PROJET MAJEURE - ROBOTIQUE DE SERVICE CPE LYON 2020/2021 5IÈME ANNÉE

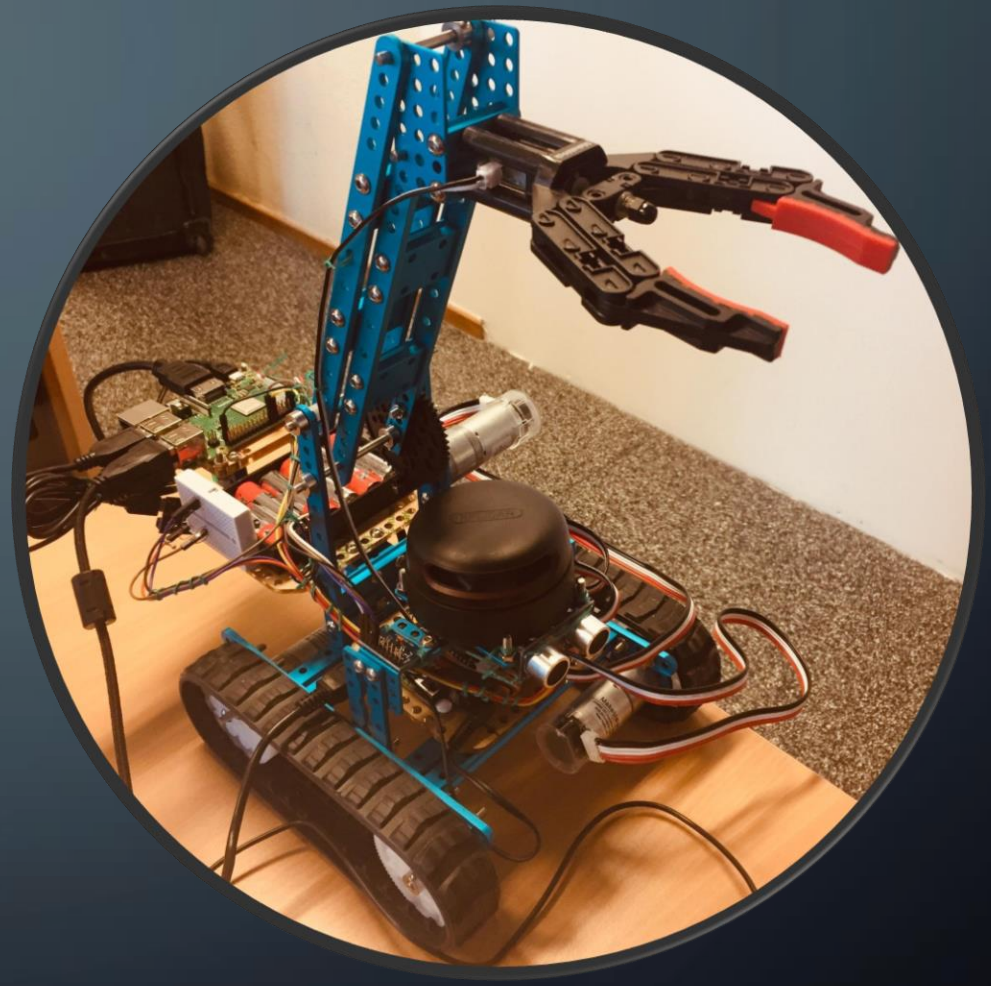
SUJET 5: AIDE À LA CONCEPTION D'UN PROTOTYPE DE ROBOT AFIN DE RÉALISER UN SCÉNARIO DE ROBOTIQUE DE SERVICE PROPOSÉ PAR LA COMPÉTITION WORLDSKILLS

GROUPE 7: BRIAND/GUY/KAHAN/MARTINEZ

ENCADRANTS: RAPHAEL LEBER & FABRICE JUMEL

SOMMAIRE:

- Le projet
- L'architecture hard/soft
- Configuration master
- Capteurs navigation
- Vision
- Implémentation sur robot réel
- IHR et HIL-3D
- Vidéo



