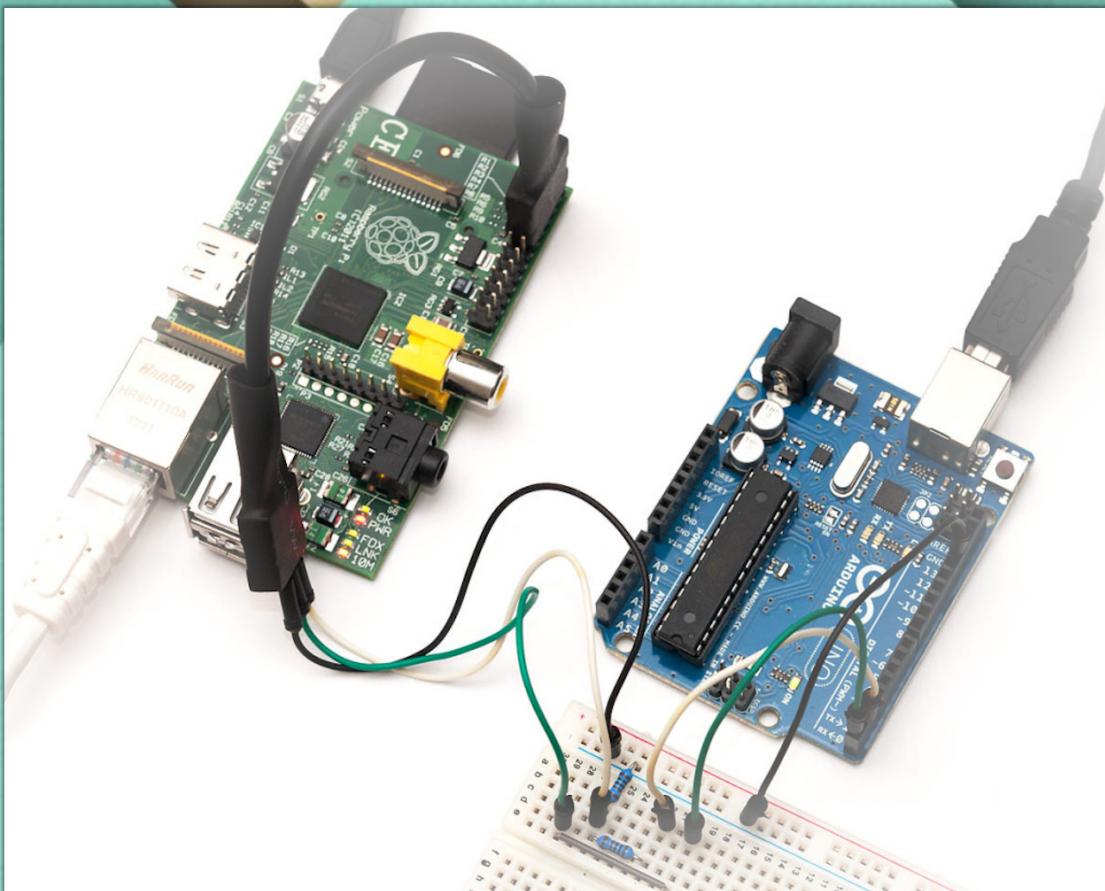




Projet tutoré – « Projet capteurs »

Rapport de projet



BARÇON Lucien
BOULLÉ Benjamin
CHATEAU Lucas
COUSSON Théo
HUG Loïc

Deuxième année
Semestre 4
DUT Informatique

Tuteur : PERROT Gilles

Remerciements

Avant d'aller plus loin dans notre sujet, nous voulons tout d'abord présenter nos sincères remerciements à :

- Monsieur Perrot, tuteur du projet. Pour son écoute, sa disponibilité et ses indications ;
- Madame Paterlini, pour ses précieuses indications concernant le rapport ;
- toutes les personnes ayant œuvré à nous permettre notre semestre au Québec ;
- vous, lecteur, qui accordez du temps à la lecture de ce rapport.

Sommaire

Introduction.....	1
Présentation du sujet – cahier des charges.....	2
I. Principe de fonctionnement.....	2
I.1. Partie système embarqué.....	2
I.2. Partie serveur.....	2
I.3. Partie interface.....	2
II. Technologies utilisées.....	3
II.1. Partie système embarqué.....	3
II.2. Partie serveur.....	4
II.3. Partie interface.....	4
Présentation du projet – réalisation.....	5
I. Partie système embarqué.....	5
I.1. Logiciel d'installation.....	5
I.2. Acquisition des données.....	6
I.3. Gestion de l'acquisition des données.....	7
II. Partie serveur.....	8
III. Partie interface.....	9
Travail en groupe.....	10
I. Constitution du groupe – recrutement de ses membres.....	10
II. Choix du sujet du projet tutoré.....	10
III. Organisation – administration du groupe.....	10
III.1. Chef de groupe.....	10
III.2. Communication.....	11
III.3. Outil de gestion de version.....	11
IV. Organisation – répartition des tâches.....	12
Conclusion.....	13
Sources des illustrations.....	14
Sitographie et Bibliographie.....	15
I. Sites internet consultés.....	15
Sommaire des annexes.....	I
Schémas concernant le projet.....	II
Programme d'installation.....	III
Acquisition des données.....	V
Interface.....	VII
TABLE DES MATIÈRES.....	IX

Introduction

De nos jours, l'informatique n'est plus seulement liée aux ordinateurs mais tend à s'étendre et à s'approprier tous les appareils électroniques pour toutes les tâches et ce quelle que soit leur nature.

