



Zeid Kootbally

Spring 2023

UMD

College Park, MD

RWA4 (v0.0)


ENPM809E: Python Applications for Robotics

Due date: **Friday, April 28, 2023, 11 pm**

Contents

1	Updates	3
2	Conventions	3
3	Package	3
4	Assignment	4
5	Tasks	6
5.1	Reading an Order	6
5.1.1	Todo	8
5.2	Retrieving Tray Poses	8
5.2.1	Todo	9
5.3	Retrieving Part Poses	10
5.3.1	Todo	11
5.4	Outputs	12
5.4.1	Todo	12
5.5	Robustness	13
6	Grading Rubric	13









1 Updates

This section describes updates added to this document since its first release. 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 an important note 
- This is a warning 
- This is a [link](#)
- This is a set of tasks to do 
- This is a  node
- This is a  topic
- This is a  frame

3 Package

1. This is a group assignment.
 -  Many of you have not emailed me their group members. Please do so ASAP.
2. Some changes have been made to the package  ARIAC, which was used in Lecture 10. Get the new updates with:
 -  `cd <path to enpm809e_final_spring2023`
 -  `git pull`
3. In the ROS workspace created in Lecture 10, create a new ROS package with the naming convention  rwa4_group# (e.g., rw4_group1). Since the groups have not been created yet on Canvas, for now name this package  rwa4_group. You will rename this package using your group number once the groups are added to Canvas. To rename a package, you will need to do the following:
 - Rename the name of the necessary folders with the new name (you have two folders to rename).
 - Rename the package in  package.xml
 - Rename the file located in the folder  resource of your package.


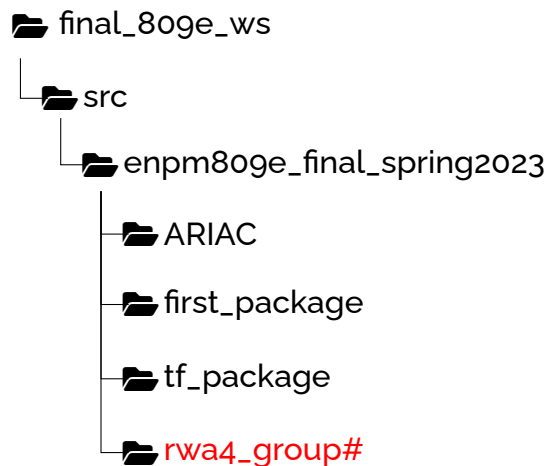
- Use the correct package name in `setup.cfg`
 - Rename the necessary fields in `setup.py`
 - Remove the folder `log`, `install`, and `build`, and rebuild and test your Node.
4. The dependencies for this package are:
- rclpy
 - geometry_msgs
 - tf2_ros
 - orocos_kdl
 - python3-pykdll
 - If more dependencies are needed, you can later add them to `package.xml`
5. Edit the maintainer tags in `package.xml`
- `<maintainer email="student1@umd.edu">student1 first last</maintainer>`
 - `<maintainer email="student2@umd.edu">student2 first last</maintainer>`
6. Once you are done with the assignment:
- Create a folder `instructions` in your package and create `instructions.txt` in this folder. In `instructions.txt` write the full command to start your node (`ros2 run <package> <executable>` or `ros2 launch <package> <launch file>`)
 - Compress (zip) `rwa4_group#` and submit it on Canvas.  Make sure you only submit this package and nothing else. Your workspace should have the structure shown in Figure 3.1.

Figure 3.1: Workspace structure for RWA4.



4 Assignment

This assignment and the next one lead to the final project, which are a simplification of the [ARIAC competition](#). The final project is about kitting, which is the process of picking up parts from bins, placing them in a tray located on an automated guided vehicle (AGV), and submitting the AGV. The

final product is called a kit (see [Figure 4.1](#)). The current assignment only performs one third of the final project. The simulated environment is depicted in [Figure 4.2](#).

📺 A video has been uploaded on Canvas showing the steps to build a kit.

Figure 4.1: A kit.

