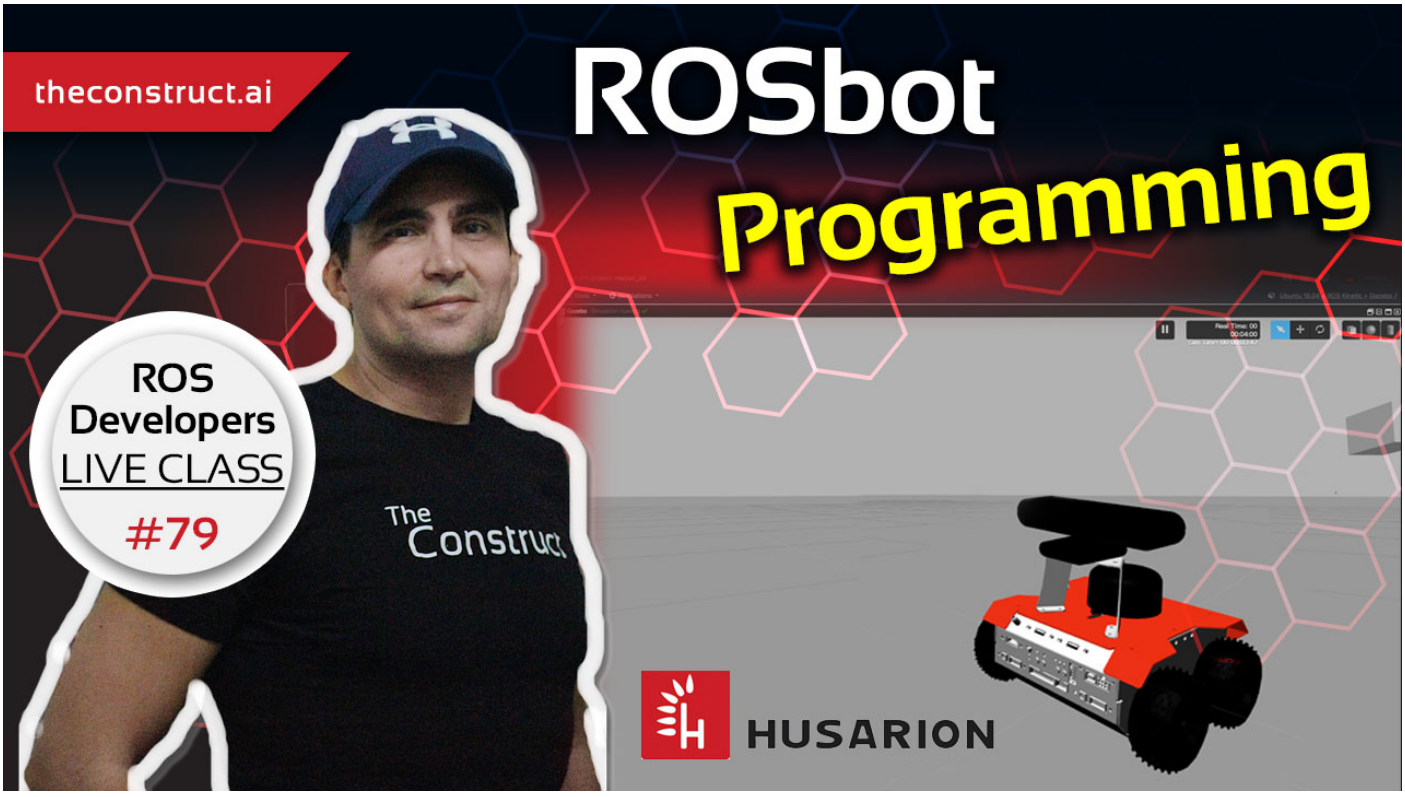
The Construct

Learn and develop for robots with ROS

PRESENTS

ROS Developers Live Class n80



Rosbot programming

In this class, you will learn how to programming the rosbot by husarion

This awesome robot is made by Husarion, if you want more information go their webpage here (https://husarion.com/manuals/rosbot-manual/)

Using the RosBot you will create a map and navigate in this map created, using all the parameters or info of the rosbot needed in the files, in order to work properly both in a simulation and in an environment with the real robot.

If you are interested in becoming a **Robotics Developer** you will need to know how to represent the robot structure in the proper way so you can program it with ROS.

(To know more about becoming a robotics developer, read this guide about How To Become a Robotics Developer (http://www.theconstructsim.com/become-robotics-developer/))

REQUIREMENTS :

- **Basics of Linux**. If you don't have that knowledge, check this FREE online course (https://www.robotigniteacademy.com/en/course/linux-robotics/details/)

- **Ros Basics**. If you don't have that knowledge, check this online course (https://www.robotigniteacademy.com/en/course/ros-in-5-days/details/)

Supplementary Content

- **Ros Navigation**. If you want more knowledge of the topic, check this online course (https://www.robotigniteacademy.com/en/course/ros-navigation-in-5-days/details/)

In this class, we'll learn:

- How to create a map and save it using the ROSBOT.

How to use this ROSject

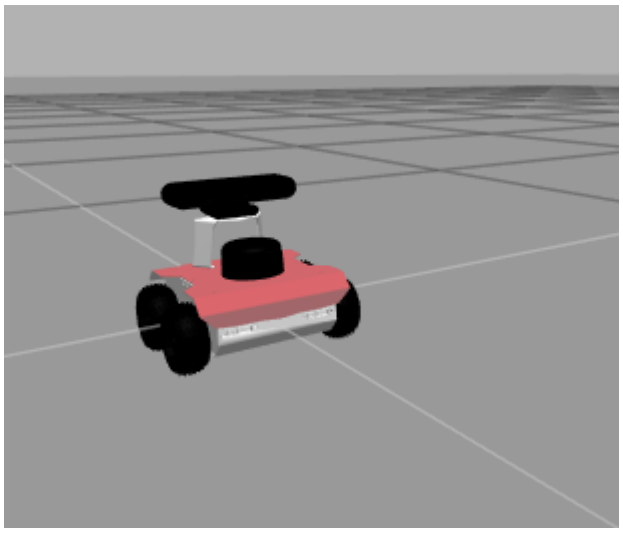
A **ROSject** (http://rosjects.com) is a **ROS project** packaged in such a way that all the material it contains (**ROS code, Gazebo simulations and Notebooks**) can be shared with any body **using only a web link**. That is what we did with all the attendants to the Live Class, we shared this ROSject with them (so they can have access to all the ROS material they contain).

Check this webinar to learn more about ROSjects and how to create your own ROSjects.

You will need to have a free account at the ROS Development Studio (http://rosds.online) (ROSDS). Get the account and then follow the indications below.

Robot for today's Live Class

Today you're going to use the Rosbot by Usarion:



As it says on its website, "ROSbot is an autonomous, open source robot platform based on ROS. Reinforced with a development platform and free online tools such as Web UI, set of tutorials, manuals, simulation model and more". In order to know more about this particular robot, we recommend you to [check husarion's tutorials (https://husarion.com/tutorials/other-tutorials/rosbot-rosds-quick-start/)](https://husarion.com/tutorials/other-tutorials/rosbot-rosds-quick-start/)

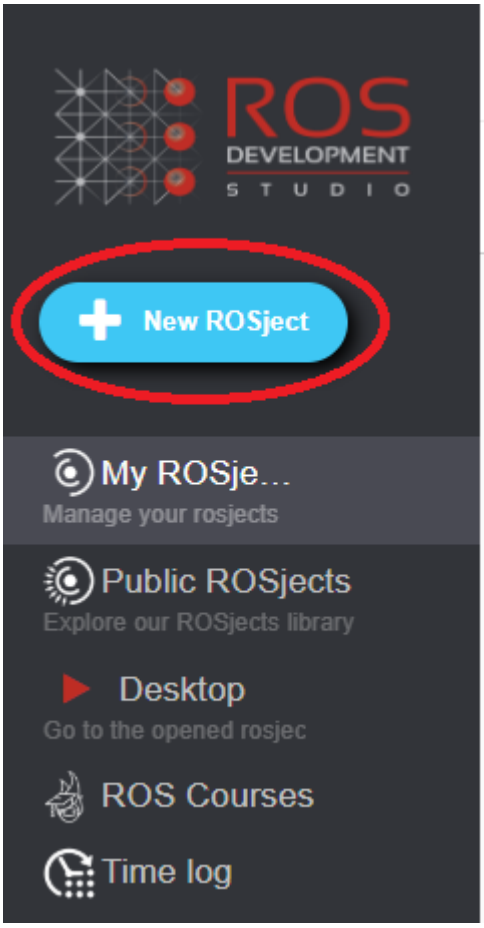
CREATING ROSJECT USING HUSARION TEMPLATE

Preparation.

