Projet Curam

EI4 AGI - Année 2015/2016

Rapport de projet

PROJET RÉALISÉ PAR :

Antoine Jamin
Pierre Cochard
Julien Monnier
Alexandre Ortiz
Dimitri Robin

PROJET ENCADRÉ PAR :

Mehdi Lhommeau
Jean-Baptiste Fasquel

Engagement de non plagiat

Nous, soussigné Antoine Jamin, Alexandre Ortiz, Pierre Cochard, Julien Monnier, Dimitri Robin, déclarons être pleinement conscient que le plagiat de documents ou d'une partie d'un document publiés sur toutes formes de support, y compris l'internet, constitue une violation des droits d'auteur ainsi qu'une fraude caractérisée. En conséquence, nous nous engageons à citer toutes les sources que nous avons utilisées pour écrire ce rapport ou mémoire.

Signé par les étudiants le 25/04/2016

Remerciements

L'ensemble de l'équipe tient à remercier Mehdi Lhommeau et Jean-Baptiste Fasquel, les responsables du projet, pour leurs conseils, leur soutien et le travail réalisé.

Nous remercions Frank Mercier pour la conception du « boitier ».

Nous remercions Madame Ribbe, ingénieur biomédicale chez Bioparhom pour toutes les informations qu'elle nous a fournis, qui ont contribué au développement de notre projet.

Nous tenons également à remercier le Professeur Georges Leftheriotis du CHU d'Angers, pour le temps qu'il nous a accordé à nous recevoir et pour ses conseils qui ont consacrés à l'avancement et l'état actuel du projet.

Introduction

Dans le cadre de l'année EI4, 2nd année du cycle d'ingénieur de l'ISTIA, les étudiants ont l'opportunité de réaliser un projet sous la responsabilité d'enseignants-chercheurs. La réalisation de ce projet à pour buts :

- Évaluer les étudiants sous plusieurs éléments tels que l'organisation du projet, le travail fournis depuis les connaissances acquises au cours de la formation d'ingénieur ou encore le respect du cahier des charges.
- Permettre aux étudiants d'acquérir de nouvelles connaissances et ainsi obtenir de nouvelles compétences relatives à leur futur métier.

Notre équipe a réalisé un projet en partenariat avec le Cancéropôle du CHU d'Angers, représenté par le Professeur Georges Leftheriotis. Notre sujet portait sur la réalisation d'un objet connecté pour le suivi médical de patients atteint du cancer.

À l'heure actuelle, il existe de nombreuses thérapies pour traiter le cancer. Cependant, tous les patients doivent être suivi régulièrement. C'est dans cette optique que notre projet vise à améliorer l'efficacité du suivi des patients. Le suivi d'un patient peut être quotidien, ce qui implique pour le patient une mobilité pouvant être difficile. L'objet connecté, nommé Curam, permettra à un patient de communiquer ses suivis depuis son domicile vers le médecin chargé de sa thérapie.

1. Organisation du projet

A. Présentation de l'équipe

Nous sommes cinq étudiant en 2nd année du cycle d'ingénieur de l'ISTIA en spécialité génie des systèmes industriels. Nous cinq, sommes en option automatique et génie informatique.

ANTOINE JAMIN

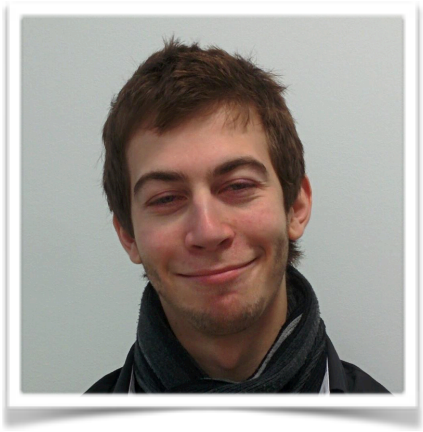
Acquisition des données capteurs
Protocoles de communication (MQTT)
Base de données (MongoDB)

PIERRE COCHARD

Raspberry pi (Installation Raspbian, Réseau)
API REST python (flask)
Site web du projet

JULIEN MONNIER

Acquisition des données capteurs
Base de données (MongoDB)



