DOSSIER DE PROJET :

Mariton Sécurité

B.T.S SN 2016

Étudiants :

SAIDJ Sofiane - PASERO Bastien
ROBIN Cédric - PUG Alexandre

Lycée Alphonse Benoît - Isle sur la Sorgue
Session 2016-2017

Tables des matières

Partie Commune

1- Présentation du système……..p.4

2- Description des éléments…….……p.5

3- Répartition des tâches……..…p.7
a- Spécification du rôle de chacun……...…..p.7

4- Diagramme de GANTT prévisionnel……...…p.8

Partie Personnelle

1- Partie 1 : Électronique et communication
1.1- EC1 : PUG Alexandre……..….p.11

2- Partie 2 : Informatique et réseau
2.1- IR1 : PASERO Bastien…….p.29
2.2- IR2 : ROBIN Cédric……..….p.73
2.3- IR3 : SAIDJ Sofiane……..….p.93

Annexes……...……....p.117

Partie Commune

1- Présentation du système :

Le projet MARI-SEC (Mariton sécurité) est un système qui à terme vise la mise en place d'une sécurité partiellement autonome dans une maison. Pour qu'un système assure la sécurisation et la surveillance en autonomie partielle, il a besoins d'être composé d'un ensemble de systèmes.
Ce projet est donc un ensemble de systèmes qui vont être mis en relation afin d'assurer les points suivant :

- Une surveillance extérieure de la maison afin de sécuriser la section non couverte de la propriété privé.
- Une visibilité sur l'extérieur de la maison à toute heure et accessible depuis n'importe quel appareil connecté au web.
- La mise en route d'un protocole afin d'alerter le propriétaire en cas d'intrusion.
- La sauvegarde des informations utiles lors du déclenchement d'une alerte.
- L'accès partiel au système de la maison depuis Internet.
- L'accès total sur l'ensemble du système depuis l'intérieur de la maison.
- La sécurisation de l'intégrité de la maison, afin de prévenir lors d'intrusion dans cette dernière
- Et enfin l'intégration de tout ces système en un seul afin de créer un seul système qui pourra assurer tout ces points en même temps.

Donc voici comment va fonctionner ce système en situation réelle. Tout d'abord une caméra IP sera placé à l'extérieur de la maison afin de surveiller la partie extérieure de la propriété, cette camera fonctionneras en corrélation avec un système de détection de mouvement qui sera lié à Internet et alertera son propriétaire en cas de détection éventuelle. Lorsqu'une détection est effectuée le propriétaire reçoit une alerte, la caméra commence à enregistrer, dans une base de données, ce qu'il se passe, et l'éclairage extérieur se met en marche. Dans cette situation le propriétaire peut accéder via le site web sécurisé de sa maison, aux images filmées en direct par la caméra de sa maison, il peut alors stopper l'alerte depuis n'importe où.

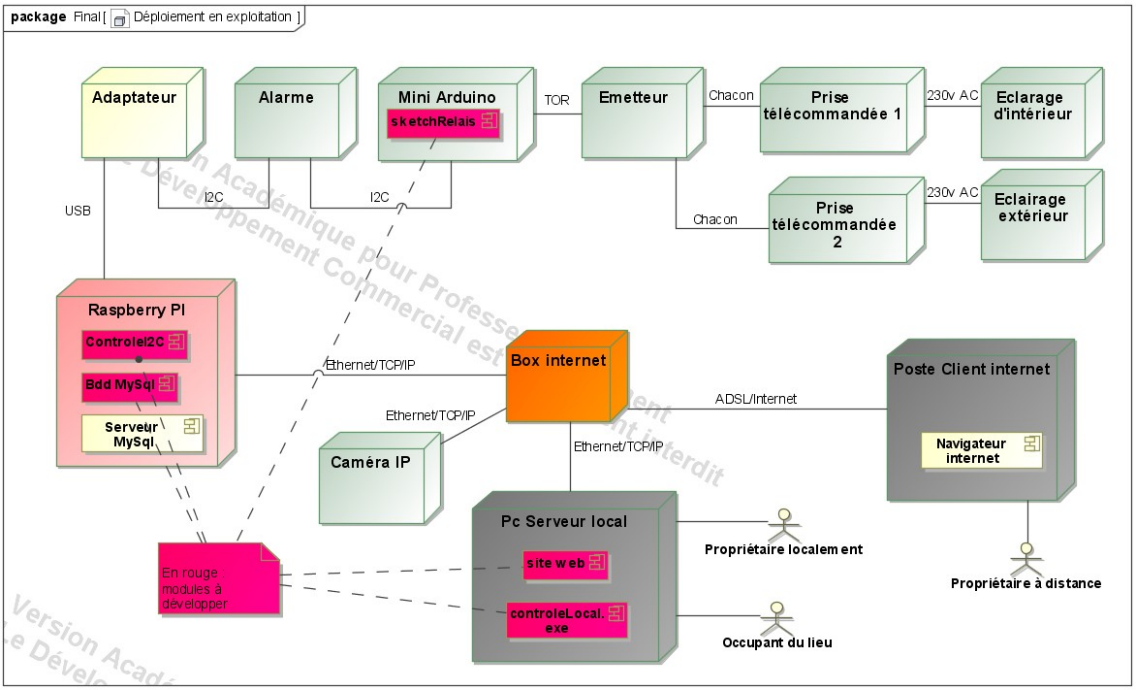
Voici maintenant une autre situation, dans le cas où la camera n'a pas détecter de présence, la maison possède des capteurs magnétique aux portes et fenêtres qui sont reliés à une centrale d'alarme. Lorsqu'un capteur est déclenché l'alarme ainsi que tout l'éclairage de la maison se met en marche, excepté si le capteur en question est le capteur de la porte d'entrée principale, dans ce cas il y a un certains délai pendant lequel l'alarme reste en attente, ce temps permet pour les propriétaire d'entrer le code de désactivation de l'alarme, ce délai passer l'alarme ainsi que l'éclairage se mettent en route et le propriétaire est la aussi prévenu. Pour des raison de vie privé ce système ne comporte pas de caméra à l'intérieur de la maison.

Nous sommes donc quatre étudiants pour la mise en place et l'intégration de ces systèmes, 3 IR (étudiants plutôt axé programmation) et 1 EC (étudiant plutôt axé électronique), dont l'ensemble des travaux devrait permettre la réalisation du projet dans les délais fournit.

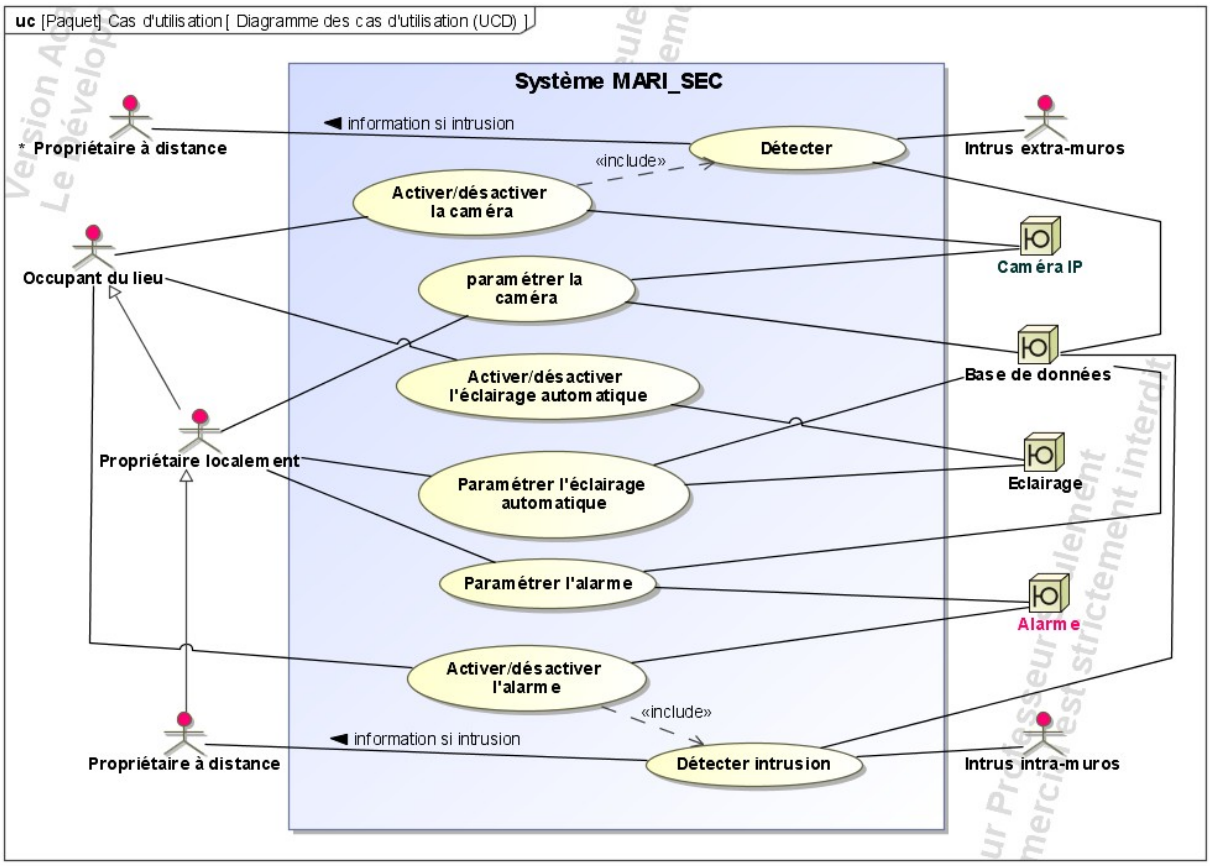
2- Description des éléments :

Donc le système qui va être mit en place sera composé des éléments suivants :
- Une camera IP, qui sera connecté au web et qui permettra la détection d'intrus à l'extérieur de la maison.
- Un serveur qui sera situé dans l'enceinte de la maison et qui assurera l'hébergement du site web ainsi que des logiciels de contrôle du système.
- Une Box Internet qui assurera la connexion à Internet ainsi que la communication des éléments entre eux
- Une Raspberry PI 2 qui aura pour but d'assurer la gestion centrale des données, elle sera placé dans l'enceinte de la maison et en étroite liaison avec le serveur, car elle va contenir la complète base de données du système ainsi que les logiciels de gestion de l'éclairage et de l'alarme.
- Connecté à la Raspberry PI 2 on trouvera un adaptateur qui va permettre la communication jusqu'à une carte Arduino Uno.
- Une carte Arduino Uno qui est donc connecté à l'adaptateur permet de gérer l'éclairage intérieur et extérieur de la maison.
- Des prises télécommandées seront donc liées à l'Arduino Uno afin de piloter l'éclairage extérieur et intérieur de la maison, en 230V.

En voici l'illustration de ces propos :



Voici comment le système va fonctionner, par rapport à chaque éléments qui le compose.



On remarque sur ce schéma l'illustration du système comme il vient d’être présenté, mais on remarque aussi une section "éclairage automatique".
Ce qui est nommé ici "éclairage automatique" est une gestion autonome de l’éclairage de la maison en cas d'absence prolongé des propriétaire (vacance par exemple).

Dès lors que les propriétaires partent pour une durée déterminé sur plusieurs jours ou semaines, ils leur suffit d'indiquer leurs dates de départ et de retour dans le logiciel créer a cet effet, et lorsqu'il sont partis le logiciel gère jusqu’à leurs retour l'éclairage de la maison. Cette autonomie permet, grâce à l’éclairage intérieur de la propriété, de simuler la présence de ses occupants.
Cette pratique sert de dissuasion aux intrus, ou peut apparaître comme technique de confusion si jamais des individus mal intentionné surveille les faits et gestes des propriétaires.

3- Répartition des tâches :

Voici comment le groupe, via une documentation fournie (Annexes 6 et 7), nous nous sommes répartit les tâches qui devront être accomplies pour mener à bien ce projet.

a- Spécification du rôle de chacun :

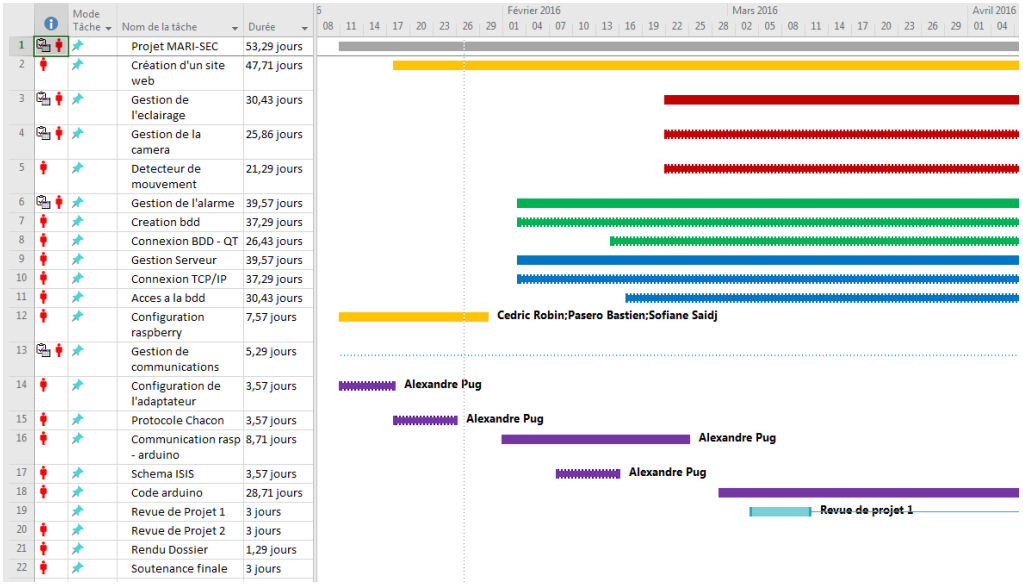
PASERO Bastien : Devra gérer la caméra et ce qu'il l'entoure à savoir, son intégration sur le site web ainsi que le programme de détection de mouvement.
Il s'occupera aussi de la gestion l'éclairage à savoir la communication entre la Raspberry PI et l'Arduino Uno ainsi que de la gestion des scénarii a mettre en place pour automatiser l'éclairage.
Enfin la gestion de la création du site web lui ait confié.

SAIDJ Sofiane : Devra gérer la communication entre le serveur PC Serveur qui contrôle localement le système et la Raspberry PI qui est, entres autres, la base de données du système. Il devra établir cette connexion via le web en suivant les contraintes du cahier des charges.

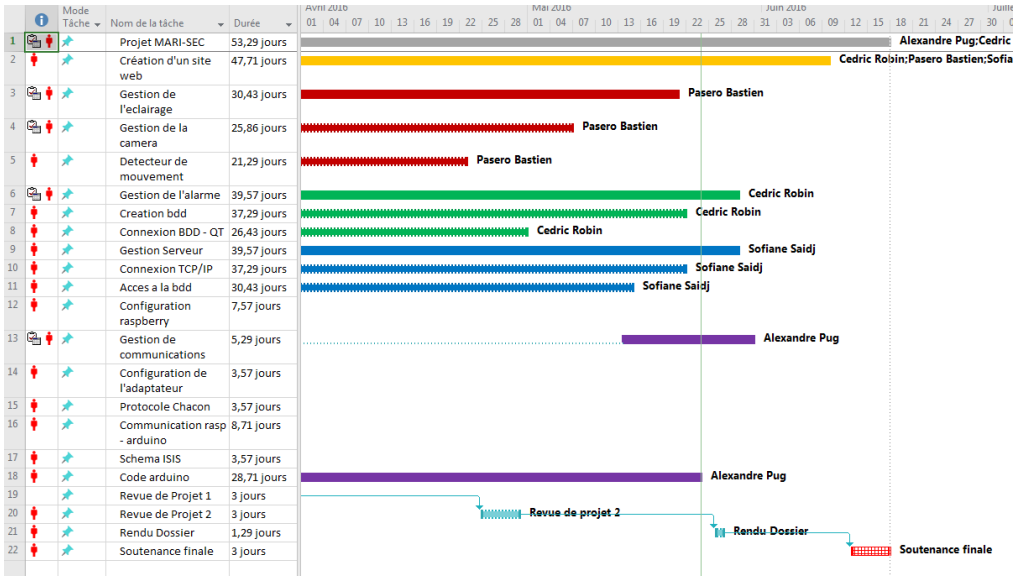
ROBIN Cédric : Devra gérer l'alarme ainsi que les composants qui l'entoure selon le cahier des charges. De plus la création et la gestion de la base de données lui aussi confié.

PUG Alexandre : Devra gérer la majeure partie du système du coté électronique, à savoir la communication entre l'Arduino Uno et les prises télécommandées.

4- Diagramme de GANTT prévisionnel :



	Mode Tâche	Nom de la tâche	Durée
1		Projet MARI-SEC	53,29 jours
2		Création d'un site web	47,71 jours
3		Gestion de l'eclairage	30,43 jours
4		Gestion de la camera	25,86 jours
5		Detecteur de mouvement	21,29 jours
6		Gestion de l'alarme	39,57 jours
7		Creation bdd	37,29 jours
8		Connexion BDD - QT	26,43 jours
9		Gestion Serveur	39,57 jours
10		Connexion TCP/IP	37,29 jours
11		Acces a la bdd	30,43 jours
12		Configuration raspberry	7,57 jours
13		Gestion de communications	5,29 jours
14		Configuration de l'adaptateur	3,57 jours
15		Protocole Chacon	3,57 jours
16		Communication rasp - arduino	8,71 jours
17		Schema ISIS	3,57 jours
18		Code arduino	28,71 jours
19		Revue de Projet 1	3 jours
20		Revue de Projet 2	3 jours
21		Rendu Dossier	1,29 jours
22		Soutenance finale	3 jours



	Mode Tâche	Nom de la tâche	Durée
1		Projet MARI-SEC	53,29 jours
2		Création d'un site web	47,71 jours
3		Gestion de l'eclairage	30,43 jours
4		Gestion de la camera	25,86 jours
5		Detecteur de mouvement	21,29 jours
6		Gestion de l'alarme	39,57 jours
7		Creation bdd	37,29 jours
8		Connexion BDD - QT	26,43 jours
9		Gestion Serveur	39,57 jours
10		Connexion TCP/IP	37,29 jours
11		Acces a la bdd	30,43 jours
12		Configuration raspberry	7,57 jours
13		Gestion de communications	5,29 jours
14		Configuration de l'adaptateur	3,57 jours
15		Protocole Chacon	3,57 jours
16		Communication rasp - arduino	8,71 jours
17		Schema ISIS	3,57 jours
18		Code arduino	28,71 jours
19		Revue de Projet 1	3 jours
20		Revue de Projet 2	3 jours
21		Rendu Dossier	1,29 jours
22		Soutenance finale	3 jours

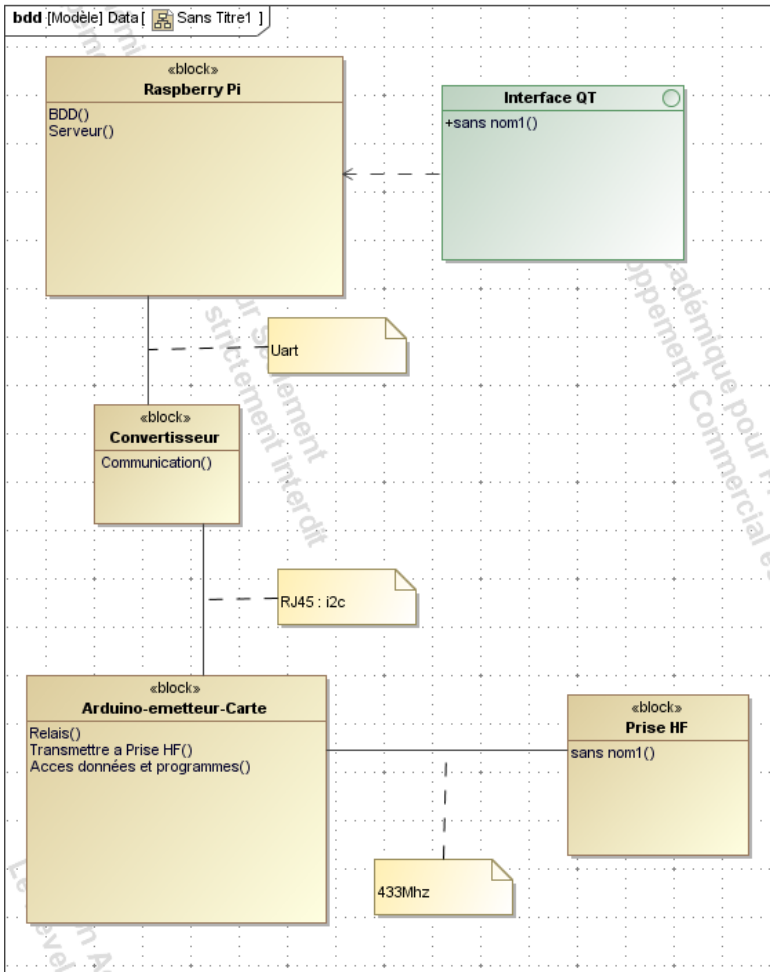
Partie Personnelle

Partie 1
Électronique et Communication

PUG Alexandre

a) Introduction

En tant que technicien EC ma tâche principale fut de créer la liaison entre l'éclairage et la Raspberry Pi. J'ai aussi en partie créé la liaison entre l'Arduino et la RPI ainsi que la carte et le boîtier permettant de lier l'Arduino, la RPI et l'émetteur HF. Il ne me manquera plus qu'à améliorer et créer une interface graphique permettant la gestion du projet via la RPI.



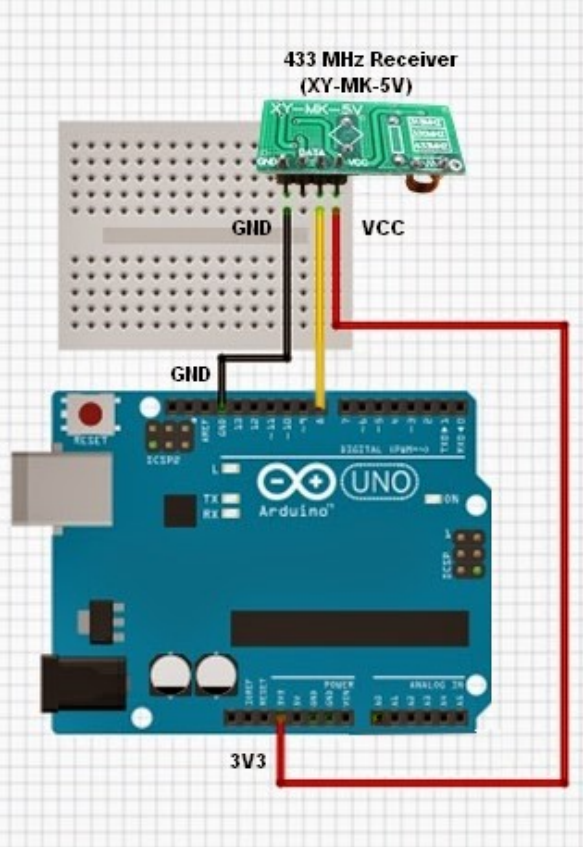
Schéma

b) Liaison Arduino/Eclairage

La gestion de l'éclairage est mise en scène via des prises Chacon qui sont des prises radiocommandées laissant passer ou non le courant selon l'ordre envoyé. L'objectif étant de pouvoir, via l'Arduino, communiquer avec ces prises et commander leur activité.

- lien du fabricant http://www.chacon.be/index.php/domotique/jardin-et-automatisme.html

Pour pouvoir comprendre le fonctionnement des prises Chacon, donc l'ordre envoyé par la télécommande pour les contrôler, ainsi que la vérification des données prises sur internet, j'ai mis en relation un Emetteur/récepteur 433mHz pour pouvoir lire et plus tard écrire les données et ainsi contrôler l'éclairage.



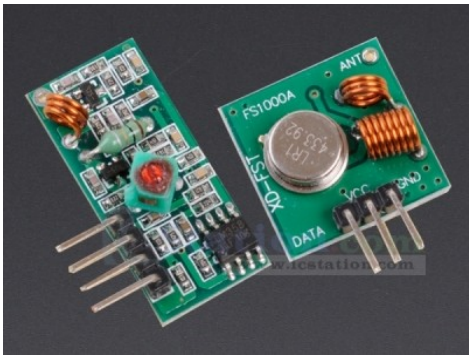
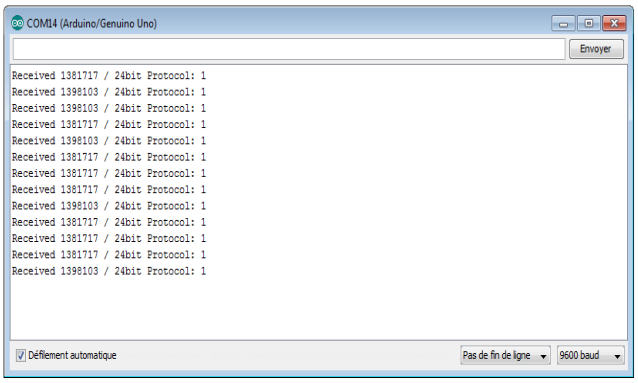
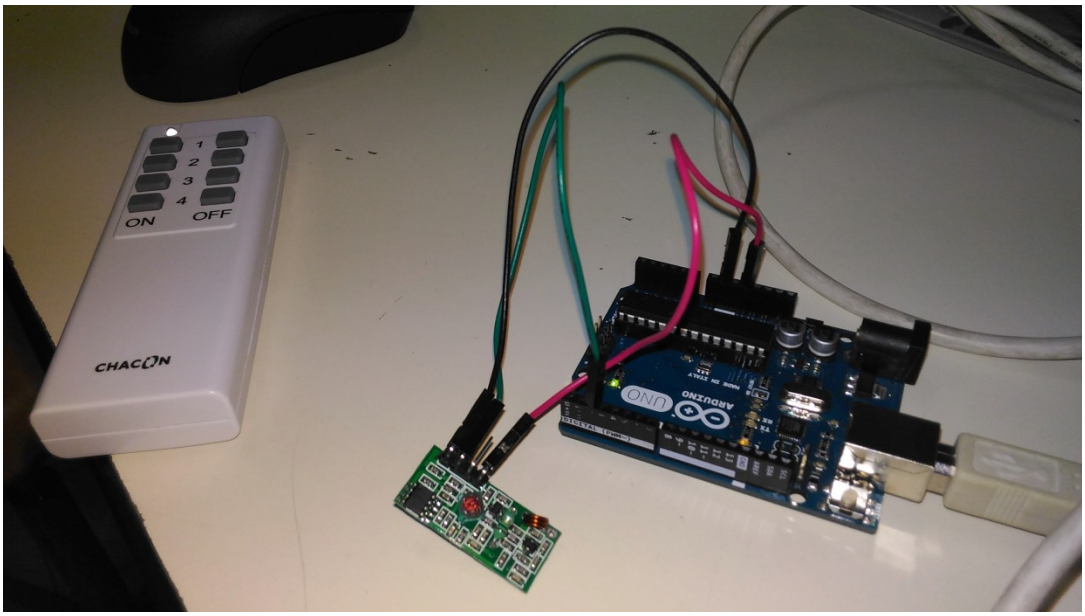


Schéma Fritzing du montage émetteur + Émetteur/récepteur

Tout d'abord j'ai câblé un récepteur 433mHz sur mon Arduino et récupéré un programme exemple dans les librairies RC-Switch. Ce programme exemple est conseillé sur plusieurs sites web pour les utilisateurs d'Arduino ainsi que pour les utilisateurs d'émetteurs/transmetteurs HF. Le nom du programme exemple utilisé pour lire les trames de la télécommande est « RC-Switch –Receive Simple ».

```
/*
  Simple example for receiving

  http://code.google.com/p/rc-switch/
*/

#include <RCSwitch.h>

RCSwitch mySwitch = RCSwitch();

void setup() {
  Serial.begin(9600);
  mySwitch.enableReceive(0);  // Receiver on inerrupt 0 => that is pin #2
}

void loop() {
  if (mySwitch.available()) {

    int value = mySwitch.getReceivedValue();

    if (value == 0) {
      Serial.print("Unknown encoding");
    } else {
      Serial.print("Received ");
      Serial.print( mySwitch.getReceivedValue() );
      Serial.print(" / ");
      Serial.print( mySwitch.getReceivedBitlength() );
      Serial.print("bit ");
      Serial.print("Protocol: ");
      Serial.println( mySwitch.getReceivedProtocol() );
    }

    mySwitch.resetAvailable();
  }
}
```





On peut voir sur la première image que la télécommande transmet 1381717 / 24bit Protocol : 1.
Cela signifie que c'est le signal qu'envoie la télécommande pour communiquer avec les prises. Il ne reste plus qu'à prendre un émetteur et ainsi envoyer ce code à la télécommande. On peut aussi appuyer sur les autres touches de la télécommande pour générer plusieurs options comme 1ON OFF, 2 ON OFF, ….

Le programme d'envoi a, à l'origine, plusieurs formats enregistrés. Cela facilite l'utilisation car on n'a plus qu'à récupérer un format d'envoi et le modifier à notre convenance.

Send Demo

```
/*
  Example for different sending methods

  http://code.google.com/p/rc-switch/

*/

#include <RCSwitch.h>

RCSwitch mySwitch = RCSwitch();

void setup() {

  Serial.begin(9600);

  // Transmitter is connected to Arduino Pin #10
  mySwitch.enableTransmit(10);

  // Optional set pulse length.
  // mySwitch.setPulseLength(320);

  // Optional set protocol (default is 1, will work for most outlets)
  // mySwitch.setProtocol(2);

  // Optional set number of transmission repetitions.
  // mySwitch.setRepeatTransmit(15);

}

void loop() {

  /* See Example: TypeA_WithDIPSwitches */
  mySwitch.switchOn("11111", "00010");
  delay(1000);
  mySwitch.switchOn("11111", "00010");
  delay(1000);

  /* Same switch as above, but using decimal code */
  mySwitch.send(5393, 24);
  delay(1000);
  mySwitch.send(5396, 24);
  delay(1000);
```

```
  /* Same switch as above, but using binary code */
  mySwitch.send("000000000000101010001000001");
  delay(1000);
  mySwitch.send("000000000000101010001000100");
  delay(1000);

  /* Same switch as above, but tri-state code */
  mySwitch.sendTriState("00000FFF0F0F");
  delay(1000);
  mySwitch.sendTriState("00000FFF0FF0");
  delay(1000);

  delay(20000);
}
```

Le protocole bit à bit

Déduction de nos données

En faisant des essais avec les différentes touches de la télécommande, voilà ce que j'en déduis :

0…51	: identifiant de la commande
52,53	: 01→ mode normal, 10→ global (touche G)
54,55	: 01→ ON, 10→ OFF
56… 59	: 0101→ A, 0110→ B, 1001→ C, 1010→ D
60… 63	: 0101→1, 0110→ 2, 1001→ 3, 1010→ 4

Cela peut paraître surprenant mais en cherchant de l'information sur le net, on trouve, qu'en fait, Chacon suit le protocole HomeEasy. Ce protocole dit que chaque trame fait 32 bits (moitié moins que ce que l'on a trouvé). L'explication, c'est l'encodage des bits. En fait, le bit 0 est encodé avec la pairede bit 01 et le bit 1 est encodé avec la paire de bits 10.

