ProTIS
Rapport technique
Vision industrielle

BELGUERRAS Orane
PETOLAT Marie
SUREAU Arthur
Vignal-Rakotoarisolo Sébastien

6 avril 2023

Table des matières

1 Introduction

Nous sommes une équipe de quatre futur.e.s ingénieur.e.s, actuellement en deuxième à **l'Institut d'Optique Graduate School (IOGS)**. Nous détaillons dans le présent rapport le projet **"Vision industrielle"** effectué pour le compte de la société **SOLEC**, experte en systèmes électroniques et partenaire de l'IOGS.

Ce projet a eu pour objectif général de renforcer nos qualités et capacités techniques en électronique acquises depuis les modules de CéTI et de IeTI de l'année dernière.

Plus concrètement, notre projet concerne l'automatisation en miniature d'un tri industriel. Ce système, une fois automatisé, pourra être utilisé à plus grande échelle dans une chaîne de production. Nous voulons trier grâce à la "computer vision" des cubes qui défilent devant une caméra grâce à un convoyeur, un tapis roulant, en fonction de leur couleur. La caméra va détecter la couleur du cube qui passe devant elle, l'information sera envoyée à une carte nucléo qui ira commander un bouton poussoir pour éjecter ou non le cube du tapis selon les sollicitations de l'utilisateur.

1.1 Notice d'utilisation

Le système est conçu de telle manière à ce qu'il soit entièrement automatisé. Pour lancer le système de tri, il vous faut compiler le programme python à votre disposition sur l'interface **"Spyder"**. Cela permettra de lancer l'acquisition de la caméra et de faire rouler le convoyeur. Vous pouvez désormais déposer des cubes de couleur sur le tapis, si possible avec un peu d'espace entre chaque cube. Observez maintenant la magie s'opérer !

1.2 Principe - Montage & Illustration

FIGURE 1 – Tapis roulant et son moteur

Le tapis roulant fonctionne grâce à un moteur pas à pas qui est lui-même commandé par nos soins grâce à des codes informatiques envoyés à une carte nucléo.

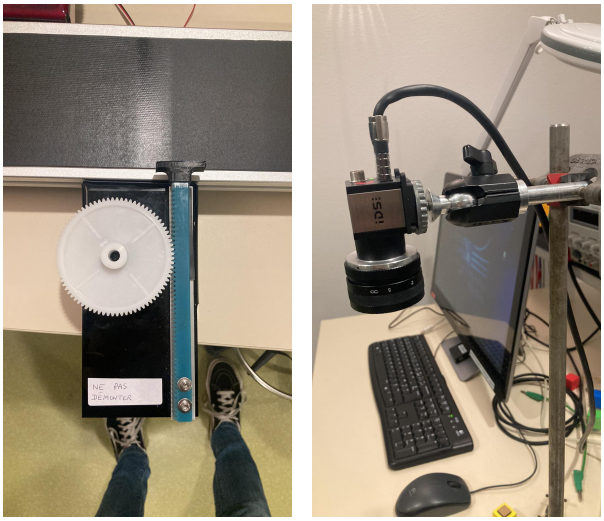

FIGURE 2 – Pistons avec servomoteurs et caméra

Les pistons,eux, sont commandés par des servomoteurs. La caméra est une **IDS UI-3240CP-C-HQ R2**.

Nous vous montrons ci-dessous le système en pleine action. Le montage électronique est situé en bas à gauche de l'image. Le cube rouge est avancé par le tapis roulant et les deux boutons poussoirs sont prêts à éjecter le cube selon la commande de l'utilisateur. La caméra n'est pas dans la photo mais nous pouvons apercevoir son pied.

