



PPD 1 : *Alarme distribuée à caméra embarquée*

Étudiants : Olivier VANEL, Sophie FERRAND, Gaïd
KANNENGIESSER, Benjamin DAUPHIN,
Romaric MARION, Kévin VU

Encadrants : Philippe DARCHE, Stefano PARIS

Mars 2014

SOMMAIRE

1. TABLE DES ILLUSTRATIONS	6
2. REMERCIEMENTS	7
3. INTRODUCTION (Gaïd & Olivier)	8
4. PRÉAMBULE	9
A. Les alarmes (Gaïd)	9
B. Les nano-ordinateurs (Olivier)	9
5. ENVIRONNEMENTS MATÉRIELS ET LOGICIELS	12
A. Équipements électroniques (Sophie)	12
B. Logiciels utilisés (Sophie)	13
6. ANALYSE	14
A. Le projet (Sophie d'après le sujet fourni par M. Darche)	14
B. L'application (Gaïd)	15
C. Cahier des charges (Gaïd)	16
D. Diagramme de cas d'utilisation (Sophie)	17
E. Diagramme de Gantt (Gaïd & Sophie)	18
8. RÉALISATION	19
A. Répartition des tâches (Sophie)	19
B. Bases de données (Kevin)	21
C. Gestion logicielle des entrées/sorties (Benjamin)	24
1. Motion et la détection de mouvement	24
2. Gestion des autres capteurs et déclenchement de l'alerte	25
3. Problème rencontré avec le serveur	26
D. Le site (Olivier)	27
E. L'application Android	31
1. Introduction à Android (Romaric)	31
2. Préparation Android (Kevin)	31
3. Activity (Kevin)	32
4. Connexion avec le serveur (Romaric)	33
5. Interface graphique (Kevin)	34
6. Limites (Kevin & Romaric)	36
9. CONCLUSION (Olivier & Gaïd)	37
10. GLOSSAIRE	38
11. ANNEXES	39
12. BIBLIOGRAPHIE	46

13. ENGLISH PART.....	47
A. Introduction (Gaïd).....	47
B. Member's work presentation.....	48
Benjamin DAUPHIN.....	48
Sophie FERRAND.....	48
Gaïd KANNENGIESSER.....	48
Romaric MARION.....	49
Olivier VANEL.....	49
Kevin VU.....	50
C. Conclusion (Gaïd).....	51

1. TABLE DES ILLUSTRATIONS

Figure 1 : Schéma d'un nano-ordinateur Raspberry Pi.....	10
Figure 2 : Microcontrôleur Arduino.....	11
Figure 3 : Schéma de l'application répartie	14
Figure 4 : DCU de l'application	17
Figure 5 : Diagramme de Gantt	18
Figure 6 : La base de données	21
Figure 7 : Table Alerte.....	21
Figure 8 : Table Appareil	22
Figure 9 : Table Utilisateur.....	23
Figure 10 : Table Zone	23
Figure 11 : Raspberry vue de haut.....	24
Figure 12 : Caméra de la Raspberry.....	24
Figure 13 : Commande de lancement du script d'envoi d'image dès que Motion détecte un mouvement.....	25
Figure 14 : Le montage réalisé par M. Darche.....	25
Figure 15 : Machine à état à l'alarme	26
Figure 16 : Page de connexion du site	27
Figure 17 : Page d'accueil du site	28
Figure 18 : Exemple de fonction codée grâce au framework CodeIgniter	30
Figure 19 : Schéma.....	34
Figure 20 : Page de connexion de l'application Android	35
Figure 21 : Sélection d'un appareil dans l'application Android	35
Figure 22 : Écran du vidéo streaming	35

2. REMERCIEMENTS

Nous tenons à remercier nos enseignants encadrants, MM. Philippe Darche et Stefano Paris, pour leur proposition de sujet et leur aide constante, mais aussi pour avoir mis à notre disposition tout le matériel dont nous avons besoin tout au long du projet.

Nous remercions également M. Ouziri pour nous avoir donné de son temps afin de nous aider au niveau du développement de l'application Android.

3. INTRODUCTION (Gaïd & Olivier)

Dans le cadre de notre formation en DUT informatique, nous avons eu le privilège de nous voir attribuer le projet pluridisciplinaire (PPD) concernant la conception et la réalisation d'une alarme distribuée pour tout type de lieu. L'élément principal du réseau de surveillance est un système embarqué muni d'une caméra communiquant via le réseau. Des Entrées-Sorties (E/S) logiques et analogiques lui ont été rajoutées pour lire des capteurs et contrôler des actionneurs. Le système de détection peut être contrôlé par une application Android ainsi qu'une application web. Ces applications seront connectées à un serveur lui-même relié à différents capteurs.

Afin de réaliser l'application pour mobile nécessaire à notre projet, nous avons dû apprendre à coder pour Android. En effet, cette plateforme nous était inconnue, ce qui nous a demandé une certaine période d'adaptation.

Lors de ce PPD, nous avons dû faire preuve d'une grande organisation pour pouvoir le mener à bien. C'est pourquoi nous avons réparti l'ensemble des tâches à effectuer lors de ce projet le plus rapidement possible, afin de ne pas perdre de temps.

4. PRÉAMBULE

A. Les alarmes (Gaïd)

Une alarme est un mécanisme permettant d'avertir d'un éventuel danger. Il existe divers types d'alarmes. Dans notre projet, l'alarme est capable de repérer une infraction : on parlera donc d'alarme anti-infraction. En effet, une alerte est déclenchée grâce à quelques dispositifs tels qu'un détecteur de mouvement, un détecteur infrarouge et un détecteur d'ouverture de porte. Le fait de mettre plusieurs moyens de détection d'intrusion pour notre alarme permet de rendre l'alerte plus précise et correcte : avoir moins de fausse alerte.

Dans notre projet, il s'agit d'une alarme distribuée. Autrement dit, il peut y avoir plusieurs alarmes et elles sont toutes indépendantes les unes des autres. De ce fait, il est possible d'ajouter ou enlever des alarmes sans que cela engendre des problèmes éventuels.

B. Les nano-ordinateurs (Olivier)

Les progrès en miniaturisation nous permettent, depuis peu, d'équiper des systèmes monocarte de composants développant une puissance considérable. Ainsi, il est possible de créer de véritables nano-ordinateurs pouvant accueillir un système d'exploitation. Par conséquent, il est possible d'utiliser les couches réseau ou de gérer des entrées/sorties.

La démocratisation des appareils connectés et systèmes embarqués découle directement de cette montée en puissance des nanosystèmes. Depuis peu, il est même possible d'acquérir dans le commerce des nano-ordinateurs monocartes fonctionnant généralement avec un processeur ARM.

Le Raspberry Pi (voir figure 1) faisant la taille d'une carte de crédit se présente comme tel. Il a été conçu par le créateur de jeux vidéo David Braben pour sa fondation caritative du même nom. Les premiers prototypes sont développés à partir de 2006, et la sortie officielle du Raspberry Pi date de février 2012.

La version A proposait une mémoire vive de 256 Mo tandis que la version B offre désormais 512 Mo. Cet ordinateur permet l'exécution de plusieurs variantes du système d'exploitation libre

GNU/Linux et des logiciels compatibles.

De base, le Raspberry Pi est fourni sans boîtier, alimentation, clavier, souris, ni écran. Ceci dans le but de minimiser les couts et de pouvoir recycler d'autres matériels. Mais sa puissance (ARM 700 MHz) et sa connectique diversifiée (2xUSB 2, HDMI, RCA, Ethernet, E/S GPIO...) en font un nano-ordinateur apte à supporter de nombreuses utilisations.

Offrant des fonctionnalités et une taille parfaite pour notre projet d'alarme distribuée, nous avons donc utilisé le Raspberry Pi.

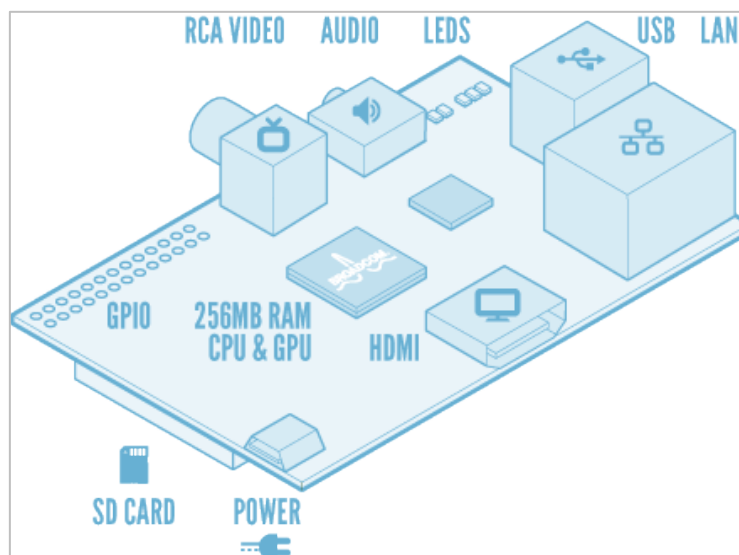


Figure 1 : Schéma d'un nano-ordinateur Raspberry Pi

Tout autant rattachées aux systèmes embarqués, mais pas dotés des mêmes fonctionnalités, certaines cartes permettent d'étendre les applications réalisables avec des nano-ordinateurs. C'est le cas pour l'Arduino (voir figure 2), que nous utilisons également dans notre projet.