LE PROJET

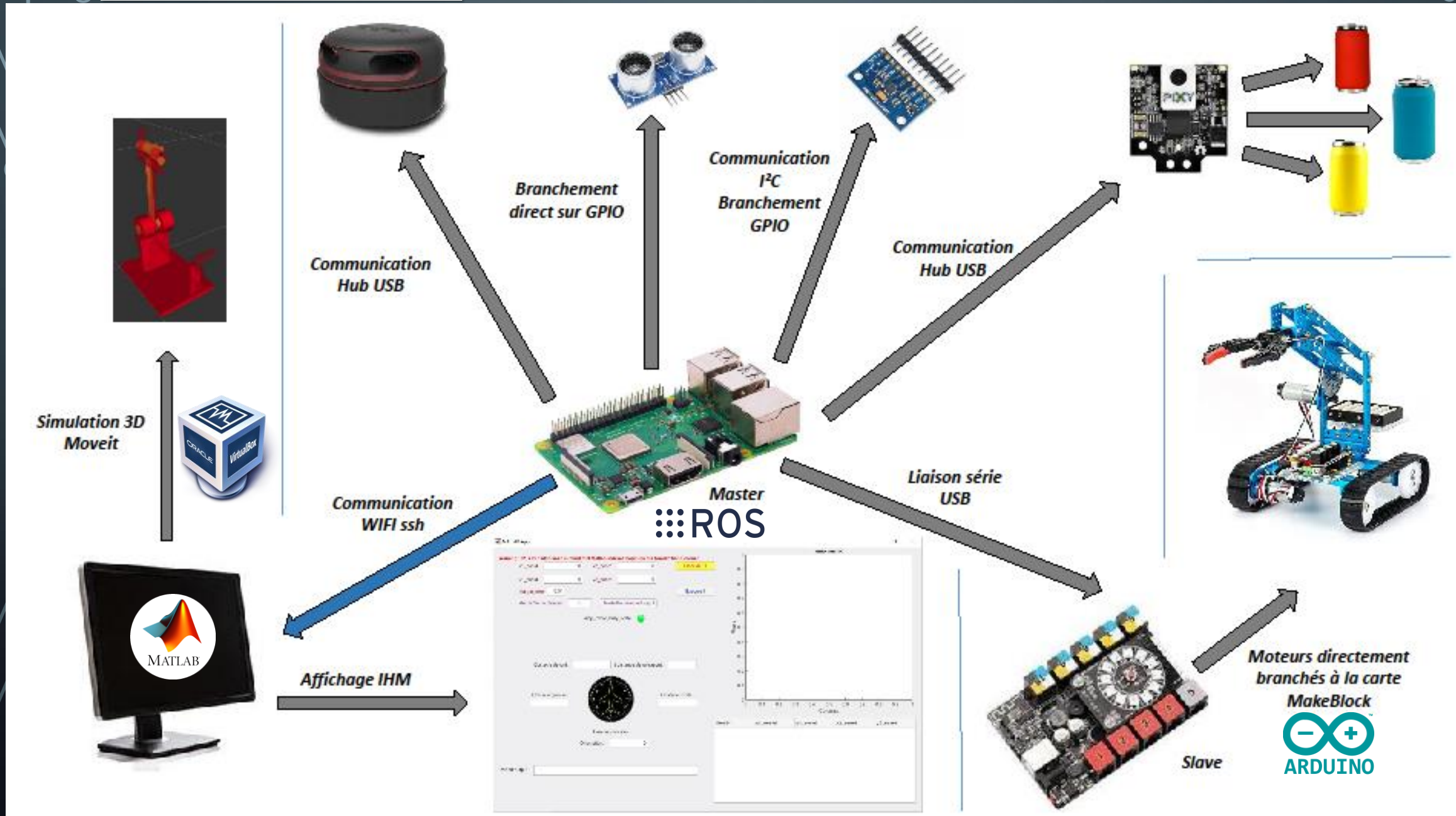
SCENARIO WORLDSKILLS

- Un robot autonome
- Un labyrinthe
- Des zones de couleurs
- Des cannettes de couleurs
- Visualisation de son états et de l'environnement

CAHIER DES CHARGES

- Robots avec capteurs
- Robot capable de se déplacer dans un environnement connu
- Capable de reconnaitre les couleurs
- Capable de reconnaitre des cannettes
- Capable d'informer sur son état et l'environnement
- Afficher les informations

ARCHITECTURE



CONFIGURATION CARTE MASTER

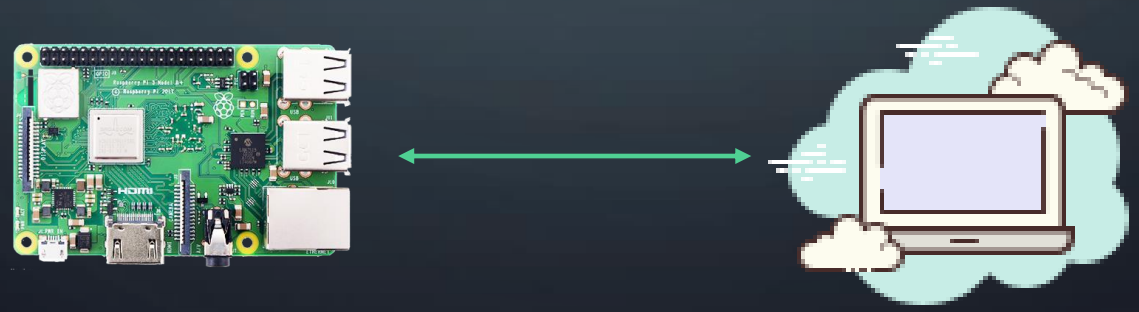
CONFIGURATION DE LA CARTE MASTER

OBJECTIFS

- Contrôler la carte à distance et lancer l'application
- Communiquer avec le PC pour mettre à jour des informations sur l'états du robot et son environnement

CONTRAINTES

- Supporter certains package ROS
- Connection ssh avec le PC
- Supporter Opencv pour darknet



CONFIGURATION DE LA CARTE MASTER

Carte: Raspberry Pi

- Wifi
- Bluetooth
- 4 port USB
- Processeur ARM Cortex-A53 - 1.4GHz
- *Ram:1Go*
- *Supporte Os linux*

OS: Ubuntu MATE 18.04

- Environnement linux
- Interface graphique (mate)
- Plus léger que Ubuntu
- Environnement maîtrisé
- Compatible avec ROS melodic

