École Centrale De Lille

Projet d'intégration

Suivi d'un grimpeur par caméra dome

Rapport de PDI
19 mars 2025

CHEVALIER Romain

Tuteur : BOURDEAUD'HUY T.

Table des matières

1 Contexte

Afin d'améliorer les services offerts aux utilisateurs de la salle d'escalade *Sac à Pof* situé à Mons-en-Barœul (59), on envisage de mettre en place un système de suivi automatisé des grimpeurs à l'aide de caméras IP dômes motorisées. Ces caméras permettront de suivre les mouvements des grimpeurs sur les différentes voies en ajustant automatiquement le cadrage et le zoom en fonction de leur position. Le services sera accessible via une application web hébergée sur un serveur local. Cette solution vise à offrir un support d'analyse post-session, afin de permettre aux grimpeur de comprendre leurs forces et faiblesses pour s'améliorer.

Afin de réalisé ce projet, nous sommes en possession de deux caméras IP dômes motorisées de la marque *AXIS* modèle *Q6304-E*[1], d'un routeur, d'une tablette Android et d'un raspberry pi 4.

1.1 Objectifs

Les objectifs de ce projet sont multiples et sont résumés dans la liste suivante :

- Prendre en main les caméras IP dômes motorisées et les intégrer dans un réseau local.
- Développer une interface web permettant de piloter les caméras, l'enregistrement de video et leur visualisation.
- Avoir un retour visuel de la caméra en temps réel sur l'interface web.
- Développer une méthode de communication entre l'application web et les programmes de contrôle des caméras.
- Développer un algorithme de suivi de grimpeur en temps réel.
- Développer une méthode permettant d'enregister sur un stockage une vidéo.
- Avoir une application modulable permettant de modifier les caméra et les paramètres de détection.

2 Architecture du système

2.1 Architecture matérielle

Le système est composé de plusieurs éléments qui interagissent entre eux, la figure 1 montre l'architecture du système. Le système repose sur un **Raspberry Pi 4**, qui joue un rôle central en assurant la communication entre les différentes composantes. Il est connecté au réseau local via un **routeur**, qui agit comme un serveur DHCP, attribuant dynamiquement les adresses IP aux appareils du système.

Les **caméras Dome IP** sont accessibles via HTTP et envoient leur flux vidéo vers le Raspberry Pi lorsqu'elles sont activées. Ces caméras disposent de positions prédéfinies correspondant aux différentes voies d'escalade, mais peuvent également être orientées dynamiquement par l'utilisateur via la **tablette** fournie ou tout autre appareil connecté au réseau local.

Le Raspberry Pi exécute un serveur d'application web,ce serveur permet :

- de piloter les caméras (changement de position, orientation manuelle),
- de gérer les enregistrements vidéo,

1. Voir fiche technique pour plus d'informations

- de stocker dans une base de données les vidéos enregistrées ainsi que les positions des caméras définies par l'utilisateur.

L'enregistrement des vidéos est déclenché manuellement depuis l'interface de la tablette et les fichiers sont conservés jusqu'à suppression par l'utilisateur.

L'architecture ainsi mise en place assure une gestion centralisée et accessible du système, tout en permettant un contrôle précis et flexible des caméras et des enregistrements.

