Software Training

Task 4 (S5)

Note: All tasks should be implemented inside a single ROS 2 workspace. You will submit a GitHub repository link containing your workspace with 3 packages (`pkg_py` for task 1, `weather_pkg` for task 2 and `turtle_pkg` for task 3)

The folder structure should look like:

```
ros2_ws (Your ROS 2 workspace)
└─ src (Source folder)
    ├─ pkg_py (Package for Task 1)
    │   └─ timer_node.py
    ├─ weather_pkg (Package for Task 2)
    │   ├─ temperature_node.py
    │   ├─ humidity_node.py
    │   ├─ pressure_node.py
    │   └─ monitor_node.py
    ├─ turtle_pkg (Package for Task 3)
    │   └─ turtle_node.py
    ├─ install
    ├─ build
    └─ log
```

TURN OVER THE PAGE

Subtask 1: Number Publisher & Counter

Requirements

1. Create a package called pkg_py with a single node inside it called timer_node.py
2. timer_node starts counting down from 10 to 0
3. Print each number in the countdown to the terminal
4. When the countdown reaches 0, log the message: 'Time is up!'

Deliverables

- ROS 2 node timer_node.py that performs the countdown.
- Screenshot of the terminal showing the countdown and the final message.

Subtask 2: Mini Weather Monitor

You are required to build a weather broadcast monitor system where you will receive three sensor readings: temperature, humidity, and pressure. These readings should be collected into a single node and logged into a text file. Take a look at example_interfaces on GitHub, where you will find standard message types such as bool, int16, uint8, and string: https://github.com/ros2/example_interfaces. You can also expand your knowledge with common_interfaces, which contains more advanced message types that could be useful for this task: https://github.com/ros2/common_interfaces.

Requirements

1. Create a package called weather_pkg with 4 nodes inside it.
2. Create a **Temperature Node** that publishes a random temperature every 1 second on topic called /temperature. Use the random library to generate a temperature value within a realistic range (for example, 15–40 ° C). The message type can either be std_msgs/msg/Int32 or sensor_msgs/msg/Temperature from common_interfaces.

   (a) Using std_msgs/msg/Int32 (Message Type: Int32)

   ```python
   from std_msgs.msg import Int32
   def timer_callback(self):
     msg = Int32()
     msg.data = random.randint(15, 40)
     #publish the message

   ```

   (b) Using sensor_msgs/msg/Temperature: **(Message Type: Temperature)** Take a look at the GitHub documentation for the message Temperature → https://github.com/ros2/common_interfaces/blob/rolling/sensor_msgs/msg/Temperature.msg
   You'll find that the Temperature message contains two fields:

   - temperature: the actual temperature value
   - variance: the estimated variance of the measurement

3. Create a **Humidity Node** that publishes a random humidity every 2 seconds on topic called `/humidity`. Use the random library to generate a humidity value within a realistic range (for example, 20–100 %). The message type can either be `std_msgs/msg/Int32` or `sensor_msgs/msg/RelativeHumidity` from `common_interfaces`.

4. Create a **Pressure Node** that publishes a random pressure every 3 seconds on topic called `/pressure`. Use the random library to generate a pressure value within a realistic range (for example, 900–1100 hPa). The message type can either be `std_msgs/msg/Int32` or `sensor_msgs/msg/FluidPressure` from `common_interfaces`.

5. Create a **Monitor Node** that subscribes to `/temperature`, `/humidity`, and `/pressure`.

6. The Monitor Node should print combined output in the form:
   `Temp = XX ° C, Humidity = YY %, Pressure = ZZ hPa.`

7. Save the readings into a text file.

Deliverables

- Working ROS 2 nodes for Temperature, Humidity, Pressure, and Monitor.
- Screenshot of terminal output showing combined sensor data.
- Screenshot of the terminals showing the 4 nodes working together
- Text file with recorded sensor readings.

Subtask 3: Turtlesim

Turtlesim is a beginner friendly package used for learning ROS. Search the ROS2 wiki (ros documentstion) for how to start turtlesim.

Requirements

- use the command line to spawn another turtle
- move each turtle from the command line
- create a node that moves the first turtle using arrow keys. The 2nd turtle using WASD keys.

Deliverables
A video of the 2 turtles moving and the Keyboard binding ros node.

Note:
To spawn a new turtle, use:

```
ros2 service call /spawn turtlesim/srv/Spawn
"{x: 5.0, y: 5.0, theta: 0.0, name: 'turtle2'}"
```

END OF DOCUMENT