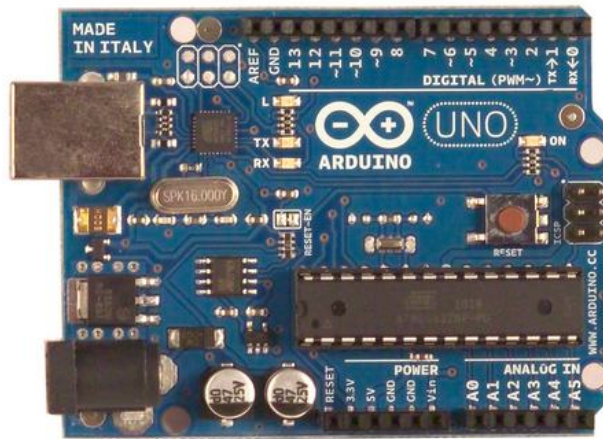


Figure 2 : Microcontrôleur Arduino

Arduino est un projet créé par une équipe de développeurs, composée de six individus. Cette équipe a créé le "système Arduino". C'est un circuit intégré contenant principalement un microcontrôleur qu'il est possible de programmer pour analyser et produire des signaux électriques afin de créer des systèmes électriques plus ou moins complexes.

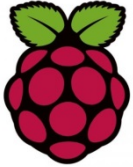
Le système Arduino permet de réaliser un grand nombre d'applications dans de nombreux domaines :

- Contrôler les appareils domestiques
- Fabriquer votre propre robot
- Faire un jeu de lumière
- Communiquer avec l'ordinateur
- Concevoir un système d'alarme
- Télécommander un appareil mobile (modélisme) ...

Son prix abordable, sa flexibilité, et la communauté gigantesque autour d'Arduino en font une carte agréable à utiliser et tout à fait adaptée à notre concept d'alarme distribuée.

5. ENVIRONNEMENTS MATÉRIELS ET LOGICIELS

A. Équipements électroniques (Sophie)



Nos professeurs encadrants ont mis à notre disposition deux nano-ordinateurs **Raspberry Pi**.



Nous avons aussi utilisé aussi un microcontrôleur **Arduino**, il a été monté avec un des nano-ordinateurs Raspberry Pi



Pour notre système de vidéo surveillance nous avons évidemment besoin d'une **caméra**, celle-ci a été montée sur le second Raspberry Pi



Afin de tester notre application Android, nos professeurs nous ont prêté une tablette **Samsung Galaxy 2**

B. Logiciels utilisés (Sophie)



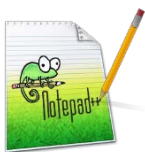
Éclipse avec le *plug-in* Android SDK afin de développer l'application Android



XAMPP un ensemble de logiciels qui nous a permis de mettre en place notre serveur Web sécurisé. Il contient Apache, MYSQL, phpMyAdmin et PHP.



Dreamweaver a été utilisé pour le style CSS de notre site pour sa fonction d'autocomplétion très utile



Notepad++ est un logiciel simple très utile pour le développement de plusieurs types d'application, nous l'avons beaucoup utilisé pour développer le site web de l'application



Photoshop a été utilisé pour le *design* de notre logo et d'autres éléments du site web et de l'application Android



Motion est le logiciel qui permet la détection de mouvement à l'aide d'une caméra, il est installé sur la seconde Raspberry Pi

6. ANALYSE

A. Le projet (Sophie d'après le sujet fournit par M. Darche)

Les systèmes embarqués sont en pleine expansion grâce à l'émergence des systèmes sur puce ou *SOC (System On Chip)*. Ces composants intégrés offrent aujourd'hui une puissance suffisante pour héberger un système Linux avec interface graphique X11 et réseau.

Ce projet consiste à concevoir et à réaliser une alarme distribuée (voir figure 3) pour tout type de lieu. Le nœud du réseau de surveillance est une carte électronique munie d'une caméra (à gauche dans la figure 1). Des entrées-sorties (E/S) logiques et analogiques lui seront rajoutées pour lire des capteurs et contrôler des actionneurs. Le système de détection pourra être contrôlé par une application sur une tablette (ou un smartphone) sous Android et via un site web.

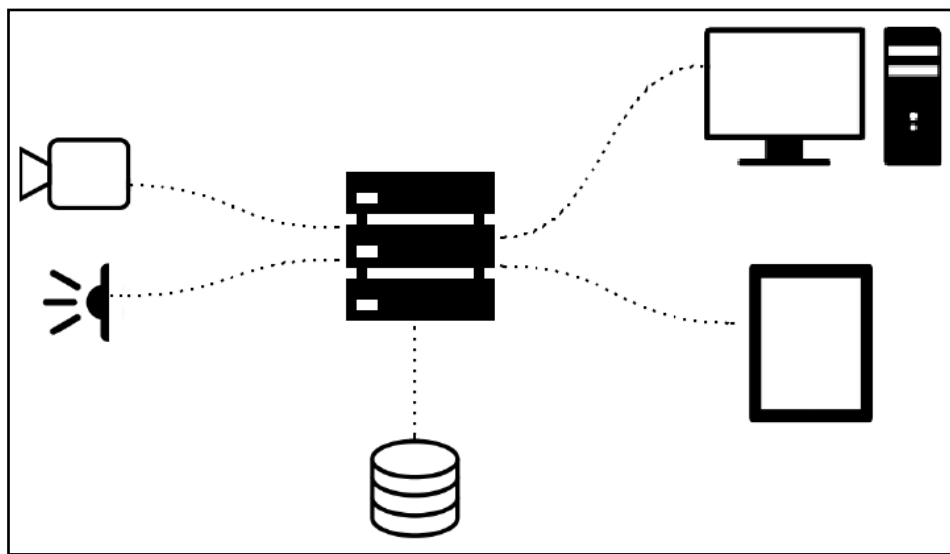


Figure 3 : Schéma de l'application répartie

La communication entre les différents sites (Raspberry Pi, serveur, site, application Android ...) est le point le plus important dans ce projet et celui qui sera le plus compliqué à gérer. L'application devra être extensible pour que l'administrateur puisse ajouter des capteurs voir des cartes sans que le système devienne inutilisable.

B. L'application (Gaïd)

Grâce à notre système de surveillance, un utilisateur peut accéder où qu'il soit au vidéo *streaming* et ainsi voir ce qui se passe à son domicile. Il pourra aussi être alerté dès qu'une intrusion est détectée soit par mail, soit par une notification Android (à condition qu'il ait installé notre application sur son terminal Android).

Pour pouvoir accéder à nos applications, web et Android, il faut que l'utilisateur ait un identifiant et un mot de passe. Ces paramètres d'authentification sont automatiquement associés à l'utilisateur du dispositif de surveillance. C'est pourquoi nous n'avons pas traité l'inscription d'un utilisateur au cours de ce projet. Une fois connecté, l'utilisateur peut accéder à un espace *streaming*. Par conséquent, il peut voir les images capturées par la caméra en temps réel. Il a également la possibilité de configurer certains paramètres comme l'envoi de l'alerte par mail, ou une notification sous Android. Une petite galerie de photos est également mise à disposition de l'utilisateur connecté. De plus, si l'utilisateur a besoin d'un renseignement, ou rencontre un problème quelconque avec notre application il pourra nous contacter.

C. Cahier des charges (Gaïd)

Nom de la fonction	Descriptif	Contraintes
Se connecter	Vérification de l'authentification par l'intermédiaire de mot de passe. On vérifie bien que l'identifiant et le mot de passe appartiennent bien à notre base de données.	Le nom d'utilisateur doit être composé d'au minimum 3 caractères et au maximum 30 et peut être composé de chiffres et de lettres. Le mot de passe doit comporter 5 à 30 caractères.
Se déconnecter	Permet à l'utilisateur de se déconnecter	Il faut être connecté pour se déconnecter.
Voir l'historique des alertes	Permet à l'utilisateur de consulter la liste des alertes.	Il faut être connecté.
Gestion de la caméra	Possibilité d'activer ou de désactiver la caméra à partir des applications.	Il faut être connecté et posséder une caméra associée à son compte.
Gestion des détecteurs de mouvements	Possibilité d'activer ou de désactiver les capteurs de mouvements.	Il faut être connecté et posséder un détecteur de mouvement associé à son compte.
Activer l'alerte par application Android	Une fois qu'un mouvement est détecté, un message est envoyé sur le smartphone de l'utilisateur, pour le prévenir de l'intrusion.	Il faut être connecté et posséder une tablette/smartphone Android sur lequel l'application est installée.
Activer l'alerte par adresse mail	Une fois qu'un mouvement est détecté, un mail est envoyé à l'utilisateur, pour le prévenir de l'intrusion.	Il faut être connecté.

D. Diagramme de cas d'utilisation (Sophie)

Avant d'écrire le cahier des charges, nous avons réalisé un Diagramme de Cas d'Utilisation (voir figure 4) afin de mieux appréhender toutes les fonctions que nous devrions développer.

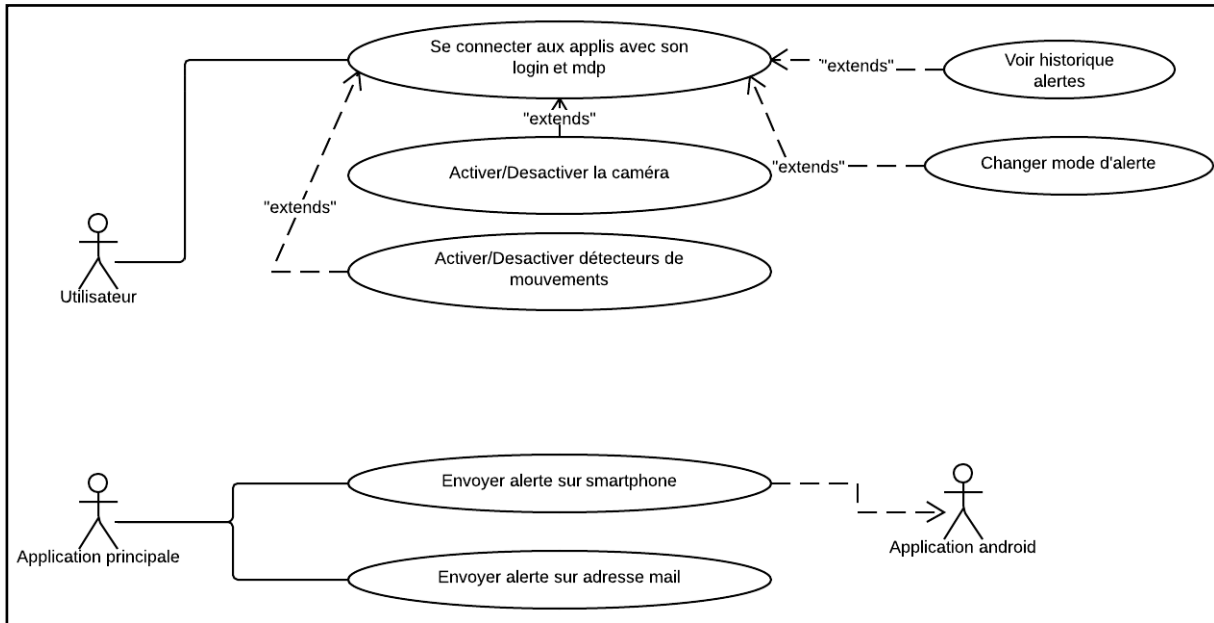


Figure 4 : DCU de l'application

E. Diagramme de Gantt (Gaïd & Sophie)

Pour démarrer ce projet, il nous a fallu un jour de réflexion, pendant lequel nous avons lu une partie de la documentation fournie par nos professeurs encadrants, au cours de notre première réunion. Nous avons également commencé ce même jour, à lister les différentes tâches sur lesquels nous avons dû travailler tout au long du projet (cette liste fût mise à jour régulièrement).

