DESIGN AND IMPLIMENTATION OF MOBILE ROBOT FOR PIPELINE INSPECTION

A Thesis Submitted in Partial Fulfillment of the Requirements for the award of the degree of

Bachelor of Science in Electrical and Computer Engineering

Control Engineering

By

NAME	ID
NARDOSE TESHOME	**ETS 0875/08**
NATNAEL AKLILU	**ETS 0882/08**
SHEWANGIZAW TSIDU	**ETS 1000/08**

Under Supervision of

Mr. MEBAYE BELETE (MSc.)

ADDIS ABABA SCIENCE AND TECHNOLOGY UNIVERSITY

January 2021

EXAMINING COMMITTEE APPROVAL SHEET
DESIGN AND IMPLIMENTATION OF MOBILE ROBOT FOR PIPELINE INSPECTION

NAME	ID	
NARDOSE TESHOME	ETS 0875/08	
NATNAEL AKLILU	ETS 0882/08	
SHEWANGIZAW TSIDU	ETS 1000/08	

Approved by the examining committee members:

	Name	Academic Rank	Signature	Date
Advisor:	MEBAYE	MSc		11/02/2021
Co-Advisor:				
Examiner:	[illegible]	Lecture		
Examiner:	BINIK T.	Lecture		25/02/2021

	Name	Signature	Date
DC Chairperson:	Hamdihun Abdie Dawed Head of Electrical and Computer Engineering Department		02/02/21
Associate Dean for Under Graduate Programs:	Mulunesh Kiros Tulu (PhD) Associate Dean for College of Electrical and Mechanical Engineering		02/02/21

College of Electrical & Mechanical Engineering

ADDIS ABABA SCIENCE AND TECHNOLOGYUNIVERSITY
COLLEGE OF ELECTRICAL AND MECHANICAL ENGINEERING
DEPARTMENT OF ELECTRICAL &COMPUTER ENGINEERING

Certificate

This is to certify that the thesis entitled "Design and implementation of mobile robot" is submitted by **NARDOSE TESHOME, NATNAEL AKLILU, SHEWANGIZAW TSIDU** *for the award of the degree of Bachelor of Science in Electrical and Computer Engineering (Control Engineering), Addis Ababa Science and Technology University is a record of original work carried out under my supervision and they fulfils the requirements of the regulations laid down by the University and meets the accepted standards with respect to originality and quality. The results of the thesis have neither partially nor fully been submitted to any other University or Institute for the award of any Degree or Diploma.*

Name of Advisor: Mr. Mebaye Belete **Signature:**

Acknowledgment

We would like to express our sincere gratitude to **Dr. Dereje Engida** President for providing all necessary infrastructure in doing and completing the project work.

We would like to convey our special thanks to **Dr.Samson,** Dean of College of Electrical and Mechanical Engineering, for inspiring us and helping us in doing and completing project work successfully.

We take this opportunity to express our profound gratitude and deep regard to our beloved, dynamic, and role model **Mr. Hamdihun Abdie**, Head of Department, for his continuous support in providing all facilities, motivation and encouragement in all aspects of the completion of project work.

We are highly indebted to **Mr Mebaye Belete,** the project supervisor for providing invaluable guidance throughout this Thesis Project. His dynamism, vision, sincerity, and motivation have deeply inspired us. He has taught us the methodology to carry out the project and to present the research work as clearly as possible. It was a great privilege and honor to work and study under his guidance.

No one walks alone on the journey of life. Just where we start to thank those that joined us, walked beside us, and helped us in many aspects. We would like to express our sincere thanks to who directly and indirectly involved in the successful completion of this project work.

Nardose Teshome

Natnael Aklilu

Shewangizaw Tsidu

ABSTRACT

Pipelines have been an integral part of our construction for many centuries. Pipelines are constructed to transport all kinds of fluids, some toxic, some highly flammable and others fairly unreactive. In every case, it is important for the transported fluid to be contained within the pipeline, and under ideal situations, they have no interaction with the surrounding environment. However, every pipe, depending on the material from which it is fabricated, deteriorates progressively with time, and the pipe becomes prone to cracks and heavy corrosion. The progressive deterioration of pipelines over time through various means creates accidents and wreaks havoc on plants, animals, and humans. It is therefore vital to prevent the occurrence of such fluid leaks and save ourselves the disaster. One very effective way of doing this is to perform regular inspection of pipelines using pipe inspection robots. The information got from the inspection then can be analyzed and the appropriate preventive action will be implemented. This helps to reduce the cost of maintenance as well as time.

A pipe inspection robot is a mobile robot that is inserted into the pipe to check for obstruction, damage, and deterioration, and corrosion. This thesis presents an effective and efficient way of design and implementation of a remotely controlled mobile robot for pipeline inspection. The pipe inspection robot in this thesis uses a four-wheeled car-like model (rear wheel for drive system and front wheel for steering) for the movement and uses a single-board computer (Raspberry Pi) for the control of the robot as well as remote communication with the user through Wi-Fi or Ethernet cable. The proposed robot uses a camera for recording of the visual report, Ultrasonic sensor for protection of the robot from collusion, and it use two DC motors, one for the drive system and the other for the steering. The speed of the robot is controlled through the control of two DC motors.

Keywords: Raspberry Pi, Mobile robot, Wi-Fi, DC motors.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

A	Ampere
AGV	Autonomous Guided Vehicle
AMR	Autonomous Mobile Robot
ANFIS	Adaptive Neuro Fuzzy Inference System
ARM	Advanced RISC Machine
BLE	Bluetooth Low Energy
CCTV	Closed Circuit Television
CPU	Central Processing Unit
DC	Direct Current
GPR	Ground Penetrating Radar
GPU	Graphical Processing Unit
HDMI	High Definition Multimedia Interface
Hz	Hertz
mA	Milliampere
P	Proportional
PI	Proportional Integral
PID	Proportional Integral Derivative
PC	Personal Computer
SoC	System on Chip
SD	Secure Digital
SSH	Secure Shell
UK	United Kingdom
USB	Universal Serial Bus
V	Volt
VNC	Virtual Network Computing
Wi-Fi	Wireless Fidelity

CHAPTER ONE

1. INTRODUCTION

Disasters have always been present everywhere in the world where technical inspection has not conformed to the norms. There had been and still are enormous substance losses, which could harm mankind and the environment, and whose origin is either engineering miscalculations leading to issues in manufacturing, or those related to a lower maintenance frequency or a negligence of the detection and pursuit of cracks as a whole. One very effective way of avoiding disaster is to perform regular inspection.

Robotics is one of the fastest growing engineering fields of today. Robots are designed to remove the human factor from labor intensive or dangerous work and to act in the inaccessible environment. The use of robots is more common today than ever before and it is no longer exclusively used by the heavy production industry. Nowadays, robots can be used in pipelines to do various tasks. A pipe inspection robot can be used to check for obstruction, damage, and deterioration, and corrosion. The inspection of pipes is relevant for improving security and efficiency in industrial plants. The pipelines are the major tools for the transportation of drinkable water, effluent water, fuel, oils, and gas.

1.1 Background

There are many areas where robots can be replaced by human; amongst them pipelines is one of the most challengeable areas. Basically robots are designed to remove human intervention from the labor intensive and hazardous work environment and to explore inaccessible work places. The inspection of pipes comes in the same category because they carry toxic chemicals, fluids and most of the time has small internal diameter or bends which become inaccessible to human.

A mobile robot is a robot that is capable of moving in the surroundings and are not fixed to one physical position. Mobile robots can be autonomous (AMR - autonomous mobile robot), which means they are capable of navigating an uncontrolled environment without the need for physical or electro-mechanical guidance devices. Alternatively, mobile robots can rely on guidance devices that allow them to travel a predefined navigation route in a relatively controlled space (AGV - autonomous guided vehicle). The basic functions of a mobile robot include the ability to move and explore, transport payloads, or revenue producing cargo, and complete complex tasks using an onboard system, like robotic arms.

Pipelines have been used in major utilities for a long time. However, many troubles like aging, corrosion, erosion, cracks, and physical damage from third parties have occurred in

pipelines. Therefore, the maintenance of pipelines is essential to keep them functional. A pipe inspection robot can be used to make maintenance easier.

1.2 Statement of the Problem

Since the past, pipes have been used as the safe fluid transmitter. However, gradually these pipes are affected by fatigue, cracking, leakage, sediment, and breaking down. Moreover, sometimes the humid environment and chemical products exist in the soil, which cause rust and fatigue pipes. All these problems lead to redundancy and impose high expenses for installation and maintenance. Besides, sometimes there are some unpleasant situations such as unfit pipes. It is obvious that in these conditions, doing inspection in the toxic arena, narrow and meandering ways is impossible for human.

In addition, most of the sewer pipelines are buried or hidden by walls for their protection and hiding present in the surroundings. Thus, the pipeline accessibility to human for maintenance activities is very limited because many of them are too small to work for human. Even though there are few big once, people don't want to work inside because of dirty as well as hazardous.

Nowadays, the structures of the pipelines are designed to withstand several environmental loading conditions to ensure safe and reliable distribution from the point of production to the shore or distribution depot. However, leaks in pipeline networks are one of the major causes of innumerable losses in pipeline operators and nature. Incidents of pipeline failure can result in serious ecological disasters, human casualties and financial losses.

1.3 Objectives

1.3.1 General Objective

The general objective of this project is to design and implement a mobile robot for pipeline inspection using a Raspberry Pi.

1.3.2 Specific Objectives

- To build a remote controlled mobile robot
- To develop a robot that is easy to use and maintain
- To monitor the condition of the pipe and avoid catastrophic failure and disasters caused by pipelines through inspection
- To make maintenance easier by using video inspection to identify areas that needs maintenance.
- To help in finding the issues that prevent the pipeline from working properly

1.4. Significance of the Project

The system can be used in the following areas:

- In Oil and gas industries, to monitor the condition of the pipe and observe if sludge or deposit is formed in the pipe
- In water pipe to detect cracks, corrosion, and misaligned pipes and save water
- It can be used to save system time and labor and materials
- To avoid digging up pipes that are in perfect working order to find one little trouble spot.
- In the drainage system to find blockages or clogs
- It is cost effective since it reduces system downtime and labor and materials
- The video inspection enables the inspector to view the whole system in less time than the traditional excavation practices, and also allows to take preventive action before damage.
- It eliminates the risk of conducting unnecessary repairs.

1.5 Scope of the Project

Scopes of this project are

- The kinematics and dynamics analysis of a four-wheeled rear drive and front steer mobile robot.
- The dynamic model, simulation and control of DC motor
- The uses a camera for visual recording of the condition of the pipe.
- Prototype development using manual control

CHAPTER TWO

2. LITERATURE REVIEW

2.1 Classification based on Locomotion

Pipe inspection robots have a long history of development in the field of robotics. So many works and researches have done up to now. This work (pipe inspection robot) can be classified into several elementary forms based on the movement pattern.

Pig type**:** - is one mostly known commercial one, which is passively driven by the fluid pressure inside pipelines. It is commonly used for inspection of pipes with large diameter [1]. The movement pattern of this type of inspection robot largely depends on the flow of the fluid inside the pipe. When the flow of the fluid varies, it will be difficult for taking good inspection. Therefore, sometimes modifications have been proposed to this type of robots by adding a propeller that will basically make the robot cope up with the speed of the flow [1].

Wheel type robot: - there are many different configurations that can be used for pipe inspection robot. The simplest of these is robots where wheels are used to drive along the bottom of the pipe. The operation is much similar with vehicle. This type of inspection robots are also called as ground runner pipe inspection robots. The problem with these type is it may get difficult for navigate around very rough surface and this type of robot is only applicable in horizontal (plain) pipelines. A number of robots of this kind have been reported up to now [2]-[15].

Caterpillar type robot**:** -the movement pattern of this type of robots are similar to wheel type but caterpillar is used instead of wheel. Caterpillar type robots are usually used in conditions that demand much more grip on the walls of the pipeline this menace if the surface of the pipe is smooth that is enough to makes wheels to sleep. [16]-[20] are of this type.

Wall-press types: - this has advantage of climbing over the previous and ensure the robot dose not stuck in any hole. These robots have the possibility of twisting as they travel down the pipe but there are two ways to counteract this. The first one uses inclination sensor to measure the tilt and counteract the tilt using actuator. The second is using passive mechanisms like suspension systems so that they can easily negotiate changes [21][22] are of this type. This type has complex mechanical design which allows it to fit the change in diameter of pipes. The friction of the robot with the wall of the pipe may cause some damages in old pipes.

Inchworm type: - is usually employed for pipes with small diameter. It generally consists of two clamp sections and extension system. They reputedly clamping the rear section against

the pipe wall, extending the front section forward ,clamping this and then retracting the back section toward the front for example [23]. Similarly as wall-press type this has complicated mechanical design and it is slow because of its movement pattern.

Helical-drive type or screw type**: -** displays the motion of screw when it advances in the pipe. It achieve its motion by its set of angled wheels. However it do not work particularly well when there is sudden erosion or when the erosion is too deep as it requires contact with the pipe through 360 degrees [24] is the best example of this type.

There are also some robots that mix two of the above mechanism to take advantage from both for example [20] employ both caterpillar and wall-press type, so that it will be able to tackle problems stated above and to increases the vertical mobility

Among these movement mechanisms our project uses the second type which is wheel type robot.

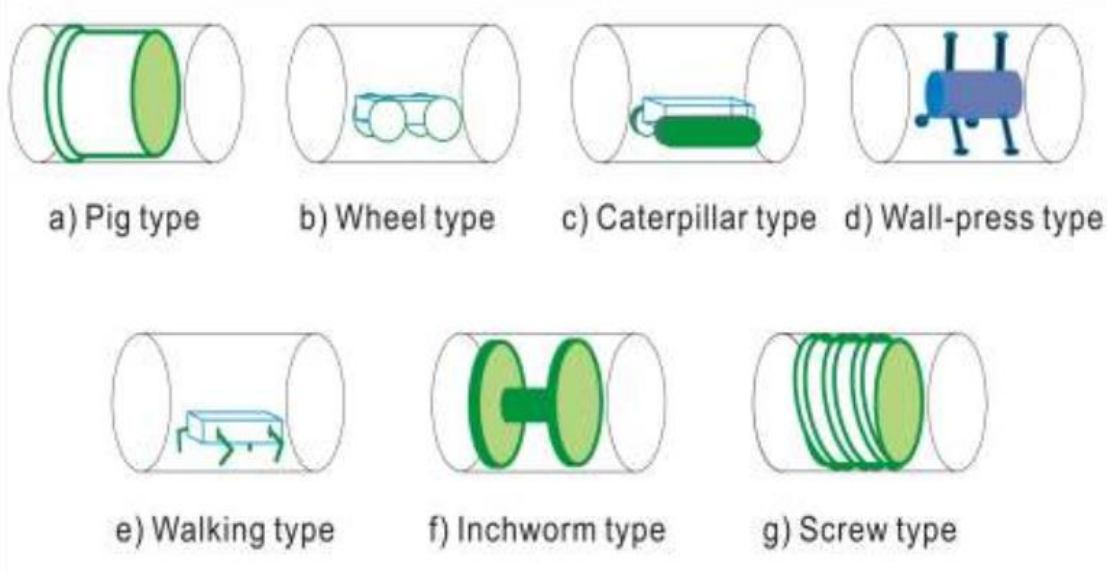


Figure 2. 1 Movement mechanisms of mobile robot

2.2 Classification based on Steering

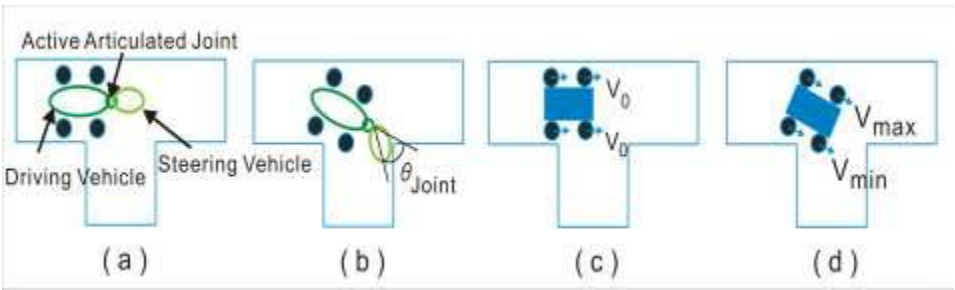


Figure 2. 2 Steering of mobile robot

Wheeled mobile robots are the vehicles whose motion is a result of interaction of wheels with the ground [4]. Wheeled mobile robots have advantages like easy to manufacture, high pay lode and high efficiency [16]. Basically wheeled mobile robots employ some kind of steering functionality in order to make the robot move. Based on the steering capability pipe inspection robots can be classified into two groups, an articulated type and differential drive type.

The differential drive type robots carries out steering by modulating the speed of driving wheels as shown in figure (c) and (d), [3]-[9] are all use differential drive mechanism for steering. Whereas [10]-[15] use articulated type steering which is the robot with active articulated joints physically similar to annelid animal in nature. It is shown in figure (a) and (b).This kind of steering also implemented in this project. The steering capability of wheeled mobile robots allows them to negotiate special fittings in the pipelines like elbows and branches.

2.3 Autonomy based Classifications

Autonomy as the word means by itself leads us to compromise on what different types of sensors, extra hardware and computational equipment we might possibly need to make the robot fully autonomous. Depending on the functionality of the robot, any given robot can be classified as Non-autonomic, Semi Autonomic or Fully autonomic robot [20].

