DE-42 (MTS) TALHA, TAHA, RASHID

AUTONOMOUS NAVIGATION OF WHEELED QUADRUPED ROBOT

COLLEGE OF
ELECTRICAL AND MECHANICAL ENGINEERING NATIONAL
UNIVERSITY OF SCIENCES AND TECHNOLOGY RAWALPINDI
2024

COLLEGE OF ELECTRICAL AND MECHANICAL ENGINEERING

DE-42 MTS
PROJECT REPORT

AUTONOMOUS NAVIGATION OF WHEELED QUADRUPED ROBOT

Submitted to the Department of Mechatronics Engineering
in partial fulfillment of the requirements
for the degree of
Bachelor of Engineering
in
Mechatronics
2024

[illegible]

Supervisor:
Prof Dr. Umar Shahbaz Khan
Prof Dr. Kunwar Faraz Ahmad Khan

Submitted By:
Talha Ahmad Shaikh
Muhammad Taha
Muhammad Rashid Ali Khan

ACKNOWLEDGMENTS

For our thesis on the project Autonomous Navigation of Wheeled Quadruped Robot, we are grateful for Allah Almighty. As His resource, we extend our gratitude for Dr. Umar Shahbaz Khan, our Supervisor and Dr. Kunwar Faraz Ahmad Khan, our co-Supervisor for their continuous guidance, encouragement, and resources. We are also thankful to the National Centre for Robotics and Automation (NCRA) for funding and supporting our project and huge credits to Engineer Hassan-Ullah for being our mentor by guiding, supporting and assisting us.

ABSTRACT

In the era of Industry 4.0, leading industries are adopting sustainable solutions in their workplace. Industrial inspection is one such task that requires high precision and can be hazardous in compromised areas. Unmanned robots are being presented as a solution to this problem and they are performing well above the mark. In this regard, a project of an autonomous wheeled quadruped robot is presented that is a cost-effective design for a convertible wheeled to-legged robot that can adapt to any environment and can easily navigate autonomously. It has vision capabilities on-board which can be accessed remotely and can be used for object detection, tracking, and identification. It can be used in hazardous environments and can be equipped with any sensor. Its software solution is distributed as ROS packages making it highly modular, easy to customize, and upgradeable. The robot is designed using 3D printing technology. It uses vision and ranging sensors to generate area maps and localizes itself in any mapped or unmapped environment. Upon giving a destination, it can generate an optimal path plan. This plan is utilized to navigate the robot to its destination. We are using FPGA as the robot low level control. Using FPGA for motion control provides parallel computation of gait motion for synchronized joint response. Furthermore, the high frequency adjustable clock of FPGA reduces the response time resulting in swift action. The robot can use its vision sensors to detect, identify and track objects in real time. Furthermore, it has the capability to detect faces on custom trained dataset and can scan QR codes. It can be navigated remotely using remote server configurations.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

Acronyms

DC	Direct Current
mAh	Milli Amp Hour
FPGAs	Field Programmable Gate Arrays
ROS	Robot Operation System
LIDAR	Light Detection and Ranging
IMU	Inertial Measurement Unit
RTK	Real Time kinematics
3D	Three Dimensions
2D	Two Dimensions
V-SLAM	Visual Simultaneous Localization and Mapping
YOLO	You Only Look Once
SORT	Simple Online and Real Time Tracking
R-CNN	Region-based Convolutional Neural Network
GPS	Global Positioning System
QR code	Quick Response code

Chapter 1 – INTRODUCTION

1.1. Overview

With the advent of Industry 4.0, modern industries are aiming for sustainable solutions that are environmentally friendly and labor friendly. They are aiming for smart solutions. The industry is always looking for solutions which are easy to implement and have good output. There is a requirement for practices that can be more sustainable and productive as compared to traditional ways of doing things. It is because with time there is a need for better solutions, which have low cost and are more efficient than traditional ones. Heavy emphasis is given to safe industry practices that are economically viable too. In the field of inspection, unmanned robots are emerging that are equipped with cutting-edge technology to replicate and improve the inspection process while ensuring worker safety in hazardous environments. They have very many capabilities can be deployed in many different environments and to many advantages as compared to manned vehicles. They have a human at a safe distance operating on them so that he can be safe. They should have capabilities that they must be able to do their work in hazardous environments without the presence of humans at the site. The robot must be able to take decisions and be able to give input about problems efficiently and be able to detect the problems of site using sensors and algorithms. The goal is to make human life safe and secure because these robots can do their work and humans just must monitor it by staying at a distance from the site. It increases the productivity of the work as the robot can work without tiring and does not need rest. The power problems can be managed very easily in any robot. In this regard, a cost-effective autonomous wheeled quadruped robot is presented that can automate the manual inspection routine tasks. It can perform autonomous navigation in mapped and unmapped environments and can be safely deployed in hazardous environments. The robot has been deploying processors, sensors, and algorithms so that it can work efficiently and give maximum output possible. The robot can switch between legs and wheels, each one having its pros and cons. The robot can work in different terrains and adopt different modes by using algorithms that are implemented inside it. Based on sensors, it can navigate in an environment. The robot can navigate swiftly in wheeled mode while in an uneven environment, it can switch to legged mode. The robot can be equipped with sensors that can detect different problems in case of any inspection industry in which it is deployed.

1.2. Significance

The salient features of the robot are its cost-effective build, smart algorithms, sensor fusion, and terrain adaptation ability with a focus on using Field Programmable Gate Arrays (FPGAs) technology to develop a robust control system for the robot. The goal is to make the robot autonomously navigate over an indoor environment for inspection without the intervention of any human. The robot must have the capability to work in harsh conditions and hazardous environments. Programs are distributed as packages of Robot Operation System (ROS) which makes the robot modular and highly adaptive to any unmanned mapping and exploration. ROS is used to control the overall functioning of the robot. From navigation to controlling LIDAR and IMU, ROS is used. There has been use of SLAM for the purpose of localization, mapping, and navigation of ROS. Different modules of ROS have been used to take data from sensors, implement SLAM, and then give instructions to FPGA for actuation. The SLAM has been implemented with the help of ROS packages, LIDAR and IMU. The ROS modules and topics have been used to implement localization and mapping. Localization and mapping have been the main goals of navigation. The modules are in the form of ROS topics which are used for the normal functioning of the robot. The robot uses the sensors for localization and to map the environment with the help of hector slam in ROS. The hector SLAM detects all the obstacles and all the surroundings and then makes the map on Rviz, a module in ROS used for the visualization of the map. The robot has been 3D printed parts by parts and then joined manually. The robot's final functioning has been done after trying different approaches and then selecting the final approach. The robot is equipped with an onboard processor that can do all the required processing, can control all the sensors, can give instructions to the FPGA board, and handle extra sensors in case of an application for which it is deployed. The robot has very sensitive and precise movement because of parallel instructions given to motors by FPGAs as the controller for the robot. The FPGA is deployed only for the smooth control of motors so that there is not any problem involved in actuation and it is smooth.

<u>Chapter 2 – LITERATURE REVIEW</u>

2.1. Background Review

Humans are always curious to build legged robots or systems which can easily navigate through difficult terrains. Humans are struggling to develop machines which can work in indoor, outdoor, and unconventional environments. In this section. work done and development in this field will be discussed.

2.2. Introduction to Autonomous Navigation and Quadruped Robots

Various kinds of robots are required for performing different tasks. An autonomous robot can perform tasks based on the inputs from the environment without any human intervention. However, terrain adaptation is a significant feature of an autonomous robot. Different terrains require different kinds of robots. For performing some kind of task on the road you require a robot which has wheels. The wheel-based robots cannot work in uneven and complex terrain. The legged platform has the capability to work in every kind of terrain. It can move in muddy, complex, and off-road conditions. Animals use their legs to move in complex terrains with maximum efficiency. They have great agility because of their legs. So, humans have always strived to develop legged platforms so that they can help in different scenarios. The wheeled legged robot has great implications for working in different kinds of terrains. Depending upon the terrain and its situation it can switch between legs and wheels.

2.2.1. Importance And Relevance in Current Robotics Research

The aim is to develop a local fully autonomous robot system which can work in an industrial environment, can reach its destination, and inspect an industrial setup. So, the system developed will help in industrial places which are hazardous and inaccessible for humans. The robot will provide valuable help in current research in locomotion and navigation of quadruped robots. It will help in developing new kinds of sensor fusion and navigation systems for the mobile robotics field. Using cameras, LIDARS and IMU can help in developing new robotic perception systems.

2.2.2. Historical Development of Quadruped Robots

In the 15th century Leonardo Da Vinci succeeded in materializing this idea by constructing articulated anthropomorphic robots [1]. This was the biggest breakthrough of his time in this field. Later, in the 1960s and 70s, advancements were made in this field. In

1968 R. Mosher General Electric quadruped robot was developed which was a landmark. Then in the 2000s the BigDog was developed by Boston dynamics [1]. It was initially developed for soldiers as a carrier for luggage and other stuff. It can easily work in rough terrains. Then Spot robot was developed by Boston dynamics which is the state of art in the category of four-legged robot. Today there are multiple companies which are developing the quadruped robot and wheeled quadruped robots for a variety of applications.

2.3. Types of Quadruped Robots

2.3.1. Legged Robot

The legged robot can work in complex terrains including rocks and rough terrains. They can even walk on stairs which make them efficient for different types of applications. They can easily pass through obstacles in different ways, increasing their applications in industrial areas. On the other hand, they need complex mechanical systems and highly efficient sensors and algorithms. They are also very costly and difficult to maintain. They are comparatively energy inefficient and slow.

2.3.2. Wheeled Legged Robot

Having both wheels and legs, they can work both on road and off road, they can walk through any uneven terrain and on road with high speed. The wheel legged robot can find their applications in any kind of mobile robots. This makes them highly capable for search and rescue applications. They have some disadvantages like they need a highly precise system to switch between legs and wheels which need intelligence as well as high quality sensors which can give highly precise information about the internal as well as external state of the robot. This makes them highly expensive.

2.4. Navigation

In a quadruped wheeled robot, it is important to have a successful system of navigation in it. For any kind of indoor and outdoor application, it is necessary that the robot reaches its destination with accuracy. Successful navigation creates the opportunity for the robot to reach the destinations easily and make its applications diverse. There are multiple approaches available for the robot to navigate in indoor and outdoor environments. A variety of hardware and algorithms can be used to reach a specific point. Navigation has always been a key aspect for mobile robotics. In literature, significant work has been done on navigation for mobile robots due to its necessity for the robot.

An excellent navigation system helps you to do a variety of tasks. For example: delivering different kinds of products, industrial inspection, and maintenance, working and inspection of hazardous unknown terrain, exploration and for search operations. To make a robot achieve its destination successfully complete navigation includes realization of your location, information of map, path planning and obstacle avoidance algorithm. There can be multiple ways for the robot to acquire data about the map and to localize itself inside the environment. Navigation includes the tasks given below.

2.5. Components of Navigation

2.5.1. Localization

Localization plays a vital role in navigation of a mobile robot. Localization is the present location of the robot as compared to the environment. Different kinds of sensors are used for this.

2.5.2. Path Planning

The second aspect of the navigation is path planning. It is planning to start from where the robot currently is and where it must reach as efficiently as possible. A perfect path is one which has minimum obstacles, shortest time and can reach its destination without any collisions.

2.5.3. Mapping

The third key aspect of navigation is mapping. It is the creation of the environment of the surroundings. Mapping uses sensor data to create a 3D or 2D map.

2.5.4. Approaches

There are several types of navigation systems for robots in indoor environments. The mini spot developed by Boston Dynamics is one of the best in the category of legged robots. It uses a system called GraphNav which is used for localization, mapping, and autonomous movement. The system works as a graph of way points and edges. The robot uses the graph to reach its destination. The maps are recorded by the spot with the help of a controller. It constructs a map which is composed of way points and edges. The localization is done by using it. The robot uses it to navigate with the help of a map. It matches the real time sensor data and the recorded map and tries to stay as close to the recorded map as possible.

One of the best robots for industrial inspection Is ANYmal. It uses LIDAR for localization and mapping. It used cameras for information about different kinds of obstacles. By using graphical user interface. The user can give the location of the inspection and it autonomously plans the path and reaches the destination for inspection.

2.6. Path Planning

Path planning is the autonomous movement of a robot that can move from the initial point to the goal by following the shortest and most optimum path. In path planning, it must be easy for the robot to move from one point to another, avoid all distractions, and reach the destinations. It is the most basic requirement of a robot so that it can work autonomously without human intervention and correction. It is very necessary for a robot to plan a path that is the most optimum using available resources. It is necessary because it can save time and energy, which is necessary in the case of a robot as it has limited energy.

In the case of a four-legged robot, it is very necessary that a good algorithm must be used so that the robot can move from the initial point to the final point autonomously so that in the future it can work in real-world applications very easily. The user is just giving the final point and robot itself with the help of algorithms plan the most optimize path and then by avoiding obstacles reach the destination.

There are multiple algorithms available that can be used for the path planning in the robot but there are multiple factors that can be key factors in deciding whether a particular algorithm should be used or not. The following are the factors that influence the selection of algorithms.

- Complexity of environment
- Type of navigation
- Real-time requirements
- Computational power of microprocessor available

All these factors must be considered while selecting a path-planning algorithm. In this case, an algorithm is needed that can move from the starting point to the goal following the most optimum path and working under the available computational sources. The following are the algorithms that are considered in this project:

- Bug algorithm
- Probabilistic road map algorithm
- ROS navigation stack
- Potential field algorithm for path planning

2.7. Bug Algorithm

It is the most basic algorithm. It follows a straight line until an obstacle comes under its way and when that happens it follows the boundary of the obstacle until it can resume its locomotion toward the goal again. It can work in an environment which are simple and does not require any complex scenarios and environments.

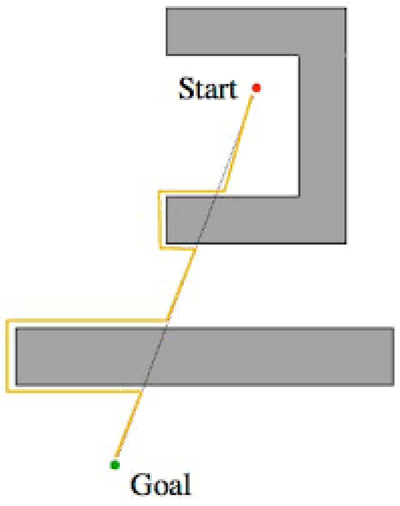


Figure 1 - Bug Algorithm (https://spacecraft.ssl.umd.edu)

Although it comes with certain advantages due to obstacle following method sometimes it may not follow the shortest path and sometimes it can reach a dead end and may be not able to reach the goal. The complexity of this algorithm Is very less as compared to other algorithms due to its simplicity.

