H-INFINITY CONTROL OF AN AUTONOMOUS MOBILE ROBOT

NUHA NAWASH

Bachelor of Science in Electrical Engineering
Cleveland State Univeristy
May, 2001

Submitted in partial fulfillment of requirements for the degree
MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

at the

CLEVELAND STATE UNIVEISTY

May, 2005

This thesis has been approved for the Department of

ELECTRICAL AND COMPUTER ENGINEERING

And the College of Graduate Studies by

_____

Thesis Committee Chairperson, Dr. Dan Simon

Electrical and Computer Engineering Department

_____

Thesis Committee Member, Dr. Ana Stankovic

Electrical and Computer Engineering Department

_____

Thesis Committee Member, Dr. Paul P. Lin

Mechanical Engineering Department

DEDICATION

This thesis is dedicated to my father, Khaled Abed who always want to see his little girl in college but unfortunately fate took him away before he had a chance to see that, and to my brother Ibrahim whose dream was to go to college himself, but due to uncontrollable circumstances he was born in an environment where dreams are not meant to come true and his life was very short.

To my mother, where there is no words in any language to thank her enough for what she did for me, her strength, beliefs, teachings still in me what makes me now.

To my husband Saleh, and my children Ibaa, Hashem, Ahmad and Baraa, for their support and patience through these years. It is their anticipation and encouragement to push me day after day to finish this thesis.

Finally, to every Palestinian woman whose dream is to get her education and empower women all over the world.

ACKNOWLEDGMENT

First of all, I want to express my sincere appreciation for my advisor, Dr. Dan Simon for his patience and kindness for allowing me to be a member of embedded control systems research lab (ECSRL) and for his supervision and support through this thesis.

Many thanks go to Dr. Ana Stankovic, a committee member, for her encouragement me to finish this thesis, and Dr. Paul P. Lin for his time reading and evaluating my thesis.

I want to thank all my teachers of Electrical and Computer Engineering who keep encourage me to finish this thesis especially a very dear teachers to my heart Dr. George Kramerich and Dr. F. Eugenio Villaseca. Special thanks to the secretaries of Electrical and Computer Engineering Adrienne Fox, and Jan Basch.

I would like to acknowledge a special teacher Professor Robert Mikel, Technology Department of Cleveland State University, who transferred a lot of his expert knowledge to me, and who welcomed me on his class any time. Professor Mikel, God bless your heart, I learned a lot from you.

H-INFINITY CONTROL OF AN AUTONOMOUS MOBILE ROBOT

NUHA NAWASH

ABSTRACT

This thesis proposes a robust trajectory-tracking solution for a two-wheeled mobile robot using H-infinity (minmax) control techniques in the presence of uncertainties that arise from neglecting some of the system dynamics (e.g., motor dynamics, sensor noise, and unmodeled vehicle dynamics). To compensate for the uncertainties in the dynamic model, this thesis illustrates how the kinematics model of the system can be used to design a robust minmax controller and compare it with a P controller and a PI controller. The nonlinear and the linearized models are simulated in MATLAB® and the results are presented to demonstrate the performance of the proposed controllers. The difference between the advanced control and the traditional control are explained. The second part of this thesis discusses the construction of the mobile robot system, which is an embedded system. It integrates hardware and software in its design and operation. Hardware includes the physical parts of the system and software includes the programs that determine the robot operation. A Microchip PIC16F877 microcontroller (programmed in assembly language) interfaces and controls the mobile robot devices.

Table of Contents

List of Figures ix

List of Tables xii

CHAPTER I- INTRODUCTION 1

1.1 History of Robotics Research 2

1.2 Literature Review on Mobile Robot Control 5

1.3 Motivation and Thesis Organization 8

CHAPTER II-SYSTEM MODELING 10

2.1 Mathematical Model Formulation 11

2.2 Linearization 15

CHAPTER III-APPLYING CONTROL SCHEMES TO THE MOBILE ROBOT 19

3.1 H-Infinity Control Schemes 20

3.1.1 The Output Feedback Control 22

3.1.2 The Full Information Estimator 28

3.2 H-Infinity Controller Implementation 30

3.3 Simulation results of H-infinity control 32

3.3.1 Simulation Terms 33

3.3.2 The Azimuth Parameter 41

3.4 P and PI Controller Implementation 42

3.4.1 Simulation Results of the P Controller 43

3.4.2 Simulation Results of PI Controller 47

3.5 Differences between the H-infinity and P and PI Controllers 56

CHAPTER IV- MOBILE ROBOT CONSTRUCTION 58

4.1 Microcontroller PIC 16F877 59

4.2 Stepper Motors Circuit 63

4.3 Ultrasonic Sensor 72

4.4 The LCD 0821 Display 76

4.5 The Mobile Robot System (ROCK) 80

4.5.1 The Software behind ROCK 82

4.5.2 Problems on the Mobile Robot Design 84

CHAPTER V-CONCLUDING REMARKS 87

5.1 conclusion 87

5.2 Future Work 88

BIBLIOGRAPHY 90

APPENDICES 94

A MATLAB Code for H-Infinity Controller 95

B MATLB Code for PI Controller..... **...99**

C PIC 16F877 Code (Assembly Program)..... **.....102**

List of Figures

Figure1: Classical Control Configuration 18

Figure 2: The Standard form for Robustness Analysis 19

Figure3: The Model of the Mobile Robot 22

Figure 4: Active–Driving Wheel in Two Dimensions 24

Figure 5: Feasible region of Translational Velocity 24

Figure 6: The Estimation Problem in Standard Form 32

Figure 7: Angle Mobile Robot-Gamma is 50 34

Figure 8: Y-Position of Mobile Robot-Gamma is 8 35

Figure 9: Y-Position of Mobile Robot-Gamma is 50 36

Figure 10: Control Signal for Left Wheel When Gamma is 50 37

Figure 11: Control Signal for Right Wheel when Gamma is 50 37

Figure12: Control Signal for The Right Wheel when Gamma is 10,000 38

Figure 13: Control Signal for The Left Wheel when Gamma is 10,000 39

Figure 14: States of the Mobile Robot System –Gamma is 50 39

Figure15: Traditional Planner with PIC as Controller 42

Figure 16: Simulink Diagram for P Controller 44

Figure 17: The Reference and X-Position for P Controller 45

Figure 18: The Y- Position for P Controller 46

Figure19: Theta Position for P controller 46

Figure 20: Control Signals for P Controllers 47

Figure 21: Simulink Diagram for PI Controller 48

Figure 22: Simulink Diagram for the Integral Part of PI Controller 49

Figure 23: X-Position for PI Controller 49

Figure 24: Y-Position when the K_p Gain 10 and KI Gain is .1 50

Figure 25: Theta for PI Controller 50

Figure26: X-Position for the PI Controller the K_p Gain 10 and KI Gain 0.1 51

Figure 27: X-Position for the PI Controller the K_p Gain 100 and KI Gain 1 52

Figure 28: Theta PI Controller the K_p Gain 50 and KI Gain 1 53

Figure 29: Y-Position under PI Controller 54

Figure 30: X-Profile for X-Position 55

Figure 31: X-Position Controlled by PI Controller 55

Figure 32: Control Signals for PI Controller 56

Figure 33: Top View for PIC 16F877 60

Figure 34: PIC 16F877 Chip Package 61

Figure 35: Mobile Robot Setup for Programming 62

Figure 36: Mobile Robot Stepper Motor 65

Figure 37: Unipolar Stepper Motor 66

Figure 38: UCN5804 Stepper Motor Driver 67

Figure 39: Stepper Motor Translator/Driver 68

Figure 40: Timing Pulses for Different Modes 69

Figure 41: Stepper Motor Circuit 70

Figure 42: Timer 555 Circuit 71

Figure 43: The Ping Wave for Ultrasonic Sensor 72

Figure 44: Ultrasonic Sensor Transducers 73

Figure 45: The Pin Connections 73

Figure46: The Ultrasonic Sensor Circuit ... 74

Figure 47: Timing Specification ... 75

Figure48: PCB for the LCD connection Pins ... 77

Figure 49: Front Image for theLCD0821 ... 77

Figure 50: Max202 Pin Configuration and the Operating Circuit ... 78

Figure 51: LCD display Circuit ... 79

Figure 52: Hardware of Mobile Robot ... 82

Figure 53: Flowchart of Mobile Robot Software ... 83

Figure 54: Diagram of Mobile Robot System ... 85

List of Tables

Table I: The Simulation Results for Mobile Robot System 35

Table II: States of Mobile Robot System 40

Table III: The D Parameter Affects the Control Signals 41

Table IV: PI controller simulation results for mobile robot system 52

Table V: Comparison between Control Signals for PI, H-infinity controllers 57

Table VI: PIC 16F877 Features 61

Table VII: Two-Phase Driver Sequence 68

Table VIII: Half-Step Drive Sequence 69

Table IX: Mobile Robot Cost 81

CHAPTER I

INTRODUCTION

Wheeled mobile robots are becoming increasingly important in industry as a means of transport, inspection, and operation because of their efficiency and flexibility. In addition, mobile robots are useful for intervention in hostile environments for performing tasks such as handling solid radioactive waste, decontaminating nuclear reactors, handling filters, patrolling buildings, minesweeping, etc. Furthermore, mobile robots can serve as a test platform for a variety of experiments in sensing the environment and making intelligent choices in response to it.