2.3.1 Non Autonomous Robots

A non-autonomous robot usually just acts as a medium to the human operator to check the subjected area, where the operator cannot reach. The human operator remotely operates the robot, and the control signals for the robot are usually sent through a tethered cable [20]. The human controller determines the conditions of the subjected pipeline by examining the output from the sensor data, which are usually the pictures from the camera attached to the robot. These non-autonomic robots are usually used in commercial plumbing inspection applications.

2.3.2 Semi-Autonomous Robots

In semi autonomic robots the assessment of the pipeline is not completely left to the human operator. The Robot often includes modules, which will enable the robot to perform actions, which are usually pre-programmed onto the robotic modules. But still the controls to start these operations have to be issued by the human operator. So this makes the robot a semi autonomic. This semi autonomic robot does not have the ability to completely perceive the condition of the pipeline without the prompt intervention from a human controller [20].

2.3.3 Fully Autonomous Robots

A Fully autonomous robot usually carries all the required modules that are required for it to assess and process the condition of the pipeline. These are usually un-tethered robots, so all the control signals are transmitted over a radio link. The control programs that usually run on the on-board computing equipment take care of the robotic navigation and the decisions on the paths to be followed by the robot [20]. And time to time all the status messages are communicated to the human inspector or an Artificial Intelligence unit over the radio link, so that if there are any adjustments that are to be made can be communicated to the robot, so that those adjustments will be adapted by the robot in further actions that will be taken by the robotic controllers. The analysis of the acquired data can be done on the robot itself and/or can be transmitted to the remote inspector for further processing.

2.4 Sensor types used for Measurement

Pipe inspection robots employ a number of approaches to provide the desired information. The different methods employed depend on the sensors used by the robot and the parameter being measured. There are many different ways for the detection and measurement of erosion, corrosion and other defects of pipe lines.

One method that has been used extensively in earlier systems as well as more resent works is CCTV (closed circuit television). A platform travels down the pipe interior. It provides large amount of details and allows a user to see the inside view of the pipe and its condition. This requires human interpretation of the entire footage. This has several advantages like the ability to make inferences in order to identify foreign objects and possibly even cracks. However it is prone to human error.

The second classes of sensors are ultrasonic based systems; these systems use ultrasonic sensors to give use full information on the surface of point inspected [21], this systems basically send ultrasonic(high frequency sound wave) pulse out and measure the signal once it has bounced back. The time between emission and reception of that sound wave can be used to calculate the distance of obstacles based on the appearance of law of physics which govern sound waves propagation. Multiple point inspection by rotating the sensor may use to detect cracks, discontinuities, porosity and other internal defects.

In some cases ground penetrating radar is used to gather information about the pipe and the surrounding by scanning from inside or outside the pipe. GPR transmits pulses of high frequency (100 MHz to 1 GHz) electromagnetic waves through an antenna and measures the reflection using receiver [21].

Liquid penetrant testing is another way of crack detection which is the oldest and the easiest method. It is used to reveal surface discontinuities by “bleed out” of a colored or fluorescent dye from the flaw [9], which is based on the liquid’s ability to flow within a crack by capillary action.

The method in which infrared scanner is used to measure the spatial distribution of the heat and how this change over time is the other method of detecting pipe abnormality which is known as thermograph.

Mechanical feeler method is the other one in which relies on contact with the inspection surface to obtain its data. Mechanical probe feeler is extended towards the eroded surface sensor then measures the distance, which tells the level of erosion [21].

Microwave Imaging Technique relies on ultra-wide band signals for crack detection, notably in concrete structures. The image reconstruction algorithm is, in fact, a combination of the delay-and-sum beam-former with full-view mounted antennas. Indeed, this technique is used for both concrete and steel based materials, however with different accounted for considerations.

In microwave imaging technique some sensor sends signals and other sensors serve as observation points with each of them recording the received signal. Nevertheless, the microwave imaging technique in concrete is cost-effective, as it by passes the use of ionizing radiation, and gives a high definition image compared to ultrasonic techniques. As a result, captures of the object under detection are much clearer, including their peripheral areas which must also be tested [8].

2.5 DC Motor Control

As mentioned before wheeled mobile robot is used for the movement inside the pipe. This implies there must be a way of driving the robot wheels. So in this project 2 DC motors are used to achieve the robot locomotion one for driving (driving DC motor) and the other for steering (steering DC motor). In at inspection time the movement of the robot has to be controlled. In other word the heart of the robot (the motors) has to be controlled. Controlling movement of the motors is controlling their velocity and acceleration.

DC motor speed controllers are widely used for motion control of robotics, industrial control and automation systems. There are various types of speed control methods used for DC motor’s speed control. Various speed control methods used for DC separately excited motors are either Armature Control Method or Field Control Method. The Armature Control is also either Armature voltage Control or Armature resistance Control [26].

Armature voltage control is preferred among the three, because of high efficiency, good transient response and good speed regulation. however Rotor resistance control method suffers from extra power loss. And Field control method is not application for permanent magnet type DC motor [26].

There are several well-known methods for controlling DC motors by Armature voltage, such as: PI, PID, Adaptive Neuro Fuzzy Inference System (ANFIS), genetic algorithms etc.

PI, PID can be easily implemented using analog electronics. The proportional-integral-derivative controller, or PID controller is largely used for speed control of DC motor[26]-[28], can be understood as a generic control device that has the great advantage of being able to be tuned to control a certain system or process. In the PID (proportional-integral-derivative) closed loop controller, the three controllers P,I and D all have different tuning adjustments which interact all together. The effects of all three individual controllers (P-I-D) summed together to produce the output value of the system Tuning of PID controller is the major task.. There are different tuning methods used to tune PID controller such as Ziegler-Nichols method, manual tuning method and MATLAB tuning method.

Table 2. 1 Effects of independent P, I and D on the system response

Controller	Rise time(Tr)	Overshoot	Settling time	Ess
Increase kp	Decrease	Increase	Small increase	Decrease
Increase ki	Small decrease	Increase	Increase	Eliminate
Increase Kd	Small increase	Decrease	Decrease	-

This table is useful for manual tuning of the controller, beginning with the value of the parameters obtained from Ziegler-Nichols method, by observing the effect of those parameters on the unit step response of the system. Manual tuning method is a simple method but it requires a lot of experience and it takes time to get the required response.

Ziegler-Nichols method is straight forward method and largely accepted method for tuning the PID controller. It is experimental way of finding the three gains for good performance of the system. Ziegler-Nichols criterion consists of finding a gain kc with which the response of the system to a unit step input for the closed loop system is oscillatory and periodic.

- The integral and derivative parameters (gains) will be set to zero.
- Slowly increase the proportional coefficient from zero to until the system just begins to oscillate continuously (sustained oscillation).
- The proportional coefficient at this point is called the Ultimate gain (Ku).
- However, the period of oscillation at this point is called Ultimate period (Tu).

After defining the Ultimate gain and Ultimate period we can find the Kp, Ki and Kd as

$Kp = Ku/1.7$

$Ki = Tu/2$

$Kd = Tu/8$

This Tuning method is used for PID controller design of DC motor in [27] and [28].

MATLAB tuning method is software based tuning of Kp, Ki and Kd and it is a method that is used in this project for the design of PID controller for speed of the DC motor.

CHAPTER THREE

3. METHODOLOGY AND SYSTEM DESIGN

3.1 Methods

The major components of the mobile robot are a controller, sensor, actuators and power system. The controller is generally a microprocessor, an embedded microcontroller or a personal computer (PC). The sensors used are dependent upon the requirements of the robot. The requirements could be dead reckoning, tactile and proximity sensing, triangulation ranging, collision avoidance, position location and other specific applications. Actuators usually refer to the motors that move the robot can be wheeled or legged. To power the mobile robot we use DC power supply.

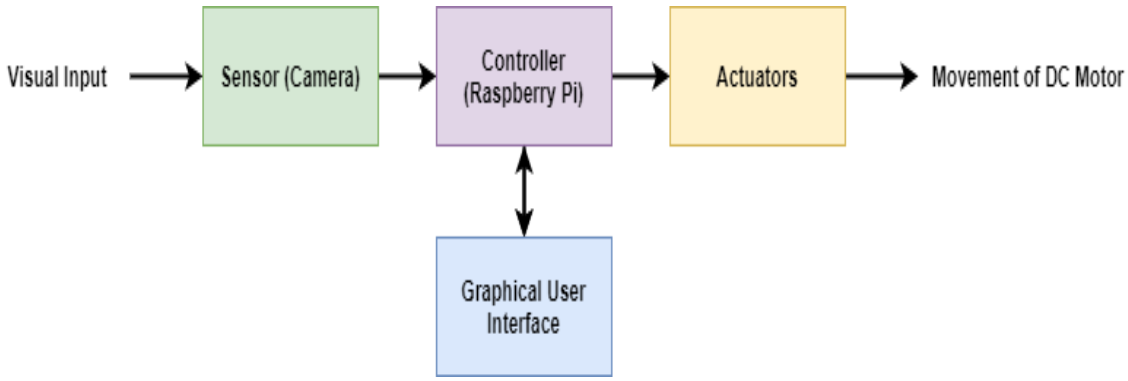


Figure 3. 1 Block diagram of the system

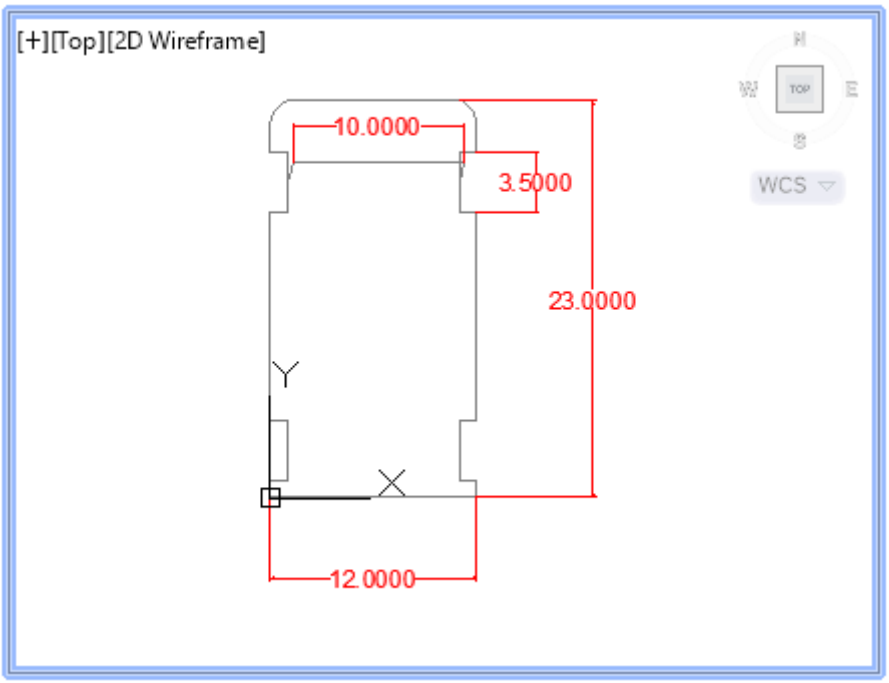
3.2 Mathematical Model

3.2.1 Derivation of the Kinematic Model

The first step taken in the derivation is to create the kinematic model by employing the non-holonomic constraints. These constraints hold under the assumption that there is no slippage at the wheel. A non-holonomic constraint is one that is not integrable. The constraints related to an automobile are those of the vehicle's velocity. As a result, the general form of the non-holonomic constraint is

$$\dot{u}\sin\theta - \dot{w}\cos\theta = 0 \tag{3.1}$$

Where $\dot{u}$ and $\dot{w}$ are the velocities of a wheel within a given (u,w) coordinate system, and θ is the angle of the wheel with respect to the x-axis. With this in mind, we examine a general front-wheel steer, rear-wheel drive vehicles. For small angles of steering, the car can be modeled as a bicycle, as shown in figure.



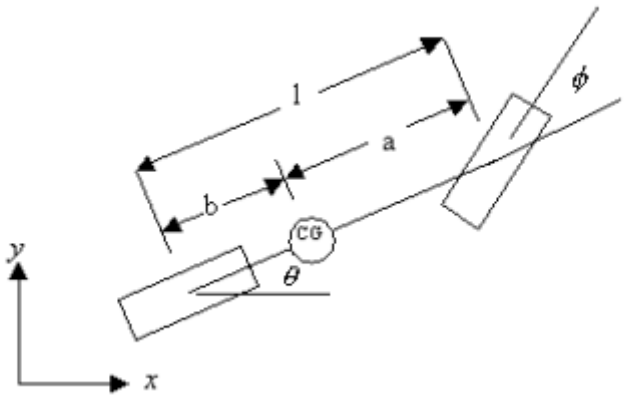


Figure 3. 2 Design and General Coordinates for a car-like robot

Denote (x; y) as being the position of the center of gravity, θ as the orientation of the vehicle with respect to the x-axis, and φas the steering angle between the front wheel and the body axis. Let (x_1; y_1) denote the position of the rear axle, and (x_2; y_2) denote the position of the front axle. This defines,

$$x_1 = x - b\cos\theta \qquad x_2 = x + a\cos\theta$$
$$y_1 = y - b\sin\theta \qquad y_2 = y + a\sin\theta$$
$$\dot{x}_1 = \dot{x} + b\dot{\theta}\sin\theta \qquad \dot{x}_2 = \dot{x} - a\dot{\theta}\sin\theta$$
$$\dot{y}_1 = \dot{y} - b\dot{\theta}\cos\theta \qquad \dot{y}_2 = \dot{y} + a\dot{\theta}\cos\theta$$

The non-holmic constraints are then written for each wheel, resulting in

$$\dot{x}_1\sin\theta - \dot{y}_1\cos\theta = 0 \qquad (3.2)$$

$$\dot{x_2}\sin(\theta+\varphi)-\dot{y}_2\cos(\theta+\varphi)=0 \tag{3.3}$$

Substituting the definition of $(\dot{x_1}, \dot{y_1})$ and $(\dot{x_2}, \dot{y_2})$ into equations 3.2 and 3.3 gives

$$\dot{x}\sin\theta-\dot{y}\cos\theta+b\dot{\theta}=0 \tag{3.4}$$

$$\dot{x}\sin(\theta+\varphi)-\dot{y}\cos(\theta+\varphi)+a\dot{\theta}\cos\theta=0 \tag{3.5}$$

Using the coordinate frame of the vehicle, as shown in figure, define the u-axis as being that which is along the length of the vehicle and the w-axis normal to the u-axis,

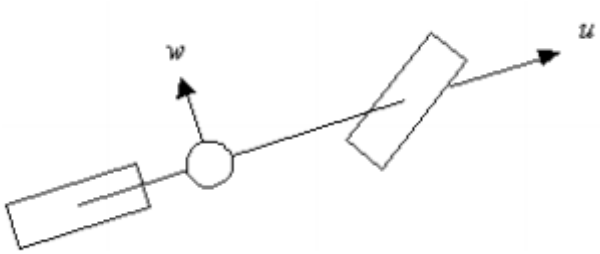


Figure 3. 3 Car body coordinates

Define

$$\dot{x}=\ v_u\cos\theta-v_w\sin\theta$$

$$\dot{y}=\ v_u\sin\theta+v_w\cos\theta$$

Where v_u and v_w are the velocities of the center of gravity along the u and w axes, respectively. Substituting these definitions into equations 3.4 and 3.5 results with

$$v_w=\dot{\theta}b \tag{3.6}$$

$$\dot{\theta}=\frac{\tan\varphi}{l}v_u \tag{3.7}$$

Thus, the derivatives of the non-holonomic equations are

$$\dot{v}_w=\ \ddot{\theta}b \tag{3.8}$$

$$\ddot{\theta}=\frac{\tan\varphi}{l}\dot{v}_u+\frac{v_u}{l(\cos\varphi)^2}\dot{\varphi} \tag{3.9}$$

3.2.2Derivation of the Dynamic Model

Inputs in a kinematic model do not directly represent actual inputs (i.e. forces and/or torques). In another words, we are neglecting dynamics of a system when dealing just with a kinematic model. Consequently, It is important to derive the dynamic model and explore its characteristics.

