ENPM-663 Building a Manufacturing Robotic Software System

Group 3:

Robert Nicholas Vandemark (UID: 117547092)

Sameep Vijay Pote (UID: 118218427)

Abhilash Pravin Mane (UID: 117402865)

Rajan Prabodh Pande (UID: 117306002)

Atharva Chandrashekhar Paralikar (UID: 117396560)

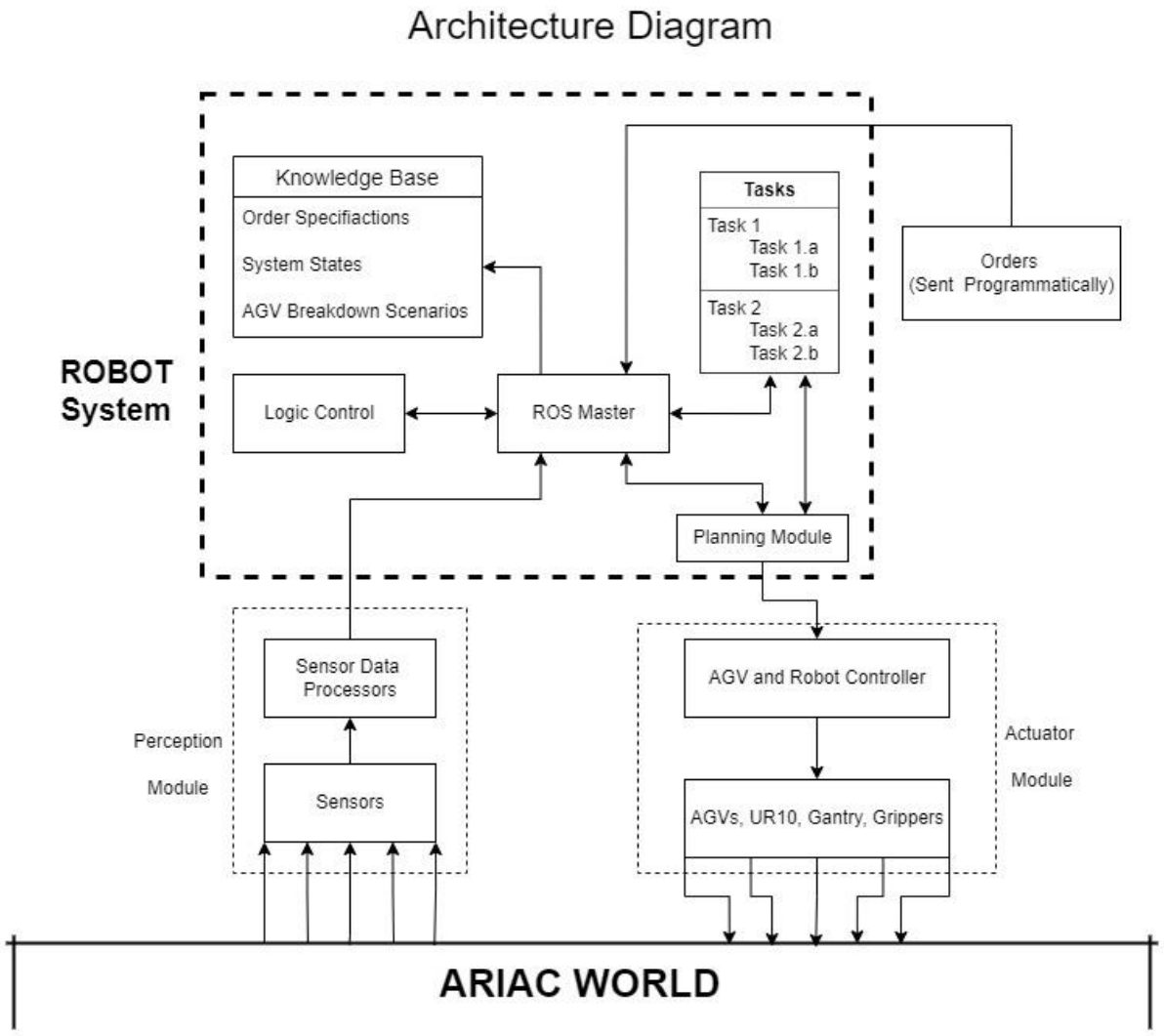
Date: 27th February 2022

Report for RWA_1

Table of Contents

Architecture:

The Architecture for the system can be divided into three major Components, the Robot System, Perception Module, and the Actuator Module. The system is given specific orders as input. We discuss the responsibilities of each of these Subsystems.



