SMART SECURE SYSTEM

20 mars 2018

Rapport de projet tuteuré de 2ème année

Floryann AGNELOT, Antoine CARQUILLE, Yvan CHAPUIS, Dorian JEANNIN, Rémi LAURENT, Lucas PHILLIPE

R&T
DUT Réseaux & Télécommunications
IUT Belfort-Montbéliard

IUT
Belfort-
Montbéliard

UBFC
UNIVERSITÉ
BOURGOGNE FRANCHE-COMTÉ

Remerciements

Nous souhaitons remercier les professeurs du département Réseaux et Télécommunications de Montbéliard pour leur écoute et leur disponibilité.

Nous souhaitons également remercier Daphné COMBET et Léa PERRET pour leur contribution à la vidéo.

Table des matières

Introduction

Le projet :

Smart Secure System ou 3S est un système de sécurité-domotique développé dans le cadre de notre projet tuteuré de deuxième année d'IUT Réseaux & Télécommunications.

Il se présente sous la forme de plusieurs mini-ordinateurs Raspberry-Pi auxquels sont connectés différents capteurs : caméra, faisceau infrarouge, température…
Ces RPI sont ensuite installés chez les clients, à moindre coût et transmettent 24h/24 7J/7 leurs données à notre serveur. L'entièreté des données est ensuite visualisable à distance via une application smartphone ainsi qu'un site web dédié, l'usager peux même interagir sur certaines fonctionnalités du système.

A qui s'adresse-t-il ?

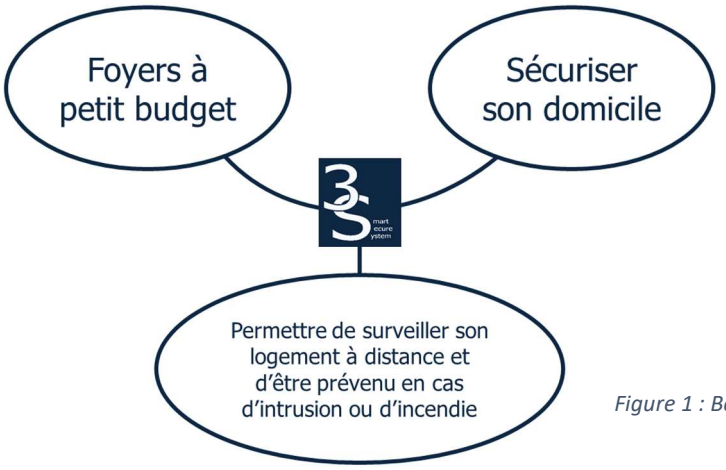


Figure 1 : Bête à cornes

Depuis le départ 3S avait comme vocation de cibler des petits budgets, ne pouvant pas forcément se payer un système d'alarme standard à quelquefois presque 1000 €, on peut donc compter dans cette catégorie :

- Les étudiants : le système est facile à mettre en place et peut être amélioré par l'utilisateur de son propre chef, il est très orienté « connecté ».

- Les personnes âgées : des proches peuvent facilement monitorer le système de leurs parents et être alerté en cas de problème.

Cahier des charges & exigences.

Quelques points étaient pour nous l'essence même du projet :

1) Le projet doit être le plus open source possible, l'utilisateur doit pouvoir regarder, comprendre et améliorer l'entièreté des programmes.
2) Le coût total du projet doit être assez bas.
3) Le système doit être facile à mettre en place, par l'usager s'il le souhaite.

Gestion de projet

FreeMind

Nous avons réalisé un *freemind* afin d'avoir une vision claire des solutions techniques envisagées pour le projet. Nous pouvions nous référer à ce document afin de voir quelle direction donner au projet.

(Vous pouvez retrouver ce freemind en annexe 1)

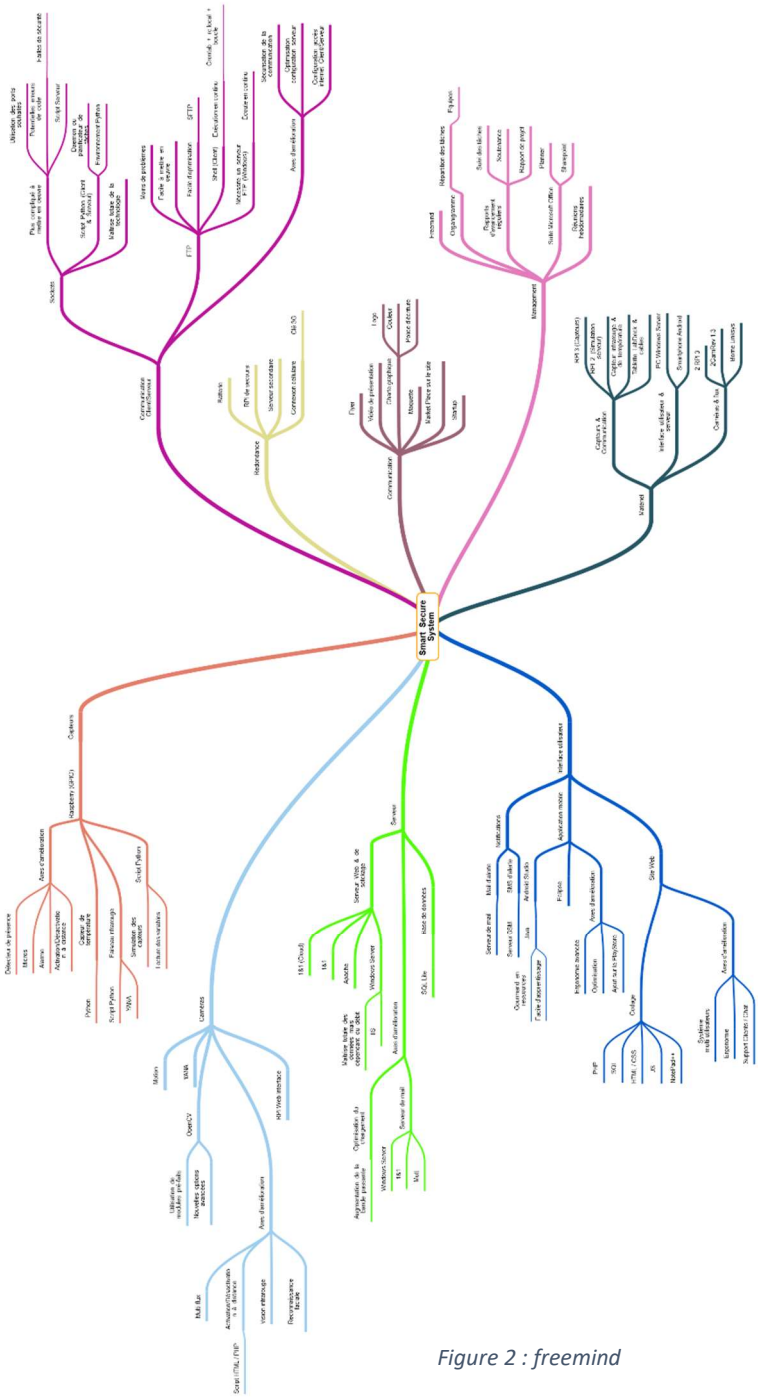


Figure 2 : freemind

Organigramme

Un groupe de 6 personnes nécessitait une hiérarchie pour une répartition des tâches efficace, pour cela nous avons nommés :

- Un chef de projet
- Un chef de projet adjoint
- Un responsable applicatif
- Un responsable réseau

Chaque responsable gère une équipe composée d'un technicien et de lui-même. Yvan et Antoine se sont proposés pour assumer ces rôles de responsable. Le chef de groupe a été élu pour recueillir au mieux la confiance des autres membres. Les équipes ont étaient constituées par affinités et par rapport aux compétences techniques. Un groupe de 6 personnes est assez compliqué à gérer, c'était une première pour nous tous mais l'expérience a été enrichissante. La bonne entente entre les membres a été primordiale et a permis de faire travailler les équipes main dans la main.

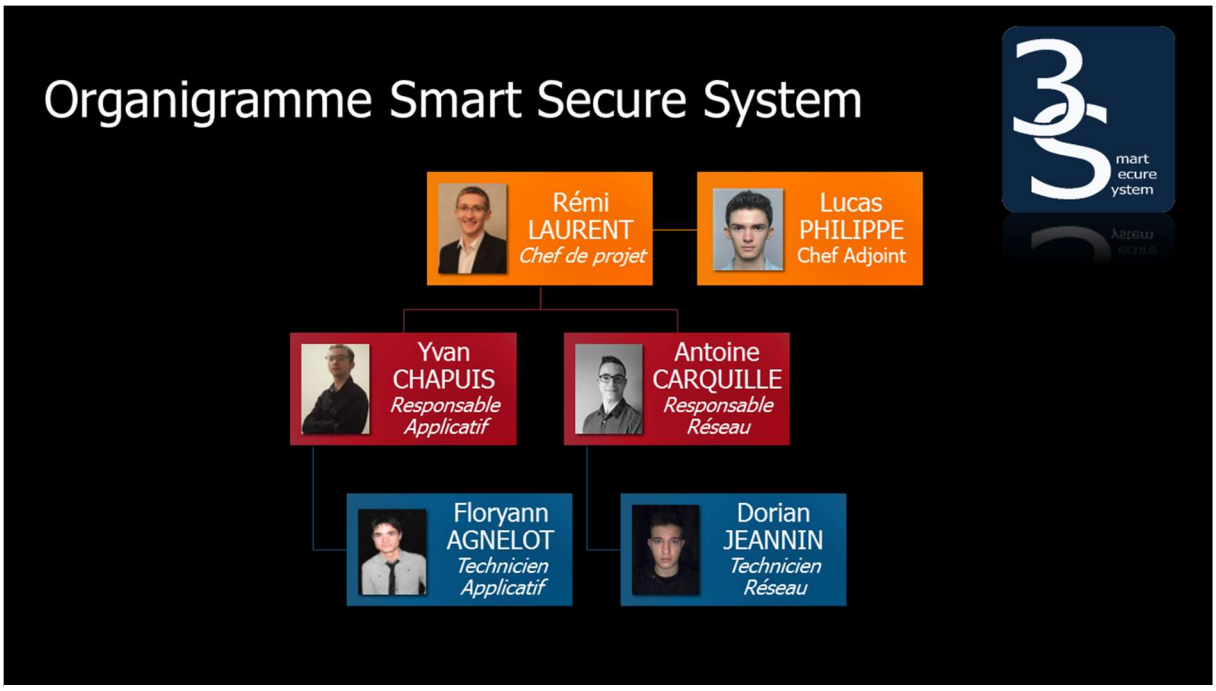


Figure 3 : Organigramme 3S

Réunions

Nous nous sommes rencontrés à plusieurs reprises lors de réunions pour suivre l'avancement du projet et transmettre les informations relatives au projet. C'est lors de ces échanges que chaque membre pouvait poser des questions ou suggérer des modifications à apporter au projet. Nous avons aussi à 3 reprises rencontrés notre tuteur de projet M.Givron pour faire le point avec lui sur l'avancement du projet. Ces durant ces réunions qu'il nous orienté en fonction de ses attentes.

Documents de travail

Pour faciliter le travail en groupe, nous avons choisi la solution de documents à compléter (rapports d'avancement, parties personnelles du rapport, diaporama, remarques, …) qui étaient ensuite centralisés et synthétisé par les chefs de groupes pour permettre une vision globale de l'avancement du projet et simplifier le travail des équipes. Pour partager ces documents, nous avons utilisés la solution d'entreprise Microsoft SharePoint. Un exemple de document à compléter est visible en annexe 6.

Parties personnelles

Floryann et Yvan, application, site web, serveur

I. Langages Web

1. HTML

HTML (HyperText Markup Language) est utilisé pour créer le contenu et la structure d'une page web. Il sera ensuite associé à d'autres langages dont je parlerais ultérieurement.

Figure 4 : By W3C, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=12736763

Il fonctionne grâce à des " balises " représentées par les symboles " < " et " > " insérées dans un texte. La plupart des balises possèdent deux parties, une partie ouvrante " <html> ", puis fermante " </html> ".

