

École Polytechnique de l'Université de Tours  
64, Avenue Jean Portalis  
37200 TOURS, FRANCE  
Tél. +33 (0)2 47 36 14 14  
[www.polytech.univ-tours.fr](http://www.polytech.univ-tours.fr)

Spécialité Informatique Industrielle  
Version : 01 -- 14/01/21

CAHIER D'ANALYSE			
Projet :	Comptage de Personnes		
Emetteur :	P.Martinez	Coordonnées : pablo.martinez@etu.univ-tours.fr	
Date d'émission :	14/01/21		
Validation			
Nom	Date	Valide (O/N)	Commentaires
Historique des modifications			
Version	Date	Description de la modification	
1.0	14/01/21	Version initiale	
1.1	16/04/21	Wiring diagram update (Matrice / Relai)	

**TABLE DES MATIÈRES**

<b>1. Introduction</b>	<b>2</b>
<b>2. Étude matérielle</b>	<b>3</b>
2.1 Cartes Linux embarqué	4
2.2 La caméra	8
2.3 Relai	13
2.4 Matrice LED	14
<b>3. Étude logiciel</b>	<b>17</b>
3.1 Carte Raspberry	18
3.2 Object detection	19
3.3 Communication entre les parties	23
<b>4. Étude de la consommation</b>	<b>25</b>
<b>5. Base de données (optionnel)</b>	<b>26</b>
<b>6. Serveur WEB de pilotage</b>	<b>27</b>
<b>GLOSSAIRE</b>	<b>28</b>
<b>BIBLIOGRAPHIE</b>	<b>29</b>
<b>ANNEXE</b>	<b>30</b>

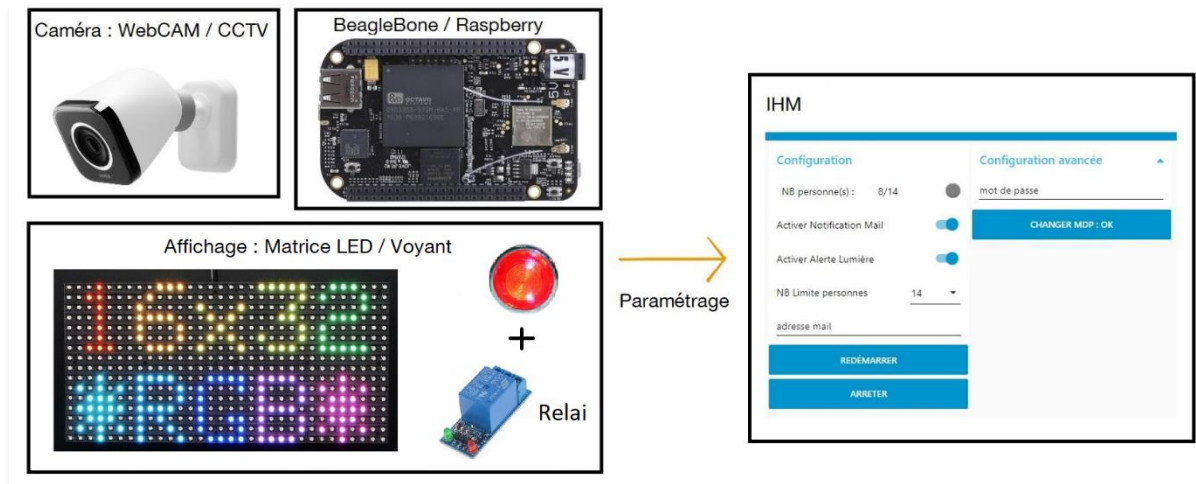
**CAHIER D'ANALYSE****1. Introduction**

Ce cahier retrace les réflexions menées sur le projet "Nombre de personnes" avec notamment l'analyse et la conception du système d'analyse d'images, l'analyse des contraintes liées à la caméra, la base de données et la partie affichage des données sur une matrice LED RGB.

**Rappel sur le projet**

En cette période de COVID, la capacité d'accueil de certaines salles est limitée. Dans ce contexte, l'idée est de créer un dispositif de comptage automatique sur la base de reconnaissance de forme. Le dispositif devra être mis en place dans une salle avec un seul accès et il permettra d'afficher à l'extérieur de la salle le nombre de personnes présentes dans celle-ci. ([Cf. 2.2 Objectifs](#))

## Rappel sur l'architecture



L'architecture sera constituée d'une carte électronique **Raspberry Pi 4** avec un linux embarqué. Sur la carte il y aura plusieurs services lancés :

- Un serveur **Base de données** de type relationnel pour la sauvegarde des informations. (optionnel)
- Un serveur **Web** pour les configurations.
- Un **programme** qui va analyser les images issues de la caméra afin de connaître le nombre de personnes dans la pièce.

Hors de la carte nous avons une **matrice LED** qui permettra d'afficher le nombre de personnes détectées dans la pièce. Un **voyant** lumineux contrôlé par un **relai** est aussi présent indiquant si la limite de personnes est dépassée.

## 2. Étude matérielle

Dans cette partie, nous allons aborder l'analyse du système au niveau matériel. Nous allons donc détailler les composants utilisés pour le projet "Nombre de personnes". On abordera ainsi les éléments fonctionnels nécessaires à son fonctionnement ainsi que les possibilités s'offrant à nous pour la réalisation. ([Cf. 3.1 Environnement du projet](#))

## 2.1 Cartes Linux embarqué



BeagleBoard.org  
BeagleBone Black

Processeur	AM3358 – ARM Cortex – A8
Fréquence processeur	1GHz
Nombre de cœur	1
Mémoire	512MB DDR3
Prix	75€ - <a href="#">farnell</a> <sup>1</sup>

Documentation : <https://github.com/beagleboard/beaglebone-black/wiki/System-Reference-Manual> (valide au 31/12/20)



Raspberry Pi 4

Processeur	Broadcom BCM2711 Cortex-A72 (ARM v8) 64-bit SoC
Fréquence processeur	1.5GHz
Nombre de cœur	4
Mémoire	4GB LPDDR4-2400 SDRAM
Prix	61€ - <a href="#">farnell</a> <sup>2</sup>

Documentation : [https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2711/rpi\\_DATA\\_2711\\_1p0\\_preliminary.pdf](https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2711/rpi_DATA_2711_1p0_preliminary.pdf) (valide au 31/12/20)

Lors des premiers tests de détection d'objet avec mon ordinateur portable, je n'ai eu aucun problème de performance je ne me suis donc pas rendu compte du fait que la carte embarquée allait devoir être puissante elle aussi.

La détection d'objet demande une puissance de calcul non négligeable afin d'avoir des performances convenables. Une chose que j'ai sous-estimé lors de mon analyse des spécifications.

Après mes recherches j'ai trouvé plusieurs façons de faire de la détection d'objet, parmi lesquelles **Tensorflow** et **Opencv** qui sont les plus utilisées dans les systèmes embarqués aux vues de leurs performances sur des cartes limitées en puissances de calculs.






<sup>1</sup> <https://fr.farnell.com/element14/bbone-black-wireless/beaglebone-black-wireless/dp/2671597>

<sup>2</sup> <https://fr.farnell.com/raspberry-pi/rpi4-modbp-4gb/raspberry-pi-4-model-b-4gb/dp/3051887>

**TensorFlow** est une API de détection d'objets créée par Google permettant de créer un réseau d'apprentissage en profondeur qui résout les problèmes de détection d'objets.

**OpenCV** (Open Computer Vision) est une bibliothèque graphique. Elle est spécialisée dans le traitement d'images, que ce soit pour de la photo ou de la vidéo.

Après études et mise en pratique des méthodes j'ai obtenu les résultats ci-dessous.

BeagleBone Black		Raspberry Pi 4	
 <b>OpenCV</b>	 <b>TensorFlow</b>	 <b>OpenCV</b>	 <b>TensorFlow</b>
<pre>fps :0.033 start net.forward() end net.forward() nb pers : 1 fps :0.034 start net.forward()</pre>	<pre>nb pers : 1 fps :0.0909 nb pers : 1 fps :0.0908 nb pers : 1 fps :0.0909</pre>	<pre>fps :1.8124652073514844 start net.forward() end net.forward() nb pers : 1 fps :1.8197961990946794 start net.forward()</pre>	<pre>nb pers : 1 fps :5.056293996394154 nb pers : 1 fps :5.060666485423957 nb pers : 1 fps :5.056078432153763</pre>
Test du 02.12.2020			
<b>BeagleBone black 512MB Cortex-A8</b> Linux version : <a href="#">AM3358 Debian 10.3<sup>3</sup></a> Release date 2020-04-06 Open cv version : 4.4.0 Tensorflow version : tflite_runtime 2.1.0 Taille de l'image : 640,480 Webcam : Microsoft HD-5000		<b>Raspberry 4 4go model B Cortex-A72</b> Linux version :  <b>Raspberry Pi OS (32-bit)</b> A port of Debian with the Raspberry Pi Desktop Released: 2020-12-02 Open cv version : 4.4.0 Tensorflow version : tflite_runtime 2.1.0 Taille de l'image : 640,480 Webcam : Microsoft HD-5000	
<b>Figure</b> : Test de performance avec différentes méthodes d'analyses d'image			

Le calcul de performance (FPS) est illustré par le Pseudo-code suivant :

```
//récupération du flux vidéo de la caméra
stream = cv2.VideoCapture(0)

TANT QUE(VRAI) {
    //récupération d'une image du flux vidéo
    image = stream.read()

    t1 -> COMMENCER mesure de temps

    //Traitement et comptage du nombre de personnes dans l'image
    int nbPersonnes = model.getPrediction(image)

    t2 -> FIN mesure de temps

    AFFICHER temps traitement (t2-t1)/1
}
```

<sup>3</sup> <https://debian.beagleboard.org/images/bone-debian-10.3-iot-armhf-2020-04-06-4gb.img.xz>

Le terme **FPS** désigne ici le nombre d'images par seconde que la carte arrive à traiter. Ce calcul est fait avec une mesure de temps avant et après l'exécution de la partie analyse d'images.

