ABSTRACT

This work aims to design and develop a Quadruped Autonomous Robot. A quadruped robot is a four-legged robot that can mimic an animal walking gait so that it can handle irregular terrain with ease. They are designed for locomotion on rough surfaces and require control of leg actuators to maintain balance as well as movement, sensors to govern where it can wander and algorithms to determine the direction of the movement. The basic idea of the project is that the quadruped will detect the obstacle or the object as it moves around and bypasses that obstacle. This can be achieved by using a Pi cam (5MP) that can shoot 1080p at 30 fps. Here Raspberry Pi is used along with Pi cam that fits best for computer vision. An Ultrasonic sensor is also used for avoiding the obstacle. The locomotion of the quadruped robot is obtained by controlling its legs with the help of a servo motor using Raspberry Pi. The detection process followed by live streaming can be done using a media player and sensor data visualization can be done using an MQTT platform.

The quadruped robot functions in two modes. They are auto mode and manual mode. In auto mode, the quadruped robot roams around by itself and avoids obstacles, whereas in manual mode the user controls the movement of the robot using a joystick controller. The robot will have forward, backward, turn left, and turn right movements. Each leg of the robot has three servo motors. Hence a quadruped robot requires a total of 12 servo motors for its effective locomotion. These provide precise motion and stability to the quadruped robot. Each leg consists of 3 servo motors, thus providing 3 ligaments like joints that provide multi-directional motion like that of a spider.

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

ARM	Advanced RISC Machines
DHT	Digital Humidity and Temperature
FPS	Frames Per Second
GHz	Gigahertz
GPIO	General Purpose Input Output
IoT	Internet of Things
MQTT	Message Queue TelemetryTransport
OpenCV	Open Computer Vision
PWM	Pulse Width Modulation
SoC	System on Chip
Wi-Fi	Wireless Fidelity

CHAPTER 1

INTRODUCTION

1.1 Quadruped Autonomous Robot

There are several scenarios where wheeled robots cannot travel. In this case, instead of a wheeled robot, legged robots can be used. Legged robots are more advantageous and versatile than wheeled robots on uneven terrain such as military operations, remote locations, dangerous environments, excavation, construction works etc. A quadruped robot is a four-legged robot that can mimic an animal walking gait so that it can handle irregular terrain with ease. They are designed for locomotion on rough surfaces and require control of their leg actuators to maintain balance as well as the movement of the robot, sensor to govern where it can wander and algorithm to detect the obstacle and determine the movement of the robot. Each leg of the robot consists of three ligaments like joints that provide multi-directional motion like that of a spider which provides precise motion and stability to the quadruped robot. This uses a remote controller for controlling the movement of the robot.

1.2 Classification and related constraints:

This project is about a quadruped robot that can be controlled manually and work automatically by choosing the path to travel, detecting, and avoiding obstacles without human intervention. It has some additional features like live video streaming, temperature, and humidity data monitoring support. All these features make it an automatic and surveillance robot.

CHAPTER 2

LITERATURE SURVEY

2.1 General Introduction

Literature Survey is an important activity that must be conducted for the implementation of the project. Since this project mainly deals with a legged robot that can detect and avoid obstacles, the literature survey carried out consists of papers related to legged and obstacle avoiding robots. Here the papers give us an insight into the design, locomotion, implementation of those robots. The papers that are referred for the implementation are described below.

2.2 Literature Survey

M. Nandhini, et.al [1] proposed a paper related to quadruped robots. It gives an in-depth view of technical and programming aspects in designing a walking robot. In this paper, they discussed the design and implementation of a four-legged robot that can mimic an animal walking pattern. The main objective was to design a robot that can walk or run depending upon the speed of the motor. The quadruped robot walking gaits are developed manually in c language and compiled in Arduino IDE. The parts of the legs are known as ‘coxa’, ‘femur’ and ‘tibia’.

Chinmayi R, et.al [2] proposed a wheeled robot that can detect and avoid an obstacle as the robot moves. Here they explained about designing and implementing a robot that detects obstacles and senses the right path as it moves. This robot was driven with an Arduino board controlled by an ultrasonic sensor. Here the obstacle is being detected at a distance of 15cm. When the robot senses the obstacle it stops, the ultrasonic sensor takes readings at 0 and 180° with the help of a servomotor, robot turns to the direction where the distance is maximum. This robot could also be controlled manually by establishing a connection between Arduino and a Bluetooth module via an android app. By assigning controls in a smartphone according to the program, the robot could move forward, backward, right, left, stop etc.

Aditya Singh Rathore [3] proposed an approach to obstacle detection for an autonomous vehicle using the OpenCV library. The project involves obstacles in an image using Python and OpenCV. It contains an approach towards algorithm development for obstacle detection and avoidance. Also, the reason and procedure for choosing grayscale instead of colour, detecting edges in an image, selecting region of interest, applying various image processing techniques has been discussed here.

Firas A.Raheem et.al [4] proposed techniques for Quadruped Robot creeping gait stability analysis and optimization using PSO. It contains an approach towards algorithm development for a better manual control system for the robot and optimizing the same for better performance.

Ravi Kishore Kodali et.al [5] proposed an approach for low-cost implementation of MQTT using Raspberry Pi. Here, the use of the MQTT platform, features and services was highlighted. The implementation of this protocol was used for remote sensing and updating the sensor values on the go in the MQTT platform’s dashboard with various types of data visualization.

CHAPTER 3

PROBLEM STATEMENT AND OBJECTIVES

3.1 Problem Statement

Currently, there are several scenarios like outdoor terrain on which wheeled vehicles/robots cannot travel safely. Since the wheeled robot has mechanical parts that are majorly compatible with even surfaces, so there is a huge chance of the robot getting damaged while travelling on rough or rocky surfaces. And conventional vehicles lack automatic obstacle and path detection systems that can avoid accidents or damages while moving in uneven terrain.

When the robot is controlled manually, establishing manual control over the robot becomes complicated. Because when the robot is far away from the user, the user does not have clear cut information about the vicinity of the robot. Since the legged robot has numerous movable joints it becomes even harder and accurate controlling becomes impossible.

3.2 Objectives

Based on the idea of the project as well as the problems that are being identified in the conventional system, the objectives of the project are listed as given below:

- Develop an algorithm to control the locomotion of the quadruped robot.
- Develop an algorithm to detect and determine the characteristics obstacles and to overcome them.
- Develop an algorithm to manually control the robot using a wireless controller.
- Implement real-time video streaming to provide the robot’s point of view to the user.
- Implement a real-time sensor data monitoring system.

