UNIVERSITY OF MARYLAND,
COLLEGE PARK

FINAL PROJECT REPORT

TEAM-4
Aalap Parimalkumar Rana - 116421393
Ishan Patel- 116169103
Patan Sanaulla Khan - 116950985
Pranav Jain - 116694797
Kulbir Singh Ahluwalia - 116836050

ENPM 809B - Building a Manufacturing Robot Software System

May 3, 2019

Abstract

The main concentration of this project is on understanding importance and implementation of agility in manufacturing process using robots . We will use concepts of object oriented programming in C++, ARIAC interface, Moveit and ROS to build a robust agile system to work in a manufacturing scenario.For the scope of this project we will use the manufacturing scene provided in ARIAC-2019.
Key words: ARIAC, Moveit,Object Oriented Programming,Agility.

Contents

List of Figures

1 Part A
Team 4 and ARIAC-2019

In this section we will discuss about Team 4 - ARIAC-2019 approach briefly. We will examine topics including world modeling, sensors system, AGVS and robot control, sensor interactions and other crucial components of the environment.

1.1 Approach Overview

Our approach was based on deliberate planning process in which a complex main task was decomposed into smaller sub-tasks(planning each sub-task), and actions were executed to achieve the goals led in these sub tasks. The Figure below give the task break down.
As observed in the figure that the main task is "Order Completion". This task can be further

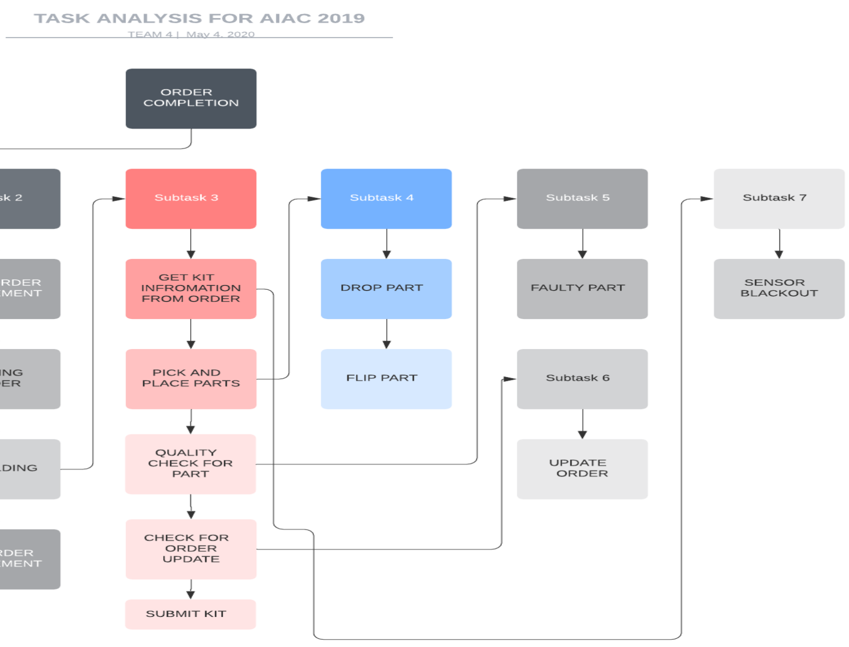

Figure 1.1: Task analysis

broken down into high level sub-tasks like:

1. Starting Competition
2. Fetching order
3. Refining Order
4. Ending Competition

We can further break "Refining order" into lower levels :

1. Starting Order Refinement

2. Reading order
3. Kit Building
4. Order Refinement Completion

While building a kit the system can experience undesired situations like sensor breakdown, order-update etc. All these challenges described for ARAIC-2019 are addressed in Kit building task, before submitting the kit.

1.2 Program Flow & Architecture

In this section we will discuss about the Architecture we followed and the program flow. We have followed the hierarchical task planning taught in the Planning lecture. We have divided the main task into three main categories:

1. Planning : Dividing the main task into sub-task and ordering it into levels.
2. Perception & Model: Placing and recording the sensor outputs
3. Acting: Lower actions performed by functions

The Planning section has:

1. Pick-up Order & Update Order: It specifies where different parts are and helps in forming a "part-station" which has all the parts needed to fulfill the order.
2. Pick-up Part: It specifies information about robot selection, moving robot to pick-up location, and other aspects like collision avoidance etc.
3. Placing Part: It specifies information about the drop location. Additionally, it handles all the challenges for ARIAC 2019. There are sub-task and action for each challenge for example drop part, flip part etc . By combining Pick-up part, placing part and pick-up order we solve the update order challenge.

