Projet de robotique ROB3 - 2026

Vous êtes réparti.e.s en groupes de 6 élèves, votre but est de concevoir et fabriquer un robot capable de réaliser un exercice défini dans le présent cahier des charges.

1 Le robot

1.1 Description générale.

Vous devez construire un robot schématisé sur la figure 1 et composé de la façon suivante :

1. Un charriot mobile de type unicycle amené à se déplacer dans un plan horizontal (le sol d'une arène de jeu).
2. Un bras à 1 ddl fixé sur le charriot mobile portant une pince.
3. Une pince montée vers l'avant du robot sur le bras.

La pince doit être capable de saisir l'objet (décrit sur la figure 2) lorsque le bras est en position basse ; le bras doit pouvoir soulever la pince et l'objet et le redéposer sur le sol ; le charriot doit pouvoir se déplacer de manière contrôlée dans le plan.

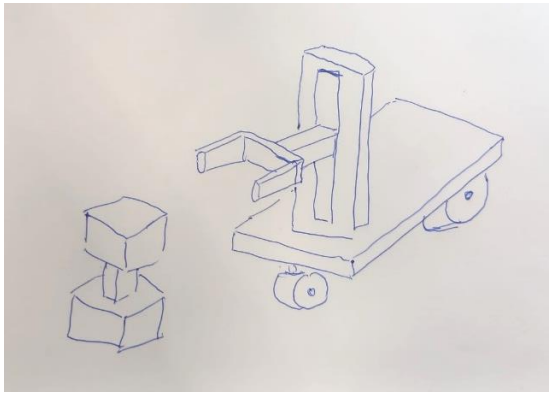


Figure 1 : allure générale du robot (la cinématique du bras n'est pas imposée)

1.2 Matériel à disposition pour la fabrication du robot et de son contrôleur

Un kit pour l'équipe composé principalement de :

1. Une pince 3551 fournie en kit, avec un servo moteur intégré.
2. Deux moteurs KTECH MS4015-V3 contrôlables en vitesse et intégrant une mesure de position (prévus pour les roues)
3. Un moteur Dynamixel commandable en position (prévu pour le bras)
4. Deux roues Pleine Guitel Hervieu Ø 50mm destinées à être motorisée et une roue pivotante Guitel Hervieu, Ø 50mm.
5. Un carte Arduino MEGA + un shield SEED pour la communication via un bus CAN avec les moteurs + un shield DYNAMIXEL
6. Deux capteurs de distance à ultrasons HC-SR04
7. Une batterie RS PRO 12V 1.2Ah. Un kit de connexion permettant d'alimenter l'Arduino avec la batterie.

Dans le FABLAB Sorbonne Université, sont également disponibles :

1. Des planches de médium, 300mmx600mm ép. 6mm ou 3mm, découpables au LASER dans le FABLAB.
2. Des petits composants électroniques et mécaniques standard.
3. Du filament pour les imprimantes 3D du FABLAB.

2 La tâche

La tâche consiste à déplacer un **objet** depuis sa configuration initiale vers la configuration finale, à l'intérieur d'une **arène**.

L'objet est constitué d'un de trois corps empilés selon un axe vertical :

1. En bas : un parallélépipède base carrée de 30 mm et de hauteur de 40 mm.
2. Au milieu : un parallélépipède de hauteur 30 mm et de section carrée de 10 mm de côté, par laquelle le robot doit le saisir.
3. En haut, un cube d'arête 30 mm.

L'assemblage est jointif, mesure au total 100 mm de hauteur ; les faces des trois solides sont perpendiculaires deux à deux et leurs trois centres sont alignés sur une droite verticale, cf. Figure 2. L'équipe qui souhaite en connaître son poids peut le peser.

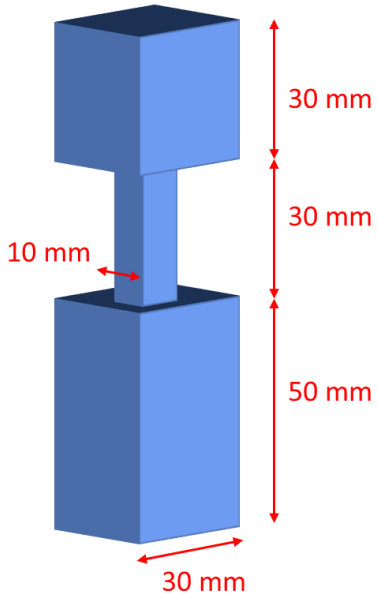


Figure 2 : objet à manipuler. Il doit être saisi avec la pince dans la partie amincie

L'arène est un plan horizontal bordé par des murets verticaux de 10 cm de hauteur formant un rectangle de 1,98 m x 1,22 m. L'arène est représentée sur la figure 3 par le rectangle bleu ABCD.