Chacune de ses balises influe sur la portion de texte située entre elles :

Figure 5 : https://fr.slideshare.net/inpixelitrust/support-de-cours-initiation-html-et-css-partie-1-html 0, .wikimedia.org/w/index.php?curid=1

Pour commencer à développer en html, il faut simplement ouvrir un fichier texte que l'on enregistrera en .htm ou .html et respecter cette structure :

< !doctype html> -> Indique le type de document

<html> ->Début du fichier html

<head> ->Contient l'en-tête du fichier

<title> 'Titre de la page'</title> ->Définit le titre apparaissant en haut du navigateur

</head> ->Termine l'en-tête

<body> ->Début du réel contenu de la page

</body> ->Fin du contenu de la page

</html> ->Fin de la page

Il existe également des optons aux balises, on les nomme les attributs. Ils apportent une information supplémentaire, ou obligatoire en fonctions des balises utilisées. Ils s'écrivent comme suit :

< " balise " attribut= " valeur "></ " balise ">

Les plus importants sont les sélecteurs, qui nous permettront de mettre en forme les éléments via le langage css.

2. CSS

Figure 6 : By Nikotaf - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=48555427

CSS (Cascading Style Sheets) ou "Feuilles de style en cascade" en français est un langage utilisé pour " styliser " un document html.

C'est le langage qui permet par exemple d'afficher un texte en rouge, modifier la police ou encore dimensionner des images.

On applique des règles de styles sur les éléments html via un sélecteur (identifiant de l'élément html), puis on déclare nos règles, et tout ça dans un fichier texte enregistré en .css :

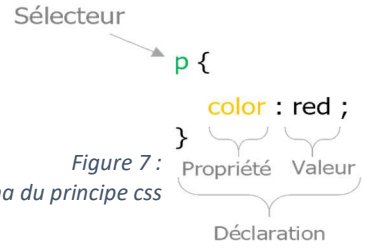


Figure 7 : Schéma du principe css

Il existe plusieurs façons de sélectionner un élément html, sans trop entrer dans les détails, voici 3 façons :

-Utiliser le nom de la balise comme sélecteur
p qui est une balise de paragraphe), cela a pour effet de sélectionner tous les éléments " p " de la page.

-Utiliser un id précédemment déclaré sur l'élément via l'attribut " id ". Dans ce cas, dans le fichier css, on écrira un " # " avant le sélecteur. Ex : #test

-Utiliser une class déclarée sur l'élément html via l'attribut " class ". Dans ce cas, au niveau du fichier css, on écrira un " . " avant le sélecteur. Ex : .test

On indique ensuite entre accolade plusieurs déclarations comme la couleur, la taille, etc

Il ne nous reste plus qu'à lier notre fichier css au fichier html existant.

Pour cela, dans la balise <head></head>, on inclut :

<link rel= " stylesheet " type= " text/css " href= " Chemin_du_fichier_css " >

Le css est primordial pour un site internet, il gère tout le design de celui-ci.

3. PHP

PHP (Hypertext Preprocessor) est un langage aussi bien utilisé pour le développement web, que pour la programmation en général.

C'est un langage dynamique de script côté serveur. Nous en sommes aujourd'hui à la version 7. Il est considéré comme une des bases de sites " dynamiques ".

Figure 8 :
By Colin Viebrock - http://php.net/logos, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=9632398

On peut insérer ce code à n'importe quel endroit dans notre fichier html, à condition qu'il soit situé entre ces balises :

< ?php

?>

Autre fait important, lorsqu'on clique droit sur une page dans son navigateur, que l'on clique sur " Voir le code source de la page ". On voit apparaître le html, le css, mais pas le php car il peut contenir des informations sensibles que nous verrons dans le prochain langage.

4. SQL

Le langage SQL (Structured Query Language) permet de dialoguer avec des bases de données. Les bases de données permettent de stocker et classer des informations aisément.

Figure 9 :
https://www.mysql.com/common/logos/powered-by-mysql-167x86.png

Ici, nous utilisons des bases de données relationnelles et nous inscrivons notre code à l'intérieur du langage php, qui contiendra des identifiants de connexions aux bases de données, qui sont des données sensibles.

Une base de données relationnelle est organisée selon le principe suivant :

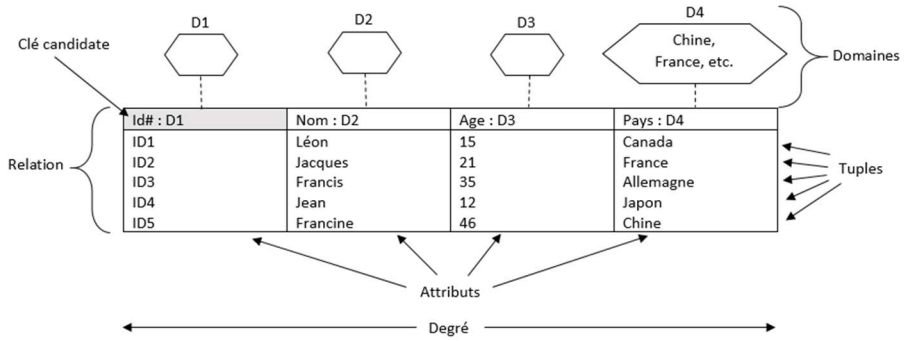


Figure 10 :
By Samdus - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=47261934

Je ne vais pas trop rentrer dans les détails mais voici une table.
Une base de données en contient plusieurs.

Au-dessus des colonnes, nous apercevons des noms de champ (id#, Nom, Age et Pays). Puis avec ces colonnes, on inscrit des enregistrements (ou tuples).
Celle-ci répertorie des utilisateurs, nous pouvons voir que le premier enregistrement nous dit que l'utilisateur Léon d'ID 1, à 15 ans et habite au Canada.

Avec des requêtes sql, nous pouvons créer une base de données, une table avec ces colonnes et des enregistrements.
Nous les récupérons ensuite pour, par exemple, créer un système de comptes sur un site ou stocker des articles à la vente sur un site d'E-commerce.

5. JavaScript

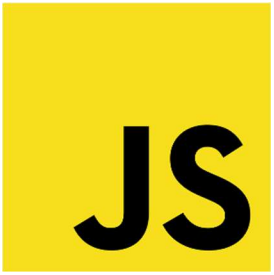


Figure 11 :
https://pixabay.com/en/javascript-js-logo-source-code-736400/
CC0 Creative Commons

Javascript est un langage de programmation de scripts dynamiques côté client, contrairement à php.
Il permet d'implémenter des mécaniques complexes sur une page web, telles des animations, menu défilants, etc.
On peut l'utiliser pour stocker des variables et créer des mécanismes qui s'actualisent en temps réel, pour modifier les propriétés d'un ou plusieurs éléments html (ce qui permet de faire des animations). Et également d'utiliser des " évènements ", par exemple, lorsque l'on clique à un endroit un bout de code sera lancé. Ce langage est très important dans les pages web actuelles.

II. Protocoles Web

1. HTTP

HTTP (Hypertext Transfer Protocol) est le protocole qui gère la communication client-serveur.

Apache est le serveur http le plus populaire dans le monde d'Internet.

Le client qui veut se connecter à une page web va communiquer avec le serveur via l'URL (ex : www.google.fr), et le serveur va lui renvoyer la page souhaitée.

Figure 12 :
By IETF HTTP Working Group (HTTPbis) - https://httpwg.github.io/asset/http.svg, Public Domain, https://commons.wikimedia.org/w/index.php?curid=36542599

L'URL correspond à l'adresse d'un serveur où sont stockés nos différents fichiers .html, .ccs, .php et .js

Il existe une version plus sécurisée de ce protocole, appelée https (le " s " signifie secure), utilisant d'autres protocoles pour sécuriser le transport des données.

2. DNS

Le DNS (Domain Name System) est le système qui nous permet de pouvoir taper des adresses internet comme www.google.fr.

DNS traduit l'adresse URL envoyés en adresse IP, pour que le navigateur puisse joindre le serveur web.

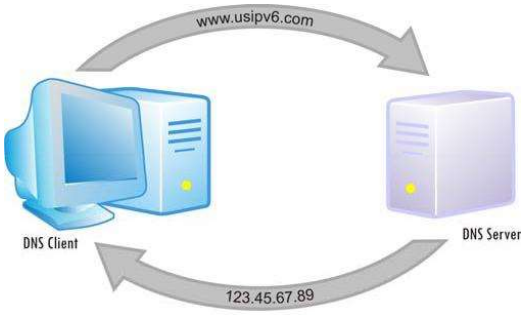


Figure 13 :
By Б.Өлзийдэлгэр - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=47865617

III. Outils

1. Notepad++

Figure 14 : By Motaz. - Notepad++ SVN repository: http://notepad-plus.svn.sourceforge.net/viewvc/notepad-plus/trunk/Power

Notepad++ est un éditeur de texte sous Windows, particulièrement adapté pour développer des codes sources (indentation automatique, correction de certaines erreurs). Il est gratuit et très simple d'utilisation.

2. WAMPSERVER

Figure 15 : By Saber omri - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=43982764

C'est une plateforme de développement WEB sous Windows de type WAMP (Windows Apache MySQL PHP).

Il permet d'installer un serveur web local sur n'importe quel PC contenant un serveur Apache, PHP, MySQL ainsi que PHPMyAdmin, permettant de gérer ses bases de données via une interface graphique.

C'est ce que nous avons utilisé pour développer le site web et le tester en amont sur un PC.

IV. Site Web

1. Le site Web

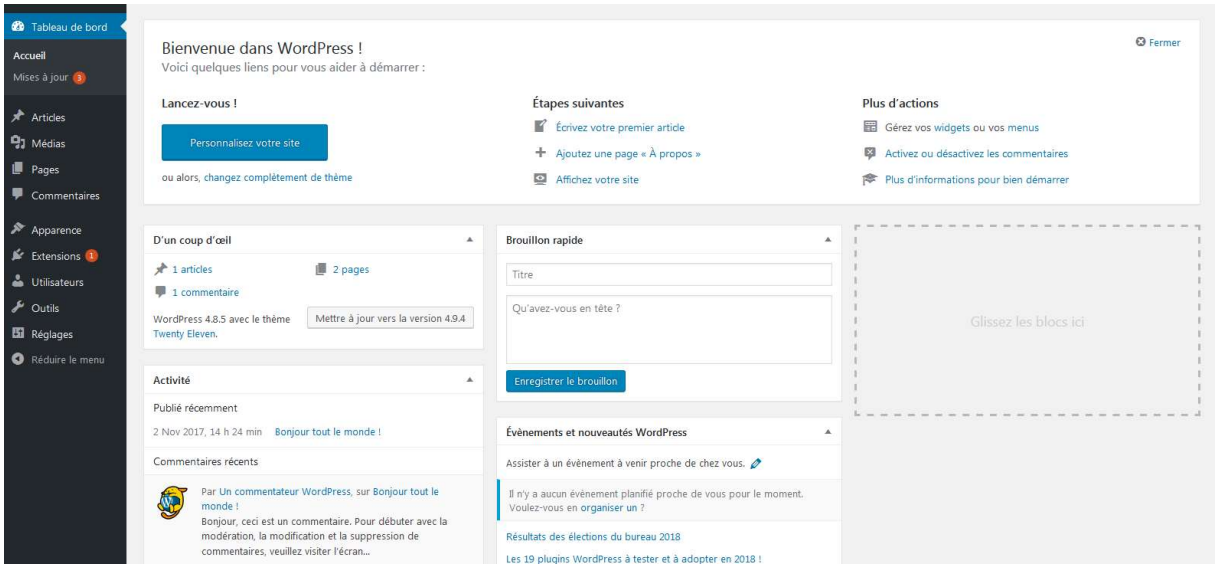
On disposait de plusieurs façons pour développer le site :

Utiliser un CMS (Content Management System) ou développer le site " à la main ".

a. Le CMS

C'est une plate-forme de développement de sites web qui proposent des templates de sites et, via une interface de gestion, permet de créer rapidement son site web.

