Final Project

Introductory Robot Programming

May 11, 2022

Students:
Darshan Jain
UID - 117454208

Pulkit Mehta
UID - 117551693

Jeffin Johny
UID - 118293929

Instructors:
Z. Kootbally
C. Schlenoff

Group:
5

Semester:
Spring 2022

Course code:
ENPM 663

Contents

1 Introduction

The overall project revolves around understanding and tackling various agility challenges, that a robot might face during kitting and assembly operations. The first step is to develop a robust control system architecture to perform efficiently, followed by the implementation of various functions to solve these challenges using the Agile Robotics for Industrial Automation Competition (ARIAC) interface, MoveIt and ROS. The project deals with kitting operation and assembling these part kits(s) into a briefcase which are to be shipped, based on an order(a set of shipments), using available parts. The robots in use are:

1. a UR10 robotic arm on a linear rail that is capable of only executing kitting operation.
2. a gantry robot, with a torso and UR10 arm attached to it that can accomplish both kitting and assembly tasks.
3. an Automated Guided Vehicle(AGV) to transfer parts from kitting to assembly station.

The parts are either present on static bins/shelves or on a moving conveyor belt, which are decided by the competitor. Once the kit has been built over an Autonomous Guided Vehicle (AGV) it is shipped for assembly. Each shipment is considered to be fully complete only when all the desired, non-faulty, parts are placed on the AGV tray in the correct pose, and the AGV is shipped to the correct assembly station for assembly to take place. The competition also includes various agility challenges such as the inclusion of faulty parts, sensor blackout, part flipping, and high priority order. The final implementation of the project includes but is not limited to ROS/C++programming, coordinate-transformation with TF library, trajectory planning using MoveIt interface, and strategies to tackle all agility challenges for an autonomous system for different scenarios.

2 Approach

2.1 Architecture

Reactive control system is the architecture selected for this project. In brief, all the planning is performed beforehand by the team, and all the robot has to do is check the competition announcement regularly and act according to it. Planning is performed for various possible combinations of shipment types and agility challenges, in order to make the system robust. In terms of time-scale, this architecture is real-time as the robot does not have to do any planning for a particular order apart from path-planning using MoveIt. Here the representation of the internal world is provided, which contains all the locations of bins, briefcases, belt and robots. In addition, the world poses of all the parts are captured by logical cameras and added to the map, and updated when a pose changes.

2.2 Data Structures

The main idea revolves around having an internal map of all the parts in the world at that given point of time. For this a struct "Product" is created of which every part in the world is an instance, which can be referred as in Fig 1.

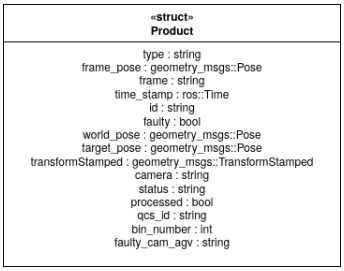


Figure 1: Struct Product

Similarly, structs have been created which have instances for each order, kitting shipments and assembly shipments, as depicted in 2

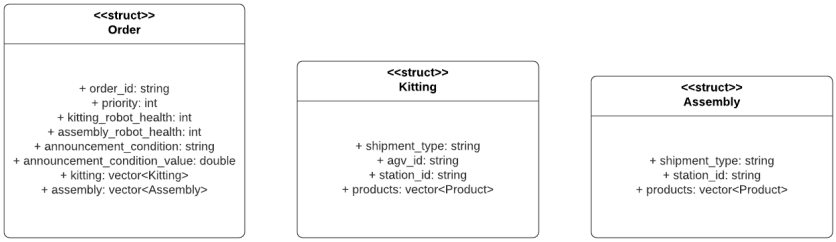


Figure 2: Structs created in order to store information from order message.

The overall class diagram structure is shown in Fig. 3.