Le but de notre projet tutoré était de nous permettre d'aborder ces aspects d'interopérabilité de l'informatique qui sont généralement délaissés dans nos cours au profit de l'algorithme. Par le biais de ce projet tutoré, nous souhaitions pouvoir informatiser tout le processus de gestion de données, de leur obtention depuis des capteurs jusqu'à une interface utilisateur en permettant la lecture, et ce, en y associant la force et les possibilités de l'outil informatique attenantes à notre formation, celles-ci nous fournissant tous les outils dont nous avons besoin pour réaliser de tels projets.

Pour pouvoir concrétiser cette volonté, nous avons choisi de baser notre projet sur la mesure de plusieurs données telles que la température ou la luminosité via des capteurs Arduino¹. Pour répondre à une remarque de notre tuteur, nous avons également rendu possible la mesure de distance qui requiert des relevés plus fréquents.

Le capteur est relié directement à une carte Arduino qui en extrait les données, lesquelles sont transmises à un Raspberry Pi² qui les traite et les renvoie à un serveur. Le serveur permet de récupérer des plages de données définies et des statistiques, qui sont rendues lisibles à l'utilisateur par l'intermédiaire d'un logiciel interface.

Dans ce rapport, nous présenterons en détail le principe de fonctionnement de notre réalisation – la manière dont elle a été construite –, nous évoquerons ensuite les choix des technologies et les difficultés de mise en place auxquelles nous avons fait face ainsi que les solutions trouvées, puis nous parlerons de notre groupe de projet en lui-même avant de finir par une conclusion.

Afin d'en faciliter la lecture et la compréhension, ce rapport est complété par des annexes et des notes de bas de page vers lesquelles le lecteur sera guidé tout au long de sa lecture.

¹ Arduino est une marque de carte microcontrôleur, un microcontrôleur étant une puce intégrant tout le nécessaire à l'exécution de programmes, l'Arduino permet d'utiliser ces puces simplement en fournissant une interface avec un PC servant à la programmer.

² Raspberry Pi est une marque de nano-ordinateurs ARM monocarte. Ces minuscules ordinateurs sont à même d'exécuter des programmes exigeants pour leur utilisation des fonctions propres à des ordinateurs personnels, tels que la mise en réseau ou l'exécution d'un système d'exploitation.

Présentation du sujet – cahier des charges

I. Principe de fonctionnement

Cette première partie illustrée par un schéma visible dans la partie annexe explique le schéma de principe de notre projet et l'articulation de ces différents éléments.

I.1. Partie système embarqué

Un capteur de données Arduino capte plusieurs données et les transmets à une carte Raspberry Pi directement via le port série³.

Le Raspberry Pi stocke temporairement les données en interne pour les renvoyer au serveur par paquets via le réseau.

Le Raspberry Pi est muni d'une interface d'administration permettant la sélection des capteurs connectés et le changement de leur fréquence de rafraîchissement.

I.2. Partie serveur

Le·s système·s embarqué·s envoie·nt les données mesurées régulièrement à un serveur “central” qui les stocke. Le serveur fournit des services de données (APIs⁴) permettant la récupération de données stockées et l'obtention de statistiques permettant aux clients un affichage plus simple à mettre en place et ne requérant aucun calcul sur les données.

I.3. Partie interface

Les données récupérées sont affichées sous forme de graphique sur différentes plateformes : site web, application mobile, application de bureau... Cette interface a besoin, pour récupérer les statistiques, d'effectuer une requête au serveur qui les lui renvoie. L'interface ne se charge d'effectuer aucun calcul pour obtenir les statistiques.

³ Le port série est en quelque sorte l'ancêtre de l'USB et son utilisation est bien plus facile que ce dernier. Le Raspberry Pi et l'Arduino étant tous deux en mesure de communiquer via cette interface, son choix est le plus pertinent pour les connecter l'un à l'autre.

⁴ Une API est un ensemble d'outils fournissant des services à un autre logiciel.

II. Technologies utilisées

Cette deuxième partie énonce les choix de technologie qui ont été faits pour chaque partie du projet et avance pourquoi telle technologie ou telle approche a été favorisée par rapport à une autre. Des notes de bas de page sont fournies afin de simplifier la compréhension de certains points qui peuvent paraître ésotériques à des néophytes ou des non-initiés.

II.1. Partie système embarqué

La partie programme de l'Arduino est codée en C⁵, celle du Raspberry Pi est en Python et en BASH⁶. La communication entre la carte Arduino et la carte Raspberry Pi est faite via le port série de ce dernier. De cette façon, le matériel n'est pas une contrainte et nous avons la possibilité d'utiliser un Raspberry Pi autant qu'un Banana Pi ou un Orange Pi pour ne citer qu'eux.

En ce qui concerne le Raspberry Pi, le programme est développé en Python, ce programme fonctionne grâce à un système GNU/Linux minimaliste au sein duquel un script init.d⁷ se charge du lancement du logiciel au démarrage. Afin de simplifier le déploiement du programme et sa configuration sur le Raspberry Pi, un programme installateur BASH muni d'une interface whiptail⁸ affichable en console a été développé. Ce programme est exécutable via SSH⁹ sans prérequis. La configuration des capteurs, quant à elle, peut être

- 5 Le C est un langage de programmation de bas niveau, donc compilé, rapide à l'exécution et ne requérant pas beaucoup de puissance. Il est donc le choix idéal pour programmer sur un Arduino.
- 6 Le Python et le BASH sont tous deux des langages interprétés ayant une place dominante dans les systèmes GNU/Linux tels que le Raspberry Pi et ses dérivés. L'utilisation du BASH est un incontournable avec Linux et le Python permet d'exploiter les capacités du système tout en offrant un grand confort de développement.
- 7 init.d est un répertoire et un listing système sous GNU/Linux qui contient tous les programmes exécutés au démarrage par systemd – qui est le premier programme lancé lors de l'allumage du système. Cette approche est sans doute la plus propre pour lancer un programme au démarrage sur un système GNU/Linux.
- 8 whiptail est une commande GNU/Linux directement accessible sans installation et qui permet de créer des interfaces comme des écrans de configuration et des champs de saisie en mode texte dans un terminal. Comme nous souhaitions garder un système minimaliste, whiptail était une approche permettant une plus grande ergonomie sans avoir recours à une interface graphique proprement dite.
- 9 SSH permet l'accès à distance d'un système GNU/Linux via le réseau, il ne prend cependant pas nativement en charge la gestion d'interfaces graphiques.

exécutée via le réseau local du Raspberry Pi grâce à l'affichage d'une page Web réalisée avec le framework¹⁰ Python Flask¹¹.

