Zagazig University, Faculty of Engineering,
Computer and Systems Engineering Dept.

ZagHexa
Design, Construction and Control of a Hexapod Walking Robot

B.Sc. Graduation Project
Submitted by

Mahmoud Mohammed Elsayed
Khaled Mohammed Risha
Mohammed Alaa Mohammed
Hend Khairy Abdelhamed
Nehad Abdelsalam Mohammed
Amira Elsayed Soliman

July, 2017

Submitted to The Computer and Systems Engineering Dept., Faculty of Engineering, Zagazig University, Egypt

Graduation Project Report to be submitted to
Zagazig University, faculty of Engineering
in partial fulfillment of the requirements for the degree
Bachelor of Science in Engineering (B.Sc.)
©2017

Date of Presentation
24. Juni 2015

Supervisors
Dr.Ing. **Mohammed Nour Abdel Gwad Ahmed**
Computer and Systems Engineering Department,
Faculty of Engineering, Zagazig University, Egypt

Defense Commete
Prof. Dr. Xyz Wuv
Prof. Dr. Abc Def

to
All, **Whom**, **we**, and **love**.
The best in my life and after ...

Abstract

This report is a documentation of the final year graduation project in electrical engineering at zagazig university. The purpose of this project is to Design, Construction and Control of a six-legged walking robot that is capable of basic mobility tasks such as walking forward, backward, rotating in place and raising or lowering the body height.

The legs are of a modular design that have three degrees of freedom each. This robot will serve as a platform onto which additional sensory components could be added, or which could be programmed to perform increasingly complex tasks.

The components that make up our final design are discussed. Also, we describe the basic robot gaits of locomotion for efficient navigation. This locomotion is tuned to make the robot faster and at same time energy efficient to navigate and negotiate difficult terrain.

The robot is an integrated multi-legged walking robot based on de-facto standard Robotic Operating System (ROS) that employs novel and different walking patterns.

Our robot is teleoperated using hand-held devices such as a smart phone or tablet or a wireless joystick. Furthermore, it has its own navigation system and a camera for instant video recording and streaming.

The power to the entire system is supplied through two 5 volts NiMH batteries. There is an additional power bank to power up the Raspberry Pi and other electronic components. We have an interactive website for robot inspection and online control in addition to leaning materials such as robot building and implementation walkthroughs and as well as step-by-setup tutorials.

Keywords – biologically inspired, legged robot, gait generation, design procedure, simulation

Acknowledgements

This graduation project consumed huge amount of work, research and dedication. Still, accomplishment would not have been possible if we did not have a support of many individuals. Therefore we would like to extend our sincere gratitude to all of them.

First of all, we would like to sincerely thank our advisors Dr.Ing. Mohammed Nour and Dr Ahmed Hamdy. They gave us the opportunity to work on great ideas with great people . When needing someone for the discussion of any problem, ... was always there and also solved a lot of ..... problems for/with us.

we are also grateful to our friends and colleagues, XYZ, ABC, and UVW. For getting into the basics of this project, we had a lot of support by XYZ who raised our first interest during the initial part of this work. ABC helped us a lot with the ..... and experiments. UVW helped so much in ..... We are indebted to ..... for making ..... easier to understand and to introduce ..... to us.

we express our warm thanks to ....., ..., and ..... for making the ..... robot available to us and the time they spend assisting us to carry out the field experiments. Without their superior knowledge and experience, the experiments would not have that like in quality of outcomes, and thus their support has been essential. In addition, we wish to express our sincere gratitude to ..... for helping when we had questions as well as frustrations.

We would like to thank all the numerous people in the internet who ask questions and provide answers for programming problems (specifically in ROS) and for the very useful code and tools they share with others.

we would like to most importantly acknowledge the effort of our families, who encouraged us to pursue higher education and support us through the difficulties associated with such a goal even when we was not sure we would make it through.

Last but not least, we would like to thank our friends for always encouraging us onwards. We can never thank them enough for their love and faith.

This work was supported by a financial aid from Zagazig University. We would like to thank the funders. They had no role in study design, data collection and analysis, decision to publish, or preparation of this work.

Contents

List of Figures

List of Tables

List of Algorithms

List of Symbols and Abbreviations

DOF	Degree of freedom
PWM	Pulse Width Modulation
I^2C	Inter-Integrated Circuit
Hexapod	six-leg walking robot
DMP	Digital Motion Processor
LiPo	Lithium Polymer
RPM	Revolution Per Minute
RPi	Raspberry Pi
GPIO	General-purpose input/output
ADC	Analog-Digital Converter
LPF	Low Path Filter
HPF	High Path Filter
FPS	Frame Per Seconds
GND	Ground

"There is nothing more difficult to take in hand, more perilous to conduct or more uncertain in its success than to take the lead in the introduction of a new order of things."

— Niccolo Machiavelli, (Italian writer and statesman, Florentine patriot, author of 'The Prince', 1469-1527)

Chapter 1
Introduction

In today's technological society, people have grown accustomed to daily use of several kinds of technology from personal computers to supercomputers, from personal vehicles to commercial airplanes, from mobile phones to communicating through the Internet and everything in between. Robotics technology has been a hot topic recent years. As such, the use of robots has become increasingly common. As robots can be used to complete repeated tasks, increase manufacturing production, carry extra weight and many other common tasks that humans do. Therefore, robots can be found everywhere. So far, all mobile robots used in extraterrestrial surface exploration missions were wheel-driven systems. However, even if such a system is equipped with a suspension system, the capability to surmount obstacles and to conquer steep inclinations is limited. Also driving on fine-grained soil can become a problem for these kind of systems. Multi-legged walking systems, in contrast, are equipped with a highly flexible locomotor system. Along with appropriate control strategies it should offer them the capability to securely maneuver in rough and steep environments. Major counter arguments for legged systems are the higher complexity regarding the mechanical design and control as well as the comparatively high power consumption. Thus, the challenge lies in minimizing these drawbacks and in exploiting the potentialities of such systems. One of the most important part of a robot is its chassis. There are several basic chassis types: wheeled, tracked and legged chassis. Wheeled chassis are fast, but not suitable for rough terrain. Tracked chassis are slower, but more suitable to rugged terrain. Legged chassis are quite slow and more difficult to control, but extremely robust in rough terrain. Legged chassis are capable to cross-large holes and can operate even after losing a leg [Saranlı, 2002]. Extensive research is conducted in this field because of its large potential. Legged chassis are especially ideal for space missions [terrain hex-limbed extra-terrestrial explorer, , Tedeschi and Carbone, 2014] . There are also several projects in military research [Dynamics, 2015b, Dynamics, 2015a].

One of the interesting features of hexapod robots such as our ZagHexa (shown in Fig.Figure 1.1) is that they can climb over obstacles larger than the equivalent sized

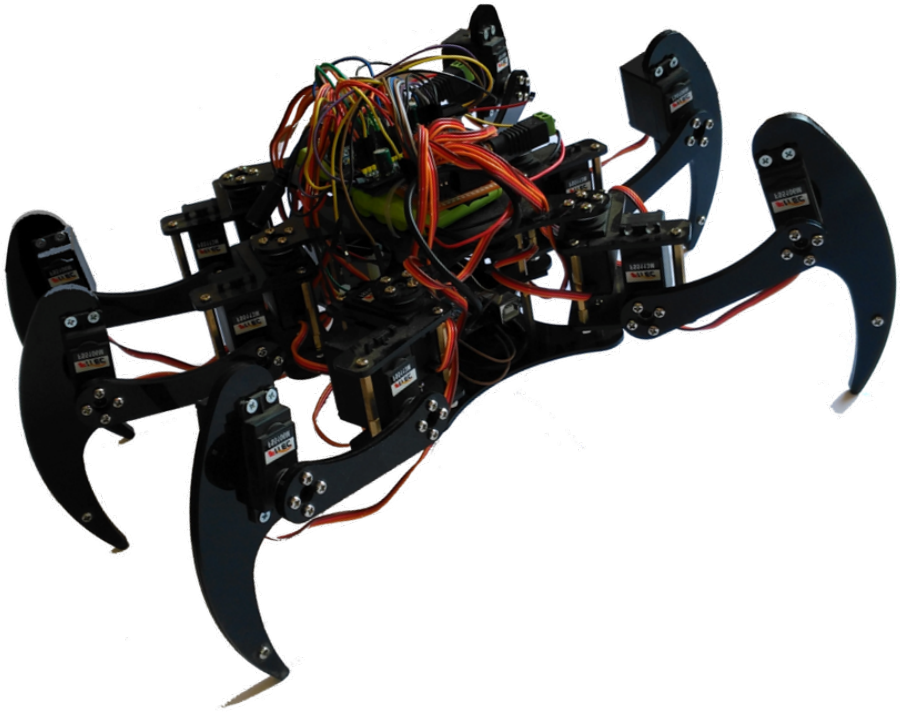

Figure 1.1.: A sex-legged walking robot.

wheeled or trucked vehicle. In fact, the use of wheels or crawlers limits the size of the obstacle that can be climbed to half the diameter of the wheels. On the contrary, legged robots can overcome obstacles that are comparable with the size of the machine leg[terrain hex-limbed extra-terrestrial explorer,]. Hexapod walking robots also benefit from a lower impact on the terrain and have greater mobility in natural surroundings. This is especially important in dangerous environments like mine fields, or where it is essential to keep the terrain largely undisturbed for scientific reasons [Tedeschi and Carbone, 2014].

Hexapod legged robots have been used in exploration of remote locations and hostile environments such as seabed [Dynamics, 2015b], in space or on planets [Dynamics, 2015a, Moore and Buehler, 2001] in nuclear power stations [Ding et al., 2010], and in search and rescue operations[Manoiu-Olaru et al., 2011]. Beyond this type of application, hexapod walking vehicles can also be used in a wide variety of tasks such as forests harvesting, in aid to humans in the transport of cargo, as service robots and entertainment. Development of hexapods is increasingly robust in the military sector. Armies all over the world are exploring ways of using hexapods to detect land mines, traverse rocky, unstable terrain, and carry out simple delivery missions in danger zones.

1.1. History

Robots inspired by insects and other animals have previously been designed with physical antennae and tactile sensors to navigate their environment, as in the work by Brooks (1989) [**?**, **?**], Cowan et al. (2005), Hartmann (2001) [**?**] and Lee et al. (2008) [Digia, 2017]; the last three works employed the use of a single tactile element rather than a pair [MohdDaud and KenzoNonami, 2012]. Because of their extreme mobility and agile adaptability to irregular terrain, insects have long been an inspiration for the designers of mobile and legged robots [Lewinger and Quinn, 2010, Lewinger and MartinReekie, 2011]. Early hexapod robots such as Genghis and later creations such as Tarry implemented insect-like mobility based on observations of insect behaviors. The inter- leg coordination system developed by Holk Cruse [**?**, **?**] has been widely implemented in legged hexapods and its basis is in behavioral experiments that qualitatively analyzed insect walking behaviors [Dürr et al., 2004].