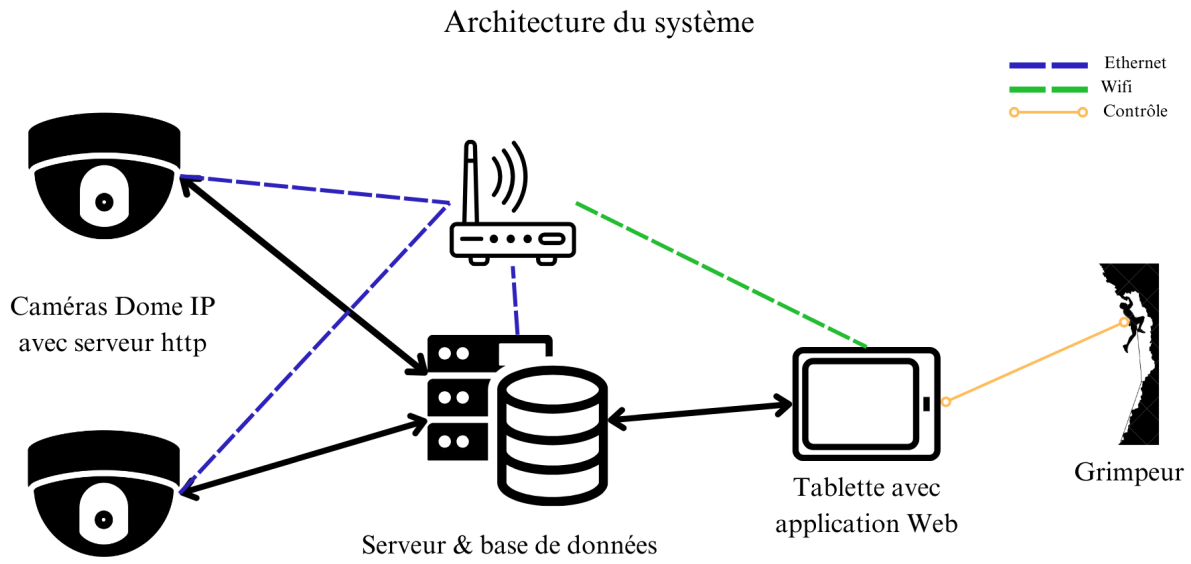


FIGURE 1 – Architecture du système

2.2 Architecture logicielle

L'architecture logicielle du système est représentée sur la figure 2. Le raspberry Pi divise ses tâches en deux parties distinctes : le serveur web et la partie gestion de l'application développée. En fonction des actions effectuées sur la page web, le serveur apache exécuté deux scripts `cgi` différents, le premier script, nommé **video.cgi** codé en `c++`, démarre au chargement de la page permet d'afficher le flux vidéo de la caméra sur la page web en lisant les images brutes dans la mémoire partagée et en les encodant en `JPEG`. Le second (**action.cgi** développé en `c`) permet de transmettre les ordres de déplacement de la caméra ou lancer un enregistrement vidéo via l'écriture de cet ordre dans un segment de mémoire partagé puis de réveillé le programme principal en lui envoyant un signal.

De l'autre côté, le programme principal (**main** programmé en `c`) est en charge de la gestion des caméras, de la détection de mouvement et de l'enregistrement des vidéos. Au démarrage, il crée un processus fils (**écriture mémoire**)qui est en charge de récupérer le flux vidéo du serveur HTTP de la caméra et de le stocker en image brute dans une mémoire partagée. Le programme principal est ensuite en attente d'un signal pour effectuer une action, comme le déplacement de la caméra, l'enregistrement de position ou encore le lancement d'un enregistrement qui va crée un processus fils (**enregistrement vidéo**) qui encode le flux video en `MP4` afin de le stocker sur le disque dur et lance la détection du grimpeur pour suivre ces mouvements à l'aide d'un second fils (**détection**). Ces deux

processus récupèrent les images en lisant la mémoire partagée et les traitent pour effectuer leur tâche. Enfin, le programme de détection peut directement agir sur la position de la caméra en lui envoyant une requête HTTP.

