TP1 : Présentation du robot Dobot et de l'application DobotStudio

Jean-Louis Boimond
Université Angers

Objectifs des 4 séances de TP sur le robot Dobot :

- TP 1 : présentation du robot et de DobotStudio (opérations élémentaires, instructions de mouvement, environnements : Teaching&Playback, Blockly),
- TP 2 : calcul du modèle géométrique direct, représentation de l'espace d'atteignabilité et simulation du bras du robot,
- TP 3 : calcul du modèle géométrique inverse, traçage d'une droite et dessin d'une figure dans l'espace articulaire.
- TP 4 : utilisation d'une caméra pour positionner le robot successivement aux centres de cercles disposés sur une feuille.

Table des matières

Le TP1 permet une initiation au robot Dobot à travers une présentation de ses caractéristiques et de l'application DobotStudio associée au robot. Deux environnements de programmation (« Teachning & Playback », « Blockly ») à même de réaliser des trajectoires du bras du robot, ainsi que des tâches en certains points, sont présentés.

I. Introduction

Dobot Magician est un bras robotisé, à visée pédagogique, muni de différents outils et accompagné d'un environnement de programmation à l'image des robots industriels. Le bras du robot peut se contrôler *via* l'application DobotStudio en proposant différents environnements de programmation tels que « Teachning & playback », « Blockly », « Script » ou en utilisant des langages plus standards comme Python, MatLab, C#. Muni de l'outil adéquat, l'application DobotStudio permet la réalisation de certaines tâches telles que : écrire/dessiner, déplacer une pièce, imprimer en 3D.

Le bras du robot comporte :

- différents corps : un socle (*base*), un bras (*rear arm*), un avant-bras (*forearm*) et un support permettant de fixer un outil (*end-effector*),
- et 5 articulations,

voir la figure suivante dans laquelle l'outil est une ventouse.

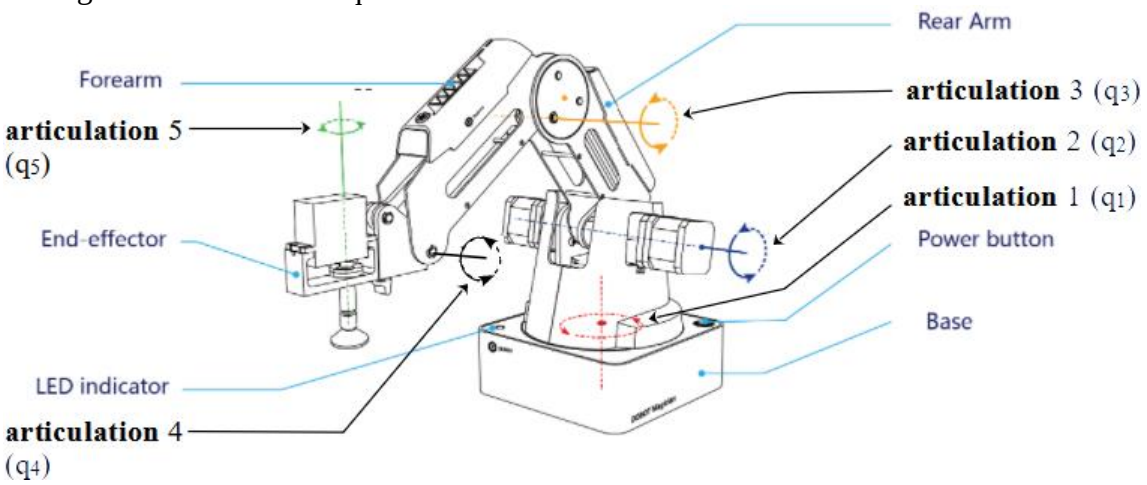

Figure 1 : Description du bras du robot muni d'une ventouse.

Localiser ces différents éléments sur le robot Dobot.

Dans le cas où l'outil est une ventouse (SuctionCup) ou une pince (Gripper), un servo-moteur est associé à l'outil pour activer l'articulation 5, par exemple, afin de maintenir une orientation constante de l'outil. Cette articulation est inactive (la valeur angulaire est nulle) lorsque l'outil est un Feutre (Pen).

L'articulation 4 n'est pas contrôlable. Sa valeur est fixée mécaniquement en fonction des valeurs angulaires appliquées aux articulations 2 et 3, ceci afin que l'axe de l'articulation 5 soit toujours orthogonale au plan sur lequel repose la base du robot.

Fixer le Feutre (sans retirer le capuchon) à l'extrémité du bras du robot, vérifier sur le robot Dobot que le Feutre est parallèle à l'axe $\overrightarrow{O_0 z_0}$ du robot, ceci quelle que soit la posture considérée.

Caractéristiques techniques :

Charge maximale	500 g
Atteignabilité maximale	320 mm
Latitude de l'articulation $q_1 (= J_1)$	$[-90°; +90°]$
Latitude de l'articulation $q_2 (= J_2)$	$[0°; +85°]$
Latitude de l'articulation $q_3 (= J_3)$	$[-10°; +85°]$
Latitude de l'articulation $q_5 (= J_4)$	$[-135°; +135°]$

Vitesse maximale (avec une charge de 250g)	Vitesse de rotation des articulations 1, 2, 3, 4 : 320°/s, vitesse de rotation de l'articulation 5 : 480°/s
Répétabilité	0.2 mm

La figure qui suit précise les dimensions des corps du bras, ainsi que du Feutre.

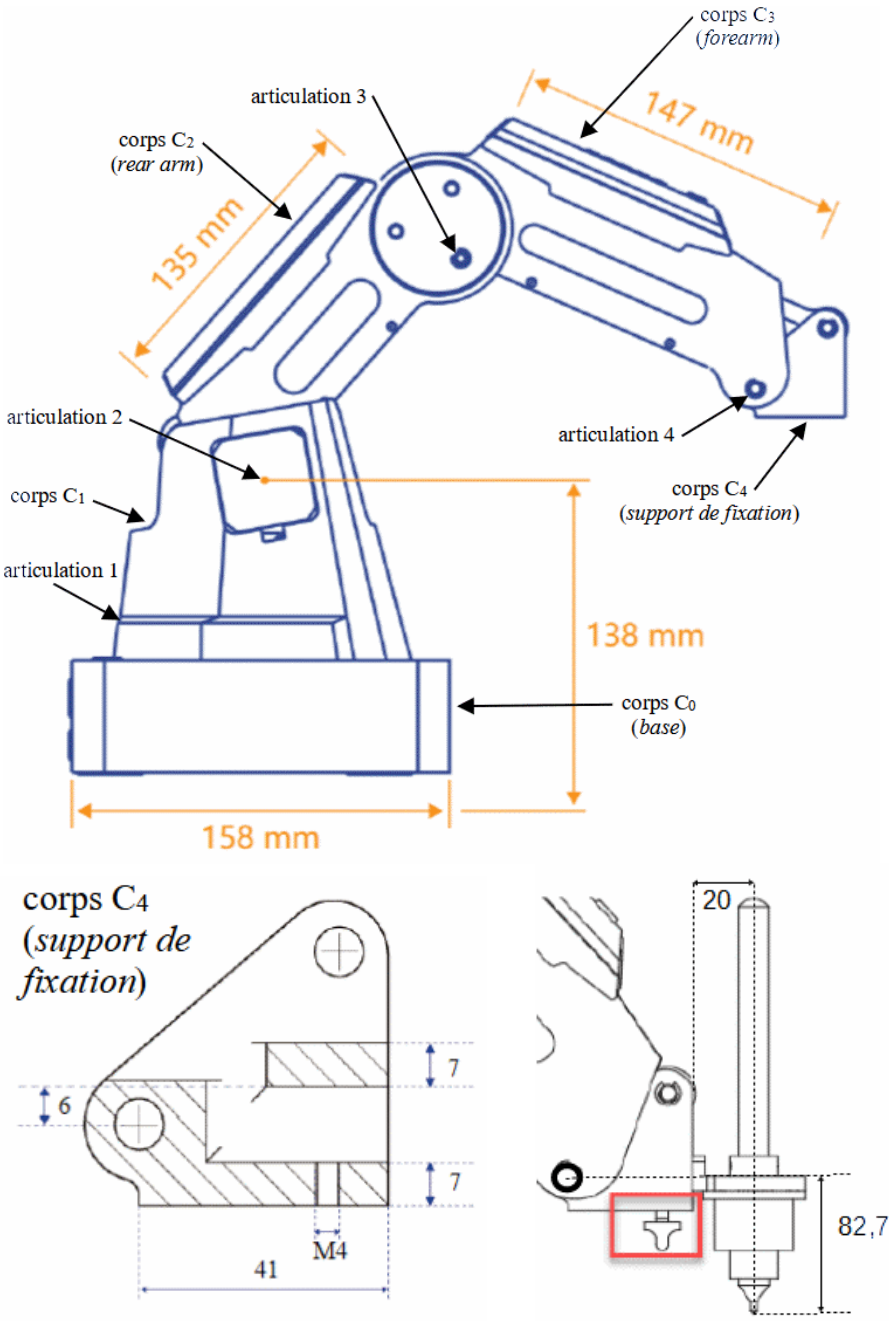

Figure 2 : Dimensions des corps du bras du robot, du support de fixation des outils, du Feutre.

