**ROS2 Package Guideline**

https://docs.ros.org/en/foxy/Tutorials/Configuring-ROS2-Environment.html

**Install Turtlesim**

sudo apt update

sudo apt install ros-foxy-turtlesim

**Execute Package**

ros2 pkg executables turtlesim

turtlesim draw\_square

turtlesim mimic

turtlesim turtle\_teleop\_key

turtlesim turtlesim\_node

**Run Turtlesim**

ros2 run turtlesim turtlesim\_node

[INFO] [turtlesim]: Starting turtlesim with node name /turtlesim

[INFO] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]

**Use Turtlesim**

Open new terminal

ros2 run turtlesim turtle\_teleop\_key

**Ros2 List**

ros2 node list

/teleop\_turtle

/turtlesim

ros2 topic list

/parameter\_events

/rosout

/turtle1/cmd\_vel

/turtle1/color\_sensor

/turtle1/pose

ros2 action list

/turtle1/rotate\_absolute

ros2 service list

No.

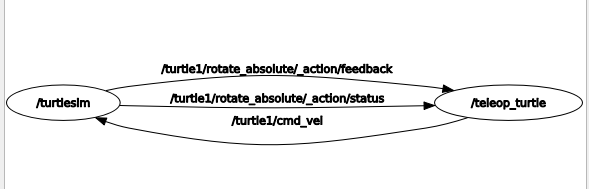
**Install Rqt**

sudo apt update

sudo apt install ~nros-foxy-rqt\*

**Rqt Graph**

rqt



**Plugins** > **Services** > **Service Caller**

Change Service => /spawn

/spawn turtlesim/srv/Spawn

X float 0.0

Y float 0.0

Theta float 0.0

Name string ‘’

Change Name string ‘turtle2’

ros2 service list

/clear

/kill

/reset

/rqt\_gui\_py\_node\_443613/describe\_parameters

/rqt\_gui\_py\_node\_443613/get\_parameter\_types

/rqt\_gui\_py\_node\_443613/get\_parameters

/rqt\_gui\_py\_node\_443613/list\_parameters

/rqt\_gui\_py\_node\_443613/set\_parameters

/rqt\_gui\_py\_node\_443613/set\_parameters\_atomically

/spawn

/teleop\_turtle/describe\_parameters

/teleop\_turtle/get\_parameter\_types

/teleop\_turtle/get\_parameters

/teleop\_turtle/list\_parameters

/teleop\_turtle/set\_parameters

/teleop\_turtle/set\_parameters\_atomically

/turtle1/set\_pen

/turtle1/teleport\_absolute

/turtle1/teleport\_relative

/turtlesim/describe\_parameters

/turtlesim/get\_parameter\_types

/turtlesim/get\_parameters

/turtlesim/list\_parameters

/turtlesim/set\_parameters

/turtlesim/set\_parameters\_atomically

Remapping

ros2 run turtlesim turtle\_teleop\_key --ros-args --remap turtle1/cmd\_vel:=turtle2/cmd\_vel

Become two tortoise [Tortoise 1 and Tortoise2]

---------------------------------------------------------------------------------------------

**Run Ros2 Node**

ros2 run <package\_name> <executable\_name>

ros2 run turtlesim turtlesim\_node

Package name = turtlesim

Executable name = turtlesim\_node

**Node List**

ros2 run turtlesim turtlesim\_node

ros2 node list

/turtlesim

ros2 run turtlesim turtle\_teleop\_key

ros2 node list

/teleop\_turtle

/turtlesim

ros2 run turtlesim turtlesim\_node --ros-args --remap \_\_node:=my\_turtle

ros2 node list

/my\_turtle

/teleop\_turtle

/turtlesim

**Ros2 Node Info**

ros2 node info <node\_name>

ros2 node info /my\_turtle

/my\_turtle

Subscribers:

/parameter\_events: rcl\_interfaces/msg/ParameterEvent

/turtle1/cmd\_vel: geometry\_msgs/msg/Twist

Publishers:

/parameter\_events: rcl\_interfaces/msg/ParameterEvent

/rosout: rcl\_interfaces/msg/Log

/turtle1/color\_sensor: turtlesim/msg/Color

/turtle1/pose: turtlesim/msg/Pose

Service Servers:

/clear: std\_srvs/srv/Empty

/kill: turtlesim/srv/Kill

/my\_turtle/describe\_parameters: rcl\_interfaces/srv/DescribeParameters

/my\_turtle/get\_parameter\_types: rcl\_interfaces/srv/GetParameterTypes

/my\_turtle/get\_parameters: rcl\_interfaces/srv/GetParameters

/my\_turtle/list\_parameters: rcl\_interfaces/srv/ListParameters

/my\_turtle/set\_parameters: rcl\_interfaces/srv/SetParameters

/my\_turtle/set\_parameters\_atomically: rcl\_interfaces/srv/SetParametersAtomically

/reset: std\_srvs/srv/Empty

/spawn: turtlesim/srv/Spawn

/turtle1/set\_pen: turtlesim/srv/SetPen

/turtle1/teleport\_absolute: turtlesim/srv/TeleportAbsolute

/turtle1/teleport\_relative: turtlesim/srv/TeleportRelative

Service Clients:

Action Servers:

/turtle1/rotate\_absolute: turtlesim/action/RotateAbsolute

Action Clients:

ros2 node info /teleop\_turtle

/teleop\_turtle

Subscribers:

/parameter\_events: rcl\_interfaces/msg/ParameterEvent

Publishers:

/parameter\_events: rcl\_interfaces/msg/ParameterEvent

/rosout: rcl\_interfaces/msg/Log

/turtle1/cmd\_vel: geometry\_msgs/msg/Twist

Service Servers:

/teleop\_turtle/describe\_parameters: rcl\_interfaces/srv/DescribeParameters

/teleop\_turtle/get\_parameter\_types: rcl\_interfaces/srv/GetParameterTypes

/teleop\_turtle/get\_parameters: rcl\_interfaces/srv/GetParameters

/teleop\_turtle/list\_parameters: rcl\_interfaces/srv/ListParameters

/teleop\_turtle/set\_parameters: rcl\_interfaces/srv/SetParameters

/teleop\_turtle/set\_parameters\_atomically: rcl\_interfaces/srv/SetParametersAtomically

Service Clients:

Action Servers:

Action Clients:

/turtle1/rotate\_absolute: turtlesim/action/RotateAbsolute

ros2 node info /turtlesim

/turtlesim

Subscribers:

/parameter\_events: rcl\_interfaces/msg/ParameterEvent

/turtle1/cmd\_vel: geometry\_msgs/msg/Twist

Publishers:

/parameter\_events: rcl\_interfaces/msg/ParameterEvent

/rosout: rcl\_interfaces/msg/Log

/turtle1/color\_sensor: turtlesim/msg/Color

/turtle1/pose: turtlesim/msg/Pose

Service Servers:

/clear: std\_srvs/srv/Empty

/kill: turtlesim/srv/Kill

/reset: std\_srvs/srv/Empty

/spawn: turtlesim/srv/Spawn

/turtle1/set\_pen: turtlesim/srv/SetPen

/turtle1/teleport\_absolute: turtlesim/srv/TeleportAbsolute

/turtle1/teleport\_relative: turtlesim/srv/TeleportRelative

/turtlesim/describe\_parameters: rcl\_interfaces/srv/DescribeParameters

/turtlesim/get\_parameter\_types: rcl\_interfaces/srv/GetParameterTypes

/turtlesim/get\_parameters: rcl\_interfaces/srv/GetParameters

/turtlesim/list\_parameters: rcl\_interfaces/srv/ListParameters

/turtlesim/set\_parameters: rcl\_interfaces/srv/SetParameters

/turtlesim/set\_parameters\_atomically: rcl\_interfaces/srv/SetParametersAtomically

Service Clients:

Action Servers:

/turtle1/rotate\_absolute: turtlesim/action/RotateAbsolute

Action Clients:

**ROS2 Topic**

New Terminal

ros2 run turtlesim turtlesim\_node

New Terminal

ros2 run turtlesim turtle\_teleop\_key

New Terminal

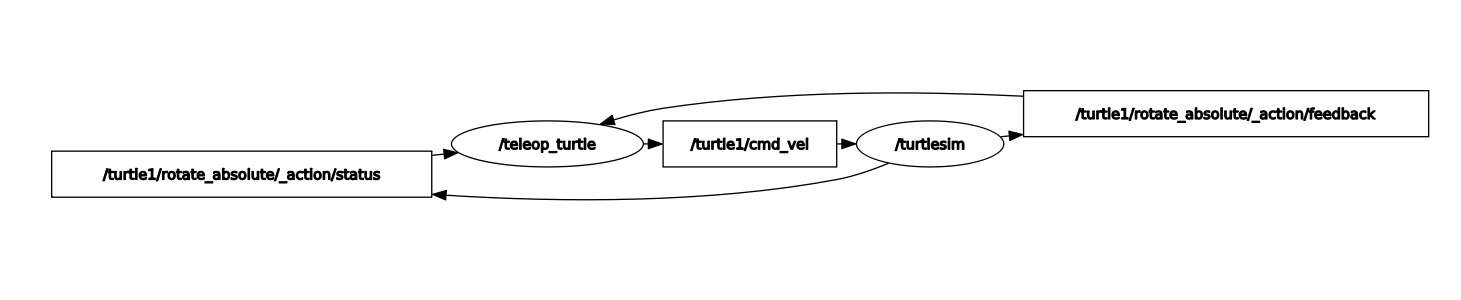
rqt\_graph

Node graph

Node topic active

Refresh

Show the graph



**/node --------(topic)--------/node**

/teleop\_turtle ------- /turtle1/cmd\_vel ----- /turtlesim

/teleop\_turtle = publisher to the topic

/turtle1/cmd\_vel = topic

/turtlesim = subscriber(subscribed to that topic to receive the data)

**ROS2 Topic List**

ros2 topic list

/parameter\_events

/rosout

/turtle1/cmd\_vel

/turtle1/color\_sensor

/turtle1/pose

**ROS2 Topic List with Topic Type**

**[Topic Name + Topic Type]**

ros2 topic list -t

/parameter\_events [rcl\_interfaces/msg/ParameterEvent]

/rosout [rcl\_interfaces/msg/Log]

/turtle1/cmd\_vel [geometry\_msgs/msg/Twist]

/turtle1/color\_sensor [turtlesim/msg/Color]

/turtle1/pose [turtlesim/msg/Pose]

**Topic Type**

In the package geometry\_msgs, there is a msg called Twist.

**ROS2 Topic Echo**

To see the data being published on a topic.

ros2 topic echo <topic\_name>

**/node --------(topic)--------/node**

/teleop\_turtle ------- /turtle1/cmd\_vel ----- /turtlesim

ros2 topic echo /turtle1/cmd\_vel

When turtose spawn, it will show the data.It includes topic interface(arguments).

linear:

x: 2.0

y: 0.0

z: 0.0

angular:

x: 0.0

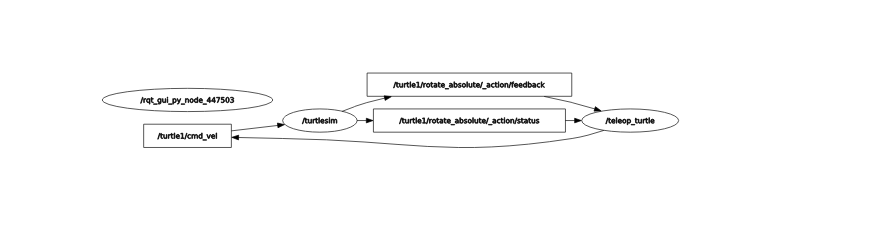
y: 0.0

z: 0.0

---

Rqt\_graph

Remove debug



**ROS2 Topic Info**

**ros2 topic info <Topic\_name>**

ros2 topic info /turtle1/cmd\_vel

It will show Topic Type

Type: geometry\_msgs/msg/Twist

Publisher count: 0

Subscription count: 1

**ROS2 Interface Show(show message)**

Nodes send data over topics using messages.

Publishers and subscribers must send and receive the same type of message to communicate.

**ros2 interface show <Topic Type>**

ros2 interface show geometry\_msgs/msg/Twist

# This expresses velocity in free space broken into its linear and angular parts.

Vector3 linear

Vector3 angular

**ROS2 topic pub**

ros2 topic pub <topic\_name> <msg\_type> '<args>'

Topic type = message type

The '<args>' argument is the actual data you’ll pass to the topic.

Args = “{arg1: {x: , y: , z: }, angular: {x: , y: , z: }}”

--once is an optional argument meaning “publish one message then exit”.

