An Autonomous Multiple Robot Registration and Control System: Design Implementation and Performance Evaluation

Udugoda Uduporawage Samantha Kumara Rajapaksha

Reg.No: DP17910378

Doctor of Philosophy (Ph.D.)

Department of Information Technology

Sri Lanka Institute of Information Technology

November, 2022

DECLARATION

I hereby declare that the thesis entitled “An Autonomous Multiple Robot Registration and Control System: Design, Implementation and Performance Evaluation" that is completed and submitted by me for the award of the degree of *Doctor of Philosophy* to Sri Lanka Institute of Information Technology(SLIIT) is a record of work carried out by me under the supervision of Prof.Chandimal Jayawardena, Dean, Faculty of Computing, SLIIT and External -supervision of Prof.Bruce MacDonald, Department of
Electrical, Computer and Software Engineering, Faculty of Engineering in University Of Auckland, New Zealand.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Certified by:

Name: U.U.Samantha Kumara Rajapaksha
Signature: [illegible]
Date:24th November 2022

Name of the Supervisor: Prof.Chandimal Jayawardena
Signature:.........................
Date:................................25th November 2022

Name of the External Supervisor: Prof.Bruce MacDonald
Signature:.........................
Date:................................25 November 2022

ABSTRACT

ROS is the most prominent middleware used by most researchers in robotic application development. Our research mainly depends on ROS technologies because most researchers currently work with ROS as middleware for many research projects. Controlling the robots through the Web interface is essential. Because in some instances, users may not be able to communicate with the robot directly because of some bad conditions in the environment where the robots are currently placed. Therefore, we have developed a Web interface to control all robots through the Internet. However, the ROS topics, nodes, and message formats used to subscribe and publish can differ from one robot to another when we work with multiple robots in the same environment. Therefore, when a user expresses high-level instructions through a Web interface, all multiple robots must understand instructions uniformly and take necessary actions accordingly without considering each robot's internal software and hardware implementation. The first contribution of the research is to develop an algorithm to register all robots based on the main components of the ROS technology through the Web interface autonomously. The robot Registration Engine was developed with algorithms to complete the autonomous robot registration task. The second contribution is identifying the relevant ROS topics and nodes for each action when a user command gives through the Web interface. The ROS topic identification algorithm was developed successfully. The third contribution was to evaluate the system performance under different conditions and derive the equations for the delay in response time through the web interface, validating the equations derived.

We have conducted several experiments to evaluate our system with delays in response time. The worst-case analysis was completed for all algorithms with Big O notation. Users and researchers can utilize Robot Registration Algorithm and ROS Topic Identification Algorithm to work with multiple robots through the Web interface. We have successfully implemented all algorithms in a simulated environment in Gazebo.

Keywords: *Multiple robot*, *Ontology*, *Robot Operating System*, *Navigation*, *Gazebo*, *Big O notation*,*Simulation*,*TurtleBot*,*Husky*,*TiaGo*.

ACKNOWLEDGEMENT

With immense pleasure and a deep sense of gratitude, I wish to express my sincere thanks to my supervisor Prof.Chandimal Jayawardena, Dean, Faculty of Computing, SLIIT, without his motivation and continuous encouragement, this research would not have been successfully completed. I have benefited greatly from your wealth of knowledge and meticulous editing. I am extremely grateful that you took me on as a student and continued to have faith in me over the years.

Thank you to my external supervisor, Prof.Bruce MacDonald, Department of Electrical, Computer and Software Engineering, Faculty of Engineering in the University of Auckland, New Zealand, for your patience, guidance, and support. I have benefited greatly from your wealth of knowledge and meticulous editing. I am extremely grateful that you took me on as a student and continued to have faith in me over the years.

I am extremely thankful to Prof.Lakshman Ratnayaka the Chancellor of SLIIT, Prof.Lalith Gamage , the Vice Chancellor of SLIIT and Prof.Nimal Rajapake , Deputy Vice Canceller of SLIIT for supporting me to carry out research in the SLIIT and also for providing me with infrastructural facilities and many other resources needed for my research.

I express my sincere thanks to Prof.Rahula Attalage, Dean, Faculty of Graduate Studies and Research, SLIIT for his kind words of support and encouragement. I like to acknowledge the support and encouragement rendered by Dr.Anuraha Jayakody, Mr.Prasanna Sumathipala and Mr.Jagath Wickramarathna in several ways throughout my research work.

I wish to extend my profound sense of gratitude to my parents for all the sacrifices they made during my research and for providing me with moral support and encouragement whenever required.

Last but not least, I would like to take the opportunity to thank my wife Kaushalya Dilrukshi and my daughter Sahansa Yenuli Nisadara Rajapaksha for their constant encouragement and moral support along with patience and understanding.

Place: SLIIT Date: 14/07/2022 U.U.Samantha Rajapaksha

LIST OF PUBLICATIONS

1. U. U. Samantha Rajapaksha, Chandimal Jayawardena, Bruce A. MacDonald,"Design, Implementation, and Performance Evaluation of a Web-Based Multiple Robot Control System", Journal of Robotics, vol. 2022, Article ID 9289625, 24 pages, 2022. https://doi.org/10.1155/2022/9289625

2. U. U. S. K. Rajapaksha, C. Jayawardena and B. A. MacDonald, "ROS BasedHeterogeneous Multiple Robots Control Using High Level User Instructions,"
   TENCON 2021 - 2021 IEEE Region 10 Conference (TENCON), 2021, pp. 163168, doi: 10.1109/TENCON54134.2021.9707460.

3. U. U. Samantha Rajapaksha, C. Jayawardena and B. A. MacDonald, "ROS BasedMultiple Service Robots Control and Communication with High Level User Instruction with Ontology," 2021 10th International Conference on Information and Automation for Sustainability (ICIAfS), 2021, pp. 381-386, doi: 10.1109/ICIAfS52090.2021.9606062.

4. U. U. S. Rajapaksha, C. Jayawardena and B. A. MacDonald, "ROS SupportedHeterogeneous Multiple Robots Registration and Communication with User Instructions," 2022 2nd International Conference on Advanced Research in Computing (ICARC), 2022, pp. 102-107, doi: 10.1109/ICARC54489.2022.9753837.

5. U. U. S. Rajapaksha and C. Jayawardena, "Ontology based Optimized Algorithms to Communicate with a Service Robot using a User Command with Unknown Terms," IEEE 2nd International Conference on Advancements in Computing (ICAC), 2020, pp. 258-262, doi: 10.1109/ICAC51239.2020.9357254.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTOFTERMSANDABBREVIATIONS

R_{delta} Time taken to retrieve the state from ROS topic xii

$R_{m,d}^{move}$ Delay in moving to specific location by Multiple Robots xii

$R_{s,d}^{move}$ Delay in moving to specific location by Single Robot xii

$R_{s,d}^{start}$ Single robot delay Time at start xii

$R_{s,d}^{start}$ Single robot delay Time at stop xii

S_{delta} Time taken to save the state in ROS topic xii

U_x^s Linear Speed of the Robot in x direction at the start in ms^{-1}. xii

U_x^s $tau_{d,ROS}$ Delay in communicating with ROS topics xii

U_x^s $tau_{d,RT}$ Delay in ROS topic identification xii

U_x^s $tau_{d,os}$ Delay in system call execution in Operating System. xii

$tau_{d,pos}$ Delay in getting the current position and orientation of the robot xii

U_x^s $tau_{d,web}$ Delay in communication through Web interface. xii

w_z^s Angular Speed of the Robot in z direction at the start in ms^{-1}. xii

w_z^s AI Artificial Intelligence xii

w_z^s AMCL Adaptive Monte Carlo Localization xii

API Application Programming Interface xii

CPE Command Publishing Engine xii

GUI Graphical User Interface xii

HMR Heterogeneous Multiple Robots xii

HTML Hyper Text Markup Language xii

IOT Internet Of Things xii

IP Internet Protocol xii

CHAPTER 1

Introduction

1.1 Introduction

Robotic programming is a complex and time-consuming task with most available middleware. ROS is popular open-source software that most researchers use to complete robotic programming tasks. It is not an operating system but a framework for developing the software with robots. ROS provides the hardware abstraction, low-level control of the devices, provides commonly-used functionalities and message passing. Our research work mainly works with the ROS environment. ROS provides a better interface to communicate and control the robots. However, it is not easy to work with ROS in robot programming. One of the main objectives of the research is to provide a Web interface to control the robot easily without considering the low-level architecture of the ROS and robot.

Human-robot communication is one of the top research areas in robotic research work. Different types of middleware are available to make this communication easy and user-friendly. However, there are several challenges in developing a robotic application for robot communication using the middleware directly because of the complexity and interoperability of the middleware. The ROS is one of the middleware used to program most robotic applications.

In an intelligent environment, there may be lots of service robots which are working at different places simultaneously. Therefore, they may have different capabilities and operate with different control systems (mobile base, wheels, two legs, four legs, tracks etc.). Moreover, most robots may have different software specifications, even running with the same middleware like ROS.

Suppose a system can hide these difficulties by providing a straightforward interface where the user can provide high-level instruction without considering any low-level technologies to communicate with a multi-robot environment. In that case, it will be very convenient for the users.

Autonomous robot registration and control is one of the complex tasks in robotic application development. ROS was developed to improve interoperability and reduce heterogeneous multiple robot programming complexities. ROS is a kind of middleware used by developers in robotic applications to reuse most existing software developed by different researchers. There are different nodes, topics, and message formats for different robots in ROS. Another project objective is registering the robot autonomously through the web interface. An algorithm was developed to find the related topics to control different robots in ROS. Therefore, our system's main component is the Robot Registration Engine (RRE) which is developed to register multiple heterogeneous robots by getting all related *rostopics*. The web interface was developed to interact with robots and users using the ROS bridge server. ROS bridge server worked as an interface between the ROS environment and Web interface. We have developed different web interfaces to interact with the user and different types of experiments in our research as described by Web Interface I to V. the following table 1.1.

Table 1.1 Types of Web Interfaces

Web Interface Types	Description of the Web Interface
Web Interface I	Single Robot without Autonomous Robot Registration
Web Interface II	Single Robot with Autonomous Robot Registration
Web Interface III	Multiple Two Robots with Autonomous Robot Registration
Web Interface IV	Multiple Four Robots with Autonomous Robot Registration
Web Interface V	Heterogeneous Robot with Autonomous Robot Registration with Semantic Instructions

Web Interface I to IV was developed to work with instructions like moving the robot to a specific location and working with multiple instructions sequentially. Web Interface V was developed to work with instructions. We have used the Gazebo simulator for our experiments. The robot's actions and initial position changed with time. Therefore, we have created a schedule for each robot to complete movement or navigation in the experiment with Web Interface V. Then, we have identified the relevant ROS topic in corresponding nodes to subscribe and publish the corresponding command values from the user command. The Command Publishing Engine (CPE) is responsible for publishing the ROS command for each action defined in the given user-level instruction.

Different architectures were used to design the heterogeneous multiple robot system, including centralized, distributed, and hybrid modes Hu et al. (2015). Our solution is based on the centralized server architecture shown in Figure 1.1.

Before publishing the command on each robot, we need to identify the ROS topics to publish or subscribe to and nodes which contain the ROS topics for each robot. There-

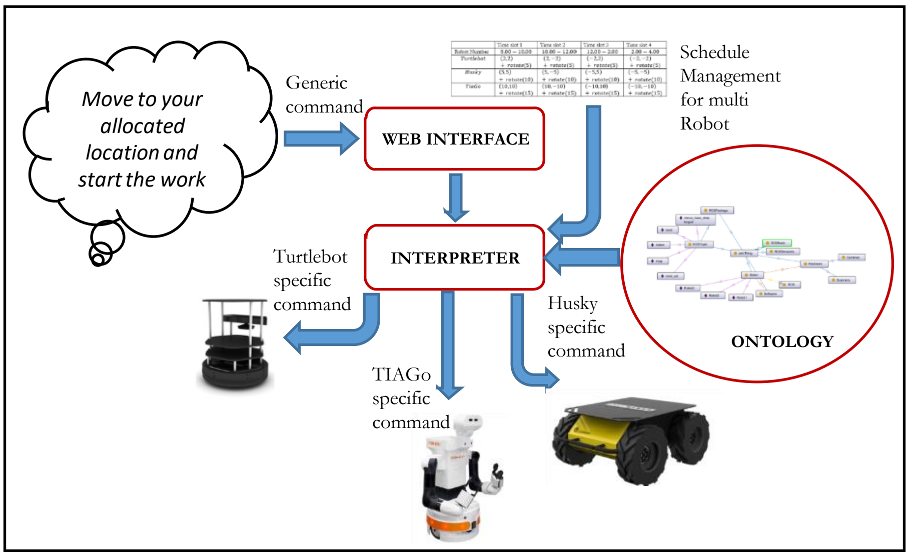

Move to your allocated location and start the work
Generic command
WEB INTERFACE
Schedule Management for multi Robot
INTERPRETER
ONTOLOGY
Turtlebot specific command
TIAGo specific command
Husky specific command

Fig. 1.1 High Level System Diagram

fore, in our solution, the one task is to automatically register the robot in Robot Registration Engine (RRE) by collecting all software-related specifications using rosnode and rostopic commands.

1.2 Problem Statements

A relatively new research initiative in autonomous robot control is to develop autonomous robot registration and control all robots through the Web interface. However, based on the analysis of current research studies, we could not find research work on autonomous robot registration and control through the web interface. Furthermore, we have identified that programming a robot and multiple robots are too complex and tedious, even with the ROS middleware.

There are different ROS topics and ROS message formats for robots in ROS. Therefore, we must identify the relevant ROS topics and nodes to publish or subscribe to different robots in ROS. Furthermore, autonomous Robot Registration and the control command issue with the corresponding ROS topic is another main problem in the research. Another research problem is managing and controlling multiple robots through a web interface.

According to our background studies, we did not find any research works to identify the ROS topic and nodes for controlling multiple robots simultaneously and autonomously. Therefore, the other main research problem is identifying the relevant ROS topics and nodes for the given user instruction.

There are different types of multiple heterogeneous robots with different capabilities. The most prominent parts of the ROS are ROS nodes, ROS topics, message formats and ROS services. There are different ROS topics and ROS message formats for different robots in ROS. We must identify the relevant ROS topics and nodes to publish or subscribe to different robots in ROS. Autonomous control and communication of the multiple robots through the web interface is one of the challenging tasks. Autonomous Robot Registration and the control command issue with corresponding ROS topic is another main problem in the research. Working with ROS is also another very complicated and tedious task.

There were several research to get the current position and orientation of the robots. With some conversion algorithms, finding the current position and orientation can be achieved with odometry reading. Moving all robots to the given location is another research problem we selected to solve.

Algorithms play a significant role in our system. It defines the steps that must be completed sequentially to convert the input into the desired output. Performance analysis of the algorithms is not a simple task when the algorithm is complex. There are several algorithm analysis techniques. Big O notation is one of the best ways to describe the complexity of the algorithms. Another research problem is analyzing the algorithms' time complexity using Big O notation.

Performance analysis with different web interfaces is another main research challenge. We need to derive the equations for performance analysis for each experiment with different web interfaces with different scenarios. Deriving the mathematical equations with performance analysis is another research problem that we need to solve.

1.3 Thesis Objectives

The main objective is to develop an algorithm to interact and control multiple robots through the web interface with autonomous robot registration and autonomous ROS topic identification.

Many research groups completed robot control and communication with ROS, but according to our research studies, we did not find any research for autonomous robot registration using any algorithms. ROS topics and nodes are critical components in the ROS environment. The robots may have different ROS topics and nodes for subscribing and publishing. One of the leading research project objectives is to develop an algorithm that can register all robots concurrently through the Web interface.

When a user issues an instruction, then our system must be able to find the corresponding ROS topis and nodes to complete the assigned task in the user command. According to the previous studies, we did not find any algorithm developed to identify the relevant ROS topics and nodes for subscription and publication. Therefore, one of the leading research project objectives is to develop an algorithm that can identify the relevant ROS topics and nodes to complete the issued task by the user.

Our system is simulated with the Gazebo environment with multiple robots. However, managing multiple robots with a Gazebo environment through a web interface is not an easy task. Most of the time, managing the errors with ROS and Gazebo is very tedious and time-consuming because fewer resources are available online for ROS and Gazebo. Therefore, one of the leading research project objectives is to learn the ROS and gazebo environment from scratch and simulate the environment for the experiments. Designing an algorithm is not an easy task because we need to consider several factors in designing a good algorithm. Then the algorithm analysis is fundamental to finding the performance of each algorithm. There are several algorithm analysis techniques. Big O notation is the optimal way to represent the algorithm's complexity. One of the leading research project objectives is to develop an optimal algorithm to get the correct output and analyze the performance of the algorithms.

Performance evaluation with the response time is another research problem that we want to solve. Several experiments must be completed with different web interfaces with different amounts of robots for different scenarios. We must design the experiment environments and evaluate the performance for response time. Again, we need to derive the mathematical equations representing each scenario's delay in response time. Therefore, another main objective is to perform analysis with derived mathematical equations for each scenario with different web interfaces.

1.4 Research Approach

As stated in the previous section, The main objective is to develop an algorithm to interact and control multiple robots through the web interface with autonomous robot registration and autonomous ROS topic identification. However, some factors must be considered when approaching the research work as described below.

ROS programming: Evan, there are many research works conducted with middleware as ROS by many researchers. However, working with ROS to develop a simple application is a challenging and tedious. Because getting online help and online resources for developing the application was very limited. Therefore, our initial approach is to learn the ROS from the basics using all available resources online and offline. There are several research papers that describe the way authors have used ROS for the development of some applications. The other approach the study the ROS in detail is to study more research papers in relation to application development with the ROS.

Gazebo environment: Our research work is simulated with multiple robots in a Gazebo environment. Developing an application with a gazebo is also not an easy task again because of fewer online and offline resources available. Therefore, our following approach is to learn the programming gazebo environment by spawning multiple robots simultaneously. Towards this completion, we need to study the research papers and available online materials to get more knowledge on Gazebo.

Design algorithms: Designing an optimized algorithm is a challenging task in the algorithm development process. There were no algorithms developed for autonomous robot registration, autonomous ROS topic and node identification with multiple robots through the web interface. The research approach to complete this task is to study algorithm design techniques and study more research papers to get more knowledge. The analysis of algorithms is another challenging task since there are many ways to identify the complexity of the algorithms. Therefore, we have designed our own algorithm and analysis of the algorithms.

Web Development: Since we need to control all robots through the Internet, we need to have good knowledge of web development and programming. Most of the languages link HTML, script language and extreme programming are essential in this research. Our approach is to study both client-side and server-side programming and ROS support with a web interface.

Programming languages: Most of the ROS programming can be completed with the Python language. Therefore, studying the python language is very important to develop some applications in the ROS library. Therefore, we need to have an excellent approach to develop a comprehensive application with ROS library using the python language. Python supports the concurrent program execution using threads.

Ontology: Ontology development is completed as additional work for the research work. Adding synonyms and semantics is another additional task that we have selected to complete. There are several research works completed with the ontology in different applications. Our research approach is to study the existing research papers and ontologies to support our system.

1.5 Contributions

There are several existing research works where researchers have developed control and manage the robots through the Web interface. There are some algorithms developed to control the multiple robots in the ROS environment and with other middleware. In my research initially, I need to study all existing systems and contributions to control multiple robots through a Web interface. Therefore, one of the main contributions is to identify the limitation and issues with the current and previous research works by thoroughly studying all existing research papers. These contributions can be used to find an optimal solution to solve my research problem.

ROS is the leading middleware that we are using to complete the research project with multiple robots. Initially, the system must be able to register all robots with Robot Registration Engine using the algorithms developed. Getting ROS topics and ROS nodes details is very important to control each robot with a user instruction. One of the leading research contributions of this research project is to control and manage all robots through the Web interface with user instructions without considering all software and hardware differences of all robots. This contribution provides a way to develop an interface that can be used to control multiple robots very easily.

Evaluation of algorithms is a complex task in algorithm designing. Identifying the time complexity of the algorithm is very important when the system is being developed. Here we complete the time complexity analysis for all algorithms developed. One of the leading research contributions of this research project is to provide the complexity analysis of the algorithm to decide the performance of each algorithm.

Performance evaluation in terms of the delay of response for each experiment is another main task of the project. We need to formulate the mathematical equations for the delay in response time for each scenario. These mathematical formulas can be utilized for the prediction of the performance of the system. One of the leading research contributions of this research project is to formulate the mathematical equations for the performance in terms of the delay in response time.

The main contribution to my research work are summarized below: First, I discovered and proposed a new algorithm for autonomous, multiple robot registration in a simulated Gazebo environment. According to the previous research studies described in chapter 02, I did not find any research on autonomous robot registration. Once a robot is connected through the Web interface, all ROS topics and nodes related to each robot are collected and stored to use later. One of the main limitations of the ROS is that robotic programming very difficult and but autonomous registration solve this issue since it collects all ROS topics and nodes necessary to publish and subscribe.

I discovered and proposed a new algorithm for the ROS topic identification when a user issued a command to control robots through the web interface. However, according to the previous research studies described in chapter 02, I did not find any research on ROS topic identification algorithms.

I discovered and developed the web interface to control multiple robots with simple commands to move robots forward and circle simultaneously using the threads in python language. However, according to the previous research studies described in chapter 02, I did not find any research on multiple robot controls through the web interface. Some research was done to control single robots through web interfaces without autonomous registration.

I discovered the worst-case complexity of the autonomous robot registration algorithm and ROS topic identification algorithm. This analysis results can be used by other researchers when they want to get an idea of the algorithm's performance.

I discovered and derived mathematical equations to represent the delay in response time for the different scenarios with all experiments with other characteristics. Furthermore, I found and validated the values for all constants in each mathematical equation.

1.6 Thesis Outline

The following sections are grouped as follows. Section 2 represents a literature survey with background readings and related research works with technologies. The methodology with algorithms and main components of the design are presented in section 3. The experiments and evaluation of the research project with results are described in section 4. Section 5 represets the discussion of the problem, solution and research findings.

Finally, section 5 describes the conclusion with future works.

CHAPTER 2

Literature Review

Several research works are currently working in the area of HMR communication in different research groups. Here we discuss some of the works which are similar to our work. We have categorized all background readings as Heterogeneous multiple robot controls, Ontology-based multiple robot control and Interface and system development for multiple robot control.

2.1 Heterogeneous Multiple Robot Controls

Many research studies were conducted by researchers with heterogeneous multiple robot controls. Here we discussed the related research works.

Some research groups have implemented heterogeneous multi-robot control with the involvement of humans. Seohyun *et al.* proposed a three-layer architecture to control a multi-robot with human intervention. They have separated the autonomous and manual parts in the interface design to control the multi robots (Jeon et al. 2012). M. Alberri *et al.* have developed ROS-based architecture to connect multi-robot heterogeneous systems with a hierarchical system. They have proposed layered architecture. The high and middle layers consist of several ROS packages with ROS nodes to provide different functionalities. The lowest layer consists of some C and C++ software packages (Alberri et al. 2018).

Another research group has developed a hybrid architecture based on ROS. This system integrates the personal computer(PC) and embedded systems with multiple heterogeneous robots. The PC is a server, and the robot is a node(Hu et al. 2015). Y. Msala *et al.* have developed a centralized architecture mainly based on cloud-distributed architecture. It controls and coordinates heterogeneous multiple robots. Furthermore, they have used an artificial intelligence-based algorithm to allocate the task to heterogeneous robots based on the robot's ability (Msala et al. 2019).

Another research group developed a new system which controls robots through a cloud called IAPcloud where most of the CPU-intensive work of the heterogeneous multi-robot can be uploaded to the remote cloud server. Song Z. *et al.* has proposed a cloud-based architecture to collaborate and control the heterogeneous robots by reducing the programming difficulties and development timing (Zheng et al. 2018).

L.F.Costa *et al.* have developed a multi-robot communication web-based interface in the same environment using ROS. They have developed two services on the webserver to monitor and control. The main operation developed in the system is moving the robot forward, to the right, to left and backwards. They have implemented three layers: robot, server, and client. The main development is to provide the environment to use the different robots by lay people with ROS-based implementations(Costa and Gonc¸alves 2016).

R.Han *et al.* have developed a system for multi-robot navigation in dynamic environments where they have used deep reinforcement learning to find the optimal path (Han et al. 2020).

N. Lashkari *et al.* have developed a novel robust control method for heterogeneous multiple robots with autonomous docking and formation. They have considered the limitation in existing formation methods like battery failure, limited transportation capacity, and manoeuvrability in developing the new model. The main goal of the developed control is autonomous docking, formation keeping/switching, and collision avoidance in dynamic environments. They conducted the experiments with a simulated virtual environment with V-Rep. A mathematical model was used to develop the formation methods of multiple robots and autonomous docking. They have shown that the followers can dock themselves to other followers in the system and then maintain the formation as a docked system (Lashkari et al. 2020).

R.Yara *et al.* have surveyed multiple heterogeneous robots. They have studied more in task allocation, task decomposition, perception and control of heterogeneous robots. The main challenges were identified and discussed in the paper. Cloud service access with Big data is one of the research issues they have identified. Security and communication are other research problems with IoT-based robotic applications. Human in the loop is one of the problem identified by researchers. Finally, communication constraints and uncertain connectivity are the main research issues that must be solved in future research works (Rizk et al. 2019).

T. Kato *et al.* have developed a formation method for multiple heterogeneous robots based on the current position of the robots. Each robot’s current position and triangular method form the groups. They used simulation methods to verify that the formation was accurate. They used the measurement system using wireless communication and ultrasonic sound. Different shapes were developed using equilateral triangles. They plan to implement this in real robots as future works (Kato et al. 2010).

B.Jungyun *et al.* have developed an Efficient Coordination of Multiple Heterogeneous Mobile Robots Considering Workload Balance. They have solved the Multiple Depot Heterogeneous Traveling Salesman Problem. They used a heuristic approach based on a primal-dual technique to solve this problem and minimize the time. They proved the algorithms that solve the problem of visiting each given target by one robot and completing the given goal by all robots with minimum time (Martinez et al. 2015).

M. Lomas *et al.* have developed an architecture to control and manage multiple heterogeneous robots with a team of operators. They experimented with multiple unmanned vehicles with teams of operators. Open API is used to develop the system with three-tier architecture. It is very flexible to work with the system dynamically assign the task for each robot (Lomas et al. 2011).

I. Tiddi *et al.* have developed an ontology-based robotic application development environment for non-expert users to develop and integrate robotic applications. They have mainly considered simplifying the time-consuming process of programming robotspecific tasks. The ontological representation was used to provide interoperability, meaning for the concepts and relationships by hiding the complexity of the given domain in robotics (Tiddi et al. 2017).

Mihai P. *et al.* have developed a system to parse natural language instruction and get the *"semantic specification"* (semspec) which can be translated to a program to execute on a simulated robot. The system tarn slates the natural language sentences with semantics into a program that can be executed on a robot using interpretation rules with semantic description(Pomarlan and Bateman 2018).

Rajapaksha *et al.* have developed a system which takes input instruction with uncertain words for a drone and converts it to machine-understandable format using the ontology (Rajapaksha et al. 2019).

V. Muthugala *et al.* have completed a review of uncertain information with natural language instructions with service robots. In addition, they have investigated and identified the limitations in existing research work for handling qualitative information in user instruction(Muthugala and Jayasekara 2018).

S.K.Rhee *et al.* have developed an Ontology-based Context and Preference Model for service robots. The automation process is achieved using ontology. They have used rule-based reasoning to work with the context in the environment. They have limited the current implementation with the location, nearby users and objects for the context. However, they plan to expand it with autonomous learning and consider other factors in working with the semantic (Rhee et al. 2012).

2.2 Interface and system development for multiple robot control

F. Mullers¨ *et al.* have developed a tool to create and edit the ROS launch files with the graphical user interface. Furthermore, users can drag and drop the new nodes to the graph and update the resulting launch files with the developed user interface(Mullers¨ et al. 2009).

C. J. Sutherland *et al.* have invented the domain-specific language named *RoboLang*, which can use the existing programming tool. Moreover, it can make small changes to the scripts to run on different platforms robot platforms(Sutherland and MacDonald 2019).

Chandimal J. *et al.* have developed a system to develop the robotic software for the given scenario with minimal modification of the program code, which can be completed quickly. The system can change the robotic software very easily and quickly without adding any errors or bugs in changing the behaviour of the robots. They have developed different engines to support the implementation(Jayawardena et al. 2016).

Chandan D. *et al.* have developed an Integrated Development Environment for visual programming by abstract textual domain-specific language. It provides the program development environment to program robotic applications very fast and very simply with the user requirements (Datta et al. 2012).

K.Takaya *et al.* have conducted experiments to present that the system developed with simulation in Gazebo can be executed on a real robot with ROS without changing any lines of code. They have developed 2D and 3D environments and 3D maps for the navigation of mobile robots. The generated map had some noise, but it can be neglected without any changes to the accuracy. The experiments have indicated that the real robot and the simulated robot with the environment worked the same without any differences (Takaya et al. 2016*a*).

S.S.Velamala *et al.* have developed a graphical user interface for robots and autonomous vehicles using ROS and QT tools. They have control of Wave Adaptive Modular Vehicle using the GUI developed. ROS does not provide GUI to control and program the robots easily. All the ROS commands and gnome commands were executed on the developed GUI. Additional components (sensors and actuators) can be added to the developed system easily. They have proved that the other autonomous system also can be controlled with the developed interface (Velamala et al. 2017).

A tool which can be used to present and visualize the ROS data in a Web browser was developed by A. Ivanov *et al.* They have tested the system with the TurtleBot3 in the Gazebo simulator. ROS Web tool was used to connect the ROS and Web Interface using roslibjs. They have controlled and collected ROS data through the Web interface. It is identified that most browsers have supported the implementation (Ivanov et al. 2021). Chandan D. *et al.* developed an Integrated Development Environment for visual programming by abstract textual domain-specific language. It provides the program development environment to program robotic applications very fast and simply with the user requirements Datta et al. (2012/10/29). Chandimal J. *et al.* developed a new concept named a coach-player model to learn from user commandsDatta et al. (2012/10/29)

Adriano S. *et al.* have reviewed motor control theory and sensory feedback applications performed in parallel. Optimal control models were developed to represent the humans' ability to behave optimally after a certain level of training. The advantage of the structural model and Hosman's descriptive model are discussed in this review Scibilia et al. (2022).

Maide Bucolo *et al.* have worked on a complex and imperfect electromechanical structure that can be used as a paradigm for an imperfect system. They have indicated that the electrical and mechanical interactions generate complex patterns because it prevents the system from reaching correct conditions Bucolo et al. (2019). Our solution may not be perfect in terms of performance characteristics.

Abdulmuttalib T Rashid *et al.* have developed a cluster matching algorithm to get the robot's orientation and localization. Each robot could estimate the neighbour robot's relative orientation within its transmission range. It can get the absolute positions and orientations of the team robots without knowing the ID of the other robots Rashid et al. (2015).

Abduladhem A. *et al.* have developed the multi robots navigation model in a dynamic environment named shortest distance. The collision-free trajectory was developed using the current orientation and position of the other robots. This algorithm is based on the concept of reciprocal orientation that guarantees smooth trajectories and collision-free paths Ali et al. (2016).

Buscarino A. *et al.* have proposed a methodology to control a group of robots without central coordination. They have proved that the system performance with noise can be improved by including long-range connections between the robots. They have the model of the network as a dynamical network Buscarino et al. (2006).

M.Zhengguang *et al.* have developed a multi-robot simulator system with the help of ROS and Qt tools. They have indicated the multi-robot system's advantages over a single robot. They have developed a powerful GUI to work with the multi-robot system with the simulation. They used the Gazebo simulator to visualize the 3D view of the robots. They have proved that this simulation was more accurate and accessible than the other methods (Ma et al. 2019).

Javier Ruiz *et al.* have proposed using personal robots as ubiquitous, multimedia, portable, self-personalized, natural and Internet interfaces. A unique robot is a subclass of a mobile service robot designed to work with humans and conduct as partners, providing entertainment and friendly communication interfaces. They have implemented a robust and versatile object recognition system based on the matching between a reference image. Users can interact with the Web and Internet applications using either the touch screen placed in the robot or speech instructions. They proposed future work to communicate successfully in a highly dynamic environment with backgrounds, variable illumination, and high noise levels in the environment Ruiz-del Solar and Ruiz-del Solar (2007).

V.Usha Rani *et al.* developed a web-controlled surveillance system to provide more security in places that humans cannot visit. The robot can be sent to any location where network access is available. The system was cost-effective and very efficient. The system was implemented with the HTML and python languages. The user can view the live stream through the web interface. The research group have developed the obstacle identification algorithm also Rani et al. (2021).

Radim Farana *et al.* designed a web service to control the robot remotely. The lowlevel protocol was developed to control the robot through the internet. This interface was created using Silverlight technology. The robot is connected to a server using a serial port (RS232). The interface was developed to control the robot using web services Richtr and Farana (2011).

Christopher *Reid et al.* have implemented a cloud computing infrastructure for networked heterogeneous robotic systems in an open-source robot operating system (ROS). The Kobuki Turtlebots and LEGO EV3 robots were used for the experiments by connecting to cloud services on the network through the Robot Operating System. The wireless network is used to create a connection to robots. ROS is used to implement in the robot’s local system to schedule data transmissions. ROS local nodes and cloudbased virtual machines were used to implement the proposed approach. The Robot Operating System was implemented to manage the data transmission to minimize the systems’ network load. The research group has indicated that local processing on networked systems contributes to better overall network performance Reid et al. (2017).

Even though there are several research works done on heterogeneous multi-robot systems, our solution is unique because of using ontology to determine the ROS topic for each action.

2.3 Synonym and Semantic of the User Instruction

Several research groups are working on heterogeneous service Robot communication and control with high-level instructions. Here we discussed some of the work that was done similarly to ours.

Jaehong K. et al. have developed a system to interpret commands for intelligent robots using ontology. Lexico semantic pattern matching has been used to retrieve the meaningful keywords from the user command. They have developed the prototype for the interpretation and the system was tested with different user commands. In addition, they created an intermediary language called FURRL (Formalized User Request Representation Language) to represent the sentences in a formal structure Kim et al. (2006).

A Mart´ınez et al. have developed a switch and router configuration with the domain semantics using the Web Ontology Language (OWL). The research group developed an Ontology-Based Information Extraction (OBIE) system from the Command-Line Interface (CLI) of network devices. A learning algorithm that has automated interpretation of CLIs' configuration capabilities in the heterogeneous network was developed. The semantic similarity function is the system's primary task, which helps in mapping between the ontologies Martinez et al. (2015).

Laurent M. et al. have developed a generic architecture which provides a natural language (NL) algorithm for command interpretation. The system mainly depends on the agent's code and its domain ontology. The two main approaches they have considered are the top-down and the bottom-up approaches. Two approaches were combined in the proposed architecture. This system operates with a minimal semantic analysis on the ontology (synonymy) Mazuel and Sabouret (2006).

The ontology-based system was developed to implement the integrated robotic application using ROS middleware by non-expert users by I. Tiddi et.al. The main objective was to reduce users' time to program robot-specific tasks. In addition, ontologybased representation was developed to provide meaningful, conceptual abstractions of complex and detailed domains, which improved interoperabilityTiddi et al. (2017).

