L2 Lab sheet- Intelligent mobile robotics (IMR)

Introduction to ROS

The **ROS Development Studio** (ROSDS) provides a ROS robotics development environment. It runs on a cloud. You can start to work with simulated robots and if everything works, you could transfer your code to run real robots.

Aims

- Create ROS Development Studio login account
- Learn useful ROS commands in terminal window
- Gain a basic understanding about nodes, topics and … in the Robot Operating System (ROS)

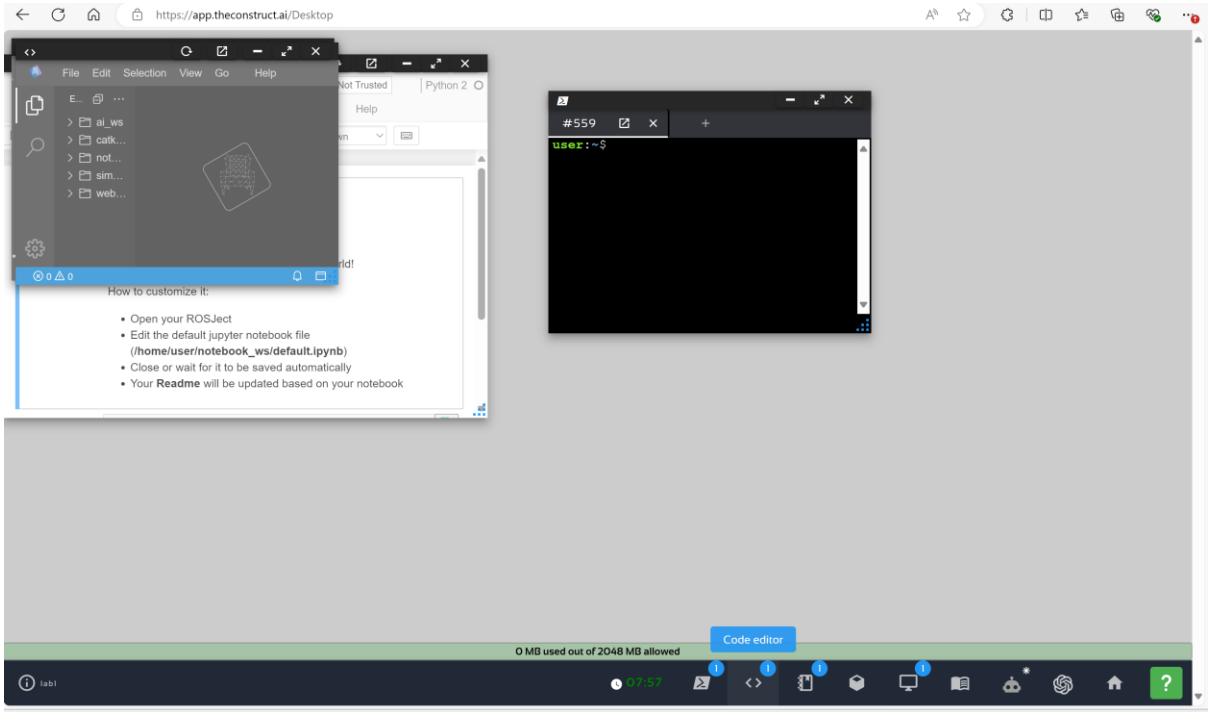
To achieve the aims do the following steps:

1- Create an account for The ROS Development Studio using the following link:

The ROS Development Studio by The Construct - The Construct

https://www.theconstructsim.com/the-ros-development-studio-by-the-construct/

2- Log in and create a new project
3- Go to My ROSJect tab and create a new project (For now you could use ROS 1 Noetic distribution)
4- Run the newly created Rosject



5- Open a web shell (a Linux terminal, or a command window) using the first tab on the left bottom side of the page, and Issue the following command to start the Master in a new terminal window:
$ roscore

- **roscore** is a collection of nodes and programs that you **must** have running to start ROS and creates the Master
- so that nodes can register with the Master and can communicate
  have a look on the subsection: '**Invoking the ROS Master using roscore**' in chapter 1 of the following book to learn more about the output:
- ROS Robotics By Example
  By Carol Fairchild, Dr. Thomas L. Harman
  The pdf of the book is available on DMU library::
  https://dmu.summon.serialssolutions.com/?s.q=ROS+Robotics+By+Example&s.cmd=#!/search?ho=t&l=en-UK&q=ROS%20by%20example

6- The terminal window used to execute 'roscore' must remain active, but it can be minimized. In another terminal window, type the following command:
$ **rosnode list**
What will be the result?