Il existe beaucoup de CMS (Drupal, Joomla, E-majine ou encore le plus connu, Wordpress)



b. Le développement à la main

Le plus gros inconvénient de cette manière de faire est que cela met beaucoup de temps avant d'arriver au résultat final, mais nous sommes maîtres de tout ce qui se passe sur notre site.

C'est donc la solution que j'ai choisie.

c. Le Site

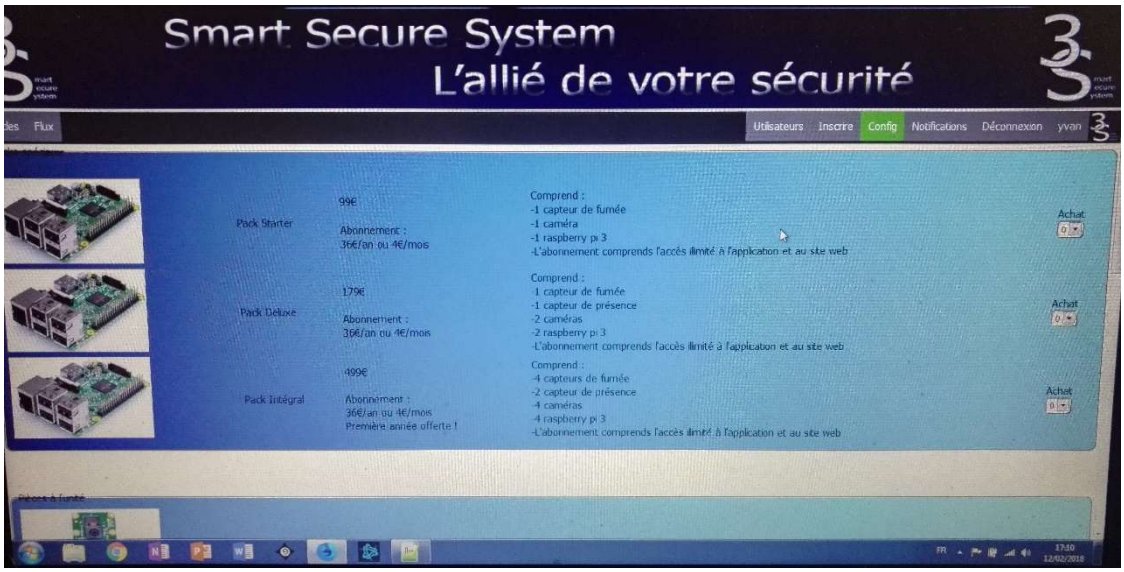


Figure 16 :
1ère version du site Web

Notre site Web " 3S " a été entièrement développé par l'équipe Smart Secure System. Il vous permet de consulter les différents flux des caméras installées chez vous en direct depuis n'importe où dans le monde.

Vous pourrez également accéder à un historique des alertes détectées par les différents capteurs (fumée, présence, faisceau laser).

Pour cela, il faudra se connecter sur la page prévue à cet effet avec des identifiants que nous vous auront communiqué au moment de l'installation.

Ses identifiants sont valides sur le site web, ainsi que sur l'application mobile.

Les utilisateurs ont également un espace personnel, où l'on peut retrouver ses informations telle que son login, sa date d'inscription ou son logo.

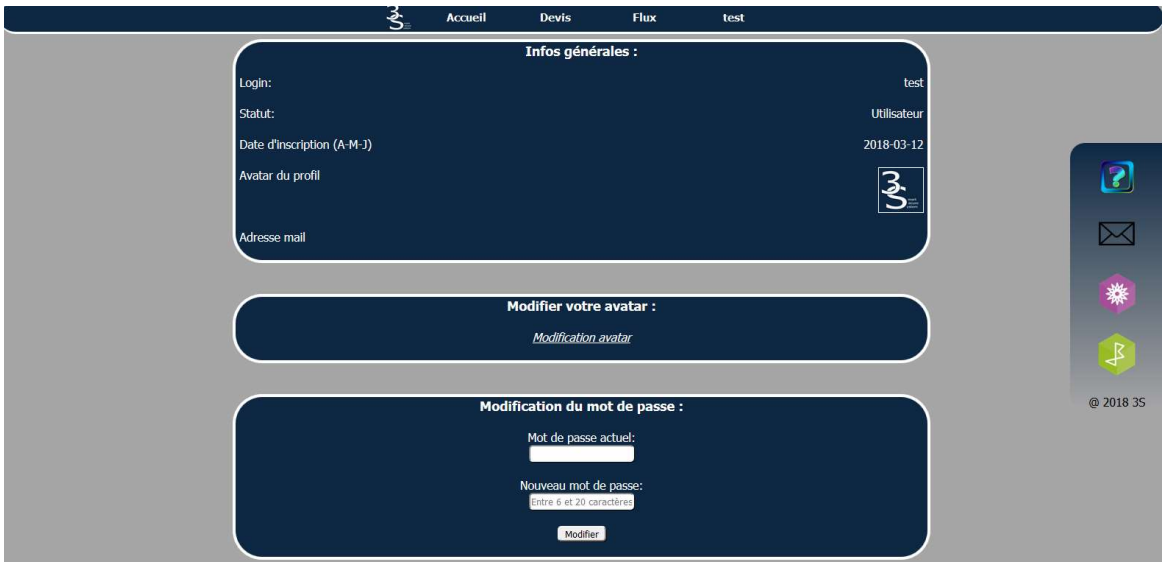


Figure 17 :
site V2
Page mon compte

L'utilisateur peut modifier son avatar, visible pour le moment, que dans cette page-ci. Il peut changer son mot de passe également et ajouter/modifier une adresse e-mail.

Une page de devis est disponible sur le site pour avoir une estimation du prix du matériel et des différents abonnements proposés.

Nous proposons des forfaits en packs et des pièces à l'unité (Raspberry pi 3, détecteur de fumée, capteur de mouvement et caméra pi).

Nous possédons également une page pour lire les données de capteurs de température sous forme de graphique, et les états des autres capteurs.

Les caméras sont également disponibles à la visualisation en direct.

Figure 18 :
Page de devis

Le site possède également une version mobile, grâce au responsive design, qui est une technique pour réaliser des sites web compatibles à la fois sur PC, smartphones et tablettes.

Il consiste à adapter les contenus existants pour les rendre pratique et confortable sur tous les supports.

d. Axes d'améliorations

Sur le site, tous les utilisateurs partagent le même flux vidéo et les mêmes données de capteurs.

Mon premier axe d'amélioration concerne la création d'un système multi-utilisateur où chacun possèderait ses caméras et ses capteurs.

L'optimisation du temps de chargement de certaines pages pourraient aussi être améliorés ainsi que l'expérience utilisateur, via des personnalisations de comptes plus poussées et la mise en place d'un tchat textuel et d'un forum.

V. Androïd

1. Généralités

Androïd est un système d'exploitation créé en septembre 2008 et basé sur le noyau Linux, les langages de programmation utilisés sont le Java et le C++ . Celui-ci fut initialement conçu pour des appareils photos. Ce n'est que plus tard que Google et T-Mobile obtiennent un partenariat de distribution exclusif d'Androïd. Aujourd'hui, Google en est le développeur et l'utilise sur de nombreux appareils embarqués : smartphone / montre connectée / tablette / télévision / GPS Automobile. Google à également décidé de rendre le code source public, de nombreuses failles sont alors détectées par le public et des patchs sont ainsi proposés pour résoudre ces failles.

Figure 19 : BugDroïd
Logo d'Androïd

2. Différentes versions

Version	Nom de Code	Date de Sortie	Version du Noyau Linux
1.0	Aucun	11 / 11 /2007	?
1.1	Petit Four	22 / 11 2008	?
1.5	Cupcake	30 / 04 / 2009	Linux 2.6.27
1.6	Donut	15 /09 /2009	Linux 2.6.29
2.x	Eclair	26 / 11 /2009	Linux 2.6.29
2.2.x	Froyo	20 / 05 /2010	Linux 2.6.32
2.3.x	Gingerbread	06 /12 /2010	Linux 2.6.35
3.x.x	Honeycomb	22 /02 /2011	Linux 2.6.36
4.0.x	Ice Cream Sandwich	19 / 10 /2011	Linux 3.0.1
4.1.x	Jelly Bean	09 / 06 /2012	Linux 3.0.31
4.2.x	Jelly Bean	13 / 11 / 2012	Linux 3.0 - 3.1
4.3.x	Jelly Bean	24 / 06 / 2013	Linux 3.0 - 3.1
4.4.x	KitKat	31 / 10 / 2013	Linux 3.4 – 3.10
5.0.x	Lollipop	3 / 11 / 2014	Linux 3.4 – 3.10
5.1.x	Lollipop	09 /03 /2015	Linux 3.4 – 3.10
6.0	Marshmallow	05 / 10 /2015	Linux 3.10 – 3.18
7.x	Nougat	22 / 08 /2016	Linux 3.18 – 4.4
8.0.x	Oreo	21 /08 /2017	Linux 4.4 – 4.9
8.1.x	Oreo	05 / 11 /2017	Linux 4.4 -4.9

Figure 20 : Androïd 2.2 (Froyo)

Figure 21 : Androïd 4.0 (Ice Cream Sandwich)

Figure 22 : Androïd 7.0 (Nougat)

3. Androïd SDK

Google a développé un ensemble complet d'outils de développement à disposition du public : Androïd SDK. Il s'agit en réalité d'un kit de développement qui inclut un débogueur, des bibliothèques logicielles et un émulateur basé sur QEMU. Cela permet donc à toutes personnes de pouvoir créer ses propres applications Androïd. Pour ce faire, l'IDE (environnement de développement) officiel était Eclipse combiné au plugin d'outils de développement d'Androïd (ADT).

Figure 23 : Logo Androïd Studio

Désormais depuis 2015, Google a revu son IDE et officialise Androïd Studio comme IDE officiel pour le SDK Androïd. Grâce à cette évolution, les développeurs peuvent désormais utiliser n'importe quel éditeur de texte pour modifier les fichiers JAVA et XML. Malgré tout, des prérequis sont obligatoire : Java Development Kit (JDK) et Apache Ant afin de pouvoir créer, construire et déboguer les applications Androïd, ainsi que contrôler les périphériques Androïd utiliser (très pratique pour pouvoir lancer un redémarrage ou même installer un logiciel à distance).

Le SDK Androïd inclut un émulateur de smartphone afin de permettre de simuler les différentes versions d'Androïd, celui permet ainsi de tester son application mais également des tester les nouveautés des versions Androïd. Bien évidemment, le SDK contient toutes les images des différentes versions d'Androïd.

4. Les langages Java et XML

a. Langage Java

Figure 24 : Logo Java

Le langage Java est un langage orienté objet créé en 1995 par la société Sun Microsystems, rachetée en 2009 par Oracle. Aujourd'hui, de nombreux logiciels utilisent ce langage, car il a la particularité et l'objectif d'être très facilement portable sur différents systèmes d'exploitation tel que Windows, Unix, GNU/Linux, Mac OS.
Le langage Java est en grande partie basé sur la syntaxe du langage C++ (langage très utilisé de nos jours).

Il existe essentiellement quatre grands Framework développé par Oracle :

- Java SE : destiné aux applications pour poste de travail
- Java EE : spécialisé dans les applications serveurs (contient un grand nombre d'API et d'extensions)
- Java ME : spécialisé dans les applications mobiles
- Java FX : spécialisé dans les applications liées aux cartes à puces et SmartCards

b. Langage XML

Figure 25 : Logo XML

Le langage XML (Extensible Markup Language) est un métalangage informatique de balisage générique qui dérive du SGML. L'objectif initial du langage XML est de faciliter l'échange automatisé de contenus complexes entre systèmes d'informations hétérogènes : « Son but est de permettre au SGML générique d'être transmis, reçu et traité sur le web de la même manière que l'est HTML aujourd'hui ». SGML est un langage de balisage utilisé notamment dans le langage HTML. Ce langage est très facilement reconnaissable grâce à l'usage des balise : < et > .