Added assumptions are that there is no friction force between the wheel sand the vehicle, and that the rear wheels are locked to be in the same orientation as the vehicle. Other assumptions, used in this paper are that there is no slip at the wheel, and that the driving force, based on the radius of the wheel and the drive torque, can be modeled as acting at the

center of the rear wheels. With the slippage assumption comes a pair of forces, one acting at each wheel, and perpendicular to that wheel. The forces involved in this derivation are shown in figure

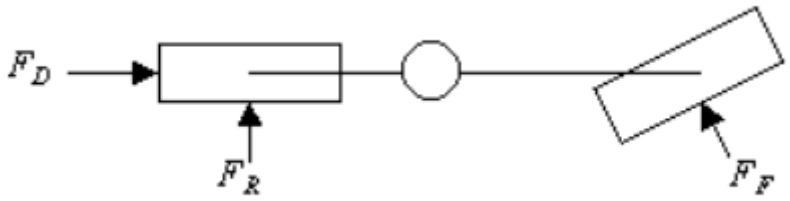


Figure 3. 4 Car Input and resultant forces

The resultant dynamic equations are

$$\dot{v}_u m = v_w \dot{\theta} - F_F \sin\varphi + F_D$$

$$\dot{\boldsymbol{v}}_{\boldsymbol{u}} = \boldsymbol{v}_{\boldsymbol{w}} \dot{\boldsymbol{\theta}} - \frac{\boldsymbol{F}_{\boldsymbol{F}} \sin\boldsymbol{\varphi}}{\boldsymbol{m}} + \frac{\boldsymbol{F}_{\boldsymbol{D}}}{\boldsymbol{m}} \tag{3.10}$$

$$\dot{v}_w m = -v_u \dot{\theta} + F_F \cos\varphi + F_R$$

$$\dot{\boldsymbol{v}}_{\boldsymbol{w}} = -\boldsymbol{v}_{\boldsymbol{u}} \dot{\boldsymbol{\theta}} + \frac{\boldsymbol{F}_{\boldsymbol{F}} \cos\boldsymbol{\varphi}}{\boldsymbol{m}} + \frac{\boldsymbol{F}_{\boldsymbol{R}}}{\boldsymbol{m}} \tag{3.11}$$

Beside the translational motion the rotational dynamics of the robot along the normal to u and w axis by the resultant lateral forces can be computed as

$$\boldsymbol{J}\ddot{\boldsymbol{\theta}} = \boldsymbol{a}\boldsymbol{F}_{\boldsymbol{F}} \cos\boldsymbol{\varphi} - \boldsymbol{b}\boldsymbol{F}_{\boldsymbol{R}}$$

$$\ddot{\boldsymbol{\theta}} = \frac{\boldsymbol{a}\boldsymbol{F}_{\boldsymbol{F}} \cos\boldsymbol{\varphi}}{\boldsymbol{J}} - \frac{\boldsymbol{b}\boldsymbol{F}_{\boldsymbol{R}}}{\boldsymbol{J}} \tag{3.12}$$

Where m and J are the mass and mass moment of inertia of the vehicle about the center of gravity. F_D is the driving force, applied at the rear axle, along the u-axis, and F_F and F_R are the resultant lateral forces on the front and rear wheels respectively. Solving equation 3.12 in terms of F_R gives

$$\boldsymbol{F}_{\boldsymbol{R}} = \frac{\boldsymbol{a}\boldsymbol{F}_{\boldsymbol{F}} \cos\boldsymbol{\varphi}}{\boldsymbol{b}} - \frac{\boldsymbol{J}\ddot{\boldsymbol{\theta}}}{\boldsymbol{b}} \tag{3.13}$$

Substituting equations 3.8 and 3.13 into equation 3.11 reveals

$$\ddot{\boldsymbol{\theta}}\boldsymbol{b} = -\boldsymbol{v}_{\boldsymbol{u}}\dot{\boldsymbol{\theta}} + \frac{\boldsymbol{F}_{\boldsymbol{F}} \cos\boldsymbol{\varphi}}{\boldsymbol{m}} + \frac{\boldsymbol{a}\boldsymbol{F}_{\boldsymbol{F}} \cos\boldsymbol{\varphi}}{\boldsymbol{b}\boldsymbol{m}} - \frac{\boldsymbol{J}\ddot{\boldsymbol{\theta}}}{\boldsymbol{b}\boldsymbol{m}}$$

$$\boldsymbol{F}_{\boldsymbol{F}} = \frac{\boldsymbol{b}^2\boldsymbol{m}+\boldsymbol{J}}{\boldsymbol{l}\cos\boldsymbol{\varphi}}\ddot{\boldsymbol{\theta}} + \frac{\boldsymbol{b}\boldsymbol{m}}{\boldsymbol{l}\cos\boldsymbol{\varphi}}\boldsymbol{v}_{\boldsymbol{u}}\dot{\boldsymbol{\theta}} \tag{3.14}$$

3.2.3 Model of Drive a DC Motor

DC motor consists of a stator, a rotor and a commutator. The stator is the housing of the motor and contains magnet, bearing and etc. The rotor is the rotating part of the motor, which contains a coil of wire through which current flows. The coil of wire in the rotor connects to the commutator and receives current through brushes. The commutator ensures that the current flows in the proper direction while the rotor turns.

The motor used in this mobile robot drive system is permanent magnet separately exited DC motor. The magnetic flux φ between the stator and the rotor is constant since it is permanent magnet.

The torque T_m developed by the motor is given by the relation:

$T_m = K_m I_a \varphi$ where K_m is the armature coil constant and I_a is armature current, but φ is constant, this leads to another constant $K_a = K_m * \varphi$

$$T_m = K_a I_a \quad (3.15)$$

The back emf e_b is proportional to the motor speed,

$$e_b = K_b * \omega_m \quad (3.16)$$

The DC motor is coupled with the wheel shaft by speed reduction gear. This gives a relation

$$\omega_s = N_m/N_s * \omega_m \text{ and } T_s = N_s/N_m * T_{L.} \quad (3.17)$$

Where,

ω_s= angular velocity of the wheel shaft

ω_m = angular velocity of the motor shaft

N_m = Number of teeth of the input (motor)

N_s = Number of teeth of the output gear (wheel shaft)

T_s = Torque of the wheel shaft

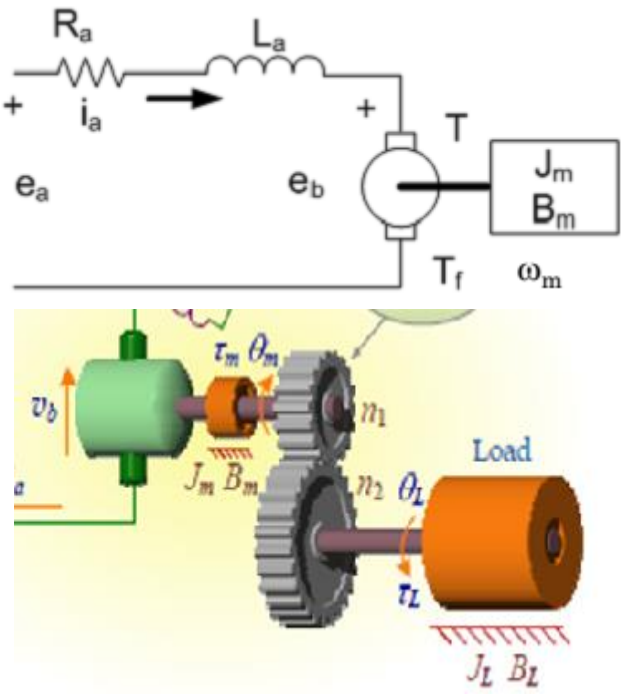


Figure 3. 5 Equivalent model of a DC motor with external load coupled by gears

The electrical and mechanical differential equations governing the dynamics of the motor derived:-

By applying Kirchhoff's law of voltages for the rotor network: -

$$Va = L\frac{dI(t)}{dt} + R\,I(t) + e_b$$

$$Va = L\frac{dI(t)}{dt} + R\,I(t) + K_b * \omega_m. \quad (3.18)$$

By applying Newton law of motion on the motor shaft

$$J_m\frac{d\omega m}{dt} + b_m\omega_m + T_L = T_m = K_aI(t) \quad (3.19)$$

From3.17

$$T_L = N_m/N_s * T_s$$

Appling Newton law of motion on the wheel shaft

$$J_s\frac{d\omega s}{dt} + b_s\omega_s = T_s \quad (3.20)$$

$$T_L = N_m/N_s \left(J_s\frac{d\omega s}{dt} + b_s\omega_s\right)$$

From 3.17

$$\omega_s = N_m/N_s * \omega_m$$

$$T_L = N_m/N_s \left(J_s\frac{dNm/Ns * \omega m}{dt} + b_s\, N_m/N_s * \omega_m\right)$$

$$T_L = (N_m/N_s)^2 \left(J_s\frac{d\omega m}{dt} + b_s\omega_m\right) \quad (3.21)$$

Substitute 3.21 in to 3.19

$$J_m\frac{d\omega m}{dt} + b_m\omega_m + (N_m/N_s)^2 \left(J_s\frac{d\omega m}{dt} + b_s\omega_m\right) = T_m = K_aI(t)$$

$$(J_m + (N_m/N_s)^2J_s)\frac{d\omega m}{dt} + (b_m + (N_m/N_s)^2b_s)\omega_m = K_aI(t) \quad (3.22)$$

Applying the Laplace transform to 3.18 and 3.22

$$Va(S) = LS\ I(S) + R\ I(S) + K_b*\omega_m(S) \quad (3.23)$$

$$(J_m + (N_m/N_s)^2J_s)\ S\ \omega_m(S) + (b_m + (N_m/N_s)^2b_s)\omega_m(s) = K_aI(s) \quad (3.24)$$

From 3.24 solve for I(s) and substitute in 3.23

$$Va(S) = (LS + R)(J_m + (N_m/N_s)^2J_s)\ S\ \omega_m(s) + (b_m + (N_m/N_s)^2b_s)\omega_m(s)/K_a) + K_b*\omega_m(S) \quad (3.25)$$

Let $J_t = J_m + J_s(N_m/N_s)^2$

$b_t = b_m + b_s(N_m/N_s)^2$

$$Va(S) = (LS + R)(J_t\ S + b_t)\omega_m(s)\ /K_a) + K_b*\omega_m(S) \quad (3.26)$$

From 3.26 and 3.17 we can relate the input voltage with the output wheel shaft velocity.

$$Va(S) = ((LS+R)(SJ_t + b_t)/K_a + K_b)\omega_m(S) = ((LS+R)(SJ_t + b_t)/K_a + K_b)\ N_s/N_m * \omega_s(S) \quad (3.27)$$

The transfer function then can be found as

$$G(S) = \omega s(S)/Va(S) = \frac{\left(\frac{Ns}{Nm}\right)Ka}{(LS + R)(SJt + bt) + Kb*Ka} \quad (3.28)$$

3.2.3.1 DC motor parameter identification

The constant parameters in the transfer function, like R, L, J_m, Js and so on, are defined from the motor datasheet and from tests done on the motor.

R: Motor armature resistance, unknown and will be measured directly. R=2.8Ω

K_b : Back-emf constant. Unknown and will be identified through steady-state analysis.

Ka: Torque constant. Unknown and will be identified through steady-state analysis. Related to K_b

b: Friction coefficient. Unknown and will be identified through steady-state analysis.

The motor has an operating voltage range of 1.5 to 4.5VDC and a no load speed of 1750 RPM (@4.5VDC, 70mA).

Moment of inertia of the rotor 0.01kg.m^2 and electrical inductance of the armature coil is 0.5 H. So by taking the no lode specification of the motor for steady state analysis we can obtain k_b from 3.18 as

$Va = L\frac{dI(t)}{dt} + R\ I(t) + K_b*\omega_m$ but $L\frac{dI(t)}{dt} = 0$ since at steady state I(t) is constant.

So, $K_b = \frac{Va - R\ I(t)}{\omega m} = \frac{4.5v - 2.8\Omega*0.07A}{183.166rad/s} = 0.0235$ v.s/rad

For a perfect motor with no energy losses, it holds that ka= k_b. However, in practice this does not hold, and we know that **$k_a = 0.65k_b$** this implies k_a =0.0153 N.m/A

Similarly, to identify the damping factor b we will substitute the steady state values in 3.19.

$J_m\frac{d\omega m}{dt} + b_m\,\omega_m + T_L = K_aI(t)$ but $T_l = 0$ since at no load and $J_m\frac{d\omega m}{dt} = 0$ since ω_m is constant. So, $b_m = \frac{Ka\ I(t)}{\omega m} = \frac{0.0116*0.07A}{183.166rad/s} = 0.58 \times 10^{-5}$N.m.s

All parameters related to the motor are identified. The remaining parameters are wheel shaft moment of inertia (J_s) and wheel shaft damping factor b_s.

By definition $J = m \times r^2$ so we will obtain Js by multiplying the mass of the wheel shaft by the wheel radius.

$r_s = 0.0145m$ and $M_s = 74g$

$J_s = m_s \times r_s^2 = 0.074kgx\ (0.0145m)^2 = 0.15 \times 10^{-4}\ kg.m^2$

$N_m = 8$ and $N_s = 40$

$J_t = J_m + J_s\ (N_m/N_s)^2 = 0.01000062\ kg.m^2$

$b_t = b_m + b_s(\ N_m/N_s)^2$ bs = 0.024 since bm is much less than bs bt becomes bs

$$G(S) = \omega s(S)/Va\ (S) = \frac{0.0765}{(0.5S\ +\ 2.8\ )(0.01000062S + 0.24\ ) +\ 0.00036}$$

$$G(S) = \frac{0.0765}{0.00536\ s^2 + 0.04036\ s + 0.06756}$$

Continuous-time transfer function.

3.2.3.1 PID tuning method

The Proportional-plus-Integral-plus-Derivative (PID) controllers have found wide acceptance and applications in the industries for the past few decades. It has simple control structure which was understood by plant operators and which they found relatively easy to tune. In spite of the simple structures, PID controllers are proven to be sufficient for many practical control problems and hence are particularly appealing to practicing engineers. Atypical structure of a PID control system is shown figure below, where it can be seen that in a PID controller, the error signal e(t) is used to generate the proportional, integral, and derivative actions, with the resulting signals weighted and summed to form the control signal u(t) applied to the plant model.

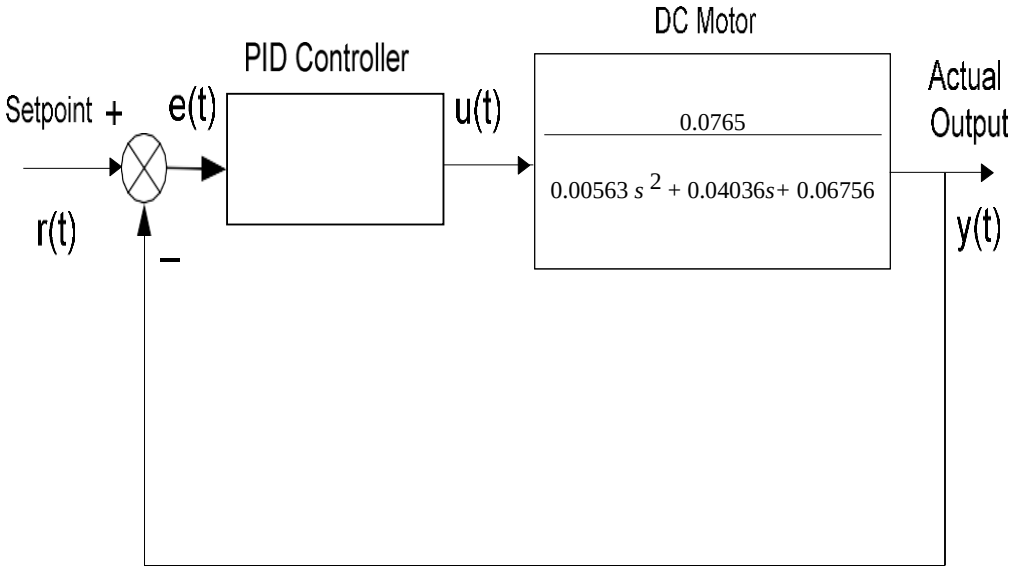


Figure 3. 6 PID control structure

A mathematical description of the PID controller is

$$u(t) = Kp[e(t) + \frac{1}{Ti}\int_0^t e(\tau)di + Td\frac{de(t)}{dt} \quad (3.29)$$

where u(t) is the input signal applied to the motor model to obtain the output signal y(t) , the error signal e(t) is defined as e(t) = r(t)-y(t) , and r(t) is the reference input signal.

3.2.3.2 The Ziegler Nichol's Step Response

A simple method of computing the parameters of a PID controller developed by Ziegler and Nichols and published in 1942 is known as Ziegler-Nichols step response method. This method is applied to the open loop step response of the load coupled DC motor displayed in Figure 4. This response is approximated by the transfer function given by a first-order plus dead time (FOPDT) given by (Nagrath & Gopal 2002):

$$G(s) = \frac{b}{1+sT}e^{-sL} \quad (3.30)$$

Where L is the apparent time delay and T is the apparent time constant, which are determined by drawing a tangent line at the inflection point of the curve and finding the intersections of the tangent line with the time axis and the steady-state level line.