2.8. Potential Field Algorithm

The potential field algorithm is also known as the artificial potential field method. In this algorithm, the whole environment is a potential field where the starting point is of the highest potential and the destination is of the lowest potential in whole field. obstacles are represented as objects of repulsive potential where potential increases as robot moves toward them. The robot moves by calculating the potential gradient at every point and then robot moves itself toward the direction of decreasing potential. It can provide a simple approach and easy obstacle avoidance. But it also has certain disadvantages like sometimes it gets stuck in local minima. It may fail in cluttered environments and narrow passages. Its computational power depends upon different factors like the complexity of the environment, and the number of obstacles in the environment. Complexity increases with these factors.

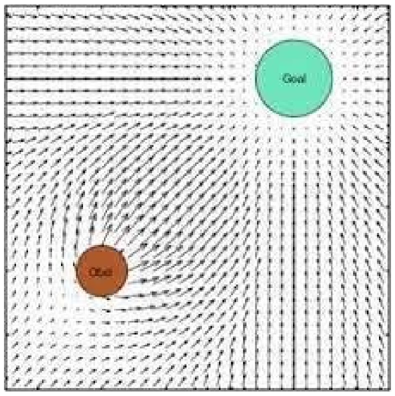


Figure 2 - Potential Field Algorithm (https://medium.com/@rymshasiddiqui)

2.9. Probabilistic Road Map

PRM stands for probabilistic road map algorithm and is an effective algorithm for path planning in case of complex environments. In these random points (nodes) are spread on the map representing the robot's position at any point. Then it Is checked that is that point is in free space, or it lies inside an obstacle. If any node is an obstacle, then it is discarded. Then all free points are connected to their neighboring points and is checked that the lines between points are in free space Then a graph is constructed where all nodes are connected together and this graph represents a road map of the complete environment, by using any shortest path finding algorithm like A* or Dijkstra the most optimum path to the goal is found by series of dots getting to goal efficiently.

It has many advantages like working in complex environments and a smooth collision-free environment. It has several disadvantages like it requires more computational power which depends on several random points required and may not be able to find a most simple path if the road map is not very dense.

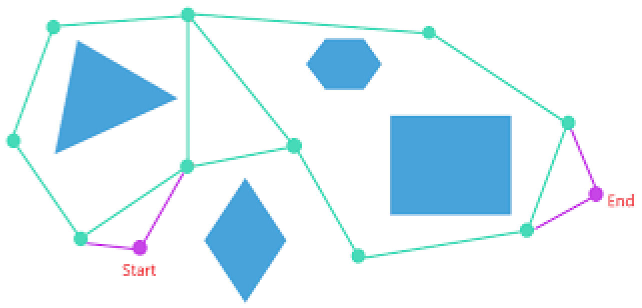


Figure 3 - Probabilistic Road Map (https://medium.com/acm-juit)

ROS is a source operating system which provides an implementation for the intercommunication of ROS applications. The following are the components of the ROS:

- Base framework
- Collection of supporting libraries for the development of applications

ROS allows you to integrate the hardware and software and achieve results with the help of it. Many international robotics companies are using ROS for simulations and real-time applications. They have developed open-source library tools for robotic engineers. Companies like ANYbotics and Boston Dynamics are using it and have made their platform online.

ROS is not an operating system like Windows or Linux. It is a middle framework that works on any operating system. It can be used in C++ or Python language. It relies on existing systems to manage hardware, software and handle tasks while working. It does not control the hardware, rather it provides software tools to control the hardware.

2.10. Applications

- It is used for connecting hardware and software elements. There are ROS topics that act as connectors for connection between them.
- It is used for connection between many robots and from a network.
- It is used for the implementation of different in-built libraries to achieve results. There is no requirement to learn library algorithms. As it is open source then by using it anyone can use them in ROS without learning them completely which makes robotics implementation easier. For example: there is a requirement to implement a SLAM (simultaneous localization and mapping algorithm) for making a map of the environment. Then there is an algorithm for it called Hector SLAM. Anyone can use it very easily without learning the basics of hector slam.
- By using tools available in ROS, instead of real-time experimentation, the sensors and

hardware can be tested in ROS and then according to the results, they can be shifted to real-time. For example: Gazebo can be used for the simulation of the robot.

- Robotics industry relies upon the ROS. So, it has a wide range of open-source tools available to get the results. From large-scale robots to small robots, it has a wide range of libraries to comply with desired applications.
- It can be used in a wide range of robots from indoor to outdoor, mobile to stationary robotic systems, and for robots which have capabilities to work in different kinds of terrains. It gives you leverage to simulate and get results in different types of domains.
- As it is open source so relying on previously existing libraries, by updating them anyone can come up with more improved and advanced libraries which are requirements for innovation.
- By using ROS, 3D simulations can be done to test the robot.

2.11. Working of Robot Operating System

The following are components in the working of ROS:

2.11.1. ROS master

It is the central part of ROS. It provides communication between different PARTS of Robot Operating System. It resolves issues and connects nodes and services.

2.11.2. Publisher

A publisher is a node that sends information to a topic so that it can be used by other nodes for different tasks. For example: a node handling a camera gives its data to a topic which gives it to an object detection node. In this case the publisher the camera handling node.

2.11.3. Nodes

Individual elements in ROS are called nodes. They have a main task which they perform during any operation. It can vary, like taking data from sensors, controlling some kind of hardware and processing, and analyzing the information. For passing information from nodes to nodes, they use publishing and subscribing and using different services.

2.11.4. Topics

They predominantly are the communication mechanism in ROS. They are used by nodes to transfer messages for the purpose of working of system. Nodes can use topics in two ways:

- By becoming publishers of a topic, nodes can transfer data to a node which can be used by another node.
- By becoming subscriber, nodes can receive messages from a topic which is communicated to it by another node.

2.11.5. Subscriber

A subscriber is a node that receives information from a topic which it uses for performing its task. For example: a node handling a camera gives its data to a topic which gives it to an object detection node. In this case, the subscriber is that object detection node. Every subscriber node receives a copy of message as it is sent by publisher.

2.11.6. Messages

They are basically the communication structure between nodes via topics. They define the format and content of data to be transferred. ROS has a variety of messages for taking data from sensors, passing information to actuators, and for many other tasks inside a ROS system. For communication messages are serialized before sending and then serialized when they are received at another end.

2.11.7. Services

Besides receiving the general information, the subscriber can request a specific type of message, this is called services in ROS, for example: a subscriber node requires a message in case of a specific angle of the robot which will be provided by the publisher node in case of request.

2.11.8. Launch Files

They allow you to use and execute multiple nodes by using a single command. Instead, a person is running multiple nodes, he makes a launch file and runs them all once. It saves their time and helps to maintain and reuse the configurations.

2.11.9. Utilities

There are many tools and packages which are used for using ROS. They are basic commands which are used in every type of ROS implementation. They help to make robot operating system an easier to use system. They provide functionalities to use ROS nodes and interacting with ROS messages and packages.

2.12. Navigation Stack

The ROS (Robot Operating System) is a collection of packages in ROS1 that provide capabilities so that the robot can move autonomously avoiding obstacles. It takes information about the environment and robot from the different sensors and then by using this information and applying algorithms it plans the path. The ROS navigation stack then integrates with ROS packages to effectively complete the movement of the robot from initial to goal so that it can reach its destination successfully. The following are the key components of the ROS navigation stack:

2.12.1. Map Server

It is the primary component responsible for providing access to the occupancy grid of the robot environment. It is responsible for the maintain the static map which is the basic map of the environment. The map is made with the help of LIDAR and other sensors. It deals with dynamic maps, interfacing with ROS topics, querying map data and updating maps as well as publishing data for other topics.

2.12.2. Localization

Localization is the component in which the robot recognizes its surroundings with the help of surroundings and estimates where it is currently on the map. It is very necessary as it is crucial so that the robot can reach its destination efficiently. localization requires a map of the environment which can be obtained through algorithms like SLAM.

2.12.3. Mapping

It is the process of creating and updating the map of the environment in which the robot must work autonomously. There are multiple mapping techniques available that can be integrated with ROS to achieve mappings like Gmapping, hector SLAM, and cartographer approach, each approach requires different sensors like LIDAR, IMU, odometry sensors, and cameras to make the map of the environment. In the mapping process, the map is continuously updated with the help of sensors to reach the destination. The map can be visualized with the help of RVIZ available in ROS.

2.12.4. Global Planner

The global planner is responsible for maintaining a high-level path from the starting point to the goal point. This path involves a sequence of points that the robot requires to move from the initial point to the goal point. the path must be collision-free to reach to the destination.it has multiple algorithms like Dijkstra algorithms, A* algorithm, and potential field algorithms.

2.12.5. Local Planner

The local path planner is responsible for making and following a low-level path for the robot based on the global planned path by the navigation stack algorithm. It ensures that the robot moves to the goal without any collision and smoothly. By taking robot's current pose and position, it makes a trajectory that should be followed to achieve specific navigated goals. It should follow a certain path that should be adhering to the global planned path.

2.12.6. Global Positioning System

GPS stands for global positioning system. It holds a key position in navigation for mobile robots. As it is always needed for the location of the robot, GPS is the best for it. In [2], the navigation is done with the help of the GPS module, which is embedded with Raspberry Pi, Arduino Mega on a mobile robot. The algorithm developed is working based on the latitude and longitude of both destination and current location. By using the haversine formula and then finding the standard deviation, the deviation from the path can easily be measured. So, in this way the algorithm helps to stick with the path. In [3]. The GPS and the LIDAR are used for mapping and localization. The GPS is used to get the current location and to get the update at every point of the path. A manometer is used to provide the direction in which the robot is moving so that any deviation from the destination can be checked easily.

2.12.7. Real Time Kinematics

GPS has an accuracy of 1-5 meters. So, there is demand for the increase in accuracy to inches level. RTK (Real Time kinematics) is a technique used to enhance the accuracy of the data obtained from GPS. It has very important applications for mobile robots. It has two stations, one reference station and a mobile receiver. The location of the receiver is known precisely. Then it compares its location with the real time location by satellite and calculates the error. Then this error correction is sent to the mobile receiver so that it can calculate its position accurately according to the error correction.

So, in [4] two RTK GPS and IMU are used for determination of attitude and localization. The two GPS and RTK are used in parallel and in accordance with the base station. The approach lessens the error in robot location due to GPS errors. The two GPS modules with RTK by using information from satellites provide their distance from the base station, by this the robot location can be determined more accurately. The approach estimates the covariance matrix based upon GPS measurements.

2.12.8. Simultaneous Localization and Mapping

It is used in mobile robots and these types of applications to determine the position and orientation of the robot as well as creating the map of the environment. The slam does it simultaneously. SLAM can work in unknown environments as well as in known environments. It uses different kinds of sensors like LIDAR, cameras, and inertial sensors. By using the data from sensors, it performs localization and creates a map of the surroundings.

2.12.9. Light Detection and Ranging Coupled with Inertial Measurement Unit

LIDAR and IMU fused together are a good combination for navigation and control. IMU gives you information about the pose of the robot while the LIDAR gives you information about the surroundings. By fusing them together a successful autonomous system can be achieved. One of the best approaches is by combination of IMU and LIDAR.in [5], a combination of IMU LIDAR and leg odometry is applied to get the desired results. For sensor fusion the LIDAR and IMU are fused with the Kalman filter algorithm.

LIDAR and IMU fused together are a good combination for navigation and control. IMU gives you information about the pose of the robot while the LIDAR gives you information about the surroundings. By fusing them together a successful autonomous system can be achieved. One of the best approaches is by combination of IMU and LIDAR.in [5], a combination of IMU LIDAR and leg odometry is applied to get the desired results. For sensor fusion the LIDAR and IMU are fused with the Kalman filter algorithm.

2.12.10. Light Detection and Ranging Coupled with Camera

LIDAR and Camera coupled together are the best combination for sensing the environment and for navigation. By fusing the data with the most efficient sensor fusion algorithm, the best navigation is possible. In the VILENS approach discussed in [6], the visual, LIDAR, Inertial and leg Odometer sensor modules are fused tightly to achieve the maximum results. The measurements of these all sensors are synchronized according to technique given in [7].

In [8], an autonomous navigation system is developed for BigDog robot using laser scanner stereo vision system and algorithms. It scans the real world and then constructs maps in 2D. by using A* algorithm, it plans the path and follows the path. IMU is also used for checking the orientation of BigDog.

2.12.11. Visual Simultaneous Localization and Mapping

V-SLAM stands for visual simultaneous localization and mapping.it relies on data from visual sensors such as cameras to determine its position and create the map of the environment. It can also incorporate data from multiple sensors. In [9], a novel approach is discussed using V-SLAM. By using a three-camera stereo vision system and surf algorithm (computer vision), it navigates. The algorithm developed in this approach not only performs localization and mapping but also Is helpful in back tracking and exploring unknown environments.

2.12.12. Sensor Fusion

Sensor fusion is an important part for any mobile robot. From performing any task to navigation, a good sensor fusion achieves a better outcome. It is necessary to use the best approach possible and fuse sensors in a way that there is maximum input from any sensor and there is cancellation of noise and removal of useless readings and data. In [5] for the IMU and LIDAR are fused with the help of Kalman filter algorithms to obtain real time results. In [10], a pronto approach is used which uses extended Kalman filter for leg odometry that uses IMU and leg odometry.

2.12.13. Obstacle Detection

The best technique for obstacle detection is by using computer vision. There are multiple algorithms available for obstacle detection in computer vision.

2.13. Computer Vision

2.13.1. Single-Shot Multibox Detector

Single-shot MultiBox detector (SSD) was created to increase the inference time of the algorithm while making it comparably accurate to the R-CNNs. The base network architecture is usually VGG or ResNet. Unlike YOLO, it is a multiscale model where first it extracts a multiscale feature map at different layers of the CNN. Afterwards, it predicts the anchor boxes by defining default boxes on potential object locations. For each prediction, the SSD predicts the bounding box offset required to adjust the location of the box for accurate object detection and class scoring to predict the class of each box. They are faster than R-CNN and Faster R-CNN and more accurate than YOLO. Its limitations surround the complexity of model making it difficult to implement in real time object classification. [11]

2.13.2. You Only Look Once Algorithm

It is a simple model, and it is different from previous models in the respect that it treats object detection as a regression problem instead of re purposing classifiers. Instead of treating the bounding box coordinates and class probabilities as different problems, the model is first per-trained for detection and is modified with extra layers and further trained for classification. It is faster and can be implemented in real time, with compromise in accuracy compared to R-CNN. The network has a fixed-sized image which is divided into grids. Each grid is responsible for predicting the object. It uses convolutional layers to extract features from the image and reduces the spatial dimensions while decreasing the number of channels (depth). It is then fed forward to the fully connected layer which outputs the bounding boxes and the class prediction for an object. This data passes through the Non-Maximum Suppression (NMS) layer. It is a post-processing layer which eliminates the bounding boxes with low-confidence levels. [12][13]

2.14. Object Tracking

For keeping track of the detected object and evaluating the trajectory in temporal dimension [14], SORT and Deep-SORT are the widely used algorithms.

2.14.1. Simple Online and Real time Tracking

Simple Online and Real time Tracking (SORT) is an object tracking algorithm that takes input from an object detection model like YOLO or R-CNN and outputs a unique ID to the bounding boxes in every frame. Each unique ID belongs to a single object being tracked. It uses a Hungarian algorithm to associate detected objects in each frame and Kalman Filter to estimate the state of each object. In [15] sort-based algorithm is discussed for mobile robots.

