QUADRUPED ROBOT

PROJECT REPORT

Submitted by

Abdul Ahad
Atul Vyshnav R
Krishnanand K
Shinas Shaji

Under the supervision of
Dr. Sunil Kumar

Department of Electrical and Electronics Engineering
Govt. Rajiv Gandhi Institute of Technology, Kottayam
Velloor, Pampady, Kottayam - 686501

Department of Electrical and Electronics Engineering
Govt. Rajiv Gandhi Institute of Technology, Kottayam
Velloor, Pampady, Kottayam - 686501

CANDIDATE'S DECLARATION

We, Abdul Ahad (Reg No: KTE18EE002), Atul Vyshnav R (Reg No: KTE18EE022), Krishnanand K (Reg No: KTE18EE040) & Shinas Shaji (Reg No: KTE18EE053) students of B.Tech at APJ Abdul Kalam Technological University, hereby declare that the Project Dissertation titled "Quadruped Robot", which is submitted by us to the Department of Electrical and Electronics Engineering, Govt. Rajiv Gandhi Institute of Technology, Kottayam, in fulfillment of the requirement for awarding of the Bachelor of Technology degree, is not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma, Fellowship or other similar title or recognition.

Abdul Ahad	**Atul Vyshnav R**	**Krishnanand K**	**Shinas Shaji**
(KTE18EE002)	**(KTE18EE022)**	**(KTE18EE040)**	**(KTE18EE053)**

Date: June 9, 2022 Place: Pampady, Kottayam, Kerala

Department of Electrical and Electronics Engineering
Govt. Rajiv Gandhi Institute of Technology, Kottayam
Velloor, Pampady, Kottayam - 686501

CERTIFICATE

This is to certify that this report titled ”Quadruped Robot” submitted by Mr. Abdul Ahad (Reg No: KTE18EE002), Mr. Atul Vyshnav R (Reg No: KTE18EE022), Mr. Krishnanand K (Reg No: KTE18EE040) and Mr. Shinas Shaji (Reg No: KTE18EE053) to APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of Degree of Bachelor of Technology in Electrical and Electronics Engineering, is a bonafide record of the project carried out by them under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

ABSTRACT

Keywords: Quadruped, Autonomous, Path Planning, Servo, Computer Vision, Inverse Kinematics, LiDAR

Quadruped robots are highly efficient and have several advantages when compared to other wheeled and two-legged robots. The fact that it has 4 legs for locomotion creates extra possibilities for movement, stability and dynamic maneuvarability. This proves to be perfect for navigating complex terrain. Additionally, the low center of gravity of these robots provides more stability and balance to its movement. Its design and gait pattern are heavily inspired from that of four-legged animals. Our project implements the movement functionalities of the quadruped and its navigation through an environment. Visual perception is done by means of computer vision coded in python for the purpose of autonomous navigation. A Raspberry Pi is used as the main brain of the robot, and an Arduino Mega controls the actuators. Several navigation routines are solved using path planning algorithms, stair detection algorithms, visual odometry etc. Inverse Kinematics enables endpoint control of each of the four legs, computing the angles for leg joints for the actuation of the servos. Stereo Camera Setups, ultrasonic sensors and a LiDAR unit is used for environment perception and mapping, and the data from these are used for the overall perception of the environment and navigation of the robot.

ACKNOWLEDGEMENT

The successful completion of any task is incomplete and meaningless without giving any due credit to the people who made it possible, and without which the project would not have been successful.

First and foremost, we are grateful to Dr. Johnson Mathew (HoD of Electrical and Electronics Engineering), Dr. Prince A (Professor), Dr. Dolly Mary A (Associate Professor), Dr. Shanifa Nissam (Assistant Professor), and Mr. Prof. Peter K Abraham (Assistant Professor) for their constant guidance and support. We would also like to thank our guide Dr. Sunil Kumar P R (Associate Professor) for his motivation and guidance in undertaking this endeavour.

We would also like to take this moment to show our thanks and gratitude to one and all who indirectly or directly have given us their hand in this challenging task. We feel happy, joyful and content in expressing our vote of thanks to all those who have helped us and guided us in presenting this work as our Final Year project. Last, but not the least, we thank our well-wishers, friends and parents for always being with us in every sense and constantly supporting us in every possible way whenever possible.

Dr. Johnson Mathew	**Dr. Prince A**	**Dr. Sunil Kumar P R**
Head of Department	**Coordinator**	**Supervisor**

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

The necessity and requirements of different kinds of artificially intelligent robots are increasing day by day in the modern world. The latest innovations in technology are making it possible for us to further develop in different areas of human life. One of such up and rising sectors is the robotic industry.

In the world of robots, quadrupeds, or robots with 4 legs, are best equipped with the ability to maneuver complex terrain much faster and with enhanced stability. It follows the gait patterns of animals and are versatile in locomotion and movement. That is the major intent behind the development of these kinds of robots by the academia, companies, and robot enthusiasts around the world.

Mobile robots like quadrupeds have extensive applications and the potential to be one of the most important innovations in the future of technology. Quadruped robots are superior in ability when commpared with wheeled and tracked robots due to its potential to explore in highly variable terrain like humans and animals. It also provides much more stability than a humanoid robots because of its 4 legged from that enables it to exploit the advantages of legged locomotion. The dynamic range of feet placement extends the reach and limits the constraints on directional movement. In addition to this, the low center of gravity accounts for enhanced balance and stability preventing the robot from toppling over in difficult circumstances, hence reducing damage and failure of the robot in operation. Considering the dynamic and stable capabilities of a quadruped robot, we have taken inspiration from the immaculate development in the realm of quadruped robots and have tried to create one of our own.

The robot works with information from stereo cam setups, LiDAR data and even auxiliary sensors like ultrasonic range sensors for perception of the environment. The robot uses cheap, commercially available parts, and has been designed with modularity in mind.

Chapter 2

Literature Survey

Our project to design and develop a quadruped robot was inspired by the Boston Dynamics Spot® robot.

1. Self Driving Car Specialization - *University of Toronto*: This course gave us a comprehensive understanding of state-of-the-art engineering practices used in the Autonomous Industry. Interacted with real data sets from an autonomous vehicle (AV) - all through hands-on projects using simulatorions.Helped us to Understand LIDAR scan matching and the Iterative Closest Point algorithm Learned various algorithms related to Visual perception, Visual odometry,State Estimation,static and dynamic object detection, Localization and autonomous navigation.

   This also Enabled us to to find ways to get the shortest path over a graph or road network using Dijkstra's and the A* algorithm, use finite state machines to select safe behaviors to execute, and design optimal, smooth paths and velocity profiles to navigate safely around obstacles while obeying traffic laws and to also build occupancy grid maps of static elements in the environment and learn how to use them for efficient collision checking.This course gaves an overall idea about the ability to construct a full self-driving planning solution, keeping the vehicle safe at all times. Many of the learnings from this course have been highly usefull for building this project.

2. Robotic Path planning A* and D* - *Carnegie Mellon*: Thwe exploredse studies underlying algorithmic techniques used for planning and decision-making in robotics and examines case studies in ground and aerial robots, humanoids, mobile manipulation platforms and multi-robot systems. Planning and Decision-making are critical components of autonomy in robotic systems. These components are responsible for making decisions that range from path planning and motion planning to coverage and task planning to taking actions that help robots understand the world around them better. This course we explored various underlying algorithmic techniques used for planning and decision-making in robotics and examines case studies in ground and aerial robots, humanoids, mobile manipulation platforms and multi-robot systems.