Natural language instructions were processed to generate ”semantic specification” (semspec) and translated into the program to be executed by a simulated robot by Mihai P. et.al. A semantic description of a natural language sentence was translated into an executable program for the robot. Interpretation rules to the semantic description were used to translate into executable codePomarlan and Bateman (2018).

Pomarlan et al. have developed a system that takes a semantic specification (semspec) obtained from parsing a natural language instruction and converts it into a program to be run by a (simulated) robot. In addition, the authors have presented a system that converts a semantic description of a natural language sentence into an executable robot program by applying interpretation rules to the semantic description. Interpretation is ruled that convert a short phrase to a longer description of the action it performs.

After analyzing all research studies, I have summarized all research studies based on the number of robots, web interface usage, autonomous registration, running time analysis and ontology usage, as shown in the Table 2.1. According to the analysis, there needs to be more research conducted on autonomous robot registration and control of the robot through the web interface. Therefore, our research mainly focuses on autonomous robot registration and control through the web interface.

2.4 Related Technologies

2.4.1 Robot Operating System(ROS)

It is a middleware which provides the interface for robotic application developers to create sophisticated software quickly and conveniently. ROS provides the tools and libraries that hide the complex hardware implementation for software developers who Table 2.1 Summary of Research Studies and Research Gap

ResearchGroups	SingleRobots	MultipleRobots	WebBasedControl	AutonomousRegistration	SynonymUsage	AutonomousManagement	RunningTimeAnalysis	OntologyUsage
Jeon et al. 2012	yes	yes	No	No	No	Partial	No	No
Alberri et al.2018	yes	yes	Yes	No	No	No	No	No
Hu et al. 2015	Yes	yes	No	No	No	No	No	No
Msala et al. 2019	Yes	yes	Yes	No	No	No	No	No
Zheng et al.2018	Yes	yes	Yes	No	No	No	No	No
Costa et al. 2016	Yes	yes	Yes	No	No	Partial	No	No
R.Han et al. 2020	Yes	yes	No	No	No	No	No	No
Lashkaret al.2020	Yes	yes	No	No	No	Partial	No	No
Rizk et al.2019	Yes	yes	No	No	No	Partial	No	No
Kato et al.2010	Yes	yes	No	No	No	No	No	No
Martinet al.2015	Yes	yes	No	No	No	No	No	No
Lomas et al. 2011	Yes	yes	No	No	No	No	No	No
Tiddi et al. 2017	Yes	No	No	No	No	No	yes	yes
Pomarl et al.2018	Yes	No	No	No	Yes	No	No	No
Rajapaksa al.2019	Yes	yes	No	No	Yes	No	No	yes
Jayasekara al.2018	Yes	No	No	No	Yes	No	No	No
Rhee et al. 2012	Yes	No	No	No	Yes	Partial	No	Yes

Mullers et al.2009	Yes	No	No	No	Yes	No	No	No
Sutheret al.2019	Yes	No	No	No	No	Partial	No	No
Takaya et al.2016	Yes	No	No	No	No	Partial	No	No
Velamala et al.2017	Yes	No	No	No	No	Partial	No	No
Ivanov et al.2021	Yes	No	Yes	No	No	Partial	No	No
Ma et al. 2019	Yes	Yes	No	No	No	Partial	No	No
Rani et al.2021	Yes	Yes	yes	No	No	No	No	No
Martinez et al.2015	Yes	No	No	No	Yes	No	No	Yes
Tiddi et al.2017	Yes	No	No	No	Yes	No	No	Yes
Bateman et al.2018	Yes	No	No	No	Yes	No	No	Yes
Laurent M. et. al.	Yes	No	No	No	Yes	No	No	Yes
Kim et al.2006	Yes	No	No	No	Yes	No	No	Yes

develop robotic applications. It is open-source software to develop the robotic application. ROS provides hardware abstraction, very low-level device controls, inter-communication between processes and packages for managing robots. It is a distributed collection of processes to reuse the software in robotic application development. ROS works with most modern programming languages like C++, Java and Python. The Unix-based platform is used to run the ROS middleware. ROS is also released with a different distribution similar to Linux versions. ROS levels will be grouped as shown in Figure 2.1.

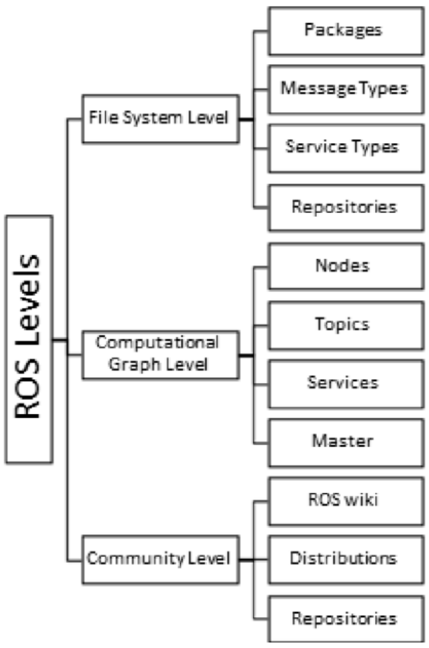


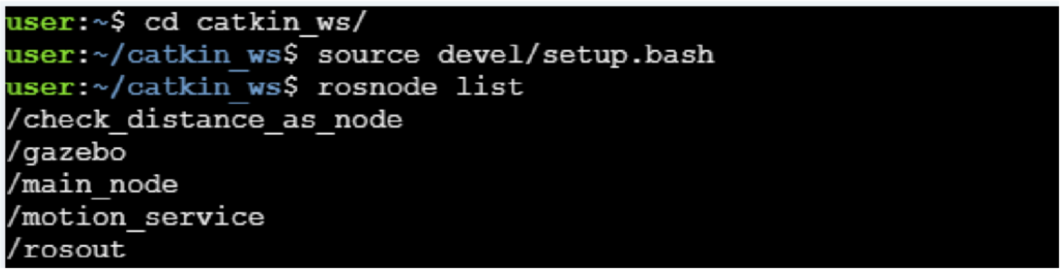
Fig. 2.1 Levels of ROS

ROS packages are used to organize the software as the smallest unit. It works as a software process with runtime library and configuration files. Message types describe the data structures used to send messages in the ROS environment. The service type describes the data structure used in request and response. All standard systems are stored as repositories.

The processes are named nodes that provide different services in the ROS environment. Each node can be assigned for different task in the robotic application. For example, one node can be allocated to navigation, and another node can be allocated for movement control. A node can publish the message on a ROS topic, and another node can subscribe to it. ROS topic is a name which can be used to identify the shared location to communicate with nodes. ROS topic acts like a pipe to communicate with nodes in the ROS environment. Master is working as a registry in the ROS environment through that all communications can be implemented. The ROS master communication is conducted with the XMLRPC, a stateless communication technique.XMLRPC server is running by default with port number 11311. The ROS master is connected with the parameter server to provide the namespaces.

There are several ROS nodes command line tools as described below: *rosnode info* command is Used to print information about a node *rosnode kill* command is Used to kill a running node *rosnode list* command is Used to list active nodes *rosnode machine* command is Used to list nodes running on a particular machine *rosnode ping* command is Used to test connectivity to a node *rosnode cleanup* command is Used to purge registration information of unreachable nodes.

The sample ROS nodes are shown in Figure 2.2 that can be taken by running the *rosnode list* commands. These ROS nodes are similar to processes in the operating system.



```
user:~$ cd catkin_ws/
user:~/catkin_ws$ source devel/setup.bash
user:~/catkin_ws$ rosnode list
/check_distance_as_node
/gazebo
/main_node
/motion_service
/rosout
```

Fig. 2.2 Sample ROS nodes

The ROS wiki provides documentation of the content as a forum. Anyone can sign into this forum and add content. The versions of ROS can be released as distributions similar to the release in Linux operating system. The code segments released by different institutes for the robotic application are stored as repositories. The interaction among ROS nodes, ROS topics and ROS master is shown in Figure 2.3

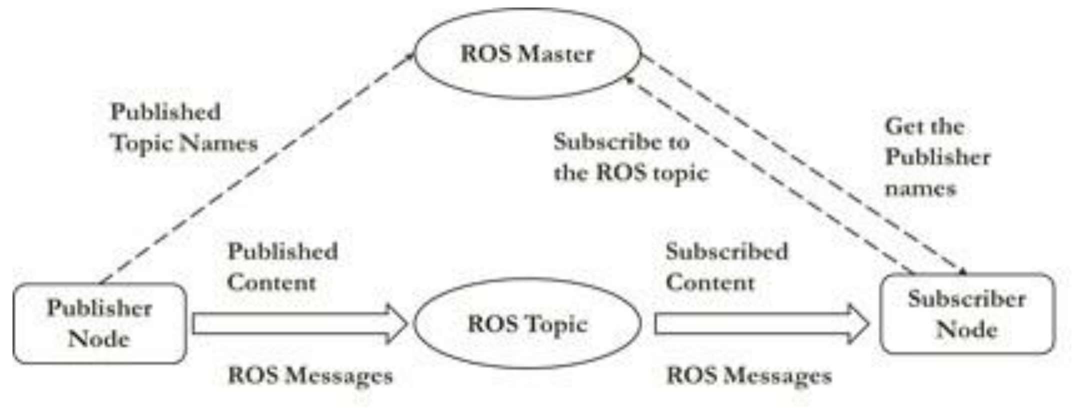


Fig. 2.3 ROS Nodes and Topics

ROS provides a client library for programmers to make applications easily. It mainly supports two languages, C++ and python. *roscpp* is the one library developed for C++ language, and *rospy* is the other library developed for python language.

Our experiments were completed with the ROS Melodic version (Released on 23rd May 2018) with Ubuntu 18.04 version.

2.4.2 ROS Topics

ROS topics are like the standard shared memory in applications where two or more processes can communicate by sharing the content through the shared memory. It is the same as the named bus system on which some messages can be shared. Nodes can publish or subscribe to the selected ROS topic to make communication. There can be many subscribers and publishers for the selected ROS topic. Unidirectional streaming communication is used in communication with the ROS topics. Each topic is strongly typed with a message type to make the communication. The transport type used to communicate in the ROS environment may be TCP/IP-based or UDP based on the application requirements. ROS nodes can negotiate the transport type at run time. It uses the Remote Procedure Call (RPC) to make successful communication.

rostopic is a command-line tool that can work with ROS topics. *rostopic*
list is a command that can get all current ROS topics.

rqt graph is a handy tool to visualize the nodes and topics in the system we are working with ROS. You can use the help option to get all the available sub-commands for the command *rostopic*. rostopic -h rostopic is a command-line tool for printing information about ROS Topics.

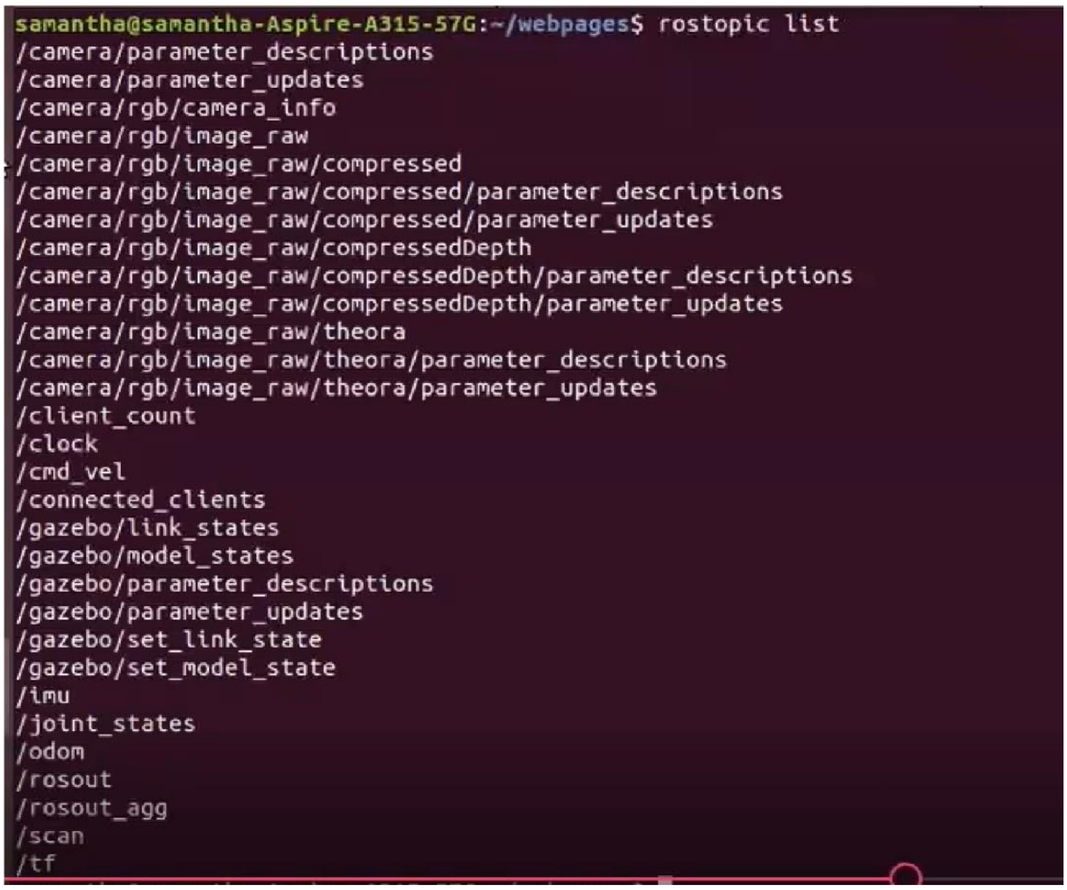
Commands:
rostopic bw command is Used to display bandwidth used by topic
rostopic echo command is Used to print messages to the screen
rostopic find command is Used to find topics by type *rostopic hz*

command is Used to display the publishing rate of topic *rostopic info* command is Used to print information about the active topic *rostopic list* command is Used to list active topics *rostopic pub* command is Used to publish data to a topic *rostopic type* command is Used to print topic type

Different ROS topics can be used to publish commands for different actions. we can use the ROS topics to publish like cmd_vel, cmd vel $\_mux$ or $cmd\_vel\ mux/input/navi$.The possible ROS topics for the movement and initial pose. The sample ROS topic list is shown below in Figure 2.4. It is generated with our experiments with the TurtleBot robot.

2.4.3 Gazebo Simulator

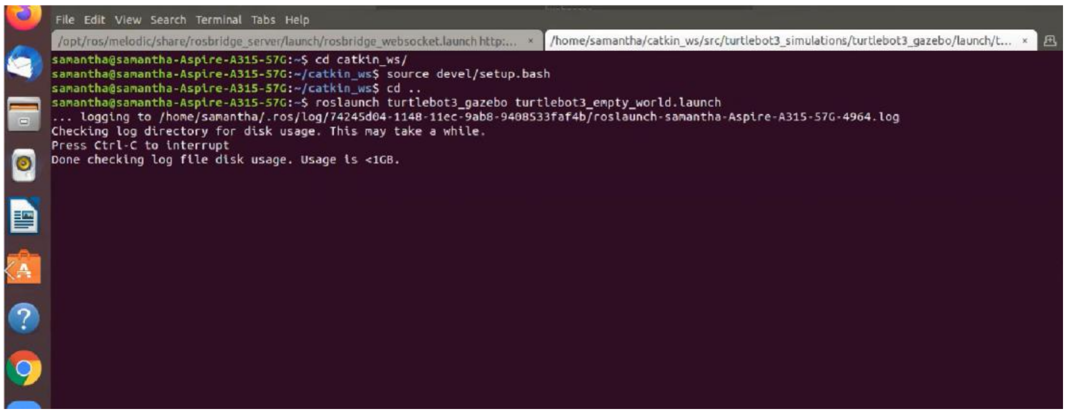
The Gazebo is an open-source simulator for robotic application development. It provides a 3D view for developers. It has a high-performance, open, dynamic engine



```
samantha@samantha-Aspire-A315-57G:~/webpages$ rostopic list
/camera/parameter_descriptions
/camera/parameter_updates
/camera/rgb/camera_info
/camera/rgb/image_raw
/camera/rgb/image_raw/compressed
/camera/rgb/image_raw/compressed/parameter_descriptions
/camera/rgb/image_raw/compressed/parameter_updates
/camera/rgb/image_raw/compressedDepth
/camera/rgb/image_raw/compressedDepth/parameter_descriptions
/camera/rgb/image_raw/compressedDepth/parameter_updates
/camera/rgb/image_raw/theora
/camera/rgb/image_raw/theora/parameter_descriptions
/camera/rgb/image_raw/theora/parameter_updates
/client_count
/clock
/cmd_vel
/connected_clients
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/set_link_state
/gazebo/set_model_state
/imu
/joint_states
/odom
/rosout
/rosout_agg
/scan
/tf
```

Fig. 2.4 Sample ROS Topics

developed with C and C++ language. The Gazebo can easily create and run experiments rapidly with solid physics and good graphics. It works with OpenGL rendering and provides many codes for sensor simulation and actuator controls. This is one of the most popular simulators in robotic application development. It is a good tool for testing new concepts, systems, and algorithms Koenig and Howard (2004). Several research groups have used the ROS and Gazebo environment to work with multiple robot programming, and testing algorithms Anggraeni et al. (2020) Sadeghian et al. (2017). Simulation is very important for testing the software programs, robot behaviours in the environment, and controlling robots with new algorithms. Testing results with the simulator in experiments agree with results in real developed environments Takaya et al. (2016*b*). Gazebo simulators can be used to test robotics algorithms, design robots, and perform regression testing with realistic scenarios with a graphical user interface. This is one of the good virtual worlds to simulate multiple robots in the same environment Yao et al. (2015). A set of ROS packages named *gazebo ROS pkgs* is worked as wrappers around the stand-alone Gazebo. Models in Gazebo can be spawned and deleted dynamically using the services gazebo spawn model and gazebo delete model. We had to simulate multiple robots in the Gazebo environment. Therefore we developed our scripts to spawn multiple robots at different places simultaneously to complete our experiments. The sample gazebo environment we used in our experiments is shown in Figure 2.6. We have developed the scripts to spawn multiple robots at different initial locations in the Gazebo environment, as shown in Figure 2.7.The roslaunch tool is used as the standard method for starting ROS nodes and bringing up robots in the ROS environment. The command *roslaunch gazebo ros empty world.launch* is used to launch the open world in Gazebo. We need to select the correct version of the ROS working with Gazebo simulation packages to develop the application with the Gazebo simulation environment. The execution of the TurtleBot on the Gazebo simulation with the launch file is shown in Figure 2.5.


File Edit View Search Terminal Tabs Help
/opt/ros/melodic/share/rosbridge_server/launch/rosbridge_websocket.launch http:...
/home/samantha/catkin_ws/src/turtlebot3_simulations/turtlebot3_gazebo/launch/t...

```
samantha@samantha-Aspire-A315-57G:~$ cd catkin_ws/
samantha@samantha-Aspire-A315-57G:~/catkin_ws$ source devel/setup.bash
samantha@samantha-Aspire-A315-57G:~/catkin_ws$ cd ..
samantha@samantha-Aspire-A315-57G:~$ roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch
... logging to /home/samantha/.ros/log/74245d04-1148-11ec-9ab8-9408533faf4b/roslaunch-samantha-Aspire-A315-57G-4964.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.
```

Fig. 2.5 Execution of the Launch File for TurtleBot in Gazebo

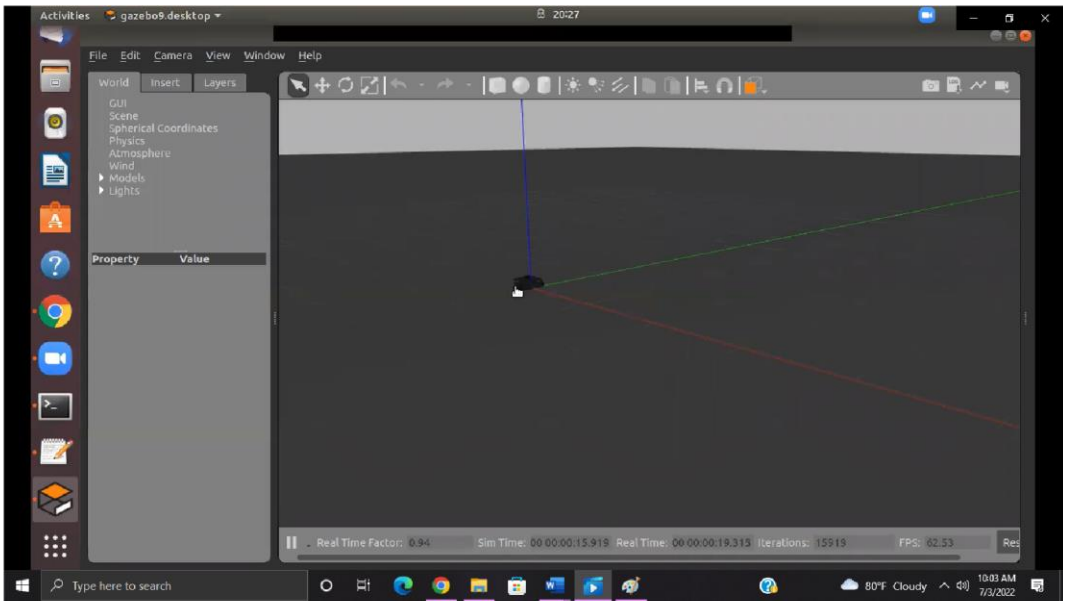


Fig. 2.6 Gazebo Simulator with Single Robot

2.4.4 ROSbridge Server

Rosbridge server is working with the WebSocket transport layer. A WebSocket is a full duplex communication protocol on a TCP connection. The communication and

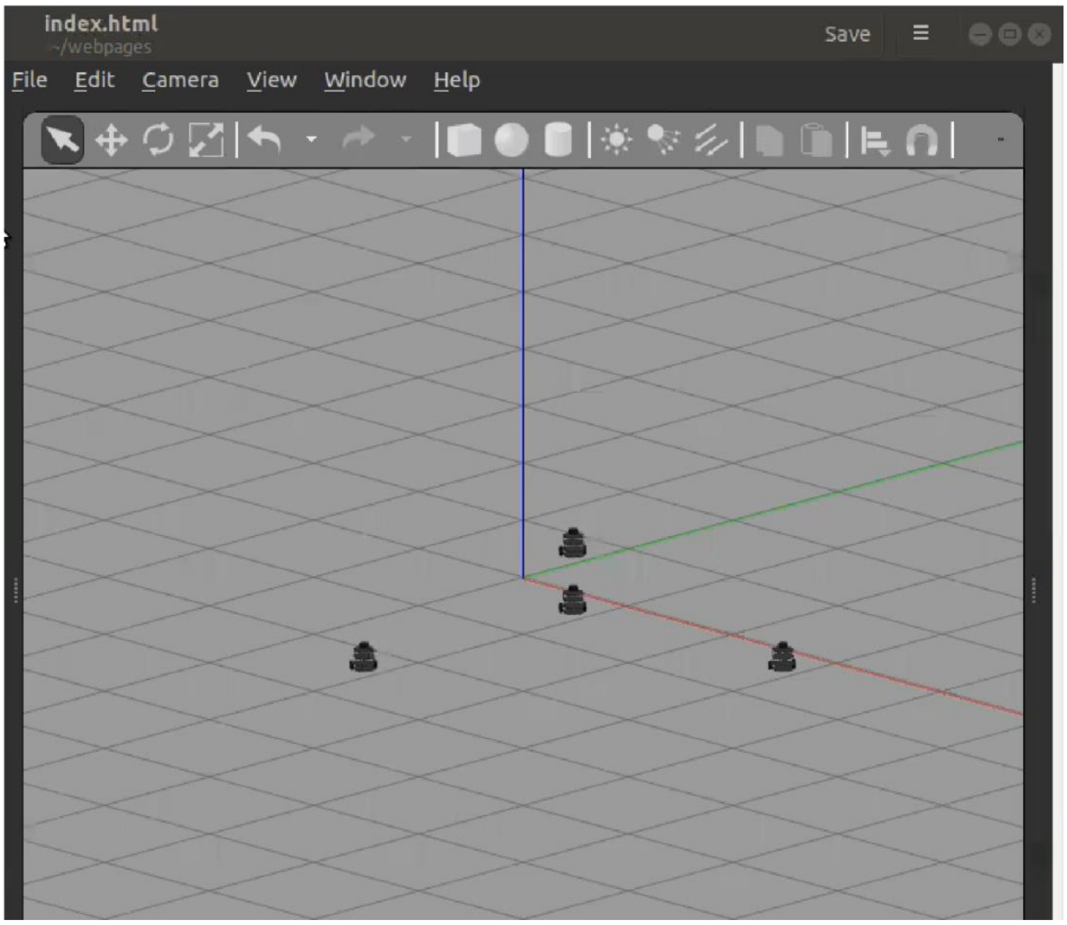
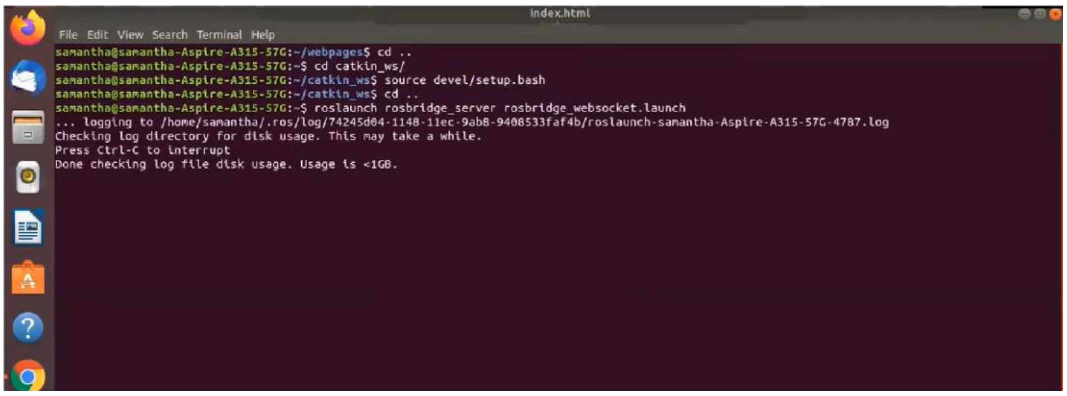


Fig. 2.7 Gazebo Simulator with Multiple Robots



```
samantha@samantha-Aspire-A315-57G:~/webpages$ cd ..
samantha@samantha-Aspire-A315-57G:~$ cd catkin_ws/
samantha@samantha-Aspire-A315-57G:~/catkin_ws$ source devel/setup.bash
samantha@samantha-Aspire-A315-57G:~/catkin_ws$ cd ..
samantha@samantha-Aspire-A315-57G:~$ roslaunch rosbridge_server rosbridge_websocket.launch
... logging to /home/samantha/.ros/log/74245d04-1148-11ec-9ab8-9408533faf4b/roslaunch-samantha-Aspire-A315-57G-4787.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.
```

Fig. 2.8 Execution of the ROSbridge Server

interaction between a web browser (or other client application) and a web server can be achieved using a WebSocket. Web pages can communicate with ROS using the ROSbridge protocol. The rosbridge protocol provides the ability to fragment messages and compress messages. ROSbridge uses the JSON format to transfer the messages. Rosbridge library uses the Python library to work on web pages, convert JSON strings into ROS messages, and vice versa. Most of the HRI research work was completed with the help of Rosbridge library support Crick et al. (2012). Roslibjs is a JavaScript library for the browser that can talk to ROS via the rosbridge server. Rosbridge provides access to ROS topics and services available over TCP sockets or WebSockets as JSON messages. This server is listening on port 9090 to create connections. Rosbridge can be used by the client's program to publish and subscribe to topic messages and invoke services in the server. Robert Codd-Downey et al. have developed an interface using Rosbridge libraries to communicate and control robots easily Codd-Downey and Jenkin (2015). The execution of the ROSbridge server using the laucnch file is shown in Figure
2.8.

2.4.5 Semantic Web

The Semantic Web is considered an expansion of the current Web and is not a separate Web. The Semantic Web provides a better meaning to the Web's content so that machines can communicate with each other like people are communicating. Several research works are going on to complete this requirement by many researchers. The Semantic Web was developed to provide a common framework that allows making machine communication easier and data to be shared. In addition, it enables software reuse across applications, enterprises, organizations and society with the community. W3C develops the Semantic Web with many researchers and industry partners. Most of the standard was developed by W3C and pubished. Resource Description Framework (RDF) was used initially to develop the Semantic Web. The semantic Web provides intelligent access to the Web, making communication among machines more accessible. Again it provides intelligent access to heterogeneous and distributed web resources, enabling software systems and agents to mediate between user needs and available information sources Lassila et al. (2000). Moreover, the software agents can communicate with others with less intervention using ontologies in the Semantic Web Hendler (2001).

2.4.6 Ontology

Ontologies play the primary role in the Semantic Web and are used as the backbone in representing knowledge in applications. Therefore, finding required and appropriate ontologies for the given domain is significant, but it is now a difficult and tedious task since the numbers of ontologies keep developing rapidly. It is a complicated task to develop an ontology starting from scratch even though there are some tools since it is time-consuming, needs a good understanding of the domain and is expensive to construct. The World Wide Web Consortium (W3C) has recommended XML, XML Schema, RDF, RDF Schema and Web Ontology Language (OWL) as standards and tools for developing the application with the semantics of the content. Therefore, ontologies play a vital role in knowledge representation for the Semantic Web. The Semantic Web represents a common framework for sharing and reusing data across multiple applications, organizations, and community boundaries. Semantic Web is based on the Resource Description Framework (RDF), but there are some limitations to the RDF. Therefore, currently, researchers are using ontology to solve the limitation of RDF. The format and the structure of RDF are straightforward to represent, and we can use RDF triples in the form of subject, predicate, and object, as shown below Yun-hua and Dan (2010). Some researchers have converted the existing database models into RDF for developing some applications with semantics Tong (2018).

Statement = (Subject, Predicate, Object)
Subject: The resources that are being described by RDF with a URI.
Predicate: This is also a kind of resource which can have a name.
Object: It is also an RDF URI reference or a blank node.
Statement: A statement contains a resource, a property, and an associated value.

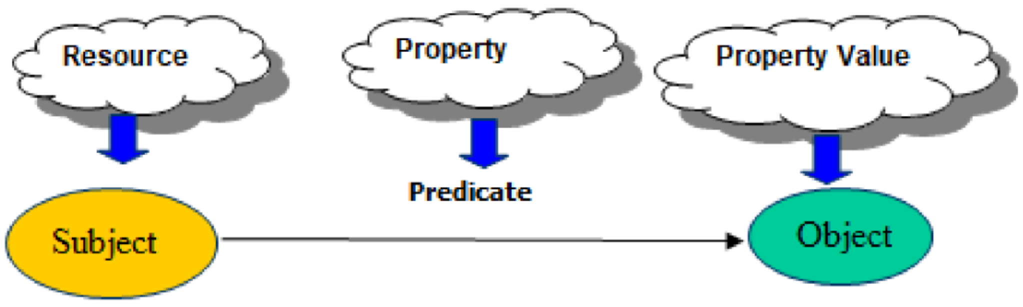


Fig. 2.9 Triple in RDF

```
<?xml version="1.0"?>
<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:a="http://description.org/schema/">
       <rdf:Description
       about="http://www.sliit.lk/Home/~Samantha">
           <a:Creator>Samantha Rajapaksha</a:Creator>
  </rdf:Description>
</rdf:RDF>
```

Fig. 2.10 XML code for RDF Triple

RDF triple with graphical representation is shown in Figure 2.9. RDF has several limitations, including it is difficult to declare the range restriction on some classes, challenging to represent the disjointness between two classes, and difficulty in implementing the union intersection and complement with classes. All these issues and limitations can be avoided using ontologyFurthermore, most of the knowledge bases can be developed using the ontology Staab et al. (2001) Maedche et al. (2003) Lim et al. (2011). There are several research challenges related to applying ontologies in real-world environments. However, it provides the meaning to the content to complete machine-tomachine communication efficiently.