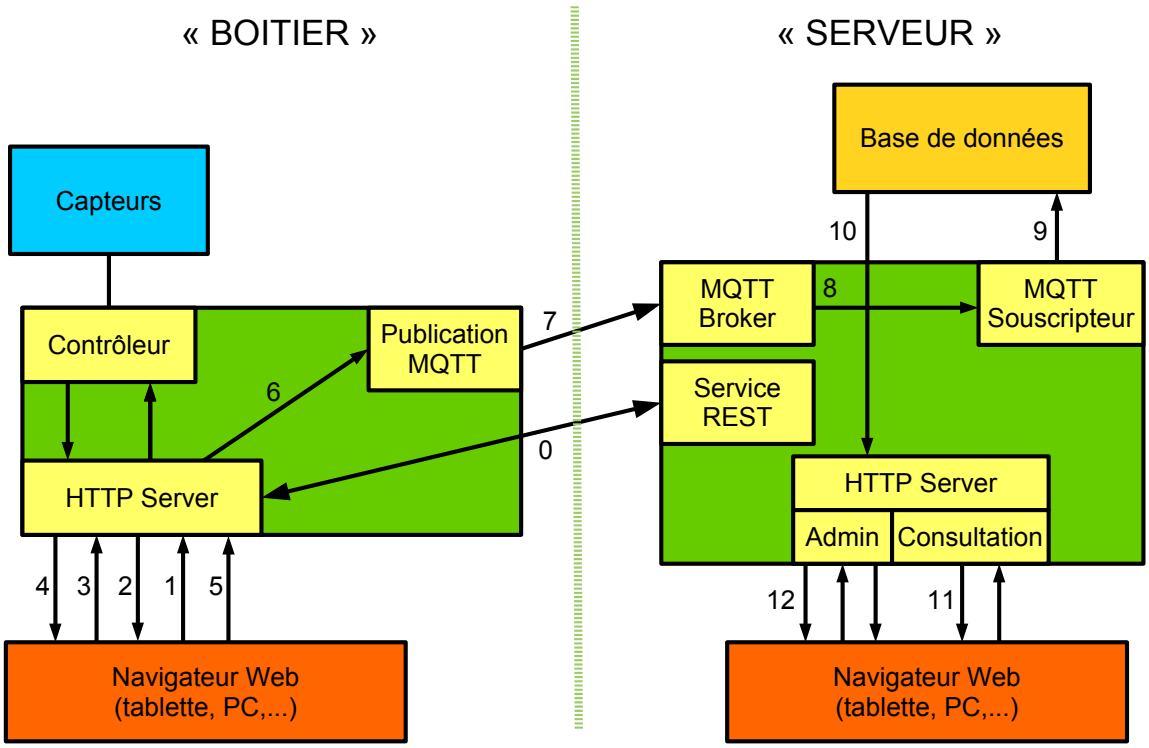
DIMITRI ROBIN

Développement application (NodeJS, JavaScript)

ALEXANDRE ORTIZ

Développement application (NodeJS, JavaScript)
Base de données (MongoDB)

B. Architecture générale

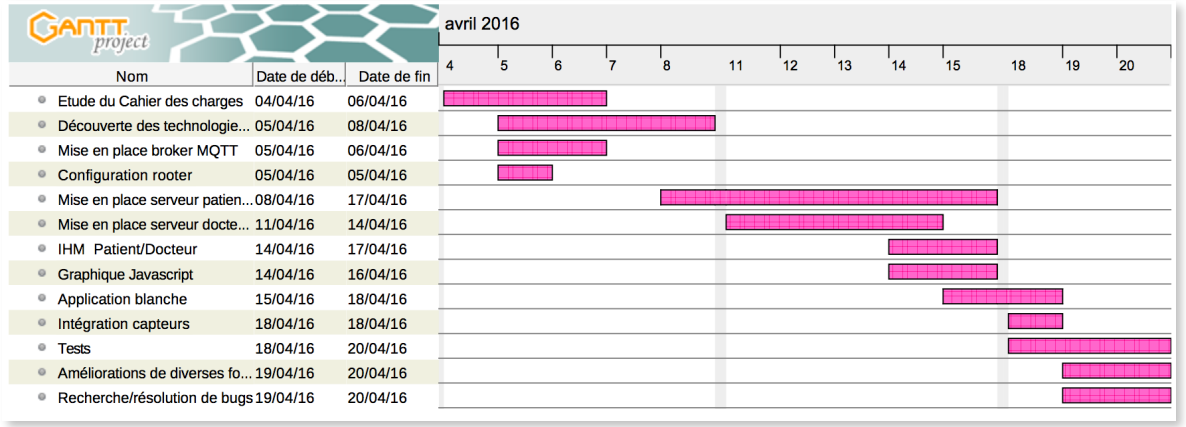


0: Accès à la configuration du questionnaire via service REST
1: Connection au point d'accès Wifi
2: Réception page HTML/JS
3: Requête (AJAX données capteurs) via API REST
4: Retour (réponse requête AJAX) et affichage de la mesure dans la page HTML
5: Validation et envoi de l'ensemble des données (questions + données capteurs)
6: Envoi des informations au « module » de publication selon protocole MQTT
7: Publication effective via l'interface réseau appropriée
8: Récupération des données
9: Sérialisation en base de données
10: Récupération des données par le serveur
11: Consultation par le médecin
12: Administration par le médecin du « compte patient »

C. Gestion du projet

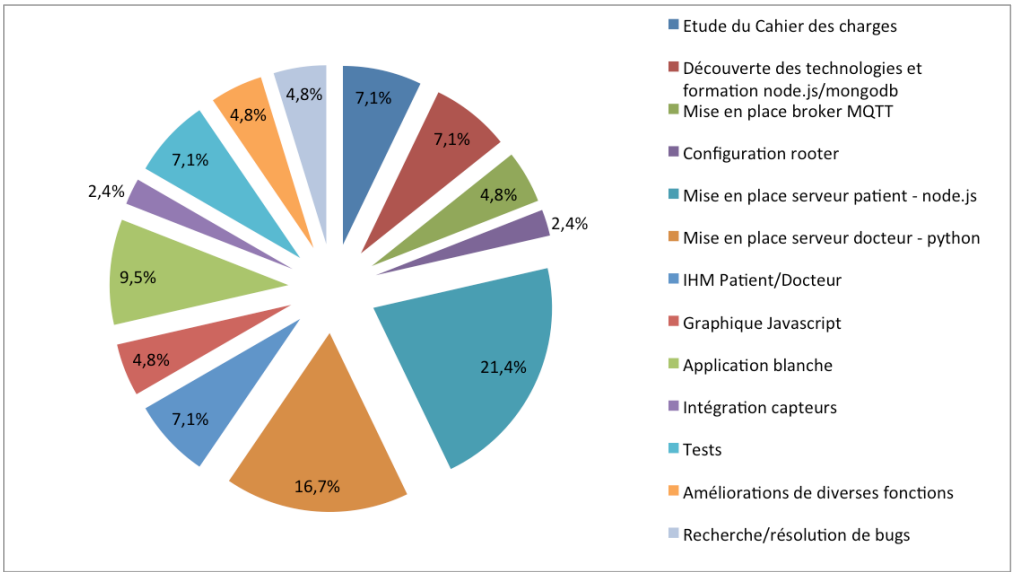
Pour assurer un avancement efficace de notre projet, nous avons organisé notre projet suivant le schéma Scrum (issu de la méthode Agile). Ainsi, au debut de chaque séance nous faisions le résumé de notre avancement et nos objectifs à résoudre pour la séance.