Le repère de référence (*World*), x_0, y_0, z_0, ainsi que le repère outil (*Tool*), x, y, z, sont représentés dans la figure suivante.

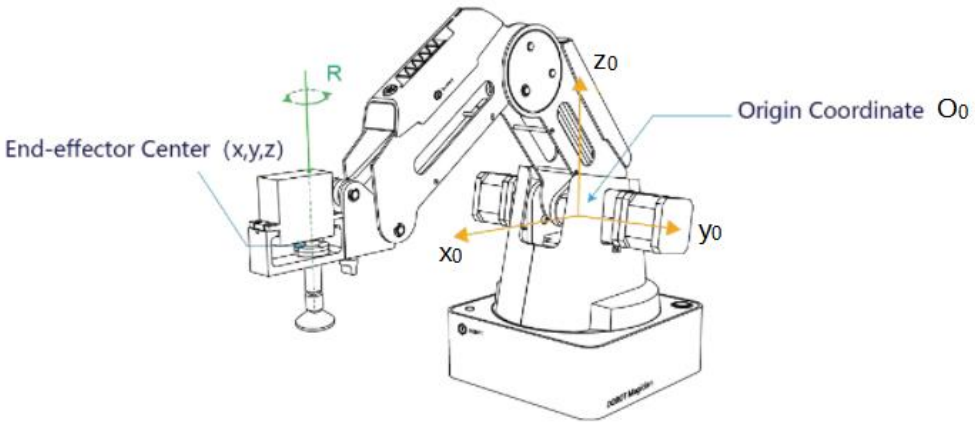

Figure 3 : Espace opérationnel, repères *World* et *Tool*.

La figure suivante représente le bras du robot, muni du Feutre, dans sa configuration initiale avec les repères associés aux différents corps constituant le bras. Le repère *World* est représenté par le repère R_0, le repère *Tool* correspond au repère R_5. Le point $O_5\ (= PF)$ représente l'extrémité du Feutre, c'est-à-dire, sa pointe.

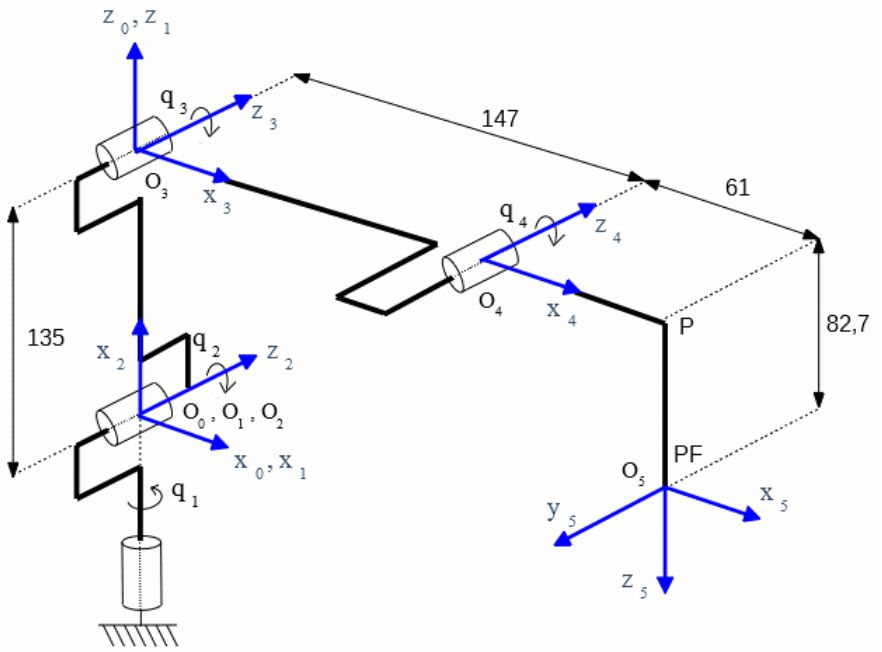

Figure 4 : Association des repères $R_0, \dots, R_5$ aux différents corps du bras du robot.

Question 1 : Situer les différents corps du robot, ainsi que le Feutre, dans la Figure 4. Déduire de la Figure les coordonnées des points O_3, O_4, P, PF dans le repère *World* $(= R_0)$ lorsque le robot est dans sa configuration initiale. Localiser ces points sur le bras du robot Dobot.

En référence à la méthode de Denavit-Hartenberg que l'on appliquera plus tard, on définit la valeur angulaire $\theta_j(t)$ de l'articulation j comme étant la valeur à l'instant t de l'angle de rotation entre les axes $\overrightarrow{O_{j-1}x_{j-1}}$ et $\overrightarrow{O_jx_j}$ autour de l'axe $\overrightarrow{O_jz_j}$.

Question 2 : Exprimer les valeurs $\theta_1(0), \dots, \theta_5(0)$ correspondant à la configuration initiale du bras du robot (voir la Figure 4).

On pose : $\theta_j(t) = q_j(t) + \theta_j(0), \forall t$, ce qui permet d'avoir $q_j(0) = 0, \forall j$ lorsque la configuration est initiale.
La valeur angulaire de l'articulation 3, c'est-à-dire, $\theta_3(t)$, est en fait égale à $q_3(t) - q_2(t) + \theta_3(0)$, autrement dit, la valeur de cette articulation dépend de $q_3(t)$ et de $\theta_3(0)$, mais également (de manière négative) de la valeur $q_2(t)$ (appliquée à l'articulation 2).
La valeur (non contrôlée) $\theta_4(t)$ de l'articulation 4 est telle que le support de fixation des outils du robot est, à tout instant, positionné à l'horizontal (comme illustré dans les Figures 1, 2 et 3), soit :

$$\overrightarrow{O_4x_4} = \overrightarrow{O_1x_1} \; ; \; \overrightarrow{O_4z_4} = \overrightarrow{O_1y_1}$$ (voir la Figure 4).

Ainsi, le Feutre sera toujours situé à la verticale (selon l'axe $\overrightarrow{O_5z_5}$) étant toujours orthogonal au plan $\left(O_0, \overrightarrow{O_0x_0}, \overrightarrow{O_0y_0}\right)$.

Question 3 : Montrer que $\theta_4(t) = -q_3(t)$.

II. Mise en route/arrêt du robot

- **Mise en route du robot :** positionner le bras et l'avant-bras du robot de manière à obtenir un angle égal à environ 45° entre eux, puis appuyer sur le bouton '**power**' situé sur la base du robot. Après environ 7 sec, un « beep » est émis et la LED, située sur la base (voir la Figure 1), passe du jaune au vert signalant que le bras est opérationnel.

Note : La LED passe au rouge lorsque le bras du robot atteint une position limite. Appuyez (continument) sur le bouton « unlock », situé sur l'avant-bras du robot (représenté par un cadenas ouvert), pour déplacer le bras dans une position (atteignable) souhaitée ; une fois le bouton relâché, la LED passe au vert.

- **Arrêt du robot :** appuyer sur le bouton « **power** » situé sur la base du robot ; l'avant-bras se replie alors (lentement) vers le bras du robot.

III. Application DobotStudio

Installation et lancement de la version 1.9.4 de DobotStudio :

- Copier sur votre bureau le fichier « DobotStudioSetup_V_1_9_4.zip » accessible *via* le lien DobotStudioSetup_V_1_9_4.zip,
- Une fois le fichier dézippé, double-cliquer sur le fichier .exe pour installer l'application, durant cette phase, accepter la demande d'installation du « Device driver »,
- Il vous reste à double-cliquer, soit sur l'icône « DobotStudio » située sur le bureau, soit sur le fichier « DobotStudio.exe », situé en principe dans le répertoire « C:\DobotStudio », pour lancer DobotStudio.

