



The  
Construct


Learn and develop for robots with ROS

ROS



# HOW TO TEACH ROS WITH NO HASSLE

A GUIDE FOR TEACHERS TO IMPROVE THEIR STUDENTS  
ROBOTICS PROGRAMMING SKILLS FAST... AND WITH NO HASSLE!

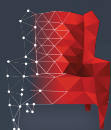


This guide has been written by  
the Robot Ignite Academy team.

[www.robotigniteacademy.com](http://www.robotigniteacademy.com)

You can distribute it as you want as  
far as you give credit to the authors.

This guide is published under  
GPL license.



The  
**Construct**

Learn and develop for robots with ROS



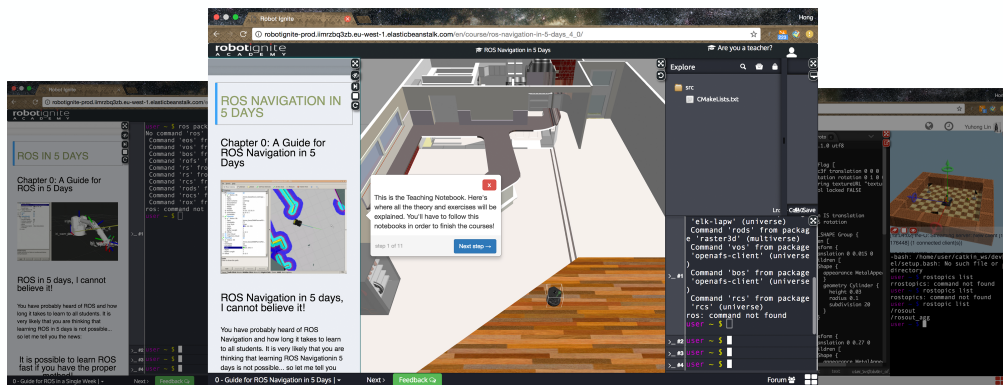
# ***Table of Contents***

- Introduction
- Teach ROS Fast : The theory behind the method
- How to implement the theory
- Available online platforms
- Conclusion

# INTRODUCTION

## *Teach ROS Effectively*

As a teacher of robotics, you are probably concentrated on teaching your students **how to localize a robot using a Monte Carlo particle filter**, or that you want your students learn **how to compute path planning for a robotic arm** in order to reach the glass in the table, or you may want your students make a robot **recognise an object on the table**. Given your experience, you believe that teaching ROS to the students will speed things up (since many algorithms are already implemented in ROS, ready to use). Also, you would be teaching the students a standard in robotics programming. Hence, you need to teach ROS.



**ROS** is a very powerful system that **allows the standardization and normalization of algorithms for robots**. But at the same time, it is a difficult framework to learn. Even if there are many good tutorials around, the novelty of the concepts is hard for students.

In this document, we propose you a method for teaching ROS fast **with little or no hassle for the teacher nor for the students**.

The method consists of going straight to the core ROS subjects and discard the rest. Also relies on tons of practice (with robots).

We have divided the rest of the document into three parts:

1. The theory behind the method
2. How to manually implement the theory for your own courses
3. How to use an already working system online

Let's start with point 1.

“ *But teaching ROS to students can be a daunting process.* ”



ROS

1

PART ONE

# *The theory behind the method*



The  
Construct

Learn and develop for robots with ROS

# The theory behind the method

## PART 1

Our claim presents a big challenge: **let's make students learn ROS in a week.**

Guau! I don't know if I can believe you, you may think.

But we have assessed it true with dozens of students that came to our facilities from everywhere of the world (Japan, USA, Spain, Italy, Portugal, UK, Israel, Germany...) and successfully learnt ROS with this method (you only have to read the comments they left us in our web page).

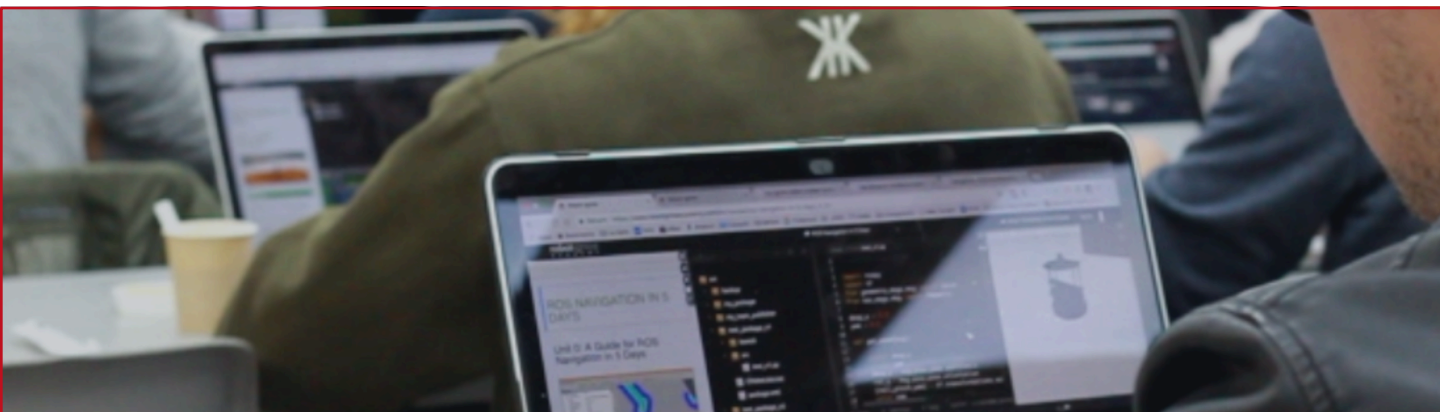
In a single week.

So **what is so special with our method that allows such fast learning?**

Well, other learning materials teach ROS to the student with all the ins and outs of every specific ROS subject, following a logic order (a logic order for a person who knows ROS, not for a newcomer). We call those teaching materials as **reference materials**. We say that those materials are **built for the teacher not for the student**, in order to allow the teacher structure in his head what he already knows,

and make him able to access the material anytime he needs to check something. Basically they are building reference guides.

Hence, it requires a big effort from the student to swallow all that material from moment one. It is like if you want to build a puzzle of 10.000 pieces, and you start looking for the piece of the top left corner, then continue looking for the piece that goes to its right, then you keep adding pieces to the right until you reach the right top corner piece. At that point, you look for the piece beneath the top left corner and fill with pieces another line beneath the previous one. You keep adding lines until you complete the whole puzzle. Sure you can achieve it, but it is going to take you ages. That is hard.





