Rapport Projet TER : Conception d'une surveillance vidéo médicale

SALVAN Corentin - MEHONG-SHIT-LI Matthieu - MARTINEZ Damien
BAIRY ONAPA Mikaël - LE LIDEC Tristan

18 avril 2021

Avant propos

L'objectif de cette unité d'enseignement est de nous préparé aux projets d'études de Master. Non content de consolider notre bagage scientifique, nous collaborons parmi un groupe de projet afin de sortir un produit fini, contexte qui pourrait se retrouver dans le milieu professionnelle également. Nous tenons avant tout à remercier Mr LAN SUN LUK Jean-Daniel pour l'aide précieuse apportée lors des points de blocages. Ce projet n'aurait pas aboutis dans le temps si nous n'avions consulter ouvrages et documents en ligne, sur lequel nous avons pu tirer informations, codes et librairies. Nous remerçions donc ces écrivains parfois anonymes.

Introduction

Table des matières

Première partie

Corps du rapport

Chapitre 1

Description

1.1 Définition du projet

Le but de ce projet est de réaliser une surveillance vidéo avec deux composantes, la détection de chute et la reconnaissance faciale. Pour la détection de chute, lorsqu'une chute sera détecté une alarme sera activé afin d'alerter le personnel soignant. Depuis une interface graphique l'utilisateur pourra visualiser en temps réel le flux vidéo, choisir la caméra, mettre en pause le flux vidéo et désactiver l'alarme. Concernant la reconnaissance faciale, l'utilisateur connaitra le nom et le prénom de la personne si il est enregistré dans la base de données de l'établissement, ce qui permettra de détecter les éventuels intrus.

1.2 Cahier des charges

Objectifs à atteindre :

CdC minimum, niveau 0 :

- Acquisition vidéo pour traitement via IP caméra
- Détection en vidéo via un code de "Machine Learning" d'une chute dans une chambre, du visage d'un patient s'étant égaré
- Interface graphique pour un poste de sécurité, boutons cliquable et visionnage vidéo des chambres
- Système d'alerte sonore et/ou visuelle, désactivé manuellement par un badge magnétique ou par l'interface graphique

CdC options supplémentaires niveau 1 :

- Micro/Haut parleurs pour communiquer patient/poste sécurité
- Localisé le personnel soignant présent sur le complexe hospitalier pour choisir l'intervenant le plus approprié et l'avertir

CdC options supplémentaires niveau 2 :

- Relevé des constantes vitale (tension, température, électrocardiogramme) et traitement des infos

1.3 Support Logiciels/ Langage de Programmation/ Librairie

Plusieurs langages de programmation ont été discuté avant la réalisation du projet, dans un premier temps, nous avons opté pour le langage JAVA, notamment pour la programmation orienté objet qui facilite le travail en équipe lors du dévellopement. Finalement nous nous sommes decidé pour le langage Python, car il bénéfie d'une grande communauté, possède de nombreuses librairies, fonctionnalités que nous avions besoin. D'autre part c'est un langage que tous les membres

de l'équipe savaient déjà programmer en Python. On utilise Virtual Box comme logiciel de simulation de différents systèmes d'exploitations. A partir des ressources d'une machine hôte ayant des performances correctes, dans notre cas un Window 10, nous créons deux machines virtuelles de types Linux (Debian et Raspberry OS). La démarche vise avant tout à régler les configurations, installer logiciels et libraires utiles et tester le code. Nous en parlerons dans la prochaine section.

Nous établissons une connexion SSH entre le Raspberry Pi (serveur) et notre PC Window (client) dû au fait que nous avions pas de moniteur de contrôle. Le protocôle SSH est un moyen de communication sécurisé et chiffré sur un réseau. (Une équivalence plus simple existe, le protocôle Telnet, mais il transfère les données de façon trop visible sur le réseau et est donc sensible au piratage).

On réalise la connexion en installant TighVNC sur Raspberry. (Tutoriel : Bibliothèque 3, p13). Sur Window est déjà présent l'outil 'Connexion à distance'. A partir d'un même réseau Wifi, nous nous servons de la reconnaissance IP pour l'asservissement.

Nous réalisons le schéma électrique de la section 2.3 avec le logiciel KiCad, qui est un logiciel multi-plaformes pouvant dessiner un schéma électronique, créer une empreinte PCB pour la réalisation de circuits imprimés ou faire une modélisation 3D du circuit.