1.1.1. Early Designs

The first hexapods can be identified as robots based on a rigidly predetermined motion so that an adaptation to the ground was not possible. Early researches in the 1950s were focused on assigning the motion control completely by a human operator manually[Schneider and Schmucker, 2006].

One of the first successful hexapod robot was constructed at University of Rome in 1972 (Fig.Figure 1.2a) as a computer-controlled walking machine with electric drives[Peternella and Salinari, 1973]. In the middle 70s, at the Russian Academy of Sciences in Moscow, a six-legged walking machine was developed with a mathematical model of motion control. It was equipped with a laser scanning range finder and was connected with a two-computer control system [Okhotsimski and Platonov, 1973]. In 1976, Masha hexapod walking robot was designed at Moscow State University (Fig.Figure 1.2b). The robot had a tubular axial chassis, articulated legs with three DoFs [Gurfinkel et al.,]. The hexapod was able to negotiate obstacles using contact on the feet and a proximity sensor. Ohio State University in 1977 developed a six-legged insect-like robot system called “OSU Hexapod” [McGhee, 1977]. This hexapod was kept tethered and was made to walk short distances over obstacles (Fig.Figure 1.2c). In 1984, Odetic Inc., California, USA, developed Odex I [Byrd and de Vries, 1990], a six-legged radially symmetric hexapod robot which used an onboard computer to play back pre-programmed motions (Fig.Figure 1.2d).

1.1.2. Recent Developments

The two last decades have been characterized by a rapid development of control systems technology. Hexapod robots were equipped with various sensing systems. Artificial

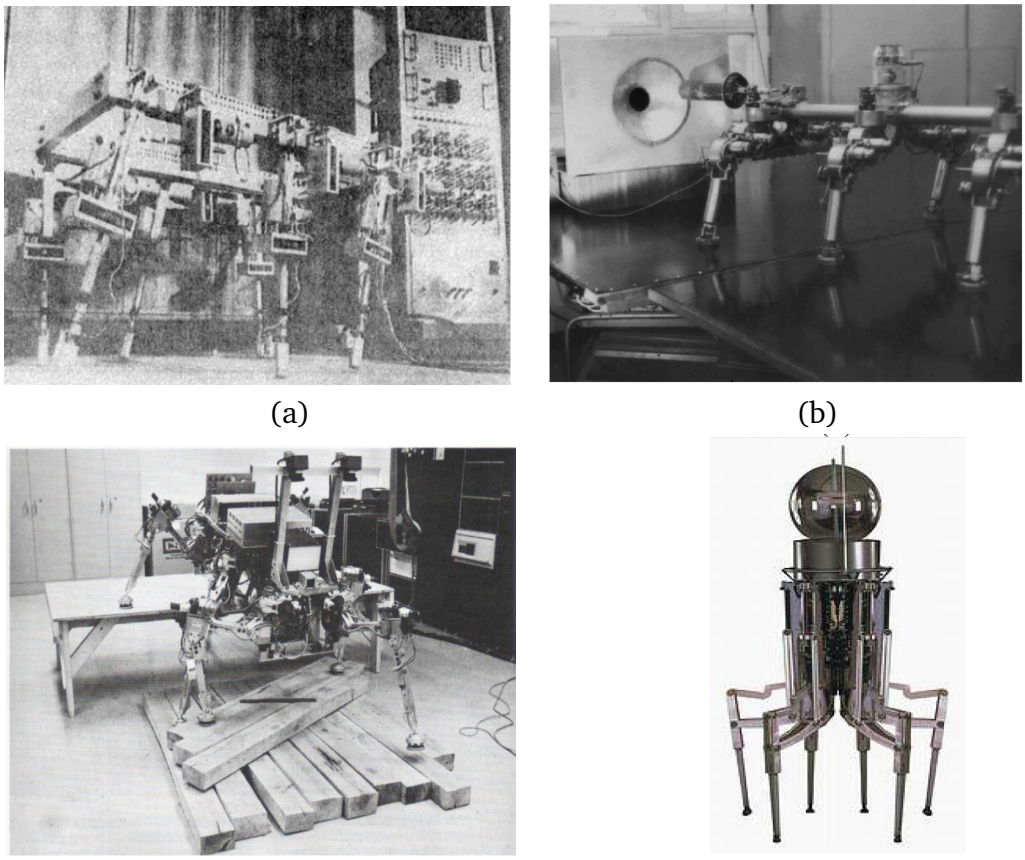

Figure 1.2.: Early hexapod design: (a) University of Rome's hexapod; (b) MASHA hexapod; (c) OSU hexapod; (d) ODEX I hexapod.

Intelligence systems were widely applied to the analysis of environment and motion of robots on a complex surface. A series of bio inspired robots was developed at Case Western Reserve University (USA) at the end the 90s, such as, for example, Robot III that had a total of 24 DoFs. Robot III architecture was based on the structure of cockroach, trying to imitate their behavior [**?**]. In particular, each rear leg had three DoFs, each middle leg four DoFs and each front leg five DoFs. Similarly, Biobot was a biomimetic robot physically modeled as the American cockroach (Periplaneta Americana) and powered by pressurized air [**?**]. This hexapod had a great speed and agility.
Each leg of the robot had three segments, corresponding to the three main segments of insect legs: coxa, femur, and tibia.

Figure 1.3.: Some Example on recent developments in hexapod design

"It is impossible for us, who live in the latter ages of the world, to make observations in criticism, morality, or in any art or science, which have not been touched upon by others. We have little else left us but to represent the common sense of mankind in more strong, more beautiful, or more uncommon lights."

— Joseph Addison, (English essayist, poet, and politician, 1672–1719), Spectator, No. 253

Chapter 2

Design Considerations

Designing hexapod legged robots is far from trivial. A very numerous and a wide range of possibilities exist to design a hexapod as also described in the previous section. Designers must take several decisions which influence the operation and technical features. Some of the most important design issues and constraints according to [**?**] can be outlined as:

- The mechanical structure of robot body.
- Leg architecture.
- Max sizes.
- Actuators and drive mechanisms.
- Control architecture.
- Power supply.
- Walking gaits and speed.
- Obstacle avoidance capability.
- Payload.
- Autonomy.
- Operation features.
- Cost.

The above mentioned design issues and constraints can be classified as design input (or key features) and design output (or main design characteristics) as shown in the scheme of Fig.Figure 2.1.

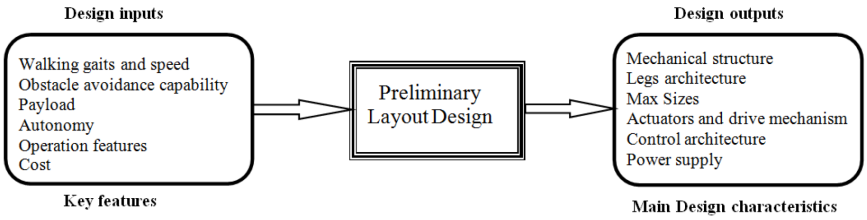

Figure 2.1.: A scheme for preliminary layout design of hexapod walking robots.

2.1. Hardware

ZagHexa is a hexapod robot with 18 DOFs (three degrees of freedom (DOF) for each leg), it can walk in any direction (translation), or turn in place (rotation), or any combination of the two. The leg lift and ride height is adjustable as well. The robot uses a distributed walking control system based on the neurobiology of insects, stepping in the sagittal plane to angled stepping, which then induces turning in the robot. It is an integrated multi-legged walking robot based on de-facto standard Robotic Operating System (ROS) that employs novel and different walking patterns. Our robot is teleoperated using hand-held devices such as a smart phone or tablet or a wireless joystick (see Fig.Figure 2.2). Furthermore, it has its own navigation system and a camera for instant video recording and streaming. The power to the entire system is supplied through two 5 volts NiMH batteries. There is an additional power bank to power up the Raspberry Pi and other electronic components. We have an interactive website for robot inspection and online control in addition to leaning materials such as robot building and implementation walkthroughs and as well as step-by-setup tutorials.

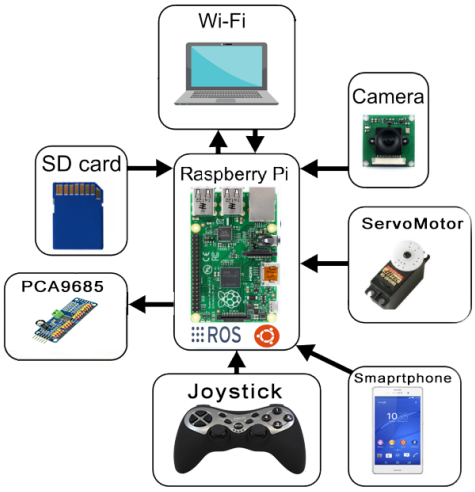

Figure 2.2.: The electronic system of the robot.

2.1.1. Components needed

Metal sheet Aluminum structure
Raspberry Pi 3 Model B
Arduino Mega
Metal Gear Servo motor standard size (13 kg.cm)
Metal Gear Servo motor standard size (7.5 kg.cm)
Metal Servo gear hub
NiMH rechargeable battery (5V -1500 mAh)
Copper Spacers
Plastic Spacers
Screws and nuts
Jumper wires
Power Bank 8000 mAh
android smart phone.

2.2. Hardware and Software Architecture

The final design of the ZagHexa robot constructed mainly with acrylic is shown in Fig.Figure 1.1. ZagHexa body moves independently of its ground contact points. To make its center of gravity shift on a horizontal plane, forward/backward, and sideways moving functions are effective. These functions can also produce a smooth body movement independently of intermittent leg traveling. The robot has been designed with three degrees of freedom in the front, middle and rear legs respectively. The physical specifications are given in Table 2.

Parameter	Description
Length	30cm
Width	27cm
Height	17cm
Weight	3Kg
Construction Material	
Actuators	DC servo motors
Motion Control	Servo Sequential Control
Leg Stroke (Max)	6cm
Leg Lift (Max)	5cm

2.2.1. Hardware Architecture

As illustrated in Fig.Figure 2.2, ZagHexa is equipped with all necessary resources to interact with the environment. The system is supplied with an embedded computing

system, which is Raspberry Pi 3 Model B running Ubuntu Xilinx.
Raspberry Pi is a miniature computer the size of a credit card, to which a standard monitor, a keyboard and a mouse can be connected.
It has extremely low power consumption (max. 3.5 W) and can run ROS based on Ubuntu operating system. There are several models, which differ in RAM, the number of USB ports or GPIO pins and the connation methods. Raspberry Pi is equipped with a USB Wi-Fi dongle, which is connected to a wireless network, and runs Qt [11] client program, that connects to the server and communicate with it. Sensor data are sent to a computer after successful connection. Client is also capable of automatically reconnect in case of connection disruption.
To sense the environment and deal with it, different modalities of sensors are imported from an attached smart hand-held device. As ZagHexa can use all the sensors form any mobile phone thanks to its own android subsystem that communicate these information with the smart phone over Wi-Fi or Bluetooth.