3. Boston Dynamics spot Documentation : From this documentation we did Exploratory study and it gave us the inspiration for building quadruped robots and learn about their various applications.Spot is an agile mobile robot that navigates terrain with unprecedented mobility, allowing you to automate routine inspection tasks and data capture safely, accurately, and frequently. The results are Safer, more efficient and more predictable operations and solutions in the field of robotic applications. It has 360 Perception

and dynamic balance.Spots base platform provides advanced mobility and perception to navigate stairs, gravel, and rough terrain while collecting 2D and 3D information with on board-sensors. We can also add payloads provided by Boston Dynamics or third-parties to enhance Spots sensing and data processing capabilities

4. Hello Real World With ROS : Robotic Operating System is a meta operating system which runs on top of other Operating systems like linux. It's beneficial in the field of robotics, because it includes several packages and functons that are helpfull in the development of robots. Its speccially dedicated for robotic systems. this course helped us in understanding the application of ROS for Robot programming and Simulation. We also learned how to use ROS communication tools (topics, services, actions) to exchange information between functional modules, Introduction simulation environment. This enalbed us to run simulations for the robot using Rviz and Gazebo, Simulation features available in ROS packages for visualization and creation of a custom environment with a robot Mapping of the robot environment and navigation with a mobile robot.

5. The simulated control framework is based on "*Hierarchical controller for highly dynamic locomotion utilizing pattern modulation and impedance control : implementation on the MIT Cheetah robot*".This thesis presents a hierarchical control algorithm for quadrupedal locomotion. We address three challenges in developing a controller for high-speed running: locomotion stability, control of ground reaction force, and coordination of four limbs. To tackle these challenges, the proposed algorithm employs three strategies. Leg impedance control provides programmable virtual compliance of each leg which achieve self-stability in locomotion. The four legs exert forces to the ground using equilibrium-point hypothesis. A gait pattern modulator imposes a desired footfall sequence. The control algorithm is verified in a dynamic simulator constructed using MATLAB and then in the subsequent experiments on the MIT Cheetah robot. The experiments on the MIT Cheetah robot demonstrates high speed trot running reaching up to the speed of 6 m/s on a treadmill. This speed corresponds to a Froude number (Fr = 7.34), which is comparatively higher than other existing quadrupedal robots.

Chapter 3

Design

3.1 Component Design

This section details the design of the components used in this project.

A very critical part of the design of a quadruped robot is the design of the mechanical legs. These legs must be able to handle the load of the robot and any other load that may be placed on it (referred to as static loads), along with withstanding the dynamic loads experienced when the robot is in motion. The design of mechanical legs involves choosing the actuator and its transmission mechanism, designing the structure of the mechanical leg, and choosing an efficient method of fabricating said mechanical leg.

The computing and control hardware also form a crucial part of the robot. The control hardware interfaces with the actuator and drives the actuator optimally to achieve the desired result. The control harware may use a lower level microcontroller to handle time-critical tasks such as leg actuator and dynamics control. A higher level computing unit may handle compute-intensive software and algorithms which are soft time-constrained, and also offer a user interface for intuitive control of the robot.

The power requirements of the actuators, compute, and control architectures also need to be met with high efficiency and with minimum weight, to minimize the dead weight of the robot and heat dissipation.

3.1.1 MG995 - Servo Motor

MG995 is a servo motor that is popular for its acceptable performance and low price. The motor is used in many applications, including robotics and drones.The servo is suited for designing robotic arm in which wear and tear of motor is high. Being metal geared, the servo has long life and can be installed on system like robotic arm were motor work is huge.

MG995 has three terminals, as mentioned in pin diagram given in 3.1. Pin function are given in 3.1.

Pin	Name	Function
1	Signal pin (Orange pin)	Control PWM signal stating axis position
2	VCC (Red pin)	Input voltage from 5V - 7.2V power supply
3	Ground (Brown pin)	Ground terminal

Table 3.1: MG995 Servo Pinout

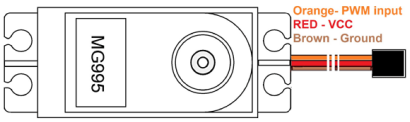


Figure 3.1: MG995 Servo Pinout

MG995 Features and Electrical characteristics

- Durable metal-geared servo
- Stable and shock proof double ball bearing design
- High speed rotation for quick response
- Fast control response
- Constant torque throughout the servo travel range
- Excellent holding power
- Weight: $55g$
- Dimension: $40.7 \times 19.7 \times 42.9mm$
- Operating voltage range: $4.8V$ to $7.2V$
- Stall torque: $9.4kg/cm\ (4.8v); 11kg/cm\ (6v)$
- Operating speed: $0.2s/60^o\ (4.8V),\ 0.16s/60^o\ (6V)$
- Rotational degree: 180^o
- Dead band width: $5\mu s$
- Operating temperature range: $0^oC\ to\ +55^oC$

- Current draw at idle: 10*mA*
- No load operating current draw: 170*mA*
- Current at maximum load: 1200*mA*

For controlling of servo there are only two important things to remember:

- Frequency of PWM: The MG995 takes in PWM signals of frequency 50Hz; any higher or lower frequency PWM will lead to error. As shown in Figure 3.2 the every single cycle of PWM needs to be of 20ms width for 50Hz frequency.
- Duty cycle of PWM: The duty cycle of PWM (or ratio of ON time to total cycle time) determines the position of servo axis.

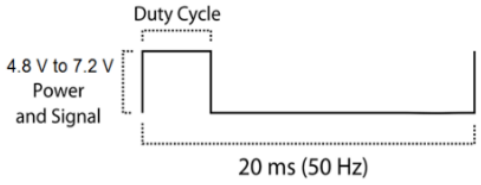


Figure 3.2: PWM Period

Schematic Diagram

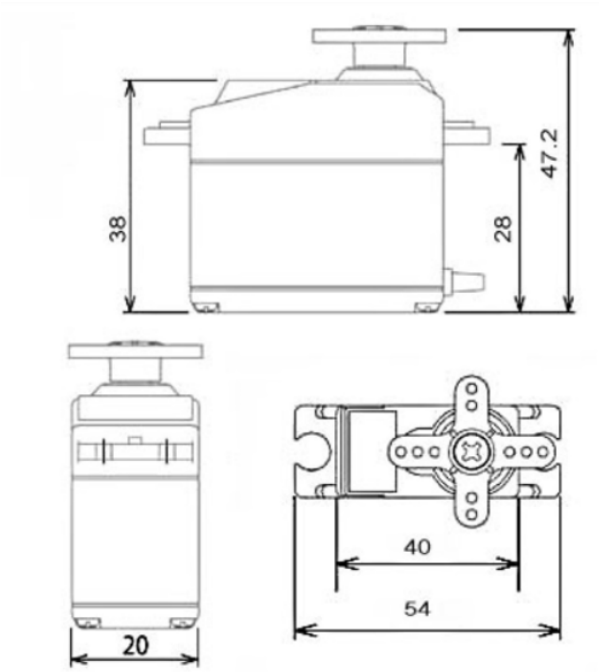


Figure 3.3: MG995 Schematic Diagram

3.1.2 Raspberry Pi

Overview

Raspberry Pi 4 Model B is the latest product in the popular Raspberry Pi range of computers. It offers ground-breaking increases in processor speed, multimedia performance, memory, and connectivity compared to the prior-generation Raspberry Pi 3 Model B+, while retaining backwards compatibility and similar power consumption. For the end user, Raspberry Pi 4 Model B (in Figure 3.4) provides desktop performance comparable to entry-level x86 PC systems.

Key features include a high-performance 64-bit quad-core processor, dual-display support at resolutions up to 4K via a pair of micro-HDMI ports, hardware video decode at up to 4Kp60, up to 4GB of RAM, dual-band 2.4/5.0 GHz wireless LAN, Bluetooth 5.0, Gigabit Ethernet, USB 3.0, and PoE capability.