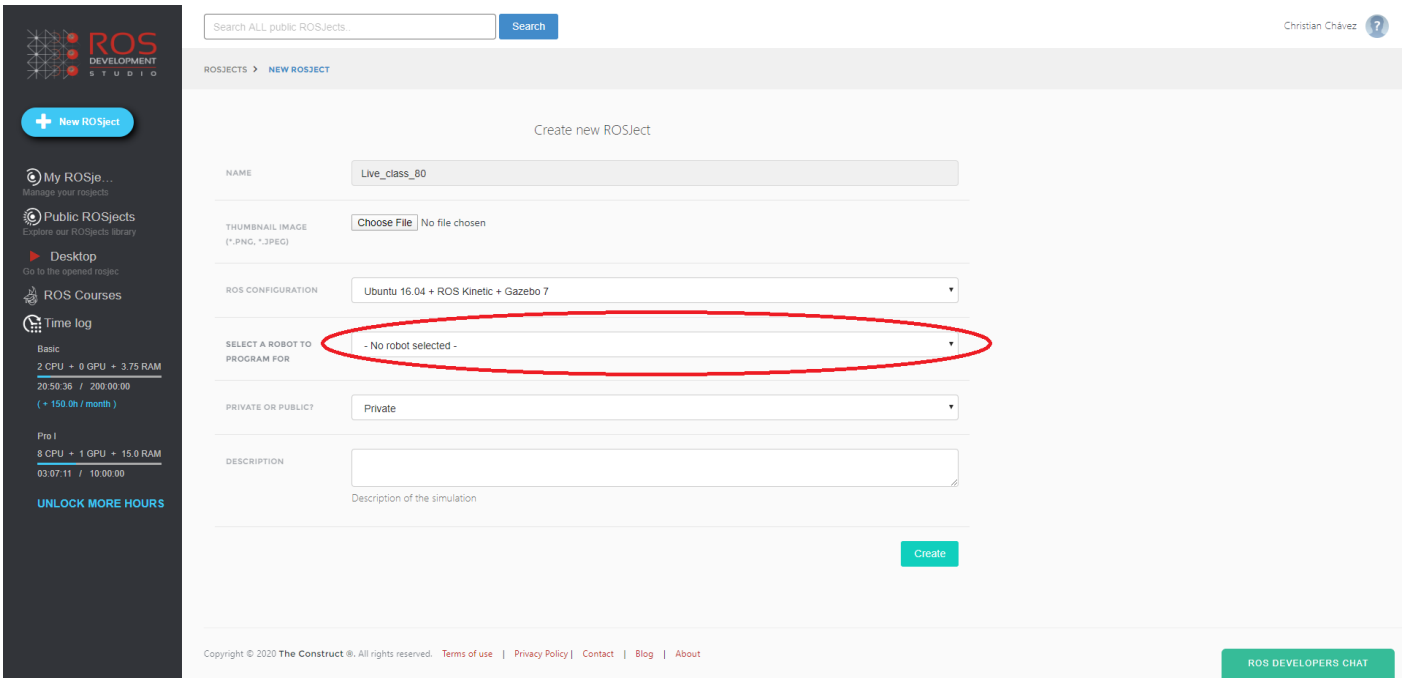
This rosject it's based on the **Security guard robot** by Husarion, you can check all the content of that project in [this link (https://husarion.com/tutorials/ros-projects/security-guard-robot/)](https://husarion.com/tutorials/ros-projects/security-guard-robot/).

In this case, it is very simple because all the needed files are alredy loaded, so you don't have to upload or copy any extra file , it's the best option! ;-)

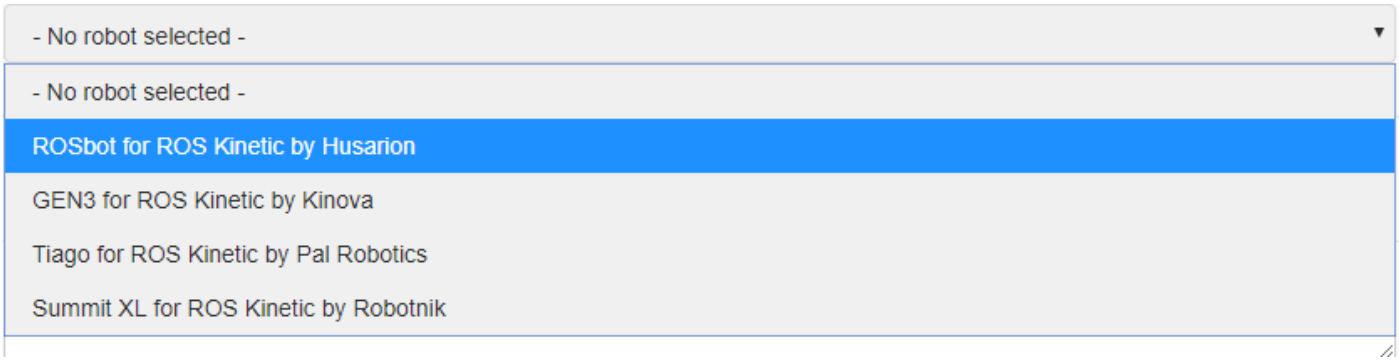
To create a new rosject with the template select the New ROSject.



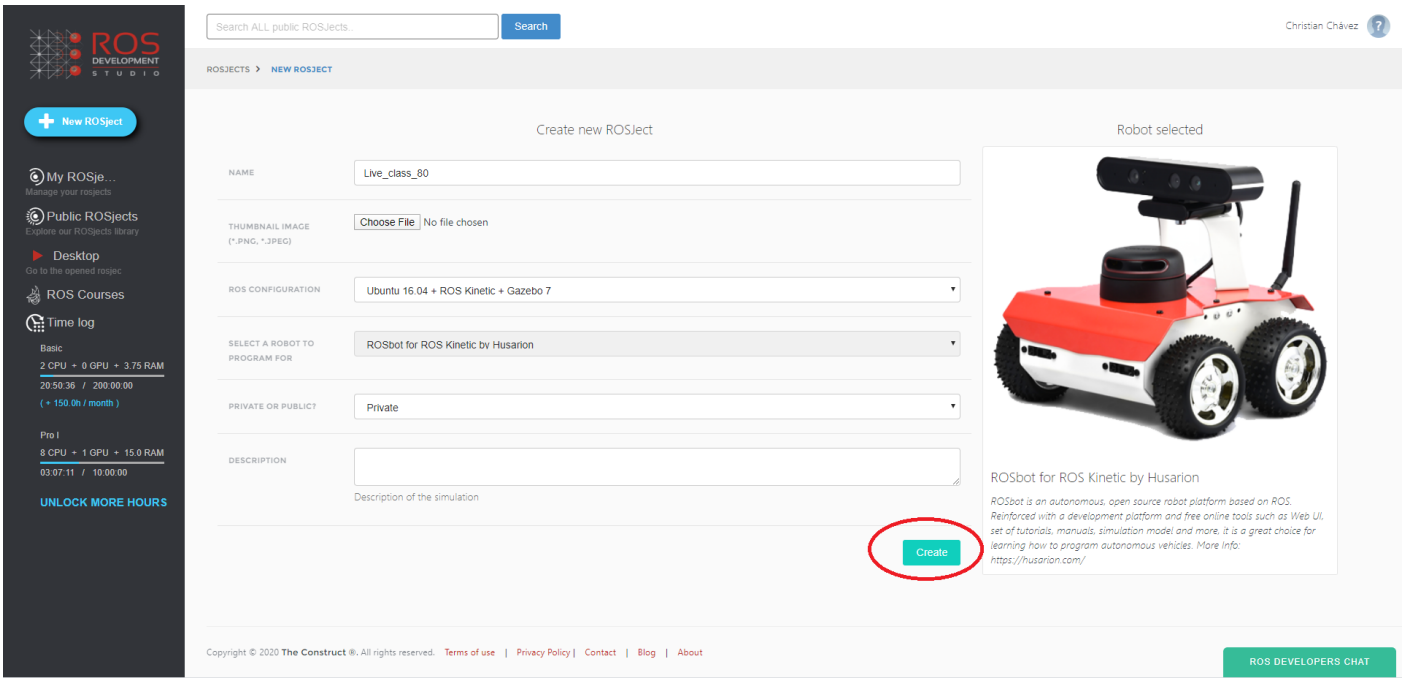
and in the section that says **Select a Robot to program for**