ROS distribution: MELODIC

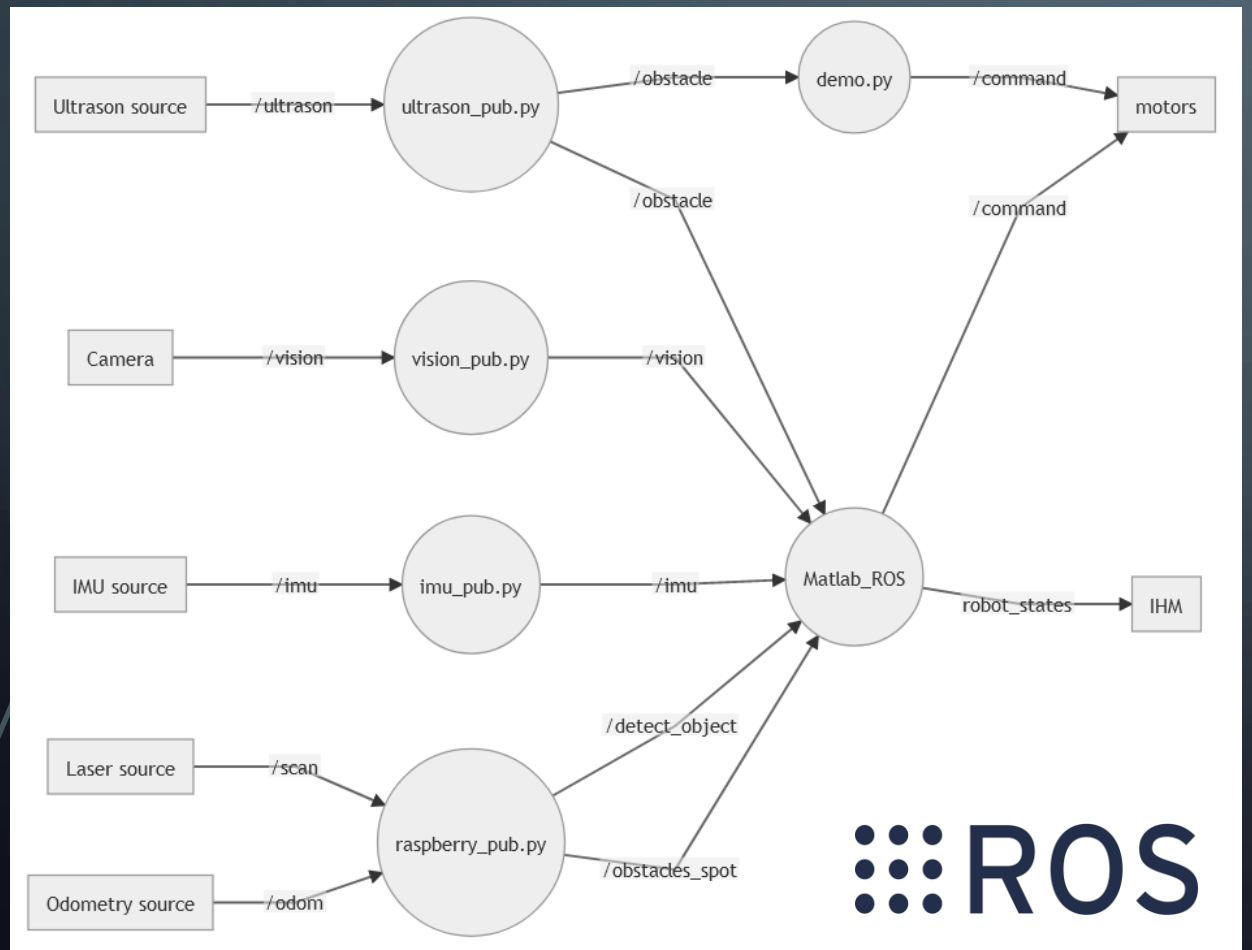
- Package rplidar → melodic

Problématiques majeurs rencontrées:

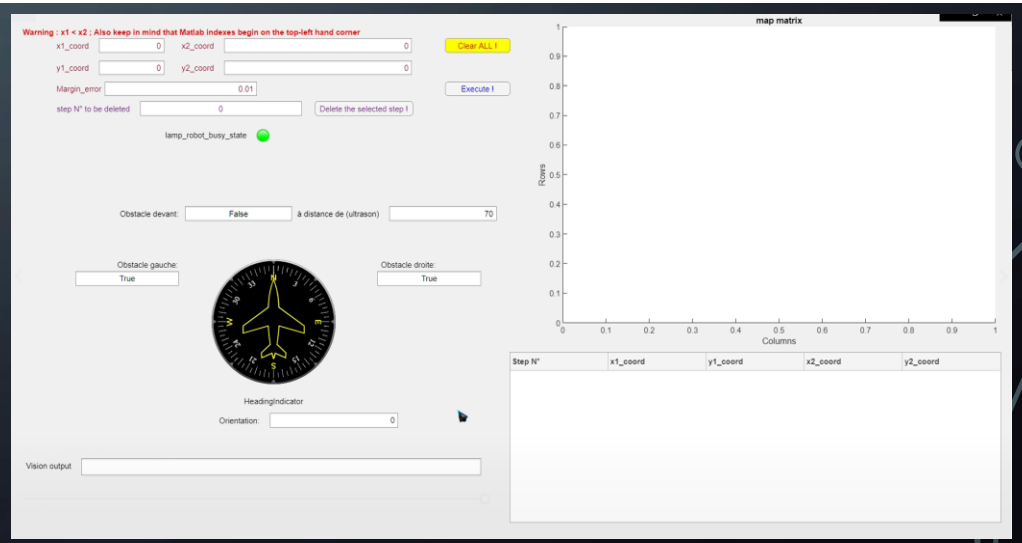
- *ROS melodic build avec python2 et non 3*
- *Le build d'un workspace demande beaucoup de RAM avec certains pkg → parfois plantage*
- *La gestion des par-feux*

CONFIGURATION DE LA CARTE MASTER & COMMUNICATION

ROS Nodes




```
Shutting down global node /matlab_global_node_28936 with NodeURI http://192.168.43.26:50477/
Initializing global node /matlab_global_node_73383 with NodeURI http://192.168.43.26:63391/
/command
/object_detected
/obstacle
/obstacle_spot
/rosout
/rosout_agg
/scan
/tf
/ultrason
Node [main_template_matlab] started>> main_and_ihm
Shutting down global node /matlab_global_node_73383 with NodeURI http://192.168.43.26:63391/
```