CHAPTER 4

PROJECT OVERVIEW AND METHODOLOGY

4.1 Block Diagram

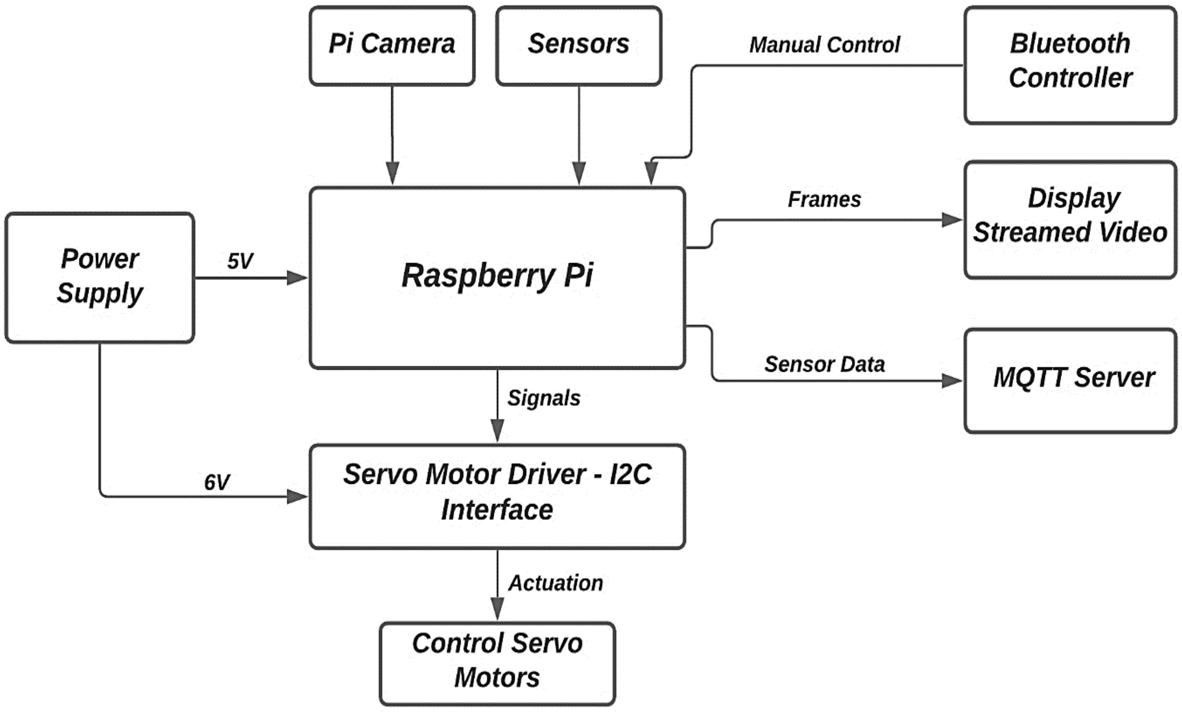


Fig 4.1 Proposed Block Diagram

4.2 Hardware Specifications

4.2.1 Raspberry Pi

The Raspberry Pi 4 is integrated with Broadcom BCM2711 Quad-core Cortex-A72 (ARM v8) 64-bit SoC clocked at 1.5GHz, 2GB LPDDR4-3200MHz SDRAM. It features a true gigabit ethernet port, 2 x USB 3.0 ports. Supports both (2.5 GHz & 5GHz) Wi-Fi standards, Bluetooth 5.0, Raspberry Pi standard 40 pin GPIO header, 2 x micro-HDMI ports (up to 4k 60fps supported). 5V-3A USB-C power supply is required for its working.

4.2.2 Raspberry Pi camera module

5MP native resolution sensor-capable of 2592 x 1944 pixel static images. Supports 1080p30, 720p60 and 640x480p60/90fps video. The camera is supported in the latest version of Raspbian (Raspberry Pi OS).

4.2.3 Li-ion batteries

ICR18650 3.7V Rechargeable 1200mAh Li-ion Battery or ICR18650 3.7v 2200mAh Li-ion Battery.

4.2.4 Servo Motors

- SG90 (180 Degree) Servo Motor
- Operating voltage: 3-7V.
- 1.5 kg-cm torque at 6v and speed up to 100RPM.

4.2.5 Servo Motor Driver

16 channel 12-bit PWM servo motor driver. It is 5V compliant but can be controlled from a 3.3V Raspberry Pi and still safely drive 6V outputs. This can drive up to 16 servos or PWM outputs over I2C using only 2 pins (SDA and SCL).

4.2.6 LM 2596 DC-DC buck converter

Step down buck converter used to convert voltage from battery to required voltage. Two buck converters are used in the project, one to power Raspberry Pi at 5.1V and the other to drive the servo motor driver at 6V.

4.2.7 Sensors

- **HCSR04 Ultrasonic sensor:** Ultrasonic sensor with an operating distance range of 2cm-400cm
- **DHT11 Temperature and Humidity sensor:** The temperature sensing range is from 0 to 50 degrees Celsius and humidity ranging from 20%-90% can be sensed.

4.2.8 Robot chassis and legs

Quadruped Spider Robot 3D Printed Parts,

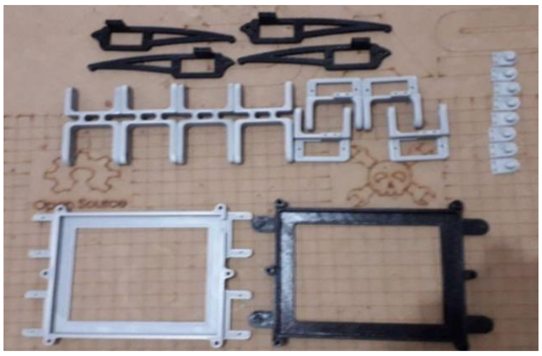


Fig 4.2 3D-printed robot parts and finished robot leg

4.3 Software Platforms

4.3.1 PyCharm IDE for Raspberry Pi programming

Dedicated interactive Integrated Development Environment (IDE) providing a wide range of essential tools for Python that supports most python modules.

4.3.2 OpenCV using Python

OpenCV is the huge open-source library for computer vision, machine learning, and image processing. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human.

4.3.3 Adafruit IO (for MQTT service)

MQTT library or client to which sensor data can be published and subscribed to and from a feed.

4.3.4 VLC Media Player

It is a pre-installed media player which can be used to stream videos to a mobile phone or laptop via Wi-Fi.

CHAPTER 5

IMPLEMENTATION AND METHODOLOGY

5.1 Hardware Implementation and Connections

In the hardware implementation part, the interface among different components and communication among each module is discussed in detail.

5.1.1 3D Parts Assembly

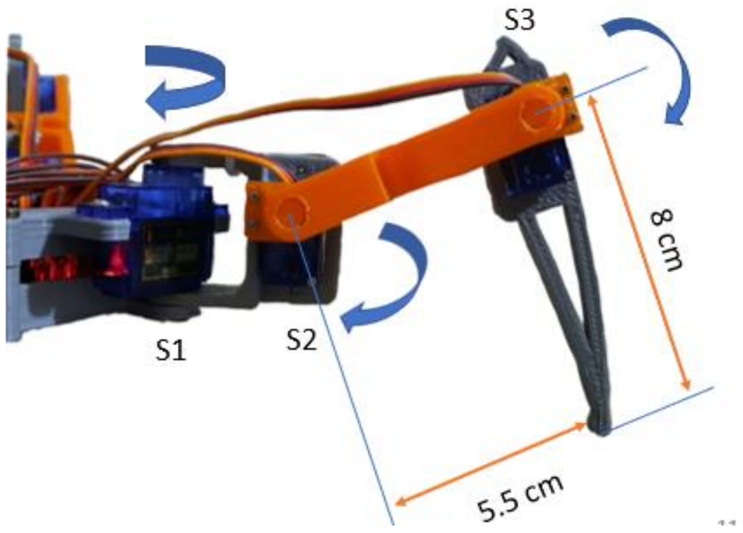


Fig 5.1 Assembled 3D printed leg

3D printed robot parts are assembled as given in the above figure. A single robot leg consists of three parts,

- **Coxa**: ‘Hip’ of the robot controlled by servo motor S1, it provides horizontal movement to the leg.
- **Femur**: ‘leg’ of the robot controlled by servo motor S2, it provides vertical movement to the leg.
- **Tibia**: ‘foot tip’ of the robot controlled by servo motor S3, it provides horizontal movement to the leg and contact to the ground.

Repeating the same procedures for the rest of the three legs and they are connected by the robot chassis.

5.1.2 Pi Camera Installation

In this project, we are using a Pi camera to capture images. It should be interfaced with the Raspberry Pi 4. So, the steps include,

- Install the Raspberry Pi camera module by inserting the cable into the Raspberry Pi. The cable fits into the slot situated between the Ethernet and HDMI ports, with the silver connectors facing the HDMI port as shown in the image below.

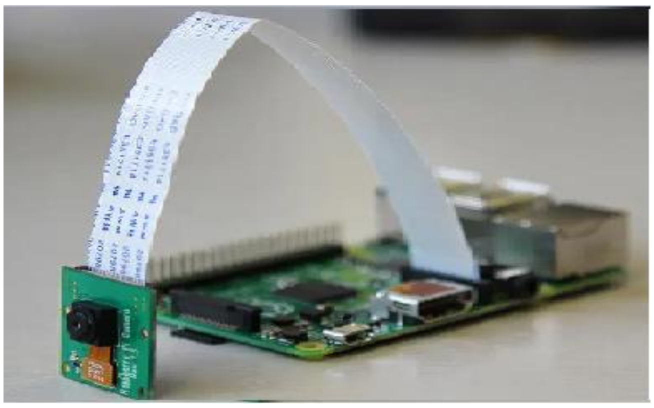


Fig 5.2 Pi cam interface

- Boot up the Raspberry Pi from the terminal, run *"sudo raspi-config"*. Navigate to the "camera" option and enable it. Select "Finish" and reboot the Raspberry Pi.

