



جامعة محمد الول المدرسة الوطنية للعلوم التطبيقية

وجدة

**UNIVERSITE MOHAMMED PREMIER**

***Ecole Nationale des Sciences Appliquées* Oujda**

**Filière Génie Télécommunications & Réseaux**

**Mémoire de Projet de Fin d'Année**

**Spécialité :** Télécommunications & Réseaux

**Options :** Réseaux et Systèmes

Présenté par: Mohammed Amine JABRI

**Asmae HAJJI Soufiane LAMRANI**

**Sujet intitulé:**

**Étude et mise en place d’un système de vidéosurveillance multi-caméras avec la carte RASPBERRY.**

**Membres de jury: Encadré par:**

M.Hamza BARBOUCHA M.Hamza BARBOUCHA

M. Mohammed AISSAOUI

Année universitaire 2016-2017

2

*Dédicace*

*On dedie notre modeste travail à:*

***N****os parents : qui ont œuvré pour notre réussite, de par leur amour, leur soutien, tous les sacrifices consentis et leurs précieux conseils, pour toute leur assistance et leur présence , recoivent à travers ce travail aussi modeste soit-il, l'expression de nos sentiments et de notre éternelle gratitude.*

***N****os professeurs de l'ENSAO qui doivent voir dans ce travail la fierté d’un savoir bien acquis.*

***N****os amis, amies et colègues qui n'ont cessé de nous encourager.*

***A*** *toutes nos familles et à tous ceux qui nous ont aidé de près ou de loin.*

***O****n vous remercie tous.*



3

## Remerciements

**A**près avoir rendu grâce à Dieu le Tout Puissant, nous tenons à remercier vivement tous ceux qui, de près ou de loin ont participé à la réalisation de ce projet de fin d'année.

**A**u terme de ce travail, on tient à exprimer notre plus profonde gratitude à notre cher encadrant **M. Barboucha** pour son suivi, ses conseils, ses orientations, et la disponibilité qu’il nous a accordée pour faire réussir ce travail tout au long de la période du projet.

**O**n exprime toutes nos reconnaissances à **M. Tidhaf** le responsable de la filière Génie des Télécommunications et réseaux de l'ENSA Oujda qui ne cesse de nous créer les conditions nécessaires à l’adaptation et à l’évolution professionnel tout au long de notre formation.

**O**n présente nos sincères remerciements à nos chers professeurs de l’ENSA Oujda qui font beaucoup d’efforts pour nous transmettre leurs connaissances. Vos compétences incontestables ainsi que vos qualités humaines vous valent l'admiration et le respect de tous.

**O**n vous adresse nos sincères remerciements pour votre patience et votre encadrement durant toutes ces années.

**N**os remerciements vont à toute personne qu'on a contacté durant notre projet, auprès de laquelle on a trouvé l'accueil chaleureux, l'aide et l'assistance dont on avait besoin.

**O**n adresse aussi nos vifs remerciements aux membres du jury pour avoir bien voulu examiner et juger ce travail.



#### Résumé:

La vidéosurveillance est devenue indispensable de nos jours, elle s’impose dans une multitude de secteurs d’activité : dans les hôpitaux, les banques, les sites de production... Elle sollicite, par ailleurs, de plus en plus le contrôle à distance pour la maîtrise des coûts, l'amélioration de la sécurité, la facilité d'utilisation et de la commodité de l'accès et fournir de nouvelles applications.

C'est suite au grand pas qu'a réalisé la technologie récente des objets connectés, qu'on a opté pour le sujet d'étude et la mise en place d’un système de vidéosurveillance multi- caméras avec la carte **RASPBERRY**.

La conception de ce travail se définit par la construction d'un système vidéosurveillance multi-caméras avec la carte Raspberry contrôlée à distance, permettant aux clients de tirer profit de l'internet, pour fournir un streaming vidéo continu sur ce dernier.

Ce document présente un récapitulatif des démarches réalisées durant les différentes phases de notre travail, afin qu'il soit accompli.

Notre travail est scindé en deux grandes parties. La première partie, comprend le matériel utilisé, ainsi que les programmes développés, pour la connexion du serveur avec le client. Ces programmes consistent à faire connecter, collecter et enregistrer les adresses IP dynamiques des objets (caméras), et simplifie l'accès du client à son objet souhaité.

La deuxième partie s'appuie sur la construction du système vidéosurveillance qui comprend le serveur collecteur et diffuseur d'images.

#### Abstract:

The video surveillance becames essential nowadays, it is imperative in a multitude of business sectors: in hospitals, banks, production sites ... besides the remote control, She requests to control the costs, to improve the safety, the ease of use and the convenience of the access and supply new applications.

The recent technology of the connected objects has token a big step in the last years, so we opted for the subject of study and the implementation of a multi-camera video surveillance system With the RASPBERRY card.

The conception of this work settles on the video surveillance multi-cameras with the card Raspberry controlled at distance (remotely), allowing the customers to benefit from the internet, to supply a continuous video streaming.

This document presents a summary on the steps realized during the various phases of our work, so that it is achieved.

Our work is split into two big parts . The first part, includes the used material (equipment), as well as the developed programs, for the connection of the server with the customer.

These programs consist in making connect, collect and register the dynamic IP addresses of objects (cameras), and simplify the access of the customer to its desired object.

The second part leans on the construction of the system video surveillance which includes the collection server and the diffuser of images.

# Table des matieres :

Dédicace 2

[Remerciements 3](#_TOC_250016)

[Résumé 4](#_TOC_250015)

[Abstract 5](#_TOC_250014)

Table des matières 6

[Liste des figures 8](#_TOC_250013)

[Introduction. 9](#_TOC_250012)

**Chapitre I**: les outils matériels et logiciels 10

* 1. Présentation de la carte Raspberry 11
     1. [Qu'est ce que c'est? 11](#_TOC_250011)
     2. [Spécifications techniques 11](#_TOC_250010)
     3. [Ce qu'on peut faire 12](#_TOC_250009)
     4. Raspberry Pi caméra v2. 12
     5. Système d'exploitation. 14
     6. Installation du Rasbian. 14
  2. Utilisation de Raspberry. 15
     1. [Premier démarrage 15](#_TOC_250008)
     2. Configuration. 15
  3. Serveur IP fixe 18

**Chapitre II**: collection des IP dynamiques des objets 19

* 1. Présentation du modèle serveur-client 19
  2. Serveur de collection des IP et application web. 20
     1. [Serveur de collection des IP 20](#_TOC_250007)
     2. L'utilisation de l'API Mysql 21
     3. [Interface web pour la redirection des utilisateurs 24](#_TOC_250006)
  3. Les objets et le programme (monIP). 24
     1. Structure ou la forme du message du client 25
     2. [Client DAEMON 26](#_TOC_250005)
     3. Configuration des routeurs 26

**Chapitre III** : présentation du système et pré requis 28

* 1. Le principe global du système 29
  2. L'utilité des photos vis-à-vis de la vidéo. 30
  3. présentation du contrôleur. 31

Chapitre IV : construction du système de surveillance 33

* 1. Prendre et envoyer les photos au contrôleur. 33
  2. Concentrateur.php. 35
     1. [stockage 35](#_TOC_250004)
     2. [diffusion. 35](#_TOC_250003)
  3. Visionnement des images 37
     1. [Authentification. 37](#_TOC_250002)
     2. [index.php. 37](#_TOC_250001)

[Conclusion. 40](#_TOC_250000)