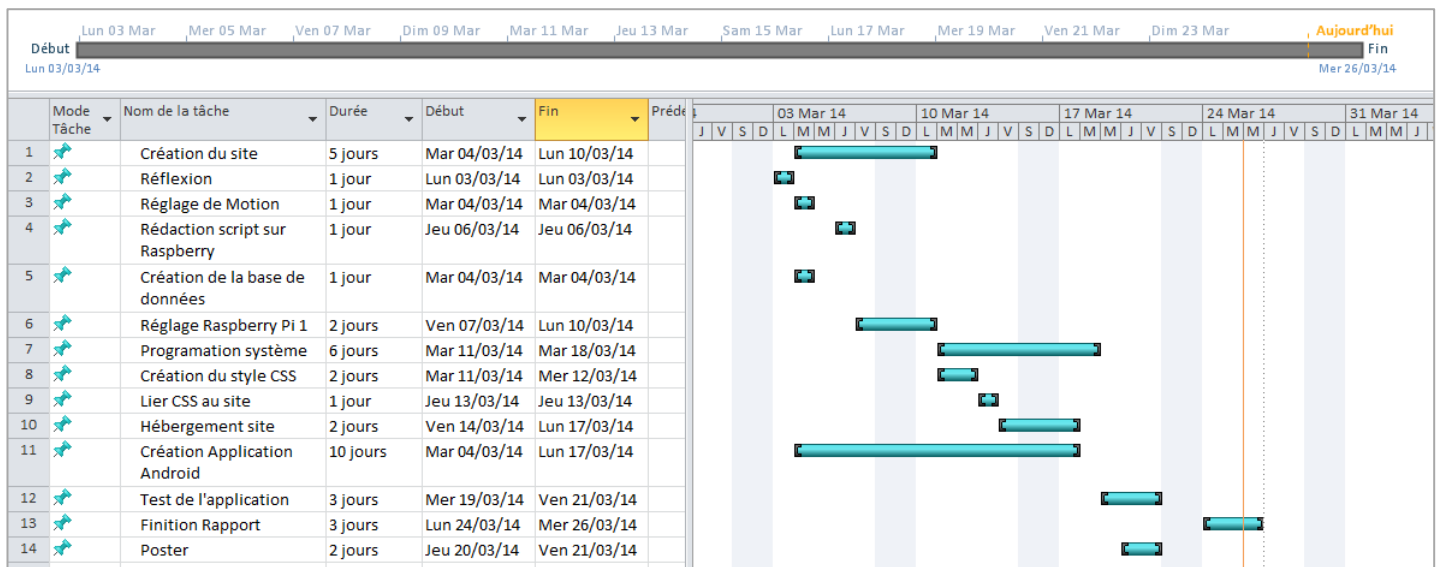


Figure 5 : Diagramme de Gantt

Nous avons vraiment pu démarrer le projet le deuxième jour. En effet, nous avons pu démarrer notre formation pour créer notre application sous Android, ainsi que débiter la création de l'application web (en s'informant dans un premier temps sur le *framework* que nous avons utilisé).

Ensuite, nous avons dû effectuer certains réglages sous Motion (notamment pour que la vidéo soit légèrement moins sensible). Nous avons également créé la base de données que nos applications (Android et web) devront utiliser. Nous avons rédigé des scripts pour Raspberry permettant par exemple d'envoyer une requête HTTP lorsqu'il y a un mouvement, ou encore un script permettant d'envoyer un signal à la carte à la détection d'un mouvement via Motion et il doit également recevoir un message d'un autre capteur pour pouvoir envoyer un signal à la carte.

Enfin, nous avons créé un style pour notre site séparément de l'ensemble du site et ses fonctionnalités dans un premier temps, puis nous l'avons lié au site. Et nous avons réalisé un ensemble de tests pour vérifier les fonctionnalités notre projet.

8. RÉALISATION

A. Répartition des tâches (Sophie)

Olivier Vanel : Chef de projet & Développeur web

Répartition des tâches

Réalisation du site web

Réalisation du poster

Gäid Kannengiesser : Intégrateur graphique & Administratrice de base de données

Création de la base de données

Mise en page (CSS/HTML)

Responsable du rapport de projet

Sophie Ferrand : Développeuse système et assistante-chef de projet

Analyste UML

Développement du script du logiciel Motion

Développement du programme d'E/S

Benjamin Dauphin : Développeur système

Développement du script du logiciel Motion

Développement du programme d'E/S

Développement du programme de liaison entre les différents sites

Kévin Vu : Développeur Android & Administrateur de base de données

Création de la base de données

Développement de l'application Android

Romarc Marion : Développeur Android

Développement de l'application Android

B. Bases de données (Kevin)

La base de données (voir figure 6) ici, comme dans la plupart des applications, à une grande place dans notre PPD. En effet, notre application « PPD : Protocole de Protection du Domicile » peut être utilisée sur un grand nombre d'appareils (caméra ou détecteur infrarouge) qui doivent être répertoriés et pour pouvoir être exploités.

Pour la base de données, nous avons choisi MySQL avec pour interface graphique phpMyAdmin.

Structure		SQL		Rechercher		Requête		Exporter		Importer		Opérations		Privilèg	
Table	Action										Lignes	Type	Interclassement		
<input type="checkbox"/> alerte												~0	InnoDB	latin1_swedish_ci	
<input type="checkbox"/> appareil												~0	InnoDB	latin1_swedish_ci	
<input type="checkbox"/> utilisateur												~1	InnoDB	latin1_swedish_ci	
<input type="checkbox"/> zone												~0	InnoDB	latin1_swedish_ci	
4 tables	Somme										1	InnoDB	latin1_swedish_ci		

Figure 6 : La base de données

Nous avons eu à recréer les tables sur MySQL après avoir appliqué les méthodes Merise.






Afficher	 Structure	 SQL	 Rechercher	 Insérer	 Exporter		
#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra
1	<u>NoAlerte</u>	int(11)			Non	Aucune	AUTO_INCREMENT
2	DateAlerte	date			Oui	NULL	
3	HeureAlerte	varchar(5) latin1_swedish_ci			Oui	NULL	
4	NoUtilisateur	int(11)			Oui	NULL	
5	NoApp	int(11)			Oui	NULL	
6	zone	int(5)			Non	Aucune	

Figure 7 : Table Alerte

La table Alerte (voir figure 7) comporte toutes les alertes envoyées par les caméras ou capteurs infrarouges. Une alerte est composée de :

NoAlerte: Clé primaire, elle permet de référencer les différentes alertes.

DateAlerte: Date de l'alerte, permet de savoir quel jour a eu lieu l'alerte.

HeureAlerte: Heure de l'alerte, avec la date, permet de savoir quand a eu lieu exactement l'alerte.

NoUtilisateur: Clé étrangère, permet de savoir à qui appartient l'appareil qui a provoqué l'alerte.

NoApp: Clé étrangère, permet de savoir quel appareil a provoqué l'alerte.

Zone: Clé étrangère, permet de savoir dans quelle zone a eu lieu l'alerte.

Afficher Structure SQL Rechercher Insérer Exporter							
#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra
1	NoApp	int(11)			Non	Aucune	AUTO_INCREMENT
2	Marque	varchar(30) latin1_swedish_ci			Non	Aucune	
3	Modele	varchar(30) latin1_swedish_ci			Non	Aucune	
4	Active	int(11)			Oui	NULL	
5	TypeApp	varchar(30) latin1_swedish_ci			Oui	NULL	
6	NoUtilisateur	int(11)			Oui	NULL	
7	zone	int(5)			Non	Aucune	

Figure 8 : Table Appareil

La table Appareil (voir figure 8) comporte tous les appareils pris en charge par l'application. Un appareil est composé de :

NoApp : Clé primaire, elle permet de référencer les différents appareils.

Marque : Marque de l'appareil.

Modèle : Modèle de l'appareil.

Active : Permet de savoir si un appareil est activé ou non.

TypeApp : « Caméra » ou « Capteur ». Permet de savoir à quel type appartient l'appareil.

NoUtilisateur : Clé étrangère, permet de savoir à qui appartient l'appareil.

Zone : Clé étrangère, permet de savoir quelle zone est prise en charge par l'appareil.






Afficher  Structure  SQL  Rechercher  Insérer  Exporter							
#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra
1	<u>NoUtilisateur</u>	int(11)			Non	Aucune	AUTO_INCREMENT
2	Login	varchar(30)	latin1_swedish_ci		Non	Aucune	
3	Mdp	varchar(60)	latin1_swedish_ci		Non	Aucune	
4	Email	varchar(50)	latin1_swedish_ci		Non	Aucune	
5	alertMail	int(11)			Oui	NULL	
6	alertAppli	int(11)			Oui	NULL	

Figure 9 : Table Utilisateur

La table Utilisateur (voir figure 9) comporte tous les utilisateurs de l'application. Elle est composée de :

NoUtilisateur : Clé primaire, elle permet de référencer les utilisateurs.