2.14.2. Deep Sort

Deep Sort is the extended version of Sort that combines the power of Convolutional Neural Network (CNN) to enhance object tracking with varying lighting conditions and large deformations. The CNN extracts the feature vectors which are then compared using embedding distance metrics like Mahalanobis distance. For certain scenarios, it can re-identify an object after occlusion or noise. In [15], there is a detailed explanation about the application of sports in mobile robotics.

2.15. Gait Motion

The robot can move in different combinations of motion of its legs. Selecting the best combination is very important. All types of motions have their advantages. Now there is brief overview of different motion types [16]:

2.15.1. Walk

The easiest form of motion is walking motion in which at any instance 3 toes of a robot are always on the ground. Robot moves in the desired direction by simply stepping one toe at a time in the concerned direction.

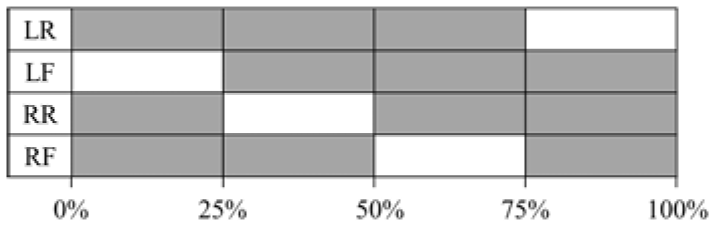


Figure 4-Gait diagram of walk gait [16]

Here the walking pattern of the legs can be observed in Figure 4-Gait diagram of walk gait [16]. Gray represents toes on ground while white represents toes in air. Walking motion is relatively slow but has greater stability due to one leg in air.

3.4.1. Trot

In trot gait the robot takes steps of diagonal legs simultaneously and moves by altering the diagonal pair. So, at any instance 2 legs which are diagonal to each other are on the ground.

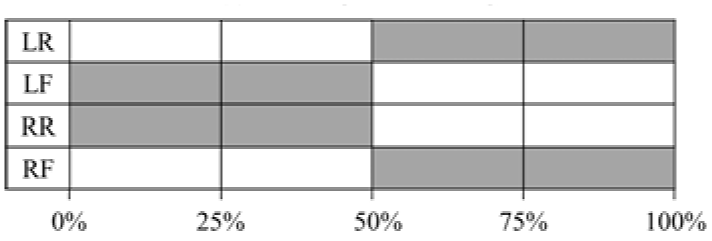


Figure 5-Gait diagram of trot gait [16]

The motion pattern of legs can be seen in the Figure 5-Gait diagram of trot gait [16]. Trot gait is faster than walk gait and consumes less energy due to high efficiency.

2.15.2. Bounce Gait

This is the fastest among all in which the robot moves by jumping. At some instances, there are 2 legs on ground and sometimes there is no leg on ground. Although it is fastest, it is quite unstable and energy inefficient. So, it is relatively avoided.

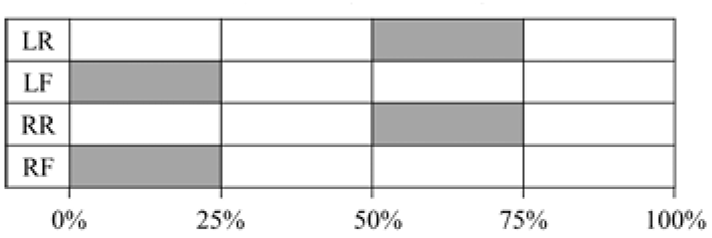


Figure 6-Gait diagram of bounce gait [16]

2.16. Stability

When it comes to stability IMU is the first thing that comes in perspective. For stable posture correction, the IMU values are used to determine the current state of the robot. Now let’s see the technique of using IMU for stability purposes,

2.16.1. Inverse kinematics

One technique is to find roll and pitching angles from the values of IMU using inverse kinematics. By acquiring values from IMU, implement an inverse kinematic equation to evaluate the angles which define the state of the robot. Then implement corrections as required.

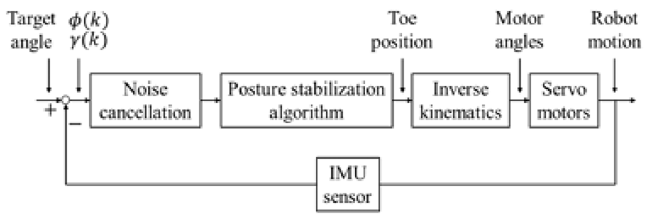


Figure 7 - Algorithm [16]

[16] Mori and Kimihiro proposed a stable trot gait system using IMU in which they discussed how to calculate angles and reduce fluctuation. Then control posture using the outputs. They tested their system physically up to 10° degrees.

2.17. Terrain Adaptation

When moving on different terrains robots adapt locomotion based on terrain. For smooth surfaces, it should be on wheels while when it is rough surfaces or irregular ones it should walk on legs. Terrain adaptation is very important. [17] Medeiros and Vivian S did work on trajectory optimization in difficult terrain.

2.18. Case Studies and Real-World Applications

2.18.1. Spot Mini

Spot mini is a robot that can work in indoor as well as outdoor environments. It was designed by Boston dynamics. It is lightweight, fully electric and can work for up to 90 minutes on one time charging. It has force sensors, stereo, depth cameras, and IMU sensors. It can carry a payload of 14 kg.it is remote controlled but can perform some of its tasks autonomously. It can have its application in industrial inspection, for industrial surveillance and for package delivery. The spot mini is one of the best due to its low weight and cutting-edge capabilities.

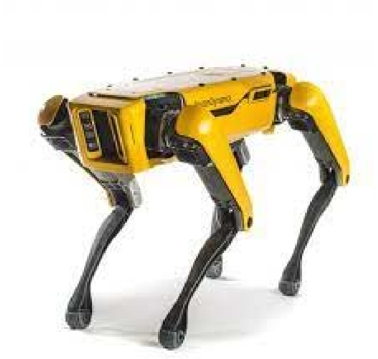


Figure 8 - Spot Mini [12]

2.18.2. ANYmal

ANYmal is a four-legged robot manufactured for working in challenging and tough environments. It was designed by ANYbotics, a private company in Switzerland. Its specialty is in working in industrial inspection. It is equipped with an IMU sensor, stereo cameras, LIDAR and temperature sensors, ultrasonic microphone, and gas sensors.it can work for 2 to 4 hours battery time which is feasible for any kind of inspection and to work in any kind of unknown environment. It can carry a payload of 15 kg. It can work in inspection of the oil and gas sector, industrial pipelines, and gas sensing.

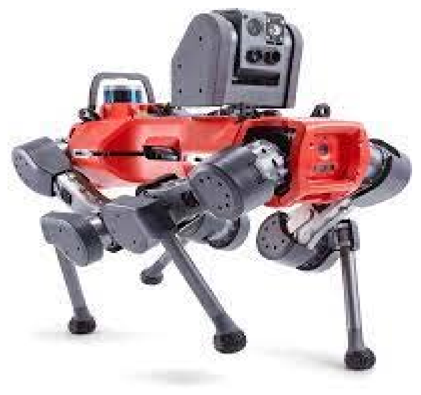


Figure 9-ANYmal [18]

2.18.3. HYQ

It is a robot which is designed for working in rough terrain. It has multiple functionalities in its working and can even kick things. It was designed and manufactured by Istituto Italiano di Tecnologia. It is equipped with stereo cameras, laser range finder, IMU and force torque sensors. It has a weight of up to 80 kg. It can work for search and rescue operations and can work in contaminated and harsh areas. It can also work in inspection and exploration tasks.

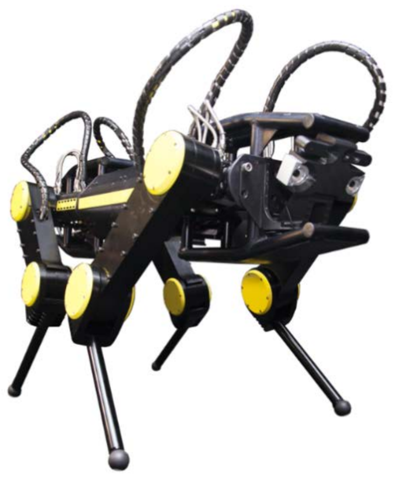


Figure 10 – HYQ [19]

2.18.4. Mini Cheetah

It is one of the robots which can do backflips. Designed by the MIT Biomimetric lab, its weight is just 80 pounds.it is equipped with IMU sensor and hall effect encoders at each motor. By using a machine learning approach, it can run at a top speed of 8.7 MPH.

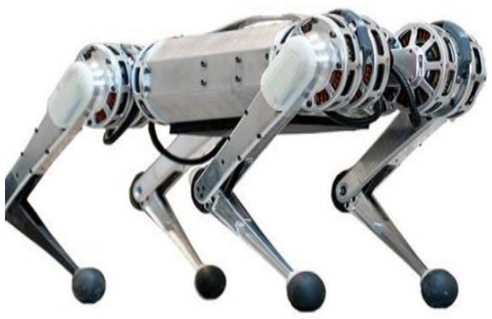


Figure 11- Mini Cheetah [20]

2.18.5. W1 Wheeled Quadruped Robot

It was designed by LimX Dynamics, a Chinese company, the robot can work autonomously with perception and motion algorithms and different kinds of sensors. Having both wheels and legs give it a unique edge over other competitors in the market.it can work in different range of environments using its dual mobility feature in different terrains. It has applications in industrial inspection, education, logistics, and distribution.

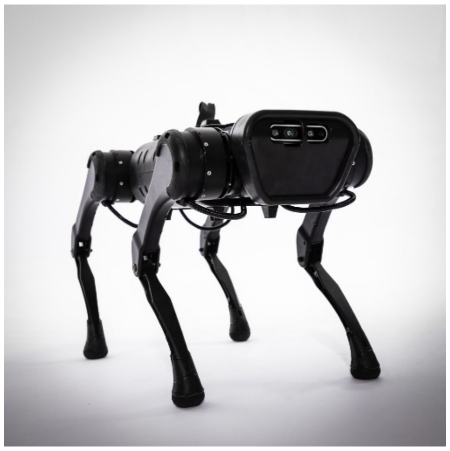


Figure 12 - W1 Wheeled Quadruped robot [21]

2.18.6. Four-legged/Wheeled Robot

This robot was designed under the National Centre for Robotics and Automation (NCRA) Pakistan. It is a robot that has both legs and wheels for locomotion. It was 3D printed part by part and then was joined together manually. Currently, it is working by using a remote controller and an ESP 8266 module. It can switch between legs easily. It can find its applications in industrial applications, exploring unknown locations and rescuing in dangerous places [1].

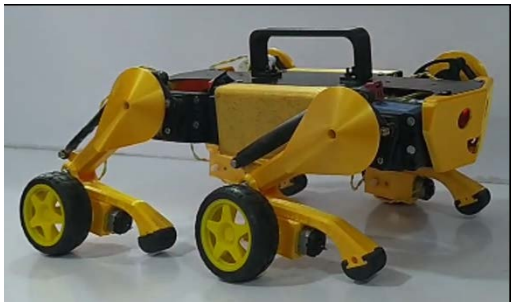


Figure 13-Four legged/Wheeled Robot [1]

Chapter 3 – HARDWARE

3.1. Body

The body of the robot consists of four legs connected to a rectangular torso. Its design resembles a robot dog. Each leg has an arm and a wrist.

3.2. Links And Joints

Each of the legs of the robot has two links, one full joint and one-half joint. The full joint is between the two links of the arm and the wrist. The half joint attaches the arm of the robot to its body. The half joint provides two degrees of freedom. And one degree of freedom is due to the full joint, so it adds up to 3 degrees of freedom in one leg. Joints are made such that there is no play in them and only moves the designated direction.

3.3. Tires

Tires are attached to each of the elbows of the robot which are driven by DC motors. It adds up the feature to convert the legged robot into a four-wheel drive. DC motor is not attached directly hence to enhance torque we have used gear set after motor and between wheels. The gear set increases torque of wheel at the expense of speed.

3.4. Manufacturing

The body and legs are 3D printed. Two metal sheets are attached, one below the base of the rectangular body and the other at the top of it. The sheets are screwed into the body and provide structural strength to the torso. The torso is empty from the inside and can be accessed after removing the upper sheet. This space provides accommodation for the electronics and the controller. Tires are procured from outside. The foot of the robot has rubber friction pads.

3.5. Sensors

3.5.1. Light Detection and Ranging

LIDAR stands for light detection and ranging. It is an essential component of any autonomous vehicle or robot. It is used for obstacle avoidance, localization, mapping implementation, and environment perception.

3.5.1.1. Working

- The lidar emits laser pulses in various directions with the help of a mechanism. The pulses are emitted in short bursts.
- When a beam collides with an object, it reflects toward the LIDAR sensor.

- The LIDAR sensor receiver detects the pulses. By measuring the time of flight, the LIDAR calculates the distance from the object to LIDAR.
- As LIDAR continuously sends and receives the laser pulses, it develops a dense cloud representation of the surroundings which can be used for multiple applications.

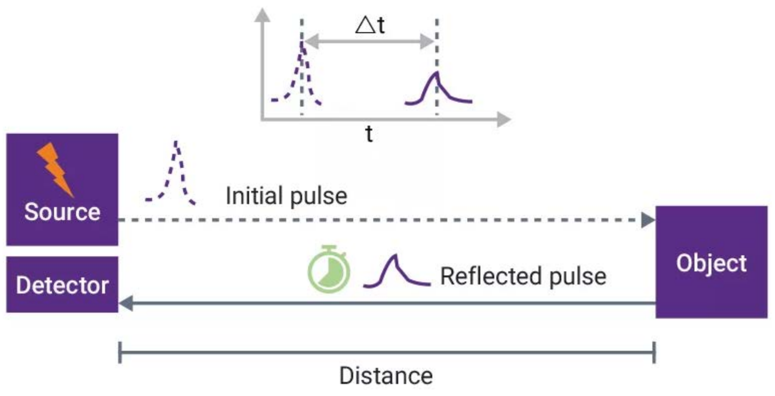


Figure 14- Working of LiDAR (https://www.synopsys.com/glossary/what-is-lidar.html)

3.5.1.2. Applications

It can be used in the following applications:

- Obstacle detection and avoidance for autonomous vehicles.
- It can be used for navigation and path planning.
- It can be used for dynamic environment monitoring.
- It can be used for localization and mapping.
- It can be used for semantic understanding of the environment.
- It can be fused with cameras and other sensors.
- It can be used in driver assistance systems in autonomous vehicles.

3.5.2. TX-20 LIDAR

The TX-20 is a LIDAR manufactured by YDLIDAR, a company that has a specialty in LIDAR manufacturing. It is a high-performance LIDAR which scans 360 degrees. It is a 2D LIDAR that can be used indoors as well as outdoors. It is equipped with all the technologies, optics, and algorithms to achieve high frequency and high precision distance measurements. It rotates 360 degrees to continuously give the angle as well as the point cloud data of the surrounding environment while working. Optical and laser lenses are used for sending and receiving laser pulses to achieve the task.

