



Rapport Stäubli

Réalisé par :

Aubin TOURAIS
Matthieu MOCHET

Tom AUGER
Nicolas FELLAH

Professeur :

Rémi CHALARD

SOMMAIRE

Introduction.....	4
Présentation Sujet.....	5
1. Cahier des charges.....	5
2. Matériel.....	5
3. Planning.....	5
Distribution des rôles.....	6
Logiciel SRS Staubli.....	7
I. Installation des licences.....	7
II. Prise en main de SRS.....	7
III. Communication réseau.....	10
Installation physique.....	14
I. Installation du robot.....	14
II. Modélisation de la table.....	14
III. Organisation (placement).....	14
Ajout des capteurs.....	15
I. Choix des capteurs.....	15
• Scrutateur Laser S300 Mini:.....	15
• Capteur de porte:.....	15
II. Réalisation des schémas électriques.....	15
III. Paramétrage.....	18
IV. Programmation.....	24
Autres capteurs.....	26
Armoire Électrique.....	30
I. Schématisation.....	30
II. Montage réel.....	31
III. Implémentation/Connexion.....	32
Essais de programmation sous Raspberry.....	34
Modélisation 3D.....	36
I. Schématisation.....	36
II. Fusion 360 (explication).....	38
III. Réalisation sur logiciel.....	41
IV. Transfert sur SRS.....	42

Exercice/Application.....	43
I. Déplacement simple.....	43
II. Variables / Données.....	43
III. Fonctions.....	44
IV. Pick And Place.....	45
Programme Base / Pick and Place :	45
Programme Tower/ Pick and Place :	46
Programme tourni / Pick and Place :	47
V. Lettre / Chiffre.....	48
Programme Test Lettre :	50
Programme Test Chiffre :	52
Programme Clavier :	53
Livrable.....	55
TP Exercices et Correction pour les prochaines années.....	56
Diaporama Soutenance.....	56
Appels Stäubli.....	57
Conclusion.....	57
Améliorations envisagées.....	58

Introduction

Dans le contexte de ce projet, nous avons effectué l'assemblage intégral d'un bras robotique Stäubli TX2-40 sur une table mobile, incorporant différents capteurs de sécurité et une interface logicielle via SRS.

L'objectif était de deux volets : construire un milieu professionnel sécurisé et efficace, tout en concevant un outil pédagogique qui permettrait d'utiliser pleinement les potentialités du robot. Nous avons donc géré tout le processus d'implémentation, de l'installation physique du robot et de son contrôleur CS9, jusqu'à l'intégration d'un scanner laser et d'un détecteur de porte qui garantissent la décélération ou l'arrêt du robot en cas d'obstacle détecté.

Grâce à la modélisation 3D via Fusion 360, nous avons pu concevoir et imprimer un support de stylo, un élément crucial pour les exercices d'écriture automatisée.

En parallèle, nous avons élaboré un programme sophistiqué pour le robot, comprenant des mouvements précis, des opérations de prise et de dépose, ainsi que la faculté d'écrire des lettres et des chiffres, ce qui renforce l'aspect éducatif du projet.

Nous avons aussi élaboré un document détaillé, accompagné d'une série de travaux dirigés corrigés, conçus pour aider les futurs étudiants à se familiariser avec le robot.

Ce projet nous a donné l'opportunité de perfectionner nos aptitudes en automatisation, en programmation de robots, en modélisation tridimensionnelle et en intégration de capteurs, tout en satisfaisant aux normes industrielles de sécurité et d'efficience.

Présentation Sujet

1. Cahier des charges

L'objectif de ce projet est de réaliser l'installation complète d'un bras robot Stäubli sur table roulante ainsi que de ces licences, l'ajout d'un scrutateur laser permettant l'arrêt d'urgence du robot ou le ralentissement du robot et l'ajout d'un capteur de porte permettant le ralentissement du robot.

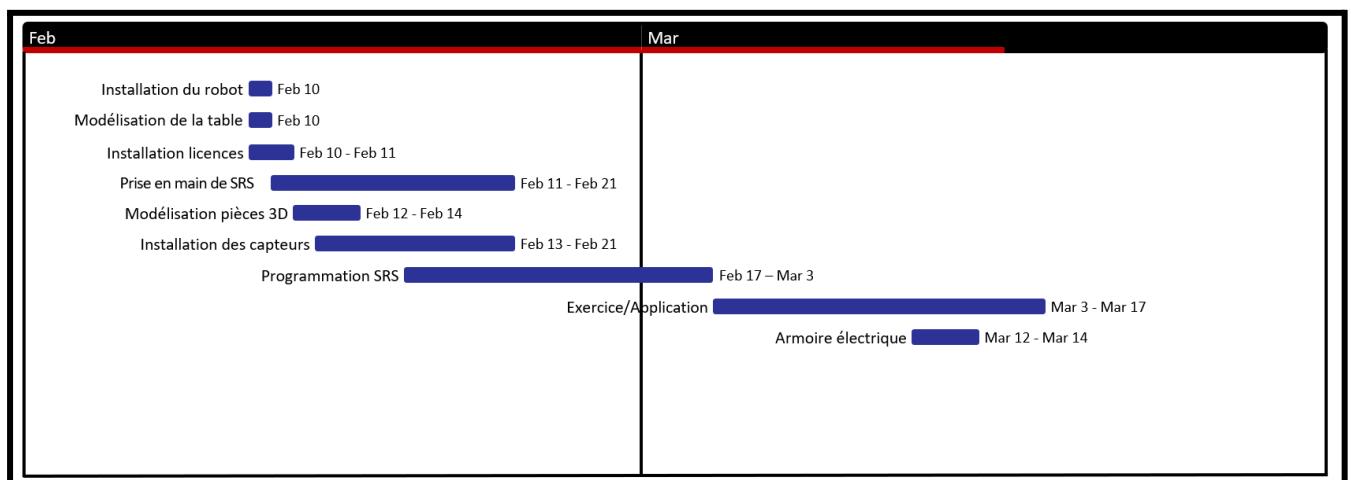
De plus, il faut modéliser et imprimer un support pour outil (dans notre cas un stylo) permettant la visualisation de nos exercices.

Enfin, il faut réaliser la programmation du robot permettant de réaliser différents exercices choisis par nous même (réaliser les lettres de l'alphabet, écrire des mots/phrases, pick and place...)

2. Matériel

- Bras Robot TX2-40
- Contrôleur CS9
- Scrutateur Laser S300 Mini Sick
- Capteur de porte Sick
- 3 Contacteurs.
- 1 boîtier armoire électrique
- 1 alimentation 24V

3. Planning



Distribution des rôles

- **Aubin :**

Chef de projet, Logiciel SRS, programmation, installation des licences, armoire électrique, exercices pour TP, correction pour TP, appels stäubli partie programmation logiciel

- **Nicolas :**

Installation des capteurs, réalisation des schémas électriques

- **Tom :**

Modélisation 3D, installation des licences

- **Matthieu :**

Paramétrage du laser, installation des capteurs, armoire électrique, ponçage, appels stäubli partie électronique

- **Melchior :**

installation des capteurs, appels stäubli partie électronique

Logiciel SRS Staubli

I. Installation des licences

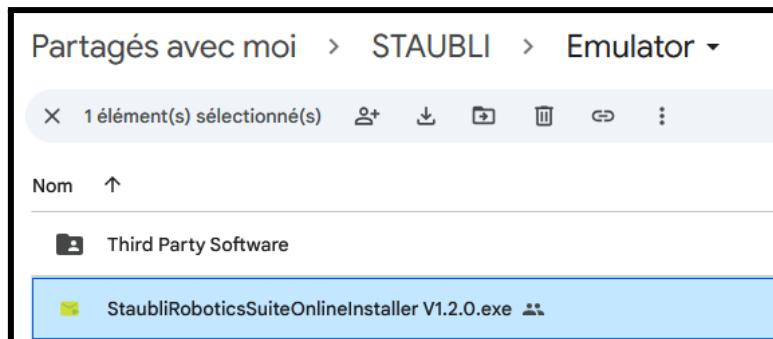
Avant de commencer, nous avons fait l'installation des licences sur les PC, pour pouvoir lancer le logiciel Staubli et utiliser les différents types de robots, il faut bien avoir le dongle, sinon les licences ne seront pas à disposition.



Pour faire l'installation sur les PC, vous pouvez y accéder de la manière suivante :

(dossier **Emulator**)

https://drive.google.com/drive/folders/1G_Ir1GZ5F7mPx9zc1ao7RvyBrbEG5pN2?usp=sharing



The screenshot shows a Google Drive interface. At the top, it says "Partagés avec moi > STAUBLI > Emulator". Below that is a toolbar with icons for search, add, download, delete, and more. The main list shows two items: "Third Party Software" and "StaubliRoboticsSuiteOnlineInstaller V1.2.0.exe". The second item is highlighted with a blue background.

Par la suite, il faut avoir accès aux droits admin pour faire les différentes installations. Ensuite, ayez le dongle lorsque vous souhaitez utiliser le logiciel, sinon cela vous posera des interdictions.

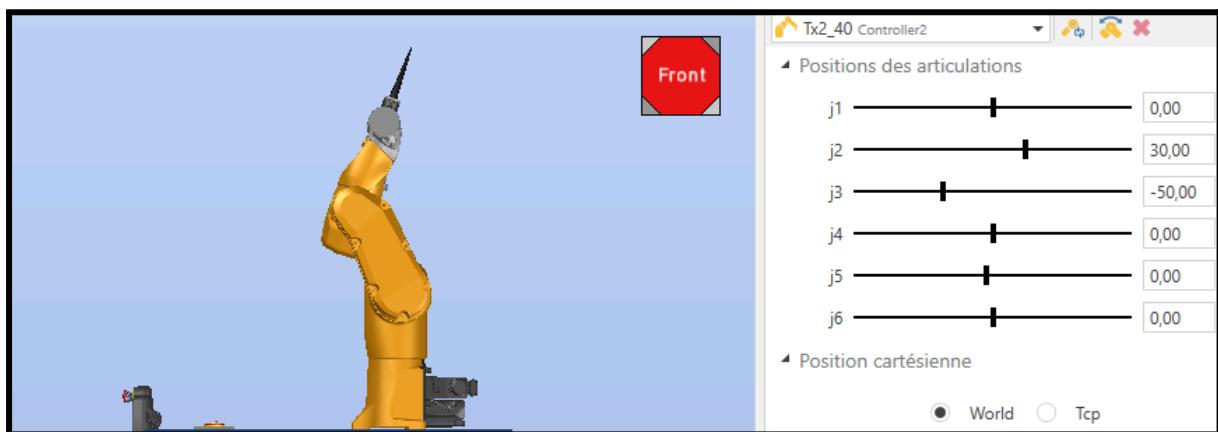
II. Prise en main de SRS

Nous avons avant tout fait la prise en main du logiciel SRS, une fois l'installation de nos licences, des bonnes options de safety, afin de pouvoir changer les ranges de nos axes etc.

Nous avons pu par la suite, faire la création 3D de notre bras TX2-40, le jumeau numérique, afin de faire les tests au préalable sur la simulation avant de l'envoyer sur le robot, ce qui permet d'éviter tout incident ou problème technique.

Nous avons par ailleurs, ajouté le monde qui entoure notre robot, comme la table sur laquelle est positionné notre robot, afin de pouvoir détecter les collisions sur notre jumeau numérique et ensuite corriger cela avant d'envoyer notre programme dans le robot. Cela nous permet de garantir plus de sécurité, et tester le tout en simulation et être entièrement tranquille lors de la compilation en réel.

Ensuite, nous avons fait de léger test, comme le fait de voir que nous pouvions bouger les axes manuellement à notre guise, et ainsi voir les positions de nos axes.



Nous pouvons ainsi, si l'on souhaite, créer un tableau pour les positions des axes, ou garder leurs positions pour les réutiliser dans un programme par exemple.

Puis, nous avons pu ajouter de léger programme justement pour tester la simulation, ajout de variable etc. Comme le programme suivant, qui fait bouger deux axes, et lors de la compilation, où nous pouvons voir le robot faire les déplacements sous la simulation.

```

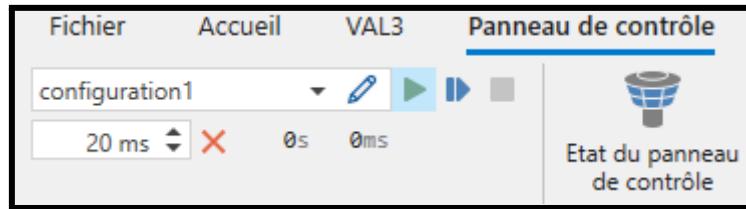
begin
  jDest.j2 = 88
  jDest.j5 = 56
  movej(jDest, flange, mNomSpeed)
  waitEndMove()
end
  
```

ATTENTION !

Nous avons pu voir justement que pour compiler tous nos programmes, il faut bien être dans l'onglet d'un programme. Sinon, cela reste grisé et nous ne pouvons pas le compiler.



Puis pour lancer un programme, si la compilation ne pose pas de problème, il faut lancer l'exécution de celui-ci.

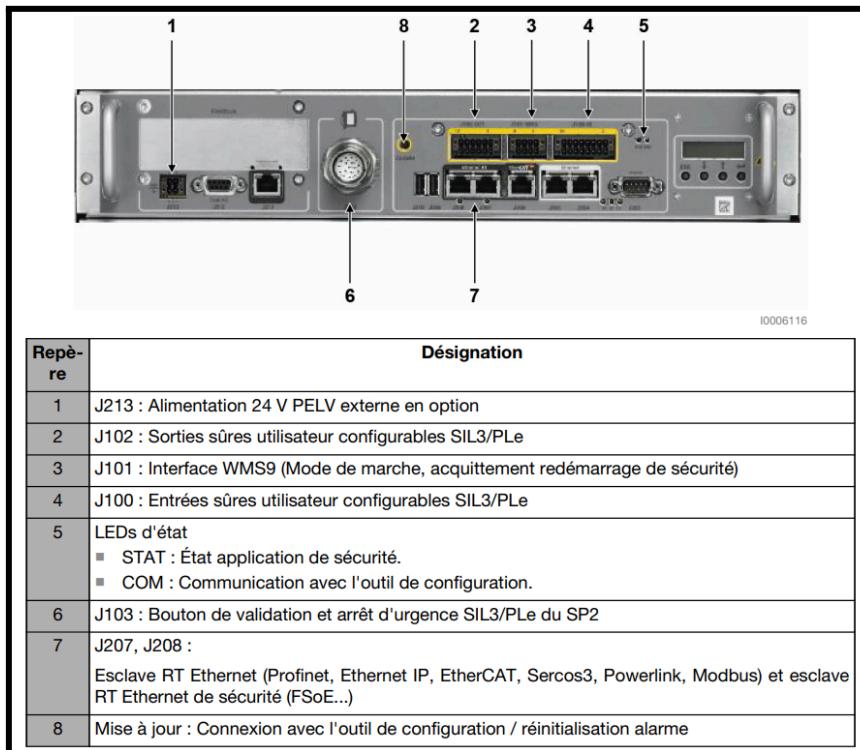


A noter, que si vous souhaitez compiler un autre programme, ou relancer une exécution, il faut penser à arrêter le programme en cours, et recompiler puis relancer l'exécution.

Si vous avez changé des configurations dans la safety, et que vous souhaitez l'envoyer dans le contrôleur. Faites les étapes suivantes :

Fonctions de sécurité > Exporter la configuration > suivant > Terminer

Puis suivez les étapes indiquées, il faut bien penser à appuyer sur le boutons Update qui se trouve sur le contrôleur. (Ici nous le voyons sur le repère 8)



On vous demandera de faire relancer le contrôleur, mais penser à attendre 2-3 minutes après le chargement pour laisser le tout bien se charger, ET pareil lors du redémarrage, 2-3 minutes, sinon vous empêcherait la carte de bien se reboot et vous serez obligé d'appeler un technicien de chez Staubli pour vous corriger cela à distance et tout vous retélécharger.

III. Communication réseau

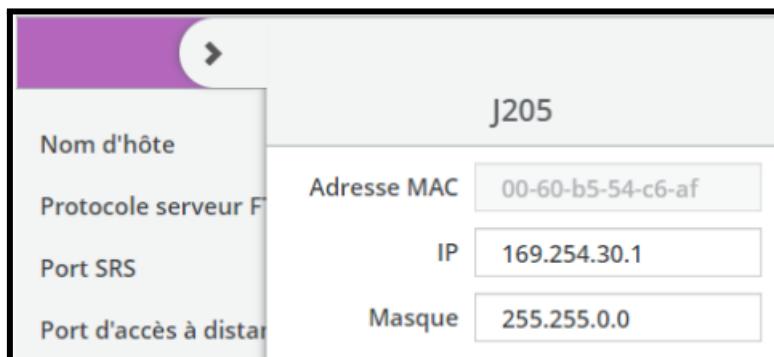
Pour faire la communication en réseau du logiciel SRS avec le contrôleur CS9 de notre TX2-40, il faut tout d'abord placer la communication sous le même masque de réseau.

