

## Table des matières

2

Fiche 1	Présentation Générale .....	3
	Introduction .....	3
Fiche 2	Présentation du matériel .....	3
	Présentation globale .....	3
	Mini-ordinateur Odroid C1+ .....	4
	Carte Teensy 3.6.....	4
	Carte d'interfaçage de capteurs et d'actionneurs.....	5
	Servomoteurs Dynamixel .....	5
	Codeurs absolus .....	6
	Effecteur .....	6
	Micro-contrôleur ATmega328PB.....	7

Capteurs de distance laser .....	7
Capteur de couleur .....	7
Pince actionnée par un servomoteur .....	7
Fiche 3    Mise en service – Logiciel MyWiz.....	9
Fiche 4    Modélisation mécanique.....	12

# Fiche 1 PRESENTATION GENERALE

## Introduction

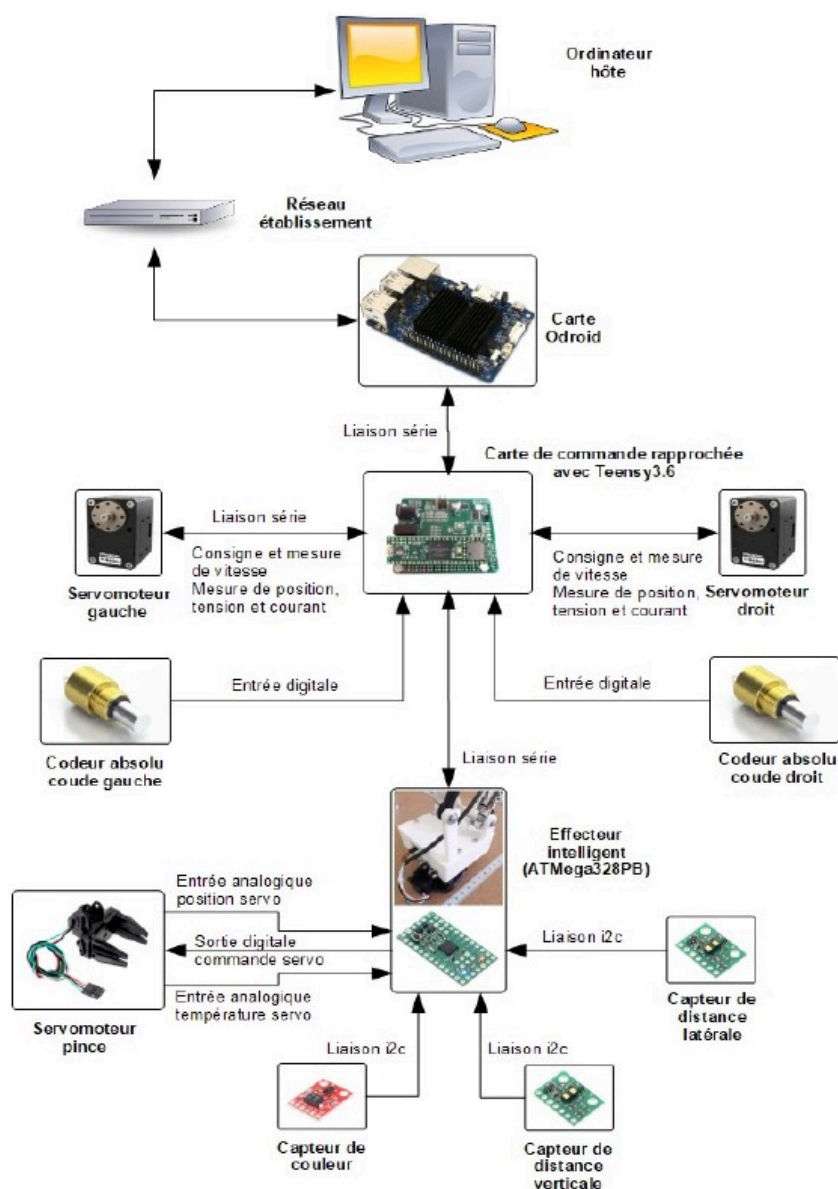
Les robots de type « delta » sont des robots dits parallèles et sont principalement utilisés pour faire du « pick and place » c'est à dire de la prise et dépose d'objet pour de l'assemblage ou du rangement.

Le Delta2D est un robot delta à 2 degrés de libertés motorisés par des servomoteurs Dynamixel, fonctionnant dans le plan vertical. Commandé par une carte Teensy 3.6 et doté d'un effecteur intelligent, il permet de réaliser différentes expériences de pilotage en coordonnées cartésiennes et articulaires, exploitant ses nombreux capteurs et actionneurs.

# Fiche 2 PRESENTATION DU MATERIEL

## Présentation globale

Le diagramme d'échange des signaux entre les différents composants est représenté sur la page suivante. Nous décrivons dans la suite de ce chapitre les différents éléments matériels intégrés dans ce système.



## Mini-ordinateur Odroid C1+

Cette carte permet d'interfacer le système à l'ordinateur hôte via le réseau local de l'établissement. Elle est « cliente » de ce réseau et peut-être vue par tous les ordinateurs qui y sont connectés.

Son facteur de forme est identique à la célèbre carte Raspberry Pi. Elle a été préférée à cette dernière pour des raisons de performances et de fiabilité.

