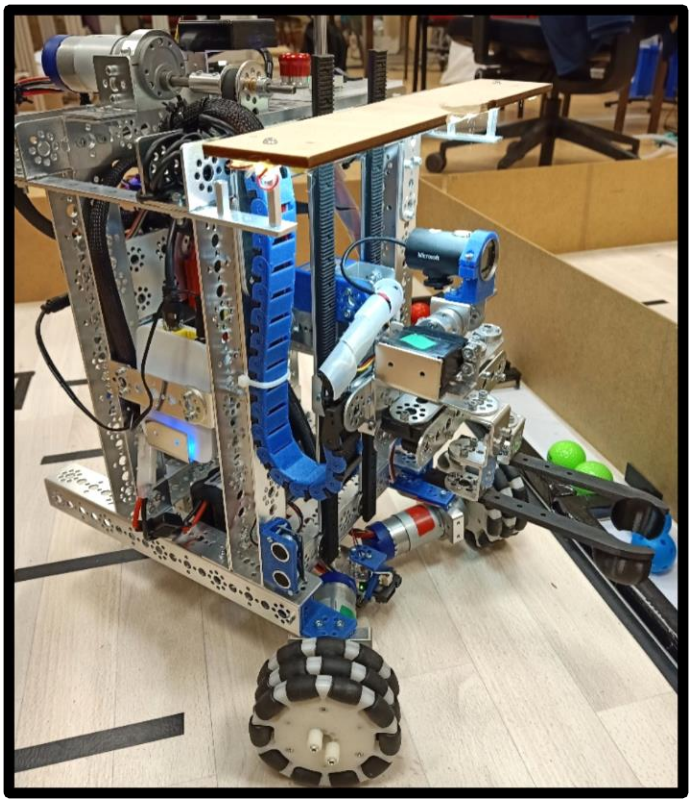
Conception et développement d'un robot logistique

Rapport de projet de fin d'études IMA 2020

Etudiants :
Mathis DUPRE IMA5
Valentin PITRE IMA5

Tuteur :
Vincent COELEN



Remerciements

Nous souhaitons tout premièrement remercier Vincent COELEN, notre tuteur école, qui nous a suivi durant nos entrainements dès la fin des finales régionales de robotique mobile. Nous sommes reconnaissants envers sa patience et le temps qu'il nous a accordé dans l'apprentissage du logiciel LabVIEW et dans la résolution des problèmes auxquels nous avons dû faire face durant ce projet.

Nous remercions également Florent CHRETIEN, notre entraineur durant la compétition, qui est venu nous rendre visite pour suivre notre avancée et nous donner de nombreux conseils.

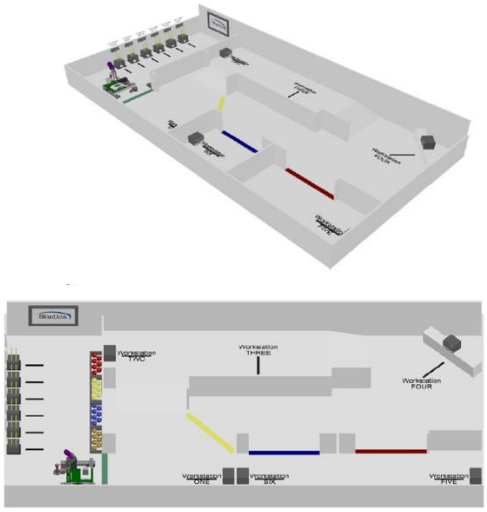
Finalement nous sommes entièrement reconnaissant envers le COFOM Worldskills France pour le financement de nos entrainements ainsi qu'une partie du matériel additionnel commandé durant le projet. Nous remercions également l'école PROMEO pour leur accueil à Beauvais et le prêt du kit de robotique.

Table des matières

Introduction

Ce projet se déroule dans le cadre de la préparation à la compétition Euroskills en robotique mobile. Les Euroskills sont la déclinaison européenne de la compétition internationale Worldskills, considérée comme les jeux Olympiques des métiers. Tous les deux ans, pendant 3 à 4 jours, l'affrontent dans une cinquantaine de métiers les champions nationaux de l'ensemble des pays du continent européen. Cette année nous sommes les représentant de l'équipe de France dans le métier robotique mobile qui permet de mettre en avant les avancés de la robotique dans le domaine industriel. Ce projet fait alors partie intégrante de notre entraînement et notre préparation à la finale européenne à Graz (Autriche) en septembre 2019.

Il reprend les objectifs du sujet de robotique mobile des Worldskills 2019 à Kazan (Russie) Ce projet consiste à la réalisation d'un robot autonome mais aussi pouvant être télé-opéré, évoluant sur un terrain simulant une usine. À partir de son point de départ, le robot doit être capable de se rendre aux différentes "Workstations" disséminées sur le plateau afin de déposer un "Component carrier" après l'avoir rempli au stand de chargement. La particularité est que sur chaque "Workstation" est inscrit un code barre allant de pair avec celui de l'un des stands de chargement. Le "Component carrier" doit donc y être acheminé sans erreur. L'ordre dans lequel le robot doit se présenter aux workstations peut être soit définie au choix du compétiteur, soit définit par un tirage au sort. Un fois tous les Components carriers aux stands de chargement, le code barre qui leur est associé indique le nombre de boules, la couleur et l'ordre dans lequel y doit être rempli. Les boules sont à disposition du robot dans des emplacements prévus à cet effet.



Plan de l'usine dans laquelle évolue le robot

1. Méthode de travail

Un projet de fin d'étude, se doit d'être réalisé de la façon la plus professionnelle possible, c'est-à-dire en suivant une méthode de gestion de projet.