Si l'on convertit ce que l'on a trouvé en binaire, ça nous donne :

0101 1001 1001 0101 0101 1001 1001 0110 0101 0101 0101 0110 1001 0110 1010 0101

Décodons en transformant les 01 en 0 et les 10 en 1 :

00 10 10 00 00 10 10 01 00 00 00 01 10 01 11 00 , en regroupant, c'est plus lisible: 0010 1000 0010 1001 0000 0001 1001 1100

Le code Hex associé est : 28 29 01 9A

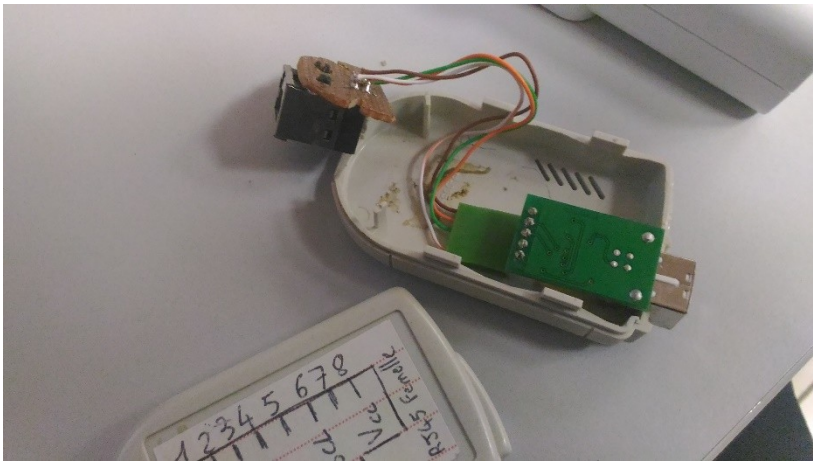
Le protocole HomeEasy

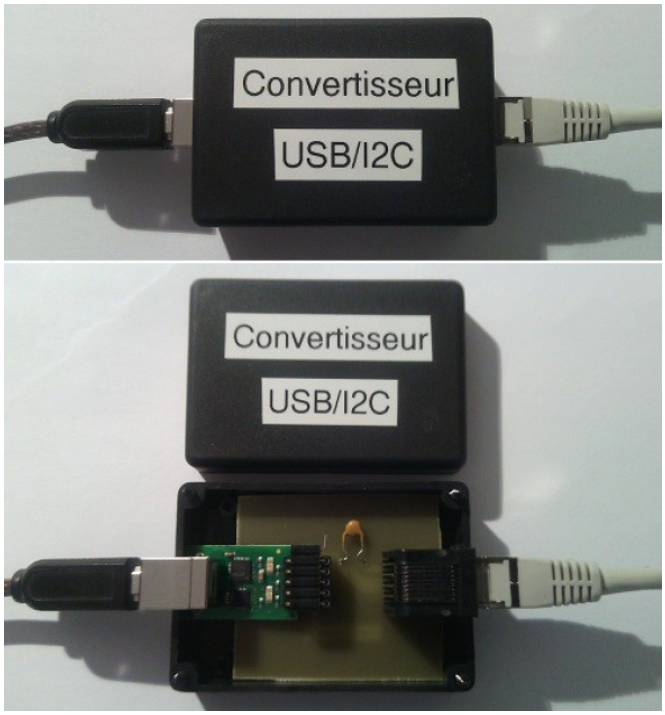
0…25 :	identifiant de la commande
26 :	0→mode normal, 1→global (touche G)
27 :	0→ON, 1→OFF
28,29 :	00→A, 01→B, 10→C, 11→D
30… 31 :	00→1, 01→2, 10→3, 11→4

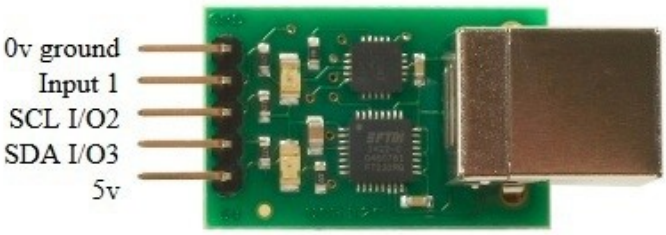
c) Liaison RPI/Arduino + RPI

La liaison entre la Raspberry Pi et l'Arduino a une contrainte qui est la différence de tension entre les deux appareils sur le bus I2C, il faut donc soit utiliser un Level shifter soit un adaptateur. Dans mon cas j'utilise un Adaptateur USB/RJ45, il permet la communication entre les deux appareils. Pour le mettre en œuvre, j'ai dû effectuer plusieurs manipulations permettant de voir si les données arrivaient bien à destination.

Le composant qui permet la manipulation est un FTDI FT232R USB chip, il manque néanmoins une partie nécessaire pour nous, la liaison avec l'Ethernet, mais on peut la rendre possible grâce aux broches de sortie. Nous nous sommes inspirés d'un ancien projet d'IR, mais par souci de qualité, nous avons créé un nouveau boîtier plus présentable.







Pour commencer et comprendre le fonctionnement du bus I2C sur la Raspberry Pi j'ai utilisé un TP fait précédemment en cours d'Electronique en modifant des lignes de codes pour lire un capteur de température. Le site de la partie USB du convertisseur m'a été bien utile car il faut s'adresser d'une manière particulière a l'installation. En effet, la partie USB fonctionne en Uart et la partie en RJ45 en I2C.

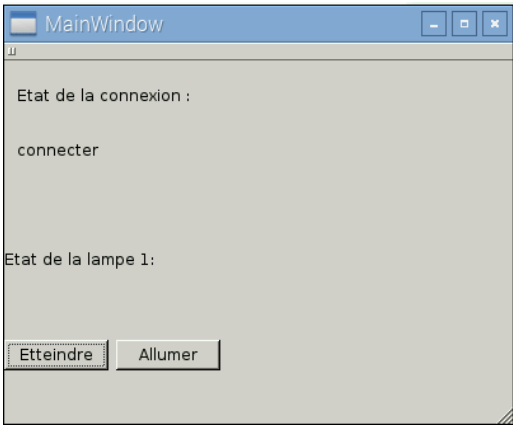
Lien du site : www.robot-electronics.co.uk/htm/usb_i2c_tech.htm

```
// Programme I2C Projet-Mari-SEC-1
#include <iostream>
#include <string.h>
#include <erno.h>
#include <wiringPi.h>
#include <wiringSerial.h>
#include <wiringPiI2C.h>
using namespace std;

#define ADRESS 0x45 //Adresse de la RPI

int main()
{
        int fd;
        int temp;
        wiringPiSetup^();
        if ((fd = serialOpen ("/dev/ttyUSB0", 19200))<0) //Ouverture de la communication en USB
{
cout<<"Impossible d'ouvrir le port /dev/ttyUSB0. Erreur : "<<strerror (erno);
return 1 ;
}
while (1)
{
serialPutchar(fd,0x53);//Ouverture du mode ecriture/lecture
serialPutchar(fd,0x10);//adresse de destination
serialPutchar(fd,0x01);//mot transmit
delay(5000); //delaie de 5secondes
serialPutchar(fd,0x53);//Ouverture du mode ecriture/lecture
serialPutchar(fd,0x10);//adresse de destination
serialPutchar(fd,0x00);//mot transmit
delay(5000); //delaie de 5secondes
}
return 0;
}
```

La communication via le port USBTY0 n'est possible que si l'adaptateur est branché, c'est pourquoi dans certains cas, en fonction de la manière dont est écrit le programme, il peut y avoir des erreurs. Le programme envoie 7 bit de données, mais le convertisseur écrit sur 8 bit, le huitième servant à dire s'il s'agit d'un envoi ou d'une écriture. Il faut donc prévoir de convertir l'adresse pour permettre la bonne réception. (L'Arduino a l'adresse 8 mais la Raspberry l'envoie à l'adresse 10).

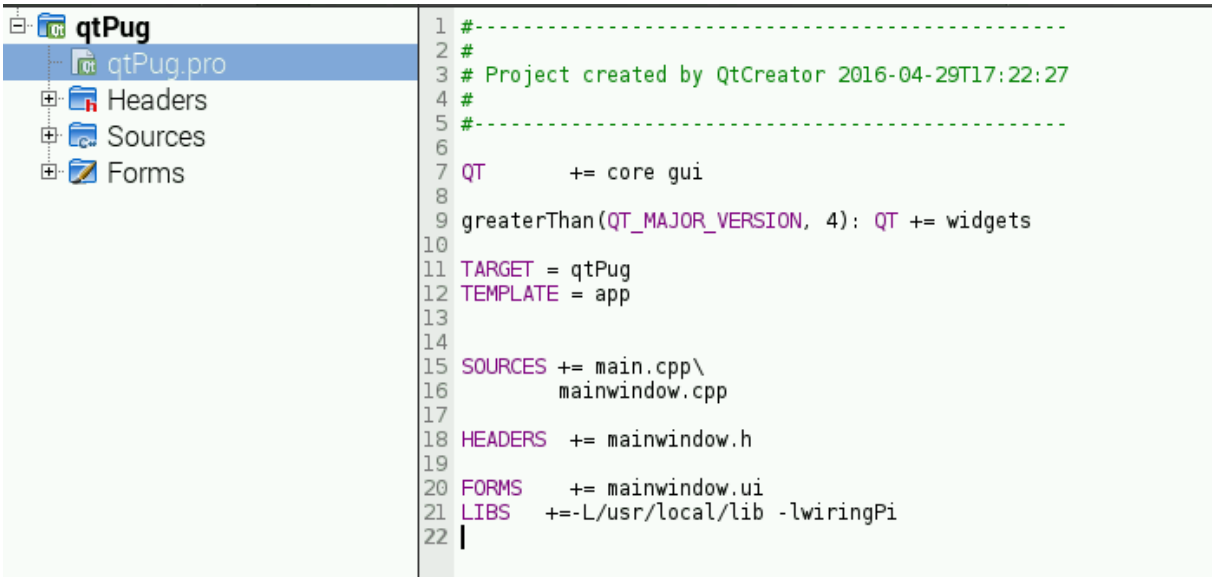
Interface QT :

Pour la réalisation de mon projet, ainsi que pour aider le technicien IR (au final il est obligé d'utiliser une autre librairie donc mon travail est plus personnel), j'ai fait une interface QT avec QTcreator sur ma RPI.



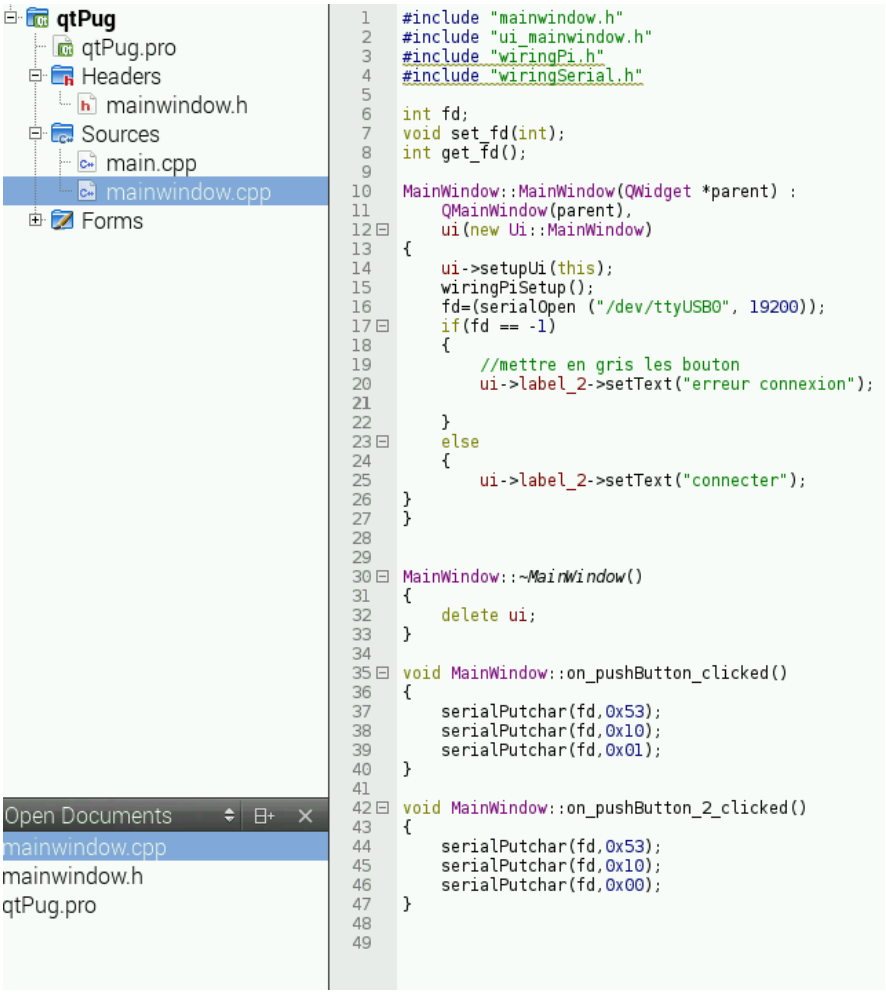
L'interface est encore très basique car il s'agissait plus d'une expérience pour voir si j'étais capable de créer l'interface et de mettre en œuvre ces librairies. C'est pourquoi l'interface ne fait pour l'instant qu'indiquer si la connexion marche et n'a qu'une lampe à commander.

Pour la création de l'interface QT, pour gérer la communication, j'ai gardé, comme dans mon programme en Bash, la librairie Wiring.pi qui fonctionne aussi sur QT créator.



Pour que la librairie fonctionne sous QT il faut l'inclure comme ceci dans le fichier principal, la ligne de commande l'incluant est la ligne avec LIBS .

Ensuite dans la fenêtre principale, on peut voir l'inclusion de wiringpi.h est wiringSerial.h qui fonctionne grâce à la ligne précédente. Le reste du programme permet de créer le port série, d'indiquer si il est fonctionnel et d'envoyer les données si l'action cliquée est effectuée sur les touches.



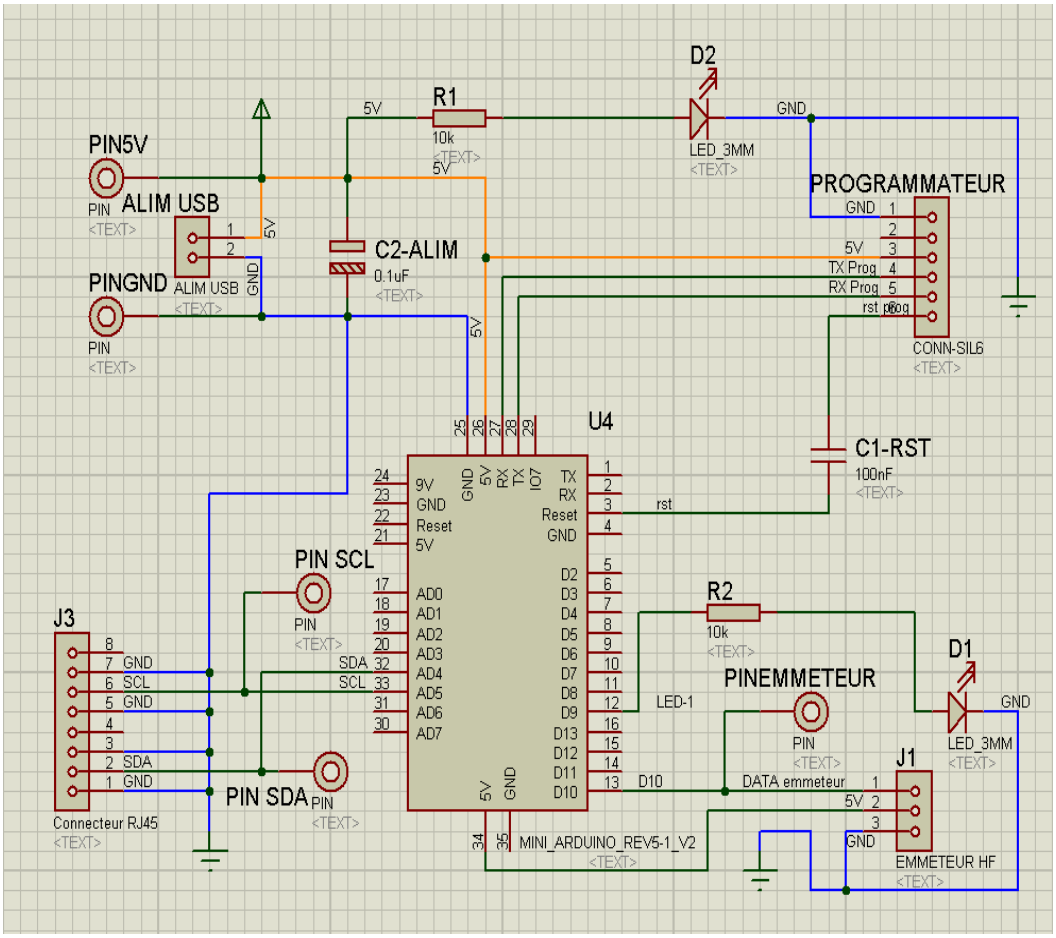

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "wiringPi.h"
#include "wiringSerial.h"

int fd;
void set_fd(int);
int get_fd();

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    wiringPiSetup();
    fd=(serialOpen ("/dev/ttyUSB0", 19200));
    if(fd == -1)
    {
        //mettre en gris les bouton
        ui->label_2->setText("erreur connexion");

    }
    else
    {
        ui->label_2->setText("connecter");
}
}


MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::on_pushButton_clicked()
{
    serialPutchar(fd,0x53);
    serialPutchar(fd,0x10);
    serialPutchar(fd,0x01);
}

