PAILLARD Mathis
REMAZEILLES Paul
GOYHETCHE Théo
ÉLOSÉGUY Jean

IÉTI

Compte rendu de notre Projet

Introduction

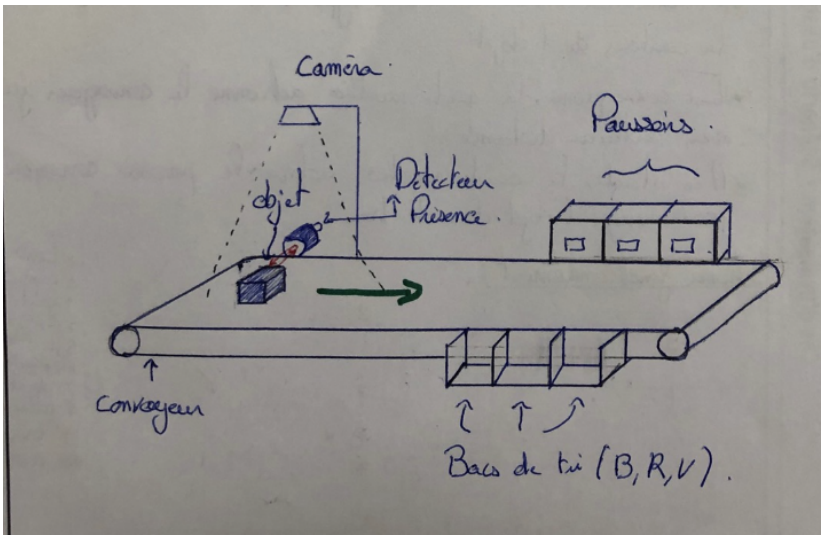

Figure 1: Schéma de principe du convoyeur

Le projet Vision Industrielle vise à mettre en place un système de tri pour des pièces industrielles de différentes couleurs. Ce projet s'inspire du convoyeur Dobot Magician, qui est couramment utilisé pour le tri de pièces ou la détection de défauts dans les chaînes de production. Ce projet comprend quatre éléments représentés dans la première figure, de gauche à droite : des pièces colorées, un détecteur de présence infrarouge, un capteur de couleurs et un convoyeur. Toutefois, pour ce projet, nous n'utiliserons pas le capteur de couleur, mais une webcam ainsi qu'un logiciel de traitement d'images. Nous avons également tous les équipements électriques et informatiques nécessaires, notamment une carte Nucleo L476RG,un GBF, une carte LP298 et LP297, des générateurs de tension et des fils électriques.

