Worksheet 4 – Module I Final Work

Autonomous Air Conditioner

The goal of this worksheet is to create an automatic Air Conditioning (AC) system by creating a ROS node, denominated **ac_controller**. Being connected and fully integrated in the ROS network, this AC unit could be controlled by another system or robot programmed in this framework. Therefore, this could be a project in the domotics!

The AC unit can actively change the room temperature through two distinct operating modes: **HEAT** (increasing) and **COOL** (decreasing). Each mode can work in three intensity levels, which in a decreasing order of intensity are: **MAX**, **MED** and **MIN**. These different intensities will make the room temperature change faster or slower, according to the operating mode. The intensity must be automatically changed accordingly to the difference, in Celsius degrees, between the desired temperature and the actual room temperature. See the Table 1 for the required behaviour. When the room temperature matches the desired temperature, with an error margin of 0.1 ºC, the AC unit must halt is functioning, entering to **IDLE** mode. When the temperature difference is outside the bounds of the error margin, the AC unit must return its functioning state to re-establish the desired room temperature. This temperature is defined by the user through a service **/desired_temperature** which must be advertised by the **ac_controller** process. The type of this message is left for the student do decide.

To ease the implementation of this project, a **thermometer** node is provided. This node simulates (on a simplified way) the thermodynamic behaviour in the room and will periodically publish, **sensor_msgs::Temperature** messages that contain the room's current temperature (in Celsius degrees) on the **/temperature** topic. In a similar way, the student should implement the **ac_controller.** The latter must simulate the AC unit's behaviour according to the room temperature and publish *AirBeacon* messages on the **/ac_air** topic. These messages must contain the active operating mode and a ROS Header field. The intensity of the operating mode is simulated by the rate at which the AirBeacon messages are published.

Table 1 - Different operation zones for each intensity and respective publish rates.

Intensity	*AirBeacon* publishing rate (Hz)	Temperature difference (ºC)
MAX	5	>= 4.5
MED	2,5	>= 1.5
MIN	1,25	>= 0.1

The **ac_controller** process should start in the **IDLE** state, requiring the user to start is functioning process through the service **/set_state**. This service's message must have two fields of type **bool**, being the request field named as "active" and the return field as "result".

The room's initial temperature is always 25 ºC. Keep in mind that this value can be changed by the teaching professor during the work's evaluation to test the behaviour of your **ac_controller** in face of sudden room temperature changes due to external causes. This can be achieved through the service **/set_temperature**, which is provided by the **thermometer** process. This service can also be used by the students only for testing purposes. During the evaluation of the **ac_controller** node, the former cannot call this service to manipulate the room temperature! The use of this service is forbidden and failing to comply with this rule will be subjected to an evaluation penalty.

Implementation:

First of all download the provided code files (**t4_package**):

- Create the **AirBeacon.msg, DesiredTemperature.srv** and **ACState.srv** files in the "msg" and "srv" folders, respectively (**HINT**: *See how the **thermometer** node uses AirBeacon messages in its callback, in order to define this custom ROS message. Therefore, do not forget that in order to compile the **thermometer** node, it is mandatory that the student first define and implement the AirBeacon message type*);

- Create a new ROS node in **t4_package**, named as **ac_controller**:

    1. Define and write the **ACController** C++ Class which implements the required ROS node that will automate the AC unit;

    2. Use the **ros::Timer** class to change the publish rate of the AirBeacon message;

    3. Compile the package in order to create the new ROS node **ac_controller.**

- Test your ROS node with the provided **thermometer** node. Then, the updated **t4_package** must be zipped and named as "T4_<NUMBER1>_<NUMBER2>_<NUMBER3>.zip", where NUMBER is your student's number (i.e., T4_44500_45124_58673.zip). This file must be submitted at Moodle platform, no later than **October 13th, 23:55**