Pour cela, nous faisons un ipconfig dans notre CMD afin de connaître notre adresse IP, ainsi que le masque de sous réseau.

```
Carte Ethernet Ethernet 3 :  
  Suffrage DNS propre à la connexion. . . . .  
  Adresse IPv6 de liaison locale. . . . . : fe80::c67e:ee7a:89d1:98ed%9  
  Adresse d'autoconfiguration IPv4 . . . . . : 169.254.30.68  
  Masque de sous-réseau. . . . . . . . . . . : 255.255.0.0  
  Passerelle par défaut. . . . . . . . . . . : 
```

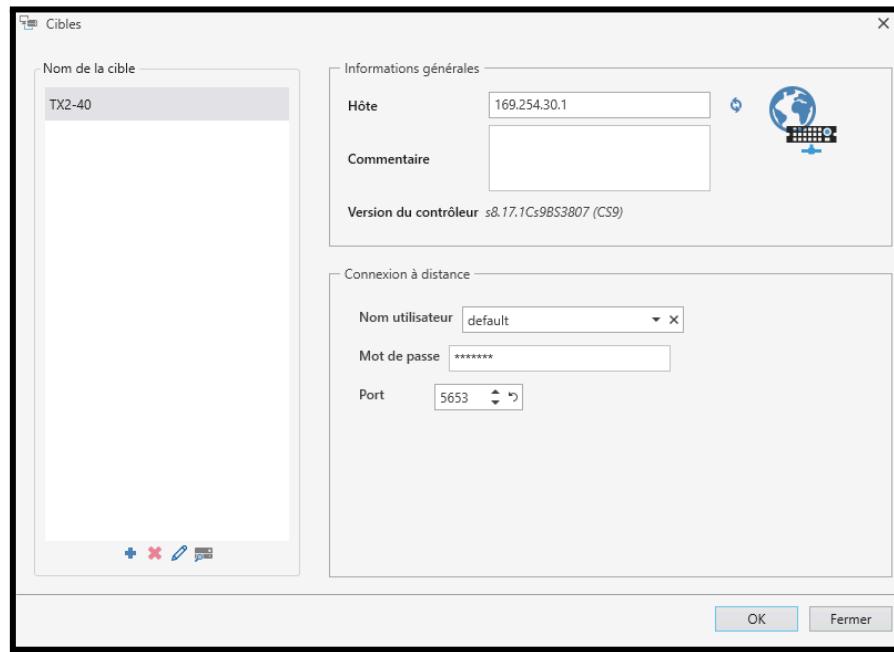
Il faudra donc que dans notre SP2, nous mettions le même masque de sous réseau, et que nous changions la fin de notre adresse afin de la différencier de notre PC.

Nous avons mis ainsi l'adresse suivante :



En ayant bien branché avec un RJ45 notre PC au port souhaité sur le contrôleur, ici nous avons le J205.

Par la suite, nous faisons la connexion entre ses deux appareils, dans l'onglet transfert et connexion du logiciel SRS.



Ensuite, une fois cela terminé, nous pouvons faire le transfert de nos données, de nos programmes et configurations dans le contrôleur.

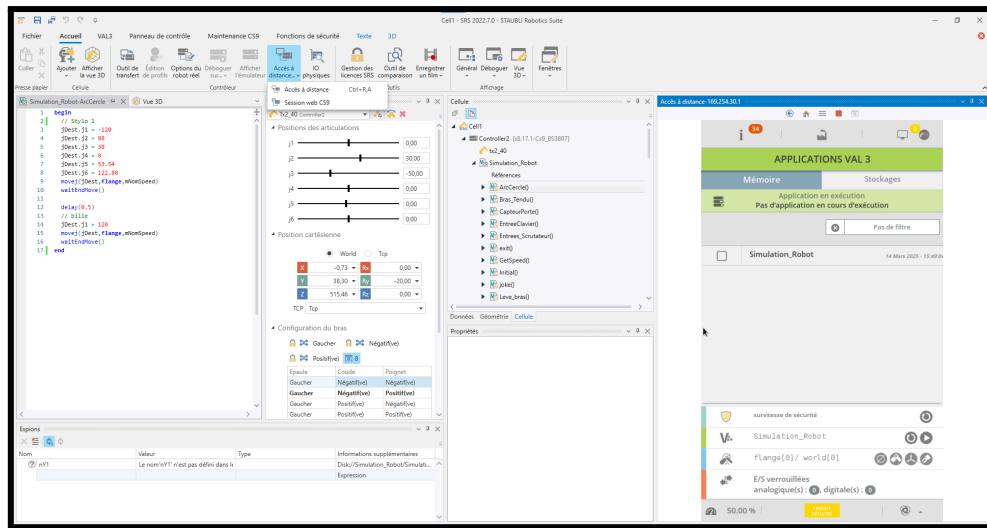
De plus, nous pouvons accéder à distance à notre contrôleur en parallèle de notre SP2, et ainsi en prendre le contrôle. Cela peut être utile, pour faire des démonstration, ou encore pour piloter le robot en mode déporté, étant donné que le SP2 est débranché afin de faire la commande de vitesse via programmation.

La session web CS9 permet avec le shunt de contrôler à distance le robot avec le logiciel SRS.



En remplaçant le handle (SP2) qui est connecté au contrôleur CS9, puis en y mettant le cache de maintenance, vous pourrez à partir du logiciel SRS, piloter le robot tout en ajustant sa vitesse à votre guise.

L'Accès à distance fonctionne en parallèle avec la tablette, quand on utilise la tablette on peut voir en même temps sur le logiciel SRS ce que la personne fait.



Pour exécuter un programme directement sur le robot via le contrôleur SP2, il faut effectuer les étapes suivantes :

VAL3 > Stockages > “Votre projet” >



Puis dans Mémoire, Sélectionner votre projet, l'actualiser si vous avez fait des modifications entre temps, et exécuter le.

A noter, qu'il faut être en mode automatique / déporté pour le lancer.

Si vous êtes en mode automatique :

- Sur le SP2, Gâchette arrière + Power et ensuite lancer le programme via le bouton II

Si vous êtes en mode déporté :

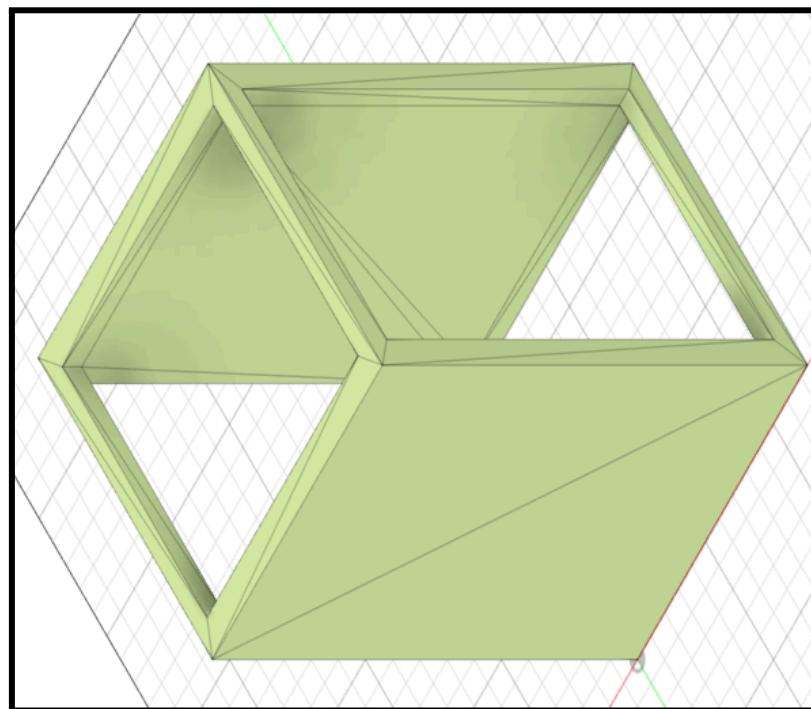
- Utiliser enablePower et disablePower dans le programme
- Et utiliser l'interface session web CS9
- SP2 retiré du contrôleur

Installation physique

I. Installation du robot

Le bras robot est installé sur la partie supérieure d'une table à double niveau sur roulette (dimensions). Le contrôleur CS9 est quant à lui installé sur la partie inférieure. A côté de ce contrôleur, nous avons une alimentation 24V permettant d'alimenter le scrutateur laser S300 Mini ainsi que le capteur de porte.

II. Modélisation de la table



III. Organisation (placement)

La table est séparée en 2 parties, une supérieure et une inférieure.

Le bras robot est placé sur la partie supérieure de la table, au milieu à l'arrière, à la gauche du bras un trou a été creusé permettant des activités avec le bras robot pour le pick and place, à droite du bras robot à l'avant est placé le boitier arrêt d'urgence ainsi que le capteur de porte. A l'avant au milieu est placé le scrutateur laser.

Sur la partie inférieure est placé le contrôleur CS9. A côté de ce contrôleur est placé l'alimentation des capteurs ainsi que l'armoire électrique contenant les contacteurs pour nos capteurs.

Ajout des capteurs

I. Choix des capteurs

- **Scrutateur Laser S300 Mini:**

Ce capteur est un détecteur de présence paramétrable, on peut choisir entre 3 ranges 1, 2 et 3 mètres. A l'intérieur de la zone, il est possible de créer 3 zones de détection, 1 champ de protection, 2 champs d'alarmes (seuls les champs d'alarmes peuvent renvoyer un signal).

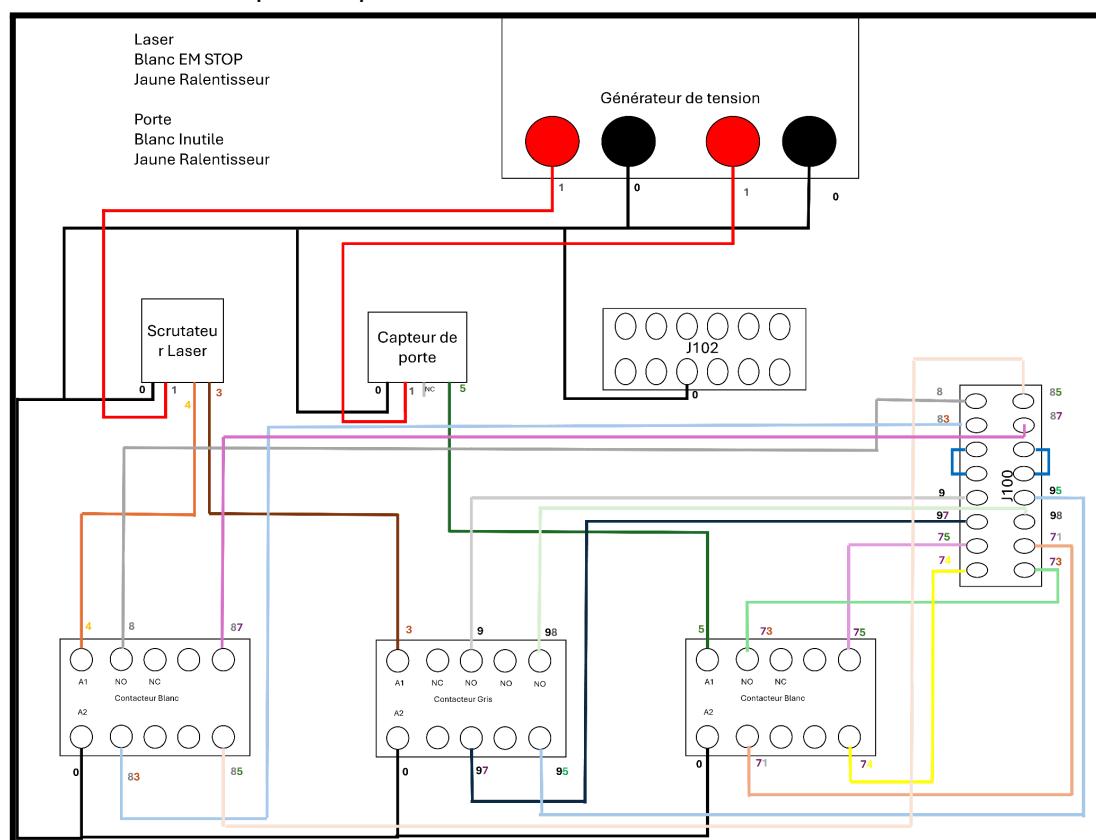
Ce scrutateur possède un afficheur 7 segments permettant d'afficher des symboles/chiffres/lettres qui peuvent être interprétées conformément à la documentation technique. Lorsque personne n'est détecté dans les zones, un signal 24V est envoyé en continu, lorsque quelqu'un est détecté dans l'une des zones, un signal de 0V est envoyé (dans le fil correspondant).

- **Capteur de porte:**

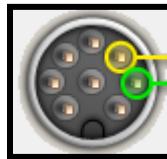
Ce capteur est scindé en 2 parties, 1 aimant simple et 1 aimant avec retour de données et alimentation. Ce capteur est alimenté en 24V et possède 2 états, lorsque l'aimant simple est collé à l'aimant avec retour de données (porte fermée), il envoie 24V continue, lorsque l'aimant simple est décollé (porte ouverte), il envoie 0V.

II. Réalisation des schémas électriques

Schéma électrique complet:



Le scrutateur laser possède un connecteur 8 pins:



Conducteur	Couleur	Fonction
1	Blanc	Sortie champ d'alarme 1
2	Marron	Tension d'alimentation +24 V CC
3	Vert	Raccordement E/S universel 1
4	Jaune	Raccordement E/S universel 2
5	Gris	Sortie de commutation OSSD1
6	Rose	Sortie de commutation OSSD2
7	Bleu	Tension d'alimentation 0 V CC
8	Blindage FE	Terre fonctionnelle/blindage

Parmis les 8 pins, nous en avons utilisé 4 principaux:

- Le Marron pour le 24V
- La Bleu pour la masse
- Le Blanc pour le 1er champ d'alarme
- Le Jaune pour le 2ème champ d'alarme.

Pour le branchement:

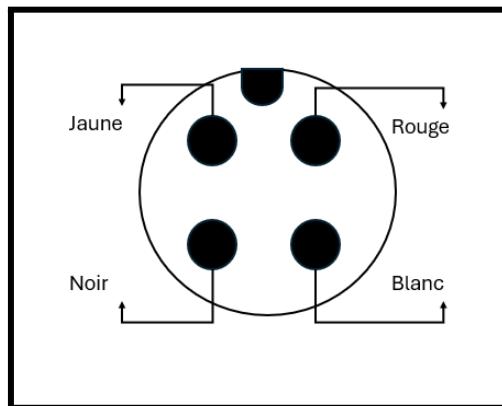
Le signal du fil jaune contrôle l'ouverture et la fermeture d'un contacteur en normalement ouvert. Quand il n'y a personne dans le champ, le scrutateur laser envoie 24V, le circuit du contacteur est fermé, le signal entre le pin 9 et 11 et 10 et 12 du J100 passe, l'arrêt d'urgence n'est pas déclenché. Quand il y a quelqu'un dans le champ, le scrutateur laser envoie 0V, le circuit du contacteur est ouvert, le signal entre le pin 9 et 11 et 10 et 12 du J100 ne passe pas, le robot ralentit .

Le signal du fil blanc contrôle l'ouverture et la fermeture d'un contacteur en normalement ouvert. Quand il n'y a personne dans le champ, le scrutateur laser envoie 24V, le circuit du contacteur est fermé, le signal entre le pin 1 et 3 et 2 et 4 du J100 passe, l'arrêt d'urgence n'est pas déclenché. Quand il y a quelqu'un dans le champ, le scrutateur laser envoie 0V, le circuit du contacteur est ouvert, le signal entre le pin 1 et 3 et 2 et 4 du J100 ne passe pas, l'arrêt d'urgence se déclenche.

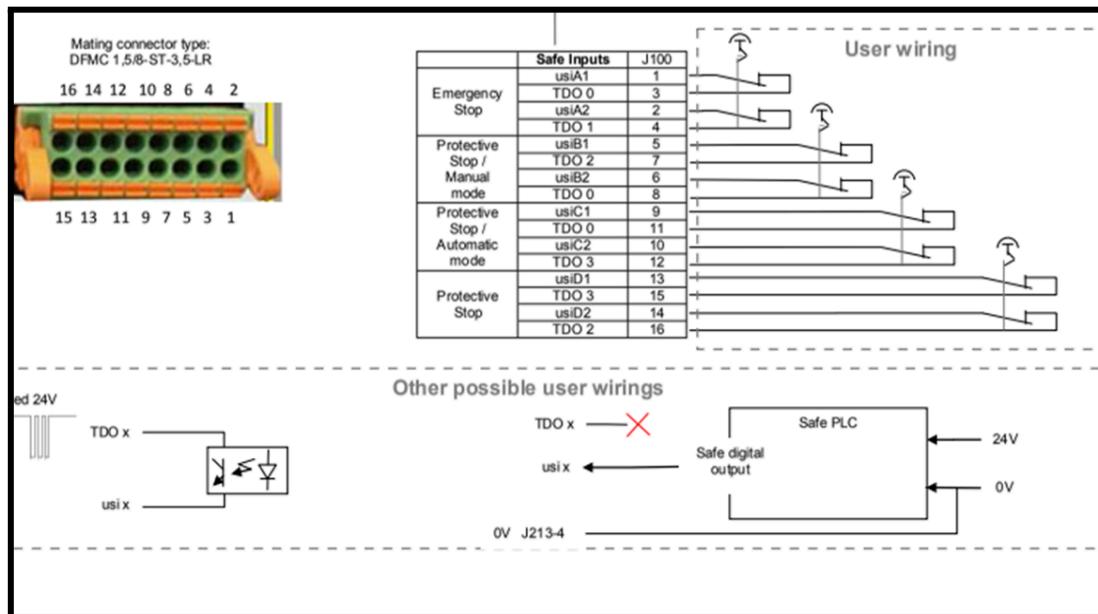
Le capteur de porte Sick est magnétique, est en 2 parties, 1 alimenté en 24V qui peut envoyer une donnée, et un aimant seul. Lorsque la porte est fermée, il envoie du 24V, lorsque la porte est ouverte, il envoie du 0V.

