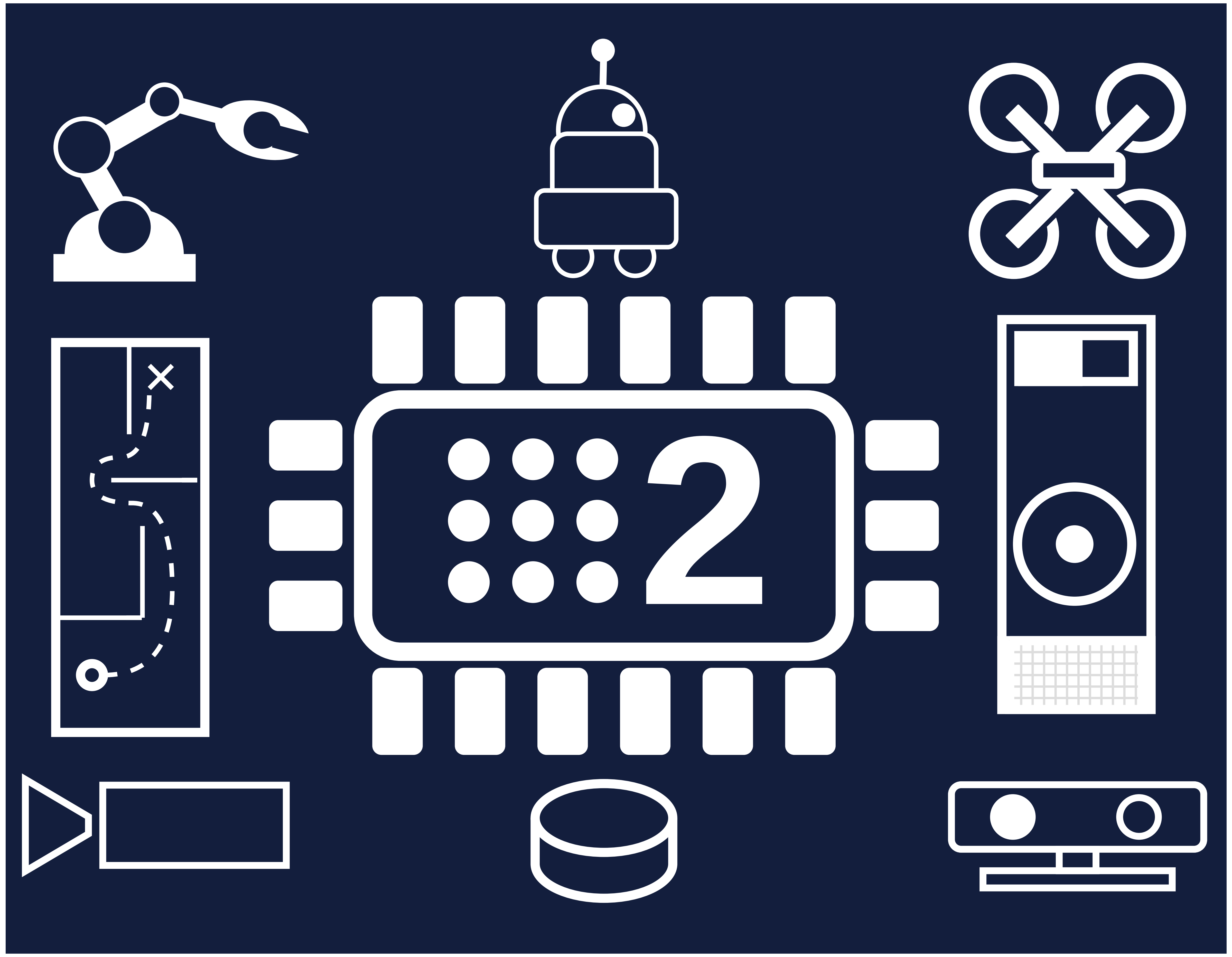


A Concise Introduction to Robot Programming in ROS2

Prof. Dr. Francisco Martín Rico

Chapter 2: First Steps with ROS2

francisco.rico@urjc.es
2022 @fmrico

Universidad Rey Juan Carlos

Intelligent Robotics Lab

First steps in the Terminal

ROS2Cli

```
$ ros2

usage: ros2 [-h] Call ‘ros2 <command> -h‘ for more detailed usage. ...
ros2 is an extensible command-line tool for ROS 2.
...
```

```
ros2 <command> <verb> [<params>|<option>]*
```

```
action      extension_points  node       test
bag         extensions        param      topic
component   interface         pkg        wtf
launch      run               daemon     lifecycle
security    doctor            multicast  service
```

Further readings:

- https://github.com/ros2/ros2cli
- https://github.com/ubuntu-robotics/ros2_cheats_sheet/blob/master/cli/cli_cheats_sheet.pdf

First steps in the Terminal

Packages

```
$ ros2 pkg list

ackermann_msgs
action_msgs
action_tutorials_cpp
...
```

```
$ ros2 pkg executables demo_nodes_cpp

demo_nodes_cpp add_two_ints_client
demo_nodes_cpp add_two_ints_client_async
demo_nodes_cpp add_two_ints_server
demo_nodes_cpp allocator_tutorial
...
demo_nodes_cpp talker
...
```

First steps in the Terminal

Packages

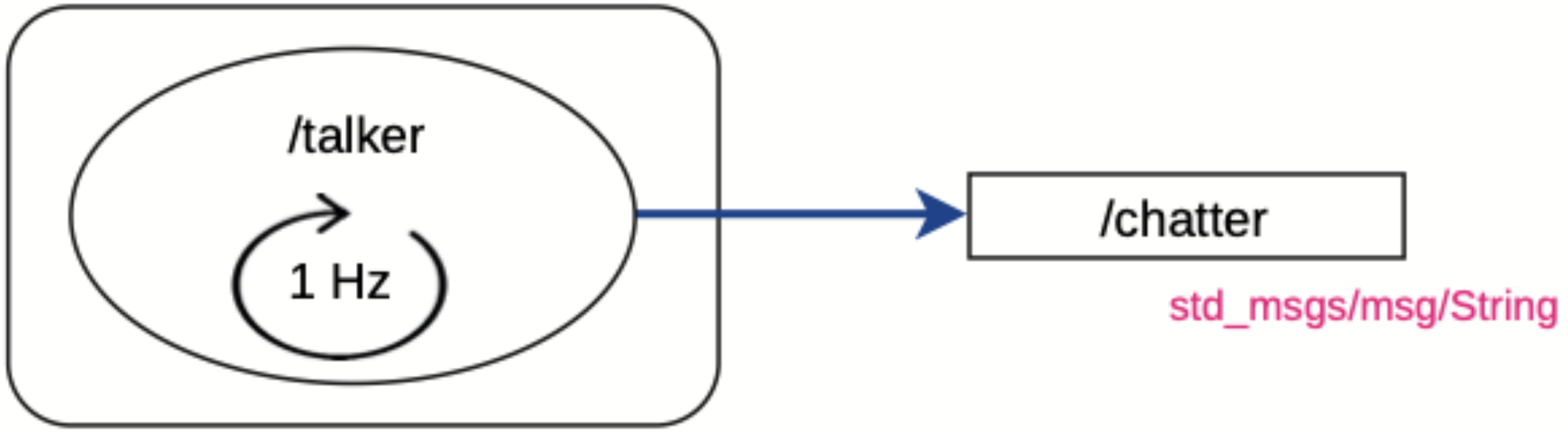
```
$ ros2 pkg list

ackermann_msgs
action_msgs
action_tutorials_cpp
...
```

```
$ ros2 pkg executables demo_nodes_cpp

demo_nodes_cpp add_two_ints_client
demo_nodes_cpp add_two_ints_client_async
demo_nodes_cpp add_two_ints_server
demo_nodes_cpp allocator_tutorial
...
demo_nodes_cpp talker
...
```

First steps in the Terminal

Running a ROS2 program

```
$ ros2 run demo_nodes_cpp talker

[INFO] [1643218362.316869744] [talker]: Publishing: 'Hello World: 1'
[INFO] [1643218363.316915225] [talker]: Publishing: 'Hello World: 2'
[INFO] [1643218364.316907053] [talker]: Publishing: 'Hello World: 3'
...
```