Au début du jeu, le robot est placé dans une configuration initiale. Dans cette configuration :

- Le côté opposé à la pince du robot (l'arrière) est en contact avec le côté AD ;
- Son essieu est parallèle au côté AD ;
- La distance a_0 entre le centre de l'essieu et le côté CD est inconnue, mais supérieure à 60 cm.

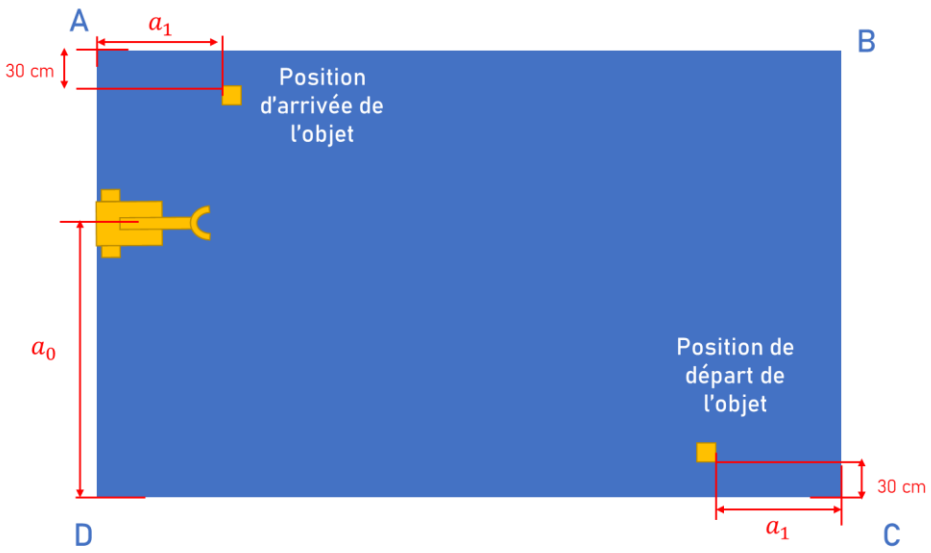


Figure 3 : arène avec le robot dans sa configuration de départ, l'objet dans ses configurations de départ (en vert) et d'arrivée (en orange), les obstacles fixes (violet).

L'objet est placé dans une position de départ avec ses faces parallèles aux côtés de l'arène et à une distance de 30 cm de CD et une distance a_1 (inconnue, inférieure à 50 cm) du côté BC.

Au top départ de l'arbitre, l'équipe actionne un bouton poussoir monté sur le robot. Le robot doit alors fonctionner en mode automatique (sans pilote ni aide humaine).

Il doit aller saisir le l'objet dans sa position de départ et l'emmener vers la position symétrique à la position de départ par rapport au centre du rectangle.

3 Premier test des éléments du KIT

Au premier jour du projet, l'un des membres de l'équipe doit installer le kit pour faire tourner le programme d'exemple, afin de vérifier le fonctionnement de tous les éléments du kit. Le kit assemblé ressemble à ça :