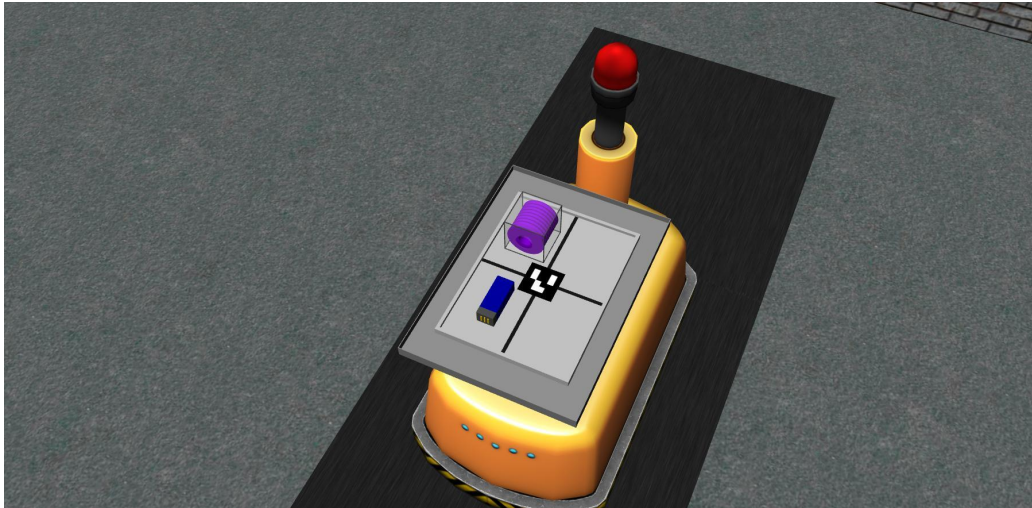
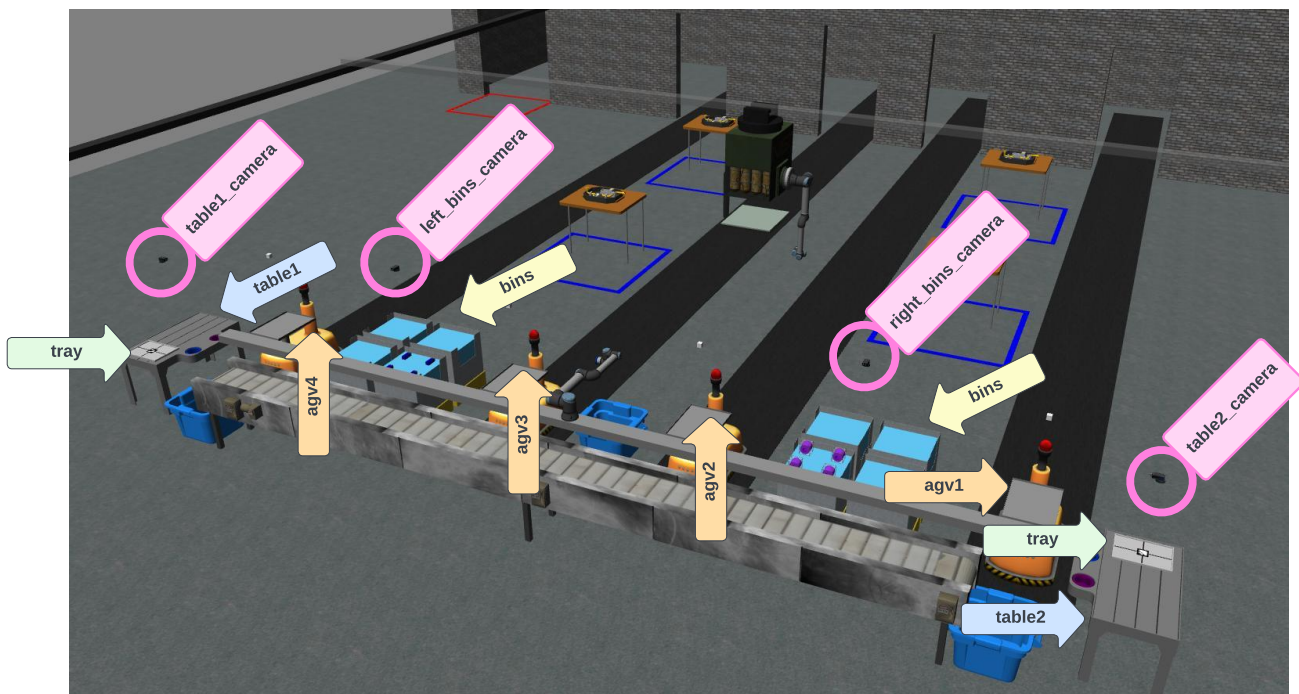


Figure 4.2: Simulated Environment.



