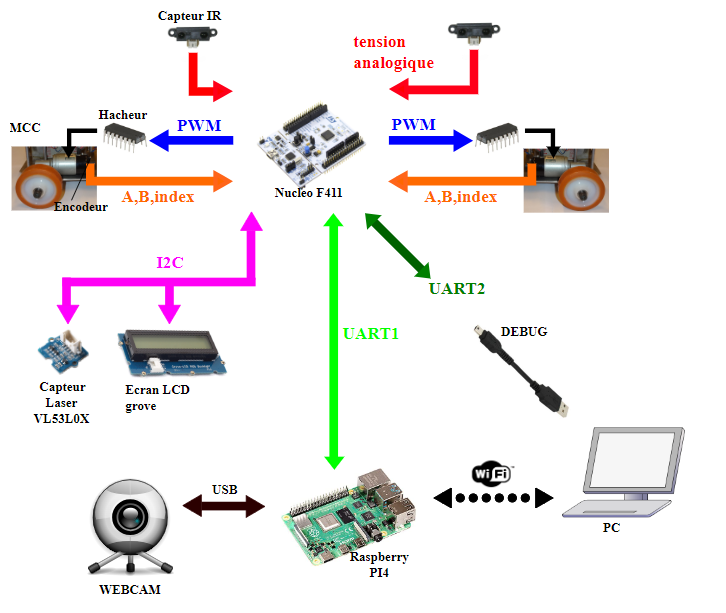
Documentation du Robot mobile ROS2



Équipe :

EZ-ZGHARI Latyfa (P1)

FLOCH Louanne

ZAHIR Ouiame (P1)

PLANTEC Jeremy

MELON Titouan

TOGNIA DJANKO DIALLO Donald

Sommaire

[**Introduction 2**](#_llxqhz2ie975)

[**Objectifs 2**](#_thhdkwwoogx7)

[**Mise en place 2**](#_yrxnyoq72e6q)

[Système d’exploitation 2](#_ajhomm3tg41s)

[Installation de l'environnement 3](#_ixrdq31o9xi3)

[Communication ROS2 4](#_nh08u05nevh8)

[**Lancement de µROS sur la RPI: 5**](#_cy5pevquhzt3)

[Adaptation de la carte sur le robot 5](#_6yx30i5ofw18)

[Branchement des capteurs 5](#_hvw3mkf8sgtc)

[Cartographie des pins de la carte STM 6](#_vd85w5f7jg50)

[Création IHM : 6](#_uvfebo9voecc)

[rqt : 6](#_s32acs1s8ylj)

[**Carte électronique 6**](#_m46fk1nn9zv8)

[**Abonnement et publication pour communiquer entre RPI, IHM et STM32 9**](#_y14bucyimxam)

[Routeur Wi-Fi 10](#_ij5vvcn70b66)

[Gestion du réseau : 11](#_sr90hglle9n5)

# Introduction

L’Objectif de ce projet est de mettre à jour le robot mobile utilisé en S7 afin de créer une maquette pédagogique utilisable par la communauté enseignante et disponible en open source.

Sur le modèle du Turtlebot ([https://emanual.robotis.com/do](https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/)

[cs/en/platform/turtlebot3/overview/](https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/) ), le système sera constitué de :

* Une carte Nucleo STM32F411 permettant le contrôle des moteurs à courant continu en boucle fermée, et l’acquisition des différents capteurs.
* Une Raspberry PI 4 permettant la communication WIFI avec l’utilisateur (PC), et le traitement des images d’une caméra.

Le PC, la RPI et la STM32 utiliseront ROS2 pour formaliser l’échange des messages.

<https://github.com/TitouanMelon/ENIB_Robot_Mobile_Ros2>

# Objectifs

À l’issue du projet, le Robot aura la possibilité d’être contrôlé à distance dans l’environnement indiqué ci-dessus, et de suivre une couleur ou une forme détectée par la caméra.

Les pièces mécaniques pour modifier l’architecture initiale seront à concevoir, et les différents environnements de développement (QtCreator-ROS, Développement croisé avec le RPI) seront à valider.

Une documentation « utilisateur » sera à fournir à destination d’enseignants souhaitant utiliser ce robot (plans, manuel d’utilisation).

# Mise en place

## Système d’exploitation



* Ubuntu 22.04.3 (LTS) x64bit

<https://ubuntu.com/download/desktop> (pour le PC ) )

<https://www.raspberrypi.com/software/> (pour la carte Raspberrypi ) )

* ROS 2 Humble

<https://docs.ros.org/en/humble/Installation.html>

<https://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debians.html>

## Installation de l'environnement

Tout d’abord, il faut installer Ubuntu 22.04 (voir les liens ci-dessus), puis on installe ROS2 Humble sur Ubuntu pour la carte Raspberry Pi 4 et sur le PC.



* Pour la carte Raspberry PI4 : 

: rpiros

: a1z2e3

* Pour le PC

: rospc

: a1z2e3

Pour programmer la carte STM32, il faut installer le logiciel Cube IDE de ST (<https://www.st.com/en/development-tools/stm32cubeide.html> ). Il faut utiliser µROS.

Pour installer ROS2

sudo apt install software-properties-common

sudo add-apt-repository universe

sudo apt update && sudo apt install curl -y

sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o /usr/share/keyrings/ros-archive-keyring.gpg

echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-keyring.gpg] http://packages.ros.org/ros2/ubuntu $(. /etc/os-release && echo $UBUNTU\_CODENAME) main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null

sudo apt update

sudo apt upgrade

sudo apt install ros-humble-desktop

sudo apt install ros-humble-ros-base

sudo apt install ros-dev-tools

# Replace ".bash" with your shell if you're not using bash

# Possible values are: setup.bash, setup.sh, setup.zsh

echo "source /opt/ros/humble/setup.bash" >> ~/.bashrc

Pour installer µROS:

Once you have a ROS 2 installation in the computer, follow these steps to install the micro-ROS build system:

