

The
Construct
Learn and develop for robots

W E B I N A R

HOW TO TEACH ROS EFFECTIVELY WITH MINIMAL PREPARATION

Ricardo Téllez | CEO of The Construct

October 17th, 2019

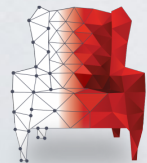
WWW.THECONSTRUCT.AI

Y
O
U

W
I
LL

LE
A
R
N

- How to have a complete ROS course based on practice that makes students learn
- How to fully remove installation problems on Windows/Linux/Mac
- How to avoid problems of the students low level on programming



MY GOAL

- The ONLY WAY to make students learn ROS is by **making them practice**
- The BEST WAY to make them practice is by using a **cloud robotics platform**



S
P
E
A
K
E
R

Ricardo Téllez, PhD
CEO of The Construct

“For ROS Developers”

 [linkedin.com/in/ricardotellez](https://www.linkedin.com/in/ricardotellez)





theconstruct.ai

C++

Ready for Robotics

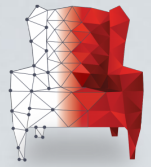
ROS
Developers
LIVE CLASS

#61

The
Construct

Every Tuesday at 18:00 CEST
On our Youtube Channel





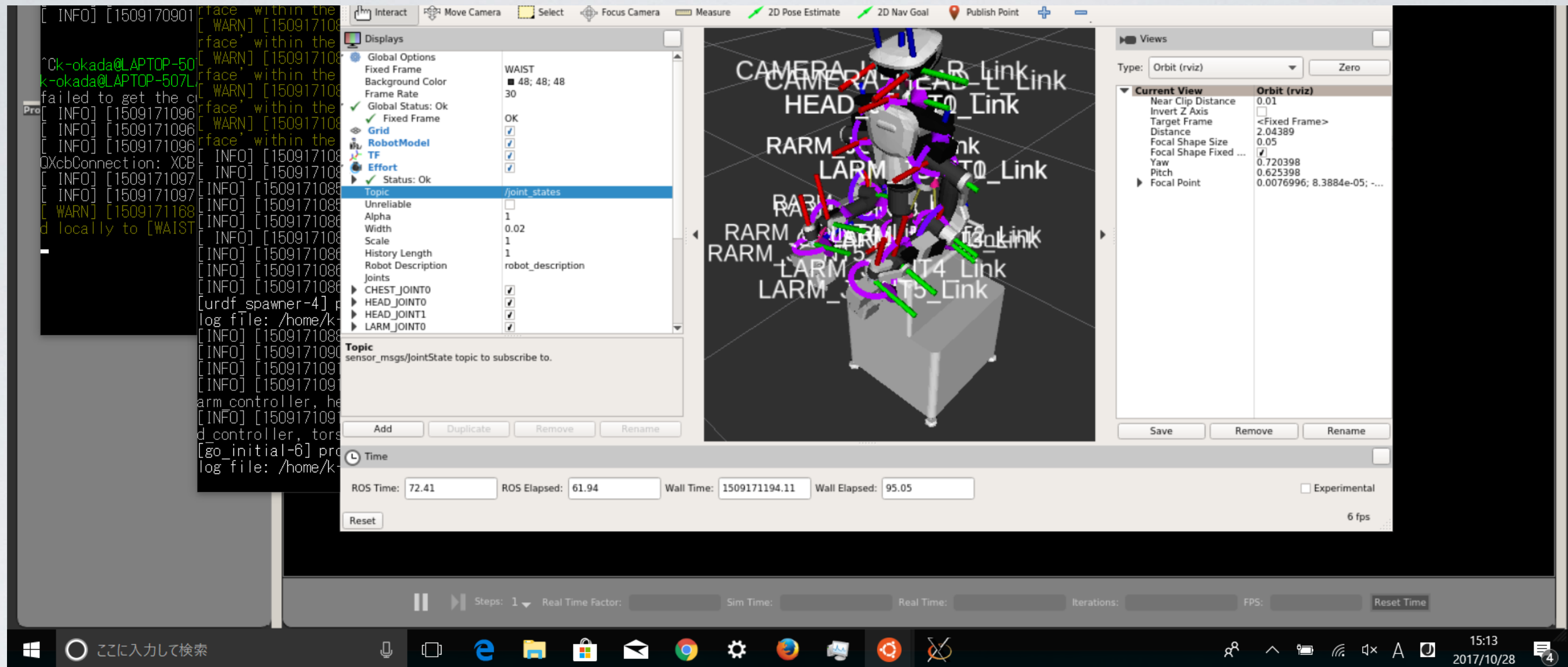
The
Construct
Learn and develop for robots