Ce capteur commande un contacteur normalement ouvert sur les ports 13 et 15 et 14 et 16. Ainsi il va changer leur état en fonction de son propre état et déclencher un ralentissement du robot.

Le fil rouge, sert à l'alimenter en 24V, le fil noir, sert pour la masse, le fil jaune sert pour la détection du contact de la porte et le fil blanc est non connecté.



Explication du connecteur J-100:

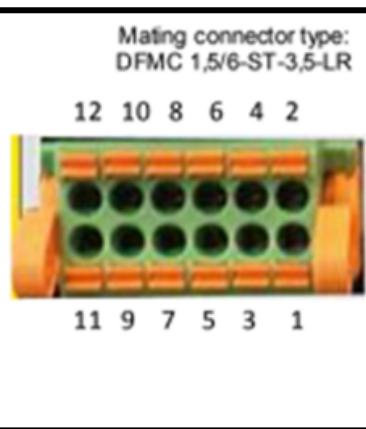


Le connecteur J-100 est divisé en 4 parties, sur chaque partie 2 entrées et 2 sorties. Exemple dans la partie Emergency Stop (TDO 0 et TDO 1 sont des sorties, USIA1 et USIA2 sont des entrées). De base les sorties sont shuntées sur les entrées (le signal de la sortie va sur l'entrée correspondante) et sont activées 2 à 2 et doivent changer d'états en même temps. (Si USIA1 change d'état USIA2 doit changer d'état en même temps).

Ainsi pour câbler un capteur sur le connecteur J-100, il faut que ce capteur contrôle interrupteur permettant de couper la connexion entre les 2 sorties et les 2 entrées, permettant ainsi de déclencher un changement d'état visible sur le logiciel SRS.

Attention: Ne pas shunter TDO 0 sur TDO 1 et USIA1 sur USIA2 cela a été déconseillé par Stäubli. (il faut forcément utiliser un contacteur avec 2 sorties et 2 entrées minimum).

Connecteur J-102:



Mating connector type: DFMC 1,5/6-ST-3,5-LR		
12 10 8 6 4 2	Safe Outputs	J102
E-Stop status	usoA1	1
	usoA2	2
	0V	3
	0V	4
Auto/Manu status	usoB1	5
	usoB2	6
	0V	7
	0V	8
Enabling status	usoC1	9
	usoC2	10
	0V	11
	0V	12

Il faut relier une masse commune au contrôleur CS9, pour cela nous avons relié une masse au Pin 7 du connecteur J-102.

III. Paramétrage

Scrutateur Laser S300 Mini:

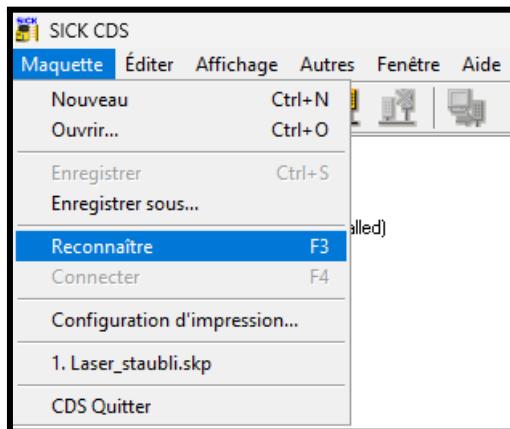
Ce scrutateur laser est configurable via un logiciel (CDS 3.7.2) téléchargeable via leur site. (<https://www.sick.com/fr/fr/catalog/produits/safety/scrutateurs-laser-de-securite/s300-mini/c/g187243?category=g569793&tab=downloads>)

(Si possible avoir tous les droits d'administrateur et désactiver la vérification de l'antivirus sur l'installateur car il est détecté comme un virus, sur la même page, installer les pilotes USB)

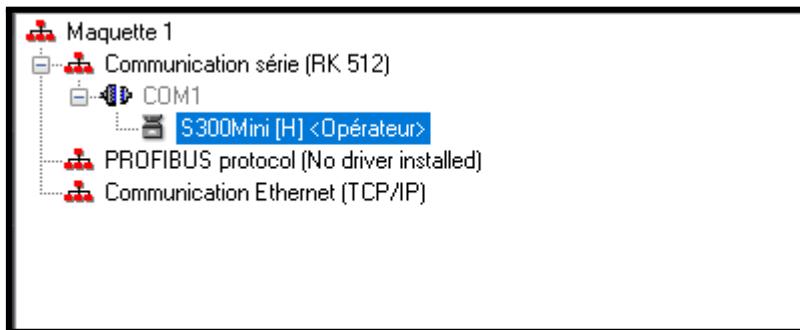
Il faut tout d'abord connecter le scrutateur en USB au pc via le port caché par le caoutchouc dans le cercle rouge avec le câble USB orange :



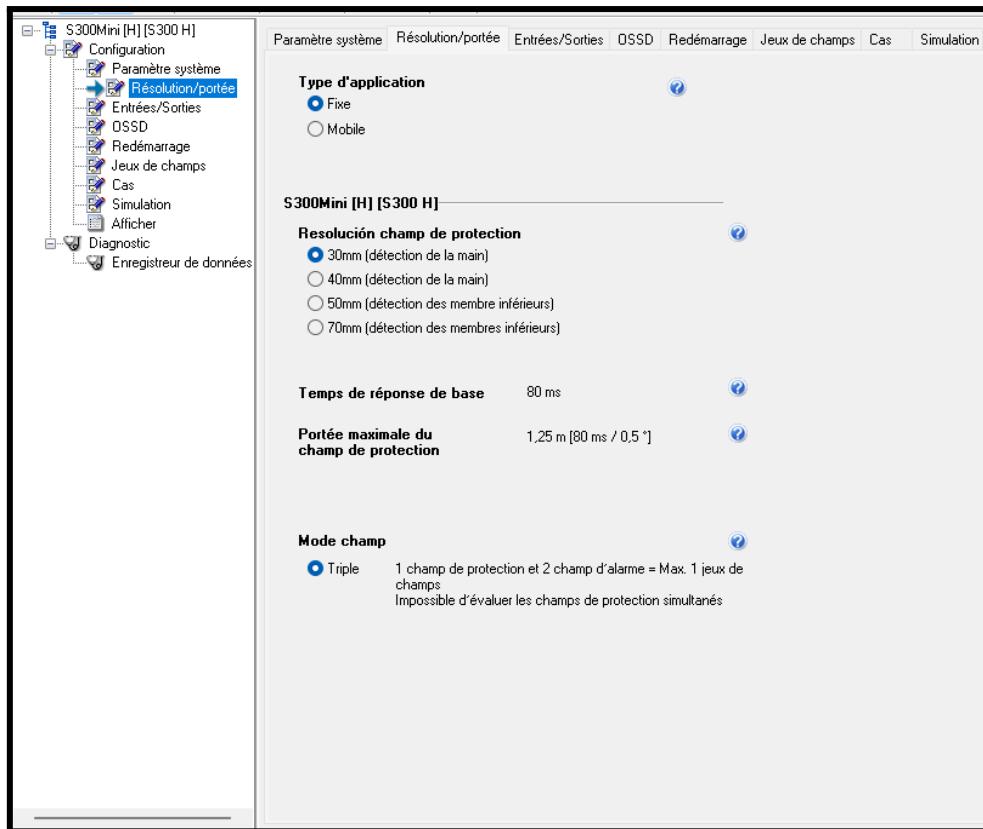
Une fois branché sur le logiciel CDS, il faut cliquer sur Maquette et faire Reconnaître.



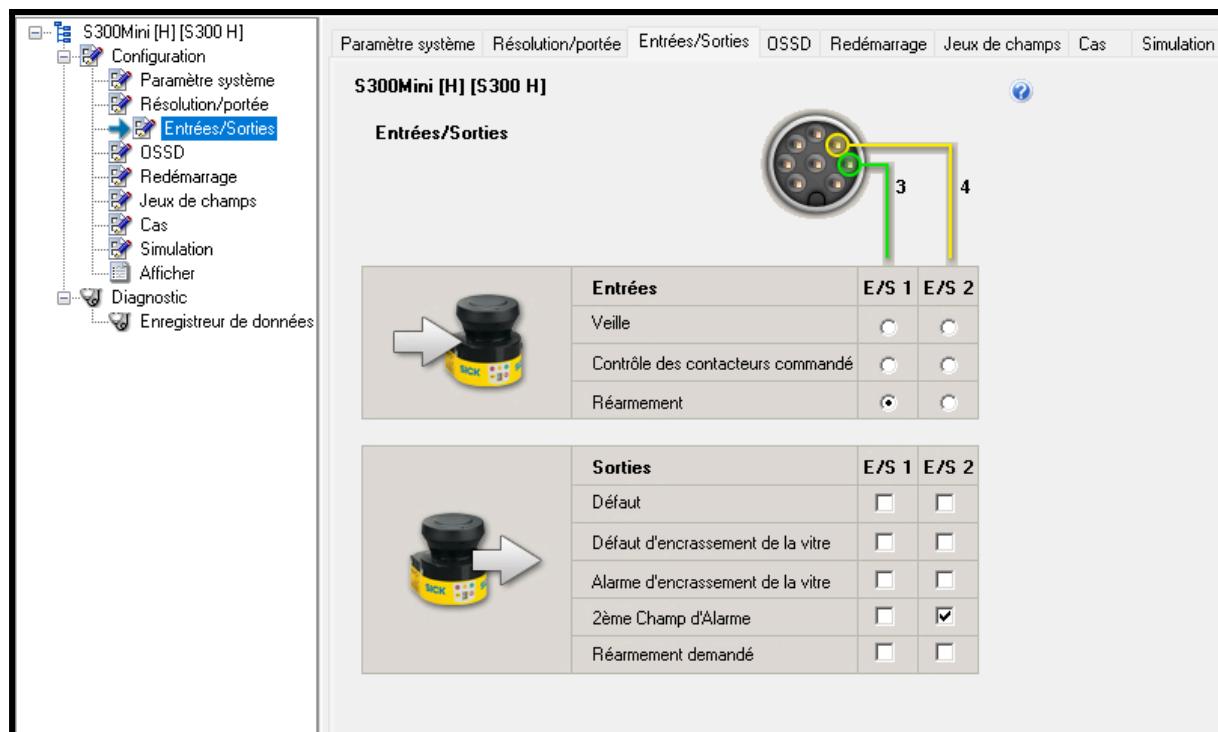
Le logiciel va reconnaître l'appareil et l'ajouter.



Pour le configurer, il faut maintenant double cliquer sur l'appareil qui a été reconnu. Cela ouvre un menu où l'on peut faire la configuration de l'appareil.



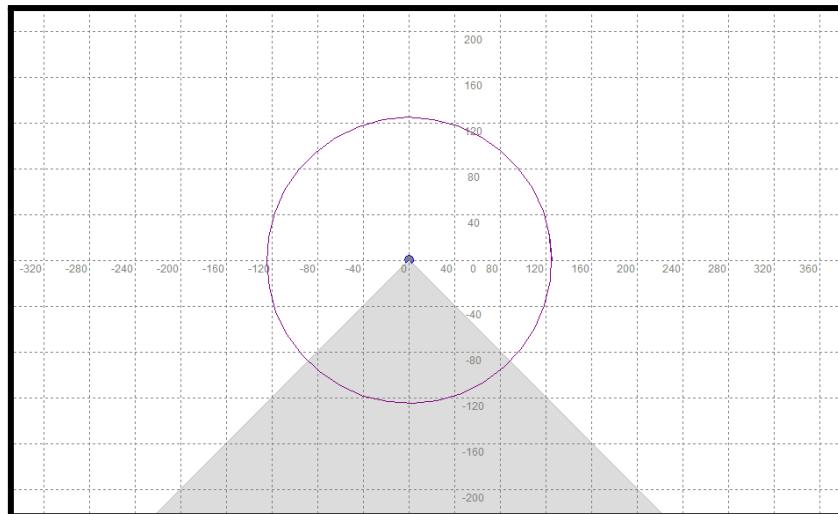
Dans résolution/portée, on peut choisir la taille minimale de l'objet à reconnaître ainsi que le mode de fonctionnement si c'est fixe ou mobile.



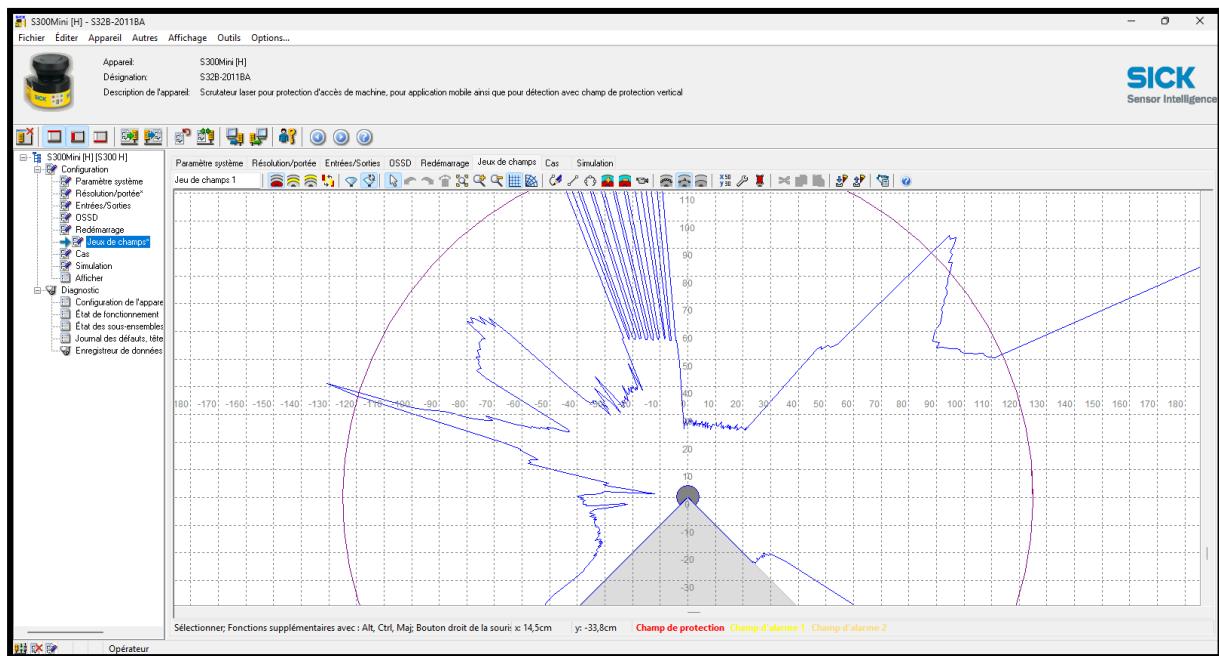
Dans le menu Entrée/Sortie, on peut configurer uniquement la sortie du fil Jaune, (le fil vert est bloqué sur réarmement), dans notre cas on a choisi le 2ème champ d'alarme, on verra son utilité plus tard.

Maintenant le menu Jeu de champs:

Dans ce menu, on voit tout d'abord, le champ de vision maximum du laser avec un cercle tracé autour ainsi que son angle mort.



Le cercle parfait c'est le champ de vision maximum, le triangle grisé, c'est l'angle mort du scrutateur.