void MainWindow::on_pushButton_2_clicked()
{
    serialPutchar(fd,0x53);
    serialPutchar(fd,0x10);
    serialPutchar(fd,0x00);
}
```

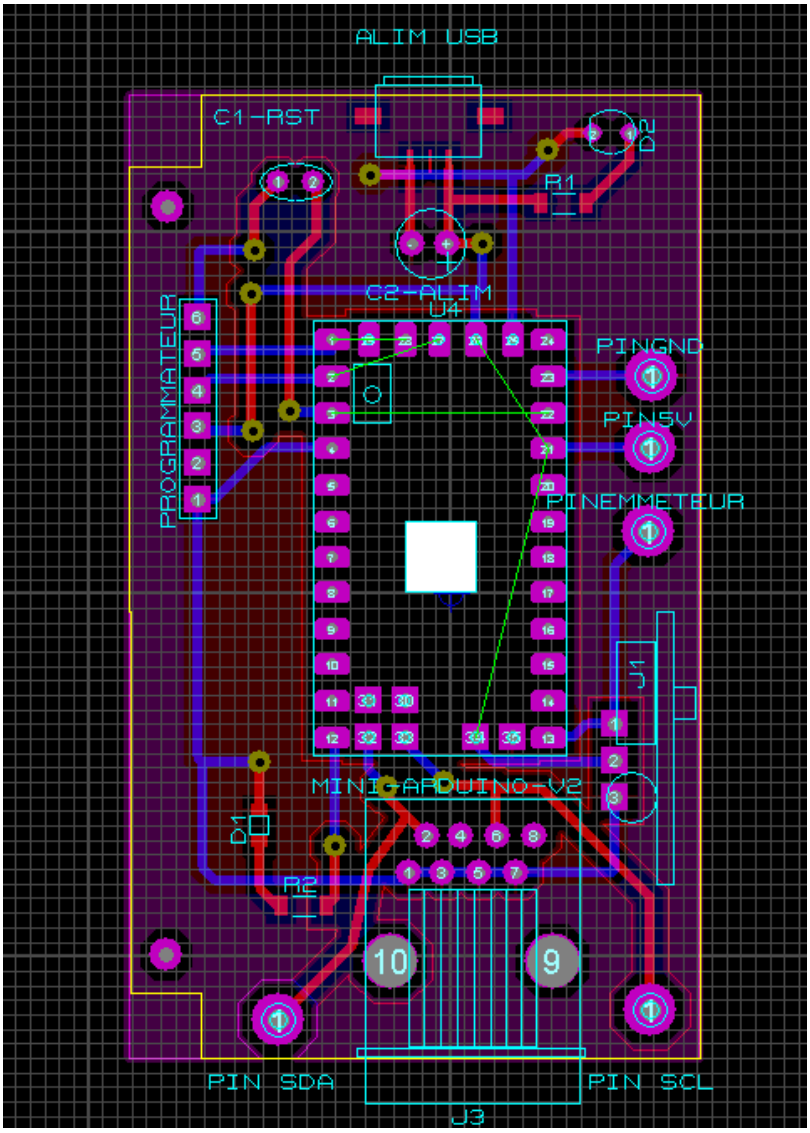
d) Fabrication

Pour l'achèvement du projet le technicien EC doit accomplir une fabrication de carte pour permettre une meilleure présentation. Pour le projet Mari-SEC la carte créée doit servir à réunir la carte Arduino mini, des principaux connecteurs ainsi que de PIN pour permettre de récupérer des informations (GND, ALIM, DATA) plus facilement.

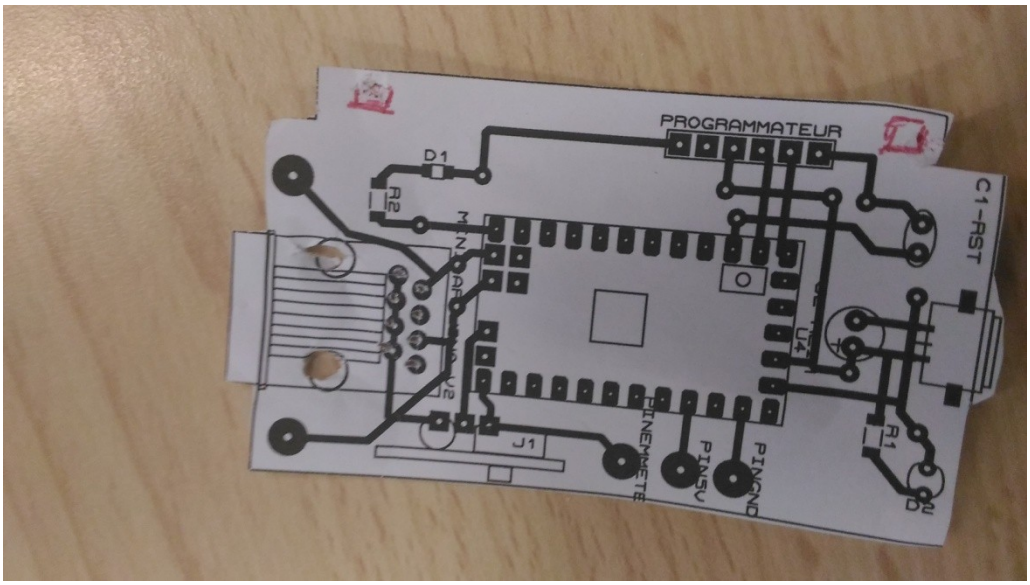
Avant de commencer la création de la carte, il faut en faire un schéma sur Proteus : ISIS. Sur ce schéma on peut y entrer les valeurs des composants, les composants, le câblage et ISIS peut permettre de faire des calculs physiques sur le système.

Une fois le schéma sur ISIS terminé, on peut « transférer » ce schéma sur ARES de Proteus. Ce logiciel permet de prendre le schéma et de créer virtuellement la future carte. On peut donc y délimiter la taille de la carte ainsi que la manière dont vont passer les différentes pistes la traversant.

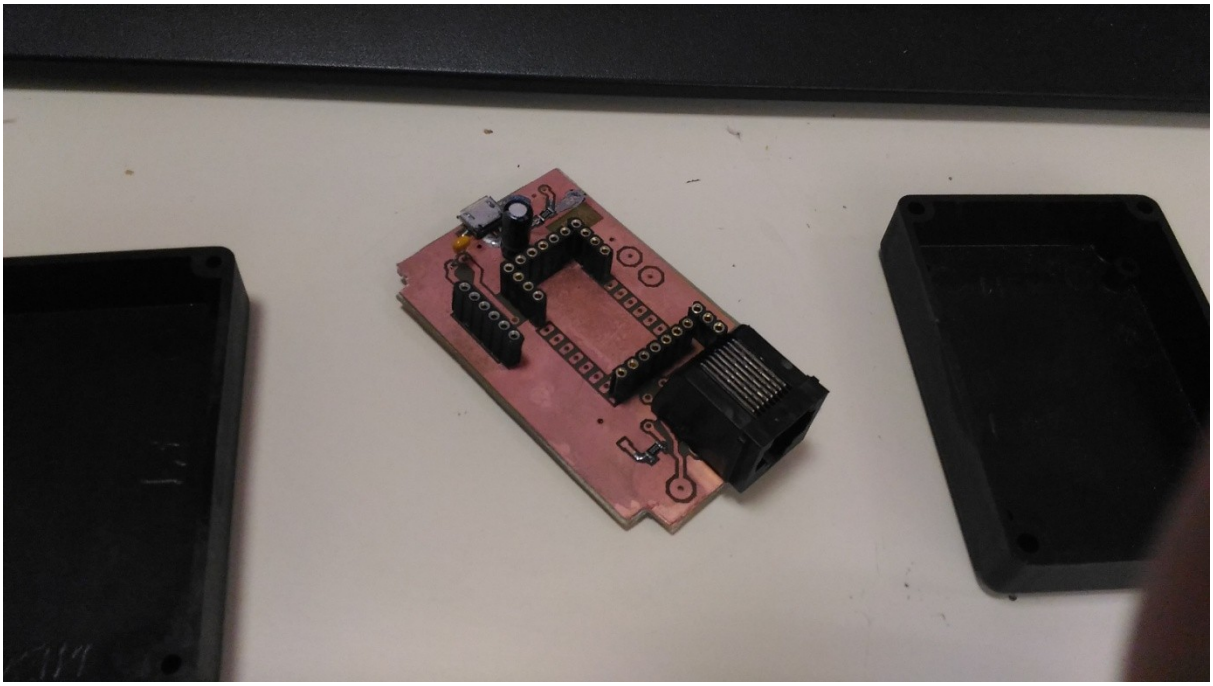




Il y a aussi un fichier Gerbert dans les dossiers fournis avec la revue de projet.



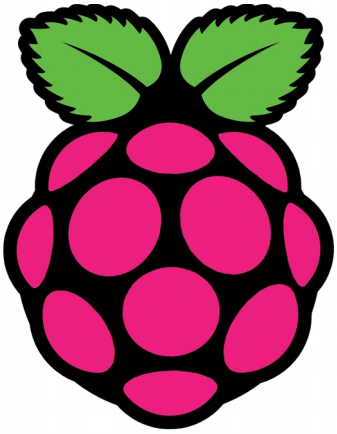
Avant d'imprimer la carte j'ai imprimé son plan papier pour vérifier qu'elle correspond bien aux dimensions du boîtier. Après avoir vérifié que la carte respecte bien cette contrainte j'ai pu lancer la fabrication physique de la carte.
Pour créer la carte il a fallu imprimer le circuit sur un transparent. Il faut faire attention au sens de l'impression pour que l'encre soit appliquée sur le transparent du côté qui sera plaqué sur le cuivre. Il faut ensuite utiliser une insoleuse pour coller le circuit sur la plaque (carte), il faut faire attention au fait que les composants soient bien collés. Après rinçage dans du perchlorure de fer, on peut considérer la carte comme finie.



Exemple carte finie (il manque quelques composants)

Partie 2
Informatique et réseaux

PASERO Bastien



RaspberryPi

AXIS COMMUNICATIONS

SOMMAIRE

1- Partie individuelle du projet
1.1- Présentation……..….p.32
1.2- Les solutions existantes…….p.33

2- Revue de projet 1
2.1- Le site Internet…….….p.35
a- Introduction…….…...p.35
b- Serveur WAMP……..p.35
b.1- Introduction……..……..……..p.35
b.2- Installation de WAMP……...……...p.35
b.3- Utilisation de WAMP……....p.38
c- Création du site Internet…….p.38
c.1- Explications du code…….p.41
2.2- La caméra IP…….p.42
a- Étude de la caméra IP……..…..p.42
2.3- Partie éclairage…….p.42
a- Introduction……....p.42
b- QSerialPort……..…….……p.43
b.1- Introduction…….….p.43
b.2- Installation…….p.43
b.3- Utilisation…….p.44
2.4- Conclusion de la revue 1……..p.44

3- Revue de projet 2
3.1- Le site web……p.45
a- Les modifications……...p.45
a.1- Page d'identification……..p.45
a.2- Explication des modifications…..…….p.45
b- La caméra……..p.47

b.1- Résolution des problèmes…….p.47
b.2- L'intégration……..p.47
c- Suite……...…p.48
3.2- Éclairage……..…p.49
a- Pilotage des lumières……p.49
a.1- Introduction……..…p.49
a.2- Fonctionnement Physique……p.49
a.3- Le logiciel……..…p.50
a.3.1- Lecture de la base de données……p.50
a.3.2- Communication série……...p.53
a.4- Conclusion……p.55
b- Interface Homme-Machine……p.57
b.1- Introduction……..p.57
b.2- Objectifs……...p.56
3.3- Caméra……...p.57
a- Introduction……p.57
b- Caractéristiques de la caméra……..p.58
c- Détection de mouvement……p.59
c.1- Introduction……p.59
c.2- Installation de JAVA et ECLIPSE……...……p.59
c.3- Installation de OpenCV……..p.61
c.4- Explications……..…p.69
c.4.1- Le Programme……...…p.69
c.4.2- L'exécutable……..…p.69
d- Suite……p.71

4- Synthèse des prévisions……..……p.71

5- Conclusion Finale……..……p.72

ANNEXES

1- Partie individuelle du projet

1.1- Présentation :

Dans ce projet, qui prolonge les activitées de la société "Mariton" dans le domaine de la domotique, il s'agit de gérer la sécurité d'une maison type, grâce à un système composé principalement d'une caméra IP de surveillance, d'une alarme, d'éclairage extérieur et intérieur, le tout géré par plusieurs cartes électroniques ainsi qu'un site web pour y accéder, le tout pour prévenir les intrusions. De plus ce système gère la gestion de l'éclairage qui sera aussi abordée pour la simulations de présence des occupants, selon différents scénarii, en fonction de certains critères. Pour ce projet nous sommes donc une équipe de quatre pour nous répartir les différentes tâches pour mener à bien ce projet.

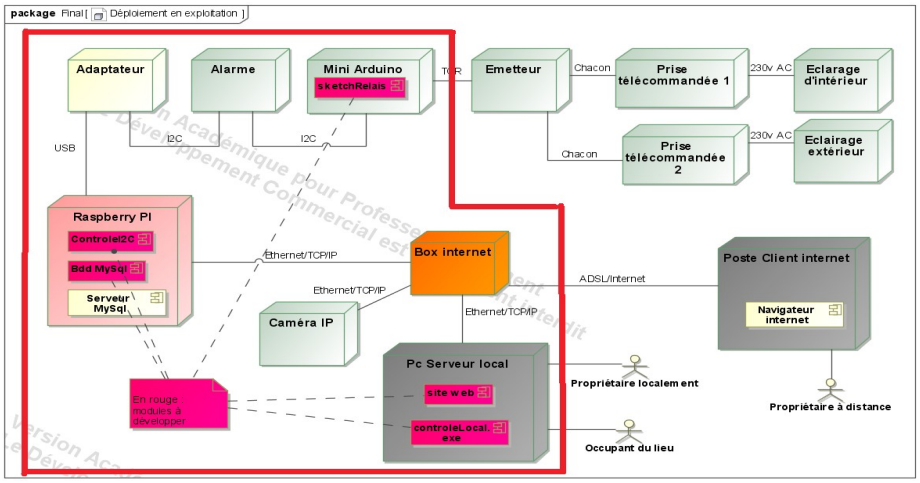
En ce qui me concerne je suis principalement chargé de la gestion de la caméra IP, située à l'extérieur, qui permettra de détecter la présence d'intrus et qui, de par cette détection, allumeras l'éclairage extérieur et préviendra aussi le propriétaire de cette intrusion. De plus Je m'occupe du site web qui sera accessible depuis le Internet pour le propriétaire, sur lequel le flux vidéo de la camera sera diffusé en direct, il y aura aussi un journal d'évènement afin de fournir au propriétaire les informations, sous forme de liste, sur toutes les actions effectuées par le système au cours de la journée. Pour finir je gère aussi la gestion de l'éclairage, à savoir le pilotage de celui-ci ainsi que la création et la mise en application des scénarii de simulation de présence.

Représentation schématique de la composition du système :

Légende :

Lorsque le propriétaire quitte sa maison, il enclenche donc le simulateur de présence afin que le voisinage mal intentionné ne remarque son absence, grâce à l'éclairage automatique des pièces. En cas d'intrusion, voici comment se coordonnent ces éléments, la caméra IP détecte un mouvement grâce au programme de détection de mouvement intégrer dans le PC Serveur, il écrit donc dans la base de données qu'il y a un mouvement, les programme d'alarme et d'éclairage qui lisent sans cesse dans cette base de données, sont donc alerté de l'intrusion, l'éclairage extérieur s'allume, l'alarme aussi, la vidéo s'enregistre dans la base de données et le propriétaire est prévenu et peut observer via le site web.

Et voici selon le schéma structurel ce qui les parties en rapport avec mon travail :

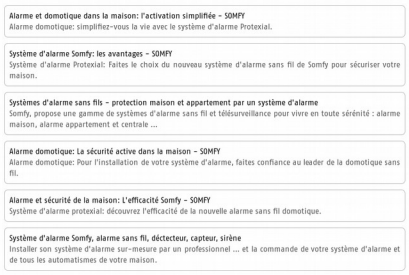


1.2- Les solutions existantes :

Acquérir un système de sécurité pour sa maison est assez simple de nos jours, en effet simplement grâce à une recherche sur le web on trouve facilement des packs complet à installer, qui contiennent tout les éléments à installer pour assurer la sécurité de sa maison. Comme par exemple ce pack trouver sur les sites suivants :

amazon.fr :

ou sur somfy.fr :



ou encore sur conrad.fr :

Ce sont des systèmes déjà tout prêt trouver sur internet, pour la plupart ils peuvent être installer de nous-même, mais généralement les sites conseillent l'installation d'un expert, pour les garanties.
De plus il existe plusieurs tutoriels pour faire un système maison avec Raspberry, notamment, qui sont vraiment très proche de ce que l'on fait dans ce projet, qui utilise une Raspberry PI une camera IP mais qui ne gèrent pas l'éclairage.

2- Revue de projet 1

2.1- Le site Internet :

a- Introduction :

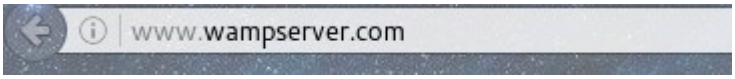
Tout d'abord j'ai prit la décision de commencer par la connexion de la camera, car je n'avais encore jamais travaillé avec des camera IP donc je pensais que cela prendrais du temps. La camera se connecte sur le site pour fournir une image au visiteur (le propriétaire depuis l'extérieur) donc pour cela j'ai dû créer le site web. Pour cela j'ai d'abord fait un plan pour qu'avec l'équipe nous soyons d'accord sur l'aspect du site a terme, en particulier de la page "Tableau de bord" car c'est elle qui contient le flux vidéo en direct ainsi que le journal d'événements.

Pour affiché un site web local sur un ordinateur il faut installer un serveur, en effet les pages HTML simple peuvent être directement interprétée par le navigateur mais lorsqu'elles contiennent du PHP il faut faire appel à un interpréteur de PHP qui est utilisé dans un serveur, le serveur hébergeras donc le site localement afin de l'afficher (cette méthode ne permet pas de gérer beaucoup de visites simultanées)

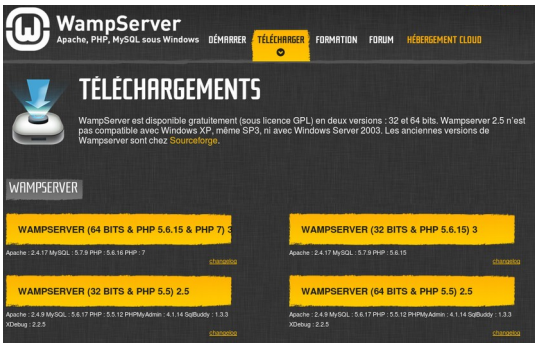
Pour que le site s'affiche sur le PC il m'a donc fallu installer "WAMP" , serveur dédié au système d'exploitation Windows.
WAMP signifie "Windows(système d'exploitation), Apache (serveur web), Mysql (serveur de base de données), Python (langage de script). Il permet de mettre en place un serveur web local, sur n'importe quelle machine Windows. C'est un outil très utile pour de petits sites ou applications web seulement car une machine basique ne peut gérer de gros flux, et ne peut supporter trop de connexions simultanées.

b.2- Installation de WAMP :

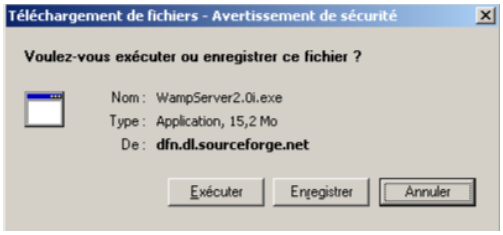
- Tout d'abord il faut commencé par télécharger le logiciel en lui-même, à cette adresse :

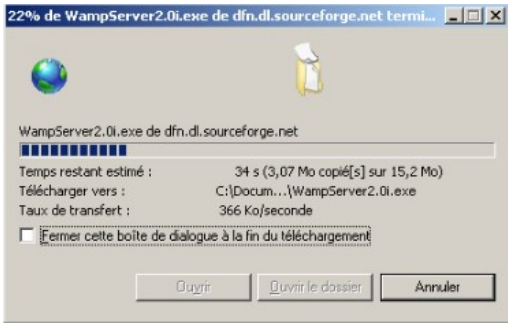


- Et choisir la version qui correspond le mieux à notre système :

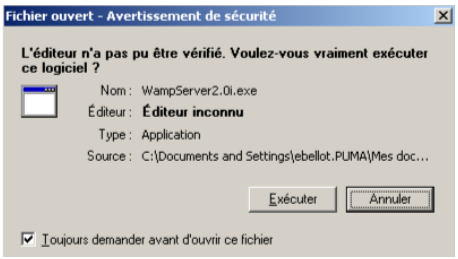


- Ensuite on enregistre et on patiente pendant le téléchargement :

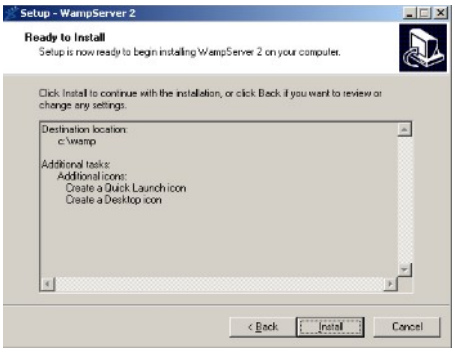


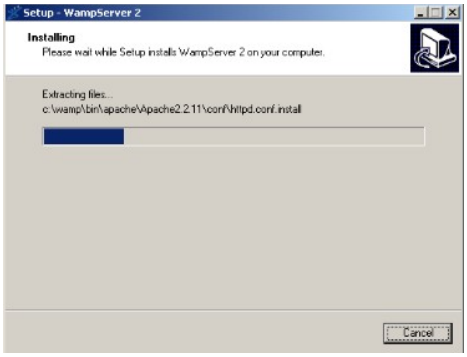


- Ensuite on exécute le programme et on arrive sur l'assistant d'installation :



- On suit l'assistant d'installation jusqu'à terminer l'installation :

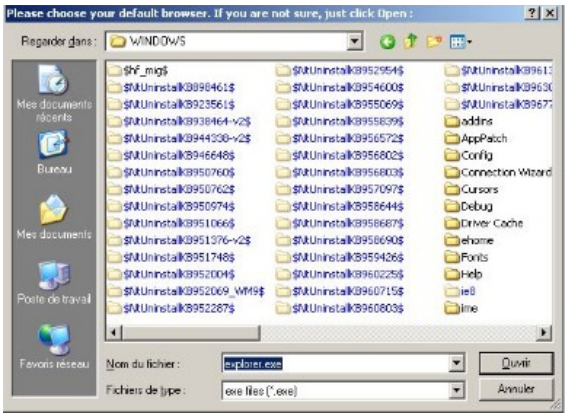




- On choisit ensuite le navigateur que l'on veut que WAMP utilise :

Par exemple personnellement j'ai utilisé Mozilla Firefox, j'ai donc utilisé ce chemin :

→ ***C:\Program Files\Mozilla Firefox\firefox.exe***



- Une fois le navigateur choisit l'installation se termine :

b.3- Utilisation de WAMP :

-Pour démarrer les services WAMP, il faut aller dans "Démarrer"→ "Tous les programmes" → "WAMP Server" → "Start Wampserver".

Une icône dans la barre des tâches apparaîtra alors et lorsqu'elle sera verte le service WAMP sera désormais effectif.

-Ensuite pour placé un site web, afin de l'afficher avec WAMP serveur il faut placé l'intégralité du site dans le dossier suivant :

→ ***C:\wamp\www***

-Pour afficher la page web d'accueil de WAMP il suffit d'inscrire dans la barre URL du navigateur "LocalHost".

L'installation de WAMP serveur est désormais terminé.

c- Création du site Internet :

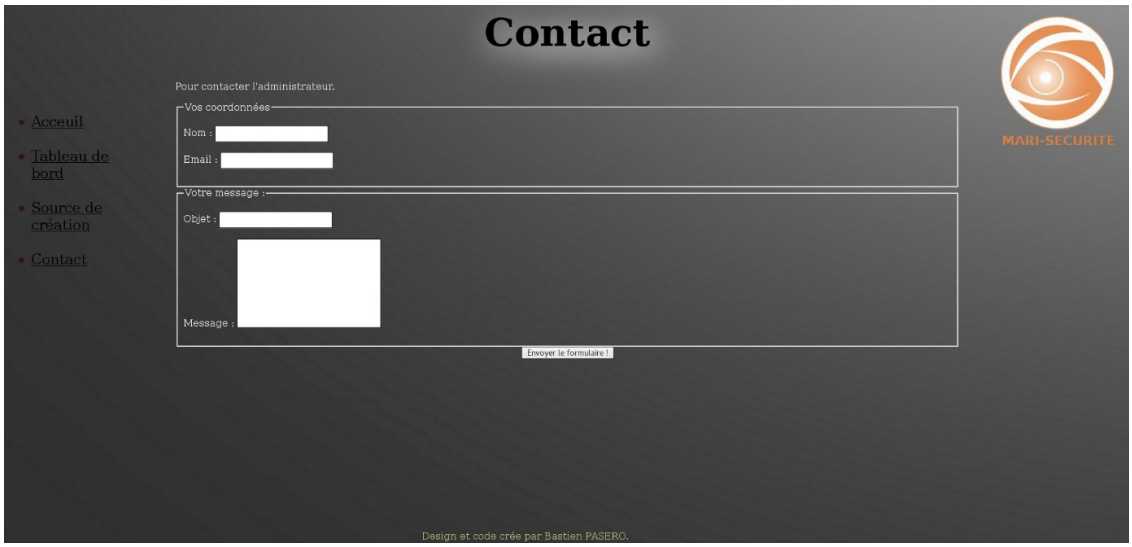
Ensuite comme indiqué sur mon journal de bord personnel (Annexe 10) j'ai codé les pages web dans l'ordre suivant, en commençant par le tableau de bord, ensuite la page d'accueil, les sources (les sources sont les pages qui m'ont été utiles pour créer le site web) et pour finir la page de contact qui a été un peu plus laborieuse, car il m'a fallu entamer des cours sur le PHP pour la construire. Et j'ai finalement terminé par l'intégration de la page d'identification au site.

Tableau de bord :

Page d'accueil :

Page des sources :

Page de contact :



c.1- Explications du code :

Le seul travail que j'ai dû fournir c'est du code,mais ici je vais surtout expliquer les techniques que j'ai utilisé plutôt que présenter du code brut.
Bien que le code en lui-même n'est pas très compliqué, créer un site web est assez laborieux et demande de l'organisation dans les dossiers et dans le code en lui-même.

Voici ci-dessous un extrait, un code-type, qui est le code de la page d'accueil. On peut remarquer l'organisation que j'ai choisi, à savoir, en premier lieu l'intégration de portion de page par PHP (les "include"), comme le menu de navigation (<nav>) ou encore le pied-de-page (<footer>). Ensuite j'ai utilisé le même CSS (le code CSS gère la mise en forme des pages d'un site web) pour toutes les pages afin que le site soit moins lourd, excepté le tableau de bord pour permettre de gérer les différents éléments spéciaux (flux vidéo, journal d'événements).
De plus j'ai fait en sorte que le code soit clair et lisible car il est a votre disposition si vous souhaitez voir les autres pages qui le compose. Ces dernières varient légèrement de l'exemple que je vous donne mais suivent le même schéma établi.

Le code :

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" href="../style.css" />
        <meta charset="utf-8" />
        <title> Acceuil </title>
    </head>

    <body>
        <h1> Bienvenue! </h1>

        <nav>
            <?php include("../include/nav.html"); ?>
        </ul>
        </nav>

        <article class="index">
                <p>
                    Bienvenue sur votre site entièrement dédié a la surveillance et a la protection de votre domicile. </br>
                </p>

                <div class="surveillance" >
                    <img src="../Images/surveillance1.png" alt="souriez vous êtes filmés" height="650" width="640" >
                </div>
        </article>

        <footer>
                <?php include("../include/footer.html"); ?>
        </footer>

    </body>
</html>
```

2.2- La caméra IP :

a- Étude de la caméra IP :

En parallèle du site web et des documents de suivi à faire que l'on doit fournir pendant le projet, j'ai commencé l'étude de la caméra.
En effet pour réussir à intégrer le flux vidéo de la camera en direct sur le site web, il me fallait d'abord récupérer l'adresse IP de la camera pour pouvoir ensuite la paramétrer sur le même réseau et ensuite intégrer tout cela dans le code du site web pour que tout fonctionne.
C'est la partie qui a été la plus laborieuse car la caméra IP qui m'a été fourni est une caméra IP standard, sans marque donc sans documentation, Il m'a donc été impossible de trouver son adresse IP, de plus l'utilisation de WireShark (programme qui analyse ce qui transite sur le réseau) n'a pas été fructueuse.
Après avoir passé de nombreux logiciels de scan d'adresses IP le problème n'a pas été résolu.

2.3- Partie éclairage :

a- Introduction :

Ici, la partie concernant l'éclairage se résume à l'installation de "QSerialPort" sur Raspberry PI. QSerialPort est une bibliothèque qui via QT-Creator, va permettre d'envoyer des trames sur une liaison série, dans notre cas ce sera par USB. Dans ce projet le gestion de la partie éclairage (pour un IR) sera de réussir à écrire sur cette liaison série afin de communiquer avec l'Arduino pour pouvoir piloter l'éclairage, et mettre en place un logiciel de gestion et d'application de scénarii de présence, en cas absence des propriétaires.

b- QSerialPort :

b.1- Introduction :

QSerialPort est donc une bibliothèque de QT-Creator, qui permet de fournir une interface unique pour le matériel et les ports série virtuels. QSerialPort contient deux classes essentielles afin de gérer les ports série, la classe "QSerialPortInfo" pour accéder à tout type d'information des ports série et la classe "QSerialPort" afin d'utiliser les ports série selon nos besoins (les ouvrir/fermer, écrire/lire, etc...)

b.2- Installation :

- Tout d'abord on commence par récupérer l'archive de la bibliothèque, elle peut être télécharger depuis Git grâce à cette commande (on utilise "sudo" pour avoir les droits administrateur) :
  → ***sudo git clone git: // code.qt.io / qt / qtserialport.git***

Pour ma part l'archive (le ".zip") m'a été fournit par mon professeur.

- Une fois l'archive décompresser on entre dans le dossier principal :
  → ***cd ./qtserialport***

- Et on y crée un dossier de compilation que l'on nomme "qtserialport-build", c'est dans ce dossier que l'on va mettre les fichier de compilation :
  → ***sudo mkdir qtserialport-build***

- Ensuite on va crée les "fichiers actions" (MakeFiles) ce sont les fichiers qui vont permettre de construire toute la bibliothèque (compiler toutes les pages de code qui compose cette bibliothèque) :
  → ***sudo qmake ../qtserialport/qtserialport.pro***

- On va ensuite exécuter ces "fichiers actions" :
  → ***sudo make***

- Et pour finir on installe la bibliothèque :
  → ***sudo make install***

b.3- Utilisation :

- Pour utiliser la librairie "QSerialPort" dans QT-Creator il faut modifier le ".pro" en rajoutant cette ligne :

```
CONFIG + = serialport
```

-Et enfin il ne reste plus qu'à inclure les fichiers d'en-tète dans le(s) "include" dans le(s) ".cpp" où l'on utilise QSerialPort :

```
#include <QtSerialPort/QSerialPort>
#include <QtSerialPort/QSerialPortInfo>
```

2.4- Conclusion de la revue 1 :

Nous sommes maintenant à la première revue de projet, donc pour conclure, le site web à déjà bien avancé (voir quasiment terminer) et il est fonctionnel bien que je suppose qu'il sera quand même légèrement améliorer avant la fin du projet.

Il me reste donc à résoudre le problème de la caméra, créer un programme pour qu'elle puisse détecter les mouvements lorsqu'il y a lieu de cela, et qui permet d'envoyer les alertes, pour que la gestion de l'alarme et de l'éclairage soit déclenchées. Il faut aussi que je me penche sur l'aspect éclairage du système, que je doit aussi gérer.

Je remarque pour l'instant que la partie que je viens d'exécuter n'a posé que peut de soucis, hormis celui de la caméra, les problèmes rencontré pour la création du site sont rester basique, donc aucun soucis majeur pour le site web.

3- Revue de projet 2 :

3.1- Le site web :

a- Les Modifications :

a.1- Page d’identification :

Après avoir terminé le site il ne lui manquais plus que la page d'identification, cette page est la page qui a une double fonction. Elle sert en premier lieu à la sécurisation du site, elle permet d'exiger au visiteur un identifiant et un mot de passe afin d’accéder aux informations qui sont disponibles sur ce site web, en effet cette sécurité est primordiale car le site permet l'accès à certains équipement de sécurité d'une propriété privée.
La seconde fonction de cette page permet de se connecter à la base de données de la maison afin que le site accède aux données de la maison, cela permet en cas d'attaque sur le site de na pas avoir accès aux données sensible de la maison (si personne n'est connecté).

a.2- Explication des modifications :

Donc la page d'identification a été confié à l'IR2, afin de permettre d'avancer plus rapidement sur ce projet. Après qu'il est rencontré certains problèmes je me suis donc à nouveau penché sur le code pour trouver une solution.

Je vais donc vous présenté ma solution, sous forme de code commenté, et expliquer cela.

```php
<?php

session_start();    // Ouverture d'une session

if(isset($_POST['connexion']))      // On entre dans la fonction lorsque on appuie sur le bouton "Envoyer"
    {

        if(empty($_POST['utilisateur']))   // On vérifie si le champ "Identifiant" a été rempli
        {
            echo "Le champ Identifiant est vide.";
        }

        else
        {
            if(empty($_POST['mdp']))    // On teste si un mot de passe a été saisi
            {
                echo "Le champ Mot de passe est vide.";
            }

            else
            {
                $Pseudo = htmlentities($_POST['utilisateur'], ENT_QUOTES, "ISO-8859-1");    // Le htmlentities() passera les guillemets en entités HTML, ce qui empêchera les injections SQL
                $MotDePasse = htmlentities($_POST['mdp'], ENT_QUOTES, "ISO-8859-1");

                $mysqli = mysqli_connect("192.168.2.185", "$Pseudo", "$MotDePasse", "Mari-Sec");  // Connexion à la base de données


                if(!$mysqli)  //on teste si la connexion a reussie
                {
                    echo "Erreur de connexion à la base de données.";
                }

                else
                {
                    echo "Vous êtes à présent connecté !";
                    header('Location: /sites/Site projet mari-sec/Acceuil/Acceuil.php');   // On affiche la page d'acceuil du site
                }
            }
        }
    }

?>
```

Maintenant que vous avez vu le code de cette page voici comment il fonctionne.

La première ligne permet d'ouvrir une session afin de dialoguer avec le serveur SQL (serveur de base de données), c'est une pratique propre au PHP.

Ensuite lorsque l'on appuie sur le bouton "Envoyer" de la page HTML, on va commencer par vérifier si tout les champs ont été remplis par l'utilisateur, de la ligne 8 à la ligne 18.

Les lignes 22 et 23 permettent de sécuriser le passage des données entrées par l'utilisateur dans des variables, grâce à "htmlentities()". Ensuite "ENT-QUOTES" permet de garder la signification des données passé avec "htmlentities()".

Une fois ceci fait on établi la connexion à la base de données de la maison avec le nom et le mot de passe qui a été fourni par le visiteur à la ligne 25, et pour finir les lignes qui suivent permettent de vérifié si la connexion à échoué et d'envoyer un message d'erreur ou si au contraire elle a réussie de nous renvoyé sur la page d’accueil du site. (Code fournis avec le dossier)

b- La caméra :

b.1- Résolution des problèmes :

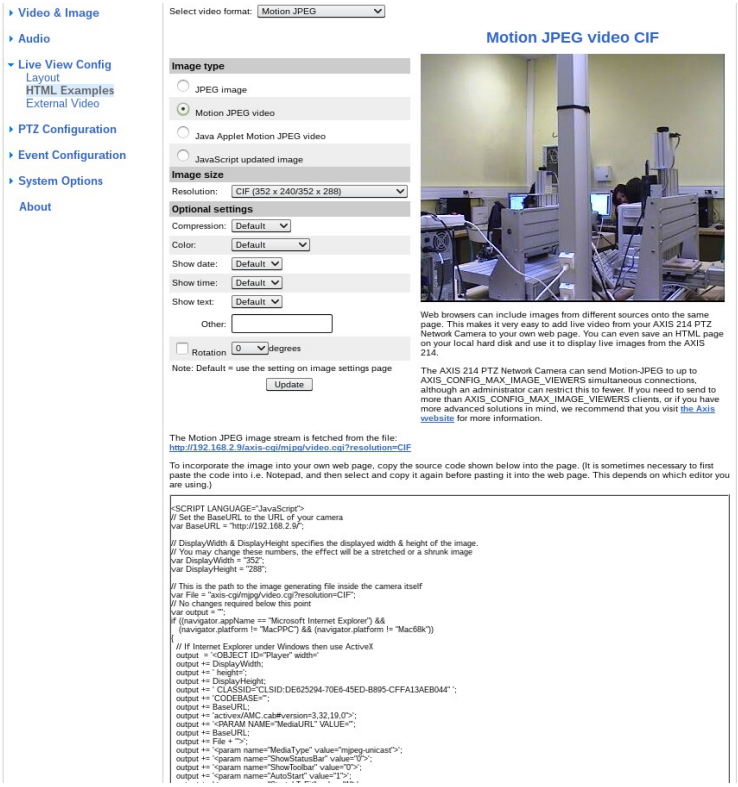
Comme vu lors de la revue de projet 1, des soucis notables avec la caméra persistaient, notamment celui qui m'empêchais de trouver son adresse IP et donc d'utiliser la caméra.
Ce problème a été résolu grâce à l'aide de mes professeurs qui m'ont informé que la caméra qui m'a été fournit en début de projet n'était pas celle qui aller être utilisée pour ce projet, il y a donc eut un changement de caméra IP. Ce changement m'a été fort utile car cette nouvelle caméra qui m'a été fournit fut beaucoup plus simple à manipuler, car elle possédait une documentation, ainsi j'ai put commencer le paramétrage de celle-ci.

Le paramétrage de la caméra fut donc assez simple, j'ai donc fait en sorte que la caméra possède une IP fixe et qu'elle ne demande pas de mot de passe pour se connecter car nous avions déjà assurer la sécurité grâce à la page d'identification présenté précédemment.

b.2- Intégration :

Une fois le paramétrage de la caméra terminé il a donc fallut l'intégrer au site afin d'afficher en direct les images que cette dernière filmait.
L'intégration du flux d'image de la caméra n'a pas été complexe mais seulement difficile à trouver.
Après avoir naviguer un moment dans les paramètres que proposait la caméra j'ai découvert cette page :

(voir page suivante)



La partie tout à gauche est le menu de navigation, ensuite l'image correspond à l'image de la caméra et le menu a gauche de l'image est ce qui nous intéresse.
Ce menu nous permet de choisir les réglages de l'image à afficher, la taille, la résolution, le format, etc.….
Une fois ce réglage effectué le cadre en bas de la page nous fournit un code JavaScript à incorporer directement dans notre code personnel (Annexe 16)

c- La suite :

Pour que le site web soit finalement complet il ne reste plus qu'à créer un journal d'événement pour y afficher, en temps réel, les actions du système.
Suite à cela il faudra résoudre les récents problèmes de la page de contacts que j'ai découvert lors de l'intégration de la caméra. Une fois ceci complété le site sera lui aussi complet.

3.2- Éclairage :

a- Pilotage des lumières:

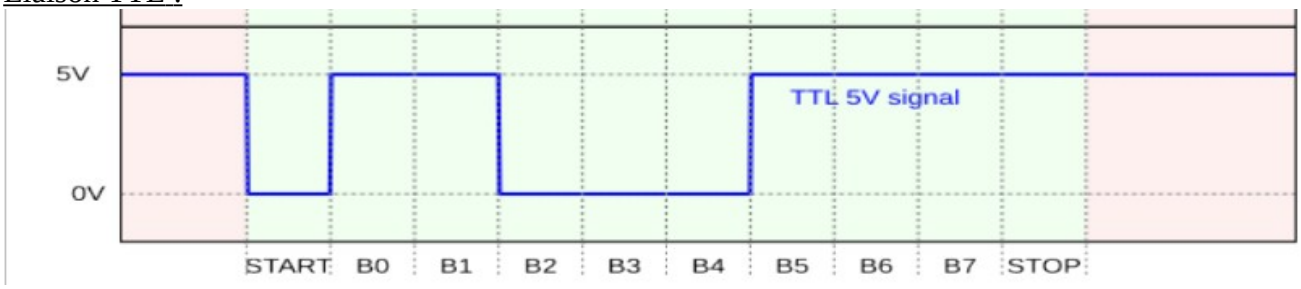
a.1- Introduction :

Selon le cahier des charges, le pilotage de l'éclairage doit se faire depuis le PC Serveur en communiquant par "Socket" à la Raspberry PI pour que cette dernière communique avec l'Arduino Uno via QSerialPort et puisse allumer les lumières. Ici on s'intéresse donc au pilotage des lumières à partir de la Raspberry PI donc à savoir, aller chercher les données envoyée par le PC Serveur, les interprétées ensuite établir un protocole de communication avec l'Arduino Uno pour finalement pouvoir envoyer des ordres à cette dernière.
Les parties qui vont suivre ne présenteront que partiellement le code fournit (seulement l'essentiel), les codes complet et commentés sont fourni avec le dossier,dans un fichier à part.

a.2- Fonctionnement Physique :

Donc pour que la liaison finale s'effectue, voyons d'abord l'USB de la Raspberry, la liaison au sein de l'USB, pour que les informations passent du microcontrôleur à USB, se fait par liaison série TTL (0V,+5V) jusqu'à l'adaptateur USB de la carte Raspberry. La liaison série TTL est une liaison série asynchrone basique et simple à mettre en place. Elle est utilisée dans notre cas, qui fait parti d'un cas général, à savoir, envoyer des données asynchrone en série.
Voici le signal d'une liaison TTL, sachant que "B0", "B1" sont des niveaux logiques 1, alors que "B2", "B3", et "B4" sont des niveaux logiques 0.