Elle est reliée à la carte Teensy 3.6 (voir plus loin) via un câble USB afin de communiquer avec cette dernière pour :

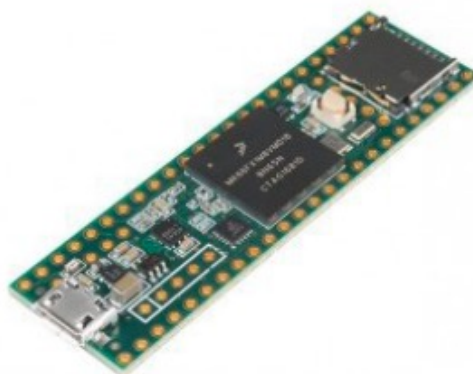
- Envoyer des ordres de pilotage
- Lire les mesures effectuées sur le robot



## Carte Teensy 3.6

La carte Teensy 3.6 est le cerveau du système. Ses caractéristiques sont les suivantes :

- Microprocesseur: ARM Cortex M4 32 bits 180 MHz à virgule flottante
- Mémoire flash: 1 MB
- Mémoire SRAM: 256 kB
- Mémoire EEPROM: 4 kB
- 32 broches d'E/S dont 22 PWM
- 25 entrées analogiques 13 bits
- 2 sorties analogiques 12 bits
- Support USB avec transferts DMA (Direct Access Memory)
- Support Ethernet 100 Mbit/sec
- Support pour carte microSD
- Bus: 6 x série, 2 x CAN, 4 x I2C et 3 x SPI
- Interface I2S
- Gestion des interruptions
- Module RTC
- 11 entrées pour capteur tactile
- 14 temporisateurs
- Régulateur 3,3 Vcc/100 mA
- Dimensions: 63 x 18 x 5 mm



## Carte d'interfaçage de capteurs et d'actionneurs

Cette carte est prise en sandwich entre l'Odroid et la Teensy, qui bénéficient toutes les deux de son convertisseur de tension 5 V 2 A.

Elle propose les fonctionnalités et connectiques suivantes :

- Passage d'une liaison série à deux voies « full-duplex » (RX / TX) en 3.3 V à une liaison série une voie « half-duplex » en 5 V pour le pilotage des servomoteurs Dynamixel via la carte Teensy 3.6
- Connecteurs reliés aux entrées de mesure de PWM de la carte Teensy pour la capture des signaux des codeurs absolus digitaux
- Connecteur pour le branchement de la ligne série de pilotage de l'effecteur intelligent

## Servomoteurs Dynamixel

Le robot est motorisé par deux servomoteurs intelligents Dynamixel XH430-W350-T, dont les caractéristiques sont les suivantes :

- Micro-contrôleur interne : ARM CORTEX-M3 (72 [MHz], 32Bit)
- Capteur de position : codeur absolu sans contact (360° sur 12 bits, AMS AS5045)
- Moteur : coreless (Maxon)
- Rapport de réduction : 353.5
- Jeu : 0.25°
- Tension d'alimentation : de 10 à 14.8 V (12 V recommandés)
- Couple et courant de blocage à 12 V : 3.4 N.m, 1.3 A
- Courant consommé à l'arrêt : 40 mA
- Vitesse à vide à 12 V : 30 tr/min
- Charge radiale max : 40 N à 10 mm du moyeu



- Charge axiale : 20 N
- Modes de pilotage :
  - Asservissement en courant
  - Asservissement en vitesse
  - Asservissement en position (entre 0 et 360°)
  - Asservissement en position étendu (multi-tour)
  - Asservissement dual en courant et en position
  - Pilotage en PWM (commande en tension en boucle ouverte)
- Mesures disponibles : position, vitesse, courant, temps, trajectoire, température, tension d'entrée, etc...
- Communication digitale série asynchrone TTL Half Duplex sur 8 bits avec 1 bit de stop, sans parité
- Vitesse de communication : de 9600 bauds à 4.5 Mbauds
- Température de fonctionnement : de -5 à 80 °
- Matériaux : réducteur en métal, faces avant et latérales en métal, face arrière en plastique
- Poids : 82 g
- Dimensions (largeur, hauteur, profondeur) : 28.5 x 46.5 x 34 mm

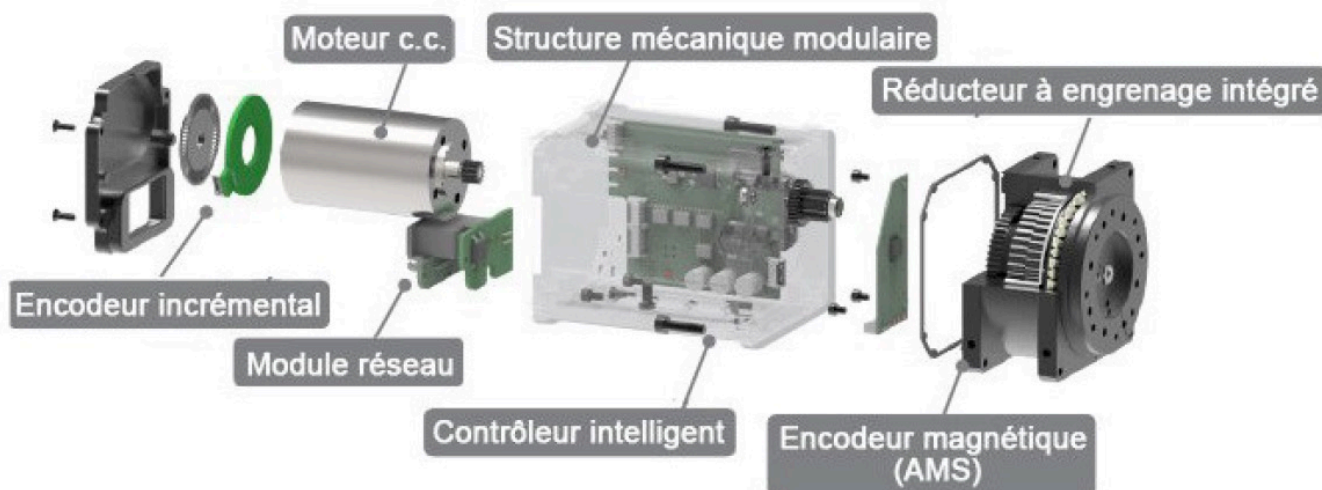
Les paramètres internes (utilisés dans le modèle de simulation) sont les suivants :

- Constante de couple : 0.01 N.m/A
- Résistance: 9.2 Ohms
- Inductance : 0.034 mH
- Inertie du rotor: 2.25e-7 kg.m<sup>2</sup>
- Frottement: 0.5 N.m.s/rad



- Rapport de réduction: 353.5

Compte-tenu de la variété des modes de pilotage et des données mesurées disponibles, on est donc très loin d'un servomoteur de modélisme.