Select the **ROSbot for Kinetiv by Husarion**



Complete all the info needed and choose create.



Now you have your rosject with Husarion Template created, now let's start with the rosject

Creating our packages.

To continue with our structure we will continue creating two packages, one with the code content and the other for simulation, which we will call **rosbot_patrol** and **rosbot_patrol_simulation** respectively.

In the case of the *rosbot_patrol* you have to go to the catkin_ws/src as you did it before, and run the following code in the shell.

In [ ]:
```
$ catkin_create_pkg rosbot_patrol roscpp
```

In the case of the *rosbot_patrol_simulation* you have to go to the simulation_ws/src as you did it before, and run the following code in the shell.

In [ ]:
```
$ catkin_create_pkg rosbot_patrol_simulation roscpp
```

Great!, now you have already created both package let's work with them.

Simulation part

Go to your pkg, running the following command into a shell.

In [ ]:
```
$ cd simulation_ws/src/rosbot_patrol_simulation
```

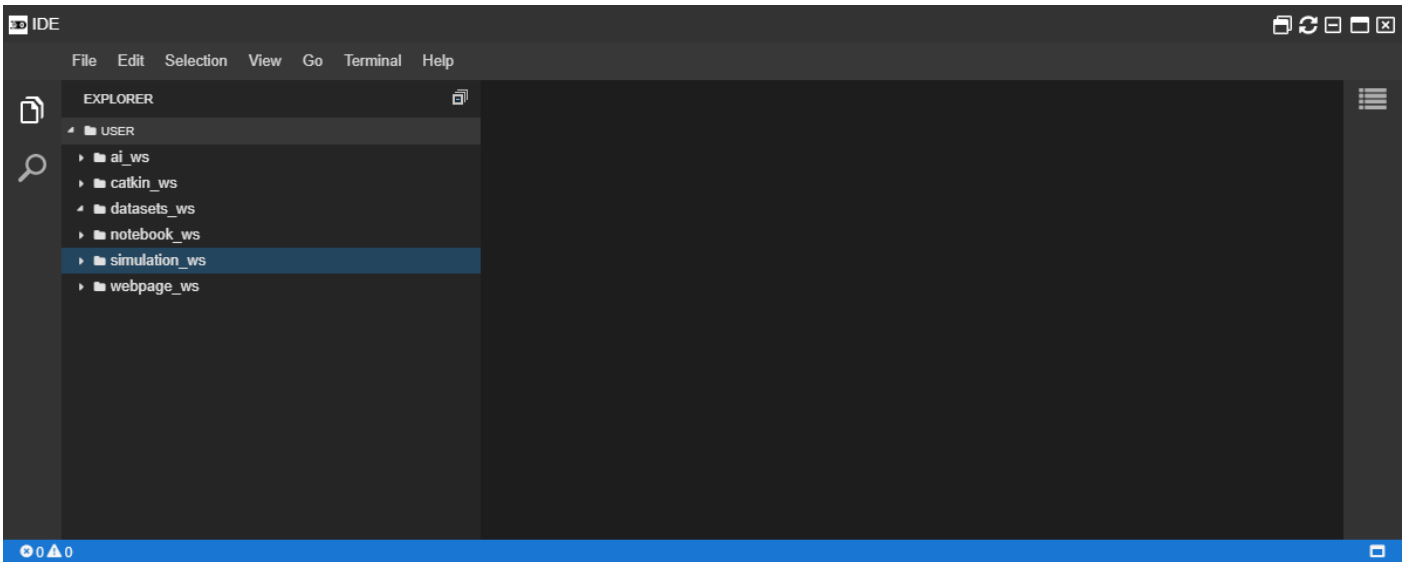
Then create a new folder and call it **wolrds**, with the following command.

In [ ]:
```
$ mkdir worlds
```

go inside this folder and create a new file and call it **model.world**

In [ ]:
```
$ touch model.world
```

Good! Now go to the **Tools** menu again, but now open the IDE.



That should looks like the image before.

The navigate to the file direction, inside simulation_ws/src/rosbot_patrol_simulation/worlds/model.world

Inside the file copy the following code located in this link (https://github.com/adamkrawczyk/rosbot_patrol_simulation/tree/master/worlds).

It just contains all the information of a world where the robot will work, the sun, some walls, etc.

Once you have already copied this file, create a new folder in the src folder of the package and call it launch, you can use the IDE tool this time.

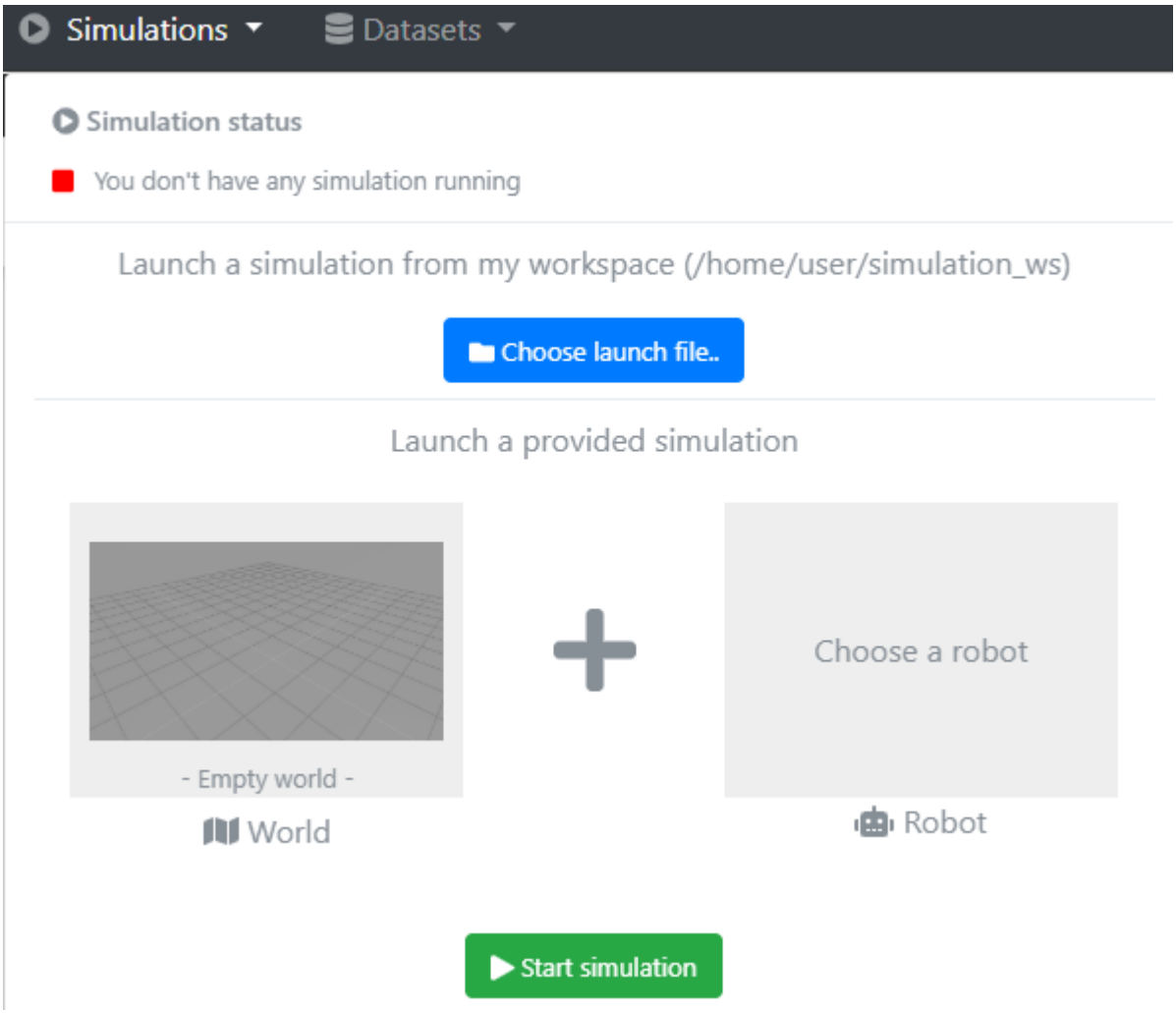
Create a new file and call it **simulation_mapping.launch**