Après discussion avec notre expert métier, il serait préférable d'adopter une gestion de projet dite "Agile", qui consiste à livrer toutes les semaines un produit opérationnel, en ajoutant à chaque fois de nouvelles fonctionnalités. Ce type de gestion de projet a plusieurs avantages :

- Permet d'avoir un logiciel et un robot opérationnel tout au long du projet
- Avoir une grande adaptation au changement (essentiel quand l'on sait que 30% du sujet change le jour de la compétition)
- Avancer par petits objectifs, les valider, s'adapter en fonction de la situation du moment, et ainsi de suite jusqu'à la fin du projet

Pour nous aider dans l'organisation du projet, nous avons mis en place plusieurs outils :

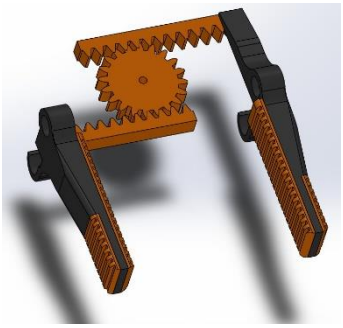
- Microsoft TEAM: on a une équipe qui regroupe notre tuteur école, notre expert métier, ainsi que d'anciens compétiteurs français. On partage donc nos avancées, nos interrogations et récupère les retours/expériences des anciens compétiteurs.
- TRELLO: nous permet de gérer le projet, et les tâches en équipe, nous l'utilisons pour planifier et organiser tout notre travail. Chaque colonne représente un état du projet, dans lequel on glisse des "cards" qui représente une tâche à réaliser. On peut donc, pour chaque aspect d'un projet, laisser nos commentaires, répartir nos tâches, centraliser des informations et suivre l'état d'avancement.
- Drive: Pour stocker et partager rapidement tous nos fichiers. Synchronisation et partage automatique de notre dossier de compétition, stocké sur nos PC respectif.

Pour continuer dans la gestion de projet dite "Agile", la livraison d'un produit opérationnel par semaine est simulée par un retour par vidéoconférence en fin de semaine à notre expert métier et tuteur école.

2. La mécanique du robot

a. Bras et pince

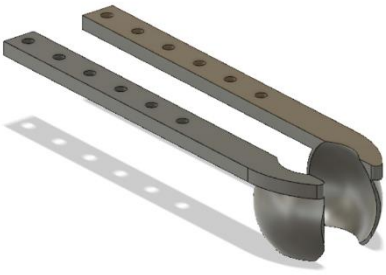
La pince a été un de nos plus grand défi mécanique de ce PFE, le kit ne fournit aucune notice de montage. Donc nous avons dû réfléchir à un système simple et robuste pour convertir le mouvement de rotation du servomoteur en un mouvement de translation pour ouvrir et fermer la pince. Deux crémaillères à engrenage perpendiculaire avec un pignon adapté, est un moyen simple de créer un mouvement linéaire avec un servomoteur. (Illustration ci jointe). La pince peut atteindre une ouverture maximum de 11 cm.

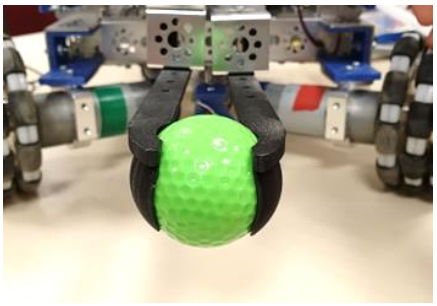


Principe mécanique de la pince

Etant donné que la pince est gérée par un servo commandé en vitesse, on ne peut pas choisir l'ouverture exacte de la pince. Un capteur de fin de course et des butées judicieusement bien placées, nous permette de commander trois ouvertures de pince : "Pince fermé", "Pince middle", "Pince full".

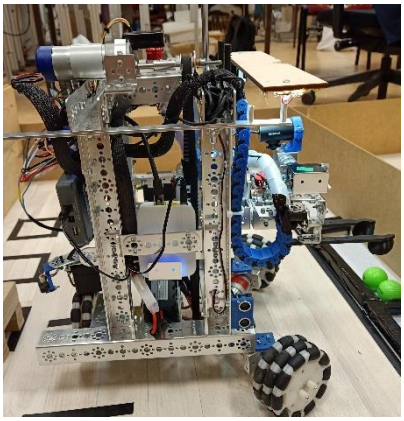
La pince se doit d'attraper une balle de golf le plus efficacement possible, pour répondre à cette problématique, les deux embouts de la pince sont des demi-sphères de diamètre identique aux balles de golf. Mais elle est aussi très fine pour pouvoir venir se glisser sous un « component carriers » et le soulever. En effet pour la prise des component carrier nous fonctionnons sous le principe d'un lève palettes. Le dessus de la pince est donc plat afin de pouvoir soutenir le « component carrier »





Modèle 3D de l'embout de la pince pour saisir les balles

Pour le mouvement de levée de lève palette un moteur couplé à une poulie entraine une courroie reliée à notre pince. Cette dernière est disposée sur un axe vertical avec des roulements à billes afin de pouvoir se déplacer aisément en hauteur. Le roulement présent dans le kit n'ayant pas des trous de vis adaptés aux pièces du même kit, nous avons conçu et imprimé une pièce en 3D servant d'adaptateur. Deux autres rails servent de guide pour ne pas faire pivoter la pince. Un capteur de fin de course est présent en position basse pour ne pas abimer le système et un encodeur sur le moteur permet de connaitre la hauteur du lève palette à n'importe quel moment.



b. Placement des roues

Le kit Studica fournit des roues holonomes mais il est aussi possible de commander des roues classiques si besoin.