5. Androïd Studio

a. Généralités

Depuis 2015, Androïd Studio est l'environnement de développement utilisé pour l'IDE officiel de Google, délaissant Eclipse. Celui-ci est utilisé afin de créer des applications mobiles, grâce à l'édition de fichiers Java/Kotlin et les fichiers de configuration graphique XML. Androïd Studio propose notamment des outils supplémentaires tels que la gestion du développement d'application multilingues et la prévisualisation de la mise en page des différentes activités grâce à différentes résolutions d'écrans.

b. Le logiciel

Le logiciel se décompose en 4 parties majeures : les fonctionnalités (Java) / le design (XML) / les librairies et la configuration de l'application (Gradle) / le paramétrage des différentes activités (Manifest).

Tout d'abord, une application se décompose en plusieurs activités (page de navigation) qui sont formés d'une page de design et d'une page de fonctionnalités.

Figure 26 : Logo Gradle

Le Gradle est une fonctionnalité implémentée dans Androïd Studio. Celle-ci permet notamment de pouvoir intégrer de nombreuses librairies sans avoir besoin de les télécharger et de les installer. Cette page permet également de définir des caractéristiques tels que l'ID de l'application ; le SDK minimum requis pour faire fonctionner l'application ; la version de l'application et bien d'autres.

Le Manifest est un fichier de paramétrages. Celui-ci permet d'utiliser des permissions d'accès à certains dossiers du smartphone dont seul l'utilisateur peut accepter ou refuser. Il est également possible de définir certains paramètres des différentes activités : l'orientation (portrait ou paysage) ; l'affichage de l'Action Bar (barre d'actions du smartphone) ; le nom de l'activité ; l'icône de l'application ; etc.

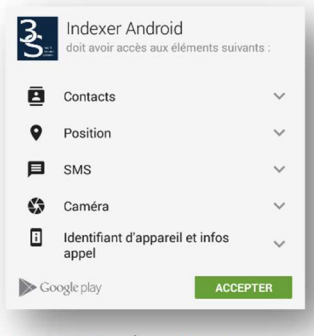


Figure 27 : Fenêtre d'Autorisations

Une page de design se programme grâce au langage XML. Pour ce faire, chaque page à un identifiant unique (commençant obligatoirement par une lettre minuscule) et qui est définit par un " .xml ". Il est possible de créer des éléments graphiques et leurs attribuer un identifiant unique.

On peut ainsi créer des zones de texte ; des boutons poussoirs ; des zones d'édition ; des images ; des cadres ; des menus ; des formulaires ; etc.

On peut également voir le rendu grâce à une fenêtre de prévisualisation entièrement paramétrable.

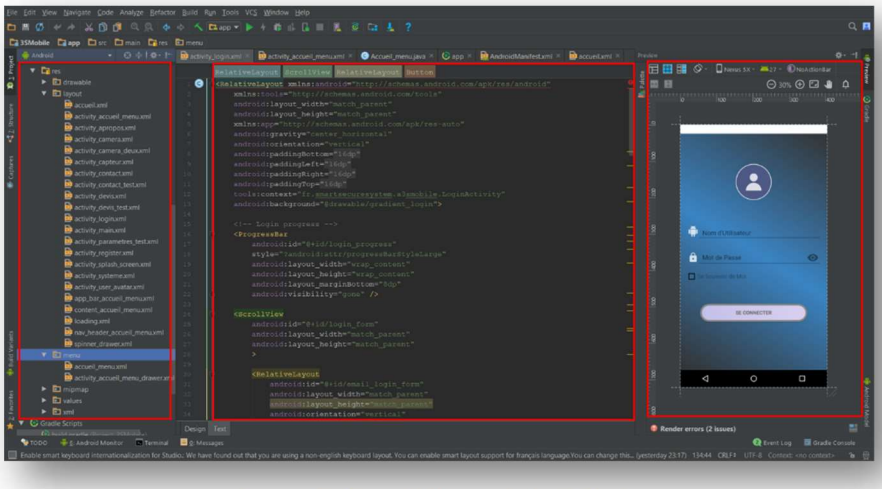


Figure 28 : Page XML sous Androïd Studio

Une page de fonctionnalités se programme grâce au langage Java. Pour ce faire, chaque page à un identifiant unique (commençant obligatoirement par une lettre majuscule) et qui est définit par un " .class ". Il est possible de récupérer les différents éléments graphiques grâce à leurs identifiants. C'est grâce à cela que l'on peut définir certaines actions lorsque l'utilisateur appuie sur son écran.

On peut ainsi afficher des valeurs ou des chaînes de caractères dans une zone de texte ; modifier l'état d'un bouton poussoir ; ouvrir un menu ; choisir un nombre parmi une sélection ; activer un script à distance ; etc.

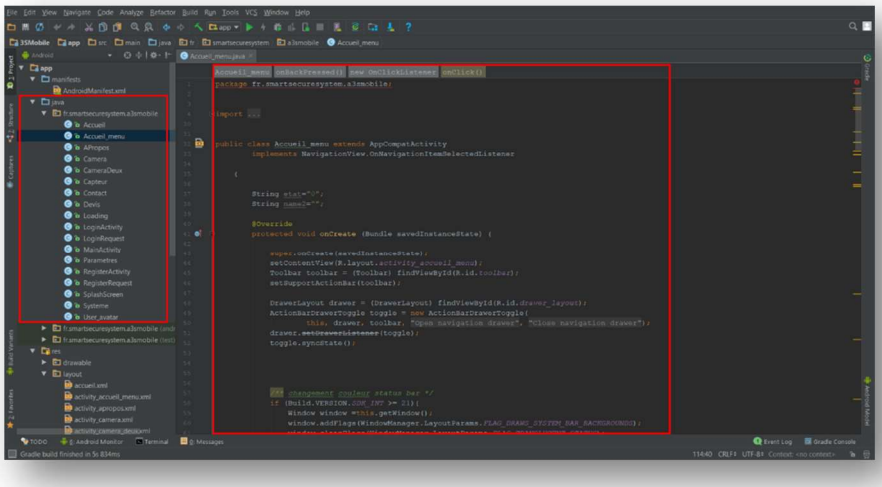


Figure 29 : Page Java Sous Androïd Studio

6. Fonctionnement d'une application

Une application mobile possède différents états distincts : Running ; Paused ; Stopped ; Destroyed. Bien évidemment, l'application ne peut être uniquement dans un seul état à la fois.
Pour schématiser cela, on peut comparer le fonctionnement d'une application mobile avec le fonctionnement d'un moteur de voiture, comme le montre les schémas ci-dessous.

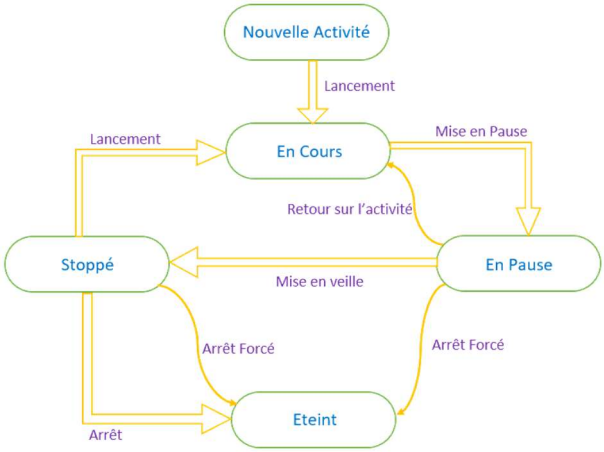


Figure 30 : Fonctionnement d'une application mobile
Cf : TP Module 3107 Réseaux Cellulaires de Mr Millet

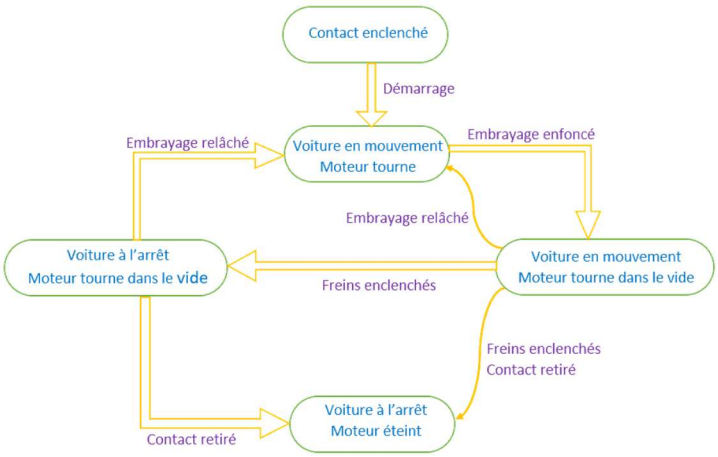


Figure 31 : Fonctionnement d'un moteur de voiture

7. Fonctionnement de l'application mobile " 3SMobile "

L'application mobile "3SMobile" est une application entièrement créée et développée par l'équipe de Smart Secure System. Celle-ci vous permet notamment de pouvoir consulter les vues des différentes caméras présentes chez vous ; mais également de voir pouvoir être informé si un capteur (fumée / présence / température) a détecté une anomalie.

Pour ce faire, il vous suffit simplement de vous connecter grâce aux identifiants de connexion fournis au moment de l'installation du matériel. Ces identifiants sont uniques pour chaque utilisateurs et identiques entre le site web et l'application mobile (utilisation de la même base de données pour les deux).
Au début du projet, la base de données entre le site web et l'application mobile n'était pas commune. Celle de l'application mobile était chez l'hébergeur web "www.webhost.com", filiale gratuite de "Hostinger", tandis que celle du site web était hébergée localement sur le Windows Server.

Une page de récapitulatif est présente sur l'application afin de résumer les informations personnelles de l'utilisateur. Dans une prochaine version de l'application, l'utilisateur aura la possibilité de modifier son mot de passe ainsi que choisir un avatar parmi une sélection prédéfinie.

L'utilisateur a également la possibilité d'allumer ou d'éteindre son système de surveillance depuis son smartphone.

Un système de devis est présent afin que l'utilisateur puisse connaître une estimation du prix du matériel si celui-ci souhaite intégrer ou alors améliorer son système de sécurité personnel. Nous proposons ainsi plusieurs équipements au détail : des RaspberryPi 3 ; des caméras ; des capteurs de fumée et des capteurs de mouvement.
Nous proposons également des forfaits sous forme de pack :

- Le pack Starter contient un capteur de fumée, une caméra, un RaspberryPi 3, l'accès à l'application et au site web.
  Son prix est de 99€ avec un abonnement de 36€/an soit 4€/mois.

- Le pack Deluxe contient un capteur de fumée, un capteur de présence, deux caméras, deux RaspberryPi 3, l'accès à l'application et au site web.
  Son prix est de 179€ avec un abonnement de 36€/an soit 4€/mois

- Le pack Intégral contient quatre capteurs de fumée, quatre capteurs de présence, quatre caméras, quatre RaspberryPi 3, l'accès à l'application et au site web.
  Son prix est de 499€ avec un abonnement de 36€/an soit 4€/mois, la 1ère année est offerte.

Un formulaire de contact est également présent afin que l'utilisateur puisse contacter le support technique si celui-ci rencontre un problème ou si celui-ci souhaite un renseignement.

Figure 32 : Accueil

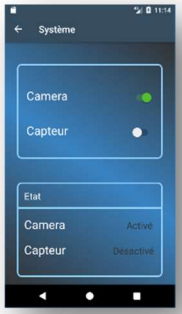


Figure 33 : Système

Figure 34 : Devis

Figure 35 : Paramètres

Figure 36 : Contact

VI. Hébergement Web

1. Nom de domaine

Fondé en 1988, l'entreprise 1&1 se spécialise dans l'hébergement de sites internet et le commerce en ligne. 1&1 propose des solutions d'hébergement web sous Linux ou Windows, des serveurs (dédiés, virtuels ou cloud), un service Wordpress (1&1 MyWebsite), des solutions de commerce en ligne, des solutions courriel, mais plus principalement des noms de domaines (.fr, .com, .eu, .net)

Un nom de domaine est un identifiant de domaine internet. Un domaine est un ensemble d'ordinateurs reliés à Internet et possédant des caractéristiques communes.
Smart Secure System possède un nom de domaine en .fr : smartsecuresystem.fr
Celui-ci a été acquis à la suite d'une offre de l'hébergeur 1&1. Cette offre avait pour but d'offrir un nom de domaine en .fr durant 1 an ainsi qu'un serveur web (incluant un accès FTP / une base de données MySQL / un espace web de 50Mo / un certificat SSL / un serveur de messagerie / un abonnement à Office 365).

