iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** VI **Month of publication:** June 2023

DOI: https://doi.org/10.22214/ijraset.2023.54189

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue VI Jun 2023- Available at www.ijraset.com

Design of Single Legged Hopper Robot in Gazebo

Chilupuri Anusha
Assistant Professor, Department of Electronics and Communication Engineering, G. Narayanamma Institute of Technology and Science, Hyderabad, India

Abstract: Robotics is one of the core areas where the bioinspiration is frequently used to design various engineered morphologies to develop novel behavioral controllers comparable to human being and animals.
Research on legged robots has been going on for over a century. The reason behind such sustained interest in legged robots is due to the fact that most of the earth's land surface is inaccessible to wheeled or tracked systems. Legged animals can, however, be found everywhere. Thus, mankind has been fascinated with the idea of a mobile legged robot that can handle difficult terrain and be useful in the fields of transportation, forestry, agriculture, fire, fighting, hazardous areas, defense (carrying weapons to soldiers, de-mining), police purposes, assistive devices for walking, entertainment (toy production), robotic pets, and ocean and space exploration.
Our project is based on one such example, single legged hopper robot a legged locomotive inspired by the jumping mechanism of click beetles. It aims to create a single legged hopper robot in Gazebo. A simulation tool that offers the ability to accurately, and efficiently, simulate populations of robots in complex indoor and outdoor environments. The Robot Operating System (ROS) works best in the LINUX OS, is the software of version Melodic + Gazebo 9, is utilized in the project.

I. INTRODUCTION

Legged robots can traverse unstructured terrain and could be used to explore areas of interest, such as craters, which rovers are unable to reach. In contrast to other legged robots, single legged is primarily built for hopping. One-legged hopping robots have long been used to study balance issues, but their dependence on off-board power has kept them tethered, literally, to the lab. Many hoppers have been hydraulic devices. But hydraulic actuation requires off- board motors and is energetic enough to pose issues outside of a controlled laboratory environment.
In general, small robots or unmanned ground vehicles (UGVs) are severely limited by their ability to negotiate obstacles. This demonstrated hopping capability allows small UGVs to overcome up to 30 obstacles that are 40-60 times their own size. In addition, hopping mobility has been shown to be 5 times more efficient than hovering for obstacles at heights less than or equal to 10 meters, which allows longer station- keeping time for the same amount of fuel.
Single legged hopping robots not only provide a simplified test bed for locomotion control algorithms, but they also demand high-speed, high force actuation to achieve safe and robust ground clearance and subject the actuator to greater mechanical stresses than do multi- legged systems. These robots also find its applications in fields like military, planetary exploration. For these reasons, single- legged hopping robots provide an ideal benchmark for actuators used in legged locomotion. The software required for this project is robotic operating system (ROS) which helps in creating the code required for to operate the robot of version Melodic + Gazebo 9.

II. LITERATURE SURVEY

A single legged hopper robot is a type of legged locomotive built for experiments on active balance and dynamics in legged locomotion. The machine has a leg that changes length, a body that carries sensors and interface electronics, and an actuated 2- axis hip. The hip is powered by hydraulics and the leg by compressed air. The one legged hopping robots are a typical and the simplest systems of the dynamic legged robots.
Most of research about the one-legged hopping robots were limited to the systems with Spring Loaded Inverted Pendulum (SLIP) model, which is composed of a point mass attached on a telescopic spring that is free to rotate around its point of contact with the ground. The SLIP model hopping systems can be stabilized without much effort in control design since the simple decoupling dynamics.
Raibert and his co-workers made a major contribution to this field through their 2D and 3D single legged hopper. Many other works have been done to improve the original legged system of Raibert. Zeglin introduced an "uniroo" that was capable of stabilizing its body pitch by continuous control during flight and stance.