*# Source the ROS 2 installation*

source /opt/ros/$ROS\_DISTRO/setup.bash

*# Create a workspace and download the micro-ROS tools*

mkdir microros\_ws

cd microros\_ws

git clone -b $ROS\_DISTRO https://github.com/micro-ROS/micro\_ros\_setup.git src/micro\_ros\_setup

*# Update dependencies using rosdep*

sudo apt update **&&** rosdep update

rosdep install --from-paths src --ignore-src -y

*# Install pip*

sudo apt-get install python3-pip

*# Build micro-ROS tools and source them*

colcon build

source install/local\_setup.bash

# Communication ROS2

Vérifier que toutes vos cartes comprennent le même ID ROS

sur linux :

* echo $ROS\_DOMAIN\_ID # affichage
* export ROS\_DOMAIN\_ID=<idNumber> # modification

Sur STM32 :

* ROS\_DOMAIN\_ID=<idNumber> #on line 43

Voici les commandes pour communiquer entre les cartes

| Info commande | Commande |
| --- | --- |
| Afficher les topics disponible | ros2 topic list |
| Afficher les infos d’un node ROS | ros2 node info /nom/du/node |
| S’abonner à un topic pour afficher les infos reçus | ros2 topic echo /nom/du/topic |
| Envoyer des infos sur un topic une seule fois | ros2 topic pub --once /nom/du/topic <data> |
| Envoyer des infos sur un topic une seule fois | ros2 topic pub -r <sendParSecond> /nom/du/topic <data> |

sendParSecond doit être un entier correspondant au nombre de messages envoyés par seconde.

<data> doit être une chaîne de caractère correspondant à un dictionnaire de la forme décrit sur le site suivant : [docs msg ros](https://docs.ros.org/en/melodic/api/std_msgs/html/index-msg.html)

# Lancement de µROS sur la RPI:

Pour lancer la communication entre la STM32 et la RPI on doit lancer micro ros sur la RPI afin de notifier qu’il doit écouter l’USB qui vient du robot.