THE PAIN **TEACHING ROS TO STUDENTS**



PROBLEM #1

STUDENTS DON'T KNOW LINUX NOR PYTHON /C++



PROBLEM #2

INSTALL ROS IN STUDENTS' COMPUTERS IS DIFFICULT

ROS Navigation

Course Project

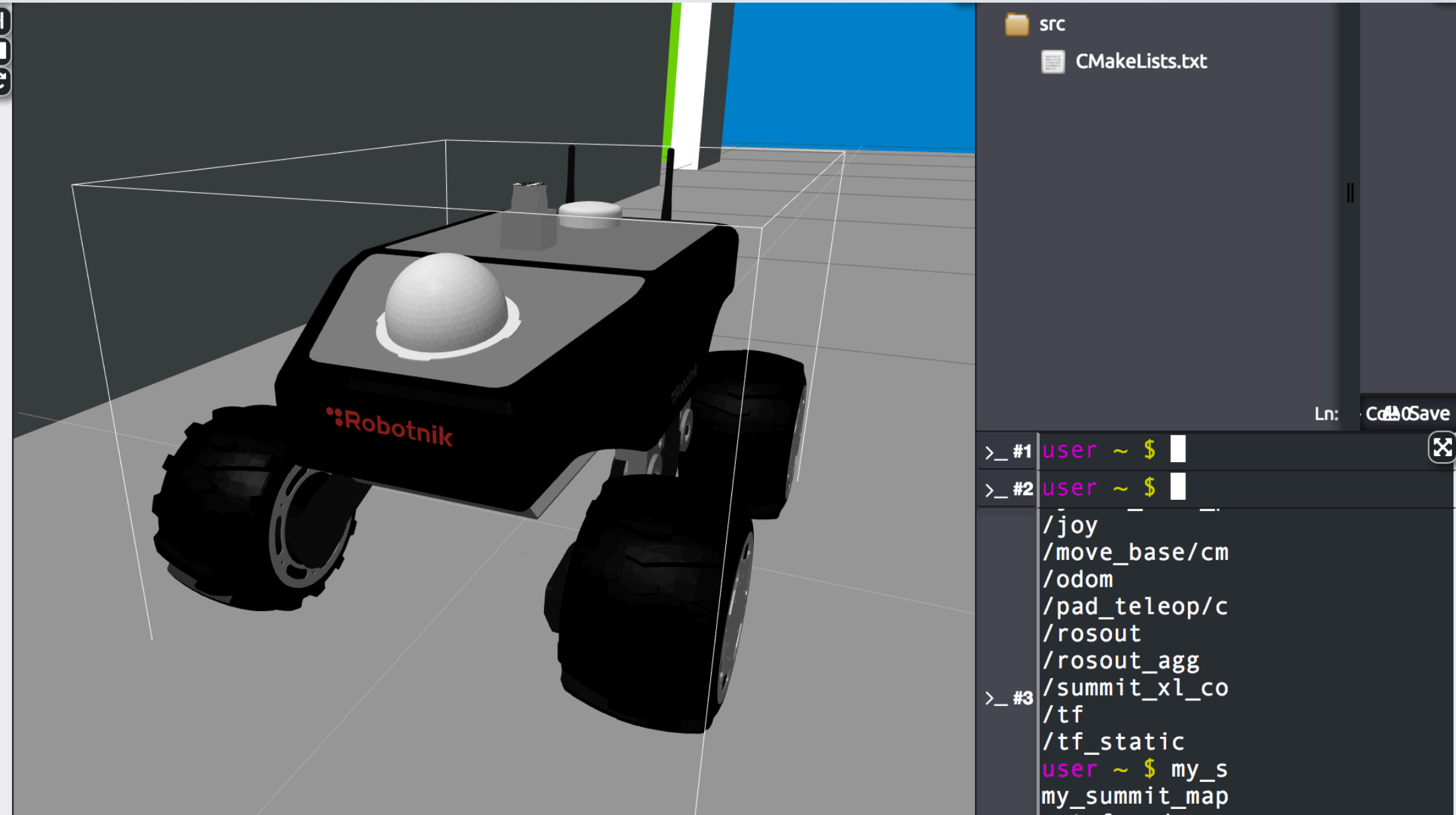


SUMMARY

Estimated time of completion: **1'5 hours**

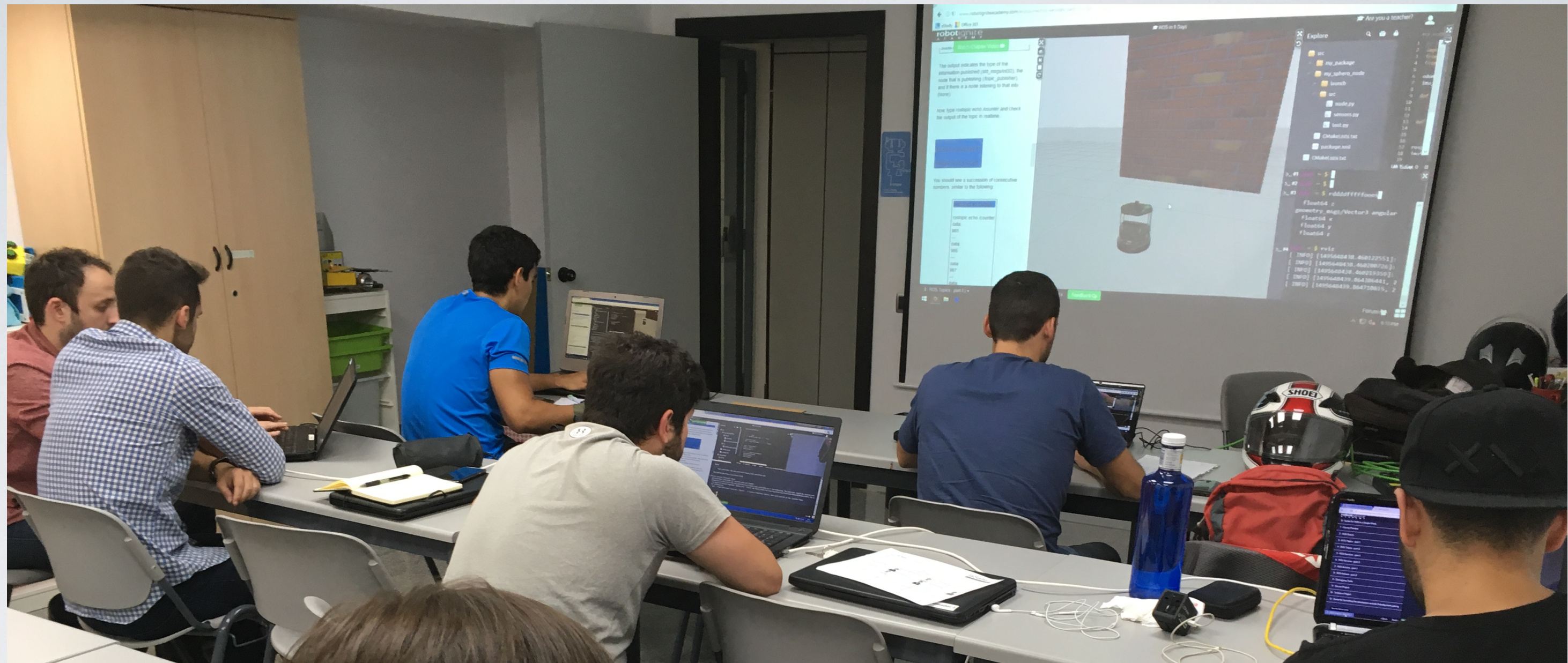
What you will learn with this unit?

