جامعة الإمام عبدالرحمن بن فيصل
IMAM ABDULRAHMAN BIN FAISAL UNIVERSITY

# COOP
# FINAL REPORT
# 2021

## DEPARTMENT OF Computer Science

Name of the department (CS, CIS, CYS)

COLLEGE OF COMPUTER SCIENCE
AND INFORMATION TECHNOLOGY

IMAM ABDULRAHMAN
BIN FAISAL UNIVERSITY
Kingdom of Saudi Arabia

Building A11 (new campus)
training.ccsit@iau.edu.s
www.iau.edu.sa

**Kingdom of Saudi Arabia**

**Ministry of Higher Education**

**Imam Abdulrahman Bin Faisal University**

**College of Computer Sciences & Information Technology**

# CO-OP Final Report – Summer 2021 at Smart Methods

**by**

**Raghad Esmael Alessa**

**Supervised by**

**CCSIT College Supervisor**

Ms. Abrar M. Alotaibi

**Smart Methods Supervisor**

Eng. Asim Ibrahim

**June 2021 – August 2021**

# ABSTRACT

*As a part of the Computer Science program in CCSIT, I got a CO-OP training opportunity at Smart Methods for ten weeks, starting from 6th June 2021 until 19th August 2021. During the training period, I had the advantage of gaining mixed knowledge where I carried out multiple tasks from interdisciplinary fields.*

*The training experience in such a company as Smart Methods, where the trainees get treated like real developers and face real-life obstacles, is very enriching and pushes their limits to their maximum. This report will show my work in the AI and Robotics track training, and the tasks will be divided based on their scope. Most of the tasks will be in Robotics development. I have experienced the challenges in such a limited resource yet exciting field, like an indoor navigation system for multiple robots. Also, I will discuss my work in AI where I have worked in integrating OpenCV for Robotic vision and machine learning model for Arabic conversation in the Robotic system.*

*In conclusion, I'm glad that I chose this training opportunity that enriched my analytical and search skills and helped shape my future career path.*

# Abbreviations Table

| Abbreviations | Detail |
|---|---|
| CCSIT | College of Computer Science and Information Technology |
| CO-OP | Cooperative Program |
| AI | Artificial Intelligence |
| IoT | Internet of Things |
| ROS | Robotics Operating System |
| SLAM | Simultaneous Localization and Mapping |
| GUI | Graphical User Interface |
| JSON | JavaScript Object Notation |
| NLTK | Natural Language Processing |
| NN | Neural Networks |
| ML | Machine Learning |
| DL | Deep Learning |
| IDE | Integrated Development Environment |
| CLI | Command Line Interpreter |

# TABLE OF CONTENTS



# TABLE OF FIGURES

# LIST OF TABLES

# LIST OF LAUNCH FILES



# LIST OF LINUXS SHELL

# Chapter – 01: Introduction

As part of Imam Abdulrahman Bin Faisal University graduation requirements of College of Computer Science and Information Technology (CCSIT), to complete the Computer Science's bachelor's degree, undergraduate students need to complete 8-10 weeks of summer CO-OP training in a technical company, to apply their theoretical knowledge in a real-life work environment. Thus, I have been accepted into the summer training program in Smart Methods for ten weeks. This chapter will briefly introduce the Smart Method company and where I was assigned to work during my summer CO-OP training.

## 1.1 About Smart Methods

Figure 1 Smart Methods Logo

Smart Methods is one of the first leading companies in Saudi Arabia to specialize in Robotics Engineering, Artificial Intelligence, and Automated Machines. It was founded by Wessam Munshi and established in 2010 as a national commercial corporation and has been registered in the Ministry of Commerce. It is considered the first commercial corporation to serve researchers and innovators in the Arab world. The company was classified by Forbes magazine in 2015 as one of the most innovative companies in the Kingdom of Saudi Arabia. Also, it was a candidate in 2020 by Monsha'at as one of the most innovative companies in the Kingdom of Saudi Arabia and received the "Ebtaker Award 2020". The company provides many services to their customers in industrial manufactures such as 3D printing, Embedded Systems, Electronic Circuits Design, Robot and Control Systems, Application Programming, Artificial Intelligence, and Fuzzy logic applications [1].

## 1.2 CO-OP Assignment in Smart Methods

The company provides multiple training tracks depending on its services: IoT, AI and robotics, Industrial and System Engineering, Energy and Electronics Engineering, and Mechanical Engineering track. The trainee can choose the training they prefer based on their preferences and experience. The training also provides a Full-Stack engineering certificate for those who prefer, that covers tasks from all the tracks mentioned previously. Each summer, the tasks vary depending on the company's current projects' requirements. As for my CO-OP assignment, I have chosen to work In the AI and Robotics track. I was assigned to work mainly on ROS development and in AI mini-projects on this track.

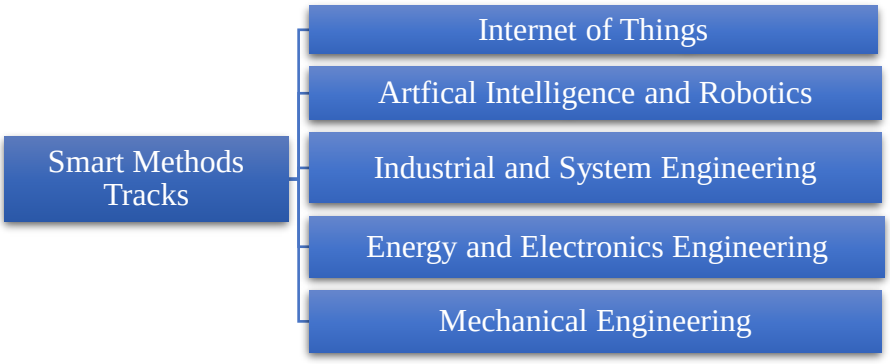



Figure 2 Smart Methods CO-OP training Tracks

In this report, I have discussed all the tasks and activities I have done during my CO-OP training period. Also, all tasks' processes will be clarified with screenshots that will be provided for each task. For tasks codes, the screenshots, Linux shell scripts and launch files will be provided in Appendix A. In the end, I will conclude the report with my CO-OP experience in Smart Methods. Also, the certificates gained and a recommendation letter will be illustrated in Appendix B of the report.

# Chapter – 02: COOP Training

In this chapter, various tasks have been discussed and organized based on the task scope. I will first go through my experience in the Robotics field in which I have carried out various tasks in ROS. Moreover, I will go through the Artificial Intelligence mini-projects, specifically in Machine Learning and Deep Learning. Lastly, I will discuss other learning opportunities I had in the company.

## 2.1 Robotics Tasks

Working in robotics development is challenging due to its limited resources, yet; it's an exciting field to tackle. As more robots are used by the military, industrial, commercial, and service use, the need for more robots' developers is increasing exponentially. Many developers don't like to work in robotics development – me as well- to deal with electronics. Still, thanks to the Robot Operating System, this isn't needed currently [2]. ROS is a meta-OS; it runs on top of Linux Ubuntu, which can abstract the hardware from the software. With that have been said, I'd like to explain the ROS environment at first briefly. As figure 3 shows, this is the architecture of ROS and the relationship between the file systems, and it's also an example of a ROS project. As we can see, all the packages we create need to be located in the source folder. As for build and devel, build is for the cmake and catkin cache, while devel is the target file. The package files are created depending on what dependencies we installed in the creation process [3].

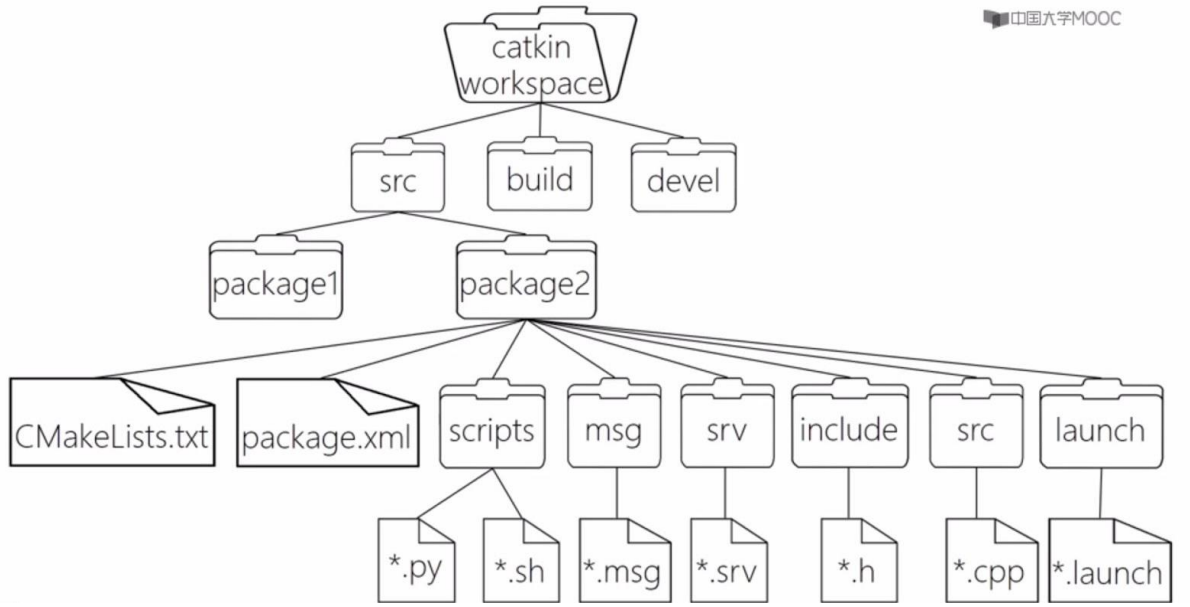
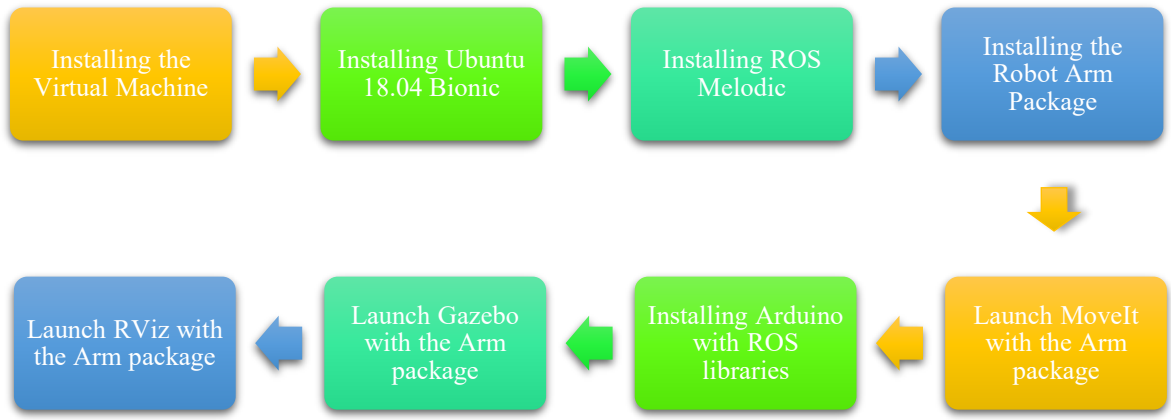



Figure 3 ROS Architecture

Table 1: Task 1 ROS and Robot Arm Package Installation and Configuration

| **Task 1.1: ROS and Robot Arm Package Installation and Configuration** | |
|---|---|
| Task Description | Tools Applied |
| In order to work with robots in the real world, Robot Operating System must be installed to be connected to the hardware parts through hardware programs, e.g., Arduino.<br>After installing ROS, the Arduino Robot Arm package from Smart Methods' GitHub must be installed into the ROS workspace by trainees to control it inside ROS environment and programs, e.g., Rviz, Gazebo, MoveIt. | • Smart Methods Arduino Robot Arm package<br>• Ubuntu 18.04 Bionic<br>• ROS Melodic<br>• Linux shell<br>• MoveIt<br>• Arduino 1.8.15 |
| Task Process | |




The first task in the AI and Robotics track is installing and configuring ROS melodic with Ubuntu 18.04 on the virtual machine (or the host machine). Since ROS is a meta-operating system, launching the programs and navigating its file system will be through the Linux shell, it doesn't have a regular GUI. After preparing the ROS environment and creating the catkin workspace, the workspace is now ready for cloning/creating any package to work with inside ROS. Then, to save time and work directly on the artificial intelligence part instead of working from scratch on the Robot Arm, the company has a pre-configured package, Arduino Robot Arm package that needs to be installed inside the catkin workspace to be used and controlled with multiple programs.

```
raghdution@raghdution-VirtualBox:~$ rosnode list
/rosout
raghdution@raghdution-VirtualBox:~$ rostopic list
/rosout
/rosout_agg
raghdution@raghdution-VirtualBox:~$
```

Figure 4 ROS Installation Successfully

Then, launching the package in MoveIt kinematics. After that, Installing Arduino with the **rosserial** package and **ros_lib** library to connect the Robot arm to the hardware and see its motion in real-life.

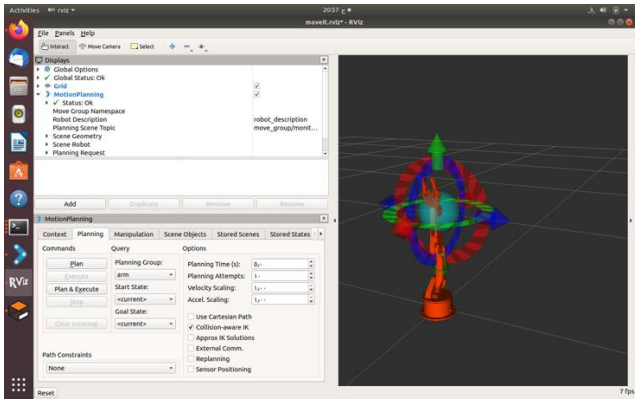


Figure 5 Arduino Arm package in MoveIt

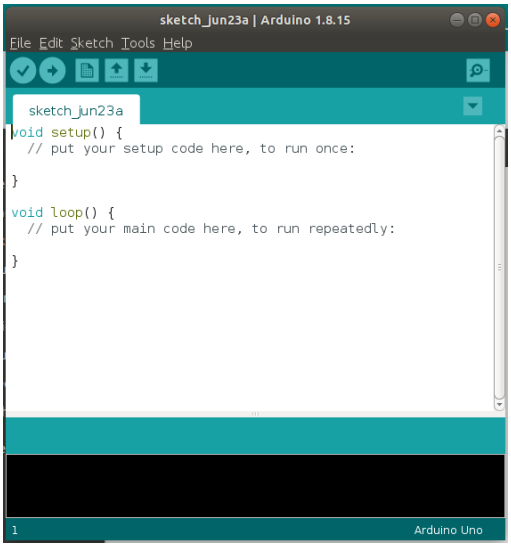



Figure 6 Arduino Installation

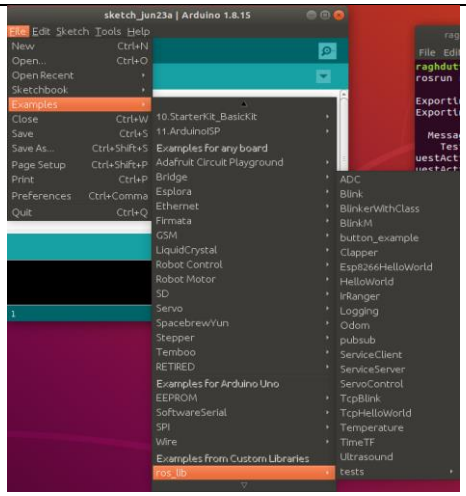



Figure 7 ros_lib Installation

| Challenges faced | Lessons learned |
|---|---|
| • After redownloading Ubuntu 20.04 LTS again from OS course, I deleted it because ROS Melodic works well only on Ubuntu 18.4 Bionic, otherwise you'll face problems.<br>• Problems with file permissions; solved by using **$ sudo nano**.<br>• Most ROS commands and installation processes takes a very long time. | • Package creation<br>• Package cloning<br>• New permission changing commands<br>• ROS commands<br>• ROS packages and programs<br>• MoveIt assistant |
| **Supporting academic courses** | **Suggestions** |
| Operating Systems. | Create a Robotics club in the university. |
| Task accomplished ☒ Task partially accomplished☐ Task unaccomplished☐ | |
| **Self-evaluation and Reasoning** | |
| Many students from other universities have struggled with the installation process for more than 5 weeks, when in the contrary, I have done it in three days due to my laptop inefficiency. Also, I noticed that trainees in my program (computer science) were unfamiliar with working on the CLI. Some trainees changed their track or withdraw from the training from the first week, because of this task. | |

Table 2: Task 1.2 Controlling the Arm Motors in Simulation using joint_state_publisher

| **Task 1.2: Controlling the Arm Motors in Simulation using joint_state_publisher** | |
|---|---|
| Task Description | Tools Applied |

| | |
|---|---|
| After learning how to launch ROS programs with the Robot Arm package MovIt. The goal of the task is launch the package in RViz for 3D visualization, Gazebo for real world simulation, to control the motors in simulation using the joint_state_publisher. | • Smart Methods Arduino Robot Arm package<br>• Ubuntu 18.04 Bionic<br>• ROS Melodic<br>• Linux shell<br>• RViz (3D visualization tool for ROS**)**<br>• Gazebo 9.0 (Robot Simulation) |

## Task Process

After Installing and configuring all the applications, packages needed, it's time to launch the Arm package on both RViz and Gazebo, figure 8 and 9. Using a script inside the package, I managed to control the arm motors in the two simulations, RViz and Gazebo at the same time, using joint_state_publisher package.