Diagramme de Gantt:



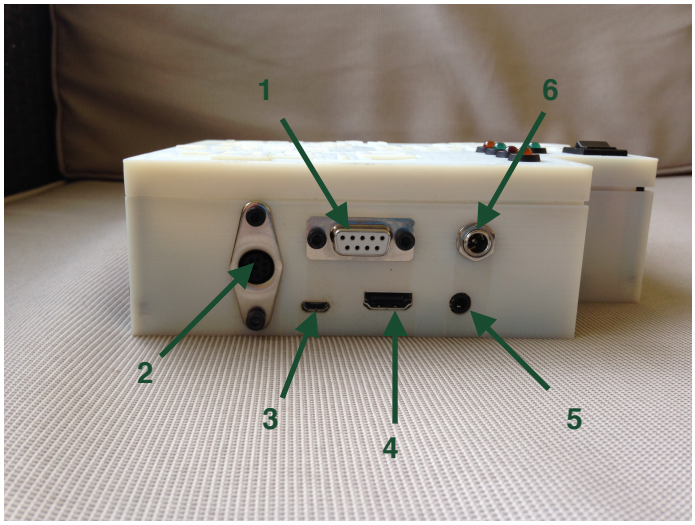
Nom	Date de déb..	Date de fin
Etude du Cahier des charges	04/04/16	06/04/16
Découverte des technologie...	05/04/16	08/04/16
Mise en place broker MQTT	05/04/16	06/04/16
Configuration rooter	05/04/16	05/04/16
Mise en place serveur patien...	08/04/16	17/04/16
Mise en place serveur docte...	11/04/16	14/04/16
IHM Patient/Docteur	14/04/16	17/04/16
Graphique Javascript	14/04/16	16/04/16
Application blanche	15/04/16	18/04/16
Intégration capteurs	18/04/16	18/04/16
Tests	18/04/16	20/04/16
Améliorations de diverses fo...	19/04/16	20/04/16
Recherche/résolution de bugs	19/04/16	20/04/16

Note importante: nos séances réservées pour ce projet était chaque mercredi de la semaine, sur le diagramme ci-dessus nous avons considéré le déroulement du projet sur tout les jours de la semaine, soit du lundi au vendredi.

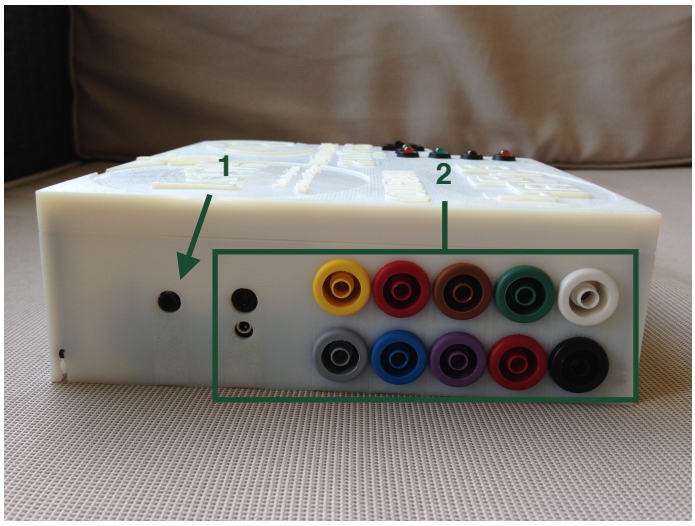


2. Le projet CURAM

A. Composition du boîtier



1: Entrée capteur
2: Entrée capteur
3: Alimentation USB Raspberry Pi
4: Sortie HDMI Raspberry Pi
5: Sortie jack Raspberry Pi
6: Alimentation batterie interne



1: Entrée capteur (température)
2: Entrées capteurs

B. Récupération des données

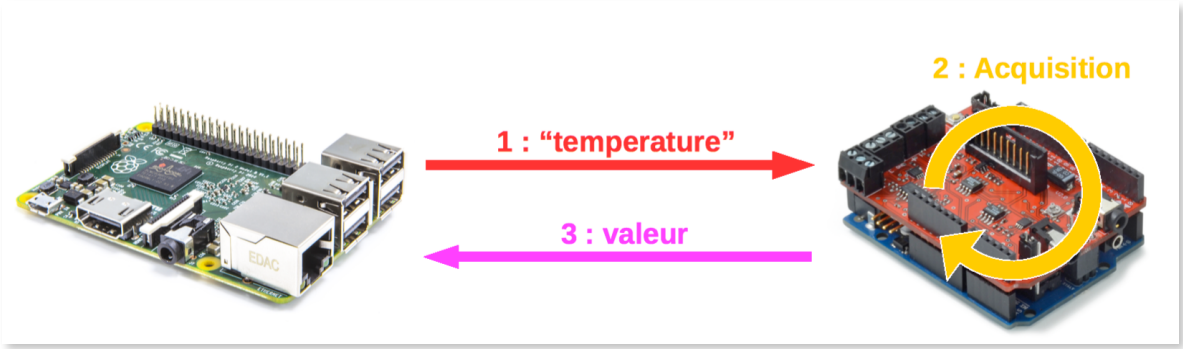
Notre « boîtier » dispose de capteurs permettant d'acquérir des grandeurs physiques relatives à la santé du patient :

- Un capteur de température mesurant la chaleur
- Un impédance-mètre mesurant la masse graisseuse

De plus le « boîtier » possède un shield, permettant d'accueillir de nouveaux capteurs. Pour le moment nous avons seulement développé l'acquisition de température et la masse graisseuse.

1. Le capteur de température

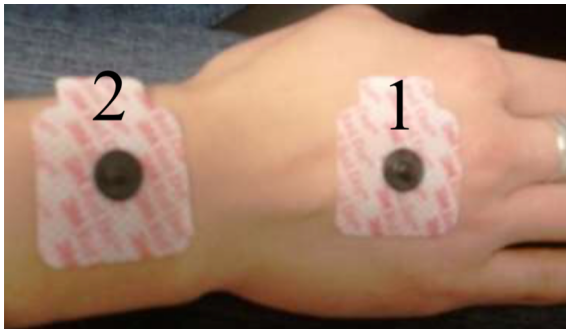
Ce capteur permet de relever la température du corps du patient. Le capteur est relié au « boitier » par le biais d'un Shield Arduino connecté à une carte Arduino UNO elle même connectée en liaison série (USB) au Raspberry Pi.