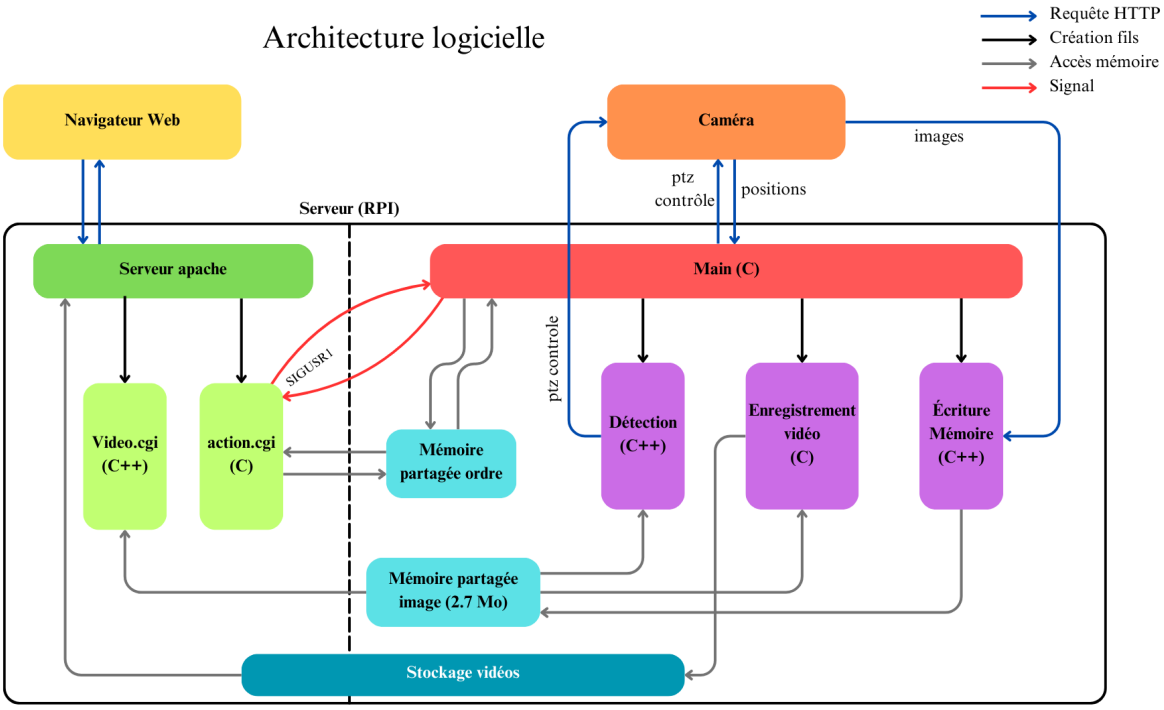


FIGURE 2 – Architecture logicielle

3 Développement de l'application

Dans cette partie, nous allons détailler les différentes étapes de développement de l'application pour chaque brique du système présent dans l'architecture logicielle en figure 2.

3.1 Application système

Le programme principal est le cœur du système, il est en charge de faire la liaison avec entre le serveur web et les différents processus, de piloter la caméra et de gérer les positions remarquables de la caméra.

3.1.1 Programme principal

Le fichier `main.c` constitue le cœur du système de gestion des caméras et de la communication inter-processus. Il est responsable de l'initialisation des caméras, de la gestion des enregistrements vidéo, de la détection des mouvements, et de l'interaction avec l'interface CGI via des segments de mémoire partagée et des sémaphores.

Initialisation du programme Lors de son démarrage, le programme effectue plusieurs étapes d'initialisation :

- **Gestionnaire de fin d'exécution** : La fonction `atexit(bye)` garantit que toutes les ressources (mémoire partagée, sémaphores, processus enfants) sont correctement libérées à la fin du programme.
- **Initialisation des sémaphores** : La fonction `initSemaphores()` crée ou ouvre des sémaphores nommés pour synchroniser les processus et gérer l'accès à la mémoire partagée, voir le paragraphe 3.2 pour leur fonctionnement.
- **Initialisation de la mémoire partagée** : La fonction `initSegmentMemoire()` configure deux segments de mémoire partagée : un pour les ordres envoyés par l'interface CGI et un autre pour stocker les images brutes capturées par les caméras.
- **Initialisation des caméras** : La fonction `initCamera()` charge les paramètres des caméras depuis un fichier JSON et configure les caméras actives.

Gestion des signaux Le programme installe un gestionnaire de signaux (`signalHandler`) pour répondre aux événements suivants :

- **SIGINT** : Interruption manuelle (Ctrl+C) pour arrêter le programme proprement.
- **SIGCHLD** : Gestion des processus enfants terminés pour éviter les processus zombies.
- **SIGUSR1** : Signal utilisé pour indiquer qu'un nouvel ordre a été reçu via la mémoire partagée.