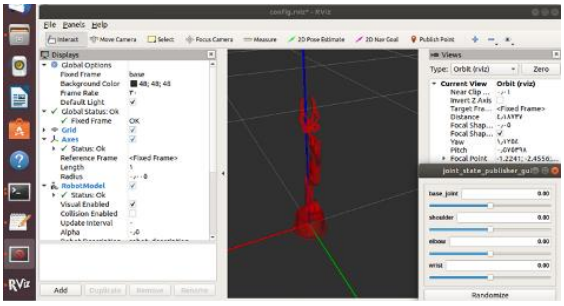


Figure 8 Arduino Arm package in RViz

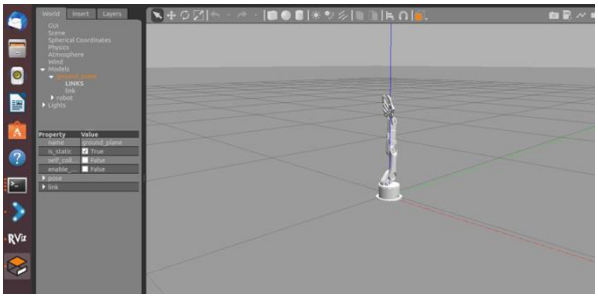


Figure 9 Arduino Arm package in Gazebo

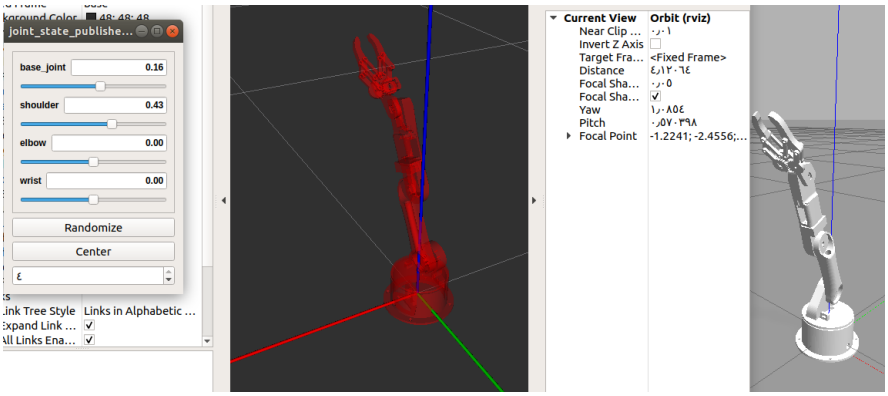



Figure 10 Controlling motors in simulation

This task is important to get used to RViz and Gazebo Interfaces, learn how to rotate or change the view perspective. And, to see the difference between RViz and Gazebo simulations, which like how Morgan Quigley (one of the original developers of ROS) stated in his book: "**rviz** shows you what the robot *thinks* is happening, while **Gazebo** shows you what is *really*

happening" [4]. Also, learning how to control two motors in two different simulations and the importance of joint_State_publisher.

| Challenges faced | Lessons learned |
|---|---|
| Running Gazebo made my laptop halt for an unknown period, its a very heavy program and needs more RAM. So, I increased the VM RAM. | • How to control two simulations at the same time using **joint_State_publisher**<br>• Working with RViz and Gazebo<br>• Difference between RViz and Gazebo |
| **Supporting academic courses** | **Suggestions** |
| Operating Systems. | No suggestions. |

Task accomplished ☒ Task partially accomplished☐ Task unaccomplished☐

**Self-evaluation and Reasoning**

I finished this task in time with no delays and the task was accepted by the company.

| المهمة | رابطها | حالتها | ملاحظات |
|---|---|---|---|
| Robot operating system installation and configuration | أنقر هنا | مقبولة | |
| Arm package on gazebo, RViz and install Arduino | أنقر هنا | مقبولة | |
| Controlling the motors in simulation using the joint_State_publisher with Gazebo and RViz | أنقر هنا | مقبولة | |

Figure 11 Task 1.2 Acceptance

The TurtleBot is a low-cost, open-source personal robot kit. Melonee Wise and Tully Foote constructed TurtleBot in November 2010 at Willow Garage. We can now build a robot that can drive around the house, see in 3D, and generate exciting applications using TurtleBot. TurtleBot has three versions, in our implementation, we'll use the latest one, Turtlebot3. The Turtlebot3 family consists of three robots: Burger, Waffle and Waffle Pi [5].

Table 3: Task 2 Turtlebot3 with SLAM approach

| **Task 2: Turtlebot3 with SLAM approach** | |
|---|---|
| Task Description | Tools Applied |
| Using Turtlebot3 with SLAM approach to create and save a map. | • TurtleBot3<br>• Gazebo<br>• RViz<br>• Linux Shell |
| Task Process | |

In this task we used Turtlebot3 with laser-based SLAM gmapping approach for creating a map of an unfamiliar region while simultaneously, keeping track of any obstacle inside it. After the Robot recognizes the environment with the help of LiDAR to detect objects, determine the map ranges, and create obstacle avoidance; a map will be generated and saved upon a user's request.

I managed to move around with the robot and navigate the whole map with Gazebo and RViz at the same time. Then, I got this map as a result.

After launching TurtleBot3 simulation World on Gazebo, figure 12, I ran the SLAM node to launch the Turtlebot3 on RViz, figure 13. And managed to move around the whole map using the teleoperation node. After collecting all the information needed of the environment, I ran **map_saver** from the **map_server** package to automatically save the generated map with all its ranges and the agents, inside a file.

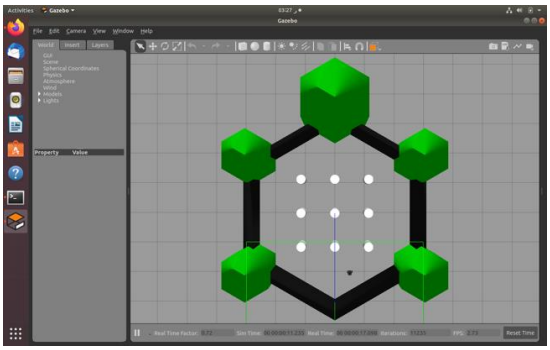


Figure 12 TurtleBot3 World

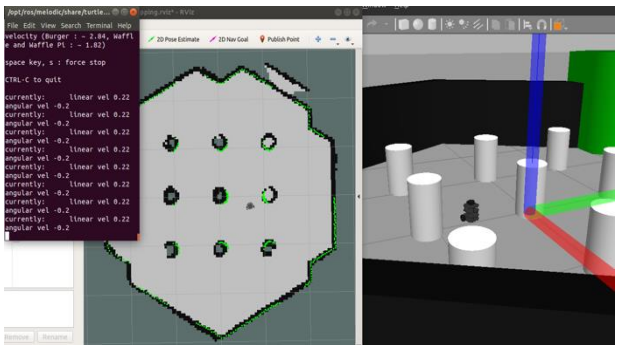


Figure 13 SLAM simulation

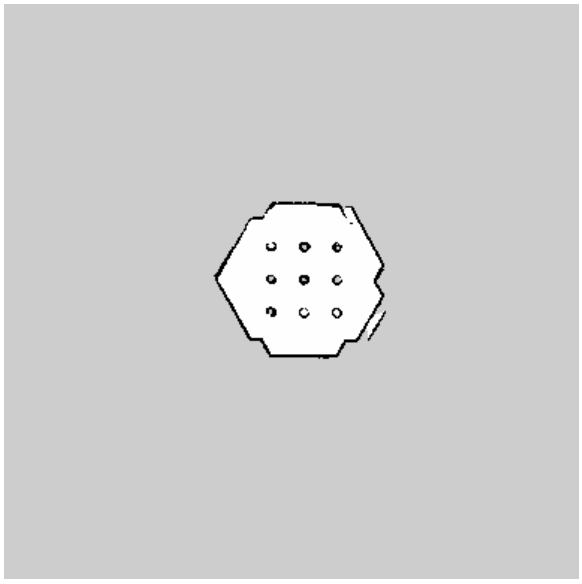


Figure 14 Task 2 Map

| Challenges faced | Lessons learned |
|---|---|
| I didn't face any challenges. | • Using the **teleop** command to move the robot inside the map<br>• Using the map_server package |

| Supporting academic courses | Suggestions |
|---|---|
| Operating Systems. | No suggestions. |
| Task accomplished ☒ Task partially accomplished☐ Task unaccomplished☐ | |
| Self-evaluation and Reasoning | |
| Taking notes during our weekly meeting saved me from having the same problems, delays and confusion other trainees faced. Thus, this habit always proves its worth. | |

After we learned how to use the Turtlebot3 in generating a map for an unknown environment using SLAM, the training challenged us to do an extensive research and find another ROS community Robot that uses SLAM and experience the challenges many ROS developers face in real world.

Table 4: Task 3 ROS Robot with SLAM Approach

| **Task 3: ROS Robot with SLAM Approach** | |
|---|---|
| Task Description | Tools Applied |
| This week's task an extension to task 2; perform an extensive search and look for another ROS robot that uses SLAM approach to create and save a map. And it needs to meet the basic requirements:<br><br>• Robot Category: Ground<br>• The Robot uses SLAM | • Dpoom Robot<br>• CHAMP Robot<br>• Two-wheeled differential drive Robot<br>• Gazebo<br>• RViz<br>• Linux Shell |
| Task Process | |
| Initially, I have searched for the task solution that meets the basic requirements in lots of resources, but I have forgot about the implicit requirements. As ROS develops day by day, many versions and many supporting programs requires regular updating, since it's an open-source operating system, problems and errors significantly arise. Most packages found were working on different versions of ROS, the instructions for the installation process of Robots' packages weren't clear at all in many resources, even when users sometimes report an issue, some developers don't replay.<br><br>Robots' package I tried and weren't successful:<br><br>1. Dpoom | |

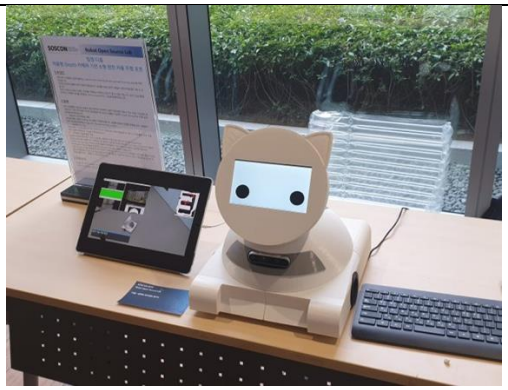


Figure 15 Dpoom in real life.

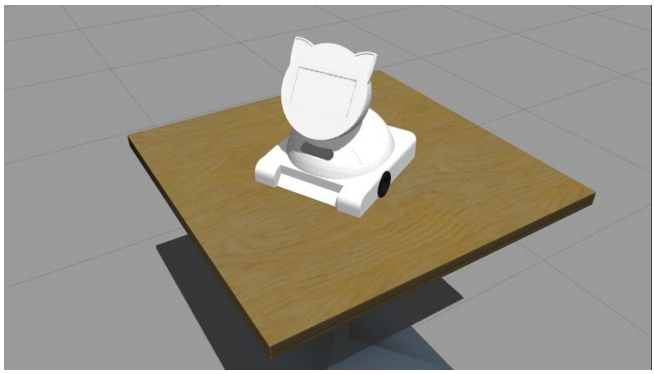


Figure 16 Dpoom in Gazebo

2. CHAMP

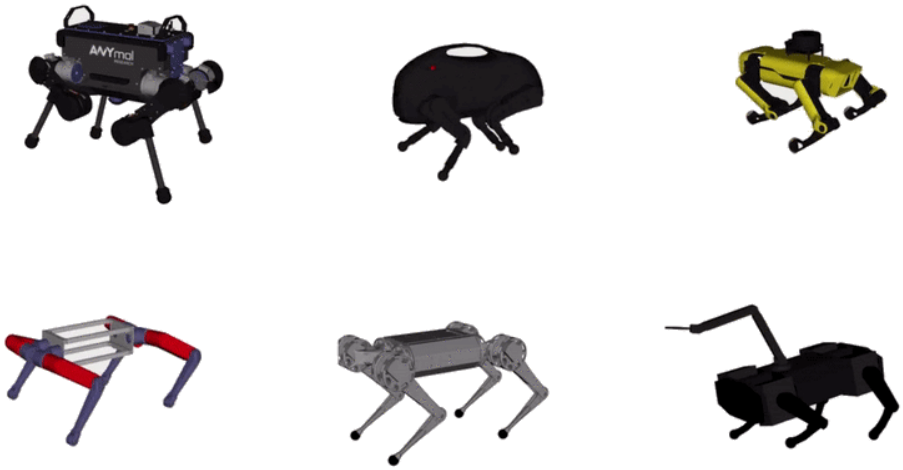


Figure 17 CHAMP in real life

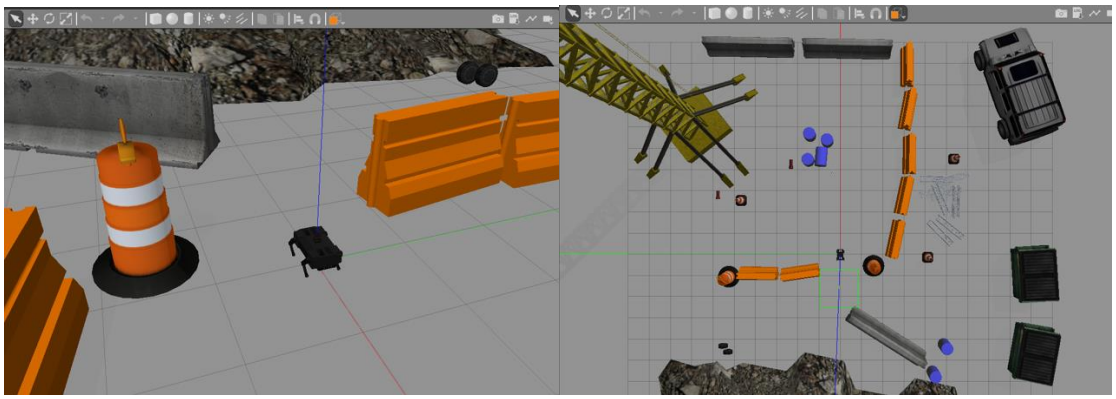


Figure 18 CHAMP in Gazebo

Successful robot package:

3. Two-wheeled differential drive Robot

I found a custom two-wheeled differential drive robot, created by the ROS developer: Devansh Dhrafani, that implements **slam_gmapping** and ROS navigation.

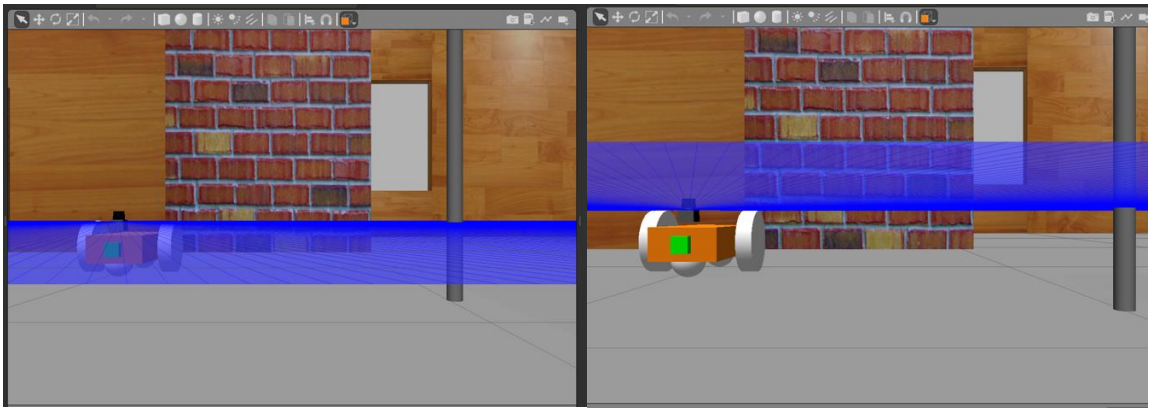


Figure 19 two-wheeled differential drive in Gazebo

The installation steps were clear and to the point, with no massive errors. The steps are as follows; First, launching the package in Gazebo, and opening the default environment turtlebot3_house. Then, running the package on RViz for simultaneous motion and clear view.