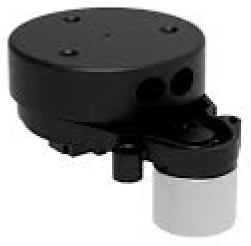


Figure 15 - TX-20 LiDAR

3.5.1.3. Features

- 360 degrees LIDAR
- The ranging distance is 20m.
- Strong resistance to light interference.
- Low power consumption and long-life span.
- Motor speed customization is available.

3.5.1.4. Key Specifications

The following are the key specifications which are in TX-20 LIDAR:

Table 1 - TX-20 Specifications

Laser wavelength	895-915nm
Ranging frequency	4000/secs
Scanning angle	0-360 degree
Baud rate	115200
Motor frequency	7hz(optimum)
Ranging accuracy	+/-4cm
Ranging distance	0.1m-20m
Angle resolution	0.63 degree

3.5.1.5. Electrical Parameters

The following are the electrical parameters which must be followed for optimal working:

Table 2 – TX-20 Electrical Parameters

Supply voltage	4.8-5.2V
Starting current	200-400mA
Working current	200-380Ma
Voltage ripple	0-100mV

3.5.1.6.Pin Interface

The following is the pin interface for the TX-20:

Table 3 - TX-20 Pin Interface

pin	Type	Description	Default Ts	Range	Remarks
VCC	Power supply	Positive	5Volts	4.8-5.2 (Volts)	
TX	Output	System serial output	-	-	Connection between LIDAR and processor
GND	Power supply	Negative	0Volts	0V	

3.5.1.7.Drivers

TX-20 can be easily integrated with ROS and Linux. It can be coupled with SLAM algorithm and other sensors to achieve the arduous tasks of navigation and mapping. The company has provided Linux and ROS drivers to use for applications. There is a complete TX-20 package to get it working with ROS and Linux. There are drivers called Software development kits (SDKs) for their LIDARs to fuse with different systems. The drivers include libraries and APIs to control the lidar for different tasks.

3.5.3. Inertial Measurement Unit

An IMU is a sensor that can measure linear acceleration, angular velocity, and sometimes orientation. It has the following components:

3.5.1.8.Accelerometer

It measures acceleration in 3 axes. The working principle behind working is inertial mass. When robot moves, the mass inside in accelerometer experiences a force which is then converted into an electric signal proportional to acceleration. By measuring acceleration in each axis, the accelerometer measures linear acceleration.

3.5.1.9.Gyroscope

It measures the angular velocity in each of the 3 axes. It's working is based upon the principle of angular momentum. The spinning mass inside the gyroscope experiences a deflection when IMU moves which is used to measure angular velocity.

3.5.1.10. Magnetometer

It measures the strength and direction of the magnetic field around IMU. The data is then used to orient the robot concerning Earth's magnetic field. There can be other components in IMU which can vary from module to module and depending upon the requirement of the application

3.5.1.11. Applications

The following are the applications of the IMU:

- It can integrate with LIDAR and a camera to achieve localization.
- It can be used for navigation of the robot.
- It can be used for motion control.
- It plays a fundamental role in the balancing of drones.

3.5.1.12. GY-86 IMU

It is a 10 DOF IMU module that combines 4 sensor modules in a single package. it consists of the following sensor packages:

- 3-axis accelerometer
- 3-axis gyroscope
- 3-axis magnetometer
- 1-axis barometer

All these sensors work together to achieve the desired outcomes. In this application, only accelerometers, gyroscopes, and magnetometers are used. There is no need for a barometer in this application.

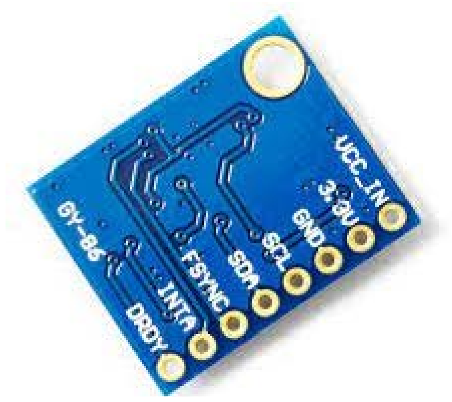


Figure 16 - GY-86 IMU Module

3.5.1.13. Technical specifications

The following are the technical specifications of the GY-86 IMU:

- I2C interface for communication
- Voltage required 3-5V.
- Dimensions 21.5mm x 16.8mm x 2mm
- Weight 30g

3.5.4. Stereo Camera Module

A stereo camera module is a combination of two cameras aligned horizontally at a fixed distance in a frame. The displacement between the two cameras is usually known but can also be calculated as an extrinsic property through compare-match algorithms like checkerboard rectification. Stereo cameras are used for depth estimation. It works on the principle of disparity. Disparity is the change in location of the object from the perspective of two different cameras. Two cameras can provide different fields of view and the objects can have different perspectives from each camera. This change can be used to estimate the distance of the objects. It can also be compared by using human eyes. Two horizontally aligned cameras with the same vertical position can have disparity only in the horizontal axis. The disparity is inversely proportional to the distance of the object from the center of the two cameras. So, a linear interpolation can be used between the inverse of disparity and the distance of the object.

3.5.1.14. Camera Models

A camera can have two different types of models: pinhole and fisheye. The most widely used camera model is a pinhole. The fisheye model has been used in this project. OpenCV provides the implementation for both types of models.

3.5.1.15. Fisheye Lens

A fisheye stereo camera was acquired. It provides two uncalibrated camera streams through a USB interface. The device is compatible with universal Windows and Ubuntu drivers. The camera stream can be captured through OpenCV. The two camera streams are combined in a single-channel horizontally stitched matrix. OpenCV stores this data in a NumPy Nd-array. The streams can be split and further processed.

3.5.1.16. Casing

A camera module casing was designed and then fabricated by using 3D printing. The case is fully enclosed and has two holes for the camera lens and one for the wire connection.

3.6. Actuators

3.6.1. Servos

For the locomotion of the four-legged robot, selecting the good actuators is critical to get smooth motion. It depends on different factors like the robot's weight and the size of the four-legged robot. The weight of the robot depends upon all the components of the robot and the physical model of the robot. There can be the selection of hydraulic and

Figure 17 - Servo Module

electronic actuators. Electronic actuators are easy to control as compared to other actuators. To make the robot autonomous and control it with the help of a processor, the SPT5430 servomotor was selected as the leg actuator. The motors can be given specific angles of moments with any microcontroller. So, it was easy to control as compared to other DC motors. They were selected for all the legs.

3.6.2. Motors

Since the four-legged platform is also equipped with tyers, it is necessary to add DC motors because they are easy to control and accessible. The configurations of torque and mechanical power can easily be changed by adjusting the speed of the DC motor.

3.5.1.17. Mathematical Representation

For calculating the motor's torque, the following equation can be used:

$$T = \frac{1}{Nw} * \frac{Dw}{2} * FT$$

Table 4 - Motor Specifications

DC Voltage	3V	5V	6V
Current(mA)	100	110	120
RPM (with wheel)	100	190	240
Reduction ratio	-	48:1	-
Noise	-	<65db	-
Weight	-	29g	-

3.7. Processor and Controller

Jetson Nano has been used for processing the ROS framework and FPGA board as the base controller of the robot.

3.7.1. Jetson Nano

Jetson Nano is a micro-computer with a Cortex-A57 ARM CPU and 128-core Nvidia Maxwell GPU. It has 2GB and 4GB variants. 4GB variant of Jetson Nano is equipped in a four-legged robot. To flash the firmware in the SD card, Jetson SDK is used. The firmware is directly bootable from the SD card. Jetson Nano has a modified Ubuntu running and comes with pre-installed libraries to utilize CUDA cores for GPU processing.

Figure 18 - Jetson Nano

3.7.2. Digilent Cmod A7

Artix 7 series CMOD A7 15T board with FPGA package of XC7A15T-1CPG236C is used in this project. It is a 48-pin DIP board built on a Xilinx Artix-7 FPGA. The board includes a USB-JTAG circuit for programming, 12 MHz oscillator as the clock source, a USB-UART bridge, SRAM as volatile memory, a Pmod connector, Quad-SPI Flash for non-volatile memory, and basic I/O devices. These components make it a compact easy-to-use platform for digital logic circuits and embedded soft-core processor designs. There are 44 digital GPI/Os and two analog inputs. The board is compact as dimensions are 0.7" by 2.75".

Field Programmable Gate Arrays (FPGAs) are semiconductor integrated circuits with configurable logic blocks. These configurable logic blocks are interconnected via programable interconnects and due to this feature, they can be used for a diverse range of applications and functionality. FPGAs are generally used in applications involving precise controls with swift response, and speed flexibility. They have the power of parallel processing.

FPGAs are configured using hardware descriptive language. Hardware descriptive language can textually describe the circuit logic. Verilog is generally used for this purpose. There are different levels of programming which are switch level, gate level, dataflow level, behavioral level. Levels are basically based on the logic used. Switch and gate levels are basic, and they are like bare metal programming. Dataflow and behavioral levels are quite advanced and have functions, loops, and if – else statements.

FPGAs require some other components for proper utilization like oscillators for clock, SRAM or flash memory, some sort of serial programming circuitry, and USB-UART bridge and other components. Alone FPGAs can't be useful, so these essential components are generally packaged on one board with FPGA for proper functioning. There are a variety of boards available, some have greater GPIOs and some focus on interface and large memory options.

FPGAs implement their logic using look-up tables. As every digital logic is based on some gate patterns and multiplexers, all gates logic could easily be expressed in form of tables. So, at hardware level circuit logic is stored in tables and based on states of inputs output is generated. This is the basic functioning of FPGA.

3.8. Electronics

3.8.1. Motor Driver

It is an integrated circuit manufactured to control DC motors. The integrated circuit consists of two H-bridge circuits. An H-bridge is a circuit that allows two-direction control of a motor with the help of current flow direction. The integrated circuit allows the control of the speed and direction of DC motors by providing the signal from a computer or any microcontroller.

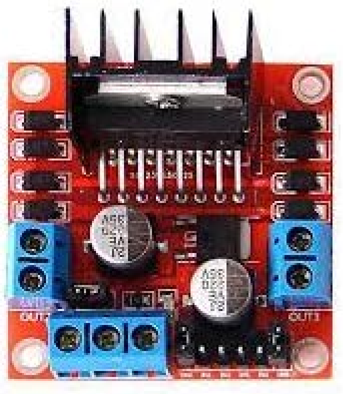


Figure 19 - L298N Motor Driver

3.5.1.18. Features and specifications:

The following image shows the specifications in the L298N integrated circuit.

Table 5 - L298N Specifications

Driver Model	L298N 2A
Driver Chip	Double H Bridge L298N
Motor Supply Voltage (Maximum)	46V
Motor Supply Current (Maximum)	2A
Logic Voltage	5V
Driver Voltage	5-35V
Driver Current	2A
Logical Current	0-36mA
Maximum Power (W)	25W

3.5.1.19. Usage In Project

In this project, it is used to control the dc motors of the tires. As robots can operate in both legged as well as wheeled modes, for the use of wheeled mode, there is a requirement of this module so that a signal can be given for movement on wheeled mode.

3.8.2. Power System

For power delivery, one battery of 3,000 mAh and another of 4,000 mAh is used. One battery is used to drive the actuators and the other one is used to power the processors, controllers and sensors. The segregation of batteries is due to the ripples created by actuators in voltages which can affect the processor functioning.

3.8.3. Buck Module

A buck converter is an electronic circuit that converts a high voltage into a low voltage. It is used in electronics to maintain a constant supply to the other devices so that they can work at their optimum level. Buck converter works on the switching principle resulting in less heating effects and energy saving. Different sensors and processing modules are in circuitry of this robot that require 5V. The following are:

- Jetson nano
- Lidar
- FPGA (field programmable gate arrays)

As the battery is giving a supply of 12V, there is a need to convert that 12V into 5V so that sensors and processors can work at optimum level without any risk of failure. For this 9A step-down buck converter is used. It can convert input voltage in the range of 7-40 volts to 1 and 35 volts. It has a current output of 9A.

Figure 20 - Buck Module

3.8.4. Relay Module

The Relay Module is used as a safety switch. It is connected in series to the ground wire of the power circuit of actuators. It is in a normally open state and controlled by FPGA. It ensures that power is delivered to the actuators only when the FPGA board is powered up.

3.8.5. FPGA Shield:

A shield is designed on a PCB board for FPGA. It provides a bed for FPGA and headers to attach wires.

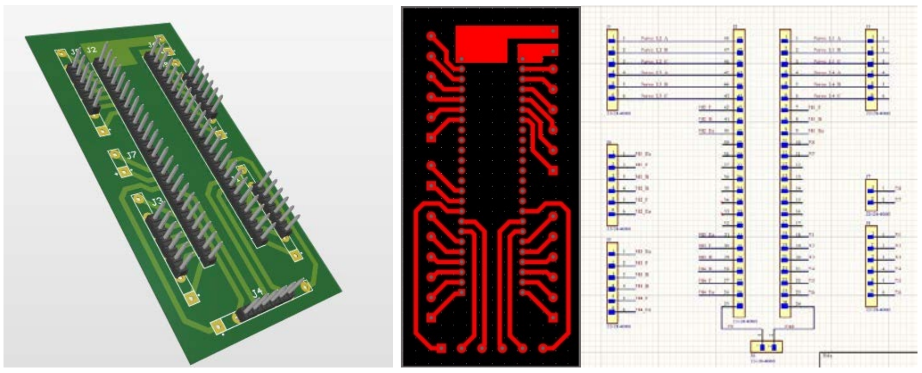


Figure 21 - FPGA Shield:- Left: 3D model, Middle: PCB Layout, Right: Schematic

Chapter 4 – METHODOLOGY

The scope of this robot is to perform industrial inspections in mapped and unmapped environments. The robot is highly modular by design. Various sensors and actuators. be set up with the robot to do a variety of inspection tasks. To achieve autonomous navigation, the industry-standard approach has been followed. Starting with area mapping and localization, followed by path planning of the robot. These tasks will be achieved using the Robot Operating System (ROS) framework on Jetson Nano with an FPGA board as a robot actuation controller. The robot is equipped with vision, inertial measurements, and ranging sensors. These sensors acquire data from their surroundings to facilitate mapping and localization. The methodology can be visualized through the Figure 22 – Methodology Flowchart.

4.1. Mapping, Localization, and Path Planning

To setup a navigation system, we require the following:

- A global map of the environment
- Position estimate of the robot
- Destination or Goal point
- Path Planning Algorithm
- Obstacle Avoidance System
- Navigation Controller

4.2. Navigation

Navigation can be achieved by the following two methods which are differentiated based on the approach of acquiring a map of the environment.

- Navigation using a prebuilt map image.
- Simultaneous mapping and localization.