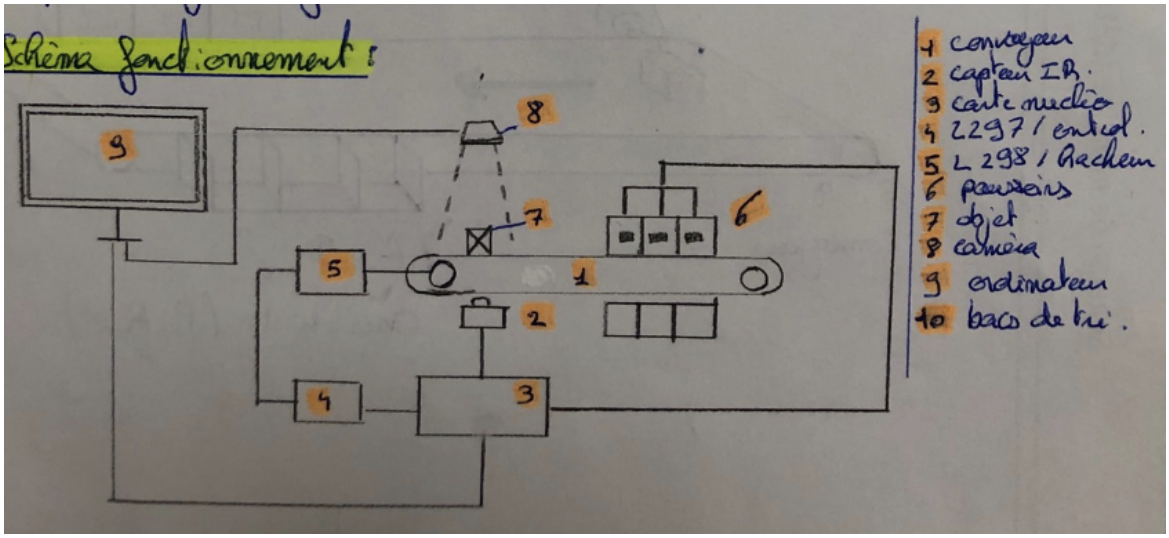

Figure 2: Schéma de principe de fonctionnement

Le fonctionnement du système est décrit ci-dessus. La carte nucléo permet de piloter le moteur, ainsi que les servomoteurs des boutons poussoirs, mais aussi de faire le pont pour envoyer les différentes informations délivrées par le traitement d'images en informations au moteur.Lorsque la pièce à trier passe devant le capteur de présence sur le convoyeur, le convoyeur s'arrête. Ensuite, la webcam prend une photo de la pièce et un programme de traitement d'images analyse la couleur de la pièce. La carte Nucleo reçoit ensuite cette information et commande le servomoteur pour qu'il tourne d'un certain angle en fonction de la couleur de la pièce. En somme, les pièces sont triées en fonction de leur couleur grâce à ce système. Pour ce faire, nous nous sommes séparés le travail en 4 parties :

- Détection des couleurs : Jean et Théo
- Mise en route du convoyeur et du capteur : Mathis
- Mise en mouvement du bras pousseur
- Établissement Liaison Python-Nucléo: Henri

Une fois toutes ces parties faîtes, nous les avons assemblées afin de réaliser le code général dans le "main".

1 Prototypage :

1.1 Détection des couleurs :

Nous allons travailler sur des cubes de couleur différentes : rouge, bleu, jaune et vert. Afin de pouvoir envoyer l'information de la couleur permettant de sélectionner le bon poussoir à activer, nous devons d'abord prendre une photo de la pièce à étudier au bon moment. Pour ce faire, nous avons utilisé une caméra de la marque IDS.

Le principe de la prise de photo repose en une liaison entre le détecteur de présence et le programme Python. Une fois la pièce devant le capteur de présence, ce-dernier envoie l'information à la caméra de prendre une photo de la pièce (si l'état du capteur est en $b'5'$ alors cela signifie que le cube n'est pas devant le capteur et donc le tapis continue d'avancer et il n'y a pas de prise de photo, en revanche si l'état du capteur est en $b'1'$ alors le cube est devant le capteur, le tapis s'arrête et la caméra prend une photo). En langage Python, cela se traduit par 2 boucles if : sur l'état $b'5'$ alors le programme renvoie simplement *riendétecté* et sur l'état $b'1'$ le programme exécute l'ensemble des lignes permettant la prise de la photo et du traitement de l'image. Une fois la photo du cube prise, il faut maintenant réaliser un programme permettant de récupérer l'information contenant la couleur du cube pour la transmettre aux poussoirs.

Les informations relatives à la couleur de l'image sont contenues dans un tableau de dimension 3×255 (nombre de pixels), avec sur chaque ligne le niveau de rouge, de bleu et de vert. Le principe de notre code repose sur l'utilisation de filtres permettant de déterminer la couleur du cube. Nous définissons alors une borne basse et une borne haute pour chaque couleur : rouge, bleu, vert et jaune. Si à l'issue du filtre les pixels de l'image sont contenus dans l'intervalle défini par une des bornes, alors le code python renvoie l'information de la couleur au poussoir; si la couleur n'est pas définie, le tapis continue d'avancer sans renvoyer d'informations aux poussoirs ("1" correspond au rouge, "2" au jaune, "3" au vert et "4" au bleu).