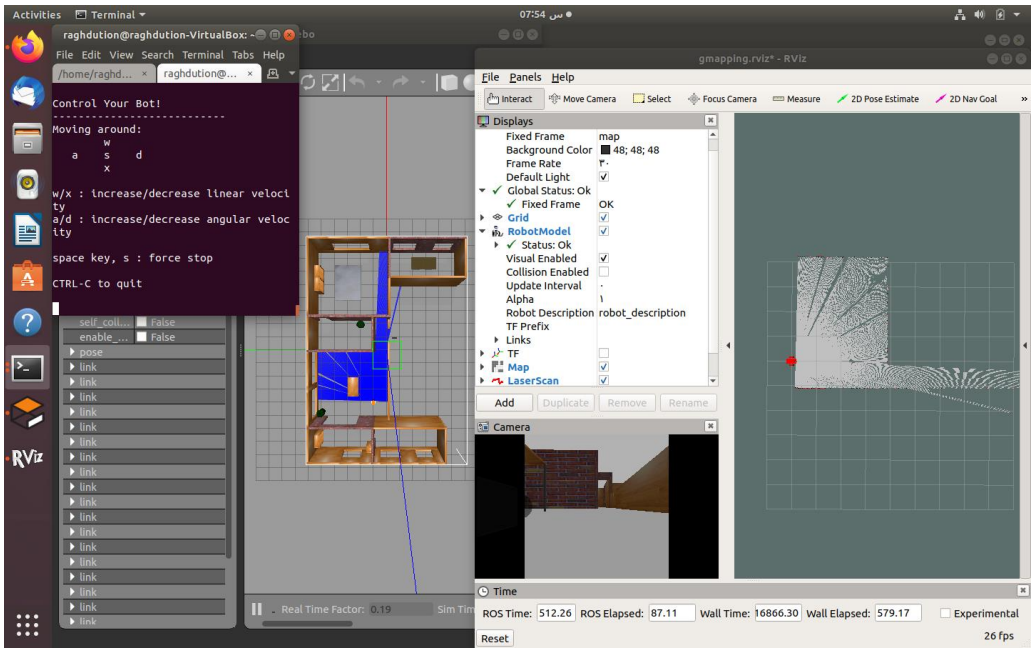



Figure 20 Task 3 RViz and Gazebo

Lastly, to generate the map, I launched the teleoperation node and moved the robot around the whole map to generate it and then, saved it in a file.

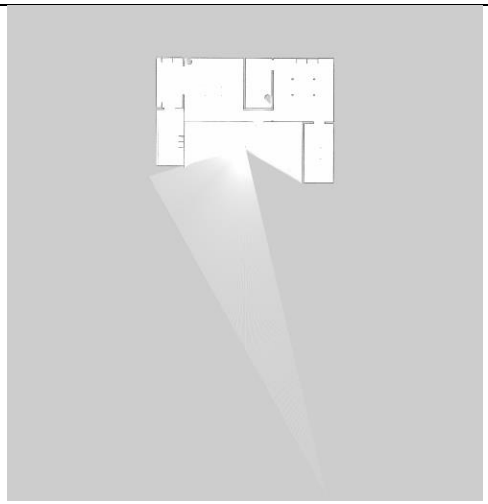


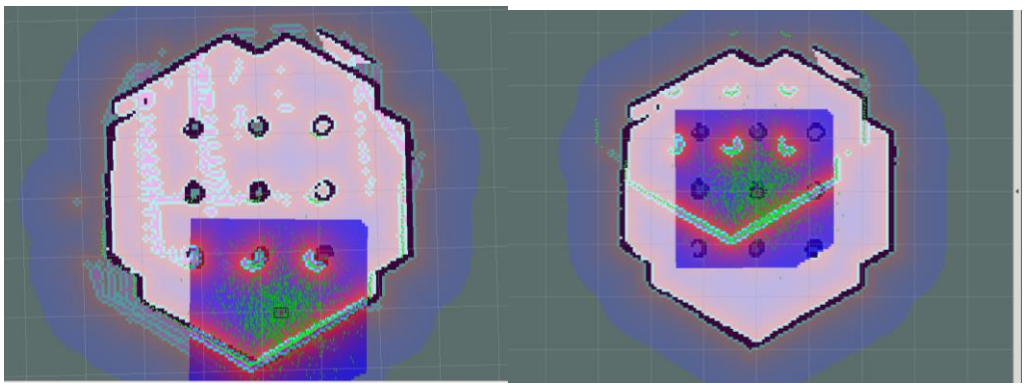
Figure 21 Task 3 map

| Challenges faced | Lessons learned |
|---|---|
| • Gazebo halted for many hours. Faced error **'[Err] [REST.cc:205] Error in REST request**': Solved by changing the server in 'config.yaml' to: url: https://api.ignitionrobotics.org<br>• For Dpoom; When I ran the teleoperation node, the Robot wasn't moving at all, and in RViz, the Robot model wasn't visible. I faced a lot of errors even when I was following the exact instructions.<br>• For CHAMP, at first, the environment wasn't visible in RViz, and the robot model was full of errors. I saw multiple users wrote issues about the robot model errors, but the authors weren't exactly responding. Also, the Robot wasn't moving in Gazebo using the teleop command. | • Experiencing ROS developers' challenges<br>• More ROS robots<br>• Understood ROS packages more deeply<br>• More capable to look for solutions<br>• The computer needs rest just as much as we humans need it<br>• Never give up |
| **Supporting academic courses** | **Suggestions** |
| Operating Systems. | I would suggest for developers to be very precise about their packages requirements and update them regularly. |
| Task accomplished ☒ Task partially accomplished☐ Task unaccomplished☐ | |
| **Self-evaluation and Reasoning** | |

Many trainees have tried one robot package, my passion didn't let me stop there. I was curious and excited to try more packages and solve errors even though some of them were beyond my knowledge.

After learning how to control the Robot's movement using the Teleop command (keyboard|), we are now asked to launch the Robot's navigation using the SLAM map we created before in task 2.

Table 5: Task 4 Launch the Navigation with TurtleBot3

| **Task 4: Launch the Navigation with TurtleBot3** | |
|---|---|
| Task Description | Tools Applied |
| Following task 2, saving the SLAM map after moving the robot around in it. This task is about using the same SLAM map to launch the navigation with RViz and Gazebo together. | • TurtleBot3<br>• Gazebo<br>• RViz<br>• Linux Shell |
| Task Process | |
| First, setting an initial pose estimation by making the robot move in the map to collect the environment information, avoid obstacles, and until the LDS sensor data is overlayed on the saved map.<br>  <br>Figure 22 one robot initial pose istimate<br>Then, after running the teleoperation node and setting the navigation goal, the robot can navigate the map by itself with the goal and direction specified and avoid the obstacles collected from navigating the environment. | |

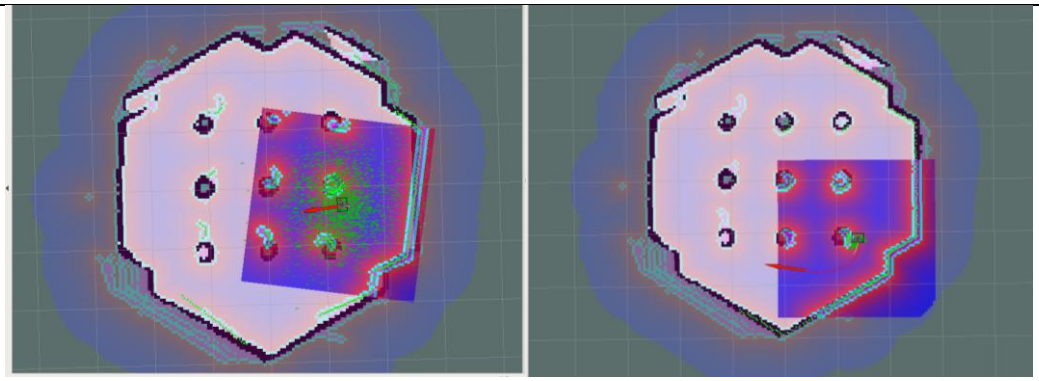


Figure 23 one robot navigation

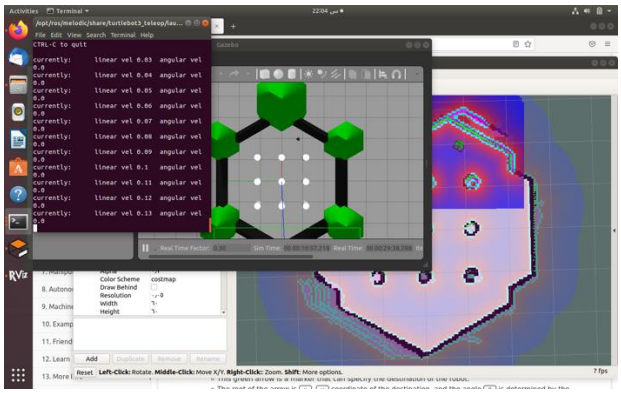


Figure 24 Navigation in both RViz and Gazebo

| Challenges faced | Lessons learned |
|---|---|
| No challenges faced. | • Launching the navigation in RViz using 2D Nav Goal |
| **Supporting academic courses** | **Suggestions** |
| Operating systems. | No suggestions. |
| Task accomplished ☒ Task partially accomplished☐ Task unaccomplished☐ | |
| **Self-evaluation and Reasoning** | |
| The task was submitted before due time and was accepted by the company. | |

In the navigation process, we can specify a position for the Robot by either using the GUI, 2D Nav Goal, or by using a flexible code that we can change whenever we want. These following sub-tasks demonstrate the process.

Table 6: Task 5.1 Publish a Position Message Between Two ROS Nodes

| **Task 5.1: Publish a Position Message Between Two ROS Nodes** | |
|---|---|
| Task Description | Tools Applied |

| This task is a sub-task of task 5.3 | • Trurtlessim node<br>• CLI |
|---|---|

## Task Process

First, I wanted to learn more about ROS nodes and topics. I started with Turtlesim 2D robot, a package that provides simple simulator to learn ROS concepts quickly. I learned how to publish a position with the command line and then gradually learning how to do this flexibly using a publisher script in Python. For the publisher (turtle_teleop_key) and subscriber (turtlesim_node) to communicate, they must send and receive the same **type** of message.

After running Turtlesim node, I can see the messages published under /turtle1/cmd_vel topic.

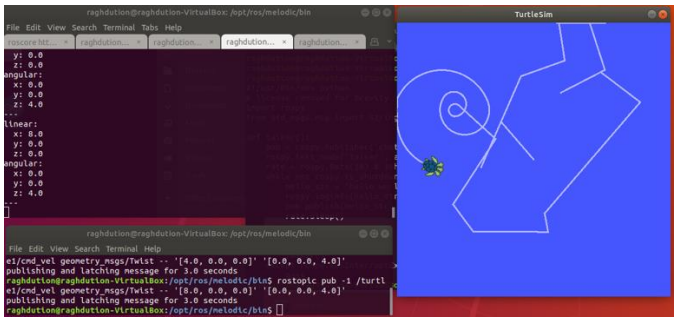



Figure 25 Task 5.1

rqt_graph makes a visual graph of all running nodes and their connections/topics.

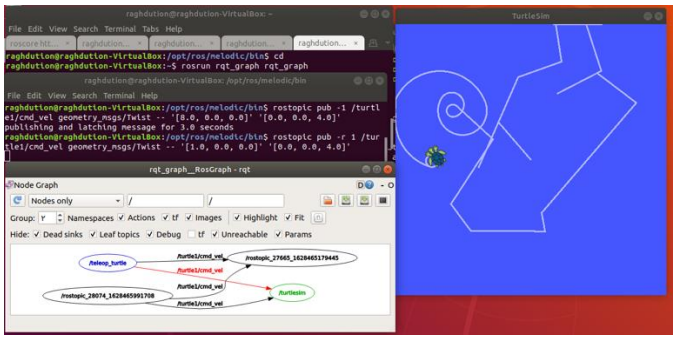



Figure 26 Task 5.1

I learned how to publish a fixed position, and how to make the Robot move with my control and with a predefined controller using these two commands:

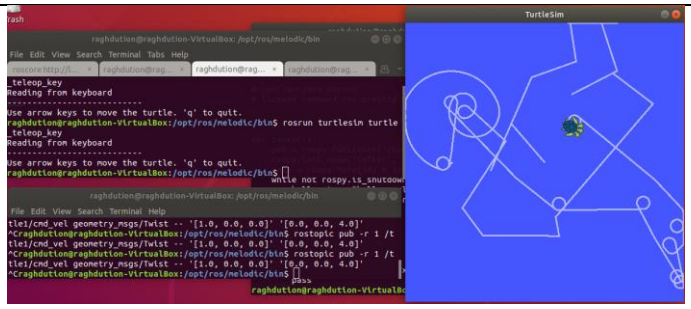



Figure 27 Task 5.1

| Challenges faced | Lessons learned |
|---|---|
| This task made task 5.3 easy, no challenges faced. | • Topic and nodes<br>• Difference between Publisher rand the subscriber and how they work<br>• Turtlesim node<br>• Publishing a position using CLI |
| **Supporting academic courses** | **Suggestions** |
| Operating systems. | I suggest for everyone trying to understand ROS concepts to start with this. |
| Task accomplished ☒ Task partially accomplished☐ Task unaccomplished☐ | |
| **Self-evaluation and Reasoning** | |
| My site supervisor was glad that I took this approach in learning and reasoning, by dividing the problem into sub tasks. | |

Table 7: Task 5.2 Publish a String Message Between Two ROS Nodes

| **Task 5.2: Publish a String Message Between Two ROS Nodes** | |
|---|---|
| Task Description | Tools Applied |
| This task is a sub-task of task 5.3 | • Python<br>• Virtual editor (Vi/Vim)<br>• CLI |
| Task Process | |
| Here I moved to make the Publisher and Subscriber scripts using Python. This can be done by creating a package with the rospy library inside it and then creating the **/scripts** folder for Python scripts. All this must be done inside the catkin workspace. After I created my scripts inside the Vi and changed their permissions to make them executable, Here is the result of their communication using **rqt_graph** | |