5.1.3 Servo Motor Driver Connection

Boot up the Raspberry Pi from the terminal, run *"sudo raspi-config"*. Navigate to the "I2C" option and enable it. Select "Finish" and reboot the Raspberry Pi. Now Raspberry Pi can interact with servo motor drivers via SDA and SCL pins. 12-bit 16-channel servo motor driver is powered by battery and buck converter at 6V.

PCA9685 pins to Raspberry Pi:

- SDA to GPIO 2
- SCL to GPIO 3
- VCC and GND pins of the servo driver is connected to 5v and ground pins of Raspberry Pi.

There are 4 legs with 3 servo motors each, hence 4*3=12 servo motors are connected to respective pins of the servo driver as given below.

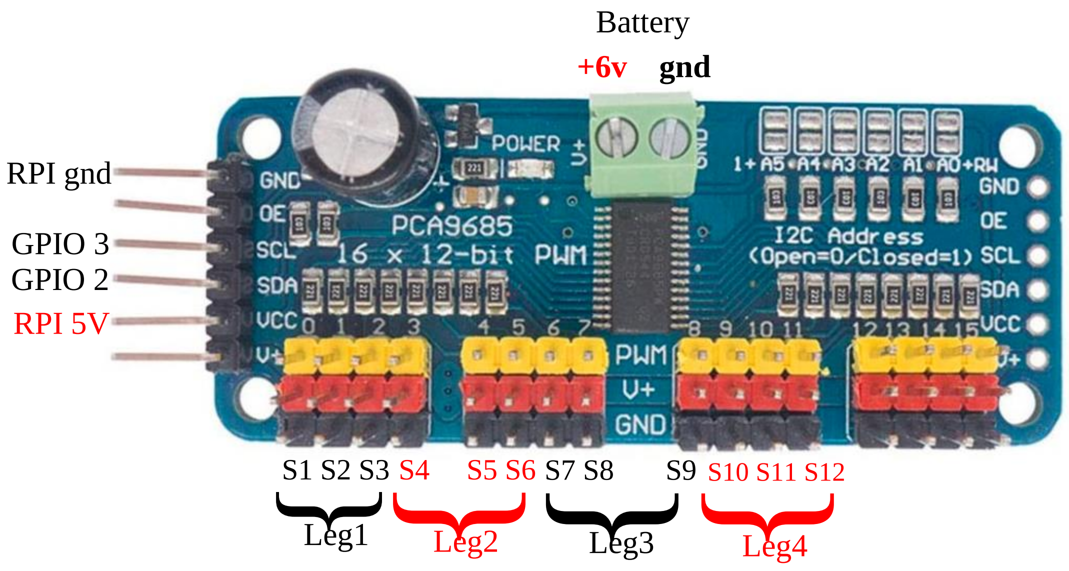


Fig 5.3 PCA9685 connections

5.2 Python Libraries

5.2.1 OpenCV (CV2)

OpenCV (Open-Source Computer Vision) is a library with functions that mainly aiming real-time computer vision. OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

5.2.2 Numpy

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices)

5.2.3 RPi.GPIO

A powerful feature of the Raspberry Pi is the row of GPIO (general-purpose input/output) pins along the top edge of the board. A 40-pin GPIO header is found on all current Raspberry Pi boards. This package provides a class to control the GPIO on a Raspberry Pi. The RPi. GPIO Python library (included with Raspbian) lets you configure, read, and write to GPIO pins.

5.2.4 Time (time)

Time library has specific functions for operations involving. It can be used to create a delay in the execution of the program. The sleep function can also do delays less than a second.

5.2.5 Threading (threading)

In python, the threading module provides a very simple and intuitive API for spawning multiple threads in a program. It can be used to link and combine various parts of the program and run all of them at once.

5.2.6 Adafruit_PCA9685

Servo motors are often driven using the PWM outputs available on most embedded MCUs. It is easy to use the PCA9685 board with Python and the Adafruit_PCA9685 module. This module allows you to easily write Python code that controls servos and PWM with this breakout.

5.3 Locomotion of Quadruped Robot

The way in which an animal or a robot walks is called a gait – it tells us how and in what order the legs are moved, how the body is balanced and how it moves forward. When the robot walks, it must keep its balance. There are two general strategies for doing that, and according to them, we divide the gaits into statically stable and dynamically stable.

For statically stable gaits it does not matter how fast they are performed, or whether the robot is stopped in a middle of a step – it is stable at any moment, at all times. Animals and people use those gaits when they want to go slowly, or when they want to be able to stop at any time. An example of such a gait is the "creep" gait, used by cats stalking their prey.

Dynamically stable gaits are much harder, as they have to be performed at a particular speed to mimic 'running' and cannot be interrupted at an arbitrary point. They are sometimes called "controlled falling", as they exploit the fact that it takes some time for the robot to fall when it's unstable, and that time can be used to move the legs in such a way as to prevent the fall. Most animal gaits are dynamically stable, as they tend to be faster and more energy-efficient. An example of a simple dynamically stable gait is the "trot" gait.

5.3.1 Creep gait

Since the robot is to be used for surveillance in which the robot needs to be quiet and stable at all points of time, a creeping gait is implemented.

5.3.1.1 Change in area of support

The area of support is the area where the mass of the robot is acting on the ground. The robot needs to raise one of its legs to move forward. Once the leg is raised, the area of support drastically changes as given in the below figure b which may cause the robot to tip over to the edge of its raised leg.

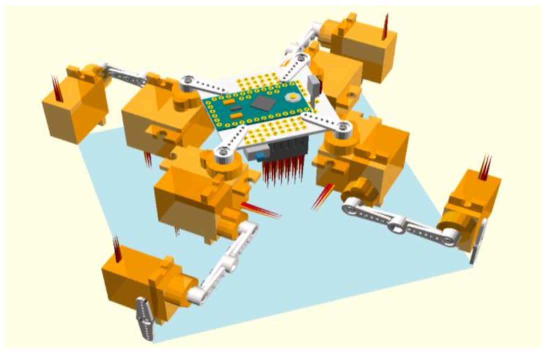
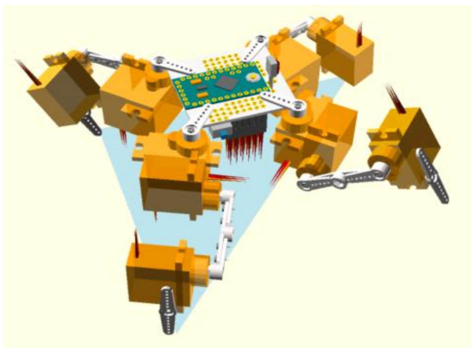



Fig 5.4   a) Area of support while standing   b) Area of support while one leg is raised

To overcome this problem, Robot needs to shift its centre of gravity slightly away from the leg to be raised. This can be achieved by shifting two legs on the side of the leg to be raised as parallel to each other and the legs on the other side to be lateral (100 degrees between them) to each other.

5.3.1.2 Balanced initial position

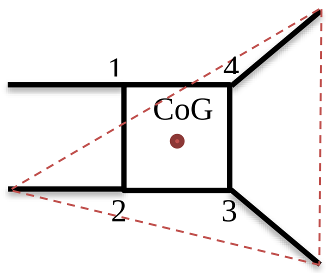


Fig 5.5 Initial balanced position

Leg1 is the leg to be lifted hence the centre of gravity is slightly shifted toward leg3. The above figure is the initial standing position of the robot and after every movement, the robot comes back to the same position may be with variation in the side of the leg to be lifted.

5.3.2 Walking algorithm using creep gait

At the beginning of every motion cycle, the robot will be initial position with one side legs being parallel and the other side being lateral to each other. For every forward, backward, left turn or right turn step, one of the legs being parallel to each other is lifted.

- Lateral =140° with respect to the robot.
- Parallel =90° with respect to the robot.
- Lateral+add =140°+30° = 170° with respect to robot.