The Perception & Model section deals with selection of sensors and placement of sensors.The main motto of this planning aspect is to form a minimal cost robust model of environment.

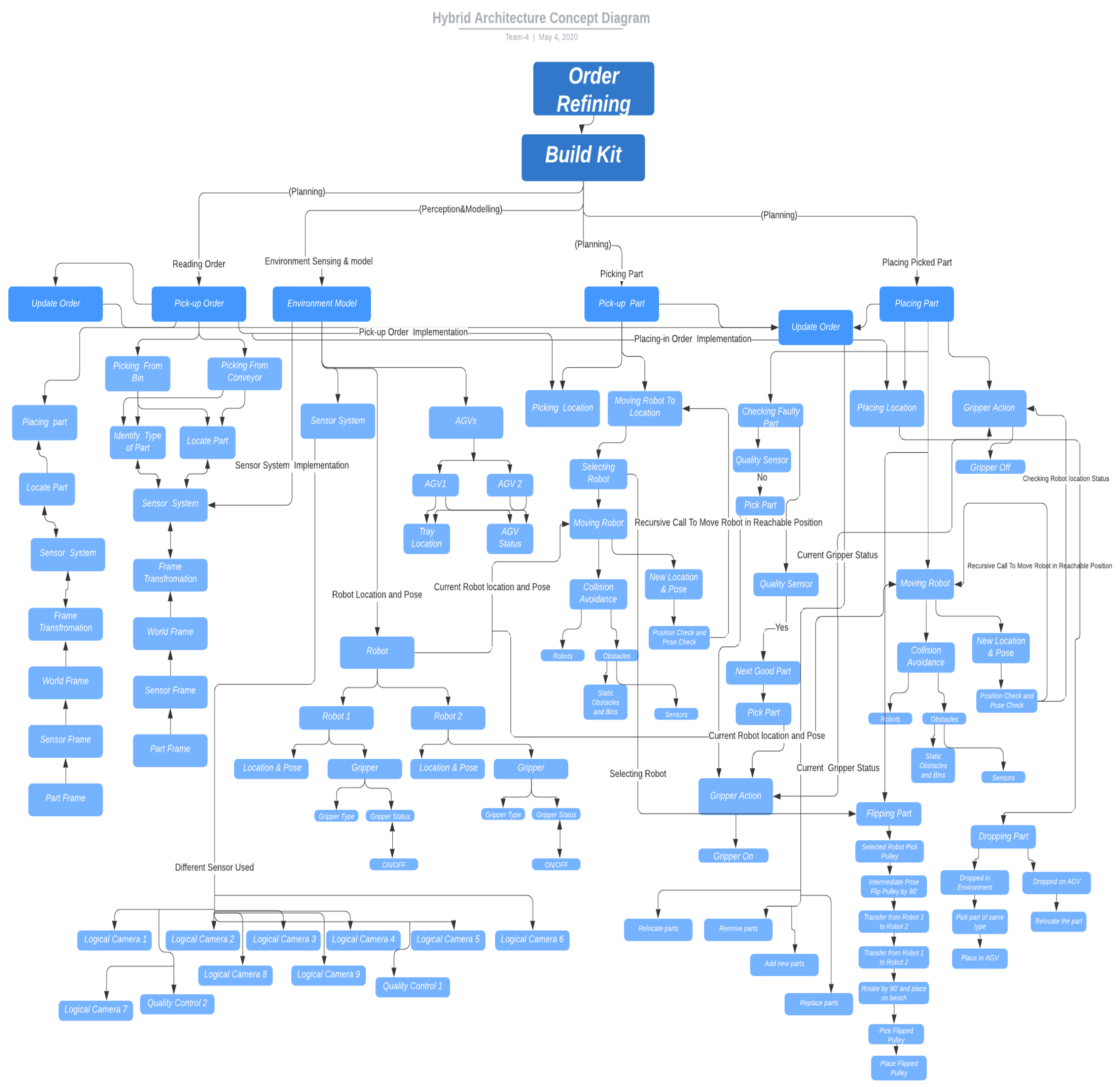

Figure 1.2: Architecture

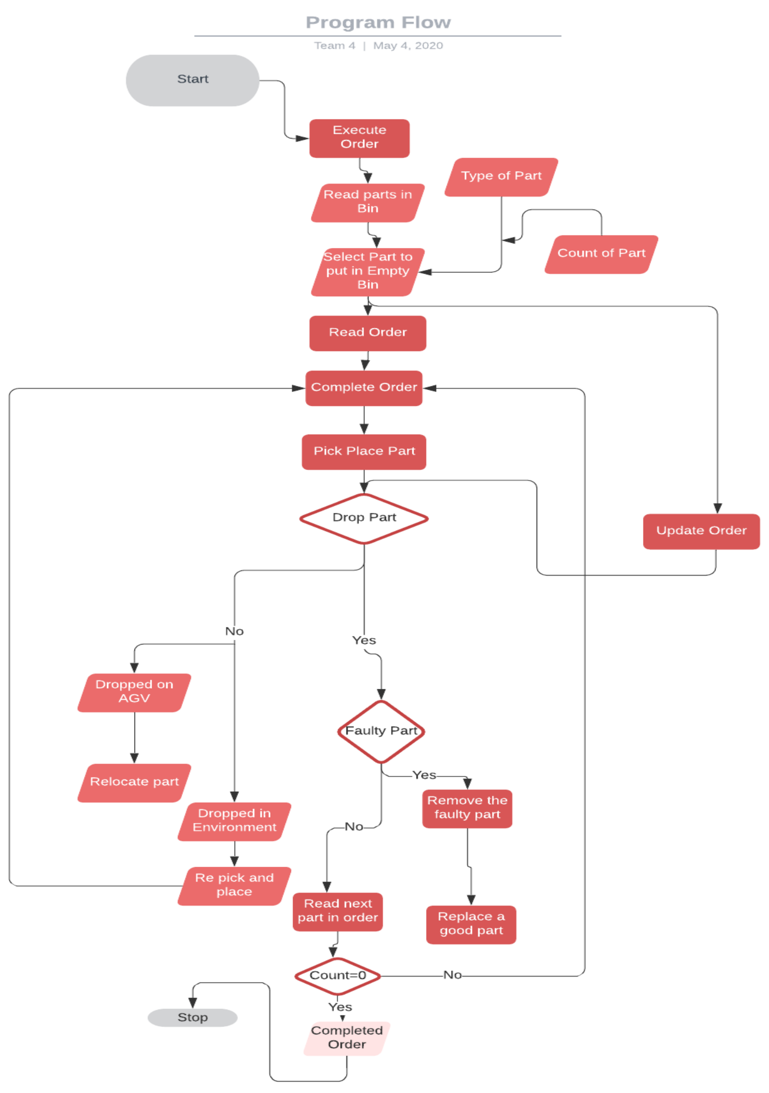

Figure 1.3: Program Flow

2 Part B
Challenge 1: Flipping Part

This section will cover the first challenge "Flipping Part". This section can be further divided into three sub-sections. First subsection, is challenge elaboration where we describe the challenge. Second subsection, covers our approach to solve the problem and finally, we will cover the difficulties we encountered while solving the challenge.

2.1 Challenge Elaboration

In this challenge our main task was to flip the given part before submitting it on to the AGV. For the scope of this project we had to only flip the pulley part.Additionally, it was noted that the pulley part had a flat collision surface on its top and bottom end making it ideal for grasping it with a vacuum gripper.Teams are not permitted to directly grasp this product from the side when a product flip is required. Moreover,it was observed that the the part that needed to be flipped had a roll value of π

2.2 Challenge Solution

1. Robot1 has pulley in its gripper which it needs to flip.
2. Robot 1 moves to an intermediate position and orients itself such that the pulley is rotated by 90 degree.
3. Robot 2 takes a pose that is mirror image of the pose of Robot 1.
4. Robot 2 is moved towards Robot 1 such that it can reach the pulley gripped by Robot1.
5. When Robot 2 has grabbed the pulley ;Robot 1 turns its gripper off.
6. Robot2 further rotates the pulley by 90 and places it on the bench such that it is in the reachable space of both the robots.
7. Depending on the Order Robot 1 or Robot 2 picks the flipped pulley and puts in the respective AGV.
8. We implemented a user defined function called Pulleyflipper() to accomplish the task.

2.3 Difficulties Faced

Fixing at a common pose and orientation was a tedious task as we had to try out many pose and orientation.