Figure 37 : Logo 1&1

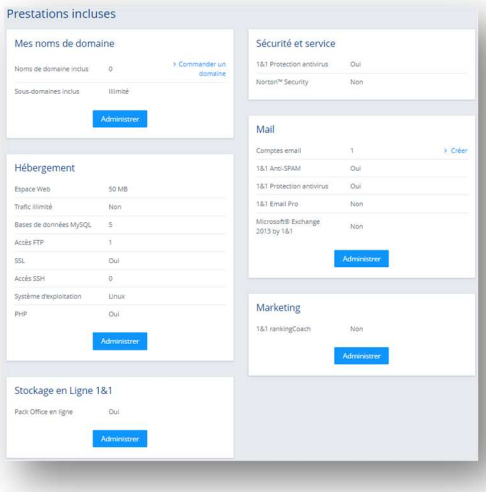


Figure 38 : Prestations incluses dans le pack choisi

Grâce au nom de domaine, si un utilisateur rentre l'adresse URL : "http://www.smartsecuresystem.fr" dans son navigateur, celui-ci sera directement redirigé sur notre site web.

Comme dit précédemment, nous disposons d'un serveur de messagerie, ce qui nous permet d'avoir une adresse mail de contact : contact@smartsecuresystem.fr
Cela nous as permis d'avoir pu mettre en place un système de report bug de façon simplifier pour l'utilisateur aussi bien sur l'application mobile que sur le site web.

VII. Hébergement Digital Océan

1. Généralités

Depuis 2011, Digital Océan est un fournisseur américain d'infrastructure de cloud payant basé à New-York. La société possède plusieurs centres de données situés un peu partout dans le monde. En décembre 2015, Digital Océan était la deuxième plus grande société d'hébergement au monde en termes d'ordinateurs connectés au Web.

Figure 39 : Logo Digital Océan

Pour proposer cela, Digital Océan utilise KVM comme hyperviseur avec la possibilité de choisir des tailles différentes (standard ou optimisé). Il propose également 7 distributions GNU/Linux (Ubuntu ; Debian ; Fedora ; FreeBSD ; CentOS ; CoreOS ; RancherOS) ; mais également diverses applications en un-clic (PhpMyAdmin ; MySQL ; WordPress ; LAMP ; MongoDB ; NodeJS ; etc).

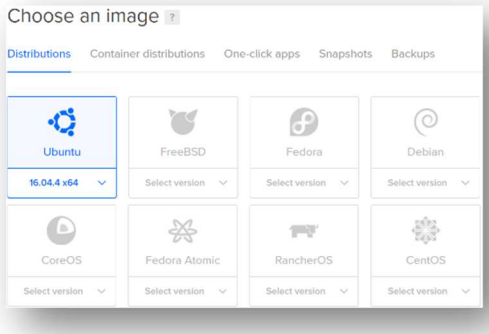


Figure 40 : Différentes images disponibles

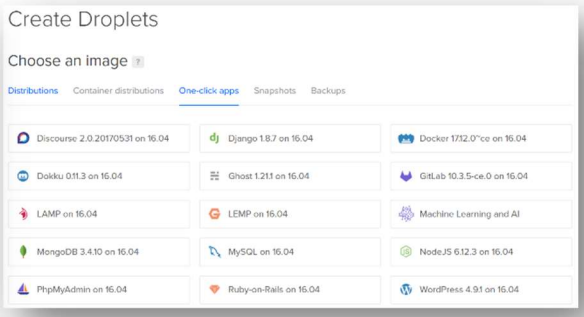


Figure 41 : Différents One-Click Apps disponibles

Toutes les machines peuvent être configurables en termes d'espace disque (entre 25GB et 3,75TB) ; de CPU (entre 1 CPU et 32 CPU) ; de mémoire (entre 1GB et 192GB) et en taux de transfert (entre 1TB et 12TB) : selon la configuration, le prix varie de 5$/mois à 960$/mois.

MEMORY	vCPUs	SSD DISK	TRANSFER	PRICE
1 GB	1 vCPU	25 GB	1 TB	$5/mo $0.007/hr
2 GB	1 vCPU	50 GB	2 TB	$10/mo $0.015/hr
3 GB	1 vCPU	60 GB	3 TB	$15/mo $0.022/hr
2 GB	2 vCPUs	60 GB	3 TB	$15/mo $0.022/hr
1 GB	3 vCPUs	60 GB	3 TB	$15/mo $0.022/hr
4 GB	2 vCPUs	80 GB	4 TB	$20/mo $0.030/hr
8 GB	4 vCPUs	160 GB	5 TB	$40/mo $0.060/hr
16 GB	6 vCPUs	320 GB	6 TB	$80/mo $0.119/hr
32 GB	8 vCPUs	640 GB	7 TB	$160/mo $0.238/hr
48 GB	12 vCPUs	960 GB	8 TB	$240/mo $0.357/hr
64 GB	16 vCPUs	1.25 TB	9 TB	$320/mo $0.476/hr
96 GB	20 vCPUs	1.88 TB	10 TB	$480/mo $0.714/hr
128 GB	24 vCPUs	2.5 TB	11 TB	$640/mo $0.952/hr
192 GB	32 vCPUs	3.75 TB	12 TB	$960/mo $1.429/hr

MEMORY	DEDICATED vCPUs	SSD DISK	TRANSFER	PRICE
4 GB	2 vCPUs	25 GB	5 TB	$40/mo $0.060/hr
8 GB	4 vCPUs	50 GB	5 TB	$80/mo $0.119/hr
16 GB	8 vCPUs	100 GB	5 TB	$160/mo $0.238/hr
32 GB	16 vCPUs	200 GB	5 TB	$320/mo $0.476/hr
64 GB	32 vCPUs	400 GB	5 TB	$640/mo $0.952/hr

Figure 42 : Tableau des différentes configurations

2. Migration du serveur

a. Migration

À la suite des problèmes énoncés juste avant, nous avons été contraints de changer de serveur d'hébergement web : il nous a donc fallu migrer le site web du Windows Server 2012 R2 vers le serveur Ubuntu de Digital Océan.
Pour ce faire, nous avons installé un serveur SSH sur le Windows Server 2012 R2. Une fois le service SSHD lancé sur le Windows Server 2012 R2, il ne reste plus qu'à autoriser l'accès SSH sur le pare-feu et en ouvrant le port 22. Une fois ceci fait, il faut mettre en place un échange de clé publique entre les deux serveurs. Le serveur SSH étant installé, il ne restait plus qu'à transférer les différents fichiers du Windows Server 2012 R2 vers le server Ubuntu grâce à la commande suivante : "scp -p root@138.68.134.243:dossier/fichier /var/www/html/ "

Figure 43 : Logo Protocole SSH

Dans un second temps, il était nécessaire de réinstaller tout ce dont nous avions besoin pour faire fonctionner le site web. C'est-à-dire Apache, PHP, MySQL et phpMyAdmin.
Il a également fallu recréer toute la base de données présente sur le Windows Server 2012 R2. C'est grâce à cette migration que le site web ainsi que l'application mobile ont pu avoir une base de données commune pour les enregistrements d'utilisateurs.

Figure 44 : Logo de Fail2ban

Enfin, comme le serveur est hébergé sur un Cloud, celui-ci est accessible par n'importe quelle personne possédant une connexion Internet. Cela présente donc un grand problème de sécurité. Pour y remédier, nous avons décidé d'installer un système de sécurité du nom de "Fail2ban". C'est un outil qui permet de bannir les adresses IP qui ont obtenu un trop grand nombre d'échecs lors de l'authentification. Pour ce faire, les règles du pare-feu sont mises à jour afin de rejeter ces adresses IP (les règles sont bien évidemment définies par l'utilisateur). C'est ainsi que si un individu échoue cinq fois au moment de son authentification, celui-ci se retrouve bannie pour un long moment. Cette sécurité est active pour l'authentification SSH ; phpMyAdmin ainsi que pour Apache.

VIII. Hébergement Windows

1. Choix du serveur

Au début du projet, on a décidé d'utiliser un serveur Windows Server 2012 R2, pour centraliser les données, car on voulait mixer les systèmes d'exploitation.

2. Outils

a. IIS

Nous avons donc installé IIS (Gestionnaires de Services Internet).

Il s'agit d'un serveur web sur Windows comportant plusieurs fonctionnalités comme FTP, PHP et d'autres …

Il permet d'avoir une interface et un fonctionnement sécurisé tout en hébergeant simplement vos sites et applications Web.

b. MySQL 5.5

C'est un SGBD (Système de Gestion de Base de Données) par le cmd et qui fait le lien vers les bases de données SQL, permettant l'administration de nos bdd et tables.

Il est complètement Open-Source et embarque un langage spécifique pour la connexion aux bases de données.

3. Service no-IP

Le service no-IP, souvent plus connu sous le nom de DNS Dynamique, consiste à créer une redirection d'URL entre un sous-domaine et votre box internet. Pour mettre en place un service no-IP, il suffit de se rendre sur un site qui propose cela : "https://www.noip.com/". Ensuite, il suffit de créer un site qui aura pour extension ".ddns.net". Enfin, il ne reste plus qu'à se rendre sur l'interface de la box internet (souvent accessible via l'adresse IP suivante : 192.168.1.1 ou 192.168.1.254) ; et à activer la redirection "Dyn DNS" et ouvrir le port 80 et 443 sur l'adresse IP souhaitée.
Désormais, votre site sera accessible via l'URL se terminant par ".ddns.net". On peut schématiser les différents échanges grâce au schéma ci-contre :

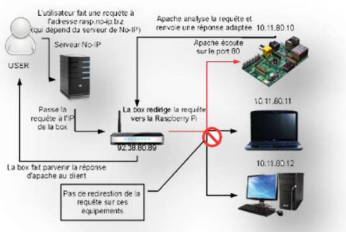


Figure 45 : Schéma échanges no-IP

Antoine, caméras

Introduction :

Au cours de notre projet de domotique Smart Secure System j'avais pour rôle de générer un flux vidéo grâce à un Raspberry Pi et une caméra pour ensuite pouvoir le visionner en direct depuis l'ordinateur ou le portable de l'utilisateur, que ce soit sur le réseau Wi-Fi ou sur le réseau des données mobiles. En parallèle, il fallait que la caméra détecte un trop grand changement de pixels, par exemple dans le cas où une personne mal intentionnée comme un voleur s'introduirait dans la maison de notre client. A ce moment-là, le système avertirait le propriétaire via une alerte sur son smartphone.

Figure 46 : Raspberry Pi et Caméra

Mes recherches :

Après avoir installé la dernière version de Raspbian, j'ai commencé par rechercher sur Internet les solutions existantes pour pouvoir streamer à partir du Raspberry Pi.

J'ai vu MotionEye, qui crée directement une interface de gestion du flux avec tous les paramètres associés et le visionnage de la vidéo sur une page Web. Cette page est accessible en se connectant sur l'adresse IP du Raspberry Pi. Cependant le logiciel est trop rigide pour effectuer les actions que l'on veut, nous sommes limités par les paramètres affichés.

J'ai découvert OpenCV (Open Computer Vision) qui est une bibliothèque graphique libre spécialisée dans le traitement d'images en temps réel. Une fois celle-ci installée et compilée dans un environnement virtuel, elle nous permet d'utiliser des modules et bibliothèques que l'on importe dans des scripts python pour le bon fonctionnement de la vidéo. A noter que si cette bibliothèque graphique est utilisée en dehors d'un environnement virtuel, celle-ci ne fonctionne pas. Nous pouvons entre autres utiliser une fonction d'envoi de mail par le protocole SMTP dans un script pour prévenir notre client qu'un problème a été détecté. Sur son téléphone le propriétaire reçoit alors une capture photo de la caméra de la personne en question encadré d'un rectangle, au moment où celle-ci a découvert le problème.

