HONORÉ Nathan
PREVOTEL Lucas
KOBI Gérald

PROJET IéTi : Vision industrielle©

Introduction

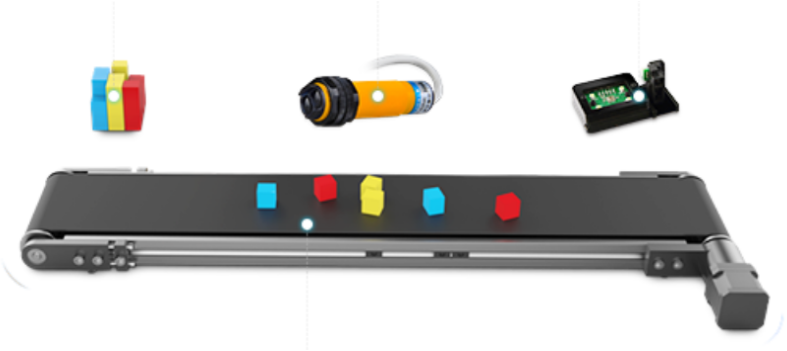

Figure 1 : Illustration des éléments du projet

Vision Industrielle© consiste en un projet portant sur la mise en œuvre d'un moyen de tri de pièces industrielles de différentes couleurs. Initialement, ce projet s'inspire du convoyeur Dobot Magician qui est utilisé pour faire du tri de pièces ou détection de défaut dans les chaînes de production.
Le kit comprend 4 éléments représentés sur la première figure (resp. de gauche a droite) : des pieces de couleur, un détecteur de présence IR, un capteur de couleurs et un convoyeur. Cependant, nous n'utiliserons pas le capteur de couleur, mais une webcam ainsi qu'un logiciel de traitement d'images.
Nous disposons également de tout le matériel électrique et informatique nécessaire : une carte nucléo L476RG, un Raspberry Pi, un GBF, des générateurs de tension et des fils électriques.

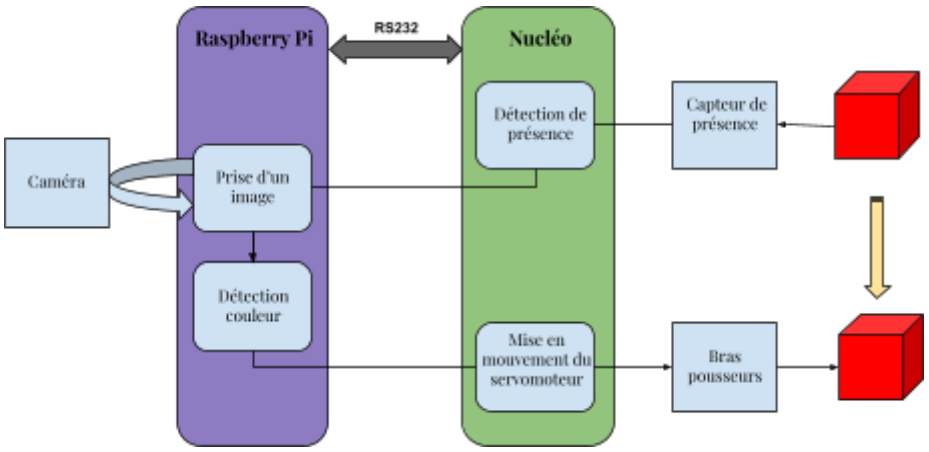

Figure 2 : Schéma de principe du système

Le principe du système est le suivant (cf figure 2) : on établit une liaison de type RS232 entre le Raspberry Pi et la carte nucléo via une liaison RS232. La carte nucléo commande à la fois le capteur de présence mais aussi le servomoteur du bras pousseur, et le Raspberry Pi commande la webcam.

Dès lors, lorsque la pièce à trier placée sur le convoyeur passe devant le capteur de présence, la carte nucléo envoie cette information au Raspberry Pi. La webcam prend une photo et un programme de traitement d'images renvoie à la carte nucléo la couleur de la pièce. Celle-ci ordonne à son tour au servomoteur de tourner d'un certain angle en fonction de la couleur reçue. Les pièces sont donc triées en fonction de leur couleur.