The PID parameters of the Ziegler-Nichols step response method is determine as given in Table where a is computed as (Åström & Wittenmark 1995).

$$a = bL/T$$

Table 3. 1 Ziegler-Nichols step response method

Controller Parameter	Kp	Ti	Td
PID Controller	1.2/a	2L	L/2

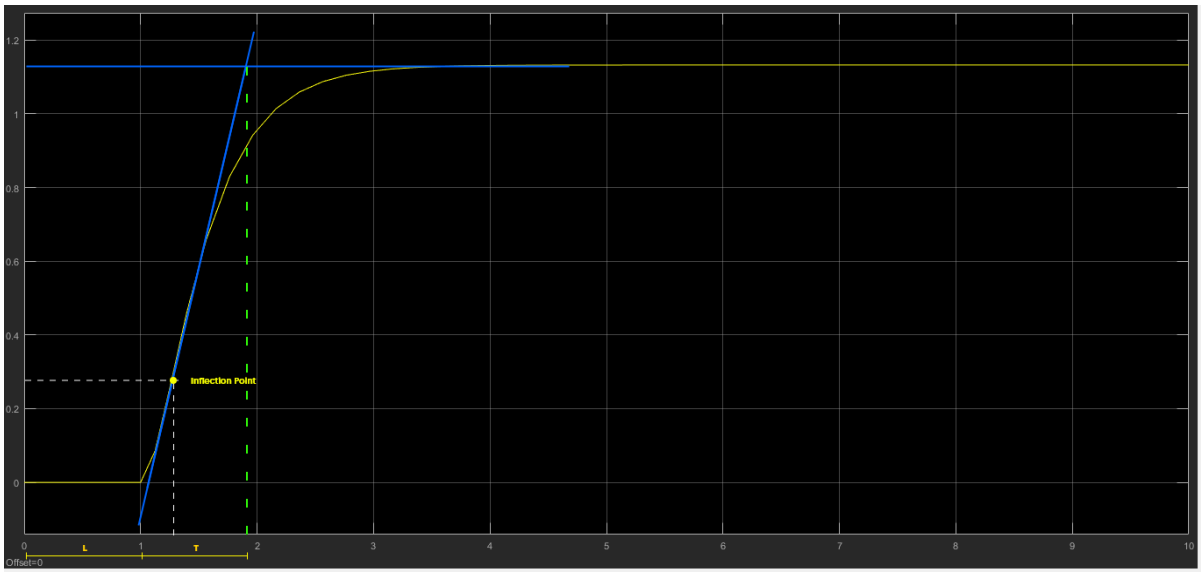


Figure 3. 7 Open loop step response of the load coupled DC motor

The inflection point was calculated using the matlab code in Appendix-III. In the above figure the step response of proportional controller added transfer function is shown. From that let us define L and T which are essential in Ziegler Nicholas method. Approximately **L=1 ,T=1.9-1=0.9 and K=1.1**

$$a = \frac{1.1 * 1}{0.9}$$

$$a = 1.22$$

$$K_p = \frac{1.2}{1.22} = 0.894$$

$$T_i = 2 * 1 = 2$$

$$K_i = \frac{K_p}{T_i} = \frac{0.894}{2} = 0.492$$

$$T_d = \frac{1}{2} = 0.5$$

$$K_d = K_p * T_d = 0.492$$

The PID controller transfer function is given by: $C = Kp + \frac{Ki}{S} + Kds$

Then the transfer function of the PID controller become:

$$C = 0.894 + \frac{0.492}{S} + 0.492s$$

3.2.4 Model of Steering DC Motor

The DC motor used for the steering mechanism is exactly the same as the DC motor used for the drive system. In this regard we need only some modification on the previous. In fact for both the drive and steering the input to the system is electric potential difference (V) but the output of the drive system is wheel velocity whereas the output of the steering system is the steering angle of the front wheels.

The steering system is a mechanism on a vehicle that serves to regulate the direction of the vehicle by means of deflecting the front wheels. In the steering system there are three main parts that are the steering column, steering gear and steering linkage. Type of steering gear that is used in the robot car is the type of rack & pinion. Rack & pinion type usage due to a simple and lightweight construction allows for construction vehicles was low.

In the steering system of this robot, there is a condition when the front wheels turn left or to right a condition in which the right and left wheels forming the same angle. This condition is called Parallel conditions. From the figure below the output of the steering system is the steering angel of the wheel.

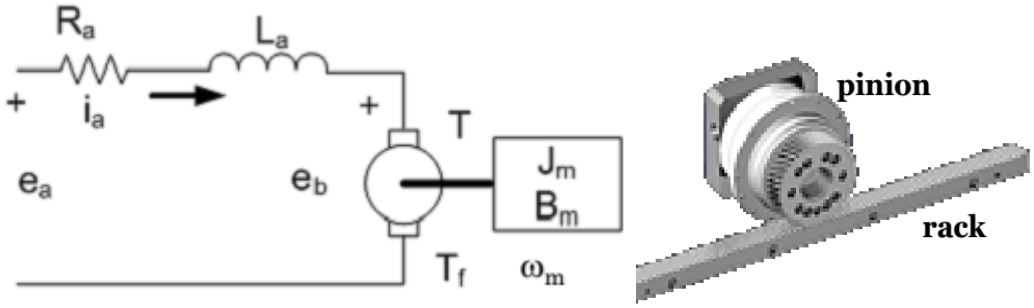


Figure 3. 8 Steering DC motor model

For this case T_L is torque of the pinion gear. Which is equal to the tangential force of the rack divided by the radius of the pinion gear.

$$T_L = F_r / r_p \text{ and } \quad \omega_m = V / r_p \tag{3.31}$$

Where, V is tangential velocity

Appling Newton law of motion on the rack link

$$m \frac{dV}{dt} + B\,V = Fr \tag{3.32}$$

From 3.27 and 3.28

$$T_L = m\frac{d\omega m}{dt} + B\omega_m \tag{3.33}$$

Converting 3.18, 3.19 and 3.30 to s domain and substituting 3.19 into 3.30.

$$V_a(s) = (Ls + R)\, I(s) + K_b\omega_m(s) \quad (3.34)$$

$$((J_m + m)s + (B + b_m))\, \omega_m(s) = K_a I(s) \quad (3.35)$$

Let $J_m + m = A$ and $B + b_m = C$

From 3.31 and 3.32

$$V_a(s) = (Ls + R)\,((J_m + m)s + (B + b_m))\,\omega_m(s) / K_a + K_b\omega_m(s)$$

$$\Theta(s) = \omega_m(s)/s$$

Therefore the transfer function of the steering system become

$$Gs(s) = \Theta(s)/V_a(s) = \frac{Ka}{(Ls + R)(As + C) + KaKb} * \frac{1}{s}$$

$$Gs(s) = \frac{0.0153}{(0.5s + 2.8)(0.01 + 0.024) + 0.00036} * \frac{1}{s}$$

From the mechanism in the above figure

$$\text{Sin}(\varphi(t)) = \frac{4.8 - 2\Pi r\Theta(t)}{1.2} = 4 - 5.235r\Theta(t)$$

$$\varphi(t) = \sin^{-}(4 - 5.235r\Theta(t))$$

3.3 System Design with Block Diagrams

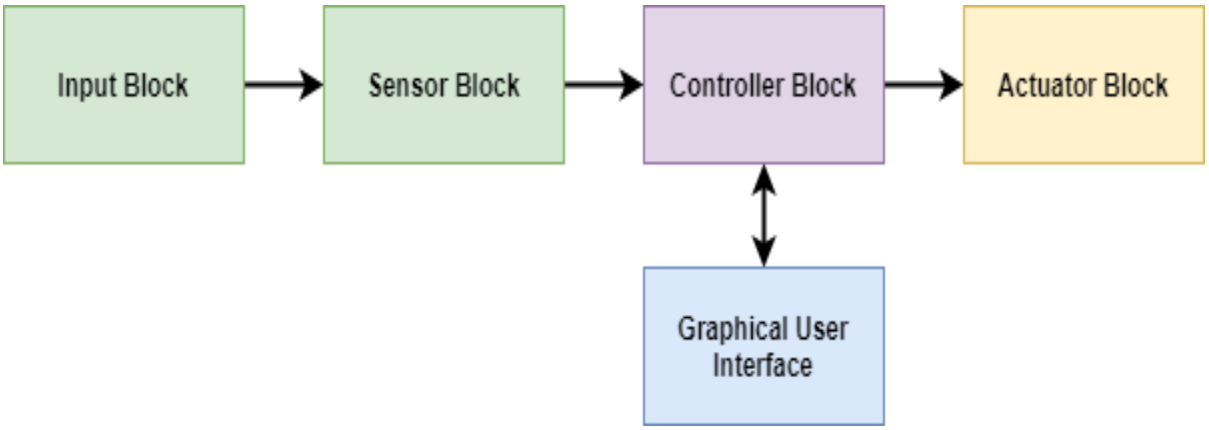


Figure 3. 9 Block diagram of the mobile robot

3.4 Description of Block Diagrams

Input Block**:** this block represent the working environment of the robot which means the pipes condition.

Sensor Block**:** Consists of the camera sensor which will be used for visual recording of the surrounding environment.

Controller Block**:** this block performs a control of the whole system. Main functions and features of this block are the operating system and soft programs, maintain communication between the user and the system thorough wireless or Ethernet communication and obtain the input data from the sensor and send correct activity to the actuator i.e. make decision based on its input

User interface block:-this block allows the user to observe and monitor the system remotely by the provided user interface through wireless or Ethernet communication.

Actuator Block**:** Consists of two DC motors. One of the DC motors is used for driving the mobile robot while the other is used for steering.

3.5 Description of Components (Materials) and Specifications

3.5.1Raspberry Pi

The Raspberry Pi is a single-board computer created by the Raspberry Pi Foundation, a charity formed with the primary purpose of reintroducing low-level computer skills to children in the UK. The aim was to rekindle the microcomputer revolution from the 1980s, which produced a whole generation of skilled programmers. Raspberry Pi supports many interfaces Display (with audio), WiFi, Ethernet, and a method of input such as a keyboard and mouse.

What is with the name?

The name, Raspberry Pi, was the combination of the desire to create an alternative fruit-based computer (such as Apple, BlackBerry, and Apricot) and a nod to the original concept of a simple computer that can be programmed using *Python* (shortened to *Pi*).

Raspberry Pi also supports many other programming languages that you can choose, from high-level graphical block programming, such as Scratch, to traditional C, right down to BASIC, and even raw Machine Code Assembler. Raspberry Pi needs an SD card with an operating system installed on it to operate.

How it works?

When the Raspberry Pi powers up, it loads some special code contained within the GPU's internal memory (commonly referred to as the binary blob by the Raspberry Pi Foundation). The binary blob provides the instructions required to read the BOOT Partition on the SD card, which (in the case of a NOOBS install) will load NOOBS from the *RECOVERY* partition. If, at this point, Shift is pressed, NOOBS will load the recovery and installation menu. Otherwise, NOOBS will begin loading the OS as specified by the preferences stored in the *SETTINGS Partition*.

When loading the operating system, it will boot via the BOOT partition using the settings defined in config.txt and options in cmdline.txt to finally load to the terminal or desktop on the *root Partition*

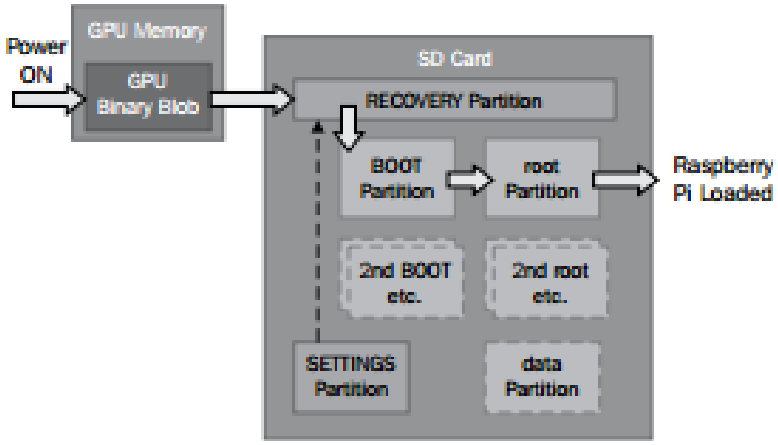


Figure 3. 10 Internal structure of the raspberry pi

Optional data Partition that allows you to keep your data files separate to the operating system.

Figure 3. 11 Raspberry Pi

Several generations of Raspberry Pi's have been released. All models feature a Broadcom system on a chip (SoC) with an integrated ARM-compatible central processing unit (CPU) and on-chip graphics processing unit (GPU). Among the various versions the Raspberry Pi 3 model B version 1.2 was used for this project.

Table 3. 2 Technical Specification of Raspberry Pi 3 model B version 1.2

Description	Values
Operating Voltage	5 VoltMicro USB power source (supports up to 2.4 Amps)
Processor	Broadcom BCM2837 64bit ARMv7 Quad Core Processor
Clock speed	1.2GHz
RAM	1GB
GPU	Broadcom Video Core IV @ 250 MHz
Wi-Fi	BCM43143 Wi-Fi on board Single band 2.4 GHz only
Ethernet	0.3 Gbps max
Bluetooth	Bluetooth Low Energy (BLE) on board
GPIO	40pin
USB 2.0 ports	4
HDMI	1 Full size HDMI
Camera port	1 CSI camera port for connecting the Raspberry Pi camera
Display port	DSI display port for connecting the Raspberry Pi touch screen display
SD port	Micro SD port for loading your operating system and storing data
Audio output	Analog via 3.5 mm phone jack and digital via HDMI

3.5.2 Camera

A camera is an optical instrument used to capture an image. At their most basic, cameras are sealed boxes (the camera body) with a small hole (the aperture) that allow light in to capture an image on a light-sensitive surface (usually photographic film or a digital sensor). Cameras have various mechanisms to control how the light falls onto the light-sensitive surface. Lenses focus the light entering the camera, the size of the aperture can be widened or narrowed to let more or less light into the camera, and a shutter mechanism determines the amount of time the photo-sensitive surface is exposed to the light.

A camera captures light photons, usually from the visible spectrum for human viewing, but in general could also be from other portions of the electromagnetic spectrum. All cameras use the same basic design: light enters an enclosed box through a converging or convex lens and an image is recorded on a light-sensitive medium (mainly a transition metal-halide). A shutter mechanism controls the length of time that light can enter the camera.

Figure 3. 12 Logitech Camera

3.5.3 DC motor

A DC motor is any of a class of rotary electrical motors that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current in part of the motor.

DC motors were the first form of motor widely used, as they could be powered from existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances. The universal motor can operate on direct current but is a lightweight brushed motor used for portable power tools and appliances. Larger DC motors are currently used in propulsion of electric vehicles, elevator and hoists, and in drives for steel rolling mills. The advent of power electronics has made replacement of DC motors with AC motors possible in many applications.

3.5.3 L293D

L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. It is designed to drive inductive loads such as relays, solenoids, DC and bipolar stepping motors, as well as other high-current/high-voltage loads in positive supply applications.

The L293D dual H driver is used to drive two DC motors.

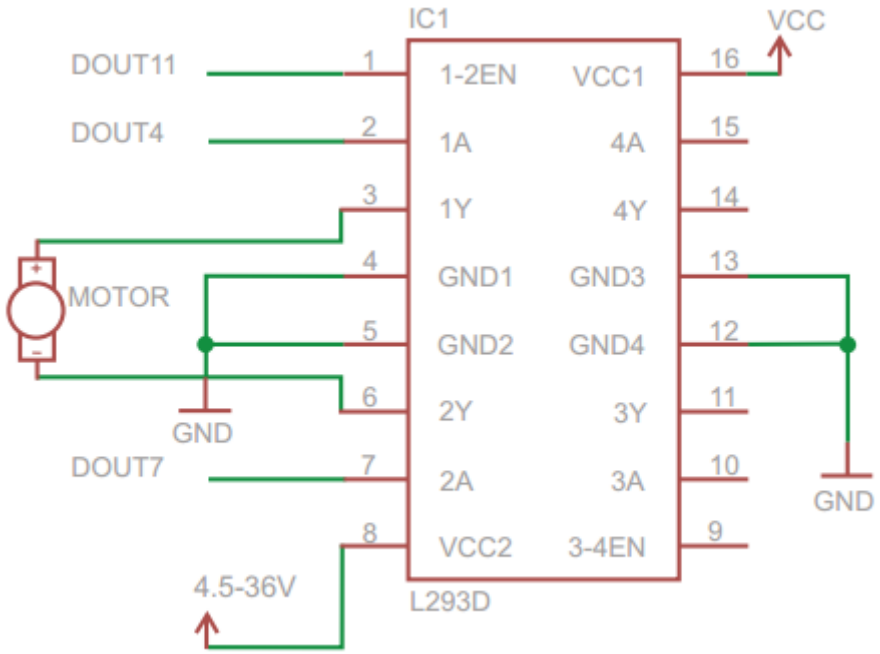


Figure 3. 13 Circuit diagram showing connections between the motor, L293D and the Raspberry Pi.

3.5.4 Servo Motor

A servomotor is a geared motor that can be set to turn to an angle, usually between 0 and 180 degrees, and is normally powered by a voltage of approximately4.8 volts. Because of their low cost and their simplicity of control, they're ideal for use in a wide variety of projects that require accurate movement. [10]

Table 3. 3 specification of the servo motor

Description	Values
Rotation	180 degree
Torque	
Supply voltage	4.8v to 6v
Average speed	60 degree per 0.2 sec
Weight	62.4g

Figure 3. 14 Servo motor

3.5.5 HCSR04 Ultrasonic Sensor

Ultrasonic sensors are devices that generate or sense ultrasound energy. They can be divided into three broad categories: transmitters, receivers and transceivers. Transmitters convert electrical signals into ultrasound, receivers convert ultrasound into electrical signals, and transceivers can both transmit and receive ultrasound.

Ultrasonic sensors use sound to determine the distance between the sensor and the closest obj ect in its path. How do ultrasonic sensors do this? Ultrasonic sensors are essentially sound sen sors, but they operate at a frequency above human hearing.

