Technical University of Crete

School of Electrical And Computer Engineering

Development of an Autonomous Hexapod Robot

Dimosthenis Myrkos

Thesis committee

Professor Michail G. Lagoudakis (ECE)

Professor Michalis Zervakis (ECE)

Professor Panagiotis Partsinevelos (MRE)

Chania, July 2022

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη ενός Αυτόνομου Εξάποδου Ρομπότ

Δημοσθένης Μύρκος

Εξεταστική Επιτροπή

Καθηγητής Μιχαήλ Γ. Λαγουδάκης (ΗΜΜΥ)

Καθηγητής Μιχάλης Ζερβάκης (ΗΜΜΥ)

Καθηγητής Παναγιώτης Παρτσινέβελος (ΜΗΧΟΠ)

Χανιά, Ιούλιος 2022

Abstract

Nowadays, autonomous six-legged robots show great promise for use in applications, where rough, unstructured terrain has to be traversed. Legged robots have advanced potential to navigate in complex environments compared with their wheeled counterparts. The subject of this diploma thesis is the development (hardware and software) of an autonomous hexapod (six-legged) robot for safe navigation and exploration in unknown environments. This thesis presents the design of the robot, the derivation of the kinematic chain of the joints and the development of its gait planning algorithm. In particular, the design of the robot consists of all the necessary components, such as sensors, 3D printing parts, batteries and the integrated control system. In addition, Simultaneous Localization and Mapping (SLAM) techniques are used, in order to produce the map of the unknown environment and determine the location of the robot within that map. Finally, the autonomous navigation system of the hexapod robot for its movement in unknown environments is described in detail. The entire project has been implemented in the Robot Operating System (ROS) framework and is available as an open source package. The proposed robotic system has been tested in indoor environments and in real-time conditions.

Περίληψη

Στις μέρες μας, τα αυτόνομα εξάποδα ρομπότ έχουν παρουσιάσει μεγάλη πρόοδο σε εφαρμογές, όπου απαιτείται να διασχίσουν ανώμαλα και ανισόπεδα εδάφη. Τα αρθρόποδα ρομπότ γενικά υπερτερούν στην πλοήγηση σε ιδιαίτερα πολύπλοκα περιβάλλοντα σε σύγκριση με τα αντίστοιχα τροχοφόρα ρομπότ. Αντικείμενο της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη (υλικό και λογισμικό) ενός αυτόνομου εξάποδου ρομπότ για την ασφαλή πλοήγηση και εξερεύνηση άγνωστων περιβαλλόντων. Η εργασία παρουσιάζει τον σχεδιασμό του ρομπότ, τον υπολογισμό της κινηματικής αλυσίδας των αρθρώσεων και την ανάπτυξη του αλγορίθμου βηματισμού του. Ειδικότερα, ο σχεδιασμός του ρομπότ αποτελείται από όλα τα απαραίτητα στοιχεία, όπως αισθητήρες, εξαρτήματα τρισδιάστατης εκτύπωσης, μπαταρίες και το ενσωματωμένο σύστημα ελέγχου. Επιπρόσθετα, γίνεται χρήση τεχνικών Simultaneous Localization and Mapping (SLAM), προκειμένου να παραχθεί ο χάρτης του άγνωστου περιβάλλοντος και να προσδιοριστεί η θέση του ρομπότ μέσα σε αυτόν. Τέλος, περιγράφεται λεπτομερώς το σύστημα αυτόνομης πλοήγησης του εξάποδου ρομπότ με σκοπό την μετακίνησή του σε άγνωστα περιβάλλοντα. Η συνολική εργασία έχει υλοποιηθεί στο πλαίσιο του Robot Operating System (ROS) και είναι διαθέσιμη ως πακέτο ανοιχτού κώδικα. Το προτεινόμενο ρομποτικό σύστημα έχει δοκιμαστεί σε εσωτερικά περιβάλλοντα και σε συνθήκες πραγματικού χρόνου.

Acknowledgements

First of all, I would like to thank my professors Michail Lagoudakis and Panagiotis Partsinevelos for their guidance and support throughout this work. My family, for their constant faith and support. My colleagues and friends in SenseLab and especially, Angelos Antonopoulos for the incredible help that he has given to me.

Contents

List of Figures

List of Tables

List of Abbreviations

ROS	**R**obot **O**perating **S**ystem
IMU	**I**nertia **M**easurement **U**nit
LIDAR	**L**aser **I**maging **D**etection **A**nd **R**anging
KF	**K**alman **F**ilter
EKF	**E**xtended **K**alman **F**ilter
SLAM	**S**imultaneous **L**ocalization and **M**apping
DoF	**D**egrees **O**f **F**redom
RVIZ	**R**OS **V**isualization

Chapter 1

Introduction

Nowadays, a big step has been taken toward the study of legged robots, such as the study of biped robot, quadruped robot, hexapod robot and octopod robot. Legged hexapod robots are programmable robots with six legs attached to the robot body. The legs are controlled with a degree of autonomy, so that the robot can move within its environment to perform intended tasks. Hexapod robots can be suitable for terrestrial and space applications and they can include features, such as omnidirectional motion, variable geometry, good stability and access to diverse terrain. Hexapod robots can have a wide spread use in exploration of hostile environments and remote locations. For instance, space or planetary exploration [1], seabed research [2], nuclear power stations applications [3] and search and rescue operations [4].

In comparison to tracked and wheeled forms of transport legged locomotion is not restricted to rough or uneven parts of terrain and is capable of choosing the best placement for the foothold. Hexapod walking robots also benefit from a lower impact on the terrain and have greater mobility in natural surroundings. This type of locomotion with gait planning have the capabilities to climb over obstacles larger than the equivalent sized wheeled or tracked vehicles.

Despite the above referenced aspects, legged locomotion have many disadvantages such as energy consumption, complex kinematics, dynamic mechanisms and difficult control algorithms. Due to active suspension of legged systems which should support legs in continuous matter, the payload of these systems is considerably lower than other two kinds of locomotion.

Autonomous mobile robots are intelligent machines capable of performing tasks in the world by themselves, without explicit human control over their movements[5]. They are programmed to act and make decisions based on sensory feedback of their external and/or internal surroundings.

Autonomous navigation has always been an interesting and challenging problem in robotics. For any mobile robot, the ability to autonomously navigate in its environment is crucial to complete any kind of task. Being able to avoid obstacles and/or find a safe route towards the target location in an unknown environment is critical in many applications.

1.1 Thesis Contribution

This thesis describes the development of a six-legged robot to autonomously navigate in an unknown environment. Implements different nature inspired gait algorithms for its locomotion. Utilizes a plethora of sensors which provide the system with the necessary information of its surrounding world. Incorporates a navigation system for autonomous transverse in the world. The entire project has been implemented within the Robot Operating System (ROS) and is supported for any six legged robot system, as long as proper adjustments and sensors are acquired.

The proposed approach can be used in a variety of missions such as mapping an unknown terrain where human intervention can be dangerous.

1.2 Thesis Outline

- **Chapter 2 - Theoretical Background:** We present all the background information needed for this thesis.It contains explanation of the kinematic analysis of the legs. It also contains basic knowledge about robot localization and mapping. Moreover, there will be a sensor model explanation as well as the usability of the ROS middleware in robotics.
- **Chapter 3 - Problem Statement:** We reference the basic problems of the thesis and also similar robotic projects.
- **Chapter 4 - Approach:** We describe the approach and the steps we took to accomplish the goal of this thesis. Starting, with the design and assemble of the robot and finishing with gait algorithm implementation.
- **Chapter 5 - Results:** We present the results of the project in real world environments.
- **Chapter 6 - Conclusion** Conclusions and future plans for extending our approach are presented.

Chapter 2

Theoretical Background

2.1 Robot Operating System

The Robot Operating System (ROS) [6] is an open-source middleware framework for modular use in robot applications. ROS, originally designed and developed by Willow Garage, is currently maintained by the Open Source Robotics Foundation. The primary advantage of ROS is that allows multiple devices to coexist and interacts with each other through a peer-to-peer network. ROS uses the concepts of nodes, topics, transforms, publishers and subscribers among many. This is a concept where nodes, which are processes that perform a particular computation or task, are combined together in a graph like structure. These nodes talk to each other using message buses which carry messages from one node to other. Nodes are separate entities and can be combined with other nodes that subscribe messages from it or publish messages for it. The publishing and subscribing mechanism are anonymous mechanisms where the only information needed is the type of message being published or subscribed without knowing the underlying idea of how those messages are generated or communicated.

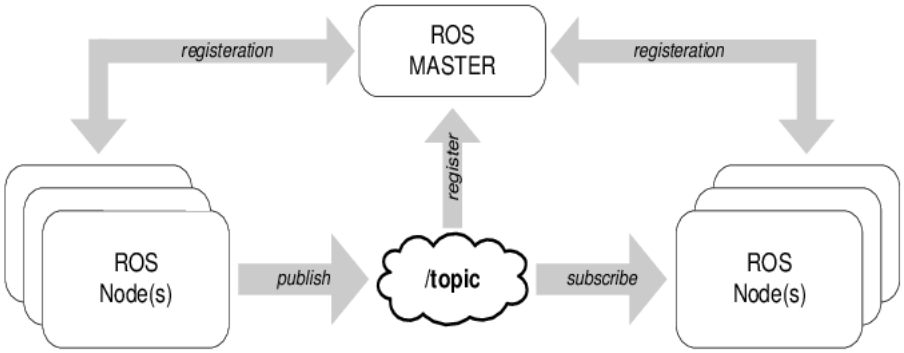

FIGURE 2.1: ROS Communication diagram, source Erick Mejia Uzeda Github

2.1.1 Rviz

Rviz, abbreviation for ROS visualization, is a 3D visualization tool for ROS applications. It provides a view for our robot model, capture sensor information from robot sensors, and replay captured data.Furthermore, it can display data from camera, lasers and point clouds.Thus, this makes it ideal for representation of our robot configuration and how it perceives its surroundings.