©IJRASET: All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue VI Jun 2023- Available at www.ijraset.com

Harbick and Sukhatme achieved to change the apex height by lengthening leg during flight. Le and He controlled the hopping height by fixed leg extension during stance phase. Despite merits of Raibert's three-part control scheme, it still suffers from the vagueness of forward speed control part, in which he did not provide the specification in determination of several parameters.
A single-legged robot inspired by the jumping mechanism of click beetles. It is 85 mm high, 60 mm long, and 41 mm wide, and weighs about 49 g. The robot has good hopping performance that the hopping height is about 4 times – 4.3 times of its body height. It is capable for rescue missions that require to enter enclosed spaces through cracks and narrow channels. In addition, hopping dynamics of the robot is important to understand its jumping mechanism and improve the robot's hopping performance.

III. IMPLEMENTATION OF HOPPER ROBOT

A. Creating a Package

For a package to be considered a catkin package it must meet a few requirements:

1) The package must contain a package.xml file.
2) That package.xml file provides meta information about the package.
3) The package must contain CMakeLists.txt which uses catkin.
4) If it is a catkin metapackage it must have the relevant CMakeLists.txt file.
5) Each package must have its own folder.
6) This means no nested packages nor multiple packages shares the same directory.

B. Meshes

Now we need meshes for the hopper robot. We use the following command to clone the repository. Once the cloning is complete, we should have a new directory with name model for our monoped.

C. URDF Creation

The URDF (Universal Robot Description Format) model is a collection of files that describe a robot's physical description to ROS. These files are used by a program called ROS (Robot Operating System) to tell the computer what the robot looks like in real life. URDF files are needed for ROS to understand and be able to simulate situations with the robot before a researcher or engineer acquires the robot.
Our monoped.urdf file contains 4 links and 3 joints. Moreover, the links have only visual properties which means we cannot yet simulate it. For simulation ability we need to define inertia and collision properties into the monoped.urdf file.

D. Defining a Link

The links in the ROS are defined as follows:

<inertial>
(optional: defaults to a zero mass and zero inertia if not specified) The inertial properties of the link.
<origin>
(optional: defaults to identity if not specified)
This is the pose of the inertial reference frame, relative to the link reference frame. The origin of the inertial reference frame needs to be at the centre of gravity. The axes of the inertial reference frame do not need to be aligned with the principal axes of the inertia.
xyz
(optional: defaults to zero vector) Represents the offset.
rpy
(optional: defaults to identity if not specified)
Represents the fixed axis roll, pitch, and yaw angles in radians
<mass>
The mass of the link is represented by the value attribute of this element
<inertia>
The 3x3 rotational inertia matrix, represented in the inertia frame. Because the rotational inertia matrix is symmetric, only 6 above-diagonal elements of this matrix are specified here, using the attributes ixx, ixy, ixz, iyy, iyz, izz. These can be found for some primitive shapes.
<link name="upperleg">

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue VI Jun 2023- Available at www.ijraset.com

<inertial>
<origin xyz="0 0 0" rpy="0 0 0"/>
<mass value="0.1168" />
</inertial>

<visual>

The visual properties of the link. This element specifies the shape of the object (box, cylinder, etc.) for visualization purposes.

name

Specifies a name for a part of a link's geometry. This is useful to be able to refer to specific bits of the geometry of a link.

<origin>

The visual object.

<mesh>

The reference frame of the visual element with respect to the reference frame of the link.

<geometry>

The shape of a tri-mesh element specified by a filename, and an optional scale that scales the mesh's axis- aligned-bounding-box. Any geometry format is acceptable but specific application compatibility is dependent on implementation.

<material>

The material of the visual element. It is allowed to specify a material element outside of the 'link' object, in the top level 'robot' element.

<color>

The color of a material specified by a set of four numbers representing red/green/blue/alpha(rgba) , each in the range of [0,1].

