

LAB #4: ROS2 USING RCLPY IN JULIA

Abdelbacet Mhamdi
Senior-lecturer, Dept. of EE
ISET Bizerte — Tunisia
a-mhamdi

Namouchi Skander
Dept. of EE
ISET Bizerte — Tunisia
Iskander000

You are required to carry out this lab using the REPL as in Figure 1.

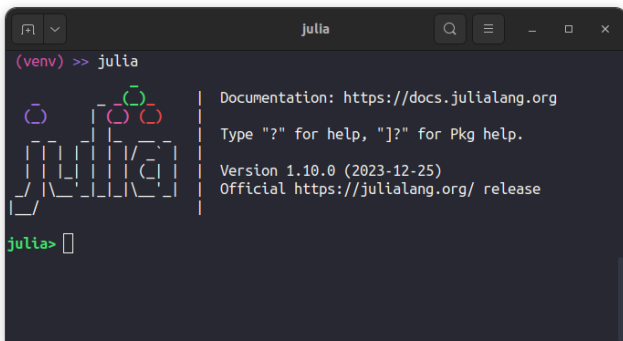


Figure 1: Julia REPL

1) instructions ::

We begin first of all by sourcing our ROS2 installation as follows:

```
source /opt/ros/humble/setup.zsh
```

- then we need to open the subscriber and the publisher programmes in two different terminal

```
using PyCall
# Import the rclpy module from ROS2 Python
rclpy = pyimport("rclpy")
str = pyimport("std_msgs.msg")

# Initialize ROS2 runtime
rclpy.init()

# Create node
node = rclpy.create_node("my_publisher")
rclpy.spin_once(node, timeout_sec=1)

# Create a publisher, specify the message type and the topic name
pub = node.create_publisher(str.String, "infodev", 10)

# Publish the message `txt`
for i in range(1, 100)
    msg = str.String(data="Hello, ROS2 from Julia!")
```

```
($(string(i)))"
    pub.publish(msg)
    txt = "[TALKER] " * msg.data
    @info txt
    sleep(1)
end

# Cleanup
rclpy.shutdown()
node.destroy_node()
```

1

```
using PyCall

rclpy = pyimport("rclpy")
str = pyimport("std_msgs.msg")

# Initialization
rclpy.init()

# Create node
node = rclpy.create_node("my_subscriber")

# Callback function to process received messages
function callback(msg)
    txt = "[LISTENER] I heard: " * msg.data
    @info txt
end

# Create a ROS2 subscription
sub = node.create_subscription(str.String, "infodev", callback, 10)

while rclpy.ok()
    rclpy.spin_once(node)
end

# Cleanup
node.destroy_node()
rclpy.shutdown()
```

- In a newly opened terminal, the execution will show to us that the subscriber listens to the messages being

¹Remember to source ROS2 installation before using it with Julia

broadcasted by our previous publisher and respond by a message

to do that we need first of all to create a nodes called subscriber and publisher

```
# Create publisher node
node = rclpy.create_node("my_publisher")
rclpy.spin_once(node, timeout_sec=1)
# Create subscriber node
node = rclpy.create_node("my_subscriber")
```

to link the two talker and the heard to a specific topic like here we choose **infodev** as a topic like showing the graph

```
pub = node.create_publisher(str.String,
"infodev", 10)

# Create a ROS2 subscription
sub = node.create_subscription(str.String,
"infodev", callback, 10)
```

- The graphical tool **rqt_graph** of Figure 2 displays the flow of data between our nodes: *my_publisher* and *my_subscriber*, through the topic we designed *infodev*