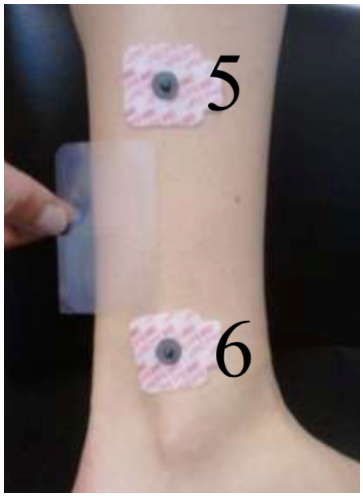


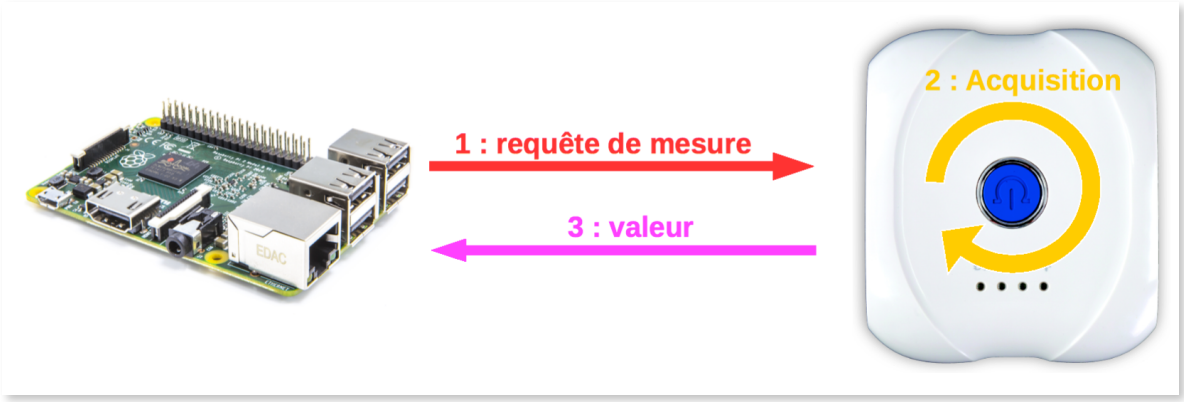
Lorsque le patient clique sur le bouton « Acquisition » l'application envoie la chaine de caractère « temperature » sur la liaison série. A la réception de cette trame la carte Arduino lance l'acquisition. Une boucle de régulation permet d'attendre que la température soit stabilisée avant d'être envoyée à la Raspberry. La valeur reçue est par la suite stockée dans les variables du patient.

2. L'impédance-mètre

L'impédance-mètre à été conçu par la société Bioparhom. Il permet de mesurer la masse grasse d'un corps. Pour ce faire L'impédance-mètre est relier au « boîtier » par USB puis des électrodes doivent être placées sur le corps (voir illustration ci-dessous) puis il génère un courant alternatif avec plusieurs niveaux de fréquences succinctes.







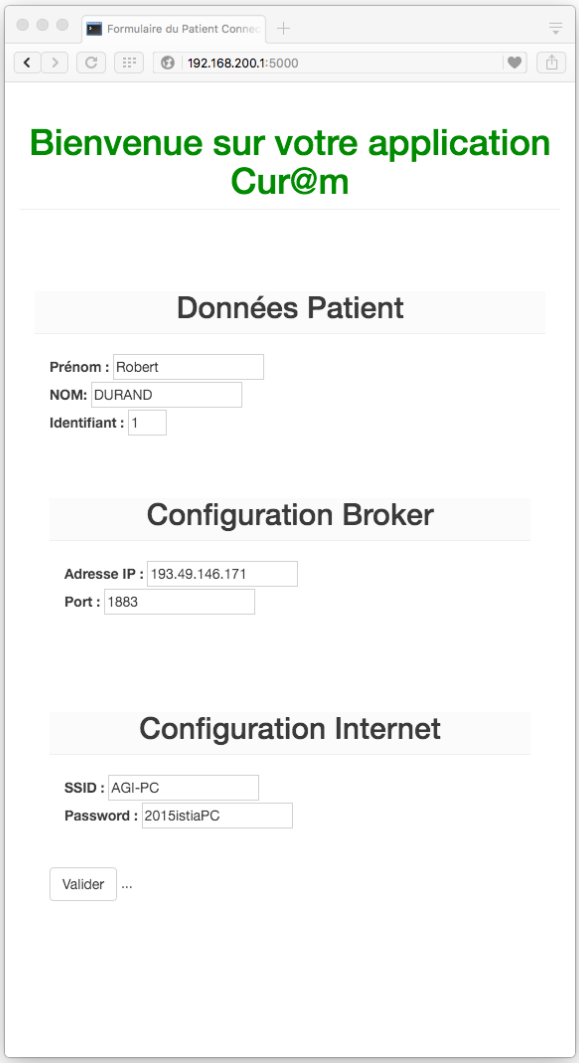
Le calcul de la masse grasse se déroule en deux étapes, dans un premier temps nous devons récupérer la valeur de la réactance et de la résistance pour chaque fréquence (5, 50, 150, 200, 250 et 325 kHz), ensuite à l'aide de ces valeurs, des équations et des données patients la masse grasse est calculée.

3. Interface patient

Un serveur web est installé sur la Raspberry Ri. Nous avons également créé un point d'accès Wi-Fi auquel le patient peut se connecter avec n'importe quel smartphone, tablette ou ordinateur. Ainsi, le patient peut accéder directement au questionnaire fourni par le médecin qui suit son traitement.
Flask est un framework python léger qui fait tourner le serveur web de notre Raspberry Pi. Python permet sur le serveur de récupérer les données du questionnaire, et de les envoyer sous forme de trame JSON au broker dans le cas où la Raspberry Pi est connectée à internet.
Pour fonctionner correctement, la Raspberry Pi est équipée de 2 clés Wi-Fi: une pour se connecter au réseau Internet du patient, l'autre pour émettre un autre réseau Wi-Fi. Il est également possible d'utiliser la connexion filaire de la carte dans le cas où il n'y aurait pas de wifi avec un SSID non caché.