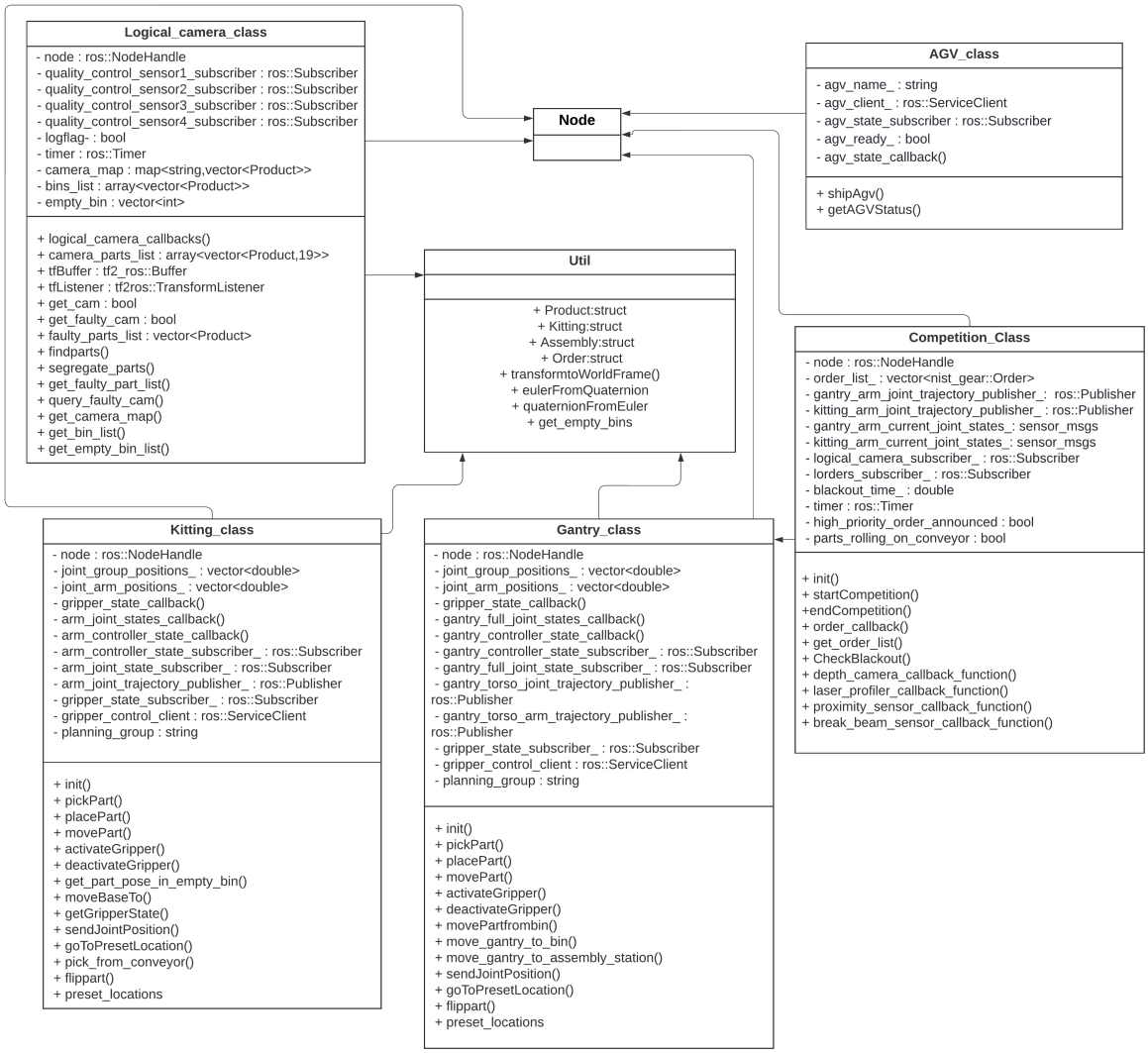


Figure 3: Class diagram layout

2.3 Reading the Orders

The initial order is always announced after the start of the competition, which is read and stored in an order list. New orders are always pushed back into this list, which can be a high priority order or just another regular order. A trigger is created to check for both high priority and regular order, access to which helps in switching order processing.

2.4 Internal World Model Creation and Sensor Placement

- The logical cameras are used to find the parts present in the environment. These cameras detect the type and the pose of the part. The locations of these cameras can be seen in fig 4.
- The callback functions for these cameras are called after every one second but a boolean flag is created to add the parts seen by the cameras in a list only once. This list is a nested list containing the parts seen by each camera.

