ENPM809B

FINAL REPORT

Building a Manufacturing Robot Software System

UNIVERSITY OF MARYLAND

A. JAMES CLARK SCHOOL OF ENGINEERING

SUBMITTED BY:

ABHISHEK KATHPAL

GROUP-3

UID: 114852373

SUBMITTED TO:

PROF. CRAIG SCHLENOFF

PROF. ZEID KOOTBALLY

CONTENTS

1) GROUP3 ARIAC APPROACH

1.1) OVERVIEW:

The basic goal of the ARIAC competition is to build the kits and fulfil the order. The first task is to break down the entire task into sub-tasks. The qualifiers provided were the perfect stepping stones for the same. Instead on focussing on solving the complete qualifier, first we tried to solve the small tasks and how to deal with them.

Some of the **unique elements** of our approach:

- Flipping the parts
- Overall hybrid architecture and for conveyor pickup reactive architecture.
- Pose Correction
- Computing the velocity of conveyor and updating part positions on conveyor

The next step after breaking down the tasks was to decide what sensors to use in order to best achieve the results. Our main goal was to focus on the **agility**, being flexible to handle various scenarios, of the process rather than lowering the overall cost of the environment. Therefore, we have used five logical cameras, four on the bins and one on the conveyor. Using logical cameras gave us the ability of handling scenarios like multiple parts on the same bin as well as different configurations of the parts. We decided not to put cameras on the other four bins since the robot cannot reach in configuration to pick parts from those bins. The camera on the conveyor is not only being used to get the part type and position, but we are also using to estimate the velocity of the conveyor since it is not constant. This computed velocity is used to update the part positions on the conveyor with time.

1.2) DETAILED DESCRIPTION:

Our approach can be seen from Fig. 1, how we solved the problem and broke it down. The highest-level task is fulfilling any kind of order that is announced. It can be broken down to subtasks like starting the competition, parsing the order, processing the order, and ending the competition. Going with this logical flow we further broke down order processing to individual kit processing which can be further broken down into:

- Get kit info
- Get Parts
- Process Individual Parts
- Quality Check
- Check for in-process kit change
- Submitting Kits

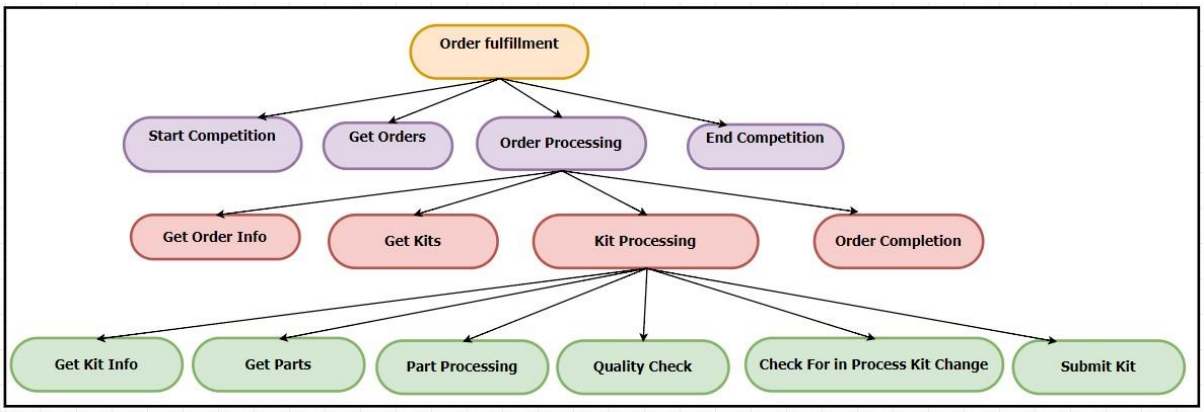

Fig. 1. Group 3 approach

The architecture used is shown in Fig. 2. According to this breakdown, we modelled the world in the code. The knowledge representation concept taught in the class came in handful while representing the world in the code. After representation of the world was finalized, we moved to jotted down the scenarios that needed to be tackled for the successful completion of the competition as follows:

- **Basic pick and place operation:**
  This is one of the most basic tasks for the competition. We have used two different approaches to complete this task:
  1) Cartesian Path - For picking and placing the parts, we used cartesian path planner provided by MoveIt! We used the ariac scene in order to get collision free path.
  2) Joint Angles - We used joint angles when we were sure that there was no collision along the path. The primary reason for using the joint angles is to improve the cycle time.
- **Getting the information from sensor:** This is another of the important step in the pipeline. We are using logical cameras and converting the position in the world frame so that it can be used by the robot.
- **Kit building:** The basic kit building process is very simple. Using the information obtained from sensor, the algorithm gets the position of the parts that can be used by the robot. The robot picks the part and puts it on the AGV.
- **Quality checking:** Before submitting the part on the AGV, we are checking the output of the quality sensor. If the quality of the part is not as required, then we are throwing it off the AGV instead of putting the part on it.
- **Dropping parts on the bin**: Whenever there is a failed pick, we are adding it to the list. At the end of the processing the order, we are processing this list to fulfill the complete order.
- **Picking the parts from conveyor**: We have a logical camera on the conveyor that we are using to keep track of the parts that are coming on it. We are also using this camera to estimate the velocity of the belt. This computed velocity allows us to update the position of the part with time. If the part is in the order and it is within the range of the robot, the robot will wait at the fixed position and pick the part as soon as it comes.

- **Processing higher priority order**: Currently we are blindly processing the higher priority order as soon as it is announced and then continue with the order that was being processed earlier.
- **Flipping the part**: Before picking the part from the bin, we are checking the orientation of the part. If the roll or pitch of the part is more than or equal to 100 degrees, then we flip the part on the bin itself and then pick the part again. We are able to pick the part again after flipping because of the cameras on the bins which allow us add the part to the inventory.
- **Pose correction**: We are also using the orientation of the part from the sensor to compute the offset in yaw with respect to the drop required drop position. This correction in this yaw helps us to get the part pose bonus at the end of the competition.