At The Construct we prefer a different approach. We believe that **learning is someway chaotic**, it doesn't follow a structured path. Instead, students jump from one place to another depending on what makes sense to them and what they want to achieve with the robot. It is only later that the whole picture, the one of the reference material, takes place in their mind.

Instead of building the puzzle starting from the top left corner piece, we would look for a piece that makes sense to us, like an eye of a face. Then we look for the rest of the pieces that conform the face of the character and keep building. Then the whole body of the person emerges. By when we are reaching the corner pieces, we have the support of the other pieces that are already in place, all of them making sense to us. Oh, now I can see it! It was a puzzle of Wolverine! This method is a lot faster... and a lot funnier!

“*Move your class from PASSIVE LISTENING to ACTIVE PRACTICING.*”  
**Robot Ignite Academy**

OK, so how do we achieve to build the puzzle of ROS starting from the middle? By following a learning method described in the book *The First 20 Hours*, by Josh Kaufman.

The method consists of four parts.

## DECONSTRUCT

We must deconstruct ROS for the students. This means that we must identify the very important parts (sub-skills) that students need to learn in order to understand most of ROS programs. The 20% that allows to understand the 80%.

Hence, we are going to teach here only those parts. This is a huge reduction of materials, but more important, is a huge reduction of time spent trying to understand things that are not relevant (at least for this stage of learning).

This step requires a deep work from the teacher, since he has to know ROS deeply, and then be able to identify the important parts that really matter. And above all, discard material. Discard, discard, discard!

I know, we want to express everything we know about the subject, but that is not what is required for fast learning.

## LEARN

We guide the student learning by focusing them on those subskills, and letting them know very early in the process when they are doing wrong. How do we do that? By using simulations of real robots with real physics. By using that simulations based environment, the student can practice from minute one and see in real time how a real robot will react to his commands and programs. This quick reward closing loop is essential for fast learning.

## REMOVE

We remove all the barriers that slow learning. First we remove all the lessons and specificities that only introduce noise in the learning of the newcomer, like for example, how to install ROS and Gazebo.



Second, we remove theory lessons and substituted them for 100% practical lessons which makes learning ROS a lot more appealing. And third, we remove practicing barriers by providing the Notebooks and the simulations already made and working for the student, which allow practicing from minute one.

## PRACTICE

Then, we make the students practice, practice, practice. All along the course. Actually, the whole teaching material must be a long practice, where small sub-skills are practiced at each lesson (using the Notebooks and the Simulations), which concludes in a full robotics project of 5 days of work.

OK all those words are very nice, but how can we actually apply them for our purpose of teaching ROS fast? Well, that is what is explained in the next section.

“

*Want to make them learn? Make them practice!*

**Robot Ignite Academy**

”

ROS

2

PART TWO

# *How to implement the theory*



The  
Construct

Learn and develop for robots with ROS

# How to implement the theory

## PART 2

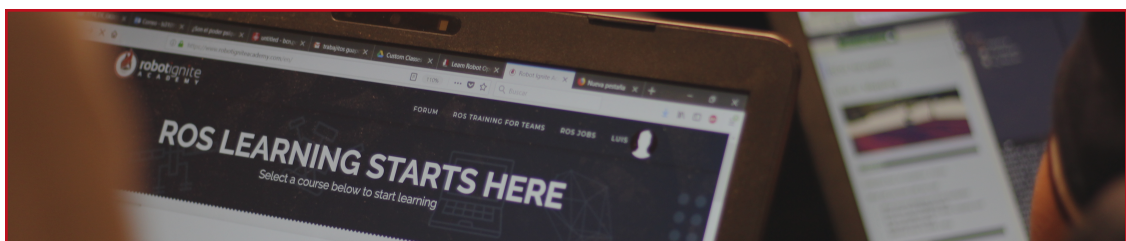
For the implementation of the theory you will need to prepare several things:

- First, you will need to **create the syllabus**. That is the content of the course that you will teach to your students. **The content will be created using Jupyter notebooks**. So basically, this point consists of deciding what are you going to teach and then writing the notebooks with the lessons that teach the content. The advantage of this point is that the notebooks can contain multiple control elements that allow the student to directly interact with the content and practice with it in real time. For example, you can embed Python code in the notebook, that the student can execute directly on the notebook and see the result right there. We are going to see below how to create that content and embed code, controllers, visualizers, etc.



- Second, you will need to **create the simulations** that will be attached to each block of teaching content. Only using the notebook only for teaching is a huge improvement in the learning experience of the student. However, the use of simulations increases the speed of learning several orders of magnitude, since the student can see the results of his actions on the (simulated) robot.
- Third, you will have to **provide access to that material** to your students. You must setup some place where the students can access the notebooks and simulations.

We call our method the NOSI approach to teaching ROS (NOtebooks + Simulations). Let's go step by step. But first, let's see what you need to install in your computer to make all the magic possible.





## What to install

In order to make everything work, you first need to install in your computer (and the ones of the students) the following software (everything for free):

- Install ROS + Gazebo packages ([www.ros.org](http://www.ros.org)).
- Install Anaconda (provides Jupiter notebooks)([www.anaconda.com/download](http://www.anaconda.com/download)).

Even if Anaconda exists for every operating system, ROS and Gazebo do only run at present for Linux and MacOS. Be aware of that when selecting your computer for the courses.

## Creating the syllabus

You have several options to build the teaching lessons for your students, being slides the most used method, and one of the worst for teaching fast.

With slides, the teacher feels comfortable and structures the content for himself. That is a very bad approach for the student who may feel like he has all the required material in the slides even if it is not (since slides have a very reduced space). Slides are a very good summary for somebody who already knows and wants to remember, but it is not a good method for students who know nothing about the subject. Also, slides provide very little space for interaction with real robots (if any). Slides are very static.

In this guide, we propose you to use instead **Python notebooks**.

Python notebooks are web-based notebooks that integrate text with Python code.

The screenshot shows a Jupyter Notebook interface in a web browser. The notebook is titled 'autonomous\_Unit2\_GPS (unsaved changes)'. The code defines a class 'MoveToGpsWayPointAServerClass' that implements a ROS action server. The code includes imports for rospy, actionlib, gps\_reader, and actionlib.msg. It defines a TestAction class with fields for goal, result, and feedback. The class constructor initializes the action server, sets up a node, and defines the feedback and result messages.

```

In [ ]: #!/usr/bin/env python
import rospy
import actionlib
from gps_reader import GpsClass, WayPoint
from actionlib.msg import TestFeedback, TestResult, TestAction
"""
### Test.action ###
int32 goal
-----
int32 result
-----
int32 feedback # This will give the distance from current pos to waypoint
"""

class MoveToGpsWayPointAServerClass(object):
    def __init__(self):
        # creates the action server
        rospy.init_node('move_to_gps_waypoint_server_node')
        # create messages that are used to publish feedback/result
        self._feedback = TestFeedback()
        self._result = TestResult()

```

Why to use Python and why to use notebooks?