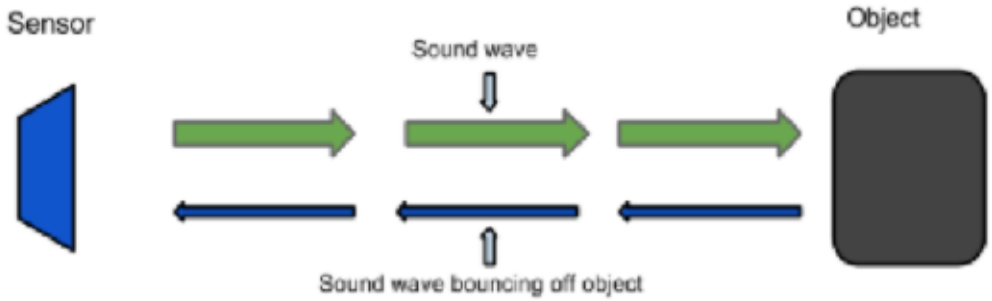


Figure 3. 15 Working mechanism of ultrasonic sensor

The sensor sends out a sound wave at a specific frequency. It then listens for that specific sound wave to bounce off an object and come back as seen in the above figure. The sensor keeps track of the time between sending the sound wave and the sound wave returning. The total distance traveled is equal to the speed multiplied by the total time taken.

Table 3. 4 Specification of ultrasonic sensor

Voltage	DC 5V
Current	15mA
Frequency	40Hz
Max Range	4m

Min Range	2cm
Measuring Angle	15 degree
Trigger Input Signal	10µS TTL pulse
Echo Output Signal	TTL level signal and the range in proportion
Dimension	45 * 20 * 15mm

Figure 3. 16 HCS04 ultrasonic sensor

3.6 Description of Software and Steps

Raspberry Pi supports many programming languages, from high-level graphical block programming to raw Machine Code Assembler. A good programmer often has to be code multilingual to be able to play to the strengths and weaknesses of each language to best meet the needs of their desired application. It is useful to understand how different languages (and programming techniques) try to overcome the challenge of converting "what you want" into "what you get" as this is what you are trying to do as well while you program.

Among the various programming languages that Raspberry Pi supports, python programming was used.

3.6.1 Python

Python is a versatile programming language that can be used for a wide range of technical tasks—computation, statistics, data analysis, game development, and more.

It is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional

programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

3.6.2 VNC (Virtual Network Computing)

Often, it is preferable to remotely connect to and control the Raspberry Pi across the network. For instance, using a laptop or desktop computer as a screen and keyboard, or while the Raspberry Pi is connected elsewhere, perhaps even connected to some hardware to which it needs to be near. VNC is just one way in which you can remotely connect to the Raspberry Pi. It will create a new desktop session that will be controlled and accessed remotely.

VNC is a graphical desktop sharing system that allows you to remotely control the desktop interface of one computer (running VNC Server) from another computer or mobile device (running VNC Viewer). VNC controls the desktop of the Raspberry Pi inside a window on your computer or mobile device. SSH is used under VNC to increase security and to get through many firewalls.

3.6.3 PuTTY

PuTTY is a free and open-source terminal emulator, serial console and network file transfer application. It supports several network protocols, including SCP, SSH, Telnet, rlogin, and raw socket connection. It can also connect to a serial port. PuTTY was originally written for Microsoft Windows, but it has been ported to various other operating systems. Official ports are available for some Unix-like platforms, with work-in-progress ports to Classic Mac OS and macOS, and unofficial ports have been contributed to platforms such as Symbian, Windows Mobile and Windows Phone.

PuTTY offers a graphical user interface that can easily be configured to allow you to tunnel other software, like your VNC viewer, over the connection. For this to work, you'll need to have a suitable SSH server installed on the remote desktop PC or server you're looking to connect to over VNC.

SSH or Secure Shell is a network communication protocol that enables two computers to communicate (http or hypertext transfer protocol, which is the protocol used to transfer hypertext such as web pages) and share data. An inherent feature of ssh is that the communication between the two computers is encrypted meaning that it is suitable for use on insecure networks. SSH is often used to "login" and perform operations on remote computers but it may also be used for transferring data. SSH Connection are highly encrypted and secure connection both from the user and the server, unlike VNC(Virtual Network Computing) which should not be used over the internet.

CHAPTER FOUR

4. RESULTS AND DISCUSSION

4.1 Circuit Diagram

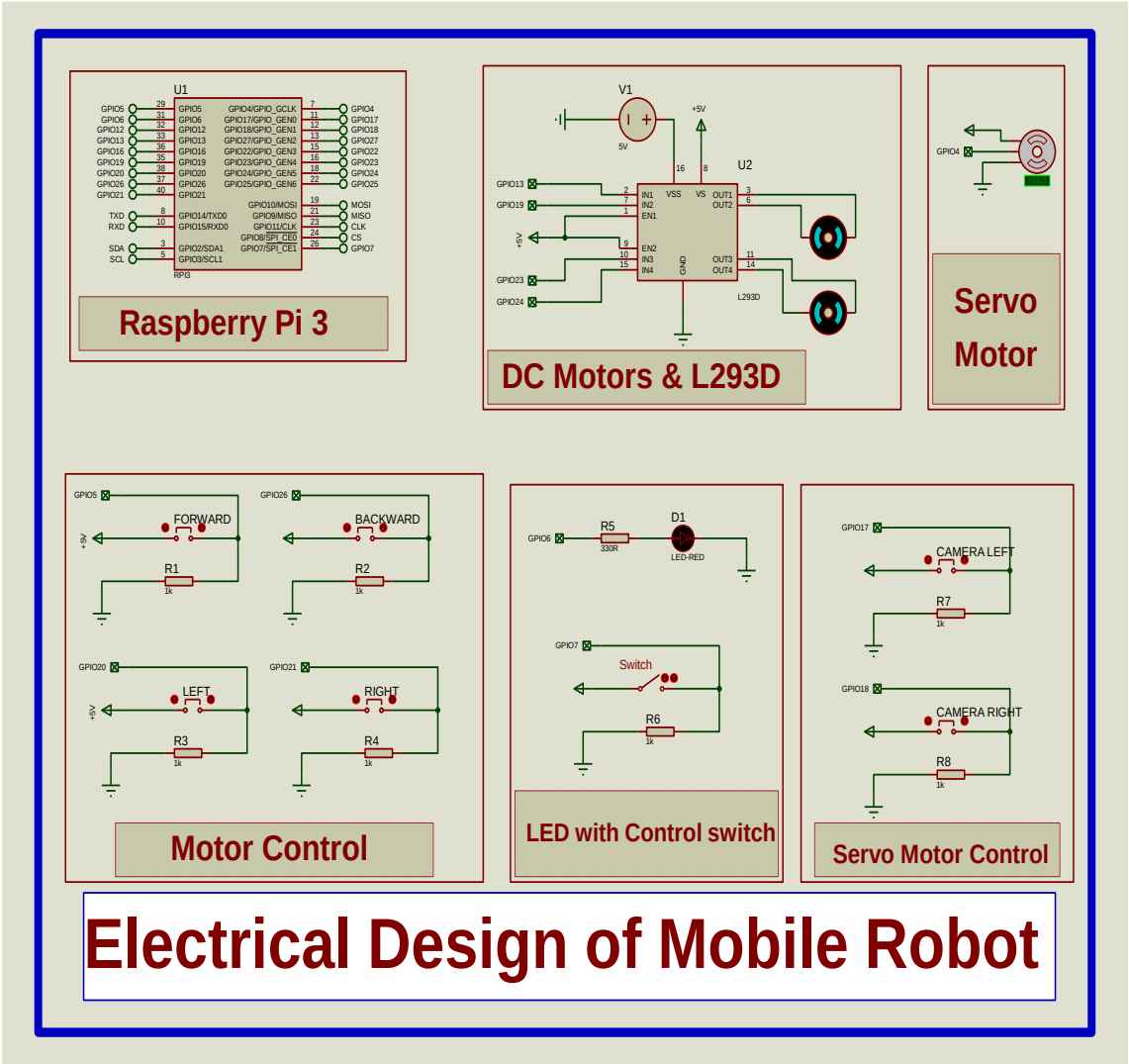


Figure 4. 1 Electrical design of mobile robot

4.2 List of Components with their specifications

Table 4. 1 list of components used in the project with their specification for electrical system

Component type/name	**Specification**
Servo motor	180 degree,62.4g
DC motor	
Controller	Raspberry Pi
Motor drive	IC (LM293D)
Bread bored	-
Camera sensor	Logitech camera
Connecting wires	-
LED	White

Table 4. 2 list of components with their specification for mechanical system for the frame work

Component type/name	**Size**
Frame	L,W 23 cm x 12cm
Wheels	D = 3.3 cm
Shaft	L = 10 cm
Rack and pinion	L = 7 cm

4.3 Operation of the circuit

The Raspberry Pi according to the program controls the two DC motors using the GPIO pins through the L293D driver. Pin 8 on the L293D provides the power for the motor. The switches are used to simulate the keyboard control of the mobile robot. The forward and backward switches are used to control the forward and backward motion of the robot respectively. The left and right switches are used to control the left and right steering respectively.

On the computer UP and DOWN arrow keys are used for the forward and backward motion and LEFT and RIGHT arrow keys are used for steering control. LED was hooked up to pin 6 to provide light for camera sensor.

