João SANTOS MingYu LIAO Leonardo BRAGA Thémis LEPORTZ Matthieu DUBERNARD

Rapport Technique de PROTIS

—

Vision industrielle

Gr3

Nous attestons que ce travail est original, que nous citons en référence toutes les sources utilisées et qu'il ne comporte pas de plagiat.

Table des matières

1 Introduction

1.1 Objectifs

L'objectif de notre groupe de travail était de fournir à la société SOLEC un outil d'automatisation de leur chaîne de production en y ajoutant des options de tri de pièces ou d'objets sur la base de leur forme. Notre système repose sur l'utilisation d'un convoyeur entraîné par un moteur pas à pas permettant de faire défiler les formes à trier, une caméra monochromatique chargée de détecter la présence et la forme des pièces, des bras pousseurs entraînés par un moteur à courant continu et une lampe pour contrôler l'éclairement des pièces. Le tout étant piloté par une carte Nucleo connectée à un PC (voir Figure 2).

1.2 Notice d'utilisation

Le dispositif est simple à l'emploi. Il suffit de disposer des pièces de différentes formes au centre du convoyeur et d'attendre qu'elles soient acheminées jusqu'à un des bras pousseurs par ce dernier afin de repousser les pièces à différents endroits hors du tapis en fonction de leur forme.

Une première phase de configuration en apparence plus compliquée est en revanche nécessaire avant de procéder à un cycle de tri des pièces. C'est une des pistes d'amélioration que nous évoquerons plus tard. En voici les détails :

1. Allumer la caméra, l'alimentation du tapis roulant, le PC. Depuis ce dernier, compiler le code (dans Keil Studio). Un retour de la caméra s'affiche avec un rectangle au centre représentant la zone de détection.

2. Allumer la lampe et trouver un angle d'incidence qui permet de distinguer clairement les pièces, sans artefact dans le rectangle du retour.

3. Mesurer la distance entre la caméra et chaque bras et la renseigner (en cm) dans la fonction principale du code (voir Figure 1).

```
int main() {
    my_pc.baud(115200);

    speed = 3.1835;  // motor velocity in cm/s
    size_object = 1.4; // object size in cm

    // All distances take the camera as reference and are in centimeters
    distance_camera_arm1 = 6.5; // in cm
    distance_camera_arm2 = 22.3; // in cm
    distance_camera_arm3 = 29.5; // in cm
```

FIGURE 1 – Lignes du code à initialiser en renseignant la distance entre la caméra et chaque bras pousseur

4. Recompiler le programme. Le dispositif est prêt à être utilisé en positionnant les formes à trier au début de la course du tapis.

1.3 Cahier des charges

Le cahier des charges qui nous a été adressé stipule que notre solution technique doit pouvoir trier 4 formes différentes avec une cadence de 10 pièces par minute, le tout avec un maximum d'une erreur toutes les 1000 pièces. Elle doit également posséder une interface Humain-Machine pour choisir les pièces à trier.

1.4 Schéma fonctionnel

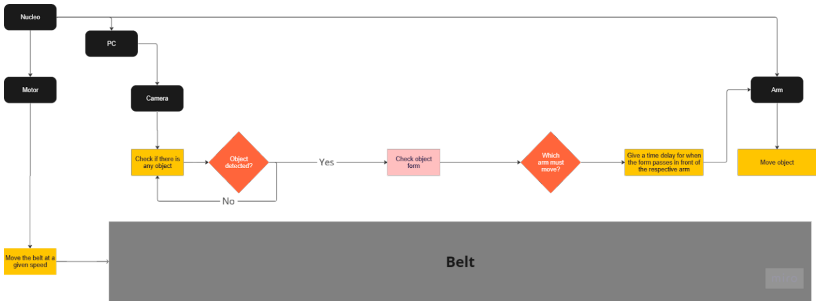


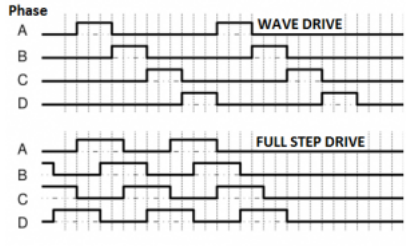
FIGURE 2 – Schéma fonctionnel final de notre dispositif

Comme on peut le voir sur la figure 2, notre dispositif peut être décomposé en trois fonctions principales dont nous détaillerons le fonctionnement par la suite. Tout d'abord, le **pilotage du tapis roulant** par un moteur pas à pas et une carte Nucleo qui permet le défilement des pièces devant la caméra et les bras pousseurs. Ensuite la **détection de formes**, réalisée par un code python, utilisant une caméra reliée à un PC. Enfin, l'**avancement des bras pousseurs** commandé par la carte Nucleo et un moteur à courant continu.