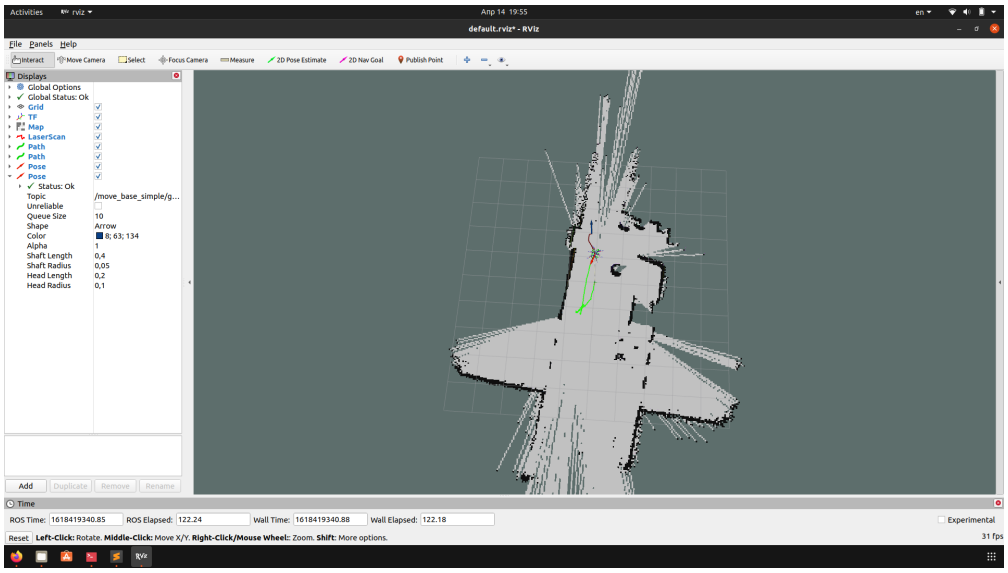

FIGURE 2.2: ROS Rviz tool

2.2 Sensors

Sensors are essential for the robot to perceive its surroundings navigate with safety and complete any objectives assigned.

2.2.1 LIDAR

LIDAR, which stands for Light Detection and Ranging primerly used in many robotic application for navigation obstacle avoidance and general spatial information. Lidar is a method in which the system emits a pulsed laser light and captures the reflected light back to receiver utilizing the principle of time-of-flight (2.1) in order to give distance measurements. There is a variety of lidar sensors such as 1D, 2D, 3D.

$$d = \frac{Speed\,of\,light * Time\,of\,flight}{2} \tag{2.1}$$

The most common use is the 2D scanning lidar witch utilizes the same technical concept . The sensor is mounted on a rotary motor making it possible to take continuous samples as it is rotating. Scanning lidar sensors are also equipped with rotary encoders to be able to map a laser sample to a rotary angle. By doing this it is possible to acquire a point cloud of samples represented in the plane. The angle between two consecutive samples is called azimuth or angular resolution and can for some lidar sensors be tuned manually by adjusting the sampling and rotational frequency of the device, as can be seen in equation (2.2). Furthermore, the term sample will relate to a single point in a point cloud and a sample frame will refer to a full point cloud.

$$azimuth = \frac{f_{sampling}}{f_{rotation} * 360} \tag{2.2}$$

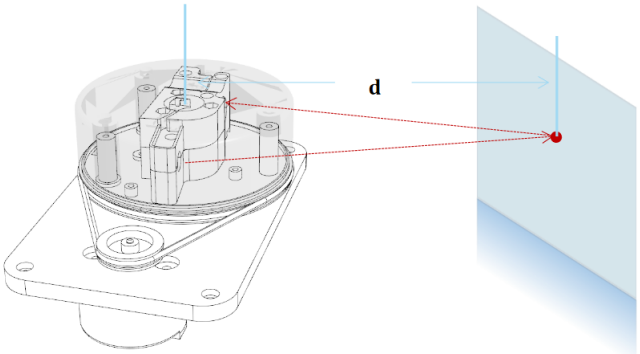

FIGURE 2.3: LiDAR pricipal of distance measure

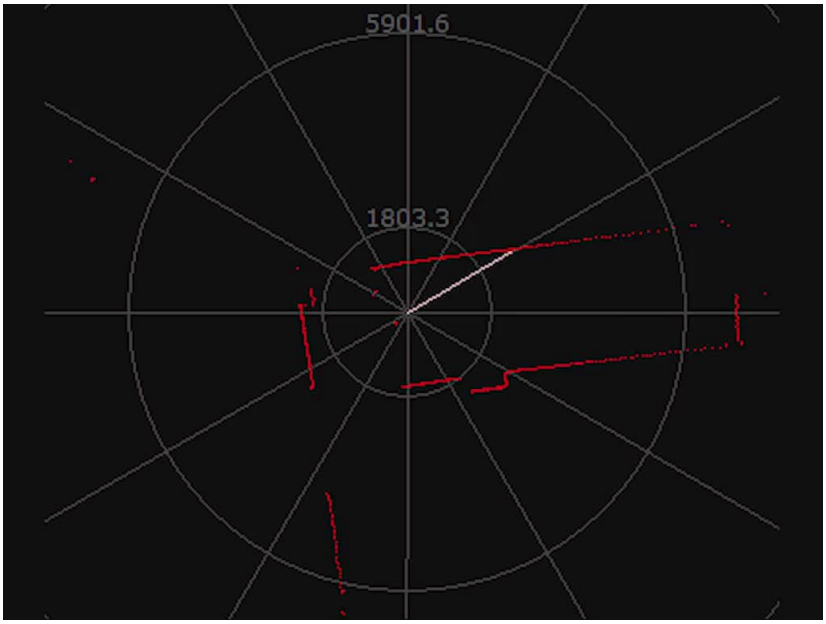

FIGURE 2.4: LiDAR sample scan

2.2.2 Inertia Measurement Unit

The inertial measurement unit (IMU) is an electronic device, a sensor that measures and reports accelaration,orientation, angular rates and other gravitational forces. It is comprise of three gyroscopes, three accelerometers and depending of the heading requirement three magnetometers. This is to say, one per axis for each of the robot axis(roll,pitch,yaw)(2.5).The data measured by the inertial sensors usually have certain errors. One type of error is offset error,

which means that accelerometers and gyroscopes have nonzero outputs even when they are stationary. In the process of using the IMU, displacement data is usually needed. Integrating the output of the accelerometer twice can get linear displacement, and integrating the output of the gyroscope once can get angular displacement. However, as time goes by, errors will accumulate so that great errors will be produced. The background noise in the working environment will also interfere with the measurement results. Therefore, in actual use, the data output by the IMU must be filtered and corrected, which can usually be achieved by using a Kalman filter (KF). In fact, in many application fields, it is difficult to obtain accurate results using the data output by the IMU alone, so the IMU usually needs to perform data fusion with other sensors to obtain better results. The KF can be also used for data fusion.

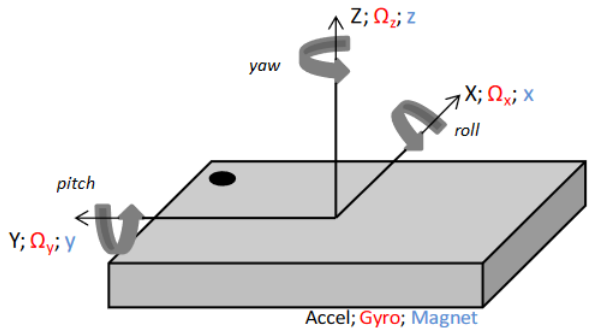

FIGURE 2.5: BNO055 coordinates system

2.2.3 Ultrasonic Distance Sonar

Another way to detect and avoid obstacles with low cost sensors is with ultrasonic distance sonars. The sensor sends an ultrasonic pulse out at 40kHz which travels the air and if there is an obstacle or object, it will bounce back to the receiver. By calculating the travel time and the speed of sound, the distance can be calculated.Ultrasonic sensors are a great solution for the detection of clear objects. Also, they can detect objects regardless of the color,surface or material which that makes them quiet reliable.

$$Distance = \frac{time\,of\,flight}{speed\,of\,sound * 2} \tag{2.3}$$

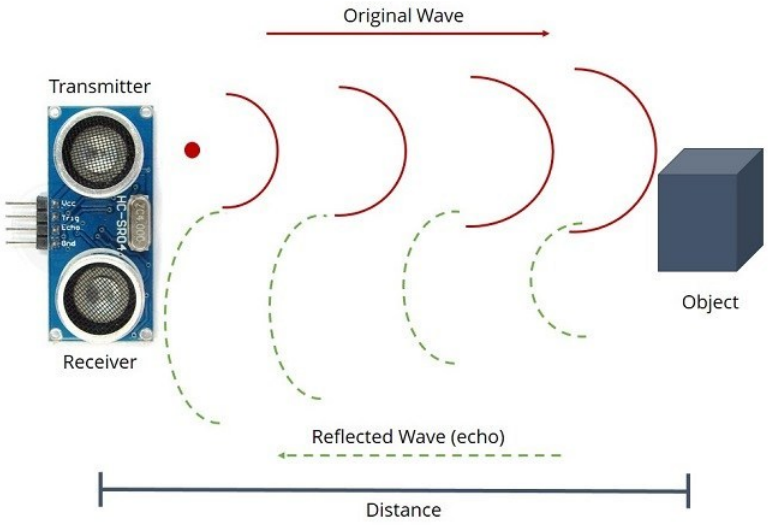

FIGURE 2.6: Ultrasonic sensor (hc-sr04)

2.3 Localization and Mapping

2.3.1 Robot Localization

The problem of robot localization consists of answering the question *Where am I?* from a robots point of view. This means the robot has to find out is location relatively to the environment, namely establish the correspondence of its local coordinate system to the global (map coordinate system), in order to express the location of any surrounding object or obstacle in relation to it. The general localization problem has number increasingly difficult problem instances.

- **Position tracking:** In the position tracking, we assume that the initial robot pose is known and we solve the localization problem through the odometry information. By compensating the robot's motion model uncertainty, we update the local robot pose belied in relation with the initial pose. Hence, this method rely on the assumption that the robot's motion model error is small and can be approximated.
- **Global localization:** The problem of the robot unawareness of its initial position in a known map and the need of locating itself in it. The robot must exploit various observation from its sensors, to infer its location in the environment and then proceed with the position tracking approach.

- **Kidnapped robot problem:** An even harder problem to solve is the kidnapped robot problem, the robot does exactly know where its is localized, but all of sudden it is transferred, or 'kidnapped' to another location without the robot being aware of this.The problem for the robot is to detect that it has been kidnapped and to find out what it is position in the map.
- **Dynamic environments** A problem that can be found in the real world where the environment can contain other moving objects, in this environments localization is significantly more difficult, since these other objects might confuse the robot about its location by corrupting the information.

