Z. Kootbally/C. Schlenoff
University of Maryland
College Park, MD

RWA-5: Order Update

ENPM809B : Spring 2020
Due **Wednesday, April 15, 2020**

Contents

Assignment

The goal of this assignment is to update an order while it is being fulfilled.

Instructions

- Write new or reuse previous ROS package(s) to update an order that is being fulfilled (See the wiki for more information on this challenge).
  - The topic ariac/orders will tell you a first order that needs to be fulfilled (order_0).
  - While you are fulfilling order_0, the topic ariac/orders will receive an updated order (order_0_update_0) and you will need to take actions to modify your current order to match order_0_update_0. In other words, order_0_update_0 is the one you want to submit.
    - Important: It may happen that the order update comes in after you finish building the kit. It would be wise that to wait and check if an order update comes in after you finish building a kit and before sending the AGV away.
  - You need to pay attention to both the pose and the type of parts in order_0_update_0.
  - An updated order can fall in one of the following categories or a combination of these categories:
    - You may need to remove parts from the tray (e.g., order_0_update_0 consists of fewer parts than order_0).
    - You may need to add new parts to the tray (e.g., order_0_update_0 consists of more parts than order_0).
    - You may need to move some parts. For instance, the pose of gear_part in order_0 is different in order_0_update_0. If your system is smart it will just move the existing gear_part in the tray to a new pose instead of removing gear_part from the tray and picking up a new gear_part from the bin.
    - You may need to replace existing parts in the tray (e.g., replace piston_rod_part with gear_part).
- The kit to build along with products specifications can be found in sample-rwa5.yaml (uploaded on Canvas). This sample config file is the same as this one. Remember that you are not allowed to directly read this file and you must use the topic ariac/orders.
- Below are specifications for sample-rwa5.yaml. Note that you are not supposed to know these specifications in a real ARIAC competition. I am providing these specifications since this is an assignment and not the final project.
  - Two orders (order_0 and order_0_update_0) but only one shipment (for order_0_update_0).
  - Conveyor belt is not used.
  - All parts can be found in bins.
  - No faulty products.
  - No flipped part challenge.
  - No dropped part challenge.
  - No sensor blackout challenge.
  - The AGV to use is not specified so you are free to use any AGV.
- You need to provide a config file for sensors/cameras.
  - You are free to use any type and any number of sensors/cameras.

Evaluating Your Code

- When we grade your submissions we will evaluate your code using a different trial config file.
  - sample-rwa5.yaml will not be used to evaluate your code.
  - We will use a different config file (eval-rwa5.yaml) that will be made available to you only after the assignment is graded.
  - Whatever you implemented in your code for sample-rwa5.yaml should work on any config file we provide. Your system must be agile.
  - Here are the similarities and differences between sample-rwa5.yaml and eval-rwa5.yaml:
    - Similarities:
      - order_0 followed by order_0_update_0.
      - Conveyor belt is not used.
      - Parts are only found in bins.
      - No flipped part challenge.
      - No dropped part challenge.
      - No sensor blackout challenge.
    - Differences:
      - Bins in eval-rwa5.yaml will not have the same part types as sample-rwa5.yaml. For instance, bin1 in sample-rwa5.yaml contains gear_part while bin1 in eval-rwa5.yaml will contain another type of part or may be empty. You really need to place sensors to cover all bins.
      - The AGV to use (agv1 or agv2) will be specified in eval-rwa5.yaml
      - Faulty products are present in eval-rwa5.yaml (reuse what you did in RWA-4 to deal with faulty products).
      - order_0 and order_0_update_0 used in eval-rwa5.yaml will be different from the ones used in sample-rwa5.yaml. Number of parts required, part types, and part poses in the two orders will be different from sample-rwa5.yaml.

- Competition mode
  - So far you have probably run your program in development mode. At the bottom of sample_environment.launch we have the following piece:

```
<node name="ariac_sim" pkg="osrf_gear" type="gear.py"
    args="--development-mode
      $(arg verbose_args)
      $(arg state_logging_args)
      $(arg gui_args)
      $(arg fill_demo_shipment_args)
      --visualize-sensor-views
      -f $(find osrf_gear)/config/sample.yaml
      $(find osrf_gear)/config/sample_user_config.yaml
      " required="true" output="screen" />
```

  - You need to remove the line --development-mode for this assignment, which will automatically run your code in competition mode. The main difference between development and competition modes is the deactivation of some services and topics in competition mode (see **Cheats** section at the bottom of this page). Competition mode also requires that you install a custom version of Gazebo in your workspace (see the section **Gazebo ROS Packages** on this page). You probably have this package already installed.

Grading Rubric (15 pts)

10 pts– **Building and submitting the updated order**: Full points if the kit is built on the specified AGV (if AGV is specified in YAML file), does not contain faulty parts, and contains all products (in the correct poses) from the updated order.

2 pts– **Package submission**: Name or rename your package in the format **groupName_RWANumber**. When we unzip group1_rwa5.zip, we should get a directory named group1_rwa5. Full points for package named per instruction.

1 pt– **Instructions**: Provide instructions on how to run your program through a Readme.txt, located inside your package. Full points for providing instructions.

2 pts– **Competition mode**: Full points for running your package in competition mode.

-4 pts– **Hardcoding information**: Picking up and placing products require that you get pose data from orders, topics, tf frames, etc. You are not allowed to hardcode these poses in your code. For instance, to grasp a product from the bin, you are not allowed to hardcode the pose of this product. Instead, you will use sensors and transforms.

Remember that we will run your code on eval-rwa5.yaml which is kept secret from you. If we see your robots behaving in strange ways we will investigate the reason. For instance, if we see your robot trying to pick up a part from an empty bin then this is red flag. We will investigate to see if this behavior is scripted.