4.3. Realtime Mapping

With the help of Lidar, and IMU, the Hector Slam can generate a map of the indoor environment while simultaneously localizing itself in the environment. LiDAR can provide an accurate 2D map of obstacles. This data is processed by Hector SLAM to generate a static map of the environment. The roll and pitch introduced by the robot motion can cause inaccuracy in map. This can be dealt with by introducing IMU. IMU can track the change in orientation to create an accurate 2D slicing of environment.

LiDAR is used primarily to locate the position and orientation of the robot and localize the robot with respect to a map of 2D static environment. IMU can provide 3D orientation information. Combining it with LiDAR, Hector SLAM can acquire an accurate 3D attitude of the robot. It will know its location in the robot so that it can move toward the required destination. The map generated can be saved for later use by saving it through Hector mapping server.

4.4. Prebuilt Map

The robot can load an image of a static map from its collected database. The robot will first localize itself with the help of the Adaptive Monte Carlo localization method. In this case the robot already has a map and AMCL is only providing localization. The robot will take readings from sensors and then based on these readings the algorithm will match the new reading with older readings of the prebuilt map. The robot will then make predictions of the reading match. Then readings are resampled, and the robot's current pose is estimated. This process is repeated until there is an accurate position of robot in the map.

For localization in the loaded map, random initial coordinates are assigned and by calculating the probabilities from Monti-Carlo localization, it approximates its position and orientation in the environment. Visualizing tools like Rviz can assist the localization process by providing a guess of initial position.

4.5. Path Planning

For path planning, the ROS navigation stack is used. It uses global and local planners which utilize local and global costmap to generate an optimum path from initial to final point.

4.6. Global Planner

The global planner is used to create a long-term path plan from the initial point to the destination. It requires two coordinates: initial coordinates and destination coordinates. The initial coordinates are provided by the localization algorithms. The destination coordinates can be given through ROS topic which can be integrated with visualization tools like Rviz to provide visual point selection from the map. The global planar takes in the static map and generates a global cost map of the map. The global cost map is a grid based potential field gradient cost map.

4.7. Local planner

The local planner is predominantly used for avoiding obstacles and has a small range in the near areas of the robot. With the help of LiDAR, the local planner marks obstacles in its local costmap. This local costmap is used to add modifications or regeneration of planned path to avoid obstacles while following the global path. The orientation of the robot also plays a vital role in deciding how much of an angular deviation is required to avoid the obstacle while remaining close to the planned path. The robot plans its path to avoid obstacles and be at a minimum distance from the global path.

4.8. Command and Interface

After the path is planned, high level robot controller is designed to provide feedback to the robot for its actuation. This controller is a ROS node that takes the current position and planned path and outputs movement commands. These commands are provided to the low-level controller which is FPGA. FPGA takes the command like move straight or make a turn and generates timing diagram for actuation of servo to achieve a desired gait pattern.

The interface of the robot is implemented with Rviz, a visualization tool provided with ROS. The custom Rviz layout file is loaded with the navigation environment. This tool can be used to visualize the map, robot attitude, optimal path, and obstacle cloud. It also provides an easy-to-use interface to provide robot destinations for navigation in the map.

Except for the robot controller, all the above steps are implemented on Jetson Nano. The software solution is developed in the ROS framework. ROS provides a modular approach for inter-communication of programs and due to its open-source nature, many industrial companies provide support and contribute to this program keeping it updated to the industry standards.

4.9. FPGA As Robot Controller

We are harnessing the power of FPGA to design a robust robot control that provides synchronous control of the robot actuators. The robot has twelve joints, each controlled by a servo motor. The joints in each leg can provide a 3 Degree of Freedom. All four legs must move synchronously according to a provided gait pattern. In this respect, a controller with parallel processing power is recommended. FPGAs are inherently customizable hardware that can be programmed to provide independent parallel processing to each leg of the robot. The clock timings are synchronized through a base clock of the FPGA. This ensures that there is no lead or lag in the system.

The FPGA is interfaced with Jetson Nano with GPIO in parallel connection. The idea behind this approach is that we minimize any delay in signal transmission. The data is binary coded parallel connection. For example, a signal of 00 on two wires indicates standing mode while a 01 indicates moving straight.

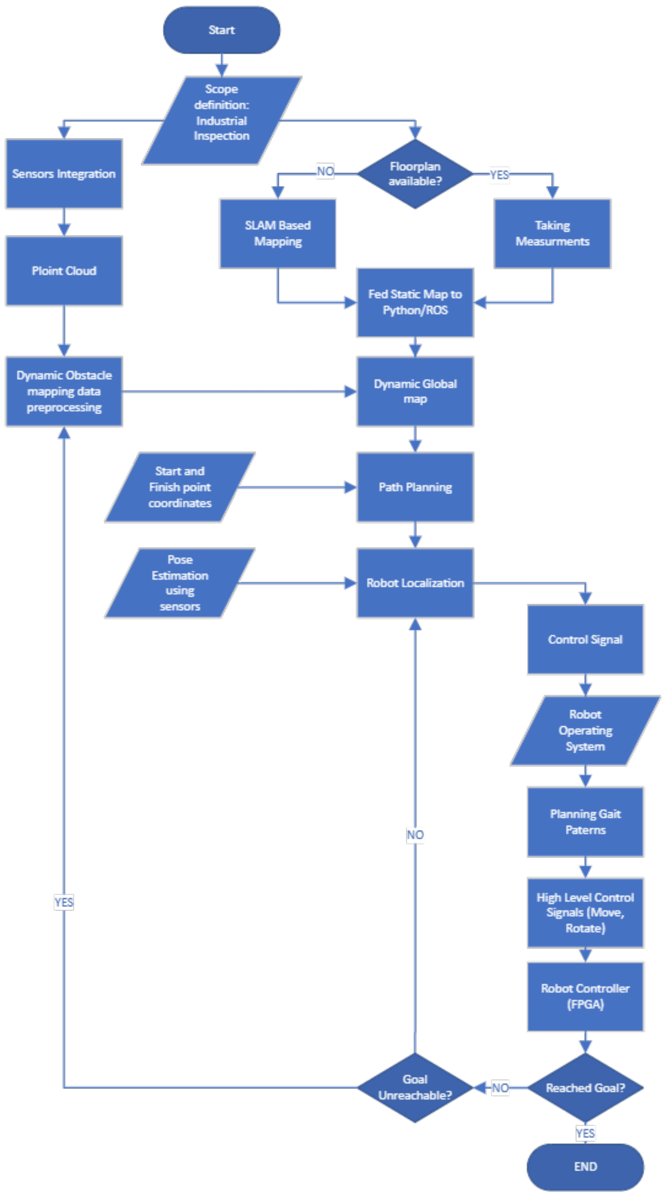


Figure 22 – Methodology Flowchart

Chapter 5 – AREA MAPPING AND LOCALIZATION

Area Mapping is the process to generate a map of an unknown environment using laser or vision sensors. The most popular method is laser scanning while modern solutions are also adopting a vision-based approach. Both approaches deal with the generation of point clouds of the objects and barriers in their surroundings. LiDAR is widely used for laser scanning while both monocular and stereo cameras can be used in vision-based point cloud generation.

5.1. Point Cloud

A mapping sensor generates a point cloud in either 2D or 3D environment. For real-time implementation, 2D point clouds are preferred as they are less resource intensive. In a 2D environment, the X-Y plane is divided into cells of occupancy grids. Each grid can hold three values:

- Occupied
- Free
- Unknown

5.2. Grid Mapping

Each point in the point cloud represents an obstacle, and the line connecting the obstacle to the scanner in the occupancy grid is free space. The area beyond the point cloud is unreachable. The occupancy grid is initialized with an unknown state. Mapping algorithms iterate over several scans of sensor to declare the cells either occupied or free. The unreachable cells remain in an unknown state.

5.3. Localization

Localization is a process in which a mobile robot estimates its orientation and position in an environment by either examining its surroundings or measuring its transformation from previous states. An internal state-measuring sensor Inertial Measurement Unit (IMU) can be used to calculate the transformation of states while an external state-measuring sensor can be used to estimate the state of the robot in a known environment.

5.4. Internal State Estimation

The change in position or orientation of a robot can be seen as an internal state estimation problem when the initial position of the robot as the fixed frame of reference is used. An IMU can easily determine the change in state.

5.4.1. Inertial Measurement Unit

An IMU is a collection of several sensors including a gyroscope and accelerometer as mandatory and magnetometer and barometer as optional sensors. The accelerometer measures changes in linear acceleration and the gyroscope provides change in angular velocity. The fact can also be considered that gravity is the form of a constant linear acceleration perpendicular to the surface of the Earth. This information can be used to estimate the pitch and roll of the robot. The limitation occurs in high-vibration systems where the accelerometer fails to provide accurate estimate of gravity. Gyroscopes can be utilized to cover this deficiency as it can measure high frequency changes accurately. But it can introduce drift when used for estimating the orientation. A magnetometer can determine the direction of magnetic field of North Pole and thus can be utilized to estimate the yaw of the robot. A barometer can report the change in pressure and can be utilized to estimate the height of the robot as the decreaseing pressure is proportional to the altitude.

5.4.2. Complementary Filter

A complimentary filter can be utilized to combine the readings from different sensors in an IMU to give the best approximation of the attitude of the robot. It covers the shortcomings of the accelerometer by applying a high pass filter and can overcome the drift of the gyroscope by applying a low pass filter. This can give a highly accurate pitch and roll attitude. A magnetometer may be fused with a high pass filter to provide the yaw attitude.

5.5. External State Estimation

The external state of a robot is the measure of its position and orientation with the frame of reference of its static surroundings. This static surrounding can be known or unknown. A known static surrounding can be in the form of a map of static obstacles and barriers, a coordinate of Global Positioning System (GPS) or the strength of electromagnetic radiation in a near-field estimation system. From a commercial industrial inspection point of view, mapped based approach is the most suitable. GPS based systems fail to work in an indoor environment and near-field devices are complicated to setup, require the installation of a near-field antenna network, and are prone to electromagnetic disturbance. For a commercially viable, high precision and non-invasive solution, map-based state estimation is opted.

5.5.1. Map Based Estimation

A map-based attitude estimation system usually acquires the map from an area mapping algorithm or saved map and compares its surroundings with the map to estimate the attitude and position of robot in the map. This can be achieved by several methods. Scan matching is one example where a map is generated and compared with the original map to estimate the correct position while another approach is to use Montecarlo probability distribution for estimation.

5.6. Simultaneous Localization and Mapping

These are the approaches that simultaneously solve the localization and mapping problem. It generates a continuous map of the environment while estimating the attitude of the robot. Several SLAM approaches are available that utilize different sensors to achieve this task. IMU and LiDAR are used as primary sensors for 2D SLAM. While stereo cameras can be used to achieve 3D SLAM due to high computational resources and limited return of investment for a UGV application, 2D SLAM is used. A 3D SLAM is better suited for aerial vehicles and mobile manipulators.

5.7. Hector SLAM

Hector SLAM is one the most popular implementations of SLAM. It implements scan matching algorithm with muti-resolution occupancy grid to generate map as well to localize in X-Y utilizing laser scan. The multi-resolution map deals with the issue of interpolating discrete map data. The laser scan of LiDAR is combined with the pose estimate of IMU for 3D localization of the robot. It has several advantages as below.

- It is a fast SLAM algorithm that requires low resources for real-time usage.
- It combines 2D LiDAR data with 3D attitude estimation from IMU to compensate for roll and pitch of base.
- Does not require loop closure.
- Utilizes the advantage of high-frequency modern LiDAR.
- Detailed documented and open source as ROS package

Chapter 6 – ROBOT OPERATING SYSTEM

ROS stands for robot operating system. It is an open-source platform for robot software development. It is a package consisting of tools that can be used to design software for any kind of robot.

6.1. Need of Robot Operating System

Robot Operating System is the industrial standard approach to design robot firmware and supporting software. It adds modularity and encapsulation of robot programs. Its aim is to achieve a standardized approach for inter-program communication protocols. ROS has a strong community of developers and industry leads. By utilizing ROS, the robot can tap into existing markets as well as gain attention in the ROS community. It can also leverage from open-source packages and libraries that are used in industry.

6.2. Big Picture of ROS

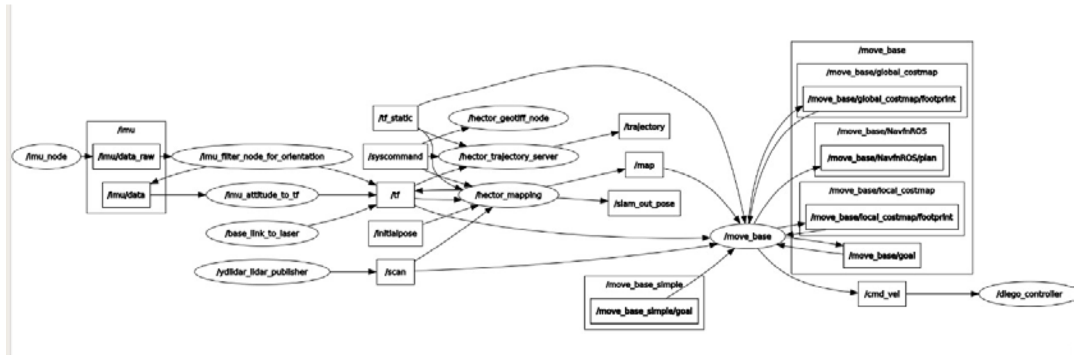


Figure 23 - RQT Graph

Figure 23 is called RQT Graph in ROS. It is a tool to visualize the nodes, topics, and the data flow. The oval shape of components are nodes, and the rectangular components are topics. All nodes are either publishing or subscribed to different topics. Nodes intercommunicate through common topics.

6.3. Interconnections

The data flow generally starts from the left to right, but some relations are complex. The dataflow starts from sensor nodes that are taking values from external environment. The data is acquired through imu and Lidar node. The imu data contains angular velocity and linear acceleration. It is fed to the complementary filter node to compute the orientation of the robot. The Lidar node is connected to Hector Slam and Navigation stack through scan topic. This topic can be visualized in Rviz. The laser scans are shown as red points in Figure 24 - LiDAR Laser Scan.