2 Description des fonctions du dispositif

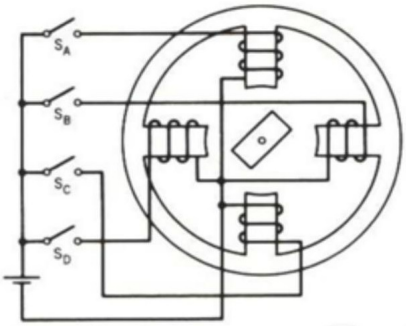
2.1 Pilotage du tapis roulant

Le tapis roulant est entraîné par un moteur pas à pas. Pour contrôler le moteur pas à pas, 2 cartes sont également utilisées, la carte puissance L298, et la carte contrôle L297. Le principe général de la commande du moteur pas à pas est le suivant : un signal d'horloge carré généré par la Nucleo est envoyé au L297. La carte L297 génère 4 sorties déphasées basées sur la fréquence du signal d'horloge, et envoie les 4 sorties à la carte L298. La carte L298 est alimentée en 5V, et émet 4 signaux spéciaux pour piloter le moteur pas à pas.[1]



[a]

Signal d'entrée quadriphasé requis pour le moteur pas à pas. [2]



[b]

Schéma de la boucle d'induction magnétique du moteur pas à pas. [3]

FIGURE 3 – Principe de fonctionnement du moteur pas à pas

Pour faire sortir un signal d'horloge de la carte Nucleo, on utilise le code présenté en Figure 4.

```
PwmOut Clock_motor(D9);

void moveBelt(){
    Clock_motor.period_us(1250);
    Clock_motor.write(0.1);          // Start moving the BELT
}
```

FIGURE 4 – Extrait de notre code permettant de générer à partir de la carte Nucleo le signal d'horloge à destination de la carte L287. L'entrée du moteur à 4 phases sera adaptée à partir de ce signal

La carte L297 convertit le signal d'horloge d'entrée en signaux à 4 phases, comme le montre la figure 3. Ces 4 signaux sont de même forme mais ont une différence de phase, et sont utilisés pour déplacer le moteur pas à pas. Le L297 envoie tous les signaux au L298 qui est basé sur un pont en H. L'alimentation est aussi connectée à cette carte pour piloter le moteur. En principe, l'alimentation peut aller au-delà de 46V mais nous avons utilisé seulement 5V pour le faire fonctionner afin de déplacer de petites choses.La carte L289 possède 4 sorties, connectez ces sorties au moteur pas à pas, le moteur est alors capable de déplacer le tapis roulant.

2.2 Détection des formes

Pour détecter les formes, nous avons principalement utilisé la bibliothèque Python "OpenCV". Nous avons utilisé le code du fabricant de la caméra uEye pour recevoir les images en direct et ensuite nous avons fait le traitement des données. Tout d'abord, nous avons augmenté le contraste et utilisé un flou gaussien pour supprimer les fréquences spatiales trop élevées et ainsi rendre le système plus robuste à des détails dans l'image tels que des poussières ou des irrégularités dans le revêtement des formes. Ensuite, nous avons utilisé une technique de "seuil binaire" pour filtrer les objets plus brillants qu'un certain seuil car les formes à trier étaient en bois clair, défilant sur le tapis noir. Nous avons

enfin calibré les objets apparaissant en clair à l'image comme étant les formes à trier.