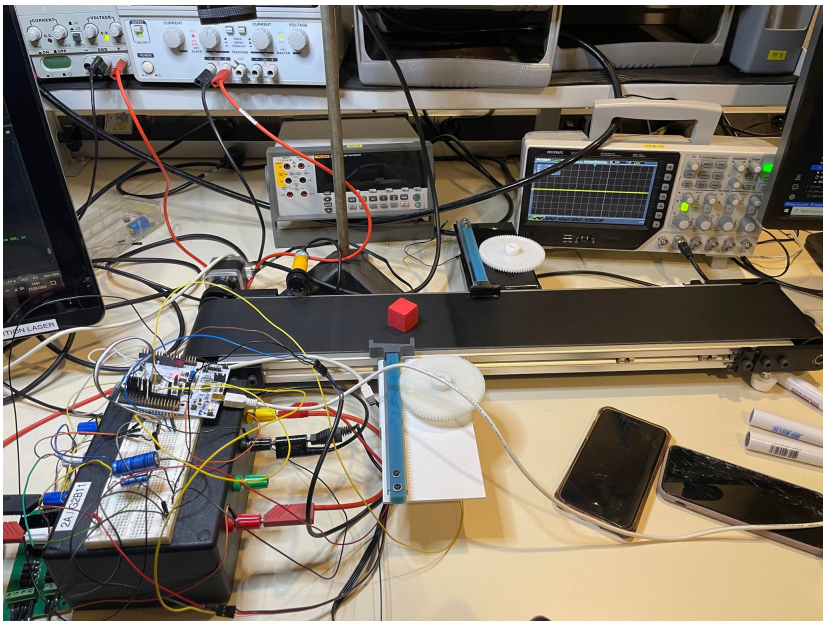

FIGURE 3 – Montage électronique - Système en fonctionnement

1.3 Cahier des charges

Notre cahier des charges initial est résumé dans le tableau suivant :

Couleur	Trier un minimum de trois couleurs : **Rouge, Vert et Jaune**
Rapidité	Analyse d'au moins 10 pièces par minute
Précision	Une erreur de détection d'une pièce sur 1000
Interface	Interface Humain-Machine permettant à un.e utilisateur.trice de choisir la couleur

TABLE 1 – Cahier des charges

L'interface évoquée ci-dessus pourra également :

1) Afficher la quantité N de pièces triées pendant un laps de temps t.
2) Afficher le temps moyen τ passé par le système pour éjecter une pièce.