Nous avons donc découpé le projet en 4 sous-projets :

- Détection des couleurs
- Mise en route du convoyeur et du capteur
- Mise en mouvement du bras pousseur
- Etablissement Liaison Python-Nucléo

Une fois chacune de ces fonctionnalités validée via la conception de systèmes indépendants, nous les combinerons pour élaborer le système final.

Réalisation du prototype

a. Détection des couleurs

L'idée est de récupérer une photo de la pièce à trier par l'intermédiaire d'une caméra. Dès lors, par un programme Python, nous allons retirer de l'image la couleur dominante qui sera considérée comme la couleur de la pièce. Les pièces sur lesquelles nous travaillons sont des cubes de couleur rouge, bleu, vert ou jaune. Dans le cas de la couleur

jaune, celle-ci n'étant pas une couleur de pixel (RGB), nous choisissont arbitrairement son taux de rouge et de vert.

L'écriture du programme Python s'effectue sur un ordinateur PiTop, utilisant une carte Raspberry Pi, et pour le traitement d'image, nous utilisons la librairie OpenCV ce qui permet de fluidifier le code.

La caméra prend des photos en continu et renvoie pour chacune, la couleur dominante. Le code simplifié est représenté sur la figure ci-dessous (cf annexe 1) :

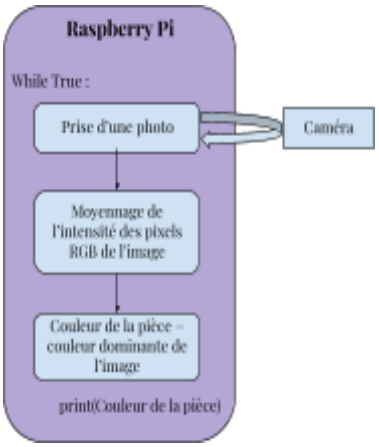

Figure 3 : Code simplifié du traitement d'images

Le principe de ce code est simple : on prend une photo à l'aide d'une commande prédéfinie de la bibliothèque OpenCV, et on récupère un tableau de dimension 3xN (avec N le nombre de pixels) contenant sur une ligne le niveau de rouge, de vert et de bleu d'un pixel.

Dès lors, il suffit de faire la moyenne des nombres de chaque colonne. On obtient un tableau de 3 éléments, et on détermine quel est l'élément maximum du tableau qui correspond à la couleur dominante.

b. Mise en route du convoyeur et du capteur

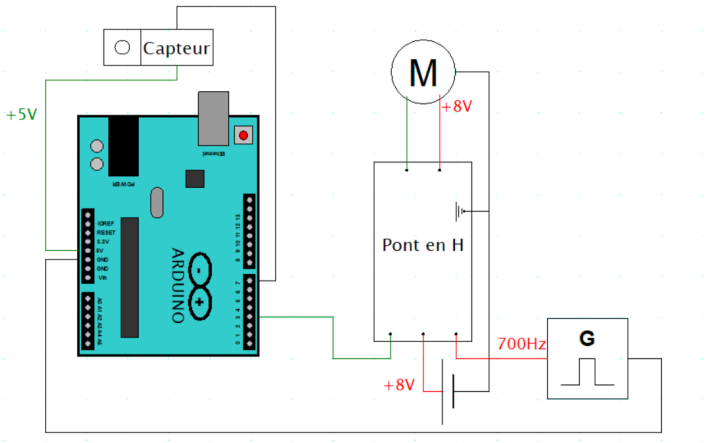

Figure 4 : Schéma électrique du montage convoyeur-capteur

Le mouvement du convoyeur est contrôlé par un moteur pas à pas alimenté en 8V. Nous l'alimentons donc avec un générateur de courant externe. Un pont en H permet de faire le lien entre le moteur et le générateur de courant externe. De plus, on commande la vitesse à l'aide d'un générateur basse fréquence qui envoie un signal créneau de 700 Hz.

Pour ce qui est du capteur de présence, on le branche simplement en 5V à la carte Nucleo. De manière passive, si il ne détecte aucune pièce (voyant éteint), il envoie un "1". Inversement, s'il détecte une pièce (voyant allumé), il envoie un "0". Ainsi, on choisit de s'intéresser au front descendant "10", c'est-à-dire lorsque la pièce entre dans le champ de détection du capteur de présence, pour considérer qu'une pièce est détectée.

c. Mise en mouvement du bras pousseur

Le bras pousseur est mis en mouvement par un servomoteur. L'avantage de ce dernier est d'avoir une limite de débattement d'angle et une grande résolution. Il permet de facilement choisir l'amplitude de mouvement du bras pousseur.