II.2. Partie serveur

Le service web est en Node.js¹² et interagit avec une base de données MySQL pour enregistrer les données via un envoi JSON¹³ réalisé en HTTP par le Rapsberry Pi. Ces données sont ensuite récupérables via une URL sémantique qui renvoie les données au format JSON, ces données sont ensuite directement exploitables par l'interface.

II.3. Partie interface

L'interface a été développée en JavaScript¹⁴, ce qui permet son utilisation en tant que telle sur un navigateur Web. D'autre part, le JavaScript permet un concept très intéressant, celui du « write once, deploy everywhere », grâce à l'intermédiaire de frameworks tels qu'Electron qui permet le déploiement d'une application JavaScript sur ordinateur ou Cordova qui en permet le déploiement sur smartphone, et ce, dans la majorité des cas sans que le système d'exploitation ne soit une contrainte, ce qui rend notre application interface très facilement portable.

- 10 Un framework – appelé cadriel en français – est un ensemble d'outils logiciels visant à aider le développeur à la réalisation d'un programme.
- 11 Flask est un Framework Python utilisé pour le développement web. Il a été choisi parce que le programme était déjà développé en Pytohn, ce qui permet d'éviter une surcharge de langages différents, et également pour sa simplicité de mise en place et sa légèreté – il ne faut pas oublier que ce service est proposé directement par le Raspberry Pi.
- 12 Node.js est une technologie permettant l'exécution de JavaScript côté serveur, le JavaScript étant à la base conçu pour un usage côté client. Cette technologie offre la possibilité de s'affranchir d'un serveur PHP lourd et dont l'installation et la configuration peuvent être fastidieuses.
- 13 Le JSON est un format de notation des objets JavaScript. Il en facilite notamment le stockage et la transmission, ce pour quoi il est utilisé ici.
- 14 JavaScript est un langage de programmation permettant entre autre de réaliser des interfaces web interactifs.

Présentation du projet – réalisation

Cette partie avance la forme qu'a pris le projet lors de sa réalisation. Durant celle-ci, certaines difficultés sont apparues et nous ont obligés à trouver des solutions ou des contournements. Ces difficultés et ces solutions sont détaillées ici.

I. Partie système embarqué

I.1. Logiciel d'installation

Très rapidement lors du développement du programme embarqué sur le Raspberry Pi a été levé le problème du grand nombre de prérequis nécessaires à son fonctionnement et la difficulté de la démarche d'installation et de paramétrage du programme, notamment pour qu'il soit exécuté au démarrage. Dès lors, la mise en place d'un logiciel d'installation se chargeant de toutes ces étapes automatiquement s'est imposée pour apporter une solution à cette difficulté.

Ce programme devant se charger de l'installation des composants requis pour faire fonctionner tout le reste, il ne peut pas se baser sur ces composants et ne peut donc être programmé qu'en BASH.

Autre difficulté, pour être le plus simple et le plus facile à utiliser, le programme d'installation doit être aussi convivial que possible, ce qui est problématique sur un Raspberry Pi volontairement dépouillé d'une interface graphique majoritairement inutile ou lors de l'emploi d'un accès en SSH. Pour cette raison, notre logiciel d'installation devait se baser sur un affichage en mode texte. La solution que nous avons employée ici est l'usage d'une TUI¹⁵.

Le programme affiche à l'utilisateur une interface texte ergonomique de laquelle on peut voir des images en annexe qui le guide pendant toute la durée de l'installation. L'installation étant majoritairement automatique, l'utilisateur n'est sollicité qu'à son début pour choisir la configuration réseau à employer et entrer les éléments de configuration du

¹⁵ TUI pour Text User Interface est une interface où toute la partie graphique est constituée par des caractères. Les fenêtres et les boîtes de dialogues autant que les boutons, champs de texte, etc. sont donc exclusivement constituées de caractères affichables en console sans nécessiter d'interface graphique.

Wi-Fi s'ils sont requis. En outre, la mise en place du lancement automatique du programme au démarrage n'est réalisée que si toutes les étapes de l'installation se sont déroulées sans rencontrer d'erreur, ceci se voulant comme une solution aux difficultés de débogage et de sauvegarde de l'intégrité du programme s'il était lancé automatiquement au démarrage sans être pleinement fonctionnel.

I.2. Acquisition des données

Ayant majoritairement des capteurs analogiques, prévus pour fonctionner avec des cartes Arduino, et ne disposant pas de convertisseurs analogiques numériques, nous avons choisi d'allier Raspberry Pi et Arduino. La carte Arduino est en charge des capteurs et envoie les données au Raspberry Pi qui stocke temporairement ces données et les envoie au serveur.