La puissance de calculs (FPS) n'est en rien un indicateur sur la qualité du résultat obtenu. C'est pour cela que les tests ont été faits avec le même seuil de confiance (je suis sûr à X% que mon résultat est juste). ([Cf. 3.2.2 Robustesse](#))

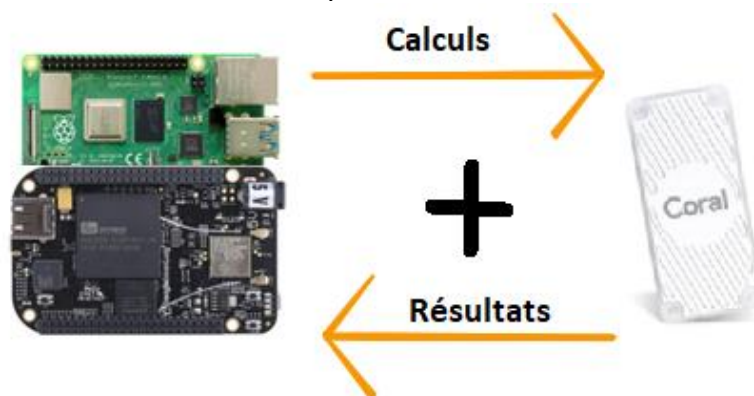
**Bilan du test :** L'étude montre les performances entre les deux méthodes d'analyse d'images, OpenCV et Tensorflow sur deux cartes différentes. On peut constater que pour l'analyse d'images de type "Object detection" l'utilisation de l'API Tensorflow a de meilleures performances. On voit aussi les très faibles performances de la BeagleBone Black sur le traitement d'images. Les résultats exclus l'utilisation de la carte BeagleBone Black seule pour ce projet en raison de son taux "FPS" trop faible. Les résultats ont été obtenus avec l'utilisation de modèles de prédictions optimisés. ([Cf. 3. Étude logiciel](#))

Nous venons de voir les deux solutions principales pour les cartes. Nous avons vu que la carte BeagleBone seule n'est pas assez puissante. Il existe d'autres moyens de déporter les calculs.

### La solution Coral

L'accélérateur USB Coral est un périphérique USB pouvant fonctionner avec les systèmes d'exploitation linux qui fournit un coprocesseur à notre carte embarquée. Il permet d'augmenter considérablement notre puissance de calcul mais il ne fonctionne qu'avec Tensorflow. Nous avons vu précédemment qu'avec une carte Raspberry 4 PI 4go avec 4 cœurs on est arrivé à avoir 5 FPS avec un modèle Optimisé Tensorflow. Avec Coral on peut estimer d'après les projets déjà présents sur internet, de 22 à 26 FPS. Une augmentation significative en termes de performances mais cette solution a un coût additionnel au projet (environ 65 euros en France).

Le principe est de déporter les calculs faits par carte lors du traitement de l'image et laisser le périphérique Coral les faire et nous renvoyer les résultats.



Il n'y a pas vraiment de désavantages à cette solution à part le prix et une consommation un tout petit peu supérieure par rapport à une carte seule. L'avantage principal est d'augmenter notre puissance de calcul tout en diminuant l'utilisation du CPU de la carte.

Pour l'instant cette solution est très avantageuse, mais n'est pas encore nécessaire en vue des résultats de la carte Raspberry 4 PI.

## La solution Ordinateur Distant

Cette solution est presque comme Coral, mais avec un ordinateur distant. L'ordinateur va effectuer les calculs et les renvoyer à la carte. Bien sûr cette solution a des avantages mais aussi des inconvénients.

Les performances seront supérieures à Coral si l'ordinateur est un minimum performant. La limite de FPS sera définie par le nombre d'images que la caméra peut fournir. Ci-dessous vous pouvez voir que ma caméra peut fournir 30 FPS. Lors de mes tests avec mon PC portable, j'ai pu atteindre 29 FPS.

Version Information	
Product Name	Microsoft® LifeCam HD-5000
Imaging Features	
Imaging Rate	Up to 30 frames per second

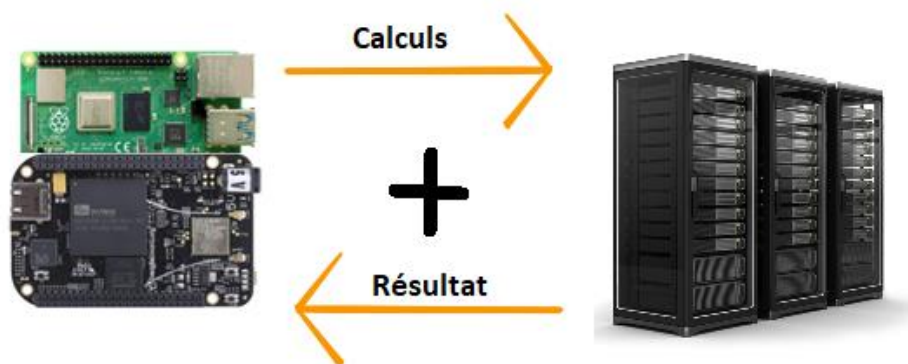
Comme avec Coral, avoir plus de FPS est une bonne chose. Mais cette solution a plus de contraintes.

Lorsque notre système est en fonctionnement, il faudra qu'en permanence un autre ordinateur soit allumé.

- Consommation augmentée
- Dépendance d'un autre ordinateur
- Mauvaise communication si problème de réseaux
- Plantage de l'ordinateur (mise à jour, coupure de courant...)
- Connexion sécurisée

On peut tout de même voir des avantages à cette solution

- Plus de FPS
- Pas obligé d'utiliser TensorFlow
- Ajout de plusieurs caméras et plusieurs traitements parallèles
- L'utilisation de méthodes d'analyse d'images non spécifiques aux cartes embarquées (beaucoup plus précis mais beaucoup plus complexe)





## 2.2 La caméra

Le choix de la caméra est une étape importante dans le projet, nous allons étudier plusieurs de ses caractéristiques et leurs importances.

### 2.2.1 Le FOV

Le FOV, en français "champ de vue" désigne l'angle de vue de la caméra. Plus le FOV de la caméra est grand plus la surface visible par la caméra est grande.

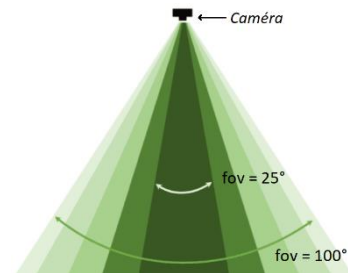


Figure 1 - Champ de vue

Le FOV apporte un réel avantage dans le monde de la prise de vues et de la vidéo, mais il a aussi ses limites. Plus le FOV est grand (objectif grand angle ou en anglais wide-angle) plus le phénomène de distorsion (fish-eyes) sera présent.

### 2.2.2 La distorsion



Figure 2 - Différence en image linéaire (sans distorsion) et avec distorsion

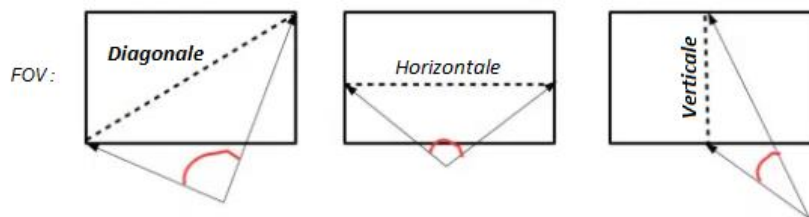
Le phénomène de distorsion est une déformation géométrique de l'image. Pour simplifier, si on photographie un objet, on s'attend à ce que la taille de cet objet ne varie pas quelle que soit sa position dans l'image. La distorsion est l'aberration qui opère cette variation de taille en fonction de la position dans l'image, lors du traitement des images, il sera donc plus compliqué de déduire les objets présents aux vues des dimensions inexactes.

On peut néanmoins appliquer un prétraitement des images pour enlever le phénomène de distorsion. Mais cela va impacter le temps de traitement de l'image global en fonction du type de distorsion à traiter (barillet, coussinet, moustache...).



