



# FANUC

ENSAM

OLYMPIADE FANUC 2018



Proposition de réalisation – Olympiade FANUC 2018

## Contenu

Présentation et motivation pour le projet .....	3
Objectif .....	3
Etude du préhenseur .....	4
Implantation du robot .....	6
Choix du robot et positionnement .....	6
Temps entre chaque point de la ligne passé par le robot .....	8
Organisation de la ligne de production – RobotGuide .....	9
Le préhenseur : .....	10
La machine d'usinage : .....	10
Les machines de mesures de balourd : .....	11
Le bac de graissage : .....	12
Les grilles de protections : .....	12
Les convoyeurs de rebuts, de sortie et d'entrée : .....	13
La vision : .....	13
Définir la commande du robot .....	14
Considération nécessaire à la programmation .....	15
Initialisation .....	15
Raisonnements principaux .....	16
Notre solution de commande .....	16
Analyse financière .....	17
Etude financière .....	17
Approximation des coûts directs .....	18
Approximation des coûts indirects .....	18
Analyse des risques .....	19
Objet .....	19
Domaine d'application .....	19
Abréviations .....	20
Synoptique du process .....	21
Protections contre les risques possibles .....	21
Trappe d'observation .....	22
Refuge .....	22
Arrêt d'urgence .....	22
Choix et application des dispositifs appliqués à notre système .....	22

Annexes .....	24
Programme de régression circulaire .....	24
Programme d'optimisation des actions du robot .....	26
Prix du préhenseur .....	34
Choix et prix des convoyeurs.....	34
Convoyeur d'entrée:.....	34
Convoyeur de rebuts et de sortie:.....	35
Prix des baies + Portes.....	37

## Présentation et motivation pour le projet

L'implantation de robot sur les chaînes de production dans les industries est devenue un enjeu majeur pour les usines du futur. Dans toute production, il est nécessaire d'allier efficacité, sécurité et rentabilité. L'utilisation de bras robotiques s'avère être une solution optimale pour répondre à ces enjeux.

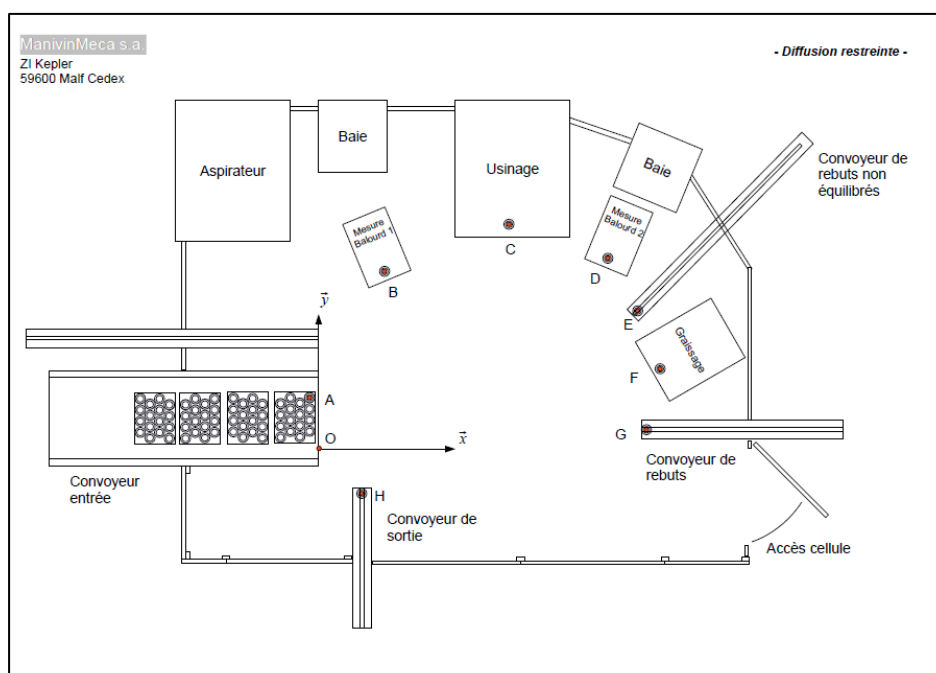
En temps qu'étudiants en formation ingénieur aux Arts et Métiers, la participation à cette épreuve nous permet de découvrir et de nous familiariser avec des outils largement utilisés dans l'industrie robotique. Nous avons ainsi pris conscience de certaines problématiques liées à la mise en place d'un robot sur une chaîne de production : son implantation, les coûts générés, l'impact sur le temps de production, l'optimisation des programmes pour améliorer les performances, la prise en compte de la sécurité et de l'intervention d'opérateur...

C'est dans une démarche d'enrichissement personnel et professionnel que nous avons mené notre projet et que nous avons cherché à répondre au défi proposé par l'entreprise FANUC.

## Objectif

Une ligne de rectification adaptée pour l'accueil d'un robot nous est proposée. Nos objectifs ont donc été de :

- Concevoir un préhenseur adapté au transport et à l'optimisation de la ligne de rectification
- Choisir et adapter un robot de la gamme FANUC à cette ligne de production
- Etudier l'organisation de la ligne afin de l'optimiser
- Programmer et proposer la ligne en conséquence
- Proposer une étude des coûts
- Mettre en valeur les risques générés par une telle installation et proposer des solutions de préventions et des moyens d'action



## Etude du préhenseur

L'étude du préhenseur fût la première partie de notre projet. Il nous a permis de nous familiariser avec les robots FANUC en nous faisant la main sur le robot mis à notre disposition dans notre centre.

Le projet étant dans un cadre industriel, le coût et donc le temps passé rentre en jeu avec une forte importance. Pour cette raison, nous avons alors fait le choix d'un préhenseur double permettant de porter deux pièces en même temps. Nos pièces étant petites, ce choix est alors bien pensé car il permet de gagner beaucoup de temps de trajet du bras du robot. En effet, le trajet du robot représente une importante partie du temps passé par la pièce dans l'enceinte lors du processus de rectification.

Pour le choix de la technologie des deux pinces du préhenseur nous nous sommes orientés vers le pneumatique, une technologie déjà très connue et largement utilisée en robotique.

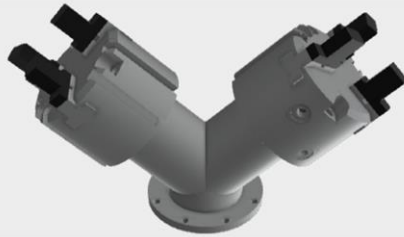
Notre pièce est cylindrique, ainsi, afin de la center il nous faut une pince à 3 mâchoires. Le pignon ayant des dents à l'extérieur il faut donc prendre la pièce par l'intérieur. Nos pignons font 55mm de diamètre interne, il faut alors respecter cette condition pour notre pince qui pourra faire une course inférieure et supérieure à 55mm afin de pouvoir prendre ou lâcher le pignon. Pour ce faire nous avons, dans un premier temps, arbitrairement choisi pour exemple la pince TH5408 de APORE.

Photo du TH5404



Cette pince permet d'avoir une course de 8mm, ce qui est pas négligeable dans notre cas sachant que nous prenons des pièces placées peu précisément dans les bacs. Son poids est de 760g.

Sur cette pince doivent être ajoutés des mors afin de pouvoir prendre le pignon. Nous avons alors modélisé un préhenseur simplifié sous robotguide afin de pouvoir l'utiliser pour notre simulation. Il est composé des mêmes éléments que ceux cités précédemment.



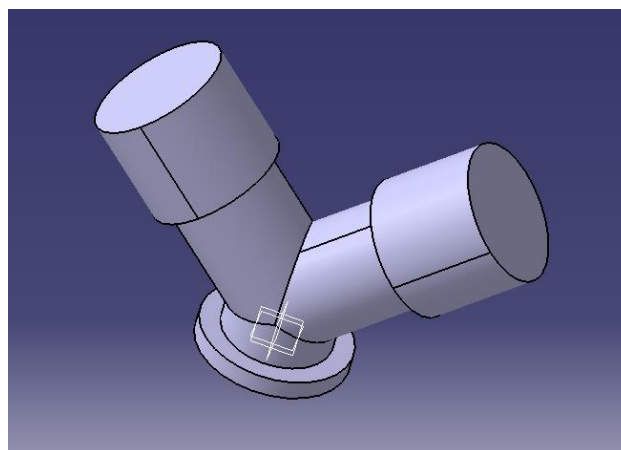
Modèle 3D du préhenseur modélisé.

Sur notre modèle on peut donc bien voir le préhenseur avec les deux pinces ainsi que les 3 mors sur chacune des pinces.

Par la suite, nous sommes allés voir une entreprise qui conçoit elle-même ses chaînes de production ainsi que ses préhenseurs : Duguit technologie. Cette dernière utilise majoritairement les robots fanuc : elle est spécialisée dans la mise en place de chaîne de production d'embouteillage. Dans son cas les préhenseurs sont majoritairement réalisés en impression 3D titane. C'est la solution que nous avons retenue pour différentes répondre à des contraintes de légèreté et de robustesse.

En effet, le titane est très résistant et permet d'optimiser les coûts et d'éviter le remplacement d'un préhenseur abîmé en ne changeant seulement que les mors. De plus, sa légèreté permet une meilleure aisance du robot lors de ses déplacements. Cependant le point noir est son prix élevé. En effet, l'impression titane est encore peu développée et reste chère.

Afin de permettre au robot d'optimiser ses mouvements, nous avons déterminé toutes les caractéristiques physiques de notre préhenseur sur CATIA. Pour cela, nous avons créé un modèle simplifié de celui-ci afin de faire nos applications.



Modèle simplifié de notre préhenseur

Nous avons trouvé les valeurs suivantes :

- Volume :  $3,012 * 10^{-4} m^3$
- Masse volumique Titane :  $4510 kg/m^3$
- Poids du préhenseur : 1,36 kg
- Poids de deux pignons : 0,396 kg
- Poids total de l'ensemble : 1,8 kg
- $I_xG$  :  $0,001 kg/m^2$
- $I_yG$  :  $0,003 kg/m^2$
- $I_zG$  :  $0,002 kg/m^2$

Dans la réalité, il est possible de faire une mesure du Payload automatiquement sur un robot en mesure des forces grâce à des capteurs à des vitesses de déplacement différentes. Dans notre cas, nous sommes en simulation numérique, il est donc impossible de le faire, c'est pourquoi nous l'avons fait d'une autre manière.

## Implantation du robot

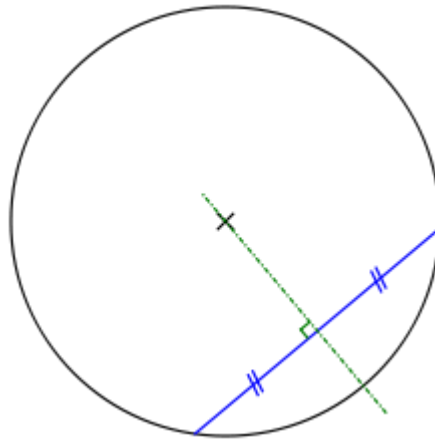
### Choix du robot et positionnement

Nous sommes partis de l'observation suivante : la ligne de production est organisée suivant une disposition circulaire. Afin de choisir un robot poly articulé plusieurs contraintes ont dû être prises en compte :

- Le coût du robot
- Le rayon d'action du robot
- La charge maximale admissible par le robot

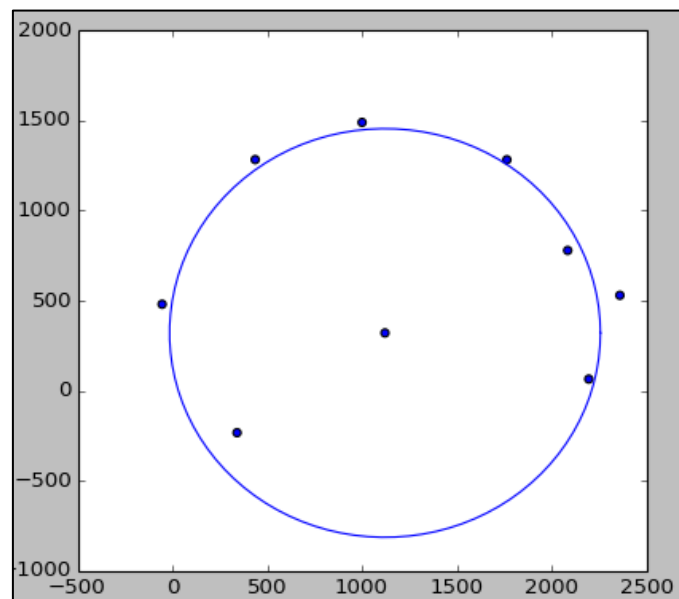
Afin de calculer le rayon d'amplitude du bras articulé il faut positionner le centre optimal de nos 7 points. Pour effectuer cela nous avons utilisé une méthode de régression circulaire particulière : la méthode géométrique de la moyenne des intersections.