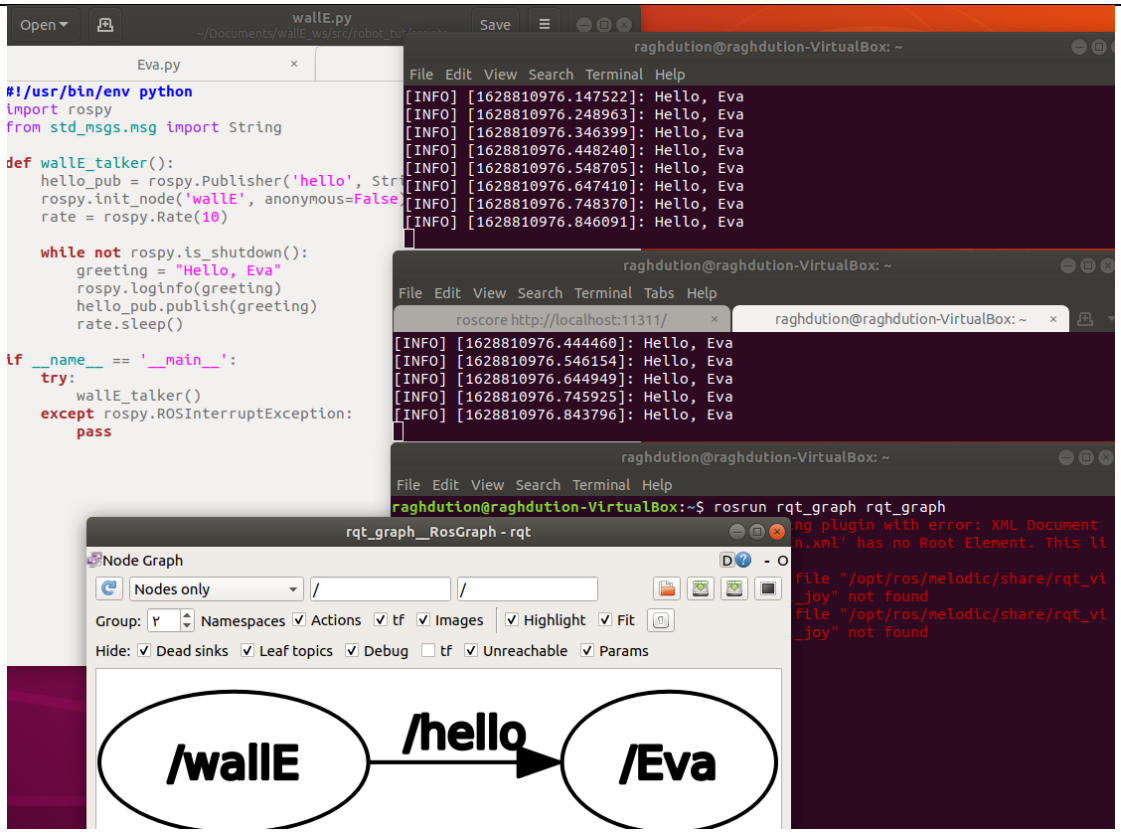
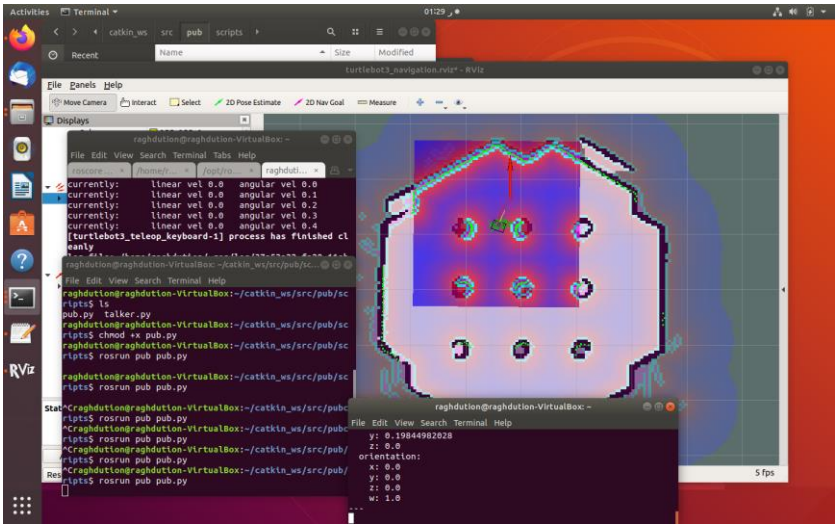



Figure 28 Publisher and subscriber codes

'Hello, Eva!' message inside the publisher script gets published in the terminal with the help of **rospy.loginfo(greeting)** line. In the subscriber scripts, the same line of code subscribes to **/hello** topic and publishes it to the terminal.

| Challenges faced | Lessons learned |
|---|---|
| All the resources for this task were outdated, I needed to learn Python for ROS, specifically, in order to modify the code to work on this task. | • Writing a Publisher and subscriber codes<br>• The importance of this task and how it demonstrates the whole ROS concept with all its nodes and topics. |
| **Supporting academic courses** | **Suggestions** |
| • Object Oriented Programming 1<br>• Fundamentals of programming<br>• Operating Systems | I suggest for everyone trying to understand ROS concepts to start with this. |
| Task accomplished ☒ | Task partially accomplished☐ Task unaccomplished☐ |
| **Self-evaluation and Reasoning** | |
| My site supervisor was glad that I took this approach in learning and reasoning, by dividing the problem into sub tasks. | |

Table 8: Task 5.3 Python script that publishes a position to /move_base_simple/goal topic

| **Task 5.3: Python script that publishes a position to /move_base_simple/goal topic** | |
|---|---|
| Task Description | Tools Applied |
| As I mentioned above, we need to have a flexible code to set the position that the Robot can go in order to flexibly modify it. | • Python<br>• CLI<br>• RViz |
| Task Process | |
| After a lot of research, I learned from the ROS website about **/move_base_simple/goal** topic, its parameters, type of messages, and what the code needs to have to publish the required information. Which are position, orientation and header, all of these are under the **goal** class.<br><br>After creating the publisher script inside Vi editor, and make it executable, I moved to the RViz simulator. I opened the map I saved before, which was in my home directory. To set the orientation we need to follow the same steps with GUI navigation which is moving around with the Robot so it can collect the needed information, recognize the environment, and avoid the obstacles in its way.<br><br>Lastly, to publish the Robot position to **/move_base_simple/goal** topic using a Python script, I ran **rostopic** to see the messages published under this topic like position, orientation... etc. The Robot can now move by any position we can define inside the Python script.<br><br>  <br>Figure 29 publish a position using python script | |
| Challenges faced | Lessons learned |
| The main problem I faced was the lack of resources in learning advanced topics in ROS. Also, because learning ROS means learning a lot of new things and building solid foundations in them. Like LINUX command line; Python, C++, ROS Navigation... etc. | • Publishing a flexible position using a python script<br>• Python for ROS<br>• How to learn about a specific topic's parameters<br>• The communication between two ROS nodes must be through the same type of message |

| | |
|---|---|
| ROS was new for me, I still didn't digest the whole system, and that's why I came up with this methodology (dividing the problem into sub tasks), and it worked well. | |
| Supporting academic courses | Suggestions |
| • OOP 1<br>• Fundamentals of programming<br>• Operating systems | I suggest for ROS developers to keep their packages updated, and state clearly in each package what version of a program or a code this package can work well with. |
| Task accomplished ☒ Task partially accomplished☐ Task unaccomplished☐ | |
| Self-evaluation and Reasoning | |
| The site supervisor was very happy about my performance in this task, the task was accepted by the company. | |

In the last weeks of training, we were challenged to launch multiple Robots in one map, this task, although its very advanced, it is essential in the Robotics world, e.g., in the Fighting Robots project.

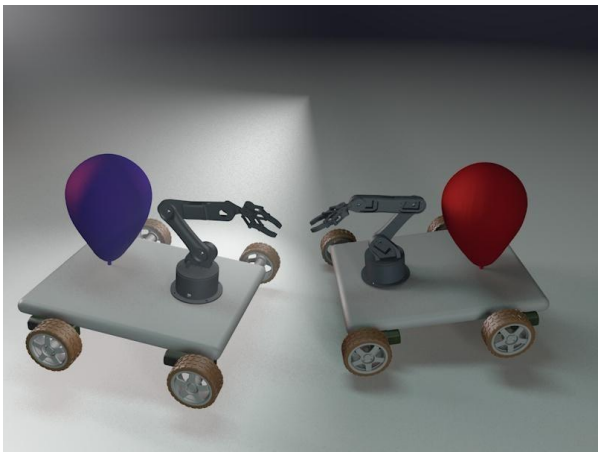


Figure 30 The Fighting Robots

Table 9 Task 6.1 Multiple Turtlebot3 using Teleop

| **Task 6.1: Multiple Turtlebot3 using Teleop** | |
|---|---|
| Task Description | Tools Applied |
| At first, we were asked to launch multiple TurtlBot3 in one map, the one we created before, using teleop. | • CLI<br>• Launch files<br>• TurtleBot3<br>• Gazebo |
| Task Process | |

After I learned advanced Linux commands for ROS. I created the package with all the decencies, then creating the launch files, which is a convenient way to start many nodes at the same time and make us write parameters during the runtime.

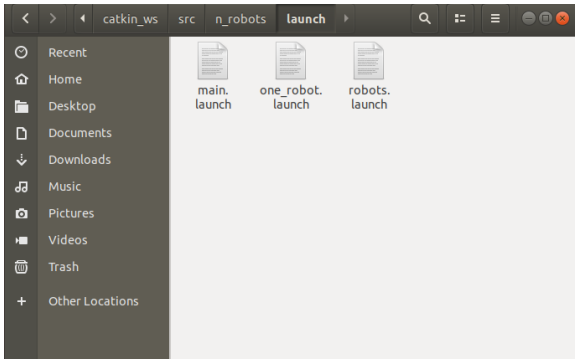



Figure 31 Task 6.1 File Structure

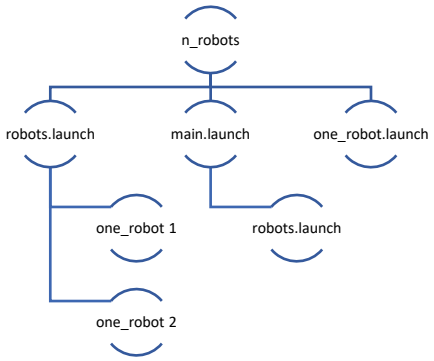



Figure 32 Task 6.1 in depth File Structure

**robots.launch** file will create instances of **one_robot.launch** file, depending on how many robots we want in the map. As Figure 33 shows, all running robots created with their running topics

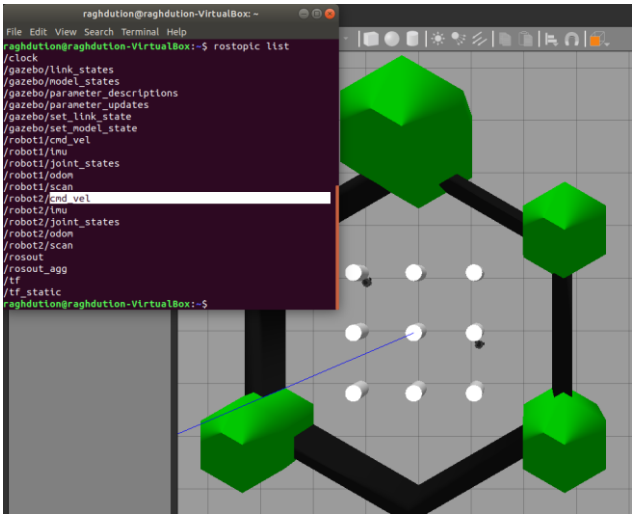



Figure 33 Task 6.1 rostopic list

And lastly, controling the robots we created using teleop in two separate windows, this will be clearer in the videos that will be illustrated in the presentation.

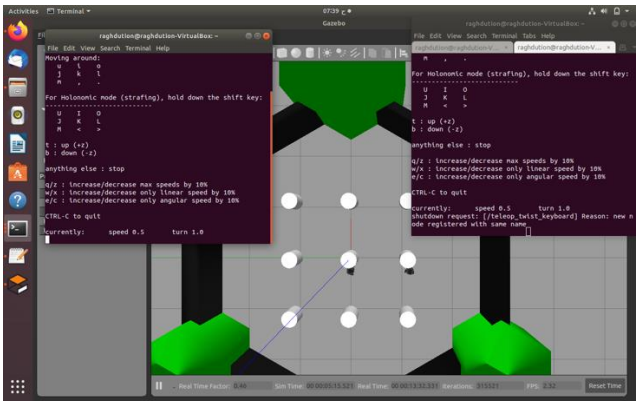


Figure 34 Task 6.1 multiple TB3 teleoperation

As you can notice, because this task is using teleoperation, no need to do this in RViz, because RViz is concerned with the navigation too.

| Challenges faced | Lessons learned |
|---|---|
| • lack of resources in ROS melodic; the existing resources are concerned with older versions of ROS and are not compatible with Melodic.<br>• The VM was halting every hour. Which made this task struggling to do all steps from beginning. | • ROS Launch files concepts<br>• ROS concepts are now clearer after this task<br>• I understood the complicated processes hidden underneath robots' projects and was extremely excited to advance in this world. |
| **Supporting academic courses** | **Suggestions** |
| Operating Systems. | I would really love to create a Robotics club in our college, this branch is advancing now more than ever, and its concepts are very important to understand, especially in the middle east where there is no focus on learning the deep concepts in robotics. |
| Task accomplished ☒ | Task partially accomplished☐ Task unaccomplished☐ |
| **Self-evaluation and Reasoning** | |
| The task was accepted by the company and done in due time with no delays. Finishing this task was very thrilling to complete the upcoming one. | |

Table 10 Task 6.2 Multiple Turtlebot3 using Navigation

| **Task 6.2: Multiple Turtlebot3 using Navigation** | |
|---|---|
| Task Description | Tools Applied |

| | |
|---|---|
| In this task, we were asked to launch multiple TurtlBot3 in one map, the one we created before, and control those robots' using navigation in RViz. | • RViz<br>• CLI<br>• Launch files<br>• Gazebo |

## Task Process

So first, I have searched for the launch files and what needs to contain to control multiple robots in one map. I navigated ROS file system and looked for trurtlebot3_navigation files. Which contained the needed file to add inside the package, I understood the concept and tried to the same, this was the file structure.

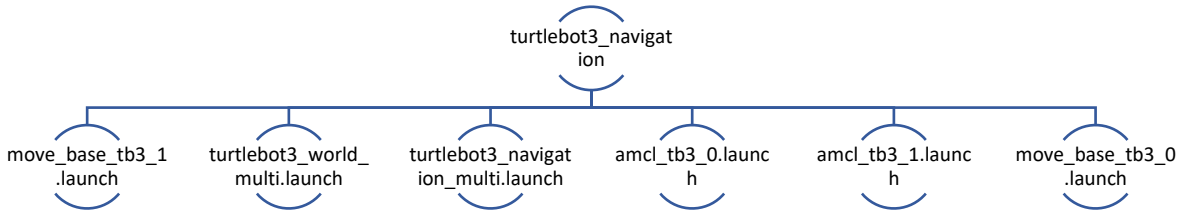



Figure 35 Taask 6.2 File structure

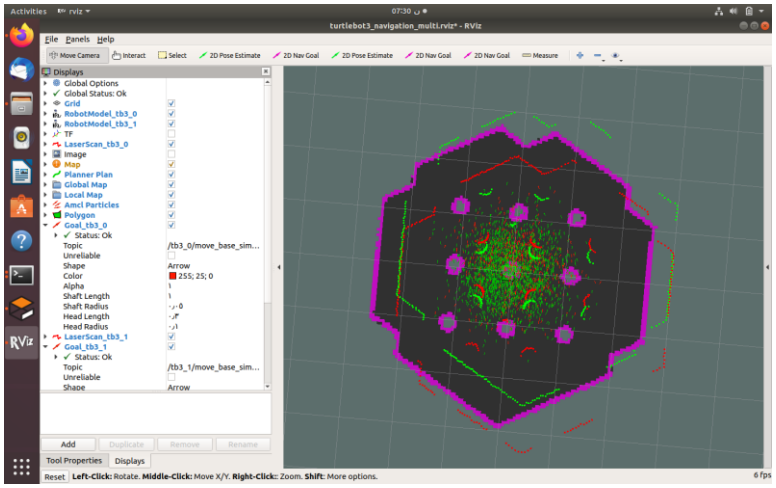


Figure 36 Task 6.2 RViz

After adjusting each robot's LDS sensor data to the map using 2D Pose Estimate (there are two **2D pose Estimate** and **2D Nav Goal** for each robot

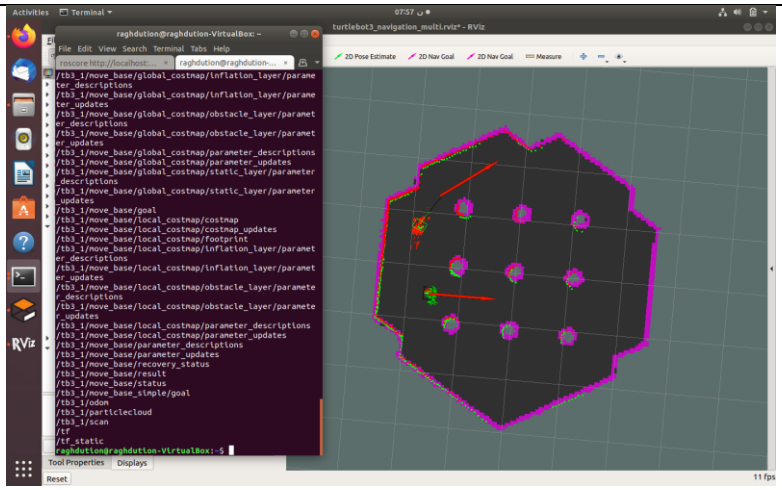


Figure 37 Task 6.2 rostopic list

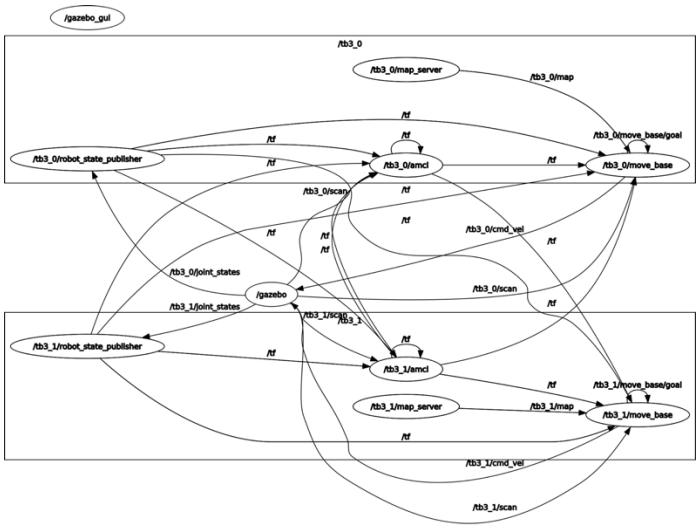



Figure 38 Task 6.2 rqt_graph

| Challenges faced | Lessons learned |
|---|---|
| The lack of resources in the Robotics field for this task specifically, due to the advanced nature of this task. I have actually seen a MS Research paper for this task. | • Multiple TB3 navigation<br>• Launch files<br>• ROS Topics |
| **Supporting academic courses** | **Suggestions** |
| Operating systems | No suggestions. |
| Task accomplished ☒ | Task partially accomplished☐ Task unaccomplished☐ |
| **Self-evaluation and Reasoning** | |

| |
|---|
| The site supervisor was very happy about my performance, for this task and all the upcoming tasks ahead, she said she was very proud to work with me. Even finishing this task which was an advanced, and challenging topic in ROS, it was an honor to do and learn in such a very limited time. |

## 2.2 AI Tasks:

This section's tasks will be implemented in Python and its libraries using ML and DL.