- Practice everything you learn through the course
- Put together everything you learn into a big project
- Create a launch file that launches each part of the Navigation



PROBLEM #3

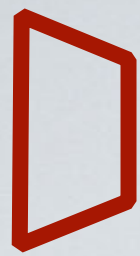
BUILD THE CURRICULUM IS A LOT OF WORK



PROBLEM #4

SHARE YOUR CODE WITH STUDENTS DOESN'T WORK

Step By Step Guide To
BUILD A ROS CURRICULUM
(THAT MAKES STUDENTS LEARN)



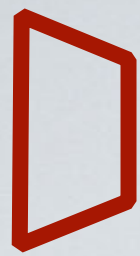
I wanted students to learn ROS by making them practice

This means, install ROS in student's computer:

- Direct ROS install
- Virtual Machine install
- Docker install

```
pi@raspberrypi: ~  
File Edit View Search Terminal Help  
pi@raspberrypi:~$ export ROS_MASTER_URI=http://192.168.1.106:11311  
pi@raspberrypi:~$ export ROS_IP=192.168.1.104  
pi@raspberrypi:~$ roslaunch rplidar_ros rplidar.launch
```

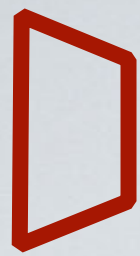
```
roscore http://ubuntu-computer:11311/  
File Edit View Search Terminal Help  
ubuntu@ubuntu-computer:~$ roscore  
... logging to /home/ubuntu/.ros/log/b03de02a-b408-11e9-8e4f-94de80023db3/roslau  
nch-ubuntu-computer-2275.log  
Checking log directory for disk usage. This may take awhile.  
Press Ctrl-C to interrupt  
Done checking log file disk usage. Usage is <1GB.  
  
started roslaunch server http://ubuntu-computer:44421/  
ros_comm version 1.14.3  
  
SUMMARY  
=====  
  
PARAMETERS  
* /roscdistro: melodic  
  
ubuntu@ubuntu-computer: ~  
File Edit View Search Terminal Help  
ubuntu@ubuntu-computer:~$ export ROS_MASTER_URI=http://192.168.1.106:11311  
ubuntu@ubuntu-computer:~$ export ROS_IP=192.168.1.106  
ubuntu@ubuntu-computer:~$ rviz
```

How to provide a ROS environment to the students that have?

- **No knowledge of Linux**
- **Low programming knowledge** (Python or C++)
- **Windows machines** (mainly)





STEP 0: SETTING UP THE ENVIRONMENT

Let's use the cloud to avoid all the setup trouble!

I'm going to show two solutions:

1. One where **you build the curriculum** (it is free)
2. Another with the **full curriculum built** (has some cost)



The
Construct
Learn and develop for robots

The screenshot shows the ROS Development Studio interface. On the left is a file explorer showing a project structure for a turtlebot simulation, including folders like 'simulation_ws', 'src', 'turtlebot3', and 'gazebo_ros_turtlebot3'. The main window is a code editor displaying the file 'zebo_ros_turtlebot3.cpp'. The code is in C++ and shows the initialization function for a Gazebo ROS TurtleBot3. It includes comments and code for initializing ROS parameters, setting robot model-specific parameters (like turning radius and distance limits for 'burger' and 'waffle' models), and initializing variables and publishers/subscribers for laser scan data.

```
35- // *****
36- * Init function
37- *****
38- bool GazeboRosTurtleBot3::init()
39- {
40-     // initialize ROS parameter
41-     nh_.param("is_debug", is_debug_, is_debug_);
42-     std::string robot_model = nh_.param<std::string>("tb3_model", "");
43-
44-     if (!robot_model.compare("burger"))
45-     {
46-         turning_radius_ = 0.08;
47-         rotate_angle_ = 50.0 * DEG2RAD;
48-         front_distance_limit_ = 0.7;
49-         side_distance_limit_ = 0.4;
50-     }
51-     else if (!robot_model.compare("waffle"))
52-     {
53-         turning_radius_ = 0.1435;
54-         rotate_angle_ = 40.0 * DEG2RAD;
55-         front_distance_limit_ = 0.7;
56-         side_distance_limit_ = 0.6;
57-     }
58-     ROS_INFO("robot_model : %s", robot_model.c_str());
59-     ROS_INFO("turning_radius_ : %lf", turning_radius_);
60-     ROS_INFO("front distance limit_ = %lf", front_distance_limit_);
61-     ROS_INFO("side distance limit_ = %lf", side_distance_limit_);
62-
63-     // initialize variables
64-     right_joint_encoder_ = 0.0;
65-     priv_right_joint_encoder_ = 0.0;
66-     // initialize publishers
67-     cmd_vel_pub_ = nh_.advertise<geometry_msgs::Twist>("/cmd_vel", 10);
68-
69-     // initialize subscribers
70-     laser_scan_sub_ = nh_.subscribe("/scan", 10, &GazeboRosTurtleBot3::laserScanMsgCallBack, this);
```

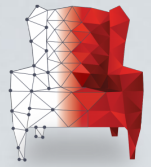
You also need to know how we represent the movement of the robot. For that we use **frames**. The frames are the axis of reference from which we will measure the position of the robot. The frame for the odometry in ROS is called `/odom`

The position of the robot in a frame of reference is called **the pose** of the robot. The pose of the robot contains the coordinates x,y,z in that frame, as well as the rotation R,P,Y.

OUR METHOD

- Many Universities around the world using it:
 - Clarkson University, **USA**
 - University Reims, **France**
 - Tokyo University, **Japan**
 - University of Sydney, **Australia**
 - University of Luxembourg, **Luxembourg**
 - University of Michigan, **USA**
 - Heriot Watt, **Scotland**
 - FH Aachen, **Germany**
 - La Salle Barcelona, **Spain**