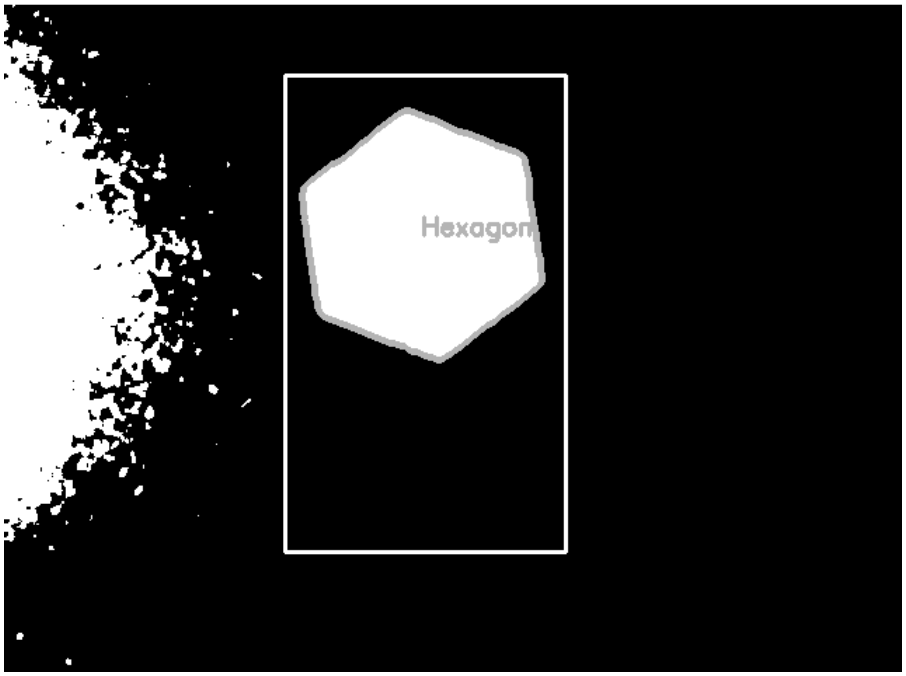


FIGURE 5 – Interface montrant la zone de détection et le nom de la forme détectée.

Avec cette information, nous pouvons utiliser les fonctions OpenCV pour détecter les contours de la forme dans l'image et faire une approximation polygonale de la forme. Nous pouvons filtrer la forme en mesurant simplement le nombre de côtés du polygone. Afin de réduire le nombre de fausses détections, nous avons établi une zone de détection délimitée par un rectangle sur lequel n'apparaissent aucun artefact tels que les réflexions parasites générées par l'éclairage par exemple. Nous avons également fixé une taille minimale pour la forme détectée (voir Figure 6). Un mauvais éclairage peut affecter grandement notre système et sera commenté plus en détail à la fin.

```python
#Find the contours
contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

for contour in contours:

    # compute the bounding box for the contour
    (x, y, w, h) = cv2.boundingRect(contour)

    # reject contours outside size range
    if w < 30 or h < 30 :
          continue
    # make sure the box is inside the frame
    if x <= RectX or y <= RectY or x+w >= RectX+RectWidth or y+h >= RectY+RectHeight:
      continue

    # cv2.approxPloyDP() function to approximate the shape
    approx = cv2.approxPolyDP(contour, 0.018 * cv2.arcLength(contour, True), True)
```

```python
    # putting shape name at center of each shape
    shapeName = ''
    if len(approx) == 3:
        shapeName = 'Triangle'

    elif len(approx) == 4:
        shapeName = 'Square'

    elif len(approx) == 5:
        shapeName = 'Pentagon'

    elif len(approx) == 6:
        shapeName = 'Hexagon'

    elif len(approx) == 12:
        shapeName = 'Etoile'
    else:
        shapeName = 'Circle'
```

FIGURE 6 – Extrait de notre code permettant de réaliser l'approximation des polygones et filtrer chaque forme.

Comme il est plus facile de traiter des caractères uniques en C++ (le langage utilisé par Nucleo), le programme Python présenté en Figure 7 n'envoie que la première lettre de la forme au Nucleo. Pour éviter les demandes redondantes, le programme n'envoie qu'une lettre toutes les secondes.

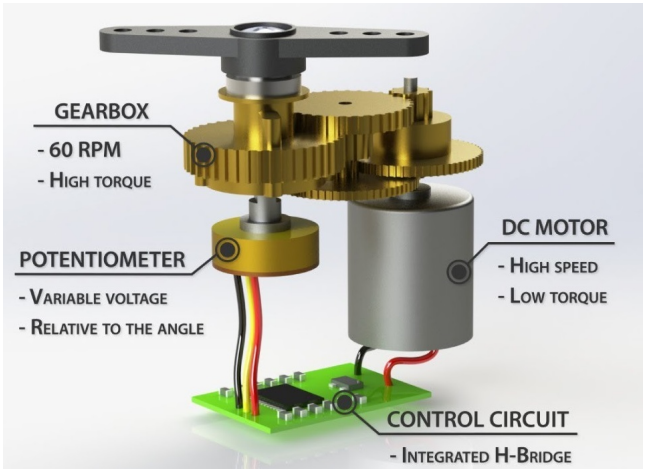
```python
if shapeName != '':
    cv2.putText(thresh, shapeName, (x, y), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (180, 0, 0), 2)
    NewWrite = time.time()

    #send to Nucleo only every 1 second
    if NewWrite - LastWrite > 1:
        LastWrite = time.time()
        print('sending data: ', shapeName)
        serNuc.write(bytes(shapeName[0], 'ascii'))
```