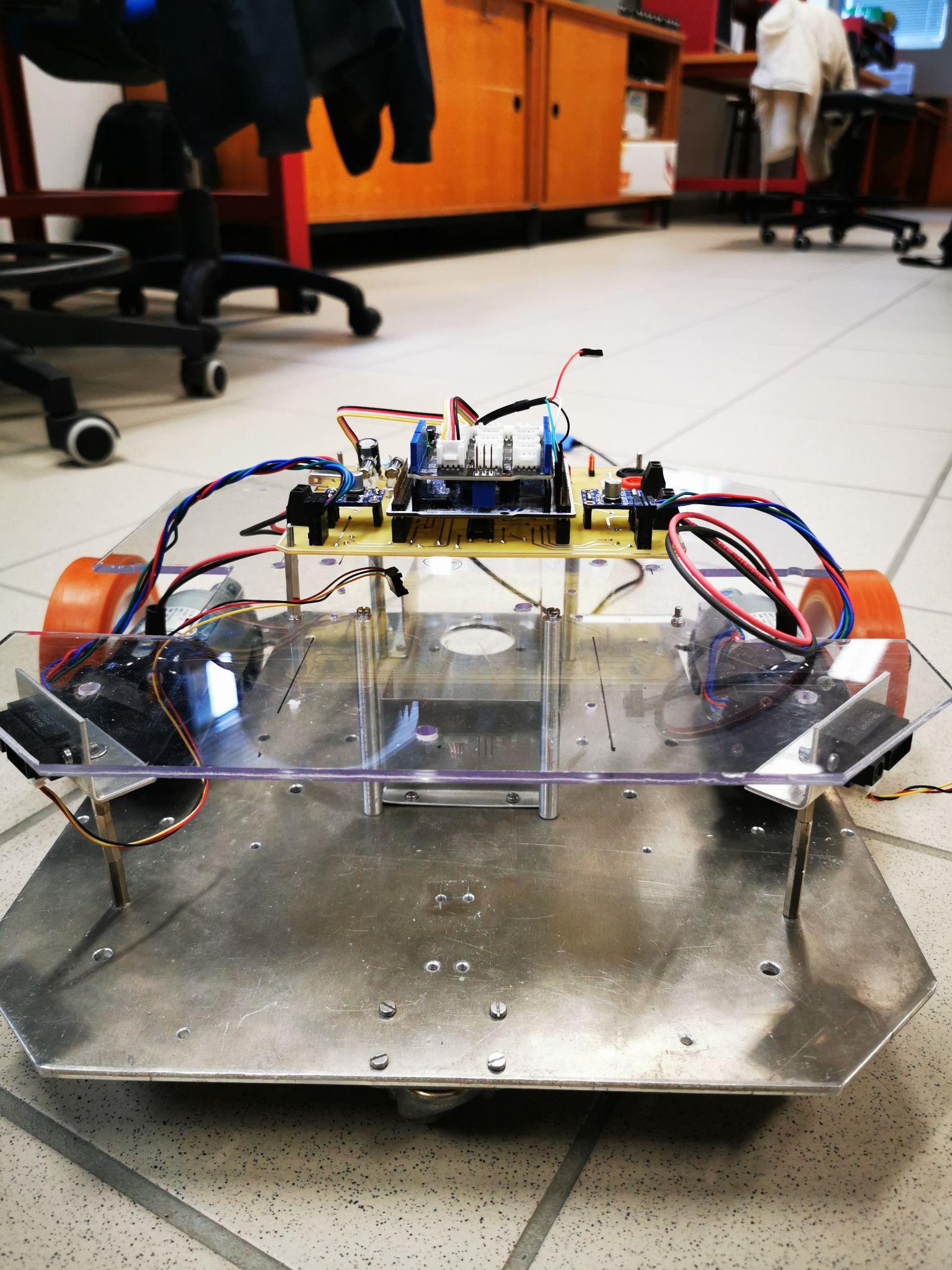
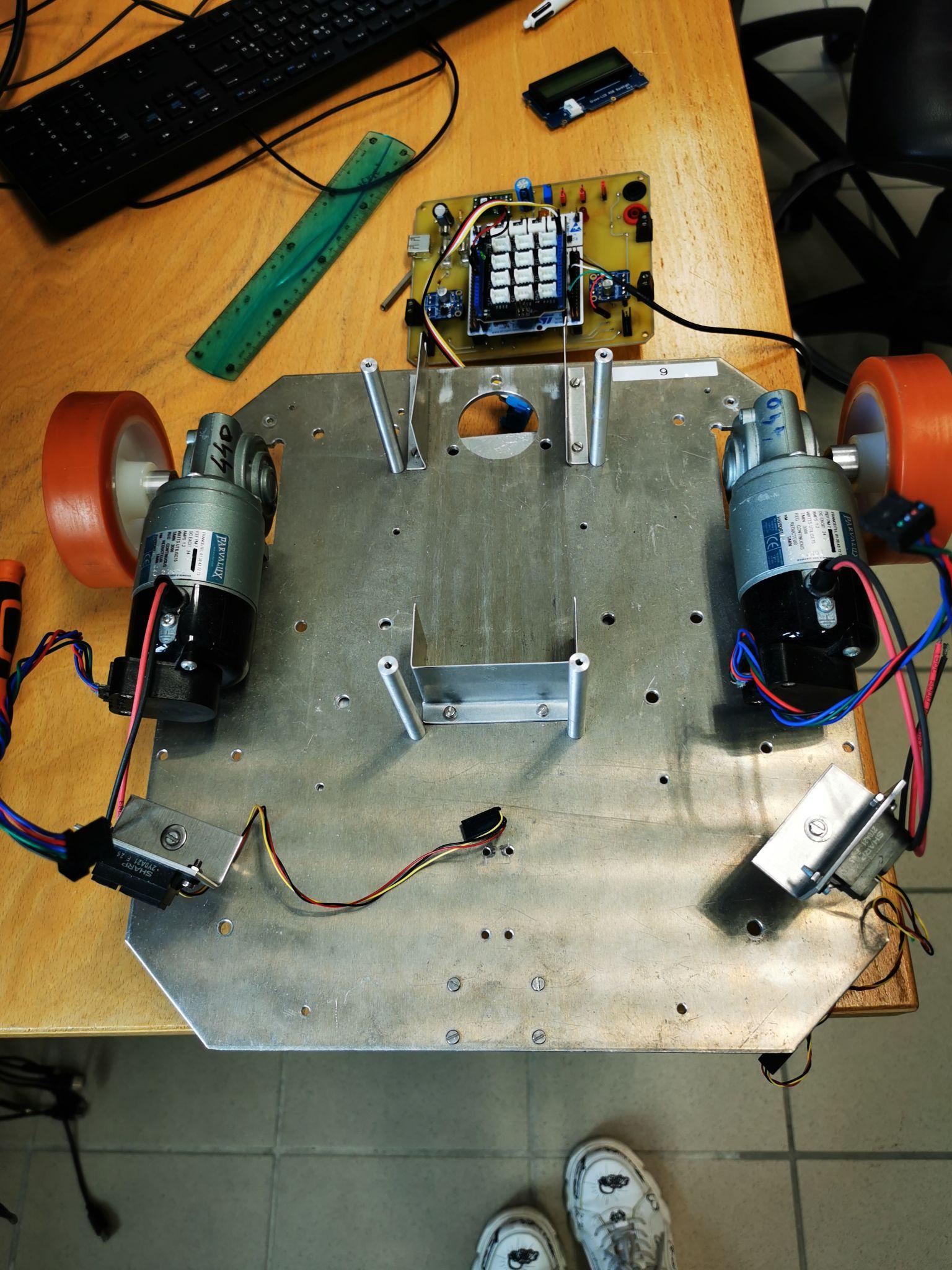
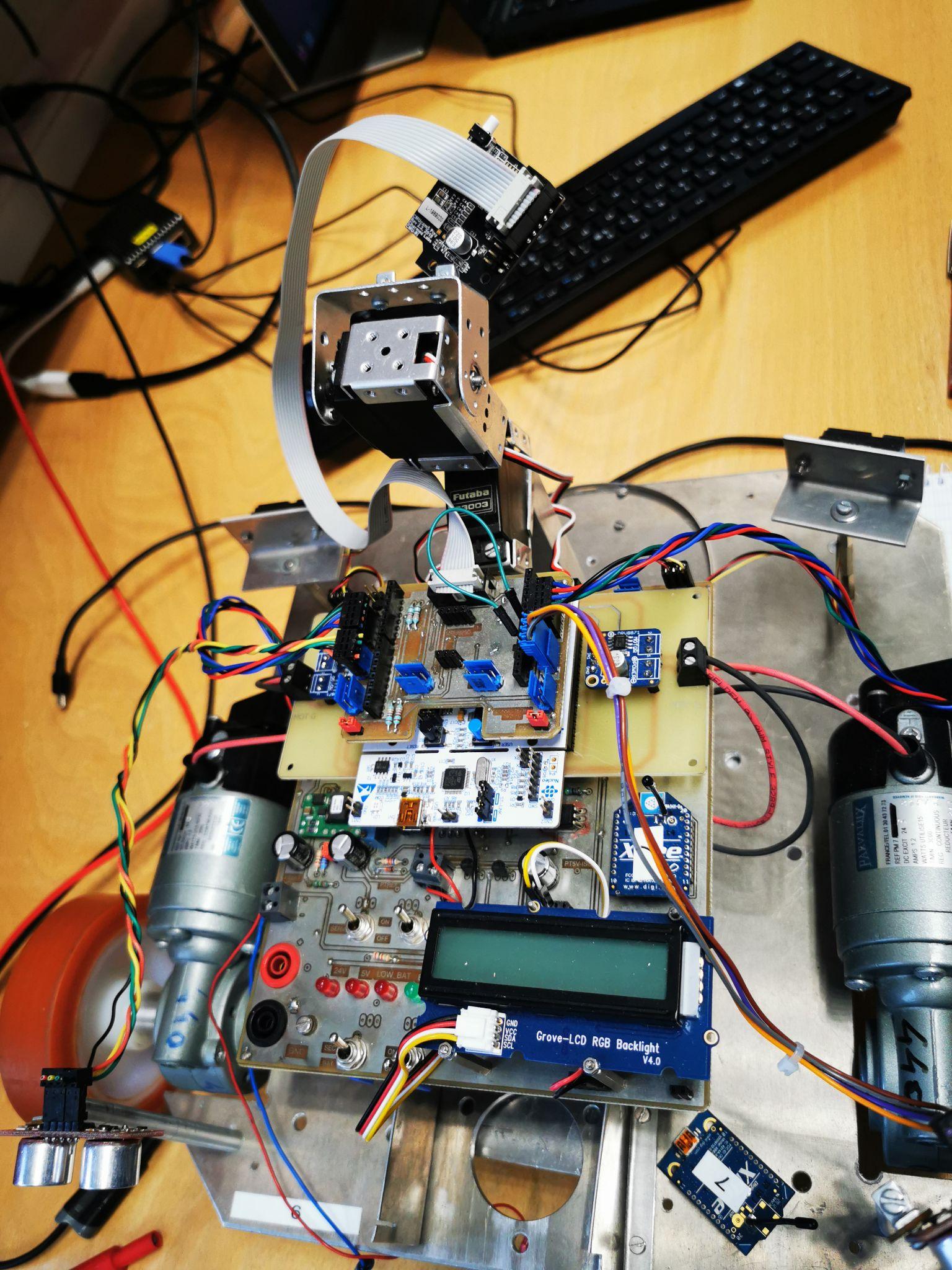
On se rend dans le répertoire ou on à installer microRos : cd ~/microros\_ws