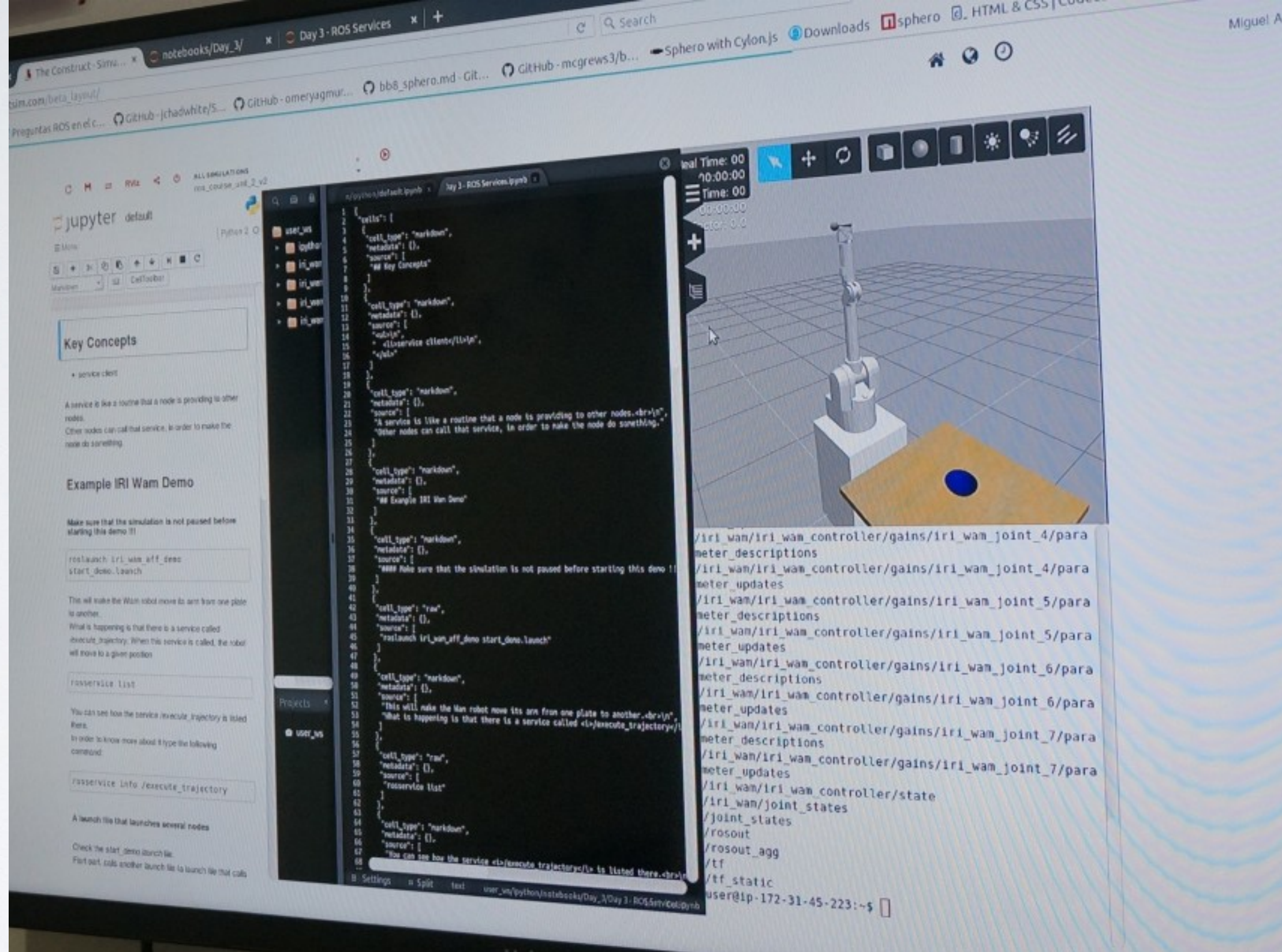
The
Construct
Learn and develop for robots