In this project the Raspberry Pi acts as the main processor. It runs all the main python programming, for navigation control and processing. It passes the control programming to the Arduino Mega.

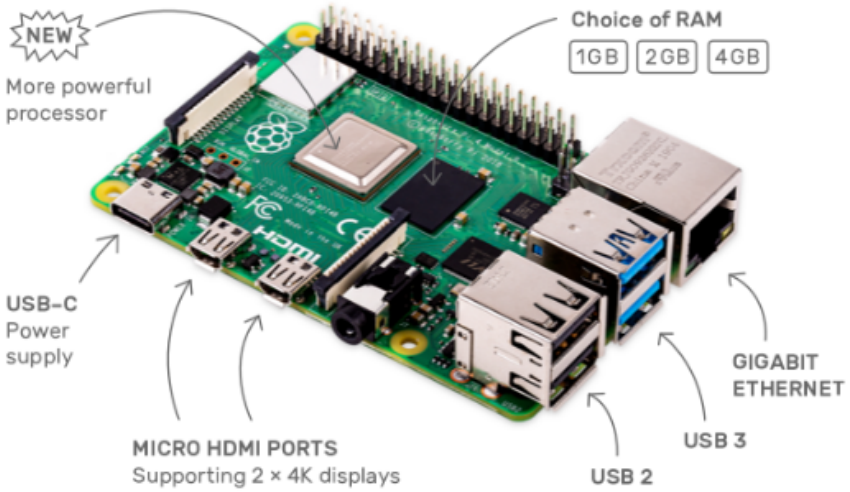


Figure 3.4: Raspberry Pi 4B

Specifications

- Processor: Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- Memory: 1GB, 2GB or 4GB LPDDR4 (depending on model)
- Connectivity: 2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac wireless; Gigabit Ethernet; 2 $\times$ USB 3.0 ports; 2 $\times$ USB 2.0 ports
- GPIO: Standard 40-pin GPIO header (fully backwards-compatible with previous boards)
- Video and sound: 2 $\times$ micro HDMI ports (up to 4Kp60 supported); 2-lane MIPI DSI display port; 2-lane MIPI CSI camera port; 4-pole stereo audio and composite video port
- Multimedia: H.265 (4Kp60 decode); H.264 (1080p60 decode, 1080p30 encode); OpenGL ES, 3.0 graphics

- SD card support: Micro SD card slot for loading operating system and data storage
- Input power: 5V DC via USB-C connector (minimum 3A1); 5V DC via GPIO header (minimum 3A1); Power over Ethernet (PoE)-enabled (requires separate PoE HAT)
- Environment: Operating temperature $0 – 50^oC$

3.1.3 Arduino Mega

Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega 2560 board is compatible with most shields designed for the Uno and the former boards Duemilanove or Diecimila.

Here we use the Arduino Mega to control the actuators. It's programmed in the arduino IDE embedded C language. Inorder to split the load, we are using 2 microcontrollers on the bot.

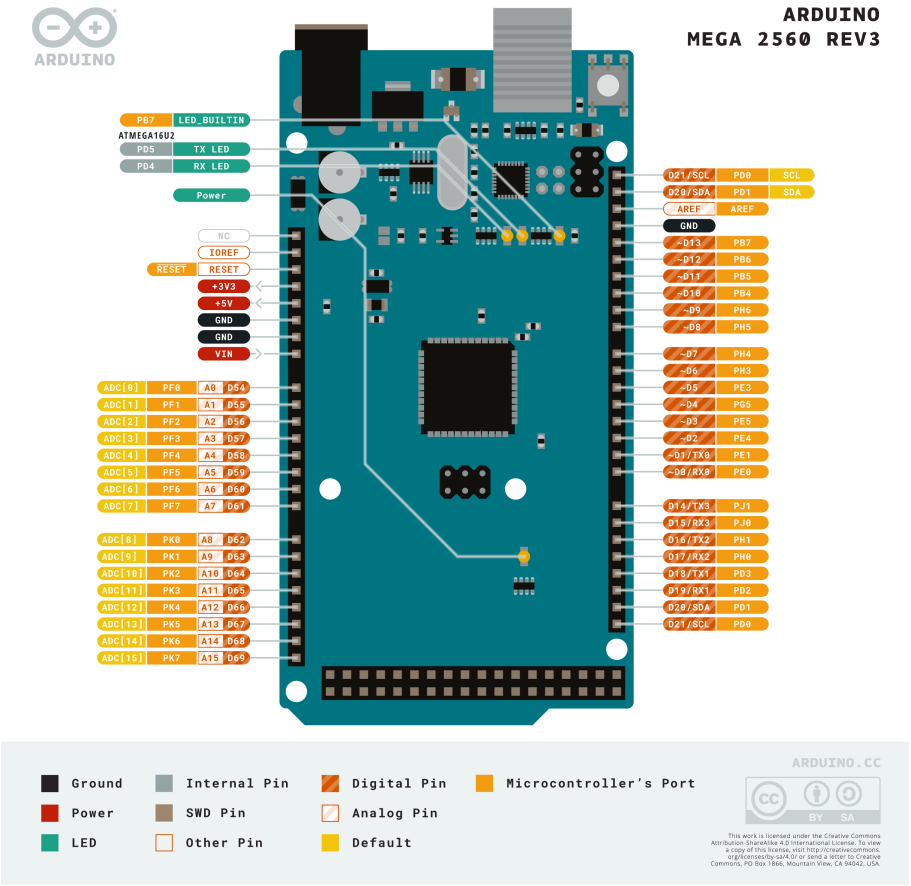


Figure 3.5: Ardunio Mega2560

Specifications

- Microcontroller: ATmega2560
- Operating voltage: $5V$
- Input voltage (recommended): $7 - 12V$
- Input voltage (limit): $6 - 20V$

- Digital I/O pins: 54 (of which 15 provide PWM output)
- Analog input pins: 16
- Direct Current per I/O pin: $20mA$
- Direct Current for $3.3V$ pin: $50mA$
- Flash memory: $256KB$, of which $8KB$ is used by bootloader
- SRAM: $8KB$
- EEPROM: $4KB$
- Clock speed: $16MHz$
- LED BUILTIN: Pin 13
- Length: $101.52mm$
- Width: $53.3mm$
- Weight: $37g$

3.1.4 RPLiDAR A1M8

Overview

The RPLIDAR A1M8 is a low cost 360 degree 2D laser scanner/LiDAR solution developed by SLAMTEC. The system can perform 360 degree scans within a 4 meter range. The produced 2D point cloud data can be used in mapping, localization and object/environment modeling. RPLIDAR A1's scanning frequency reaches 5.5 Hz when sampling 360 points each rotation, and can be configured up to 10 Hz at maximum. The RPLIDAR A1 is basically a laser triangulation range measurement system. It can work well in all kinds of indoor environments and in outdoor environments without sunlight.

The RPLIDAR A1 (shown in Figure 3.6) contains a range scanner system and a motor system. After powering on, the RPLIDAR A1 starts rotating and scanning clockwise. The user can get range scan data through the Serial port/USB communication interface. It comes with a speed detection and adaptive scanning system, adjusting the frequency of the laser scanner automatically according to motor speed.

Figure 3.6: RPLiDAR

Mechanism

The RPLIDAR is based on laser triangulation ranging principle and uses high-speed vision acquisition and processing hardware developed by SLAMTEC. The system measures distance data more than 2000 times per second and provides high resolution distance output ($<$1% of the distance, d in Figure 3.7).