1.4 Schéma fonctionnel

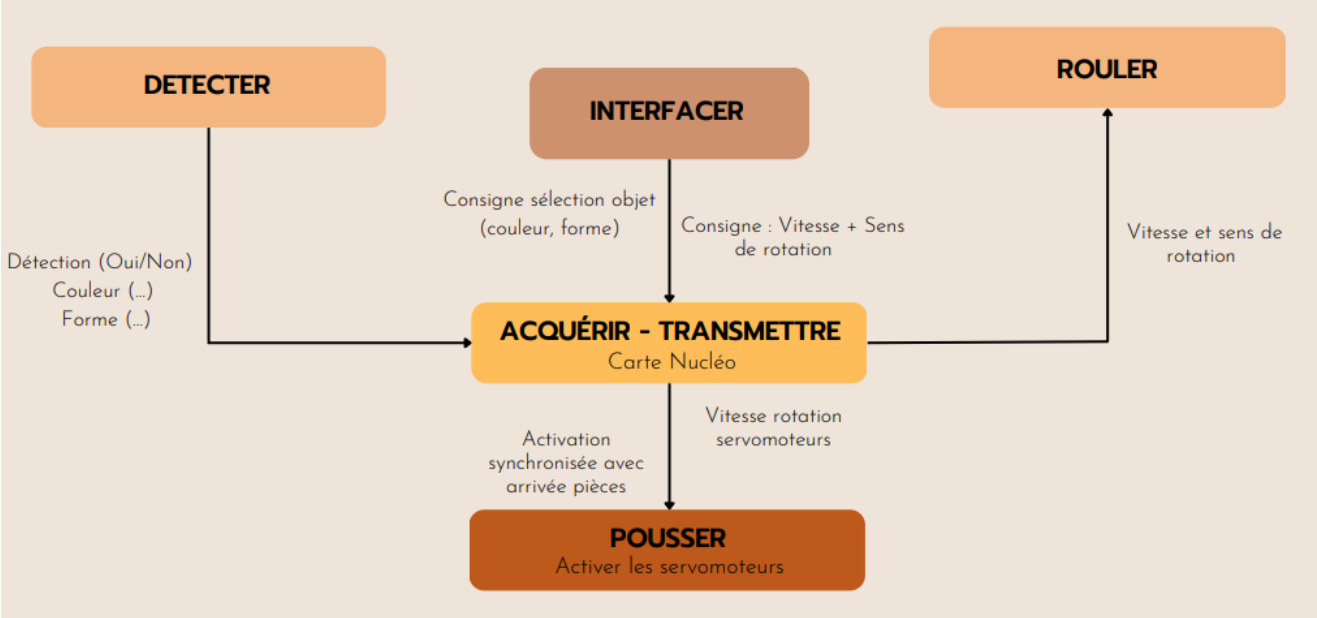

FIGURE 4 – Schéma fonctionnel de notre système

Ce schéma fonctionnel va illustrer les parties qui suivent. Dans un premier temps, nous allons détailler la détection des cubes et de leur couleur grâce à la caméra (**DETECTER**). Ensuite, nous évoquerons un peu plus en détail le fonctionnement du tapis roulant et des boutons poussoirs (**ROULER & POUSSER**). Toutes ces fonctions sont reliées à la carte nucléo qui transmet les informations aux différents composants (**ACQUERIR & TRANSMETTRE**).

2 Détection de la couleur avec la caméra

La détection de la couleur des cubes avec la caméra et l'envoi de l'information à la carte Nucléo se fait en 6 étapes :

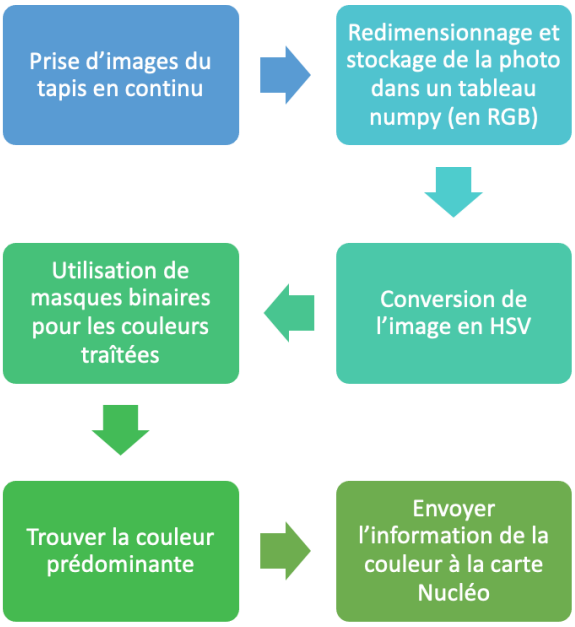

FIGURE 5 – Schéma des 6 étapes nécessaire à la détection de couleur

Toutes ces fonctions sont réalisées dans une boucle while pour permettre de faire les actions en continu.

2.1 Prise d'image du tapis en continu

Cette tâche est assurée par la fonction *ueye.get_data*(**pcImageMemory, width, height, nBitsPerPixel, pitch, copy=True**). La fonction *ueye.get_data*() est utilisée dans le cadre du développement de logiciels pour les caméras industrielles IDS. Cette fonction récupère les données de l'image à partir de la mémoire tampon de l'image de la caméra et les copie dans la mémoire tampon de l'ordinateur dans une variable.

Les paramètres de la fonction sont les suivants :
• pcImageMemory : l'adresse de la mémoire tampon de l'image de la caméra
• width : la largeur de l'image en pixels
• height : la hauteur de l'image en pixels
• nBitsPerPixel : le nombre de bits par pixel
• pitch : la largeur de la ligne en octets
• copy : un booléen indiquant si les données de l'image doivent être copiées dans la mémoire tampon de l'ordinateur ou simplement pointées vers la mémoire tampon de l'image de la caméra.

