

**Université de Nice – Sophia Antipolis**

**IUT de Nice – Côte d’Azur**

**Département Informatique**

41 bd. Napoléon III – 06200 Nice

RAPPORT DE STAGE POUR L’OBTENTION DU

DIPLÔME UNIVERSITAIRE DE TECHNOLOGIE

SESSION 2016 – 2017



**Création d’une architecture pour déploiement de services IoT et d’une application dans le cadre de personnes en perte d’autonomie**

Présenté par Sébastien SAEZ

**Université Da Nang**

54 Nguyễn Lương Bằng

Đà Nẵng, Vietnam

Sous la direction de :

**M. VO Trunh Hung** – tuteur entreprise

**Mme. Marie-Agnès Péraldi** -  tutrice IUT

# Remerciement

Je tiens à remercier mon tuteur de stage M. Vo Trung Hung, qui m’a suivi tout au long de ce stage.

Qu’il me soit permis de remercier M. Nguyen Dinh Lam Directeur du Département des Sciences, Technologique et des Relations internationales ainsi que Mme Le Thi Hong Oanh Responsable des relations internationales qui ont su rendre mon séjour agréable.

Mes remerciements s’adressent particulièrement M Nguyen Le Duc Huy Responsable du CNF de Danang pour m’avoir fourni une salle de travail très bien équipée.

Je remercie également Mme Marie-Agnès Peraldi enseignante responsable de mon stage en entreprise pour le soutien et l’aide dont elle a fait preuve à mon égard tout au long de mon stage.

Enfin, je tiens à témoigner toute ma reconnaissance à M. Nhan Le Thanh, chef du département Informatique de l’IUT, pour m’avoir permis de réaliser ce stage au Vietnam et les moyens mis en œuvre pour que notre séjour se passe au mieux.

# Résumé

Pour obtenir un Diplôme Universitaire de Technologie en informatique, un stage de fin de parcours, d’une durée de onze semaines est obligatoire. En ce qui me concerne, je l’ai réalisé du 10 Avril au 23 juin, au sein du laboratoire DNIIT (Danang National Institut of Information & Technology) qui est un institut conjoint entre Université Côte d’Azur et l’Université de Danang

L’objectif de ce stage est de pouvoir tracer les activités quotidiennes d’une personne âgée vivant à domicile seul. Il faut proposer une architecture sécurisée pour ce type d’application qui prend en compte les aspects indispensables de sécurité et confidentialité des données. Un réseau de capteurs (présence, humidité, lumière, température…) est disposé dans le domicile de la personne. Le but de ce projet est de pouvoir tracer une journée normale et selon certains cas qui ont été définis déclencher des alertes. Les alertes et l’activité de la personne sont visibles depuis une interface graphique se situant sur un serveur externe.

# Summary

To obtain a Two-year technical degree in computing, an internship, with a duration of eleven weeks is mandatory. . I have made this internship from April 10th until June 23rd, within the DNIIT institute from UCA and University of Danang

The objective of this internship is to be able to follow the daily activities of an elderly person living alone, at home. It is necessary to propose a secured-architecture for this type of application who takes into account the essential aspects of safety (security) and data privacy. Sensors' network (presence, humidity, light, temperature) are arranged in the place of residence of the person. The purpose of this project is to be able to draw a normal day and according to certain cases which were defined to activate(start) alerts. The alerts and the activity of the person are visible

**Table des matières**

[Remerciement 2](#_Toc485044959)

[Résumé 3](#_Toc485044960)

[Summary 4](#_Toc485044961)

[Introduction 7](#_Toc485044962)

[1.Présentation de l’entreprise 8](#_Toc485044963)

[1.1 DNIIT : UCA et Université de Danang 8](#_Toc485044964)

[1.2 Projets de recherche – MIRE 9](#_Toc485044965)

[1.3 Projet SLEGO 11](#_Toc485044966)

[2. Présentation du stage 12](#_Toc485044969)

[3. Analyse et conception 13](#_Toc485044970)

[3.1 Besoins essentiels et principes associés 13](#_Toc485044972)

[3.2 Besoin complémentaire 14](#_Toc485044973)

[3.3 Conception 14](#_Toc485044974)

[3.3.1 Diagramme de classe 14](#_Toc485044975)

[3.3.2 Architecture matérielle 16](#_Toc485044976)

[3.3.3 Protocole de transmission des données MQTT 18](#_Toc485044977)

[3.3.4 Format des trames MQTT 19](#_Toc485044978)

[3.3.5 Architecture logicielle 20](#_Toc485044979)

[3.3.6 Base de donnée 21](#_Toc485044980)

[4 - Développement 23](#_Toc485044981)

[4.1 Choix des langages 23](#_Toc485044982)

[4.2 Les librairies utilisées 23](#_Toc485044983)

[4.3 Codage du dispacher pour récupération de la trame MQTT 24](#_Toc485044984)

[4.4 Programmation des SmartSensor 25](#_Toc485044985)

[4.5 Programmation du Monitoring Controller et stockage dans la base de donnée 26](#_Toc485044986)

[4.6 Création de l’IHM 27](#_Toc485044987)

[4.7 Déploiement des activités sur Raspberry 28](#_Toc485044988)

[6. Résultat 29](#_Toc485044989)

[6.1 Etat final de l’application par rapport au cahier des charges 29](#_Toc485044990)

[6.2 Difficulté et contraintes 29](#_Toc485044991)

[6.3 Quantification 30](#_Toc485044992)

[6.4 Planning 30](#_Toc485044993)

[Conclusion 31](#_Toc485044994)

[Glossaire 32](#_Toc485044995)

[Bibliographie 33](#_Toc485044996)

[Tables des illustrations 34](#_Toc485044997)

# Introduction

Ce rapport présente les travaux que j’ai réalisés dans le cadre du projet SLEGO qui est a l’institut DNIIT :de l’Université Côte d’Azur et l’Université de Danang au Vietnam) dont la durée s’étend du 10avril 2017 au 27 juin 2017.

Ce projet à pour but de tracer les activités quotidiennes d’une personne âgée vivant seule à domicile en utilisant une architecture sécurisée à base d’un réseau de capteurs qui prend en compte les aspects indispensables de sécurité et confidentialité des données. L’objectif est de permettre le suivi, par sa famille ou une équipe médicale, de la journée d’une personne pour pouvoir éventuellement intervenir rapidement en cas de détection de problème, en générant des alertes auprès des aidants.

Le but du stage est donc de réaliser une implémentation en Java sur les Raspberry Pi, qui implémente des scénarios de suivi des personnes pour répondre aux besoins de cette application. Pour cela il faudra établir une relation entre les réseaux de capteurs déployés au domicile des personnes pour collecter les données les analyser et vérifier l’activité des personnes. Différents scénarios d’activité sont implémentés. Les résultats de l’activité des personnes sont mémorisés dans des bases de données sur un serveur externe où nous pourrons les consulter avec une Interface Homme Machine.

Comme ce projet est collaboratif entre 3 équipes, nous avons organisé ce travail en deux phases. Une première qui consiste à élaborer un cahier des charges pour réaliser cette implémentation en prenant en compte les exigences des autres équipes cad à la fois les éléments de la plate-forme matérielle et des capteurs mais également les contraintes de l’outil (Gemoc Studio) qui permet de décrire les scénarios d’activité. La deuxième phase est une phase de déploiement de services qui permet d’expérimenter la communication des capteurs, le transfert et le stockage des données et enfin de prendre en compte la génération automatique des scénarios à partir de l’outil GEMOC sur la plateforme Raspberry d’exécution.

Ce rapport est constitué d’une présentation de l’entreprise et du contexte du stage. Nous verrons ensuite les phases Analyse et Conception, puis le Développement. Enfin nous réalisons une analyse des résultats obtenus au terme de ce projet.

# 1.Présentation de l’entreprise

## 1.1 DNIIT : UCA et Université de Danang

DNIIT est le résultat de la coopération internationale de recherche scientifique entre les enseignants/chercheurs des deux universités, pour renforcer le développement des activités d'enseignement et de recherche au niveau de l'enseignement supérieur entre L’université Côte d’Azur et l’Université de Danang. .  
  
En conséquence, l'événement a également marqué la création du bureau du projet MIRE - Centre de recherche chargé de la recherche collaborative, de la recherche d'applications, de la création, de l'échange d'informations scientifiques et du transfert de technologie. Nice CAMPUS - une unité de formation et scientifique chargée de la coopération en matière d'enseignement, d'enseignement et de communication dans la communauté de DNIIT.  
  
À partir de 2006, la relation de coopération entre l’Université de Danang et l’Université Nice Sophia (UNS) a subi 11 ans de développement de partenariats durables et a obtenu des résultats remarquables. Les deux parties ont collaboré pour ouvrir la maîtrise des sciences en informatique, gestion des ressources en eau, MBA et systèmes embarqués. Plus de 20 employés de l’Université de Danang étudient en Master et PhD à l’UNS. En moyenne, il y a plus de 10 échanges par an de personnel, de conférenciers et d'étudiants entre les deux universités dans le cadre de divers programmes de coopération. Avec la fonction d'un centre de recherche collaborative, d'échange d'informations scientifiques et de transfert de technologie.

MIRE répond à la création de recherches collaboratives et de projets créatifs liés à des domaines d'application spécifiques, en particulier dans le domaine de la santé et des villes intelligentes.

