ENPM809B- Final Project

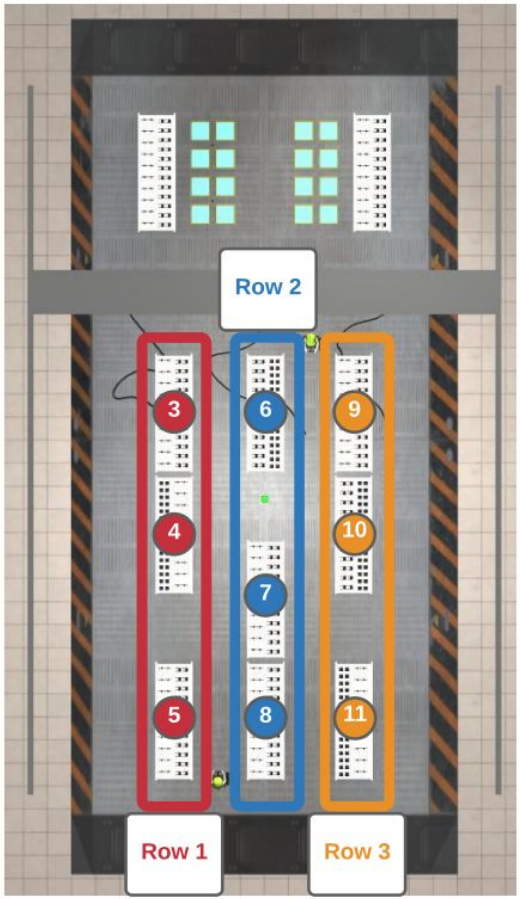
Group 4

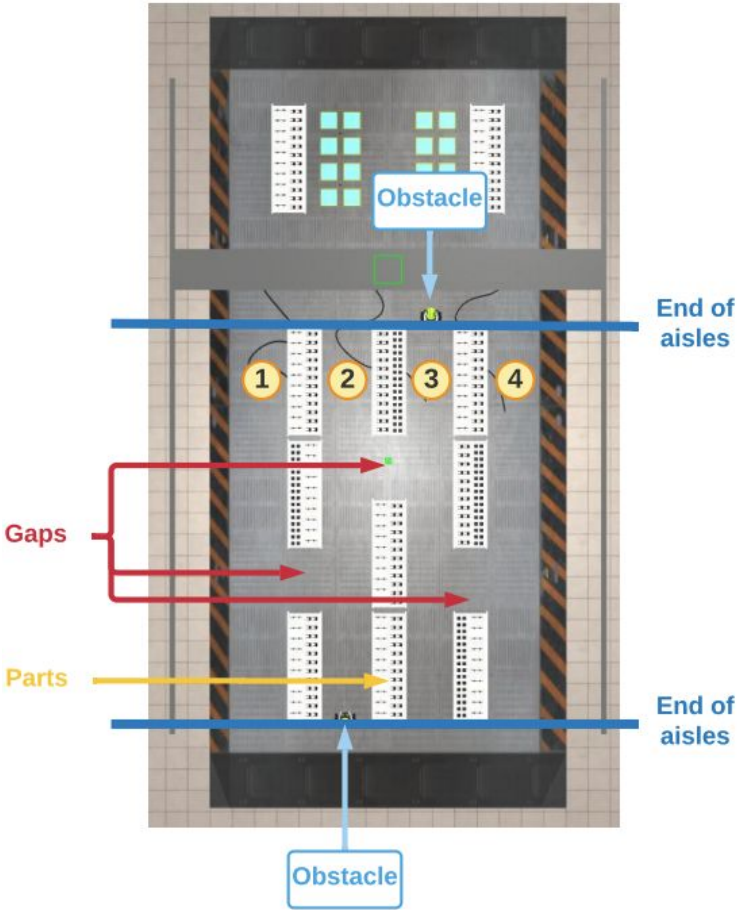
Markose Jacob - 117000269
Aditya Goswami - 116951968
Nalin Das - 116698290
Saumil Shah - 116745338
Varun Asthana - 116696500