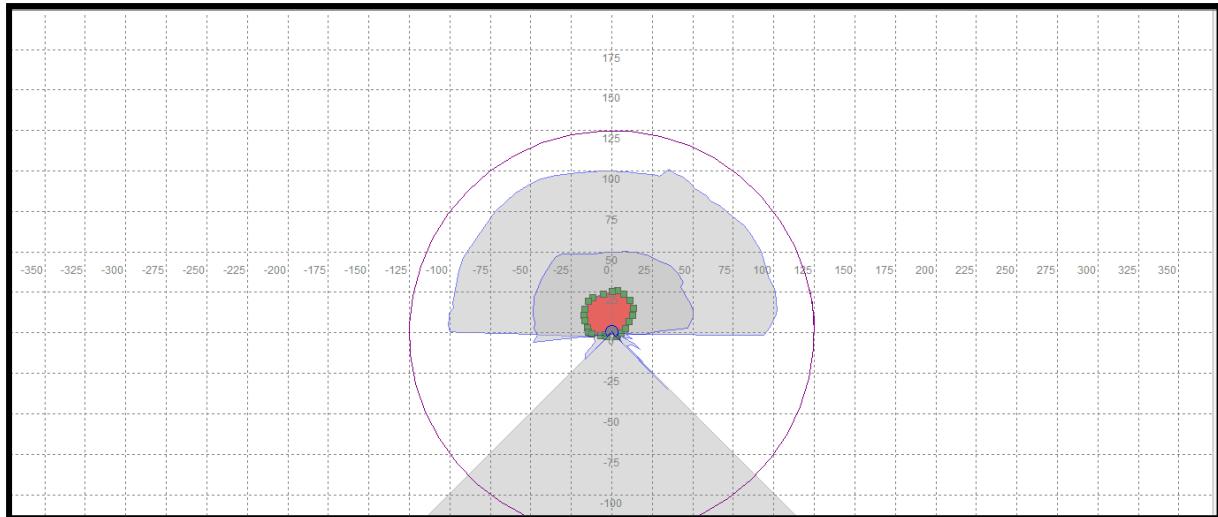
Le logiciel fait une simulation en direct du champ de vision du scrutateur laser, il voit les différents éléments de l'environnement autour de lui.

Ensuite, via les boutons suivants, on peut définir 3 champs:

Le champ de protection, le 1er champ d'alarme, le 2ème champ d'alarme.

Via ce bouton: , on peut dessiner ces champs à la main pour définir quand est-ce que le laser va afficher une alerte et envoyer un signal.

Nous avons décidé de dessiner nos champs de la manière suivante. Le champ de protection (non détectable à l'heure actuelle, très proche de l'appareil, le 1er champ d'alarme, à maximum 50 cm de l'appareil et le 2ème champ d'alarme à 1 mètre de distance.



Le signal du 1er champ d'alarme passe dans le fil blanc du connecteur (n'est pas configurable dans le logiciel, il est donc fixe), un signal 24V continu lorsque personne n'est dans le champ, un 0V si quelqu'un passe dans le champ.

Le signal du 2ème champ d'alarme passe dans le fil jaune du connecteur, un signal 24V continu lorsque personne n'est dans le champ, un 0V si quelqu'un passe dans le champ.

Une fois la configuration faite, pour la transférer dans l'appareil, il faut appuyer sur ce bouton: 

Une fois que la configuration est transférée, le laser va agir de la manière suivante: Via son afficheur 7 segment, il peut afficher les zones dans lesquelles il détecte une personne:



<input type="checkbox"/>	Objet dans le champ de protection	Aucune erreur
<input checked="" type="checkbox"/>	Objet dans le champ d'alarme 1	Aucune erreur
<input type="checkbox"/>	Objet dans le champ d'alarme 2	Aucune erreur

Documentation:

https://cdn.sick.com/media/docs/9/29/529/operating_instructions_s300_mini_safety_laser_scanner_fr_im0040529.pdf

(messages 7 segments: p97 et p105-110)

Le scrutateur laser possède un connecteur 8 pins:



Parmis les 8 pins, nous en avons utilisé 4 principaux:

- Le Marron pour le 24V
- La Bleu pour la masse
- Le Blanc pour le 1er champ d'alarme
- Le Jaune pour le 2ème champ d'alarme.

Capteur de porte:



Le capteur ne nécessite aucun paramétrage, il suffit seulement de le câbler.

J10X		
Rs9IO		
Entrées digitales		
lenabling1	Enabling 1 (J103-7)	Rs9IO\lenabling1
lenabling2	Enabling 2 (J103-12)	Rs9IO\lenabling2
Mcp eStop 1 (J103-5)	Mcp eStop 1 (J103-5)	Rs9IO\mcpEstop1
Mcp eStop 2 (J103-3)	Mcp eStop 2 (J103-3)	Rs9IO\mcpEstop2
IwmsAutoL	WMS local (J101-1)	Rs9IO\iwmsAutoL
IwmsAutoR	WMS remote (J101-2)	Rs9IO\iwmsAutoR
IwmsManuS	WMS manu (J101-3)	Rs9IO\iwmsManuS
IwmsRestart	WMS restart (J101-4)	Rs9IO\iwmsRestart
lusA1	usiA1 (J100-1)	Rs9IO\lusA1
lusA2	usiA2 (J100-2)	Rs9IO\lusA2
lusB1	usiB1 (J100-5)	Rs9IO\lusB1
lusB2	usiB2 (J100-6)	Rs9IO\lusB2
lusC1	usiC1 (J100-9)	Rs9IO\lusC1
lusC2	usiC2 (J100-10)	Rs9IO\lusC2
lusD1	usiD1 (J100-13)	Rs9IO\lusD1
lusD2	usiD2 (J100-14)	Rs9IO\lusD2

Simulation_Robot : dEntree_Scrutateur[0]
 Simulation_Robot : dEntree_Scrutateur[1]

Dans cet onglet SRS, on retrouve les entrées digitales, les entrées "USI" correspondent aux entrées du connecteur J-100 et permettent donc de récupérer leur valeur, ces variables récupèrent en temps réel l'état de l'entrée correspondante du connecteur J-100 et permettent ainsi d'agir sur le programme au travers des entrées.

Les capteurs fonctionnant sur des entrées 2 à 2, "USIA1" et "USIA2" ont la même valeur et changent d'état au même moment.

- USIA1 USIA2 correspondent aux entrées 1 et 2 du J-100
- USIB1 USIB2 correspondent aux entrées 5 et 6 du J-100
- USIC1 USIC2 correspondent aux entrées 9 et 10 du J-100
- USIB1 USIB2 correspondent aux entrées 13 et 14 du J-100

IV. Programmation

Pour effectuer la programmation de nos capteurs, il faut lire les ports où sont branchés nos capteurs, donc lire un état 1 ou 0.

Il faut donc créer des variables booléennes, ici appelé dio.

Index	Io
0	Rsi9IO\lusiC2
1	Rsi9IO\lusiD1

dans le programme suivant, nous testons les différentes entrées en faisant une table de vérité afin de tester quand notre robot doit ralentir et dans quel cas :

```

1 begin
2   while true
3     jDest.j2 = 0
4     movej(jDest, flange, mNomSpeed)
5     waitEndMove()
6     delay(0.5)
7
8     if((dEntree_Scrutateur[0] == true) and (dEntree_Scrutateur[1] == true))
9       setMonitorSpeed(75)
10      jDest.j2 = 40
11      movej(jDest, flange, mNomSpeed)
12      waitEndMove()
13      // Sinon dEntree_Scrutateur[0] == false
14     elseif((dEntree_Scrutateur[0] == false) and (dEntree_Scrutateur[1] == true))
15       setMonitorSpeed(50)
16       jDest.j2 = 40
17       movej(jDest, flange, mNomSpeed)
18       waitEndMove()
19     elseif((dEntree_Scrutateur[0] == true) and (dEntree_Scrutateur[1] == false))
20       setMonitorSpeed(25)
21       jDest.j2 = 40
22       movej(jDest, flange, mNomSpeed)
23       waitEndMove()
24       // ((dEntree_Scrutateur[0] == false) and ((dEntree_Scrutateur[1] == false)))
25     else
26       setMonitorSpeed(10)
27       jDest.j2 = 40
28       movej(jDest, flange, mNomSpeed)
29       waitEndMove()
30     endIf
31     delay(0.5)
32   endwhile
33 end

```

Voici pour résumé et faire simple, nous deux capteurs de base, le capteur de porte, et le scrutateur laser.

- Capteur Porte = 0
- Laser = 0

Vitesse de travail forte 75% (personne dans la zone)

- Capteur Porte = 0
- Laser = 1

Vitesse de travail ralenti 50% (quelqu'un s'approche de la cage, potentiellement envie d'ouvrir)

- Capteur Porte = 1
- Laser = 0

Vitesse de travail faible 25% (quelqu'un a ouvert la cage, mais n'est plus présent dans la zone)

- Capteur Porte = 1
- Laser = 1

Vitesse de travail très lente 10% (une personne dans la zone et la porte est ouverte)

De plus nous pouvons voir en direct l'état de nos entrées :

Nom	Valeur	Verro...	Info	Noms logiques
Controller2 synchronize on	CO Default			
Rsi9IO\lusiD1	<input type="checkbox"/>	<input type="checkbox"/>		Simulation_Robot : dEntree_Scrutateur[1]
Rsi9IO\lusiC2	<input checked="" type="checkbox"/>	<input type="checkbox"/>		Simulation_Robot : dEntree_Scrutateur[0]

Ici par exemple, nous avons ici un des capteurs qui détecte donc on ralentit le robot.

Autres capteurs

Capteurs Supplémentaire (Sécurité Machine)

Capteur photoélectrique:



Ce capteur est en 2 parties: 1 diode laser infrarouge avec récepteur et 1 miroir. La diode envoie un faisceau infrarouge jusqu'au miroir qui le reflète sur le récepteur. Le capteur fonctionne de la manière suivante:

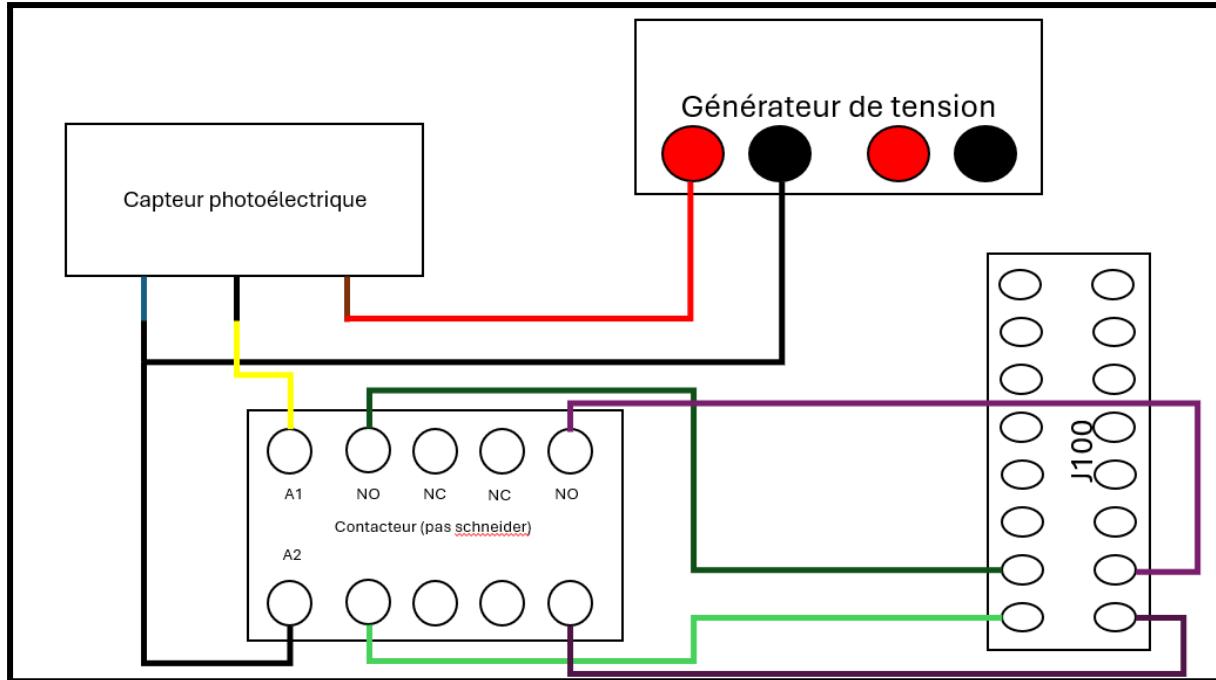
- Si le faisceau est détecté (il n'y a personne dans le champ) il envoie 0V, (il fonctionne donc en inversé par rapport au scrutateur laser).
- Si le faisceau n'est pas détecté (il est coupé, il y a une personne dans le champ), il envoie 24V.

Ainsi, si l'on veut couper le courant pour le faire fonctionner en arrêt d'urgence ou l'utiliser pour faire ralentir le robot, il faut utiliser un contacteur normalement fermé.

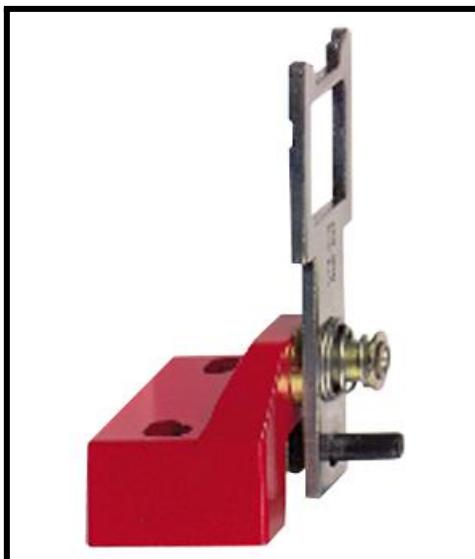
Ce capteur ne nécessite aucune configuration, il suffit simplement de le brancher.

- Masse: Bleu
- Signal: Noir
- 24V: Marron

Schéma de câblage:



Modules de sécurité:



XCSZ03
Capteur de porte



XCS PA591

Ce capteur est en 2 parties, la clé (XCSZ03) et la serrure (XCS PA591), lorsque la clé est clenched, ça fait contact, le courant passe, il n'y a pas d'arrêt d'urgence, lorsque la clé n'est pas enclenchée, le courant ne passe pas, il y a donc un arrêt d'urgence.



Preventa XPS AC



Arrêt d'urgence ZBE-101

Le préventa est un automate de sécurité, il possède plusieurs contacteurs à l'intérieur et permet de brancher en série plusieurs capteurs de sécurité et qu'ils soient tous gérés ensemble.

L'arrêt d'urgence est un bouton permettant de couper le courant dans le robot.

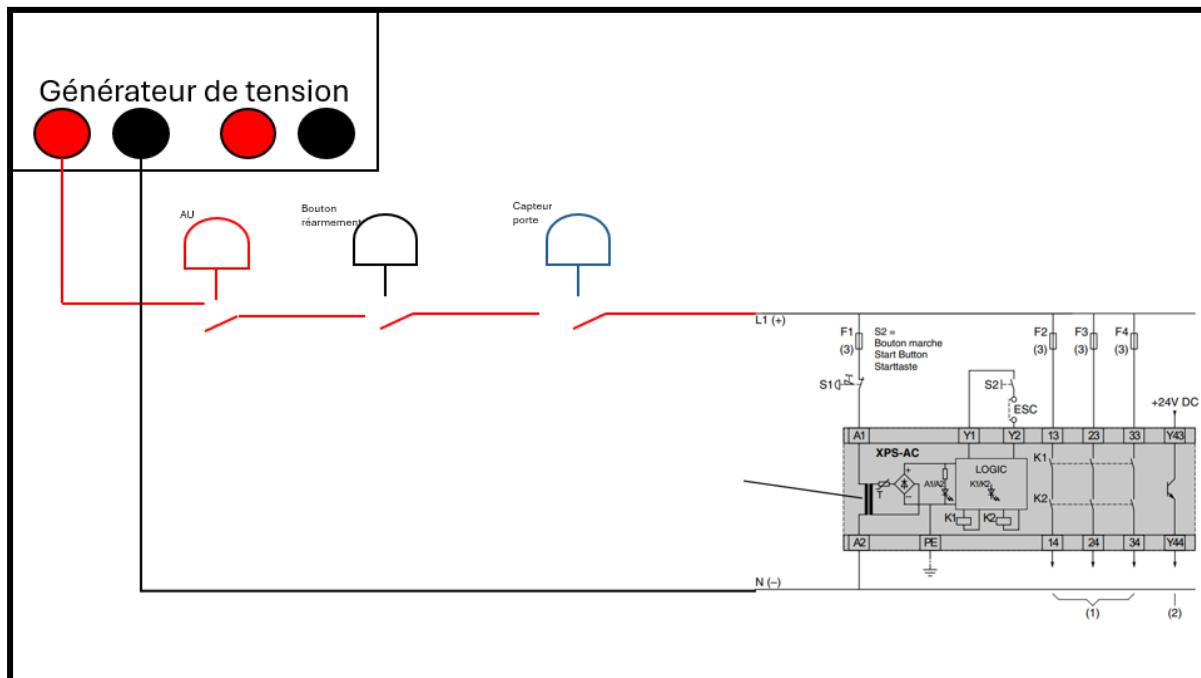


Bouton de réarmement

Le bouton de réarmement permet de remettre en marche le préventa, il permet de choisir de remettre sous tension le robot malgré que tous les arrêts d'urgence soient des-enclenchés. (équivalent du bouton restart sur le boîtier WMS).

Les boutons d'arrêt d'urgence, de réarmement et le capteur de porte sont à mettre en série sur le préventa.