Boucle principale Le programme entre dans une boucle infinie en appelant `pause()`, ce qui le met en attente d'un signal. Lorsqu'un signal est reçu, le gestionnaire de signaux (`signalHandler`) est exécuté pour traiter l'événement.

Gestion des ordres Lorsqu'un signal `SIGUSR1` est reçu, la fonction `gestionOrdres()` est appelée pour lire et exécuter les ordres envoyés par l'interface CGI. Les types d'ordres gérés incluent :

- **Déplacement de la caméra** : Commandes de mouvement (`move`) ou de zoom (`zoom`).
- **Enregistrement vidéo** : Commandes d'enregistrement (`reco`) pour démarrer ou arrêter un processus de capture vidéo.
- **Détection de mouvement** : Commandes de détection (`detc`) pour analyser les images et suivre les mouvements.
- **Gestion des positions prédéfinies** : Commandes (`rout`) pour ajouter, supprimer ou afficher des positions prédéfinies.
- **Changement de caméra active** : Commandes (`cam`) pour basculer entre les caméras configurées.

Gestion des processus enfants Le programme utilise des processus enfants pour effectuer des tâches spécifiques :

- **Processus d'écriture mémoire** : Capture le flux vidéo brut des caméras et le stocke dans la mémoire partagée.
- **Processus de détection** : Analyse les images pour détecter les mouvements.
- **Processus d'enregistrement vidéo** : Encode les images en vidéo MP4 et les stocke sur le disque.

Libération des ressources Lorsque le programme se termine, la fonction `bye()` est appelée pour libérer toutes les ressources :

- Les processus enfants encore actifs sont arrêtés.
- Les segments de mémoire partagée sont détachés et supprimés.
- Les sémaphores sont fermés et supprimés.

3.2 Écriture mémoire

Le fichier `ecritureMemoire.cpp` est un processus dédié à la capture des images vidéo depuis une source donnée (par exemple, une caméra IP) et à leur écriture dans un segment de mémoire partagée. Ce processus est essentiel pour permettre la communication inter-processus entre les différents modules du système.

Initialisation du programme Lors de son démarrage, le programme effectue plusieurs étapes d'initialisation :

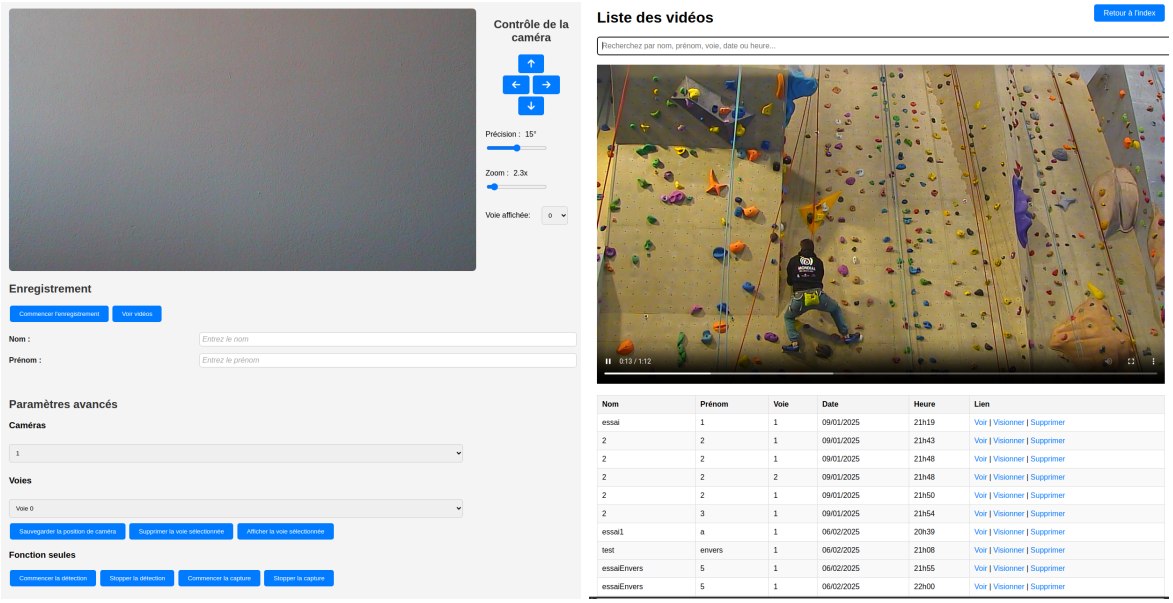
- **Gestionnaire de signaux** : Le programme installe un gestionnaire de signaux (`signalHandler`) pour gérer les interruptions (`SIGINT`) et les arrêts (`SIGTERM`) de manière propre.
- **Ouverture des sémaphores** : Les sémaphores nommés sont ouverts pour synchroniser l'accès à la mémoire partagée entre les processus :
  - `SEM_READERS` : Compte les lecteurs actifs.
  - `SEM_WRITER` : Contrôle l'accès en écriture à la mémoire partagée.
  - `SEM_NEW_FRAME` : Indique qu'une nouvelle image est disponible.
  - `SEM_ACTIVE_READERS` : Compte les lecteurs actifs qui attendent une nouvelle image.
- **Ouverture de la mémoire partagée** : Le segment de mémoire partagée est ouvert et mappé en mémoire virtuelle pour permettre l'écriture des images capturées.

