

**RAFIK HARIRI UNIVERSITY**

**BOMB DISPOSAL ROBOT**

Done by

**ABED EL RAHMAN HAMMOUD**

**MOUNIR NAHLE**

**PIERRE GERGES**

**RAMI ASSAF**

Submitted to

**DR. HASSAN HARIRI**

This senior project is submitted in partial fulfillment of the requirements of the  
BE degree of Mechatronics Engineering Major of the College of Engineering at Rafik Hariri  
University

**MECHREF, LEBANON**

**APRIL 2022**

Copyright©2022, all rights reserved

Abed El Rahman Hammoud (2018-0003)

Mounir Nahle (2017-0035)

Pierre Gerges (2020-0617)

Rami Assaf (2020-0603)

## ACKNOWLEDGMENTS

This project could not have been successfully completed without the supervision of our advisor Dr. Hassan Hariri.

Recognition is owed to Rafik Hariri University Alumnus, Eng. Hassan Daoud for his Kinect Camera donation, Eng. Hussein Harb for the time he spent with us during the training workshops we had and finally Eng. Imad Khodor for his donation of the tank and the robotic arm. Their help was crucial to the success of the project.

Special thanks to our parents and friends who walked with us through all the phases of the project and provided us with the emotional and mental support.

## ABSTRACT

Bombs are considered one of the most dangerous creations done by mankind. Mixing chemical components with technologies lead to the creation of devices capable of massive amounts of destruction. This destruction is not limited to property damage only, but also to loss of life. To combat this, technology rises against technology once more. What is created to destroy, can be also created to make peace and save lives. This project paper discusses the potential to create a robot capable of disposing explosives from a safe range, safeguarding precious blood and preventing loss of life.

The intention of this project is to provide a prototype robotic machine, capable of providing a live feed of the explosive's environment using a Kinect camera, and the ability to control a tank equipped with a robotic arm capable of disposing and securing the destructive charge, thus removing the needed human element from interfacing with the explosives. A safe distance away, the user is given control of the machine in order to perform the necessary action of disposal.

**Keywords:** Bombs – Technologies – Prototype robotic machine – Disposal – Robotic arm – Tank – Control – Raspberry Pi – Kinect camera.

# TABLE OF CONTENTS

## Table of Contents

ACKNOWLEDGMENTS .....	i
ABSTRACT.....	ii
TABLE OF CONTENTS.....	iii
LIST OF FIGURES .....	vi
LIST OF TABLES .....	viii
CHAPTER 1 INTRODUCTION .....	1
1.1 Problem Statement .....	1
1.2 Solution .....	2
1.3 Goals.....	2
1.4 Objectives.....	2
1.5 Project and Team SWOT Analysis .....	2
1.6 Conclusion.....	3
CHAPTER 2 LITERATURE REVIEW .....	5
2.1 Introduction .....	5
2.2 In-Market Competitors.....	7
2.2.1 Caliber T5: Compact, Two-Man Portable System.....	7
2.2.2 Talon Small Mobile Robot.....	9
2.2.3 Vanguard.....	10
2.2.4 Mini-Andros.....	11
2.3 Interview with Experts .....	12
2.4 Our Approach.....	13
2.4.1 Tank Navigation.....	13
2.4.2 Robotic Arm.....	14
2.4.3 RGBD Kinect Camera .....	15
CHAPTER 3 DESIGN.....	16
3.1 Chosen Specifications .....	16
3.2 Design.....	17
3.2.1 Robotic Arm Selection.....	17
3.2.2 Torque Sizing of Our Arm Servo Motors .....	18
3.2.3 Tank Selection .....	21
3.2.4 Torque Sizing for the Tank Motors .....	21

3.2.5	Gripper Selection .....	22
3.2.6	Waterproofing Our Robot.....	23
CHAPTER 4	HARDWARE USED.....	24
4.1	TS-100 Tank Chassis .....	24
4.2	Lewansoul Robotic Arm .....	25
4.3	RGB-D Kinect Camera .....	27
4.4	Reverse Engineering Design .....	27
4.4.1	Robot Battery Sizing.....	27
4.4.2	Power Bank Battery Sizing.....	30
4.4.3	Torque Sizing.....	31
4.5	Bill of Material .....	32
4.6	General Wiring Diagram.....	33
CHAPTER 5	IMPLEMENTATION AND RESULTS .....	35
5.1	Tank Chassis Implementation .....	35
5.2	Robotic Arm Implementation.....	37
5.3	RGB-D Kinect Camera .....	39
5.4	Arduino Integration .....	41
CHAPTER 6	CONCLUSION AND FUTURE WORK.....	44
6.1	Future Work in Navigation .....	44
6.2	Future Work in Bomb Grasping.....	44
6.3	Future Work in Vision and Bomb Detection .....	45
APPENDIX A	SIMULATION WORKSHOP.....	46
i	Robot Tank Simulation.....	46
ii	Lewansoul Robotic Arm Simulation .....	50
iii	Dobot Robotic Arm Simulation.....	61
iv	Assembling Tank and Arm as One Robot .....	69
APPENDIX B	CODES USED.....	71
i	Arduino Code Used to Control the Tank .....	71
ii	Arduino Code Used to Control the Arm.....	74
APPENDIX C	DATA SHEETS .....	78
i	Tank DC Motors Data Sheet.....	78
ii	Robotic Arm Servo Motors Data Sheets .....	79
APPENDIX D	SURVEY .....	81
i	About this Questionnaire .....	81

ii Questions .....	81
APPENDIX E MEETING MINUTES.....	84
APPENDIX F STANDARDS .....	100
REFERENCES .....	107

## LIST OF FIGURES

Figure 1: Caliber T5 .....	8
Figure 2: Talon Robot .....	10
Figure 3: Vanguard Robot .....	11
Figure 4: Selected Robotic Arm .....	17
Figure 5: Front and Side Reach .....	17
Figure 6: Robotic Arm Links and Joints .....	18
Figure 7: Towerpro MG996R Servo Motor .....	19
Figure 8: DS3235 Servo Motor .....	20
Figure 9: Corona BL1029HV Servo Motor .....	20
Figure 10: Motor Placement of Our Arm .....	20
Figure 11: Selected Tank Chassis .....	21
Figure 12: Gripper Selected .....	22
Figure 13: Captured Photo of the TS-100 Tank Chassis .....	25
Figure 14: Captured Photo of the Lewansoul Robotic Arm .....	26
Figure 15: Robotic Arm Wiring Diagram .....	27
Figure 16: Measurement in University Lab for Motor Drive .....	28
Figure 17: Measurement in University Lab for DC-DC Converter .....	29
Figure 18: Raspberry Pi Input Power .....	30
Figure 19: Robot Weight Scaling .....	31
Figure 20: Wheel Design on SolidWorks .....	32
Figure 21: General Wiring Diagram .....	34
Figure 22: Bottom Components .....	36
Figure 23: Schematic of the Tank Wiring .....	36
Figure 24: Arm Servo Motors .....	37
Figure 25: Default Joints on Rviz .....	38
Figure 26: Changes Applied to mtrDegree Variable .....	38
Figure 27: Changes Applied for Mirroring .....	39
Figure 28: Different Sensors Inside the Kinect Camera .....	39
Figure 29: Photo Captured from the Camera .....	40
Figure 30: Arduino Launch File Script .....	42
Figure 31: Assembled Robot .....	43
Figure 32: Tank Dimensions .....	46
Figure 33: Robot Tank Assembly .....	46
Figure 34: Tank Model on Rviz .....	47
Figure 35: Added Gazebo Plugin .....	48
Figure 36: Topics Available .....	48
Figure 37: Odometry .....	49
Figure 38: Assembly of the Arm .....	50
Figure 39: Arm Module Loaded on Moveit .....	51
Figure 40: Self-Collision Checking .....	51
Figure 41: Virtual Joint Created .....	52
Figure 42: Planning Groups Settings Window .....	52
Figure 43: Chosen Planning Groups .....	53
Figure 44: Robot Arm Pose .....	53



Figure 45: Define the End Effector.....	54
Figure 46: ROS Controllers Setup .....	54
Figure 47: Generate Configuration Files .....	55
Figure 48: Arm Visualization on Rviz.....	55
Figure 49: Transmission Tag for J1 .....	56
Figure 50: Gazebo Plugin Added.....	56
Figure 51: Gripper Controller Added .....	57
Figure 52: Arm Controller Added.....	57
Figure 53: Argument Added .....	58
Figure 54: Error 1 Solution .....	58
Figure 55: Robotic Arm Upside Down on Gazebo.....	59
Figure 56: Added Fixed Link.....	59
Figure 57: Dobot Assembly .....	61
Figure 58: Dobot Arm Module Loaded in Moveit.....	62
Figure 59: Self-Collision Matrix.....	62
Figure 60: Choose Planning Groups Links .....	63
Figure 61: Home Pose.....	64
Figure 62: Generate Config Files.....	64
Figure 63: Rviz Simulation.....	65
Figure 64: Transimmison Tags Added .....	66
Figure 65: Gazebo Plugin Added.....	66
Figure 66: Arm Controller Added.....	67
Figure 67: Argument Added .....	67
Figure 68: Joint Added.....	69
Figure 69: Assembled Robot .....	70
Figure 70: Tank DC Motors Data Sheet .....	78

## LIST OF TABLES

Table 1: Project SWOT Analysis.....	3
Table 2: Team SWOT Analysis.....	3
Table 3: Caliber T5 Specifications .....	8
Table 4: Theoretical Specifications Table .....	17
Table 5: Datasheet of Battery Used .....	30
Table 6: Bill of Material .....	33
Table 7: LDX-335MG .....	79
Table 8: LDX-218MG .....	79
Table 9: LDF-06 .....	80
Table 10: LD-1501MG .....	80
Table 11: ABET KPIs.....	106

# CHAPTER 1

## INTRODUCTION

Robots have always been perceived to be the path for the future, due to their versatility and ability to mimic human actions and skills. This has caused a race between international companies and organizations to produce the best and closest humanoid automations. Using these machines, humans should be able to find solutions to their problems, such as being in more than one place at a given time, handling dangerous environments and producing an increased labor force for the jobs that cause issues for humans.

The following sections contain a discussion for the problem statement, its solution as well as the main idea development. It also contains the goals of the project, the objectives, project SWOT analysis, and team SWOT analysis (stands for: Strength, weakness, Opportunities, and Threats). In the conclusion part of the chapter, a short description shows the contents of the project report.

### **1.1 Problem Statement**

Technicians are often greeted with challenges that pose significant safety risks to themselves as well as those within their proximity. The risks and safety concerns for technicians becomes particularly prevalent when dealing with the process of disarming and diffusing improvised explosive devices also commonly referred to as “IEDs”. Due to the unpredictable environment that is associated to the deployment of IEDs, technicians are often reluctant to engage in the situation. As a result, robotic devices are sent as replacement in applicable situations. A downside to this inventive solution is that robotic devices lack the capabilities to intuitively act and make decisions as they do not have the humanistic thinking capabilities. Therefore, they lack the ability to disarm the bombs and need human interaction to diffuse it. It should be noted that a significant percentage of robots are controlled remotely with the operator having to look at the display screen that shows the field where the bomb is placed and having to control it using many push buttons and key knobs to control the speed of the robot, camera movement, turning the arms on or off and a joystick to control the direction of the robot and its different joints. Having all those actions may cause high pressure and low efficiency during the operation therefore may be life threatening.

## 1.2 Solution

The solution would be creating a robot composed of a small tank mounted with a robotic arm that can be controlled wirelessly using a control unit that will be used by the bomb disposal technician.

## 1.3 Goals

By the end of the senior year, the team aims to successfully achieve the following goals:

- Prevent human interference in life threatening situations.
- Making sure that the robotic tank can move in any direction and over any terrain being controlled by the operator using a keyboard.
- Enable the robotic arm to handle different objects.
- Allow the robotic arm to move the joints at the same time rather than one at a time by performing ROS manipulation and perception (object detection).

## 1.4 Objectives

The objectives of the team that are to be fulfilled by the end of the senior year are:

- Finalizing a well-functioning prototype.
- Disposal of a bomb.
- Handle different kind of objects not only bomb shaped objects.
- Receive live feed of the situation by using a Kinect camera by performing ROS perception.

## 1.5 Project and Team SWOT Analysis

Internal factors (Strength and Weakness) and external Factors (Opportunities and Threats), the advantages and disadvantages of the project will be shown in Table 1 SWOT analysis of the team will be shown in Table 2.

<b>Strength</b>	<b>Weakness</b>
<ul style="list-style-type: none"><li>- Supports a very large number of individual movements and articulations.</li><li>- Meets certain weight restrictions.</li><li>- Not too brittle, flexes to store and release mechanical energy from certain impacts.</li></ul>	<ul style="list-style-type: none"><li>- Can be sabotaged by hackers.</li><li>- The project's arm may not be able to carry heavy loads.</li><li>- Restrictions concerning accuracy levels.</li></ul>
<b>Opportunities</b>	<b>Threats</b>
<ul style="list-style-type: none"><li>- Can be used among military equipment.</li><li>- Can lead exploration missions.</li><li>- Surgical operations.</li></ul>	<ul style="list-style-type: none"><li>- Cyber-attacks.</li><li>- Control errors from software.</li></ul>

- Bomb disposing capabilities.	- Mechanical failures that can slow the workflow of the project. - Expensive repairing part especially during the Lebanese economy crisis.
--------------------------------	---

Table 1: Project SWOT Analysis

<b>Strength</b>	<b>Weakness</b>
- The team is composed of four mechatronics engineering students. - The team is eager for making the Bomb Disposal Robot. - Our colleagues are good with ROS and programming and we can use their help.	- First time to design and implement a big scale project. - Difficulty in programming. - Economy Crisis in Lebanon and its repercussions on our project budget and capabilities.
<b>Opportunities</b>	<b>Threats</b>
- Learn more about mechanical engineering by understanding the work of the hardware available to us. - Get better in software programming.	- The arm's joints may fail to operate as they should. - Wrong calculation makes the arm lose accuracy. - Covid-19 pandemic and the unfortunate chance of losing physical contact with teammates.

Table 2: Team SWOT Analysis

## 1.6 Conclusion

This section has been used to introduce our project's idea. We explained what the problem is about and how we will solve it. We mentioned our goals and objectives that we aim to achieve by the end of our senior year.

In this report, we extensively discussed in 6 chapters the procedure that we followed to reach our goal. This first chapter includes the introduction, the second one features the literature review that consists of the previous work done by other competitors, our BE1 period, and our approach. Afterwards in chapter 3, we introduced the theoretical specifications and constraints, and all the design and calculations done by the team. Whereas in chapter 4, all the hardware used in this project and how they were implemented is discussed. Chapter 5 discusses all the procedures that were implemented to accomplish a fully functional Bomb Disposal Robot and the results obtained. Finally, chapter 6 covers the conclusion of how to deal with all the problems faced and

the future work to be done in order to enhance the prototype and make it a fully operational industrial and commercial robot.

## CHAPTER 2

### LITERATURE REVIEW

#### 2.1 Introduction

In the recent decade, mobile robotics has advanced significantly, with a growing number of robots entering actual field duty. Military and law enforcement have been the two most active mobile robot application sectors. For nearly 40 years, bomb disposal robots have been employed to securely deactivate explosive devices, and they have been deployed hundreds, if not thousands, of times. The Oxford English Dictionary defines a robot as "a machine capable of performing a complex range of actions autonomously." However, bomb disposal robots are incapable of making contextual judgements or operating autonomously. Instead, they are more accurately classified as drones since, like unmanned vehicles, they are remotely controlled by a human operator from a distance. These robots serve as a remote presence for bomb disposal professionals. These robots act as a remote presence for bomb disposal specialists, known as "bomb doctors" in the British Army. This enables them to fully evaluate equipment without putting themselves or others in danger. After checking the device, the robot should be able to deactivate the explosive.

One of the first bomb disposal robots created was the Wheelbarrow Mark 1. In 1972, Lieutenant-Colonel Peter Miller of the British Army came up with the idea of using the chassis of an electrically-powered wheelbarrow to tow suspect devices, such as car bombs, so they could be safely detonated without harming anyone. (Allison, 2016)

Bomb disposal robots have undergone a dramatic transformation – from the first Wheelbarrow. “In the past 10 to 15 years, the amount of R&D and talent that’s associated with just the robotics field, in general, has really improved. Bomb disposal robots have really fed off of that,” says Ultralife Corporation application engineering manager Jonathan DiGiacomandrea.

“The pace at which these robots are improving has increased significantly. It’s really a rapidly developing field now. Robots in general are becoming much more precise and reliable than they were in the early days.” (Gardner, 2020)

The biggest difference between modern bomb disposal units and early models is the method of user control. By advanced technology, a telecommunications cable was used to transmit commands to the robot’s electrical systems. However, cables gave the bomb disposal robot a

restricted operational radius. There was also the risk of the cable becoming caught or tangled on objects – much like you might encounter with a garden hose or a vacuum cleaner (Allison, 2016).

Bomb disposal robots' design has remained the same since they were first created. Despite the advancements in technology, they still have the same basic idea: to destroy explosive devices. Bomb disposal robots' mobility has evolved from a single pair of tank-like caterpillar tracks to variations with two pairs of caterpillar tracks and others with six or more wheels. This enables bomb disposal robots to cross increasingly tough terrain. The arm of bomb disposal robots provides for a significant deal of flexibility. Most bomb disposal teams now carry a variety of instruments that may be added to them. This enables the robot to avoid different impediments that might otherwise limit its movement, such as cutting wire fences with wire cutters and grapping. Given that bomb disposal robots are designed to work in a range of adverse environments, they can sustain substantial punishment. "The majority of the expense goes towards making the electronics and sensors resilient under incredibly harsh environments," Vijayakumar explains. "Not quite as much as in space, but close." Bomb disposal robots range in size from backpack-sized robots that can be carried on a soldier's back and hurled into buildings to the size of a ride-on lawnmower equipped with x-ray machines and explosives detectors. (Allison, 2016)

Initially, the controls for these robots were sophisticated, demanding specialist training; now, they are controlled by gaming console controllers. Furthermore, the remote controllers used to operate many robot designs are increasingly being outfitted with haptic sensory feedback systems, which vibrate to provide danger flags - similar to gaming. In such high-pressure environments, the controls must be as intuitive and simple as possible. (Gardner, 2020)

Innovations in battery power supply have also allowed bomb disposal systems to be constructed considerably lighter, making them more portable, and to function for much longer periods of time without needing to be recharged, ensuring that they do not fail at a critical time. "Given the circumstances and the environment, battery failure could be catastrophic, but all too likely in hostile surroundings", says DiGiacomandrea. Previously, lead acid batteries were used in robotics, but recent developments in lithium-ion technology mean that battery weight may be reduced by half or even two-thirds, which will be critical as robots grow lighter and more agile while packing in twice the energy density. (Gardner, 2020)

Because of advancements in robotics and remote control systems, bomb disposal robots will become more adaptive to their environments in the future. Prototypes that can jump over



barriers and land on the other side are being created. Others are being created with two arms, giving bomb disposal robots greater physical dexterity, such as the ability to open car trunks and examine inside. Instead of relying on a single robot, function-specific robots are being developed. These will work in groups, with one robot tasked with sniffing out explosives and another with the disposing of them. The number of lives saved by bomb disposal robots has increased as technology advances. "One of the target areas in terms of utilization of robots is getting into risky circumstances," Vijayakumar concludes. "Robots can go in, be controlled from a safe distance, and be sacrificed in the worst-case situation." (Allison, 2016)

## **2.2 In-Market Competitors**

In this section, we will talk about projects done by other organizations and engineers. These projects are somehow similar to our final project. We will showcase the researched projects' main features and specifications over a wide range of fields. By doing that, our aim is to gain some experience on how our project will be shaped and implemented (Hardware and Specifications). We researched four distinct and highly reputable bomb disposal robots that are used by armies all around the world. The surveyed robots are: CALIBER® T5 – Talon – Vanguard – Mini-Andros.

### ***2.2.1 Caliber T5: Compact, Two-Man Portable System***

The CALIBER® T5 is a high-end, technologically advanced robot that is created by ICOR Technology. Small and lightweight, the CALIBER® T5 is a compact, two-man portable system. The T5 is best suited to assist EOD (Explosive Ordnance Disposal Specialist) and SWAT teams in inspecting and retrieving of suspicious devices from narrow passages of buses, trains and planes. Its turreted claw/disruptor arm integrates the remote handling capabilities of a robotic claw with the render-safe and breaching capabilities of a disruptor. The CALIBER® T5 comes equipped with many standard top-notch features, as listed below:

- Command and Control Unit (CCU) with multi-function control and variable-speed joystick.
- On deck 3D robot layout with LED button and user feedback.
- Data and video frequency (RF) wireless systems.
- Two-way communication (talk/listen) with transmitter and receivers and push to talk functionality.
- Two separate battery packs of 8 Ah 24V DC lead acid, rechargeable.
- Rubber tracks on 6 solid core wheels with quick-release coupler.

- 3-speed drive system with manual anti-flip bar for climbing stairs.
- Lifts up to 45 lbs. (20 kg) arm retracted and 18 lbs. (8 kg) arm extended.
- 360° rotating robotic claw with color Charge-Coupled Device (CCD) camera on telescoping arm.
- Single-mounting bracket for recoilless disruptor and 4 isolated circuits.
- 36X optical zoom pan-tilt-zoom (PTZ) color camera.
- Front- and rear-drive color CCD cameras with infrared (IR) light.
- Standard 500 ft. (152 m) tether reel with slip ring.
- LED lights (arm and PTZ camera).

ICOR Technology specified all of its specifications so that potential customers can have a clear look of the product as listed in table 3 below. According to the company, customers range from ordinary and everyday people, governments and armies or top secret special forces groups. (ICOR Technology, 2019)

Mission time	2-4+ hours
Width	43 cm
Height (in stowed position)	56 cm
Length	91 cm
Weight	68 Kg with batteries
Ground Clearance	7 cm – allows for driving in snow and sand
Drag Capacity	113 Kg
Stair-climbing angle	20 cm stairs at 45° with suitable traction
Weather resistance	Environmentally sealed, Chem-Bio wash-down capability
Third party tested	National institute of standards and technology
Price	74300\$

Table 3: Caliber T5 Specifications

Finally, you can have a look at the robot in figure 1 below that is released by the designing company on their official website.



Figure 1: Caliber T5

### **2.2.2 Talon Small Mobile Robot**

Talon is a powerful, lightweight, versatile robot designed for missions ranging from reconnaissance to weapons delivery. The Talon robot is used for bomb disposal. It is operated by radio frequency and equipped with four video cameras that enable troops to determine which areas enemy soldiers occupy. In addition, the Talon is waterproof up to 100 feet, allowing it to search for explosives off-land. The Talon also was used to locate victims and debris at the World Trade Center. It was developed for the EOD Technology Directorate of the Army's Armament Research, Development and Engineering Center at Picatinny Arsenal, NJ by the engineering and technology development firm Foster-Miller. (Duddu, 2020) All the Talon's features are listed below:

- Its large, quick-release cargo bay accommodates a variety of sensor payloads, making TALON a one robot solution to a variety of mission requirements.
- Built with all-weather, day/night and amphibious capabilities standard, TALON can operate under the most adverse conditions to overcome almost any terrain.
- The suitcase-portable robot is controlled through a two-way RF or F/O line from a portable or wearable Operator Control Unit (OCU) that provides continuous data and video feedback for precise vehicle positioning.
- TALON's payload and sensor options include: multiple cameras (color, black and white, infrared, thermal, zero light), a two-stage arm, gripper manipulators, pan/tilt, two-way communications, NBC (nuclear/biological/chemical) sensors, radiation sensors, UXO/countermine detection sensors, grenade and smoke placing modules, breaching tools, communications equipment, distracters and disrupters.
- The robot features fixed-focus infrared illuminated gripper-mounted camera, elbow and rear cameras, dimmable LED lights and a 26x optical-12x digital auto focus color zoom camera (300:1). It can be optionally fitted with 200m camera (40:1), thermal color or black and white cameras, MV-14-night vision, pan / tilt / mast and WARRVS / Fish Eye cameras.
- The system runs off AC power or lithium batteries. The control box weighs about 30 pounds.

As revealed by the company, the Talon can be seen in figure 2 below.



Figure 2: Talon Robot

### 2.2.3 *Vanguard*

The Digital Vanguard Remotely Operated Vehicle (ROV) is a multipurpose robot for responding to EOD and CBRN (Chemical, biological, radiological and nuclear) threats, surveillance and tactical missions. Its low profile, mobility and dexterity enable this robot to access confined spaces such as under vehicles, as well as the aisles and storage compartments of airplanes, buses and trains. The Digital Vanguard, also called the DV, bomb disposal robot has an auxiliary port and telescopic arm to deploy an array of bomb disposal equipment including disruptors, cameras, ECM, X-ray equipment and CBRN sensors. The features of this robot as disclosed by the US Department of Justice are as follows: (US Department of Justice, 2004)

- Auxiliary port for additional sensing and detection devices.
- Permits simultaneous use of multiple features.
- Continuous rotation claw.
- 2 independent firing circuits.
- 3 standard cameras and an optional fourth IR or disruptor camera.
- Variable speed control and excellent lifting capability.
- Powerful zoom camera and two-way digital audio.
- Mission duration of 5+ hours dependent on mission.
- Compact storage for transport in small response vehicle.
- Telescopic and articulated arm with 6 axes of movement.
- Ascends and descends stairs.

The Vanguard robot is shown in the figure 3 below.