Nous avons commencé à développer une application basique de communication entre les 2 cartes. Le Raspberry Pi envoie un message représentant la donnée que l'Arduino doit mesurer. Ensuite, l'Arduino envoie la donnée mesurée au Raspberry Pi qui peut la stocker temporairement puis l'envoyer au serveur. En cas d'échec de l'envoi au serveur, la donnée est conservée sur le Raspberry Pi qui essaiera de la retransmettre ultérieurement.

L'application est écrite en python, qui permet facilement de communiquer avec l'Arduino via le port série grâce à la bibliothèque *pyserial*.

La difficulté suivante a été de pouvoir obtenir des données à des intervalles irréguliers, sans connaître à l'avance le nombre de capteurs utilisés.

Cela a été résolu en exécutant répétitivement l'acquisition des données dans un *Thread*¹⁶ toutes les x secondes pour chaque capteur. Cette architecture a nécessité la mise en place de *mutex*¹⁷ pour garantir d'avoir un seul accès simultané au port série de la carte Arduino et de ne pas mélanger les données.

¹⁶ Un Thread est un processus léger qui s'exécute en parallèle du programme principal.

¹⁷ Un mutex est une variable utilisée pour éviter que des objets partagés ne soient utilisés en même temps.

I.3. Gestion de l'acquisition des données

Afin de gérer les différents capteurs connectés au système embarqué il a été décidé de créer une interface simple d'utilisation.

La solution retenue a été de développer une interface de gestion sur le web accessible depuis n'importe quel ordinateur présent sur le réseau local du système embarqué.

L'application étant déjà écrite en python, nous avons décidé de continuer à utiliser ce langage. Le framework Flask, utilisé pour le développement web, a été choisi pour sa simplicité de mise en place et sa légèreté – il ne faut pas oublier que ce service sera proposé directement depuis le Raspberry Pi.

L'interface de gestion se décompose en plusieurs pages – dont des captures d'écran sont visibles en annexe – avec notamment :

- la page principale avec les capteurs utilisés ;
- une page pour ajouter un capteur et modifier la fréquence des mesures ;
- une page pour modifier un capteur en cours d'utilisation.

Il est également possible de supprimer des capteurs simplement.

II. Partie serveur

La majeure difficulté de cette partie était de savoir de quelle façon stocker les données dans la base de données, les contraintes étant les suivantes :

- le client ne connaît pas les capteurs utilisés au préalable, il doit donc pouvoir interroger le serveur sur le type des données stockées ;
- le serveur ne connaît pas au préalable les capteurs utilisés par les systèmes embarqués ;
- les données doivent être organisées par type – par exemple longueur – et par unité de mesure – par exemple centimètres ;
- les données doivent contenir une valeur et une date ;
- le système embarqué doit pouvoir envoyer plusieurs données avec une seule requête, afin de ne pas bombarder le serveur de requêtes ;
- par souci d'optimisation, les données doivent être stockées dans des tables différentes en fonction de leur type et unité afin de ne pas chercher un type précis dans l'ensemble des données.

Les contraintes ont été résolues de la façon suivante : Une table *types* référençait tous les duos type-unité envoyés par les systèmes embarqués. Quand le système embarqué envoie une série de données, celui-ci référence le type et l'unité de ces données. Si ce duo type-données n'est pas référencé dans la table *types*, un nouveau type est créé ainsi qu'une table de nom *type-unite*. Les données envoyées sont alors rajoutées dans cette table. Le client peut savoir les types et unités stockés avec une requête retournant les entrées de la table *types*. Le système embarqué peut donc envoyer pour une requête plusieurs données en précisant en entête le type de donnée et l'unité de mesure, puis dans un tableau des séries de valeurs et de timestamps¹⁸. Toutes les dates rentrant et sortant du serveur étaient sous forme de timestamps afin que le client traduise comme il le veuille la date en chaîne de caractère.

¹⁸ Un timestamp est une valeur numérique représentant une date. Il désigne une quantité de temps écoulé depuis une date de référence. Le timestamp utilisé représente le nombre de secondes écoulées depuis le 1^{er} janvier 1970.

Un autre avantage de ce format est qu'il est bien plus compact qu'une chaîne de caractère écrivant la date.

III. Partie interface

Pour la récupération des données, le but était de recueillir les données du serveur à l'aide d'un service web qui nous renvoyait un JSON pour ensuite les afficher dans les graphiques avec les moyennes sur l'année, le mois, le jour et enfin l'heure. La plus grande difficulté était de créer les bons timestamps dans nos DAO¹⁹ pour obtenir les bonnes données.

Le principal défi de l'affichage des données était de réaliser une interface à la fois facile d'utilisation, mais aussi cohérente et esthétique. Pour cela, nous nous sommes tournés vers le framework « Materialize » qui permet l'utilisation facilité de certains composants ainsi que la mise en place d'une application web *responsive*²⁰. Pour ce qui est de l'affichage des données sous forme de graphiques, nous avons choisi d'utiliser la bibliothèque JavaScript « Chartist ». Nous avons donc dû adapter les données reçues pour cette bibliothèque.

¹⁹ Un DAO est une classe utilisée pour récupérer les données stockées dans la base de donnée

²⁰ Une interface responsive est une interface s'adaptant à la résolution du support sur lequel il est affiché (comme un ordinateur, un téléphone ou une tablette).

Travail en groupe

I. Constitution du groupe – recrutement de ses membres

Le groupe est constitué de :

- Barçon Lucien ;
- Bouillé Benjamin ;
- Chateau Lucas ;
- Cousson Théo ;
- Hug Loïc.

Ce groupe est composé à l'exclusivité d'élèves ayant fait leur troisième semestre d'études au Québec au CÉGEP de Matane. La création de notre groupe s'est imposée comme une évidence dans la mesure où nous avons tous appris à travailler ensemble au Québec où nous avons partagé de bons moments et avons acquis une certaine complicité.