Notation : Les valeurs (en degré) des articulations 1, 2, 3 et 5 sont notées dans DobotStudio « Joint1, Joint2, Joint3 et Joint4 (plutôt que Joint5) », voir l'encadré situé à droite de la Figure qui suit. Elles sont telles que :

$$\text{Joint1}(t) = q_1(t), \text{Joint2}(t) = q_2(t), \text{Joint3}(t) = q_3(t), \text{Joint4}(t) = q_5(t).$$

A. Introduction

L'application DobotStudio permet de contrôler le bras du robot à travers plusieurs environnements de programmation tels que : « Teaching & Playback » ; « Blockly », voir la figure et le tableau suivants.

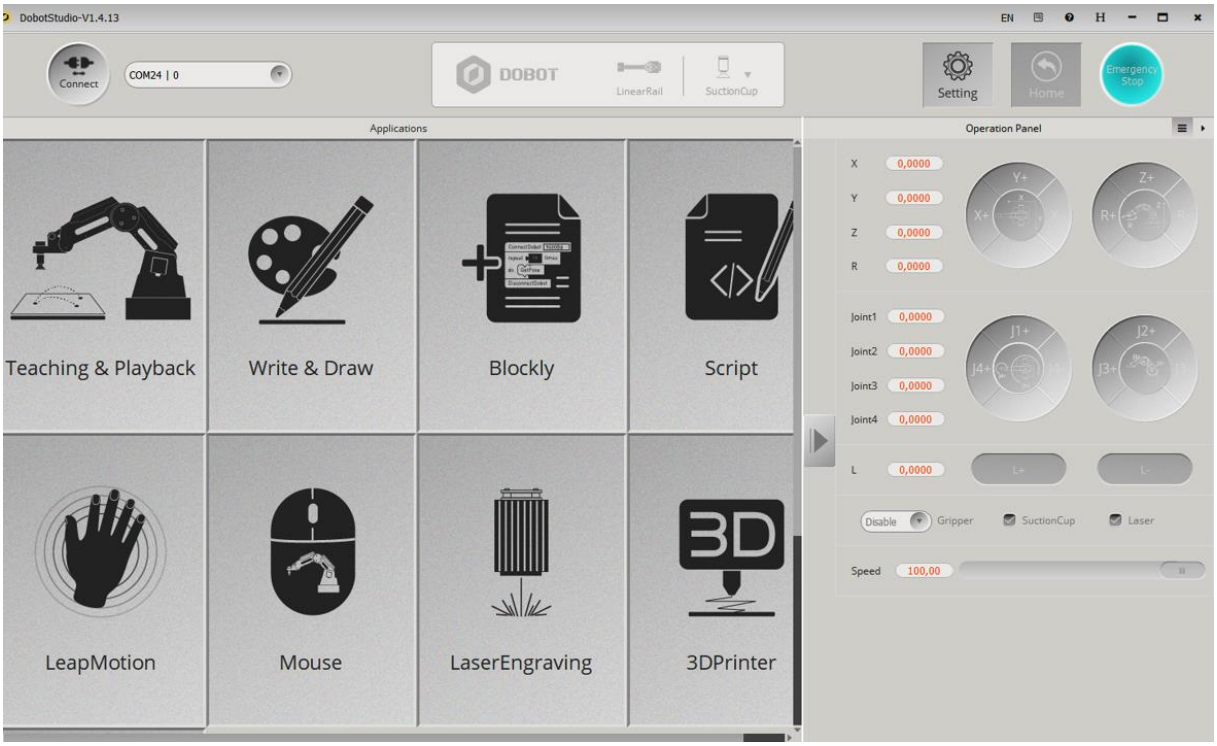

Figure 5 : Fenêtre principale de l'application DobotStudio.

Teaching & Playback	Permet d'associer de manière simple la phase d'apprentissage/d'acquisition de points aux instructions de mouvement.
Blockly	Permet à travers une approche graphique la génération de trajectoires et de tâches en certains points, par *Drag and Drop* de blocs (les blocs étant représentés sous la forme de puzzle).

B. Connexion/déconnexion du robot à DobotStudio

Cliquez sur le bouton « **Connect** » situé en haut à gauche dans la fenêtre de l'application. Le robot est connecté à l'application DobotStudio lorsque le bouton « Connect » se transforme en « Disconnect », voir la figure suivante. Il est déconnecté lorsque le bouton « **Disconnect** » se transforme en « Connect ».

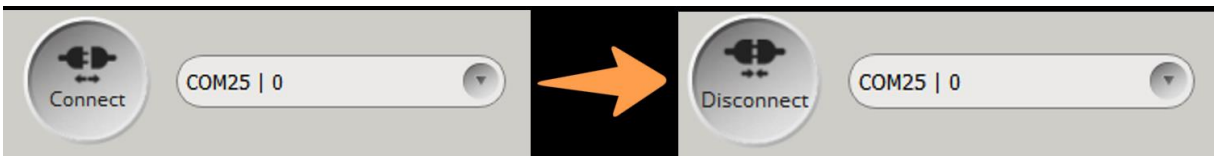

Figure 6 : Connexion à l'application DobotStudio.

Notez qu'il est possible de connecter le robot en mode « rapide » si la précision des mouvements n'est pas requise. Une boîte de dialogue apparaît à ce propos lors de la phase de connexion, voir la figure qui suit.

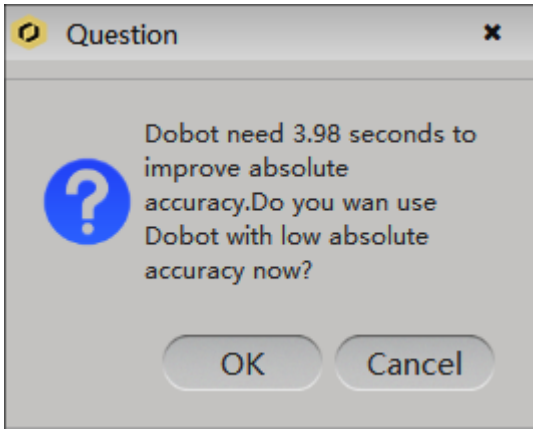

Figure 7 : Boîte de dialogue pour une connexion rapide (OK) ou non (Cancel).

C. Opérations élémentaires

Le menu déroulant, positionné au centre de la figure suivante, est situé dans l'encadré localisé en haut à droite dans la fenêtre principale de l'application. Il permet de sélectionner le type d'outil (« **SuctionCup** » (ventouse) dans la figure) fixé à l'extrémité du bras du robot. Les outils possibles sont : « **SuctionCup**, **Gripper**, **Laser**, **Pen**, **Advanced** », « Advanced » permettant de caractériser soi même un outil à travers des coordonnées, notées *xBias, yBias, zBias*.

Figure 8 : Choix de l'outil, opérations « **Setting** », « **Home** », « **Emergency Stop** ».

Dans cet encadré (voir figure précédente), trois opérations élémentaires : « Setting », « Home », « Emergency Stop », sont également accessibles ; le tableau qui suit les décrit brièvement.

Setting	Permet notamment de fixer des paramètres de base, voir le tableau 5.2 du User Guide (situé p. 32-33) pour les détails.
Home	Permet au bras de revenir dans sa position initiale pour obtenir une position correcte de référence. Réaliser cette opération avant une opération de mouvement nécessitant d'être précis, suite à un blocage du mouvement dû à un obstacle ou à un contact du bras avec une de ses butées.
Emergency Stop	Permet de stopper le bras du robot en cas d'urgence.

D. Contrôle manuel du robot

L'encadré « Operation Panel », situé à droite dans la fenêtre principale de l'application (voir la figure qui suit), permet :

- de déplacer manuellement (de manière non automatique) le bras du robot en considérant l'espace articulaire $(\text{Joint1}, \text{Joint2}, \text{Joint3}, \text{Joint4})$ ou opérationnel (X, Y, Z, R),
- de contrôler un outil de type Pince (Gripper), Ventouse (SuctionCup) ou Laser.