Figure 3: Vanguard Robot

#### ***2.2.4 Mini-Andros***

The ANDROS is a series of remote control military robots designed by REMOTEC, a subsidiary of Northrop Grumman. The ANDROS series is primarily designed for military, explosive ordnance disposal (EOD), and law enforcement or SWAT applications. The ANDROS gained notability in 2016 for being involved in the first instance of a robot being used to kill another person, Micah Xavier Johnson, the gunman involved in the 2016 shooting of Dallas police officers. The newly redesigned Mini-Andros allows it to maneuver in areas otherwise inaccessible to larger robots. Utilizing the same proven technology as other systems, the Mini-Andros will set the standard for future robots. (NORTHROP GRUMMAN, 2016)

The features of the Mini-Andros as disclosed by REMOTEC are the following:

- The Mini-Andros travels on two tracks equipped with movable "arms." When extended, these arms broaden the vehicle's base, enabling it to climb and descend stairs as well as cross small ditches. When retracted, the vehicle can maneuver more easily in tight spaces. This feature, called "articulated tracking," is patented by Remotec, the company that sells and manufactures the robot. Speed: variable, between five and 70 feet per minute
- An operator can control the Mini-Andros either by a portable or fiber optic cable—or remotely, by radio control. The operator's "control station" can be mounted on a two-wheeled cart, in a briefcase or in a backpack.
- Mini-Andros can fit in the trunk of a car.

- It has a moveable arm that can lift 15 lbs when fully extended. It can reach 24" horizontally and 36" above the floor.
- The robot comes equipped with three low-light video cameras. The cameras are located on the robot's movable arm, on an arm on top of the vehicle, and on the front of the chassis. Mini-Andros can also be equipped with infrared cameras.
- Mini-Andros allows for two-way audio communication; both a microphone and a speaker are mounted on the robot.
- A 12-gauge sidewinder shotgun can be mounted on the robot, as can a window breaker, a water disrupter (used for disabling bombs), or a charge dropper assembly.
- In nuclear contamination situations, Mini-Andros can be equipped with a smear sampler, a contamination containment box, and a radiation detector.

### **2.3 Interview with Experts**

As part of determining the appropriate specifications of our prototype, and the need of the Lebanese Market in such robots, we had to communicate with experts within the bomb disposal field.

In Lebanon, experts range from non-governmental organizations or entities like the Mines Advisory Group (MAG), and governmental organizations like the Lebanese Army. MAG assists people affected by landmines, unexploded ordnance, and small arms and light weapons. They take a humanitarian approach to landmine action. They focus on the impact of their work on local communities. This approach recognizes that although the number of landmines in an area may be small, the effect on a community can be crippling. Targets are therefore determined locally, in response to liaison with affected communities, and local authorities. Likewise, the Lebanese Army possess a specialized division within its operations groups that is skilled in bomb and mine disposal.

For that, we contacted both organizations separately, and we were able to get an interview with the Lebanese Army specialized division at Chekri Ghanem Military Base in Yarze. The interview was conducted by our colleague Abed El Rahman Hammoud and he was able to deliver all our questions to the lieutenant in charge and extract all the appropriate information that were essential to determine our robot specifications. The interview's questions and answers can be referred to in Appendix D.

## 2.4 Our Approach

As previously said, bomb disposal robots serve as a remote presence for bomb disposal professionals since they are controlled remotely from a distance by a human operator, keeping them safe. To do this, we designed a technique for the robot to navigate its environment by transmitting orders from the computer keyboard. However, during the disposal phase, we add a 6 DOF robotic arm, giving our robot a lot of freedom.

Therefore, to achieve proper functionality for such a complex system using the latest well-known robust algorithms for each of the previously mentioned blocks, our approach is to utilize the power of the “Robot Operating System” which is well known as “ROS”.

ROS is a free and open-source meta-operating system for robots. It delivers the features you'd expect from an operating system, such as hardware abstraction, low-level device control, implementation of widely used functions, message transmission between processes, and package management. It also includes tools and libraries for acquiring, constructing, writing, and executing programs on numerous machines. (AmandaDattalo, 2018).

### 2.4.1 Tank Navigation

To allow the tank to travel, we must first run the simulation to ensure that all aspects are correct. To do this, we use SolidWorks software to create a drawing of the tank with the identical dimensions. The drawing is then converted to a URDF file using an extension (sw urdf exporter) SolidWorks to URDF exporter.

The URDF (Universal Robot Description Format) model is a set of files that explain the physical description of a robot to ROS. This application (ROS) uses these files to inform the computer what the robot looks like in real life. URDF files are required for ROS to comprehend and simulate situations with the robot before a researcher or engineer purchases the robot. (How URDF Models and 3D Models Can Help Your Next Robotics Project, n.d.) for detailed tutorial check Appendix A.

To visualize our tank robot, we use Rviz and Gazebo after we have uploaded the URDF file. Rviz is a three-dimensional visualization tool for ROS applications. It displays your robot model, captures sensor data from robot sensors, and replays acquired data. It can show data from cameras, lasers, 3D and 2D devices, as well as images and point clouds (AWS RoboMaker, 2022). Gazebo, on the other hand, is a 3D simulator, whereas ROS acts as the robot's interface. When you combine the two, you have a strong robot simulator. You may use Gazebo to construct a 3D scene

on your computer that includes robots, barriers, and a variety of other items. Gazebo also use a physical engine for illumination, gravity, inertia, and other functions. You can analyze and test your robot in challenging or hazardous settings without endangering it. Most of the time, running a simulator instead of starting the entire scenario on your real robot is faster. Gazebo was originally intended to assess robot algorithms. Many applications, such as error handling, battery life, localization, navigation, and grasping, require that your robot application be tested. Gazebo was created and updated in response to the demand for a multi-robot simulator. (Mazzari., 2015)

The difference between the two can be summed up in the following excerpt from Morgan Quigley (one of the original developers of ROS) in his book *Programming Robots with ROS*: “**rviz** shows you what the robot *thinks* is happening, while **Gazebo** shows you what is *really* happening.” (Sears-Collins, 2020).

We are now ready to let our tank travel inside the gazebo environment using the turtlebot3 teleop package after visualizing our tank and ensuring that everything is working properly. To operate the mobile robot remotely, we must run many ROS nodes: ROS master node is the initial node. The second node is teleop twist keyboard, which is a normal ROS node. This node is continually monitoring which keys are pressed on a PC keyboard and publishing twist messages on the /cmd vel topic according on the keys pressed. The Twist message specifies the linear and rotational speeds of a mobile robot. That brings us to the finish of our work on the first stage of our project, which is to have our tank travel inside any area using the keyboard from our computer from safe distance.

### **2.4.2 Robotic Arm**

To allow our robot to be more flexible when disposing a bomb, we added a 6DOF robotic arm to our tank. To operate it, we first worked on mimicking our arm. We utilize the same processes as in tank, drawing our arm on SolidWorks and then using extension to convert from SolidWorks to urdf.

We chose MoveIt - a ROS package intended to drive a robot arm utilizing the latest in motion planning, robot manipulation, kinematics, control, and navigation – to drive the robot arm. MoveIt is definitely far more advanced than would be expected of what is essentially a toy robot arm. It should, however, give a useful learning platform that we can potentially apply to a more advanced robot in the future. MoveIt, like the rest of ROS, appears to be rather difficult, but we only aim to utilize it for kinematic calculations, motion planning, and robot control. One useful



feature of MoveIt is its graphical setup assistance. The assistant is a graphical user interface that may be used to configure any robot for usage with MoveIt. Its main purpose is to create a Semantic Robot Description Format (SRDF) file for your robot. It also creates the additional configuration files required by the MoveIt package. At this point, we have mastered simulation and are ready to begin implementation. (McElhinney, 2019)

The first step in controlling the robotic arm is to inspect the wiring. The servo motors are linked to the PWM pins of the Arduino UNO. A 6v external power supply is connected to the servo motor's ground and VCC connections. Serial communication is used to link the Arduino UNO to the Raspberry Pi. The code is quite simple. The Arduino receives MoveIt's 'joint states' message and subscribes to it. This message specifies the angle status of each joint as the robot arm goes along the predetermined path to the target position.

### ***2.4.3 RGBD Kinect Camera***

In our project, the Kinect Camera is utilized to visualize the environment and to provide to the operator the ability to know the depth of the bomb-like object he wants to grasp. For that, we have to download all the packages and the dependencies of the Kinect camera to the raspberry pi, which are called “Freenect Packages” as will be discussed and detailed in chapter 5 of the report. After downloading all the packages and dependencies, and being able to launch and run the camera, we have to launch RVIZ and add the Depthcoudl Topic that will enable us visualize the environment live on RVIZ.

# CHAPTER 3

## DESIGN

In this chapter, we will discuss the design and calculations processes done to deliver a fully functional robot taking into consideration the theoretical specifications extracted from our meeting with the Lebanese Army. The design will enable us to choose all the appropriate components, from motors, to batteries to the drivetrain, and finally the robotic arm to accommodate our predefined specifications.

### 3.1 Chosen Specifications

The specifications that we hypothetically chose for our robot were the result of a thorough research made about bomb disposal robots that are in the market and in service as the ones mentioned earlier, and were immensely affected by our meeting with the Lebanese Army. We wanted to put together the best specs and features available to achieve the best build in the lowest price attainable to be competitive in the market. To do that, for those specifications, we didn't take into consideration some constraints that we might face including: allocated budget (funding), time, component availability, etc... However, we established a set of constraints like: the bomb has to be on the ground, and the user should have a laptop so that all the required packages would be supplied by us. All hypothetical specifications chosen are indicated in table 4 below.

<b>Specifications</b>	
Mission time (Hours)	3-4+
Width*Height*Length (cm)	68*20*44.5
Weight (Kg)	10
Ground clearance (cm)	7
Gripper lift capacity (kg)	0.3-0.5
Stair-climbing angle	35°-45°
Speed (km/h)	2-4
Terrain type	All type of terrains
Resistance capabilities	Water Resistant IPX4 standard
Range of covered area (m <sup>2</sup> )	50
Command and Control Unit CCU	Portable Computer
Robotic Arm	6 D.O.F

Object detection	Via RGBD Stereo Camera
Cost	To be Determined

Table 4: Theoretical Specifications Table

### 3.2 Design

#### 3.2.1 Robotic Arm Selection

The selected robotic arm is the: Meca500 Six-Axis Industrial Robot Arm:

When fully extended, the reach of this small industrial robotic arm is about 330 mm. It weighs 4.5 kg, reaches speeds up to 7 km/h, carries a maximum payload of 1 kg and is fully waterproof. It can be mounted in any orientation allowing it to move freely in all directions among the tank. The chosen robotic arm can be seen in figure 4, and its front and side reach can be seen on Rviz in figure 5 below.



Figure 4: Selected Robotic Arm

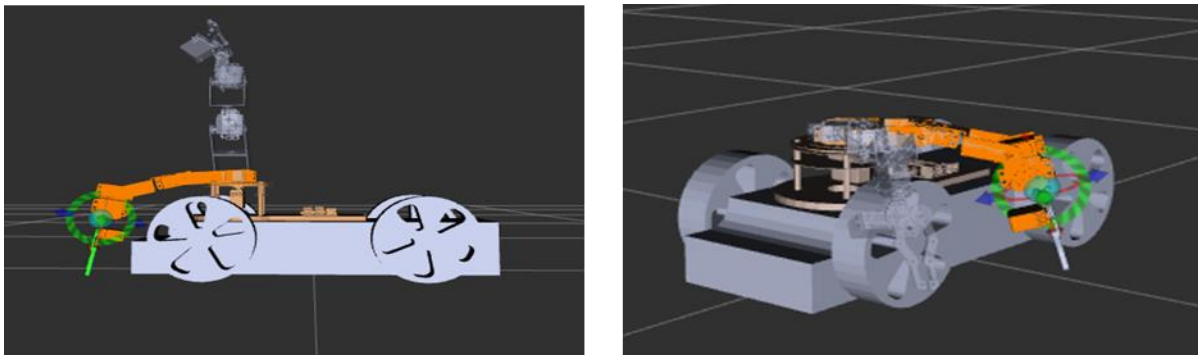


Figure 5: Front and Side Reach

### 3.2.2 Torque Sizing of Our Arm Servo Motors

In this part, we will demonstrate the formulas used to calculate the required torque of the servo motors of our robotic arm. Figure 6 below illustrates all the links and joints of our arm.

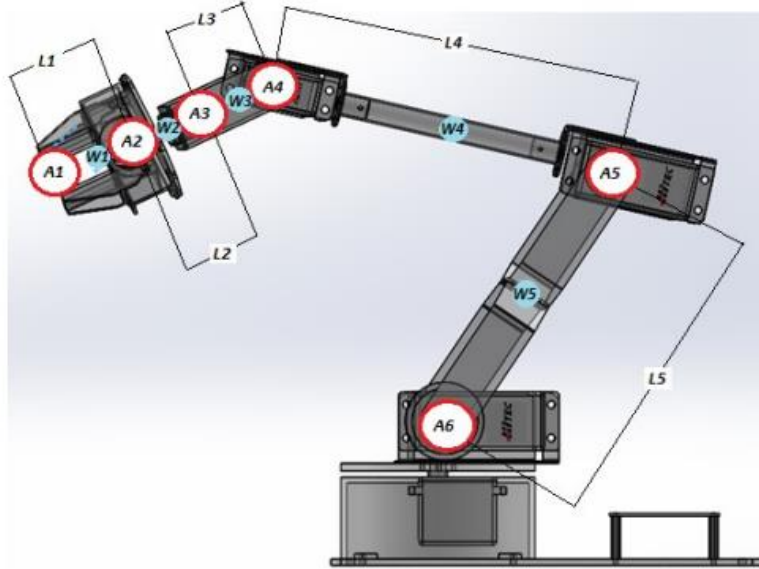


Figure 6: Robotic Arm Links and Joints

The formulas used to calculate the torque are the following:

$$T6 = (L1 + L2 + L3 + L4 + L5) * A1 + (0.5 * L1 + L2 + L3 + L4 + L5) * W1 + (L2 + L3 + L4 + L5) * A2 + (0.5 * L2 + L3 + L4 + L5) * W2 + (L3 + L4 + L5) * A3 + (0.5 * L3 + L4 + L5) * W3 + (L4 + L5) * A4 + (0.5 * L4 + L5) * W4 + (L5) * A5 + (0.5 * L5) * W5$$

Equation 1

$$T5 = (L1 + L2 + L3 + L4) * A1 + (0.5 * L1 + L2 + L3 + L4) * W1 + (L2 + L3 + L4) * A2 + (0.5 * L2 + L3 + L4) * W2 + (L3 + L4) * A3 + (0.5 * L3 + L4) * W3 + (L4 + L5) * A4 + (0.5 * L4 + L5) * W4$$

Equation 2

$$T4 = (L1 + L2 + L3) * A1 + (0.5 * L1 + L2 + L3) * W1 + (L2 + L3) * A2 + (0.5 * L2 + L3) * W2 + (L3) * A3 + (0.5 * L3) * W3$$

Equation 3

$$T3 = (L1 + L2) * A1 + (0.5 * L1 + L2) * W1 + (L2) * A2 + (0.5 * L2) * W2$$

Equation 4

$$T2 = (L1) * A1 + (0.5 * L1) * W1$$

Equation 5

As a demonstration, for a payload of  $A1 = 300$  grams and a maximum height = 400 mm, the following torque values are obtained:

- $T1 = 0.86$  kg/cm
- $T2 = 1.887$  kg/cm
- $T3 = 3.04$  kg/cm
- $T4 = 13.27$  kg/cm
- $T5 = 24.79$  kg/cm
- $T6 = 39.1$  kg/cm

In order of being able to carry a maximum load of 300 grams, first we must select the perfect servomotors for the job.

The motors selected to satisfy our requirements are: 1x Corona BL1029HV Servomotor, 2x DS3235 Servos along with 3x Towerpro MG996R. The use of different servos in this case allows us to achieve the suitable stability of our robotic by appropriate motors for the torque needed in order to achieve the task successfully.

A Towerpro MG996R servo operates at 4.8v-6v; it has a maximum rotation of 160 degrees which is an ideal rotation for the movement of the gripper. These servos have a torque equal to 11kg.cm and will be placed at the gripper and the wrists. This servo motor is shown in figure 7 below.



Figure 7: Towerpro MG996R Servo Motor

A DS3235 servo operates with a maximum rotation of 270 degrees in each direction which is an ideal rotation for the arm allowing it full rotation. They come with a torque of 35.2 kg.cm and will be placed on the two servomotors directly above of the base servomotor. This servo motor is shown in figure 8 below.



Figure 8: DS3235 Servo Motor

A CORONA BL1029HV servo operates with a maximum rotation of 180 degrees in each direction which is an ideal rotation for the base. They come with a torque of 50kg.cm and it will be placed on the base of the arm. This servo motor is shown in figure 9 below.



Figure 9: Corona BL1029HV Servo Motor

The robotic arm's motors, links and joints locations are illustrated in figure 10 below.

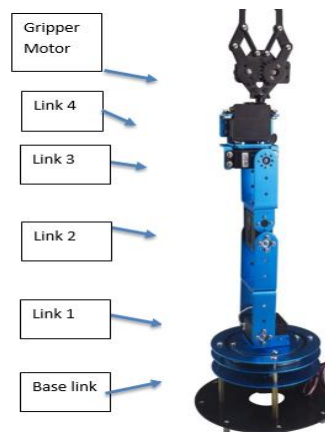


Figure 10: Motor Placement of Our Arm

### 3.2.3 Tank Selection

The selected tank is the TR500 Heavy-Duty Tracked Tank Chassis and it has the following specifications:

- Dimensions: 680\*445\*200mm.
- Tank Net Weight: 10 Kg.
- Running Speed: 3.6 km/h.
- Maximum Climbing Angle: 40°.
- Maximum Load: 50 Kg.

The selected tank is illustrated in figure 11 below.



Figure 11: Selected Tank Chassis

### 3.2.4 Torque Sizing for the Tank Motors

Assuming the system's total weight according to the predetermined specifications = 10 kg and a maximum payload of 300 grams, we now have a total weight of 10.3 kg. So the motors should be able to move at least 10.3 Kg.

The formula used to calculate the torque needed is:

$$T = \frac{m}{4} * g * r * F$$

Equation 6

With:

- $m$  = Mass of the robot = 10.3Kg. We are dividing the mass by 4 because we are considering it equally distributed over the 4 wheels.

- $g = 9.81 \text{ m/s}^2$ .
- $r = \text{radius of the wheel} = 0.08\text{m}$  (as our real robot)
- $F = \text{Coefficient of friction between wheels and dry asphalt} = 0.7$

$$T = \frac{10.3}{4} * 9.81 * 0.08 * 0.7 = 1.414602 \text{ N.m}$$

Equation 7

We know that  $1\text{N.m} = 10.19716 \text{ kg.cm}$ , so  $T = 14.42492 \text{ Kg.cm}$ .

So we need to find a tank that has 4 motors with a minimum torque of respectively  $14.42492/4 = 3.606 \text{ kg.cm} \Rightarrow 4\text{kg.cm}$  (safety factor taken into consideration) to move our robot with all its components easily and at full speed.

### 3.2.5 Gripper Selection

The selected gripper is the MEGP 25LS Electric Parallel Gripper and is shown in figure 12 below.



Figure 12: Gripper Selected

The Gripper has the following Specifications:

- Stroke per jaw: 24 mm.
- Stroke for all gripper:  $24 + 24 = 48 \text{ mm}$ .
- Maximum Gripping Force: 40 N.
- Gripper Weight: 0.136 Kg.

So the arm will be able to hold:  $1\text{kg}$  (Max payload) –  $0.136$  (Gripper weight) =  $0.864 \text{ Kg}$ .



### ***3.2.6 Waterproofing Our Robot***

To waterproof our system according to the IPX4 Standard, we must protect our electric components; this is achievable via epoxy resin.

Epoxy resins enjoy wide popularity among electronics industries due to their excellent electrical and mechanical properties; Epoxy resins insulate circuit boards to protect them from harsh environments, such as moisture, corrosive agents, and chemicals.

To apply this protective measure, we buy any epoxy resin tube and apply it to our electric components which will allow it to create a solid barrier that makes it waterproof.

However, the design discussed in this chapter is purely theoretical and is only applicable in case we had funding. The hardware that we used don't accommodate the design in this chapter as further constraints were imposed by the hardware that we used. The hardware used, its constraints and the design verification of it will be discussed in the following chapter.

## CHAPTER 4

### HARDWARE USED

Bomb Disposal Robot as any mechatronics project is a combination of simple and complex mechanisms that function independently or as a system which allow this project to achieve its task. This project requires 3 major sets or mechanisms that allow the robot to run and operate effectively. Those 3 major sets are composed of many components or sub-parts, thus, choosing those components must be according to our specific needs. The 3 major sets of our robot are: the tank, the robotic arm and the camera. However, in our project, those components pose some specific constraints that we were not accounting for because the components are utilized from the university lab and we could not get the appropriate components to our predefined specifications, so we had to go with what we've got. The new constraints are the robotic arm, the tank chassis and that the robot will operate in doors, because our components are restricted to their own limited capabilities.

The utilized actuators that assure the navigation of the tank are 4 high torque DC motors with metal gears that come equipped with built-in encoders, however, we dismissed the use of the encoders mainly because the navigation of the tank isn't autonomous, so there's no actual use of them. Moreover, the actuators of the Lewansoul robotic arm are servo motors that we will discuss their features and specifications furthermore in this chapter.

Similarly, one sensor in particular is chosen considering its type, range, accuracy and precision. It is used to achieve object detection (bomb detection), this sensor is an RGBD Kinect Camera that we will discuss all of its characteristics later on in this chapter.

#### **4.1 TS-100 Tank Chassis**

The TS-100 tank chassis is the main base of our robot as it holds all the other components. The TS100 tank chassis is constructed of all-metal aluminum with a variety of holes reserved on the chassis. It is free to be equipped with development boards and sensors and is compatible with Arduino / Seeeduino. The kit is made of aluminum and you can assemble it using matching tools. The parts that come with the tank are driving and bearing wheels, 2 tracks that can be adjusted according to the length needed, screws and the DC motors. Once assembled, the development board and motor drive module can be used to control motor motion. (Seed Studio, 2022)

In our case, the development and motor drive used to control the motor motion were an Arduino Mega 2560 microcontroller board and an L298N motor driver. We used a 12V lithium based battery to provide power for all tank components.

As for the motors that are used to move the tank, it comes equipped with 4 high torque DC motors with metal gears. All the motors' specifications can be referred to, in appendix C. The tank chassis illustrated in figure 13 below.



Figure 13: Captured Photo of the TS-100 Tank Chassis

#### **4.2 Lewansoul Robotic Arm**

The UBOT robotic arm is the second major part of our robot. It is responsible of grabbing the bomb-like object using ROS manipulation. It is a programmable robotic arm based on Arduino. The all metal design strengthens the robot's body. It features digital servo with high torque and high-temperature resistant. You can control the robotic arm to grab the object from different directions using ROS as discussed in the tutorial in appendix A.

To reach 6 degrees of freedom, the arm should consist of 6 servo motors. Servo motors are used in the robotic arm to control it. Servo motors contains a closed loop control circuit that allows the user to specify the target angle the motor. The robotic arm consists of more than one type of servo motors. There are 4 different types of servos, and they are 6 servo motors which will let the arm have 6 DOF. The types of the servos are: LDX-335MG servo (1 motor), LDX-218MG servos (2 motors), LD-1501MG digital servo (1 motor) and LFD-06 servos (2 motors). All these different motors are used because every one of them is used to do a task different from the other one.

In the base of the robotic arm, high torque motor is required to be able to rotate the system without any problem. Therefore, LDX-335MG servo motor is chosen to be fixed in the base of the robotic arm. Second type of motors are LDX-218 MG servo motors, which are a bi-axial high torque motor. Two of these motors are used at the base of the link 1 and link 2 of the robotic arm. Third type of motors is the LDF-06 servo motor. This servo is a bi-axial servo motor with a lower torque as shown in the image below. Final type of servo motors is the servo motor mounted on the gripper. Each motor's specs can be referred to in appendix C in the report.

The aim is to control the robotic arm through Wi-Fi. For this reason, the microcontroller should have 6 PWM signals to control the 6 servo motors. In addition, the microcontroller should support Wi-Fi. To satisfy these requirements a combination of two microcontrollers is used. Arduino Uno to control the servo motors and a Raspberry Pi to support Wi-Fi and control of the motors angles through ROS and Rviz.

Finally, the arm can be seen in figure 14 below. Moreover, a wiring diagram done using TinkerCad is illustrated in figure 15 also.



Figure 14: Captured Photo of the Lewansoul Robotic Arm

As mentioned before, the wiring diagram is done using TinkerCad. The servo motors are connected to the Arduino Uno PWM pins. Base, link 1, link 2, link 3, link 4 and Gripper motors are connected to pins 2, 3, 4, 5, 7 and 7 respectively. A 6V DC-DC converter is connected to the ground and VCC pins of the servo motor.