Assume there is a resource created by Samantha Rajapaksa, then it can be represented using RDF triple as shown in Figure 2.10

Samantha Rajapaksha is the creator of the resource http://www.sliit.lk /Home / Samantha

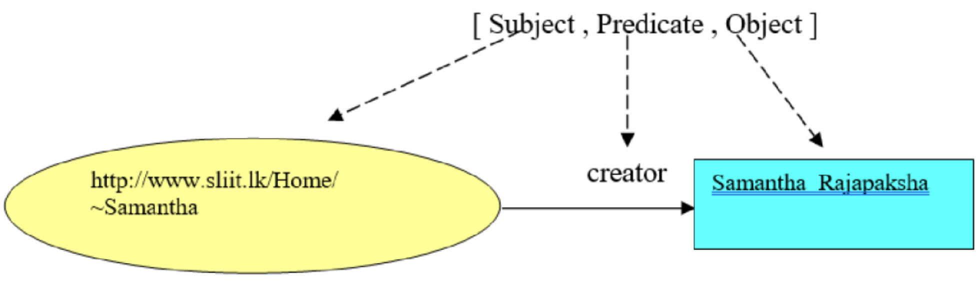


Fig. 2.11 Triple in RDF as example

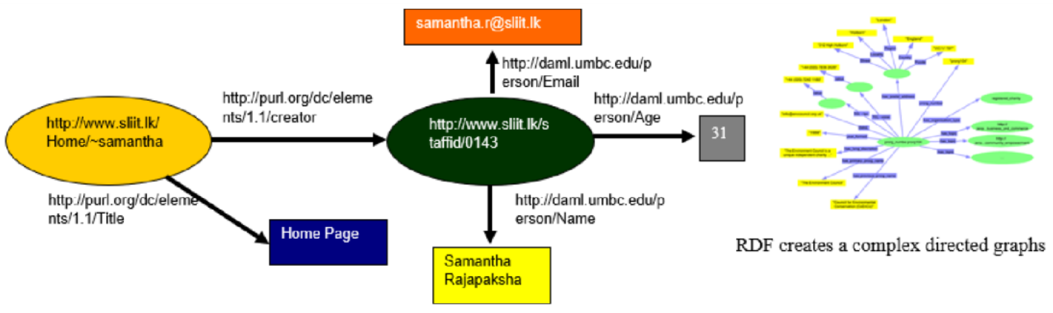

samantha.r@sliit.lk
http://daml.umbc.edu/person/Email
http://purl.org/dc/elements/1.1/creator
http://daml.umbc.edu/person/Age
http://www.sliit.lk/Home/~samantha
http://www.sliit.lk/staffid/0143
31
http://purl.org/dc/elements/1.1/Title
Home Page
http://daml.umbc.edu/person/Name
Samantha Rajapaksha
RDF creates a complex directed graphs

Fig. 2.12 RDF Graph with more concepts

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:p=" http://daml.umbc.edu/person">
     <rdf:Description about="http://www.sliit.lk/Home/~samantha">
         <dc:Creator>
              <rdf:Description about="http://www.sliit.lk/staffid/0143">
                   <p:Name>Samantha Rajapaksha</p:Name>
                   <p:Email>samantha.r@sliit.lk</p:Email>
                   <p:Age>31</p:Age>
              </rdf:Description>
         </dc:Creator>
         <dc:Title> Home Page </dc:Creator>
     </rdf:Description>
</rdf:RDF>
```

Fig. 2.13 XML Code for Extended RDF graph

The RDF file can be extended as shown in Figure 2.12 to represent more concepts and their relationship. The Web Ontology Language (OWL) is used to represent the ontology for the selected domain. The valuable rich, and complex knowledge about things, groups of things, and relations between things can be represented using the semantic web language called W3C web ontology language. It is a computational logicbased language that can be used to knowledge expressed in OWL can be expressed by computer programs. The OWL web ontology language is designed to develop web applications that can process the content with semantics without just presenting the content to humans. It provides very high machine interoperability with web content than provided by XML, RDF, and RDF Schema.OWL provides three sublanguages based on the expressive power of the content OWL lite, OWL Descriptive Logic and OWL full. Several tools can be used to create the ontology from scratch. Protege is one of the handy open-source ontology editors that can be used to create our ontology.

Ontologies are used to describe taxonomies and formally define the structure of knowledge for various domains. For example, in a given domain, the nouns can be represented as classes of objects, and the verbs can be used to represent relations between the objects. The OWL Web Ontology Language is an international standard that can be used to encode and exchange ontologies to a developed application that is needed with the semantic meaning of the content.

In some instances, user instruction may have some synonyms, so our system must identify the corresponding synonym to select the relevant ROS topics and nodes. Some research groups completed ontology-based synonym identification with the use of deep neural network technologies Shen et al. (2018). Some researchers have used the multistrategy to extract the synonym using a page ranking algorithm, pattern matching algorithm, and literal similarity algorithm Lu et al. (2009). Some researchers have used Neural network-based technologies to manage the user instructions with semantics Luo and Chen (2017). The other research problem is identifying the user instruction’s synonyms and related ROS topics and nodes for subscription and publication.

CHAPTER 3

Methodology

The authors have implemented a web interface to interact with the robots and users.ROS Bridge Server was used to connect the web interface and ROS middleware. The Web interfaces were developed to interact with different experiments in our research as described by the table 1.1. Web Interface I to IV was developed with simple instructions like moving the Robot forward, moving the robot circle and getting the Robot’s current position. Web Interface V was developed to work with instructions with synonyms. We have used the Gazebo simulator for our experiments. The standard ROS JavaScript Library provided by the ROS Web Tools (http://robotwebtools.org/) was used to connect ROS with the Web interface. In the last experiment, the user can issue multiple instructions to all robots placed at different positions. Figure 3.1 represents the system architecture of our system.

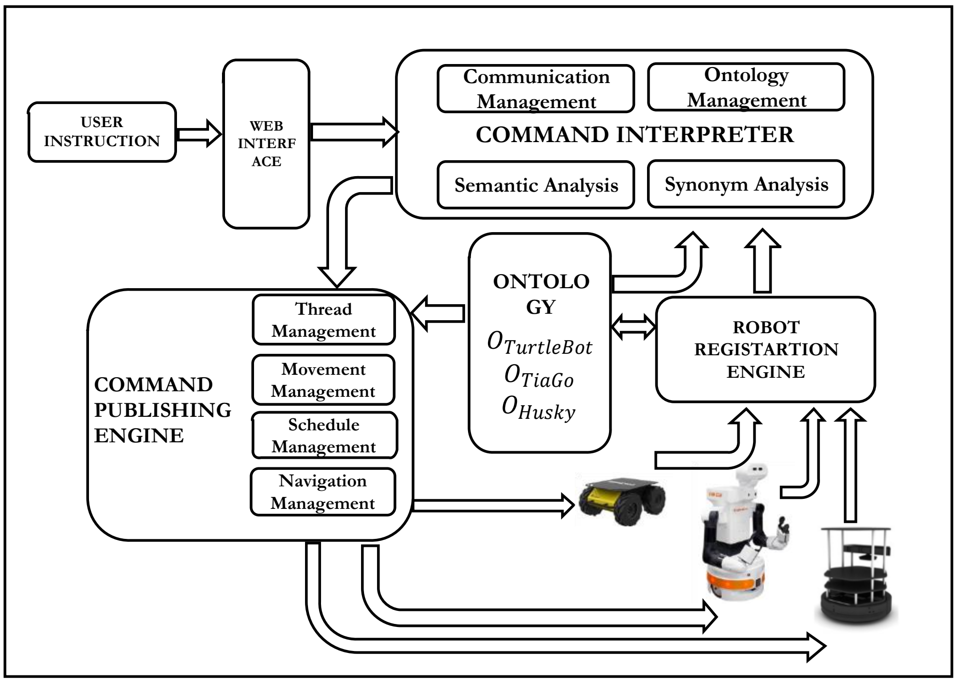


Fig. 3.1 System Architecture Diagram

3.1 Robot Registration Engine

All multiple robots want to register with our robot registration engine by providing the software details and hardware-related specifications semi-automatically. Figure 3.2 describes the robot registration algorithm. Firstly, it implements a node in ROS named ”regRobot” to fulfil all the lines in the algorithms.

INOUT: *IP address list, URDF file and ontology*

OOTPUT: *Updated ontology with ROS topic, Nodes and services*

ALL_ROBOT_REGISTRATION_ALGORITHM (*ipAddressList, URDF_File, RobotOntology* **)**

```
1.  Begin
2.  Develop a Process as "regRobot" using ROS node//All lines will be run by this process.
3.  Foreach ipAddress ∈ ipAddressList do
4.  //The hardware details were updated with user involvement
5.  Connect the Robot using IP address
6.  If Robot has wheels Then
7.  Foreach $i \in \{1,2,3,4,5,6,7,8\}$ do
8.  Get the hardware details with user and URDF_File
9.  Update RobotOntology
10. Endfor
11. Else
12. Foreach $j \in \{bipedal, tripedal, quadrupodel\}$ do
13. Get the hardware details with user and URDF_File
14. Update RobotOntology
15. Endfor
16. Endif
17. //Update the software details with ROS commands
18. Foreach item in $\{rostopic, rosnode, rosservice\}$ do
```

```
19.     Select the necessary option for each command
20.     Execute item on command prompt with the execl() system call
21.     Collect the output lists and set as L
22.     Foreach Line in L do
23.     Update the ontology with relevant class
24.     Endfor
25.     Edfor
26.     Endfor
27.     Publish the ontology with "regOnto".
28.     End
```

Fig. 3.2 Robot Registration Algorithm.

Each Robot is connected using the *ip* address from the given *ip* address list. If the Robot is already registered, then the ontology has no changes. Otherwise, the hardware specification can be collected from the URDF file with human intervention. Software specification can be collected by running the ROS commands using the *execl*() system call. Available ROS topics, nodes, message types, and other details have been updated in the ontology. Finally, this algorithm Publish the updated ontology on the ROS topic named "*regOnto*".

We initially created a node called "*regRobot*" to complete the rest of the line execution of the algorithm. IP addresses were extracted from the given IP address list named "*ipList*". Use the IP address to connect all heterogeneous service robots in the gazebo environment. Then the hardware details were collected using human involvement with the URDF file. Next, ROS commands were executed to collect the Software specification, which used the *execl()* system call by the ROS node created earlier. Finally, ontology named "*Registration Ontology*" is created to represent available ROS details.

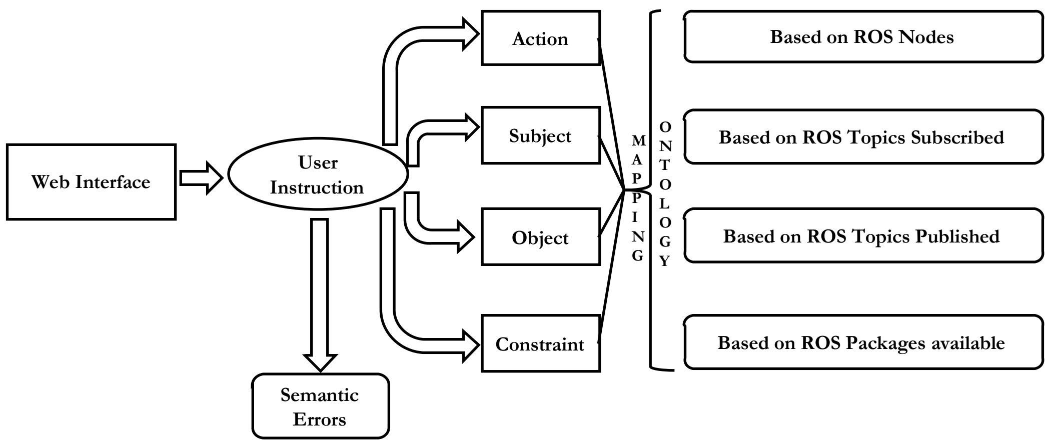


Fig. 3.3 Initial Interpretation Process.

Figure 3.4 represents the Flowchart for the algorithm we initially developed for registration robots.

3.2 Command Interpreter

When a user issues a high-level user instruction on the web interface provided by the system, the instruction is analyzed by the command interpreter to separate the action, subject, object, and constraint, as shown in Figure 3.3. First, the processed instruction can be sent to the synonyms analysis and semantic analysis process. Then it needs to find out relevant ROS nodes, ROS topics for subscription, and publication with the algorithm. Finally, the ROS Topic Identification algorithms are used to identify the related ROS topic to control the robots. The ROS Topic Identification algorithm will be explained later in this chapter. When users issue multiple instructions sequentially, we need to interpret them separately. Therefore, we have designed a state transition engine with multiple instructions. The system is designed to work with multiple instructions one by one issued by the user using a state transition diagram with the states' description as shown in Figure 4.17. The robot state is saved in the ROS topic to retrieve the robot state from time to time. When the Robot is ready, it will accept the user's instruction and complete the assigned work accordingly.

When a user issues multiple instructions to the Robot through the Web interface, the related Flowchart with the state transition is shown in Figure 3.6. Initially, a robot must register with the Robot Registration Engine and update the state as Ready in the

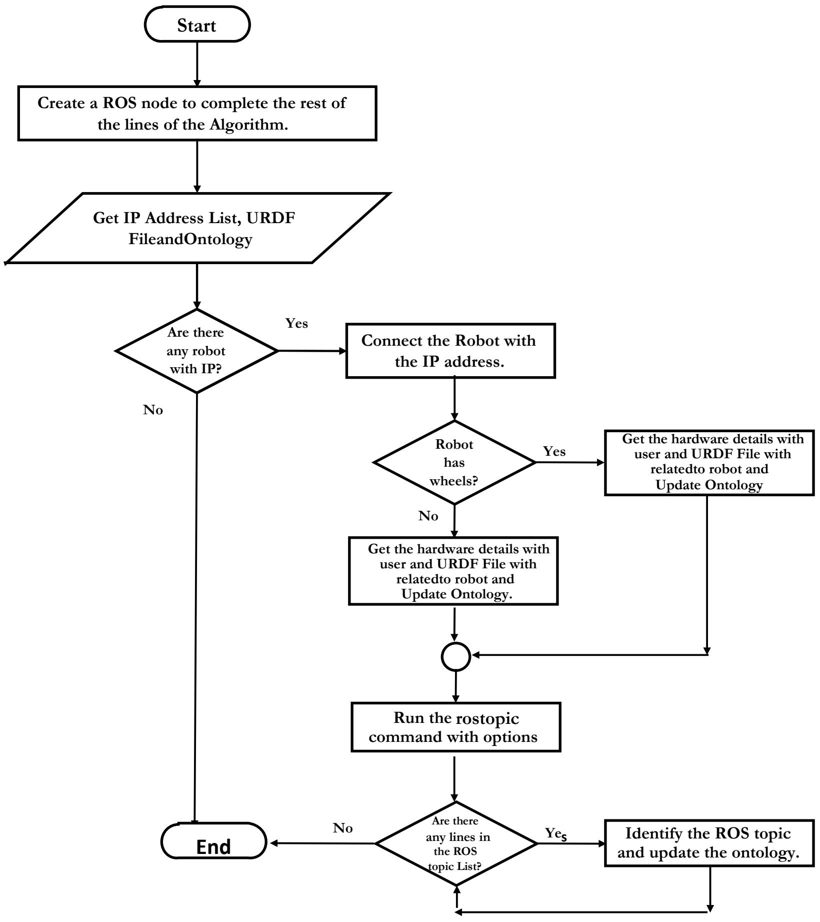
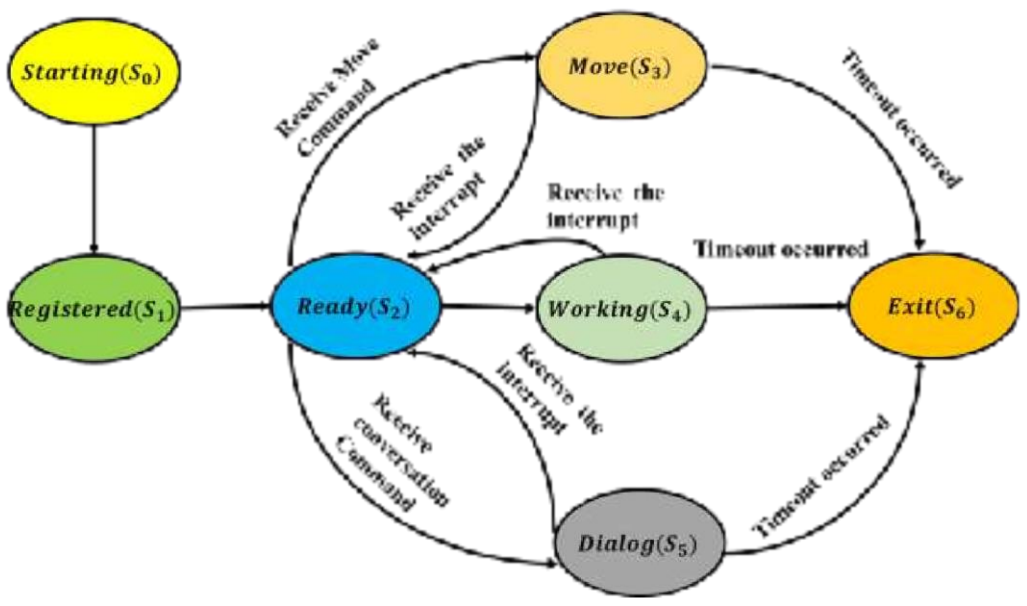

Start
Create a ROS node to complete the rest of the lines of the Algorithm.
Get IP Address List, URDF FileandOntology
Are there any robot with IP?
Yes
No
Connect the Robot with the IP address.
Robot has wheels?
Yes
Get the hardware details with user and URDF File with relatedto robot and Update Ontology
No
Get the hardware details with user and URDF File with relatedto robot and Update Ontology.
Run the rostopic command with options
Are there any lines in the ROS topic List?
No
Yes
Identify the ROS topic and update the ontology.
End

Fig. 3.4 Robot Registration Algorithm.

ROS topic. Then the Robot can work according to the instruction given by the user. While the first instruction is processed, the user can issue another instruction then the Robot must be interrupted to handle the second instruction. Based on the priority of the instruction, the Robot must be able to decide to continue the current work or start the second instruction. The work state has the highest priority, the motion state has the second-highest priority, the dialogue state has the third priority, and the ready has the lower priority. Each Robot will exit the system if the instructions are not received within the defined time interval.

3.3 Movement Management

The most critical component of our experiments is the robot movement using different instructions and different interfaces. Once a robot is registered with the RRE, it uses the ROS Topic Identification Algorithm to identify the corresponding ROS topic for the Movement. We have used different techniques to move the robots in different experiments. In experiment 01, the authors used teleoperation to move robots forward and circle in an open environment in Gazebo. In experiments 02,03, and 04, authors



S_0:*Starting State*
S_1:*Registred State*
S_2:*Ready State*
S_3:*Move State*
S_4:*working State*
S_5:*Dialog State*
S_6:*Exit State*
$S_0 \rightarrow S_1$ *if Robot has registred.*
$S_1 \rightarrow S_2$ *if Robot is ready for inputs.*

$S_2 \rightarrow S_3$ *if Move command is received.*
$S_2 \rightarrow S_4$ *if Work command is received.*
$S_2 \rightarrow S_5$ *if dialog command is received.*

S_3
$S_4\} \rightarrow S_2$ *if any interrupt command is received.*
S_5

S_3
$S_4\} \rightarrow S_6$ *if any timeout has occured.*
S_5

Fig. 3.5 State Transition Diagram.

used the Web-based interface to move robots forward and circle in an open environment in Gazebo with multiple robots. In experiment 05, the Robot was moved to a specific location using the algorithm given in Figure 3.7. The notations in the Flowchart are described in Table 3.1.

3.4 Ontology

Ontology is a model used to represent the concept and the relationships among all related concepts; for example, if we select the Robot's ontology, we can represent all concepts in the robot domain and the relationships among all concepts related to robots. We have created ontology to represent the concepts. User intervention is needed to update the ontology. Finding concepts from Ontology is the one that takes more time because the running time complexity of the searching algorithm is given by *O(n)* where *n* is the number of classes in the given ontology. The part of the ontology we created is shown in Figure 3.8.

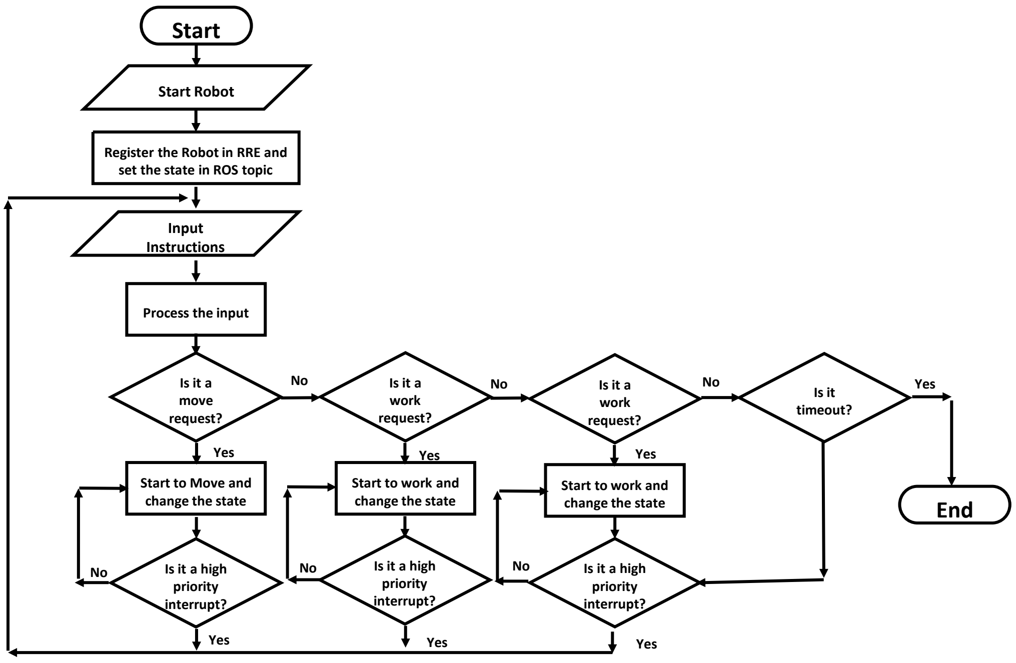

Start
Start Robot
Register the Robot in RRE and set the state in ROS topic
Input Instructions
Process the input
Is it a move request?
No
Is it a work request?
No
Is it a work request?
No
Is it timeout?
Yes
End
Yes
Start to Move and change the state
Yes
Start to work and change the state
Yes
Start to work and change the state
No
Is it a high priority interrupt?
No
Is it a high priority interrupt?
No
Is it a high priority interrupt?
Yes
Yes
Yes

Fig. 3.6 Flowchart for Multiple Instruction Handling.

Table 3.1 Notations used in the Flowchart and Experiments

U_x^s ω_z^s	*Description* *ms*$^{-1}$.
ω_z^e θ	Linear Speed of the Robot in x direction at the start in Angular Speed of the Robot in the z-direction at the start in Angular Speed of the Robot in the z-direction at the stop in Current Robot Orientation in quaternion form, The difference between Current Robot Orientation and Goal orienta- *ms*$^{-1}$. *ms*$^{-1}$. tion in quaternion form,

3.5 Synonym Analysis

Users can enter different types of instructions, and the system accepts only commands and commands with the condition. Some commands with different verbs with the same meaning can be called synonyms. Robots may only be able to understand synonyms once it is appropriately programmed. Therefore, we implemented ontology created with the Web Ontology Language property called *"sameAs"* to find the synonyms in the given instruction. We have used the *"owl:sameAs"* statement to identify the two Uniform Resource Identifiers. That means each individual has the same *"identity"*. We can take the example as synonyms for instruction *"move"* are *"shift, go, proceed, walk and advance"*. Users can update ontology manually. Synonym identification is used in the ROS topic Identification algorithm for publishing commands. Different heterogeneous service robots can use different ROS topics; therefore, we must find the correct ROS topic to publish the commands. The semantic analysis algorithm is described in Figure 3.11.

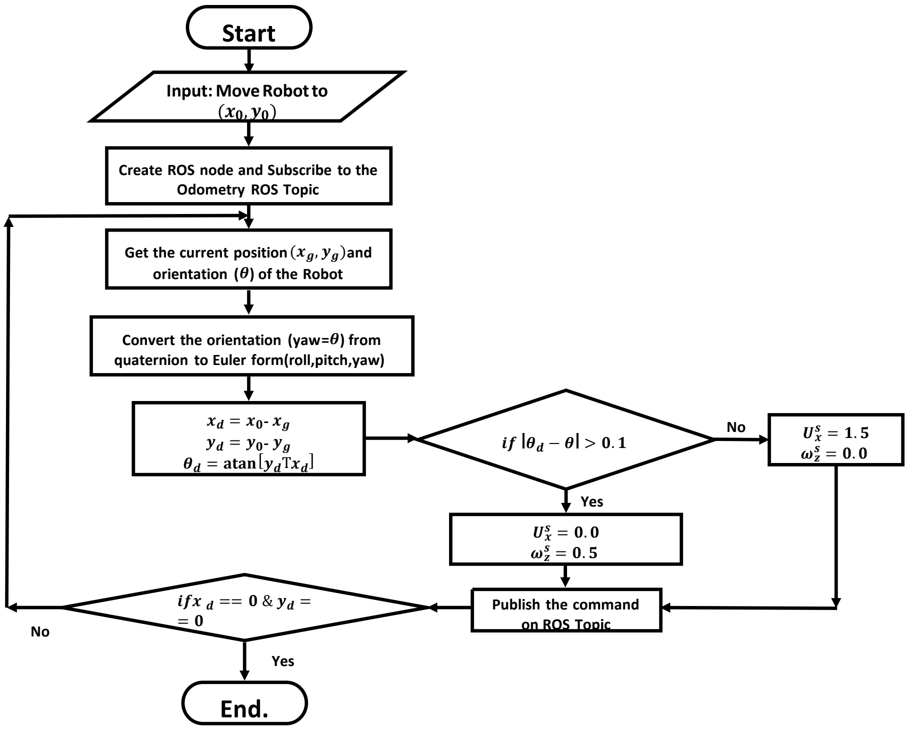

Start
Input: Move Robot to (x_0, y_0)
Create ROS node and Subscribe to the Odometry ROS Topic
Get the current position (x_g, y_g) and orientation (θ) of the Robot
Convert the orientation (yaw=θ) from quaternion to Euler form(roll,pitch,yaw)
$x_d = x_0 - x_g$
$y_d = y_0 - y_g$
$\theta_d = \text{atan}[y_d \mathrm{T} x_d]$
$if\ |\theta_d - \theta| > 0.1$
No
$U_x^s = 1.5$
$\omega_z^s = 0.0$
Yes
$U_x^s = 0.0$
$\omega_z^s = 0.5$
Publish the command on ROS Topic
$if x_d == 0 \ \& \ y_d == 0$
No
Yes
End.

Fig. 3.7 Flowchart for moving Robot to a specific Goal.

3.6 Semantic Analysis

The semantic meaning of the command is one of the main tasks in interpreting userlevel instructions. Suppose a robot can detect a semantic error in the given user-level instruction that will better implement the Robot's intelligence. Therefore, understanding the semantics of the command can be achieved if we can detect the semantic error of the instruction using ontology. For example, when a user issues a user-level instruction with the verb "go," we can guarantee that the next part should be a location or destination. Figure 3.13 describes the semantic analysis algorithm.

The ontology code has a property that requires restricting all robots from moving to a specific position. *"owl:allValuesFrom"* is the property that can be used to define the class with all possible values of the given property defined by *"owl:onProperty"*. If the object is not in the restricted value list, it is considered an invalid command and gets the user intervention.

3.7 Command Publishing Engine

According to the user-level instruction, the command interpreter can identify the action *(move, navigate, Identify)* subject, constraint, and object defined in the user instruction. Then, the command publishing engine needs to identify the corresponding ROS topics relevant to the action to publish and subscribe for initiation of the action. For example,

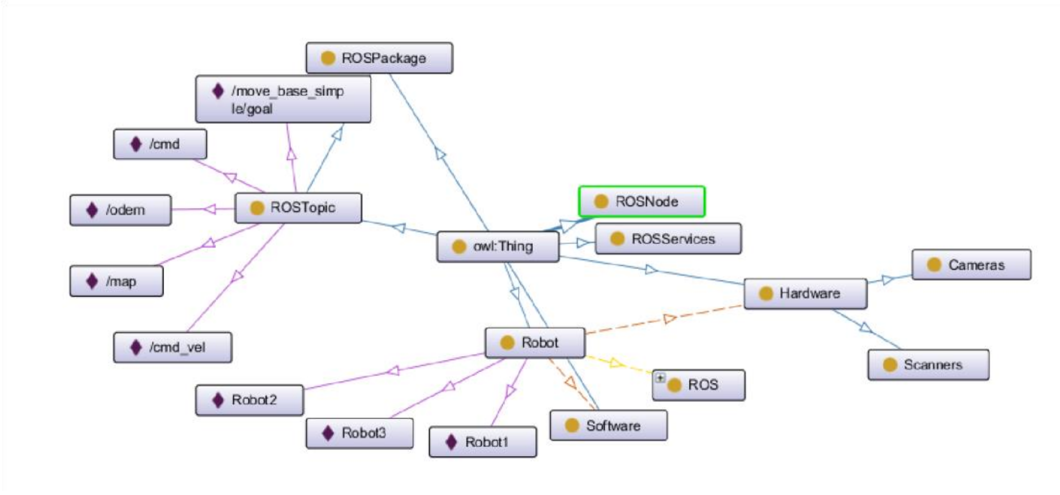


Fig. 3.8 Fragment of the Ontology

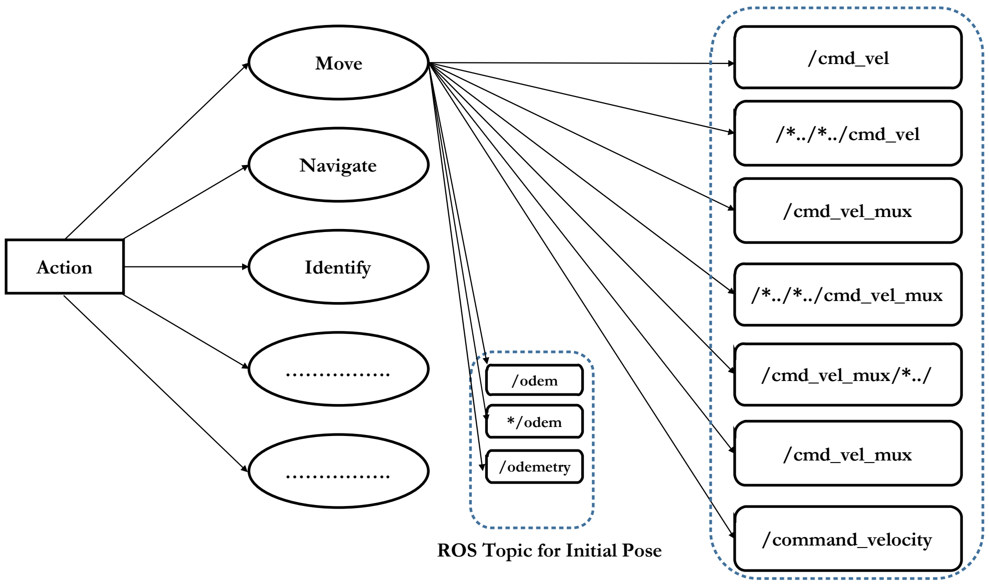


Fig. 3.9 ROS topics for the Movement.

if we want to move the Robot to a specific location, we can publish the command on ROS topics like *cmd_vel*, *cmd_vel_mux* or *cmd_vel_mux/input/navi*. Of course, these ROS topics will vary from Robot to Robot in heterogeneous environments. For example, the possible ROS topics for the Movement and ROS topic for the initial pose is shown in Figure 3.9.

The Get Position and Orientation algorithm is defined with the Flowchart as shown in Figure 3.14. Initially, it connects with the Robot using the IP address and gets the ROS topic list from the ontology updated by the Robot Registration Engine. Then, it searches for the ROS topic related to the *odometry* to get the current position and orientation of the Robot by subscribing to the *odometry* related ROS topic. If the relevant ROS topic is unavailable, we have taken the user interventions to find the ROS topic for the position and orientation.

```
<rdf:Description rdf:about="#move">
       <owl:sameAs rdf:resource="#go"/>
       <owl:sameAs rdf:resource="#progress"/>
       <owl:sameAs rdf:resource="#advance"/>
       <owl:sameAs rdf:resource="#shift"/>
</rdf:Description>
```

Fig. 3.10 OWL:sameAS Syntax

SYNONYM_ANALYSIS_ALGORITHM (*action , ontology* **)**

```
1.  Start
2.  Create a ROS node as "synAlgo" //Rest of the codes execute by this node.
3.  for each class ∈ ontology do
4.  //Get all classes of the ontology
5.  if action == class then    // Find the class for the action
6.  Call Get_ROSTOIC(action)
7.  else
8.  Get the sameAs List
9.  for each i ∈ sameAs List do
10. if action == i then    // Find the synonym for the action
11. Call Get_ROSTOIC(action)
12. else
13. Get the user inputs
14. endif
15. endfor
16. endif
17. endfor 18. End.
```

Fig. 3.11 Synonym Analysis Algorithm

When a user enters the instruction to all heterogeneous service robots, we need to initiate the action for each Robot. This task is completed by Command Publishing Engine (CPE), which can publish the action on the corresponding ROS topic. Initially, CPE can locate the current position of each Robot using the optimized algorithm. Get Robot Position algorithm of each Robot is defined in Figure 3.16. The algorithm has used the *ip* address and the undated ontology to get the initial position and the orientation.

We have created a node in ROS called "*initPos*". It is responsible for running the remaining lines of the defined algorithm. In addition, this node can find the relevant ROS topics related to the initial position and orientation of the Robot.

Each Robot may have a different ROS topic to subscribe to and publish for different operations. Therefore, we need to identify these topics before executing commands on each Robot. Figure 3.17 describes the ROS topic identification algorithm. Initially, the system used the given IP address list and ports list to connect with all robots. Used the ROS topic in the ontology, which the RRE generated previously to create a shared file as

```
<owl:Class rdf:about="#Robots">
      <rdfs:subClassOf>
            <owl:Restriction>
                  <owl:onProperty rdf:resource="#moveTo"/>
                  <owl:allValuesFrom rdf:resource="#Location"/>
            </owl:Restriction>
      </rdfs:subClassOf> </owl:Class>
```

Fig. 3.12 OWL:Restriction Syntax

INPUT: *User Instruction, Ontology*

OUTPUT: *Identify the instruction as valid or invalid*

SEMANTIC_ANALYSIS_ALGORITHM (*Instruction, object, RobotOntology* **)**

```
1.   Begin
2.   Develop a process using ROS node as "semAlgo" //This is resposible to run following lines.
3.   Foreach $class \in ontology$ do
4.   //Get all classes of the ontology
5.   If $action == class$ then      // Find the class for the action
6.   Foreach $restriction \in class$ do
7.   If $allValueFrom == object$ then
8.   Call Get_ROSTOIC(action)
9.   Else
10.  Output as invalid request
11.  Endif
12.  Endfor
13.  Else
14.  Get the sameAs List
15.  Foreach $i \in sameAs\ List$ do
16.  If $action == i$ then      // Find the synonym for the action
17.  For each $restriction \in class$ do
18.  If $allValueFrom == object$ then
19.  Call Get_ROSTOIC(action)
20.  Else
21.  Output as invalid request
22.  Endif
```