Table 11 Task 7 Real-time Face Detection Using OpenCV

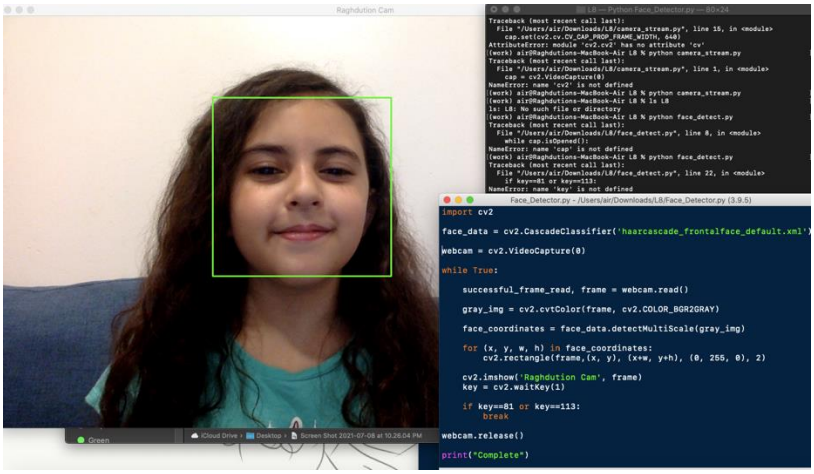
| **Task 7: Real-time Face Detection Using OpenCV** | |
|---|---|
| Task Description | Tools Applied |
| Using OpenCV to make a real-time face detection for Robotics vision system. | • Python IDE 3.8<br>• OpenCV |
| Task Process | |
| I tried to do it first with pictures to ensure that the OpenCV is working; then, I tested the camera with a code that opens the webcam without any detecting. After that, I launched the live face detection program.<br>  <br> | |
| Challenges faced | Lessons learned |
| Although this task was simple. I faced lots of problems, all of which I overcome, which made it painful and joyful at the same time.<br>• OpenCV Installation problems.<br>• Many Python problems in MacOS | I learned how OpenCV works, calling the classifiers, convert the frame or image to gray, highlighting the object coordinates with a rectangle, ending the program with a command or a keyboard letter to quit the camera. |

Figure 39 Task 7 Face recognition

| | |
|---|---|
| • MacOS wasn't granting access to camera. | |
| Supporting academic courses | Suggestions |
| Fundamentals of programming. | I suggest doing this task in recognition rather than detection only for more interaction with the robot. |
| Task accomplished ☒ Task partially accomplished☐ Task unaccomplished☐ | |
| Self-evaluation and Reasoning | |
| I finished this task, and it was accepted by the company. After this task, I received a certificate that will be illustrated in Appendix B, due to my excellence in the training. | |

Table 12 Task 8 Arabic Chatbot Rating Model

| **Task 8: Arabic Chatbot Rating Model** | |
|---|---|
| Task Description | Tools Applied |
| In this task we were supposed to build an Arabic chatbot model suitable for the Rating Robot that will be presented in the final event for the company. | • Spyder IDE<br>• Miniconda<br>• Keras<br>• NLTK<br>• Tkinter GUI<br>• Python 3.7 |
| Task Process | |
| I had absolutely no knowledge in Chatbots and in NLP in general, but that didn't stop me. My approach was to create a Retrieval-based Arabic Chatbot using NLTK, Keras and Tkinter. Here I have used Natural Language Toolkit: The ISRI Arabic Stemmer for the Arabic model, because it's already inside NLTK library and no need for installation.<br><br>Trying Arabic libraries with stop words and stemming. | |

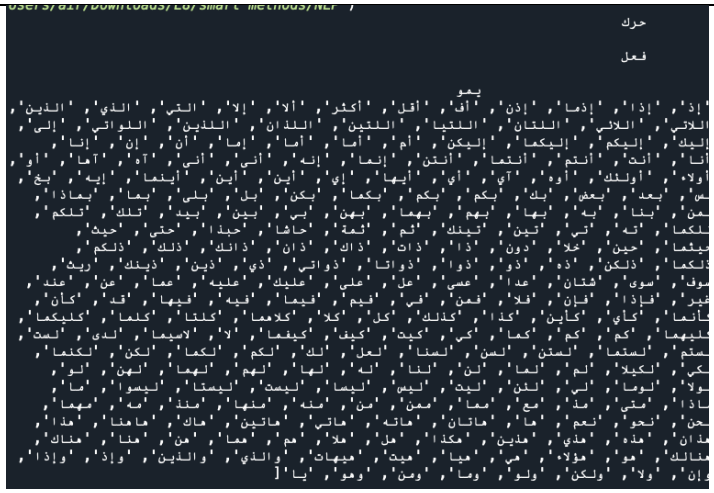


Figure 40 Task 8 stop words

You can clearly view the classes after training.

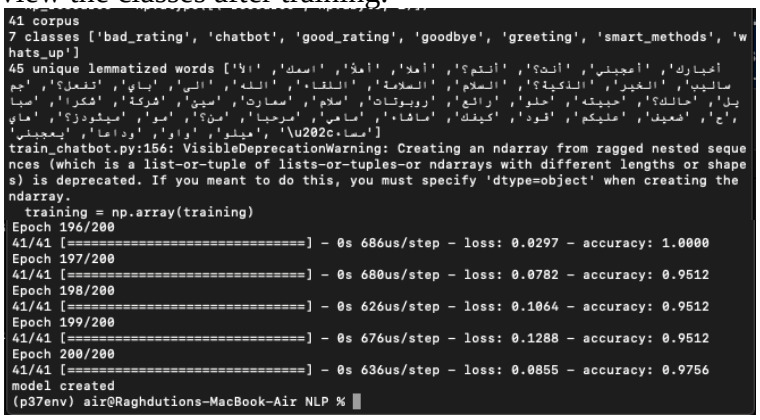



Figure 41 Training the Arabic model

Depending on how many hidden layers we have in our DL Nueral network, we will have more depth and more accuracy in our projects, because deep refers to the number of hidden layers in the NN. And each hidden layer in DL can be seen as an individual ML algorithm in its own.

| Challenges faced | Lessons learned |
|---|---|
| I have never worked on NLP projects, which is widely known as one of the trickiest subjects in machine learning, even when I took a machine learning course which had a deep learning part, the task was still very challenging to be done in one week only. Because it had to be in Arabic libraries which had low accuracy compared to when I trained the model in English language. | • The general approach of working on NLP projects<br>• The difference between machine learning and deep learning<br>• Discovered some interesting Arabic language libraries made by Arabic developers and was very proud to see their great work an effort in NLP |
| **Supporting academic courses** | **Suggestions** |
| • OOP 1<br>• Fundamentals of programming<br>• Algorithm analysis and design | I suggested to the company to do sentiment analysis to know the rating retrieved by the chatbot. Unfortunately, due to limited time, this was hard to do. |
| Task accomplished ☒ Task partially accomplished☐ Task unaccomplished☐ | |

| Self-evaluation and Reasoning |
| --- |
| I was so excited to work on this task, because I'm very interested in linguistics and as a programmer; in NLP. Most trainees did this task and the chat process inside the terminal without using a GUI, I was the only one that used Tkinter GUI.<br>I managed to use Arabic libraries to remove stop words and stemming the words to their roots, while achieving a 0.9 – 1.0 accuracy.<br>The site supervisor was very thrilled and excited about my performance through all these weeks, as much it was an honor for her, it was more an honor for me to work with her. |

## 2.3 Other learning opportunities:

I have attended two workshops during my training that was hosted by different engineers, which has helped me a expand my knowledge in the Robotics world lot.

### 2.3.1 Workshops

The first workshop was hosted by Eng. Sultan Almutairi, a NEOM employee, where he explained his journey working on CNC 3D printers -Computer Numerical Control-. He talked about the problems he faced, how he was the first to use it in the middle east, no guidance, and no support was given to him, only negative opinions from society. One of the problems he faced because of his lack of knowledge was when he brought the small CNC machines from China in his early career; they couldn't let him pass the machines with him through the airport, so he lost all his money. On the bright side, he talked about how he managed to benefit from these machines financially and on a career and social level; he was the first in this field in the middle east, and in many companies, engineers have benefited from his opinion, experience, and journey in using these machines. At last, he talked about his mistakes, his advice to the new generation in using these machines, and in life and career in general.

The second workshop was about GitHub to submit our codes to be reviewed and accepted by the company, I learned how to clone the Arm package in this workshop.

# Chapter – 03: Conclusion

In conclusion, these ten weeks have shown me the importance of how doing what you love and passionate about can make you very patient, enjoy the obstacles, problems, back pain, and self-learning to the max! the robotics field is as expected; it is very limited in resources, making it exciting and challenging. I enjoyed working with the company to the max, because this was my passion since middle school. I was thrilled to join the Smart Methods team as a COOP student, which was only the starting point towards achieving my dream. My Site Supervisors were outstanding in the Robotics field, from the first robot life cycle to the end. It was truly an honor to work with them. I'm very excited that I have chosen the training that suits my needs and dreams, and I hope to achieve something that would leave a mark in life, and help people through technology make their lives easier.

# References

[1]- Smart Methods:

https://www.s-m.com.sa/en.html

[2]- The Construct:

https://www.theconstructsim.com/become-robotics-developer/

[3]- Developer Notes:

https://developpaper.com/study-notes-of-autolabor-2-5-3-working-space-and-compiling-system-of-ros/

[4] - B. G. W. D. S. Morgan Quigley, Programming Robots with ROS: A Practical Introduction to the Robot Operating System, 2015.

[5]- TurtleBot3:

https://www.turtlebot.com/

# Appendix – A

## 1. Linux Shell Commands

Task 1.1 Linux shell with instructions for each command

```
1. Setup the computer to accept software from packages.ros.org.

$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" >
/etc/apt/sources.list.d/ros-latest.list'

2. Setup the keys

$ sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key
C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654

3. make sure the Debian package index is up-to-date

$ sudo apt-get update

4. Desktop-Full Install:

$ sudo apt install ros-melodic-desktop-full

5. To find available packages

$ apt search ros-melodic

6. Environment setup

$ echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
source ~/.bashrc

7. Dependencies for building packages

$ sudo apt install python-rosdep python-rosinstall python-rosinstall-generator python-wstool
build-essential

8. Initialize rosdep

$ sudo apt install python-rosdep

$ sudo rosdep init

$ rosdep update

9. Installing the prebuilt Package catkin

$ sudo apt-get install ros-melodic-catkin

10. Create a ROS catkin workspace and source the environemnt

$ source /opt/ros/melodic/setup.bash

$ mkdir -p ~/catkin_ws/src

$ cd ~/catkin_ws/

$ catkin_make

11.  Installing the Arduino robot arm package

$ cd ~/catkin_ws/src
$ sudo apt install git

$ git clone https://github.com/smart-methods/arduino_robot_arm.git
```

```
12.  Install all The package dependencies

$ cd ~/catkin_ws

$ rosdep install --from-paths src --ignore-src -r -y

$ sudo apt-get install ros-melodic-moveit

$ sudo apt-get install ros-melodic-joint-state-publisher ros-melodic-joint-state-publisher-gui

$ sudo apt-get install ros-melodic-gazebo-ros-control joint-state-publisher

$ sudo apt-get install ros-melodic-ros-controllers ros-melodic-ros-control

12.  Compline the package

$ catkin_make


13. Load the moveit assistant
$ roslaunch moveit_setup_assistant setup_assistant.launch

14. Launch the arm package with moveit in Rviz and Gazebo, respectiely
$ roslaunch moveit_pkg demo.launch
$ roslaunch moveit_pkg demo_gazebo.launch


15.  Download Arduino with linux distribution from their official site

16. Install rosserial for Arduino

$ sudo apt-get install ros-melodic-rosserial-arduino
$ sudo apt-get install ros-melodic-rosserial

17. Install ros_lib for Arduino

$ cd <sketchbook>/libraries
$ rm -rf ros_lib
$ rosrun rosserial_arduino make_libraries.py .
```

Linux CLI 1 Task 1.1

## Task 1.2

```
18.  Launch the arm package in RViz

$ sudo nano ~/.bashrc
at the end of the (bashrc) file add the follwing line
(source /home/raghdution/catkin_ws/devel/setup.bash)
then
ctrl + o
source ~/.bashrc
$ roslaunch robot_arm_pkg check_motors.launch

19.  Launch the arm package in Gazebo

$ roslaunch robot_arm_pkg check_motors_gazebo.launch

20. Controlling motor in simulation using joint_states_publisher
$ rosrun robot_arm_pkg joint_states_to_gazebo.py

21. Change the premission
$ cd catkin/src/arduino_robot_arm/robot_arm_pkg/scripts
$ sudo chmod +x joint_states_to_gazebo.py
```

Linux CLI 2 Task 1.2

## Task 2

```
- Install TurtleBot3 Packages

$ sudo apt-get install ros-melodic-dynamixel-sdk
$ sudo apt-get install ros-melodic-turtlebot3-msgs
$ sudo apt-get install ros-melodic-turtlebot3

- Set TurtleBot3 Model Name
$ echo "export TURTLEBOT3_MODEL=burger" >> ~/.bashrc

- Install Simulation Package
$ cd ~/catkin_ws/src/
$ git clone -b melodic-devel https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
$ cd ~/catkin_ws && catkin_make
$ cd

- Launch simulation world
$ export TURTLEBOT3_MODEL=burger (1st tab)
$ roslaunch turtlebot3_gazebo turtlebot3_world.launch

- Run SLAM node
$ export TURTLEBOT3_MODEL=burge (2nd tab)
$ roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=gmapping

- Run teleoperate node
$ export TURTLEBOT3_MODEL=burger (3rd tab)
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

- save the map
$ rosrun map_server map_saver -f ~/map
```

Linux CLI 3 Task 2

## Task 3

```
$ cd catkin_ws/src

- cloning the author package to the source folder
$ git clone https://github.com/devanshdhrafani/diff_drive_bot.git
$ cd ..
 $ catkin_make

- installing the dependencies
$ sudo apt-get install ros-melodic-dwa-local-planner
$ sudo apt-get install ros-melodic-joy

$ roslaunch diff_drive_bot gazebo.launch

- open RViz
$ roslaunch diff_drive_bot gmapping.launch

- keyboard teleoperation
$ rosrun diff_drive_bot keyboard_teleop.py

- save the map
$ rosrun map_server map_saver -f ~/my_map
```

Linux CLI 4 Task 3

## Task 4

```
- Launch Simulation World
$ export TURTLEBOT3_MODEL=burger (1st tab)
$ roslaunch turtlebot3_gazebo turtlebot3_world.launch

- Run Navigation Node
$ export TURTLEBOT3_MODEL=burger (2nd tab)
$ roslaunch turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map.yaml


- Run teleoperate node
```

```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

Linux CLI 5 Task 4

## Task 5.1

```
$ roscore (1st tab)

$ rosrun turtlesim turtlesim_node (2nd tab)

- Run teleoperation
$ rosrun turtlesim turtle_teleop_key

- print the messages published under any topic
$ rostopic echo /turtle1/cmd_vel

- see nodes and their associated topic visually
$ rosrun rqt_graph rqt_graph

- publish once and exit
$ rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist -- '[1.0, 0.0, 0.0]' '[0.0, 0.0, 4.0]'

- publish the message repeatedly at the rate of one second
$ rostopic pub  /turtle1/cmd_vel geometry_msgs/Twist -r 1 -- '[1.0, 0.0, 0.0]' '[0.0, 0.0, 4.0]'

- make the robot move with your control AND with a predefined controller
$ rostopic pub  /turtle1/cmd_vel geometry_msgs/Twist -r 1 -- '[1.0, 0.0, 0.0]' '[0.0, 0.0, 4.0]'
$ rosrun turtlesim turtle_teleop_key
```

Linux CLI 6 Task 5.1

## Task 5.2

```
$ cd ~/catkin_ws/src

- create the package with its dependencies

$ catkin_create_pkg pub std_msgs rospy roscpp

$ cd ..