Le servomoteur est alimenté en 5V et est contrôlé en sortie Pwm par la Nucléo. Le code sous Mbed est extrêmement simple (cf annexe 2) : on initialise l'angle du servomoteur avec la fonction *pulsewidth*. Avec la même fonction, on décide d'un certain angle en fonction de l'information envoyée par le Raspberry Pi, chaque angle correspondant à un certain déplacement de la pièce sur le convoyeur. On revient ensuite à la position initiale.

Ainsi, n'utilisant qu'un unique bras pousseur, le tri des pièces s'effectue selon leur position sur le convoyeur : Pour les pièces jaunes, pas de déplacement, les bleues un peu, les vertes un peu plus et les rouges beaucoup.

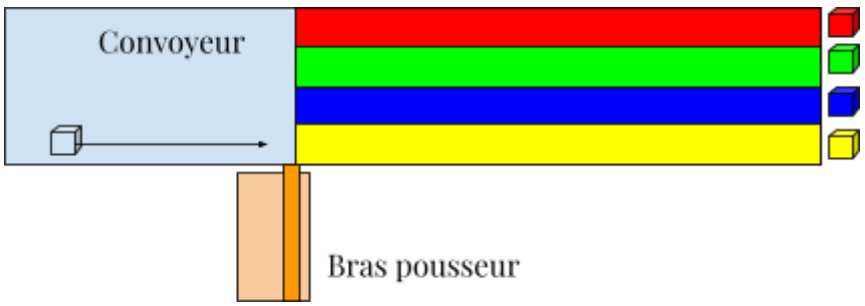

Figure 5 : Principe du tri par action d'un bras pousseur

d. Liaison Python - Nucleo

Pour réaliser la liaison entre le PiTop et la carte Nucleo, on utilise un câble RS232. Cette étape est cruciale car elle permet la synchronisation de la détection, le traitement d'image et le tri. La Nucleo repère les fronts descendants et envoie alors un caractère vers le PiTop. Arbitrairement on a choisi le caractère "1" . Ainsi, lorsque Python reçoit ce caractère, il transmet la couleur de l'objet à la Nucleo par le renvoie d'un autre caractère dépendant de la couleur : "V" pour vert, "R" pour rouge, "B" pour bleu et "J" pour jaune.

Pour le programme Python (cf annexe 1), on import “serial” tandis que pour la Nucleo, sous MBed-OS 6 on utilise les fonctions pc.write et pc.read. L’utilisation d’une fonction pc.readable n’est pas nécessaire ou plutôt est inutile car même si la Nucleo n’envoie pas un “1”, Python reçoit un “0” en continu.

