Rapport final de projet

Pompier

RIO Guillaume & BHIKIR Oumnia

1. Introduction :

Pompier, est un robot autonome de dimension 20*40*30 cm entraîné à détecter et reconnaître une porte. Muni de capteurs, il se déplace sur des chenilles en évitant les obstacles pour se positionner droit devant la poignée.

Pesant 4.5kg, sa vitesse est de 0,4m/s dans un milieu avec une inclinaison maximale de 40°.

A l'aide d'un bras robotique articulé, à cinq degrés de liberté, il saisit la poignée dans un mouvement de flexion pour ouvrir la porte.

Le châssis contient 2 moteurs CC alimenté à 7.2V, alors que le mouvement du bras est assuré par 6 servomoteurs alimentés à 6.3V.

2. Structure :

Le prototype (figure 1) est composé d'un châssis à chenilles métalliques à 2 étages.
Sur le 1er étage se trouve la Jetson Nano et les câbles la reliant à la caméra.
Le 2ème étage est un support pour la boîte paramétrique cubique contenant : L'Arduino Mega, le module PCA9685 <driver pour servomoteurs>, la carte Crypton <driver pour moteur CC et source d'alimentation pour l'Arduino>.
Au-devant nous retrouvons les modules capteurs laser de distance de part et d'autre de la caméra Logitech.
Au-dessus est vissé le bras robotique entièrement imprimé en PETG, puis assemblé de la base à la pince et enfin relié avec une communication I2C à l'Arduino.

Figure 1 Pompier, prototype

3. Schéma électrique

La carte Arduino MEGA représente le centre de commande, son GND est la masse commune de l'ensemble des composants.
Les flèches (figure 2) représentent les câbles, il relie l'Arduino par les pin RX-TX à la Jetson Nano et par les pin 4-5-6-7 au pont en H qui contrôle les moteurs CC et alimente entre autres la carte.
Le SCL et SDA sont communs au driver des servomoteurs et capteurs laser.
Un convertisseur AC-DC convertit la tension du secteur en 6.3V nécessaire au bon fonctionnement des **6** servomoteurs du bras.