5.3.2.1 Forward movement algorithm

- Lift leg1, shift to lateral, put the leg down.
- Leg2 to lateral, leg3 to lateral+add, leg4 to parallel pushing the robot forward.
- Leg3 to parallel

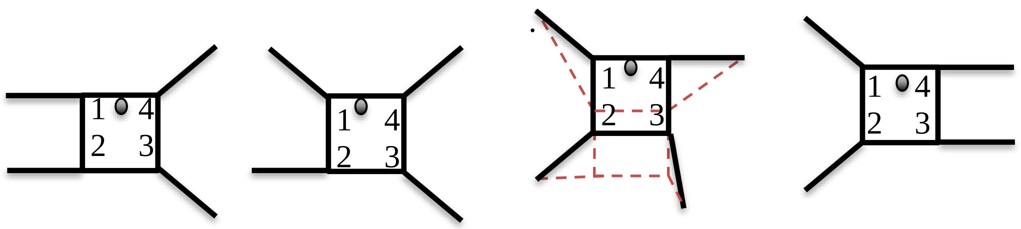


Fig 5.6 Forward motion

5.3.2.2 Backward movement algorithm

- Lift leg3, shift to lateral, put the leg down.
- Leg4 to lateral, leg1 to lateral+add, leg2 to parallel pushing the robot forward.
- Leg1 to parallel

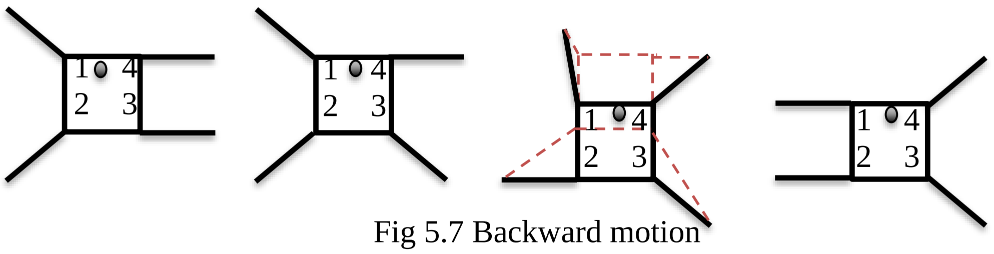


Fig 5.7 Backward motion

5.3.2.3 Right turn movement algorithm

To turn right, the right side of the robot tries to move backwards, and the left side of the robot move forward creating a pivotal motion towards the right.

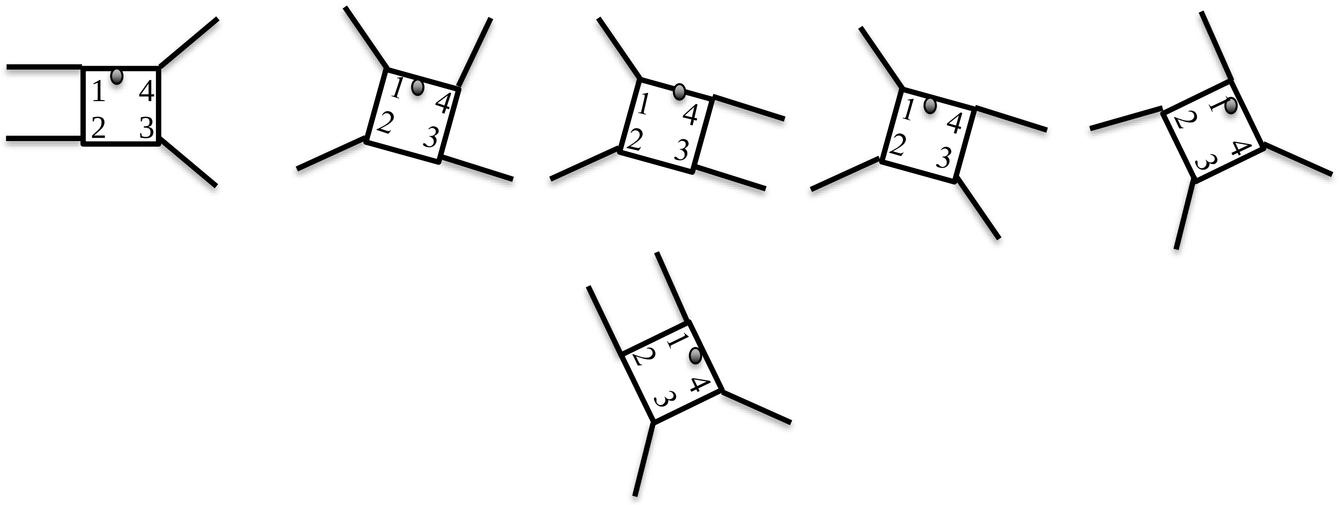


Fig 5.8 Right turn

5.3.2.4 Left turn movement algorithm

To turn left, the left side of the robot tries to move backwards, and the right side of the robot moves forward creating a pivotal motion towards the left.

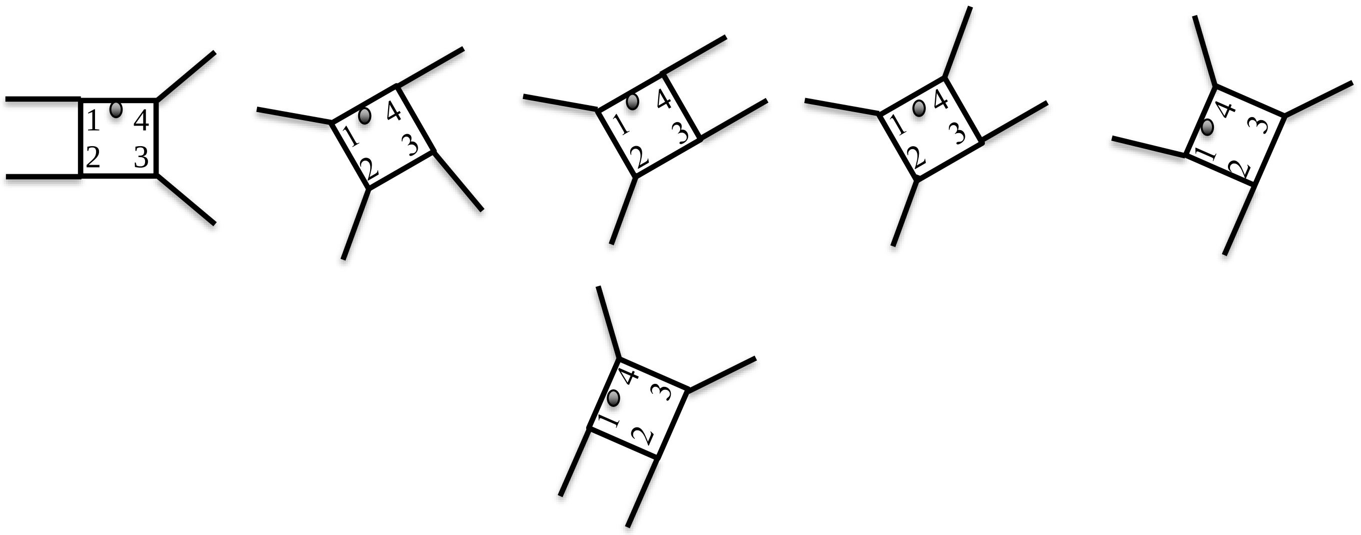


Fig 5.9 Left turn

5.3.3 Controller used for locomotion

Wi-Fi computer keyboard is used as a remote controller for the robot, pressing keys of the keyboard will run functions associated with them.

'W' → Forward.

'S' → Backward.

'A' → Turn left.

'D' → Turn right.

The wi-fi receiver is directly connected to the USB port of Raspberry Pi.

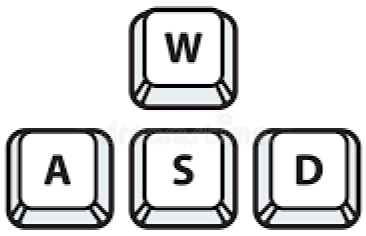


Fig 5.10 Controller and keys

CHAPTER 6

Obstacle Detection and Avoidance

6.1 Using Pi Camera and Image Processing

6.1.1 Capturing, Decoding Video Frame by Frame and Conversion

So, the video is captured at a specific frame rate. If it is a 30 FPS video, then there are 30 frames per second. So, the operation cannot be performed on the entire video. For that, the video should be segmented into frames and operation on each frame is to be performed. Converting to grayscale is simply reducing complexity, from a 3D pixel value (RGB) to a 1D value. Many tasks do not fare better with 3D pixels like in our case for edge detection. So, there is a need of converting a colour image to grayscale. Operation on a 1-channel image is easy than a 3-channel image.