Bibliographie et Webographie 41

### Liste Des figures

*Figure 1 : La carte Rapverry Pi3 modèle B 12*

*Figure 2 : Raspberry Module camera 13*

*Figure 3 : Connexion au SHELL du Raspberry par ssh 16*

*Figure 4 : Image capturée par le module camera 17*

*Figure 5 : Configuration du serveur. 18*

*Figure 6 : Modèle Serveur-Client 19*

*Figure 7 : Foncionnement du serveur de collections des IP. 20*

*Figure 8 : Tableau des objets. 21*

*Figure 9 : command mysql\_config 22*

*Figure 10 : compliation première étape 22*

*Figure 11 : compilation deuxière étape 22*

*Figure 12 : les objets éxistants dans la BBD 24*

*Figure 13 : la forme du message de l'objet pour enregistrer son IP. 25*

*Figure 14: Principe du port forwarding 27*

*Figure 15 : Configuration du routeur pour le port forwarding 27*

*Figure 16 : Raspberry de contrôle 30*

*Figure 17 : Figure 17 : Image origine (270kB) 36*

*Figure 18 : Image compressée (Taille : 3.2 kB) 36*

*Figure 19 : Authentification d'utilisateurs. 37*

*Figure 20 : photo prise par cam1 38*

*Figure 21 : photo prise par cam2 39*

**Introduction:**

**Introduction**

La vidéosurveillance est un champs d'application illimité, de la vie quotidienne en passant par le monde du travail au domaine public, nécessitent de plus en plus une sécurité fiable et efficace, d'où sa contribution en grande partie à la protection et à la sécurité, protection, sécurité et facilité d'accès et de commodité à distance

Le projet « étude et la mise en place d’un système de vidéosurveillance multi-caméras avec la carte **RASPBERRY**. » se repère dans ce contexte.

Il consiste d'une part, de faire connecter les objets (caméras), ayant des adresses IP instantanées, avec le serveur, ce dernier qui sera impérativement hébergé dans une machine d'adresse IP fixe. Ainsi, il présente une facilité d'accès des clients pour les différentes demandes des différents objets (caméras) sollicités. D'une autre part, ce projet présente une solution efficace, pour qu'un serveur puisse fournir un streaming vidéo continu sur Internet.

Ce rapport présente l'évolution de notre projet de fin d'année et une description bien détaillée sur les différentes étapes réalisées. Il consiste, dans la première partie, à présenter de façon générale la carte Raspberry Pi et sa commodité, et la caméra utilisée, ainsi, le serveur enregistreur des adresses IP des objets connectés, en intégrant une méthode d'authentification et de chiffrement des clients après l'établissement de la connexion avec ce serveur. Ceci est pour simplifier tout accès aux objets souhaités. Tandis que, la deuxième partie se porte sur la construction du système de surveillance, c'est-à-dire, prendre, collecter, enregistrer et diffuser les images en temps réel.

**Remarque** : Les codes sources de ce projet ainsi que la version numérique de ce rapport se trouve dans le site: <http://tinyurl.com/pfa555>

#### Chapitre I :

**Les outils matériels et logiciels :**

Résumé:

Ces deux premiers chapitres présentent une introduction générale de la carte Raspberry et sa commodité, et elle énonce la première problématique et son objectif. Et finalement, elle expose les démarches suivies pour la réalisation de cet objectif déjà fixé.

Problématique:

Les fournisseurs d'accès ISP procurent à chaque objet sa propre adresse IP, bien que ces adresses ne sont pas statiques, vu qu'à chaque fois que le routeur est rebooté, la dite adresse s'actualise automatiquement, d'où l ' a p p e l l a t i o n d ' a d r e s s e d y n a m i q u e . C e c i p o s e d e s p r o b l è m e s d'**enregistrement des différentes adresses actualisées** et d'**identification instantanée de l'objet par son adresse dynamique**. Ce chapitre propose les différentes démarches pour la résolution de ces deux problématiques indispensables pour la conduite du projet.

1. **– Présentation de la carte Raspberry Pi :**

A – Qu'est-ce que c'est?

Le Raspberry Pi est un ordinateur dont les particularités sont la très petite taille (la taille d'une carte de crédit) et le prix modique (25-30 euros- 600 DH au Maroc ).

Il a été créé par l'anglais David Braben, dans le cadre de sa fondation Raspberry Pi, dans le but d'encourager l'apprentissage de la programmation informatique. Pour la petite histoire,r*aspberry* signifie *framboise* en anglais.

Le Raspberry Pi rappelle quelque peu l'[Arduino](http://arduino.cc/), un circuit imprimé - dont les plans sont publiés sous licence libre - sur lequel se trouve un microcontrôleur programmable et objet fétiche des partisans de l'*open hardware*.

B – Spécifications techniques:

Physiquement, il s'agit d'une carte mère seule avec un processeur ARM, de petite taille (environ la taille d'une carte de crédit ou d'un paquet de cigarette).

Il existe actuellement deux modèles : le modèle A et le modèle B .

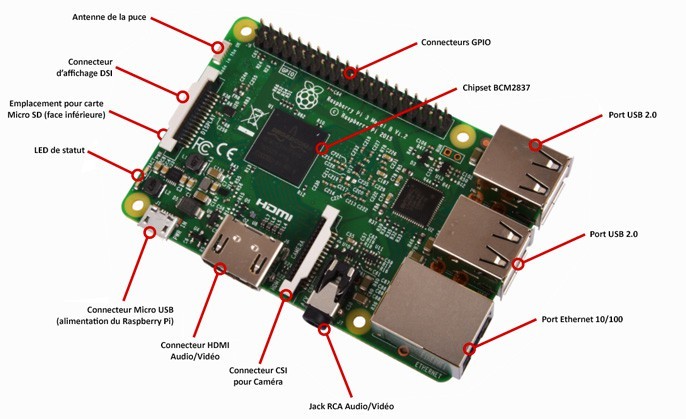


Figure 1 : La carte Rapberry Pi3 modèle B

C – Ce qu'on peut faire :

Même si à la base le Raspberry Pi est un gadget, il a suffisamment de capacités (512 Mo de mémoire, circuit graphique BMC Videocore 4) et de sorties (ports USB, ethernet) pour convenir à différentes utilisations : serveur, pare-feu... Certains l'ont même transformé en [Super Nitendo](http://www.raspberrypi.org/phpBB3/viewtopic.php?f=78&amp;t=20461), en [alarme](http://li0r.wordpress.com/), [émetteur radio FM](http://blog.makezine.com/2012/12/10/raspberry-pi-as-an-fm-transmitter/), en [liseur vocal de texte](http://itp.nyu.edu/%7Ewdl225/work/?p=286) ou carrément en [mini-hélicoptère](http://www.botched.co.uk/picopters-maiden-flight/).

Il pourrait convenir pour petite machine bureautique, mais même s'il permet de lire correctement des vidéos, il n'est pas ultra rapide et surtout certains packages potentiellement utiles (flash...) pour le web ne sont pas forcément bien portés sous ARM.

D – Raspberry Pi camera v2:

Nous allons utiliser la Raspberry Pi camera v2, c'est la plus compatible avec RPI, on peut d'ailleurs utiliser d'autre type de camera avec cette carte (webcam par exemple),

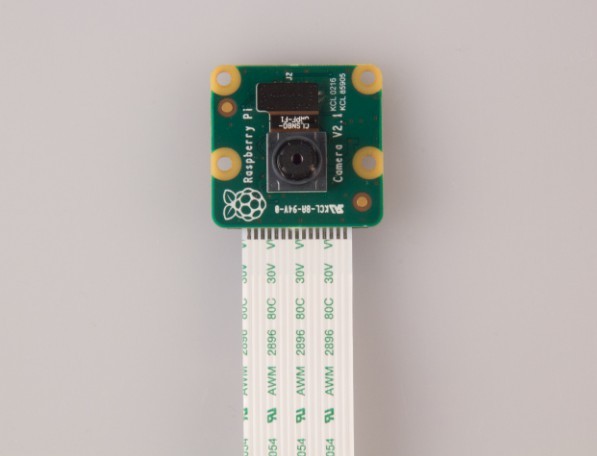


Figure 2 : Module de camera de Raspberry.

