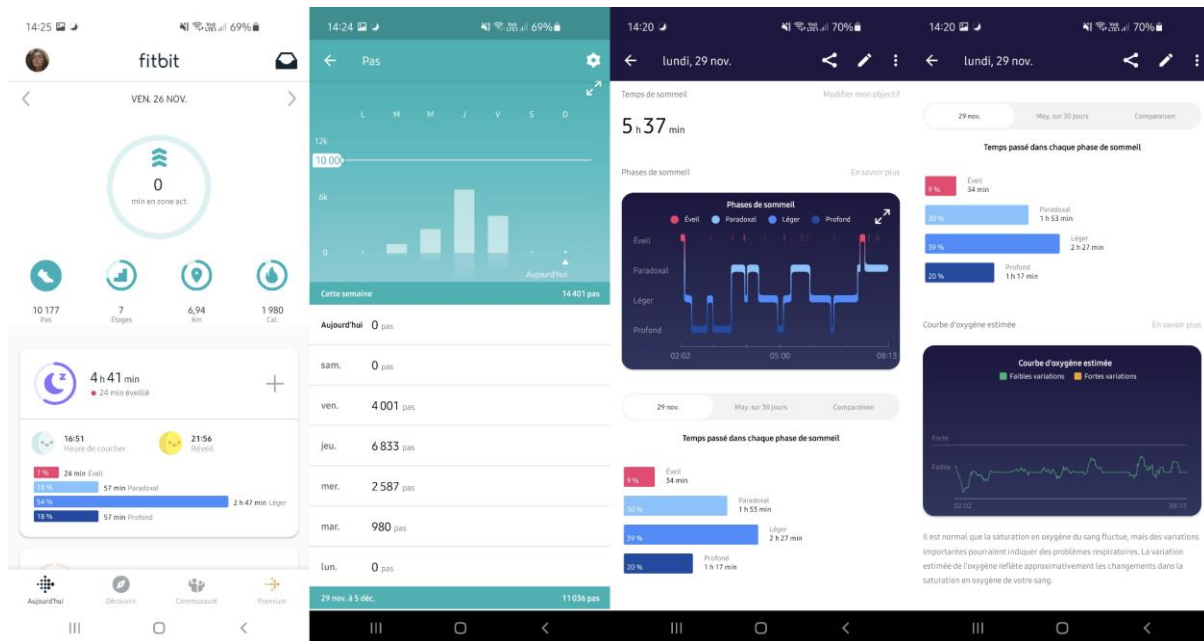


Figure 15 : Quelques images de l'application FitBit

La Versa 2 ne possède pas de GPS intégré, elle utilise donc celui du téléphone lié à la montre afin d'obtenir une distance parcourue ou un tracé du trajet sur une carte. Il faut bien entendu avoir le téléphone sur soi pour que cela fonctionne.

4.2.2. Xiaomi Mi Band 6

En plus de la montre connectée FitBit Versa 2, nous avons également choisi de travailler avec le bracelet connecté Mi Band 6 de la marque Xiaomi. Ce bracelet est lui aussi compatible avec les systèmes d'exploitation Android et IOS, cependant, en termes de connectivité, le bracelet est compatible avec Bluetooth 5.0 uniquement.

Voici la liste des capteurs présents sur le bracelet connecté Mi Band 6 :

- Cardiofréquencemètre avec SpO2
- Accéléromètre
- Gyroscope



Figure 16 : Xiaomi Mi Band 6

Il est possible de visualiser les données du bracelet grâce à l'application Mi Fit. Cependant, cette application ne permet pas d'exporter les données afin de les envoyer au serveur OM2M. Nous avons donc dû passer par une application mobile afin de récupérer les données. Cette application se nomme GadgetBridge et est une solution open-source permettant de récupérer les données de nombreux objets connectés dont les bracelets Mi Band 6. Il serait donc possible à partir de cette application, et sous réserve de rajouter quelques développements de lire et d'envoyer les données de localisation et d'altitude vers le serveur OM2M.



Figure 17 : Export des données à l'aide de l'application GadgetBridge

4.2.3. Capteurs environnementaux :

Nous avons voulu aussi rajouter d'autres types de données permettant le suivi du patient dans l'évolution du kératocône et permettant aussi de mettre en application toute la force de OM2M dans l'interconnexion entre différents objets connectés utilisant chacun son propre langage. Pour cela, nous voulons analyser l'environnement dans lequel se situe le patient, tel que l'humidité, la température ou bien la luminosité. Pour cela nous utilisons une carte Raspberry Pi Sense 3 avec un kit de capteurs environnementaux, le *Grove Base Kit for Raspberry Pi*, ce kit permet d'avoir des capteurs de luminosité, de moisissure et un capteur de température et d'humidité dont nous avons besoin.

Raspberry est un nano-ordinateur monocarte à processeur ARM (Advanced RISC Machines), nous avons choisi de travailler sur ce type de carte car elle dispose de pins GPIO (General Purpose Input/Output), qui sont des ports d'entrées-sorties permettant la connexion de cartes d'extensions et d'autres composants électroniques nous permettant ainsi de rajouter les différents types de capteurs dont nous avons besoin pour réaliser les différentes mesures sur l'environnement de l'utilisateur.

Pour initier une Raspberry Pi, un système d'exploitation doit être flashé sur une carte SD qui permettra par la suite de la faire tourner. Pour l'utiliser deux cas sont possibles. Tout d'abord en reliant un écran, un clavier et une souris à la carte permettant de l'utiliser comme un mini-ordinateur, ou alors en utilisant un logiciel permettant le contrôle à distance d'ordinateur.

Dans notre cas, nous avons utilisé le logiciel VNC Viewer pour accéder à la carte depuis les serveurs du CHL. Cette manipulation est possible à partir du protocole SSH (Secure Shell) et amenant la connexion à un hôte distant via l'adresse IP de l'hôte, d'une authentification par mot de passe et d'un câble Ethernet reliant l'ordinateur et la carte (figure 18).

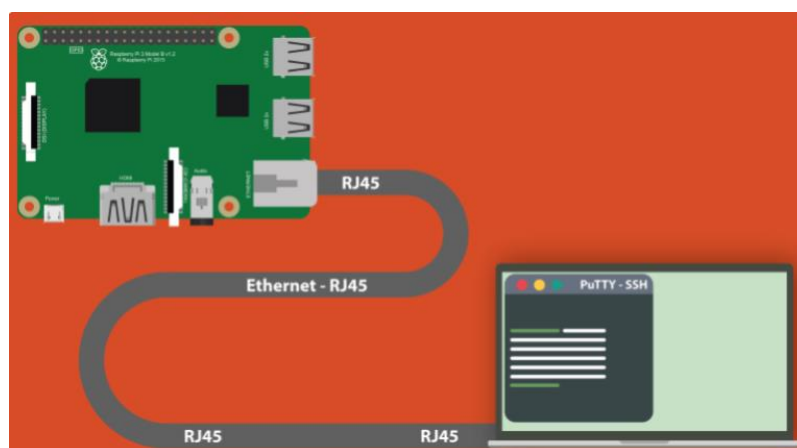


Figure 18 : Connexion à distance de la Raspberry

Concernant le kit de capteurs environnementaux, ce dernier est composé d'une platine à chapeau qui fait office de passerelle entre les modules Grove et la Raspberry Pi en recevant et en envoyant des données ainsi qu'en exécutant des commandes envoyées par la Raspberry Pi.

Les modules Grove regroupent 3 différents types de connecteurs, analogique, digital, et le bus i2c. Les avantages de se servir de ce type de module sont :

- Les connecteurs sont standards ce qui permet de relier facilement un capteur avec une carte GrovePi.
- Un nombre important de capteurs différents existe, ce qui permet d'éviter de souder pour ajouter un capteur spécifique.

Elle contient un microcontrôleur qui est relié via les bus I2C et SPI au Raspberry Pi. On peut du coup depuis le Raspberry Pi communiquer via i2c, qui est un moyen de communication synchrone multi target/multicontroller (figure 19).