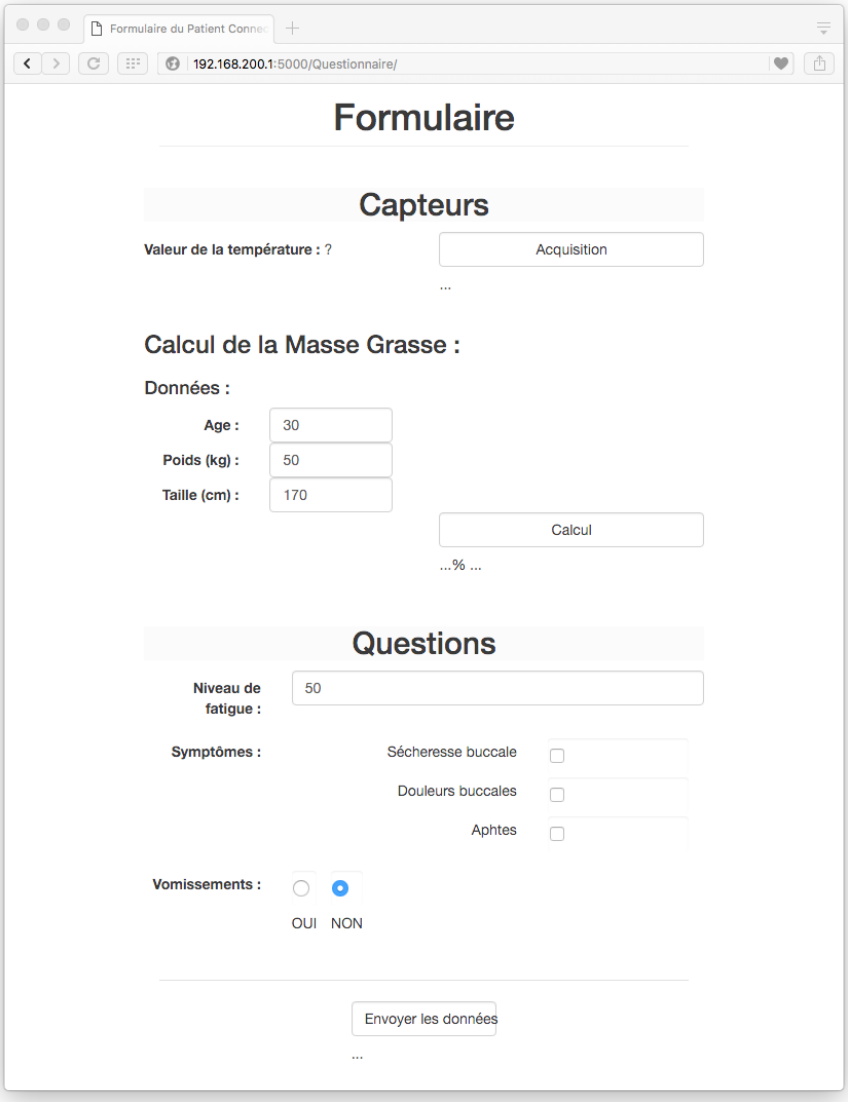
La démarche de fonctionnement se fait comme suit: le patient se connecte au Wi-Fi émit par la Raspberry Pi. Lors de sa première connexion, une page d'authentification s'affiche. Le patient rentre ses informations ainsi que le SSID et le mot de passe de sa box internet, pour que la Raspberry Pi puisse s'y connecter. Enfin, après rafraichissement de la page, le questionnaire s'affiche avec les boutons d'acquisition (température, …).

L'image ci-dessous représente la page d'accueil (route '/') de l'application Flask elle permet de configurer les différents paramètres de l'API. Une fois toutes les informations saisies, l'utilisateur appui sur la touche « Valider » qui envoi une requête Ajax (route '/conf/') permettant de fixer les variables du patient et du Broker MQTT ainsi que de se connecter au réseau WIFI renseigné (à l'aide de la librairie python « wifi »).



L'image ci-après représente la page (route '/Questionnaire/') permettant au patient de récupérer les données des capteurs et de renseigner les différentes parties d'un questionnaire. L'utilisateur appui dans un premier temps sur le bouton « Acquisition » afin de récupérer les données de la température, cet appui envoi une requête Ajax (route '/rTemp/') qui exécute le relevé des température détaillé auparavant. Après

avoir renseigné son âge, son poids et sa taille, le patient clique sur le bouton « Calcul » qui enverra une requête Ajax (route '/rMG/'), celle-ci permettra de récupérer la masse grasse du patient à l'aide du bioimpédancemètre comme détaillé auparavant. Une fois les parties du questionnaire remplies, le patient appui sur le bouton « Envoyer les données » envoyant une requête Ajax (route '/Send/') permettant de se connecter au Broker MQTT et d'envoyer la trame JSON contenant toutes les valeurs du patient, ce protocole sera détaillé par la suite.



C. Transmission des données

1. Protocole MQTT

Message **Q**ueuing **T**elemetry **T**ransport est un protocole basé sur TCP/IP, il permet l'échange de données allant jusqu'à 256Mo par système d'abonnés/publieurs. L'avantage de ce protocole pour notre application, est qu'il assure une communication des données sans avoir besoin d'un réseau haut-débit.