Ensuite on active l’environnement virtuel : source install/setup.bash

Ensuite on lance l’agent microRos : ros2 run micro\_ros\_agent micro\_ros\_agent serial --dev /dev/ttyUSB0 -b 115200

Si une erreur ‘access denied’ apparaît, exécuter la commande suivante pour accorder les droits à tout le monde et relancer la commande précédente : chmod 777 /dev/ttyUSB0

# Adaptation de la carte sur le robot



Afin de mettre la nouvelle carte sur le robot, une modification du robot est nécessaire. Tout d’abord, la carte étant plus grande que la précédente, nous avons choisi de fixer la carte sur une plaque de plexiglas, elle-même fixée sur le socle du robot. La plaque de plexiglas nous permet d’isoler la carte de la partie avec les batteries afin d'éviter que les pins de la carte entre en contact avec les connecteurs de la batterie. Un support pour le capteur arrière et le LCD sera fabriqué à partir d’imprimante 3D.

Mettre catia plexi glass

mettre lcd et capteur arrière

# Branchement des capteurs

Les capteurs sont branchés sur un shield qui vient se mettre sur la carte STM32.

# Cartographie des pins de la carte STM

Moteur :

PA4 → PWM 3 moteur 1

PC7 → PWM 3 moteur 2

PB3 → enable moteur

Capteurs :

PA4 → capteur avant ADC channel 8

PB0 → capteur avant ADC channel 4

I2C → capteur arrière

Encodeur moteur :

Timer 1 : PA8, PA9, PB10

Timer 2 : PA6, PA9, PC0

# Création IHM :

# **ROBOT GUI**

| Composant GUI | Quantité | Fonction |
| --- | --- | --- |
| · QGraphicsView  · QGraphicsScene | 1 | Affichage des trames d’image en provenance de la caméra du Robot |
| · QRadioButton | 3 | Gestion des modes de fonctionnement du robot c’est-à-dire Manual, Auto et Track |
| · QPusButton | 4 | Commande pour la direction du robot |
| · QCheckBox | 1 | Commande de Démarrage/Arrêt du GUI |
| · QSlider | 6 | Détection de couleur par HSV |
| · QLCDNumber | 1 | Affichage des valeurs HSV |
| · QSlider | 1 | Gestion Vitesse Robot |
| · QLCDNumber | 1 | Affichage de la Vitesse |

# **Installations**

Ø Prérequis :

Se rassurer d’avoir ros2 humble installé (cf. Installation de l’environnement)

Se rassurer d’avoir Qt5 installé (https://doc.qt.io/qt-5/gettingstarted.html)

Ø Téléchargement :

Lien de téléchargement du workspace : <https://drive.google.com/file/d/13sd1yhIsnY0r00ch8bwKGmijEXPClhoe/view?usp=drive_link>

Ø Installations :

o Ouvrir le terminal linux et se rendre dans le sous répertoire « robot\_mobile\_pkg\_cpp » du répertoire « build » du workspace téléchargé :

cd /chemin/vers/le/workspace/robot\_mobile\_ihm/build/ robot\_mobile\_pkg\_cpp/

o Effacer tout ce qui se trouve dans ce répertoire

rm –rv \*

o Se rendre au répertoire parent “robot\_mobile\_ihm” et compilé le workspace

cd ..

cd ..

colcon build –packages-select robot\_mobile\_pkg\_cpp

o « Source » le workspace afin de pouvoir l’exécuter via la commande « ros2 run … »

source /chemin/vers/le/workspace/robot\_mobile\_ihm/install/setup.bash

Ou mieux encore l’ajouté dans le fichier système «.bashrc » :