Login : Login de l'utilisateur.

Mdp : Mot de Passe

Email : Adresse e-mail de l'utilisateur, permet d'envoyer un mail à l'utilisateur lors d'une alerte si l'utilisateur le souhaite.

AlertMail : Permet de savoir si l'utilisateur souhaite qu'on lui envoie un mail lors d'une alerte.

AlertAppli : Permet de savoir si l'utilisateur souhaite qu'on lui envoie une notification sur son application Android lors d'une alerte.






Afficher  Structure  SQL  Rechercher  Insérer  Exp							
#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra
1	idZone	int(5)			Non	Aucune	AUTO_INCREMENT
2	libZone	varchar(30)	latin1_swedish_ci		Non	Aucune	

Figure 10 : Table Zone

La table Zone (voir figure 10) permet de coupler différents appareils afin de mieux cibler le mouvement. La table zone est composée de :

IdZone : Clé primaire, elle permet de référencer les zones.

LibZone : elle permet de savoir à quoi correspond exactement la zone.

C. Gestion logicielle des entrées/sorties (Benjamin)

Afin de mener à bien notre projet, nos professeurs encadrants nous ont prêté des ordinateurs de type Raspberry Pi (voir figure 11). Le Raspberry Pi est un nano-ordinateur monocarte qui permet l'exécution de légers programmes. Le système d'exploitation de ces ordinateurs est *Raspian* (un système d'exploitation de type Linux) c'est pourquoi nous avons principalement travaillé sur Linux.

Dans notre application les deux Raspberry Pi ont deux fonctions bien distinctes la première servant à détecter les mouvements à l'aide du logiciel Motion et la deuxième servant à gérer les différents capteurs et l'état de l'alarme.

1. Motion et la détection de mouvement

Le capteur photographique est le capteur clé de notre système d'alarme, en effet il devait permettre la récupération de photographies ou de vidéos et ainsi servir de preuve dans le cas d'un cambriolage.



Figure 11 : Raspberry vue de haut



Figure 12 : Caméra de la Raspberry

Le logiciel Motion remplit bien ces fonctions, il est capable de détecter des mouvements en comparant les images qu'il récupère de la caméra pixel par pixel. À chaque mouvement détecté, il enregistre les photographies et la vidéo de celui-ci.

Néanmoins, les paramètres d'origine du logiciel rendaient le système trop sensible (un changement de lumière était détecté comme un mouvement). Ainsi nous avons commencé à configurer Motion avant même de commencer à programmer sur la seconde Raspberry Pi. Ensuite afin que l'envoi des photographies s'effectue dès qu'un mouvement est détecté nous avons créé un

script python appelé par le logiciel :

```
630 # Command to be executed when a motion frame is detected (default: none)
631 ; on_motion_detected value
632 on_motion_detected python /home/pi/motion_rb/detect.py
```

Figure 13 : Commande de lancement du script d'envoi d'image dès que Motion détecte un mouvement

2. Gestion des autres capteurs et déclenchement de l'alerte

Pour cette partie nous nous sommes servis d'un montage réalisé par monsieur Darche d'un ordinateur Raspberry Pi couplé à une carte Gertduino

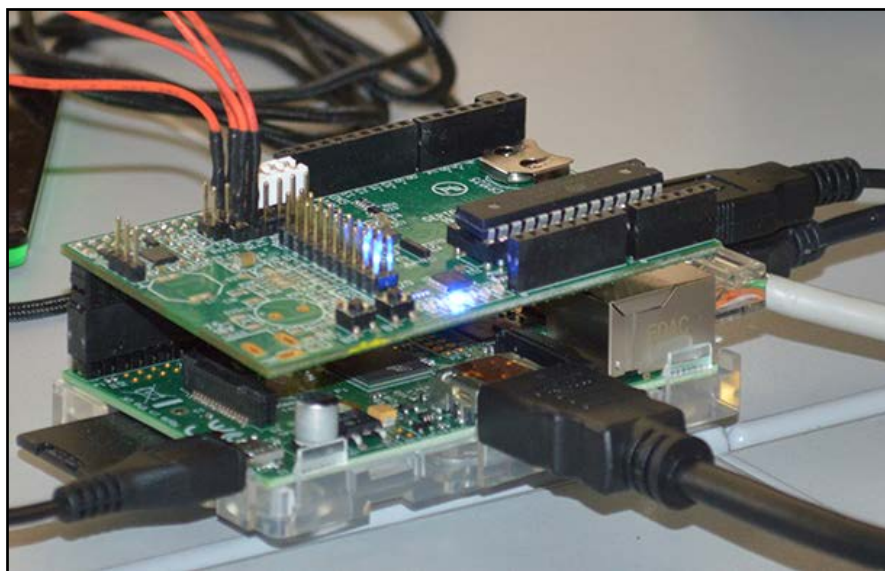


Figure 14 : Le montage réalisé par M. Darche

Les autres capteurs sont connectés à la carte d'entrée/sortie Gertduino (carte du haut sur la figure 14) et leur état (activé ou non) et récupérés par un programme C tournant sur cette carte. Ce programme écrit alors sur la liaison série avec la Raspberry Pi (carte du bas de la figure 14) l'état des différents capteurs afin que le nano-ordinateur effectue les traitements nécessaires. L'un des boutons de la Gertduino sert à simuler l'activation ou la désactivation par l'utilisateur de l'alarme, comme s'il avait entré le code de sécurité.

Sur la Raspberry Pi, un programme tourne en permanence afin de gérer l'état de l'alarme (désactivée, surveillance ...). Ce programme envoie une alerte au serveur si le capteur de la porte s'enclenche et que l'alarme n'est pas désactivée dans les trente secondes, ou immédiatement si la caméra et le capteur infrarouge s'activent. La réception des mouvements détectés par la caméra a

lieu grâce à un autre programme tournant en parallèle et dédié à la réception via le réseau des informations envoyées par le script Python (voir figure 15).