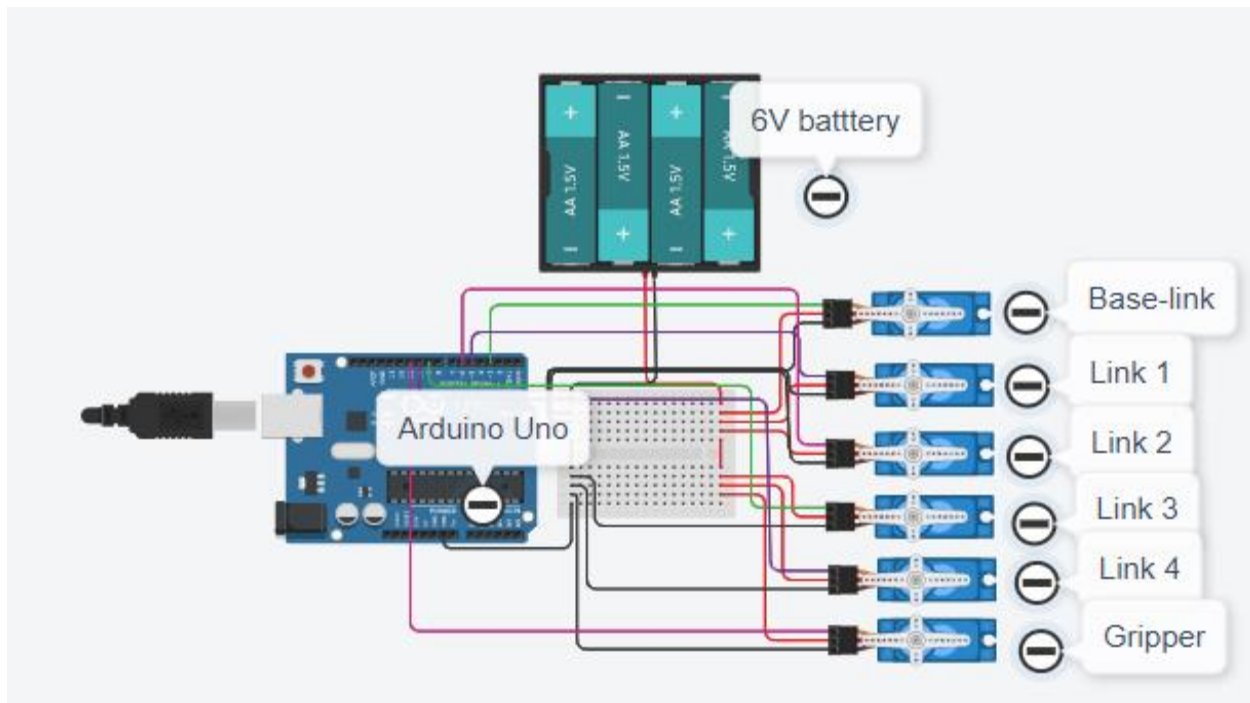


Figure 15: Robotic Arm Wiring Diagram

### 4.3 RGB-D Kinect Camera

The Kinect Camera is the only sensor that we are using in this project. Since couple of years RGB-D cameras have a huge impact on the research in the Computer Vision community as well as on related fields like Robotics and Image Processing. These cameras provide dense depth estimations together with color images at a high frame rate. It consists of one depth and one color camera. The depth image records in each pixel the distance from the camera to a seen object. The Kinect measures the depth with the Pattern Projection principle, where a known infrared pattern is projected into the scene and out of its distortion the depth is computed. (Wasenmuller & Stricker, 2018)

The camera in our robot is used only to visualize the environment. The only purpose of using it instead of a regular RGB camera is because we want to visualize properly the depth of the bomb-like object that we want to grasp.

### 4.4 Reverse Engineering Design

#### 4.4.1 Robot Battery Sizing

The power consumed by each component:

- Motor Drive: The motor drive is connected to the motors of the tank so the power consumption is the power measured at the H-Bridge. The current drawn below in this figure is the result of the 4 motors moving at their highest speed.

Input Voltage = 11.1V

Rated Measured Current = 0.25A

$$P = U * I = 11.1V * 0.25A = 2.75W$$

Equation 8



Figure 16: Measurement in University Lab for Motor Drive

- DC/DC Converter: It's connected to the servo motors of the robotic arm. The power measured on its terminals is the consumption of the combined motors of the robotic arm operating all together.

Input voltage = 6V

Rated Measured Current = 0.42A

$$\text{Power Consumed: } 6V * 0.42A = 2.52W$$

Equation 9



Figure 17: Measurement in University Lab for DC-DC Converter

- RGBD Kinect Camera: As per (Yoon, 2010), the power consumed by the Kinect Camera is 12W.
- Full Load Power Consumption:

$$P_t = 2.52 + 2.75 + 12 = 17.27W$$

Equation 10

- Depth of Discharge (DoD):

To calculate DoD, we multiply the maximum cell typical capacity of the battery with the maximum power consumption used of it. In our case we use only 70% of the battery, so:

$$DoD \text{ of } 70\% = 12.58 * 0.7 = 8.906Wh$$

Equation 11

The datasheet of the battery we are using is illustrated in the table below. (ENERDAN, 2001)

Type of Cell	Sealed Lithium-Ion Cylindrical Rechargeable battery
Cell Brand	Panasonic
Cell Size	18650
Cell Typical Capacity	3400 mAh (12.58Wh)
Cell Minimum Capacity	3250 mAh (12.02Wh)

Number of Cell Used	1PC
---------------------	-----

Table 5: Datasheet of Battery Used

- Operating Time of the Robot = 2.5 Hours.
- Final battery calculation to determine how many batteries we need for our robot to operate for 2.5 Hours:

$$\text{Total Consumption Over 2.5 Hours} = 17.27 * 2.5 = 43.175Wh$$

Equation 12

$$\text{Number of Batteries} = \frac{43.175}{8.906} = 4.847 = 5 \text{ Batteries}$$

Equation 13

#### 4.4.2 Power Bank Battery Sizing

Power Bank Specifications:

- Capacity = 10000 mAh
- Input = Output = 5V & 2.1A
- The power bank operating time will depend on how much current is being drawn from it.

Power Consumed by the components:

- Raspberry Pi:  $P = 5 * 2.5 = 12W$  (Pi, 2018)

**Input power:** 5 V/2.5 A DC via micro USB connector  
5 V DC via GPIO header

Figure 18: Raspberry Pi Input Power

- Arduino Mega:  $P = 5 * 0.73 = 3.65W$   
As per (DiyIoT, 2020), the maximum current consumption of the Arduino Mega 2560 is 0.73A, so we are taking this maximum value as a reference, although the actual current consumption might be lower than this.
- Current Consumption:  $It = 2.5 + 0.73 = 3.23A$
- For 3 hours, the power consumption is  $3.23 * 3 = 9.69Ah$ .
- The power Bank can last up to 3 hours easily since its capacity is 10Ah.



### 4.4.3 Torque Sizing

For the torque sizing we needed to know the maximum weight or load of the whole robot that we have in order to calculate the torque needed by the tank motors to move the robot. So instead of getting the weight of each component from non-reliable references, we put the actual robot on a scale to do it experimentally with all of its components attached to it. The weight of the robot came to be 4.7 Kg as shown in figure 19 below.



Figure 19: Robot Weight Scaling

- So our motors must move 4.7 Kg at least.
- Formula for the torque needed:

$$T = \frac{m}{4} * g * r * F$$

Equation 14

With:

- $m$  = Mass of the robot = 4.7 Kg. We are dividing the mass by 4 because we are considering it equally distributed over the 4 wheels.
- $g = 9.81 \text{ m/s}^2$ .
- $r$  = radius of the wheel = 0.08m. The diameter of the wheel is shown in our Solidworks design in the figure below.

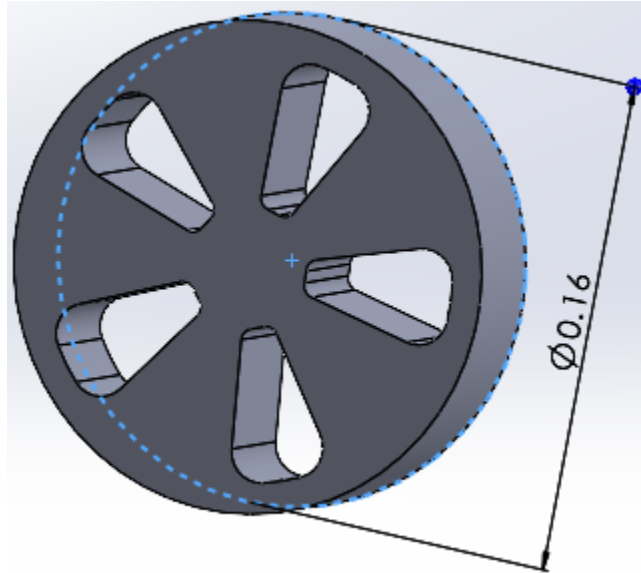


Figure 20: Wheel Design on SolidWorks

- $F =$  Coefficient of friction between wheels and dry asphalt  $= 0.7$

$$T = \frac{4.7}{4} * 9.81 * 0.08 * 0.7 = 0.645498 \text{ N.m}$$

Equation 15

We know that  $1\text{N.m} = 10.19716 \text{ kg.cm}$ , so  $T = 6.5822 \text{ Kg.cm}$

- As indicated in the datasheet of the tank motors in Appendix C-1, the torque of each motor is  $3\text{Kg.cm}$ , so the torque of the 4 motors combined is  $12\text{Kg.cm}$ .
- In conclusion, the motors that we have are enough to move the robot with all its components easily and at maximum speed.

#### 4.5 Bill of Material

Along with all the main hardware that represent the main shape of the robot, we used a lot of small components that are necessary to complete our prototype. Those components range from microcontrollers that are essential to establish communication and control the robot's motor, batteries as power source for the robot, along with a battery charger, a solderable breadboard and jumper wires. You can check all the components used in the project in the table 6 below that represents the Bill of Material.

Components	Available	To be Purchased	Quantity	Price (\$)
Ts-100 tank chassis	X		1	0
Lewansoul robotic arm	X		1	0
412 Raspberry Pi 3 A+	X		1	0

Arduino Mega 2560 (Made in China)		X	1	11
Arduino Uno	X		1	0
25 Li-ion 3.7V 18650 3400mAh Panasonic Battery		X	3	27
25 Lithium Battery Charger 4.2V Dual		X	1	7
86 CON 40 Male 2.54MM Mid Black Header Pins		X	2	0.25
36 Converter DC/DC PCB down 20W 40V to 35V		X	1	3
Laptop to be used as CCU	X		1	0
RGBD Kinect Camera	X		1	0
86*Jumper wires (M/M + M/F + F/F)		X	3 packets	3*1.35=4.05
L298N Motor Driver		X	1	2.75
Breadboard Solderable ss 7*5cm		X	1	0.25
<b>Total</b>				55.3

Table 6: Bill of Material

However, it's worth mentioning that the total cost in the Bill of Material above doesn't give a full representation of the cost of the full robot, it only specifies the cost of the components that we didn't have in our possession and that we had to buy as extra.

Taking into account the total price of the robot, including the Ts-100 tank chassis (68.38\$) (Store D. , 2022), the Lewansoul Robotic Arm (130.63\$) (Store W. , 2022) , the raspberry pi (33.8\$) (Katranji, 2022), and the Kinect camera (106\$) (Store H. , 2022), the total price of the robot will be equivalent to 394.11\$.

At the end of this section, we want to thank our colleague Imad Khodor, alumni at RHU, for he is the individual who donated the TS-100 tank chassis.

#### 4.6 General Wiring Diagram

In this section, we provide a wiring diagram (figure 21) that indicates how the different components of the Bill of Material are connected together. It must be noted that the detailed wiring and pinouts will be discussed in details later in chapter 5 of the report.

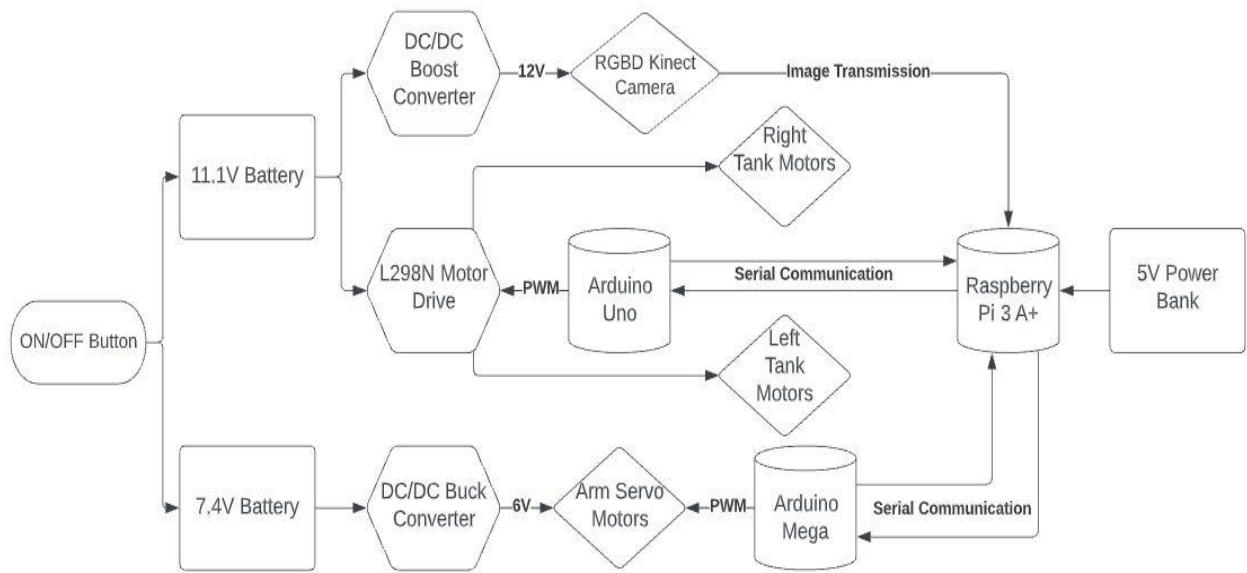


Figure 21: General Wiring Diagram

## CHAPTER 5

### IMPLEMENTATION AND RESULTS

In this chapter, we will discuss the implementation procedure done in order to get the full build of the robot. The simulation process will not be thoroughly covered in this chapter as it is detailed in Appendix A. The chapter will be divided into 4 sub-sections: the first will cover the implementation of the tank chassis, the second will talk about the robotic arm, the third sub-section will cover the RGB-D Kinect Camera implementation, and finally we will explain how we integrated two Arduinos with the raspberry pi via serial communication.

#### **5.1 Tank Chassis Implementation**

The tank simulation has to be done before starting to work on the real robot. All the simulation's steps and procedures are thoroughly discussed in Appendix A,i. Moreover, a video of the working Gazebo simulation can be accessed by using the link in this reference (Tank, 2022).

After finishing all the simulation testing, it was time to implement what we have done on ROS in the real world. The assembly of the tank chassis was already in place, so we were mainly responsible of all the wiring and components distribution on the tank. The tank is the base of all our robot, so it is holding all our different components, from the robotic arm, the camera, the power supplies and all the other microcontrollers and wires.

We first started by placing the robotic arm appropriately on top of the tank, in a way that it can reach its goal. Moreover, we placed the Kinect Camera at the front end of the tank in way that it can oversee the working environment. The arm was fixed on the tank using screws and bolts, whereas the camera was attached to it using wax.

After fixing the 2 major components on top of the tank, it was time to allocate a place for all the other components. The microcontrollers that operate the system are also located on top of the tank. The raspberry pi is waxed on the tank, the Arduino Mega was attached on top of the raspberry pi, and the Arduino Uno was waxed on top of the Kinect Camera.

All the other components were waxed and taped on the bottom of the tank in order to conceal the wiring. The L298N motor drive, 2 battery holders, a DC-DC converter, an ON-OFF switch and a breadboard were all fixed on the bottom of the tank. All the components that are fixed on the bottom are illustrated in figure 22 below.

As seen in figure 22, the motors are directly connected to the L298N motor drive. Each pair of motors are connected to one channel of the motor drive, the 2 front motors are connected to one channel of the motor drive, and the 2 back motors are connected to the other channel of the motor drive. A schematic indicating the connections of the motors with the motor drive is illustrated in figure 23 below.

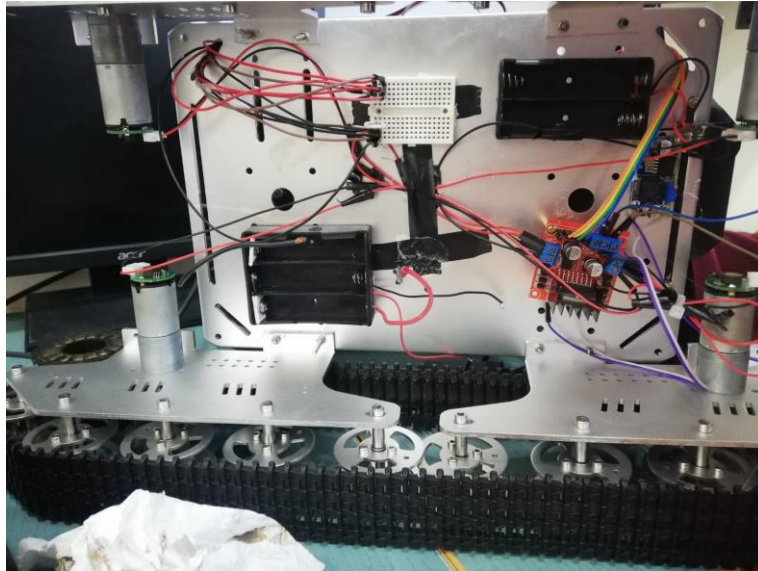


Figure 22: Bottom Components

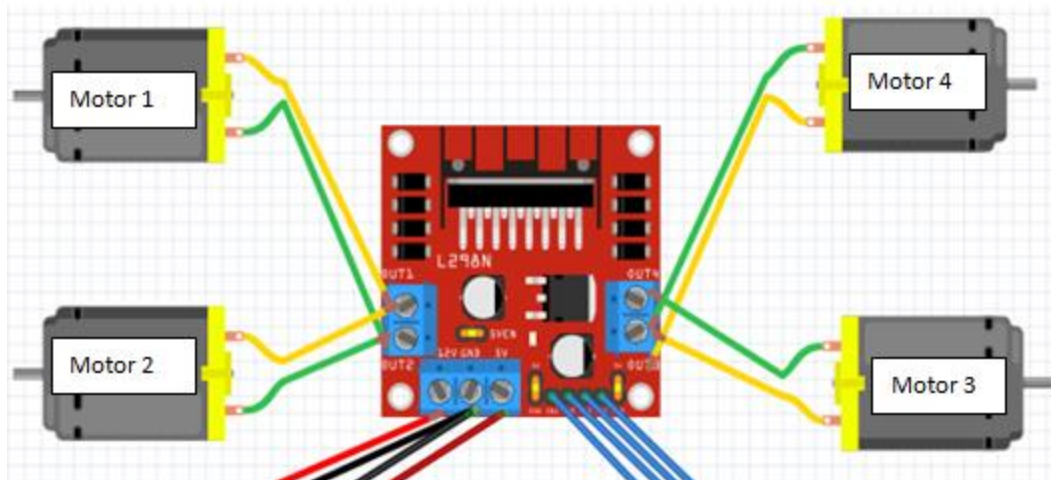


Figure 23: Schematic of the Tank Wiring

Finally, the 4 inputs of the motor drive, are connected to the PWM pins of the Arduino Uno that is powered by the raspberry pi via a USB cable (pins 6, 9, 10, 11 respectively). The L298N motor drive is powered by an 11.1V power supply, and a common ground wire is

established between the motor drive and the Arduino Uno to allow the current to flow back to the battery.

## 5.2 Robotic Arm Implementation

As mentioned in the previous sub-section, the robotic arm is attached on top of the tank's chassis in a way that it can reach its desired goal. After finalizing the Moveit package that is responsible of implementing the motion planning of the arm, and after concluding that it is functioning properly the real world environment simulation of gazebo as per Appendix A (ii), it was time to implement it on our robot. A video of the working Gazebo and Moveit simulation can be accessed via the link provided in the following reference (Arm, 2022).

First of all, the arm consists of 6 servo motors, with each servo motor having 3 wires, a VCC wire, a ground wire and a signal wire that will be connected to the Arduino Mega. The power wires (VCC and Ground) are arranged to go to the bottom of the tank and be connected to a breadboard that is powered by a regulated voltage of 6V. However, the signal wires go on top of the tank where the Arduino mega is located. The motors are numbered from 0 to 5 going from the base-link to the gripper, and their respective signal wires are connected to PWM pins 2, 3, 4, 5, 6 and 7 of the Arduino mega that is powered by the raspberry pi. It's worth noting that a common ground is established between the Arduino Mega and the DC-DC converter that is regulating the voltage coming from the 7.4V batteries. The arm links and motors pinout are illustrated in figure 24 below.

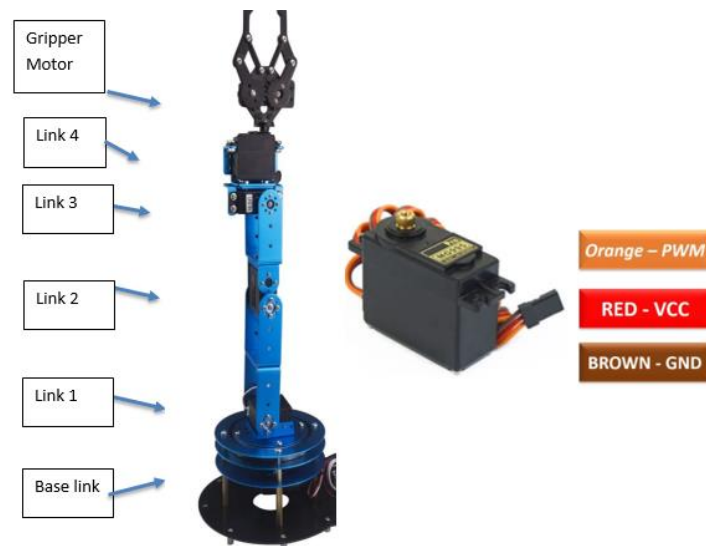


Figure 24: Arm Servo Motors

After finishing the wiring, it was time to implement what was done on simulation using the Moveit Package and Rviz. The package generated using Moveit is opened on Rviz in order to control the robotic arm in the real world. On simulation, the default starting angles of each servo motor going from the base-link to the motor that rotates the gripper (from joint 1 to joint 5 on Rviz) are as shown in the figure 25 below. However, in our real robotic arm, the motors are set in a way that they take a 90° angle when they are first powered. So there's an offset between the simulation and the real robotic arm that had to be compensated either in the code, or by disassembling the motors and rearranging them. We chose the code fixing approach. The first change was adding the offset angles to the motor degrees variables that are subscribed from ROS as shown in figure 26 below.

Group joints of goal state

Joint Name	Value
j1	0°
j2	0°
j3	86°
j4	0°
j5	0°

Figure 25: Default Joints on Rviz

```
mtrDegree0 = trimLimits(radiansToDegrees(cmd_msg.position[0]) + 90);
mtrDegree1 = trimLimits(radiansToDegrees(cmd_msg.position[1]) + 90);
mtrDegree2 = trimLimits(radiansToDegrees(cmd_msg.position[2]) + 4);
mtrDegree3 = trimLimits(radiansToDegrees(cmd_msg.position[3]) + 90);
mtrDegree4 = trimLimits(radiansToDegrees(cmd_msg.position[4]) + 90);
mtrDegree5 = trimLimits(radiansToDegrees(cmd_msg.position[5]) + 90);
```

Figure 26: Changes Applied to mtrDegree Variable

The second change was testing each motors rotation at its own on Rviz and seeing how the real robotic arm rotates, if it was rotating the same direction as Rviz no changes was made, but if it rotated the other way (in a mirrored way), the applied angle was subtracted from a 180° angle as shown in figure 27 below.



```

myservo0.write(mtrDegree0);
myservo1.write(mtrDegree1);
myservo2.write(180-mtrDegree2);
myservo3.write(mtrDegree3);
myservo4.write(mtrDegree4);
myservo5.write(mtrDegree5);

```

Figure 27: Changes Applied for Mirroring

### 5.3 RGB-D Kinect Camera

As said before, the Kinect Camera is used in our project to visualize the environment. For that, it was fixed on the front end of the robot facing the environment that the robot will move in. The Kinect camera requires a 12V power supply and to be connected via USB cable to the raspberry pi. Figure 28 below shows all the different sensors that are integrated inside the Kinect.

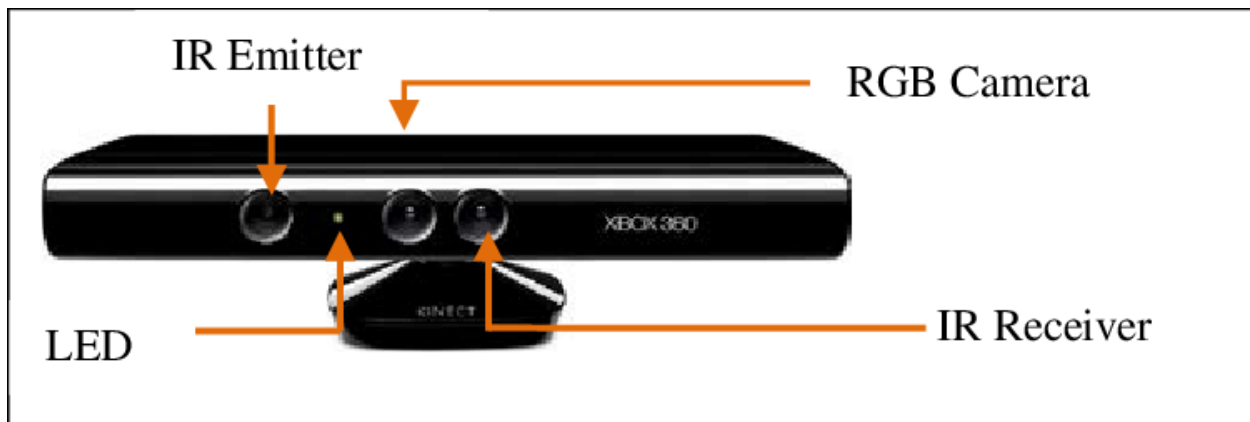


Figure 28: Different Sensors Inside the Kinect Camera

In this image as we can see, we have an RGB camera which is a standard camera, we have also an Infrared Emitter and Receiver that are used as a depth camera to get the depth in a way that we can get an image with depth information integrated with each pixel by creating a Depthcloud which is a 3D representation around the robot. Since we are using the version 1 of the Kinect, the minimum sensing depth distance is 40cm and the maximum is 4.5m.

As said before, the Kinect is connected to the pi using a USB2 cable, so just like any other component, we have to install its driver on the raspberry pi. The driver can be found easily on GitHub. The steps followed to install the driver and to make it running were taken from a website called “ProgrammerSought” (ProgrammerSought, 2022).

So first of all we have to clone the driver which is called “libreenect” using the following command: **git clone https://github.com/OpenKinect/libfreenect**. After downloading the driver, we have to access the driver’s directory and create a new folder named “**build**”, then we go into build and we compile the driver using this command: **cmake -L .. # -L**, then we type **make** to continue and finalize the compilation. We use now **sudo make install** to install the driver on Ubuntu. Now when we plugin the Kinect through USB2 cable, the Pi will identify it as a Kinect sensor and we can now interact with it.