1.2 Mise en mouvement du bras pousseur

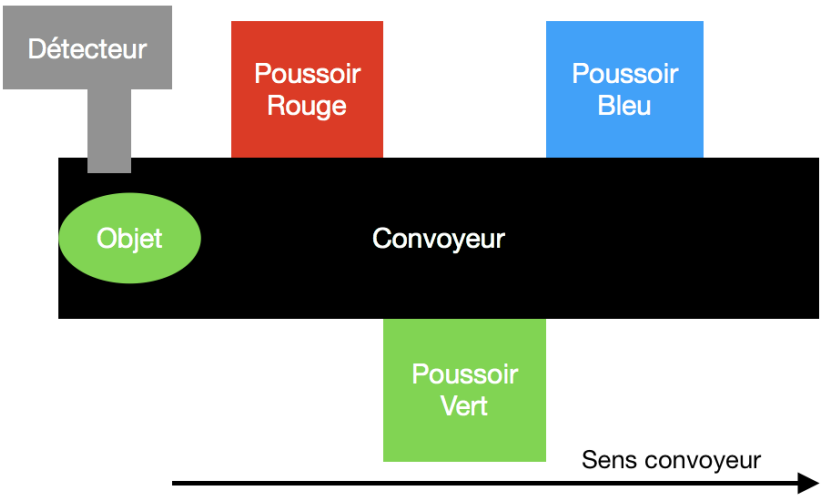

Figure 3: Schéma de principe de fonctionnement

Le bras pousseur est mis en mouvement par un servomoteur. Le servomoteur est alimenté en 5V et est contrôlé en sortie Pwm par la Nucléo. Dans notre cas, nous utilisons trois bras pousseurs pour les 3 couleurs : Rouge, Vert et Bleu. En fonction de l'information envoyée par l'ordinateur sur la couleur de l'objet, nous faisons intervenir le bon bras pousseur. Pour respecter le bon timing, nous avons chronométré les délais entre le capteur et les trois bras pousseurs, à vitesse constante. Nous les avons ensuite pris en compte pour les activer au moment adéquat. Pour chaque bras activé, nous poussons l'angle au maximum avant qu'il revienne à sa position initiale.

1.3 Liaison Serial

Pour que l'ordinateur puisse communiquer des informations à la nucléo et inversement, on met en place une liaison serial. On commence par brancher la nucléo à l'ordinateur en utilisant un câble RS232. En fonction de si on est du côté de l'ordinateur ou de la nucleo, on ne code pas l'envoie et la réception de données de la même manière.

Du côté de Phyton

On commence par déterminer sur quel port est branché la nucléo. Pour cela, on lance le code "serial_detect" (cf. Annexes) qui nous renvoie le nom du bon port. On peut ensuite travailler dans le "Main final" et après avoir importé le module "serial", on peut utiliser les fonctions suivante.

Pour initialiser la connexion sur le port 'COM12' à une fréquence de 9600, on écrit :

```
self.ser = serial.Serial(port='COM12', baudrate = 9600, timeout = 1)
```

Pour vérifier si on a reçu une donnée, on vérifie la condition suivante :

```
self.ser.inWaiting()>0
```

Pour lire une donnée avec serial, on écrit :

```
self.ser.readline()
```

Pour envoyer une donnée avec serial, on écrit :

```
self.ser.write(bytes(str(couleur),"ascii"))
```

Du côté de Nucleo

Voici les différentes fonctions qui nous permettent de faire la liaison serial.

Pour définir la variable issue du cable auquel elle est branché la nucléo (TX : Broche de transmission - sortie, RX : Broche de réception – entrée) :

```
UnbufferedSerial my_pc(USBTX, USBRX);
```

Pour définir la fréquence à laquelle on reçoit des données :

```
my_pc.baud(9600);
```

Pour vérifier si on a reçu une donnée, on vérifie la condition suivante :

```
my_pc.readable() > 0
```

Pour recevoir une donnée de l'ordinateur :

```
my_pc.read(&data, 1);
```

Pour envoyer une donnée à l'ordinateur :

```
data_to_send = 'data';
my_pc.write(&data_to_send, len(data_to_send));
```

Ces différentes fonctions nous permettent alors de réaliser le cycle de communication suivant :

- La nucleo envoie "0" à l'ordinateur pour initialiser la liaison.
- Le capteur détecte un objet, la nucléo envoie le caractère "1" à l'ordinateur.
- L'ordinateur renvoie le caractère '1', '2', ou '3' en fonction de la couleur.
- À la fin de l'acheminement de l'objet, la nucléo envoie '1' ou '0' en fonction de s'il y a un objet devant le capteur, le cycle recommence.

2 Résultat final

Ayant réussi à faire fonctionner indépendamment chacun des différents modules, on a pu rassembler tous les codes dans un Main et les montages sur une seule nucleo. Après plusieurs tentatives, tout a fonctionné et le convoyeur peut maintenant trier des pièces de couleurs différentes automatiquement.