<visual>
<origin rpy="0.0 0.0 0.0" xyz="0.0 0.0 0.0"/>
<geometry>
<mesh filename="package://my_legged_robots_description/meshes/leg
/upperleg.dae" scale="1 1 1"/>
</geometry>
<material name="blue"/>
</visual>

<collision>

The collision properties of a link. Note that this can be different from the visual properties of a link, for example, simpler collision models are often used to reduce computation time.

<origin>

The reference frame of the collision element, relative to the reference frame of the link.

rpy

Represents the fixed axis roll, pitch, and yaw angles in radians.

<collision>
<origin rpy="0.0 0.0 0.0" xyz="0.0 0.0 0.0"/>
<geometry>
<mesh filename="package://my_legged_robots_description/meshes/leg
/upperleg.dae" scale="1 1 1"/>
</geometry>
</collision>

E. Defining a Joint

The joint element describes the kinematics and dynamics of the joint and also specifies the safety limits of the joint.
The joint element has two attributes:

name

Specifies a unique name of the joint.

type

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue VI Jun 2023- Available at www.ijraset.com

Specifies the type of joint, where type can be one of the following:

1) Revolute - a hinge joint that rotates along the axis and has a limited range specified by the upper and lower limits.
2) Continuous - a continuous hinge joint that rotates around the axis and has no upper and lower limits.
3) Prismatic - a sliding joint that slides along the axis, and has a limited range specified by the upper and lower limits.
4) Fixed - This is not really a joint because it cannot move. All degrees of freedom are locked. This type of joint does not require the axis, calibration, dynamics, limits, or safety controller.
5) Floating - This joint allows motion for all 6 degrees of freedom.
6) Planar - This joint allows motion in a plane perpendicular to the axis.

The joint element has following elements:

<origin>

This is the transform from the parent link to the child link.

xyz

Represents the offset.

rpy

Represents the rotation around the fixed axis: first roll around x, then pitch around y and finally yaw around z. All angles are specified in radians.

<parent>

Parent link name with mandatory attribute.

link

The name of the link that is the parent of this link in the robot tree structure.

<child>

Child link name with mandatory attribute.

link

The name of the link that is the child link.

<axis>

The joint axis specified in the joint frame. This is the axis of rotation for revolute joints, the axis of translation for prismatic joints, and the surface normal for planar joints. The axis is specified in the joint frame of reference. Fixed and floating joints do not use the axis field.

xyz

Represents the components of a vector. The vector should be normalized.

<limit> (required only for revolute and prismatic joint) An element can contain the following attributes:

1. lower (optional, defaults to 0) : An attribute specifying the lower joint limit (radians for revolute joints, meters for prismatic joints). Omit if the joint is continuous.
2. upper (optional, defaults to 0) : An attribute specifying the upper joint limit (radians for revolute joints, meters for prismatic joints). Omit if the joint is continuous.

```
<joint name="haa_joint" type="revolute">
<origin xyz="0.0 0.0 -0.15000" rpy="2.037036 1.570 - 1.107148717794"/>
<parent link="base"/>
<child link="hipassembly"/>
<limit effort="2000" lower="-1.6" upper="1.6" velocity="20.0"/>
<axis xyz="0 0 1"/>
</joint>
```

F. Defining a Transmission

The transmission element is an extension to the URDF robot description model that is used to describe the relationship between an actuator and a joint. The transmission element has one attribute:

name (required)

Specifies the unique name of a transmission.

©IJRASET: All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue VI Jun 2023- Available at www.ijraset.com