In general, robotics is about building systems. Locomotion actuators, manipulators, control systems, sensor suites, efficient power supplies, well-engineered software – all of these subsystems have to be designed to fit together to create the whole system. For that reason, a roboticist must be a generalist. Building a robot requires expertise beyond simply programming. The robot designer must own a compendium of basic skills from fields such as mechanical engineering, electrical engineering, computer science, and artificial intelligence. In this chapter, the history of robotics research discussed in Section 1.1. Section 1.2 reviews the literature on mobile robot control. Section 1.3 explains the motivations behind this thesis and its organization.

1.1 **History of Robotics Research**

Before we discuss the history of robotics research, we need to know what a robot actually is and where robots come from. If we look at the Oxford American Dictionary, it defines a robot as: "A machine capable of carrying out a complex series of actions automatically,

especially one programmed by a computer." The American Heritage Dictionary has this definition: "An externally manlike mechanical device capable of performing human tasks or behaving in a human manner." The question here is what does the robotics industry have to say on this subject? The definition presented by the Robot Institute of America is given as follows [7]: "A reprogrammable, multifunctional manipulator designed to move materials, parts or specialized devices through variable programmed motions for the performance of a variety of tasks." Although this definition focuses on industrial robots, it is still widely influenced by many dictionary definitions. The Japanese Industrial Robot Association (JIRA) is also concerned with robots and robotics, but it creates a whole robot classification system. It includes manually operated manipulators, sequential manipulators, numerically controlled robots, sensate robots, adaptive robots, smart robots and intelligent mechatronic systems [21]. The Japanese are certainly on the right track for defining each robot according to its functionality and the tasks that it does.

The term robot comes to us from the Czech word robota, which means forced labor or subservient labor [24]. In Czech, a robotnik is a peasant or serf. The term was first introduced in Karel Capek's play R.U.R. (Rossum's Universal Robots). Capek wrote R. U. R. in 1920, and it premiered in Prague in 1921. The play was introduced to the West when it was performed in New York in 1922, and subsequently in England in 1923. R. U. R. was controversial and widely debated in intellectual circles, and the term robot quickly replaced the earlier term automaton.

Robotics is closely tied to the continuing development of technology on a broad front in such diverse fields as materials, mechanics, controls, and computers. Although one can list some important events in a more or less historical line, the computer invention was the

real revolution in the robotics world. The key to diversification and extension in manufacturing has been the computer, with its capability to organize information and eventually perform automated processes. The first general-purpose digital computer was introduced at the Massachusetts Institute of Technology (M.I.T.) [17].

Mobile robotics development started at Stanford University when Nils Nilsson [9] developed the mobile robot SHAKEY in 1969. This robot possessed a visual range finder, camera and binary tactile sensors. It was the first mobile robot to use artificial intelligence to control its actions. Its main objective was to navigate through highly structured environments such as office buildings. The JPL lunar rover [11], developed in the 1970s at the Jet Propulsion Laboratory, was designed for planetary exploration. Using a TV camera, laser range finder and tactile sensors, the robot categorized its environment as traversable, not traversable or unknown. In the late of 1970s Hans Moravec [12] developed CART in the Artificial Intelligence Laboratory at Stanford. The robot was capable of following a white line on a road. A television camera mounted on a rail on the top of CART took pictures from several different angles and relayed them to a computer, which performed obstacle avoidance by gauging the distance between CART and obstacles in its path. In 1994, the CMU Robotics Institute's DanteII [13], a six-legged walking robot, explored the Mt. Spurr Volcano in Alaska to sample Volcano gases. In 1997 NASA's Mars Pathfinder delivered the Sojourner Rover [15] to Mars. Sojourner sent back images of its travels on the distant planet. In the same year, Honda showcased the P3 [16], an extraordinary prototype in humanoid robotic design. In 1999, Sony began selling its AIBO robotic pet [17]. The product is almost affordable and extremely sophisticated. Anti-cutesy Web site critics the world over shudder at the thought of personal AIBO pet pages (there are now hundreds). In 2000, Battlebots

premiered on Comedy Central [18]. Gearheads all over America headed to the garage to cannibalize the family mower.

Robots are now expected to become the next generation of rehabilitation assistance for elderly and disabled people, and one of the researched areas in assistive technology is the development of intelligent wheelchairs. By integrating an intelligent machine with a powered wheelchair, a robotic wheelchair has the ability to safely transport the user to a destination. Bipedal humanoid robots are currently being manufactured in Japan and the robots themselves are very complicated to build in Japan. Japanese engineers are trying to complete such a mechanical marvel by 2008. Can scientists make robots so advanced that imitate humans in their functionalities? Actually it is difficult but not impossible; because scientists and engineers are working together in order to achieve this monumental goal.

In the last few decades, looking to biology and nature for design inspiration (called biomimicry) has had a huge impact on robotics R&D. This has given rise to behavior-based robotics (BBR), a scheme for creating robots that react directly to their environment (sense-act) rather than building maps of it first (sense-plan-act). One of prominent scientist in this area is Professor Vladimir J. Lumelsky who tries to "learn robot" how to sense and react to their surroundings [45]. Robot competitions from wrestling robots, to combat robots, to robot "Olympics" are becoming a major way in which robots "evolve." As builder Mark Tiden stated in Chapter 2 of his book Robot Evolution [26], "A human is a way in which a robot builds a better robot." Just as nearly every robot engineer and pundit has a slightly different theory about which robot evolutionary path will be taken in the end, they also have different takes on where and how the first permanent settlement of robo-kind will be established. On the battlefield, in the corporate office, in every nook and cranny where

humans fear to tread, in the gladiator arena, and in our living rooms, we will see some kind of robotic involvement.

1.2 Literature Review on Mobile Robot Control

Voluminous literature exists on the subject of mobile robotics motion control. It is a heavily researched area due to both the challenging theoretical nature of the problem and its practical importance. Previous work in autonomous mobile robot control generally involves path planning and path tracking control. It can be classified into the following four categories: linear [5, 8], nonlinear [1, 2, 3, 4], geometrical [12, 13], and intelligent [13] approaches. Many methods, however, require too much computational effort to be implemented in real time, such as sliding control, nonlinear steering control, and path tracking based on a Lyapunov function. The path error in the robot navigation depends on the smoothness of the reference path given in the planning stage. The various path planning methods using a smooth curve with curvature continuity have been addressed in several previous works [1, 3, 6, 8, 13, 14, 16]. These works include the clothoid pair method, polynomial curve method, and time-optimal trajectory planning.

At the beginning of the new millennium the PID controller continues to be the key component of industrial control, including robotics control [25]. During this century many different structures of control have been proposed to overcome the limitations of PID controllers. Because of their simplicity and usefulness, they are an important part of the industrial process. The present- day structure of P, PI, and PID controllers is quite different from the original P, PI, PID controllers. Now the implementation of P, PI, and PID controllers are based on digital design. These digital controllers include many algorithms

such as anti-windup, auto-tuning, adaptive, and fuzzy fine tuning to improve their performances, but the basic actions remain the same [12]. During the last two decades, the general reluctance of researchers to use PID controllers has begun to disappear [13]. Many of the new capabilities of digital PID controllers have been introduced by the research community, the industrial control users apply these innovations easily (even enthusiastically), and PID control has become one of the most important ways for scientific specialists in control and users of industrial control to work together [16]. P, PI, and PID controllers are considered classical (conventional) control which is concerned with single input and single output (SISO). It is largely based on Laplace transform theory and its use in system representation in block diagram form.

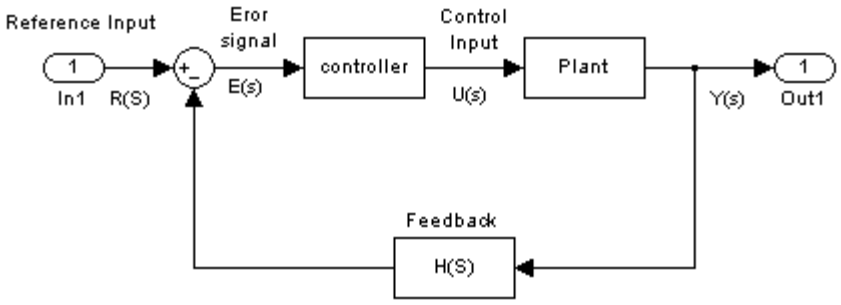


Figure 1: Classical Control Configuration

From Figure 1, we see the reference input *R(s)*, the input *U(s)*, and the output *Y(s)*, where s is the Laplace variable. Notice that the control signal is determined by the error signal and the controller. All the variables are not readily available for feedback; in many cases only one output variable is available for feedback.

In the past two decades there have been great advances in the theory of modern control. The design of robust uncertainty-tolerant multivariable feedback control systems has

been resolved, at least partially [11, 12]. Many of the questions that created the gap of the 1970's between the theory and practice of control design was the concern of the control theorists, such as stability margins, sensitivity, disturbance attenuation and so forth. Out of this renewed concern has emerged the singular value as a key indicator of multivariable feedback system performance [11, 12]. The singular value thus joins such previously used measures of multivariable feedback system performance as dominant pole locations (related to disturbance rejection bandwidth and transient response). The real problem in robust multivariable feedback control system is to synthesize a control law which maintains system response and error signals to within prespecified tolerances despite the effects of uncertainty on the system. Uncertainty may take many forms but among the most significant are noise, disturbance signals and transfer function modeling errors. Another source of uncertainty is unmodeled nonlinear distortion. Consequently people have adopted a standard quantitative measure for the size of the uncertainty, called the H-infinity norm. More detail will be given in Chapter 3.