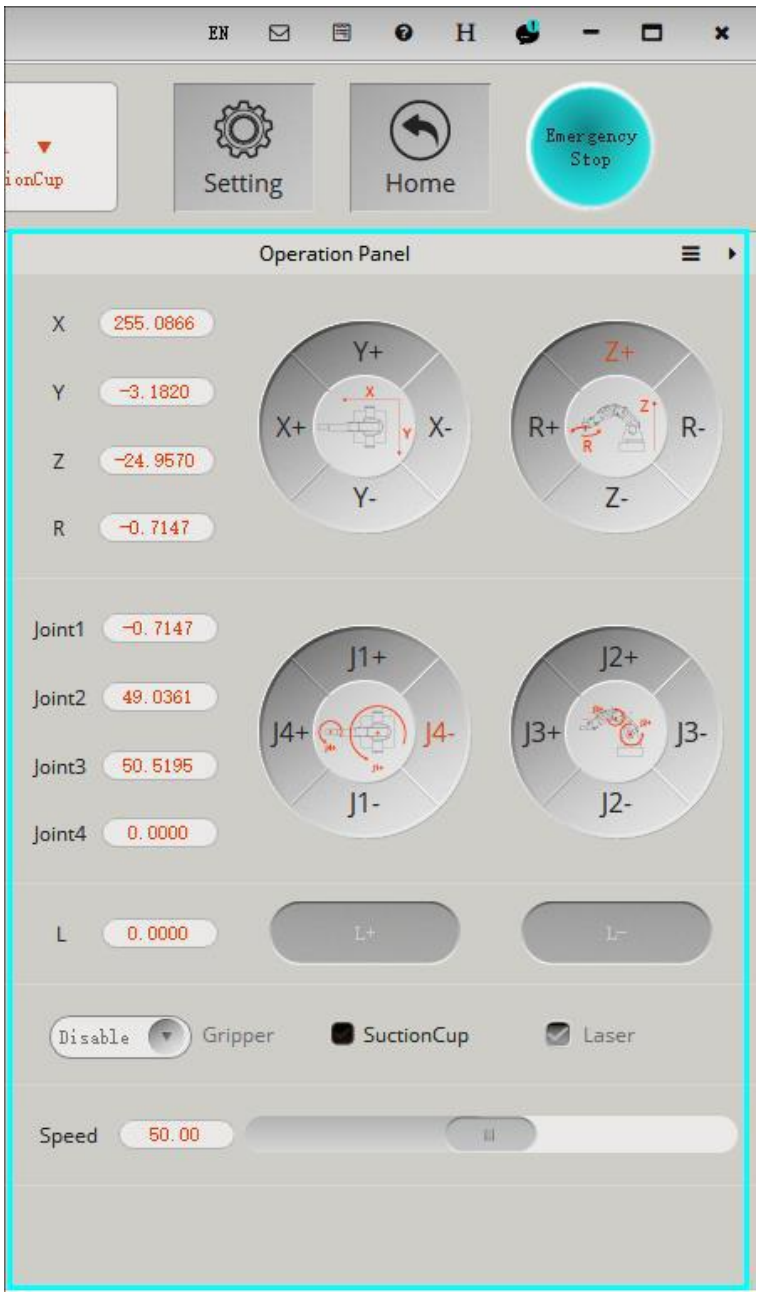

Figure 9 : Encadré « Operation Panel » pour un contrôle manuel du bras du robot et de certains outils.

Note : il est possible de ne pas afficher le contrôle du rail (*Linear rail control*) en sélectionnant le sigle ≡ situé en haut à droite de l'encadré « Operation Panel » afin de décocher la case « Linear rail control ».

1. Mouvement dans l'espace articulaire

La mise en mouvement du bras se fait en cliquant sur ***J*1+/-**, ***J*2+/-**, ***J*3+/-**, ***J*4+/-**, voir la figure qui suit.

Figure 10 : Commande des articulations *J*1, *J*2, *J*3, *J*4 dans le sens + ou -.

Le mouvement se fait axe par axe. Les valeurs angulaires $J1, J2, J3$ se rapportent respectivement aux articulations 1, 2, 3. La valeur $J4$ se rapporte à l'articulation 5 (activée par un servo-moteur fixé à l'outil lorsque celui-ci est une Pince, une Ventouse ou un Laser).

A titre d'exemple, l'articulation 1 évolue dans le sens positif (*i.e.*, dans le sens inverse des aiguilles d'une montre (*counterclockwise*)) en cliquant sur le bouton *J*1+ ; il évolue dans le sens négatif en cliquant sur le bouton *J*1-.

Vérifier que le fait de mettre en mouvement l'articulation 2 met en mouvement l'articulation 3, comme cela est mentionné dans la partie « I. Introduction, p. 5 ».

2. Mouvement dans l'espace opérationnel

La mise en mouvement du bras se fait en cliquant sur ***X*+/-**, ***Y*+/-**, ***Z*+/-** (*X*, *Y*, *Z* représentant les axes du repère de base du robot, *cf.* Figure 3), ou l'axe ***R*+/-** (*R* correspondant à l'angle entre les axes $\overrightarrow{x_0}$ et $\overrightarrow{x_5}$ autour de l'axe $\overrightarrow{z_0}$, c'est-à-dire, $J_1 + J_4$ où J_1 est l'angle entre les axes $\overrightarrow{x_0}$ et $\overrightarrow{x_1}$ et J_4 est l'angle entre les axes $\overrightarrow{x_1} = (\overrightarrow{x_4})$ et $\overrightarrow{x_5}$ (autour de l'axe $\overrightarrow{z_0}$)).

L'extrémité du bras du robot se met en mouvement le long des axes *X*, *Y*, *Z*, dans le sens positif ou négatif selon l'appui sur les boutons *X*+/-, *Y*+/-, *Z*+/-.

A titre d'exemple, l'extrémité du bras du robot évolue le long de l'axe *X* dans le sens positif en cliquant sur le bouton *X*1+, il évolue dans le sens négatif en cliquant sur le bouton *X*1-.

Le fait de cliquer sur *R*+/- provoque une rotation de l'outil (effective en présence du servo-moteur fixé à l'outil Pince, Ventouse ou Laser) autour de l'axe *R* (dans le sens positif ou négatif).

Notez que la mise en mouvement le long de l'axe *Y* provoque également la mise en mouvement de l'axe *R* (effectif en présence du servo-moteur fixé à l'outil Pince, Ventouse ou Laser), afin d'assurer une posture constante de l'outil au cours du mouvement.

Figure 11 : Mise en mouvement, dans le sens + ou -, le long des axes *X*, *Y*, *Z* du repère de base, ou autour de l'axe *R*.

Question 4 : L'outil Feutre (muni de son capuchon) étant fixé à l'extrémité du bras du robot, vérifier que le bras, une fois mis dans sa position/configuration initiale (voir la Figure 4), est tel que :

- les coordonnées du point 0_4, calculées dans la question 1, correspondent à celles (notées *X*, *Y*, *Z*) obtenues dans l'encadré « Operation Panel » en sélectionnant préalablement, dans l'outil « Advanced », les valeurs adéquates de *xBias*, *yBias et zBias* sachant que ces axes se rapportent au repère (O_4, x_5, y_5, z_5) (**attention :** il s'agit bien des axes x_5, y_5, z_5 et non des axes x_4, y_4, z_4 !!) ;
- les coordonnées du point *P*, calculées dans la question 1, correspondent à celles obtenues dans l'encadré « Operation Panel » en sélectionnant préalablement les valeurs adéquates de *xBias*, *yBias et zBias*. Vérifiez que ces coordonnées correspondent également à celles obtenues en sélectionnant l'outil 'Pen' ;
- les coordonnées du point *PF*, calculées dans la question 1, correspondent à celles obtenues dans l'encadré « Operation Panel » en sélectionnant préalablement les valeurs adéquates de *xBias*, *yBias et zBias*.

E. Types de mouvement : Point To Point, Continuous Path, ARC

Présentons les types de mouvements applicables au bras du robot :

- Mouvement en mode « Point To Point » (PTP) : MOVJ, MOVL et JUMP.

1) Instruction MOVJ (Joint movement) : Le calcul de la commande se fait dans l'espace articulaire, chaque articulation évolue simultanément de sa position angulaire initiale/courante vers celle de destination, donnant lieu à la trajectoire de l'outil, décrite dans la figure qui suit, du point initial/courant (noté A dans la figure) au point B (associé à l'instruction : MOVJ B).

2) Instruction MOVL (Rectilinear movement) : Chaque articulation évolue simultanément de manière à déplacer l'outil en ligne droite, comme décrit dans la figure qui suit, du point initial/courant A au point de destination B (associé à l'instruction). Le calcul de la commande, plus complexe, se fait dans l'espace opérationnel.

Figure 12 : instructions MOVJ B, MOVL B.

3) Instruction JUMP : Le mouvement est similaire à celui réalisé avec MOVJ avec en plus un dégagement au niveau du point initial/courant (A) et une approche au niveau du point de destination B (associée à l'instruction). Chaque articulation évolue simultanément de manière à réaliser la séquence de mouvements suivante, comme décrite dans la figure qui suit :
   - Echappement/dégagement de l'outil du point initial/courant A vers un point, noté $\bar{A}$, situé à la verticale du point A à une hauteur h_A,
   - Mouvement horizontal (selon un comportement analogue à MOVJ) du point $\bar{A}$ vers un point $\bar{B}$ (situé à une hauteur h_B à la verticale du point B),
   - Approche/descente de l'outil pour atteindre le point de destination B.