4.4 Simulation Diagram

The open loop and closed loop matlab simulink model of the dc motor shown below

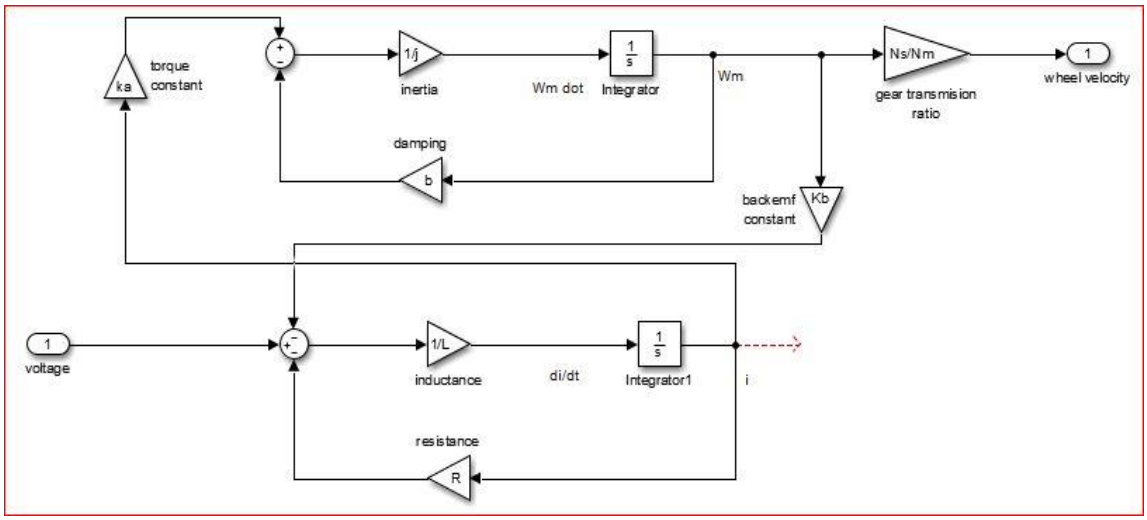


Figure 4. 2 Open loop model of the DC motor

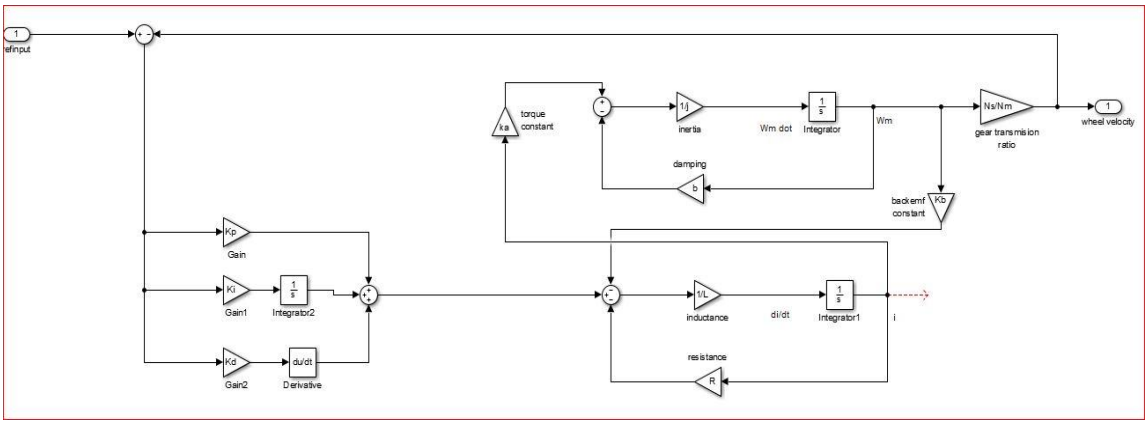


Figure 4. 3 Closed loop PID model

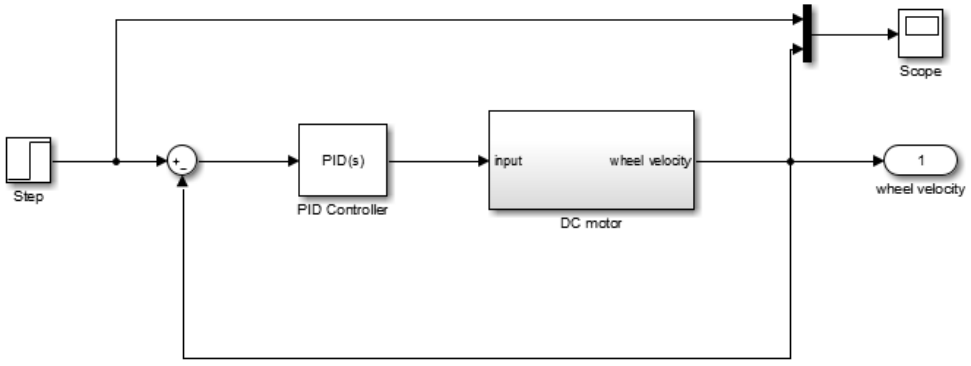


Figure 4. 4 Matlab PID tuner with DC motor

4.5 Simulation Results and their Interpretations

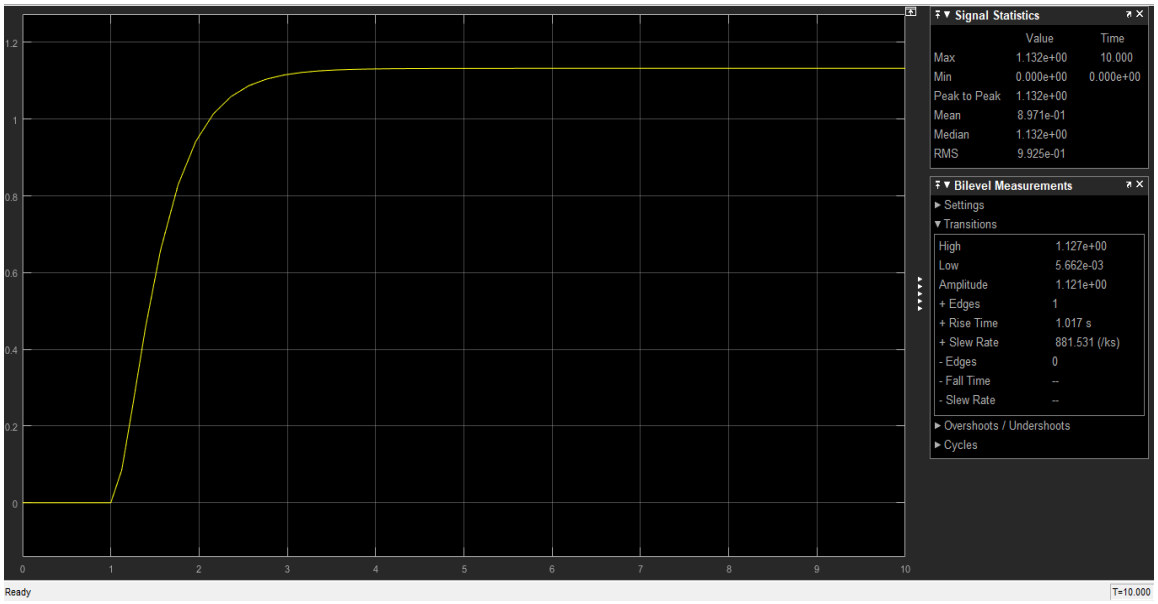


Figure 4. 5 Open loop step response

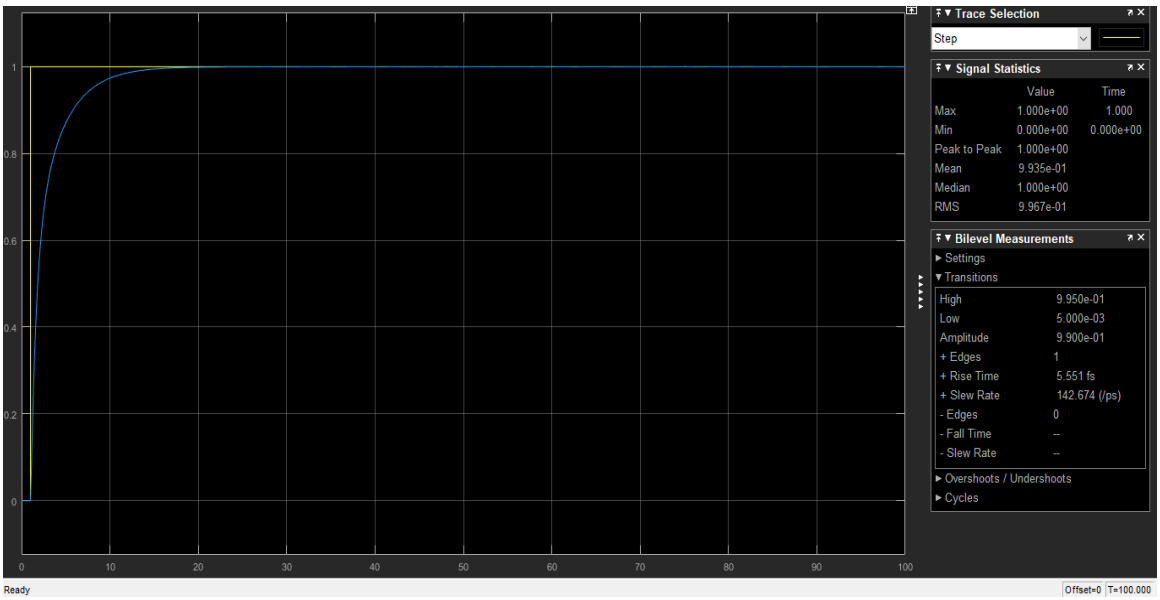


Figure 4. 6 Experimental result of PID control for Ziegler-Nichols Step Response

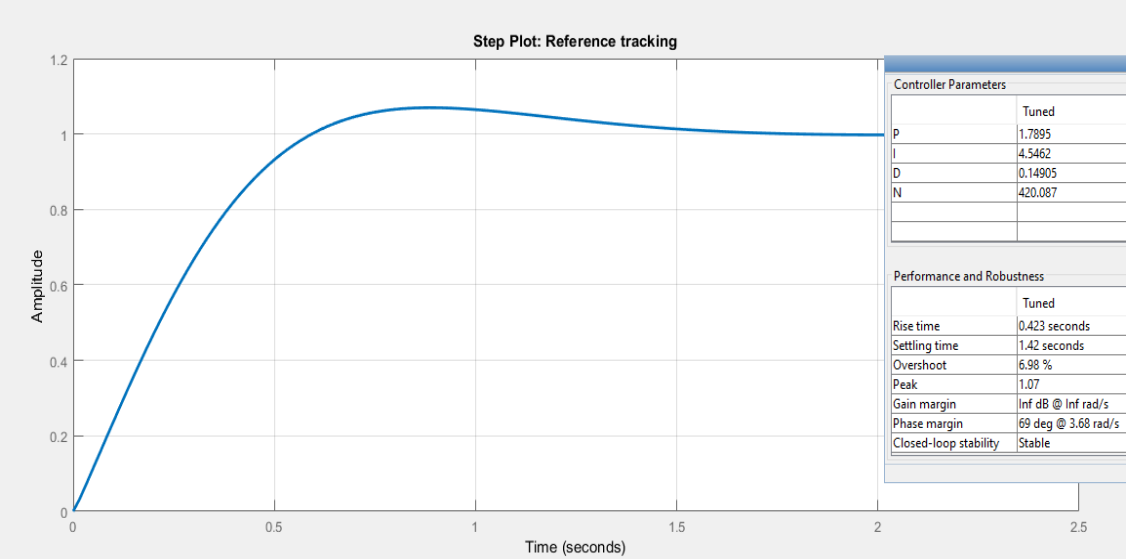


Figure 4. 7 Experimental result of matlab PID tuner Step response

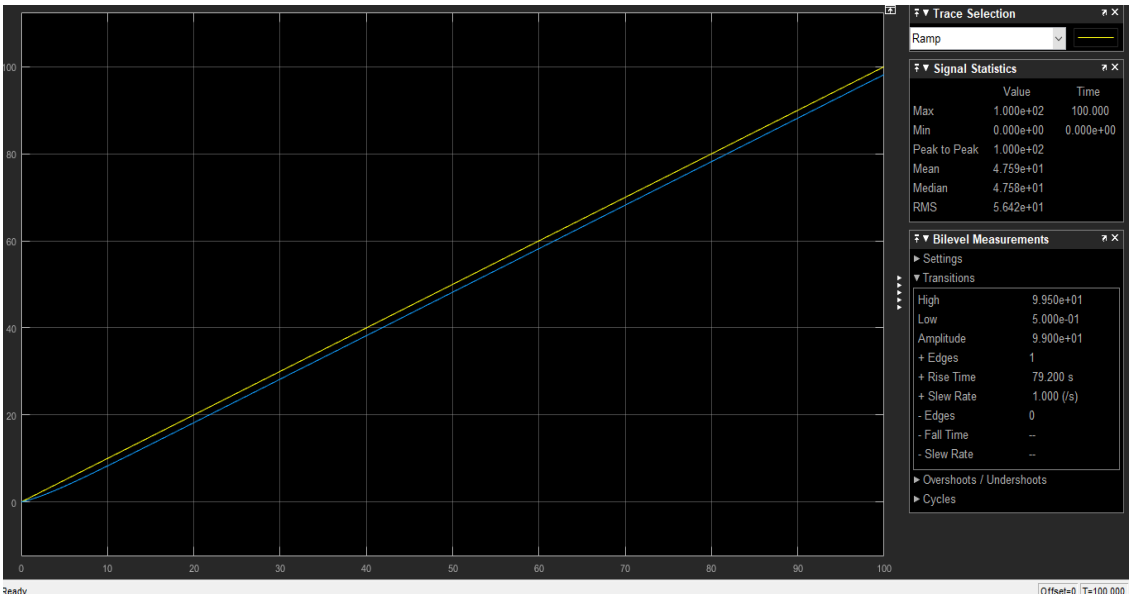


Figure 4. 8 Experimental result of PID control for Ziegler-Nichols for ramp input

In the above experimental results the speed of a DC motor with load coupled by gears is controlled using PID controller. Regarding to the tuning of PID controller Ziegler-Nichols methods has been presented. These methods are implemented using MATLAB/SIMULINK simulation. From the obtained results shown in Figure above it is concluded that Ziegler-Nichols methods gives a good performance for the system. Using the matlab PID auto tuner other PID gains are obtained that shows different performance of the system incorporating a derivative filter in the PID controller. Comparing the two results it was observed that Ziegler-Nicholas PID tuner gives good performance. In order to check the robustness of the system we also a ramp input it shows the gives a good tracking performance.

4.6 Hardware Project Results

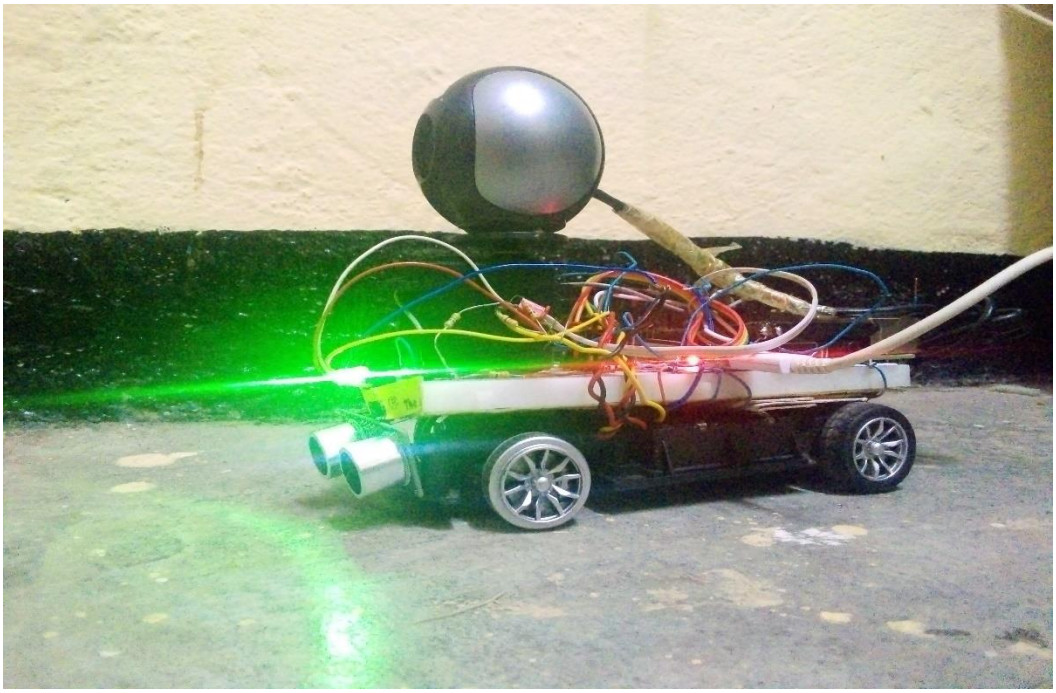


Figure 4. 9 Mobile robot prototype (a)

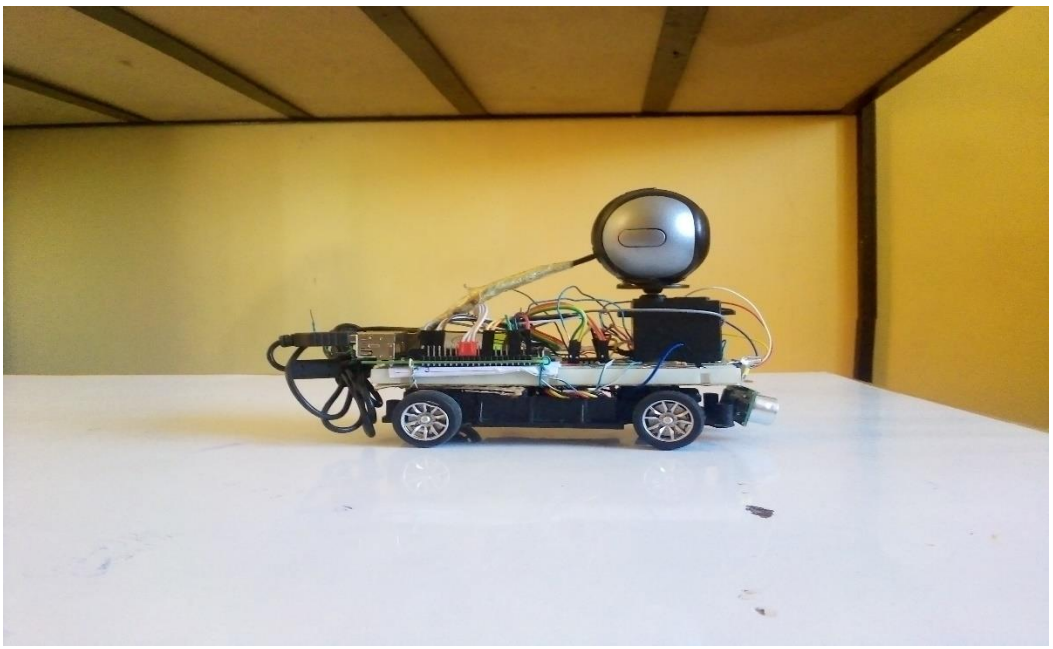


Figure 4. 10 Mobile robot prototype (b)

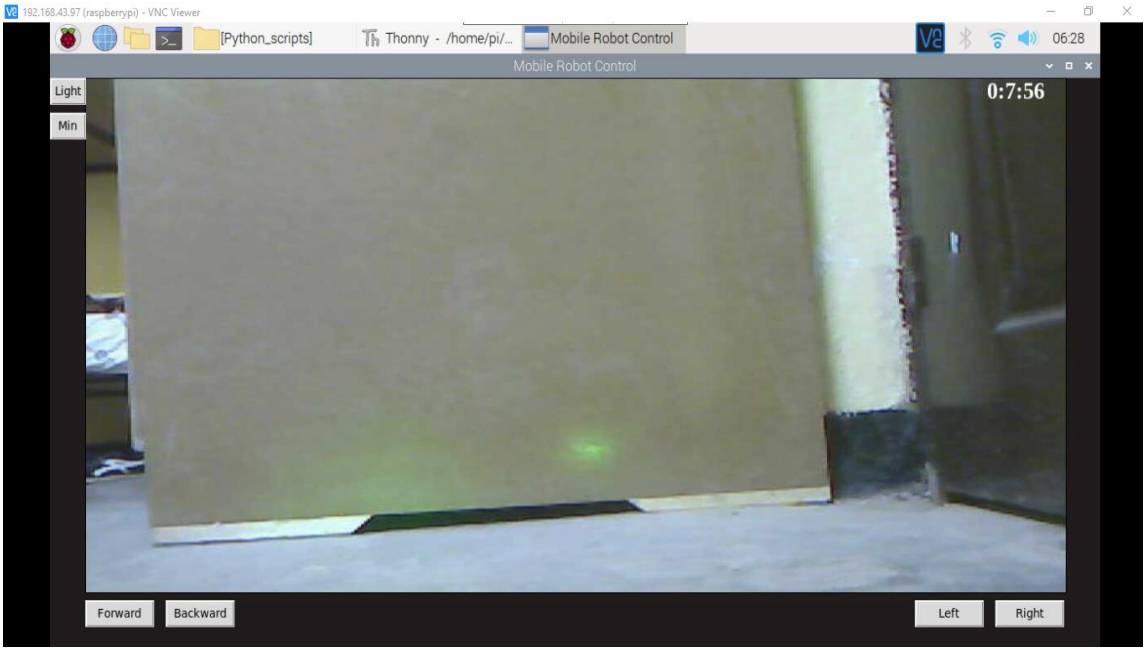


Figure 4. 11 User interface control of the mobile robot

CHAPTER FIVE

5. CONCLUSIONS AND SCOPE FOR FUTURE WORK

5.1 Conclusions

The principal goal of this capstone was to build a working prototype of a mobile robot system able to move and traverse pipes, as it provides a live streaming of the visual recording of the camera. In pipe-inspection robots are mainly differ by steering mechanism, power source and their application. Existing in-pipe inspection robots are able to move freely inside a straight pipe, elbow and T-joints of different diameter pipes. But driving mechanism of all the pipe inspection robot's is complex. So, simple driving in-pipe inspection robot has been developed. Developed in-pipe inspection robot is able to pass through straight pipe line, elbows and T-joints depending on the diameter of the pipe.

In this thesis controlling the speed of DC motor is used to control the speed of the mobile robot. The developed mobile robot can be used in Industries, gas pipe lines, sewage and other areas where pipes are used for transportation of fluids.

5.2 Scope for Future Work

Extending the camera's field of view to 360 degrees is part of future work, using one of several solutions, such as a rod being rotated by a servo motor and steadily turning the camera lens to cover a 360 degrees angle.

The mobile robot can be designed to be autonomous with a powerful computing system and sensitive sensors. So, one of the core challenges that we should be taking care of in the future implementations is to make the mobile robot intelligent enough to avoid taking the paths which are already explored, so in a way we will be saving the power for exploring the unexplored paths of the pipeline.

Many of the reviewed papers focuses around a kinematic controller and mentioned that the effects of the dynamics are unknown. Some mention with the dynamic of the system defined, better controllers can be derived around those dynamics. A more detailed dynamic analysis for future can help for better controller and mechanical design.

Future design should include issues like Mobility, Steer ability, Turning radius, Size and shape adaptability, Online adaptability, Flexibility, Stability, Autonomous operation and obstacle avoidance, Efficiency at uneven surface, Safe operation, Material selection, Type of task to be performed inside the pipe, Operation in active pipe line, Retrieval of robot, User friendly navigation and control system, Braking system, Range of operation, Quantitative analysis of defects inside the pipe.

REFERENCES

[1]. J. Okamoto, Jr., J. C. Adamowski, M. S. G. Tsuzuki, F. Buiochi, and C.S. Camerini, "Autonomous system for oil pipelines inspection," *Mechatronics*,vol. 9, pp. 731–743, 1999.

[2]. Maung Than Zaw,"Kinematic And Dynamic Analysis Of Mobile Robot" MSc thesis, national university of Singapore 2003.

[3]. Se-gonRoh and Hyouk Ryeol Choi, "Differential-Drive In-Pipe Robot for Moving Inside Urban Gas Pipelines" IEEE TRANSACTIONS ON ROBOTICS, VOL. 21, NO. 1, FEBRUARY 2005

[4]. Itseoritseagba Godwin, Donald O Ene, "Pipeline Inspection for Corrosion using a Mobile Robotic System", International Journal ofRobotic Engineering, 2015.

[5]. krzysztofkozłowski_, dariuszpazderski, "Modeling And Control Of A 4-Wheel Skid-Steering Mobile Robot", Int. J. Appl. Math. Comput. Sci., 2004, Vol. 14, No. 4, 477–496

[6]. Edouard Ivanjko, Toni Petrini, Ivan Petrovi, "Modelling Of Mobile Robot Dynamics" International Journal of Robotics and Automation

[7]. Maciej Trojnacki, "Dynamics Model of a Four-Wheeled Mobile Robot for Control Applications" Industrial Research Institute for Automation and Measurements PIAP, Warsaw, Poland, available on https://WWW.researchgate.net/publication

[8]. AfafRemani, "Crack Detection Inside Pipelines Using Mechatronics And Computer Vision" AlAkhawayn University ,University Honors Program 2018

[9]. Jiya, Anwar, N. S. N., Abdullah, and M. Z., "Detection of Cracks in Concrete Structure Using Microwave Imaging Technique," International Journal of Microwave Science and Technology, 19-Jun-2016. [Online]. Available: https://www.hindawi.com/journals/ijmst/2016/3195716/. [Accessed: 13-Nov-2020].