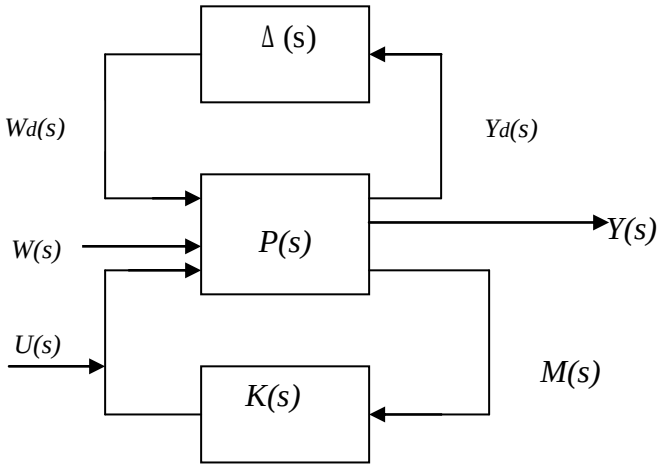


Figure 2: The Standard form for Robustness Analysis

This common framework has the perturbation normalized and in the feedback loop, as shown in Figure 2. The plant *P(s)* has three inputs and three outputs (in general each of these inputs and outputs can be a vecotor). The inputs consists of the perturbation $W_d(s)$, input, the disturbance input *W(s)*, and the control input *U(s)*. The outputs consist of the output perturbation $Y_d(s)$, the reference output *Y(s)*, and the measured output *M(s)*.

1.3 Motivation and Thesis Organization

The purpose of this thesis is two-fold. The first purpose is to provide a practical approach for robotic technology development and explore the basic principles of sensory systems, data acquisition systems, and actuation systems. The second purpose is to apply modern control technologies such control, and compare it to traditional controllers such as proportional and proportional integral controllers.

This thesis focuses on path following algorithms for a mobile robot with velocity constraints on the wheels. The path that the mobile robot follows is a straight line in an indoor environment, such as maneuvering down a hallway. This thesis is divided into two parts. The first part is the theoretical part where the nonlinear and the linear systems were modeled and simulated with MATLAB®. An H-infinity, *P*, and *PI* controller are simulated and compared. The second part is the practical part where a mobile robot is built as an embedded system, which includes integration of hardware and software in its design and operation. The hardware includes the physical parts of the system and the software includes the programs (set of instructions) that determine the robot operation. Then a closed loop controller is implemented to study the behavior of the mobile robot and compare it with the

theoretical results. The main purpose of this thesis is to gain practical experience with control schemes and embedded systems interfaces.

The thesis is organized as follows. The history of mobile robotics and the motivation of this thesis are introduced in this chapter. In Chapter II, the mathematical model for the nonlinear and linearized system will be introduced. Chapter III discusses the control schemes of the mobile robot, simulation results and the difference between the H-infinity, P, and PI controllers. Chapter IV explains the hardware design and the electrical and mechanical parts of the mobile robot, and how it is controlled by the microcontroller PIC16F877. Conclusions and future research are discussed in Chapter V.

CHAPTER II

SYSTEM MODELING

The mobile robot was modeled as a rigid body that satisfies a nonholonomic constraint, which means the motion of the system is not completely free. Nonholonomic systems are systems in which the instantaneous velocities of system components are restricted, thereby limiting the local movement of the system. This means, for example, that the mobile robot cannot move sideways. Common examples of vehicles with nonholonomic motion constraints are automobiles and vehicles with trailers. Parallel parking is a familiar illustration of the type of difficulty associated with even this simple path planning problem. Other contexts in which nonholonomic constraints occur include when there is a rolling contact, such as with a fingered hand on a surface, or when conservation of angular momentum is a significant factor, as in the case of free-flying robots. Nonholonomic systems are not locally controllable, yet they are in many cases globally controllable. In this chapter, the mathematical model for the nonlinear system of the mobile robot will be explained in Section 2.1. The linearized robot system will be introduced in section 2.2.

This thesis is addressed the kinematics model of the mechanical system of a two-wheeled mobile robot. The kinematics model of the robot has been chosen because it deals with the geometry of robot motion with respect to a fixed reference coordinate frame as a function of time, without regard to the forces and moments that cause the motion. This leads us to achieve a state feedback control that yields a steady and stable wall following trajectory in straight-line segments to meet the kinematics constraints.

2.1 Mathematical Model Formulation

Our system is a two-wheeled mobile robot whose position is defined by three coordinates in a plane: positions x and y, and robot heading angle in an absolute frame. The robot can move along the path that originates from the current posture (x, y,) in the configuration space. Figure 3 [36] below shows a two-wheeled robot, where C is the center of motion of the mobile robot. The center of gravity of the platform is at the origin (o), (V_x, V_y) represents linear speed or tangential velocity, and w is the angular velocity.

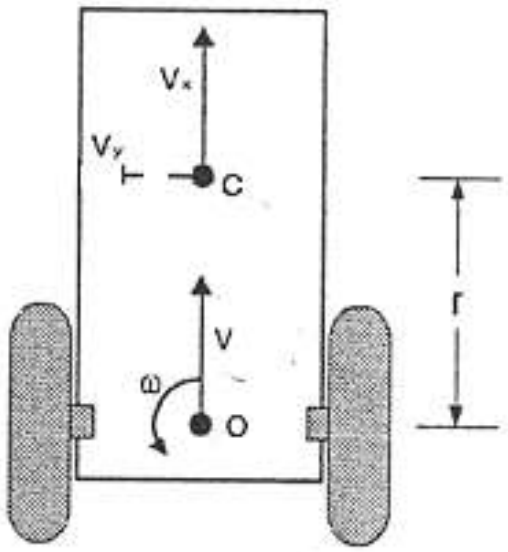
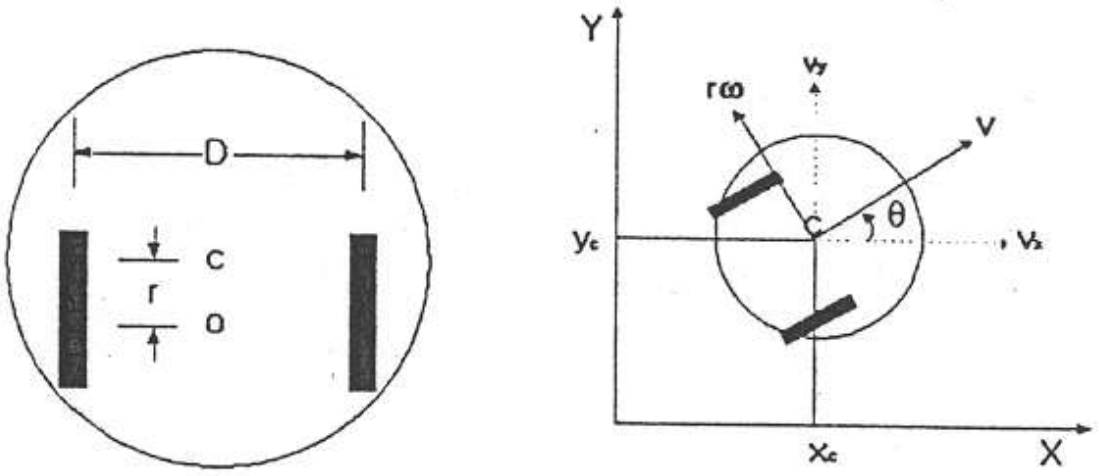


Figure 3: The Model of the Mobile Robot

Figure 4A represents the active–driving wheel in two dimensions where *r* represents the radius of the wheels, and *D* the azimuth length between the wheels [36]. *C* is the center of motion of the mobile robot. Figure 4B [36] represents the (V_x, V_y) absolute (Cartesian) coordinate system of linear speed with the center of the platform at the origin (o), and w is the angular velocity. Theta ( ) is the heading angle of the turn in radians. This system is represented in x and y

coordinate (and orientation) change with respect to time. At any instant, the x the y coordinates of the robot's center point are changing based on its speed and orientation. Although the states change with time, the physical laws that govern the behavior of the mobile robot do not change with time, which means the system is time invariant.