1. By using Python to teach ROS you are smoothing the learning curve of ROS. Even if ROS can work with C++ or Python, it is a lot easier to build Python ROS programs than C++ programs. It is a lot faster to test Python programs and do not have to deal with compilation issues (that in a subject like ROS, will take you most of the time).

*Notebooks can be shared, updated and published, and what they contain is alive actionable knowledge*

**Robot Ignite Academy**

2. You can **execute code embedded in the notebook** without having to create actual Python files. This is very interesting when you want to present to your students with a code example that already works without any additional action from the student (which allows to progressively introduce a subject to the students). On a notebook, you can embed Python code or Linux shell code.
  
3. It also allows to **generate data on the fly**. This is very interesting for making the students see how the explanation of the notebook changes based on the data obtained at the precise moment. That is, the things that the notebook shows can depend on the data that you provide. For example, you can embed into the notebook a scan of the laser of the robot, with the actual values scanned at that precise moment. I do not mean to take scan at your home and put it as a figure on the notebook. I mean, that you put some code that when executed will connect to the laser topic, obtain a current scan of the robot and show the laser data integrated in the notebook. If you move the robot, the scan showed will change. Everything in real time.



4. In the same sense, you can embed controls in the notebook that can be used to act upon the robot.

As you can see, using a notebook increases in several orders of magnitude what the teacher can do to present to the students. It increases interactivity and decreases the step into each topic.

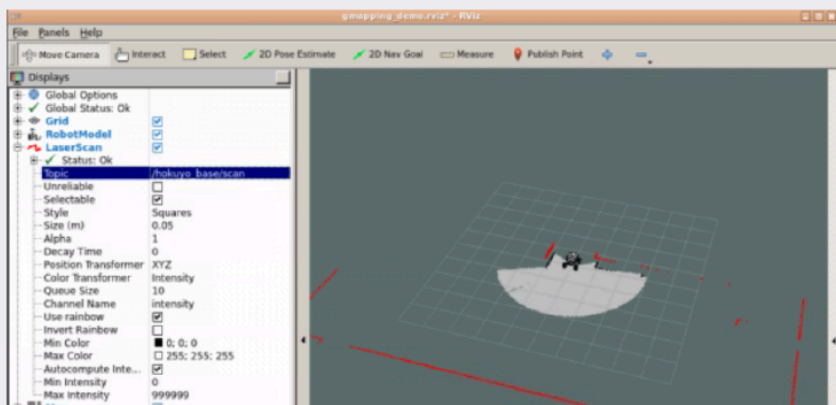
So what you need to do first is to install in your computer the software that will allow you to create Python notebooks. The software is called Jupyter and you can download from [here](https://jupyter.org) (multiplatform): [jupyter.org](https://jupyter.org)

4. Launch the node using the launch file you've just created, and create a map of the environment.

In order to move the robot around the environment, you can use the keyboard teleop. To launch the keyboard teleop, just execute the following command:

```
roslaunch summit_xl_gazebo keyboard_teleop.launch
```

Also remember to launch Rviz and add the proper displays in order to visualize the map you are generating. You should get in RViz something like this:



Launch the Jupiter notebook with the following command:

```
> jupyter --notebook
```

Jupyter is the editor, the program that you will use to create your notebooks. Familiarize with it and configure it accordingly so you can access your ROS code from any notebook you create. How can you test that you can access ROS from the notebook? Very simple. Just type the following in a cell of the notebook:

```
import rospy
```

Then press *Shift+Return* to execute that cell (or the Play icon button). If the notebook does not show an error message, then it means that you can access ROS from the Jupiter notebooks. In case you cannot, please read properly the documentation of Jupyter. Take into account that, in order to be able to access ROS from the notebook, you must have started it from a shell that has access to ROS itself.

“

*Create a very successful MOOC by applying the NOSI framework (and use the RDS to distribute it for free ;)*

**Robot Ignite Academy**

”

If installation was successful with ROS access from the notebook itself, then you are able to start creating your notebooks which contain ROS code. Do tests with typing different text, add code, add controllers, etc...

Just one remark: if you want to execute ROS code from the notebook, you will need to have running a *roscore*, either in a robot connected to your computer or in a simulation in your own computer. Ensure that your `ROS_MASTER_URI` points to the proper IP.

Now let's proceed to include the material required for the course.

*c*

You can include text explanations at anytime in the notebook.

In order to include text, select an empty cell and just type. You may want to select the format in which you are typing (use the drop down menu at the top of the page). If you use HTML or Markdown format to write the text, select the *Markdown* format of the cell.

Once you have typed the text in the appropriate format, just press *Shift+Enter* to Execute the content of that cell. Because the format is some of the text types, by executing, the notebook will render the text and show it with the proper visual look.

### ***Embedding code***

You can embed code on a cell by selecting it and typing the code there. For code cells, select the *Code* format.

You can paste the code straight on the cell. Whenever somebody presses *Shift+Enter* when that cell has focus, it will automatically execute the code. When the cell code is being executed, the number of the cell (on the top left side of the cell) will change from the actual number to a star (\*). That will indicate that the code is still running and has not finished yet (it may never finish if you wrote a spin cycle). To stop the program at anytime, press the *Stop* button (with the cell of the code selected).

The result of the execution of a program will be shown below the code cell.



## Shell code

You can embed bash shell code in a cell and when you execute, it will be executed on a bash shell on the computer. The format for a shell code in a cell is as follows:

```
!your_shell_command
```

There is an exclamation mark before the command. That is all. Let's do an example: select a cell, change its format to *Code*, and then type the following on it:

```
!ls
```

Then press *Shift+Enter* to execute the cell code and watch the result.