Ces principales caractéristiques sont les suivantes:

* Capteur 8 Mégapixels pour Raspberry Pi.
* Dimension du capteur 3280 x 2464 pixels.
* Résolutions vidéo 1080p30, 720p60 et 640x480p90.
* Résolution photo : 3280 x 2464 pixels.
* Dimension du pixel 1,4 µm X 1,4 µm avec technologie OmniBSI.
* Dimension de la partie optique 1/4″.
* Compatible avec toutes les cartes Raspberry.
* elle peut être avec ou sans infra-rouge.

E – Systeme d'exploitation:

La fondation Raspberry Pi recommande d'utiliser [Raspbian](http://www.raspbian.org/), une distribution GNU/Linux optimisée pour le matériel du Raspberry Pi. Raspbian est basée sur Debian, embarquant l'environnement de bureau LXDE et le navigateur web Midori.

D'autres distributions GNU/Linux ont été testées avec succès avec le Raspberry Pi. C'est le cas de Slackware, Arch et Gentoo dans leurs versions destinées aux processeurs ARM. Red Hat recommande d'utiliser [Raspberry Pi Fedora Remix](http://zenit.senecac.on.ca/wiki/index.php/Raspberry_Pi_Fedora_Remix) et les debianeux s'orienteront vers Raspbian.

[Android](http://lea-linux.org/documentations/Android) fonctionne également sur le Raspberry Pi, tout comme FreeBSD et NetBSD.

F – Installation de Raspbian:

**Raspbian** est un système d'exploitation libre et gratuit basé sur Debian GNU/Linux optimisé pour fonctionner sur un [Raspberry Pi](https://fr.wikipedia.org/wiki/Raspberry_Pi).

Raspbian est un [mot-valise](https://fr.wikipedia.org/wiki/Mot-valise) formé par la fusion des mots “Raspberry Pi” et “Debian”.Il s'agit d'une modification de Debian spécifiquement adaptée pour les [systèmes sur une](https://fr.wikipedia.org/wiki/Syst%C3%A8me_sur_une_puce) [puce](https://fr.wikipedia.org/wiki/Syst%C3%A8me_sur_une_puce) de type [ARMv](https://fr.wikipedia.org/wiki/ARM_(soci%C3%A9t%C3%A9))6.

Le système d'exploitation dans une carde SD d'un stockage 8GB au minimum.Tout d'abord on va installer l'image du Raspbian su la carte SD. Donc on commence par télécharger l'image du raspbian en utilisant la commande suivante :

# wget<http://raspbian-france.fr/download/raspbian_latest.zip>

Puis nous allons utiliser la commande « dd », pour copier l’image de Raspbian sur la carte :

# sudo dd bs=1M if=/PATH/OF/IMG of=/dev/mySD

L’écriture de Raspbian sur la carte SD peut être un peu longue.

En suite,on a qu'à brancher la carte SD sur la Raspberry et démarrer cette dernière.

###### – Utilisation de Raspberry Pi :

A- Premier démarrage:

Après avoir installé l'image de raspbian sur la carte SD, nous allons pouvoir allumer notre Raspberry Pi pour la première fois, et installer Raspbian .

Dans un premier temps, on branche la Raspberry à l’écran et au clavier, sans l’alimenter, et on insère dessus la carte SD.

Puis, on branche l’alimentation de la Raspberry. Le premier démarrage peut-être un peu long, car lors de celui-ci, la Raspberry va installer le système Raspbian.

Une fois le système démarré, il va (peut-être) demander de s'identifier. Lors du premier démarrage le login par défaut est « pi », et le password est « raspberry ».

B- Configurations:

**Changement du mot de passe de l'utilisateur pi :**

Tout d'abord on change le mot passe par défault :

$ sudo passwd pi

On fait entrer le nouveau mot de passe Activation du service SSH :

Pour notre utilisation nous avons besoin seulement du shell de raspbian, et un acces à distance depuis des machines linux. Donc pour activer le service SSH on tape la commande suivante :

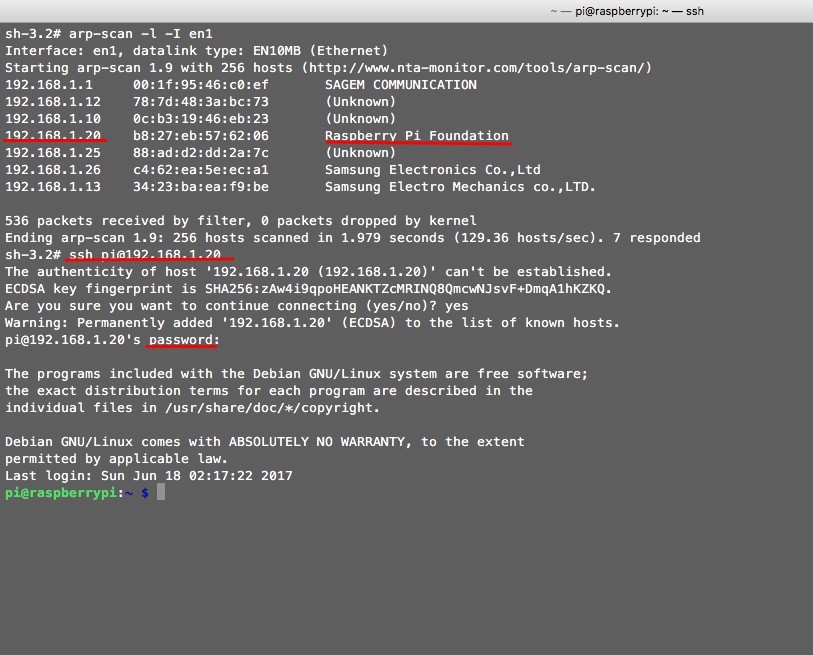
$ sudo raspi-config

Puis on selectionne l'option «Interfacing Options» puis « ssh » puis « enabled » et

« OK ». Maintenant le service est activé sur le port 22(par default). Pour acceder à notre

raspberry pi depuis une machine linux il suffit de tapper la command suivante

# ssh pi@<addresse\_ip\_de\_RPI>

puis faire entrer le mot mot de passe, et voilà nous somme dans le shell de la RPI

*Figure 3 : Connexion au SHELL du Raspberry par ssh.*

**Activation du support de la caméra dans Raspbian:**

de la même manière que pour activer le service ssh on, on tape la commande:

$ sudo raspi-config

on sélectionne l'option «Interfacing Options» puis «camera» ensuite «enabled» et « OK ». Maintenant on vérifie si notre camera marche bien en tapant la command qui capture une photo:

$ raspistill -o image.jpg

Le résultat:

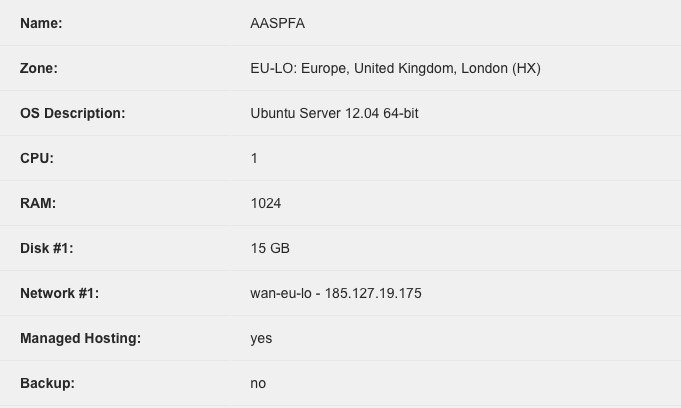


*Figure 4 : Image capturée par le module camera.*

###### – Serveur à IP fixe:

Pour la construction de serveur de collection des IP des objets (ceci sera détaillé dans le chapitre suivant), on va se servir d'un serveur linux (ubuntu server) ayant une adresse IP fixe (c'est obligatoire). Pour notre utilisation limitée, on va tirer profit d'un VPS(Virtual Private Server), hébergé dans le cloud de la société vpsserver.com dont les deux premiers mois sont gratuits.