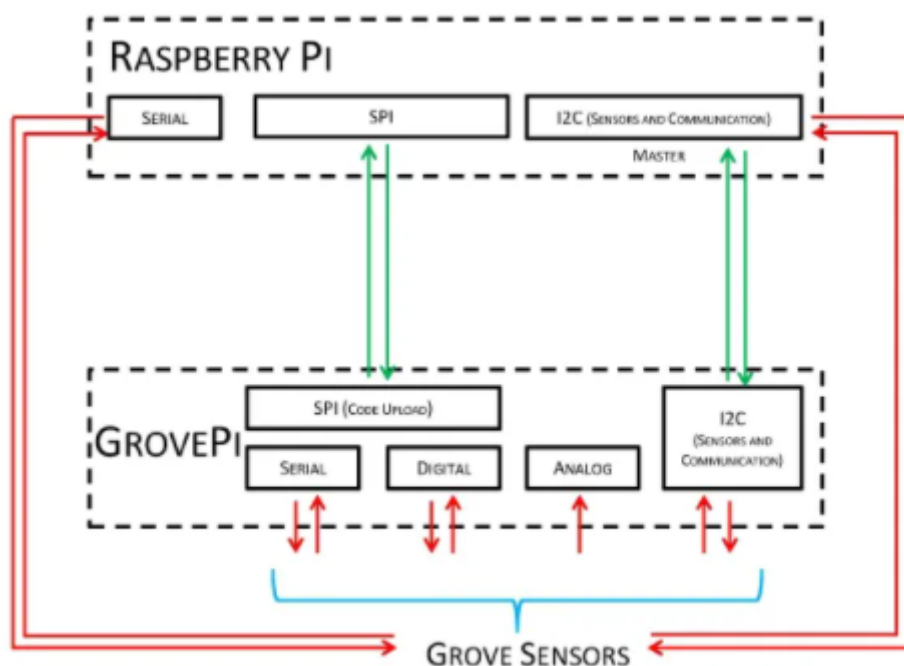


Figure 19 : Architecture logicielle entre Raspberry Pi et Grove Sensor

Elle permet ainsi d'interfacer pratiquement tous les capteurs/actionneurs de la famille Grove. On peut les télécharger depuis Github, ce qui permet une utilisation facile de ces derniers et permettant la manipulation des capteurs à partir de plusieurs langages de programmation comme Python.

Dans notre cas, nous avons utilisé le capteur de température et d'humidité DHT11 (figure 20).

Ce capteur permet une très bonne compatibilité avec l'interface Grove. Ses caractéristiques sont :

- Plage de Température : 0°C à $50^{\circ}\text{C} \pm 1^{\circ}\text{C}$
- Plage d'humidité : 20% à 90% RH $\pm 2\%$ RH

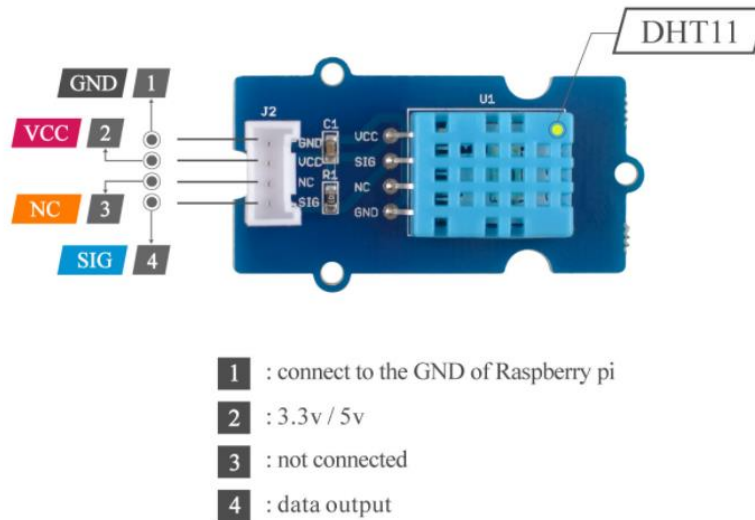


Figure 20 : Capteur d'humidité et de température DHT11

On se retrouve ainsi avec le capteur d'environnement relié à l'interface Grove, elle-même reliée à la carte Raspberry qui échange ces données avec le serveur.

Pour manipuler les données du capteur, l'installation de l'import Grove sur Python est nécessaire. Cet import permet d'amener différentes librairies qui vont permettre de manipuler facilement les données des capteurs.

Les détails du code sont référencés au sein de l'annexe 1.

A partir des données récupérées, il a fallu les transmettre au serveur OM2M. Pour accéder à la plateforme depuis la carte, ce dernier doit tourner sur les serveurs, il suffit ensuite d'appeler le serveur local sur lequel se trouve la plateforme à partir du lien HTTP et de l'adresse IP du serveur (figure 21).

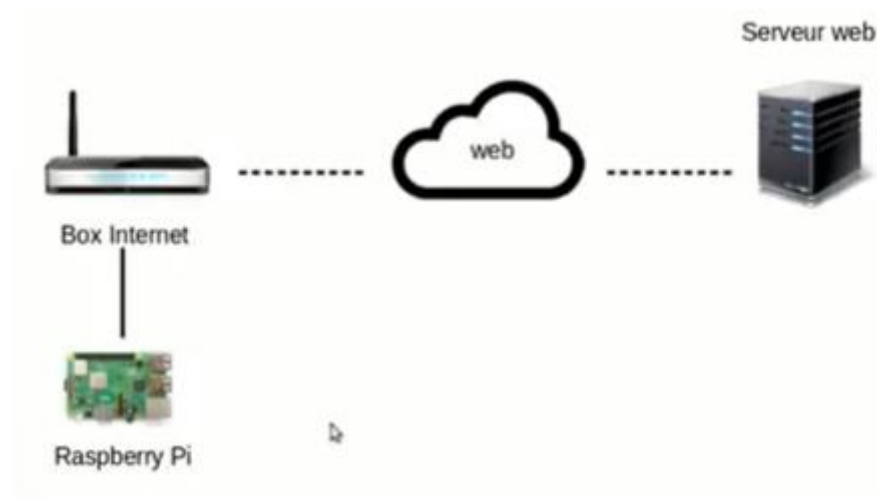


Figure 21 : Envoi des données vers OM2M

A partir du moment où la carte capte les données et arrive à communiquer avec OM2M, il faut structurer le squelette de l'application qui va contenir les données. Pour cela, nous avons d'abord réalisé la création d'une application MY_Environnement à partir d'une méthode POST sur le serveur (figure 22).

Attribute	Value
rn	MY_ENVIRONNEMENT
ty	2
ri	/in-cse/CAE290592346
pi	/in-cse
ct	20211212T153304
lt	20211212T153304
lbl	<ul style="list-style-type: none"> Type/sensor Category/temperature Location/home Category/humidity
acpi	<div>AccessControlPolicyIDs</div> <div>/in-cse/acp-656543691</div>
et	20221212T153304
api	app-sensor
aei	CAE290592346
rr	false

Figure 22 : Création de l'application MY_Environnement

Après la création de l'application, nous avons créé un descripteur qui va permettre de décrire l'application. Nous avons de plus créé un Content Instance permettant de structurer les ressources que nous allons recueillir. On se retrouve ainsi avec deux conteneurs, un pour l'humidité (figure 23) et un pour la température (figure 24).

con	Attribute	Value
	type	Humidity_Sensor
	location	Home
	appld	MY_ENVIRONNEMENT
	getValue	/in-cse/in-name/MY_ENVIRONNEMENT/DATA/la

Figure 23 : Descripteur humidité

con	Attribute	Value
	type	Temperature_Sensor
	location	Home
	appld	MY_ENVIRONNEMENT
	getValue	/in-cse/in-name/MY_ENVIRONNEMENT/DATA/la

Figure 24 : Descripteur température

A partir des descripteurs de ressources il ne nous restait plus qu'à construire un conteneur DATA qui stockera les données décrites et structurées par les descripteurs.