Schéma de câblage des capteurs sur le préventa:



Programme Ajout capteur Porte pour Sécurité Machine :

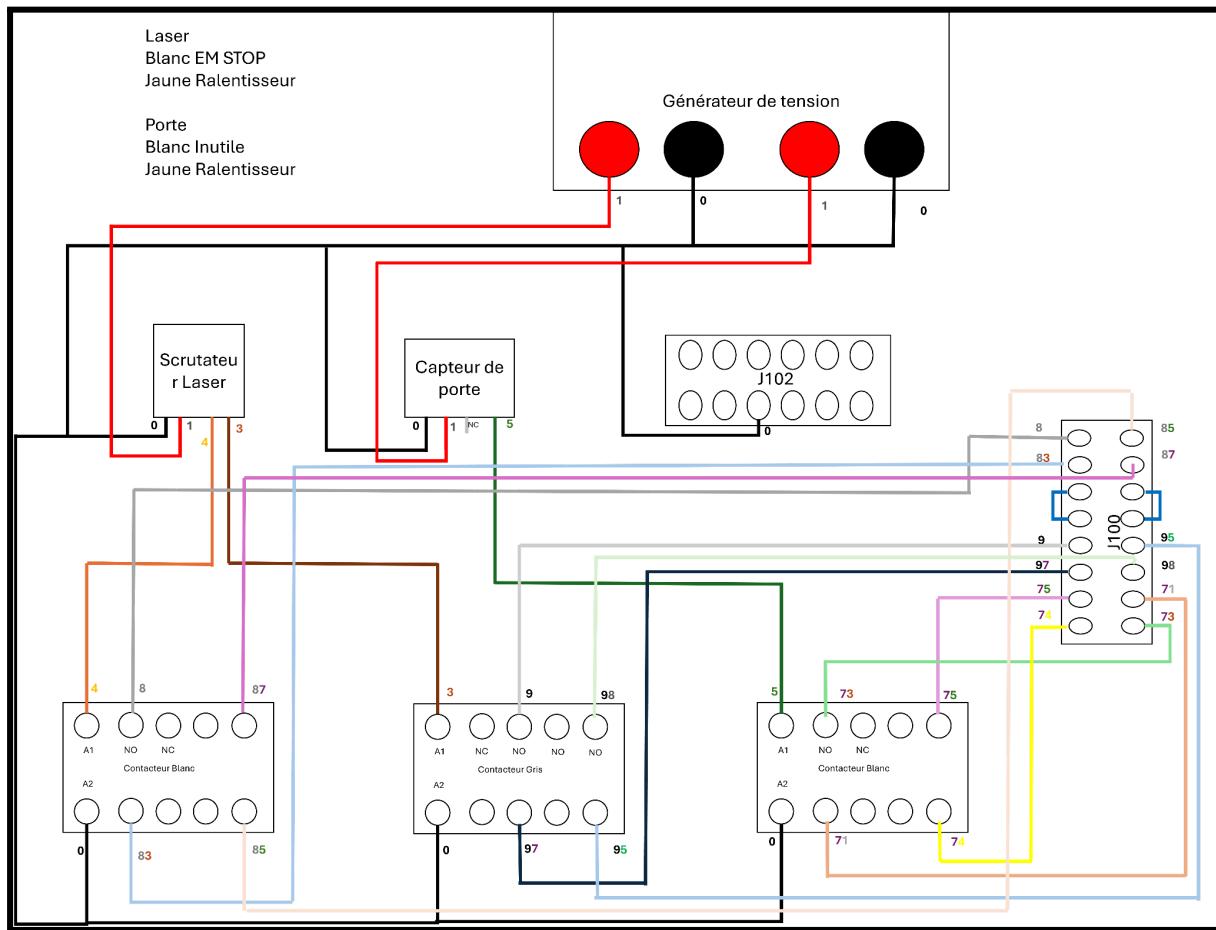
```

1 begin
2   while true
3     call CapteurPorte()
4   endWhile
5 end
  
```

<pre> 1 begin 2 call Initial() 3 4 jDest.j2 = 100 5 movej(jDest, flange, mNomSpeed) 6 waitEndMove() 7 call Speed() 8 9 jDest.j2 = 0 10 jDest.j3 = 90 11 movej(jDest, flange, mNomSpeed) 12 waitEndMove() 13 call Speed() 14 15 // Vers la droite 16 jDest.j1 = -80 17 movej(jDest, flange, mNomSpeed) 18 waitEndMove() 19 call Speed() 20 21 jDest.j3 = 0 22 jDest.j2 = 110 </pre>	<pre> 23 movej(jDest, flange, mNomSpeed) 24 waitEndMove() 25 call Speed() 26 27 // Vers la gauche 28 jDest.j2 = 0 29 jDest.j3 = 90 30 movej(jDest, flange, mNomSpeed) 31 waitEndMove() 32 call Speed() 33 34 jDest.j1 = -80 35 jDest.j3 = 0 36 movej(jDest, flange, mNomSpeed) 37 waitEndMove() 38 call Speed() 39 40 jDest.j3 = 90 41 movej(jDest, flange, mNomSpeed) 42 waitEndMove() 43 call Speed() 44 end </pre>
--	--

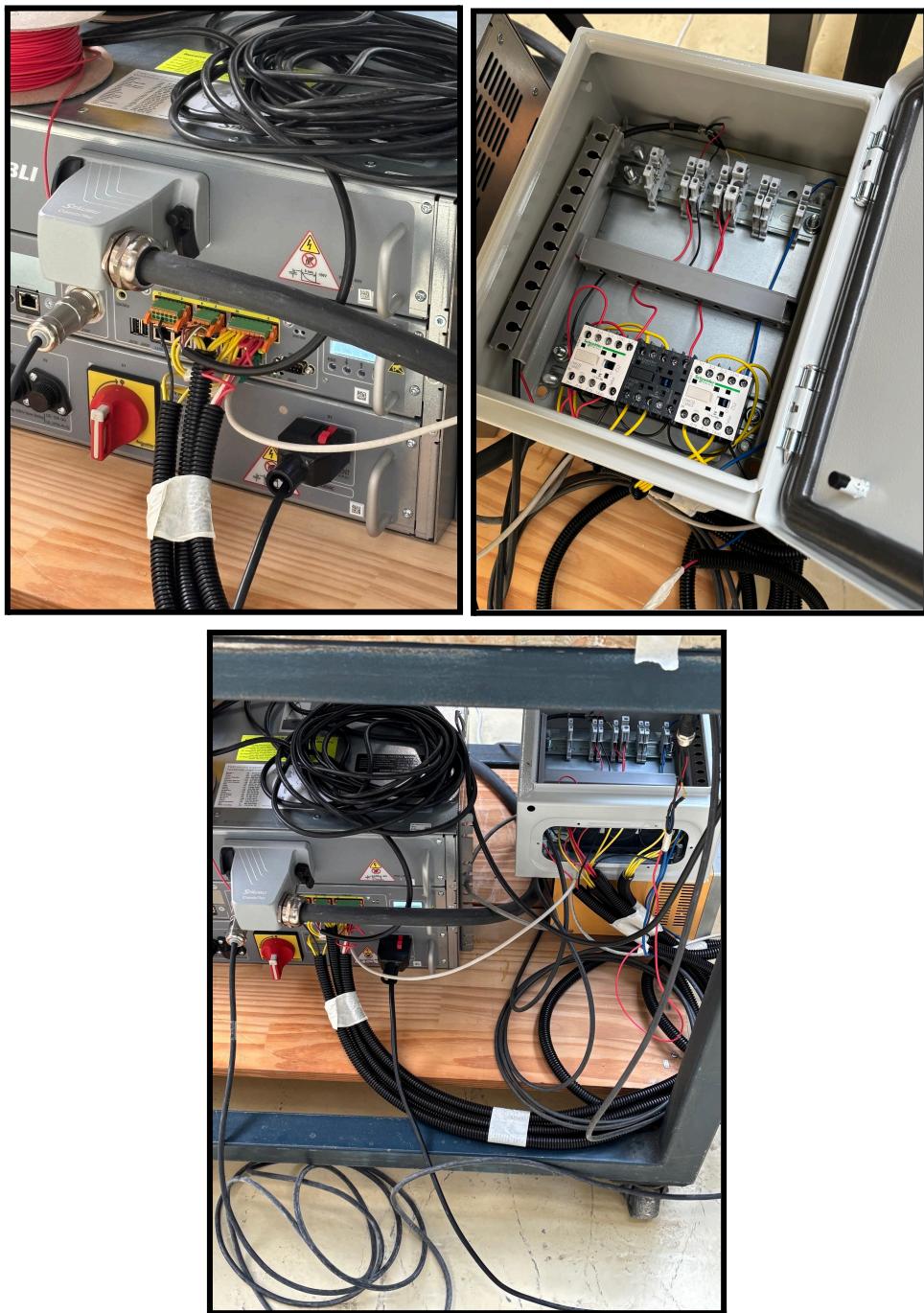
Armoire Électrique

I. Schématisation

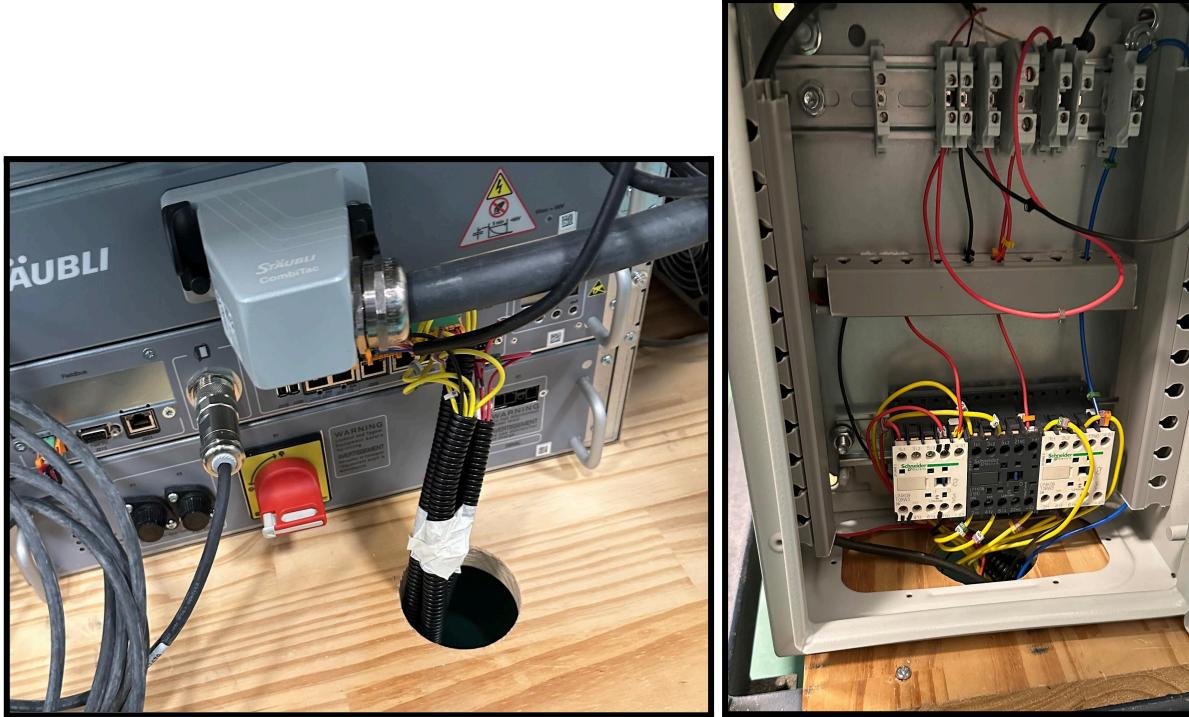


Tous les câbles ont été camouflés dans des gaines, les contacteurs ont été rangés dans un boîtier d'armoire électrique. Pour faciliter les connexions et éviter les court-circuits, les câbles ont été branchés sur des borniers.

II. Montage réel

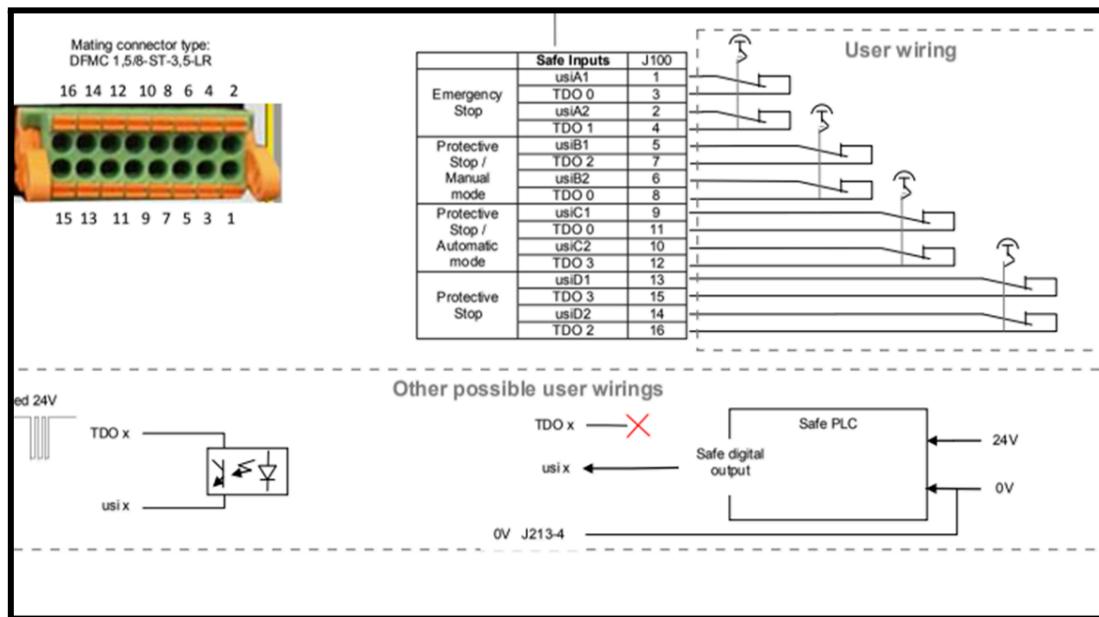


Nous avons par la suite fait des trous dans notre planche, afin de pouvoir y faire passer nos différents capables allant de l'armoire électrique à notre contrôleur.
 Ce qui rend le tout plus propre et rend un visuelle plus agréable, notamment avec l'armoire électrique qui est désormais positionnée à la verticale.



III. Implémentation/Connexion

Explication du connecteur J-100:



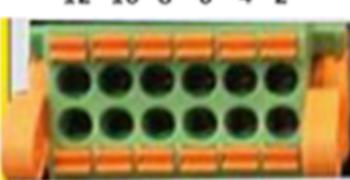
Le connecteur J-100 est divisé en 4 parties, sur chaque partie 2 entrées et 2 sorties. Exemple dans la partie Emergency Stop (TDO 0 et TDO 1 sont des sorties, USIA1 et USIA2 sont des entrées). De base les sorties sont shuntées sur les entrées (le signal de la sortie va sur l'entrée correspondante) et sont activées 2 à 2 et doivent changer d'états en même temps. (Si USIA1 change d'état USIA2 doit changer d'état en même temps).

Ainsi pour câbler un capteur sur le connecteur J-100, il faut que ce capteur contrôle interrupteur permettant de couper la connexion entre les 2 sorties et les 2 entrées, permettant ainsi de déclencher un changement d'état visible sur le logiciel SRS.

Attention: Ne pas shunter TDO 0 sur TDO 1 et USIA1 sur USIA2 cela a été déconseillé par Stäubli. (il faut forcément utiliser un contacteur avec 2 sorties et 2 entrées minimum).

Connecteur J-102:

Mating connector type:
DFMC 1,5/6-ST-3,5-LR



12	10	8	6	4	2
11	9	7	5	3	1

	Safe Outputs	J102
E-Stop status	usoA1	1
	usoA2	2
	0V	3
	0V	4
Auto/Manu status	usoB1	5
	usoB2	6
	0V	7
	0V	8
Enabling status	usoC1	9
	usoC2	10
	0V	11
	0V	12

Il faut relier une masse commune au contrôleur CS9, pour cela nous avons relié une masse au Pin 7 du connecteur J-102.

Essais de programmation sous Raspberry

L'objectif de cette partie était d'ajouter des capteurs externes reliés à une carte raspberry pi 5, nous permettant ainsi d'effectuer différentes actions de pilotage du robot en fonction de l'état des capteurs.

Pour nos essais, nous sommes parties sur un l'utilisation d'un télémètre à ultrason HC-SR04 reliés à une raspberry pi 5, nous avons ensuite branché cette raspberry via un cable RJ45 au contrôleur CS9 du bras robot pour essayer de lui envoyer des données.

