


The final project is supposed to be solved in teams of up to three people (no more than three under any circumstance). While you can solve it alone, if you wish, past experience shows that this is not a good idea. You are strongly encouraged to form or join a team.

First follow the setup for lab 7 (either follow the instructions for kinetic or melodic).

SETUP ROS **MELODIC**:

Then, download and put these files in the specified folders:

- [final2019.world](#) in the folder husky/husky_gazebo/worlds
- [huskyfinal.launch](#) and [playpenfinal.launch](#) in the folder husky/husky_gazebo/launch
- [mapfinal.yaml](#)  and [mapfinal.pgm](#) in the folder husky/husky_navigation/maps
- [amcl_final.launch](#) in the folder husky/husky_navigation/launch

SETUP ROS **KINETIC**:

Download and put these files in the specified folders with the provided commands:


NOTE: Downloaded files should be in the 'Downloads' directory located in your home directory for these commands to work.

- [final2019.world](#) in the folder husky/husky_gazebo/worlds

```
roscd husky_gazebo/worlds/  
sudo mv ~/Downloads/final2019.world .
```

- [huskyfinal.launch](#) and [playpenfinal.launch](#) in the folder husky/husky_gazebo/launch

```
roscd husky_gazebo/launch/  
sudo mv ~/Downloads/huskyfinal.launch .  
sudo mv ~/Downloads/playpenfinal.launch .
```

- [mapfinal.yaml](#)  and [mapfinal.pgm](#) in the folder husky/husky_navigation/maps

```
roscd husky_navigation/maps/  
sudo mv ~/Downloads/mapfinal.yaml .  
sudo mv ~/Downloads/mapfinal.pgm .
```

- [amcl_final.launch](#) in the folder husky/husky_navigation/launch

```
roscd husky_navigation/launch/  
sudo mv ~/Downloads/amcl_final.launch .
```

After you have copied the files, build your workspace, source the file `devel/setup.bash` and you are ready to go.

To launch the simulation environment, run

```
roslaunch husky_gazebo huskyfinal.launch
```

This will start an environment similar to what you have used in the labs, i.e., an husky in an indoor-like environment. The `mapserver` node provides an accurate map of the walls in the environment. However, the environment features some additional objects that are not in the map, i.e., tables and mailboxes. Write a ROS program that explores the environment and finds all the mailboxes and tables that are inside. "Finding" an object means reporting its approximate location in the map frame (e.g., mailbox at position 4,6).

- how to recognize and distinguish the objects? use the laser. You can assume that the objects will have the same size of those in the provided example environment.
- how to explore the environment? While a random exploration strategy would do, you are encouraged to use the map provided by `map_server`
- **important:** your solution must be generic with respect to the map. When you will demo your solution, your code will be run with a different environment. However, `map_server` will still provide an accurate map.
- how many nodes should you write? That's a design choice up to you. There is no "best" answer.

At the end you will submit one solution per group and a document (at most two pages) explaining your strategy.

Q&As

Q: can I use external libraries?

A: yes.

Q: can I use ROS packages that are not covered in the lecture notes or in class?

A: yes, you are encouraged to do that.

Q: can my solution consist of multiple programs?, For example, I first run some nodes (e.g., to do some pre-processing on the map) and then run some other nodes to move the robot around.

A: yes

Q: can I use a language different from C++?

A: while multiple languages offer ROS interfaces, this is probably not a good idea. First, C++ and python are the two most common languages used with ROS. Consequently, you will find plenty of online documentation for C++ and python, but much less for the other languages. Second, the TAs will provide assistance if you have issues with C++, but not with other languages (in the lab we just use C++ when interfacing with ROS).