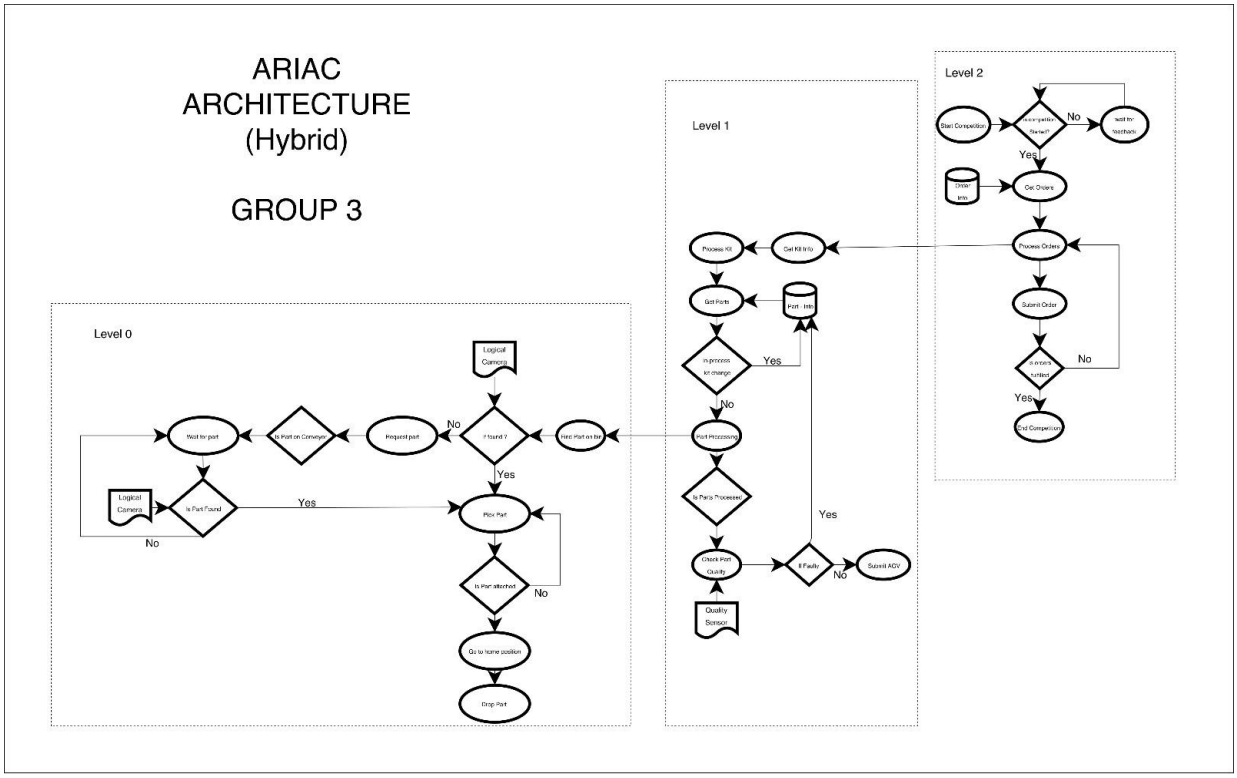

Fig. 2. ARIAC Architecture

2) INDIVIDUAL CONTRIBUTION

Group 3 has 4 members: me (Abhishek), Neel, Anirudh and Gunjan. All of us have valuable contributions in achieving the final results for Agile Robotics Industrial Automation Competition (ARIAC). In this section, I am listing down some of my contributions that I made throughout this competition:

- **Order Manager Class:** After getting the scanned list from the camera, I have written the order manager class to use the list of scanned parts to fulfil the order. The order list is retrieved from "/ariac/orders" topic. As soon as the part is picked from bin, that part will be popped from the list of scanned parts. Kit and part type are retrieved from that topic, which are further used as an argument while submission of AGV.

- **In-Process Kit Change:** To solve this scenario, I used "ros::spin()" to get the updated list from the callback of "/ariac/orders". After picking of every part, it will check whether any new order arrived or not. A new order can be detected from the change of order name in the message of the above-mentioned topic. Once a new order is there, it will consider that as higher priority and will working to fulfilling that first by making kit on other unused AGV.