Version Information	
Product Name	Microsoft® LifeCam HD-5000
Imaging Features	
Sensor	CMOS sensor technology
Resolution	<ul style="list-style-type: none"> <li>• Motion Video: 1280 X 720 pixel resolution*</li> <li>• Still Image: 1280 X 800</li> </ul>
Imaging Rate	Up to 30 frames per second
Field of View	66° diagonal field of view
Imaging Features	<ul style="list-style-type: none"> <li>• Digital pan, digital tilt, vertical tilt, and swivel pan, and 4x digital zoom**</li> <li>• Auto focus, range from 6" to infinity</li> <li>• Automatic image adjustment with manual override</li> <li>• 16:9 widescreen</li> <li>• 24-bit color depth</li> </ul>

Lors des tests j'ai utilisé une caméra Microsoft HD-5000 avec une résolution de 1280 X 720 (16:9) en vidéo. Cette caméra ne présente ou peu de distorsion par rapport à son faible FOV. Vous pouvez voir dans la documentation ci-dessus qu'elle a un FOV **Diagonale** de 66°. Il faut faire attention, il y a plusieurs types de FOV.

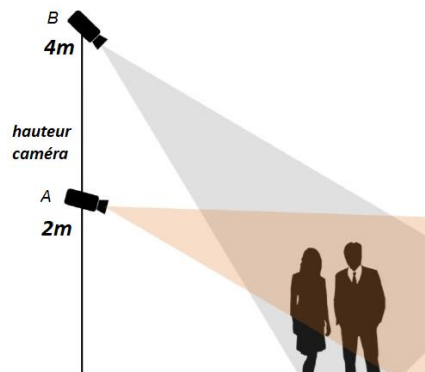


Le tableau ci-dessous permet d'estimer rapidement notre FOV en fonction des caractéristiques de notre caméra.

		rapport hauteur / largeur (Aspect Ratio)													
		1:1		5:4		4:3		3:2		16:10		16:9		2.39:1	
		H	V	H	V	H	V	H	V	H	V	H	V	H	V
FOV Diagonale	20°	14.2 °	14.2 °	15.7 °	12.6 °	16.1 °	12.1 °	16.7 °	11.2 °	17.0 °	10.7 °	17.5 °	9.9 °	18.5 °	7.8 °
	30°	21.5 °	21.5 °	23.6 °	19.0 °	24.2 °	18.3 °	25.1 °	16.9 °	25.6 °	16.2 °	26.3 °	15.0 °	27.8 °	11.8 °
	40°	28.9 °	28.9 °	31.7 °	25.6 °	32.5 °	24.6 °	33.7 °	22.8 °	34.3 °	21.8 °	35.2 °	20.2 °	37.1 °	16.0 °
	50°	36.5 °	36.5 °	40.0 °	32.5 °	40.9 °	31.3 °	42.4 °	29.0 °	43.2 °	27.8 °	44.2 °	25.8 °	46.6 °	20.4 °
	60°	44.4 °	44.4 °	48.5 °	39.7 °	49.6 °	38.2 °	51.3 °	35.5 °	52.2 °	34.0 °	53.4 °	31.6 °	56.1 °	25.1 °
	70°	52.7 °	52.7 °	57.3 °	47.3 °	58.5 °	45.6 °	60.5 °	42.5 °	61.4 °	40.7 °	62.8 °	37.9 °	65.7 °	30.2 °
	80°	61.4 °	61.4 °	66.5 °	55.3 °	67.7 °	53.4 °	69.8 °	49.9 °	70.9 °	48.0 °	72.4 °	44.7 °	75.5 °	35.9 °
	90°	70.5 °	70.5 °	76.0 °	64.0 °	77.3 °	61.9 °	79.5 °	58.0 °	80.6 °	55.8 °	82.1 °	52.2 °	85.4 °	42.2 °
	100°	80.2 °	80.2 °	85.9 °	73.3 °	87.3 °	71.1 °	89.5 °	66.9 °	90.6 °	64.6 °	92.2 °	60.6 °	95.4 °	49.4 °
	110°	90.6 °	90.6 °	96.2 °	83.5 °	97.6 °	81.2 °	99.8 °	76.8 °	100.9 °	74.2 °	102.4 °	70.0 °	105.6 °	57.7 °
	120°	101.5 °	101.5 °	107.0 °	94.5 °	108.4 °	92.2 °	110.5 °	87.7 °	111.5 °	85.1 °	113.0 °	80.7 °	115.9 °	67.5 °
	130°	113.2 °	113.2 °	118.3 °	106.5 °	119.5 °	104.3 °	121.5 °	99.9 °	122.4 °	97.3 °	123.7 °	92.9 °	126.4 °	79.2 °
	140°	125.5 °	125.5 °	130.0 °	119.5 °	131.1 °	117.5 °	132.7 °	113.5 °	133.5 °	111.0 °	134.7 °	106.8 °	136.9 °	93.4 °
	150°	138.5 °	138.5 °	142.1 °	133.6 °	143.0 °	131.9 °	144.3 °	128.4 °	144.9 °	126.4 °	145.8 °	122.7 °	147.6 °	110.5 °
	160°	152.0 °	152.0 °	154.6 °	148.5 °	155.1 °	147.2 °	156.1 °	144.7 °	156.5 °	143.2 °	157.1 °	140.4 °	158.4 °	130.9 °
	170°	165.9 °	165.9 °	167.2 °	164.1 °	167.5 °	163.4 °	168.0 °	162.1 °	168.2 °	161.3 °	168.5 °	159.8 °	169.2 °	154.5 °
	180°	180.0 °	180.0 °	180.0 °	180.0 °	180.0 °	180.0 °	180.0 °	180.0 °	180.0 °	180.0 °	180.0 °	180.0 °	180.0 °	180.0 °

Nous avons donc avec un FOV Diagonale de **66°** en rapport hauteur/largeur 16:9 un FOV Horizontale d'environ 58° et un FOV Verticale de 35°.

### 2.2.3 Angle d'inclinaison de la caméra par rapport à l'objet

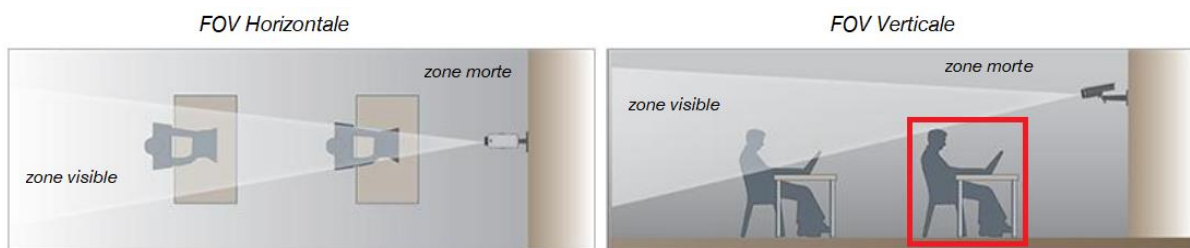


Il faut privilégier une caméra placée en hauteur, cette vue en perspective permet de faciliter la détection et le suivi de personnes proches. Une vue de face est plus appropriée pour l'identification de détails spécifiques liés à la cible (visage, détails des vêtements, etc.).

Cette partie sera à prendre en compte dans le choix de l'algorithme. Une position en hauteur sera plus appropriée pour un algorithme de type **détection**, alors qu'une caméra plus basse sera pour de la **reconnaissance**.

### 2.2.4 Les angles morts

La "dead zone" «zone morte» ou angle mort d'une caméra est la zone où la caméra ne peut pas voir et doit être prise en compte lors de l'installation de notre système.



Dans l'exemple ci-dessus on voit que le FOV Horizontale est suffisant pour la détection des deux personnes, cependant le FOV Verticale est trop faible, la personne la plus avancée est dans l'angle mort de la caméra. Dans ce scénario on voit que les réglages ne sont pas appropriés si la fonction principale du système est de détecter les deux personnes dans la pièce.

Piste d'amélioration :

- FOV Vertical trop faible
- Position centrale et hauteur à revoir
- Angle d'inclinaison de la caméra trop faible

Le champ de vision du système doit être vérifié (horizontalement et verticalement) et étalonné en fonction de la pièce et du scénario de fonctionnement.

Si la gestion des angles morts est une priorité, il faudra alors privilégier l'utilisation de plusieurs caméras. Dans notre cas, ce n'est pas une priorité.

### 2.2.5 Luminosité

Dans certaines conditions les questions d'éclairage sont cruciales, il est généralement facile et efficace en termes de coûts d'ajouter un éclairage supplémentaire, pour assurer les conditions d'éclairage nécessaires à la bonne détection. Il faut donc être vigilant aux différents changements de luminosité durant la période de la journée (matin, midi...etc).

### 2.2.6 Les périodes sombres / très lumineuses

Lors de périodes sombres, l'image captée par la caméra sera sujette à une faible luminosité appelée sous-exposition. On aura à ce moment-là une perte de détails dans l'image qui se traduira par une analyse d'image moins précise et une potentielle erreur de prédiction du nombre de personnes.

Pendant la période très lumineuse, la caméra capte beaucoup de lumière, le phénomène ici est une surexposition. Pareil que pour la période sombre, cela donne lieu à une perte de détails et donc une mauvaise estimation du nombre de personnes.

On peut considérer que l'éclairage de notre pièce est régulé par des étudiants pendant les horaires d'ouverture de l'école.

Il faut savoir que, les webcams et une bonne partie des caméras de maintenant ont une auto-régulation de ces deux phénomènes et maintiennent une image correcte dans la plupart des cas, saufs extrêmes.