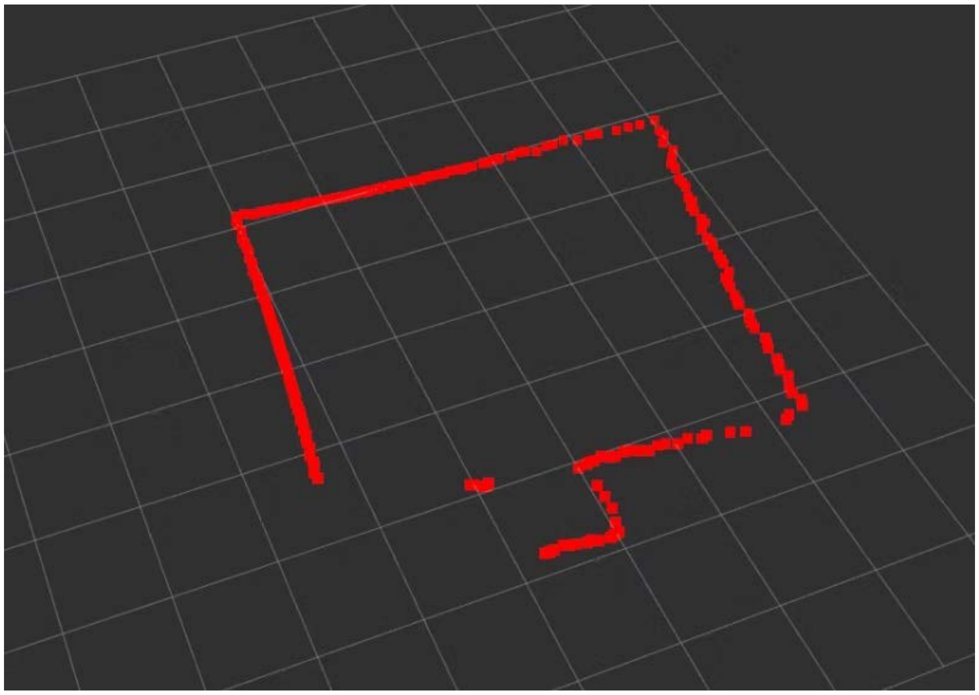


Figure 24 - LiDAR Laser Scan

The Hector Slam has three nodes running. The main node is the hector mapping node responsible for SLAM implementation. The supporting nodes provide trajectory output and image saving features. The hector mapping generates a static map of the environment and provides the location and orientation estimate of the robot. This map is published to map topic which can be visualized in Rviz.

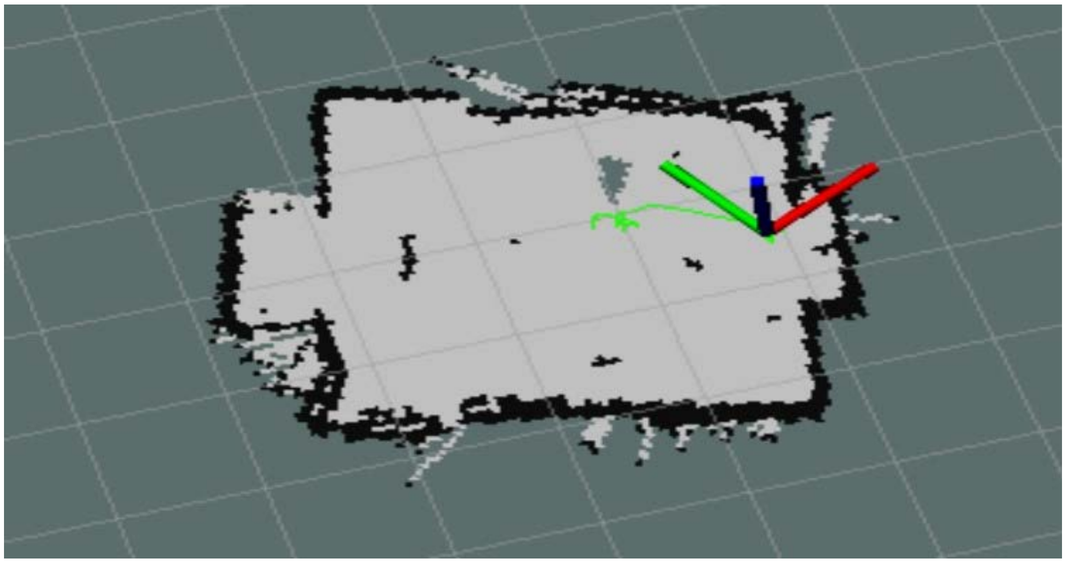


Figure 25 - Map generated by Hector SLAM

The navigation stack has its node named move base which is responsible for planning the optimal path for the robot. It subscribes to the map and scan topic and generates local and global cost map. It takes destination coordinates from the Rviz move base topic and plans an optimal path.

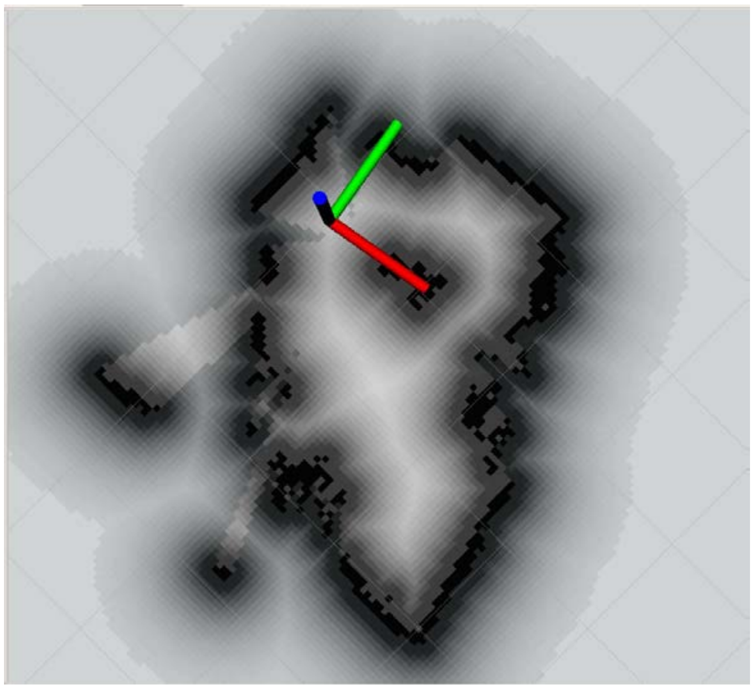


Figure 26 - Costmap generated by Navigation Stack

The planned path is published to command velocity topic which is subscribed by controller node encoding motion command into binary code that is transmitted to GPIO pins.

Chapter 7 – GAIT MOTION CONTROL

Motion of the robot involves walking and a wheeled mode. The walking motion of the robot is achieved using 4 legs. Each leg has 3 degrees of freedom. This is accomplished using 3 servos on each leg for each degree of freedom. So, summing up for 4 legs there are 12 servos for 12 degrees of freedom. Servo motion requires precise control, as it requires a pulse width modulation (PWM) signal of 50Hz to communicate. Hence at each instance of time in achieving a step, there is the requirement of different widths of PWM wave. This is why the usage of FPGA for motion control is required. FPGA has the power of parallel processing. It generates 12 PWM waves of 50Hz simultaneously and most importantly they are independent of each other. And there is the luxury of changing PWM width according to requirement at any instance of time for achieving a step. Secondly in FPGA's there can have a track of pins for up to clock cycles which gives precision, accuracy, and robustness. Comparatively in other processors or controllers PWM signals are generated using timers which inherently indicates the problem that timers are limited so there is an upper limit. And other methods are also available like PWM generation module is used and in which data is sent serially for each channel. This is a compromise on time as data for each PWM is transferred serially. In this method there is no changing two PWM signals at same instance of time. So, using an FPGA was one step forward in advance controls for achieving a smooth motion, with robust control and having a luxury of actuating two or more servos at same time.

Second is a wheeled mode in which robot moves swiftly on a smooth surface using 4 wheels. This mode is used in an environment with smooth surface and requires rapid transit. 4 dc motors are driven using 2 L298N motor drivers. Dc motors are quite easy to operate as there is a need for only one PWM signal of 1.5kHz for speed control and dc signals for switching direction. To make it swift and synchronous, FPGA implementation was required.

7.1. Gait Motion

Different types of gaits were discussed in 2.15 . In walk gait, there is always a need of at least three toes touching the surface of ground. So, motion is achieved by putting one toe forward followed by next and then similarly all four legs move after one another. It is the most stable form of motion but have a shortcoming of time. Hence it takes longer time which increases the operational time as well resulting in poor energy efficiency in respect to distance travelled. So due to its shortcomings, this technique was not implemented.

Trot gait is the optimum technique for most cases as it keeps a balance between the travel time and stability. In this technique, at least two diagonal toes touch the surface making it less stable than walk gait, but body is still quite stable. This has significantly reduced the time by half resulting in making it power efficient in terms of distance travelled. So, this technique is implemented in legged motion as it maintains a balance in velocity, stability, and energy efficiency.

Bounce gait is the fastest type of motion in which a robot jumps to move. As in this type of motion, at least two toes, either front two or back two are touching the ground surface. This is very unstable and power efficiency is also quite low making it less favorable. It was not used due to its shortcomings.

7.2. FPGA Implementation

FPGAs are semiconductor integrated circuits which have configurable logic blocks. FPGA is responsible for the implementation of motion. The programming of FPGA is done in Verilog, a hardware descriptive language used to describe electronic circuits and systems. The programming for this was performed in Xilinx Vivado Design Suite 2019 software. As already discussed, Artix-7 series CMOD A7 15T FPGA board is used in this project. This is a compact board and in the next section logic implementation method is given.

7.3. Programming

The programming is in Verilog. The programming is done on behavioral level as it has advanced functionalities of loops and if statements etc. For base clock in program, input board clock of 12 MHz is available. The program is quite complex, comprising 72 modules only for legged mode and 13 modules for wheeled mode. The main task of PWM is achieved by a module which takes servo angle and clock as an input and generates 50 Hz PWM wave. The servo angles are assigned to a variable by another module independently which is based on timing map of that specific servo. That angle stored in a variable taken as input of first module makes PWM wave. So, this is how a timing map is implemented in Verilog. A timing map for forward motion is given below.

7.4. Flashing The Program

The board has 512 kb SRAM which is easy to use but issue with that is it is volatile memory. As board is power off and starts a new power cycle it must be programmed which could be quite hectic. For prototyping it was used but later when a board is placed in robot it is not good practice to program again and again.

For this reason, the Quad-SPI flash Micron part number N25Q032A of 4 MB available on board was used. This is a non-volatile flash. The flashing of program includes flashing a temporary program into FPGA which will write the program into Quad-SPI flash. But using Xilinx tools this is made bit easier. First after writing a program, pins are assigned with hardware addresses. Then the .bit file is generated. After this file is generated, memory configuration options are opened, and the flash memory is selected. And .bin file is generated from the bit file. Then by opening hardware manager, the flash memory is programmed.

Chapter 8 – COMPUTER VISION

8.1. Objective

The mobile inspection robot has the capability to track targets. A stereo camera is used to generate a depth map which can be used to provide distance estimation of the target. Custom-trained models of tracking objects can be used to detect the target and, when combined with distance estimation, can be used to effectively locate, and track a person.

8.2. Calibration

Before image processing, the stereo module needs to be calibrated. A fisheye stereo camera is used for calibration. Its rectification involves two steps:

- Fisheye Rectification
- Stereo Calibration
- Stereo Rectification Pipeline

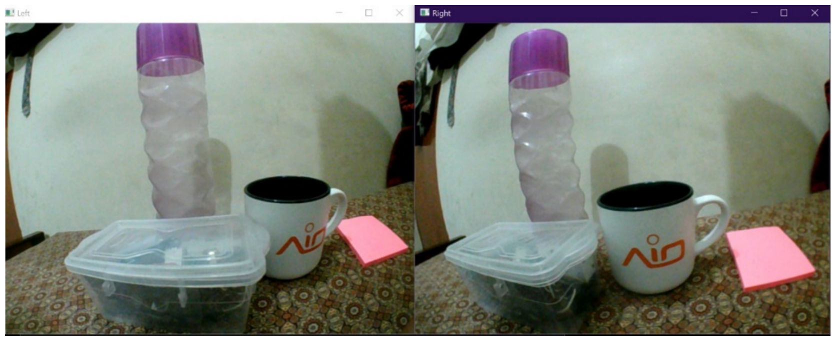


Figure 27 - Uncalibrated Fisheye Image

8.2.1. Fisheye Rectification

For the fisheye rectification, the checkerboard technique is used. A checkerboard of a n x m number of boxes is printed and attached to a solid board and using fisheye cameras, around 40 samples of it were taken. Taking more samples is computationally expensive while less samples have degraded results. The samples taken from both cameras should be synchronous in time. After taking the samples, the OpenCV library is used to find the intrinsic parameter of each camera. The effective parameter is the Distortion Coefficient. These parameters are converted to a transformation map and saved in the XML file.

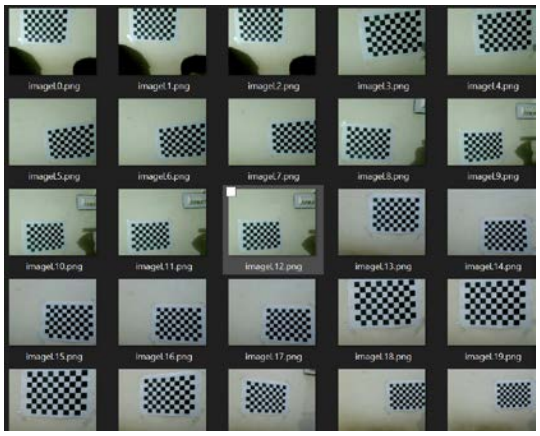


Figure 28 - Checkerboard Dataset Folder

8.2.2. Stereo Calibration

This is done to find the extrinsic parameters of the stereo module. These parameters include the Rotation Matrix and Translation Matrix for relating the stereo transformation of the two cameras in 3D space. These parameters are also converted to a transformation map and saved in the XML file.

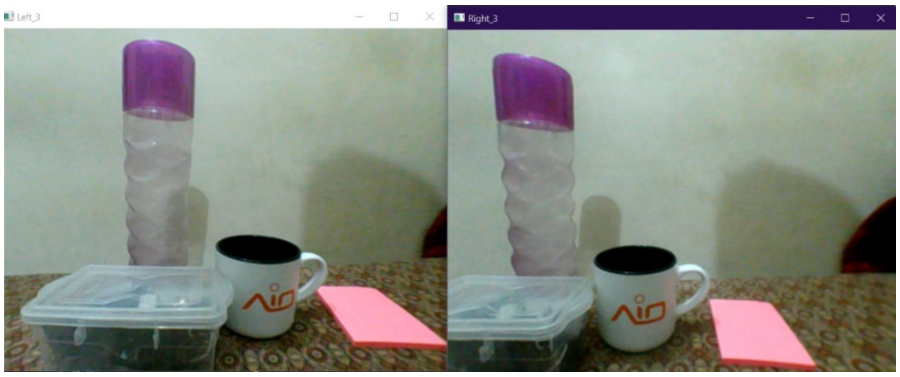


Figure 29 - Stereo Calibrated Images

8.2.3. Stereo Rectification Pipeline

Once the remapped files are saved, they can be loaded into the application program by creating a stereo rectification pipeline code. The remap function can be used to apply the transformation.

8.2.4. Depth Estimation

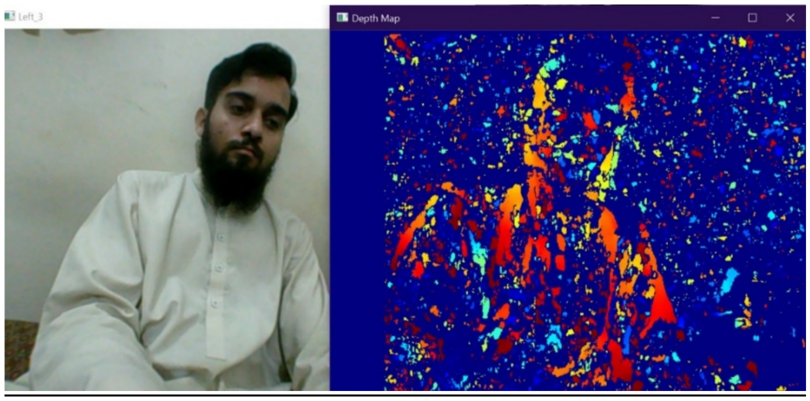


Figure 30 - Depth Map

The Depth Map can be generated from the stereo images using the OpenCV module. Using linear interpolation, the linear equation can be found for depth estimation by comparing two known distances with the internal values of the depth map. Furthermore, appropriate filters can be applied to enhance the depth map and reduce noise in the image.