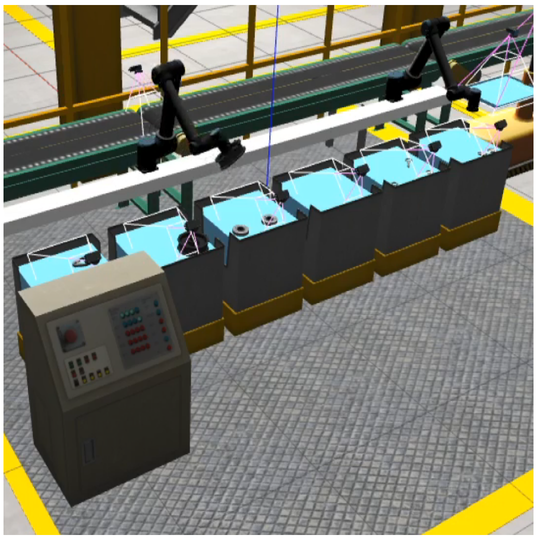
Figure 2.1: Robot 1 pick pulley

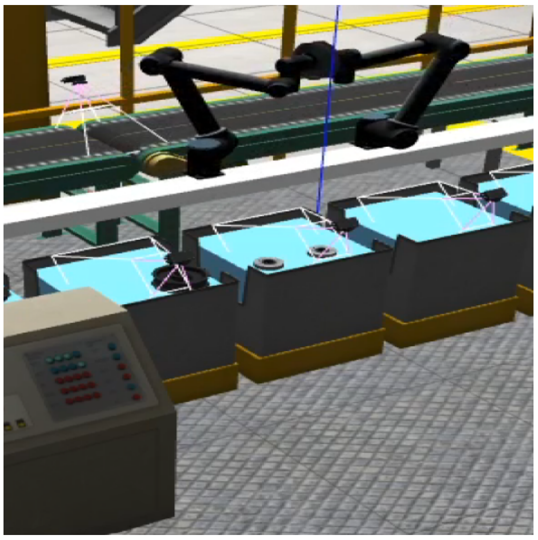
Figure 2.2: Robot 1 transfers pulley to Robot 2

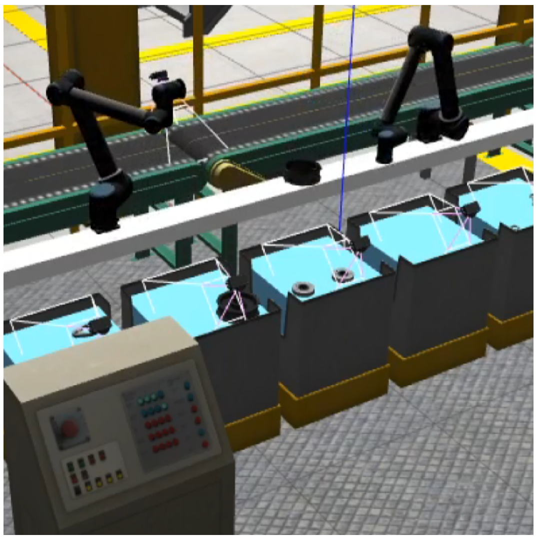
Figure 2.3: Flipped Pulley

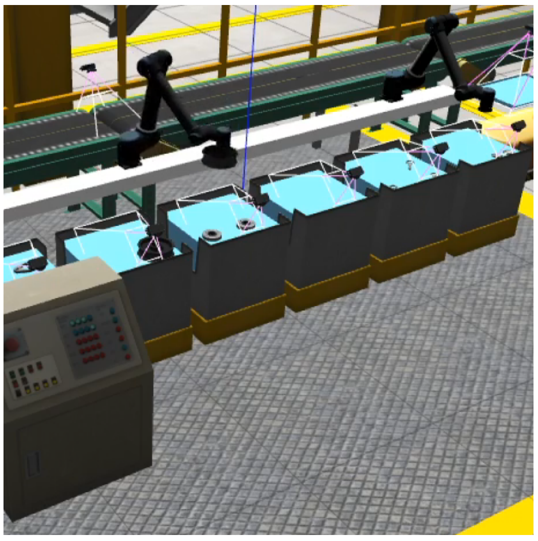
Figure 2.4: Robot 1 re-picks the flipped pulley

3 Part C
Challenge 2: Faulty Part

This section will cover the second challenge "Faulty Part". This section can be further divided into three sub-sections. First subsection, is challenge elaboration where we describe the challenge. Second subsection, covers our approach to solve the problem and finally, we will cover the difficulties we encountered while solving the challenge.

