# C:\Users\lembach_\Desktop\logo\logos OK\newlogo-rvb.png

**Table des matières**

[***Introduction***](#_heading=h.30j0zll) ***3***

[***Partie 1 : ESP32 TTGO***](#_heading=h.1fob9te) ***4***

[Matériel utilisé](#_heading=h.3znysh7) 4

[Montage réalisé](#_heading=h.2et92p0) 4

[Principe de fonctionnement](#_heading=h.jx6xjre2jjux) 5

[Installation des librairies et de l’environnement](#_heading=h.tyjcwt) 5

[Utilisation de la caméra](#_heading=h.3dy6vkm) 5

[**Esp32 avec Deep Sleep**](#_heading=h.1t3h5sf) **6**

[Capteur infra rouge et Deep Sleep](#_heading=h.4d34og8) 6

[Limites de la solution](#_heading=h.2s8eyo1) 7

[**Esp32 avec Wifi**](#_heading=h.17dp8vu) **7**

[Gestion du WiFi](#_heading=h.guzxrzk7yffq) 7

[Limites de la solution](#_heading=h.1pbrbbr93dhq) 7

[Conclusion](#_heading=h.1jgpic6gih63) 8

[**Partie 2 : RASPBERRY PI**](#_heading=h.yq3h56k0tnp5) **9**

[Matériel utilisé](#_heading=h.29kkfpwaz9) 9

[Montage réalisé](#_heading=h.2v2dh4g5duq6) 9

[Principe de fonctionnement](#_heading=h.ao5yefj1vnu8) 10

[Installation des librairies et de l’environnement](#_heading=h.liz2zwig063) 11

[Utilisation de la caméra](#_heading=h.86kob04elnrr) 11

[Raspberry en traitement-serveur](#_heading=h.sxacr1xg64z3) 11

[Une alternative pour se servir des deux matériels](#_heading=h.aslnj6u6d9q) 11

[Des protocoles de communications difficiles pour un partage de vidéo](#_heading=h.vsnt48rbbvtg) 11

[Raspberry en alternative totale](#_heading=h.9r6w2w40t94j) 12

[Une puissance machine faisant office de web server](#_heading=h.40nq37w1s2rg) 12

[Une utilisation très gourmande en énergie](#_heading=h.vdsgpfi7f9si) 12

[Conclusion](#_heading=h.hhke87klswqy) 12

# Introduction

Les systèmes de détection de flux de personnes actuels sont des solutions existant depuis des années, voir des décennies. L’idée de nos travaux était de pouvoir trouver une alternative à ces systèmes en utilisant l’analyse vidéo pour détecter le passage d’une personne. Le principe est de pouvoir proposer un système facile à positionner et fiable, pouvant être mis en place pour des évènements éphémères. Le but recherché était d’utiliser un esp32 mis à notre disposition pour réaliser ceci.

Le but recherché est de détecter le flux de personnes passant par un sas. La finalité de ce projet est d’apporter avec précision des réponses aux problématiques concernant la disponibilité (places disponibles) et la sécurité (évacuation des locaux). Pour nos travaux nous nous sommes basés sur l’exemple du SCD de l’UTT car il présente un cas simple : on souhaiterait savoir le nombre de personnes se trouvant à un instant t dans la pièce.

La recherche de l’existant a été amené avec le dispositif récemment ajouté à l’entré du SCD de l’UTT nommé Affluences. C’est une application qui indique combien de place reste-t-il à la bibliothèque en indiquant seulement un pourcentage et non une valeur précise comme dans notre cas. De plus, les mesures sont fait grâce à des capteurs de présence positionner de part et d’autre des côtés de la porte.

Pour cela, nous avons envisagé plusieurs solutions que nous détaillerons ensuite dans ce rapport. Une première consiste à utiliser uniquement l’esp32 équipé d’une caméra et d’un capteur de présence. Une deuxième solution est d’avoir recours à un équipement avec de plus grandes ressources, la raspberry.

# Partie 1 : ESP32 TTGO

Cette première partie sera divisée en deux solutions. La première est celle qui tire parti du mode Deep Sleep de l’esp32, mais nous verrons que cela pose d’autres problèmes, résolus dans la seconde solution, qui soulève également de nouvelles interrogations.