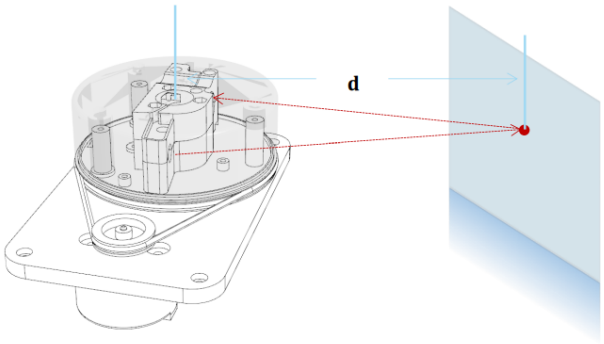


Figure 3.7: The RPLIDAR A1 Working Schematic

The RPLIDAR emits modulated infrared laser signal, which is then reflected by the object to be detected as shown in Figure 3.7. The returning signal is sampled by the vision acquisition system in the RPLIDAR A1. The DSP embedded in RPLIDAR A1 processes the sample data and outputs the range and heading to the object through the communication interface. A 3.3V-TTL serial port (UART) is used as the communication interface. The power interface of the RPLiDar is shown in Figure 3.8.

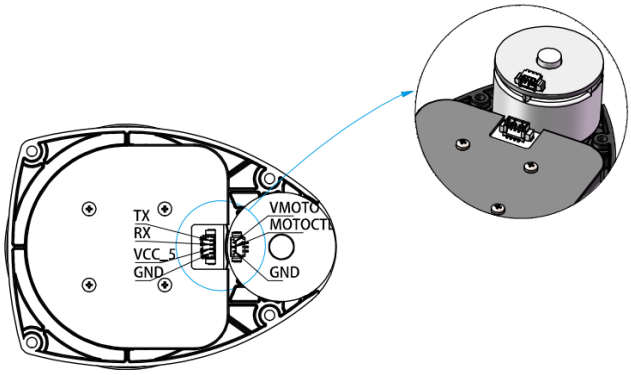


Figure 3.8: RPLIDAR A1 Power Interface

3.1.5 Stereo Camera

Overview

A stereo camera is a type of camera with two or more lenses with a separate image sensor or film frame for each lens. This allows the camera to simulate human binocular vision, and therefore gives it the ability to capture three-dimensional images, a process known as stereo photography. The distance between the lenses in a typical stereo camera (the intra-axial distance) is about the distance between one's eyes (known as the intra-ocular distance) and is about 6.35 cm, though a longer base line (greater inter-camera distance) produces more extreme 3-dimensionality.

Calibration

3.1.6 Ultrasonic Sensor

Overview

The ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact range measurement, with a ranging accuracy that can reach up to 3mm. The modules include ultrasonic transmitters, receivers and control circuits. The basic usage of the module is as follows.

- Use IO trigger for at least 10us with a high level signal.
- The module automatically sends eight 40 kHz pulses and detect whether there is a pulse signal back.
- If a signal is recieved back, time of high output IO duration is the time from sending the ultrasonic signal to returning.
- The Pinout diagram of the module is shown in Figure 3.9.
  (1) 5V Supply
  (2) Trigger Pulse Input
  (3) Echo Pulse Output
  (4) 0V Ground

Specification

- Working Voltage: DC 5 V
- Working Current & Frequency: 15mA @ 40Hz
- Range: 2cm - 4m
- MeasuringAngle: 15 degree
- Trigger Input Signal: 10uS TTL pulse
- Echo Output Signal: Input TTL lever signal and the range in proportion
- Dimension: 45*20*15mm

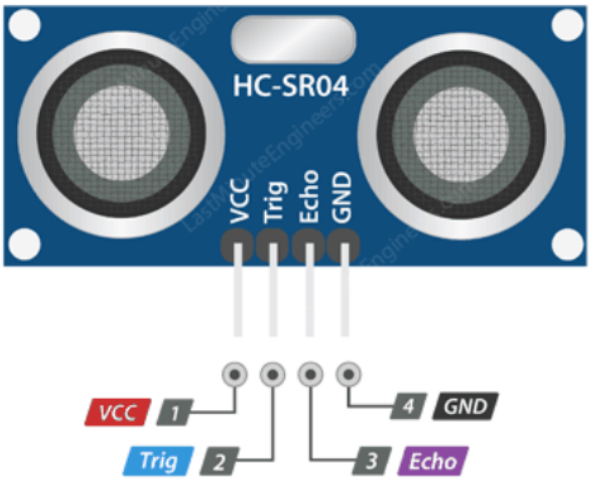


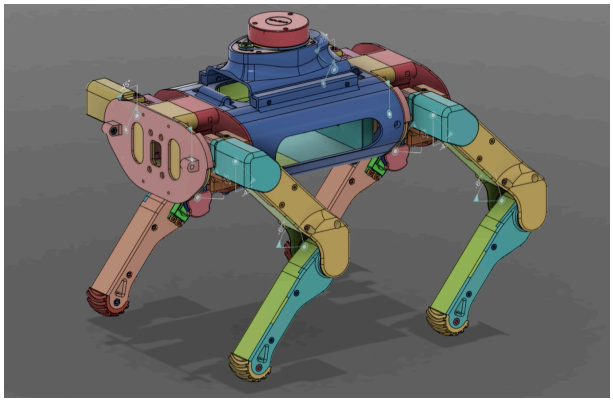
Figure 3.9: Ultrasonic Sensor Pinout diagram

3.1.7 SMPS

Overview

3.2 CAD Design and Modelling

This section discusses the design of the hardware of the robot. The platform used for the design of the robot is Autodesk Fusion360, which is free for personal use.



Chapter 4

Processes

4.1 Inverse Kinematics

Inverse Kinematics, abbreviated as *IK*, is a powerful method of control, with which the control problem of positioning the endpoint of a multiple DoF (Degree of Freedom) system can be solved. Here, the endpoint of the multiple DoF system is specified, and the control inputs for the system to achieve the endpoint are computed by the inverse kinematics algorithm.

This find wide application in systems like robotic arms, where the positioning of the arm endpoint can be specified, and joint angles can be computed for positioning the endpoint at the desired location. Similarly, it can be applied to the quadruped leg to compute the leg joint angles from a specified leg endpoint position.

In our quadruped robot, it is used to solve for the three joint angles in each of the four legs, making up a 12 DoF system. Here, we use a [*Hip, Shoulder, Elbow*] nomenclature, which, we admit, is not anatomically accurate. For us, however, it seemed intuitive to think of it in such a way, and thus will continue to use it through the derivation below. The angles are, namely:

θ_0 : hip joint angle
θ_1 : shoulder joint angle
θ_2 : elbow joint angle

Here, we derived our own inverse kinematics algorithm using trigonometric principles, a summary of which is given below.

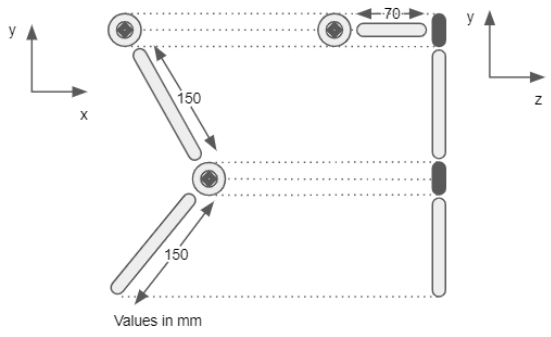


Figure 4.1: Leg views from front, side

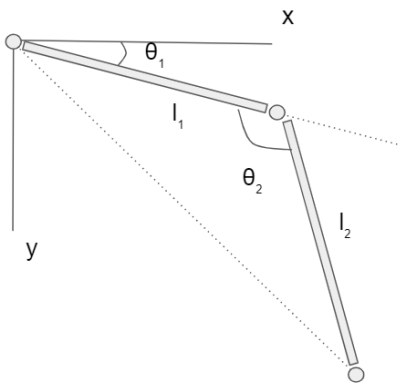


Figure 4.2: Annotated side view

The notations used in the figures are given below:

x = position of the leg endpoint along the x axis
y = position of the leg endpoint along the y axis
z = position of the leg endpoint along the z axis
l_0 = length of leg segment from hip to shoulder
l_1 = length of leg segment from shoulder to elbow
l_2 = length of leg segment from elbow to foot
l_3 = length of front projection from hip to foot
l_4 = length of front projection from shoulder to foot
r = projected leg length from shoulder to foot
θ_0 = hip joint angle
θ_1 = shoulder joint angle
θ_2 = elbow joint angle