Liaison TTL :



Ensuite via un adaptateur "TTL-USB" intégré dans la Raspberry Pi la liaison continue par USB, sachant que cette dernière est une liaison série composée de 4 fils, dont ils possèdent au maximum une tension de 5V et un courant maximum de 500mA.
Pour ces 4 fils, le premier est l'alimentation (+5V) le second et le troisième servent pour le transfert de données (Data+ et Data-) et le dernier est la masse.

Numéro de broches	Couleurs des câbles	Fonction
1	Rouge	V_{BUS} (5 volts)
2	Blanc	D-
3	Vert	D+
4	Noir	Masse

Pour finir on passe par le dernier adaptateur, qui est l'adaptateur USB-I2C.
L'I2C fonctionne sur 3 fils, un fils de données (SDA), un fil d'horloge (SCL) et un fil de masse. L'I2C sur Arduino Uno fonctionne en 3.3V.

Et maintenant nous allons voir au niveau du code comment se passe cette liaison de l'USB à l'I2C de l'Arduino Uno.

a.3- Le logiciel :

a.3.1- Lecture de la base de données :

Afin que l'écriture se fasse nous avons besoin de savoir si l'on doit allumer ou éteindre les lumières.
Pour cela le PC Serveur va communiquer avec la base de données par Socket, et nous, nous devons donc lire dans la base de données ce qu'a écrit le PC Serveur.
Pour cela nous allons créer un "Thread", ce sera un code qui s'exécutera en tache de fond (qui n'interviendra pas directement sur l'interface), pour lire constamment dans la base de données les ordres à envoyer.

- Création du thread :

```
class lecture_bdd : public QThread
```

- Une fois le Thread créer on va devoir le lancer, un thread se lance grâce à une fonction nommée "start()". On va donc le lancer en même temps que le programme (dans le constructeur) :

```
QThread::start();    // On démarre le Thread en même temps que le programme
```

- Le Thread lancé la première chose qu'il doit faire c'est se connecter à la base de données pour, par la suite, y récupérer les données voulues :

```
db.setHostName("127.0.0.1");    // Serveur sur lequel la BDD est présente
db.setDatabaseName("Mari-Sec");  // Nom de la BDD
db.setUserName("marisec");       // Nom d'utilisateur avec lequel on se connecte
db.setPassword("marisec");       // Mot de passe de l'utilisateur en question
```

- Maintenant connecté à la base de données on crée une boucle ["while(1)"] dans laquelle on fait constamment la requête qui nous permet de récupérer l'état de la lumière et on l'envoie, pour que cette valeur soit écrite sur le port série :

```
QSqlQuery requete;   // Permet d'effectuer des requètes
int etat_lumiere_ext = requete.exec("SELECT etat_lumiere_exterieure FROM Lumiere ");   // Requete pour récupéré l'etat de la lumeire
Q_EMIT(sig_etat_lumiere_ext(etat_lumiere_ext));    // Envoie de l'etat de la lumiere pour l'ecrire sur le port série
```

Nous sommes désormais à un point assez avancé avant l'écriture sur le port série, maintenant que la valeur a été envoyé, on la récupère dans une autre fonction (via un connect). Cette fonction va nous permettre de construire et d'écrire dans un fichier texte la trame qui va devoir être envoyée sur le port série.

- Voici ce qui se passe dans notre fonction dans le cas où on doit éteindre la lumière :

```
case 0:     // Si "etat_lumiere_ext" est égal à 0
{
    if (!DataFile.open(QIODevice::ReadOnly | QIODevice::Text))   // Ouvre "data.txt" en lecture de type texte
    {
        msgBox.setText("Impossible d'ouvrir le fichier d'instruction");  // Afiche une pop-up avec le message d'erreur
        msgBox.exec();  // Execute/lance la pop-up
    }
    else
    {
        DataFile.resize(0);  //efface le contenu du fichier
        QTextStream texte (&DataFile);  // QTextStream va nous servir a ecrire dans le QFile
        texte << "531000";  // On ecrite la trame pour éteindre les lumieres
    }
} //Case 0
```

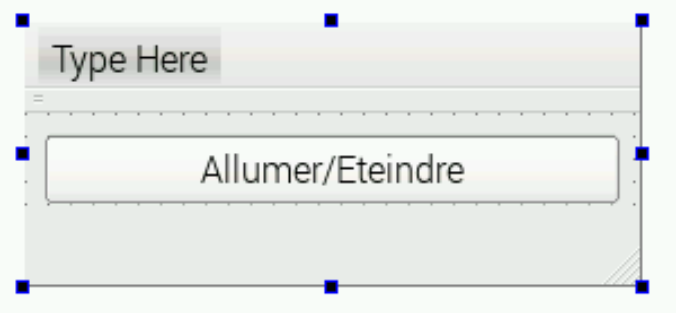
Donc on créer un fichier texte on écrit la trame d'extinction des lumière, sachant que "0x53" correspond au fait que l'on veut écrire sur la liaison série, "0x10" correspond à l'adresse de l'adaptateur et enfin "0x00" au code d'extinction des lumières.
- Si jamais l'ont doit allumer la lumière c'est sensiblement la même chose :

```
case 1:     // Si "etat_lumiere_ext" est égal a 1
    {

        if (!DataFile.open(QIODevice::ReadOnly | QIODevice::Text))   // Ouvre "data.txt" en lecture de type texte
        {
            msgBox.setText("Impossible d'ouvrir le fichier d'instruction");  // Afiche une pop-up avec le message d'erreur
            msgBox.exec();  // Exécute\lance la pop-up
        }
        else
        {
            DataFile.resize(0);  //efface le contenu du fichier
            QTextStream texte (&DataFile);  // QTextStream va nous servir a ecrire dans le QFile
            texte << "531001";  // On ecrite la trame pour eteindre les lumieres
        }

    } //Case 1
```

On observe ici qu'en cas d'erreur éventuelle une "pop-up" s'ouvre pour afficher l'erreur produite.
Maintenant que l'on peut lire les informations transmises par le PC Serveur, on va désormais tenter de communiquer avec l'adaptateur USB-I2C, pour pouvoir dialoguer avec l'Arduino Uno.

a.3.2- Communication série :

Donc nous allons voir comment a été fait le programme qui va nous permettre de communiquer avec l'Arduino Uno.
Pour commencer ce programme n'est pas censé avoir d'interface mais pour démontrer sont fonctionnement on a besoin de celle-ci, donc voici sont aspect (très basique) :



C'est un interface très simple qui grâce à un bouton nous permet d'écrire sur la liaison série. L'état du programme (message d'erreur, ou de succès) seront afficher dans la barre de statut.

- Tout d'abord, comme dit dans l'installation de QSerialPort, il nous faut inclure les bibliothèques de ce dernier afin de pouvoir utilisé ses fonctions, on inclut aussi la bibliothèque qui va nous permettre d'accéder aux informations qui sont récupérées par QSerialPort :

```
#include <QtSerialPort/QSerialPort>        //pour utiliser serialport
#include <QtSerialPort/QSerialPortInfo>    //Acces au infos recupérer par QSerialPortInfo
```

- Ensuite la première chose à faire est de récupérer les informations des ports disponibles et par conséquent la liste des ports qui sont disponibles :

```
QList<QSerialPortInfo> serialPortInfoList = QSerialPortInfo::availablePorts();
// les infos sur les ports disponibles contenues dans "QserialPortInfo" sont lister dans "serialPortInfoList"
```

Après sa, on doit définir comment les trames vont devoir transitées afin que l'adaptateur USB-I2C puisse "comprendre" ce que nous lui envoyons. On cherche donc dans la documentation de l'adaptateur (*http://www.robot-electronics.co.uk/htm/usb_i2c_tech.htm*) quel protocole de communication il suit, et en fonction nous entrons les paramètres avec QSerialPort afin de les établir de notre coté aussi :

```
serialPort.setBaudRate(QSerialPort::Baud19200);  // Vitesse de transmission
serialPort.setDataBits(QSerialPort::Data8);      // Nombre de bits de données
serialPort.setParity(QSerialPort::NoParity);     // Parité
serialPort.setStopBits(QSerialPort::TwoStop);    // Nombre de bits de parité
```

- Une fois les paramètres de communication établis, on doit désormais ouvrir, en mode écriture, le premier port disponible qui a été choisi automatiquement, grâce à cette fonction :

```
if (!serialPort.open(QIODevice::ReadWrite)) //ouvre le port serie en lecture/ecriture
```

Maintenant que la liaison a été établie, en considérant qu'il n'y a pas eut d'erreur on peut commencer à envoyer les données sur le port (nommé par la carte "ttyUSB"). Et pour faire cela on utilise le fichier texte vu précédemment qui possède la trame à envoyer à l'adaptateur, à savoir "0x53" pour l'écriture, "0x10" pour l'adresse de l'adaptateur et ensuite "0x00" pour éteindre les lumières ou "0x01" pour les allumées.

- Ici nous ouvrons le fichier texte nommé "data.txt" :

```
QFile DataFile("data.txt");    // Recupère le contenu du fichier "data.txt"
if (!DataFile.open(QIODevice::ReadOnly))   // On ouvre le fichier en testant si sa a marché ou non
```

- Et ensuite nous pouvons lire ce qu'il y a à l'intérieur :

```
QString writeData(DataFile.readAll());   // Recupere les données du fichier en QString
DataFile.close();    // On ferme le fichier
```

- Étant donné qu'il n'y a pas d'erreur on peut continuer. Maintenant pour que cette trame soit correctement lue et interprétée par l'adaptateur il va falloir la convertir sous le bon format à savoir en octets, voici comment nous faisons :

```
QByteArray trame(qPrintable(writeData));    // Convertion de QString en octets qui porront etre envoyé
qint64 Written = serialPort.write(QByteArray::fromHex(trame));         // Ecriture des données / envoie des donées
// On met le résultat de la fonction "write" dans Written pour pouvoir verifier par la suite si elle a reussie
```

Après avoir fait diverses vérification de l'intégrité de la trame on définit un temps limite à ne pas dépasser pour l'envoi de cette trame, si le temps est trop long on considère que l'envoi a échoué :

```
else if (!serialPort.waitForBytesWritten(5000))  // Temps max d'attente pour l'envoie de données
```

On arrive à la fin, si tout s'est déroulé normalement, sans erreurs, devrait s'exécuter cette ligne qui affiche dans la barre de statut que l'envoi a réussi :

```
cout = QObject::tr("\n Donnees correctement ecrite sur %1 \n").arg(PortName);
Q_EMIT(sig_afficher_cout(cout));
```

a.4- Conclusion :

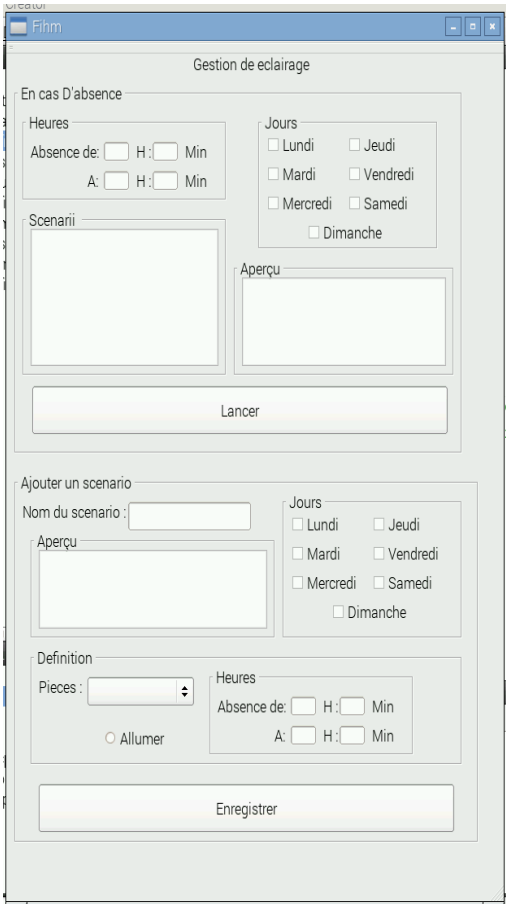
Les tests on été effectués en branchant l'Arduino Uno et la Raspberry Pi à l'adaptateur USB-I2C, tout fonctionne parfaitement (en liaison asynchrone). A ce stade le programme lit dans la base de données mais ne peut récupérer l'état de la lumière (la table n'existe pas encore). Le programme est opérationnel, mon collège PUG Alexandre peut en attester car nous avons effectué les test en étroite collaboration (les tests ont été effectué en simulant l'écriture de la base de données).

b- Interface Homme-Machine :

b.1- Introduction :

Dans cette partie nous allons créer une IHM (Interface Homme-machine) qui va nous permettre de gérer les scénarii, afin d'automatiser l'éclairage de la maison en cas d'absence de propriétaires.
A terme cette IHM devra être composé, en arrière plan, du programme de pilotage d'éclairage et du programme de communication Socket (PC Serveur - Raspberry) en intégrant ces deux programmes en un seul nous créons un logiciel qui gère seul l'éclairage, composé d'une seule interface.

b.2- Objectifs :



Voici à quoi devrait ressembler l'IHM(si elle ne change pas).

Cette Application va permettre, une fois terminer, à l'utilisateur d'inscrire l'heure à laquelle il part de chez lui (donc l'heure à laquelle le scénario de présence va fonctionner), l'heure à laquelle il estime son retour et de cocher le ou les jours pendant lesquels il s'absente.
Une fois ces champs renseignés plusieurs scénarii s'affichent, l'utilisateur en sélectionne un au choix. Une fois un scénario sélectionné un aperçu apparaît donc, et si le scénario lui convient il lui suffit de cliquer sur le bouton "Lancer", pour mettre en route la simulation.
La seconde partie permet d'ajouter un scénario à la base de données. On nomme le scénario, on le définit sur un ou plusieurs jours, on inscrit les heures de fonctionnement, on choisi dans quelle pièce il va être effectif, en sélectionnant "allumer". On clique ensuite sur "Enregistrer", et il suffit de recommencer pour chaque pièce que l'on veut éclairer à certaines heures, un aperçu apparaîtra au fur et à mesure des enregistrements.
Tout ceci est le fonctionnement futur de l'application car seule l'interface a été faite, cette application n'est pas encore fonctionnelle.

3.3- Caméra :

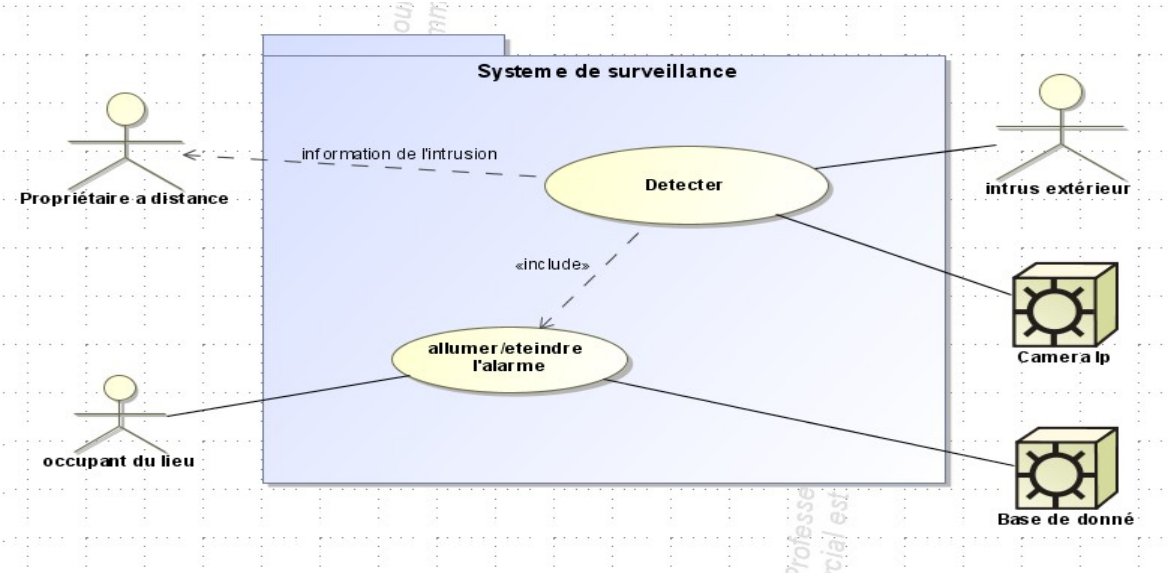
a- Introduction :

Maintenant que le problème de la caméra à été résolu et qu'ainsi nous possédons une nouvelle caméra paramétrée, comme expliqué dans la partie site web plus haut.
Désormais nous allons donc fournir un programme qui va se charger de détecter les mouvements filmés par la caméra.

Pour répondre à cette problématique, le cahier des charges ne nous donne aucune contraintes, de ce fait je choisi de faire un programme en JAVA pour une raison, un programme en JAVA est un programme assez pratique, il ne demande pas beaucoup de ressources machine, ce qui est un aspect non négligeable car ce programme fonctionneras non stop, pour assuré la sécurité de la maison.

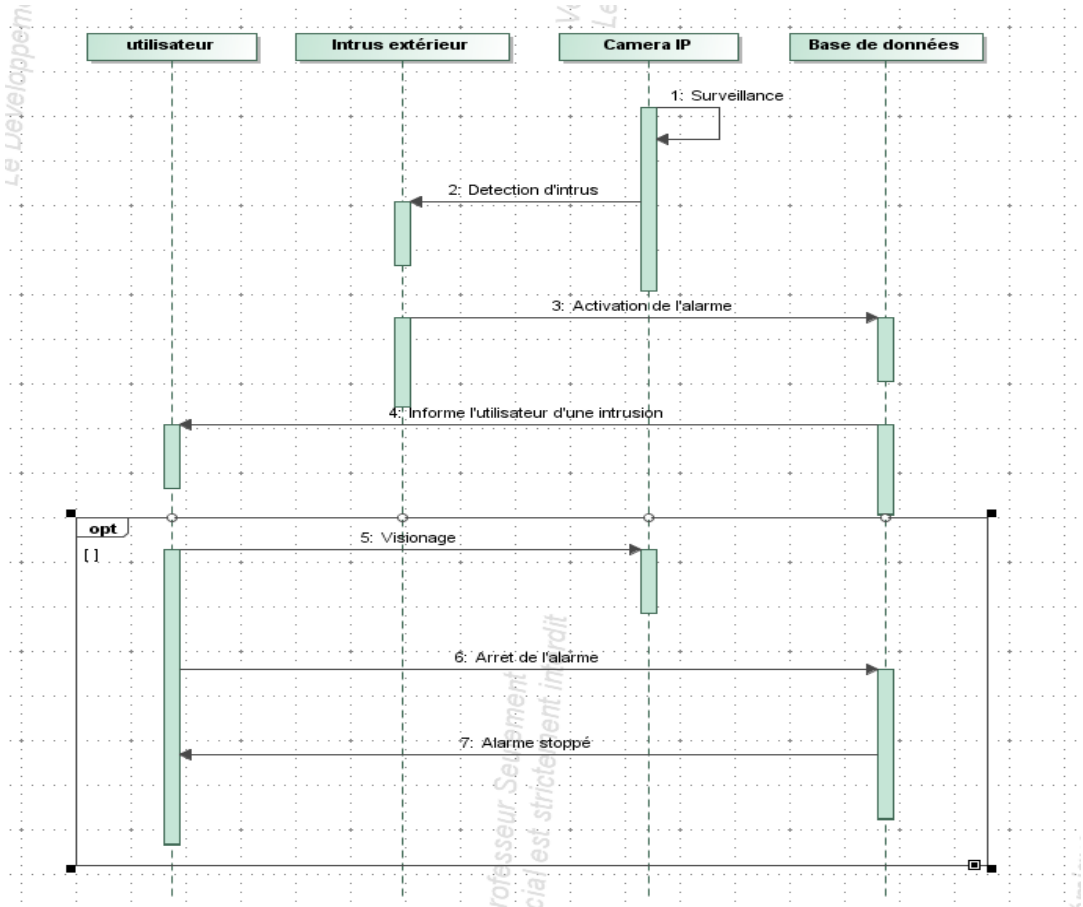
Ici je vais donc vous présenter, via plusieurs diagrammes SYSML, mon analyse du cahier des charges concernant la place du détecteur de mouvement dans le système.

USD – Diagramme de cas d'utilisation - Détection d'intrus :



Ce diagramme de cas d'utilisation nous permet de voir parfaitement comment doit fonctionner le programme de détection de mouvement de la camera IP par rapport au système dans son ensemble.

OSD-S – Diagramme de séquence - Détection d'intrus :



Ce digramme SYSML de séquence nous permet de visualiser le déroulement des actions pour la caméra et ses périphériques lors d'une détection d'intrus extra-muros.

b- Caractéristiques de la caméra :

Cette caméra possède une résolution pouvant aller de 160x120 pixels à 704x576 pixels. Elle possède une fréquence de 25 images par seconde.

De plus le flux de données vidéo qui transite par la caméra ce fait par "MOTION-JPEG" c'est un codec vidéo qui permet de compresser les images une à une, en JPEG (codec : circuit électronique capable de compresser un signal numérique).
Pour finir elle possède une connexion RJ-45, pour se connecter au réseau, ainsi qu'une alimentation de 13V, pour une puissance de 14W (soit environ un peu moins de 108mA d'intensité).

c- Détection de mouvement :

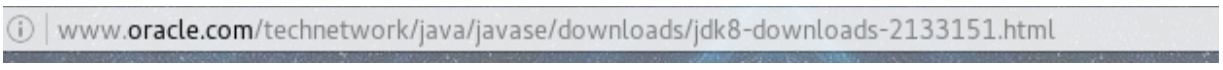
c.1- Introduction :

Tout d'abord ma première solution fut d'utilisé Python afin de gérer la détection de mouvement car c'est un langage simple à coder, néanmoins après avoir effectué toutes les installation et codé le programme afin qu'il affiche l'image de la caméra, il n'affichait rien car la connexion ne se faisait point. De ce fait, après avoir planché sur le problème un certains temps j'ai dû trouver une autre solution pour tenter de respecter au mieux les délais. J'ai donc choisi la solution de JAVA, bien que la solution de Python fut entièrement déployé (mais non fonctionnelle), et que le JAVA soit un langage un peu plus complexe que le Python, il y aurait une possibilité, à terme, d'afficher cette application directement sur le site.

Donc pour pouvoir gérer le traitement des images et des vidéos avec JAVA on va utilisé une bibliothèque qui se nomme "OpenCV", qui est une bibliothèque spécialisée dans le traitement graphique. Ensuite nous nous servirons de "ECLIPSE" qui est un environnement de développement pour JAVA.
Cependant la première étape est d'installer JAVA sur Windows.

c.2- Installation de JAVA et ECLIPSE :

- Donc en premier lieu pour procéder à l'installation de JAVA Développement il faut le télécharger, le téléchargement se fait à cette adresse :



- Direction la page de téléchargement :

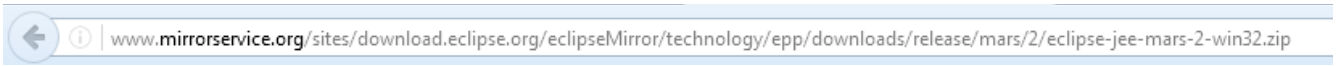
- Et nous, nous choisissons la version pour Windows 32bits :

Windows x86	182.11 MB	jdk-8u91-windows-i586.exe

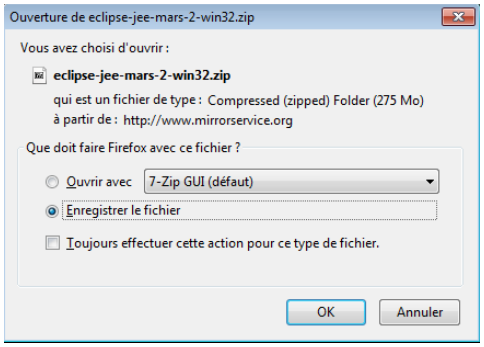
Ici aussi pour le reste de l'installation il suffit simplement de suivre l'assistant d'installation, qui devrait se dérouler sans problème.

<u>Viens maintenant l'installation de ECLIPSE :</u>

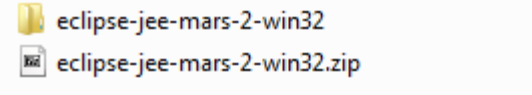
- On se rend donc a cette adresse afin de télécharger l'archive compressée du programme :



- On enregistre ensuite cette dernière dans un dossier :



- On décompresse ensuite l'archive afin d'obtenir un dossier à coté de l'archive, comme suit :



- En ouvrant ce dossier on peut lancer directement ECLIPSE :

Ici pas besoin d'installation, ce dossier contient tout les fichiers nécessaires afin de faire fonctionner ce logiciel, cela permet un certaine portabilité du programme (solution non envisagé dans ce projet car mal organisé et difficile à gérer lorsqu'il y a plusieurs programme, comme c'est le cas dans notre situation).

c.3- Installation de OpenCV :

- Maintenant que JAVA est prêt à accueillir OpenCV nous pouvons le télécharger :

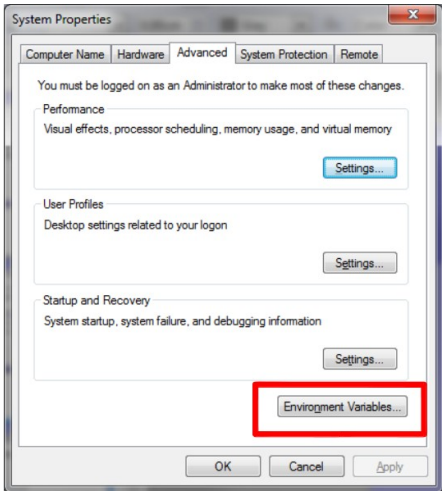
- Ensuite on exécute OpenCV, et on suit les étapes d'installation, j'ai installer directement ce dernier sur le bureau pour une certaine facilité au niveau des chemins d'accès.
- Après on peut télécharger à cette adresse le "CMake", qui est ce que l'on appelle un "moteur de production", il va nous permettre de construire et configurer OpenCV :

www.cmake.org/files/v2.8/cmake-2.8.11.2-win32-x86.exe

- CMake, pour construire une bibliothèque, à besoin de compilateur en C et C++, c'est pour cela que l'on va télécharger "MinGW" qui est un compilateur Windows basique de C et C++ :

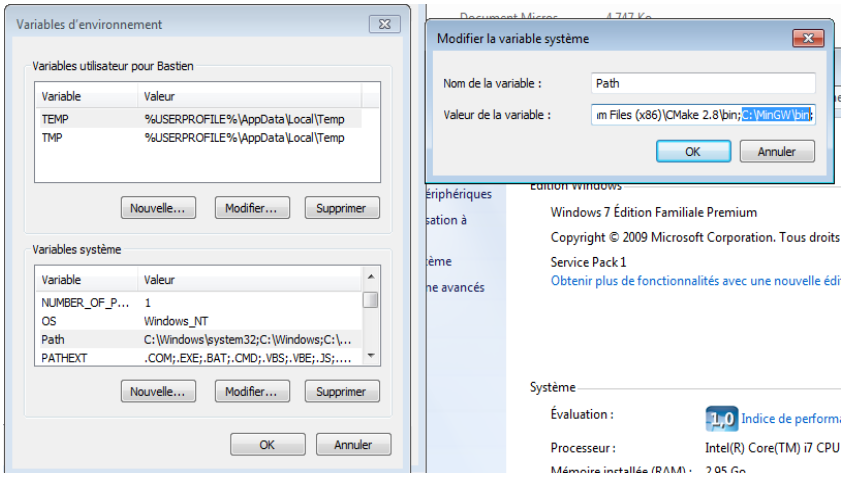
Même procédure on suit l'assistant d'installation, pour une installation basique.

- Ensuite on doit ajouter ce compilateur au système Windows pour qu'il puisse s'exécuter avec le CMake, on place alors le chemin de MinGW dans le "PATH" de Windows (PATH : emplacement des variables systèmes).
Pour cela on accède au "Panneau de configuration" puis dans "Système" et Options Avancées" pour arriver ici :

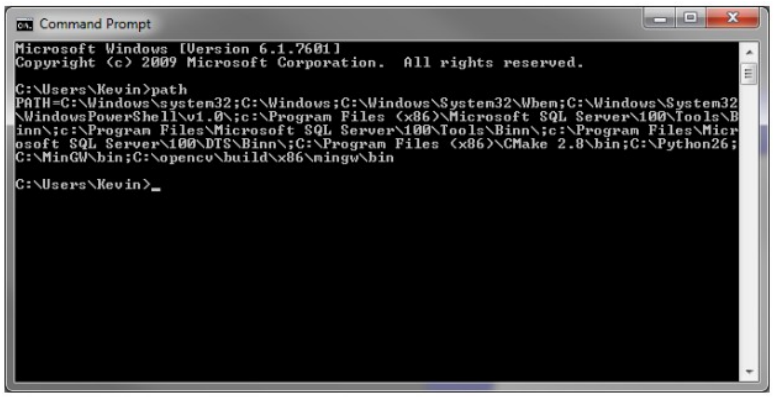


→ Pour accéder aux variables systèmes.

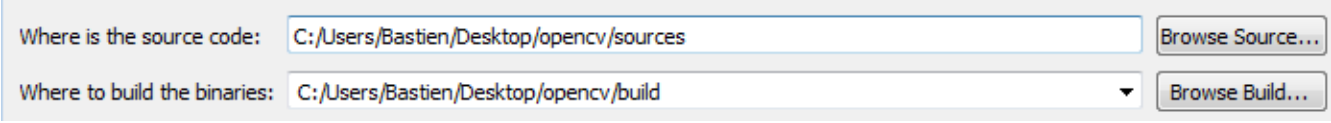
- On entre alors le chemin d'accès a MinGW :



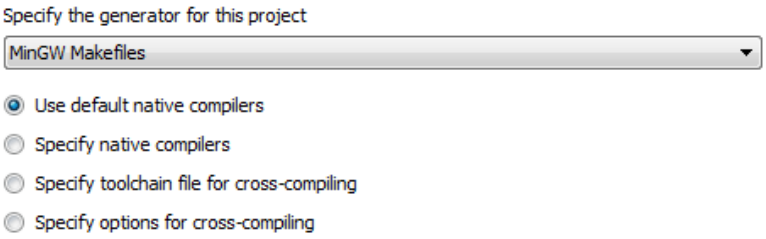
- On vérifie, après avoir redémarrer la machine, via la commande "path" si notre chemin a été ajouté :



- Une fois terminé, on peut maintenant lancer l'interface graphique du CMake, et entrer la localisation du fichier OpenCV à traiter :



- Ensuite on clique sur le bouton "Configurer" et nous apparaît un boite de dialogue qui nous demande d'indiquer les compilateurs (C et C++) que l'on veut utiliser :



On indique donc MinGW et on coche la première case, car ce compilateur fait maintenant par des variables systèmes de Windows.
- Une fois l'opération terminée on clique sur le bouton "Générer".