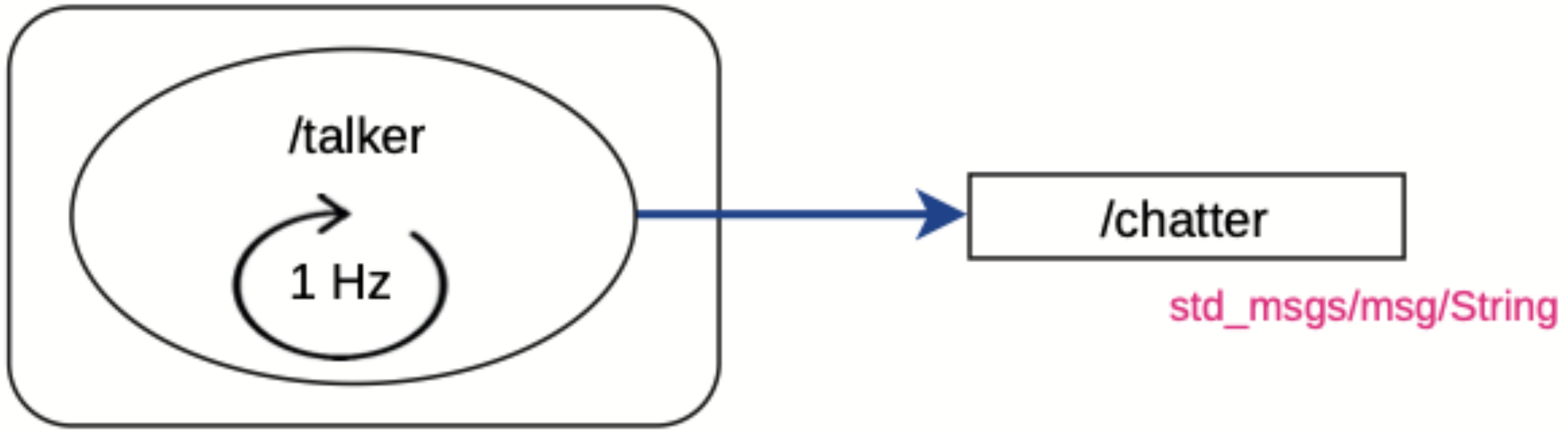


First steps in the Terminal

Running a ROS2 program

```
$ ros2 node list

/talker
```

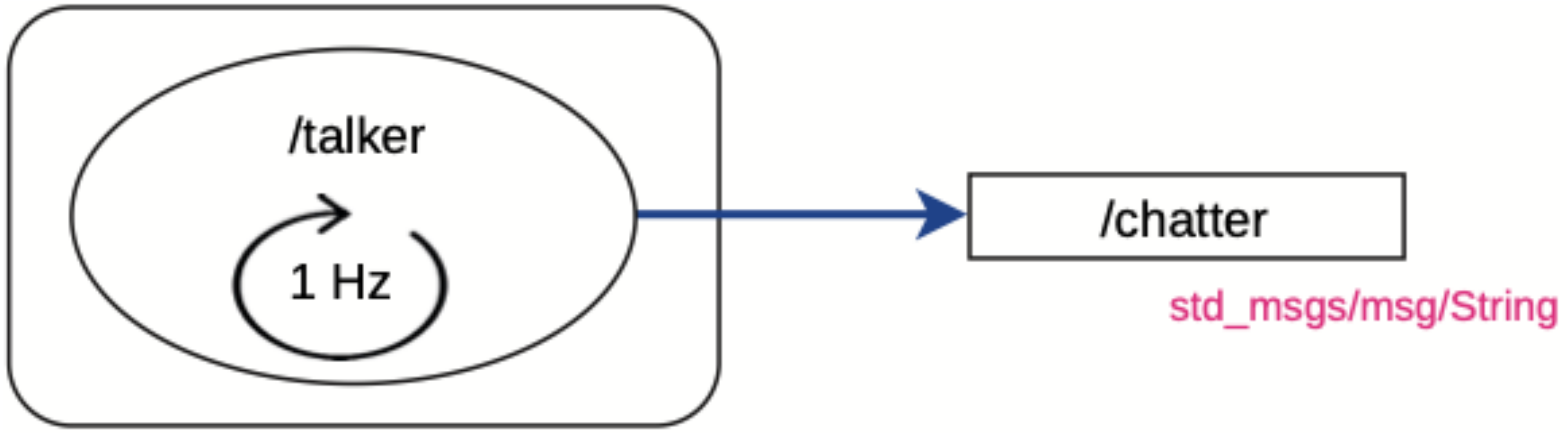
```
$ ros2 topic list

/chatter
/parameter_events
/rosout
```



First steps in the Terminal

Running a ROS2 program

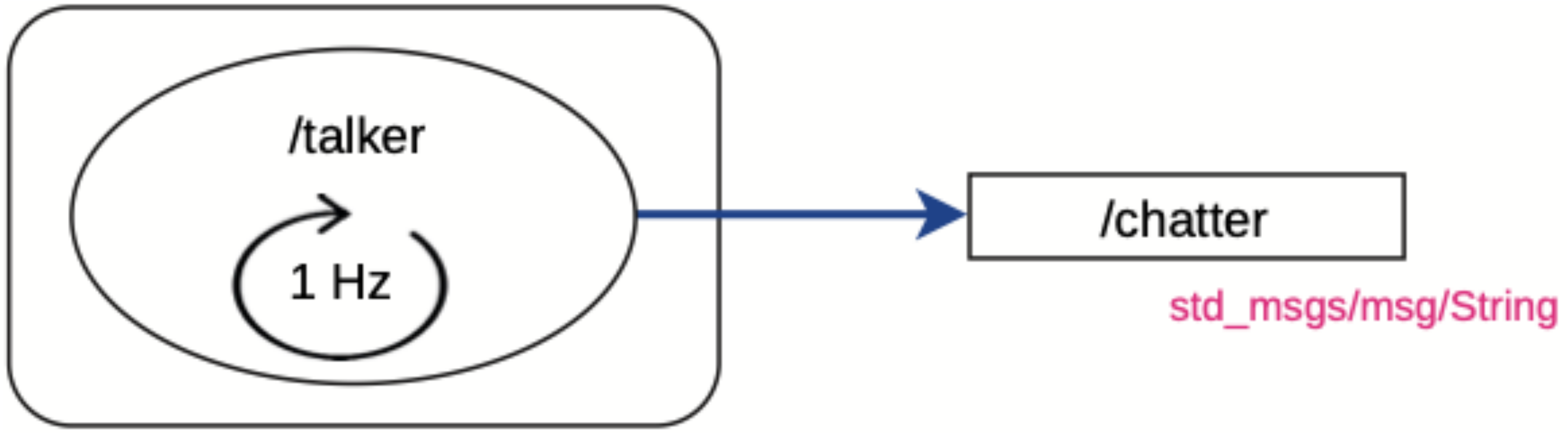
```
$ ros2 node info /talker

/talker
   Subscribers:
       /parameter_events: rcl_interfaces/msg/ParameterEvent
   Publishers:
       /chatter: std_msgs/msg/String
       /parameter_events: rcl_interfaces/msg/ParameterEvent
       /rosout: rcl_interfaces/msg/Log
   Service Servers:
...
```



First steps in the Terminal

Running a ROS2 program

```
$ ros2 topic info /chatter

Type: std_msgs/msg/String
Publisher count: 1

Subscription count: 0
```



First steps in the Terminal

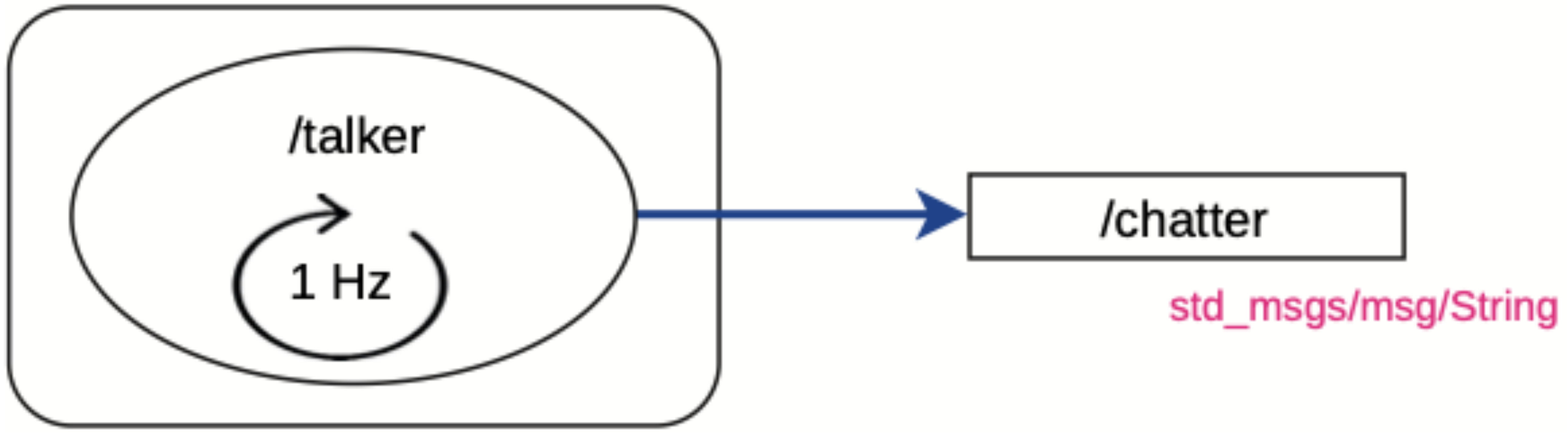
Interfaces

```
$ ros2 interface list

Messages:
    ackermann_msgs/msg/AckermannDrive
    ackermann_msgs/msg/AckermannDriveStamped
    ...
    visualization_msgs/msg/MenuEntry
Services:
    action_msgs/srv/CancelGoal
    ...
    visualization_msgs/srv/GetInteractiveMarkers
Actions:
    action_tutorials_interfaces/action/Fibonacci
    ...
```

```
$ ros2 interface show std_msgs/msg/String

... comments

string data
```