1. Organiser et créer une plate-forme interactive entre les facultés, les chercheurs, les étudiants et les entreprises pour identifier et répondre aux besoins réels, afin de développer des projets multidisciplinaires et intersectoriels.
2. Organiser les échanges de recherches scientifiques entre les deux universités ainsi que d'autres partenaires pour établir une recherche collaborative, promouvoir le développement de projets de recherche interdisciplinaires.
3. Organiser des publications scientifiques avec des partenaires nationaux et internationaux, par le biais de conférences, de séminaires, de revues numériques, notamment par le biais des archives internationales multidisciplinaires de HAL. Promouvoir le transfert de technologie et le démarrage, en particulier des projets de recherche de MIRE. Construction et développement des projets de MIRE.

L'Institut international de technologie DNIIT est le résultat d'un effort inlassable de coopération et d'échange entre UD et UNS. La naissance de l'Institut international de technologie a officiellement ouvert un nouveau chapitre sur le développement intensif entre les deux universités. Les deux parties reconnaîtront conjointement les activités communes de recherche et de formation. Il s'agit d'une étape importante dans la mise en œuvre du modèle de coopération «sans mur» entre UD et UNS qui a germé depuis 2010.  
  
Avec un engagement fort en faveur de la coopération, en partageant les ressources de trois parties (UD, UNS-UCA et AUF), l'Institut international de technologie DNIT se développera de plus en plus avec de nouveaux projets contribuant à promouvoir la recherche internationale et le transfert de technologie pour UD et ses membres.

# 1.2 Projets de recherche – MIRE

 L'initiative MIRE (Maison de l'Innovation et de la Recherche) est un projet de collaboration de recherche entre l'Université Côte d'Azur (UCA), en France et l'Université de Danang (UD) au Vietnam. MIRE, qui a débuté en 2015, est soutenu financièrement par l'Agence Supérieure de l'Education et de la Recherche pour la Francophonie (Agence Universitaire de la Francophonie - AUF), le Ministère de l'Education Nationale, de l'Enseignement Supérieur et de La Recherche - MENESR), et par le programme UCA-JEDI Idex. Les objectifs de MIRE sont (a) la promotion de projets de recherche collaborative entre UCA et UD, (b) la formation du personnel universitaire de l'UD par la mise en œuvre de thèses de doctorat communes et (c) le soutien des programmes de maîtrise de l'UCA en créant des cours à distance à Danang.

L'objectif de cette session spéciale est de donner un aperçu des résultats obtenus jusqu'ici dans les différents projets développés dans le cadre de MIRE. Ces projets sont:

1. PAPIoT (Plate-forme de surveillance de la position et de l'activité basée sur la technologie IoT), le projet vise à développer, réaliser et tester un dispositif IoT portatif compact pour la surveillance des activités de la vie quotidienne (ADL) des personnes âgées. Le système est basé sur l'utilisation de nouvelles normes sans fil IoT à faible portée à longue portée.
2. EMOTICA (Emotion Capture), le projet vise à réaliser un prototype de système basé sur les techniques d'apprentissage supervisées pour la quantification et la classification des signaux émotionnels.
3. POSCA (Pollution Scan): le projet vise à réaliser un système mobile de détection de pollution basé sur l'utilisation d'un micro-spectromètre et des algorithmes d'apprentissage machine.
4. SLEGO (domaine spécifique du langage pour l'orchestration globale de l'expérience): Une partie du projet vise à déployer les services IoT pour l'expression et le suivi des activités de Daily Livings (ADL) pour une personne ayant une perte d'autonomie. Les ADL sont exprimées avec le langage spécifique au domaine dédié à la modélisation de scénario. Une analyse en ligne / hors ligne permet de détecter un scénario positif / négatif.

# 

# 1.3 Projet SLEGO

La transmission des connaissances et du savoir-faire est un des moteurs principaux du développement dans toutes les dimensions : sociale, économique, technologique, scientifique, etc. La description d’une expérience nécessite la représentation de l’enchaînement de phases spécifiques au domaine applicatif ainsi que des données qui sont soit des informations personnelles ou professionnelles de la personne qui détient cette expérience ou les informations liées à l’expérience elle-même. Dans plusieurs secteurs, tels que la santé, l’économie, la sécurité, ce mélange devient un obstacle, pour tous ceux qui veulent partager leurs expériences mais aussi protéger les données de sa vie personnelle et/ou professionnelle.

SLEGO se veut dépasser cet obstacle en fournissant des supports permettant à chacun de décrire et de transmettre ses expériences avec une séparation claire entre les éléments de l’expérience et les données qu’elle manipule ou qui lui sont associées. Une fois modélisée l’expérience, SLEGO permettra de jouer cette expérience soit en mode simulation dans l’environnement de développement et/ou de jouer en temps réel cette expérience et de réaliser un suivi temps réel de son exécution.

# 2. Présentation du stage

L’objectif principal est la réalisation d’une application de suivi pour des personnes en perte d’autonomie. Cette application va être déployée au domicile des personnes et permettra via des capteurs d’enregistrer les activités des personnes et de remonter des alarmes en cas d’anomalies de comportement

Le projet est découpé est trois sujets de stage avec une collaboration entre quatre étudiants en DUT (deux du département GEII et deux d’Informatique).

La figure 1 donne une vue globale de l’architecture déployée pour réaliser le suivi des personnes. Cette architecture comprend une partie hardware et le développement d’un réseau de capteurs réalisé par deux étudiants en DUT Génie Electrique et Informatique Industrielle. Une seconde partie software avec l’utilisation de l’outil GEMOC studio pour la modélisation de scénarios et ma partie qui sera de déployer ces scénarios dans l’architecture.

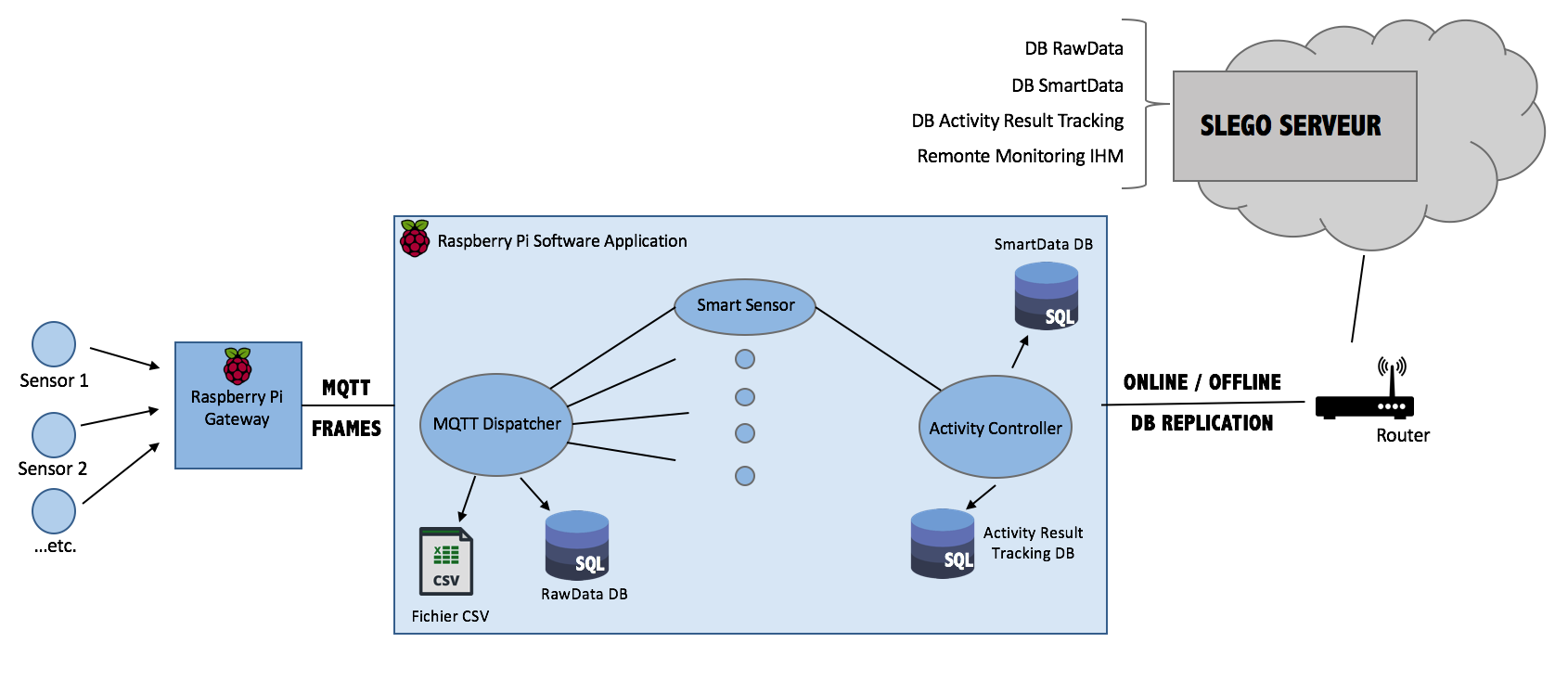


Figure 1 : Schéma représentant l'ensemble de l'architecture du projet

La partie sur laquelle j’ai été affecté consiste à proposer un déploiement automatique des scénarios dans une architecture matérielle comprenant un réseau de capteurs ainsi que deux Raspberry Pi comme indiquer sur la figure 1, en prenant en compte les contraintes du hardware. Les différentes données reçues devront être stockées dans des bases de données (données brutes smart data et résultats d’activité) et dans un fichier CSV. Il faudra faire un traitement des différentes données brutes reçues pour pouvoir les convertir en smart données, qui sont des données facilement compréhensibles par une personne. L’application devra lever des alarmes en fonction des données et des situations.