2.2.2. Joint Actuators

The leg joints are actuated by metal gear servos: FS5113M on coxa, femur and FS5106M on tibia joints. These motors were chosen for their high torque. Each of the three joint actuators per leg directly drives its associated leg segment. By attaching the leg segment directly to the servo output horns, the mechanical design of the joints is simplified. This direct connection also allows the joints to take advantage of the nominal $\pm 90 deg$ range-of motion (RoM) of the servos, which is approximately the same RoM as the insect [KanYoneda, 2007].

It is important for the leg control system to know current joint angles (servo position) and joint loads (current consumption). As this information is not available from standard servos, the motors were modified. The servo internal PCB, which is responsible for receiving PWM position commands from a host and converting those commands into servo output positions. All servomotors are connected and driven by the PCA9685 controller. This controller receives the desired walking pattern from the Raspberry through I2C pins controlling the leg lift-swing-shift sequence.

2.3. Software

The high–level functionality and control of the robot are implemented in ROS packages. The Robot Operating System (ROS) is a standard and open–source operating system for robot control [Cousins, 2011]. ROS is not an operating system in the traditional sense of process management and scheduling; rather, it provides a structured communication layer above the host operating systems of a heterogeneous compute cluster. Our software ROS packages interact with each of the subsystems in C++ and Python for

direct system control. The system described uses a Linux based software framework as an operating system (OS) for providing the advantages of using an OS, which supports developing additional modules that can be easily implemented and integrated.

ROS provides operating system like service for the robot. It is a meta-operating system, which loads on top of an operating system to provide a standardized set of software framework and APIs. These facilities cannot only help robots but also other embedded systems with a rich set of tools to successfully manage the complexity. With ROS handling the basic communications and data exchange.

Chapter 3
Simulation

Programming directly on a real robot gives us good feedback and it is more impressive than simulations, but not everybody has possible access to real robots. For this reason, we have programs that simulate the physical world.
The first phase of robot manufacturing is its design and modeling. We can design and model the robot using CAD tools such as Solid Works, Blender, and so on. One of the main purposes of modeling robot is simulation. The robotic simulation tool can check the critical flaws in the robot design and can confirm the working of the robot before it goes to the manufacturing phase.

The virtual robot model must have all the characteristics of real hardware, the shape of robot may or may not look like the actual robot but it must be an abstract, which has all the physical characteristics of the actual robot.

If we are planning to create the 3D model of the robot and simulate using ROS, you need to learn about some ROS packages which helps in robot designing. ROS has a standard meta package for designing, and creating robot models called robot model, which consists of a set of packages called urdf, robot state publisher and so on. These packages help us create the 3D robot model description with the exact characteristics of the real hardware.

In this chapter, we will cover the following topics:

1. ROS packages for robot modeling
2. Understanding robot modeling using URDF
3. Creating our URDF model
4. Watching the 3d model in RVIZ
5. Making our robot movable

3.1. ROS packages for robot modeling

The way ROS uses the 3D model of a robot or its parts, to simulate them. ROS provides some good packages that can be used to build 3D robot models.
In this section, we will discuss some of the important ROS packages that are commonly used to build robot models:

robot model: ROS has a meta package called robot model, which contains important packages that help build the 3D robot models. We can see all the important packages inside this meta- package:

URDF: One of the important packages inside the robot model meta package is urdf. The URDF package contains a C++ parser for the Unified Robot Description Format (URDF), which is an XML file to represent a robot model.

We can define a robot model, sensors, and a working environment using URDF and can parse it using URDF parsers.
We can only describe a robot in URDF that has a tree-like structure in its links, that is, the robot will have rigid links and will be connected using joints. Flexible links can't be represented using URDF.
The URDF is composed using special XML tags and we can parse these XML tags using parser programs for further processing. We can work on URDF modeling in the upcoming sections.

joint state publisher: This tool is very useful while designing robot models using URDF.
This package contains a node called joint state publisher, which reads the robot model description, finds all joints, and publishes joint values to all non fixed joints using GUI sliders.
The user can interact with each robot joint using this tool and can visualize using RViz. While designing URDF, the user can verify the rotation and translation of each joint using this tool.

kdl parser: Kinematic and Dynamics Library (KDL) is an ROS package that contains parser tools to build a KDL tree from the URDF representation. The kinematic tree can be used to publish the joint states and also to forward and inverse kinematics of the robot.

robot state publisher: This package reads the current robot joint states and publishes the 3D poses of each robot link using the kinematics tree build from the URDF. The 3D pose of the robot is published as ROS tf (transform). ROS tf publishes the relationship between coordinates frames of a robot.

xacro: Xacro stands for (XML Macros) and we can define how xacro is equal to URDF plus add-ons. It contains some add-ons to make URDF shorter, readable, and can be used for building complex robot descriptions. We can convert xacro to URDF at any time using some ROS tools. We will see more about xacro and its usage in the upcoming sections.

3.2. Understanding robot modeling using URDF

We have discussed the urdf package. In this section, we will look further at the URDF XML tags, which help to model the robot. We have to create a file and write the relationship between each link and joint in the robot and save the file with the .urdf extension.
The URDF can represent the kinematic and dynamic description of the robot, visual representation of the robot, and the collision model of the robot.

The following tags are the commonly used URDF tags to compose a URDF robot model:
link: The link tag represents a single link of a robot. Using this tag, we can model a robot link and its properties. The modeling includes size, shape, color, and can even import a 3D mesh to represent the robot link. We can also provide dynamic properties of the link such as inertial matrix and collision properties.
The syntax is as follows:

```
<link name="<name of the link>">
    <inertial>...........</inertial>
    <visual> ............</visual>
    <collision>..........</collision>
</link>
```

The following is a representation of a single link. The Visual section represents the real link of the robot, and the area surrounding the real link is the Collision section. The Collision section encapsulates the real link to detect collision before hitting the real link.
joint: The joint tag represents a robot joint. We can specify the kinematics and dynamics of the joint and also set the limits of the joint movement and its velocity. The joint tag supports the different types of joints such as revolute, continuous, prismatic,fixed, floating, and planar.

The syntax is as follows:

```
<joint name="<name of the joint>">
  <parent link="link1"/>
  <child link="link2"/>
  <calibration .... />
  <dynamics damping ..../>
```

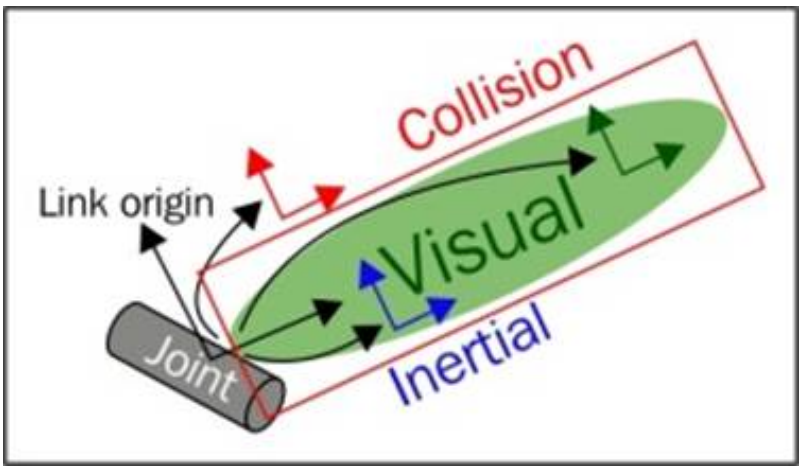

Figure 3.1.: Visualization of a URDF link

```
  <limit effort .... />
</joint>
```

A URDF joint is formed between two links; the first is called the Parent link and the second is the Child link. The following is an illustration of a joint and its link:

robot: This tag encapsulates the entire robot model that can be represented using

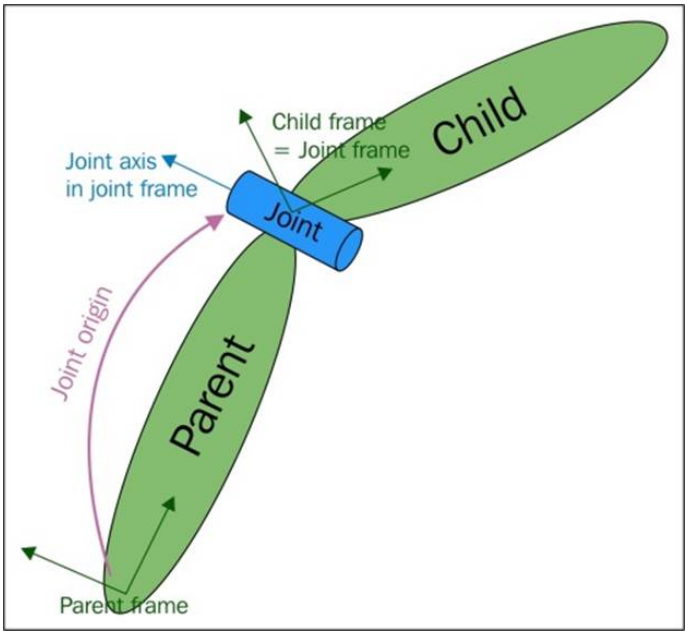

Figure 3.2.: Visualization of a URDF joint

URDF. Inside the robot tag, we can define the name of the robot, the links, and the joints of the robot. The syntax is as follows:

```
<robot name="<name of the robot>"
  <link> ..... </link>
  <link> ..... </link>
  <joint> ..... </joint>
  <joint> ..... </joint>
</robot>
```

A robot model consists of connected links and joints. Here is a visualization of the robot model:

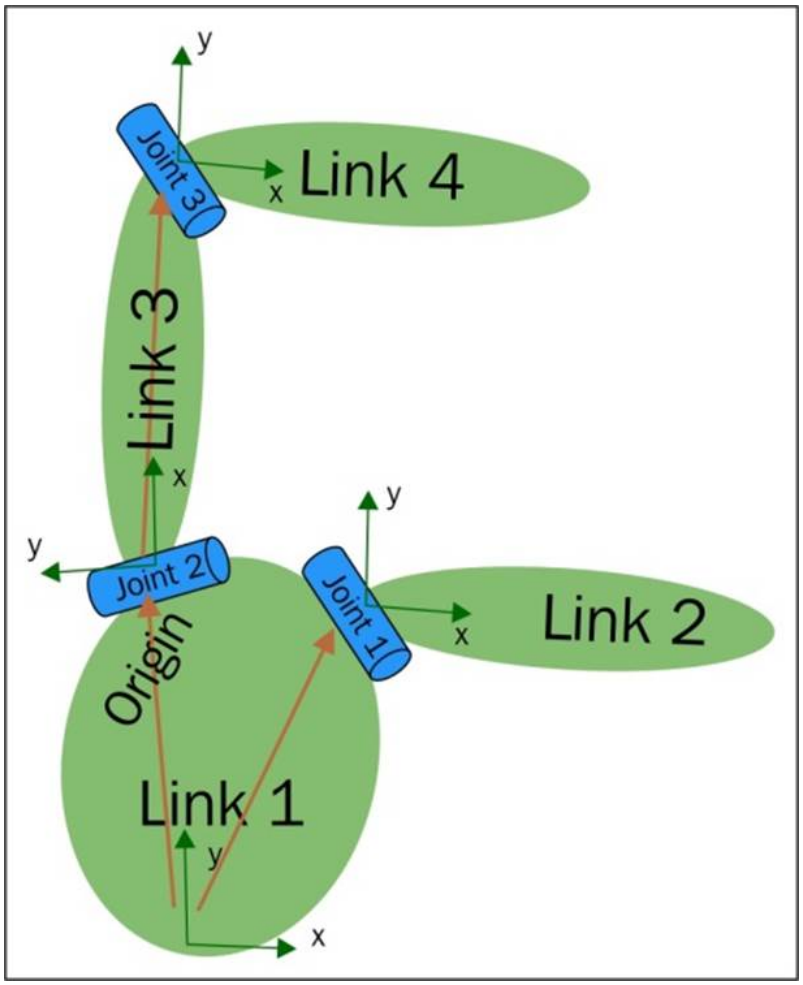

Figure 3.3.: Visualization of a robot model having joints and links

3.3. Creating our URDF model

To start first we create the urdf file, let's call it *zaghexasim.urdf* and put in the following code; this URDF code is based on XML.As you will see in the code, there are two

principal fields that describe the geometry of a robot: links and joints.
the first link has the name base link; this name must be unique to the file

```
<?xml version="1.0" ?>
<robot name="zaghexa" xmlns:xacro="http://ros.org/wiki/xacro">
 <!-- Build the body frame -->
<link name="base_link"/>
<joint name="base_joint" type="fixed">
<parent link="base_link"/>
<child link="box"/>
<origin rpy="0 0 0" xyz="0 0 0"/>
</joint>
<link name="box">
<visual>
<origin rpy="0 0 0" xyz="0 0 0"/>
<geometry>
<mesh filename="package://zaghexa_sim/meshes/box.STL"/>
</geometry>
<material name="grey">
<color rgba="0.5 0.5 0.5 1"/>
</material>
</visual>
</link>
```

In the joint field we define the name which must be unique as well also we define the type of joint(fixed,revolute,continous,floating or planar) the parent, and the child.
in our case tibia,femur and leg centre joint are the children of base link which is fixed but all of other joints are revolute.
this is a sample of one leg and how does it build

```
<!-- Joint properties -->
<!-- Leg macros -->
<!-- Build robot model -->
<joint name="leg_center_joint_r1" type="fixed">
<origin rpy="0 0 0" xyz="0.087598 -0.050575 0"/>
<parent link="box"/>
<child link="leg_center_r1"/>
</joint>
<link name="leg_center_r1"/>
<joint name="coxa_joint_r1" type="revolute">
<origin rpy="0 0 -1.0471975512" xyz="0 0 0"/>
<parent link="leg_center_r1"/>
<child link="coxa_r1"/>
```

```
<axis xyz="0 0 -1"/>
<limit effort="10000" lower="-1.5" upper="1.5" velocity="100"/>
</joint>
<link name="coxa_r1">
<visual>
<origin rpy="0 0 0" xyz="0 0 0"/>
<geometry>
<mesh filename="package://zaghexa_sim/meshes/coxa_r.STL"/>
</geometry>
<material name="">
<color rgba="0.7 0.7 0 1"/>
</material>
</visual>
</link>
<joint name="femur_joint_r1" type="revolute">
<origin rpy="-1.57079632679 0 0" xyz="0.0294 0 0"/>
<parent link="coxa_r1"/>
<child link="zaghexa"/>
<axis xyz="0 0 -1"/>
<limit effort="10000" lower="-1.5" upper="1.5" velocity="100"/>
</joint>
<link name="zaghexa">
<visual>
<origin rpy="0 0 0" xyz="0 0 0"/>
<geometry>
<mesh filename="package://zaghexa_sim/meshes/femur_r.STL"/>
</geometry>
<material name="">
<color rgba="0 0.7 0.7 1"/>
</material>
</visual>
</link>
<joint name="tibia_joint_r1" type="revolute">
<origin rpy="3.14159265359 0 1.57079632679" xyz="0.08 0 0"/>
<parent link="zaghexa"/>
<child link="tibia_r1"/>
<axis xyz="0 0 1"/>
<limit effort="10000" lower="-1.5" upper="1.5" velocity="100"/>
</joint>
<link name="tibia_r1">
```

You can check the syntax of the urdf whether we have errors, we can use: check urdf command tool:

```
$ rosrun urdf_parser check_urdf zaghexa_sim.urdf
```

If you want to see it graphically, you can use the urdf to graphiz command tool

```
$ rosrun urdf_parser urdf_to_graphiz "`rospack find zaghexa_sim`/
    urdf/zaghexa_sim.urdf"
```

The following is what you will receive as output:

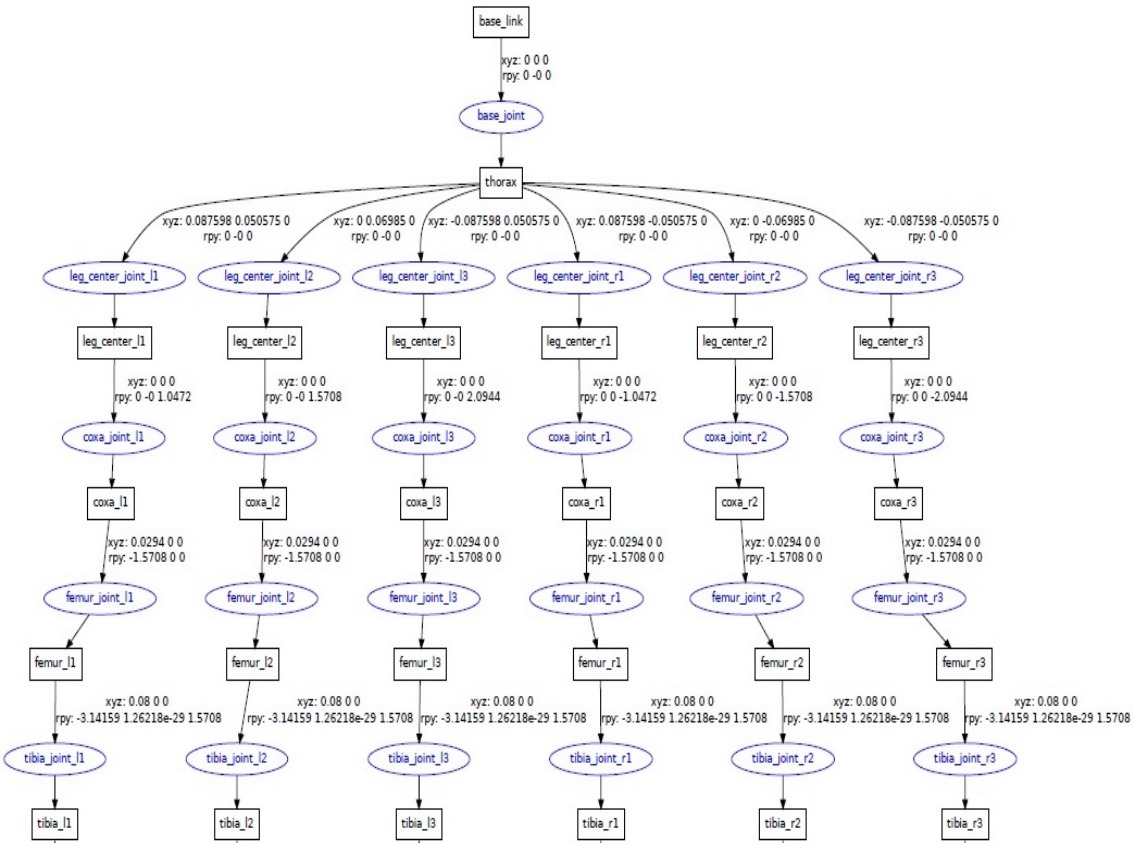

Figure 3.4.: output of urdf to graphics

3.4. Watching the 3D model in RVIZ

Now that we have the model of our robot, we can use it on rviz to watch it in 3D and see the movements of the joints.
We will create the display.launch file in zaghexa-sim/launch folder, and put the following code in it:

```
<launch>
```

```
<arg
name="model" />
<arg
name="gui"
default="True" />
<param
name="robot_description"
command="$(find xacro)/xacro.py '$(find zaghexa_sim)/models/
   zaghexa_model.xacro'" />
<param
name="use_gui"
value="$(arg gui)" />
<param
name="rate"
value="25" />
<rosparam param="source_list">
[leg_joints_states]
</rosparam>
<node
name="joint_state_publisher"
pkg="joint_state_publisher"
type="joint_state_publisher" />
<node
name="robot_state_publisher"
pkg="robot_state_publisher"
type="state_publisher" />
<node
name="rviz"
pkg="rviz"
type="rviz"
args="-d $(find zaghexa_sim)/urdf.rviz" />
</launch>
```

We will launch it with the following command:

```
$ roslaunch zaghexa_sim display_model.launch model:="`rospack find
   zaghexa_sim`/urdf/zaghexa_sim.urdf"
```

if every thing is fine and you have no errors, it will load RVIZ and you will see:

3.5. Making our robot movable

A good way of testing whether or not the axis and limits of the joints are fine by running rviz with joint state publisher GUI