The transmission has the following elements:
<type> (one occurrence)
Specifies the transmission type.
<joint> (one or more occurrences)
A joint the transmission is connected to. The joint is specified by its name attribute, and the following sub-elements:
<hardwareInterface> (one or more occurrences)
Specifies a supported joint-space hardware interface. Note that the value of this tag should be EffortJointInterface when this transmission is loaded in Gazebo and hardware_interface/EffortJointInterface when this transmission is loaded in RobotHW.
<actuator>
An actuator the transmission is connected to. The actuator is specified by its name attribute, and the following sub-elements:
<mechanicalReduction>
Specifies a mechanical reduction at the joint/actuator transmission. This tag may not be needed for all transmissions.
<hardwareInterface> (optional) (one or more occurrences)
Specify supported joint-space hardware interface. Note that <hardwareInterface> tag should only be specified here for ROS releases prior to Indigo. The correct place to specify this tag is in <joint> tag.

```
<transmission name="haa_joint_trans">
<type>transmission_interface/SimpleTransmission</type>
<joint name="haa_joint">
<hardwareInterface>hardware_interface/EffortJointInterface
</hardwareInterface>
</joint>
<actuator name="haa_jointMotor">
<hardwareInterface>hardware_interface/EffortJointInterface
</hardwareInterface>
<mechanicalReduction>1</mechanicalReduction>
</actuator>
</transmission>
```

G. Creating World File

Now we will simulate this robot. To do so first we need to create a world file. We should create a directory named worlds inside the below path.
Path: ~simulation_ws/src/my_legged_robots_sims/ directory.
Using the IDE create a file low_gravity.world inside worlds directory. In this file the tag gravity helps us to manipulate the gravity inside the gazebo world.

H. Launch Files

Next, to launch the world we need to create a launch file main.launch. Before creating the file, create a launch directory inside the following path.
Path: ~simulation_ws/src/my_legged_robots_sims/ directory.
This launch file will only launch an empty world in gazebo with zero gravity. To spawn the monoped we need to create another launch file inside the launch directory.
This main.launch file includes two files, spawn_monoped and monped_control launch files.

```
<include file="$(find my_legged_robots_sims)/launch/spawn_monoped.launch"/>
<include file="$(find my_legged_robots_sims)/launch/monoped_control.launch"/>
```

1) Spawn_monoped Launch File

We have to include the urdf file to spawn the robot.

```
<include file="$(find my_legged_robots_sims)/launch/spawn_monoped.urdf"/>
```

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue VI Jun 2023- Available at www.ijraset.com

2) Monoped_control Launch File

To launch the control, we need a monoped_control.launch file. Here we have included the yaml file and also the controllers (joint state controller, haa_joint_position_controller,hfe_joint_position_controller,kfe_joint_position_control ler). Yaml allows things like vectors . It also allows putting some parameters into a nested namespace. These namespaces are relative to the yaml file's own namespace. Yaml files allow parameters with complex types, nested namespaces of parameters, and reusing the same parameter values in multiple places.

<rosparamfile="$(findmy_legged_robots_sims)/config/monoped.yaml"command="load"/>

I. Defining a Sensor

The sensor element describes basic properties of a visual sensor .It contains two attributes.

name (string)

The name of the link itself

update_rate (optional) (float) (Hz)

The frequency at which the sensor data is generated. If left unspecified, the sensor will generate data every cycle.

It has the following elements:

link (required) (string)

The name of the link this sensor is attached to.

<origin> (optional: defaults to identity if not specified)

This is the pose of the sensor optical frame, relative to the sensor parent reference frame. The sensor optical frame adopts the conventions of z-forward, x-right, and y-down. **xyz** (optional: defaults to zero vector)

Represents the offset.

rpy (optional: defaults to identity if not specified)

Represents the fixed axis roll, pitch, and yaw angles in radians.

<gazebo>
<pluginname="gazebo_ros_imu_controller"filename="libgazebo_ros_imu.so">
<robotNamespace>/monoped</robotNamespace>
<topicName>imu/data</topicName>
<serviceName>imu/service</serviceName>
<bodyName>base</bodyName>
<gaussianNoise>0</gaussianNoise>
<rpyOffsets>0 0 0</rpyOffsets>
<updateRate>50.0</updateRate>
<alwaysOn>true</alwaysOn>
<gaussianNoise>0</gaussianNoise>
</plugin>
</gazebo>

a) IMU Sensor

An IMU (Inertial Measurement Unit) is a specific type of **sensor** that measures angular rate, force, and sometimes magnetic field. Technically, the term “IMU” refers to just the sensor, but IMUs are often paired with **sensor** fusion software which combines data from multiple **sensors** to provide measures of orientation and heading. We need to use an IMU sensor. This type of sensor can measure and report the robot's specific force, angular rate, and the magnetic field surrounding the robot in all three directions (X, Y and Z) and for sure for the helping hand.