First steps in the Terminal

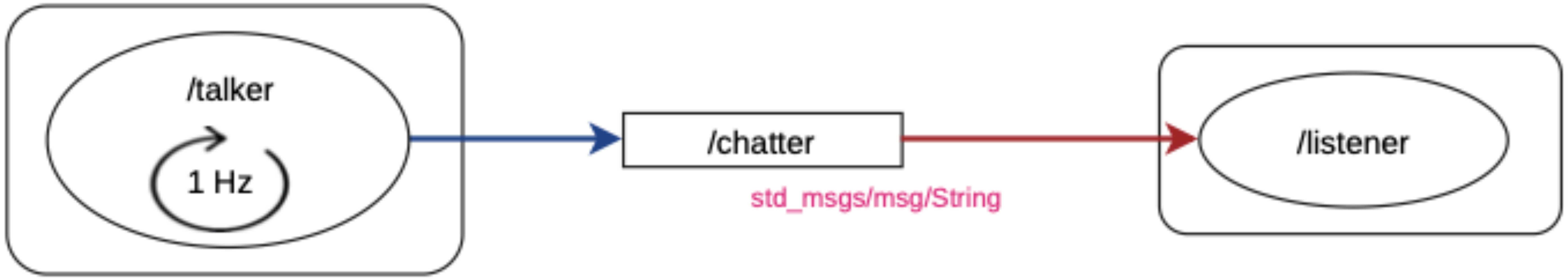
```
$ ros2 topic echo /chatter

data: 'Hello World: 1578'
---
data: 'Hello World: 1579'
...
```



First steps in the Terminal

Running a listener

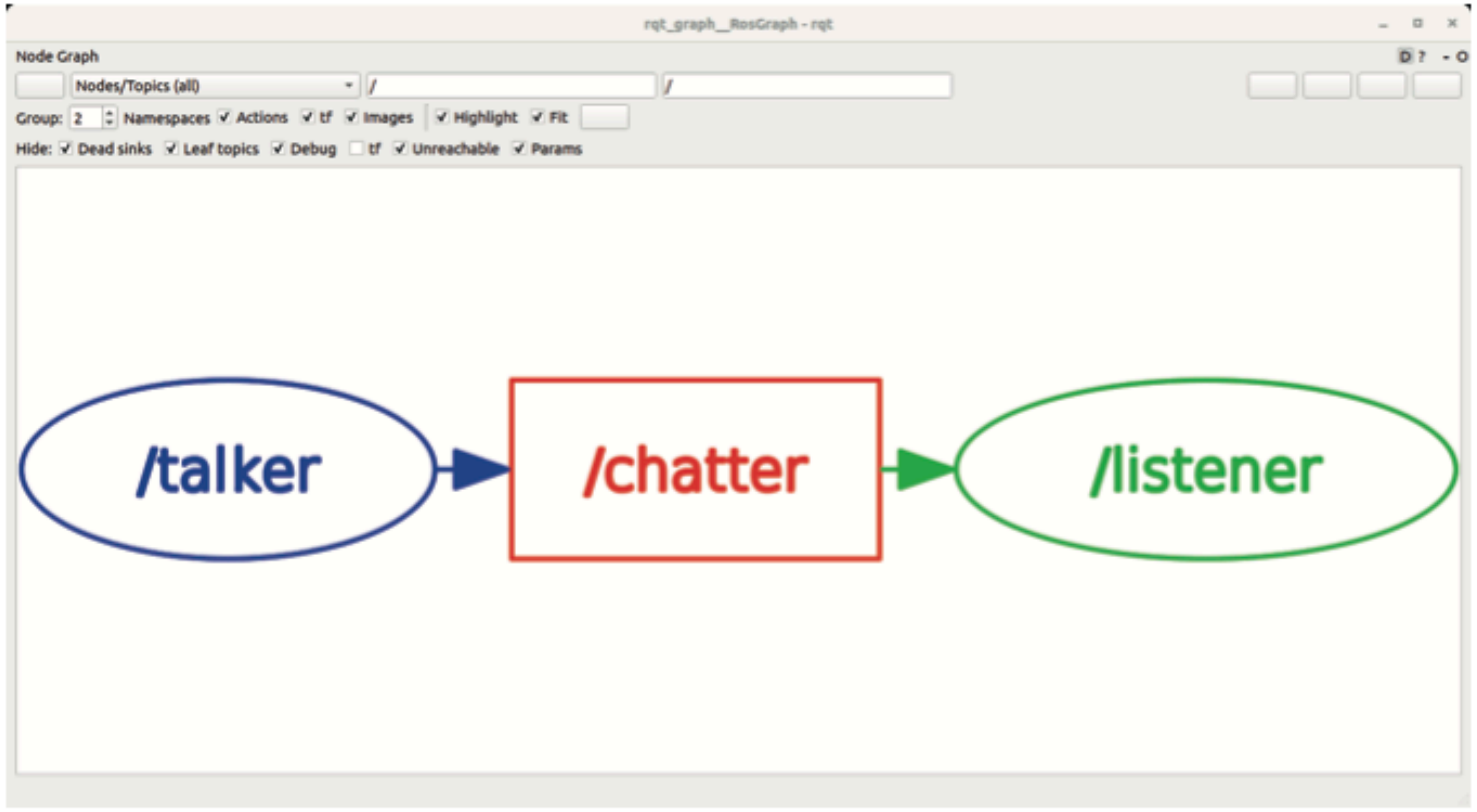
```
$ ros2 run demo_nodes_py listener

[INFO] [1643220136.232617223] [listener]: I heard: [Hello World: 1670]
[INFO] [1643220137.197551366] [listener]: I heard: [Hello World: 1671]
[INFO] [1643220138.198640098] [listener]: I heard: [Hello World: 1672]
...
```



First steps in the Terminal

RQT Tools

```
$ ros2 run rqt_graph rqt_graph
```



Francisco Martín Rico francisco.rico@urjc.es @fmrico

Developing your first node

Package creation

```
$ cd ~/bookros2_ws/src

$ ros2 pkg create my_package --dependencies rclcpp std_msgs
```

Package my_package

```
my_package/
├── CMakeLists.txt
├── include
│   └── my_package
├── package.xml
└── src
    └── simple.cpp
```

Developing your first node

Package.xml

```xml
<?xml version="1.0"?>
<?xml-model href="http://download.ros.org/schema/package_format3.xsd"
 schematypens="http://www.w3.org/2001/XMLSchema"?>
<package format="3">
  <name>my_package</name>
  <version>0.0.0</version>
  <description>TODO: Package description</description>
  <maintainer email="john.doe@evilrobot.com">johndoe</maintainer>
  <license>TODO: License declaration</license>

  <buildtool_depend>ament_cmake</buildtool_depend>

  <depend>rclcpp</depend>
  <depend>std_msgs</depend>

  <test_depend>ament_lint_auto</test_depend>
  <test_depend>ament_lint_common</test_depend>

  <export>
    <build_type>ament_cmake</build_type>
  </export>
</package>
```

Developing your first node

First program

- The simplest node

```cpp
#include "rclcpp/rclcpp.hpp"

int main(int argc, char * argv[]) {
  rclcpp::init(argc, argv);

  auto node = rclcpp::Node::make_shared("simple_node");

  rclcpp::spin(node);

  rclcpp::shutdown();

  return 0;
}
```

Developing your first node

First program

- How to make a node

```
#include "rclcpp/rclcpp.hpp"

int main(int argc, char * argv[]) {
  rclcpp::init(argc, argv);

  auto node = rclcpp::Node::make_shared("simple_node");

  rclcpp::spin(node);

  rclcpp::shutdown();

  return 0;
}
```

```
1. std::shared_ptr<rclcpp::Node> node = std::shared_ptr<rclcpp::Node>(
   new rclcpp::Node("simple_node"));

2. std::shared_ptr<rclcpp::Node> node = std::make_shared<rclcpp::Node>(
   "simple_node");

3. rclcpp::Node::SharedPtr node = std::make_shared<rclcpp::Node>(
   "simple_node");

4. auto node = std::make_shared<rclcpp::Node>("simple_node");

5. auto node = rclcpp::Node::make_shared("simple_node");
```

Intelligent Robotics Lab

Developing your first node

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.5)
project(basics)

find_package(ament_cmake REQUIRED)
find_package(rclcpp REQUIRED)

set(dependencies
  rclcpp
)

add_executable(simple src/simple.cpp)
ament_target_dependencies(simple ${dependencies})

install(TARGETS
  simple
  ARCHIVE DESTINATION lib
  LIBRARY DESTINATION lib
  RUNTIME DESTINATION lib/${PROJECT_NAME}
)

if(BUILD_TESTING)
  find_package(ament_lint_auto REQUIRED)
  ament_lint_auto_find_test_dependencies()
endif()

ament_export_dependencies(${dependencies})
ament_package()
```

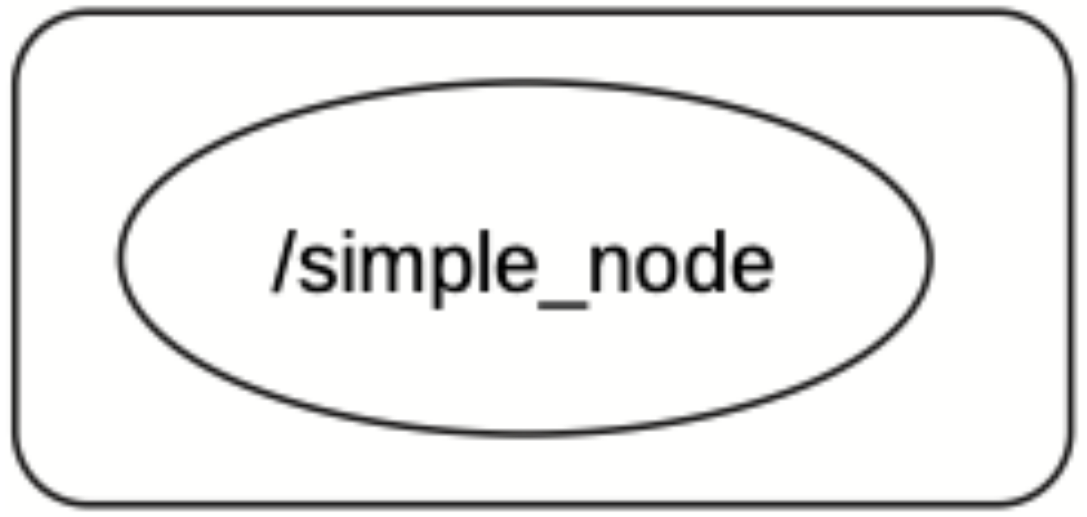
Intelligent Robotics Lab

Developing your first node

Build and execute