- Apparaît alors dans notre dossier OpenCV un fichier nommé "MakeFile", il faut donc se rendre dans ce dossier via ligne de commande et inscrire la commande suivante :

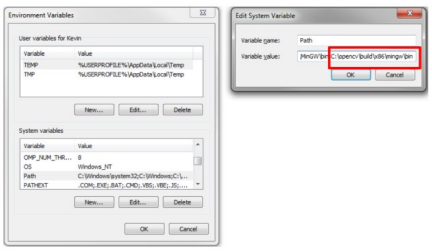
→ ***mingw32-make***

-Et ensuite la commande :

→ ***mingw32-make install***

Ces deux commandes permettent d'exécuter les "fichiers actions" (MakeFiles) créer par le CMake afin de compiler et construire la bibliothèque OpenCV.

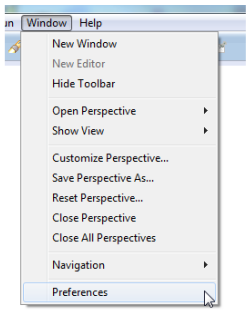
- Maintenant il faut ajouter le compilateur de OpenCV que l'on a crée, dans les variables systèmes de Windows, avec la même procédure que vu précédemment :



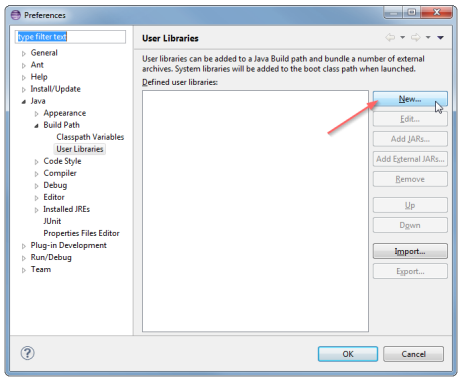
Ainsi tout est opérationnel.

Maintenant on peut ajouter la librairie OpenCV à ECLIPSE afin de pouvoir coder une application JAVA.

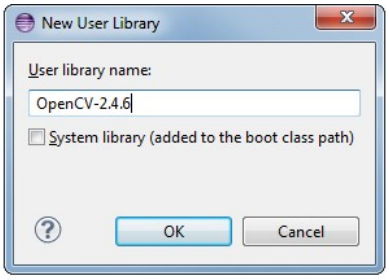
- La première étape est de lancer ECLIPSE est de se rendre dans "Windows" et "Préférences" :



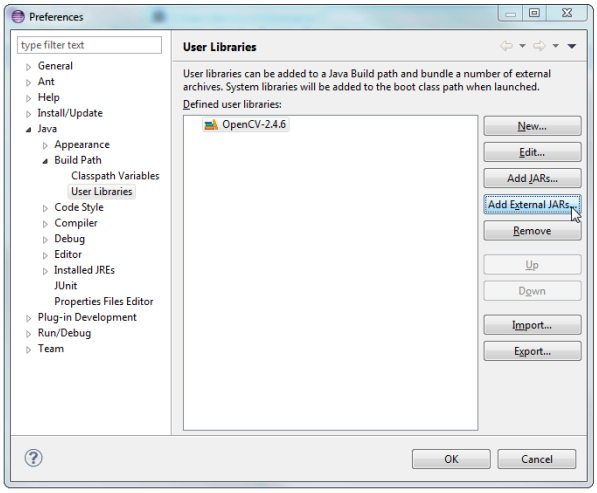
- Ensuite dans l'arborescence on se rend dans "JAVA" puis "Users Library" puis "nouveau" :



- On nomme cette nouvelle librairie :



- Ensuite on va récupérer le chemin de la librairie :

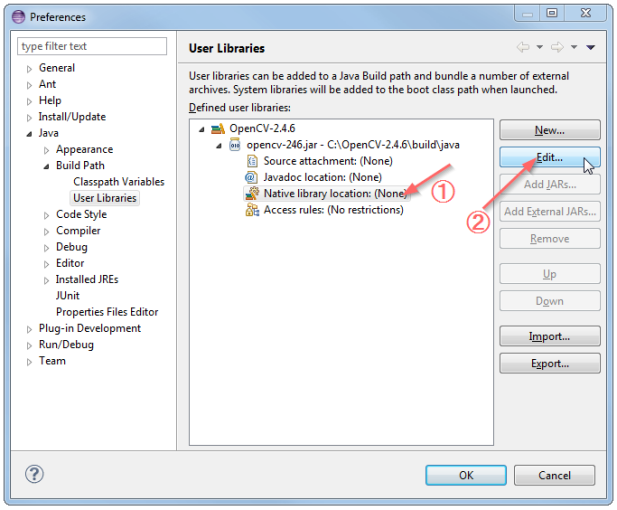


- On se rend donc ensuite dans :

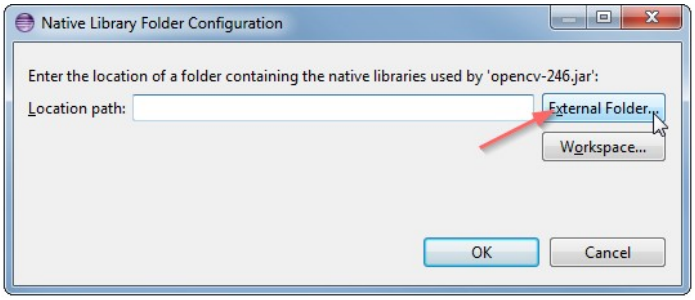
→ ***C:\OpenCV-2.4.6\build\java***

Puis on sélectionne le fichier nommé "opencv-246.jar"

- Ensuite va indiquer le chemin de sa librairie de base :

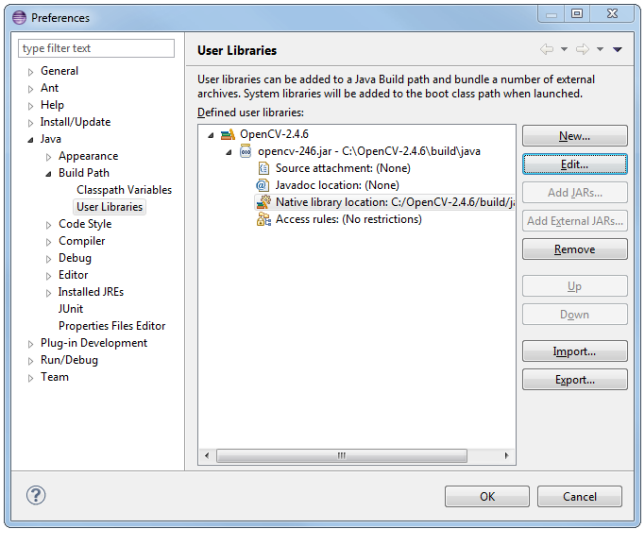


- On indique donc le chemin dans "dossier externe" :

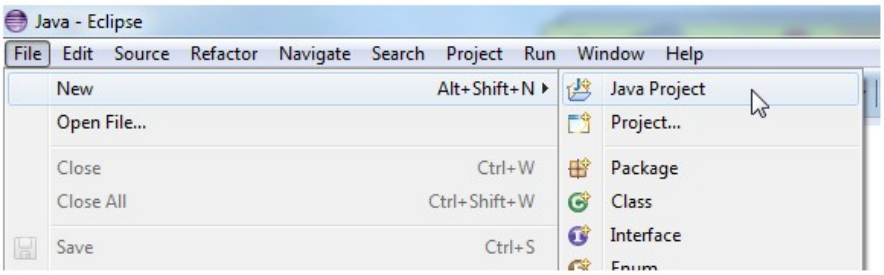


Pour nous le chemin se situe ici :
→ ***C:\OpenCV-2.4.6\build\java\x86***

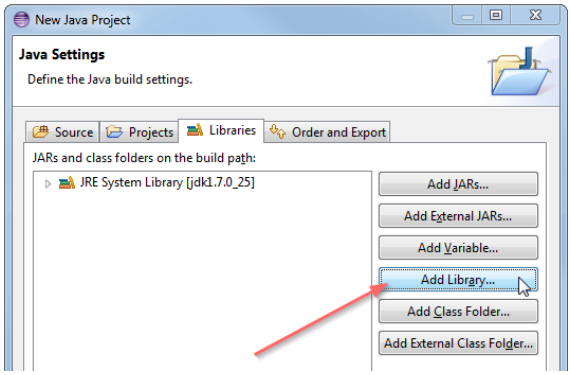
Et voici le résultat final :



- Pour finir lorsque l'on crée un nouveau projet JAVA :



- Lorsque l'on définit les paramètres de notre projet il suffit d'ajouter notre librairie au projet :



Nous pouvons désormais utiliser les librairies OpenCV avec notre projet JAVA, sans problèmes.

c.4- Explications :

c.4.1- Le programme :

A ce stade, le programme JAVA que l'on à crée, est un "Applet" c'est-à-dire que c'est une application web, une application qui peut être exécuter avec un navigateur Internet.
Pour le moment le code ne se compose seulement des fonctions permettant de récupérer et d'afficher un flux vidéo (le code est visible en annexe 17). Tout comme le programme Python effectué avant celui-ci le code est bon (sans erreur) mais rien ne fonctionne, sans que je sache encore pourquoi.
Je suis actuellement à la recherche du problème, deux solutions s'offrent à moi :
La première étant celle où je trouve une solution est où je peut continuer mon code en JAVA.
La seconde étant celle où je décide de trouver un autre moyen pour accéder au flux de la camera, comme je l'ai fait en passant d'une solution sous Python à une solution sous JAVA.

c.4.2- L'exécutable :

Ici je vais vous présenté la solution sur laquelle j'ai travaillé lors du développement du programme sous Python, cette solution permet de rendre un programme Python exécutable.

Afin de rendre ce programme exécutable comme tout programme, pour évité au client d'avoir a se servir de Python, car tout le monde ne sait pas forcément s'en servir.
On va donc créer un exécutable grâce à une librairie Python fournie lors du téléchargement de Python, elle se nomme "Py2exe".

- Pour utilisé cette librairie nous allons devoir créer un programme Python relativement simple, pour cela on se rend dans le dossier du programme Python et va créer un dossier que l'on nommeras "Executions" :
  - → ***C:\Python27\Executions***

- Ensuite nous avons besoins du "Shebang" (fichier d'en-tête) afin que le programme sache quel interpréteur utiliser et quel encodage :
  - → ***#! /usr/bin/python***
  - → ***# -*- coding: utf-8 -*-***

- Ensuite on importe "Py2exe" ainsi que la librairie à laquelle elle-même est rattachée :
  - → ***import py2exe***
  - → ***from distutils.core import setup***

- Maintenant on peut coder ce qu'il va se passer à l'appel du programme :

```
→ 1. setup(
  2.     name="Nom_du_programme",
  3.     version="1.300",
  4.     description="description_du_programme",
  5.     author="Auteur_du_programme",
  6.     license="2.03",
  7.     url="http://python.jpweb.com",
  8.     zipfile=None,
  9.     windows=[{
 10.         "script": "nom_du.py"}]
 11.     )
```

Donc le fonctionnement est très simple, la ligne 10 est le chemin vers le programme Python que l'on veut rendre exécutable, la ligne 9 permet de rendre l'exécutable, un programme avec une fenêtre d'exécution comme bien connu. Ensuite les lignes 1 à 7 permettent de définir le programme, lui donner un nom, des informations, etc.… La ligne 8 quand à elle définit comment nous allons récupérer le programme exécutable à la fin du processus, ainsi que ses dépendances,("zipfile=None") ce programme sera mit dans un dossier qui sera créer et non dans un fichier Zip.

- Désormais nous possédons notre propre programme Python pour créer des exécutable personnalisé, il nous suffit maintenant, dans le programme Python que l'on va créer d'ajouter ces en-têtes afin d'éviter les erreurs :

```
→ import sys
→ import os
→ import imps
```

Ce sont les fichiers qui permettent au programme de gérer son futur environnement, car son environnement sera une fenêtre Windows.

- Pour finir on va utiliser notre programme "setup" pour rendre un programme Python exécutable, grâce à cette commande :

```
→ C:\Python27\Executions> C:\Python27\python.exe setup.py py2exe
```

Cette commande va chercher notre programme "setup" pour l'exécuter avec "Py2exe" et notre programme "setup" se charge d'aller chercher notre programme Python avec le chemin qu'on lui a donné dans la fonction "script".

Résultat :
Les tests qui ont été effectué lors de l'écriture de ce "tutoriel" se sont parfaitement déroulés, les seules erreurs qui ont été rencontrée sont celle des fichiers d'en-tête a placé dans le programme Python à rendre exécutable, ainsi que les commandes Windows à cause de l'habitude de travail sous Linux.
Bien que non utilisé, j'ai présenté cette solution car elle m'a prise énormément de temps, le programme Python pour la camera est en annexe 18.

d- Suite :

Étant donné été informé qu'il serait, dans la pratique, plus évident si les logiciels fonctionnant sous Windows étaient transportables. De ce fait une solution à été trouvée afin de rendre tout ces programme déplaçable d'une machine Windows à une autre. La solution est un logiciel de création d'installateur nommé "InnoSetup". "InnoSetup" serviras donc seulement si le projet arrive à terme, afin de pouvoir avoir un installateur du système que l'on pourra installer chez le client.

Pour la suite du projet pour qu'il soit complètement opérationnel dans ma partie, il me reste à résoudre les problèmes actuels énoncés ainsi que trouver une solution afin de pouvoir enregistrer une vidéo dans la base de données lorsqu'un mouvement est détecté.

De plus je doit trouver comment il serait possible de piloter la caméra à distance, depuis le site ou même depuis le PC Serveur, quoi qu'il en soit cette partie pilotage ne sera abordé que seulement en dernier, car c'est une partie optionnelle, en effet lors de la surveillance on place la caméra dans un grand angle, pour surveiller un espace donné, donc le déplacement de celle-ci n'est pas une fonction vitale.

4- Synthèse des prévisions

Grâce aux GANTT effectués (GANTT prévisionnel et GANTT réel) qui sont fournis en annexe 8 et 9.
On remarque donc que toutes les prévisions n'ont pas été suivies, en effet lors de la construction du GANTT prévisionnel nous ne connaissions que très peu le travail qui nous attendaient, bien que nous avons essayé de prévoir les erreurs éventuelles il était peu probable que les prévisions soit exactes.
Néanmoins les prévisions que nous avons faites nous permettes de nous situer dans notre avancement actuel par rapport a l'avancement idéal.
Je vous invite donc à comparé les annexes 8 et 9, pour le constater par vous-même.

5- Conclusion Finale

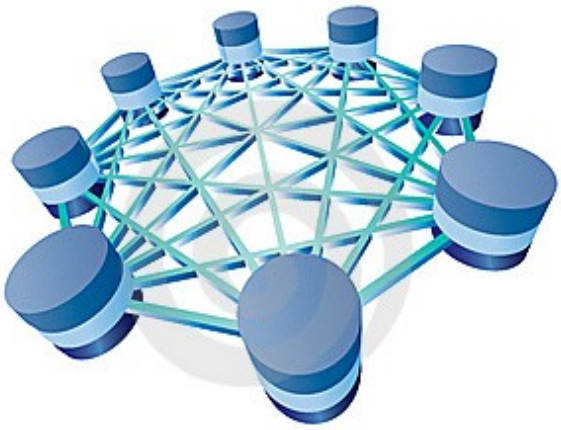
Pour conclure sur ce projet, je pense qu'en premier lieu avec plus de temps j'aurais eu l'occasion de mieux finir les détails, les optimisations, les documentations nécessaires, en soit, l'amélioration globale du projet. Bien que nous possédions seulement 200h de travail, ce projet fut tout de même enrichissant, au niveau personnel, car il m'a permis avant tout de connaître les besoins réels d'une entreprise et de savoir comment y répondre.

J'ai par la même occasion apprit à travailler en autonomie et à lier mon travail au sein d'un groupe, ainsi qu'à être organisé tant bien dans la gestion des informations que je devais gérer, que dans le code que j'ai du fournir.

Bien que le travail fournit dans ce projet a majoritairement été des solutions, qui au final ont été abandonnée car non adapté à la situation ou trop longue, j'ai appris qu'il fallait se tromper pour pouvoir réussir.

Ce projet resteras donc une expérience utile pour la suite de mes études.

ROBIN Cédric



Sommaire :

Partie A :

Introduction et mise en place la base de données……....p.75

QtCreator et Alarme virtuelle……..p.79

Le site web……...p.81

Partie B : Mes activités

Alarme……p.83

base de données……...p.89

Partie C : Physique

Protocole Ethernet……....p.90

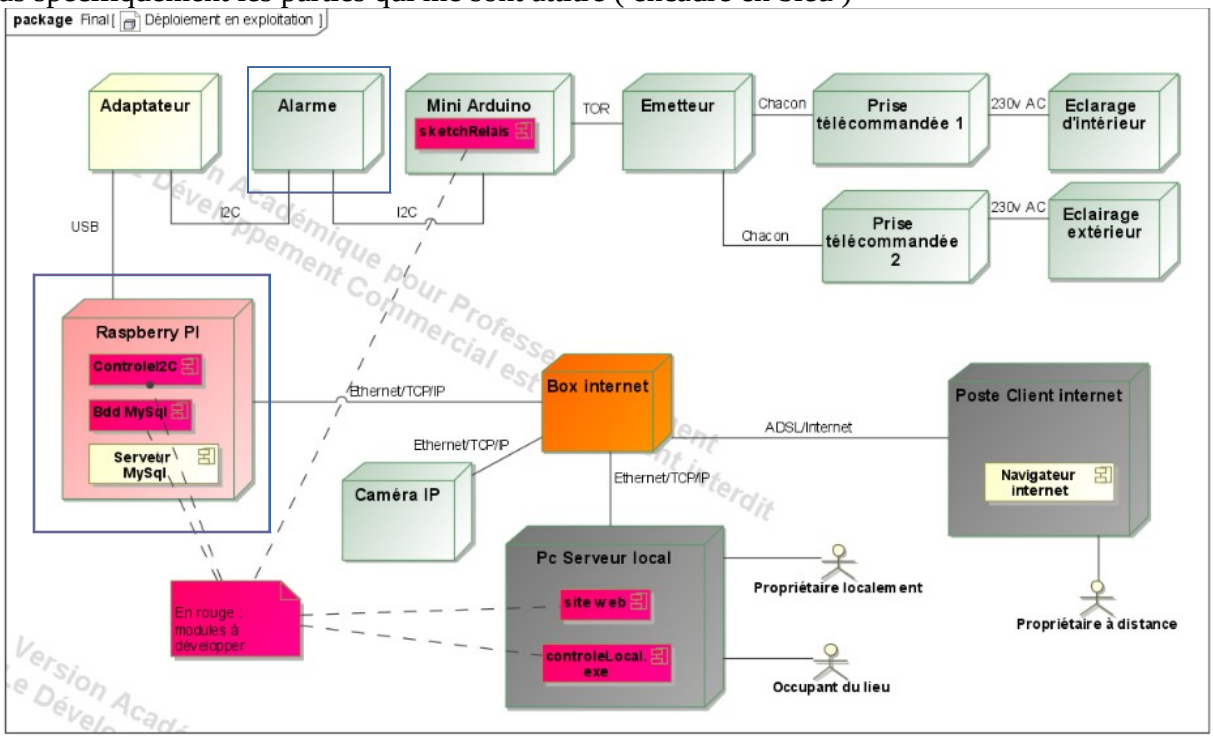
Annexes :

journal de bord

A- Introduction et mises en place

Mes taches principales sont de créer une base de donnée et une alarme virtuelle .
L'alarme sera simulé par un logiciel QT , qui devra émettre un son en cas d'intrusions tout en relayant les informations à la base de données et pourrons être visible depuis le site web .

Diagramme de déploiement du projet global :
et plus spécifiquement les parties qui me sont attitré (encadré en bleu)



a)La base de données

La base de données situé sur la carte raspberry Pi est administré via interface graphique grâce à phpMyAdmin . Il m'a fallut pour cela installer au par avant un serveur lamp sur la carte (qui servira également pour l'hébergement du site web)et pour ce faire j'ai utilisé les commandes :

```
sudo apt-get update && sudo apt-get -y upgrade && sudo apt-get -y dist-upgrade
```

qui a pour objectif la mise à jour de la carte raspberry et des logiciel présent sur la carte,

```
sudo apt-get install apache2 php5 mysql-server phpmyadmin
```

cette commande a pour but d'installer en une seule operation apache2 ,php5 , le serveur mysql et phpmyadmin .

Une fois l'installation terminé et la carte redémarré il faut vérifier que le serveur SQL est bien en route , pour cela j'utilise la commande :

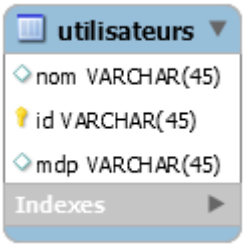
```
mysqladmin -u root -p status
```

Une fois le serveur fonctionnel il est temps d'administrer la base de données pour simplifier l'opération j'ai choisis d'utiliser PhpMyAdmin :

Le nom d'utilisateur utilisé pour acceder à la base de données est : Marisec .
Le mot de passe utilisé pour acceder à la base de données et : marisec .

La base de données à pour but de répertorier tout ce qui ce passe sur le système de surveillance (intrusion , activation et désactivation de l'alarme , mode utilisé …)

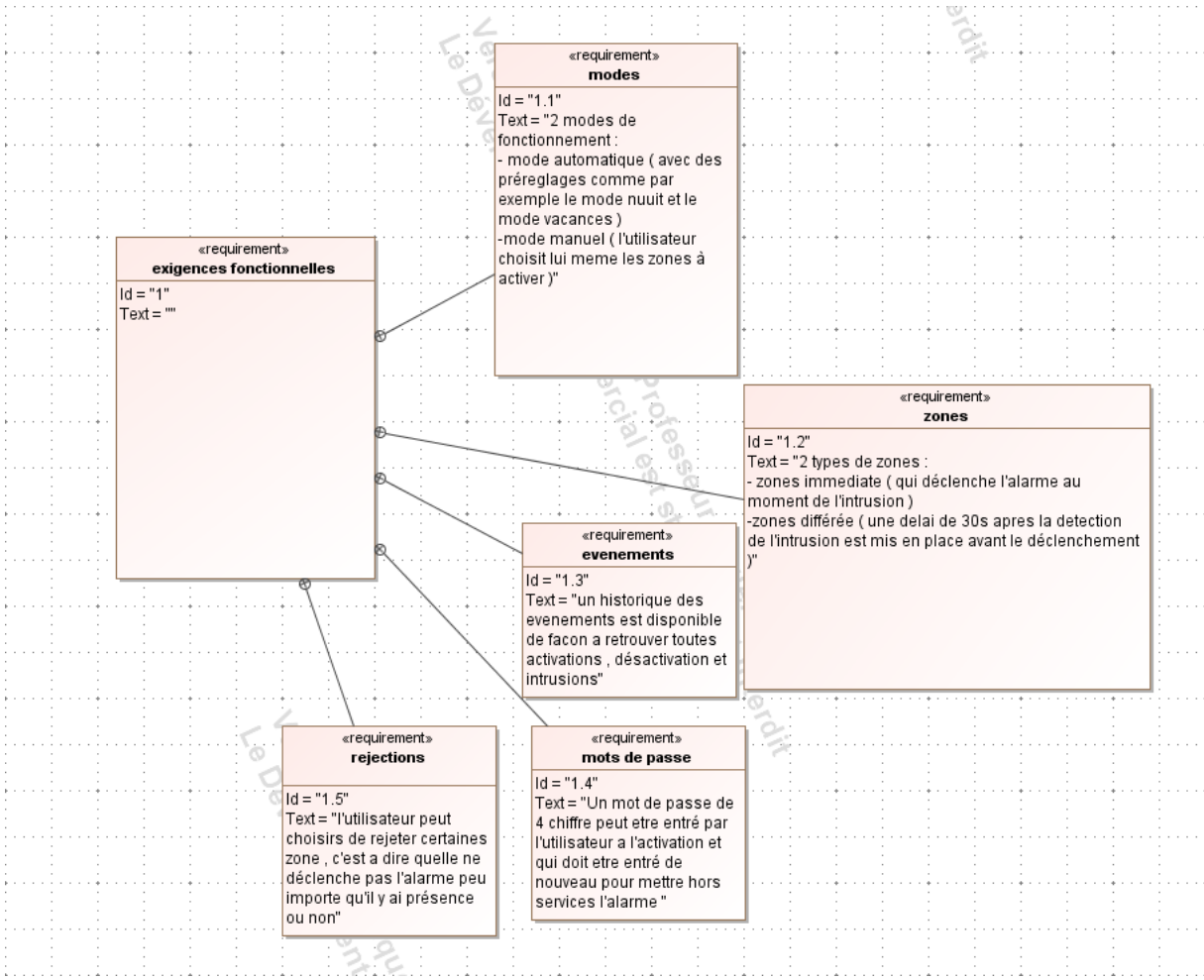
Ainsi que la liste des utilisateurs autorisé à ce connecter au site pour sécuriser l'accès à l'alarme et au paramètres .



b)QtCreator et Alarme virtuelle

Pour le développement il m'a tout d'abord fallut mettre en avant les fonctionnalités d'une alarme physique afin de les reproduires au plus précisément dans une alarme virtuelle .

Diagramme des exigences de l'alarme :
(voir page suivante)



Tout comme une alarme physique , notre alarme virtuelle sera capable de gérer plusieurs zones de déclenchement (qu'elle soit immédiates ou bien différée) mais égalements de ne pas prendre en compte certaines zones .
L'utilisateur est également capable d'entrer un mot de passe de son choix .
Il est aussi nécessaire d'offrir au utilisateurs un journal des évènements .

Pour réaliser ce projet j'ai donc choisit d'utiliser le logiciel QtCreator :
- d'une part pour cela portabilité multi plateforme (pour mon développement j'utilise un pc sur windows , tandis que le projet final est hébergé sur une raspberry utilisant un os rapsbian)
-d'autres part car j'ai eu une bonne connaissance de cet outils de développement .

Il a donc fallut installer QtCreator sur la Raspberry :

sudo apt-get install qt4-dev-tools
ainsi que :
sudo apt-get install qtcreator
La raspberry est donc fin prete .

c)Le site web

Dans le cadre de ma partie , le site web va avoir pour but de p que d'afficher un journal des événements :
ce journal fournira un affichage de toutes les activités de l'alarme depuis la dernière lecture du journal .
Il y aura également possibilité de voir les enregistrement plus ancien .

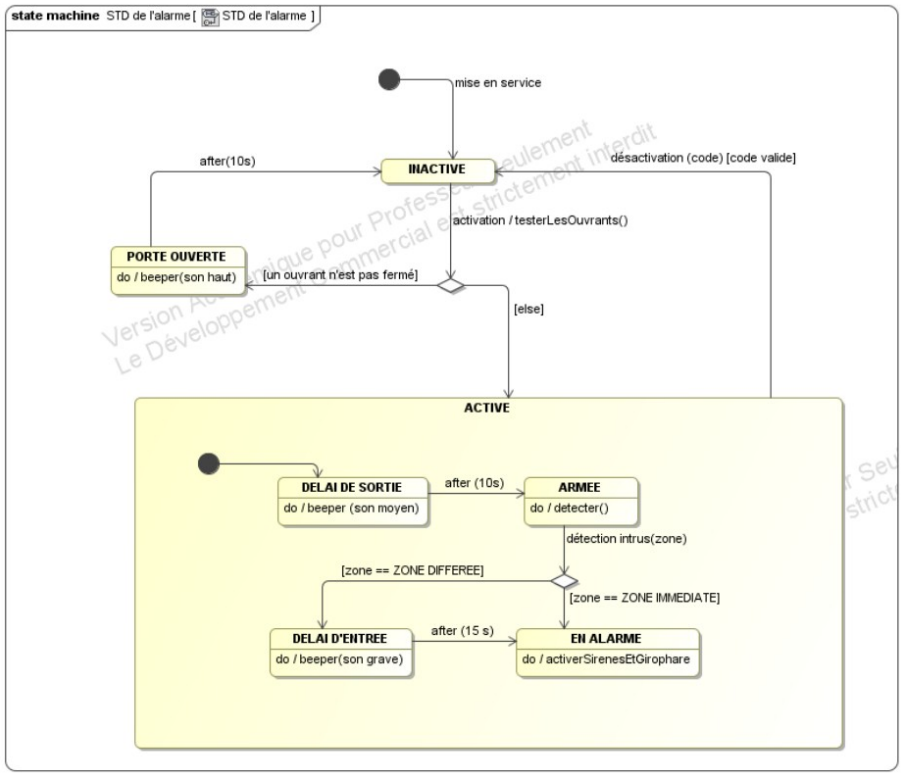
C'est donc à cet effet qu'une table « journal » et « archive » sont présentes dans la base , de façon à faciliter la récupération des données .
Le site web servira d'intermédiaire entre la base de données , l'alarme , la caméra et l'utilisateur .

B)Mes activités

Mes taches dans le projet Mari-Sec sont d'une part la gestion de l'alarme ainsi que la mise en place et la gestion de la base de données .

L'alarme Mari-Sec est représenté par un programme sous QT son rôle est , en cas d'intrusion de faire remonter l'information à la base de données hébergé sur la raspberry . L'alarme peut également etre configuré en mode automatique ou manuel .
La base de données quand à elle devra stocker les informations de l'alarme mais également celle de la gestion des lumières .

