**Robotics Application**

**(COT 5930)**

**Logo, company name

Description automatically generated**

**Assignment: ROS2 Commands**

**Submitted To**

Prof. Zvi Roth, Dept. EECS

**Submitted By**

Neelima Rajawat, MS in AI

Tajheim Peart, undergraduate



**ROS2 Commands**

**See this document for instructions:**[**Understanding ROS2 Commands (Assignment)Links to an external site.**](https://docs.google.com/document/d/1zgGoPijEV5sjRgLmBN7oxnfIVo6CITKhLyh5ZKQLxsg/edit?usp=sharing)

**For this guide, you will be using many different ros2 commands to perform different tasks on the robot.**

**You can submit this assignment either**

**1. as a video where you explain the commands and what they do, along with showing the robot performing the associated tasks**

**or**

**2. A video demonstrating the tasks and a word/pdf document explaining how the commands work.**

**Background:**

There are many ways to control what the robot does. In the past assignment, we saw that we could run a script and move the robot with our keyboard. But we can also control the robot using ROS2 commands, which can be single line commands from the ubuntu terminal. We can also create programs in either C++ or Python, which makes it easier to do more complex tasks.

For now, we will just be running simple ROS2 single line commands to perform various tasks.

The Create3 robot has the ability to interpret things, like moving in arc, in one command and move the wheels in different directions/speeds automatically.

You can press Ctrl + C to stop the current command. Each command can take from a few seconds up to 30 seconds to run.

Instruction: **Make sure your robot is on the ground.**

**Instruction - 1**

First, you need to install the correct messages On your User-PC, connect to the fau network. Then run the following command:  
**sudo apt install -y ros-humble-irobot-create-msgs**Then when that is completed, switch back to the RoboticApp network.

**Explanation:**

We followed the instruction 1 for installing package **ros-humble-irobot-create-msgs** using **apt** package manager on our system. Here, **sudo** grants superuser privileges to perform system level installation, and **-y** flag auto-confirms prompts during installation.

**Instruction - 2**

When you run **ros2 topic list -t** you will see a list of all the ROS2 topics currently available along with the type of message.

**Note:** If you see only a few items show up, try again. Otherwise, check your connections between your user computer, the pi, and the create3 and that the network works.

**Explanation:**

We used this command **ros2 topic list -t** to see the list of available topics, and here -t options are used for specifying the topics with types should be displayed. Basically, this command helps us to identify and navigate topics within the ROS2 system.

**Instruction - 3**

First, put your create3 on the ground and on the doc, make sure the green light on the dock turns on. We can tell the robot to perform specific goals. Run the following command to undock:

ros2 action send\_goal /undock irobot\_create\_msgs/action/Undock "{}"

What happens? (Don’t just say it undocked. Describe what it did to get to its next position.)

If this doesn’t work, first double check the dock is connected properly. If it still doesn’t work, don’t worry about it, but think about why that might be? (Think about what is involved in undocked and what needs to be done first)

**Explanation:**

The command `ros2 action send\_goal` is used in ROS 2 to send a goal to an action server.

In this case, it's targeting the action server named `/undock`. The goal is for the `Undock` action defined in the `irobot\_create\_msgs` package, and it's an empty JSON object, implying that it's initiating an "undocking" task for a robot or system. The specific behavior triggered by this command depends on the implementation of the `Undock` action and the underlying robot or system.

**Video:** **Instruction-3** is explaining & showing the robot movements

**Instruction - 4**

There are 6 lights on the light ring. You can change each one of them individually.  
Run the following command but set your own color values. Change the 0s to match the color you want.  
  
ros2 topic pub /cmd\_lightring irobot\_create\_msgs/msg/LightringLeds "{override\_system: true, leds: [{red: 0, green: 0, blue: 0}, {red: 0, green: 0, blue: 0}, {red: 0, green: 0, blue: 0}, {red: 0, green: 0, blue: 0}, {red: 0, green: 0, blue: 0}, {red: 0, green: 0, blue: 0}]}"   
  
Hint: use <https://www.google.com/search?q=color+picker> to choose colors, and take the R, G, B values and Red, green, and blue.  
  
When you are done, run the following command to revert the leds back to it’s default values:  
ros2 topic pub /cmd\_lightring irobot\_create\_msgs/msg/LightringLeds "{}"

**Explanation:**

Here **ros2 topic pub** is used to publish messages to a ros2 topic **/cmd\_lightring.**

This is sending the **LightringLeds** type messages from **irobot\_create\_msgs** package. This message is containing data to control LED lights, setting up to different RGB colors with an override to ensure this action takes precedence. The message will likely instruct a robot or system to turn off specific LEDs on a light ring.