Lorsque l'école n'est pas ouverte, la lumière ne sera donc plus régulée par les élèves, mais ce n'est pas gênant sachant que le système ne devrait pas détecter de personne dans la pièce ([Cf. 3.1.2 Optimisation CPU](#))



*Figure 3 Importance de l'éclairage sur la qualité d'image*

## 2.2.7 L'autofocus

L'autofocus, en français "mise au point automatique" est une fonction qui permet de régler automatiquement l'optique d'un objectif pour mettre au point un sujet.

La plupart des caméras modernes possèdent ce système, dans notre projet c'est aussi bien un avantage, qu'un inconvénient.

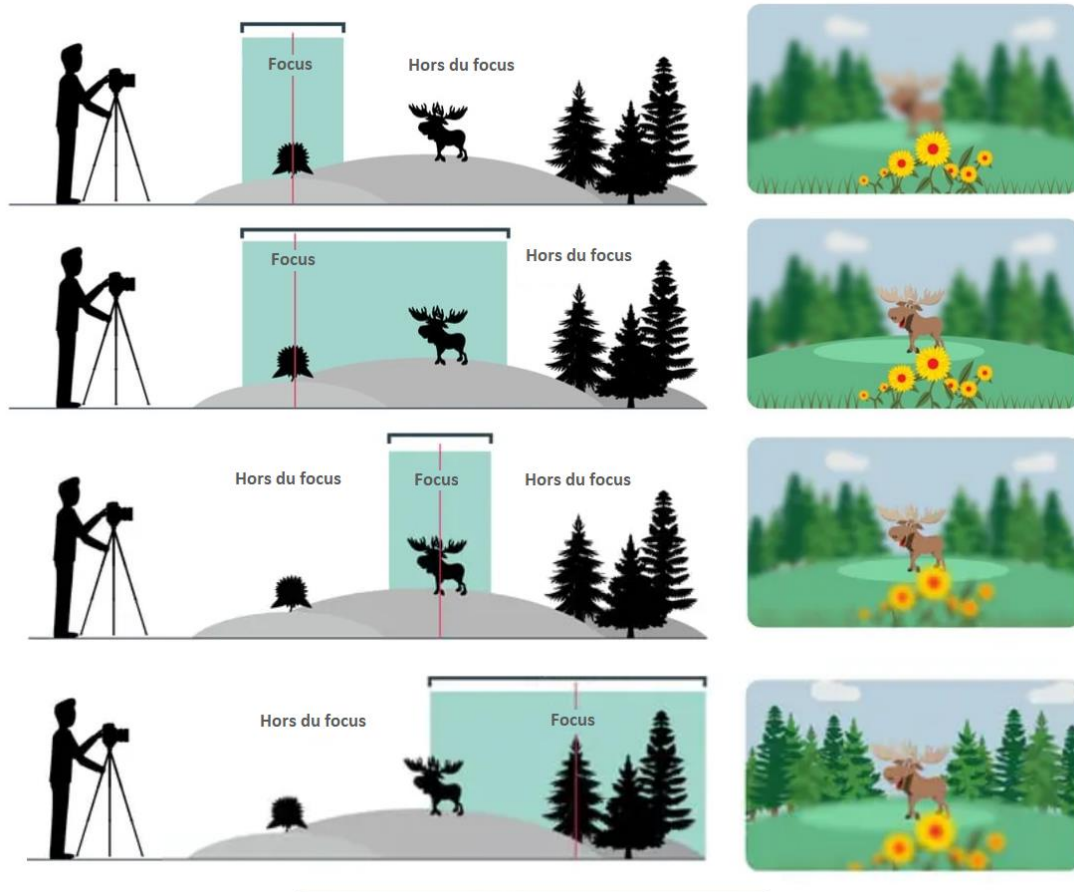


Figure 4 Impact du focus

Comme on voit sur la figure ci-dessus, le focus permet donc de faire la mise au point sur certaines zones, mais au détriment des autres. Dans notre projet cela va permettre d'avoir une meilleure détection des personnes proches et lointaines de la caméra. On va arriver à avoir plus de détails sur l'image surtout sur les personnes.

Mais pour nous, l'autofocus est aussi un inconvénient. Lorsque nous récupérons toutes les images issues de la caméra pour la détection d'objet, nous récupérons aussi les moments où le focus est en train de se faire. Cela donne des images très floues et donc inutilisables.

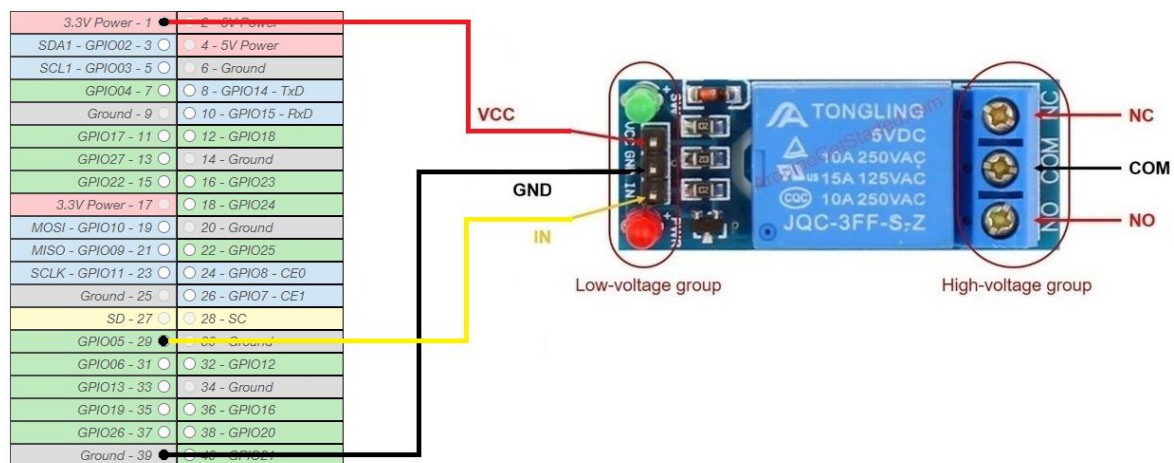
## 2.2.8 Conclusion caméra

Pour un choix optimum de la caméra dans notre projet il faut prendre en compte tous les paramètres énoncés précédemment, il n'y a pas de formule magique, se sera toujours en fonction de la taille de la pièce et des conditions d'utilisation. On peut trouver un FOV optimisé pour notre pièce mais il faut avoir au préalable les dimensions. Le FOV à

utiliser dans une pièce de 20m<sup>2</sup> ne sera pas le même que pour une pièce de 80m<sup>2</sup>. Il faut privilégier une position de la caméra en hauteur et dans un coin de la pièce pour une meilleure couverture. Il faut prendre en compte la luminosité et les périodes de la journée si c'est une pièce avec des ouvertures. Nous allons devoir faire des tests en conditions réelles afin de voir les limites et les meilleurs réglages pour la pièce.

### 2.3 Relai

Pour le contrôle du voyant lumineux, j'ai choisi d'utiliser un relai contrôlé par une GPIO de la carte Raspberry. La carte ne pouvant pas délivrer assez de courant pour l'alimentation du voyant sur les GPIO (**16 mA recommandés, maximum 50mA**), il faut obligatoirement une commande en puissance.



Credit à ArduinoGetStarted.com pour l'image Relai

Figure 5 Schéma de câblage relai

- VCC » branché sur le 3.3V
- GND branché sur une masse (Ground)

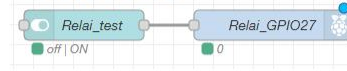
L'entrée « IN » du Relai est reliée à une GPIO de la carte (**GPIO 27 pin 13**)

Le "low-voltage group" est la partie commande côté Raspberry contrairement au "High-voltage group" qui permet de contrôler la partie puissance, ici notre témoin lumineux.



**Interface graphique Node-Red**

Relai\_test

**Flow Node-Red**

Relai\_test

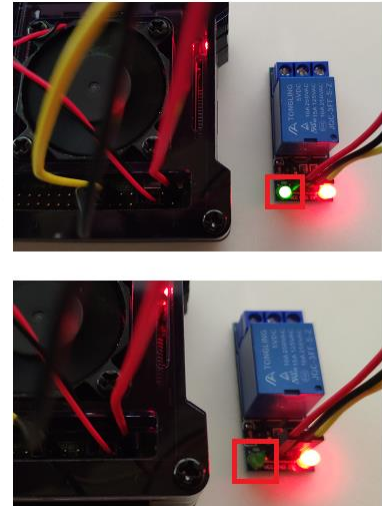
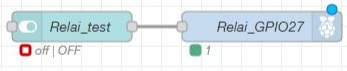


Figure 6 Preuve de concept control relai via Node-Red avec un Raspberry 4

Le test a été fait sans témoin lumineux (je ne l'ai pas encore) mais le relai commute bien lors du changement d'état sur l'interface graphique issue du serveur web NodeRED. La configuration des nodes sera en **annexe**.

**Il faut faire attention.** Lors de mon premier test avec un script python, le relai fonctionnait avec le VCC en **5V**. Cela ne fonctionne pas avec cette même tension sur NodeRED, il faut bien alimenter le relai en **3.3V** pour établir une tension de seuil correct autrement il ne commute plus.