II. Choix du sujet du projet tutoré

Lors de la réflexion quant au sujet de notre projet tutoré, nous avons tous manifesté le souhait, d'une part, de nous démarquer des sujets choisis par les autres groupes, et, d'autre part, de mettre en pratique ce que nous avons appris durant nos deux années de formation. Ces deux points essentiels nous ont poussés à aborder le développement d'un système alliant plusieurs technologies à la fois, ce qui nous paraissait un défi dans la mesure où il s'agissait pour nous à la fois d'une première personnelle mais également d'une inconnue, puisque cet aspect de l'informatique n'est pas abordé par nos cours, et que nous disposions de beaucoup moins de temps pour le faire que nos collègues restés en France.

III. Organisation – administration du groupe

III.1. Chef de groupe

Au sein d'un groupe de projet de petite taille, il est possible d'employer la « méthode agile », très à la mode et utilisée quasi exclusivement par nos professeurs au Québec. Nous avons donc continué pour ce projet à employer cette méthode dont nous

avons testé l'efficacité. À ce titre, nous n'avions pas de « chef de projet », chacun se voyait confier des responsabilités et des attentes, ce qui, dans notre cas, a semblé permettre un fonctionnement relativement satisfaisant.

III.2. Communication

Compte tenu de la situation exceptionnelle actuelle due à la crise sanitaire provoquée par le Coronavirus (COVID-19), nous avons mis en place puis utilisé comme moyen exclusif de communication un serveur Discord, par le biais duquel nous échangions et prenions nos décisions.

III.3. Outil de gestion de version

Bien que cela paraisse l'évidence même d'employer un tel outil pour le développement, il serait sans doute judicieux d'insister davantage sur son utilité auprès des élèves de première année qui sont loin de tous en avoir connaissance.

Afin de rendre praticable quelque démarche de développement que ce soit, notre groupe a mis en place, avant de commencer une quelconque ébauche de développement, un repository GitHub²¹. Cet outil nous a par la suite apporté l'assurance que nous travaillions tous sur le même code et tous sur la version la plus à jour, tout en nous préservant d'effacer par erreur le travail d'un collègue.

Ce point était primordial dans le développement de notre projet. Comme dit précédemment, nous convoitions au travers de ce projet de créer un outil se basant sur plusieurs technologies à la fois, tant au niveau matériel que logiciel. Pour cette raison, nous devions nous garantir de nous affranchir de la moindre incompatibilité de code, dans la mesure où elle ruinerait immédiatement et intégralement le fonctionnement du projet.

²¹ Un « repository Github », ou « dépôt Github » en Français est une plateforme en ligne permettant le stockage de code source de projets informatiques. Ce service en ligne se base sur l'outil de versionnement GIT développé à l'origine par le créateur de Linux. L'intérêt de cet outil est décrit dans la suite du paragraphe.

IV. Organisation – répartition des tâches

La connaissance de chacun et la méthode agile nous a permis de rapidement déterminer quelle personne devait s'occuper de telle ou telle tâche. Ainsi, si certains sont plus à leur aise sur du front-end, certains préfèrent le back-end, ou encore passer de l'un à l'autre pour aider ses collègues. Dans cette mesure nous n'avons rien imposé au départ, nous avons laissé faire. Le résultat a été une atmosphère de travail très agréable où chacun pouvait, en cas de difficulté, se faire aider par qui que ce soit et ne pas rester à la fois bloqué et seul avec en plus la peur de décevoir les autres.

Conclusion

Notre projet tutoré se voulait comme la mise en pratique de la théorie que nous avons apprise durant notre formation.

Comme l'informatique touche les domaines les plus divers, allant de l'électronique aux smartphones, nous souhaitions intégrer à notre projet ces deux éléments. C'est chose faite de l'Arduino, permettant de traiter des signaux électriques avant même toute informatisation jusqu'à l'interface utilisateur, utilisable depuis n'importe quel navigateur dont les smartphones mais sans s'y limiter.

Entre temps, les données transitent par le réseau, font l'objet d'interprétation, de conversion ; elles servent de prétexte à l'implémentation d'un ensemble de parties qui mobilisent nos connaissances en plus de donner vie à un vrai projet répondant à un besoin réel.

Grace à ces points, ce projet nous a permis de répondre à nos attentes et, encore une fois, d'apprendre à travailler en groupe. Plus spécifiquement peut-être, de nous apprendre que le travail au sein d'un groupe de personnes passionnées est toujours enrichissant, ce qui ne faisait pas partie, à l'origine, de la liste de nos exigences.

Sources des illustrations

- domoticx.com :
<http://domoticx.com/wp-content/uploads/2016/05/Raspberry-en-Arduino-seriele-communicatie.jpg>
 - ↪ Image de la page de garde représentant une carte Raspberry PI et une carte Arduino reliées entre elles via un breadboard ;

