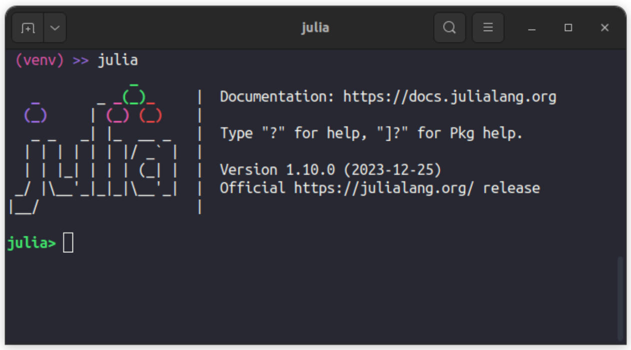
Lab #4: ROS2 using RCLPY in Julia

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

Nemri Adam
Senior-lecturer, Dept. of EE
ISET Bizerte — Tunisia
adamnemr

Hosni Marnisi
Senior-lecturer, Dept. of EE
ISET Bizerte — Tunisia
hosnimarnisi

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



```
(venv) >> julia
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.10.0 (2023-12-25)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

julia>
```

Figure 1: Julia REPL

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

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

The command "source /opt/ros/humble/setup.zsh" executes a setup scénario for ROS (Robot Operating System) within the Zsh shell. This scénario configures the environment by setting variables, aliases, and other configurations needed for working with ROS effectively. By running this command, you prepare your environment to utilize ROS tools and features seamlessly within Zsh.

```julia
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 Julia scénario uses PyCall to interact with the ROS2 (Robot Operating System 2)

1. Imports necessary Python modules (
   ```
   rclpy
   ```
   for ROS2 functionality and
   ```
   std_msgs.msg
   ```
   for courant glorification types).
2. Initializes the ROS2 runtime.
3. Creates a ROS2 node named "my_publisher".
4. Creates a publisher within the node to publish messages of type
   ```
   std_msgs.msg.String
   ```
   on the topic "infodev".
5. Publishes "Hello, ROS2 from Julia!" messages with an incremental number every participant for 100 iterations.
6. Prints logging moderne to the console with each published glorification.
7. Cleans up by shutting down the ROS2 runtime and destroying the node.

```julia
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()
```

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

This code combines two scripts for interacting with ROS2 (Robot Operating System 2) using Julia and PyCall:

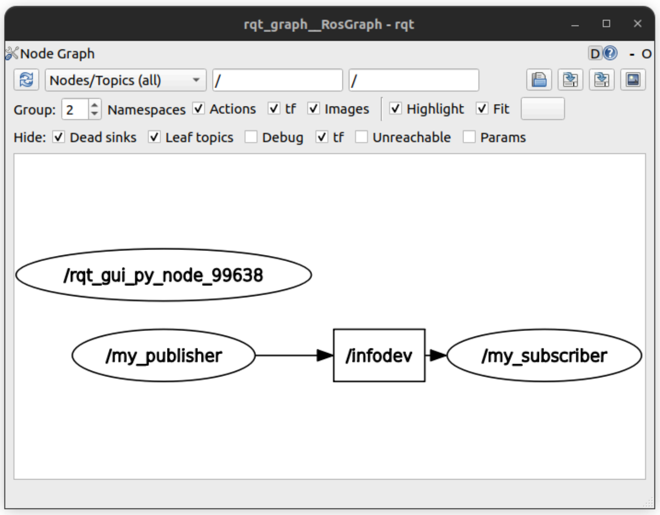
1. Publishing Script:
   - Initializes the ROS2 runtime.
   - Creates a publisher node that sends messages containing "Hello, ROS2 from Julia!" with incremental numbers to the "infodev" topic.
   - Shuts down the ROS2 runtime after publishing the messages.
2. Subscribing Script:
   - Initializes the ROS2 runtime.
   - Creates a subscriber node that listens for messages on the "infodev" topic.
   - Defines a callback function to process incoming messages by appending a prefix and logging them.
   - Continuously spins the node to handle incoming messages until the ROS2 runtime is shut down.

Additionally, the code includes commands to set up the ROS environment and visualize the ROS graph using

```
rqt_graph
```

.

In summary, this code demonstrates a simple interaction with ROS2, showcasing both message publishing and subscribing functionalities, along with environment setup and visualization of the ROS graph.



Nodes/Topics (all) / /

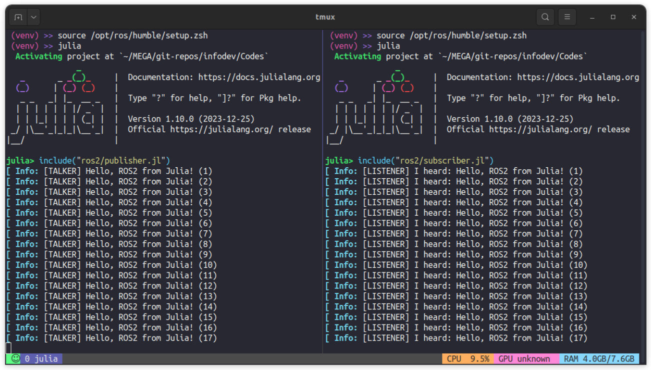
Group: 2 [✓ Namespaces] [✓ Actions] [✓ tf] [✓ Images] [✓ Highlight] [✓ Fit]

Hide: [✓ Dead sinks] [✓ Leaf topics] Debug [✓ tf] Unreachable Params

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.

- In this version, the publisher sends messages like "[Info [TALKER] Julia says, "Hello ROS2!" (1)]" up to 100 times, and the subscriber responds with "[Info [LISTENER] I heard, "Hello ROS2!" (1)]" for each message it receives.



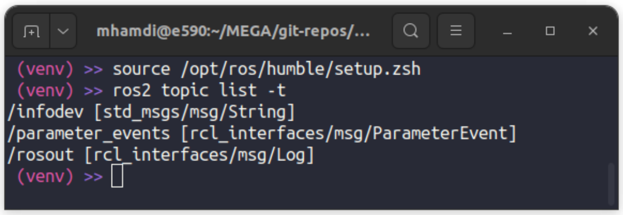
```
(venv) >> source /opt/ros/humble/setup.zsh
(venv) >> julia
 Activating project at `~/MEGA/git-repos/infodev/Codes`

               _
   _       _ _(_)_     |  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)
```

```
0 julia                                   CPU  9.5%  GPU unknown  RAM 4.0GB/7.6GB
```

Figure 3: Minimal publisher/subscriber in ROS2

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



```
(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

This line imports the PyCall module, which allows Julia code to interact with Python code.

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

Here, we use PyCall to import the rclpy module from Python.

```
rclpy
```

is a Python client library for the Robot Operating System (ROS) 2. This line enables us to access ROS 2 functionality from within Julia.

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

Similarly, this line imports the

```
std_msgs.msg
```

module from Python. The

```
std_msgs.msg
```

module typically contains message definitions for standard ROS 2 message types. By importing this module, we gain access to these message types, which we can use to define the structure of messages sent over ROS 2 topics.
So, in summary, the publisher code starts by importing necessary Python modules (

```
rclpy
```

for ROS 2 functionality and

```
std_msgs.msg
```

for standard message types) into Julia using PyCall. This allows us to leverage existing Python libraries and interact with ROS 2 in our Julia codebase.