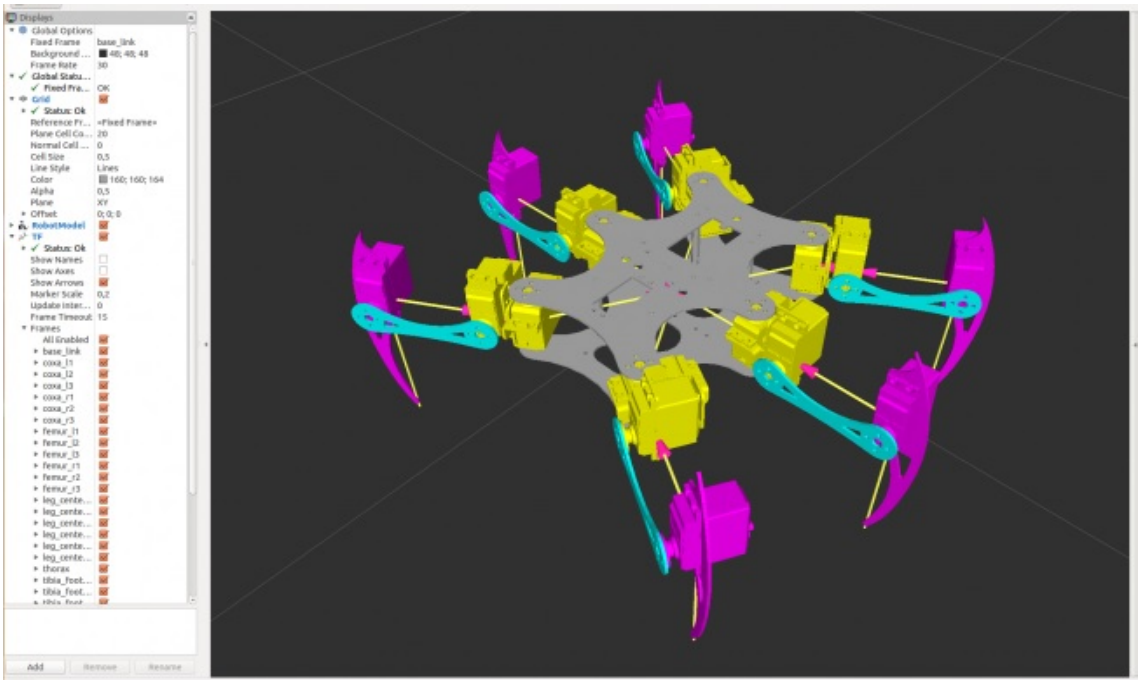

Figure 3.5.: output of urdf to graphics

```
$ roslaunch zaghexa_sim display.launch model:="‘rospack find
    zaghexa_sim‘/urdf/zaghexa_sim.urdf" gui:=true
```

you will see a GUI with some sliders each of them controls one joint of the 18 joints so we have 18 sliders:
In the next figures you will see the effect of changing sliders values to the joints angles and positions

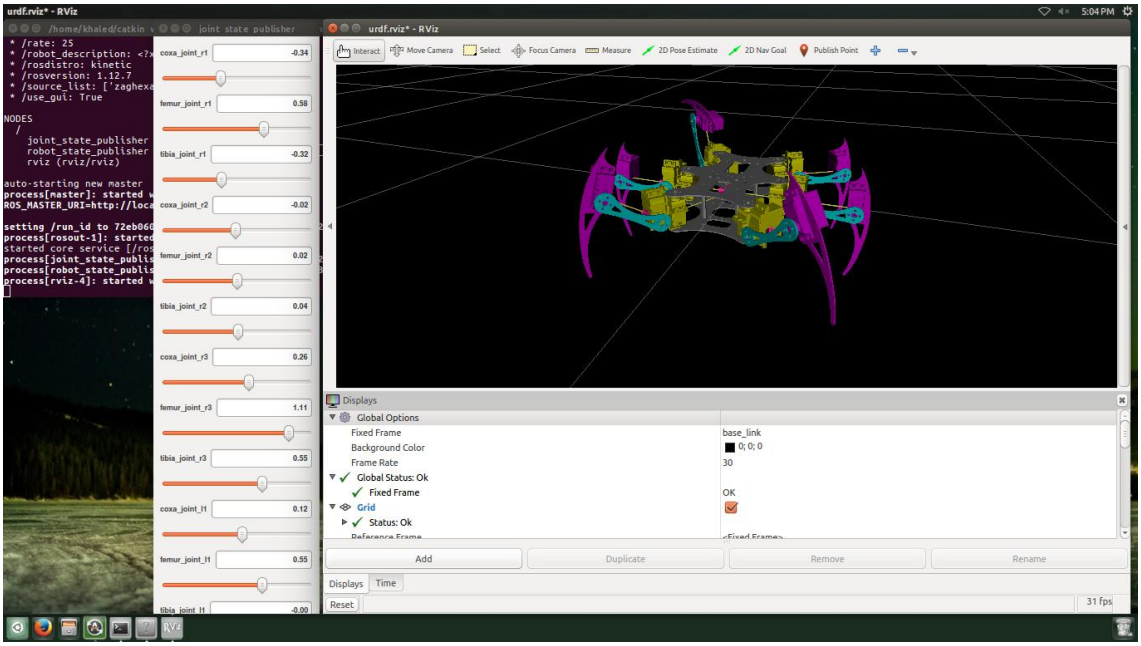

Figure 3.6.: Joint state publisher GUI with its control sliders and their effect on the robot

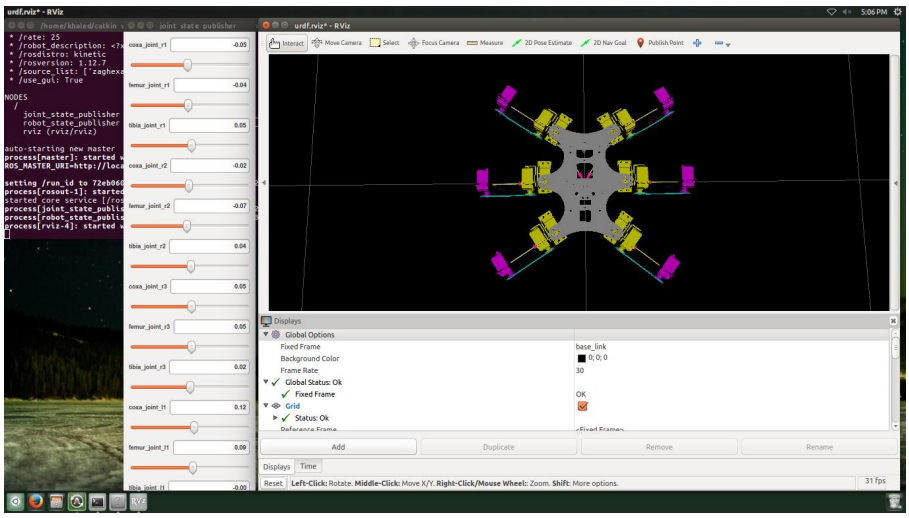

Figure 3.7.: top view of the robot

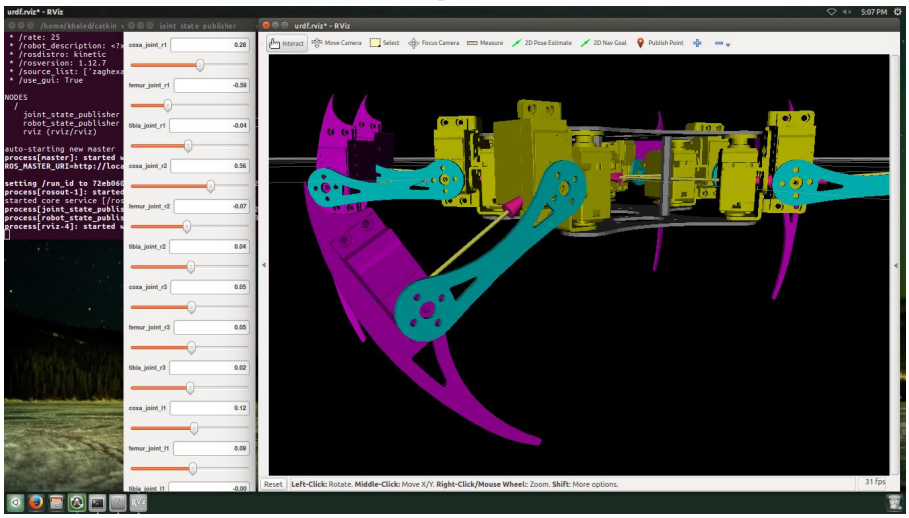

Figure 3.8.: Different views of the robot

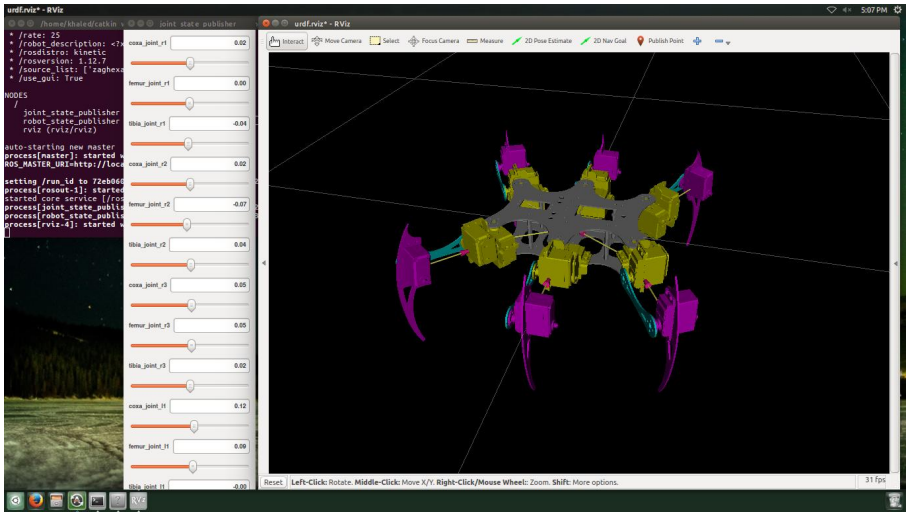

Figure 3.9.: Different views of the robot

"A computer would deserve to be called intelligent if it could deceive a human into believing that it was human."