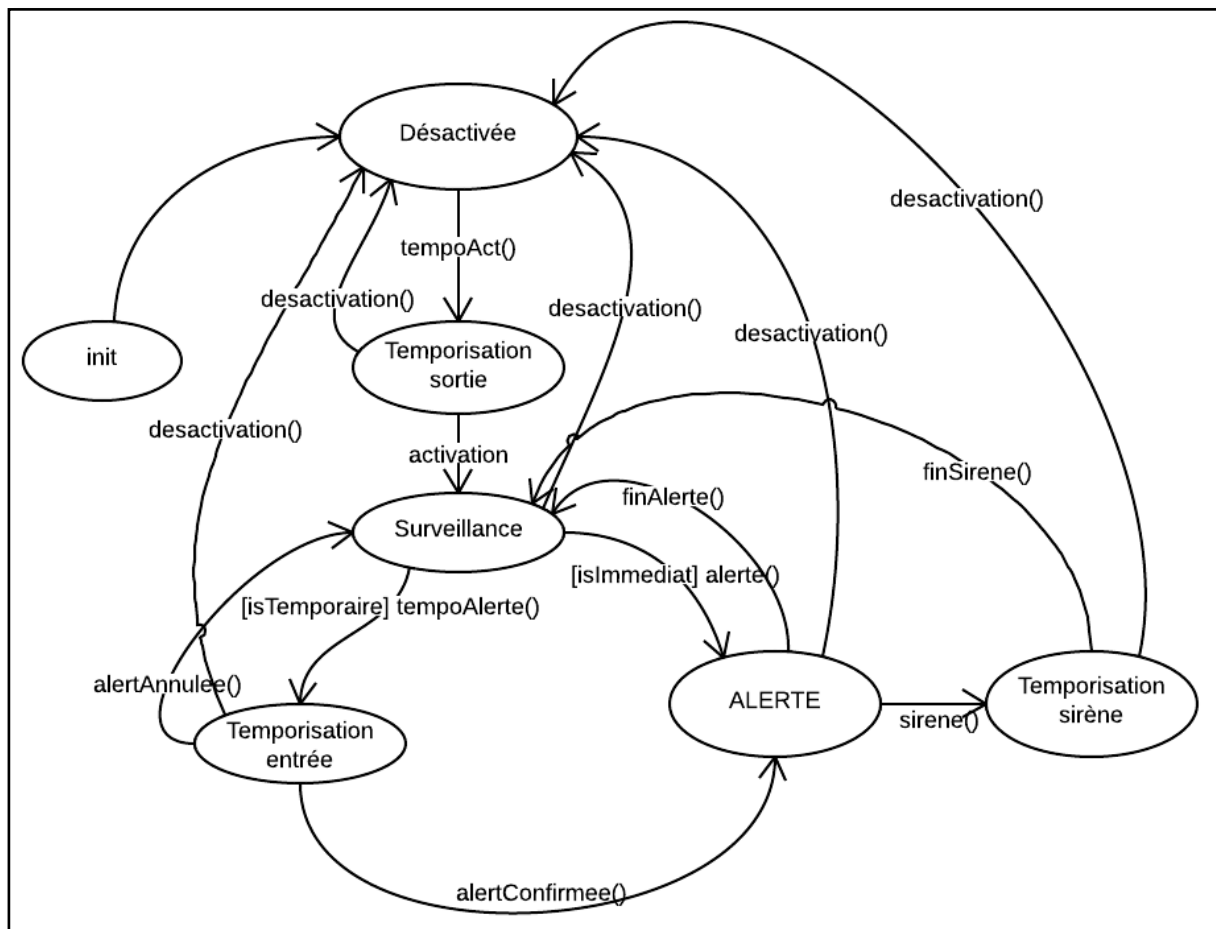


Figure 15 : Machine à état à l'alarme

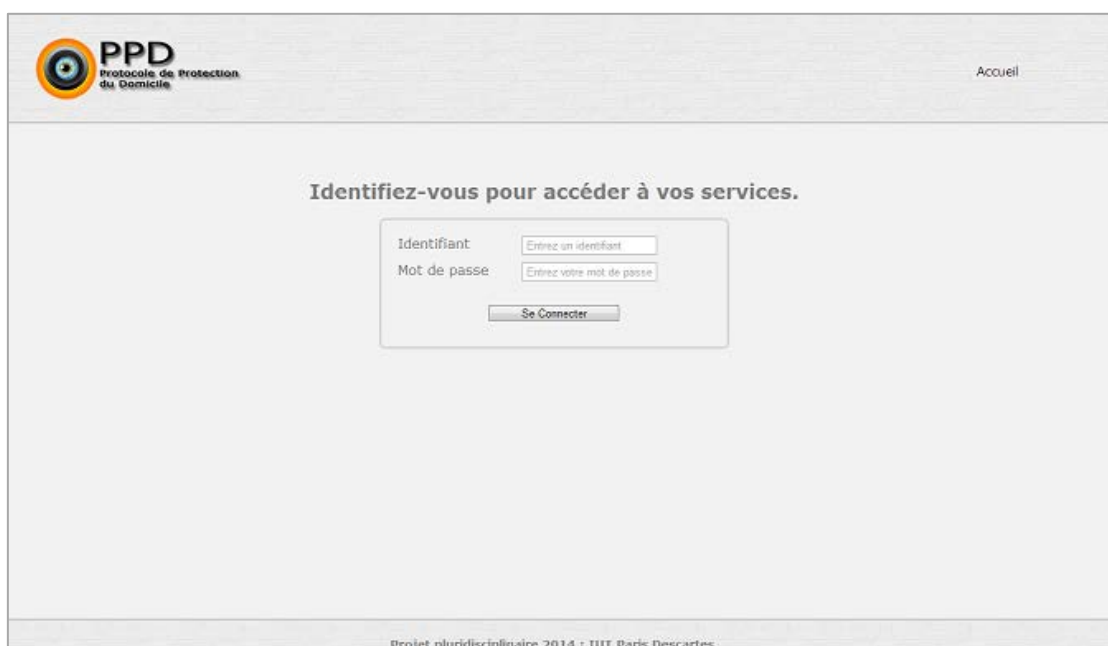
3. Problème rencontré avec le serveur

Initialement, nous comptons installer le serveur sur la carte Raspberry contenant le système d'alarme. En effet, ce genre d'ordinateur est tout à fait apte à héberger un site web et nous avons trouvé beaucoup de tutoriels nous indiquant la démarche. Bien que l'installation du serveur en lui-même fût réussie, nous ne sommes jamais parvenus à faire fonctionner le site sur ce serveur. Nous avons donc dû transférer le site sur un autre serveur AMP.

D. Le site (Olivier)

Notre but étant de réaliser une application permettant à un utilisateur d'accéder à son dispositif de sécurité quand il le désire nous ne pouvions proposer qu'une application pour Android. En effet, une personne ne possédant pas de tablette ou de smartphone (ou possédant un autre système d'exploitation tel que iOS ou Windows Phone) n'aurait pas pu utiliser notre application.

C'est pourquoi il nous a paru logique de développer en parallèle de notre application Android un site web (voir figures 16 et 17), rendant ainsi plus appropriée la consultation de son système d'alarme à un utilisateur qui se trouverait par exemple à son travail.



The image shows a web browser window displaying the login page of a website. In the top left corner, there is a logo for 'PPD' (Protocole de Protection du Domicile) featuring a stylized eye icon. In the top right corner, the word 'Accueil' is visible. The main heading in the center reads 'Identifiez-vous pour accéder à vos services.' Below this, there is a login form with two input fields: 'Identifiant' with the placeholder text 'Entrez un identifiant' and 'Mot de passe' with the placeholder text 'Entrez votre mot de passe'. A 'Se Connecter' button is positioned below these fields. At the bottom of the page, a footer line states 'Projet pluridisciplinaire 2014 : IUT Paris Descartes'.

Figure 16 : Page de connexion du site



Figure 17 : Page d'accueil du site

Notre étude de cas d'utilisation (voir Figure 2) nous a permis de définir les fonctionnalités principales et indispensables à notre système.

- Un utilisateur doit pouvoir se connecter de façon sécurisée à l'aide d'identifiants qui lui sont confiés lors de l'acquisition du matériel (voir figure 16).
- Une fois connecté, l'utilisateur a accès à son panneau de configuration. Il peut gérer les paramètres basiques de son installation. Ceci comprend la possibilité d'activer ou de désactiver la surveillance du domicile, ainsi qu'un historique des alertes.
- Un onglet « streaming » permet à l'utilisateur authentifié d'accéder à sa caméra afin d'observer son domicile en temps réel.
- Une page de déconnexion et un système de session expirant toutes les heures afin de renforcer la sécurité.

Dans un deuxième temps, des fonctionnalités supplémentaires ont été prévues :

- Création d'une page d'accueil explicative et ajout d'une galerie d'images relatives au projet.
- Création d'un onglet : « mode d'emploi ».
- Optimisation du site web pour d'autres navigateurs web (Chrome, Internet Explorer)

Afin de développer ce site, nous avons opté pour l'utilisation d'un *framework* PHP. Et ce, pour plusieurs raisons : Lors de nos recherches d'entreprises pour les stages ou l'alternance, nous avons remarqué beaucoup d'offres demandant des notions dans l'utilisation de *framework*.

Ainsi, n'ayant jamais été formés sur ce genre de technologies, nous avons donc trouvé opportun de profiter de ce projet pour en comprendre les grandes lignes. De nos jours, les *frameworks* sont utilisés par la plupart des sites web professionnels (Openclassroom, Dailymotion...). De plus, la structure de base d'un *framework* PHP nous donne la possibilité d'utiliser la programmation orientée objet. Cette manière de programmer ayant été une très grosse partie de notre enseignement à l'IUT pendant ces deux années, nous nous sommes trouvés avantagés. Notre productivité a doublé au niveau de la programmation web.

Il existe de nombreux *frameworks* PHP disposants de centaines de modules, tels que Symfony. Néanmoins, et afin de ne pas perdre trop de temps, nous avons choisi le *framework* CodeIgniter. En effet, il a été conçu pour n'offrir que le strict minimum en termes de modules, le reste étant optionnel. La légèreté de CodeIgniter fut un avantage sur deux points : la rapidité de formation sur cet outil et le temps de génération de nos pages web.

```

public function connexion()
{
    if($this->session->userdata('logged_in'))
    {
        $this->load->view('accueil/v_error_connect');
    }

    else
    {
        /*regles du formulaire*/
        $this->form_validation->set_rules('pseudo', '"Pseudo"',
        'trim|required|min_length[3]|max_length[30]|alpha_dash|encode_php_tags|xss_clean');
        $this->form_validation->set_rules('mdp', '"Mot de passe"',
        'trim|required|min_length[5]|max_length[30]|alpha_dash|encode_php_tags|xss_clean');

        if($this->form_validation->run()) //run renvoie TRUE si le formulaire est valide
        {
            /*On récupère les valeurs du formulaire*/
            $pseudo = $this->input->post('pseudo');
            $mdp = $this->input->post('mdp');

            /*On charge le model*/
            $this->utilisateur_model->connexion($pseudo,$mdp);

            if($this->utilisateur_model->isLoggedIn())
            {
                $this->index();
            }
        }
    }
}

```

Figure 18 : Exemple de fonction codée grâce au framework CodeIgniter

L'utilisation d'un *framework* PHP nous a permis de réaliser un site web structuré en MVC (Modèle, Vue, Contrôleur), de qualité professionnelle, stable, sécurisé et très facilement maintenable.

E. L'application Android

1. Introduction à Android (Romaric)

Notre programme de surveillance devant être accessible depuis n'importe où, notre intérêt s'est directement porté vers les smartphones et les tablettes tactiles. Aujourd'hui, il y a deux systèmes d'exploitation (OS) se démarquant très largement en popularité. Il s'agit d'iOS d'Apple avec 14,4 % et Android de Google avec 74,9 %.

Notre choix s'est rapidement porté sur le dernier. En plus d'être de loin le plus populaire, le kit de développement (SDK) du système d'exploitation le plus connu du monde permet le développement d'application en Java. Ce langage nous a été enseigné durant nos années d'études en DUT Informatique. Il nous sera donc plus simple de nous adapter au développement d'application sur appareil mobile.

L'autre avantage de ce système d'exploitation par rapport à son concurrent premier est son ouverture. En effet, il est très simple d'installer son application sur Android contrairement à iOS où il faut souscrire à une licence payante (79 euros en version standard individuelle).

Dans cette partie, nous allons donc vous présenter notre progression sur le développement de l'application Android. Pour cela, nous allons tout d'abord vous décrire notre apprentissage du SDK. Ensuite, nous allons vous décrire notre avancée dans les différentes parties du programme. En premier lieu, nous vous décrirons comment nous avons créé les différentes activités. Puis, nous vous parlerons de la communication entre l'application et la base de données commune avec le site web. Ensuite, nous vous dépeindrons la création de l'interface graphique. Finalement, nous aborderons les aspects que nous souhaitions au départ implémenter, mais que nous avons abandonnés, soit par faute de temps, soit à cause de sa complexité.

2. Préparation Android (Kevin)

Avant de commencer à programmer notre application Android, nous avons utilisé un tutoriel du site *OpenClassrooms* (anciennement Site du Zéro) afin de savoir de quels logiciels nous aurions besoin. Nous avons utilisé Eclipse comme IDE (environnement de programmation), car c'est celui qui nous avait été conseillé, mais aussi, car nous utilisons déjà Eclipse pour le développement en Java et C++. Eclipse en lui-même ne suffit pas à développer des applications pour Android. Il nous

a fallu utiliser aussi :

- Le JDK (Java Development Kit) qui contient le JRE (Java Runtime Environment) afin d'exécuter les applications Java, mais aussi un ensemble d'outils pour compiler et déboguer notre code (ce kit qui était déjà indispensable quand nous développions en Java)
- Le SDK d'Android (Kit de Développement) qui nous fournit un ensemble d'outils qui nous permettent de développer des applications pour Android.
- L'émulateur Android qui nous permet de tester notre application sans avoir à l'installer sur un smartphone ou une tablette.