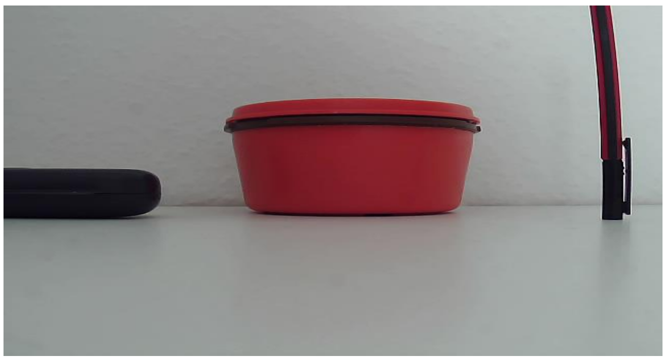


Fig 6.1 Original Image

6.1.2 Edge Detection

An edge corresponds to a region in an image where there is a sharp change in the intensity/colour between adjacent pixels in the image. A strong gradient is a steep change and vice versa is a shallow change. So, in a way, we can say an image is a stack of a matrix with rows and columns of intensities.

6.1.2.1 Applying filters to reduce the noise in video frames

When an image is acquired or a frame is captured from a video, more often the frame is not usable for what it is intended to be used. The image may not be consistent and may have many random variations in illumination or poor quality and contrast, which must be taken care of in the initial stage of processing. Image noise reduction can be done using Gaussian Filter, Median Filter, Normalized Box Filter or Bilateral Filter.

The bilateral filter technique is used for smoothing the image and making sure that the image does not have any kind of unwanted noise on the captured frame. One of the advantages of using a bilateral filter is that, unlike other filtering techniques, while it reduces the noise in an image it keeps the edges sharp which is the most important factor for us in choosing this filter method.

6.1.2.2 The Canny Edge Detection Technique

The goal of edge detection is to identify the boundaries of objects within images. Detection is used to try and find regions in an image where there is a sharp change in intensity. We can recognize an image as a matrix or an array of pixels. A pixel contains the light intensity at some location in the image. Each pixel's intensity is denoted by a numeric value that ranges from 0 to 255, an intensity value of zero indicates no intensity if something is completely black whereas 255 represents the maximum intensity of something being completely white. A gradient is a change in brightness over a series of pixels. A strong gradient indicates a steep change whereas a small gradient represents a shallow change.

The Canny edge detection process contains multiple steps and those are

- Image filtering
- Image gradient calculation
- Non-maximum suppression
- Double threshold
- Edge tracking by hysteresis

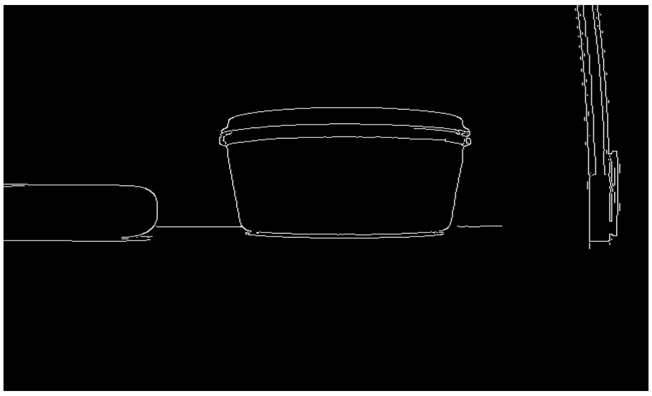


Fig 6.2 Canny Edge detection output

6.1.2.3 Comparing different edge detection methods

Operator	Advantages	Disadvantages
Classical (Sobel, Prewitt, Kirsch)	Simplicity, detection of edges and their orientations.	Sensitivity to noise, inaccurate.
Zero-Crossing (Laplacian, Second directional derivative)	Detection of edges and their orientations. Having fixed characteristics in all directions.	Responding to some of the existing edges, sensitivity to noise.
Laplacian of Gaussian (LoG) (Marr-Hildreth)	Finding the correct places of edges, testing wider area around the pixel.	Malfunctioning at the corners, curves and where the grey level intensity function varies. Not finding the orientation of the edge because of using the Laplacian filter.
Gaussian (Canny, Shen-Castan)	Using probability for finding error rate, localization, and response. Involving signal to noise ratio, better direction especially in noise conditions.	Complex computations, false zero crossings, time-consuming.

Table 6.1 Comparing different edge detection methods

6.1.3 Finding the Region of Interest and working on that part

6.1.3.1 Thresholding

Thresholding is one of the most important steps used in differentiating background and foreground. It helps in assigning pixel values to the threshold provided. In this technique, each pixel value in the frame is compared to the threshold. If the pixel value is lower than the threshold it is set to 0 and if the values are greater than the threshold it is set to a maximum which is 255. This will help in making the foreground completely white and the background black or vice versa. The function used for this is ‘cv2.threshold’.

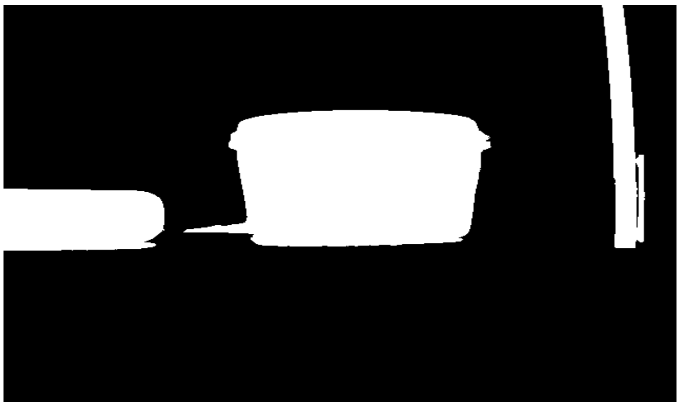


Fig 6.3 Image Thresholding output

6.1.3.2 Contour detection

Contour detection is a very useful method used in the detection of the shape of an object in front of the camera. It is a closed curve joining all continuous points having the same Colour intensity. But when contour detection is applied on any image it detects internal contours too which are not required for obstacle detection. Hence, only external contours are detected by providing the function ‘findContours’ a filtered image as an input.

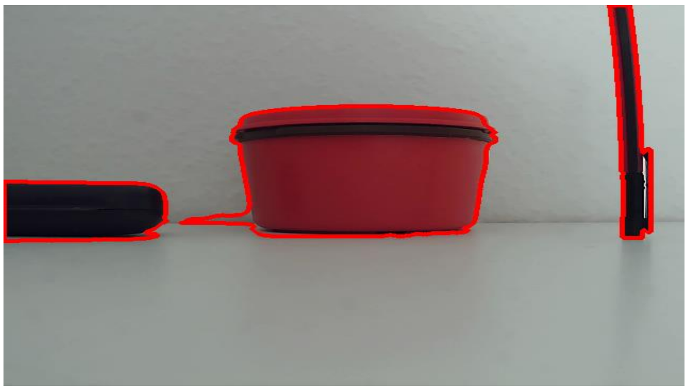


Fig 6.4 Contour detection output

6.1.3.3 Representation of edges

An empty matrix is created to store the coordinates of the edges. A for loop is used to go through every row at intervals of 5 and a nested for loop is used to determine the height of the edge at every row. Once the coordinates for all the edges are obtained, a line is drawn connecting all the heights of the edges and from the bottom of every row.

6.1.3.4 Dividing the frame into chunks

The dimensions of the entire frame are 640 by 480. Therefore, if the edges are obtained from the rows at every 5 intervals, a total of 128 edge coordinates are expected to be obtained. The first step is to decide which direction the rover should move to avoid an obstacle is to divide the frame into chunks. If the frame is divided into three chunks, approximately 42 different edge coordinates are expected to be obtained in each chunk. The green lines in an image show that the robot can move freely

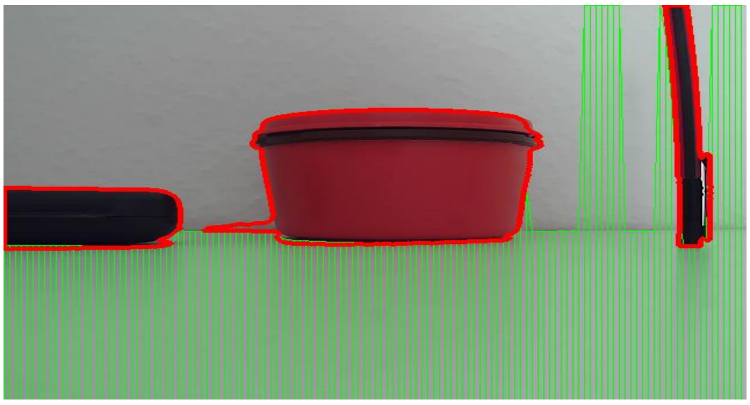


Fig 6.5 Lines drawn towards the objects

6.1.3.5 Deciding direction of movement

The average edge coordinates are obtained from each chunk and a line is drawn from the midpoint of the frame towards the chunks. If an obstacle is detected at the centre chunk, the longest line drawn towards the chunk at the sides will be considered and the robot will decide on which direction to move by following the longest line.