For this next step, we need to install a package called freenect stack that can automatically read the data from the Kinect and publishes it to topics related to the Kinect. For that, we go to catkin\_ws/src on the raspberry pi, and we clone the package using the following command: **git clone https://github.com/ros-drivers/freenect\_stack.git**. After downloading it, we compile our workspace using **catkin\_make**.

Next, we installed the **rgbd-launch** package using **sudo apt-get install ros-melodic-rgbd-launch**. After installing it, we can now launch the camera using the command: **roslaunch freenect\_launch freenect.launch**, and we open Rviz.

After opening Rviz, we choose camera-link as the fixed frame, then we add a transformation so we can see the camera. After that, we go to the topics so we can see all the topics related to the camera. For example, we added an RGB image color topic in order to test if the camera is working properly, and the photo captured from the camera can be seen in figure 29 below.

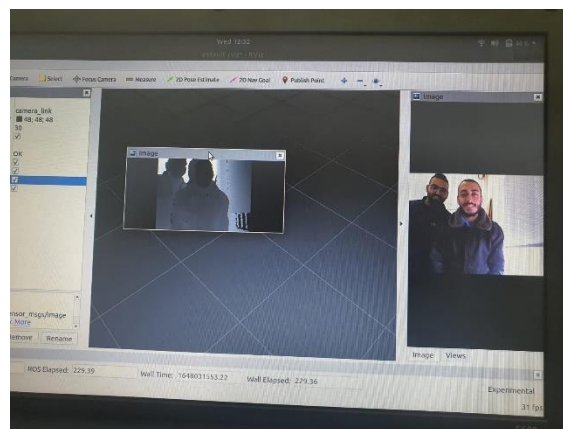


Figure 29: Photo Captured from the Camera

However, as described above, this installation and implementation process was done on our Linux laptop, but problems directly struck when we wanted to implement the same thing on the raspberry pi. After we tried the same process, we couldn't compile the workspace that we were working on as many unexplainable errors showed up in the terminal.

After conducting some research, we discovered that the raspberry pi 3 that we were using wasn't optimized to use and read the RGB-D camera libraries. We tried many installation techniques of the driver and the required libraries but all of them failed.

Finally, and in order to get around this problem, and in order to avoid debugging the errors that showed up because it would take a lot of time, we installed the required packages and libraries on both the Linux laptop and the raspberry pi, and to launch the camera, we launch the `freenect_launch` package on the raspberry pi using this command: **`roslaunch freenect_launch freenect.launch`**. Moreover, we launch the `rgbd_launch` package on the Linux Laptop using the command: **`roslaunch rgbd_launch Kinect_frames.launch`**.

The problem was that the raspberry pi wasn't able to launch the `rgbd_launch` package for some reason as it's not optimized to work with those packages. A video of the camera implementation on the robot can be accessed through this link:

<https://1drv.ms/v/s!ApBYgYC3eBSnjgZk8H9AKSioN7bi?e=LzuLKS>

## 5.4 Arduino Integration

Since our robot has two Arduinos controlling its motors (Arduino Uno for the tank motors, and Arduino Mega for the arm motors), it's essential that both Arduinos establish serial communication with the raspberry pi to ensure the remote operator control of the robot.

The problem in this is that you can only run the same `roserial` node once, and in our case you have to actually run it twice in two different terminals to be able to communicate with both Arduinos. When we tried to do this, the already running node shuts down by force as the second one starts running.

To solve this problem, we kept the arm servo motors running on the initial node by using the default ROS command: `roslaunch roserial_python serial_node.py`. And we created a launch file in the `roserial_python` package that launches the second node in another terminal without interfering with the first one and shutting it down. The content of the python launch file is shown in figure 30 below.

```

arduino.launch
1 <launch>
2 <remap from="/arduino_zero/cmd_vel" to="/cmd_vel" />
3 <node ns="arduino_zero" name="nero_arduino_zero" pkg="rosserial_python" type="serial_node.py" args="/dev/ttyACM0" output="screen" />
4 </launch>

```

Figure 30: Arduino Launch File Script

The script of this launch file is explained as follows:

- As per launch file syntax rules, the launch must start and end with a launch tag.
- The second line of script performs remapping of the topic. As discussed earlier, the topic that we are publishing to, to move the robot is /cmd\_vel. However, after creating the file and launching it for the first time, the topic that is available using rostopic list is /arduino\_zero/cmd\_vel. So we have to remap the new topic to the original topic in order to be able to move the tank motors.
- The third script line names the node that we created, indicates the package we are operating in, and finally gives as argument the port that we are using to physically connect the Arduino Uno to the Raspberry Pi.

Some problems were faced during the implementation:

- At first, we were not able to locate the rosserial\_python package using the graphical user interface. So we had to access and create the launch file inside of it using the terminal. The way to do that is by going to the appropriate directory and using the sudo nano filename command to create and modify the script of the file.
- After creating and saving the file, we were not able to launch it as we didn't have ownership permissions. To solve this, we had to change the file owner using the sudo chown user filename command, and then giving the file execution permissions using sudo chmod +rwx filename.

After solving those issues, we were able to successfully communicate with both Arduinos, by using the created launch file and the default rosserial node.

Finally, the robot full assembly with all the electronic component is illustrated in the figure 31 below. And a video of a full demo can be accessed via the following link:

<https://1drv.ms/u/s!ApBYgYC3eBSnjX-UnM5R7Zj76pKI?e=1MLfGP>



Figure 31: Assembled Robot

## CHAPTER 6

### CONCLUSION AND FUTURE WORK

Integrating all the system hardware and software aspects resulted in a fully functional prototype that proves the concept. The robot can navigate according to the operator's command using a keyboard, grasp and release a bomb-like object by its robotic arm gripper as it's being driven by the operator using the Moveit package. Moreover, the robot can visualize the environment using RGB-D Kinect Camera that provides a live feed in depth image of the environment on the operator's screen.

The following are some improvements that can be applied, in the future, to ensure a better performance in the second iteration.

#### **6.1 Future Work in Navigation**

- The robot can be developed to navigate autonomously, by using top-notch localization and orientation kits, while safeguarding the importance of the operator's intervention especially in tight and critical situations.
- Using GPS outdoor for robot localization and for pinpointing the locations of bombs and storing them in dataframes for later analysis.
- Integrating automated obstacle avoidance in the navigation system using appropriate sensors.
- Formulating an algorithm, such as waypoints, that enables the robot to navigate without taking commands from the user through Rviz.
- The tank chassis can be developed to climb stairs that can enable it to maneuver easily through different floors of a hostile building.
- The developed tank has to be water resistant according to the IPX4 water resistance standard as mentioned in our specifications.

#### **6.2 Future Work in Bomb Grasping**

- The available robotic arm poses a huge constraint on our work since the motors are not powerful enough to operate smoothly and grasp the demanded bomb weight. A way to work around this problem is to buy new motors for every joint or to buy a totally new robotic arm that meets our specifications.

- The robotic arm can be developed to operate autonomously without the operator's intervention using path planning and obstacle avoidance. This can be achieved by taking the coordinates of the bomb provided by the Kinect Camera and feeding them to the robotic arm through ROS so the arm can go to the appropriate location of the bomb at its own. The arm obstacle avoidance can be achieved by using an ultrasonic sensor that will detect the presence of an obstacle that interrupts the path of the arm. After detection, the arm will correct its path and move around the detected obstacle.
- The robotic arm has to be water resistant according to the IPX4 standard specified in our specifications.

### **6.3 Future Work in Vision and Bomb Detection**

- Our robot is equipped with an RGB-D Kinect Camera which make it vulnerable in outdoor situations. The robot can only detect depth indoors as the available camera is equipped with an infrared sensor that gets blind outdoors while being exposed to sun rays. We can make the robot operate outdoors by using a pricier and more sophisticated camera like the RGB-D Intel Camera.
- The robot can be equipped with multiple cameras that can feed various angles for the operator for better understanding of the environment. Our robot has only one camera that can see only in front of the robot, however, we can place 3 more cameras on the sides of the robot and on its rear in order to visualize all the environment.
- We highly suggest to use a NVIDIA Jetson Nano Developer Kit instead of the raspberry pi for image processing since it has more GPU Power which will enable the operator to have access to a smooth live feed.

The above mentioned improvements are essential to make the robot more robust and more competitive in the market. During our work on the robot, we weren't able to implement those advancements due to time, Covid-19, lockdown, budget constraints as the pressing financial and economic situation in Lebanon is growing and the components needed are either very expensive or hard to find as shortages in supply of those components struck the Lebanese markets. We advise whoever plans to work on the robot to start where we left off, refer to all the appendices in this report for further and detailed guidance and start working directly on improving the robot.

# APPENDIX A

## SIMULATION WORKSHOP

### i Robot Tank Simulation

The Robot Tank is the base and one of the key elements of our project. To do the simulation, the following steps were followed.

- We start by drawing our tank robot using Solidworks. The dimensions were measured in the lab on the actual tank and they came as shown in figure 32 below.

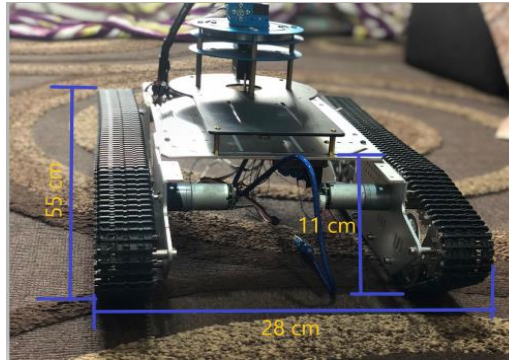


Figure 32: Tank Dimensions

- To simplify our work, we start by drawing the middle part as a box, then we locate the place of the motors on the box. The tank has chains as shown in the figure above, however for simplification purposes we draw the chains as wheels – two wheels on each side. Finally, after drawing the wheels and the body, we assemble our tank. The final assembly came as shown in figure 33 below.

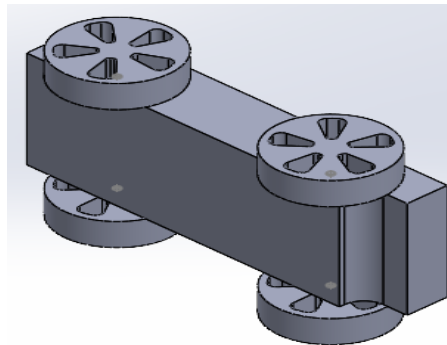


Figure 33: Robot Tank Assembly



- To export the robot tank model to URDF, we have to specify first the center of each wheel, the center of the body, the rotational axis, and the coordinate systems.
- To export to URDF, we have to download a Solidworks plugin extension called “Solidworks to URDF Exporter” from github.
- After downloading the extension, we launch it inside Solidworks. We specify the base-link and the 4 links that are connected to it – the wheels. Finally, we preview and export URDF and meshes and a URDF package is created that can be launched to visualize the model inside RVIZ and Gazebo.
- Now after exporting the package to the Linux environment, we create our workspace entitled tank\_ws. Inside this workspace we create a source file that we extract the package in. To create the workspace, we use the command **catkin\_make**.
- After creating it, we source it using **source devel/setup.bash**.
- We open the file the launches RVIZ using this command: **roslaunch Robot\_Tank\_Package display.launch**. First we set our fixed frame to base\_link then we add the Robot Model, the transforms and we see that we can rotate the 4 wheels.

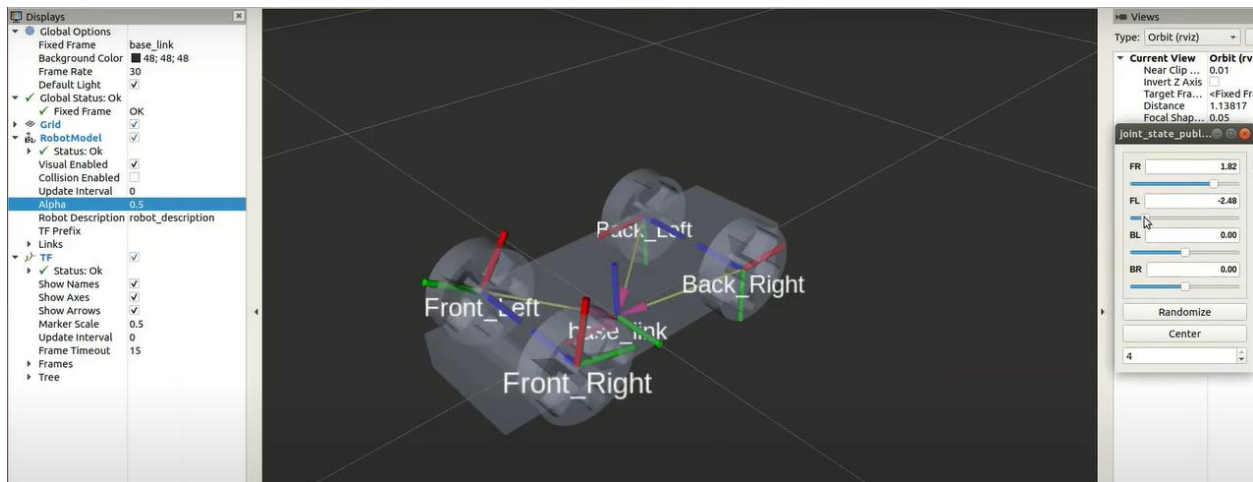


Figure 34: Tank Model on Rviz

- To visualize the robot inside gazebo, we use the command **roslaunch Robot\_Tank\_Package gazebo.launch**.

- The tank in Gazebo doesn't have a steering mechanism, so in order to control it, we have to add a gazebo plugin by simply searching gazebo plugin URDF on google and adding the code part of Skid Steering Drive to the URDF script and we update the example to match our variable names as shown in the below figure.

```

274 <gazebo>
275   <plugin name="skid_steer_drive_controller" filename="libgazebo_ros_skid_steer_drive.so">
276     <updateRate>100.0</updateRate>
277     <robotNamespace>/</robotNamespace>
278     <leftFrontJoint>FL</leftFrontJoint>
279     <rightFrontJoint>FR</rightFrontJoint>
280     <leftRearJoint>BL</leftRearJoint>
281     <rightRearJoint>BR</rightRearJoint>
282     <wheelSeparation>0.4</wheelSeparation>
283     <wheelDiameter>0.215</wheelDiameter>
284     <robotBaseFrame>base_link</robotBaseFrame>
285     <torque>20</torque>
286     <topicName>cmd_vel</topicName>
287     <broadcastTF>false</broadcastTF>
288   </plugin>
289 </gazebo>

```

Figure 35: Added Gazebo Plugin

- Now we reopen gazebo, and if we type **rostopic list** in a new terminal, we can see that now we have /cmd\_vel and /odom which are the command velocity and odometry of the robot.

```

/calibrated
/clock
/cmd_vel
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/set_link_state
/gazebo/set_model_state
/odom
/rosout
/rosout_agg
/tf

```

Figure 36: Topics Available

- To move the robot, we use the command **rostopic pub /cmd\_vel geometry\_msgs/twist "linear:** and then we change the linear x direction and we can see the robot moving forward. We can add rotation also by changing the angular z direction also.

- To see the robot tank moving on gazebo, refer to the URL in the references. (Tank, 2022)
- Finally, while the robot is moving, we can see the odometry by typing the following command: **rostopic echo /odom** as seen in the below figure.

```

z: 0.0
orientation:
  x: -0.000280543719601
  y: -0.00029412637385
  z: 0.0459522817111
  w: 0.998943553255
covariance: [0.0001, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0001, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 1000000000000.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1000000000000.0, 0.0,
0.0, 0.0, 0.0, 0.0, 1000000000000.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.01]
twist:
twist:
  linear:
    x: 0.483522419119
    y: -0.00785781854497
    z: 0.0
  angular:
    x: 0.0
    y: 0.0
    z: -0.196708089264
covariance: [0.0001, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0001, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 1000000000000.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1000000000000.0, 0.0,
0.0, 0.0, 0.0, 0.0, 1000000000000.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.01]
---
```

Figure 37: Odometry

- Now in order to move the tank using your keyboard, you have to download the turtlebot3\_teleop package.
- We used an Arduino code in order to control the motors and to introduce the /cmd\_vel topic. You can refer to the code used in Appendix B.
- We serially connect our Arduino with the raspberry pi using rosrn rosserial command.
- Finally, to move the robot, we use the following command:
- **Roslaunch turtlebot3\_teleop turtlebot3\_teleop\_key.launch.**

It's worth mentioning that the work done on the tank simulation was made more understandable thanks to the guided YouTube video of Mr. Wajih who was a student at RHU that worked previously on the tank. A link of the video is available in the report's references section. (Wajih, 2021)

## ii Lewansoul Robotic Arm Simulation

The Lewansoul arm is one of two main options that we can use as robotic arm. To do its simulation, the following steps were followed.

- We already had the simplified CAD files of all the parts of the robotic arm. We had to do the assembly of it. The final assembly came as follows.

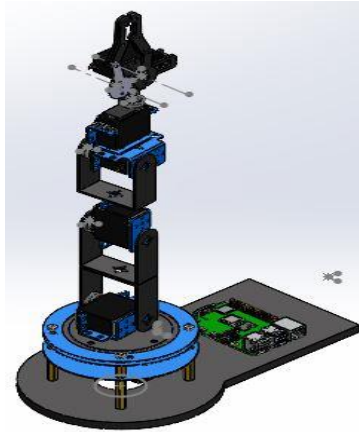


Figure 38: Assembly of the Arm

- After finalizing the assembly, we had to export the arm model as URDF by using the previously uploaded plugin. So we launch it inside Solidworks, and we specify all the links and their connections. We end up with 7 links and joints, 5 for the arm and 2 for the gripper. After previewing and exporting URDF and meshes, a URDF package is created called robot\_arm. This package contains two launch files that can be launched to visualize the arm on RVIZ and Gazebo respectively.
- Now after exporting the package to the Linux environment, we create our workspace entitled catkin\_ws. Inside this workspace we create a source file that we extract the package in. To create the workspace, we use the command **catkin\_make**.
- After having the URDF file ready, we need to control the robot either by simulation or using real hardware. One method of controlling the robot is using the Moveit commander.
- So first you need to launch the moveit setup assistant using the following command: **roslaunch moveit\_setup\_assistant setup\_assistant.launch**.

- After it is launched, press on the create new moveit button and browse your robot's URDF and open it, then press load files. After choosing your robot's URDF, the robot's module should appear on the right as shown in the figure below.

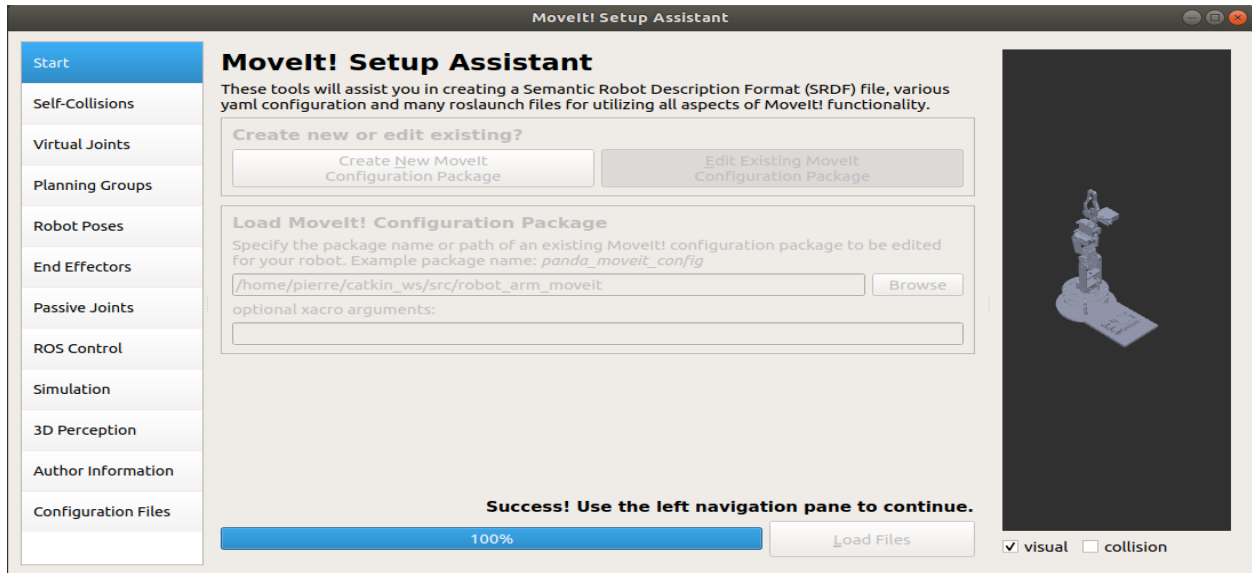


Figure 39: Arm Module Loaded on Moveit

- Choose the self-collisions option and press generate collision matrix.

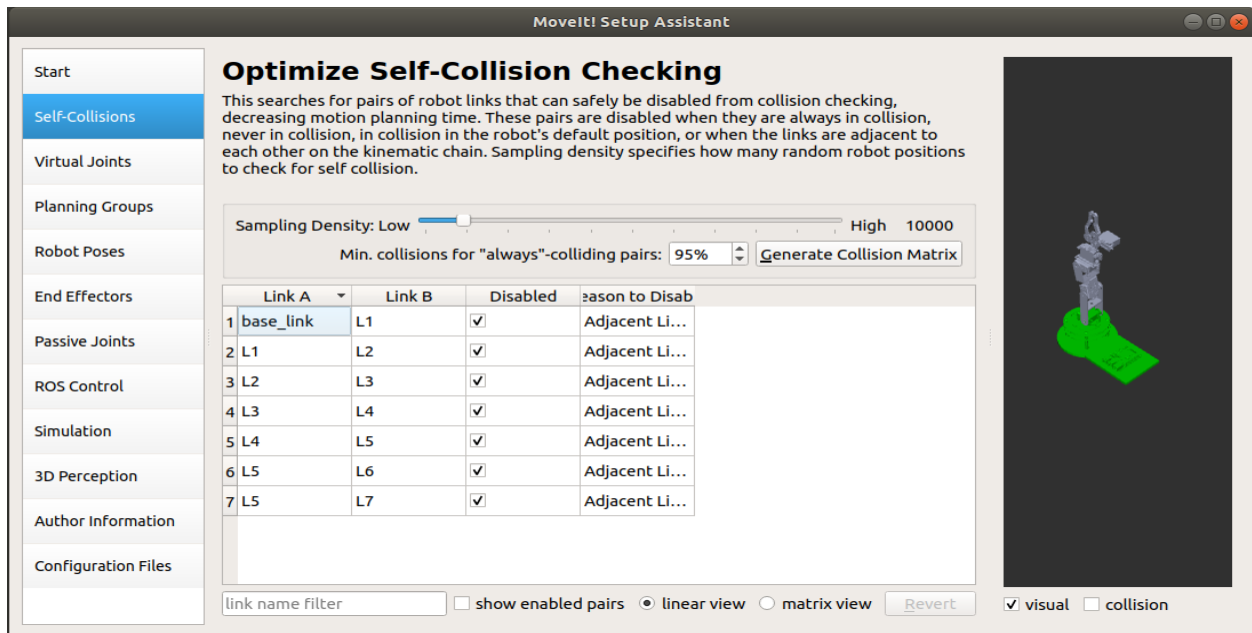


Figure 40: Self-Collision Checking

- In the virtual joints tab, you can create a virtual joint. This step is optional, but here we created a fixed joint to link the base\_link of the robot to the ground.

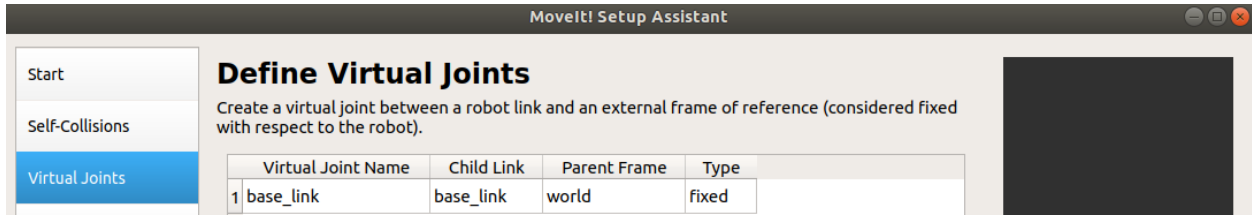


Figure 41: Virtual Joint Created

- In the planning groups tab, you should choose the planning groups that you wish to control. Press on the add group button. Here we have two groups, the arm and the gripper. For the arm group we have to customize some of the settings before adding the appropriate joints. However, for the gripper group we leave the settings as default. The settings that we're talking about are shown in the below figure.
- In the below window you should fill the following:
  - Group name: with the group name you choose.
  - Kinematic solver: for the arm choose kdl kinematics plugin.
  - Group default planner: choose RRT.
- Then finally press the add links button to choose the links to add to the planning group.



Figure 42: Planning Groups Settings Window

- After adding the links, we get two planning groups as follows.