   Concernant les valeurs de h_A, h_B :
   - si $Z_A = Z_B$ alors $h_A = h_B = JumpHeight$ où $JumpHeigh$ en *mm* est à définir au préalable dans l'onglet Setting/Playback/JumpParam (voir Figure 14),
   - si $Z_A > Z_B$ alors $h_A = JumpHeight$ (avec $h_B > JumpHeight$),
   - si $Z_A < Z_B$ alors $h_B = JumpHeight$ (avec $h_A > JumpHeight$).

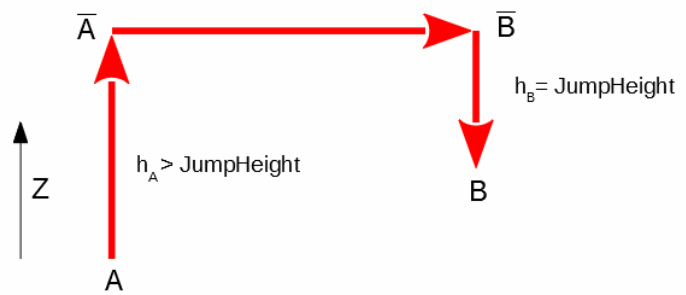

Figure 13 : instruction JUMP B dans le cas où $Z_A < Z_B$.

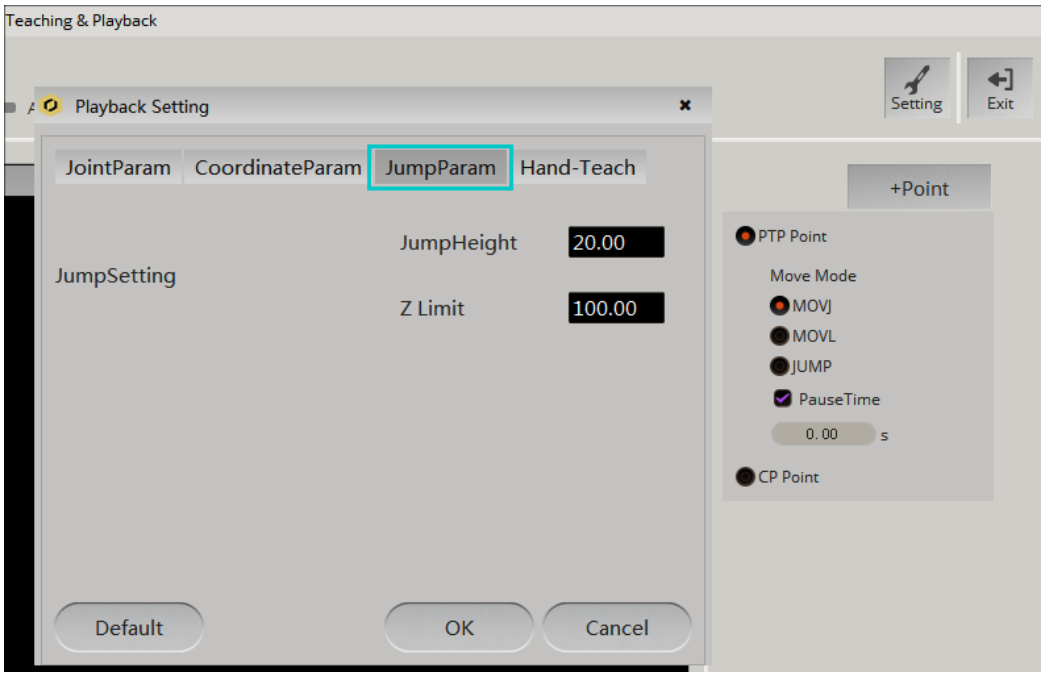

Figure 14 : assignation du paramètre JumpHeight.

- Mouvement en mode « Continuous Path » (CP). Ce mode permet un mouvement de l'outil du point initial/courant vers un point (associé à l'instruction) sans s'y arrêter (contrairement au mode PTP), ceci pour aller, *via* une autre instruction de mouvement, vers un autre point (sans s'y arrêter ou de destination), ce qui permet une certaine fluidité du mouvement (sans à-coups) lors du passage par ce point.

- Mouvement en mode « ARC » à partir du point initial/courant, noté A, sous la forme d'arc défini par 2 autres points B, C (un point intermédiaire (noté *cirPoint*), un point final (noté *toPoint*)) non alignés, comme décrit dans la figure qui suit, pour permettre, par exemple, le contournement d'un obstacle.

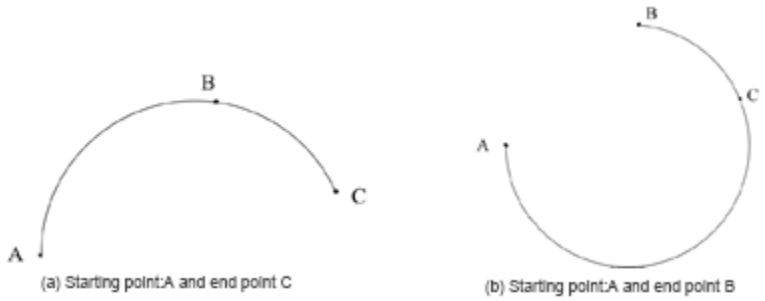

Figure 15 : Trajectoire en arc.

NB : Les mouvements en mode « Continuous Path » ne sont pas accessibles dans les environnements « Teaching & Playback » et « Blockly », ceux en mode « ARC » ne sont pas accessibles dans l'environnement « Blockly ».

F. Environnements de programmation « Teachning & Playback » et « Blockly »

Il existe plusieurs manières de réaliser dans l'application DobotStudio un programme de commande du bras du robot, notamment à travers les environnements : « Teaching & Playback » et « Blockly » décrits ci-dessous.

1. Teaching & Playback

Après avoir connecté le robot à l'application DobotStudio, l'environnement de développement, décrit dans la figure qui suit, apparaît en cliquant sur « Teaching & Playback » dans la page principale de DobotStudio.

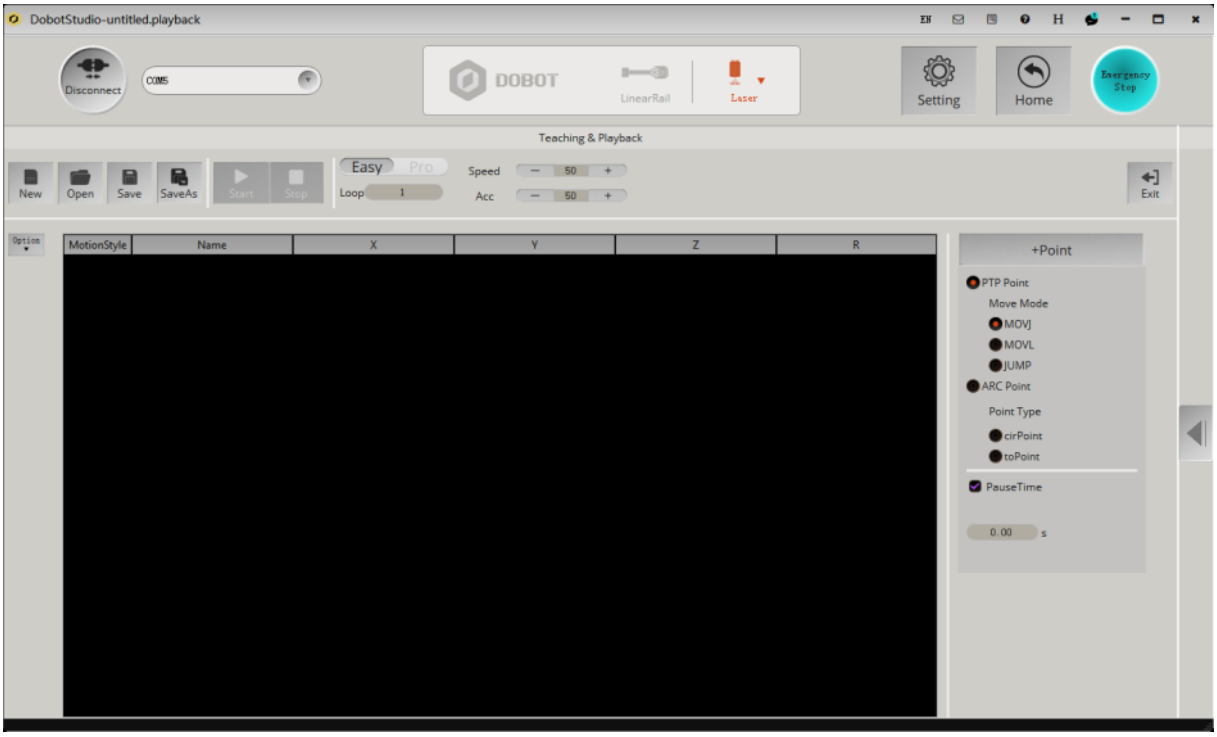

Figure 16 : Fenêtre « Teaching & Playback ».