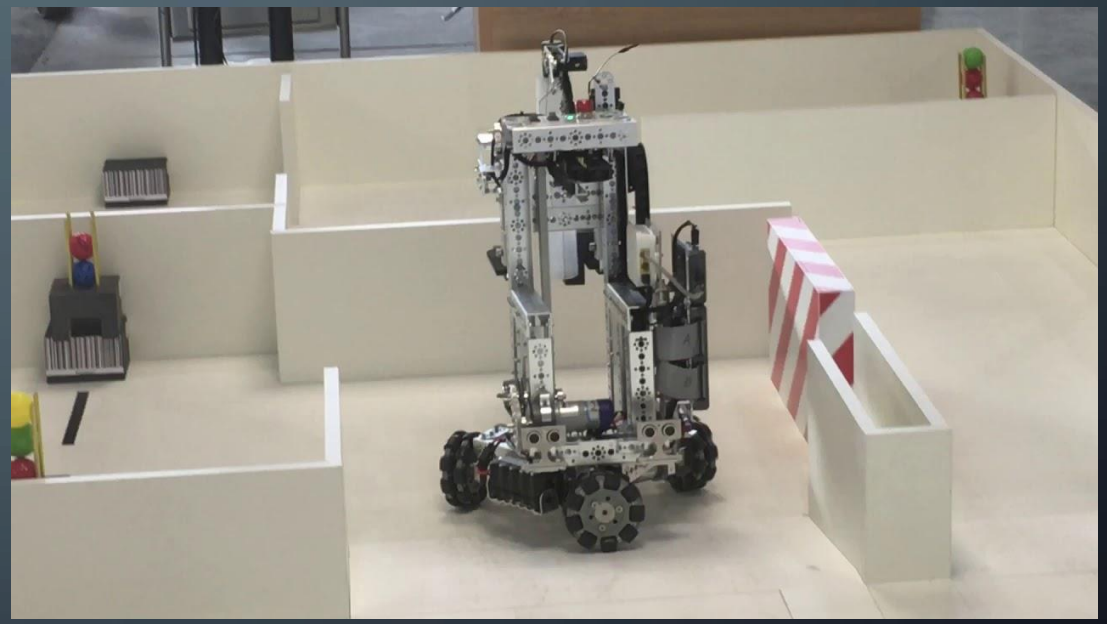


GESTION CAPTEURS

GESTION DES CAPTEURS DE NAVIGATION

- Objectifs :

- Naviguer en autonomie
- Connaître sa position, son orientation
- Pouvoir mapper son environnement
- Pouvoir éviter des obstacles



- Contraintes :

- Tous les capteurs doivent communiquer avec la RaspberryPi
- Le prix des capteurs ne doit pas être supérieur à 300 euros

CHOIX DES CAPTEURS (1)

- Capteurs à ultrason HC-SR04

Permet d'évaluer la distance entre un obstacle et la base roulante

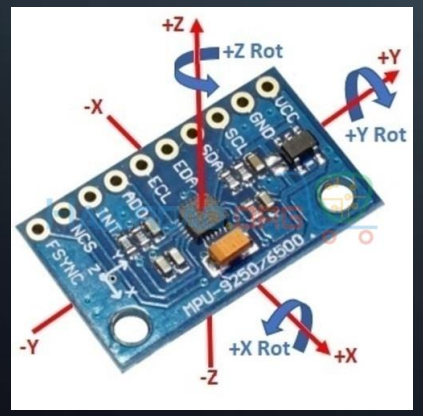
- Utilise des ultrasons pour déterminer la distance
- Echo de l'onde émise proportionnelle à la distance
- Divise le temps par 2 pour avoir la distance
- Prix très faible (Environ 3 euros)
- Mesure de haute précision et stable
- Création d'un Publisher ROS "command"

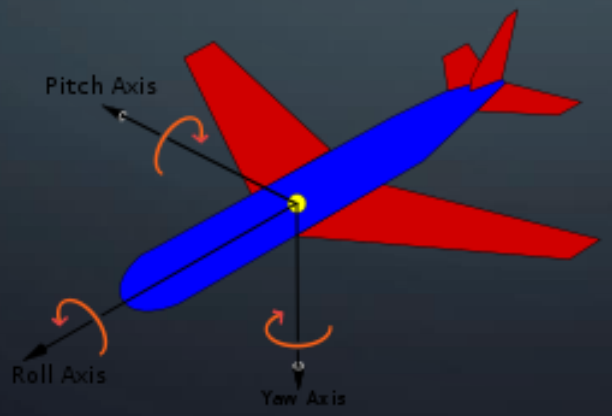
CHOIX DES CAPTEURS (2)

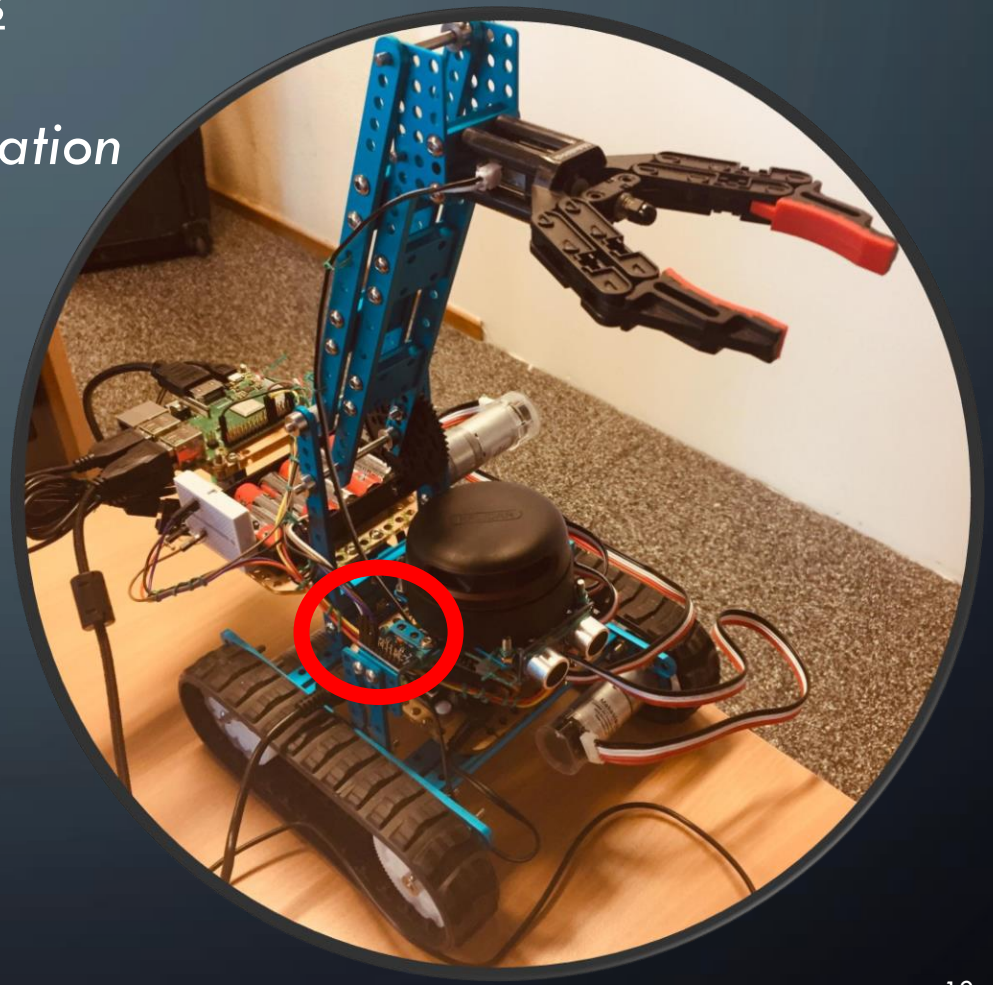
- Centrale inertielle – MPU9250 – 9 axes

