# BUILD:

## cd /workspace/action\_ws

colcon build

source install/setup.bash

## cd /workspace/dev\_ws

colcon build --packages-select basic\_mobile\_robot

colcon build --packages-select py\_srvcli

source install/setup.bash

export GAZEBO\_MODEL\_PATH=$GAZEBO\_MODEL\_PATH:/workspace/dev\_ws/src/basic\_mobile\_robot/models/

# ODPALANIE:

## SERVER:

ros2 run py\_srvcli jazda

## CLIENT:

cd /workspace/dev\_ws

python client\_bot.py

## ROS:

ros2 launch basic\_mobile\_robot basic\_mobile\_bot\_v5.launch.py  
 Jak chcemy odpalać z **RVIZ** to na którychś labach była komenda co podawać ja nie pamiętam tego dopisku

## EDYCJA MAPY:

gazebo /workspace/dev\_ws/src/basic\_mobile\_robot/worlds/basic\_mobile\_bot\_world/projectWorldEnd.world

## ROBOCIK:

dev\_ws\src\basic\_mobile\_robot\models\basic\_mobile\_bot\_description\model.sdf

### TEKSTURY ROBOTA:

dev\_ws\src\basic\_mobile\_robot\models\basic\_mobile\_bot\_description\meshes.sdf

TESKTURY PRZYPISUJEMY DO ODPOWIEDNICH KOMPONENTÓW, NAJLEPIEJ EDYTOWAĆ W **BLENDERZE** DANE TEKSTURY, wrzucać nowe obiekty i podmieniać je w miejsce XYZ starych obiektów, I potem PODMIENIAĆ W FOLDERZE **MESHES**dev\_ws\src\basic\_mobile\_robot\models\basic\_mobile\_bot\_description\meshes

## ETAPY PRACY:

## **Zrobienie własnej mapy w gazebo**, można odpalić czyste gazebo, potem zrobić mapę, **DRZWI powinny mieć szerokość min. 2.5m** (dwuklik na obiekt w gazebo podczas tworzenia mapy i zmiana parametrów)

## Podmienienie mapy:

* 1. gazebo /workspace/dev\_ws/src/basic\_mobile\_robot/worlds/basic\_mobile\_bot\_world/projectWorldEnd.world -> nowa mapa, może być ta sama nazwa, starą należy wywalić (zmienić nazwę)

LUB

basic\_mobile\_bot\_v5.launch.py  
**linia 28:**  world\_file\_name = 'basic\_mobile\_bot\_world/projectWorldEnd.world'

## Zmiana modelu robota (meshy w blenderze) i wrzucenie do folderu, nazwy mogą zostać, możliwe będzie dostosowania odległości między poszczególnymi częściami (estetyka)

## Podmienienie modelu świata i zapisanie na nowo

## Skanowanie mapy z robotem z rviz, można sobie wtedy używać tego **odometry** co było na zajęciach i suwakiem nakurwiać (poprawiałem skrypty więc można zakurwiać ile wlezie i dalej skanuje)

## CAŁY SERWER JEST W PLIKU **JAZDA.PY**

dev\_ws\src\py\_srvcli\py\_srvcli\jazda.py

klient jest sklepany na kolanie i ten sam folder client\_bot.py -> wykonuje komendy w bashu zamiast wysylać jakieś zapytania

## ROBOT RADAR:

schowany aby oszczędzić fpsów trochę

RADAR MA 90 JEDNOSTEK NA 180 stopni i zaczyna się od 45 stopnia (jazda.py linia 204)

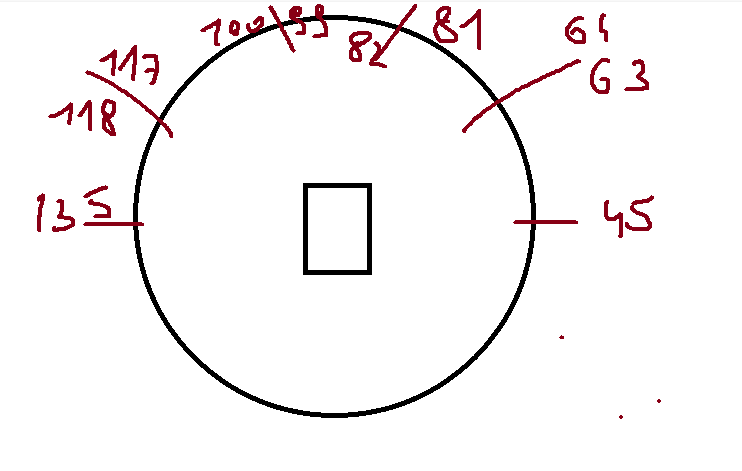
'right': min(min(scan\_data.ranges[45:63]), 10),

            'fright': min(min(scan\_data.ranges[64:81]), 10),

            'front': min(min(scan\_data.ranges[82:99]), 10),

            'fleft': min(min(scan\_data.ranges[100:117]), 10),

            'left': min(min(scan\_data.ranges[118:135]), 10),



*## KĄT FOUNDING  KEKW*

310 linia

tu jest sprawdzanie kątów I innych dupereli

            state\_description = 'case 9 - backup'

273 tutaj się wycofuje ustawia sobie state jakiś tam, state-y masz opisane w 180 linii ( po metodach idzie dojść )

112 linia

   def goToThePoint(self, type, goal\_handle):

tutaj ogarnia gdzie ma iść, na podstawie współrzędnych