2.2 Redimensionnage et stockage de la photo dans un tableau numpy (en RGB)

La deuxième fonction, **np.reshape()**, permet de redimensionner la variable précédente en une matrice numpy à trois dimensions (RGB). Cette matrice numpy est stockée dans une variable. Les dimensions de la matrice sont déterminées par les paramètres **height.value, width.value, et** $bytes\_per\_pixel$, où height.value et width.value contiennent les valeurs de hauteur et de largeur de l'image, respectivement, et $bytes\_per\_pixel$ est le nombre d'octets par pixel pour l'image.

2.3 Convertir l'image en HSV

L'espace de couleurs HSV est souvent utilisé en traitement d'images pour séparer les informations de teinte, de saturation et de valeur des pixels de l'image. Cela permet de traiter séparément les informations de couleur, de luminosité et de saturation pour des opérations telle que la détection de couleurs. Pour réaliser la conversion, il faut utiliser la fonction **cv2.cvtColor()** de la bibliothèque OpenCV. La fonction prend deux arguments : l'image d'entrée à convertir et le code de conversion à utiliser pour spécifier la conversion d'espace de couleurs. Dans notre cas, le deuxième argument est $cv2.COLOR\_BGR2HSV$ pour convertir l'image de RGB à HSV. Le résultat de la conversion est stocké dans une nouvelle variable.

2.4 Appliqué des masque binaires pour les couleurs traîtées

La fonction **cv2.inRange()** de la bibliothèque OpenCV est utilisée pour créer un masque binaire à partir d'une image. Le masque binaire est une image qui ne contient que deux valeurs de pixel : 0 (noir) et 255 (blanc). La fonction prend trois arguments : l'image d'entrée, un tuples de 3 valeurs HSV minimales et un tuple de 3 valeurs HSV maximales pour créer le masque. Tous les pixels compris dans cette plage de valeurs deviennent blancs et les autres deviennent noirs. Nous avons créé 3 masques : un vert, un jaune et un rouge.
Nous appliquons ensuite ces 3 masques à l'image obtenue à la partie précédente et nous nous retrouvons donc plus avec une image mais avec trois images binaires associées aux couleurs jaune, rouge ou verte.

2.5 Trouver la couleur prédominante

Pour trouver la couleur prédominante, nous avons choisi de parcourir les pixels de 3 images et de compter les pixels blancs présents dans chaque image. Par exemple, si la photo initiale contient un cube rouge, seul l'image résultante du masque binaire rouge aura un nombre significatif de pixels blancs. Nous comparons le nombre de pixels blancs des trois images résultantes des trois masques et si l'image qui contient le plus de pixels blancs dépasse une seuil de pixels blancs que nous avons défini expérimentalement (pour pas qu'il n'y ait de couleur détecté dans des images sans cube), alors la variable R, G ou Y sera envoyée à la carte Nucléo.

Pour permettre à notre code d'être plus rapide, nous nous sommes contenté.es de parcourir les images de 10 pixels en 10 pixels.

2.6 Transmission de l'information à la carte Nucléo

Pour cela, nous avions au préalable choisi dans le code le bon port USB. Pour envoyé l'information de la couleur sous forme de caractère (R, G ou Y), le code utilise une communication série pour envoyer des données à un périphérique. La lettre correspondante à la couleur du cube est envoyée via le port série à l'aide de la méthode **serNuc.write()**. Lorsque toutes les données ont été envoyées, la connexion série est fermée en appelant la méthode **serNuc.close()**.

3 Roulement du tapis

Dans un premier temps, nous avons pris en main notre système et notre première priorité a été de faire avancer le tapis roulant fonctionnant grâce à un moteur pas à pas.
Pour commencer, nous avons pris un autre moteur pas à pas et une carte de puissance et nous avons réalisé le code permettant de le mettre en marche sur **mbed**.