Enfin, ce programme de suivi et d’aide aux personnes devra être déployé dans une architecture comprenant des capteurs et deux Raspberry Pi.

# 3. Analyse et conception

Cette partie du rapport présente une synthèse de la phase d’analyse et conception ayant été effectué lors de ce stage. Les résultats de cette phase d’analyse et conception ont été utilisés comme contexte de démarrage.

## 3.1 Besoins essentiels et principes associés

Le but de mon projet est donc de réaliser une implémentation en Java sur Raspberry Pi, qui implémente des scénarios de suivi de personne.

La première partie contient les besoins en relation avec le hardware.

BE1 – Acquisition des données capteurs

BE2 – Implémentions de l’application sur Raspberry Pi

BE4 - Utilisation de broker de données MQTT de type publish-subscribe pour la récupération des données

Et la seconde contient les besoins en relation avec l’application

    BE6 - Écriture des traces de capteurs et stockage de celle-ci dans un fichier CSV

    BE11 – Traitement, conversion des données brutes en smartdata

    BE12 – Stockage dans des bases de données interne/externe de l’ensemble des données (rawdata, smartdata, et des résultats de l’activité)

## 3.2 Besoin complémentaire

BC1 – Affichage sur une page web de l’activité de la personne

BC2 - Sécurisation du transfert des données brutes et des résultats

## 3.3 Conception

Pour débuter correctement le projet, il a fallu réaliser plusieurs schémas tels qu’un schéma flux, un schéma d’architecture et un schéma logiciel. Dans cette phase, il a été défini qu’il y aurait 2 cas d’usage. Le premier cas est le suivi d’une activité dite « de prise de douche » de la personne dans sa salle de bain avec la détection de l’entrée dans la pièce, la restitution de l’heure de début et fin de douche, …etc.

Un second cas est le suivi de l’activité de la personne dans la maison avec des capteurs de présence. On pourra donc situer quels ont été les déplacements de la personne dans une journée et mesurer ainsi son niveau d’activité physique.

### 3.3.1 Diagramme de classe

La phase de conception a pour objectif la réalisation d’un modèle structurel de la solution à implémenter. Elle a été basée sur l’implémentation sur Eclipse des différents models se trouvant sur GEMOC (Agent,Port...etc). Pour la conception de ce projet, nous avons choisi de suivre une approche orientée Objet. Ce modèle permet d’établir une architecture constituée de modules autonomes. De surcroît, il admet une importante flexibilité technique, ce qui facilite une implémentation multiplateforme.

La COO (Conception Orientée Objet) permet d’établir les classes à utiliser, ainsi que leurs attributs et méthodes. Les attributs d’un objet décrivent son état, et ses méthodes décrivent son comportement.

Cette architecture conceptuelle du système est représentée avec des diagrammes de classes UML. Ceux-ci permettent de visualiser le schéma de données, notamment les liens entre les classes, ainsi que l’état et le comportement de chaque objet.

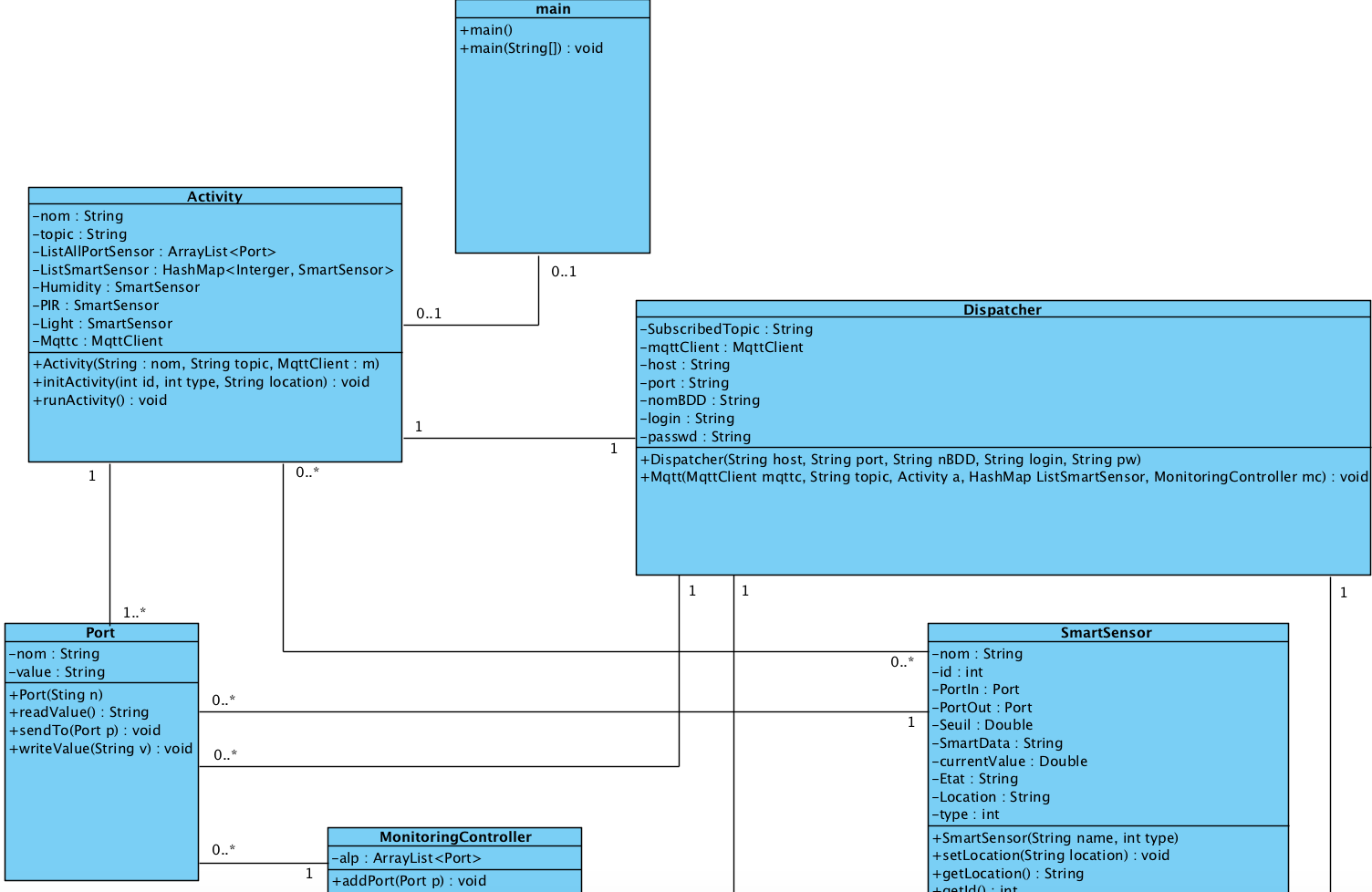
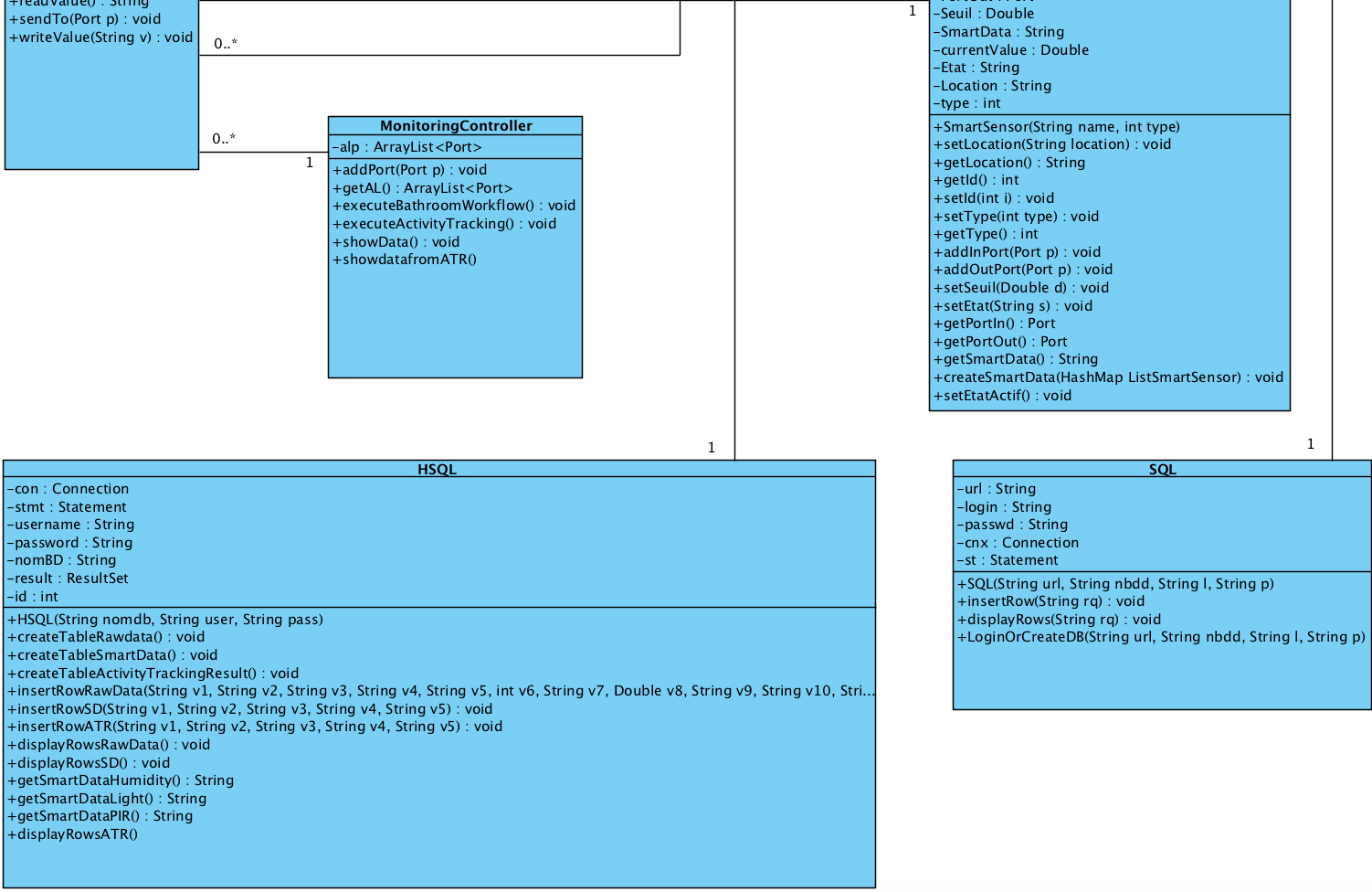