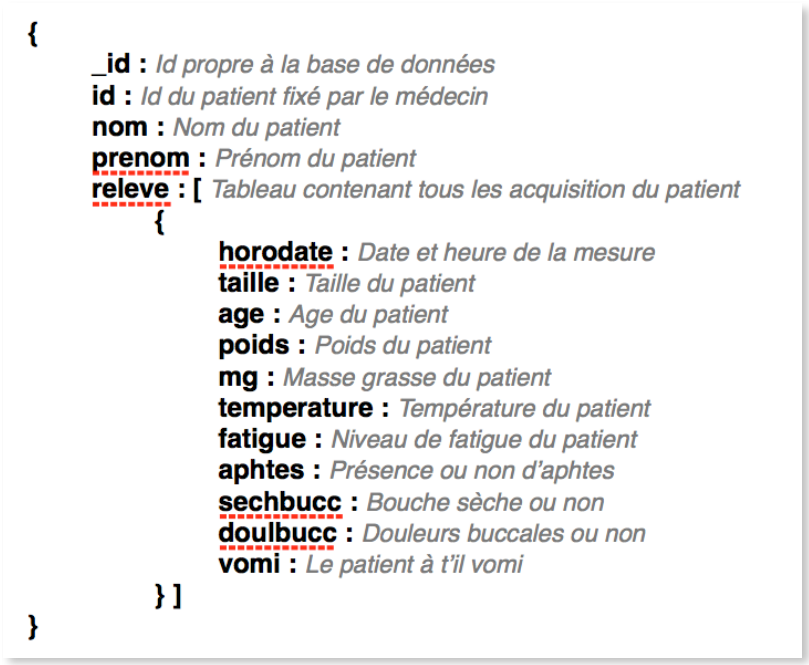
Dans notre application, la box patient rempli le rôle de publieur et le serveur médecin celui d'abonné. Tous ces échanges sont possibles car le Broker (serveur MQTT) est installé sur le serveur médecin.

Lorsque le patient clique sur le bouton « Envoyer les données », l'application se connecte au Broker et publie les données au format JSON dans le Topic « dataPatient/ ». Pour exploiter ce protocole nous avons développé une application python de scrutation exécutée sur le serveur médecin qui s'abonne au Topic et qui ajoute dans la base de données MongoDB les données du patient à chaque fois qu'un message arrivent sur ce Topic.

2. Base de données MongoDB

MongoDB est un système de gestion de base de données NoSQL orientée documents, dans notre application il permettra de stocker au format JSON tous nos patient et leur données dans un seul document.

Les données du patient suivent le schéma le suivant :



Le script python de scrutation ainsi que l'API-rest NodeJS utilisent cette base. Pour ajouter une trame dans la base, l'application Scrutation regarde si le patient est déjà présent dans la base (clef primaire : id), si oui il ajoute une case au tableau relevé avec les nouvelles mesures, sinon il ajoute le patient dans sa globalité. L'API médecin quand à elle, consulte les données de la base, peut supprimer un patient de la base et peut aussi modifier le nom et prénom d'un patient de la base.

D. Exploitation des données

1. NodeJS

Pour ce projet, nous avons choisis d'utiliser le Framework « **NodeJS** », qui permet de faire du JavaScript côté serveur et côté client. Ainsi, nous pouvons grâce à **NodeJS**, exploiter une base de donnée « **MongoDB** » et gérer l'affichage des pages grâce à « **Express** ».

2. Express

Le Framework « **Express** » fournit des outils pour la création d'applications web par exemple. Il permet de gérer facilement les routes, afin de faciliter la navigation sur le site. Il nous permet de coder en quelques lignes un site web efficace sans être trop compliqué.
Exemple de code Express:

```
var express = require('express');

var app = express();

app.get('/', function(req, res) {
    res.setHeader('Content-Type', 'text/plain');
    res.end('Vous êtes à l\'accueil');
});

app.listen(8080);
```

Ci-dessus, un exemple très simple de l'utilisation du Framework « **Express** ». Il va créer une page web d'accueil, et l'application va écouter le port 8080. Il faut installer Express, en faisant un « **npm install express** » pour pouvoir l'utiliser de cette façon.

3. Mongoose

Mongoose est une librairie NodeJS qui permet l'utilisation de la base de données MongoDB. MongoDB est un système de gestion de base de données orientée documents. Son principal intérêt est qu'il utilise le format JSON, sans schéma prédéterminé, nous pouvons donc réécrire le document à volonté facilement. Pour avoir un accès facile en écriture et en lecture sur la base de données, nous utiliserons le programme « **MongoChef** ».

MongoChef

Le format utilisé sera le suivant :

```
{
    "_id" : ObjectId("56e92a6e81fd28bb1b800afe"),
    "id" : NumberInt(1),
    "nom" : "Jamin",
    "prenom" : "Antoine",
    "releve" : {
        "age" : NumberInt(10),
        "fatigue" : NumberInt(28),
        "temperature" : 37.8,
        "sechbucc" : NumberInt(0),
        "doulbucc" : NumberInt(1),
        "aphtes" : NumberInt(1),
        "vomi" : NumberInt(0),
        "horodate" : "02/02/02 10:10",
        "poids" : NumberInt(50),
        "taille" : NumberInt(150),
        "mg" : number(5)
    }
}
```