## Python code (ROS code too)

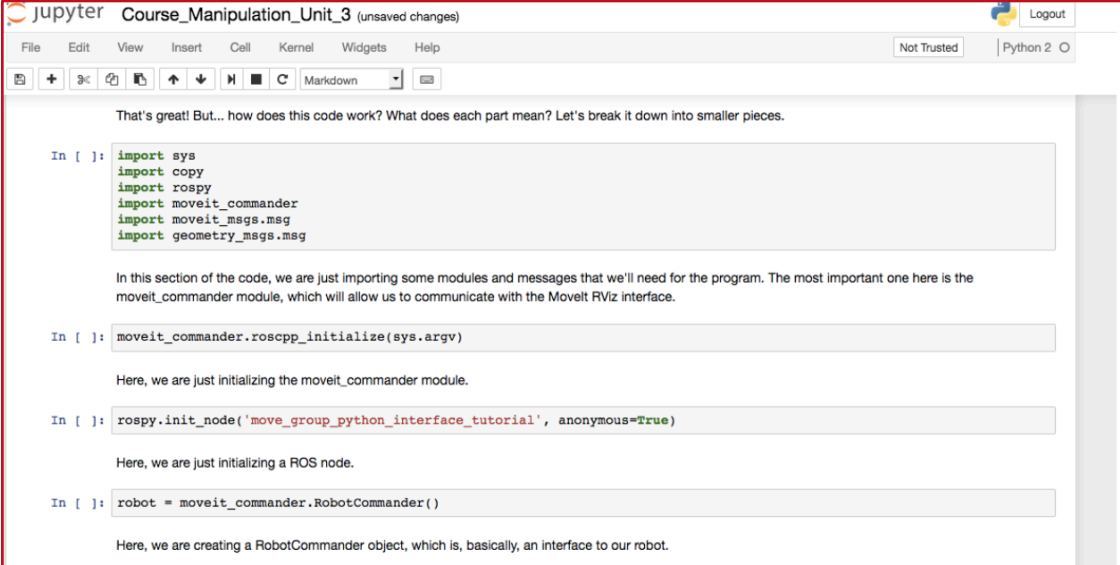
You can write Python code directly in any cell and execute it by pressing *Shift+Enter* (remember to previously select the format of the cell as *Code*). This Python code can be anything you want. It can be straight Python code, the code of a Python library you have installed in your system for machine learning or a full ROS program. It doesn't matter what you write there, provided that it is valid Python code, and that the *PYTHONPATH* of the shell where you launched the notebook has access to your libraries. If you want to use ROS you will also need that the launching shell has the *ROS\_PACKAGE\_PATH* properly set up.

Here there is an example of code of a node that subscribes to a topic. Type the code in a cell and execute it with *Shift+Enter*. While the code is running in the notebook, go to a shell in your computer and check that the */counter* topic is there (remember *roscore* must be running). Observe what happens in the notebook if you publish something in the topic through the shell.

```
#!/usr/bin/env python
import rospy
from std_msgs.msg import Int32
def callback(msg):
    print msg.data
rospy.init_node('topic_subscriber')
sub = rospy.Subscriber('/counter', Int32, callback)
rospy.spin()
```

In any notebook cell, you can include full ROS programs with node initialization, callbacks, classes and else. Just bear in mind that if you initialize a ROS node in a cell, you will have to restart the kernel of the notebook if you want to execute that code again. To restart the kernel, press the Restart button on the top menu.

You don't have to have all the code of a program in the same cell. Code can be divided into different parts, each part in a different cell. This is very useful in cases where you execute only a part of the code, then embed some explanation, then include another part of the code, and keep adding explanations. For example, imagine that you are doing a code for grasping an object. You can first put the code that identifies the position of the object. Execute only that part and then explain how it works, make the students test that part, modify it for obtaining better results, etc.



The screenshot shows a Jupyter Notebook window titled "Course\_Manipulation\_Unit\_3 (unsaved changes)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for file operations and execution, and a status bar indicating "Not Trusted" and "Python 2". The notebook content consists of several cells: a text cell explaining the purpose of the code, a code cell with import statements, a text cell explaining the imports, a code cell for initializing the moveit\_commander module, a text cell explaining the initialization, a code cell for initializing the ROS node, a text cell explaining the node initialization, a code cell for creating a RobotCommander object, and a text cell explaining the object creation.

```
In [ ]: import sys
import copy
import rospy
import moveit_commander
import moveit_msgs.msg
import geometry_msgs.msg
```

In this section of the code, we are just importing some modules and messages that we'll need for the program. The most important one here is the moveit\_commander module, which will allow us to communicate with the MoveIt RViz interface.

```
In [ ]: moveit_commander.roscpp_initialize(sys.argv)
```

Here, we are just initializing the moveit\_commander module.

```
In [ ]: rospy.init_node('move_group_python_interface_tutorial', anonymous=True)
```

Here, we are just initializing a ROS node.

```
In [ ]: robot = moveit_commander.RobotCommander()
```

Here, we are creating a RobotCommander object, which is, basically, an interface to our robot.

Next you add the part for approaching the object with the robotic arm. Then you make it pick up. Finally you make it lift the object. Every part can be done in a subroutine, and executed and analyzed separately.

This approach allows a step by step teaching where each important concept can be explained, executed, acted upon it, and seen its results, resulting in a more deep knowledge of the concept.

Your teaching should always be promoting the interaction with the code, if you want to deepen the students learning. For example, make them launch a code, then make them interact with the expected result from the shell. Then make them change the code in some way to change the results to be obtained. This whole cycle is what makes the student really learn about the subject.

### ***Embedding graphics***

You can embed pictures, videos and real-time generated plots in a notebook. Embedding that kind of stuff allows a richer interaction of the students with the subject at hands.



## Pictures

Let's see first how to embed static pictures. That is very easy to do and can be used to show a setup, an expected result, a graphic providing necessary data, or just an example of what you expect them to achieve.

To embed a picture in the notebook, select a cell, then change its format to *Markdown*. Finally, introduce the following code in the cell:

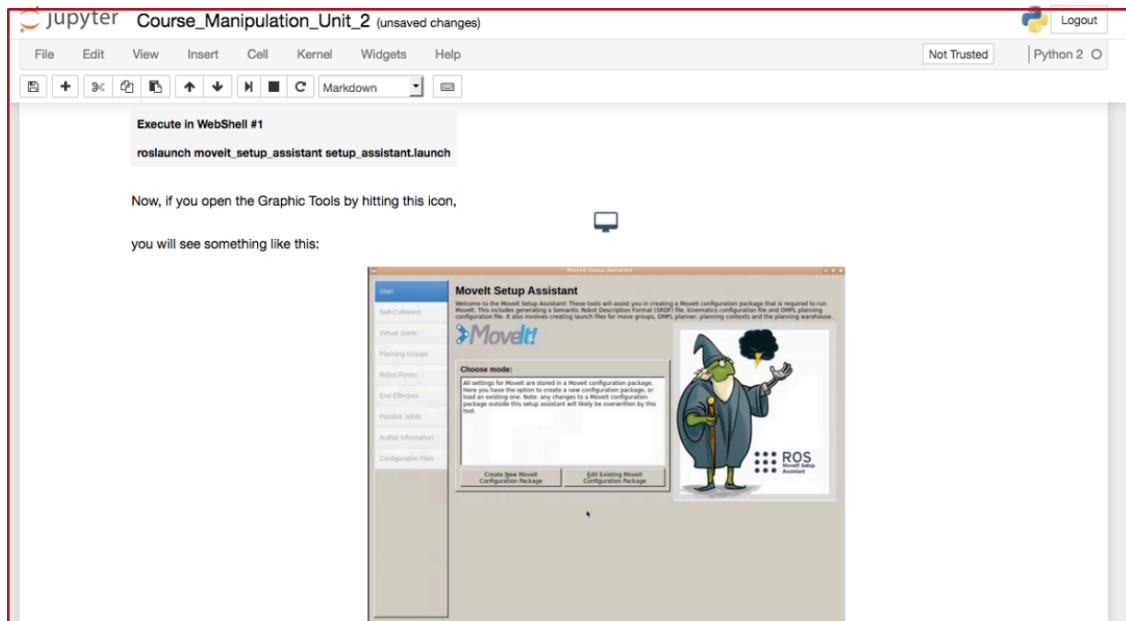
```

```

In the previous code, the *src* contains the name of your image file. That is the path to the file from the location of the notebook. We recommend to create an *img* directory besides the notebook and put all the pictures there (typical organization for a web page).

Since the code for images is HTML code, you can even embed images that are stored in the web, so you don't have to place them with your notebook. If you use this option, remember that you will need access to internet when the notebook is open, in order to be able to show the images.

Additionally to including images, you can use the same method to **include animated gifs**. We use this resource a lot in order to embed a small **gif animation that shows the expected result of the exercise**. So, without providing the solution, you are indicating the expected result to the student.



## Videos

You can embed YouTube or Vimeo videos in the notebooks in order to include a richer experience. In order to include a video, first you must declare the cell as a Code cell. Then you will need the following code:

```
from IPython.display import HTML
# Youtube
HTML('the_iframe_code_that_Youtube_provides')
# Vimeo
HTML('the_iframe_code_that_Youtube_provides')
```

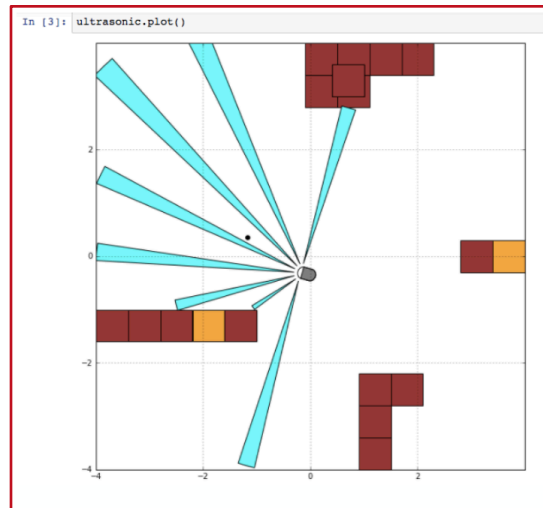
For example, let's try the following code to embed a YouTube video:

```
from IPython.display import HTML
HTML('<iframe width="560" height="315"
src="https://www.youtube.com/embed/udHlvH6TGvo"
frameborder="0"allowfullscreen></iframe>')
```

## Real time generated plots

Now, all those figures that you have included in the notebook so far are just static figures, created some time ago, and they do not change based on the data available.

You can generate figures based on the current data being generated by your program or robot.



For example, you can make the robot read the data from a topic, process it, and then present the result to the user, everything without creating or launching a ROS package (all the code embedded in the notebook).

For generating a graphic, first you need to capture the data points, for example subscribing to a topic. Then you can process it and finally, present the result on a plot.

For example, let's use an array of data and plot it. Now type the following code:

```
import matplotlib.pyplot as plt
%matplotlib inline
data=[1,1,2,3,5,8,13,21]
plt.figure()
plt.plot(data)
```

If you type that code in a notebook and execute it you will get a plot as a result.

Now, you can apply the same to the capture of a topic. You can subscribe to a topic, capture its data in the callback, and then plot it on the screen. Topics information can include laser points, point clouds, images... whatever For all that you only need to use *matplotlib*. If you want more fancy plots, I would recommend you to use [Plotly](#).



## Embedding controls

Apart from embedding graphics and code, you can also embed controls that allow you to interact with the robots. For example, you can embed a control that allows you to modify the speed of the robot or its direction, or make a joint move.

In order to embed controllers we are going to use *iPywidgets*. There are many types of interactive widgets. You can explore them in [this link](#). Here we are just going to explore a single one just to introduce you to the power of them for creating interactive notebooks for robotics. Copy the following code in a notebook and execute it.

```
from ipywidgets import interact, interactive, fixed,
interact_manual
import ipywidgets as widgets
def f(x):
    return x
interact(f, x=10);
```

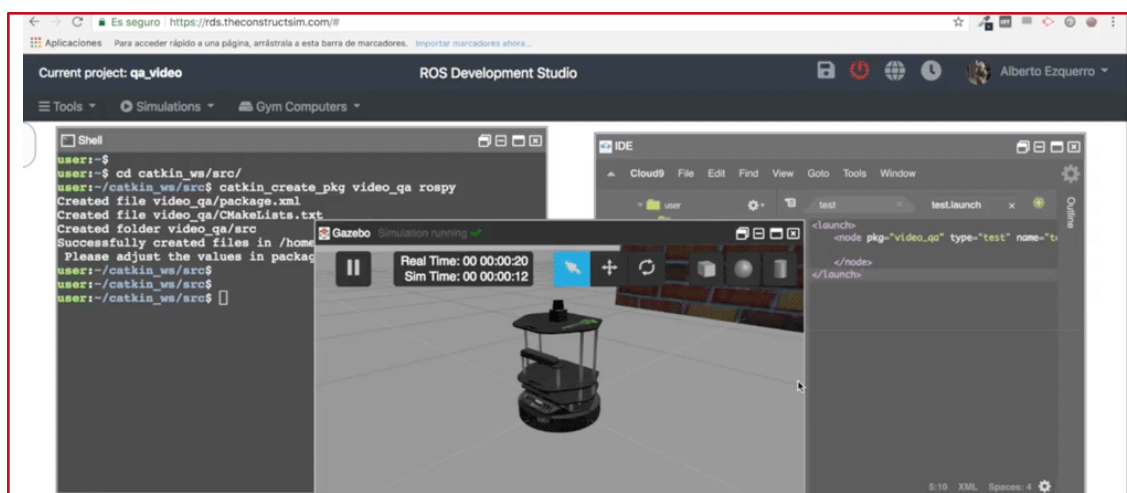
That code, just expresses how a slide can modify a value. Integrated with a ROS code you can make the slide vary the speed of the robot or the position of a joint, to name a couple of examples. You can make the student interact with the notebook and the robot without having to program, speeding up the comprehension of what they are going to code.