- **Utilisation avec Python VCC -> 5V**
- **Utilisation sur Node-Red VCC -> 3.3V**

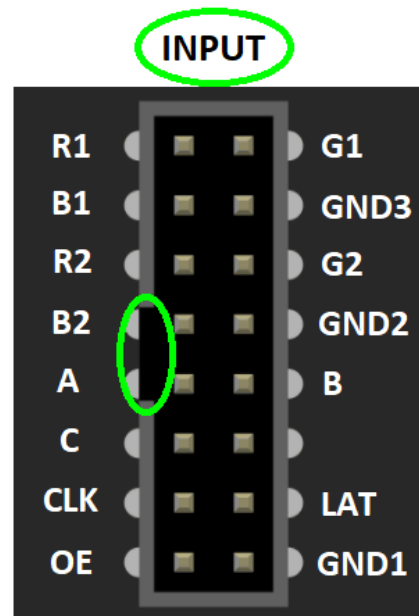
## 2.4 Matrice LED

Pour la partie affichage, j'ai choisi d'utiliser une matrice LED 32x16 Adafruit. Cette solution a déjà été mise en place par des internautes sur une Raspberry PI 4. La faisabilité est donc garantie. La matrice LED est une solution peu chère mais gourmande en énergie. Cette solution a été retenue pour sa facilité de mise en place et surtout pour sa visibilité. L'affichage du nombre de personnes doit être assez gros et surtout lisible. La consommation n'était pas un critère important puisque le système d'alimentation va être relié directement au réseau.

La connexion entre la Raspberry PI 4 et la matrice LED se fait de la façon suivante :

	3.3V Power - 1	2 - 5V Power
	SDA1 - GPIO02 - 3	4 - 5V Power
	SCL1 - GPIO03 - 5	6 - Ground
LAT	GPIO04 - 7	8 - GPIO14 - Tx D
GND2	Ground - 9	10 - GPIO15 - Rx D
CLK	GPIO17 - 11	12 - GPIO18
G1	GPIO27 - 13	14 - Ground
A	GPIO22 - 15	16 - GPIO23
	3.3V Power - 17	18 - GPIO24
B2	MOSI - GPIO10 - 19	20 - Ground
G2	MISO - GPIO09 - 21	22 - GPIO25
R1	SCLK - GPIO11 - 23	24 - GPIO8 - CE0
GND1	Ground - 25	26 - GPIO7 - CE1
	SD - 27	28 - SC
	GPIO05 - 29	30 - Ground
	GPIO06 - 31	32 - GPIO12
	GPIO13 - 33	34 - Ground
	GPIO19 - 35	36 - GPIO16
	GPIO26 - 37	38 - GPIO20
	Ground - 39	40 - GPIO21

Raspberry PI 4



Matrice LED 32x16

Nous allons utiliser 13 GPIO's de la raspberry PI 4 et les connecter sur le port J-IN de la matrice LED. Il y a plusieurs façons de la connecter, soit en "regular" avec des câbles fils par fils, soit en utilisant un [rgb matrix HAT](#)<sup>4</sup> (une carte faite exprès pour connecter des matrices LED) en utilisant la nappe incluse avec la matrice LED (connecteur **HUB75**).



Figure 8 HAT matrice LED pour Raspberry

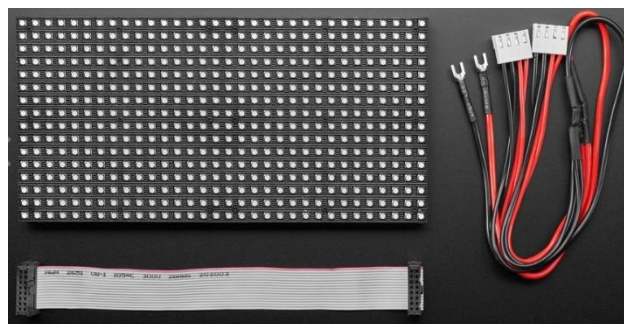


Figure 8 Contenu matrice LED

La consommation totale de la matrice LED ne devrait pas excéder les 2A bien qu'en réalité, le système lumineux ne sera pas utilisé à 100%. ([CF. Étude de la consommation](#))

Pour le contrôle de la matrice LED nous allons utiliser l'API de HZeller qui nous permet l'utilisation de nombreuses fonctions pour le pilotage de la matrice.

#API général

<https://github.com/hzeller/rpi-rgb-led-matrix>

#L'API pour l'utilisation du langage python

<sup>4</sup> <https://learn.adafruit.com/adafruit-rgb-matrix-plus-real-time-clock-hat-for-raspberry-pi> (valide au 31/12/20)



<https://github.com/hzeller/rpi-rgb-led-matrix/blob/master/bindings/python>

La partie qui nous intéresse est avec python.

Lors de la configuration de l'API il faudra faire attention à la connexion que nous avons faite entre la carte Raspberry et la matrice. Entre "regular" ou avec le HAT.

Dans notre code il faut bien mettre :

```
# Configuration for the matrix
options = RGBMatrixOptions()
options.rows = 32 -> 16
options.chain_length = 1
options.parallel = 1
options.hardware_mapping = 'regular' # If you have an Adafruit HAT: 'adafruit-hat'
```

#Option.rows à 16 puisque nous utilisons une matrice 32(largeur)x16(hauteur).

```
options.rows = 16
```

```
options.hardware_mapping = 'regular'
```

```
# regular          # Following this project wiring and using these PCBs
# adafruit-hat     # If you have a RGB matrix HAT from Adafruit
# adafruit-hat-pwm # If you have an Adafruit HAT with PWM hardware mod.
# regular-pi1      # If you have an old Pi1 and regular didn't work.
# classic          # (deprecated) Classic Pi1/2/. Not used anymore.
# classic-pi1      # (deprecated) Classic pinout on Raspberry Pi 1
```

## Affichage par image

```
#!/usr/bin/env python
import time
import sys

from rgbmatrix import RGBMatrix, RGBMatrixOptions
from PIL import Image

image = Image.open(image_file)
# Adapter l'image à notre matrice.
image.thumbnail((matrix.width, matrix.height), Image.ANTIALIAS)

matrix.SetImage(image.convert("RGB"))
```

Vous pouvez retrouver le code complet sur mon **github** :

[https://github.com/Layapanda/PFE\\_personnes/blob/main/display\\_image.py](https://github.com/Layapanda/PFE_personnes/blob/main/display_image.py)

### Affichage par pixel

```
import RPi.GPIO as GPIO
import time

delay = 0.000001

#Définition pins
GPIO.setmode(GPIO.BCM)
red1_pin = 17

#Définition des autres pins
#.....

#Setup pins
GPIO.setup(red1_pin, GPIO.OUT)

#Setup des autres pins
#.....

#initialisation de la taille de la matrice 32x16
screen = [[0 for x in xrange(32)] for x in xrange(16)]

#fonction permettant de changer la couleur d'un pixel à un endroit
def set_pixel(x, y, color):
    screen[y][x] = color
```

Vous pouvez retrouver le code complet sur mon **github** :

[https://github.com/Layapanda/PFE\\_personnes/blob/main/display\\_pixel.py](https://github.com/Layapanda/PFE_personnes/blob/main/display_pixel.py)

L'utilisation d'une image est le moyen le plus facile d'obtenir un affichage complexe sur la Raspberry 4 PI. Pour l'instant aucune des 2 méthodes n'est à privilégier pour l'affichage du nombre de personnes.

### 3. Étude logiciel

Après avoir beaucoup parlé de matériel, nous allons maintenant parler de la partie logicielle du système. Nous allons voir comment les différentes parties du système fonctionnent,

communiquent ensembles et comment les données sont structurées.

### 3.1 Carte Raspberry

#### 3.1.1 Paramètres de configuration

Pour le stockage des paramètres de configuration et les données sensibles sur la carte je recherche plusieurs qualités :

- ✓ Human readable/editable : pour réaliser des modifications à la main facilement
- ✓ Machine readable/editable : le principe de base, le logiciel doit pouvoir lire et éditer sa configuration.
- ✓ Évolutif en compatibilité ascendante : La montée en fonctionnalités doit être possible

J'ai choisi d'utiliser le format de données **JSON** qui dispose de toutes les qualités précédentes. L'avantage du json c'est qu'il est disponible dans tous les langages et il est facile d'utilisation. La gestion native du JSON par le langage Python 3 est un avantage non négligeable.

Les paramètres seront stockés dans un fichier au sein de la carte dans un format ressemblant à celui présenté ci-dessous, il se peut qu'il manque encore des données à ce stade du projet :

Le fichier sera situé dans **/pfe/parametre\_projet.json**

```
{
  "adresse_mail": "superprojet@gmail.com",
  "temps_limite": 30, //valeur en seconde
  "limite_personne": 12,
  "mdp": "test_projet_personne",
  "active_mail": true,
  "active_voyant": true
}
```

Seul le serveur pourra agir sur l'écriture du fichier. Les autres processus pourront savoir en tout temps les modifications effectuées dans le fichier. ([Cf. 3.3 Communication entre les parties](#))

#### 3.1.2 Optimisation CPU

Dans cette partie nous allons voir les méthodes que nous allons mettre en place pour diminuer le taux CPU lorsque le système sera en fonctionnement.