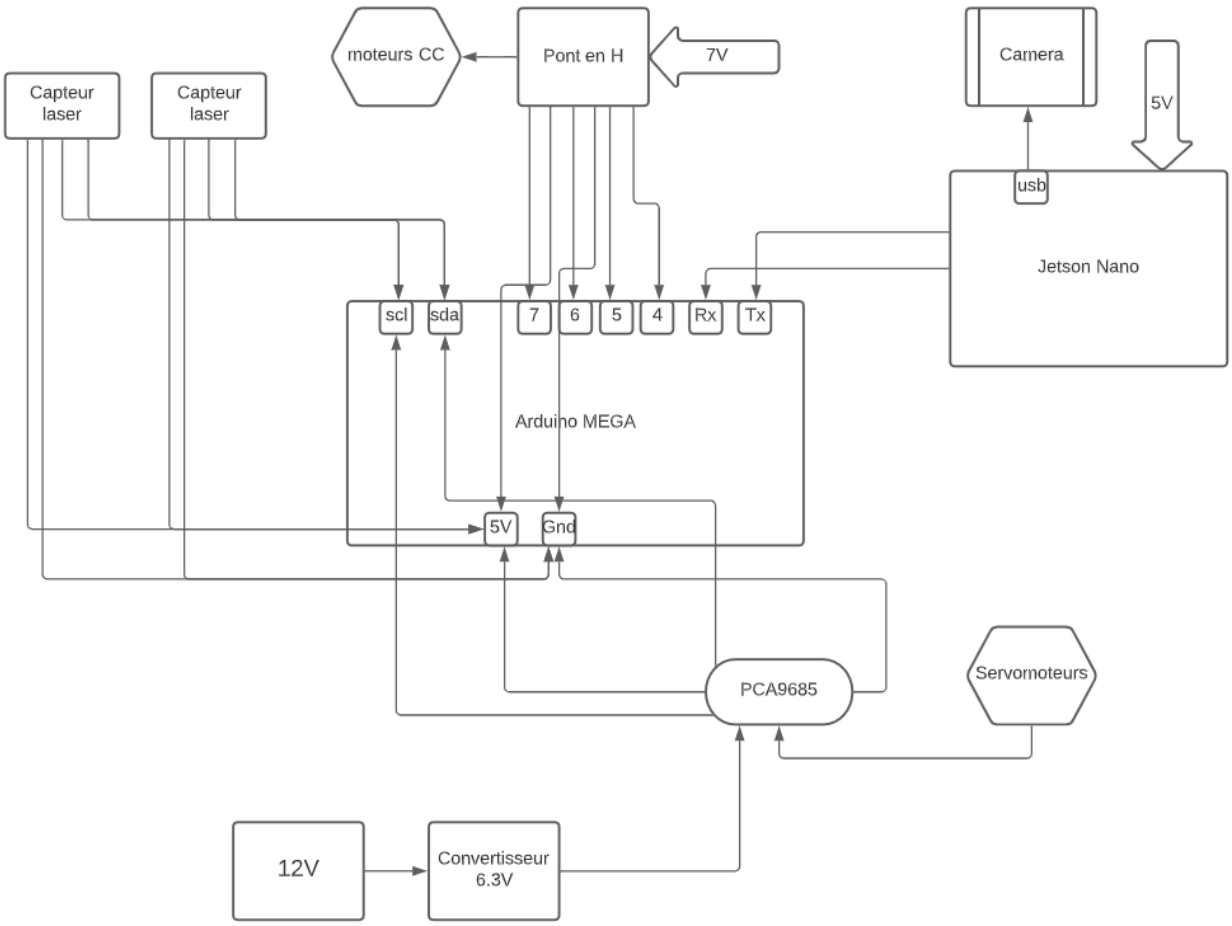

Figure 2 Schéma électrique du projet

4. Algorithme de fonctionnement

Nous allons décrire dans un premier temps, l'organisation de nos fichiers au sein des deux cartes (Arduino Mega et Jetson Nano).

La carte Arduino, contient 2 classes et un fichier main qui sont répartis de la manière suivante (figure 3):

- Main.ino
- CapteurLaser.ino et son fichier de définition CapteurLaser.h
- ControleMoteur.ino et son fichier de définition ControleMoteur.h