7- In the same terminal window, list the active topics by typing:
$ rostopic list
What is the results?

Note that the /rosout node and the /rosout topic have the same name but they need to be distinguished. The rosout node subscribes to the /rosout topic. Debug messages of all the active nodes are published to the /rosout topic, and the message published to this topic are useful to debug a program. see http://wiki.ros.org/rosout for more information.
The rosout node is connected to every other active node in the system.
The /rosout_agg topic receives messages only from the rosout node so it does not have to connect to all of the nodes and thus saves time at system startup.

8- Most of the ROS commands have help screens that are usually helpful. Type the following command to get help:
$ rosnode -h
By using the subcommand name, you could get help for the sub command, for example:
$ rosnode list -h

9- The *rosrun* command takes the arguments *[package name] [executable name]*:
rosrun [package name] [executable name]
for example:
$ rosrun turtlesim turtlesim_node
The previous command will run the 'turtlesim' package

10- Run the appropriate commands to get the list of active node and topics
11- Continue to work on the chapter 1 from the book in the reference [1]. For instance try to get access to the message which is in a topic related to turtlesim package:
rostopic type <topic name>
rostopic echo <topic Nam>

- Have a look on the subsection: 'Turtlesim – the first ROS robot simulation' 26 in chapter 1, and the following table in the following book to learn more about the output:

- ROS Robotics By Example
  By Carol Fairchild, Dr. Thomas L. Harman
  The pdf of the book is available on DMU library::
  https://dmu.summon.serialssolutions.com/?s.q=ROS+Robotics+By+Example&s.cmd=#!/search?ho=t&l=en-UK&q=ROS%20by%20example

Topics and messages for the /turtlesim node			
Topic name	**Topic type**	**Message format**	**Message**
`$ rostopic list`	`$ rostopic type [topic name]`	`$ rosmsg show [topic type]`	`$ rostopic echo [topic name]`

`/turtle1 /color_sensor`	`turtlesim/Color`	`uint8 r`
		`uint8 g`
		`uint8 b`
			`r: 69`
			`g: 86`
			`b: 255`

`/turtle1/pose`	`turtlesim/Pose`	`float32 x`
		`float32 y`
		`float32 theta`
		`float32 linear_velocity`
		`float32 angular_velocity`
			`x: 5.54444456`
			`y: 5.54444456`
			`theta: 0.0`
			`linear_velocity: 0.0`
			`angular_velocity: 0.0`

Command	Action	Example usage and subcommand examples
`roscore`	This starts the Master	`$ roscore`
`rosrun`	This runs an executable program and creates nodes	`$ rosrun [package name] [executable name]`

`rosnode`	This shows information about nodes and lists the active nodes	`$ rosnode info [node name]`
		`$ rosnode <subcommand>`
		Subcommand: `list`

`rostopic`	This shows information about ROS topics	`$ rostopic <subcommand> <topic name>` Subcommands: `echo`, `info`, and `type`
`rosmsg`	This shows information about the message types	`$ rosmsg <subcommand> [package name]/ [message type]` Subcommands: `show`, `type`, and `list`
`rosservice`	This displays the runtime information about various services and allows the display of messages being sent to a topic	`$ rosservice <subcommand> [service name]` Subcommands: `args`, `call`, `find`, `info`, `list`, and `type`
`rosparam`	This is used to get and set parameters (data) used by nodes	`$ rosparam <subcommand> [parameter]` Subcommands: `get`, `set`, `list`, and `delete`

Reference:

- [1] ROS Robotics By Example
  By Carol Fairchild, Dr. Thomas L. Harman
  The pdf of the book is available on DMU library::
  https://dmu.summon.serialssolutions.com/?s.q=ROS+Robotics+By+Example&s.cmd=#!/search?ho=t&l=en-UK&q=ROS%20by%20example