La configuration de notre serveur est comme suit:



*Figure 5 : Configuration du serveur.*

**ces ressources sont largement suffisantes pour notre utilisation.**

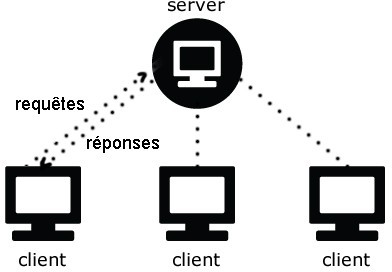
**Chapitre II**

**Collection des IP dynamiques des objets:**

1. **– Présentation du Modele serveur/client :**

L'environnement client-serveur est un mode de communication entre plusieurs programmes à travers un réseau: l'un, qualifié de [client](https://fr.wikipedia.org/wiki/Client_(informatique)), envoie des requêtes; l'autre ou les autres, qualifiés de [serveurs](https://fr.wikipedia.org/wiki/Serveur_informatique), attendent ensuite répondent aux requêtes des clients

Par extension, le client désigne également l'[ordinateur](https://fr.wikipedia.org/wiki/Ordinateur) ou la [machine virtuelle](https://fr.wikipedia.org/wiki/Machine_virtuelle) sur lequel est exécuté le logiciel client, et le serveur, l'ordinateur ou la machine virtuelle sur lequel est exécuté le logiciel serveur.



*Figure 6 : Modèle Serveur-Client.*

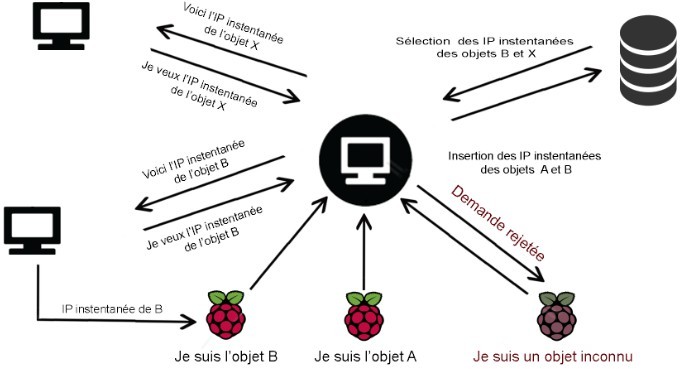
Nous avons développé deux programmes (programme serveur et programme client), le serveur attend les connexions des objets, chaque objet possède un ID et un mot de

passe, après l'authentification le serveur enregistre les IP dynamiques de ces objets dans une base de données, ces objets sont programmés pour qu'ils demandent la connexion auprès du serveur dans chaque intervalle du temps.

###### – Serveur de collection IP et l'application WEB :

A – Serveur de collection des IP :

Notre serveur est hébergé dans une machine dont l'adresse IP est fixe, lorsqu'un utilisateur désire consulter un objet, il n'a qu'à demander à ce serveur de le rediriger vers cet objet souhaité. De ce fait, notre machine possède un serveur web qui héberge une application d'authentification des utilisateurs et fait la redirection vers l'objet désiré.



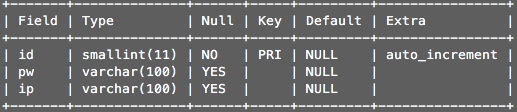
*Figure 7 : Foncionnement du serveur de collections des IP.*

Avec le langage C , nous avons programmé le serveur de collection qui va écouter sur un port (on a choisi le port 5000 qui n'est pas inclus dans l'intervalle 0-1023 des ports standards), il attend une demande de connexion des clients, puis, il établie cette connexion. Pour ce faire, on utilise le protocole TCP qui est orienté connexion.

Une fois la connexion est établie, le serveur commence à recevoir les messages de

clients, contenants le ID, le mot de passe et la clé de déchiffrement pour déchiffrer le mot de passe. **Pour le chiffrement on avait écrit un code en C qui sert à chiffrer la conversation, utilisant le méthode de chiffrement XOR par bloc.**

Une fois le message est reçu, puis déchiffré, le programme authentifie l'objet en consultant une base de données qui contient un tableau appelé «objets» contenant les champs suivants:



*Figure 8 : Tableau des objets.*

* id : l'identifiant de l'objet.
* pw : le mot de passe de l'objet.
* Ip : l'adresse IP dynamique de l'objet après sa dernière connexion.

B – Utilisation de l'API Mysql:

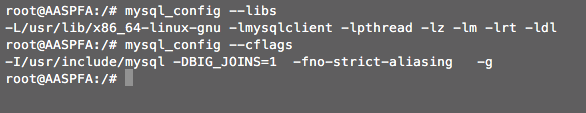
l'API Mysql a pour objectif de se connecter à une base de données et de faire des requêtes SQL depuis un programme *C.*

Pour créer un programme d'interaction avec les bases de données gérées par Mysql, on aura besoin d'une bibliothèque client Mysql, ainsi que des fichiers d'en-têtes. Ces fichiers sont déjà installés avec l'installation du serveur Mysql, et localisés successivement dans des répertoires :

“/usr/local/lib/mysql” et “ /usr/local/include/mysql”

Pour informer le compilateur gcc l'emplacement des fichiers d'en-têtes, on utilise l'option -I suivie par un argument qui est le chemin du répertoire contenant ces fichiers. On fait de même pour la bibliothèque client avec l'option -L .

la commande “mysql\_config –cflags” sert à localiser le chemin du répertoire des fichiers d'en-têtes ainsi que d'autres options pour la compilation. Et la commande “mysql\_config –libs” pour localiser le chemin du répertoire de la bibliothèque de client.



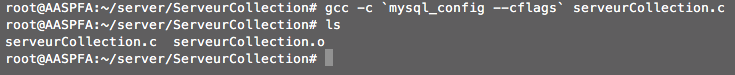
*Figure 9 : command mysql\_config.*

Pour la compilation on va passer par deux étapes :

-Première étape: consiste à la commande génératrice d'un fichier objet.

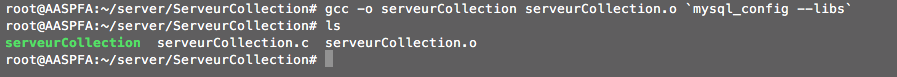
-Deuxième étape: consiste à la commande génératrice du fichier exécutable.

*# gcc -c `mysql\_config --cflags` serveurCollection.c*



*Figure 10 : compilation première étape .*

*# gcc -o ServeurCollection serveurCollection.o `mysql\_config --libs`*



*Figure 11 : compilation deuxière étape .*

enfin, on obtient le fichier exécutable qui offre le service de collection des IP lors de son exécution. Dans ce qui suit, on va montrer comment transformer le programme *serveurCollection* en service.