Dans un deuxième temps, je me suis penché encore une fois sur du code python afin de créer des scripts d'activation et de désactivation du système à distance. En d'autres termes l'utilisateur peut, que ce soit à partir du site Web ou de l'application mobile, activer ou désactiver la ou les caméras de son système domotique.

Travail pratique :

Dans un premier temps j'ai installé Raspbian Stretch sur mon Raspberry Pi, dernière version en date du système d'exploitation libre et gratuit basé sur Debian GNU/Linux.

Après avoir configuré plusieurs paramètres comme le Wi-Fi, l'activation de la caméra ou encore le SSH, je commence mes recherches sur la première solution que nous avions envisagée : MotionEye. J'ai donc installé le logiciel et pris connaissance de celui-ci. Comme dit précédemment cette solution ne pouvait pas être retenue : en effet nous n'avions aucun accès à aucun code mais uniquement aux paramètres graphiques directement réglables sur la page Web. De plus ce logiciel remplaçait totalement le système d'exploitation sur le Raspberry, ce qui veut dire que nous ne pouvions même pas utiliser un terminal pour effectuer des commandes sur notre système.

En poursuivant mes recherches je suis donc tombé sur OpenCV, qui propose une bonne alternative en comparaison à MotionEye puisqu'avec celui-ci je peux avoir un total accès aux codes python qui le compose et ainsi les modifier. Je peux également conserver mon système d'exploitation Raspbian Stretch et taper des commandes via un terminal. Petit bémol : cette solution doit être dans un environnement virtuel pour fonctionner. Enfin, ce n'est pas une obligation mais c'est fortement recommandé. Cela permet d'avoir des environnements python isolés, notamment dans le cas où plusieurs projets sont développés.

Après avoir compilé et installé OpenCV, je pouvais enfin m'interroger sur les scripts qui plus tard permettront la diffusion du flux vidéo. Justement, par la suite je tombe sur un site Web où un projet similaire a déjà été effectué. Celui-ci est composé de plusieurs scripts python, je me suis donc basé sur ceux-ci pour la suite du projet. Une fois les avoir modifiés à ma guise, j'obtiens pour la première fois un flux vidéo que l'on peut visionner sur le réseau Wi-Fi via l'adresse IP du Raspberry Pi.

Maintenant j'avais un tout nouveau but : permettre le visionnage du flux vidéo en dehors du réseau local, c'est-à-dire depuis le réseau des données cellulaires. Pour cela j'effectue des modifications sur le routeur pour faire une redirection de ports. En effet il faut savoir que l'adresse IP du Raspberry Pi est accessible depuis n'importe quel appareil sur le réseau local, mais celle-ci est modifiée si nous quittons ce même réseau. La redirection de ports permet donc une liaison entre le monde extérieur (Internet) et notre adresse IP locale.

Je place également mon Raspberry Pi dans la DMZ : celle-ci est une zone intermédiaire entre Internet et le réseau local. Si le Raspberry Pi se fait pirater et que l'auteur a pour but de diffuser un virus sur notre réseau privé via celui-ci, il ne pourra tout simplement pas. En plus de définir un nouveau mot de passe sécurisé il est important de conserver une sécurité maximale, car nous rappelons qu'une fois que le Raspberry Pi est accessible depuis Internet, n'importe quel individu mal intentionné peut vouloir pirater notre système. La DMZ permet de palier à ce problème.

À ce moment précis nous pouvions accéder au Stream vidéo sur le réseau local via l'adresse IP privée du Raspberry Pi ou sur les données mobiles en tapant l'adresse IP publique de celui-ci. Cependant cette adresse publique peut être changée par le fournisseur d'accès à Internet, on dit qu'elle est dynamique : cela permet une meilleure sécurité. Je décide donc d'utiliser NoIp, logiciel en ligne qui permet de lier l'adresse IP publique avec un nom de domaine. Cela même si l'adresse est dynamique.

Désormais, nous pouvons accéder au flux vidéo en tapant le nom de domaine dans l'url de notre navigateur au lieu de taper l'adresse IP publique.

Nous arrivons maintenant aux scripts d'activation et de désactivation du système à distance. Après quelques recherches je réussis à créer ces deux scripts et je commence à réfléchir par quel moyen ceux-ci pourraient être exécutés depuis le serveur, là où notre site Web est configuré. Je crée deux nouveaux scripts qui seront sur le serveur, un pour activer le système et un autre pour le désactiver. Ces deux codes appelleront tout simplement ceux présents sur le Raspberry en local pour qu'ils s'exécutent.

Nous sommes maintenant au moment où il ne manque plus que le lien entre le site Web et la page diffusant le flux vidéo. Avec mon collègue chargé de configurer notre site nous organisons une mise en page similaire à celui-ci sur la page du Stream et créons le lien. Désormais, lorsque nous cliquons sur l'onglet « Caméras » présent sur le site Web de notre système domotique nous sommes redirigés sur la page où il y a le flux vidéo.

Problèmes rencontrés :

Le tout premier problème rencontré s'est passé lors de l'installation du système d'exploitation sur le Raspberry Pi. En effet une fois celui-ci installé il m'était impossible de récupérer des fichiers sur le réseau et donc de faire des mises à jour. J'ai donc installé Raspbian Stretch grâce à une autre source.

Le problème suivant m'a donné du fil à retordre. Entre deux sessions de travail sur le projet, le Stream vidéo ne voulait tout à coup plus fonctionner alors que je n'avais effectué aucun changement au niveau logiciel. Le Raspberry Pi ne détectait plus la caméra, qui était pourtant connectée. Ceci a bien sûr posé problème et le script de démarrage du flux vidéo ne se lançait plus. N'ayant pas trouvé la raison de ce problème, je décide de reset le Raspberry Pi. Je n'avais plus qu'à tout recommencer. Mais par grande surprise la caméra n'était toujours pas détectée : je conclus donc que si ce n'était pas un problème logiciel, cela devait être un problème physique. Par la suite je me suis rendu compte que la caméra était défectueuse, j'ai donc remis à zéro mon système pour rien.

Troisième problème qui est arrivé sur la fin du projet et qui est également le plus incompréhensible : lors de connexions multiples par SSH sur le Raspberry (nous mettions en place une nouvelle mise en page pour la page du flux) et suite à un transfert d'une image par SCP la bibliothèque graphique OpenCV n'était plus reconnue. L'environnement virtuel avait soudain disparu et un fichier de configuration important avait été modifié. Gros problème gros moyen : je réinitialise encore une fois mon système d'exploitation.

Axes d'amélioration :

Au niveau des axes d'amélioration je vais séparer cette partie en deux. D'une part je vais parler de ce que j'aurais pu améliorer dans ma partie si j'en aurais eu les moyens et d'autre part ce qu'on aurait pu améliorer, au sein de notre équipe, pour un rendu final de notre projet encore plus abouti.

En ce qui concerne la partie caméra, nous aurions pu approfondir les scripts d'activation et de désactivation du système à distance. Le fait que le serveur soit décentralisé et non sur le réseau local nous a compliqué la tâche. Avec plus de temps je pense que nous aurions pu implémenter une nouvelle page sur le site Web pour activer/désactiver le système, ainsi que de créer un nouvel onglet sur l'application mobile Android pour effectuer le même type d'action.

Un autre axe d'amélioration aurait été apprécié : le visionnage d'une même caméra sur plusieurs appareils et plusieurs réseaux. En effet nous pouvons bien voir le flux vidéo mais sur une seule page à la fois. C'est-à-dire que si l'utilisateur regarde le Stream depuis son portable avec ses données mobiles, personne d'autre ne pourra visionner le flux, que ce soit sur le réseau Wi-Fi ou de l'opérateur. Malgré des heures de recherches et de modifications des codes je n'ai pas su passer outre ce problème.

Si je pouvais améliorer quelque chose vis-à-vis de mon équipe pendant les mois où nous avons développé ce grand projet ce serait la communication au sein du groupe. Même si des réunions étaient organisées assez régulièrement et que les différents problèmes qu'un membre pouvait rencontrer nous étaient partagés, une communication et une cohésion sans faille est toujours difficile à mettre en place à notre niveau et pour un groupe composé de six étudiants.

Rémi, marketing et gestion de groupe

Introduction :

Suite à un vote, je me suis retrouvé chef de groupe. Mon rôle a été de coordonner les 6 membres de l'équipe et de développer l'aspect commercial du projet.

Mes recherches :

J'ai commencé à chercher des outils pour faciliter la communication à l'intérieur du groupe, j'ai trouvé l'outil proposé par Microsoft : Planner. C'est un outil de planification de tâches collaboratif où chaque membre peut indiquer aux autres où il en est dans son travail. C'est en utilisant Planner que j'ai découvert Sharepoint, une suite d'outils en ligne destinée à la gestion de projet, très utile pour partager des documents, notes de réunions, …

Figure 47 : Logo Microsoft Planner

Mon travail s'est divisé en deux parties, de la gestion de groupe, faire en sorte que tout le monde travail main dans la main malgré un groupe assez important, ainsi que tout l'aspect communication et marketing autour du projet.

Nous avons réalisé un *freemind* d'après les recherches effectuées par les équipes afin d'avoir un fil conducteur. Ce schéma a été compléter durant toute la réalisation du projet, car au fur et à mesure de l'avancement, des solutions ont était mise de côté car nous n'étions pas spécialistes de la domotique.

Depuis le début du projet, l'idée était de développer ce projet comme étant une startup naissante. Nous avons donc décidé de nous hiérarchiser entre nous. Pour déterminer le chef de groupe, nous avons tout simplement votés, il en est ressorti que j'avais une voix de plus que Lucas, c'est pour cela que nous avons décidés qu'il y aurait un chef ainsi qu'un adjoint. J'ai par la suite établi 3 différentes équipes en fonction de leur domaine de compétence, pour simplifier les échanges. Communiquer à 6 est assez compliqué mais a été facilité par le nommage d'un responsable par équipe. Ce responsable a la parfaite connaissance de l'avancement de son binôme, il peut donc échanger avec le chef de groupe pour discuter des différentes *dead-line* à respecter ainsi que des problèmes rencontrés. Cela gagne du temps pour tout le monde car le chef de groupe n'a qu'à s'entretenir avec 2 personnes au lieu de 4, ce qui permet de faire très rapidement le point sans forcément organiser des réunions chronophages.

J'ai pris l'initiative d'organiser des réunions régulières pour (stimuler le groupe) faire des points tous ensemble sur l'aspect technique et humain du projet. Dans la joie

et la bonne humeur car « un salarié heureux est un salarié performant »[1]. Ces réunions étaient aussi l'occasion de régler les différends entre les membres du groupe. Il fallait faire retomber la pression au plus vite afin que ça ne mette pas en péril l'avancement du projet. Les responsables d'équipe m'ont à quelques reprises fait part de déconvenues au sein de leur équipe. J'ai par la suite contacté les personnes en question pour régler le problème avant qu'il s'envenime. La gestion d'équipe sans réelle hiérarchie est assez compliquée car lorsque l'on demande des travaux à rendre pour une date donnée avec des consignes claires, il est assez compliqué de faire respecter ces *dead-lines* et les consignes, tout est un travail d'ajustement et de négociation avec les différentes parties. La gestion des équipes a été compliqué mais très enrichissante et donne un aperçu des difficultés de l'entreprise.

Pour les besoins de la porte ouverte, nous avons réalisés une vidéo sous forme de publicité, avec un côté décalé pour susciter l'attention des visiteurs. J'ai écrit un pitch avec les idées principales pour expliquer succinctement à mes camarades le déroulé de la vidéo. Après leur accord, nous avons écrit le scénario détaillé avec Lucas afin de nous éviter de tourner des scènes inutiles. Nous avons été contraints par la météo de tourner la vidéo en deux jours. Malgré toute notre attention, nous pouvons voir des faux raccords a beaucoup d'endroit, eh oui, nous ne sommes pas réalisateurs professionnels et cela se ressent. Après le tournage, la longue phase de montage à suivie. Puis la sélection des musiques et de la voix off. Pour la bande son, nous avons choisis des musiques connues afin d'entrainer le spectateur. Les droits musicaux n'ont pas été un problème car la SACEM[2] m'a indiqué que tant que la vidéo n'était pas diffuser au grand public (Youtube,…) il n'y avait pas de droits à payer. Nous avons choisi de faire intervenir une voix off féminine pour casser avec la vidéo quasiment uniquement interprétée par des hommes.