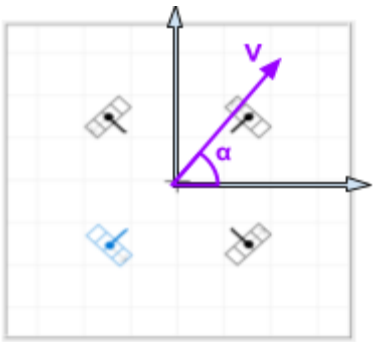
Roue holonome

- Les roues classiques ne permettent qu'un déplacement dans une seule direction et pose donc un problème en mécanique robotique. Puisque les changements de direction ne sont pas instantanés, il faut définir une trajectoire courbe dans le cas d'une géométrie directionnelle d'Ackermann (ex: voitures) ou opter pour la méthode de direction différentielle (ex: 2 roue simples et une roue folle).
- Les roues holonomes, permettent quant à elles un déplacement très libre et changement instantané de direction, mais elles ne sont utilisables que sur des terrains relativement plats et dont la surface est dure. Elles ont la particularité de disposer de galets en forme d'olive seront une répartition tangentielle le long de la roue. Chaque galet peut tourner sur lui-même de manière indépendante, ce qui confère à la roue sa propriété omnidirectionnelle.

Le sujet impose au robot d'être très maniable et le terrain n'est pas accidenté. Le choix des roues holonomes est le plus judicieux. Il peut dans ce cas se mouvoir dans toutes les directions de manière fluide et sans perte de temps en limitant les rotations du robot. Cependant toutes les architectures de robots ne sont pas compatibles avec les roues holonomes. Chaque roue doit être judicieusement disposées selon un angle et une position précise afin que le robot ait lui-même une propriété omnidirectionnelle. Durant nos tests nous avons pu évaluer 3 modèles :

- 4 roues holonomes espacées de 90° chacune avec des angles de roue à 45°. Celui-ci offre une excellente stabilité au robot. Les équations pour connaître la vitesse à appliquer à chaque roue en fonction du vecteur vitesse du robot sont :

$$\begin{cases} V_{roue\ avant\ gauche} = \|\vec{V}\| \sin(\alpha - \frac{3\pi}{4}) \\ V_{roue\ avant\ droite} = \|\vec{V}\| \sin(\alpha - \frac{\pi}{4}) \\ V_{roue\ arrière\ droite} = \|\vec{V}\| \sin(\alpha + \frac{\pi}{4}) \\ V_{roue\ arrière\ gauche} = \|\vec{V}\| \sin(\alpha + \frac{3\pi}{4}) \end{cases}$$



Pour l'odométrie les équations restent semblables :

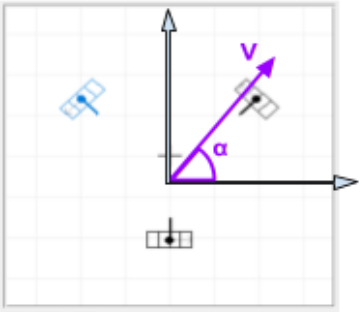
$$\begin{cases} d_{roue\ avant\ gauche} = \|\vec{d}\| \sin(\alpha - \frac{3\pi}{4}) \\ d_{roue\ avant\ droite} = \|\vec{d}\| \sin(\alpha - \frac{\pi}{4}) \\ d_{roue\ arrière\ droite} = \|\vec{d}\| \sin(\alpha + \frac{\pi}{4}) \\ d_{roue\ arrière\ gauche} = \|\vec{d}\| \sin(\alpha + \frac{3\pi}{4}) \end{cases}$$

Le problème majeur d'un robot à 4 roues est de maintenir l'ensemble des roues au sol. En effet lorsque le chemin est irrégulier et sans système d'amortisseur, il y a un très fort risque que seul 3 roues touchent le sol. Dans ce cas il est très difficile de se mouvoir dans la bonne direction ou par odométrie. Nous avons donc imaginé un châssis sur lequel la partie avant et arrière du robot sont reliées selon une liaison pivot, rendant le châssis "flexible". Ainsi les 4 roues sont constamment maintenues au sol.

Cependant nous sommes limités à seulement 4 moteurs pour la conception de notre robot. Nous avons donc fait le choix d'abandonner l'option à 4 roues afin de récupérer un actionneur pour la gestion de la hauteur du lève palette.

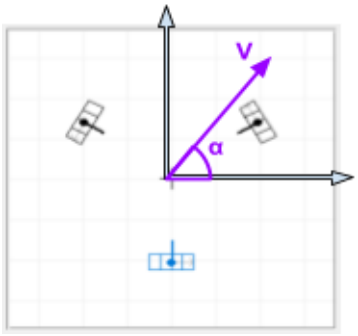
- 3 roues holonomes avec une formation en triangle isocèle. Les pièces du kit n'offrant que la possibilité d'angles multiples de 45° nous avons imaginé cette architecture avec 2 roues à 45° et -45° espacées toutes 2 de 90°. Elles sont donc à 135° de la roue arrière. Les équations deviennent donc :

$$\begin{cases} V_{roue\ avant\ gauche} = \|\vec{V}\| \sin(\alpha - \frac{3\pi}{4}) \\ V_{roue\ avant\ droite} = \|\vec{V}\| \sin(\alpha - \frac{\pi}{4}) \\ V_{roue\ arrière} = \|\vec{V}\| \sin(\alpha + \frac{\pi}{2}) \end{cases}$$



Cette architecture est tout aussi fonctionnelle qu'à 4 roues mais beaucoup moins stable. Nous sommes confrontés également un autre problème majeur, l'angle des roues à 45° imposent une résistance trop imposante pour le robot. Les galets des roues ne parviennent pas à tourner convenablement ce qui engendre de fortes oscillations latérales lors des déplacements vers l'avant, allant même jusqu'à la chute du robot. Avec ce choix là nous sommes donc fortement limités sur les directions de déplacement et la vitesse du robot. Nous abandonnons donc ce modèle pour la dernière architecture.