Figure 2 : Diagramme de classe

Dans la figure 2, il est indiqué les différentes classes développées. Premièrement, la classe Activity va permettre de créer une activité ainsi que d’initialiser les différents SmartSensor se trouvant dans la pièce. Le Dispatcher va récupérer la trame MQTT puis extraire les données pour ensuite les envoyer au SmartSensor adapter. La classe SmartSensor va alors traiter la valeur brute qu’elle recevra et créera une nouvelle donnée appelée smartdata (donnée compréhensible par une personne). Cette donnée sera ensuite envoyé à la classe Monitoring Controller. Cette classe va permettre de suivre l'activité de la personne et d'alerter en cas d’anomalie. Par la même occasion elle stocke dans la base de donnée local et externe les smartdata. Pour finir Port va gérer les échanges de données entre les différentes classes. HSQL et SQL permette l’implémentation des bases de données en local et la connexion a une base de données sur le serveur SLEGO.

### 3.3.2 Architecture matérielle

Tout d’abord, un schéma complet de l’architecture matérielle a été réalisé. Sur la figure 3, sont indiqués les différents types de communication ainsi que les différents matériels et capteurs présents dans l’application ainsi que les données échangées entre ces équipements.

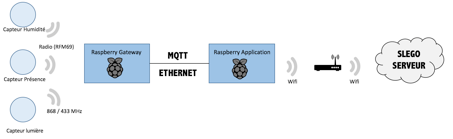
****

Figure 3 : Schéma de l'architecture matérielle

Tout d’abord cette architecture comprend un réseau de capteurs, tel que des capteurs d’humidité, présence, température, lumière, pression...etc. Les capteurs ont été reliés à un premier Raspberry pi appelé “Raspberry Gateway” qui communique avec les capteurs par une liaison radio de fréquence du 868 et 443 MHz. Ce Raspberry s'occupera de récolter les données capteurs et pourra ensuite les envoyer au second Raspberry appelé “Raspberry Application”. Le Raspberry Gateway et le Raspberry Application sont reliés avec un câble Ethernet par lequel des trames au format de protocole MQTT contenant les données capteurs seront envoyées. Pour finir, le Raspberry application s’occupera de traiter les données et de les envoyer sur un serveur externe, dans des bases de données...etc.

Sur ce schéma d’architecture matérielle, nous avons représenté sur la figure 4 l’ensemble des flux de communication entre les équipements et leur type.

Sur la figure 4, en jaune sont indiqués les flux non sécurisés tant dis que les flux sécurisés sont en orange. La transmission de données entre le réseau de capteurs et le Raspberry Gateway est un flux non sécurisé car l’utilisation des radios ne permet pas un accès depuis une grande distance. Le choix d’un flux non sécurisé entre les deux Raspberry a été décidé car le Raspberry Gateway n’est pas connecté à internet. Le Raspberry Application a donc été relié avec lui avec un câble Ethernet pour utiliser un réseau local.

Le flux de communication entre le serveur SLEGO distant et le Raspberry application est sécurisé, dû au fait de la transmission de données confidentielles.

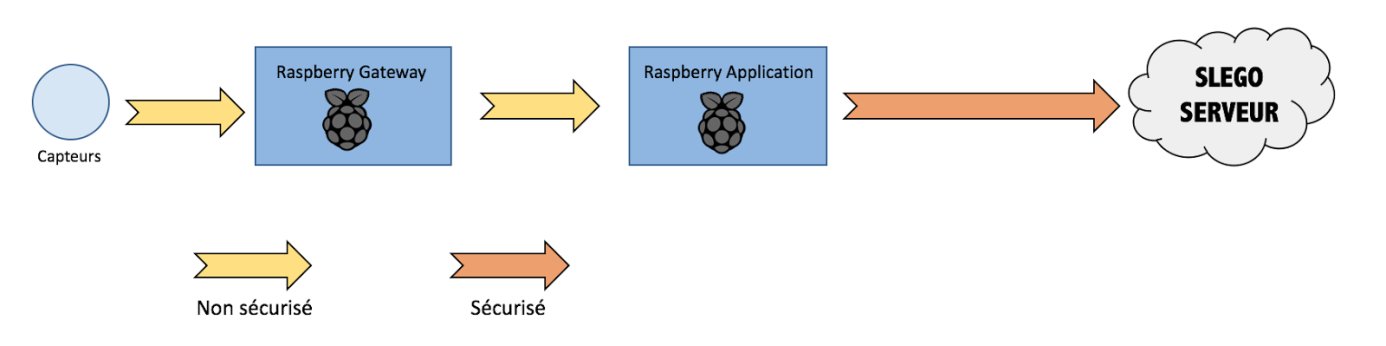
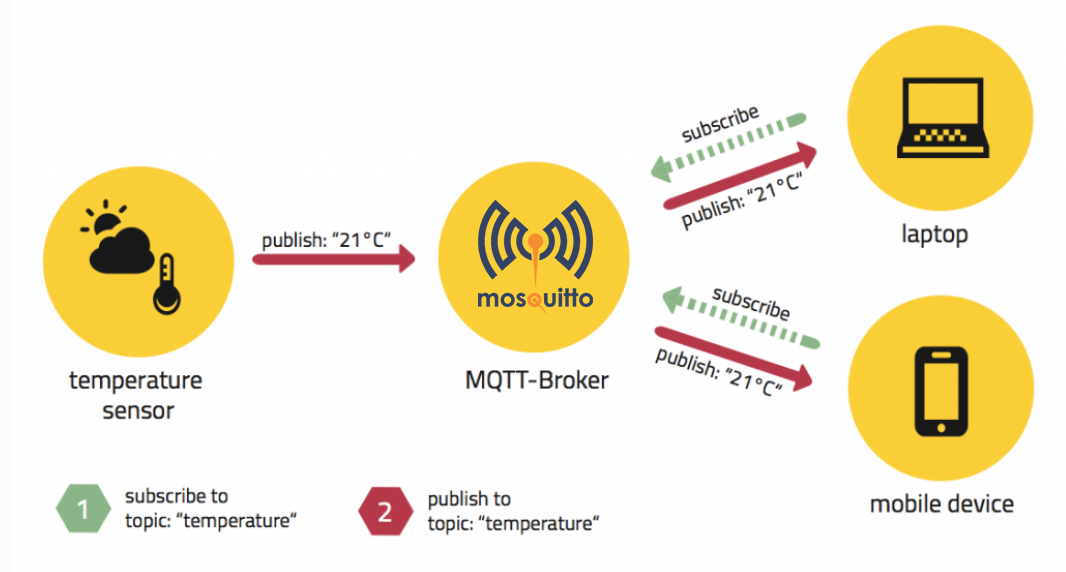


Figure 4 : Schéma de flux

## 3.3.3 Protocole de transmission des données MQTT

Pour la communication entre les deux Raspberry Pi il a été décidé d’utiliser le protocole MQTT, qui est un protocole de « publish-subscribe ».

Le principe du publish subscribe est représenté sur la figure 5. Dans ce protocole, un broker se charge de récolter les informations publiées par des capteurs et de les publier aux équipements qui se sont abonnés (subscribe) à ces données.



##### 

Figure 5 : Schéma du fonctionnement d'un broker MQTT

Le Raspberry Gateway va donc récupérer les données des capteurs et ensuite les publier sur un broker MQTT Mosquitto (broker qui est utilisé dans ce projet) qui se trouve sur le Raspberry Application. L’application va s’abonner à un certain topic pour qui récupère les données qu’elle souhaite.

Les données capteurs sont taguées dans un certains topic (Arborescence dans lequel se trouve les données) ce qui permet qu’elles soient associées à des activités particulière. Cette association se fait de par le format de la trame MQTT pour laquelle nous avons défini un format très particulier.

Sur la figure 5 on peut voire peut voir que la donnée capteur température est envoyé sur le broker dans un topic. Tant dis que l’ordinateur et le téléphone vont s’abonner au topic pour recevoir la valeur de la température.

## 3.3.4 Format des trames MQTT

Une trame MQTT contient les informations suivantes, séparées par des ; pour faciliter la décomposition et l’intégration dans un fichier CSV.