A **B**

Figure 4: Active–Driving Wheel in Two Dimensions

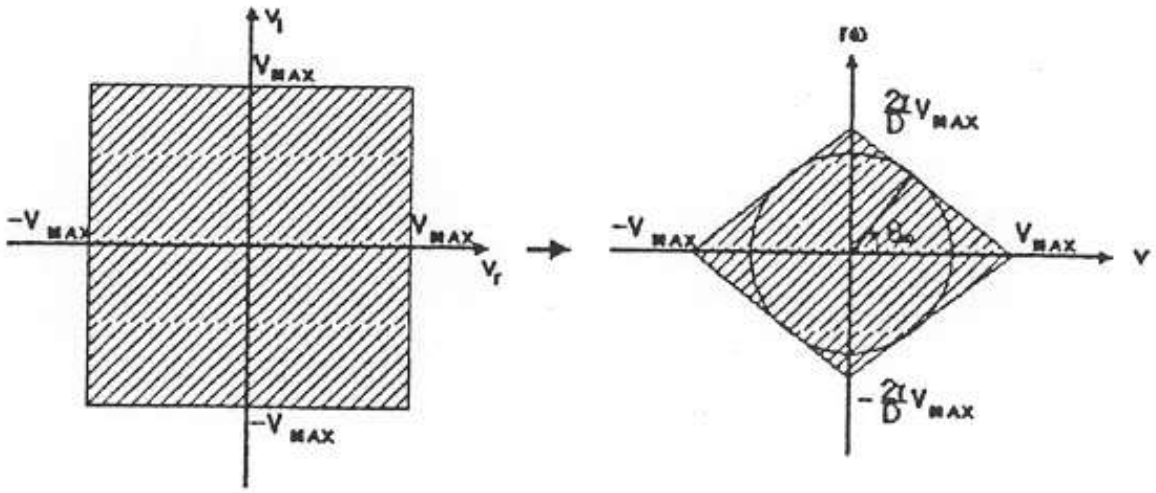


Figure 5: Feasible region of translational velocity

Figure 5 represents the nonholonomic constraints which are limited to the shaded region [36]. From the global translational velocity factors V_x, V_y, which they are rotational transformation of V, that lead to set the conditions of linearization of the system. More details will be given in Section 2.2. The mathematical equations behind our system model as represented in Figures 4 and 5 are as follows:

$$V \quad \sqrt{(V_x^2 \quad V_y^2)} \tag{2.1}$$

$$Vx \quad V_C \cos \tag{2.2}$$

$$Vy \quad V_C \sin \tag{2.3}$$

$$\dot{X}_C \quad V_x \tag{2.4}$$

$$\dot{Y}_C \quad V_y \tag{2.5}$$

$$C \quad \tan^{1} \frac{V_y}{V_x} \tag{2.6}$$

$$W_C \quad \frac{r_w}{D(W_R \quad W_L)} \tag{2.7}$$

$$V_C \quad \frac{r_w}{2(W_R \quad W_L)} \tag{2.8}$$

$$\dot{X} \quad V_C \cos \quad \frac{r_w}{2(W_R \quad W_L)} \cos \tag{2.9}$$

$$\dot{Y}_C \quad V_C \sin \quad \frac{r_w}{2(W_R \quad W_L)} \sin \tag{2.10}$$

$$W_C \quad \frac{r_w}{D(W_R \quad W_L)} \tag{2.11}$$

where:

W_R = angular velocity for right wheel

W_L = angular velocity for left wheel.

D = azimuth length between the wheels.

r_w = radius of the wheels.

V_C = tangential velocity, or linear velocity

W_C = angular velocity, or steering velocity

Our control inputs are (W_R, W_L) and our control outputs are (*X, Y,).* Therefore, our desired trajectory is *X (t), Y(t), (t).*

2.2 Linearization

The mobile system model is a 3rd order system and nonlinear. It is very difficult and complicated to study this type of system. Some nonlinear equations can be approximated by linear equations under certain conditions. Most of the systems have a trajectory called the nominal trajectory, and around this trajectory the system behaves like a linear system. In our case we chose these conditions for the system behavior and modeled a linear system as follows.

$X(t) \quad V_0 t$ (2.12)

$Y(t) \quad C$, where C is a constant (2.13)

$$\tan^{-1}\frac{V_y}{V_x} \quad \tan^{-1}\frac{0}{V_0} \tag{2.14}$$

and the initial values for $x_0(t)$, $y_0(t)$, and $_0(t)$ are 0. Suppose the input function u_0 (t) and some initial state x_0, the solution of this system is:

$$\dot{x} \quad f(x,u) \quad \begin{matrix} f_1 \\ f_2 \\ f_3 \end{matrix} \tag{2.15}$$

$$F \quad \| \frac{df}{dx} \|_{u0,\, x0} \qquad G \quad \| \frac{df}{du} \|_{u0,\, x0} \tag{2.16}$$

$$x \quad F\ x \quad G\ u \tag{2.17}$$

$$x \quad x \quad x_0 \tag{2.18}$$

$$u \quad u \quad u_0 \tag{2.19}$$

$$F \quad \begin{matrix} \frac{df_1}{dx_1} & \frac{df_1}{dx_2} & \frac{df_1}{dx_3} \\ \frac{df_2}{dx_1} & \frac{df_2}{dx_2} & \frac{df_2}{dx_3} \\ \frac{df_3}{dx_1} & \frac{df_3}{dx_2} & \frac{df_3}{dx_3} \end{matrix} \qquad G \quad \begin{matrix} \frac{df_1}{du_1} & \frac{df_1}{du_2} \\ \frac{df_2}{du_1} & \frac{df_2}{du_2} \\ \frac{df_3}{du_1} & \frac{df_3}{du_2} \end{matrix} \tag{2.20}$$

So if we differentiate our nonlinear system model we get:

$$F \quad \begin{matrix} 0 & 0 & V_0 \sin & x_1 \\ 0 & 0 & V_0 \cos & x_2 \\ 0 & 0 & 0 & x_3 \end{matrix} \tag{2.21}$$

$$G \quad \begin{matrix} \frac{r_w}{2\cos} & \frac{r}{2\cos} \\ \frac{r_w}{2\sin} & \frac{r}{2\sin} \\ \frac{r_w}{D} & \frac{r_w}{D} \end{matrix} \quad \begin{matrix} u_1 \\ u_2 \end{matrix} \tag{2.22}$$

The mobile robot should walk in a straight line, which means both wheels have the same initial speed. To develop a forward kinematics equation, the specify a frame of reference in which an arbitrarily chosen point is treated as stationary. All other points in the system are treated as moving relative to the reference point. The robot is considered a rigid body. By approaching the problem in this manner, we will gain insight which can be applied to the more general problem of modeling the robot's motion in an absolute frame of reference. The point that selected as our reference is the center point of the left wheel. This is the point where an idealized wheel makes contact with the floor. Again, all motion in this frame of reference is treated relative to the left wheel point. Because the right wheel is mounted perpendicular to the axle, its motion in the frame of reference follows a circular arc with a radius corresponding to the length of axle.

Now the central point itself may be in motion, but the actual path of the right wheel will not necessarily correspond to that particular circular arc. The change in orientation is not restricted to the robot's frame of reference. Because the robot is treated as a rigid body, all points in the system undergo the same change in orientation. The kind of drive that has been used to establish our equations is called a differential drive, and the choice was the simplest form where V_L and V_R are equal. In this case the radius *D* is infinite and the robot moves in a straight line, but we should notice that the robot does not move in a straight line, but rather

follows a curved trajectory about a point a distance *D* away from the center of the robot, changing both the robot's position and orientation. This happens when the mobile robot is not on any kind of control scheme. It happened when the mobile robot under open control loop and has the same speed on his two wheels assuming this will lead the mobile robot to move in a straight line, which in fact the mobile robot follows a curved trajectory. A differential drive vehicle is very sensitive to the relative velocity of the two wheels. Small errors in the velocity provided to each wheel results in different trajectories, not just a slower or faster robot. Differential drive robots typically have to use castor wheels for balance. Thus, differential drive wheels are sensitive to slight variations in the ground plane. This limits applicability in non-laboratory environments.

CHAPTER III

APPLYING CONTROL SCHEMES TO THE MOBILE ROBOT

From the analysis in Chapter II it is seen that mobile robot control is difficult since the control inputs have to be generated to satisfy the constraint while the robot moves. A mathematical model is never a perfect representation of physical system. The real challenge in motion planning is to develop a planning scheme integrated with a control system that is able to detect and recognize unexpected events on the basis of sensory information, and adjust and modify the base plan at a high rate to cope with time and location variations in the occurrence of events without re-planning. In this chapter, Section 3.1 explains the H-infinity control scheme, where Section 3.1.1 discusses the H-infinity feedback controller and Section 3.1.2 explores the full information estimator. Section 3.2 deals with H-infinity implementation on the mobile robot system. Section 3.3 discusses the results of the H-infinity simulation, where Section 3.3.1 explains the simulation terms and Section 3.3.2 shows the D parameter affects on the stability of the system. Section 3.4 describes the P and PI controller implementation. The Simulation results of the P and PI controllers are explained in Section 3.5. Section 3.6 shows the differences between the H-infinity and P and PI controllers.

3.1 H-Infinity Control Schemes

One of the key ideas that has recently emerged in the field of modern control is the use of H-infinity control to give a systematic procedure for the design of feedback control systems. This technique also has been researched in several recent papers [13, 14]. Within the modern control framework, one approach to designing robust control systems is to begin with a plant

model, which not only models the nominal plant behavior, but also models the uncertainties, which are expected. Since a perfect model is not available, the discrepancy between the mathematical model of the plant and the actual plant should be quantified. There are many types of uncertainties and perturbations that affect robustness, stability, and performance in the face of changes in the plant dynamics and errors in the plant model. H-infinity control can be applied to control the robot position to follow walls and keep a certain distance from them. Uncertainty might arise from neglecting some of the system dynamics; e.g., motor dynamics, sensor noise, and unmodeled robot dynamics.

The mobile robot was modeled as a rigid body satisfying nonholonomic constraints, which means the motion of the system is not free. Whatever the form of the uncertainty Δ in an uncertain system, the controller should control the system and keep the mobile robot in the path that has been planned for it. A control system is said to be robustly stable if it is stable for all admissible perturbations. A control system is said to perform robustly if it satisfies the performance specifications for all admissible perturbations. Note that stability and performance robustness depends on the controller, the nominal model, and the set of perturbations. Motivated by the above considerations and the importance of the problem, this thesis addresses the kinematics model of the mechanical system of a two-wheeled mobile robot.