Une fois le matériel en place, il nous a fallu ensuite nous préparer à programmer notre application. Pour cela, nous avons commencé par étudier le code du programme de départ (Hello World) généré automatiquement à la création d'un nouveau projet.

Ce code de départ utilisait les fragments, mais pour une raison de complexité nous avons fini par utiliser des activités simples.

Nous avons utilisé de nombreux tutoriels dont vous trouverez les liens des sites dans la bibliographie.

3. Activity (Kevin)

Notre application Android comprend de nombreuses *Activity* qui permettent essentiellement d'afficher nos différentes « Vues ». L'enchaînement de ces vues permet d'avoir une arborescence montrant l'enchaînement possible des différentes vues.

Voir annexe page 40 le diagramme des activity.

Dans notre application, chaque Activity est généralement composé :

- D'une méthode *onCreate(Bundle savedInstanceState)* qui est exécutée au lancement de l'activité et qui fait appel à une méthode *SetContentView* (permet de afficher une vue). Cette méthode permet aussi parfois de charger une *ListView* à partir du serveur.
- D'une méthode *SetOnClickListener* qui permet de lier un événement aux différents boutons de la vue. Ces boutons servent généralement à lancer une nouvelle *Activity*.

4. Connexion avec le serveur (Romaric)

Notre programme doit être capable de récupérer différentes informations étant stockées à distance. Il faut qu'à l'ouverture de l'application, l'utilisateur entre son identifiant ainsi que son mot de passe afin d'avoir accès aux différentes fonctionnalités. Ensuite, l'utilisateur doit être capable de regarder l'historique des notifications ainsi que l'accès aux différentes caméras qu'a souscrit l'utilisateur.

Nous avons pensé à différentes manières de connecter l'application Android à la base de données. La première est de passer par un script en PHP qui effectuera les requêtes SQL sur le serveur. La seconde méthode est presque identique à la précédente : il s'agit d'utiliser Java EE. La dernière est d'effectuer les requêtes SQL directement depuis l'application.

Méthode du PHP

Le site web étant hébergé sur un serveur Apache à l'aide du module PHP, nous avons en premier lieu pensé à envoyer les informations à traiter à un script PHP depuis l'application Android à l'aide de la requête POST utilisant le protocole HTTP.

Pour cela, nous avons utilisé la classe `HttpURLConnection` disponible dans la bibliothèque `java.net`.

Une fois la commande effectuée, le script PHP a pour fonction d'effectuer les requêtes SQL et de renvoyer le résultat à l'application Android.

Méthode du Java EE

Cette méthode est identique à la précédente. C'est celle que nous avons mise entre parenthèses au départ, car il faut installer un module supplémentaire sur le serveur pour exécuter les servlets.

Notre avons fait le choix de privilégier la présence de moins de modules nécessaire.

Méthode de la requête « SQL direct »

Finalement, la dernière méthode consiste à effectuer les requêtes SQL directement depuis l'application Android. Nous avons été réticents sur l'utilisation de ce procédé, car nous l'avons trouvé plus lente au niveau de l'exécution, car toute la base de données doit être chargée dans la mémoire de l'appareil mobile puis exécuter la requête contrairement aux deux requêtes précédentes.

Nous avons abandonné la méthode du PHP, car après avoir réussi à envoyer des données d'un programme en Java à un programme en PHP, nous n'avons pas réussi à faire l'inverse. Concernant la méthode du « SQL direct », les données de la base de données transitant par le réseau Internet, un manque de sécurité est flagrant. Cette méthode a donc été abandonnée. Finalement, nous avons opté pour cette méthode, car c'était, logiquement, la plus simple, utilisant les mêmes langages.



Figure 19 : Schéma

5. Interface graphique (Kevin)

Pour notre PPD, nous avons utilisé une interface simple permettant d'accéder aux fonctionnalités de l'application. Cette interface est essentiellement composée de *Widgets* tels que des : *ListView*, Boutons ou *TextView*.

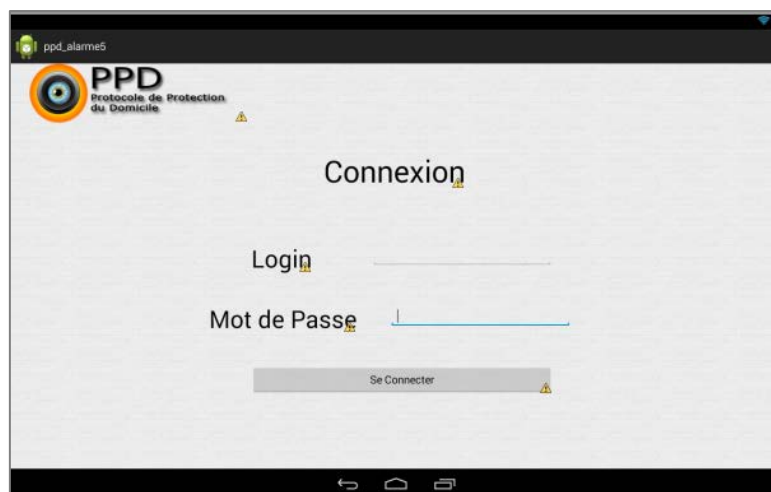


Figure 20 : Page de connexion de l'application Android

Pour pouvoir sélectionner un élément parmi une liste d'élément, nous avons décidé d'utiliser des RadioBoutons.

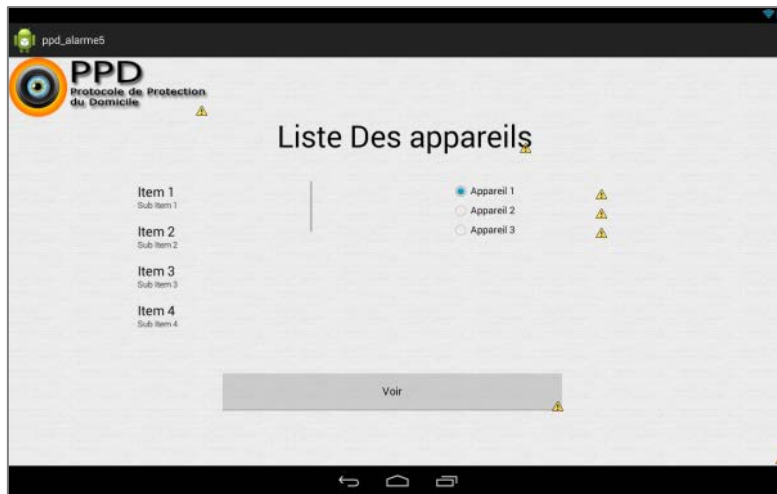


Figure 21 : Sélection d'un appareil dans l'application Android

Nous avons utilisé des *listViews* chargés à partir des données récupérées du serveur.

Il nous est aussi possible d'accéder au live de la caméra grâce au navigateur web.

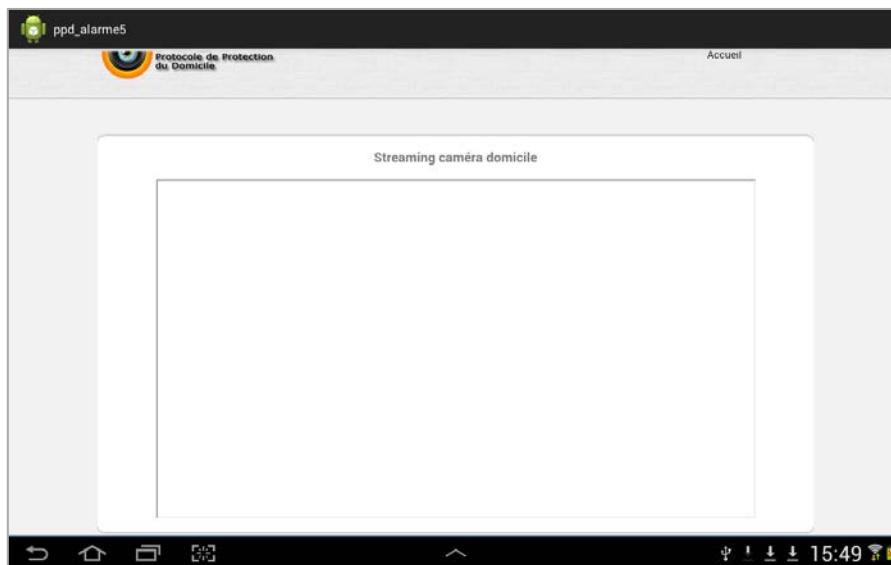


Figure 22 : Écran du vidéo streaming

Les *listViews* sont chargés dynamiquement à partir de la base de données.

6. Limites (Kevin & Romaric)

Nous avons rencontré certaines difficultés durant le développement de notre application Android. Nous avons eu 2 difficultés principales.

La première difficulté a concerné la communication entre notre application Android et notre serveur distant. Nous avons essayé de nombreuses méthodes comme l'utilisation de TomCat au lieu de phpMyAdmin. Nous avons aussi essayé d'utiliser un script PHP au niveau du serveur. Les conséquences de ce problème sont l'incapacité de notre application à récupérer toutes les informations contenues dans la base de données comme les alertes ou les appareils. Cependant, nous avons tout de même finalement réussi à rendre opérationnel le système d'identification après de nombreux efforts.

Le deuxième problème que nous avons rencontré concerne le streaming (vidéo en temps réel) de la caméra accessible à partir de l'application. Pour accéder au streaming de la caméra, la meilleure solution selon nous était de créer un lien qui permet à la tablette d'accéder au streaming de la caméra sur notre site avec le navigateur Internet de la tablette. Nous avons réussi à créer ce lien vers notre site cependant la vidéo du streaming ne pouvait être affichée sur la tablette. Il s'agirait selon nous d'un problème de format de la vidéo et de transfert de la vidéo (TCP), mais faute de temps, nous n'avons malheureusement pas pu régler le problème.