5 Tasks

This assignment focuses on reading one Order published on the Topic `t /ariac/orders`, locate trays and parts needed for the Order, and logs the parts and tray locations in the terminal. In future assignments you will task the robot to move trays and parts (as shown in the video).

5.1 Reading an Order

Orders are published on the Topic `t /ariac/orders`. Each Order published on this Topic consists of one of the three tasks: Kitting, Assembly, or Combined (Kitting + Assembly). In this assignment and in the next ones, only Kitting tasks are considered.

Listing 5.1 shows one Order published on `t /ariac/orders`:

- `id`: A unique alphanumeric id.
- `type`: Type of the task (Kitting: 0, Assembly: 1, and Combined: 2)
- `priority`: The priority of this Order (high priority: 1 and low priority: 0). If multiple Orders are announced, the ones with high priorities must be built first.
- `kitting_task`: Information about the kit to build.
 - `agv_number`: The AGV to use to build the kit (1, 2, 3, or 4).
 - `tray_id`: The tray to use to build the kit.
 - `destination`: The destination to send the AGV when the kit is complete. For kitting, the destination is always 3, which represents the warehouse.
 - `parts`: Information on parts needed to build the kit.
 - ◊ `color`: The color of the part. An integer is used for a part color (see Figure 5.1).
 - ◊ `type`: The type of the part. An integer is used for a part type (see Figure 5.1).
 - ◊ `quadrant`: The location of the part within the tray. The location is identified by a quadrant number (see Figure 5.2).
- `assembly_task` and `combined_task`: These tasks are not used in this course as the focus is only on `kitting_task`.

The Order shown in Listing 5.1 can be described as follows: Build a kit on AGV 4 using tray id 3. The kit consists of a blue battery in quadrant 1 and a purple pump in quadrant 2. Once the kit is built, ship the AGV to the warehouse.

Listing 5.1: Example of an order.

```
1 id: MMB30H56
2 type: 0
3 priority: false
4 kitting_task:
5   agv_number: 4
6   tray_id: 3
7   destination: 3
8   parts:
```

```
9  - part:
10      color: 2
11      type: 10
12      quadrant: 1
13  - part:
14      color: 4
15      type: 11
16      quadrant: 2
17  assembly_task:
18      agv_numbers: []
19      station: 0
20      parts: []
21  combined_task:
22      station: 0
23      parts: []
24  ---
```

Figure 5.1: Part colors and types.

Part Colors

red (id=0)

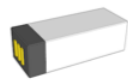
green (id=1)

blue (id=2)

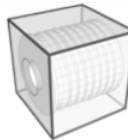
orange (id=3)

purple (id=4)

Part Types



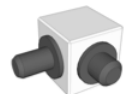
battery (id=10)



pump (id=11)

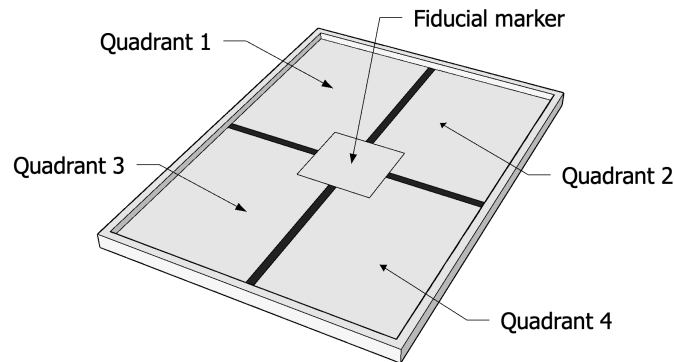


sensor (id=12)





regulator (id=13)

Figure 5.2: Quadrants in a kit tray.