Ouvrir le fichier .bashrc : gedit ~/.bashrc

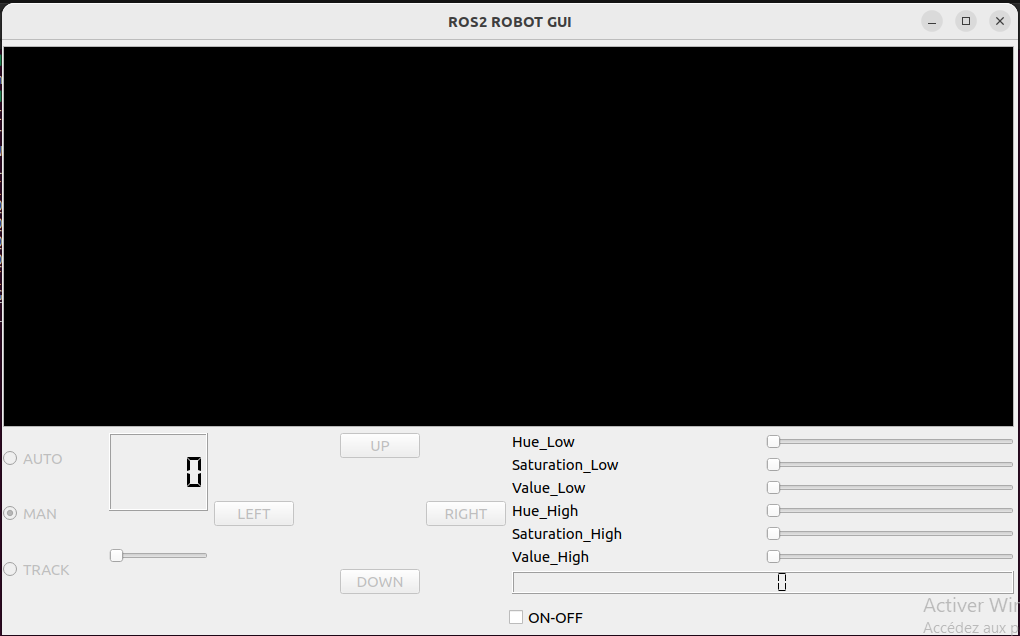
Ajouter ladite commande à la fin du fichier « .bashrc » et enregistrer :

source /chemin/vers/le/workspace/robot\_mobile\_ihm/install/setup.bash

N.B : Après avoir « source » le workspace, il faut redémarrer le terminal avant de passer à la phase d’exécution.

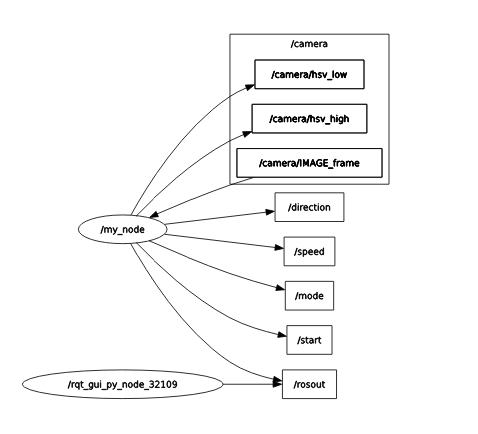
o Exécution

ros2 run robot\_mobile\_pkg\_cpp my\_node\_ihm



# **Intégration ROS2**

Le GUI communique avec le robot grâce aux fonctionnalités de communication (DDS) de ROS2 intégré à l’interface via la librairie rclcpp (rcl : Ros Client Library, cpp : C++). Nous pouvons voir les différents topics permettant la communication entre l’IHM et le robot grâce à l’outil rqt de ros2. Ouvrez le terminal après avoir démarrer l’interface et entrez « rqt ».



Comme nous pouvons le constater, notre GUI représenter par le nœud /my\_node possède 07 publishers et 01 subcriber c’est-à-dire :

· Un publisher vers la sortie standard /rosout

· Un publisher pour publier l'état (On/Off) de l’IHM sur le topic /start

· Un publisher pour publier le mode sur le topic /mode

· Un publisher pour publier la vitesse sur le topic /speed

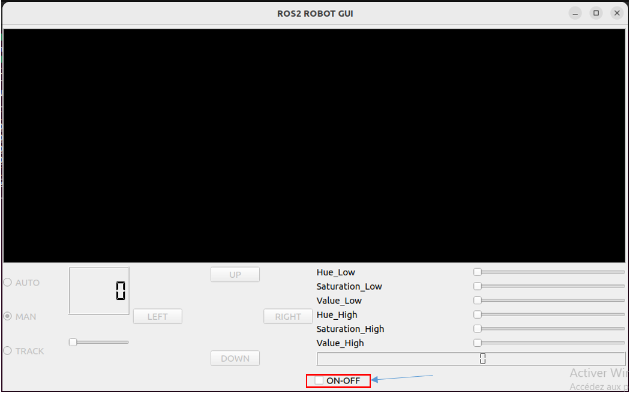
· Un publisher pour publier la commande de direction sur le topic /direction

· Deux publishers pour publier les valeur HSV low et HSV high relative à la détection des couleurs en mode Tracking sur respectivement les topics /camera/hsv\_low et /camera/hsv\_high

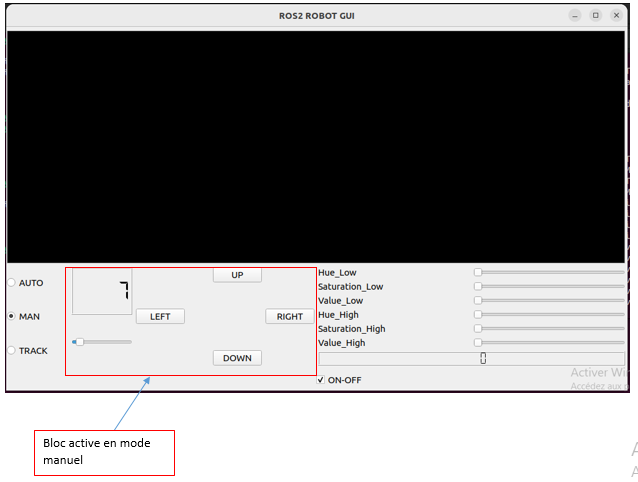
· Et enfin un subscriber sur le topic /camera/IMAGE\_frame pour récupérer les trames d’images en provenance de la camera du robot.

# **Exemples d'utilisation**

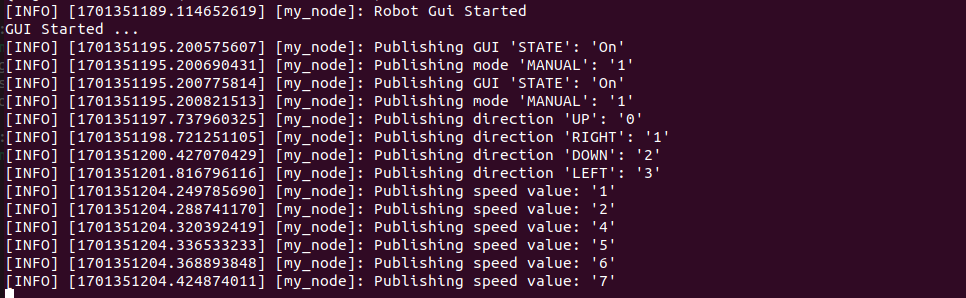
A l’exécution de l’ihm, tous les composants graphiques sont verrouillés à l’exception du bouton de démarrage.