## Codeurs absolus

Les angles entre les bras supérieurs et inférieurs sont mesurés par deux codeurs absolus (dont la fiche technique, MA3\_datasheet.pdf, est fournie dans le répertoire « References » de l'archive), respectivement placés au point D et au point E.

L'axe du codeur étant monté sur roulement à bille, il sert à la fois à la mesure et à la liaison entre les bras.

- Axe de rotation monté sur roulement à bille
- Interface digitale (les capteurs sont connectés sur les broches 5 et 6 de la carte Teensy afin de permettre une mesure du rapport cyclique de PWM)
- Plage de mesure de 0 à 360 degrés
- Résolution sur 12 bits (soit 0.35 degrés)

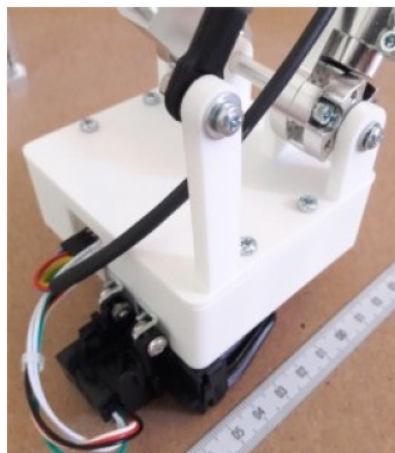
Le nombre de rotations (en millions) supportable par le codeur en fonction de la charge radiale est égal à avec : charge radiale en kg



## Effecteur

L'effecteur est fixé au robot aux points F (bras inférieur droit) et H (bielle inférieure).

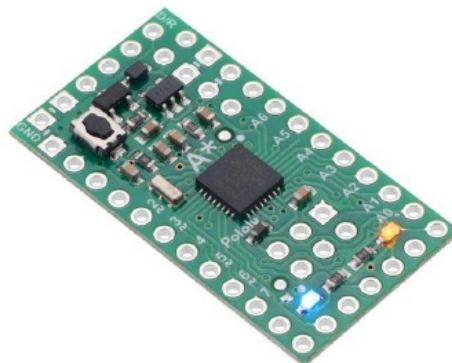
Il intègre les éléments décrits ci-dessous.



## Micro-contrôleur ATmega328PB

les capteurs et actionneurs de l'effecteur sont gérés par un micro-contrôleur ATmega328PB, relié à la carte Teensy par une liaison série. L'utilisation d'un micro-contrôleur dédié dans l'effecteur plutôt qu'un pilotage direct par la Teensy est motivée par les éléments suivants :

- L'i2c (interface utilisée par les composants intégrés à l'effecteur) a été conçu pour des distances de communication courtes
- Une liaison série permet de réduire le nombre de fils entre l'électronique de commande principale et l'effecteur
- L'ATmega328PB est utilisé plutôt qu'un autre car il possède deux voies i2c séparées, ce qui est très utile pour gérer les deux capteurs VL53L1X (voir plus loin) car leur adresse n'est pas modifiable.



## Capteurs de distance laser

L'effecteur intègre deux mini-LIDAR VL53L1X (dont la fiche technique, VL53L1X.pdf, est fournie dans le répertoire « References » de l'archive).

Ils communiquent en i2c avec l'ATmega328PB et mesurent respectivement :

- La distance entre la base de l'effecteur et le socle
- La distance entre le côté droit de l'effecteur et le panneau latéral droit



Ces deux mesures, dont la résolution est de 1 mm, permettent de connaître la position cartésienne de l'extrémité du robot pour (par exemple) la comparer avec les calculs réalisés à partir des angles des moteurs et des équations cinématiques directes.

## Capteur de couleur

L'effecteur intègre un capteur de couleur APDS. Il communique en i2c avec l'ATmega328PB et renvoie, entre autres, les 3 composantes Rouge – Vert – Bleu de l'objet qui se trouve dans son champ de détection.

Attention : la détection de couleur jaune n'est pas fiable, comme permet de le vérifier le cube jaune fourni. Par conséquent, il est recommandé de ne pas faire d'expériences qui impliquent une détection impérative du cube jaune par le capteur.



## Pince actionnée par un servomoteur

Une pince actionnée par un servomoteur est fixée sur l'effecteur pour prendre, typiquement, des cubes de couleur (fournis). L'amplitude du mouvement est de 32 mm. L'ouverture peut être mesurée grâce à un signal analogique spécifique provenant du servomoteur et fournissant une tension comprise entre 0.5 V (pince complètement ouverte) et 2.2 V (pince complètement fermée).