- A method is created which consists of all the callbacks for the logical cameras. This lets the user create the list of parts only when needed instead of populating the list every time the callback is entered.
- The world pose of each part is stored before adding the part into the list. This is done by broadcasting new frames for part from each camera, with the child frame id's being associated with a random number to avoid name clashing. The method 'transformToWorldPose' from 'Util' is used to accomplish this task.
- The world pose of each part is then checked and a bin number is assigned to each part based on its location in that particular bin in the environment.
- A map with the type of part as the key and a list of parts of that particular type is created. This helps in segregating the parts. The parts also have a camera id associated with them as referred in 1. This id helps in picking the correct part for both the kitting and assembly operation, i.e., parts on the bins and agvs near the bins are used for kitting(including replacing faulty parts) and the parts on the agvs near the assembly stations are used for assembly operation.

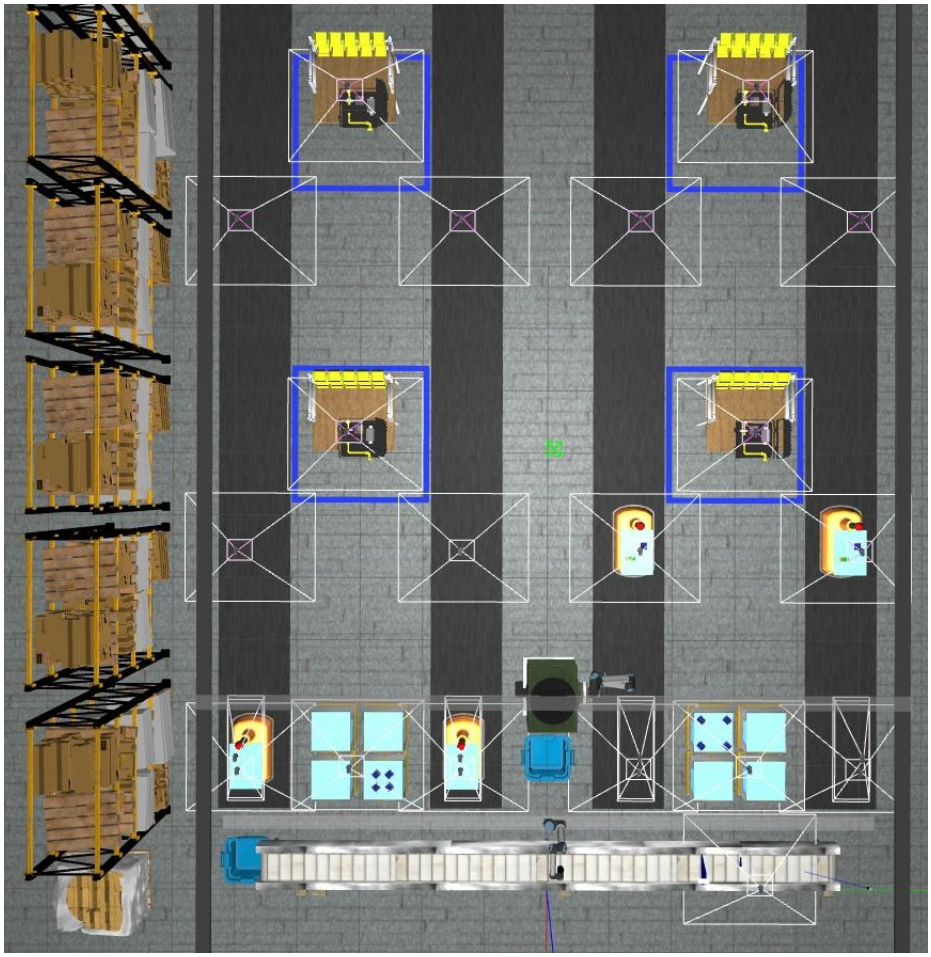


Figure 4: Camera locations and its field of view

2.5 Order Processing

- The orders are processed sequentially and a check is performed whether the current order is already processed or not.
- The first order is read and a list of parts for kitting and assembly is created from the order. The number of shipments processed for both kitting and assembly are tracked and if the number is less than the required number of shipments to be processed, then the whole process is looped again until all the shipments are processed. Initially, when the list of parts in a shipment is created, the "processed" attribute for the part is set by default to false. Only after the part is completely checked to be non-faulty and is placed in the correct location this flag is switched to true and the shipment count is incremented.
- For each part stored in the kitting and assembly lists, the camera map is used to find the required part and a loop is executed within this list of required part type. The part is first placed in the required location by either the kitting or the gantry robot. For parts placed in bins near to the conveyor belt the kitting arm is used and for parts away from the belt the gantry robot is used.
- The scenario for sensor blackout is checked and if there is blackout then the current part is saved in a list of parts to be checked later for their quality, i.e., faulty or non-faulty and the next part in the shipment starts processing. If there is no blackout the below steps are performed.