For the purpose of deriving the relationships of these parameters to the joint angles θ_0, θ_1, and θ_2, we define

$\theta_{1\ effective}$ = effective angle between l_3 and the x axis

From Figure 4.2 and Figure, we get l_3 and l_4 as

$$l_3 = \sqrt{y^2 + z^2}$$

$$l_4 = \sqrt{l_3^2 + l_0^2}$$

Now, r can be found to be

$$r = \sqrt{x^2 + l_4^2}$$

Then, we can compute the hip angle θ_0 as

$$\theta_0 = \arccos\left(\frac{zl_0 + yl_4}{l_3^2}\right)$$

Solving for the knee angle, θ_2

$$\cos\theta_2 = \frac{l_1^2 + l_2^2 - r^2}{2l_1l_2}$$

Thus, the knee angle, θ_2 is given by

$$\theta_2 = \arccos\left(\frac{l_1^2 + l_2^2 - r^2}{2l_1l_2}\right)$$

Now, we need to solve for the shoulder angle θ_1. For this,

$$\theta_{1\ effective} = \arctan\left(\frac{l_4^2}{x^2}\right)$$

also,

$$\cos\bar{\theta}_2 = -\cos\theta_2$$

$$\sin\bar{\theta}_2 = \sqrt{1-\cos\bar{\theta}_2}$$

Furthermore, we can write

$$\bar{\theta_1} = \arctan\frac{l_2\sin\bar{\theta}_2}{l_1 + l_2\cos\bar{\theta}_2}$$

Finally, we get

$$\theta_1 = \theta_{1\ effective} - \bar{\theta}_1$$

Thus, we have computed the hip, shoulder and elbow angles for one leg of the robot. The results can now be applied to the servo angles after accounting for their offsets and characteristics, which is done with the data obtained from calibrating the servos in each joint.

4.2 Visual Odometry

Visual Odometry is the process of finding the position and orientation of a robot by analyzing the changes the motion induces on the images of the onboard cameras. It thus intends to find the change in position and rotation of a robot from the $(k)^{th}$ to the $(k+1)^{th}$ image. The result of the visual odometry algorithm is thus a composite translation and rotation matrix, denoted as RT, which can be decomposed to their constituent translation T and rotation R matrices, and integrated over several such image pairs to produce an estimate of the state of the robot.

The main advantages of Visual odometry over conventional methods like wheel odometry are :

- Not affected by wheel slip in uneven terrain, rainy/snowy weather or other adverse weather conditions.
- More accurate trajectory estimates are provided.

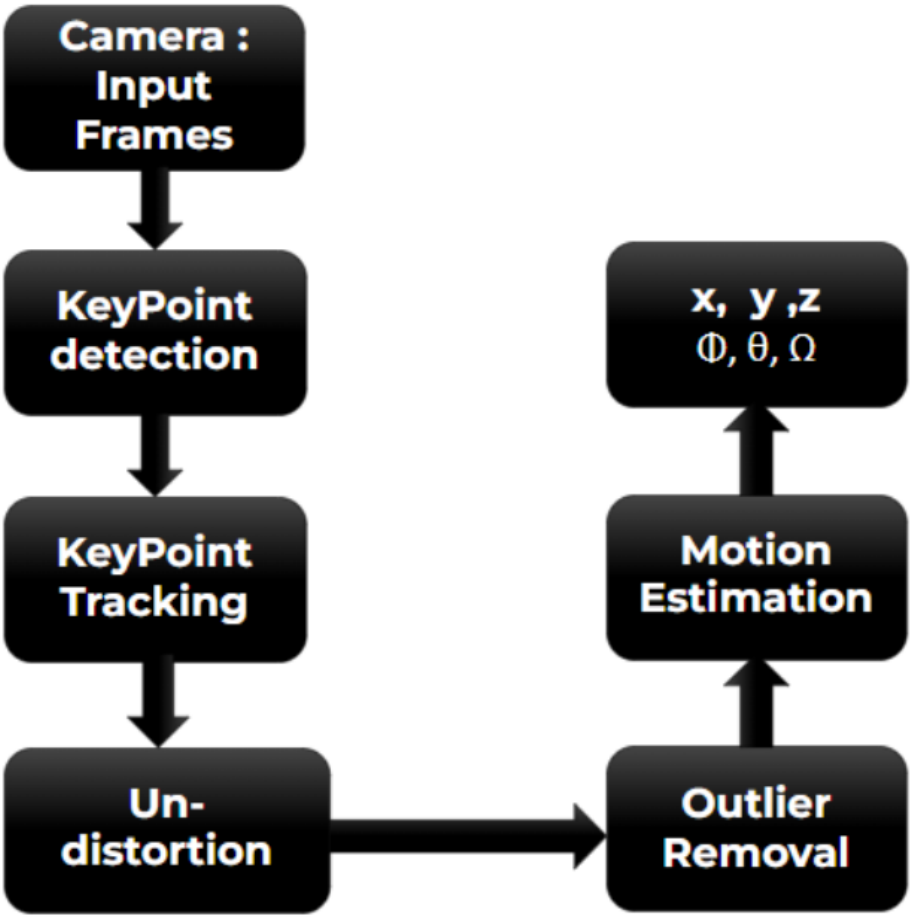


Figure 4.3: Flowchart

Algorithm 1 Visual odometry algorithm

Require: *img* ← *Image data from stereo camera*

1: *Apply KeyPoint detection algorithm to detection*
2: *Apply KeyPoint tracking algorithm*
3: *Undistortion is performed on the image*
4: *Perform outlier removal*
5: **return** *Pose*