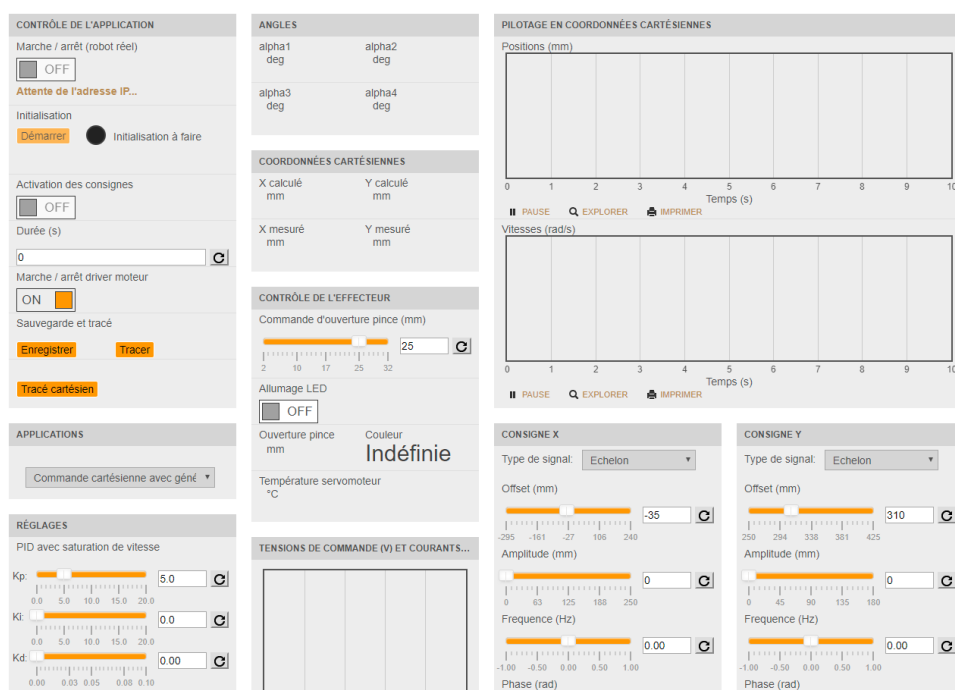
## Fiche 3 MISE EN SERVICE – LOGICIEL MyWiz

La mise en oeuvre du syst.me se fait en suivant ces différentes étapes :

- Brancher le câble Ethernet du syst.me sur une prise réseau murale RJ45
- Brancher l'alimentation 12V / 5A fournie sur le connecteur jack d'alimentation de la carte « intermédiaire » .
- Attendre environ 1 minute le démarrage du syst.me
- Lancer le logiciel MyViz
- Ouvrir le menu . Systèmes → Robots → Delta2D → Tableaux de bord. La page suivante s'affiche alors dans MyViz :

DELTA2D RÉEL	APPLICATIONS SUR ROBOT RÉEL	DELTA2D SIMULÉ	APPLICATIONS SUR ROBOT SIMULÉ
	Commande cartésienne, générateur de signal <a href="#">Ouvrir</a>		Commande cartésienne, générateur de signal <a href="#">Ouvrir</a>
	Commande cartésienne, formules <a href="#">Ouvrir</a>		Commande cartésienne, formules <a href="#">Ouvrir</a>
	Commande cartésienne, table <a href="#">Ouvrir</a>		Commande cartésienne, table <a href="#">Ouvrir</a>
	Commande articulaire, générateur de signal <a href="#">Ouvrir</a>		Commande articulaire, générateur de signal <a href="#">Ouvrir</a>
	Commande articulaire, formules <a href="#">Ouvrir</a>		Commande articulaire, formules <a href="#">Ouvrir</a>
	Commande par programme Python <a href="#">Ouvrir</a>		Commande par programme Python <a href="#">Ouvrir</a>
	Commande par programme G-code <a href="#">Ouvrir</a>		Commande par programme G-code <a href="#">Ouvrir</a>
	Commande par machine à états finis <a href="#">Ouvrir</a>		Commande par machine à états finis <a href="#">Ouvrir</a>

- Elle permet d'accéder en un simple clic aux tableaux de bord permettant de piloter la version réelle ou la version simulée du robot
- Choisir par exemple le tableau de bord de pilotage du robot réel « Commande cartésienne, générateur de signal » en cliquant sur le bouton « Ouvrir » correspondant.
- Ce tableau de bord s'ouvre dans MyViz :



The screenshot displays the MyViz software interface with several control panels:

- CONTRÔLE DE L'APPLICATION:** Includes buttons for 'Marche / arrêt (robot réel)' (OFF), 'Attente de l'adresse IP...', 'Initialisation' (Démarrer, Initialisation à faire), 'Activation des consignes' (OFF), 'Durée (s)' (0), 'Marche / arrêt driver moteur' (ON), 'Sauvegarde et tracé' (Enregistrer, Tracer), and 'Tracé cartésien'.
- APPLICATIONS:** A dropdown menu showing 'Commande cartésienne avec géné...'.
- RÉGLAGES:** A section for PID control with sliders for Kp, Ki, and Kd, each with a numerical input field.
- ANGLES:** A table showing calculated and measured angles for alpha1, alpha2, alpha3, and alpha4 in degrees.
- COORDONNÉES CARTÉSIENNES:** A table showing calculated and measured X and Y coordinates in mm.
- CONTRÔLE DE L'EFFECTEUR:** Includes a slider for 'Commande d'ouverture pince (mm)' and a button for 'Allumage LED' (OFF).
- TENSIONS DE COMMANDE (V) ET COURANTS...:** A section for monitoring voltage and current.
- PILOTAGE EN COORDONNÉES CARTÉSIENNES:** Two large graphs showing 'Positions (mm)' and 'Vitesses (rad/s)' over time (0 to 10 seconds). Each graph has buttons for PAUSE, EXPLORER, and IMPRIMER.
- CONSIGNE X and CONSIGNE Y:** Two panels for Cartesian coordinate control, each with a dropdown for 'Type de signal' (Echelon), a slider for 'Offset (mm)', a slider for 'Amplitude (mm)', a slider for 'Fréquence (Hz)', and a slider for 'Phase (rad)'.