ARIAC 2020 Competition Overview

- **Goal** - Build a kit in a simulated warehouse environment with a dual arm gantry robot
- Organized by National Institute of Standards and Technology (NIST)
- GEAR Software developed by Open Source Robotics Foundation (OSRF)

Environment Layout





Environment Setup

- Conveyor belt
- 16 Bins
- Robot (Gantry with two UR10 Arms Attached)
- 2 AGVs
- 11 Shelves
- Moving Obstacles
- Parts - Pulley, Disk, Piston Rod, Gear, Gasket (Colors - Blue, Red, Green)
- Sensors - Logical Camera, Quality Control, Break beam

We used 17 logical cameras, 24 break beam sensors and 2 quality control sensors

Software & Hardware dependencies

In order to run our code, we recommend the following dependencies

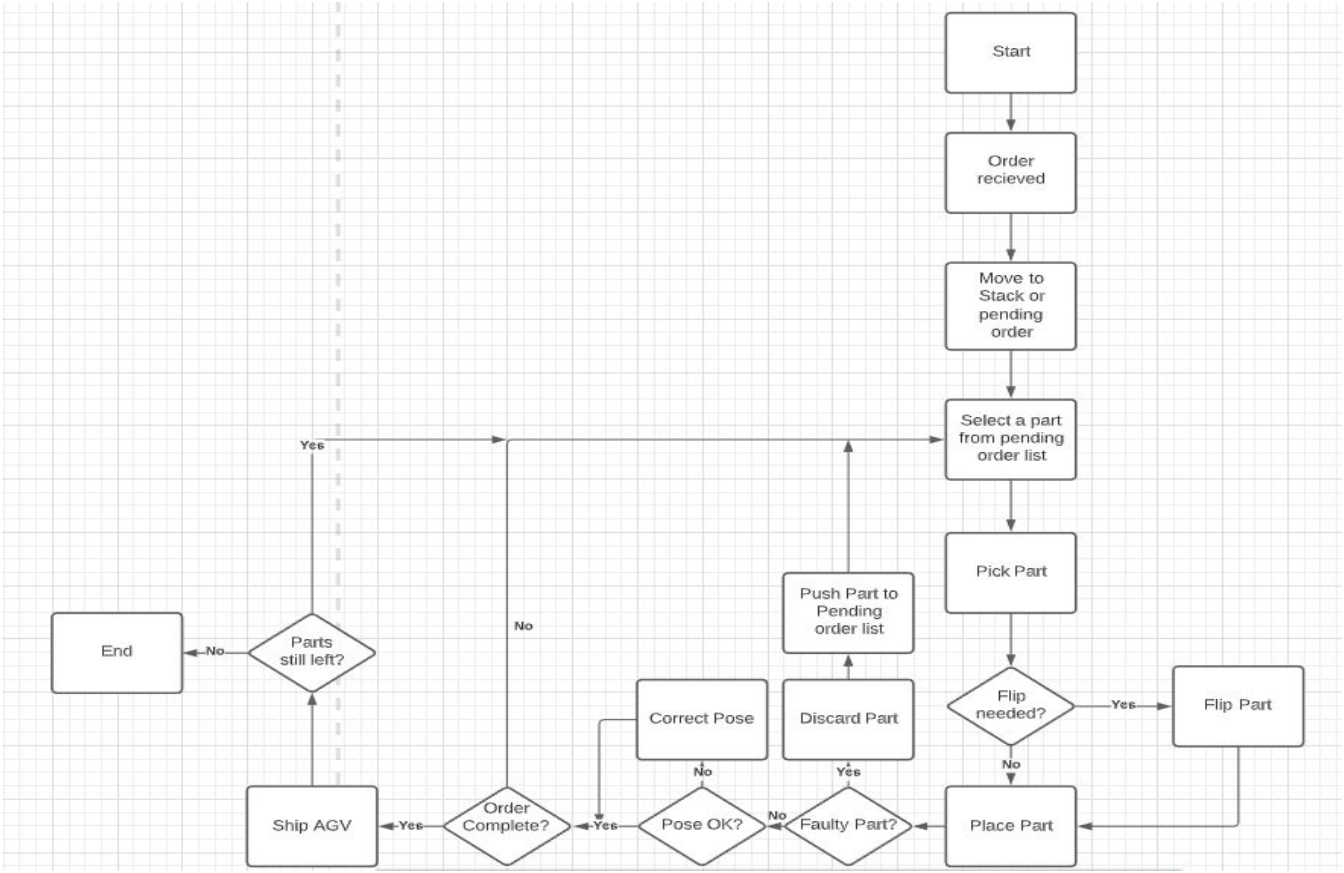
Required

- ROS Melodic
- MoveIt
- GEAR Software (ARIAC 2020)
- Ubuntu Desktop 18.04 Bionic
- Gazebo >= 9.14
- C++ 11/14