[10]. Mathieu Deremetz, Roland Lenain, et al; "Path Tracking Of A Four-Wheel Steering Mobile Robot: A Robust Off-Road Parallel Steering Strategy"The European Conference on Mobile Robotics, September 6-8th, 2017, Paris, France

[11]. Alexandru Bara Sanda Dale; "Dynamic Modeling and Stabilization of Wheeled Mobile Robot", 5th WSEAS Int. Conf. on DYNAMICAL SYSTEMS and CONTROL

[12]. Pourboghrat, F., and M. P. Karlsson. "Adaptive Control of Dynamic Mobile Robots with Nonholonomic Constraints" Pergamon Computers and Electrical Engineering. 28: 241- 253

[13]. G. Bayar, M. Bergerman and A. B Koku "Improving the trajectory tracking performance of autonomous orchard vehicles using wheel slip compensation" Biosystems Engineering, vol. 146, pp. 149-164.

[14]. C. Cariou, R. Lenain, B. Thuilot and M. Berducat . "Automaticguidance of a four-wheel steering mobile robot for accurate field operations" Journal of Field Robotics, vol. 26, no 6-7, pp. 504-518.

[15]. R. Lenain, B. Thuilot, C. Cariou and P. Martinet " High accuracy path tracking for vehicles in presence of sliding: Application to farm vehicle automatic guidance for agricultural tasks" Autonomous Robots, vol. 21, no. 1, pp. 79-97

[16]. Michał Ciszewski, Tomasz Buratowski, et al; "Virtual Prototyping Design And Analysis Of An In-Pipe Inspection Mobile Robot", Journal Of Theoretical And Applied Mechanics 52, 2, pp. 417-429, Warsaw 2014

[17]. Michał Ciszewski, Michał Wacławski; "Design, Modelling And Laboratory Testing Of A Pipe Inspection Robot", Archive Of Mechanical Engineering, VOL. LXII 2015 Number 3

[18]. Michał Ciszewski, Łukasz Mitka, "Modeling and Simulation of a Tracked Mobile Inspection Robot in MATLAB and V-REP Software" Journal of Automation, Mobile Robotics & Intelligent Systems, VOL 11, N° 2 ,2017

[19]. P. Sapaty " Military robotics: Latest trends and spatial grasp solutions". International Journal of Advanced Research in Artificial Intelligence, vol. 4, no 4, p. 9-18.

[20]. Jong-Hoon Kim, "Design Of A Fully Autonomous Mobile Pipeline Exploration Robot" MSc thesis, Seoul National University of Technology in Seoul, South Korea, 2008

[21]. Matthieu Lincoin Jones, "Real time pipe inspection robot prototype development", MSc thesis, Massay University, New Zealand, 2014

[22]. Raspberry Pi Cookbook for Python Programmers

[23]. Hameedah Sahib Hasan and Dr.P.RameshBabu; "Analysis And Control Of Mobile Robot For Pipe Line Inspection" International Journal of Mechanical Engineering and Technology (IJMET), Volume 4, Issue 5October 2013, pp. 01-09

[24]. Ankit Nayak, S. K. Pradhan; "Design of a New In-Pipe Inspection Robot"12th Global Congress On Manufacturing And Management, GCMM 2014

[25]. Eric N Moret; "Dynamic Modeling and Control of a Car-Like Robot", MSc thesis,Virginia Polytechnic Institute and State University February 5, 2003

[26]. Santosh Mohan Rajkumar and Sayan Chakraborty; “Development of Embedded Speed Control System for DC Servo Motor using Wireless Communication”, National Institute of Technology, Silchar December 2019

[27]. Walaa M Elsrogy, et al; “Speed Control of DC Motor Using PID Controller Based on Changed Intelligence Techniques”, International Journal of Swarm Intelligence and Evolutionary Computation, Volume 7 • Issue 1

[28]. Juan Pablo Trujillo Lemus, “PID Controller Design for DC Motor”, Contemporary Engineering Sciences, Vol. 11, 2018, no. 99, 4913 – 4920

[29]. https://datasheetspdf.com/pdf/1380136/ETC/HC-SR04/1

[30]. https://www.raspberrypi.org/documentation/remote-access/vnc/

[31]. https://en.wikipedia.org/wiki/PuTTY

APPENDIX-I

Work Plan and Budget

6.1 Work Plan

		No. of weeks from Dec. 3/2020 up to Jan. 23/2021						
	List of proposed tasks	Week 1	Week2	Week 3	Week 4	Week 5	Week 6	Week 7&8
1	Proposal submission and presentation	✓						
2	Data collection		✓					
3	Simulation with appropriate methods and software		✓	✓				
4	Components Collection	✓	✓	✓				
5	Testing circuit and recording of results				✓	✓		
6	Design appropriate improvement methods					✓	✓	
7	Compile the final paper work							✓
8	Prepare PPT and ready for presentation							✓

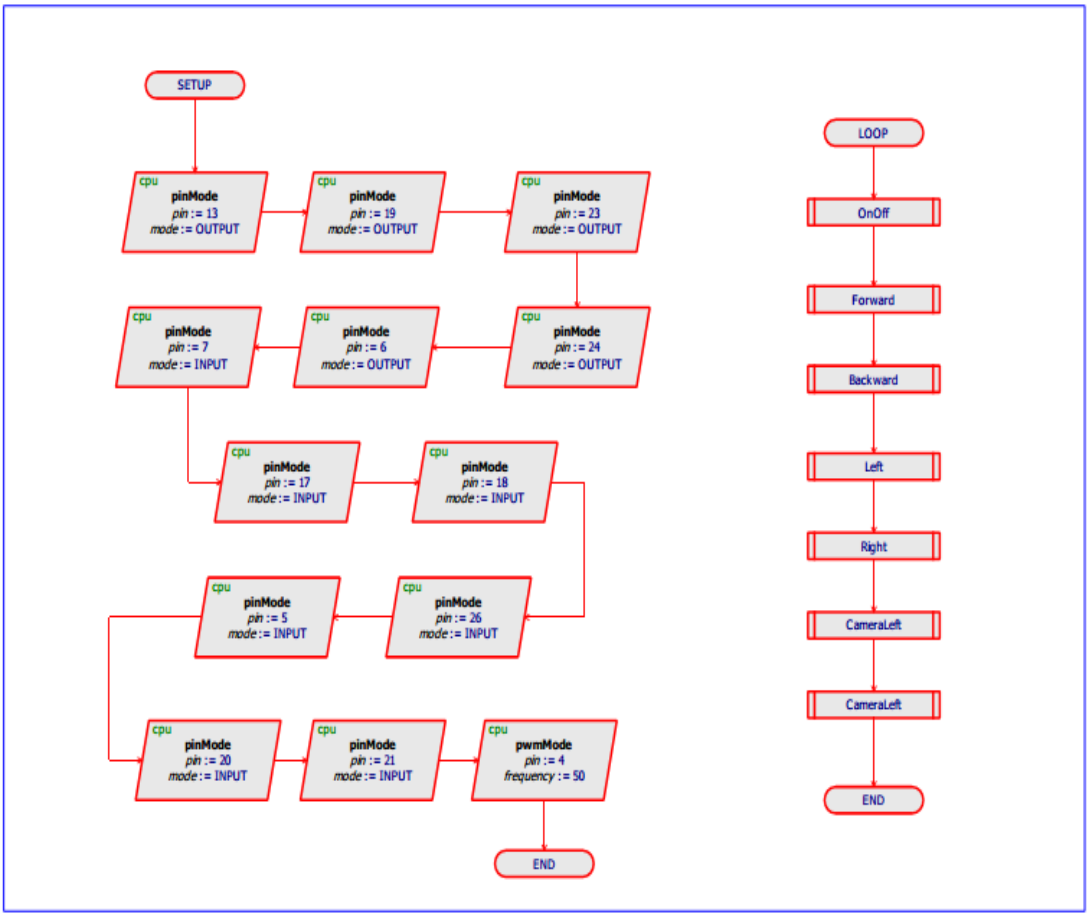
6.2 Budget

S.No	Name of Component	Quantity	Cost of each item in ETB	Total Cost in ETB
1	DC motor	2	300	300
3	Raspberry Pi	1	3000	3000
4	IC (LM293D)	1	100	100
5	Bread bored	1	100	100
6	Camera	1	1000	1000
7	Power supply	1	200	200
8	Car frame	$0.25m^2$		200
9	Glue	1	15	50
			Grand total	4950 Birr