Pour nous guider dans cette partie, nous avons trouvé un site expliquant une communication d'une raspberry avec une ancienne version du logiciel SRS.

https://bvd़p.inetdoc.net/wiki/doku.php?id=staubli#informations_sur_l_infrastructure_de_capteurs_communiquants

Code utilisé:

```
import RPi.GPIO as GPIO
import time
import socket

# Configuration des broches du capteur ultrasonique
TRIG = 23
ECHO = 24

GPIO.setmode(GPIO.BCM)
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)

# Configuration de l'envoi UDP
CS9_IP = "192.168.1.100" # Remplacer par l'adresse IP du contrôleur CS9
CS9_PORT = 5369 # Remplacer par le port configuré sur le CS9

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

def get_distance():
    """Mesure la distance à l'aide du capteur ultrasonique."""
    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)

    debut = time.time()
    while GPIO.input(ECHO) == 0:
        debut = time.time()
    while GPIO.input(ECHO) == 1:
        fin = time.time()
```

```
duree = fin - debut
distance = (duree * 34300) / 2
return round(distance, 2)
```

try:

```
    while True:
        distance = get_distance()
        message = f"{distance:.2f}" # Format texte
        sock.sendto(message.encode(), (CS9_IP, CS9_PORT))
        print(f"Distance envoyée : {message} cm")
        time.sleep(0.1) # Envoi toutes les 100 ms
```

except KeyboardInterrupt:

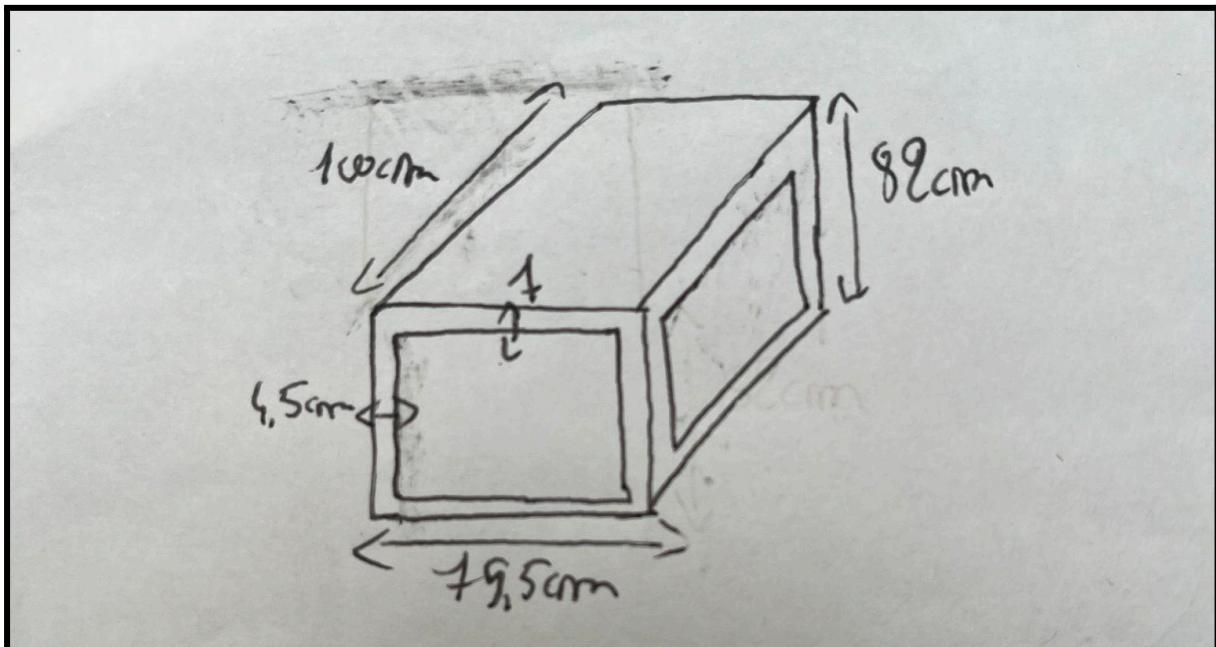
```
    print("\nArrêt du programme")
    GPIO.cleanup()
    sock.close()
```

La lecture de données fonctionne très bien, le télémètre nous repère, mais nous ne savons pas si le contrôleur reçoit bien les données car le contrôleur n'accepte pas 2 connexions via câble RJ45 et cette lecture se fait via le logiciel SRS.

Modélisation 3D

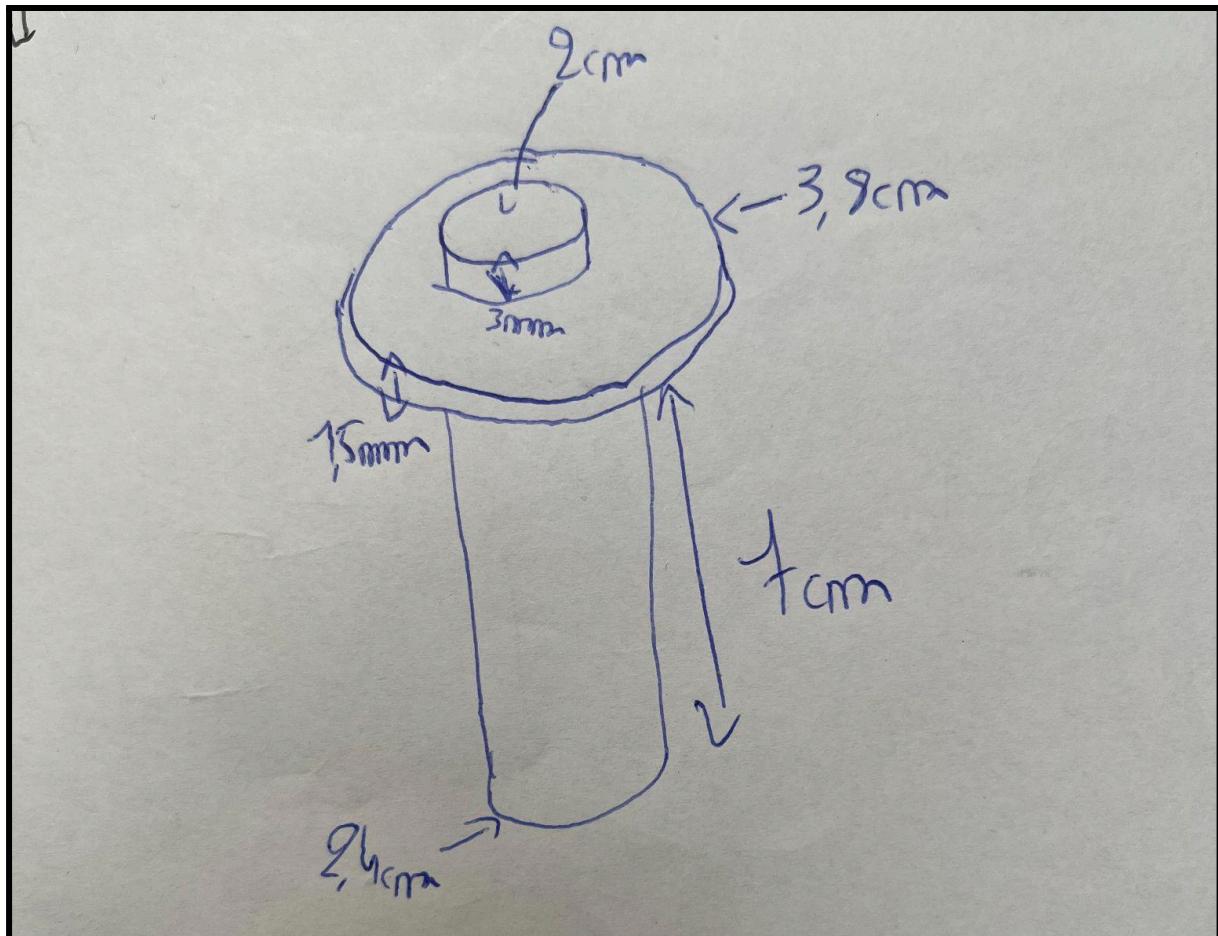
I. Schématisation

Pour pouvoir manipuler facilement le robot et faire comme si il était fixé dans la vraie vie on a dû modéliser la même table avec les mêmes dimensions pour pouvoir faire les tests en temps réel sur l'application. Tout d'abord on a dû prendre les mesures de tous les côtés.

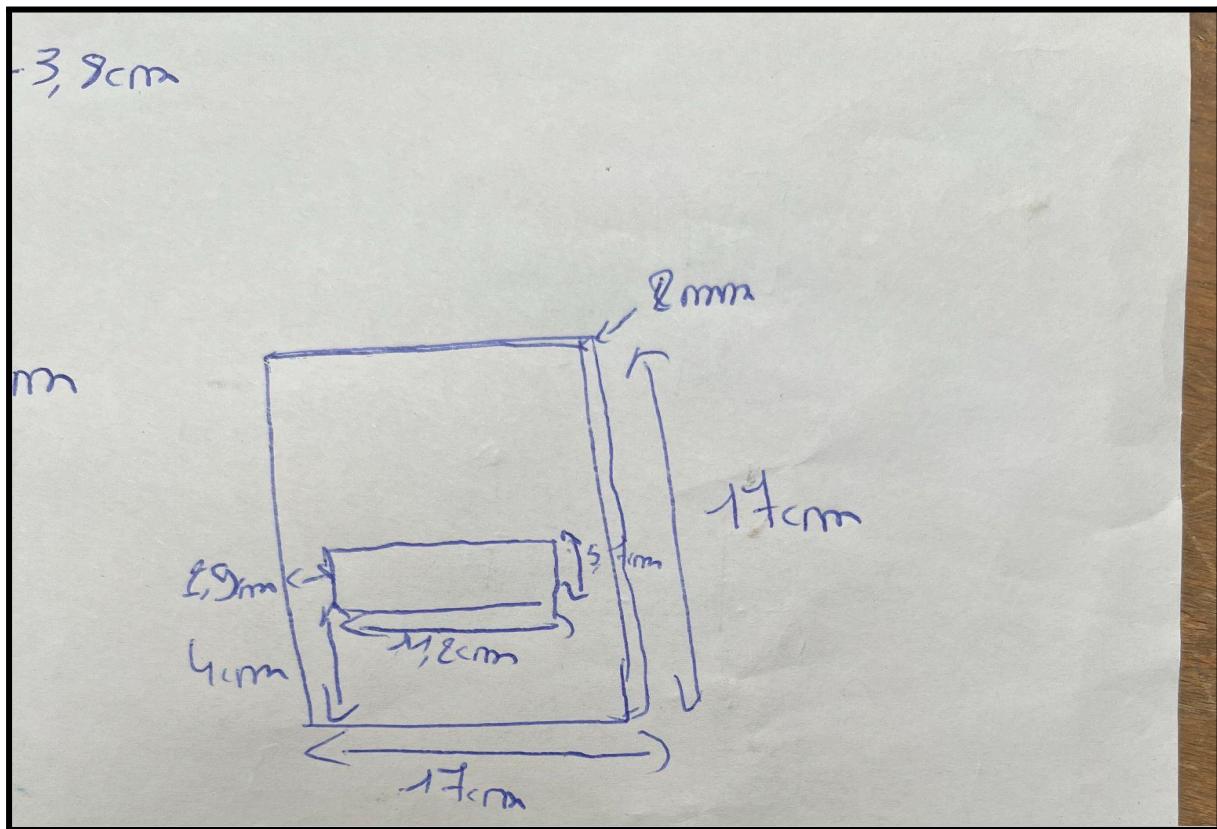


Après avoir fait ces mesures, on va la créer virtuellement en 3D grâce au logiciel Fusion 360.

Schématisation du support stylo :



Schématisation de la Plaque Staubli :



II. Fusion 360 (explication)

Fusion 360 est un logiciel de conception 3D, il permet de concevoir, modéliser, simuler, usiner et assembler dans un seul et même environnement des pièces.

I. Solid : Modélisation Solid



Create : extrusion, révolution, balayage, loft — création de volumes de base ou complexes.

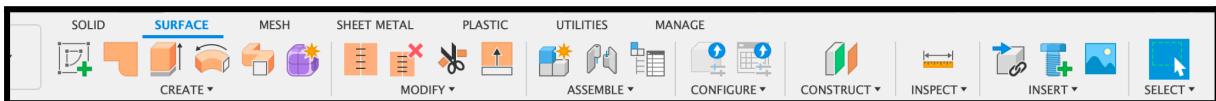
Modify : outils pour modifier la géométrie (chanfrein, congé, shell, offset face, etc.).

Assemble : insertion de composants et définition des liaisons mécaniques (rotule, pivot, coulisse, etc.).

Construct : création de plans, axes et points de référence pour positionner ou contraindre la géométrie.

Inspect : vérification des dimensions, angles, analyses d'interférences ou d'épaisseurs.

II. Surface : Modélisation surfacique



Utile pour concevoir des formes complexes et légères, notamment dans le design industriel ou les pièces plastiques.

Création de surfaces par extrusion, révolution, balayage, etc. Outils pour découper, étendre, raccommoder ou fusionner des surfaces.

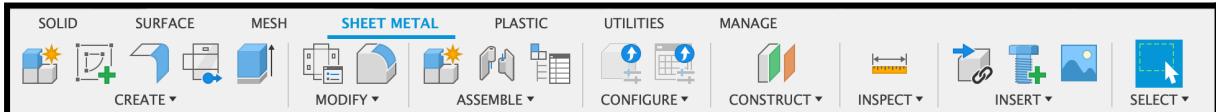
III. Mesh : Modélisation Maillée



Permet de manipuler des fichiers STL ou OBJ, souvent issus de scans 3D ou d'imprimantes.

Importation, réparation, conversion de maillages en corps solides. Idéal pour retravailler des modèles issus d'une autre source.

IV. Sheet Metal : Tôle



Spécifique à la conception de **pièces en tôle pliée**.

Outils pour créer des pliages, languettes, découpes et déplisés. Très utile pour des fabrications industrielles en découpe laser/plieuse.

V. Plastic : Conception Plastique

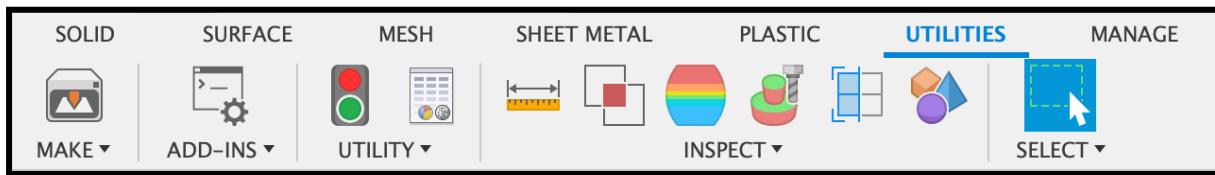


Outils dédiés à la modélisation de coques, clips, assemblages à emboîtement, etc.

Permet de concevoir des pièces injectées avec précision.

Fonctions comme "Snap Fit", "Bosses", "Ribs" très utilisées dans l'industrie électronique ou automobile.

VI. Utilities : Outils utilitaire

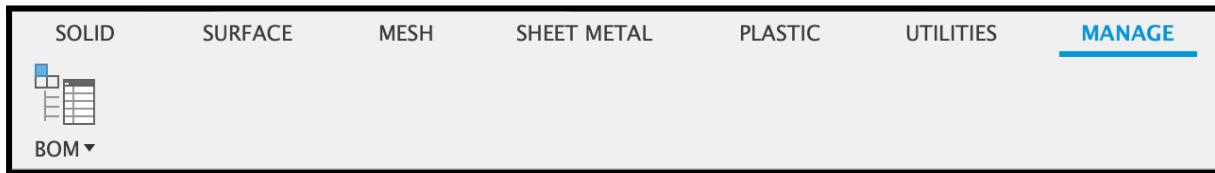


Outils transverses pour analyser, convertir ou optimiser votre conception :

Analyse de volume, masse, centre de gravité.

Exportations ou conversions de fichiers.

VII. Manage : Gestion de donnée



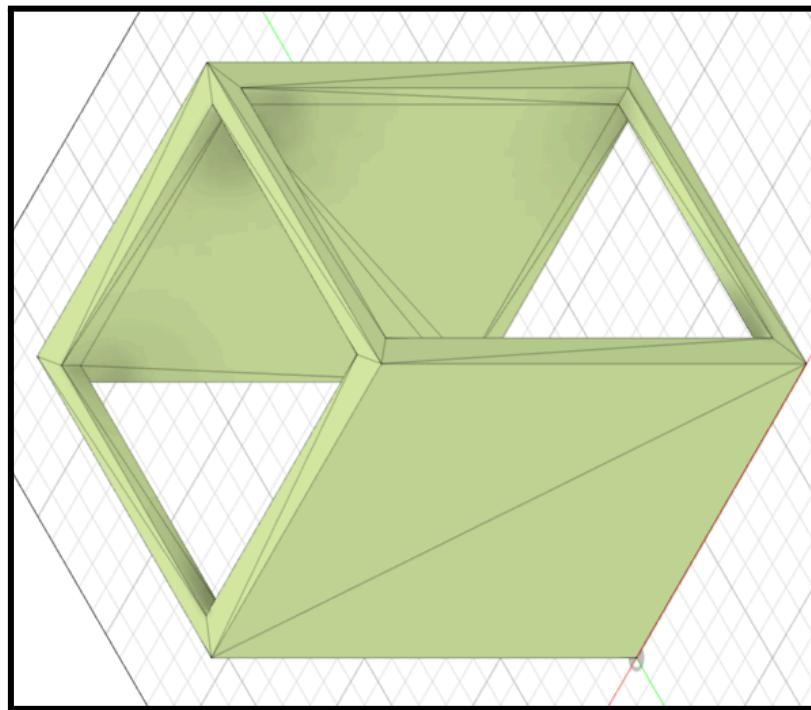
Gestion des liaisons entre composants :

Contrainte de positionnement (fixer, aligner, insérer).