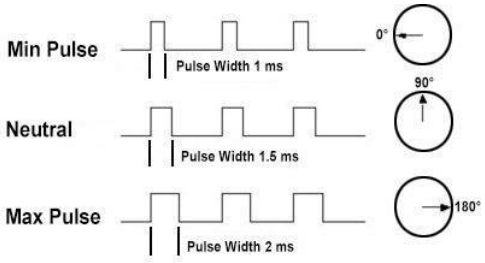
FIGURE 7 – Extrait de notre code permettant d'envoyer la première lettre de la forme détectée à la carte Nucleo au maximum toutes les secondes.

2.3 Avancement des bras pousseurs

Dans notre configuration, les bras sont entraînés par un servomoteur. Le principe d'un servomoteur est illustré dans la Figure 8



Structure d'un servomoteur. [4]



Signal de commande du servomoteur. [5]

FIGURE 8 – Principe de fonctionnement du servomoteur

Le servomoteur est directement contrôlé par la carte Nucleo, 3 entrées sont nécessaires pour un servomoteur : Vcc, Ground et le signal. Vcc et Ground sont les sources d'alimentation, le signal est destiné à contrôler l'angle du servomoteur.

Cependant, selon la fiche technique de notre servomoteur, la largeur d'impulsion doit être de 500μs pour un angle de 0° et de 2530μs pour un angle de 180°[6]. Mais nous n'utilisons pas toute la plage du servomoteur, à cause des restrictions mécaniques, le bras ne peut pas atteindre 180°. Le code permettant de contrôler le servomoteur est donc illustré en Figure 9.

```
PwmOut arm1(D10);
void move_arm1(){
    arm1.pulsewidth_us(2420);// move arm forwards
    thread_sleep_for(3*1000); // wait 3 sec
    arm1.pulsewidth_us(900); // move arm backwards
}
```

FIGURE 9 – Extrait de notre code permettant de déplacer le bras.L'impulsion PWM d'une durée de 2420μs sert à faire avancer le bras et l'impulsion PWM d'une durée de 900μs sert à le faire reculer.