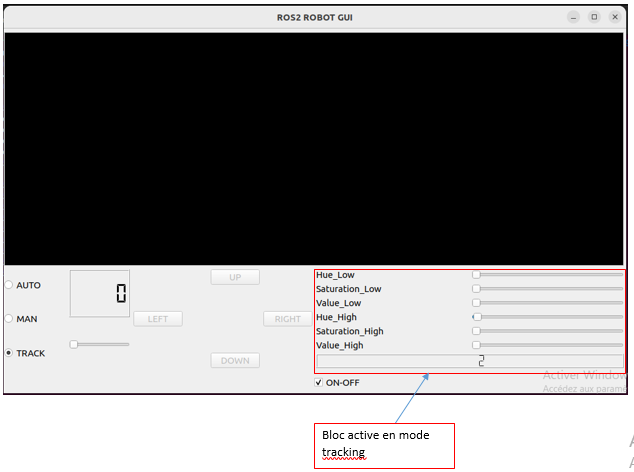
Cliquez sur « ON-OFF » pour démarrer l’IHM. Au démarrage, le mode par défaut est Manuel donc seuls les composants relatifs à la gestion du mouvement (direction et vitesse) du robot sont activés.

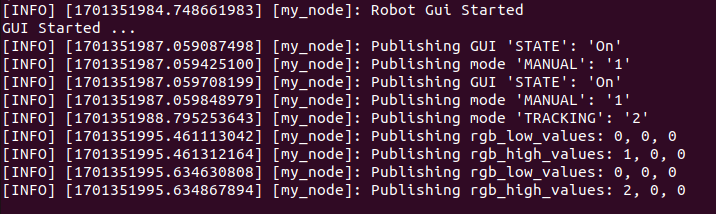


Le terminal lors de la manipulation des composants actifs nous renseigne sur le type de commande et la donnée émise ou reçue.

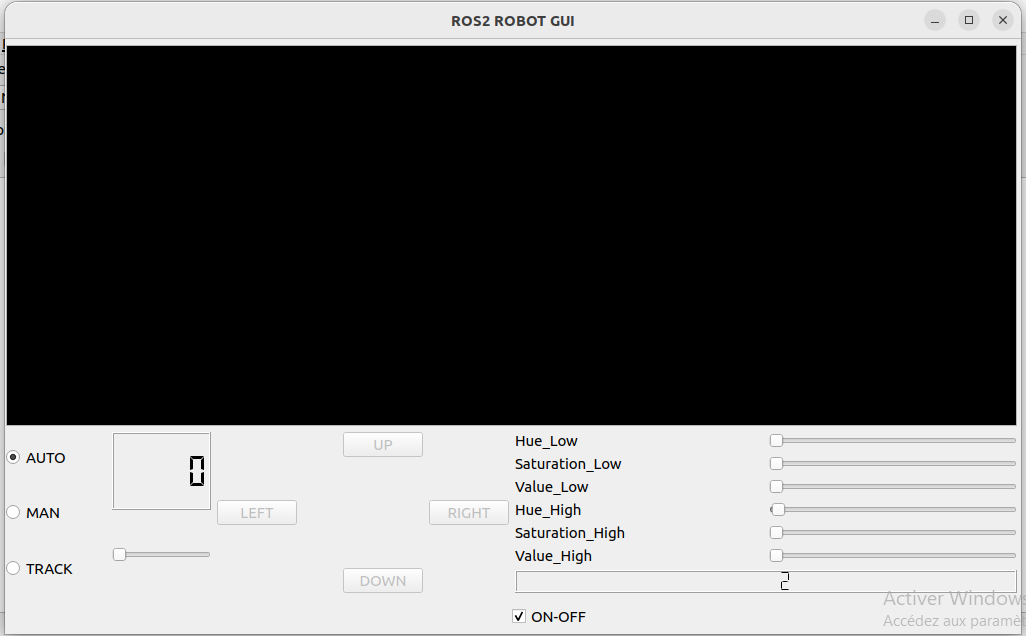


En mode tracking, seuls les composants relatifs à la gestion des couleurs sont activés.





En mode Auto, aucune commande ne peut être émise vu que le robot est en autonomie



La zone en noir en cas de connexion au robot affiche les trames d’images en provenance de la camera du robot

# **Résolution de problèmes**

En cas de problème de communication avec le robot :

Ø Se rassurer que les deux entités (ihm, robot) sont connectés au même réseau

Ø Redémarrer l’IHM

Quelques commandes utiles:

ros2 topic list

ros2 topic info “topic\_name”

ros2 node list

ros2 node info “node\_name”

ros2 topic echo “topic\_name”

#### rqt :

Permet de créer une IHM en python avec qt intégrée directement avec ROS.

Mais il existe un seul et unique tutoriel.

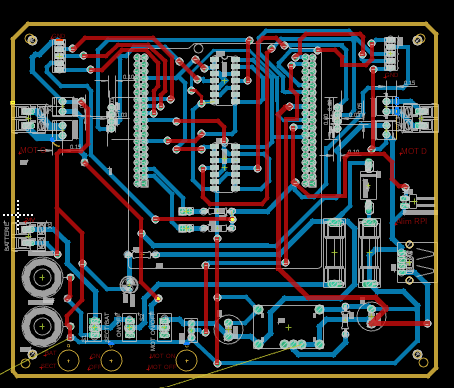
<http://wiki.ros.org/rqt/Tutorials/Create%20your%20new%20rqt%20plugin#Attributes_of_library_element_in_plugin.xml>

<https://robotics.stackexchange.com/questions/101140/rqt-plugin-tutorial-not-working>

# Carte électronique

La carte électronique a été réalisée sur Fusion 360.