« Version ; UserId ; Timezone ; Date ; Heure ; Samplekind ; IdSensor ; NameSensor ; DataSensor ; UnitSensor ; TypeSensor ; Localisation »

* La Version sera la version de notre trame MQTT (100) puis si modification (XXX),
* L’UserId sera l’identifiant de notre patient,
* La Timezone étant le décalage horaire en UTC,
* La Date,
* L’heure,
* Le Samplekind qui est le type d’enregistrement soit Périodique soit par Interruption,
* L’IdSensor qui est l’identifiant du capteur,
* Le NameSensor qui est le nom du capteur,
* Le DataSensor qui est le donnée brute reçu,
* UnitSensor est l’unité de la valeur du capteur,
* Le type de capteur,
* Sa localisation dans la maison.

Le but de cette étape est de récupérer la trame MQTT d’un capteur spécifique sur son topic, le capteur de présence (PIR) qui est situé dans la douche va envoyer sa donnée dans le topic Home/Bathroom/PIR, le capteur d’humidité va envoyer sa donnée dans le topic Home/Bathroom/Humidity…etc. Pour récupérer plusieurs topics plusieurs opérande son possible comme le + ou le #. Le + va permettre de récupérer tout les topics du niveau suivant, tant dis que le # va récupérer tout les topics.

Home/+/PIR : va permettre de récupérer tout les capteurs PIR qui se trouve dans Bathroom / Livingroom / Room.

Home/Bathroom/# va récupérer tout les topics capteurs qui se trouvent dans la Bathroom comme PIR, Humidity, Light…etc.

### 3.3.5 Architecture logicielle

Pour finir, un schéma de l’architecture logicielle a été conçu. Sur la figure 6 seul le Raspberry Application a été modélisé. Le programme qui est exécuté contient plusieurs parties.

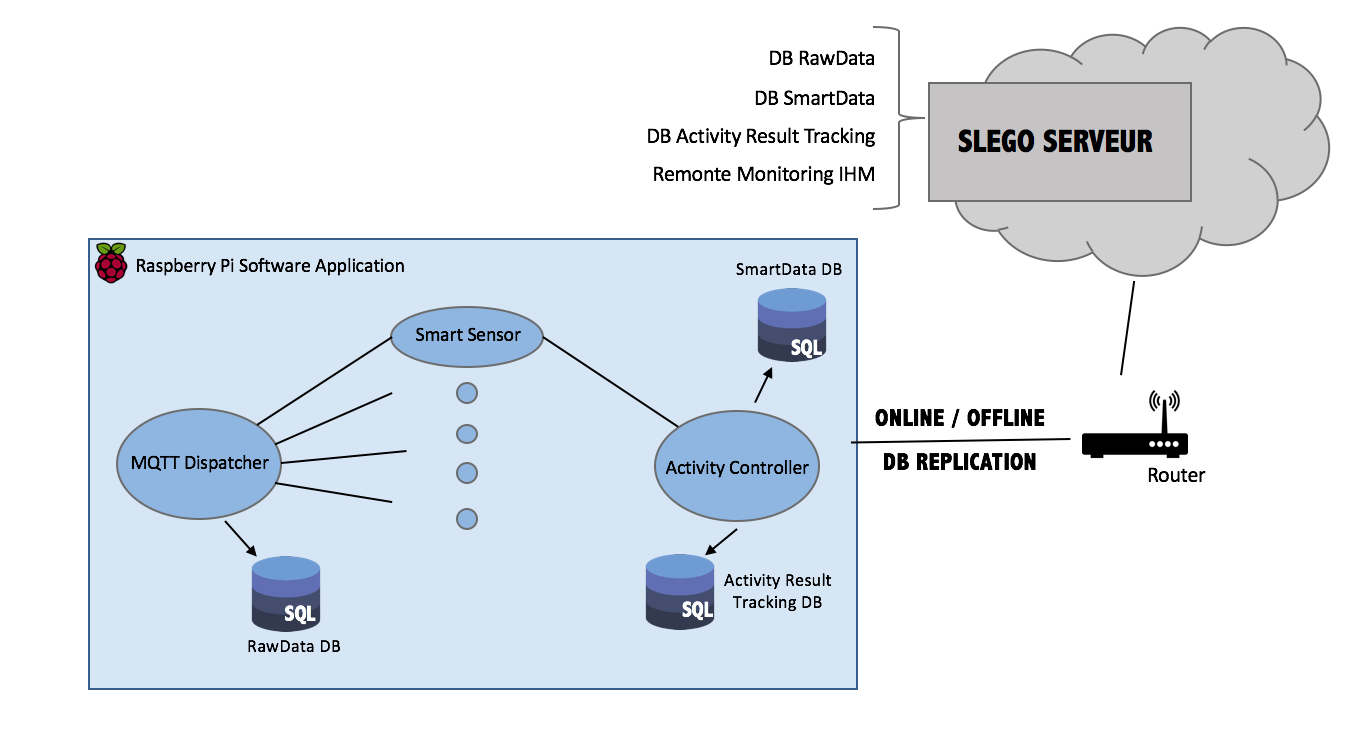
****

Figure 6 : Schéma représentant l'architecture logiciel

Sur cette archi logicielle se trouve :

Le *MQTT Dispatcher* s’occupe de récupérer les données de la trame MQTT et qui le stocke ensuite sur une base de donnée locale qui se situe sur le Raspberry lui-même et sur une base de donnée distante qui se trouve sur le serveur SLEGO. De plus ces données sont envoyées à la partie “SmartSensor” qui va permettre le traitement des données.

Dans la partie comprenant les SmartSensor et le MQTT Dispatcher les données brutes sont converties en donnée facilement compréhensible par une personne, ces données sont appelées les “Smart donnée”. Si le SmartSensor reçoit en entrée une donnée brute tel que l’humidité de la douche est supérieure au seuil défini il créé une smart donnée “L’humidité est élevé”.

Pour Finir, le SmartSensor transmet cette donnée a la dernière partie appelé “Activity Controller” qui gère les situations et émet des alertes. Il s’occupera par la même occasion de stocker dans une base de donnée les smart données dans une table appelé SmartDataDatabase. L’activité de la personne, par exemple si elle est rentrée dans la salle de bain et a prit sa douche, dans la table AcitivityResultTrackingDatabase.

## 3.3.6 Base de donnée

Pour la bonne réalisation de l’application et le suivi des activités, il a été nécessaire de créer une base de donnée comportant 3 tables comme il est montré sur la figure 7.

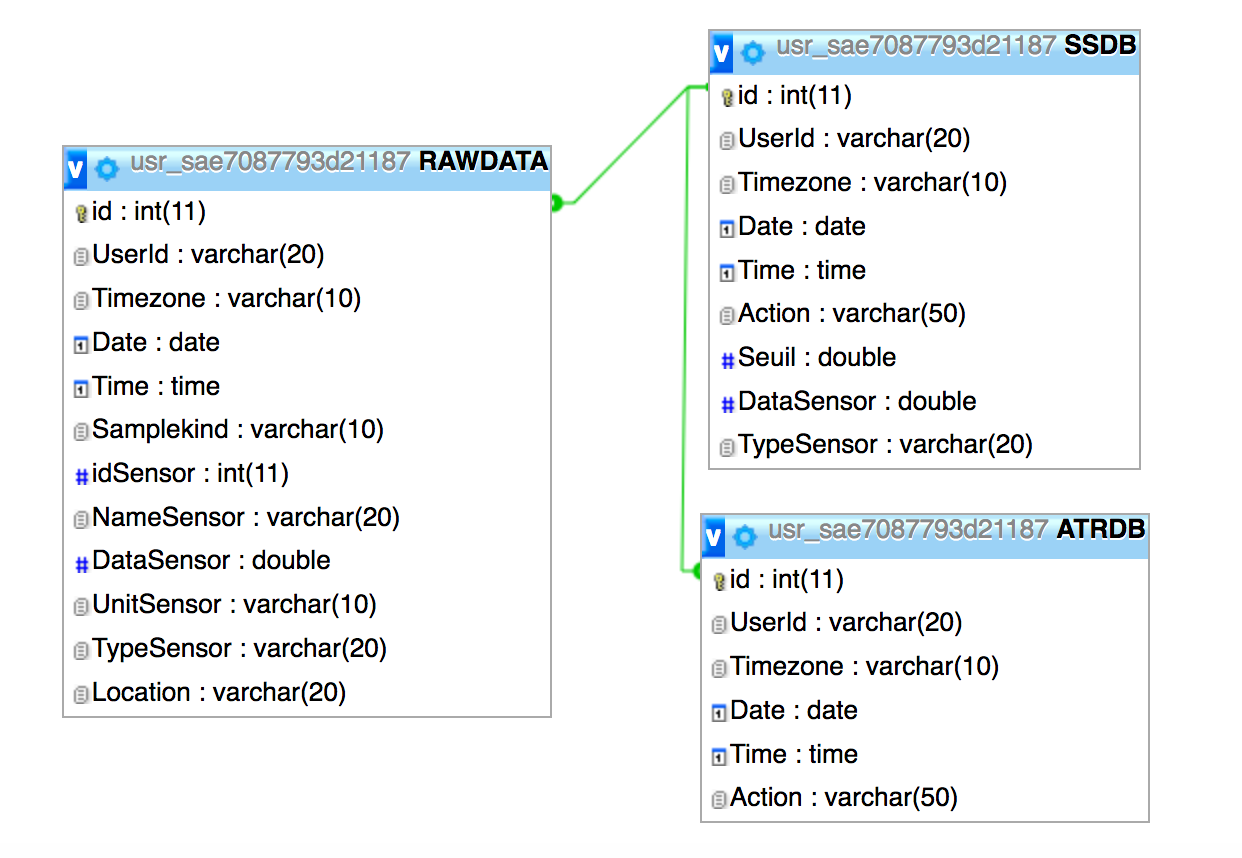


Figure 7 : Schéma relationnel

Tout d’abord, la table RAWDATA va stocker toutes les données brutes reçues des capteurs, chaque enregistrement aura un id qui permettra de faire la relation avec la table SmartData(SSDB). La table RAWDATA reprend l’ensemble des champs de la trame MQTT.