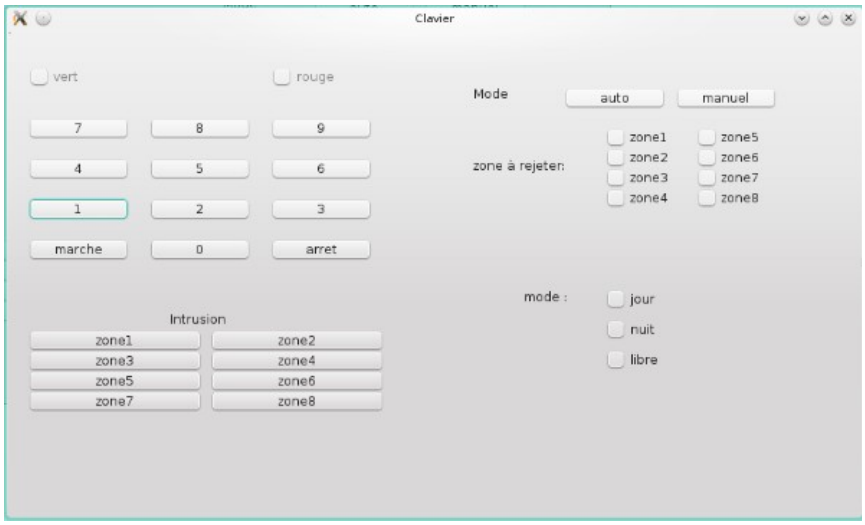
Le site web quand à lui a pour but d'offrir une vision globale au utilisateurs :
Un journal d'évènements présentant les dernières informations relatives à l'alarme



a) Alarme :

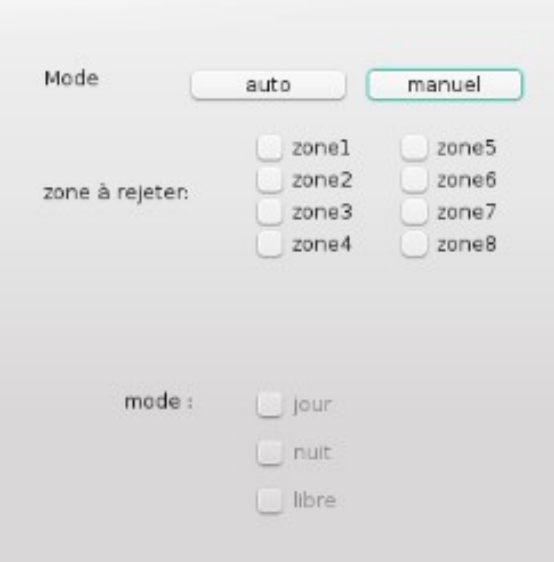
1) présentation :

Le système d'alarme pour le projet Mari-Sec doit pouvoir surveiller une résidence , stocker et faire remonter les informations relevé par l'installation .
Les informations sont stockée dans une base de données de façon a avoir une trace de chaque évènements (mise en marche , arrêt, intrusion …) , mais également pour pouvoir les exploiter via le site web et présenter un journal à l'utilisateur .
Comme présenté par le diagramme des exigences
-L'alarme doit comprendre plusieurs zones de surveillance (pouvant être des zones immédiates , qui déclenche l'alarme ;des zones différée , qui déclenche l'alarme après une durée donnée ; mais également des zones de réjections , utilisé pour rejeter certaines zones de façon à ce que celle si ne déclenche pas l'alarme) .
-L'utilisateur définis son mots de passe a chaque utilisations de l'alarme .
-Deux mode de fonctionnement peuvent être adopté par l'utilisateur le mode manuel , et le mode automatique .
-Un historique des évènements est à disposition , enregistrant les intrusions , mise en marche et arrêt de l'alarme .



2) mode manuel :

Le mode manuel permet à l'utilisateur de sélectionner les zones qu'il veut ne pas voir soumise a l'alarme . Il peut choisir autant de zone qu'il le souhaite tout comme il peut ne pas en choisir .
En cas d'intrusions dans une zone rejeté il ne se passeras rien (par exemple la nuit , la zone du couloir et rejeté et l'utilisateur passe par le couloir , il n'y aura aucun retour de l'alarme à la base de données . Alors que si la zone n'est pas rejeté l'alarme va envoyé une information a la base de données) .



3) mode automatique :

Le mode automatique propose a l'utilisateur des ensembles de rejections générale , comme un mode jour (ou toutes les zones déclenche l'alarme) , un mode nuit (ou les zones « utile » de nuit de déclenche pas l'alarme) .

4) codage :

D'une part il fallait faire en sorte que l'alarme envoie les données à notre base de données :
je déclare donc en public dans le fichier .h QSqlDatabase db; pour indiquer que ma base de données sera connut sous le nom de db .

Il faut donc ensuite définir les informations de connection à la base de données (dans l'ordre : l'adresse de notre base de données ; le nom de la base de données ; l'utilisateur et le mot de passe ainsi que le port)

```
//connection bdd
db = QSqlDatabase::addDatabase("QMYSQL");
db.setHostName("192.168.2.209");
db.setDatabaseName("Mari-Sec");
db.setUserName("marisex");
db.setPassword("marisec");
db.setPort(3306);
```

Pour que les enregistrements soit à la bonne heure j'ai du donc utilisé QdateTime::currentdatetime de façon a ce que l'heure des données soit en accord avec l'heure locale de l'alarme . Il faut ensuite convertir le QdateTime en qstring .

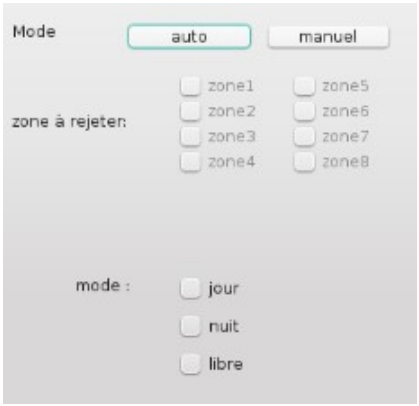
```
//heurelocale
time = QDateTime::currentDateTime();

QString lama = time.toString(); //temps en Qstring pour la bdd
```

Si une intrusions (simulé par un bouton) est détecté une requête est envoyé a la base de données pour enregistrer la zone , et l'heure de l'intrusion .

```
void Clavier::on_pushButton_zone1_clicked()
{

    if (!ui->checkBox_zone1->isChecked())
    {
        QMessageBox Message ;
        QString lama = time.toString();

        Message.setText("Intrusion Zone 1");
        Message.exec() ;


        db.open();
        QSqlQuery requeteur;
        if (requeteur.prepare("INSERT INTO intrusions VALUES (:heure,1);"))
        { requeteur.bindValue(":heure",lama);
            requeteur.exec();
        }}
}
```

De façon à ce que l'utilisateur ne puisse pas sélectionner de zone quand il est en automatique ou ne puisse pas choisir de mode prédéfinis quand il est en manuel ; le mode manuel est vérrouiller en automatique et l'automatique lui est verrouiller en manuel .

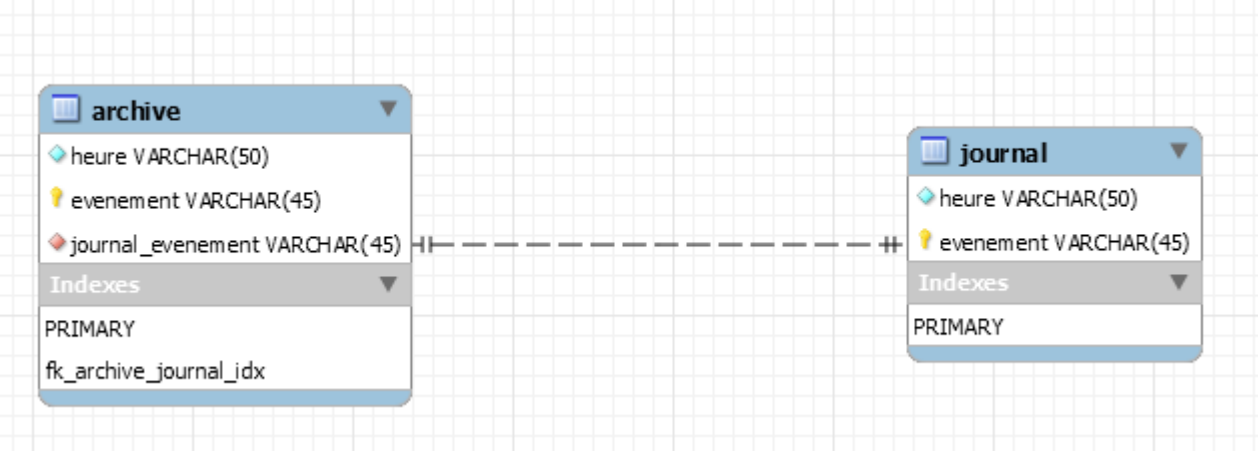
```
void Clavier::on_pushButton_auto_clicked()
{
    qDebug() << "bouton auto appuye.";
    //ui->statusBar->showMessage("test");
    ui->checkBox_zone1->setEnabled(false);
    ui->checkBox_zone2->setEnabled(false);
    ui->checkBox_zone3->setEnabled(false);
    ui->checkBox_zone4->setEnabled(false);
    ui->checkBox_zone5->setEnabled(false);
    ui->checkBox_zone6->setEnabled(false);
    ui->checkBox_zone7->setEnabled(false);
    ui->checkBox_zone8->setEnabled(false);
    ui->checkBox_jour->setEnabled(true);
    ui->checkBox_nuit->setEnabled(true);
    ui->checkBox_libre->setEnabled(true);
}

void Clavier::on_pushButton_manuel_clicked()
{
    ui->checkBox_zone1->setEnabled(true);
    ui->checkBox_zone2->setEnabled(true);
    ui->checkBox_zone3->setEnabled(true);
    ui->checkBox_zone4->setEnabled(true);
    ui->checkBox_zone5->setEnabled(true);
    ui->checkBox_zone6->setEnabled(true);
    ui->checkBox_zone7->setEnabled(true);
    ui->checkBox_zone8->setEnabled(true);
    ui->checkBox_libre->setEnabled(false);
    ui->checkBox_jour->setEnabled(false);
    ui->checkBox_nuit->setEnabled(false);
}
```



b) Base de données

La base de données à pour rôle de stocker les informations relatives a Mari-Sec :
Heures de mises en marche de l'alarme ,
Heures d'arrêt de l'alarme ,
Intrusions ,
Zone d'intrusions .

j'ai pour cela créée deux tables , une tables journal et une table archive .
Quand un enregistrement est reçut dans une de ces tables il est dupliquer dans la seconde .
Nous utilisons une table archive pour avoir une trace de tout éventement qui a eu lieux . La table journal quand a elle servira à retourner les nouveaux enregistrements depuis la dernière lecture (après chaque lecture les entrées de la table journal serons supprimé , si l'on veux de nouveaux ses information il faudra demander a la table archive)



C) **Physique :**

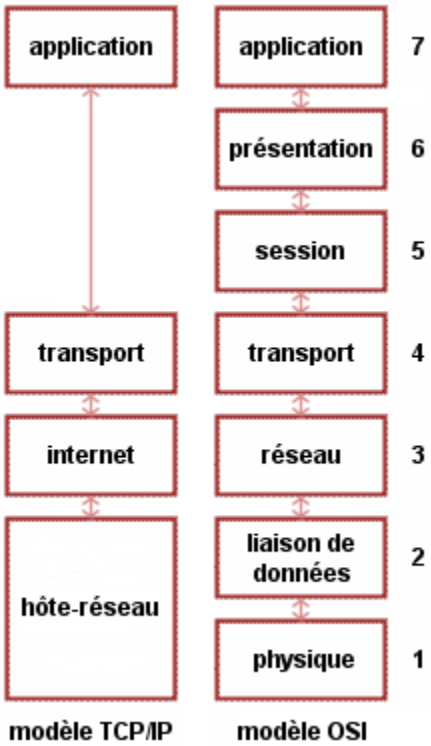
Protocole ethernet :

Ethernet est un protocole de réseau local à commutation de paquets (Chaque paquet est composé d'un en-tête contenant des informations sur son contenu et sa destination, qui permet au commutateur d'aiguiller le paquet sur le réseau vers son point final) .
Ce protocole répond au norme ISO/IEC 8802-3
Il peut également être utilisé dans des réseaux autres que ipv4 , comme par exemple arcnet , tokenring , etc... .

Les paires torsadées sont le matériel porteur le plus utilisé dans les installations , même si on trouve de plus en plus de fibre optique en raisons de son meilleur débits !

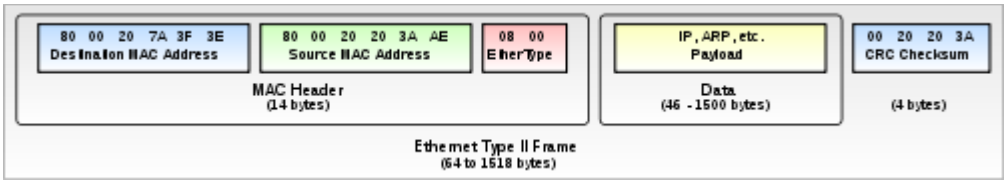
L'Ethernet est basé sur le principe de membres (pairs) sur le réseau, envoyant des messages dans ce qui était essentiellement un système radio, captif à l'intérieur d'un fil ou d'un canal commun . Chaque pair est identifié par une clé globalement unique, appelée adresse MAC, pour s'assurer que tous les postes sur un réseau Ethernet aient des adresses distinctes.

Ethernet et compatible avec plusieurs débits : 10 , 100 , 1000 Mbit/s et 10Gbit/s .



Ethernet ce situe au niveau deux du modele OSI , c'est à dire liaison de données .

Analyse d'une trame :



Dans une trame ethernet on trouve deux partie distinctes (sans compter le CRC) :
-La première partie dite « MAC » contient l'adresse MAC de destination (sur six octets), l'adresse MAC d'émission (également sur six octets) ainsi que « l'ethernet type » (qui sert à déterminer le type de réseaux utilisé , codé sur deux octets) .
-La seconde partie contient quand à elle les données ainsi que les informations relatives au type de réseaux (adresses ip sur réseaux ip , etc.) codé sur un maximum de 1500 octets .

Procédure principale

a- Trame prête à être transmise.
b- Si le medium n'est pas libre, attendre jusqu'à ce qu'il le devienne puis attendre la durée intertrame (9,6 µspour l'Ethernet 10 Mbit/s) et démarrer la transmission.
c- Si une collision est détectée, lancer la procédure de gestion des collisions. Sinon, la transmission est réussie.

Procédure de gestion des collisions

1. Continuer la transmission à hauteur de la durée d'une trame de taille minimale (64 octets) pour s'assurer que toutes les stations détectent la collision.
2. Si le nombre maximal de transmissions (16) est atteint, annuler la transmission.
3. Attendre un temps aléatoire dépendant du nombre de tentatives de transmission.
4. Reprendre la procédure principale.

D) CONCLUSION

Le projet fut dans l'ensemble une activité très enrichissante d'une part grâce à l'autonomie qui nous a était donné sur la réalisation (un choix vaste de solution pour arriver au but final) et donc les choix à faire pour progresser de la façon la plus efficace possibles .
Mais d'une autre part car j'ai également pus approfondir mes connaissances en SQL , C++ ainsi que découvrir le PHP.

Cependant au cours du développement j'ai rencontré de nombreux problèmes :
les erreurs sur la mise en place de la base de données , des problèmes suites à un mauvais adressage de la carte

A l'heure actuelle le projet n'est pas encore entièrement fonctionnel , le journal d'évènements n'est pas encore mis en place , l'IHM de l'alarme doit encore être modifié pour la rendre plus claire et simple d’accès ; il faudra également nettoyer la base de données des tables et enregistrements de test .

SAIDJ Sofiane

PARTIE PERSONNELLE ÉTUDIANT 3 :

SAIDJ Sofiane

Communication Client/Serveur

Sommaire

I –INTRODUCTION……...p.96

Présentation du candidat……..p.96

II- PRÉSENTATION DU PROJET DANS SON ENVIRONNEMENT…….p.97

Synoptique de l'architecture matériel……..…p.98

III- OBJECTIFS PROFESSIONNELS DU PROJET……..…p.99

IV – TRAVAIL DE L'ÉTUDIANT……..p.99

Analyse UML.....…….....p.100

Quelques définitions……..…p.102

V- GANTT…….……p.104

Diagramme de GANTT réel……..p.104

VI – PRÉSENTATION DÉTAILLÉE DU SITE WEB……..….p.105

Configuration…….p.105

Codage……...…p.106

Test de fonctionnalité…….p.108

VII - PRÉSENTATION DÉTAILLÉE DU CLIENT /SERVEUR……...….p.109

Configuration……..…p.109

Codage……...…p.109

Test de fonctionnalité……..…p.114

VIII - PROBLÈMES RENCONTRES……...p.114

IX- NOTIONS DE PHYSIQUE AU SEIN DU PROJET ……..…p.115

X – CONCLUSION…….……p.116

I-INTRODUCTION

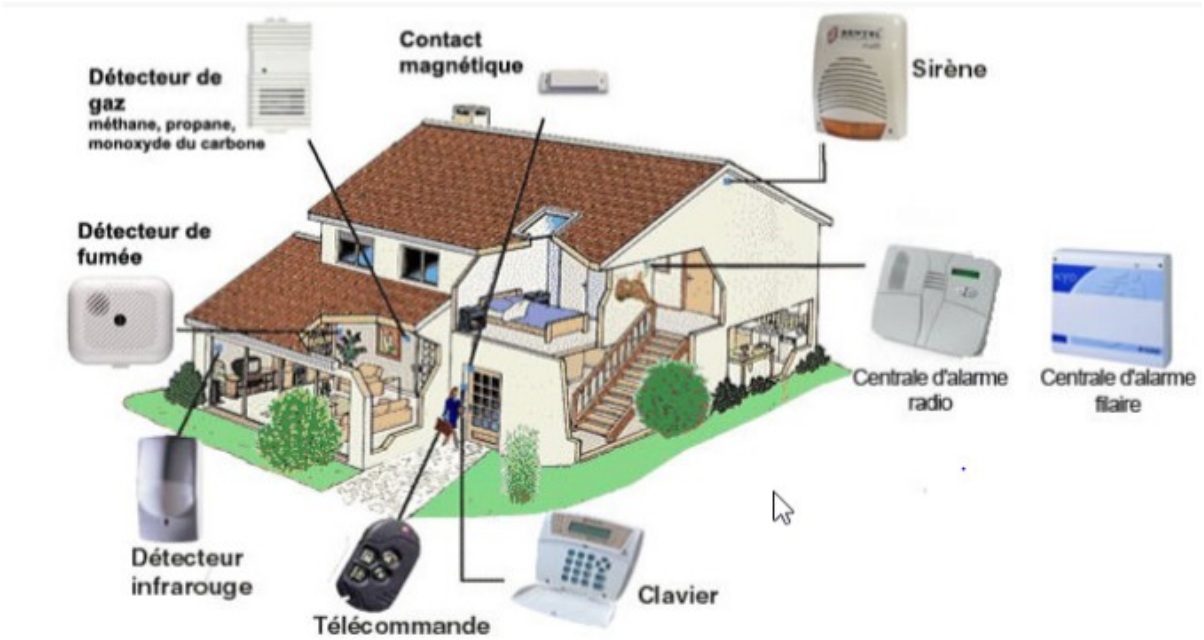
En vue d'obtenir ce BTS, un projet de fin d'année doit être réalisé durant la 2e année de celui-ci. Ce projet est conçu avec un groupe d'élèves et dirigé par un professeur.

Présentation du candidat

Je me nomme Sofiane Saidj, j'ai effectué un BAC STI2D. Présentant quelques connaissances dans le domaine de l'informatique j'ai décidé de me diriger vers un BTS Systèmes Numériques (SN) avec l'idée de continuer dans ce domaine après l'obtention de celui-ci.

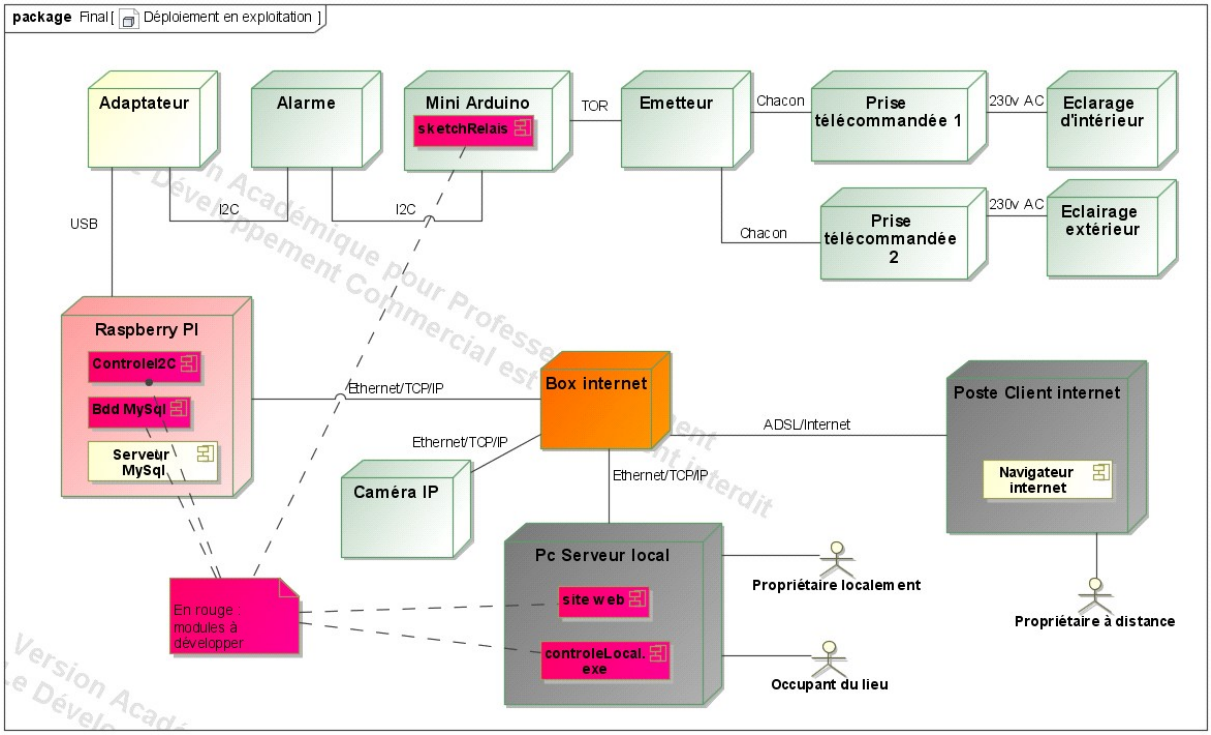
Dans ce projet ma partie sera de réaliser la communication entre le PC serveur local et la carte électronique Raspberry PI par client/serveur avec les sockets QT.

II- PRESENTATION DU PROJET DANS SON ENVIRONNEMENT



Notre projet a pour but de prolonger les activités de la société dans le domaine de la domotique et d'acquérir une autonomie dans la fourniture de matériels et logiciels nécessaires pour l'intégration aux produits existants. Dans ce projet il s'agit plus précisément de gérer la sécurité au moyen d'une caméra placée à l'extérieur (surveillance du jardin), d'une alarme et de prévenir en simulant la présence des occupants par allumage de la lumière selon des scénarii réalistes afin d'éviter les repérages en cas d'absences (vacances ou travail).

Synoptique de l'architecture matériel :



Ce projet sera constitué d'une caméra IP ainsi qu'un PC Serveur local contenant un site web et un contrôle local relié sur le réseau pour le propriétaire localement et pour l'occupant du lieu. Le propriétaire à distance pourra également se connecter via le site web grâce a des identifiants. Le PC Serveur Local sera relié par des sockets QT sur la carte Raspberry PI. Sur cette carte se trouvera un contrôle I2C et une Base de données et, le serveur prendra les informations de la base de données afin de l'envoyer par USB sur l'alarme en passant par un Adaptateur qui convertira l'USB en I2C, puis a la mini Arduino et à l'émetteur par un Tout Ou Rien et l'émetteur sera relié à 2 prises télécommandée en 230v pour gérer l'éclairage intérieur et extérieur.

III- OBJECTIFS PROFESSIONNELS DU PROJET

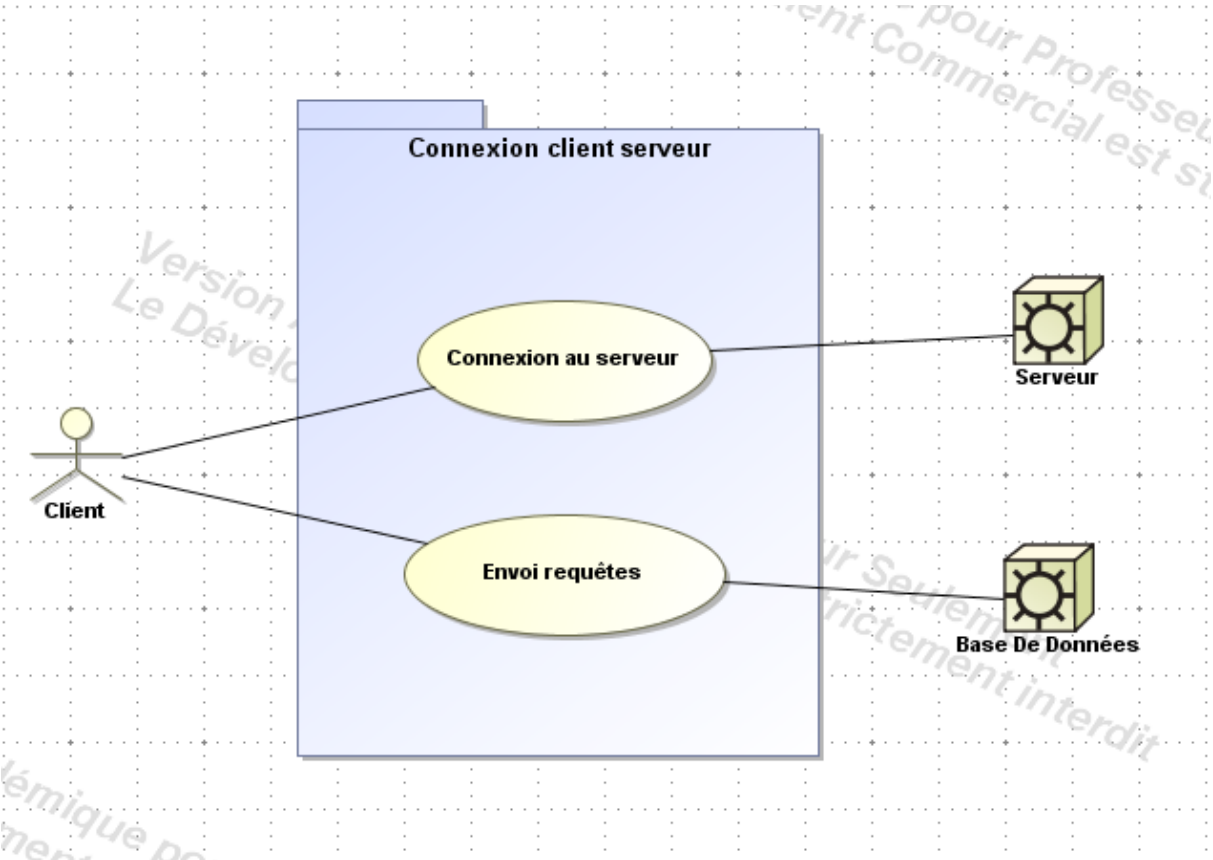
Afin de bien se répartir l'ordre des tâches à effectuer nous nous sommes concerté en équipe pour savoir lors de la réunification des tâches laquelle serait plus judicieuse de faire en premier. Nous avons fait des analyses, nous avons trouvés des alternatives nous nous sommes résumer la démarche d'élaboration du projet professionnel. Pour ma part j'ai donc commencé par la page d'identification du site web puis je me suis ensuite penché vers la communication client/serveur ainsi qu'à l'envoi de requêtes SQL a la base de données pour pouvoir ajouter ou récupérer des valeurs.

IV – TRAVAIL DE L'ÉTUDIANT

Dans ce projet ma partie sera de réaliser la communication entre le PC serveur local et la carte électronique Raspberry PI par client/serveur avec les sockets QT. Afin de réaliser cette tâche j'utiliserais l'environnement de développement QT Creator

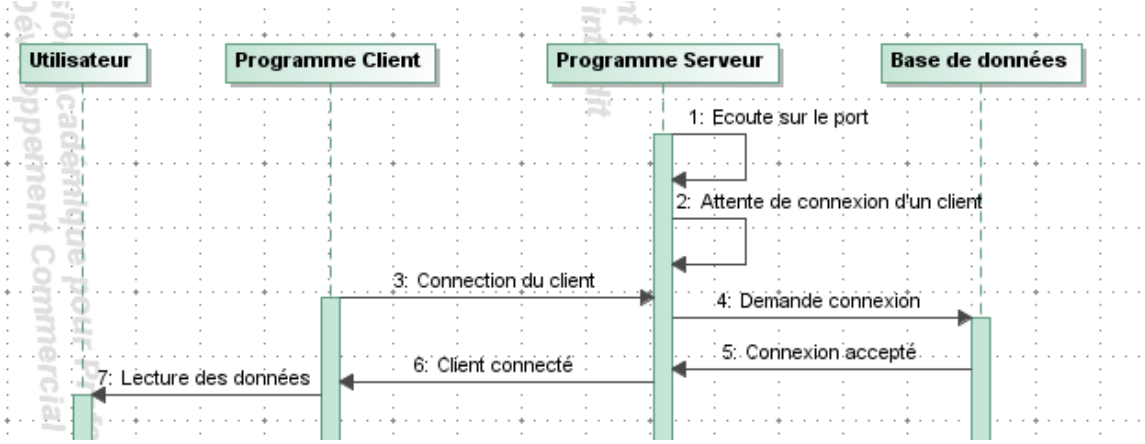
Diagramme de cas d'utilisation :

Le diagramme de cas d'utilisation est un diagramme qui nous permet d'avoir une vision plus globale du projet. Là, nous aurons une vision plus globale de la connexion client/serveur.