— Alan Turing, (British pioneering computer scientist, cryptanalyst,···, and philosopher, 1912–1954)

Chapter 4
Mechanical Design

ZagHexa was first implemented in SolidWorks to verify its mechanical structure and produce the required workshop drawings to develop and cut its metal and plastic parts. In addition, some preliminary motion analysis simulations were also conducted. The final resulting design is given in Figure 4.1. To further develop the kinematic model of

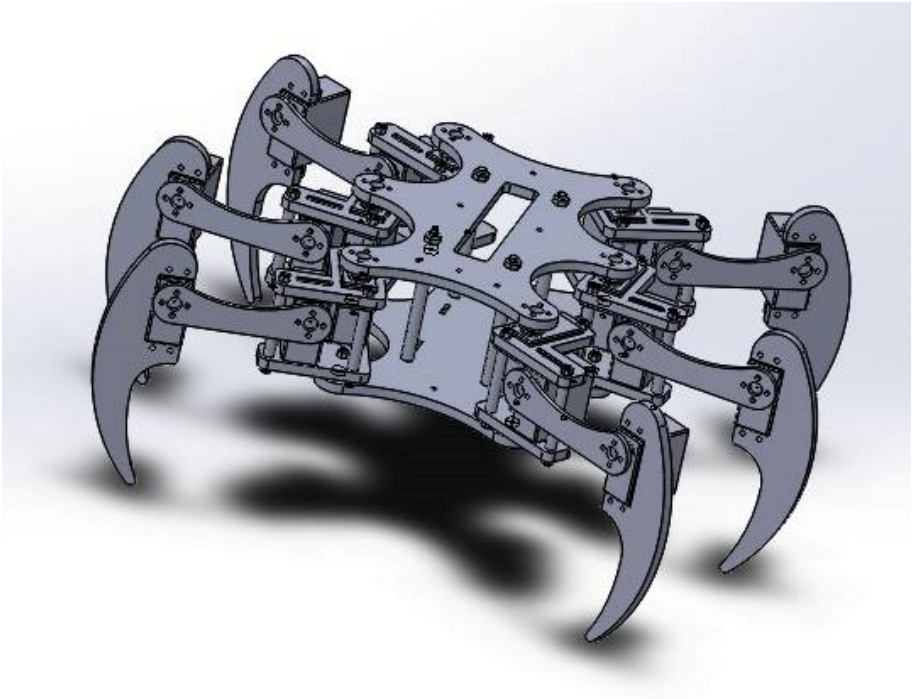

Figure 4.1.: CAD rendering of the robot.

the robot, the coordinate systems (Cartesian frames [**?**]) for all parts of the robot need to be identified. Next, the detailed development of such frames is given.

4.0.1. Robot body frame

The origin of the robot base frame will be in the center of the body, structured with Z-axis pointing up, the X-axis positioning right and Y-axis pointing forwards with respect to the robot front side as depicted in Figure 4.2.

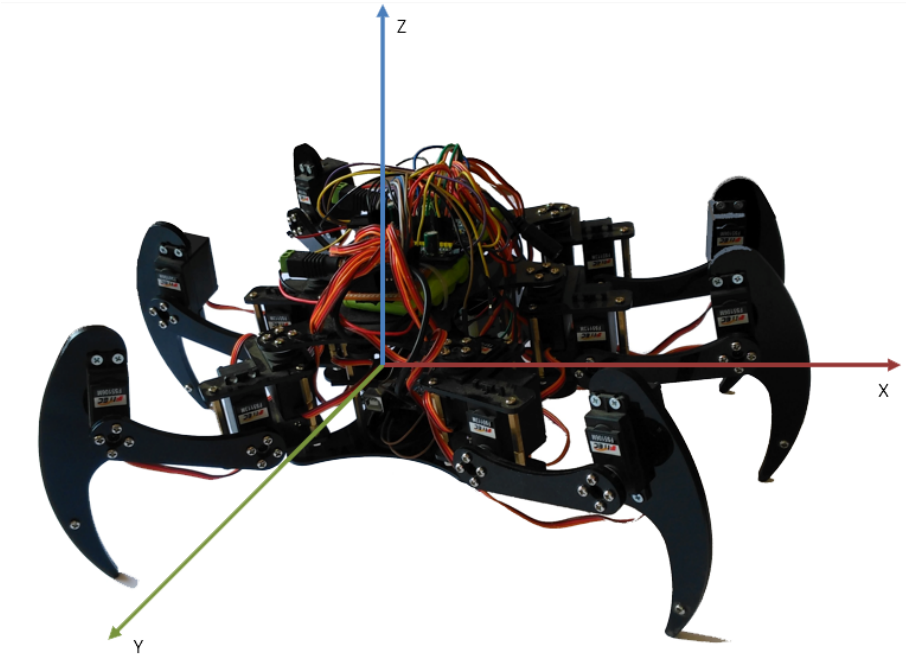

Figure 4.2.: Location of body frame relative to robot hardware.

4.0.2. Leg frames and notations

The design of hexapod constitutes the kinematic configuration of a hexapod robot, with each leg acting as an independent serial manipulator with three degrees of freedom. Figure Figure 1.1 shows the actual prototype of our robot.

The final leg design and its links and joints notations are given in Figure 4.3. The robot leg is made of links and joints as noted on Figure 4.4, different links of robot leg are called Femur, Tibia and Tarsus. As depicted in figure, the robot leg frame starts with link 0, which is the point where the leg is attached to the body, link 1, is Femur, link 2 is the Tibia and link 3 is Tarsus. The joints are located at the inner end of their respective link. Frames are attached to outer end of their respective links, this means that joint 2 rotates about the Z-axis of frame 1.

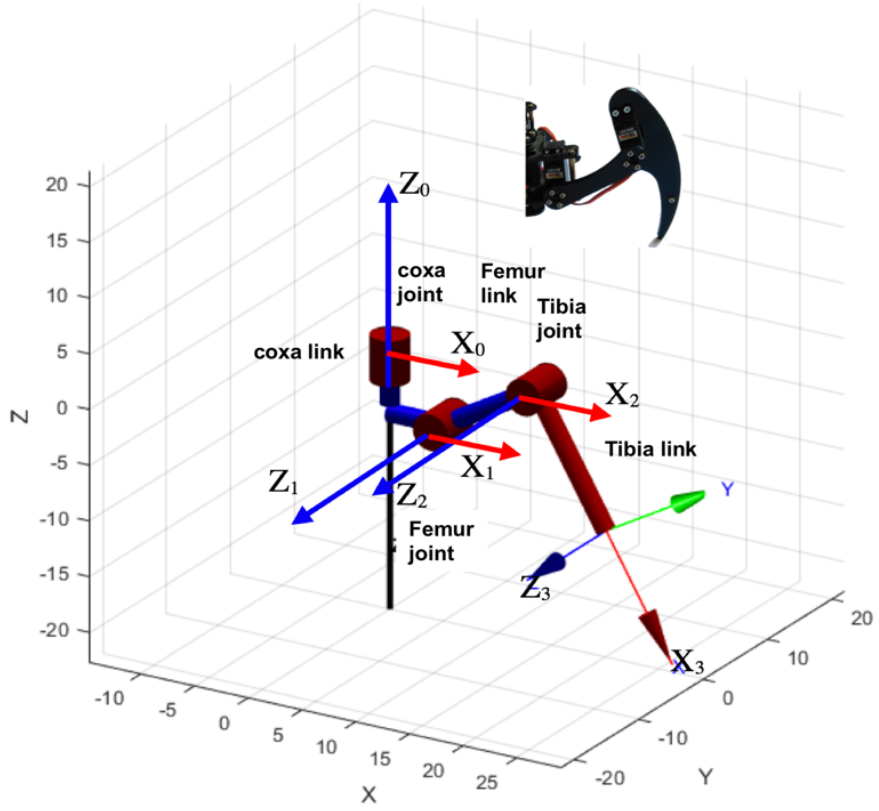

Figure 4.3.: Final leg design (top right) and its notations, reference frames, joints and links.

4.0.3. Robot Leg Parameters

Following the well-known Denavit-Hartenberg (DH) notation, coordinate frames for the robot leg are assigned. The assigned frames are shown in Figure 4.4. In figure, the body b and the zeroth 0 reference frames are attached to the stationary robot body. Therefore, they can be both considered as inertial frames. The axes of the body frame are arranged to be in accord with the actual robot-body orientation. The DH link parameters based on Figure 4.4 are given in Table II.

The resulting homogeneous transformation matrices between the body and the zeroth frame and between the sequential link frames are given in (1). In the formulas, the variables represented by a stand for the length of the i^{th} link (namely, the length of the portion of the link between the origins of $(i-1)thandi^{th}$ reference frames). The variables represented by θ_{ij} mean the sum of the i^{th} and j^{th} joint angles ($\theta_{ij} = \theta_i + \theta_j$). C and S are for cos(.) and sin(.) functions, respectively. The exact values of these variables corresponding to ZagHexa robot are: $\psi = 45°, a_1 = 5cm, a_2 = 9cm, a_3 = 18cm$

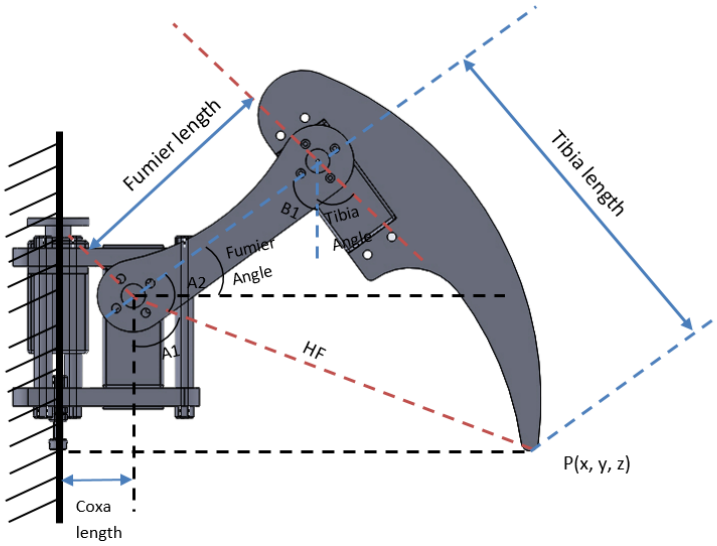

Figure 4.4.: Final leg design (top right) and its notations, reference frames, joints and links.