Pour le choix de l'appareil et des alertes, nous avons utilisé des *radiobuttons*. Nous voulions à l'origine ajouter dynamiquement les éléments dans une *listview*, mais à cause de sa complexité, on nous a vivement conseillé de choisir une méthode moins adaptée, mais plus simple que sont les *radiobuttons*.

9. CONCLUSION (Olivier & Gäid)

Finalement, d'un point de vue technique, nous avons réussi à programmer le Raspberry Pi relié à la caméra afin de lui permettre d'envoyer un message HTTP au second Raspberry couplé à l'Arduino.

Ce dernier permet de faire interagir plusieurs capteurs à l'aide d'un programme en C. Par la suite, une fois les conditions réunies, un message HTTP est envoyé vers le serveur apache afin d'alerter l'utilisateur d'une intrusion dans son domicile.

Pour ce qui concerne le site web, les fonctionnalités principales définies en début de projets ont été réalisées. Il est possible d'accéder à un espace.

Ce projet nous a permis d'améliorer nos connaissances dans divers domaines tels que la programmation web, la programmation en C, la création d'application pour Android, la gestion et la création de bases de données, et la modélisation de l'ensemble des fonctions que notre projet doit être capable de réaliser. Nous avons pu compter sur les compétences de chacun, ce qu'on appréhendait aux démarrages du projet. En effet, nous avons craint un déséquilibre dans la répartition des tâches, mais nous avons tous su nous impliquer de la même façon dans ce projet. Ainsi chacun a pu faire ce qu'il appréciait.

Les quelques difficultés rencontrées ont réussi à être surmontées. La programmation sous Android a été l'une de nos plus grandes difficultés, puisqu'aucun des membres du groupe n'avait codé pour Android dans le passé. Débuter de zéro sur une plateforme qui nous était inconnue fut un challenge que nous sommes parvenus à surmonter avec plaisir.

Ce travail de groupe fut constructif, mais aussi, un très bon moyen d'apprendre à être autonome et organiser notre temps de travail. Ainsi se fut une très bonne préparation pour nos futurs stages.

Le sujet de notre PPD étant très audacieux, nous sommes loin de l'avoir pleinement exploité, cependant nous nous sommes efforcés de rendre notre application la plus extensible possible. Ainsi nous espérons qu'un futur groupe de PPD pourra s'appuyer sur notre travail afin de rendre cette application encore plus fonctionnelle et utile.

10. GLOSSAIRE

Activity (Android) : classe Java exécutée lors de l’affichage d’une vue

Framework : ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d’une partie d’un logiciel (architecture)

ListViews : fournis l’infrastructure pour afficher un groupe d’éléments de données dans différentes dispositions ou différents affichages. Par exemple, il se peut qu’un utilisateur souhaite afficher des éléments de données dans une table et trier ses colonnes.

RadioButton : élément visuel des interfaces graphiques (un widget). Les boutons radio sont toujours utilisés en groupe puisque leur objectif est de permettre à l’utilisateur de choisir une seule option parmi plusieurs possibles

TextViews : Classe Java qui permet d’afficher du texte à l’utilisateur.

11. ANNEXES

Diagramme des Activity de l'application Android	40
Code de gestion de l'alarme, partie 1	41
Code de gestion de l'alarme, partie 2	42
Code de gestion de l'alarme, partie 3	43
Code de gestion de l'alarme, partie 4	44
Code de gestion de l'alarme, partie 5	45

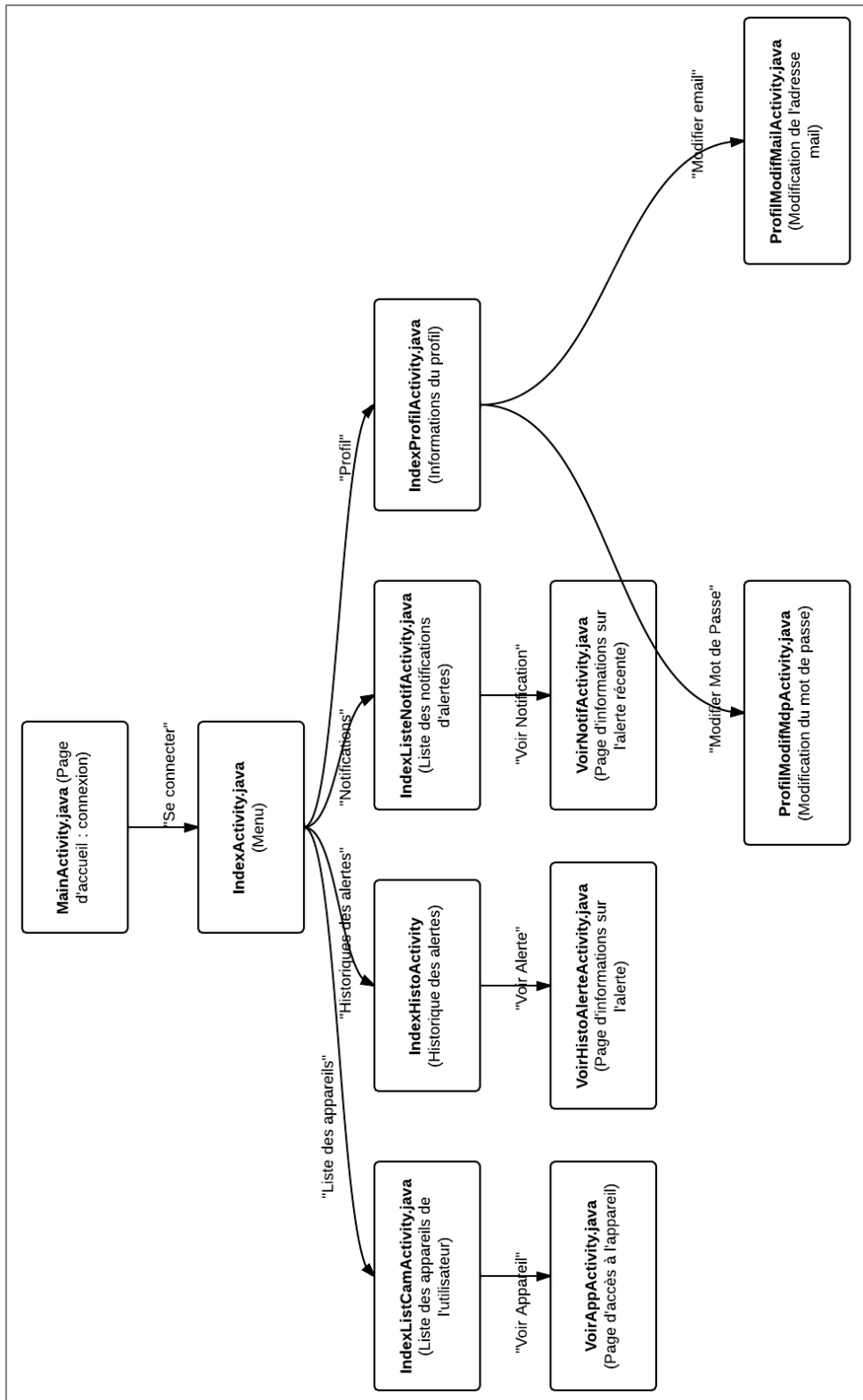


Diagramme des Activity de l'application Android

```

enum EtatAlarme {
    DESACTIVE,
    SURVEILLANCE,
    ALERTE,
    TEMP_SORTIE,
    TEMP_ENTREE,
    TEMP_SIRENE
};

struct Zone {
    int maskCapt[NB_CAPT + NB_CAM];
};

int zoneDetect(char capteurs[], struct Zone zone);

int main() {
    enum EtatAlarme etat = DESACTIVE;
    char capteurs[NB_CAPT + NB_CAM], etatChar;
    int i, nbblc = 0, serial = serialOpen("/dev/ttyAMA0", 9600);
    time_t tpsPrev, tpsActu;

    int tubeCam[2], tubeCapt[2]; //, nbCharLus;
    char chtube[10], chserial[10], msg[25], lgMsg;
    struct pollfd pollTubes[2];

    pipe(tubeCapt);
    pipe(tubeCam);

    if (fork() == 0) {
        close(tubeCam[0]); // fermeture tubeCam en lecture dans le fils
        sprintf(chtube, "%d", tubeCam[1]); // int -> char
        execl("./recepCam", "recepCam", chtube, NULL);
    }
    if (fork() == 0) {
        close(tubeCapt[0]);
        sprintf(chtube, "%d", tubeCapt[1]);
        sprintf(chserial, "%d", serial);
        execl("./capteurs", "capteurs", chtube, chserial, NULL);
    }

    close(tubeCam[1]);

```

Code de gestion de l'alarme, partie 1

```

close(tubeCapt[1]);

struct Zone zoneBouton = {{1, 0, 0, 0}},
zonePorte = {{0, 1, 0, 0}},
zoneInterne = {{0, 0, 1, 1}};

while (1) {
    if (!(nbblc++ % 40))
        fprintf(stderr, "boucle\n");

    pollTubes[0].fd = tubeCapt[0];
    pollTubes[0].events = POLLIN;
    pollTubes[1].fd = tubeCam[0];
    pollTubes[1].events = POLLIN;

    poll(pollTubes, 2, -1);

    // Message sur tubeCapt
    if (pollTubes[0].revents & POLLIN) {
        read(tubeCapt[0], msg, NB_CAPT);
        for (i = 0; i < NB_CAPT; ++i) {
            capteurs[i] = msg[i];
        }
    }
    // Message sur tubeCam
    if (pollTubes[1].revents & POLLIN) {
        read(tubeCam[0], &lgMsg, 1);
        read(tubeCam[0], msg, lgMsg);
        sscanf(msg, "%d %c", &i, &etatChar);
        capteurs[i] = etatChar;
        fprintf(stderr, "camera capteurs[%d] = '%c'\n", i, etatChar);
    }

    switch (etat) {
    case DESACTIVE:
        if (zoneDetect(capteurs, zoneBouton) > 0) {
            etat = TEMP_SORTIE;
            printf("Sortie\n");
            tpsPrev = 0;
        }
    }
}