Cette méthode consiste à déterminer le point d'intersection de différentes médiatrices dont nous déterminons les équations et que nous exploitons afin de déterminer la position de la moyenne des intersections de nos différentes médiatrices prises deux à deux. On obtient ainsi une position précise du centre de notre cercle optimal.



Nous avons ensuite traduit cette résolution mathématique par un programme sur python (voir le code en Annexe 1) pour déterminer les coordonnées du centre de notre cercle (ici la position du robot) ainsi que le rayon moyen d'action du robot.

En indiquant au programme les coordonnées des postes nous obtenons les coordonnées du point I, future position du robot.



Coordonnées			
Poste	Nom du point	x	y
Prise sur Bac	A	-55	480
Mesure Balourd 1	B	435.7	1283.8
Usinage	C	999	1490.5
Balourd 2	D	1762.3	1283
Rebut non équilibré	E	2083	779
Graissage	F	2358	529.5



Rebut non équilibré	G	2194	64
Sortie	H	340	-234
Point Central	I	1119.7	320.4

De même par cette méthode on a pu déterminer la distance bras robot moyenne (qui correspond au rayon de notre cercle) calculé par notre programme python:

On trouve ici  $R_{moy} = 1256 \text{ mm}$

On choisit un robot de la serie M-20, et plus particulièrement le **M-20iA 12L** qui possède un rayon d'action supérieur à 1256 mm.

En effet :

$$2009\text{mm} > 1256\text{mm}$$

De plus on a la charges maximale admissible par ce robot étant de 12kg et on a

Éléments transportés :	2 Pignons	Préhenseur en titane (le plus léger)	Total
Poids (kg) :	0.198*2=0.4kg	1,36 kg	1.8 kg

On a bien :

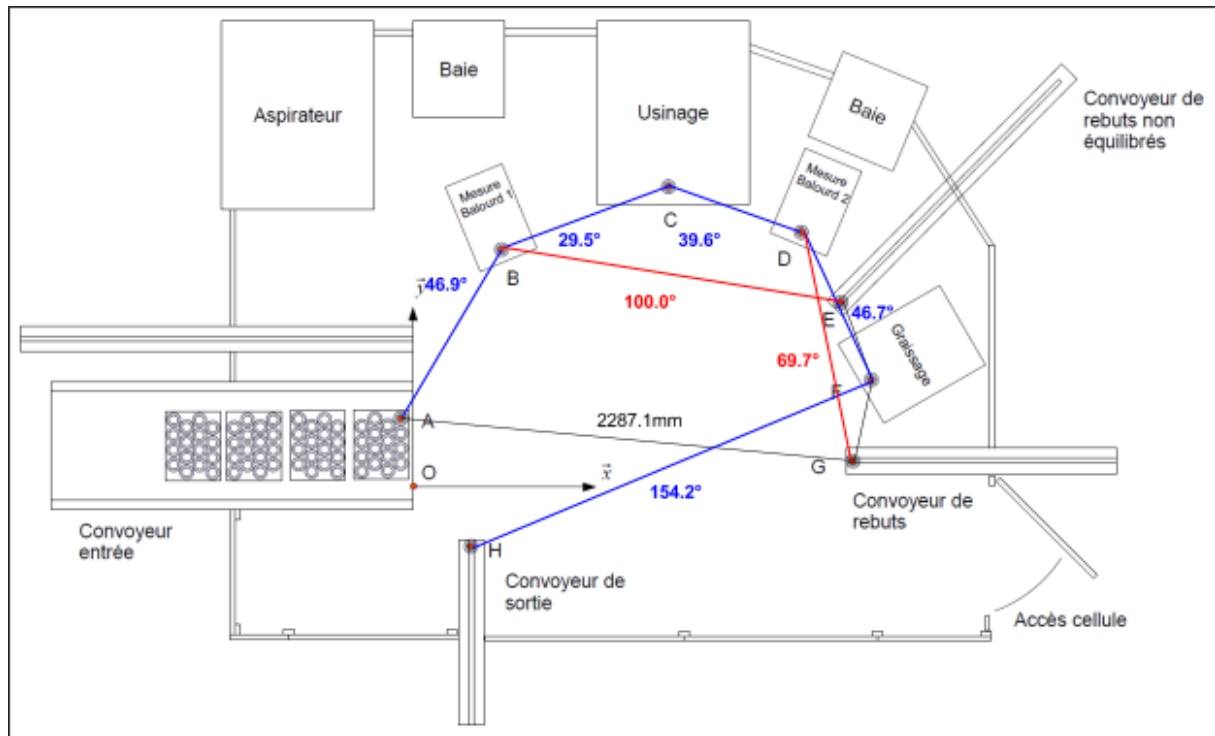
$$1.8\text{kg} < 12\text{kg}$$

Robot			Contrôleur				Capacité de charge max. admissible au poignet (kg)	Rayon (mm)	Axes	Répétabilité (mm)	Masse unité mécanique (kg)	
Série	Version	Type	Version	Type d'armoire								
				Open Air	Mate	A						B
M-20	iA	12L	R-30iB	-	o	●	o	12	2009	6	± 0.04**	250

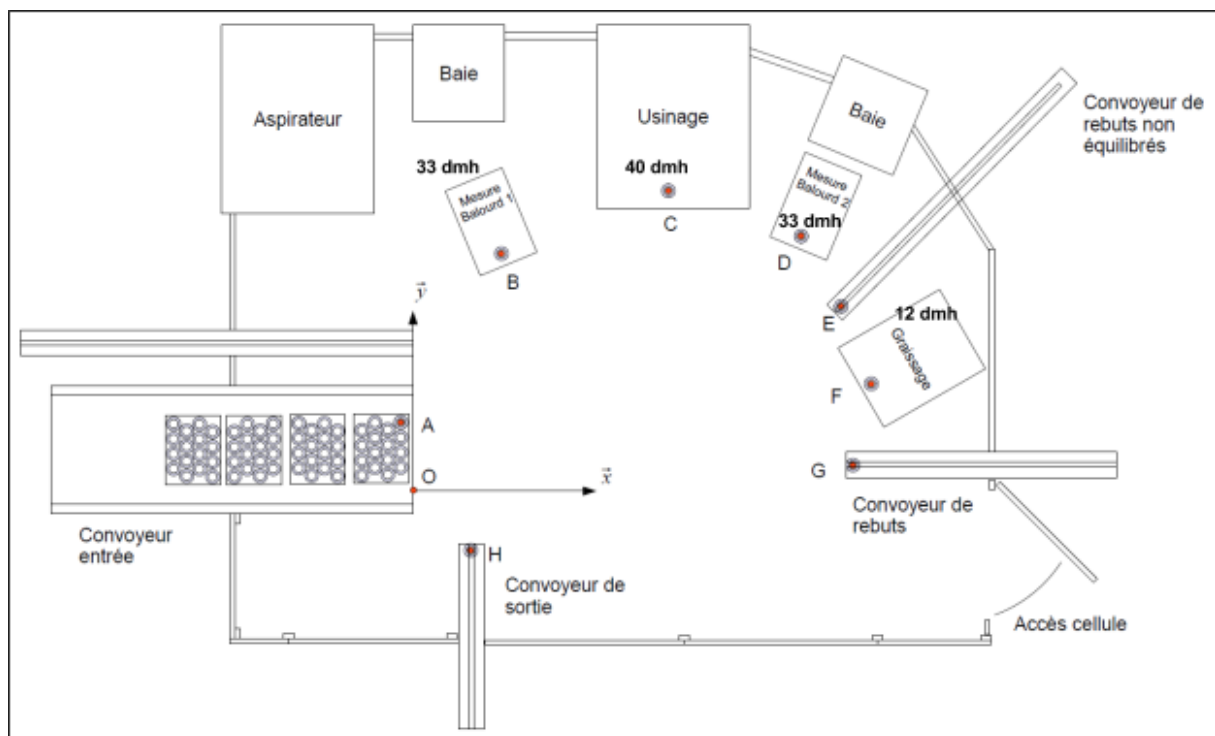
## Temps entre chaque point de la ligne passé par le robot

Grâce aux coordonnées des points trouvés précédemment, nous déduisons à l'aide d'une feuille de calcul Excel, l'angle formé entre tous les postes avec pour origine la position du robot.

On présente sur l'image ci-dessous, les angles utiles pour le déplacement d'une roue dentée provenant du convoyeur d'entrée et sortant au convoyeur de sortie si les mesures sont correctes. Ces distances sont relativement intéressantes car elles permettent de déduire une approximation du temps relatif de déplacement du bras entre deux postes. Si le robot avance à sa vitesse maximale entre chaque poste, il peut atteindre des délais de quelques dmH à peine, on comprend bien que l'approche augmente sérieusement ce délai.



Nous disposons en parallèle dans le cahier des charges de la durée de travail de chaque poste, résumé sur l'image ci-dessous.



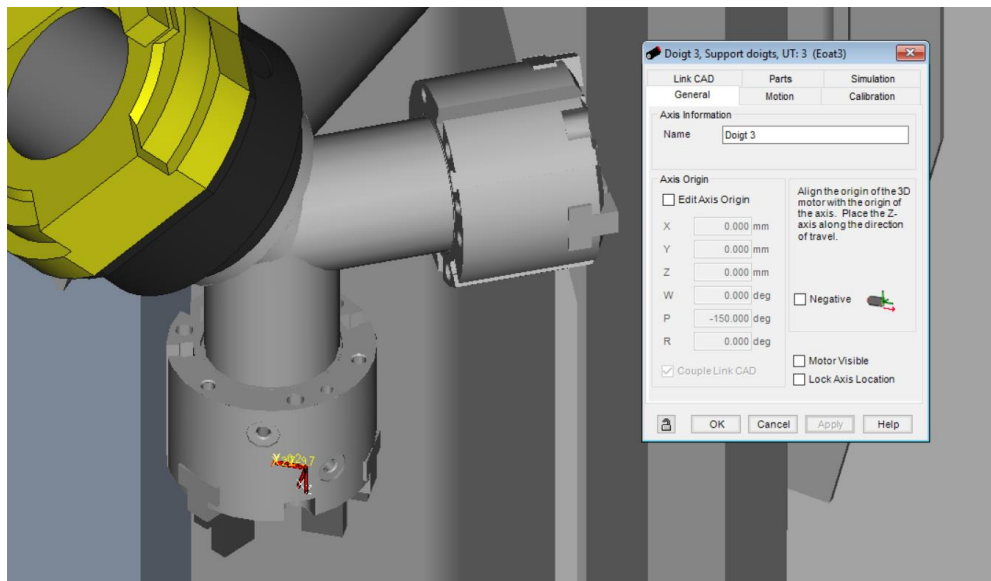
## Organisation de la ligne de production – RobotGuide

Nous avons modélisé notre cellule roboguide en faisant différents choix qui nous permettaient d'avoir une cohérence entre notre modèle roboguide et les choix réalisés.