Our state space system is multi-input, multi-output (MIMO). The robot should follow a wall in a straight line, which means the robot moves at a constant forward speed along a predefined geometric path that is given in a time-free parameterization. Thus, our controller should compute the distance of the robot to the wall, minimize the error between the robot's main axis and the tangent to the path, and act on the angular velocity to drive both error and angular velocity to zero.

In general, the goal of robust control is to measure the MIMO system stability margin using a proper, non-conservative analytical tool. In other words, the goal is to find out how big Δ can be before instability occurs. Robust stability in the presence of structured perturbations and robust performance both depend on the supremum of the singular value from the perturbation input to the perturbation output. The supremum of the largest structure singular value (SSV) is bounded by the ∞-norm of the diagonal-scaled, closed loop system [14, 15]. The norm is a real valued function of the elements of a linear space. The norm provides a measure of the size of a vector, signal, or system. The ∞-norm is given as $\|G\|$ $\mathrm{Sup}_w \bar{\sigma}|G(jw)|$. The maximum gain of a generic system over all frequencies is by the system ∞-norm. In other words, the ∞- norm of a system provides a bound of the maximum system gain, where the gain is defined in terms t he signal 2-norm:

$$\|g(t) \quad w(t)\|_2 \quad \|G\| \quad \|w(t)\|_2 \qquad (3.1)$$

Where g(t) w(t) is the input convolved with the impulse response matrix, which yields the time domain system outputs[22].

So what is the H-infinity control problem and how can it be formulated? How does the H-infinity controller control the system (the mobile robot)? What is the H -infinity estimator? What are the properties and the conditions for H-infinity control?

3.1.1 The Output Feedback Control

An important application of the H-infinity ($H\infty$) control problem arises when studying robustness against model uncertainties. It turns out that the condition that a control system is robustly stable in spite of a certain kind of model uncertainties can be expressed quantitatively in terms of an $H\infty$ norm bound which the control system should satisfy.

The optimization of the ∞-norm has application in both maximizing performance and robustness. Control and estimation problems, with the goal of minimizing the system's ∞-norm, are termed $H\infty$ optimization problems, which lead to design techniques such as full information control and output estimation. When they are combined together, they form $H\infty$ output feedback control. The $H\infty$ output feedback controller (simply the $H\infty$ controller) utilizes partial state measurements, corrupted by disturbances, to generate the control. The following state model gives the plant:

$$\dot{x} \quad Ax(t) \quad [B_u \; B_w] \begin{matrix} u(t) \\ w(t) \end{matrix} \tag{3.2}$$

Where the B_u *is* the control matrix and B_w is the reference input matrix (the disturbance).

$$\begin{matrix} m(t) \\ y(t) \end{matrix} \quad \begin{matrix} C_m \\ C_y \end{matrix} \; x(t) \quad \begin{matrix} 0 & D_{mw} \\ D_{yu} & 0 \end{matrix} \; \begin{matrix} u(t) \\ w(t) \end{matrix} \tag{3.3}$$

C_m *is* the feedback matrix and C_y is the output matrix in Equation 3.3. The matrices C_y and D_{yu} are assumed to satisfy the following:

$$\begin{matrix} D_{yu}^T C_y & 0 \\ D_{yu}^T D_{yu} & I \end{matrix} \tag{3.4}$$

These conditions in Equation (3.4) require that the reference output consist of an output dependent only on the state and a distinct output dependent only on the control input. Furthermore, the portion of the output that depends on the control is simply equal to the control input or an orthogonal transformed version of this input. Another condition is that the plant

(the system that to control, which in this case is the mobile robot system) should be stable. The matrices B_w and D_{mw} are assumed to satisfy the following conditions:

$$\begin{matrix} D_{mw}B_w^T & 0 \\ D_{mw}D_{mw}^T & I \end{matrix} \tag{3.5}$$

These conditions require that the disturbances entering the plant via the state equation and the disturbances entering the plant via the measurement equation must be distinct. These conditions guarantee the existence of a steady state H∞ controller.

The suboptimal H∞ control problem is to find a feedback controller for the above plant such that the ∞-norm of the closed-loop system is bounded:

$$\left\| G_{yw} \right\|_{\prime[0,tf]} \quad \sup \left\| w(t) \right\|_{2[0,tf]} \; 0 \frac{\left\| y(t) \right\|_{2.[0,tf]}}{\left\| w(t) \right\|_{2.[0,tf]}} \tag{3.6}$$

where γ is called the performance bound. The closed–loop system is also required to be internally stable when the final time is infinite. The solution of the optimal H∞ control problem is to minimize the closed loop ∞–norm. Finding the suboptimal H∞ controller, it is assumed that the full information controller exists. The ∞–norm bound in Equation (3.6) that defines the suboptimal H∞ controller is equivalent to requiring that the inequality

$$J \quad \left\| y(t) \right\|_{2[0,tf]}^2 \quad {}^2 \left\| w(t) \right\|_{2[0,tf]}^2 \quad {}^2 \left\| w(t) \right\|_{2[0,tf]}^2 \tag{3.7}$$

be satisfied for some positive ϵ and for all disturbance inputs. Since all possible outputs can be generated from the state and disturbance input, the inequality in Equation (3.7) can be written in terms of the full information Riccati solution as:

$$J \quad \left\| u(t) \quad B_u^T P(t)x(t) \right\|_{2,[0,tf]}^2 \quad {}^2\left\| w(t) \quad {}^2 B_w^T P(t)x(t) \right\|_{2,[0,tf]}^2 \quad {}^2\left\| w(t) \right\|_{2,[0,tf]}^2 \tag{3.8}$$

where *P(t)* is the solution of a Riccati equation, which will be discussed in more detail below. A Riccati Equation is an equation that is fully specified by the state transition matrix of a Hamiltonian system. This equation has only final conditions and can be solved backward in time using any numerical integration package. Since the feedback gain only depend on this matrix and Bu, a differential equation for *P(t)* can be generated by taking the derivative of this equation:

$$p(t) \quad P\ t\ x(t) \tag{3.9}$$

$$\dot{p}(t) \quad \dot{P}(t)\,x(t) \quad P(t)\,\dot{x}(t) \tag{3.10}$$

Where *P(t)* is the matrix of proportionality between the costate *p(t)* and the state. This matrix of proportionality is fully specified by the state transition matrix of the Hamiltonian system:

$$\begin{matrix} \dot{x}(t) \\ \dot{p}(t) \end{matrix} \quad \begin{matrix} A \\ C_y^T C_y \end{matrix} \quad \begin{matrix} B_u B_u^T \\ A^T \end{matrix} \quad {}^2 B_w B_w^T \quad \begin{matrix} x(t) \\ p(t) \end{matrix} \quad \begin{matrix} x(t) \\ p(t) \end{matrix} \tag{3.11}$$

Substituting for $\dot{p}(t)$ and $\dot{x}(t)$, and using the Hamiltonian system in equation (3.11) yields:

$$C_y^T C_y\ x(t) \quad A^T p(t) \quad \dot{P}(t)\,x(t) \quad P(t)\{Ax(t) \quad (B_u B_u^T \quad {}^2(B_w B_w^T)\}p(t) \tag{3.12}$$

Rearranging equation (3.12) and dropping the (t) from each term above for notational convenience leads to equation (3.13):

$$PA \quad A^T P \quad P(B_u B_u^T \qquad {}^2 B_w B_w^T)P \quad C_y^T C_y \quad 0 \tag{3.13}$$

This equation is the Riccatti differential equation for H-infinity suboptimal control problem. The Riccati solution for *P(t)* is a symmetric matrix (if it exists), which can be found by solving equation (3.13) backward in time from the final condition. The final condition is obtained by letting *P(t) = 0.*

As mentioned above, this equation has been solved by using any numerical integration package. In our case, MATLAB® using the command CARE (Continuous Time Algebraic Riccatti Equation). For more information see [5, 7, 15]. The suboptimal solution that internally stabilizes the closed-loop system and bounds the closed-loop ∞ –norm exists if and only if there is a positive semi-definite solution of algebraic Riccati equation.

$$P(t_f) \quad 0 \tag{3.14}$$

By assuming the condition on Equation (3.14) above, we can find the controller equation that is u(t) = $- B_u^T P(t)x(t)$ as the suboptimal control. It exists if the Riccatti Equation has a solution over the entire time interval from 0 to t_f and $Ax(t) \quad (B_u B_u^T \qquad {}^2 B_w B_w^T)$ is stable, that is, all of the eigenvalues of this matrix have negative real parts. The suboptimal controller is then given as

$$u(t) \qquad B_u^T P(t)x(t) \qquad Kx(t) \tag{3.15}$$

The control law in Equation (3.15) has been shown to satisfy the infinity norm bound on Equation (3.6) and it is exists as long the Riccati Equation has a solution. The following question immediately comes to mind: what happens if the Riccati Equation has no solution? Is it possible to find a full information controller that satisfies the bound on Equation (3.6)? The answer for this question is no! The existence of a solution to the Riccati Equation is both necessary and sufficient for the existence of a solution to the H-infinity suboptimal control problem.