**La première méthode** sera de développer un « mode sleep », lorsque le système ne détecte plus personne pendant un long moment, le système fonctionnera plus lentement. Ce ralentissement permettra de diminuer le nombre d'images à analyser et ainsi baisser le taux CPU.

La **deuxième méthode** sera de réguler dynamiquement le nombre d'images à analyser en fonction du débit de personnes dans la salle. Dans les périodes creuses, on pourra demander moins d'analyses d'images et lorsque le débit de personnes augmente on augmente aussi le nombre d'analyses.

La **troisième méthode** sera d'utiliser l'heure de la carte afin de détecter les plages horaires d'ouverture et de fermeture de l'école.

### 3.2 Object detection

La partie object detection sera faite par l'API Tensorflow qui est conçu pour faciliter le machine learning sur les appareils en périphérie du réseau. Cela permet d'exécuter le code sur l'appareil directement sans connexion internet. Le fait de tout faire sur la machine directement améliore :

- ✓ Latence : pas d'allers-retour des données avec un serveur
- ✓ Confidentialité : toutes les données sont conservées sur l'appareil
- ✓ Connectivité : aucune connexion Internet n'est requise
- ✓ Consommation d'énergie : les connexions réseau sont très énergivores

A présent, qu'est-ce que le machine learning ? Le machine learning en français « **apprentissage automatique** » est l'étude des algorithmes informatiques qui s'améliorent automatiquement grâce à l'expérience. C'est un sous-ensemble de l'intelligence artificielle. Les algorithmes d'apprentissage automatique construisent un **modèle** basé sur des échantillons de données, appelés « **données d'apprentissage** », afin de faire des prédictions ou des décisions sans être explicitement programmés pour le faire.

Dans notre cas, Tensorflow a déjà beaucoup de modèles précréés (pré-entraînés) pour certaines utilisations. Ils sont alors déjà compilés et peuvent être utilisés

#### 3.2.1 Paramètres / données de sortie

**En entrée** le modèle prend un seul paramètre, une image. Cette image doit avoir une taille spécifique propre au modèle.

**En sortie** le modèle produit quatre tableaux, mappés aux indices 0-4. Les tableaux 0, 1 et 2 décrivent N objets détectés, avec un élément dans chaque tableau correspondant à chaque objet.

Indice	Nom	La description
0	Emplacements	Tableau multidimensionnel de [N][4] valeurs à virgule flottante entre 0 et 1, les tableaux internes représentant des cadres de délimitation sous la forme [haut, gauche, bas, droite]
1	Des classes	Tableau de N entiers (sortie sous forme de valeurs à virgule flottante) indiquant chacun l'index d'une étiquette de classe à partir du fichier d'étiquettes

2	Les scores	Tableau de N valeurs à virgule flottante entre 0 et 1 représentant la probabilité qu'une classe a été détectée
3	Nombre de détections	Valeur entière de N

Lorsqu'une image est ensuite fournie au modèle, il génère une liste des objets qu'il détecte, l'emplacement d'un cadre englobant contenant chaque objet et un score qui indique la certitude que la détection était correcte.

Par exemple, le modèle traite une image contenant 2 personnes.

Classe (identification)	But (confidence)	Emplacement
personne	0,92	[18, 21, 57, 63]
personne	0,88	[100, 30, 180, 150]
chaise	0,51	[6, 42, 31, 58]
personne	0,23	[42, 66, 57, 83]

#### Score de confiance

Pour interpréter ces résultats, nous pouvons regarder le score et l'emplacement de chaque objet détecté. Le score est un nombre entre 0 et 1 qui indique la certitude que l'objet a été réellement détecté. Plus le nombre est proche de 1, plus le modèle est confiant.

Pour l'exemple actuel, un seuil raisonnable est un score de 0,5 (ce qui signifie une probabilité de 50% que la détection soit valide). Dans ce cas, le dernier objet du tableau serait ignoré car ce score de confiance est inférieur à 0,5.

Classe (identification)	But (confidence)	Emplacement
personne	0,92	[18, 21, 57, 63]
personne	0,88	[100, 30, 180, 150]
chaise	0,51	[6, 42, 31, 58]
personne	0,23	[42, 66, 57, 83]

On peut alors voir deux problèmes :

- **Les faux positifs** (objets qui sont mal identifiés, ou les zones de l'image qui sont identifiées à tort comme des objets alors qu'elles ne le sont pas)
- **Les faux négatifs** (objets authentiques qui sont manqués parce que leur confiance était faible)

Par exemple, ici un élément a été identifié à tort comme une «**chaise**». Ceci est un exemple de faux positif qui pourrait être ignoré en sélectionnant un seuil approprié. Dans ce cas, un seuil de 0,6 (ou 60%) exclurait confortablement les faux positifs.

Le problème sera de trouver le bon compromis entre détecter le plus de personnes dans les images et ne pas avoir trop de faux positifs.

La partie emplacement des objets dans l'image n'est pas utilisée.

Ci-dessous le calcul du nombre de personnes en sortie du modèle avant traitement.

```
interpreter.set_tensor(input_details[0]['index'],input_data) #Calcul détection
interpreter.invoke()

classes = interpreter.get_tensor(output_details[1]['index'])[0] #Classe
scores = interpreter.get_tensor(output_details[2]['index'])[0] #Confidence

for i in range(len(scores)): #Boucle sur toute les détections de l'image
    if(labels[int(classes[i])] == "person"): #On regarde si c'est une personne
        if ((scores[i] > min_conf_threshold) and (scores[i] <= 1.0)):
            list.append(1) #Si la confidence est bonne j'ajoute une personne
        else:
            list.append(0) #Autrement je n'ajoute personne

nombre_de_personne = sum(list) #Somme de la liste donne le nombre de personne dans
la liste
```

Il faudra faire attention **lorsque aucune personne** n'est détectée dans l'image, cela peut être une image floue (donc une mauvaise détection) ou alors qu'il n'y a vraiment personne dans la pièce. Cette valeur devra être nuancée pour ne pas erroner le résultat des méthodes de filtrage.

On pourrait par exemple redemander une nouvelle détection.

### 3.2.2 Robustesse

On peut voir trois indices de robustesse des données :

- Comme on l'a vu précédemment, la **confiance** sur les résultats
- Le **mAP** des modèles
- Traitement après prédiction du modèle

Lors du choix du modèle sur le site de **Tensorflow**, on peut constater une colonne mAP. Le mAP (moyenne de précision moyenne) est le produit de la précision et du rappel lors de la détection des boîtes englobantes. C'est une bonne mesure combinée de la sensibilité du réseau aux objets d'intérêts et de sa capacité à éviter les fausses détections. Plus le score mAP est élevé, plus le réseau est précis, mais cela se fait au détriment de la vitesse d'exécution.

Il faut prendre en compte que le mAP de Tensorflow est calculé sur des cartes Nvidia Titan

qui sont bien plus puissantes que notre carte Raspberry. L'indice de vitesse n'est donc pas réel pour notre système. Il nous permet juste de comparer les modèles entre eux.

### Traitement après prédiction du modèle

L'idée du traitement après prédiction est de minimiser les fausses erreurs et proposer un nombre de personnes dans la pièce plus stable.

Voici les idées qui pourraient être mises en place en tant que filtres. Prendre plusieurs prédictions du modèle sur une période de temps les **stocker dans une liste**. Il faut toujours afficher une valeur entière.

- Afficher la **moyenne** de la liste **arrondie** à l'entier supérieur
- Afficher la valeur avec le plus **d'occurrence**
- Afficher la **médiane** de la liste
- Accordé plus **d'importance** aux **dernières** valeurs qu'aux premières
- Utilisation de la **précédente prédiction** comme valeur de référence de décision (en utilisant le taux de confiance de l'ancienne prédiction)

Ces traitements peuvent être faits sur une liste réactualisée ou alors en glissement de valeur. On peut rapidement voir certains **problèmes** de ces **méthodes** en fonction de la **situation**. Exemple avec 6 images prises sur 2 secondes :

[1,1,1] , [3,3,2]



nombre issu des images prises pendant la première seconde



nombre issu des images prises pendant la deuxième seconde

Résultats des méthodes :

- Moyenne entière – 2 personnes
- **Plus d'occurrences – 1 personnes**
- Médiane – 1.5 personnes soit 2 personnes

On ne peut pas vraiment tirer de conclusion sur une seule série de données et estimer la meilleure méthode, pour cela on va devoir faire des tests sur des vidéos de "références" avec une forte et faible affluence dans une salle et d'analyser les résultats. Travailler sur des vidéos va nous permettre de répéter les tests dans les mêmes conditions. Ce qui est pour l'instant un peu **compliqué** à mettre en place avec le **confinement**...

### 3.3.3 Choix du modèle

Pour le choix du modèle nous avons plusieurs critères à définir. Un modèle se caractérise par sa rapidité de traitement et sa précision.