```
23.                 Endfor
24.                 Else
25.                 Get the user inputs
26.                 Endif
27.                 Endfor
28.                 Endif 29.    Endfor 30. End.
```

Fig. 3.13 Semantic Analysis Algorithm

rtList. Then it called the Get ROSTopic() algorithm to get the corresponding ROS topics for each action. This algorithm was used to find the ROS topics for each action defined in the user instruction. For example, suppose the action is to move the Robot from one location to another location. In that case, we need to find the corresponding ROS topic used from the identified list as *'cmd', 'vel', 'cmd vel', 'velocity', 'speed', 'travel', 'run'*. If the identified ROS topics list did not match the ROS topics received from

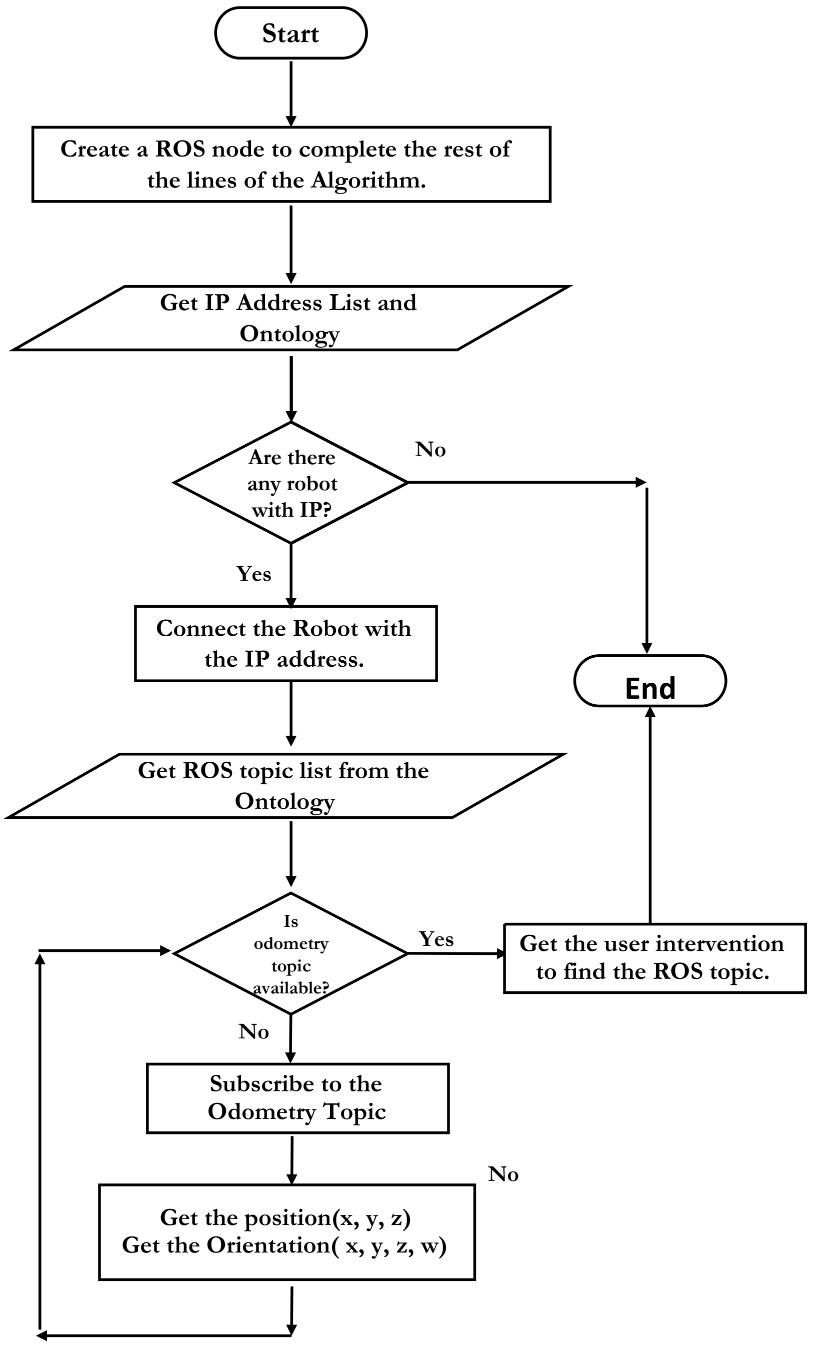

Start
Create a ROS node to complete the rest of the lines of the Algorithm.
Get IP Address List and Ontology
Are there any robot with IP?
No
Yes
Connect the Robot with the IP address.
End
Get ROS topic list from the Ontology
Is odometry topic available?
Yes
Get the user intervention to find the ROS topic.
No
Subscribe to the Odometry Topic
No
Get the position(x, y, z)
Get the Orientation(x, y, z, w)

Fig. 3.14 Get Initial Position Algorithm

the RRE, we called Get Uncertain ROSTopic() to find the ROS topics with synonyms of the action based on the ontology. This algorithm uses the synonyms for the given action to find the corresponding ROS topic. If we can find one, we can use the topic for subscribing or publishing the action; otherwise, we need to get the user input to resolve the problem. The interaction among ROS nodes, ROS topics and ROS master is shown in Figure 3.15

We have conducted experiments with the system using the two robots with level 01, level 02, and level 03 instructions with twenty different instructions. The twenty

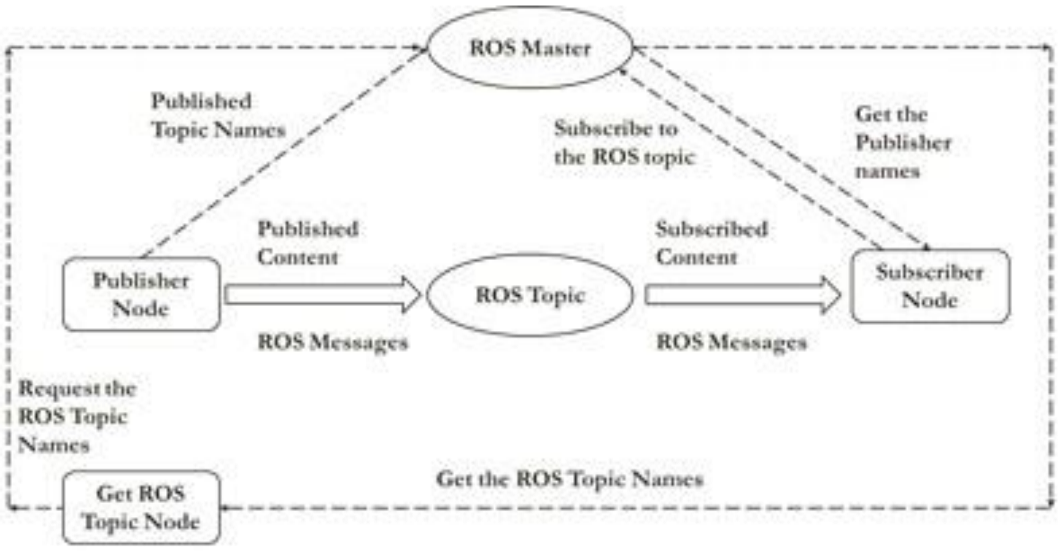


Fig. 3.15 ROS Nodes and Topics

INPUT: *Ip List and RobotOntology*

OUTPUT: *Position and Orentation*

Get_Initial_Position_Orientation (*ipList, RobotOntology***)**

```
1.      Begin
2.      Develop a process withh node in ROS as "initPos" //This will run following lines.
3.      Using the ip address connect with the Robot.
4.      Get the rostopic list from the RobotOntology.
5.      Foreach element in rostopic list do
6.      If element is in {odom,odometry,odomet} Then
7.      Use the element to get odometry content formthe ROS topic
8.      Retrive the position details in the form of (x, y, z)
9.      Retrive the Orientation details in the form of ( x, y, z, w)
10.     Add the content in the RobotOntology
11.     Else
12.     User needs to provide the ROS topic
13.     Use the element to get odometry content formthe ROS topic
14.     Retrive the position details in the form of (x, y, z)
15.     Retrive the Orientation details in the form of ( x, y, z, w)
16.     Add the content in the RobotOntology
17.     Endif
18.     Return the updated ontology.
19.     Endfor
20.     End
```

Fig. 3.16 Get Initial Position Algorithm

instructions were developed using different synonyms for the given verb *move*. Some synonyms were not implemented in the ontology. Therefore, it has generated errors for synonyms not in the ontology.

1. Level 01: Move 50m with velocity $1ms^{-1}$.

2. Level 02: Move to the point (20, 20) with velocity $1ms^{-1}$.

```
ROS_TOPIC_IDENTIFICATION_ALGORITHM (ipList, portList)

    1.  foreach $ip_i \in ipList$ & $port_i \in portList$ do
    2.  Get the rostopic list form $Ontology_{ip}$
    3.  Publish ROS Topic as shared file named as $rtList_i$
    4.  foreach $action_i \in$ {move, navigate, identify, etc.} do
    5.  $rt_i \leftarrow$ GET_ROSTOPIC($action_i$)
    6.  Record the ROS Topics get as $rt_i$

GET_ROSTOPIC ($action_i$)

    1. Open shared file $f_i \leftarrow rtList_i$
    2. if $action_i$ = move then
    3. foreach $rt \in$ {'cmd','vel','cmd_vel','velocity','speed','travel','run'} do
    4. foreach line $l_i$ in file $f_i$ do 5.     if rt is in $l_i$ then  //Exact Matching
    6.      Record ROS Topic rt for move.
    7.      else  //Non-Exact Matching
    8.      GET_UNCERTAIN_ROSTOPIC($action_i$)
    9.      if $action_i$ = navigate then
    10.     foreach $rn \in$ {'move_base','map_server','robot_amcl'} do
    11.     if rt is in $l_i$ then  //Exact Matching 12.     Record ROS Topic rn for navigate.
    13.       else  //Non-Exact Matching
    14.       GET_UNCERTAIN_ROSTOPIC($action_i$)
    15.       foreach $rt \in$ {'cmd','vel','cmd_vel','velocity','speed','travel',
    16.       'run','goal',move','amcl','navigation','scan','map','cloud',
    17.       'rt','local','global','odom'} do
    18.       foreach line $l_i$ in file $f_i$ do
    19.       if rt is in $l_i$ then  //Exact Matching 20.       Record ROS Topic rt for navigate.
    21.    else  //Non-Exact Matching
    22.    GET_UNCERTAIN_ROSTOPIC($action_i$)   GET_UNCERTAIN_ROSTOPIC ($action_i$)

    1.  Get the ontology ($O_a$) developed for $action_i$
    2.  foreach class $\in$ {$O_a$} do
    3.  If actioni == class then
    4.  Get the synonym list $SL_i$ from the ontology form the class
    5.  foreach $rt \in$ {$SL_i$} do
    6.  foreach line $l_i$ in file $f_i$ do
    7.  if rt is in $l_i$ then  //Exact Matching 8.      Record ROS Topic rt for navigate.
    9.      else  //Non-Exact Matching
    10.     Get the User Inputs for $action_i$
```

Fig. 3.17 ROS Topic Identification Algorithm

3. Level 03: Move to the given goal in the map.

We have used three levels of instructions for the experiment as defined in Table 3.2. The Level 01 instruction moves all robots to a given distance with the given velocity from the current position with the current orientation.

```
Level01_Interpretation(cmd, ipList, ontology)
1.  Start
2.  Create a ROS node as "lev01Intr" //Rest of the codes execute by this node.
3.  Get the distance and velocity from the cmd
4.  Set the values for linear(x, y, z)and angular(x, y, z)
5.  for each ip ∈ ipList do
6.  Connect with running robot using the ip.
7.  Subscribeto regOnto ROS topic to Get the rtList
8.  for each item ∈ rtList do
9.  if (cmd_velis in item ) then
10. Publishlinear(x, y, z) and angular(x, y, z) on this ROS topic
11. Set the stopping time based on the distance and velocity
12. else if (cmd_vel_muxis in item ) then
13. Publishlinear(x, y, z) and angular(x, y, z) on this ROS topic
14. Set the stopping time based on the distance and velocity
15. else
16. Some error in input or rtList
17. endif
18. endfor
19. end
```

Fig. 3.18 Level 01 Interpretation Algorithm

Figure 3.19 represents the level 02 interpretation algorithm where all robots move to the given new position with the given velocity from the current position facing the new orientation.

Figure 3.20 represents the level 03 interpretation algorithm where all robots move to the given goal with a given navigation path with an obstacle in the environment. Assume that the map is being created for each Robot using the scan topic. The map file is created and saved in the map server. This map is stored in the map server and is used by all robots to navigate their path. We need to maintain a separate *amcl* (Adaptive Monte Carlo Localisation) launch file and *move base* launch file for each Robot. In the *amcl* launch file, we need to set Robot-specific ROS topic (eg.*scan*, *odometry*, *initialpose* and *particlecloud*) for each Robot for the localization. Then we need to remap the Robot specific ROS topic (eg.*cmd vel*, *goal*, *odem*, *local plan*, *global plan* ,*footprint* and *costmap*) for the *move base* node for each Robot and store them in the launch file for the Movement of the Robot. The namespace avoids conflict with the same name with different robots in the ROS environment.

```
Level02_Interpretation(cmd, ipList, ontology)
1.  Start
2.  Create a ROS node as "lev02Intr" //Rest of the codes execute by this node.
3.  Get the new position and velocity from the cmd
4.  for each ip ∈ ipList do
5.  Connect with running robot using the ip.
6.  Subscribeto "initPosOnto" ROS topic to get the current position and orientation
7.  Set the values for linear(x, y, z)and angular(x, y, z)
8.  Set the values for new position(x, y, z) and orientation(x, y, z, w)
9.  Calculate the new orientation using new position and current position
10. Add delay time using sleep() to avoid the collision of Robots
11. Subscribeto regOnto ROS topic to Get the rtList
12. for each item ∈ rtList do
13. if (cmd_velis in item ) then
14. Publishlinear(x, y, z) and angular(x, y, z) on this ROS topic 15.   else if
    (cmd_vel_muxis in item ) then
16.     Publishlinear(x, y, z) and angular(x, y, z) on this ROS topic
17.     else
18.     Some error in input or rtList
19.     endif
20.     endfor
21.     end
```

Fig. 3.19 Level02 Interpretation Algorithm

3.8 Schedule Management

In our solution, we have assigned scheduled work and location for each Robot for a given time slot. The Robot can execute user instructions only if it is a free time slot; otherwise, the Robot needs to complete the allocated task. The CPE can publish or subscribe to the relevant values for each ROS topic. Each heterogeneous Robot has given a specific goal ($G_{i,j}$) or position to move with specific allocated work ($T_{i,j}$) based on the given time allocation as shown in the Table 3.3. According to the given time slot, the location to move (Goal) and task to be completed for each Robot is displayed in the Goal and Task Scheduling table.

3.9 Navigation Management

Autonomous navigation of the Robot is one of the main research areas in Robotic programming. ROS is implemented to work with the navigation stack that is used to easily navigate from one location to another by hiding most of the complex tasks in autonomous robot navigation. Navigation can be implemented using the *ROS topics, message formats* and *shapes of foot print* of the Robot and selecting the relevant values for the ROS topics for each Robot. Odometry and sensor information were used as main inputs for the ROS navigational stack then it generated the corresponding velocity for Table 3.2 Experiment Details

Level of the experiment	Experiment details
Level 01	Move forward all robots to 50m with the velocity of 10ms-1 from the current position in the current orientation.
Level 02	Move all robots to a specific position by facing to given position with given velocity with time gap to avoid the robot collision.
Level 03	Move all robots to a specific goal by facing to given position with a given velocity using separate navigation path for each Robot.

```
Level03_Interpretation(cmd, ipList, ontology, goals, map, amclList, movebaseList)
1.  Start
2.  Create a ROS node as "lev03Intr" //Rest of the codes execute by this node.
3.Subscribeto "initPosOnto" ROS topic to get the current position and orientation
4.          for each ip ∈ ipList do
5.          Connect with running robot using the ip.
6.          Get the current position and orientation from initPosOnto" ROS topic
7.          Calculate the new orientation using new goal and current position
8.          Subscribeto regOnto ROS topic to Get the rtList
9.          Publishthe pgm and yaml file on the ROS topic /map.
10.         for each item ∈ rtList do
11.         if (scan is in item ) then
12.         Remap and update the amcl launch file
13.         Remap and update the move_base launch file
14.         else if (odometryis in item ) then
15.         Remap and update the amcl launch file
16.         Remap and update the move_base launch file
17.         else if ……
18.         //Similarly we need to map initialpose, particlecloud,
19.         //cmd_vel, goal, odem, local_plan, global_plan footprint and
20.         endif
21.         Create the single launch file and execute
22.         endfor
23.         end
```

Fig. 3.20 Level03 Interpretation Algorithm

the mobile base. According to the ROS specification, we can find that the mobile base is controlled by *xisvelocity,yisvelocity,andthetaisvelocity*, and a 2D planer laser is mounted on the mobile base. The navigation is successful on the square-shaped robots Robotics (n.d.).

Table 3.3 General Goal and Task Scheduling Table

	Time slot 1	Time slot 2	Time slot 3	Time slot 4
Robot Name	$t_0 - t_1$	$t_1 - t_2$	$t_2 - t_3$	$t_3 - t_4$
R_1	$Goal_{1,1} + Task_{1,1}$	$Goal_{1,2} + Task_{1,2}$	$Goal_{1,3} + Task_{1,3}$	$Goal_{1,4} + Task_{1,4}$
R_2	$Goal_{2,1} + Task_{2,1}$	$Goal_{2,2} + Task_{2,2}$	$Goal_{2,3} + Task_{2,3}$	$Goal_{2,4} + Task_{2,4}$
R_3	$Goal_{3,1} + Task_{3,1}$	$Goal_{3,2} + Task_{3,2}$	$Goal_{3,3} + Task_{3,3}$	$Goal_{3,4} + Task_{3,4}$

The map server was used to store the created map file. All heterogeneous service robots used the map stored in the map server to navigate obstacles from one location to another. amcl (Adaptive Monte Carlo Localization) file and *move_base* file for each Robot were maintained as launch files to localize and move the Robot in the given environment. (eg.ROS scan, ROS odometry, ROS initial pose and ROS particle cloud) topics were used in the amcl launch file for each Robot for the localization. (eg.ROS topic cmd vel, ROS topic goal, ROS topic odem, ROS topic local plan, ROS topic global plan ,ROS topic footprint.) were used for remapping the ROS topic move base node for each Robot.

3.10 Thread Management

Since we need to control and coordinate multiple robots simultaneously, threads can be used to complete the task efficiently. Furthermore, a thread is a lightweight process inside a process. Therefore, concurrency can be developed using the threads quickly. The Thread library in Python implements multiple threads in our application. These threads can complete each task independently.

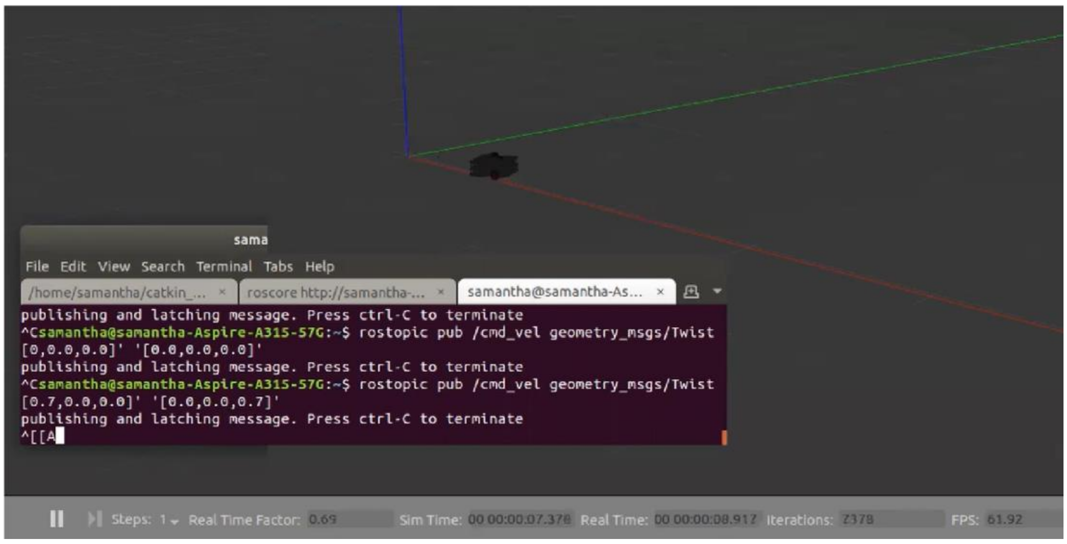
3.11 ROS Implementation

ROS Melodic Morenia distribution was installed in Ubuntu 18.04 LTS to complete the experiments with TurtleBot, Husky and TiaGO robots. We have installed the DesktopFull Install version in Ubuntu. Then, catkin make command is used to create the workspace for our project. It has created a CMakeLists.txt file. catkin _create _pkg command is used to create different ROS packages for our application. We have created several launch files to execute the ROS applications. There is a command named as roslauch to launch the ROS program. The running program can be interrupted by pressing the ctrl+c command in key bord. One of the main errors we received was the launch file permission error. However, it can be eliminated easily using the chmod command in ubuntu operating system. The roscore must be executed first to execute all the other processes. The roscore is the primary process that manages all other processes executed on the ROS environment.

CHAPTER 4

Evaluation and Results

We have conducted the experiments with Web interface I to V for simple instructions and measured the response time of the robot start and stop with the web interface. The initial experiment was conducted without the web interface. We have used the following notation for our experiments as shown below table 3.1. All experiments were completed in a simulation environment with Gazebo. The researchers have shown that the system developed with the Gazebo environment can be easily ported to the real robots without any changes to the original codes (Takaya et al. 2016*a*). Therefore, all our developed codes can be executed on real robots without any modifications.