Ensuite, dans la table SmartData(SSDB) sont stocké les données brutes après leur traitement. L’id de la smart data insérer pointe vers l’id de la rawdata convertie. Cette table comprend aussi les champs UserId, Heure, Timzone, date, DateSensor, TypeSensor qui seront complété grâce a l’id de la rawdata correspondante. La colonne seuil permet de savoir à quel moment une alerte doit être levée et dans Action sera indiqué l’activité (Humidité élevé…etc.).

Pour finir, la table ActivityResultTracking stocke où se trouve la personne dans la maison et son comportement (Début de la douche, fin de la douche…etc.). La colonne id est en relation avec l’id de la table SmartData. Ce permet de stocker les smartdata de position et les le résultat de différentes smartdata (Humidité et présence par exemple).

# 

# 4 - Développement

## 4.1 Choix des langages

Le principal langage utilisé dans ce projet est le langage orienté objet Java. Il permet de créer des logiciels compatibles avec de nombreux systèmes d’exploitation (Windows, Linux, Macintosh…etc.). Il a permis de déploiement facile et rapide sur Raspberry Pi

La création de la base de données située sur le serveur SLEGO s’est faite via l’application PHPMyAdmin. Celle-ci offre une gestion des bases de données MySQL intuitive et efficace.

L’interface graphique a été développé en utilisant le HTML et le CSS. L’usage de ce couple de langage permet une création simple et efficace de pages web. La maniabilité et la lisibilité du HTML en font un outil de choix pour le développement des interfaces. Son association avec le CSS permet une gestion des styles graphiques encore plus simplifiées, grâce à la création de feuilles de style.

La liaison entre l’interface graphique et la base de données est assurée grâce au langage PHP. Ce langage permettant une gestion des données.

## 4.2 Les librairies utilisées

Dans ce projet 4 librairies ont été nécessaires pour le bon développement et fonctionnement de l’application :

* **org.eclipse.paho.mqtt.utility** et **org.eclipse.paho.client.view** : Elles nous ont permis de gérer un client MQTT et ainsi de récupérer les trames grâce à leurs méthodes proposées.

* **mysql-connector-java** et **hsqldb** : Avec ces deux librairies il a été possible de gérer les bases de données avec Java, tout d’abord mysql-connector-java a permis de stocker les données sur les bases de données situées sur le serveur SLEGO. Tant dis que hsqldb a permis de créer une base de données interne au raspberry avec un fichier.

## 4.3 Codage du dispacher pour récupération de la trame MQTT

Pour réaliser cette tâche la classe nommée Dispatcher a été créée. Il a fallu rechercher les librairies nécessaires pour le développement et l’utilisation du protocole MQTT. Le choix a été d’utiliser les librairies fournies par Paho (voir ci-dessus).

Deux méthodes principales de la classes “mqttClient” de Paho ont été utilisées. Tout d’abord la méthode “connectionLost()” qui permet de lever une exception si la connexion est interrompue et “messageArrived()” qui va récupérer la trame MTT sous forme de chaine de caractère. Dans un premier temps comme il est indiqué dans la figure 8, lors de la réception du message, un fichier .csv est créé en fonction du topic où se trouve le message. La trame entière sera ensuite écrite dans le fichier CSV.

Ensuite, la trame mqtt reçue (chaîne de caractères) est ensuite découpé en plusieurs chaines permettant ainsi d’identifier chaque champ, tel que la valeur du capteur, son id, son type...etc. Après avoir effectué ce découpage, la liste de smartsensor se trouvant dans l’activité associé au dispatcher sera initialisée. Grâce à cela, il est donc possible d’affecter une valeur capteur au smartsensor adapté pour que le traitement de la valeur soit correct.

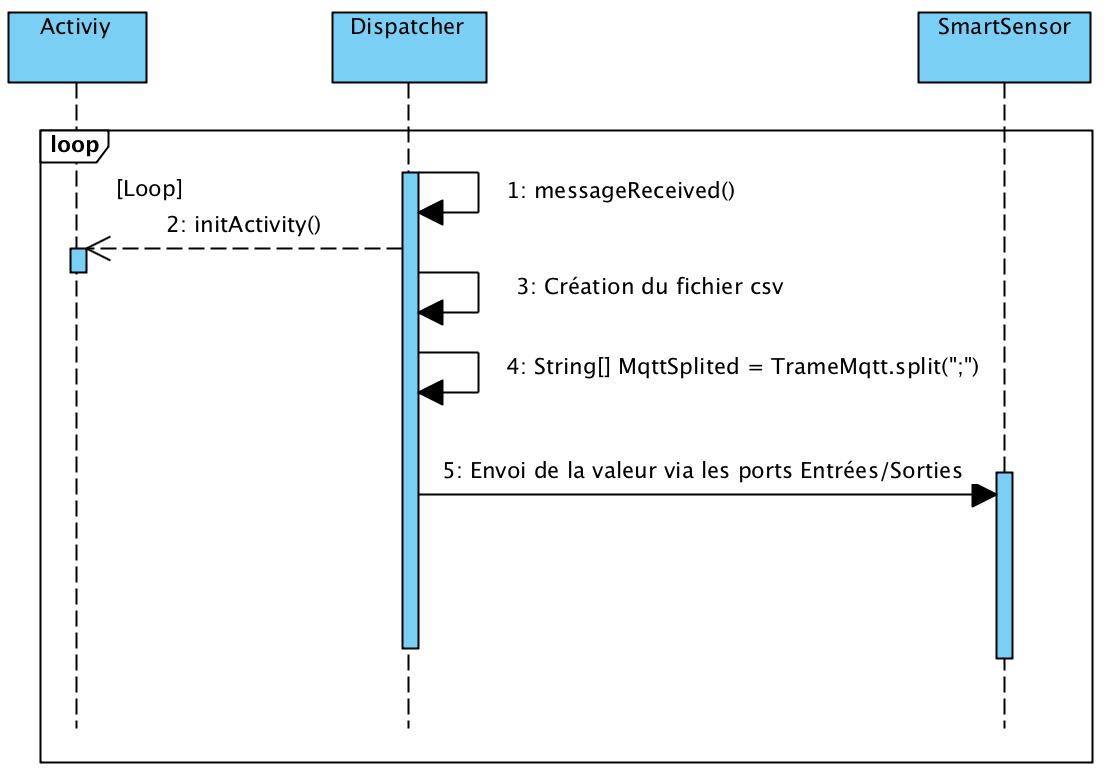


Figure 8 : Diagramme de séquence du Dispatcher

## 4.4 Programmation des SmartSensor

Dans la figure 9, la classe SmartSensor va nous permettre de faire la transition entre les capteurs et une personne en utilisant la méthode “createSmartData()” qui a été créée. Il va lire sur ses ports d’entrée où se trouve la donnée brute du capteur qui a été préalablement envoyé par le Dispatcher via ses ports de sortie. Après avoir récupéré cette valeur il va exécuter le traitement adéquat en relation avec son id, type et location.