Notre code fonctionne sur un principe très simple. Nous créons tout d'abord 4 configurations possibles de notre moteur pas à pas :
- **Configuration 1** : bobines 1 et 3 alimentées et bobines 2 et 4 éteintes
- **Configuration 2** : bobines 2 et 3 alimentées et bobines 1 et 4 éteintes
- **Configuration 3** : bobines 2 et 4 alimentées et bobines 1 et 3 éteintes
- **Configuration 4** : bobines 1 et 4 alimentées et bobines 2 et 3 éteintes

On vient ensuite se placer alternativement dans ces configurations en incrémentant un compteur dans une boule while infinie (on prend le compteur modulo 3 ainsi le compteur prend successivement comme valeur 0,1,2,3,0,1,..). Si le compteur vaut 0, on se met dans la configuration 1, s'il vaut 1 dans la configuration 2, etc... .
Les bobines s'alimentant à tour de rôle permettent de faire tourner le moteur pas à pas et par conséquent de faire avancer le tapis.

Possédant un code fonctionnel, nous nous sommes alors attelé aux branchements électroniques du moteur pas à pas de notre tapis roulant.
A l'aide des documentations techniques et des informations trouvées sur internet nous avons pu finaliser le branchement du moteur pas à pas ainsi que du pont en H permettant son fonctionnement. Nous avons alors téléversé notre code précédemment réalisé et après quelques modifications (notamment pour que le moteur pas à pas tourne à une vitesse assez élevée en gérant le temps d'attente entre chaque incrémentation du compteur) le tapis a pu se mettre en marche.

Nous avons ensuite maximiser la vitesse du tapis afin de pouvoir trier un maximum de pièces par minute et ainsi respecter le cahier des charges.

4 Tri des cubes - Boutons poussoirs

Le tapis étant fonctionnel, nous avons alors cherché à pousser les pièces hors du tapis à l'aide de pistons fonctionnant grâce à des servomoteurs.
Cependant, pour pouvoir pousser les pièces il fallait tout d'abord avoir une information sur leurs positions afin d'actionner le piston au bon moment. Nous avons donc utilisé une petite caméra de détection permettant d'envoyer un signal lorsqu'une pièce se trouvait devant cette dernière.

FIGURE 6 – Caméra de détection de présence

Ainsi, chaque front montant du signal de la caméra de détection nous indiquait qu'une pièce se trouvait devant la caméra. Nous avons ensuite créé une fonction d'interruption se déclenchant à chaque front montant.
Grâce à la fonction **"servo_mot.pulsewidth_us()"**, nous activions le servomoteur et par conséquent le piston.

Cependant en utilisant la fonction d'interruption, nous quittions la boucle infinie permettant de faire avancer le tapis roulant. Nous avons donc du, dans la fonction d'interruption, demander premièrement au tapis d'avancer pendant un certain laps de temps. Nous avons ensuite ajusté ce laps de temps pour faire en sorte que la pièce soit au bon endroit lors de la mise en marche du piston.
Nous mettons ensuite en route le piston puis nous le remettons à sa position initiale.
Pour ce qui est de la couleur, la caméra (ayant détectée la couleur de la pièce) envoie à la carte Nucléo une lettre correspondant à la couleur identifiée. On vient ensuite, à l'aide d'une fonction "if", enclencher soit le piston 1 soit le 2 soit aucun.

Nous avons pu vérifier le bon fonctionnement de cette fonction en posant une

pièce de couleur et en vérifiant que celle-ci était bien poussée par le bon piston.

L'algorigramme simplifié de notre système est donc :