```
cd ~/bookros2_ws

colcon build --symlink-install
```

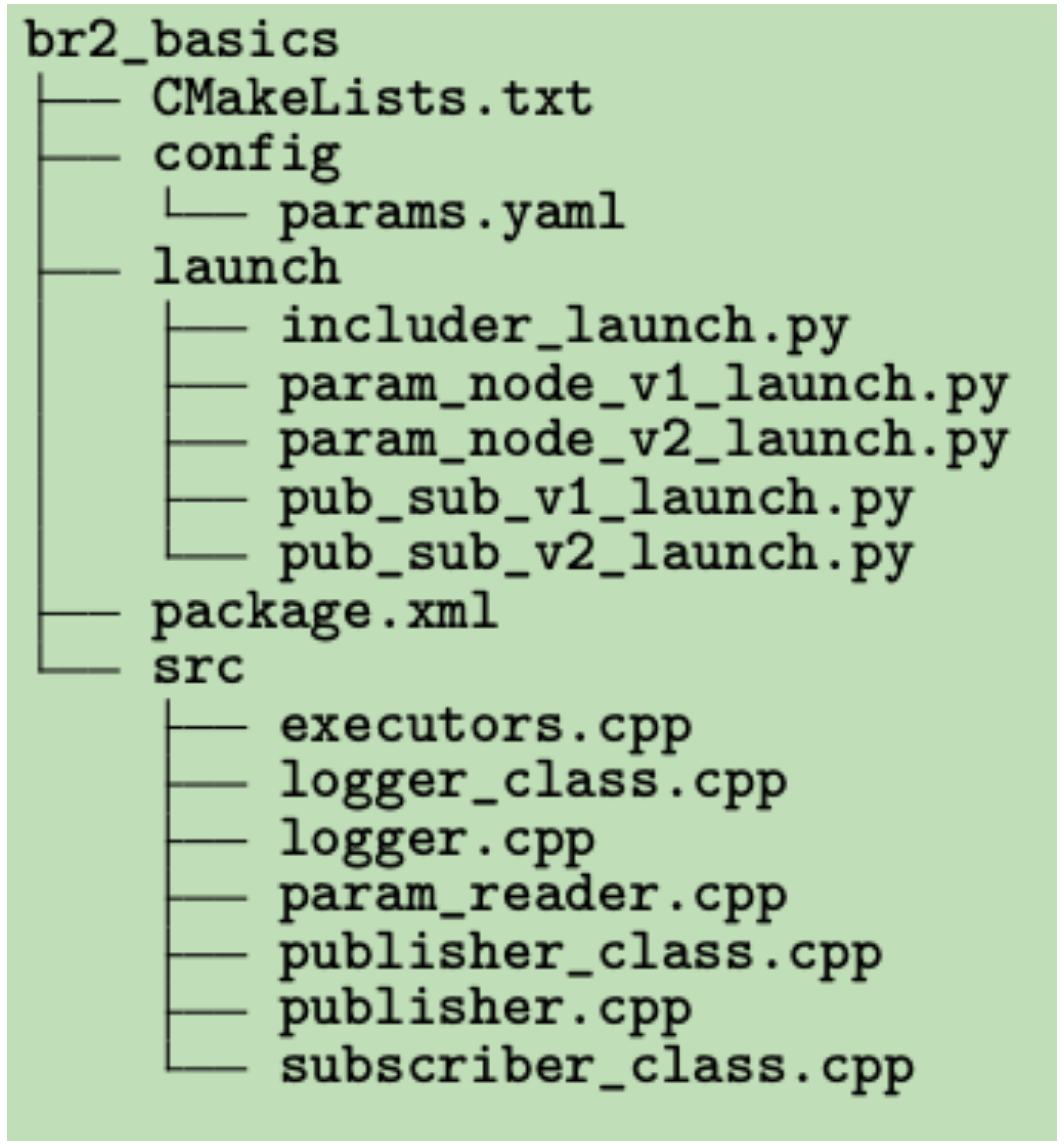
```
$ ros2 run my_package simple
```



```
$ ros2 node list

/simple_node
```

The br2_BASICS Package

Package content



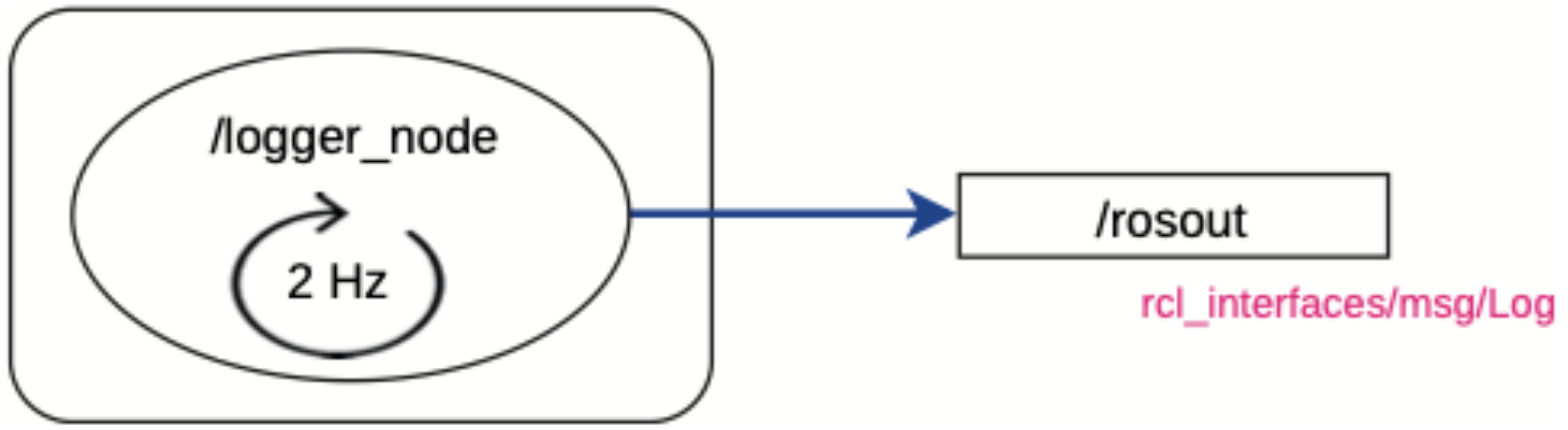
```
br2_basics
├── CMakeLists.txt
├── config
│   └── params.yaml
├── launch
│   ├── includer_launch.py
│   ├── param_node_v1_launch.py
│   ├── param_node_v2_launch.py
│   ├── pub_sub_v1_launch.py
│   └── pub_sub_v2_launch.py
├── package.xml
└── src
    ├── executors.cpp
    ├── logger_class.cpp
    ├── logger.cpp
    ├── param_reader.cpp
    ├── publisher_class.cpp
    ├── publisher.cpp
    └── subscriber_class.cpp
```

The br2_BASICS Package

logger.cpp

- Use RCLCPP_* to show messages
- Control execution frequency with rclcpp::Rate
- spin() and spin_some()

```
auto node = rclcpp::Node::make_shared("logger_node");

rclcpp::Rate loop_rate(500ms);
int counter = 0;

while (rclcpp::ok()) {
  RCLCPP_INFO(node->get_logger(), "Hello %d", counter++);

  rclcpp::spin_some(node);
  loop_rate.sleep();
}
```



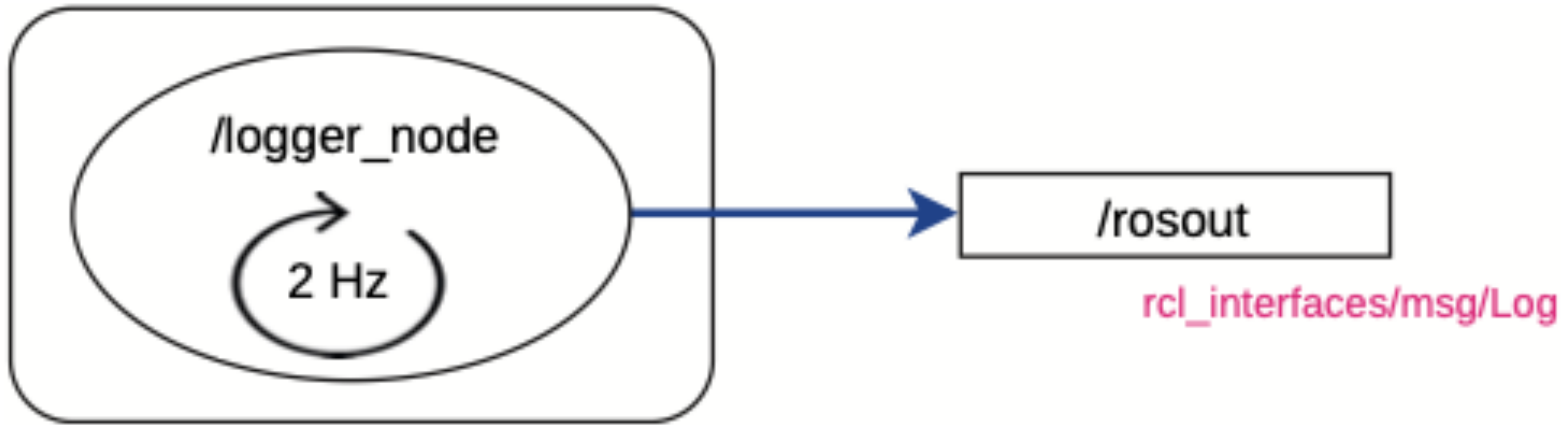
```
$ ros2 run br2_basics logger --ros-args --log-level debug
```

The br2_BASICS Package

logger.cpp