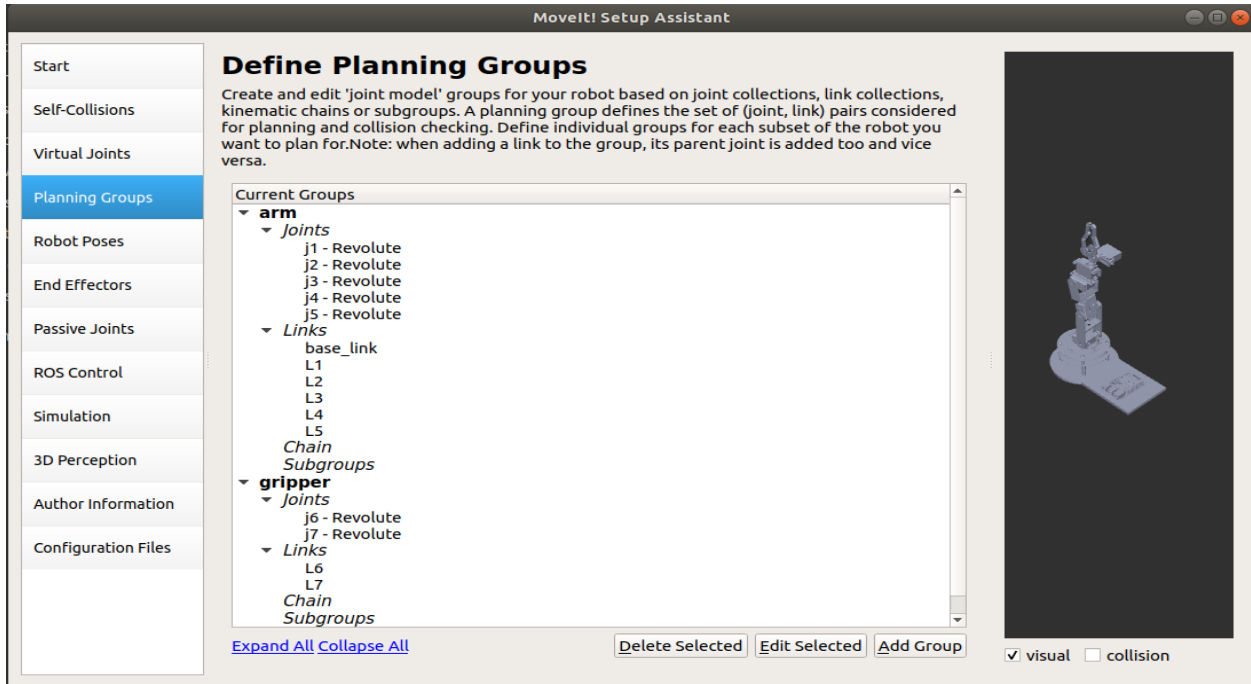


Figure 43: Chosen Planning Groups

- Next, select robot poses to define default poses for the robot, then modify the joints to reach your desired position and choose your desired planning group. An example of a chosen pose is illustrated in the figure below.

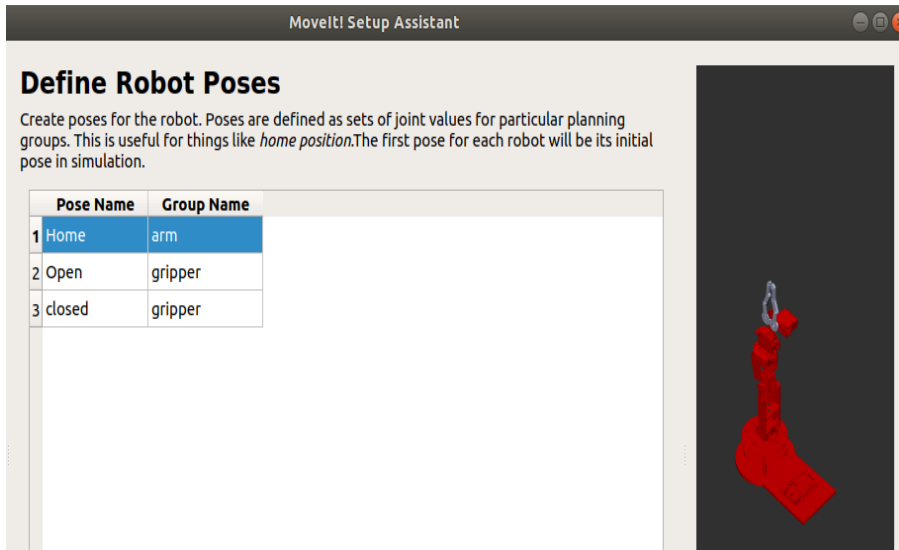


Figure 44: Robot Arm Pose

- To perform gripping we add the end effector option.



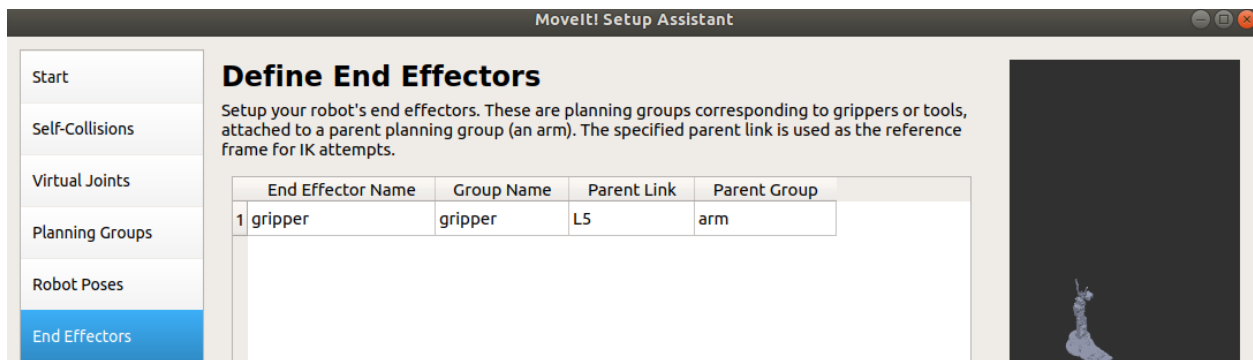


Figure 45: Define the End Effector

- Select ROS control option and press auto add followjointtrajectory. This will generate the yaml file to control the robotic arm.

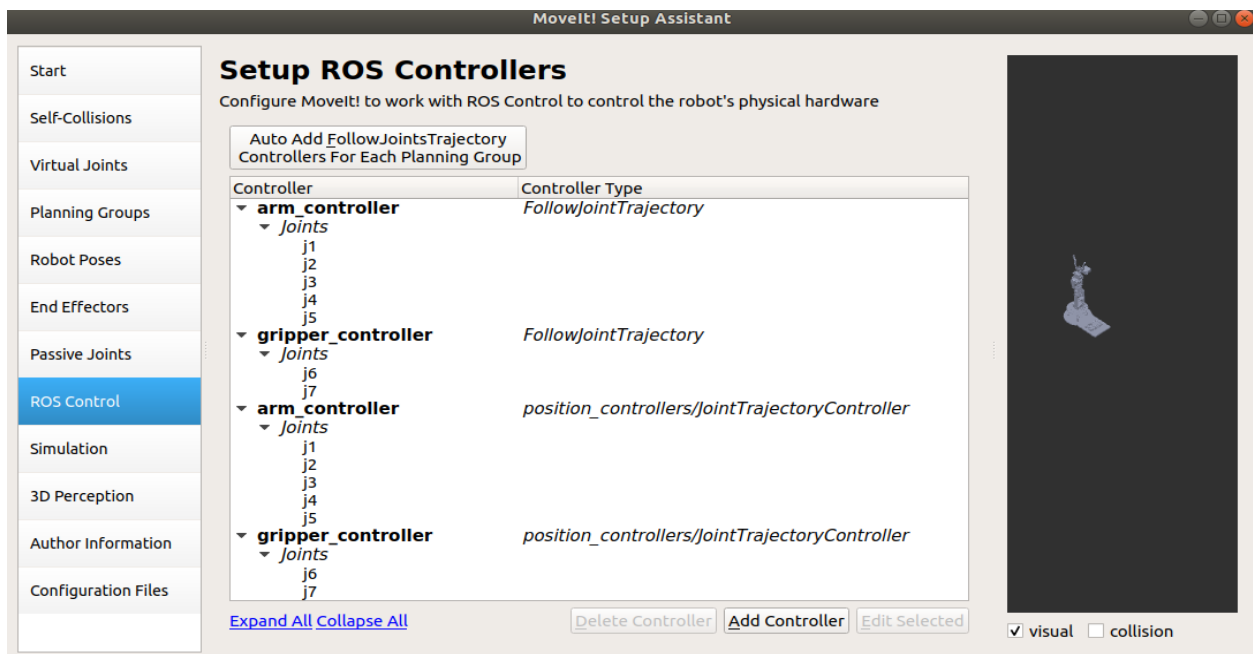


Figure 46: ROS Controllers Setup

- Select the author information and fill in the contact info.
- Finally, we choose the path where we want to install the package. We create a package named robot\_arm\_moveit and direct the package into it. Finally, we select the generate package option and we close the setup assistant when everything is done.



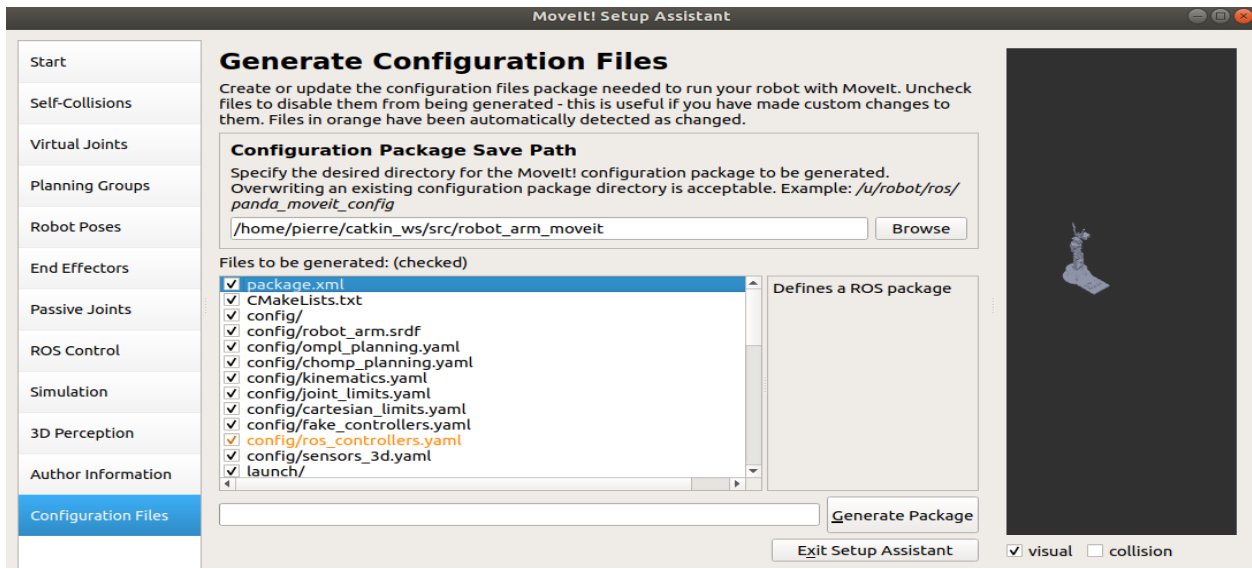


Figure 47: Generate Configuration Files

- Now you can test the package by visualizing the data on RViz. You can launch it by typing: **roslaunch robot\_arm\_moveit demo.launch**. RViz should open with the robot model inside it.
- Now you can start planning, press on approx. IK solution and you are now able to drag the robot to your desired position.
- After dragging it to your desired position, press on plan and execute to see the joint\_states changing and the robot model moving as shown in the below figure.

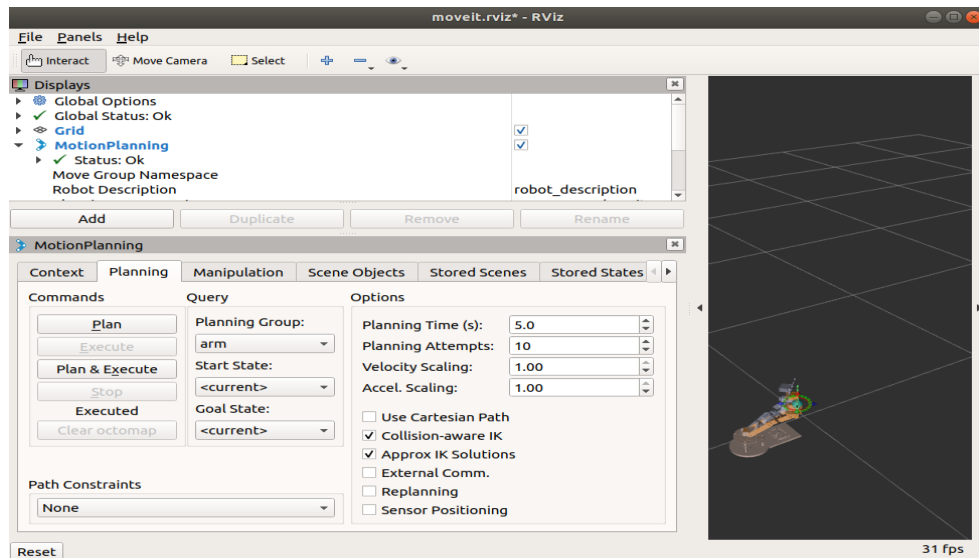


Figure 48: Arm Visualization on Rviz

- In RViz we can only visualize the data. What we actually need is to control a real or simulated robot. For simulation purposes we can use gazebo. In the moveit package you can launch a demo controlling the simulated robot using RViz by typing the following command: **roslaunch robot\_arm\_moveit demo\_gazebo.launch**
- However, when you launch it you will notice that the arm does not move in gazebo when it's moving using RViz. This is because you should configure the controllers manually by following the after mentioned steps.
- You should first add transmissions to the original URDF. Go to robot\_arm package, open the URDF file and add transmission tags in the end of the URDF for each joint. An example for a transmission tag for joint 1 is illustrated below.

```
<transmission name="trans_j1">
<type>transmission_interface/SimpleTransmission</type>
<joint name="j1">
<hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
</joint>
<actuator name="j1_motor">
<hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
<mechanicalReduction>1</mechanicalReduction>
</actuator>
</transmission>
```

Figure 49: Transmission Tag for J1

- After adding all transmission tags you should also add the gazebo plugin in the end of the URDF file as follows.

```
<gazebo>
<plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
<robotNamespace>/</robotNamespace>
</plugin>
</gazebo>
</robot>
```

Figure 50: Gazebo Plugin Added

- Now you need to configure the controller for the simulated robot. To do that go to the following directory: robot\_arm\_moveit/config and open ros\_controllers.yaml

file. Once you open it, add your controller's name joints and pid values for each planning group as follows.

```
gripper_controller:  
  type: position_controllers/JointTrajectoryController  
  joints:  
    - j6  
    - j7  
  gains:  
    j6:  
      p: 100  
      d: 1  
      i: 1  
      i_clamp: 1  
    j7:  
      p: 100  
      d: 1  
      i: 1  
      i_clamp: 1
```

Figure 51: Gripper Controller Added

```
arm_controller:  
  type: position_controllers/JointTrajectoryController  
  joints:  
    - j1  
    - j2  
    - j3  
    - j4  
    - j5  
  gains:  
    j1:  
      p: 100  
      d: 1  
      i: 1  
      i_clamp: 1  
    j2:  
      p: 100  
      d: 1  
      i: 1  
      i_clamp: 1  
    j3:  
      p: 100  
      d: 1  
      i: 1  
      i_clamp: 1  
    j4:  
      p: 100  
      d: 1  
      i: 1  
      i_clamp: 1  
    j5:  
      p: 100  
      d: 1  
      i: 1  
      i_clamp: 1
```

Figure 52: Arm Controller Added

- Finally, you should go to robot\_arm\_moveit/launch and open ros\_controllers.launch and add your controllers to the arguments.

```
<node name="controller_spawner" pkg="controller_manager" type="spawner" respawn="false"
  output="screen" args="arm_controller gripper_controller joint_state_controller"/>
```

Figure 53: Argument Added

- After configuring the controllers, you can use the previous command and control the simulated robot arm using moveit and RViz. A video of the simulation can be accessed through the link in the references. (Arm, 2022)

The process to achieve this simulation wasn't easy, because we faced a lot of errors while learning how to do it. Some of the common errors we encountered will be discussed below.

- Error 1: RLEException: unused args [execution\_type].

Cause: This error is mainly caused by a bug in the latest moveit update that happens while generating the package after finalizing the configuration of the robot.

Solution: This problem is solved by adding this unused argument in the robot\_arm\_moveit\_controller\_manager.launch.xml file that is located in the moveit generated package. The solution is shown in the following figure.

```
robot_arm_moveit_controller_manager.launch.xml X
home > pierre > catkin_ws > src > robot_arm_moveit > launch > robot_arm_moveit_controller_manager.launch.xml
1  | launch
2
3  | <!-- loads moveit_controller_manager on the parameter server which is taken as argument
4  | | if no argument is passed, moveit_simple_controller_manager will be set -->
5  | <arg name="moveit_controller_manager" default="moveit_simple_controller_manager/MoveItSimpleControllerManager" />
6  | <param name="moveit_controller_manager" value="$(arg moveit_controller_manager)"/>
7  | <arg name="execution_type" default="unused" />
8
9
10 | <!-- loads ros_controllers to the param server -->
11 | <rosparam file="$(find robot_arm_moveit)/config/ros_controllers.yaml"/>
12 | </launch>
```

Figure 54: Error 1 Solution

- Error 2: The UBOT appears upside down in gazebo. This design error is shown in the figure below.

Cause: This is caused by a design error while assembling the parts of the arm in Solidworks before exporting it into URDF.

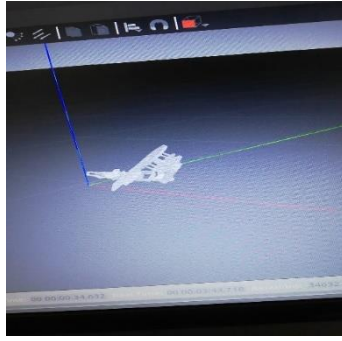


Figure 55: Robotic Arm Upside Down on Gazebo

Solution: This error actually has two different solutions. The first one can be applied while assembling the parts in Solidworks, it's described as follows: you have to do 2 different mates between the base\_link and the top plane. The first mate is to make them parallel; the second one is to make them concentric. This solution didn't actually work for us. So we tried a second solution which is to make the base\_link fixed with the world by identifying it in the URDF file at the beginning of the script. The added fixed link tag is shown in the figure below.

```
<link name="world" />
<joint name="world_to_base_link" type="fixed">
  <parent link="world" />
  <child link="base_link" />
</joint>
```

Figure 56: Added Fixed Link

- Error 3: Gazebo terminal error while launching any gazebo file: [Err] [REST.cc:205] Error in REST request.

Cause: Bug happening while downloading ROS initially on Linux.

Solution: go to home, show hidden files, go to .ignition/fuel/config.yaml. Change the already existing URL or comment it and add the following correct URL: “url: <https://api.ignitionrobotics.org>”.

Finally, we want to give credits to Mr. Hussein Harb who's guidance and references helped us in accomplishing successfully the robotic arm simulation.

Accordingly, his YouTube video detailing the steps followed to convert from Solidworks to URDF can be accessed by the link provided in the references at the end of the report. (Harb, 2021)

### iii Dobot Robotic Arm Simulation

The Dobot was the second robotic arm that we worked on, but we will not use it since it requires a main sector electric supply (AC/220V) and we are not planning to change the electronic components supplied by the arm. However, in order to do its simulation, we must proceed with the following steps.

- We have the simplified Solidworks CAD files of the robotic arm's components. We had to put it together. The final assembly looked as shown in the below figure.

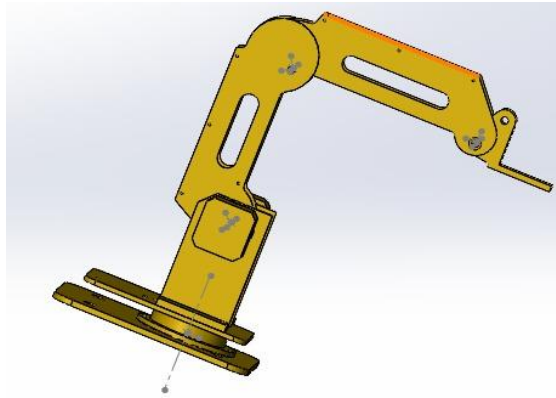


Figure 57: Dobot Assembly

- After finalizing the assembly, we had to export the arm model as URDF by using the previously uploaded plugin. So we launch it inside Solidworks, and we specify all the links and their connections. We end up with 6 links and 4 joints. After previewing and exporting URDF and meshes, a URDF package is created called `dobot_v1`. This package contains two launch files that can be launched to visualize the arm on RVIZ and Gazebo respectively.
- Now after exporting the package to the Linux environment, we create our workspace entitled `dobot_ws`. Inside this workspace we create a source file that we extract the package in. To create the workspace, we use the command **`catkin_make`**.
- After having the URDF file ready, we need to control the robot either by simulation or using real hardware. One method of controlling the robot is using the **Moveit** commander.
- So first you need to launch the moveit setup assistant using the following command: **`roslaunch moveit_setup_assistant setup_assistant.launch`**.

- After it is launched, press on the **create new moveit button** and browse your robot's URDF and open it, then press load files. After choosing your robot's URDF, the robot's module should appear on the right as shown in the figure below.

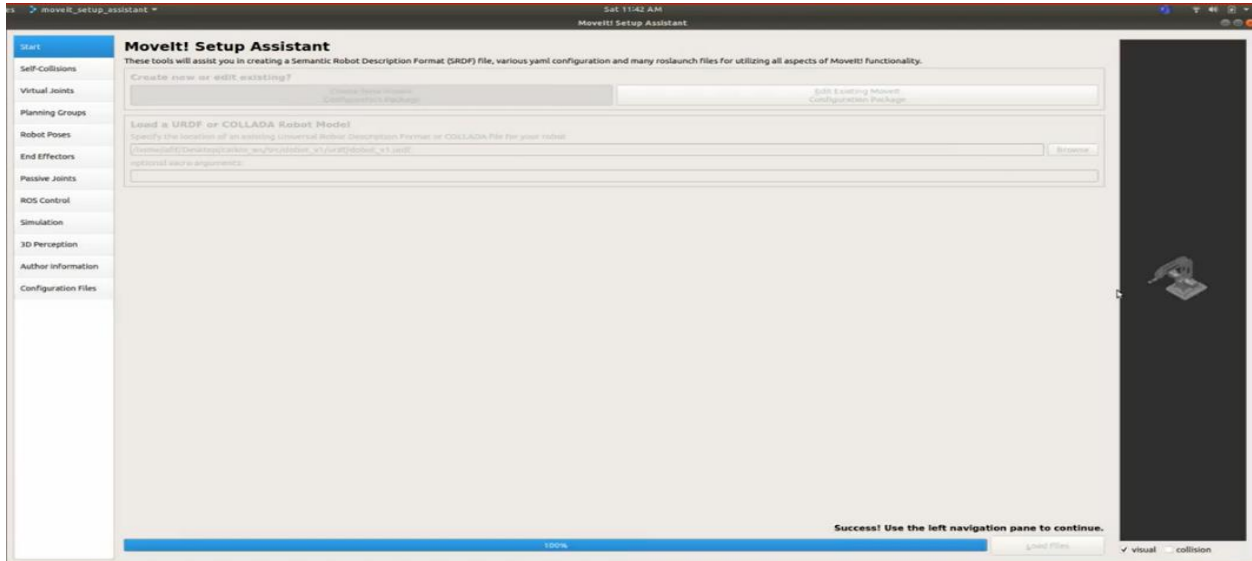


Figure 58: Dobot Arm Module Loaded in Moveit

- Choose the self-collisions option and press generate collision matrix.

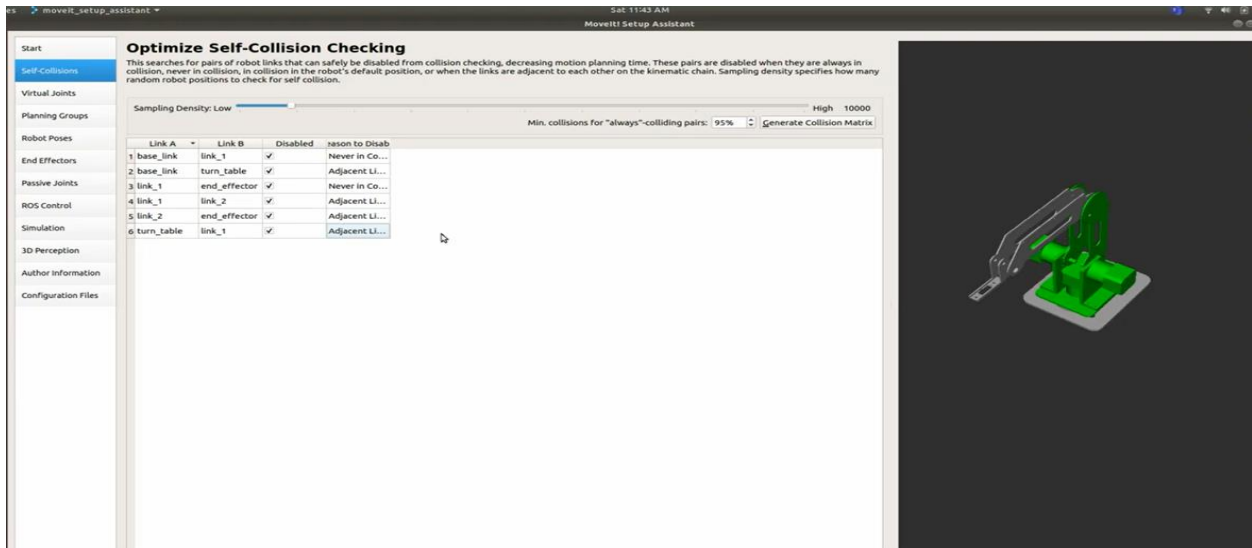


Figure 59: Self-Collision Matrix



- In the virtual joints tab, you can create a virtual joint. This step is optional, but here we created a fixed joint to link the base\_link of the robot to the ground like we did for the UBOT.
- In the planning groups tab, you should choose the planning groups that you wish to control. Press on the add group button. Here we have only one group, which is the arm. We have to customize some of the settings to the arm group before adding the appropriate joints. The settings that we're talking about are shown in the below figure.
- Then finally press the add links button to choose the links to add to the planning group.