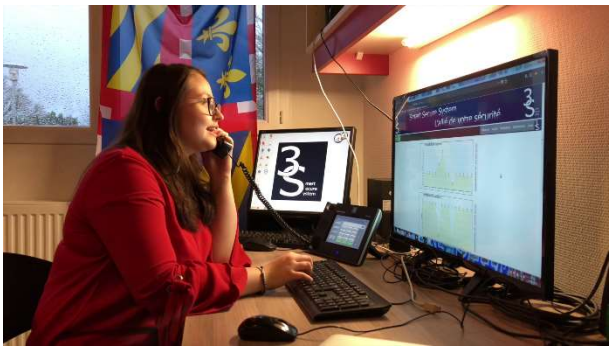


Figure 48 : Capture d'écran de la vidéo de présentation

Pour continuer dans l'aspect communication-marketing, nous avons aussi réalisé un flyer de présentation qui regroupe les principales informations relatives à notre projet, vous pouvez le retrouver en annexe 4. Nous avons aussi décidé de créer un logo pour continuer dans l'esprit start-up. Ce logo s'est accompagné d'une charte graphique avec comme par exemple la police « Tahoma » que nous nous sommes imposés dans tous les documents relatifs au projet (vidéo, flyer, rapports, …). Nous avons aussi créé des badges à l'effigie de Smart Secure System pour renforcer l'aspect startup.

[1]Citation de Marc-Alphonse Forget / consultant, coach et formateur auprès de grandes entreprises et de structures publiques

[2] SACEM : Société des auteurs, compositeurs et éditeurs de musique

Je remercie mes collègues pour leur travail de qualité et leur bienveillance qui a largement contribué au bon déroulement du projet.

Lucas, capteurs, communication client-serveur & production

Introduction :

Je me suis occupé pour le projet 3S de deux grandes parties :

Une technique : la gestion de la partie capteur (infrarouge, température…), ce qui est installé sur le client ainsi que de la communication des informations capturées avec le serveur.

L'autre moins technique mais tout aussi essentiel : la production, et cela en collaboration avec mon camarade Rémi. La production passe par la gestion des réunions mensuelles, les objectifs à atteindre, les différents rapports, carte mentale, la communication autour du projet, les différents supports, gestion de la soutenance et l'avancement général du projet. Pour cela referez-vous aux parties Introduction et Gestion du projet.

Mes recherches :

Partie capteur :

Le postulat de base était que le capteur devait être installé sur une carte électronique assez simple à mettre en place, chez le client et communiquant par wifi. Deux solutions donc :

-RPI (3)

-Arduino

Le choix c'est donc porté sur les Raspberry pi étant donné la facilité avec laquelle on peut se les procurer pour le projet ainsi que mon manque d'expérience sur Arduino.

Ensuite c'est posé la question de quel capteur ?

-Faisceau infrarouge -> obligatoire pour un système de sécurité-intrusion comme 3S.

-Capteur de présence ->quasiment nécessaire néanmoins sera simulé car non acheté.

-Capteur de température -> dans la dynamique anti-incendie du projet, n'est que simulé.

-Alarme et/ou micros-> très bon axe d'amélioration du projet, n'a pas été retenue par manque de temps et moins intéressant à mettre en place.

La totalité de ces capteurs se branchent au RPI via les bornes GPIO. Une tablette Labdec ainsi qu'un certain nombre de fils ont été requis pour mener à bien les expérimentations.

Au niveau du langage pour la gestion des capteurs, python c'est trouvé comme une évidence, car en plus d'être extrêmement performant avec des scripts usant de la GPIO il permettait de mettre en place des simulations pour les capteurs manquants. De plus plusieurs tutoriels basiques existent sur internet.

Vient ensuite la communication client-serveur :

Le postulat de base était d'envoyer les données de façon simple dans un dossier prévu à cet effet sur le serveur. Après une réflexion avec le tuteur ainsi que les personnes en charge du serveur il a été décidé d'envoyer les données sous la forme d'un .txt où chaque ligne correspondrait à la variable d'un capteur.

Les technologies potentielles pour cela sont donc :

-Les sockets : créer directement dans un langage (python) des scripts sockets client et serveur permettant l'envoi du fichier.

-FTP : protocole utilisant le port 22 ainsi que des fonctions prédéfinies pour le dialogue entre client et serveur. Nécessite un script client ainsi qu'un serveur FTP installé.

L'avantage principal des sockets est un control total de la technologie utilisée pour le projet.
Bien sûr la solution FTP est plus facile à mettre en place et sera donc gardé comme solution B, aux cas où.

Travail pratique & problèmes rencontrés :

Capteurs :

Au niveau capteur le travail c'est effectué de façon assez fluide. Il a fallu dans un premier temps me renseigner sur les branchements adéquats sur la GPIO.

Voir GPIO_RPI3 en annexe

Puis mettre en place des scripts en python pour lire les données des différents capteurs.

C'est un travail assez facile car il existe beaucoup de tutoriel en python sur la gestion des capteurs en GPIO. Il a bien évidement fallut les modifier pour correspondre au projet.

```
if(GPIO.input(3) == 1):
        faisceau = 0
if(GPIO.input(3) == 0):
        faisceau = 1
else faisceau = 0
```

```
temp1=GPIO.input(4)
```

A la base, une autre solution pour les simulations avait été envisagé mais elle a été abandonnée par manque d'optimisation : elle utilisait à elle toute seule un script particulier en Bash. Il a été bien plus facile d'incorporer les simulations aux scripts python des capteurs.

```
#!/bin/bash
let "faisceau = $[($RANDOM % ($[1 - 0] + 1))]"
let "temp1 = $[($RANDOM % ($[100 - 1] + 1)) + 1]"
let "temp2 = $[($RANDOM % ($[100 - 1] + 1)) + 1]"
let "presence = $[($RANDOM % ($[1 - 0] + 1))]"
let "fume = $[($RANDOM % ($[1 - 0] + 1))]"

>data.txt
echo $faisceau >> data.txt
echo $temp1 >> data.txt
echo $temp2 >> data.txt
echo $presence >> data.txt
echo $fume >> data.txt
```

Les capteurs manquants ont donc été remplacés par des simulations crédibles à l'intérieur du python.

```
temp2=random.randint(18,25)
```

```
presence=random.randint(0,1)
fume=random.randint(0,1)
```

Une fois les variables lues où simulées il a fallu les écrire dans un fichier texte prédéfinis.

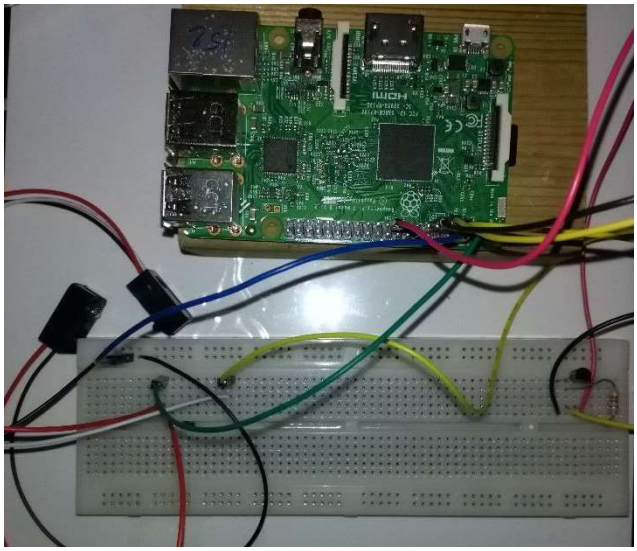
```
fichier = open("/home/pi/data.txt", "w")
fichier.write(str(faisceau)+"\n"+str(presence)+"\n"+str(fume))
fichier.close()
```

Le postulat de base était d'avoir un fichier « data.txt » où une ligne correspondrait à une variable, mais en travaillant en collaboration avec Yvan qui a géré la partie serveur l'on s'est rendu compte qu'il était plus facile pour lui et plus claire de différencier clairement les variables binaires des variables analogiques (température).

Ainsi l'on s'est mis d'accord sur la création de deux fichiers .txt. Le plus simple et claire a donc été de faire deux script python différent pour chaque fichier de donnée.

Voir capteur.py et capteur2.py en annexe

Tout cela bien sûr a été entrecoupé de phase de test, vérifier les branchements, que les composant ne chauffe pas trop, que les programmes marchaient sur le temps, que les informations était correcte et dans le cas des simulations, réalistes…



Communication :

Une partie un peu plus corsée, deux technologies ont été travaillées.

Sockets : *Les sockets servent à communiquer entre deux hôtes appelés Client / Serveur à l'aide d'une adresse IP et d'un port ; ces sockets permettront de gérer des flux entrant et sortant afin d'assurer une communication entre les deux (le client et le serveur).*

Ainsi donc, après m'être renseigné sur la technologie j'ai commencé à éditer des scripts trouvés sur internet.

Voir script_socket_client et script_socket_serveur en annexe.

L'idéal aurait été de créer un script client qui se connecterais au serveur et qui enverrai le fichier ainsi qu'un script serveur qui irait écouter en continue et n'accepterai que la connexion du client cible.

Il est assez facile de trouver des scripts utilisant la technologie socket pour importer un fichier mais beaucoup moins pour exporter un fichier du client au serveur.

Mon premier test pour importer un fichier (j'ai simulé le serveur avec un deuxième RPI) à marcher, ensuite n'ayant pas trouvé comment exporter les données j'ai simplement eu l'idée d'inverser les scripts :
C'est-à-dire que le client écoute en continue et le serveur se connecte ponctuellement pour récupérer un fichier cible.

Ainsi donc ce script à vocation à être mis en place sur le serveur et s'activité de façon périodique.

```
file_name = input(">> ") # utilisez raw_input() pour les anciennes versions python
s.send(file_name.encode())
file_name = 'data/%s' % (file_name,)
r = s.recv(9999999)
with open(file_name,'wb') as _file:
    _file.write(r)
print("Le fichier a été correctement copié dans : %s." % file_name)
```

On demande quel est le fichier à récupérer puis on envoie le contenu et on créer une copie de ce fichier puis on écrit les données à l'intérieur.

Puis, plus tard, après à plusieurs tests et recherches approfondies sur les sockets et leurs utilisations en python, j'ai trouvé une solution pour exporter le fichier du client.

Ceci me permettait donc une utilisation beaucoup plus simple des scripts : celui client s'activera ponctuellement et celui serveur en continue, comme dans la situation idéale.