```
$ cd ~/bookros2_ws

$ colcon build --symlink-install --packages-select br2_basics
```

```
$ ros2 run br2_basics logger
[INFO] [1643264508.056814169] [logger_node]: Hello 0
[INFO] [1643264508.556910295] [logger_node]: Hello 1
...
```



The br2_BASICS Package

logger.cpp

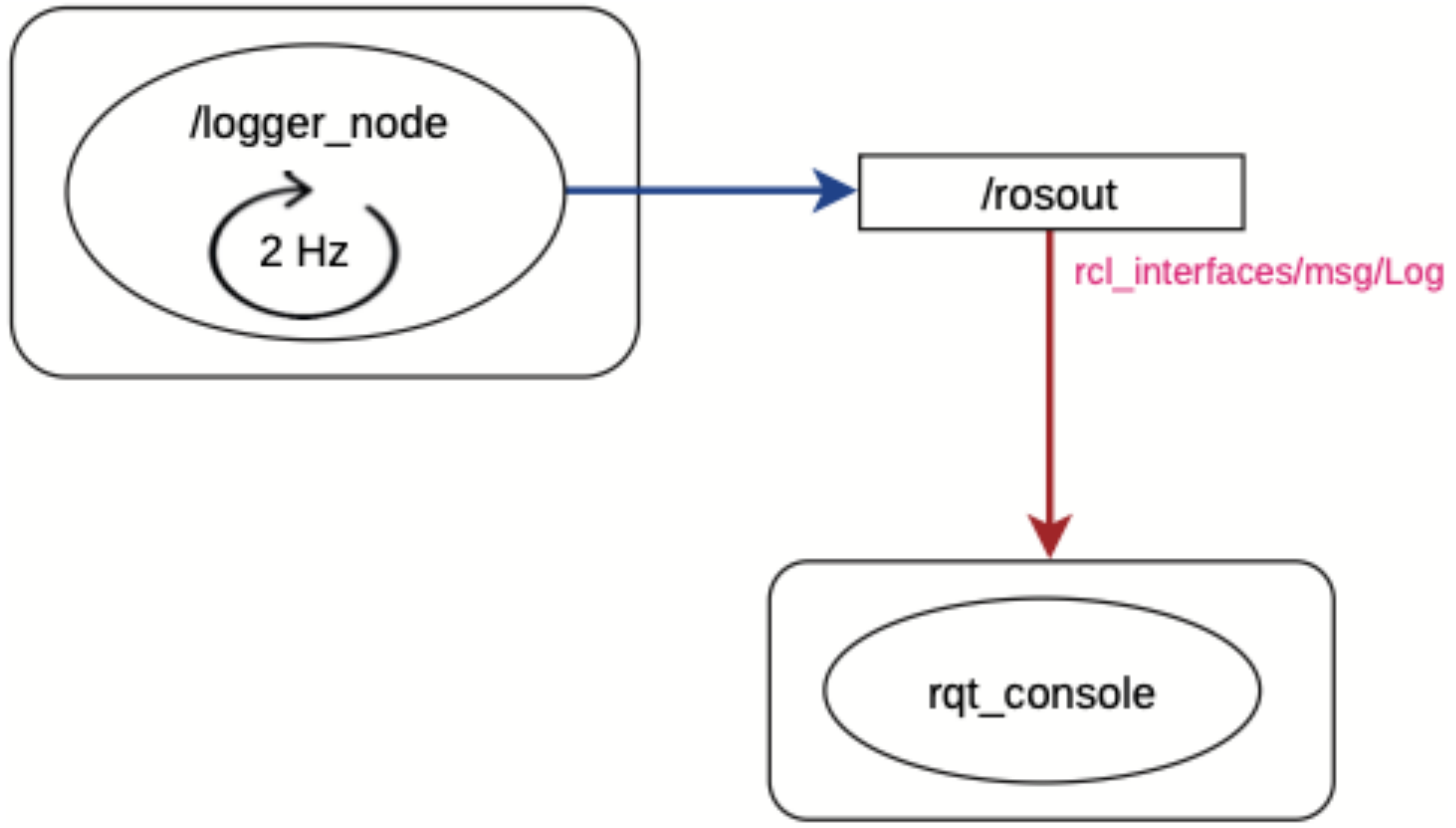
```
$ ros2 topic echo /rosout

stamp:
    sec: 1643264511
    nanosec: 556908791
level: 20
name: logger_node
msg: Hello 7
file: /home/fmrico/ros/ros2/bookros2_ws/src/book_ros2/br2_basics/src/logger.cpp
function: main
line: 27
---
stamp:
    sec: 1643264512
    nanosec: 57037520
level: 20
...
```

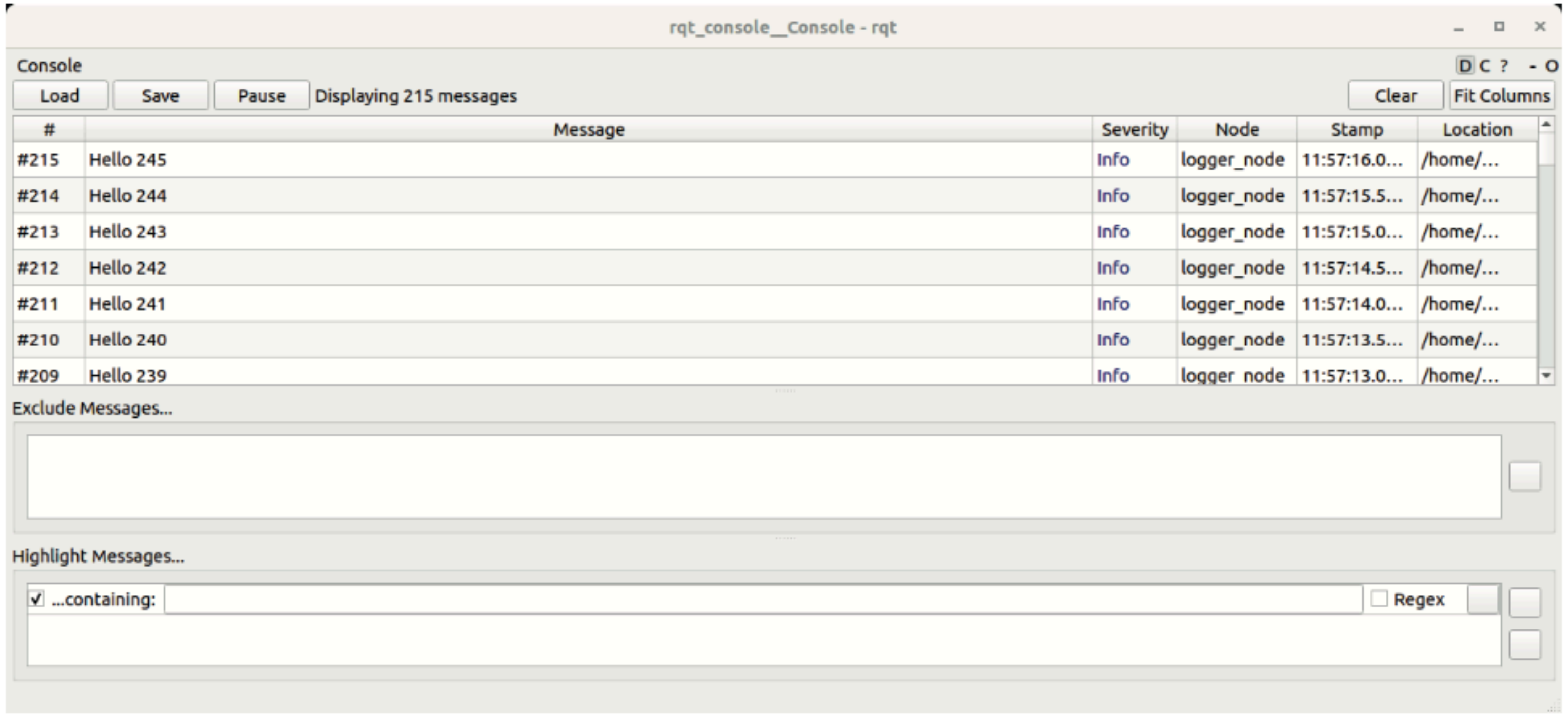
```
$ ros2 interface show rcl_interfaces/msg/Log
```

The br2_BASICS Package

RQT Console

```
$ ros2 run rqt_console rqt_console
```





Displaying 215 messages

#	Message	Severity	Node	Stamp	Location
#215	Hello 245	Info	logger_node	11:57:16.0...	/home/...
#214	Hello 244	Info	logger_node	11:57:15.5...	/home/...
#213	Hello 243	Info	logger_node	11:57:15.0...	/home/...
#212	Hello 242	Info	logger_node	11:57:14.5...	/home/...
#211	Hello 241	Info	logger_node	11:57:14.0...	/home/...
#210	Hello 240	Info	logger_node	11:57:13.5...	/home/...
#209	Hello 239	Info	logger_node	11:57:13.0...	/home/...