- Pendant quelques secondes, le message suivant s'affiche sous le bouton ON / OFF en haut à gauche :

**CONTRÔLE DE L'APPLICATION**

Marche / arrêt (robot réel)

☐ OFF

Attente de l'adresse IP...

- Il signifie que votre ordinateur est en train de récupérer son adresse IP sur le réseau auquel est connecté le robot
- Puis, si le robot est libre, le message suivant s'affiche :

**CONTRÔLE DE L'APPLICATION**

Marche / arrêt (robot réel)

☐ OFF

Execution autorisée

- Vous pouvez alors commuter le bouton sur « ON » pour démarrer le système

Le message suivant s'affiche alors :

**CONTRÔLE DE L'APPLICATION**

Marche / arrêt (robot réel)

☒ ON

Démarrage en cours, veuillez patienter...

Le robot est en train de démarrer et d'initialiser le programme de la carte Teensy

- Après quelques secondes, le message suivant s'affiche :

**CONTRÔLE DE L'APPLICATION**

Marche / arrêt (robot réel)

☒ ON

Exécution en cours

Par ailleurs, des courbes commencent à défiler. Vous ne pouvez encore rien faire tant que le robot n'a pas été initialisé. (Positionnement des bras à l'horizontale)

- Pour ce faire, cliquer sur le bouton d'initialisation :

Initialisation

Le robot monte alors ses deux bras à l'horizontale et les valeurs correspondantes des angles des coudes et des positions cartésiennes de l'effecteur sont initialisées

- L'activation des consignes devient alors possible, pour une durée infinie (si la valeur est égale à 0) ou non :

Activation des consignes

☐ OFF

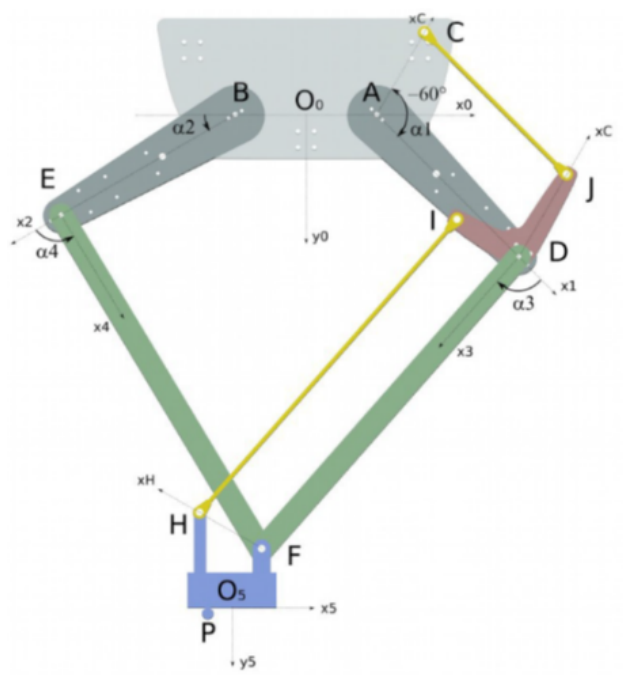
Durée (s)

- Une fois les consignes activées, il est possible de piloter le robot, de modifier les gains de l'asservissement de position,...

- Noter que l'incrémentation du temps repart de 0 à chaque commutation de OFF vers ON du bouton d'activation des consignes
- Pour arrêter le fonctionnement de façon permanente, commuter l'interrupteur sur OFF en haut à gauche du tableau de bord

## Fiche 4 MODELISATION MECANIQUE

La géométrie du robot est représentée ci-dessous :



Les dimensions sont les suivantes :

- $O_0A = O_0B = a = 60 \text{ mm}$
- $AD = BE = CJ = l = 170 \text{ mm}$
- $AC = DJ = 80 \text{ mm}$
- $DF = EF = IH = L = 330 \text{ mm}$
- $DI = FH = 60 \text{ mm}$
- $O_5F = (25, -50) \text{ mm}$
- $O_5H = (-27, -80) \text{ mm}$
- $O_5P = (-25, 25) \text{ mm}$
- $FP = (-35, -75) \text{ mm}$

Par ailleurs :

- L'altitude de l'axe des moteurs par rapport au socle est de 430 mm
- La hauteur de l'effecteur, hors attaches verticales et pince, est de 30 mm
- Dans tout ce qui suit, la position de l'effecteur correspond à la position du point P
- Les butées des bras supérieurs sont à un angle égal à  $-32 \text{ deg}$
- Lorsque  $\alpha_1$  et  $\alpha_2$  sont nuls,  $\alpha_3$  et  $\alpha_4$  sont égaux à  $134.2 \text{ deg}$
- Lorsque  $\alpha_3$  et  $\alpha_4$  sont nuls, la position de l'effecteur (point P) est égale à  $(-35, 312)$
- Le capteur de couleur se trouve à  $(25, 0) \text{ mm}$  du point P

Les masses sont les suivantes :

- Bras supérieurs: 66 g
- Bras de renvoi: 24 g
- Bras inférieur gauche: 93 g
- Bras inférieur droit: 108 g
- Bielle supérieure: 9 g
- Bielle inférieure: 13 g
- Effecteur: 150 g

La construction géométrique est telle que les points (A, C, J, D) et (D, F, H, I) forment à tout instant deux parallélogrammes, assurant en permanence l'horizontalité de l'effecteur.