and copy the next code inside it.

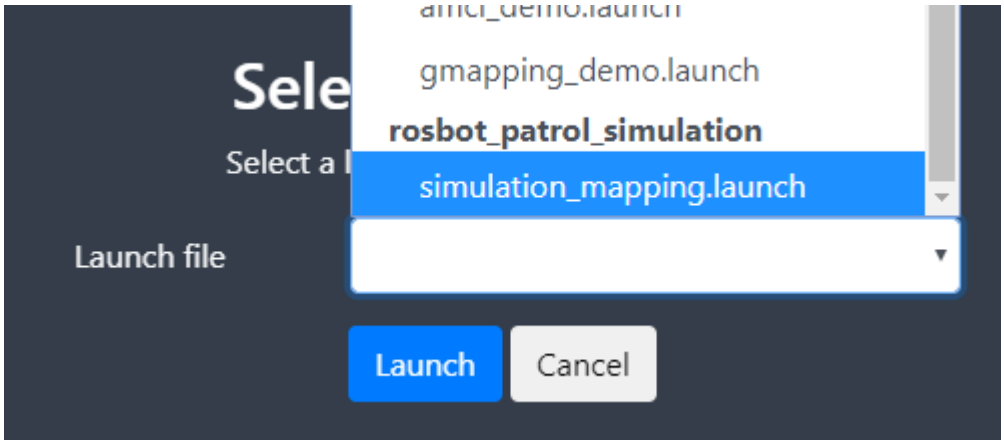
In [ ]:
```
<?xml version="1.0" encoding="UTF-8"?>
<launch>
    <param name="use_sim_time" value="true"/>
    <arg name="world" default="empty"/>
    <arg name="paused" default="false"/>
    <arg name="use_sim_time" default="true"/>
    <arg name="gui" default="true"/>
    <arg name="headless" default="false"/>
    <arg name="debug" default="false"/>

    <include file="$(find gazebo_ros)/launch/empty_world.launch">
        <arg name="world_name" value="$(find rosbot_patrol_simulation)/worlds/
model.world"/>
    </include>
    <include file="$(find rosbot_description)/launch/rosbot_gazebo.launch"/>
    <node pkg="tf" type="static_transform_publisher" name="laser_broadcaster"
args="0 0 0 3.14 0 0 base_link laser_frame 100" />

</launch>
```

As you can se we define the world here, we load the world we add before, the we add the rosbot, with the rosbot_description package and finally we add a static_transform_publisher between the base_link and the laser of the rosbot. As you can see, there is only the data of the simulation.

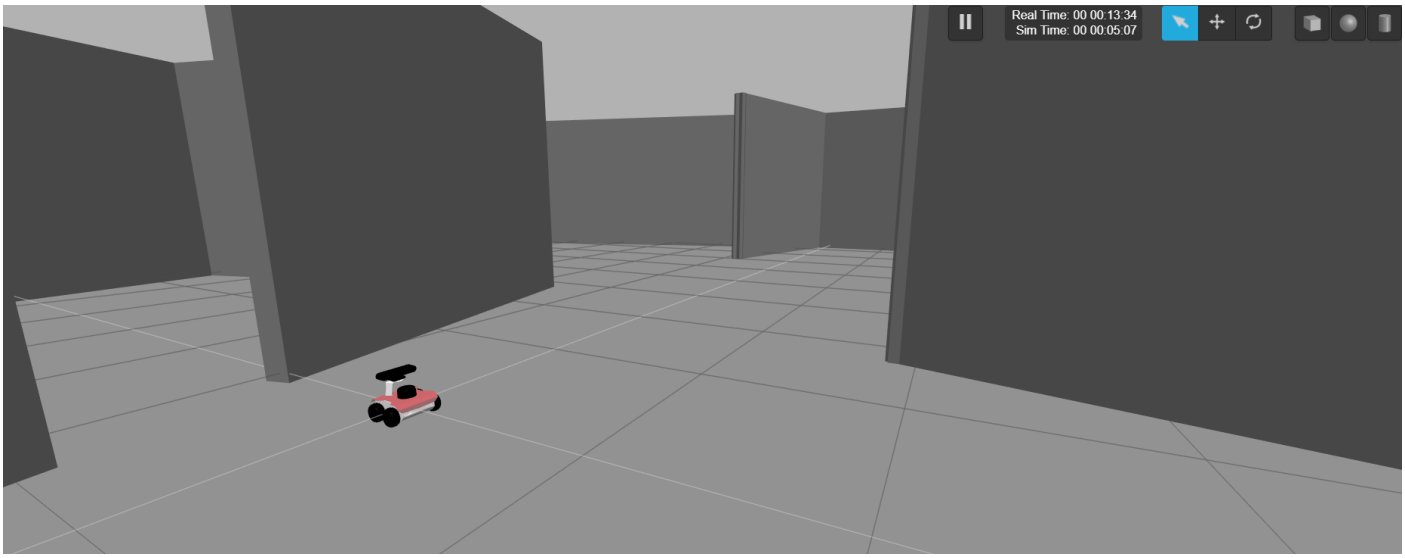
Let's try the simulation now. go to the **Simulations** menu and select **Choose launch file..**



From the package **rosbot_patrol_simulation** launch the launch **simulation_mapping.launch**.



Now you have the simulation running.



Mapping code

Gmapping

Now you have the simulation ready , go to the *catkin_ws/src/rosbot_patrol/src* and create a folder and call it **launch** inside it create a new file and call it **gmapping_only.launch**. Now copy the following code into it.