Format JSON utilisé

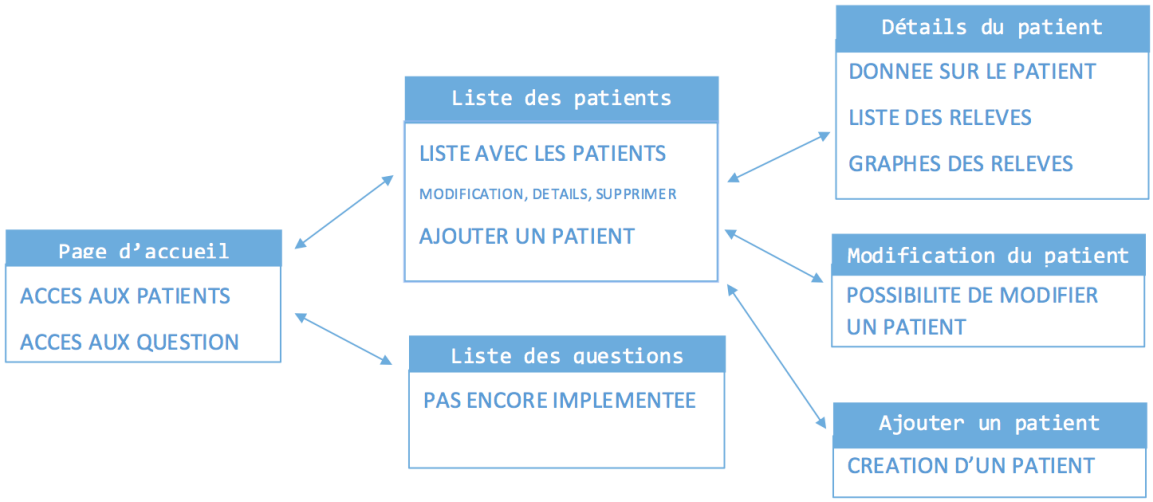
La donnée « _id » est une donnée gérée par défaut par « MongoDB », alors que « id » est une donnée que nous allons utiliser afin de trier nos patients.

Les patients feront des relevés tous les jours donc nous avons choisis d'utiliser un tableau de données. Celui-ci contient toutes les données que le médecin voudra que le patient renseigne.

4. EJS

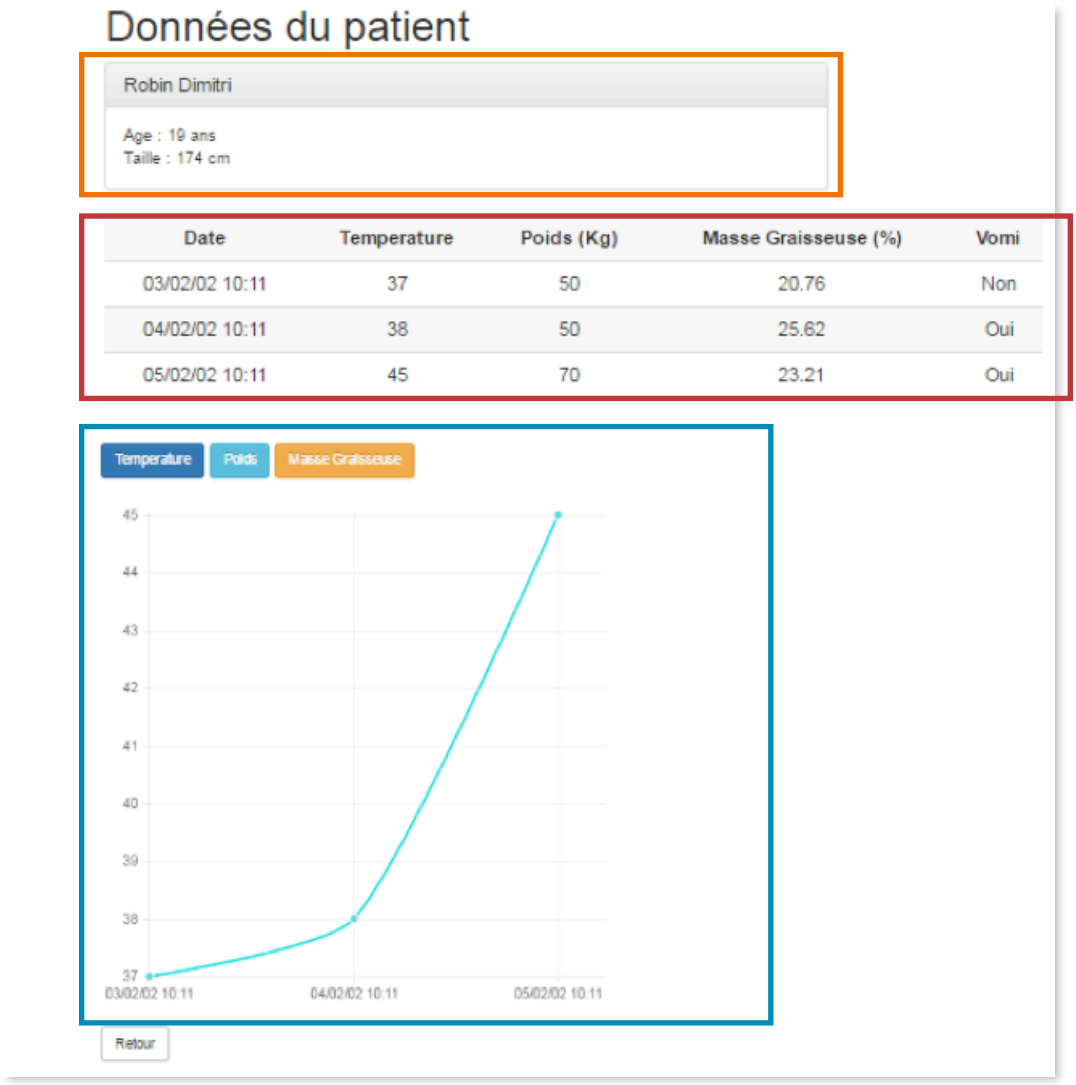
Pour notre application, nous avons opté pour l'utilisation du template EJS au lieu de JADE qui est un format plus difficile à prendre en main. EJS ne nécessite pas l'apprentissage d'un nouveau langage car il peut être écrit en langage HTML, sinon l'apprentissage du JavaScript qui est essentiel pour le NodeJS.

Dans la partie qui va suivre, nous allons expliciter l'architecture de notre site. Et ensuite les modifications que nous pourrions ajouter puisque nous n'avons pas encore fait, telle que la sécurité sur la base de données, la connexion via un login/mot de passe pour le médecin ainsi que pour le patient.



Architecture actuel du site

5. Page de consultation



Dans l'encadré orange l'image ci-dessous, nous avons les informations sur le patient. Elles pourront être plus riches si le médecin a besoin de plus de données (comme le type de cancer, le début du traitement, le temps passé depuis le début du traitement, etc…). L'encadré rouge, nous montre les données issues des relevés effectuées par le patient. Et enfin l'encadré bleu permet de changer la nature du graphique, si nous voulons afficher la température, le poids ou la masse graisseuse dans ce cas.