--rate = ros2 topic pub to publish the command in a steady stream at 1 Hz.

ros2 topic pub --once /turtle1/cmd\_vel geometry\_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 1.8}}"

publisher: beginning loop

publishing #1: geometry\_msgs.msg.Twist(linear=geometry\_msgs.msg.Vector3(x=2.0, y=0.0, z=0.0), angular=geometry\_msgs.msg.Vector3(x=0.0, y=0.0, z=1.8))

publisher: beginning loop

publishing #1: geometry\_msgs.msg.Twist(linear=geometry\_msgs.msg.Vector3(x=2.0, y=0.0, z=0.0), angular=geometry\_msgs.msg.Vector3(x=0.0, y=0.0, z=1.8))

ros2 topic pub --rate 1 /turtle1/cmd\_vel geometry\_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 1.8}}"

publisher: beginning loop

publishing #1: geometry\_msgs.msg.Twist(linear=geometry\_msgs.msg.Vector3(x=2.0, y=0.0, z=0.0), angular=geometry\_msgs.msg.Vector3(x=0.0, y=0.0, z=1.8))

publishing #2: geometry\_msgs.msg.Twist(linear=geometry\_msgs.msg.Vector3(x=2.0, y=0.0, z=0.0), angular=geometry\_msgs.msg.Vector3(x=0.0, y=0.0, z=1.8))

ros2 topic hz /turtle1/pose

average rate: 62.471

min: 0.015s max: 0.016s std dev: 0.00041s window: 65

ros2 topic echo /turtle1/pose

x: 7.560444355010986

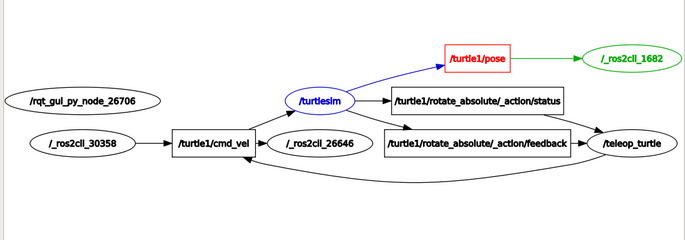
y: 5.544444561004639

theta: -0.2351837158203125

linear\_velocity: 0.0

angular\_velocity: 0.0

locity: 0.0



/turtlesim is now publishing to the pose topic, and a new echo node is subscribed.

ROS2 Services

New Terminal

ros2 run turtlesim turtlesim\_node

New Terminal

ros2 run turtlesim turtle\_teleop\_key

ros2 service list

/clear

/kill

/reset

/spawn

/teleop\_turtle/describe\_parameters

/teleop\_turtle/get\_parameter\_types

/teleop\_turtle/get\_parameters

/teleop\_turtle/list\_parameters

/teleop\_turtle/set\_parameters

/teleop\_turtle/set\_parameters\_atomically

/turtle1/set\_pen

/turtle1/teleport\_absolute

/turtle1/teleport\_relative

/turtlesim/describe\_parameters

/turtlesim/get\_parameter\_types

/turtlesim/get\_parameters

/turtlesim/list\_parameters

/turtlesim/set\_parameters

/turtlesim/set\_parameters\_atomically

Focus on :

/clear, /kill, /reset, /spawn, /turtle1/set\_pen, /turtle1/teleport\_absolute, and /turtle1/teleport\_relative

ROS2 Service Type

ros2 service type <service\_name>

ros2 service type /clear

std\_srvs/srv/Empty

ros2 service list -t

[Service Name + Service Type]

/clear [std\_srvs/srv/Empty]

/kill [turtlesim/srv/Kill]

/reset [std\_srvs/srv/Empty]

/spawn [turtlesim/srv/Spawn]

/teleop\_turtle/describe\_parameters [rcl\_interfaces/srv/DescribeParameters]

/teleop\_turtle/get\_parameter\_types [rcl\_interfaces/srv/GetParameterTypes]

/teleop\_turtle/get\_parameters [rcl\_interfaces/srv/GetParameters]

/teleop\_turtle/list\_parameters [rcl\_interfaces/srv/ListParameters]

/teleop\_turtle/set\_parameters [rcl\_interfaces/srv/SetParameters]

/teleop\_turtle/set\_parameters\_atomically [rcl\_interfaces/srv/SetParametersAtomically]

/turtle1/set\_pen [turtlesim/srv/SetPen]

/turtle1/teleport\_absolute [turtlesim/srv/TeleportAbsolute]