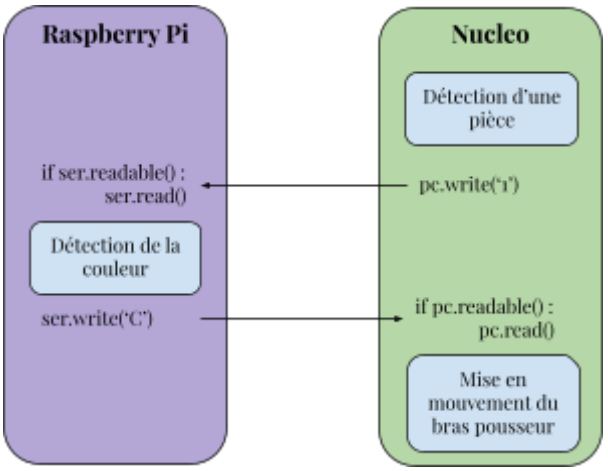

Figure 6: code simplifié de la communication entre la nucléo et le Raspberry Pi

Résultat final

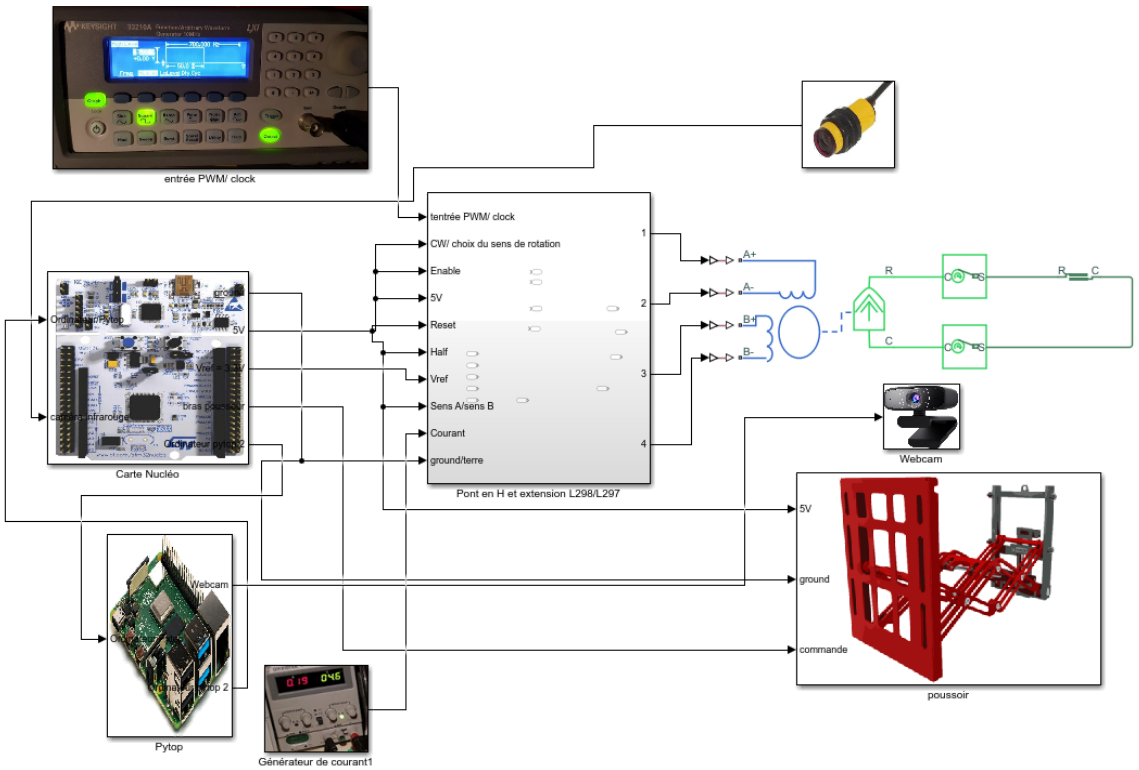

Figure 7 : Schéma électrique du montage final

Après avoir validé manuellement chacun des modules, nous avons entrepris la conception du modèle final dont le schéma est représenté sur la figure 7.

On doit placer le capteur à une position précise de telle sorte que le bras pousseur s'active bien lorsque la pièce est en face. Pour celà, on calcul simplement le temps à partir de la vitesse et des positions relatives des composants. La caméra est placée en amont du capteur de telle manière qu'elle est le temps d'avoir la couleur de la pièce avant que celle-ci ne soit détectée par le capteur de présence.

Les pièces sont déposées manuellement sur le convoyeur au niveau de la bande correspondant à la couleur jaune (cf.partie c.).

Il a fallu faire quelques retouches au niveau du temps d'attente pour le bras pousseur car il faut compter aussi un retard dû au à l'échange de données des deux systèmes communiquant et aussi des retouches au niveau du positionnement de la caméra et de l'éclairage pour que la détection des couleurs se réalise sans accroc.

Le prototype est un succès, il réalise la tâche qui lui est confiée à savoir trier des pièces en fonction de leur couleur, avec une rapidité tout à fait raisonnable (environ 20 pièces par minute). Le montage réel est représenté ci-dessous :