Sitographie et Bibliographie

I. Sites internet consultés

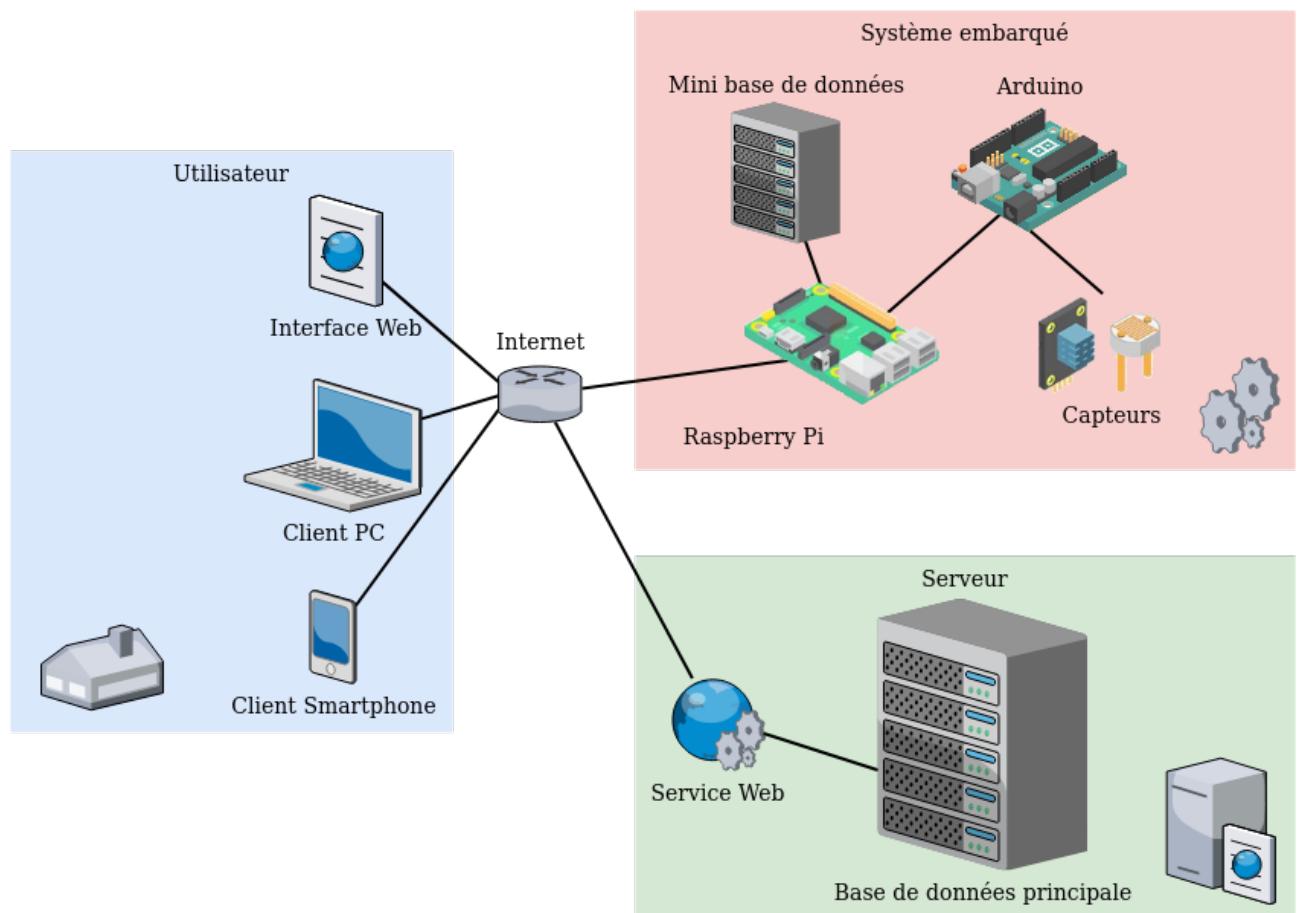
- TkkrLab :
https://tkkrlab.nl/wiki/Arduino_37_sensors
 - ↪ Informations sur les capteurs
- Documentation Flask:
<https://flask.palletsprojects.com/en/1.1.x/>
 - ↪ Utilisation de Flask
- Documentation Materialize :
<https://materializecss.com/>
 - ↪ Utilisation de materialize
- Documentation Chartist :
<https://gionkunz.github.io/chartist-js/>
 - ↪ Utilisation de chartist
- Documentation de Whiptail :
https://en.wikibooks.org/wiki/Bash_Shell_Scripting/Whiptail
 - ↪ Utilisation de Whiptail
- forum Stackoverflow :
<https://stackoverflow.com/>

Sommaire des annexes

Schémas concernant le projet.....	II
Schéma de principe du projet.....	II
Programme d'installation.....	III
Configuration du Wi-Fi.....	III
Progression de l'installation.....	IV
Acquisition des données.....	V
Capteurs en cours d'utilisation.....	V
Ajout d'un capteur.....	V
Modification d'un capteur.....	VI
Interface.....	VII
Menu principal.....	VII
Page détaillée.....	VII