### Matériel utilisé

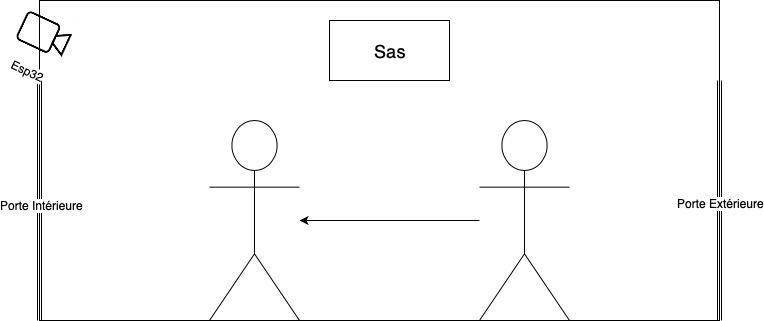
Dans cette première partie nous avons décidé d’utiliser uniquement un esp32 équipé d’une caméra et d’un capteur Infra-Rouge. Le modèle est fabriqué par l’entreprise LilyGo et se nomme TTGO. En voici une photo : Une image contenant mur, équipement électronique

Description générée automatiquement



### Montage réalisé

Pour comprendre le montage réalisé, il faut comprendre ce que l’on cherche à faire. Voici le cas d’usage que nous avons envisagé.

Afin de savoir si une personne rentre dans la pièce ou en sort, elle fera face ou sera dos à la caméra. De cette manière on peut détecter si elle entre ou sort et ainsi incrémenter ou décrémenter le compteur. Pour cela, nous plaçons la caméra au-dessus de la porte intérieure, face à l’autre porte du sas. Nous avons envisagé une autre solution dans la Partie 2 qui sera détaillée ultérieurement. Voici le schéma du montage en figure 1 : 

### Principe de fonctionnement

On peut voir sur ce schéma les deux portes, intérieure et extérieure. Lorsque l’une d’elle est ouverte le capteur infra-rouge détecte une présence, et donc active la capture vidéo. A ce moment-là deux solutions sont possibles. Soit un visage est détecté, alors cela signifie que la personne se déplace en direction de la caméra et donc vers l’intérieur de la pièce. On va alors considérer que si le visage est détecté deux fois de suite, cela signifie que la personne rentre effectivement. On incrémente alors le compteur. Si à l’inverse le visage n’est pas détecté sur plusieurs images consécutives, cela signifie que la personne sort et donc qu’on décrémente le compteur.

Lorsque l’on a incrémenté ou décrémenté ce compteur on met en « sommeil » l’esp, qui sera plus tard réveillé par un mouvement détecté par le capteur IR.

### Installation des librairies et de l’environnement

Pour programmer sur esp32 il a fallu que je télécharge et installe esp-idf, qui sert à utiliser les librairies Espressif sur l’environnement de développement (IDE) Arduino. Il a aussi été nécessaire d’installer le driver relatif à ce modèle d’esp32, car il n’était pas détecté par mon ordinateur. Ces étapes préalables sont indispensables afin d’utiliser au mieux le matériel.

### Utilisation de la caméra

L’esp32 que nous avons utilisé dans cette partie contient un module caméra (espCam), que nous utilisons pour la capture vidéo. Pour cela, il est nécessaire d’importer des librairies : « esp\_camera.h » et « camera\_pins.h ». La première contient les fonctions pour utiliser la caméra et la seconde contient les pins associés à la caméra (qui diffèrent d’un modèle d’esp32 à l’autre).

Il faut dans un premier temps configurer la caméra à un niveau matériel. Cela est réalisé à la manière de la figure 3.Une image contenant texte

Description générée automatiquementUne image contenant équipement électronique, intérieur

Description générée automatiquement

Le câblage est réalisé selon des constantes qui sont définies dans le fichier « camera\_pins.h ». On y retrouve 8 broches pour les données (de pin\_d0 à pin\_d7), deux broches d’horloges (pin\_xclk et pin\_pclk), un pin pour la synchronisation verticale (pin\_vsync), un pour la référence de tension (pin\_href), pour les deux canaux i2c (pin\_sda et pin\_scl), le pin d’alimentation (pin\_pwdn), et enfin le reset (pin\_reset). On attribue ensuite une fréquence d’horloge de 20MHz, et un format d’image en jpeg. On finit ensuite par définir la résolution de l’image, sur du QVGA dans notre cas. Le QVGA est une taille de 320\*240 pixels, c’est peu mais au vu de la puissance de la puce esp32 une plus grande résolution aurait été plus problématique.