A mobile robot can solve the localization problem by using two different classes of on-board sensors, proprioceptive sensors and extereoceptive sensors. Proprioceptive sensors, such as encoders or inertial measure units ,measure the motion of the robot, acquiring data that can be integrated to estimate relative robot displacement. This method of localization is called odometry or dead reckoning, and when used alone, the integrated error in global position grows without bound over time. Extereoceptive sensors, such as laser range scanners or cameras take measurements from the external environment. This data can be correlated at subsequent robot positions to compute relative pose or displacement estimates, which can improve and sometimes replace odometry. Externally sensed data may also be correlated with data from a global map giving a global position measurement and bounding the overall position error. If a global map is not initially available , it is possible for the robot to build a global map with the externally sensed data while using this map to localize this approach is commonly called Simultaneous Localization and Mapping (SLAM).

2.3.2 Occupancy Grid Mapping

Occupancy Grid Mapping refers to a family of computer algorithms in probabilistic robotics for mobile robots which address the problem of generating maps from noisy and uncertain sensor measurement data, with the assumption that the robot pose is known. The basic idea of the occupancy grid is to represent a map of the environment as an evenly spaced field of uniformly-distributed binary/ternary variables indicating the status of cells (occupied, free or undetected). Besides, as a practical instrument for environmental understanding, the occupancy grid map is very useful for integrating different sensor measurements (radar, LiDAR, vision system) into a unified representation.

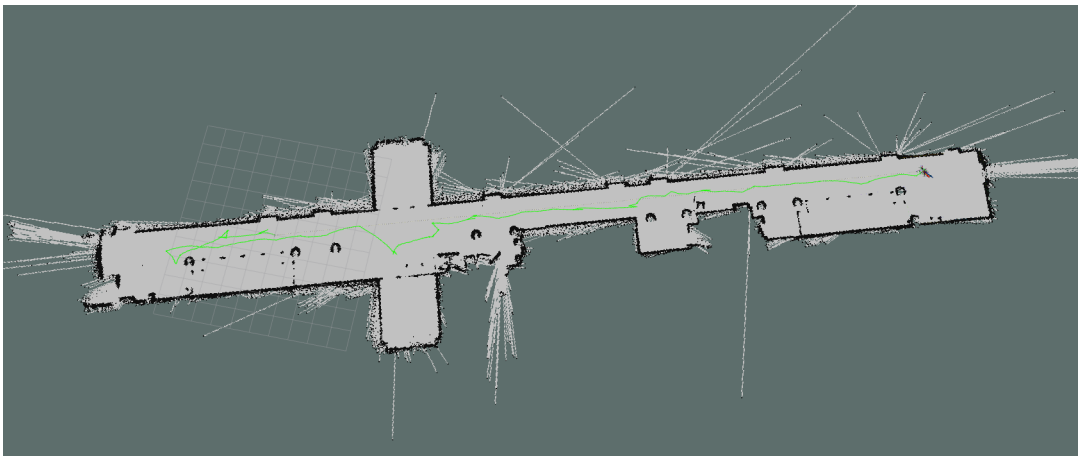

FIGURE 2.7: Occupancy grid map in Mineral Resources Engineering building in the Technical University of Crete

2.3.3 Simultaneous Localization and Mapping (SLAM)

Simultaneous Localization and Mapping (SLAM) is a process by which a mobile robot can build a map of an environment and at the same time use this map to deduce its location. In SLAM, both the trajectory of the platform and the location of all landmarks are estimated online without the need of any a priori knowledge of location. The SLAM problem can be define as the problem of building a map of the environment while simultaneously determining the robot's position relative to this map given noisy data.

There are two main forms of SLAM problem , the full SLAM and the online SLAM which are both of equal practical importance. In particular,

- The full SLAM it involves estimating the posterior over the entire robot path together with the map

$$p(X_t, m|Z_t, U_t). \tag{2.4}$$

- The online SLAM seeks to recover the present robot location, instead of the entire path and can be define via

$$p(x_t, m|Z_t, U_t). \tag{2.5}$$

  where x_t is the location of the robot and $X_t = (x_0, x_1, x_2, ..., x_T)$ the sequence of locations or *path*. m refers to the map . Let u_t denote the odometry that characterized the motion between time $t-1$ and t and $U_t =$

$(u_1, u_2, u_3, ..., u_T)$ the sequence of this. Finally, the $Z_t = (z_1, z_2, z_3, .., z_T)$ is the sequence of measurements between features in m and the robot location x_t

2.4 Robot Locomotion

The locomotion of a legged robot generally simulate a human or animal gait in our case of a hexapod robot is inspired of insects.The first problem we have to solve is the kinematics formula of each leg and finally the gait algorithm we have to incorporate. Kinematics is the mathematical process of calculating the variable joint parameters needed to place the end of a kinematic chain in a given position and orientation relative to the start of the chain shown in Figure 2.8. The hexapod robot, which we are working on has 3 Degrees of Freedom (DoF) for each leg, totally 18 (DoF). Its a biologically inspired design based on spiders anatomy. Each leg has three servomotors.

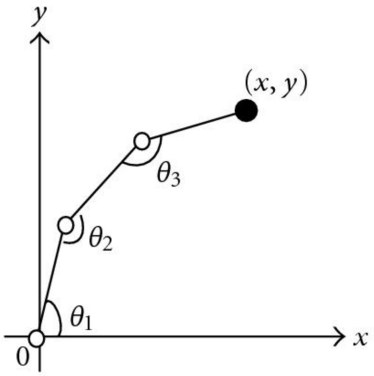

FIGURE 2.8: Inverse Kinematics of 3DoF model

2.4.1 Kinematics Analysis

First of all, we simplify the architecture of the robot in 7 modules, a rectangular trunk and 6 limps between the trunk and the ground as shown in the Figure 2.9. Two coordinates frames are assigned. The first is (O) on the ground, and the second on the trunk (O'). Forward motion is considered as in the direction of X' .

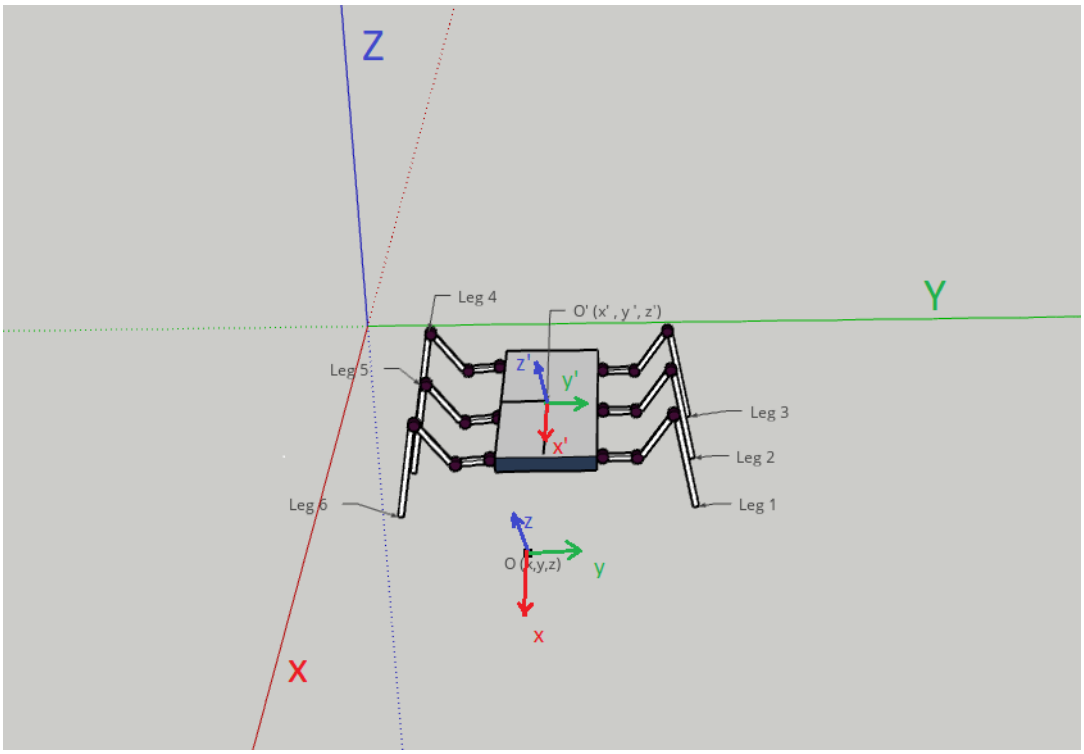

FIGURE 2.9: Hexapod coordinate frame assignment ,ground frame (O) and trunk frame (O')

The main idea of solving inverse kinematics of this robot came from a modular view of floating trunk and serial kinematics chain. To obtain the kinematic chain and find out the joint variables, we define a homogeneous transformation matrix to transform the Leg's tips from the ground frame to trunk coordinate frame. This transformation matrix can be written as below:

$$ {}^{O'}_{O}T = \begin{bmatrix} R_z(\theta_z)R_y(\theta_y)R_x(\theta_x) & -OO' \\ 0 & 1 \end{bmatrix} \tag{2.6} $$

R_z, R_y and R_x are rotational transformation matrices around Z, Y and X respectively and OO' is the distance from O' to O. In coordinate frame O', $Leg_1..Leg_i..Leg_6$

Forward and Inverse Kinematic Analysis of one Leg

Each leg can be seen as a serial manipulator where its base is fixed on the trunk and its end point on the ground. For forward kinematic analysis, using Denavit-Hartenberg parameters of one leg [7][8], we can write transformation matrix of each joint, based on frames shown in Figure 2.10 and Table 2.1. Three

joints and five frames are defined from the initialized frame to the end point of the leg. The homogeneous transformation matrices of leg links based on DH parameters in table one are presented as below:

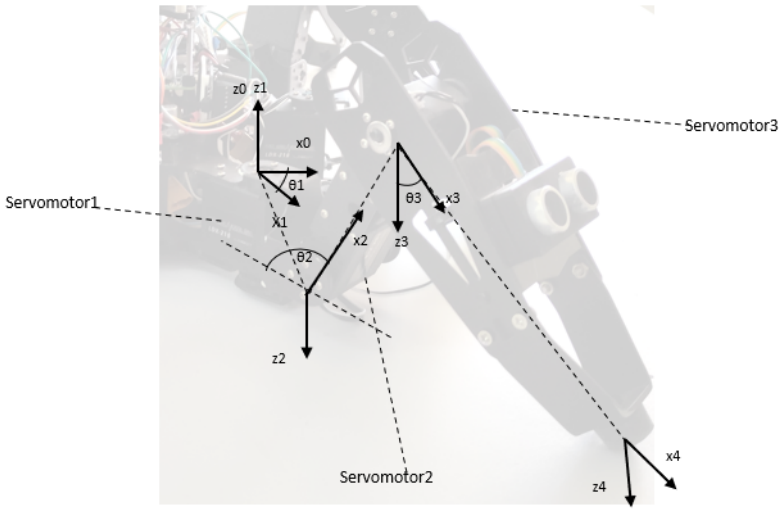

FIGURE 2.10: Leg frame points for finding Denavit-Hartenberg parameters

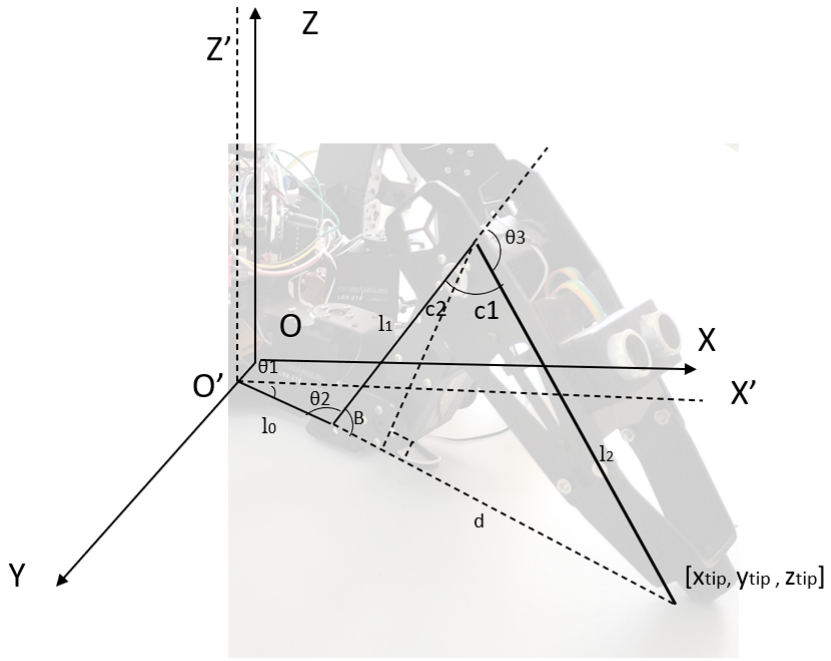

FIGURE 2.11: 3 DoF hexapod leg link assignment and parameters for inverse kinematic analysis of one leg

$$
{}^{0}_{4}T_i = \begin{bmatrix} cos_{1,2}cos_1 & -sin_{1,2}cos_1 & -sin_1 & cos_1(l_0 + l_1cos_2 + l_2cos_{2,3} \\ cos_{1,2}sin_1 & -sin_{1,2}sin_1 & cos_1 & sin_1(l_0 + l_1cos_2 + l_2cos_{2,3} \\ -sin_{2,3} & -cos_{2,3} & 0 & -l_1sin_2 + l_2sin_{2,3} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7}
$$

where $sin_1, sin_{1,2}, sin_{2,3}, cos_1, cos_{1,2}, cos_{2,3}$ stands for $\sin\theta_1, \sin(\theta_1 + \theta_2)$, $\sin(\theta_2 + \theta_3), \cos\theta_1, \cos(\theta_1 + \theta_2)$ and $\cos(\theta_2 + \theta_3)$ respectively. ${}^{0}_{4}T_i$ transforms points from end point which is the leg's tip to the base coordinate frame. The position of the leg tip in *X'Y'Z'* coordinate frame can be found using homogeneous transformation matrix from base coordinate frame to endpoint coordinate

frame. The reason that O' is defined adjacent to O is the vertical distance between the shaft of servomotors 1 and 2.

$$\begin{bmatrix} x_{tip_i} \\ y_{tip_i} \\ z_{tip_i} \\ 1 \end{bmatrix} = {}^{0}_{4}T_i \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.8}$$

$$x_{tip_i} = \cos\theta_{1_i}(l_0 + l_1\cos\theta_{2_i} + l_2\cos(\theta_{2_i} + \theta_{3_i})) \tag{2.9}$$

$$y_{tip_i} = \sin\theta_{1_i}(l_0 + l_1\cos\theta_{2_i} + l_2\cos(\theta_{2_i} + \theta_{3_i})) \tag{2.10}$$

$$z_{tip_i} = -l_1\sin\theta_{2_i} + l_2\sin(\theta_{2_i} + \theta_{3_i}) \tag{2.11}$$

where x_{tip_i},y_{tip_i} and z_{tip_i} are the leg tip posiotion in O' and θ_1,θ_2 and θ_3 are the joint angles. Based on Figure 2.9, Inverse Kinematics equations for one leg can be written as :

$$\theta_{1_i} = \arctan 2(y_{tip_i}, x_{tip_i}) \tag{2.12}$$

$$d_{1_i} = \sqrt{(x_{tip_i} - l_0\cos\theta_1)^2 + (y_{tip_i} - l_0\sin\theta_1)^2 + z_{tip_i}^2} \tag{2.13}$$

$$B_i = \arccos(\frac{d_i^2 + l_1^2 - l_2^2}{2l_1d_i}) \tag{2.14}$$

$$\theta_{2_i} = \arcsin(\frac{-z_{tip_i}}{d_i}) - B_i \tag{2.15}$$

$$C1_i = \arccos(\frac{l_i\sin B_i}{l_2}) \tag{2.16}$$

$$C2_i = \frac{\pi}{2} - B_i) \tag{2.17}$$

$$\theta_{3_i} = \pi - C1_i - C2_i \tag{2.18}$$

Link	a_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	-π/2	l_0	0	θ_2
3	0	l_1	0	θ_3
4	0	l_2	0	0

TABLE 2.1: Denavit-Hartenberg parameters of 3 DoF hexapod leg

Chapter 3

Problem Statement

3.1 Autonomous Navigation

In order to achieve complete autonomy we are called to solve two main problems. The primary problem we have to solve is how the robot need to be able to construct the map of their surrounding environment and also localize themselves in it. Secondly it must have the ability to explore unknown terrains while maintain system stability and dynamically avoids obstacles, without human interference.

3.2 Gait management/Locomotion of a Legged Robot

A crucial problem is how a legged system can stand and move using the architecture of its legs efficiently. We have to consider a pattern of steps that can maintain a balance between stability, power consumption and velocity. One of the main challenges lies in the difficulty of controlling the multi-legs of the robots with coordination to a complex dynamic environment.

3.3 Related Work

One of the industry leading legged robot is SPOT, the robot dog developed by Boston Dynamics, shown in Figure 3.1. Spot is a compact, nibble four-legged robot. Spot is designed to be rugged and customizable it's well suited for unstructured environments and is fully capable of climbing stairs and traversing rough terrain. It has a 360 degrees perception that allows him to avoid obstacles. Spot can be used in a wide variety of applications, in hazardous or non-hazardous areas.

FIGURE 3.1: Boston Dynamics SPOT, source www.therobotreport.com

Chapter 4

Approach

4.1 Robot Construction / Hexapod

4.1.1 Hexapod frame

The robot‘ s frame manufacturing company is Lobot shown in Figure (4.1). The only parts that was retained from the original hexapod is the body frame and the leg mechanisms. This implies the number of Degrees of Freedom (DOF) per leg remains fixed to three. For our project purposes we had to add an embedded system for brain like function, sensors, power management and servo drivers.

FIGURE 4.1: lobot-cr6

4.1.2 Embedded System

Arguably, one of the most important part of the system. We decided to use a raspberry pi 4 model, an embedded ARM architecture computer with exceptional capabilities for our intensive algorithms and ROS setup. Is equipped with a quad-core Cortex-A72 64bit SoC at 1.5Ghz clock speed, 8GB LPDDR4 memory, 2.4Ghz and 5.0Ghz IEEE 802.11b/g/n/ac wire-less, LAN, Bluetooth 5.0, BLE, Gigabit Ethernet, 2 USB 3.0 ports, and 2 USB 2.0 ports. You can see the model we used at Figure 4.2.

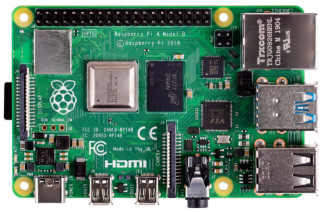

FIGURE 4.2: Raspberry Pi 4

4.1.3 Servo Drivers and Servos

The robot utilizes 18 servo motors for locomotion.The servos we used were the LDX-218 Figure 4.3 with full metal gear and up to 17kg torque depended on the voltage (15kg 6.6v - 17kg 7.4v). Control angle of 180 degrees. To drive the servos we choose the Adafruit PCA9685 chip. The driver offers an onboard PWM controller which has the capability of handling up to 16 servos with no additional processing overhead from the raspberry. Our final product used two drivers chained together. In Figure 4.4 is the aforementioned driver.

FIGURE 4.3: Hiwonder servo LDX-218

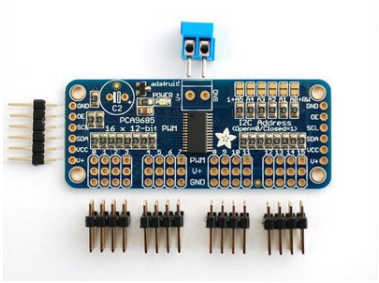

FIGURE 4.4: Adafruit PCA9685

4.1.4 Sensors

For our robot to be able to perceive it's surroundings and to autonomous navigate in the them we implemented different sensors in the design. Firstly, the Rplidar A1, which is a low cost 360 degree 2d laser scanner (LIDAR) developed by SLAMTEC Figure 4.5. The system can perform 360 degree scan within 6 meter range. The produced 2D point cloud data were used in mapping and localization. Rplidar A1 has scanning frequency of 5.5hz when sampling 360 points each round and can be configured up to 10hz. Additionally, we implemented four ultrasonic ranging modules in the sheen of the legs(two forward and two backward) to achieve better coverage in lower obstacles than of the LIDAR's height. The modules were the HC-SR04 Figure 4.6, which have low cost and provides a working range of 2cm-400cm and working angle < 15 degrees. Finally, the use of an IMU was necessary for safely navigating and calculate the location and path of the robot. We choose the bosch BNO055 sensor Figure 4.7, because it provides a triaxial 14-bit accelerometer, an accurate close-loop triaxial 16-bit gyroscope, a triaxial geomagnetic sensor and a 32-bit microcontroller running the Bosch Sensortec sensor fusion software. The on-board fusion facilitates the computations needed by the raspberry pi and make a more robust end product.