$ catkin_make

$ . ~/catkin_ws/devel/setup.bash

$ roscd pub

$ mkdir scripts

$ cd scripts

$ vi wallE.py
```

```
1.  wallE.py
2.  #!/usr/bin/env python
3.    import rospy
4.    from std_msgs.msg import String
5.
6.
7.    def wallE_talker():
8.        hello_pub = rospy.Publisher('hello', String, queue_size = 10)
9.        rospy.init_node('wallE', anonymous=False)
10.       rate = rospy.Rate(10)
11.
12.
13.       while not rospy.is_shutdown():
14.           greeting = "Hello, Eva"
15.           rospy.loginfo(greeting)
16.           hello_pub.publish(greeting)
```

```
            rate.sleep()


    if __name__ == '__main__':
        try:
            wallE_talker()
        except rospy.ROSInterruptException:
            pass
```

```
$ roscore (new tab)

- make the *publsiher* file executable

$ chmod +x wallE.py

- message gets published by the help of # rospy.loginfo(greeting)
$ python wallE.py

$ rosrun rqt_graph rqt_graph

$ rostopic echo /hello

- create the subscriber code

$ vi Eva.py
```

```
Eva.py
#!/usr/bin/env python
  import rospy
  from std_msgs.msg import String


  def Callback_str(data):
      rospy.loginfo(data.data)



  def Eva_listener():
      rospy.init_node('Eva', anonymous=False)
      rospy.Subscriber('hello', String, Callback_str)
      rospy.spin()


  if __name__ == '__main__':
      Eva_listener()
```

```
- make the subscriber file executable
$ chmod +x Eva.py

$ python Eva.py

$ python wallE.py
```

Linux CLI 7 Task 5.2

## Task 5.3

```
$ cd ~/catkin_ws/src

$ catkin_create_pkg pub std_msgs rospy roscpp

$ cd ..

$ catkin_make
$ . ~/catkin_ws/devel/setup.bash
```

```
- Writing the Publisher Node

$ roscd pub

$ mkdir scripts

$ cd scripts

$ vi pub.py
```

```
1.  #!/usr/bin/env python
2.    import rospy
3.    from geometry_msgs.msg import PoseStamped
4.
5.
6.    def pub():
7.        rospy.init_node("pub")
8.        goal_publisher = rospy.Publisher("/move_base_simple/goal", PoseStamped,
    queue_size=10)
9.
10.
11.       goal = PoseStamped()
12.
13.
14.       goal.header.seq = 5
15.       goal.header.frame_id = "map"
16.
17.       goal.pose.position.x = 1.65549182892
18.       goal.pose.position.y = 0.19844982028
19.       goal.pose.position.z = 0.0
20.
21.       goal.pose.orientation.x = 0.0
22.       goal.pose.orientation.y = 0.0
23.       goal.pose.orientation.z = 0.0
24.       goal.pose.orientation.w = 1.0
25.
26.
27.       rospy.sleep(1)
28.       goal_publisher.publish(goal)
29.       rospy.spin()
30.
31.
32.   if __name__ == '__main__':
33.        try:
34.            pub()
35.        except rospy.ROSInterruptException:
36.            pass
37.
```

```
- open the RViz simulator

$ roscore (1st tab)

$ export TURTLEBOT3_MODEL=burger (2nd tab)

$ roslaunch turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map.yaml

$ export TURTLEBOT3_MODEL=burger (3rd tab)

$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

- Publishing the robot position to the topic

$ rostopic echo /move_base_simple/goal (4th tab)
```

```
$ rosrun pub pub.py (5th tab)
```

Linux CLI 8 Task 5.3

## Task 6.1

```
$ cd catkin_ws/src

$ catkin_create_pkg n_robots rospy gazebo_ros

$ cd ..

$ catkin_make

$ . ~/catkin_ws/devel/setup.bash

$ cd src/n_robots

$ mkdir launch

$ cd launch

$ vi one_robot.launch
```

```
1.  <launch>
2.        <arg name="robot_name"/>
3.        <arg name="init_pose"/>
4.
5.
6.        <node name="spawn_minibot_model" pkg="gazebo_ros" type="spawn_model"
7.         args="$(arg init_pose) -urdf -param /robot_description -model $(arg robot_name)"
8.         respawn="false" output="screen" />
9.
10.
11.       <node pkg="robot_state_publisher" type="state_publisher"
12.             name="robot_state_publisher" output="screen"/>
13.
14. </launch>
```

```
$ vi robots.launch
```

```
1.  <launch>
2.      <!-- No namespace here as we will share this description.
3.           Access with slash at the beginning -->
4.      <param name="robot_description"
5.        command="$(find xacro)/xacro.py $(find
    turtlebot3_description)/urdf/turtlebot3_burger.urdf.xacro" />
6.
7.  <!-- NOTE: the file can be found in: /opt/ros/melodic/share/turtlebot3_description/urdf
8.  ROSDS: $(find turtlebot_description)/robots/kobuki_hexagons_asus_xtion_pro.urdf.xacro  -
    ->
9.
10.
11.     <!-- BEGIN ROBOT 1-->
12.     <group ns="robot1">
13.       <param name="tf_prefix" value="robot1_tf" />
14.       <include file="$(find n_robots)/launch/one_robot.launch" >
15.         <arg name="init_pose" value="-x 1 -y 1 -z 0" />
16.         <arg name="robot_name"  value="Robot1" />
17.       </include>
18.     </group>
19.
20.
21.     <!-- BEGIN ROBOT 2-->
22.     <group ns="robot2">
23.       <param name="tf_prefix" value="robot2_tf" />
24.       <include file="$(find n_robots)/launch/one_robot.launch" >
```

```
        <arg name="init_pose" value="-x -1 -y 1 -z 0" />
        <arg name="robot_name"  value="Robot2" />
      </include>
    </group>


  </launch>
```

```
$ vi main.launch
```

```
<launch>
    <param name="/use_sim_time" value="true" />


    <!-- start world -->
    <node name="gazebo" pkg="gazebo_ros" type="gazebo"
     args="$(find turtlebot3_gazebo)/worlds/turtlebot3_world.world" respawn="false"
    output="screen" />

    <!-- the file can be found in:
    catkin_ws/src/turtlebot3_simulations/turtlebot3_gazebo/worlds/turtlebot3_world.world-->


    <!-- start gui -->
    <node name="gazebo_gui" pkg="gazebo_ros" type="gui" respawn="false"
    output="screen"/>


    <!-- include our robots -->
    <include file="$(find n_robots)/launch/robots.launch"/>
  </launch>
```

```
- open Gazebo
$ roslaunch n_robots main.launch

- to see running topics and robots
$ rostopic list

- control the reboots using teleop
$ rosrun teleop_twist_keyboard teleop_twist_keyboard.py /cmd_vel:=/robot1/cmd_vel (1st window)
$ rosrun teleop_twist_keyboard teleop_twist_keyboard.py /cmd_vel:=/robot2/cmd_vel (2nd window)
```

Linux CLI 9 Task 6.1

## Task 6.2

```
$ cd /catkin_ws/src

$ git clone https://github.com/raghdutionn/N-Robots-Navigation

$ cd ..

$ catkin_make

$ cd

$ source ~/catkin_ws/devel/setup.bash

$ sudo apt-get install ros-melodic-turtlebot3

$ sudo apt-get install ros-melodic-turtlebot3-gazebo

$ sudo apt-get install ros-melodic-turtlebot3-navigation

$ roscore (1st tab)
```

```
$ export TURTLEBOT3_MODEL=burger (2nd tab)

$ roslaunch turtlebot3_gazebo turtlebot3_world_multi.launch
$ export TURTLEBOT3_MODEL=burger (3rd tab)

$ roslaunch turtlebot3_navigation turtlebot3_navigation_multi.launch

 - to see the topics
$ rostopic list

$ rosrun rqt_graph rqt_graph
```

Linux CLI 10 Task 6.2

## 2. Launch Files

Task 6.2 Launch Files

```
1.  <launch>
2.
3.    <!-- Arguments -->
4.
5.    <arg name="scan_topic"     default="scan"/>
6.
7.    <arg name="initial_pose_x" default="1.0"/>
8.
9.    <arg name="initial_pose_y" default="1.0"/>
10.
11.   <arg name="initial_pose_a" default="0.0"/>
12.
13.   <!-- AMCL -->
14.
15.   <node pkg="amcl" type="amcl" name="amcl">
16.
17.     <param name="min_particles"             value="500"/>
18.
19.     <param name="max_particles"             value="3000"/>
20.
21.     <param name="kld_err"                   value="0.02"/>
22.
23.     <param name="update_min_d"              value="0.20"/>
24.
25.     <param name="update_min_a"              value="0.20"/>
26.
27.     <param name="resample_interval"         value="1"/>
28.
29.     <param name="transform_tolerance"       value="0.5"/>
30.
31.     <param name="recovery_alpha_slow"       value="0.00"/>
32.
33.     <param name="recovery_alpha_fast"       value="0.00"/>
34.
35.     <param name="initial_pose_x"            value="$(arg initial_pose_x)"/>
36.
37.     <param name="initial_pose_y"            value="$(arg initial_pose_y)"/>
38.
39.     <param name="initial_pose_a"            value="$(arg initial_pose_a)"/>
40.
41.     <param name="gui_publish_rate"          value="50.0"/>
42.
43.     <remap from="scan"                      to="$(arg scan_topic)"/>
44.
45.     <param name="laser_max_range"           value="3.5"/>
46.
47.     <param name="laser_max_beams"           value="180"/>
48.
```

```
49.     <param name="laser_z_hit"               value="0.5"/>
50.
51.     <param name="laser_z_short"             value="0.05"/>
52.
53.     <param name="laser_z_max"               value="0.05"/>
54.
55.     <param name="laser_z_rand"              value="0.5"/>
56.
57.     <param name="laser_sigma_hit"           value="0.2"/>
58.
59.     <param name="laser_lambda_short"        value="0.1"/>
60.
61.     <param name="laser_likelihood_max_dist" value="2.0"/>
62.
63.     <param name="laser_model_type"          value="likelihood_field"/>
64.
65.     <param name="odom_model_type"           value="diff"/>
66.
67.     <param name="odom_alpha1"               value="0.1"/>
68.
69.     <param name="odom_alpha2"               value="0.1"/>
70.
71.     <param name="odom_alpha3"               value="0.1"/>
72.
73.     <param name="odom_alpha4"               value="0.1"/>
74.
75.     <param name="odom_frame_id"             value="/tb3_0/odom"/>
76.
77.     <param name="base_frame_id"             value="/tb3_0/base_footprint"/>
78.
79.     <param name="global_frame_id"           value="/map"/>
80.
81.     <remap from="static_map"                to="/static_map"/>
82.
83.     <param name="use_map_topic"             value="true"/>
84.
85.   </node>
86.
87. </launch>
88.
89.
90.
```

Launch File 1 Task 6.2 turtlebot3_world_multi.launch

```
1.  <launch>
2.
3.    <!-- Arguments -->
4.
5.    <arg name="model" default="$(env TURTLEBOT3_MODEL)" doc="model type [burger, waffle,
    waffle_pi]"/>
6.
7.    <arg name="map_file" default="$(find turtlebot3_navigation)/maps/map.yaml"/>
8.
9.    <arg name="open_rviz" default="true"/>
10.
11.   <arg name="move_forward_only" default="true"/>
12.
13.   <arg name="first_tb3"  default="tb3_0"/>
14.   <arg name="second_tb3" default="tb3_1"/>
15.
16.   <arg name="first_tb3_x_pos" default=" 0.0"/>
17.   <arg name="first_tb3_y_pos" default=" 0.0"/>
18.   <arg name="first_tb3_z_pos" default=" 0.0"/>
19.   <arg name="first_tb3_yaw"   default=" 0.0"/>
20.
```

```
  <arg name="second_tb3_x_pos" default=" 0.0"/>
  <arg name="second_tb3_y_pos" default=" 0.0"/>
  <arg name="second_tb3_z_pos" default=" 0.0"/>
  <arg name="second_tb3_yaw"   default=" 0.0"/>

<param name="/use_sim_time" value="true"/>

  <!-- First tb3 group launch -->

  <group ns = "$(arg first_tb3)">

   <param name="tf_prefix" value="$(arg first_tb3)"/>

  <!-- Map server -->

  <node pkg="map_server" name="map_server" type="map_server" args="$(arg map_file)">

   <param name="frame_id" value="/map" />

  </node>

   <!-- AMCL -->

   <include file="$(find turtlebot3_navigation)/launch/amcl_tb3_0.launch"/>

   <!-- Move base -->

   <include file="$(find turtlebot3_navigation)/launch/move_base_tb3_0.launch">

    <arg name="model" value="$(arg model)" />

    <arg name="move_forward_only" value="$(arg move_forward_only)"/>

   </include>

  </group>

  <!-- Second tb3 group launch -->

  <group ns = "$(arg second_tb3)">

   <param name="tf_prefix" value="$(arg second_tb3)"/>

   <!-- Map server -->

   <node pkg="map_server" name="map_server" type="map_server" args="$(arg map_file)">

    <param name="frame_id" value="/map" />

   </node>

   <!-- AMCL -->

   <include file="$(find turtlebot3_navigation)/launch/amcl_tb3_1.launch"/>

   <!-- Move base -->

   <include file="$(find turtlebot3_navigation)/launch/move_base_tb3_1.launch">

    <arg name="model" value="$(arg model)" />

    <arg name="move_forward_only" value="$(arg move_forward_only)"/>

   </include>

  </group>

<!-- rviz -->

  <group if="$(arg open_rviz)">
```

```
91.
92.     <node pkg="rviz" type="rviz" name="rviz" required="true"
93.
94.           args="-d $(find turtlebot3_navigation)/rviz/s2.rviz"/>
95.
96.   </group>
97.  </launch>
```

Launch File 2 Task 6.2 turtlebot3_navigation_multi.launch

```
1.  <launch>
2.
3.    <!-- Arguments -->
4.
5.    <arg name="scan_topic"     default="scan"/>
6.
7.    <arg name="initial_pose_x" default="1.0"/>
8.
9.    <arg name="initial_pose_y" default="1.0"/>
10.
11.   <arg name="initial_pose_a" default="0.0"/>
12.

13.   <!-- AMCL -->
14.
15.   <node pkg="amcl" type="amcl" name="amcl">
16.

17.     <param name="min_particles"             value="500"/>
18.
19.     <param name="max_particles"             value="3000"/>
20.
21.     <param name="kld_err"                   value="0.02"/>
22.
23.     <param name="update_min_d"              value="0.20"/>
24.
25.     <param name="update_min_a"              value="0.20"/>
26.
27.     <param name="resample_interval"         value="1"/>
28.
29.     <param name="transform_tolerance"       value="0.5"/>
30.
31.     <param name="recovery_alpha_slow"       value="0.00"/>
32.
33.     <param name="recovery_alpha_fast"       value="0.00"/>
34.
35.     <param name="initial_pose_x"            value="$(arg initial_pose_x)"/>
36.
37.     <param name="initial_pose_y"            value="$(arg initial_pose_y)"/>
38.
39.     <param name="initial_pose_a"            value="$(arg initial_pose_a)"/>
40.
41.     <param name="gui_publish_rate"          value="50.0"/>
42.

43.     <remap from="scan"                      to="$(arg scan_topic)"/>
44.
45.     <param name="laser_max_range"           value="3.5"/>
46.
47.     <param name="laser_max_beams"           value="180"/>
48.
49.     <param name="laser_z_hit"               value="0.5"/>
50.
51.     <param name="laser_z_short"             value="0.05"/>
52.
53.     <param name="laser_z_max"               value="0.05"/>
54.
55.     <param name="laser_z_rand"              value="0.5"/>
```

```
56.
57.     <param name="laser_sigma_hit"           value="0.2"/>
58.
59.     <param name="laser_lambda_short"        value="0.1"/>
60.
61.     <param name="laser_likelihood_max_dist" value="2.0"/>
62.
63.     <param name="laser_model_type"          value="likelihood_field"/>
64.