Joint	θ_i	α_i	a_i	d_i
1	θ_1	$\pi/2$	a_1	0
2	θ_2	0	a_2	0
3	θ_3	0	a_3	0

Homogeneous matrices are used in derivation of positional relations between the successive frames. In (3) the leg tip point position with respect to the body frame is given. The rotation matrices between the frames are given in (2). These rotation matrices are used in vector equations, especially while deriving the dynamic equations.

$$H^{(b,0)} = \begin{bmatrix} 0 & \cos(\psi) & \sin(\psi) & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H^{(K-1,K)} = \begin{bmatrix} \cos\theta_k & -\cos\alpha_k \sin\theta_k & \sin\alpha_k \sin\theta_k & a_k \cos\theta_k \\ \sin\theta_k & \cos\alpha_k \cos\theta_k & -\sin\alpha_k \cos\theta_k & a_k \sin\theta_k \\ 0 & \sin\alpha_k & \cos\alpha_k & d_k \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

$$H^{(0,3)} = H^{(0,1)} H^{(1,2)} H^{(2,3)} H^{(b,3)} = H^{(b,0)} H^{(0,3)}$$

$$C^{(b,0)} = \begin{bmatrix} 0 & \cos(\psi) & \sin(\psi) \\ -1 & 0 & 0 \\ 0 & -\sin(\psi) & \cos(\psi) \end{bmatrix}$$

$$C^{(K-1,K)} = \begin{bmatrix} \cos\theta_k & -\cos\alpha_k \sin\theta_k & \sin\alpha_k \sin\theta_k \\ \sin\theta_k & \cos\alpha_k \cos\theta_k & -\sin\alpha_k \cos\theta_k \\ 0 & \sin\alpha_k & \cos\alpha_k \end{bmatrix} \tag{4.2}$$

$$P_e^{(K-1,K)}(\theta) = \begin{bmatrix} C\psi(a_1 S\theta_1 + a_2 S\theta_1 C\theta_2 + a_3 S\theta_1 C\theta 23) + S\psi(a_2 S\theta_2 + a_3\theta_{23}) \\ -(a_1 C\theta_1 + a_2 C\theta_1 C\theta_2 + a_3 C\theta_1 C\theta_{23}) \\ -S\psi(a_1 S\theta_1 + a_2 S\theta_1 C\theta_2 + a_3 S\theta_1 C\theta 23) + C\psi(a_2 S\theta_2 + a_3\theta_{23}) \end{bmatrix} \tag{4.3}$$

To derive the dynamic equations, first the inertia matrices of the links should be determined. Since the k^{th} reference frame is stationary with respect to the k^{th} link, the inertia tensor of the k^{th} link around its center of mass appears to be a constant matrix with respect to the k^{th} reference frame, as in (4). The values used in these formulations belong to the Hexapod robot. The resulting matrices for each link are in the form of (5).

$$\{J_K\}^{(K)} = J_K^{(K)} = J_K \tag{4.4}$$

$$J_K = \begin{bmatrix} J_{K1} & 0 & 0 \\ 0 & J_{K2} & 0 \\ 0 & 0 & J_{K3} \end{bmatrix} \tag{4.5}$$

4.0.4. Inverse kinematics

The forward kinematics (FK) is a simple equation used to calculate the position of the end effectors for the leg in the robot base frame, by injecting values of each joint angle. But the reverse operation, namely inverse kinematics (IK), is more complex. IK is employed to find all the joint angles given the position of the end effectors. In general, solving the IK equations can be a bit of a challenge. Some positions cannot be reached at all, as the physical system is unable to get there, and some end effectors positions can have more than one solution, and not all of them are desirable.

We solve the IK problem for each leg separately, as this makes it possible to solve it geometrically, by setting up some constrains. The first constraint for solving the IK equations due the fact that all robot joints allow rotation about one axis only. The second constraint is that the Femur, Tibia joints always rotate on parallel axes. The third set of constraints arises from the physical limitations for each joint, giving us some angular interval for each joint in which the servos can actually rotate the link. In Figure 4.4, the angles of movement are shown. First, the coxa angle can be found directly by knowing the end effectors position then simply using $atan2(y,x)$ to calculate it. Equations (6) through (14) are used to find the individual joint angles.

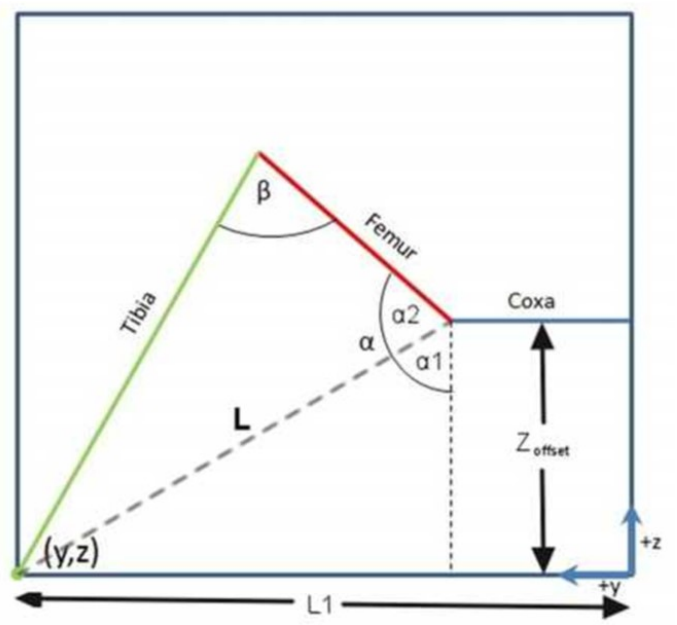

Figure 4.5.: Illustration of the 2D triangle with vertices in the coxa, the femur, and tibia link from origin.

$$\frac{x}{y} = \tan(y) \rightarrow \gamma = \tan^{-1}\frac{x}{y} \tag{4.6}$$

$$L = \sqrt{Z_{offset}^2 + (L_1 + \cos A)^2} \tag{4.7}$$

$$a_L = \cos^{-1}(\frac{Z_{offset}}{L}) \tag{4.8}$$

$$Tibia^2 = Femar^2 + L^2 - 2(Femar)(L)\cos\alpha_2 \tag{4.9}$$

$$\alpha_2 = \cos^{-1}(\frac{Tibia^2 - Femar^2 - L^2}{-2(Femar)(L)}) \tag{4.10}$$

$$\alpha = \alpha_1 + \alpha_2 \tag{4.11}$$

$$\alpha = \cos^{-1}(\frac{Z_{offset}}{L}) + \cos^{-1}(\frac{Tibia^2 - Femar^2 - L^2}{-2(Femar)(L)}) \tag{4.12}$$

$$\beta = \cos^{-1}(\frac{L^2 - Femar^2 - Tibia^2}{-2(Femar)(Tibia)} \tag{4.13}$$

4.1. Walking Pattern

In this section, the generation of the robot gait will be described. We provided the robot with a group of programmed gait sequences used for different purposes. For example, a Tripedal gait is used as the basic movement for the robot which provide

speed and longer traverse length. Metachronical gait is used for rough terrain traverse which provide better stability but slower motion [**?**, **?**]. Main types of gaits used in ZagHexa robot are shown in Figure 4.6. The description of these gaits is given next.

4.1.1. Wave gait (Metachronical Gait)

In this gait mode, the robot move one leg at a time, it starts by lifting one leg and then lowering it down gradually until the foot touches the ground and then the next leg starts to move, as mentioned before this gait sequence is rather slow but it provides maximum stability for the robot, and it enables the robot to walk on rough terrain. This is illustrated in Figure 4.6.

4.1.2. Ripple gait (Two wave Gait)

In this gait, two legs move at a time, since it has two independent wave gaits. The opposite sides legs are 180 degrees out of phase and it needs three beats to complete one cycle. Figure 4.6. shows the Ripple gait.

4.1.3. Tripedal Gait

This gait is the fast gait for the hexapod; it completes a cycle in two beats. In this gait, the robot lift three legs simultaneously while leaving three legs on the ground, which keeps the robot stable. Figure 4.6. shows Tripedal gait reaction [**?**].

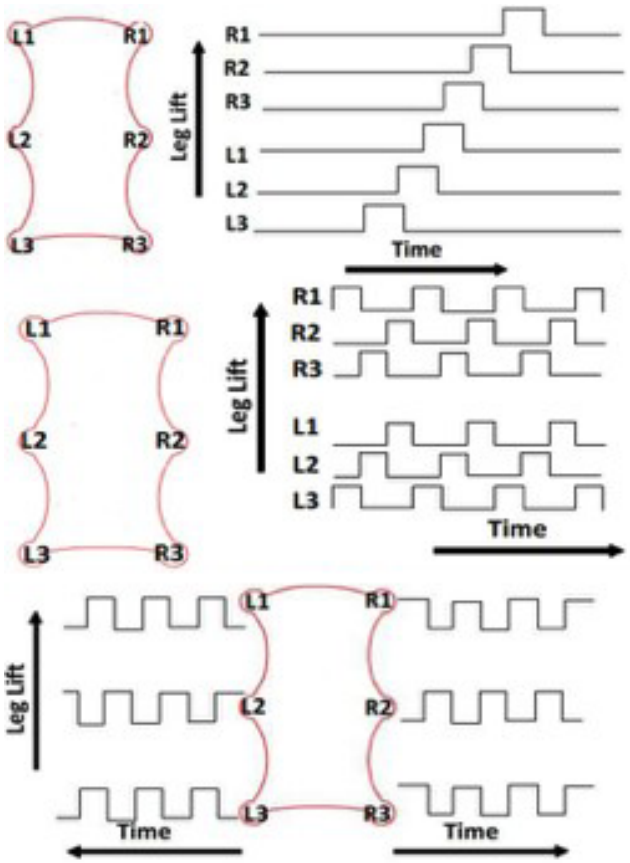

Figure 4.6.: Different walking gaits: wave (top), ripple (middle), and tripod (middle).

"It doesn't matter how beautiful your theory is, it doesn't matter how smart you are. If it doesn't agree with experiment, it's wrong."

— *Richard P. Feynman, (American theoretical physicist, 1918–1988)*

Chapter 5
Experiments and Results

To verify the designed and constructed robot as well as its software framework and control methods, a series of field experiments were performed. The first set of experiments was to test the robot basic movements (such as forward, backward walking and right and lift turn). Some results of these experiments are shown in Figure 5.1 (top). The second set was to test the implemented body kinematics. Example results such experiments are given in Figure 5.1 (bottom) which shows the robot raising and lowering its body height.

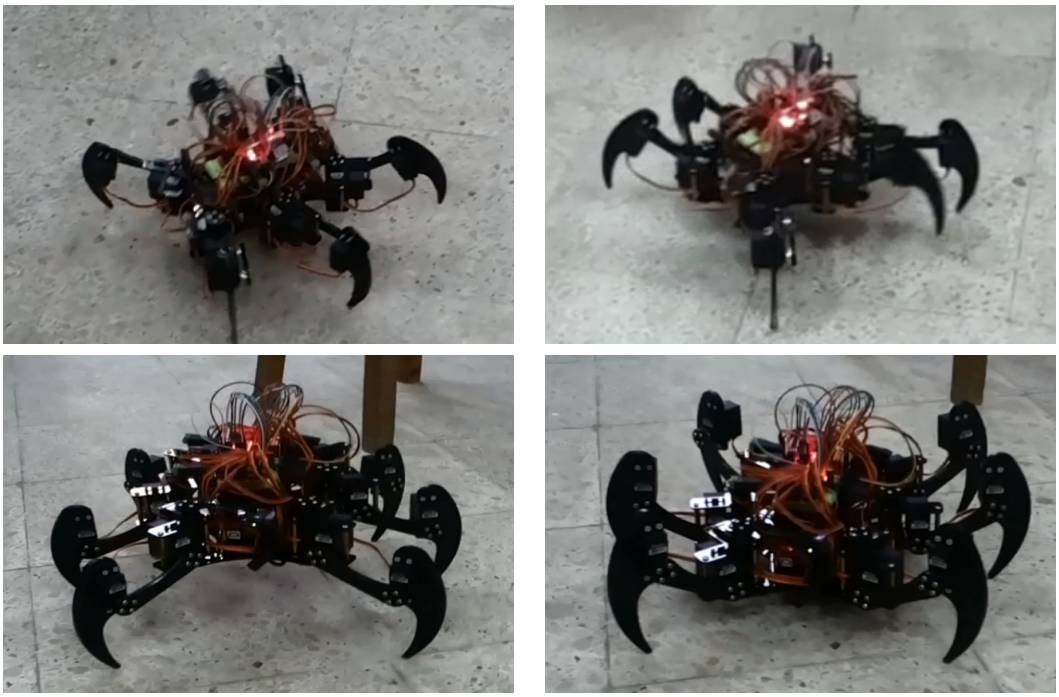

Figure 5.1.: Experiment results: forward and backward move (top) and raising and lowering the body height (bottom).