- 3 roues holonomes avec une formation en triangle équilatérale. Ici nous nous rapprochons beaucoup plus de l'architecture classique d'un robot omnidirectionnel à 3 roues avec un espacement de 120° entre chaque roue et des angles à 60°. Les pièces du kit n'offrant pas la possibilité de formation de ces angles, une pièce a été conçue puis imprimées en 3D avec des trous espacés du bon angle. Les équations deviennent alors :



$$\begin{cases} V_{roue\ avant\ gauche} = \|\vec{V}\| \sin(\alpha - \frac{5\pi}{6}) \\ V_{roue\ avant\ droite} = \|\vec{V}\| \sin(\alpha - \frac{\pi}{6}) \\ V_{roue\ arrière} = \|\vec{V}\| \sin(\alpha + \frac{\pi}{2}) \end{cases}$$

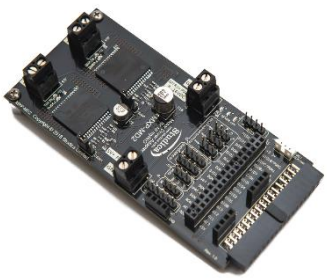
Le robot a gagné en stabilité par rapport à l'architecture précédente, et le problème d'oscillation est entièrement résolu grâce à un angle plus important entre les 2 roues avant.

3. Électronique

a. Cartes de contrôle

Le premier élément dont a besoin un robot afin de fonctionner est un cerveau électronique. La contrainte dans cette compétition est d'utiliser le MyRIO-1900 de chez National Instrument. Cette carte est principalement utilisée pour réaliser des systèmes temps réel, c'est à dire délivrer des informations exactes dans les délais imposés par les contraintes temporelles du système. Pour cela elle embarque un FPGA de Xilinx et un processeur double cœur ARM Cortex-A9. Ces deux éléments sont programmables grâce à un logiciel propriétaire : Labview. Le myRIO-1900 propose 6 sorties analogiques, 10 entrées analogiques, 40 entrées/sorties numériques. Il dispose également d'une puce wifi permettant le téléversement sans fil et un retour d'information des algorithmes vers l'ordinateur.

Afin de faciliter les branchements des différents capteurs et actionneurs sur le MyRIO, il est nécessaire d'utiliser les cartes MD2 fournies par STUDICA qui viennent directement se brancher sur ses ports MXD. Ces extensions proposent des ports de branchement pour 3 servomoteurs, 1 capteurs infrarouges, un capteur de distance à ultrasons, 2 encodeurs et 1 capteur de ligne. Chaque carte permet de gérer 2 moteurs grâce à un contrôleur moteur embarqué. Ces extensions doivent être alimentées séparément entre 8V et 16V pour le contrôle des moteurs. Un abaisseur de tension à 5V intégré permet d'alimenter les différents capteurs jusqu'à 3A. Une autre entrée 5V permet quant à elle d'alimenter séparément les servomoteurs.



MyRIO-1900 et carte d'extension MD2 de Studica

Lors de nos premières semaines de programmation nous rencontrons de nombreux problèmes concernant le wifi de myRIO. Le réseau sans fil est très instable, les coupures sont intempestives faisant perdre régulièrement le contrôle sur le robot depuis l'ordinateur, et la bande passante très limitée empêche une retransmission vidéo depuis la webcam embarquée. Nous nous sommes donc procuré chez STUDICA, parmi le matériel autorisé pour la compétition, un kit de communication sans fil amélioré. Il comprend un module d'extension pour le MyRIO permettant d'y ajouter un port Ethernet et 2 ports USB supplémentaires. Le second élément est un module OM5P-AC offrant un point d'accès sans fil double bande jusqu'à 1.17 Gbps. Il permet de communiquer avec le myRio via un réseau sécurisé, robuste et stable, ce qui est particulièrement important dans un environnement de compétition internationale.

Le module doit être programmé afin de pouvoir communiquer avec le MyRIO. Pour il est nécessaire de le brancher en Ethernet sur un ordinateur puis utiliser un logiciel fourni par Studica qui facilite sa mise en route. Cependant l'opération n'est pas si simple. En effet nous nous sommes rendu compte que le programme n'est pas compatible qu'avec un Windows anglophone. De plus la carte wifi et Bluetooth interfère la reconnaissance du OM5P-A par le logiciel. Il faut donc basculer le windows de l'ordinateur en anglais et désactiver ces 2 cartes dans la gestion des périphériques. Une fois le firmware installé, le numéro d'équipe et le nom du robot renseigné, il est prêt à être branché sur le robot et l'ordinateur peut être remis à l'état initial.

OM5P-A (Point d'accès sans fil blanc) et X-Hub (extension grise du myRIO)

b. Actionneurs

Le choix des actionneurs est un élément important dans la création d'un robot. Les moteurs du robot doivent être correctement dimensionnés afin de correspondre à l'utilisation du robot, à ses dimensions et à sa masse. De notre côté nous n'avons encore une fois pas le choix dans l'utilisation

des moteurs. Nous disposons de 4 moteurs 12V (nominal) avec un réducteur de 60:1 pouvant tourner à vide à 100rpm. Ils possèdent un très bon couple de 50 kg.cm permettant de ne pas être trop limité pour la masse maximale du robot. Comme énoncé précédemment seulement 3 moteurs seront finalement employés pour le déplacement du robot. Le 4ème est quant à lui utilisé pour régler la hauteur de notre préhenseur.

