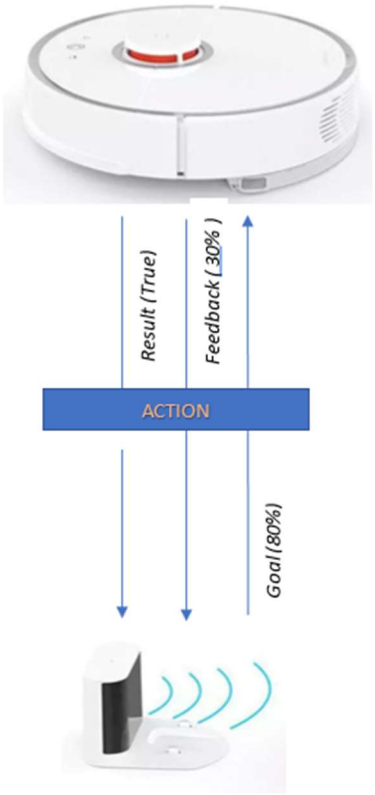
Exercise 3: ROS Actions

The scenario for this exercise is the same as the previous 2 homeworks. We still have a robot vacuum cleaner that communicates with a charging station using ROS. This time, the robot and the charging station interact during the battery charging phase, in particular the scenario will be the following one:

- Suppose that the robot is already arrived and attached to the charging station.
- The robot has a low level of battery (eg. 5%). Thus, it needs to recharge its battery.
- The communication starts with a request (i.e. a Goal) from the charging station that contains the level of battery that the charging station can provide. This request is part of a ROS action that will be the core element of this exercise.
- The robot takes 1 minute to reach the requested level of battery (i.e., implement this logic by yourself). During this time, the robot is writing the current state of battery level in the feedback topic of the ROS action.
- Once the robot arrives at the requested charge state, it communicates the result with the charging station. This is the result of the ROS action.

An example of the proposed scenario is given below:



Given the previous scenario, we request you to implement the following structure:

1. Create a custom ROS action that contains the following:
    a. **Goal** with a std_msgs/Header and an integer that represent the maximum level of battery to charge.
    b. **Feedback** with a std_msgs/Header and the current level of charging. This feedback has to be updated every second (1Hz frequency).
    c. **Result** with a std_msgs/Header and a boolean flag to confirm that the goal is reached.
2. Create the ROS node of the robot that handles the action's goal and sends the feedback and result to the charging node. Remember that this node must implement a "fake" charging procedure (e.g., the robot takes 1 minute to reach the requested battery level, whatever is the input value).
3. Create the charging node that sends the goal to the robot and handles the feedback from the robot. The node must print the feedback in a terminal window.