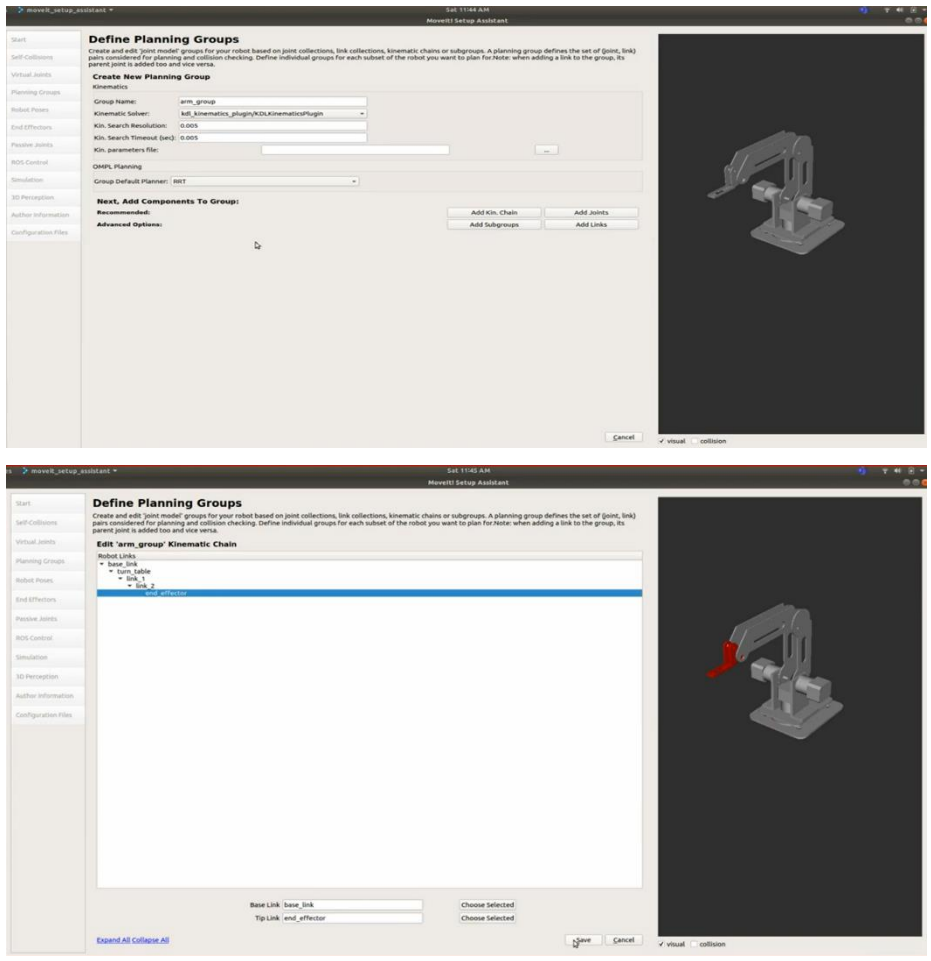


Figure 60: Choose Planning Groups Links

- Next, select robot poses to define default poses for the robot, then modify the joints to reach your desired position and choose your desired planning group.

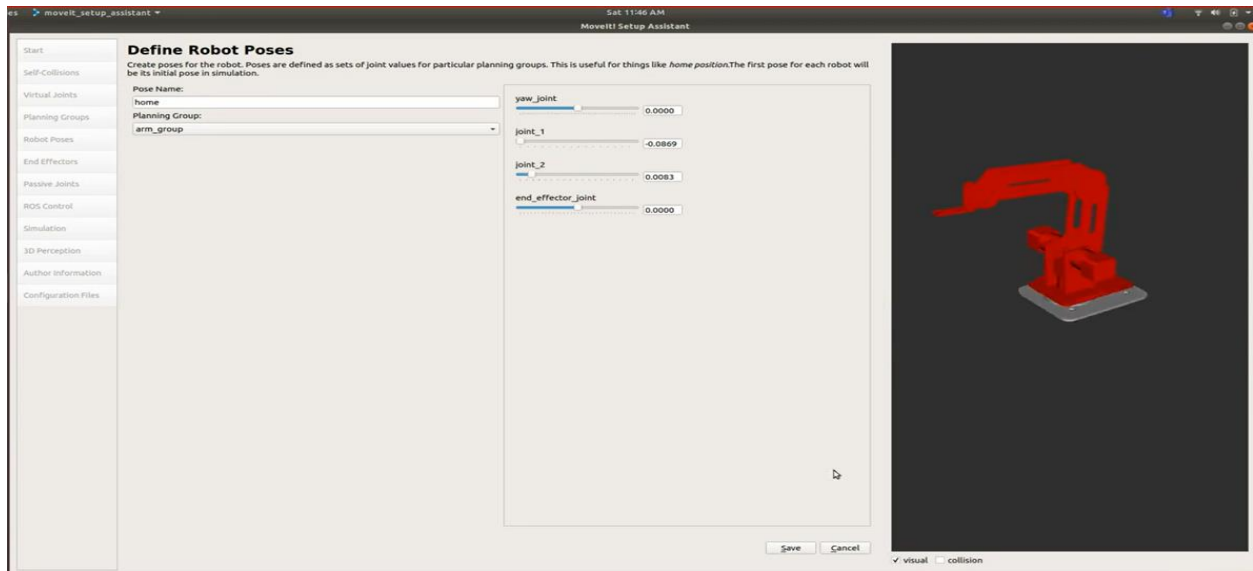


Figure 61: Home Pose

- Select ROS control option and press auto add followjointstrajjectory. This will generate the yaml file to control the robotic arm.
- Select the author information and fill in the contact info.
- Finally, we choose the path where we want to install the package. We create a package named dobot\_moveit\_config and direct the package into it. Finally, we select the generate package option and we close the setup assistant when everything is done.

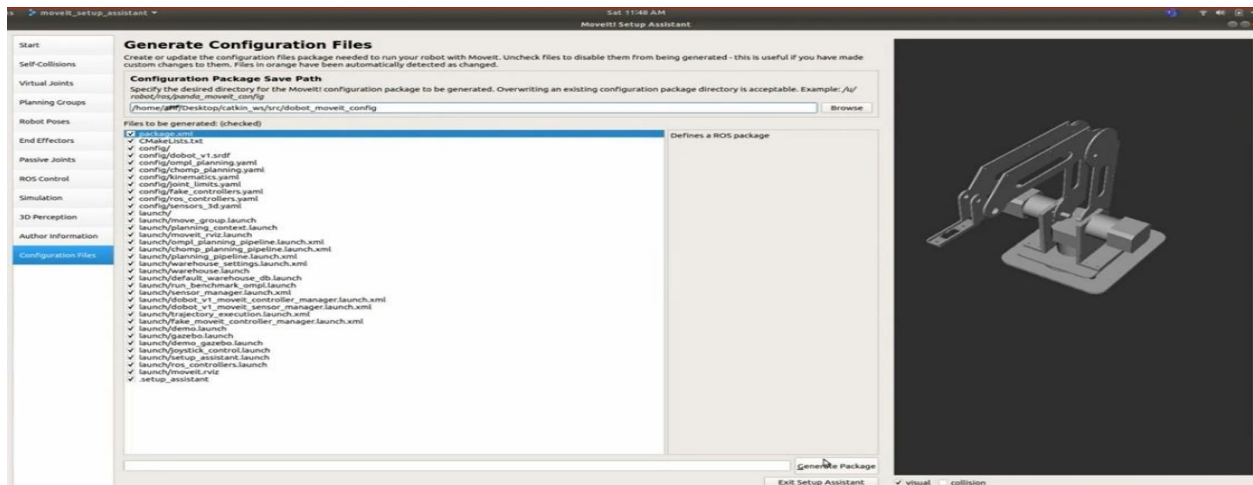


Figure 62: Generate Config Files

- Now you can test the package by visualizing the data on RViz. You can launch it by typing: **roslaunch demo\_gazebo.launch**. RViz should open with the robot model inside it.
- Now you can start planning, press on approx. ik solution and you are now able to drag the robot to your desired position.
- After dragging it to your desired position, press on plan and execute to see the joint\_states changing and the robot model moving as shown in the below figure.

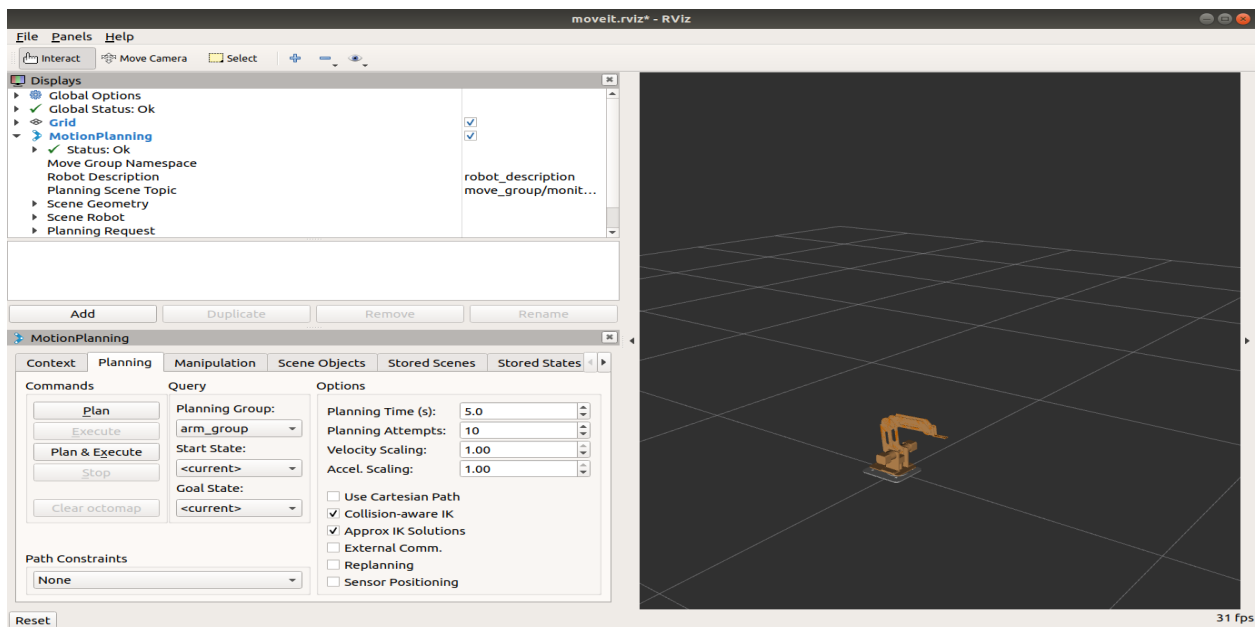


Figure 63: Rviz Simulation

- In RViz we can only visualize the data. What we actually need is to control a real or simulated robot. For simulation purposes we can use gazebo. In the moveit package you can launch a demo controlling the simulated robot using RViz by typing the following command: **roslaunch demo\_gazebo.launch**.
- However, when you launch it you will notice that the arm does not move in gazebo when it's moving using RViz. This is because you should configure the controllers manually by following the after mentioned steps. (and if you don't add them you will see that the arm is broken in gazebo)

- You should first add transmissions to the original URDF. Go to robot\_arm package, open the URDF file and add transmission tags in the end of the URDF for each joint.

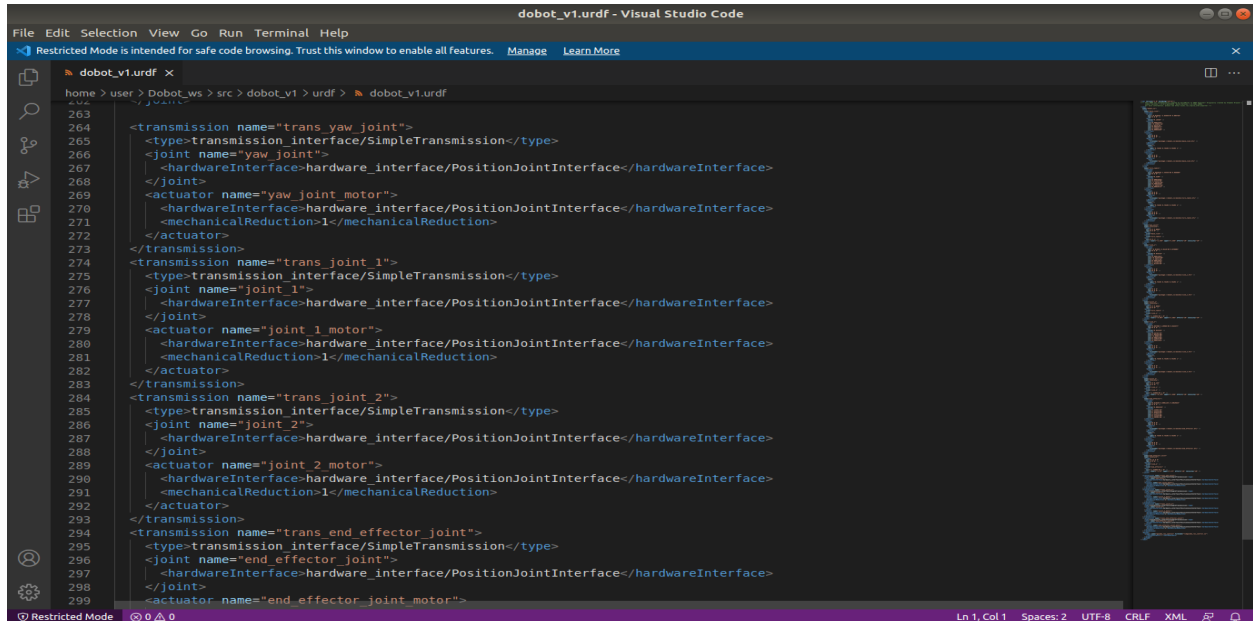


Figure 64: Transimmission Tags Added

- After adding all transmission tags you should also add the gazebo plugin in the end of the URDF file as follows.

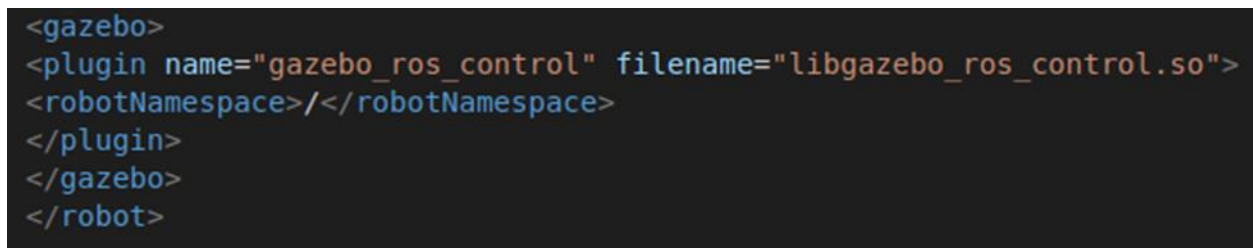
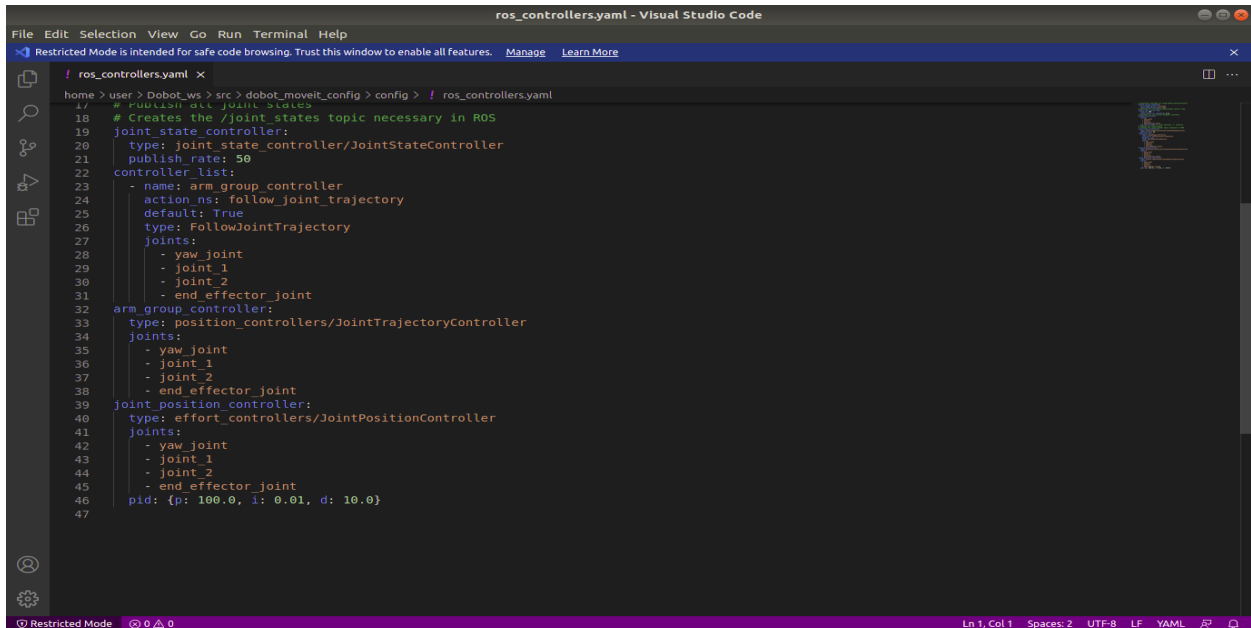


Figure 65: Gazebo Plugin Added

- Now you need to configure the controller for the simulated robot. To do that go to the following directory: dobot\_moveit/config and open ros\_controllers.yaml file. Once you open it, add your controller's name joints and pid values for each planning group as such.
- You need to download the controllers using the following command:

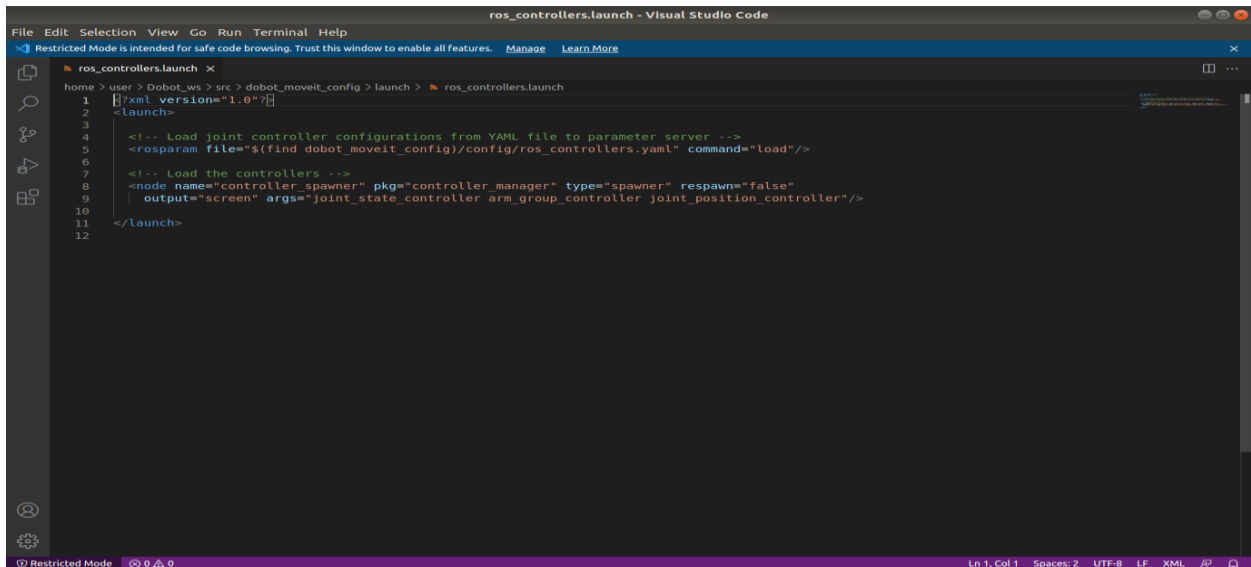
## Sudo apt-get install ros-melodic-ros-control ros-melodic-ros-controllers

- Finally, you should go to dobot\_moveit\_config/launch and open ros\_controllers.launch and add your controllers to the arguments.



```
ros_controllers.yaml - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
! ros_controllers.yaml x
home > user > Dobot_ws > src > dobot_moveit_config > config > ! ros_controllers.yaml
17 # Publish all joint states
18 # Creates the /joint_states topic necessary in ROS
19 joint_state_controller:
20   type: joint_state_controller/JointStateController
21   publish_rate: 50
22 controller_list:
23   - name: arm_group_controller
24     action_ns: follow_joint_trajectory
25     default: true
26     type: FollowJointTrajectory
27     joints:
28       - yaw_joint
29       - joint_1
30       - joint_2
31     - end_effector_joint
32 arm_group_controller:
33   type: position_controllers/JointTrajectoryController
34   joints:
35     - yaw_joint
36     - joint_1
37     - joint_2
38   - end_effector_joint
39 joint_position_controller:
40   type: effort_controllers/JointPositionController
41   joints:
42     - yaw_joint
43     - joint_1
44     - joint_2
45   - end_effector_joint
46   pid: {p: 100.0, i: 0.01, d: 10.0}
47
```

Figure 66: Arm Controller Added



```
ros_controllers.launch - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
ros_controllers.launch x
home > user > Dobot_ws > src > dobot_moveit_config > launch > ros_controllers.launch
1 <?xml version="1.0"?>
2 <launch>
3
4 <!-- Load joint controller configurations from YAML file to parameter server -->
5 <rosparam file="$(find dobot_moveit_config)/config/ros_controllers.yaml" command="load"/>
6
7 <!-- Load the controllers -->
8 <node name="controller_spawner" pkg="controller_manager" type="spawner" respawn="false"
9   output="screen" args="joint_state_controller arm_group_controller joint_position_controller"/>
10
11 </launch>
12
```

Figure 67: Argument Added

- The errors we faced during this simulation were the same ones we faced for the UBOT. The solution for the errors can be seen in part ii of the Appendix A.
- At the end of this section, credits should be given to Mr. Afif Sweidan, who's references and material he provided for us helped us in accomplishing the Dobot simulation successfully. A YouTube video is accordingly mentioned in the references at the end of the report that shows his work on the Dobot. (Sweidan, 2021)

#### iv Assembling Tank and Arm as One Robot

In order to visualize both the tank and arm as one robot on Rviz, we had to combine their two already created URDFs. The method is really simple; its steps are detailed below.

- First of all, you open the tank URDF, then you copy the arm's URDF script and paste it below the script of the tank.
- Now we have one big and combined script for both chassis.
- In order to make them connected with each other, so the arm is on top of the tank, we had to add a new joint, that defines the tank\_arm as a fixed joint. Then we adjust the origin of the joint in order to center it on top of the tank. The origin coordinates are estimated and are concluded after some trials. The parent link is the base link of the tank, and the child link of the joint is the base link of the arm.
- The script of the joint added is shown in the figure below.

```
295 <joint
296   name="tank_arm"
297   type="fixed">
298   <origin
299     xyz="-0.17 -0.06 0.11"
300     rpy="0 0 1.57" />
301   <parent
302     link="base_link" />
303   <child
304     link="base_link_1" />
305   <axis
306     xyz="0 0 -1" />
307 </joint>
```

Figure 68: Joint Added

- In order to visualize it on Rviz, we use the following command: `roslaunch Robot_Tank_Package display.launch`. After Rviz opens up, we add the robot model, and the robot that shows up is shown in the figure below.