5.1.1 Todo



1. Create a Node `n rwa4` with a subscriber to the Topic `t /ariac/orders`.
2. For each Order published on this Topic, store it in your program as a Python class.  In this document, each time a Python class is mentioned, it can be substituted to Python classes. The only fields that are not stored in the Python class are `assembly_task` and `combined_task` fields.
3. Test your Node as follows:
 - *terminal 1*: Run your Node first. The subscriber will listen to Orders published to `t /ariac/orders`. If you run the simulation first and then your Node, your Node may miss some Orders published on `t /ariac/orders`. : Make sure that your Node prints the attributes of your Python class in the terminal (override `__str__()`) to check that your Python class properly stored information from the Order.
 - *terminal 2*: `>- ros2 launch ariac_gazebo ariac.launch.py trial_name:=kitting`



5.2 Retrieving Tray Poses

Kitting requires the use of a kit tray, which is originally located on one of the two tray tables (table1 or table2). A camera (table1_camera) is located above table1 and a camera (table2_camera) is located above table2. Both cameras can detect trays located on the tables. Outputs for table1_camera are shown in Listing 5.2.

- `tray_poses` provides information on trays that are detected by the camera.
 - `id`: The id of the detected tray.
 - `pose`: Pose of the tray in the camera frame.

- `sensor_pose` provides the pose of the camera in the `f world` frame.

5.2.1 Todo



1. Create subscribers for `table1_camera` and `table2_camera`.
2. In the subscriber callbacks, write the code to store the detected trays in a Python class. The pose of the detected trays must be converted to `f world` frame in your Python class. Use KDL frames to perform the conversion directly in the callbacks by using the method `_multiply_pose()`, as seen in Lecture 10. This method is located in the package `tf_package`. You need to copy this method into your own program before using it.
 - To preserve bandwidth, store only the first Message published on these camera Topics and omit Messages published after the first one. Approaches to store only the first Message from a callback were shown in class.



Listing 5.2: Message for a detected tray.



```
---
part_poses: []
tray_poses:
- id: 3
  pose:
    position:
      x: 1.0650071639601342
      y: 0.43000709739440995
      z: 0.03999613950616021
    orientation:
      x: 0.49999996947669534
      y: -0.49999997474591845
      z: -0.4999968671158639
      w: 0.5000031886415385
sensor_pose:
  position:
    x: -1.3
    y: -5.8
    z: 1.8
  orientation:
    x: -0.5000006633870012
    y: -0.49999933659210455
    z: 0.5000038267902595
```

```
w: -0.499996173200466
```

```
---
```

5.3 Retrieving Part Poses

The next task consists of storing parts detected by cameras in a Python class. Parts are located in bins and can be detected by the cameras located above the bins: `right_bins_camera` and `left_bins_camera`.

✍ One of these cameras was used during Lecture 10. Outputs for `left_bins_camera` are shown in Listing 5.3.

Listing 5.3: Outputs from `left_bins_camera`.

```
---
part_poses:
- part:
  color: 2
  type: 10
  pose:
    position:
      x: 1.0765712783497767
      y: -0.1550004566938431
      z: -0.20600744024714407
    orientation:
      x: -0.005393963181549608
      y: -0.70500548021254
      z: -0.00537051521979574
      w: 0.7091610082327963
- part:
  color: 2
  type: 10
  pose:
    position:
      x: 1.0765591403311898
      y: -0.5150003594573335
      z: -0.20600611351203402
    orientation:
      x: -0.005933545638302661
      y: -0.7093682958073616
      z: -0.005986976325473967
      w: 0.7047876063425038
- part:
  color: 2
  type: 10
  pose:
    position:
      x: 1.0765655693809077
```

```

    y: -0.15500023907735283
    z: -0.5660059651822175
  orientation:
    x: -0.005935743917295585
    y: -0.7093057462876493
    z: -0.0059888422212663805
    w: 0.70485052244878
- part:
  color: 2
  type: 10
  pose:
    position:
      x: 1.0765954459037081
      y: -0.5150003575965527
      z: -0.5660075793516259
    orientation:
      x: -0.005211072051811127
      y: -0.7050556254500294
      z: -0.0051884029227772235
      w: 0.7091138767676527
tray_poses: []
sensor_pose:
  position:
    x: -2.286
    y: -2.96
    z: 1.8
  orientation:
    x: -0.7071045443232116
    y: -1.2248082623837582e-07
    z: 0.7071090180427863
    w: -1.2246817995071293e-07
---
```


- `part_poses` provides information on parts detected by the camera.
 - `part` provides information on a part.
 - ◊ `color` and `type` report the color and the type of the detected part, respectively.
 - `pose` reports the pose of the detected part in the camera frame.
- `sensor_pose` provides the pose of the camera in the `f world` frame.