- Il est possible : de commuter entre les modes « **Easy** » (par défaut) et « **Pro** » (le mode Pro n'ayant pas d'intérêt dans le cadre du TP) ; d'assigner le nombre de boucle (loop), la vitesse et l'accélération (en pourcentage). Voir la figure et le tableau descriptif qui suivent.

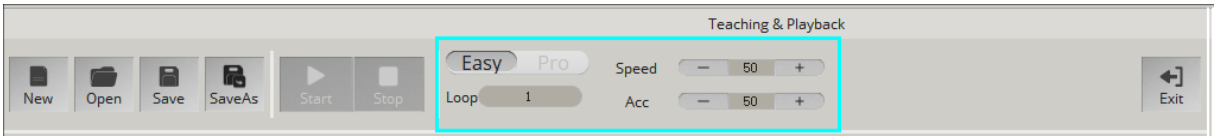

Figure 17 : Assignation Easy/Pro, Loop, Speed et Acceleration.

Items	Description
Easy/Pro	Le mode Pro permet des fonctionnalités supplémentaires par rapport au mode Easy comme le mode hors ligne, l'interface d'E/S.
Loop	Définit le nombre de fois par lequel le bras du robot repasse les étapes enregistrées (valeur par défaut : 1, plage de valeurs : 1-999999).
Speed	Définit le rapport de vitesse lors du playback (valeur par défaut : 50%, plage de valeur : 0-100%).
Acceleration (Acc)	Définit le rapport d'accélération lors du playback (valeur par défaut : 50%, plage de valeurs : 0-100%).
Exit	Retour à la page principale de DobotStudio.

- Les lignes de commande sont programmées dans la fenêtre centrale de développement (voir la Figure 16) :
  - en définissant le mode de mouvement sachant que 4 instructions de mouvement sont possibles : MOVJ, MOVL, JUMP, ARC,
  - en sauvant les points à associer aux instructions,
  - en fixant les temps de pause au niveau des points,

  voir le tableau descriptif suivant.

Items	Description
Motion mode	Sélectionne le mode de mouvement : - Point To point (PTP) avec un accès aux instructions MOVJ, MOVL ou JUMP, - ARC étant donné un point initial (atteint par exemple *via* une instruction MOVJ), un point intermédiaire « *cirPoint* » et un point de destination « *toPoint* ».
+Point	Crée un nouveau point dans la liste des points sauvés.
Pause time	Définit le temps de pause (en seconde) associé à un point.

Plus précisément, une ligne de commande est définie par les champs suivants :
- « MotionStyle » pour sélectionner le mode de mouvement MOVJ, MOVL, JUMP ou ARC,
- « Name » pour associer un nom à la ligne de commande,
- « *X, Y, Z, R* » pour définir les coordonnées du point (*X, Y, Z* sont les coordonnées cartésiennes dans le repère de base, *R* est l'angle de rotation du servo-moteur de l'articulation $J5$) associé à l'instruction (auquel s'ajoutent les coordonnées « *X', Y', Z', R'* » pour définir le second point dans le cas d'instruction ARC). Le déplacement du bras vers une position souhaitée se fait en enfonçant le bouton « Déverrouiller » situé sur l'avant-bras du robot et représenté par un cadenas ouvert. Le fait d'appuyer sur le bouton « **+Point** » permet d'enregistrer le point en lien avec l'instruction de mouvement en cours (remarque : il est possible d'enregistrer un point dès lors que vous relâchez le bouton « Déverrouiller » si vous cochez au préalable la fonction « Enable Handhold Teaching » (en choisissant « release the UNLOCK button » dans le menu déroulant) située dans la fenêtre « Setting>PlayBack>HandHold Teaching »).
- « PauseTime » pour associer un temps de pause (en seconde) au point associé à l'instruction de mouvement.

Un champ d'une ligne de commande peut être modifié (copié, collé, coupé, etc.) en le mettant en surbrillance (par un double-clique), voir la figure et le tableau descriptif qui suivent.

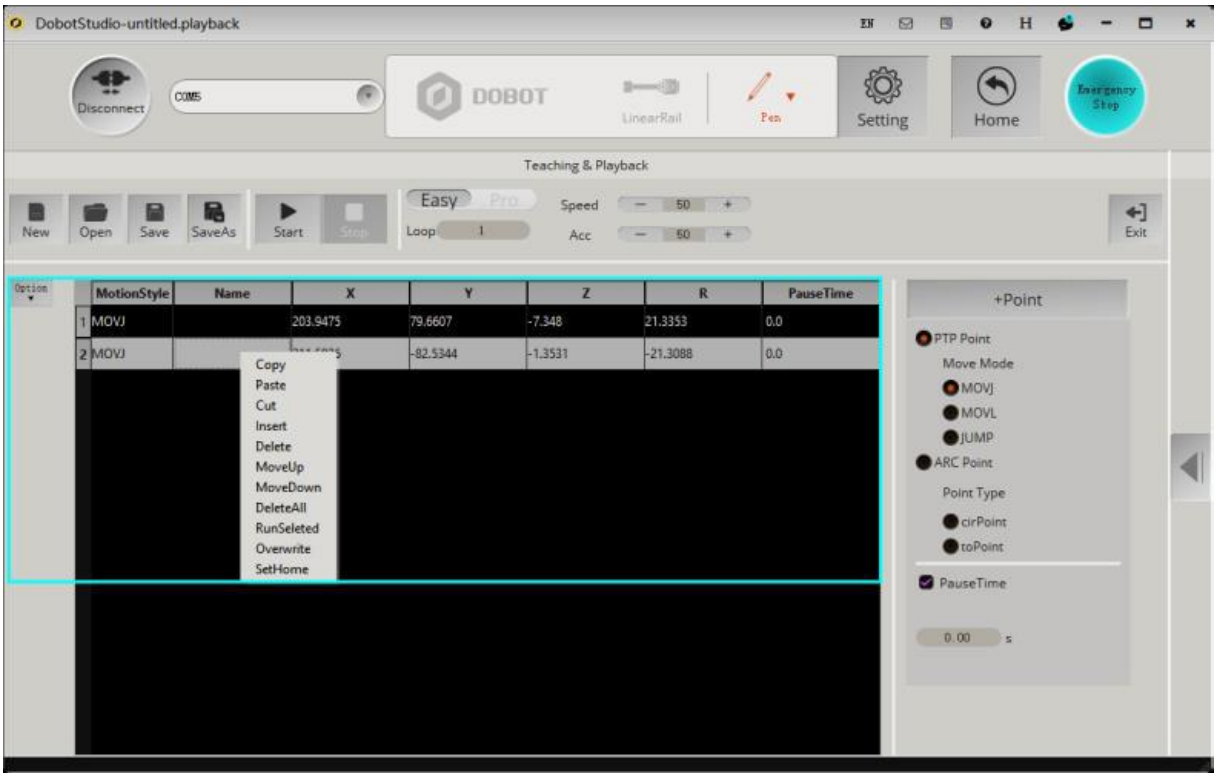

Figure 18 : La liste des points sauvés.

Items	Description
Clique droit	Dans le menu Popup, vous pouvez modifier un point sauvé mis en surbrillance avec copier, coller, couper, etc.
Double-clique	Double-clique un champ pour modifier sa valeur.

Question 5 : Réaliser un programme dans l'environnement Teaching & Playback qui permette de tracer, à l'aide du Feutre, sur la feuille fournie en TP une trajectoire continue du point A au point B, puis du point C au point D, ceci sans sortir de la zone autorisée.

2. Blockly

Basé sur un projet open source de Google, Blockly est un environnement de programmation graphique permettant de constituer dans la fenêtre de travail des lignes de commande en important par Drag and Drop des blocs (représentés par des pièces de puzzle) issus d'une bibliothèque.