b) Lower Leg Contact Sensor

This is the sensor placement for the lower portions of the body, used for applications such as running and knee control. It gives data about whether the monoped is in contact with the ground.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

c) Odometry Sensor

Odometry is the use of data from motion sensors to estimate change in position over time. Odometry is the use of motion sensors to determine the robot's change in position relative to some known position. For example, if a robot is traveling in a straight line and if it knows the diameter of its wheels, then by counting the number of wheel revolutions it can determine how far it has travelled. It is used in robotics by some legged or wheeled robots to estimate their position relative to a starting location of the robot.

IV. RESULTS

Locomotion is one of the most challenging topics in robotics. Creating the hard- coded algorithms and the kinematics models is not an easy task. Single Legged hopper robot with Lower leg contact sensor, IMU sensor and Odometry sensor is designed and launched in the gazebo world. We created a Monopod robot simulation working in Gazebo. It allows you to separate learning algorithms from the physical/simulated robot, so that you can test different learning algorithms easily. The robot is launched in the world as shown in Fig. 5.1, data from IMU sensor is published is shown in Fig. 5.2, data from Odometry sensor is published in Fig. 5.3 and the data from Low Leg contact sensor is published in Fig. 5.4.

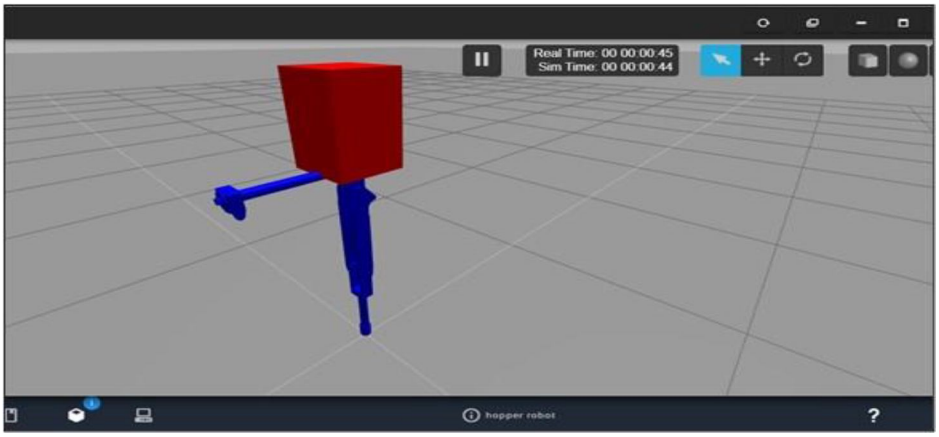


Fig. 5.1 Hopper Robot

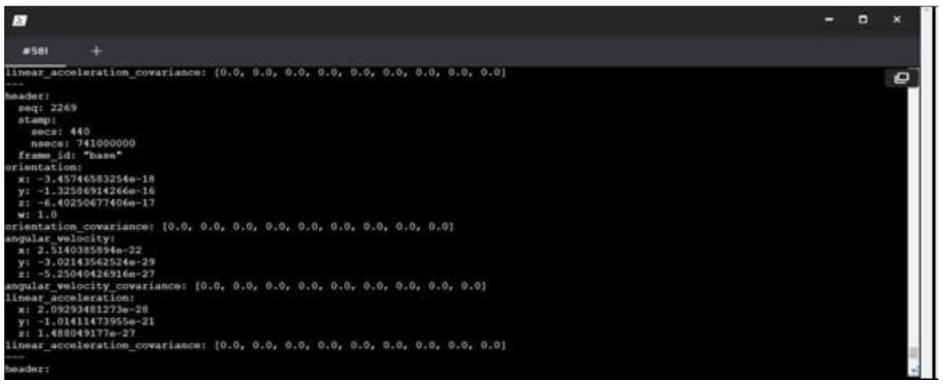


Fig. 5.2 Data published from IMU Sensor

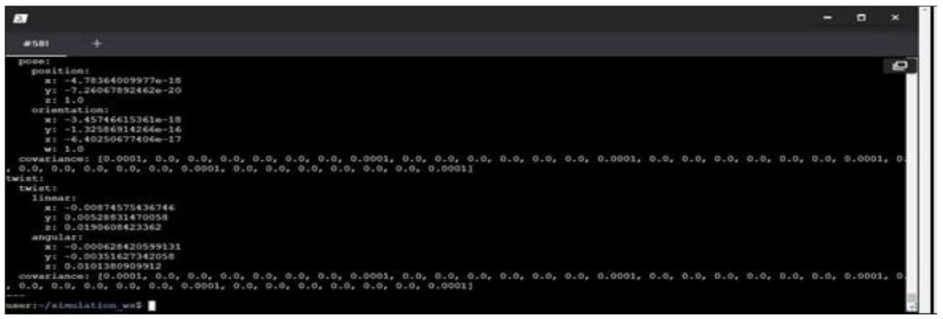


Fig. 5.3 Data published from Odometry sensor



Fig. 5.4 Data published from Low Leg contact sensor

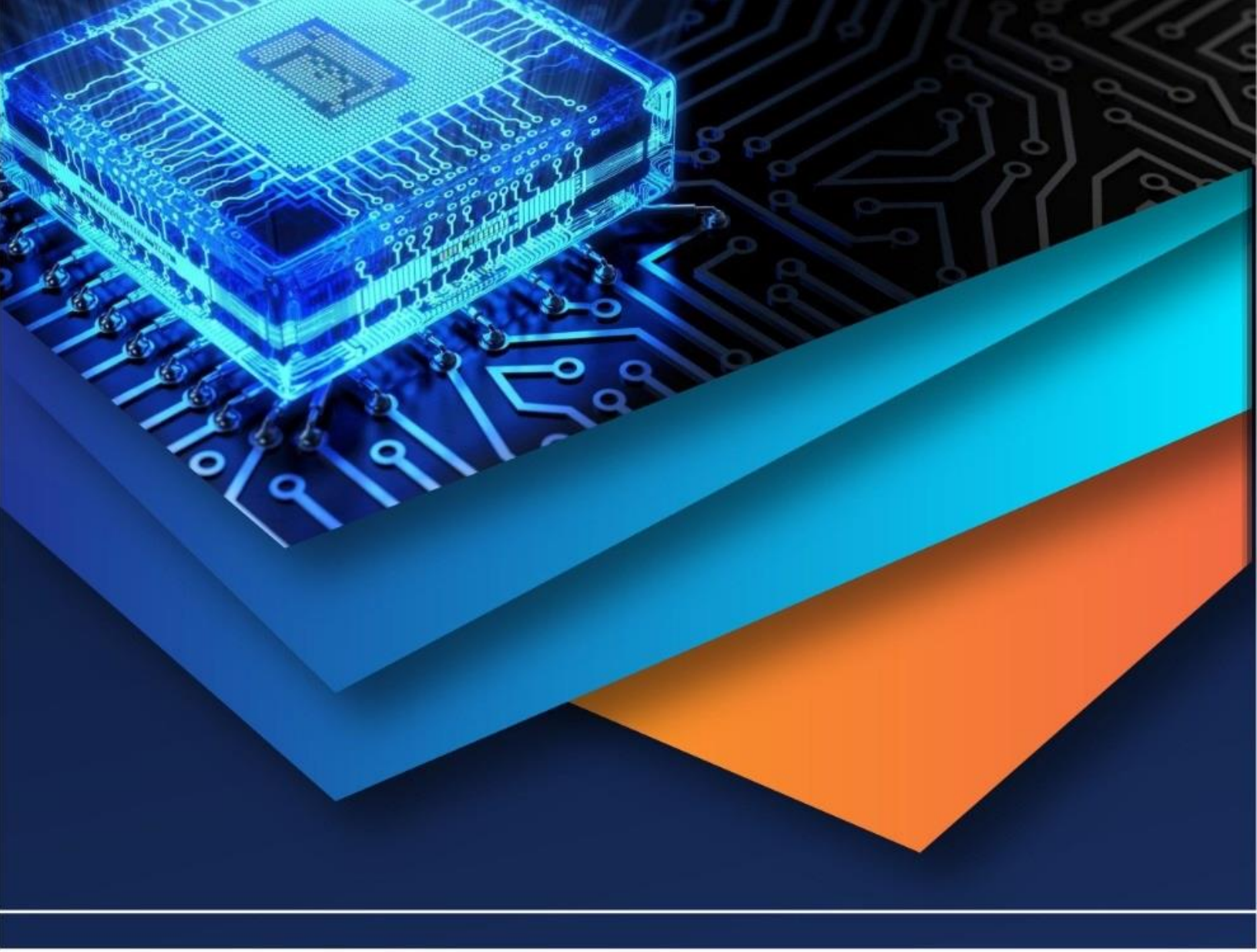
V. CONCLUSION

Using simulation tool gazebo along with meta-operating system ROS, Hopper robot is implemented which has been tested under various conditions. The main aim of the project is to replace the man power in hazardous situations like natural calamities, going through uneven plains, plateaus for rescue operations and even helps in the field of research. This is successfully achieved in the project. In comparison to multi legged robots, single-legged robots have only one type of locomotion gait, i.e., hopping, which represents a highly nonlinear dynamical behaviour consisting of alternating flight and stance phases. This robot when implemented with hardware components with small modification can be used for various real-life scenarios like planetary explorations, military operations etc.

REFERENCES