## Final comments about notebooks

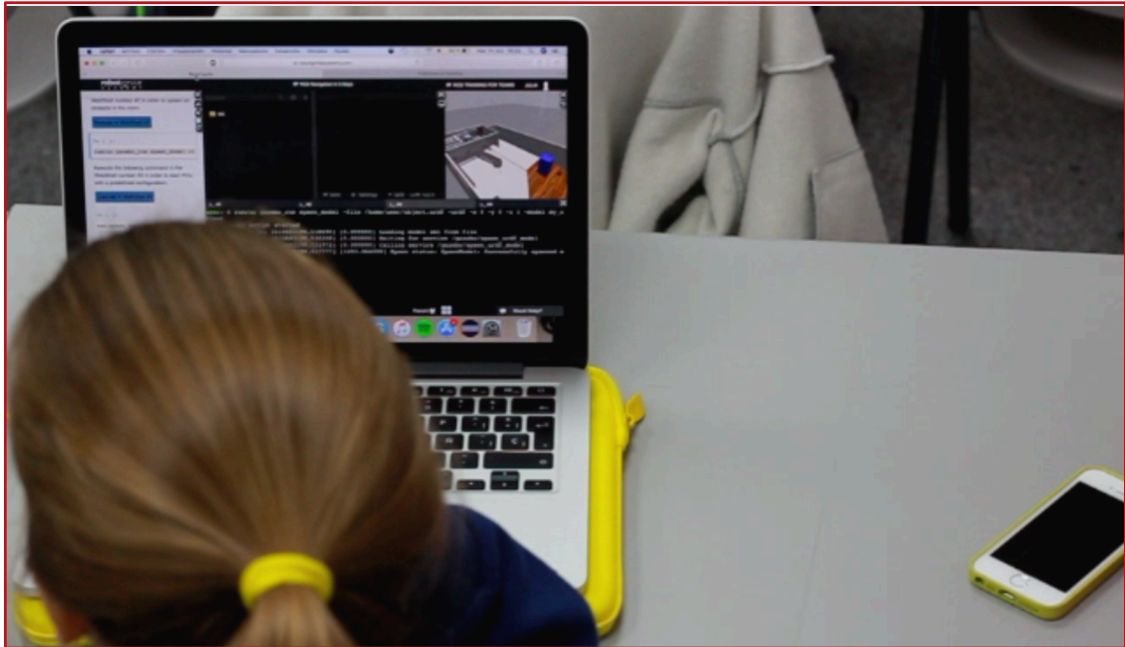
You can learn a lot more about what you can do with notebooks by looking at [this video](#).

Also we recommend you the [following post](#) for an example on how to structure a notebook for an interactive teaching of Linear Regressions (combining almost everything you learnt above).

Remember that at the end, what will make the learning of the student fast is the fact that you are discarding a lot of information, and that you only include in the notebooks the subjects that are really required to build the programs that make the robots do things. If you use the same approach as in the slides, trying to explain the *whole ROS Universe*, then the notebook will have no advantage to the slides.



## Creating the simulations



The notebooks alone are a super good improvement from just slides, because they contain a lot of more interactive information that can even adapt to the current situation.

However, the biggest impact in learning speed arrives when the student can do things and see the results of their actions in real time on a robot<sup>1</sup>. For this purpose, we are going to attach simulations of robots to the notebook experience.

---

<sup>1</sup> See section about the theory to understand why

By attaching simulations of robots, the students will be able to program and test on a robot while following the notebook. Since the simulated robot will provide the same exact interface than the real robot (because they are based on ROS), the student will be actually doing a real life learning. All the things that the student programs on the simulation would have the same effect on the real robot. This has two implications:

- 1.What you are teaching is real code that can be used straight in real robots. Those are no toy examples. Hence your students will be more prepared to the market.
- 2.If you have at least one real robot, they will be able to test their programs in the real one. Actually this is how real robotics companies work (they first program in simulation and when working, they test in real robot).

We recommend to use the Gazebo simulator since it is shipped by default with ROS and is very well integrated with it.

Hence, depending on your subject, you will have to create a simulation according to that subject. For example, if you are teaching

object recognition, you will have to create a simulation with a robot that has a device for capturing the environment (usually a camera or depth sensor).

You can find many simulations already working on the internet. So it is very likely that you will find a simulation that already works for your class. Otherwise, you will have to build your own. If that is the case, check the excellent tutorials about it at the gazebo website ([www.gazebo.org](http://www.gazebo.org)).

For instance, our company ([The Construct](#)) uses many simulations for the courses we create for the Robot Ignite Academy. In order to support the Open Source community in robotics, we are providing an open repository to all those simulations. You can check our simulations repo here.

## Organizing the syllabus of the course

Now that you have all the elements for building your notebooks, you must decide the content of them and organize them into several parts that will rise to a complete course.



We have tested many ways of organizing the courses in the following parts for maximizing learning speed, and we found that the following organization works most of the times.

## Units

Divide all the course in separated **Units**. Units are the chapters of a course. Each Unit must be about a single topic that you want the students to learn (for example topic publishers, action clients, debugging tools). Include in that Unit only the explanations, examples and exercises related to that topic.

## Practices

Organize each Unit around a series of **practices** that the student must do. Avoid as much as possible the descriptive parts of the subject. For example, do not start describing what a topic is, but instead, start by indicating that we need to move the robot wheel and that for that, we need to use something that is called a topic. Then show how to use a topic for making the robot move. Make the robot move and only then, you can,

if needed, define what a topic is (and at that point nobody would care!).