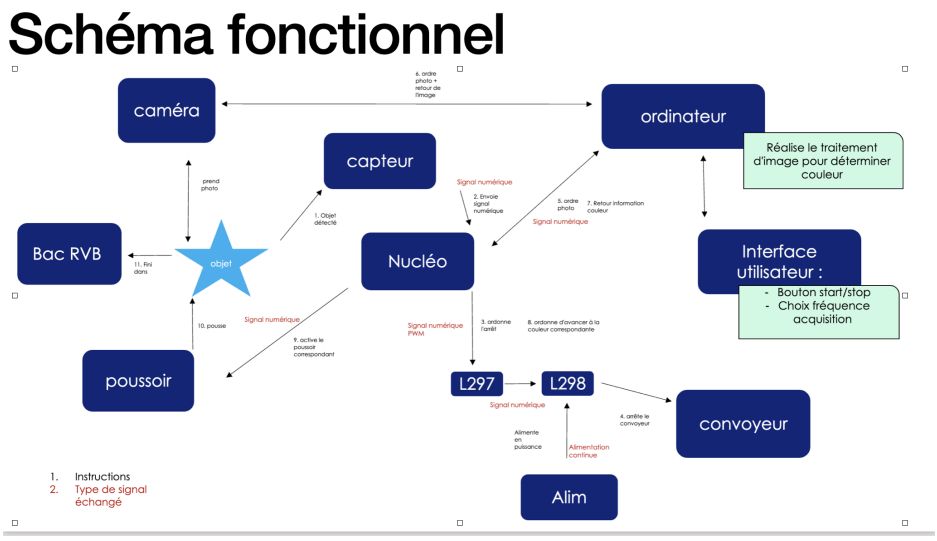

Figure 4: Montage expérimental

3 Planning

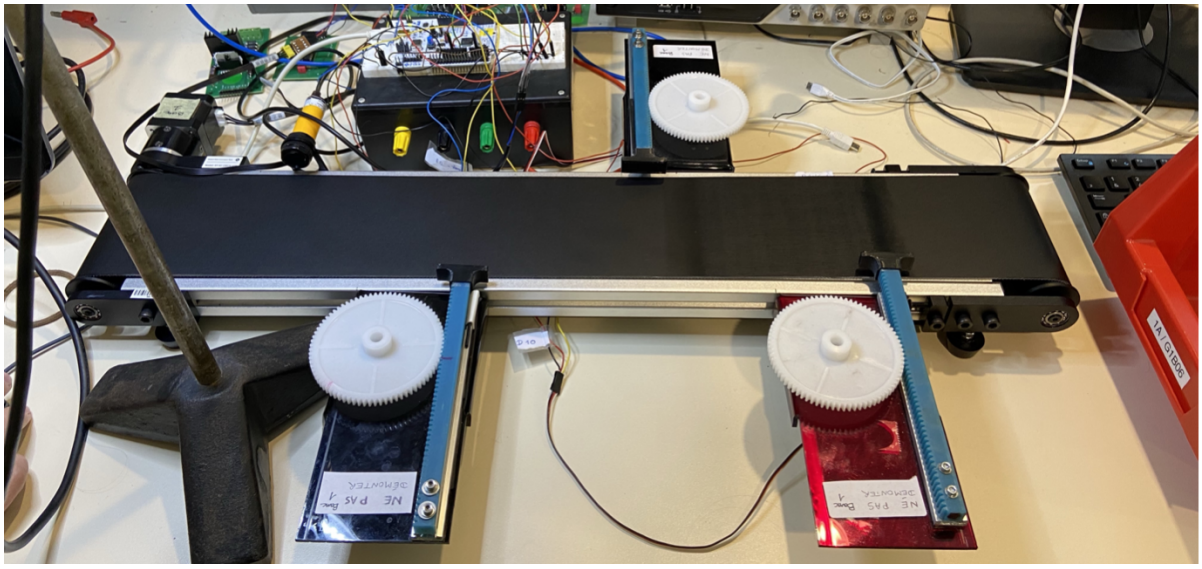

Figure 5: Montage final du groupe

4 Planning

Séance 1	Découverte du projets et mise en place du cahier des charges
Séance 2	Répartition des laches Lecture de la doc
Séance 3	Prise en main du moteur Code de détection de couleur Premiers codes pour les poussoirs
Séance 4	Mise en route du capteur de présence Mise en mouvement du convoyeur et des poussoir Calcul des temps et distances d'arrêts
Séance 5	Début des liaisons serial python/nucleo Mise en place de la detection pour arrêter le convoyeur
Séance 6	Fin de la mise en place des liaisons et résolution des problèmes concernant cet aspect très important du projet
Séance 7	Test du prototype Optimisation de la vitesse du convoyeur Création du diaporama
Séance 8	Présentation finale

Figure 6: Planning du groupe

5 Difficultés rencontrées

1. La plus grosse difficulté était la mise en place de la liaison serial et son optimisation. En effet, il a fallu optimiser la place de la liaison serial dans les programmes pour ne pas utiliser de la mémoire vive inutilement.

2. Le traitement d'image a aussi posé un problème, car au début, on cherchait à avancer en comprenant toutes les lignes de code alors que ce n'était pas forcément nécessaire.

3. La compréhension du moteur du convoyeur

4. Les premières séances où nous avons réunis les travaux de chacun étaient aussi compliquées, car tout le monde n'avait pas développé les mêmes compétences.

Conclusion du projet

En conclusion, ces mois de travaux nous ont permis de développer des compétences transverses et nous ont montrés l'importance de la communication dans un travail en équipe. L'importance de persévérer se reflète dans ce projet et nous pouvons être fier de nous sur la finalité d'avoir rendu un système fonctionnel.