USING THE CLOUD TO **BUILD THE CURRICULUM**

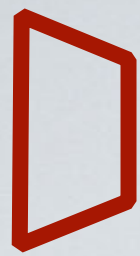
Completely Free solution

ROSDS

ROS DEVELOPMENT STUDIO

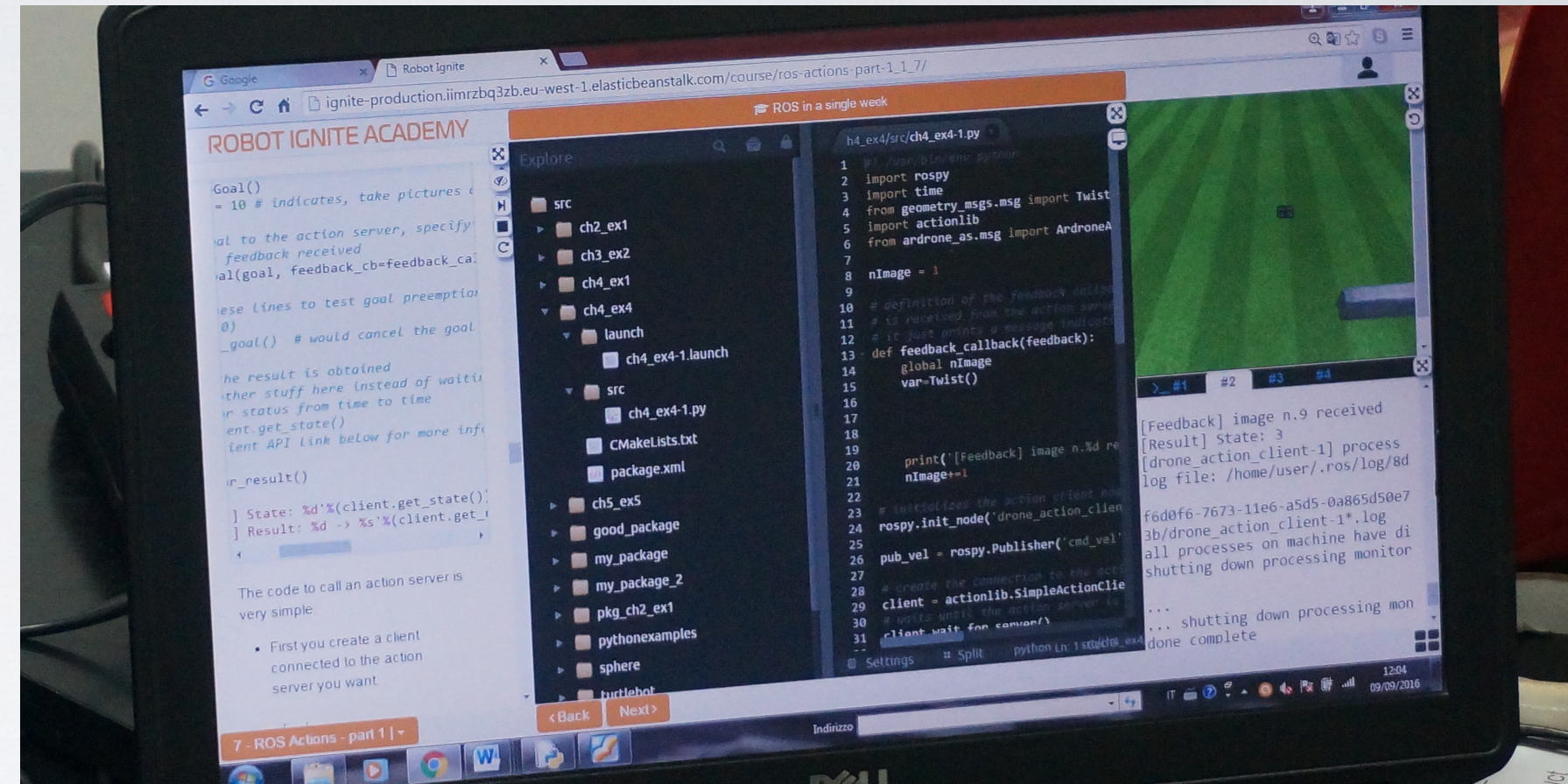


Create a free account at the ROSDS by visiting:
<http://rosds.online>



STEP 1: Decide Subject To Teach

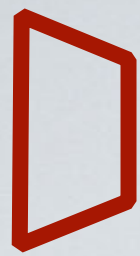
- Decide a ROS Distribution
- Decide a robotics subject
- Decide a programming language



Let's work on **ROS Kinetic**

Let's create a course about **Robot Navigation with ROS**

Let's use **Python** for programming



STEP 2: Decide Units Of The Course

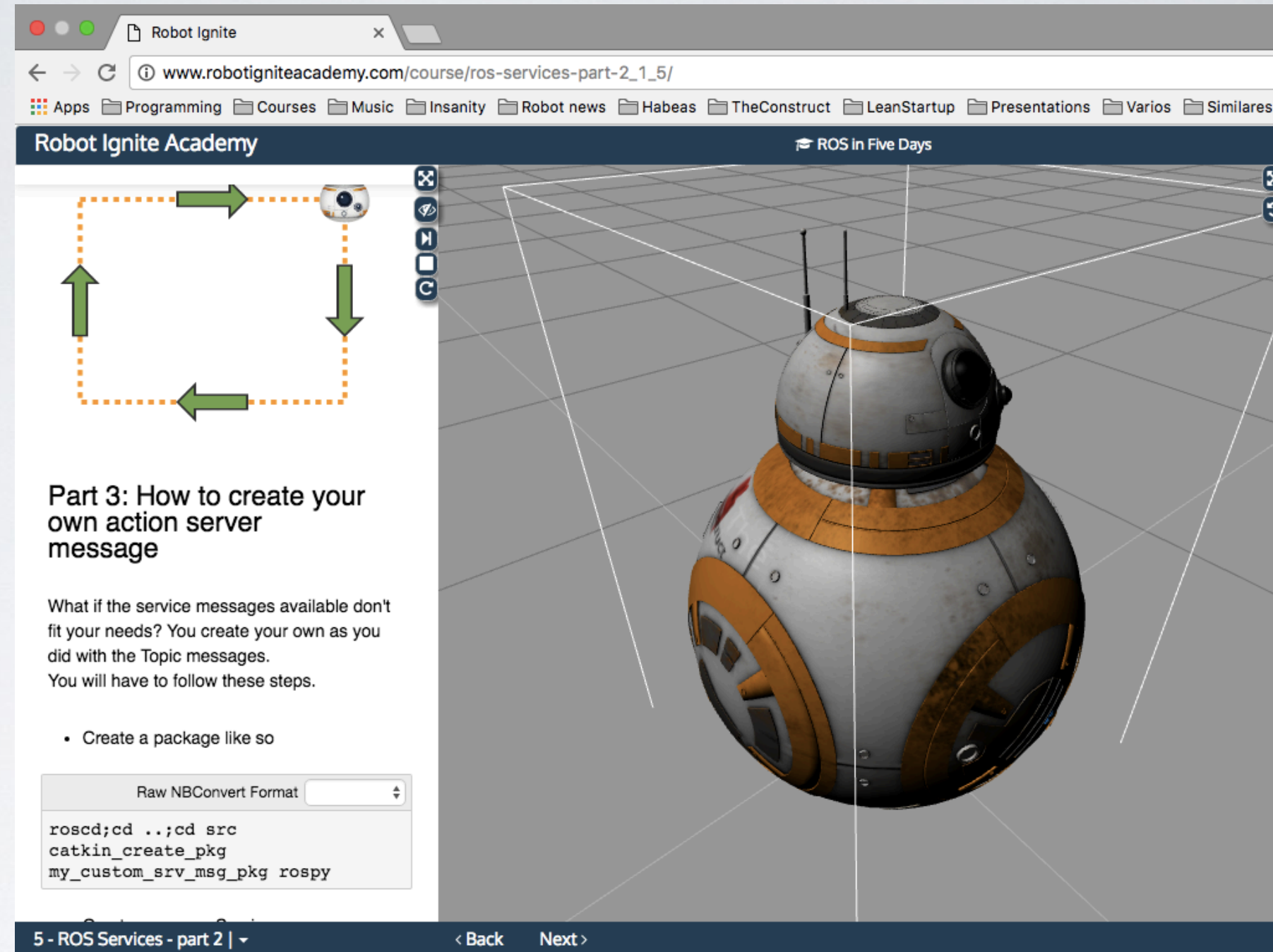
Six Units for Robot Navigation:

- **Unit 1:** Odometry Based Navigation
- **Unit 2:** Sensors For Robot Navigation
- **Unit 3:** SLAM Map Building
- **Unit 4:** Monte Carlo Localization
- **Unit 5:** Rapid Random Trees Path Plan
- **Unit 6:** Dynamic Window Approach
- **Project:** Patrol Robot