In [ ]:
```
<launch>

  <node pkg="gmapping" type="slam_gmapping" name="slam_gmapping" output="scree
n">
    <remap from="/base_scan" to="/scan"/>

    <param name="base_frame" value="base_link"/>
    <param name="map_frame" value="map"/>
    <param name="odom_frame" value="odom"/>

    <param name="map_update_interval" value="5.0"/>
    <param name="maxUrange" value="16.0"/>
    <param name="sigma" value="0.05"/>
    <param name="kernelSize" value="1"/>
    <param name="lstep" value="0.05"/>
    <param name="astep" value="0.05"/>
    <param name="iterations" value="5"/>
    <param name="lsigma" value="0.075"/>
    <param name="ogain" value="3.0"/>
    <param name="lskip" value="0"/>
    <param name="srr" value="0.1"/>
    <param name="srt" value="0.2"/>
    <param name="str" value="0.1"/>
    <param name="stt" value="0.2"/>
    <param name="linearUpdate" value="0.05"/>
    <param name="angularUpdate" value="0.05"/>
    <param name="temporalUpdate" value="0.5"/>
    <param name="resampleThreshold" value="0.5"/>
    <param name="particles" value="30"/>
    <param name="xmin" value="-50.0"/>
    <param name="ymin" value="-50.0"/>
    <param name="xmax" value="50.0"/>
    <param name="ymax" value="50.0"/>
    <param name="delta" value="0.05"/>
    <param name="llsamplerange" value="0.01"/>
    <param name="llsamplestep" value="0.01"/>
    <param name="lasamplerange" value="0.005"/>
    <param name="lasamplestep" value="0.005"/>
  </node>

</launch>
```

- Verify that you have a remap from **/base_scan** to **/scan**
- The base_frame is the **base_link**
- The map_frame will be **map**
- The odom_frame is **odom**

And let the rest of the values as the example before, you can check the code in the Husarion example here (https://husarion.com/tutorials/ros-projects/security-guard-robot/)

Move_base

after that, create another file and call it **move_base_only.launch**, copy the following code inside it.

In [ ]:
```
<launch>

    <node pkg="move_base" type="move_base" name="move_base" output="log">
      <param name="controller_frequency" value="25.0"/>
        <rosparam file="$(find tutorial_pkg)/config/costmap_common_params.yam
l" command="load" ns="global_costmap" />
        <rosparam file="$(find tutorial_pkg)/config/costmap_common_params.yam
l" command="load" ns="local_costmap" />
        <rosparam file="$(find tutorial_pkg)/config/local_costmap_params.yaml"
command="load" />
        <rosparam file="$(find tutorial_pkg)/config/trajectory_planner.yaml" c
ommand="load" />
        <rosparam file="$(find rosbot_patrol)/config/global_costmap_params.yam
l" command="load" />
    </node>

</launch>
```

Notice that one file should be in rosbot_patrol_simulation pkg - it's required to create that file because slightly different params will be used.

So let's create the file

Create a new folder in the catkin_ws/src/rosbot_patrol/src and call it **config**, next go to the folder and create a new file and call it **global_costmap_params.yaml**, copy the following code into it.

In [ ]:
```
global_costmap:
  update_frequency: 0.5
  publish_frequency: 0.5
  transform_tolerance: 0.5
  width: 35
  height: 35
  static_map: false
  rolling_window: true
  inflation_radius: 2.5
  resolution: 0.01
```

It is no need to make rest of this files, they are in tutorial_pkg that you have already cloned.

Creating the launch file and save the map

Now go to the launch folder again and create a new file and call it **running_gmapping.launch** copy the following code.

In [ ]:
```
<?xml version="1.0" encoding="UTF-8"?>
<launch>


    <include file="$(find rosbot_patrol)/launch/gmapping_only.launch" />
    <include file="$(find rosbot_patrol)/launch/move_base_only.launch" />

    <node name="teleop_twist_keyboard" pkg="teleop_twist_keyboard" type="teleo
p_twist_keyboard.py" output="screen"/>



</launch>
```

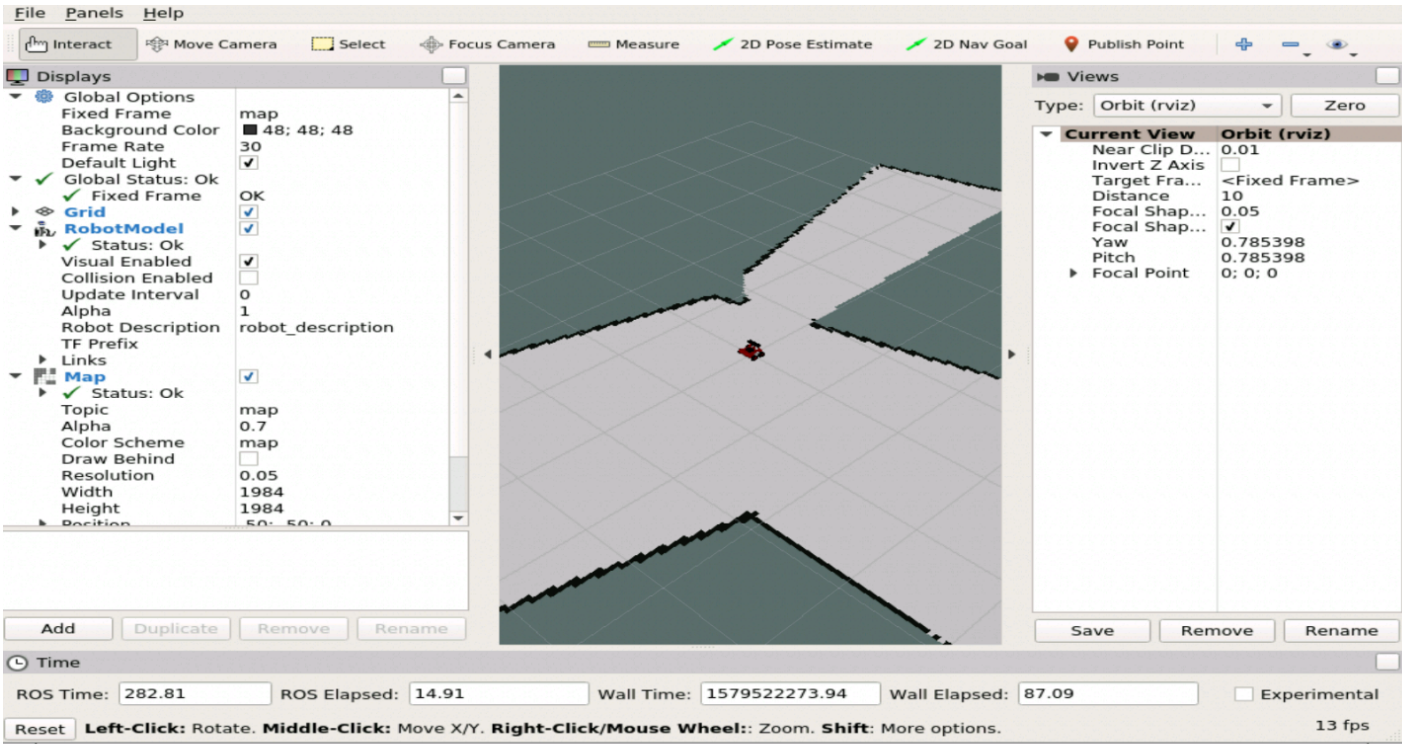
In order to launch this file go to a Shell a run the following command.You can see topics visualization opening **graphical tools** in the **Tools** menu.

In [ ]:
```
$ roslaunch rosbot_patrol running_gmapping.launch
```

create the rviz visualization using the next command in another Shell

In [ ]:
```
$ rviz
```

Now Rviz is opened, add the components needed in order to see the robot model, and the map, don't forget to put "map" as fixed frame and choose the correct topic in the map topic.



now use the teleop_twist_keybord in order to complete the map, and once you have already all the map, go to a new Shell in order to save the using the following commands into a Shell

In [ ]:
```
$ cd
```

In order to exit of any direction in your shell

In [ ]:
```
$ cd catkin_ws/src/rosbot_patrol/src
```

In order to go to this direction

In [ ]:
```
$ mkdir maps
```

In order to create a new folder where save the map

In [ ]:
```
$ cd maps
```

In order to go inside the folder to run the command that will save the map

In [ ]:
```
$ rosrun map_server map_saver -f rosbot_map
```

In order to save the map

Amcl to navigate

Once the map is already saved we need to create launch for amcl to make the robot find it's location on that map. In launch directory make new file called **amcl_only.launch** and copy the following content.

