Sorbonne Université – Département de Sciences De l'Ingénieur
ROS and experimental robotics
February 2023, 1h30.

You work in an underwater robotics company and you have to code a measurement solution. The drone in question is a monitoring drone allowing to monitor and check underwater installations (pipeline ...). The drone is equipped with 4 turbines, a battery, lighting, a camera and many sensors including a pressure sensor that allows to estimate its depth.

Figure 1: CHASING M2 PRO (https://www.chasing.com)

The drone is controlled from a remote which is equipped with joysticks, a pressure sensor and leds indicating the current status of different elements on the drone, like the battery charging level, or warnings about e.g. too far distance between the drone and the pilot.

You are in charge of the depth estimation of the drone, which must be computed from the pressure measured onboard and on the remote. You will have to use this measurements to create an alert to warn the pilot that it is necessary to make the drone go up before losing communication with the drone or damage it due to high pressure.

The depth (in meters) can be estimated with the following relation (considering that the water is incompressible).

$$Depth = \frac{P_{robot} - P_{remote}}{\rho * g} \tag{1}$$

with ρ the density of salt water = 1030 kg/m^3, g = 9.80665 N/kg, P_{robot} (Pa) the pressure measured on the robot and P_{remote} (Pa) the pressure measures on the remote at sea level.

Your colleagues have already developed a ROS node `submarinedrone` running on the drone, and communicating directly with the hardware on the system, and publishing the sensors values on dedicated topics. For now, the engineers are able to publish (i) the pressure measured on the robot by the corresponding sensor in the `/pressure_robot` topic (in Pa), and (ii) the pressure measured on the remote by the corresponding sensor in the `/pressure_remote` topic (in Pa).

In addition, the ROS node `submarinedrone` is also in charge of triggering some LEDs on the remote controller used by the pilot. Your colleagues have so far only implemented the triggering of the `DEPTH WARN` indicator, which is illuminated in green if the drone is not too deep. The change in color of the LED is performed by calling the service `/depth_warning`, which uses as an input the text string "NORMAL" if the drone is not too deep or "WARNING" if it is limited.

Your objective is to write a ROS node called `exam`, which must publish an estimation of the drone depth on a new `/depth` topic (in m), and trigger if needed the `DEPTH WARN` indicator on the remote by calling the appropriate service.

Evaluation

At the end of the exam, you have to upload two files to Moodle:

- A PDF report, written with `openoffice` for example (in french or english), where you answer to the questions below and where you can copy and past the outputs of the different commands you use to answer the questions. For example, for the question "6. List all the running topics", you must put in your report :
  - the exact command line you use to answer the question,
  - and the corresponding output you get.

  Obviously, you have to comment all your responses. A simple copy and past without a sentence explaining the outputs is **not** considered as a full response to the questions.
- A `zip` file of your `exam` package that will be used to access your code, but also to actually launch it.

Preparation

1. To begin with, download the file `submarinedrone.zip` from Moodle. Extract it in the `src` folder of your catkin workspace. Launch `catkin_make` to build your ROS workspace and source you `.bashrc`.
2. Go to the `src` folder of the package `submarinedrone` you just downloaded, and make executable all the python scripts in there with the shell command `chown +x python_file.py`.
3. Quickly test that all is OK by running the node `submarine_drone_node`. Contact your teacher if it is not the case.

1 Exploration of ROS and of the nodes

4. Run the node `submarine_drone_node` from the package `submarinedrone`. Then list all the running nodes.
5. List the information about the `submarine_drone_node` node. Explain each line of the output you get.
6. List all the available topics. Explain the role of the `/rosout` topic.
7. What kind of message is available on the `/pressure_robot` topic?
8. Display a few messages available on the topic `/pressure_robot`. At which rate are they published?

2 The exam node

You have to write a new ROS node `exam` which must compute and publish the altitude of the drone.

9. Create a package `exam` in your `catkin` workspace, with dependencies to `rospy`, `std_msgs` and `submarinedrone` ;
10. Create the node `exam_node` inside the package `exam` which must subscribe to the two topics `/pressure_robot` and `/pressure_remote`. To begin with, you might only display the pressures values received on these topics.

Evaluation

It is important you comment **extensively** your code, even for elementary lines of code. Your node will be assessed by running it (it must obviously run as expected and provide the functionalities listed above), but the quality of the code and its comments will also be taken into account.

11. On the basis on pressure values obtained before, compute the depth in m thanks to Equation (1). The result must then be published on a new topic `/depth` **with a rate of 10 Hz**.

The following questions are mostly independant.

12. Write a launchfile in the `exam` package launching both `submarine_drone_node` and `exam_node` nodes.
13. On the basis on the depth computed before, trigger the call to the `/depth_warning` service to change the color of the `DEPTH WARN` indicator to red or green.
14. The company plans to propose multiple drone models in the future, so that the current depth limit of 150m could be extended.
    (a) To propose a more generic way to deal with these various models, add a `threshold` parameter in the launchfile, and use it in your code to make the `DEPTH WARN` indicator turn red depending on this parameter value.
    (b) In practice, one would want the depth warning to be triggered only if the drone stays under the `threshold` **more than 5s**. Modify your code to implement such a delay. The `DEPTH WARN` indicator must still go back to green as soon as the drone go above the `threshold`.