Nous disposons également 3 servomoteurs de 3 types différents fournis dans le kit :

- Un servomoteur asservi en position entre -135° et +135°. Il est utilisé pour basculer la webcam en position de lecture de code bar, ou en position prise de balle.
- Un servomoteur à fort couple pouvant être asservis en rotation continue ou en position jusqu'à 3 tour et demi. Ce dernier est surtout utilisé pour sa caractéristique de contrôle en position et non pour son couple. Associé à un capteur de distance il permet grâce à sa rotation de mesurer son environnement jusqu'à 360 degrés autours de lui avec un unique dispositif.
- Un servomoteur à rotation continue. Ce dernier est utilisé pour entraîner la fermeture et l'ouverture de pince.

c. Capteurs

Encore une fois nous n'avons pas eu le choix des capteurs à utiliser. Le kit Studica met à notre disposition :

- Un NavX, capteur inertiel et magnétique à 9 axes, il offre une haute précision, dans les angles de lacet, tangage et de roulis. Il possède un accéléromètre, un gyroscope et un détecteur de perturbations magnétiques. On l'utilise principalement pour réguler le lacet du robot.

Panneau de contrôle du NavX

- Deux capteurs Ultra Son, d'une portée de 2cm à 3 mètres, positionnés de part et d'autre du côté du robot. Ils permettent notamment d'effectuer les suivis de mur. Le fonctionnement est très simple : le capteur émet une onde sonore inaudible (ultrasons) et mesure le temps que met l'onde pour aller vers l'obstacle, puis se revenir vers lui. En connaissant la vitesse du son dans l'air (340,29 m/s) il est alors aisé de mesurer la distance par rapport à l'objet.

- Deux capteurs de distance Infrarouges, l'un à l'avant et le second à l'arrière associé à un servomoteur contrôlé en position, il a ainsi une vision à 360 degrés sur son environnement. Pour récupérer la distance du capteur SHARP, sa courbe caractéristique a été tracée afin de convertir la tension renvoyée en distance.

- Quatre capteurs de fin de course, on en utilise deux, l'un sur le bras pour initialiser la position basse du bras (faire le "zéro" du bras, c'est à dire reset l'encoder moteur), et l'autre sur la pince, étant donné que la pince est gérée par un servo commandé en vitesse, on ne peut pas choisir l'ouverture exacte de la pince. Le capteur de fin de course et des butées judicieusement bien placées, nous permettent de commander trois ouvertures de pince : "Pince fermé", "Pince middle", "Pince full". Sur chaque fin de course est installé un montage pull down pour la protection des entrées.

- Des encodeurs sont également présents sur chaque moteur, ils sont directement gérés par le FPGA. Ils permettent de connaître leur vitesse de rotation ou encore la distance parcourue par une roue. Pour trouver les bons paramètres de l'encodeur pour relever une distance parcourue, il est important lire la datasheet du moteur, afin de relever le nombre de tics de l'encodeur par rotation qu'il faut diviser par le coefficient réducteur du moteur. Pour nos moteurs nous obtenons 24*60 = 1440.

  L'équation devient : $Distance = \frac{nb\_tics}{1440} * \pi * rayon\_roue$

- Nous disposons également d'un capteur de ligne composé lui-même de 4 émetteur/récepteurs infrarouges.

d. Alimentation

Le robot est alimenté avec une batterie "Pitsco TETRIX" de 12V, 3000mAh de type NiMH (nickel-hydrure métallique). Elle possède un fusible à 20A pour assurer la protection de la batterie.

Dans un premier temps, nous branchions directement tous les divers éléments du robot directement sur la batterie. Le 5V des servomoteurs était quant à lui récupéré de la sortie 5V du MyRIO. Cependant ce dispositif engendre de nombreux problèmes : La sortie 5V du MyRIO ne peut délivrer que 500mA ce qui n'est pas suffisant pour alimenter correctement les servomoteurs. L'activité des moteurs génère également une chute de la tension ce qui engendre le reboot du MyRIO. Nous nous sommes donc procuré un régulateur de tension dans la boutique Studica, autorisée par la compétition, afin de séparer les différents éléments et résoudre ces désagréments.

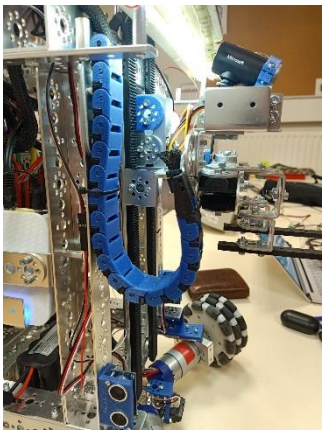
Ce régulateur propose 2 sorties 12V jusqu'à 2A que nous relions l'une à l'alimentation du myRIO et l'autre à celle du module wifi. En effet ces 2 éléments nécessitent une courant de 1.5A. Ces éléments étant désormais alimentés et régulés séparément, l'activité des moteurs n'engendre plus de dysfonctionnements. Les sorties 5V, 2A du régulateur servent alors à alimenter les

Régulateur de tension

servomoteurs via l'entrée 5V des MD2. Les moteurs restent tout de même directement reliés à la sortie de la batterie puisqu'ils peuvent nécessiter jusqu'à 8.7A chacun ce qui n'est pas supportable par le régulateur.

L'ajout d'un petit voltmètre LCD nous permet d'avoir un retour sur la tension de la batterie et de surveiller sa capacité et un potentiel rechargement nécessaire.

e. Câblage



La gestion du câblage est un point important tant en compétition qu'en prototypage. Il est nécessaire d'organiser les câbles de manière à identifier rapidement à quoi il est relié et n'avoir aucun fil qui traine, au risque de perdre des points. Pour le câblage, avons adopté un code couleur pour se repérer entre les tensions des fils d'alimentation, le noir est pour le GND, le rouge le 12V et le bleu le 5V. Du scotch de couleur à chaque extrémité des câbles nous permet de différencier les différents actionneurs/capteurs. Pour une connexion et déconnexion rapide et facile entre les fils lors de la phase de développement nous avons opté pour des connecteurs "Wago".