Ainsi, grâce à ce squelette (figure 25), nous pouvons récupérer les données aussi bien de l'humidité que de la température, il ne reste plus qu'à y injecter les données récoltées par la carte.

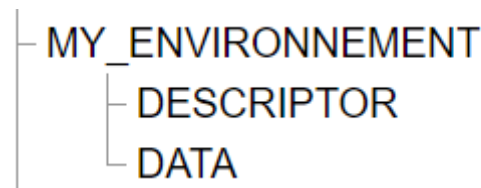


Figure 25 : Hiérarchie MY_Environnement

Pour envoyer les données, il suffit de les envoyer à partir d'une méthode POST sur le conteneur DATA correspondant à l'application MY_Environnement

Ainsi nous avons pu envoyer ou essayer d'envoyer des données vers OM2M à l'aide de plusieurs types d'appareils afin de tester et de prendre en main la plateforme. Nous avons aussi pu utiliser toute la force de OM2M en permettant de récupérer différents types et formats de données et de pouvoir les stocker au sein d'un même environnement.

4.3 Tests, validation et pistes d'amélioration

Pour résumer l'avancement et l'évolution du projet aujourd'hui, nous avons tout d'abord réussi à installer OM2M sur un serveur spécifiquement mis en place au CHL.

En parallèle, nous avons également réalisé un guide d'installation d'OM2M dans lequel sont répertoriés les différents problèmes rencontrés ainsi que les solutions trouvées.

En ce qui concerne la montre et le bracelet connecté, nous avons pu récupérer les données concernant les mouvements de l'utilisateur, mais nous avons rencontré quelques problèmes lors de l'envoi de ces données vers le serveur OM2M.

Concernant les capteurs environnementaux, ces derniers permettent de récupérer la valeur de la température et de l'humidité ambiante en les mesurant toutes les 3 secondes.

Pour valider les échanges réalisés entre la carte et OM2M, nous avons d'abord réalisé une méthode GET pour récupérer les données contenues au sein de notre serveur et pour vérifier la connexion établie avec ce serveur.

Nous avons ensuite envoyé les données mesurées par la carte vers l'application de OM2M dédiée (figures 26 et 27).

Attribute	Value
appld	MY_ENVIRONNEMENT
category	temperature
data	19
data	19
data	19
data	19
unit	celsius

Figure 26 : Données de température récupérées

Attribute	Value
appld	MY_ENVIRONNEMENT
category	humidity
data	52
data	52
data	52
data	52
unit	%

Figure 27 : Données d'humidité récupérées

A ce niveau du projet, nous pouvons donc visualiser et stocker les données récupérées par les capteurs.

5. Gestion du projet et de l'équipe

5.1. Outils de communication et de partage

Pour pouvoir échanger entre nous, nous avons choisi d'utiliser Messenger, un service de messagerie instantanée développé par Facebook. Pour communiquer avec le commanditaire et notre tuteur école, nous avons réalisé principalement des échanges par mail afin de les tenir au courant de l'avancement des étapes tout en posant des questions sur certaines spécifications. Nous avons également réalisé des réunions d'avancement avec tous les intervenants du projet à l'aide de l'application Zoom. Enfin, notre tuteur école nous a également permis de rejoindre le serveur discord du CHL pour tout ce qui était question de

l'utilisation des locaux du CHL, ou des commandes de capteurs. Nous avons également créé un serveur Slack avec notre commanditaire, pour pouvoir lui poser certaines questions de niveau technique.

Aussi, pour travailler collaborativement, nous avons utilisé Google Drive, un service cloud permettant de partager des documents.

Partagés avec moi > PTUT LAAS-CNRS









Nom ↑	Dernière modif...
 Devis	17 nov. 2021
 Données	17 nov. 2021
 Rapport	17 nov. 2021
 Recherches	17 nov. 2021
 Réunions	7 déc. 2021
 Fiche_Projet_FIE4_5_LAAS-CNRS.docx	8 nov. 2021
 Horaires	8 nov. 2021
 Présentation	11 déc. 2021

Figure 28 : Dossier Google Drive

5.2. Gestion des tâches

Afin de pouvoir suivre en temps réel les activités et tâches que nous avons à réaliser, nous avons pris la décision d'utiliser l'application Trello. C'est un outil de gestion de projet en ligne. Il repose sur une organisation des projets en planches listant des cartes, chacune représentant des tâches. En fonctionnant avec une méthode « TO DO – DOING – DONE » (« A Faire - En Cours – Fait »), ce qui permettait à tous de connaître les avancements de chacun et les tâches qu'il restait à accomplir. Avec cet outil, il est aussi possible d'attribuer les différentes tâches aux différents membres du projet.

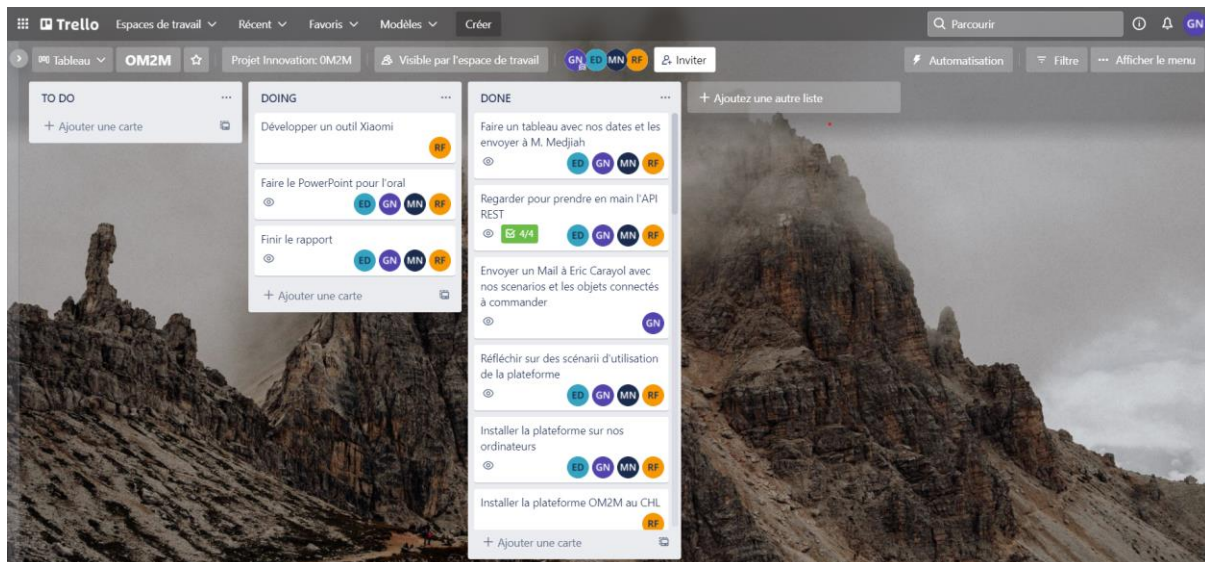


Figure 29 : Capture du Trello utilisé

Pour avoir une représentation plus grossière, mais aussi plus imagée de la temporalité des différents évènements qui ont marqués le projet, nous avons réalisé le Diagramme de Gantt ci-dessous (figure 30).

On peut alors voir que le mois d'Octobre a été consacré à la mise en place du projet avec notamment l'installation d'OM2M ainsi que la première rencontre avec les différents acteurs du projet au cours de la réunion du 19 Octobre. Au cours de cette dernière, nous avons identifié les objectifs du projet et nous avons défini les premières missions.

Alors dans un second temps, il a été décidé que l'installation de la plateforme OM2M au CHL pouvait être enrichissante, c'est pourquoi nous avons décidé de nous y installer pour la suite du projet, afin de permettre de transmettre les données de nos objets connectés directement vers le serveur du CHL.