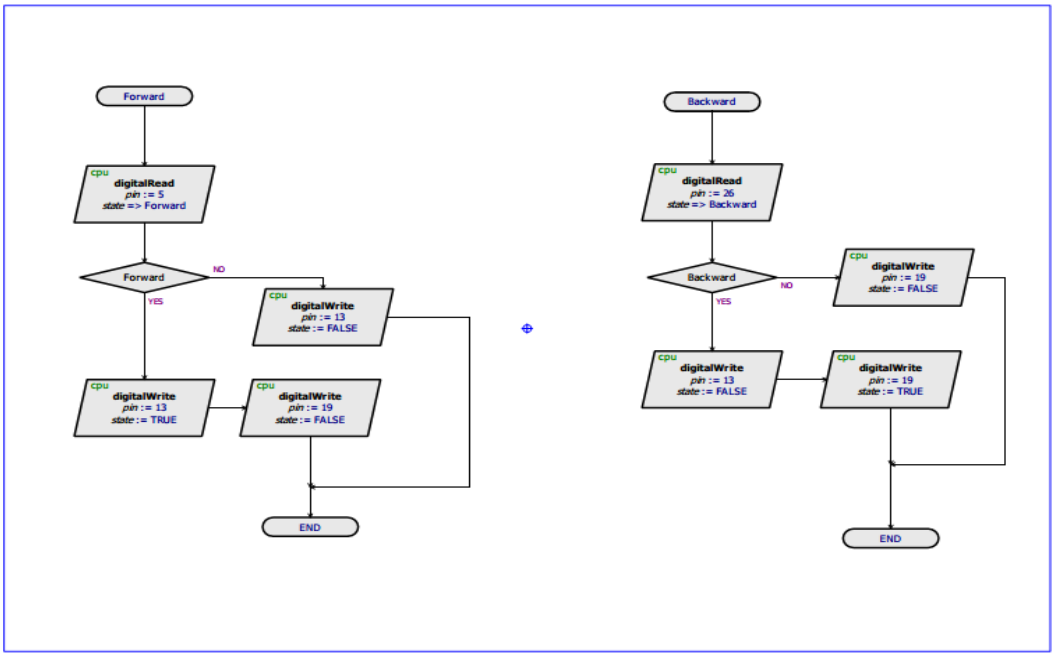
APPENDIX-II

PROJECT PROTEUS SIMULATION CODE

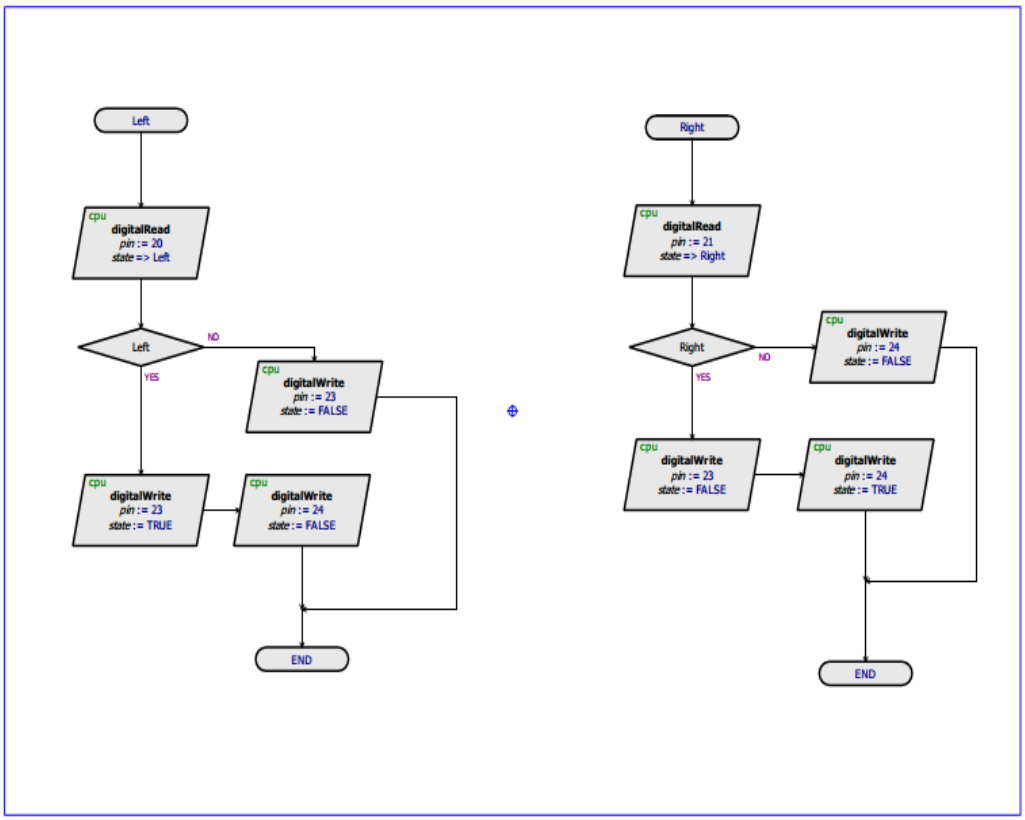
main



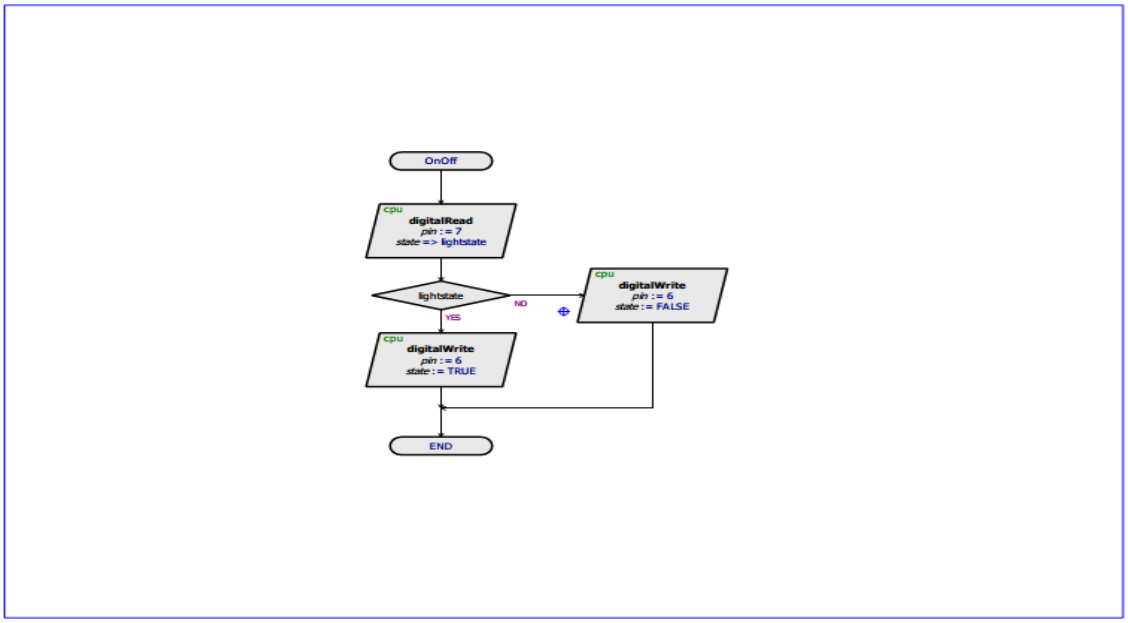
forBack



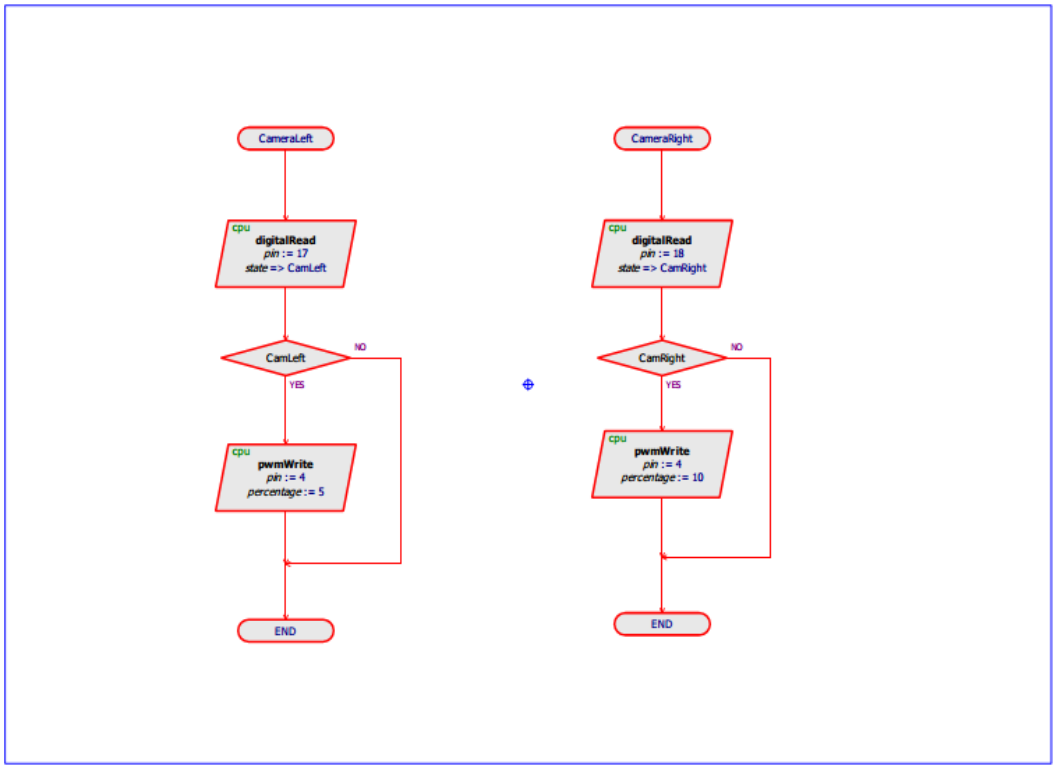
leftRight



lightState



servoControl



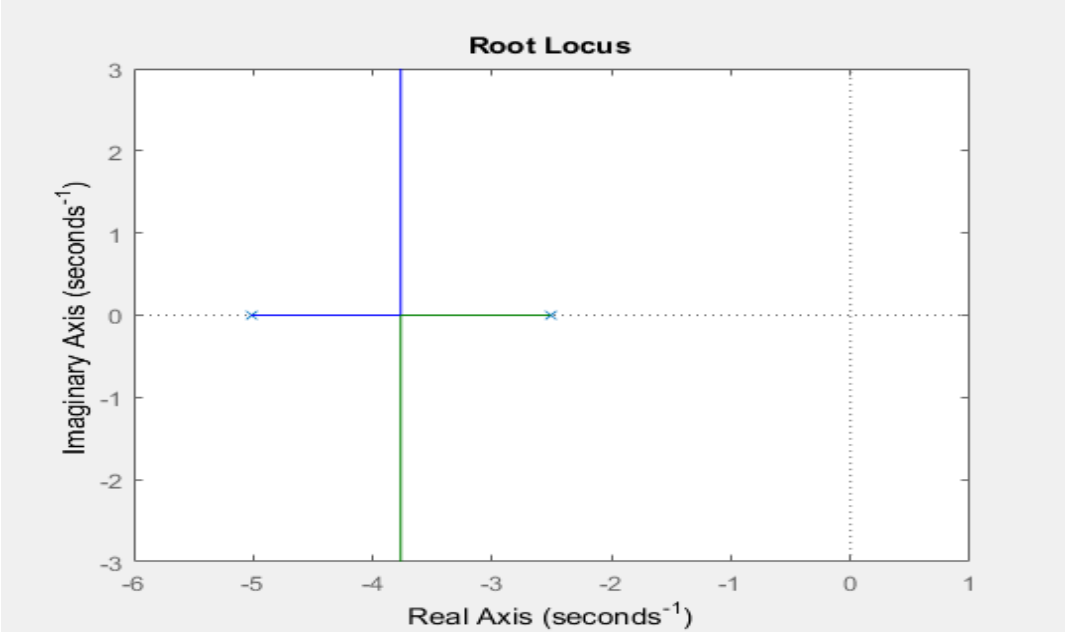
APPENDIX-III

MATLAB SIMULATION CODE

```
>> R=2.8;

>> L=0.5;

>> J=0.01;

>> b=0.024;

>> Kb=0.0235;

>> Ka=0.0153;

>> u= [L R];

>> v=[J b];

>> D=(conv(u,v)+Kb*Ka);

>> Ns=40;

>> Nm=8;

>> Rg=Ns/Nm;

>> N=Rg*Ka;

>> G=tf(N,D)
```

```
G   =          0.0765
      --------------------------------
  0.00536 s^2 + 0.04036 s + 0.06756

Continuous-time transfer function.
>> zpk(G)
ans =
          14.274
      -------------------
  (s+5.019) (s+2.512)

Continuous-time zero/pole/gain model.
```

>> rlocus(G)



>> A=[-b/J Ka/J;-Kb/L -R/L]

A =

-2.4000 1.5300

-0.0470 -5.6000

>> B=[0; 1/L]

B =

0

2

>> C=[Nm/Ns 0]

C =

0.2000 0

>> D=[0]

D = 0

>> DCmotor=ss(A,B,C,D)

DCmotor =

A =

x1 x2

x1 -2.4 1.53

x2 -0.047 -5.6

B =

u1

x1 0

x2 2

```
C =
      x1   x2
 y1  0.2    0

D =
     u1
 y1   0

Continuous-time state-space model.
```

```
>> P=ctrb(DCmotor)

P =
         0    3.0600
    2.0000  -11.2000

>> rank(P)

ans =  2

>> Q=obsv(DCmotor)

Q =
    0.2000         0
   -0.4800    0.3060

>> rank(Q)

ans= 2
```

Matlab code to find inflection point of open loop step response

```
Ts = 0.0005
Gs = tf(N,D)
Gz = c2d(Gs,Ts,'zoh')
[y,t] = step(Gz)
plot(t,y)
hold on
ypp = diff(y,2)
t_inf1 = fzero(@(T)interp1(t(2:end-1),ypp,T,'linear','extrap'),0)
y_inf1 = interp1(t,y,t_inf1,'linear')
plot(t_inf1,y_inf1,'ro')
```

```
t_inf1 = 0.2764
y_inf1 = 0.2835
```

APPENDIX-IV

PYTHON CODE

```
import tkinter
import cv2
from tkinter import Button
from PIL import Image, ImageTk
import RPi.GPIO as GPIO
import time
import threading

class App:
    def __init__(self, window, window_title, video_source=0):
        self.window = window
        self.window.title(window_title)
        self.window.config(background="#1F1B1C")
        self.window.state("normal")

        #pc screen resolution
        self.w, self.h = self.window.winfo_screenwidth(), self.window.winfo_screenheight()
        print(self.w,self.h)
        #desired screen resolution
        self.dimX,self.dimY = self.w-86,self.h-128

        #camera window size
        self.maxCameraWindow_size = False

        # open video source (by default this will try to open the computer webcam)
        self.video_source = video_source
        self.vid                                                                    =
MyVideoCapture(self.w,self.h,self.video_source,self.maxCameraWindow_size)

        #Mobile Robot motion
        self.motionStatus = ['Parking','Forward','Backward','Turning Left','Turning Right','']
```

```
self.intMotion = 0
self.light = 0

# Create a canvas that can fit the above video source size
#adjust canvas size
self.xLen = 640
self.yLen = 480
if self.maxCameraWindow_size:
    self.xLen = self.dimX
    self.yLen = self.dimY

self.canvas = tkinter.Canvas(window, width = self.xLen,
    height = self.yLen,highlightthickness=1,highlightbackground='black')
self.posx = int((self.w-self.dimX)/2)
self.xPos = int((self.w - 640) / 2)

if self.maxCameraWindow_size:
    self.xPos = self.posx
self.canvas.place(x=self.xPos,y=0)

#On screen control of the mobile robot
self.onScreenControl()
#mr Option
self.mrStatus()

#Binding key events
self.window.bind('<Left>', self.leftKey)
self.window.bind('<Right>', self.rightKey)
self.window.bind('<Up>', self.upKey)
self.window.bind('<Down>', self.downKey)
self.window.bind("<Key>", self.key_pressed)

#reset time for the keyboard
self.waitTime = 100
```

```
self.pressedTime = [0,0,0,0]

#control port for mobile robot drive
GPIO.setmode(GPIO.BCM)
self.ports = [14,23,24,13,19,26]
self.lightPort = 21
#us control pins
self.trigPin = 7
self.echoPin = 8
#servo pin
self.servoPin = 25
self.incAngle = 10

#current distance of the robot
self.distance = 0

GPIO.setup(self.lightPort,GPIO.OUT)
GPIO.setup(self.trigPin,GPIO.OUT)

GPIO.setup(self.servoPin,GPIO.OUT)
self.pwm = GPIO.PWM(25,50)
self.pwm.start(0)

GPIO.setup(self.echoPin,GPIO.IN)
for o in self.ports:
    GPIO.setup(o,GPIO.OUT)
     # After it is called once, the update method will be automatically called
      #every delay milliseconds
self.delay = 15
self.update()
self.sonic = threading.Thread(target=self.usDistance,args=[])
self.sonic.start()
self.window.mainloop()
```

```
def update(self):
    # Get a frame from the video source
    ret, frame = self.vid.get_frame()

    if ret:
        self.photo = ImageTk.PhotoImage(image = Image.fromarray(frame))
        self.canvas.create_image(0, 0, image = self.photo, anchor = tkinter.NW)
        self.mrStatus()
    self.window.after(self.delay, self.update)
    #update intMotion
    self.intMotion = 5

    for i in range(0,4):
        if time.time()-self.pressedTime [i] > 0.2:
            if i==0:
                GPIO.output(self.ports[1],0)
            if i==1:
                GPIO.output(self.ports[2],0)
            if i==2:
                GPIO.output(self.ports[4],0)
            if i==3:
                GPIO.output(self.ports[5],0)
    if self.intMotion ==0:
        GPIO.output(self.ports[1],0)
        GPIO.output(self.ports[2],0)
        GPIO.output(self.ports[4],0)
        GPIO.output(self.ports[5],0)


def onScreenControl(self):
    btn_width = len('Forward')
    yGap = 10
   self.btn_top=Button(self.window,text="Forward", width=btn_width,height=1,command=
   lambda:self.handleButton(0))
```

```
    self.btn_top.place(x=self.posx,y=self.dimY+yGap)

    self.btn_left = Button(self.window, text="Left", width=btn_width,height=1,command =
    lambda :self.handleButton(3))
    self.btn_left.place(x=self.dimX-20*btn_width,y=self.dimY+yGap)

    self.btn_right = Button(self.window, text="Right", width=btn_width,height=1,command
    = lambda :self.handleButton(4))
    self.btn_right.place(x=self.dimX-6*btn_width, y=self.dimY+yGap)

    self.btn_down=Button(self.window,text="Backward",
    width=btn_width,height=1,command = lambda :self.handleButton(2))
    self.btn_down.place(x= self.posx + 14*btn_width,y= self.dimY+yGap)

    #self.btn_light =Button(self.window,image=self.flashIco, text="Light", width=self.posx-
     5,height=20)
       self.btn_light =Button(self.window, text="Light", width=int(btn_width/2)-
       1,height=1,command = lambda :self.turnLight())
    self.btn_light.place(x= 0,y= 0)

    self.btn_cam = Button(self.window, text='max', width=int(btn_width / 2) - 1,
    height=1,command=lambda: self.cameraResolution())
    self.btn_cam.place(x=0, y=40)

def mrStatus(self):
    info = self.motionStatus[self.intMotion]
    self.canvas.create_text(2*self.posx, 15, fill="darkblue", font="Times 20 italic bold",
    text=info)

    uptime = time.process_time()
    upTime_h = int(uptime//3600)
    upTime_m = int((uptime%3600)/60)
    upTime_s = int((uptime%3600)%60)
```

```
    display_time = str(upTime_h) + ':' + str(upTime_m) + ':' + str(upTime_s)
    self.canvas.create_text(self.xLen - 60, 15, fill="white",
        font="Times 20 bold", text=display_time)

def usDistance(self):
    while True:
        #set trigger pin to high
        GPIO.output(self.trigPin,1)
        #set trigger to low after 0.01ms
        time.sleep(0.00001)
        GPIO.output(self.trigPin,0)
        startTime = time.time()
        stopTime = time.time()

        #save start time
        while GPIO.input(self.echoPin)==0:
            startTime = time.time()
        #save time of arrival
        while GPIO.input(self.echoPin)==1:
            stopTime = time.time()
        stopTime = time.time()
        duration = stopTime - startTime
        #sonic speed is 34300 cm/s
        #divide distance by 2, because it is back and forth distance of the signal
        distance = (duration * 34300) / 2
        self.distance = distance
        print(distance)
        if distance < 10:
            self.intMotion = 0
        time.sleep(1)



def setAngle(self,angle):
    duty = angle /9 + 2
```

```
    GPIO.output(25,True)
    self.pwm.ChangeDutyCycle(duty)
    GPIO.output(25,False)
    #self.pwm.ChangeDutyCycle(0)

def leftKey(self,event):
    self.intMotion = 3
    self.pressedTime [0] = time.time()
    GPIO.output(self.ports[0],1)
    GPIO.output(self.ports[1],1)
    GPIO.output(self.ports[2],0)

def rightKey(self,event):
    self.intMotion = 4
    self.pressedTime[1] = time.time()
    GPIO.output(self.ports[0],1)
    GPIO.output(self.ports[1],0)
    GPIO.output(self.ports[2],1)
def upKey(self,event):
    self.intMotion = 1
    self.pressedTime[2] = time.time()
    GPIO.output(self.ports[3],1)
    GPIO.output(self.ports[4],1)
    GPIO.output(self.ports[5],0)
def downKey(self,event):
    self.intMotion = 2
    self.pressedTime[3] = time.time()
    GPIO.output(self.ports[3],1)
    GPIO.output(self.ports[4],0)
    GPIO.output(self.ports[5],1)
def key_pressed(self,event):
    if(event.char=='s'):
        self.intMotion = 0
    if(event.char=='d'):
```

```
            self.distance = round(self.usDistance(),2)
            distStr = 'Distance: ' + str(self.distance)
            self.canvas.create_text(2 * self.posx,self.yLen - 20 , fill="darkblue",
             font="Times 20 italic bold", text=distStr)

        if(event.char=='q'):
            self.setAngle(self.incAngle)
            if self.incAngle <100:
                self.incAngle+=10

        if(event.char=='w'):
            self.setAngle(self.incAngle)
            if self.incAngle >0:
                self.incAngle-=10


    def handleButton(self,val):
        self.intMotion = val
        if val==1:
            self.intMotion = val
            self.pressedTime[2] = time.time()
            GPIO.output(self.ports[3],1)
            GPIO.output(self.ports[4],1)
            GPIO.output(self.ports[5],0)
        if val==2:
            self.intMotion = val
            self.pressedTime[3] = time.time()
            GPIO.output(self.ports[3],1)
            GPIO.output(self.ports[4],0)
            GPIO.output(self.ports[5],1)
        if val==3:
            self.intMotion = val
            self.pressedTime [0] = time.time()
            GPIO.output(self.ports[0],1)
```

```
            GPIO.output(self.ports[1],1)
            GPIO.output(self.ports[2],0)
        if val==4:
            self.intMotion = val
            self.pressedTime[1] = time.time()
            GPIO.output(self.ports[0],1)
            GPIO.output(self.ports[1],0)
            GPIO.output(self.ports[2],1)
    def turnLight(self):
        if self.light==0:
            self.light = 1
            GPIO.output(self.lightPort, self.light)
            return
        if self.light==1:
            self.light = 0
            GPIO.output(self.lightPort, self.light)
            return

    def cameraResolution(self):
        if self.maxCameraWindow_size==True:
            self.maxCameraWindow_size = False
            self.btn_cam.configure(text='Max')
            #update canvas size
            self.xLen = 640
            self.yLen = 480
            self.canvas.configure(width=self.xLen,height=self.yLen)
            self.xPos = int((self.w-self.xLen)/2)
            self.canvas.place(x=self.xPos,y=0)
            self.vid.maxCamWindow_size = False
            return
        if self.maxCameraWindow_size==False:
            self.maxCameraWindow_size = True
            self.btn_cam.configure(text='Min')
            #update canvas size
```

```
            self.xLen = self.dimX
            self.yLen = self.dimY
            self.canvas.configure(width=self.xLen,height=self.yLen)
            self.xPos = int((self.w-self.dimX)/2)
            self.canvas.place(x=self.xPos,y=0)
            self.vid.maxCamWindow_size = True
            return


class MyVideoCapture:
    def __init__(self,width=640,height=360,video_source=0,camSize=False):
        self.w = width
        self.h= height
        self.maxCamWindow_size = camSize
        # Open the video source
        self.vid = cv2.VideoCapture(video_source)
        if not self.vid.isOpened():
            raise ValueError("Unable to open video source", video_source)

        # Get video source width and height
        self.width = self.vid.get(cv2.CAP_PROP_FRAME_WIDTH)
        self.height = self.vid.get(cv2.CAP_PROP_FRAME_HEIGHT)
        print(self.width,self.height)


    def get_frame(self):
        if self.vid.isOpened():
            ret, frame = self.vid.read()
            if ret:
                # Return a boolean success flag and the current frame converted to BGR
                #frame = cv2.flip(frame,1)

                scale_percentX = (self.w/self.width)*100  # percent of original size
                scale_percentY = (self.h / self.height) * 100
```

```
            #maximize camera window size
            if self.maxCamWindow_size:
                width = int(frame.shape[1] * scale_percentX / 100)
                height = int(frame.shape[0]*scale_percentY / 100)
            else:
                #maintain original camera resolution
                width = int(frame.shape[1])
                height = int(frame.shape[0])

            dim = (width, height)
            resizedFrame = cv2.resize(frame, dim, interpolation = cv2.INTER_AREA)
            return (ret, cv2.cvtColor(resizedFrame, cv2.COLOR_BGR2RGB))
        else:
            return (ret, None)
    else:
        return (ret, None)

# Release the video source when the object is destroyed
def __del__(self):
    if self.vid.isOpened():
        self.vid.release()
        GPIO.cleanup()

# Create a window and pass it to the Application object
App(tkinter.Tk(), "Mobile Robot Control")
```