### Le préhenseur :

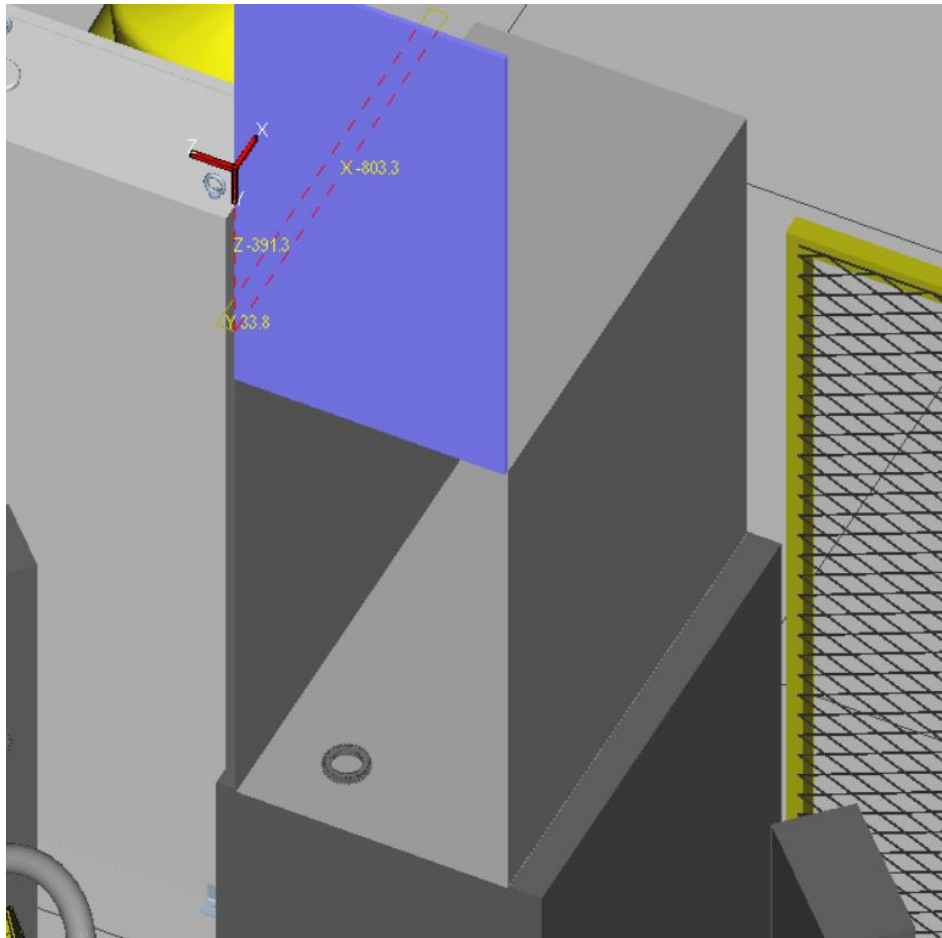
Nous avons tenté d'exporter notre modèle Catia sur roboguide, cependant, il n'était pas possible d'animer les pinces de notre préhenseur, nous avons donc refait un modèle à l'aide du logiciel de création de préhenseur interne à roboguide en lui retirant ses pinces. Nous avons ensuite intégré 6 pinces (3 pour chacun des bras de notre préhenseur) auxquelles nous avons affectés un Do pour modéliser leurs déplacements radiaux. Il est ainsi possible de voir visuellement l'ouverture et la fermeture de de notre pince.

Il a ensuite été nécessaire de définir 2 outils à partir de notre préhenseur. Cependant, nous avons afin de faciliter nos changements d'outils décidé de changer nos repères outils, nous avons ainsi pu éviter de refaire plusieurs fois les mêmes programmes, cependant, un des repère outil est de ce fait moins judicieux.



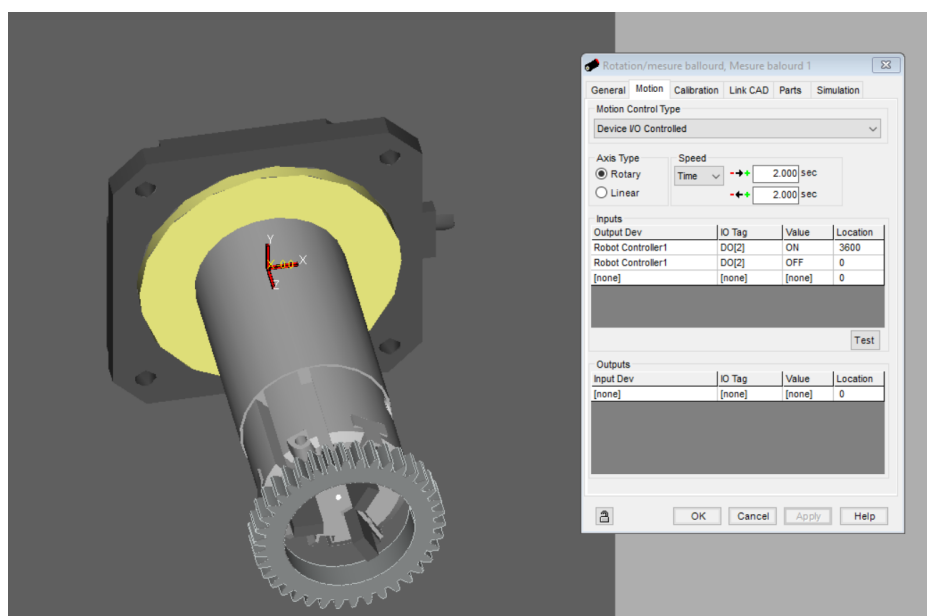
### La machine d'usinage :

Nous avons décidé de modéliser notre machine d'usinage comme un conteneur qui peut être ouvert ou fermé à l'aide d'une porte fictive, cette dernière étant animé à l'aide d'un link : cela modélisant la porte de la machine d'usinage : l'usinage ayant lieu lorsqu'une pièce se trouve à l'intérieur et que la porte est fermée.



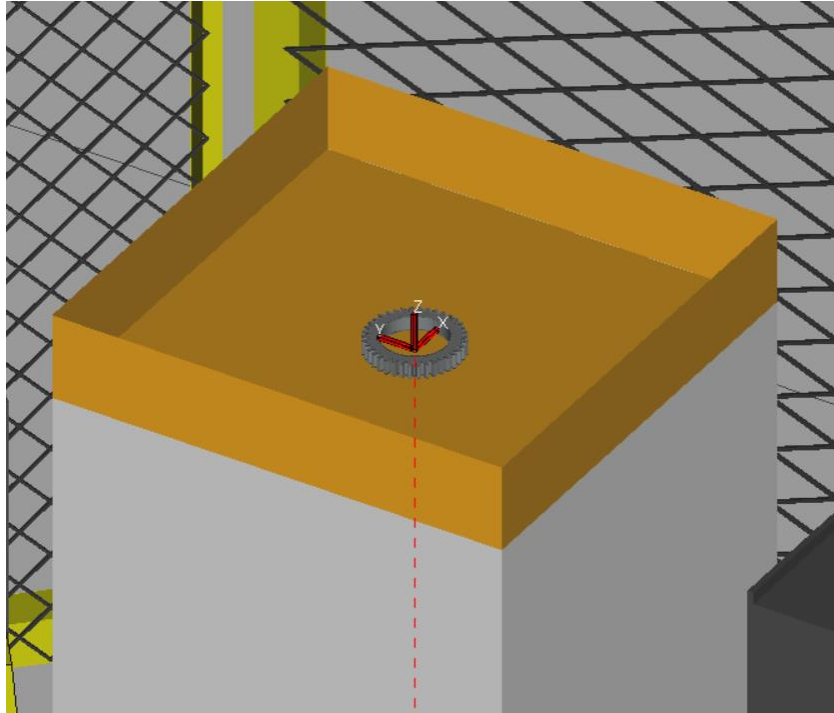
### Les machines de mesures de balourd :

La recherche des machines de balourds s'est révélé être difficile, nous avons donc modélisées ces dernières comme des cylindres tournant autour de leurs axes principaux lors des mesures. On fixe les pignons sur ces cylindres à l'aide d'un préhenseur fixé sur la machine de mesure de balourd adapté aux dimensions de nos pignons.



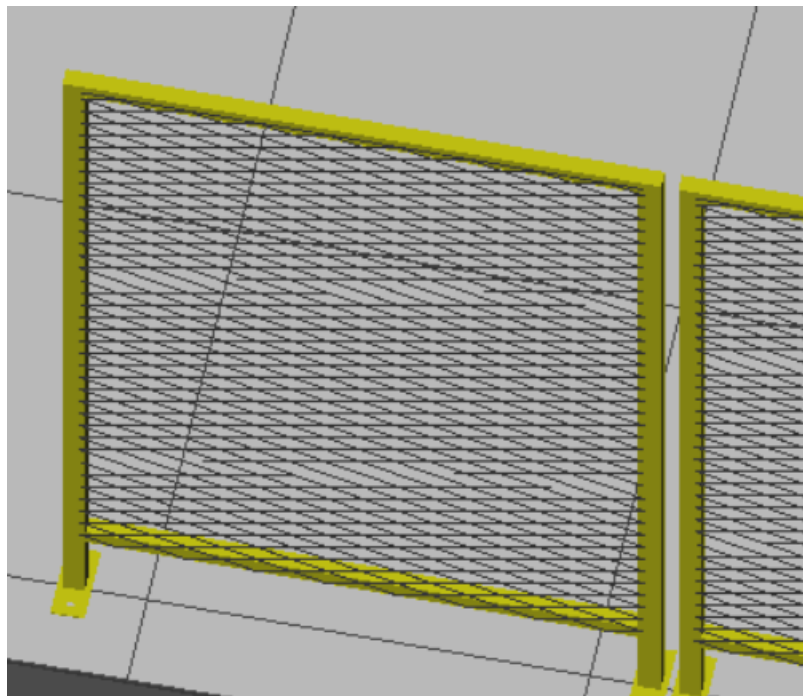
### Le bac de graissage :

Le bac de graissage a été modélisé comme un simple bac dans lesquels on laisse tremper nos pièces, cependant, cette aurait très bien pu être une machine avec un tuyau arrosant notre pièce d'huile de graissage, nous avons donc décidé de laisser nos pièces dans le bac un certain temps au lieu de simplement la plonger afin de ne pas écarter cette possibilité.