/turtle1/teleport\_relative [turtlesim/srv/TeleportRelative]

/turtlesim/describe\_parameters [rcl\_interfaces/srv/DescribeParameters]

/turtlesim/get\_parameter\_types [rcl\_interfaces/srv/GetParameterTypes]

/turtlesim/get\_parameters [rcl\_interfaces/srv/GetParameters]

/turtlesim/list\_parameters [rcl\_interfaces/srv/ListParameters]

/turtlesim/set\_parameters [rcl\_interfaces/srv/SetParameters]

/turtlesim/set\_parameters\_atomically [rcl\_interfaces/srv/SetParametersAtomically]

Ros2 Service Find

ros2 service find <Service\_Type>

ros2 service find std\_srvs/srv/Empty

/clear

/reset

ROS2 Interface Show

ros2 interface show <Service\_Type>.srv

ros2 interface show std\_srvs/srv/Empty.srv

---

ros2 interface show turtlesim/srv/Spawn

float32 x

float32 y

float32 theta

string name # Optional. A unique name will be created and returned if this is empty

---

string name

ROS2 Service Call

ros2 service call <service\_name> <service\_type> <arguments>

<arguments> part is optional.

ros2 service call /clear std\_srvs/srv/Empty

ros2 service call /spawn turtlesim/srv/Spawn "{x: 2, y: 2, theta: 0.2, name: ''}"

Parameter

Parameter is the arguments which is included in the topic and service.

New Terminal

ros2 run turtlesim turtlesim\_node

New Terminal

ros2 run turtlesim turtle\_teleop\_key

ROS2 Param List

ros2 param list

/teleop\_turtle:

scale\_angular

scale\_linear

use\_sim\_time

/turtlesim:

background\_b

background\_g

background\_r

use\_sim\_time

ROS2 Param Get

ros2 param get <node\_name> <parameter\_name>

ros2 param get /turtlesim background\_g

Integer value is: 86

ROS2 Param Set

ros2 param set <node\_name> <parameter\_name> <value>

ros2 param set /turtlesim background\_r 150

Set parameter successful

ROS2 Param Dump

ros2 param dump <node\_name>

ros2 param dump /turtlesim

Saving to: ./turtlesim.yaml

Load parameter file

To start the same node using your saved parameter values,

ros2 run <package\_name> <executable\_name> --ros-args --params-file <file\_name>

ros2 run turtlesim turtlesim\_node --ros-args --params-file ./turtlesim.yaml

Action

New Terminal

ros2 run turtlesim turtlesim\_node

[INFO] [1617174253.788979796] [turtlesim]: Starting turtlesim with node name /turtlesim

[INFO] [1617174253.791558242] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]

New Terminal

ros2 run turtlesim turtle\_teleop\_key

Reading from keyboard

---------------------------

Use arrow keys to move the turtle.

Use G|B|V|C|D|E|R|T keys to rotate to absolute orientations. 'F' to cancel a rotation.

'Q' to quit.

ROS2 Node Info

ros2 node info /turtlesim

/turtlesim

Subscribers:

/parameter\_events: rcl\_interfaces/msg/ParameterEvent

/turtle1/cmd\_vel: geometry\_msgs/msg/Twist

Publishers:

/parameter\_events: rcl\_interfaces/msg/ParameterEvent

/rosout: rcl\_interfaces/msg/Log

/turtle1/color\_sensor: turtlesim/msg/Color

/turtle1/pose: turtlesim/msg/Pose

Service Servers:

/clear: std\_srvs/srv/Empty

/kill: turtlesim/srv/Kill

/reset: std\_srvs/srv/Empty

/spawn: turtlesim/srv/Spawn

/turtle1/set\_pen: turtlesim/srv/SetPen

/turtle1/teleport\_absolute: turtlesim/srv/TeleportAbsolute

/turtle1/teleport\_relative: turtlesim/srv/TeleportRelative

/turtlesim/describe\_parameters: rcl\_interfaces/srv/DescribeParameters

/turtlesim/get\_parameter\_types: rcl\_interfaces/srv/GetParameterTypes

/turtlesim/get\_parameters: rcl\_interfaces/srv/GetParameters

/turtlesim/list\_parameters: rcl\_interfaces/srv/ListParameters

/turtlesim/set\_parameters: rcl\_interfaces/srv/SetParameters

/turtlesim/set\_parameters\_atomically: rcl\_interfaces/srv/SetParametersAtomically

Service Clients:

Action Servers:

/turtle1/rotate\_absolute: turtlesim/action/RotateAbsolute

Action Clients:

/turtle1/rotate\_absolute action for /turtlesim is under Action Servers

This means /turtlesim responds to and provides feedback for the /turtle1/rotate\_absolute action.