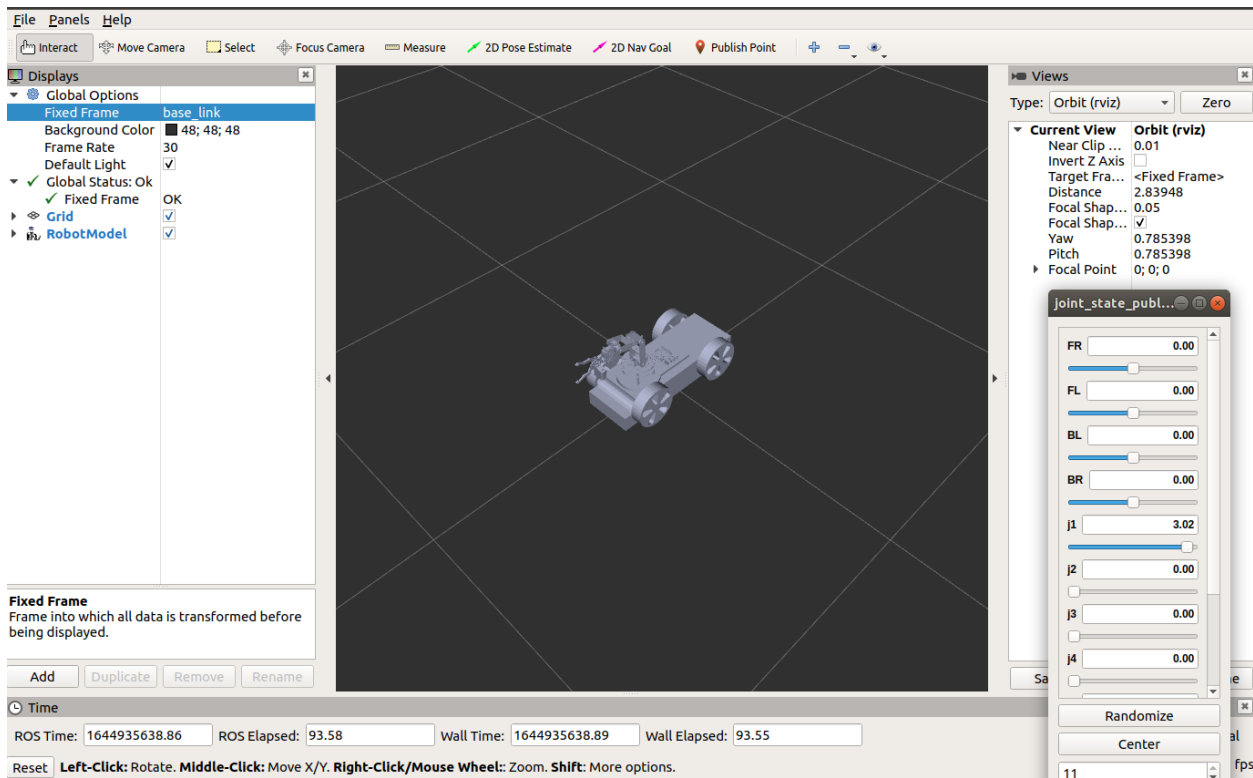


Figure 69: Assembled Robot

- In order to be able to perform motion planning with the arm on top of the tank, we had to follow the same procedure followed earlier regarding creating a Moveit package that will enable us to perform motion planning on Rviz and to visualize the arm's movement on Gazebo as a life-like simulation. The Moveit configuration steps will not be discussed here as they are detailed in section 2 of the appendix and you can follow them to get the perfect result.
- Finally, the full robot simulation can be accessed and seen by following the link attached in this reference. (Simulation, 2022)



## APPENDIX B

### CODES USED

#### i Arduino Code Used to Control the Tank

```
#include "ros.h"
#include <std_msgs/String.h>
#include <geometry_msgs/Twist.h>
#include <ros/time.h>
ros::NodeHandle nh;
float wheel1;
float wheel2;
float wheel1a;
float wheel2a;
float demandx;
float demandz;
float speed_act_left; // actual left wheel speed in m/s
float speed_act_right; // actual right wheel speed in m/s
unsigned long currentMillis;
unsigned long previousMillis;
int loopTime = 10;
// ** ROS callback & subscriber **
void velCallback( const geometry_msgs::Twist& vel)
{
    demandx = vel.linear.x;
    demandz = vel.angular.z;
    demandx = demandx * 350;
    demandz = demandz * 75;
}
ros::Subscriber<geometry_msgs::Twist> sub("cmd_vel" , velCallback); //create a subscriber
for ROS cmd_vel topic
void setup() {
```

```

nh.initNode();          // init ROS
nh.subscribe(sub);      // subscribe to cmd_vel
pinMode(6, OUTPUT);    // motor PWM pins
pinMode(11, OUTPUT);
pinMode(10, OUTPUT);
pinMode(9, OUTPUT);
}
void loop() {
  nh.spinOnce();
  // drive motors
  if (wheel1 > 0) {
    wheel1a = abs(wheel1);
    analogWrite(10, wheel1a);
    analogWrite(9, 0);
  }
  else if (wheel1 < 0) {
    wheel1a = abs(wheel1);
    analogWrite(9, wheel1a);
    analogWrite(10, 0);
  }
  else {
    analogWrite(9,0);
    analogWrite(10, 0);
  }
  // other motor
  if (wheel2 < 0) {
    wheel2a = abs(wheel2);
    analogWrite(11, wheel2a);
    analogWrite(6, 0);
  }
  else if (wheel2 > -0) {

```

```
wheel2a = abs(wheel2);
analogWrite(6, wheel2a);
analogWrite(11, 0);
}
else {
  analogWrite(6, 0);
  analogWrite(11, 0);
}
```

### **Code Explanation:**

To begin, we include the ROS library that we downloaded in the Arduino Ide and include the `std_msgs`, which includes wrappers for ROS primitive types that are described in the `msg` standard. It also has the `Empty` type, which may be used to deliver an empty signal. These kinds, however, do not transmit semantic information about their contents: each message only includes a field named "data." In this situation, the message type is `string`, and the key hit on the keyboard is read. After that we include the `geometry twist` message which gives the linear and angular velocity. Following that, we declare our variables and write a call back function, which is a function that you define rather than call. Typically, the function pointer is sent to another component, which will call your procedure when it appears suitable. In most circumstances, in ROS, configuring a callback is equivalent to configuring a message handler. In our code, we handle the `geometry Twist` message and construct a `cmd vel` subscriber to move the motors in the void setup. We also initialize the node handler to let all nodes to connect with each other and initiate the subscriber, whose purpose is to read the message (`Twist`). And we specify the output `pwm` pins of our Arduino UNO. In the `Void` loop we spin the node handler to make sure we listen for ROS messages and activate the callback if there is one and finally we handle the way our motors rotate by controlling the pins of our L298n Motor drive.

## ii Arduino Code Used to Control the Arm

```
#include <ros.h>
#include <sensor_msgs/JointState.h>
#include <Servo.h>
#include "ArduinoHardware.h"
// Set ROS - handler, subscribe message, publish message (debugging)
ros::NodeHandle nh;
Servo myservo0;
Servo myservo1;
Servo myservo2;
Servo myservo3;
Servo myservo4;
Servo myservo5;
int motor0 = 2;
int motor1 = 3;
int motor2 = 4;
int motor3 = 5;
int motor4 = 6;
int motor5 = 7;
int angle0 = 90;
int angle1 = 90;
int angle2 = 90;
int angle3 = 90;
int angle4 = 90;
int angle5 = 90;
int currentstate0 = 90;
int currentstate1 = 90;
int currentstate2 = 90;
int currentstate3 = 90;
int currentstate4 = 90;
int currentstate5 = 90;
```

```

int goalstate0;
int goalstate1;
int goalstate2;
int goalstate3;
int goalstate4;
int goalstate5;
int servoDegree[7];
// Define Motor position variables
double mtrDegree0;
double mtrDegree1;
double mtrDegree2;
double mtrDegree3;
double mtrDegree4;
double mtrDegree5;
double mtrDegree6;
// Function move motor to ROS angle
void servo_cb(const sensor_msgs::JointState& cmd_msg)
{
    goalstate0 = cmd_msg.position[0];
    goalstate1 = cmd_msg.position[1];
    goalstate2 = cmd_msg.position[2];
    goalstate3 = cmd_msg.position[3];
    goalstate4 = cmd_msg.position[4];
    goalstate5 = cmd_msg.position[5];
    mtrDegree0 = trimLimits(radiansToDegrees(cmd_msg.position[0]) + 90);
    mtrDegree1 = trimLimits(radiansToDegrees(cmd_msg.position[1]) + 90);
    mtrDegree2 = trimLimits(radiansToDegrees(cmd_msg.position[2]) + 4);
    mtrDegree3 = trimLimits(radiansToDegrees(cmd_msg.position[3]) + 90);
    mtrDegree4 = trimLimits(radiansToDegrees(cmd_msg.position[4]) + 90);
    mtrDegree5 = trimLimits(radiansToDegrees(cmd_msg.position[5]) + 90);
    myservo0.write(mtrDegree0);

```

```

myservo1.write(mtrDegree1);
myservo2.write(180-mtrDegree2);
myservo3.write(mtrDegree3);
myservo4.write(mtrDegree4);
myservo5.write(mtrDegree5);
}
ros::Subscriber<sensor_msgs::JointState> sub("joint_states", servo_cb);
void setup()
{
  myservo0.attach(motor0);
  myservo1.attach(motor1);
  myservo2.attach(motor2);
  myservo3.attach(motor3);
  myservo4.attach(motor4);
  myservo5.attach(motor5);
  nh.getHardware()->setBaud(115200); //changing baud rate speed because /joint_states is a big
topic (message)
  nh.initNode();
  nh.subscribe(sub);
}
void loop()
{
  nh.spinOnce();
}
// Convert radians to degrees
double radiansToDegrees(float position_radians)
{
  position_radians = position_radians * 57.2958;
  return position_radians;
}
// Sometimes servo angle goes just above 180 or just below 0 - trim to 0 or 180

```

```

double trimLimits(double mtr_pos)
{
  if (mtr_pos > 180) {
    mtr_pos = 180;
  }
  if (mtr_pos < 0) {
    mtr_pos = 0;
  }
  return mtr_pos;
}

```

### **Code Explanation:**

The code is quite simple. We begin by adding the necessary libraries, which are `ros.h`, `sensor_msgs/JointState.h`, and `Servo.h`, and then we specify the PWM pins of our Arduino Mega, which were linked to the signal pins of our servos. And we established our initial angle of our arm, which is the zero position where the arm stands vertically; in our instance, the angle was  $90^\circ$ , which we determined through experimentation on actual hardware. And we establish our current state, which is always zero, as well as our objective state for each motor, both of which are integer variables. Then the Arduino subscribes to the 'joint\_states' message, which is published by MoveIt. This message gives the angle state for each joint as the robot arm moves along the planned path to the desired position. This is sent as an array and the angles are in radians. The Arduino code then converts each joint to degrees. In the call back function, we see if there is an offset between the angle found in rviz and the real robotic arm so after checking each motor alone we see that motor 3 have offset in rviz its  $86^\circ$  while the initial position needed is  $90^\circ$  so we add on it  $4^\circ$ . We start the node handler in the void setup to allow all nodes to communicate with each other and we link the servo variable to the Pins in. IN the void loop, spin our nodes, and build an equation that converts the servo angle from degree to radiant, as well as a limit for the servo angle.

## APPENDIX C

### DATA SHEETS

#### i Tank DC Motors Data Sheet

<b>1.Standard Operating Conditions</b>			
NO.	Item	Specification	Test Method
1.1	Rated Voltage	DC 9.0V	Multimeter
1.2	Gear Ratio	1/75	
1.3	Rotation	CW	Handle
1.4	Motor Position	All Position in horizontal	Handle
1.5	Temperature	0 Degree - 30 Degree Celsius	Thermometer
1.6	Humidity	30% -95%	Hygroscope
<b>2.Performance Of Motors</b>			
NO.	Item	Specification	Test Method
2.1	No-load Speed	11500±10% rpm	Flash Speed Indicator
2.2	No-load Current	180mA (MAX)	DC Power Supply
2.3	Stall Current	4500mA (MAX)	DC Power Supply
2.4	Stall Torque	160g.cm	Torque Measure
<b>3.Performance of Gear Motors</b>			
NO.	Item	Specification	Test Method
3.1	Output Speed	150±10% rpm	Flash Speed Indicator
3.2	No-load Current	200mA (MAX)	DC Power Supply
3.3	Stall Current	4500mA (MAX)	DC Power Supply
3.4	Stall Torque	9.5kg.cm	Torque Measure
3.5	Rated Torque	3000g.cm	Torque Measure
3.6	Rated Current	1200mA (MAX)	DC Power Supply
3.7	Rated Speed	100±10%rpm	Flash Speed Indicator
3.8	Noise(30cm)	56dB	Digital Sound Levld Meter
<b>4.The Dimension</b>			
NO.	Item	Specification	Test Method
4.1	The Outside Shaft Length	14.5mm	Vernier Calipers
4.2	Shaft End Play	0.05-0.50mm	Frock
4.3	Screw Size	M3.0	Frock
4.4	Dia of shaft	φ 4mm D3.5	Vernier Calipers
4.5	Outline Monting Dimension	Refer to the Outline Drawing	Calipe

Figure 70: Tank DC Motors Data Sheet



## ii Robotic Arm Servo Motors Data Sheets

- LDX-335MG

<b>Specifications</b>	
Working Voltage	6-7.4V
Rated voltage	6V
No-load Current	100mA
Speed	0.16sec/60° (7.4V)
Torque	15kg.cm(6.0V) 17kg.cm(7.4V)
Precision	3us
Wire length	30cm
Weight	About 60g
Size	40*20*40.5mm

Table 7: LDX-335MG

- LDX-218MG

<b>Specifications</b>	
Working Voltage	6-7.4V
No-load Current	100mA
Speed	0.16sec/60° (7.4V)
Torque	15kg.cm(6.0V) 17kg.cm(7.4V)
Accuracy	0.3°
Controllable angle range	From 0 to 180 degrees
Size	40*20*40.5mm

Table 8: LDX-218MG

- LDF-06

<b>Specifications</b>	
Working Voltage	6-8.4V
No-load Current	100mA
Speed	0.16sec/60° (7.4V)
Torque	6kg.cm (6.6V)
Wire Length	50 cm
Weight	47g
Size	40*20*40.5mm

Table 9: LDF-06

- LD-1501MG

<b>Specifications</b>	
Working Voltage	6-7.4V
Rated voltage	6V
No-load Current	100mA
Speed	0.16sec/60° (7.4V)
Torque	15kg.cm(6.0V) 17kg.cm(7.4V)
Precision	3us
Wire length	30cm
Weight	About 60g
Size	40*20*40.5mm

Table 10: LD-1501MG

# APPENDIX D

## SURVEY

### **i About this Questionnaire**

As part of fulfilling the requirements to obtain a bachelor's degree in engineering, we are a 4-students-group working on our final/senior year project. Our project is entitled "Bomb Disposal Robot". The aim of our project is to design a fully functional robotic prototype able of detecting and disposing bombs/mines safely while the Operator can sit on a clear distance of the bomb, controlling the robot, where he can be totally safe, preventing the loss of human life.

This is an open-answer questionnaire. It will be used to obtain your feedback based on your experiences in this field. The questions are oriented in a way for us to get a better idea of what we are dealing with, the markets' needs, and finally the components that we might need in order to perfectly implement our project.

### **ii Questions**

1. What are the constrains of any mine/bomb disposal mission?

The constraints of any mine/bomb disposal mission are usually the nature of the land, the funding, the planning phase, equipment processing, the type of the mines/bombs, loss of enough data while gathering information about the land, the mission being close to households...

2. Yearly, what is the number of mines that you are finding and disposing in Lebanon?

Around 17,000 (Most of which are located in South Lebanon).

3. How do your mine/bomb disposal teams actually handle mines/bombs found on sites?

There are two types of missions, the first being that someone informed about a possible mine/bomb somewhere, in this case someone specialized will be sent to the field to investigate the scenario and act accordingly. The other case is when it is a known mine field, in this case a risk assessment plan will be prepared and then the SOP standards of public safety will be followed.

4. What is the average weight of most mines that you are disposing? (This is necessary in order to estimate the approximate drag and lift capacity of our tank and robotic arm respectively).

Small bombs usually weigh around 300/400g and can go up to 5kg

5. What are the safety measures taken before responding to any mine/bomb disposing mission?

Before the mission, preparation for the mission with the necessary gear needed, a proper risk assessment, a study of the nature of the land, and then an intervention plan is put. During the

mission, insuring the safety of the Deminer and the surrounding people, insuring that the Deminer knows how to use the gear provided, and site visits are conducted to insure quality assurance.

6. What are the tools needed in order to Detect and Dispose mines/bombs?

A pickup arm since the bomb's explosion is localized, so being a bit far from a small bomb won't have severe consequences; Rope (when necessary); a container when needed to dispose bombs for long distances; Sand bags (since sand responds to the explosion effect).

7. What sensors do you use in order to detect underground mines?

A magnetic field sensor is used when trying to locate mines that are made out of metal, this usually finds the mine but it doesn't necessarily mean that it contains explosives just the metallic body. However, underground radar is used to detect explosives inside of the mines.

8. What are the scenarios that robots can operate in? Can they be used in any situation?

Robots can only be used in disposing processes and are not recommended to be used in detection or diffusing processes.

9. Will the use of a remotely controlled robot make the mission easier? Or will it complicate things even more?

If the mission is prioritized on disposing the bomb from one place to another it will make it much easier, but if detecting or diffusing is involved it would make the process a lot more difficult.

10. Do your personnel have any previous experience with any bomb disposal and detection robots?

Yes, some of the personnel's have some previous experience using robots in general and are trained to use them although most of them are military robots and not in the field of disposing and detecting. However, they are not frequently used.

11. What is the main cause behind the lack of use of mine/bomb disposal robots in Lebanon? Is it a financial problem, regional conflicts or they deemed it to be unnecessary?

It is a financial problem for sure and especially in these days with the inflation of the Lebanese currency, a robot requires a high maintenance cost. Also, disposing cases are very few most of the cases require acting immediately on site.

12. Do you think that investing in a mine/bomb detection and disposal robot will result in a high ROI? (Economically and also regarding the safety of your personnel).

The investment seen here is an investment in the safety of the Deminer and not an economical investment, and the safety investment is for sure more beneficial.

13. If you are willing to invest in a robot what are the features and specifications that you will be looking for as your top priority?

The robot is required to work in heavy duty meaning that it can work on the nature of the land in most Lebanese ground and it can withstand the difficulty of the missions, also it must be a certain weight to not trigger any bomb/mine, the user must be able to control it from a safe distance (up to 100m), it must have low cost maintenance and a powerful battery.

14. How long does it take for you to train your personnel to become specialists in disposing mines/bomb?

Two-week training is sufficient for personnel to become a Deminer and specialist in disposing bombs/mines. However, a robot needs constant training and practice to execute perfectly.

15. Do you think that robot evolution will have the ability to replace the human role in disposing mines and bombs?

Not in the near future, maybe in the field of disposing bombs we might see the replacing of the human role for safety measures, but in the field of detecting and diffusing a robot takes longer time and costs more money.

APPENDIX E  
MEETING MINUTES  
MINUTES OF MECA 595A MEETING (1)  
COLLEGE OF ENGINEERING – MME Department– RHU  
ON OCTOBER 21<sup>st</sup>, 2021 AT 10:00AM

---

**Present:** DR.Hassan Hariri, Pierre Gerges, Rami Assaf, Abed el Rahman Hammoud.

**Absent:** Mounir Nahle.

---

The meeting came to order at 10:15 am and ended at 10:37 am. The meeting was held online using MS Teams.

***1. Updates***

---

In this week we have made general researches about what do Bomb Disposal Robots actually do. We discussed the following:

- We had a general understanding of the goal of the robot and its actual mission.
- We have presented the abstract and the introduction of the project stating the problem statement, the solution, goals, objectives and the project and team SWOT analysis.

***2. Advisor Comments and Recommendations***

---

Dr. Hariri recommendation was to start searching for actual systems available in the market or in research labs that are done by competitors in order to see their specifications.

***3. Expected Deliverables for Next Meeting***

---

We have to do a research about competitors that have already worked on such a system – We need to specify all their project's specifications – Make a comparison table between them showing the differences in their design – prepare a PowerPoint presentation showcasing the work done in this week.

***4. Assessment***

---

NONE.

The next meeting is scheduled for Thursday 28 October 2021.

Minutes taken by: Pierre Gerges

**MINUTES OF MECA 595A MEETING (2)**  
**COLLEGE OF ENGINEERING – MME Department– RHU**  
**ON OCTOBER 28<sup>th</sup>, 2021 AT 11:00AM**

---

**Present:** DR.Hassan Hariri, Pierre Gerges, Rami Assaf, Abed el Rahman Hammoud, Mounir Nahle.

**Absent:** NA.

---

The meeting came to order at 11 am and ended at 11:48 am. The meeting was held online using MS Teams.

***1. Updates***

---

In this week we have made researches different competitors that has their own robots in the market. We discussed the following:

- We had a general understanding of the goal of the robot and its actual mission.
- We have presented all the specifications and the features of each competitor we showed.

***2. Advisor Comments and Recommendations***

---

Dr Hariri recommendation was to make a table comparing all the competitors' features and specs to know which one is better in terms of the price allocated to them.

***3. Expected Deliverables for Next Meeting***

---

We have to make a big table comparing all the competitors' specs and features – we have to dig and research deeper about all the features and the hardware they used and the problems they encountered to figure out how they solved them in order to understand all the technologies behind building a well-designed robot.

***4. Assessment***

---

NONE.

The next meeting is scheduled for Thursday 4 November 2021.

Minutes taken by: Pierre Gerges

**MINUTES OF MECA 595A MEETING (3)**  
**COLLEGE OF ENGINEERING – MME Department– RHU**  
**ON November 4<sup>th</sup>, 2021 AT 11:00AM**

---

**Present:** DR.Hassan Hariri, Pierre Gerges, Rami Assaf, Abed el Rahman Hammoud.

**Absent:** Mounir Nahle.

---

The meeting came to order at 11 am and ended at 12 pm. The meeting was held on campus in the meeting room.

### ***1. Updates***

---

In this week we have made researches about the different technologies behind every feature of our competitors. We discussed the following:

-We analyzed a specifications comparison table to try to figure out which robot was better in order to aim for it.

-We have presented all the features that we found for the following robots (CALIBER T5, TALON, and VANGUARD), explained the technologies behind them and the solutions that helped in achieving those features.

### ***2. Advisor Comments and Recommendations***

---

Dr Hariri recommendation was to try to get some photos of those robots near actual objects like cars or laptops to just get an idea about their real size and appearance in real life.

### ***3. Expected Deliverables for Next Meeting***

---

We have to put our own hypothetical specification table for our own project – We have to make our robot competitive with the other robots in the markets by getting into it as much features as we can in the minimum cost – We have to make a table containing all the components and price tags that we need to make the robot in order to estimate the final cost of the robot in the market (Bill of Materials).

### ***4. Assessment***

---

NONE.

The next meeting is scheduled for Thursday 11 November 2021.

Minutes taken by: Pierre Gerges



**MINUTES OF MECA 595A MEETING (4)**  
**COLLEGE OF ENGINEERING – MME Department– RHU**  
**ON November 11<sup>th</sup>, 2021 AT 3:20 PM**

---

**Present:** DR.Hassan Hariri, Pierre Gerges, Rami Assaf, Abed el Rahman Hammoud, Mounir Nahle.

**Absent:** NA.

---

The meeting came to order at 3:20 PM and ended at 4:28 PM. The meeting was held online using MS Teams.

***1. Updates***

---

In this week we have established the specs and features of our robot, we put into perspective our competitor's robot's sizes near actual and real life things. Moreover, we have prepared some interview questions that we can ask to customers like the Lebanese army or other entities in order to get better information about what is desired in the market and the budget that these people or organizations allocate for such technologies.

***2. Advisor Comments and Recommendations***

---

Dr Hariri recommendation was to insert the photos that we got for the robots in their respective specs table for better understanding – he recommended that we put all our specs and our competitor's specs in an excel sheet so we can compare them directly.

***3. Expected Deliverables for Next Meeting***

---

We have to make adjustments on our specs table – design an excel sheet containing all the specs and figures for all the robots comparing them to ours – Concepts Generation.

***4. Assessment***

---

NONE.

The next meeting is scheduled for Thursday 18 November 2021.

Minutes taken by: Pierre Gerges

**MINUTES OF MECA 595A MEETING (5)**  
**COLLEGE OF ENGINEERING – MME Department– RHU**  
**ON November 25<sup>th</sup>, 2021 AT 11 PM**

---

**Present:** DR.Hassan Hariri, Pierre Gerges, Rami Assaf, Abed el Rahman Hammoud.

**Absent:** Mounir Nahle

---

The meeting came to order at 11 AM and ended at 12:10 PM. The meeting in the meeting room in the MME department.

***1. Updates***

---

In this week we have established an updated specs table. We presented our competitor's detection robots features and technologies. Moreover, we talked about what we have to do regarding the navigation of our robot and decided that the robot will be totally controlled by an operator and it will not be autonomous.

***2. Advisor Comments and Recommendations***

---

There were no comments.

***3. Expected Deliverables for Next Meeting***

---

We decided that we have to do a brainstorming session next week where we will discuss all the following ideas to finalize phase 1 of the project: Survey analysis – working environment – user interaction – robot interaction – comparison matrix – possible solutions and concepts generation.

***4. Assessment***

---

NONE.

The next meeting is scheduled for Thursday 2 December, 2021.

Minutes taken by: Pierre Gerges

**MINUTES OF MECA 595A MEETING (6)**  
**COLLEGE OF ENGINEERING – MME Department– RHU**  
**ON December 2<sup>sd</sup>, 2021 AT 3:20 PM**

---

**Present:** DR.Hassan Hariri, Pierre Gerges, Rami Assaf, Abed el Rahman Hammoud, Mounir Nahle.

**Absent:** NA.

---

The meeting came to order at 3:20 PM and ended at 6:20 PM. The meeting was held in CLAB3 in the MME department.

***1. Updates***

---

In this week we have done our brainstorming session where we agreed on the primary concept of our project after we discussed all the following topics thoroughly: Survey analysis – working environment – user interaction – robot interaction – comparison matrix – possible solutions and concepts generation.

***2. Advisor Comments and Recommendations***

---

Dr. Hariri said that the meeting was fruitful and that we are going in the right direction.

***3. Expected Deliverables for Next Meeting***

---

we have to make out bill of materials in 2 cases or scenarios: funded project or P.O.C. Additional tasks will be added later via WhatsApp by Dr. Hariri.

***4. Assessment***

---

NONE.

The next meeting is scheduled for Thursday 9 December, 2021.

Minutes taken by: Pierre Gerges

**MINUTES OF MECA 595A MEETING (7)**  
**COLLEGE OF ENGINEERING – MME Department– RHU**  
**ON December 20<sup>th</sup>, 2021 AT 1 PM**

---

**Present:** DR.Hassan Hariri, Pierre Gerges, Rami Assaf, Abed el Rahman Hammoud.

**Absent:** Mounir Nahle.

---

The meeting came to order at 1 PM and ended at 2 PM. The meeting was held online using teams.

***1. Updates***

---

In this week we have presented our task management or GANTT chart (tasks function of time) – plus we explained how we will ensure communication between the CCU and the robot with transmitter and receiver like XBEE S2C 2mw.

***2. Advisor Comments and Recommendations***

---

Dr. Hariri said that he will follow up with us regarding doing a workshop for robotic arm manipulation and for vision and grasping.

***3. Expected Deliverables for Next Meeting***

---

Each one of us will work on his assigned task: Rami will continue working on the drivetrain – Abed el Rahman will continue working on XBEE – Pierre will start working on the manipulation simulation.

***4. Assessment***

---

NONE.

The next meeting is scheduled for Monday 27 December, 2021.

Minutes taken by: Pierre Gerges

**MINUTES OF MECA 595A MEETING (8)**  
**COLLEGE OF ENGINEERING – MME Department– RHU**  
**ON January 5<sup>th</sup>, 2022 AT 7 PM**

---

**Present:** DR.Hassan Hariri, Pierre Gerges, Rami Assaf.

**Absent:** Mounir Nahle, Abed el Rahman Hammoud.

---

The meeting came to order at 7 PM and ended at 7:50 PM. The meeting was held online using teams.

***1. Updates***

---

In this week we have presented our progress in the simulation of the University Robotic Arm (6 DOF UBOT):

- Export Solidworks to URDF.
- Error debugging procedure that was followed.
- UBOT visualization and path planning on RVIZ and Gazebo.

