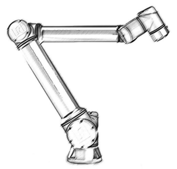


Z. Kootbally & C. Schlenoff
Spring 2023
UMD
College Park, MD

RWA3 (v0.0)

ENPM663: Building a Manufacturing Robot Software System

Due date: **Sunday, April 09, 2023, 9 pm**

Contents

1 Updates

This section describes updates added to this document since its first released. Updates include addition to the document and fixed typos. The version number seen on the cover page will be updated accordingly. There may be some typos even after proofreading the document. If this is the case, please let me know.

- **v0.0**: Original release of the document.

2 Conventions

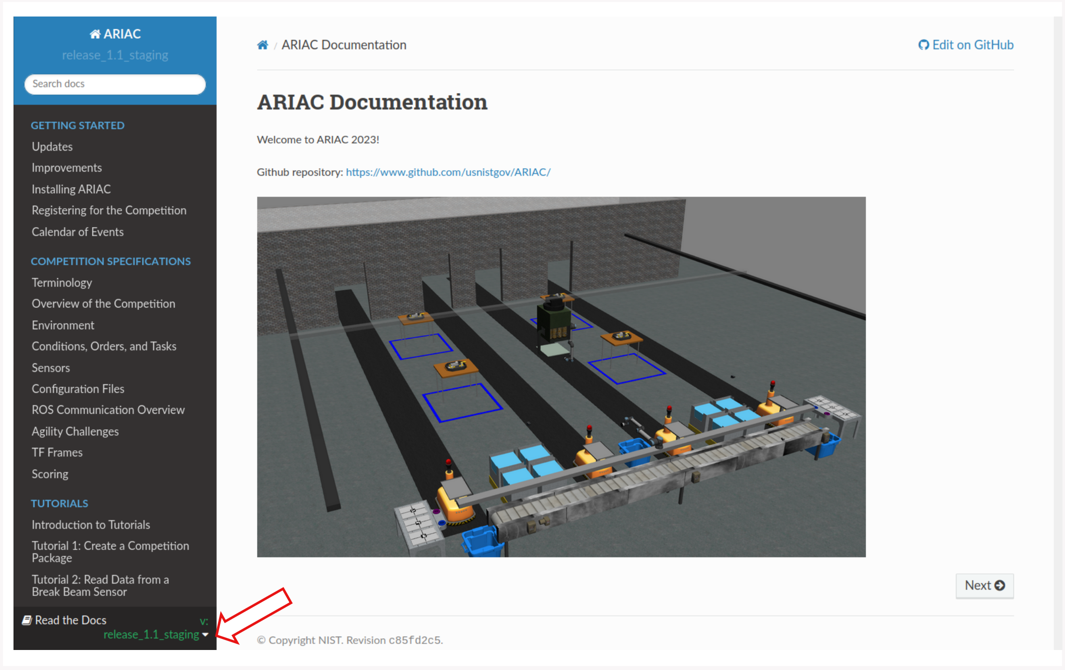
In this document, you will find a number of text styles that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning.

- This is a file.txt
- This is a folder
- This is a package
- This is an important note
- To do
- This is a warning
- This is a link
- This is a t topic
- This is a s service

3 Prerequisites

- Read about the sensors and cameras available in ARIAC.
- Learn about sensor and camera Topics.
- Learn about sensor configuration files.
- Retrieve the file rwa3.yaml from Canvas and place it in the config folder as shown in Figure 3.2.

Figure 3.1: ARIAC documentation.



ARIAC Documentation

Welcome to ARIAC 2023!

Github repository: https://www.github.com/usnistgov/ARIAC/

© Copyright NIST. Revision c85fd2c5.

Figure 3.2: Trial file location.

```
ariac_ws
└ src
  └ ARIAC
    └ ariac_gazebo
      └ config
        └ trials
          └ rwa3.yaml
```

4 Assignment Description

- ⚠ This is a group assignment.
- Reuse the package from RWA2 and augment it to handle the tasks required in this assignment.
- Reuse the package provided in lecture 7 which contains methods using MoveIt to perform pick-and-place using the floor robot.
- Resources needed for the assignment can be found in the official documentation.

   - The most up-to-date documentation is version *release_1.1_staging*. This version will be merged into the main branch and when this happens, you will need to select the *latest* version. I will notify you when this merge happens and I will update this document accordingly.

This assignment consists of building one kitting task order. The order is announced at the beginning of the competition. This order needs 2 parts from the conveyor belt and 2 parts from the bins.

Your competitor control system (CCS) is required to perform the following in this assignment:

1. Start the competition.
2. Parse the Order published on t /ariac/orders (see Tutorial 4 if you are using Python).
3. Locate parts required in the Order (use cameras).
4. Locate tray required in the Order (use cameras).
5. Perform the task within the Order:
   - Move parts from conveyor belt to bins (write this method).
   - Move tray from table to AGV (use methods provided in lecture 7).
   - Move parts from bins to tray (use methods provided in lecture 7).
6. Move the AGV to the warehouse (see Tutorial 5 if you are using Python).
7. Submit the Order.
8. End the competition if all orders have been announced. You need to keep track of the state of the competition before ending the competition. The conditions required to end the competition were explained in RWA2.

