default

May 9, 2024

1 ROS2 Basics in 5 Days

1.1 Course Project

1.1.1 In this rosject, you will have to apply all that you have learned along the ROS2 Basics course to a real robot.

You will practice with a simulation and real robot.

The real robot is a Turtlebot3 that is running in Barcelona, Spain. You will connect remotely to it and practice on it.

2 How to proceed

This rosject is composed of three parts. You should do each part when the **ROS2 Basics in 5 Days** course indicates it.

- **PART I**: practice for topic publishers and subscribers
- **PART II**: practice for services
- **PART III**: practice for actions

Go to the part that the course indicated to you and try to finish it. But before doing that, **launch the simulation** of the project. You will use the simulation to practice with the simulated environment.

2.1 Try in simulation first

You should use the simulation to test your code while you are trying to get the exercises of this project done.

To launch the simulation, do the following:

1. Open a terminal by pressing on the *terminal icon* at the bottom left.
2. Type the following command that launches the simulation

[ ]:
```
source ~/simulation_ws/install/setup.bash
ros2 launch turtlebot3_gazebo main_turtlebot3_lab.launch.xml
```

To see the simulation, press the Gazebo button in the bottom left side of your screen, and select *Gazebo Window*

Wait about 30 seconds maximum for the simulation to start and the simulation should appear on the Gazebo window.

3 PART I: Topics

In this part, you are going to practice how to use topics to control a robot. Your goal is to create a ROS program that makes the robot have a wall-following behavior.

3.0.1 Wall-following behavior

The wall-following behavior is a behavior that makes the robot follow the wall on its right-hand side. This means that the robot must be moving forward at 30cm (1 feet) of distance from the wall at all times, having the wall on its right-hand side.

To achieve this behavior in the robot, you need to do two things:

Subscribe to the laser topic of the robot You need to subscribe to the laser topic and capture the rays. Select the ray of the right (the one that makes 90^o to the right with the front of the robot) and use it to measure the robot's distance to the wall.

- If the ray distance is **bigger than 0.3m**, you need to move the robot a little toward the wall, by adding some rotational speed to the robot
- If the ray distance is **lower than 0.2m**, you need to move the robot away from the wall, by adding rotational speed in the opposite direction
- If the ray distance is **between 0.2m and 0.3m**, just keep the robot moving forward

IMPORTANT

When the robot is moving along a wall, it can reach the next wall crossing its way. At that point in time, you should include a behavior that progressively transitions the robot from the current wall it's following to the next one.

For that, we recommend that you to use the front laser ray. If the distance measured by that ray is shorter than 0.5m, then make the robot turn fast to the left (moving forward at the same time).

The result of this behavior must be that the robot moves along the whole environment (see video below).

3.0.2 Test your wall-following program on the simulation

Create the wall-following program and test it on the simulation. Until it works on the simulation, you cannot try it on the real robot.

If the program doesn't work on the simulation, it is 100% not going to work on the real one.

Real life robot development works this way. First, try it on the simulation. When it works there, then you try it on the real. Never the other way around.

To test in simulation:

1. Make sure the simulation is launched as explained above
2. On another terminal, launch the keyboard teleop with the following command:

[ ]:
```
source ros2_ws/install/setup.bash
ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

3. Now use the keys to move the robot to a convenient position to test your ROS program. This means, close to a wall, looking along it. Remember that to be able to move the robot with the keys, **the terminal where you launched the teleop has to have the focus**.

IMPORTANT: close this *teleop* program once you have the proper position. Otherwise, it will interfere with your program.

4. Now launch your ROS program and see the results. If the robot doesn't behave correctly, debug your program and try to figure out the reason.

3.1 If working in simulation, go to the real robot

Once the robot is moving along the walls of the simulated environment, it is time for you to test it on the real robot.

The steps are: 1. Book a date and time in the *Real Robot Lab* to test you program (see Appendix below that teaches how to do that). 2. On the day and time selected, open this rosject. 3. Connect to the real robot lab from within this rosject (see Appendix below). 4. Launch the keyboard program and move the real robot to the proper position to start when you launch your program. 5. Launch your program and see if it works properly. Chances are it will not because simulation is not an exact copy of the real environment. Now is your time to debug. 6. Test on the real robot and debug your errors until you have it working properly.

4 PART II: Services

4.0.1 Create a ROS service that looks for the wall

In this part, you will have to do three things: 1. Create a ROS service server that when called, will make the robot go to the nearest wall 2. Modify the PART I program so that it calls the ROS service server before starting to follow the wall 3. Create a new launch file to launch both nodes

Step 1: Create the ROS service server

- Create a new ROS node that contains a service server named *find_wall*. The server uses a *FindWall* message, which you must create.

[ ]:
```
FindWall.srv

