Robot Operating System (ROS2)

Vilma Muço

Université de Toulon, Master Ingénierie des Systèmes Complexes (ISC) -Erasmus Mundus MIR

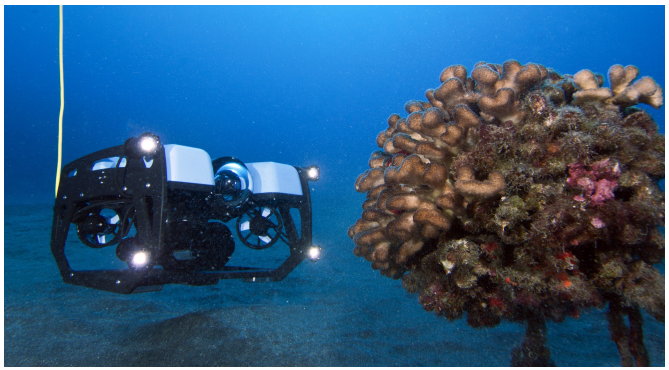
December, 2024

Table of Contents

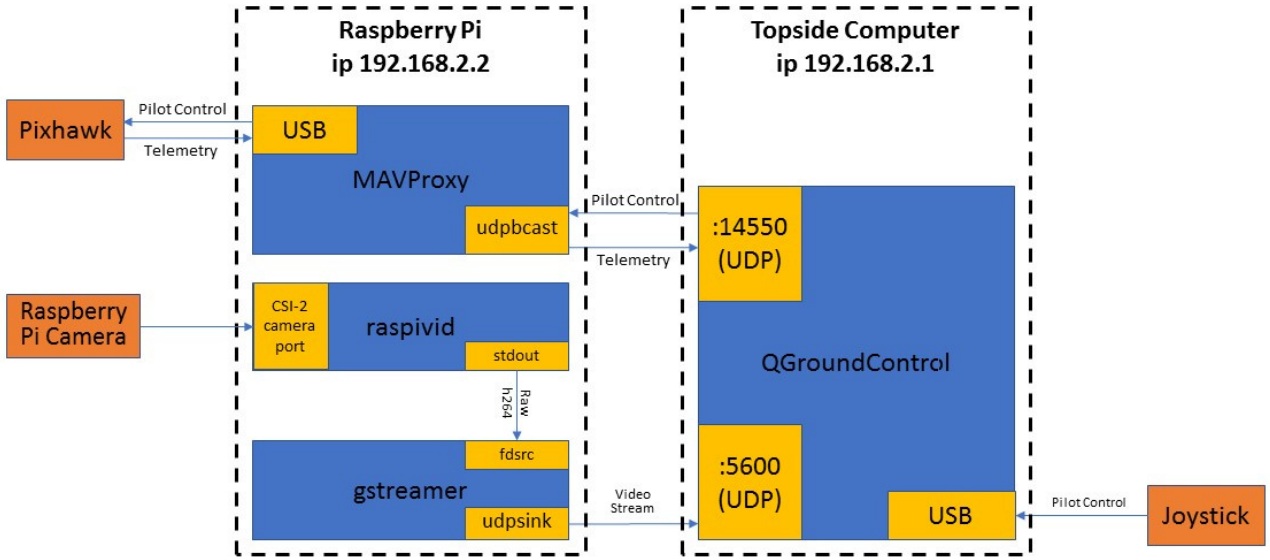
BlueRov2

Goal of this session:

- Learn basic workflow using ROS2
- Learn industry standard tools
- Understand the challenges and difficulties of deploying a robot in simulation



BlueRov2 Architecture



- Gazebo is a simulation tool for robotic systems.
  - A robust physics engine
  - High-quality graphics
  - Convenient programmatic and graphical interfaces
- It can be used as a standalone application or integrated as a ROS package.
- Gazebo Garden (Gazebo Classic (EOL january 2025) $\rightarrow$ Ignition $\rightarrow$ Gazebo)