```
publishing and latching message. Press ctrl-C to terminate
^Csamantha@samantha-Aspire-A315-57G:~$ rostopic pub /cmd_vel geometry_msgs/Twist
[0,0.0,0.0]' '[0.0,0.0,0.0]'
publishing and latching message. Press ctrl-C to terminate
^Csamantha@samantha-Aspire-A315-57G:~$ rostopic pub /cmd_vel geometry_msgs/Twist
[0.7,0.0,0.0]' '[0.0,0.0,0.7]'
publishing and latching message. Press ctrl-C to terminate
^[[A
```

sama
File Edit View Search Terminal Tabs Help
/home/samantha/catkin_...
roscore http://samantha-...
samantha@samantha-As...
Steps: 1
Real Time Factor: 0.69
Sim Time: 00 00:00:07.378
Real Time: 00 00:00:08.917
Iterations: 7378
FPS: 61.92

Fig. 4.1 Single Robot Interaction without Web Interface

4.1 Experiment 01: Single Robot Interaction with simple instruction without using the web interface.

Initially, the authors completed the experiment with a single robot without using the web interface in the Gazebo simulator with TurtleBot3. The authors have issued instructions to move the robot forward and move in a circle using the terminal interface with the rostopic pub command. We have evaluated the average response time of the robot for a Table 4.1 Single Robot Average Start/Stop Response Time Without Web Interface

$StartResponse(s)$	U_x^s= $0.5ms^{-1}$	U_x^s= $1.0ms^{-1}$	U_x^s= $1.5ms^{-1}$
$\omega_z^s = 0.0ms^{-1}$	0.871	0.807	0.787
$\omega_z^s = 0.5ms^{-1}$	0.657	0.541	0.531
$\omega_z^s = 1.0ms^{-1}$	0.561	0.512	0.499
$\omega_z^s = 1.5ms^{-1}$	0.511	0.501	0.476
$StopResponse(s)$	U_x^s= $0.5ms^{-1}$	U_x^s= $1.0ms^{-1}$	U_x^s= $1.5ms^{-1}$
$\omega_z^e = 0.0ms^{-1}$	1.211	1.728	2.161
$\omega_z^e = 0.5ms^{-1}$	1.039	1.631	1.981
$\omega_z^e = 1.0ms^{-1}$	1.001	1.431	1.871
$\omega_z^e = 1.5ms^{-1}$	0.988	1.181	1.761

Table 4.2 Testing to Determine the Constant c_1 in Equations 4.1

$StartResponse(s)$	U_{xs} in ms^{-1}	ω_{zs} in ms^{-1}	$Test01$ (c_1 = 0.23)	$Test02$ (c_1 = 0.24)	$Test03$ (c_1 = 0.25)
0.871	0.5	0.0	0.851	0.871	0.891
0.657	0.5	0.1	0.622	0.632	0.642
0.561	0.5	1.0	0.363	0.553	0.559
0.511	0.5	1.5	0.509	0.514	0.519
0.807	1.0	0.0	0.621	0.631	0.641
0.541	1.0	0.1	0.545	0.552	0.558
0.512	1.0	1.0	0.508	0.513	0.518
0.501	1.0	1.5	0.486	0.490	0.494

start and stop instructions. We have conducted experiments with different robot linear and angular speeds for a start and stop instructions. The experiment results will be displayed in table 4.1. The interaction with TurtleBot3 with the terminal without using a Web interface is shown in Figure 4.1. The response delay for a start and stop of the robot is represented by the equation 4.1and 4.2 where $R_{s,d}^{start}$ and $R_{s,d}^{stop}$represents the single robot delay at the start and stop respectively, $\tau_{d,os}$ represents the delay in system call execution in Operating System, $\tau_{d,ROS}$ is used to represents the delay in communicating with ROS topics and c_1, c_2 are constants.

$$R_{s,d}^{start} = \tau_{d,os} + \tau_{d,ROS} + \frac{c_1}{\{U_x^s + \omega_z^s\}} \tag{4.1}$$

$$R_{s,d}^{stop} = \tau_{d,os} + \tau_{d,ROS} + c_2\{U_x^s + \omega_z^s\} \tag{4.2}$$

We have determined the constant c_1 using the experiment results, assuming the values of $\tau_{d,os}$ and $\tau_{d,ROS}$ are constant for the given angular and linear speeds. The validation of the constant c_1 was completed with three Testing named Test 01, Test 02 and Test 01 with three different constant values for $c_1 = 0.23$, $c_1 = 0.24$ and $c_1 = 0.25$. Three Table 4.3 Pearson value (r) for each Testing

$TestName$	$PearsonV\,alue(r)$
$Test01$ ($c_1 = 0.23$)	0.90611
$Test02$ ($c_1 = 0.24$)	0.90618
$Test03$ ($c_1 = 0.25$)	0.90624

Table 4.4 Testing to Determine the Constant c_2 in Equations 4.2

$StartResponse(s)$	U_{xs} in ms^{-1}	ω_{zs} in ms^{-1}	$Test01$ ($c_2 = 0.84$)	$Test02$ ($c_2 = 0.85$)	$Test03$ ($c_2 = 0.86$)
1.211	0.5	0.0	1.156	1.161	1.166
1.039	0.5	0.1	1.586	1.596	1.606
1.001	0.5	1.0	2.027	2.042	2.057
0.988	0.5	1.5	2.457	2.477	2.497
1.228	1.0	0.0	1.576	2.020	1.596
1.332	1.0	0.1	2.005	0.552	2.035
1.431	1.0	1.0	2.426	2.446	2.466
1.811	1.0	1.5	2.856	2.881	2.906

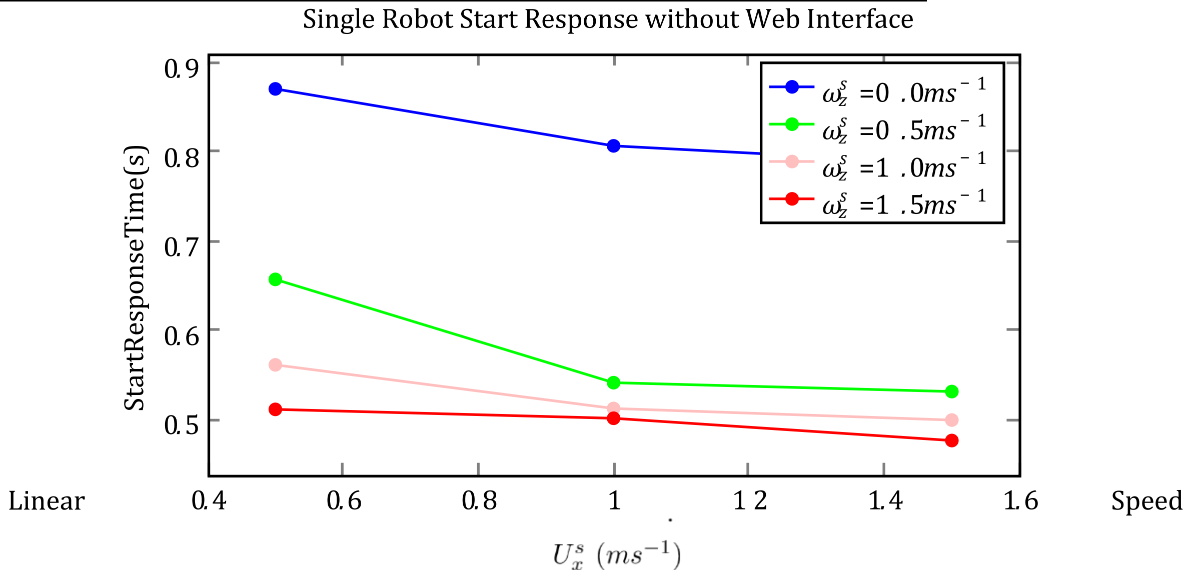
test results were calculated using the equation we have derived with the use of constant value c_1 we have calculated from the initial experimental results as shown in Table 4.2. We have used the person() value to find the correlation between the value generated from the equation and the experiment response value, as shown in Table 4.3. The pearson() value is higher for the value $c_1 = 0.25$. Therefore, we can select $c_1 = 0.25$ as the more accurate value for the equation 4.1. $\tau_{d,os}$ represents the delay in system call execution in Operating System, $\tau_{d,ROS}$ represents the delay in communicating with ROS topics and c_1, c_2 are constants.

Figure 4.2 represents the robot's average start and stop response time for each instruction. The average start response time gradually decreases when the linear and angular speed increases, while the average stop time increases when the linear and angular speed increases.

We have determined the constant c_2 using the experiment results, assuming the values of $\tau_{d,os}$ and $\tau_{d,ROS}$ are constant for the given angular and linear speeds. The validation of the constant c_2 was completed with three Testing named Test 01, Test 02 and Test 03 with three different constant values for $c_2 = 0.84$, $c_2 = 0.85$ and $c_2 = 0.86$. Three

Table 4.5 Pearson value (r) for each Testing

TestName	*PearsonV alue*(*r*)
*Test*01 ($c_2 = 0.23$)	0.52004
*Test*02 ($c_2 = 0.24$)	0.52022
*Test*03 ($c_2 = 0.25$)	0.52039



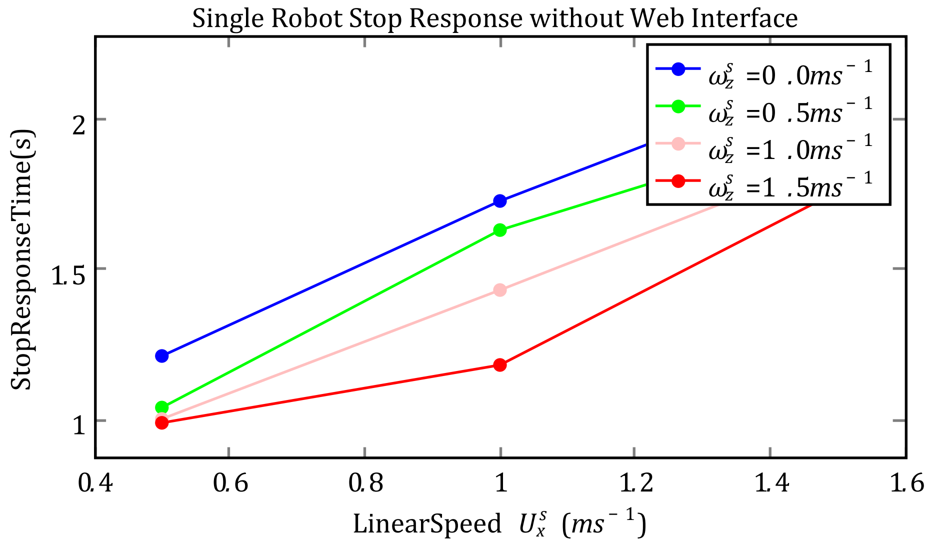


Fig. 4.2 Single Robot Interaction without Web Interface

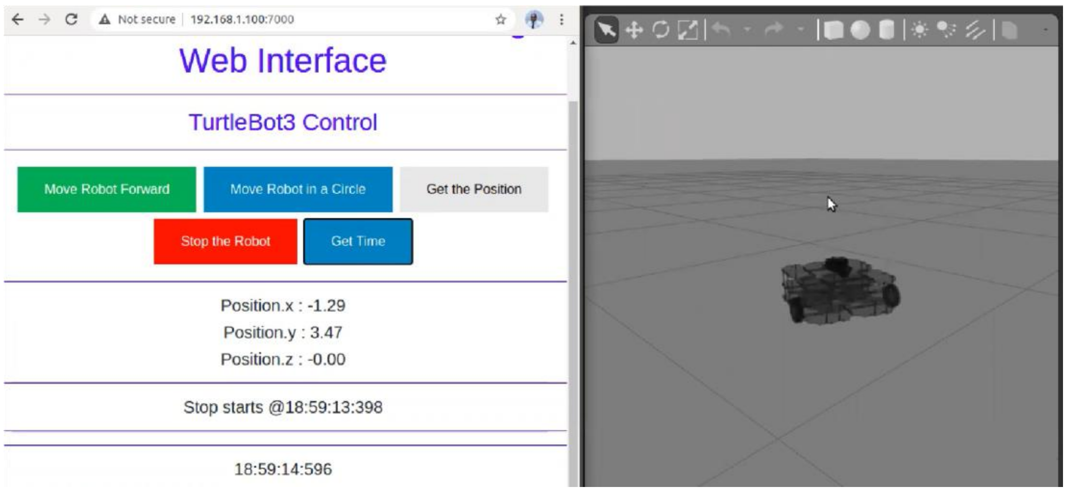
test results were calculated using the equation we have derived with the use of constant value c_2 we have calculated from the initial experimental results as shown in Table 4.4. We have used the person() value to find the correlation between the value generated from the equation and the experiment response value, as shown in Table 4.5. The pearson() value is higher for the value $c_2 = 0.86$. Therefore, we can select $c_2 = 0.86$ as the more accurate value for the equation 4.2. $\tau_{d,os}$ represents the delay in system call execution in Operating System, $\tau_{d,ROS}$ represents the delay in communicating with ROS topics and c_1, c_2 are constants.

4.2 Experiment 02: Single Robot Interaction with simple instruction with web interface without autonomous robot registration.

The authors developed the web interface to interact with the robot using the ROS Bridge Server and JavaScript library ROS Web Tools (http://robotwebtools.org/). The authors have issued instructions to move the robot forward and in a circle using the buttons provided in the Web interface with the robot. We have evaluated the average response time of the robot for a start and stop instructions. We have conducted experiments with different robot linear and angular speeds for a start and stop instructions. The experiment results will be displayed below in table 4.6. The interaction with TurtleBot3 with the terminal with Web interface is shown in the following Figure 4.3. The response delay for a start and stop of the robot is represented by the equation 4.3and 4.4 where $R_{s,d}^{start}$ and $R_{s,d}^{stop}$represents the single robot delay at the start and stop respectively, $\tau_{d,Web}$ represents the delay in communication through Web interface, $\tau_{d,ROS}$ is used to represents the delay in communicating with ROS topics and c_3, c_4 are constants.

$$R_{s,d}^{start} = \tau_{d,web} + \tau_{d,ROS} + \frac{c_3}{\{U_x^s + \omega_z^s\}} \tag{4.3}$$

$$R_{s,d}^{stop} = \tau_{d,web} + \tau_{d,ROS} + c_4\{U_x^s + \omega_z^s\} \tag{4.4}$$



Position.x : -1.29
Position.y : 3.47
Position.z : -0.00
Stop starts @18:59:13:398
18:59:14:596

Fig. 4.3 Single Robot Interaction with Web Interface

Figure 4.4 represents the robot's average start and stop response time for each instruction. The average start response time gradually decreases when the linear and angular speed increases, while the average stop time increases when the linear and angular speed increases. According to the analysis, the authors have identified that web communication is slightly faster than communication through the terminal.

We have determined the constant c_3 using the experiment results, assuming the valTable 4.6 Single Robot Average Start/Stop Response Time With Web Interface

$StartResponse(s)$	U^s_x= $0.5ms^{-1}$	U^s_x= $1.0ms^{-1}$	U^s_x= $1.5ms^{-1}$
$\omega^s_z = 0.0ms^{-1}$	0.811	0.789	0.766
$\omega^s_z = 0.5ms^{-1}$	0.753	0.732	0.699
$\omega^s_z = 1.0ms^{-1}$	0.611	0.601	0.544
$\omega^s_z = 1.5ms^{-1}$	0.571	0.577	0.501
$StopResponse(s)$	U^s_x= $0.5ms^{-1}$	U^s_x= $1.0ms^{-1}$	U^s_x= $1.5ms^{-1}$
$\omega^e_z = 0.0ms^{-1}$	1.031	1.402	1.981
$\omega^e_z = 0.5ms^{-1}$	1.001	1.267	1.812
$\omega^e_z = 1.0ms^{-1}$	0.981	1.101	1.602
$\omega^e_z = 1.5ms^{-1}$	0.911	0.999	1.201

Table 4.7 Testing to Determine the Constant c_3 in Equations 4.3

$StartResponse(s)$	U_{xs} in ms^{-1}	ω_{zs} in ms^{-1}	$Test01$ (c_3 = 0.24)	$Test02$ (c_3 = 0.25)	$Test03$ (c_3 = 0.26)
0.811	0.5	0.0	0.811	0.831	0.851
0.753	0.5	0.1	0.585	0.595	0.605
0.611	0.5	1.0	0.515	0.521	0.528
0.571	0.5	1.5	0.485	0.490	0.495
0.789	1.0	0.0	0.571	0.581	0.591
0.732	1.0	0.1	0.505	0.552	0.518
0.601	1.0	1.0	0.475	0.511	0.485
0.577	1.0	1.5	0.461	0.465	0.469

Table 4.8 Pearson value (r) for each Testing

$TestName$	$PearsonV alue(r)$
$Test01$ (c_3 = 0.24)	0.76559
$Test02$ (c_3 = 0.25)	0.76737
$Test03$ (c_3 = 0.26)	0.76901

Single Robot Start Response with Web Interface

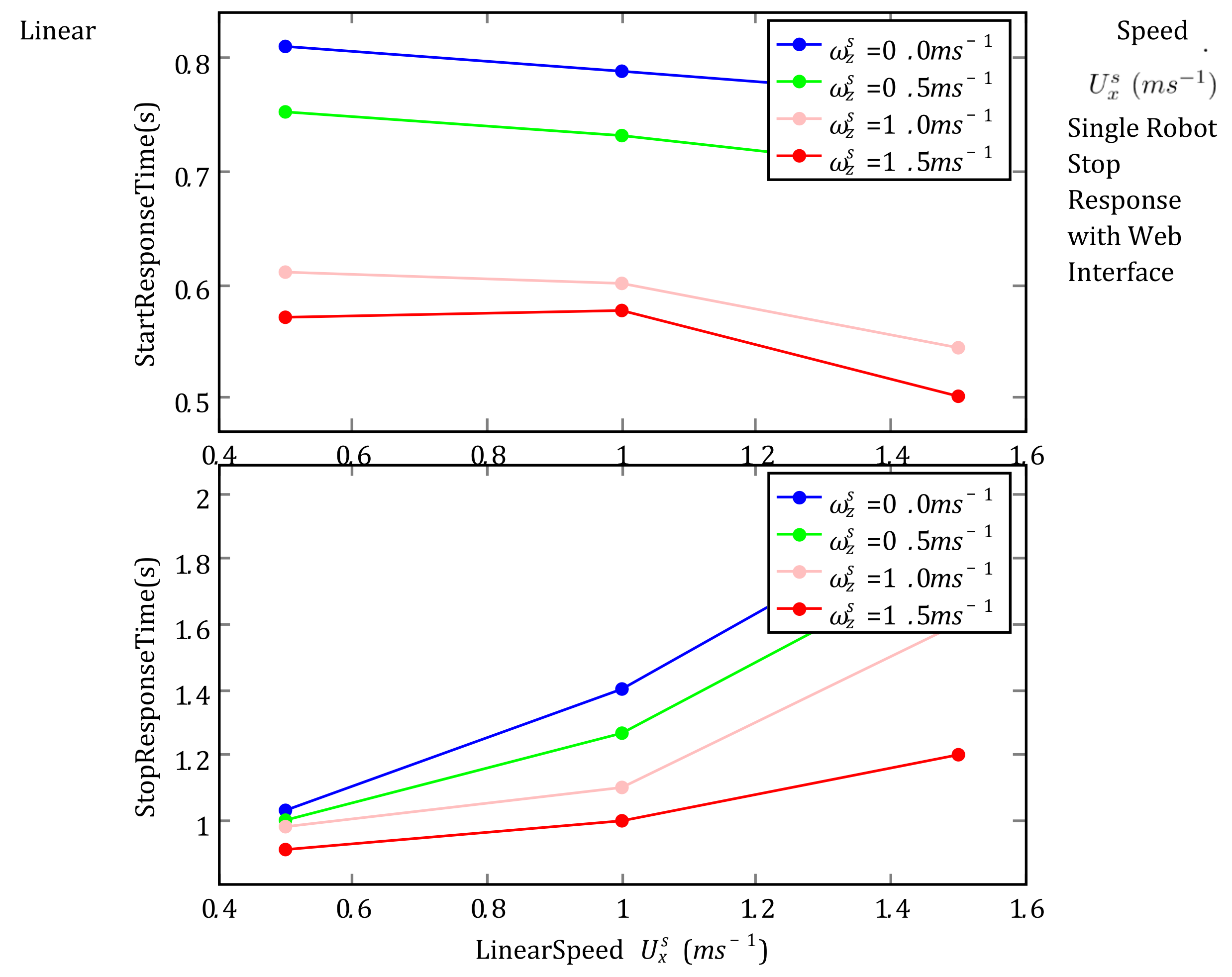


Fig. 4.4 Single Robot Interaction with Web Interface

ues of $\tau_{d,Web}$ and $\tau_{d,ROS}$ are constant for the given angular and linear speeds. The validation of the constant c_3 was completed with three Testing named Test 01, Test 02 and Test 03 with three different constant values for $c_3 = 0.24$, $c_3 = 0.25$ and $c_3 = 0.26$. Three test results were calculated using the equation we have derived with the use of constant value c_3 we have calculated from the initial experimental results as shown in Table 4.7. We have used the person() value to find the correlation between the value generated from the equation and the experiment response value, as shown in Table 4.8. The pearson() value is higher for the value $c_3 = 0.26$. Therefore, we can select $c_3 = 0.26$ as the more accurate value for the equation 4.3.$\tau_{d,Web}$ represents the delay in communication through the Web interface, $\tau_{d,ROS}$ is used to represents the delay in communicating with ROS topics and c_3, c_4 are constants.

We have determined the constant c_4 using the experiment results, assuming the valTable 4.9 Testing to Determine the Constant c_4 in Equations 4.4

$StartResponse(s)$	U_{xs} in ms^{-1}	ω_{zs} in ms^{-1}	$Test01$ (c_4 = 0.95)	$Test02$ (c_4 = 0.96)	$Test03$ (c_4 = 0.97)

1.031	0.5	0.0	1.031	1.036	1.041
1.001	0.5	0.1	1.516	1.526	1.536
0.981	0.5	1.0	2.001	2.016	2.031
0.911	0.5	1.5	2.486	2.507	2.526
1.102	1.0	0.0	1.506	1.516	1.526
1.267	1.0	0.1	1.991	2.006	2.021
1.101	1.0	1.0	2.476	2.496	2.516
1.981	1.0	1.5	2.961	2.986	3.011

Table 4.10 Pearson value (r) for each Testing

$TestName$	$PearsonV\,alue(r)$
$Test01$ ($c_4 = 0.95$)	0.57113
$Test02$ ($c_4 = 0.96$)	0.57116
$Test03$ ($c_4 = 0.97$)	0.57119

ues of $\tau_{d,Web}$ and $\tau_{d,ROS}$ are constant for the given angular and linear speeds. The validation of the constant c_4 was completed with three Testing named Test 01, Test 02 and Test 03 with three different constant values for $c_4 = 0.95$, $c_4 = 0.96$ and $c_4 = 0.97$. Three test results were calculated using the equation we have derived with the use of constant value c_4 we have calculated from the initial experimental results as shown in Table 4.9. We have used the person() value to find the correlation between the value generated from the equation and the experiment response value, as shown in Table 4.10. The pearson() value is higher for the value $c_4 = 0.97$. Therefore, we can select $c_4 = 0.97$ as the more accurate value for the equation 4.4.$\tau_{d,ROS}$ is used to represents the delay in communicating with ROS topics and c_3, c_4 are constants.

Table 4.11 Pearson value (r) for Experiment 01 and Experiment 02 Comparison

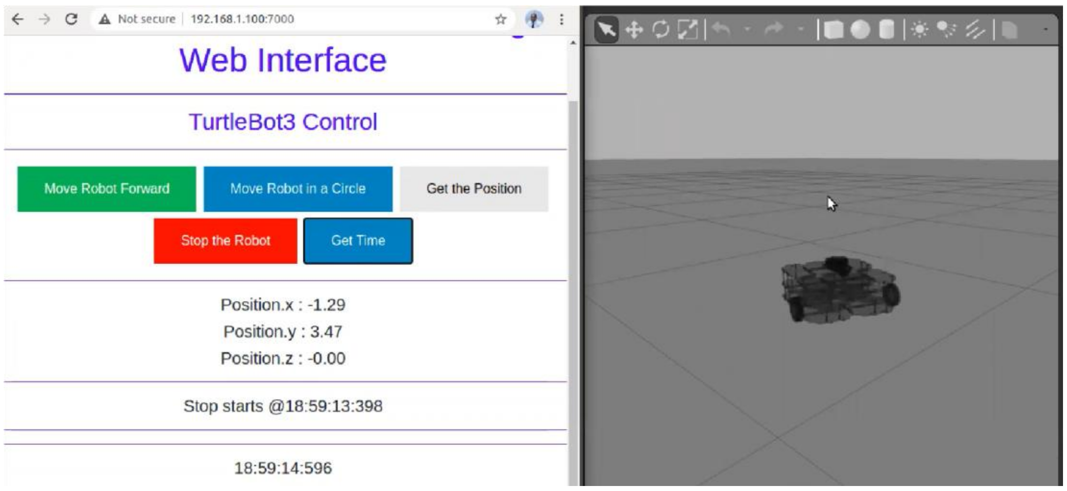
$ExperimentDetails$	$PearsonV\,alue(r)$
Start Response with$U_x^s = 0.5ms^{-1}$	0.93188
Start Response with$U_x^s = 1.0ms^{-1}$	0.81706
Start Response with$U_x^s = 1.5ms^{-1}$	0.82985
Stop Response with$U_x^s = 0.5ms^{-1}$	0.77143
Stop Response with$U_x^s = 1.0ms^{-1}$	0.97121
Stop Response with$U_x^s = 1.5ms^{-1}$	0.94949

4.3 Experiment 03: Single Robot Interaction with simple instruction with a web interface with autonomous robot registration.

The Robot Registration Engine was developed to collect all robot details, including all ROS topics necessary to subscribe and publish. The ROS Topic Identification Algorithm was developed to select the relevant ROS topics for each action defined in the user instruction. We have evaluated the average response time of the robot for a start and stop instructions. We have conducted experiments with different robot linear and angular speeds for a start and stop instructions. The experiment results will be displayed below in table 4.12. The interaction with TurtleBot3 with the terminal with Web interface is shown in Figure 4.6. The response delay for a start and stop of the robot is represented by the equation 4.5and 4.6 where $R_{s,d}{}^{start}$ and $R_{s,d}{}^{stop}$represents the single robot delay at the start and stop respectively, $\tau_{d,Web}$ represents the delay in communication through Web interface, $\tau_{d,ROS}$ is used to represents the delay in communicating with ROS topics, $\tau_{d,RT}$ represents the delay in ROS topic identification and c_5, c_6 are constants.

$$R_{s,d}^{start} = \tau_{d,web} + \tau_{d,ROS} + \tau_{d,RT} + \frac{c_5}{\{U_x^s + \omega_z^s\}} \tag{4.5}$$

$$R_{s,d}^{stop} = \tau_{d,web} + \tau_{d,ROS} + \tau_{d,RT} + c_6\{U_x^s + \omega_z^s\} \tag{4.6}$$



Web Interface

TurtleBot3 Control

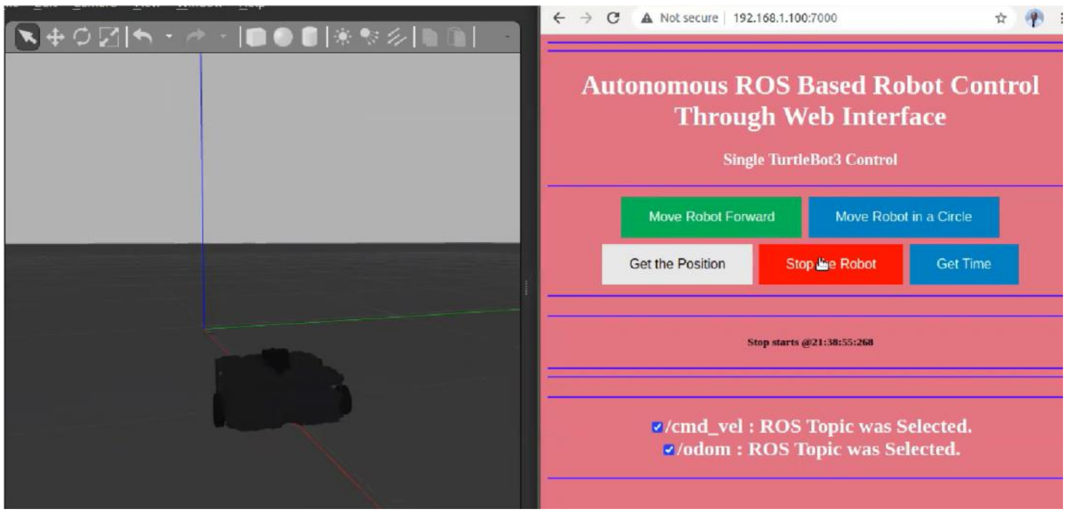
Position.x : -1.29
Position.y : 3.47
Position.z : -0.00

Stop starts @18:59:13:398

18:59:14:596

Fig. 4.5 Single Robot Interaction with Web Interface

Figure 4.7 represents the robot's average start and stop response time for each instruction. The average start response time gradually decreases when the linear and angular speed increases, while the average stop time increases when the linear and angular speed increases. According to the analysis, authors have identified that autonomous robot communication is slightly slower than communication through the Web without


Not secure | 192.168.1.100:7000
Autonomous ROS Based Robot Control Through Web Interface
Single TurtleBot3 Control
Move Robot Forward
Move Robot in a Circle
Get the Position
Stop the Robot
Get Time

Stop starts @21:38:55:268

☑/cmd_vel : ROS Topic was Selected.
☑/odom : ROS Topic was Selected.

Fig. 4.6 Single Robot Interaction with Web Interface Auto Registration

Table 4.12 Single Robot Average Start/Stop Response Time With Web Interface Autonomous

$StartResponse(s)$	U_x^s= $0.5ms^{-1}$	U_x^s= $1.0ms^{-1}$	U_x^s= $1.5ms^{-1}$
$\omega_z^s = 0.0ms^{-1}$	1.011	1.001	0.981
$\omega_z^s = 0.5ms^{-1}$	1.001	0.987	0.956
$\omega_z^s = 1.0ms^{-1}$	0.987	0.872	0.789
$\omega_z^s = 1.5ms^{-1}$	0.861	0.761	0.712
$StopResponse(s)$	U_x^s= $0.5ms^{-1}$	U_x^s= $1.0ms^{-1}$	U_x^s= $1.5ms^{-1}$
$\omega_z^e = 0.0ms^{-1}$	1.345	1.765	2.552
$\omega_z^e = 0.5ms^{-1}$	1.241	1.451	2.222
$\omega_z^e = 1.0ms^{-1}$	1.109	1.431	1.988
$\omega_z^e = 1.5ms^{-1}$	1.011	1.344	1.765

autonomous registration.

We have determined the constant c_5 using the experiment results assuming the values of $\tau_{d,Web}$, $\tau_{d,Web}$ and $\tau_{d,RT}$ are constant for the given angular and linear speeds. The validation of the constant c_5 was completed with three Testing named Test 01, Test 02 and Test 03 with three different constant values for $c_5 = 0.11$, $c_5 = 0.12$ and $c_5 = 0.13$. Three test results were calculated using the equation we have derived using the constant value c_5 we calculated from the initial experimental results as shown in Table 4.13. We have used the person() value to find the correlation between the value generated from the equation and the experiment response value, as shown in Table 4.14. The pearson() value is higher for the value $c_5 = 0.13$. Therefore, we can select $c_5 = 0.13$ as the more accurate value for the equation 4.5.$\tau_{d,Web}$ represents the delay in communication through the Web interface, $\tau_{d,ROS}$ is used to represent the delay in communicating with ROS topics, $\tau_{d,RT}$ represents the delay in ROS topic identification and c_5, c_6 are constants.

We have determined the constant c_6 using the experiment results assuming the valTable 4.13 Testing to Determine the Constant c_5 in Equations 4.5

$StartResponse(s)$	U_{xs} in ms^{-1}	ω_{zs} in ms^{-1}	$Test01$ (c_5 = 0.11)	$Test02$ (c_5 = 0.12)	$Test03$ (c_5 = 0.13)
1.011	0.5	0.0	1.181	1.201	1.221
1.001	0.5	0.1	1.081	1.091	1.101
0.987	0.5	1.0	1.054	1.061	1.067
0.861	0.5	1.5	1.046	1.051	1.056
1.001	1.0	0.0	1.071	1.081	1.091
0.987	1.0	0.1	1.044	1.051	1.057
0.872	1.0	1.0	1.036	1.041	1.046
0.761	1.0	1.5	1.035	1.039	1.043

Table 4.14 Pearson value (r) for each Testing

$TestName$	$PearsonValue(r)$
$Test01$ (c_5 = 0.11)	0.54836
$Test02$ (c_5 = 0.12)	0.55785
$Test03$ (c_5 = 0.13)	0.56558

Table 4.15 Testing to Determine the Constant c_6 in Equations 4.6

$StartResponse(s)$	U_{xs} in ms^{-1}	ω_{zs} in ms^{-1}	$Test01$ (c_6 = 1.115)	$Test02$ (c_6 = 1.16)	$Test03$ (c_6 = 1.17)
1.345	0.5	0.0	1.345	1.350	1.355
1.241	0.5	0.1	1.914	1.924	1.934
1.011	0.5	1.0	2.482	2.497	2.512
0.861	0.5	1.5	3.051	3.071	3.091
2.552	1.0	0.0	1.924	1.934	1.944
2.222	1.0	0.1	2.492	2.507	2.522
1.988	1.0	1.0	3.061	3.081	3.101
3.765	1.0	1.5	3.629	3.654	3.679

Table 4.16 Pearson value (r) for each Testing

$TestName$	$PearsonValue(r)$
$Test01$ (c_6 = 1.15)	0.47464
$Test02$ (c_6 = 1.16)	0.47460
$Test03$ (c_6 = 1.17)	0.47456

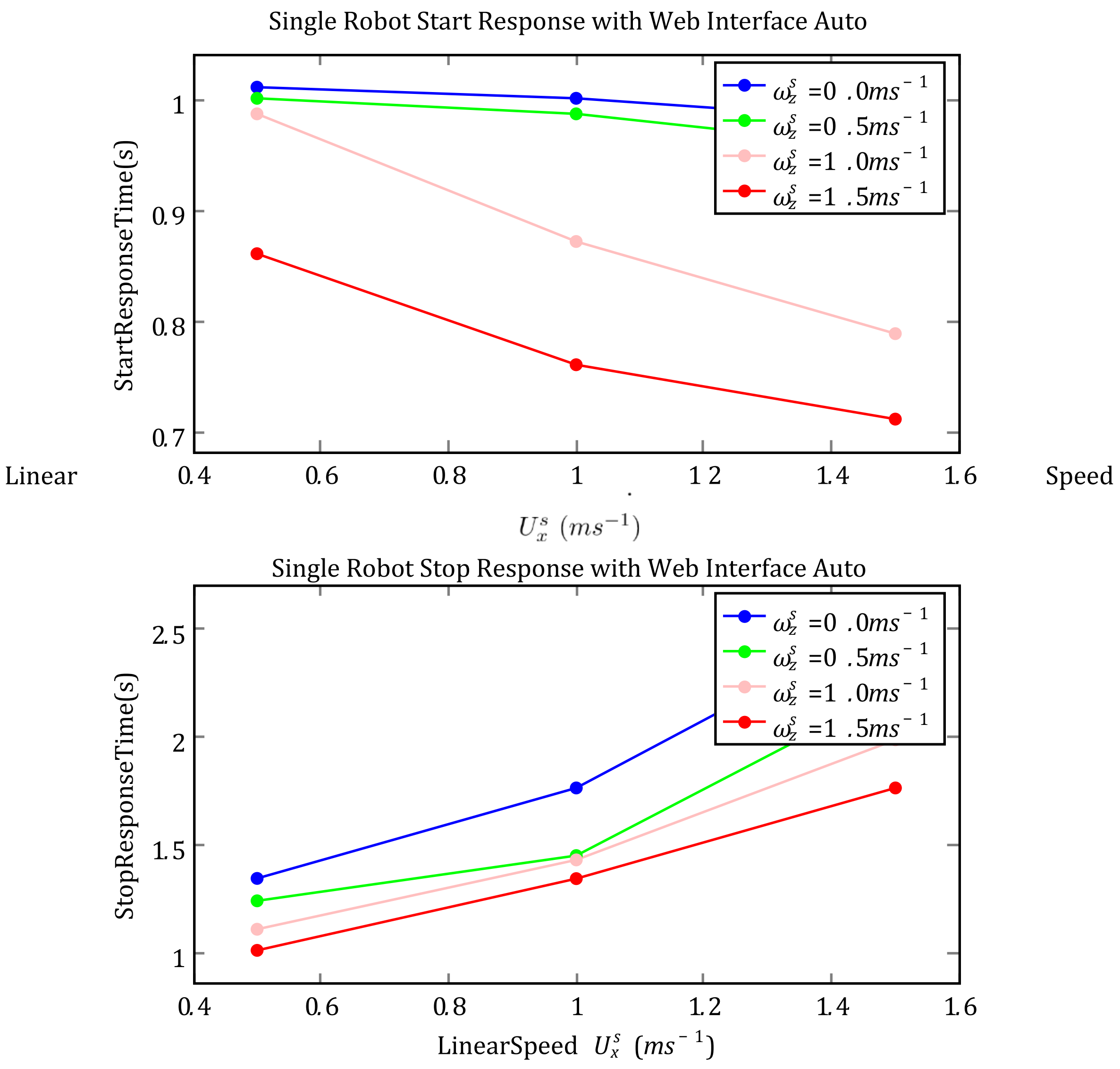


Fig. 4.7 Single Robot Interaction without Web Interface

ues of $\tau_{d,Web}$, $\tau_{d,Web}$ and $\tau_{d,RT}$ are constant for the given angular and linear speeds. The validation of the constant c_6 was completed with three Testing named Test 01, Test 02 and Test 03 with three different constant values for $c_6 = 1.15$, $c_6 = 1.16$ and $c_6 = 1.17$. Three test results were calculated using the equation we have derived using the constant value c_6 we calculated from the initial experimental results as shown in Table 4.15. We have used the person() value to find the correlation between the value generated from the equation and the experiment response value, as shown in Table 4.16. The pearson() value is higher for the value $c_6 = 1.17$. Therefore, we can select $c_6 = 1.17$ as the more accurate value for the equation 4.6.$\tau_{d,Web}$ represents the delay in communication through the Web interface, $\tau_{d,ROS}$ is used to represent the delay in communicating with ROS topics, $\tau_{d,RT}$ represents the delay in ROS topic identification and c_5, c_6 are constants.

Table 4.17 Pearson value (r) for Experiment 02 and Experiment 03 Comparison

ExperimentDetails	*PearsonV alue(r)*
Start Response with U_x^s= $0.5ms^{-1}$	0.86692
Start Response with U_x^s= $1.0ms^{-1}$	0.93539
Start Response with U_x^s= $1.5ms^{-1}$	0.98509
Stop Response with U_x^s= $0.5ms^{-1}$	0.82333
Stop Response with U_x^s= $1.0ms^{-1}$	0.91021
Stop Response with U_x^s= $1.5ms^{-1}$	0.89119

Table 4.18 Multiple Robots Average Start/Stop Response Time With Web Interface Autonomous

StartResponse(s)	U_x^s= $0.5ms^{-1}$	U_x^s= $1.0ms^{-1}$	U_x^s= $1.5ms^{-1}$
SingleRobot	1.011	1.001	0.981
TwoRobots	1.129	1.078	1.016
FourRobots	1.456	1.241	1.112
StopResponse(s)	U_x^s= $0.5ms^{-1}$	U_x^s= $1.0ms^{-1}$	U_x^s= $1.5ms^{-1}$
SingleRobot	1.345	1.765	2.552
TwoRobots	1.674	1.987	2.987
FourRobots	1.987	2.134	2.456

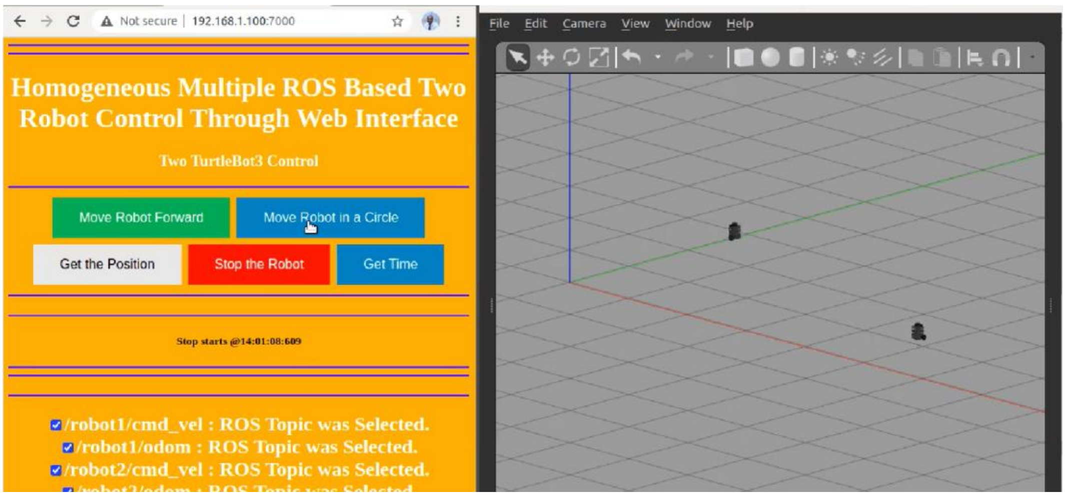
4.4 Experiment 04: Homogeneous Multiple Robot Interaction with simple instruction with a web interface with autonomous robot registration.

The authors have developed the launch file to create multiple robots in the same Gazebo environment. Initially, two TurtleBot robots were spawned in the empty Gazebo world at two different locations. The simple move instructions were issued to both robots simultaneously and evaluated the average response time for the start and stop instructions. The separate namespaces were used to identify each ROS topic for each robot. The first robot was named robot1, and the second was named robot2. The interaction with multiple two TurtleBot with the terminal with Web interface is shown in Figure

4.8. The response delay for a start and stop of the robot is represented by the equation 4.7and 4.8 where $R_{m,d}^{start}$ and $R_{m,d}^{stop}$represents the multiple robots delay at the start and stop respectively, $\tau_{d,Web}$ represents the delay in communication through the Web interface, $\tau_{d,ROS}$ is used to represents the delay in communicating with ROS topics, $\tau_{d,RT}$ represents the delay in ROS topic identification and c_7, c_8 are constants. α and β represent the number of robots in the environment. For example, when the number of robots is two, then α = 2 and β = 2 and When the number of robots is four, then α = 4 and β = 4.

$$R^{start}_{m,d} = \alpha\{\tau_{d,web} + \tau_{d,ROS} + \tau_{d,RT}\} + \frac{c_7}{\{U^s_x + \omega^s_z\}} \tag{4.7}$$

$$R^{stop}_{m,d} = \beta\{\tau_{d,web} + \tau_{d,ROS} + \tau_{d,RT}\} + c_8\{U^s_x + \omega^s_z\} \tag{4.8}$$



Homogeneous Multiple ROS Based Two Robot Control Through Web Interface

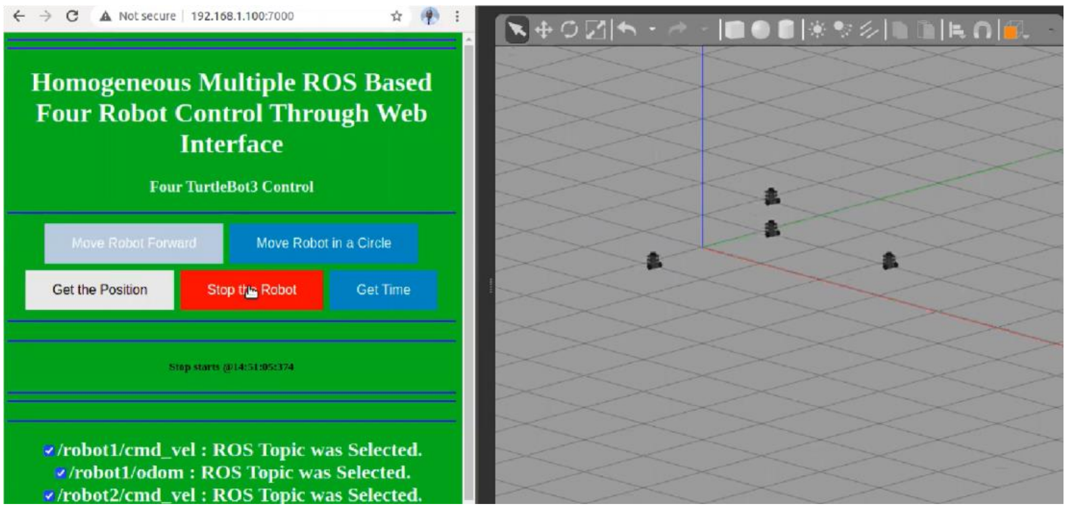
Two TurtleBot3 Control

Stop starts @14:01:08:609

✓/robot1/cmd_vel : ROS Topic was Selected.
✓/robot1/odom : ROS Topic was Selected.
✓/robot2/cmd_vel : ROS Topic was Selected.

Fig. 4.8 Multiple Two Robots Interaction with Web Interface Auto Registration

Secondly, the authors have spawned another four robots in the same Gazebo environment for the experiment. Separate namespaces were given for each robot to avoid conflicts with the same ROS topic. The simple move instructions were issued to both robots simultaneously and evaluated the average response time for the start and stop instructions. The experiment results will be displayed below in table 4.18. The interaction with multiple four TurtleBot with the terminal with Web interface is shown in the following Figure 4.9.



Homogeneous Multiple ROS Based Four Robot Control Through Web Interface

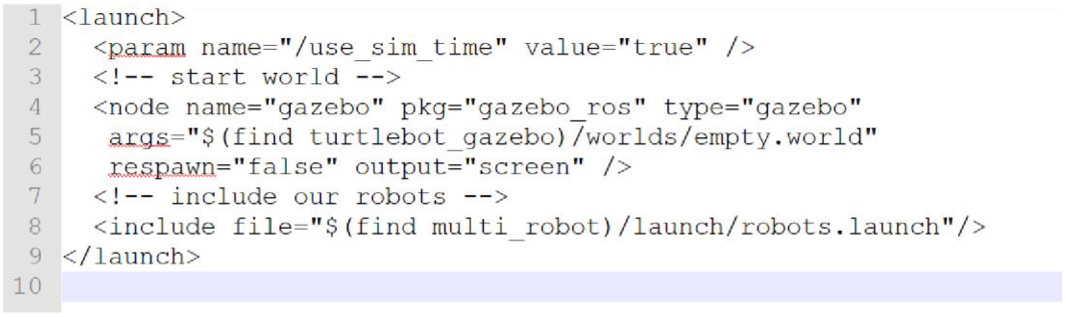
Four TurtleBot3 Control

Stop starts @14:51:05:374

✓/robot1/cmd_vel : ROS Topic was Selected.
✓/robot1/odom : ROS Topic was Selected.
✓/robot2/cmd_vel : ROS Topic was Selected.

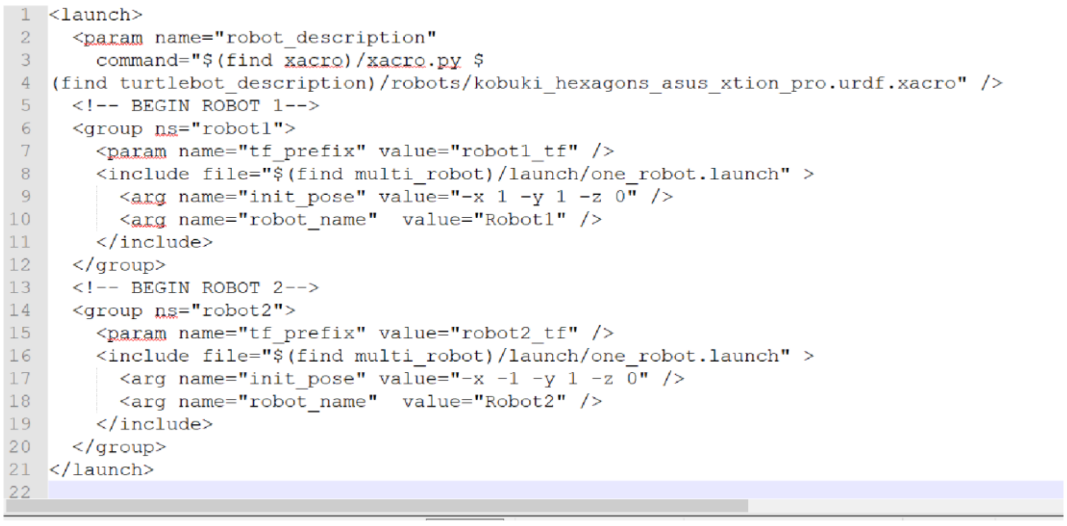
Fig. 4.9 Multiple Four Robots Interaction with Web Interface Auto Registration

The main launch file used to create two robots in the gazebo environment is shown in Figure 4.10. The launch file, which describes the robot's position and namespaces with robot description, are defined in Figure 4.11. Loading each robot is completed by another launch file named as described in Figure 4.12.



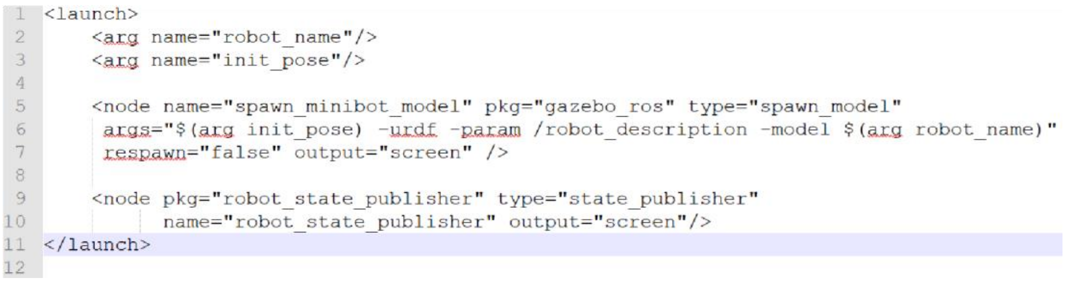
```
<launch>
  <param name="/use_sim_time" value="true" />
  <!-- start world -->
  <node name="gazebo" pkg="gazebo_ros" type="gazebo"
   args="$(find turtlebot_gazebo)/worlds/empty.world"
   respawn="false" output="screen" />
  <!-- include our robots -->
  <include file="$(find multi_robot)/launch/robots.launch"/>
</launch>
```

Fig. 4.10 Main Launch File Two Launch Two Robots



```
<launch>
  <param name="robot_description"
    command="$(find xacro)/xacro.py $
(find turtlebot_description)/robots/kobuki_hexagons_asus_xtion_pro.urdf.xacro" />
  <!-- BEGIN ROBOT 1-->
  <group ns="robot1">
    <param name="tf_prefix" value="robot1_tf" />
    <include file="$(find multi_robot)/launch/one_robot.launch" >
      <arg name="init_pose" value="-x 1 -y 1 -z 0" />
      <arg name="robot_name"  value="Robot1" />
    </include>
  </group>
  <!-- BEGIN ROBOT 2-->
  <group ns="robot2">
    <param name="tf_prefix" value="robot2_tf" />
    <include file="$(find multi_robot)/launch/one_robot.launch" >
      <arg name="init_pose" value="-x -1 -y 1 -z 0" />
      <arg name="robot_name"  value="Robot2" />
    </include>
  </group>
</launch>
```

Fig. 4.11 Launch file to describe Position and Robot Description