```
$ ros2 run br2_basics logger --ros-args --log-level debug
```

Intelligent Robotics Lab

The br2_BASICS Package

logger_class.cpp

- Inherit from rclcpp::Node helps to organize better your code
- Control execution cycle **internally** with timers

```
class LoggerNode : public rclcpp::Node
{
public:
  LoggerNode() : Node("logger_node")
  {
    counter_ = 0;
    timer_ = create_wall_timer(
      500ms, std::bind(&LoggerNode::timer_callback, this));
  }

  void timer_callback()
  {
    RCLCPP_INFO(get_logger(), "Hello %d", counter_++);
  }

private:
  rclcpp::TimerBase::SharedPtr timer_;
  int counter_;
};

int main(int argc, char * argv[]) {
  rclcpp::init(argc, argv);

  auto node = std::make_shared<LoggerNode>();

  rclcpp::spin(node);

  rclcpp::shutdown();
  return 0;
}
```

The br2_BASICS Package

logger_class.cpp

```
add_executable(logger_class src/logger.cpp)
ament_target_dependencies(logger ${dependencies})

add_executable(logger_class src/logger_class.cpp)
ament_target_dependencies(logger_class ${dependencies})

install(TARGETS
  logger
  logger_class
  ...
  ARCHIVE DESTINATION lib
  LIBRARY DESTINATION lib
  RUNTIME DESTINATION lib/${PROJECT_NAME}
)
```

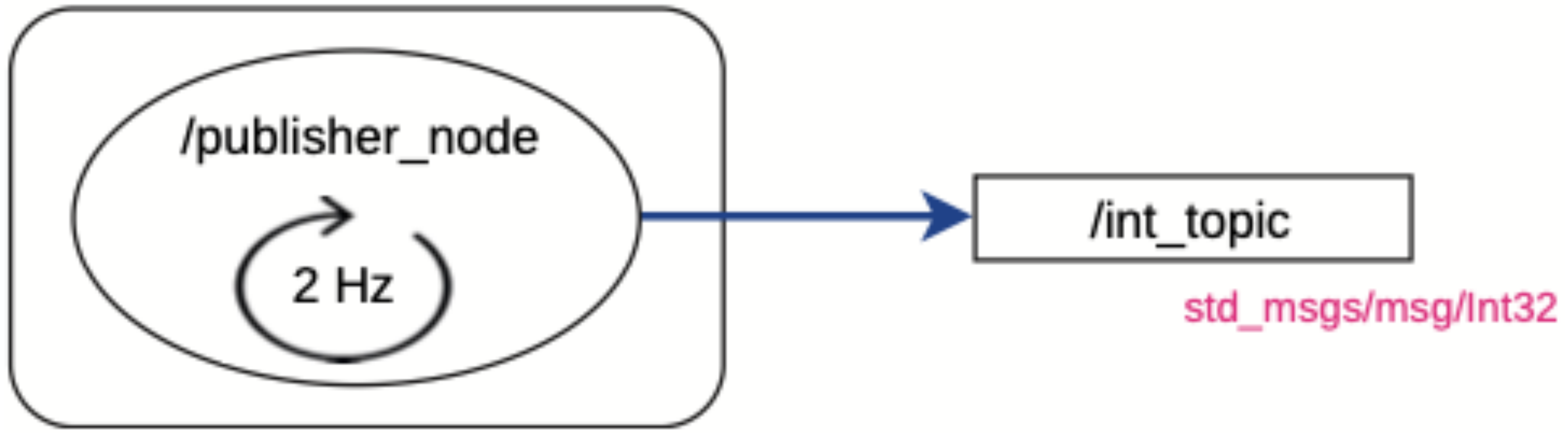
```
$ ros2 run br2_basics logger_class
```

The br2_BASICS Package

Publishing

```
class PublisherNode : public rclcpp::Node
{
public:
  PublisherNode() : Node("publisher_node")
  {
    publisher_ = create_publisher<std_msgs::msg::Int32>("int_topic", 10);
    timer_ = create_wall_timer(
      500ms, std::bind(&PublisherNode::timer_callback, this));
  }

  void timer_callback()
  {
    message_.data += 1;
    publisher_->publish(message_);
  }

private:
  rclcpp::Publisher<std_msgs::msg::Int32>::SharedPtr publisher_;
  rclcpp::TimerBase::SharedPtr timer_;
  std_msgs::msg::Int32 message_;
};
```

```
// For std_msgs/msg/Int32
#include "std_msgs/msg/int32.hpp"

std_msgs::msg::Int32 msg_int32;


// For sensor_msgs/msg/LaserScan
#include "sensor_msgs/msg/laser_scan.hpp"

sensor_msgs::msg::LaserScan msg_laserscan;
```



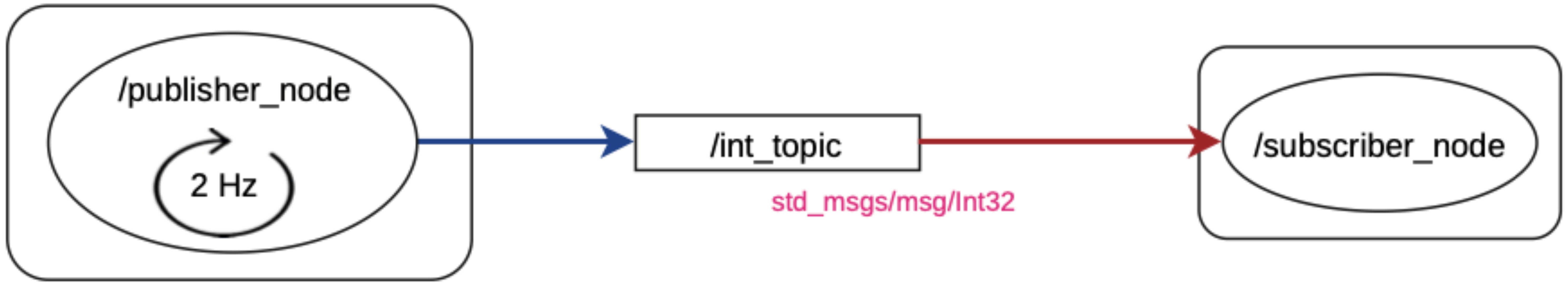
```
$ ros2 run br2_basics publisher_class
```

The br2_BASICS Package

Subscribing

```
class SubscriberNode : public rclcpp::Node
{
public:
  SubscriberNode() : Node("subscriber_node")
  {
    subscriber_ = create_subscription<std_msgs::msg::Int32>("int_topic", 10,
      std::bind(&SubscriberNode::callback, this, _1));
  }

  void callback(const std_msgs::msg::Int32::SharedPtr msg)
  {
    RCLCPP_INFO(get_logger(), "Hello %d", msg->data);
  }

private:
  rclcpp::Subscription<std_msgs::msg::Int32>::SharedPtr subscriber_;
};
```

```
$ ros2 run br2_basics subscriber_class
```



Francisco Martín Rico francisco.rico@urjc.es @fmrico

The br2_BASICS Package

About QoS

Default	Reliable	Volatile	Keep Last
Services	Reliable	Volatile	Normal Queue
Sensor	Best Effort	Volatile	Small Queue
DParameters	Reliable	Volatile	Large Queue

```
publisher = node->create_publisher<std_msgs::msg::String>(
  "chatter", rclcpp::QoS(100).transient_local().best_effort());
```

```
publisher_ = create_publisher<sensor_msgs::msg::LaserScan>(
  "scan", rclcpp::SensorDataQoS().reliable());
```

Compatibility of QoS **durability** profiles		**Subscriber**	
		Volatile	**Transient Local**
Publisher	**Volatile**	Volatile	No Connection
	Transient Local	Volatile	Transient Local

Compatibility of QoS **reliability** profiles		**Subscriber**	
		Best Effort	**Reliable**
Publisher	**Best Effort**	Best Effort	No Connection
	Reliable	Best Effort	Reliable

The br2_BASICS Package

Launchers

- Declaratives
- Alternatives: xml and yaml

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
  pub_cmd = Node(
    package='basics',
    executable='publisher',
    output='screen'
  )

  sub_cmd = Node(
    package='basics',
    executable='subscriber_class',
    output='screen'
  )

  ld = LaunchDescription()
  ld.add_action(pub_cmd)
  ld.add_action(sub_cmd)

  return ld
```

```
install(DIRECTORY launch DESTINATION share/${PROJECT_NAME})
```

```
$ ros2 launch br2_basics pub_sub_v2_launch.py
```

The br2_BASICS Package

Parameters

- Use parameters for configure node's behavior
- Declare parameters and get their values

```
class LocalizationNode : public rclcpp::Node
{
public:
  LocalizationNode() : Node("localization_node")
  {
    declare_parameter<int>("number_particles", 200);
    declare_parameter<std::vector<std::string>>("topics", {});
    declare_parameter<std::vector<std::string>>("topic_types", {});

    get_parameter("number_particles", num_particles_);
    RCLCPP_INFO_STREAM(get_logger(), "Number of particles: " << num_particles_);

    get_parameter("topics", topics_);
    get_parameter("topic_types", topic_types_);

    if (topics_.size() != topic_types_.size()) {
      RCLCPP_ERROR(get_logger(), "Number of topics (%zu) != number of types (%zu)",
        topics_.size(), topic_types_.size());
    } else {
      RCLCPP_INFO_STREAM(get_logger(), "Number of topics: " << topics_.size());
      for (size_t i = 0; i < topics_.size(); i++) {
        RCLCPP_INFO_STREAM(
          get_logger(),
          "\t" << topics_[i] << "\t - " << topic_types_[i]);
      }
    }
}

private:
  int num_particles_;
  std::vector<std::string> topics_;
  std::vector<std::string> topic_types_;
};
```