R
D
S

<http://rosds.online>



The screenshot shows a web browser window with the URL `www.robotigniteacademy.com/course/ros-services-part-2_1_5/`. The page is titled "Robot Ignite Academy" and features a "ROS In Five Days" badge. On the left, there is a diagram of a square path with a robot icon at the top right corner, indicating a navigation task. Below the diagram, the text reads: "Part 3: How to create your own action server message". It explains that if existing service messages don't fit, one can create their own, following specific steps. A bullet point states: "Create a package like so". Below this, a code block shows the following commands:

```
roscd; cd ..; cd src
catkin_create_pkg
my_custom_srv_msg_pkg rospy
```

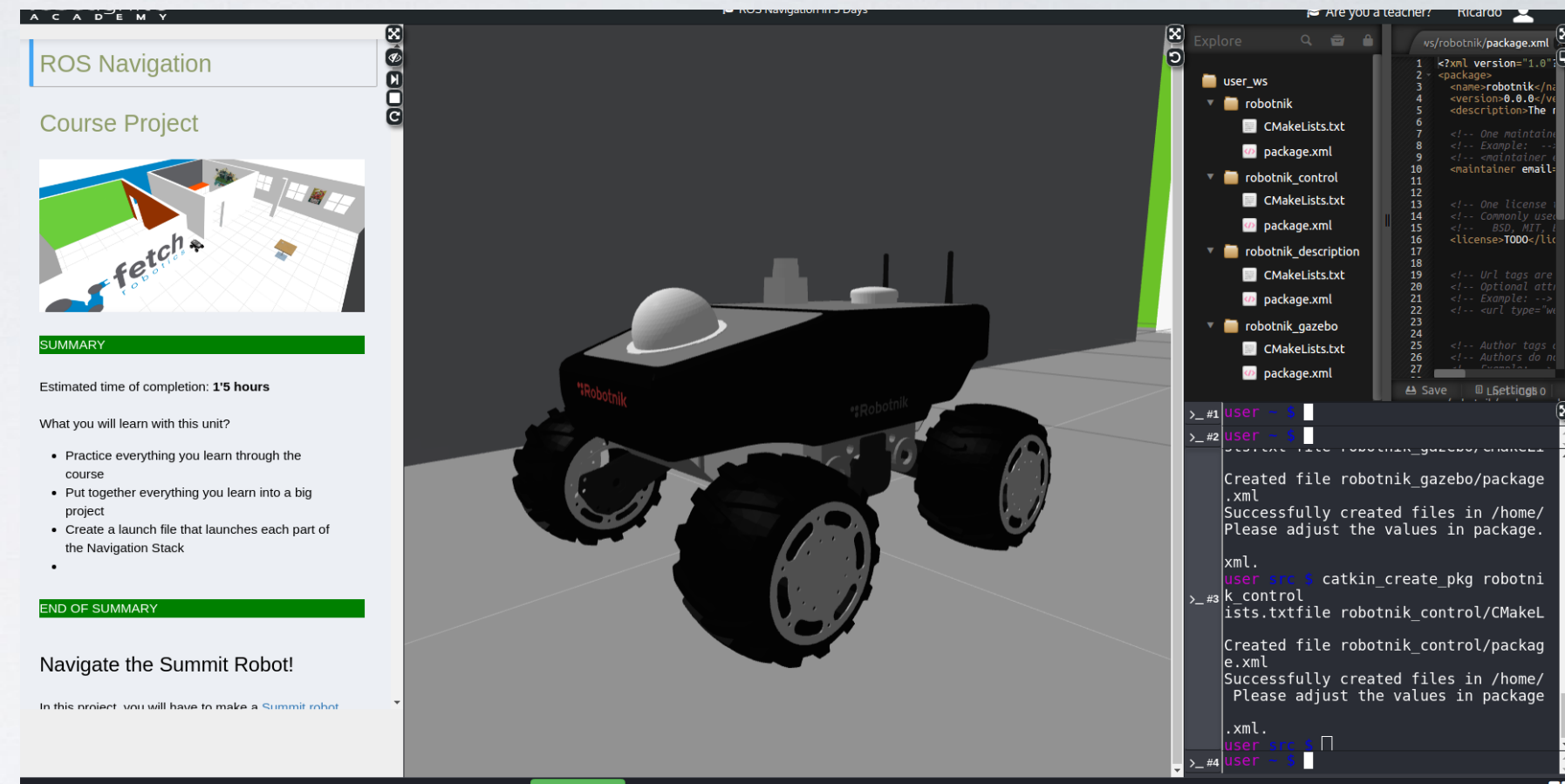
 The right side of the page displays a 3D model of a BB-8 robot in a virtual environment. At the bottom, navigation links for "5 - ROS Services - part 2" and "Back/Next" are visible.

Create a ROS project for first Unit
Unit Title: Unit I_Odometry_Based_Navigation



STEP 3: Decide Which Robots To Use

- Unit 1: **ROSBot** by Husarion
- Unit 2: **Husky** by Clearpath
- Unit 3: **Turtlebot 2** by OSRF
- Unit 4: **Jackal** by Clearpath
- Unit 5: **Summit XL** by Robotnik
- Unit 6: **RB-I** by Robotnik
- Project: **Turtlebot 2** by OSRF

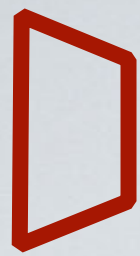


- Need to use robot simulations to practice
- Selected robots must be suitable for the subject to teach
- Get simulations from repos or included in ROSDS
- The Construct simulations repo (for Kinetic or Melodic):

<https://bit.ly/2Gp60lm>



The
Construct
Learn and develop for robots



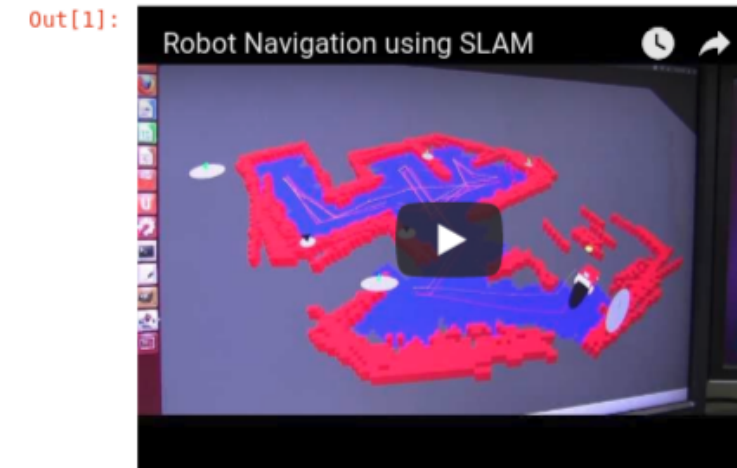
STEP 4: Create The Notebook

Using Jupyter Notebook

- Include text explanations
- Embed videos and pictures
- Embed Python code
- Interact with the robot directly from the notebook