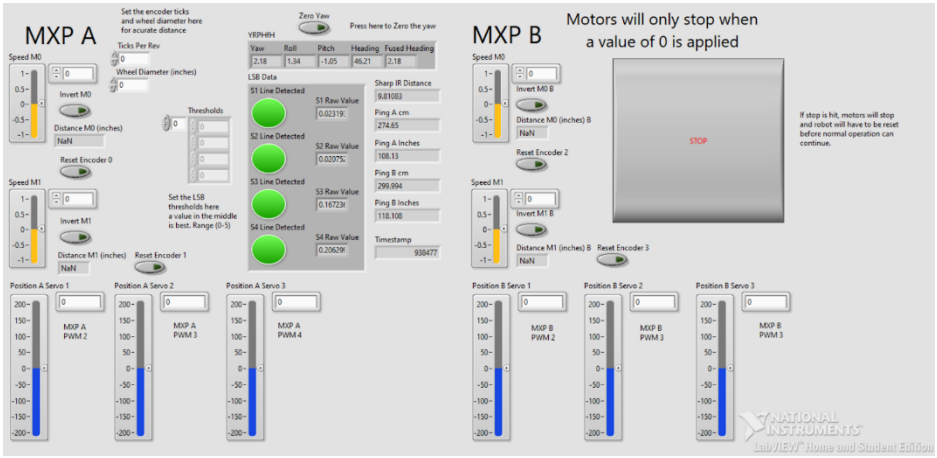
Tous les câbles sont attachés et regroupés dans des gaines passe câbles avant d'être redirigé vers les cartes MD2 ou d'alimentation. Pour les câbles du bras qui sont en mouvement, nous avons imprimé en 3D une chaîne passe câbles qui suit le mouvement du bras et donc ne les endommage pas.

Chaine passe câbles

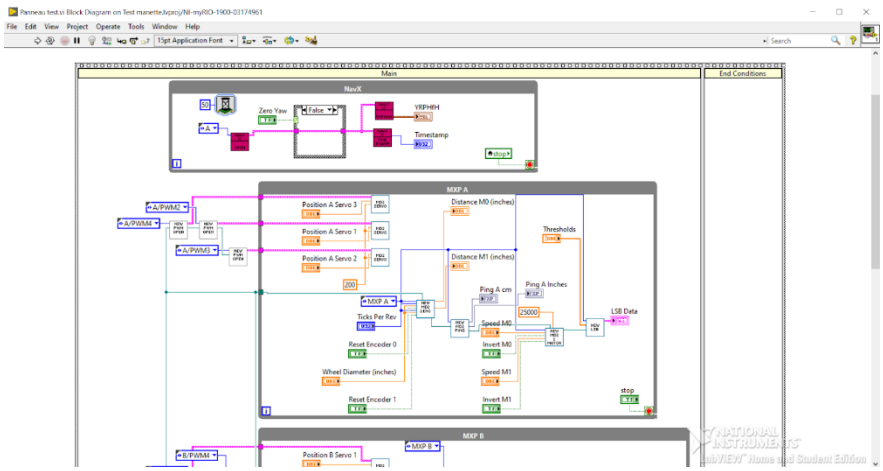
4. Programme LabVIEW

a. Logiciel

LabVIEW est un logiciel créé par National Instrument permettant la programmation de ses contrôleurs sous forme visuelle, avec des blocs à relier entre eux. Il est tout d'abord nécessaire de créer un projet qui regroupe l'ensemble des fonctions utilisées sur le robot et les caractéristiques de la carte électronique. Les programmes peuvent au choix être lancés sur l'ordinateur ou sur le myRIO en fonction de la situation. Chaque programme est appelé VI. Il peut prendre des paramètres en entrée et des sorties de tous les types usuels. Les VI sont divisés en 2 aspects : une partie programmation graphique où les fonctions sont dessinées, puis la partie panneau de contrôle. Sur ce panneau de contrôle peuvent être inclues des "Indicateurs" qui affichent graphiquement ou textuellement différentes valeurs du programme. Il existe également des "Contrôleurs" qui permettent de modifier en temps réel des variables du programme. Ces derniers peuvent être représentés sous la forme d'un bouton, d'une glissière, etc...



Exemple d'un visuel de panneau de contrôle permettant de tester les capteurs et les actionneurs

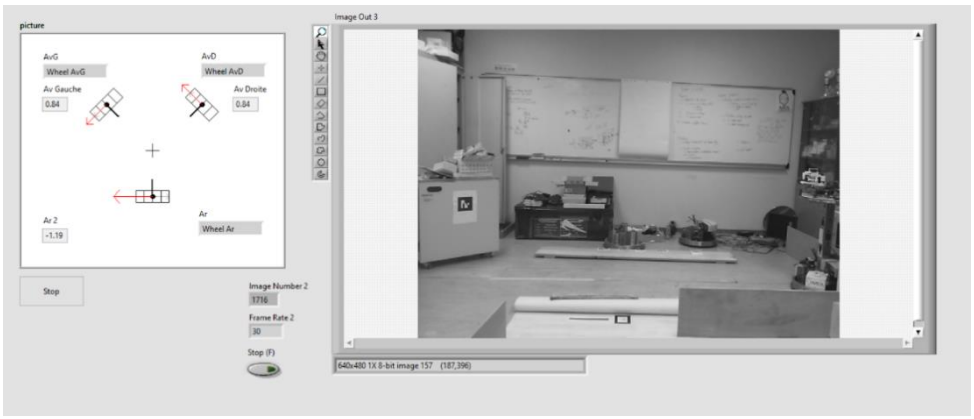


Morceau du programme correspondant

b. Télé-opération

La télé-opération du robot est notre première mission après la fabrication du châssis. Pour cela nous disposons d'une manette usb Logitech F310 homologuée par la compétition. Celle-ci est branchée directement sur l'ordinateur sur lequel tourne un VI. Ce VI utilise différentes fonctions de LabVIEW de la librairie "external devices" qui permet d'ouvrir un périphérique voulu, de l'associer à un clavier, une souris, ou dans notre cas une manette. Il interprète les messages envoyés par la manette, que nous récupérons et associons de manière plus explicite à des noms boutons. L'ensemble de la valeur des boutons et des joysticks est alors envoyé vers une variable partagée entre l'ordinateur et le myRIO. Ce dernier peut alors lire l'état de la manette pour effectuer les mouvements correspondants : aller en diagonale, baisser le bras, ouvrir, ouvrir la pince, etc..