The br2_BASICS Package

Parameters

- Use parameters for configure node's behavior
- Declare parameters and get their values

```
$ ros2 run br2_basics param_reader
```

```
$ ros2 run br2_basics param_reader --ros-args -p number_particles:=300
```

```
$ ros2 run br2_basics param_reader --ros-args -p number_particles:=300
-p topics:= '[scan, image]' -p topic_types:='[sensor_msgs/msg/LaserScan,
sensor_msgs/msg/Image]'
```

```python
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    param_reader_cmd = Node(
        package='basics',
        executable='param_reader',
        parameters=[{
            'particles': 300,
            'topics': ['scan', 'image'],
            'topic_types': ['sensor_msgs/msg/LaserScan', 'sensor_msgs/msg/Image']
        }],
        output='screen'
    )

    ld = LaunchDescription()
    ld.add_action(param_reader_cmd)

    return ld
```

The br2_BASICS Package

Parameters

- Use parameters for configure node's behavior
- Declare parameters and get their values

config/params.yaml

```yaml
localization_node:
  ros__parameters:
    number_particles: 300
    topics: [scan, image]
    topic_types: [sensor_msgs/msg/LaserScan, sensor_msgs/msg/Image]
```

```
$ ros2 run br2_basics param_reader --ros-args --params-file
install/basics/share/basics/config/params.yaml
```

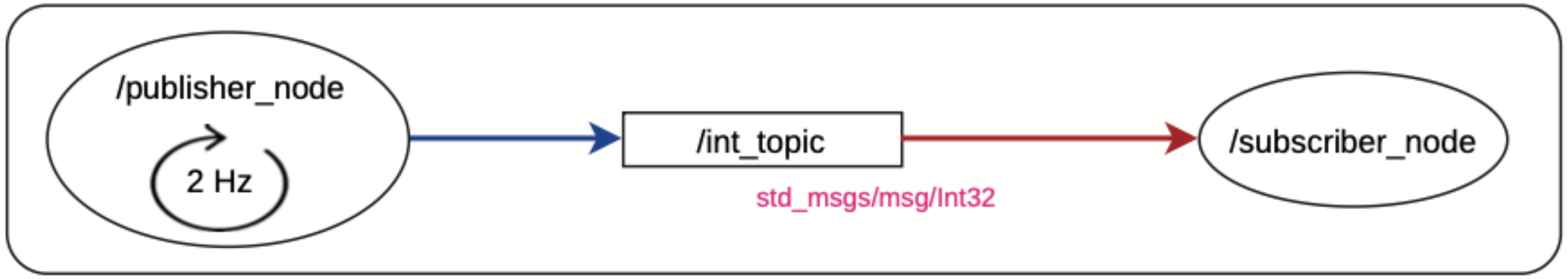
```python
def generate_launch_description():
    ...
    param_reader_cmd = Node(
        package='basics',
        executable='param_reader',
        parameters=[param_file],
        output='screen'
    )
```

The br2_BASICS Package

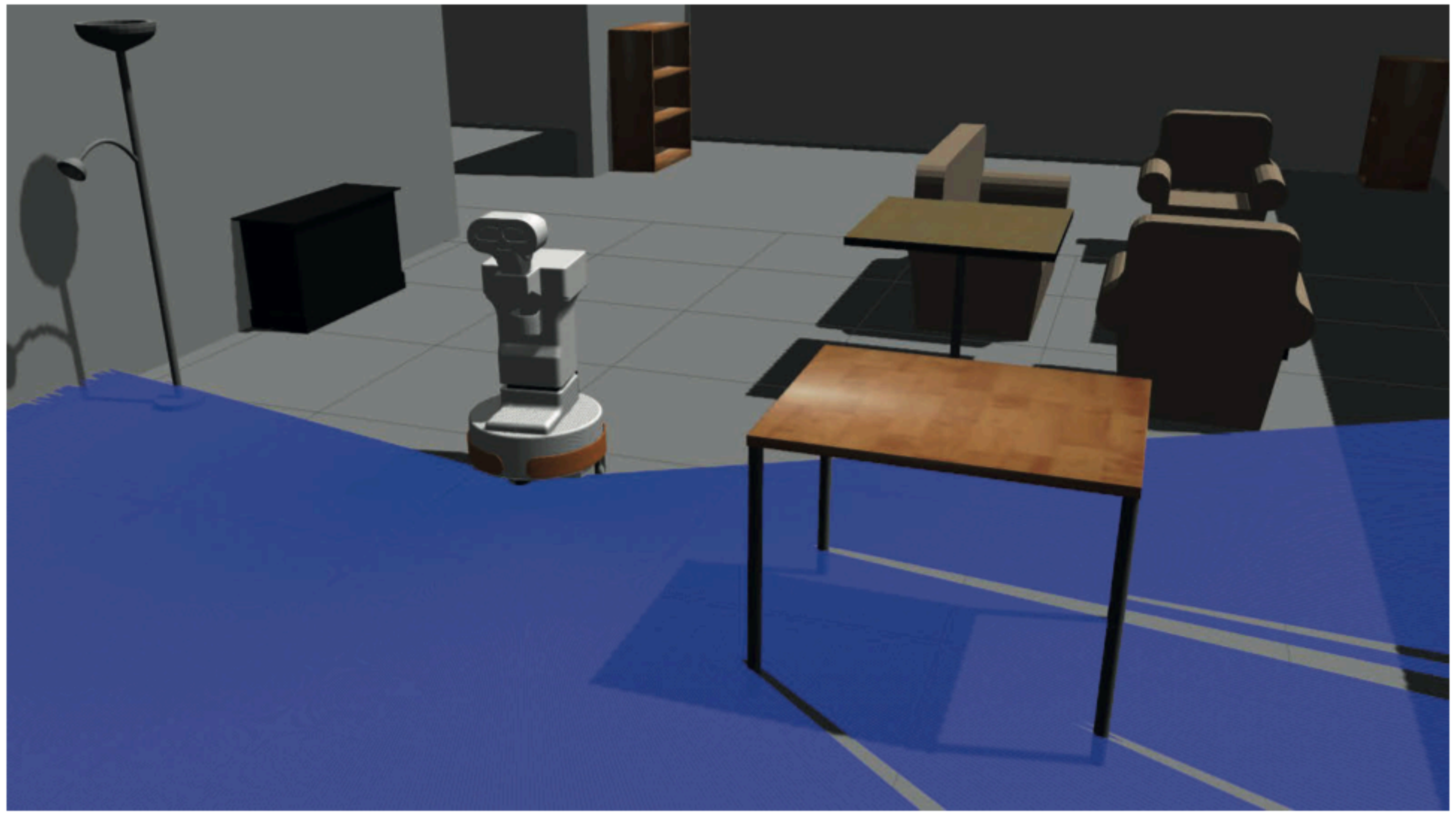
Executors

```
int main(int argc, char * argv[]) {
  rclcpp::init(argc, argv);

  auto node_pub = std::make_shared<PublisherNode>();
  auto node_sub = std::make_shared<SubscriberNode>();

  rclcpp::executors::SingleThreadedExecutor executor;

  executor.add_node(node_pub);
  executor.add_node(node_sub);

  executor.spin();

  rclcpp::shutdown();
  return 0;
}
```

```
auto node_pub = std::make_shared<PublisherNode>();
auto node_sub = std::make_shared<SubscriberNode>();

rclcpp::executors::MultiThreadedExecutor executor(
      rclcpp::executor::ExecutorArgs(), 8);

executor.add_node(node_pub);
executor.add_node(node_sub);

executor.spin();
}
```



Simulated Robot Setup



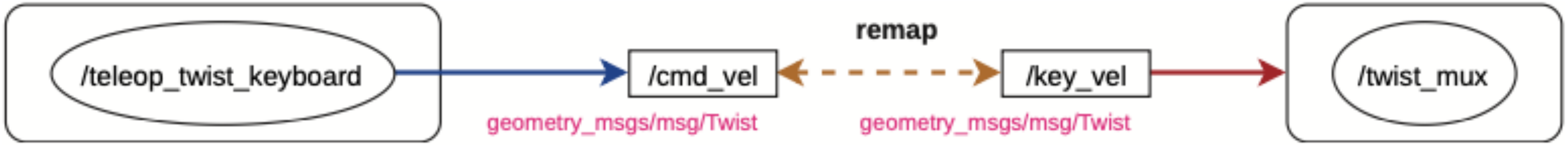
```
$ ros2 launch br2_tiago sim.launch.py world:=factory
$ ros2 launch br2_tiago sim.launch.py world:=featured
$ ros2 launch br2_tiago sim.launch.py world:=pal_office
$ ros2 launch br2_tiago sim.launch.py world:=small_factory
$ ros2 launch br2_tiago sim.launch.py world:=small_office
$ ros2 launch br2_tiago sim.launch.py world:=willow_garage
```

Simulated Robot Setup

Topics and Remaps

```
$ ros2 topic list
```

```
$ ros2 run teleop_twist_keyboard teleop_twist_keyboard --ros-args -r
cmd_vel:=key_vel
```