**Pour réaliser un service, tout se passe dans ces dossiers : /etc/init.d/ et /usr/bin/**

pour transformer le programme en service on va passer par les étapes suivantes:

Tout d'abord il faut créer un petit script en shell qui lance le serveur (on va lui donner le même nom que notre programme *serveurCollection*), sur Linux le script existe déjà dans le fichier /etc/init.d/skeleton, et on a qu'à changer quelques informations concernant notre service :

*# touch serveurCollection*

*# cp /etc/init.d/skeleton ./serveurCollection # nano ./serveurCollection*

PATH=/usr/sbin:/usr/bin:/sbin:/bin NAME=*serveurCollection* DAEMON=/usr/bin/$NAME DAEMON\_ARGS="--options args" PIDFILE=/var/run/$NAME.pid SCRIPTNAME=/etc/init.d/$NAME

On a précisé le nom de notre service et laissé les autres informations telles qu'elles sont. On n'a pas besoin de modifier *DAEMON\_ARGS* car notre programme ne prend aucun argument.

Maintenant, on doit copier le programme serveurCollection dans les répertoires

/usr/sbin , puis le script dans le répertoire */etc/init.d/ :*

*# cp ./server/serveurCollection /usr/bin/serveurCollection # cp ./serveurCollection /etc/init.d/serveurCollection*

On doit rendre le script exécutable :

*# chmod +x /etc/init.d/serveurCollection*

Le programme qui va gérer notre service est « update-rc.d » Il crée plusieurs liens depuis

/etc/rc0.d/serveurCollection vers /etc/init.d/serveurCollection.

Il faut « enregistrer » le script:

*#* update-rc.d launcher defaults

**Et voila notre programme est devenu un service .**

C – Interface web pour la redirection des utilisateurs :

Avec les outils de développement Web, on développe une application de redirection des utilisateurs vers les objets souhaités. Elle consiste à consulter la base de données, sélectionner les objets et fournir leurs IP instantanées.



*Figure 12 : les objets éxistants dans la BBD.*

Nous n'avons que deux objets dans la base de données, en cliquant sur l'image de l'objet, on est redirigé vers son adresse IP.

###### – Les objets et le programme ( monIP ):

Le programme **monIP** est un programme qui sert à authentifier les objets (les Cartes Raspberry Pi) au serveur collecteur des IP, après cette authentification, son adresse sera enregistrée dans le serveur .

Ce programme est un DAEMON qui est exécuté une fois que la machine sera lancée. Si jamais le programme s'arrête, il sera relancé automatiquement dans quelques secondes. Il fonctionne comme suit:

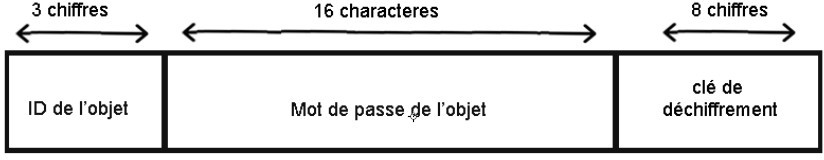
* chaque 5 secondes il génère une clé de chiffrement aléatoire et différente.
* Il chiffre le mot de passe de l'objet.
* Il crée le message d'authentification avec la forme ci-dessous.
* il ouvre une «socket», demande la connexion avec le serveur de collection des IP.
* Il envoie son message d'authentification sur la «socket».
* Il ferme la «socket», attend 5 secondes, et refait la même procédure.

**Remarque:** une fois l'objet est connecté à internet, le serveur de collection reçoit son message et enregistre son IP. **monIP est un autorun**.

A – La structure ou la forme du message du client:

**Remarque**: la forme du message est statique et trop limitée vu le besoin limité dans ce projet.

La forme du message envoyé par le client (l'objet) :



*Figure 13 : la forme du message de l'objet pour enregistrer son IP.*

L'ID de chaque objet est constitué de 3 chiffres, un mot de passe contenant 16 caractères et la clé qui est générée d'une façon pseudo aléatoire est constituée de 8 chiffres .

B – Client DAEMON:

DAEMONTOOLS est une collection des outils pour la gestion des services UNIX. Qui sert à démarrer le programme client automatiquement, et le redémarrer si ce programme est tué.

*# sudo apt-get install daemontools*

On commence par démarrer le DAEMONTOOLS en rajoutant la ligne :

csh -cf '/command/svscanboot &'

dans la fin du fichier “/etc/rc.local.”

Ensuite on va créer un dossier contenant un lien symbolique pointant sur notre programme client:

*# mkdir ~/runApp*

*# ln -s ~/monIP ~/runApp/run*

Enfin on édite le fichier crontab en rajoutant à sa fin cette ligne

@reboot supervise /home/pi/monIP

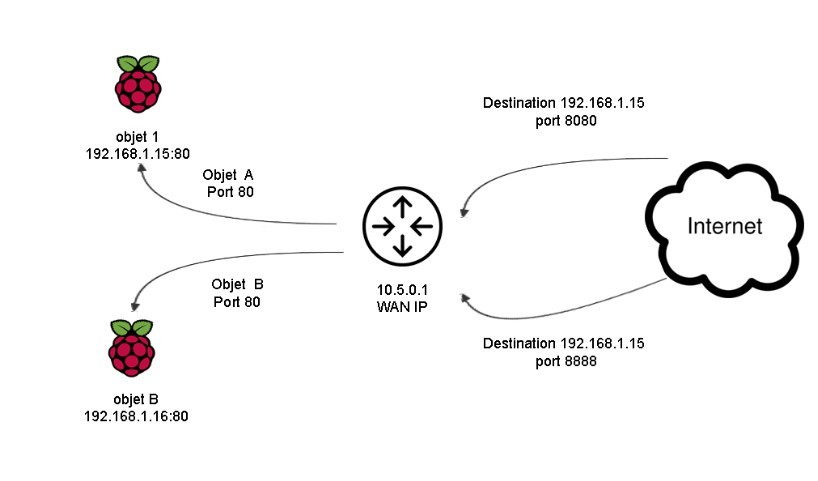
Et voila notre programme est autorun, et se relance directement s'il est tué.

C – configuration du routeur:

L'utilisateur cherche toujours à accéder aux objets. Ces derniers qui sont généralement, dans des réseaux LAN et qui ont un accès à l'internet moyennant un routeur, ne sont pas détectables depuis l'extérieur du réseau local sans une bonne configuration de ce routeur, car tout dispositif du LAN possède seulement une adresse IP privée, et seulement le routeur qui a deux adresses IP, l'une est privée, c'est avec laquelle il est connecté au LAN, l'autre est public, avec laquelle il est connecté à internet.

Pour faire connecter les dispositifs à internet le routeur utilise le protocole NAT, (NETWROK ADRESS TRANSLATION).

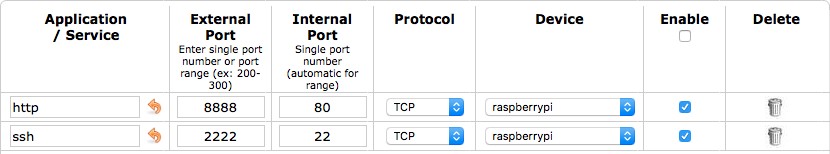
Pour diriger toute demande de service de connexion aux objets vers ces derniers, on utilisera le PNAT avec lequel on utilise les ports pour la translation des adresses (port forwarding). Le schéma suivant montre le principe de la configuration du routeur:



*Figure 14: Principe du port forwarding.*

Deux services sont accessibles depuis l'internet. Le serveur SSH pour le contrôle à distance de l'objet. Et le serveur WEB pour l'interaction avec l'utilisateur .

Pour se faire, on configure le routeur en rajoutant les deux règles montrées ci-dessous :



*Figure 15 : Configuration du routeur pour le port forwarding.*

Jusqu'à maintenant, on a tout ce qui est nécessaire pour pouvoir accéder aux objets souhaité

#### Chapitre III

**Présentation du systeme et pré requis:**

Résumé :

Ce chapitre, et le chapitre suivant se posent sur la création d'un système de surveillance qui fonctionne en temps réel, et sauvegarde un historique des images en cas de besoin.