Une retransmission vidéo est également installée afin de pouvoir contrôler le robot sans qu'il soit dans le champ de vision de l'utilisateur. Pour cela nous avons fait le choix de n'envoyer qu'une image en nuance de gris pour fluidifier au mieux le streaming et conserver une retransmission en 30 images par seconde constamment.



Retour vidéo et affichage des commande des moteurs

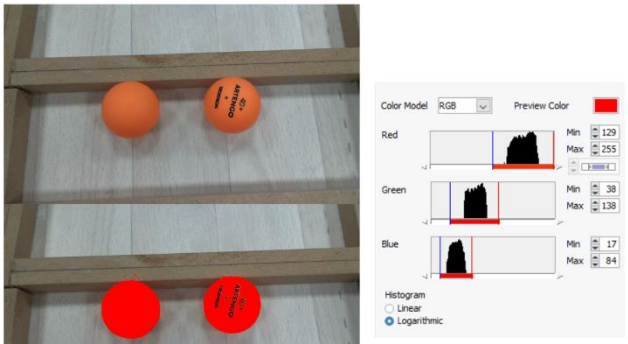
Fonctionnalités de la manette

c. Reconnaissance de balles

La saisie d'une balle dans les emplacements prévu à cet effet n'est pas une étape aussi facile qu'on pourrait le penser. En effet les balles ne se situent pas à un endroit précis mais dans une zone rectangulaire, il est donc impossible de les attraper par simple odométrie. Une reconnaissance et un suivi par caméra est donc indispensable. Afin d'identifier une balle sur une image plusieurs traitements d'images seront nécessaires.

- Seuil colorimétrique :

Les balles étant colorées et le sol étant de couleur clair, elles peuvent facilement se démarquer en analysant la valeur colorimétrique des différents pixels de l'image. La première étape pour localiser une balle orange est donc de binariser l'image en mettant à 1 les pixels correspondant à de l'orange et à 0 les autres. Il faut donc définir ce qu'est la couleur orange, pour ce faire il faut relever son code RGB. Cependant la luminosité ambiante peut varier, et les ombres sur la balle peuvent changer cette valeur. Il est donc important de ne pas prendre des valeurs discrètes mais plutôt des plages de valeurs couvrant l'ensemble des combinaisons RGB pouvant être originaire de l'élément à reconnaître. Sur l'image de calibration des seuils RGB ci-contre nous avons sélectionné une zone de l'image dans laquelle se trouve uniquement la balle et afficher l'histogramme de cette zone. Cela permet de régler facilement les seuils en prenant en compte les variations de couleur. Nous laissons tout de même une marge d'erreur dans nos réglages afin d'être insensible à des potentiels variations de luminosité ambiante. Comme vous pouvez le voir sur l'image binarisée les balles sont parfaitement identifiables.



Exemple d'un seuil colorimétrique

- Filtre de particules :

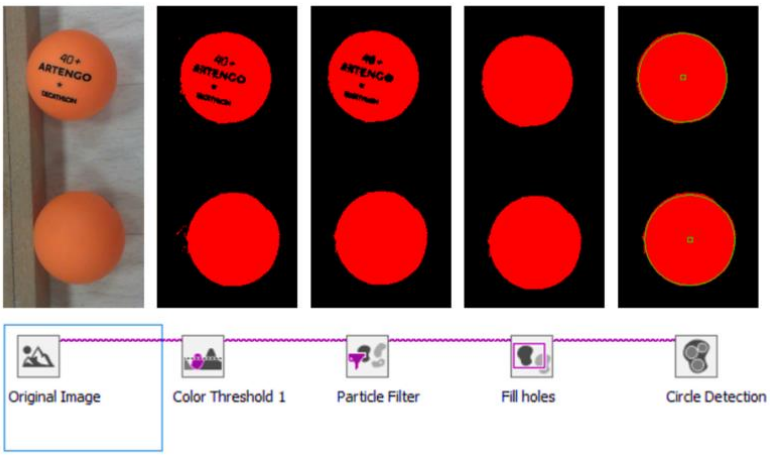
Le seuil colorimétrique n'est pas parfait, il peut alors apparaître dans des environnements peu propices ou peu lumineux des pixels éparpillés étant identifié comme étant de la couleur de la balle. Afin d'éliminer ce bruit nous appliquons un filtre qui identifie les formes dans l'image selon une connexité 4/8 et élimine toutes celles de petite surface.

- Remplissage des trous **:**

Comme nous pouvons le voir sur les images, les balles possèdent des écritures qui ne sont pas reconnues comme "orange". Cela crée des trous dans l'image binaires qui sont remplis grâce à cet algorithme afin d'uniformiser la zone.

- Reconnaissance de cercles :

Cette dernière fonction permet d'identifier des formes circulaires dans l'image, qui correspondent ici à nos balles. Elle renvoie le rayon et les coordonnées dans l'image du centre des cercles.