What is this unit about? This unit is about making a robot move around by sending velocity commands to its wheels and by using odometry to figure out where in the space the robot is. Watch the following video to understand what all that means (select the next cell and press the *Run Cell* button to load the video).

```
In [1]: from IPython.display import YouTubeVideo
# an example of odometry based navigation
YouTubeVideo('SeNLUW79_-c')
```

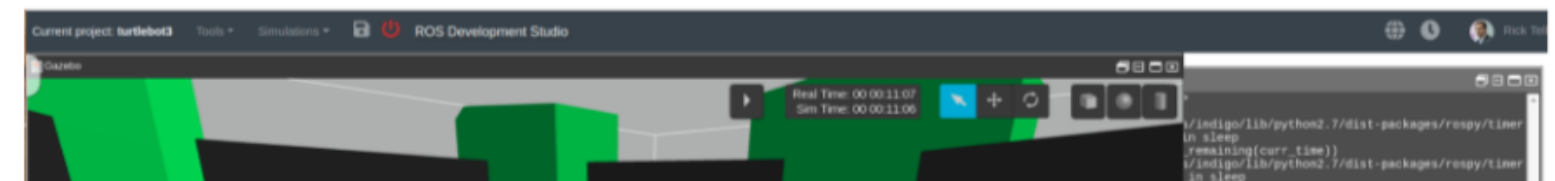


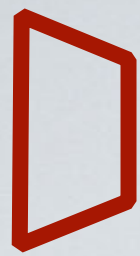
By the end of this unit you will be able to:

- Send commands to the wheels of a ROS based robot
- Understand what odometry is and how to obtain it from a ROS based robot
- Move a ROS based robot around using *Dead Reckoning* (odometry + wheel commands)

Simulation

- Go to the Simulations menu and select the *Launch simulation*
- Select the launch file *turtlebot3_world.launch* from package *turtlebot3_gazebo* to launch the simulation
- Press *launch* to launch it. You should see the following image appear

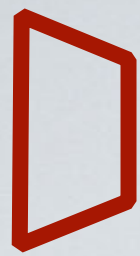




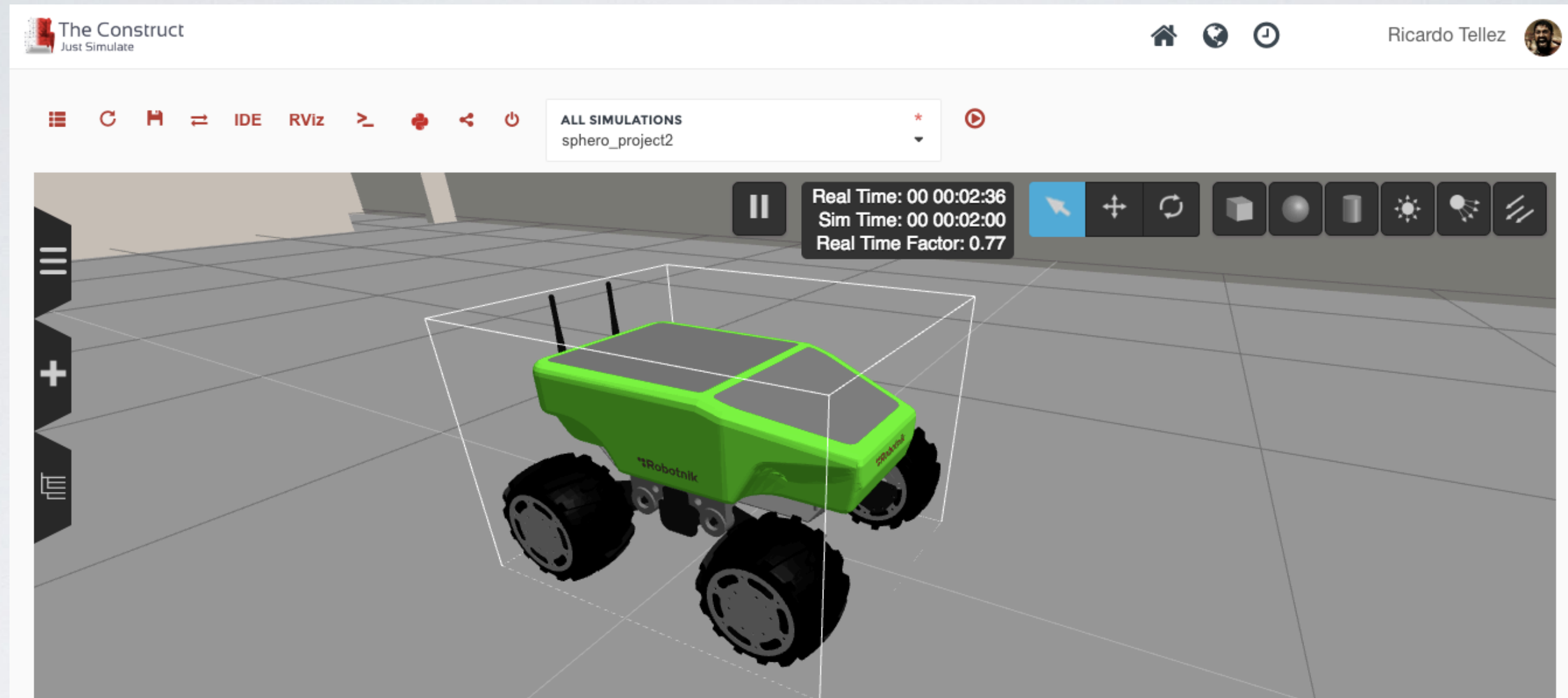
STEP 5: Provide Example Code

- Add some code to the Unit so the student can use or modify.
- You can provide it as a template