Face Top:



Face Bottom:

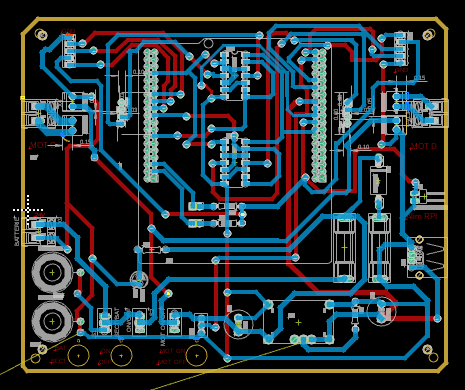
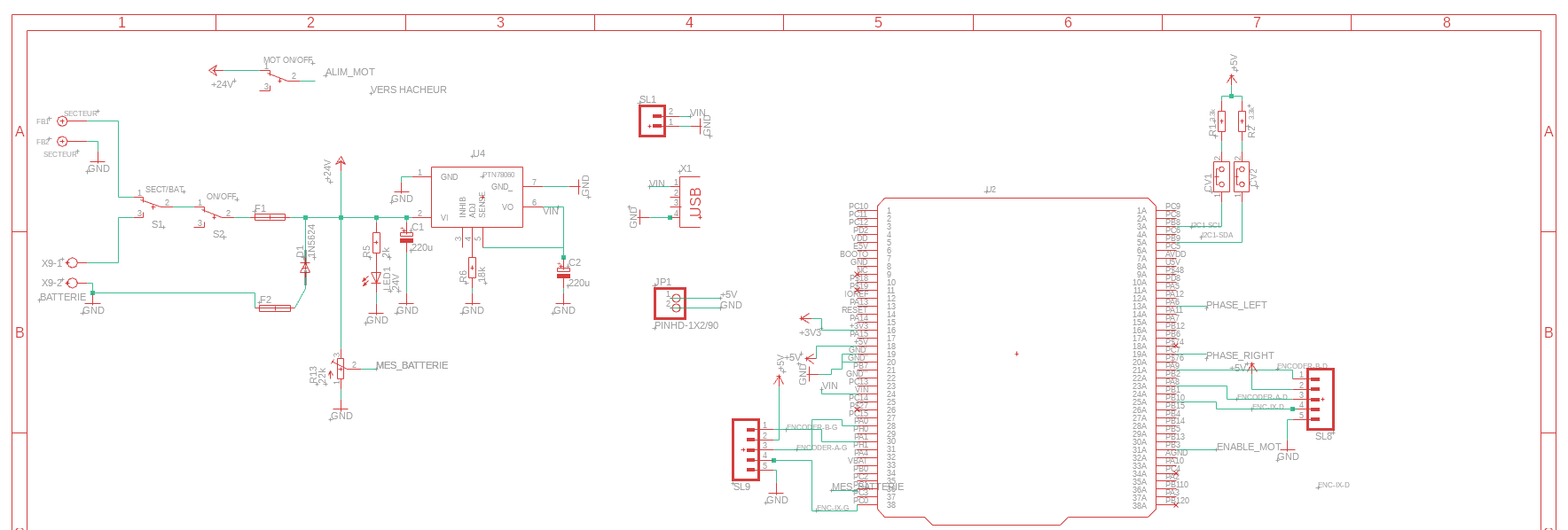
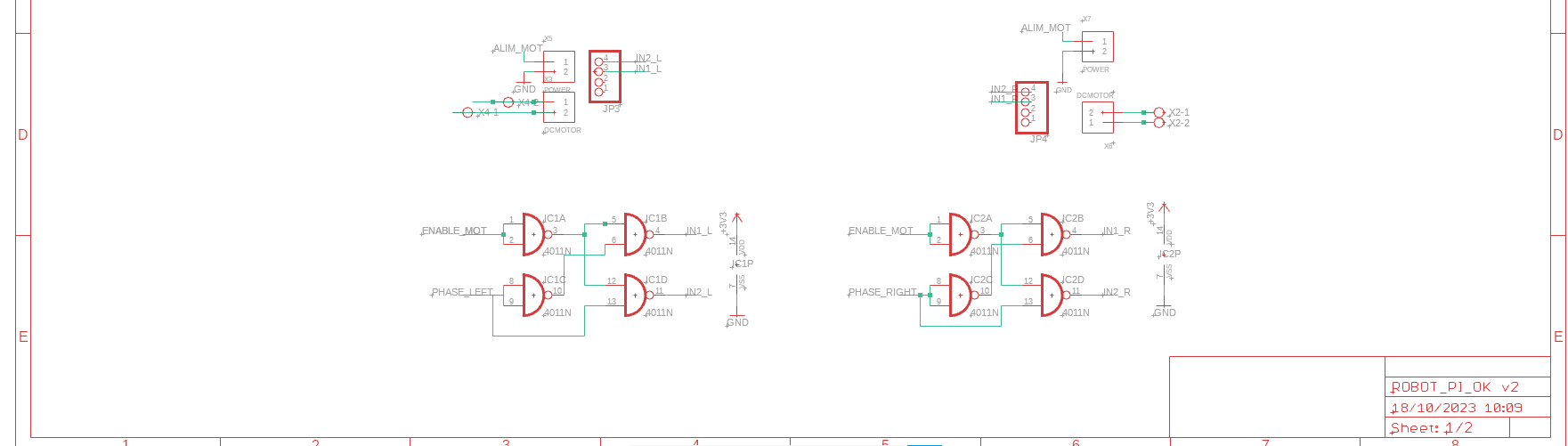