Gazebo tutorials

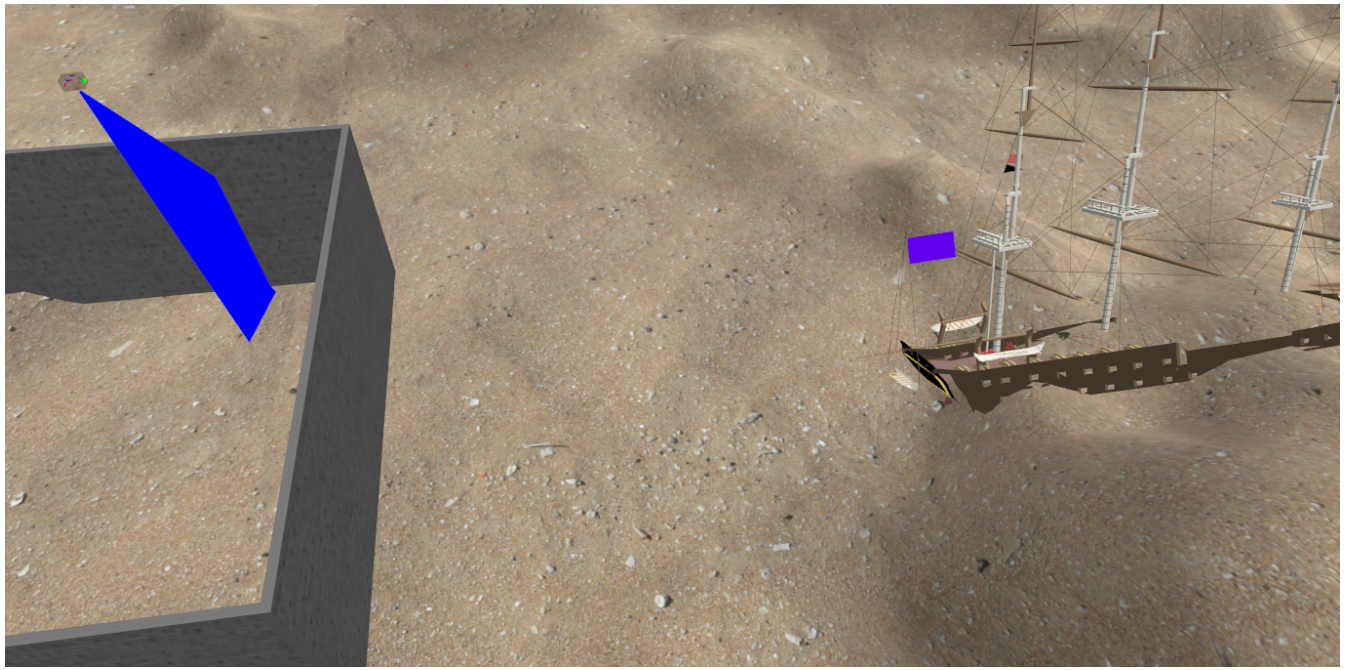
https://gazebosim.org/docs/garden/tutorials
https://classic.gazebosim.org/tutorials

To run a Gazebo simulation you need:

- A world file
  - A file with extension .world that contains all the elements in a simulation, including robots, lights, sensors, and static objects
  - Formatted using the Simulation Description Format (SDF).
- Model files
  - SDF files used to describe objects and robots
  - One object per file - only a single <model>... </model>

The components of a model are:

- Links
  - Physical properties of one body of the model
  - Geometry used for collision checking
  - Visual elements (optional)
  - An inertial element to describe the dynamic properties of the link(such as mass and rotational inertia matrix)
  - Sensor elements for use in plugins (optional)
  - Light elements to describe light sources attached to the link (optional)
- Joints (optional): A joint connects two links (parameters such as axis of rotation, and joint limits)
- Plugins (optional): A plugin is a shared library created by a third party to control a model



Model and underwater world for BlueRov2

https://github.com/clydemcqueen/bluerov2_ignition
https://github.com/remaro-network/remaro_worlds

- To run a gazebo simulation of underwater vehicles we need:
  - BuoyancyPlugin, HydrodynamicsPlugin and ThrusterPlugin
  - ardupilot_gazebo plugin for ardupilot-gazebo communication

gazebo plugins

https://github.com/ardupilot/ardupilot_gazebo/

S.I.T.L (Software In The Loop)

- The SITL (software in the loop) simulator allows you to run Plane, Copter or Rover without any hardware.
- SITL allows you to run ArduSub. It is a build of the autopilot in C++ that allows you to test the behaviour of the vehicle.
- SITL can simulate:
  - multi-rotor aircraft
  - fixed wing aircraft
  - ground vehicles
  - underwater vehicles
  - camera gimbals
  - antenna trackers
  - wide variety of optional sensors, such as Lidars and optical flow sensors

Advantages of ArduSub

- Access to the full range of development tools
- Extensive capabilities out of the box including feedback stability control, depth and heading hold, and autonomous navigation.
- ArduSub works seamlessly with Ground Control Station software that can monitor vehicle telemetry and perform powerful mission planning activities.