- The status of high priority order being announced is checked and if so, the current order processing is halted and switching to high priority order happens. The processing flow remains same for both the orders, thus, the steps below apply to this case as well.
- The faulty cameras are triggered to generate a list of faulty parts and if the placed part location matches with any of the parts in the faulty part list, the part is picked up using the kitting arm and dropped into the faulty part bin. The next part in the list of parts of the required type is picked on and the process continues until the part placed on the agv is non faulty. If the part is not faulty, the shipment count is incremented.
- If all the parts in the shipments are processed, the agv is shipped to the desired station and the next shipment starts processing or if there are no more shipments, the next regular order starts processing.

The overall sequence flow for kitting and assembly process is depicted in Fig 14 and Fig 13 respectively.

2.6 Kitting Robot Movement

The kitting robot movement is quite simple as it is an arm attached to a joint on a linear rail which allows its movement along the y-axis. This movement is constrained between the locations for agv1 and agv4, which allows manipulation of parts from agvs at kitting stations and from the bins placed near to the conveyor belt.

For picking up the parts using the kitting arm, the base of the arm joined to the linear rail is moved so that it matches with the y-coordinate of the part and then inverse kinematics is performed to pick the part. This is done using the MoveIt interface.

2.7 Gantry Robot Movement

The gantry robot movement unlike the kitting robot is quite complex as its base has two degrees of freedom. To allow smooth movement of the gantry, avoiding collision and efficient manipulation several preset locations for the gantry robot are defined, which can be referred from Fig 5. Another motivation of using the preset locations is to eliminate the issues that may arise due to solving inverse kinematics. This also helps in terms of the movement being time efficient.

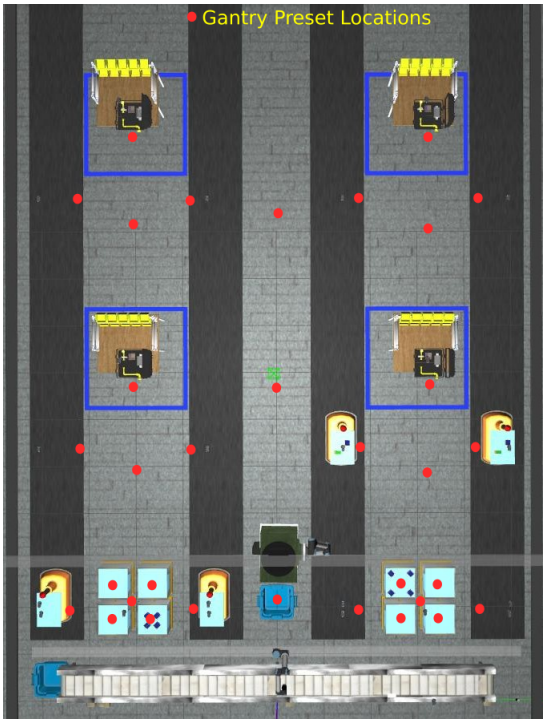


Figure 5: Gantry preset locations

3 Challenges

In ARIAC 2021, the following challenges(including agility challenges) will occur during kit building and during assembly, so the system needs to be agile and generic to handle all of them. Any combination of the challenges can be present. They are as follows:

3.1 Faulty Part

- In this challenge, the criteria is to avoid using faulty parts to build kits. The quality checking can be performed by checking the quality control sensors above each AGV. These provide the pose of faulty parts in their field of view and a sample output of the same is shown in the figure 6. The gazebo screen with a faulty part and a quality control sensor is as shown in figure 7


```
models:
  -
    type: "model"
    pose:
      position:
        x: 0.695297813039
        y: 0.109115222642
        z: -0.37875728957
      orientation:
        x: 0.500680786833
        y: 0.494623370661
        z: -0.502106198448
        w: -0.502549337252
pose:
  position:
    x: -2.393393
    y: -1.325227
    z: 1.506952
  orientation:
    x: -0.707106896726
    y: -1.22474483074e-07
    z: 0.707106665647
    w: -1.22474523098e-07
```


(a) Quality Control Sensor Topic message when faulty part is detected


```
models: []
pose:
  position:
    x: -2.393393
    y: -1.325227
    z: 1.506952
  orientation:
    x: -0.707106896726
    y: -1.22474483074e-07
    z: 0.707106665647
    w: -1.22474523098e-07
```


(b) Quality Control Sensor Topic message in absence of faulty part.

Figure 6: Terminal output of quality control sensor.

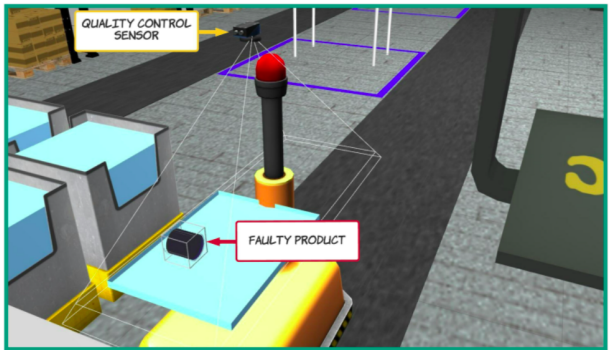


Figure 7: Gazebo screen depicting faulty product and quality control sensor

- The kitting robot is tasked to replace the faulty part on the AGV.

- The faulty part challenge is solved using a simple approach, whenever a part is placed on an AGV, it is checked whether it is faulty or not and if it is faulty, it is replaced immediately with a new part by either the kitting or the gantry robot.
- If the part is placed on an AGV and a sensor blackout is triggered the part is appended to a faulty list that will be checked after the sensor blackout is completed and the next part in the shipment starts processing.
- The Faulty part is always discarded in the faulty bin using the kitting robot, by sending it to a preset location near the faulty bin.
- The AGV shipment is blocked until the faulty part list becomes empty. The flow chart of the faulty part handling is depicted in figure 8

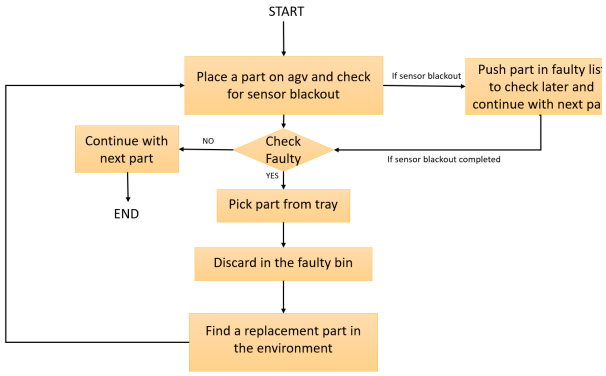


Figure 8: Flow chart for handling the faulty part

3.2 Sensor Blackout

- In the phase of sensor blackout, all the sensors loses communication with the environment temporarily. The duration of the blackout can last for as long as 50 seconds. The critical aspect of this challenge is to continue the order processing smoothly during the blackout.
- As an internal world model of all the parts in the environment is created initially, the need to use any of the logical cameras is eliminated until any assembly operation starts.
- Then as the part is placed on the AGV, the condition for sensor blackout is checked. For detecting this, a timer is created which resets whenever a callback function of the logical camera is called. In the event of blackout the callbacks are not called and the time value in the timer exceeds a prescribed value and it is assumed that blackout has occurred.
- The most challenging part in the challenge being that of keeping track of quality of parts placed during blackout. This is solved by creating a list of parts placed during blackout and by querying the quality control sensors after each shipment to check if the part needs to be replaced or not.