La première idée a été de créer mon propre modèle, de l'entraîner pour seulement détecter des personnes et non une multitude d'objets. Mais après avoir parlé avec plusieurs professeurs et camarades de classe qui avaient utilisé l'object detection sur de précédents

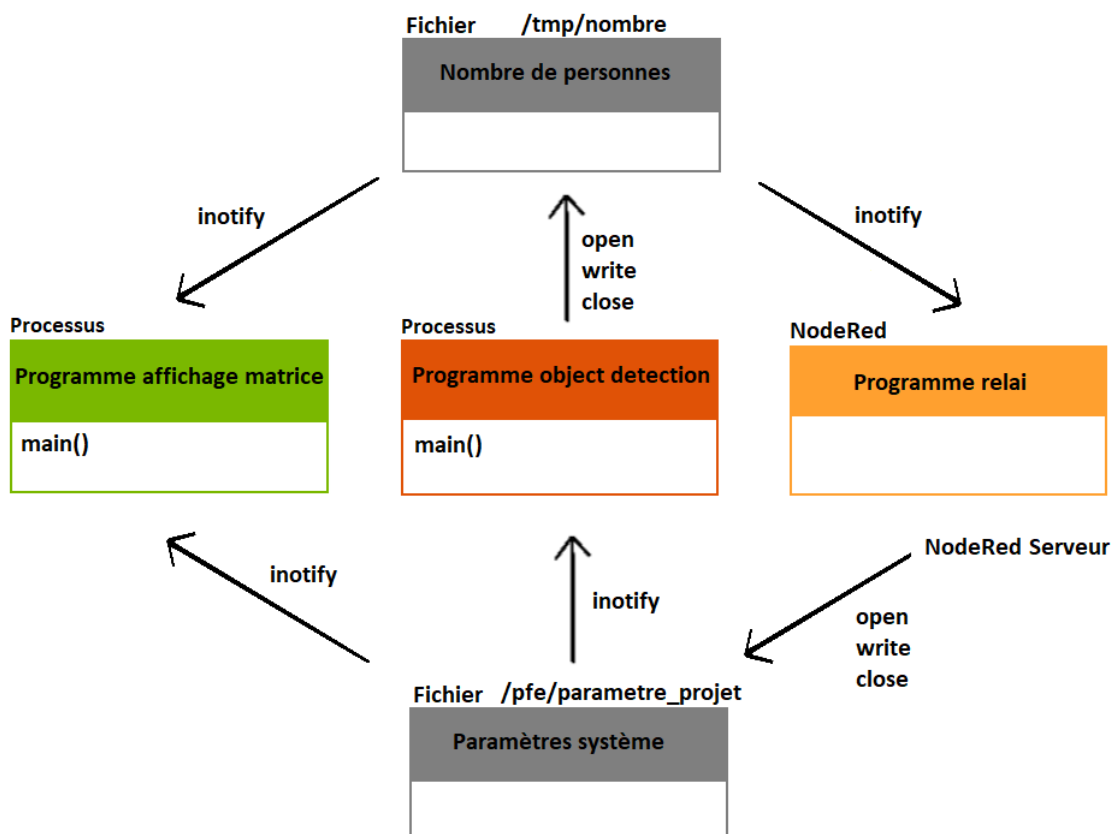


projets d'ingénieur ; la conclusion a été d'oublier. Il est assez facile de spécialiser un modèle déjà optimisé pour renforcer un type de détection. Par exemple avec une tasse c'est un cas qui marche très bien ou avec des objets qui ne changent pas trop de forme / couleur. Les personnes sont un cas très complexe en vue de tous les paramètres à prendre en compte. Ce n'est pas impossible, mais cela demandera beaucoup de temps rien que pour avoir un modèle assez fiable. Mais le point important c'est son temps de détection, il ne faut pas oublier que nous avons une puissance de calcul très limitée.

Pour bien choisir le modèle à utiliser il faut déjà écarter ceux qui ne sont pas utilisables. Par exemple les modèles très précis demandent une puissance de calcul importante, sur notre Raspberry cela se traduit par un temps d'exécution très long. On va devoir donc faire un compromis entre rapidité et précision. Ci-dessous la sélection des modèles par compromis.

Nom du modèle	Vitesse Nvidia Titan (ms)	mAP
ssd_mobilenet_v1_coco	30	21
ssd_mobilenet_v2_coco	31	22
ssdlite_mobilenet_v2_coco	27	22
ssd_inception_v2_coco	42	24

### 3.3 Communication entre les parties



Inotify est un mécanisme d'attente passive sur des fichiers ou répertoires. Un programme

peut attendre un ou plusieurs évènements sans consommer de CPU et il sera débloqué par le noyau lorsque ladite action sera effectuée sur le fichier. C'est aussi un moyen de réaliser une communication interprocessus autre que par un pipe, un socket ou un tube nommé.

Lien vers la documentation de inotify sous python : <https://pypi.org/project/inotify-simple/?fbclid=IwAR278dXvG3MPvXMC05J3RAmTMyOkmdYY1EnJhZ2yHPNDLHbqcPKVOd6RQ5o> (valide au 03/01/21)

La synchronisation entre les processus sera plus simple avec le Inotify que d'utiliser plusieurs threads dans le code principal pour la gestion du relai et de la matrice LED. Cela fait un découpage en sous-parties plus simple. Cela permettra aussi d'éviter des lectures en continu pour la récupération des informations et les potentiels problèmes de synchronisation (problèmes rédacteurs/lecteurs).

Le principe est d'avoir **deux fichiers** :

- Un pour le nombre de personnes
- Un pour les paramètres de configuration du système

Ces deux fichiers vont être les points clefs de la synchronisation de notre système.

Une fois que le programme object detection aura fait son traitement de l'image et aboutit à un nombre de personnes, il va regarder si ce nombre n'est pas le même que son précédent traitement. Si le nombre est différent, il va alors écrire dans le fichier **/tmp/nombre** la nouvelle valeur. Les programmes Relais et matrice LED vont être notifiés d'un changement de valeur dans le fichier. La matrice LED va alors mettre à jour la valeur de son affichage. Ce principe permet aussi de rafraichir l'affichage que lorsque c'est utile.

**La partie relai** va être un peu plus complexe. Il faudra prendre en compte une gestion du temps. Lors de la notification de changement de valeurs, il va devoir regarder si ce nouveau nombre est supérieur à la limite de personnes dans la salle. Si cette valeur est supérieure il faudra fermer le **relai** et allumer le **témoin lumineux**. Si cette limite de personne est dépassée pendant une durée **"temps\_limite"** nous allons devoir envoyer un mail à l'adresse mail **"adresse\_mail"**.

Lors d'un changement de paramètres sur l'IHM WEB Node-Red, la nouvelle configuration sera écrite dans le fichier **/pfe/parametre\_projet**. Les autres processus seront notifiés du changement et pourront mettre à jour leurs paramètres localement.

**La partie matrice** rafraichira son affichage à chaque fois que le fichier **/tmp/nombre** sera modifié.

#### 4. Étude de la consommation

Dans cette partie nous allons étudier la consommation générale de notre système. D'après la documentation la carte Raspberry doit être alimentée par une source de **5V 3A** :

- **5V DC** via connecteur **USB-C** (minimum 3A)
- **5V DC** via le connecteur **GPIO** (minimum 3A)

La consommation à vide est entre **700 mA – 900 mA**. Lors de la phase de démarrage, on peut constater des pics à **1,15A**.



On obtient une consommation d'environ **1,5A** avec le programme de détection et on peut constater une **forte augmentation de la température** atteignant les **84°C** qui sur le long terme pourrait endommager la carte. Pour pallier ce problème de température, on peut mettre des **dissipateurs thermiques** et un **petit ventilateur** alimenté en 5V par la carte. Avec ce système on arrive à stabiliser la température à **45°C** pendant les tests.

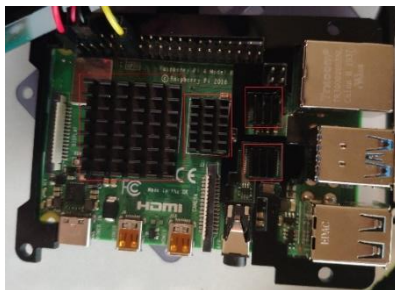
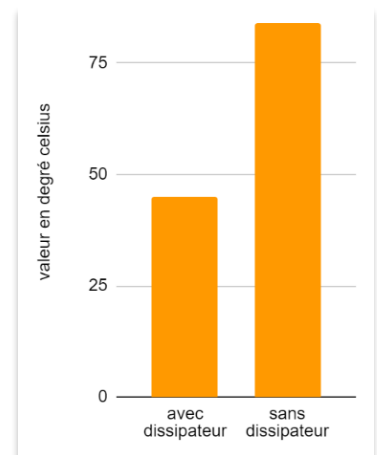


Figure 10 Dissipateurs thermiques

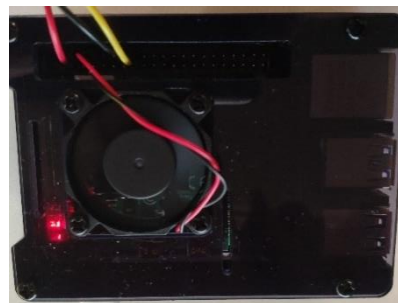


Figure 9 Ventilateur

La datasheet de ma caméra n'a pas d'informations sur sa consommation. D'après la carte Raspberry, si j'utilise moins de 500mA sur les lignes 5V, je peux descendre mon alimentation à 2,5A. Je ne pense pas que la consommation de la caméra soit supérieure à 500mA sachant que les ports USB des ordinateurs ne proposent que du 5V 500mAh.

Le relai en position fermé consomme **5mAh**.