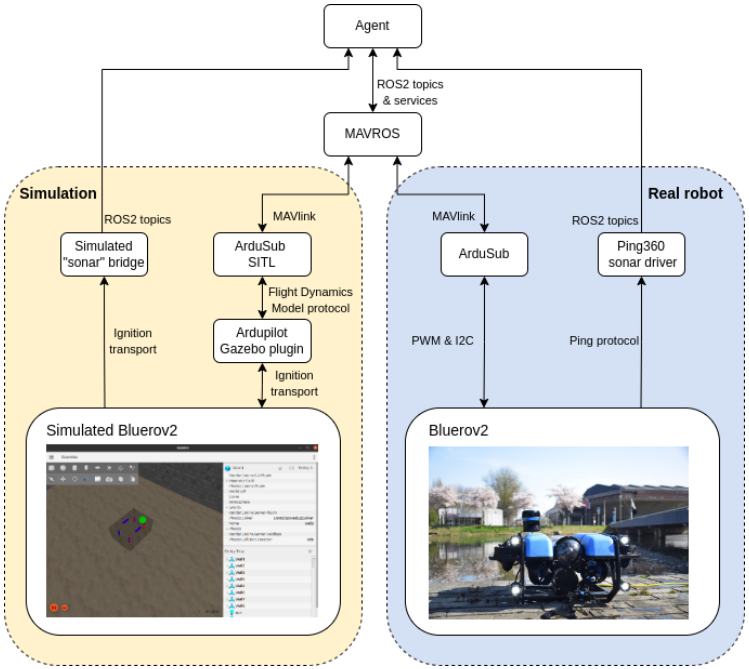
MAVLink

- MAVLink is a very lightweight messaging protocol for communicating with drones (and between onboard drone components)
- MAVLink follows a modern hybrid publish-subscribe and point-to-point design pattern

- Key Features
  - Very efficient.
  - Very reliable.
  - Many different programming languages can be used, running on numerous microcontrollers/operating systems
  - Allows up to 255 concurrent systems on the network (vehicles, ground stations, etc.)
  - Enables both offboard and onboard communications (e.g. between a GCS and drone, and between drone autopilot and MAVLink enabled drone camera).

Mavros2

- A package that provides a communication driver for various autopilots with MAVLink communication protocol. Additional it provides UDP MAVLink bridge for ground control stations (e.g. QGroundControl).


Agent
ROS2 topics
& services
MAVROS
Simulation
ROS2 topics
MAVlink
Simulated
"sonar" bridge
ArduSub
SITL
Flight Dynamics
Model protocol
Ignition
transport
Ardupilot
Gazebo plugin
Ignition
transport
Simulated Bluerov2
Real robot
MAVlink
ROS2 topics
ArduSub
Ping360
sonar driver
PWM & I2C
Ping protocol
Bluerov2

bluerov2.py(bluerov2_agent)

```
1  import sys
2  import rclpy
3  from rclpy.node import Node
4  from rclpy.qos import QoSProfile
5  from mavros_msgs.msg import State
6  from mavros_msgs.srv import CommandBool
7  from mavros_msgs.srv import SetMode
8  from rcl_interfaces.srv import SetParameters
9  from rcl_interfaces.msg import Parameter
10 from rcl_interfaces.msg import ParameterType
11 from geometry_msgs.msg import PoseStamped
12 from mavros_msgs.msg import OverrideRCIn
13 from sensor_msgs.msg import LaserScan
14
15 class BlueRovInterface(Node):
16     def __init__(self, node_name='bluerov_interface'):
17         super().__init__(node_name)
18         self.status = State()
19         self.state_sub = self.create_subscription(
20             State, 'mavros/state', self.status_cb, 10)
21
22         self.local_pos = PoseStamped()
23         self.local_pos_received = False
24         local_position_sub_qos = QoSProfile(
25             reliability=rclpy.qos.ReliabilityPolicy.BEST_EFFORT, depth=5)
26         self.local_position_sub = self.create_subscription(
27             PoseStamped,
28             'mavros/local_position/pose',
29             self.local_pos_cb,
30             local_position_sub_qos)
31
```