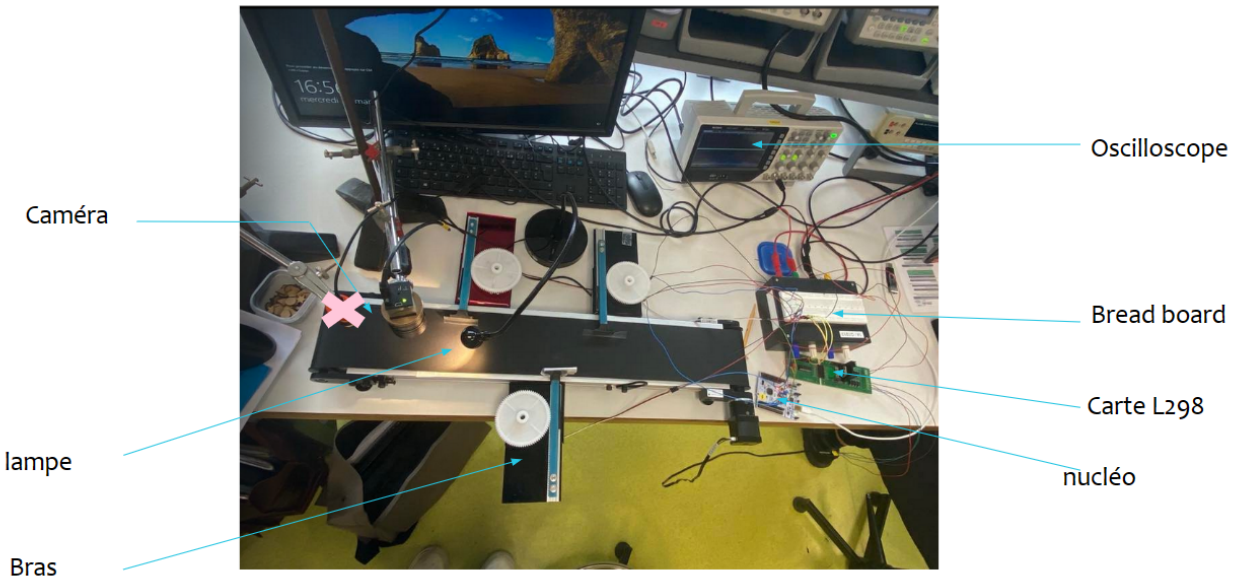


FIGURE 10 – Photographie notre dispositif

3 Bilan

3.1 Avancement final et pistes d'amélioration

Lors de ce projet, nous avons réussi à mettre en place un dispositif qui trie différentes pièces par formes. Le dispositif est bel et bien capable de trier un maximum de 4 formes différentes assignables à chacun des trois bras pousseurs et à la fin de course du convoyeur.

Néanmoins, nous n'avons pas pu être à la hauteur de toutes les attentes du cahier des charges. Nous avons réussi à trier en moyenne 10 pièces par minute et rencontrons un taux d'erreur de 1/15 contre 1/1000 attendu. Le taux d'erreur pourrait être grandement amélioré en contrôlant mieux l'éclairage des pièces dont dépend sensiblement la bonne détection des formes. Pour cela, nous pourrions concevoir et en fixer une enceinte opaque, percée d'un trou pour la caméra et d'un second pour la lampe en début de course du tapis. Ainsi l'éclairage des pièces serait parfaitement contrôlé. Pour ce qui est de la cadence, une première option serait de changer le moteur pilotant le convoyeur pour un autre permettant d'atteindre des vitesses plus élevées. Une solution plus sérieuse consisterait à revoir toute la structure du programme afin de piloter l'actionnement des bras pousseurs par fonctions d'interruption.

De plus, nous n'avons pas eu le temps de développer une interface Humain-Machine qui permettrait d'associer aisément un bras à une forme à trier. On pourrait de même imaginer pouvoir rentrer la distance séparant chaque bras de la caméra directement depuis cette interface de sorte à ne plus avoir à modifier le code à chaque fois que les bras sont déplacés ou bien supprimer les degrés de liberté sur les bras et la caméra.

3.2 Retour d'expérience

Au début du projet, nous avons planifié de le réaliser tel le schéma de la figure 10. Si on compare les schémas initial et final, on peut voir que la différence principale réside dans le fait qu'au début nous avons utilisé un sensor infrarouge qui détectait les objets avant d'allumer la caméra pour les distinguer. Grâce au sensor, nous étions capables de tester notre algorithme pour avancer un bras dans le bon moment dès qu'il avait détecté quelque chose même si nous n'avions pas encore fini le codage

de détection des formes. Pourtant, après l'intégration MBED et Python, nous n'avions plus besoin du sensor pour faire la détection des objets, ce qui a rendu le projet plus simple.

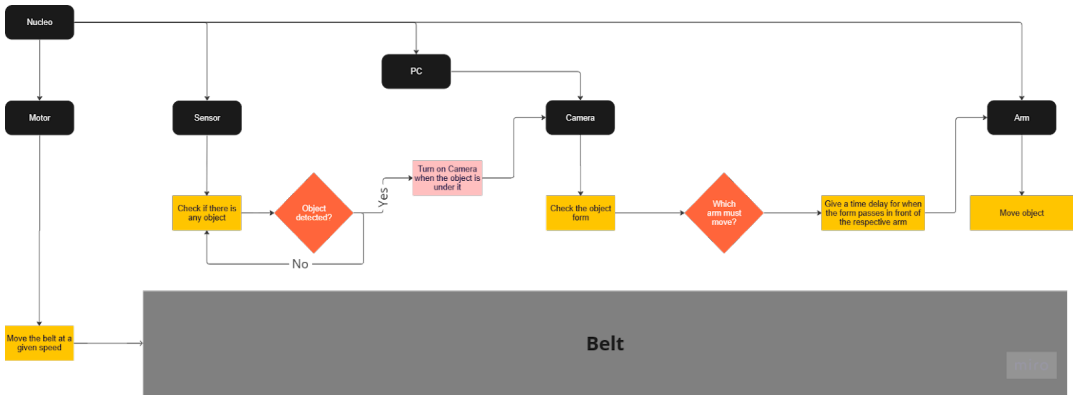


FIGURE 11 – Schéma de principe initial de notre dispositif

D'après le schéma initial (Figure 11), nous avons pu développer le diagramme de Gantt présenté en figure 12 pour partager les tâches et faire un chronogramme du projet.