Preferred:

- Modern Multi-Core CPU, e.g Intel Core I5 or better
- >= 8Gb RAM
- Discrete graphics card, eg. Nvidia GTX 650 or better

Flow chart


Start
Order recieved
Move to Stack or pending order
Select a part from pending order list
Pick Part
Flip needed?
Yes
Flip Part
No
Place Part
Faulty Part?
Yes
Discard Part
Push Part to Pending order list
No
Pose OK?
No
Correct Pose
Yes
Order Complete?
Yes
No
Ship AGV
Parts still left?
Yes
No
End

Agility Challenges

1. Flip part
2. Faulty gripper
3. Faulty part
4. Moving Obstacle
5. Sensor Blackout
6. High Priority Order

Flip Part

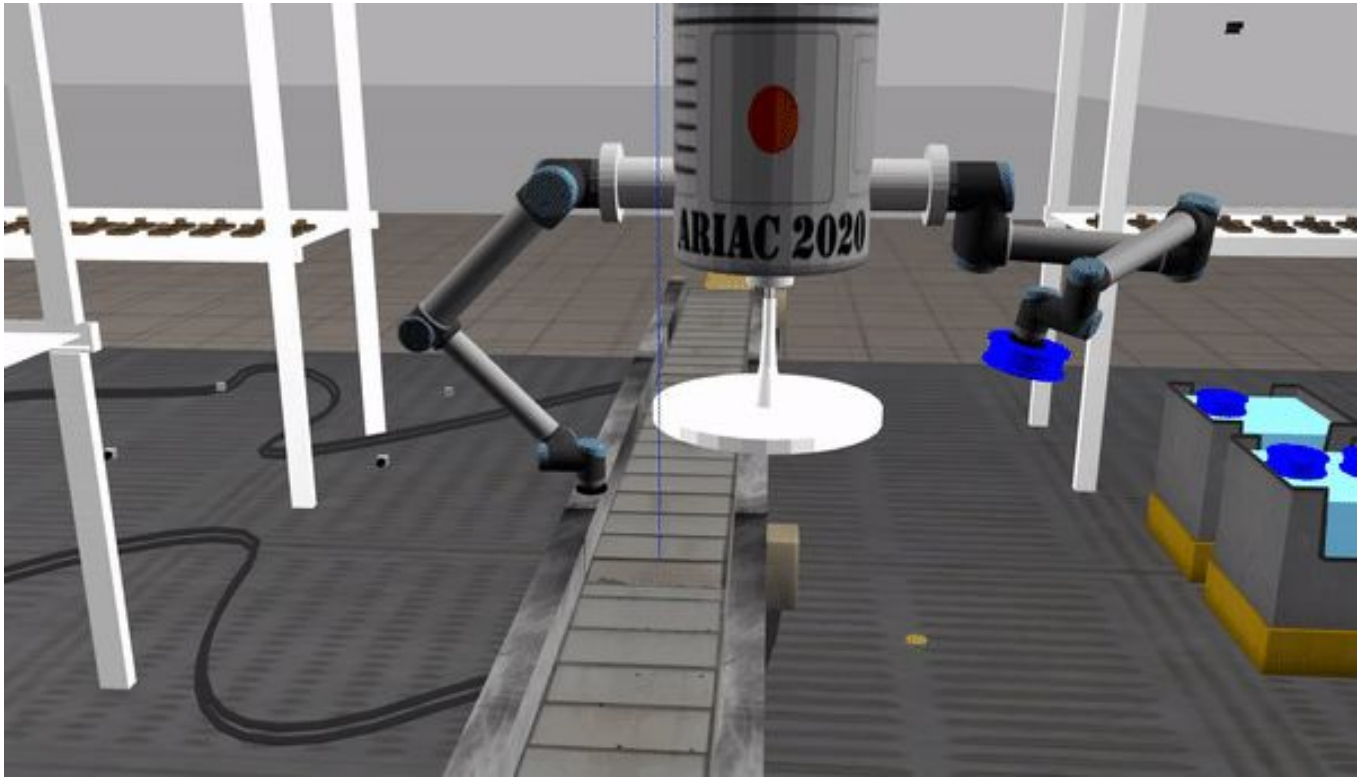
Detection:

- Get roll value from the target pose of the part
- If roll = pi, then the part needs to be flipped

Solution:

- Pick the part using the left arm
- Align the grippers of both the arms on the same axis, so that the right arm can grasp the part and the left arm can detach (done on top of the conveyor)

Flip Part



Faulty gripper

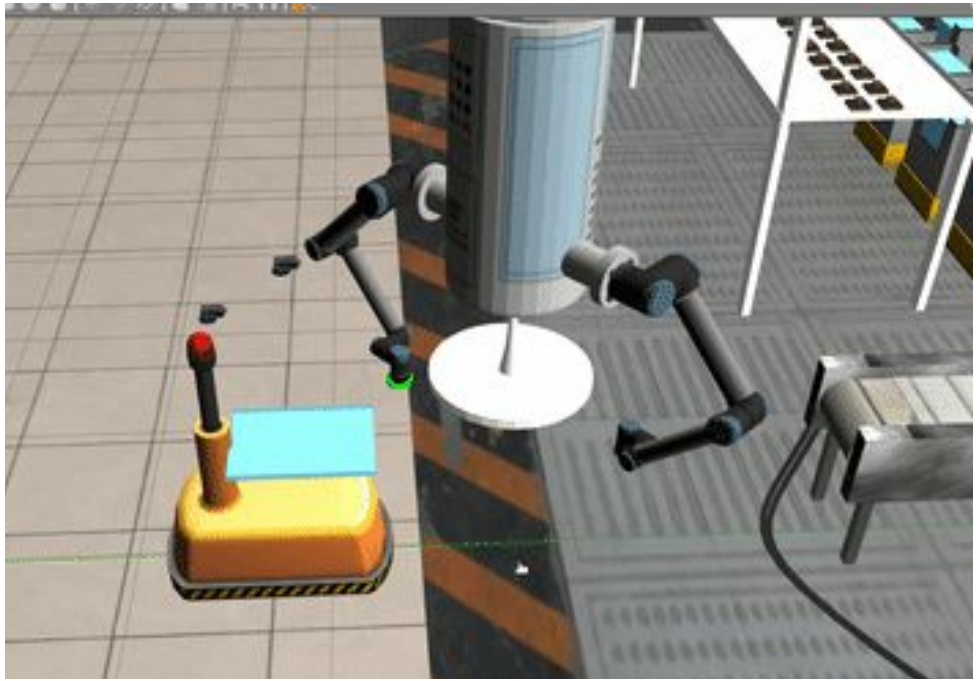
Detection:

- Subscribe to logical cameras on both AGVs, Get pose of dropped part.
- Compare the pose from the logical camera with the desired target pose
- If both are not equal, then it is a dropped (wrongly placed) part

Solution:

- Pick the dropped part and place it in the desired target pose
- Keep repeating this process until the pose (p1) from the camera and the desired target pose (p2) matches and are within the thresholds -
  - abs(p1.x - p2.x) < 0.03
  - abs(p1.y - p2.y) < 0.03