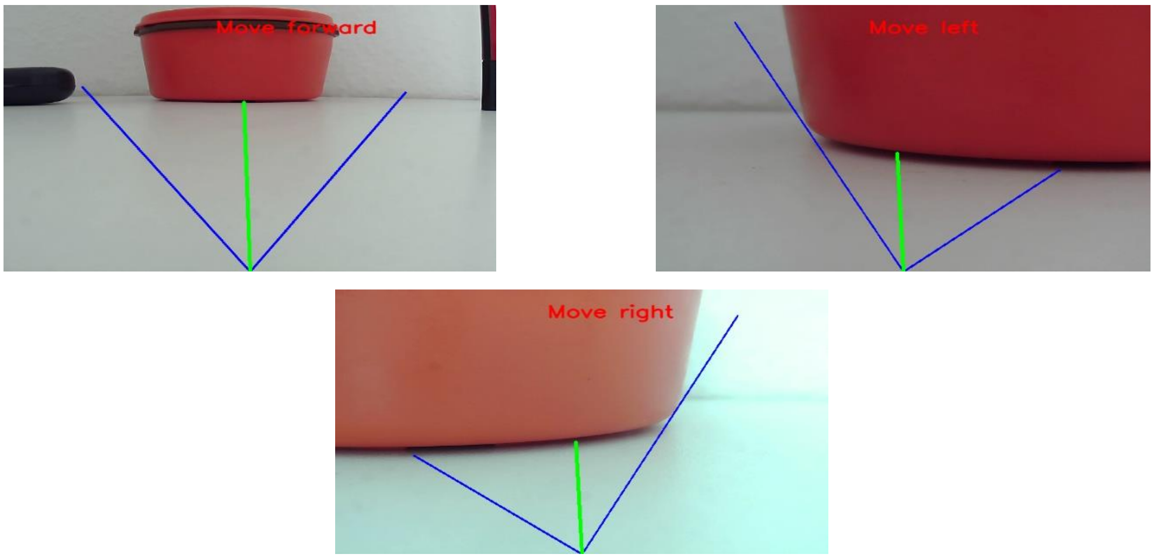


Fig 6.6 Deciding Direction of movement

6.1.4 Conclusion

OpenCV and NumPy libraries were used to write the algorithm for obstacle avoidance. Canny edge detection was used to detect the edges of the objects in the image. This algorithm works well for a clean background that does not have many colour variations. In such a scenario we could successfully detect all the objects kept in the frame.

6.2 Ultrasonic Sensor for Obstacle Detection

Since obstacle detection using image processing has its limitations in low light environments, a simple ultrasonic sensor module is used to ensure the safety of the robot.

Ultrasonic sensors work by sending out a sound wave at a frequency above the range of human hearing using an ultrasonic transmitter and determine the distance to a target by measuring time lapses between the sending and receiving of the ultrasonic pulse.

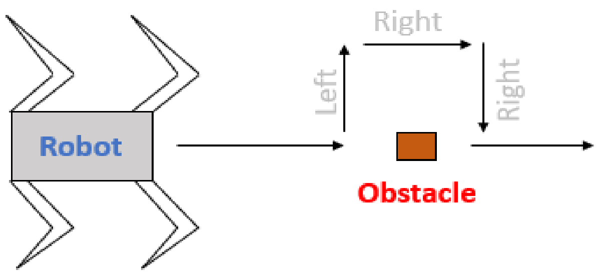


Fig 6.7 Bypassing the obstacle

If an obstacle is detected, the robot will bypass the obstacle by walking beside it and informs the user that there is an obstacle.

6.3 Video Streaming

Robot's surroundings can be continuously viewed and judged wirelessly using video streaming using Pi camera. Manual controlling of the robot without a video streaming service will become difficult since there is an absence of point of view of the robot. The real-time footage from the Raspberry Pi camera is transmitted wirelessly to the user device using VLC media player software. The VLC media player is opened in both Raspberry Pi and stream display devices.

- **Command to be entered on command prompt of Raspberry Pi:**

  raspivid -o - -t 0 -n -w 600 -h 400 -fps 12 | cvlc -vvv stream:///dev/stdin --sout '#rtp{sdp=rtsp://:8554/}' :demux=h264

  where,

  '-w' indicates the width of the video frame (available range: 64 to 1920)

  '-h' indicates the height of the video frame (available range: 64 to 1080)

  '-fps' indicates frames per second of the video (available range: 2 to 30)

- **Command to be entered in the VLC media player of the display device:**

  rtsp://###.###.###.###:8554/

  where ###.###.###.### represents IP address of Raspberry Pi.

CHAPTER 7

SENSOR MONITORING AND POWER SUPPLY DESIGN

7.1 Sensor Data Monitoring

The robot’s environment is monitored using a DHT11 temperature and moisture sensor and the data is communicated to the Adafruit MQTT server. Sensor data can be viewed continuously and can be visualized in the desired format. Thingspeak (MQTT service provider) is being used to monitor sensor readings and plot the values.

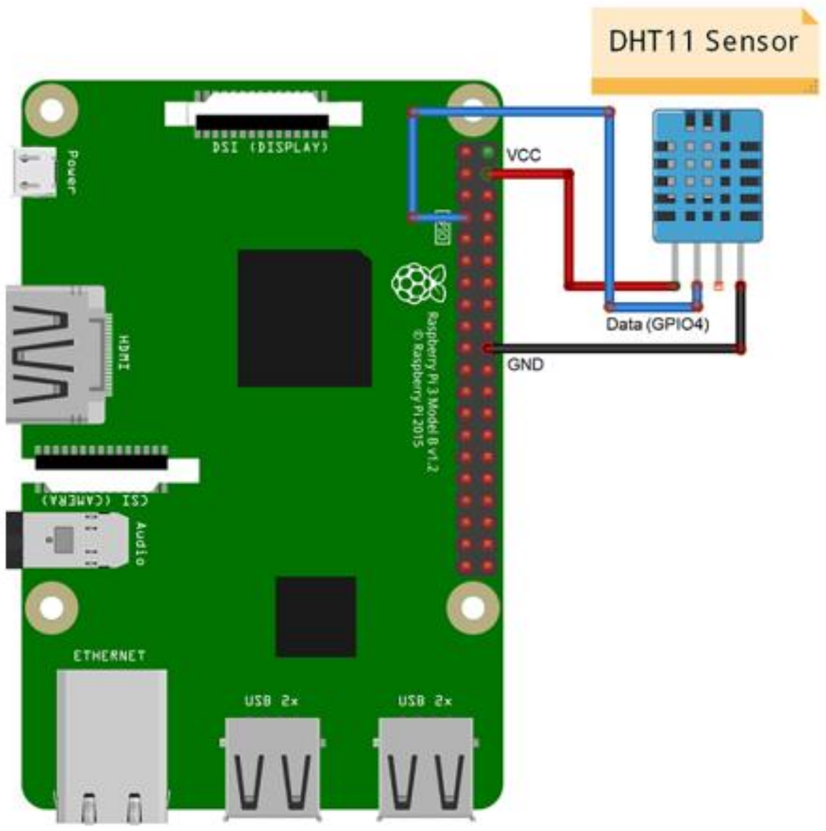


Fig 7.1 DHT11 sensor to Raspberry Pi connection

Data can be remotely monitored and visualized since the quadruped robot is connected to the internet, Pi camera and DHT11 sensor provide quite a lot of information regarding robots surrounding