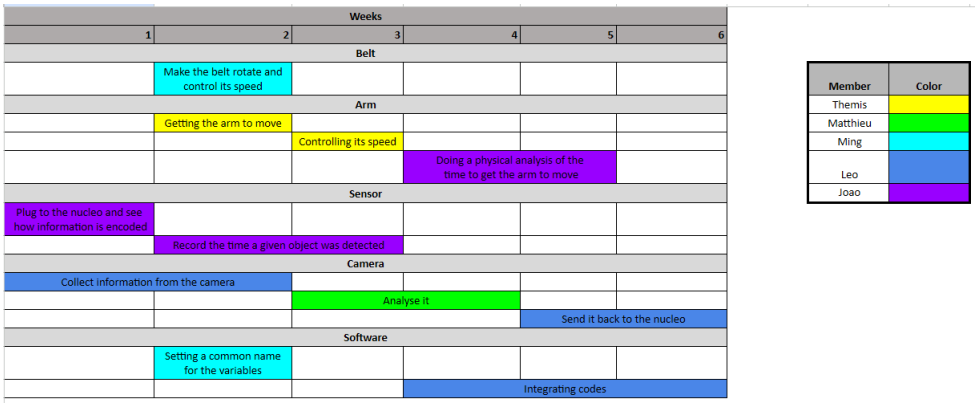


FIGURE 12 – Diagramme de Gantt du groupe

Suite à nos diverses séances, nous nous sommes rendu compte qu'il fallait adapter notre chronogramme à cause de quelques tâches qui ont pris plus de temps que prévu. L'évolution du groupe par séance peut être vue par l'emploi du temps présenté dans la figure 13.

Weeks					
1	2	3	4	5	6
Belt					
	Make the belt rotate and control its speed				
Arm					
	Getting the arm to move				
			Controlling its speed		
			Doing a physical analysis of the time to get the arm to move		
Sensor					
Plug to the nucleo and see how information is encoded					
		Record the time a given object was detected			
Camera					
Collect information from the camera					
		Analyse it			
				Send it back to the nucleo	
Software					
			Setting a common name for the variables		
			Integrating codes		
					Verify the specifications

Status	Color
Done	
Doing	
To Do	

FIGURE 13 – Emploi du temps du groupe

Cette expérience a été très enrichissante pour nous, à la fois d'un point de vue de gain en connaissances scientifiques mais également en travail de groupe. En effet, nous avons pu apprendre plus sur la programmation, sur MBED ou Python, le traitement d'image et de signal. Nous avons également acquis une grande rigueur sur l'établissement de montages électroniques. En effet, nous avons compris l'importance de vérifier sur oscilloscope chaque branche de notre montage électronique pour s'assurer de bien envoyer le bon signal.

D'un point de vue travail de groupe, nous avons très vite intégré l'importance de l'organisation préalable du groupe et de la communication. Nous pouvions ainsi s'entraider lorsque l'un avait fini sa tâche car il était accessible et disponible à tous la progression de chacun. Nous avons également découvert l'importance de communiquer avec les groupes qui travaillent sur les mêmes choses que nous. En effet, nous pouvions nous entraider sur les problèmes similaires rencontrés.

Références

[1] "Lense instruction pour pilotage d'un convoyeur." `http://lense.institutoptique.fr/projet-pilotage-dun-convoyeur/`. Acessed on April 5th, 2023.

[2] "Moteur pas à pas." `https://www.powerelectronictips.com/unipolar-vs-bipolar-drive-for-stepper-motors-part-1-principles-faq/`. Acessed on April 5th, 2023.

[3] "Moteur pas à pas." `https://steppersmith1986.wordpress.com/`. Acessed on April 5th, 2023.

[4] "Structure d'un servomoteur." `https://howtomechatronics.com/how-it-works/how-servo-motors-work-how-to-control-servos-using-arduino/?utm_content=cmp-true`. Acessed on April 5th, 2023.

[5] "Signal de commande du servomoteur." `https://www.researchgate.net/figure/Electric-signal-control-servo-motor_fig1_344422714`. Acessed on April 5th, 2023.

[6] "Datasheet du servomoteur." `https://www.parallax.com/product/parallax-standard-servo/#:~:text=The%20Parallax%20Standard%20Servo%20was%20designed%20for%20Parallax%E2%`

80%99s,and%20is%20easily%20interfaced%20with%20any%20Parallax%20microcontroller.
Acessed on April 5th, 2023.