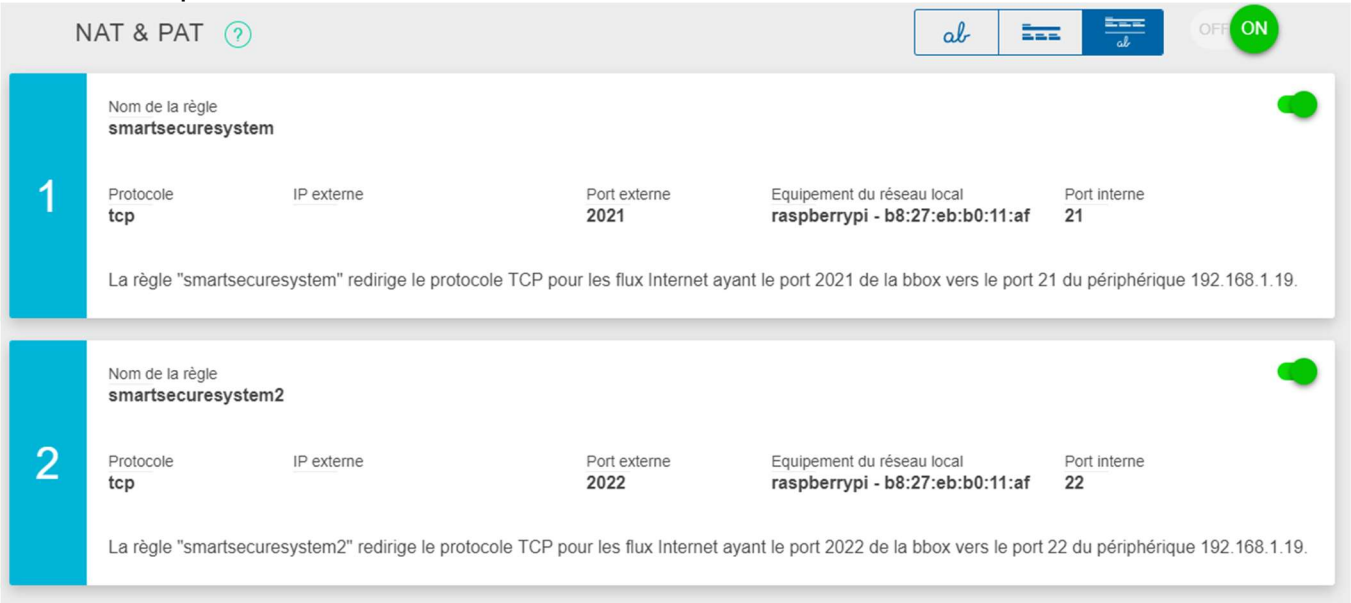
J'en ai aussi profité pour travailler sur le thread de mes scripts pour permettre un système multi-utilisateur multi-flux

```
def run(self):

    print("Connection de %s %s" % (self.ip, self.port, ))

    r = self.clientsocket.recv(2048)
    print("Ouverture du fichier: ", r, "...")
    fp = open(r, 'rb')
    self.clientsocket.send(fp.read())

    print("Client déconnecté...")
```

Voir rpi.py.old en annexe pour le programme complet.

Le plus intéressant aurait été de parler du script serveur, que j'ai essayé de construire avec des threads, de sécuriser, d'optimiser… Mais qui malheureusement se trouve sur une carte SD qui a été corrompus sans réelle raison à la suite de la démonstration lors des JPO. C'est après cette perte de donnée (aucune sauvegarde n'avait encore été faite) et plusieurs problèmes d'intégration python et d'automatisation du script serveur sur le serveur Windows qu'il a été jugé en réunion avec les équipes en charge du serveur et sachant que les deadlines pour cette partie avaient déjà été franchis que j'allais essayer une autre solution, plus simple à mettre en place coté serveur.

Tout au long de la création de la communication client-serveur il ne faut pas oublier de prendre en compte que nous somme derrière une BOX (du côté client et du coté serveur), il y a donc différentes redirections de port à effectuer que ce soit dans une dynamique de sécurité ou juste pour que la communication marche. Tout cela se gère depuis l'interface d'administration de la box en particulier, ici un exemple avec la mienne pour une connexion à distance FTP et SSH.



L'autre solution a donc été FTP : *File Transfer Protocol (protocole de transfert de fichier), ou FTP, est un protocole de communication destiné au partage de fichiers sur un réseau TCP/IP. Il permet, depuis un ordinateur, de copier des fichiers vers un autre ordinateur du réseau.*

Premièrement il a fallu mettre en place un serveur FTP en écoute continue, et un minimum sécurisé : n'autoriser que mon client cible, fermer le reste pour éviter des dénis de service, gérer les permissions avec pare feux installés surtout consistant la méthode passive de FTP (que j'utilise).

C'est le grand avantage comparé à la méthode sur sockets : le serveur est plus simple à mettre en place (graphique sous Windows), plus simple à débogué, et ne demande

pas d'autre technologie pour qu'il puisse écouter en continue (compris dans l'installation de base, en mode « standalone »).

Ensuite il a fallu créer des scripts clients sur le RPI pour se connecter et transférer les données.

```
ftp -p -n $HOST $PORT<<END_SCRIPT
quote USER $USER
quote PASS $PASSWD
binary
put $FILE
quit
END_SCRIPT
```

On notera le « -n » qui signifie l'utilisation passive de FTP, très utile sachant que notre client se trouve derrière une BOX et est donc natté (avec tout le lot de permission et redirection que cela comprend).
En méthode passive c'est le client qui gère toutes les connexions : de donnée (port dynamique) et de communication des informations de contrôle (21 du serveur)

Sachant qu'à la base un seul fichier était prévu, un script Shell usant la commande ftp pour transférer un fichier était suffisant, ensuite, j'ai choisi d'en faire deux par souci de clarté.

On a donc deux groupes de scripts qui gère de A à Z l'acquisition, l'écriture et l'envoie des données

```
capteur2.py  capteur.py  data2.txt  data.txt  ftp2.sh  ftp.sh
```

Maintenant il faut que les données soient envoyées périodiquement. Et c'est surtout pour cela qu'il a été décidé que deux fichiers de données, deux scripts d'acquisition capteur et deux script ftp seraient utilisés :

Capteur binaire en continue -> système de notification d'alerte.

Capteur de température toute les 1 min -> mise à jour du graphique sur le site périodiquement.

Pour cela j'ai créé deux scripts pour « superviser » l'ensemble des deux opérations en parallèles

```
#!/bin/sh
while true
do
        /home/pi/capteur2.py&
        /home/pi/ftp2.sh
        sleep 60
done
```

1) Pour les capteurs de températures

```
#!/bin/sh
while true
do
        /home/pi/capteur.py
        /home/pi/ftp.sh
done
```

2) Pour les capteurs binaires

Via une boucle « while true » et le « & » je m'assure que les scripts python et ftp s'exécute en continue et l'un après l'autre, pour être sûr que les données envoyées soient à jour (dernière valeur lue).

(Si vous vous demandez pourquoi je n'ai pas simplement utilisé le crontab c'est parce que le daemon crontab ne permet pas de gérer des automatisations en dessous de toutes les minutes)

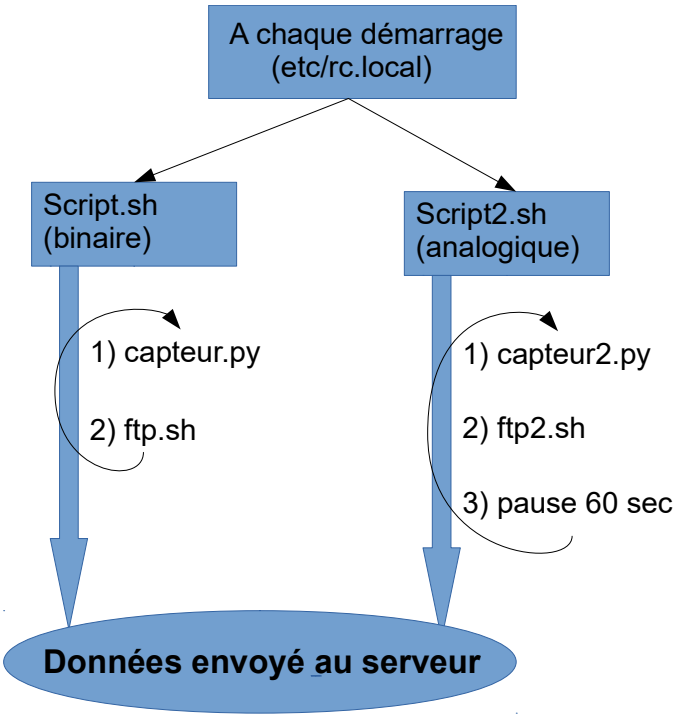
Ensuite il faut bien évidement que ces deux scripts soit activé une fois chez le client pour que l'on rentre dans la boucle infinie.
Alors bien sûr dès que le RPI s'éteint il faut tout relancer.

Pour parer à ça j'ai utilisé le fichier rc.local :

```
if true
/home/pi/scrypt.sh&
/home/pi/scrypt2.sh&
then
exit 0
fi
```

Rc.local est un fichier qui sur plusieurs distribution (utilisant inet.d) de Raspbian est exécuté au début de la session utilisateur (après le démarrage). Et donc en y incorporant nos scripts, permet de rentrer dans la boucle, même en cas de reboot (coupure de courant par exemple). J'exécute donc les deux scripts d'activation des scripts python et ftp à chaque boot du RPI.

Axes d'amélioration et conclusion :



En conclusion, rc.local exécute à chaque boot deux scripts Shell responsable de l'exécution parallèle des deux groupes (binaire et analogique) comprenant des scripts capteurs (en python) et ftp (en shell).

Différents composants expliqués plus haut s'assurent du bouclage en continue des scripts dans le bon ordre et de la temporisation nécessaire.

Les axes d'amélioration peuvent être nombreux, premièrement avec plus de capteur j'aurais pu créer un système plus complet : alarme, micros, capteur de présence… Et pourquoi pas sécuriser entièrement une pièce (fenêtres, portes…).

Un autre axe d'amélioration aurait pu être l'utilisation finale des sockets pour créer sa propre technologie de communication sécurisée, ou utiliser SFTP. En fin d'année de PEL nous avons survolé les sockets sur Java, et après de courte recherche, je suis profondément convaincu que cela peut être la solution idéale.

SFTP était à la base prévue, malheureusement suite à un choix de l'équipe en charge du serveur, le serveur a migré de Windows à Linux. J'ai donc décidé de laisser ça de côté pour faire fonctionner ma communication sur le serveur Linux, ce qui malheureusement faute de temps et de bug dans les fichiers de configuration n'a pas pu se faire.

Une solution de redirection des deux fichiers de donnée du serveur Windows à Linux via des commandes scp est actuellement en cours de test pour avoir une démonstration fonctionnelle à la soutenance.

Conclusion

Le projet Smart Secure System a été un défi très enrichissant aussi bien au point de vue technique qu'au point de vue humain. Les problèmes techniques rencontrés ont pu être résolus grâce à la persévérance des équipes qui se sont beaucoup investies dans ce projet. C'était aussi une première pour nous de travailler dans une équipe de 6 personnes, car jusque-là nous étions au maximum 3. Ce projet a reflété les difficultés de l'entreprise mais nous avons essayés y faire face de manière efficace.

Le projet est déjà très abouti mais des améliorations sont encore possibles, la redondance électrique et la redondance de la connexion au réseau sont toujours en travaux. D'autres solutions techniques peuvent être envisagées, pour permettre de pérenniser le projet.

Depuis le commencement, le projet a été pensé pour être transmis à de futurs étudiants de réseaux et télécoms : le serveur a été migré vers un hébergeur leur permettant de continuer le travail commencé, l'ensemble des systèmes de capteur a été pensé avec des solutions open source, enfin le rapport de projet se veut le plus détaillé possible pour permettre une compréhension de la vision globale du projet. Nous espérons réellement qu'un autre groupe reprendra ce projet qui nous a tenus à cœur et dans lequel nous nous sommes beaucoup investis.

Smart Secure System, l'allié de votre sécurité !

Abstract

Smart Secure System **Automation** **Security**

1) For our project, Smart Secure System, our objectives are to create a fully working automation security system for small budgets. This include a lot of sensors like camera, infrared break beam, presence sensor, temperature sensor and many others. All the sensors are directly plug into an electronic card (in fact a mini computer) named Rapsberry Pi and then upload their data into our server. Finally, the user can remote access to all his data (camera flow, alerts from sensors…) thanks to our dedicated web server or our phone app on android.

2) Our group is form with 6 students, it's kind of huge group for a scholar project: in fact, it's the size of a fully and really working professional team. So, as you can see in the organizational chart, we decide to split the group between 3 duos: production and project management (and a bit of "I do nothing precisely, but I help everyone"), server web & app and camera and camera flow on RPI. We also insisted on the group communication by monthly meeting with all the crew, by mini meeting when the peoples need it and by frequent progress report for all the duos.

3) Our goal is to create a real product, maybe more a prototype, make to be sell, make to be useful and all over, yield all of our work to new R&T student because we truly think that can be an innovation. There is plenty of improvement axis, like more responsive apps, multi-flux server and system, secure all the communication between client and server, creating support for sensor…

Smart Secure System, the security for your family !