Lab #4: ROS2 using RCLPY in Julia

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

Ala Boughanmi
Dept. of EE
ISET Bizerte — Tunisia
MATRIXBINARY

Skander Namouchi
Dept. of EE
ISET Bizerte — Tunisia
Iskander000

***Abstract* — Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do..**

I. Introduction

in this lab they are two part in the first part ("Application") we gonna use also julia REPL to write down ROS2s codes as the figure (1) and in the second part ("clarafication") we gonna explain every commandes and her function
Figure 1.

Figure 1: Julia REPL

in this lab i can't simulate ROS2 with my laptop for that i'm gonna use the simulation picture in Images/ infodev folder

II. Application

- first of all we need to install ROS2and then we gonna start sourcing our ROS2 installation as follows:

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

- second we gonna open up a julia terminal and write down the codes underneath or luckly we can open it from our folder infodev/codes/ros2

The first programme is the publisher code

```
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()
```

The second programme is the subscriber code

After write down the two programme we need to execute each one of them in a newly opened terminal, Right then the subscriber will listen to the message brodcasted by the publisher

```
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 lunch The graphical tool rqt_graph we need to write down this line to let the data fow between the publisher and subscriber by link it bouth of them to a node called “infodev” like showing figure 2 by write down this codes lines

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

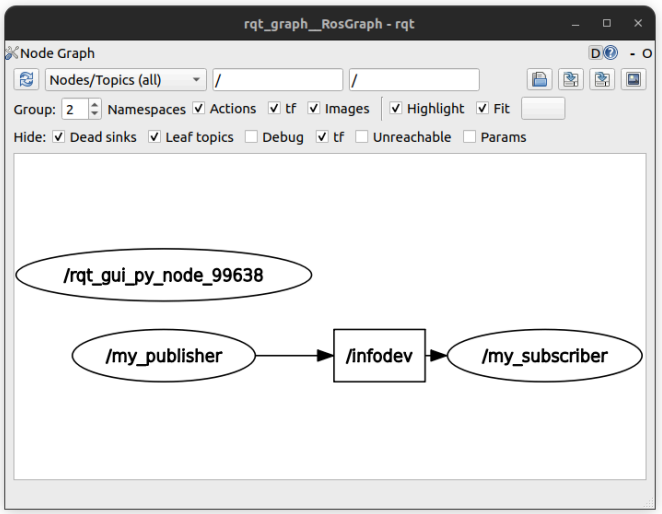

Figure 2: rqt_graph

- After linket the publisher and the subscriber the publisher will publish this message one hundred times to the node linked with subscriber

[Info [TALKER] Hello, ROS2 from Julia!(1...100)]

then the subscriber will respond, in the node ,by

[Info [LISTNER] I heard Hello, ROS2 from Julia!(1...100)]

as in figure 3

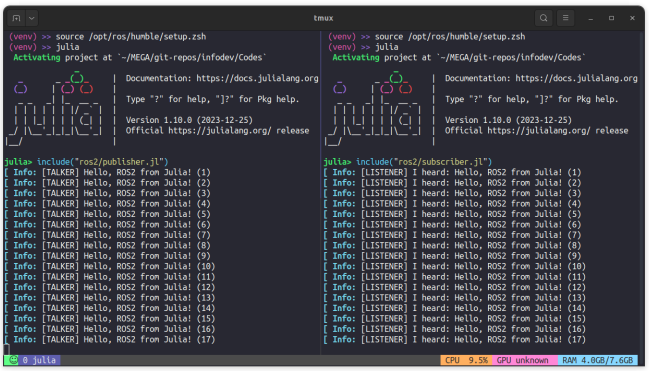

Figure 3: the dialog between the publisher and the subscriber

,

- Hint :

to know the current active topic we should write down this code then the terminal will show you the topic list as in figure 4

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

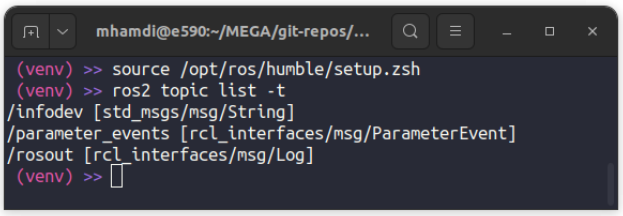

Figure 4: List of topics

III. Clarafication :

- in this part we gonna explain each code line and we gonna start with the pulisher code first :

The First programme is the spublisher code

```
using PyCall
```

- this package can be useful when you want to leverage existing Python libraries or utilize Python-specific

```
##Import the rclpy module from ROS2 Python
rclpy = pyimport ("rclpy")
```

- import the rclpy module from Python using PyCall in Julia. rclpy is a Python client library for the Robot Operating System (ROS) 2

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

- import the std_msgs.msg module from ROS 2 into Julia

```
rclpy.init()
```

- Initialize ROS2 runtime

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

- create a node named “my_publisher” using the rclpy

```
rclpy.spin_once(node, timeout_sec=1)
```

- use the spin_once function from the rclpy to execute a single iteration of the ROS 2 event loop within a given timeout period

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

- create a publisher within a ROS 2 node named node using the create_publisher function from the rclpy module in Python. This publisher will publish messages of a certain type on a particular topic and particular name

```
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
```

- create a publisher node in Julia using PyCall to communicate with a ROS 2 system. It publishes messages to a topic named “infodev” with a string message containing “Hello, ROS2 from Julia!” along with an incrementing number from 1 to 99.

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

- Deliting the rcply and destroy the node

The second programm : the subscriber code

```
rclpy = pyimport("rclpy")
```

- Import the rclpy module in Python using PyCall in Julia. This module is part of the Robot Operating System 2 (ROS 2) ecosystem and provides functionality for creating ROS 2 nodes, publishers, subscribers, and more.

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

- import the std_msgs.msg module from ROS 2 into Julia using PyCall. This module contains message types commonly used in ROS 2, such as standard messages for data types like strings, integers, floats, etc.

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

- creat a node called my subscriber in a specific topic

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

- define a callback function in Julia that will be called when messages are received by a subscriber. This function will print out the received message data along with a prefix indicating that it was received by the listener node.

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

- creat a subscriber within a ROS 2 node named node using the create_subscription function from the rclpy module in Python. This subscriber subscribes to messages of type std_msgs.msg.String on the topic “infodev” and invokes the callback function when messages are received.

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

- creat a loop in Python that continuously spins the ROS 2 node until the ROS 2 context (rclpy.ok()) is still valid. This loop ensures that the node continues to process messages and callbacks as long as the ROS 2 context is valid.