```
<launch>
    <arg name="robot_name"/>
    <arg name="init_pose"/>

    <node name="spawn_minibot_model" pkg="gazebo_ros" type="spawn_model"
     args="$(arg init_pose) -urdf -param /robot_description -model $(arg robot_name)"
     respawn="false" output="screen" />

    <node pkg="robot_state_publisher" type="state_publisher"
          name="robot_state_publisher" output="screen"/>
</launch>
```

Fig. 4.12 One Robot Launch File

Figure 4.13 represents the average start and stop response time for the single robot, two robots, and four for each instruction where linear speed is changed. However, the angular speed is kept constant to avoid collision among the robots. The average start response time gradually increases when the number of robots increases, while the average stop time increases when the number of robots increases.

We have determined the constant c_7 using the experiment results assuming the values of $\tau_{d,RT}$, $\tau_{d,Web}$ and $\tau_{d,ROS}$ are constant for the given angular and linear speeds.

The validation of the constant c_7 was completed with three Testing named Test 01, Test 02 and Test 03 with three different constant values for $c_7 = 0.022$, $c_7 = 0.032$ and

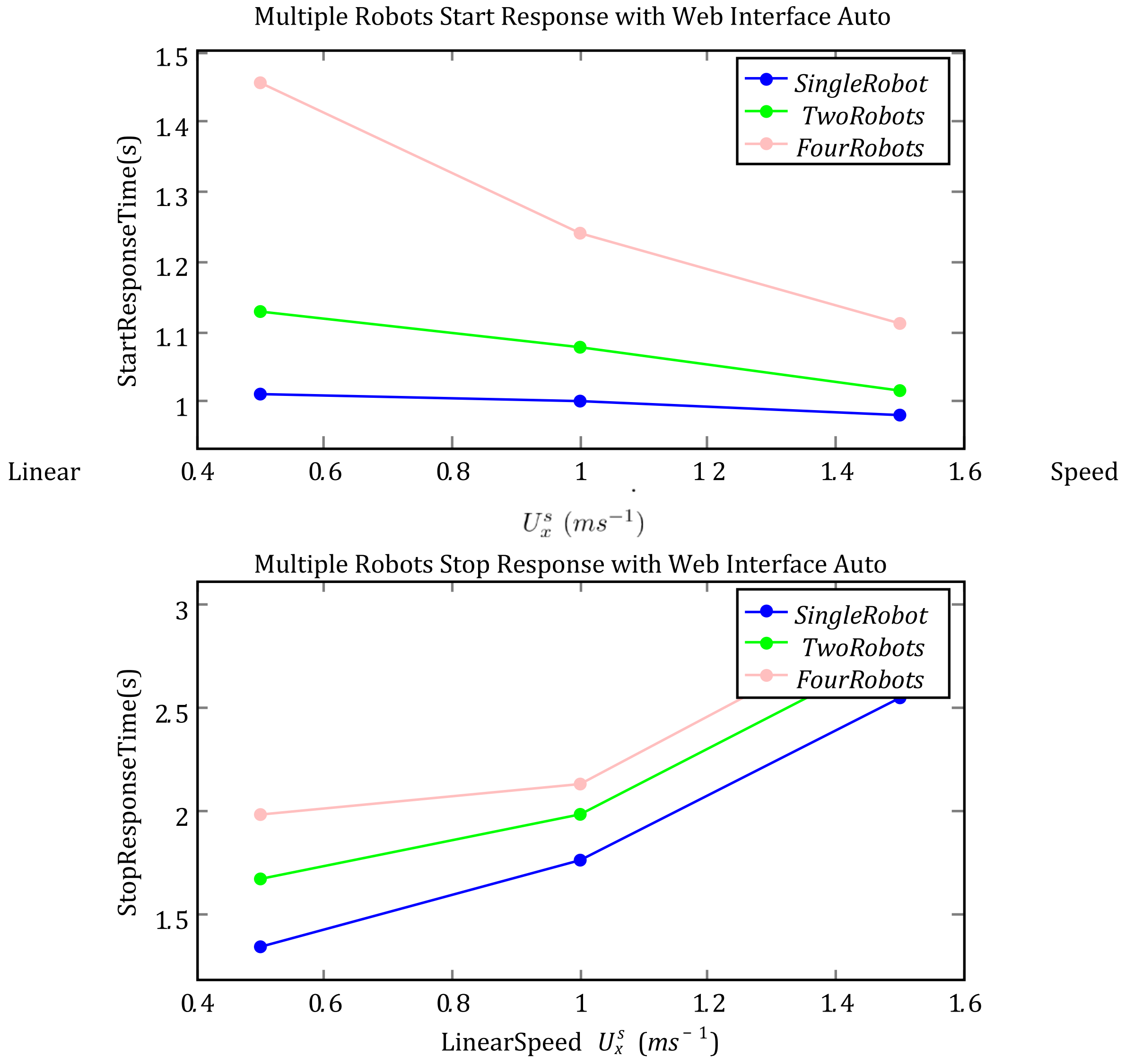


Fig. 4.13 Multi Robot Interaction with Web Interface

$c_7 = 0.042$. Three test results were calculated using the equation we have derived with the use of constant value c_7 we have calculated from the initial experimental results as shown in Table 4.19. We have used the person() value to find the correlation between the value generated from the equation and the experiment response value, as shown in Table 4.20. The pearson() value is higher for the value $c_7 = 0.022$. Therefore, we can select $c_7 = 0.022$ as the more accurate value for the equation 4.7.$\tau_{d,Web}$ represents the delay in communication through the Web interface, $\tau_{d,ROS}$ is used to represent the delay in communicating with ROS topics, $\tau_{d,RT}$ represents the delay in ROS topic identification and c_7, c_8 are constants. α and β represent the number of robots in the environment. For example, when the number of robots is two, then α = 2 and β = 2 and When the number of robots is four, then α = 4 and β = 4.

We have determined the constant c_8 using the experiment results assuming the valTable 4.19 Testing to Determine the Constant c_7 in Equations 4.7

$StartResponse(s)$	U_{xs} in ms^{-1}	ω_{zs} in ms^{-1}	$Test01$ (c_7 = 0.022)	$Test02$ (c_7 = 0.032)	$Test03$ (c_7 = 0.042)
1.011	0.5	0.0	1.011	1.031	1.051
1.001	1.0	0.0	0.969	0.979	0.989
0.981	1.5	0.0	0.946	0.953	0.960
1.129	0.5	0.0	1.011	1.031	0.989
1.078	1.0	0.0	0.969	0.979	0.960
1.016	1.5	0.0	0.946	0.953	1.051
1.456	0.5	0.0	1.011	0.953	0.989
1.242	1.0	0.0	0.969	1.031	0.960

Table 4.20 Pearson value (r) for each Testing

$TestName$	$PearsonValue(r)$
$Test01$ (c_7 = 0.022)	0.49650
$Test02$ (c_7 = 0.032)	0.49418
$Test03$ (c_7 = 0.042)	0.49246

Table 4.21 Testing to Determine the Constant c_8 in Equations 4.8

$StartResponse(s)$	U_{xs} in ms^{-1}	ω_{zs} in ms^{-1}	$Test01$ (c_8 = 1.20)	$Test02$ (c_8 = 1.20)	$Test03$ (c_9 = 1.20)
1.345	0.5	0.0	1.345	1.370	1.395
1.765	1.0	0.0	1.345	2.009	2.059
2.552	1.5	0.0	1.949	2.647	2.722
1.674	0.5	0.0	2.552	1.370	1.395
1.987	1.0	0.0	1.345	1.999	2.049
2.987	1.5	0.0	1.949	2.627	2.702
1.987	0.5	0.0	2.552	1.370	1.395
2.134	1.0	0.0	1.345	1.999	2.049

Table 4.22 Pearson value (r) for each Testing

$TestName$	$PearsonValue(r)$

$Test01$ ($c_8 = 1.20$)	0.858045
$Test02$ ($c_8 = 1.25$)	0.855090
$Test03$ ($c_8 = 1.30$)	0.855204

ues of $\tau_{d,RT}$, $\tau_{d,Web}$ and $\tau_{d,ROS}$ are constant for the given angular and linear speeds. The validation of the constant c_8 was completed with three Testing named Test 01, Test 02 and Test 03 with three different constant values for $c_8 = 1.20$, $c_8 = 1.25$ and $c_8 = 1.30$. Three test results were calculated using the equation we have derived with the use of constant value c_8 we have calculated from the initial experimental results as shown in Table 4.21. We have used the person() value to find the correlation between the value generated from the equation and the experiment response value, as shown in Table 4.22. The pearson() value is higher for the value $c_8 = 1.20$. Therefore, we can select $c_8 = 1.20$ as the more accurate value for the equation 4.8.$\tau_{d,Web}$ represents the delay in communication through the Web interface, $\tau_{d,ROS}$ is used to represent the delay in communicating with ROS topics, $\tau_{d,RT}$ represents the delay in ROS topic identification and c_7, c_8 are constants. α and β represent the number of robots in the environment. For example, when the number of robots is two, then $\alpha = 2$ and $\beta = 2$ and When the number of robots is four, then $\alpha = 4$ and $\beta = 4$.

4.5 Experiment 05: Move the robots to a specific location with a web interface with autonomous robot registration.

Authors have completed the experiment to move the robot(single robot, two robots, and four robots) to a given target location by an instruction using the Web interface. On average, the robots were placed at different positions to move the same distance. The following map represents the initial position and target locations of two and four robots as shown in Figure 4.14. The robots linear speed and angular speed are set as $U_x^s = 0.8\ ms^{-1}$ and$\omega_z^s = 0.2\ ms^{-1}$ respectively based on the defined condition to find the specific location.

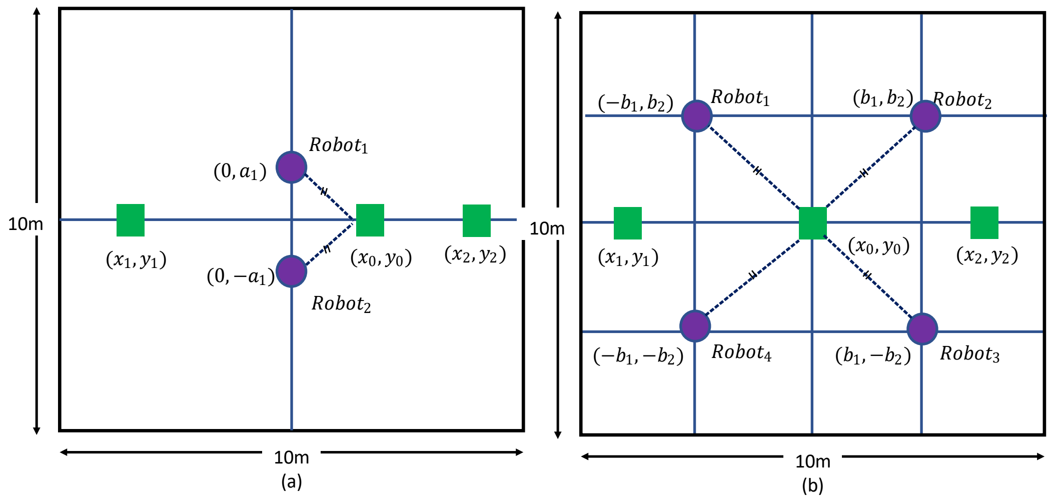


Fig. 4.14 Initial Position and Target Locations (a) Two Robots (b) Four Robots

Authors have conducted the experiments with a single robot, two robots, and four

Table 4.23 Average moving Time for Multiple Robots with Single Instruction

Average move Time (s)	Move to (x_0,y_0)	Move to (x_0,y_0) and (x_1,y_1)	Move to (x_0,y_0),(x_1,y_1) and (x_2,y_2)
$SingleRobot$	2.01	4.22	7.01
$TwoRobots$	2.24	5.01	7.34
$FourRobots$	3.05	6.21	8.01

Table 4.24 Testing to Determine the Constant c_9 in Equations 4.9

$Movetime(s)$	U_{xs} in ms^{-1}	ω_{zs} in ms^{-1}	$Test01$ ($c_9 = 1.1$)	$Test02$ ($c_9 = 1.2$)	$Test03$ ($c_9 = 1.3$)
2.01	0.8	0.2	3.01	3.11	3.21
4.22	0.8	0.2	5.23	5.51	5.87
7.01	0.8	0.2	8.54	9.11	9.71

robots with a single instruction to move the robot to a specific location given by (x,y) coordinates. The average Time taken by robots to a specific location was measured and presented in Table 4.23. The average move time increases with the number of robots and distance, as shown in Figure 4.15. The delay for moving a single robot and multiple robots is represented by the equation 4.9, and 4.10 where $R_{s,d}^{move}$ and $R_{m,d}^{move}$represent the single and multiple robots average moving Time to specific location respectively, $\tau_{d,Web}$ represents the delay in communication through the Web interface,$\tau_{dis,x,y}$ is used to represent the distance travelled by the robots, $\tau_{d,ROS}$ is used to represent the delay in communicating with ROS topics, $\tau_{d,RT}$ represents the delay in ROS topic identification, $\tau_{d,pos}$ is used to represent a delay in getting the current position and orientation of the robot, and c_9 and c_{10} are constants. α and β represent the number of robots in the environment. For example, when the number of robots is two, then $\beta = 2$ and When the number of robots is four, then $\beta = 4$.

$$R_{s,d}^{move} = \tau_{d,web} + \tau_{d,ROS} + \tau_{d,RT} + \tau_{d,pos} + \frac{c_9 \tau_{dis,x,y}}{\{U_x^s + \omega_z^s\}} \tag{4.9}$$

$$R_{m,d}^{move} = \beta\{\tau_{d,web} + \tau_{d,ROS} + \tau_{d,RT} + \tau_{d,pos}\} + \frac{c_{10} \tau_{dis,x,y}}{\{U_x^s + \omega_z^s\}} \tag{4.10}$$

We have determined the constant c_9 using the experiment results assuming the val-

Table 4.25 Pearson value (r) for each Testing

$TestName$	$PearsonV alue(r)$
$Test01$ ($c_9 = 1.1$)	0.998921

$Test02$ ($c_9 = 1.2$)	0.998843
$Test03$ ($c_9 = 1.3$)	0.999294

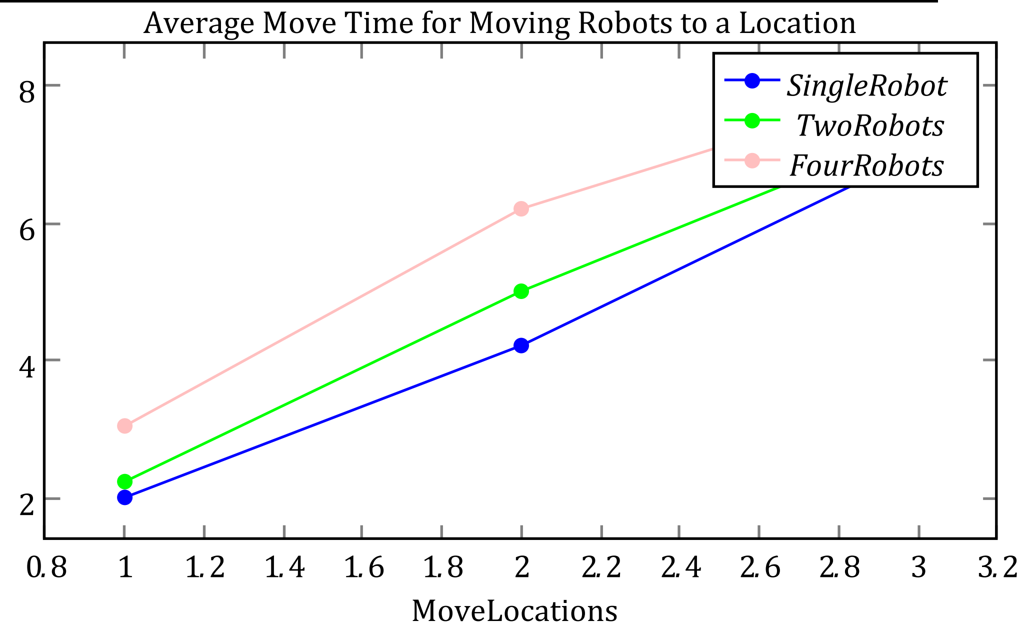


Fig. 4.15 Average Move Time for Moving a Robot to a Specific Location

ues of $\tau_{d,RT}$, $\tau_{d,Web}$ and $\tau_{d,ROS}$ are constant for the given angular and linear speeds. The validation of the constant c_9 was completed with three Testing named Test 01, Test 02 and Test 03 with three different constant values for $c_9 = 1.1$, $c_9 = 1.2$ and $c_9 = 1.3$. Three test results were calculated using the equation we have derived with the use of constant value c_9 we have calculated from the initial experimental results as shown in Table 4.24. We have used the person() value to find the correlation between the value generated from the equation and the experiment response value, as shown in Table 4.25. The pearson() value is higher for the value $c_9 = 1.3$. Therefore, we can select $c_9 = 1.3$ as the more accurate value for the equation 4.9. $\tau_{d,Web}$ represents the delay in communication through the Web interface,$\tau_{dis,x,y}$ is used to represent the distance travelled by the robots, $\tau_{d,ROS}$ is used to represent the delay in communicating with ROS topics, $\tau_{d,RT}$ represents the delay in ROS topic identification, $\tau_{d,pos}$ is used to represent a delay in getting the current position and orientation of the robot, and c_9 and c_{10} are constants. α and β represent the number of robots in the environment. For example, when the number of robots is two, then $\beta = 2$, and When the number of robots is four, then $\beta = 4$.

We have determined the constant c_{10} using the experiment results assuming the values of $\tau_{d,RT}$, $\tau_{d,Web}$ and $\tau_{d,ROS}$ are constant for the given angular and linear speeds. The validation of the constant c_{10} was completed with three Testing named Test 01, Test 02 and Test 03 with three different constant values for $c_{10} = 1.45$, $c_{10} = 1.55$ and $c_{10} = 1.65$. Three test results were calculated using the equation we have derived with

Table 4.26 Testing to Determine the Constant c_{10} in Equations 4.10

$MoveTime(s)$	U_{xs} in ms^{-1}	ω_{zs} in ms^{-1}	$Test01$ ($c_{10} = 1.45$)	$Test02$ ($c_{10} = 1.55$)	$Test03$ ($c_{10} = 1.65$)
2.24	0.8	0.2	2.90	3.00	3.10
5.01	0.8	0.2	4.35	4.65	4.95
7.34	0.8	0.2	7.25	7.85	8.05
3.05	0.8	0.2	2.90	3.00	3.10
6.21	0.8	0.2	5.80	6.20	6.40
8.01	0.8	0.2	11.6	12.5	13.00

Table 4.27 Pearson value (r) for each Testing

$TestName$	$PearsonValue(r)$
$Test01$ ($c_{10} = 1.45$)	0.900510
$Test02$ ($c_{10} = 1.55$)	0.904809
$Test03$ ($c_{10} = 1.65$)	0.903141

the use of constant value c_{10} we have calculated from the initial experimental results as shown in Table 4.26. We have used the person() value to find the correlation between the value generated from the equation and the experiment response value, as shown in Table 4.27. The pearson() value is higher for $c_{10} = 1.55$. Therefore, we can select $c_{10} = 1.55$ as the more accurate value for the equation 4.10. $\tau_{d,Web}$ represents the delay in communication through the Web interface, $\tau_{dis,x,y}$ is used to represent the distance travelled by the robots, $\tau_{d,ROS}$ is used to represent the delay in communicating with ROS topics, $\tau_{d,RT}$ represents the delay in ROS topic identification, $\tau_{d,pos}$ is used to represent a delay in getting the current position and orientation of the robot, and c_9 and c_{10} are constants. β represents the number of robots in the environment. For example, when the number of robots is two, then $\beta = 2$, and When the number of robots is four, then $\beta = 4$.

4.6 Experiment 06: Robot Interaction with multiple instructions with a web interface with autonomous robot registration.

Robots working in a real environment may get multiple instructions sequentially to complete multiple tasks. Our system must be able to handle the multiple instructions that are issued by users sequentially. A state transition machine is one of the most optimal solutions to handle this research problem Sen Gupta et al. (2002) Park et al. (2003) Bauer et al. (2018).

We have completed the experiment with the multiple instructions issued by the user sequentially with the state transition diagram. The sample interaction between the user instruction through the web interface and the robot is shown in Figure 4.16. This diagram represents only three user instructions that the user issues to control the robot. The experiment was conducted with three instructions to move the robot to three different locations. The target locations were represented as (x_0,y_0),(x_1,y_1) and (x_2,y_2). These ta get locations were selected to ensure all robots move an equal distance on average.

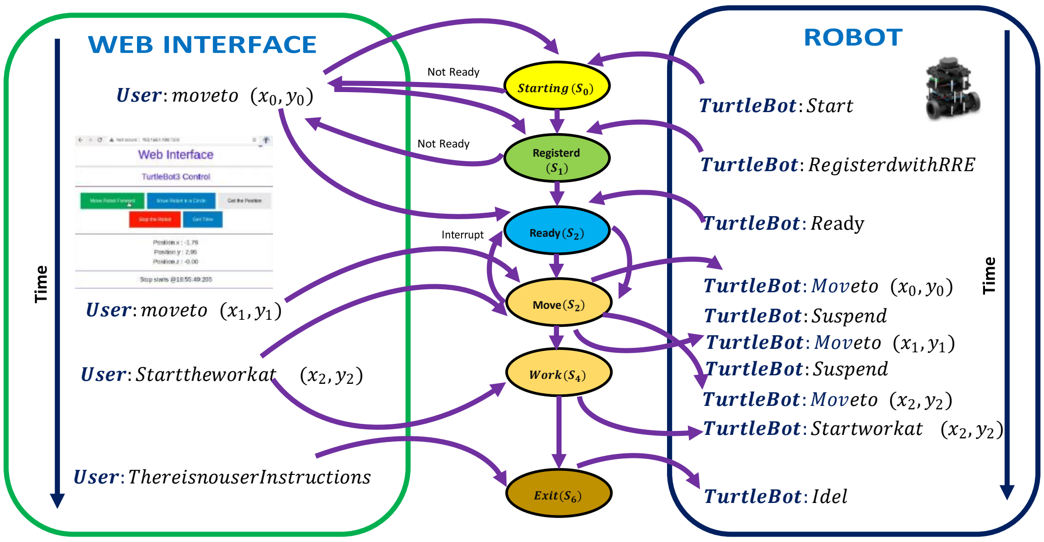
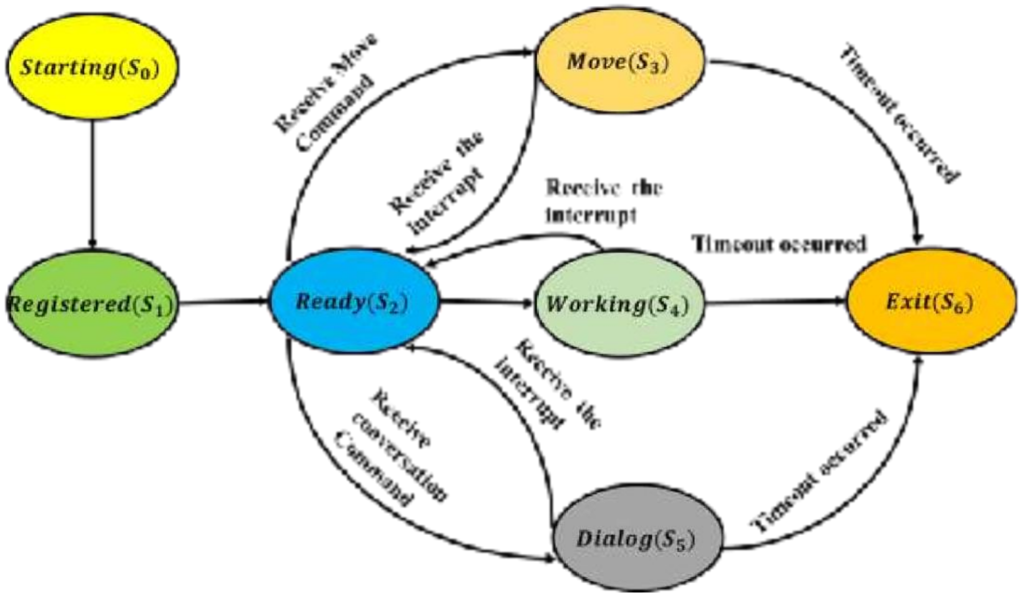

WEB INTERFACE
User: moveto (x_0, y_0)
Web Interface
TurtleBot3 Control
User: moveto (x_1, y_1)
User: Starttheworkat (x_2, y_2)
User: ThereisnouserInstructions
Time
Not Ready
Not Ready
Interrupt
Starting (S_0)
Registerd (S_1)
Ready (S_2)
Move (S_2)
Work (S_4)
Exit (S_6)
ROBOT
TurtleBot: Start
TurtleBot: RegisterdwithRRE
TurtleBot: Ready
TurtleBot: Moveto (x_0, y_0)
TurtleBot: Suspend
TurtleBot: Moveto (x_1, y_1)
TurtleBot: Suspend
TurtleBot: Moveto (x_2, y_2)
TurtleBot: Startworkat (x_2, y_2)
TurtleBot: Idel
Time

Fig. 4.16 Multiple Instructions and Robot Interaction

The initial robot positions for two and four robots are represented in the map given in Figure 4.18. The robots were initially placed concerning the target locations where each robot must move the same distance. The blue colour circle represents the initial robot position. The green colour square represents target locations given by user instructions. The target locations are identified to ensure all robots travel equal distances on average.

The system is implemented by handling multiple instructions one by one issued by the user using a state transition diagram with the states' description as shown in Figure 4.17. The robot state is saved in the ROS topic to retrieve the robot state from Time to Time. When the robot is ready, it will accept the user's instructions and complete the assigned work accordingly.

The equation representing the delay occurs because multiple instructions issued by the user were developed using the mathematical notation. We have used δ_{ij} as State transition time from i to j, $\forall(i,j) \in \{1,2,3,4,5,6\}$, S_δ as Time taken to save the state in ROS topic,R_{δ} as Time taken to retrieve the state from ROS topic,ϵ_n as Transition delay by n instructions where $n \in \{1,2,3,\ldots,l\}$. Total state transition delay time$\epsilon_n^s$for single instruction n = 1 is shown in the equation number 4.11. Total state transition delay time$\epsilon_n^m$for multiple instructions n = 1,2,3,..l is shown in the equation number 4.12. The delay for moving a single robot and multiple robots to a specific location with multiple instructions sequentially is represented by the equation 4.13 and 4.14 where $R_{s,d}{}^{mIns}$ and $R_{m,d}{}^{mIns}$represent the single and multiple robots delay in moving to specific



S_0:*Starting State*
S_1:*Registred State*
S_2:*Ready State*
S_3:*Move State*
S_4:*working State*
S_5:*Dialog State*
S_6:*Exit State*
$S_0 \rightarrow S_1$ *if Robot has registred.*
$S_1 \rightarrow S_2$ *if Robot is ready for inputs.*

$S_2 \rightarrow S_3$ *if Move command is received.*
$S_2 \rightarrow S_4$ *if Work command is received.*
$S_2 \rightarrow S_5$ *if dialog command is received.*
$\left.\begin{matrix} S_3 \\ S_4 \\ S_5 \end{matrix}\right\} \rightarrow S_2$ *if any interrupt command is received.*
$\left.\begin{matrix} S_3 \\ S_4 \\ S_5 \end{matrix}\right\} \rightarrow S_6$ *if any timeout has occured.*

Fig. 4.17 State Transition Diagram.

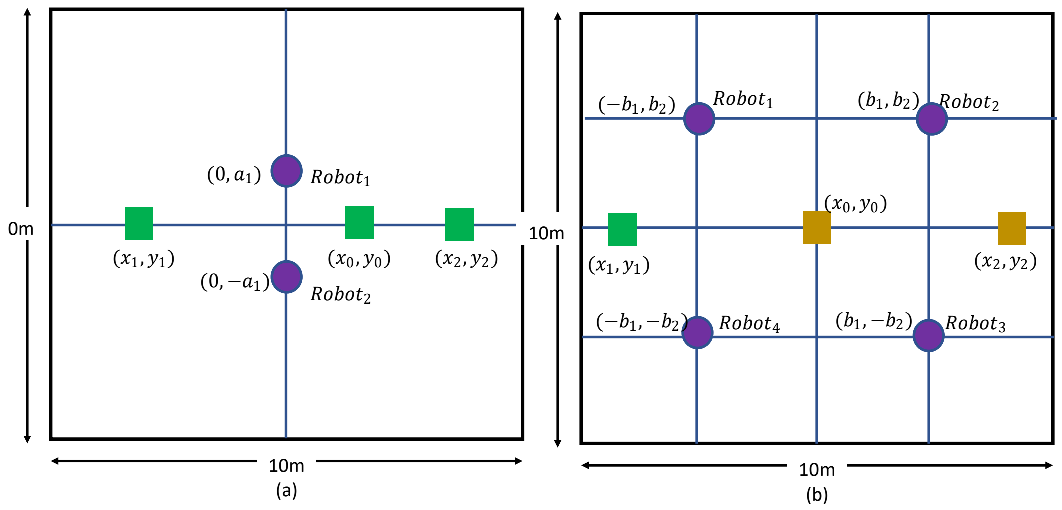


Fig. 4.18 (a) Initial positions of Two robots (b) Initial positions of Four robots location respectively.

$$\epsilon_n^s = \delta_{01} + \delta_{12} + \delta_{2j} + \delta_{j6} + 5S_\delta + 5R_\delta, \forall j \in \{3,4,5\}. \tag{4.11}$$

$$\epsilon_n^m = \delta_{01} + \delta_{12} + \sum_{n=1}^{n=l} \delta_{2j} + \delta_{j6} + (3+l)(S_\delta + R_\delta), if n = l, \forall j \in \{3,4,5\}. \tag{4.12}$$

$$R_{s,d}^{mIns} = \epsilon_n^s + \tau_{d,web} + \tau_{d,ROS} + \tau_{d,RT} + \tau_{d,pos} + \frac{c_1}{\{U_x^s + \omega_z^s\}} \tag{4.13}$$

$$R_{m,d}^{mIns} = \beta\{\epsilon_n^m + \tau_{d,web} + \tau_{d,ROS} + \tau_{d,RT} + \tau_{d,pos}\} + c_2\{U_x^s + \omega_z^s\} \tag{4.14}$$

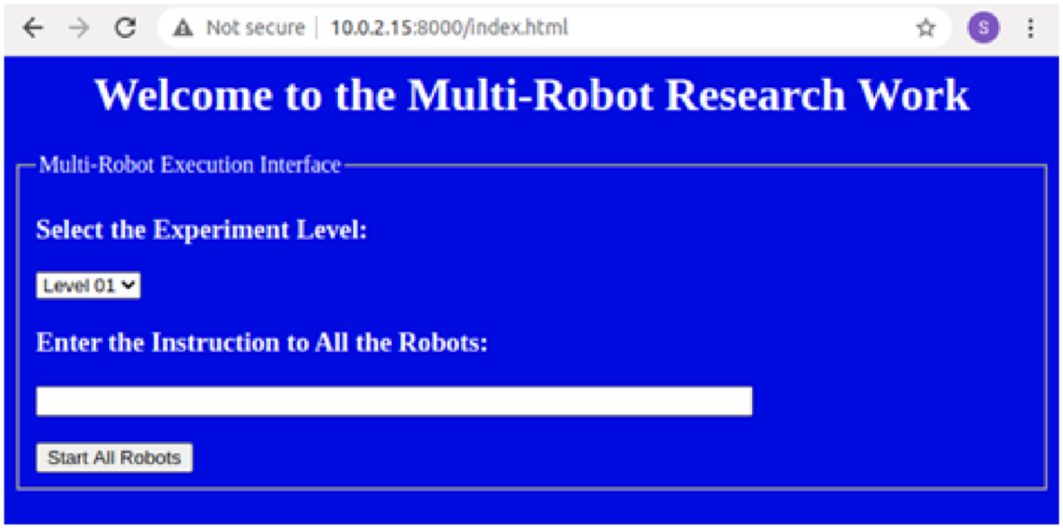
4.7 Experiment 07: Heterogeneous Multiple Robot Interaction with three levels of instruction.

Our system was evaluated using the two robots "*turtlebot*" and "*Tiago*" on the gazebo simulator. (*Python–mhttp.servercommand*) was implemented to publish the web pages for the web interface, which mainly works with java scripts. Finally, the rosbridge Server was implemented to provide the interface between ROS and non-ROS clients. In level 01 instruction, the robots were instructed to move 50m with the velocity of 1 ms^{-1}. Figure. 4.19 shows the web interface where users can select the experiment level for each input for all robots. In the level 02 instruction, the robots were given a specific location to reach from the current position. In the level 03 instruction, the robots had to use a map stored in the map server. Each robot was given a different goal to reach with obstacles in the environment.

We have conducted experiments with the system using the two robots with level 01, level 02, and level 03 instructions with twenty different instructions. The twenty instructions were developed using different synonyms for the given verb *move*. Unfortunately, some synonyms were not implemented in the ontology. Therefore, it has generated errors for synonyms not in the ontology.

1. Level 01: Move 50m with velocity $1ms^{-1}$.
2. Level 02: Move to the point (20, 20) with velocity $1ms^{-1}$.
3. Level 03: Move to the given goal in the map.

We have conducted the experiments with the instructions shown in Table 3.2 to test our system using a gazebo simulation environment with *turtlebot* and *Tiago* robots. Table 4.28 represents the results of the experiments. According to the experiment results, the error rate increases from level 1 to level 3 instructions. Levels 1 and 2 are



Welcome to the Multi-Robot Research Work

Multi-Robot Execution Interface

Select the Experiment Level:

Enter the Instruction to All the Robots:

Fig. 4.19 Web Interface Table
4.28 Experiment Results

Instruction Level	Language Translation Errors	Error Rate	Interpretation errors	Error Rate	Interpretation errors with navigation	Error Rate
Level 01	2	0.10	3	0.15	0	0.00
Level 02	3	0.15	4	0.20	0	0.00
Level 03	0	0.00	5	0.25	6	0.30

independent of navigation errors because they do not have any navigation but language translation and interpretation errors.

We have used the running time complexity analysis of the algorithm with the Big O notation, where we have assumed that the number of robots as i, the number of nodes as n, the number of topics as t, and the number of classes which are used to build the ontology as c.

Robot _Registration Algorithm()

This algorithm has four nested loops starting from the number of robots in the initial loop. Therefore, the Big O value of the Robot Registration Algorithm is O(i×n×t×c).

Get _Position and _Orientation Algorithm ()

This algorithm is used to get the current position and orientation of the robots; therefore, it does not depend on the number of robots. Therefore, the Big O value of the Get position and orientation Algorithm is O($n \times t \times c$).

Level 01 _Interpretation Algorithm ()

This algorithm needs to send the command to all robots using the loop and select all ROS topics. Therefore, the Big O value of the Level 01 Interpretation Algorithm () is O($i \times t \times c$).

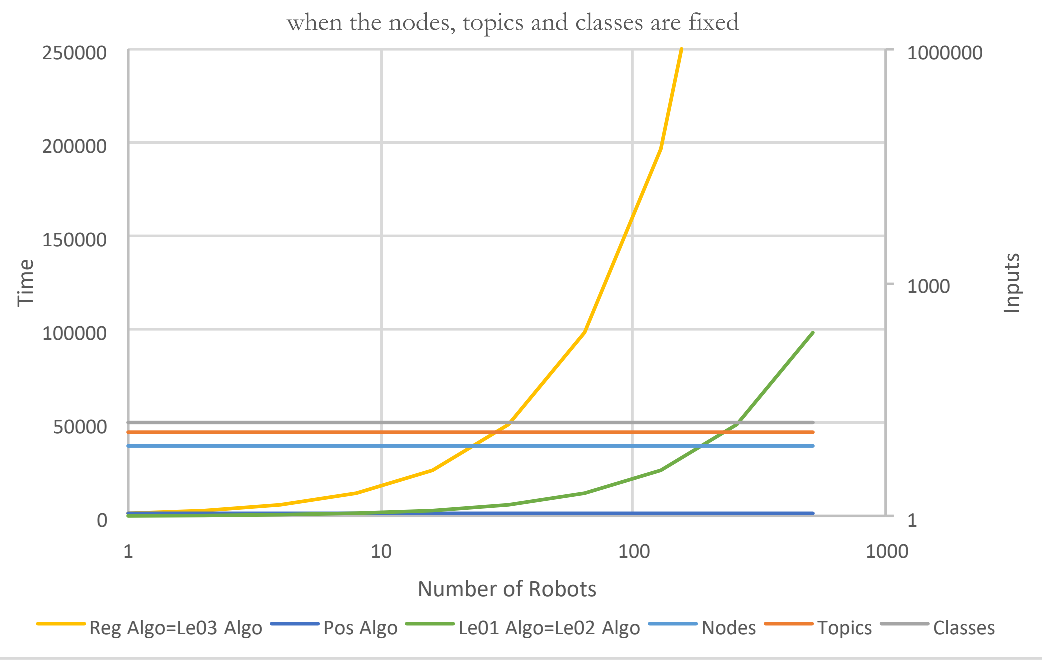


Fig. 4.20 Running Time Vs Static Inputs

Level 02 _Interpretation Algorithm ()

In this algorithm, we need to send the command to all robots using the loop. Therefore, the Big O value of the Level 02 Interpretation Algorithm () is O($i \times t \times c$).

Level 03 _Interpretation Algorithm ()

Here we need to apply the instructions for all robots using the loop. Each node (eg */scan* and *move _base*) is selected using the next level of the loop. We need to remap the topics in *amcl* and *move _base* launch files using the loop. Therefore, the Big O value of the Level 03 Interpretation Algorithm () is O($i \times n \times t \times c$). A summary of the Time Complexities is given in Table 4.29.

Therefore, the Big O value of the Registration Algorithm and Level03 Interpretation Algorithm has the same value as O($i \times n \times t \times c$). In addition, the Level 01 Interpretation Algorithm and Level 02 Interpretation Algorithm have the same value as O($i \times t \times c$).

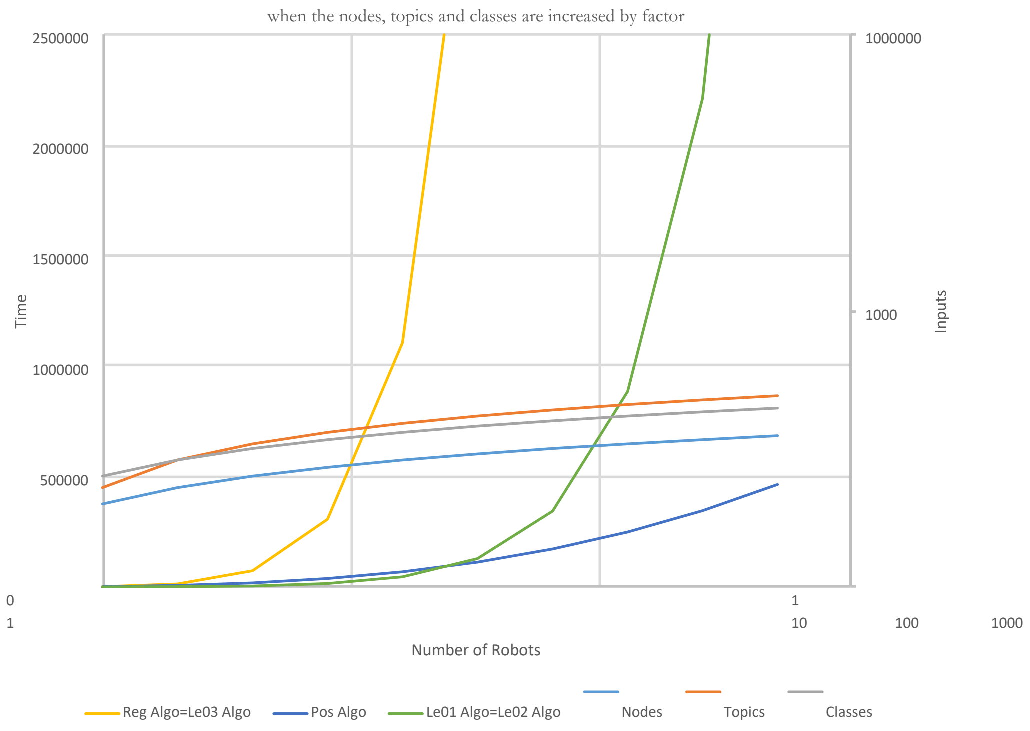


Fig. 4.21 Running Time Vs dynamic Inputs

Figure 4.20 represents the running time of the algorithm against the number of robots where we assume the number of nodes, the number of topics, and the number of classes of ontology as static. Then it indicates that the running Time is increasing slowly as a quadratic function like $y = x^2$.

Figure 4.21 represents the running time of the algorithm against the number of robots where we assume the number of nodes, the number of topics, and the number of classes in ontology as dynamic. Then it indicates that the running Time is increasing faster as cubic functions like $y = x^3$.

4.8 Experiment 08: Heterogeneous Multiple Robot Interaction with Semantic instruction with a web interface with autonomous robot registration.

Heterogeneous multiple robot control with very high-level instruction is one of the challenging issues in research groups in robotics. We have evaluated our system in the Gazebo environment using three robots *turtlebot*, *husky* and *TiaGo*. The virtual environment, available in Python httpserver (Python – mhttp), was executed to implement necessary web pages with java scripts for the web interface. We have used the rosbridge Server to work as an interface between ROS and non-ROS clients. The user has added the instruction on the web interface provided by the system to interact with the multiple robots. Table 4.31 shows the instruction types used to test our system. Type I was a Table 4.29 Running Time Analysis

Algorithms	NumberofRobots(i)	Numberofnodes(n)	Numberoftopics(t)	Numberofclasses(c)	TimeComplexity
Robot Registration Algorithm()	i=i	n=n	t=t	c=c	$O(i \times n \times t \times c)$
Get Position and Orientation Algorithm()	i=1	n=n	t=t	c=c	$O(n \times t \times c)$
Level 01 Interpretation Algorithm()	i=i	n=1	t=t	c=c	$O(i \times t \times c)$
Level 02 Interpretation Algorithm()	i=i	n=1	t=t	c=c	$O(i \times t \times c)$
Level 03 Interpretation Algorithm()	i=i	n=n	t=t	c=c	$O(i \times n \times t \times c)$

general instruction with no synonym or semantic issue. The synonym was added to instruction Type II, where a synonym analysis algorithm processed it. The semantics of the instruction are not clear in instruction Type III. Instruction type IV has both synonym and semantic issues. The synonym and semantics were not programmed for the instruction Type V where the user has to handle the synonym and semantic issues. The system was tested with many instructions, the Type I to Type V.

Table 4.30 Goal and Task Scheduling Table

	Time slot 1	Time slot 2	Time slot 3	Time slot 4
Robot Name	$t_0 - t_1$	$t_1 - t_2$	$t_2 - t_3$	$t_3 - t_4$
Turtlebot	$A(2,2) + rotate(5)$	*FreeTime*	$B(-2,2) + rotate(5)$	$C(2,-2) + + rotate(5)$
Husky	$D(5,5) + rotate(10)$	$E(5,-5) + rotate(10)$	*FreeTime*	$F(0,5) + rotate(10)$
TiaGo	*FreeTime*	$G(1,-1) + rotate(15)$	$H(0,1) + rotate(15)$	$I(-1,1) + rotate(15)$

The identification of the synonym and the semantic issues were performed by our algorithms accurately. Furthermore, we have completed the time complexity analysis of our algorithm to measure the system’s performance using the Big O notation. The time

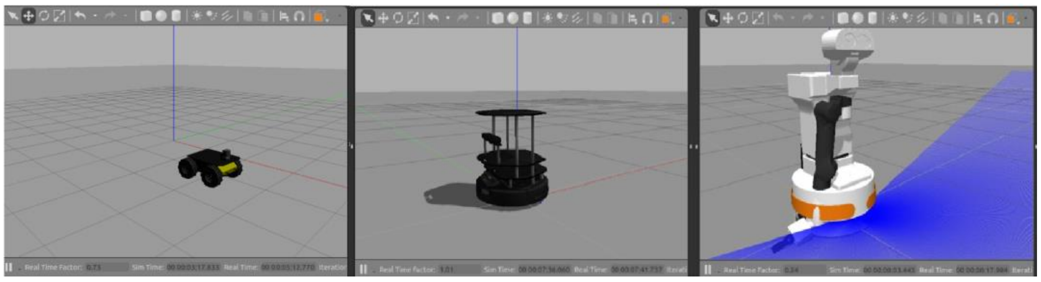


Fig. 4.22 Husky, Turtlebot and TiaGo Robots in Empty World

complexities of all algorithms are shown in Table 4.32. Time complexity is calculated using the number of loops used by each algorithm where *n* is the input size. The graph of the time complexity for all algorithms is shown in Figure 4.23. According to the time complexity analysis, we can identify that the Robot Registration Algorithm and ROS Topic Identification Algorithm have poor performance because time complexity is $O(n^4)$.

Table 4.31 Instruction Types used for Testing

Instruction Type	Description	Example
Type I	Instruction without synonym or semantic issue	Move to A and clean
Type II	Instruction with synonym	shift to B and clean
Type III	Instruction with semantic issue	Move to roof and clean
Type IV	Instruction with synonym and semantic issue	Shift to sky and clean

Type V	Instruction with synonym and semantic issue (Not programmed where user involvement is needed)	Proceed to sea and clean

Table 4.32 Time Complexity of Algorithms

Algorithm Name	Time Complexity in Big O notation
Robot Registration Algorithm()	$O(n^4)$
Synonym Analysis Algorithm()	$O(n^2)$
Semantic Analysis Algorithm()	$O(n^3)$
Get Position and Orientation Algorithm()	$O(n)$
ROS Topic Identification Algorithm()	$O(n^4)$

Time complexity analysis with Big O notation for each type of instruction is shown in Table 4.33. Command Interpreter have used the Synonym Analysis Algorithm(), and

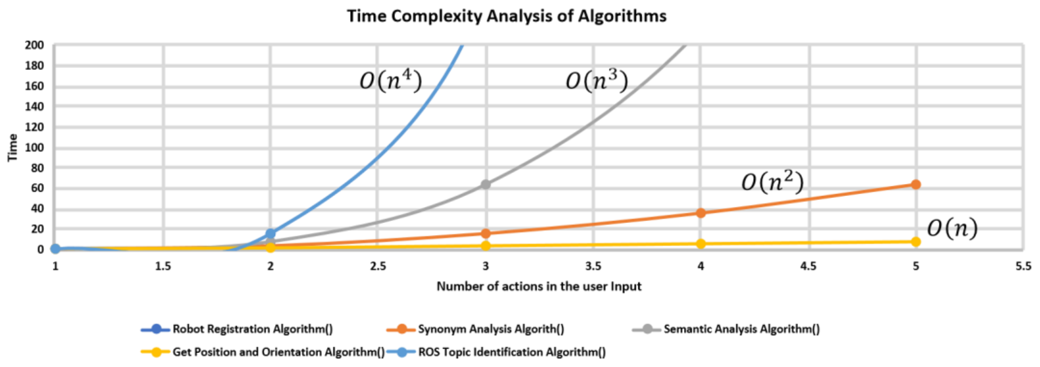


Fig. 4.23 The Graph of the Time Complexity of all algorithms

Semantic Analysis algorithm() where Synonym Analysis Algorithm() has taken $O(n^2)$, and Semantic Analysis algorithm() has taken $O(n^3)$ running time-based on the asymptotic notation in algorithm analysis. Therefore instruction type II is poor compared to instruction type III. Instruction type V is worse because user interaction is needed to solve the synonym and semantic issue in the instruction since synonyms and semantics are not programmed.

Table 4.33 Instruction Types with Time Complexity

Types	Algorithms used in Command Interpreter	Time Complexity	Algorithms used in Robot Registration and Command publishing Engine	Time Complexity
Type I	Analysis algorithm is not needed	$O(1)$	RR Algorithm()+ ROS TI Algorithm()	$O(n^4)$

Type II	Synonym Analysis Algorithm()	$O(n^2)$	RR Algorithm()+ ROS TI Algorithm()	$O(n^4)$
Type III	Semantic Analysis Algorithm()	$O(n^3)$	RR Algorithm()+ ROS TI Algorithm()	$O(n^4)$
Type IV	Synonym Analysis Algorithm()+ Semantic Analysis Algorithm()	$O(n^3)$	RR Algorithm()+ ROS TI Algorithm()	$O(n^4)$
Type V	Synonym Analysis Algorithm() + Synonym Analysis Algorithm()+Human Intervention is needed.	$O(n^3)$	RR Algorithm()+ ROS TI Algorithm()	$O(n^4)$

In addition to the above-discussed time complexity analysis for instruction type I to V, we have conducted two types of experiments with the Gazebo environment with *Turtlebot*, *Husky* and *TiaGo* robots. In the first experiment type, we have moved all heterogeneous robots to a given goal in the open world in the Gazebo. The second type Table 4.34 Experiment Results for without Navigation

Robot	Goal without Navigation			
Experiment	Goal 01 Success rate 08.0010.00	Goal 02 Success rate 10.0012.00	Goal 03 Success rate 12.0002.00	Goal 04 Success rate 02.0004.00
Turtlebot	0.65	0.85	0.90	0.95
Husky	0.50	0.65	0.70	0.80
TiaGo	0.45	0.55	0.65	0.85

Table 4.35 Experiment Results for with Navigation

Robot	Goal with Navigation			
Experiment	Goal 01 Success rate 08.0010.00	Goal 02 Success rate 10.0012.00	Goal 03 Success rate 12.0002.00	Goal 04 Success rate 02.0004.00
Turtlebot	0.40	0.55	0.75	0.80
Husky	0.35	0.40	0.55	0.70
TiaGo	0.30	0.45	0.60	0.75

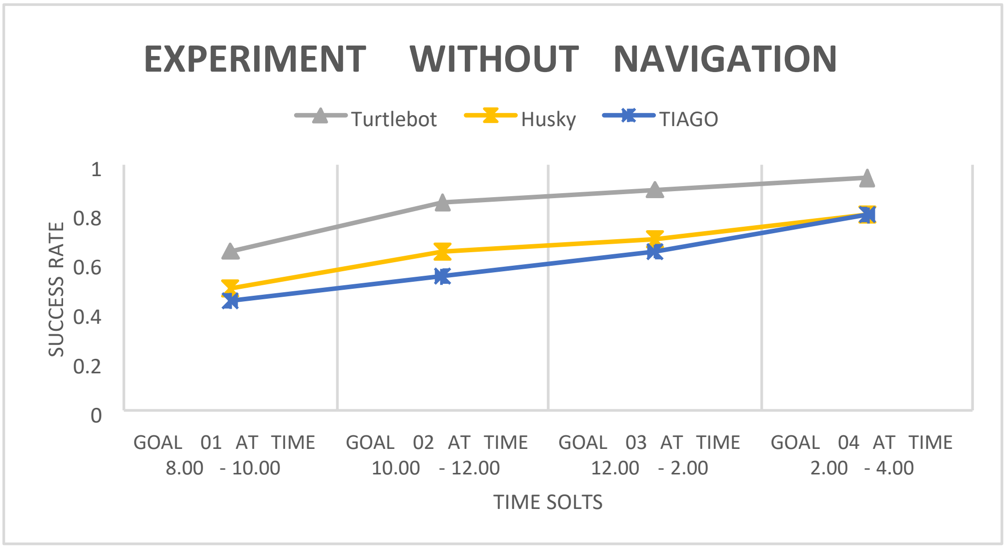


Fig. 4.24 Experiment without Navigation Success Rate

of experiment is to navigate all heterogeneous robots to a given goal with obstacles in the Gazebo. All three robots (turtle bot, husky, and Tiago) in an open world in the Gazebo are shown in Figure 4.22. Experiments were conducted using the system above multiple robots with movement and navigation using 20 type IV instructions. Users can update the goal and task assigned for each robot for the different schedules in Table
4.30. We have added the self-rotation for each robot to simulate the task completed

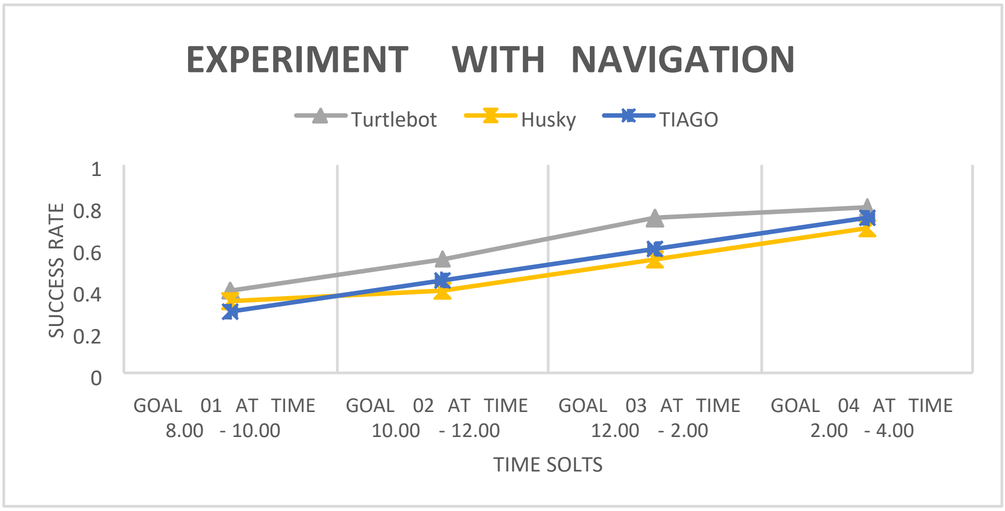


Fig. 4.25 Experiment with Navigation Success Rate

by robots based on the scheduled task. We found some errors in Robot Registration Algorithm and ROS Topic Identification Algorithm() for movements and navigation.

There were more ROS topic settings than the robot's movement in an open world in navigation.

The results of the experiment are represented in the Table for three robots *Turtlebot*, *Husky* and *TiaGo* where we have tested 20 times for each goal at four different time slots as *(8.00 -10.00 am), (10.00 -12.00 noon), (12.00 -2.00 pm), (2.00-4.00 pm)*. We received different ontology searching errors, robot registration errors, ROS topic identification, and command publishing errors in each time slot. Therefore, we gradually minimized the error with the experienced we had in each experiment with the timing. The success rate is measured with 20 tests. It defines the number of successful tests without errors out of 20 tests for each robot in each type of experiment.

The results of experiment type 01 (without navigation) are shown in Table 4.34. According to the analysis, we have identified that the *turtlebot* has a higher success rate compared to other robots, as shown in Figure 4.24.

The results of experiment type 02 (with navigation) are shown in the table 4.35. The success rate is also increasing, similar to experiment 01, as shown in Figure 4.25.

The running time of the Robot Registration Algorithm and ROS Topic Identification Algorithm is $O(n^4)$ where *n* is the number of actions defined in the user instruction. These two algorithms had the highest time complexity compared to other algorithms we have developed in our system.

In general, the delay in response time for the start decreases when the linear and angular speed increases. However, the delay in response time for the stop increases when the linear and angular speed increases. In addition, a delay occurs when the robot is controlled without the Web interface because of the delay with system call execution through the operating system and communication with ROS functions. When a robot is controlled through the Web without auto registration, a delay occurs in communication through the Web and communication with ROS through the ROS Bridge server. When the auto registration was added to the system, we needed to add the delay taken by the algorithm for the ROS topic identification. The delay time increases with the number of robots increased. When the robot is sent to a specific location, we need to add Time taken to get the current position and orientation for the delay time. When multiple instructions control a robot, we use a state transition system. Therefore, we need to add the Time taken by the state transition system for saving and retrieving the state to the delay time to get more accurate results. According to the analysis, the authors have identified that web communication is slightly faster than communication through the terminal

CHAPTER 5

Discussion

5.1 Problem and Solution

Here we discuss the main problem in the research and the solution achieved. One of the research problems we have solved with our solution is providing an environment that hides all difficulties and complexities of programming a robot or multiple robots concurrently with a simple interface. Another research problem we have solved is learning ROS from scratch and developing a comprehensive robotic application that can provide a simple interface for controlling robots through the Internet. We have developed the interfaces to manage and control multiple robots through a web interface. The other main research problem is to develop an algorithm that can be used to register all robots by getting all software-related specifications and hardware specifications. We successfully developed the algorithm to register all reboots concurrently through the web interface. The other research problem is identifying the relevant ROS topics and nodes for the given user instruction. The ROS topic identification algorithm was completed successfully with the use of ontology.

Moving all robots to the given location is another research problem we selected to solve. Therefore, the algorithm was developed to move all robots to a specific location in the environment of the gazebo simulator. Finally, another research problem is analysing the algorithms' time complexity using Big O notation. We have completed the complexity analysis of the algorithm to compare its performance of the algorithm.

5.2 Research Findings