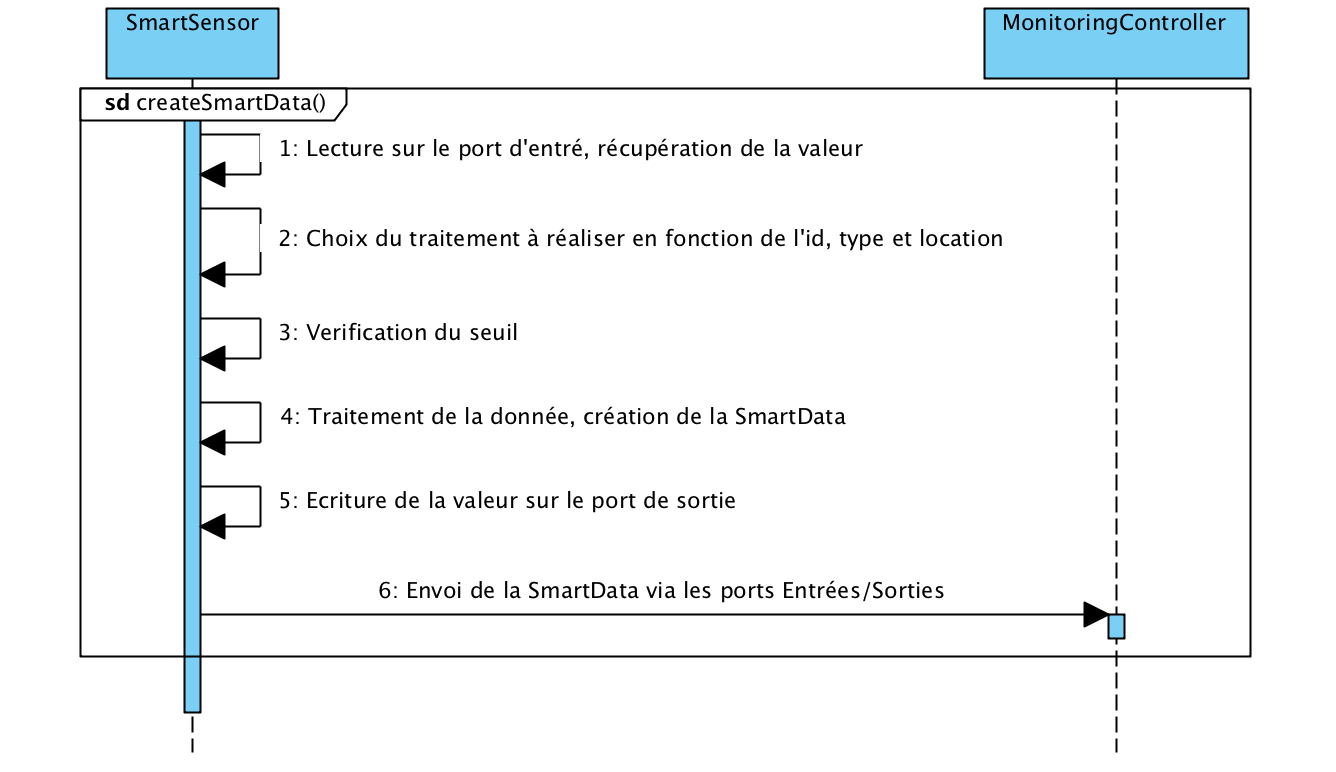


Figure 9 : Diagramme de séquence d'un SmartSensor

Comme le montre la figure 10, chaque SmartSensor a un seuil qui va permettre d'identifier l’activité de la personne. Si le SmartSensor d’humidité reçoit une valeur à son port d’entrée supérieur à son seuil alors il est possible de dire que l’humidité est élevée sinon elle est basse. Ce comportement (humidité élevée...etc.) est alors stocké comme SmartData qui est écrite sur le port de sortie du SmartSensor et est ensuite envoyé à un port d’entrée du MonitoringController.

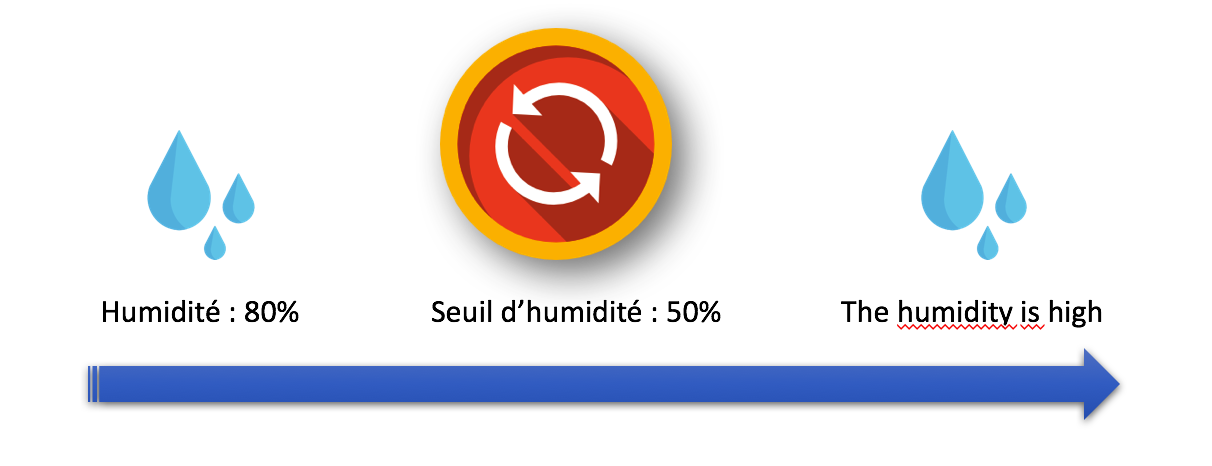


Figure 10 : Schéma représentant la conversion de la donnée

## 

## 4.5 Programmation du Monitoring Controller et stockage dans la base de donnée

Cette classe a pour but de gérer entièrement le programme, elle va permettre de gérer en même temps les smartsensor et ainsi pouvoir détecter une activité concrète telle que le début de la douche d’une personne ou encore la fin de celle-ci.

Sur la figure 11, le MonitoringController va lire sur ses ports d’entrée et vérifier certains cas pour valider une action. Si la personne ne se trouve pas dans la salle de bain mais que l’humidité est élevé alors l’alerte humidité seulement sera exécutée dans le cas contraire si la personne se trouve dans la salle de bain et que l'humidité est élevée alors l’alerte du début de la douche sera exécuté...etc.

Elle va par ailleurs stocker les smartdata reçu sur ses ports d’entrées dans la base de donnée sous la table appelée SSDB et le résultat de la combinaison des smartdata dans la table ATRDB.

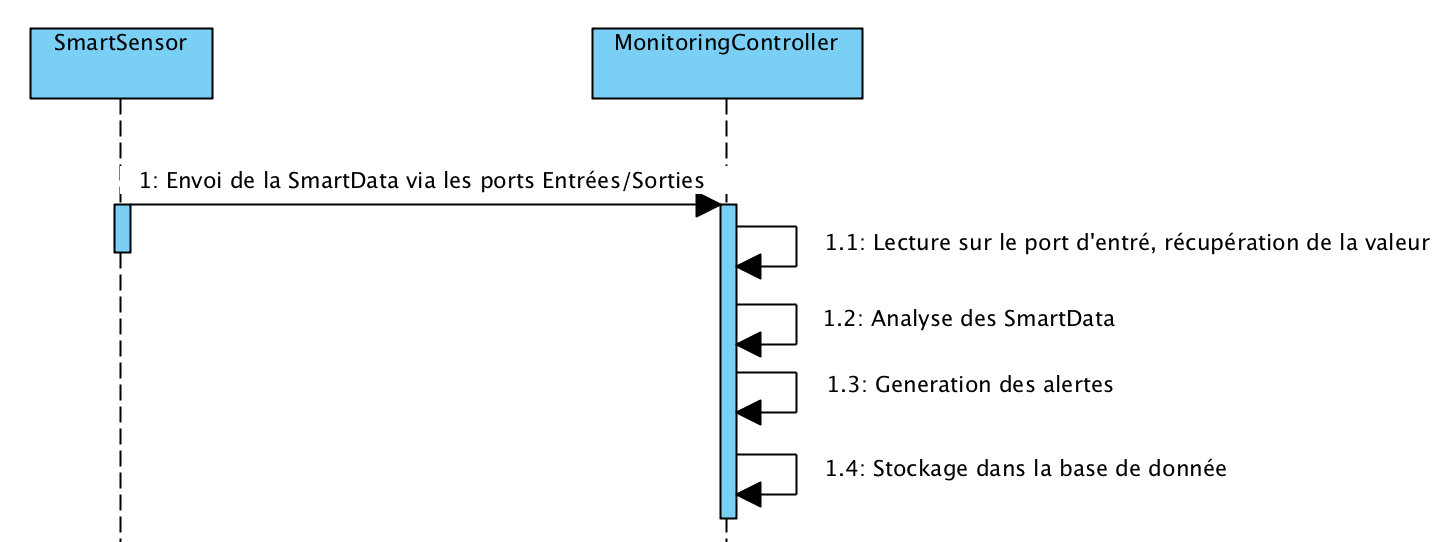


Figure 11 : Diagramme de séquence du MonotoringController

## 4.6 Création de l’IHM

Pour pouvoir consulter les données relatives à une personne, il a été décidé de réaliser une Interface Homme Machine. Comme dit précédemment, les langages HTML et CSS ont été utilisé pour l’aspect graphique de la page tant dis que l’on a utilisé du PHP pour faire la connexion avec la base de données et ainsi pouvoir avoir les informations propres à un utilisateur.

On peut voir sur la figure 12, il est indiqué en temps réel les valeurs brutes des différents capteurs et la dernière l'activité de la personne. Un historique d’activité se trouvant sur la droite de l’interface permet de voir les différentes actions de la personne durant sa journée.

Cette interface permet de faciliter le suivi de la personne en manque d’autonomie par sa famille, médecin...etc.



Figure 12 : Interface web permettant le suivi de la personne

## 4.7 Déploiement des activités sur Raspberry

Dans ce sujet il est demandé de baser notre programme sur deux cas d’usage. Un premier qui a pour but de suivre l’activité de la personne lorsqu’elle se trouve dans la salle de bain et un second qui va s’occuper de localiser la personne dans sa maison au cours de la journée.

Tout d’abord, le programme a été réalisé sur ordinateur pour ensuite être déployé sur un Raspberry Pi. Pour le déploiement, un fichier d’extension .jar a été créé. Pour exécuter l’application, il faudra lancer dans un terminal la commande “java -jar nomdufichier.jar” suivi de l’adresse IP du Raspberry Application pour faire communiquer avec Raspberry Gateway comme indiqué sur la figure 13.

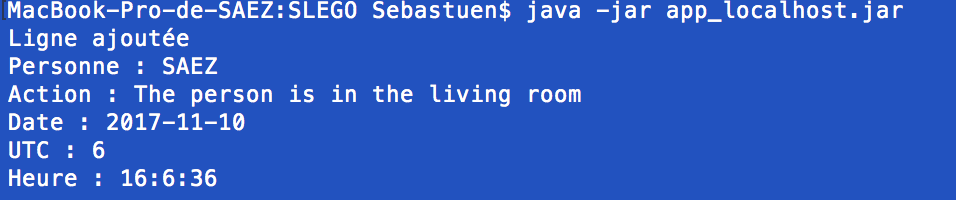


Figure 13 : Application fonctionnant sur Raspberry Pi

# 6. Résultat

## 6.1 Etat final de l’application par rapport au cahier des charges

En reprenant le cahier des charges initiales, les parties suivantes ont été réalisées et finalisées :

- Récupération des trames MQTT

- Création de la base de donnée

- Interprétation des données avec les SmartSensors

- Suivi d’une personne en fonction des 2 scénarios demandés

- Création d’une IHM

L’application a entièrement été réalisée et finalisées

## 6.2 Difficulté et contraintes

Différentes difficultés ont été rencontrées durant ce stage

Durant ce stage l’équipe du projet SLEGO a surtout travaillé en autonomie.