Schémas concernant le projet

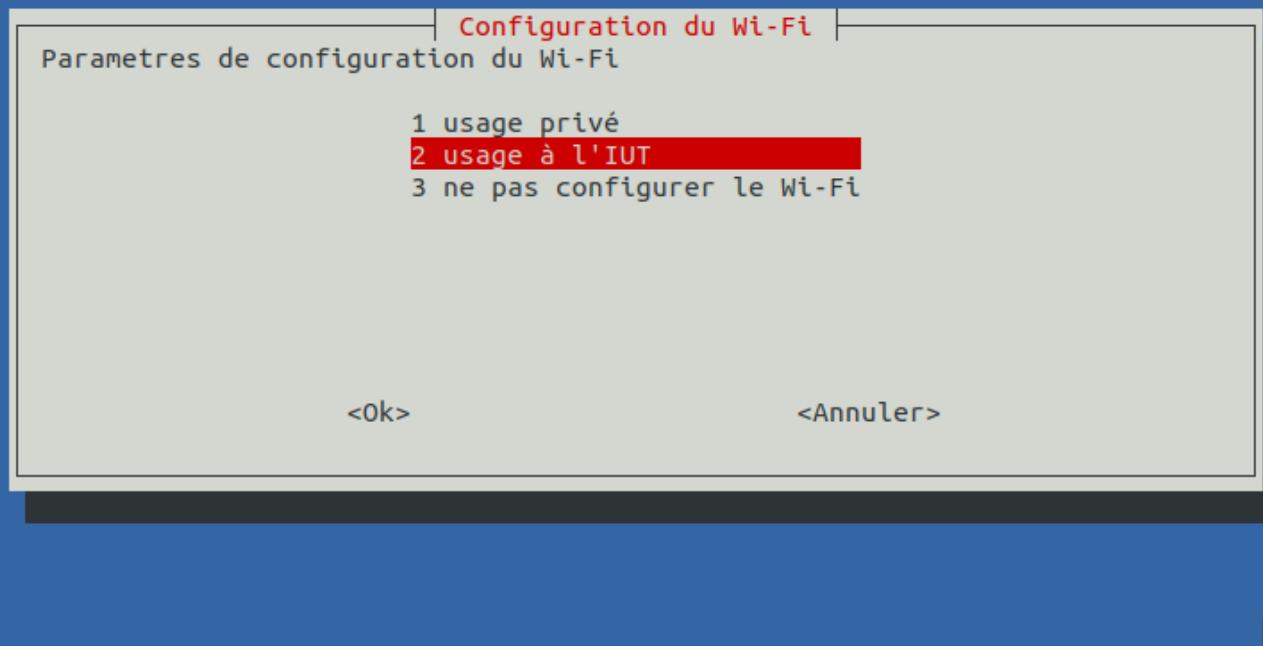
Schéma de principe du projet



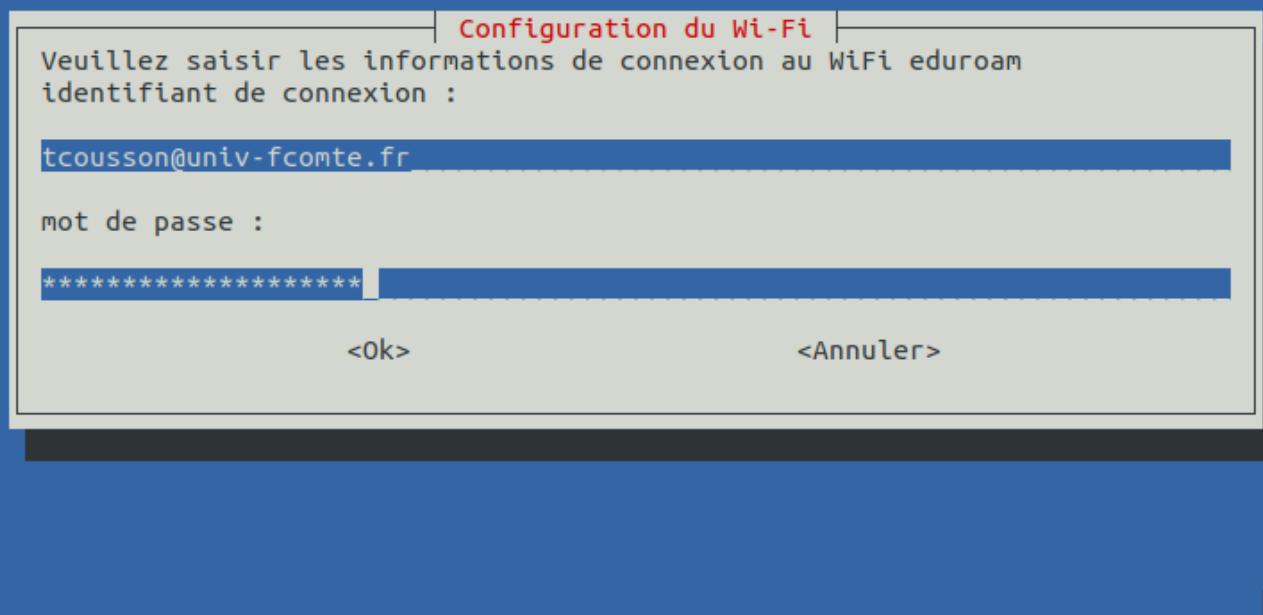
Programme d'installation

Configuration du Wi-Fi

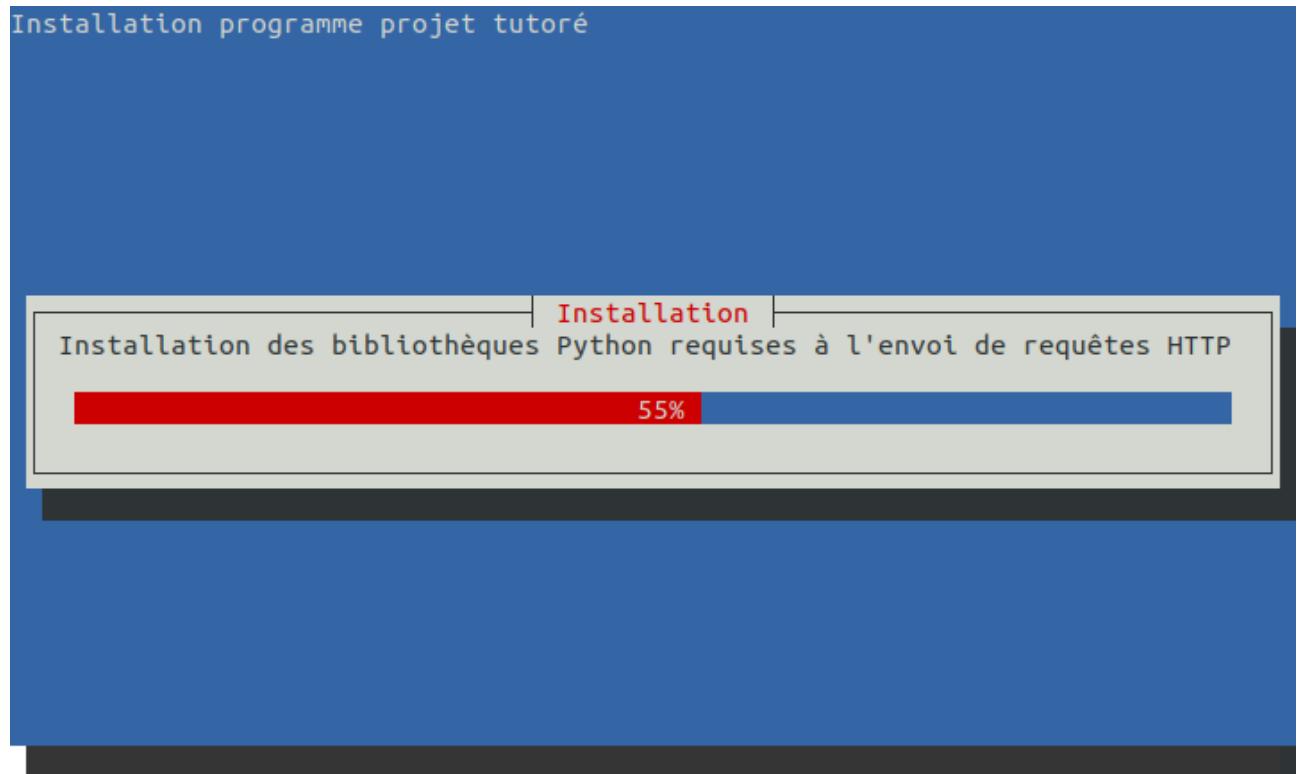
Installation programme projet tutoré



Installation programme projet tutoré

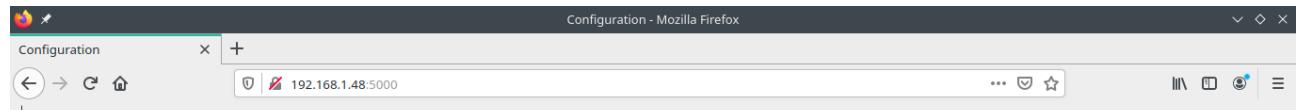


Progression de l'installation

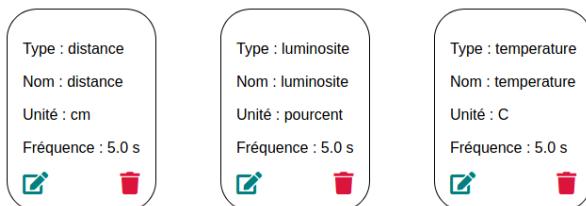


Acquisition des données

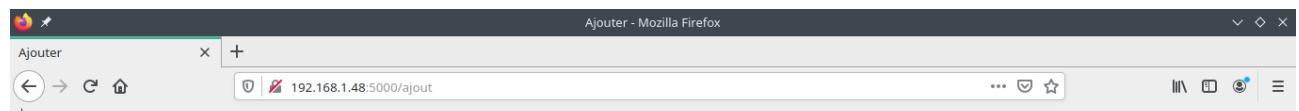
Capteurs en cours d'utilisation



Administration des capteurs



Ajout d'un capteur



Ajout d'un capteur

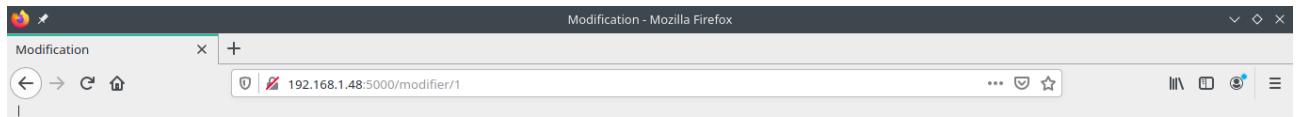
Home icon

Type :

Nom :

Fréquence des mesures : s

Modification d'un capteur



Modification d'un capteur



Type : temperature

Nom :

Fréquence : s

Interface

Menu principal



Page détaillée



TABLE DES MATIÈRES

Remerciements.....	3
Sommaire.....	5
Introduction.....	1
Présentation du sujet – cahier des charges.....	2
I. Principe de fonctionnement.....	2
I.1. Partie système embarqué.....	2
I.2. Partie serveur.....	2
I.3. Partie interface.....	2
II. Technologies utilisées.....	3
II.1. Partie système embarqué.....	3