5.1. Simulation

To help test the design robot and verify software modules and the overall behaviour, the robot is modelled and simulated. The first model for the whole robot is performed in Rviz to test the different ROS nodes and the whole software framework. Rviz is one of ROS tools. It is capable of visualizing many different kinds of information in the same interface. You can load visualization plugins. In the parameters of these plugins you generally define the topic name to which the plugin subscribes, this is fairly straightforward. Rviz configuration file is provided with the sources and started by default in the launch file. Within our ROS-Rviz setup, we simulated the robot movement to test it in different conditions. That made it much easier and faster than apply all the scenarios on the real robot.

In the second simulation, we modelled the robot in MATLAB and employing the Robotics Toolbox. The main purpose of this simulation is to calculate and simulate the kinematics of robot. To create the six-legged walking robot, we started by creating a three-axis robot arm that we used as a leg. Then we implemented a trajectory for the leg that is suitable for walking. Finally, we instantiated six instances of the leg to create the walking robot. The equations given in Sec.4 are programmed first for one leg and tested on successful working, the whole body kinematics were also programmed and tested. The results were very useful in modifying the walking gaits of the robot which then implemented in the real robot. Figure 10 shows one such simulations in which the same experiment performed on the real robot to test the whole body kinematics for raising and lowering the body height.

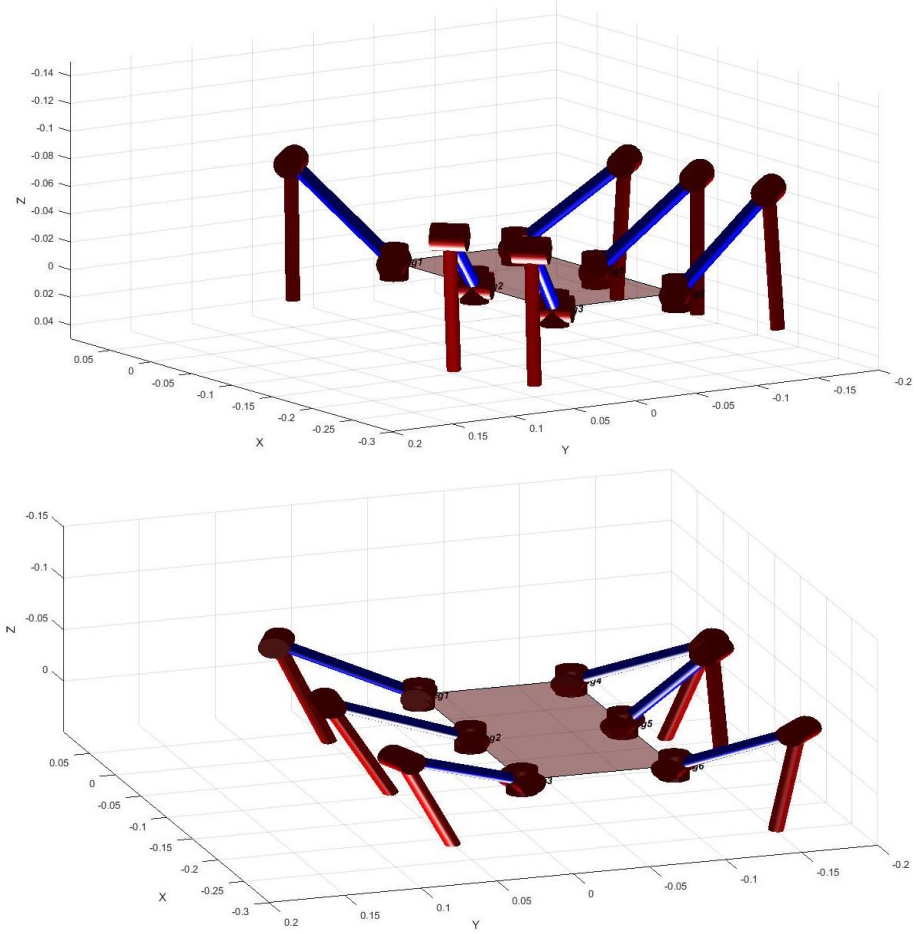

Figure 5.2.: Hexapod robot Simulation.

The true function of philosophy is to educate us in the principles of reasoning and not to put an end to further reasoning by the introduction of fixed conclusions.

— George Henry Lewes, (English philosopher and critic of literature, 1817–1878)

Chapter 6
Conclusions and Future Outlook

This paper presents a system description and the main aspects related to the design, construction, and implementation of six-leg robot named ZagHexa. The robot is a legged robot for search and rescue missions. It benefits form the reliability of its legged locomotion with the flexibility and versatility required to operate in different types of surface. The robot was constructed and tested to walk using tripod, wave and ripple gaits, can rotate and it is equipped with different sensors.

The robot was tested on different surfaces and in rugged terrain. The repeatability of the robot movement as well as the sensor system was also tested. These features are mainly achieved due to its original movement that make it deal with different surfaces. Additionally, its shape and weight give it more stability, and its ability to continue with its moving and sensing capabilities after collisions or even small falls.
However, more tests and experiments to improve and validate the design and sensor performance are to be carried out to optimize the system performance. Finally, we are working on tackling some issues should to have fully autonomous operation and integration into a heterogeneous system. To make the integration of ZagHexa into different missions easier, an effort is being carried to provide it with a standard connectivity over the ROS framework.

Appendix A
This is My Appendix Title

Bibliography

[Byrd and de Vries, 1990] Byrd, J. and de Vries, K. A. (1990). six-legged telerobot for nuclear applications development. *Int. J. Robot*, 9:43–52. [3]

[Cousins, 2011] Cousins, S. (2011). Exponential growth of ros [ros topics]. *IEEE Robotics Automation Magazine*, 18(1):19–20. [10]

[Digia, 2017] Digia (2017). Qt project. `http://qt-project.org`. [3]

[Ding et al., 2010] Ding, X., Rovetta, A., Zhu, J. M., and Wang, Z. (2010). Locomotion analysis of hexapod robot. *INTECH Open Access Publishe*. [2]

[Dynamics, 2015a] Dynamics, B. (2015a). Dedicated to the science and art of how things move, Boston dynamics. [1, 2]

[Dynamics, 2015b] Dynamics, B. (2015b). *Boston Dynamics*. Boston dynamics. [1, 2]

[Dürr et al., 2004] Dürr, V., Schmitz, J., and Cruse, H. (2004). "behaviour-based modeling of hexapod locomotion: linking biology and technical application". *Arthropod Structure & Development*, 33:237–250. [3]

[Gurfinkel et al.,] Gurfinkel, V., Gurfinkel, E., Devjanin, E., Efremov, E., Zhicharev, D., Lensky, A., Schneider, A., and Shtilman, L. I. o. r. In six-legged walking model of vehicle with supervisory control; nauka press: Moscow, russia, 1982;. *p*, pages 98–147. [3]

[KanYoneda, 2007] KanYoneda (2007). Light weight quadruped with nine actuators. *journal of robotics & mechatronics*, 19(2). [10]

[Lewinger and MartinReekie, 2011] Lewinger, W. A. and MartinReekie, H. (2011). A hexapod robot modeled on the stick insect carausiusmorosus. In *the 15th international conference on advanced robotics*, Tallinn. [3]

[Lewinger and Quinn, 2010] Lewinger, W. A. and Quinn, R. D. (2010). A hexapod walks over irregular terrain using a controller adapted from an insects nervous system. In *the IEEE/RSJ international conference on intelligent robots & systems (IROS)*, pages 18–22, Taiwan. IEEE/RSJ. [3]

[Manoiu-Olaru et al., 2011] Manoiu-Olaru, S., Nitulescu, M., and Stoian, V. (2011). Hexapod robot. mathematical support for modeling and control. In *System Theory, Control, and Computing (ICSTCC), 15th International Conference on*, pages 1–6. [2]

[McGhee, 1977] McGhee, R. (1977). Control of legged locomotion systems. *In Proceedings of the*, 18:205–215. [3]

[MohdDaud and KenzoNonami, 2012] MohdDaud and KenzoNonami (2012). Autonomous walking over obstacles by means of lrf for hexapod robot comet-iv. *robotics & Mechatronics*, 24(1). [3]

[Moore and Buehler, 2001] Moore, E. Z. and Buehler, M. (2001). Stable stair climbing in a simple hexapod robot. Technical report, DTIC Document. [2]

[Okhotsimski and Platonov, 1973] Okhotsimski, D. and Platonov, A. (1973). Control algorithm of the walking climbing over obstacles. In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, CA, USA, 20. Stanford. [3]

[Peternella and Salinari, 1973] Peternella, M. and Salinari, S. (1973). Simulation by digital computer of walking machine control system. In Genova, I., editor, *Proceedings of the 5th IFAC Symposium on Automatic Control in Space of the Conference*. [3]

[Saranlı, 2002] Saranlı, U. (2002). *Dynamic locomotion with a hexapod robot*. PhD thesis, The University of Michigan. [1]

[Schneider and Schmucker, 2006] Schneider, A. and Schmucker, U. (2006). Force sensing for multi-legged walking robots: Theory and experiments part 1: Overview and force sensing. In Intelligence, M. and Buchli, J., editors, *Mobile Robotics*, pages 125–174. Germany; Austria, ; Pro Literatur Verlag ARS. [3]

[Tedeschi and Carbone, 2014] Tedeschi, F. and Carbone, G. (2014). Designissues for hexapod walking robots. *Robotics*, 3(2):181–206. [1, 2]

[terrain hex-limbed extra-terrestrial explorer,] terrain hex-limbed extra-terrestrial explorer, N. A. 2009. [1, 2]