FIGURE 4.5: SLAMTEC RPLIDAR A1

FIGURE 4.6: Ultrasonic range module, HC-SR04

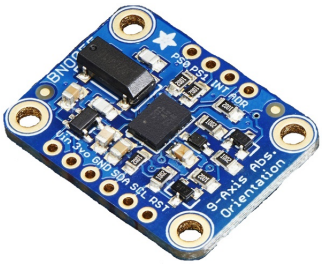

FIGURE 4.7: BOSCH IMU , BNO055

4.1.5 Power management

In order for our robot to be autonomous and capable of covering large areas, the use of a battery was essential. The size of the frame was limited so two batteries were fitted in parallel. The batteries we choose was Gen ace high discharge rechargeable 7.4V Lipo 5.300mAh. The raspberry has a 5V input limitation, thus we used a step down DC-DC converter.

FIGURE 4.8: Left: Li-po rechargeable battery. Right: DC-DC converter

4.1.6 3D printed parts

For the housing of certain sensors we needed to 3D print some parts to assure stability and optimal position.In Figure 4.9 we present the final designs.

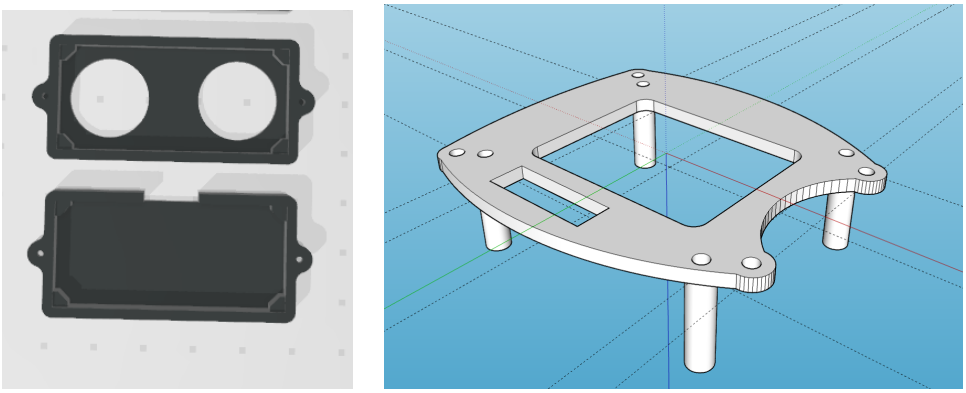

FIGURE 4.9: Left: Case for HC-SR04 ultrasonic sensor. Right: Rplidar A1 mount

Finally, we present the block diagram of the robot system as shown in Figure 4.10 and the final product after assembling the aforementioned parts is shown in Figures 4.11, 4.12.

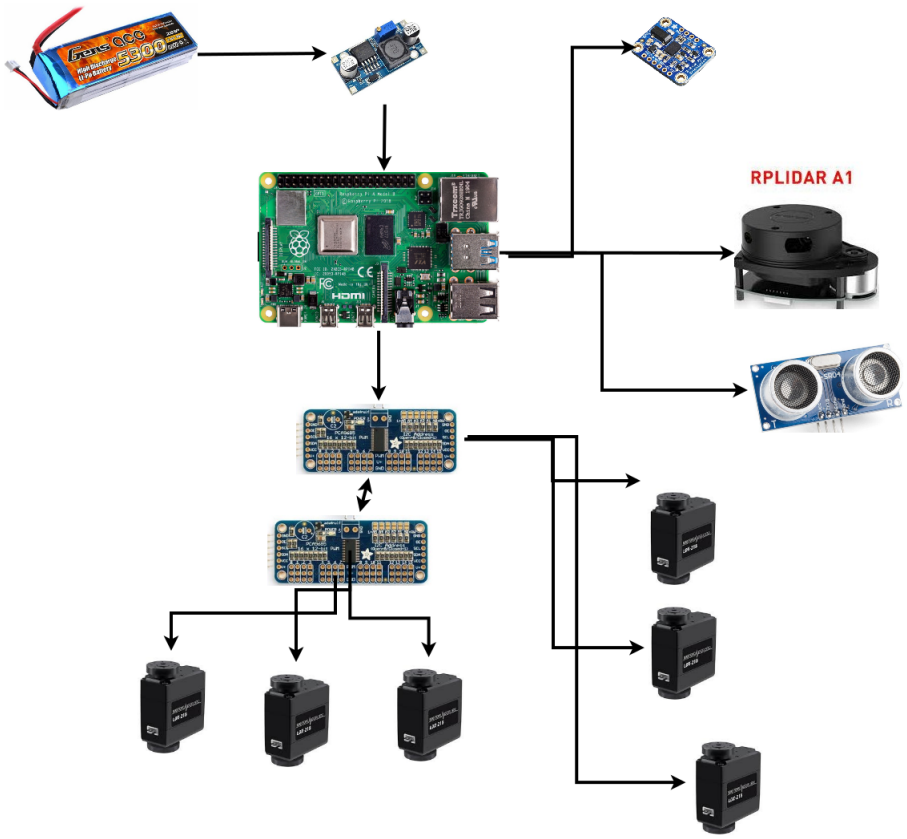

FIGURE 4.10: block diagram of the hexapod

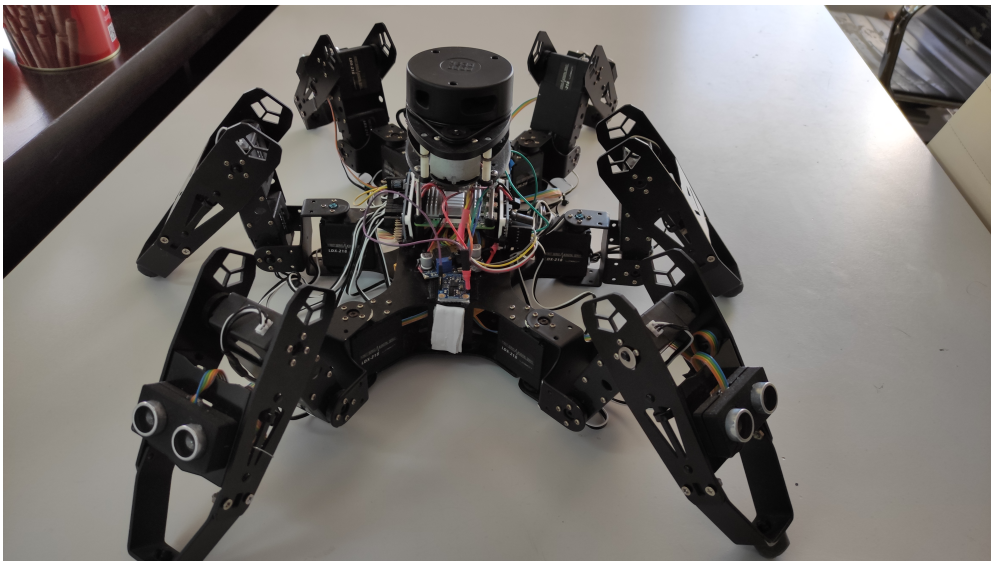

FIGURE 4.11: Complete assembly of the robot

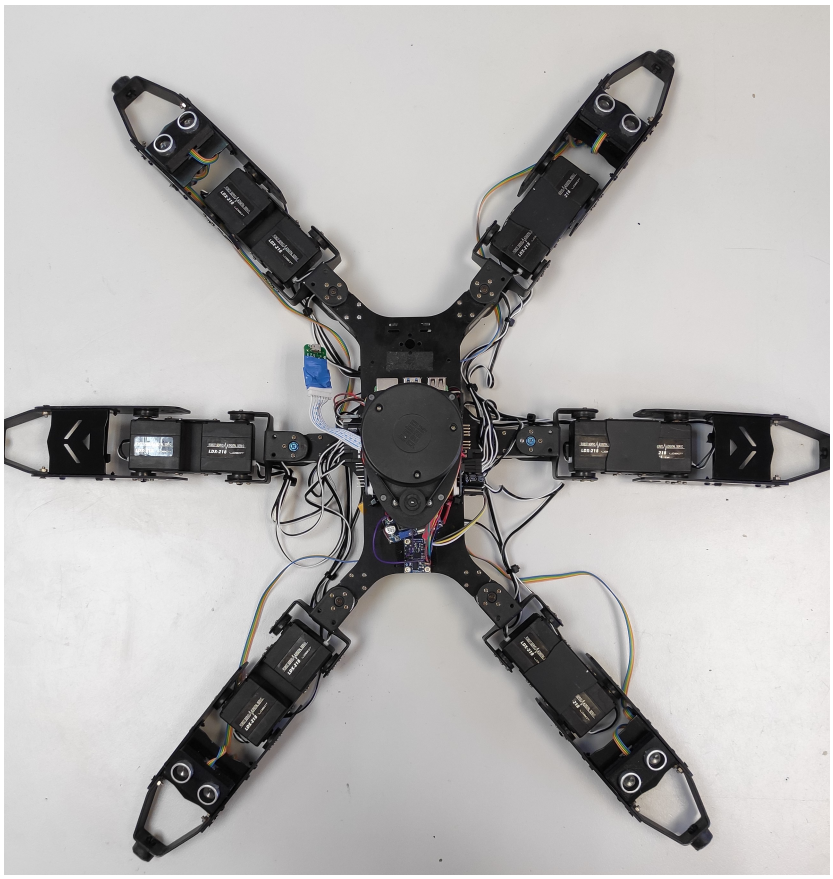

FIGURE 4.12: Complete assembly of the robot top view

4.2 ROS coordinate system and tf2 library

Robots consists of many parts, joints and sensors. Thus, is essential for the robot to know where it is itself as well where the rest of the world is in relation to itself.With the use of ROS integrated transform library, the ***tf2***, which can maintain the relationship between coordinate frames in a tree structure buffered in time, and lets the user transform points, vectors, etc between any two coordinate frames at any desired point in time.[9] We created a dedicated ROS node (appendix **??**) to provide the hexapod static transforms,including the various sensor frames. We precisely measured the sensor position in reference to hexapod body point (assumed the base frame origin) and included the resulted real world positions into the tf tree. By this way, the tf tree contains frame transformations described in real world units, and hence, any processed measurement and extracted results will be spatially described in the same way. The hexapod tf tree node is illustrated in Figure 4.13 and its structure visualization in the RViz plug-in, shown in Figure 4.14.