Permet de connaitre la position angulaire et accélération

- Magnétomètre AK8963 → Champ magnétiaue
- Accéléromètre (MPU6050) → Accélération
- Gyroscope (MPU6050) → Position angulaire
- Permet de connaitre le PITCH/YAW/ROLL
- Connection I2C avec la RaspberryPi
- Création d'un Publisher ROS "IMU"





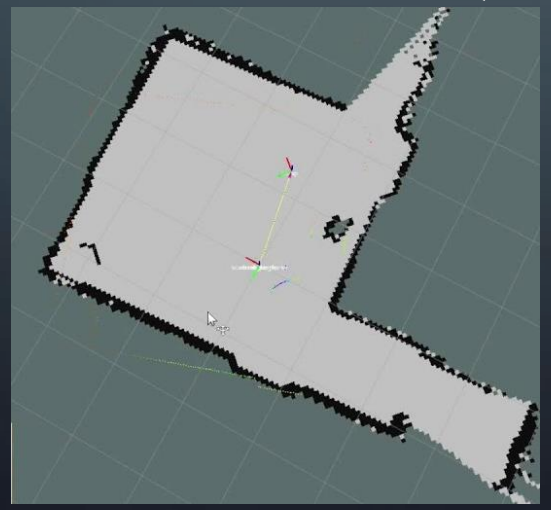


CHOIX DES CAPTEURS (3)

- RPLidar A2 de chez SLAMTEC

Permet de mapper l'environnement et connaitre la position d'obstacles autour du robot

- Rayon d'action de 16 mètres
- Beaucoup de package ROS déjà existant
- Possibilité d'utiliser SLAM pour avoir un mapping
- Prix plutôt raisonable (250 euros)
- Récéption des coordonnées du mapping grace au topic /scan/ranges (liste avec toutes les coordonés)



GESTION DE LA VISION

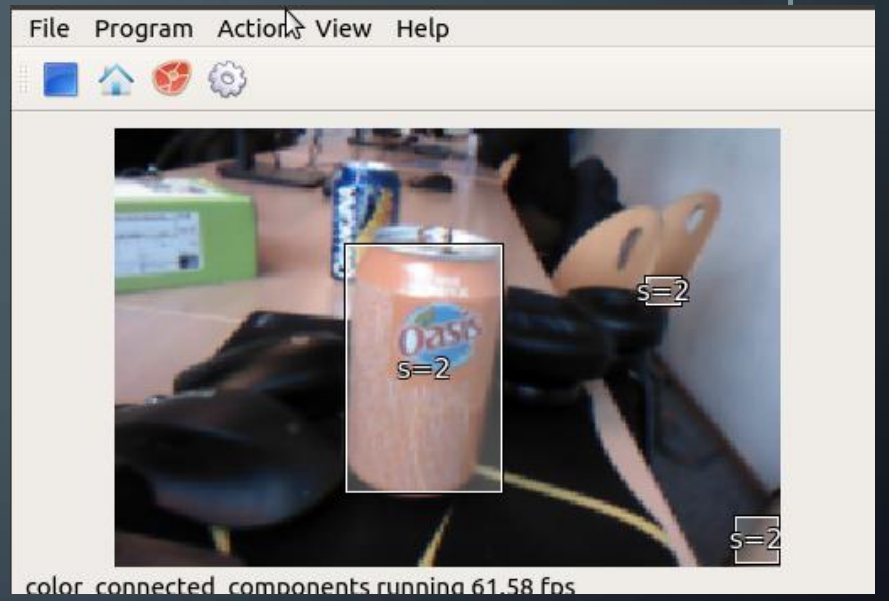
GESTION DE LA VISION DU ROBOT

- Objectifs :

- Détecter des canettes dans l'environnement
- Sélectionner une canette particulière
- Déterminer la position de la canette

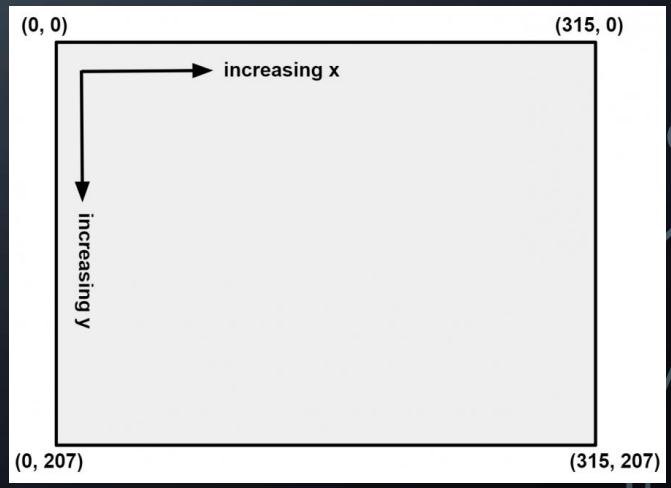
SOLUTION 1 : UTILISATION DE PIXY

La camera Pixy2 est une camera intelligente qui est capable de reconnaitre des couleurs