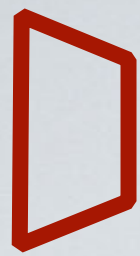




STEP 6: Continue With Next Unit



- Repeat the whole cycle for the rest of units

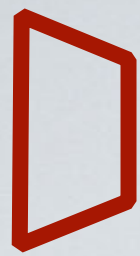


STEP 7: ADD A PROJECT



- The project must contain an exercise that includes all the units knowledge
- Include if possible **connection with real robot**

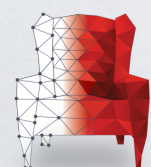


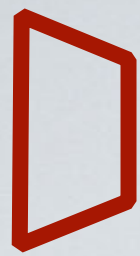


STEP 8: ADD AN EXAM



- **EXAM IS SUPER IMPORTANT!!!**
- Not only to evaluate, but also to make them learn





STEP 9: Share With Students

- Generate the rosject link and share with the students

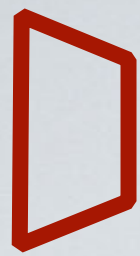


The
Construct
Learn and develop for robots



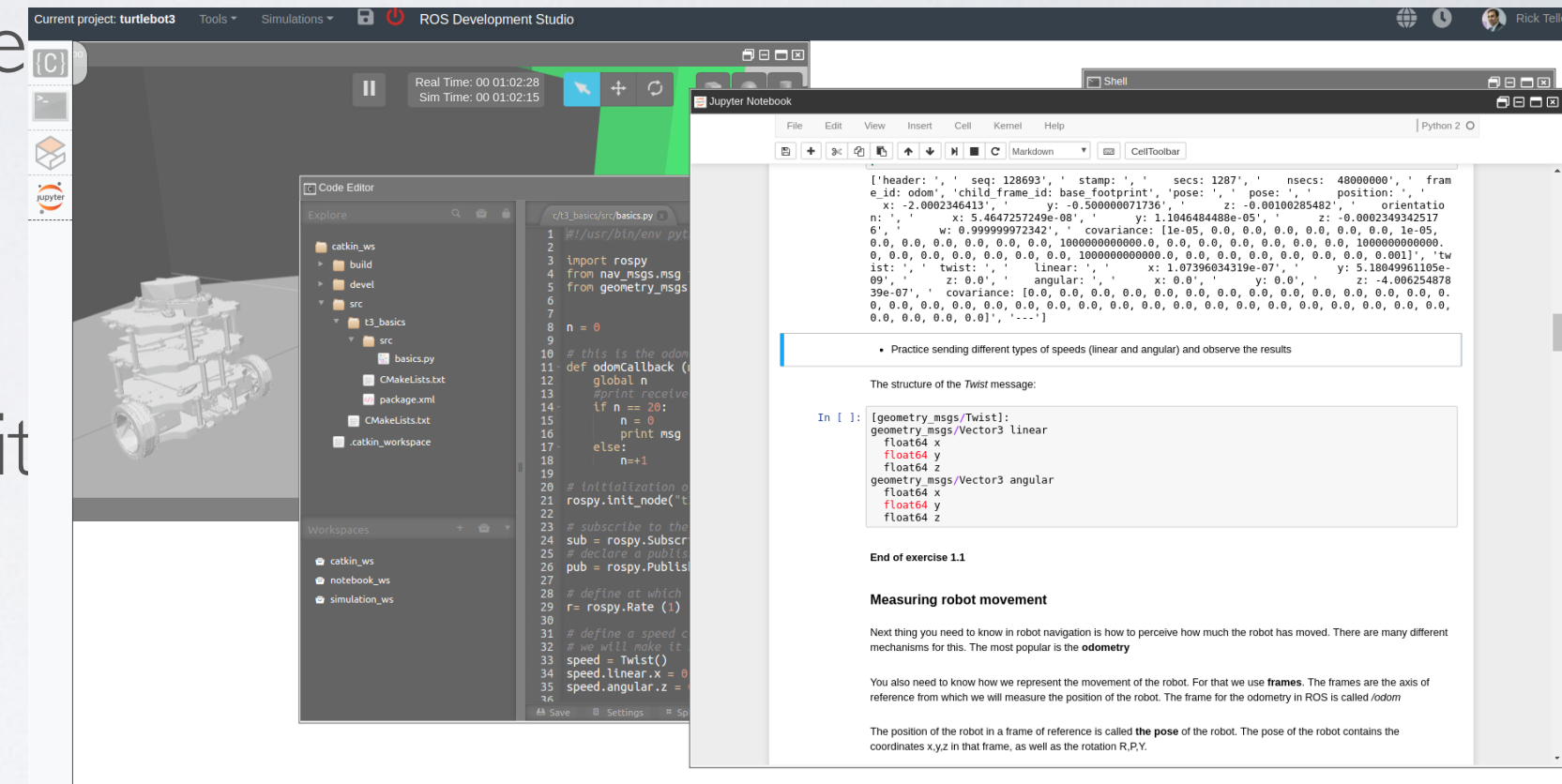
Exam: <http://www.rosject.io/l/b5e14b5/>

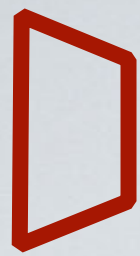




SUMMARY OF THE WHOLE PROCESS

1. Decide **subject** and **programming language**. Also, decide **units** of the course
2. Create a **rosject** for each unit
3. Get a **robot simulation** for each unit
4. Create a **Jupyter notebook** for each unit
5. Create some **sample code**
6. **Repeat** for each unit, project and exam
7. **Share** with students





ATTENTION!

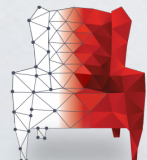
- Provide some previous training about:
 - **Linux**
 - **Python**



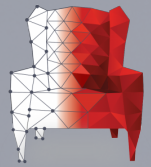
Free online courses:

Linux for robotics: <https://tinyurl.com/yxuo5urh>

Python for robotics: <https://tinyurl.com/y2en8pl8>



The
Construct
Learn and develop for robots



The
Construct
Learn and develop for robots

ROS WEBINAR QUESTIONS ?

Ricardo Téllez | CEO of The Construct



What our clients say



“With The Construct our students can jump right into ROS without all the hardware and software setup problems. And the best: they can do this from everywhere”

Steffen Pfiffner

Lecturer at University of Weingarten