Lab #4: ROS2 using RCLPY in Julia

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

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

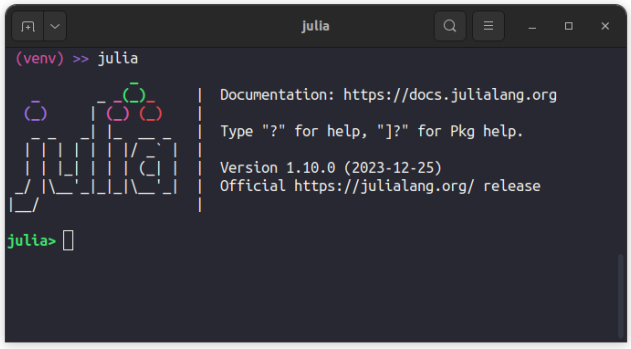

Figure 1: Julia REPL

Exo 1: Minimal Publisher/Subscriber Setup

The combination of Julia and rclpy opens up opportunities for developing efficient and performant robotics applications with the benefits of ROS2s ecosystem.

Make sure to read the instructions thoroughly, follow each step precisely, and ask for clarification if needed.
We begin first of all by sourcing our ROS2 installation as follows:

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

Always start by sourcing ROS2 installation in any newly opened terminal.

Open a *tmux* session and write the instructions provided at your Julia REPL.

```
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()
```

In a newly opened terminal, we need to setup a subscriber that listens to the messages being broadcasted by our previous publisher[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
```

[1]Remember to source ROS2 installation before using it with Julia

```
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()
```

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*.

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

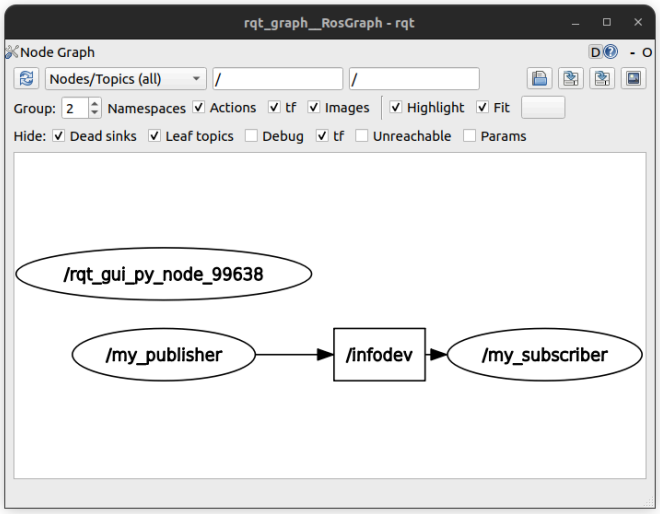

Figure 2: rqt_graph

Figure 3 depicts the publication and reception of the message *"Hello, ROS2 from Julia!"* in a terminal. The left part of the terminal showcases the message being published, while the right part demonstrates how the message is being received and heard.

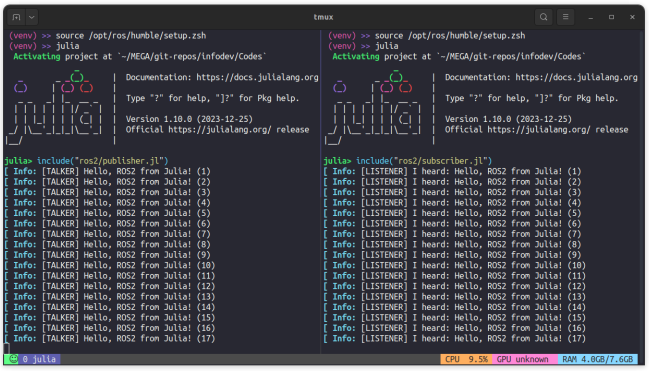

Figure 3: Minimal publisher/subscriber in ROS2

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

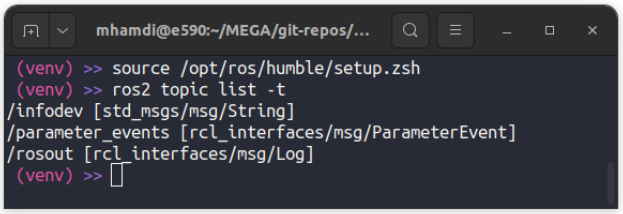

Figure 4: List of topics