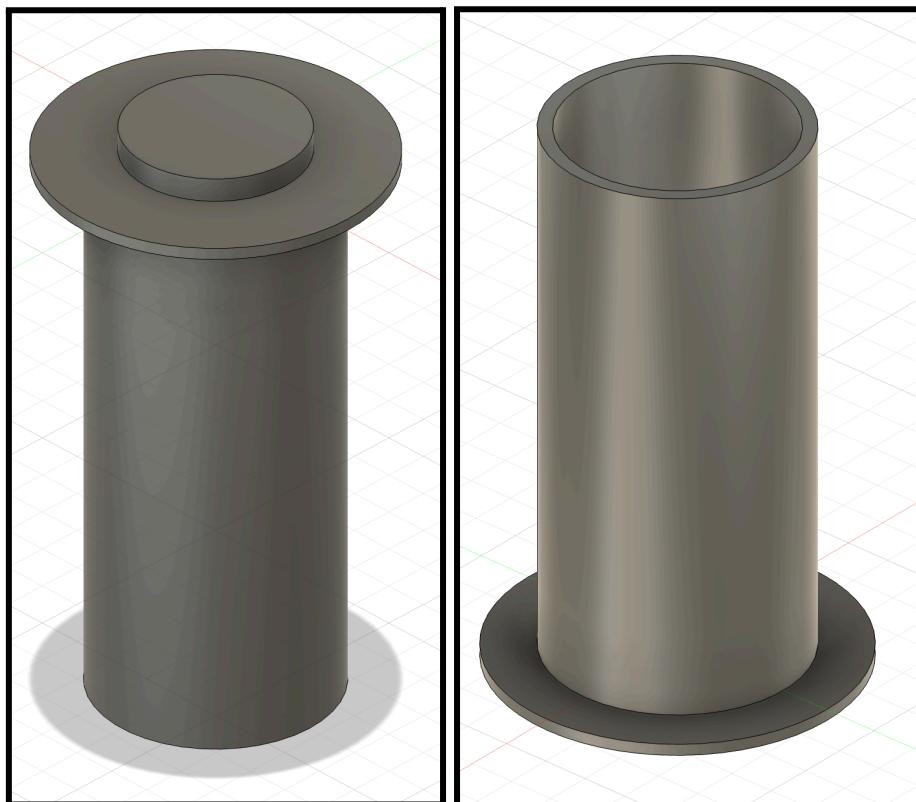
Simulation de mouvement ou de collision entre pièces.

III. Réalisation sur logiciel

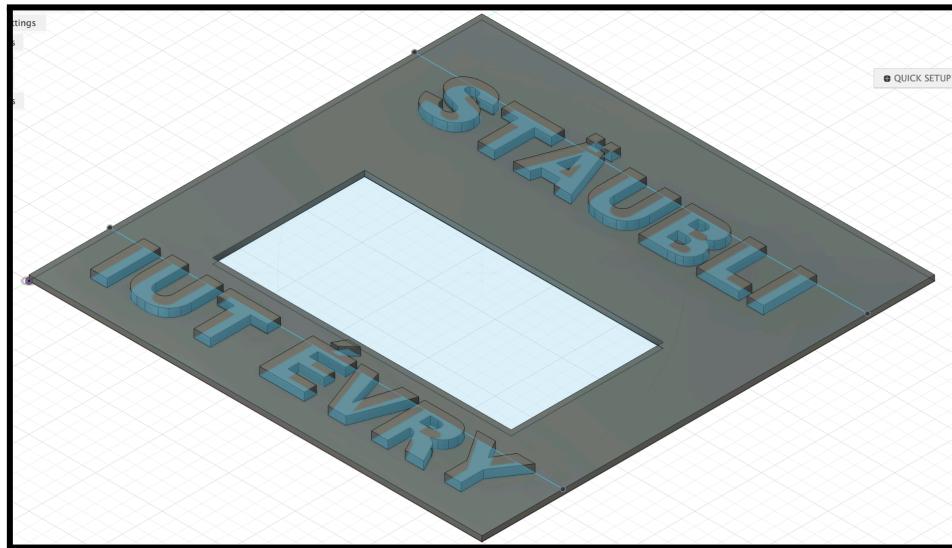
Modélisation 3D table:



Modélisation 3D Support stylo :

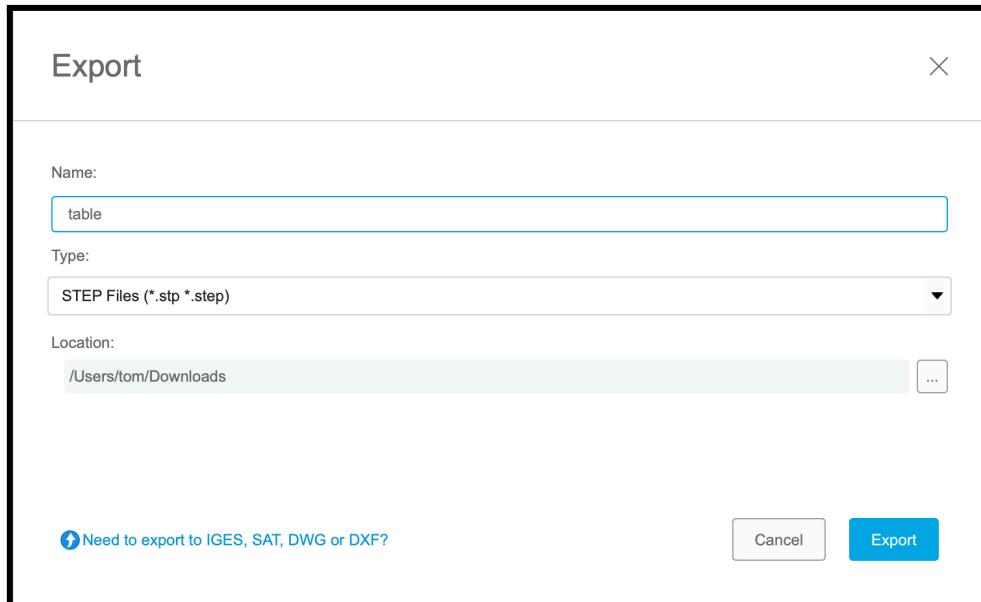


Modélisation 3D Plaque Staubli :

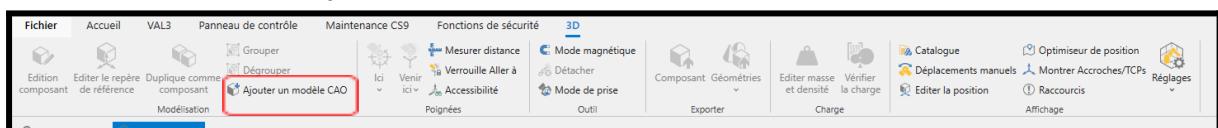


IV. Transfert sur SRS

Une fois que la table à été modélisée, il faut l'enregistrer sous fichier .step. Pour se faire, il faut exporter le projet et le mettre en fichier .step



Une fois que le fichier à été exporté, on va ouvrir le logiciel SRS, aller dans 3D dans la barre de menu et cliquer sur "Ajouter un modèle CAO".



Ouvrir le fichier et l'objet que vous avez sélectionné va apparaître dans le modèle 3D.

Exercice/Application

I. Déplacement simple

Pour effectuer des déplacements simples, nous devons faire utiliser la fonction movej, afin d'aller d'un point à un point, de déplacer des axes à notre souhait etc.

Voici un exemple de chaque pour les déplacements simple :

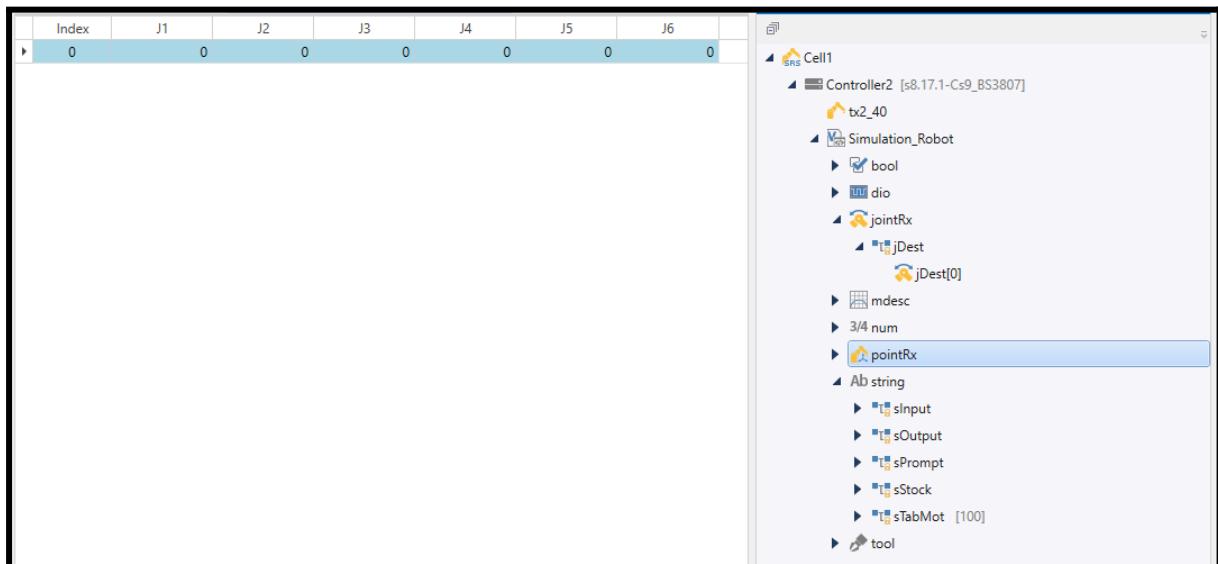
```

begin
  jDest.j2 = 88
  jDest.j5 = 56
  movej(jDest, flange, mNomSpeed)
  waitEndMove()
end
  
```

II. Variables / Données

Il y a un onglet Variables / données qui permet de déclarer des variables avec divers types, numérique, booléen, string etc.

Afin de pouvoir par exemple tester une limite de vitesse, une entrée sur le contrôleur, ou juste tester le contenu d'une variable.



Ici par exemple, nous pouvons voir les positions de nos différents axes du bras robotique via la variable jDest[0].

Ou alors nos variables numériques dans l'onglet num, nos chaîne de caractères dans l'onglet string, où nous avons fait une page html pour pouvoir permettre à l'utilisateur d'écrire un mot, que le robot va traiter avant de l'écrire sur la table.



Clavier vers le Robot



Stockage :

Mot souhaité :

Suivant

III. Fonctions

Nous avons également programmer via diverses fonctions comme :

- movej -> Déplacement libre et direct
- movel -> Déplacement linéaire
- movec -> Déplacement circulaire
- setMonitorSpeed -> Paramètre la vitesse du robot
- getMonitorSpeed -> Informe de la vitesse du robot
- enablePower -> Active la puissance du robot
- disablePower -> Désactive la puissance du robot

etc.

IV. Pick And Place

Programme Base / Pick and Place :

<pre> 1 begin 2 while true 3 call pick() 4 call place() 5 endWhile 6 end </pre>	<pre> 1 begin 2 // Place 1 3 4 // Leve le bras 5 jDest.j2 = 70 6 movej(jDest,flange,mNomSpeed) 7 8 // deplace a cote 9 jDest.j1 = -87.29 10 movej(jDest,flange,mNomSpeed) 11 12 // descend le bras 13 jDest.j2 = 83.08 14 movej(jDest,flange,mNomSpeed) 15 16 open(Ventouse) 17 18 waitEndMove() 19 end </pre>
---	--

Ce pick and place récupère un objet dans son embout au niveau de la tour, et le dépose dans le bac qui se trouve en dessous de la table.

Nous avons fait divers pick and place, que l'on peut retrouver directement dans la soutenance avec diverses démonstrations.

Programme Tower/ Pick and Place :

```

1   begin
2   □ while true
3       call PickPlace2Dingue()
4   endWhile
5   end

1 begin
2 call Speed()
3 // Pick Tower
4 jDest.j1 = 107.29
5 jDest.j2 = 115.27
6 jDest.j3 = -30.54
7 jDest.j4 = -37.06
8 jDest.j5 = -62.74
9 jDest.j6 = 0
10 movej(jDest,flange,mNomSpeed)
11 waitEndMove()
12
13 delay(2)
14 call Speed()
15
16 // Releve
17 jDest.j1 = 107.29
18 movej(jDest,flange,mNomSpeed)
19 jDest.j4 = 0
20 jDest.j5 = -80

21 movej(jDest,flange,mNomSpeed)
22 waitEndMove()
23 call Speed()
24
25 // Pose Tower
26 jDest.j1 = -138
27 jDest.j2 = 118.77
28 jDest.j3 = -34
29 jDest.j5 = -106.14
30 movej(jDest,flange,mNomSpeed)
31 waitEndMove()
32 call Speed()
33
34 jDest.j4 = -121.13
35 movej(jDest,flange,mNomSpeed)
36 waitEndMove()
37 call Speed()
38 end

```

Celui-ci récupère un objet (une clé usb par exemple) dans une tour que nous avons créé et va déposer cet objet dans un trou dans la table où nous avons mis un bac en dessous pour stocker l'objet.



Programme tourni / Pick and Place :

```

1   begin
2   □ while true
3   |   call Tourni()
4   |   endWhile
5   end

1  begin
2   call Speed()
3   // Pick Tower
4   jDest.j1 = 107.29
5   jDest.j2 = 115.27
6   jDest.j3 = -30.54
7   jDest.j4 = -37.06
8   jDest.j5 = -62.74
9   jDest.j6 = 0
10  movej(jDest,flange,mNomSpeed)
11  waitEndMove()
12
13 | delay(2)
14 | call Initial()
15
16 | movej(pTourni[0],flange,mNomSpeed)
17 | waitEndMove()
18
19 | movej(pTourni[1],flange,mNomSpeed)
20 | waitEndMove()
21
22 | movej(pTourni[2],flange,mNomSpeed)
23 | waitEndMove()

24 movej(pTourni[3],flange,mNomSpeed)
25 waitEndMove()
26
27 delay(0.5)
28
29 movej(pTourni[1],flange,mNomSpeed)
30 waitEndMove()
31
32 movej(pTourni[3],flange,mNomSpeed)
33 waitEndMove()
34
35 // Pose Tower
36 jDest.j1 = -138
37 jDest.j2 = 118.77
38 jDest.j3 = -34
39 jDest.j5 = -106.14
40 movej(jDest,flange,mNomSpeed)
41 waitEndMove()
42 call Speed()
43
44 jDest.j4 = -121.13
45 movej(jDest,flange,mNomSpeed)
46 waitEndMove()
47
48 end

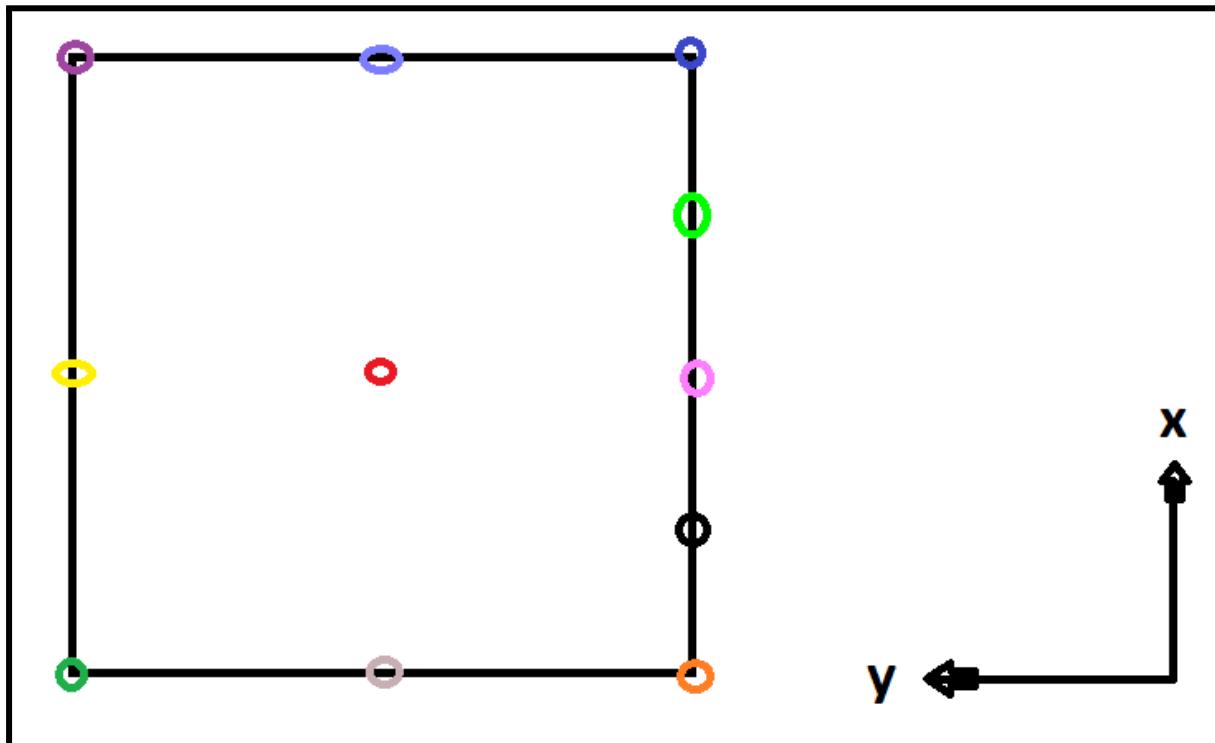
```

Ce programme fait le même mécanisme que celui d'avant, mais va faire tourner un plan dans l'espace pour faire tourner notre objet, ici nous avions mis une balle de baby foot, et celui-ci tenait tellement.