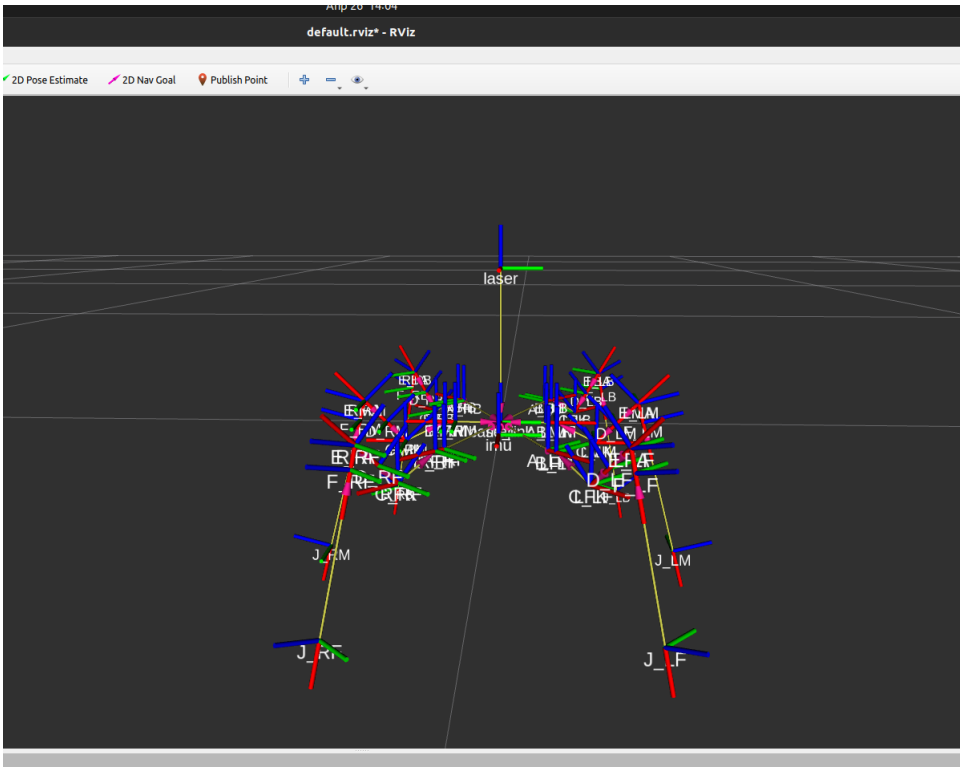

FIGURE 4.13: Views of tf tree in Rviz

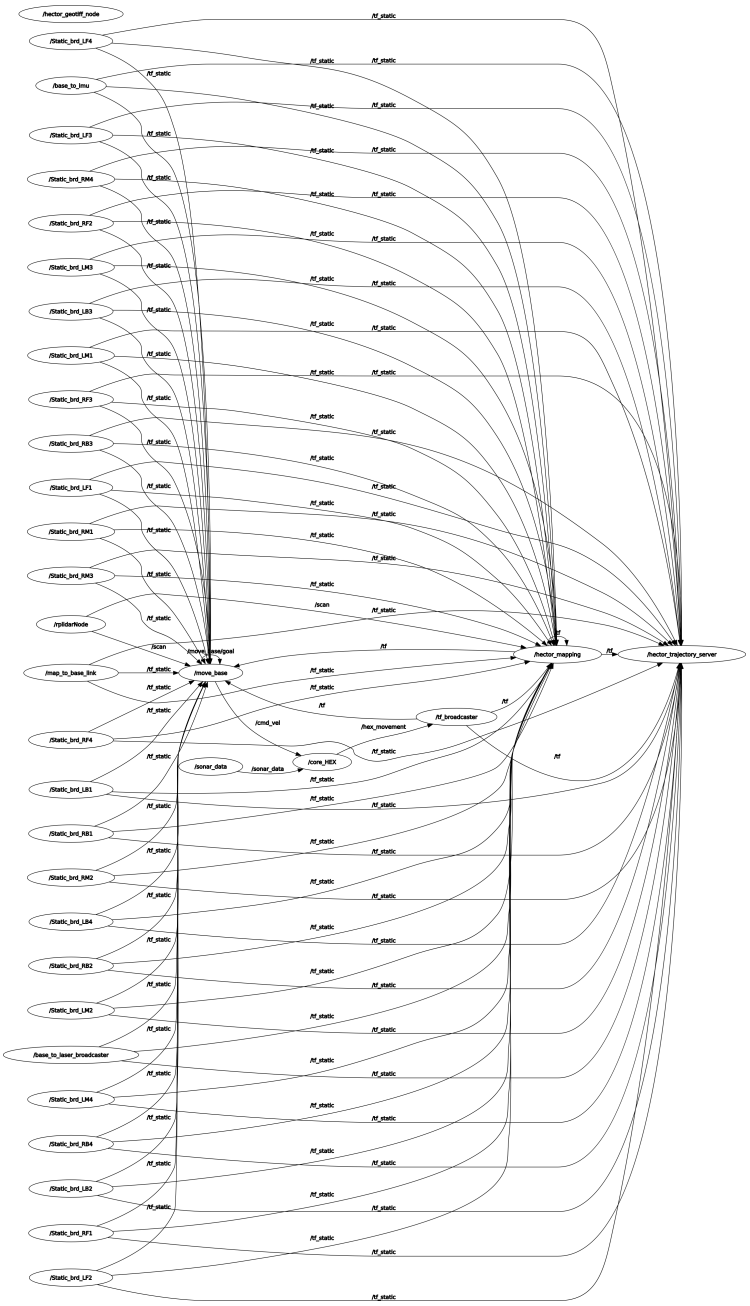

FIGURE 4.14: Hexapod system Tf tree

In robotics applications, many different coordinate systems can be used to define where robots, sensors, and other objects are located. In general, the location of an object in 3-D space can be specified by position and orientation values.

4.3 Hector SLAM Algorithm

During the navigation, the LiDAR system might exhibit 6DoF motion, so its laser scans are transformed through hexapod pose estimation and the tf tree information, into the hexapod's stabilized base frame coordinate system. Every laser scan is converted into a point cloud of scan endpoints, expressed in the base frame coordinate system. The implemented LiDAR data node transforms laser scans data into point cloud form, hence, we filter the outliers to enhance SLAM procedure.

We chose Hector SLAM algorithm because of the low error estimation and that it doesn't require odometry data. Hector SLAM is a 2D SLAM system dependent on laser scans. Like many other SLAM systems, hector SLAM uses occupancy grid map approach to perform SLAM. It also utilizes scan matching process to align laser scans with an existing map. The first laser scan is written to the map and subsequent scans are matched with that map. It uses a Gauss-Newton approach explained in to find transformations that gives the best alignment of laser scans with the map. LIDAR scans are written to the map depending on criterions of minimum displacement in translation or rotation relative to the location of the previous map writing. Hector SLAM uses a Gauss-Newton approach and presents optimized rigid transformations as

$$\xi^* = argmin_\xi \sum_{i=1}^{n} [1 - M(S_i(\xi)]^2 \tag{4.1}$$

Where $\xi = (p_x, p_y, \phi)^T$ is the rigid transformation needed and $S_i(\xi)$ denotes the world coordinates of the scan end point $s_i = (s_{i,x}, s_{i,y})^T$. Solving for $\Delta\xi$ yields the Gauss-Newton equation for the minimization problem,

$$\Delta\xi = H^{-1} \sum_{i=1}^{n} [\nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi}]^T [1 - M(S_i(\xi))] \tag{4.2}$$

with Hessian matrix ,

$$\mathbf{H} = [\nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi}]^T [\nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi}] \tag{4.3}$$

the previous two equations 4.2, 4.3 depend on the gradient of the map $\nabla M(P_m)$ at coordinate P_M which is calculated by using a bi-linear filtering. The gradient $\nabla M(P_m)$ can be approximated by using the four closest integer coordinates

$P_{00}..P_{11}$ as depicted in Figure 4.15. The derivatives can be approximated by :

$$\frac{\partial M}{\partial x}(P_m) \approx \frac{y - y_o}{y_1 - y_0}(M(P_{11}) - (M(P_{01})) + \frac{y_1 - y}{y_1 - y_0}(M(P_{10}) - (M(P_{00})) \quad (4.4)$$

$$\frac{\partial M}{\partial y}(P_m) \approx \frac{x - x_o}{x_1 - x_0}(M(P_{11}) - (M(P_{01})) + \frac{x_1 - x}{x_1 - x_0}(M(P_{10}) - (M(P_{00})) \quad (4.5)$$

Where x_0, x_1, y_0, y_1 are coordinates of P as shown in figure 4.15. Finally, the scan match uncertainty can be calculated as the inverse Hessian times some scaling constant σ whitch depends on the properties of the laser scanner

$$R = Var[\xi] = \sigma^2 H^{-1} \quad (4.6)$$

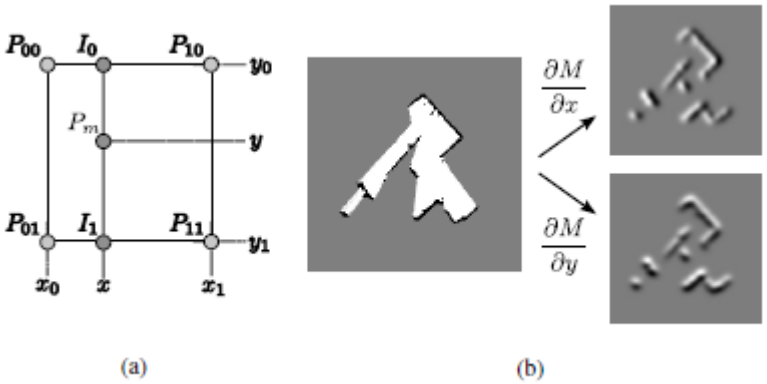

FIGURE 4.15: (a) Bilinear filtering of the occupancy grid map. Point P_m is the point whose value shall be interpolated. (b) Occupancy grid map and spatial derivatives. [10]

4.4 Navigation

For the navigation of the robot we chose to take two approaches. First, with *way point* approach meaning that we give a single point in the map that the robot has to reach. Second, to just explore the environment until there is no more to explore.