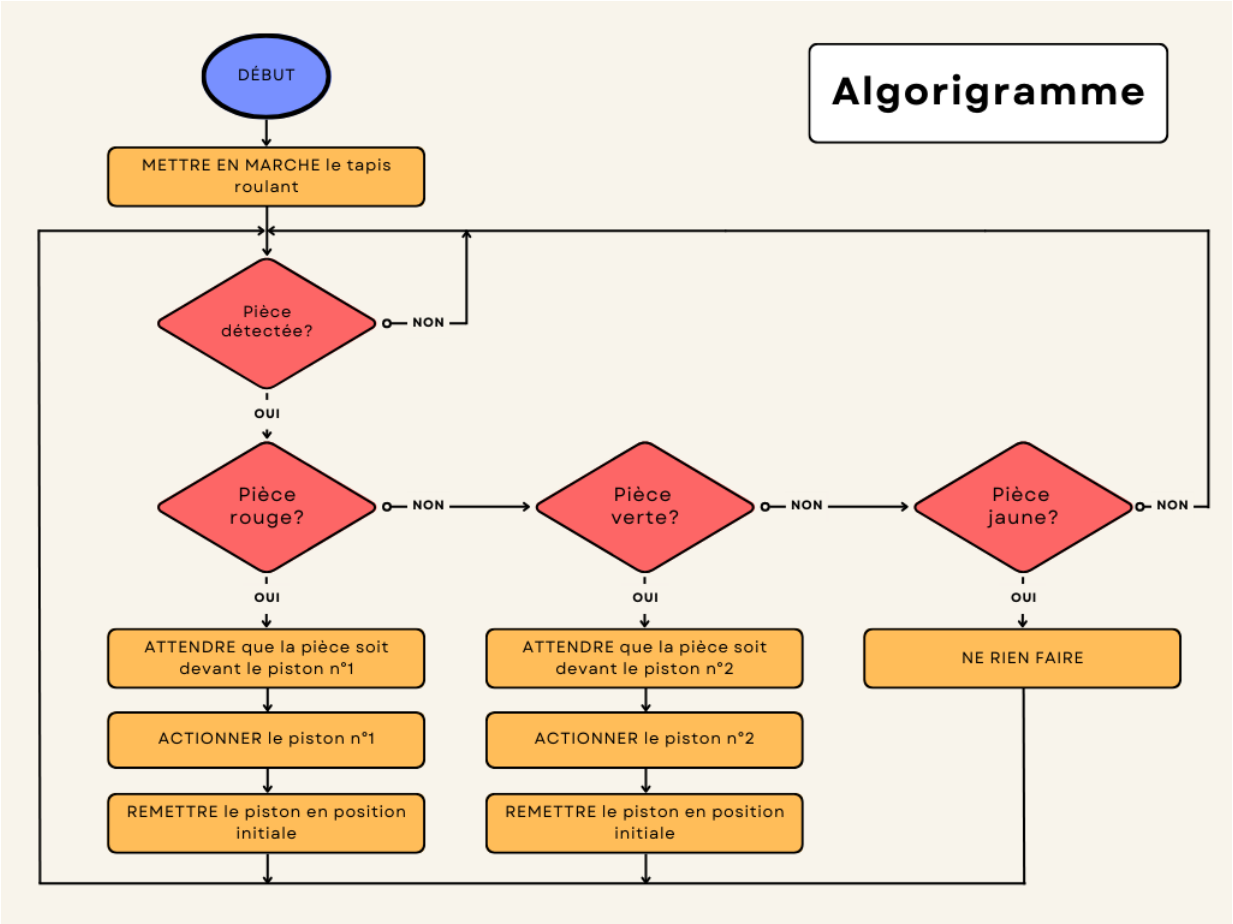

FIGURE 7 – Algorigramme

5 Bilan

5.1 Bilan technique - Avancement final

Par rapport au cahier des charges, nous avons pu cocher la moitié des cases.

1) Concernant les couleurs, nous avons bien réussi à trier les 3 couleurs prévues : **Rouge, Vert et Jaune**.
2) La rapidité de notre système est de **9 pièces par minute** avec une vitesse du tapis de **3,9 cm/s**.
3) Nous n'avons pas pu vérifier le critère d'une pièce sur 1000.
4) Nous n'avions pas eu le temps de réaliser l'interface Humain-Machine, cela sera expliqué un peu plus bas dans le rapport.

Nous pouvons proposer quelques voies d'amélioration pour celles et ceux qui reprendraient le projet. Du point de vue esthétique et de la praticité, il faudrait songer à souder le montage électrique car un mauvais contact des fils a par exemple entraîné une perte d'une heure sur une séance. Le branchement prenait assez de temps en début de chaque séance donc un soudage serait vraiment idéal.