/teleop\_turtle node has the name /turtle1/rotate\_absolute under Action Clients meaning that it sends goals for that action name.

ros2 node info /teleop\_turtle

/teleop\_turtle

Subscribers:

/parameter\_events: rcl\_interfaces/msg/ParameterEvent

Publishers:

/parameter\_events: rcl\_interfaces/msg/ParameterEvent

/rosout: rcl\_interfaces/msg/Log

/turtle1/cmd\_vel: geometry\_msgs/msg/Twist

Service Servers:

/teleop\_turtle/describe\_parameters: rcl\_interfaces/srv/DescribeParameters

/teleop\_turtle/get\_parameter\_types: rcl\_interfaces/srv/GetParameterTypes

/teleop\_turtle/get\_parameters: rcl\_interfaces/srv/GetParameters

/teleop\_turtle/list\_parameters: rcl\_interfaces/srv/ListParameters

/teleop\_turtle/set\_parameters: rcl\_interfaces/srv/SetParameters

/teleop\_turtle/set\_parameters\_atomically: rcl\_interfaces/srv/SetParametersAtomically

Service Clients:

Action Servers:

Action Clients:

/turtle1/rotate\_absolute: turtlesim/action/RotateAbsolute

ROS2 Action List

ros2 action list

/turtle1/rotate\_absolute

ros2 action list -t

[Action List + Action Type]

/turtle1/rotate\_absolute [turtlesim/action/RotateAbsolute]

ROS2 Action Info

ros2 action info /turtle1/rotate\_absolute

Action: /turtle1/rotate\_absolute

Action clients: 1

/teleop\_turtleAction servers: 1

/turtlesim

For the /turtle1/rotate\_absolute action,

/teleop\_turtle node has an action client

/turtlesim node has an action server

ROS2 Interface Show

ros2 interface show turtlesim/action/RotateAbsolute.action

Ros2 Action Send\_Goal

ros2 action send\_goal <action\_name> <action\_type> <values>

<values> need to be in YAML format.

ros2 action send\_goal /turtle1/rotate\_absolute turtlesim/action/RotateAbsolute "{theta: 1.57}"

Waiting for an action server to become available...Sending goal: theta: 1.57

Goal accepted with ID: f8db8f44410849eaa93d3feb747dd444

Result: delta: -1.568000316619873

Goal finished with status: SUCCEEDED

All goals have a unique ID, shown in the return message.

Writing simple publisher and subscriber

ros2 pkg create --build-type ament\_cmake cpp\_pubsub

cd cpp\_pubsub/src/publisher\_member\_function.cpp

package.xml

CMakeLists.txt,

//standard C++ headers

#include <chrono>

#include <functional>

#include <memory>

#include <string>

//To use the most common pieces of the ROS 2 system

#include "rclcpp/rclcpp.hpp"

//Built-in message type you will use to publish data

#include "std\_msgs/msg/string.hpp"

//node’s dependencies

using namespace std::chrono\_literals;

//Node Class : inheriting from rclcpp::Node

class MinimalPublisher : public rclcpp::Node

public:

//Class function : public constructor node, Init count to 0

MinimalPublisher() : Node("minimal\_publisher"), count\_(0)

{

//Publisher = this refer to node->create publisher

//Publisher is initialized with string type(topic\_name,queue size to limit messages in the event of a backup.

publisher\_ = this->create\_publisher<std\_msgs::msg::String>("topic", 10);

//timer is initialize with timer\_callback function to be executed twice a second.

timer\_ = this->create\_wall\_timer(500ms, std::bind(&MinimalPublisher::timer\_callback, this));

private:

//timer\_callback function is where the message data is set and the messages are actually published.