3.1 Challenge Elaboration

The main focus of this challenge is to remove faulty parts from the kit as they should not be used to fulfill the order. To address the task quality control sensors are attached above each AGV. Additionally, users cannot specify the locations of these sensors as they are set by ARIAC-2019.These sensors publish the pose of faulty products that they see on the tray. The output tf frames of faulty parts that the quality sensors provides can be used to remove faulty parts from the order.

3.2 Challenge Solution

1. Robot1/Robot 2 picks part from the desired location based on the order.
2. It takes it to the respective AGV and places it on the tray.
3. The Quality sensor attached to the AGV checks for the faulty part.
4. If the part is found to be faulty, then the Robot1/Robot 2 picks the part from the tray and drops the picked faulty part in the environment.
5. Robot 1/Robot 2 replaces the faulty part by non-faulty part at the end of shipment.
6. We have used ariac topic ariac quality sensor1 to find out the faulty part for orders on AGV1 and quality sensor2 for orders on AGV2.

3.3 Difficulties Faced

We had planned to scan for the faulty part keeping the part attached to the robot.But it was found that the faulty parts get detected only after it is placed on the AGV. Thus, we had to re-plan our solution.

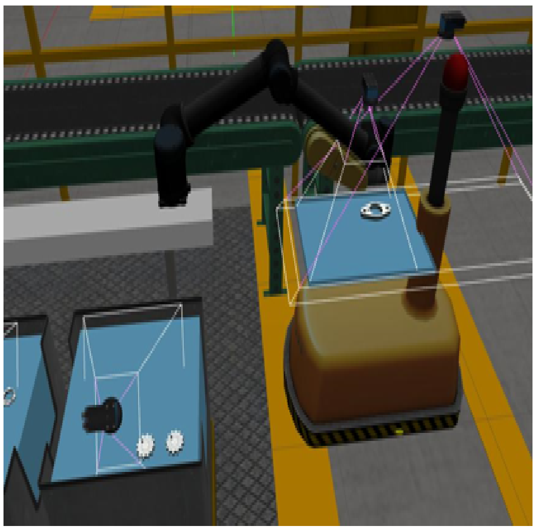
Figure 3.1: Placing Faulty Part on AGV

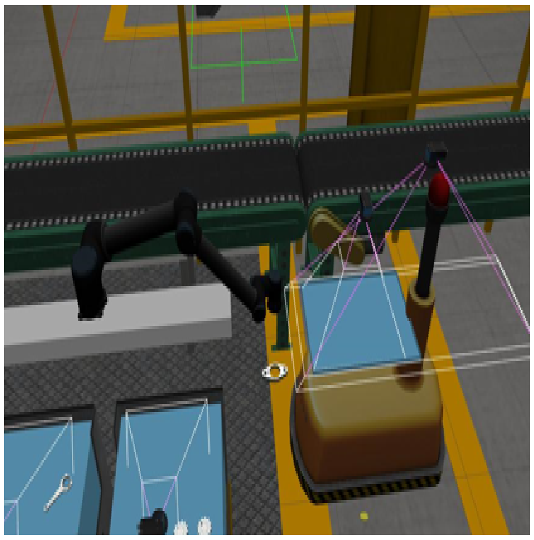
Figure 3.2: Removing Faulty Part

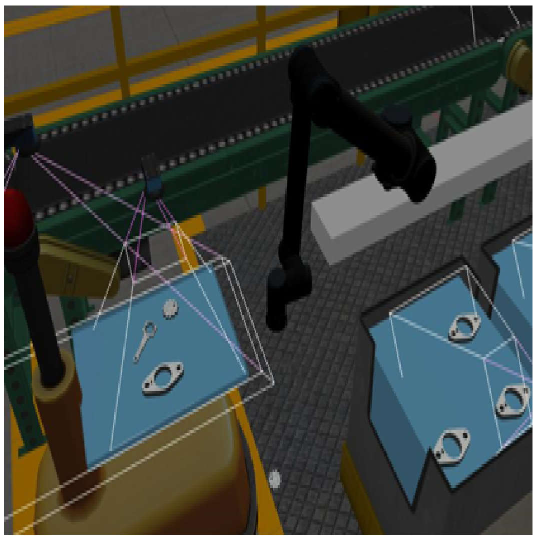
Figure 3.3: Replacing Faulty Part

4 Part D
Challenge 3: Drop Part

This section will cover the third challenge "Drop Part". This section can be further divided into three sub-sections. First subsection, is challenge elaboration where we describe the challenge. Second subsection, covers our approach to solve the problem and finally, we will cover the difficulties we encountered while solving the challenge.