Pour la partie matrice LED, la documentation préconise une alimentation de 5V et 2A lorsque toutes les LED sont allumées. Un cas que nous n'allons pas rencontrer lors de l'affichage du nombre de personnes. Après il n'est pas exclu d'avoir un affichage plus complet et d'ajouter par exemple l'heure actuelle ou la limite de personnes.

Si on prévoit un adaptateur secteur pouvant fournir **5V 5A** continu, alors ce sera suffisant pour l'alimentation de la carte, le relai (GPIO + partie basse puissance), la caméra et la matrice en même temps. On pourrait cependant dissocier les alimentations :

- **5V 2.5-3A** pour la carte, le relai et la caméra
- **5V 2A** pour la matrice.

## 5. Base de données (optionnel)

La partie base de données du type relationnel permettra de stocker des valeurs de personnes en fonction de l'heure. Cela nous permettra d'avoir un support visuel de l'affluence dans la salle, ces données permettront l'amélioration du système (débit de personnes, heure d'affluence...). La BDD sera composée de 1 seule table comme suit :

nombre_personnes mesure
date_insertion : timestamp
# nbpersonnes : int(2)

Le script de conception de la base de données est disponible sur mon **github** :

[https://github.com/Layapanda/PFE\\_personnes/blob/main/connection.py](https://github.com/Layapanda/PFE_personnes/blob/main/connection.py)

**date\_insertion** est un timestamp, c'est un type de données temporelles qui détient la combinaison de la date et l'heure. Le format d'un timestamp est le suivant :

**AAAA-MM-JJ HH: MM: SS**

**nbpersonnes** est un entier naturel correspondant aux nombres personnes

Voici un exemple d'insertion dans la base de données d'un timestamp et un nombre de personnes sous **NodeRED**.

```
var nb_personnes = msg.payload
msg.topic = "Insert into Mesure value (CURRENT_TIMESTAMP," + nb_personnes + ")"
return msg;
```

✓ Affichage des lignes 0 - 3 (total de 4, traitement en 0,0000 seconde(s).)

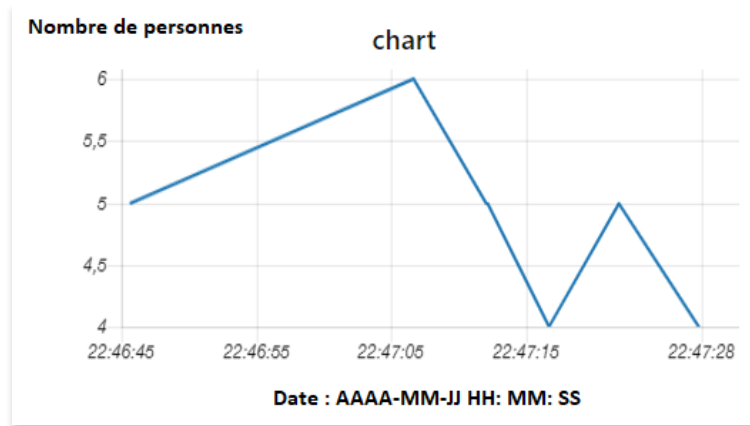
`SELECT * FROM `mesure``

+ Options

	date_insertion	nbpersonnes
<input type="checkbox"/> Éditer Copier Supprimer	2021-01-14 22:36:00	2
<input type="checkbox"/> Éditer Copier Supprimer	2021-01-14 22:36:55	3
<input type="checkbox"/> Éditer Copier Supprimer	2021-01-14 22:37:00	1
<input type="checkbox"/> Éditer Copier Supprimer	2021-01-14 22:37:04	5

Figure 11 Exemple de données coter BDD

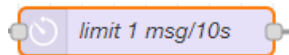
Les données permettront d'avoir un affichage comme celui-ci sur notre IHM WEB :



Ce type de graphique est très facile à mettre en place sous NodeRED avec le nœud.



Ce qui est intéressant avec la mise en place de l'insertion des données côté serveur NodeRED, c'est que nous allons pouvoir contrôler la quantité de données à introduire dans la base de données très simplement. On sera notifié que de nouvelles données sont disponibles par le inotify du fichier **/tmp/nombre** mais on pourra prendre en compte ou non cette info avec le nœud.



## 6. Serveur WEB de pilotage

Le serveur WEB correspond uniquement à une interface de configuration graphique qui est compatible PC et mobile. Sur cette interface, on pourra modifier les paramètres des périphériques du système. Globalement on ne disposera que d'un ensemble de boutons, sliders, listes et champs de textes pour piloter les éléments. Étant novice en site WEB, j'ai proposé de partir sur une solution compatible avec les systèmes embarqués pour l'IOT et l'automatisme. Cette solution s'appelle NodeRED. On dispose d'un dashboard sur lequel on va pouvoir créer des interfaces graphiques et d'une interface d'administration pour la création de chaînes d'événements appelées « flows ». La programmation du site WEB est alors graphique et comprend front-end et back-end dans un seul package et un seul serveur. C'est donc pour moi (je suis complètement étranger au WEB) un excellent moyen de réaliser les objectifs en un minimum de temps. La solution est alors robuste et simple à maintenir et à faire évoluer. Avec NodeRED, on peut aisément agir sur des processus et fichiers du système.

## GLOSSAIRE

Dans cette partie on doit trouver, classés par ordre alphabétique, les définitions des termes courants utilisés, des termes techniques, abréviations, sigles et symboles employés dans l'ensemble du document.

**API** : Programming Interface ou en français interface de programmation d'application est une solution informatique qui permet à des applications de communiquer entre elles et de s'échanger mutuellement des services ou des données.

**BDD** : Une base de données (que nous nommons BDD par commodité) est une collection d'informations organisées afin d'être facilement consultable, gérable et mise à jour.

**BeagleBone** : La BeagleBoard est une carte électronique industrielle de type ordinateur de faible puissance. Il s'agit d'un matériel libre produit par Texas Instruments en collaboration avec Digi-Key. La BeagleBoard a également été conçue en ayant à l'esprit le développement de logiciels open source.

**DC** : Direct current en anglais est le courant continu

**IHM** : L'interface homme-machine (IHM) est l'interface utilisateur qui relie l'opérateur au dispositif de commande d'un système industriel.

**LED** : Light Emitting Diode, ou diode électroluminescente est un composant électronique pouvant émettre de la lumière sous l'effet du courant électrique qui la traverse. C'est un composant polarisé dans lequel le courant ne peut circuler que dans un sens à la façon d'une diode.

**Object Detection** : La détection d'objets est une technique de vision par ordinateur qui nous permet d'identifier et de localiser des objets dans une image ou une vidéo. Avec ce type d'identification et de localisation, la détection d'objets peut être utilisée pour compter les objets dans une scène et déterminer et suivre leurs emplacements précis, tout en les étiquetant avec précision.

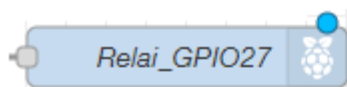
**OS** : (Operating system) Un système d'exploitation, ou «OS», est un logiciel qui communique avec le matériel et permet à d'autres programmes de s'exécuter.

## **BIBLIOGRAPHIE**



## ANNEXE

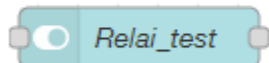
## Configuration nodes NodeRED



## Properties

● Pin

3.3V Power - 1 <input type="radio"/>	2 - 5V Power <input type="radio"/>
SDA1 - GPIO02 - 3 <input type="radio"/>	4 - 5V Power <input type="radio"/>
SCL1 - GPIO03 - 5 <input type="radio"/>	6 - Ground <input type="radio"/>
GPIO04 - 7 <input type="radio"/>	8 - GPIO14 - TxD <input type="radio"/>
Ground - 9 <input type="radio"/>	10 - GPIO15 - RxD <input type="radio"/>
GPIO17 - 11 <input type="radio"/>	12 - GPIO18 <input type="radio"/>
GPIO27 - 13 <input type="radio"/>	14 - Ground <input type="radio"/>
GPIO22 - 15 <input type="radio"/>	16 - GPIO23 <input type="radio"/>
3.3V Power - 17 <input type="radio"/>	18 - GPIO24 <input type="radio"/>
MOSI - GPIO10 - 19 <input type="radio"/>	20 - Ground <input type="radio"/>
MISO - GPIO09 - 21 <input type="radio"/>	22 - GPIO25 <input type="radio"/>
SCLK - GPIO11 - 23 <input type="radio"/>	24 - GPIO8 - CE0 <input type="radio"/>
Ground - 25 <input type="radio"/>	26 - GPIO7 - CE1 <input type="radio"/>
SD - 27 <input type="radio"/>	28 - SC <input type="radio"/>
GPIO05 - 29 <input checked="" type="radio"/>	30 - Ground <input type="radio"/>
GPIO06 - 31 <input type="radio"/>	32 - GPIO12 <input type="radio"/>
GPIO13 - 33 <input type="radio"/>	34 - Ground <input type="radio"/>
GPIO19 - 35 <input type="radio"/>	36 - GPIO16 <input type="radio"/>
GPIO26 - 37 <input type="radio"/>	38 - GPIO20 <input type="radio"/>
Ground - 39 <input type="radio"/>	40 - GPIO21 <input type="radio"/>



## Properties

On Payload Off Payload