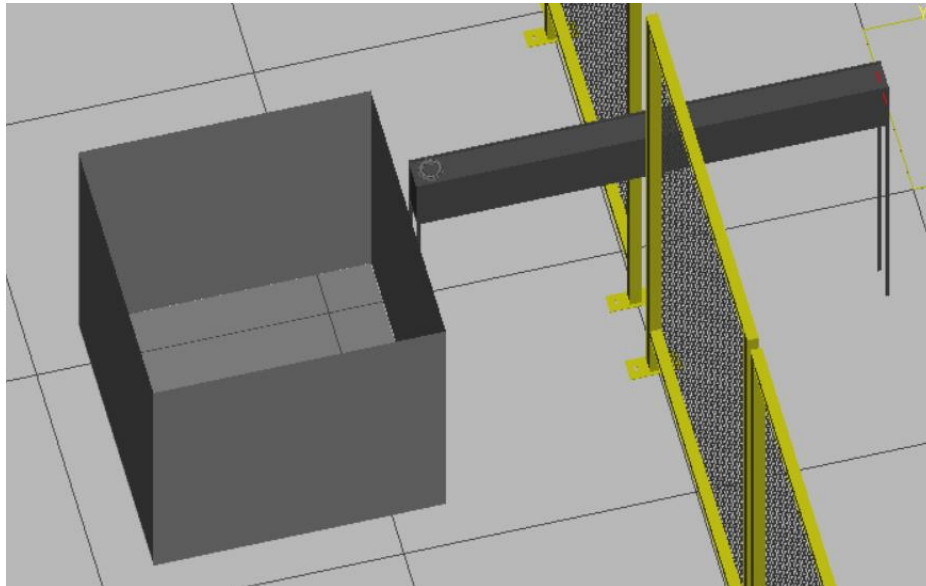
### Les grilles de protections :

Les grilles de protections ont été récupérées directement dans la bibliothèque de fanuc, nous les avons simplement redimensionnées afin qu'elles conviennent avec les dimensions de notre cellule.



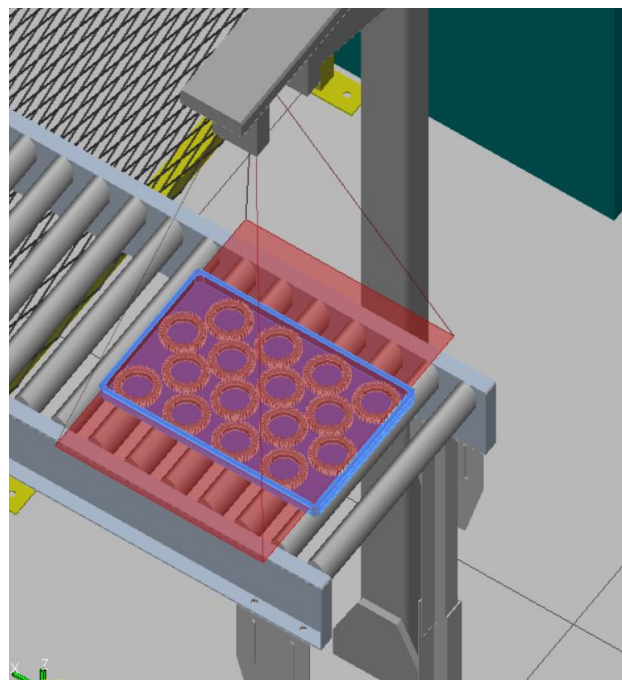
### Les convoyeurs de rebuts, de sortie et d'entrée :

Les convoyeurs ont été redimensionnés afin de pouvoir accueillir ; le bac d'entrée pour le convoyeur d'entrée et simplement un pignon pour les autres convoyeurs. Nous avons aussi décidé de les animer afin de modéliser le déplacement du pignon sur les différents convoyeurs.



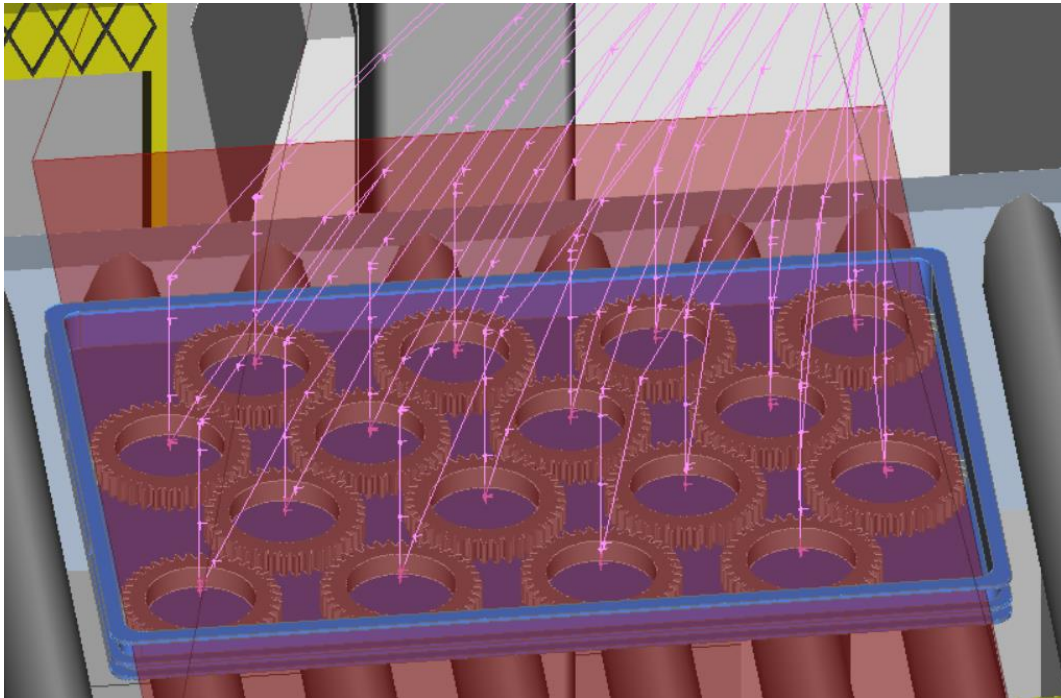
### La vision :

Nous avons tenté de récupérer les pignons du bac d'entrée à l'aide de l'outil de vision de roboguide en définissant un offset de vision par rapport à une position de référence d'un des pignons. Cependant, après moult essais, nous n'avons pas réussi à obtenir de bons offsets, ceux renvoyés par notre cellule de vision étant différentes de celles de notre programme TP de vision (nous avons pourtant bien alignés notre repère avec celui de la plaque de calibration, défini les bonnes échelles de repères, bien orienté la caméra et bien défini nos points de repères mais rien n'y fait.)



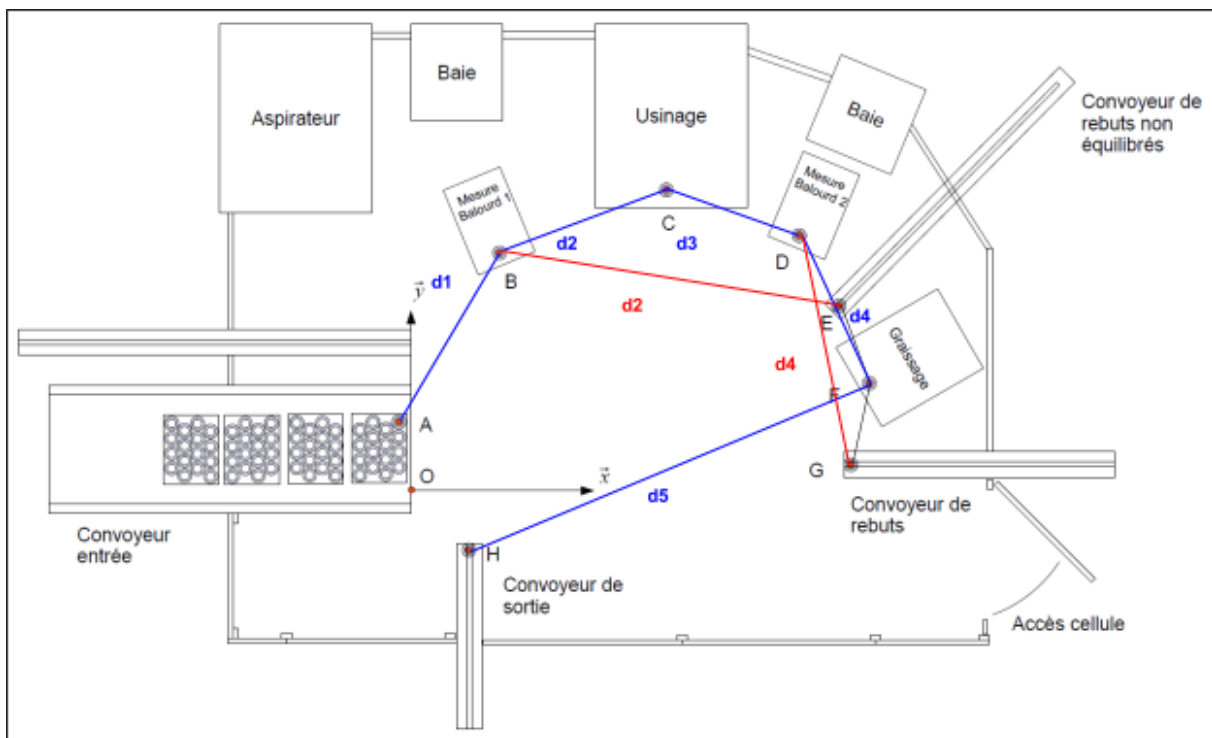


Nous avons donc réalisé la prise des pignons dans le bac d'entrée manuellement à l'aide d'offsets en considérant alors que ces derniers sont bien positionnés dans le bac et que le bac est lui-même.



### Définir la commande du robot

Le cahier des charges impose le trajet des roues dentées au travers de chaque poste. En considérant que les roues n'ont aucun défaut elles auront le trajet idéal en bleu, si la pièce a un défaut elle prend le trajet rouge :



Le bras articulé gère 16 pièces par bac, il dispose d'une liberté totale dans l'ordre des actions effectuées mais celle-ci ont un impact direct sur la durée. C'est pourquoi nous avons décidé de simuler informatiquement la prise et la pose de chacune des roues afin de trouver un trajet de bras articulé optimal.

### Considération nécessaire à la programmation

Nous avons décidé d'effectuer une modélisation qui se met à la place du robot et se demande à chaque instant qu'elle est le choix le plus optimal de la prochaine action à effectuer.

L'idée étant donc de simuler tous les déplacements de roues à l'aide de matrices, et de calculer en parallèle le temps nécessaire à l'exécution de la liste de commande générée par le programme. Dès que le robot dispose de plusieurs choix possibles, il prend une décision de manière aléatoire.

Chaque simulation génère une liste de commande à suivre, la visualisation sous forme de matrice de la position des roues dentées à chaque instant et de ce que tient dans ses pinces le préhenseur à chaque instant. C'est bien la durée globale de chaque simulation que nous visons à minimiser, on prendra alors la simulation de durée la plus faible.

Pour pouvoir comparer les décisions du robot entre chaque simulation il est nécessaire que le robot soit soumis pour chaque simulation exactement à la même demande. Nous avons donc pris la décision de considérer toutes les pièces comme sans défaut.

Les pièces seront également considérées sans défaut dans robot guide puisque la commande du robot est issue de ce programme.

L'utilisation d'un programme informatique à des fins de simulation est d'autant plus intéressant que notre bras articulé dispose de deux préhenseurs ce qui complexifie grandement les choix de commandes possibles.

### Initialisation

Nous avons indiqué au programme le délai de déplacement entre chaque poste, ces délais sont résumés dans le tableau ci-dessous :

Temps de trajet (arondis en dmH)	A	B	C	D	E	F	G	H
A	0	2	2	3	4	4	4	1
B	2	0	1	2	3	3	3	2
C	2	1	0	1	2	2	3	3
D	3	2	1	0	1	1	2	4
E	4	3	2	1	0	1	1	4
F	4	3	2	1	1	0	1	4
G	4	3	3	2	1	1	0	3
H	1	2	3	4	4	4	3	0

Ils sont calculés à partir des angles entre les postes et les caractéristiques techniques du robot.



Nous avons ajouté à cela un délai d'approche associé à chaque poste qui considéré identique pour une prise ou une pause de roue. Afin de faciliter la simulation cette valeur a été fixé à 5dmH.

### Raisonnements principaux

En bref le robot dispose de deux actions principales :

- Prendre une roue dentée
- Déposer une roue dentée

Lorsque l'on souhaite déposer une roue dentée il est nécessaire que la machines la recevant soit libre.