[1] Sayyad, Ajij; Seth, B; Seshu, P.Robotica, “Single-legged hopping robotics research-A review” Cambridge Vol. 25, Iss. 5 (Sep 2007)
[2] Benjamin Stephens, “Humanoid Push Recovery, Humanoids” , (2007 IEEE)
[3] J. Pratt, J. Carff, S. Drakunov, A.Goswami, “Capture Point: A Step toward Humanoid Push Recovery” , Humanoids’06 (2006 IEEE)
[4] K. Harbick and G. Sukhatme, “Height control for a one-legged hopping robot using a one-dimensional mode” , Technical Report (IRIS-01-405, Institute for Robotics and intelligent Systems, USC, 2001)
[5] G. Zeglin, B.S.Thesis, Uniroo, “A one legged dynamic hopping robot”, (May 1991)
[6] Z. Li and J. He, “An Energy Perturbation Approach to Limit Cycle Analysis in Legged Locomotion Systems”, Proceedings of 29th IEEE Conference on Decision and Control, Honolulu, USA (Dec. 1990)
[7] M. H. Raibert, “Legged robot that balance”, Cambridge, MIT Press, 1986
[8] P.Gregorio, M. Ahmadi, and M. Buehler, “Design, Control, and Energetics of an Electrically Actuated Legged Robot” , IEEE(Feb. 1986)
[9] https://ieeexplore.ieee.org/document/8266176
[10] https://www.sciencedirect.com/book/9780128037669/bioinspired-legged- locomotion
[11] https://www.intechopen.com/books/numerical-analysis-theory-and- application/dynamics-and-control-for-a-novel-one-legged-hopping-robot-in- stance- phase

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)