One of the research findings was identifying the limitations and issues with the current and previous research works by thoroughly studying all existing research papers. One of the other research findings was designing and developing the control and managing algorithms for all robots through the Web interface with user instructions without considering all software and hardware differences of all robots. According to the experiments we have conducted with a web interface with multiple robots, the research finding was that communication and control of multiple robots could be achieved with speed as close to the speed without a web interface. One of the other research findings was deriving the mathematical equation for the delay in each experiment and the algorithm’s time complexity to decide each algorithm’s performance.

CHAPTER 6

Conclusion

This research study has developed a system to issue instruction through the web interface and controls multiple robots simultaneously. Previous research work for control and communication with multiple robots through the internet had several limitations and issues. Initially, all multiple robots need to register with Robot Registration Engine. The autonomous robot registration and autonomous ROS topic identification algorithms were implemented successfully. One of the research components that we completed was the development of the Robot Registration Engine to collect each robot's details in our system through the web interface. This was completed very successfully with our algorithms. When a user has given instructions, the system must identify the corresponding ROS topic with our Ros topic identification algorithm. The ROS topic identification algorithm was implemented very successfully.We have analyzed the delay time in response to all experiments. The delay time is increased with the introduction of these algorithms. We have derived the mathematical equations for each delay time which varies based on the inputs and system characteristics. The experiment result indicated that the autonomous robot registration was successful, and the communication performance through the Web decreased gradually with the number of robots registered. The running time of the Robot Registration Algorithm and ROS Topic Identification Algorithm is $O(n^4)$.

The main objective is to develop an algorithm to interact and control multiple robots through the web interface with autonomous robot registration and autonomous ROS topic identification. We found that the number of resources we could use for the ROS was minimal. Therefore, one of the leading research project objectives is to learn the ROS from scratch and develop control applications with different robots with some automation. We have successfully achieved this objective by studying the available resources. We have studied ROS from the beginning and developed a good application with multiple robots through a web interface.

Many research groups completed robot control and communication with ROS, but according to our research studies, we did not find any research for autonomous robot registration using any algorithms. Therefore, one of the leading research project objectives is to develop an algorithm that can register all robots concurrently through the Web interface. We successfully developed Robot Registration Engine with our Robot Registration algorithm through the web interface.

According to the previous studies, we did not find any algorithm developed to identify the relevant ROS topics and nodes for subscription and publication. Therefore, one of the leading research project objectives is to develop an algorithm that can identify the relevant ROS topics and nodes to complete the issued task by the user. As a result, we successfully implemented the algorithm to identify the relevant ROS topics and nodes for subscription and publication.

Managing multiple robots with a Gazebo environment through a web interface is complex. Most of the time, managing the errors with ROS and Gazebo is very tedious and time-consuming because fewer resources are available online for ROS and Gazebo. Therefore, one of the leading research project objectives is to learn the ROS and gazebo environment from scratch and simulate the environment for the experiments. As a result, we simulated multiple robots in Gazebo to successfully make the experiments even though we got more issues at the beginning of the simulation process.

Analyzing the algorithm is very important to find the performance of each algorithm. There are several algorithm analysis techniques. Big O notation is the optimal way to represent the algorithm's complexity. One of the leading research project objectives is to develop an optimal algorithm to get the correct output and analyze the performance of the algorithms. We successfully developed all algorithms and evaluated the performance using the best time complexity analysis technique.

Performance evaluation with response time is another research problem that we want to solve. Several experiments must be completed with different web interfaces with different amounts of robots for different scenarios. Again, we need to derive the mathematical equations from representing each scenario's delay in response time. Therefore, another main objective is to perform analysis with derived mathematical equations for each scenario with different web interfaces. We have successfully evaluated the performance in terms of delay in response time and derived the mathematical equation for each scenario.

According to the experiments we have conducted with a web interface with multiple robots, the research finding was that communication and control of multiple robots could be achieved with speed as close to the speed without a web interface.

6.1 Contribution

The main contribution to my research work are summarized below:

First, I discovered and proposed a new algorithm for autonomous, multiple robot registration in a simulated Gazebo environment. According to the previous research studies described in chapter 02, I did not find any research on autonomous robot registration. Once a robot is connected through the Web interface, all ROS topics and nodes related to each robot are collected and stored to use later. One of the main limitations of the ROS is that robotic programming very difficult and but autonomous registration solve this issue since it collects all ROS topics and nodes necessary to publish and subscribe.

I discovered and proposed a new algorithm for ROS topic identification when a user issued a command to control robots through the web interface. However, according to the previous research studies described in chapter 02, I did not find any research on ROS topic identification algorithms. This task minimizes the robot's programming difficulties with ROS since our algorithm can find relevant ROS topics for each robot to publish and subscribe to control.

I discovered and developed the web interface to control multiple robots with simple commands to move robots forward and circle simultaneously using the threads in python language. However, according to the previous research studies described in chapter 02, I did not find any research on multiple robot controls through the web interface. Some research was done to control single robots through web interfaces without autonomous registration.

I discovered the worst-case complexity of the autonomous robot registration algorithm and ROS topic identification algorithm. This analysis results can be used by other researchers when they want to get an idea of the algorithm's performance.

I discovered and derived mathematical equations to represent the delay in response time for the different scenarios with all experiments with other characteristics. Furthermore, I found and validated the values for all constants in each mathematical equation.

6.2 Limitation

When we consider real robots, different mechanical components are necessary to complete a task. Therefore, the robot's response time depends on the mechanical components' delay. Therefore, the response delay time we have calculated will be significant and can be changed when implemented with real robots. Furthermore, these response times can be changed depending on the robot type. The kinematics and dynamics of the robot were not considered because all experiments were completed in a simulation environment.

All simulations were conducted only in an empty world in the Gazebo environment. When the navigation algorithms are implemented with the Robot Registration algorithm, we can implement robot control in the environment with obstacles on the given path. This is another limitation of our solution.

The other limitations of our project is a simulation of the system with the Gazebo simulator. I have simulated multiple robots in the Gazebo environment and used the namespace to identify each robot uniquely. The launch files were developed to spawn all multiple robots successfully at different places. All experiments were completed in the simulated environment and did not use real-time robots for testing. Of course, we can enhance all results by implementing all experiments in a real environment with robots.

If the robots are not implemented with the ROS, then our solution will not be able to use because we have developed an algorithm to work with only a ROS-based system. However, we have not implemented any algorithm to control the robots with other middleware.

Our experiments were conducted with user commands to move all robots forward with different linear speeds, move all robots to a circle with different angular speeds from different positions and move all robots to specific locations. We have not considered the synchronization of all robots, access control of all robots, collision avoidance of all robots and relative motion of all robots. These limitations were not considered in all experiments we have implemented to get the results. Navigation algorithms were not implemented through the web interface when we controlled multiple robots.

We have designed a state transition engine that can handle the multiple instructions issued by users sequentially. A state transition machine is one of the most optimal solutions to handle multiple instructions sequentially.We have not implemented the state transition engine to work with instructions in a simulated environment.

Performance evaluation with response time is another research problem that we want to solve. Several experiments must be completed with different web interfaces with different amounts of robots for different scenarios. Again, we need to derive the mathematical equations from representing each scenario's delay in response time. The evaluation results can be slightly different from real-world implementation results because we have derived all mathematical equations with some assumptions. This can be another limitation of our evaluation.

There can be some robots that are not using the ROS but may be using their middleware then. Our system may need to be fixed with this kind of robot. Then we need to have a mapping with ROS and other middleware to work with our system. This will be another limitation of our implementation.

CHAPTER 7

Future Works

7.1 Future Works

Working with multiple instructions sequentially is very important when considering real robots. However, we have only designed the state transition diagram to work with multiple robots, which can be implemented in future works. The state transition diagram we have developed can be converted into program code and implemented in real robots to work with multiple instructions.

The semantics and synonyms of the user instructions can be implemented with the ontology. That may provide the very intelligent aspect of the robot. However, we have not implemented ontology directly in our solution. The semantics and synonym of the instructions can be easily implemented with ontology in future work.

Our solution conducted all simulations in an empty world in the Gazebo environment. As a future work, the navigation algorithms can be implemented with the robot registration algorithm. Then the robot can move even with obstacles on the given path.

Many techniques can be used to improve the performance of the algorithms we have developed. Dynamic programming, greedy method, parallel algorithms, distributed algorithms, and Artificial intelligence-based algorithms. These algorithm optimization techniques can be used to improve the performance algorithm.

All experiments were completed in the simulated environment and did not use actual real-time robots for testing. Therefore, we can enhance all results by implementing all experiments in a real environment with robots. However, most existing research studies indicated that the simulated experiment results are closely related to the real environment. Therefore, the actual implementation of the system in real robots with a real environment can be completed as future work.

We have not used ontology directly in our application. However, we have simulated the ontology model in our system since working with ontology is not our primary objective of the project in identifying the synonym for user commands. However, we have used the algorithm to simulate some properties of the ontology to get the synonym for user commands. Because synonym identification is additional work in our project. In future work, we can create a complete ontology and manage the system with a high-processing CPU to get more accurate results.

We have not considered the synchronization of all robots, access control of all robots, collision avoidance of all robots and relative motion of all robots. These limitations were not considered in all experiments we have implemented to get the results. Navigation algorithms were not implemented through the web interface when we controlled multiple robots. We can develop synchronization algorithms with multiple robots for navigation and avoiding collisions with each other. The access controlling issue can be easily tested with actual real-time robots. Some algorithms were already developed for group formation for multiple robots in some research groups.

Many systems are developed with a state transition engine to complete the complicated task. We have designed a state transition engine that can handle the multiple instructions issued by users sequentially. Therefore, one of the other main research objectives is to design a theoretical state transition machine to work with multiple instructions that are issued by users sequentially. We have successfully designed the state transition engine theoretically and evaluated its performance. However, we have not implemented the state transition engine to work with instructions in a simulated environment. This engine can be developed and integrated into the system as future work to complete the multiple instructions a user gives.

Performance evaluation with response time is another research problem that we want to solve. Several experiments must be completed with different web interfaces with different amounts of robots for different scenarios. Again, we need to derive the mathematical equations from representing each scenario's delay in response time. Then, the performance evaluation can be tested with real robots to get more accurate results for future work.

There can be some robots that are not using the ROS but may be using their middleware then. Our system may not be working with this kind of robot. Then we need to have a mapping with ROS and other middleware to work with our system. This will be another limitation of our implementation. We can develop another registration engine to get the services of other middleware as web services and control the robots being developed middleware other than ROS.

7.2 Funding Details

This work was supported by the Sri Lanka Institute of Information Technology under Grant number FGSR/RG/FC/2021/05.

7.3 Acknowledgement

We thank SLIIT(Sri Lanka Institute of Information Technology) for the support given towards this Research Project.

REFERENCES

Alberri, M., Hegazy, S., Badra, M., Nasr, M., Shehata, O. M. and Morgan, E. I. (2018), Generic ros-based architecture for heterogeneous multi-autonomous systems development, *in* '2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES)', pp. 1–6.

Ali, A. A., Rashid, A. T., Frasca, M. and Fortuna, L. (2016), 'An algorithm for multirobot collision-free navigation based on shortest distance', *Robotics and Autonomous Systems* 75, 119–128.

Anggraeni, P., Rokhim, I. and Salam, R. M. (2020), Design and development of multiple mobile manipulator robots using gazebo-ros, *in* '2020 International Conference on Applied Science and Technology (iCAST)', pp. 672–676.

Bauer, A. S., Schmaus, P., Albu-Schaffer, A. and Leidner, D. (2018), Inferring seman-¨ tic state transitions during telerobotic manipulation, *in* '2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', pp. 1–9.

Bucolo, M., Buscarino, A., Famoso, C., Fortuna, L. and Frasca, M. (2019), 'Control of imperfect dynamical systems', *Nonlinear Dynamics* 98(4), 2989–2999.

Buscarino, A., Fortuna, L., Frasca, M. and Rizzo, A. (2006), 'Dynamical network interactions in distributed control of robots', *Chaos: An Interdisciplinary Journal of Nonlinear Science* 16(1), 015116.

Codd-Downey, R. and Jenkin, M. (2015), Rcon: Dynamic mobile interfaces for command and control of ros-enabled robots, *in* '2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)', Vol. 02, pp. 66–73.

Costa, L. F. and Gonc¸alves, L. M. G. (2016), Roboserv: A ros based approach towards providing heterogeneous robots as a service, *in* '2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR)', pp. 169–174.

Crick, C., Jay, G., Osentoski, S. and Jenkins, O. C. (2012), Ros and rosbridge: Roboticists out of the loop, *in* '2012 7th ACM/IEEE International Conference on HumanRobot Interaction (HRI)', pp. 493–494.

Datta, C., Jayawardena, C., Kuo, I. H. and MacDonald, B. A. (2012), Robostudio: A visual programming environment for rapid authoring and customization of complex services on a personal service robot, *in* ‘2012 IEEE/RSJ International Conference on Intelligent Robots and Systems’, pp. 2352–2357.

Datta, C., MacDonald, B., Jayawardena, C. and Kuo, I. (2012/10/29), Programming behaviour of a personal service robot with application to healthcare, *in* ‘International Conference on Social Robotics, Springer, Berlin, Heidelberg’, pp. 228–237.

Han, R., Chen, S. and Hao, Q. (2020), Cooperative multi-robot navigation in dynamic environment with deep reinforcement learning, *in* ‘2020 IEEE International Conference on Robotics and Automation (ICRA)’, pp. 448–454.

Hendler, J. (2001), ‘Agents and the semantic web’, *IEEE Intelligent Systems* 16(2), 30–37.

Hu, C., Hu, C., He, D. and Gu, Q. (2015), A new ros-based hybrid architecture for heterogeneous multi-robot systems, *in* ‘The 27th Chinese Control and Decision Conference (2015 CCDC)’, pp. 4721–4726.

Ivanov, A., Zakiev, A., Tsoy, T. and Hsia, K.-H. (2021), Online monitoring and visualization with ros and reactjs, *in* ‘2021 International Siberian Conference on Control and Communications (SIBCON)’, pp. 1–4.

Jayawardena, C., Kuo, I., Broadbent, E. and MacDonald, B. A. (2016), ‘Socially assistive robot healthbot: Design, implementation, and field trials’, *IEEE Systems Journal* 10(3), 1056–1067.

Jeon, S., Jang, M., Lee, D., Chang-Eun Lee and Cho, Y. (2012), Control architecture for heterogeneous multiple robots with human-in-the-loop, *in* ‘2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)’, pp. 274–278.

Kato, T., Watanabe, K. and Maeyama, S. (2010), A formation method for heterogeneous multiple robots by specifying the relative position of each robot, *in* ‘Proceedings of SICE Annual Conference 2010’, pp. 3274–3277.

Kim, J., Jang, M. and Sohn, J. (2006), An ontological approach for natural language command interpretation and its application in robotics, *in* ‘2006 SICE-ICASE International Joint Conference’, pp. 3874–3878.

Koenig, N. and Howard, A. (2004), Design and use paradigms for gazebo, an opensource multi-robot simulator, *in* ‘2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)’, Vol. 3, pp. 2149–2154 vol.3.

Lashkari, N., Biglarbegian, M. and Yang, S. X. (2020), ‘Development of a novel robust control method for formation of heterogeneous multiple mobile robots with autonomous docking capability’, *IEEE Transactions on Automation Science and Engineering* 17(4), 1759–1776.

Lassila, O., van Harmelen, F., Horrocks, I., Hendler, J. and McGuinness, D. (2000), ‘The semantic web and its languages’, *IEEE Intelligent Systems and their Applications* 15(6), 67–73.

Lim, G. H., Suh, I. H. and Suh, H. (2011), ‘Ontology-based unified robot knowledge for service robots in indoor environments’, *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 41(3), 492–509.

Lomas, M., Moffitt, V. Z., Craven, P., Cross, E. V., Franke, J. L. and Taylor, J. S. (2011), Team-based interactions with heterogeneous robots through a novel hri software architecture, *in* ‘2011 6th ACM/IEEE International Conference on Human-Robot Interaction (HRI)’, pp. 193–194.

Lu, Y., Zhang, C. and Hou, H. (2009), Using multiple hybrid strategies to extract chinese synonyms from encyclopedia resource, *in* ‘2009 Fourth International Conference on Innovative Computing, Information and Control (ICICIC)’, pp. 1089–1093.

Luo, R. C. and Chen, C.-J. (2017), Recursive neural network based semantic navigation of an autonomous mobile robot through understanding human verbal instructions, *in* ‘2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)’, pp. 1519–1524.

Ma, Z., Zhu, L., Wang, P. and Zhao, Y. (2019), Ros-based multi-robot system simulator, *in* ‘2019 Chinese Automation Congress (CAC)’, pp. 4228–4232.

Maedche, A., Motik, B., Stojanovic, L., Studer, R. and Volz, R. (2003), ‘Ontologies for enterprise knowledge management’, *IEEE Intelligent Systems* 18(2), 26–33.

Martinez, A., Yannuzzi, M., de Vergara, J. E. L., Serral-Gracia, R. and Ram` ´ırez, W. (2015), An ontology-based information extraction system for bridging the configuration gap in hybrid sdn environments, *in* ‘2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)’, pp. 441–449.

Mazuel, L. and Sabouret, N. (2006), Generic command interpretation algorithms for conversational agents, *in* ‘2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology’, pp. 146–153.

Msala, Y., Hamlich, M. and Mouchtachi, A. (2019), A new robust heterogeneous multirobot approach based on cloud for task allocation, *in* ‘2019 5th International Conference on Optimization and Applications (ICOA)’, pp. 1–4.

Mullers, F., Holz, D. and Behnke, S. (2009), ‘rxDeveloper: GUI-Aided Software Devel-¨ opmemt in ROS’, *Proc. of ICRA Workshop on Software Development and Integration in Robotics (SDIR)* p. 2009.

Muthugala, M. A. V. J. and Jayasekara, A. G. B. P. (2018), ‘A review of service robots coping with uncertain information in natural language instructions’, *IEEE Access* 6, 12913–12928.

Park, J., Lee, B. and Chung, W. (2003), Reflective force navigation control for a mobile robot using a state transition diagram, *in* ‘Proceedings 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003)’, Vol. 1, pp. 52– 57 vol.1.

Pomarlan, M. and Bateman, J. (2018), ‘Robot program construction via grounded natural language semantics and simulation robotics track’, *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS* 2, 857–864.

Rajapaksha, S., Illankoon, V., Halloluwa, N. D., Satharana, M. and Umayanganie, D. (2019), Responsive drone autopilot system for uncertain natural language commands, *in* ‘2019 International Conference on Advancements in Computing (ICAC)’, pp. 232–237.

Rani, V. U., Sridevi, J. and Sai, P. M. (2021), Web controlled raspberry pi robot surveillance, *in* ‘2021 International Conference on Sustainable Energy and Future Electric Transportation (SEFET)’, pp. 1–5.

Rashid, A. T., Frasca, M., Ali, A. A., Rizzo, A. and Fortuna, L. (2015), ‘Multirobot localization and orientation estimation using robotic cluster matching algorithm’, *Robotics and Autonomous Systems* 63, 108–121.

Reid, C., Samanta, B. and Kadlec, C. (2017), Network performance of wireless cloudbased robots with local processing, *in* ‘SoutheastCon 2017’, pp. 1–6.

Rhee, S. K., Lee, K. and Kim, H. (2012), Ontology-based context and preference model for personal service robot, *in* ‘2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)’, pp. 216–217.

Richtr, L. and Farana, R. (2011), Remote control the robot using web service, *in* '2011 12th International Carpathian Control Conference (ICCC)', pp. 326–330.

Rizk, Y., Awad, M. and Tunstel, E. (2019), Cooperative heterogeneous multi-robot systems: A survey, *in* 'ACM Computing Surveys', Vol. 52, pp. 1–31.

Robotics, O. (n.d.), 'Ros navigation stack'. URL: *http:// wiki. ros. org /navigation.*

Ruiz-del Solar, J. and Ruiz-del Solar, J. (2007), Personal robots as ubiquitousmultimedial-mobile web interfaces, *in* '2007 Latin American Web Conference (LAWEB 2007)', pp. 120–127.

Sadeghian, R., Zarei, M., Shahin, S. and Masouleh, M. T. (2017), Vision based control and simulation of a spherical rolling robot based on ros and gazebo, *in* '2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI)', pp. 0304–0309.

Scibilia, A., Pedrocchi, N. and Fortuna, L. (2022), 'Human control model estimation in physical human–machine interaction: A survey', *Sensors* 22(5), 1732.

Sen Gupta, G., Messom, C. and Sng, H. (2002), State transition based supervisory control for a robot soccer system, *in* 'Proceedings First IEEE International Workshop on Electronic Design, Test and Applications '2002', pp. 338–342.

Shen, Y., Li, Y., Deng, Y., Zhang, J., Yang, M., Chen, J., Si, S. and Lei, K. (2018), 'Gastroenterology ontology construction using synonym identification and relation extraction', *IEEE Access* 6, 52095–52104.

Staab, S., Studer, R., Schnurr, H.-P. and Sure, Y. (2001), 'Knowledge processes and ontologies', *IEEE Intelligent Systems* 16(1), 26–34.

Sutherland, C. J. and MacDonald, B. (2019), 'RoboLang: A Simple Domain Specific Language to Script Robot Interactions', *2019 16th International Conference on Ubiquitous Robots, UR 2019* pp. 265–270.

Takaya, K., Asai, T., Kroumov, V. and Smarandache, F. (2016*a*), Simulation environment for mobile robots testing using ros and gazebo, *in* '2016 20th International Conference on System Theory, Control and Computing (ICSTCC)', pp. 96–101.

Takaya, K., Asai, T., Kroumov, V. and Smarandache, F. (2016*b*), Simulation environment for mobile robots testing using ros and gazebo, *in* '2016 20th

International Conference on System Theory, Control and Computing (ICSTCC)', pp. 96–101.

Tiddi, I., Bastianelli, E., Bardaro, G., D'Aquin, M. and Motta, E. (2017), An ontologybased approach to improve the accessibility of ROS-based robotic systems, *in* 'Proceedings of the Knowledge Capture Conference, K-CAP 2017', number December. Tong, Q. (2018), 'Mapping object-oriented database models into rdf(s)', *IEEE Access* 6, 47125–47130.

Velamala, S. S., Patil, D. and Ming, X. (2017), Development of ros-based gui for control of an autonomous surface vehicle, *in* '2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)', pp. 628–633.

Yao, W., Dai, W., Xiao, J., Lu, H. and Zheng, Z. (2015), A simulation system based on ros and gazebo for robocup middle size league, *in* '2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)', pp. 54–59.

Yun-hua, G. and Dan, L. (2010), Web resources description model based on rdf, *in* '2010 International Conference on Computer Application and System Modeling (ICCASM 2010)', Vol. 9, pp. V9–222–V9–225.

Zheng, S., Lin, Z., Zeng, Q., Zheng, R., Liu, C. and Xiong, H. (2018), 'Iapcloud: A cloud control platform for heterogeneous robots', *IEEE Access* 6, 30577–30591.

Appendices

Appendix A

Fragment of Ontology

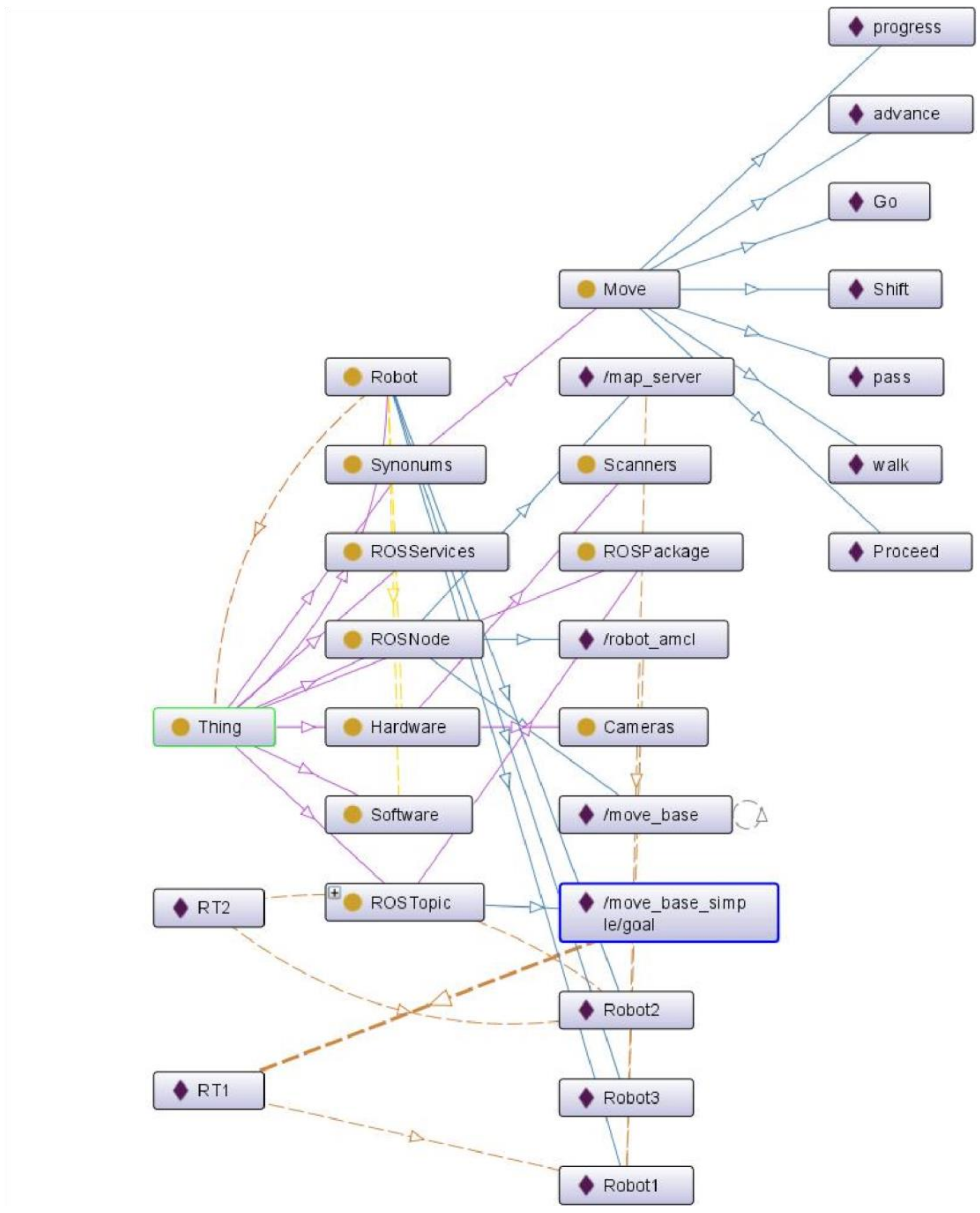


Fig. A.1 Part of the Ontology Developed

A.1 Part of the OWL file Created Using Protege Tool

$<?xmlversion = "1.0"? >$

$<!DOCTY PErdf : RDF[<!ENTITY owl"http : //www.w3.org/2002/07/owl\#" >$

$<!ENTITY xsd"http : //www.w3.org/2001/XMLSchema\#" >$

<!ENTITY rdfs"http : //www.w3.org/2000/01/rdf − schema#" >

<!ENTITY rdf'http : //www.w3.org/1999/02/22 − rdf − syntax − ns#" >

<!ENTITY untitled − ontology − 16

"http : //www.semanticweb.org/staff/ontologies/2021/1/ untitled − ontology − 16#" >

] >

< rdf : RDFxmlns = "http : //www.semanticweb.org/staff/ontologies /2021/1/untitled − ontology − 16#" xml : base = "http :

//www.semanticweb.org/staff/ontologies/2021

/1/untitled − ontology − 16" xmlns : rdfs = "http : //www.w3.org/2000/01/rdf − schema#" xmlns : owl = "http : //www.w3.org/2002/07/owl# xmlns : xsd = "http : //www.w3.org/2001/XMLSchema#" xmlns : rdf = "http : //www.w3.org/1999/02/22 − rdf − syntax − ns# xmlns : untitled − ontology − 16 = "http : //www.semanticweb.org/ staff /ontologies/2021/1/untitled − ontology − 16#" >

< owl : Ontologyrdf : about = "http : //www.semanticweb.org/staff

/2021/1/untitled − ontology − 16"/ >

<!−−///ObjectProperties

/// −− >

<! −−http : //www.semanticweb.org/staff/ontologies/2021/1/ untitled − ontology − 16#cosistof −− >

$< owl : ObjectPropertyrdf : about = "anduntitled - ontology - cosistof" >$ 16;

$< rdfs : rangerdf : resource = "anduntitled - ontology - Hardware"/ >$ 16;

$< rdfs : domainrdf : resource = "anduntitled - ontology -$
$Robot" / >$ 16;
< rdfs : rangerdf : resource = ”anduntitled − ontology − 16;

Software”/ >
< /owl : ObjectProperty >

<! −−http : //www.semanticweb.org/staff/ontologies/2021/1

/untitled − ontology − 16#has −− >

$< owl : ObjectPropertyrdf : about = "anduntitled - ontology -$
$has" >$ 16;
< rdfs : domainrdf : resource = ”anduntitled − ontology − 16;

Robot”/ >
< /owl : ObjectProperty >

<! −−///

Dataproperties
/// −− >

<! −−http : //www.semanticweb.org/staff/ontologies/2021/1

/untitled − ontology − 16#CoG −− >

$< owl : DatatypePropertyrdf : about = "anduntitled - ontology -$
$CoG" >$ 16;
< rdfs : domainrdf : resource = ”anduntitled − ontology − 16;

Robot”/ >
< rdfs : rangerdf : resource = ”andowl;real”/ >
< /owl : DatatypeProperty >

<! −−http : //www.semanticweb.org/staff/ontologies/2021/1/ untitled −

ontology − 16#FootPrint −− >

$< owl : DatatypePropertyrdf : about = "anduntitled - ontology -$
$FootPrint" >$ 16;
< rdfs : domainrdf : resource = ”anduntitled − ontology − 16;

Robot"/ >
< rdfs : rangerdf : resource = "andxsd;string"/ >
< /owl : DatatypeProperty >

<! −−*http : //www.semanticweb.org/staff/ontologies/*2021/1/ *untitled* −

ontology − 16#*IPAddress* −− >

$< owl : DatatypePropertyrdf : about = "anduntitled - ontology - IPAddress" >$ 16;
< rdfs : domainrdf : resource = "anduntitled − ontology − 16;

Robot"/ >
< rdfs : rangerdf : resource = "andxsd;decimal"/ >
< /owl : DatatypeProperty >

<! −−*http : //www.semanticweb.org/staff/ontologies/*2021/1/ *untitled* −

ontology − 16#*PortNumber* −− >

$< owl : DatatypePropertyrdf : about = "anduntitled - ontology - PortNumber" >$ 16;
< rdfs : domainrdf : resource = "anduntitled − ontology − 16;

Robot"/ >
< rdfs : rangerdf : resource = "andxsd;integer"/ >
< /owl : DatatypeProperty >
$<! - -http : //www.semanticweb.org/staff/ontologies/2021/1/ untitled - ontology - 16\#ROSNode\_NAme - - >$

$< owl : DatatypePropertyrdf : about = "anduntitled - ontology - ROSNode\_NAme"/ >$ 16;

<! −−*http : //www.semanticweb.org/staff/ontologies/*2021/1/ *untitled* −

ontology − 16#*ROSTopicName* −− >

$< owl : DatatypePropertyrdf : about = "anduntitled - ontology - ROSTopicName"/ >$ 16;

<! −−*http* : *//www.semanticweb.org/staff/ontologies/*2021/1/ *untitled* −

ontology − 16#*RobotName* −− >

< *owl* : *DatatypePropertyrdf* : *about* = ”*anduntitled* − *ontology* −
RobotName” > 16;
< *rdfs* : *domainrdf* : *resource* = ”*anduntitled* − *ontology* − 16;

Robot”/ >
< *rdfs* : *rangerdf* : *resource* = ”*andxsd;string*”/ >
< /*owl* : *DatatypeProperty* >
<! −−*http* : *//www.semanticweb.org/staff/ontologies/*2021/1/ *untitled* −

ontology − 16#*move* −− >

< *owl* : *DatatypePropertyrdf* : *about* = ”*anduntitled* − *ontology* − 16; *move*”

>

< *rdfs* : *rangerdf* : *resource* = ”*andxsd;string*”/ >
< /*owl* : *DatatypeProperty* >

<! −−*http* : *//www.semanticweb.org/staff/ontologies/*2021/

1/*untitled* − *ontology* − 16#*navigation* −− >

< *owl* : *DatatypePropertyrdf* : *about* = ”*anduntitled* − *ontology* − 16;

navigation” >

< *rdfs* : *rangerdf* : *resource* = ”*andxsd;string*”/ >
< /*owl* : *DatatypeProperty* >

<! −−*http* : *//www.w3.org/*2002/07/*owl*#*topDataProperty* −− >

< *rdf* : *Descriptionrdf* : *about* = ”*andowl;topDataProperty*” >
< *rdfs* : *subPropertyOf rdf* : *resource* = ”*anduntitled* − *ontology* −

16;*ROSTopicName*”/ > < /*rdf* : *Description* >

<! −−*//*

Classes