//RCLCPP\_INFO macro ensures every published message is printed to the console.

void timer\_callback()

{

auto message = std\_msgs::msg::String();

message.data = "Hello, world! " + std::to\_string(count\_++);

RCLCPP\_INFO(this->get\_logger(), "Publishing: '%s'", message.data.c\_str());

publisher\_->publish(message);

}

//Declaration of the timer, publisher, and counter fields

rclcpp::TimerBase::SharedPtr timer\_;

rclcpp::Publisher<std\_msgs::msg::String>::SharedPtr publisher\_;

size\_t count\_;

MinimalPublisher class is main, where the node actually executes.

int main(int argc, char \* argv[]){

{

//rclcpp::init initializes ROS 2

//rclcpp::spin starts processing data from the node, including callbacks from the timer.

rclcpp::init(argc, argv);

rclcpp::spin(std::make\_shared<MinimalPublisher>());

rclcpp::shutdown();

return 0;

}

**Subscriber Node**

wget -O subscriber\_member\_function.cpp https://raw.githubusercontent.com/ros2/examples/foxy/rclcpp/topics/minimal\_subscriber/member\_function.cpp

**Subscriber\_member\_function.cpp**

#include <memory>

#include "rclcpp/rclcpp.hpp"

#include "std\_msgs/msg/string.hpp"

using std::placeholders::\_1;

//Class : public rclcpp::node{}

class MinimalSubscriber : public rclcpp::Node{

public:

//Class() : Node(node name)

MinimalSubscriber() : Node("minimal\_subscriber")

{

subscription\_ = this->create\_subscription<std\_msgs::msg::String>(

"topic", 10, std::bind(&MinimalSubscriber::topic\_callback, this, \_1));

}

private:

//topic\_callback function receives the string message data published over the topic

void topic\_callback(const std\_msgs::msg::String::SharedPtr msg) const

{

RCLCPP\_INFO(this->get\_logger(), "I heard: '%s'", msg->data.c\_str());

}

rclcpp::Subscription<std\_msgs::msg::String>::SharedPtr subscription\_;

};

int main(int argc, char \* argv[])

{

rclcpp::init(argc, argv);

rclcpp::spin(std::make\_shared<MinimalSubscriber>());

rclcpp::shutdown();

return 0;

}

For publisher node, spinning meant starting the timer

For subscriber node, it simply means preparing to receive messages whenever they come.

**Add dependencies**

<?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>cpp\_pubsub</name>

<version>0.0.0</version>

#description, maintainer, license

#<description>Package description</description>

<description>Examples of minimal publisher/subscriber using rclcpp</description>

<maintainer email="marlarchan.30@gmail.com">marlar</maintainer>

#<license>TODO: License declaration</license>

<license>Apache License 2.0</license>

<exec\_depend>rclpy</exec\_depend>

<exec\_depend>std\_msgs</exec\_depend>

<buildtool\_depend>ament\_cmake</buildtool\_depend>

<test\_depend>ament\_lint\_auto</test\_depend>

<test\_depend>ament\_lint\_common</test\_depend>

#This declares the package needs rclcpp and std\_msgs when its code is executed.

<depend>rclcpp</depend>

<depend>std\_msgs</depend>

<export>

<build\_type>ament\_cmake</build\_type>

</export>

</package>

#### CMakeLists.txt

cmake\_minimum\_required(VERSION 3.5)

project(cpp\_pubsub)

# Default to C++14

if(NOT CMAKE\_CXX\_STANDARD)

set(CMAKE\_CXX\_STANDARD 14)

endif()

#Compiler

if(CMAKE\_COMPILER\_IS\_GNUCXX OR CMAKE\_CXX\_COMPILER\_ID MATCHES "Clang")

add\_compile\_options(-Wall -Wextra -Wpedantic)

endif()

# find dependencies

find\_package(ament\_cmake REQUIRED)

#Add New dependencies manually

find\_package(rclcpp REQUIRED)

find\_package(std\_msgs REQUIRED)

#For Publisher

add\_executable(talker src/publisher\_member\_function.cpp)

#For Listener

add\_executable(listener src/subscriber\_member\_function.cpp)

ament\_target\_dependencies(talker rclcpp std\_msgs)

install(TARGETS

talker

listener

DESTINATION lib/${PROJECT\_NAME})

ament\_package()

**Build and run**

rosdep install -i --from-path src --rosdistro <distro> -y

**Colcon Build Package**

colcon build --packages-select cpp\_pubsub

**Source Setup file**

. install/setup.bash

**Run Talker node**

ros2 run cpp\_pubsub talker