Faulty gripper



Faulty Part

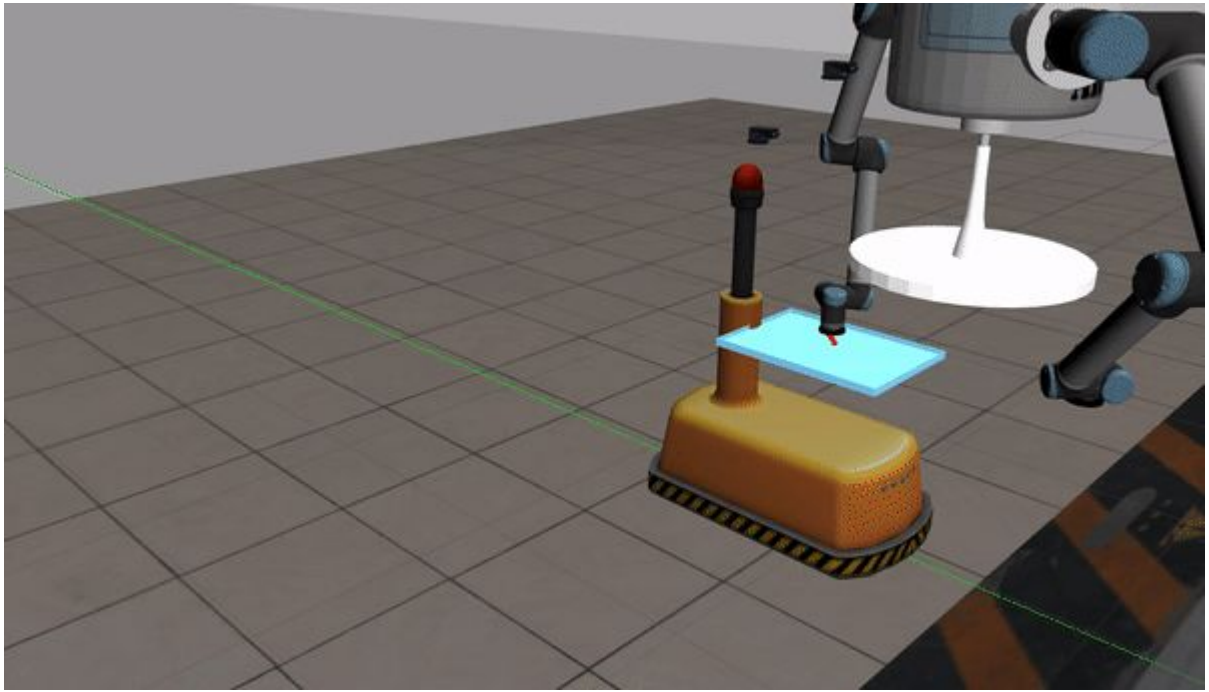
Detection:

- Subscribe to quality control sensor over both AGVs
- Output of sensor checked each time a new part is placed on the AGV

Solution:

- Pick the faulty part and discard it
- Add the part data into the pending order list
- Fetch the new part of the same type

Faulty Part



Moving obstacle

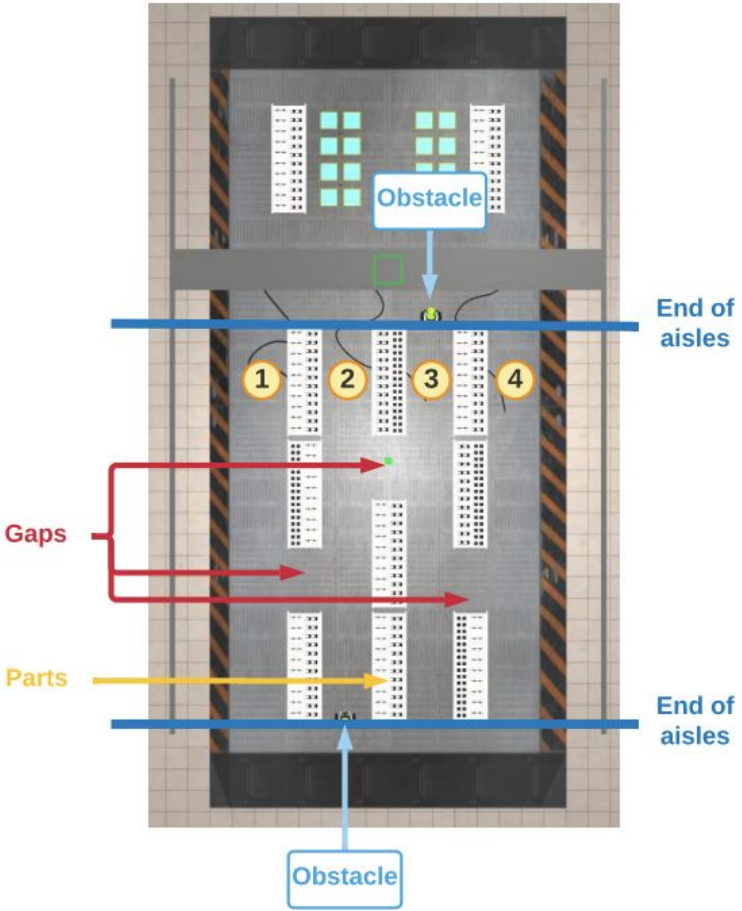
Detection:

- If moving obstacle is present in any aisle, then corresponding breakbeam value toggles.
- Based on the breakbeam sensor, we are able to detect which aisles are clear

Solution:

- Robot moves to the closest shelf gap (if required intermediate gap is also used)
- Once the Aisle clear flag is set, robot moves to pick the part
- Once the part is picked, then the escape behavior is triggered.

Moving obstacle



Sensor Blackout

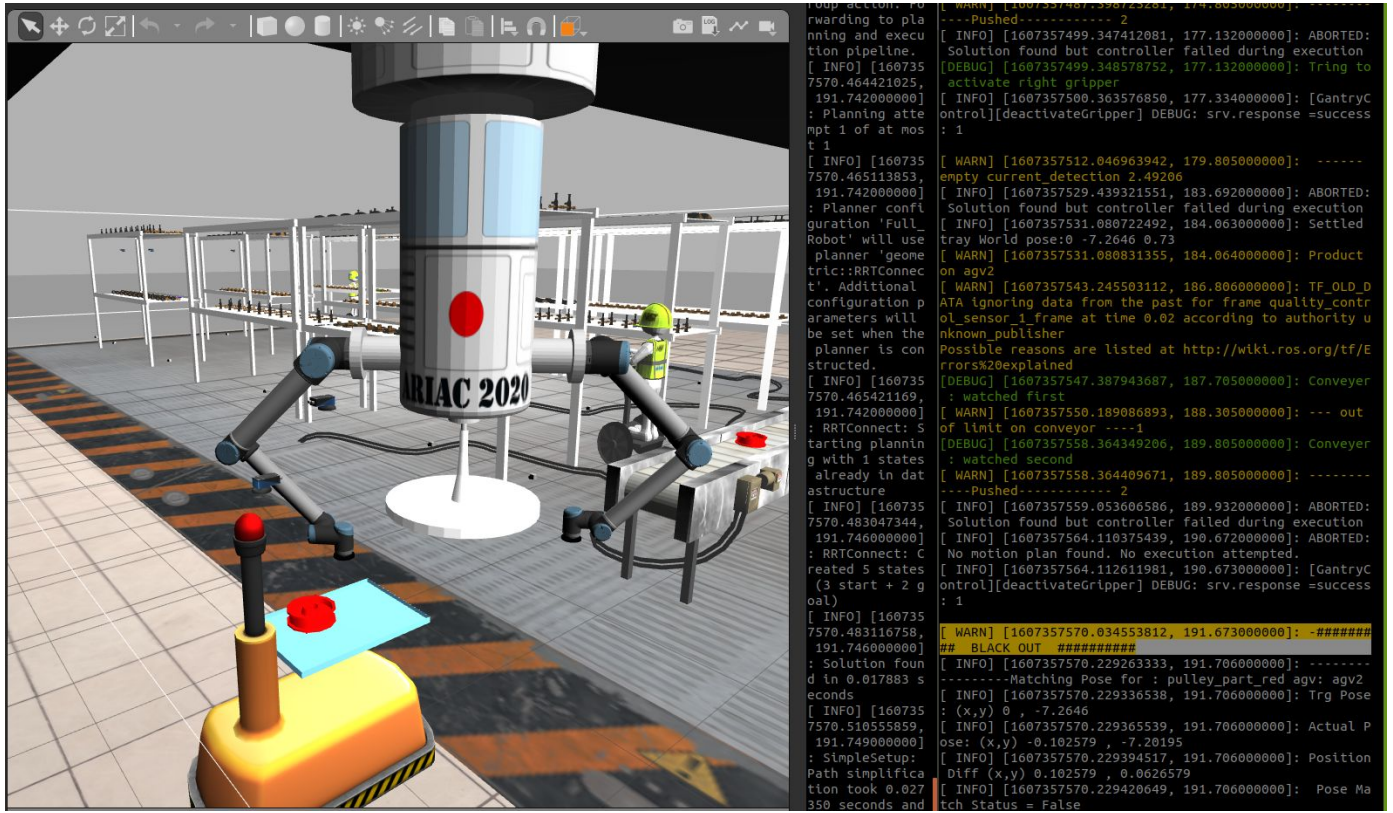
Detection:

- A counter is being updated in the quality control sensor callback
- If the counter value does not change, then classified as a sensor blackout

Solution:

- Robot halts in its current position until the sensor blackout is over (i.e counter value changes)

Sensor Blackout



High Priority Order (HPO)

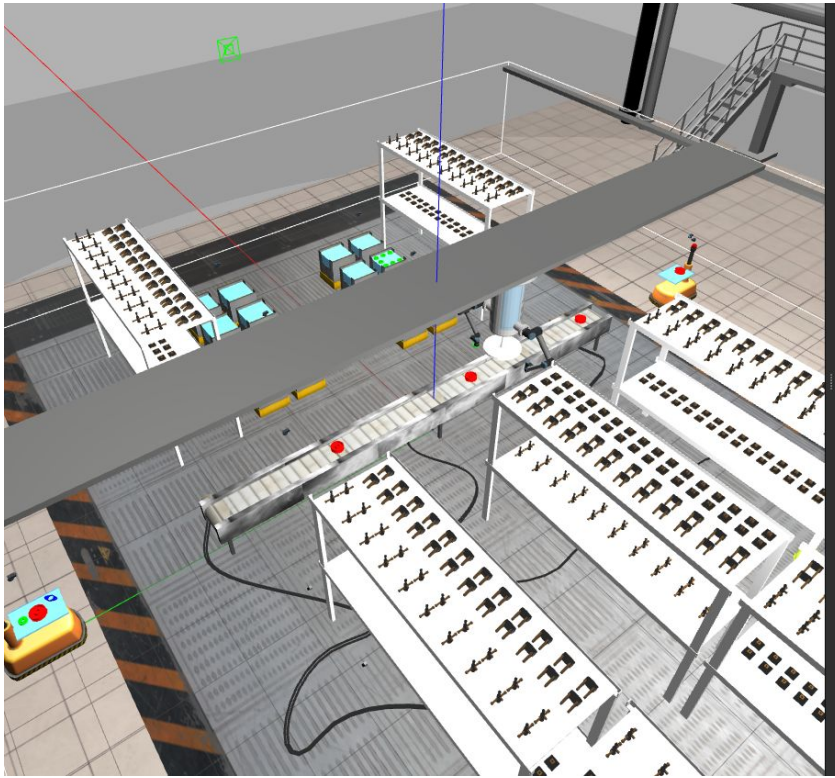
Detection:

- Second order is assumed to be a high priority order
- Any new order received is added to the pending order data structure

Solution:

- Use of linked list data structure which gives priority to the topmost element
- By using this, we are able to indirectly execute high priority order first
- Maximum priority is given to the conveyor parts
- If HPO has to be built on an AGV which is already in use, then we first clear the AGV, before we start building the HPO