La carte Raspberry Pi, dans cette partie, est utilisée comme étant un objet qui contrôle une caméra (ou des caméras) et diffuse le contenue à l'utilisateur.

Problématique:

Si on souhaite regarder un flux video de bonne qualité et grande résolution on aura besoin d'une large bande passante ce qui n'est pas possible avec une ligne ADSL qui a un débit montant très faible par rapport au débit descendant et ne dépasse pas les 50 kB/s. Ce qui rend la diffusion de plusieurs vidéos en même temps est presque impossible.

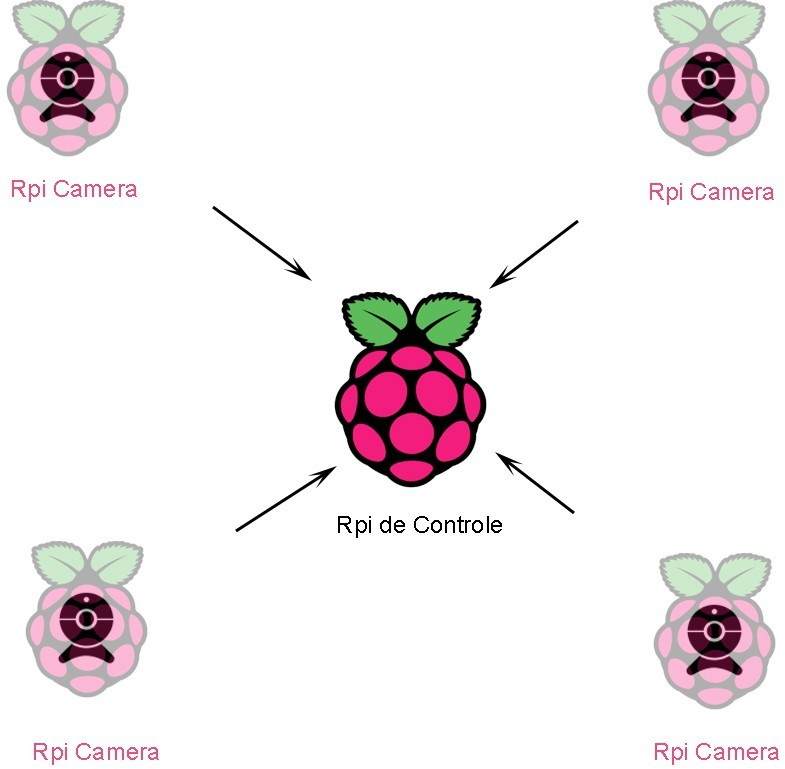
Pour résoudre ce problème, on peut soit **baisser la qualité de l’image**, soit **baisser le nombre d’images prises par seconde** (rappelons qu’une vidéo n’est autre qu’un enchainement de photos). Dans notre système nous avons fait les deux !

1. **– Le principe global du systeme:**

Le système se compose d’une ou plusieurs Raspberry Pi équipées chacune d’une caméra (quelque soit son type) et d’une Raspberry Pi de «contrôle» qui centralise les images. Les Raspberry Pi se contentent de prendre une photo pendant des intervalles de temps réguliers et de les envoyer via WiFi ou Ethernet au Raspberry Pi de «contrôle». Chaque image est identifiée par le **nom de la caméra** d'où elle provient, et **l'instant pendant lequel elle a été prise** afin de pouvoir la situer dans le temps.

La Raspberry Pi de «contrôle» archive toutes les images reçues et peut y effectuer des traitements de détection des mouvements par exemple. Elle est également équipée d’une interface Web permettant de consulter les images à distance depuis n’importe quel service ou navigateur web.

Nous verrons comment créer un système de télé-surveillance en commençant par son noeud central, la Raspberry Pi de «Contrôle».



*Figure 16 : Raspberry de contrôle.*

Dans notre projet nous n'avons qu'une seule camera et une seule carte Rpi donc on va se contenter de ces deux outils pour construire notre système.

1. **– L’utilité des photos vis-à-vis de la vidéo:**

Un système de surveillance, ou de télé-surveillance, domestique doit permettre deux choses :

* + voir en direct ce qui se passe dans le lieu de surveillance à distance. des photos prises régulièrement feront l'affaire.
  + avoir une trace d’un évènement qui s’est déroulé. Cela nécessite de stocker les images pendant un instant donné.

L’important, c’est d’avoir les images en **temps réel**, de pouvoir les **stocker** et les

**visionner** facilement, sur place ou à distance. Dans ces conditions, nous pouvons nous

satisfaire d’une photo par seconde, voire moins. C’est exactement le principe du [MJPEG](http://fr.wikipedia.org/wiki/Motion_JPEG) qui est largement utilisé dans de nombreuses caméras IP. Le célèbre logiciel de détection de mouvement “Motion”, fonctionne également sur ce principe .

**Le gros avantage de la photo par rapport à la vidéo pour la télésurveillance:**

Demander à une caméra de prendre une photo pendant un intervalle régulier plutôt qu’une vidéo, a également de nombreux avantages et permet une grande souplesse d’utilisation :

* entre chaque photo, la caméra ou la Raspberry Pi est disponible pour faire autre chose.
* une photo peut être prise et conservée en haute résolution puis facilement réduite pour être affichée à distance. On peut ainsi garder un historique des photos ou évènements en haute définition 1080p, et consulter avec le mobile une miniature de ces photos.
* une photo prise, peut être immédiatement envoyée sur un serveur distant pour y être sauvegardée et archivée.
* il est très facile de détecter un mouvement en comparant deux photos.
* il est très facile de créer une vidéo à partir d’images qui se suivent.

##### – Présentation du contrôleur :

Comme déjà mentionné. Le contrôleur aura deux fonctions principales : Une fonction de

**concentration** des images et l'autre pour l'**interface Web** dédiée à l'utilisateur.

**Remarque :** Pour réaliser ce contrôleur, on peut utiliser n’importe quelle Raspberry Pi (ou n’importe quel ordinateur) reliée au réseau LAN et internet, en WiFi ou Ethernet.

La première étape consiste à installer et configurer un serveur web Apache et PHP5. Ce serveur web servira à récupérer les images envoyées en HTTP POST et à diffuser à l’interface web de visionnage.

A – Installation d’Apache:

Avant d’installer le serveur, on doit mettre à jour le système :

*# sudo aptitude update # sudo aptitude upgrade*

Une fois la Raspberry Pi est à jour, on installe le serveur Apache:

*# sudo aptitude install apache2*

B – Installer PHP:

On fait appel à l’administrateur pour installer PHP avec la ligne de commande.

*# sudo aptitude install php5*

C – Dossiers de concentrations des images:

Maintenant dans le dossier du serveur web nous allons créer deux dossiers, l'un est pour l e **stockage** (ou mémorisation) des images en **bonne qualité**. L'autre c'est pour la **diffusion**. C'est dans ce dernier dossier qu'on va stocker les images diffusées de **qualité moyenne.**

Chacun des deux dossiers contient d'autres dossiers correspondent aux caméras utilisées.

**Jusqu'à maintenant, on a tout ce qui est nécessaire pour la construction de notre système de surveillance.**

**Chapitre IV:**

**Construction du systeme de surveillance.**

**Principe:** les images prises seront envoyées en HTTP POST par la Raspberry Pi équipée d’une caméra, il suffit donc de créer un petit script en shell pour capturer et envoyer ces photos, et un programme en PHP pour les recevoir.

##### – Prendre et envoyer des photos au contrôleur :

*(le script de cette partie s'appelle cam1.sh , on doit exécuter le même scrpit dans chaque carte utilisée dans le système après tout changement du nom de sa propre caméra)*

Donc chaque Raspberry dispose d'un script shell qui prend une image régulièrement et la renvoie vers le contrôleur.