Capture vidéo Le programme utilise OpenCV pour capturer les images vidéo depuis une source spécifiée en argument :

- **Initialisation de la capture vidéo** : La capture est ouverte avec `cv::VideoCapture` en utilisant l'argument passé au programme. Les propriétés de la capture sont configurées :
  - Fréquence d'images (FPS) : 25 images par seconde.
  - Résolution : 1280x720 pixels.
- **Vérification de la capture** : Si la capture ne peut pas être ouverte, le programme affiche une erreur et se termine.

Boucle principale Le programme entre dans une boucle principale pour capturer les images et les écrire dans la mémoire partagée :

- **Capture d'une image** : Une image est capturée à l'aide de `cap.read(frame)`. Si la capture échoue, le programme affiche une erreur et quitte la boucle.



(a) Index de l'application (b) Page web de visualisation des vidéos

FIGURE 3 – Pages web de l'application

- **Écriture dans la mémoire partagée** : Le sémaphore `SEM_WRITER` est utilisé pour bloquer l'accès en écriture pendant que l'image est copiée dans la mémoire partagée. Les données de l'image sont copiées avec `std::memcpy`.
- **Notification des lecteurs** : Le programme utilise le sémaphore `SEM_NEW_FRAME` pour notifier tous les lecteurs actifs (`SEM_ACTIVE_READERS`) qu'une nouvelle image est disponible.

Libération des ressources Lorsque le programme se termine (par un signal ou une erreur), il libère toutes les ressources utilisées :

- **Libération de la mémoire partagée** : Le segment de mémoire partagée est détaché (`munmap`) et fermé (`close`).
- **Fermeture des sémaphores** : Tous les sémaphores ouverts sont fermés avec `sem_close`.
- **Libération de la capture vidéo** : La capture vidéo est libérée avec `cap.release`.

Gestion des signaux Le gestionnaire de signaux (`signalHandler`) permet de gérer les interruptions et les arrêts du programme :

- **SIGINT** : Interrompt immédiatement le programme.
- **SIGTERM** : Arrête proprement le programme en mettant fin à la boucle principale.

3.3 Application web

Le serveur web est composée de deux pages web voir figure 3, la première permet de visualiser le flux vidéo de la caméra, de la piloter et de piloter l'application. La seconde permet de visualiser les vidéos enregistrées et de les télécharger, le code de ces pages est écrit en `HTML`, `CSS` et `JAVASCRIPT` peut-être consulté sur la page GitHub.