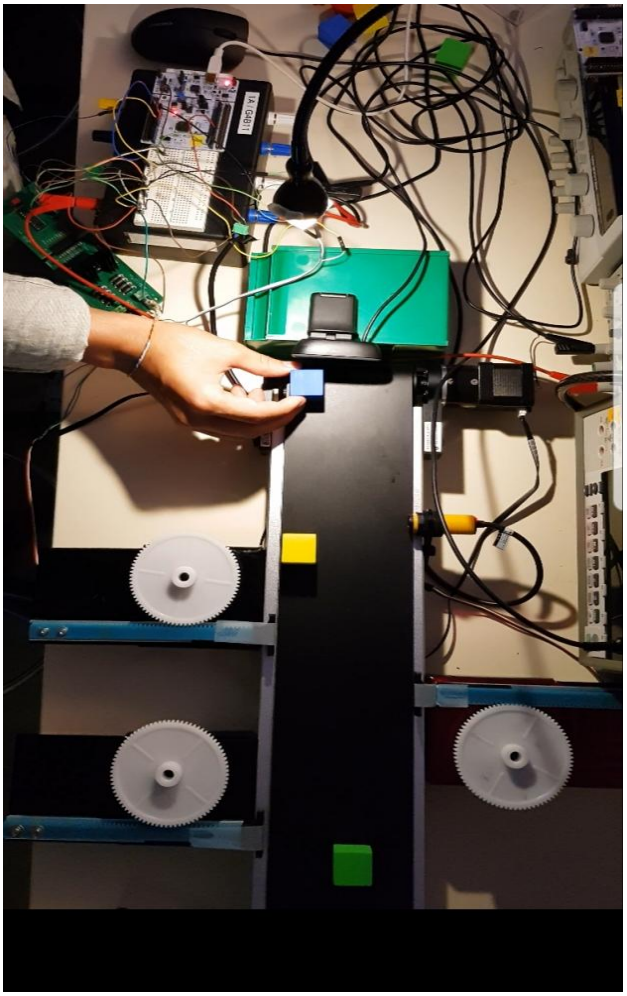

Figure 8 : Montage final

Planning

Séance 1	- Découverte du projet et des composants mis à dispositions - Lecture de la documentation technique - Réflexion sur les directions à prendre
Séance 2	- Début de la programmation du code pour la détection des couleurs - Prise en main du moteur pas à pas avec le pont en H.
Séance 3	- Détection instantanée et en continu des couleurs rouge, vert, bleu et jaune. - Mise en mouvement du convoyeur.
Séance 4	- Mise en route du capteur de présence et vérification de sa fonctionnalité. - Mise en mouvement du bras pousseur.
Séance 5	- début de la mise en place de la liaison Python-Nucleo avec envoie d'un caractère lors de la détection d'une pièce par le capteur de présence.
Séance 6	- Transmission de la couleur de Python vers la Nucleo avec une réponse d'un bras pousseur.
Séance 7	- Synchronisation de tous les composants et premiers tests du prototype. - Changement de méthode de détection. On prend des photos à intervalles réguliers plutôt qu'une vidéo.
Séance 8	- Derniers tests du prototype avant la présentation. - Présentation orale.

Difficultés rencontrées

Au cours de ce projet, nous avons fait face à de nombreuses difficultés :

- La prise en main de composants nouveaux : le moteur pas à pas, le servomoteur, le PiTop,...

- Le temps de traitement d'images. Initialement, nous avions codé nous même le programme et il était très lent. La bibliothèque OpenCV est donc indispensable.
- Sur les trois bras pousseurs, Seul un fonctionnait convenablement. C'est pourquoi, plutôt que de pousser les pièces hors du convoyeur pour les trier, nous avons choisi de les trier directement sur le tapis du convoyeur.
- Une mise à jour de MBed-OS a rendu toutes les fonctions liées à la communication entre les deux systèmes inutilisables. Initialement, on utilisait une connexion "Serial" avec les fonctions pc.putc et pc.getc mais on a dû passer à une connexion "UnbufferedSerial" avec les fonctions pc.write et pc.read. On a downgradé la version de MBed pour essayer de retrouver celle sur laquelle avait été codée notre programme mais nous ne sommes pas parvenus à trouver la bonne version.
- Pour la caméra, l'idée de départ était qu'elle prenne une vidéo en continu et que le traitement d'images s'effectue sur certaines images de celle-ci. Indépendamment du système, celà fonctionnait très bien, on pouvait pointer n'importe quelle pièce et on avait la couleur en instantanée. Mais une fois intégrée à notre système, la caméra avait une latence d'une dizaine de secondes ce qui rendait le prototype inutilisable.