Puis va ranger la balle comme, pour la clé.

V. Lettre / Chiffre

Création d'une matrice pour créer un tableau avec les points de repère.



Index	X	Y	Z	Rx	Ry	Rz	Shoulder	Elbow	Wrist
0, 0	275	-100	-194	0	-180	0	ssame	esame	wsame
0, 1	275	-75	-194	0	-180	0	ssame	esame	wsame
0, 2	275	-50	-194	0	-180	0	ssame	esame	wsame
0, 3	290	-50	-194	0	-180	0	ssame	esame	wsame
1, 0	300	-100	-194	0	-180	0	ssame	esame	wsame
1, 1	300	-75	-194	0	-180	0	ssame	esame	wsame
1, 2	300	-50	-194	0	-180	0	ssame	esame	wsame
1, 3	310	-50	-194	0	-180	0	ssame	esame	wsame
2, 0	325	-100	-194	0	-180	0	ssame	esame	wsame
2, 1	325	-75	-194	0	-180	0	ssame	esame	wsame
2, 2	325	-50	-194	0	-180	0	ssame	esame	wsame
2, 3	310	-100	-194	0	-180	0	ssame	esame	wsame

Vous pourrez retrouver ci-dessus, les différentes coordonnées de nos points par rapport à notre matrice.

Puis, pour dessiner une lettre, nous aurons juste à appeler la lettre où que l'on souhaite faire dans le programme principal “start()”, et pour dessiner la lettre, nous aurons un sous-programme qui part d'un point extrême de notre lettre puis se déplace en linéaire si besoin ou même un arrondie.

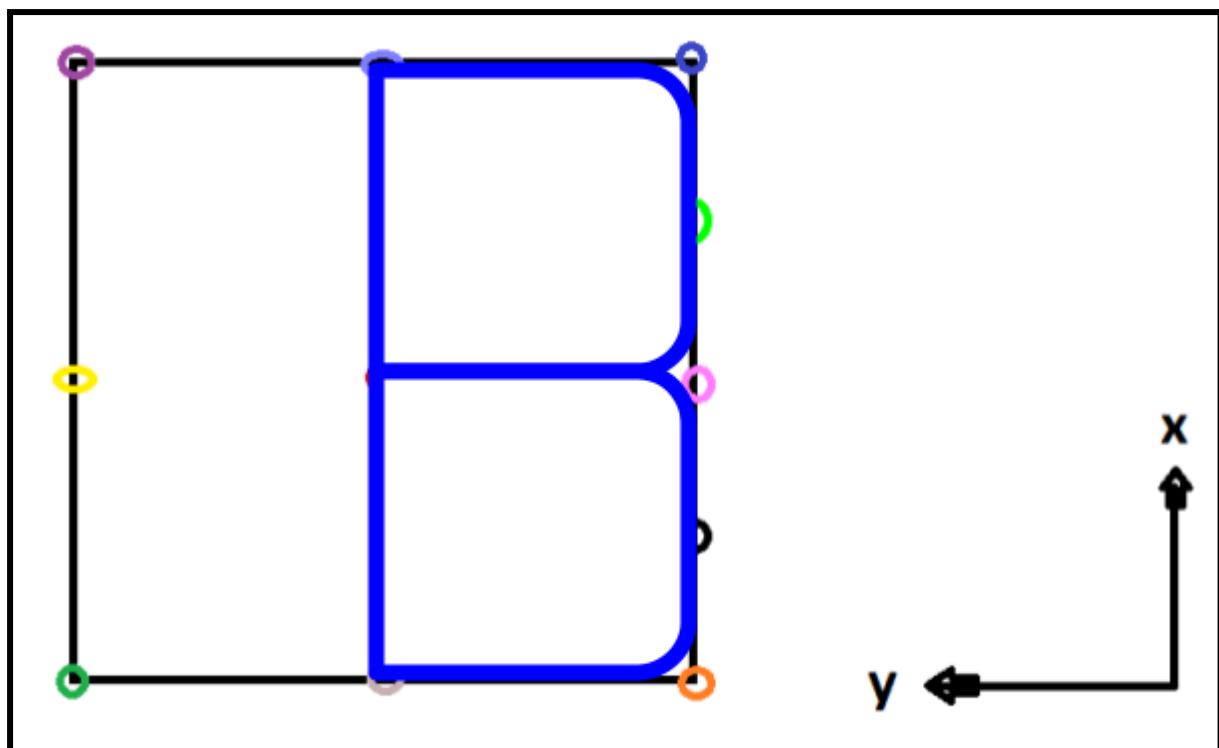
Nous avons donc réalisé tous les cas possibles, les 26 lettres de l'alphabet, les chiffres, les espaces, décalage etc.

Par exemple : Lettre B

```

begin
    // Contour du B
    movej(pLettre[0,1], Ventouse, mNomSpeed)
    waitEndMove()
    call Speed()
    movec(pLettre[0,3], pLettre[1,1], Ventouse, mNomSpeed)
    waitEndMove()
    call Speed()
    movec(pLettre[1,3], pLettre[2,1], Ventouse, mNomSpeed)
    waitEndMove()
    call Speed()
    movel(pLettre[0,1], Ventouse, mNomSpeed)
    waitEndMove()
    call Speed()
    call Leve_bras()
    call Speed()
    call Decallage()
end

```



avec le programme pour décalage :

```

1 begin
2     // Décalage de la taille des matrices
3     nY1 = nY1 + 60
4     nY2 = nY2 + 60
5     nY3 = nY3 + 60
6     pLettre[0,0].trsfc.y = nY1
7     pLettre[0,1].trsfc.y = nY2
8     pLettre[0,2].trsfc.y = nY3
9     pLettre[0,3].trsfc.y = nY3
10
11    pLettre[1,0].trsfc.y = nY1
12    pLettre[1,1].trsfc.y = nY2
13    pLettre[1,2].trsfc.y = nY3
14    pLettre[1,3].trsfc.y = nY3
15
16    pLettre[2,0].trsfc.y = nY1
17    pLettre[2,1].trsfc.y = nY2
18    pLettre[2,2].trsfc.y = nY3
19    pLettre[2,3].trsfc.y = nY3
20 end

```

Programme Test Lettre :

```

1 begin
2   while true
3     call TestLettre()
4     call D()
5   endWhile
6 end

```

```

1 begin
2   switch sTabMot[nVar]
3     // Majuscule
4     case "A"
5       call A()
6       break
7     case "B"
8       call B()
9       break
10    case "C"
11      call C()
12      break
13    case "D"
14      call D()
15      break
16    case "E"
17      call E()
18      break

```

Ce programme teste chaque lettre entrée dans le clavier via l'IHM (si elles sont présentes dans le mot) et avec un switch case elles sont testées, que ce soit en majuscule ou bien en minuscule, tous les cas sont testés.

Avec la fonction call, un sous programme est appelé, correspondant à la lettre entrée, la lettre peut ainsi être dessinée

Programme Test Chiffre :

```

1   begin
2   □   while true
3   █   call TestChiffre()
4   █   call Cinq()
5   █   endWhile
6   █   end

```

```

1   begin
2   □   switch sTabMot[nVar]
3   █   case "1"
4   █     call Un()
5   █     break
6   █   case "2"
7   █     call Deux()
8   █     break
9   █   case "3"
10  █     call Trois()
11  █     break
12  █   case "4"
13  █     call Quatre()
14  █     break
15  █   case "5"
16  █     call Cinq()
17  █     break

```

Ce programme teste un chiffre qui est entré dans notre clavier via la page HTML, est testé dans la fonction switch case, et va appeler un sous-programme du chiffre qui correspond.

Ainsi, le robot va écrire le chiffre en question.

Programme Clavier :

Nous avons également créé un programme, qui ouvre une page IHM, demandant à l'utilisateur d'écrire un mot pour que le robot puisse l'afficher, et l'écrire sur la table.

```

1 begin
2   call Clavier()
3   call TestChiffre()
4   call TestLettre()
5 end
  
```

```

1 begin
2   // sPrompt = "Ecrit une valeur : "
3   sInput = ""
4   bExit = false
5
6   nRet = userPage("main")
7   if(userPagePrompt(sPrompt,sInput) == true)
8     sOutput = sInput
9   else
10    sOutput = "User canceled prompt"
11   endIf
12   while bExit == false
13     delay(0)
14   endWhile
15 end
  
```

Dans ce programme, nous avons des inputs, qui ouvrent un clavier permettant de récupérer une donnée, elle sera stockée dans une chaîne de caractère et sera traitée par la suite dans nos testChiffre ou testLettre.

```

1 begin
2     sPrompt = "Mot souhaité ? "
3     call Clavier()
4     sStock = sOutput
5
6     sPrompt = "Combien de lettres du coup ? "
7     call Clavier()
8
9     nVar = 0
10    switch sOutput
11        case "1"
12            nStock = 1
13            break
14        case "2"
15            nStock = 2
16            break
17        case "3"
18            nStock = 3
19            break
20        case "4"
21            nStock = 4
22            break
23        case "5"
24            nStock = 5
25            break
26        case "6"
27            nStock = 6
28            break
29    endSwitch
30
31    case "7"
32        nStock = 7
33        break
34    case "8"
35        nStock = 8
36        break
37    case "9"
38        nStock = 9
39        break
40    default
41        nStock = 0
42        break
43    endSwitch
44
45    while(nVar < nStock)
46        sPrompt = "Donne les moi, une par une s'il te plait bg :"
47        call Clavier()
48
49        sTabMot[nVar] = sOutput
50
51        nVar = nVar + 1
52    endwhile
53
54    nVar = 0
55
56    while(nVar < nStock)
57        call TestChiffre()
58        call TestLettre()
59
60        nVar = nVar + 1
61    endwhile
62
63 end

```

Livrable

Nous avons eu la chance d'être les premiers à travailler sur ce robot, ce qui nous a permis d'explorer un domaine entièrement nouveau. En partant de zéro, nous avons dû apprendre à programmer et à simuler le robot sous SRS, en passant par des phases de tests, d'expérimentations et de découvertes continues. Chaque étape a été l'occasion d'approfondir notre compréhension et d'adapter nos choix pour optimiser le fonctionnement du robot.

Étant donné que nous avons été les pionniers sur ce projet, nous avons pris soin de documenter notre travail à travers un livrable détaillé. Ce livrable constitue non seulement un récapitulatif de notre démarche, mais aussi un véritable guide pour les futures équipes qui souhaiteraient reprendre ou étendre le projet. Il présente nos avancées, nos choix techniques, ainsi que les ajustements effectués tout au long du développement.

Ce livrable est conçu pour être une ressource complète, facilitant la prise en main du robot et permettant de comprendre le processus de programmation et de simulation. Il offre aux futurs utilisateurs une base solide pour effectuer des modifications, ajuster le robot à de nouveaux besoins ou simplement comprendre les choix et solutions qui ont été apportés lors de la création du projet.

Ainsi, ce document pourra être utilisé non seulement comme support d'apprentissage, mais aussi comme référence pour ceux qui souhaitent améliorer ou réutiliser le travail accompli, en offrant un aperçu complet des possibilités du robot et des ajustements réalisés.

TP Exercices et Correction pour les prochaines années

Nous avons réalisé une série d'exercices pratiques visant à programmer et simuler un robot Staubli sous SRS. Ces exercices ont couvert des aspects essentiels de la robotique industrielle, tels que la manipulation d'objets, la gestion de la vitesse en fonction des capteurs, le traçage de lettres et de chiffres, ainsi que l'intégration du robot dans un environnement simulé.

Chaque exercice a été soigneusement documenté et exécuté, avec des corrections et ajustements réalisés au fur et à mesure pour garantir la précision et l'efficacité des opérations. Ces corrections ont permis d'optimiser les performances du robot, d'assurer la cohérence des déplacements et d'adapter les actions aux conditions de l'environnement simulé.

Le TP a été conçu pour être un support pédagogique réutilisable, et peut servir comme référence pour d'autres projets ou démonstrations similaires. Les corrections apportées durant les exercices, notamment celles liées à la gestion des capteurs de sécurité et au calibrage du robot, permettent de mieux comprendre comment adapter un robot à des scénarios réels et dynamiques.

Ainsi, ce TP et ses corrections peuvent être utilisés comme base pour des présentations, des analyses ou des formations futures sur la robotique et la simulation de robots industriels.

Diaporama Soutenance

Dans le cadre de notre projet, nous avons conçu un diaporama détaillé présentant l'ensemble de notre travail. Ce diaporama inclut une introduction au projet, une présentation du robot utilisé, ainsi qu'une démonstration du robot en action à travers divers exercices.

Ces démonstrations illustrent concrètement le fonctionnement du robot et les différentes tâches qu'il est capable d'accomplir dans un environnement simulé. Le diaporama est conçu pour être un support complet et réutilisable, pouvant servir à la fois pour la présentation du projet et comme outil de référence pour de futures démonstrations ou explications.

Il pourra également être utilisé comme support pédagogique ou de formation, permettant à toute personne souhaitant comprendre le fonctionnement du robot, du logiciel et des différents exercices de suivre pas à pas l'évolution du projet.

Ainsi, ce diaporama peut également être utilisé comme référence dans des situations où une présentation ou un retour sur les résultats obtenus serait nécessaire.

Appels Stäubli

Dans le cadre de ce projet, nous avons dû à plusieurs reprises faire appel à Stäubli afin de comprendre le fonctionnement des différents composants du robot. Nous avons ainsi appelé Stäubli pour les raisons suivantes:

- Nous aider à comprendre le connecteur J-100 afin d'y ajouter des capteurs de sécurité externes et pouvoir effectuer différentes actions sur ce dernier (Arrêt d'urgence, ralentissement du bras robot).
- Nous aider à installer la licence supplémentaire SafeCell+ afin de pouvoir effectuer d'autres actions via le connecteur J-100.
- Enfin essayer de piloter le robot via une raspberry, la personne appelée qui s'occupe de la partie application, ne savait pas comment piloter un contrôleur via une raspberry et nos différents essais se sont révélés infructueux, le contrôleur n'acceptait pas 2 connexions via câble RJ45 en simultané.

Conclusion

L'ensemble des objectifs du projet a été atteint avec succès, garantissant une installation fonctionnelle et un environnement d'apprentissage optimisé pour les futurs utilisateurs. Le bras Stäubli TX2-40 est désormais pleinement opérationnel, équipé de tous les dispositifs de sécurité nécessaires, et capable d'exécuter des tâches variées allant de la simple manipulation d'objets à l'écriture de lettres et de chiffres.

L'installation physique a été menée avec rigueur : la table roulante a été aménagée pour accueillir le robot, son contrôleur CS9, ainsi que l'armoire électrique regroupant l'ensemble des contacteurs et câblages nécessaires. Grâce à une organisation minutieuse des composants et un câblage optimisé, nous avons assuré une structure fiable et ergonomique.

Côté logiciel, nous avons su exploiter les capacités du logiciel SRS, en intégrant des simulations 3D, en développant des programmes de gestion des capteurs de sécurité, et en optimisant la gestion des vitesses et des trajectoires du robot. L'ensemble du travail réalisé a été documenté dans un livrable complet, accompagné de travaux pratiques corrigés, garantissant une transmission efficace des connaissances aux promotions suivantes.

Ce projet représente ainsi une réalisation aboutie, offrant une solution clé en main pour l'apprentissage de la robotique industrielle avec un robot Stäubli. Il constitue une base solide pour d'éventuelles évolutions, comme l'intégration de nouveaux capteurs, l'optimisation des algorithmes de trajectoire ou encore le développement de nouvelles applications pédagogiques.

Améliorations envisagées

- Réussir à implémenter un contrôle du robot avec une raspberry.
- Ajout d'un cadena de sécurité
- Fixation des câbles sur la table
- Ajout porte / capteurs Clé porte XCSZ03, XCS PA591
- Ajout d'une pince au bout du robot avec mécanisme de moteur
- Impression de cube pour faire un pick and place
- Ajout d'une planche pour écrire sur une feuille
- Remplacer le bloc d'alimentation par une alimentation 24V implanté dans l'armoire en direct
- Système de rail pour capteur de porte
- Ajout de planche partie basse de la table pour enfermer la partie puissance et commande
- Ajouter si possible (trouver une autre solution que raspberry) une caméra pour faire asservissement visuel
- Fixation du scrutateur laser (système rail possible pour déplacer à notre guise le capteur)
- Ajout de capteurs supplémentaires
- Implantation d'un automate de sécurité pour mettre tous les capteurs