```

Code de gestion de l'alarme, partie 2

```

        break;

    case SURVEILLANCE:
        if (zoneDetect(capteurs, zoneBouton) > 0) {
            etat = DESACTIVE;
            printf("Desactive\n");
        }
        // Temporisation d'entree si la porte est ouverte
        else if (zoneDetect(capteurs, zonePorte) > 0) {
            etat = TEMP_ENTREE;
            printf("Entree\n");
            tpsPrev = 0;
        }
        // Alerte si au moins 2 capteurs internes sont enclenches
        else if (zoneDetect(capteurs, zoneInterne) >= 2) {
            etat = ALERTE;
            printf("Alerte\n");
        }
        break;

    case ALERTE:
        if (zoneDetect(capteurs, zoneBouton) > 0) {
            etat = DESACTIVE;
            printf("Desactive\n");
        }
        // TODO Envoyer alerte au site
        etat = TEMP_SIRENE;
        printf("Sirene\n");
        tpsPrev = 0;
        break;

    case TEMP_SORTIE:
        if (zoneDetect(capteurs, zoneBouton) > 0) {
            etat = DESACTIVE;
            printf("Desactive\n");
        }
        else if (tpsPrev == 0) {
            tpsPrev = time(NULL);
            tpsActu = time(NULL);
        }
        else if (tpsActu - tpsPrev < DELAI_SORTIE) {

```

Code de gestion de l'alarme, partie 3

```

        tpsActu = time(NULL);
    }
    else {
        etat = SURVEILLANCE;
        printf("Surveillance\n");
    }
    break;

case TEMP_ENTREE:
    if (tpsPrev == 0) {
        tpsPrev = time(NULL);
        tpsActu = time(NULL);
    }
    else if (tpsActu - tpsPrev < DELAI_ENTREE) {
        if (zoneDetect(capteurs, zoneBouton) > 0) {
            etat = DESACTIVE;
            printf("Desactive\n");
        } else {
            tpsActu = time(NULL);
        }
    }
    else {
        etat = ALERTE;
        printf("Alerte\n");
    }
    break;

case TEMP_SIRENE:
    if (zoneDetect(capteurs, zoneBouton) > 0) {
        etat = DESACTIVE;
        printf("Desactive\n");
    }
    else if (tpsPrev == 0) {
        tpsPrev = time(NULL);
        tpsActu = time(NULL);
    }
    else if (tpsActu - tpsPrev < DELAI_SIRENE) {
        tpsActu = time(NULL);
    }
    else {
        etat = SURVEILLANCE;
    }

```

Code de gestion de l'alarme, partie 4

```

        etat = SURVEILLANCE;
        printf("Surveillance\n");
    }
    break;

default:
    break;
}

serialFlush(serial);
if (etat == DESACTIVE) {
    serialPutchar(serial, 'D');
} else if (etat == SURVEILLANCE) {
    serialPutchar(serial, 'S');
} else if (etat == TEMP_SIRENE) {
    serialPutchar(serial, '^');
} else if (etat == TEMP_SORTIE) {
    serialPutchar(serial, 'O');
} else if (etat == TEMP_ENTREE) {
    serialPutchar(serial, 'E');
} else {
    serialPutchar(serial, 'A');
}
}

close(tubeCam[0]);
close(tubeCapt[0]);
serialClose(serial);

return 0;
}

int zoneDetect(char capteurs[], struct Zone zone) {
    int nbDetection = 0, i;

    for (i = 0; i < NB_CAPT + NB_CAM; ++i) {
        nbDetection += ((capteurs[i] == CAPT_DETECT) && zone.maskCapt[i]);
    }

    return nbDetection;
}

```

Code de gestion de l'alarme, partie 5

12. BIBLIOGRAPHIE

Sites Web

<http://www.lafermeduweb.net/>

<http://fr.openclassrooms.com/>

http://codeigniter.fr/user_guide/index.html

<https://developer.android.com/index.html>

<http://www.developpez.com/>

<http://theopentutorials.com/>

<http://www.tutorialspoint.com/>

<http://blog.idleman.fr/raspberry-pi-13-detecter-louverture-des-portes-et-fenetres/>

<http://www.framboise314.fr/une-camera-de-surveillance-video-avec-le-raspberry-pi/>

<https://medium.com/p/2d5a2d61da3d>

Livres

30 Arduino Projects for Evil Genius, Simon Monk

13. ENGLISH PART

A. Introduction (Gaïd)

Protocole de Protection du Domicile is an application developed for CCTV (closed-circuit television); Thanks to our program a user can see live what is happening in his home by connecting to the application's website or by the android application. This subject was proposed by our professor Philippe DARCHE and Stefano PARIS; we thought it would be an interesting challenge.

To develop this application we worked on three axes: web development, Android development and *sensor integration*. If the web development wasn't very difficult, it wasn't the case for other axes. In fact, we had to learn how to program for Android; It was hard, but it allowed us to acquire a solid basic for this new technology. For the *Sensor Integration* part we had difficulty in that we never worked on that kind of hardware before (Raspberry Pi and Arduino) so the beginning was hard. But when we understood how it worked we were more effective.

Our PPD consisted of six members, Olivier VANEL (project manager and web developer), Sophie FERRAND (project manager assistant and system developer), Romaric MARION (Android developer), Kévin VU (Android developer), Benjamin DAUPHIN (system developer), and then Gaïd KANNENGIESSER (graphics integrator & database administrator). We were supervised by Philippe DARCHE and Stefano PARIS.

B. Member's work presentation

Benjamin DAUPHIN

I was in charge of the server so the first few days I thought about the architecture of our application. Since this is a distributed application its structure is very important (more than in a centralized application). We decided that one of the Raspberry Pi, provided by our supervisors Mr. PARIS and Mr. DARCHE, would be used as a server.

I wrote the script of the second Raspberry Pi (with the camera); this script is triggered by the software Motion when it detects motion. I worked on this part with Sophie because she knew how Motion worked.

After that, we installed the server and started to write different script.

Sophie FERRAND

Since I have already worked on electronic devices when I was at High school, we decided that I would be in charge of the hardware (Raspberry Pi). My first task was to change the settings of the Motion; (This is the application that detects motions.) I had to change these settings because in the beginning, the sensor was too sensitive, in fact it detected a simple light variation and that would cause problems (false alarms).

Next, I worked with Benjamin on the configuration of the second Raspberry Pi, this one is used to manage the different sensors (infrared sensor, motion sensor, proximity sensor). This was the first time we had worked on that kind of device and program, so we had difficulty here. The Raspberry Pi is also used as a server, so we had to configure it and install the website created by Olivier.

Gaïd KANNENGIESSER

Throughout this project, I learned a lot of things. In fact, I have improved my knowledge in web programming and learned about programming for Android. Then, I also worked on database management and creation with Kevin, it was a really captivating task. I worked specifically on the views design which means I used HTML and CSS. I offered my help to work on web design

because I really liked the website's aesthetics.

One of the things I liked the most in this project was to be independent, because we had to find every step of our project by ourselves and plan every task not to waste time.

Moreover knowledge acquired and deepened as, this was nice team work. Thanks to this project we all learned how to work in a team, as well as how to share tasks.

Romarc MARION

I have been interested in software engineering for touch-screen devices since its democratization. With the approval of every member of the group, I chose to work on a mobile application. As I didn't have the opportunity to learn about it during my studies, I thought it was an opportunity to finally know more about the mobile environment.

Obviously, the difficulty was to use tools I had never used before. The first step was to read the documentation and tutorials found on the Internet. Then, we had enough knowledge to start programming for the operating system imposed: Android. Kevin and I worked on it.

Moreover, I helped Benjamin with the installation of the server (Apache) on Linux because I already worked on it for personal projects.

Olivier VANEL

During one of my courses, Mr. Stefano Paris was talking about a project which mixes hardware and software. I was intrigued by this subject, so I proposed it to my friends, and they liked it!

Web programming is my favorite part of computer sciences. As our project had to have a website, I immediately proposed to develop it myself. I didn't want to develop this website like we did in qeb Programming courses. So I learned to use CodeIgniter, which is a basic framework, recommended by one of my teammates: Romarc.

Frameworks are used by some companies to develop their websites because it's much better to maintain. To create our site, I used conventional technologies like HTML, PHP or JavaScript...

Moreover, during my search to find an internship, I noticed that knowledge of Frameworks

is required.

This project was a good opportunity to improve my skills and learn new ways of working. During our work time, I looked at the other positions to remain informed of the progress of our project and to help my teammates if they needed. I was very happy to work on this subject with my friends, it was an opportunity to develop our skills faster in good conditions.

Kevin VU

I liked working on databases so I immediately my services to create the database for our application. I worked on this task with Gaïd, and before encoding this database we made a modeling of it with the Merise methodology (MLD).

The database was finished in two days, so I decided to begin a whole new task with Romaric: the creation of an application for Android devices. Our teachers lent us a Samsung Galaxy tablet so we were very motivated to programming this application. The beginning was really hard, as we only trained by reading documentation and following tutorials for two days. During the DUT we never had the opportunity of working on a similar technology, the first challenge was to manage the display.

Although work on this task was difficult, I'm glad I learned new computer skills, especially since I'm sure they will help me in my career.

C. Conclusion (Gaïd)

In conclusion, we had little difficulty knowing where to start, considering we had to work on things we had never learned in course (for example Android's programming) so it was a hindrance to the smooth running of our project.

This project allowed us to improve our skills in different matters, on a new platform with different languages. We also learned to work as a team, and organize our work time.

Finally, it was also very interesting project because it allowed us to widen our knowledge in Android application, web application, and system programming. We all enjoyed working together.