Robot System:

Consists of following components:

1) Knowledge Base: Used to store ruleset for the competition tasks, Systems states and dynamically store sensor/robot breakdown scenarios. This can be accessed by the Logic and Planning Module through the ROS Master.
2) Logic Control Module: Responsible for all the decision making based on order priority and kitting scenarios

3) Planning Module: Responsible for planning of tasks to ensure efficient use of AGVs, Robotic Arms and Gantry Robot
4) Task Manager: Responsible to keep track of order priority and dynamically update task list.
5) ROS Master: Responsible for seamless communication between modules.

Perception Module:

Consists of following Components:

1) Sensors: Quality Camera, Logic Camera, Proximity and Break beam Sensor modules to gather various information about the position, quality and orientation of the kitting and assembly equipment
2) Data Processors: Responsible for sensor data processing.

Actuator Module:

Consists of following Components:

1) Robots: Gantry Robot, UR10 Robot used for Kitting, Handling and Assembly for Components
2) AGVs: Used to transport components to the substations for assembly of the Components.
3) Robot and AGV Controllers: Responsible for the controlling of all the Robot and AGVs according to the commands of the Planning and Logic Control Modules.

Sequence Diagram:

We have two distinct operations namely, Kitting and Assembly. Hence, we have two sequence diagrams for the same.

1) Kitting Procedure:
   This operation requires us to ensure that correct components are sent to the correct substation for assembly according to the order. It is also necessary to ensure that only the good quality components are kitted, and defective ones are discarded in the bin.

2) Assembly Procedure:
   This operation requires us to assemble the kitted components at the specific substation. This is to be done keeping in mind the right orientation and location of each of the component. This requires the movements of the assembly robot be precise since any collision with the environment leads to a penalty.