```
//////////////////////////////////////////////////// −− >

    <! −−http : //www.semanticweb.org/staff/ontologies/2021/1 /untitled −
ontology − 16#Cameras −− >

    < owl : Classrdf : about = ”anduntitled − ontology − 16;Cameras” >
< rdfs : subClassOfrdf : resource = ”anduntitled − ontology − 16;
Hardware”/ >
< /owl : Class >

    <! −−http : //www.semanticweb.org/staff/ontologies/2021/1/ untitled −
ontology − 16#Hardware −− >
    < owl : Classrdf : about = ”anduntitled − ontology − 16;Hardware”/ >

    <! −−http : //www.semanticweb.org/staff/ontologies/2021/1/ untitled −
ontology − 16#Move −− >

    < owl : Classrdf : about = ”anduntitled − ontology − 16;Move” >
< rdfs : subClassOfrdf : resource = ”anduntitled − ontology − 16;
Synonums”/ >
< /owl : Class >

    <! −−http : //www.semanticweb.org/staff/ontologies/2021/1/ untitled −
ontology − 16#ROSNode −− >

    < owl : Classrdf : about = ”anduntitled − ontology − 16;ROSNode”/ >

    <! −−http : //www.semanticweb.org/staff/ontologies/2021/1/ untitled −
ontology − 16#ROSPackage −− >

    < owl : Classrdf : about = ”anduntitled − ontology − 16;ROSPackage” >
< rdfs : subClassOfrdf : resource = ”anduntitled − ontology − 16;
ROSTopic”/ >
< /owl : Class >

    <! −−http : //www.semanticweb.org/staff/ontologies/2021/1/ untitled −
ontology − 16#ROSServices −− >

    < owl : Classrdf : about = ”anduntitled − ontology − 16;ROSServices”/ >

    <! −−http : //www.semanticweb.org/staff/ontologies/2021/1/ untitled −
ontology − 16#ROSTopic −− >

    < owl : Classrdf : about = ”anduntitled − ontology − 16;ROSTopic” >
< rdfs : subClassOfrdf : resource = ”anduntitled − ontology − 16;
ROSPackage”/ >
< /owl : Class >

    <! −−http : //www.semanticweb.org/staff/ontologies/2021/1
/untitled − ontology − 16#Robot −− >
    < owl : Classrdf : about = ”anduntitled − ontology − 16;Robot”/ >

    <! −−http : //www.semanticweb.org/staff/ontologies/2021/1/ untitled −
ontology − 16#Scanners −− >

    < owl : Classrdf : about = ”anduntitled − ontology − 16;Scanners” >
< rdfs : subClassOfrdf : resource = ”anduntitled − ontology − 16;
Hardware”/ >
< /owl : Class >

    <! −−http : //www.semanticweb.org/staff/ontologies/2021/1/ untitled −
ontology − 16#Software −− >
```

< owl : *Classrdf* : *about* = ”*anduntitled* − *ontology* − 16;*Software*”/ >

<! −−*http* : //*www.semanticweb.org*/*staff*/*ontologies*/2021/1/ *untitled* − *ontology* − 16#*Synonums* −− >

< owl : *Classrdf* : *about* = ”*anduntitled* − *ontology* − 16;*Synonums*”/ >

<! −−///
Individuals
/// −− >

<! −−*http* : //*www.semanticweb.org*/*staff*/*ontologies*/2021/1
/*untitled* − *ontology* − 16#/*cmd* −− >

$< owl : NamedIndividualrdf : about = ”anduntitled - ontology -$ 16;
$/cmd” >$
< *rdf* : *typerdf* : *resource* = ”*anduntitled* − *ontology* − 16;
ROSTopic”/ >
< *moverdf* : *datatype* = ”*andxsd*;*string*” >
25< /*move* >
< *cosistofrdf* : *resource* = ”*anduntitled* − *ontology* − 16;/*cmd*”/ >
< /*owl* : *NamedIndividual* >
$<! - -http : //www.semanticweb.org/staff/ontologies/2021/1$
$usepackage/untitled - ontology - 16\#/cmd\_vel - - >$
$< owl : NamedIndividualrdf : about = ”anduntitled - ontology -$ 16;
$/cmd\_vel” >$
< *rdf* : *typerdf* : *resource* = ”*anduntitled* − *ontology* − 16;
ROSTopic”/ >
< *moverdf* : *datatype* = ”*andxsd*;*string*” >
$24< /move >$
$< cosistofrdf : resource = ”anduntitled - ontology -$
$/cmd\_vel”/ >$
$< /owl : NamedIndividual >$ 16;

<! −−*http* : *//www.semanticweb.org/staff/ontologies/*2021/1 */untitled* − *ontology* − 16#*/map* −− >

< *owl* : *NamedIndividualrdf* : *about* = ”*anduntitled*−*ontology*−16;*/map*” >
< *rdf* : *typerdf* : *resource* = ”*anduntitled* − *ontology* − 16;*ROSTopic*”/ >
< *moverdf* : *datatype* = ”*andxsd;string*” > map<
/move >
$< cosistofrdf : resource = ”anduntitled - ontology - 16; /map”/ >$
$< /owl : NamedIndividual >$

$<! - -http : //www.semanticweb.org/staff/ontologies/2021/1/$
$untitled - ontology - 16\#/map\_server - - >$

$< owl : NamedIndividualrdf : about = ”anduntitled - ontology -$
$/map\_server” >$
$< rdf : typerdf : resource = ”anduntitled \quad ontology \quad 16$
16;

– –

;*ROSNode*”/ >
< *RobotNamerdf* : *datatype* = ” > *string*” > *Turtle*
$< hasrdf : resource = ”anduntitled - ontology - 16; Robot1”/ >$
$< /owl : NamedIndividual >$

$<! - -http : //www.semanticweb.org/staff/ontologies/2021/1/$
$untitled - ontology - 16\#/move\_base - - >$

$< owl : NamedIndividualrdf : about = ”anduntitled - ontology - 16; /$
$move\_base” >$
$< rdf : typerdf : resource = ”anduntitled - ontology -$
$ROSNode”/ >$
16;
$< moverdf : datatype = ”andxsd; string” >$
$24< /move >$
$< cosistofrdf : resource = ”anduntitled - ontology -$
$/move\_base”/ >$
$< /owl : NamedIndividual >$
16;
$<! - -http : //www.semanticweb.org/staff/ontologies/2021/1/$
$untitled - ontology - 16\#/move\_base\_simple/goal - - >$

$< owl : NamedIndividualrdf : about = "anduntitled - ontology -$ 16;
$/goal" >$

$< rdf : typerdf : resource = "anduntitled - ontology - 16;$

$< hasrdf : resource = "anduntitled - ontology - 16;RT1"/ >$

$<! --http : //www.semanticweb.org/staff/ontologies/2021/1/ untitled -$

$ontology - 16\#/odem -- >$

$< owl : NamedIndividualrdf : about = "anduntitled - ontology -$ 16;
$/odem" >$

$< rdf : typerdf : resource = "anduntitled - ontology - 16;$

$ROSTopic"/ >$

$< RobotNamerdf : datatype = "andxsd;string" >$

$< /RobotName >$

$< moverdf : datatype = "andxsd;string" >$

$< /move >$

$< cosistofrdf : resource = "anduntitled - ontology - 16;$

$/odem"/ >$

$< /owl : NamedIndividual >$

$<! - -http : //www.semanticweb.org/staff/ontologies/2021$
$/1/untitled - ontology - 16\#/robot\_amcl - - >$

$< owl : NamedIndividualrdf : about = "anduntitled - ontology -$ 16;
$/robot\_amcl" >$

$< rdf : typerdf : resource = "anduntitled - ontology -$ 16;
$ROSNode"/ >$

$< hasrdf : resource = "anduntitled - ontology -$ 16;
$Robot1"/ >$

$< /owl : NamedIndividual >$

$<! --http : //www.semanticweb.org/staff/ontologies/$

2021/1

$/untitled - ontology - 16\#Go -- >$

$< owl : NamedIndividualrdf : about = "anduntitled - ontology -$
$16; Go" >$

$< rdf : typerdf : resource = "anduntitled - ontology -$
$Move" / >$ 16;

< *owl* : *sameAsrdf* : *resource* = ”*anduntitled* − *ontology* − 16; *advance*”/ >

< /*owl* : *NamedIndividual* >

<! −−*http* : //*www.semanticweb.org/staff/ontologies*

/2021/1/ *untitled* − *ontology* −
16#*Proceed* −− >

$< owl : NamedIndividualrdf : about = "anduntitled - ontology -$
$16; Proceed" >$
$< rdf : typerdf : resource = "anduntitled - ontology -$
$Move" / >$ 16;

< *owl* : *sameAsrdf* : *resource* = ”*anduntitled* − *ontology* − 16; *advance*”/ >

< /*owl* : *NamedIndividual* >

<! −−*http* : //*www.semanticweb.org/staff/ontologies/*2021 /1/*untitled* −

ontology − 16#*RT*1 −− >

< *owl* : *NamedIndividualrdf* : *about* = ”*anduntitled* − *ontology* − 16;

*RT*1” >

< *ROSTopicNamerdf* : *datatype* = ” > *string*” > *cmd*

$rdf : datatype = " \ string" >$
cmd_vel
$< hasrdf : resource = "anduntitled - ontology - 16; Robot1" / >$
$< /owl : NamedIndividual >$

<! −−*http* : //*www.semanticweb.org/staff/ontologies/*2021/1 /*untitled* −
ontology − 16#*RT*2 −− >

< *owl* : *NamedIndividualrdf* : *about* = ”*anduntitled*−*ontology*−16;*RT*2” >

< *ROSTopicNamerdf* : *datatype* = ” > *string*” > *odem* <
/*ROSTopicName* >

< *ROSTopicNamerdf* : *datatype* = ” *string*” > *odometry* <
/*ROSTopicName* >

< *hasrdf* : *resource* = ”*anduntitled* − *ontology* − 16;*Robot*2”/ >

< /owl : NamedIndividual >

<! −−http : //www.semanticweb.org/staff/ontologies/2021 /1/untitled − ontology − 16#Robot1 −− >

< owl : NamedIndividualrdf : about = ”anduntitled − ontology − 16;Robot1” >

< rdf : typerdf : resource = ”anduntitled − ontology − 16;

Robot”/ >

< /owl : NamedIndividual >

<! − −http : //www.semanticweb.org/staf f/ontologies/ 2021/1/untitled − ontology − 16#Robot2 − − >

< owl : NamedIndividualrdf : about = ”anduntitled − ontology − 16;

Robot2” >

< rdf : typerdf : resource = ”anduntitled − ontology − 16

;Robot”/ >

< IPAddressrdf : datatype = ” > string” >

198.34.56.44 < /IPAddress >

< FootPrintrdf : datatype = ” > string” >

34 ∗ 45 < /FootPrint >

< PortNumberrdf : datatype = ” > string” >

45676 < /PortNumber >

< RobotNamerdf : datatype = ” > string” >

Husky < /RobotName >

< hasrdf : resource = ”anduntitled − ontology − 16;RT2”/ >

< /owl : NamedIndividual >

<! −−http : //www.semanticweb.org/staff/ontologies/

2021/1/untitled − ontology − 16#Robot3 −− >

< owl : NamedIndividualrdf : about = ”anduntitled − ontology − 16

;Robot3” >

< rdf : typerdf : resource = ”anduntitled − ontology − 16;

Robot"/ >
< *IPAddressrdf* : *datatype* = " > *string*" >
193.45.67.77 < /*IPAddress* >
< *FootPrintrdf* : *datatype* = " > *string*" >
34 ∗ 56 < /*FootPrint* >

< *PortNumberrdf* : *datatype* = " > *string*" >
56563 < /*PortNumber* >
< *RobotNamerdf* : *datatype* = " > *string*" >
TiaGo < /*RobotName* >
< /*owl* : *NamedIndividual* >

<! −−*http* : //*www.semanticweb.org/staff/ontologies/* 2021/1/*untitled* −

ontology − 16#*Shift* −− >

< *owl* : *NamedIndividualrdf* : *about* = "*anduntitled* − *ontology* −

16;*Shift*" >

< *rdf* : *typerdf* : *resource* = "*anduntitled* − *ontology* − 16;

Move"/ >
< *owl* : *sameAsrdf* : *resource* = "*anduntitled* − *ontology* − 16; *advance*"/ >

< /*owl* : *NamedIndividual* >

<! −−*http* : //*www.semanticweb.org/staff/ontologies/*

2021/1/*untitled* − *ontology* − 16#*advance* −− >

$< owl : NamedIndividualrdf : about = "anduntitled - ontology -$
$advance" >$ 16;
$< rdf : typerdf : resource = "anduntitled - ontology -$
$Move"/ >$ 16;
$< owl : sameAsrdf : resource = "anduntitled - ontology -$
$pass"/ >$ 16;
$< owl : sameAsrdf : resource = "anduntitled - ontology -$
$progress"/ >$ 16;
$< owl : sameAsrdf : resource = "anduntitled - ontology -$
$walk"/ >$ 16;
< /*owl* : *NamedIndividual* >

<! −−http : //www.semanticweb.org/staff/ontologies/2021 /1/untitled − ontology − 16#pass −− >

< owl : NamedIndividualrdf : about = ”anduntitled − ontology − 16; pass” >
< rdf : typerdf : resource = ”anduntitled − ontology − 16;Move”/ >
< /owl : NamedIndividual >

<! −−http : //www.semanticweb.org/staff/ontologies
/2021/1/untitled − ontology − 16#progress −− >

< owl : NamedIndividualrdf : about = ”anduntitled − ontology − 16;
progress” >
< rdf : typerdf : resource = ”anduntitled − ontology − 16
;Move”/ >
< /owl : NamedIndividual >
<! − −http : //www.semanticweb.org/staff/ontologies/
2021/1/untitled − ontology − 16#walk − − >

< owl : NamedIndividualrdf : about = ”anduntitled − ontology − 16; walk” >
< rdf : typerdf : resource = ”anduntitled − ontology − 16;Move”/ >
< /owl : NamedIndividual >
< /rdf : RDF >

Appendix B

Selected Robots with ROS Topics

Robot	**Ros Topic for movement**	**The message format for the topic**
Turtlebot	cmd_vel - *ROS Hydro and later* command_velocity- *For ROS Groovy and earlier* *cmd_vel_mux*	Vector3 linear Vector3 angular

 Husky	joy_teleop/cmd_vel twist_marker_server/cmd_vel move_base/cmd_vel	Vector3 linear Vector3 angular
iRobot 	cmd_vel (geometry_msgs/*Twist)* *cmd_vel_mux*	Vector3 linear Vector3 angular
Kobuki 	*velocity (geometry_msgs/Twist) cmd_vel (geometry_msgs/Twist)* *cmd_vel_mux*	Vector3 linear Vector3 angular
TIAGo 	*cmd_vel (geometry_msgs/Twist)* *TIAGo specific individual joints* *cmd_vel_mux*	Vector3 linear Vector3 angular
 NAO	cmd_vel (geometry_msgs/Twist) footstep (humanoid_nav_msgs/StepTarget) nao_footstep NAO specific individual joints	Vector3 linear Vector3 angular geometry_msgs/Pose2D pose # step pose as relative offset to last leg uint8 leg # which leg to use (left/right, see below) uint8 right=0 # right leg constant uint8 left=1 # left leg constant

Fig. B.1 Selected Robots with ROS Topics