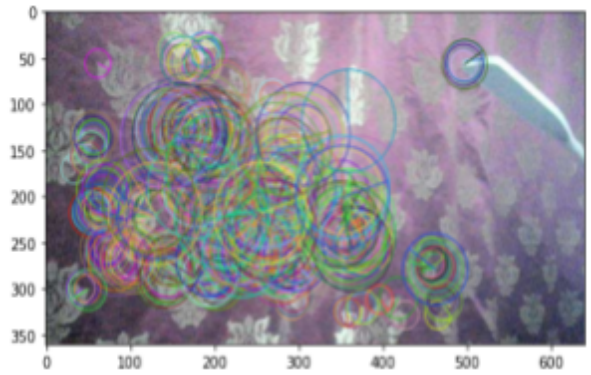


Figure 4.4: Feature Detection

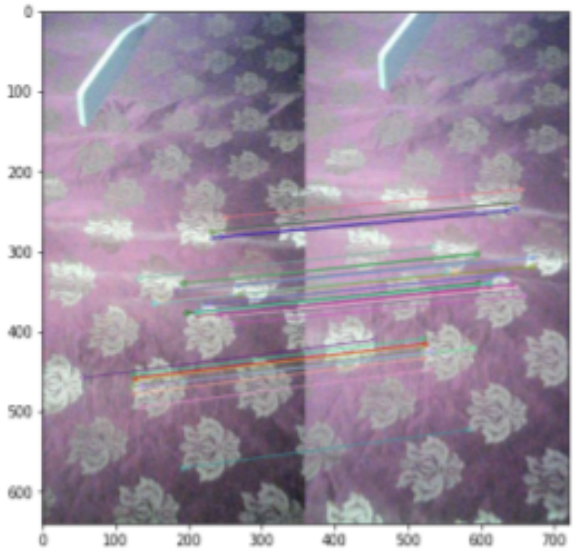


Figure 4.5: Feature Matching

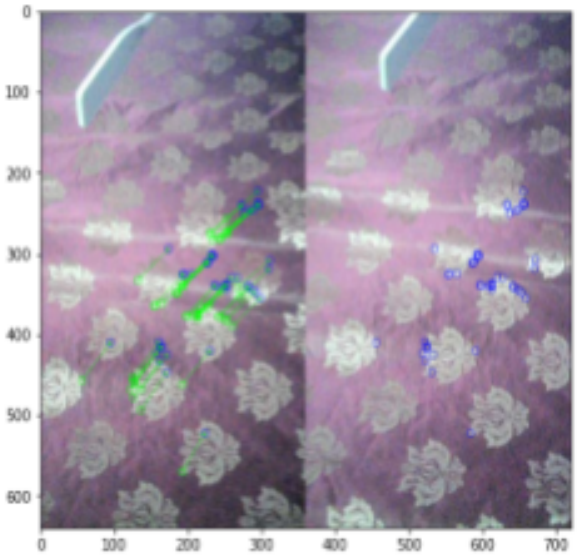


Figure 4.6: Motion Estimation

4.3 Stair Detection

In real-world application, the staircase is a common terrain occurrence. The robot is expected to navigate this terrain autonomously and with ease, thus requiring the implementation of stair detection. The stair detection algorithm takes images of stairs as primary input from the stereo camera setup. With the help of OpenCV fused with Hough Lines algorithm, the raw image data is further processed and refined to produce information about the stair contours, number of steps and step height.

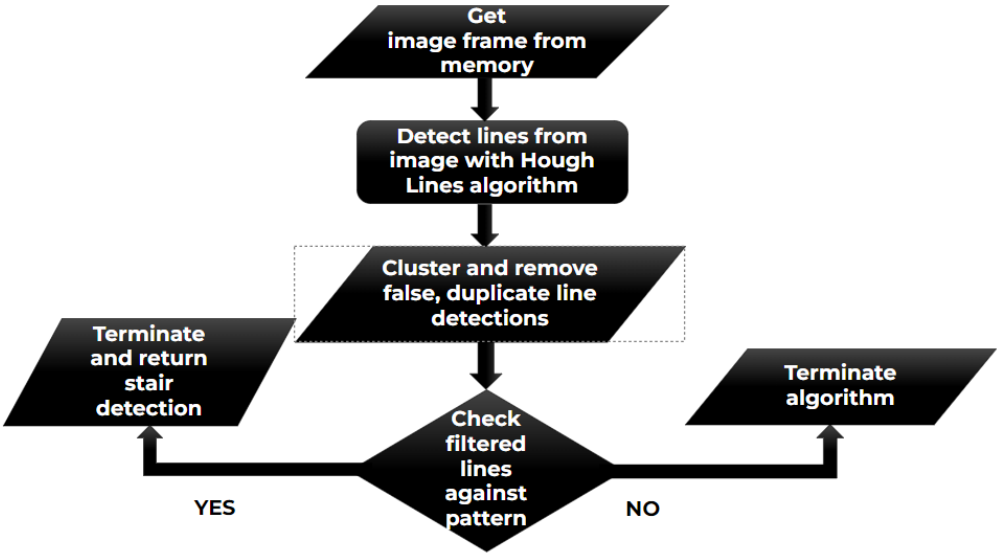


Figure 4.7: Flowchart

Algorithm 2 Stair detection algorithm

Require: $img \leftarrow Image\ data\ from\ stereo\ camera$

Binarize img to reduce noise
Apply Canny Edge filter to img
Apply Hough Lines filtering algorithm to img
if $0.1 < \theta < 0.8$ **then**
    $\theta = Stair\ side\ angle : Left$
    $angleL \leftarrow \theta$
else if $2.4 < \theta < 3$ **then**
    $\theta = Stair\ side\ angle : Right$
    $angleR \leftarrow \theta$
end if
Print angleL, angleR
if $91 \times \frac{\pi}{180} < \theta < 90 \times \frac{\pi}{180}$ **then**
    $\theta = Stair\ angle : Vertical$
    $angleV \leftarrow \theta$
end if
Overlay lines on img
return *img*

Figure 4.8: Test image

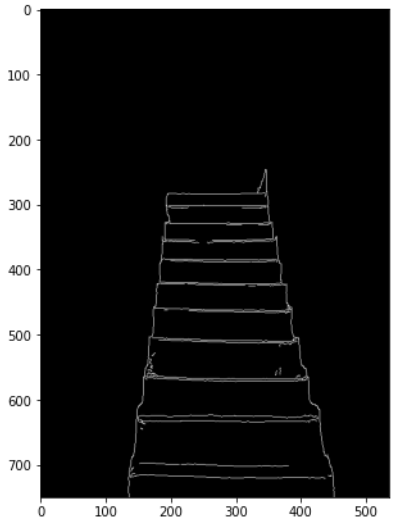


Figure 4.9: Processed image

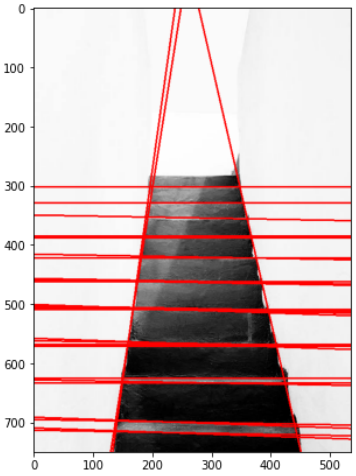


Figure 4.10: Applying Hough Lines algorithm

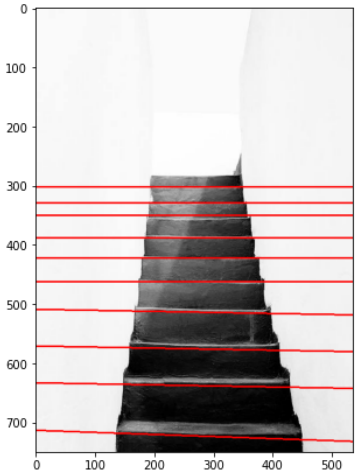


Figure 4.11: Flitered final image

4.4 Path Planning

A very cruicial part of autonomous navigation is *path planning*. The robot should be able to successfuly avoid obstacles in its path and choose the shortest path possible to reach its destination to maximize efficiency. For this purpose we need an intelligent and efficient algorithm to enable successful traversal of complex terrain.

In engineering, there are two approaches to solving this problem: mathematical and heuristic approaches. The mathematical approach is more concerned with the solution than with ensuring that the computations are feasible for real-time algorithms. The latter use multiple ways to analyse the overall problem from start to finish. Examples of this apporach include the Euclidean approach to look for geometric patterns and "the look ahead approach". These algorithms, however, don't find the minimum optimal distance. In the heuristic approach, the algorithm uses special knowledge of the problem space. After exploring and studying several algorithms written for path planning, the A* path planning algorithm was chosen for implementing in our robot. The A* algorithm, when used with a proper heuristic for the distance to the destination can generate an optimal path in a graph efficiently. It is the most efficient free-space searching algorithm for path planning and obstacle avoidance, and uses a combination of heuristic searching, and searching based on the shortest path.

The A* algorithm is classified as a *best-first* algorithm, because each cell in the problem space is evaluated according to the cost of traversal, which is based upon the overall distance of the path. The main benefit of the A* algorithm is its ability to find an optimal solution in a reasonable amount of time, versus the uninformed search of looking for a path. One of the disadvantages of the algorithm is the inability to react to unexpected added or moving obstacles in the testing area. The A* algorithm cannot adjust the list made for creating a path to take or the new bounds not known. The default is there is no path found.