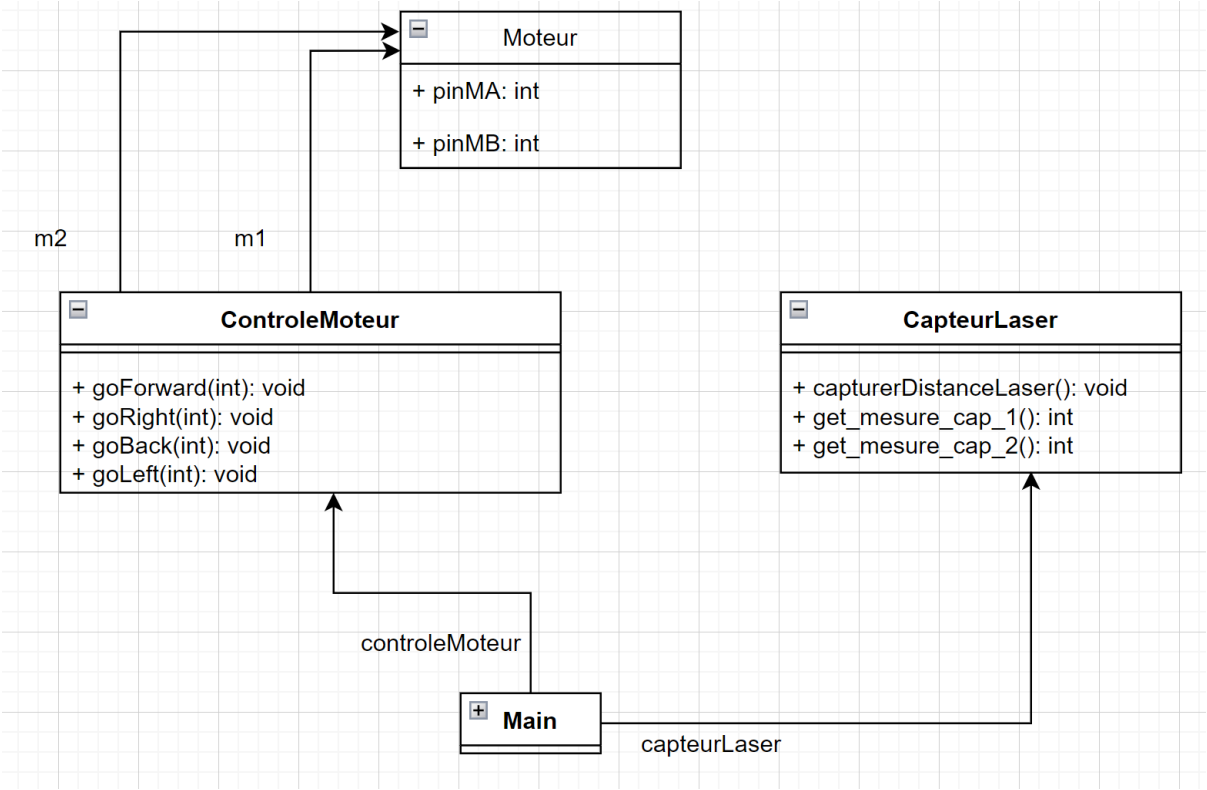

Figure 3 Diagramme de Classe UML

Sur la carte Jetson nano, nous avons 3 fichiers python :

- ser.py contenant le paramétrage de la communication UART et les fonctions servants à recevoir et envoyer des messages.
- vision.py contenant les fonctions liées à la vision, on cite : prendre une photo, chercher la porte, détecter la poignée puis trouver le nombre de pixels entre le milieu de la porte et le milieu de la photo.
- main.py contient les 2 threads utilisant les photos des 2 fichiers précédents.

On rappelle qu'un thread est l'unité de base à laquelle un système d'exploitation alloue du temps processeur. Chaque thread a une priorité de planification et maintient un ensemble de structures utilisé par le système pour enregistrer le contexte du thread quand l'exécution du thread est en pause.

Nous allons maintenant décrire un peu plus en détail le fonctionnement. Tout d'abord, l'Arduino va envoyer périodiquement la distance mesurée par les capteurs vers la Jetson Nano ainsi qu'une variable indiquant si le robot est en mouvement ou non.

Du côté de la Jetson, si le robot n'est pas en mouvement, la carte va donner l'ordre de prendre une photo et de l'analyser. Ensuite on va analyser la position de la porte comparée au centre de l'image et tenter d'obtenir la position de la poignée. Enfin, un message sera envoyé vers la carte Arduino contenant l'information concernant le sens de rotation du robot (tourner à droite, à gauche ou avancer) accompagnée de la position de la poignée. L'Arduino va finalement envoyer les commandes aux moteurs afin de tourner et bouger le bras, on passe alors dans l'état mouvement actif.
Une fois le mouvement exécuté, on se remet en état initial et la boucle reprend.

Voici les étapes à suivre pour utiliser notre robot :

- Etape 1

Alimenter l'ensemble des cartes.

- Etape 2

Connecter un ordinateur au même réseau que la carte Jetson Nano et se connecter en ssh.

- Etape 3

Lancer les commandes :

- cd /mnt/sd/IA
- python main.py

5. Coût du projet

Nous estimons tout d'abord le coût des matériaux utilisés :

Composant	Prix (€)
Châssis métallique à chenilles + 2 moteurs CC	80
Boite paramétrique (6 planches de contreplaqué)	9
Porte	20
Bras (500g PETG)	4
Convertisseur AC-DC	18
4 servomoteurs MG + 2 micros Sg90	143
Jetson Nano	182
Arduino Mega	51
Caméra Logitech	52
2 capteurs laser	26
Crypton (Pont en H)	30
Visserie	20
TOTAL	**635**

L’ensemble des enseignements robotique expérimentale a duré 100h, ajoutant à cela 220h de travail en dehors des cours.
Le projet nous a pris au total 320h, équivalant à 7630 €/personne.