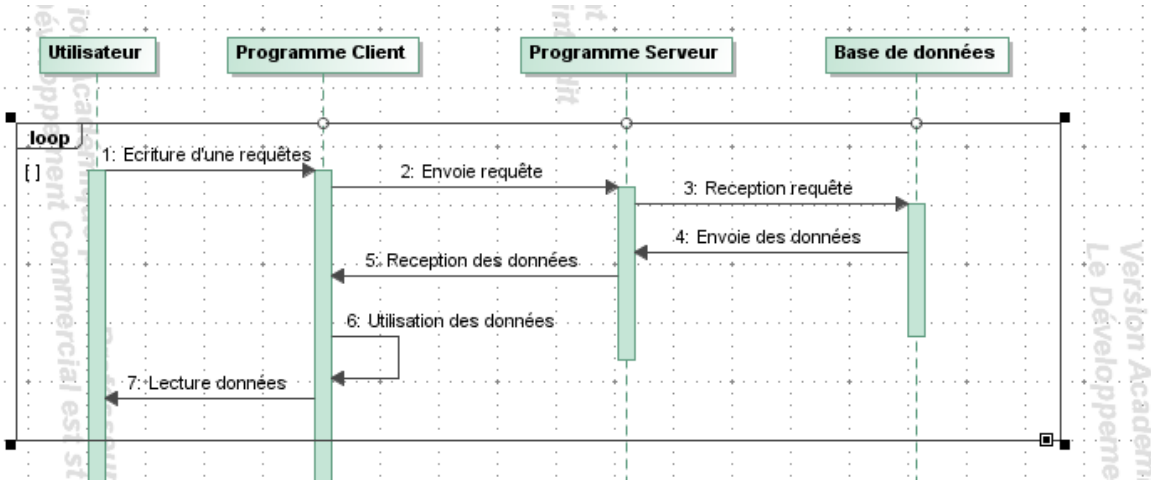
OSD-S SN Connexion au serveur :

Le diagramme de séquence représente graphiquement des interactions entre les acteurs et le système selon un ordre chronologique dans la formulation. On retrouve ici le diagramme de séquence nominal de la connexion au serveur.



OSD-S SN Envoi requête :

Voici ici la représentation du diagramme de séquence nominatif de l'envoi de requêtes à la base de données jusqu'à la lecture des donnés par l'utilisateur



Quelques définitions :

I2C : L'I2C (Inter-Integrated Circuit) est un bus Informatique. C'est un dispositif de transmission de données partagé entre plusieurs composants d'un système numérique sur une même ligne de transmission.

BDD : Une BDD (base de données) est un conteneur stockant des données telles que des chiffres, des dates ou des mots, pouvant être retraités par des moyens informatiques pour produire une information. Par exemple, des chiffres et des noms assemblés et triés pour former un annuaire.

MySQL : MySQL est un système de gestion de bases de données relationnelles .C'est un serveur de bases de données relationnelles SQL. Il supporte le langage de requêtes SQL

SQL : SQL (Structured Query Language) est un langage informatique servant à exploiter des bases de données relationnelles. La partie langage de manipulation des données de SQL permet de rechercher, d'ajouter, de modifier ou de supprimer des données dans les bases de données relationnelles.

SOCKETS : Les sockets servent à communiquer entre deux hôtes appelés client/serveur a l'aide d'une adresse IP et d'un port. Elles permettent de gérer le flux entrant et sortant afin d'assurer une communication entre les deux.

PROTOCOLE CHACON : Le protocole CHACON, procure à chacun des périphériques émetteurs un ID unique

CARTE RASPBERRY PI : La carte Raspberry Pi est un nano-ordinateur mono-carte qui a la taille d'une carte de crédit il permet l'exécution de plusieurs variantes du système d'exploitation libre GNU/Linux et des logiciels compatibles permettant un apprentissage de l'informatique

APACHE 2 : Apache est un serveur http

SERVEUR HTTP : C’est un logiciel servant des requêtes respectant le protocole de communication client-serveur

PHP : est un langage de programmation libre, principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP

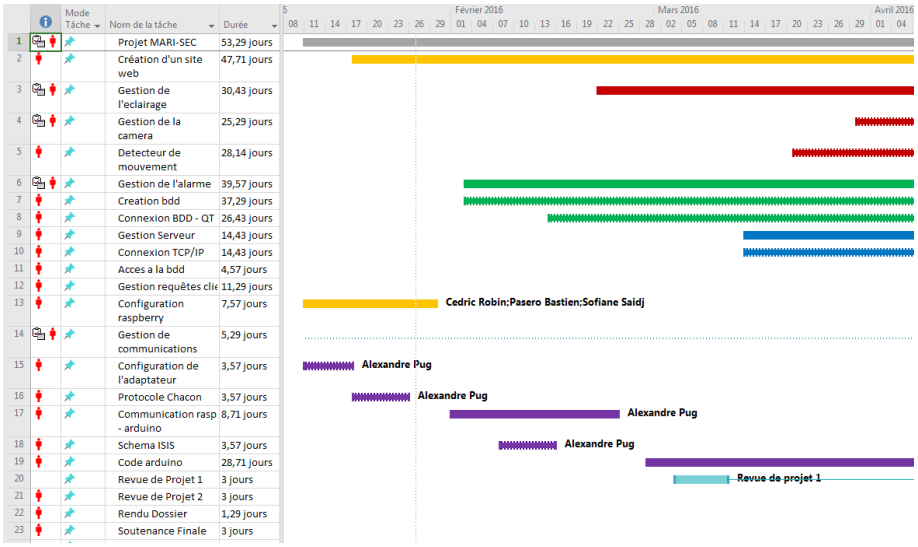
QT CREATOR : Qt Creator est un environnement de développement intégré multi plate-forme faisant partie du framework Qt. Il est donc orienté pour la programmation en C++.

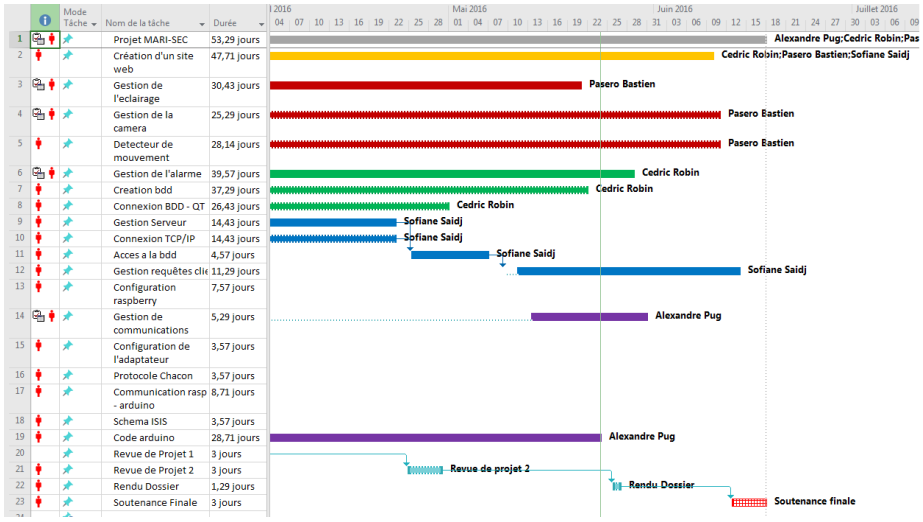
CSS : (*Cascading Style Sheets*), c'est un langage qui vient compléter le HTML notamment pour la mise en forme de pages web.

V- GANTT

Le Diagramme de GANTT Prévisionnel à été vu dans la partie commune.
Voici donc le diagramme de GANTT réalisé selon le déroulement des tâches au cours du projet.

Diagramme de GANTT Réel :





Suite au vol de ma carte Raspberry PI j'ai du changer ma prévision des tâches à réaliser, c'est donc pour cela que mes tâches ne sont pas toutes respectées.

VI – PRÉSENTATION DÉTAILLE DU SITE WEB

Le PC Serveur sous Windows est doté d'un site web accessible par identifiants et mot de passes, il permet d'accéder aux informations de la base de données (Zones, modes, événements,…) situés dans la Raspberry PI.

La carte électronique Raspberry PI sera composée d'une Base De Donnée intégrant les informations concernant la maison (Porte ouverte, alarme déclenchée,..). Elle jouera aussi le rôle de serveur car le site web devra se connecter à la base de données pour acquérir les informations sur la sécurité de la maison.

Configuration

Le site web sera lui constitué d'une page d'identification pour le propriétaire localement et l'occupant du lieu qui se connecteront localement au PC Serveur et pour le propriétaire à distance qui se connectera depuis un autre poste depuis un navigateur web.

Pour la création de ce site web nous le développerons aussi sur mobile tablettes et autre objets connectées. Pour réaliser ceci nous utiliserons bootstrap, un framework destiné à faciliter la création d'applications web

Ce site web sera constitué une page d'accueil, d'une page sources, d'une page de contact et dd'une page « Tableau de bord « su laquelle nous pourrons accéder au contrôle de la caméra extérieure ainsi qu'a un journal d'événement, nous permettant de savoir si l'alarme à été activé, désactivé, si une porte n'est pas fermée…

Lorsque la connexion au site sera effectuée nous pourront accéder aux notifications concernant l'alarme, au flux vidéo de la caméra ayant une vue sur le jardin et nous pourront programmer une ou plusieurs actions permettant de faire croire qu'une personne est dans la maison (ex : Lumière allumée pour 17h dans la cuisine, pour 18h dans la salle de bain,..)

Pour créer la page d'identification j'ai dû dans un premier temps installer Wamp server pour la tester localement, WAMP server est une plate-forme de développement Web sous Windows pour des applications Web utilisant le serveur Apache 2, du langage de scripts PHP et d'une base de données MySQL. Puis j'ai créé un fichier .php pour y commencer l'écriture de ma page d'identification. Dans cette page j'ai édité l'image ci-dessus en html et j'ai utilisé le langage php pour réaliser la connexion sur la base de donnée à travers une identification.

Codage :

Code html pour avoir l'interface graphique présente ci-dessus :

```
<form action="./Acceuil/Acceuil.php" method="post" class="id">
        <div class="identification">
                <input type="text" class="form-control" placeholder="Identifiant" name="utilisateur">
        </div>
         <div class="identification">
                <input type="password" class="form-control" placeholder="Mot de passe" name="mdp">
         </div>

         <div class="identification">
                <input type="submit" class="btn btn-primary btn-block btn-flat" name="Submit">
          </div>

 </form>
```

La méthode POST est indispensable pour des codes non ASCII, des données de taille importante, et elle est recommandée pour modifier les données sur le serveur, et pour les données sensible.
Code php pour l'identification :

```php
<?php

if(isset($_POST['Submit']))
{
    $bdd = mysqli_connect("192.168.2.214:80", "root", "pipod", "Mari-Sec");

    $result = mysqli_query($bdd, "SELECT * FROM utilisateurss WHERE nom = ''");
    while ($row = mysqli_fetch_assoc($result))
    {
       $utilisateur = $row["nom"];
       $motdepasse = $row["mdp"];
    }
    if($_POST["utilisateur"] == $utilisateur)
        {
            echo "utilisateur correcte";
            if($_POST["mdp"] == $motdepasse)
            {
                echo "mdp correcte";
            }
            session_start();
        }
}

```

La ligne 5 permet de ce connecter a la base de donnée
La ligne 8 retourne ligne par ligne tout les résultats des utilisateurs de la base de donnée.
La ligne 13 permet de vérifier que l'utilisateur entré sur le site est le même que celui correspondant à la base de donnée pour pouvoir ce connecter
La ligne 16 est identique à la 13e ligne sauf que cette fois nous effectuons la vérification du mot de passe

Le code permettant la liaison entre le formulaire et le code php est :

```
<form action="./Acceuil/Acceuil.php" method="post" class="id">
```

Site web version Mobile :

Pour que le site web soit disponible sur téléphones, tablettes et autre objets connectés j'ai dû dans un premier temps installer bootstrap ainsi que la librairie Java Script JQuery. Ensuite dans le <head> de mon index j'ai dû rajouter le chemin de mon bootstrap ainsi que le chemin de mon CSS.

```
<link href="/lib/bootstrap/css/bootstrap.min.css" rel="stylesheet">
<link href="/css/microcms.css" rel="stylesheet">
```

Il a fallu modifier quelques lignes pour permettre aux appareils connectés en réseau local de pouvoir accéder au site.
Dans le fichier httpd.conf → C:\wamp64\bin\apache\apache2.4.17\conf

```
<Directory "C:/wamp64/www/web/">
  Allow from all
 Require all granted
</Directory>
```

et ajouter le chemin du site web DocumentRoot "C:/wamp64/www/web"

Test :

Les premiers tests ont étés assez compliqué quant au placement des images et textes sur le site web car nous n'utilisions pas CSS au début, puis nous avons utilisé CSS et la mise en page du site a été plus simple.

Le test sur la base de données, sur nos premiers tests le site ne voulait pas se connecter à la base de données. Mais après vérification du code nous avons pu nous connecter à la base de données et ainsi pouvoir nous connecter sur le site web en passant par la page d'identification.

Le test de fonctionnement du site sur WampServeur à très bien fonctionné

VI - PRESENTATION DETAILLE DU CLIENT /SERVEUR

Configuration :

Pour réaliser la communication entre la carte Rasberry PI et le PC serveur local nous utiliseront le protocole TCP/IP basé sur la relation client serveur. Dans notre cas, le PC serveur jouera le rôle de client et la carte jouera le rôle de serveur.

Codage :

Explication de code : création serveur :

Dans un premier temps il a fallu créer un serveur héritant de QObject pour simplifier la communication avec les signaux et slots.

Ensuite j'ai fait dans le constructeur afin qu'il soit réalisé dès le lancement du programme un connect en mettant un pointeur vers l'objet qui émet le signal (&server), le signal que l'on souhaite intercepter (newConnection()), le pointeur vers l'objet qui contient le slot (ici This), et le slot qui doit s'exécuter lorsque le signal se produit (slt_demande_connexion()) :

```
connect(&server, SIGNAL(newConnection()), this, SLOT(slt_demande_connexion()));
```

Puis nous fessons écouter le serveur sur toutes les adresses du port 2222 ainsi qu'une vérification d'écoute du serveur par un simple « if ».

```
server.listen(QHostAddress::Any,2222);
   if (server.isListening())
      qDebug() << "Server en écoute.";
```

Dans le slot slt_demande_connexion :

Nous allons récupérer la connexion du client pour la stoker dans la variable clientConnection, ensuite nous ferons un qDebug pour voir qu'un client ce connecte et nous ferons un connect avec l'objet pointé clientConnection, le signal readyRead(), le pointeur vers l'objet qui contient le slot et le slot slt_lecture() *

```
void serveur::slt_demande_connexion()
 {
  clientConnection = server.nextPendingConnection();
  qDebug() << "Connexion d'un client" << endl;
    connect(clientConnection, SIGNAL(readyRead()),this, SLOT(slt_lecture()));

}
```

Dans le slot lecture:

Nous allons ici faire une boucle ou on lira la ligne retournée par la socket et on lit les lignes tant qu'il y en a. Puis on affiche avec qDebug « message reçu ».

```
void serveur::slt_lecture()
{
   while(clientConnection->canReadLine())    // tant qu'on peut lire sur la socket
   {
      clientConnection->readLine(); //On lit une ligne
      qDebug()<< "Message reçu " <<endl;
}
```

Explication de code : Création du client :

Le programme client est un programme qui permet d'établir la connexion au serveur et d'envoyer des requêtes au serveur, qui lui même enverra les requêtes à la base de données et communiquera au client les informations stockés dans la base de données. Le client que je dois réaliser est un client lourd, cela veut dire que le traitement de la réponse à la requête du client utilisateur va mettre en œuvre un travail combine entre l'ordinateur serveur et le poste client

Le programme client héritant tout comme du serveur du QObject, nous allons dans un premier temps effectuer un connect dans le constructeur avec le pointeur qui émet vers l'objet (&client), le signal que l'on va intercepter (connected()), le pointeur vers l'objet qui contient le slot (this), et le slot qui s'exécute lorsque le signal ce produit (startTransfert()).

Connect (&client, SIGNAL (connected ()), this, SLOT (startTransfer()));

Dans le void start ():

Dans le "void start" on fera un connectToHost pour ce connecter au serveur, comme nous testerons le serveur en local nous mettrons en paramètres "localhost" et le port 2222 car nous avons définis dans le programme serveur qu'il devait écouter sur le port 2222.

```
void tcpclient::start()
{

   client.connectToHost("localhost",2222);
   qDebug() << "Demande de connexion.";
}
```

Lorsque le client est bien connecté il peut envoyer des requêtes qui iront au serveur et les enverra à la base de données qui renverra les informations jusqu'à l'utilisateur.

```
void Fihm::on_BtRecuperer_clicked()
{
   QSqlQuery query;
   query.prepare("INSERT INTO utilisateurs (id, mdp, nom) "
                 "VALUES (:id,:mdp,:nom)");
   query.bindValue(":id", 02);
   query.bindValue(":mdp", "saidj");
   query.bindValue(":nom", "saidj");
   query.exec();

   if (!query.exec()) {
       qDebug() << "Erreur lors de l'exécution de la requête: ";
   }
```

Connexion serveur/Base de données :

Pour effectuer la connexion vers la base de données :

```
   test = QSqlDatabase::addDatabase("QMYSQL"); // Création d'une connection vers BDD
   test.setHostName("192.168.2.209"); //Adresse ou la BDD est installée
   test.setDatabaseName("Mari-Sec"); //Nom de la BDD
   test.setUserName("root"); // Nom d'utilisateur de la BDD
   test.setPassword("admin"); // Mot de passé de la BDD
```

Test :

La base de données : J'ai testé dans un premier temps la base de données. La base de données se connectait bien.

La connexion client/serveur : j'ai essayé de tester mes 2 programmes client-serveur sur le même PC dans un premier temps, sur Windows le programme serveur ne marchait pas car il manquait des "dll". J'ai donc testé la connexion sur Linux car le programme serveur sera sur la carte Rasberry PI et la connexion entre le client et le serveur a bien fonctionné.

Ensuite j'ai testé les requêtes, j'ai eu quelques soucis quant a l'envoi de requêtes sur la base de données car la base de données n'était simplement pas ouverte.

VII - PROBLÈMES RENCONTRES

Dans ce projet nous avons rencontrés quelques problèmes comme la compréhension de nos tâches à réaliser dans le projet ainsi que l'ordre des priorités de ces tâches.
Nous avons eu aussi quelques soucis quant au codage de certaines tâches, mais grâce à quelques tutoriels sur internet nous avons pu nous en sortir et régler nos problèmes seuls.

VII- NOTIONS DE PHYSIQUE AU SEIN DU PROJET

Dans mon projet les programmes passent par des câbles Ethernet, il y a donc plusieurs choses à expliquer comme les câbles Ethernet, la liaison ADSL ainsi que le protocole utilisé, ici le TCP/IP.

Ethernet :

Le câble Ethernet que nous utilisons est un câble de catégorie 7 ce qui veut dire que le câble est composé de 4 paires torsadées et de classe F qui veut dire que la Fréquence Maximale possible à avoir est de 600 MHz. Ce câble est un câble SFTP ce qui veut dire que le blindage est un blindage (feuillard & tresses), il possède une vitesse de 9600Bps et répond à la norme ISO/CEI 11801 :2002
Notre câble possède une vitesse de 40 Gigabit sur 50m et de 100 Gigabit sur 15m mais les chercheurs de 'The Pennsylvania State University ' pensent que les circuits de 22 ou 32nm permettront des vitesses de 100 Gigabits sur 100m.

La liaison ADSL :

Tout d'abord la liaison ADSL signifie : « *asymmetric digital subscriber line* ». Le flux de données est plus important dans un sens que l'autre car comme son nom l'indique le débit est asymétrique. Elle permet d'avoir un débit plus rapide sur le net. Il y a une modulation de 4000 symboles/secondes ce qui correspond à 4000 trames/secondes.

Voici quelques caractéristiques techniques de notre boxe Internet :

Atténuation montante	16 dB
Atténuation descendante	27 dB
Marge au bruit sens montant	6 dB
Marge au bruit sens descendant	5 dB
Débit montant maximum	1 017 kb/s
Débit descendant maximum	1 6304 kb/s

Protocole TCP/IP :

Il existe 2 type de protocoles : Le protocole TCP/IP et le protocole UDP.
Pourquoi le protocole TCP/IP et pas le protocole UDP ? Le protocole UDP est un protocole non orienté connexion et ne fournit pas de contrôle d'erreur. Nous utiliserons donc le protocole TCP/IP.
Le protocole TCP (Transmission Control Protocol)/IP (Internet Protocol) est utilisé pour le transfert des données sur Internet. Le modèle TCP/IP est composé de 4 couches physiques : La couche Application qui inclue HTTP, SSH, DNS, RIP, FTP, la couche transport qui permet de résoudre les échanges de données et de faire arriver le message dans l'ordre correct et détermine aussi à quelle application chaque paquet de données doit être délivré.
La couche Internet est aussi utilisée pour résoudre le problème d'acheminement de paquets à travers un seul réseau et la couche Accès réseau qui permet de spécifier comment les paquets sont transportés sur la couche physique et de décrire les caractéristiques physiques de la communication.

IX- CONCLUSION

Ce projet de BTS s'est bien déroulé, il a été agréable a réalisé et a été bénéfique pour moi car il m'a permis d'apprendre plusieurs choses sur la programmation et aussi, du fait d'avoir utilisé un cahier des charges il m'a également permis de mieux appréhender les besoins d'une entreprise. D'autre part d'un point de vue plus personnel, j'ai appris à travailler en équipe et a être à l'écoute des autres membres du projet.

ANNEXE 1 (fiches de réunion)

Ces tableau sont les résumer des diverses réunions du groupe, ainsi que des décisions qui ont été prises au cours de celles-ci.

1ere Réunion de Projet : MARI SEC	
Personnes présentes :	Objectif : Accord sur les limites de chacun dans la répartitions des taches. Organiser la communication. Prévoir le temps à chaque étape pour la prochaine réunion. Discussion des problèmes éventuels
PUG PASERO SAIDJ ROBIN	
Décisions prises: -PC Serveur local sous Windows ou Linux ? Windows La création du site web est telle nécessaire ? Oui Alarmes Virtuelle, réelle, ou maquette. ? Virtuelle Adaptateur I2C ? Oui Matériel nécessaire -> Maquette, Raspberry Pi, logiciel QT, caméra.	

2nd Réunion de Projet : MARI SEC	
Personne présentes :	Objectif : Résolution des détails techniques utiles pour la continuité du projet.
PUG PASERO SAIDJ ROBIN Mr SERRE	
Réponses aux questions principales : - Adaptateur I2C fourni ou a faire ? Fourni ainsi que sa documentation. - Qui s'occupe entièrement du site web ? Bastien et Sofiane -Utilisateur et droits de la BDD? "Mari-sec" sans restrictions (adresse 192.168.2.209) - Deux interface pour les deux utilisateurs ? Oui, tout sur le site web pour la visualisation, le contrôle est en local (PC Serveur) - Qui joue le rôle du serveur et du client ? RPI est le serveur, son client est le "PC Serveur"	

ANNEXE 2 (fiches de réunion)

<table>
<tr><td colspan="2">3e Réunion de Projet : MARI SEC</td></tr>
<tr><td>Personne présentes :</td><td rowspan="2">Objectif : Résolution de problèmes sur le projet</td></tr>
<tr><td>PUG PASERO
SAIDJ ROBIN
Mr SERRE</td></tr>
<tr><td colspan="2">Faut-il créer une détection de mouvement ou utiliser celle de la camera déjà définie ? Il faut la créer.

Peut-on passer le PC serveur local sous Raspberry ? Non

Quelles sont les fonctions principales des requêtes ? Insérer et récupérer des valeurs</td></tr>
</table>

ANNEXE 3 (Diagrammes fournis)

Diagrammes de déploiement :

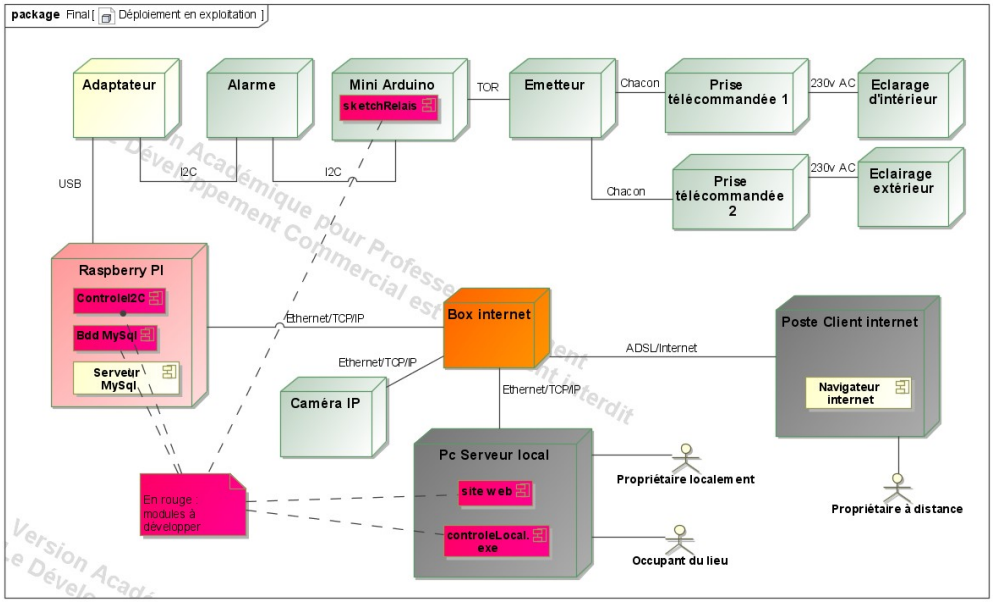
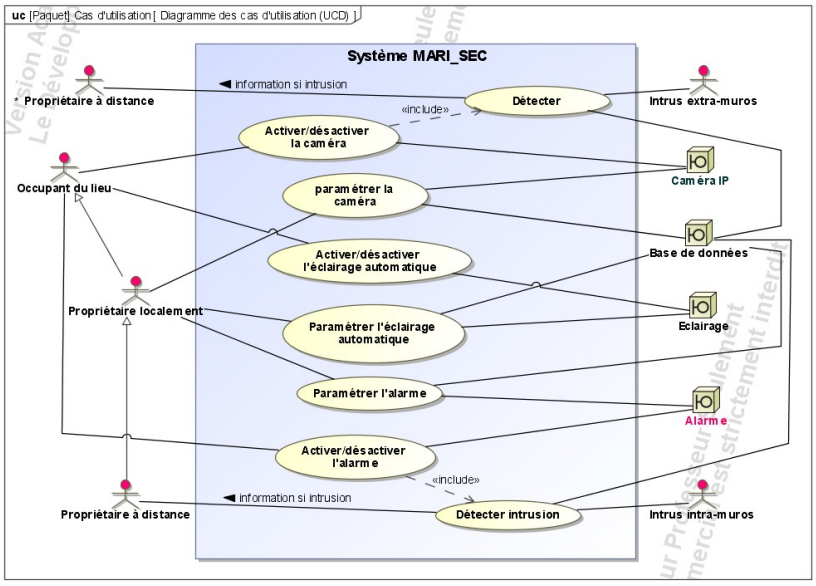


Diagramme de cas d'utilisation :

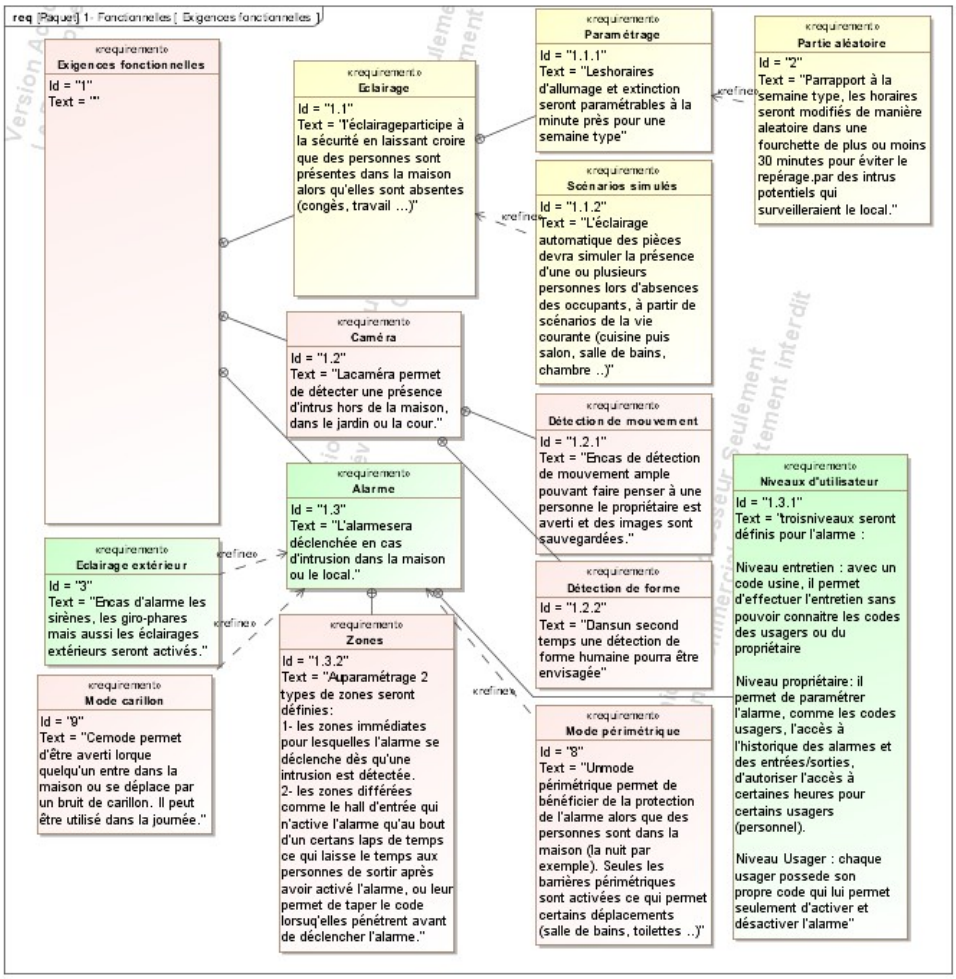


ANNEXE 4 (Cahier des charges)

Cahier des charges – Expression du besoin

Il s'agit de prolonger les activités de la société dans le domaine de la domotique et d'acquérir une autonomie dans la fourniture de matériels et logiciels nécessaires pour l'intégration aux produits existants. Dans ce projet il s'agit plus précisément de gérer la sécurité au moyen d'une caméra placée à l'extérieur (surveillance du jardin), d'une alarme et de prévenir en simulant la présence des occupants par allumage de la lumière selon des scénarii réalistes afin d'éviter les repérages en cas d'absences (vacances ou travail).