In this algorithm, the whole space is divided into cells or nodes. The starting node and the end node are to be defined initially. The path is found with respect to the distance between these two nodes. The neighbouring nodes of the starting node are first taken into consideration. There are two costs involved in the algorithm, the $G-cost$ and the $H-cost$. The $G-cost$ is the distance between the initial node and the current node, while the heuristic cost or $H-cost$ is the distance between the current node and the final node, which is an assumption based on distance formulas like the manhattan distance that we have used here. An overall cost called the $F-cost$ is found for the distance evaluation.

The equation is given below.

$$F-cost = G-cost + H-cost$$

An open list keeps track of the nodes that need to be explored and begins with the start node. This open list, often a *priority queue* data structure is used to keep track of all upcoming nodes. When a path runs out of scope or is not possible to traverse any longer, another nearest neighbour node is taken from the open list.

$$Open = \{(F-cost,\ Node1), (F-cost2,\ Node2), (F-cost2,\ Node2)....\}$$

This open list is continually updated during the operation of the algorithm.

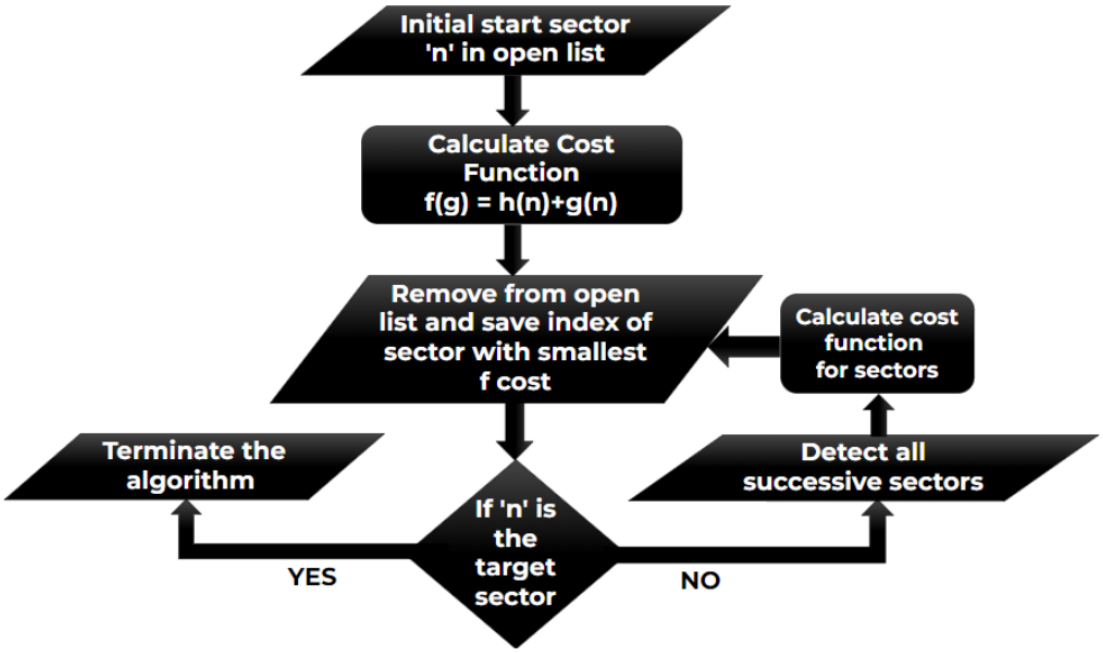


Figure 4.12: Flowchart

Algorithm 3 A* Path Planning

Require: *ogrid* ← *Occupancy grid data from RPLiDAR*

1: *Define the start and end nodes*
2: *First consider unoccupied neighbouring nodes of start node*
3: *Evaluate G − cost and H − cost*
4: *Add node with lowest F − cost to open set*
5: *Consider neighboring nodes with lowest F − cost and repeat from step* 3
6: *Terminate the algorithm once H − cost is* 0
7: *Trace back last nodes to find optimum path*

For example, consider the graph shown in Figure 4.13 with nodes A, B, C and D interconnected with weighted edges, i.e., multiple distance values are associated with the graph.

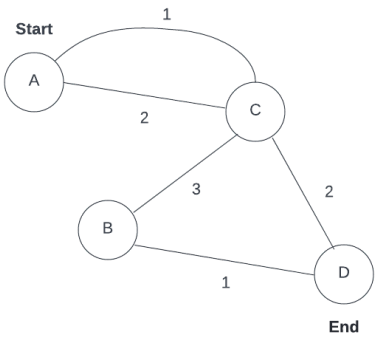


Figure 4.13: Weighted graph with nodes A, B, C and D

Here the Start node is A and the End node is D. The distance assciated with multiple paths are shown in figure. ie There are 2 ways from A to C, with distance values 1 and 2. The lowest weighted edge is chosen between these two. In this case, the shortest distance between them is 1 and that path is chosen. Therefor, on reaching C, the G-cost becomes 1 and so on. In our implementation, these nodes are considered as the cells of the occupancy grid and the weights for the edges would initially be set by default as 1, because all the adjacent cells are in equal distances essentially. i.e. If the robot has to cross 2 cells to reach the destination, the distance or the cost would be 2. The intention of the algorithm would be to find the optimum path to reach the destination, here the optimum F-cost would 3 when the path A-C-D is traversed.

To start off with the algorithm we have to insert the start node, along with the F-cost (initially 0), into the open set, represented by a priority queue.

$$Open = \{(F-cost,\ Node)\} = \{(0,\ A)\}$$

The inital Costs of the respective nodes are represented in the table.Since we dont consider the start node as a neighbouring node all the values are set defuult to 0. while the other nodes B, C and D have a possibility of being considered as the Neighbouring node and thus the values are set as default to ∞.

Nodes	F-cost	G-cost	H-cost
A	0	0	0
B	∞	∞	∞
C	∞	∞	∞
D	∞	∞	∞

Table 4.1: Nodes and their costs

Now the first neighbouring node C is considered and the lowes G cost available to it is 1 and using the manhattan distance formula the h-cost is assumed to be 1 although the value might be incorrect. Then the overall f cost is 2. and the last node we came from is A.

Nodes	F-cost	G-cost	H-cost	Last Node
A	0	0	0	
B	∞	∞	∞	
C	2	1	1	A
D	∞	∞	∞	

Table 4.2: Nodes and their costs

Now this node aloong with its F-cost is added to the open set of neighbouring nodes.

$$Open = \{(2,\ C)\}$$

In the open set there is only one node so there is no comparisson to make with respect to the shortest F-cost.Now we consider the Node C and its neighbours, D and B. In the case of B the G-cost is 4 and H-cost is assumed, while in the case of D, the G-cost is 3. The table is updated as shown in Table 4.3.

Nodes	F-cost	G-cost	H-cost	Last Node
A	0	0	0	
B	6	4	2	C
C	2	1	1	A
D	3	3	0	C

Table 4.3: Nodes and their costs

Therefore the node with the lower F-cost, D, is added to the open set and the node B is ignored.The value of H-cost is 0 for node D, since it is the end node.

$$Open = \{(2,\ C)\}, \{(3,\ D)\}$$

While comparing the F-costs of the nodes in the open set, we find the End node and its taken out of the open set to complete the algorithm.

Now all we have to do is backtrack all the last nodes. i.e. D from C, C from A, and thus we get the optimum path, A-C-D.

For the purpose of visualization additional python packages were used. *PyGame* is a 2-D graphics module for python, which is easy to use and was used to implement the visualization tool effectively.