L'environnement de développement apparaît en cliquant sur Blockly dans la page principale de Dobot Studio. Considérons l'exemple décrit dans la figure qui suit.

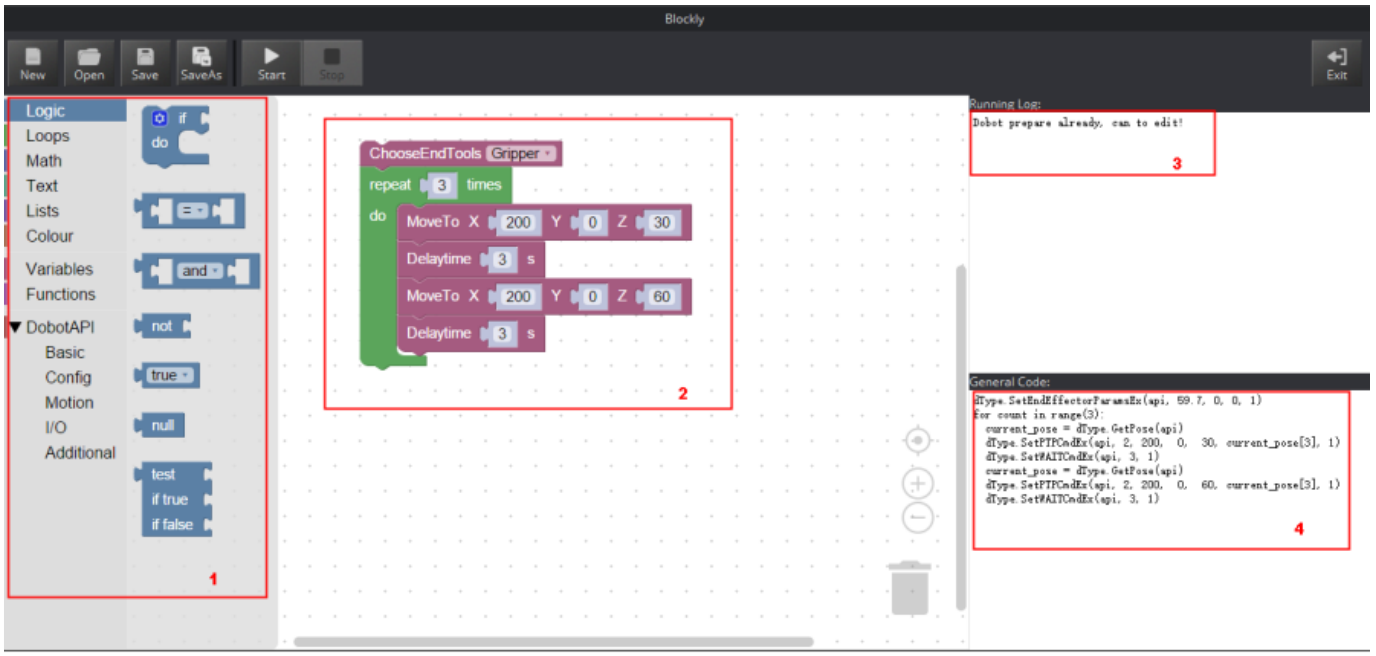

Figure 19 : Environnement de développement Blockly.

Les boutons « **New**, **Open**, **Save**, **SaveAs** », situés en haut à gauche, permettent de créer, ouvrir, sauvegarder un fichier sachant que le chemin par défaut, qu'il est possible de modifier *via* SaveAs, est : C:..\DobotStudio\config\bystore.

Les boutons « **Start**, **Stop** », situés après les boutons « New, Open, Save, SaveAs », permettent le démarrage, l'arrêt du programme en cours.

La table qui suit décrit les différentes fenêtres de l'environnement de développement.

No.	Description de la fenêtre
1	Fenêtre contenant les blocs, développés sous Blockly, utilisés pour programmer le robot. Pour faciliter leur utilisation, ces blocs sont rangés dans différents modules (Logic, Loops, Math, Lists, … et DobotAPI). Ce dernier, spécifique à DobotStudio, permet de : fixer la vitesse/accélération ; paramétrer l'outil ; mettre en mouvement le bras du robot dans l'espace articulaire ou opérationnel ; configurer les interfaces I/O.
2	Fenêtre de travail où sont importés, par Drag and Drop, les blocs issus de la fenêtre 1 pour constituer le programme Blockly à même de réaliser la trajectoire du bras du robot et les tâches à effectuer durant la trajectoire.
3	Fenêtre « Running Log » où est décrit le log d'exécution du programme Blockly en cours.
4	Fenêtre « General Code » contenant les lignes de codes API (Application Programming Interface), générées automatiquement, équivalentes à chacun des blocs du programme Blockly en cours.

A titre d'exemple, le programme Blockly décrit dans la figure qui suit permet au bras du robot, muni de sa Pince, d'effectuer 3 fois un mouvement d'aller-retour le long de l'axe Z avec une pause d'une seconde aux 2 points situés à l'extrémité de la trajectoire.

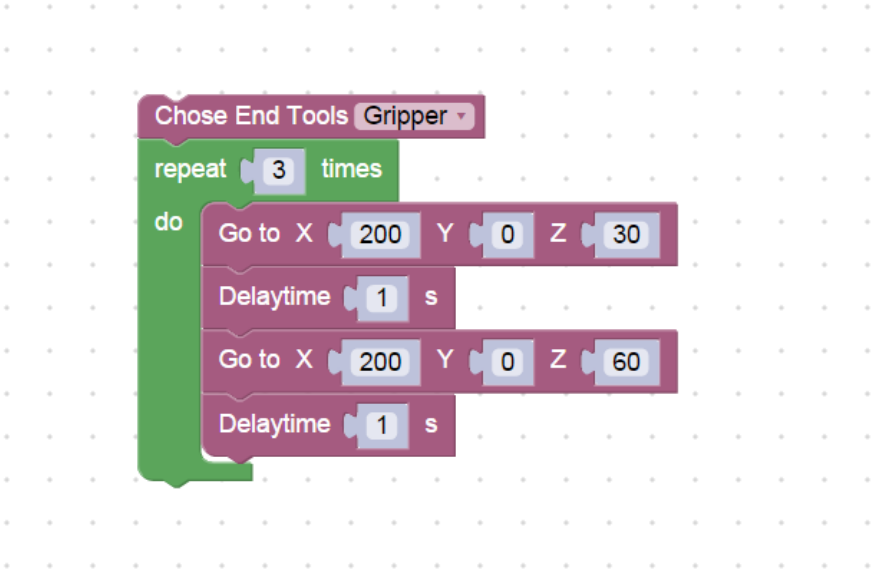

Figure 20 : Exemple de programme Blockly.

Description des instructions du programme précédent :

+ « Chose End Tools Gripper » indique que l'outil est la Pince (Gripper) ; ainsi le modèle géométrique du robot sera tel que le point extrême considéré sera l'extrémité de la Pince,
+ « repeat 3 times do » permet de répéter 3 fois les 4 instructions suivantes contenues dans la boucle :
    + « Go to X 200 Y 0 Z 30 » met en mouvement dans l'espace opérationnel le bras du robot afin que l'extrémité de la Pince atteigne le point (défini dans le repère de base) de coordonnés $(200, 0, 30)$,
    + « Delaytime 1 s » provoque une pause d'1 seconde au point courant,
    + « Go to X 200 Y 0 Z 60 » met en mouvement dans l'espace opérationnel le bras du robot afin que l'extrémité de la Pince atteigne le point de coordonnés $(200, 0, 60)$,
    + « Delaytime 1 s » provoque une pause d'1 seconde au point courant.

- **Description de quelques blocs propres au robot (contenus dans le module « DobotAPI »)**

+ blocs de base du sous module « Basic » :

Home	Positionne le bras du robot dans sa posture par défaut (valeurs par défaut ($q_2 = 90°, q_3 = 0°$) modifiables *via* « Setting>Initial Pos ») pour obtenir une référence de position correcte.
GetTime	
Delaytime 0 s	Provoque une pause (ici 0) en seconde entre 2 commandes.