Utilisation :

- 7 signatures de couleurs
- Récupération de la position des bounding box
- Calcul approximatif de la position en z à partir de la taille de l'objet
- Création d'un fichier vision_publisher sous ROS
- On publie un msg contenant : signature + position en x + position en z pour chaque signature
- La position en x est relative au centre du robot

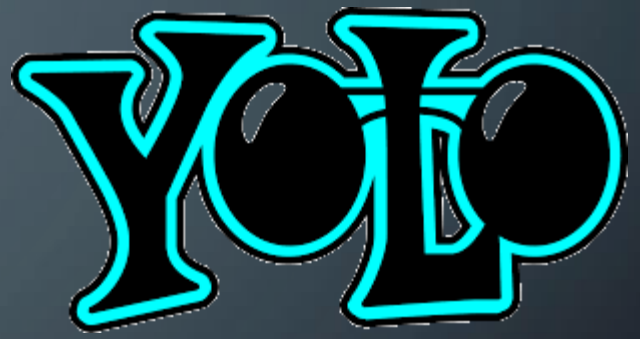


Problèmes :

- Très sensible à la luminosité
- Besoin d'objets de couleurs uniformes

SOLUTION 2 : UTILISATION DE DARKNET

On entraine un détecteur d'objet personnalisé grâce à darknet



Utilisation :

- Détecte les canettes
- Récupération de la position des bounding box
- Création d'un fichier personal_darknet_publisher sous ROS
- On publie un msg contenant : position en x pour chaque objet détecté
- La position en x est relative au centre du robot

Problèmes :

- Ne fais pas la différence entre les différentes canettes

FUSION DES SOLUTIONS

On utilise darknet et le module pixy2 pour déterminer l'objet et la couleur voulue

- On créé des listeners aux deux publishers précédemment décrit
- On publie la position x relative au robot

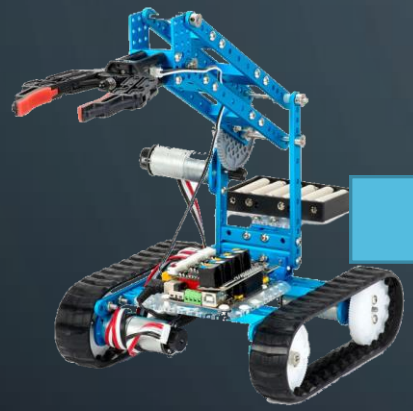
En principe on essaye de centrer le robot sur l'objet que l'on cherche à récupérer

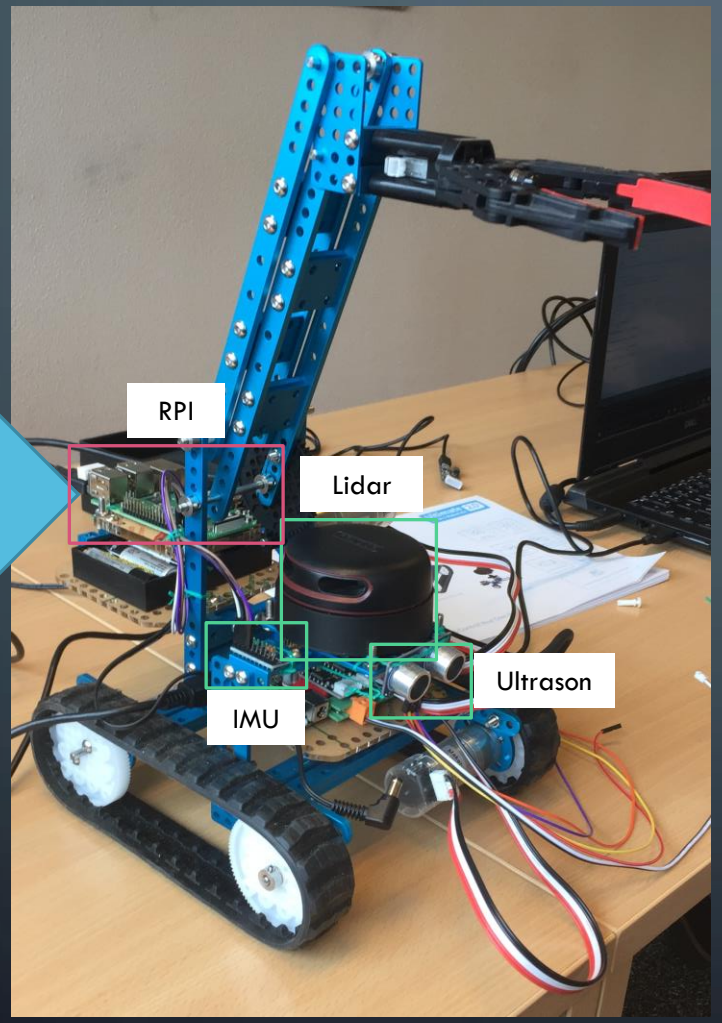
Ceci nous permet :

- D'utiliser l'ultrason pour calculer la distance restante à l'objet
- D'avoir le bras en face de l'objet pour le récupérer

IMPLEMENTATION SUR ROBOT RÉEL

IMPLÉMENTATION SUR ROBOT RÉEL





Objectifs:

- Ajouter de la Raspberry PI
- Ajouter des capteurs:
  - Lidar
  - IMU
  - Ultrason
- Programmation des mouvements sur Arduino IDE et ROS
  - Topic /command → met a jour la cmd

```
0 stop
1 forward
2 turn_left
3 turn_right
4 bras_haut_pince_ouverte
5 bras_haut_pince_fermee
6 bras_bas_pince_ouverte
7 bras_bas_pince_fermee
```

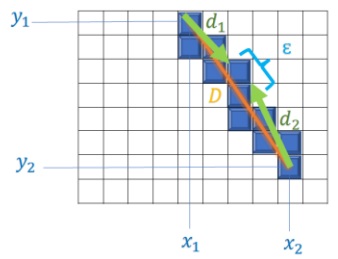
INTERFACE HOMME MACHINE ET SIMULATION DU ROBOT