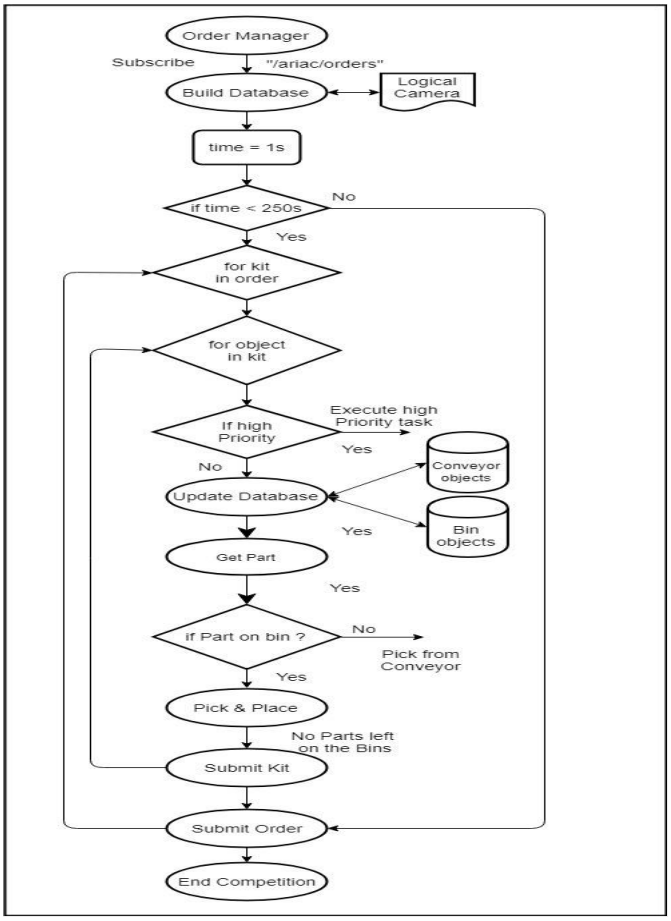

Fig. 3. Order Manager Flow Diagram

- **Picking Conveyor Part (Approach 2):** Initially, the code from picking parts is written by Neel, the approach is more mathematically accurate but the results we are getting were not that good because it was slow and the UR10 was only able to only pick one part from conveyor. So, I suggested different approach which uses some part of Neel's approach to get the velocity and updated parts list. Instead of varying the y-position as part moves, I implemented the algorithm to go to a specific future position of part and wait for part to come. As soon as the part comes below the arm, according to the predicted position, it will go to a lower temporary position which varies depending on height of part to be picked.

- **Adding Part Offsets while picking:** We are facing the issue to properly picking the part because of their different heights, so me and Anirudh computed different offsets for each part and integrated that in our Order Manager class.

- **Giving higher priority to conveyor:** Initially, we were giving priority to bins over conveyor. But later I realised that if we do this, we are going to miss all the parts from conveyor. So, I wrote the code for setting priority to conveyor if there is part on both bin and conveyor and if there is no part on either bin or conveyor it will wait for part to appear on conveyor.

- **Part Pose Correction:** Another important contribution is the ability to place the parts with correct position and orientation. The drop position and orientation were received in the orders topic. From that, I got the target position and orientation but that was with respect to AGV. Using TF, I found the correct drop pose of the part. To rotate the part, I converted the quaternion of drop pose into roll pitch and yaw and then generated the new quaternion using a fixed roll, a fixed pitch and the computed yaw from drop pose to orient the part properly.

- **Improving Speed by using joint space:** Initially, we were using waypoints and computing the path in cartesian space. But that was slower that using joint state. So, me and anirudh tried to reduce the use of cartesian space only where the joint space was failing. Joint space was not working well when picking and dropping part. So, except these two stages, we switched all the other movements to joint space to make our robot faster.

- **Removing Cameras (Not Implemented in Final Submission):** As every team is trying to remove cameras, so me and Anirudh wanted to give that a shot. I was able to get all the parts list with bin number from get_materials_location topic. I made a map from whose key was part type and list associated with that part is the set of possible positions where the part may lie depending on the orientations we observed in the qualifiers. But later, all the members of group thought that this makes the system less agile. So, we decided to keep all the cameras. We need cameras to encounter possible scenarios like, multiple types of parts on bins and picking the dropped part on bin.

3) IMPROVEMENTS

To improve the performance of the code in different possible scenarios and making system more agile, the following improvisations can be done:

- In the current version, Kits are building on only one AGV at a time. Since only limited parts come on the conveyor, this approach is not suitable to build the kits in some scenarios since the parts on the conveyor were not there. Implementing this approach will give the ability to process the parts on the conveyor parts for both the kits simultaneously rather than processing the parts on the bin. This is the improvement which will make sure that no part of conveyor will be missed.
- The current approach of picking parts from conveyor is simple and the robot waits till the required part on the conveyor at the fixed position on the conveyor. Initially, Neel tried to use another approach in which robot travels to the part and follow it and slowly pick the part. That is a better approach but it was not implemented properly. If implemented properly, this will reduce the time as well as the error of picking the wrong part from the conveyor. This is one of most important improvements that should be made in the current implementation.
- Right now, higher priority orders are being processed by the robot without checking any condition. It should first check whether the previous order parts are on conveyor. If they are on conveyor, we should first pick that part and then start higher priority. This approach will make sure that all the orders are completely fulfilled and thus improving the scores.
- In the current implementation, parts dropped on the AGV are not considered to save the cost. The improvement for this is trying to pick the part around the position where the part was dropped in a circle. Once the part is picked then the type can be checked under the camera. This approach will increase the time taken to complete the competition but since this scenario is rare, it won't affect the performance of the code.
- Currently, the priority of picking parts is assigned to conveyor, so the robot will wait until the part appears on conveyor. This approach can be improved by processing the parts smartly i.e. skipping the part, adding the current part to the list, and processing the next part. Once the part appears on the conveyor then that part can be processed. So, this modification of skipping part in order helps to reduce save time.
- Instead of using one camera per bin, one camera can be used to detect the parts and the configuration on two bins. This will reduce the cost of the system without reducing the agility factor.
- Another improvement is to integrate the PDDL planner. It might reduce the effort to write code for any other future possible scenarios.