Here, we modify the command, and added a few showing a few colors from RGB

ros2 topic pub /cmd\_lightring irobot\_create\_msgs/msg/LightringLeds "{override\_system: true, leds: [{red: 247, green: 12, blue: 165}, {red: 247, green: 12, blue: 32}, {red: 40, green: 12, blue: 247}, {red: 12, green: 142, blue: 247}, {red: 12, green: 247, blue: 102}, {red: 247, green: 224, blue: 12}]}"

Then ran this command and got output shown in below figures I4, and I4C

Also please refer the **video** **Instruction-4** for seeing results.

A robotic vacuum cleaner on the floor

Description automatically generated

Figure. I4

A computer on a desk

Description automatically generated

Figure. I4C

Then we used this command, to revert the leds back to its default values.

Ros2 topic pub /cmd\_lightring irobot\_create\_msgs/msg/LightringLeds “{}”

Please refer the video **Instruction-4(1)** for explanation and seeing clearing off lights

**Instruction- 5**

You can also have the Create3 play notes. Run the following command, changing the frequency to your own values. Use two different frequencies between 500 and 1000 Hz.

ros2 topic pub –once /cmd\_audio irobot\_create\_msgs/msg/AudioNoteVector “{append: false, notes: [{frequency: 0, max\_runtime: {sec: 1,nanosec: 0}}, {frequency: 0, max\_runtime: {sec: 1,nanosec: 0}}]}”  
  
Notice in this command, we use –once so the command only runs one time.

**Explanation:**

This command **ros2 topic pub --once** is used to publish a single message to a ROS2 topic named **/cmd\_audio**. The message is of type **AudioNoteVector** from **the irobot\_create\_msgs** package. It specifies a note vector with two entries, each indicating a note with a frequency of 0 and a maximum runtime of 1 second. The append parameter is set to false, suggesting that the new note vector will replace any existing data on the **/cmd\_audio** topic, and the notes provided are for audio-related instructions, such as playing specific notes on a robot or system.

ros2 topic pub –once /cmd\_audio irobot\_create\_msgs/msg/AudioNoteVector “{append: false, notes: [{frequency: 500, max\_runtime: {sec: 1,nanosec: 0}}, {frequency: 700, max\_runtime: {sec: 1,nanosec: 0}}, [{frequency: 600, max\_runtime: {sec: 1,nanosec: 0}}, [{frequency: 800, max\_runtime: {sec: 1,nanosec: 0}},]}”

Please refer **video Instruction-5** for seeing results of this audio command.

**Instruction- 6**

Let’s take a moment to understand the structure of the commands above. It looks like there are three different parts of the command. What are they? Hint: look at the topics from ros2 topic list and ros2 action list.

**Note**: I am not looking for “ros2” “topic” “pub” or “--once”, I am asking about the three parts after that. You should notice that these are generally the same for many of the commands such as the ones below or the ones on the create3 documentation.

**Explanation:**

Let’s break down the structure of the commands:

1. **Topic and Message Type:**

* **/cmd\_audio:** This is the ROS 2 topic to which the message will be published.
* **irobot\_create\_msgs/msg/AudioNoteVector:** This specifies the message type that you are publishing on the topic. It tells ROS 2 what the structure of the message should be.

1. **Message Content:**

The content of the message is enclosed in curly braces `{ }` and includes the message fields and their values. In this case, it appears to be a complex message with nested fields, such as `notes`, each containing subfields like `frequency` and `max\_runtime`. This part specifies the actual data we want to send in the message.

1. **Message Data:**

The data you want to publish is specified within the double quotes `" "`. It includes a list of audio notes with their frequencies and maximum runtimes. The message fields and their values are structured according to the `irobot\_create\_msgs/msg/AudioNoteVector` message type.

The command follows the general pattern of `ros2 topic pub` with specific arguments for the topic name and message type, The command is used to send audio note information to the specified ROS 2 topic `/cmd\_audio`, and it appears to be related to controlling an iRobot Create device's audio functionality.

**Instruction- 7**

In the command below:  
ros2 action send\_goal /drive\_distance irobot\_create\_msgs/action/DriveDistance "{distance: 0.5,max\_translation\_speed: 0.15}"  
  
Change the distance (meters) and the speed (speed is below 0.306 m/s) to your own values and then run the command.

What happens?

**Explanation:**

The command ros2 action **send\_goal** is used to send a goal to a ROS2 action server named **/drive\_distance.** It targets the **DriveDistance** action defined in the **irobot\_create\_msgs** package. The goal instructs a robot to drive a distance of 0.9 units with a maximum translation speed of 0.6. The robot should execute this motion in response to the goal.