4.4.1 Move Base Package

For the autonomous navigation we utilized the **move_base** package [11] provided by ROS ecosystem. The package comprises of five main parts. These are the global and local costmaps,the global and local planners and the recover behaviors. The global costmap takes in the map of the hexapod's environment and inflates any objects on that map by a user specified radius. This inflated area is used as a buffer zone around objects that the path cannot intersect with in order to prevent a collision. The local costmap is used to inflate objects detected by the sensors in a small area around the robot. Both costmaps use the **costmap_2d** package [12] and feed into the recovery behaviors, which is used to determine the robots action in case is stuck. The global planner is used to plan a path from the robots start location to the goal location. It uses the global costmap to ensure that the path does not intersect with any obstacle. There are multiple global planners in ROS. We decided to use the **global_planner** package [13].

The global planner utilizes different algorithms for finding the shortest path. We decided to use the A* algorithm because of its better performance compare to Dijkstra's algorithm by taking advantage heuristics to guide its search. Finally, for the local planner we had to chose between the **base_ local_ planner** and the **dwa_local_planner** (**d**ynamic **w**indow **a**pproach). Both were tested and it was determined that **dwa_local_planner** was more suitable for our project based on the limited acceleration the system outputs[14]. The **dwa_ local_ planner** package provides a controller that drives a mobile base in the plane. This controller serves to connect the path planner to the robot. Using a map, the planner creates a kinematic trajectory for the robot to get from a start to a goal location. Along the way, the planner creates, at least locally around the robot, a value function, represented as a grid map. This value function encodes the costs of traversing through the grid cells. The controller's job is to use this value function to determine dx,dy,dtheta velocities to send to the robot.[15]

Additionally, the **move base** node provides a recovery behavior which is crucial for the robot to get unstuck or engage in a different route. By default, the move base node will take the following actions to attempt to clear out space: First, obstacles outside of a user-specified region will be cleared from the robot's map. Next, if possible, the robot will perform an in-place rotation to clear out space. If this too fails, the robot will more aggressively clear its map, removing all obstacles outside of the rectangular region in which it can rotate in place. This will be followed by another in-place rotation. If all this fails, the robot will

consider its goal infeasible and notify the user that it has aborted as shown in Figure 4.17.

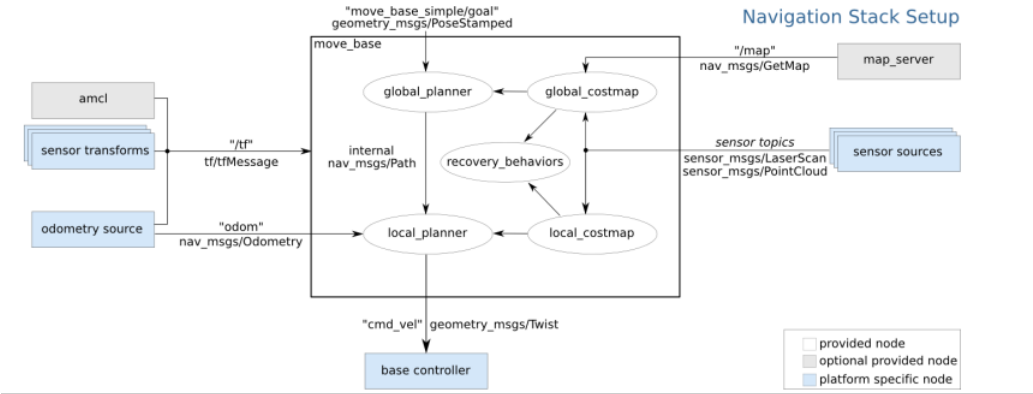

FIGURE 4.16: move base package default navigation stack[11]

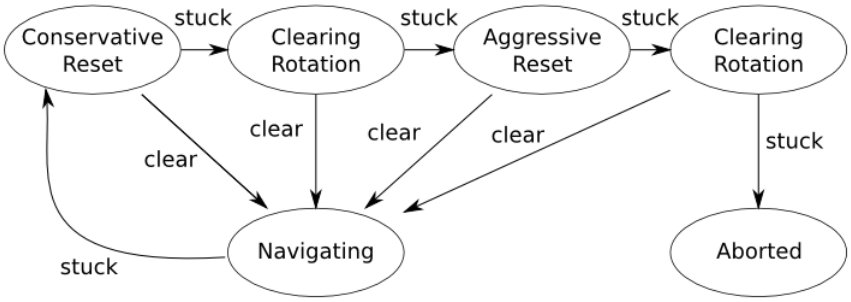

FIGURE 4.17: Move Base Recovery Behavior [11]

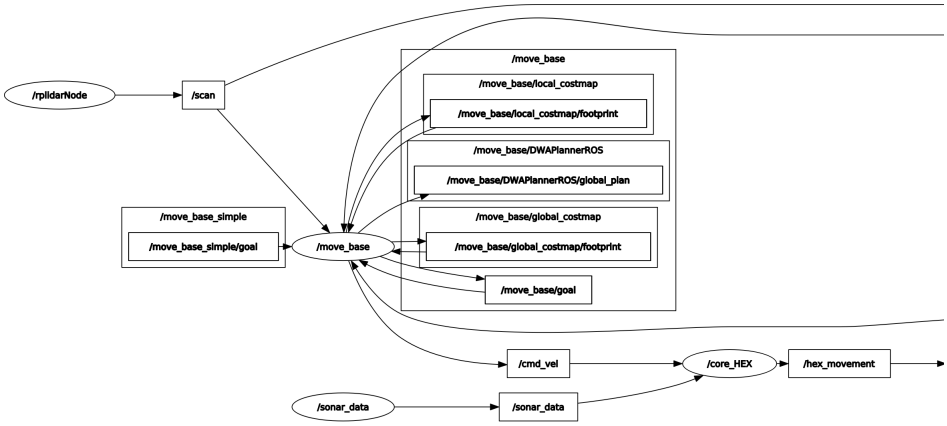

FIGURE 4.18: implementation of navigation stack on the hexapod

4.5 Gait models

Gait is a periodic pattern of motion which defines the locomotion of the system through a sequence of steps, where each step comprise of return stroke and power stroke of the legs. Power stroke is when the leg is on the ground, supporting the body and return stroke is when the leg lifts of moving forward from the body.

We have examined and implemented three known gait archetypes, the wave gait, tripod gait and ripple gait each with certain pros and drawback. At any instance, the system should ensure that at least three legs are in contact with the ground for stability.

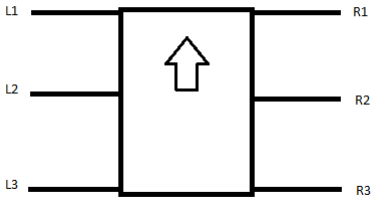

FIGURE 4.19: Leg positions of the robot

4.5.1 Wave Gait

The wave gait is the slowest of all three gaits as it requires only one leg to swing at a time. This gait has a cyclic pattern of six steps, where in each step one leg lifts off the ground to swing forward while the other five legs keep contact with the ground and provide propulsion for the system.

The only advantage of the wave gait is the stability of the system. However, in the real world application even when the terrain requires stability,the gait is so slow and energy inefficient that renders it impracticable.

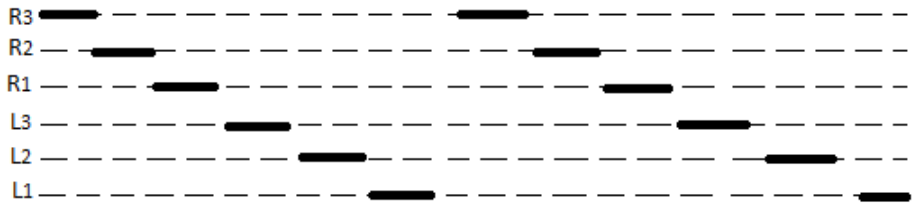

FIGURE 4.20: Diagram of Wave gait foot pattern

4.5.2 Ripple Gait

The ripple gait follows a cyclic pattern of six steps, where the mechanics of the gait are move one leg while the moving leg is on the power phase, the diametrically opposite starts its return phase in order to remain stable.

Although, the ripple gait is little faster than the wave gait and retains the stability factor, remains inefficient as it still transverses really slowly the terrain.

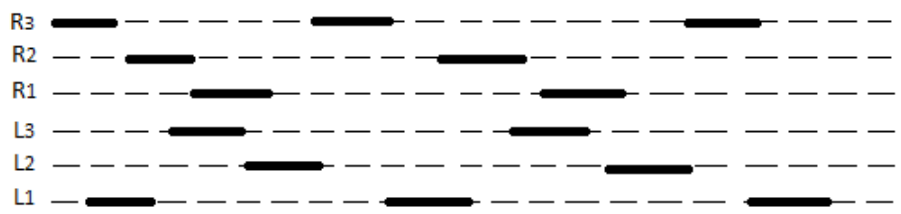

FIGURE 4.21: Diagram of Ripple gait foot pattern

FIGURE 4.22: From top left to right the steps of the ripple gait

4.5.3 Tripod Gait

The tripod gait of a hexapod robot comprises of two step cycle. In the first step, three legs of the system which are not all on one side of the robot, lift up and begin to swing forward, while the other three legs keep contact with the ground and help propel the robot. In the next step the three legs make ground contact and the other set of three legs begin to swing forward.

After we experimented with all three gaits we came to the conclusion that the tripod gait is the better of the three, because it provides stability while improves the velocity of the system.

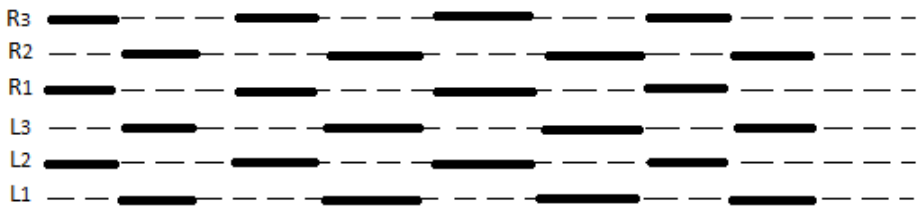

FIGURE 4.23: Diagram of Tripod gait foot pattern

FIGURE 4.24: From top left to right the steps of the tripod gait

Chapter 5

Results

5.1 Final version of the robot

After numerous changes in the design of the robot, we conclude at the final version of the robot as shown in Figures 5.1, 5.2. The final product has the lowest possible weight and the possible best weight distribution.