INTERFACE HOMME-ROBOT : CONFIG

Acquisition

Création de carte

- On choisit quelles cases griser entre 2 points pour tracer l'obstacle :



The Jérémie's Law :

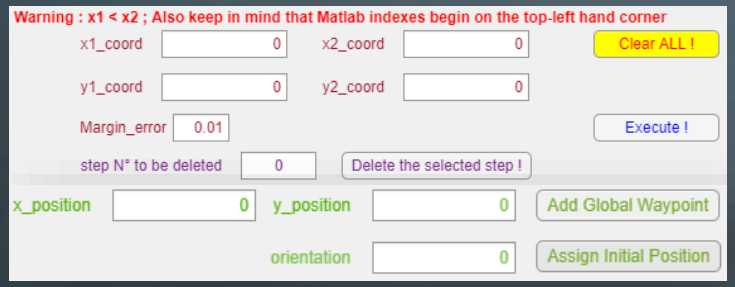
$$d_1 + d_2 < D + \varepsilon$$

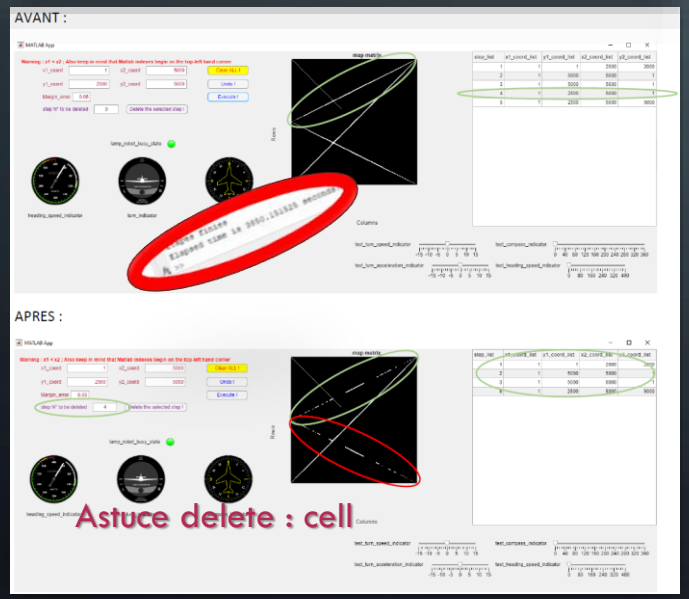
Avec ε une marge définie de manière arbitraire.

The Jérémie's "ensurance" 2nd Law :

On automatise la création de la marge en imposant chaque case grisée d'avoir un voisin grisé également. De cette façon, on s'assure d'avoir une ligne reconstituée sans trou.

La step d'augmentation de la marge est arbitraire et varie selon la puissance de calcul de la machine.



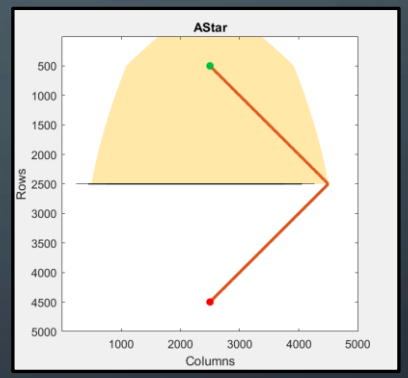


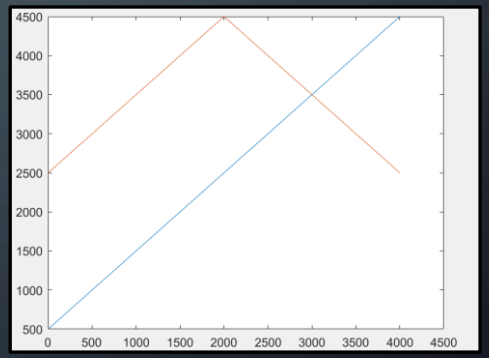
Mise en mémoire pour UX

map_matrix	5000x5000 double
margin_error_list	[]
step_count	0
step_list	[]
step_operations_list	[]
waypoints	[]
x1_coord_list	[]
x2_coord_list	[]
x_operations_list	0x0 cell
y1_coord_list	[]
y2_coord_list	[]
y_operations_list	0x0 cell

Traitement

Global planner





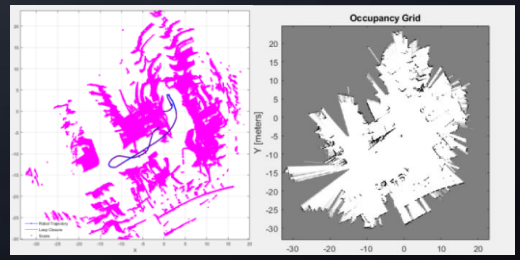
MATLAB Variable: path
15 janv. 2021

	1	2
1	500	2500
2	501	2501
3	502	2502
4	503	2503
5	504	2504
6	505	2505
7	506	2506
8	507	2507
9	508	2508
10	509	2509
11	510	2510
12	511	2511
13	512	2512
14	513	2513
15	514	2514
16	515	2515
17	516	2516
18	517	2517
19	518	2518
20	519	2519
21	520	2520
22	521	2521
23	522	2522
24	523	2523