Ensuite, le 17 novembre puis le 24 novembre, nous avons respectivement reçu la FitBit Versa 2, puis les 2 Mi Band et nous avons dès lors commencé à créer des données et essayer de développer des applications afin de récupérer les données qui nous intéressent (notamment au niveau de l'altitude ou de l'accélération).

Et finalement, très peu de temps après, nous avons reçu les capteurs ainsi que la Raspberry, ce qui nous a permis de travailler sur l'analyse des données environnementales, pour pouvoir, à terme, juger de leur influence sur le frottement des yeux.

Projet OM2M

DIAGRAMME DE GANTT

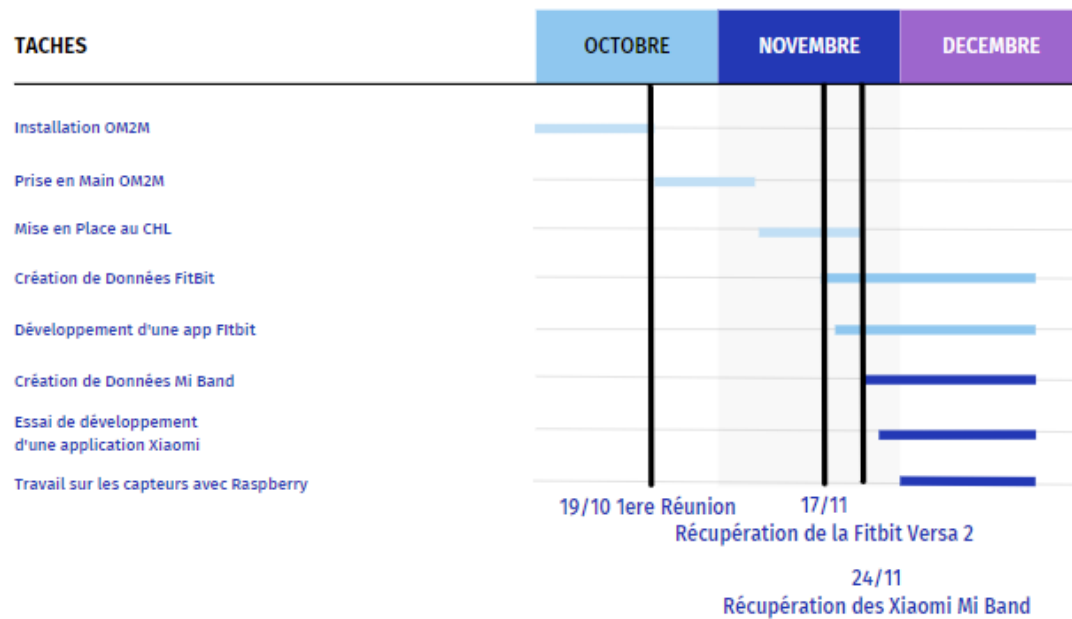


Figure 30 : Diagramme de Gantt

Conclusion et perspectives

Au cours de ce projet, nous avons eu l'occasion de travailler avec de vrais acteurs de la recherche médicale autour du sujet du kératocône. Nous avons alors eu l'occasion de réfléchir, ensemble, à la création de différents objets connectés qui pourraient aider à accompagner le traitement de cette maladie. De plus, grâce à ces échanges nous avons d'abord discuté autour de l'outil à développer, nous avons pensé à une bague connectée, des lunettes connectées, mais finalement nous sommes partis sur l'idée de développer un objet de type bracelet connecté (avec la montre Fitbit Versa 2 ainsi que la Xiaomi Mi Band 6). Alors, avec ses objets, nous avons pensé le développement d'applications pour essayer de détecter le mouvement des bras vers les yeux, ce qui nous a amené à rencontrer certaines difficultés.

En effet, en termes de temporalité, il a très vite été décidé, en raison des contraintes de temps (notre projet durant 3 mois), de se focaliser sur l'exportation de données exploitables. Traiter les données récoltées afin d'être capable de reconnaître une élévation de la main ou un frottement avec une partie du corps demande beaucoup d'apprentissage informatique et paraissait un peu trop complexe pour cette partie.

C'est pourquoi les différentes avancées de notre projet nous ont amenées à la création d'outils de travail (tel que le CookBook) pour une éventuelle suite du projet.

Les perspectives envisagées pour le projet seraient d'arriver à faire communiquer la montre Fitbit avec OM2M au travers des requêtes HTTP. Il serait aussi possible d'envisager de développer une application mobile qui permet de récupérer aussi bien les données de la montre que des capteurs à partir de OM2M pour tout centraliser sur un téléphone et rendre ça plus visible.

Webographie

OM2M

<https://www.eclipse.org/OM2M/>

<https://wiki.lafabriquedesmobilites.fr/wiki/OM2M>

<https://wiki.eclipse.org/OM2M/one>

<https://wiki.eclipse.org/OM2M>

<https://www.onem2m.org/>

[https://fr.wikipedia.org/wiki/Machine to machine](https://fr.wikipedia.org/wiki/Machine_to_machine)

<https://www.matooma.com/fr/s-informer/univers-du-m2m-et-iot#chapter-one-url>

Kératocône

<https://www.chu-toulouse.fr/-qu-est-ce-que-le-keratocone-#:~:text=D%C3%A9finition%20du%20k%C3%A9ratoc%C3%B4ne,-Publi%C3%A9%20le%2017&text=Autrement%20dit%2C%20le%20k%C3%A9ratoc%C3%B4ne%20est,la%20d%C3%A9formation%20de%20la%20corn%C3%A9e.>

<https://www.keratocone.net/definition.html>

<https://www.centreophtalmologiejeanjaures.fr/pathologies-ophtalmologiques/pathologies-maladies-de-la-cornee/symptomes-traitement-keratocone-toulouse.html>

<https://fr.wikipedia.org/wiki/K%C3%A9ratoc%C3%B4ne>

<https://www.dencott.com/fr/blog/stades-du-keratocone-queelles-sont-les-options-therapeutiques.html>

Fitbit

<https://dev.fitbit.com/build/guides/application/>

<https://dev.fitbit.com/getting-started/#overview>

<https://dev.fitbit.com/build/guides/geolocation/>

<https://www.lesnumeriques.com/montre-connectee/fitbit-versa-2-p53609/test.html>

https://developer.mozilla.org/fr/docs/Web/API/Fetch_API/Using_Fetch

<https://developer.mozilla.org/fr/docs/Web/API/Headers>

<https://developer.mozilla.org/fr/docs/Web/HTTP/Headers/Content-Type>

<https://developer.mozilla.org/fr/docs/Web/HTTP/Headers/Access-Control-Allow-Origin>

GadgetBridge

<https://github.com/Freeyourgadget/Gadgetbridge>

<https://codeberg.org/Freeyourgadget/Gadgetbridge/wiki>

Annexes

Annexe 1 : Les différents code utilisé tout au long de ce projet sont intégré au sein du dépôt Github

https://github.com/EmmaDupuy/ptut2021-2022_DUPUY_FAUCHER_GOETSCHER_NOURRISSON

Annexe 2 : CookBook



Guide de bonnes pratiques et aide au déploiement de OM2M

Réalisé par

Emma Dupuy

Ruby Faucher

Numa Goetschel

Marie Nourrisson

Table des matières

Installation de OM2M	38
1. Prérequis	38
2. Installation d'Eclipse	38
3. Cloner le projet	39
4. Importer le projet sur Eclipse	39
5. Installation du Plugin Tycho	40
6. Lancer un build du projet	43
6.1 Avec Eclipse	43
6.2 En ligne de commande	43
7. Tester OM2M	43
8. Ajout d'un plugin Bluetooth	44

Installation de OM2M

Le tutoriel complet de l'installation et du lancement d'OM2M est présent sur le lien suivant :

<https://wiki.eclipse.org/OM2M/one/Clone>

Il est cependant possible de faire face à quelques complications au cours de cette installation, nous avons donc réalisé ce guide d'installation qui reprend chaque étape que nous avons dû effectuer ainsi les différents problèmes rencontrés et leur résolution.