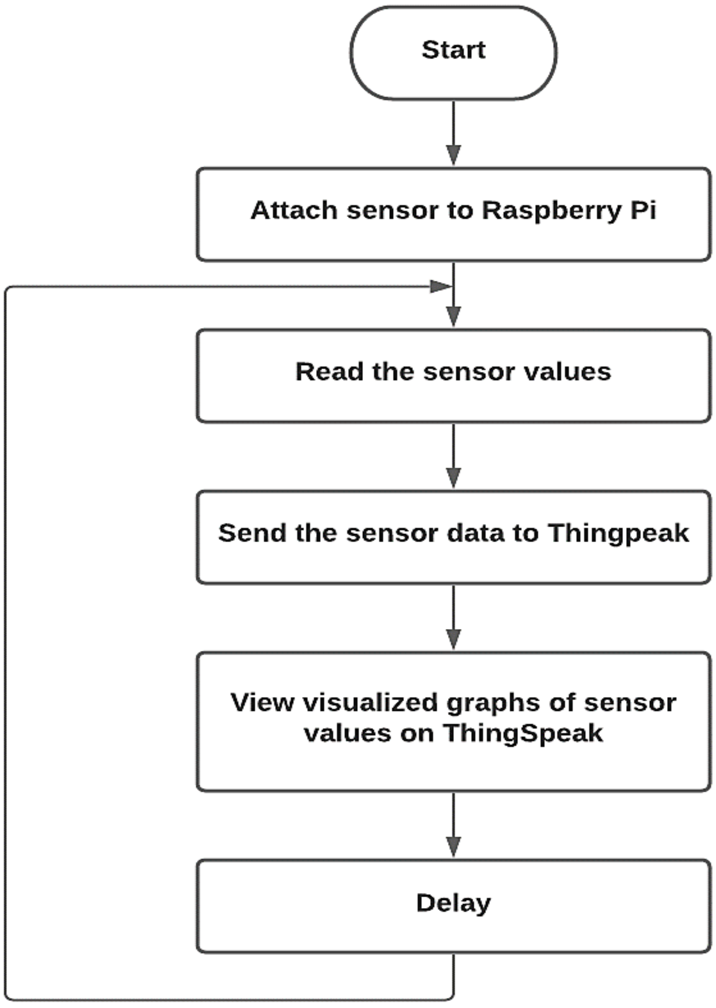

Start
Attach sensor to Raspberry Pi
Read the sensor values
Send the sensor data to Thingpeak
View visualized graphs of sensor values on ThingSpeak
Delay

Fig 7.2 Flowchart

7.2 Power Supply design

- **Inductance**

$$L1 = \frac{(V_{in} - V_{out})V_{out}}{0.3 * \mathrm{I_{out}} * F_{SW} * V_{out}}$$

$$= \frac{(9-5)5}{0.3*3*250\mathrm{k}*9} = 9.876\mu H, \text{ choose } 8.2\mu F$$

- **Input ripple voltage** must be lesser than 1% of input voltage = 1% of 9v = 90mV

$$V_{ri} = \frac{\mathrm{I_{out}}}{4*F_{SW}*C_{in}} \quad \text{choose } 40\mu\mathrm{F}$$

- **Output ripple voltage** must be lesser than 1% of output voltage = 1% of 5v = 50mV

$V_{ro}=\frac{(V_{in}-V_{out})V_{out}}{8*V_{in}} = \frac{1}{F_{SW}*F_{SW}*L*C_{out}}$ Choose 180μF

- **Output voltage divider circuit selection**

$R_{FB2}= \frac{V_{out}}{1.285}$ - 1* R_{FB1}

If $R_{FB1} = 1k\Omega$, R_{FB2}=2.87kΩ

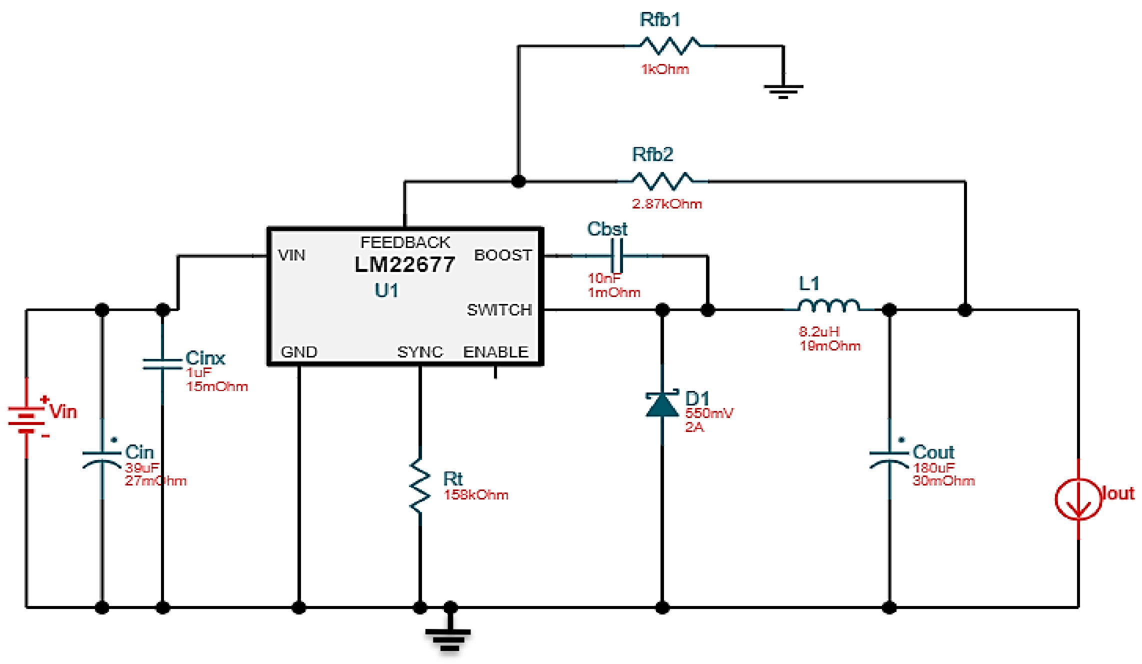


Fig. 7.3 Power Supply circuit

CHAPTER 8

RESULTS AND DISCUSSION

Quadruped Robot's 3D printed parts were assembled by polishing and grinding some joints due to misalignment and uneven parts for smoother movement of the robot. All 12 servo motors are placed and tightened using screws.