+ blocs de configuration du sous module « Config » :

ChooseEndTools SuctionCup (Gripper, Laser ou Pen)	Indique le type d'outil standard (SuctionCup, Gripper, Laser ou Pen) fixé à l'extrémité du bras.
Set End Effector Params XBias 0 YBias 0 ZBias 0	Indique, à travers les valeurs XBias, YBias, ZBias rapportées au repère (O_4, X_5, Y_5, Z_5) indiqué dans la Figure 4, l'outil (non standard) fixé à l'extrémité du bras.
SetMotionRatio VelocityRatio 20 AccelerationRatio 50	Fixe le ratio de la vitesse (ici 20% de 500mm/s) et de l'accélération (ici 50% de 500mm/s^2).
SetJointSpeed Velocity 20 Acceleration 50	Fixe le ratio de la vitesse (ici 20% de 500mm/s) et de l'accélération (ici 50% de 500mm/s^2) des articulations 1, 2, 3, 5.
SetCoordinateSpeed Velocity 20 Acceleration 50	Fixe le ratio de la vitesse (ici 20% de 500mm/s) et de l'accélération (ici 50% de 500mm/s^2) relativement aux coordonnées cartésiennes *X*, *Y*, *Z* et de la rotation *R* (*i.e.*, *J*4).
SetJumpHeight Height 20	Fixe la hauteur de levage (ici 20mm) utilisée avec l'instruction JUMP.

+ blocs de mouvement du sous module « Motion » :

JumpTo X 200 Y 0 Z 0	Met en mouvement en mode JUMP le bras du robot dans l'espace opérationnel pour atteindre le point de coordonnées X, Y, Z, égal en mm à 200, 0, 0.
MoveTo X 200 Y 0 Z 0	Met en mouvement en mode MOVL le bras du robot dans l'espace opérationnel pour atteindre le point de coordonnées X, Y, Z, égal en mm à 200, 0, 0.
MoveDistance ΔX 0 ΔY 0 ΔZ 0	Met en mouvement en mode MOVL le bras du robot dans l'espace opérationnel pour atteindre un point de coordonnées défini par les incréments $\Delta X, \Delta Y, \Delta Z$, par rapport au point courant.
SetR 0	Permet de fixer la valeur angulaire *R*, ici à 0, sans modifier les autres angles.
SetJointAngle Joint1 0 Joint2 45 Joint3 45	Met en mouvement en mode MOVJ le bras du robot dans l'espace articulaire correspondant aux valeurs *J*1, *J*2, *J*3 ici égales en degré à 0, 45, 45 respectivement (la valeur de *J*4 restant inchangée).
GetCurrentCoordinate x (y, z ou r)	Permet d'obtenir la valeur de la coordonnée cartésienne x, y, z ou r (ici x).
GetJointAngle Joint1 (Joint2, Joint3 ou Joint4)	Permet d'obtenir la valeur de la coordonnée angulaire $J1, J2, J3$ ou $J4$ (ici $J1$).

SuctionCup ON (OFF)	Met en action la pompe à air (ON) ou non (OFF).
Gripper Gripper (Release ou OFF)	Ferme la pince (Gripper), relâche la pince (Release) ou rend inactive la pince (OFF).

Les blocs contenus dans le sous-module « I/O » permettent d'interagir avec l'interface d'entrées/sorties. Les blocs contenus dans le sous-module « Additional » permettent de préciser quelques paramètres complémentaires relatif au Laser, au capteur photo-électrique.

Remarque : Le bloc « set (item) to », issu du module « Variables », permet l'usage de variables dans un programme. Par exemple, les 2 lignes de commande décrites dans la figure qui suit permettent :
- d'attribuer la coordonnée de l'axe Z de la position courante (voir bouton « Déverrouiller », situé sur l'avant-bras du robot, pour positionner le bras) à la variable « z »,
- de déplacer le bras du robot vers la position définie par les coordonnées cartésiennes $(200, 0, z)$.

Figure 21 : Utilisation de variable dans Blockly.

Question 6 : Munir le robot de la Pince (Gripper), voir la figure qui suit pour le câblage. Les cubes « 1 » et « 2 » étant initialement positionnés respectivement sur les emplacements *A* et *B* (voir le schéma décrit sur la feuille fournie en TP), il s'agit de réaliser un programme Blockly permettant de :
- positionner le cube « 1 » sur l'emplacement *C*,
- empiler le cube « 2 » sur le cube « 1 »,
- attendre 2 secondes,
- positionner les cubes « 1 » et « 2 » sur les emplacements *D* et *E* respectivement.

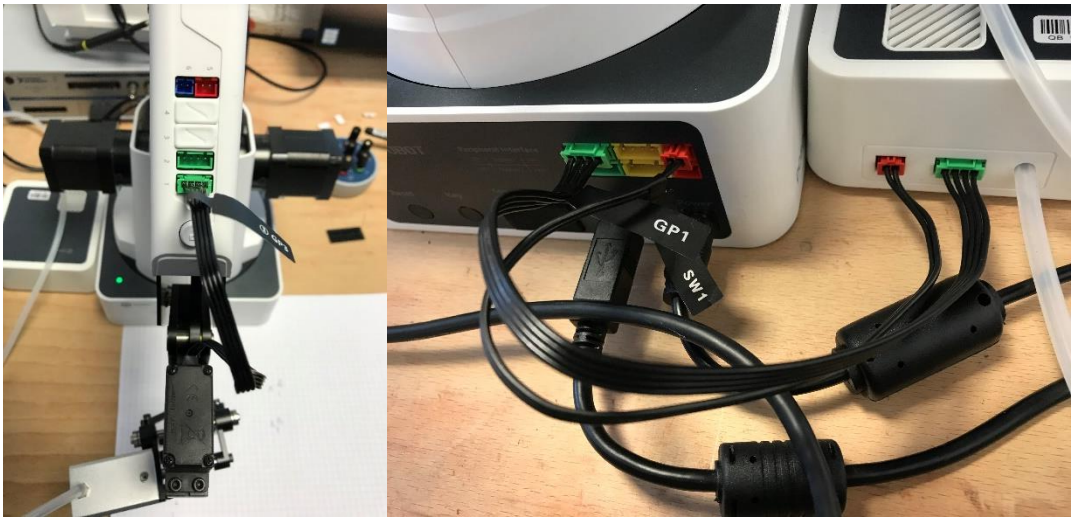

Figure 22 : Connexion de la pince et du mini-compresseur au robot.

- **Interprétation des blocs sous la forme d'API spécifiques au robot**

A chacun des blocs correspond des lignes de commande faisant appel à des API. Soit, par exemple, le bloc Blockly :

SetJoint Angle Joint1 0 Joint2 45 Joint3 45

pour lequel correspond les 2 lignes de codes suivantes :

```
current_pose = dType.GetPose(api)
dType.SetPTPCmdEx(api, 4, 0, 45, 45, current_pose[7], 1)
```

Pour permettre l'exécution du code correspondant aux blocs Blockly, des lignes de code sont placées en amont et sans apparaître dans la fenêtre 4 telles que :

```
import DobotDllType as dType

CON_STR = {
dType.DobotConnect.DobotConnect_NoError:  "DobotConnect_NoError",
dType.DobotConnect.DobotConnect_NotFound: "DobotConnect_NotFound",
dType.DobotConnect.DobotConnect_Occupied: "DobotConnect_Occupied"}

#Load Dll
api = dType.load()

#Connect Dobot
state = dType.ConnectDobot(api, "", 115200)[0]
print("Connect status:",CON_STR[state])
```

pour respectivement :
- importer la bibliothèque *DobotDllType* en la renommant *dType*,
- charger les API en obtenant l'objet, nommé *api*, de type Store,
- connecter le robot à l'application DobotStudio (en écrivant l'état de la connexion).

L'instruction *current_pose = dType.GetPose(api)* fait appel à l'API *GetPose*, décrite p.13 du document « Dobot Magician API description », définie par :

*int GetPose(Pose *pose)*

et permet d'obtenir la position en temps réel du bras du robot., au sens où *current_pose* est un tableau contenant la position et les valeurs articulaires, à savoir, x, y, z, r, J1, J2, J3, J4.

L'instruction *dType.SetPTPCmdEx(api, 4, 0, 45, 45, current_pose[7], 1)* fait appel à l'API *SetPTPCmdEx*, décrite p.36 du document « Dobot Magician API description », définie par :

*int SetPTPCmd(PTPCmd *ptpCmd, bool isQueued, uint64_t* *queuedCmdIndex)

et permet la mise en mouvement du bras du robot pour atteindre le point sachant que :
- $PTPMode = 4$ est tel que le mouvement s'opère dans l'espace articulaire (in Joint coordinate system) avec comme valeurs articulaires $J1 = 0, J2 = 45, J3 = 45, J4 = current\_pose[7]$ où $current\_pose[7]$ contient la valeur articulaire *J*4 de la position courante,
- $isQueued = 1$ pour ajouter la commande dans la « queue d'exécution » (chaque commande placée dans la « queue d'exécution » étant exécutée dans l'ordre de son arrivée dans la queue).