In [ ]:
```
<launch>
    <node pkg="amcl" type="amcl" name="amcl" output="screen">
    <remap from="scan" to="/scan"/>
    <param name="odom_frame_id" value="odom"/>
    <param name="odom_model_type" value="diff-corrected"/>
    <param name="base_frame_id" value="base_link"/>
    <param name="update_min_d" value="0.5"/>
    <param name="update_min_a" value="1.0"/>
    </node>

</launch>
```

Final launch file

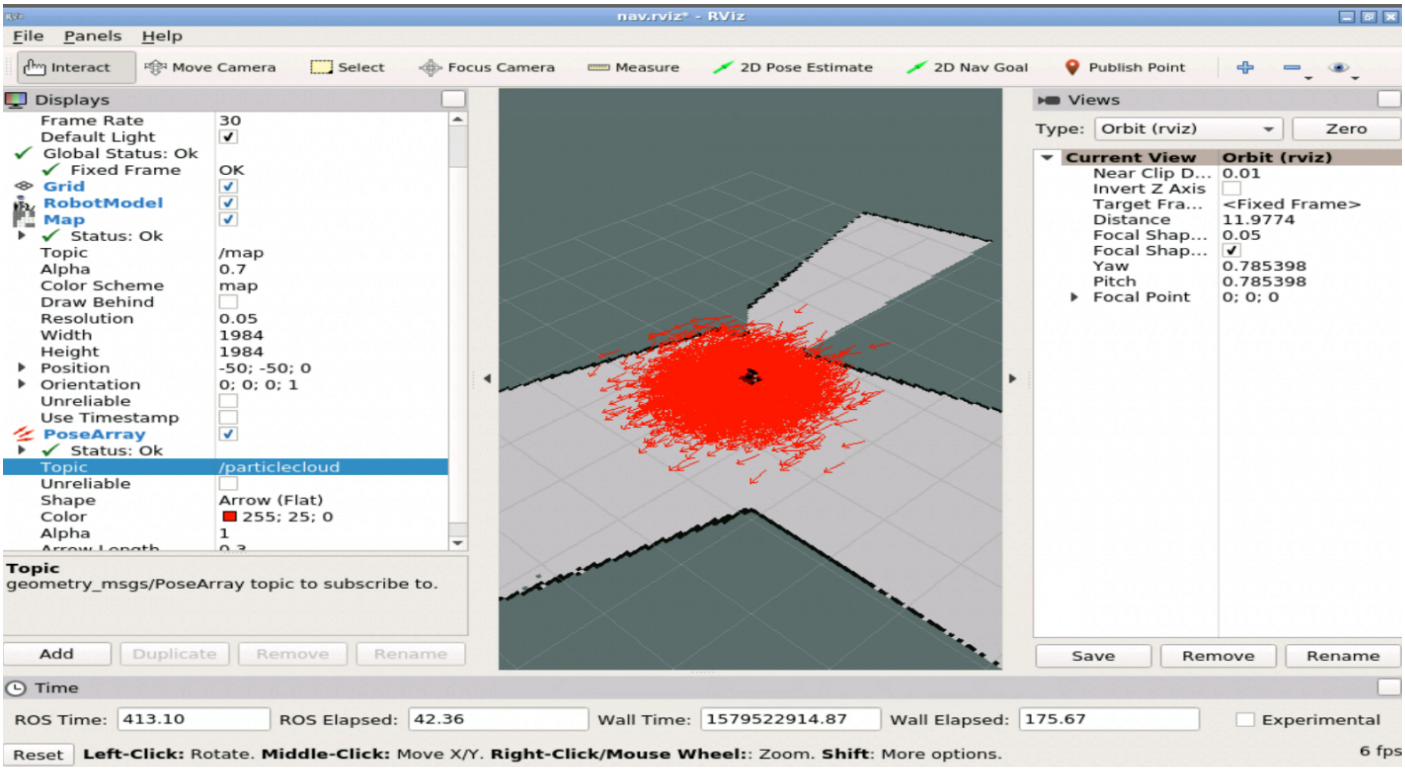
Now create a new file to prove all the content of the rosject. Call this new file as **nav_rosbot.launch**

In [ ]:
```
<?xml version="1.0" encoding="UTF-8"?>
<launch>

    <include file="$(find rosbot_patrol)/launch/move_base_only.launch" />
    <include file="$(find rosbot_patrol)/launch/amcl_only.launch"/>

    <node name="teleop_twist_keyboard" pkg="teleop_twist_keyboard" type="teleo
p_twist_keyboard.py" output="screen"/>

    <!--map server with simul map-->
    <arg name="map_file" default="$(find rosbot_patrol)/maps/rosbot_map.yaml"/
>
    <node name="map_server" pkg="map_server" type="map_server" args="$(arg map
_file)" respawn="false" />



</launch>
```

Launch and see how it works with the following command into a Shell.

In [ ]:
```
$ roslaunch rosbot_patrol nav_rosbot.launch
```

Open Graphical Tools in order to see the visualization of the topics



Connect to the real robot

Once you have installed **rosds_real_robot_connection** that you can check the instructions here (https://www.theconstructsim.com/use-real-robot-connection-rosdevelopementstudio/)

Turn On the Real Robot Connection from the Robots side

Now you have to follow these simple steps:

- Open a web browser. We recommend google chrome.
- Type in the URL: IP_DEVICE:3000
- Here is an example of what you should get if the IP_DEVICE=192.168.1.170 and the user_name_in_device=panandtilt

Not secure | 192.168.1.170:3000

ROSDS Real Robot CLIENT

Device name

panandtilt

The desired name for your robot device

Turn On

Status

Connection: Not ready (No communication)

Device name: **panandtilt**

Robot URL:
-- none --

Provided by TheConstructSim

- Now you have to click on Turn ON. This will generate the Robot URL that you need to make the connection in ROSDS.

192.168.1.170:3000

ROSDS Real Robot CLIENT

Device name

panandtilt

The desired name for your robot device

Turn Off

Status

Connection: Ready! (No communication)

Device name: **panandtilt**

Robot URL:
https://app.husarnet.com/husarnet/[redacted]

Provided by TheConstructSim

- To TURN OFF the connection from the device side, just click on the TURN OFF button. This will sever the link and ROSDS won't be able to connect anymore until you turn it ON again and update the connection with the new Robot URL generated.

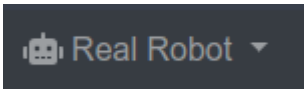
Establish the connection from ROSDS side

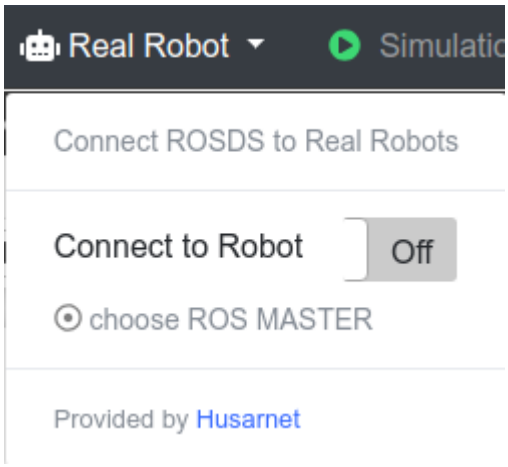
For this last step, you need from the previous step:

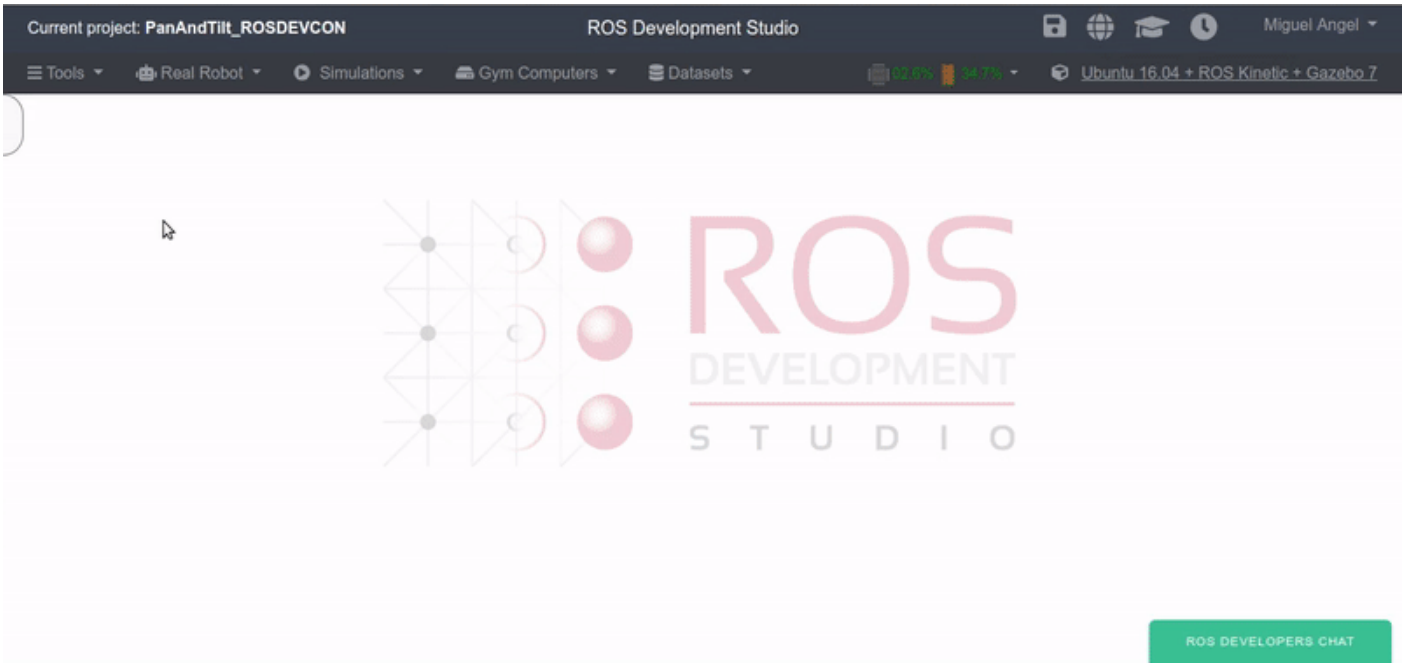
- Robot URL
- Device Name

Follow these steps:

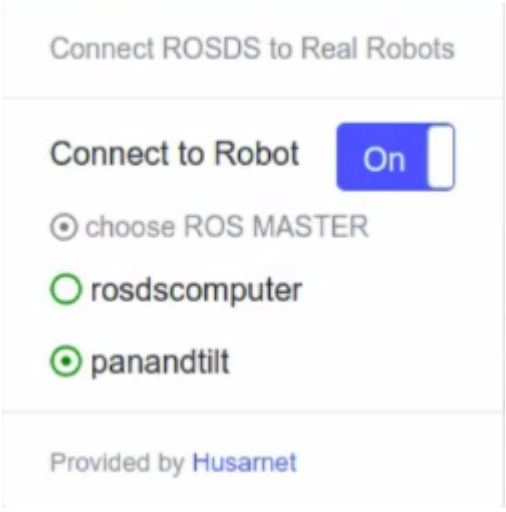
- You have to click the RealRobot tab, Connect to Robot ON, and after a few minutes, you will be greeted with the configuration window.







- Place in the corresponding form input the Robot URL and the Device Name.
- Click on CONNECT.
- After around 5-30 seconds the connection will have been established.

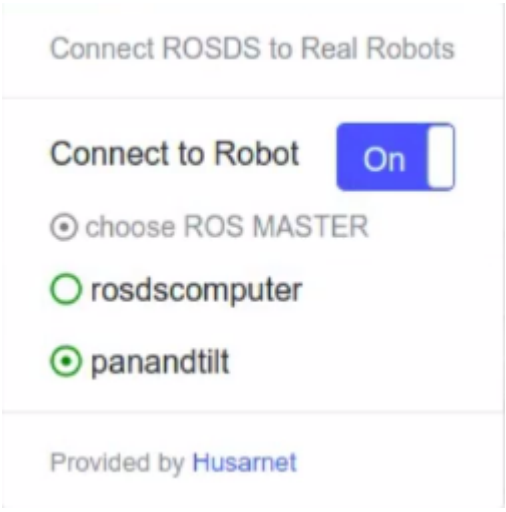


- Now the CONNECTION is ESTABLISHED. By default, the new device is the ROS_MASTER. If you need to change it to ROSDS computer just select it.

Basic ROS Test

Now we can test if ROS is working:

- Remember that you have to decide who is the ROS_MASTER, and therefore where you will have to launch the ROSCORE.



- Inside the Device: rostopic pub /device_test std_msgs/String "data: 'I am The Device'" -r1
  - Output: ERROR: Unable to communicate with master!.
  - Of course, you have to launch the ROSCORE first inside the device if that's the one you had set up as ROSMASTER!
  - Inside the Device: roscore
  - Output2: Nothing. That means that the rostopic publish is working.
- Inside ROSDS web shell: rostopic echo /device_test
  - You should see the message: I am The Device

And now let's test the other way round:

- Inside ROSDS web shell: rostopic pub /rosds_test std_msgs/String "data: 'I am ROSDS'" -r1
- Inside the Device: rostopic echo /rosds_test

Trying the running_gmapping.launch.

Once the real conection is stablished, launch the file and see how the real conection will work.

In [ ]:
```
$ roslaunch rosbot_patrol running_gmapping.launch
```

GOOD JOB!

Mission completed!!

But is there something to install while I'm working with ROSDS?

If you are working in **ROS DEVELOPMENT STUDIO** you have all the components needed for this rosject already installed, but if you are working in a local computer you have to follow the next steps.

What to install: Mailbox (setup this on robot also): chose internet setup set remaining parts as default during installation and setup - sudo apt install postfix - sudo service postfix reload - sudo apt install mailutils - sudo apt-get install sendmail - sudo dpkg-reconfigure postfix - sudo /etc/init.d/postfix reload yaml parser This can be installed anywhere eg. or in <ros_ws/src>. Go to desired directory and paste:

- git clone https://github.com/jbeder/yaml-cpp.git (https://github.com/jbeder/yaml-cpp.git)
- cd yaml-cpp
- mkdir build
- cd build
- cmake ..

If you liked this video, please support us!

Really... we need your support!!!!

How can you support us?

1. Subscribe to our ROS online academy and become a Master of ROS Development

Go to our online academy. There is no faster way and funnier to learn ROS because we use the same method we did here.

We call the 30/70 method

- **30% of the time learning theory**
- **70% of the time practicing with simulated robots**

robotignite ACADEMY

Search Paths, Courses and Live Classes ... Search

RICARDO

- Your learning — Continue your learning
- Paths — Follow a guided path to ROS learning
- Courses — Explore our Courses library
- Live Classes — Attend a live/recorded ROS class
- ROS Development Studio — Practise and share your knowledge
- Accomplishments — View your certificates of completion

Basic ROS · 5 Day(s)

ROS Basics in 5 Days

Learn the fundamentals of ROS to understand and be able to program robots.

Python

Start Course | Details

Basic ROS · 5 Day(s)

ROS Basics in 5 Days (C++)

Learn the fundamentals of ROS to understand and be able to program robots.

C++

Start Course | Details

Basic ROS · 5 Day(s)

ROS2 Basics for C++ in 5 days

Learn ROS2 basics now. It doesn't matter if you are new to ROS or a veteran, ROS2 is ...