---
bool wallfound
```

When the service is called, the robot must do the following behavior:

1. Identify which of the laser rays is the shortest one. We assume that that is the one that is pointing to a wall.
2. Rotate the robot until the front of the robot is facing the wall. This can be done by rotating the robot until the ray 0 is the smaller one.
3. Move the robot forward until the ray 0 is shorter than 0.3m.
4. Now rotate the robot again until ray number 270 of the laser range is pointing to the wall.
5. At this point we consider that the robot is situated to start moving along the wall
6. Return the service message with a *True*.

How to test the service

- Launch the service server node
- Use the terminal to test the service with a service call command

[ ]:
```
ros2 service call find_wall
```

This should start the robot movement looking for the wall.

Step 2: Modify the PART I program

- Add a service client to the node of the part I.
- Call the service before the control loop of that node, so the robot gets prepared autonomously before starting to follow the wall

How to test it

- Launch the service server node
- Launch the wall following node.

This should start the robot movement (first the robot looks for the wall, then starts to follow the wall).

Step 3: Create a new launch file

- Create a new launch file named *main.launch* that launches both nodes: first the service server node and then the wall-following node.

This should start the robot movement (first the robot looks for the wall, then starts to follow the wall).

IMPORTANT: test everything on the simulation. When you have it working on the simulation, book a date in the *Real Robot Lab* and make it also work there.

5 PART III: Actions

5.0.1 Create an action server that records odometry

In this part, you will have to do:

1. Create an action server that when called, starts recording odometry
2. Add a call to the action server from the wall-follower node
3. Include the launch of the action server in the *main.launch*

Step 1: Create the action server

- Create a new ROS node that contains an action server named *record_odom*. The server uses a *OdomRecord.action* message, which you must create.

[ ]:
```
OdomRecord.action

---
geometry_msgs/msg/Point32[] list_of_odoms
---
float32 current_total
```

- The server must start recording the (x,y,theta) odometry of the robot as a Point32, one measure every second.
- As feedback, the action server must provide the total amount of meters that the robot has moved so far
- When the robot has done a complete lap, it has to finish and return the list of odometries recorded

How to test the server

- Launch the action server node
- Use the terminal to launch the *axclient* and test the action server

Step 2: Modify the PART I program

- Add an action service client to the node of part I.
- Call the action server before the control loop of that node, so the robot starts recording odometry before starting to follow the wall

How to test it

- Launch the action server node.
- Launch the *main.launch*.

This should start the robot movement and start recording.

- Check the feedback topic of the action server to see how it is going.

Step 3: Create a new launch file

- Add to *main.launch* the launch of the action server node.

This should start the robot movement and odometry recording.

IMPORTANT: test everything on the simulation. When you have it working on the simulation, book a date in the *Real Robot Lab* and make it also work there.

6 PART IV: Book a date to present your project on Youtube (optional)

An important part of your learning is to teach to others what you have learned.

In order to get the most out of your learning, you should deliver a live presentation of your project.

If you want to take this brave step, contact us at info@theconstructsim.com requesting a presentation and we will agree on the day and time.

You will have to: 1. Prepare your presentation. You will have to explain your project in 20 minutes, how you solved it, and how it works. 2. On the day agreed, you will need to have a camera and mic ready. 3. We will contact you on the day and prepare the whole broadcast. We will handle everything, you only need to be ready to present.

The event will be broadcast on Youtube so anybody around the world can attend and watch your presentation.

7 PART V: Share your project on your LinkedIn or social networks

The rosject that you created with this project is a demonstration of your value as a ROS Developer.

You can share your rosject on any website to show the ROS projects you have worked on. This can help future employers hire you.

In order to share your rosject: * go to your list of rosjects * click on the *Share* button of your rosject to get the share link.

- then you can publish the link anywhere

7.1 APPENDIX: How to connect to the real robot in RoBoX

Once you know the basics of the operation with a simulated version, it's time to use the real Turtlebot3 robot.

For that, you have to follow these simple steps:

7.1.1 STEP1: Book a session

In the main dashboard, you can book a session by clicking on this icon:

Here a booking page will appear where you can do two things:

- Check your previously-made bookings
- Make a new booking

You can also click on the **BOOK A SESSION** button to make a new one, which will take you to this menu, where you will select:

- The **Type of Robot** you want to book. For the moment, only Turtlebot3 is available.
- The Date and Time of your reservation.

The **date and times shown are the ones available**. They come in **periods of 25 minutes**.

There is also a limitation on the number of bookings per week a user can make.

This depends on your particular license and subscription.

7.1.2 STEP 2: Launch the RoBoX-Turtlebot3 ROSJect

To have the best experience, you need to launch the rosject with the simulation of Turtlebot3. Chances are that if you are reading this, you have already launched it.

When **one hour or less** is remaining until your booking session, a warning will be shown both in **the Dashboard** and inside the rosject **desktop**. This will be indicated with an **orange dot** on the **RoBox** icon.

It will then turn into a **red dot** when your booking session has started.

Note: Also, when you are inside your booking, the icon for the streaming window will appear in the desktop inside the rosject.

7.1.3 STEP3: Turn ON the RoBoX connection to the robot you booked

Once you are inside the rosject and in your **BOOKED TIME**, a toggle for **turning on the connection** will appear.

WARNING: Nothing will appear unless you have a booking. So if you didn't make a booking in the dashboard, feel free to do so.

You now just have to click and it will start the **turning-on process**. The terminals will be restarted, and when the system is ready, you will be able to open them again.

Now you should be able to do the folowing command, getting a list of topics similar to this. It could take an extra 30 seconds to appear:

Execute in WebShell #1

[ ]:
```
ros2 topic list
```

Note: Take into account that you **will not be able to push git changes when turned on, because there is no interenet connection**.

So to push any changes in your remote gits in the rosject, you will have to turn off the connection.

You are now connected to the robot! So let's try moving the robot around and seeing the lasers and the camera in rviz.

Execute in WebShell #1

[ ]:
```
ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

Execute in WebShell #2

[ ]:
```
rviz2
```

You can then add an image and scan elements in the **base_link** frame, getting something like this: