

**DEVELOPMENT OF TASK DISTRIBUTION
ALGORITHM FOR MULTI-ROBOT
COORDINATION SYSTEM**

SIN HAN BIN

UNIVERSITI TUNKU ABDUL RAHMAN

**DEVELOPMENT OF TASK DISTRIBUTION ALGORITHM FOR
MULTI-ROBOT COORDINATION SYSTEM**

SIN HAN BIN

**A project report submitted in partial fulfilment of the
requirements for the award of Bachelor of Engineering
(Honours) Mechatronics Engineering**

**Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman**

April 2022

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : SIN _____

Name : SIN HAN BIN _____


ID No. : 17UEB04516 _____

Date : 1/4/2022 _____

APPROVAL FOR SUBMISSION


I certify that this project report entitled **“DEVELOPMENT OF TASK DISTRIBUTION ALGORITHM FOR MULTI-ROBOT COORDINATION SYSTEM”** was prepared by **SIN HAN BIN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Mechatronics Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : 

Supervisor : Ir. Dr. Ng Wee Kiat

Date : 24 Apr 2022

Signature : 

Co-Supervisor : Dr. Kwan Ban Hoe

Date : 24 Apr 2022

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2022, SIN HAN BIN. All right reserved.

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project I would like to express my gratitude to my research supervisor, Ir. Dr. Danny Ng Wee Kiat, co-supervisor, Dr. Kwan Ban Hoe for their invaluable advice, guidance, and enormous patient throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parents and friends who had helped and give me encouragement as well as supported me along the process of implementing project research. Without having these supports, I would not be able to achieve great success in my final year project.

ABSTRACT

Multi-Robot system consists of a group of autonomous robots that work together to accomplish the given tasks to achieve a specific goal. In the era of advanced technology, a single robot is replaced by multiple robots with different capabilities in the industry due to their fault-tolerance and efficiency in terms of cost and time for task execution. Delivery is the most common task implemented by a Multi-Robot system in a manufacturing industry. However, task allocation to a group of heterogeneous robots is challenging as allocation of the same task to multiple robots is commonly occurred. The problem always arises when robots are navigating such as the collision between robots and obstacles. Hence, an effective task distribution algorithm is important to allocate tasks correctly among the robots with optimum cost and time utilization. Besides, proper navigation techniques allow robots to avoid collisions and obstacles during task execution. Software such as Robot Operating System 2 (ROS2), Gazebo, ROS Visualization 2 (RViz2), and Robotics Middleware Framework (RMF) are utilized for the development and simulation of the project. During the development process, the simulation environment is modeled with desired task points, robots, and traffic lanes. Algorithms are integrated with RMF for task distribution and path planning for the robots. Whereas robots' motion at the designated path is visualized through RViz2. The project simulation is carried out in Gazebo to validate the performance of the algorithms. RMF Panel provides an online platform to perform task submission and enables real-time visualization of task distribution of robots as well as robots' status. Effective task distribution and self-navigation of robots are simulated in Gazebo and RViz2 respectively with the integration of the RMF framework and ROS2. The delivery tasks are distributed efficiently among the respective robots which have obstacle avoidance and traffic conflict resolved capabilities in the simulation.

TABLE OF CONTENTS

DECLARATION	i
APPROVAL FOR SUBMISSION	ii
ACKNOWLEDGEMENTS	iv
ABSTRACT	v
TABLE OF CONTENTS	vi
LIST OF TABLES	ix
LIST OF FIGURES	x
LIST OF SYMBOLS / ABBREVIATIONS	xii
LIST OF APPENDICES	xiii
CHAPTER 1	1
INTRODUCTION	1
1.1 General Introduction	1
1.2 Importance of Study	3
1.3 Problem Statement	3
1.4 Aim and Objectives	3
1.5 Scope and Limitation of the Study	4
1.6 Contribution of the study	4
1.7 Outline of the Report	5
CHAPTER 2	6
LITERATURE REVIEW	6
2.1 Introduction	6
2.2 Task Distribution in Centralized and Decentralized Multi-Robot System	8

2.3	Explicit and Implicit Communication in Multi-Robot System in Task Distribution	10
2.4	Process of Task Distribution in Multi-Robot System	11
2.4.1	Task Decomposition	11
2.4.2	Task Allocation	12
2.4.3	Motion Planning and Control of Multi-Robot System	13
2.4.4	Task Execution	13
2.5	Applicable Fields for Task Distribution of Multi-Robot System	14
2.5.1	Multi-Robot Task Distribution for Exploration and Destruction	14
2.5.2	Multi-Robot Task Distribution for Search and Rescue	16
2.5.3	Multi-Robot Task Distribution for Hospital and Warehouse Logistic	17
2.5.4	Multi-Robot Task Distribution for Manufacturing Industry	20
2.5.5	Comparison Of Task Distribution Approaches In Multi-Robot System	21
2.6	Simulation Software	22
2.6.1	Robot Operating System 2 (ROS2)	22
2.6.2	Navigation 2 (NAV2)	23
2.6.3	Simultaneous Localization And Mapping (SLAM)	24
2.6.4	Gazebo	26
2.7	Summary	26
CHAPTER 3		28
METHODOLOGY AND WORK PLAN		28
3.1	Introduction	28
3.2	Modeling of Simulation Environment	29
3.3	Task Distribution Algorithm for Multi-Robot System	30

3.4	Motion Planning of Robots	32
3.5	Evaluation and Testing of Algorithm	33
3.6	Summary	34
CHAPTER 4		36
RESULTS AND DISCUSSION		36
4.1	Introduction	36
4.2	Modeling of Simulation Environment	37
4.3	Task Distribution of Heterogeneous Robots	38
4.4	Path Planning of Heterogeneous Robots	42
4.5	Simulation and Evaluation of Task Distribution of Heterogeneous Robots	45
4.6	Summary	46
CHAPTER 5		48
CONCLUSIONS AND RECOMMENDATIONS		48
5.1	Conclusions	48
5.2	Recommendations for future work	49
REFERENCES		50
APPENDICES		53

LIST OF TABLES

Table 2.1:	Comparison of Task Distribution Approaches	21
------------	--------------------------------------------	----

LIST OF FIGURES

Figure 2.1:	Centralized Coordination (Ravankar et al., 2018)	8
Figure 2.2:	Decentralized Coordination (Ravankar et al., 2018)	9
Figure 2.3:	Simulation results of 10 robots with 40 suspicious regions (Dai et al.,2020)	16
Figure 2.4:	Task distribution process of MRS (Jeon, Lee and Kim, 2017)	18
Figure 2.5:	Simulated warehousing environment (Fan, Deng, and Shi, 2020)	19
Figure 2.6:	Message transmission between nodes and topics (Maruyama, Kato and Azumi, 2016)	23
Figure 2.7:	Overview of NAV2 architecture (Macenski <i>et al.</i> , 2020)	24
Figure 2.8:	State of art of SLAM (Khairuddin, Talib and Haron, 2016)	25
Figure 3.1:	Overview of the simulated manufacturing environment	30
Figure 3.2:	Communication flow during task distribution	31
Figure 3.3:	Communication architecture between traffic schedule database, fleet adapter and robots	33
Figure 3.4:	Process flow of task distribution in Multi-Robot system	35
Figure 4.1:	2D layout of simulation environment	37
Figure 4.2:	Traffic lanes of robots for delivery task distribution	39
Figure 4.3:	RMF Panel for task submission	39

Figure 4.4:	RMF Panel for monitoring of task execution progress and robot status	40
Figure 4.5:	Task distribution during robots' battery threshold	41
Figure 4.6:	Visualization of traffic conflict among robots	43
Figure 4.7:	Traffic conflict resolved among robots	43
Figure 4.8:	Collision and obstacle avoidance by tiny robot in Gazebo	44
Figure 4.9:	Simulation of task distribution of heterogeneous robots in Gazebo	46

LIST OF SYMBOLS / ABBREVIATIONS

BT	Behavior Tree
D	Destination
DR1C	Delivery Robot 1 Charger
DR2C	Delivery Robot 2 Charger
MRS	Multi-Robot system
NAV2	Navigation 2
ROS	Robot Operating System
ROS2	Robot Operating System 2
RMF	Robotics Middleware Framework
RViz2	Robot Operating System Visualizer 2
SLAM	Simultaneous Localization And Mapping
STVL	Spatio-Temporal Voxel Layer
TP	Task Point
TR1C	Tiny Robot 1 Charger
TR2C	Tiny Robot 2 Charger

LIST OF APPENDICES

APPENDIX A:	Multiple task .json file sample	53
APPENDIX B:	Launch files for simulation	54
APPENDIX C:	Part 1 Gantt Chart	61
APPENDIX D:	Part 2 Gantt Chart	62

CHAPTER 1

INTRODUCTION

1.1 General Introduction

In the era of advanced technology, a wide variety of robots with different capabilities are utilized in many sectors especially in the industry to achieve specific goals effectively and efficiently. During the past few decades, the Multi-Robot system (MRS) has been introduced in the industry to take over Single-Robot Systems mainly due to its fault-tolerance and robustness (Dai et al., 2020). MRS consists of a group of autonomous robots working together to accomplish a series of tasks to meet an industry's goal. On the other hand, task distribution in MRS has gained significant attention from researchers since the 1990s. Multi-Robot task distribution is a process whereby a common task is separated into several subtasks and then distributed to each robot. A task distribution algorithm in the MRS will define the best allocation of tasks to each of the robots which can complete the given task within an optimum time and cost (Panchu K., Rajmohan, Sundar, and Baskaran, 2018). For instance, a task with the combination of cutting and transporting is decomposed into subtasks such as cutting subtask and transporting subtask. After that, these subtasks are distributed and done by each of the robots with their respective abilities. The purpose of task distribution across MRS is to maximize the profit after task execution with minimum cost used (Li et al., 2017).

Generally, task distribution in MRS can be categorized into 3 main stages which are planning of task, distribution of task, and followed by planning of robotic movement during task execution (Olive & Guerrero, 2011). First and foremost, a sophisticated task is divided into several subtasks with a simpler goal that can be finished by each of the robots which have different capabilities. During the task distribution stage, the task distribution algorithm will assign each of the subtasks to the respective robot with the best performance such as shorter execution time and minimum cost usage. Finally, the shortest path will be planned by each of the robots

from their initial position to the subtask position without collision between the robots during the task execution.

As moving forward to Industry 4.0, many computer networking and automation technologies are progressively integrated into the manufacturing system of the industry to implement smart manufacturing. Nowadays, advanced production systems have continuously dominated the manufacturing industry to take over manual production because of its advantages in manufacturing strategies without the intervention of humans.

Task distribution in the Multi-Robot system is extremely crucial in the manufacturing industry because it plays an important role in determining the efficacy and competitiveness of an industry. With the emergence of high demand for small batch and customized products, the manufacturing industry has to adopt different strategies to satisfy the inconsistent requirements. However, the nature of the current manufacturing industry mostly relies on mass manufacturing with inadequate flexibility. Therefore, flexible production of customized products with cost-effective avenues can be achieved by applying MRS. This can be achieved by utilizing different robots with the respective capability to manage complex tasks.

After the emergence of technology in MRS, numerous researchers have proposed diverse task distribution algorithms for MRS such as the Auction algorithm, Vacancy Chain algorithm, Hungarian algorithm, etc (Dai et al., 2020). The primal intention of these algorithms is to obtain optimization in time usage, cost, and load balance. The shorter the time used and the lower the execution cost, the better the performance of robots. Load balance indicates the number of tasks allocated to a single robot. It may happen whereby a robot has to perform many tasks but there is also a robot with no task given. Thus, the total task execution time is increased, affecting the overall efficiency of the system. Undeniably, it is hard to attain optimum time, cost, and load balance simultaneously in the task distribution of a Multi-Robot System. Currently, ongoing research is being implemented to tackle the problems encountered in task distribution in MRS.

1.2 Importance of Study

The significance of the study of this project is to develop a suitable algorithm to control a fixed number of autonomous robots to carry out specific tasks. The tasks with different requirements will be distributed to the robots with respective capabilities for execution. The algorithm should be able to distribute the tasks according to the robots which are capable of accomplishing the tasks within the shortest period with minimum costs. By achieving these goals, the algorithm is useful for the manufacturing industry whereby production time and expenses can be conserved simultaneously. By utilizing the developed algorithm, autonomous robots in manufacturing industries can execute the tasks given effectively and efficiently without the intervention of humans. Depending on the algorithm, the robots will be able to navigate themselves in a manufacturing environment and prevent collisions between robots during tasks execution. In short, the importance of this study is to develop an algorithm to perform task distribution to multiple robots with heterogeneous capabilities in a mimic manufacturing environment.

1.3 Problem Statement

Undeniably, problems arise during the assignment of tasks to a group of heterogeneous robots to maximize productivity at the same time preserving the quality and expenses of a manufacturing process. Usually, collisions between robots during task execution as well as the crashing of robots that are being selected for the same task are the major problems in the task distribution of MRS. Obstacles avoidance will be another issue for multiple robots while performing self-navigation in an environment.

1.4 Aim and Objectives

This project aims to develop a task distribution algorithm for the Multi-Robot system which can assign various types of tasks to the robots with respective capabilities. The purpose of this project is to mimic the operation of the Multi-Robot System in a manufacturing environment via simulation software. To achieve this goal, several objectives are stated below :

- Able to perform efficient task distribution for a heterogeneous Multi-Robot system.
- Enable multiple robots to perform self-navigation in a known manufacturing environment through simulation.
- Enable multi robots to avoid obstacles and collisions during navigation in the environment through simulation.

1.5 Scope and Limitation of the Study

The scope of this project is to simulate the task distribution process in a Multi-Robot system in an environment based on the manufacturing industry. With the use of specific software like ROS2 and SLAM, the robots can self navigate and avoid collisions between robots while executing the task. The tasks distributed will have specific requirements whereby only the robots with corresponding abilities will be able to execute the task. Therefore, the developed algorithm will be able to optimize the time, cost, and load balance during the task distribution process of multiple robots. However, the manufacturing environment is different for all industries. Hence, the simulated manufacturing environment may not be best suited to some of the manufacturing industries. The simulation model for this study is limited hence the simulation results may be different when the simulation is done on other models.

1.6 Contribution of the study

With the completion of this project, one of the contributions is providing an efficient task distribution solution for a Multi-Robot system with the integration of RMF in ROS2 which can be adopted in a real manufacturing environment. Moreover, the RMF traffic editor is introduced as one of the useful modeling tools to build a 3D simulation model in Gazebo which eases the modeling process. A simulation model of a manufacturing environment is created to be used for evaluation of the task distribution process which provides an added advantage for the collection of results on delivery task distribution among a heterogeneous Multi-Robot system. RMF traffic editor also provides another technique that eases the

task distribution among heterogeneous robots which is suitable to be utilized in real-world scenarios.

1.7 Outline of the Report

This report consists of five main chapters which are the introduction, literature review, methodology, result and discussion as well as conclusion. In the first chapter which is the introduction, the problem encountered in task distribution of a Multi-Robot system and general concepts of task distribution in heterogeneous robots are discussed. Whereas literature review summarizes the research and example regarding the task distribution of a Multi-Robot system, including the coordination, communication, approaches, and process of task distribution with a discussion on some of the real-world examples. The simulation software such as ROS2, SLAM, NAV2, and Gazebo are also talked through which are useful to run a simulation that can mimic the real-world examples.

On the other hand, the methodology will provide the details steps to construct the project. The detailed plan and design of the simulation are included such as the components and layout to model a simulation environment and the task distribution approach to be implemented. The result and discussion chapter will include all the simulation results of the project with a detailed discussion on each element of the results. The last chapter is the conclusion that provides a summary of the whole project and some recommendations on the project for future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Task distribution in the Multi-Robot system (MRS) has become one of the hot topics nowadays. Researchers all over the world have participated in the exploration of this technology so that this technology can be beneficial to a wide range of sectors including industry, health care, security, and surveillance (Khamis, Hussein, and Elmogy, 2015). Task distribution is the apportioning of tasks to a group of autonomous robots according to a set of specific parameters such as cost and time. Robots that have the best optimization in the parameters will be firstly assigned with a task. This process is repeated until all the tasks have been allocated to the robots. When the robots have accomplished the tasks then the main goal of task distribution in MRS is achieved.

Initially, task distribution is implemented in a single robot system in which a robot is loaded with a variety of capabilities and able to carry out a complex task individually. However, the single robot system is slowly replaced by a Multi-Robot system mainly due to the lack of robustness. In contrast, MRS consists of several robots with respective capabilities working together to complete a sophisticated task. With the attractive features of MRS, a complex task is simplified upon execution. As a result, MRS has a better ability to tackle a task with higher complexity that may be impossible to be solved by a single robot. One of the major drawbacks of a single robot system is task execution time as some of the tasks that should be done simultaneously are performed sequentially by a single robot. Task execution time in MRS is greatly reduced because multitasks can be executed at the same time by MRS. Thus, MRS has higher performance than a single robot system. The reliability of a single robot is lower as compared to MRS whereby the whole process is affected when the robot is malfunctioning. In the case of MRS, when a robot is broken down, the process can be continued as other robots are still in good functionality (Khamis, Hussein, and Elmogy, 2015). In fact, robots in MRS have

simpler designs that are easier to be maintained and set up. For long-term investment, MRS is typically cheaper than a single robot with multiple capabilities.

Basically, there are many types of task distribution in a Multi-Robot system. For instance, homogeneous task distribution, heterogeneous task distribution, dynamic task distribution as well as constricted task distribution. A bunch of robots that have the same ability to work on the tasks given is classified as homogeneous task distribution. Contradictorily, heterogeneous task distribution requires cooperation between multiple robots with a combination of different capabilities to finish a given task. On the other hand, dynamic task distribution is defined as the sudden addition of extra tasks to the MRS or task execution of robots is implemented in an unknown environment. The environment may have undefined obstacles and the robots have constrained information regarding the tasks or environment. The robots will need to have the intelligence in handling uncertainty independently. Whereas for constricted task distribution, tasks with stated deadlines or limited task execution time may be allocated to the robots (Luo, Chakraborty, and Sycara, 2015). Similarly, the robots may encounter limited shared resources during task execution.

Undoubtedly, the creation of a robust task distribution algorithm in a Multi-Robot system is not a one or two day's work. Much effort and hard work are paid by researchers to achieve success in the algorithm. Until today, many challenging barriers are faced by current researchers such as optimization in the distribution of tasks among the robots. This is subjected to certain constrictions to obtain optimal performance of the system. MRS will definitely encounter problems like a breakdown of certain components, malfunction of sensors, unpredictable changes in the environment, and also delay or limitation of communication between the robots in the system (Li, Fan, and Dai, 2017). These uncertainties will cause major failures of the algorithms, affecting the overall performance of the system. Therefore, these problems are gaining attention from researchers to find an alternative avenue to tackle these problems.

2.2 Task Distribution in Centralized and Decentralized Multi-Robot System

The coordination between the robots in the Multi-Robot system is important to ensure the successfulness of task execution. The distribution of a common task to more than one robot is prevented. With superior coordination among the robots in the system, the process of task execution can be run swiftly and efficiently. So, the main goal of the task distribution algorithm can be attained effectively. There are two common types of coordination in MRS which are centralized and decentralized.

In centralized coordination, there is one agent which acts as the main core of the robot system (Doriya, Mishra, and Gupta, 2015). The agent is playing a significant role in receiving the information from all the robots and then carries out data processing, followed by sending out execution signals back to the robots. As an example, when a task appears in the system, all the robots will report their respective distance to the position of the task to the agent. After that, the agent will undergo data processing and then transmit an execution command to the robot which has the nearest distance to the task given. As a result, only the particular robot will be able to get the task. In short, there is only communication between the agent and robots and vice versa but no communication among the robots. The agent will have the full authority to assign the tasks among the robots.

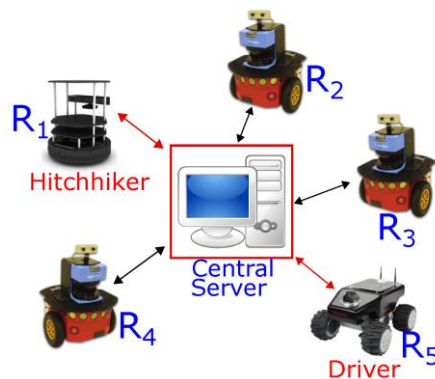


Figure 2.1: Centralized Coordination (Ravankar et al., 2018)

The advantage of centralized coordination is time and cost-saving as data processing is carried out once in the task distribution process. Simpler robots can be designed because communication is only required between the agent and robots. The main bottleneck of this approach is that all the execution of robots is dependent on the agent and failure may result in the whole system when the agent fails to work (Khamis, Hussein, and Elmogy, 2015). Besides, the system scale is restricted due to all the robots are connected to a single agent. It is troublesome when a huge number of robots are being connected. The whole system may collapse when there is too much connection. Hence, the centralized method may be the best choice for task distribution in MRS when the scale of the robot is small and the environment of the system is constant and known.

In decentralized coordination, there is no agent in the system and the robots are independent of each other. Each robot is having full control over making decisions to execute the task itself. Each robot will receive the information of tasks given and its information is exchanged through communication between the other robots in the system (Khamis, Hussein, and Elmogy, 2015). A decision is made to execute the task when the robot is having the best result among all the robots. For example, when a task is given in the system, all the robots will exchange their information on the distance toward the task position with each other. While the robot has the best result upon the exchange of information, which is the shortest distance toward the task position, the task will be allocated to that robot. Thus, instead of having an agent, communication among the robot is applied to decide the distribution of tasks.

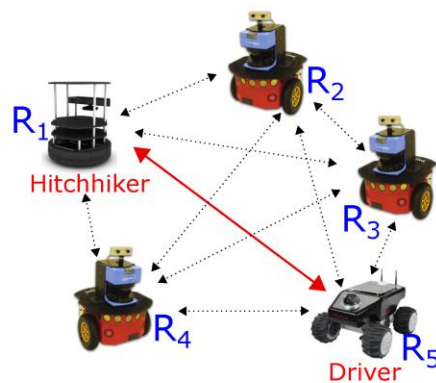


Figure 2.2: Decentralized Coordination (Ravankar et al., 2018)

The benefits of decentralized coordination are flexibility and robustness. Each of the robots is working independently with the capability of storing the local information about the task and also sharing its information among the robots. The system will not suffer the drawbacks of centralized coordination and scalability problems as no agent is implemented in the system. The other robots will continue running when one robot malfunctions, leading to higher fault tolerance of the system. Furthermore, other robots can be added to the environment without affecting the system. However, decentralized coordination may generate a semi optimum solution in task distribution in MRS.

2.3 Explicit and Implicit Communication in Multi-Robot System in Task Distribution

Communication among the robots or between the agent and the robots in a Multi-Robot system is crucial. Therefore, the robot or the agent has the latest update on the robots' status and information regarding the environment or tasks given. With the implementation of effective communication, the robots will be able to finish the tasks given efficiently without any issue. The communication approaches can be categorized into two types which are explicit communication and implicit communication.

Firstly, explicit communication indicates the direct exchange of data between the robots such as the message is shared or delivered verbally by the robots in the system (Doriya, Mishra, and Gupta, 2015). As an example, a robot broadcasts its distance towards the task position to other robots. Then this process is repeated and a comparison is made between the robots to decide which robot will get the respective task. Clear information is exchanged through explicit communication whereby interpretation or data processing of the information is not required. In MRS, explicit communication is commonly implemented through the internet, voice recognition, or Bluetooth.

Moreover, implicit communication indicates the indirect exchange of information between the robots in the system. For instance, a robot equipped with various sensors can analyze the environment and read the distance between the

other robots and the task position. By getting the distance information of other robots, decision-making can be done through comparison among the robots. This communication can be done independently without any interaction from other robots in the communication. The robot can convey the data by observing the other robots' status in the environment to get the required information, therefore data analysis and processing are needed (Gildert *et al.*, 2018). Generally, implicit communication involves the gathering and interpreting of surrounding information to make a judgment rather than exchanging messages directly through verbal or codified output.

2.4 Process of Task Distribution in Multi-Robot System

Task distribution comprises a series of workflows in which the autonomous robots equipped with specific capabilities are assigned with a mission or task to be completed to fulfill a general goal. This process is implemented without the intervention of humans to increase the accuracy of the system by eliminating possible human errors. Task distribution in the Multi-Robot system is important to assure the smooth operation of all the machines and robots. Thus, cost and time can be optimized as well as retaining the productivity and quality of the final product simultaneously. Typically, task distribution in MRS can be further divided into several processes which are task decomposition, task allocation, motion planning and control of MRS, and task execution (Rizk, Awad and Tunstel, 2019).

2.4.1 Task Decomposition

The main purpose of task decomposition is to break down single or multiple complicated missions into some simpler subtasks to be accomplished by the robots in the system. As the complexity of the task is reduced, robots with simpler designs are greatly utilized to meet the task requirements. Characteristics of each of the robots in the system, task requirement, environmental information, and also resource distribution status shall be taken into consideration during the task decomposition process. Hence, clear and explicit information regarding the task and environment is received by the robots to make sure the best optimization of

resource usage. The efficacy and effectiveness of the whole robot system can be improved through collaboration between the robots (Chen, Yang, and Wei, 2010). As a result, a proactive task decomposition technique can lead to not only a reduction of task complexity but also an increase in the executability of subtask by robots that are armed with primitive and simplex abilities.

2.4.2 Task Allocation

After task decomposition, the MRS will undergo task allocation to distribute the subtasks to each of the robots in the system by analyzing the current status of the robots and their respective capabilities. During this process, the task allocation algorithm is crucial to determine which robot will have the best performance for the task execution via mathematical equations to calculate the suitability of specific robots. By considering several parameters such as time and cost, the algorithm will be able to define an optimum solution to allocate the subtask to the robots which will generate the best result. Therefore, a robust algorithm is essential to maintain the best output in the task allocation process.

For instance, in an auction-based algorithm, explicit or direct communication is implemented between the robots to archive the task allocation process. The robot with the least cost among the robots will win the authority to execute the task (Dai *et al.*, 2020). This bidding process is repeated until all the tasks are allocated to the robots. As a result, more than one task may be allocated to a robot and the executor of each task is determined preliminary before any task execution.

On the other hand, another approach such as the vacancy chain algorithm utilizes implicit communication among the robots to select the most suitable robot to perform the task. In contrast, the robots that implement this algorithm will be allocated only one task for execution in a task bidding process (Dai *et al.*, 2020). The bidding process is repeated when there are robots in an idle state and the subsequent task will be assigned to a particular robot. So that the task allocation process is done initially and repeated after each completion of task execution.

2.4.3 Motion Planning and Control of Multi-Robot System

Motion planning of the robot is significant after task allocation as it will define a series of actions that the robot shall go through to finish the allocated task. With the integration of sensors and powerful algorithms in the robot system, a robot will have the ability to collect information regarding the environment and perform object detection to avoid collisions with obstacles or collisions between the robots. This is vital to enable a robot to move smoothly within an environment and carry out the given task efficiently.

To test the reliability and validity of developed algorithms, simulation can be done to mimic the real working environment of robots. Simulation allows the robots to model the environment from simulated data of sensors. Then the result of the movement of robots in the environment is gained as well as the success rate of the tasks is determined. Simultaneous Localization And Mapping (SLAM) is a useful software to enable the robots to localize and navigate themselves in an environment. At the same time, a mapping is created corresponding to the environment. Apart from that, SLAM also allows the integration of sensors on the robot through simulation which recovers almost the same as running a robot in a real environment. Thus, through simulation, a real condition of the environment can be described to foresee the possible problems that may occur during the task execution of robots (Rizk, Awad, and Tunstel, 2019). Modification and improvement can also be made to increase the accuracy and reliability of the algorithms implemented.

2.4.4 Task Execution

The last stage of task distribution in the Multi-Robot system is task execution whereby this stage will show the successfulness of the developed algorithm in the real environment. This phase will show the coherence between the constructed simulation and the real condition, therefore determining whether the best outcome is achieved in the real environment. During task execution, the robot may encounter dynamic issues such as a sudden change in the environment and sensory issues (Cheng *et al.*, 2019). As a result, a well-developed algorithm may be able to

overcome these problems and obtain success in task execution. Moreover, safety issues will bring a great impact on task execution. Hence, an assessment must be made to evaluate the performance of the robot during task execution to look for any improvement to enhance the whole task distribution process in MRS.

2.5 Applicable Fields for Task Distribution of Multi-Robot System

Nowadays, the advantages and robustness of the Multi-Robot System (MRS) are gaining attention from researchers who are involved in a variety of areas. Hence, this led to the introduction of many types of approaches to utilize MRS in their respective areas. Due to the flexibility of MRS, it has been successfully implemented in diverse fields such as environment exploration, research, and rescue, the logistic sector as well as the manufacturing industry. Undoubtedly, researchers are putting much effort and investment into the research and development of MRS so that it can be widely applied to other sectors.

2.5.1 Multi-Robot Task Distribution for Exploration and Destruction

According to Dai et al. (2020), there are three different task distribution algorithms of MRS being implemented and compared for exploration and destruction problems which include the Auction-based approach, the Vacancy Chain approach, and the Deep Q-learning approach. First and foremost, all the robots rather than having information regarding the environment, have only several doubtful regions to explore. The robots will identify several unknown targets which are needed to be destroyed during their exploration tasks. The problems become more complicated as the targets have different levels of defending abilities. So, the robots which have the higher attack abilities will be able to destroy the target otherwise the robots will suffer from damage. When all destruction and exploration tasks are completed, the goal is said to be achieved.

In the Auction-based approach, the robots will send their respective information regarding the given exploration tasks to each other through the decentralized approach. There are only exploration tasks in the starting phase of the problem. Then, the robot with the highest bid which is the shortest distance to a

given task will win the exploration task. When a robot has been assigned a task, the next bidding will be based on the position of the assigned task. This bidding process is repeated until all the exploration tasks have been successfully distributed among the robots before the task execution. The destruction task will dynamically appear when a target is found and the task will be added priorly to the task execution sequence of the robot based on the bidding process. When the robot is damaged the destruction task will be distributed to the nearest robot with a higher attack ability to be executed immediately. The auction-based approach has the lowest cost but higher task execution time as multiple tasks may be assigned to a robot although there are vacant robots with no task being assigned. Hence, it shows a bad load balance in the robots.

On the other hand, the Vacancy Chain approach only distributes a task to each of the available robots through the bidding process. The robot will get another task when the previous task is completed through implicit communication among the robots. Thus, each robot will be allocated at most one task at a time and select another task when it is idle. The assignment of the destruction task is similar to the Auction-based approach whereby it must be implemented priorly. As all of the tasks will be distributed almost equally to the robots, it achieves optimum time in accomplishing the tasks and load balance in the robots. However, the cost of this approach is higher.

For the Deep Q-Learning approach, instead of having task distribution rules, sample data is fed into the algorithm for training purposes. The neural network in the algorithm will automatically distribute the tasks to the respective robots by learning and mimicking the sample data through a reasonable parameter setting. It will have poor performance when the scale of robots and tasks increases, causing insufficient learning of the algorithm. Therefore, this approach has the worst result in all the criteria as compared to both the previous approaches. The figure below shows the simulation results of these three approaches to the exploration and destruction problems.

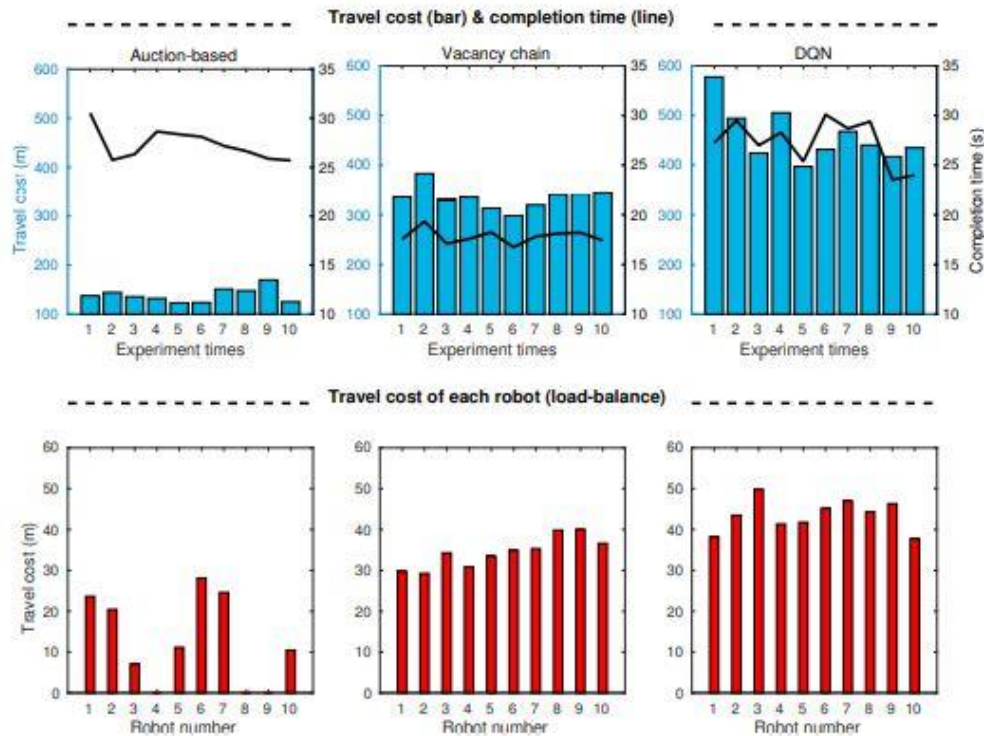


Figure 2.3: Simulation results of 10 robots with 40 suspicious regions (Dai et al.,2020)

2.5.2 Multi-Robot Task Distribution for Search and Rescue

Multi-Robot task distribution for search and rescue problems is one of the important researches. The rescuers who involve in the saving of victims from any natural disaster such as an earthquake may encounter danger or unpredictable accidents which will endanger their lives. As a result, with the implementation of MRS, robots may replace the rescuers to explore the environment, looking for the survivors as well as transporting the survivors to a safe place.

Two algorithms were proposed by Wei, Hindriks, and Jonker (2016), which are an auction-based method and a prediction method. In this problem, the robots need to first explore the environment and transport the survivor back to the initial place when a survivor is found during the exploration. The robots will need to switch between search and rescue modes to fulfill the task. As the rescue task has a higher priority so the rescue task will be immediately implemented when a survivor

is found. The main goal is achieved when all the survivors are transported to the initial place although not all the exploration tasks are completed.

In the auction-based method, all the robots will bid for the exploration tasks at the beginning. After that, they will proceed with the task execution when the robots with the shortest distance to the task point which has the lowest cost win the task. The distribution and execution of tasks are running parallelly whereby during task execution, the robot can bid for the next task. The cost to accomplish all the previous tasks and the next task will be counted when a robot is trying to bid for a new task. Hence, all the tasks can be distributed evenly among the robots and prevent the overloading of robots. While a survivor is found, a rescue task will be immediately distributed to a specific robot via the same bidding process. The rescue task will always appear in the first place in the task execution order of robots so that the common goal can be reached as fast as possible.

For the prediction method, its main difference from the auction-based method is that the task distribution and execution are done in series in which a task is being executed after the task is distributed. Thus, all the robots will have only one task to perform at a moment and the next task will be distributed when a robot has finished the previous task. The bidding process is the same as in the auction-based method and the rescue task will also be prioritized by the robots.

2.5.3 Multi-Robot Task Distribution for Hospital and Warehouse Logistic

Hospital is one of the crowded places where many types of tasks need to be executed and many workforces are required. Otherwise, each worker will need to carry out several tasks at a time. Jeon, Lee, and Ki (2017) have proposed a method to implement MRS to perform delivery services in a hospital environment which can bring the advantage of reducing the burden on hospital workers. The delivery task includes scheduled tasks like meals delivered to the patients and clothes delivered to the laundry at predefined pick-up and drop-off locations. The other delivery task is the urgent task such as transporting specimens and reports back and forth from the laboratories.

According to Jeon, Lee, and Kim (2017), a server with all the information regarding the mapping of the hospital is built. The server has the responsibility to store the delivery task and distribute the task to the specific robots. Firstly, all the robots will be located at their respective charging stations and their position will be uploaded to the server. Then, a multi-task allocation (MTA) algorithm will be utilized to distribute all the tasks to the robots with the shortest distance to reach the task point. While the robots are executing the tasks, the assignment of tasks will continue by the server, considering the current location of the robots to determine which robot is best suited for the task. Multiple tasks will be distributed to a robot and the robot has to perform each task in the pre-specified order. The server will check for the available task and status of the robots from time to time to ensure the condition of the robots and prevent missing out on any undistributed task. When an urgent task is requested, the server will select a robot with an optimum status and the shortest distance to the task point. Then, it will place the task in the first place in the task queue of that particular robot. Various sensors are mounted on the robots to allow the robots to navigate themselves during task execution as well as prevent obstacle collision.

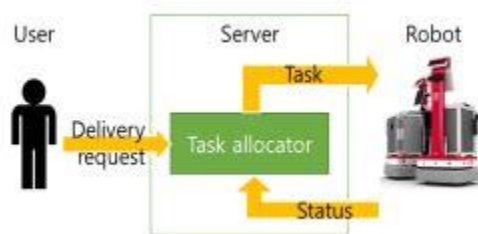


Figure 2.4: Task distribution process of MRS (Jeon, Lee and Kim, 2017)

Apart from that, MRS is widely utilized in the warehouse environment to increase the efficiency of the goods transporting system in the warehouse and also reduce labor costs. An auction-based algorithm is implemented in MRS to complete the constant tasks given in the warehouse. The warehousing environment is known by the robots. The role of the robots is to transport the shelves of the goods to respective picking stations and return the shelves to their original position (Fan,

Deng, and Shi, 2020). The performance of the robots is determined by the travel cost which is the distance between the robot and the shelf as well as the task completion time. When distributing the next task to the robot, the cost and time of the robot to complete the previous tasks will be taken into account. A robot with the best performance will win the task. Then the task execution is done after all the tasks have been allocated. The figure below illustrates the sample warehousing environment in simulation.

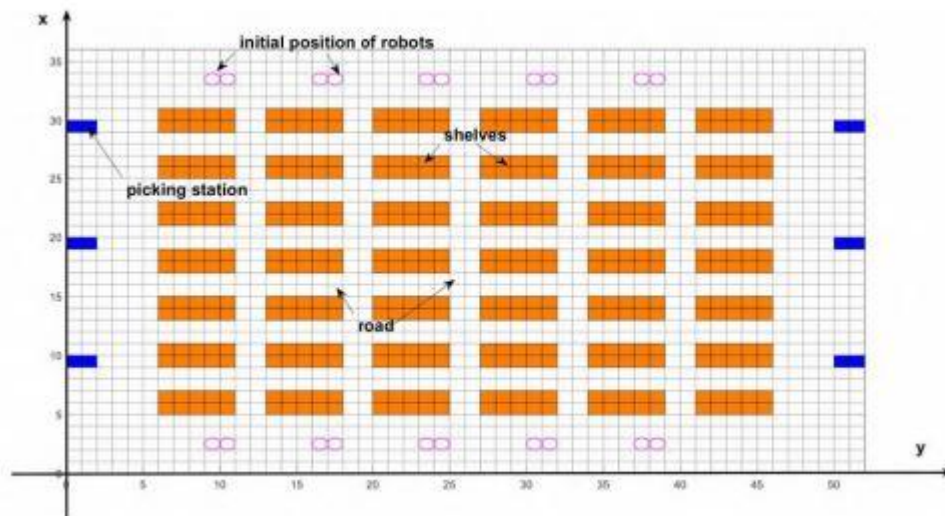


Figure 2.5: Simulated warehousing environment (Fan, Deng, and Shi, 2020)

However, dynamic tasks may appear in the warehouse which has a higher priority to be executed than other constant tasks. Another auction algorithm is proposed by Ma et al. (2016) whereby it can be applied in MRS to solve dynamic task distribution. The single task allocation method is utilized in which all the robots are distributed with one task for execution and the next task is distributed when a robot has accomplished the previous task. The performance of the robot is determined by the distance between the robot and the task point. At first, an auctioneer will form a sequence of tasks to be executed based on their priorities. Then, it will distribute the task to the robot with the shortest distance to the task position. When a task is completed, it will be terminated from the task set and the

subsequent task will be allocated to the robot which has completed the task. Whilst there is a dynamic task, a new task set will be formed by the auctioneer and the sequence is arranged based on the tasks' priorities. After that, the distribution process is repeated by choosing the robots with the optimal performance.

2.5.4 Multi-Robot Task Distribution for Manufacturing Industry

Undeniably, MRS has been implemented in the manufacturing industry to replace human workforces due to their high capabilities of performing repetitive tasks and reliability. Thus, task distribution in MRS is significant to boost the productivity of the industry while at the same time maintaining the quality of the products.

Huang, Zhang, and Xiao (2019) have proposed an auction algorithm to perform task distribution in heterogeneous MRS. Collaboration between heterogeneous robots is required to achieve success. These robots consist of two major capabilities which are transportation and detection abilities. Therefore, there are two types of tasks, mainly transportation and detection tasks at the beginning stage. These tasks have to be separated into two groups so that the tasks are suited to the respective robots. Then, each robot will calculate its respective earnings after completing the given tasks. The auctioneer will allocate the task to the robot with maximum earning. When there is more than one robot with the same earnings bidding for a common task, the auctioneer will allocate the task to the robot with the closest distance towards the task point to avoid bidding conflicts.

On the other hand, Cheng et al. (2019) have done research regarding task distribution for MRS with resources constriction in the assembly industry. There is a common workspace to perform the assembly activities. Multiple robots have been distributed with a task that can only be done at a particular workspace. Whereas each robot can perform one task for each visit to the workspace. As a result, the order of robots using the workspace is determined by the distance between the robots and the workspace. The robot which is the nearest to the workspace will have the priority to execute its task first followed by the subsequent robots until all the tasks have been executed.

2.5.5 Comparison Of Task Distribution Approaches In Multi-Robot System

Table 2.1: Comparison of Task Distribution Approaches

Type of problem	Algorithm	Coordination	Communication	Type of robot	Type of task
Exploration and destruction	Auction	Decentralized	Implicit	Homogeneous	Both static and dynamic tasks with an unknown environment
	Vacancy Chain	Decentralized	Implicit		
	Deep Q-learning	Decentralized	Implicit		
Search and rescue	Auction	Centralized	Explicit	Homogeneous	Both static and dynamic tasks with an unknown environment
	Prediction	Decentralized	Implicit		
Hospital and warehousing logistics	Auction	Centralized	Explicit	Homogeneous	Both static and dynamic tasks with a known environment
Manufacturing industry	Auction	Centralized	Explicit	Heterogeneous	The static task with a known environment

2.6 Simulation Software

To test the effectiveness of the task distribution algorithm, it shall be applied in the simulation software before implementing it with the hardware of the MRS. Simulation tools are useful to visualize the outcome and determine the workability of the developed algorithm. A simulation environment can be built which can mimic the real environment to predict the possible issues that will be encountered while running the algorithm. Several powerful simulation tools can be implemented together for the simulation of the task distribution algorithm in MRS. These tools include Robot Operating System 2 (ROS2), Navigation 2 (NAV2), Simultaneous Localization And Mapping (SLAM), and Gazebo.

2.6.1 Robot Operating System 2 (ROS2)

Robot Operating System 2 (ROS2) consists of a wide variety of development tools and libraries that are crucial for the software development of automated robot systems. ROS2 is capable of integrating the developed algorithm with the hardware to control an automated robotic system (Eros et al., 2019). ROS2 can apply in MRS with an outstanding communication network among the robots in the system. Besides that, it can be utilized in robots with different applications such as assembly robots, automated guided vehicles (AGVs) as well as self-driving vehicles. Two different types of programming languages can be used to develop an algorithm in ROS2 which are C++ and python which means that various types of microcontrollers are supported. The other benefit of ROS2 is real-time control whereby this ability is important to improve the performance of robots in a manufacturing environment.

The communication network of ROS2 is based on Data Distribution Service (DDS) which can support data transmission between multiple robots. According to Maruyama, Kato, and Azumi (2016), several aspects such as publisher, subscriber, nodes, topics, data writer, and data reader are essential to building the communication network among the robots. The nodes are connected to the respective topics whereby a publisher may transfer the message through nodes to

various topics. Then a subscriber can obtain the message from various topics via nodes. A data writer allows the transfer of a message from nodes to topics whereas a data reader allows the receiving of a message from topics to nodes. Node works as an agent to pass or receive a message. Whereas, the topic acts as a platform to allow the transmission of messages whereby the same type of message is categorized under one topic. With the implementation of these elements, data can be transmitted among the robots within the system, easing the execution of multiple robots. The figure below shows the example of message transmission between nodes and topics.

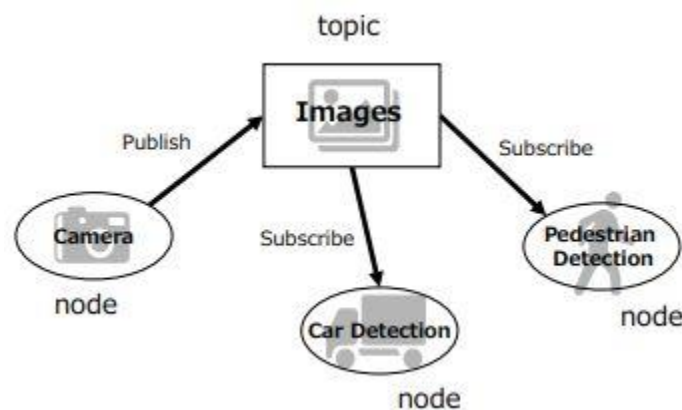


Figure 2.6: Message transmission between nodes and topics (Maruyama, Kato and Azumi, 2016)

2.6.2 Navigation 2 (NAV2)

Navigation 2 (NAV2) is a navigation tool integrated with ROS2 for the use of robot navigation in an environment such as a warehouse, stadium, and other open areas. The architecture of NAV2 is that it implements a behavior tree (BT) to control the task-based planning, control, and recovery servers whereby each server delegates a node in ROS2 (Macenski *et al.*, 2020). The main role of BT is to configure task planning and execution in robots. BT can be customized based on user requirements for the construction of a navigation scheme. The BT navigator is the most important

component in NAV2 which functions to perform and keep track of the tasks in all three servers, thus navigating a robot in an environment.

The planner server is responsible to calculate the shortest path for the robot to its destination. The controller server will obtain the information from the sensors on the robot to compute an optimum path for robot movement by sending the output control signals. Spatio-Temporal Voxel Layer (STVL) can provide a temporary 3D mapping over time by the data read from the sensors such as RGBD camera and laser scanner. With the combination of planner and STVL, a robot is capable of performing 3D perception and preventing collision with obstacles by obstacle detection and tracks formation in a dynamic environment. Moreover, the task in the recovery server is executed when a robot has encountered navigation issues. For example, a waiting task is executed by the robot when it is blocked by a dynamic obstacle to collect more sensors' information.

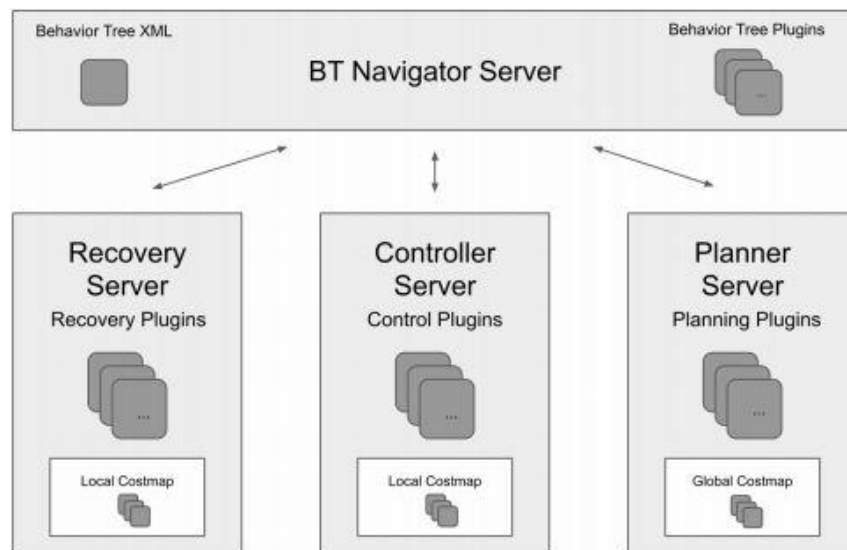


Figure 2.7: Overview of NAV2 architecture (Macenski *et al.*, 2020)

2.6.3 Simultaneous Localization And Mapping (SLAM)

Simultaneous Localization And Mapping (SLAM) is an impressive tool that enables a mobile robot to perform exploration tasks in an unknown environment.

Both localization and mapping processes can be done simultaneously by the robot while it is moving in an environment. Basically, SLAM comprises three main processes which are localization, mapping, and navigation (Khairuddin, Talib, and Haron, 2016).

First and foremost, the mapping process is done via the processing of data from the sensors on the robot to create a map regarding the environment explored. Then this map is used by the robot to proceed with the localization process. During localization, the robot can compute its trajectory and landmark its location from the created map. The localization process allows the robot to recognize the environment as well as avert the obstacles. In the navigation process, the robot may plan a proper route by analyzing the data gained from the sensors during mapping and localizing. The robot is capable to keep track of the path it moves so that it can retreat to its original position after completing an exploration task without getting lost in the environment. These three processes are performed recursively by the robot to allow it to move effectively in an environment without any prior knowledge.

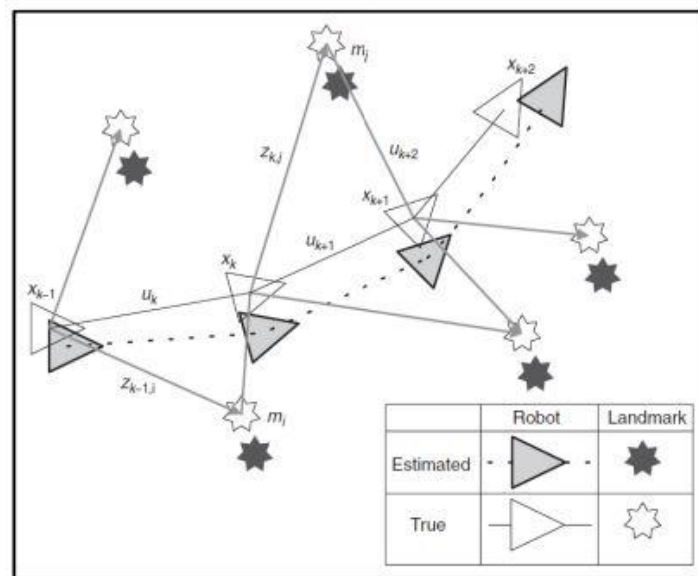


Figure 2.8: State of art of SLAM (Khairuddin, Talib and Haron, 2016)

2.6.4 Gazebo

The Gazebo is a simulation tool that is capable of generating a 3D environment for visualization and simulation of MRS. It has played a crucial role to test and evaluate the efficacy of developed concepts and algorithms in a virtual world (Robotic Simulation Services, 2021). With Gazebo, much money and time can be saved as the simulation is run virtually without the integration of hardware components. Therefore, the effectiveness of a robotic project can be assessed without huge expenses. The Gazebo can achieve high similarity between the simulation environment and the real-world environment with the mass, velocity, and other real-life parameters being applied to the simulated object, showing a high reality environment. Furthermore, Gazebo allows the integration of various sensors on the robots and evaluates the workability of the sensors with the robots. Without access to any hardware, numerous robots can be tested in Gazebo concurrently which is one of the most attractive features of this simulation tool.

2.7 Summary

In short, the process of task distribution in the Multi-Robot System (MRS) can be classified into four major steps which are task decomposition, task distribution, motion planning and control of MRS, and task execution. Task decomposition functions to break down a complex task into several simpler subtasks. Whereas task distribution is allocating the subtasks to the specific robots. Then, the robots have to plan their path and avoid obstacles while moving to the task position. Task execution is the last phase of the process whereby the algorithm and MRS are evaluated in the real environment. The coordination of MRS can be categorized into centralized and decentralized. A central agent is commonly used in the centralized approach which functions to control and communicate with the robots explicitly. Whereas no agent exists and implicit communication is implemented between the robots in the system in a decentralized approach.

Task distribution in MRS is implemented in many different problems such as exploration and destruction, search and rescue, hospital and warehousing logistics as well as the manufacturing industry. These problems consist of

heterogeneous or homogeneous MRS, static and dynamic task distribution. Various algorithms are introduced such as auction, vacancy chain, deep Q-learning, and prediction algorithm. Simulation software like ROS2, NAV2, SLAM, and Gazebo are useful in assessing the efficiency of the developed algorithm through simulation before implementing it in the real environment. ROS2 mainly provides a communication framework for the data transmission between the other tools. NAV2 allows the self-navigation of robots with the integration of sensors in an environment. SLAM provides the 3D perception of the environment and obstacle avoidance for the robots. Last but not least, Gazebo contributes a powerful virtual environment for the simulation and assessment of the algorithm on MRS.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

To step forward to the era of industrial 4.0, the Multi-Robot system (MRS) is being integrated persistently into the manufacturing industry to achieve a fully automated manufacturing system. When the system is completely automated without any intervention of a human, it will not only increase the productivity significantly but also maintain the good quality of the product by eliminating human errors. Therefore, the control system of the MRS is vital to allow the collaboration of the robots to accomplish a complex task. Without a doubt, task distribution will be involved in MRS so that a common goal can be attained by completing a sequence of tasks.

The main purpose of this project is to enable efficient task distribution for MRS. The manufacturing industry will indispensably utilize heterogeneous robots because a manufacturing process will consist of a combination of a group of different tasks. Hence, the assignment of the tasks to the specific robots is pivotal to enhance the efficacy of a manufacturing process. A powerful algorithm is essential to allow appropriate allocation of tasks to all robots in the system thus, saving time and cost for the manufacturing process. Upon distributing the tasks to the robots, the robots will need to move within the manufacturing environment during task execution. As a result, self-navigation of robots is important especially when many robots are moving in a restricted environment. Hence sensors can be integrated into the robots, allowing them to have the capabilities to avoid obstacles and collisions during task execution.

To achieve the objectives of this project, it can be categorized into several processes such as modeling of a manufacturing environment in the simulation, development of task distribution algorithm for MRS, motion planning of robots as well as evaluation and testing of the integration of the previous three steps in

simulation to assess its performance and workability. Traffic lanes will be utilized to navigate the robots in the environment. ROS2 will act as the main transmission line for the exchange of information and data for the utilized software. Last but not least, the model will be simulated virtually in Gazebo to visualize the overall implementation of this project.

3.2 Modeling of Simulation Environment

The first stage of task distribution in MRS is modeling a simulation environment. The robots can move freely within the environment to get a distributed task at a particular task point and perform task execution. All the events including robots' movement, task distribution, and task execution are being carried out within the modeled environment. The architecture of the modeled environment is based on a typical manufacturing environment. As the modeled environment is considered as a known environment whereby the location of the task points, obstacles, robots' initial location, and other features of the environment are predetermined.

There are two different types of tasks which are delivery tasks from the warehouse to the workspace and delivery tasks between specific workspaces. Thus, heterogeneous robots are utilized in which the robots have different abilities and features that will suit the different tasks respectively. The first type of robot has a smaller size and load capability that functions to transport the required item from the warehouse to a specific zone. Whereas the second type of robot with a larger size and load capability is responsible to transfer the item from the zones to other workspaces or zones. The items specified in the simulation environment will be the square boxes.

The simulation environment will be modeled via traffic editor which is a software modeling tool in Robotics Middleware Framework (RMF). RMF is a software tool that can build a simulation environment. It can also perform communication between the heterogeneous robots and the infrastructures with the integration with ROS2. The manufacturing environment is made up of a total of four robots with two robots for each of the delivery tasks respectively. There are four task points in the warehouse and five zones in the production area. All the

robots are initially located at their respective charging locations which are their original position in the environment. Moreover, the traffic editor will be used to model the walls, obstacles, and task points of the manufacturing environment. The draft of the simulation environment is shown in the figure below:

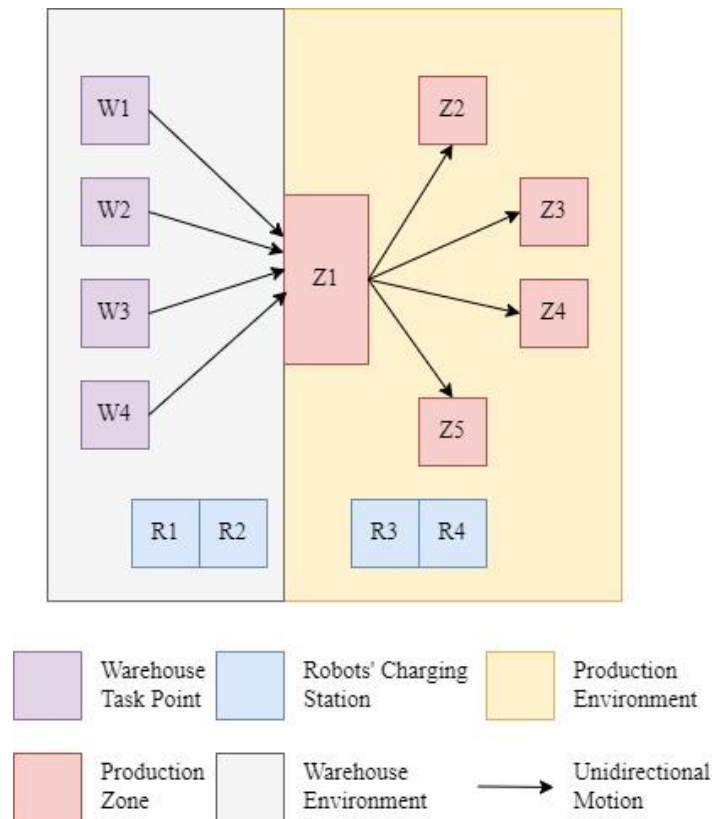


Figure 3.1: Overview of the simulated manufacturing environment

3.3 Task Distribution Algorithm for Multi-Robot System

After modeling the simulation environment, the subsequent stage will be the development of a task distribution algorithm for the MRS. The algorithm developed is based on an auction-based approach whereby a bidding system is implemented to assign the task to a particular robot. Undeniably, RMF can manage the task distribution in MRS through the multi-fleet system in which each type of robot has a fleet adapter. A fleet adapter will act as the central agent for the robot which enables the communication between the main system and the robot. A Dispatcher node will manage the message transfer between the main system and the fleet adapter of each robot.

Firstly, when there is a new task, the main system will pass the task request to the Dispatcher node. Then, the Dispatcher will further transmit this request to each fleet adapter. At this moment, the task request will become a bidding request whereby the fleet adapters of the robots that are capable of performing the task will submit a bidding request for the task back to the Dispatcher. After that, the Dispatcher will evaluate the bidding request from the fleet adapter which may depend on the parameters such as the duration of task execution and the distance between the task point and the robot. Then it will judge which robot has the best performance on the task. Lastly, the Dispatcher will allocate the task to that robot by ignoring the other robots that have sent the bidding request. The selected robot will then proceed with the task execution.

The task distribution process is repeated when a new task is dispatched. Only robots with the suited capability will be able to participate in the task bidding process and one task will be assigned to only one robot with the shortest task execution time. Furthermore, a dynamic task that is considered an urgent task can be added to the task queue at any time which will preempt the other tasks in the queue. The task shall be arranged in the first order so that it can be distributed to the robots as soon as possible. Hence, the development of a task distribution algorithm is crucial to ensure that the task is distributed correctly to a specific robot. The figure below shows the communication within the system during the task distribution process.

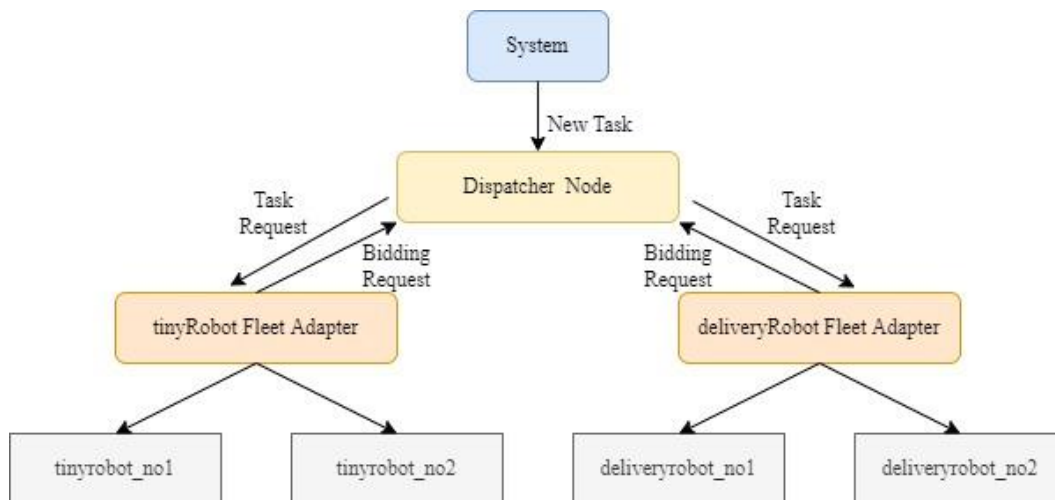


Figure 3.2: Communication flow during task distribution

3.4 Motion Planning of Robots

Apart from that, the motion planning of MRS is the next significant phase after the task distribution process of the robots. The motion planning of robots is dependent on the traffic lane built in the simulation environment that is connecting to their initial positions. The traffic lane must be designed properly so that the robots can move smoothly within the environment with minimum traffic conflicts. As the simulation environment is static with all the task points are pre-known by the robots thus the robots will plan for the path to reach the task point and deliver the cardboard box to the respective destination.

On the other hand, the fleet adapter will utilize the path information of the robots' traffic lanes to calculate the shortest path for the robots to move to their desired location such as the task point and destination. This feature will optimize the time for the robots to travel around the environment during task execution. When the robot encounters a dynamic obstacle, the fleet adapter will command the robot to perform a corresponding action to prevent the robot from colliding with the obstacles. For instance, the robot may wait for a particular time for the obstacle to leave or replan the moving path to pass by the obstacle. As a result, traffic lanes created in the environment will prevent the collision of robots with the obstacles which will further smoothen the motion of the robots.

RMF traffic is another robust tool that utilizes the main platform named Traffic Schedule Database that can resolve the traffic conflicts between the robots. The database functions to collect the information of the robots including their trajectories and locations which is submitted by the fleet adapter of each of the robots. When there is a change in moving path or delays, the fleet adapter will again update the information to the database. Therefore, by acquiring the required data from the traffic schedule, the fleet adapters can perform path planning of respective robots to achieve their desired destination. With the data collected in the traffic schedule, the conflict between the paths of the robots can be avoided. However, the prevention of traffic conflicts between the robots may not work perfectly due to some unforeseen circumstances. So, the RMF traffic has introduced another scheme which is Negotiation whereby a system integrator will allow a higher priority robot

to proceed with the path with another robot aborting its original path. The negotiation will take place between the fleet adapters when the traffic scheduler detects an upcoming conflict which may be due to an urgent task added and it will notify the fleet adapters of the robots. The figure below shows the communication architecture between the traffic schedule database, fleet adapter, and the robots.

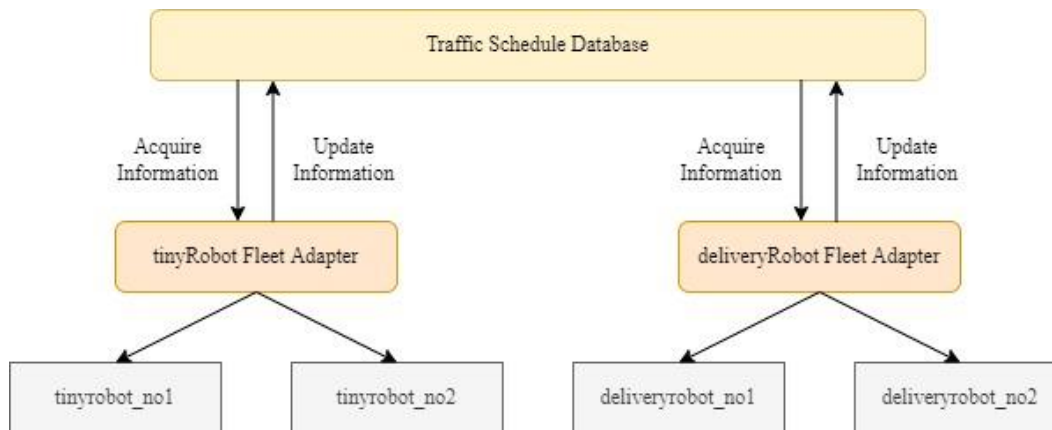


Figure 3.3: Communication architecture between traffic schedule database, fleet adapter and robots

3.5 Evaluation and Testing of Algorithm

After developing the algorithm for task distribution in MRS, the algorithm must undergo testing to evaluate its effectiveness and workability. The simulation of the whole model will be carried out in Gazebo, a powerful simulation tool that can mimic almost every activity in the real manufacturing environment. Apart from that, the motion planning of robots can be visualized through RViz2 which is a ROS visualizer to monitor the robots' motion in the simulation environment.

Undoubtedly, the modeled manufacturing environment can be brought into the simulation. Besides, the algorithm to perform task distribution of MRS and motion planning of the robots can also be simulated through Gazebo. Any possible problem that will be encountered by the robots can be identified through simulation. Then modification and improvement can be implemented to tackle the issue as soon as possible. Therefore, instead of improving the hardware part, the modification can be done in the simulation software whereby much time and cost can be saved as well as easing the whole designing process. The simulation tool also allows the

designer to endlessly repeat the testing and modification process of the developed algorithm until a satisfactory result is obtained.

3.6 Summary

In short, this project aims to develop an algorithm that can implement the task distribution in a heterogeneous Multi-Robot system in a manufacturing environment efficiently. To achieve the main goal of this project the development process is classified into four phases which are modeling of the simulated manufacturing environment, the development of an algorithm to perform task distribution in MRS, followed by motion planning of the robots within the environment, and lastly is the testing and evaluation of the developed algorithm and designed model through simulation to assess their capability and performance.

First and foremost, the modeling process of the simulated manufacturing environment is done via RMF traffic editor so that the modeled environment is static and a known environment. Several settings are made such as the environment is modeled based on the manufacturing industry, heterogeneous robots will be utilized with a total number of four robots, and the locations of the robots and the task points are predefined. In the second phase, an algorithm will be developed which will implement a bidding process to distribute the task evenly among the robots. RMF fleet adapter will act as the communication agent between the centralized system and the robot to bid for the tasks given. An urgent task can be added dynamically into the system whereby this task will preempt other delivery tasks and will be executed by the robots priorly.

After that, motion planning and control of the robots in the MRS will be implemented via traffic lanes and fleet adapters of robots. The fleet adapters will control the robots to self-navigate within the environment during task execution with the ability to avoid obstacles and prevent collision. RMF traffic scheduler is a database that has all the updated information regarding the location and planning path of the robots. It has an important role to provide the robot's information to the fleet adapter to plan the path for the robot and handle the traffic conflicts between the robots through a negotiation scheme. The last phase will be the assessment and

evaluation of the designed model and the developed algorithm in a simulation tool which is Gazebo. Modification of the designed model as well as the algorithm can be made when any possible issue happened during simulation. Hence, the performance of the algorithm is tested repeatedly through simulation until an optimum result is achieved.

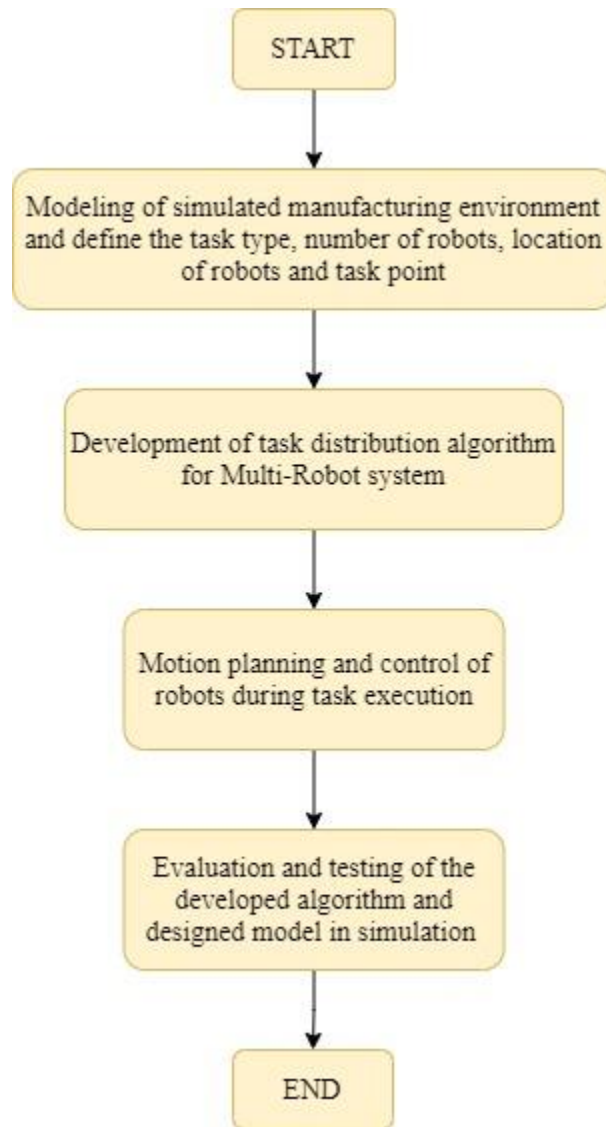


Figure 3.4: Process flow of task distribution in Multi-Robot system

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

After planning of the project, the first development process is the design and construction of a simulation environment which is referencing to a manufacturing environment that consists of only one floor. The position and number of the models which are essential for constructing the environment such as tables, items to be delivered, and robots are properly initialized in the environment. Then, the traffic lane with proper design is added to the environment to create a moving path for the robots. It is crucial to optimize the movement of robots within the environment with minimum traffic conflicts.

The finalized modeled environment will be simulated in Gazebo to validate its workability. The type of task to be distributed among the robots is the delivery task. RMF is integrated into the simulation to enable the fleet adapters of the robots to bid for the newly created delivery task. After assigning the task to the robot with the best bidding result, the fleet adapter will initiate the movement of the robot to the pickup task point to deliver the desired item to the drop-off task point. RMF Panel provides an online platform to ease the user to perform the submission of delivery tasks to the robots in the simulation. The task bidding process is repeated when a new task is created and all the assigned tasks will be added to the task queue which will be executed by the specific robots orderly.

Upon task execution, the robots will move in the constructed traffic lane automatically to perform the deliveries. The moving path is planned by the fleet adapter of robots after receiving the task point position. When more than one robot moves in the same lane, there is a traffic conflict whereby it will be resolved by the fleet adapters of the robots.

The simulation of the delivery tasks is evaluated in Gazebo whereas the robots' planned paths and their real-time locations are visualized in RViz2. Other

than that, RMF Panel also allows the user to monitor the robots' status during task execution. The task distribution algorithm is evaluated by the simulation among these three platforms with the integration of RMF.

4.2 Modeling of Simulation Environment

The first development process of this project is the modeling of a manufacturing environment which will be used for the simulation of the task distribution of a Multi-Robot system. The modeling tool of the simulation environment is traffic editor which is one of the useful software integrated within RMF. The figure below shows the 2D layout of the simulation environment in the traffic editor.

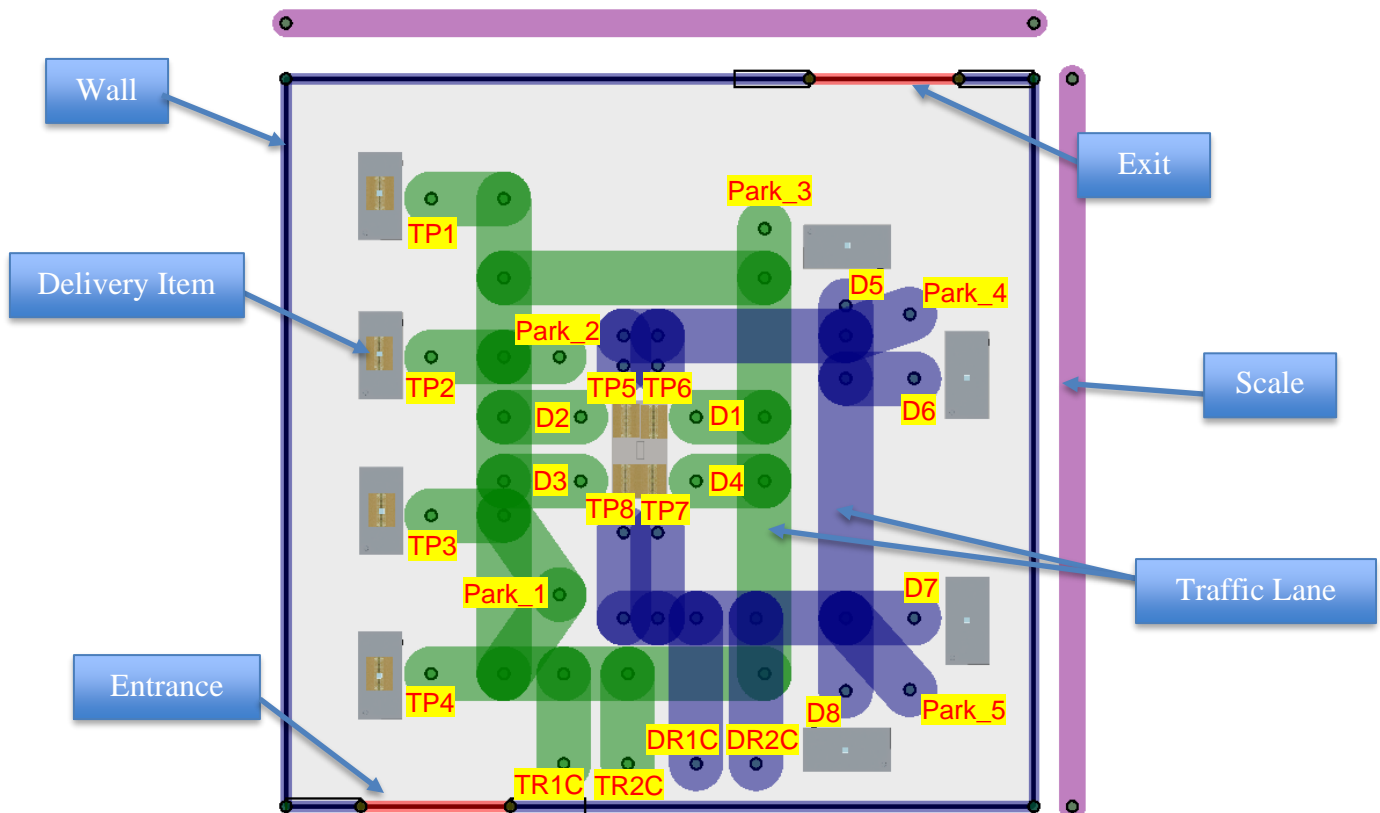


Figure 4.1: 2D layout of simulation environment

The modeled environment is referenced to a typical manufacturing environment in an industry. It consists of one floor with an area of 10m X 10m which is indicated by the scaling line in purple color and surrounded by a wall with a height of 2m. The entrance and exit of the environment are indicated by a double sliding door respectively with a width of 2m as shown in the diagram above. The

main components that are crucial for the simulation of the delivery tasks are adjustable tables, rectangle tables, and cardboard boxes.

As shown in Figure 4.1, four adjustable tables are located on the left side of the environment which made up task points 1 to 4 (TP1, TP2, TP3, TP4). Whereas, the right side of the environment consists of another four adjustable tables that build up destinations 5 to 8 (D5, D6, D7, D8). Task points 5 to 8 (TP5, TP6, TP7, TP8) and destinations 1 to 4 (D1, D2, D3, D4) are located in the middle of the environment which consists of a rectangle table. The delivery item that is used for the delivery tasks is a cardboard box whereby it is placed on the table of every task point.

Moreover, there are four robot charging stations in the environment as shown at the bottom of Figure 4.1, represented by tiny robot 1 charging (TR1C), tiny robot 2 charging (TR2C), delivery robot 1 charging (DR1C), and delivery robot 2 charging (DR2C). Hence, there are two heterogeneous robots used for the simulation of delivery tasks which are two tiny robots and two delivery robots. The green and blue paths represent the traffic lane for tiny robots and delivery robots respectively. There are five holding points for the robots to wait at those locations to let another robot with higher priority to use the lane first when there is traffic conflict between the robots. Park_1, Park_2, and Park_3 are the holding points for tiny robots and Park_4 and Park_5 are the holding points for delivery robots.

4.3 Task Distribution of Heterogeneous Robots

After modeling the simulation environment, the task distribution algorithm is integrated into the simulation to allow efficient allocation of the delivery tasks among the robots which are two tiny robots and two delivery robots. As illustrated in the figure below, tiny robots are responsible for the delivery tasks created at TP1, TP2, TP3, and TP4 whereas delivery robots are responsible for the tasks at TP5, TP6, TP7, and TP8. The cardboard box at the task point will be delivered to the corresponding destination such as TP1 to D1, TP2 to D2, TP3 to D3, TP4 to D4, TP5 to D5, TP6 to D6, TP7 to D7, and TP8 to D8. The orange and blue paths indicate the traffic lane for tiny robots and delivery robots respectively. As both

types of robots are assigned to different traffic lanes therefore the robots only compete for the tasks within the same traffic lane without interrupting the task bidding of the other robots in the different traffic lanes.

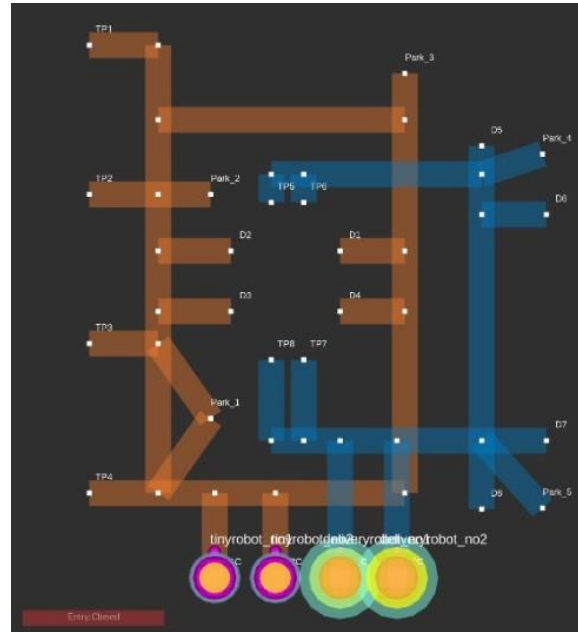


Figure 4.2: Traffic lanes of robots for delivery task distribution

RMF Panel acts as an online platform to ease the submission of tasks by the user to trigger the task bidding among the robots and robot motion for task execution in the simulation environment. The layout of the RMF Panel is shown below.

RMF Panel

RMF: Manufacturing World

Time is: 0:0:11

Task Submissions

Submit a Task

Select a request type

Set start time (mins from now)

Choose a priority (Default: 0)

Schedule a Delivery Request

Select delivery task

SUBMIT REQUEST

Submit a List of Tasks

SELECT FILE

```
eg. [
{"task_type": "Loop", "start_time": 0, "priority": 0, "description":
{"num_loops": 5, "start_name": "coe", "finish_name": "lounges"}},
{"task_type": "Delivery", "start_time": 0, "priority": 0, "description":
{"option": "coke"}}]
```

SUBMIT TASK LIST

Figure 4.3: RMF Panel for task submission

RMF Panel not only allows submission of a single task but also supports multiple tasks submitted at once by the user. To submit a single task, a user needs to select the task type and task point as well as input the desired value for the task priority and start time from the display window of the RMF Panel as shown in Figure 4.3. On the other hand, the user is required to upload a .json file that contains a list of tasks to be executed if multiple tasks are submitted at once. An example of a .json file is displayed in APPENDIX A.

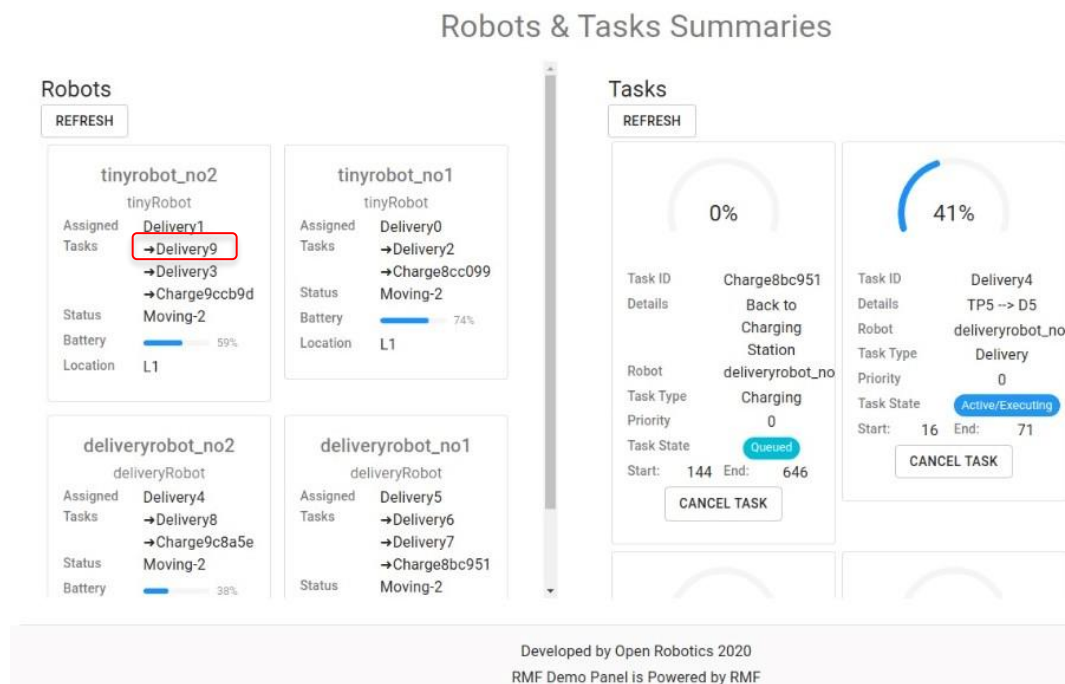


Figure 4.4: RMF Panel for monitoring of task execution progress and robot status

Upon task submission, a task request is sent to the fleet adapter of each type of robot from the dispatcher node. The fleet adapter which has the task point in its traffic lane for the delivery task responds to the task request. Then, the fleet adapter submits the task execution time for each of its robots as the bidding request to the dispatcher node. The dispatcher node is the auctioneer which allocates the task to the respective robot with the best bidding result through a centralized approach. The task execution time is calculated based on the distance between the robot and the task point and the robot with the shortest execution time is selected to perform the delivery task. After that, the fleet adapter initiates the movement of the selected robot.

RMF Panel also enables the user to monitor the task execution progress by the robots as well as robots' status such as the currently assigned task, task queue, and battery status. After the task bidding process, the delivery task is added to the task queue of the robots. A charging task is assigned to the robot when it finishes its last delivery task to prevent the battery of the robot from draining off. As a result, the robots are always charged up and ready to execute another task. While an urgent task that has a higher priority is distributed, the task is allocated to the first place in the task queue of the robot as highlighted in Figure 4.4. Hence, the urgent task is completed immediately with minimum delay.

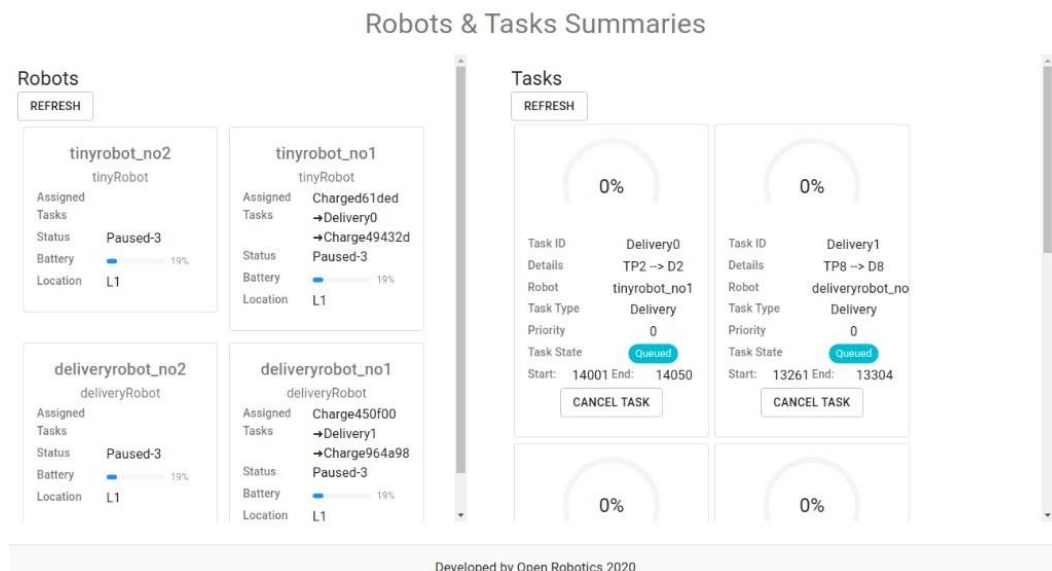


Figure 4.5: Task distribution during robots' battery threshold

Moreover, when a task is distributed to a robot with a battery level lower than its threshold value, a charging task will be assigned to the robot to enable it to be charged up before executing the delivery task which can be shown in Figure 4.5. Once the robot's battery is full, it will resume the assigned delivery task. Thus, the robot does not face any battery shortage while carrying out the following delivery task.

4.4 Path Planning of Heterogeneous Robots

The subsequent process after task distribution in a Multi-Robot system is path planning of robots during task execution of the delivery tasks. After the robot has been allocated with a delivery task, its corresponding fleet adapter initiates the movement of the robot. Therefore, the robot is self-navigating to the respective task point to pick up the delivery item which is a cardboard box followed by navigating to the desired destination to drop off the cardboard box. Once the cardboard box has been delivered from a task point to the corresponding destination, the delivery task is deemed to be accomplished successfully. The path planning and motion of robots are visualized in ROS Visualization 2 (RViz2).

During self-navigation, the robots only move along the predetermined paths whereby tiny robots navigate along the orange path and delivery robots navigate along the blue path. Tiny robots never intervene in the navigation path of delivery robots and vice versa so that the traffic conflicts among the robots are minimized as well as enhancing the navigation of the robots within the simulation environment. The two types of robots are controlled by their respective fleet adapters through explicit communication. Once there is a task assigned to a robot, the fleet adapter of the robot is triggered to perform path planning to seek the shortest path for the robot to move to its task point and from the task point to its destination. Both the fleet adapters will update the planned path and the real-time location of their robots to the traffic schedule database from time to time. Then, the fleet adapters will also acquire the planned path from each other through subscription to the traffic schedule database. As a result, the traffic schedule database is playing the role as the messages and information transmission center for the communication between fleet adapters of heterogeneous robots.

The traffic schedule database is significant as it contains all the essential path information of the robots which is required by the fleet adapters to conduct negotiation while there is a traffic conflict among the robots. As some portion of the traffic lane of tiny robots and delivery robots is overlapped, thus traffic conflicts occur frequently during the navigation of the robots along their respective paths. Besides that, the robots encounter traffic conflicts when more than one robot is

using the same path. Figures 4.6 and 4.7 indicate traffic conflict scenarios among the robots during the execution of the delivery tasks.

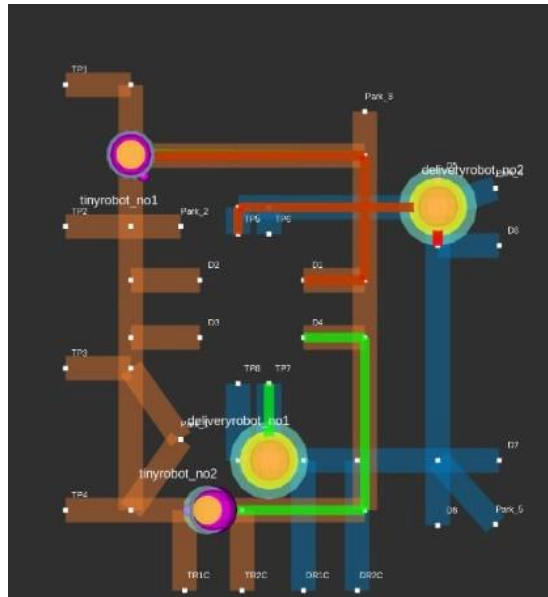


Figure 4.6: Visualization of traffic conflict among robots

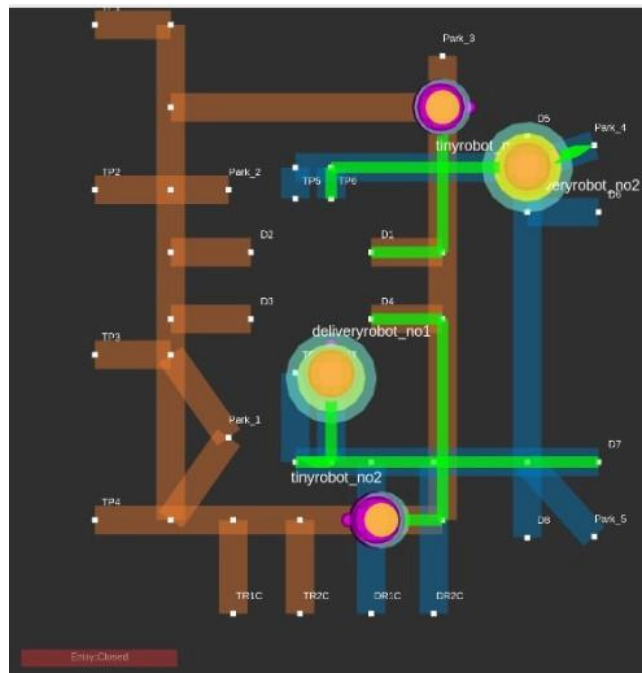


Figure 4.7: Traffic conflict resolved among robots

The red path shown in Figure 4.6 indicates there is a traffic conflict among the robots which is due to the overlapping paths of robots. When there is a traffic conflict, negotiation is implemented among the fleet adapters of the robots to

determine which robot is going to use the path priorly. The green path indicates there is no traffic conflict among the robots or a traffic conflict is resolved which is illustrated in Figure 4.7. During traffic conflict, the robot that has a shorter task execution time will have higher priority as compared with the other robot. Hence, in the case shown in Figure 4.7, `tinyrobot_no1` is preceding `deliveryrobot_no2` to move in the path because it has almost arrived at its destination whereby has a shorter task execution time as compared to `deliveryrobot_no2`. Therefore, `deliveryrobot_no2` will wait at the nearby holding point which is `Park_4` to allow the motion of `tinyrobot_no1`. Then, `deliveryrobot_no2` will resume its path after `tinyrobot_no1` has passed by and the traffic conflict is resolved.

During navigation, the robot will also stop to avoid collision with the other robots or obstacles. This is shown in the figure below whereby the tiny robot stops its motion when it encounters an obstacle ahead which is a cardboard box. After stopping, the robot will wait infinitely until the cardboard box is moved away from its path. As a result, the robot has obstacles and collision avoidance capability while navigating by waiting in its path until the obstacles are removed.

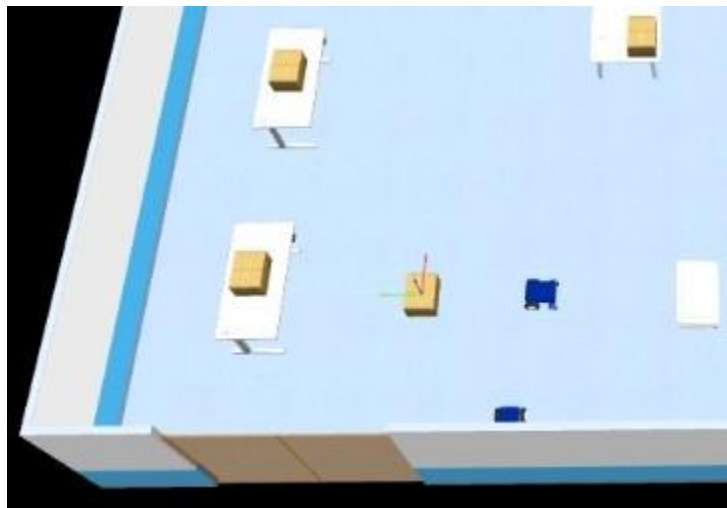


Figure 4.8: Collision and obstacle avoidance by tiny robot in Gazebo

4.5 Simulation and Evaluation of Task Distribution of Heterogeneous Robots

To evaluate and validate the workability of the algorithms, the simulation model created in the traffic editor is converted to a file that can be loaded into Gazebo to enable the simulation. In addition, a model file that consists of the 3D models of the simulation environment such as the walls, double sliding doors, and floor is created during the conversion. This model file is added to the Gazebo model path along with the other model files including tiny robot, delivery robot, cardboard box, adjustable table, and rectangle table. As a result, the simulation environment is loaded successfully into Gazebo without any missing model.

Furthermore, the traffic lanes in the traffic editor are converted to the navigation graphs of the tiny robots and delivery robots respectively. The traffic lanes are the core elements to visualize the path planning and motion of the robots in RViz2. With the integration of the navigation paths into the simulation environment, it allows the fleet adapters to command their respective robots to move along the paths within the environment automatically. Undeniably the navigation graphs of robots are important to provide a better visual for the user to monitor the motion of the robots in the simulation environment.

With the installation of Gazebo plugins and RViz2 plugins in RMF, the distribution of delivery tasks among the robots and robots' movements are simulated perfectly. The launch files to bring up the simulation in Gazebo and visualization in RViz2 are appended in APPENDIX B. The delivery tasks are allocated effectively to the correct robots during task distribution with the proper utilization of traffic lanes to distinguish the different types of robots in a heterogeneous Multi-Robot system. Hence, the distribution of delivery tasks to the robots with unmatched capability is prevented. The figure below illustrates the simulation of delivery task distribution among the heterogeneous robots in a mimic manufacturing environment.

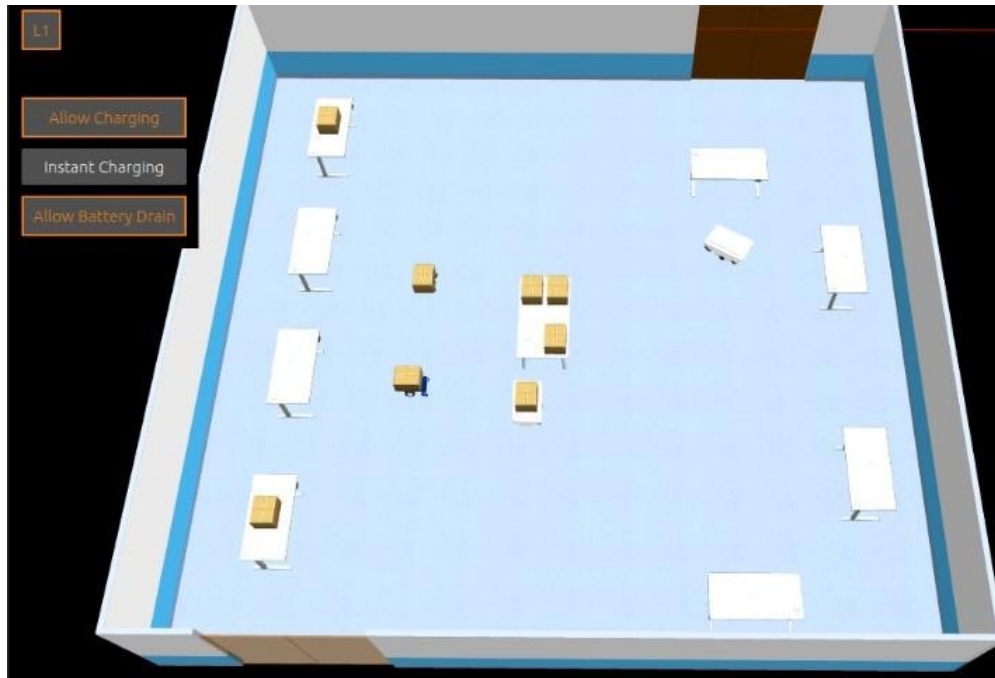


Figure 4.9: Simulation of task distribution of heterogeneous robots in Gazebo

4.6 Summary

In a nutshell, the RMF traffic editor is the modeling tool to design a 2D layout of the simulation environment. During the modeling process, some of the basic elements are drawn such as the floor, walls, and doors. Then, the essential components for the simulation of robots are added which include adjustable tables, cardboard boxes, rectangle tables, tiny robots, and delivery robots. After that, the traffic lanes are added to the tiny robots and delivery robots respectively to ease the task distribution process among the robots. Several task points are set to the tiny robots which are TP1 to D1, TP2 to D2, TP3 to D3, and TP4 to D4 whereas task points TP5 to D5, TP6 to D6, TP7 to D7, and TP8 to D8 are responsible by the delivery robots. A delivery task is said to be completed successfully when a cardboard box is delivered from a task point to its corresponding destination.

Before the task distribution process, RMF Panel provides an online platform for the user to either submit a single task or multiple tasks at once. Once a delivery task is submitted, the fleet adapter of the robots which has the corresponding task point in its traffic lane is initiated to submit the bidding request of its robots to the dispatcher node. The bidding request is the task execution time of the robot to carry

out the delivery task whereby it is calculated by the distance of the robot to the task point. Whereas the dispatcher node will act as the auctioneer to assign the task to the robot with the shortest execution time. The RMF Panel also allows the user to monitor the task execution progress by each of the robots as well as the status of each of the robots. When there is an urgent task being submitted that has a higher priority, the task is assigned to the first place in the task queue of the respective robot. Thus, the urgent task is executed as fast as possible. A charging task is assigned to the robot after it has finished its final delivery task to ensure it is charged up, preventing its battery from draining off. When the battery level of the robot is below its threshold value, a charging task will be executed by the robot followed by the delivery task to make sure the robot has enough battery to perform the task.

During task execution, the fleet adapters submit the planned path and real-time location of robots to the traffic schedule database from time to time to perform negotiation when a traffic conflict occurs between the robots. The path planning and robots' real-time location are visualized in RViz2. Furthermore, the robot waits at the nearest holding points which are Park_1, Park_2, Park_3 or Park_4 to let the other robot to use the path first as a solution to resolve the traffic conflict. The robot with lower task execution time left will use the path priorly during a traffic conflict. When there is an obstacle present in the path, the robot stops immediately to avoid collision and waits for the obstacle to move away from the path.

The modeled environment in the traffic editor is converted to a file to be loaded into Gazebo and the traffic lanes of robots are translated to navigation graphs to allow path planning and self-navigation of robots to execute a delivery task. With the integration of Gazebo, RViz2, and RMF, the simulation of task distribution of heterogeneous robots is implemented successfully in Gazebo with visualization of path planning and robots' real-time location in RViz2.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

In conclusion, the main objective of this project is to perform efficient task distribution for a heterogeneous Multi-Robot system whereby it is achieved by integration of RMF. The misallocation of the delivery tasks to the robots with unmatched capabilities is eliminated by the separation of different traffic lanes for the respective robots and task points. Hence, the same type of robot is allocated in the same traffic lane which is connected to the delivery task points and destinations. With the integration of two different traffic lanes to the robots, the tiny robots do not intervene in the delivery task created at the task points of delivery robots and vice versa which shows a success in the initial distribution process of the delivery task.

During task distribution, a delivery task is assigned to a robot with the shortest task execution time. Task execution time indicates the shortest distance for the nearest robots to reach the task points. When an urgent task with higher priority is created, the task is allocated in the first place in the task queue of robots in which the algorithm enables effective task distribution among the robots. Besides that, a charging task is performed by the robots after their last delivery task is completed to ensure the robots are charged up which mimics the scenario in a real manufacturing environment.

Last but not least, the robots are able to navigate automatically for task execution in the simulation environment with the integration of traffic lanes as the navigation graphs for the robots and communication among the fleet adapters of the robots. A traffic conflict occurs when more than one robot is moving in the same traffic lane which requests the fleet adapter to replan the path of the robots. Thus, a robot stops and waits at the nearest holding point or uses another path to reach its task point as a solution to resolve the traffic conflict. As a result, resolving the traffic

conflict enable the robots to prevent collision with other robots. Therefore, the second and third objectives of the project are met.

5.2 Recommendations for future work

Some of the recommendations for future development of the project include improvement on the task distribution algorithms for the robots by assigning the robots to execute a charging task immediately when the battery level of the robots drops below its threshold value while the robots are executing their respective delivery tasks. Apart from that, a more complicated simulation environment with multiple floors can be built by the integration of different types of doors and lifts to evaluate the path planning of the robots during task executions.

In addition, moving obstacles such as humans can also be added into the environment to test the navigation capabilities of the robots. The solution to resolve the navigation conflict is suggested to improve by allowing the robots to replan their navigation path when a static obstacle unpredictably presents in the robots' path. Lastly, the algorithm can be improved to support more task types which allow the simulation of a wide variety of tasks in different types of industries.

REFERENCES

- Chen, J., Yang, Y. and Wei, L., 2010. Research on the approach of task decomposition in soccer robot system. *Proceedings - 2010 International Conference on Digital Manufacturing and Automation, ICDMA 2010*, 2, pp.284–289.
- Cheng, Y., Sun, F., Zhang, Y. and Tao, F., 2019. Task allocation in manufacturing: A review. *Journal of Industrial Information Integration*, [online] 15, pp.207–218. Available at: <<https://doi.org/10.1016/j.jii.2018.08.001>>.
- Dai, W., Lu, H., Xiao, J., Zeng, Z. and Zheng, Z., 2020. Multi-Robot Dynamic Task Allocation for Exploration and Destruction. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 98(2), pp.455–479.
- Doriya, R., Mishra, S. and Gupta, S., 2015. A brief survey and analysis of multi-robot communication and coordination. *International Conference on Computing, Communication and Automation, ICCCA 2015*, pp.1014–1021.
- Eros, E., Dahl, M., Bengtsson, K., Hanna, A. and Falkman, P., 2019. A ROS2 based communication architecture for control in collaborative and intelligent automation systems. *Procedia Manufacturing*, [online] 38(2019), pp.349–357. Available at: <<https://doi.org/10.1016/j.promfg.2020.01.045>>.
- Fan, Y., Deng, F. and Shi, X., 2020. Multi-robot Task Allocation and Path Planning System Design. *Chinese Control Conference, CCC, 2020-July(2)*, pp.4759–4764.
- Gildert, N., Millard, A.G., Pomfret, A. and Timmis, J., 2018. The need for combining implicit and explicit communication in cooperative robotic systems. *Frontiers Robotics AI*, 5(JUN), pp.1–6.
- Huang, Y., Zhang, Y. and Xiao, H., 2019. Multi-robot system task allocation mechanism for smart factory. *Proceedings of 2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference, ITAIC 2019*, (Itaic), pp.587–591.
- Jeon, S., Lee, J. and Kim, J., 2017. Multi-robot task allocation for real-time hospital logistics. *2017 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2017*, 2017-Janua, pp.2465–2470.
- Khairuddin, A.R., Talib, M.S. and Haron, H., 2016. Review on simultaneous localization and mapping (SLAM). *Proceedings - 5th IEEE International*

Conference on Control System, Computing and Engineering, ICCSCE 2015, (November), pp.85–90.

Khamis, A., Hussein, A. and Elmogy, A., 2015. Multi-robot task allocation: A review of the state-of-the-art. *Studies in Computational Intelligence*, 604(May), pp.31–51.

Li, D., Fan, Q. and Dai, X., 2017. Research status of multi - Robot systems task allocation and uncertainty treatment. *Journal of Physics: Conference Series*, 887(1).

Luo, L., Chakraborty, N. and Sycara, K., 2015. Distributed Algorithms for Multirobot Task Assignment with Task Deadline Constraints. *IEEE Transactions on Automation Science and Engineering*, 12(3), pp.876–888.

Ma, H., Wu, X., Gong, Y., Cui, Y. and Song, J., 2016. A Task-Grouped Approach for the Multi-robot Task Allocation of Warehouse System. *Proceedings - 2015 International Conference on Computer Science and Mechanical Automation, CSMA 2015*, pp.277–280.

Macenski, S., Martin, F., White, R. and Clavero, J.G., 2020. The marathon 2: A navigation system. *IEEE International Conference on Intelligent Robots and Systems*, (February), pp.2718–2725.

Maruyama, Y., Kato, S. and Azumi, T., 2016. Exploring the performance of ROS2. *Proceedings of the 13th International Conference on Embedded Software, EMSOFT 2016*, pp.0–9.

Ravankar, A., Ravankar, A.A., Kobayashi, Y., Hoshino, Y., Peng, C.C. and Watanabe, M., 2018. Hitchhiking based symbiotic multi-robot navigation in sensor networks. *Robotics*, 7(3), pp.1–28.

Rizk, Y., Awad, M. and Tunstel, E.W., 2019. Cooperative heterogeneous multi-robot systems: A survey. *ACM Computing Surveys*, 52(2).

Robotic Simulation Services, 2021. *ROS Gazebo: Everything You Need To Know - Robotic Simulation Services*. [online] Available at: <<https://roboticsimulationservices.com/ros-gazebo-everything-you-need-to-know/>> [Accessed 27 Jul. 2021].

Wei, C., Hindriks, K. V. and Jonker, C.M., 2016. Dynamic task allocation for multi-robot search and retrieval tasks. *Applied Intelligence*, [online] 45(2), pp.383–401. Available at: <<http://dx.doi.org/10.1007/s10489-016-0771-5>>.

Zaidi, L., Sahnoun, M. and Bettayeb, B., 2019. Task allocation based on shared resource constraint for multi-robot systems in manufacturing industry. *IFAC-PapersOnLine*, [online] 52(13), pp.2020–2025. Available at: <<https://doi.org/10.1016/j.ifacol.2019.1>

APPENDICES**APPENDIX A: Multiple task .json file sample**

```
[
{"task_type":"Delivery", "start_time":0, "priority":0,
"description": {"option": "TP1"}},
{"task_type":"Delivery", "start_time":0, "priority":0,
"description": {"option": "TP4"}},
{"task_type":"Delivery", "start_time":0, "priority":0,
"description": {"option": "TP2"}},
{"task_type":"Delivery", "start_time":0, "priority":0,
"description": {"option": "TP3"}},
{"task_type":"Delivery", "start_time":0, "priority":0,
"description": {"option": "TP5"}},
{"task_type":"Delivery", "start_time":0, "priority":0,
"description": {"option": "TP8"}},
{"task_type":"Delivery", "start_time":0, "priority":0,
"description": {"option": "TP7"}},
{"task_type":"Delivery", "start_time":0, "priority":0,
"description": {"option": "TP6"}}
]
```

APPENDIX B: Launch files for simulation

- i. Environment.launch.xml file to bring up robot.launch.xml and simulation.launch.xml files.

```
<?xml version='1.0' ?>

<launch>
  <arg name="gazebo_version" default="11"/>
  <arg name="use_sim_time" default="true"/>
  <arg name="failover_mode" default="false"/>

  <!-- Robot launch -->
  <include file="$(find-pkg-share test)/launch/robot.launch.xml">
    <arg name="use_sim_time" value="$(var use_sim_time)"/>
    <arg name="failover_mode" value="$(var failover_mode)"/>
  </include>

  <!-- Simulation launch -->
  <include file="$(find-pkg-share test)/launch/simulation.launch.xml">
    <arg name="map_name" value="Environment" />
    <arg name="gazebo_version" value="$(var gazebo_version)" />
  </include>
</launch>
```

- ii. robot.launch.xml file to bring up tinyRobot_adapter.launch.xml and deliveryRobot_adapter.launch.xml files as well as initialize RViz2 and RMF Panel web platform.

```
<?xml version='1.0' ?>

<launch>
  <arg name="use_sim_time" default="false"/>
  <arg name="failover_mode" default="false"/>

  <!-- Common launch -->
  <include file="$(find-pkg-share test)/launch/common.launch.xml">
    <arg name="use_sim_time" value="$(var use_sim_time)"/>
    <arg name="viz_config_file" value="$(find-pkg-share
test)/Map/Environment.rviz"/>
    <arg name="config_file" value="$(find-pkg-share
test)/Map/Project.building.yaml"/>
    <arg name="dashboard_config_file" value="$(find-pkg-share
test)/Environment/dashboard_config.json"/>
  </include>
```

```

<!-- TinyRobot fleet adapter and robot state aggregator needed for the
TinyRobot slotcar_plugin -->
<group>
  <let name="fleet_name" value="tinyRobot"/>
  <include file="$(find-pkg-share
test)/launch/tinyRobot_adapter.launch.xml">
    <arg name="fleet_name" value="$(var fleet_name)"/>
    <arg name="use_sim_time" value="$(var use_sim_time)"/>
    <arg name="nav_graph_file" value="$(find-pkg-share
test)/Map/nav_graph/0.yaml" />
  </include>
  <include file="$(find-pkg-share
rmf_fleet_adapter)/robot_state_aggregator.launch.xml">
    <arg name="robot_prefix" value="tinyrobot"/>
    <arg name="fleet_name" value="$(var fleet_name)"/>
    <arg name="use_sim_time" value="$(var use_sim_time)"/>
    <arg name="failover_mode" value="$(var failover_mode)"/>
  </include>
</group>

<!-- DeliveryRobot fleet adapter and robot state aggregator needed for
DeliveryRobot slotcar_plugin -->
<group>
  <let name="fleet_name" value="deliveryRobot"/>
  <include file="$(find-pkg-share
test)/launch/deliveryRobot_adapter.launch.xml">
    <arg name="fleet_name" value="$(var fleet_name)"/>
    <arg name="use_sim_time" value="$(var use_sim_time)"/>
    <arg name="nav_graph_file" value="$(find-pkg-share
test)/Map/nav_graph/1.yaml" />
  </include>
  <include file="$(find-pkg-share
rmf_fleet_adapter)/robot_state_aggregator.launch.xml">
    <arg name="robot_prefix" value="deliveryrobot"/>
    <arg name="fleet_name" value="$(var fleet_name)"/>
    <arg name="use_sim_time" value="$(var use_sim_time)"/>
    <arg name="failover_mode" value="$(var failover_mode)"/>
  </include>
</group>

</launch>

```

iii. simulation.launch.xml file to launch the model in Gazebo.

```

<?xml version='1.0' ?>

<launch>
  <arg name="map_package" default="test" description="Name of the map
package" />
  <arg name="map_name" default="Environment" description="Name of the
rmf_demos map to simulate" />
  <arg name="use_crowdsim" default='0' />

  <arg name="gazebo_version" default="11" />
  <let name="world_path" value="$(find-pkg-share $(var
map_package))/Map/$(var map_name).world" />
  <let name="model_path" value="$(find-pkg-share $(var
map_package))/Map/models:$(find-pkg-share
rmf_demos_assets)/models:/usr/share/gazebo-$(var gazebo_version)/models" />
  <let name="resource_path" value="$(find-pkg-share
rmf_demos_assets)/usr/share/gazebo-$(var gazebo_version)" />
  <let name="plugin_path" value="$(find-pkg-prefix
rmf_robot_sim_gazebo_plugins)/lib/rmf_robot_sim_gazebo_plugins:$(find-pkg-
prefix
rmf_building_sim_gazebo_plugins)/lib/rmf_building_sim_gazebo_plugins:/usr/s
hare/gazebo-$(var gazebo_version)" />

  <!-- Use crowd sim if `use_crowdsim` is true-->
  <let name="menge_resource_path" if="$(var use_crowdsim)" value="$(find-
pkg-share $(var map_package))/Map/config_resource"/>
  <let name="menge_resource_path" unless="$(var use_crowdsim)" value=""
/>

  <executable cmd="gzserver --verbose -s libgazebo_ros_factory.so -s
libgazebo_ros_init.so $(var world_path)" output="both">
    <env name="GAZEBO_MODEL_PATH" value="$(var model_path)" />
    <env name="GAZEBO_RESOURCE_PATH" value="$(var resource_path)" />
    <env name="GAZEBO_PLUGIN_PATH" value="$(var plugin_path)" />
    <env name="GAZEBO_MODEL_DATABASE_URI" value="" />
    <env name="MENGE_RESOURCE_PATH" value="$(var menge_resource_path)"/>
  </executable>
  <group unless="$(var headless)">
    <executable cmd="gzclient --verbose $(var world_path)" output="both">
      <env name="GAZEBO_MODEL_PATH" value="$(var model_path)" />
      <env name="GAZEBO_RESOURCE_PATH" value="$(var resource_path)" />
      <env name="GAZEBO_PLUGIN_PATH" value="$(var plugin_path)" />
    </executable>
  </group>

</launch>

```

iv. `tinyRobot_adapter.launch.xml` file to initialize tiny robot fleet adapter and set physical parameters for tiny robots.

```
<?xml version='1.0' ?>

<launch>

  <arg name="fleet_name" default="tinyRobot" description="Name of this
fleet of tinyRobot robots"/>
  <arg name="use_sim_time" default="false" description="Use the /clock
topic for time to sync with simulation"/>
  <arg name="nav_graph_file" description="Nav graph required by fleet
adapter"/>

  <group>
    <include file="$(find-pkg-share
rmf_fleet_adapter)/fleet_adapter.launch.xml">

      <!-- The name and control type of the fleet -->
      <arg name="fleet_name" value="$(var fleet_name)"/>
      <arg name="control_type" value="full_control"/>

      <!-- The graph that this fleet should use for navigation -->
      <arg name="nav_graph_file" value="$(var nav_graph_file)" />

      <!-- The nominal linear and angular velocity of this fleet's vehicles
-->
      <arg name="linear_velocity" value="0.5"/>
      <arg name="angular_velocity" value="0.6"/>

      <!-- The nominal linear and angular acceleration of this fleet's
vehicles -->
      <arg name="linear_acceleration" value="0.5"/>
      <arg name="angular_acceleration" value="2.0"/>

      <!-- The radius of the circular footprint of this fleet's vehicles -
->
      <arg name="footprint_radius" value="0.3"/>
      <!-- Other robots are not allowed within this radius -->
      <arg name="vicinity_radius" value="0.5"/>

      <!-- Whether to use sim time -->
      <arg name="use_sim_time" value="$(var use_sim_time)"/>
    </include>
  </group>
</launch>
```



```
<!-- How long it can be delayed before we give up and start over -->
<arg name="delay_threshold" value="15.0"/>

<!-- Don't make the tinyRobot wait long to retry -->
<arg name="retry_wait" value="10.0"/>

<!-- Give everything time to discover -->
<arg name="discovery_timeout" value="60.0"/>

<!-- Can the robot drive backwards -->
<arg name="reversible" value="true"/>

<!-- Whether it can perform deliveries -->
<arg name="perform_deliveries" value="true"/>
<!-- Whether it can perform loop -->
<arg name="perform_loop" value="false"/>
<!-- Whether it can perform cleaning -->
<arg name="perform_cleaning" value="false"/>
<!-- What the robot should do once it finishes its tasks -->
<arg name="finishing_request" value="charge"/>

<!-- TODO Update these values with actual specs -->
<!-- Battery parameters -->
<arg name="battery_voltage" value="12.0"/>
<arg name="battery_capacity" value="24.0"/>
<arg name="battery_charging_current" value="5.0"/>

<!-- Physical parameters -->
<arg name="mass" value="20.0"/>
<arg name="inertia" value="10.0"/>
<arg name="friction_coefficient" value="0.22"/>

<!-- Power systems -->
<arg name="ambient_power_drain" value="20.0"/>
<arg name="tool_power_drain" value="0.0"/>

<!-- Whether to consider battery drain for task planning -->
<arg name="drain_battery" value="true"/>

<!-- Battery level at which the robot ceases to operate -->
<arg name="recharge_threshold" value="0.2"/>

</include>
</group>
</launch>
```

v. `deliveryRobot_adapter.launch.xml` file to initialize delivery robot fleet adapter and set physical parameters for delivery robots.

```
<?xml version='1.0' ?>

<launch>

  <arg name="fleet_name" default="deliveryRobot" description="Name of this
fleet of deliveryRobot robots"/>
  <arg name="use_sim_time" default="false" description="Use the /clock
topic for time to sync with simulation"/>
  <arg name="nav_graph_file" description="Nav graph required by fleet
adapter"/>

  <group>
    <include file="$(find-pkg-share
rmf_fleet_adapter)/fleet_adapter.launch.xml">

      <!-- The name and control type of the fleet -->
      <arg name="fleet_name" value="$(var fleet_name)"/>
      <arg name="control_type" value="full_control"/>

      <!-- The graph that this fleet should use for navigation -->
      <arg name="nav_graph_file" value="$(var nav_graph_file)" />

      <!-- The nominal linear and angular velocity of this fleet's vehicles
-->
      <arg name="linear_velocity" value="0.5"/>
      <arg name="angular_velocity" value="0.6"/>

      <!-- The nominal linear and angular acceleration of this fleet's
vehicles -->
      <arg name="linear_acceleration" value="0.75"/>
      <arg name="angular_acceleration" value="2.0"/>

      <!-- The radius of the circular footprint of this fleet's vehicles -
->
      <arg name="footprint_radius" value="0.6"/>
      <!-- Other robots are not allowed within this radius -->
      <arg name="vicinity_radius" value="0.8"/>

      <!-- Whether to use sim time -->
      <arg name="use_sim_time" value="$(var use_sim_time)"/>
    </include>
  </group>
</launch>
```

```
<!-- How long it can be delayed before we give up and start over -->
<arg name="delay_threshold" value="15.0"/>

<!-- Don't make the deliveryRobot wait long to retry -->
<arg name="retry_wait" value="10.0"/>

<!-- Give everything time to discover -->
<arg name="discovery_timeout" value="60.0"/>

<!-- Can the robot drive backwards -->
<arg name="reversible" value="true"/>

<!-- Whether it can perform deliveries -->
<arg name="perform_deliveries" value="true"/>
<!-- Whether it can perform loop -->
<arg name="perform_loop" value="false"/>
<!-- Whether it can perform cleaning -->
<arg name="perform_cleaning" value="false"/>
<!-- What the robot should do once it finishes its tasks -->
<arg name="finishing_request" value="charge"/>

<!-- TODO Update these values with actual specs -->
<!-- Battery parameters -->
<arg name="battery_voltage" value="24.0"/>
<arg name="battery_capacity" value="40.0"/>
<arg name="battery_charging_current" value="8.8"/>

<!-- Physical parameters -->
<arg name="mass" value="20.0"/>
<arg name="inertia" value="10.0"/>
<arg name="friction_coefficient" value="0.22"/>

<!-- Power systems -->
<arg name="ambient_power_drain" value="20.0"/>
<arg name="tool_power_drain" value="0.0"/>

<!-- Whether to consider battery drain for task planning -->
<arg name="drain_battery" value="true"/>

<!-- Battery level at which the robot ceases to operate -->
<arg name="recharge_threshold" value="0.2"/>

</include>
</group>
</launch>
```

APPENDIX C: Part 1 Gantt Chart

Project Activities	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14
FYP title registration, understanding and planning on the project title, gathering information for preparation of project start up.														
Construction of literature review of the research topic with collected information and data														
Data Analysis and propose methodology for the research topic														
Familiarize with ROS2 and other simulation tools														
Progress report finalization and oral presentation														

APPENDIX D: Part 2 Gantt Chart

Project Activities	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14
Modeling of simulated manufacturing environment in Gazebo with detail location of robots and task points	■	■	■	■										
Development of task distribution algorithm for heterogeneous multi-robot system			■	■	■	■	■	■	■					
Implement motion planning of robots in simulated environment						■	■	■	■	■	■			
Evaluate and testing of developed algorithm in Gazebo							■	■	■	■	■	■	■	
Final report writing, poster design, and oral presentation												■	■	■