4.1 Challenge Elaboration

This problem focuses on a situation when gripper becomes faulty. Faulty gripper may result in dropping parts above AGVs when the robot is trying to place the parts in trays.One part will be dropped above each AGV. Dropped parts will still end up in the tray but in the wrong pose. You will need to re-position those parts.

4.2 Challenge Solution

1. Robot1/Robot 2 picks part from the desired location based on the order.
2. It takes it to the respective AGV and tries placing it on the tray.
3. While placing it at the desired orientation specified in the order, some extraneous event occurs, which causes the part to be dropped either on the tray(unwanted location) or some where in the environment.
4. Next, we check whether the part is dropped on the tray or else where , if it is dropped on the tray we pick the part and reorient it to the correct location.
5. If the part is dropped in the environment, we pick a new part of the same type and place it on the tray.
6. We implemented a user defined function called Droppart() to accomplish the task.

4.3 Difficulties Faced

When sensor blackout occurs it becomes difficult to find whether the part dropped at wrong location on the tray or elsewhere in the environment.Thus, the act of picking new part to compensate for missing part or re-orientation of the existing part becomes uncertain.

5 Part E
Challenge 4: Sensor Blackout

This section will cover the fourth challenge "Sensor Blackout". This section can be further divided into three sub-sections. First subsection, is challenge elaboration where we describe the challenge. Second subsection, covers our approach to solve the problem and finally, we will cover the difficulties we encountered while solving the challenge.

5.1 Challenge Elaboration

During a certain period of time all sensors will stop reporting data. You need to take necessary actions i.e. wait until the sensors are working again before continuing,or continue building kits during the blackout(recommended).

5.2 Challenge Solution

1. We initiate a variable in the call back function of logical camera, that stores the time stamp.
2. We have implemented an user defined function called Sensorblackout() that compares the current time stamp and the value stored in the variable in the call back function.
3. If the difference between the value stored in the variable (form call back) and the current time stamp is greater than a threshold we return True and detect a sensor black out.

5.3 Difficulties Faced

Setting a robust value for threshold was difficult task.

6 Part F
Challenge 5: Update Order

This section will cover the fifth challenge "Update Order". This section can be further divided into three sub-sections. First subsection, is challenge elaboration where we describe the challenge. Second subsection, covers our approach to solve the problem and finally, we will cover the difficulties we encountered while solving the challenge.

6.1 Challenge Elaboration

The order can be updated in between the process.The system must response to the updated order.

6.2 Challenge Solution

1. Detect different parts on the tray through the logical camera.
2. Read the new order and find the difference between the old order and new updated order.
3. For updating the order we have following cases :
    a) When the number of parts and type of parts are same in the new and old order, but the relative position of parts are different in both the orders.
    b) When the number of parts are less in the new order than the old order.
    c) When the new order has more parts than the old order.
    d) When order has different parts then the previous order.
4. If the new order is of the type "a", we perform swapping between the parts.
5. If the new order is of the type "b", we throw unwanted part form the AGV tray out in the environment.
6. If the new order is of the type "c", we add new parts.
7. If the new order is of the type "d", we replace unwanted parts by new desired parts.

6.3 Difficulties Faced

We faced challenge while re-positioning the part on the tray. There were scenarios where the correct position for the part was already occupied by some other part. Thus, we had to relocate the part that pre-occupied the correct position for the mentioned part in update order to some other location. This increased the complexity of task.