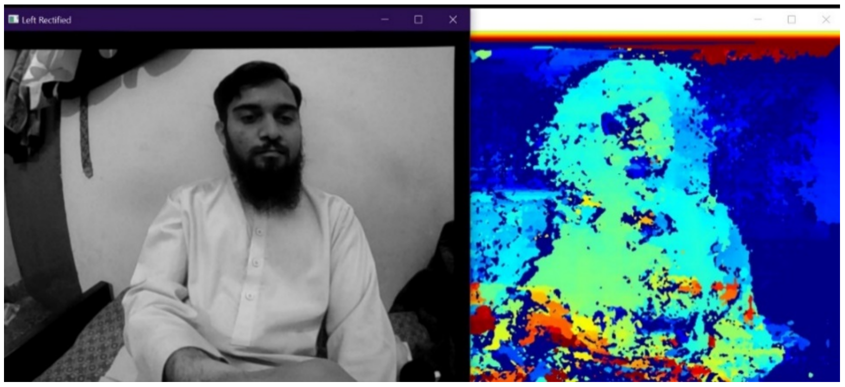


Figure 31 - Depth with Noise filters.

8.3. Detection and Recognition

YOLOv8 is used to detect objects which can be combined with stereo to estimate their depth. Facial Recognition YOLOv8 model can be easily trained with custom classes for facial recognition. Alternatively, DeepFace-based feature matching can be used to verify face. The face can be extracted through models like YOLOv8. This feature can be implemented to detect imposters during industrial inspection.

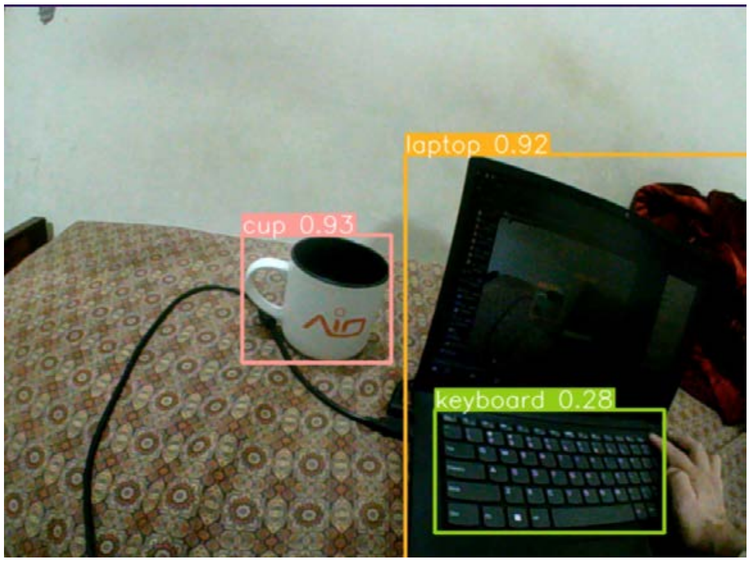


Figure 32 - Object Detection

Chapter 9 – RESULTS AND CONCLUSION

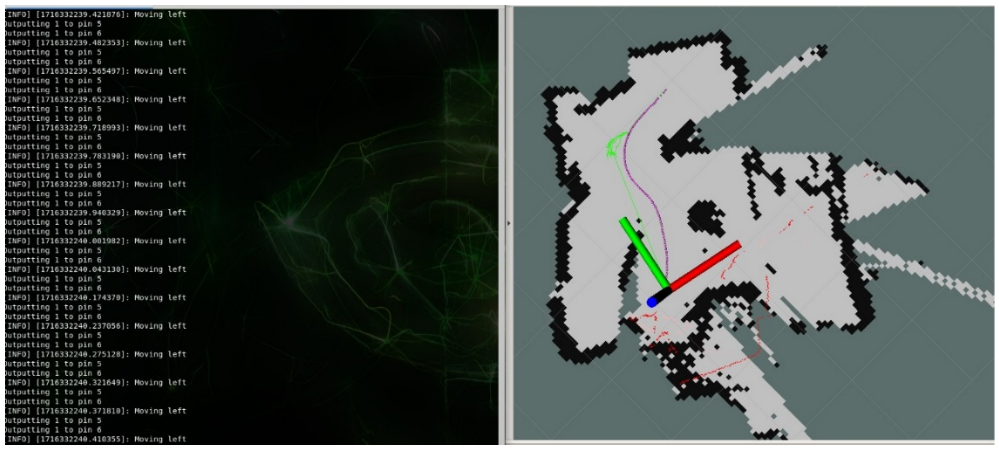
The ROS nodes were first unit tested. The LiDAR used was YDLIDAR TX20. Initially, the LiDAR failed to initialize using ROS driver. Upon research, a vendor specified GitHub repository was found with initializing parameters of all the LiDAR's of the company. So, we adjusted the driver parameters, mainly the frequency of operation of LiDAR after which the initialization become successful. For MPU 6050, we used the MPU6050 ROS driver. The driver provides linear acceleration and angular velocities. So, a complementary filter node was added to get quaternion orientation. The navigation stack was modified to use Hector SLAM for localization. In Figure 33, a successful demonstration was carried out all the process from mapping area till sending commands to base controller. Firstly, the robot generated the hector map while doing localization through hector SLAM. In the right figure, the dark grey area is unmapped, and the light grey is free space. The black region indicates obstacles. The coordinate frame with three axis indicates the location and orientation of the robot. The red axis is X axis, green is Y and blue is Z. The thin green line represents the past trajectory of the robot from the point of initialization. When the goal point was given, the navigation algorithm generated a path to the destination which can be seen in purple line. Then the algorithm started sending the commands to the base controller which can be seen in the left figure. The ROS was giving correct instructions to Jetson Nano as to which pins of the GPIO should be high to go to the destination. The robot should follow this path and reach the destination.

The FPGA is tested separately for different movements starting from moving front. Difficulty was faced to align the robot to move in a straight line due to structural deformities and weight balancing problem. Move left and move right commands were easier to implement but had imbalance in rotation speed. This imbalance can be catered from the programming end.

9.1. Limitations

- Since our lidar can work up to 8hz and hector slam has been designed to work for high frequency Lidars which is a mismatch. Although the slam works it cannot be used for high-speed mobile robots as it faces difficulty computing its change in attitude resulting in localization problems. The proposed four-legged robot is not focused on rapid speed and only performs walk motion; thus, it compensates for the frequency of the Lidar and the system works and performs mapping.

- The robot body wears out and degrades with time due to cost effective 3D printing material. It serves well for prototyping but must be replaced with a strong material in production.
- The body was not designed to accommodate double batteries and two controllers, Jetson Nano and FPGA. This created weight balancing issues that affected the movement of the robot.
- The battery system does not have a battery management system which created power outage and battery problems.



```
[INFO] [1716332239.421876]: Moving left
Outputting 1 to pin 5
Outputting 1 to pin 6
[INFO] [1716332239.482353]: Moving left
Outputting 1 to pin 5
Outputting 1 to pin 6
[INFO] [1716332239.565497]: Moving left
Outputting 1 to pin 5
Outputting 1 to pin 6
[INFO] [1716332239.652348]: Moving left
Outputting 1 to pin 5
Outputting 1 to pin 6
[INFO] [1716332239.718993]: Moving left
Outputting 1 to pin 5
Outputting 1 to pin 6
[INFO] [1716332239.783190]: Moving left
Outputting 1 to pin 5
Outputting 1 to pin 6
[INFO] [1716332239.889217]: Moving left
Outputting 1 to pin 5
Outputting 1 to pin 6
[INFO] [1716332239.940329]: Moving left
Outputting 1 to pin 5
Outputting 1 to pin 6
[INFO] [1716332240.001982]: Moving left
Outputting 1 to pin 5
Outputting 1 to pin 6
[INFO] [1716332240.043130]: Moving left
Outputting 1 to pin 5
Outputting 1 to pin 6
[INFO] [1716332240.174370]: Moving left
Outputting 1 to pin 5
Outputting 1 to pin 6
[INFO] [1716332240.237056]: Moving left
Outputting 1 to pin 5
Outputting 1 to pin 6
[INFO] [1716332240.275128]: Moving left
Outputting 1 to pin 5
Outputting 1 to pin 6
[INFO] [1716332240.321649]: Moving left
Outputting 1 to pin 5
Outputting 1 to pin 6
[INFO] [1716332240.371810]: Moving left
Outputting 1 to pin 5
Outputting 1 to pin 6
[INFO] [1716332240.410355]: Moving left
```

Figure 33 - Left: Movement command along with GPIO pin values, right: Path Planned by Navigation Stack

9.2. Conclusion

The proposed four-wheeled legged robot is made with state-of-the-art technology involving GPU-based processing and FPGA controller. It is a multi-purpose robot with the primary aim of industrial inspection. Its terrain adaptability feature enables it to work in a versatile environment. The robot is highly modular and can be modified for any use case of unmanned operation. The software solutions of the robot are distributed as ROS packages. All the packages are compliant with ROS package design norms. The robot can do autonomous navigation in mapped and unmapped environments, can be controlled wirelessly, and is equipped with vision sensors for object detection and tracking.

9.3. Future Recommendations

In the future, there is dire need of updates and work to make the robot autonomously moveable in harsh industrial environments. The following are the directions in which work is needed.

- There should be updates in the robot's body which will help to make it more stable. The larger body will help in making a stable gait motion for making motion more perfect.
- Implementation of AI algorithms and sensors to perform the industrial inspection can be upgraded.
- Upgrade to high frequency lidar for faster navigation.
- SLAM can be changed into a 3D slam so that the robot can perform mapping and localization in a better way.
- To add structural strength, the body can be manufactured from metal or high strength 3D printing material.

REFERENCES

[1] Hassan Ullah, et al. " Design of a Robotic Leg for a Four-Legged Mule Robot" *Frontiers in Robotics and AI* 7 (2020): 68.

[2] Al Arabi, Abul, et al. "Autonomous rover navigation using gps based path planning." *2017 Asia Modelling Symposium (AMS)*. IEEE, 2017.

[3] Patoliya, Jignesh, et al. "A robust autonomous navigation and mapping system based on GPS and LiDAR data for unconstrained environment." *Earth Science Informatics* 15.4 (2022): 2703-2715.

[4] Aghili, Farhad, and Alessio Salerno. "Attitude determination and localization of mobile robots using two RTK GPSs and IMU." *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009.

[5] Li, Zhaodong, Zhibao Su, and Tingting Yang. "Design of intelligent mobile robot positioning algorithm based on imu/odometer/lidar." *2019 International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC)*. IEEE, 2019.

[6] Wisth, David, Marco Camurri, and Maurice Fallon. "VILENS: Visual, inertial, lidar, and leg odometry for all-terrain legged robots." *IEEE Transactions on Robotics* 39.1 (2022): 309-326.

[7] Wisth, David, et al. "Unified multi-modal landmark tracking for tightly coupled lidar-visual-inertial odometry." *IEEE Robotics and Automation Letters* 6.2 (2021): 1004-1011.

[8] Wooden, David, et al. "Autonomous navigation for BigDog." *2010 IEEE international conference on robotics and automation*. Ieee, 2010.

[9] Savaria, Daniel T., and Ramprasad Balasubramanian. "V-SLAM: Vision-based simultaneous localization and map building for an autonomous mobile robot." *2010 IEEE Conference on Multisensor Fusion and Integration*. IEEE, 2010.

[10] Camurri, Marco, et al. "Pronto: A multi-sensor state estimator for legged robots in real-world scenarios." *Frontiers in Robotics and AI* 7 (2020): 68.

[11] Liu, Wei, et al. "Ssd: Single shot multibox detector." *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer International Publishing, 2016.

[12] Gorobets, N. G., K. A. Kuznetsov, and O. V. Tsvietaieva. "BOSTON DYNAMICS."

[13] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[14] Foa, Alessandro. "Object Detection in Object Tracking System for Mobile Robot Application." (2019).

[15] Pereira, Ricardo, et al. "Sort and deep-SORT based multi-object tracking for mobile robotics: Evaluation with new data association metrics." *Applied Sciences* 12.3 (2022): 1319.

[16] Mori, Kimihiro, et al. "A Study of Trot Gait Control System of a Quadruped Robot Using IMU Sensor." *2022 61st Annual Conference of the Society of Instrument and Control Engineers (SICE)*. IEEE, 2022.

[17] Medeiros, Vivian S., et al. "Trajectory optimization for wheeled-legged quadrupedal robots driving in challenging terrain." *IEEE Robotics and Automation Letters* 5.3 (2020): 4172-4179.

[18] Hutter, Marco, et al. "Anymal-a highly mobile and dynamic quadrupedal robot." 2016 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, 2016.

[19] Semini, Claudio. "HyQ-design and development of a hydraulically actuated quadruped robot." *Doctor of Philosophy (Ph. D.), University of Genoa, Italy* (2010).

[20] Katz, Benjamin, Jared Di Carlo, and Sangbae Kim. "Mini cheetah: A platform for pushing the limits of dynamic quadruped control." *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019.

[21] Lim, David. “W1 Wheeled Quadruped Robot.” LIMX DYNAMICS, 2023.

ANNEXURE A - VERILOG CODE

a) PWM generation Module

For servo control we have a PWM generation Module. This module basically corelate angles into PWM signal. It executes the Linera equation which turns degrees between limit 180 degrees into number of clocks of high and low signal which are implemented in another always block. It takes clock for synchronization of wave signal and an 8-bit angle value on which servo is directed as inputs. It generates a PWM signal based on the angle as an output.

Code

```
module PWM_gen(clk,servo_dig,pwm);
input clk;
input wire [7:0] servo_dig;
output reg pwm;

reg [31:0] val=0;
reg [31:0] counter=0;

always @(servo_dig)
        begin
                val=210000 + (24000 * servo_dig) / 180;
        end
always @(posedge clk)
        begin
                if (counter<=240000)
                        begin
                                counter=counter+1;
                        end
                else
                        begin
                                counter=0;
                        end
                if (counter<=val)
                        begin
                                pwm=0;
                        end
                else
                        begin
                                pwm=1;
                        end
        end
endmodule
```

b) Leg Angles Module

This is one of the most crucial modules in forming a step of robot from synchronous motion of 12 servos. It takes following inputs and clock for synchronization of signal. A, B, C and D are 4 signals from Jetson nano for mode determination. This module detects the mode and writes 8-bit servo angles accordingly in their respective registers. Which are read by PWM module for PWM generation. One more important aspect is that it counts the time by using clock and changes the 8 bit servo value accordingly to execute the 1 full step of robot which is programmed.