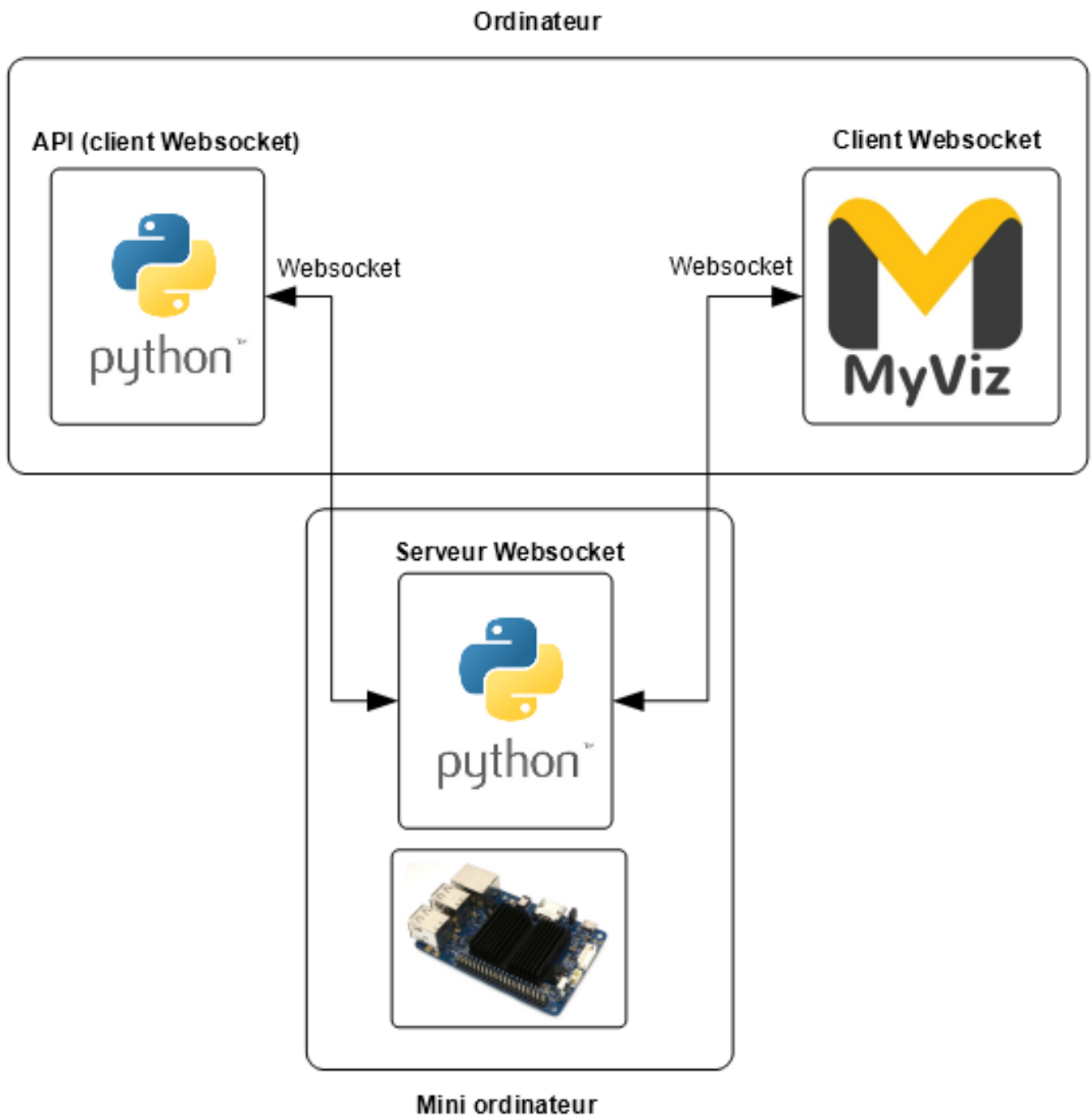
## Fiche 5 PILOTAGE PAR PROGRAMME PYTHON

