

# LAB #4: ROS2 USING RCLPY IN JULIA

Mohamed Bejaoui  
Dept. of EE  
ISET Bizerte — Tunisia  
📧 bejaouimohamed

Jamila Gharbi  
Dept. of EE  
ISET Bizerte — Tunisia  
📧 Gharbijamila

## I. INTRODUCTION

In this report, we will explain how we have use Julia and ROS2 to create a basic ROS 2 communication setup where one node publishes messages and another node subscribes to those messages on a specified topic. This allows for data exchange between different components of a robotic system or between multiple robotic systems.

## II. CREATION OF THE PUBLISHER :

### A. Installation of ROS2 and writing code :

We begin first of all by sourcing our ROS2 installation by writing this command in the terminal :

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

After that , we create a first julia file named **publisher** and we write this commands as shown in code below :

```
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 code essentially simulates a ROS 2 talker node in Julia, where it continuously publishes messages to a specific topic at a certain frequency.

### B. Explanation of the code :

In This part , we will explain what that code does from the beginning to the end :

1. It imports the necessary modules from ROS 2 Python (rclpy and std\_msgs.msg).
2. Initializes the ROS 2 runtime
3. Creates a ROS 2 node named **"my\_publisher"**
4. Creates a publisher for the **"infodev"** topic with a message type of String.
5. Enters a loop to publish message to the topic in a range from 1 to 100.
6. Constructs the public message with the format **"Hello, ROS2 from Julia!"** where i is the current loop iteration.
7. Publishes the message to the topic.
8. Logs the published message using info .
9. Sleeps for 1 second before the next iteration
10. Cleans up by shutting down the ROS 2 runtime and destroying the node

Some of the commands can give us some informations about :

- The topic **"infodev"**

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

- The name of publisher **"my\_publisher"**:

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

- Message published **"Hello, ROS2 from Julia!"**

```
msg = str.String(data="Hello, ROS2 from Julia!"
$(string(i)))")
```

### III. CREATION OF THE SUBSCRIBER :

#### A. Writing code :

Now , we create a second julia file named subscriber and we write this commands as shown in code below :

```
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()
```

This code creates a subscriber node in Julia that listens for messages on a specific topic in the ROS 2 system.

#### B. Explanation of the code :

We will explain the code like we do with the first one :

- It imports the necessary ROS 2 modules (rclpy and std\_msgs.msg) using PyCall and initializes the ROS 2 runtime environment.
- Creates a ROS 2 node named **"my\_subscriber"**
- Defines a callback function named callback to process received messages. The function prepends **"[LISTENER] I heard: "** to the received message
- Sets up a subscription for the **"infodev"** topic, specifying the message type String and the callback function to handle received messages.
- Enters a loop that runs as long as the ROS 2 system is operational.

- When a message is received on the **"infodev"** topic, the **'callback'** function is invoked to handle the message
- Once the loop exits (e.g., when the ROS 2 system shuts down), the node is destroyed to release its associated resources.
- Finally, the ROS 2 runtime environment is shut down.

without explain this command :

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

We can have the name of the subscriber which is **"my\_subscriber"**

### IV. RUNNING THE TWO CODES :

Firstly , we start by launching The graphical tool rqt\_graph by writing down this line to let the data flow between the publisher and subscriber by link it bouth of them to a node called **"infodev"** like showing figure 1

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

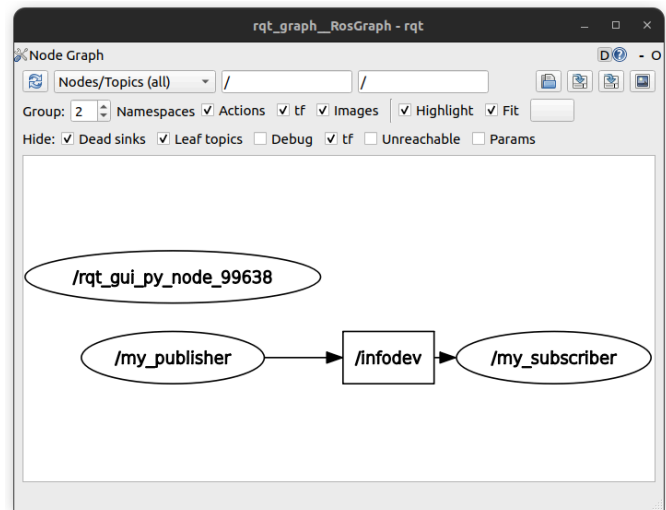
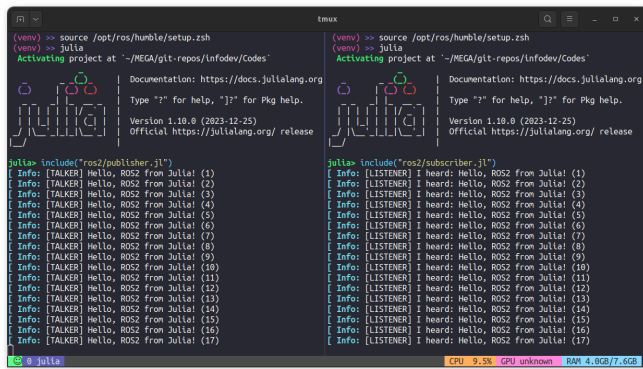


Figure 1: rqt\_graph

After, opening and running the two codes **"publisher.jl"** and **"subscriber.jl"** in the correct way and we obtain this result in the terminal :



The image shows two terminal windows side-by-side, both running in a tmux session. The left window shows a Julia script acting as a publisher, and the right window shows a Julia script acting as a subscriber. Both scripts are using the ROS2 interface.

```
(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, "j?" 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)

CPU: 9.5% CPU unknown RAM: 4.8GB/7.6GB
```

```
(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, "j?" 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 2: Minimal publisher/subscriber in ROS2

Figure 2 shows the publication and reception of the message ***“Hello, ROS2 from Julia!”*** in this terminal. The left part of the terminal shows us the message being published, while the right part demonstrates how the message is being received and heard. we also can have the numbers of publishing and receiving the message which is one hundred times .