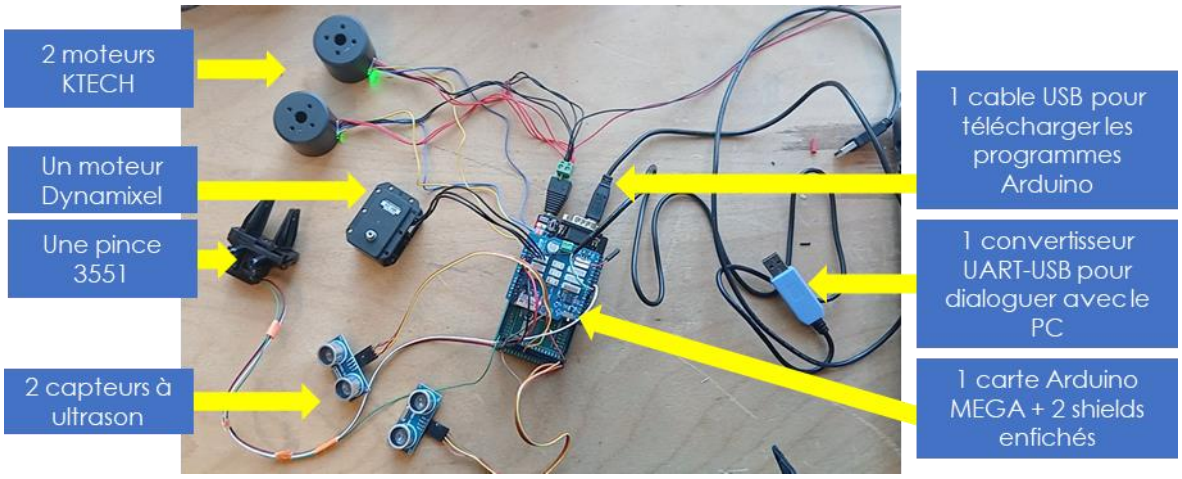


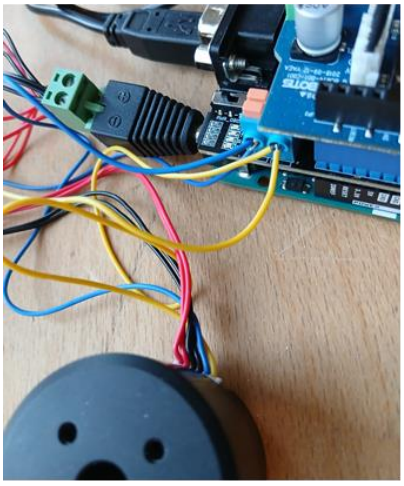
Figure 4 : Kit assemblé pour un premier test.

3.1 Branchements :

ATTENTION : à chaque fois que vous branchez / débranchez un fil, les alimentations doivent être coupées ; les câbles USB vers votre PC doivent être débranchés car ils véhiculent aussi l'alimentation de votre PC. N'alimentez que lorsque les branchements sont terminés

3.1.1 Enfichez le shield CAN sur l'Arduino MEGA

3.1.2 Reliez le bus CAN entre le shield CAN et les moteurs KTECH



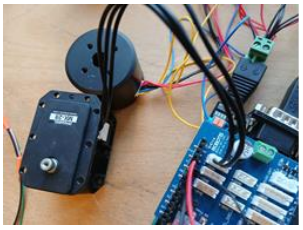
Les moteurs KTECH sont raccordés via un câble à 6 fils (2 rouges pour le +12V, 2 noirs pour le 0V, 1 bleu et un jaune pour la liaison CAN) fourni dans le kit. Les fils sont libres d'un côté et reliés à un connecteur blanc de l'autre, à raccorder au moteur.

- Reliez le fil bleu de chacun des moteurs KTECH sur le bornier bleu du shield CAN (contact le plus extérieur, à gauche sur la photo)
- Reliez le fil jaune de chacun des moteurs KTECH sur le bornier bleu du shield CAN (contact le plus intérieur, à droite sur la photo)

Attention : dénudez les câbles sur une longueur suffisante pour qu'ils touchent le fond du bornier une fois insérés (sinon cela se déconnecte souvent)

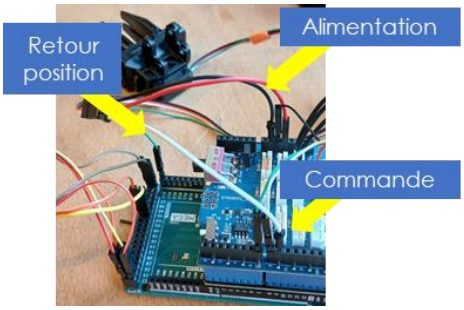
3.1.3 Enfichez le shield DYNAMIXEL sur le shield CAN

3.1.4 Branchez le moteur DYNAMIXEL



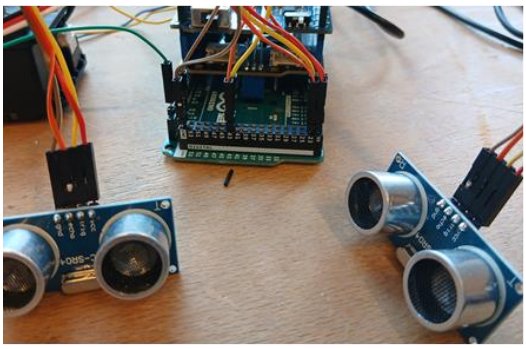
Il faut utiliser le câble noir à trois fils avec un connecteur blanc à chaque extrémité, entre le moteur et le shield Dynamixel (trois embases disponibles sur le shield). Ce câble transmet à la fois la puissance et le signal de commande.