Puisque nous n'avons qu'une seule carte RPI qui dispose d'une seule caméra, et qui joue le rôle du contrôleur en même temps, deux solutions seront possibles. On peut déplacer directement l'image dans le dossier correspondant au caméra en précisant l'output de la commanderaspistill:

*# raspistill -t 1 -o $img\_name*

l'autre solution est celle qu'on va utiliser, on va se servir de l'adresse localhost comme adresse de contrôleur avec la ligne :

dst\_url="<http://localhost/pfa/concentrateur.php>"

après avoir préciser le nom de la caméra et l'adresse de destination, les lignes

refresh=1 (4ème ligne) et sleep $refresh (15ème ligne) servent à préciser le

délais entre chaque photo. while true ; do #code ...

done

pour une boucle infinie .

raspistill -t 1 -w 640 -h 480 -q 50 -o $img\_name

c'est la commande pour prendre une photo depuis notre module caméra. Elle prend les arguments suivants:

* + -t pour préciser le délais.
  + -w et -h respectivement la largeur et la hauteur.
  + -q la qualité de la photo (entre 0 et 100).
  + -o le nom de l'image ainsi que son répertoire.

On a va nous servir de la commande curl (client url qui s'utilise pour récupérer le contenu d'une ressource accessible par un réseau informatique), pour envoyer les photos vers le contrôleur:

curl --form camera\_image=@$img\_name --form camera\_name=$camera\_name --form token=azerty $dst\_url.

On va envoyer notre image avec le nom de la caméra et le petit mot de passe 'token' sous forme d'un formulaire avec l'option –form. Puis, on précise la destination du fichier traitant cette image avec l'argument $dst\_url.

Une fois l'image est envoyée elle sera supprimée par la ligne

rm $img\_name

**Remarque**: une ligne dans le script shell ou bien commande c'est la même chose. Enfin l'image sera traitée par le fichier de contrôleur **concentrateur.php.**

##### – concentrateur.php:

Lorsque le collecteur reçoit une image moyennant la page concentrateur.php (contenue dans la variable super globale $\_FILES["camera\_image"]["tmp\_name"]), il vérifie la réception du nom de la caméra ( $\_REQUEST['camera\_name'] ), il examine ainsi, si cette image est envoyée vraiment par l'une des caméras du système utilisant le mot de passe 'token'($\_REQUEST['token']).

Comme mentionné dans le chapitre précédent nous avons deux dossiers:

A – Stockage :

Ce premier dossier est un dossier de stockage. Le concentrateur vérifie si le dossier correspondant au nom de la caméra existe déjà, if(!is\_dir($stock\_dir.'/'.$camera\_name)),*” tous les dossiers de chaque caméra se trouvent dans un dossier nommé caméra dont le nom est contenu dans la variable $stock\_dir.”* Si oui, il va enregistrer les images dans ce dossier, sinon il crée un nouveau dossier, nommé de même nom de la caméra, dans lequel toutes les images provenantes de cette caméra seront enregistrées.

* + Pour la création du dossier :

mkdir($stock\_dir.'/'.$camera\_name, 0777, true);

* + Pour l'enregistrement d'image :

move\_uploaded\_file($\_FILES["camera\_image"]["tmp\_name"],

$stock\_dir.'/'. $camera\_name.'/'.$img\_name); **.**

B – Diffusion :

Ce dossier de diffusion fonctionne de la même manière que celui de stockage, autrement, le concentrateur vérifie l'existence du dossier, s'il n'existe pas, on le crée, et c'est ainsi pour les dossiers de chaque caméra.

Dans cette partie, le concentrateur colle les images prises par la caméra dans des images de petite taille et avec une faible résolution (compressées), et c'est grâce aux fonctions de traitement d'images:

* + imagecreatefromjpeg(): création d'une image à partir d'un fichier jpeg.
  + imagecreatetruecolor($width,$height):création d'une image vide d'une
  + taille spécifiée avec ces deux arguments .
  + Imagecopyresampled(): copier le contenue d'une image sur l'autre.
  + imagejpeg($img2,$direct,50): retransformer l'image $img2 dans le fichier

$direct .



*Figure 17 : Image origine (270kB)*

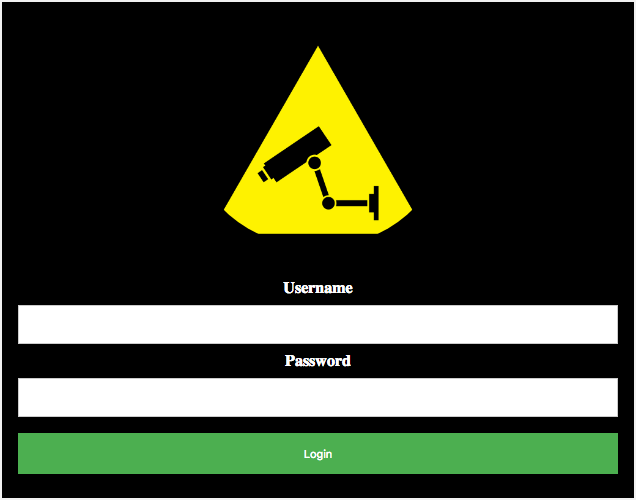


*Figure 18 : Image compressée (Taille : 3.2 kB)*

##### – Visionnement des images :

Nous avons vu dans le premier chapitre que l'utilisateur s'authentifie dans le serveur de collection des IP pour sélectionner son objet désiré. Après il est redirigé vers l'application web hébergée dans chaque contrôleurs des objets . Maintenant, on va expliquer le fonctionnement de cette application Web.

A – Authentification:



*Figure 19 : Authentification d'utilisateurs.*

Pour pouvoir accéder à la page de diffusion des photos de surveillance, on va se servir d'une page d'authentification qu'on va appeler login.php, pour des fins de contrôle d'identification pour les utilisateurs non authentifiés.

Une fois l'utilisateur est authentifié la page **index.php** va lui afficher notre système de surveillance.

B – index.php:

C'est la partie centrale de notre application elle affiche la dernière image de chaque caméra.

Après l'authentification d'utilisateur, ce dernier reçoie la page web contenant des division dont le nombre est celui des cameras spécifié dans le tableau :

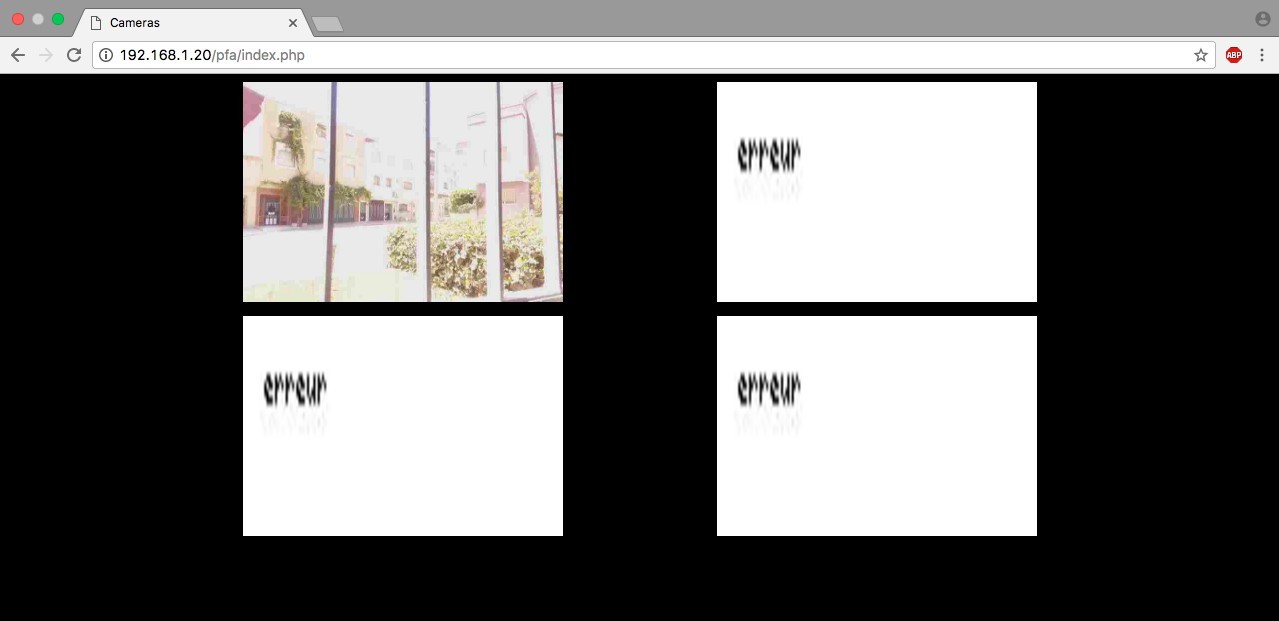
$cameras=array('cam1','cam2','cam3','cam4');