3.3 High Priority Order

- This challenge involves stalling the current kit building or assembly operation and switching to another order with high priority and then completing the former.
- To tackle the challenge a boolean is created which triggers when a high priority order is announced and read by the order callback function.

- After placing a part on the AGV or the assembly station the high priority announcement condition is checked and if it is true the robots start processing the high priority order first and then after ensuring completion switch back to the former order. The overall flow can be seen in Fig 9

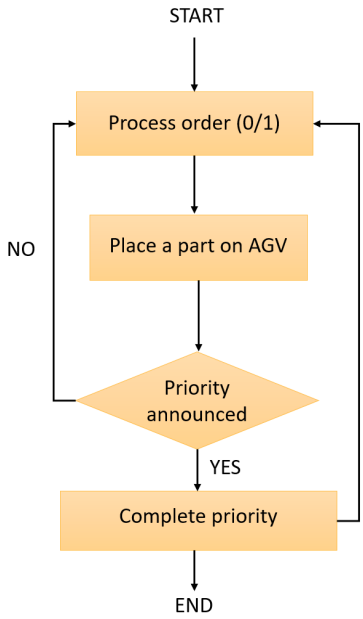
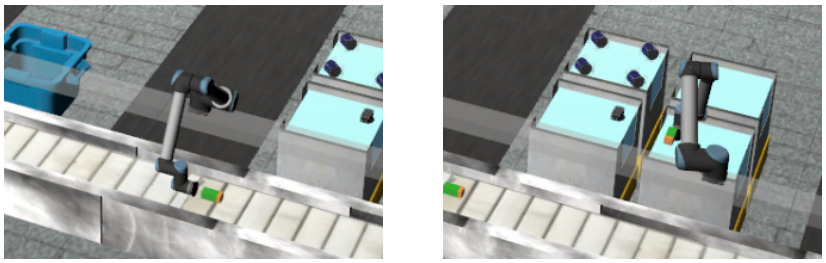


Figure 9: Flow chart for handling the High Priority Order

3.4 Conveyor Pick and Place

- To deal with the scenario of required part in order missing in the bins, parts from the conveyor have to be picked up and placed on an empty bin to complete the order processing.
- The challenge is tackled by:
  1. Initial map creation and querying the logical cameras to find locations of empty bins.
  2. After an initial wait period of 30 seconds, the breakbeam sensor is used to detect the presence of parts on the conveyor belt. If the parts are detected to be spawning on the conveyor, a trigger is initiated to start picking up a certain number parts from the belt.
  3. If the parts are needed to be picked up, the kitting robot is positioned in a predefined location to pick the part and then place it in any of the available empty bin.
  4. Figure 10 shows the predefined pick position and the place operation by the kitting robot.

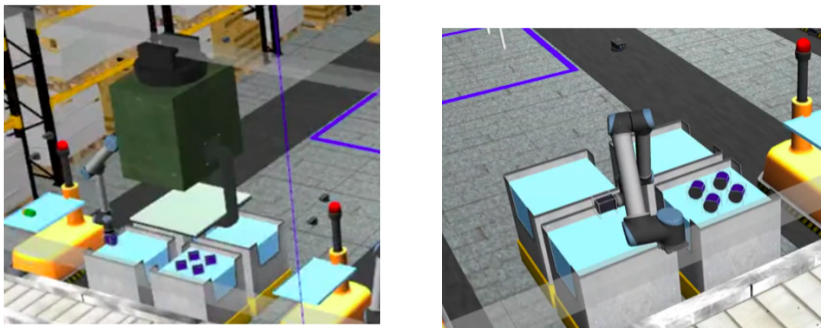


(a) Conveyor Pick Up (b) Bin placement

Figure 10: Conveyor Pick and Place

3.5 Flipped Part

- This challenge involves flipping a pump before placing it on the AGV.
- To tackle the challenge the first step is to check whether the pump needs to be flipped or not, to be flipped roll value of pump in order should be 'π'.
- If the pump is located in the bins near to the conveyor, the kitting robot is tasked to flip the pump otherwise the gantry robot flips the pump.
- The part is picked up and placed on the predefined location on any available empty bin and the robot arm is oriented to pick the pump from the side and rotate the end effector by π radians and drop on the predefined center location on the empty bin. Thus, the flipping task is performed.
- Figure 11 shows the process followed for flipping a part and figure 12 shows the flow diagram of the process.