5.3.1 Todo



1. Create subscribers for `left_bins_camera` and `right_bins_camera`.
2. In the subscriber callbacks, write the code to store the detected parts in a Python class. The pose of the detected parts must be converted to `f world` frame in your Python class. Use KDL frames to perform the conversion directly in the callbacks by using the method

`_multiply_pose()`.

- Store only the first Message published on these camera Topics and omit Messages published after the first one.
3.  In Lecture 10 we hard coded the pose of the part and the pose of the camera. This was done for demonstration purposes. You have to retrieve this information from your subscriber callbacks, which is then stored in a Python class.



5.4 Outputs

The final task consists of logging information in the terminal. The different steps to log relevant information in the terminal are provided below.

5.4.1 Todo



1. Parse the Order stored in your Python class (from 5.1.1) and retrieve the id of the tray and the needed parts.
2. Parse your Python class (from 5.2.1) to retrieve the tray needed for the Order. The id of the tray is used to retrieve the correct tray.
3. Parse your Python class (from 5.3.1) to retrieve the parts needed for the Order. The color and the type of the parts are used to retrieve the correct parts.
4. Log in the terminal the pose of trays and parts required for the Order. Listing 5.4 shows the outputs for the current simulation environment.



Listing 5.4: Example of outputs for RWA4.



```

1  -----
2  --- Order MMB30H56 ---
3  -----
4  Tray:
5    - id: 3
6    - pose:
7      - position: [-0.8699999990119835, -5.840000010414677, 0.7349894114585056]
8      - orientation: [4.917872024646517e-08, -5.204199039114117e-07, 0.9999999999989833, 1.326796506638446e-06]
9  Part:
10   - Blue Battery
11     - pose:
12       - position: [-1.720000846038678, -2.8049997609226516, 0.723430849637019]
13       - orientation: [0.00315348225744318, -0.008431955602604466, 0.9999594773102691, 3.757284443427965e-05]
14   - Purple Pump
15     - pose:
16       - position: [-1.7200001244660335, 2.8049998427283165, 0.7228902403444724]
17       - orientation: [-0.0014227523651666137, -0.0013557428126529125, 0.9999980688560564, 4.611838618251952e-06]
```

Explanation:

- Line 2: The id of the Order.

- Line 5: The id of the tray needed in the Order.
- Lines 6–8: The pose of the tray in the `f world` frame. If multiple trays with the id are provided in the environment, you need to provide the pose of only one of them.
- Lines 10–13: Information on one of the parts needed in the Order. The environment has four blue batteries. The Order requires only one blue battery, therefore, only one blue battery is logged in the terminal. Note that on line 10 the part color and type are used instead of integers. The pose of the part is in the `f world` frame.
- Lines 14–17: Information on the other part needed in the Order.


5.5 Robustness

If your code is properly implemented, it should be able to handle the following variations:

- Order with different part types, part colors, and part quantities.
- Different part locations.
- Different id for the tray.

Once you are done with the implementation, you can check if your program is robust by launching a different environment with the command below:

- `> ros2 launch ariac_gazebo ariac.launch.py trial_name:=kitting2`

 Your submission will be evaluated using a third environment.

6 Grading Rubric

- This assignment is worth **30 pts**.
 - ▷ **5 pts** for properly storing the Order using OOP.
 - ▷ **5 pts** for properly storing Tray information using OOP.
 - ▷ **5 pts** for properly storing Part information using OOP.
 - ▷ **5 pts** for accurate outputs in the terminal.
 - ▷ **5 pts** for good OOP implementation. This includes the correct use of encapsulation and code convention.
 - ▷ **5 pts** for proper docstring documentation. You must document all classes and methods.
 - ▷ **-2 pts** for not editing the maintainer tags in `package.xml`
 - ▷ **-15 pts** for hard coding information in the terminal rather than retrieving the information dynamically. For example, if in your code you write `self.get_logger().info(f" - id: 3)` for the tray id, you will get an automatic deduction of 15 pts. This id should be retrieved from the Python class where tray information is stored. If you are not sure what and what not to hard code, ask me.
- You will get 0 on the assignment if we notice similar code between groups. If we realize you use an AI software to generate code for the assignment you will also get a 0. Each time you break the rules of code of conduct you are reported to MAGE. Do not make these kind of mistakes even if the temptation is there...