***2. Advisor Comments and Recommendations***

---

Dr. Hariri said that we have to speed up the process and finish totally with the simulation to start the implementation phase – he recommended to start learning computer vision for object detection and recognition – finally we agreed to document the procedure we followed in the simulation of the arm in order to establish a proper tutorial that we can add in our report and so that future students can learn from.

***3. Expected Deliverables for Next Meeting***

---

For next week we will finalize the simulation part of the robotic arm – we have to do also the simulation of the robotic tank (as done in Wajih’s tutorial).

***4. Assessment***

---

NONE.

The next meeting is scheduled for Monday 13 January, 2022.

Minutes taken by: Pierre Gerges

**MINUTES OF MECA 595A MEETING (9)**  
**COLLEGE OF ENGINEERING – MME Department– RHU**  
**ON January 14<sup>th</sup>, 2022 AT 7 PM**

---

**Present:** DR.Hassan Hariri, Pierre Gerges, Rami Assaf, Mounir Nahle, Abed el Rahman Hammoud, Hassan Hareb.

**Absent:** None.

---

The meeting came to order at 7 PM and ended at 8 PM. The meeting was held online using teams.

### ***1. Updates***

---

The following was discussed in this week's meeting:

- Dr. Hariri set some basic rules that will be applied this semester regarding the delivery of meeting minutes and the preparation of agenda items for each meeting.
- We started the meeting by showing the complete simulation of the tank robot and deciding that we will be moving it with the keyboard using the teleop\_twist\_keyboard package.
- We talked about the Dobot robotic arm simulation and we showed its results via videos on WhatsApp. However, we excluded its use due to its high voltage necessities because that will be burden to our robot.
- Then we talked about the UBOT simulation and that we still have one small error to fix in order to complete it and move to the implementation phase. The error will be furtherly discussed with Mr. Hussein Hareb in order to solve it.
- Then we talked about the method of communication that we will be using. We decided to exclude RF communication and that we will use Wi-Fi as a way of communication: either by mobile hotspot or by putting a modem on the robot itself.
- Finally, we decided that we will use an RGBD camera (Kinect camera) for object detection instead of RGB and ultrasonic sensor because it's more advanced and accurate.

### ***2. Advisor Comments and Recommendations***

---

Dr. hariri said that even though we excluded the use of the dobot robotic arm, we still have to document the work done with it in order to establish a tutorial about it at the end of our report. Finally, he recommended that we start working on perception on the turtlebot because it has an RGBD camera so we get a thorough idea of its use.

### ***3. Expected Deliverables for Next Meeting***

---

For next week we have to:

- Finalize the simulation of the UBOT.
- Finalize the tank implementation (wiring, coding and moving the tank with keyboard).
- Start practical implementation of the simulation on the UBOT itself
- Provide report updates.
- Establish a Bill of Material that corresponds with the changes we made.

#### *4. Assessment*

---

NONE.

The next meeting is scheduled for Friday 21 January, 2022.

Minutes taken by: Pierre Gerges

**MINUTES OF MECA 595A MEETING (10)**  
**COLLEGE OF ENGINEERING – MME Department– RHU**  
**ON January 21<sup>st</sup>, 2022 AT 7 PM**

---

**Present:** DR.Hassan Hariri, Rami Assaf, Abed el Rahman Hammoud, Pierre Gerges.

**Absent:** Mounir Nahle.

---

The meeting came to order at 7 PM and ended at 8 PM. The meeting was held online using teams.

### *1. Updates*

---

The following was discussed in this week's meeting:

- The meeting started with the introduction of the agenda items.
- And then we gave updates on our progress regarding the implementation of the robotic arm.
- Then we showed the full and complete simulation tutorials of the UBOT, the Dobot and the tank in the report's appendix.
- Finally, we discussed what happened with the tank implementation that we have done in the lab and the problems that we had faced concerning the hardware (like the fried motor drive).

### *2. Advisor Comments and Recommendations*

---

Dr. hariri said that we have to make slight corrections in the appendix of the report, like: putting the video links of the simulation in the references. Moreover, we have to add the video links of Mr. Wajih and Mr. Hussein Hareb as references for the work that we've done.

### *3. Expected Deliverables for Next Meeting*

---

For next week we have to:

- Finalize the tank implementation (wiring, coding and moving the tank with keyboard).
- Resume working on the UBOT robotic arm implementation.
- Make slight corrections in the report's appendix then continue working on the report's body content.
- Establish a Bill of Material that corresponds with the changes we made.
- CAD and 3D print an object that might be used to test grasping.

### *4. Assessment*

---

NONE.

The next meeting is scheduled for Friday 28 January, 2022.

Minutes taken by: Rami Assaf



**MINUTES OF MECA 595A MEETING (11)**  
**COLLEGE OF ENGINEERING – MME Department– RHU**  
**ON January 28<sup>th</sup>, 2022 AT 7 PM**

---

**Present:** DR.Hassan Hariri, Rami Assaf, Abed el Rahman Hammoud, Pierre Gerges.

**Absent:** Mounir Nahle.

---

The meeting came to order at 7:10 PM and ended at 8:10 PM. The meeting was held online using teams.

### *1. Updates*

---

The following was discussed in this week's meeting:

- The meeting started with the introduction of the agenda items.
- Then we gave updates regarding what happened in our interview with the Lebanese army.
- We finalized the implementation of the tank.
- We discussed our progress regarding the work being done on the UBOT robotic arm and the problems we faced during its implementation.
- We presented our Bill of Material (BOM).
- Finally, we presented the part that can be used to test grasping with the gripper.

### *2. Advisor Comments and Recommendations*

---

Dr. Hariri said that we must make the tank wiring more robust to keep it safe and operational while moving. Plus, he recommended to divide the tasks done more evenly to finalize the work in a faster time.

### *3. Expected Deliverables for Next Meeting*

---

For next week we have to:

- Keep working on the UBOT implementation.
- Finalize the report's literature review.
- Gazebo simulation of the prototype's whole assembly (Tank and UBOT together).
- Battery design for the robot.

### *4. Assessment*

---

NONE.

The next meeting is scheduled for Friday 4 February, 2022.

Minutes taken by: Pierre Gerges

**MINUTES OF MECA 595A MEETING (12)**  
**COLLEGE OF ENGINEERING – MME Department– RHU**  
**ON February 11<sup>th</sup>, 2022 AT 7 PM**

---

**Present:** DR.Hassan Hariri, Rami Assaf, Abed el Rahman Hammoud, Pierre Gerges, Mounir Nahle..

**Absent:** N/A.

---

The meeting came to order at 7:05 PM and ended at 8:10 PM. The meeting was held online using teams.

### ***1. Updates***

---

The following was discussed in this week's meeting:

- The meeting started with the introduction of the agenda items.
- We discussed our progress regarding the work being done on the UBOT robotic arm and the problems we faced during its implementation, and the work that will be done on Saturday February 12, 2022 regarding wiring and implementation.
- We talked about battery sizing.
- Finally, we talked about the full simulation of the whole robot on gazebo and the problems faced while doing it especially combining the two urdf files into one urdf.

### ***2. Advisor Comments and Recommendations***

---

Dr. Hariri said that we must make the tank wiring more robust to keep it safe and operational while moving. Plus, he recommended to divide the tasks done more evenly to finalize the work in a faster time.

### ***3. Expected Deliverables for Next Meeting***

---

For next week we have to:

- Keep working on the UBOT implementation.
- Finalize the report's literature review.
- Gazebo simulation of the prototype's whole assembly (Tank and UBOT together).
- Battery design for the robot.

### ***4. Assessment***

---

NONE.

The next meeting is scheduled for Friday 4 March, 2022.

Minutes taken by: Pierre Gerges

**MINUTES OF MECA 595A MEETING (13)**  
**COLLEGE OF ENGINEERING – MME Department– RHU**  
**ON March 4<sup>th</sup>, 2022 AT 7 PM**

---

**Present:** DR.Hassan Hariri, Rami Assaf, Abed el Rahman Hammoud, Pierre Gerges, Mounir Nahle.

**Absent:** None.

---

The meeting came to order at 7:00 PM and ended at 8:30 PM. The meeting was held online using teams.

***1. Updates***

---

The following was discussed in this week's meeting:

- The meeting started with the introduction of the agenda items.
- We showed the full simulation of our assembled robot and we discussed the Kinect camera simulation and the problems that we faced while doing it.
- We gave updates about battery sizing and motor selection calculations.

***2. Advisor Comments and Recommendations***

---

None.

***3. Expected Deliverables for Next Meeting***

---

For next week we have to:

- Full simulation including the camera.
- Continue the implementation of the arm.
- Finalize the battery sizing and motor selection calculations.

***4. Assessment***

---

NONE.

The next meeting is scheduled for Friday 11 march, 2022.

Minutes taken by: Pierre Gerges

**MINUTES OF MECA 595A MEETING (14)**  
**COLLEGE OF ENGINEERING – MME Department– RHU**  
**ON March 19<sup>th</sup>, 2022 AT 7:45 PM**

---

**Present:** DR.Hassan Hariri, Rami Assaf, Abed el Rahman Hammoud, Pierre Gerges.

**Absent:** Mounir Nahle.

---

The meeting came to order at 7:45 PM and ended at 8:30 PM. The meeting was held online using teams.

### *1. Updates*

---

The following was discussed in this week's meeting:

- The meeting started with the introduction of the agenda items.
- We showed the whole assembly of our robot prototype.
- We showed the tank movement in full functionality.
- We showed how we implemented the homing (Zeroing) of the robotic arm's servo motors.
- We talked about the implementation of the Kinect Camera.
- We discussed motor design of the arm and the tank motors (Theoretically).

### *2. Advisor Comments and Recommendations*

---

Draw a Schematic of the robotic arm to understand how the joints work.

### *3. Expected Deliverables for Next Meeting*

---

For next week we have to:

- Show a Video of the robotic arm moving in a full workspace.
- Work on the implementation of the vision part.
- Continue working on the design process.

### *4. Assessment*

---

NONE.

The next meeting is scheduled for Friday 25 march, 2022.

Minutes taken by: Pierre Gerges

**MINUTES OF MECA 595A MEETING (15)**  
**COLLEGE OF ENGINEERING – MME Department– RHU**  
**ON March 25<sup>th</sup>, 2022 AT 6:40 PM**

---

**Present:** DR.Hassan Hariri, Rami Assaf, Abed el Rahman Hammoud, Pierre Gerges, Mounir Nahle.

**Absent:** NA.

---

The meeting came to order at 6:40 PM and ended at 7:50 PM. The meeting was held online using teams.

### ***1. Updates***

---

The following was discussed in this week's meeting:

- The meeting started with the introduction of the agenda items.
- We further discussed the design process of the robotic arm and tank's motors according to our goal specifications.
- We talked about the problems that we have in regards of the motors of our robotic arm and that the solution is to buy and replace the base-link motor with a better one with higher torque and that the use of the Dobot Robotic arm as a replacement is will not be considered due to time constraints.
- We talked about the successful implementation of the Kinect Camera and the steps followed to achieve it connecting it to a laptop, and the error that we faced while compiling the installed package on the raspberry pi.

### ***2. Advisor Comments and Recommendations***

---

NA.

### ***3. Expected Deliverables for Next Meeting***

---

For next week we have to:

- Try to implement the motor change on the robotic arm.
- Work on the implementation of the vision part on the raspberry pi.
- Continue working on the design process.

### ***4. Assessment***

---

NONE.

The next meeting is scheduled for 1 April, 2022.

Minutes taken by: Pierre Gerges

# APPENDIX F

## STANDARDS

### **Ingress Protection (IP) Rating (IEC Standard 60529)**

The IP system is an internationally recognized method to indicate the degree of protection against the ingress of dust, solid objects and moisture into an enclosure. The letters "IP" are followed by two numerals.

#### **- First Numeral**

Protection of persons against contact with or approach to live parts and against contact with moving parts, other than smooth rotating shafts and the like, inside the enclosure and protection of the equipment against ingress of solid foreign bodies in accordance with IEC 60598-1:2003

- 0 Not protected
- 1 Protected against solid objects 50 mm in diameter or greater. (A large surface of the body, such as a hand and no protection against deliberate access).
- 2 Protected against solid objects 12 mm in diameter or greater. (Fingers or similar objects not exceeding 80mm in length).
- 3 Protected against solid objects 2.5 mm in diameter or greater. (Tools, wires, etc., of diameter or thickness greater than 2.5 mm).
- 4 Protected against solid objects 1mm in diameter or greater. (Wires or other similar solid material of thickness greater than 1mm in diameter).
- 5 Dust protected. (Dust does not enter in sufficient quantity to interfere with satisfactory operation of equipment).
- 6 Dust tight.

#### **- Second Numeral**

The second numeral indicates the degree of protection against the ingress moisture as defined in IEC 60598-1:2003.

- 0 Not protected
- 1 Protected against dripping water. (Dripping water (vertically falling drops) shall have no harmful effect).

- 2 Protected against dripping water when tilted up to 15°. (Vertically dripping water shall have no harmful effect when the enclosure is tilted at an angle up to 15° from its normal position).
- 3 Protected against spraying water. (Water falling as a spray at any angle up to 60° from the vertical shall have no harmful effect).
- 4 Protected against splashing water. (Water splashing against the enclosure from any direction shall have no harmful effect).
- 5 Protected against water jets. (Water projected by a nozzle against enclosure from any direction shall have no harmful effects).
- 6 Protected against heavy seas. (Water from heavy seas or projected in powerful water jets shall not enter the enclosure in harmful quantities).
- 7 Protected against the effects of temporary immersion. (Ingress of water in harmful quantity shall not be possible when the enclosure is immersed in water under defined conditions of pressure and time).
- 8 Protected against continuous immersion. (The equipment is suitable for continuous submersion in water under conditions which shall be specified by the manufacturer).

## **Impact Protection (IK) Rating**

Degrees of protection provided by enclosures for electrical equipment against external mechanical impacts in accordance with IEC 62262:2002 and IEC 60068-2-75:1997.

- IK00 Not protected
- IK01 Protected against 0.14 joules impact. (Equivalent to impact of 0.25 kg mass dropped from 56 mm above impacted surface).
- IK02 Protected against 0.2 joules impact. (Equivalent to impact of 0.25 kg mass dropped from 80 mm above impacted surface).
- IK03 Protected against 0.35 joules impact. (Equivalent to impact of 0.25 kg mass dropped from 140 mm above impacted surface).
- IK04 Protected against 0.5 joules impact. (Equivalent to impact of 0.25 kg mass dropped from 200 mm above impacted surface).
- IK05 Protected against 0.7 joules impact. (Equivalent to impact of 0.25 kg mass dropped from 280 mm above impacted surface).
- IK06 Protected against 1 joules impact. (Equivalent to impact of 0.25 kg mass dropped from 400 mm above impacted surface).
- IK07 Protected against 2 joules impact. (Equivalent to impact of 0.5 kg mass dropped from 400 mm above impacted surface).
- IK08 Protected against 5 joules impact. (Equivalent to impact of 1.7 kg mass dropped from 300 mm above impacted surface).
- IK09 Protected against 10 joules impact. (Equivalent to impact of 5 kg mass dropped from 200 mm above impacted surface).
- IK10 Protected against 20 joules impact. (Equivalent to impact of 5 kg mass dropped from 400 mm above impacted surface).



## ABET KPIs

	<i>How was it addressed in your SLP?</i>	<i>Where was it addressed in your SLP?</i>
<b>1. An ability to identify, formulate, and solve complex engineering problems by applying principles of engineering, science, and mathematics</b>		
1.1 An ability to apply knowledge of mathematics	<i>Formulas used to calculate the torque needed for our arm servo motors</i>	<i>Chapter 3 sec. 3.2.2</i>
1.2 An ability to apply knowledge of Science	<i>Choosing the right coefficient of friction for our desired operating terrain</i>	<i>Chapter 3 sec. 3.2.4</i>
1.3 An ability to apply knowledge of Engineering	<i>Applied Knowledge in: ROS Arduino Programming Mechatronics System Design</i>	<i>SLP Report</i>
<b>2. An ability to apply engineering design to produce solutions that meet specified needs with consideration of public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors</b>		
2.1 Design a system/component of a system or a process to meet specific needs while respecting safety, health and welfare of the public and adhering to cultural, social, environmental and economic factors.	<i>The Design Process to select the right components according to our</i>	<i>Chapter 3 &amp; chapter 4 sec. 4.4</i>

	<i>specifications and the reverse engineering design to verify that the components that we possess are adequate to satisfy our requirements.</i>	
2.2 Modify a system/component of a system or a process to meet specific needs while respecting safety, health and welfare of the public and adhering to cultural, social, environmental and economic factors.	<i>Utilizing RGB-D Kinect Camera</i>	<i>Chapter 2 sec. 2.4.3</i>
<b>3. An ability to communicate effectively with a range of audiences</b>		
3.1 Ability to write a well-structured formal report/technical document that addresses an audience with diverse educational-background	<i>Senior Report</i>	<i>SLP Report</i>
3.2 Ability to deliver a well-structured formal presentation that addresses an audience with diverse educational-background <sup>1</sup>	<i>Simulation Demo Presentation</i>	<i>SLP Presentation</i>
<b>4. An ability to recognize ethical and professional responsibilities in engineering situations and make informed judgments, which must consider the impact of engineering solutions in global, economic, environmental, and societal contexts</b>		
4.1 Identify global, economic, environmental, and societal impact of implementing engineering solutions using applicable engineering code of ethics to differentiate between ethical/unethical behaviors	<i>Utilizing RGB-D Kinect Camera</i>	<i>Chapter 2 sec. 2.4.3</i>
4.2 Identify global, economic, environmental, and societal impact of implementing engineering solutions using applicable engineering standards and codes to differentiate between professional/unprofessional behaviors	<i>Isolation of wiring of the arm and tank and color coding.</i>	<i>SLP Report and Appendix E for Meeting Minutes</i>

	<i>Punctuality in attending meetings.</i>	
<b>5. An ability to function effectively on a team whose members together provide leadership, create a collaborative and inclusive environment, establish goals, plan tasks, and meet objectives</b>		
5.1 Ability to plan and organize team tasks collectively to meet established goals	<i>Scheduling Future Tasks in Meetings</i>	<i>Appendix E</i>
5.2 Ability to carry out tasks assigned by the team to attain set objectives.	<i>Tasks Completed and Discussed in Meetings</i>	<i>Appendix E</i>
<b>6. An ability to develop and conduct appropriate experimentation, analyze and interpret data, and use engineering judgment to draw conclusions</b>		
6.1 An ability to design experiments.	<i>Simulation process of the robotic arm. Trial and error for fixing the errors encountered.</i>	<i>Appendix A, Section ii</i>
6.2 An ability to conduct experiments.	<i>Different experiments were conducted to test the motors of the robotic arm.</i>	<i>Chapter 5, Section 5.2</i>
6.3 An ability to draw apt evidence-based conclusions by analyzing and interpreting data.	<i>Data Collected by the Kinect Camera to</i>	<i>Chapter 5, Section 5.3</i>

	<i>Analyze the Environment.</i>	
<b>7. An ability to acquire and apply new knowledge as needed, using appropriate learning strategies.</b>		
7.1 Identify necessary skills and tools of contemporary engineering practice to solve a problem at hand.	<i>Robotic Arm Servo Motors Dismounting and re-assembling to test the torque of the motors.</i>	<i>Chapter 5 Section 5.2</i>
7.2 Apply self-learned skills and tools of contemporary engineering practice to solve a problem at hand.	<i>Our ROS Approach</i>	<i>Chapter 2, section 2.4</i>

Table 11: ABET KPIs

## REFERENCES

- Allison, P. R. (2016, July 15). What does a bomb disposal robot actually do.
- AmandaDattalo. (2018, 8 8). *ROSIntroduction*. Retrieved from Wiki.Ros:  
<http://wiki.ros.org/ROS/Introduction>
- Arm. (2022). *Loom*. Retrieved from Loom Web site:  
<https://www.loom.com/share/661354c8321b4b5b9d78158a7ceeda87>
- DiyIoT. (2020). Retrieved from DiyIoT: <https://diyio.com/arduino-mega-tutorial/#:~:text=With%20a%209V%20power%20supple,current%20draw%20of%2016.74mA>.
- Duddu, P. (2020). TALON Tracked Military Robot. *Army Technology*.
- ENERDAN. (2001). *Lithium-Ion Battery*. Max-Planck Str. 3, 12489 Berlin: ENERDAN.
- Gardner, E. (2020, January 20). Bomb disposal robots: the new frontier.
- Harb, H. (2021, June 2). Retrieved from Youtube:  
[https://www.youtube.com/watch?v=CgeQVWac-HY&list=PLjwYOLz0Mk31Tmijz9McGbeVFicIe7XFU&index=39&ab\\_channel=HassanHariri](https://www.youtube.com/watch?v=CgeQVWac-HY&list=PLjwYOLz0Mk31Tmijz9McGbeVFicIe7XFU&index=39&ab_channel=HassanHariri)
- How URDF Models and 3D Models Can Help Your Next Robotics Project*. (n.d.). Retrieved from STANLEY INNOVATION : [https://stanleyinnovation.com/urdf-models-3d-models-robotics-projects/#:~:text=The%20URDF%20\(Universal%20Robot%20Description,looks%20like%20in%20real%20life](https://stanleyinnovation.com/urdf-models-3d-models-robotics-projects/#:~:text=The%20URDF%20(Universal%20Robot%20Description,looks%20like%20in%20real%20life).
- ICOR Technology. (2019). CALIBER® T5: Compact, Two-Man Portable System.
- Katranji. (2022). Retrieved from Katranji: <https://www.katranji.com/Item/438398>
- Mazzari., V. (2015, February 26). *Robotic simulation scenarios with Gazebo and ROS*. Retrieved from Génération Robots : <https://www.generationrobots.com/blog/en/robotic-simulation-scenarios-with-gazebo-and-ros/#:~:text=Gazebo%20is%20a%203D%20simulator,%2C%20gravity%2C%20inertia%2C%20etc>.
- McElhinney, K. (2019, September 18). *Simple robot arm controlled by ROS and MoveIt* . Retrieved from Ken McElhinney - My Blog:  
<http://ken.mcelhinney.net.au/servoarm/servoarm-simple-robot-arm-part-2/>
- NORTHROP GRUMMAN. (2016). *Mini-Andros II - Hazardous-Duty Mobile Robot*.
- Pi, R. (2018). *Raspberry Pi 3 Model A+*. Raspberry Pi Foundation.
- ProgrammerSought. (2022). *ProgrammerSought*. Retrieved from ProgrammerSought:  
<https://www.programmersought.com/article/54031154905/>
- Sears-Collins, A. (2020, June 24). Retrieved from Automatic Addison:  
<https://automaticaddison.com/what-is-the-difference-between-rviz-and-gazebo/>
- Seed Studio*. (2022). Retrieved from <https://www.seedstudio.com/TS100-shock-absorber-tank-chassis-with-track-and-DC-g geared-motors-Kit-p-4107.html>
- Simulation, F. R. (2022, April 12). *Loom* . Retrieved from Loom:  
<https://www.loom.com/share/da85414e52bc4b73be2377b7ad5a1bec>
- Store, D. (2022). Retrieved from Ali Express:  
[https://www.aliexpress.com/item/32994424086.html?spm=a2g0o.detail.0.0.1594314f19X5Rx&gps-id=pcDetailBottomMoreThisSeller&scm=1007.13339.169870.0&scm\\_id=1007.13339.16](https://www.aliexpress.com/item/32994424086.html?spm=a2g0o.detail.0.0.1594314f19X5Rx&gps-id=pcDetailBottomMoreThisSeller&scm=1007.13339.169870.0&scm_id=1007.13339.16)

- 9870.0&scm-url=1007.13339.169870.0&pvid=6e76cac5-f65a-41a9-96fa-d25eb7266e1e&\_t=gps-id:pcD
- Store, H. (2022). Retrieved from Ali Express:  
[https://www.aliexpress.com/item/1005002565843684.html?spm=a2g0o.search0304.0.0.6d6f1f64HK3jjm&algo\\_pvid=ee4196ec-4391-436f-8251-486fb81ddaa7&algo\\_exp\\_id=ee4196ec-4391-436f-8251-486fb81ddaa7-2](https://www.aliexpress.com/item/1005002565843684.html?spm=a2g0o.search0304.0.0.6d6f1f64HK3jjm&algo_pvid=ee4196ec-4391-436f-8251-486fb81ddaa7&algo_exp_id=ee4196ec-4391-436f-8251-486fb81ddaa7-2)
- Store, W. (2022). Retrieved from Ali Express: <https://www.aliexpress.com/i/32837087900.html>
- Sweidan, A. (2021, June 2). Retrieved from YouTube:  
[https://www.youtube.com/watch?v=WCH3ViS64yM&list=PLjwYOLz0Mk31Tmijz9McGbeVFicIe7XFU&index=38&ab\\_channel=HassanHariri](https://www.youtube.com/watch?v=WCH3ViS64yM&list=PLjwYOLz0Mk31Tmijz9McGbeVFicIe7XFU&index=38&ab_channel=HassanHariri)
- Tank. (2022). *Loom*. Retrieved from Loom Web site:  
<https://www.loom.com/share/d3039cf1b6714efda72b8cf6ff2f4ff4>
- US Department of Justice. (2004, July). Vanguard Robot Assessment. *IN SHORT* .
- Wajih. (2021, September 13). Retrieved from Youtube:  
[https://www.youtube.com/watch?v=DwbjKCWdDFU&list=PLjwYOLz0Mk31Tmijz9McGbeVFicIe7XFU&index=33&ab\\_channel=HassanHariri](https://www.youtube.com/watch?v=DwbjKCWdDFU&list=PLjwYOLz0Mk31Tmijz9McGbeVFicIe7XFU&index=33&ab_channel=HassanHariri)
- Wasenmuller, O., & Stricker, D. (2018). *Comparison of Kinect v1 and v2 Depth Images in Terms of Accuracy and Precision*. German Research Center for Artificial Intelligence (DFKI).
- Yoon, A. (2010). Kinect teardown: two cameras, four microphones, 12 watts of power, no controller. *Engadget*.