Une fois toute cette configuration réalisée, il suffit d’initialiser la caméra avec la fonction esp\_init\_camera() prenant en paramètre la configuration ci-dessus. Une fois cela fait, on déclare un capteur à partir de cette caméra, sur lequel on attribuera toutes les caractéristiques de la captation en elle-même.

## Esp32 avec Deep Sleep

### Capteur infra rouge et Deep Sleep

Une fois la capture vidéo mise en place, il convient de se servir du capteur infra-rouge pour deux objectifs : en premier lieu pour pallier aux faux-positifs de la caméra (détecter un visage à cause d’une ombre par exemple), et en second lieu pour économiser l’énergie (la finalité d’un esp32 est de fonctionner sur batterie).

D’abord l’utilisation du capteur Infra-Rouge : Il est branché sur un GPIO, son usage est donc assez simple. On commence par déclarer une variable du numéro de broche sur laquelle le capteur est branché. Dans notre cas il s’agit de la broche 33. Grâce à la fonction pinMode, on déclare cette broche en entrée (input). Quand on veut lire la valeur du capteur, il ne reste qu’à faire une lecture (digitalRead) sur cette broche, puis d’ensuite en déduire des actions qui seront vues plus bas dans ce rapport.

Le lien entre capteur Infra-rouge et mode Deep Sleep à présent. Le mode Deep Sleep (ou sommeil) est présent sur esp32 et a pour finalité d’économiser de l’énergie en plaçant l’esp en mode veille profond. L’intérêt est certain pour ce projet, car comme on peut l’imaginer, le capteur n’est pas sollicité tout le temps, le flux de personnes n’est pas nécessairement continu, par moments personne ne franchira le sas et donc il ne sera pas utile de capturer une vidéo en continu. Le processus est alors simple. Un mouvement dans le sas réveille l’esp, via une interruption (qui sera détaillée juste en dessous). On va alors lancer la vidéo et détecter des visages. Ici deux choix sont possibles : Si on détecte plusieurs fois un visage (sur 2 images) on considère qu’une personne rentre dans le sas, et donc on incrémente le compteur. Si au contraire on ne détecte jamais de visage, on va décrémenter, car cela signifiera que la personne tourne le dos à la caméra et donc qu’elle sort de la pièce. On considère que si au bout de 9 lectures de la valeur du capteur infra-rouge, celui-ci est toujours à LOW, i.e. ne détecte pas de mouvement, plus personne ne se trouve dans le sas et l’esp peut à nouveau se rendormir. Si jamais on a détecté un visage on va repasser en mode sommeil également, sauf si le capteur IR détecte à nouveau un mouvement.

Parlons à présent de l’implémentation du mode Deep Sleep. Pour commencer, il faut savoir qu’à chaque reprise du système après un réveil, les variables du programme auront été effacées. Cela pose un problème car nous voulons conserver d’un cycle à l’autre le compteur. Pour cela il faut déclarer la variable avec l’attribut RTC\_DATA\_ATTR. La conséquence de cette indication est que la variable s’effacera uniquement lors d’un reset matériel (avec un appui sur le bouton reset) ou lorsque l’esp tombe à court d’énergie.

Pour définir que le capteur IR provoquera le réveil de l’esp, il faut appeler la fonction esp\_sleep\_enable\_ext0\_wakeup(GPIO\_NUM\_33, 1). Il provoque un réveil à partir d’une interruption externe, sur le GPIO numéro 33, lorsque celui-ci passe à un, c’est-à-dire lorsqu’un mouvement est détecté. Maintenant une fois que le traitement vidéo est terminé, on souhaite repasser en mode sommeil l’esp. Pour cela il suffit d’appeler la fonction esp\_deep\_sleep\_start(). Cela interrompt le programme et met en veille l’esp jusqu’à ce qu’un mouvement le réveille.

### Limites de la solution

La solution utilisant le mode Deep Sleep permet d’économiser de l’énergie, toutefois elle soulève un autre gros problème. L’esp restant en mode sommeil la plus grande partie de son temps, il n’est pas possible de créer un serveur web pour l’affichage du compteur en ligne (car inaccessible lors de la veille). Une deuxième solution est alors possible, celle de ne pas utiliser le Deep Sleep. Cela impose néanmoins de garder l’esp alimenté sur secteur.