Processus de reconnaissance de balles

Une fois cette analyse faite, il est alors aisé de demander au robot de placer sa pince au-dessus de la balle. Nous enregistrons le point dans l'image où doit être idéalement la balle. Une petite fonction permet de sélectionner le cercle le plus proche de ce point et envoie la commande de mouvement du robot pour corriger sa position. Des régulateurs PID sont utilisés pour maintenir le robot parallèle à la "Bin" et atteindre la balle de manière fluide.

d. Déplacements du robot

Pour assurer le déplacement du robot, nous avons développé tout un panel de fonctions que nous réutilisons pour se rendre d'un point A à un point B. En effet, vu le temps imparti pour le projet nous commençons par définir des points de départs stratégiques sur la carte d'où le robot partira pour faire une action et reviendra. Nous pouvons ainsi programmer manuellement les mouvements du robot en fonction de son environnement en ayant toujours une position initiale constante.

Pour cela nous avons notamment programmé des fonctions d'odométrie avec les équations précédemment calculées, du suivi de mur de tous les côtés du robot, de rotation avec le navX et une fonction pour se positionner à une certaine distance d'un mur. L'ensemble des mouvements est régulé via des PID. Dans la plupart des fonctions est également inséré une régulation du lacet du robot afin qu'il garde son cap et ne dérive pas. Le navX se dérèglant au fil du temps, une fonction permet de recalibrer le compas par rapport à un mur en se positionnant parallèle à lui grâce aux capteurs.

La position des Workstations et des Component carriers étant toujours indiquée par une ligne noire au sol, nous avons également créé différentes fonctions permettant de suivre une ligne noire ou de se positionner le robot par rapport à elle. Ce sont des fonctions cruciales pour la dépose des balles et la préhension du lève palette.

Toute une série de fonctions permettent également de gérer l'ouverture de la pince, la calibration de l'encodeur du bras et la consigne de hauteur de bras.

Pour finir nous disposons d'un ensemble de routines regroupant un ensemble d'actions à effectuer. Ainsi avec un seul VI prenant en entrée quelques paramètres nous pouvons demander au robot de lire le code barre du 4ème component carrier puis revenir, ou encore de lui demander d'aller à la Workstation 5 sur le terrain. Il est alors très facile de structurer le programme principal avec ces fonctions.

e. Problèmes rencontrés

Pendant les 5 premières semaines toujours freiné par un problème qui nous empêchait d'utiliser à la fois la partie bras du robot et la partie déplacement. Il était donc impossible de tester toutes les fonctions ensemble et de faire des enchaînements simples.

Pour trouver la source du problème nous avons établi une routine de test avec un enchaînement de petites actions qui provoquaient des problèmes afin d'essayer de remonter dans l'architecture des VI (fonctions) et trouver le problème. Nous avons finalement trouvé un code d'erreur indiquant un problème d'allocation de mémoire. Avec l'aide de Vincent Coelen, nous avons exploré cette piste en affichant en temps réel l'utilisation de la mémoire du myRio. Nous avons alors remarqué qu'il ne possède que 71Mo de mémoire libre et qu'au fil des actions elle commençait à réduire jusqu'à saturation. En remontant jusqu'à la racine de la librairie de lecture des différentes entrées/sorties, nous avons découvert qu'à chaque appel d'un capteur/actionneur la fonction copie en mémoire le bitfile du FPGA afin de pouvoir communiquer avec. Cependant ce fichier n'est pas supprimé et avec une taille de 2.8Mo, il remplit rapidement la mémoire rendant impossible la lecture de nouveaux capteurs.

La solution proposée est donc de restructurer en partie cette librairie fournie par le constructeur. Nous avons donc refait les fonctions d'ouverture du FPGA, des DIO et des PWM en prenant en entrée directement le pointeur du bitfile et ainsi éviter de créer une copie en local. Une fois ces différentes fonctions de base refaites il faut actualiser l'ensemble de la librairie de STUDICA pour fonctionner avec et que toutes prennent également le bitfile en entrée pour également éviter les doublons internes. Nous disposons maintenant de notre version de la librairie avec une optimisation de l'utilisation de la mémoire. Cependant il faut également actualiser avec cette dernière toutes nos fonctions précédemment développées.

Nous profitons de cette restructuration pour améliorer l'ensemble de nos programmes. À la manière de ROS nous souhaitons séparer la partie haut niveau de la partie bas niveau du projet. Au lieu de faire appel à chaque capteur/actionneur dans nos fonctions, nous faisons fonctionner deux boucles en parallèle du programme principal. La première lit la valeur de l'ensemble des capteurs et la publie dans une variable partagée (sous la forme d'une classe) comme le ferait ROS avec les topics. La seconde permet, elle, de lire sur une autre variable partagée les commandes qui sont données au robot et de l'envoyer aux différents actionneurs. Le programme principal ne fait donc plus que des calculs et se contente de lire la valeur des capteurs et publier la commande correspondante.

Conclusion

Dans le cadre de notre entrainement pour la compétition Euroskills 2020, ce projet a été une aubaine pour nous familiariser au mieux avec le kit de robotique imposé et l'outil de programmation Labview. Au fil des semaine Nous avons appris à développer nos compétences en mécanique en imaginant un système de pince et de lève palette tout en respectant les contraintes du sujet des Worldskills 2019. Nous avons également programmé de nombreuses fonctions de déplacement du robot, et de traitement d'image réutilisables par la suite. Nous nous sommes heurtés à de nombreux problèmes que nous avons su analyser et résoudre et qui seront un gain de temps non négligeable pour la suite de nos entrainements.

Le robot est actuellement entièrement fonctionnel tout en respectant les objectifs fixés par le sujet. Nous consacrerons le temps restant avant la compétition à continuer d'améliorer nos algorithmes afin d'aboutir sur un robot robuste, rapide et complètement optimisé. Pour cela nous exploiterons des algorithmes de SLAM (simultaneous localization and mapping) ainsi que des filtres de Kalman pour améliorer le retour des capteurs.