The H-infinity output controller utilizes partial state measurements, corrupted by disturbances, to generate the control. This controller can be synthesized by combining an H-infinity full information controller with an H-infinity estimator. The H-infinity estimator gain depends on what linear combination of the states is being estimated. The worst case disturbance input that appears in the full information minimax problem must be included in the H-infinity estimator equations. The suboptimal H-infinity control problem is defined in the state equations in equation (3.16) below:

$$\begin{array}{l} x(t) \quad Ax(t) \quad B_u(t) \quad B_w w\ t \\ m(t) \quad C_m x\ t \quad D_{mw} w\ t \\ y(t) \quad C_y x(t) \quad D_{yu} u(t) \end{array} \tag{3.16}$$

The matrices B_w and D_{mw} are assumed to satisfy the following:

$$\begin{array}{l} D_{mw} D_w^T \quad 0 \\ D_{mw} D_{mw}^T \quad I \end{array} \tag{3.17}$$

These conditions require that the disturbances entering the plant and the measurement be distinct and that the output equations of the plant be scaled to normalize the measurements

noise. The H-infinity filter estimates linear combinations of the state, given the measured output of the plant:

$$y(t) \quad C_y x(t) \tag{3.18}$$

3.1.2 The Full Information Estimator

An optimal H-infinity estimator generates estimates that minimize the worst-case gain between the disturbance input and the estimation error e(t) = y(t) $\hat{y}$(t). The bound in equation (3.19) can be used in formulating a suboptimal $H\infty$ output estimation problem :

$$\|G\| \quad \sup_{w(t) \quad {}^2B_w^T P(t)x(t) \quad 0} \frac{\left\| u(t) \quad B_u^T P(t)x(t) \right\|_{2[0\ tf]}}{\left\| w(t) \quad {}^2B_w^T P(t)x(t) \right\|_{2[0\ tf]}} \tag{3.19}$$

The solution for this problem leads directly to a suboptimal controller. This output estimation problem is stated as follows: Estimate the full information control input, given the measurement *m(t),* such that the ∞–norm of the transfer function between the disturbance input $w(t) \quad w(t) \quad {}^2B_w^T P(t)x(t)$ and the estimation error is bounded:

$$\|G\| \quad \sup_{w(t) \quad 0} \frac{\left\| y(t) \quad \hat{y}(t) \right\|_{2[0\ tf]}}{\left\| w(t) \quad {}^2B_w^T P(t)x(t) \right\|_{2\ [0\ tf]}} \tag{3.20}$$

and this equation is equal to :

$$\|G\| \quad \sup_{w(t)\ 0} \frac{\left\| B_u^T P(t)x(t) \quad u(t) \right\|_{2[0\ tf]}}{\left\| w(t) \quad {}^2 B_w^T P(t)x(t) \right\|_{2[0\ tf]}} \tag{3.21}$$

This equation above is equal to equation (3.20) since the absolute value of $k \| x \| \|x\|$. The state model of the plant in this estimation problem has $w(t)$ as the disturbance input and m(t) as the measured output becomes:

$$\begin{array}{l} x(t) \quad Ax(t) \quad B_u u(t) \quad B_w w(t) \quad {}^2 B_w B_w^T P(t)x(t) \quad {}^2 B_w B_w^T P(t)x(t) \\ [A \quad {}^2 B_w B_w^T P(t)x(t) \quad B_u u(t) \quad B_w \quad w(t) \end{array}$$

(3.22)

The measurement equation for this model can be generated by subtracting ${}^2 D_{mw} B_w^T Px$, which equals zero because $D_{mw} B_w^T \quad 0$, one of our conditions that guarantees the existence of a steady state $H\infty$ controller:

$$\begin{array}{l} m(t) \quad C_m x(t) \quad D_{mw} w(t) \quad {}^2 D_{mw} B_w^T P(t)x(t) \\ C_m x(t) \quad D_{mw} \quad w(t) \end{array} \tag{3.23}$$

The suboptimal $H\infty$ estimator for the problem specified in the Equations (3.16) - (3.23) generates estimates of the full information control, and the estimator becomes an output feedback controller, since it generates control inputs from the measurements. Therefore, the estimator is a suboptimal $H\infty$ controller. The Riccati equation for the estimator is equal to:

$$Q(t) \quad Q(t)A^T \quad AQ(t) \quad B_w B_w^T \quad Q(t)(C_m^T C_m \quad {}^2 C_y^T C_y)Q(t) \tag{3.24}$$

This Riccati equation is solved forward in time from the initial condition

$$Q(0) \quad 0 \tag{3.25}$$

The estimator gain can be written in terms of this new Riccati solution:

$$G(t) \quad Q\,t\,C_m^T \tag{3.26}$$

Equations (3.24)-(3.26) completely specify the suboptimal $H\infty$ estimator, and it should be to recognize that the Riccati equation and estimator gain depend on the output being estimated.

3.2 H-Infinity Controller Implementation

Figure 6 shows the plant and the estimator. To see how they work together and how the estimator becomes the controller; let's apply the equations to our robotic system. Starting with the state equations that represent our linear system:

$$\begin{aligned} x(t) &\quad Ax(t) \quad B_u(t) \quad B_w w\,t \\ m(t) &\quad C_m x\,t \quad D_{mw} w\,t \\ y(t) &\quad C_y x(t) \quad D_{yu} u(t) \end{aligned} \tag{3.27}$$

The A, B_u, C_m, D_{mw}, C_y, and D_{yu} matrices are given in Appendix A. The system in Equation (3.27) can then be written as

$$\begin{aligned} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & V \\ 0 & 0 & 0 \end{bmatrix} x(t) \begin{bmatrix} r/2 & r/2 \\ 0 & 0 \\ r/d & r/d \end{bmatrix} u(t) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} w\,t \\ m(t) &\quad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} x\,t \\ y(t) &\quad \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} x(t) \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} u(t) \end{aligned} \tag{3.28}$$

where $V_0 = \frac{r}{2}(u0(1) \quad u0(2))$, and u_0 is the nominal control signal for the left and right wheels. Equations (3.13) and (3.24) are solved by MATLAB® because of their complexity. In Figure 6,

the existence of conditions for the suboptimal H-infinity controller imply the existence of both a suboptimal H-infinity full information controller and a suboptimal H-infinity filter for estimating the reference output. The implied existence of a full information controller can be understood by noting that full information includes information on all possible outputs. Therefore, output feedback is a special case of full information control.

In Figure 6, the upper part represents the mobile robot system (the original system), while the bottom represents the estimator (in our case the controller). Note that the input, output, and the state of the estimator are distinct from the input, output, and the state of the original system. Given the output equation for the original system

$$y(t) \quad C_y x(t) \tag{3.29}$$

The H-infinity estimator generates estimates that minimize the worst-case gain between the disturbance input and the estimation error e(t) = $y(t) \quad \hat{y}(t)$ The bound in

$$J \quad \|G_{ew}\|_{2[0\ tf]} \quad \sup_{w(t)\ 0} \frac{\left\| y(t) \quad \hat{y}(t) \right\|_{2[0\ tf]}}{\|w(t)\|_{2[0\ tf]}} \tag{3.30}$$

If the difference between $y(t) \quad \hat{y}(t)$ is zero, no correction is needed.