(a) Part picked up by gantry and placed on predefined location in the empty bin

(b) Kitting robot performing flipping

Figure 11: Flipping a part

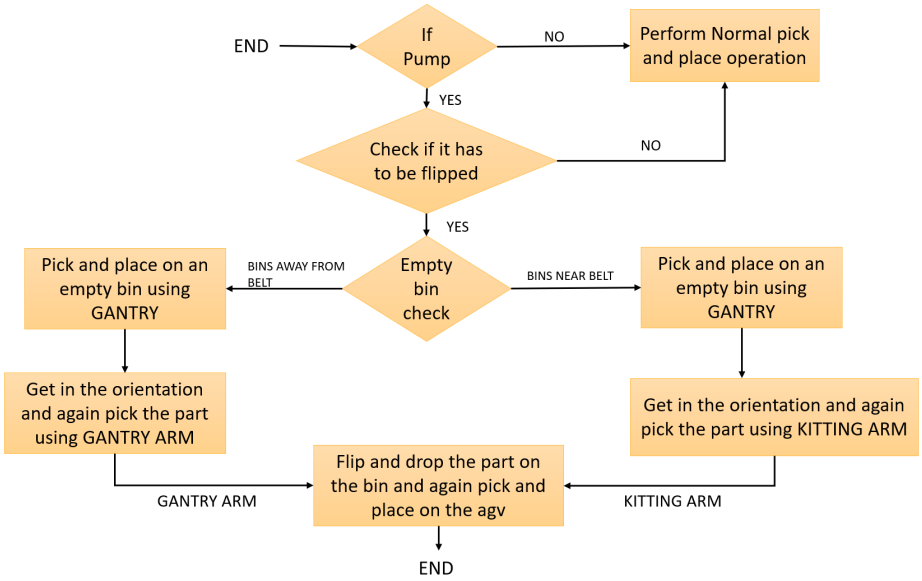


Figure 12: Flow chart for Flipping a Part

4 Project Contribution

Since most of the tasks required running and testing the code simultaneously on multiple systems, each team member played a role in structuring and implementing the code. Some particular areas of contribution apart from common ones include the ones below:

- Pulkit : Creating internal world map, gantry movement, kitting movement, main function execution.
- Darshan: Conveyor pick up, kitting movement, part flipping using kitting arm, sensor blackout.
- Jeffin: Architecture design, gantry movement, sequence flow design, sensor placement and fine tuning.

5 Resources

- ARIAC official Repository: https://github.com/usnistgov/ARIAC
- ROS wiki for cpp: https://github.com/usnistgov/ARIAC

6 Course Feedback

Overall the course was very informative and challenging. A few suggestions for the course structure may be:

- Access to a remote server or system that will eliminate the issues faced due to varying real time factor across different systems of members of a team.
- A few more lectures on alternate implementation methods during the second half of the course.
- Date for final demo may be extended to a day in the week of final exams so that an extra lecture helps in receiving feedback and fine-tuning the code.