## Esp32 avec Wifi

### Gestion du WiFi

Dans cette seconde solution, nous allons créer un point d’accès WiFi directement sur l’ESP. Il contiendra un serveur web avec une page HTML (basique dans notre cas) qui affichera la valeur du compteur au moment où l’utilisateur se connectera à la page.

Pour cela nous avons utilisé le travail que nous avons réalisé lors d’un tp de SY23 (une TM de filière que nous suivons ce semestre) qui consistait à envoyer sur une page web les données d’un capteur de température et d’hygrométrie, tout en utilisant un esp32. Ce travail a été fait sous la forme d’une librairie, il est donc aisé de récupérer celle-ci pour l’intégrer à notre projet. Cependant une modification a dû être faite. La librairie gère le capteur de température en l’intégrant directement en tant qu’objet dans la classe. Nous n’en avions pas besoin ici, étant donné que l’on manipule simplement un entier. La fonction gérant les requêtes de connexion au serveur (handleCompteurRequest()) prend donc maintenant un entier en paramètre, qui n’est autre que le compteur.

Lors de la phase d’initialisation de l’esp, il est nécessaire de créer le point d’accès ainsi que le serveur web. Pour cela nous avons nommé le point d’accès « esp-compteur » et ayant pour mot de passe « 88888888 » (pas sécurisé mais pouvant être changé sur demande). Il suffit ensuite d’appeler les fonctions de la classe « GestionWifi » (notre librairie) qui créent le point d’accès et le serveur. Dans la fonction loop on appellera simplement la fonction handleCompteurRequest() qui gèrera les demandes de connexion.

### Limites de la solution

Nous l’avons vu dans la première solution, le principal avantage à utiliser l’esp32 est la faible consommation d’énergie. Or on sait qu’en embarqué la principale source énergivore est la communication réseau. Le fait d’avoir un système qui oblige le WiFi à rester activé en continu va à l’encontre de ce principe. On entrevoit ici les limites de cette partie.

## Conclusion

La solution d’utiliser l’esp semblait la plus logique, la plus portable et la moins compliquée à mettre en œuvre. Elle présente toutefois un nombre important de points faibles.

Tout d’abord, lié à la puissance matérielle. On le sait, le traitement d’image est très gourmand en ressources. Sur des ordinateurs fixes ce n’est pas un problème, car on dispose d’une puissance très importante, mais sur un si petit système ça l’est. Pour avoir une fréquence de capture de 7fps on est obligé de diminuer la résolution en format QVGA (320\*240 px). On comprend très rapidement le problème qui se pose, reconnaitre un visage sur une si petite résolution est moins facile et plus soumis à des soucis de capture. La caméra de l’esp est également très sensible aux changements de luminosité (lors de nos essais la détection de visage fonctionnait parfaitement mais dans d’autres conditions (lumière changeante, trop lumineux, trop sombre) la détection ne fonctionnait parfois jamais ! On peut imaginer que dans le contexte d’un sas on fixe les conditions lumineuses mais cela rend l’installation moins portative.

On a pu également se rendre compte de la problématique de gestion de l’énergie. Dans un cas on peut en économiser, mais cela implique de devoir lire les données via une liaison série, donc de connecter physiquement un autre appareil à l’esp. Dans l’autre cas la Wifi est utilisée, rendant l’accès aux données très simplifié, mais c’est très énergivore.

Les limitations logicielles sont aussi présentes. L’utilisation de l’esp impose de se servir des librairies esp-who pour la détection de visages, à cause de la mémoire disponible. Cette librairie fonctionne uniquement avec des visages de face, limitant grandement la marge de placement de l’esp dans le sas.

Suite à cette impasse dans l’utilisation de l’esp avec sa caméra nous est venue l’idée d’un système, certes plus énergivore, mais moins limitant en termes de puissance et de librairies, la Raspberry Pi. C’est ce que nous allons aborder dans la partie 2.

# Partie 2 : RASPBERRY PI

Cette seconde partie exposera notre solution avec la Raspberry Pi. La raspberry n’étant pas initialement notre choix pour analyser le flux vidéo, nous avions plusieurs idées d’utilisation. C’est pourquoi cette partie sera également diviser en deux sections : l’utilisation traitement-serveur et l’utilisation totale. D’une manière générale, cette partie fera aussi office de comparaison par rapport à notre première solution sur esp32.