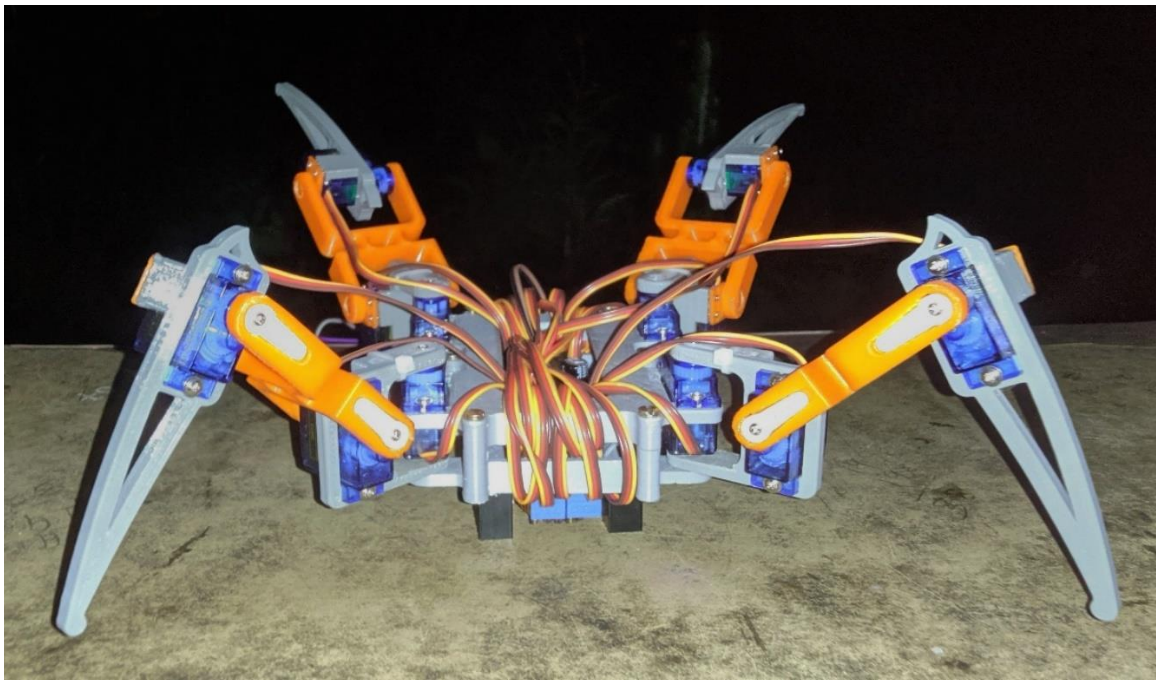
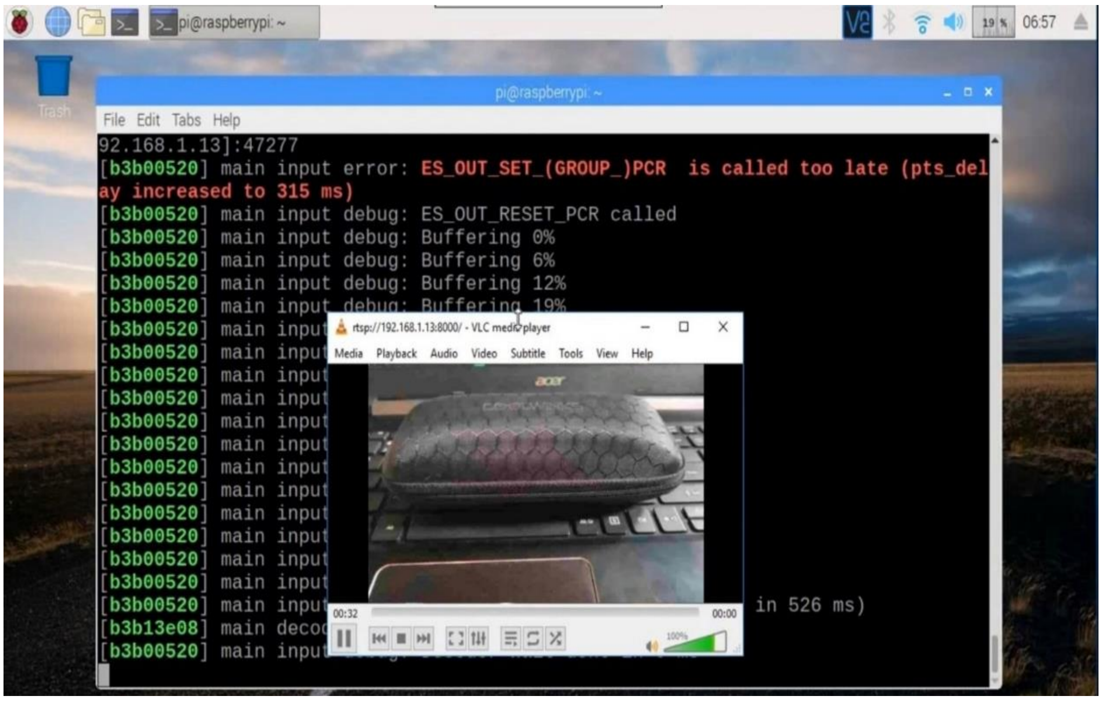


Fig 8.1 Assembled robot

An algorithm for locomotion i.e., forward, backward, left turn, right turn movements of the robot were developed and programmed. Initially, the robot was unable to stand up on its feet because of insufficient torque caused by sudden changes in servo angle. It was overcome by increasing servo angle step by step from one angle to the other keeping the robot stable.

Video streaming using the VLC media player was executed. Since there was high latency of video transfer, the frame rate was reduced to 10fps from 30fps. Sensor data monitoring using the Adafruit MQTT server is done and sensor data were visualized in the Adafruit platform.


pi@raspberrypi: ~
File Edit Tabs Help
rtsp://192.168.1.13:8000/ - VLC media player
Media Playback Audio Video Subtitle Tools View Help
00:32
00:00

```
92.168.1.13]:47277
[b3b00520] main input error: ES_OUT_SET_(GROUP_)PCR  is called too late (pts_del
ay increased to 315 ms)
[b3b00520] main input debug: ES_OUT_RESET_PCR called
[b3b00520] main input debug: Buffering 0%
[b3b00520] main input debug: Buffering 6%
[b3b00520] main input debug: Buffering 12%
[b3b00520] main input debug: Buffering 19%
[b3b00520] main input
[b3b00520] main input
[b3b00520] main input
[b3b00520] main input
[b3b00520] main input
[b3b00520] main input
[b3b00520] main input
[b3b00520] main input
[b3b00520] main input
[b3b00520] main input
[b3b00520] main input
[b3b00520] main input
[b3b00520] main input                                    in 526 ms)
[b3b13e08] main decod
[b3b00520] main input
```

Fig 8.2 Video streaming using VLC

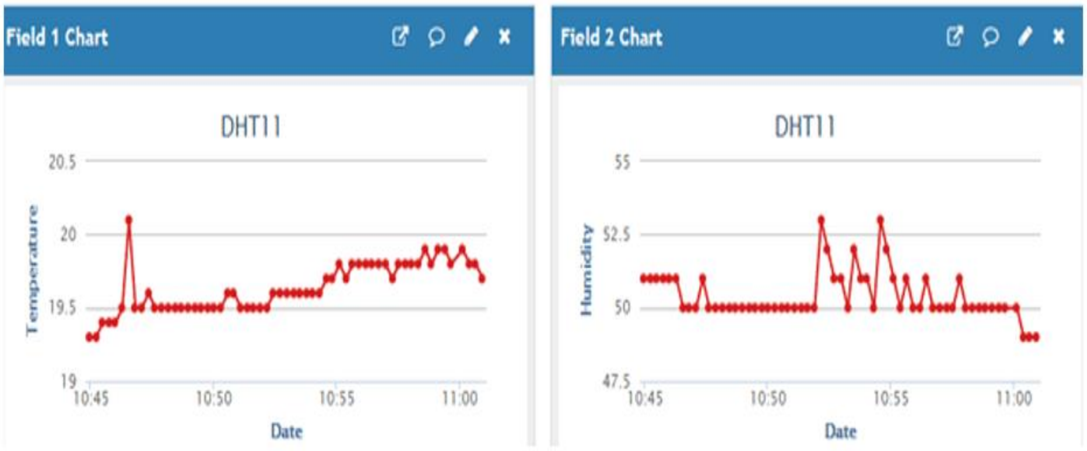


Fig 8.3 Sensor data visualization

Obstacle detection algorithm was implemented using canny edge detection method, thresholding, contouring and dividing the frame into chunks to decide the direction of movement. Obstacle detection using an ultrasonic sensor was implemented successfully on Raspberry Pi.

CHAPTER 9

CONCLUSION AND FUTURE SCOPE

The quadruped autonomous robot is a four-legged robot that mimics the walking of an animal for its motion. The creep gait used in this project is the walking gait used by predators while hunting. Creep gait is the least noisy, highly stable gait thus the robot can be used for military surveillance, locomotion in uneven terrains etc. The video streaming ability increases applications of the robot in the military domain.

There are numerous ways in which this robot can be updated and upgraded in the future. Dynamically stable trot gait can be implemented using the same robot which in turn makes the robot move faster, machine learning can be implemented for such robots for automation purposes, the robot lacks the sense of depth; it can be solved using an Accelerometer or ground faced proximity sensors etc.

For better obstacle detection, the robot can be trained using convolutional neural networks and the detection accuracy can be increased substantially. Also, this method can work on any background and type of images as per the number of input datasets used for training.

REFERENCES

[1] M. Nandhini, V. Krithika, K. Chittal, “Design of four-pedal quadruped robot”, IEEE International conference on power, Controls, Signals and instrumentation engineering (ICPCSI-2017).

[2] Chinmayi R, Yogesh Kumar Jayam, Venkatesh Tunuguntla, “Obstacle detection and avoidance robot”, IEEE International Conference on Computational Intelligence and Computing Research 2018.

[3] Adithya Singh Rathore, “Obstacle Detection for Autonomous vehicles using OpenCV Library”, International Research Journal of Engineering and Technology (IRJET) [Volume:06, Issue:01, January 2019].

[4] Firas A.Raheem, Murtadha Khudhair Flayyih, “Quadruped Robot Creeping Gait Stability Analysis and Optimization Using PSO”, 2017 Second Al-Sadiq International Conference on Multidisciplinary in IT and Communication Science and Applications (AIC-MITCSA).

[5] Ravi Kishore Kodali, Shishir Kopulwar, “A low-cost implementation of MQTT using Raspberry Pi”, 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I).