Chapitre 2

Modélisation

2.1 Machine virtuelle OS Linux(Debian)

La nécessité de simuler un environnment Linux fut déclenché à la compilation d'une certaine librairie, mfrc522 (télécharger sur Pypi.org), qui avait des dépendances Linux. Même si la bibliothèque utilisée changea par la suite, le besoin de tester notre code étape par étape pour valider notre code resta instact. De ce fait, via VirtualBox de Oracle, nous créons un machine virtuelle de stockage allouée dynamiquement à partir d'un fichier .iso du système. (2048 Mo de mémoire vive et 15 Go de stockage)

Après démarrage de la machine, nous installons Python 3.9 (même si une version 2.7 est déjà intégré) et ses dépendances. Nous importons également 'pip3', un gestionnaire de paquets très utiles pour les librairies Python. Nous parvenons aussi à installer VS code via un logiciel nommé SNAP.

L'ensemble des commandes utilisées sur cette machine est donnée en annexe. (Annexe.3, page15)

La machine étant désormais opérationnelle, nous compilons l'ébauche du code que nous avions. L'intégration du flux vidéo, l'interface graphique s'est bien déroulé, mais surtout la librairie qui nous posait problème a pu s'intégrer. Nous avons levé une difficulté.

Néanmoins, un nouveau problème est apparu à cette phase : la lecture des ports GPIO étant propre au Raspberry, il nous fallait une machine de virtualisation plus adapté.

2.2 Machine virtuelle OS Raspberry Pi 2

Afin de virtualiser un système d'exploitation Raspberry Pi, nous étions d'abord parti sur la piste de Qemu : un émulateur pour processeurs. Nous l'installons sur notre système debian présenté dans la section d'avant. Les quelques lignes de commandes sont notifiées en annexe. (Annexe.4, page15)

Nous configurons le processeur sur Cortex-A7 (Raspberry 2), étant donnée que Cortex-A53 (Rasperry 3) n'est pas supporté. Mais nous recontrons une erreur de la procédure. De plus, selon certains développeurs, ce logiciel est particulièrement lent et présente plusieurs bugs.

Notre enseignant tuteur nous a donc conseillé de créer une machine virtuelle du système Raspberry Pi directement sur VirtualBox, le système d'exploitation .iso étant directement disponible depuis leur site officiel (https ://www.raspberrypi.org/). Malheuresement, seul l'OS du Raspberry 2 est actuellement disponible, l'OS du 3 en fichier iso est en développement. On suit un tutoriel donnée en bibliographie. (Tutoriel : Bibliothèque 4, p13) Attention cependant, une des étapes donnée imprime l'OS sur une clé USB externe pour formater un appareil. Cette étape n'est pas pertinante pour nous.

On affecte 4Go de mémoire vive et 15Go de stockage allouée dynamiquement. On réalise un diagnostic de rapidité à l'aide de la commande 'ping', dû à certains ralentissement rencontrés.

```
$ ping -a 192.168.56.1
```

L'adresse IP présent ici étant l'adresse de la machine. On envoie des messages de 64 bytes. Paquets transmis/reçu 60, moyenne : 0.713 ms. Temps total : 312 ms. Nous avons un bon débit, la lenteur venait donc visiblement de la machine hôte.

A l'instar de la machine Debian, on installe Python3.9 et pip sur la machine de la framboise. VsCode n'étant pas compatible, nous privilègions le logiciel 'Mu' pour la lecture et la compilation du code. Nous testons donc à nouveau, les élèments vidéo et interface s'excécutent, et nous pouvons désormais intégrer les ports GPIO via les bibliothèques rpi.gpio et pyrc522 0.1.2. (disponible sur pypi.org). Après le teste de quelques commandes de type serial (input/output) et la compilation de libraires, nous sommes passer enfin à la programmation sous notre Raspberry Pi 3b+ physique.

2.3 Schéma de montage Hardware et description des composants

Sur la partie matériel nous avons choisi d'utiliser un Rasberry Pi, du fait de son aspect compact et de sa grande capacité de calcul. Le modèle utilisé est le Raspberry 3b+, on énumère ici quelques caractéristiques :

- Mémoire vive :1 GB, 1.2GHz Quad-Core ARM Cortex-A53
- Nombre de coeur : 4
- Alimentation : 5V, 2A
- Ports USB : 4
- Module Bluetooth 4.1 et Wifi
- Interface Carte Graphique : PCI-E
- Processeur : 4 x ARM(v7) 7100
- GPU : Dual Core VideoCore IV

On liste ici les ports GPIO de raspberry utile pour la connexion avec le module RFID :

RFID	Vcc +3.3V	GND	MISO	MOSI	SCK	SDA	RST
Raspberry	01-3.3V	06-Ground	21-GPIO09	19-GPIO10	23-GPIO11	24-GPIO08	22-GPIO25

Chapitre 3

Réalisation Software-Hardware

3.1 Acquisition vidéo

Grâce à la librairie openCV, nous pouvons faire l'acquisition de tout type de caméra, en effet il suffit que la caméra soit branché à l'ordinateur pour qu'on l'utiliser. La lbrairie prend aussi en charge les caméras IP et le fichier vidéo.
A chaque appel de la fonction capture.read(), la librairie capture l'image du flux vidéo en entré, cette image va ensuite être utilisé pour faire les différents traitements et notamment la détection de chute.