II.2. Partie serveur.....	4
II.3. Partie interface.....	4
Présentation du projet – réalisation.....	5
I. Partie système embarqué.....	5
I.1. Logiciel d'installation.....	5
I.2. Acquisition des données.....	6
I.3. Gestion de l'acquisition des données.....	7
II. Partie serveur.....	8
III. Partie interface.....	9
Travail en groupe.....	10
I. Constitution du groupe – recrutement de ses membres.....	10
II. Choix du sujet du projet tutoré.....	10
III. Organisation – administration du groupe.....	10
III.1. Chef de groupe.....	10
III.2. Communication.....	11
III.3. Outil de gestion de version.....	11
IV. Organisation – répartition des tâches.....	12
Conclusion.....	13
Sources des illustrations.....	14
Sitographie et Bibliographie.....	15
I. Sites internet consultés.....	15
Sommaire des annexes.....	I
Schémas concernant le projet.....	II
Schéma de principe du projet.....	II
Programme d'installation.....	III
Configuration du Wi-Fi.....	III
Progression de l'installation.....	IV
Acquisition des données.....	V
Capteurs en cours d'utilisation.....	V
Ajout d'un capteur.....	V

Modification d'un capteur.....	VI
Interface.....	VII
Menu principal.....	VII
Page détaillée.....	VII