High Priority Order (HPO)



```
reated 4 states
 (2 start + 2 g
oal)
[ INFO] [160735
7746.881039319,
 222.446000000]
: Solution foun
d in 0.018112 s
econds
[ INFO] [160735
7746.894128309,
 222.449000000]
: SimpleSetup:
Path simplifica
tion took 0.012
931 seconds and
 changed from 3
 to 2 states
[ INFO] [160735
7766.863670650,
 225.962000000]
: Controller /a
riac/gantry/gan
try_controller
successfully fi
nished
[ INFO] [160735
7766.863997605,
 225.962000000]
: Controller /a
riac/gantry/rig
ht_arm_controll
er successfully
 finished
[ INFO] [160735
7767.573037206,
 226.062000000]
: Controller /a
riac/gantry/lef
t_arm_controlle
r successfully
finished
[ INFO] [160735
7767.573195718,
 226.062000000]
: Completed tra
jectory executi
```

```
Possible reasons are listed at http://wiki.ros.org/tf/E
rrors%20explained
[ INFO] [1607357386.199591207, 151.352000000]: ABORTED:
 Solution found but controller failed during execution
[ INFO] [1607357386.200604923, 151.352000000]: [GantryC
ontrol][deactivateGripper] DEBUG: srv.response =success
: 1

[DEBUG] [1607357389.062397469, 152.050000000]: Setting
in moving list part  pulley_part_red
[ INFO] [1607357389.062450485, 152.050000000]: Received
 order:
order_id: order_1
shipments[]
  shipments[0]:
    shipment_type: order_1_shipment_0
    agv_id: agv2
    products[]
      products[0]:
          type: pulley_part_red
        pose:
          position:
            x: 0
            y: -0.15
            z: 0
          orientation:
            x: 1
            y: 0
            z: -0
            w: -1.03412e-13
      products[1]:
          type: disk_part_green
        pose:
          position:
            x: 0
            y: 0.15
            z: 0
          orientation:
            x: 0
            y: 0
            z: 0
            w: 1

[DEBUG] [1607357389.062503343, 152.050000000]: Setting
in static list part  disk_part_green
[ INFO] [1607357389.062537978, 152.050000000]: I heard:
 order_1
```

Implementation challenges

Challenges:

- Picking from conveyor
- Picking from AGV tray
- Placing parts with the correct orientation on the AGVs
- Picking parts with small thickness
- Dynamically assigning waypoints for picking parts
- Sending AGV for delivery before time limit is exceeded

Validation and testing

- Rigorous testing was done for all the 15 test cases provided
- Issues were highlighted and appropriate actions were taken.
- Custom test cases were developed and tested

Results

- Successfully completed the ARIAC 2020 kit building competition
- Package developed using good software engineering practices with the agile process.
- Successfully handled all agility challenges - faulty part, flipped part, high priority order, faulty gripper, sensor blackout and moving obstacles
- Able to successfully execute all the 15 test cases
- Github repo - https://github.com/varunasthana92/ROS_ARIAC

Scope for future development

- Orientation correction when placing with the right arm end effector
- Implement sensor blackout behavior instead of waiting
- When an HPO is to be built on an AGV which is already in use, then instead of clearing the AGV we may re-utilize some of the existing parts
- Update part pose periodically in the data structure for the static parts in the environment
- Use of both arms to complete the order faster

Contribution

1. **Varun Asthana:**
    a. Order read and store as linked list, High priority order, Moving obstacles, Sensor Blackout agility challenges
    b. Check for time limit exceed and ship partially build orders
    c. Part orientation correction on AGV tray
    d. Next part allotment based on various checks of moving obstacle, high priority order, availability on conveyor and agv availability
    e. Overall code architecture
    f. Debugging and rectification
2. **Saumil Shah:**
    a. Order read and store as linked list,, High priority order, Moving obstacles, Sensor Blackout agility challenges
    b. Part orientation correction on AGV tray
    c. Conveyor part tracking and trigger signal to pick the part from the right location. Retry functionality if part missed to pick by allocating new part of same type from the tracked parts on the conveyor.
    d. Overall code architecture
    e. Debugging and rectification
3. **Markose Jacob:**
    a. Dynamic waypoints, finding gaps
    b. Faulty part, faulty gripper agility challenges
    c. Code testing
4. **Nalin Das**
    a. Flip part, faulty part, faulty gripper agility challenges
    b. Presentation
    c. Doxygen commenting and documentation
5. **Aditya Goswami**
    a. Flip part, faulty part, faulty gripper agility challenge
    b. Report

References

1. https://github.com/usnistgov/ARIAC/tree/master/wiki/documentation
2. Slides and class lectures
3. MoveIt Tutorials — moveit_tutorials Noetic documentation
4. ROS/Tutorials - ROS Wiki