---

### Présentation

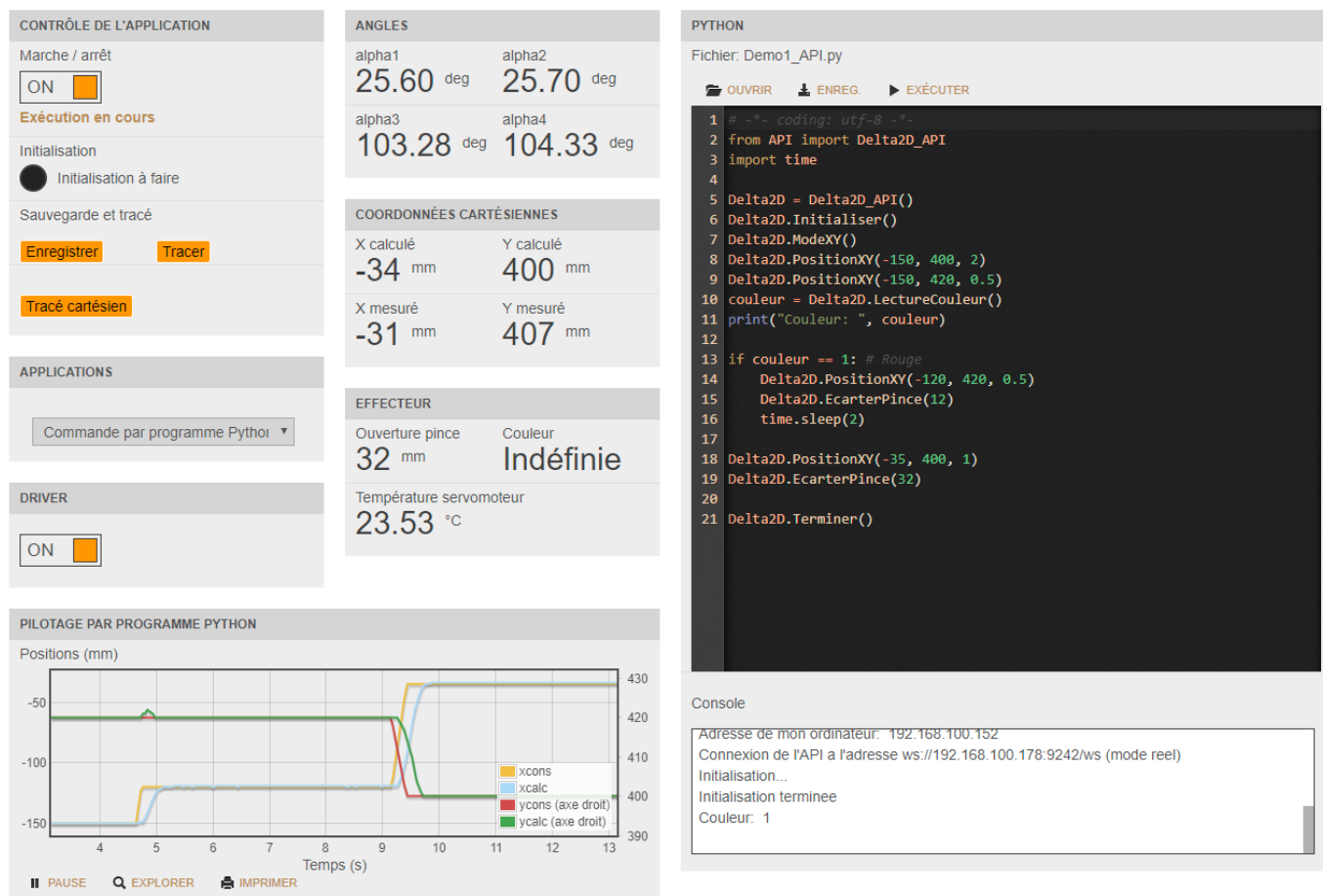
Cette expérience permet de piloter le robot non pas de façon manuelle via des curseurs dans un tableau de bord MyViz, mais par programme via une bibliothèque de commandes Python. Une telle bibliothèque s'appelle couramment « API » (Application Programming Interface) en informatique. Ce terme pourra par conséquent être parfois utilisé dans cette documentation ou sur les tableaux de bord.

L'architecture du système est la suivante :



l'API agit comme un client WebSocket supplémentaire communiquant avec le robot. C'est la raison pour laquelle le tableau de bord MyViz de cette activité ne comporte que des éléments de visualisation mais pas de pilotage : cela a un sens de lire (et éventuellement représenter) des valeurs de signaux sur de multiples clients, mais cela n'en aurait aucun d'autoriser de multiples clients. Envoyer des ordres de pilotages car ceux-ci pourraient être contradictoires. Dans cette activité, seule l'API a le droit d'envoyer des commandes de pilotage au robot.





Par rapport au tableau de bord de pilotage cartésien avec génération de signal tous les éléments de pilotage ont été supprimés, à l'exception de la possibilité de désactiver le driver des moteurs, par sécurité.

D'autre part, le bouton d'initialisation et l'interrupteur d'activation des consignes ne sont également plus présents : ils sont remplacés par des fonctions Python spécifiques.

Enfin, un éditeur Python fait son apparition. Attention, ce n'est pas un interpréteur mais un éditeur : on peut écrire du code Python ou charger un script existant mais on ne peut pas faire de l'exécution ligne par ligne. Un clic sur le bouton lien « Ouvrir » au-dessus de l'éditeur permet de charger des scripts de démonstration.

Attention : le robot en mode « simulation » ne peut utiliser que les scripts suffixés par « simulation ». Le robot en mode « réel » ne peut utiliser que les autres (ceux non suffixés par « simulation »).

Une fois le robot démarré, un lien « Exécuter » fait son apparition au-dessus de l'éditeur. Cliquer sur ce lien pour démarrer l'exécution du script chargé dans l'éditeur. Lorsque le script est en cours d'exécution, il faut d'abord arrêter le script (en cliquant sur le lien « Stop ») avant de pouvoir arrêter le robot.

La zone « Console » sous l'éditeur permet d'afficher des messages d'état internes ainsi que toutes les sorties des commandes « print » utilisées dans votre script.

## Fonction de l'API

L'API de pilotage du Delta2D en Python (« Delta2D \_API ») se trouve dans le fichier API.py, dans le répertoire « Python » de l'archive.

Elle contient les fonctions suivantes :

**Delta2D\_API(mode)**

- Paramètre d'entrée :
  - mode : chaîne de caractères. 'reel' : travail sur le vrai robot, 'simulation' : exécution du modèle de simulation  
Valeur par défaut : 'reel'
- Description : fonction d'initialisation à exécuter impérativement au début de chaque programme

**Reglages(vmax, Kpp, Kip, Kdp)**

- Paramètres d'entrée :
  - vmax (optionnel) : réel, vitesse maximale autorisée des axes, en rad/s (compris entre 0.1 et 1.9)  
Valeur par défaut : 1.9 rad/s
  - Kpp (optionnel : réel, gain proportionnel de la boucle de position (compris entre 0 et 20)  
Valeur par défaut : 5
  - Kip (optionnel : réel, gain intégral de la boucle de position (compris entre 0 et 20)  
Valeur par défaut : 0
  - Kdp (optionnel : réel, gain dérivé de la boucle de position (compris entre 0 et 0.1)  
Valeur par défaut : 0
- Description : cette fonction permet de modifier les principaux réglages des asservissements

### ModeXY()

- Description : cette fonction permet de définir un pilotage en mode cartésien. Il est impératif de l'appeler avant d'exécuter la commande PositionXY() (voir plus loin)

### ModeAngles()

- Description : cette fonction permet de définir un pilotage en mode articulaire. Il est impératif de l'appeler avant d'exécuter la commande Angles() (voir plus loin)

### Initialiser()

- Description : cette fonction initialise les bras en position horizontale (consignes angulaires nulles). Elle doit être appelée au début de l'exécution de chaque programme.