1. Prérequis

Il est nécessaire d'avoir git afin de cloner le projet.

Il faut avoir la version 1.8 de java. Pensez donc à vérifier que votre variable JAVA_HOME pointe bien vers un jdk 1.8. Pour cela vous pouvez taper la commande suivante dans votre terminal :

Sous linux : `echo $JAVA_HOME`

Sous Windows: `echo %JAVA_HOME%`

La version 3 de maven est également nécessaire. Il est possible de vérifier la version de maven en tapant la commande suivante dans votre terminal :

`mvn -version`

2. Installation d'Eclipse

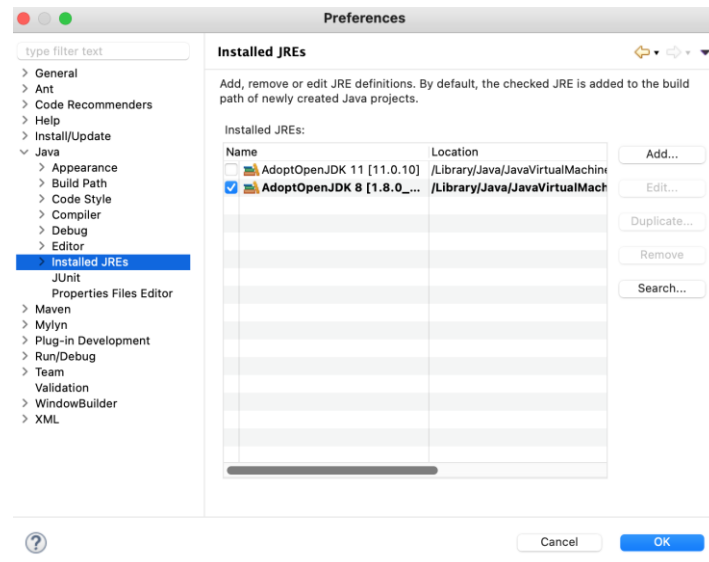
Avant de lancer le projet, il est nécessaire d'installer « Eclipse Luna for RCP and RAP Developers ».

Vous pouvez télécharger Eclipse en passant par ce lien :

<https://eclipse.org/downloads/packages/eclipse-rcp-and-rap-developers/lunasr2>

Une fois sur Eclipse, pensez à vérifier encore une fois que le JRE utilisé par Eclipse correspond bien à la version 1.8. Pour cela, allez dans Eclipse->Préférences

Cliquez sur Java->Installed JREs, et vérifiez que le bon jdk soit coché.



3. Cloner le projet

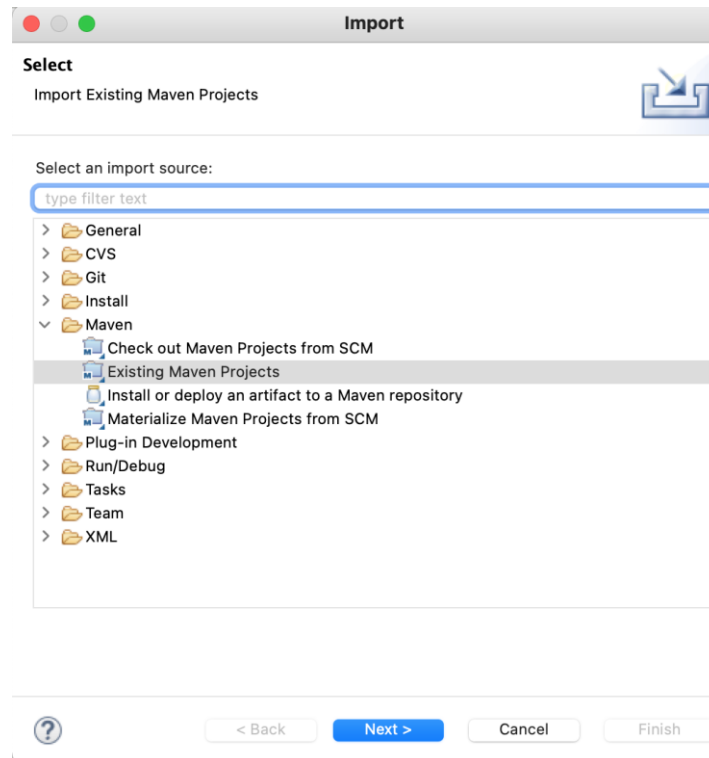
Vous pouvez cloner le projet en utilisant la commande suivante sur votre terminal :

```
git clone https://git.eclipse.org/r/OM2M/org.eclipse.OM2M
```

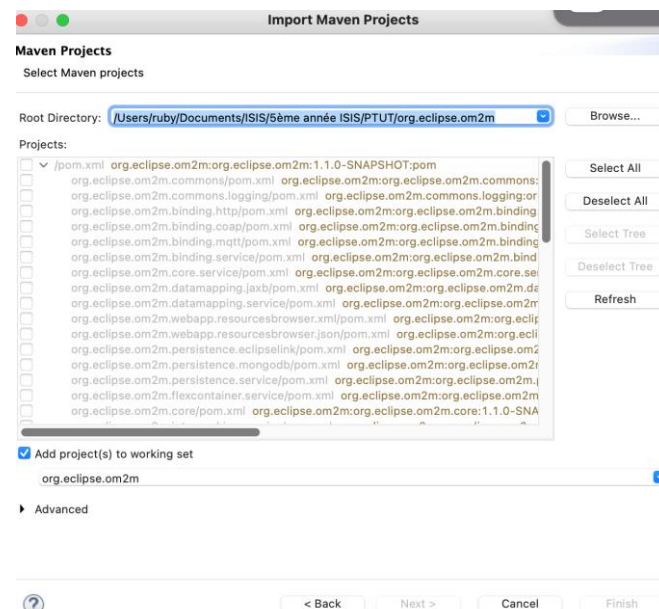
4. Importer le projet sur Eclipse

Une fois Eclipse lancé, importez votre projet en cliquant sur File->Import.

Cliquez ensuite sur Maven->Existing Maven Projects->Next.



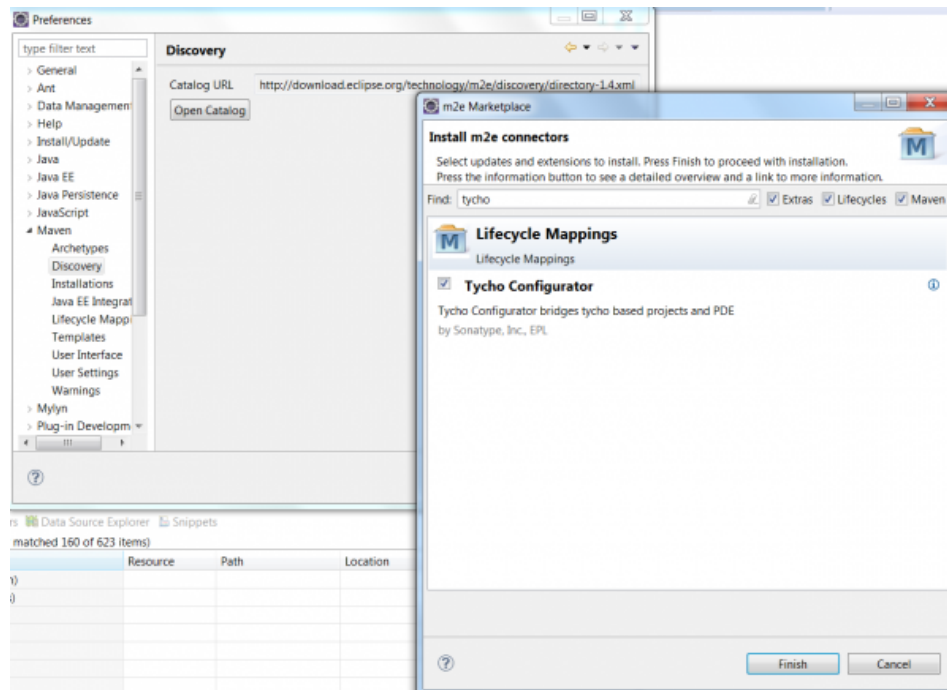
Sélectionnez ensuite votre Projet en cliquant sur “Browse” et faites “Next”.