Ces division contiennent les dernieres images prises par chaque caméra. **Une fonction JavaScript se charge de raffraichir les images à intervalles réguliers.**

Au niveau du php, deux fonction sont responsables d'envoyer les photos vers l'utilisateur:

* La fonction LoadJpg crée une nouvelle image depuis un fichier ou une URL moyenant la fonction php imagecreatefromjpeg.Cette fonction renvoie une image contenant le mot “erreur” si l'image n'a pas été crée.
* Et la deuxième fonction c'est showLastImage qui prend en argument le nom d'une camera et renvoie la derniere image prise par cette camera.

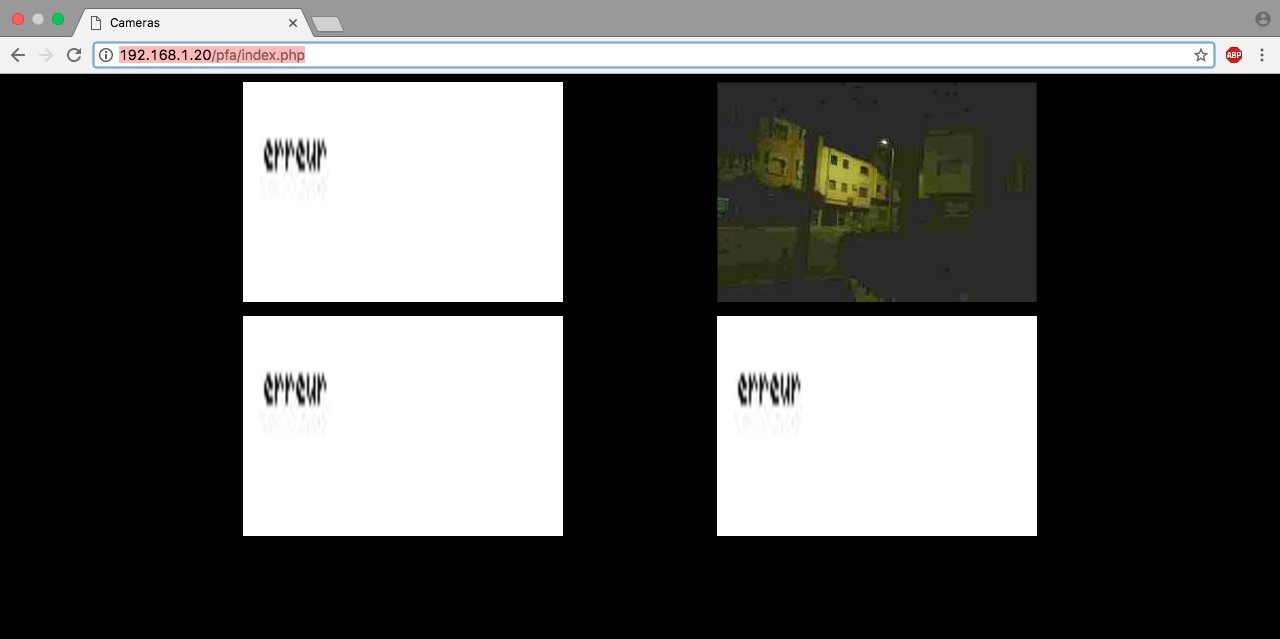
Cette fonction qui prend en argument le nom du camera, va créer une variable

$dir qui prend en valeur le chemain vers le dossier correspond à cette camera, puis la variable $imgs=glob($dir) contient tout les images dans ce doissier.Et voila

*Figure 20 : photo prise par cam1.*

La figure N°21 montre le résultat final de notre application.

Nous avons obtenu 4 divisions qui sont supposées l'emplacements d'affichage des images des caméras (cam2,cam3 et cam4 ). Maintenant on va modifier le script de capture d'images



*Figure 21 : photo prise par cam2.*

et on va nommé la camera “cam2” au lieux du “cam1”. Nous allons obtenir le résultat suivant

Maintenant il nous reste que l'appel à un script php delImgs.php pour effacer les images diffusées pour ne pas ralentir le fonctionnement de notre application, on va lui appelé apartir du script index.php avec la fonction:

exec ("php /var/www/html/pfa/delImgs.php");

Tant que le shell qui va interpreter la command

php /var/www/html/pfa/delImgs.php

on doit spécifier au début du fichier delImgs.php quel interpreteur va nous interpreter ce script en rajoutant la ligne: #!/usr/local/bin/php. Dans le script on spécifie le délais que restre l'image dans le dossier difus, dans notre application nous avons choisi 15s.

# Conclusion:

Durant le présent projet de fin d’année, Il nous a été confié la mission, d’étude et la mise en place d’un système de vidéosurveillance multi-caméras avec la carte **RASPBERRY**. Pour cela, notre travail a été décomposé en deux principales parties.

La première avait pour but de découvrir la carte Raspberry Pi, et sa caméra v2, ainsi de développer des différents programmes pour la réalisation du serveur enregistreur d'adresses IP dynamiques des objets (caméras) pour la résolution du premier objectif de notre projet. La seconde partie consistait à la construction de notre système de surveillance, en exploitant les différents programmes et traitements pour qu'elle soit accomplie. Ce projet a donné lieu à un système de surveillance multi-caméras avec carte Raspberry qui assure plusieurs fonctionnalités de surveillance.

La période de la réalisation de ce projet, était un excellent exercice professionnel qui nous a permis d’appliquer les différentes étapes de la réalisation d’un projet et notamment de développer nos compétences en termes de savoir-faire et savoir être.

En perspectives, nous cherchons à améliorer les résultats obtenus avant de faire une démonstration sur un lieu de surveillance donné.

# Bibliographie :

* + Christophe Blaess,Scripts shell Linux et Unix;2e edition ; Eyrolles ;2012
  + Samia Bouzefrane,Systèmes d'exploitation : Unix, Linux et Windows XP avec C et Java;DUNOD;2003

# Webographie :

* + [**https://www.raspberrypi.org/**](https://www.raspberrypi.org/): Site internet officiel de la fondation RASPBERRY PI.(consultée le Juin 2017)
  + [**https://doc.ubuntu-fr.org/**](https://doc.ubuntu-fr.org/): Site officiel de la documentation de la distribution liux 'UBUNTU'(consultée le Juin 2017)
  + [**http://php.net/**](http://php.net/): site officiel du php (consultée le Juin 2017)