3.3.1 Page web index (figure 3a)

Au chargement de cette page, une fonction `javascript` permet de charger les voies disponibles dans la base de données et de les ajouter dans une liste déroulante pour permettre à l'utilisateur de sélectionner la voie qu'il souhaite visualiser. Une seconde fonction `js` charge les caméras disponibles et les ajoute dans une seconde liste déroulante. Puis le script `video.cgi` est exécuté pour afficher le flux vidéo de la caméra sélectionnée.

Le processus lit dans la mémoire partagée les images brutes a l'aide d'`OPENCV` pour récupérer une matrice de pixel, puis le profil colorimétrique de l'image est changé car `OPENCV` travaille en *BGR (blue green red)* et le `JPEG` utilise un profil (*RGB (red green blue)*) puis est encodé en `JPEG` à l'aide de la librairie `libjpeg`. Enfin, on ajoute les entêtes `HTTP` pour que le navigateur puisse interpreter l'image et on écrit l'image encodée dans la sortie standard.

Sur la gauche de cette page, on retrouve tout les boutons de contrôle de la caméra (haut, bas, gauche et droite) associé à un curseur pour la précision de l'angle afin de premettre un déplacement plus précis. On retrouve un second curseur pour gérer le zoom de la caméra entre 1x et 18x. La pression de l'un de ses boutons créent une requête `AJAX` pour exécuter le script `action.cgi` qui va écrire dans la mémoire partagée l'action à effectuer puis réveiller le programme principal pour qu'il effectue l'action et attendre le feedback du main ou un timeout de 5 secondes avec l'intruction `SIGALRM` (il récupère le pid du main dans un fichier que ce dernier crée au démarrage). Finalement, il envoie une réponse `HTTP` pour indiquer que l'action a bien été effectuée.

En dessous de ces curseurs, il y a une liste déroulante pour choisir la voie que l'on souhaite observer. À la selection d'une voie on vient changer de caméra si besoin et la positionner au bon endroit.

En dessous de la video, on retrouve un bouton pour enregistrer une vidéo avec deux champ pour entre le nom et prenom du grimpeur, un exemple de requête `AJAX` est présenté ci-contre : `http://localhost/cgi-bin/action.cgi?enrg=on&nom=Chevalier&prenom=Romain&voie=0`. On récupère le nom et la voie afin de produire un nom de fichier unique et pouvoir le récupérer facilement dans la page de visualisation des vidéos.

Paramètres avancés En bas de page, on trouve les options pour configurer une nouvelle voie. Il y a une liste déroulante pour choisir la caméra, puis une seconde liste déroulante pour choisir une voie que l'on souhaite supprimer ou afficher sur l'appuie du bouton adéquat. Il y a également un bouton pour sauvegarder une nouvelle position de la caméra dans la base de données en affichant une fenêtre popup pour demander le numéro de voie. En addition, on peut également lancer ou stop un enregistrement seul sans suivi du grimpeur et à l'inverse, on peut lancer seulement et stopper la détection du grimpeur sans enregistrement.

3.3.2 Page web de visualisation des vidéos (figure 3b)

Cette page permet de visualiser les vidéos enregistrées, de les télécharger et de les supprimer. Elles sont répertoriées dans un tableau avec le nom du grimpeur, la voie, la date et l'heure de l'enregistrement. On peut, de plus, rechercher une video particulière à l'aide de la barre de recherche en haut de page. Avec les fonction du navigateur, on peut lire, mettre en pause, accélérer, ralentir ou télécharger la vidéo. Enfin, on peut voir la video en haut de page en cliquant sur `voir`, la visionnier dans un nouvel onglet en cliquant sur `visionnier` ou la supprimer en cliquant sur `supprimer`.

4 Tests et résultats

5 Perspectives d'amélioration

- Amélioration de l'interface Web, avoir une interface plus intuitive et plus ergonomique.
- Reformater le code et tout traduire en anglais
- Afficher sur l'écran du Raspberry Pi, un meilleur status de l'execution de l'application.
- Utiliser les boutons du Raspberry Pi pour relancer l'application.
- Ajouter un système de notification pour les erreurs.
- Mieux gérer le stockage des vidéos.
- Ajouter un système de détection de chute.