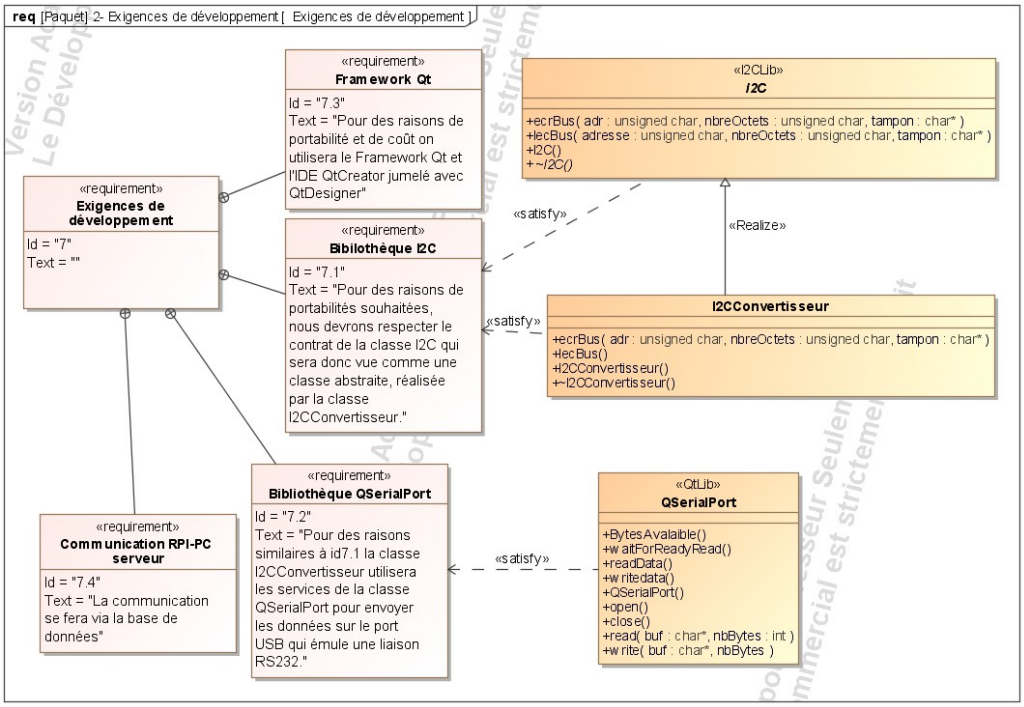
Diagramme d'exigence :



ANNEXE 5 (Contraintes)

Contraintes de réalisation :

- Contraintes de développement (matériel et/ou logiciel imposé / technologies utilisées) :



Contraintes qualité (conformité, délais, …) :

Les cas d'utilisation devront être respectés et testés. Le code devra être commenté, la notation hongroise utilisée, la traçabilité sera rigoureuse (rétro-conception conseillée).
Les délais devront être respectés.
Le dossier sera unique pour les 4 étudiants, les données seront de couleur blanche, parties communes seront de couleur bleu clair,
Et chaque étudiant aura une couleur personnelle (claire). Les pieds de page porteront le nom de l'étudiant rédacteur.

Contraintes de fiabilité, sécurité :

Le logiciel devra être robuste, les contrôles de saisie systématiques, les retours de fonction testés, les exceptions traitées.
Les accès ainsi que la communication seront sécurisés.

ANNEXE 6 (Répartition des taches)

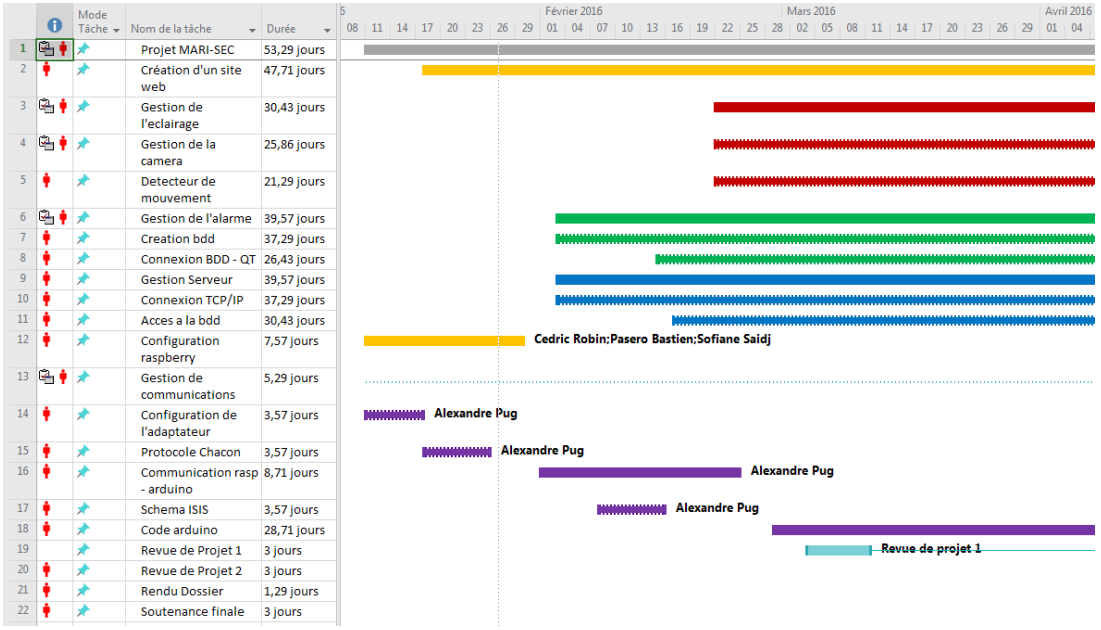
	Fonctions à développer et tâches à effectuer	
Étudiant 1 IR 1	*Liste des fonctions assurées par l'étudiant* Gestion de la détection par la caméra. Gestion de l'éclairage.	Installation : Qt. MySql. Mise en œuvre : Pilotage de l'éclairage via le bus I2C et le PCF8574. QtSerialPort. Récupération d'une image de la caméra. Configuration : Réalisation : Pilotage de l'éclairage selon des scénarii stockés dans la BDD. Détection d'un déplacement puis éventuellement dans un deuxième temps reconnaissance de forme. Documentation : Installation, mise en service, dossier de développement.
Étudiant 2 IR 2	*Liste des fonctions assurées par l'étudiant* Gestion de l'alarme.	Installation : Qt, MySql. Mise en œuvre : MySql. Graphique sous Qt. Configuration : MySql. Réalisation : Etablissement des scénarii de l'éclairage et stockage dans la BDD. Alarme. Documentation : Installation, mise en service, dossier de développement.

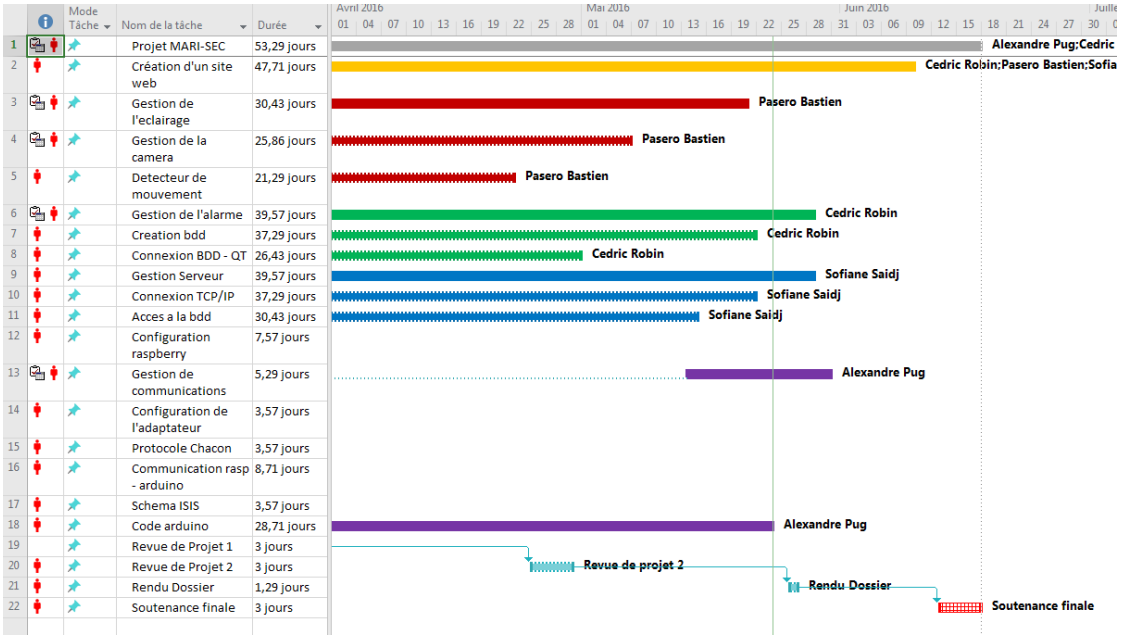
ANNEXE 7 (Répartition des taches)

Étudiant 3 IR 3	*Liste des fonctions assurées par l'étudiant* Communication entre le PC serveur local et la Raspberry PI par client/serveur avec les sockets Qt.	Installation : Qt. Mise en œuvre : Sockets Qt. Configuration : Réalisation : Communication client lourd/serveur avec les sockets Qt. Documentation : Installation, mise en service, dossier de développement.

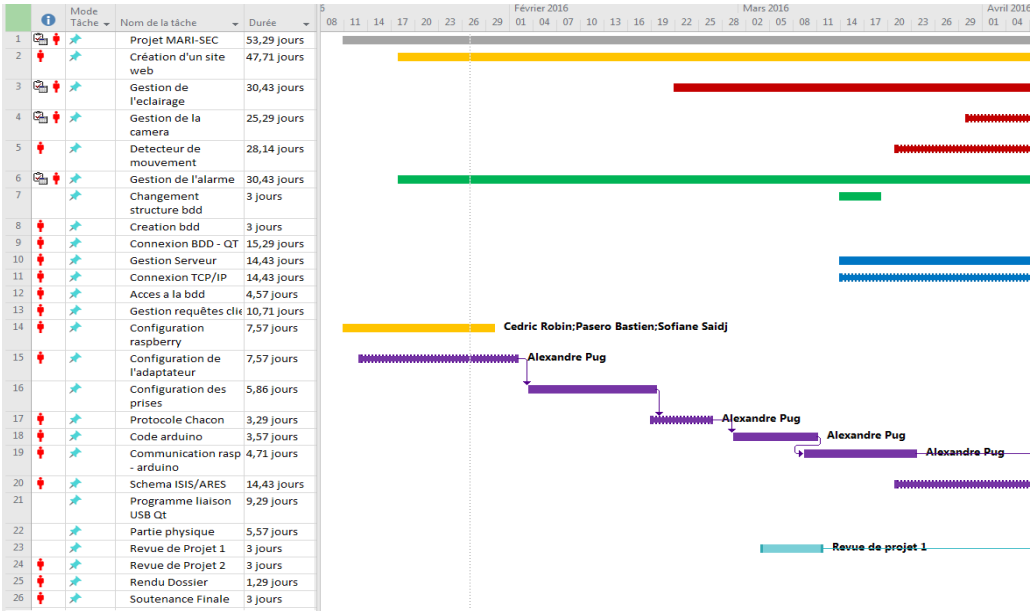
Étudiant 4 EC 1	*Liste des fonctions assurées par l'étudiant* Mise en œuvre de la communication entre la carte Raspberry PI et la carte Arduino mini. Pilotage de l'émetteur des prises télécommandées.	Installation : Qt, librairies Arduino, librairie wiringPi Mise en œuvre : Communication I2C Rpi ↔ Arduino Communication Arduino → prises gigogne (protocole Chacon) Configuration : Réalisation : Prototypage rapide avec Arduino UNO R3, puis fabrication avec Arduino mini Documentation : Installation, prototypage/mise au point, documents de fabrication et programmes.
Tous les étudiants	*Domaines de Sciences Physiques à traiter par l'ensemble des étudiants de l'équipe projet :* • *4 2 : Transmissions par fréquence porteuse* • *4.3 : Antennes* • *5.1 Colorimétrie* • *5.3 Images numériques*	

ANNEXE 8 (GANTT prévisionnel)

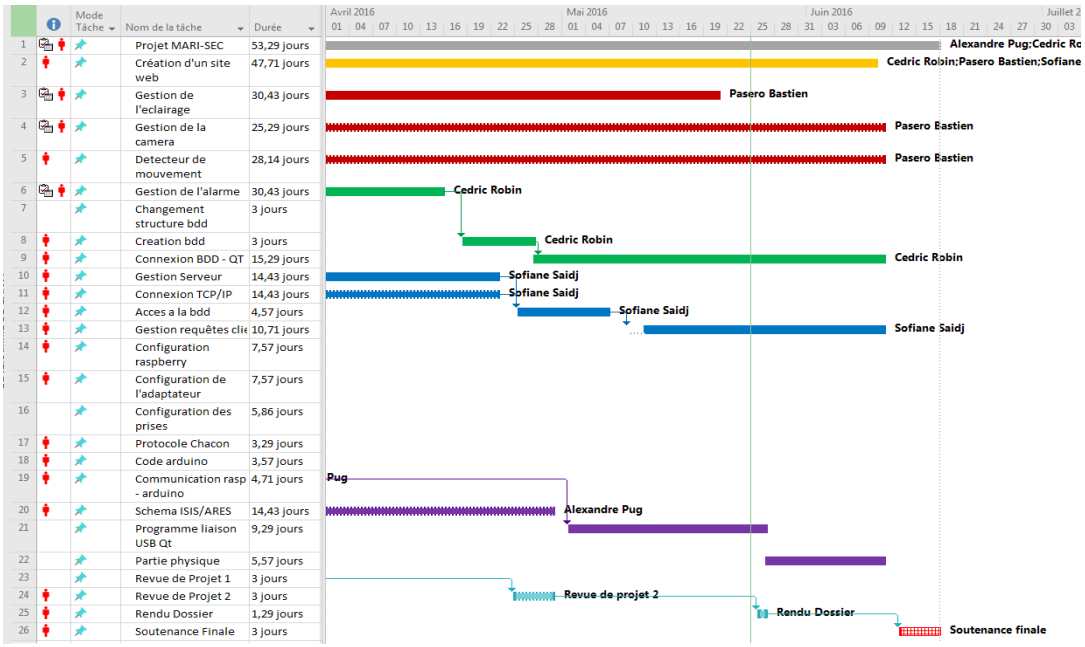




ANNEXE 9 (GANTT réel)



	Mode Tâche	Nom de la tâche	Durée
1		Projet MARI-SEC	53,29 jours
2		Création d'un site web	47,71 jours
3		Gestion de l'eclairage	30,43 jours
4		Gestion de la camera	25,29 jours
5		Detecteur de mouvement	28,14 jours
6		Gestion de l'alarme	30,43 jours
7		Changement structure bdd	3 jours
8		Creation bdd	3 jours
9		Connexion BDD - QT	15,29 jours
10		Gestion Serveur	14,43 jours
11		Connexion TCP/IP	14,43 jours
12		Acces a la bdd	4,57 jours
13		Gestion requêtes clie	10,71 jours
14		Configuration raspberry	7,57 jours
15		Configuration de l'adaptateur	7,57 jours
16		Configuration des prises	5,86 jours
17		Protocole Chacon	3,29 jours
18		Code arduino	3,57 jours
19		Communication rasp - arduino	4,71 jours
20		Schema ISIS/ARES	14,43 jours
21		Programme liaison USB Qt	9,29 jours
22		Partie physique	5,57 jours
23		Revue de Projet 1	3 jours
24		Revue de Projet 2	3 jours
25		Rendu Dossier	1,29 jours
26		Soutenance Finale	3 jours



	Mode Tâche	Nom de la tâche	Durée
1		Projet MARI-SEC	53,29 jours
2		Création d'un site web	47,71 jours
3		Gestion de l'eclairage	30,43 jours
4		Gestion de la camera	25,29 jours
5		Detecteur de mouvement	28,14 jours
6		Gestion de l'alarme	30,43 jours
7		Changement structure bdd	3 jours
8		Creation bdd	3 jours
9		Connexion BDD - QT	15,29 jours
10		Gestion Serveur	14,43 jours
11		Connexion TCP/IP	14,43 jours
12		Acces a la bdd	4,57 jours
13		Gestion requêtes clie	10,71 jours
14		Configuration raspberry	7,57 jours
15		Configuration de l'adaptateur	7,57 jours
16		Configuration des prises	5,86 jours
17		Protocole Chacon	3,29 jours
18		Code arduino	3,57 jours
19		Communication rasp - arduino	4,71 jours
20		Schema ISIS/ARES	14,43 jours
21		Programme liaison USB Qt	9,29 jours
22		Partie physique	5,57 jours
23		Revue de Projet 1	3 jours
24		Revue de Projet 2	3 jours
25		Rendu Dossier	1,29 jours
26		Soutenance Finale	3 jours

ANNEXE 10 (Journal de Bord : PASERO)

Lundi 11/01/2016 :
Après-midi : Mise en place du projet, discussion, brainstroming, mise en situation.
(réunion sans fiche de réunion)

Mercredi 13/01/2016 :
Après-midi : Étude des diagrammes

Vendredi 15/01/2016 :
Matin : Configuration Raspberry.
Après-midi : Téléchargement et installation de Microsoft Project (+licence)

Lundi 18/01/2016 :
Après-midi : Commencement du Gantt prévisionnel

Mercredi 20/01/2016 :
Après-midi : Réunion pour accord du Gantt

Vendredi 22/01/2016 :
Matin : Configuration Raspberry.
Après-midi : Finalisation du Gantt

Lundi 25/01/2016 :
Après-midi : Étude de la caméra

Mercredi 27/01/2016 :
Après-midi : Journal de Bord

Vendredi 29/01/2016 :
Matin : Configuration Raspberry.
Après-midi : Problème avec la caméra

Lundi 01/02/2016 :
Après-midi : Constitution d'un plan pour le site web

Mercredi 03/02/2016 :
Après-midi : Début de Site web

Vendredi 05/02/2016 :
Matin : Configuration Raspberry.
Après-midi : Site web : Tableau de bord

08/02/2016 au 19/02/2016 : Vacances d'hiver

Lundi 22/02/2016 :
Après-midi : Site web : Acceuil + Sources

Mercredi 24/02/2016 :
Après-midi : Site web : Contact

ANNEXE 11 (Journal de Bord : PASERO)

Vendredi 26/02/2016 :
Matin : Commencement du Dossier de revue de projet 1.
Après-midi : Site web : Contact

Lundi 29/02/2016 :
Après-midi : Constitution du dossier + Diaporama

Mercredi 02/03/2016 :
Après-midi : Dossier + Diaporama

Vendredi 04/03/2016 :
Matin : Finalisation du dossier + Diaporama
Après-midi : Revue de Projet 1

Lundi 07/03/2016 :
Après-midi : Intégration du site de la caméra dans le site web.

Mercredi 09/03/2016 :
Après-midi : Intégration du site de la caméra dans le site web.

Vendredi 11/03/2016 :
Matin : Installation QT-creator sur Raspberry PI
Après-midi : Installation QT-creator sur Raspberry PI

Lundi 14/03/2016 : *Après-midi :* Programme liaison série

Mercredi 16/03/2016 :
Après-midi : Apprendre à utilisé QSerialPortInfo

Vendredi 18/03/2016 :
Matin : Apprentissage QSerialPortInfo
Après-midi : Fabrication IHM liaison série

Lundi 21/03/2016 :
Après-midi : Utilisation QSerialPortInfo

Mercredi 23/03/2016 :
Après-midi : Utilisation QSerialPortInfo

Vendredi 25/03/2016 :
Matin : Apprentissage QSerialPort
Après-midi : Apprentissage QSerialPort

Lundi 28/03/2016 :
Après-midi : Apprentissage QSerialPort

Mercredi 30/03/2016 :
*Après-midi :*Utilisation QSerialPort

01/04/2016 au 15/04/2016 : Vacances de printemps

Lundi 18/04/2016 :
Après-midi : Préparation dossier revue 2

ANNEXE 12 (Journal de Bord : PASERO)

Mercredi 20/04/2016 :
Après-midi : Préparation dossier + diapo revue 2

Vendredi 22/04/2016 :
Matin : Préparation diaporama revue 2
Après-midi :Préparation diaporama revue 2

Lundi 25/04/2016 :
Après-midi : Revue de projet 2

Mercredi 27/04/2016 :
Après-midi : Programme liaison série

Vendredi 29/04/2016 :
Matin : Programme liaison série
Après-midi : Programme liaison série

Lundi 02/05/2016 :
Après-midi : Programme liaison série partie BDD

Mercredi 04/05/2016 :
Après-midi :Programme liaison série partie BDD

Après-midi : Détection mouvement

Lundi 09/05/2016 :
Après-midi : Détection mouvement

Mercredi 11/05/2016 :
Après-midi :Détection mouvement

Vendredi 13/05/2016 :
Matin : Détection mouvement
Après-midi :Détection mouvement

Lundi 16/05/2016 :
Après-midi : Construction du dossier final.

Mercredi 18/05/2016 :
Après-midi :Construction du dossier final.

Vendredi 20/05/2016 :
Matin : Construction du dossier final.
Après-midi :Construction du dossier final.

Lundi 23/05/2016 :
Après-midi : Construction du dossier final + rassemblement codes

ANNEXE 13 (Journal de Bord : ROBIN)

Lundi 11/01/2016 :
Mise en place du projet, discussion, brainstroming, mise en situation. (réunion sans fiche de réunion)
Mercredi 13/01/2016 :
Etude des diagrammes
Vendredi 15/01/2016 :
Configuration Raspberry.
Téléchargement et installation de microsoft project (+licence)
Lundi 18/01/2016 :
Commencement du Gantt prévisionnel
Mercredi 20/01/2016 :
Réunion pour accord du Gantt
Vendredi 22/01/2016 :
Configuration Raspberry.
Lundi 25/01/2016 :
Étude d'un systeme d'alarme fonctionnel
Mercredi 27/01/2016 :
Étude d'un systeme d'alarme fonctionnel
Vendredi 29/01/2016 :
Configuration Raspberry.
Installation LAMP
Lundi 01/02/2016 :
configuration base de données
Mercredi 03/02/2016 :
configuration base de données
Vendredi 05/02/2016 :
Configuration Raspberry
configuration base de données
Lundi 22/02/2016 :
Squelettes communication base de données , QtCreator
Mercredi 24/02/2016 :
Squelettes communication base de données , QtCreator

ANNEXE 14 (Journal de Bord : ROBIN)

Vendredi 26/20/2016 :
Squelettes communication base de données , QtCreator
Lundi 29/02/2016 :
Constitution du dossier + Diaporama
Mercredi 02/03/2016 :
Constitution du dossier + Diaporama
Vendredi 04/03/2016 :
Finalisation du dossier + Diaporama
Revue de Projet 1
Lundi 07/03/2016 :
Squelettes de l'alarme virtuel basé sur le fonctionnement d'une alarme physique
Mercredi 09/03/2016 :
Squelettes de l'alarme virtuel basé sur le fonctionnement d'une alarme physique
Vendredi 11/03/2016 :
Squelettes de l'alarme virtuel basé sur le fonctionnement d'une alarme physique
Lundi 14/03/2016 :
Squelettes de l'alarme virtuel basé sur le fonctionnement d'une alarme physique
Mercredi 16/03/2016 :
Squelettes de l'alarme virtuel basé sur le fonctionnement d'une alarme physique
vendredi 18/03/2016 :
changements structure base de données
lundi 21/03/2016 :
changements structure base de données
mercredi 23/03/2016
intégration communication base de données , alarmes
vendredi 25 /03/2016 :
intégration communication base de données , alarmes
lundi 28/03/2016 :
intégration communication base de données , alarmes
mercredi 30/03/2016 :
intégration communication base de données , alarmes

ANNEXE 15 (Journal de Bord : ROBIN)

vendredi 02/04/2016 :
période de test du programme
lundi 18/04/2016 :
période de test du programme
préparation revue 2
mercredi 20/04/2016 :
prépartion revue 2
vendredi 22/04/2016 :
préparation revue 2
lundi 25/04/2016 :
présentation revue 2
mercredi 27/04/2016 :
changements de structure de la base de données
vendredi 29/04/2016 :
changements de structure de la base de données
lundi 02/05/2016 :
mise en ordre du code et commentaire
mardi 04/05/2016
mise en ordre du code et commentaire
vendredi 06/05/2016
mise en ordre du code et commentaire
lundi 09/05/2016
amélioration ihm
mercredi 11/05/2016
création journal d'événements
vendredi 13/05/2016
création du journal d'événements
Du lundi 16/05/2016 au Mercredi 25/05/2016
dossier revue finale

ANNEXE 16 (code fourni par la caméra)

```
<SCRIPT LANGUAGE="JavaScript">

    var BaseURL = "http://192.168.2.9/"; //permet d'acceder au site de la camera

    var DisplayWidth = "704";    //
    var DisplayHeight = "576";   // On définit la taille du cadre pour l'image

    var File = "axis-cgi/mjpg/video.cgi?resolution=2CIFEXP&compression=90&color=1&clock=1&date=1&text=1&textstring=Camera exterieure";
                // C'est le chemin où l'on va hercher l'image seule de la camera

    var output = "";

    // Dans le cas ou le navigateur est Internet Explorer
    // Portion de code fourni par le site de la camera AXIS

    if ((navigator.appName == "Microsoft Internet Explorer") && (navigator.platform != "MacPPC") && (navigator.platform != "Mac68k"))
    {
        output  = '<OBJECT ID="Player" width='
        output += DisplayWidth;
        output += ' height=';
        output += DisplayHeight;
        output += ' CLASSID="CLSID:DE625294-70E6-45ED-B895-CFFA13AEB044" ';
        output += 'CODEBASE="';
        output += BaseURL;
        output += 'activex/AMC.cab#version=3,32,19,0">';
        output += '<PARAM NAME="MediaURL" VALUE="';
        output += BaseURL;
        output += File + '">';
        output += '<param name="MediaType" value="mjpeg-unicast">';
        output += '<param name="ShowStatusBar" value="0">';
        output += '<param name="ShowToolbar" value="0">';
        output += '<param name="AutoStart" value="1">';
        output += '<param name="StretchToFit" value="1">';
        // Remove the '//' for the ptz settings below to use the code for click-in-image.
        //  output += '<param name="PTZControlURL" value="';
        //  output += BaseURL;
        //  output += '/axis-cgi/com/ptz.cgi?camera=1">';
        //  output += '<param name="UIMode" value="ptz-relative">'; // or "ptz-absolute"
        output += '<BR><B>Axis Media Control</B><BR>';
        output += 'The AXIS Media Control, which enables you ';
        output += 'to view live image streams in Microsoft Internet';
        output += ' Explorer, could not be registered on your computer.';
        output += '<BR></OBJECT>';
    }
    else
    {
        // If not IE for Windows use the browser itself to display
        theDate = new Date();
        output  = '<IMG SRC="';
        output += BaseURL;
        output += File;
        output += '&dummy=' + theDate.getTime().toString(10);
        output += '" HEIGHT="';
        output += DisplayHeight;
        output += '" WIDTH="';
        output += DisplayWidth;
        output += '" ALT="Camera Image">';
    }
    document.write(output);
    document.Player.ToolbarConfiguration = "play,+snapshot,+fullscreen"
    // document.Player.UIMode = "MDConfig";
    // document.Player.MotionConfigURL = "/axis-cgi/operator/param.cgi?ImageSource=0"
    // document.Player.MotionDataURL = "/axis-cgi/motion/motiondata.cgi";
</SCRIPT>
```

ANNEXE 17 (code JAVA)

```
import java.applet.*;  //Importation de toutes les librairies

public class detection extends Applet {
    // Méthode appelée par le navigateur lorsque l'applet est chargée
    private static final long serialVersionUID = 1L;
     static Mat imag = null;

        public static void main(String[] args) {

            JFrame jframe = new JFrame("HUMAN MOTION DETECTOR FPS");
            jframe.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            JLabel vidpanel = new JLabel();
            jframe.setContentPane(vidpanel);
            jframe.setSize(640, 480);
            jframe.setVisible(true);

            Mat frame = new Mat();
            Size sz = new Size(640, 480);
            VideoCapture camera = new VideoCapture("http://192.168.2.9axis-cgi/mjpg/video.cgi");

            while (true) {
                if (camera.read(frame)) {
                    Imgproc.resize(frame, frame, sz);
                    ImageIcon image = new ImageIcon(Mat2bufferedImage(frame));
                    vidpanel.setIcon(image);
                    vidpanel.repaint();

                }
            }
        }

        public static BufferedImage Mat2bufferedImage(Mat image) {
            MatOfByte bytemat = new MatOfByte();
            Imgcodecs.imencode(".jpg", image, bytemat);
            byte[] bytes = bytemat.toArray();
            InputStream in = new ByteArrayInputStream(bytes);
            BufferedImage img = null;
            try {
                img = ImageIO.read(in);
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            return img;
        }
```

ANNEXE 18 (code Python)

```
#/usr/bin/python2.7
#

# -*- coding: utf-8 -*-


import sys
import os
import imps
import cv2

video="http://192.168.2.9/axis-cgi/mjpg/video.cgi"
capture =cv2.VideoCapture(video)

cv2.namedWindow('Video Stream', 1 )

while True:
 #Capture la première frame
  frame = cv2.read(capture)

if frame is None:
    break   #si la première frame n'est pas récupérée on ferme le programme

  else:
    #detect(frame)
    cv2.ShowImage('Video Stream', frame)

  if k == 0x1b: # echap
    print 'Fermeture programme ...'
    break
```

ANNEXE 19 (recette curative)

Fiche de maintenance curative

En cas de panne sur une ou plusieurs parties du systèmes il y a une courte liste de paramètres à vérifier pouvant résoudre la majorité des problèmes susceptibles de survenir .

En cas de panne :

- vérifier que la carte Raspberry PI est bien alimenté.
- vérifier que le PC Serveur est bien alimenté.
- vérifier que la caméra est bien alimenté et connectée au réseau.
- vérifier la connectivité au réseau de la carte Raspberry PI.
- vérifier le câblage de l'adaptateur USB – I2C.
- vérifier que le serveur TCP est lancé (le lancer dans le cas contraire).
- vérifier que le client TCP est lancé (le lancer dans le cas contraire) .
- vérifier que le serveur SQL est lancé (le lancer dans le cas contraire).
- vérifier que le serveur web est lancé (le lancer dans le cas contraire).
- vérifier que le site web est accessible.
- vérifier l'adresse réseaux de la Raspberry PI.
- vérifier les paramètres de connexion de l'alarme.
- vérifier les identifiants utilisé lors de la connexion.
  (que ce soit sur la base de données ou le site web).

-s'assurer de la connexion entre l'alarme et la base de données.
-vérifier l'état des lumières (hors services ou abîmé : les remplacer).