Note 4.1: Gripper change.

Before picking up a part or a tray, you need to check the type of gripper on the robots. Use the Topic t /ariac/floor_robot_gripper_state or t /ariac/ceiling_robot_gripper_state.

A gripper change is required if a robot has a *part_gripper* and needs to pick up a tray or if the robot has a *tray_gripper* and needs to pick up a part. When the environment starts, each robot has a *part_gripper*.

5 Sensor/Camera Placements

This assignment requires the use of cameras and sensors to:

- Locate parts on the conveyor belt.
- Locate parts in bins.
- Locate trays on tray tables.

You need to create a sensor configuration file for your sensors and cameras. By convention we name this file `sensors.yaml` but for this assignment and future assignments this file should be named using the convention `<group_name>_sensors.yaml` (e.g., `group1_sensors.yaml`). Instructions on how to place sensors and cameras in the workcell and on how to write the sensor configuration file were provided in lecture 7.

- ☐ Create the folder `config` in your package.
- ☐ Create `<group_name>_sensors.yaml` in `config`
- ☐ Add your sensors and cameras in `<group_name>_sensors.yaml`
- ☐ Test your sensor configuration file with the following command (note the absence of the suffix **.yaml**):

```
ros2 launch ariac_gazebo ariac.launch.py competitor_pkg:=<package>
sensor_config:=<sensor_file> trial_name:=rwa3
```

- For instance, group#1 will run the following command:

```
ros2 launch ariac_gazebo ariac.launch.py competitor_pkg:=group1
sensor_config:=group1_sensors trial_name:=rwa3
```

- ☐ Check the sensors and cameras are receiving data with `ros2 topic echo <topic>` commands (see sensor topics).
  - The competition must be started before sensors and cameras can start publishing data.

Note 5.1: Extra Credit.

In class we saw how to set up cameras in the environment using advanced logical cameras. These cameras are very powerful and you are welcome to use them. You will get 10 pts extra credit if you can successfully identify parts and trays without the use of any advanced logical camera. Basic logical cameras, rgb cameras, and rgbd cameras are good alternatives. You will need to do a bit of OpenCV and maybe use other approaches.

The extra credit will be canceled if at least one advanced logical camera is used. This extra credit will be applied only once, to the current assignment, to one of the future assignments, or to the final project. We will apply it to where it is most advantageous for your team.

6 Parts on the Conveyor Belt

Parts travel on the conveyor belt at a constant speed of 0.2 m/s. Once a part reaches the end of the belt, it is forever gone. As explained in the instructions for RWA2, you can find out parts that will spawn on the conveyor belt by subscribing to the Topic t /ariac/conveyor_parts. To pick up parts from the belt you could use a break beam sensor which alerts when a part crosses the beam (see Tutorial 2). To know which parts crossed the beam, you need a camera above the conveyor belt. There is a possibility to only use a camera and not combine it with a break beam sensor, however, this is more complicated.

Note 6.1: Picking up parts from the belt.

The functionality to pick up parts from the belt was not provided in lecture 7. You will need to write this method. It is expected that you struggle with this method as this is the first method you will write yourself using MoveIt. This is where you will start getting experience with MoveIt and Industrial robotics control.
Tutorial 7 shows how to use Python to task the robot to perform a C++ method. If you are using Python then this tutorial will be helpful.

Note 6.2: Storing parts in bins.

It is recommended to store parts in bins after they are picked up from the belt. Just a reminder that parts on the belt disappear once they reach the end of the belt. It is recommended but not imposed to store all parts in the same bin. When parts required for orders are placed in the bins then the CCS can start working on the kits.

7 Package Submission

As mentioned in an email, we only accept packages submitted on Canvas. Before compressing your package, make sure you do the following:

1. Remove the build, log, and install folders.
2. Rebuild the workspace.
3. Re-run your Nodes with this assignment.

Make sure to include the instructions on how to run your Node or Nodes. The instructions must written in instructions.txt and placed in the folder etc in your package (create this folder if it does not exist). In RWA2 you were asked to place the instructions in the folder documentation. From now on you should use the folder etc

8 Approach

Although this assignment does not include agility challenge, it consists of two main exercises: Locating parts in bins/on belt and picking up parts from the belt. It is expected you spend the majority of the assignment on these two exercises. You are given 2 weeks for this assignment, so make sure to properly split the different tasks among teammates.

9 Grading Rubric

- This assignment is worth 30 pts.
  - 15 pts will be awarded if your package can perform the trial.
  - 5 pts will be awarded for documentation. Although it is not required to generate HTML documentation for this assignment, it is crucial that you document your code. All Python classes, functions, and methods must be documented with docstring. All C++ classes, functions, and methods must be documented with Doxygen. You can document Python code with Doxygen but I never tried it.
  - 10 pts will be awarded if your code can handle a variation of rwa3.yaml. This will showcase that your CCS is agile and you did not hard code dynamic aspects. Your code should be able to perform the trial given the following modifications to rwa3.yaml:
    1. Change the offsets of the parts on the belt.
    2. Change the order parts are spawned on the belt from *random* to *sequential*.
    3. Change the spawn rate on the belt.
    4. Use different bins and change the slots for parts within the bins.
    5. Change the quadrant for each part on the tray.
    6. Change the tray id.