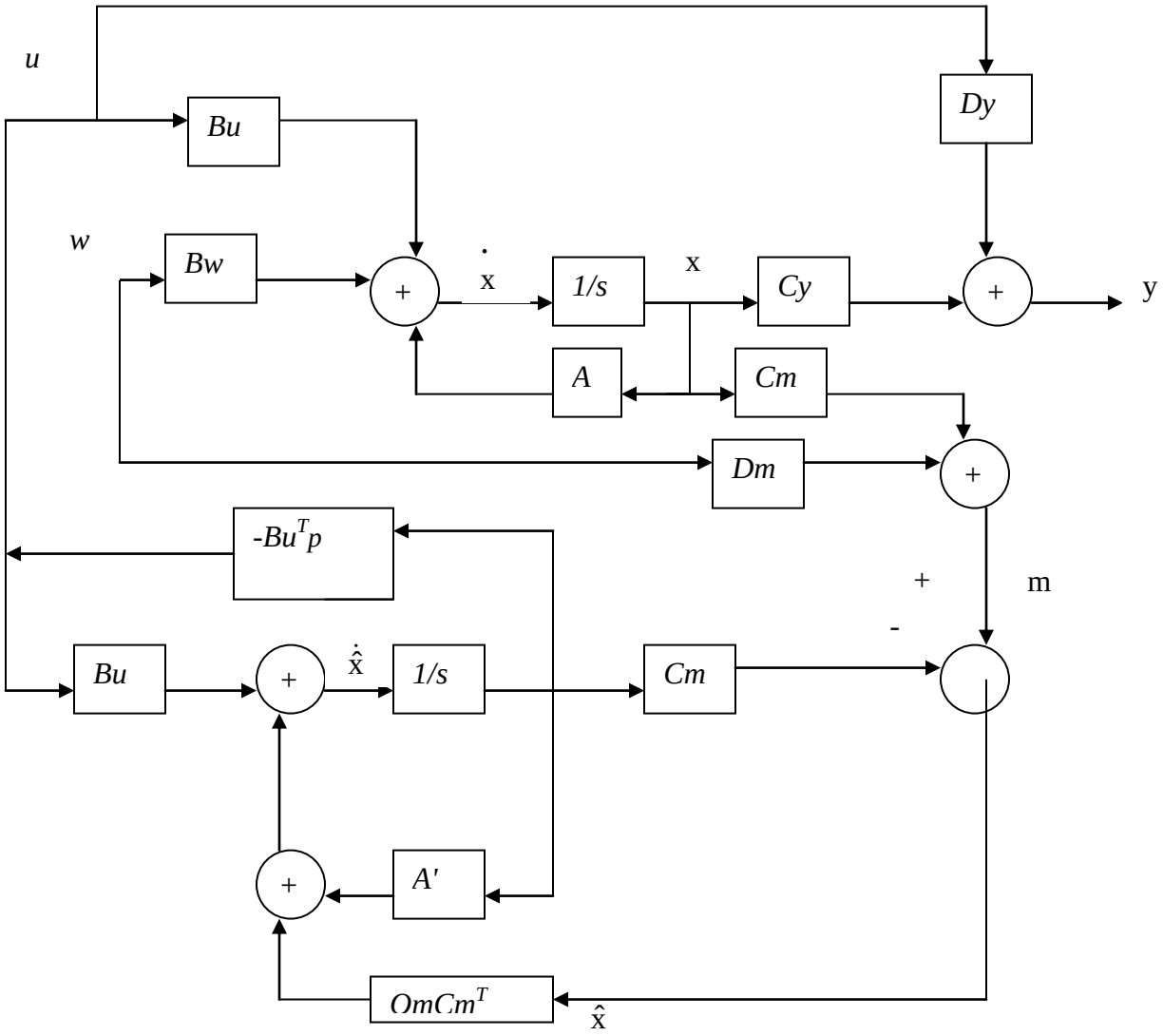


Figure 6: The Estimation Problem in Standard Form

3.3 Simulation results of H-infinity control

After translating the above analysis into MATLAB® programs, simulated the nonlinear system, the linearized system to get the steady state value for the Riccati equation, then we use the solution as a gain for the estimator part of the H-infinity design. Designing MATLAB® program to minimize the error between the original plant and the estimator by changing the performance index gamma (γ) was not an easy task. We tested the system by

adding white noise disturbance to the feedback signal and input disturbances to the plant dynamics equations. We tuned the performance index γ to get the best results. The simulation results have been run under different settings by changing the amount of noise and disturbances, changing the step size of simulation and the time of simulation (length of simulation). We experimented with different values for the mechanical variables to see how they affect the system's robustness and performance.

3.3.1 Simulation Terms

The "randn" function in Matlab generates a random number and is used in our simulations to model disturbances in multivariable control. The output of randn is zero mean Gaussian white noise with standard deviation = 1. A simple random process that is often employed in disturbance rejection is white noise. Examples of white noise in our system come from wheel slippage, and sensor nonlinearties. White noise is an idealization of zero-mean random input noises and does not exist in nature. Actually colored noise disturbances are more appropriate [25]. In MATLAB® program, the white noise is inserted in the measured output feedback equation, while the input disturbance is inserted to the dynamics equations of the system.

Figure 7 shows the robot angle when gamma is 50, the white noise is 0.01 randn, the unit of the noise is radians/second, the input disturbance is 0.01 randn, and the unit of the input disturbance is radians/second. The simulation has been run for different gamma values to test our system robustness and performance.

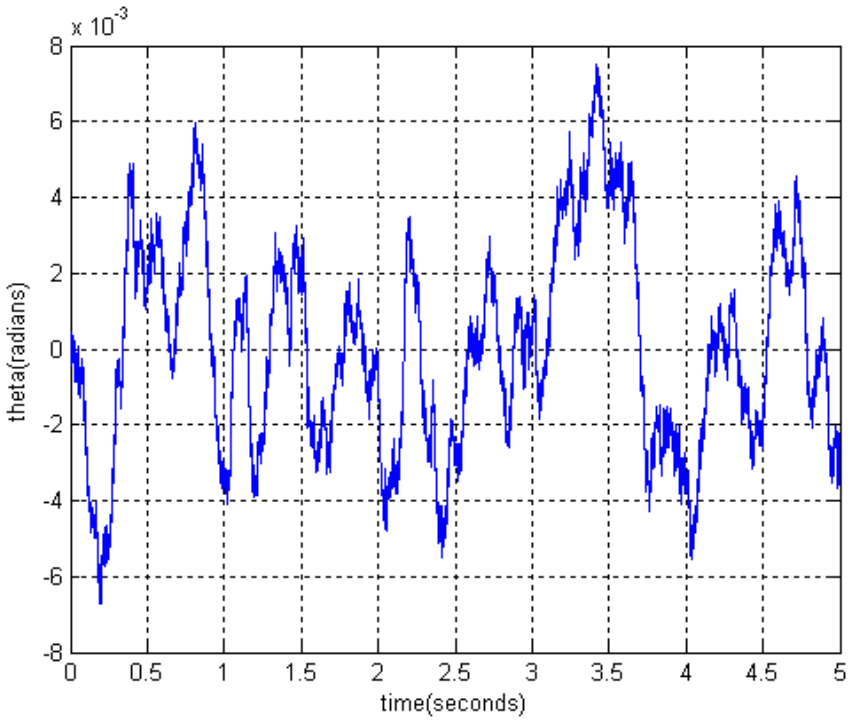


Figure 7: Angle Mobile Robot-Gamma is 50

The results of simulation are recorded in Table I for different settings for all the system variables. In Table I, the setting for the simulation is that the white noise is 0.01 randn which is very good amount to test the system for its robustness and performance and the step size of simulation is 0.01 seconds and the disturbance inputs are 0.01 randn. The magnitude of the systems variables in Table I is standard deviation (Std) measurement, and it has been taken for each variable involved in the system.

gamma	Control signal L Radian/seconds	Control signal R Radian/seconds	Y-position Inches	Theta-Radians
8	0.00686	0.00685	0.00227	0.00970
10	0.00468	0.00468	0.00216	0.00973
15	0.00336	0.00336	0.00309	0.00814
20	0.00350	0.00350	0.00278	0.00955
30	0.00318	0.00318	0.00231	0.00886
40	0.00303	0.00303	0.00272	0.00938
50	0.00273	0.00273	0.00239	0.00829

60	0.00242	0.00242	0.00336	0.01021
100	0.00329	0.00329	0.00408	0.01013
120	0.00322	0.00322	0.00481	0.01387
150	0.00302	0.00302	0.00383	0.00901
200	0.00364	0.00364	0.00375	0.01307
10000	0.00319	0.00319	0.00272	0.01051

Table I: The Simulation Results for Mobile Robot System

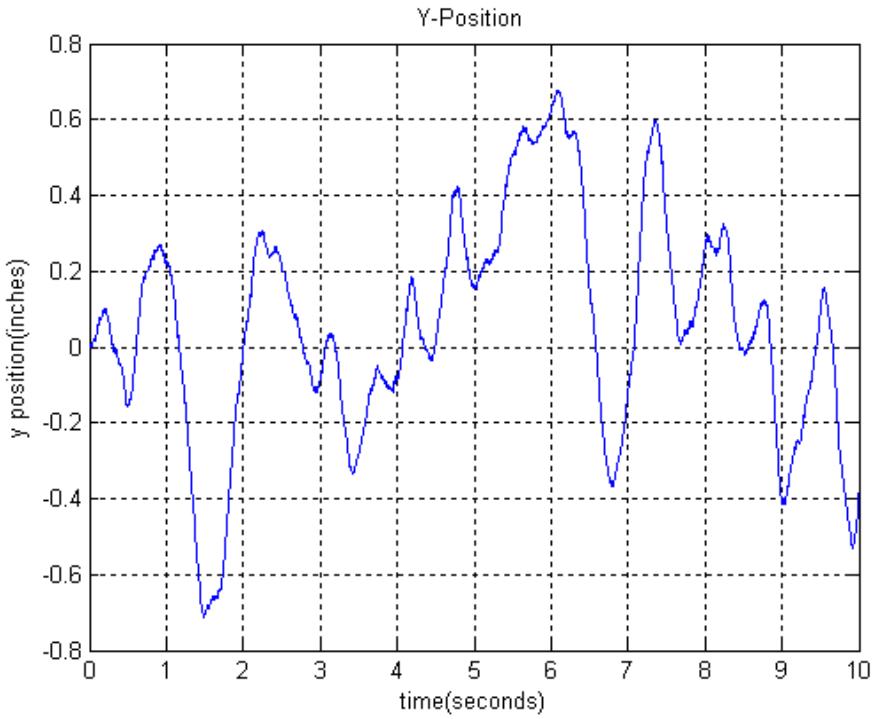


Figure 8: Y-position of mobile robot-gamma is 8

In Figure 8, the simulation was for gamma = 8. The magnitude value for the Std measurement is 0.002272 inches. If we study the results of simulation in Table I for the Y-position values we notice the system shows a lot of robustness and stiffness against disturbances. When gamma increased the Y-position values goes up a little bit and goes down again, but no abrupt jumps at all in the Y signal. It keeps it's stability for any gamma values larger than 8. When gamma is under 8, the system gives no answer for the values, which means the H-infinity controller has no solution and is out of range of the system's parameters. With

both large disturbances and small disturbances our controller shows effectiveness in controlling the system.

Figure 9 shows the Y- position of the mobile robot when Gamma is 50.