Nous pourrions également trier une couleur de plus, le **bleu** par exemple et nous pouvons encore aller plus loin en triant des entités par leur forme. Une autre équipe a travaillé sur ce projet, une mise en commun de nos travaux et des leurs est donc possible pour aboutir à un système encore plus efficace.

Enfin, nous pourrions faire fonctionner un troisième piston et aussi améliorer la vitesse du tapis pour trier encore plus de pièces, encore plus rapidement.

5.2 Quelques difficultés techniques surmontées...

Lors de ce projet, nous avons du faire face à plusieurs problèmes :
• Le premier était le bruit. En effet, nous avons du mélanger circuit de puissance (alimentation) et circuit de signaux. Ceci à entraîné beaucoup de bruits qui étaient vraiment dérangeants puisque la carte Nucléo pensait détecter des fronts montants alors qu'il n'y avait rien.
Pour résoudre cela, nous avons branché des condensateurs entre la masse (commune à nos deux circuits puissance et signaux) et les 5V d'alimentation du tapis et des pistons. Ces condensateurs fonctionnant comme des passes-bas ont permis de réduire drastiquement le bruit et de résoudre notre problème de fausse détection.

• Nous avons aussi rencontré des problèmes avec l'utilisation de l'interface Spyder et avec l'aspect de traitement de l'image du projet. En effet, nous sommes resté.e.s bloqué.e.s quelques séances car nous avions du mal à trouver des documentations sur notré caméra et sur l'interface Open CV. De plus, il nous a fallu attendre d'assister à un cours du Traitement de l'image pour penser à passer en HSV, ce qui nous a ensuite facilité la tâche.

• Lors de notre présentation, l'ordinateur ne reconnaissait plus le port USB sur lequel était branché la carte Nucléo ce qui nous a empêché de montrer l'entierté de notre travail.

5.3 Retour d'expérience sur l'équipe

Cette collaboration avec l'entreprise SOLEC a été l'occasion pour nous d'être ambitieux.se.s. Nous avons voulu au départ utiliser des méthodes de **Machin Learning** pour innover et tester de nouvelles techniques. Les deux premières séances ont été utiles sur ce point car nous nous sommes renseigné.e.s sur la méthode et elle s'est avérée plus complexe que prévu. Nous avons également compris la limite de temps qui nous était imposé. Il fallait aller au plus efficace même si cela reste plus simple.

Si vous qui lisez le présent rapport, veniez à retravailler sur ce projet, nous vous donnons quelques conseils sur la gestion du temps. Nous nous sommes divisés en deux équipes : une équipe caméra et une équipe convoyeur pour avancer dans le projet, cela a été une excellente idée. Néanmoins, nous avons sous-évalué le temps que nous allions passer à synchroniser les deux domaines. En regardant le diagramme de Gantt ci-dessous, nous pouvons constater avoir prévu 5 séances pour coder la caméra afin qu'elle puisse trier les couleurs. Finalement, cela a pris 2 séances de plus et tout le monde a fini par se mobiliser sur la caméra. Cela a raccourci le délai pour synchroniser le tout et nous n'avions malheureusement pas eu assez de temps de créer l'interface Humain-Machine.