Avant de prendre une roue dentée on s'assure qu'on sera en mesure de la déposer par la suite. Pour cela on simule chaque prise de roue dentée et on s'assure que le robot n'est pas bloqué avec deux roues dentées qui ne peuvent pas être posé.

Le robot favorise la dépose de roue dentée dès que celle-ci est possible.

S'il n'arrive pas à poser les roues qu'il tient dans ses préhenseurs, il en prend une nouvelle roue afin de libérer les machines.

Lorsque le robot ne porte aucune roue dentée sur lui il décide d'en chercher une nouvelle au convoyeur d'entrée.

### Notre solution de commande

Les raisonnements imposés au paragraphe précédent au robot sont perceptibles sur la simulation de commande trouvé.

On ne présentera pas ici la liste des commandes effectuées par le robot car cela n'a pas d'intérêt, on retrouve ces commandes dans l'Excel associé ou bien en regardant la vidéo issue de Robot Guide.

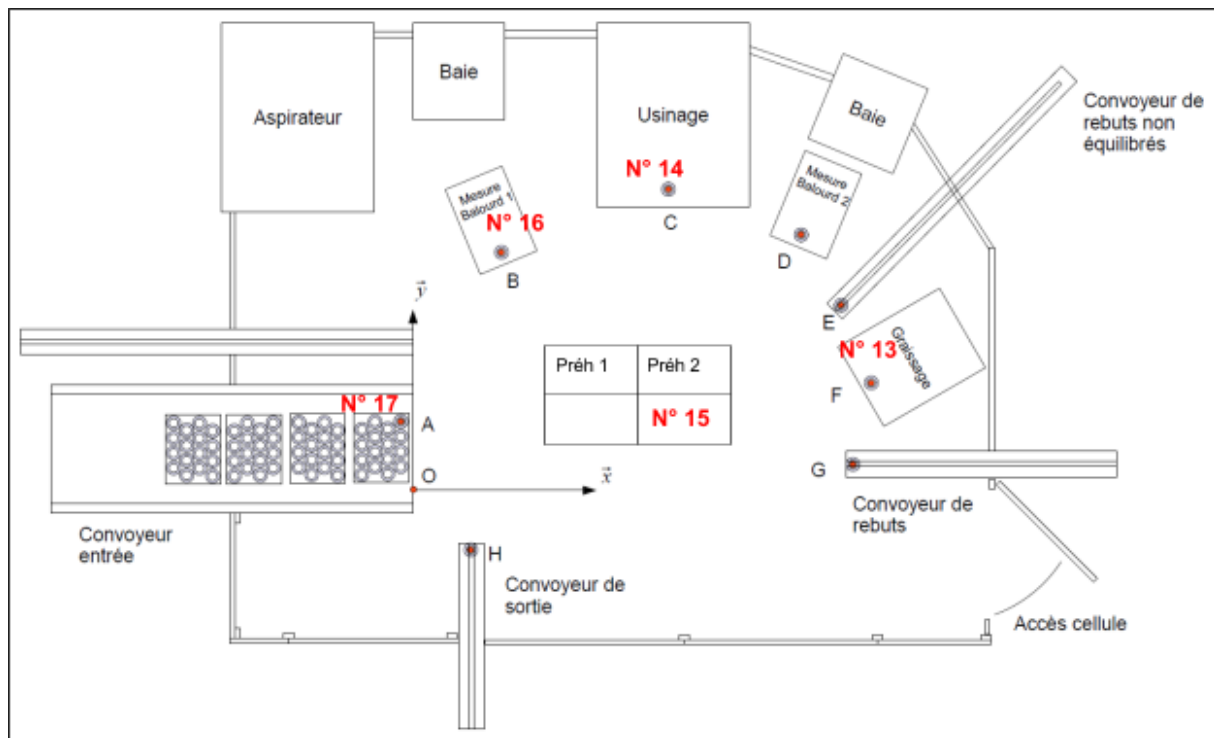
Ce qui est par ailleurs intéressant c'est de visualiser l'occupation des machines et des préhenseur à chaque instant de la simulation.

On appelle instant toutes les nouvelles configurations machines. On associe à chaque poste le numéro de la roue dentée qui s'y trouve.

Ci-dessous sont présentés les 24 derniers instants, ils suffisent à comprendre le raisonnement :

Poste A	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17
Poste B	16	16	16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Poste C	14	0	0	15	15	15	15	15	15	0	0	16	16	16	16	16	0	0	0	0	0	0	0
Poste D	0	0	14	14	14	14	14	0	0	0	15	15	15	15	0	0	0	16	16	16	0	0	0
Poste E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Poste F	13	13	13	13	13	0	0	0	14	14	14	14	0	0	0	15	15	15	0	0	0	16	0
Poste G	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Préh. 1	0	14	0	0	16	16	16	16	16	16	16	0	14	0	15	0	16	0	15	0	16	0	16
Préh. 2	15	15	15	0	0	13	0	14	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0

La configuration surlignée ci-dessous correspond à la situation suivante :



Nous remarquons après avoir finalisé la simulation et importé les résultats dans Robotguide que nos choix de délais sont extrêmement faibles et quasi inaccessible en conditions réelles.

Si on prête minutieusement attention aux décisions prise par la machine, c'est comme si elle considérait son temps de déplacement comme nulle et qu'elle se précipite pour récupérer les pièces dès qu'elles sont prête et vite les déposer dans une autre machine afin d'optimiser les temps machines.

L'implémentation de l'ensemble des commandes dans Roboguide est coûteuse en temps, nous avons décidé de ne pas relancer de simulation avec des paramètres différents.

Le programme utilisé a été programmer sous python est disponible en annexe.

## Analyse financière

### Etude financière

Afin de faire une étude complète de la cellule de rectification, nous avons fait une étude précise des coûts engendrés. Les coûts sont de sources diverses et se divisent en deux types différents : les coûts directs et coûts indirects. Dans les coûts sont pris en compte les différentes étapes qui peuvent avoir lieu pour une pièce comme par exemple la conception, la fabrication ou l'installation.

Le prix du préhenseur comprend le coût du matériau, le coût de la conception mais aussi le coût d'utilisation d'une imprimante 3D titane. Les coûts de fonctionnement électriques sont basés sur les prix de l'électricité EDF.

## Approximation des coûts directs

Coûts Directs			Cout
Robot (ML 20)	.....	.....	37 800,00 €
	Préhenseur	HT.	35 000,00 €
		.....	1 500,00 €
		Conception	500,00 €
	Installation	Fabrication	1 000,00 €
		.....	3 000,00 €
		Personnels	3 000,00 €
Analyseur optique (Effilux)	.....	Transport	????
	.....	.....	2 600,00 €
	.....	.....	1 600,00 €
Convoyeur entrée	.....	.....	1 000,00 €
	.....	.....	2 799,00 €
	Cout direct	.....	2 499,00 €
Mesure Balourd 1	Installation	.....	300,00 €
	.....	.....	570,00 €
	Cout	.....	500,00 €
Machine Usinage	Installation	.....	70,00 €
	.....	.....	6 000,00 €
	Cout direct	.....	5 000,00 €
Mesure Balourd 2	Installation	.....	1 000,00 €
	.....	.....	570,00 €
	Cout	.....	500,00 €
Convoyeurs de rebuts non équilibrés	Installation	.....	70,00 €
	.....	.....	1 835,00 €
	Cout	.....	1 635,00 €
Graissage	Installation	.....	200,00 €
	.....	.....	110,00 €
	Cout	.....	100,00 €
Convoyeur de rebut	Installation	.....	10,00 €
	.....	.....	1 835,00 €
	Cout	.....	1 635,00 €
Convoyeur de Sortie	Installation	.....	200,00 €
	.....	.....	1 835,00 €
	Cout	.....	1 635,00 €
Baies + Portes	Installation	.....	200,00 €
	.....	.....	2 180,00 €
	Cout	.....	1 980,00 €
Aspirateur	Installation	.....	200,00 €
	.....	.....	1 300,00 €
	Cout	.....	1 000,00 €
TOTAL	Installation	.....	300,00 €
	.....	.....	58 134,00 €

## Approximation des coûts indirects

Coûts indirects / mois (240 heures de fonctionnement)			Cout
Robot (ML 20)	.....	.....	533,60 €
	Consomation	1 kWh	33,60 €
	Maintenance et contrôle	.....	500,00 €
Analyseur optique (effilux)	négligeable		
Convoyeur entrée	.....	.....	12,43 €
	Consomation	0.37 kWh	12,43 €
	.....	.....	.....
Mesure Balourd 1	négligeable		
Machine Usinage	.....	.....	60,48 €
	Consomation	1.8 kWh	60,48 €
	.....	.....	.....
Mesure Balourd 2	négligeable		
Convoyeurs de rebuts non équilibrés	.....	.....	6,05 €
	Consomation	0.18 kWh	6,05 €
	.....	.....	.....
Graissage	.....	.....	100,00 €
	Consomation Huile	20 L d'huile	100,00 €
	.....	.....	.....
Convoyeur de rebut	.....	.....	6,05 €
	Consomation	0.18 kWh	6,05 €
	.....	.....	.....
Convoyeur de Sortie	.....	.....	6,05 €
	Consomation	0.18 kWh	6,05 €
	.....	.....	.....
Aspirateur	.....	.....	25,20 €
	Consomation	0.750 kWh	25,20 €
	.....	.....	.....
<b>TOTAL</b>			<b>749,86 €</b>

## Analyse des risques

### Objet

Afin de répondre aux exigences réglementaires, une installation d'une chaîne de production doit respecter des normes. L'objet de cette partie est d'évaluer l'ensemble des risques engendrés par l'installation relatifs à la norme ISO 12100 : 2010 et l'EN 13306 : 2010. Nous déterminerons aussi les actions correctives et préventives à mener dans chaque situation.

### Domaine d'application

Cette analyse s'applique à :

- L'installation de la machine : mises au point et tests

- La mise en service : test de production et mise en cadence
- L'utilisation : l'exploitation et la maintenance

Au cours de l'exploitation de la chaîne de production nous distinguerons :

- Les événements prévus
- Les événements non prévus

Au cours des opérations de maintenance, nous distinguerons :

- Les actions préventives
- Les actions correctives
- Les actions majeures

Cette analyse s'applique en interne (opérateurs intervenant sur la chaîne).

### Abréviations

<b>PPA</b>	protections de production actives
<b>PPN</b>	protections de production neutralisées
<b>AP</b>	actions préventives
<b>AC</b>	actions correctives

## Synoptique du process

N°	Qui	Quoi - étapes	Problèmes	Actions
1	Convoyeur	Transport des pignons en entrée	Caisses mal positionnées / Blocage	Capteurs / Glissières de repositionnement
2	Machine de mesure	Mesure de balourd n°1	Pannes / Dysfonctionnements	Contrôles de maintenance / Observations
3		Conforme ?		
4		Evacuation sur convoyeur de rebuts équilibrés	Obstacles / Bourrage / Blocage	Capteurs / Glissières de repositionnement
5	Machine d'usinage	Usinage	Pannes / Dysfonctionnements / Bourrage matière	Contrôles de maintenance / Observations
6	Machine de mesure	Mesure de balourd n°2	Pannes / Dysfonctionnements / Bourrage matière	Contrôles de maintenance / Observations
7		Conforme?		
8	Convoyeur	Evacuation des rebuts	Obstacles / Bourrage / Blocage	Capteurs / Glissières de repositionnement
9	Machine de graissage	Graissage	Pannes / Dysfonctionnements / Bourrage matière	Contrôles de maintenance / Observations
10	Convoyeur	Transport vers la sortie	Caisses mal positionnées / Blocage	Capteurs / Glissières de repositionnement

## Protections contre les risques possibles

Chaque étape de production sera appelée par son numéro qui lui est associé dans la synoptique proposé ci-dessus.