### PositionXY(xcons, ycons, duree)

- Paramètres d'entrée :
  - xcons : réel ou chaîne de caractères (expression Python valide du type '-35 + 100 \* sin(t)', t étant reconnu comme le temps courant).  
Consigne de position cartésienne en x (mm)
  - ycons : réel ou chaîne de caractères (expression Python valide du type '340 + 40 \* sin(t)', t étant reconnu comme le temps courant).  
Consigne de position cartésienne en y (mm)
  - duree (optionnel) : réel. Durée de la consigne en secondes.  
Le programme est bloqué tant que cette durée n'est pas écoulée.  
Valeur par défaut : 0
- Description : cette fonction donne une consigne cartésienne au robot.
- Remarques :
  - xcons est saturé en interne entre -295 et 240 mm
  - ycons est saturé en interne entre 250 et 425 mm

### DesactivationConsignes()

- Description : cette fonction désactive les consignes de position. Noter qu'elles sont automatiquement réactivées si une consigne est demandée après l'exécution de cette fonction. Cette dernière est par conséquent plutôt destinée à être appelée en fin de programme. Elle permet également de redémarrer d'arrêter la capture des signaux et de la redémarrer grâce à l'envoi d'une consigne.

### couleur = LectureCouleur()

- Paramètre de sortie :
  - couleur : entier. Couleur détectée (0 : indéfinie, 1 : rouge, 2 : vert, 3 : bleu, 4 : jaune)
- Description : cette fonction active le capteur de couleur de l'effecteur et renvoie la couleur lue
- Remarque : le capteur de couleur se trouve à la position (25, 0) mm par rapport à la pince

### Angles(alpha1\_cons, alpha2\_cons, duree)

- Paramètres d'entrée :
  - alpha1\_cons : réel ou chaîne de caractères (expression Python valide du type ' $28 * \sin(t)$ ', t étant reconnu comme le temps courant).  
Consigne de position angulaire du bras droit (deg)
  - alpha2\_cons : réel ou chaîne de caractères (expression Python valide du type ' $28 * \sin(t)$ ', t étant reconnu comme le temps courant).  
Consigne de position angulaire du bras gauche (deg)
  - duree (optionnel) : réel. Durée de la consigne en secondes.  
Le programme est bloqué tant que cette durée n'est pas écoulée.  
Valeur par défaut : 0
- Description : cette fonction donne une consigne de position angulaire aux deux bras motorisés du robot.
- Remarques :
  - alpha1\_cons est saturé en interne entre -28 et 30 deg
  - alpha2\_cons est saturé en interne entre -28 et 30 deg



### **mesure\_ecartement = EcarterPince(consigne\_ecartement)**

- Paramètre d'entrée :
  - consigne\_ecartement : entier. Consigne d'écartement des mâchoires de la pince, en mm (compris entre 0 et 32)
- Paramètre de sortie :
  - mesure\_ecartement : entier. Mesure de l'écartement des mâchoires de la pince, en mm
- Description : cette fonction actionne la pince et renvoie la mesure de l'écartement de ses mâchoires

### **valeur = LireVariable(variable)**

- Paramètre d'entrée :
  - variable : chaîne de caractères , variable à lire, parmi la liste suivante :
    - Temps : temps courant (s)
    - xcons : consigne de position cartésienne en x (mm)
    - ycons : consigne de position cartésienne en y (mm)
    - xcalc : position cartésienne en x calculée à partir du modèle cinématique direct (mm)
    - ycalc : position cartésienne en y calculée à partir du modèle cinématique direct (mm)
    - x : position cartésienne en x mesurée à partir du capteur de distance (mm)
    - y : position cartésienne en y mesurée à partir du capteur de distance (mm)
    - alpharef1 : consigne de position angulaire du bras droit (deg)
    - alpharef2 : consigne de position angulaire du bras gauche (deg)
    - alpha1 : position angulaire du bras gauche (deg)
    - alpha2 : position angulaire du bras gauche (deg)
    - alpha3 : position angulaire du coude gauche (deg)
    - alpha4 : position angulaire du coude gauche (deg)
    - wref1 : vitesse de consigne de la boucle d'asservissement en vitesse du moteur droit (rad/s)
    - w1 : vitesse de rotation du moteur droit (rad/s)
    - w2 : vitesse de rotation du moteur droit (rad/s)
    - i1 : courant mesuré dans le moteur droit (A)

- i2 : courant mesuré dans le moteur gauche (A)
- u1 : tension de commande du moteur droit (V)
- u2 : tension de commande du moteur gauche (V)
- ouverture\_pince : écartement des mâchoires de la pince (mm)
- temperature : température du servomoteur de la pince (°C)
- fin\_initialisation : état de l'opération d'initialisation :
  - 0 : l'initialisation n'a pas encore démarré
  - 1 : l'initialisation est en cours
  - 2 : la temporisation de deux secondes est en cours
  - 3 : l'initialisation est terminée, tous les offsets sont calculés et le robot est maintenant pilotable
- Paramètre de sortie :
  - valeur : entier ou réel. Valeur de la variable
- Description : cette fonction permet de lire la valeur d'une variable du système