4) CLASS CONCEPTS

Most of the concepts that are taught in the course can be used for different problems. I myself used them in my planning course. Some of the class concepts that I find really useful are described below:

1) **ROS Concepts:**
   For me, this is the most important lecture. I think there should be more lectures related to Robot Operating System and MoveIt. The first assignment on TurtleBot was a good start to review the concepts taught in class. This lecture helped review all the concepts of topics, services , messages and actions.

2) **Knowledge Representation**:
   Knowledge acquisition and knowledge representation really helped to structure our code properly. After the Ontology assignment, we (group 3) find the relevant agents, variables and tasks. Using that data, we changed our code. We included different class for order manager, sensor and ur10 controller. This process made our code more readable.

3) **Architectures:**
   In the architecture lecture, there are many new and interesting stuff, that I learnt. Firstly, we wrote the code but we are not aware of what architecture we should be using and why it is better than others. After the architecture assignment, all the members of our group decided to go with the hybrid architecture and for conveyor pick up, we went for reactive architecture.

4) **PDDL Planner:**
   The task -level planning lecture by Prof. Zeid Kootbally was really interesting especially because of the hands-on work. The implementation of ARIAC competition using popf-tif-clp planner leads to modification of our functions. After this lecture, we made functions for high level planning work like gotoConveyor, pick and place etc. We tried integrating the planner, but we got our hands full with our work. It would have been interesting if we were able to integrate the planner with ROS.

5) FEEDBACK

This course has been a great learning experience for me. In the initial two weeks, I spent most of my time learning and understanding the concepts of Robot Operating System (ROS), MoveIt! (Motion Planning Framework) and C++-11. It would have really helped to have more lectures related to these three topics, as all the framework for the ARIAC code is based on these three things. I enjoyed learning about the Knowledge Representation, Ontologies and different types of architectures. This helped our group to write our code in a more structural way. Another lecture, that was really interesting is the one in which we used the Task-Level Planning Domain Definition Language (PDDL) planner. It was interesting to write the high-level planner to automatically generate the appropriate sequence of actions needed to achieve a particular goal. I will probably try to fully integrating the planner with our ARIAC code this summer.

I think this course is a bit challenging and demanding, but once you understood all the basics related to ROS, C++ and MoveIt, this course becomes a lot more fun and interesting. One thing I suggest is to include some of the content related to Machine Learning and Artificial Intelligence and how we can integrate that in ARIAC Competition. It would also have been better if the some of the initial assignments are more focussed towards C++ and ROS. I really enjoyed spending sleepless nights working on the code to make the UR10 as agile as possible, considering various scenarios where the robot could fail and how to solve them. Sometimes it also gets frustrating, when for no reason Gazebo keeps spawning the UR10 in wrong configuration and just because of that the implementation of our code didn't worked. Another thing I enjoyed, was working with my three other team members. It has been a roller-coaster ride, especially for me, as this was my first semester, but I really learned and implemented a lot of interesting stuff. In the end, I would really like to thank Prof. Craig Schlenoff and Prof. Zeid Kootbally for teaching this course and making it more interesting. In future, I will definitely participate in one of the ARIAC competitions. I am pretty sure the concepts taught in course would help me find solution for different possible failure scenarios.

I also have few suggestions for the ARIAC Competition:

- A bin should be present in the ARIAC Environment to drop the faulty parts there. This would be more realistic rather than dropping that faulty part anywhere.
- The cost metric should also consider an agility factor, which is computed from how well the system performed to tackle different scenarios like dropping part on AGV, bin etc.
- The "get_materials_location" to retrieve the bin and part info should not be allowed to use when competition is running.
- Flipping a part should carry more weightage in score in comparison to normal part orientation as it is a more computationally intensive task.