3.1.5 Branchez la pince



- Alimentation + : Fil rouge sur le +5V de l'Arduino (accessible depuis le shield Dynamixel)
- Alimentation - : Fil noir sur le GND de l'Arduino (accessible depuis le shield Dynamixel)
- Retour position : Fil vert sur A15 de l'Arduino
- Commande servo : Fil blanc sur la sortie digitale PIN 9 de l'Arduino (accessible depuis le shield Dynamixel)

3.1.6 Branchez les capteurs à ultrason

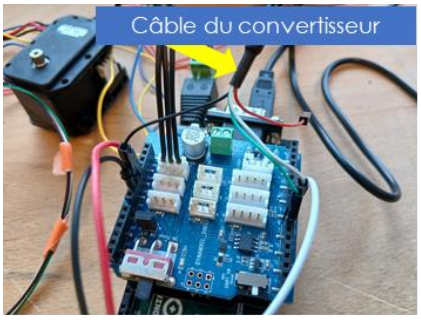


Sur le connecteur de l'Arduino MEGA contenant les entrées / sortie logiques 22 à 53.

- VCC du capteur sur l'alimentation +5V de l'Arduino MEGA (à droite sur la photo), ici un avec un fil rouge
- GND du capteur sur GND de l'Arduino (à gauche sur la photo) ici avec un fil marron.
- Capteur 1 : trig sur PIN 22 (ici jaune) et echo sur PIN 23 (ici orange)
- Capteur 2 : trig sur PIN 42 (ici jaune) et echo sur PIN 43 (ici orange)

3.1.7 Branchez le convertisseur UART-USB

Les convertisseurs UART-USB ont souvent 4 fils : un noir (la masse), un rouge (le +) et deux de couleurs variable selon le modèle (TX,RX). Parfois ils n'ont que 3 fils, le rouge est absent : ce n'est pas important pour vous dans la mesure où vous allez connecter le convertisseur sur votre PC dont le port USB est alimenté. **Il est donc inutile de brancher le fil rouge**.



- RX sur le PIN 7 du connecteur d'E/S numérique de l'Arduino MEGA (accessible depuis le shield Dynamixel, ici fil vert)
- TX sur le PIN 8 du connecteur d'E/S numérique de l'Arduino MEGA (accessible depuis le shield Dynamixel, ici fil blanc),
- 0V sur GND de l'Arduino (accessible depuis le shield Dynamixel, ici fil noir)
- Fil rouge non connecté

3.1.8 Branchez l'alimentation

Attention, branchez les fils sur le bornier à vis sans que celui-ci ne soit connecté à la carte. Une fois le bornier branché, enfichez la prise d'alimentation puis allumez l'alimentation.

Branchez sur le bornier à vis d'alimentation :

- Sur le +, le +12V (alimentation stabilisée ou batterie) et les fils rouges des moteurs KTECH (2 fils rouges par moteur)
- Sur le -, le 0V (alimentation stabilisée ou batterie) et fils noirs des moteurs KTECH (2 fils noirs par moteur)

Attention : le stator des moteurs KTECH doit être sur la table, sinon les fils vont s'entortiller quand vous allez mettre en marche !

3.2 Installation de Arduino IDE et programme example.ino

3.2.1 Fichiers fournis et bibliothèques

Est fourni un répertoire Arduino avec deux sous-répertoires :

4. **forMonitorOnly** qui contient un programme vide (forMonitorOnly.ino). Comme son nom l'indique, ce fichier sera ouvert avec l'Arduino IDE et ne sera ni

compilé, ni téléchargé. L'IDE sera connecté au port USB de votre ordinateur sur lequel vous aurez branché le convertisseur USB-UART ce qui vous permettra d'utiliser l'outil « moniteur » pour dialoguer avec l'Arduino par ce canal. A noter que vous ne pouvez pas dialoguer avec l'Arduino via son port USB classique (celui que vous utiliserez pour télécharger les programmes) à cause du shield Dynamixel qui parasite cette communication en pilotant le moteur dynamixel. Ainsi les commandes telles que Serial.println ne sont pas utilisables. Elles sont remplacées par PC_SERIAL.println qui envoie les caractères sur l'UART.

5. **example**, qui contient le programme example.ino destiné à tester le kit et à vous donner des exemples sur l'utilisation des divers éléments (envoyer une vitesse aux moteurs KTECH, lire un capteur ultrason, etc.). Le répertoire contient aussi deux fichiers (.h et .cpp) qui contiennent les fonctions pour la commande des moteurs.

Pour installer tout ça :

1. Installez sur votre ordinateur l'Arduino IDE (logiciel qui permet de compiler du code Arduino et de le télécharger sur les cartes Arduino). Il se trouve gratuitement sur internet.
2. Installez les deux bibliothèques qui sont nécessaires pour utiliser la communication CAN. Pour cela ouvrez Arduino IDE et cliquez, dans la colonne de gauche, sur l'icône représentant des livres :
    a. Cherchez la bibliothèque CAN_BUS_Shield et cliquez sur installer
    b. Puis cherchez la bibliothèque 107-Arduino-mcp2515 et cliquez sur installer

    Les bibliothèques sont installées dans
    Documents\Arduino\libraries\CAN_BUS_Shield et
    Documents\Arduino\libraries\107-Arduino-MCP2515.

    Ce n'est pas fini. **Vous devez ensuite éditer le fichier** :

    Documents\Arduino\libraries\CAN_BUS_Shield\src\mcp2515_can.h

    Pour changer la ligne :

    virtual byte begin(uint32_t speedset, const byte clockset = MCP_**16**MHz);

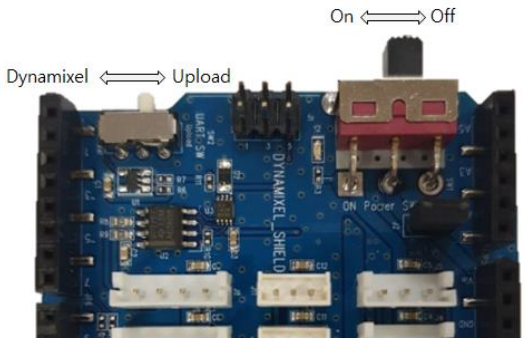
    Par :

    virtual byte begin(uint32_t speedset, const byte clockset = MCP_**8**MHz);

3. Installez les bibliothèques pour le moteur Dynamixel : « dynamixel shield » et « DYNAMIXEL2Arduino ».

Pour télécharger le fichier :

- Connectez de façon classique votre PC à l'Arduino à l'aide du câble USB.
- Placez les deux interrupteurs du Shield Dynamixel sur « **Upload** » (c'est dans cette position que vous pouvez télécharger) et « **Off** » (pour couper la puissance sur le DYnamixel), voir ci-dessous



- Quand le téléversement est terminé, placez les deux interrupteurs sur « Dynamixel » et « On ».
- Appuyez sur le bouton reset de l'Arduino.

Ça doit fonctionner en principe de la façon suivante :

- S'il n'y a pas d'obstacle devant les capteurs US, alors toutes les deux secondes :
  - La pince s'ouvre ou se ferme
  - Le moteur Dynamixel passe d'une position à une autre
  - Les deux moteurs KTECH changent de sens de rotation
- Si vous placez un obstacle plan et n'absorbant pas les ultrasons à environ 5 cm de l'un des deux capteurs :
  - Les moteurs KTECH s'arrêtent de tourner
  - **Ou** la pince et le moteur Dynamixel s'arrêtent de changer de position toutes les deux secondes.
- Si vous mettez l'obstacle devant l'autre capteur, cela arrête les deux moteurs qui ne s'étaient pas arrêtés avec l'essai sur le premier capteur.

Le programme example.ino fonctionne. Il peut vous servir d'exemple pour construire le code de votre projet, mais il n'utilise pas toutes les fonctions possibles des moteurs. Voyez les possibilités des bibilothèques dynamixel sur :

https://emanual.robotis.com/docs/en/parts/interface/dynamixel_shield/#dynamixel-shield-libraries

et les fonctions et variables disponibles dans la classe des moteurs Ktech dans Ktech_motor.h :

```
class Motor
{
  public:
    Motor(int motorID);
    void readState(mcp2515_can CANCom);
    void ON(mcp2515_can CANCom);
    void OFF(mcp2515_can CANCom);
    void sendVelocityCommand(long int vel, mcp2515_can CANCom);
    float posInDeg;
    float velInDegPerSec;
    float prevPosInDeg;
    int posEncoder;
    int offsetEncoder;
```

```
    int revolNumber;
  private:
    int _motorID;
};
```

Notez que :

- L'identifiant des moteurs Ktech doit être vérifié avec les enseignants au début du projet (dans le code example.ino, c'est 1 et 2 par défaut, mais les moteurs ont pu être configurés différemment)
- A chaque fois que vous appelez la fonction `sendVelocityCommand`, une lecture de l'état du moteur est réalisée et affecte les variables : `posInDeg`, `velInDegPerSec`, `prevPosInDeg`, `posEncoder`, `offsetEncoder` et `revolNumber` qui contiennent respectivement, la position en degrés, la vitesse en degrés par seconde, la position en point codeur, l'offset codeur et le nombre de tours (signé) effectué depuis l'initialisation. Il vous suffit donc d'accéder à se variables pour connaître leur valeur lors de la dernière commande.