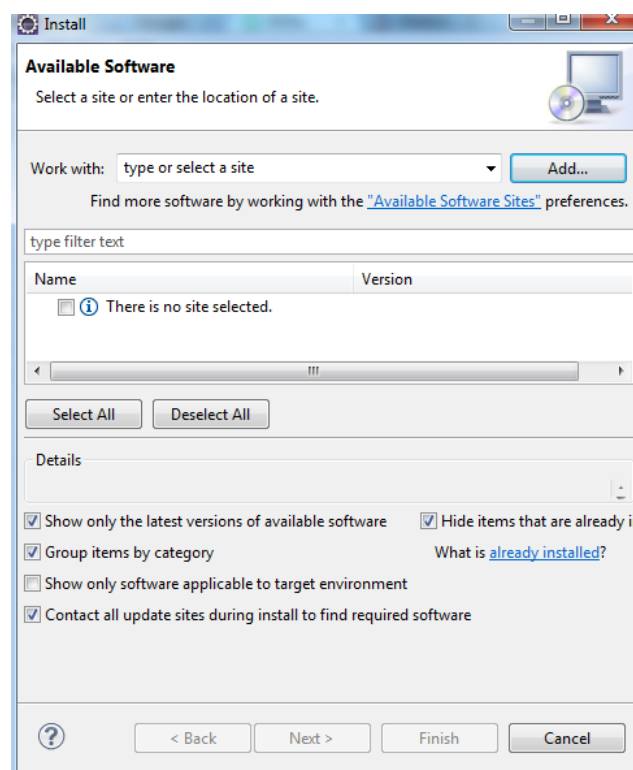
5. Installation du Plugin Tycho

Il est possible d’installer rapidement tycho en passant par Eclipse Maven MarketPlace. Pour cela il faut suivre les étapes suivantes :

Window -> Preferences -> maven -> discovery -> open catalog et taper “Tycho”. Cocher la case “Tycho Configurator” et cliquer sur “Finish”.

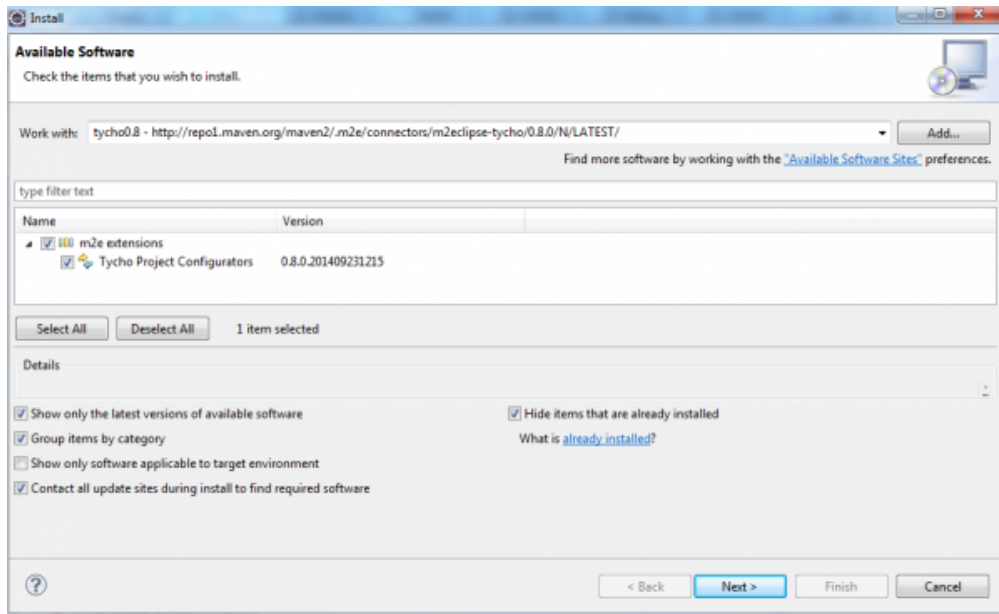


Il se peut cependant qu'un message d'erreur apparaisse, et qu'il ne soit pas possible d'installer tycho en passant par Eclipse Maven MarketPlace. Dans ce cas-là, il faut alors sélectionner Help->Install New Software et cliquer sur le bouton Add.

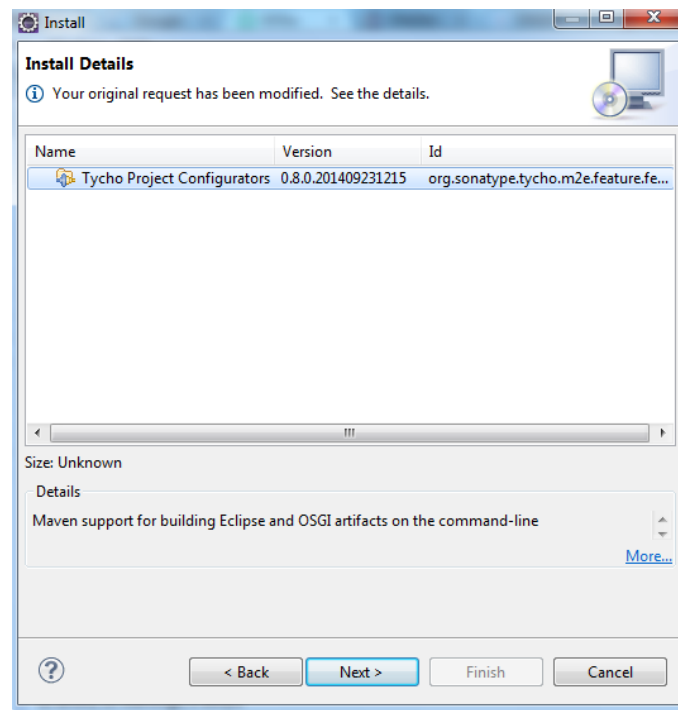


Il faut ensuite taper "tycho0.8" dans le champ "name", et dans le champ "field" le lien suivant : <https://repo1.maven.org/maven2/.m2e/connectors/m2eclipse-tycho/0.8.0/N/LATEST/>.

Cliquez alors sur le bouton “Ok” et cochez les cases “m2e extensions” et “Tycho Project Configurators” avant de cliquer sur “Next”.



Sélectionnez alors “Tycho Project Configurators” et cliquez sur “Next”.



Enfin, il faut accepter les termes d'utilisations et cliquer sur “Finish”.

Pensez ensuite à mettre votre projet à jour en sélectionnant le package “org.eclipse.OM2M” , click droit, maven, update Project.

6. Lancer un build du projet

6.1 Avec Eclipse

Sélectionnez le package “org.eclipse.OM2M”, click droit->Run as->maven install.