### Trappe d'observation

Toute trappe d'observation doit respecter les dimensions indiquées par la norme ISO 13857. L'utilisation d'une telle trappe ne doit se faire que dans un temps limité avec le port des EPI. Elle sert à l'observation en PPN, et peut éventuellement permettre d'effectuer des mesures sur machine par passage d'appareil de mesure léger par la trappe.

Normes en vigueur
ISO 13 857

### Utilité pour étape n°2, 5, 6 et 9

#### Refuge

Un refuge doit être une zone dans laquelle un opérateur peut observer le processus de production en cours de fonctionnement. Cette zone et son accès sont réglementés par les normes ISO 13855 et ISO 13857.

#### Accès en zone avec arrêt des fonctions dangereuses :

L'accès au refuge doit se faire dans un mode au cours duquel les zones dangereuses des machines sont désactivées. Une fois que l'opérateur est entré dans la zone, la mise en route du processus ne doit pas pouvoir se faire depuis l'extérieur. Arrivé dans le refuge, l'opérateur doit pouvoir remettre en route le processus en activant un bouton (maintien du bouton), par fermeture du sas de protection. En cas de sortie de l'éloignement de l'opérateur (sortie de zone ou relâchement du bouton), l'arrêt des fonctions dangereuses de la machine doit être immédiat. Des signaux d'avertissement doivent être actifs pendant toute la durée d'accès à la zone pour éviter qu'un nouvel opérateur ne s'introduise intempestivement en PPA.

#### Accès en zone sans arrêt :

La zone doit être accessible facilement dans une zone normée par ISO 11161.

### Utilité pour toute étape

Normes en vigueur
ISO 13 855
ISO 13 857
ISO 11 161

#### Arrêt d'urgence

L'arrêt d'urgence doit survenir en complément des autres dispositifs de sécurité mais ne doit pas les remplacer. Un tel arrêt doit se faire en cas extrême. L'opérateur doit pouvoir actionner l'arrêt d'urgence depuis l'intérieur ou l'extérieur de la zone de production

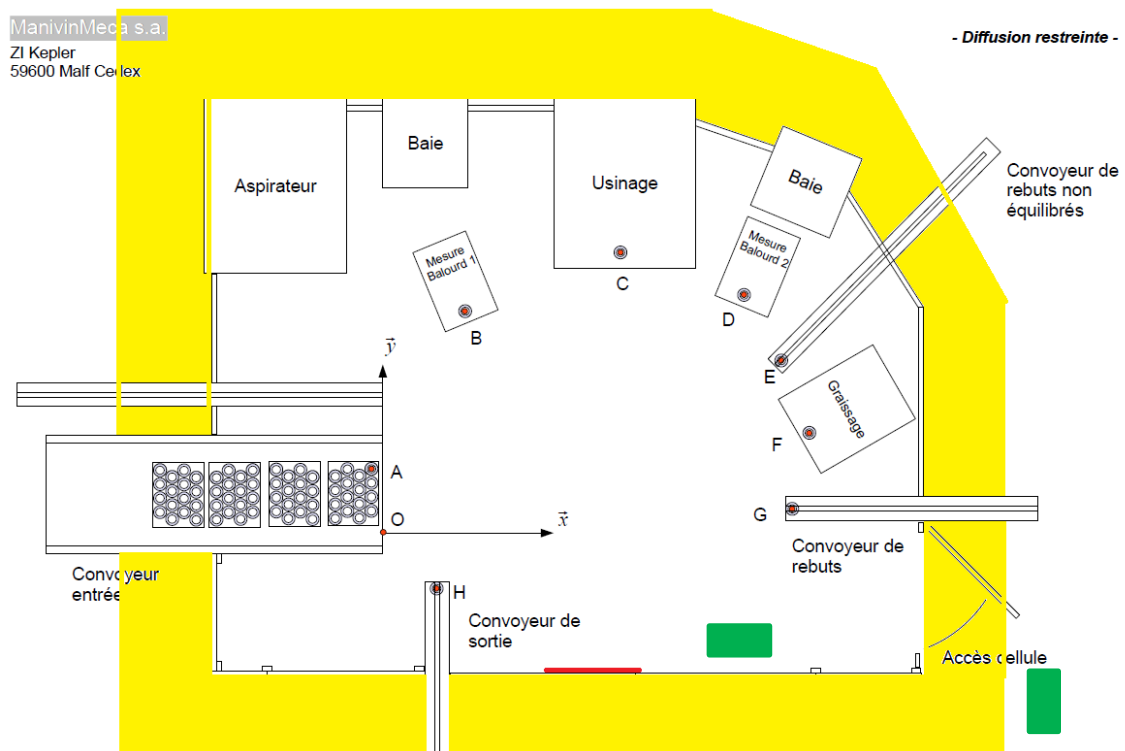
### Choix et application des dispositifs appliqués à notre système

Les dimensions de l'espace de production étant relativement restreinte, la création de zone de refuge n'est pas possible pour cette chaîne. Une trappe d'observation apparaît aussi inutile. La zone

est délimitée par un grillage. Depuis la position repérée en rouge, il est alors possible d'observer l'ensemble du processus et de détecter un dysfonctionnement obligeant un opérateur à intervenir.

Malgré le grillage de sécurité, un périmètre de sécurité doit aussi être prévu (en jaune).

De plus, en cas d'intervention, l'opérateur doit pouvoir accéder rapidement à la zone par la porte prévue à cet effet. Son ouverture arrête l'ensemble de la chaîne par une serrure électromagnétique. L'opérateur doit pouvoir aussi contrôler l'arrêt depuis l'intérieur de la cellule et l'arrêt et la reprise depuis l'extérieur de la cellule grâce à des commandes mises à sa disposition (en vert).





## Annexes

### Programme de régression circulaire

```
from math import *
```

```
"""https://fr.wikipedia.org/wiki/R%C3%A9gression_circulaire#M%C3%A9thode_des_moindres_carr%C3%A9s_totaux"""
```

```
A= (-55, 480)
B= (435.7, 1283.8)
C= (999, 1490.5)
D= (1762.3,1283)
E= (2083, 779)
F= (2358, 529.5)
G= (2194, 64)
H= (340,-234)
```

```
points = []
points.append(A)
points.append(B)
points.append(C)
points.append(D)
points.append(E)
points.append(F)
points.append(G)
```

```
def distance(A1,B1):
    return(((B1[0]-A1[0])**2+(B1[1]-A1[1])**2)**(1/2))
```

```
def carrée(A1,B1,r):
    return((((B1[0]-A1[0])**2+(B1[1]-A1[1])**2)**(1/2)-r)**2)
```

```
def f(x,y,L,r): #pseudo écart-type
    S=0
    M=(x,y)
    for i in range(len(L)):
        S+=carrée(L[i],M,r)
    return(S)
```

```
def wijk(pt1,pt2,pt3):
    (x1,y1)=pt1
    (x2,y2)=pt2
    (x3,y3)=pt3
    return(x1*(y2-y3)+x2*(y3-y1)+x3*(y1-y2))
```

```
def wijkt(pt1,pt2,pt3):
    (x1,y1)=pt1
    (x2,y2)=pt2
    (x3,y3)=pt3
    print((x1,y1),(x2,y2),(x3,y3))
    print(x1**2*(y2-y3)+x2**2*(y3-y1)+x3**2*(y1-y2))
    return(x1**2*(y2-y3)+x2**2*(y3-y1)+x3**2*(y1-y2))
```

```
def Xijkt(pt1,pt2,pt3):
    (x1,y1)=pt1
    (x2,y2)=pt2
    (x3,y3)=pt3
    return((x1-x2)*(x2-x3)*(x3-x1))
```

```
def Zijk(pt1,pt2,pt3):
    (x1,y1)=pt1
    (x2,y2)=pt2
    (x3,y3)=pt3
    return(y1*(x2-x3)+y2*(x3-x1)+y3*(x1-x2))
```

```

def Zijkt(pt1,pt2,pt3):
    (x1,y1)=pt1
    (x2,y2)=pt2
    (x3,y3)=pt3
    return(y1**2*(x2-x3)+y2**2*(x3-x1)+y3**2*(x1-x2))

def Yijkt(pt1,pt2,pt3):
    (x1,y1)=pt1
    (x2,y2)=pt2
    (x3,y3)=pt3
    return((y1-y2)*(y2-y3)*(y3-y1))

def combin(n, k):
    """Nombre de combinaisons de n objets pris k a k"""
    if k > n//2:
        k = n-k
    x = 1
    y = 1
    i = n-k+1
    while i <= n:
        x = (x*i)//y
        y += 1
        i += 1
    return x

L=[A,B,C,D,E,F,G,H]

from matplotlib.pyplot import*
from numpy import*

X=[l[0] for l in L]
Y=[l[1] for l in L]
scatter(X,Y)

def cercle(xc,yc,r):
    theta=linspace(0,2*pi,100)
    x=[xc+r*cos(th)for th in theta]
    y=[yc+r*sin(th)for th in theta]
    plot(x,y)

def cerclemedian(L):
    n=len(L)
    C=combin(n,3)
    xc=0
    yc=0
    r=0
    for i in range(n-2):
        for j in range(i+1,n-1):
            for k in range(j+1,n):
                print(xc)
                print(wijk(L[i],L[j],L[k]))
                xc+=(wijk(L[i],L[j],L[k])-Yijkt(L[i],L[j],L[k]))/wijk(L[i],L[j],L[k])
                yc+=(Zijkt(L[i],L[j],L[k])-Xijkt(L[i],L[j],L[k]))/Zijk(L[i],L[j],L[k])

    xc=1/(2*(C))*xc
    yc=1/(2*(C))*yc
    M=(xc,yc)
    for i in range(n):
        r+=1/n*distance(L[i],M)
    print("xc=",xc)
    print("yc=",yc)
    print("r=",r)
    cercle(xc,yc,r)
    scatter(xc,yc)
    show()

cerclemedian(L)

```

## Programme d'optimisation des actions du robot

```

1 import numpy as np
2 import random
3 import openpyxl
4
5 simulationN = 1
6
7 def initializeVariables():
8     global machines, machinesT, machinesD, machinesH, bras, brasM, brasH, roue,
9     roueH, brasD, brasC, approcheD, lastPosition, lastRoueDone, positions, t, tH
10
11     machines = np.zeros((8,1), int) #Est une matrice qui represente à l'instant t le
12     numéro de la roue dentée possédé par chaque machine
13     machines[0,0]=1 #La première roue dentée arrive sur le convoyeur d'entrée
14     machinesT = np.zeros((8,1), int) #Est une matrice qui représente pour chaque
15     machine le temps depuis lequel la machine est occupé en dmh
16     machinesD = np.array([0,33,40,33,0,12,0,0]).reshape(8,1) #Est une matrice qui
17     contient les durées de traitement de chaque machine
18     machinesH = machines #Est une matrice qui contient l'historique de toutes les
19     matrice "machines"
20
21     bras = np.zeros((2,1), int) #Est une matrice qui contient la roue dentée possédé
22     par chaque préhenseur du bras articulé
23     brasM = np.zeros((2,1), int) #Est une matrice qui contient le numéro de la
24     machine dont provient chaque pièce porté par le bras
25     brasH = bras #Est une matrice qui contient l'historique de toutes les matrices
26     "bras"
27
28     brasD = np.array([[0, 2, 2, 3, 4, 4, 4, 1],
29     [2, 0, 1, 2, 3, 3, 3, 2],
30     [2, 1, 0, 1, 2, 2, 3, 3],
31     [3, 2, 1, 0, 1, 1, 2, 4],
32     [4, 3, 2, 1, 0, 1, 1, 4],
33     [4, 3, 2, 1, 1, 0, 1, 4],
34     [4, 3, 3, 2, 1, 1, 0, 3],
35     [1, 2, 3, 4, 4, 4, 3, 0]]) #Est une matrice qui contient tous les délais de
36     déplacement entre chaque machines
37
38     approcheD = np.ones((8,1), int) #Est une matrice qui contient les délais
39     d'approche, de prise ou pose et de retrait associé à chaque machine
40
41     brasC = np.array([0,1,2,3,5,7,8]) #Est une matrice qui contient la suite des
42     numéros de machine que doit suivre chaque roue dentée
43
44     roue = np.zeros(1, int) #Est une matrice qui associe à chaque roue dentée le
45     numéro de la machine sur laquelle celle-ci se situe
46
47     lastPosition = 0 #Contient le dernier numéro de machine où est passé le bras
48     articulé
49
50     lastRoueDone = 0 #Contient la dernière roue dentée qui est sorti du système
51
52     positions = np.array([[],[],[]]) #Est une matrice qui contient l'historique de
53     toutes les actions effectué par le robot (prise ou pose sur une machine donnée avec
54     un numéro de préhenseur donnée)
55
56     t=0 #Représente le temps écoulé en dmh depuis le départ du robot
57     tH = np.array(t) #Contient l'historique de tous les temps t
58
59
60 def prehLibre():
61     ###Renvoie la position d'un préhenseur libre ou bien la valeur 2
62     global bras
63     a=bras