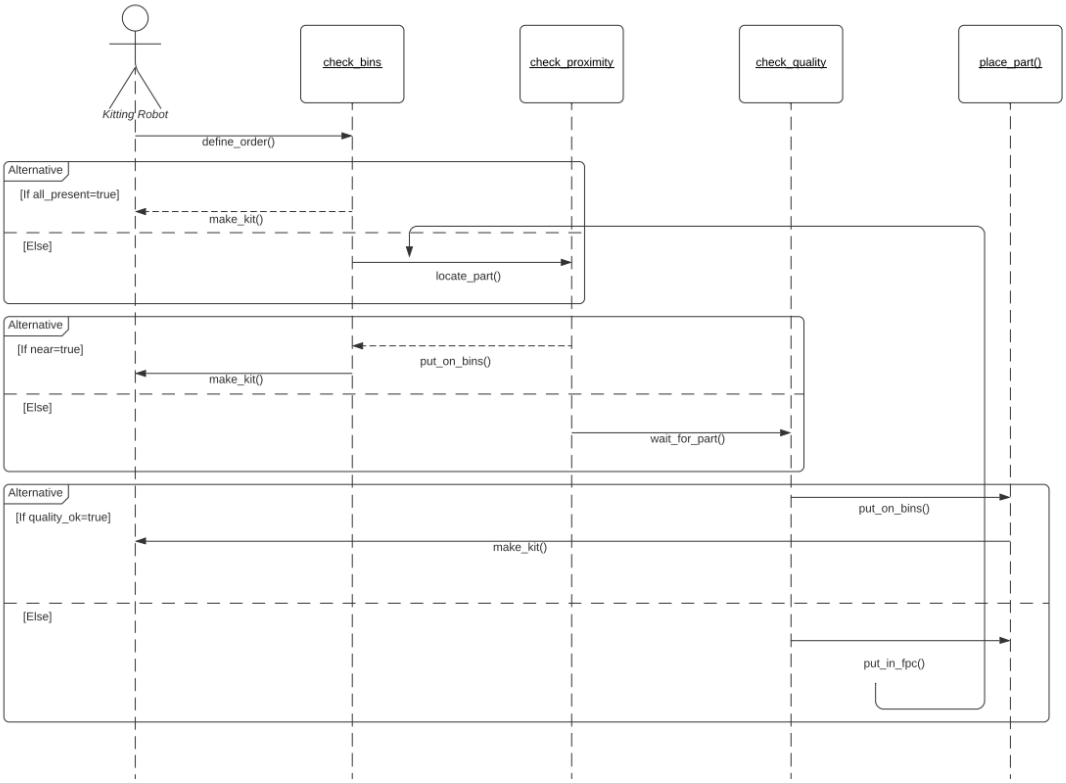


Figure 1:Sequence Diagram for Kitting Operation

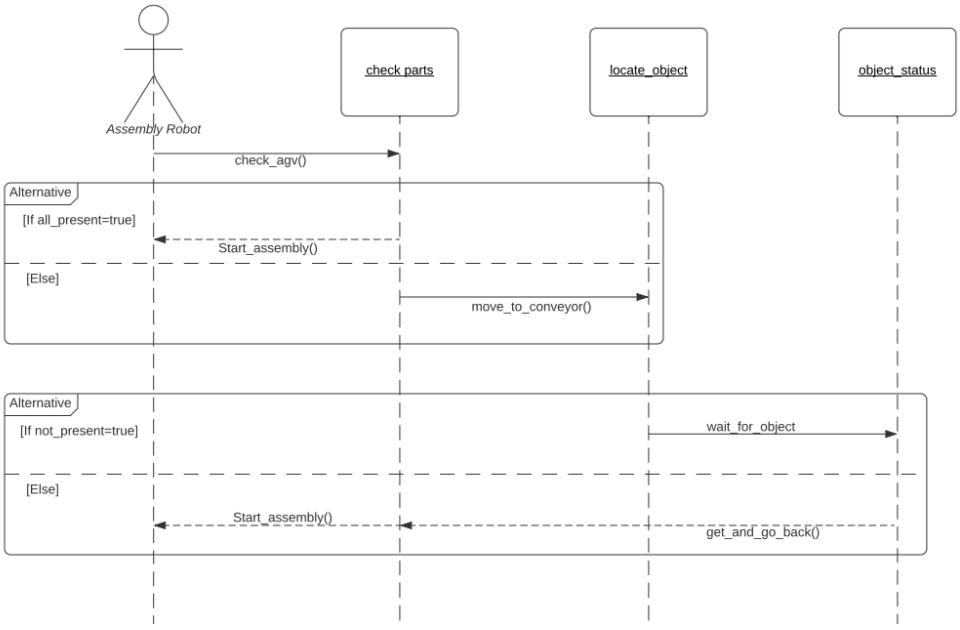


Figure 2: Sequence Diagram for Assembly Procedure

UML and Class Diagram:

Our code has two classes: *AriacAgv* and *Sensor*. The UML diagram has two blocks for each class, with following skeleton.

Class name
declared_variable1_in_class: datatype + declared_variable2_in_class
+ function_in_class (param1, param2): return Type - func_in_class (param1, param2)

Here,

# denotes all declarations that are protected
+ denotes all declarations that are public
- denotes all declarations that are private

AriacAgv has functions related to AGV and *Sensor* has functions related sensor reading

In *AriacAgv*;

- *number* is an integer that represents the AGV number which ranges from 1 to 4
- *state_sub* is object of *ros::Subscriber* having datatype of string message; it receives the updated state of the AGV
- *station_sub* is object of *ros::Subscriber* having datatype of string message; it gives the updated station message
- *submit_shipment_scl* creates an object of the service *AGVToAssemblyStation* and responds in boolean of success along with the message of datatype string
- *curr_state* stores the value received from subscriber *state_sub*
- *curr_station* stores the value received from subscriber *station_sub*
- *state_callback* is a callback function of *state_sub*
- *station_callback* is a callback function of *station_sub*
- *submit_shipment* is a boolean function that return the status of shipment submission and depends on the response of *submit_shipment_scl*

In *Sensor*;