6.2 En ligne de commande

Si le build maven ne passe pas par eclipse, lancer maven install en lignes de commandes. Dans votre terminal, rendez-vous dans votre projet org.eclipse.OM2M et tapez la ligne suivante:

```
mvn clean install -DskipTests
```

7. Tester OM2M

Une fois le Build effectué, dans votre terminal, rendez-vous dans le dossier suivant :

```
“org.eclipse.OM2M/org.eclipse.OM2M.site.in-cse/target/products/in-cse/<os>/<ws>/<arch>”
```

Une fois dans le dossier, si vous tapez la commande ls, vous devriez voir deux fichiers nommés “start.bat” et “start.sh”. Si vous êtes sur linux ou Mac OS, vous devrez utiliser la commande “start.sh”, et “start.bat” si vous êtes sous Windows. Avant d’utiliser la commande, vérifiez que vous avez bien les droits sur ces commandes en tapant :

```
ls -l
```

Si vous n’avez pas les droits , ajoutez-les à l’aide de la commande:

```
chmod u+x start.sh
```

Lancez alors la commande

Sous linux: `./start.sh`

Sous Windows: `start.bat`

Lorsque In-cse aura démarré, vous devriez voir une console OSGI apparaître. Tapez “ss” pour voir un rapport résumant les plugins installés.

```

INFO] - org.eclipse.om2n.core.Activator
OSGI Started
[INFO] - org.eclipse.om2n.uebapp.resourcesbrowser.xml.Activator
HttpService discovered
[INFO] - org.eclipse.om2n.uebapp.resourcesbrowser.xml.Activator
Register /webpage http context
osgi> ss
Framework is launched."

id      State      Bundle
1       RESOLVED   javax.servlet_3.1.0.v20140303-1511
2       RESOLVED   javax.xml_1.3.4.v201005080400
3       RESOLVED   org.apache.commons.codec_1.6.0.v201305230611
4       RESOLVED   org.apache.commons.logging_1.1.1.v201101211721
5       ACTIVE     org.eclipse.osgi_3.10.2.v20150203-1939
6       ACTIVE     org.eclipse.felix.gogo.command_0.10.0.v201209301215
7       ACTIVE     org.eclipse.felix.gogo.runtime_0.10.0.v201209301036
8       RESOLVED   org.eclipse.felix.gogo.shell_0.10.0.v201212101605
9       RESOLVED   org.apache.httpcomponents.httpclient_4.3.6.v201411290715
10      RESOLVED   org.apache.httpcomponents.httpcore_4.3.3.v201411290715
11      ACTIVE     org.eclipse.equinox.console_1.1.0.v20140311-1639
12      ACTIVE     org.eclipse.equinox.http.jetty_3.0.200.v20131021-1843
13      ACTIVE     org.eclipse.equinox.http.servlet_1.1.500.v20140318-1755
14      RESOLVED   org.eclipse.equinox.launcher_1.3.0.v20140415-2008
15      RESOLVED   org.eclipse.jetty.continuation_9.1.16.v20140903
16      RESOLVED   org.eclipse.jetty.http_9.1.16.v20140903
17      RESOLVED   org.eclipse.jetty.io_9.1.16.v20140903
18      RESOLVED   org.eclipse.jetty.security_9.1.16.v20140903
19      RESOLVED   org.eclipse.jetty.server_9.1.16.v20140903
20      RESOLVED   org.eclipse.jetty.servlet_9.1.16.v20140903
21      RESOLVED   org.eclipse.jetty.util_9.1.16.v20140903
22      ACTIVE     org.eclipse.om2n.binding.coap_1.0.0.20151112-1027
23      RESOLVED   org.eclipse.om2n.binding.http_1.0.0.20151112-1027
24      RESOLVED   org.eclipse.om2n.binding.service_1.0.0.20151112-1027
25      RESOLVED   org.eclipse.om2n.commons_1.0.0.20151112-1027
26      RESOLVED   org.eclipse.om2n.commons.logging_1.0.0.20151112-1027
27      ACTIVE     org.eclipse.om2n.core_1.0.0.20151112-1027
28      RESOLVED   org.eclipse.om2n.core.service_1.0.0.20151112-1027
29      ACTIVE     org.eclipse.om2n.datanapping.jacob_1.0.0.20151112-1027
30      RESOLVED   org.eclipse.om2n.datanapping.service_1.0.0.20151112-1027
31      RESOLVED   org.eclipse.om2n.interrupting.service_1.0.0.20151112-1027
32      RESOLVED   org.eclipse.om2n.lite.sample_1.0.0.20151112-1027
33      ACTIVE     org.eclipse.om2n.persistence.eclipseLink_1.0.0.20151112-1027
34      RESOLVED   org.eclipse.om2n.persistence.service_1.0.0.20151112-1027
35      ACTIVE     org.eclipse.om2n.uebapp.resourcesbrowser.xml_1.0.0.20151112-1027
osgi> _
  
```

Vous pouvez alors ouvrir votre navigateur à l'adresse suivante pour accéder à OM2M:

<http://127.0.0.1:8080/webpage>



Entrez le nom d'utilisateur "admin" et le mot de passe "admin" pour vous connecter.

Félicitations, vous avez démarré OM2M !

8. Ajout d'un plugin Bluetooth

Pour aller plus loin, il est possible d'ajouter un plug-in permettant de repérer les objets connectés en Bluetooth au projet OM2M. Pour cela, Il faut tout d'abord se positionner dans le dossier du projet "org.eclipse.OM2M" et cloner le code du plugin à l'aide de la ligne de code suivante:

```
git clone https://github.com/semrola/OM2M-bt-ble-zwave.git
```

Sur Eclipse, rafraîchissez alors le projet OM2M en faisant click droit -> refresh.

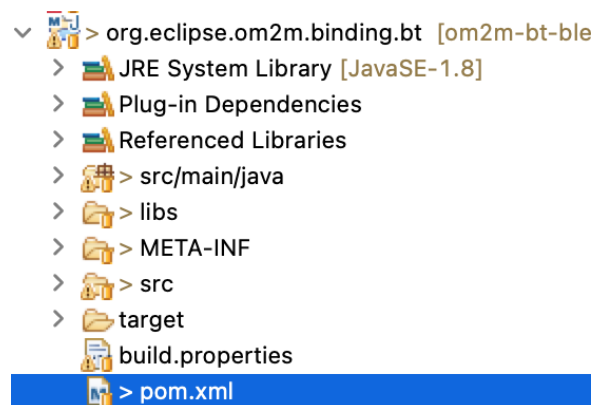
Pensez également à mettre à jour votre projet.

Vous devriez alors avoir trois nouveaux packages dans votre projet.

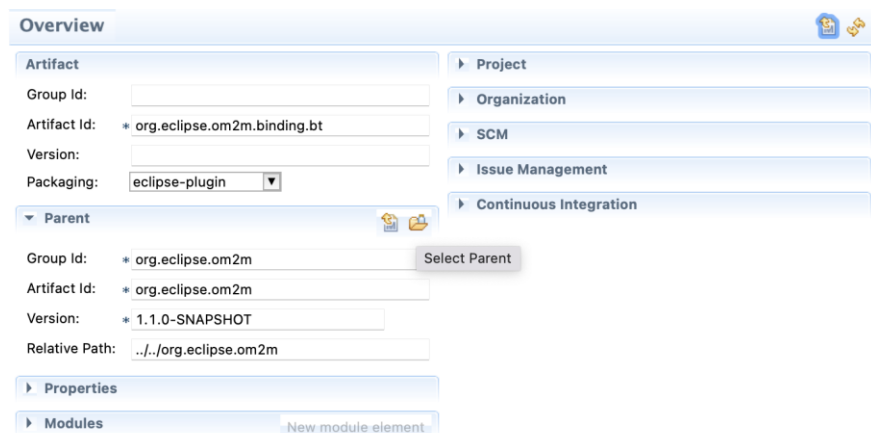
- org.eclipse.OM2M.binding.bt
- org.eclipse.OM2M.binding.ble
- org.eclipse.OM2M.binding.zwave

Dans ce cas-là, nous regardons le package “org.eclipse.OM2M.binding.bt”.

Rendez-vous dans le fichier pom de ce dernier.



Dans la partie “Overview” cliquez sur le petit dossier “select a parent”.