The occupancy grid is generated from the data produced by the RPLiDAR unit. For purposes of visualization, we set the Start Node (in Orange), End node (in Turquoise), and Occupied Nodes (in Black), with the unoccupied Nodes represented in white. The nodes already visited and closed off as non traversable are removed from the open set as they are not considered further and are represented in red. The reconstructed path is represented in purple, and is the optimum path for the robot to reach its destination.

The input and output images of the Pygame window are given in Figure 4.14 and Figure 4.15.

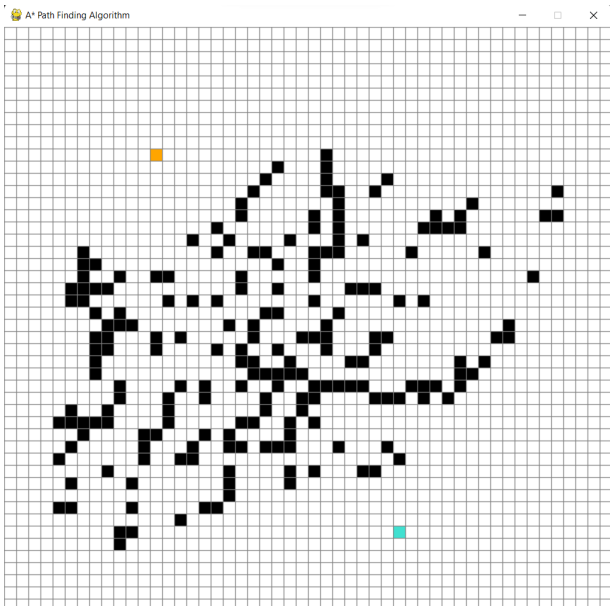


Figure 4.14: Occupancy Grid

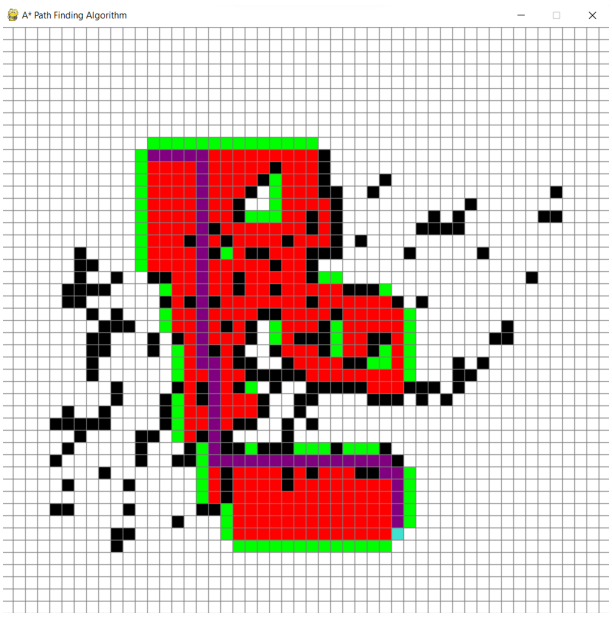
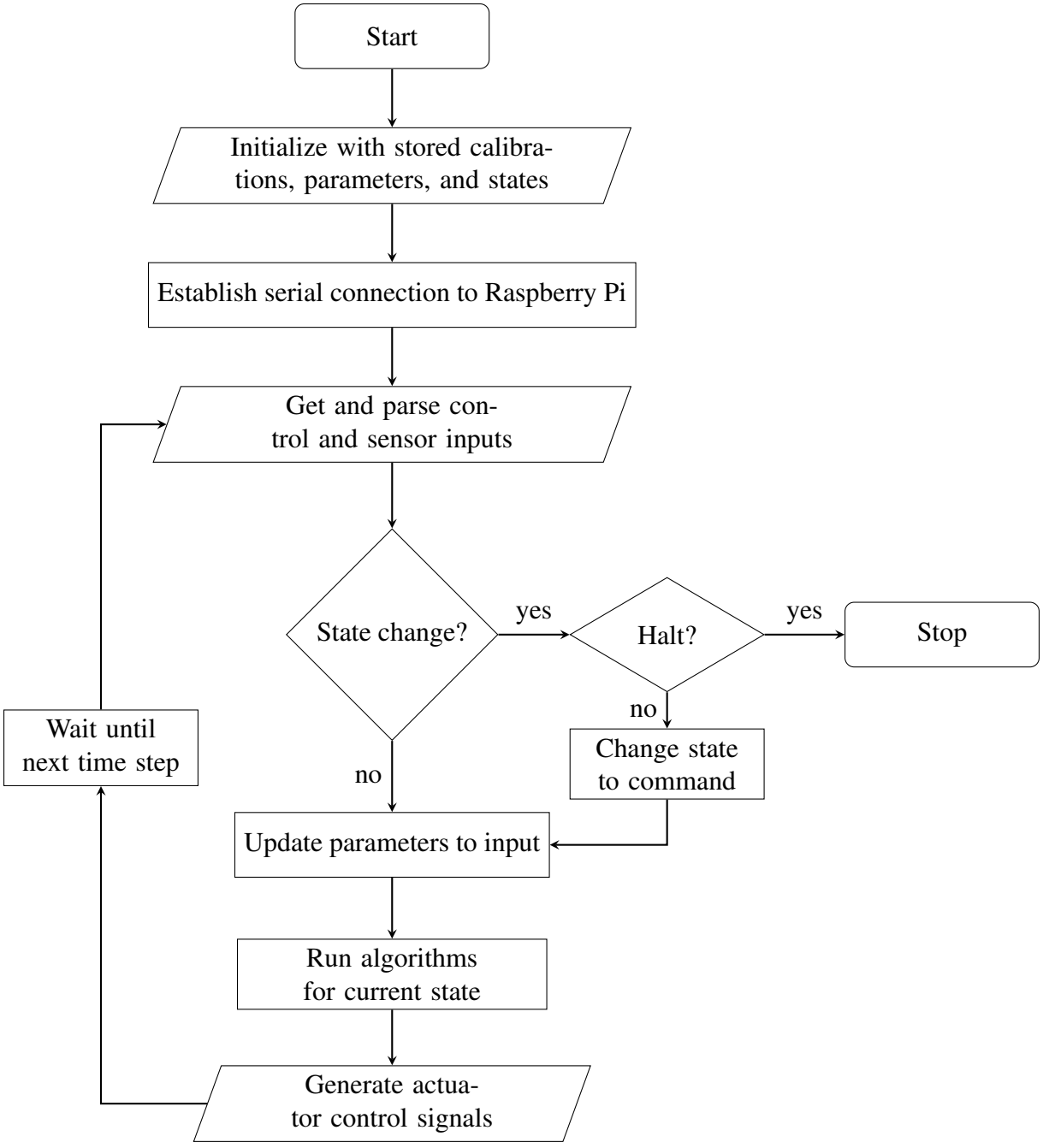


Figure 4.15: Reconstructed Path

4.5 Implementation on Arduino

The processes mentioned in the preceding sections are implemented on either the Raspberry Pi, the robot's higher level computer, or on the Arduino Mega, it's lower level computer. The higher level computer is well suited to soft real-time tasks and tasks that can be done offline. This includes processes like LiDAR scanning and mapping, stereo vision and depth mapping, visual odometry, and other compute and memory intensive tasks that cannot be done on the Aruduino Mega. The lower level computer is well suited to real-time control and sensing needs, and directly controls the actuators and some sensors on board the robot.

Here, we discuss the current implementation of the control loop that runs on the Arduino Mega controller.



Chapter 5

Results and Discussion

Chapter 6

Future Scope

Chapter 7

Conclusion

Designed and developed a fully functional Quadruped Robot which can map out the environment and perform a range of tasks.

References