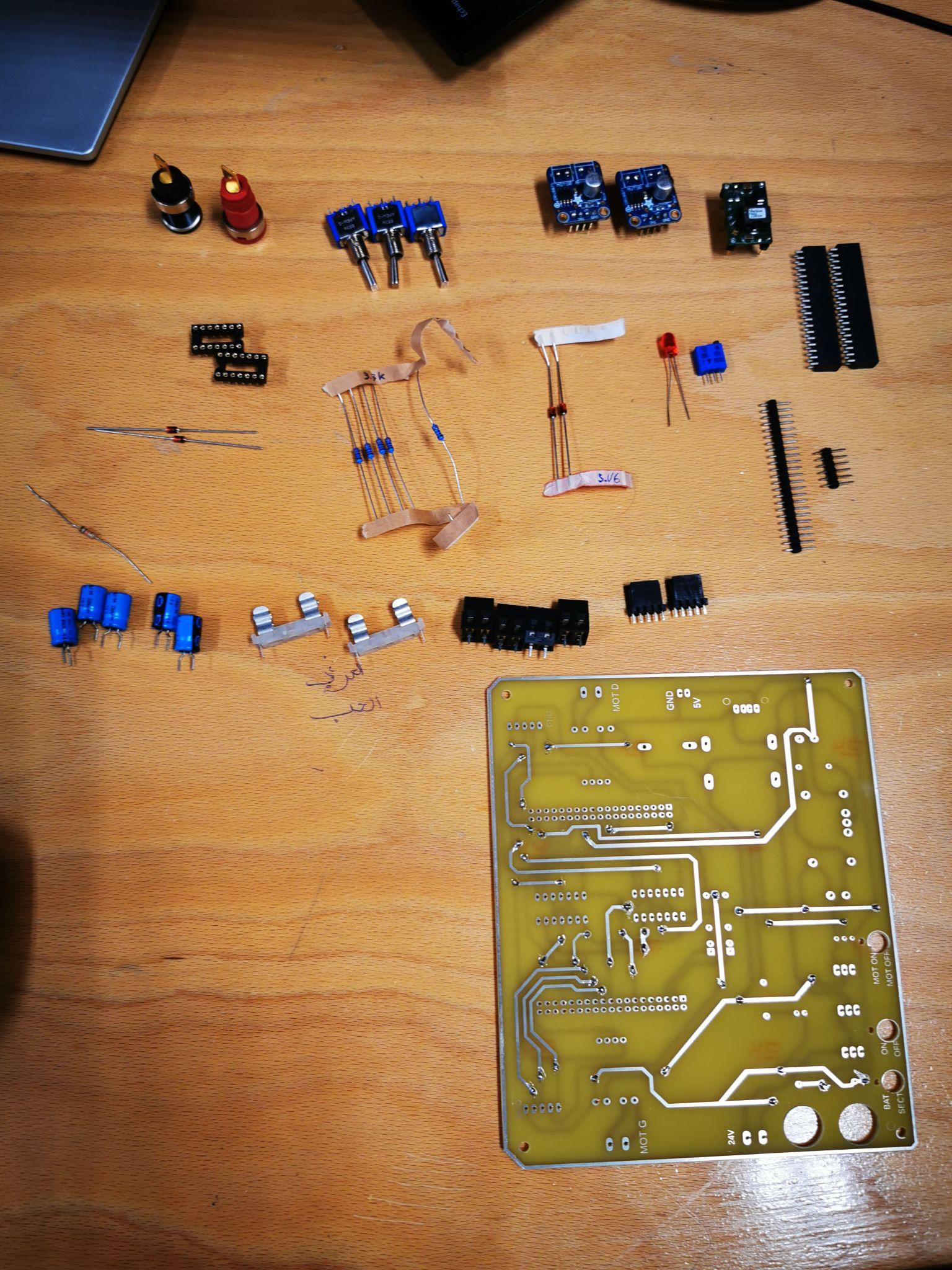
Schéma électrique :





Liste des composants pour la carte

* 2 x Fiche banane (1 noir, 1 rouge)
* 2 x adaptateur pour encodeur (Support DIL 14)
* 3 x interrupteur (5639 APEM-2)
* 2 x barrette pour STM32
* 1 x LED 5mm R
* 2 x Support APO
* 2 x Fusible 23 A
* 2 x support fusible
* 2 x condensateur 220µ
* 2 x hacheur (DRV8871)
* 1 x résistance 18k (marron gris orange or)
* 2 x Résistance 3.3k
* 1 x convertisseur 24V 5V
* 4 x connecteur borné à vise
* 1 x zener ( 5V6 ou 3V9)
* 1 x potentiomètre 22k
* 2 x CD4011UBE
* Barrettes



# Abonnement et publication pour communiquer entre RPI, IHM et STM32

Liste des abonnements (sub) et publications (pub) :

Sub → caméra

↳ x

↳ y

↳ img

Sub → télécommande

↳ dir

Pub → capteur

↳ dir

Sub → config

↳ mode

↳ speed

Pub → etat

↳ mode

↳ speed

| RPI (3 topics)  3 PUB  ↳camera/ x  ↳camera/y  ↳camera/img |
| --- |
| STM 32 (8 topics)  5 SUB  ↳camera/ x  ↳camera/y  ↳telecommande/dir  ↳config/mode  ↳config/speed  3 PUB  ↳capteur/ dir  ↳etat/mode  ↳etat/speed |
| IHM (9 topics)  6 SUB  ↳camera/ x  ↳camera/y  ↳camera/img  ↳capteur/dir  ↳etat/mode  ↳etat/speed  3 PUB  ↳telecommande/dir  ↳config/mode  ↳config/speed |

## Routeur Wi-Fi

Nom du réseau wifi : e021

Mot de passe du réseau wifi : ppeenib2023

Nom d’utilisateur pour l’interface de gestion : kerhoas

Mot de passe de l’interface de gestion : stangouach29

### Gestion du réseau :

Chaque rpi génère son hot spot wifi :

Avantage :

* Simple

Problème :

* Ajouter une carte wifi au réseau de la salle
* Config réseaux des PC de la salle

Routeur wifi central :

Problème :

* Demande de laisser les élèves connecter leur PCs sur le réseau ENIB

Tutoriel création node python :

<https://github.com/ros2/examples/tree/rolling/rclpy>

<https://docs.ros.org/en/humble/Tutorials/Beginner-Client-Libraries/Writing-A-Simple-Py-Publisher-And-Subscriber.html>