Analyse du travail d'équipe et conclusion

Ce projet qui nécessite rigueur et organisation, nous a permis de développer un esprit d'équipe et de coopération fort. En effet, il a fait ressortir en chacun de nous les qualités nécessaires au bon fonctionnement du groupe tout au long de ces 5 derniers mois. Nous sommes tous fiers du résultat et des efforts que nous avons fournis. Des pistes d'améliorations peuvent être envisagées : Un code de détection de forme pour s'assurer que la couleur dominante de l'image est bien la couleur de la pièce, Un contrôle du convoyeur à l'aide de la Nucleo pour pouvoir choisir la vitesse ou faire des arrêts, l'installation d'une "boîte noire" pour s'assurer un éclairage neutre et uniforme,...

Annexe

Annexe 1 : Programme python détection couleur et liaison avec la carte Nucleo

```
import cv2 as cv
import serial

ser = serial.Serial('/dev/ttyACM0',9600,timeout = 1) #ouverture du
port série entre la nucléo et python

def recuperationphoto():
    cap=cv.VideoCapture(0)
    ret,frame = cap.read()
    return frame

c = '' # Initialisation de la variable couleur

while True :
    image = recuperationphoto()
    mean = cv.mean(image)[:3] # Moyenne des pixels RGB de l'image
    couleur = max(mean) # la couleur de la pièce est la couleur
dominante de l'image
    if couleur == mean[2]:
        if mean[1] > 0.93*couleur :
            c = 'J' #La couleur est jaune
        else :
            c = 'R' #La couleur est rouge
    elif couleur == mean[1]:
        if mean[2] > 0.93*couleur :
            c = 'J' #La couleur est jaune
        else :
            c = 'V' #La couleur est verte
    else :
        c = 'B' #La couleur est bleue


    if ser.readable(): #Si il y a quelque chose à lire dans le câble
==> True (fonction inutile en pratique)
        a = int.from_bytes(ser.read(),"little") #On récupère le
caractère reçu via le serial
        if a == 49: #équivalent caractère "1"
```

```
        # On envoie renvoie la couleur de la pièce par la fonction
ser.write().
            print('signal reçu')
            if c == 'J':
                ser.write(b'J')
                print('signal envoyé',c)
            elif c == 'B':
                ser.write(b'B')
                print('signal envoyé',c)
            elif c =='V':
                ser.write(b'V')
                print('signal envoyé',c)
            else:
                ser.write(b'R')
                print('signal envoyé',c)
```

Annexe 2 : Programme MBed pour la carte Nucleo avec détection d'une pièce, liaison avec Python et mise en mouvement du bras pousseur

```
#include "mbed.h"
#include "platform/mbed_thread.h"


PwmOut          servo_mot(D3);
UnbufferedSerial          pc(USBTX, USBRX);
InterruptIn     capteur(D7);

void envoie(void);

void envoie(){
    char c = '1';
    pc.write(&c,1);
}

int main() {
    servo_mot.pulsewidth_us(850);
            wait_us(500000) ;
    int i, time = 1500;
    servo_mot.period_ms(20);          // Initialisation période
    servo_mot.pulsewidth_us(500);   // Initialisation en position

    pc.baud(9600);
    char couleur;

    // transmission d'un caractère vers le PC
        //pc.printf("%d0\r\n", capteur.read());
    capteur.fall(&envoie); //on détecte la présence d'un objet.

    while(1){

    // transmission d'un caractère vers le PC
        //pc.printf("%d\r\n", capteur.read());

    // réception d'un caractère envoyé par le PC
        if (pc.readable()) {
            char r;
            pc.read(&r,1);
            // couleur = pc.read(); //on récupère la couleur de l'objet.
            servo_mot.pulsewidth_us(500);
            wait_us(2800000) ;
            if (r == 'R'){
                wait_us(0);
                servo_mot.pulsewidth_us(2200);   // Angle positif
                wait_us(500000);
                servo_mot.pulsewidth_us(500);
            }
            if (r == 'V'){
                wait_us(0);
                servo_mot.pulsewidth_us(1150);   // Angle positif
                wait_us(500000);
                servo_mot.pulsewidth_us(500);

            }
            if (r == 'B'){
                wait_us(0);
                servo_mot.pulsewidth_us(850);   // Angle positif
                wait_us(500000);
                servo_mot.pulsewidth_us(500);

            }
        }
    }
}
```

Annexe 3 : Chaîne fonctionnelle du système