```

```

50     if a[0,0]==0:
51         b=0
52     elif a[1,0]==0:
53         b=1
54     else:
55         b=2
56     return b
57
58 def prehLibreN():
59     """Renvoie le nombre de préhenseur qui sont libres
60     global bras
61     return np.count_nonzero(bras==0)
62
63 def prise(i,j):
64     """Le bras se déplace du poste j vers le poste i, prend la roue dentée situé au
    poste i et calcul le temps que cela prend
65     global bras
66     global brasM
67     global machines
68     global machineT
69     global roue
70     global positions
71     t = brasD[i,j] + approcheD[i]
72     a = prehLibre()
73     positions = np.append(positions,np.array([[ "prise" ],[i], [a]]), 1)
74     bras[a,0]=machines[i]
75     brasM[a,0]=i
76     #on augmente le numéro de l'écrou si on prend une roue sur le convoyeur d'entree
77     if i==0:
78         machines[0,0]+=1
79         roue=np.append(roue, [0])
80     else:
81         machines[i]=0
82         machinesT[i]=0
83     majMachines(t)
84     return t
85
86 def prendre(i):
87     """La commande de prise est envoyé à la machine (celle ci vérifie que le
    préhenseur possède bien une place de libre et qu'il y ai bien une roue dentée à
    prendre)
88     global lastPosition
89     if prehLibre==2:
90         raise NameError('Préhenseur plein')
91     if machines[i]==0:
92         raise NameError('Rien à prendre ici')
93     return prise(i,lastPosition)
94
95
96 def pose(i,j,k):
97     """Le bras se déplace du poste j vers le poste i, pose la roue dentée porté par
    le préhenseur k au poste i et calcul le temps que cela prend
98     global machines
99     global roue
100    global bras
101    global brasM
102    global lastRoueDone
103    t = brasD[i,j] + approcheD[i]
104    if i==7:
105        machines[i]=0
106        lastRoueDone = bras[k,0]
107    else:
108        machines[i] = bras[k,0]
109
110    roue[bras[k]-1]=i

```

```

111     bras[k,0]=0
112     brasM[k,0]=0
113     majMachines(t)
114     return t
115
116 def poser(i,k):
117     """La commande de pose est envoyé à la machine (celle ci vérifie que le
    préhenseur possède bien une roue dentée à poser et que la machine est bien libre)
118     global lastPosition
119     global positions
120     positions = np.append(positions, np.array([["pose"],[i],[k]]),1)
121     if machines[i] !=0:
122         raise NameError('impossible de poser ici')
123     if bras[k]==0:
124         print(brasH)
125         print(machinesH)
126         raise NameError('Rien à poser ici')
127     return pose(i,lastPosition, k)
128
129
130 def majMachines(t=0):
131     """Met à jour le temps d'occupation de chaque machines
132     global machinesT
133     machinesT = machinesT + (np.zeros((8,1), int) != machines)*t
134
135
136 def machineCible():
137     """Retourne la liste des machines qui sont Ciblenibles
138     return machinesT > machinesD
139
140
141 def etatMachines(roue):
142     """Renvoie la liste des machines avec leur disponibilité
143     vLigne = np.zeros((1,8), int)
144     for i in range(len(roue)):
145         vLigne.put(roue[i],True)
146     return vLigne.reshape((8,1))
147
148
149
150 def correctionTerminaison(b):
151     """On simule ici la prise de chaque pignon afin de s'assurer que le robot ne se
    trouve pas bloqué avec deux pignons qu'il ne peut pas poser
152     global bras, roue, machines, d
153
154     bS=np.copy(b)
155
156     for j in range(len(b)):
157         i= len(b)-1-j
158         brasS=np.copy(bras)
159         roueS=np.copy(roue)
160         machinesS=np.copy(machines)
161
162
163         a = prehLibre()
164         brasS[a,0]=machines[b[i]]
165         machinesS[i] = 0
166         d=prochaineMachinesVides(brasS, roueS, machinesS)[0]
167
168         print(np.array(prochaineMachinesVides(brasS, roueS, machinesS)[0]).shape[1])
169         if np.array(prochaineMachinesVides(brasS, roueS, machinesS)[0]).shape[1]==0:
170
171             bS=np.delete(bS, i, 0)
172
173     print("final")

```

```

174     print(bS)
175
176     return b
177
178
179 def prochaineMachinesCible():
180     """Renvoie la liste des premières machines qui se libèrent
181     global t
182     global brasM
183     i=0
184
185
186     b = np.zeros(0)
187
188     while b.shape[0]<=0:
189         t+=1
190         majMachines(1)
191
192         a = machineCible() * etatMachines(roue)
193         a = np.delete(a,0,0)
194
195
196         b = np.array(a.nonzero()[0])+1
197
198     #On s'assure de bien piocher des pièces qui assure la terminaison du programme
199     if(prehLibreN()==1):
200         b=correctionTerminaison(b)
201     return b
202
203
204 def prochaineMachineCible():
205     """Selectionne une machine aléatoirement parmi la liste des machines cibles
206     randomI = random.randint(0, prochaineMachinesCible().shape[0]-1)
207     return prochaineMachinesCible()[randomI]
208
209
210 def destinationBras(bras, roue):
211     """Renvoie la liste des machines que le bras doit cibler pour poser les roues
212     dentée qu'il possède
213     a=np.zeros(0, int)
214     b=np.zeros(0, int)
215     for i in bras:
216         if i[0]!=0:
217             a=np.append(a,roue[i[0]])
218         else:
219             a=np.append(a,0)
220     print(a)
221
222     for i in a:
223         if i==0:
224             b=np.append(b,0)
225         else:
226             b=np.append(b, brasC[np.where(i==brasC)[0][0]+1])
227     return b
228
229 def prochaineMachinesVides(bras, roue, machines):
230     """Renvoie la liste des machines que le robot peut cibler par rapport aux roues
231     dentées qu'elle possède dans ses préhenseurs
232     c=destinationBras(bras, roue)
233     a= machines.take(c)
234     b = np.array(a== np.zeros(len(a)))
235
236     return c.take(np.nonzero(b)), np.nonzero(b)
237
238 def prochaineMachinesVidesN(bras, roue, machines):

```

```

237     ###Renvoie le nombre de machines Ciblenible par rapport a ce que le robot a dans
    ses préhenseurs
238     return prochaineMachinesVides(bras, roue, machines)[0].shape[1]
239
240 def prochaineMachinesVide(bras, roue, machines):
241     ###Choisi la prochaine machine vide parmi la liste générée des machines
    accessibles
242     a=prochaineMachinesVides(bras, roue, machines)
243     randomI = random.randint(0, a[0].shape[0]-1)
244     return (a[0][randomI,0], a[1][randomI][0])
245
246 def createExcel():
247     ###Crée une feuille excel
248     global wb, sheet, sheet2, sheet3
249     wb = openpyxl.Workbook()
250     sheet = wb.active
251     sheet.title='Les actions du robots'
252
253     sheet2 = wb.create_sheet()
254     sheet2.title='Historique des machines'
255
256     sheet3 = wb.create_sheet()
257     sheet3.title='Historique du bras'
258
259 def addExcel(tableau, temps):
260     ###ajoute le tableau des résultats et le temps trouvé dans excel
261     global wb, sheet, simulationN
262     columnN = (simulationN-1)*4
263     sheet.cell(row=1, column=columnN+1).value = "Test N° %s" % (simulationN)
264
265     sheet.cell(row=1, column=columnN+2).value='Temps Total'
266     sheet.cell(row=1, column=columnN+3).value=temps
267     sheet.cell(row=2, column=(columnN+1)).value='Action'
268     sheet.cell(row=2, column=(columnN+2)).value='Position'
269     sheet.cell(row=2, column=(columnN+3)).value='Prehenseur'
270
271     for j in range(0, tableau.shape[0]):
272         for i in range(0, tableau.shape[1]):
273             sheet.cell(row=j+3, column=j+1+columnN).value=tableau[j,i]
274
275
276 def addExcel2(tableau):
277     global wb, sheet2, simulationN
278
279     lineN = (simulationN-1)*8
280     sheet2.cell(row=1+lineN, column=1).value = "Test N° %s" % (simulationN)
281
282     for j in range(0, tableau.shape[0]):
283         for i in range(0, tableau.shape[1]):
284             sheet2.cell(row=j+2+lineN, column=i+1).value=tableau[j,i]
285
286 def addExcel3(tableau):
287     global wb, sheet3, simulationN
288
289     lineN = (simulationN-1)*8
290     sheet3.cell(row=1, column=lineN+1).value = "Test N° %s" % (simulationN)
291
292     for j in range(0, tableau.shape[0]):
293         for i in range(0, tableau.shape[1]):
294             sheet3.cell(row=j+2+lineN, column=i+1).value=tableau[j,i]
295
296 def saveExcel():
297     ###Sauvegarde l'excel généré
298     global wb
299     wb.save('write_example.xlsx')

```

```

300
301 def afficher():
302     ###permet d'afficher les historiques machines, bras et le temps écoulé depuis le
    début entre chaque nouveaux instants
303     global machinesH, machines, brasH, bras, tH, t
304
305     machinesH = np.append(machinesH, machines, 1)
306     if machinesH.shape[1] >= 25:
307         machinesH = np.delete(machinesH,0,1)
308     print(machinesH)
309
310     brasH = np.append(brasH, bras, 1)
311     if brasH.shape[1] >= 25:
312         brasH = np.delete(brasH,0,1)
313     print(brasH)
314
315     tH = np.append(tH,t)
316     print(t)
317
318     print(roue)
319
320 def commande():
321     ###Les réflexions principales qui permettent de dicter les commandes du robot
322     global t
323     global machines
324     global lastRoueDone
325     global machinesH
326     global brasH
327     global tH
328     global positions
329     global bras, roue
330
331     t=0
332     while machines[0] <= 16:
333         if prehLibreN() == 2:
334             t+=prendre(0)
335         elif prochaineMachinesVidesN(bras, roue, machines) > 0:
336             a=prochaineMachinesVide(bras, roue, machines)
337             t+=poser(a[0],a[1])
338         else:
339             t+=prendre(prochaineMachineCible())
340             afficher()
341
342     sortieRoues()
343
344
345
346     return positions
347
348 def sortieRoues():
349     ###gère la sortie des rouees dentée manuellement, car cela est plus difficile à
    implémenter que le raisonnement actuel
350     global t
351
352
353
354     prendre(5)
355     afficher()
356     poser(7,1)
357     afficher()
358     prendre(1)
359     afficher()
360     poser(1,0)
361     afficher()
362     prendre(3)