C++

Start Course | Details

Basic ROS · 5 Day(s)

Python 3 for Robotics

Master the basics of Python 3 for robot programming

Free Python

Start Course | Details

Basic ROS · 5 Day(s)

Linux for Robotics

Learn the Linux fundamentals you'll need for robotics development

Free Python

Start Course | Details

Robot Creation · 5 Day(s)

Your First Robot with ROS

Creating your first ROS based Robot from Scratch.

Artificial Intelligence · 5 Day(s)

Deep Learning with Domain

Learn how to train any robot to recognize an object and pinpoint its 3D location with

Navigation · 5 Day(s)

Fuse Sensor Data to Improve Localization

Learn how to fuse GPS, IMU, odometry and other sources of localization.

ROS Debug · 5 Day(s)

Unit Testing with ROS

Learn how to perform Unit Tests with ROS on the 3 main levels of testing: Python tests,

Navigation · 5 Day(s)

ROS Navigation in 5 Days

Learn how to make your robot navigate autonomously by using the ROS Navigation

GET HELP

https://www.robotigniteacademy.com/en/course/unit-testing-ros/details/

robotignite ACADEMY

Search Paths, Courses and Live Classes ... Search

RICARDO

- Your learning — Continue your learning
- Paths — Follow a guided path to ROS learning
- Courses — Explore our Courses library
- Live Classes — Attend a live/recorded ROS class
- ROS Development Studio — Practise and share your knowledge
- Accomplishments — View your certificates of completion

Most Popular

Basic ROS · 5 Day(s)

ROS Basics in 5 Days

Learn the fundamentals of ROS to understand and be able to program robots.

Python

Start Course | Details

Basic ROS · 5 Day(s)

ROS Basics in 5 Days (C++)

Learn the fundamentals of ROS to understand and be able to program robots.

C++

Start Course | Details

Basic ROS · 5 Day(s)

ROS2 Basics for C++ in 5 days

Learn ROS2 basics now. It doesn't matter if you are new to ROS or a veteran, ROS2 is ...

C++

Start Course | Details

Basic ROS · 5 Day(s)

Python 3 for Robotics

Master the basics of Python 3 for robot programming

Free Python

Start Course | Details

Basic ROS · 5 Day(s)

Linux for Robotics

Learn the Linux fundamentals you'll need for robotics development

Free Python

Start Course | Details

Robot Creation · 5 Day(s)

Your First Robot with ROS

Creating your first ROS based Robot from Scratch.

Artificial Intelligence · 5 Day(s)

Deep Learning with Domain

Learn how to train any robot to recognize an object and pinpoint its 3D location with

Navigation · 5 Day(s)

Fuse Sensor Data to Improve Localization

Learn how to fuse GPS, IMU, odometry and other sources of localization.

ROS Debug · 5 Day(s)

Unit Testing with ROS

Learn how to perform Unit Tests with ROS on the 3 main levels of testing: Python tests,

Navigation · 5 Day(s)

ROS Navigation in 5 Days

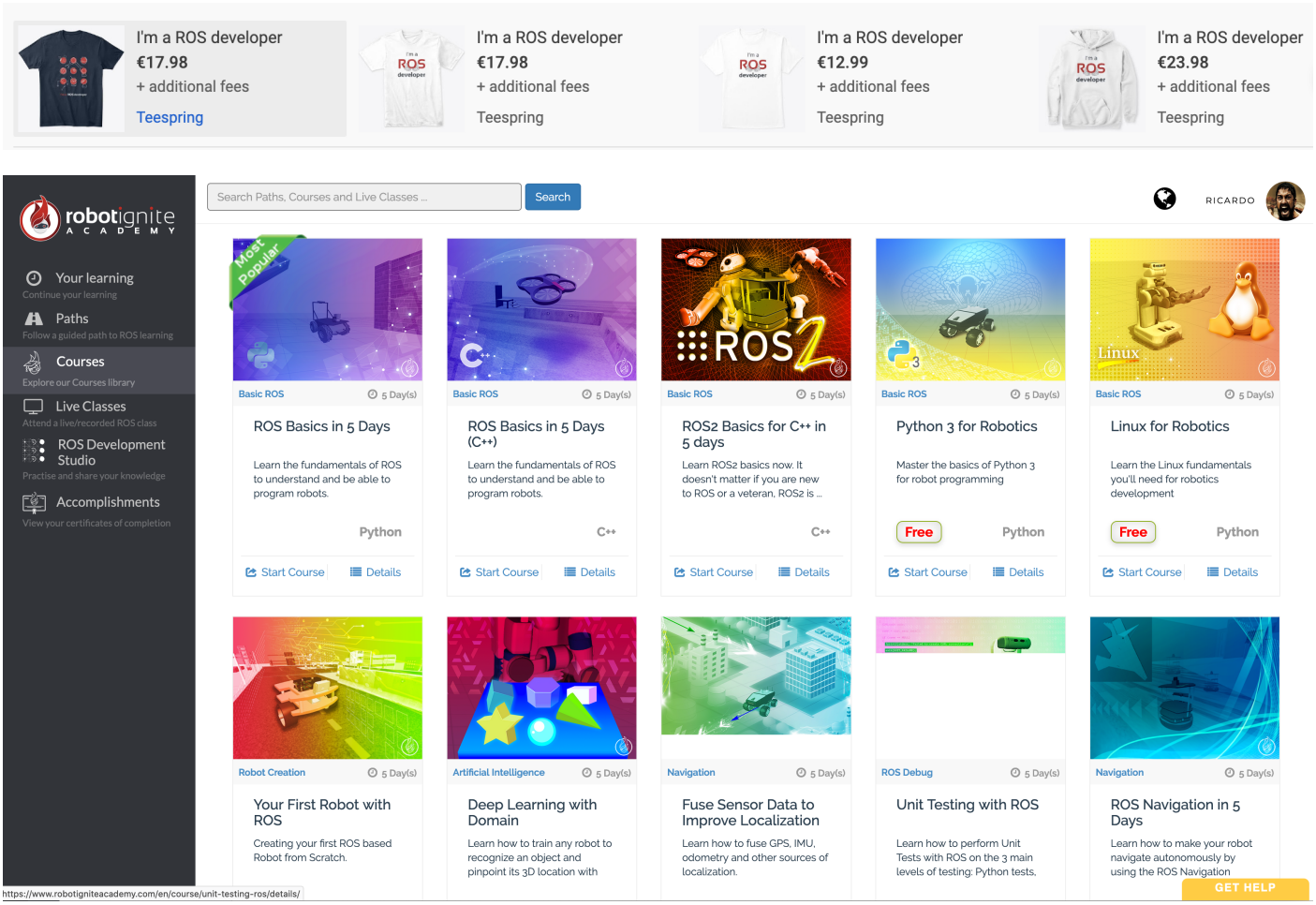
Learn how to make your robot navigate autonomously by using the ROS Navigation

GET HELP

https://www.robotigniteacademy.com/en/course/unit-testing-ros/details/

Check it out at [http://robotignite.academy (http://robotignite.academy)](http://robotignite.academy)

2. Buy one ROS Developers T-shirt!



You can buy them at our Teespring area ([https://teespring.com/stores/ros-developers (https://teespring.com/stores/ros-developers)](https://teespring.com/stores/ros-developers))

3. Give us a like in Youtube and subscribe to the channel

- **Go to our Youtube Channel (https://www.youtube.com/channel/UCt6Lag-vv25fTX3e11mVY1Q (https://www.youtube.com/channel/UCt6Lag-vv25fTX3e11mVY1Q)) and subscribe (it is free!!!)**
- **Give us a like to this video**

KEEP PUSHING YOUR ROS LEARNING WITH PATIENCE AND GOOD HUMOUR!

Build the future, Become a ROS DEVELOPER