### Matériel utilisé

Dans cette seconde partie, nous avons utilisé en premier lieu utilisé une raspberry pi 3 lorsque nous travaillions en salle de labo D206. Puis, grâce au matériel d’un membre du groupe nous avons vite évolué vers la raspberry pi 4 disposant de plus de ressources et permettant ainsi un plus grand nombre d’image par seconde pour les analyses vidéos. Il est important de remarquer ici que la taille et le poids du matériel a augmenté comparé à la première solution. En voici une photo :

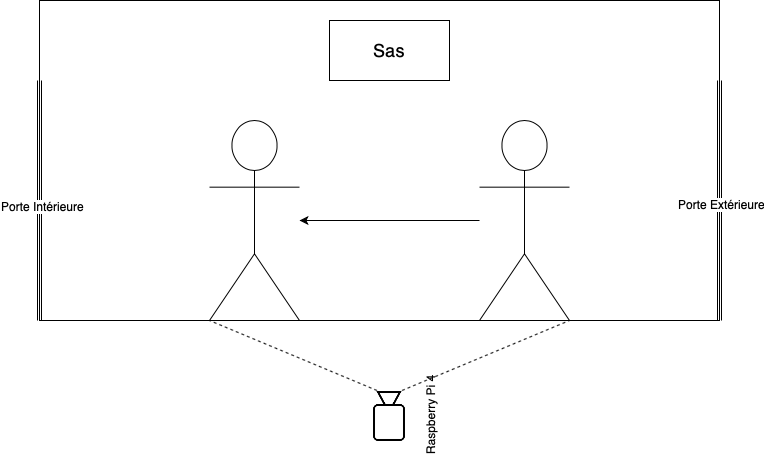
Une image contenant mur, équipement électronique

Description générée automatiquement

### Montage réalisé

A l’inverse de la solution proposée sur esp32 où nous disposions seulement de quelques librairies pour la reconnaissance du visage, la solution sur raspberry propose un grand panel de librairies sur beaucoup de parties du corps grâce à la bibliothèque graphique libre open-cv. Ainsi après avoir fait plusieurs essais des différentes librairies disponibles, nous avons opté pour la librairie upperboddy-mcs (haut du corps en français) qui semble être la plus précise autant de face que de profil. L’essentiel pour comprendre notre montage est qu’il faut pouvoir observer le buste d’une personne qui rentre ou sort du sas.

Afin de savoir si une personne rentre dans la pièce ou en sort cette fois-ci, elle sera de profil à caméra. De cette manière on peut détecter si elle entre ou sort et ainsi incrémenter ou décrémenter le compteur. La raspberry sera idéalement placé au milieu du sas à une distance de minimum 1m de l’axe du flux. Il était ici moins évident de mettre la raspberry en haut d’une porte du fait de sa taille et de son poids. Voici le schéma du montage en figure 1 : 



### Principe de fonctionnement

Dans le cas de notre raspberry, à l’inverse de l’esp32, nous ne possédions pas de détection de présence pour lancer la capture vidéo. Elle est admise pour fonctionner continuellement sans non plus possibilité de veille. Le compteur s’incrémente et décrémente suivant un algorithme que nous avons mis en place :

Lorsqu’un buste est détecté, la valeur en abscisse en abscisse est stockée pour pouvoir être comparé à une deuxième image de détection. Si la valeur en x a baissé, la personne détectée s’oriente vers la porte intérieure (autrement dit la sortie) et le compteur décrémente de 1. Par opposition, si la valeur en x a augmenté, la personne détectée s’oriente vers la porte extérieure (autrement dit l’entrée) et le compteur s’incrémente de 1. Pour éviter de compter plusieurs fois la même personne sortir/entrer nous avons utilisé des booléens permettant de contrôler la situation en fonction des valeurs des abscisses des détections successives. Seule une variation de sens de marche ou de changement brutal d’abscisse permet de dire qu’une nouvelle personne entre/sort de la pièce. Enfin pour contrer les faux positifs (comme par exemple en cas de mauvais éclairage) nous avons établi la règle d’une personne entrante ou sortante doit obligatoirement être détectée avant le tiers du cadre de l’image et par conséquent ne peut apparaître comme par magie au milieu de l’image sans avoir été détectée auparavant : comme un faux positif par exemple.

### Installation des librairies et de l’environnement

Pour programmer sur raspberry, nous avons utilisé l’IDE Qt creator pour programmer en c++. L’installation des librairies s’est faite grâce à open-cv que l’on a utilisé à travers Qt en déclarant simplement le chemin de la librairie que nous souhaitions utiliser.