Le projet a donc coûté au total 15895€.

6. Les problèmes surmontés

Notre bibliographie manquait de comparaison avec les projets existants, nous aurions dû approfondir nos recherches, et établir une liste des problèmes techniques rencontrés dans les précédents aboutissements.
De ce fait, nous aurions pu éviter un bon grand nombre de complications.

1. le paramétrage de la carte Jetson Nano : nous avons tenté d’utiliser Jupyter avec un conteneur docker donné par Nvidia, cependant notre carte est une version JN30D, pytorch était mal installé, ce qui rendait le conteneur inutilisable. Nous avons donc décidé de tout réaliser à même la carte et donc d'installer Python et toutes les librairies nécessaires sur la carte puis router la carte SD.

2. L’utilisation d’une IA afin de reconnaître notre porte : avec Keras et Tensorflow nous avons codé notre propre IA. Toutefois, les résultats étaient mitigés.
C’est alors que nous avons changé d 'approche et sommes repartis sur une analyse d’image pour détecter notre porte. Pour ce faire, nous avons installé OpenCV et à l’aide des fonctions disponibles, la réalisation de notre algorithme de détection a été accomplie.

3. La carte Arduino, ne contient pas de fonction thread, ce qui contraignait l'exécution de nos tâches, nous nous sommes alors servis d’une librairie (easyrun) afin de lancer des tâches périodiquement pour éviter que les buffers UART ne saturent.

4. Le bus I2C : nous avons plusieurs composants utilisant ce bus, or la carte ne dispose que de 2 bus différents.

Nous avons donc branché les capteurs sur un même bus, en alternant la communication entre eux.

5. Le bras : fragile, il a fallu changer les palonniers liant les pièces au moteur souvent et éviter les mouvements brusques qui pourraient désassembler les composants.

Encapsuler les servomoteurs dans une unique pièce aurait donné plus de robustesse à la modélisation.

6. Les calculs : Afin de déterminer les angles de mouvement du bras, il fallait effectuer des calculs de matrices de rotation qui deviennent vite compliquées dans un espace à 5 degrés de liberté. Puis tout transcrire en code, l’imprécision de l’Arduino et les bruits parasites altéraient énormément la fluidité du mouvement.

7. Conclusion et perspective

Nous avons su mettre en place un robot mobile de façon autonome apte à détecter une porte via une caméra, néanmoins la vitesse de déplacement vers la porte est inférieure à nos estimations.
L'implémentation du bras a engendré un faux contact, ce qui nous a poussé à envisager la réduction du nombre de câbles afin de pouvoir réagir plus activement dans de telles situations.
Par ailleurs, ouvrir la porte était notre objectif et nous l'avons presque atteint. Seule la pince n'est pas intégrée dans le code contrôlant le bras. Le bras peut ainsi se déplacer dans l'espace vers un point choisi par l'utilisateur, dans notre cas ce point représente bien les coordonnées de la poignée ; cependant il ne peut la saisir.
Avec plus de temps et de travail, nous pouvons faire en sorte d'avoir un rendu plus efficace. Toutefois nous ne sommes pas surs de vouloir travailler sur le même projet durant nos prochaines années d'étude, mais nous en sortons grandis et instruits. Pompier, était notre premier robot autonome.

8. Bibliographie

http://www.vorobotics.com/wiki/index.php?title=Bras_Robot_Educatif
https://forum.arduino.cc/t/resolu-fonctions-trigonometriques-et-variables-a-virgule-flottante/235306/12
https://www.academia.edu/39731111/Etude_et_r%C3%A9alisation_dun_bras_de_robot_%C3%A0_base_de_carte_Arduino
https://www.kongakura.fr/article/OpenCV_Python_Tutoriel#:~:text=Comment%20afficher%20une%20image%20avec,et%20l'image%20%C3%A0%20afficher.
https://www.nvidia.com/fr-fr/autonomous-machines/embedded-systems/jetson-nano/
https://developer.nvidia.com/embedded/community/jetson-projects
https://sites.google.com/view/aide-python/applications/traitements-et-analyses-dimages