Lab #4: ROS2 using RCLPY in Julia

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

Houimli Ala Eddine
Dept. of EE
ISET Bizerte — Tunisia
Alaaa20

I. Introduction

In this lab we gonna control subscriber - publisher with ROS2 and that required to carry out this lab using the REPL as in Figure 1.

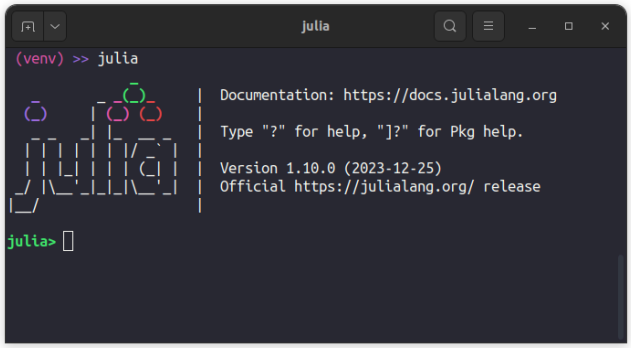

Figure 1: Julia REPL

II. Applications :

In this part of the report we gonna explain the fonction of the programme of subscriber and publisher
1) *Publisher:*
We begin first of all by sourcing our ROS2 installation as follows:

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

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()
```

this programme **the publisher** will send a sepecific message by using this code to write down the message and repeated in loop

```
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
```

then we need to published on specifique topic in way that the subscriber can read it

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

We can change the publisher name by write down this line

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

2) *subscriber:*

In a newly opened terminal, we need to setup a subscriber that listens to the messages being broadcasted by our previous publisher like explain the graph under

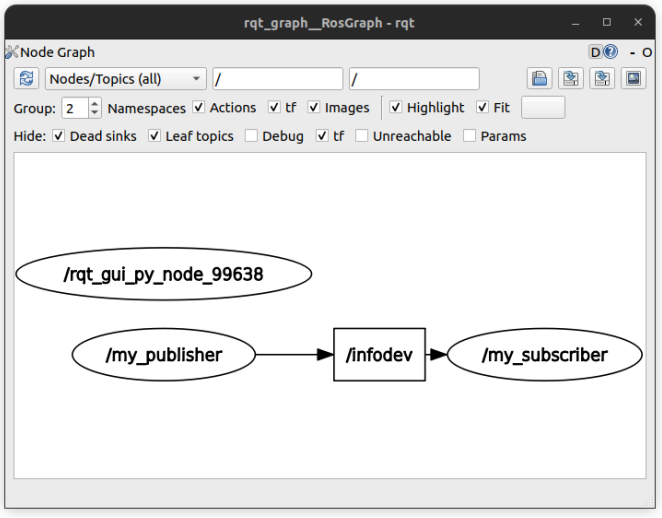

Figure 2: rqt_graph

```
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()
```

To do that we gonna use subscriber programme as in the first programme to creat a subscriber with specifique name we need to use this lines

```
node = rclpy.create_node("my_subscriber")
```

then to link the two the subscriber and the publisher we should write them in the same topic like

```
sub = node.create_subscription(str.String,
"infodev", callback, 10)
```

in this way we linke the two in a topic named infodev To know that the message has delivered sucessfuly we need from the subscriber to respond by a massage for that we write down this:

```
function callback(msg)
    txt = "[LISTENER] I heard: " * msg.data
    @info txt
end
```

To simulate we should open each programme in a newly terminal in this way the terminal will show us what fig 3 show from the right the publisher message and on the left the subscriber respond

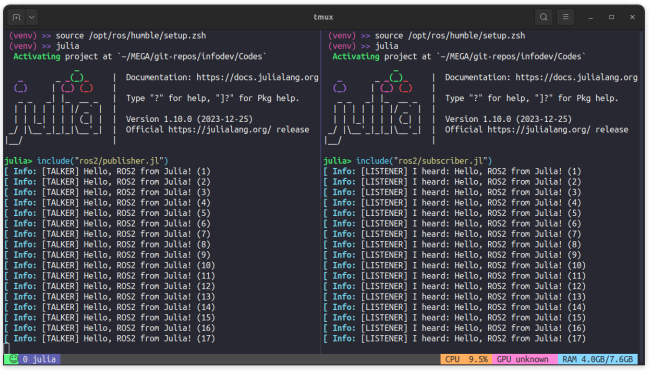

Figure 3: Minimal publisher/subscriber in ROS2