FIGURE 5.1: final version hexapod robot

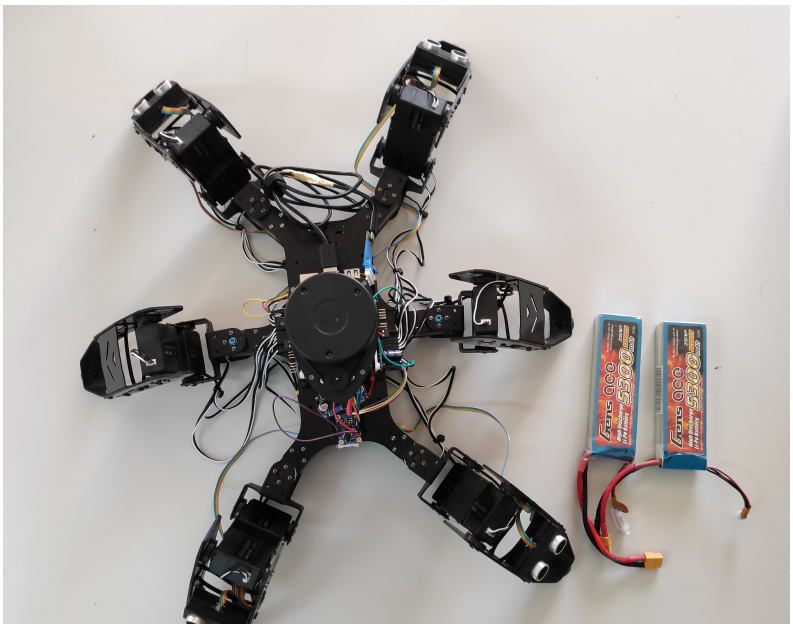

FIGURE 5.2: final version hexapod robot static

5.2 Autonomous Navigation Experiments

Our system was tested in two different environments. The first environment, was a populated room with various in height and length objects (Figures 5.3,5.4). The second was a more open corridor with makeshift obstacles and ledges (Figures 5.6,5.7,5.8). In both environments the hexapod navigate without any collision and creating the optimum path.

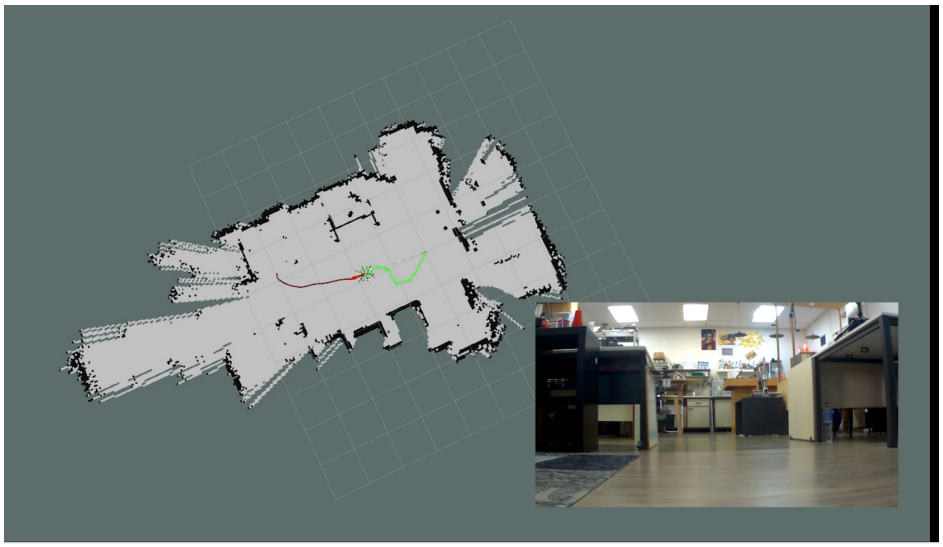

FIGURE 5.3: SenseLab room experiment

FIGURE 5.4: SenseLab room experiment

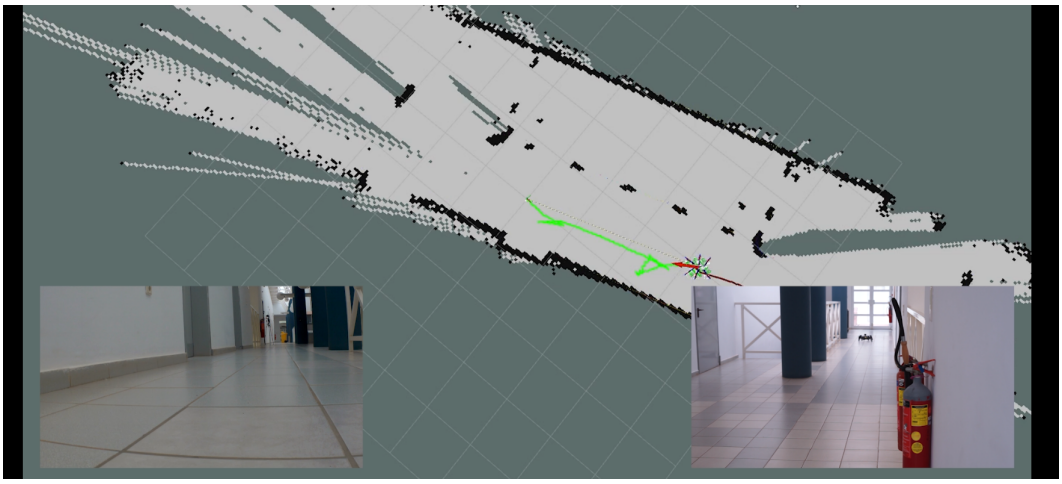

FIGURE 5.5: Open corridor experiment

FIGURE 5.6: Open corridor with obstacles

FIGURE 5.7: Open corridor with obstacles

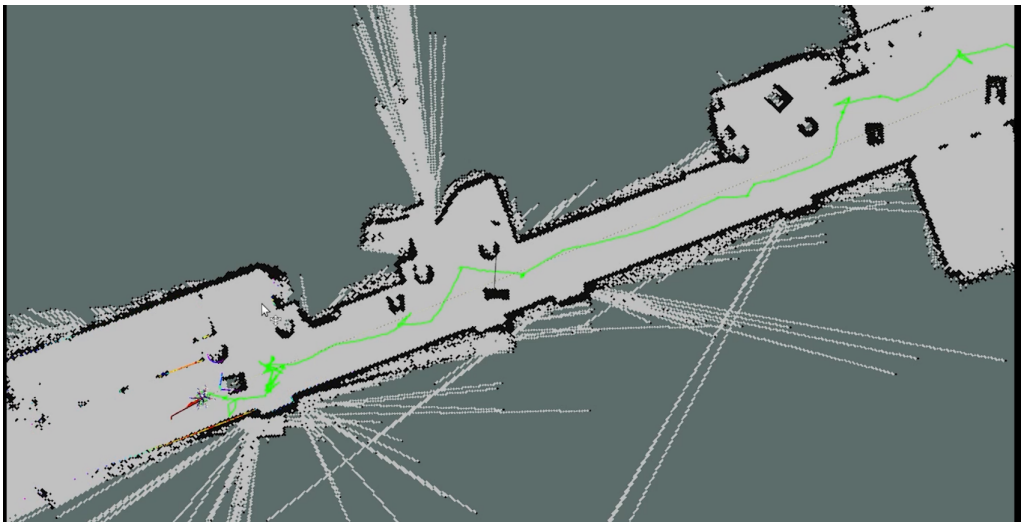

FIGURE 5.8: Open corridor with obstacles Map

Chapter 6

Conclusion

6.1 Conclusions

This thesis undertakes a difficult task of developing a hexapod robot and autonomously navigate in an unknown environment. At first, we had to implemented bio-mechanical and nature inspired gait algorithms for the locomotion of the robot. Utilizing the laser scan data and imu the robot was able to perform SLAM in order to create a map of the environment and localize in it. Finally, a custom implementation of a navigation stack was performed to achieve autonomous navigation and exploration of the terrain. Each system part is presented in detail and the entire system is validated with real-world experiments. The entire project has been implemented within the Robot Operating System (ROS) and is available as an open source package.

6.2 Future Work

6.2.1 Machine Learning Gait Generation

In our approach, we implemented three different gait algorithms each one with its own pros and cons. By utilizing the machine learning techniques, we can create a Reinforcement learning algorithm in order to discover a new gait pattern that minimizes the disadvantages that the gait may have.

6.2.2 Camera approach

A 3D camera can be added in the robot for object detection and human recognition. This way the system can be developed for search and rescue scenarios. On of the main advantage the hexapod robot has over the wheeled counterpart is that it can transverse uneven terrains that makes it suitable for scenarios

such as caves or debris of buildings. The camera module except the recognition also can give feed of the place for better reconnaissance.

References

[1] M Pavone, P Arena, and L Patané. "An innovative mechanical and control architecture for a biomimetic hexapod for planetary exploration". In: *Space Technology-Abingdon* 26.1-2 (2006), pp. 13–24.

[2] Bong-Huan Jun et al. "Development of seabed walking robot CR200". In: *2013 MTS/IEEE OCEANS-Bergen*. IEEE. 2013, pp. 1–5.

[3] T G Bartholet. "Robot applications for nuclear power plant maintenance". In: (Mar. 1985). URL: https://www.osti.gov/biblio/5705520.

[4] Jia Uddin. "An Intelligent Hexapod Rescue Robot". In: 2018.

[5] George A Bekey. *Autonomous robots: from biological inspiration to implementation and control*. MIT press, 2005.

[6] *ROS*. URL: https://www.ros.org/.

[7] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. "A kinematic notation for lower-pair mechanisms based on matrices". In: *Journal of Applied Mechanics* 22.2 (1955), 215—221.

[8] Richard P Paul. *Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators*. Richard Paul, 1981.

[9] Wim Meeussen Tully Foote Eitan Marder-Eppstein. *Tf2*. URL: http://wiki.ros.org/tf2.

[10] Stefan Kohlbrecher et al. "A flexible and scalable SLAM system with full 3D motion estimation". In: *2011 IEEE international symposium on safety, security, and rescue robotics*. IEEE. 2011, pp. 155–160.

[11] Eitan Marder-Eppstein. *move base*. URL: http://wiki.ros.org/move_base.

[12] Dave Hershberger Eitan Marder-Eppstein David V. Lu!! *costmap-2d*. URL: http://wiki.ros.org/costmap_2d.

[13] Author: David Lu!! *global-planner*. URL: http://wiki.ros.org/global_planner.

[14] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. "The dynamic window approach to collision avoidance". In: *IEEE Robotics & Automation Magazine* 4.1 (1997), pp. 23–33.

[15] Eitan Marder-Eppstein. *dwa-local-planner*. URL: http://wiki.ros.org/dwa_local_planner.