We used the command with 0.9 meters distance and at 0.6 speed.

ros2 action send\_goal /drive\_distance irobot\_create\_msgs/action/DriveDistance "{distance: 0.9,max\_translation\_speed: 0.6}"

Please refer **Instruction-7 video** for more explanation and seeing robot movements.

**Instruction- 8**

In the command below:  
ros2 action send\_goal /rotate\_angle irobot\_create\_msgs/action/RotateAngle "{angle: 0,max\_rotation\_speed: 0.5}"

Change the angle (radians) and the rotation speed.

Hint: The angle is in radians. So = 1.57.

What happens?

**Explanation:**

This command sends a goal to a ROS 2 action server named "rotate\_angle" in the "irobot\_create\_msgs" package. The goal specifies an angle of rad and a maximum rotation speed of 0.5. It is used to request the robot to perform a rotation with these parameters.

ros2 action send\_goal /rotate\_angle irobot\_create\_msgs/action/RotateAngle "{angle: 1.03,max\_rotation\_speed: 0.5}"

Please refer **Instruction-8 video** for more explanation and seeing robot movements.

**Instruction- 9**

In the command below:  
ros2 action send\_goal /drive\_arc irobot\_create\_msgs/action/DriveArc "{angle: 1.03,radius: 0.6,translate\_direction: 1,max\_translation\_speed: 0.3}"

Change the angle (in radians) and radius (meters) to your own values. Note: You can use negative values. For the angle, it will make it turn the other way.

What happens when you run the command?

**Explanation:**

This command sends a goal to a ROS 2 action server named "drive\_arc" in the "irobot\_create\_msgs" package. The goal specifies an arc with a zero angle, a radius of 0.3, a forward translation direction, and a maximum translation speed of 0.3. It is used to instruct the robot to drive in a curved path with these parameters.

Please refer **Instruction-9 video** for more explanation and seeing robot movements.

**Instruction- 10**

The following command  
ros2 action send\_goal /navigate\_to\_position irobot\_create\_msgs/action/NavigateToPosition "{achieve\_goal\_heading: true,goal\_pose:{pose:{position:{x: 1,y: 0.2,z: 0.0}, orientation:{x: 0.0,y: 0.0, z: 0.0, w: 1.0}}}}"  
will move the robot to a specific position from your current location.  
  
The robot will take a rotate -> translate -> rotate approach to achieve the goal position. First rotating from its current heading to face the goal position, then driving straight to the goal position, then optionally rotating to achieve the goal heading."  
  
Try to play around with the values and see what happens and explain them.

Pose position: {0,0,0}

X=0.4, y=0.5, z=0.6

**Explanation**

ros2 action send\_goal /navigate\_to\_position irobot\_create\_msgs/action/NavigateToPosition "{achieve\_goal\_heading: true,goal\_pose:{pose:{position:{x: 1,y: 0.2,z: 0.0}, orientation:{x: 0.4,y: 0.5, z: 0.6, w: 1.0}}}}"

This ROS 2 command sends a goal to the **/navigate\_to\_position** action server. The goal instructs an iRobot Create to achieve a specific heading and reach a target pose. The target pose includes position coordinates (x: 1, y: 0.2, z: 0.0) and an orientation (x: 0.4, y: 0.5, z: 0.6, w: 1.0) in a 3D space. The action server will then execute the necessary actions to achieve this goal.

Please refer **Instruction-10 video** for more explanation and seeing robot movements.

**Instruction- 11**

Run the following command, but make sure to change the values to your own values:

ros2 topic pub -r 20 /cmd\_vel geometry\_msgs/msg/Twist “{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}”

**Explanation:**

This ROS 2 command publishes messages to the **/cmd\_vel** topic at a rate of 20 Hz. It sends a **Twist** message with zero linear and angular velocities, effectively commanding a robot or vehicle to stop moving.

we tried this command 3-times.

ros2 topic pub -r 20 /cmd\_vel geometry\_msgs/msg/Twist “{linear: {x: 0.2, y: 0.5, z: 0.5}, angular: {x: 0.3, y: 0.4, z: 0.7}}”

Please refer **video instruction-11**, **video instruction-11(2)**, **video instruction-11(3)** for explanation and seeing robot movements.

**Instruction- 12**

Using the link below, find some other action, goal, or read a status event. Run the command you find, explain what it does, and show a video if applicable.

Do not use anything from previous parts of the assignment or their “opposites”. i.e., Don’t try docking since undocking is on here. If unsure, email Josh Thaw

This tutorial is based on <https://iroboteducation.github.io/create3_docs/api/ros2/>

Use the link to find more examples of ros2 commands.

**Explanation:**

For this, we worked with wall\_follow command

/wall\_follow [irobot\_create\_msgs/action/WallFollow]

A computer screen with text and images

Description automatically generated

Please refer **Instruction-12 video** for more explanation and seeing robot movements.