# Documentation sur la librairie dynamique RobotDriver.dll

## Informations techniques :

La librairie a été programmée en langage C# sur le framework .NET 4. Elle est programmée pour un environnement windows et n’a pu être testée que sous cet environnement.

## Prérequis :

Le robot à piloter doit au préalable être connecté via une connexion USB à l’ordinateur où sera exécuté le programme de pilotage.

## Fonctionnement de RobotDriver :

Pour fonctionner correctement, RobotDriver doit effectuer les opérations suivantes dans l’ordre :

1. Récupération du port de communication avec le robot = port COM sur lequel est branché le robot
2. Récupération des origines sur les deux axes en lançant une procédure de prise d’origine
3. Récupération des informations du robot via un appel à la fonction updateAxis dès que des données sont reçues sur le port série
4. Optionnel : affectation des vitesses de déplacement des axes (une vitesse par défaut est sélectionnée si ces données ne sont pas renseignées)
5. Optionnel : affectation des longueurs d’axes (une valeur par défaut est sélectionnée si ces données ne sont pas renseignées)
6. Optionnel : engagement ou désengagement du frein de l’axe Z (permet de libérer le mouvement de l’axe Z pour un déplacement à la main)
7. Optionnel : arrêt de tout mouvement du robot

## Description des classes de la librairie RobotDriver :

* Classe ROBOT
  + Attribut Public axisY : classe objet Axe – instance de classe axe
  + Attribut Public axisZ : classe objet Axe – instance de classe axe
  + Attribut (propriété) Private port (Public Port set) : SerialPort (port COM) – port utilisé par le robot
  + Méthode Public void setSpeed (float speed) : Function
  + Méthode Public void setStroke (float stroke) : Function
  + Méthode Public void goToPosition (float y, float z) : Function
  + Méthode Public Boolean initAxisOrigins () : Function
  + Méthode Public stopMotion () : Function
  + Méthode Public switchBrake (int brakeState) : Function
  + Méthode Public void updateAxis (string datas) : Function
* Classe AXE
  + Attribut (propriété) Internal name (Public Name get/set) : String – nom de l’axe
  + Attribut (propriété) Internal speed (Public Speed get) : Float – vitesse configurée sur l’axe (en mm/s)
  + Attribut (propriété) Internal position (Public Position get) : Float – position actuelle de l’axe (en mm)
  + Attribut (propriété) Internal stroke (get) : Float – course configurée sur l’axe (en mm)
  + Attribut (propriété) Internal status (Public Status get) : Integer – indicateur d’utilisation de l’axe (1 si en mouvement, 0 si à l’arrêt)
  + Attribut (propriété) Internal limitStart (Public LimitStart get) : Integer – indicateur sur le capteur limite d’origine de l’axe (0 ou 1)
  + Attribut (propriété) Internal limitEnd (Public LimitEnd get) : Integer – indicateur sur le capteur de fin de course de l’axe (0 ou 1)
  + Attribut (propriété) Internal encoder (Public Encoder get) : Integer – position du codeur sur l’axe
  + Attribut (propriété) Internal origin (Public Origin get) : Integer – indicateur de prise d’origine de l’axe (0 ou 1)
  + Attribut (propriété) Internal brake (Public Brake get) : Integer – indicateur d’utilisation du frein (0 ou 1)

## Documentation des méthodes de la classe :

* **Méthode Public Void setSpeed (float speed) :** permet d’envoyer la consigne de vitesse en mètres par seconde au robot. Cette consigne sera appliquée sur l’ensemble des axes.
* **Méthode Public Void setStroke (float strokeY, float strokeZ) :** permet d’ajuster la longueur des axes si nécessaire. Le robot considérera alors comme limite 2 cette valeur si le capteur de fin de course n’est pas actionné avant.
* **Méthode Public Void goToPosition (float y) :** permet d’envoyer au robot une consigne de déplacement en mm sur le premier axe uniquement. Tant qu’une limite ou que la position demandée ne sont pas atteintes, l’axe voit sa valeur status passée à 2 pour signifier qu’il est en mouvement.
* **Méthode Public Void goToPosition (float y, float z) :** permet d’envoyer au robot une consigne de déplacement en mm sur le premier et le deuxième axe. Tant qu’une limite ou que la position demandée ne sont pas atteintes, chaque axe voit sa valeur status passée à 2 pour signifier qu’il est en mouvement.
* **Méthode Public Boolean initAxisOrigin :** permet d’envoyer au robot une consigne pour prendre les origines des axes. La fonction demande d’abords à l’utilisateur si la zone est dégagée. Chaque axe voit sa valeur status passée à 2 pour signifier qu’il est en mouvement, et à 1 une fois le mouvement terminé et l’origine prise.
* **Méthode Public Void stopMotion :** permet d’envoyer au robot une commande d’arrêt de mouvement.
* **Méthode Public Void switchBrake (int brakeState) :** permet d’envoyer au robot une commande pour engager/relâcher le frein. Engage le frein de l’axe Z lorsque brakeState vaut 1, le relâche lorsqu’il vaut 0.
* **Méthode Public Void updateAxis (string datas) :** traite les informations renvoyées par le robot afin de renseigner la classe Axe de chaque axe.