Post-Traitement

SLAM

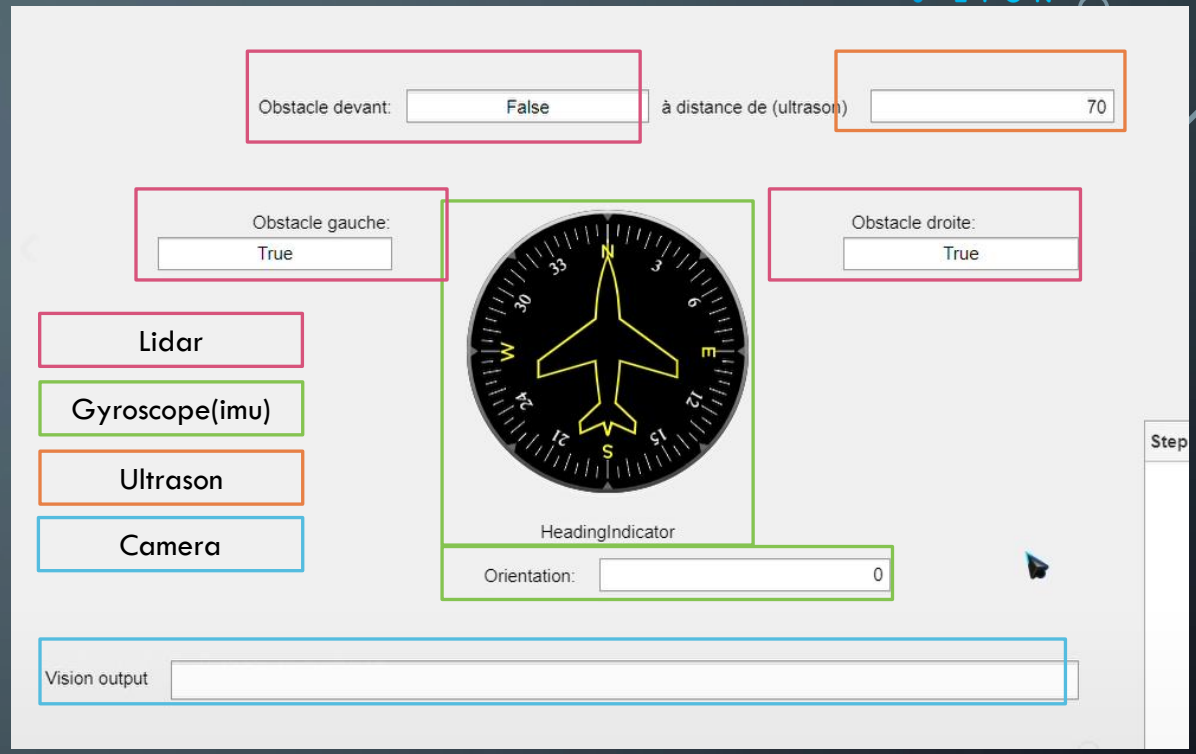
Rosbag



IHM ET CAPTEURS

Objectif:
- Afficher et transmettre l'états actuel du robot
- Transmettre les consignes

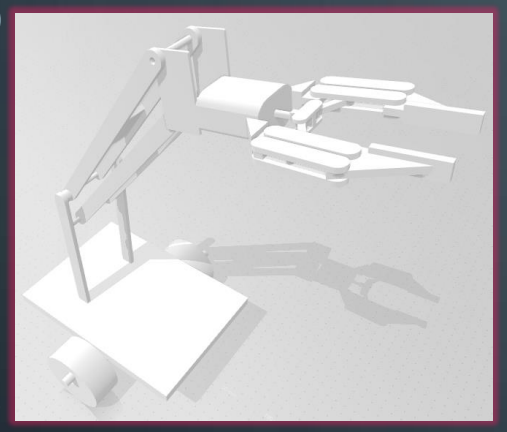
Solutions:
- ROS matlab
  - Connection à la Raspberry et ses topics
  - Le topics /command met à jour l'IHM et le robot 3D
- Création d'un objet « robot »
  - Permet de protéger des variables
  - Création de méthodes spécifique à matlab
  - Compatible avec l'IHM pour récupérer des donnée
- Les fonctions callback sur matlab mettent à jour les propriétés de « robot »



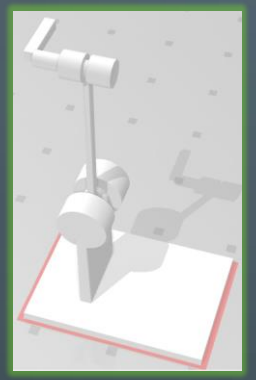
HIL : VISUALISATION TRIDIMENSIONNELLE

Acquisition

CAO



Limitation Passive J. | Simplification commande | Axage TF



URDF

```
HIL_pkg
├── CMakeLists.txt
├── launch
│   ├── HIL.rviz
│   └── HIL_sim.launch
├── meshes
│   ├── bras.STL
│   ├── chassis.STL
│   └── pince.STL
├── package.xml
├── scripts
└── urdf
    └── Assemblage_Robot.urdf
```

- ✓ Pas d'origin pour les Links
- ✓ Pas de collision
- ✓ Pas de physique
- ✓ Pas d'export SdWs .STL

MAIS

- ➢ Définition manuelle Joint
  - ❑ Axis
  - ❑ Origin

Traitement

MoveIt Setup

Define Planning Groups

Create and edit 'joint model' groups for your robot based o
subset of the robot you want to plan for.Note: when adding

Current Groups

- **HIL_group**
  - *Joints*
    - chassis_bras - Revolute
    - bras_pince - Revolute
  - *Links*
  - *Chain*
    - robot/chassis -> robot/pince
  - *Subgroups*

Define Robot Poses

Create poses for the robot. Poses are defined as s

	Pose Name	Group Name
1	bras_haut_pince_ouverte	HIL_group
2	bras_haut_pince_fermee	HIL_group
3	bras_bas_pince_ouverte	HIL_group
4	bras_bas_pince_fermee	HIL_group

MoveIt Demo : HIL

Projet Majeur
Video de démonstration

CONCLUSION

CE QUI FONCTIONNE

- IHM qui se met à jour
- Détection obstacles (arrêt de mvt) & objectifs (cannettes)
- Suite de commandes via la raspberry pi
- HIL
- Global Planner

A AMÉLIORER

- Local Planner
- Commande de la pince en fonction de la vision
- Commande depuis Matlab après le global planner
- Local planner -> evitement

RÉPARTITION DES TÂCHES

	Intégration	hardware	Vision	IHM	HIL
Briand	+++	+		+	
Guy	+	+++			
Kahan				+++	+++
Martinez			+++		