```



```
363     afficher()
364     poser(5,0)
365     afficher()
366     prendre(2)
367     afficher()
368     poser(3,0)
369     afficher()
370     poser(2,1)
371     afficher()
372     prendre(1)
373     afficher()
374     prendre(5)
375     afficher()
376     poser(7,1)
377     afficher()
378     prendre(3)
379     afficher()
380     poser(5,1)
381     afficher()
382     prendre(2)
383     afficher()
384     poser(3,1)
385     afficher()
386     poser(2,0)
387     afficher()
388     prendre(5)
389     afficher()
390     poser(7,0)
391     afficher()
392     prendre(3)
393     afficher()
394     poser(5,0)
395     afficher()
396     prendre(2)
397     afficher()
398     poser(3,0)
399     afficher()
400     prendre(5)
401     afficher()
402     poser(7,0)
403     afficher()
404     prendre(3)
405     afficher()
406     poser(5,0)
407     afficher()
408     prendre(5)
409     afficher()
410     poser(7,0)
411     afficher()
412
413
414 def main():
415     """Gère le nombre de simulation et coordonne leurs affichage dans une feuille
    excel
416     global simulationN, t, machinesH, brasH
417     createExcel()
418
419     initializeVariables()
420     addExcel(commande(), t[0])
421     addExcel2(machinesH)
422     addExcel3(brasH)
423     simulationN += 1
424
425     initializeVariables()
426     addExcel(commande(), t[0])
```

```
427     addExcel2(machinesH)
428     addExcel3(brasH)
429     simulationN += 1
430
431     saveExcel()
432
433 #On execute la fonction main pour lancer le programme à l'exécution
434 main()
```

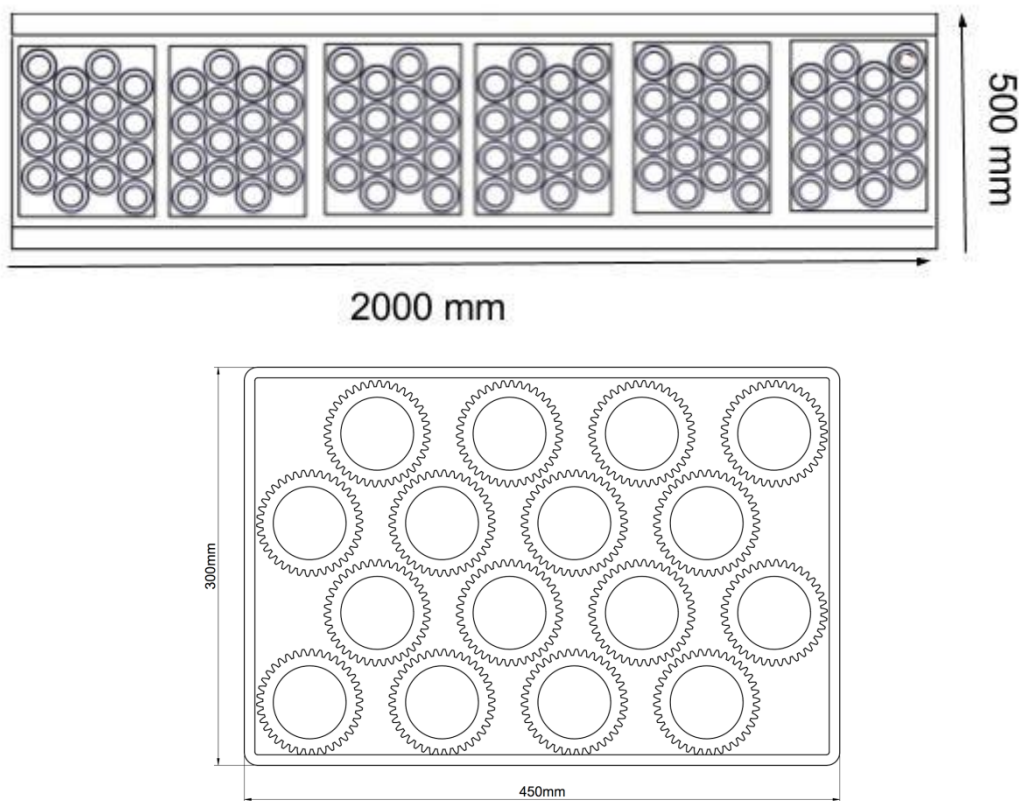
### Prix du préhenseur

On réalise notre préhenseur par impression 3D Titane, le prix de la poudre de titane étant d'environ 400 € le kilogramme.

Prix au kilo	Kilo de métal utilisé	Prix total
400 €	1,36 Kg	544 €

### Choix et prix des convoyeurs













#### Convoyeur d'entrée:



Longueur : 2 m (pour 6 bacs)













Largeur : 500 mm (> largeur d'un bac)

Longueur (mm)	Nombre de bacs (u)
2000	6.67

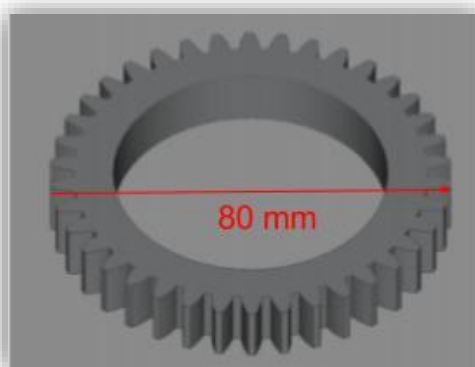
 <a href="#">A011795</a>	Modèle	<b>LONGUEUR 2 M</b>	Longueur (m)	<b>2 m</b>	<b>2 499,00 €</b>	 <b>Livré par notre fournisseur</b> Livré le 7 févr.	<b>1</b>	
	 Voir les documents techniques				Vendu par 1 Pièce			
 <a href="#">A011796</a>	Modèle	<b>LONGUEUR 3 M</b>	Longueur (m)	<b>3 m</b>	<b>2 665,00 €</b>	 <b>Livré par notre fournisseur</b> Livré le 7 févr.	<b>1</b>	
	 Voir les documents techniques				Vendu par 1 Pièce			
 <a href="#">A011797</a>	Modèle	<b>LONGUEUR 4 M</b>	Longueur (m)	<b>4 m</b>	<b>2 955,00 €</b>	 <b>Livré par notre fournisseur</b> Livré le 7 févr.	<b>1</b>	
	 Voir les documents techniques				Vendu par 1 Pièce			

<b>Prix (€)</b>
<b>2499</b>

## Convoyeur de rebuts et de sortie:

 <a href="#">A011886</a>	Modèle	<b>LONGUEUR 1 M</b>	Longueur (m)	<b>1 m</b>	<b>1 539,00 €</b>	 <b>Livré par notre fournisseur</b> Livré le 2 févr.	<b>1</b>	
	 Voir les documents techniques				Vendu par 1 Pièce			
 <a href="#">A011887</a>	Modèle	<b>LONGUEUR 2 M</b>	Longueur (m)	<b>2 m</b>	<b>1 635,00 €</b>	 <b>Livré par notre fournisseur</b> Livré le 2 févr.	<b>1</b>	
	 Voir les documents techniques				Vendu par 1 Pièce			
 <a href="#">A011888</a>	Modèle	<b>LONGUEUR 1.5 M</b>	Longueur (m)	<b>1.5 m</b>	<b>1 589,00 €</b>	 <b>Livré par notre fournisseur</b> Livré le 2 févr.	<b>1</b>	
	 Voir les documents techniques				Vendu par 1 Pièce			

Prix unitaire (€)	Nombre	Coût total (€)
1635	3	4905



Longueur : 1,50 m

Largeur : 100 mm (> largeur pignon)

**Prix des baies + Portes**

### Axelent – Cloison grillagée pour la protection des machines X-GUARD CLASSIC, ...

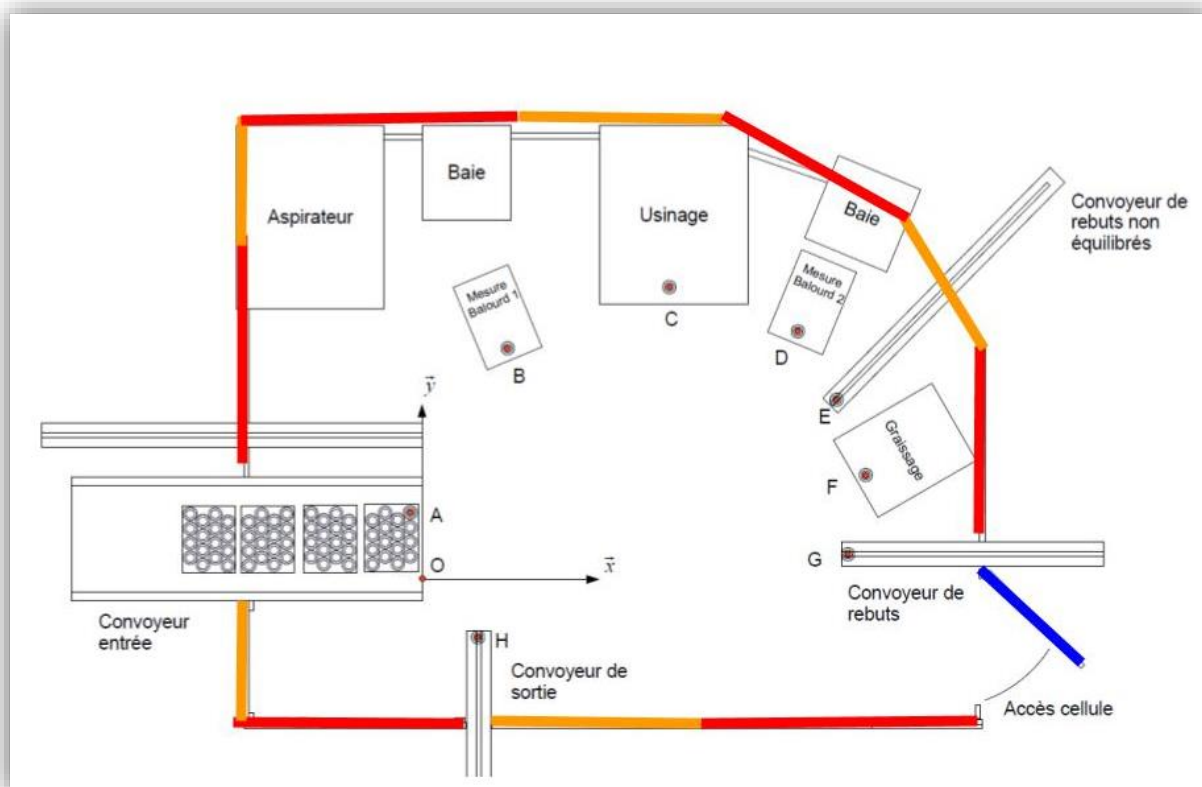
hauteur 1900 mm

★★★★★



- Système de cloisons de protection des machines et installations électroniques – distance de 200 mm de la source présentant des risques
- Modèles conformes à la norme 2006/42/CE concernant les machines
- Dimensions des mailles répondant aux prescriptions: 30 x 50 mm – une main ne peut pas passer
- Peinture époxy
- Kit de porte transformant les cloisons en portes simples, doubles ou coulissantes
- Montage rapide – système extensible à volonté

Plus de Axelent | Autres Axelent Cloisons grillagées



Prix moyen d'une section grillagée (€)	Nombre	Coût total (€)
170 €	11	1870 €

Prix Porte
110 €