6. Évolutions possibles

À l'issue des réunions que l'on a assisté au CHU d'angers, nous avons eu des suggestions de la part des médecins et des professeurs afin d'améliorer notre système.

- Système d'authentification login/mot de passe, méthodes déjà possibles avec l'API flask.
- Suivi des patients par visiophonie, à l'image des services déjà existants (Hangouts, Skype, …) pour assurer les acquisitions correctement.
- Ajout de la possibilité de se connecter pour le médecin, dans le cas où plusieurs médecins travaillent avec le système et qu'ils ne voient que leurs patients.
- La possibilité de choisir les questions auxquelles les patients doivent répondre. Dans ce cas il faudra repenser le format JSON vers celui-ci :

```
{
    "_id" : ObjectId("56e92a6e81fd28bb1b800afe"),
    "id" : NumberInt(1),
    "nom" : "Jamin",
    "prenom" : "Antoine",
    "releve" : {
        "age" : NumberInt(10),
        "fatigue" : NumberInt(28),
        "temperature" : 37.8,
        "aphtes" : NumberInt(1),
        "vomi" : NumberInt(0),
        "horodate" : "02/02/02 10:10",
        "poids" : NumberInt(50),
        "taille" : NumberInt(150),
    },
    "question" : [ false, false, true,true, false]
}
```

Format JSON modifié

Ici, nous demanderons au médecin de remplir une page pour assigner les questions aux patients.

Le tableau « question » va servir pour cela, si la donnée est à « **true** » le médecin a besoin de cette donnée sinon non. Ce qui nous permettra de moduler l'apparence du site en fonction de ce tableau. Le tableau représente les données : sécheresse buccale, douleurs buccales, aphtes, vomi et masse graisseuse.

- Amélioration de l'aspect physique afin d'effectuer plus facilement des maintenances ou l'ajout de composant hardware.

7. Conclusion

En entamant le projet, après avoir simplement eu la description du cahier des charges, on se doutait que le projet allait demander un investissement en temps important. Aujourd'hui, nous sommes satisfaits que ces 4 mois de projets aboutissent sur un système qui fonctionne, car il nous aura permis d'appliquer ce que nous avons appris en cours, voire même d'en apprendre plus.

Les démonstrations régulières, autant celle des EI5 que les nôtres, et l'échange avec le CHU ont coupé de la routine habituelle d'un projet. Nous avons ainsi apprécié le fait de travailler avec les docteurs, pour sortir du cadre informatique et s'intéresser à la médecine, ne serait-ce qu'un peu.

De plus, découvrir de nouvelles technologies (nodejs, api rest, flask, …) et des méthodes de travail est très enrichissant personnellement, et le travail en équipe ayant bien fonctionné, nous avons pu avancer rapidement vers les objectifs fixés par le cahier des charges.

Le projet ouvre des perspectives pour les années suivantes, d'ailleurs une partie du groupe est prête à le continuer en EI5. La collaboration étroite avec M. Lhommeau et M. Fasquel est en grande partie la raison de ce choix.

8. Bibliographie

Test simple de communication client/serveur:
http://www.eclipse.org/paho/clients/python/

Exemple de déploiement MQTT:
https://blog.guiguiabloc.fr/index.php/2014/11/13/mqtt-faites-communiquer-vos-objets-simplement/
http://www.hivemq.com/blog/how-to-get-started-with-mqtt

Aide protocole MQTT sur Raspberry Pi
http://www.framboise314.fr/linternet-des-objets-iot-sur-raspberry-pi-avec-mqtt/

Vidéo tutoriel utilisation mosquitto:
https://www.youtube.com/watch?v=SP9Vv3Rksm8

Base de données MongoDB:
https://www.mongodb.org

Aide communication capteurs/MQTT:
https://www.ibm.com/developerworks/websphere/library/techarticles/1106_maynard/1106_maynard.html

Pour installer mosquitto sur Mac:
http://macappstore.org/mosquitto/
http://www.xappsoftware.com/wordpress/2014/10/30/install-mosquitto-on-mac-os-x/

Script Python pour gestion MQTT:
http://air.imag.fr/index.php/Mosquitto
https://www.cloudmqtt.com/docs-python.html

Communication Arduino port serie (USB) en python:
http://www.jujens.eu/posts/2014/Jan/11/communication-serie-facile-python/

Projet réalisé par : Antoine Jamin, Dimitri Robin, Julien Monnier, Pierre Cochard et Alexandre Ortiz
Projet encadré par : Jean-Baptiste Fasquel, Mehdi Lhommeau

RÉSUMÉ

L'avancée des objets connectés est très rapide, malheureusement souvent la médecine n'est pas concernée. Dans le cadre de notre projet, nous devons améliorer une "mallette connectée" commandité par le CHU, qui relie les patients à leur médecin, développée par des EI5. Le patient peut effectuer des relevés à domicile et répondre à un questionnaire sur son état physique. Ces relevés et les réponses au questionnaire sont envoyés sur un serveur et consultables par le médecin depuis le web. Le médecin aura accès à toutes les données sous forme de tableaux ou de graphes. L'affichage, ainsi que les questions sont personnalisables par le médecin.

Mots-clés :Objets Connectés, Raspberry, Médecine, NodeJS, Serveur, Flask, API REST, Python, AJAX,

SUMMARY

The connected object's uprising is very quick, however the medicine isn't often affected. Through our project, we have to improve a "connected briefcase" developped by EI5 students, as we've been asked by the CHU. This briefcase bind patient to their physician. The patient can perform health records and answer a survey about his state. These records and answers are accessible through the web. The physician will access all the datas as tables or charts. The display and the survey is customizable by the physician.

Keywords : Connected object, Raspberry, Medicine, NodeJS, Server, Flask, API REST, Python, AJAX