### 

### Utilisation de la caméra

Nous avons utilisé la caméra interne de la raspberry appelée raspicam. Il est nécessaire de veiller à son bon fonctionnement en allant dans les configurations des paramètres de la raspberry à l’aide de la commande *raspi-config*. Nous déplorons seulement le fait de ne pas avoir un meilleur support de caméra.

La configuration de la caméra dans le code est faite en noir et blanc avec pour format 640\*480, une résolution pas trop gourmande pour disposer d’une vingtaine d’images par seconde.

## Raspberry en traitement-serveur

### Une alternative pour se servir des deux matériels

La première tâche que nous avons su faire sur esp32 était d’envoyer le flux vidéo sur une page web. Face aux problèmes d’analyse d’image avec un matériel disposant très peu de ressources comme l’esp32, nous avons pensé à le combiner avec une raspberry pi pour que celui-ci capture et envoie l’image pendant que la raspberry traite la reconnaissance de personne et la transmission des informations sur un site web. L’avantage est ici de pouvoir profiter pleinement de la discrétion par la faible taille de l’esp32 à l’entrée d’une pièce tout en analysant avec un plus grand nombre de ressources les vidéos grâce à la raspberry.

### Des protocoles de communications difficiles pour un partage de vidéo

Malgré le fait de pouvoir envoyer du contenu vidéo sur une page web, nous n’avons pas trouvé de solution valable de manière à récupérer ce flux à partir de la raspberry pi. Nous avons alors cherché d’autres protocoles de communication permettant un échange de données de tailles importantes et là encore nos recherches n’ont aboutis qu’à un protocole d’échange nommé MQTT pourtant très conseillé pour l’IOT mais pas pour la vidéo en raison de la taille des données à transmettre.

## Raspberry en alternative totale

### Une puissance machine faisant office de web server

Il est clair que les résultats sont très satisfaisants avec cette solution mais dépendent aussi de l’éclairage. La librairie choisie reconnaît aussi bien une personne de face que de profil en s’appuyant sur les caractéristiques suivantes : menton, cou, épaules, buste. La distance minimum est estimée à 1 mètre et la distance maximum à 3 mètre. L’utilisation d’une caméra fisher pourrait réduire cette distance minimum. Un élément important à prendre en compte : notre solution ne détecte qu’une seule personne à la fois dans le cadre de l’image. Il est donc nécessaire de filtrer le flux (sas, barrière) de manière à ce qu’une personne traverse le champ de l’image à la fois.

Comme pour l’esp32, nous avons transféré la raspberry en web server permettant ainsi à n'importe quel utilisateur du sous réseau wifi sur lequel est connecté la raspberry de pouvoir obtenir sur une page web le nombre de personne(s) dans la pièce (la variable compteur). Pour cela nous avons utiliser un script GUI et les classes *QTcpServeur.h* et *QTcpSocket.h*.

### Une utilisation très gourmande en énergie

L’un des gros points négatifs de cette solution est évidemment l’énergie. On estime qu’une raspberry pi 4b consomme 640 mA lorsqu’elle capture une vidéo et pouvant aller jusqu’à 885 mA en utilisant des applications demandant des ressources gourmandes. De plus, avec une raspberry il nous est impossible de mettre en sommeil le composant sans tout arrêter.

## Conclusion

La solution concernant la raspberry est celle qui permet de remédier aux défauts de la première solution sur esp32 au détriment de l’énergie et la portabilité.

En effet, la raspberry ne possède pas de mode d’économie d’énergie et est nettement plus énergivore que l’esp32 en nécessitant une alimentation de 5V et consommant autour de 700mA lors de la capture vidéo couplé au serveur web.

Au niveau portabilité, le composant est de taille 85 mm x 56 mm. Il est donc difficile de ne pas le remarquer à moins d’être bien agencé dans une boîte. Sans oublier qu’avec une batterie de raspberry pi permet en général le fonctionnement de l’appareil durant environ 7 heures. Il est donc encore question ici de raccorder l’appareil à une prise secteur.

Enfin concernant l’algorithme établi, une amélioration consisterait à comptabiliser le flux de plusieurs personnes en même temps dans le cadre vidéo. D’autre part, nous avons essayé de gérer au mieux les faux positifs mais le meilleur remède reste le bon éclairage.