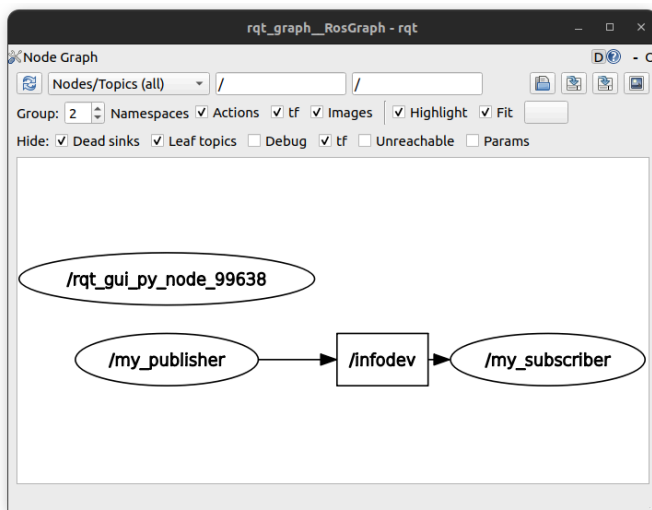


Figure 2: rqt_graph

- after linked publisher / subscriber together and then to a specific topic the execution terminal will show us what in fig 3

```
(venv) >> source /opt/ros/humble/setup.zsh
(venv) >> julia
Activating project at ~/MEGA/git-repos/infodev/Codes'
Documentation: https://docs.julialang.org
Type "?" for help, "]" for Pkg help.
Version 1.10.0 (2023-12-25)
Official https://julialang.org/ release

julia> include("ros2/publisher.jl")
Info: [TALKER] Hello, ROS2 from Julia! (1)
Info: [TALKER] Hello, ROS2 from Julia! (2)
Info: [TALKER] Hello, ROS2 from Julia! (3)
Info: [TALKER] Hello, ROS2 from Julia! (4)
Info: [TALKER] Hello, ROS2 from Julia! (5)
Info: [TALKER] Hello, ROS2 from Julia! (6)
Info: [TALKER] Hello, ROS2 from Julia! (7)
Info: [TALKER] Hello, ROS2 from Julia! (8)
Info: [TALKER] Hello, ROS2 from Julia! (9)
Info: [TALKER] Hello, ROS2 from Julia! (10)
Info: [TALKER] Hello, ROS2 from Julia! (11)
Info: [TALKER] Hello, ROS2 from Julia! (12)
Info: [TALKER] Hello, ROS2 from Julia! (13)
Info: [TALKER] Hello, ROS2 from Julia! (14)
Info: [TALKER] Hello, ROS2 from Julia! (15)
Info: [TALKER] Hello, ROS2 from Julia! (16)
Info: [TALKER] Hello, ROS2 from Julia! (17)

(venv) >> source /opt/ros/humble/setup.zsh
(venv) >> julia
Activating project at ~/MEGA/git-repos/infodev/Codes'
Documentation: https://docs.julialang.org
Type "?" for help, "]" for Pkg help.
Version 1.10.0 (2023-12-25)
Official https://julialang.org/ release

julia> include("ros2/subscriber.jl")
Info: [LISTENER] I heard: Hello, ROS2 from Julia! (1)
Info: [LISTENER] I heard: Hello, ROS2 from Julia! (2)
Info: [LISTENER] I heard: Hello, ROS2 from Julia! (3)
Info: [LISTENER] I heard: Hello, ROS2 from Julia! (4)
Info: [LISTENER] I heard: Hello, ROS2 from Julia! (5)
Info: [LISTENER] I heard: Hello, ROS2 from Julia! (6)
Info: [LISTENER] I heard: Hello, ROS2 from Julia! (7)
Info: [LISTENER] I heard: Hello, ROS2 from Julia! (8)
Info: [LISTENER] I heard: Hello, ROS2 from Julia! (9)
Info: [LISTENER] I heard: Hello, ROS2 from Julia! (10)
Info: [LISTENER] I heard: Hello, ROS2 from Julia! (11)
Info: [LISTENER] I heard: Hello, ROS2 from Julia! (12)
Info: [LISTENER] I heard: Hello, ROS2 from Julia! (13)
Info: [LISTENER] I heard: Hello, ROS2 from Julia! (14)
Info: [LISTENER] I heard: Hello, ROS2 from Julia! (15)
Info: [LISTENER] I heard: Hello, ROS2 from Julia! (16)
Info: [LISTENER] I heard: Hello, ROS2 from Julia! (17)
```

Figure 3: Minimal publisher/subscriber in ROS2

- to change the message or the number of the messages prodcasted or respondet we can use this line

```
# Publish the message `txt`
for i in range(1, 100)
    msg = str.String(data="Hello, ROS2 from Julia!"
    ($(string(i)))")
    pub.publish(msg)
    txt = "[TALKER] " * msg.data
    @info txt
    sleep(1)
end

# Callback function to process received messages
function callback(msg)
    txt = "[LISTENER] I heard: " * msg.data
    @info txt
end
```

- if you haad a problem in the topic wich one you should use you can check from the topic list by right down this line

```
source /opt/ros/humble/setup.zsh
ros2 topic list -t
```

Figure 4 shows the current active topics, along their corresponding interfaces.

```
mhamdi@e590:~/MEGA/git-repos/...
(venv) >> source /opt/ros/humble/setup.zsh
(venv) >> ros2 topic list -t
/infodev [std_msgs/msg/String]
/parameter_events [rcl_interfaces/msg/ParameterEvent]
/rosout [rcl_interfaces/msg/Log]
(venv) >>
```

Figure 4: List of topics