3.2 Détection de chute

La détection se décompose en deux parties, la première consiste à extraire le sujet de la scène et la deuxième étudier son déplacement afin de savoir si oui ou non il y a bien eu une chute.

3.2.1 Extraction du sujet

Pour extraire le sujet deux méthodes ont été etudiés, la première methode consiste à utiliser une technique de classification. Cela consiste à faire de l'apprentissage supervisé. Il faut d'abord créer une base de données afin d'entrainer le modèle pour pouvoir reconnaitre le pattern souhaité. Cette méthode est coûteuse en ressource, mais peut être très efficace si la base de données est grande.Pour notre prblème utiliser cette méthode n'est pas la mieux adapté. En effet avec la base de donnée inclu dans la librairie openCV permet d'identifier les corps debout, le visage vers la caméra et le corps ne doit pas être partiellement caché par un objet. Pour contrer ce problème, nous avions décider de chercher une base de donnée plus complète ou de créer nous même la base de donnée. Cependant aucune de ses pistes n'a donné de résultat concluant. Nous avons donc suivi une autre méthode pour l'extraction du sujet. Plus simple à appréhender et à coder cette méthode consiste à utiliser la soustraction d'images. En faisant la soustraction de l'image précende avec l'image actuel on isole tous les changement ayant eu lieu entre les deux images et notamment les mouvements d'une personne.

3.2.2 Critère de détection de chute

3.3 Reconnaissance faciale

3.4 Interface Graphique

Nous avons cherché à créer une interface graphique pour notre porjet qui permette de modifier certains paramètres, de changer la caméra.

Pour réaliser cette interface nous avons utilisé la librairie intégrée de Python, Tkinter. Cette librairie permet de créer des interfaces graphiques assez simplement. Nous pouvons y ajouter des boutons des canvas (rectangle dans lequel nous pouvons placer du contenu).
La première chose que nous avons cherché à réaliser est l'intégration de l'acquisition vidéo à notre interface.

3.4.1 Intégration de l'acquisition vidéo

Pour intégrer l'acquisition vidéo à notre interface, nous devions faire en sorte qu'elle puisse être raffraichie pour modifier l'image affichée. La première idée que nous avons eu consistait à créer deux objets différents, un objet interface qui contiendrait les éléments de l'interface et permettrait d'en créer une. Puis un second objet acquisition, qui devait être l'acquisition vidéo. Nous n'aurions eu alors qu'à raffraichir l'image affichée.
Nous n'avons pas réussi à mettre cette solution en œuvre car l'acquisition vidéo ne possède pas de constructeur qui permette de créer un objet.
La seconde option que nous avons exploré consistait à réaliser un unique code qui contiendrait et l'acquisition vidéo et l'interface.
C'est cettte dernière option qui a été retenue. Elle était plus simple de mise en œuvre.

3.5 Commande du matériel

Chapitre 4

Bilan du projet

4.1 Difficulté du projet

4.2 Analyse du résultat confrontée au cahier des charges

4.3 Piste d'amélioration

4.3.1 Qualité du matériel

4.3.2 Résolution du système

4.3.3 Performance du code et du temps de calcul

Afin de gagner en vitesse de calcul, nous pouvons utiliser une programmation qui utilise les diffèrents coeurs du processeur. En effet, nous utilisons des threads pour parallèliser nos tâches, mais en l'état, le système décide lui-même de l'emplacement du calcul. Si on stocke les variables dans un buffer et qu'on affecte un thread à un coeur spcifique, le temps de calcul serait considérablement diminué.

Nous avons trouvé un tutoriel sur Youtube pour faire du multicore sur un Raspberry Pico. (Tutoriel : Bibliothèque 5,p13) Le Raspberry pico possède 2 coeurs, et est déjà équipé d'une bibliothèque Pico Multicore. Nous n'avons pas trouvé une équivalence pour Raspberry 3B+.

Il semble qu'une bibliothèque existe pour programmer du multicore, du nom de openMP. Nous n'avons pas développer plus, étant arriver au terme du projet.

4.4 Timeline

1. Semaine du 03/02/2021
   - Définition du projet
   - Ebauche du cahier des charges
   - Traitement de l'image avec OpenCV : documentation
   - Caractéristiques du Raspberry
2. Semaine du 10/02/2021
   - Importation du flux vidéo
   - Communiquer aux ports GPIO
   - Bibliothèque MFRC522
   - Choix de Tkinter pour l'IHM
3. Semaine du 17/02/2021
   - Résolution de bug pour l'association OpenCV/Tkinter
   - Recherche d'un programme pour la détection de visage

— Recherche d'algorithme sur la détection de chute

4. Semaine du 24/02/2021
   — Installation d'une machine virtuelle Debian
   — Code fonctionnelle détection une personne chuté
   — Interface graphique liée à la détection
   — Recherche sur les capacités mémoires du Raspberry pour flux vidéo selon une résolution
5. Semaine du 03/03/2021
   — Tentative de simulation Raspberry 2 avec Qemu
6. Semaine du 10/03/2021
   — Simulation d'un OS Raspberry Desktop et validation du code
7. Semaine du 17/03/2021
   — Commande port GPIO, code de l'alerte pour le hardware

Le temps restant à servi à corriger des bugs, surtout lier à la mise en commun des diffèrents codes et à rédiger notre rapport, en exploration les pistes d'amélioration.

Bibliographie

[1] Ben Nuttall Revision, Edit on GitHub, *Librairie : GPIO zero.*
Consulté le 03/04/2021

`url = https://gpiozero.readthedocs.io/en/stable/index.html`

[2] Raspberry Pi FR, *Utiliser un lecteur RFID avec Raspberry.*
Consulté le 03/04/2021

`url = https://raspberry-pi.fr/rfid-raspberry-pi/`

[3] Raspberry Lab fr, *Etablir une connexion SSH.*
Consulté le 17/03/2021

`url = https://raspberry-lab.fr/Debuter-sur-Raspberry-Francais/Connexion-Bureau-a-distance-Rasp`

[4] Raspberrypi.org, *Installer OS Raspberry Desktop : machine virtuelle.*
Consulté le 10/03/2021

`url = https://projects.raspberrypi.org/en/projects/install-raspberry-pi-desktop`

[5] Youtube, *Programmation Multicore Raspberry Pico.*
Consulté le 17/04/2021

`url = https://www.youtube.com/watch?v=aIFElaK14V4`

Deuxième partie

Annexes

.1 Code principal du système

.2 Code librairie : RFID.py

.3 Commande Machine vituelle Debian

```
Installation de Debian 64bit (Linux) sur un système hôte Window 10.
    Id : Mikael
    Mdp : cf344d
    Root Mdp : qwerty (correspondance sur clavier azerty)

Mettre à jour le docker (concevoir, tester et déployer des applications dans des conteneurs de lo
    $ sudo apt update
    $ sudo apt install software-properties-common apt-transport-https curl

Installer Python 3.9 (source : https://linuxize.com/post/how-to-install-python-3-9-on-debian-10/
    $ sudo apt update
    $ sudo apt install build-essential zlib1g-dev libncurses5-dev libgdbm-dev libnss3-dev libssl-
    $ wget https://www.python.org/ftp/python/3.9.1/Python-3.9.1.tgz
    $ tar -xf Python-3.9.1.tgz
    $ cd Python-3.9.1
    $ ./configure --enable-optimizations
    $ make -j 2
    $ sudo make altinstall
    Vérifier après installation
    $ python3.9 --version

Installer pip pour Python:
    $ sudo apt install python3-pip
    $ pip3 --version

Installer SNAP (équivalence de Windows Store pour Linux)
    $ sudo apt install snapd

Installer VSCode (Linux)
    $ sudo snap install --classic code
    $ sudo apt update
    Redémarrer le système

Installer les bibliothèques nécessaires
    $ pip3 install RPi.GPIO==0.7.0
    $ pip3 install opencv-contrib-python
```

.4 Commande Machine vituelle Debian

```
Tutoriel :
s://openclassrooms.com/fr/courses/5281406-creez-un-linux-embarque-pour-la-domotique/5464241-emule
Installer qemu :
    $ sudo apt-get install qemu-system-arm
Voir les plateformes supportés
    $ qemu-system-arm -machine help
```

Lister les processeurs supportés par l'émulateur pour cette plateforme

```
$ qemu-system-arm -machine raspi2 -cpu help
```

On fini par rencontré un message d'erreur:

```
root@debianMikachu:~# qemu-system-arm
-M raspi2 -cpu cortex-a7
-append "rw earlyprintk loglevel=8 dwc_otg.lpm_enable=0 root=/dev/mmcblk0p2"
-dtb bcm2709-rpi-2-b.dtb
-drive file=~/2018-04-18-raspbian-stretch-lite.img,if=sd,format=raw
-kernel ~/kernel7.img
-m 1G
-smp 4
No protocol specified

(qemu-system-arm:3530): dbind-WARNING **: 11:10:04.850: Could not open X display
qemu-system-arm: Unable to copy device tree in memory
Couldn't open dtb file bcm2709-rpi-2-b.dtb
```