7 Appendix

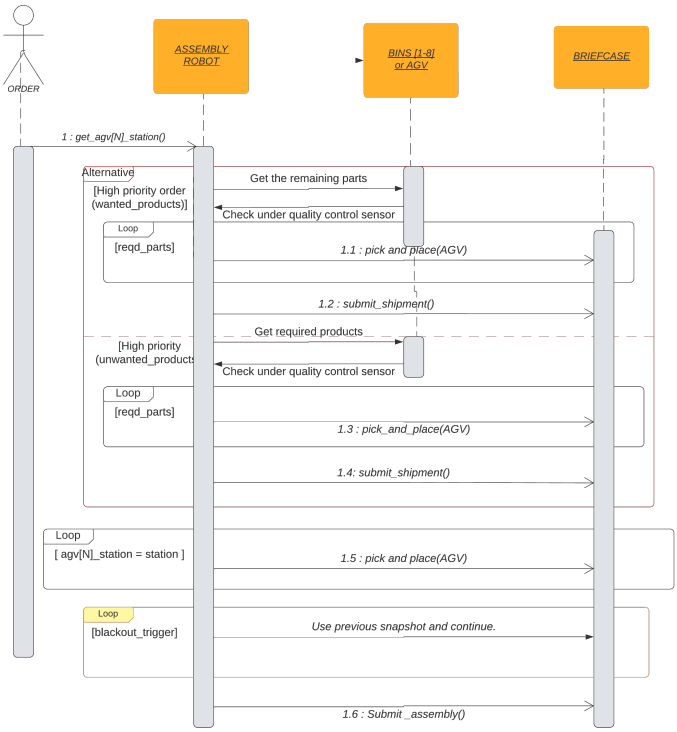


Figure 13: Assembly Sequence Diagram

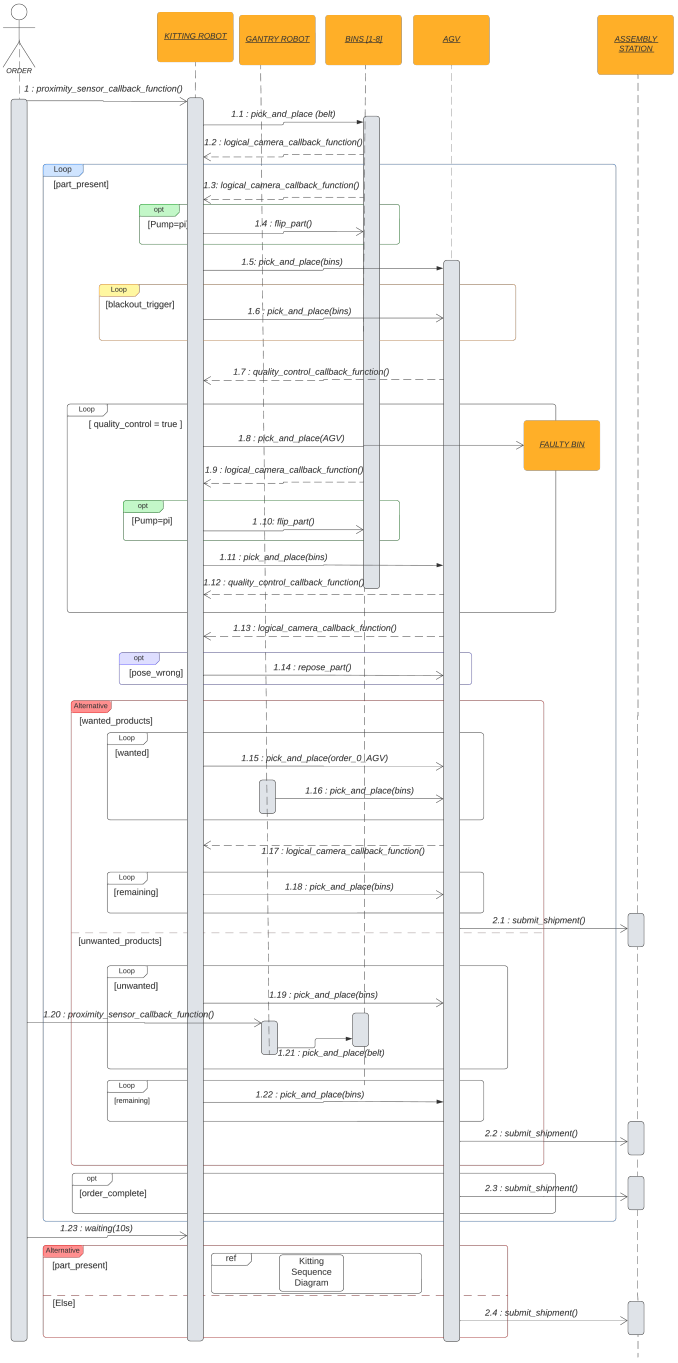


Figure 14: Kitting Sequence Diagram