65.     <param name="odom_model_type"           value="diff"/>
66.
67.     <param name="odom_alpha1"               value="0.1"/>
68.
69.     <param name="odom_alpha2"               value="0.1"/>
70.
71.     <param name="odom_alpha3"               value="0.1"/>
72.
73.     <param name="odom_alpha4"               value="0.1"/>
74.
75.     <param name="odom_frame_id"             value="/tb3_0/odom"/>
76.
77.     <param name="base_frame_id"             value="/tb3_0/base_footprint"/>
78.
79.     <param name="global_frame_id"           value="/map"/>
80.

81.     <remap from="static_map"                to="/static_map"/>
82.
83.     <param name="use_map_topic"             value="true"/>
84.

85.   </node>
86.
87. </launch>
88.
89.
```

Launch File 3 Task 6.2 amcl_tb3_0.launch

```
1.  <launch>
2.
3.    <!-- Arguments -->
4.
5.    <arg name="scan_topic"     default="scan"/>
6.
7.    <arg name="initial_pose_x" default="-1.0"/>
8.
9.    <arg name="initial_pose_y" default="1.0"/>
10.
11.   <arg name="initial_pose_a" default="0.0"/>
12.

13.   <!-- AMCL -->
14.
15.   <node pkg="amcl" type="amcl" name="amcl">
16.

17.     <param name="min_particles"             value="500"/>
18.
19.     <param name="max_particles"             value="3000"/>
20.
21.     <param name="kld_err"                   value="0.02"/>
22.
23.     <param name="update_min_d"              value="0.20"/>
24.
25.     <param name="update_min_a"              value="0.20"/>
26.
```

```
27.     <param name="resample_interval"         value="1"/>
28.
29.     <param name="transform_tolerance"       value="0.5"/>
30.
31.     <param name="recovery_alpha_slow"       value="0.00"/>
32.
33.     <param name="recovery_alpha_fast"       value="0.00"/>
34.
35.     <param name="initial_pose_x"            value="$(arg initial_pose_x)"/>
36.
37.     <param name="initial_pose_y"            value="$(arg initial_pose_y)"/>
38.
39.     <param name="initial_pose_a"            value="$(arg initial_pose_a)"/>
40.
41.     <param name="gui_publish_rate"          value="50.0"/>
42.

43.     <remap from="scan"                      to="$(arg scan_topic)"/>
44.
45.     <param name="laser_max_range"           value="3.5"/>
46.
47.     <param name="laser_max_beams"           value="180"/>
48.
49.     <param name="laser_z_hit"               value="0.5"/>
50.
51.     <param name="laser_z_short"             value="0.05"/>
52.
53.     <param name="laser_z_max"               value="0.05"/>
54.
55.     <param name="laser_z_rand"              value="0.5"/>
56.
57.     <param name="laser_sigma_hit"           value="0.2"/>
58.
59.     <param name="laser_lambda_short"        value="0.1"/>
60.
61.     <param name="laser_likelihood_max_dist" value="2.0"/>
62.
63.     <param name="laser_model_type"          value="likelihood_field"/>
64.

65.     <param name="odom_model_type"           value="diff"/>
66.
67.     <param name="odom_alpha1"               value="0.1"/>
68.
69.     <param name="odom_alpha2"               value="0.1"/>
70.
71.     <param name="odom_alpha3"               value="0.1"/>
72.
73.     <param name="odom_alpha4"               value="0.1"/>
74.
75.     <param name="odom_frame_id"             value="/tb3_1/odom"/>
76.
77.     <param name="base_frame_id"             value="/tb3_1/base_footprint"/>
78.
79.     <param name="global_frame_id"           value="/map"/>
80.

81.     <remap from="static_map"                to="/static_map"/>
82.
83.     <param name="use_map_topic"             value="true"/>
84.

85.   </node>
86.
87. </launch>
88.
89.
90.
```

Launch File 4 Task 6.2 amcl_tb3_1.launch

```
1.  <launch>
2.
3.    <!-- Arguments -->
4.
5.    <arg name="model" default="$(env TURTLEBOT3_MODEL)" doc="model type [burger, waffle,
    waffle_pi]"/>
6.
7.    <arg name="cmd_vel_topic" default="cmd_vel" />
8.
9.    <arg name="odom_topic" default="odom" />
10.
11.   <arg name="move_forward_only" default="false"/>
12.

13.   <!-- move_base -->
14.
15.   <node pkg="move_base" type="move_base" respawn="false" name="move_base"
    output="screen">
16.
17.     <param name="base_local_planner" value="dwa_local_planner/DWAPlannerROS" />
18.
19.     <rosparam file="$(find turtlebot3_navigation)/param/costmap_common_params_$(arg
    model).yaml" command="load" ns="global_costmap" />
20.
21.     <rosparam file="$(find turtlebot3_navigation)/param/costmap_common_params_$(arg
    model).yaml" command="load" ns="local_costmap" />
22.
23.     <rosparam file="$(find turtlebot3_navigation)/param/local_costmap_params.yaml"
    command="load" />
24.
25.     <rosparam file="$(find turtlebot3_navigation)/param/global_costmap_params.yaml"
    command="load" />
26.
27.     <rosparam file="$(find turtlebot3_navigation)/param/move_base_params.yaml"
    command="load" />
28.
29.     <rosparam file="$(find turtlebot3_navigation)/param/dwa_local_planner_params_$(arg
    model).yaml" command="load" />
30.
31.     <param name="global_costmap/robot_base_frame" value="/tb3_0/base_footprint"/>
32.
33.     <param name="local_costmap/robot_base_frame" value="/tb3_0/base_footprint"/>
34.
35.     <param name="local_costmap/global_frame" value="/tb3_0/odom"/>
36.
37.     <param name="DWAPlannerROS/min_vel_x" value="0.0" if="$(arg move_forward_only)" />
38.
39.     <!-- remap from="map" to="/map" / -->
40.
41.     <!-- remap from="cmd_vel" to="$(arg cmd_vel_topic)"/ -->
42.
43.     <!-- remap from="odom" to="$(arg odom_topic)"/ -->
44.
45.   </node>
46.
47. </launch>
```

Launch File 5 Task 6.2 move_base_tb3_0.launch

```
1.  <launch>
2.
3.    <!-- Arguments -->
4.
5.    <arg name="model" default="$(env TURTLEBOT3_MODEL)" doc="model type [burger, waffle,
    waffle_pi]"/>
6.
```

```
7.    <!-- Ho tolto / da cmd_vel, era /cmd_vel -->
8.
9.    <arg name="cmd_vel_topic" default="cmd_vel" />
10.
11.   <arg name="odom_topic" default="odom" />
12.
13.   <arg name="move_forward_only" default="false"/>
14.

15.   <!-- move_base -->
16.
17.   <node pkg="move_base" type="move_base" respawn="false" name="move_base"
    output="screen">
18.
19.     <param name="base_local_planner" value="dwa_local_planner/DWAPlannerROS" />
20.
21.     <rosparam file="$(find turtlebot3_navigation)/param/costmap_common_params_$(arg
    model).yaml" command="load" ns="global_costmap" />
22.
23.     <rosparam file="$(find turtlebot3_navigation)/param/costmap_common_params_$(arg
    model).yaml" command="load" ns="local_costmap" />
24.
25.     <rosparam file="$(find turtlebot3_navigation)/param/local_costmap_params.yaml"
    command="load" />
26.
27.     <rosparam file="$(find turtlebot3_navigation)/param/global_costmap_params.yaml"
    command="load" />
28.
29.     <rosparam file="$(find turtlebot3_navigation)/param/move_base_params.yaml"
    command="load" />
30.
31.     <rosparam file="$(find turtlebot3_navigation)/param/dwa_local_planner_params_$(arg
    model).yaml" command="load" />
32.
33.     <param name="global_costmap/robot_base_frame" value="/tb3_1/base_footprint"/>
34.
35.     <param name="local_costmap/robot_base_frame" value="/tb3_1/base_footprint"/>
36.
37.     <param name="local_costmap/global_frame" value="/tb3_1/odom"/>
38.
39.     <param name="DWAPlannerROS/min_vel_x" value="0.0" if="$(arg move_forward_only)" />
40.
41.     <!-- remap from="map" to="/map" / -->
42.
43.     <!-- remap from="cmd_vel" to="$(arg cmd_vel_topic)"/ -->
44.
45.     <!-- remap from="odom" to="$(arg odom_topic)"/ -->
46.
47.   </node>
48.
49. </launch>
50.
```

Launch File 6 Task 6.2 move_base_tb3_1.launch

## 3. Other Code Files

Task 7 Screen Shots

Face_Detector.py

```python
import cv2

face_data = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

webcam = cv2.VideoCapture(0)

while True:

    successful_frame_read, frame = webcam.read()

    gray_img = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    face_coordinates = face_data.detectMultiScale(gray_img)

    for (x, y, w, h) in face_coordinates:
        cv2.rectangle(frame,(x, y), (x+w, y+h), (0, 255, 0), 2)

    cv2.imshow('Raghdution Cam', frame)
    key = cv2.waitKey(1)

    if key==81 or key==113:
        break

webcam.release()

print("Complete")

```

Figure 42 Task 7 Face_Detector.py

Task 8 Screen Shots

train_chatbot.py > ...

```python
# -------------- Importing the libraries --------------------

import nltk
# CAMeL Tools is suite of Arabic natural language processing tools
# developed by the CAMeL Lab at New York University Abu Dhabi.
#from nltk.stem import WordNetLemmatizer
from nltk.stem.isri import ISRIStemmer #arabic stemmer
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

from camel_tools.tokenizers.word import simple_word_tokenize #arabic tokenizer
from camel_tools.utils.dediac import dediac_ar
from camel_tools.utils.normalize import normalize_alef_maksura_ar
from camel_tools.utils.normalize import normalize_alef_ar
from camel_tools.utils.normalize import normalize_teh_marbuta_ar
import json
import pickle
import numpy as np
import random


# importing Keras libraries
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.optimizers import SGD

# -------------- load the dataset --------------------

data_file = open('dataset.json').read()
dataset = json.loads(data_file)

# -------------- Cleaning the text --------------------
#                preprocess data


lemmatizer = WordNetLemmatizer()

# create a list of all the words in the file
words=[]
#create a list of classes for our tags
classes = []
# craate teh corpus after cleaning the text
corpus = []
ignore_chars = ['؛', '!','،',',','.','.','?']



# loop through the dataset.json file
for data in dataset['intents']:
    for pattern in data['patterns']:
    # remove stopwords with a for loop
    # after cleaining we'll add it to corpus

        #tokenize each statement into words
        w = nltk.word_tokenize(pattern)
```

Figure 43 Task 8 train.py

train_chatbot.py > ...

```
            #tokenize each statement into words
            w = nltk.word_tokenize(pattern)
            # add the tokenize words to words[]
            words.extend(w)
            #add each tokenized words to its tag e.g., ['مرحبا','أهلا'] greeting
            corpus.append((w, data['tag']))

            # add all the tags to our classes list
            if data['tag'] not in classes:
                classes.append(data['tag'])

    #return the root of the word
    words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore_chars]


    #------------------


    #Arabic_stop_words = list(stopwords.words("arabic"))


    #for w in words:
     #   if w in Arabic_stop_words:
      #      words.remove(w)
       # # Normalize alef variants to 'ا'
        #nw = normalize_alef_ar(w)
        # Normalize alef maksura 'ى' to yeh 'ي'
      #  nw = normalize_alef_maksura_ar(w)
       # # Normalize teh marbuta 'ة' to heh 'ه'
        #nw = normalize_teh_marbuta_ar(w)
        # removing Arabic diacritical marks
        #nw = dediac_ar(w)
        #words.extend(nw)

    stemmer = ISRIStemmer()

    #words = [stemmer.stem(w) for w in words if w not in ignore_chars]

    Arabic_stop_words= list(stopwords.words("Arabic"))


    [words.remove(w) for w in words if w in Arabic_stop_words]


    #--------------------------

    # sort words
    words = sorted(list(set(words)))
    # sort classes
    classes = sorted(list(set(classes)))

    #             #  print #              #

    # print corpus size
    print (len(corpus), "corpus")
```

Figure 44 Task 8 train.py

train_chatbot.py > ...

```
110
111  # print corpus size
112  print (len(corpus), "corpus")
113  # print classes size with classes = intents
114  print (len(classes), "classes", classes)
115  # print words size = all lemmatized words, vocabulary
116  print (len(words), "unique lemmatized words", words)
117
118  # to save time instead of biulding them each time we open the GUI
119  pickle.dump(words,open('normalized_words.pkl','wb'))
120  pickle.dump(classes,open('classes_2.pkl','wb'))
121
122  # -------------- Create training and testing data -------------------
123  #                creating the Bag of Words model
124
125  # Splitting the dataset into the Training set and Test set
126  # input = the pattern, output = the class our input pattern belongs to
127  # But the computer doesn't understand text, so we will convert text into numbers:
128
129
130  # create the training data
131  training = []
132  # create an empty array for the output that matches the size of classes tags
133  array_empty = [0] * len(classes)
134
135  # training set, bag of words for each sentence
136  for cor in corpus: # tokenized words with thier tags: ['مرحبا','اهلا']greeting
137      # initialize the bag of words
138      bag = []
139      # list of tokenized words for the pattern [0]['مرحبا','اهلا']
140      pattern_words = cor[0]
141      # lemmatize each word - create base word, in attempt to represent related words
142      pattern_words = [lemmatizer.lemmatize(w.lower()) for w in pattern_words]
143
144      # create our bag of words array with 1, if word match found in current pattern
145      for w in words:
146          bag.append(1) if w in pattern_words else bag.append(0)
147      # output is a '0' for each tag and '1' for current tag (for each pattern)
148      output_row = list(array_empty)
149      output_row[classes.index(cor[1])] = 1 # cor[1] = greeting
150
151      # create the training data(0's and 1's) with the [bag of words][classes]
152      training.append([bag, output_row])
153
154  # shuffle our features and turn into np.array
155  random.shuffle(training)
156  training = np.array(training)
157  # create train and test lists. X - patterns, Y - intents, classes
158  train_x = list(training[:,0]) # all rows, first col
159  train_y = list(training[:,1]) # all rows, 2nd col, which pattern belongs to which class
160  print("Training data created")
161
162
163  # -------------- Bilding the model -------------------
164
165
```

Figure 45 Task 8 train.py

```python
# -------------- Bilding the model -------------------



# Define model architecture, linear stack of layers
model = Sequential()

# Create the model
# 3 layers. First layer 128 neurons, second layer 64 neurons
# 3rd output layer contains number of neurons
# equal to number of intents to predict output intent with softmax
model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu')) #train_x = size
model.add(Dropout(0.5)) # to control overfitting problem
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]), activation='softmax')) # which pattern belongs to which class

# Compile model. Stochastic gradient descent with Nesterov accelerated gradient gives good results for this model
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True) #learning rate =0.01
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

# Fitting and saving the model
hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5, verbose=1)
model.save('chatbot_arabic_model.h5', hist) # to save time instead of training the model each time
print("model created")
```

Figure 46 Task 8 train.py

dataset.json > ...