Code

```
module Leg_angles(run_clk, A, B, C, D, pos_dig_1, pos_dig_2, pos_dig_3, pos_dig_4, pos_dig_5,
pos_dig_6, pos_dig_7, pos_dig_8, pos_dig_9, pos_dig_10, pos_dig_11, pos_dig_12);
input run_clk, A, B, C, D;
output wire [7:0] pos_dig_1, pos_dig_2, pos_dig_3, pos_dig_4, pos_dig_5, pos_dig_6, pos_dig_7,
pos_dig_8, pos_dig_9, pos_dig_10, pos_dig_11, pos_dig_12;

reg [7:0] pos1  = 100;
reg [7:0] pos2  = 100;
reg [7:0] pos3  = 100;
reg [7:0] pos4  = 100;
reg [7:0] pos5  = 100;
reg [7:0] pos6  = 100;
reg [7:0] pos7  = 100;
reg [7:0] pos8  = 100;
reg [7:0] pos9  = 100;
reg [7:0] pos10 = 100;
reg [7:0] pos11 = 100;
reg [7:0] pos12 = 100;
reg [31:0] count = 0;

always @ (posedge run_clk)
        begin
                if (count==4_000_000)
                begin
                        count = 0;
                end
                else
                begin
                        count = count + 1;
                end
        end
always @ (posedge run_clk)
begin
  if (A == 0 && B == 0 && C == 0 && D == 0) //Standing
          begin
                 if (count<=4_000_000)
                begin
          pos1  = 105;
          pos2  = 145;
          pos3  = 95;
          pos4  = 105;
          pos5  = 65;
          pos6  = 115;
          pos7  = 110;
          pos8  = 60;
          pos9  = 70;
          pos10 = 110;
          pos11 = 150;
          pos12 = 110;
```

```
                    end
            end
    else if (A == 0 && B == 0 && C == 0 && D == 1) //Forward
        begin
                      if (count<=1_000_000)
                    begin
          pos1  = 105;
          pos2  = 155;
          pos3  = 120;
          pos4  = 105;
          pos5  = 75;
          pos6  = 140;
          pos7  = 110;
          pos8  = 45;
          pos9  = 40;
          pos10 = 110;
          pos11 = 150;
          pos12 = 85;
                    end
                    else if (count>=1_000_000 && count<=2_000_000)
                    begin
          pos1  = 105;
          pos2  = 165;
          pos3  = 90;
          pos4  = 105;
          pos5  = 75;
          pos6  = 115;
          pos7  = 110;
          pos8  = 35;
          pos9  = 75;
          pos10 = 110;
          pos11 = 150;
          pos12 = 110;
       end
```

```
                else if (count>=2_000_000 && count<=3_000_000)
                begin
        pos1  = 105;
          pos2  = 145;
          pos3  = 70;
          pos4  = 105;
          pos5  = 55;
          pos6  = 80;
          pos7  = 110;
          pos8  = 65;
          pos9  = 95;
          pos10 = 110;
          pos11 = 160;
          pos12 = 135;
       end
                else if (count>=3_000_000 && count<=4_000_000)
                begin
          pos1  = 105;
          pos2  = 145;
          pos3  = 95;
          pos4  = 105;
          pos5  = 40;
          pos6  = 120;
          pos7  = 110;
          pos8  = 65;
          pos9  = 70;
          pos10 = 110;
          pos11 = 170;
          pos12 = 105;
       end
   end

else if (A == 0 && B == 0 && C == 1 && D == 0) //Right
   begin
```

```
                if (count<=1_000_000)
               begin
   pos1  = 105;
   pos2  = 155;
   pos3  = 120;
   pos4  = 105;
   pos5  = 40;
   pos6  = 120;
   pos7  = 110;
   pos8  = 60;
   pos9  = 70;
   pos10 = 110;
   pos11 = 150;
   pos12 = 85;
               end
               else if (count>=1_000_000 && count<=2_000_000)
               begin
   pos1  = 105;
   pos2  = 165;
   pos3  = 90;
   pos4  = 105;
   pos5  = 50;
   pos6  = 80;
   pos7  = 110;
   pos8  = 60;
   pos9  = 95;
   pos10 = 110;
   pos11 = 150;
   pos12 = 110;
end
               else if (count>=2_000_000 && count<=3_000_000)
               begin
   pos1  = 105;
   pos2  = 145;
```

```
        pos3  = 70;
        pos4  = 105;
        pos5  = 65;
        pos6  = 115;
        pos7  = 110;
        pos8  = 35;
        pos9  = 75;
        pos10 = 110;
        pos11 = 160;
        pos12 = 135;
      end
                else if (count>=3_000_000 && count<=4_000_000)
                begin
        pos1  = 105;
        pos2  = 145;
        pos3  = 95;
        pos4  = 105;
        pos5  = 65;
        pos6  = 140;
        pos7  = 110;
        pos8  = 45;
        pos9  = 40;
        pos10 = 110;
        pos11 = 170;
        pos12 = 105;
      end
    end

  else if (A == 0 && B == 0 && C == 1 && D == 1) //Left
    begin
                 if (count<=1_000_000)
                begin
        pos1  = 105;
        pos2  = 155;
  pos3  = 120;
  pos4  = 105;
  pos5  = 65;
  pos6  = 140;
  pos7  = 110;
  pos8  = 45;
  pos9  = 40;
  pos10 = 110;
  pos11 = 150;
  pos12 = 85;
         end
         else if (count>=1_000_000 && count<=2_000_000)
         begin
  pos1  = 105;
  pos2  = 165;
  pos3  = 90;
  pos4  = 105;
  pos5  = 65;
  pos6  = 115;
  pos7  = 110;
  pos8  = 35;
  pos9  = 75;
  pos10 = 110;
  pos11 = 150;
  pos12 = 110;
end
         else if (count>=2_000_000 && count<=3_000_000)
         begin
  pos1  = 105;
  pos2  = 145;
  pos3  = 70;
  pos4  = 105;
  pos5  = 50;
  pos6  = 80;
```

```
            pos7  = 110;
            pos8  = 60;
            pos9  = 95;
            pos10 = 110;
            pos11 = 160;
            pos12 = 135;
        end
                    else if (count>=3_000_000 && count<=4_000_000)
                    begin
            pos1  = 105;
            pos2  = 145;
            pos3  = 95;
            pos4  = 105;
            pos5  = 40;
            pos6  = 120;
            pos7  = 110;
            pos8  = 60;
            pos9  = 70;
            pos10 = 110;
            pos11 = 170;
            pos12 = 105;
        end
    end

else if (A == 0 && B == 1) //Wheeled
    begin
        if (count<=4_000_000)
            begin
                pos1  = 105;
                pos2  = 120;
                pos3  = 35;
                pos4  = 105;
                pos5  = 90;
                pos6  = 170;
```

```
                    pos7  = 110;
                    pos8  = 90;
                    pos9  = 160;
                    pos10 = 110;
                    pos11 = 120;
                    pos12 = 35;
                end
        end
end
assign pos_dig_1  = pos1;
assign pos_dig_2  = pos2;
assign pos_dig_3  = pos3;
assign pos_dig_4  = pos4;
assign pos_dig_5  = pos5;
assign pos_dig_6  = pos6;
assign pos_dig_7  = pos7;
assign pos_dig_8  = pos8;
assign pos_dig_9  = pos9;
assign pos_dig_10 = pos10;
assign pos_dig_11 = pos11;
assign pos_dig_12 = pos12;

endmodule
```

c) Wheeled Module

This module is responsible for control of wheeled mode. It is similar to leg angles module, but it executes Wheeled motion by controlling 4 PWMs for speed and direction pins accordingly. It takes clock for synchronization of signal. A, B, C and D are 4 signal from Jetson nano for mode determination. It reads the mode and implement it based on pre-programmed behaviour.

Code

```
module Wheels(run_clk, A, B, C, D, M1_Enbl, M1_Fwd, M1_Bcwd, M2_Enbl, M2_Fwd, M2_Bcwd,
M3_Enbl, M3_Fwd, M3_Bcwd, M4_Enbl, M4_Fwd, M4_Bcwd);
input run_clk, A, B, C, D;
output wire M1_Enbl, M1_Fwd, M1_Bcwd, M2_Enbl, M2_Fwd, M2_Bcwd, M3_Enbl, M3_Fwd,
M3_Bcwd, M4_Enbl, M4_Fwd, M4_Bcwd;

reg  [7:0] M1_Percentage    =0;
reg  M1_Forward             =0;
reg  M1_Backwards           =0;
reg  [7:0] M2_Percentage    =0;
reg  M2_Forward             =0;
reg  M2_Backwards           =0;
reg  [7:0] M3_Percentage    =0;
reg  M3_Forward             =0;
reg  M3_Backwards           =0;
reg  [7:0] M4_Percentage    =0;
reg  M4_Forward             =0;
reg  M4_Backwards           =0;

always @ (posedge run_clk)
begin
   if (A == 0 && B == 1 && C == 0 && D == 0) //Standing
            begin
         M1_Percentage   = 0;
         M1_Forward      = 0;
         M1_Backwards    = 0;
         M2_Percentage   = 0;
         M2_Forward      = 0;
         M2_Backwards    = 0;
         M3_Percentage   = 0;
         M3_Forward      = 0;
         M3_Backwards    = 0;
         M4_Percentage   = 0;
```

```
        M4_Forward      = 0;
        M4_Backwards    = 0;
           end

else if (A == 0 && B == 1 && C == 0 && D == 1) //Forwad motion
      begin
          M1_Percentage   = 100;
          M1_Forward      = 1;
          M1_Backwards    = 0;
          M2_Percentage   = 80;
          M2_Forward      = 1;
          M2_Backwards    = 0;
          M3_Percentage   = 80;
          M3_Forward      = 1;
          M3_Backwards    = 0;
          M4_Percentage   = 100;
          M4_Forward      = 1;
          M4_Backwards    = 0;
      end

else if (A == 0 && B == 1 && C == 1 && D == 0) //Right motion
    if (D == 0) //Right motion
      begin
          M1_Percentage   = 100;
          M1_Forward      = 0;
          M1_Backwards    = 1;
          M2_Percentage   = 100;
          M2_Forward      = 1;
          M2_Backwards    = 0;
          M3_Percentage   = 100;
          M3_Forward      = 1;
          M3_Backwards    = 0;
          M4_Percentage   = 100;
          M4_Forward      = 0;
```

```
            M4_Backwards    = 1;
        end

    else if (A == 0 && B == 1 && C == 1 && D == 1) //Left motion
      else if (D == 1) //Left motion
        begin
            M1_Percentage   = 100;
            M1_Forward      = 1;
            M1_Backwards    = 0;
            M2_Percentage   = 100;
            M2_Forward      = 0;
            M2_Backwards    = 1;
            M3_Percentage   = 100;
            M3_Forward      = 0;
            M3_Backwards    = 1;
            M4_Percentage   = 100;
            M4_Forward      = 1;
            M4_Backwards    = 0;
        end
end
assign M1_Fwd     = M1_Forward;
assign M1_Bcwd    = M1_Backwards;
assign M2_Fwd     = M2_Forward;
assign M2_Bcwd    = M2_Backwards;
assign M3_Fwd     = M3_Forward;
assign M3_Bcwd    = M3_Backwards;
assign M4_Fwd     = M4_Forward;
assign M4_Bcwd    = M4_Backwards;
Motor_PWM M1_p(run_clk, M1_Percentage, M1_Enbl);
Motor_PWM M2_p(run_clk, M2_Percentage, M2_Enbl);
Motor_PWM M3_p(run_clk, M3_Percentage, M3_Enbl);
Motor_PWM M4_p(run_clk, M4_Percentage, M4_Enbl);

Endmodule
```

d) Main Module

This the main module which calls all other modules in it for execution of logic. It has clock and jetson nano 4 pins as inputs and outputs ground relay and signal pins for 12 servos and 4 motors.

Code

```
module gen(gen_clk, A, B, C, D, Led, Gnd_Relay, PWM_L1_A, PWM_L1_B, PWM_L1_C, PWM_L2_A,
PWM_L2_B, PWM_L2_C, PWM_L3_A, PWM_L3_B, PWM_L3_C, PWM_L4_A, PWM_L4_B,
PWM_L4_C, M1_En, M1_F, M1_B, M2_En, M2_F, M2_B, M3_En, M3_F, M3_B, M4_En, M4_F, M4_B);
input gen_clk, A, B, C, D;
output Gnd_Relay, Led, PWM_L1_A, PWM_L1_B, PWM_L1_C, PWM_L2_A, PWM_L2_B, PWM_L2_C,
PWM_L3_A, PWM_L3_B, PWM_L3_C, PWM_L4_A, PWM_L4_B, PWM_L4_C, M1_En, M1_F, M1_B,
M2_En, M2_F, M2_B, M3_En, M3_F, M3_B, M4_En, M4_F, M4_B;

wire [7:0] dig_L1_A;
wire [7:0] dig_L1_B;
wire [7:0] dig_L1_C;
wire [7:0] dig_L2_A;
wire [7:0] dig_L2_B;
wire [7:0] dig_L2_C;
wire [7:0] dig_L3_A;
wire [7:0] dig_L3_B;
wire [7:0] dig_L3_C;
wire [7:0] dig_L4_A;
wire [7:0] dig_L4_B;
wire [7:0] dig_L4_C;

Leg_angles Walk(gen_clk, A, B, C, D, dig_L1_A, dig_L1_B, dig_L1_C, dig_L2_A, dig_L2_B, dig_L2_C,
dig_L3_A, dig_L3_B, dig_L3_C, dig_L4_A, dig_L4_B, dig_L4_C);

PWM_gen servo_L1_A(gen_clk, dig_L1_A, PWM_L1_A);
PWM_gen servo_L1_B(gen_clk, dig_L1_B, PWM_L1_B);
PWM_gen servo_L1_C(gen_clk, dig_L1_C, PWM_L1_C);
PWM_gen servo_L2_A(gen_clk, dig_L2_A, PWM_L2_A);
PWM_gen servo_L2_B(gen_clk, dig_L2_B, PWM_L2_B);
PWM_gen servo_L2_C(gen_clk, dig_L2_C, PWM_L2_C);
PWM_gen servo_L3_A(gen_clk, dig_L3_A, PWM_L3_A);
PWM_gen servo_L3_B(gen_clk, dig_L3_B, PWM_L3_B);
PWM_gen servo_L3_C(gen_clk, dig_L3_C, PWM_L3_C);
PWM_gen servo_L4_A(gen_clk, dig_L4_A, PWM_L4_A);
PWM_gen servo_L4_B(gen_clk, dig_L4_B, PWM_L4_B);
PWM_gen servo_L4_C(gen_clk, dig_L4_C, PWM_L4_C);

assign Gnd_Relay=1;
Wheels Move(gen_clk, A, B, C, D, M1_En, M1_F, M1_B, M2_En, M2_F, M2_B, M3_En, M3_F, M3_B,
M4_En, M4_F, M4_B);

endmodule
```