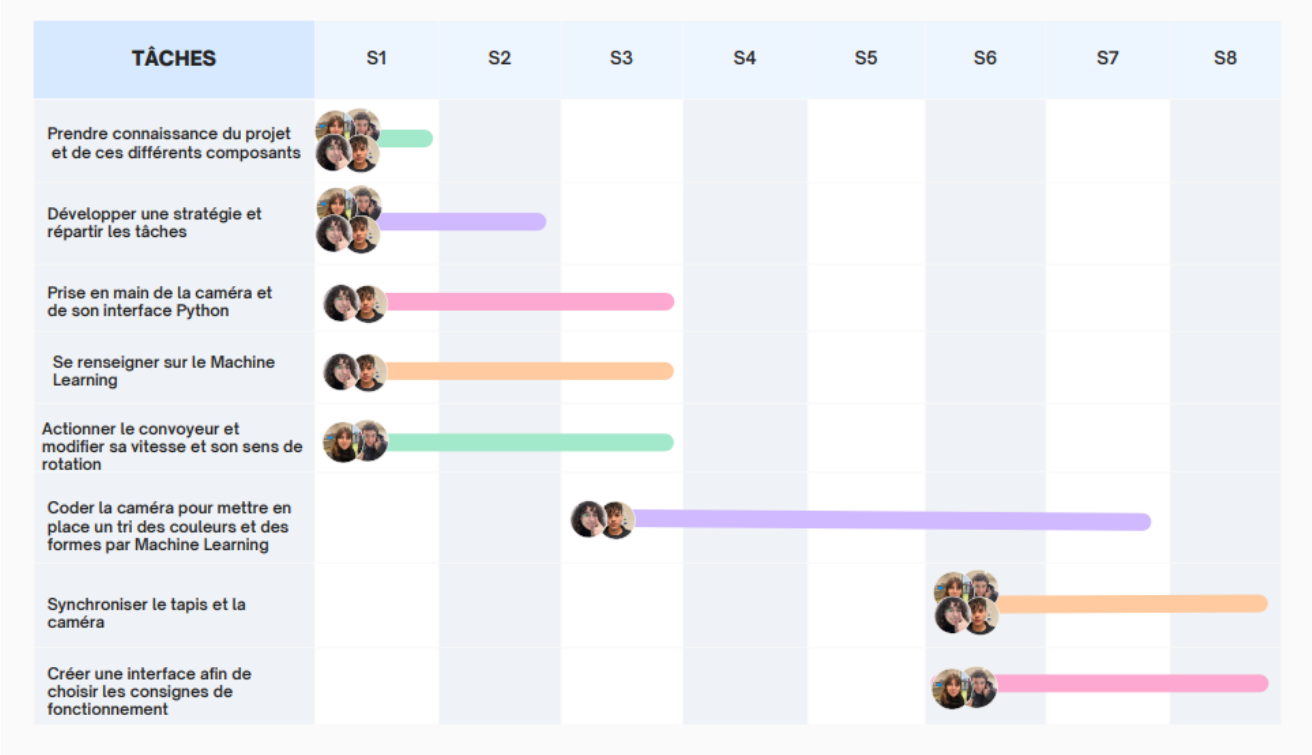

FIGURE 8 – Diagramme de Gantt

Nous retenons tout le positif à l'issue de ce projet. Il nous a permis de renforcer des liens dans une équipe déjà soudée. Les binômes et la répartition des tâches ont bien fonctionné. Il ne va sans dire que chacun s'est investi dans tous les domaines et que les efforts ont été communs, nous pouvons quand même souligner que chacun.e a pu s'illustrer dans son domaine de prédilection et a pu donner des conseils aux autres : Marie et Arthur dans la technicité et la pratique ainsi qu'une ingéniosité à toute épreuve, Orane dans les codes et Sébastien dans tout ce qui a été transverse. Nous étions très complémentaires et c'est une des qualités principales pour la bonne gestion d'un projet.

Finalement, nous étions bien organisés sur **Teams** en mettant à jour, à l'issue de toutes les séances, nos avancées et nos références. Cette centralisation a été très effi-

cace notamment pour la préparation des audits qui se sont excellemment passés. Du point de vue technique, nous avons toutes et tous une bonne fois pour toute compris l'intérêt d'un condensateur dans un circuit, l'intérêt d'une boucle infinie while vide ou même les rouages du codage en Python...

6 Conclusion

Nous voulons conclure ce projet sur le tri industriel par "computer vision" en collaboration avec l'entreprise **SOLEC** en soulignant que c'est une vraie réussite pour nous. Nous en ressortons encore plus confiant.e.s dans nos capacités en électronique et en systèmes embarqués. Cette confiance est nécessaire pour aborder notre future carrière d'ingénieur. Nous remercions évidemment l'**IOGS**, les expert.e.s techniques de chez **EmbSys** et de **InVino Veritas**, ainsi que les conseillers.ères en management de chez **Activ'Management** de nous avoir toujours épaulé.e.s et de nous avoir permis d'aboutir à ce projet et au présent rapport.