```
{"intents": [
        {"tag": "greeting",
         "patterns": ["سلام", "السلام عليكم", "هيلو", "هلا", "أهلا", "مساء الخير" , "صباح الخير", "مرحبا", "هاي", "أهلا"],
         "responses": ["سررت برؤيتك، تفضل", "أهلا، كيف بإمكاني أخدمك؟", "حياك الله", "أهلا، كيف يمكنني مساعدتك؟", "مرحبًا بك", "أهلا وسهلا!"],
         "context_set": [""]
        },
          {"tag": "whats_up",
         "patterns": ["كيفك" , "كيف حالك؟" , "أخبارك"],
         "responses": ["أنا بخير، شكرا لك" ,"أنا بأفضل حال، وأنت؟"],
         "context_set": [""]
        },
        {"tag": "goodbye",
         "patterns": ["وداعا", "إلى اللقاء", "مع السلامة", "تود باي", "باي"],
         "responses": ["نتشوف على خير", "سررنا برؤيتك، وداعا", "مع السلامة، نورتنا", "إلى اللقاء", "تراك قريبًا، شكرا لزيارتك"],
         "context_set": [""]
        },
        {"tag": "smart_methods",
         "patterns": ["روبوتات", "ما هي سمارت ميثودز؟", "ما هي الأساليب الذكية؟", "شركة"],
         "responses": ["متخصصة في المعدات الآلية والروبوت والذكاء الصناعي وتعتبر المنشأة التجارية الأولى من نوعها في خدمة الباحثين والمبتكرين"],
         "context_set": [""]
        },
        {"tag": "bad_rating",
         "patterns": ["سيء","ليس جميل","غير جميل" , "ليس حلو" , "سيئ" , "لم يعجبني" , "لا يعجبني"],
         "responses": ["رأيك يهمنا، سنحاول التطوير، شكرا لك" , "سنحاول التطوير، شكرا لك","عفوًا، شكرا لإبداء رأيك"],
         "context_set": [""]
        },
          {"tag": "good_rating",
         "patterns": ["شكرا", "حلو", "جميله", "ماشاء الله جميل", "أعجبني", "واو" , "رائع", "جميل"],
         "responses": ["أسعدنا برأيك، شكرًا لك", "رأيك يهمنا، شكرًا لمرورك", "أسعدنا تقييمك، شكرا لك"],
         "context_set": [""]
        },
        {"tag": "etc",
         "patterns": [],
         "responses": ["هل يمكنك إعادة صياغة كلامك؟", "عذرًا، لم أفهم" , "عفوا لم أفهمك"],
         "context_set": [""]
        },
          {"tag": "chatbot",
         "patterns": ["ما اسمك؟", "من أنت؟", "من أنتم؟", "من؟", "ماذا تفعل؟"],
         "responses": ["أنا هنا لمساعدتك،إسألني أي شيء", "أنا نظام محادثة ذكي، صممت من قبل متدربين شركة الأساليب الذكية"],
         "context_set": [""]
        }
    ]
}
```


Figure 47 Task 8 dataset.json

GUI_ChatBot.py > ...

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# -------------- Importing the libraries --------------------


import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import pickle
import numpy as np
from keras.models import load_model
model = load_model('chatbot_arabic_model.h5')
import json
import random


intents = json.loads(open('dataset.json').read())
words = pickle.load(open('normalized_words.pkl','rb'))
classes = pickle.load(open('classes_2.pkl','rb'))

# -------------- Cleaning the text --------------------

def clean_up_sentence(sentence):
    # tokenize the pattern - split words into array
    sentence_words = nltk.word_tokenize(sentence)
    # stem each word - create short form for word
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_words]
    return sentence_words


# -----------return bag of words array--------------
#: 0 or 1 for each word in the bag that exists in the sentence

def bow(sentence, words, show_details=True):
    # tokenize the pattern
    sentence_words = clean_up_sentence(sentence)
    # bag of words - matrix of N words, vocabulary matrix
    bag = [0]*len(words)
    for s in sentence_words:
        for i,w in enumerate(words):
            if w == s:
                # assign 1 if current word is in the vocabulary position
                bag[i] = 1
                if show_details:
                    print ("found in bag: %s" % w)
    return(np.array(bag))
```

Figure 48 Task 8 GUI_ChatBot.py

GUI_ChatBot.py > ...

```python
# ---------------- Importing the libraries --------------------


import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import pickle
import numpy as np
from keras.models import load_model
model = load_model('chatbot_arabic_model.h5')
import json
import random


intents = json.loads(open('dataset.json').read())
words = pickle.load(open('normalized_words.pkl','rb'))
classes = pickle.load(open('classes_2.pkl','rb'))

# ---------------- Cleaning the text --------------------

def clean_up_sentence(sentence):
    # tokenize the pattern - split words into array
    sentence_words = nltk.word_tokenize(sentence)
    # stem each word - create short form for word
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_words]
    return sentence_words


# -----------return bag of words array--------------
#: 0 or 1 for each word in the bag that exists in the sentence

def bow(sentence, words, show_details=True):
    # tokenize the pattern
    sentence_words = clean_up_sentence(sentence)
    # bag of words - matrix of N words, vocabulary matrix
    bag = [0]*len(words)
    for s in sentence_words:
        for i,w in enumerate(words):
            if w == s:
                # assign 1 if current word is in the vocabulary position
                bag[i] = 1
                if show_details:
                    print ("found in bag: %s" % w)
    return(np.array(bag))

# -----------prediction of words depending on the keras model--------------

def predict_class(sentence, model):
    # filter out predictions below a threshold
    p = bow(sentence, words,show_details=False)
    res = model.predict(np.array([p]))[0]
    ERROR_THRESHOLD = 0.25
    results = [[i,r] for i,r in enumerate(res) if r>ERROR_THRESHOLD]
    # sort by strength of probability
    results.sort(key=lambda x: x[1], reverse=True)
    return_list = []
```

Figure 49 Task 8 GUI_ChatBot.py

GUI_ChatBot.py > ...

```
    results.sort(key=lambda x: x[1], reverse=True)
    return_list = []
    for r in results:
        return_list.append({"intent": classes[r[0]], "probability": str(r[1])})
    return return_list

# ------------predict the response-------------

def getResponse(ints, intents_json):
    tag = ints[0]['intent']
    list_of_intents = intents_json['intents'] #dataset.json
    for i in list_of_intents:
        if(i['tag']== tag):
            #create a random response fromt he list or responeses
            result = random.choice(i['responses'])
            break
    return result

def chatbot_response(text):
    # predict the masseage based on the trained model
    ints = predict_class(text, model) # chatbot_model.h5
    # predict the response from the function
    res = getResponse(ints, intents)
    return res



#----------Creating GUI with tkinter--------------

import tkinter
from tkinter import *
def send():
    msg = EntryBox.get("1.0",'end-1c').strip()
    EntryBox.delete("0.0",END)
    if msg != '':
        ChatLog.config(state=NORMAL)
        ChatLog.insert(END, "انت: " + msg + '\n\n')
        ChatLog.config(foreground="#442265", font=("Verdana", 12 ))
        res = chatbot_response(msg)
        ChatLog.insert(END, "ChatBot: " + res + '\n\n')
        ChatLog.config(state=DISABLED)
        ChatLog.yview(END)
base = Tk() #parent window
base.title("الأساليب الذكية")
base.geometry("400x500")
base.resizable(width=FALSE, height=FALSE)
#Create Chat window
ChatLog = Text(base, bd=0, bg="white", height="15", width="100", font="Arial",)
ChatLog.config(state=DISABLED)
#Bind scrollbar to Chat window
scrollbar = Scrollbar(base, command=ChatLog.yview, cursor="heart")
ChatLog['yscrollcommand'] = scrollbar.set
#Create Button to send message
SendButton = Button(base, font=("Verdana",12,'bold'), text="ارسل", width="12", height=3,
                    bd=0, bg="green", activebackground="light green",fg='#ffffff', command = send )
```

Figure 50 Task 8 GUI_ChatBot.py

```
#Create the box to enter message
EntryBox = Text(base, bd=0, bg="white",width="100", height="3", font="Arial")
#EntryBox.bind("<Return>", send)
#Place all components on the screen
scrollbar.place(x=376,y=6, height=386)
ChatLog.place(x=6,y=6, height=386, width=370)
EntryBox.place(x=128, y=401, height=90, width=265)
SendButton.place(x=6, y=401, height=90)
base.mainloop()
```

Figure 51 Task 8 GUI_ChatBot.py

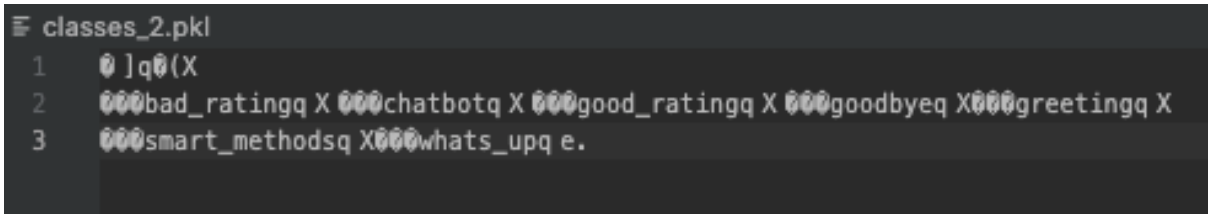



Figure 52 Task 8 classes_2.pkl

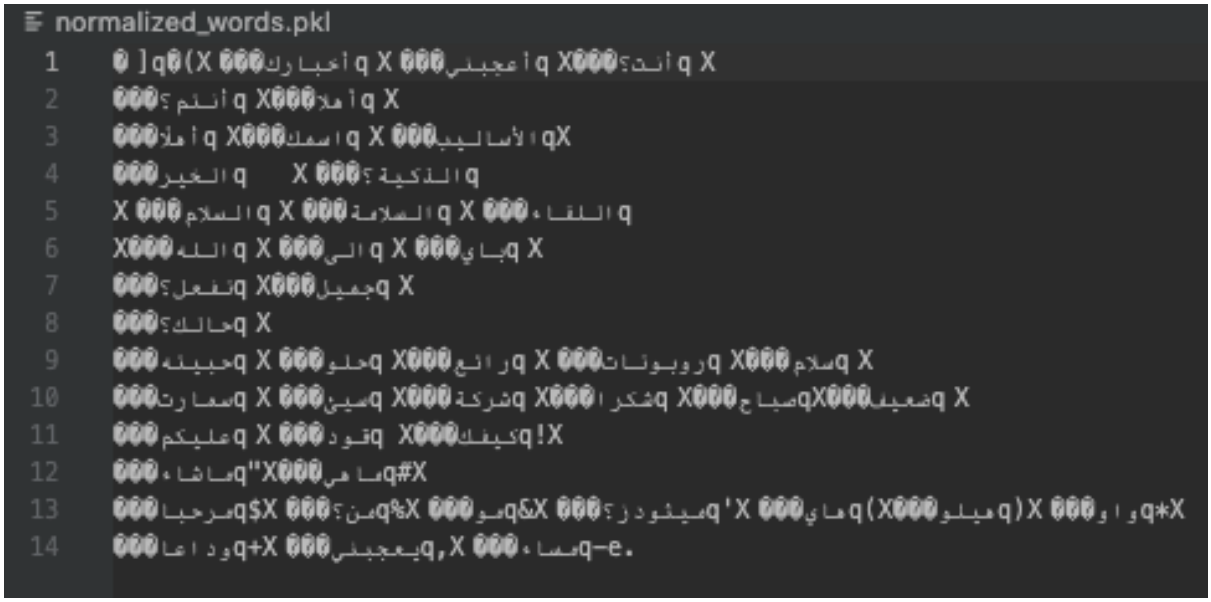



Figure 53 Task 8 normalized_words.pkl

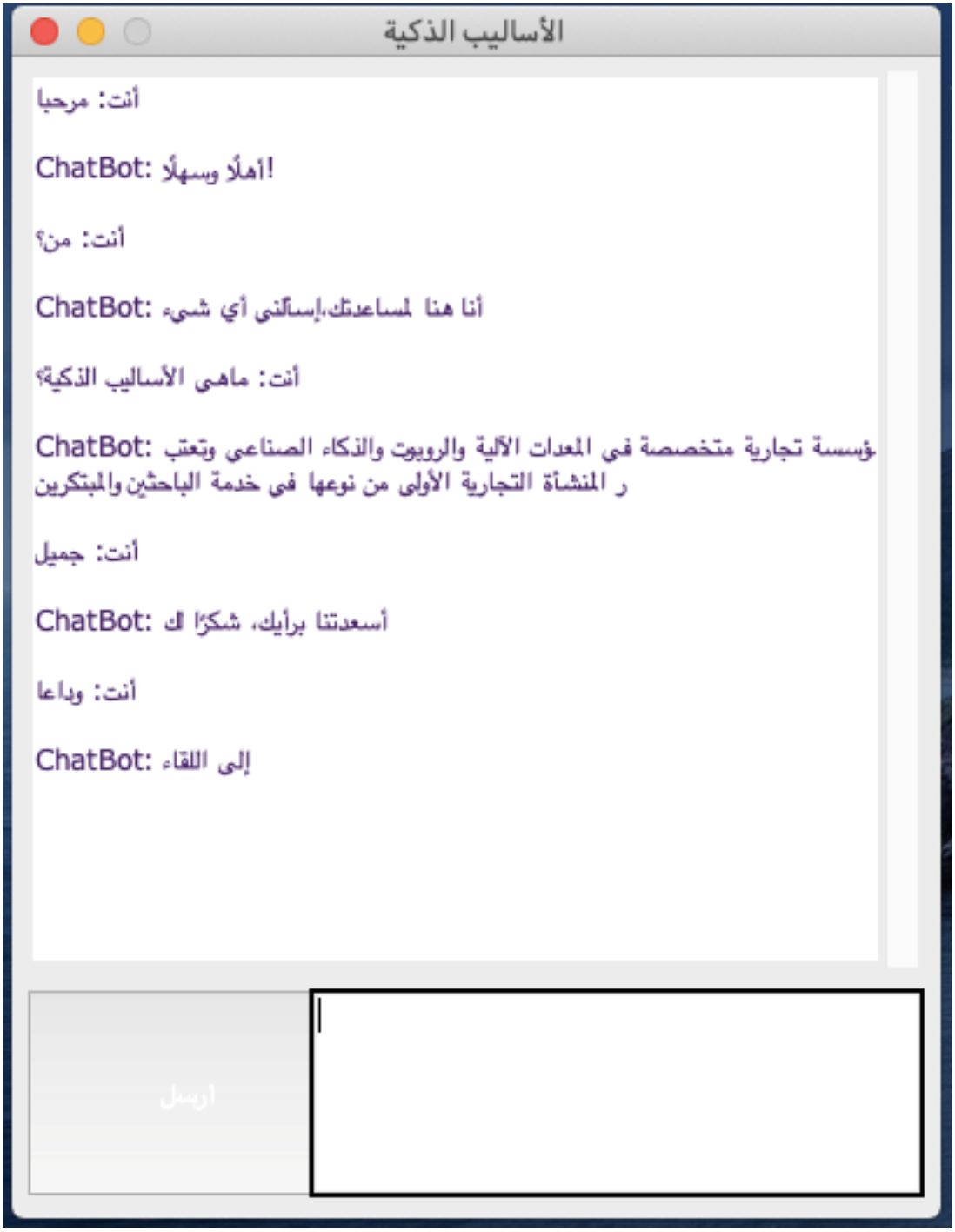


Figure 54 Task 8 Graphical User Interface of the Chatbot

# Appendix – B

After the fourth week of training, CEO. Eng. Wessam Munshy told us that only eleven trainees in the summer training from all tracks, will receive the certificate early in the website training gate due to their excellence in the training. And I was proudly, one of them.

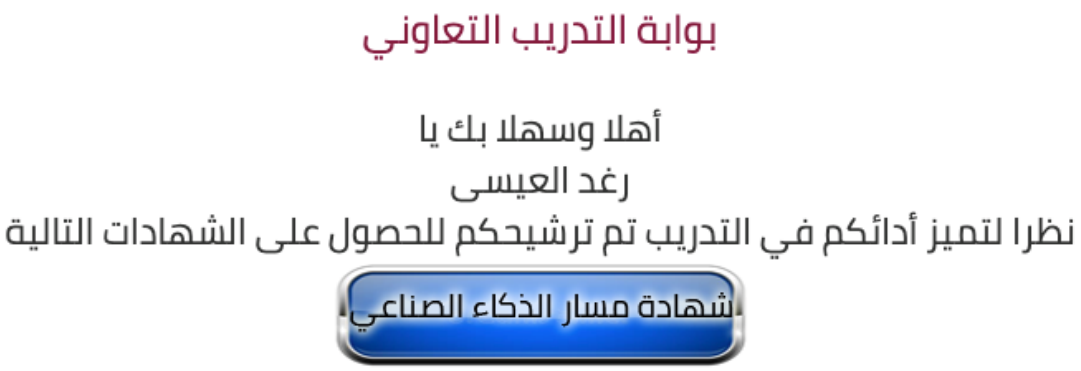



Figure 55 Excellence Certificate

Summer internship 2021

SmartMethods
الأساليب الذكية

إفادة

تفيدكم مؤسسة الأساليب الذكية بأن السيد/ السيدة
رغد اسماعيل العيسى
قد حضر البرنامج التعاوني الصيفي لعام 2021 وقد تم إنجاز عدد من المهام وقدرها 13 مهمة
والمقام خلال الفترة بين 6 / 6 /2021 إلى 31 / 8 / 2021 بمعدل 420 ساعة
وقد أعطيت له هذه الإفادة له بناء على طلبه بشكل ألي دون أدنى مسوؤلية تجاه الغير..
والله ولي التوفيق

The Smart Methods Est inform that Mr./Ms
Raghad Esmael Alessa
Has attended the community training at SmartMethods Est.
during the summer of 2021 and during the internship, he/she completed 13 tasks held between 6 /6 /2021, to 31 /8 /2021, which is equivalent to 420 working hours
this document has been autogenerated upon the request of the abovementioned consignee with no liability whatsoever from our side.

CEO
Wessam Munshi

مؤسسة الأساليب الذكية للتجارة
مركز الأبحاث والتطوير
س.ت ٤٠٣١٠٦٠٤٨٥
C.R. 4031060485
R & D Center
Smart Methds Teading Est

المملكة العربية السعودية - مكة المكرمة - وادي مكة - معامل الابتكار
www.s-m.com.sa

Figure 56 Statement of Completion