- *breakbeam0_sub* is a *ros::Subscriber* object which has a datatype of *nist_gear::Proximity* and it subscribes to the *break beam sensor* topic
- *breakbeam0_change_sub* is similar to *breakbeam0_sub* but it is triggered only when the state changes
- *logical_camera_bins0_sub* is a *ros::Subscriber* object which has a datatype of *nist_gear::LogicalCameraImage* which subscribes to the logical camera sensor topic mounted on the bin
- *logical_camera_station2_sub* is a *ros::Subscriber* object which has a datatype of *nist_gear::LogicalCameraImage* which subscribes to the logical camera sensor topic mounted on the station
- *proximity_sensor_0_sub* is a *ros::Subscriber* object which has a datatype of *sensor_msgs::Range* which subscribes to the proximity sensor topic

· *laser_profiler_0_sub* is a *ros::Subscriber* object which has a datatype of *sensor_msgs::LaserScan* which subscribes to the laser profiler sensor topic
· *quality_control_sensor_[n]_sub* is a *ros::Subscriber* object which has a datatype of *nist_gear:: LogicalCameraImage* which subscribes to the quality control sensor topic (here n ranges from 1 to 4)
· *n* is a *ros::NodeHandle*
· *break_beam_callback* is a callback function which triggers on receiving data from *breakbeam0_sub*
· *break_beam_change_callback* is a callback function which triggers on receiving data by *breakbeam0_change_sub*
· *laser_profiler_callback* is a callback function which triggers on receiving data by *laser_profiler_0_sub*
· *proximity_sensor_callback* is a callback function which triggers on receiving data by *proximity_sensor_0_sub*
· *quality_callback[n]* is a callback function which triggers on receiving data by *quality_control_sensor_[n]_sub* (here n ranges from 1 to 4)
· *logical_camera_callback* is a callback function which triggers on receiving data by *logical_camera_bins0_sub*
· *logical_camera_callback2* is a callback function which triggers on receiving data by *logical_camera_station2_sub*
· *startdetect* is a function in which all subscribers continuously called

In *AGVToAssemblyStation* service:

· *Request* is sent to the service with *assembly_station_name* (of datatype string) and *shipment_type* (of datatype string)
· *Response* is received from the service with *success* (of datatype bool) and *message* (of datatype string)

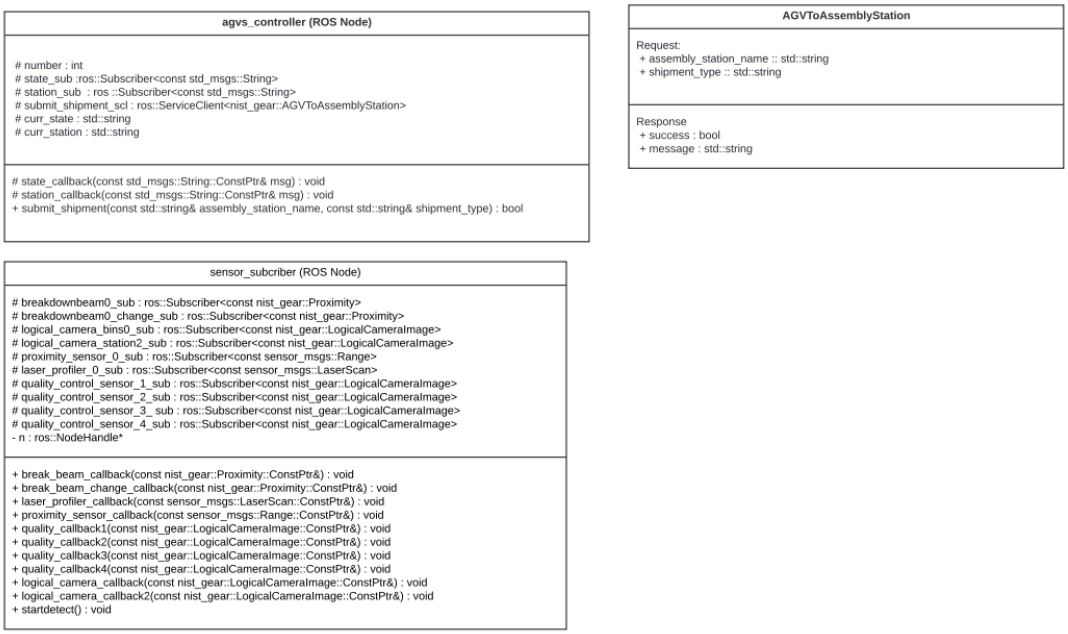


Figure 3: UML Class Diagram