## Project

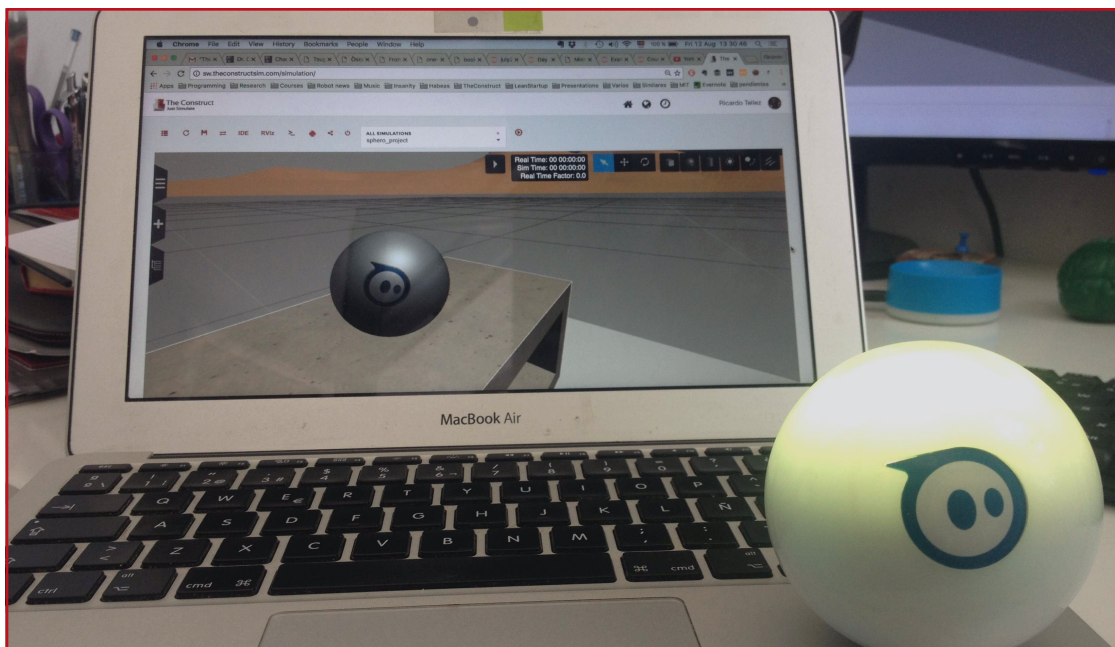
Create a **Project**. Even if along the teaching lessons there are a lot of exercises that need to be done on the spot, the Academy contains also an additional activity that is called a project. A project is some programming exercise that the student has to do from beginning to end, which requires the student to use all the parts he is learning during the whole course.

You should allow the students to work on the project every time they finish a Unit. Actually the project has to contain as many parts as Units has the course. Each part of the project will focus on applying what has been learnt in the corresponding Unit.

However, the project has to be seen by the student as a whole thing. I mean, the project is not just a series of exercises like the ones included in each Unit. The project aims at making a robot do something interesting for the student (and remember that, since you have access to the simulation of any robot, you can provide the most interesting things to the student). In order to make the robot do that

interesting thing, he will have to apply the knowledge of all the Units together.

Projects are done in the simulation too. However, the simulations must be of real life robots with a known ROS interface. This means that, what the students do for the project simulation, can be used exactly like that on the real robot.



If you have the chance, it would be very interesting to get the real robot and allow the students to transfer their project programs to the real robot and see how it behaves. We have done it with Sphero

robot, and, even if it was a little stressing for the teacher, the students liked a lot to see how their programs on the screen really applied to the real robots.

Also, a new set of problems appear, that provide the student the possibility to know about the differences between using simulations and using real robots.

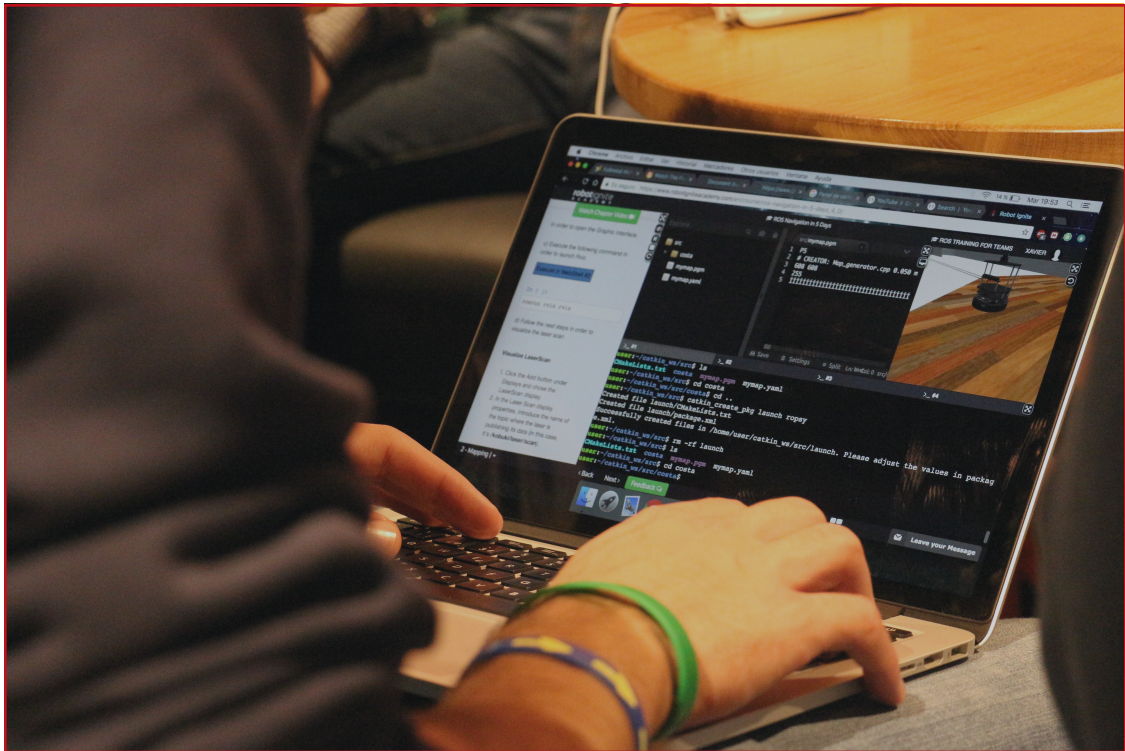
## Exam

You will need to create an **Exam**. No good teaching program is complete without doing an exam!

The exam requires the student to write a ROS set of nodes that allow the robot to do something. The exam has exactly the same structure as the project, but with a different robot, environment, and for a different task. Apart from that, the structure of the exam is the same as in the project, in the sense that, if the student has completely understood the project, he must be able to do the exam.



The exam must be activated on demand to prevent people watching it prior to the exam. We allocate 2 hours to do the exam. During the exam, students can review the lessons they have done, the programs they created, and even documentation online. The goal here is not that they remember everything but that they really understood how to create such types of programs and where the relevant information can be found.





ROS

3

## CHAPTER THREE

***Available online  
platforms***



The  
Construct

Learn and develop for robots with ROS

# Available online platforms

## PART 3

Of course, preparing the notebooks, testing the simulations, having the exams ready, all that is a lot of work!!

OK now, you as a teacher, you are faced with a decision: having known now that there is a more effective and fulfilling method to teach ROS, would you still want to go back to your slides method?