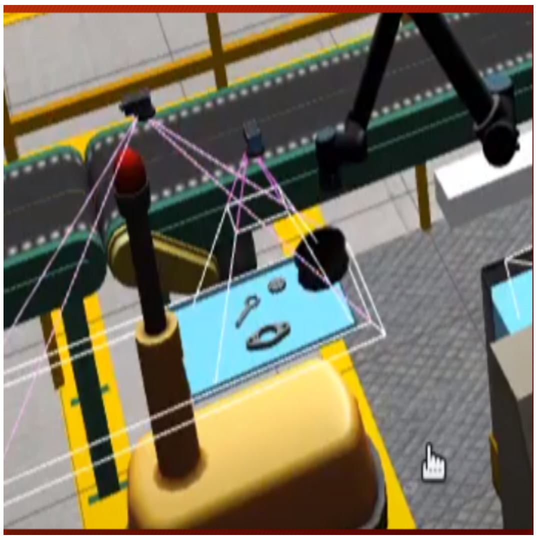
Figure 6.1: Old Order

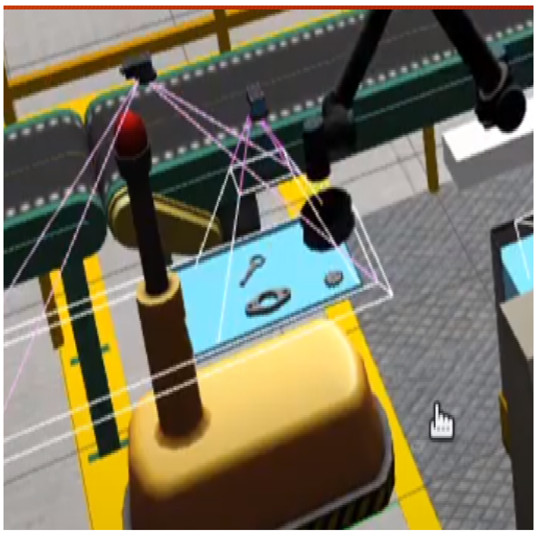
Figure 6.2: Updated Order

7 Part G
Project Contribution

7.1 Aalap Parimalkumar Rana

1. Challenge Flipping part -Pulley Flipper
2. Part Availability
3. Task level Planning
4. Report and presentation

7.2 Ishan Patel

1. Challenge Flipping part -Pulley Flipper
2. Part Availability
3. Task level Planning
4. Report and presentation

7.3 Patan Sanaulla Khan

1. Challenge Sensor blackout
2. Report and presentation
3. Debugging and testing code
4. Exception handling

7.4 Pranav Jain

1. Challenge Faulty part
2. Challenge Update order
3. Report and presentation
4. Debugging and testing code

7.5 Kulbir Singh Ahluwalia

1. Challenge Faulty part
2. Challenge Drop part
3. Report and presentation
4. Debugging and testing code

8 Future Works & Feedback

8.1 Feedback

The class was highly informative and was a great learning experience for the team. The team invested a huge amount of time learning concepts of ROS, C++ programming ARIAC interface and MoveIt. This provided a good understanding of the software and debugging the code written for ARIAC. There were many new things that the team learned during the course work. This included the following:

1. We were able to have hands-on experience of ROS.
2. We implemented Object Oriented Programming concepts in C++.
3. We were able to use various functions of MoveIt.
4. Knowledge representation and Ontology.
5. Planning and Acting in ARIAC.
6. Coding a planner in PDDL.
7. Different experiments and test beds to evaluate a system.
8. Different components of measurement science and how they interact with each other.

The team initially found the challenges difficult as it lacked experience in ROS and had little experience in C++ programming, but by the end of the course everyone was able to contribute an equal share to the project and gained the same level of proficiency. The team enjoyed spending sleepless nights working on the errors and making the system agile, it did get frustrating some times when we missed few scenarios and had to re plan the whole code to include the solution. However, this course not only taught the team technical skills required for a great professional life but also taught many important other skills like team-work, conflict resolution, decision making and has cratered in forming a strong bond among the teammates. I would like to thank Prof. Craig Schlenoff and Prof. Zeid Kootbally for making the course interesting and smooth flowing. Here are few suggestion we would like to have for the future:

1. More lecture on PDDL and a programming assignment in PPDL.
2. It would be nice if we get few pre-recorded lectures on ROS and C++ basic concepts prior to the start of the course so that more time can be allocated to Planning and Acting.
3. There should be a garbage collector bin for faulty parts rather than throwing the parts in environment.
4. The gripper can be changed to a more complex gripper which needs in depth understanding of concepts of grasping and modeling.

8.2 Future Work

1. We would like to implement Hybrid architecture by programming the ARIAC challenges in PDDL.
2. We have used logical cameras which are expensive sensors and would like to try out other sensor to reduce the cost of modeling the world.
3. At present the code does not use multi-threading concepts of ROS. We would like to further reduce the run time of the system by operating both the robots simultaneously.

References

[1] Lecture videos

[2] Lecture power point presentation

[3] https://bitbucket.org/osrf/ariac/wiki/2019/Home