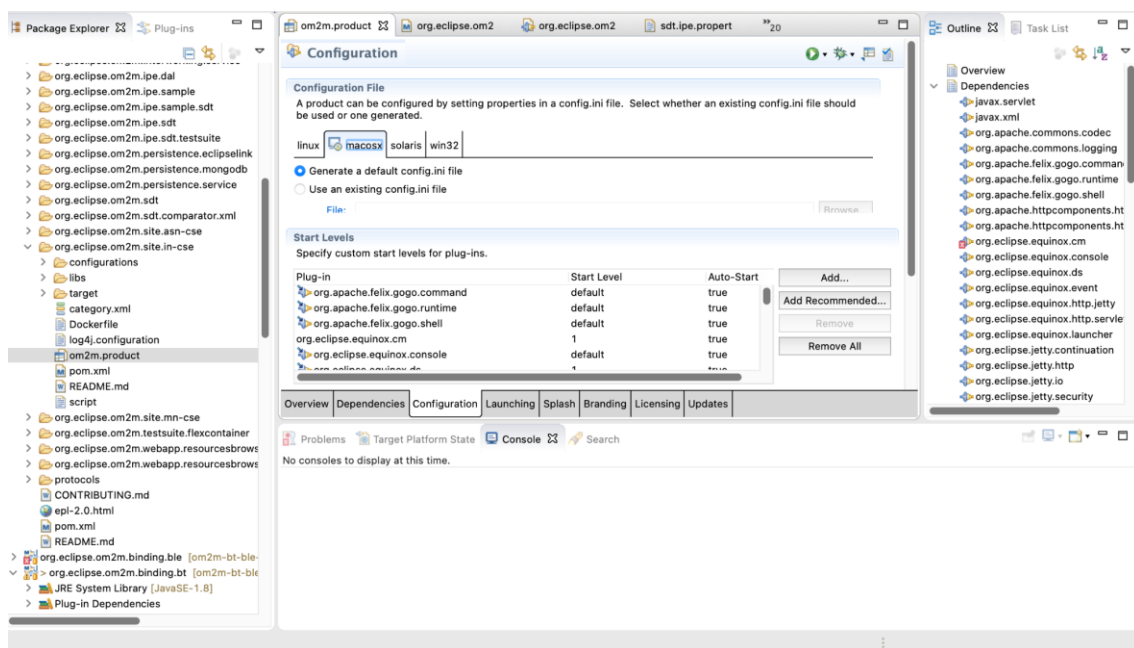
Sélectionnez le projet “org.eclipse.OM2M”, puis faites “Ok”.

Group Id:
 Artifact Id:
 Version:
 Enter groupId, artifactId or sha1 prefix or pattern (*):

 Index downloads are disabled, search results may be incomplete.
 Search Results:

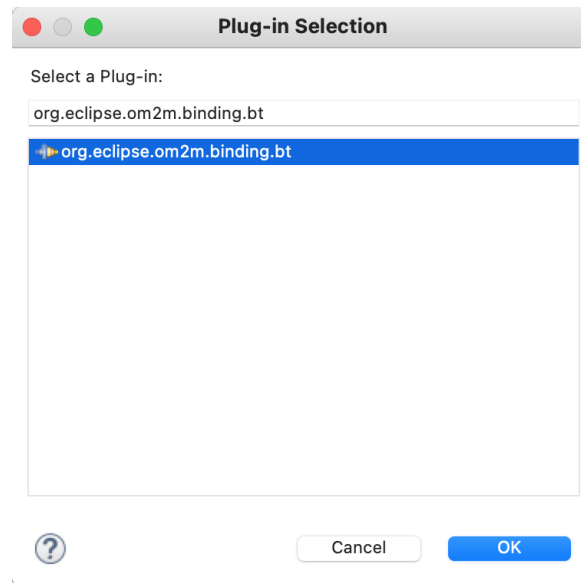
- ☒ org.eclipse.om2m org.eclipse.om2m
- > ☐ org.eclipse.om2m org.eclipse.om2m.protocols.hue
- > ☐ org.eclipse.om2m org.eclipse.om2m.sdt
- > ☐ org.eclipse.om2m org.eclipse.om2m.sdt.home.applications
- > ☐ org.eclipse.om2m protocols

Allez maintenant dans le dossier “org.eclipse.OM2M.site.in-cse” et ouvrez le fichier “OM2M.products”.

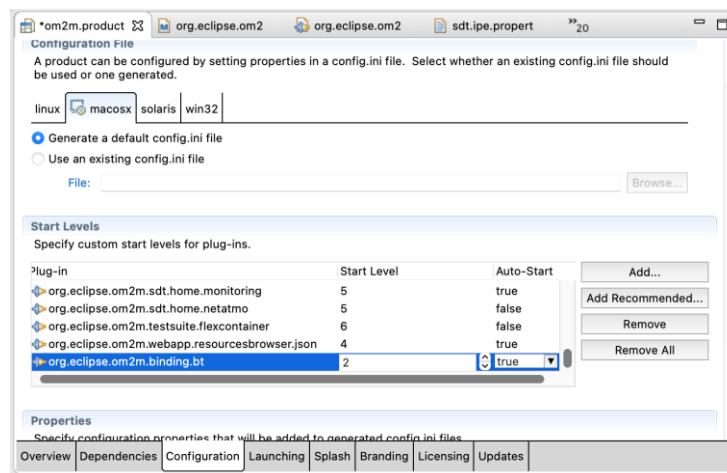


Cliquez sur l’onglet “Dependencies” et dans la partie “Plugins and Fragments” cliquez sur “Add”.

Recherchez le plugin “org.eclipse.OM2M.binding.bt” et une fois sélectionné, faites “Ok”.



Il faut également l'ajouter dans l'onglet "Configuration", dans la partie "Start Levels". Dans "Auto Start" il faut mettre la valeur à "true".



Enfin, vous verrez sûrement des erreurs Java apparaître dans le fichier "RequestSender.java" du dossier "org.eclipse.OM2M.binding.bt". Les erreurs concernent la fonction TargetId. Il faut simplement remplacer cette fonction par SetTo et ce, dans tous le fichier.

```

public static ResponsePrimitive getRequest(String targetId){
    RequestPrimitive request = new RequestPrimitive();
    request.setFrom(Constants.ADMIN_REQUESTING_ENTITY);
    request.setTo(targetId);
    request.setReturnContentType(MimeMediaType.OBJ);
    request.setOperation(Operation.RETRIEVE);
    request.setRequestContentType(MimeMediaType.OBJ);
    return CSE.doRequest(request);
}
  
```

Vous pouvez désormais mettre à jour votre projet et le lancer, votre plugin est installé.

Table des figures

Figure 1 : Architecture d'une API Rest	9
Figure 2 : Standard horizontal.....	11
Figure 3 : Structure système OM2M.....	11
Figure 4 : Architecture globale OM2M.....	12
Figure 5 : Architecture détaillée OM2M	13
Figure 6 : Requête HTTP	13
Figure 7 : Différence entre une cornée normale et une cornée malade	14
Figure 8 : Diagramme de cas d'utilisation de la montre connectée.....	16
Figure 9 : Diagramme de cas d'utilisation des capteurs environnementaux	17
Figure 10 : Architecture du système.....	17
Figure 11 : Architecture type d'une application Fitbit Studio	19
Figure 12 : Code récupérant l'altitude.....	19
Figure 13 : Requête POST	20
Figure 14 : Fitbit Versa 2.....	21
Figure 15 : Quelques images de l'application FitBit	22
Figure 16 : Xiaomi Mi Band 6	23
Figure 17 : Export des données à l'aide de l'application GadgetBridge	23
Figure 18 : Connexion à distance de la Raspberry	24
Figure 19 : Architecture logicielle entre Raspberry Pi et Grove Sensor	25
Figure 20 : Capteur d'humidité et de température DHT11	26
Figure 21 : Envoi des données vers OM2M	27
Figure 22 : Création de l'application MY_Environnement	27
Figure 23 : Descripteur humidité	28
Figure 24 : Descripteur température.....	28
Figure 25 : Hiérarchie MY_Environnement.....	28
Figure 26 : Données de température récupérées	29
Figure 27 : Données d'humidité récupérées.....	29
Figure 28 : Dossier Google Drive	30
Figure 29 : Capture du Trello utilisé	31
Figure 30 : Diagramme de Gantt.....	32