Be sincere here. **What are we talking about?**

In this document, we are talking about nurturing better students, making them grow faster. That is our job as teachers/instructors. If you are also concerned about it, then you may agree that the slides system is not the best way to do it. It is comfortable for us as instructors, yes. But it pales in teaching/learning experience when compared to a notebook+simulation based system.

Even if convinced of the superiority of the NOSI system you may still worry yourself: do I have to prepare all that material?

Well **you have three options:**

1. **You prepare everything**, from simulation to notebook to the environment in the students computers. Use Jupyter Notebooks, Gazebo simulations and go for it. You can prepare Virtual Machines with all that content so the students only have to execute the VM to have everything running... Good luck! You are going to need it!
2. **You only prepare the notebook content** and use an online system for providing the environment and robot simulations to the students: use the [ROS Development Studio](#).
3. **You rely everything on an online platform** and you just dedicate to teach and support your students: use the [Robot Ignite Academy](#).





## ROS Development Studio (RDS)

The RDS is an online platform that provides a full development environment for ROS programs using only a web browser. You can use any type of computer to program with ROS: Mac, Linux, or even Windows so neither the students, nor the teachers are required to install Ubuntu on the computers and then ROS on top. The RDS is mainly used for programming ROS based robots, but it can also be used to teach ROS (like for example *A.G. Mundet High School* or *FH Aachen University of Applied Sciences* have done).

Students only need to use a web browser like Chrome or Firefox to program the ROS based robots. And what is more interesting, they can do it from anywhere.



The advantages of using this platform for your ROS class are the following:

- 1.The RDS provides off the shelf many robot simulations that can be launched with a single click. So you can use those simulations for your courses.
- 2.It provides a full ROS+Gazebo development environment on any computer with a web browser, off-the-shelf. So you do not have to install anything in the students computers and you know that what you build there will work for all students.
- 3.It provides an integrated Jupyter Notebook environment that you can use to create your own notebooks associated to the robot simulations. You can then, share the notebook with your students and they will be able to open it and follow your lessons.

Basically, the RDS provides you with all that you need to create your NOSI courses online, and then share them with the students. You will only need to focus on creating the notebooks at your style.



## Robot Ignite Academy

This Academy, is an online platform that provides the teacher with the full package. ROS development environment, Gazebo simulations and complete list of ROS courses completely ready and fully operational from any web browser. All the courses provided in the Academy have been created using the philosophy behind this document, with the main aim of teaching ROS fast.

**By using this platform, students and teachers will have to install and prepare nothing to learn or teach ROS.**

All the courses provided at the Academy follow the same structure: on the left hand side of the course there is the notebook that describes step by step how to learn the subject. For example, for ROS TF 101, it teaches what is tf and how to use in robots. Then the students use the development environment (located in the middle of the screen) to program and launch tests. The results can be seen on the simulated robot on the left.

You as a teacher may be thinking: if everything is already provided, what I am going to do during the classes? Well, you have two options here:

1. **you let the students follow the tutorial by themselves** and only interfere in their learning when some student has a problem understanding something. Then you can explain the problem to the whole class (very likely many of them will have the same problem). This is the approach used by University of Weingarten.
2. **you and the students follow the notebook together**, and you use the content as a guide for your explanation to the students.

“

*I have learnt more in this platform in 3 hours than I have in several months moving through Internet*

**Damian Russo** | User of Robot Ignite Academy

”

This is the approach we follow in our live courses and at LaSalle University, Barcelona where we teach ROS, because it is clearer for the students and also leads to debates and interesting questions.

It is up to you. Both approaches work.

The thing is that while following the tutorial, every few steps the student will have to do an exercise, a small task or something related to the simulated robot on the right top corner. You as the teacher, you must let the student do that part on his own since it is this the part that provides the learning to the student. Just be there ready to answer questions about it and to provide clues about how to solve. Let the student try new things based on your clues but not to provide to him how to solve the exercise.

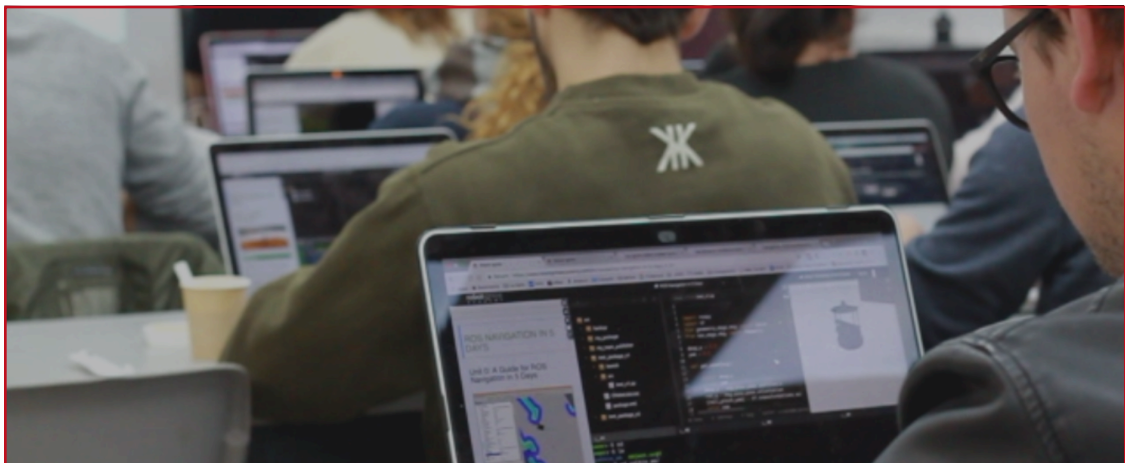
At this point students will have a lot of errors. That is normal. Just let them have it, try to solve them, and then let them ask you what is happening. Then give them some hints in order to them solve the problem.

# CONCLUSION

## *Teach ROS Effectively*

Now you know how to build ROS courses that make the students learn such a difficult framework in record time. We have taught this method many times and certified that it works(specially if the teacher not only provides the notebooks to the students but also supports them with additional explanations to problems).

As a success example, we would like to mention the case of Jose and Salva, two students of Institute La Guineueta who were able to **build a ROS based product in only three months starting from zero knowledge of ROS**. You can read their story [here](#).



TEACH ROS EFFECTIVELY

# *Get a Free Demo*

See for yourself why 500+ universities from 20 countries trust Robot Ignite Academy to teach ROS effectively.

**Contact us**

[courses@robotigniteacademy.com](mailto:courses@robotigniteacademy.com)



The  
**Construct**

Learn and develop for robots with ROS