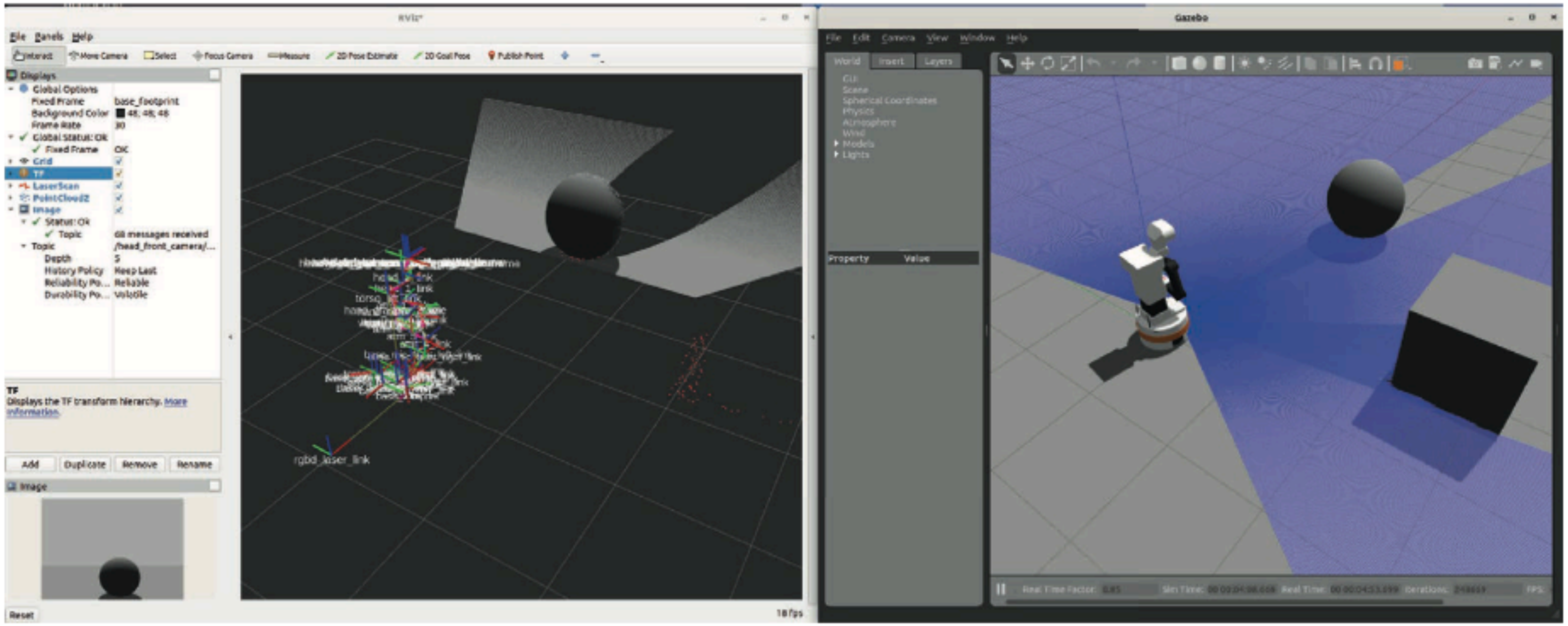


```
$ ros2 topic echo --no-arr /scan_raw

$ ros2 topic echo --no-arr /head_front_camera/rgb/image_raw
```

Simulated Robot Setup

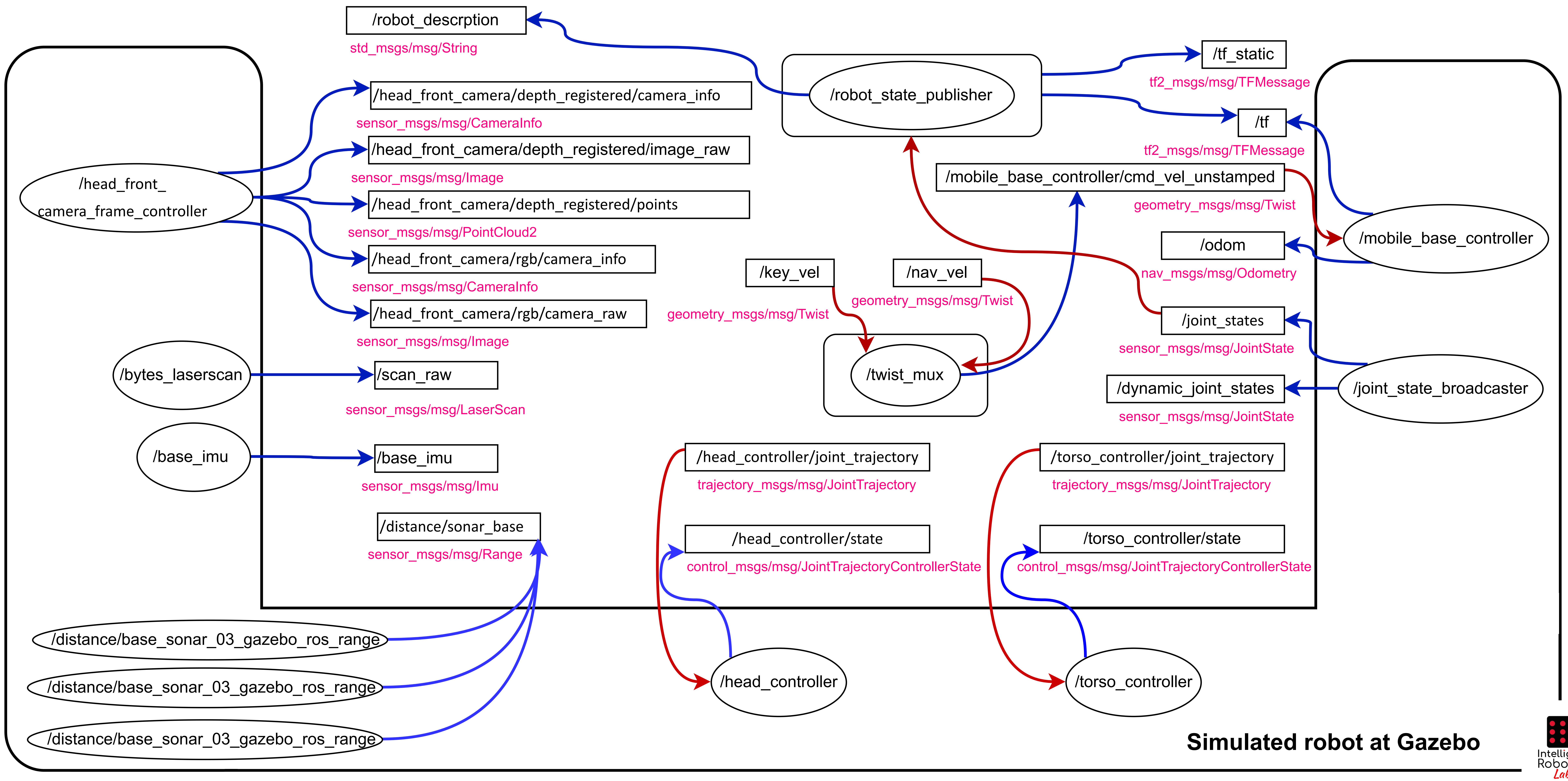
Rviz2

```
$ ros2 run rviz2 rviz2
```



Global Options	
Fixed Frame	base_footprint
Background Color	■ 48; 48; 48
Frame Rate	30
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	✓
TF	✓
LaserScan	✓
PointCloud2	✓
Image	✓
✓ Status: Ok	
✓ Topic	68 messages received
Topic	/head_front_camera/...
Depth	5
History Policy	Keep Last
Reliability Po...	Reliable
Durability Po...	Volatile

TF
Displays the TF transform hierarchy. More information.


/robot_descrption
std_msgs/msg/String
/head_front_camera/depth_registered/camera_info
sensor_msgs/msg/CameraInfo
/head_front_camera/depth_registered/image_raw
sensor_msgs/msg/Image
/head_front_camera/depth_registered/points
sensor_msgs/msg/PointCloud2
/head_front_camera/rgb/camera_info
sensor_msgs/msg/CameraInfo
/head_front_camera/rgb/camera_raw
sensor_msgs/msg/Image
/head_front_ camera_frame_controller
/bytes_laserscan
/scan_raw
sensor_msgs/msg/LaserScan
/base_imu
/base_imu
sensor_msgs/msg/Imu
/distance/sonar_base
sensor_msgs/msg/Range
/distance/base_sonar_03_gazebo_ros_range
/distance/base_sonar_03_gazebo_ros_range
/distance/base_sonar_03_gazebo_ros_range
/robot_state_publisher
/tf_static
tf2_msgs/msg/TFMessage
/tf
tf2_msgs/msg/TFMessage
/mobile_base_controller/cmd_vel_unstamped
geometry_msgs/msg/Twist
/mobile_base_controller
/odom
nav_msgs/msg/Odometry
/key_vel
/nav_vel
geometry_msgs/msg/Twist
geometry_msgs/msg/Twist
/twist_mux
/joint_states
sensor_msgs/msg/JointState
/dynamic_joint_states
sensor_msgs/msg/JointState
/joint_state_broadcaster
/head_controller/joint_trajectory
trajectory_msgs/msg/JointTrajectory
/head_controller/state
control_msgs/msg/JointTrajectoryControllerState
/head_controller
/torso_controller/joint_trajectory
trajectory_msgs/msg/JointTrajectory
/torso_controller/state
control_msgs/msg/JointTrajectoryControllerState
/torso_controller
Simulated robot at Gazebo