bluerov2.py(bluerov2_agent)

```
1
2           self.rc_override_pub = self.create_publisher(
3               OverrideRCIn, 'mavros/rc/override', 10)
4
5           self.override_timer = None
6           self.pitch = 0
7           self.roll = 0
8           self.throttle = 0
9           self.yaw = 0
10          self.forward = 0
11          self.lateral = 0
12          self.camera_pan = 0
13          self.camera_tilt = 0
14          self.light_level1 = 0
15          self.light_level2 = 0
16          self.video_switch = 0
17
18      def status_cb(self, msg):...
19
20      def local_pos_cb(self, msg):...
21
22      def call_service(self, srv_type, srv_name, request):...
23
24      def set_mavros_param(self, name, type, value):...
25
26      def set_mode(self, mode):...
27
28      def arm_motors(self, arm_motors_bool):...
29
30      def rc_override_publish_cb(self):...
31
```

bluerov2.py(bluerov2_agent)

```
    # arguments should be a normalized float value ranging from -1 to 1
    # 0 is translated to 1500 pwm, -1 to 1100, and 1 to 1900
    def set_rc_override_channels(self, pitch=0, roll=0, throttle=0, yaw=0,
                                 forward=0, lateral=0, camera_pan=0,
                                 camera_tilt=0, light_level1=0,
                                 light_level2=0,video_switch=0):...

    def toogle_rc_override(self, run_boolean):...


def mission(ardusub):

    service_timer = ardusub.create_rate(2)
    while ardusub.status.mode != "MANUAL":
        ardusub.set_mode("MANUAL")
        service_timer.sleep()

    print("Manual mode selected")

    while ardusub.status.armed == False:
        ardusub.arm_motors(True)
        service_timer.sleep()

    print("Thrusters armed")
```

bluerov2.py(bluerov2_agent)

```
1
2       print("Initializing mission")
3
4       timer = ardusub.create_rate(0.5) # Hz
5
6       ardusub.toogle_rc_override(True)
7       ardusub.set_rc_override_channels(forward=0.5)
8       timer.sleep()
9       ardusub.set_rc_override_channels(lateral=0.5)
10      timer.sleep()
11      ardusub.set_rc_override_channels(forward=-0.5)
12      timer.sleep()
13      ardusub.set_rc_override_channels(lateral=-0.5)
14      timer.sleep()
15      ardusub.set_rc_override_channels(lateral=0)
16      ardusub.toogle_rc_override(False)
17
18      print("Mission completed")
```

bluerov2.py(bluerov2_agent)

```
1
2   if __name__ == '__main__':
3    print("Starting Bluerov agent node")
4
5    # Initialize ros node
6    rclpy.init(args=sys.argv)
7
8    ardusub = BlueRovInterface("ardusub_node")
9
10   thread = threading.Thread(target=rclpy.spin, args=(ardusub, ),
11   daemon=True)
12   thread.start()
13
14   mission(ardusub)
15
16   ardusub.destroy_node()
17   rclpy.shutdown()
18   thread.join()
19
```

- Instructions how to run the node with docker:

GitHub Repository

https://github.com/vilmamuco/bluerov2_exercise

- Publish an approximation of the height of the wall using the sonar data (try to use the ALT_HOLD mode).
- Before starting the exercise run the following command on the lab computers:

```
1 chmod 777 /tmp/.X11-unix/X[0-1]
```

Tip

Use the command line to find the topic and the field of the message.

Thank you!!!
Questions?

References

https://docs.ros.org/en/jazzy/index.html
https://discuss.bluerobotics.com
https://github.com/remaro-network/tudelft_hackathon
https://sir.upc.edu/projects/rostutorials/
https://ardupilot.org/
http://wiki.ros.org/mavros
https://www.ardusub.com/
http://classic.gazebosim.org/tutorials
https://docs.docker.com/