Pour finir, l’utilisation de Gemoc étant compliqué à cause des différents problèmes de librairies. Il a été décider de changer d’environnement de développement en ré implémentant les différents models de Gemoc. Ensuite les données devaient d’abord être enregistrées dans la base de données se trouvant sur le Raspberry et ensuite le MonitoringController devant répliquer cette base sur le serveur externe. Cependant, le processus de réplication prenant beaucoup trop de temps car il doit vérifier que les enregistrements n’existaient pas déjà et ensuite stocker les nouveaux. Cette partie prenait beaucoup trop de temps et ralentissait le programme. Il a alors été décidé de stocker sur le serveur SLEGO en même temps que le stockage dans la base de donnée local du Raspberry s'exécutent, cela a grandement réduit le temps d’exécution.

## 6.3 Quantification

Lors du développement de l’application des quantifications sur le code sources ont été établies.

Environ 1700 lignes de code au total comprenant environ 1300 en JAVA et 400 en HTML, CSS et PHP lors du développement de l’IHM. En plus de ces lignes environ 200 lignes de commentaires ont été écrites. L’application comprend 8 classes et 49 méthodes

|  |  |  |
| --- | --- | --- |
| Nombres de lignes | | |
| JAVA | HTML,CSS, PHP | Commentaire | Nombres de classes | Nombre de méthodes |
| 1274 | 386 | 192 | 8 | 49 |

Figure 14 : Quantification du travail

## 6.4 Planning

Suite à ces 10 semaines de stage, un planning réel peut être réalisé. Il ne diffère presque pas du planning prévisionnel malgré la modification du cahier des charges initialement rédigées.

Sur le planning effectif conçu, figure 15. Le projet a tout d’abord commencé avec une prise en main rapide de Gemoc. La semaine suivante, le cahier des charges a été réalisé. La phase d’analyse et de conception a été réaliser durant les semaines suivant le cahier des charges pour ensuite aboutir à la phase de développement comprenant aussi les tests. Pour finir les deux dernières on permit le déploiement de l’application sur Raspberry et la préparation pour la soutenance. Le rapport de stage a été écrit au fur et à mesure du projet.

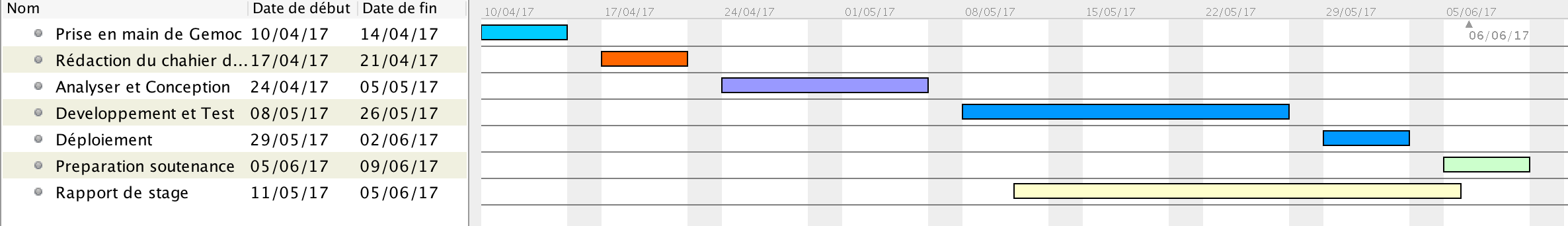


Figure 15 : Planning effectif

# Conclusion

Ce stage d’obtention du DUT, effectué au sein du groupe de recherche SLEGO a consisté à développer une application de suivi de personne isolée déployé sur Raspberry PI. Il fallait créer cette application en accord avec le langage Gemoc.

Tout au long de ce projet, des modifications du cahier des charges ont dû être prise en compte. Elles concernent la modification des champs de la trame MQTT ainsi que les modifications sur la base de donnée et des différents champs.

Le langage Java a été utilisé après l’analyse pour implémenter les différents models de Gemoc et faciliter le déploiement sur le Raspberry Pi.

Par la suite ce projet devrait être récupéré par le laboratoire de recherche MIRE et sera ensuite utilisé pour des personnes avec un manque d’autonomie surveillé par un médecin ou une autre personne. Sachant que se projet fait partie du domaine de la e-santé, il sera probablement l’objet de différentes évolutions.

Ce stage m’a permis de découvrir le milieu professionnel et le système embarqué en travaillant avec des étudiants d’une filière différente de la mienne. Cela ma fait découvrir une partie de l’informatique plus orientée électronique et matérielle. Il a fallu faire preuve d’autonomie et apprendre en autoformation les différentes méthodes de développement.

C’est une expérience très enrichissante qui m’a permis d’acquérir de nombreuses connaissances complémentaires à ma formation.

# Glossaire

**Gemoc**: Environnement de développement. Le Studio GEMOC fournit des composants génériques à travers les technologies Eclipse pour le développement, l'intégration et l'utilisation de langages de modélisation exécutables hétérogènes.

**CSV :** Fichier [tableur](http://www.commentcamarche.net/contents/1103-introduction-a-la-notion-de-tableur), contenant des données sur chaque ligne séparés par un caractère de séparation (généralement une virgule, un point-virgule ou une tabulation).

**Eclipse**: Environnement de développement utilisé pour ce projet

**Java**: Langage de programmation orienté objet

**Models**: Langage de modélisation dans gemoc

**MQTT**: Protocole de messagerie publish-subscribe basé sur le protocole TCP/IP.

**Object**: Approche de modélisation informatique

**MySQL**: Type de SGBD libre ou propriétaire faisant partie des plus utilisés dans le monde.

**Paho**: Le projet Eclipse Paho fournit des implémentations client open-source de protocoles de messagerie MQTT et MQTT-SN destinés à des applications nouvelles, existantes et émergentes pour Internet of Things.

**PHPMyAdmin**: Outil permettant la gestion graphique d’une base de données MySQL.

**SQL**: Langage normalisé qui sert à effectuer des opérations sur des bases de données.

**Raspberry** **Pi**: C’est un nano-ordinateur mono carte, qui est destiné a encourager l’apprentissage de la programmation informatique

**UML**: Unified Modeling Language Langage de modélisation permettant de représenter visuellement la conception d’un système

# Bibliographie

**Hivemq.com - Fonctionnement MQTT**

[**http://www.hivemq.com/blog/how-to-get-started-with-mqtt**](http://www.hivemq.com/blog/how-to-get-started-with-mqtt)

**Mosquitto.org - Installation, utilisation broker Mosquitto**

[**https://mosquitto.org/download/**](https://mosquitto.org/download/)

**Projetsdiy – Installation, utilisation Mosquitto Raspberry**

[**https://projetsdiy.fr/mosquitto-broker-mqtt-raspberry-pi/#.WTE06smkLR0**](https://projetsdiy.fr/mosquitto-broker-mqtt-raspberry-pi/#.WTE06smkLR0)

**Commentcamarche.net – Création fichier CSV**

[**http://codes-sources.commentcamarche.net/forum/affich-136644-creer-un-fichier-texte-avec-createnewfile**](http://codes-sources.commentcamarche.net/forum/affich-136644-creer-un-fichier-texte-avec-createnewfile)

**Github – Installation, utilisation Paho**

[**https://github.com/eclipse/paho.mqtt.java**](https://github.com/eclipse/paho.mqtt.java)

**Stackoverflow.com – Réception message paho**

[**https://stackoverflow.com/questions/22715682/subscribe-and-read-mqtt-message-using-paho**](https://stackoverflow.com/questions/22715682/subscribe-and-read-mqtt-message-using-paho)

**Developpez.com – Utilisation HSQL**

[**http://baptiste-wicht.developpez.com/tutoriels/java/db/hsql/**](http://baptiste-wicht.developpez.com/tutoriels/java/db/hsql/)

**jmdoudoux.fr – Utilisation jdbc**

[**https://www.jmdoudoux.fr/java/dej/chap-jdbc.htm**](https://www.jmdoudoux.fr/java/dej/chap-jdbc.htm)

# Tables des illustrations

Figure 1 : Schéma représentant l'ensemble de l'architecture du projet 10

Figure 2 : Diagramme de classe 13

Figure 3 : Schéma de l'architecture matérielle 14

Figure 4 : Schéma de flux 15

Figure 5 : Schéma du fonctionnement d'un broker MQTT 16

Figure 6 : Schéma représentant l'architecture logiciel 18

Figure 7 : Schéma relationnel 19

Figure 8 : Diagramme de séquence du Dispatcher 22

Figure 9 : Diagramme de séquence d'un SmartSensor 23

Figure 10 : Schéma représentant la conversion de la donnée 24

Figure 11 : Diagramme de séquence du MonotoringController 25

Figure 12 : Interface web permettant le suivi de la personne 26

Figure 13 : Application fonctionnant sur Raspberry Pi 26

Figure 14 : Quantification du travail 28

Figure 15 : Planning effectif 28