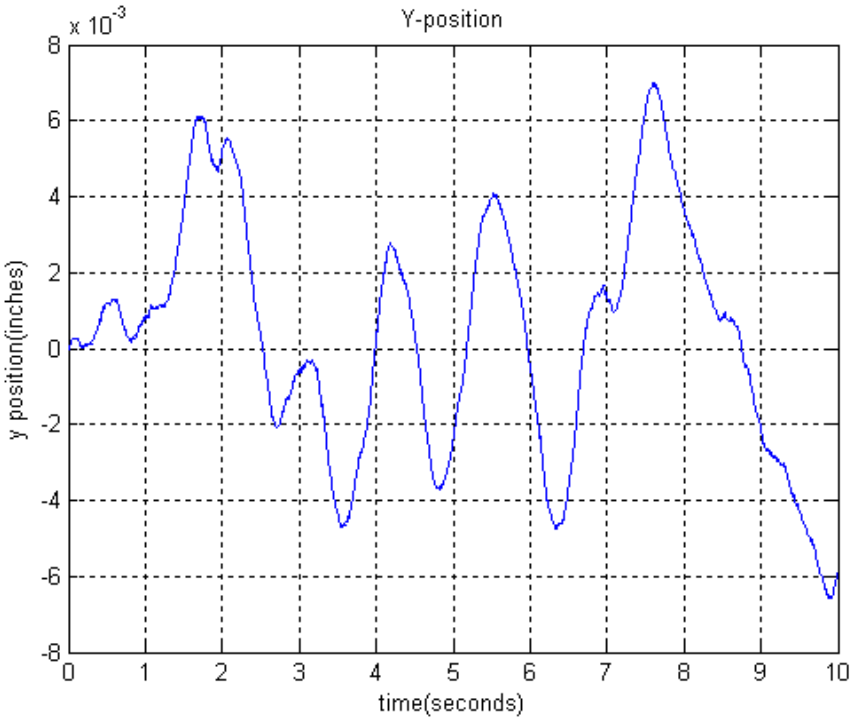


Figure 9: Y-position of mobile robot-gamma is 50

The control signals for the right wheel and the left wheel have the same magnitude under both settings which practically is very ideal due the difference between the motors, and the wheels. Through our control simulation the same signal was used for both wheels which is one of the conditions to force the robot to follow the wall in a straight line. Simulation of the program with different gamma shows that the signal control for both wheels is very relaxed as long gamma is increased.

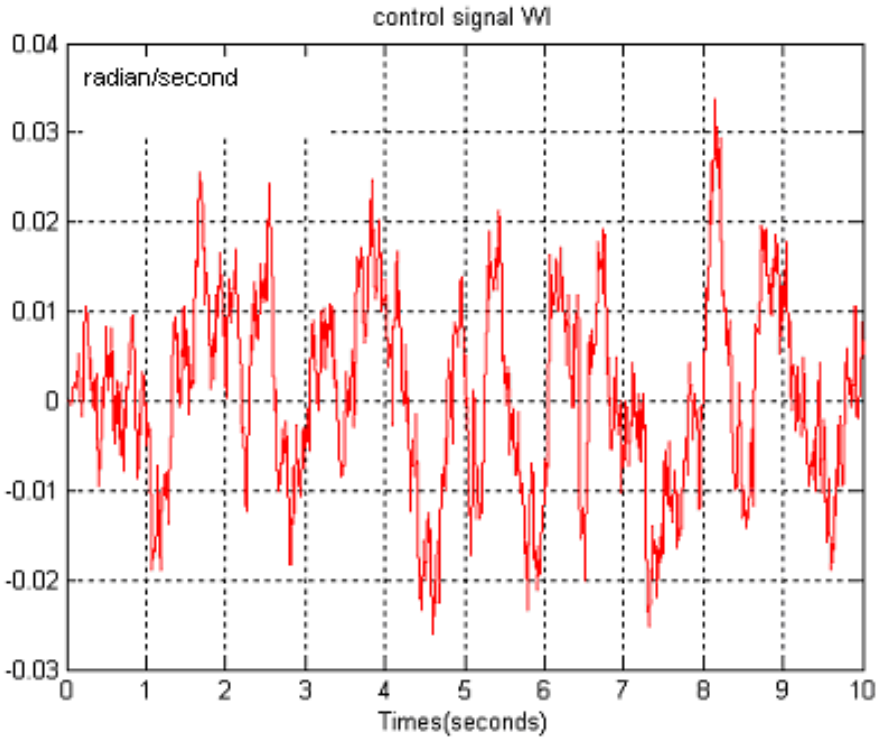


Figure 10: Control Signal for Left wheel when Gamma is 50

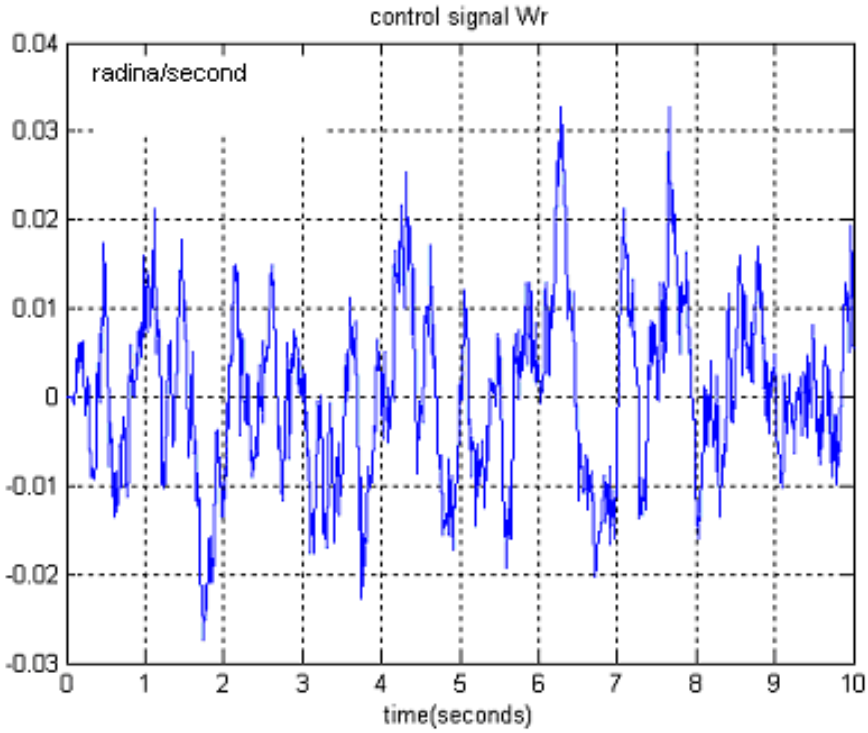


Figure 11: Control Signal for Right Wheel when Gamma is 50

Studying the data that was collected from the simulation which is recorded in Table I shows the increasing gamma minimizes the Std deviation of the control signals for awhile then the magnitude increased again but there is no abrupt jump in the values, which shows the stability of the mobile robot under H-infinity control. Although Gamma is increased to show how much the system can handle the changing of the system index performance, still it shows robustness and good disturbances rejection.

Figures 12, 13 represent the right and the left wheel control signals respectively when gamma is 10000 and the white noise is 0.01 randn and the disturbances are 0.01 randn. The unit for both is radians /inches.

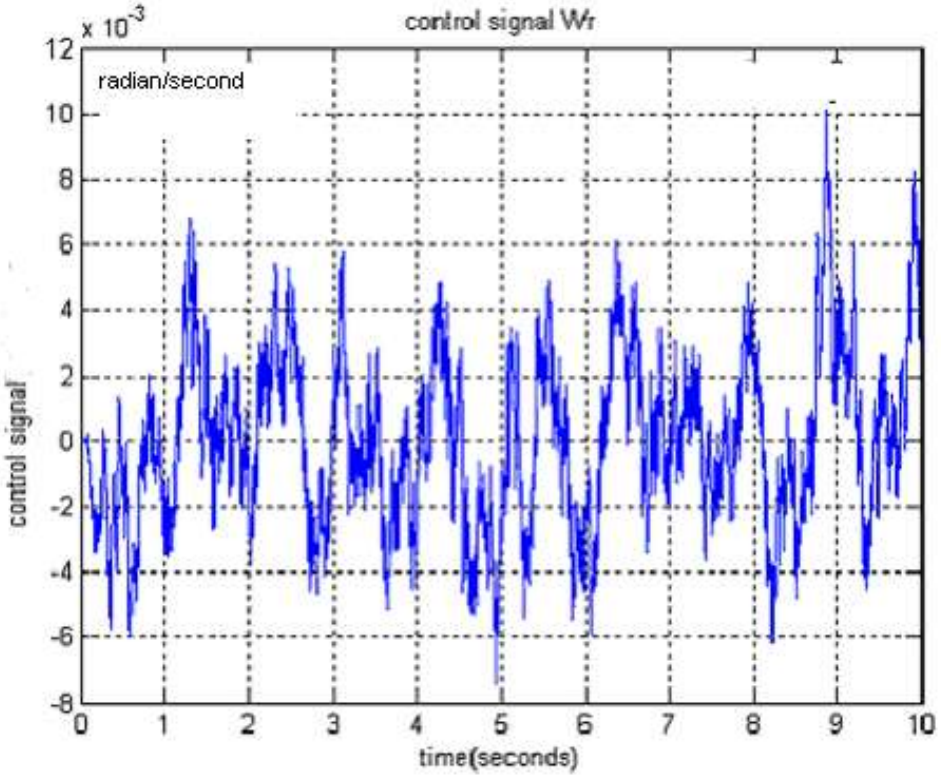


Figure 12: Control Signal for The Right Wheel when Gamma is 10,000

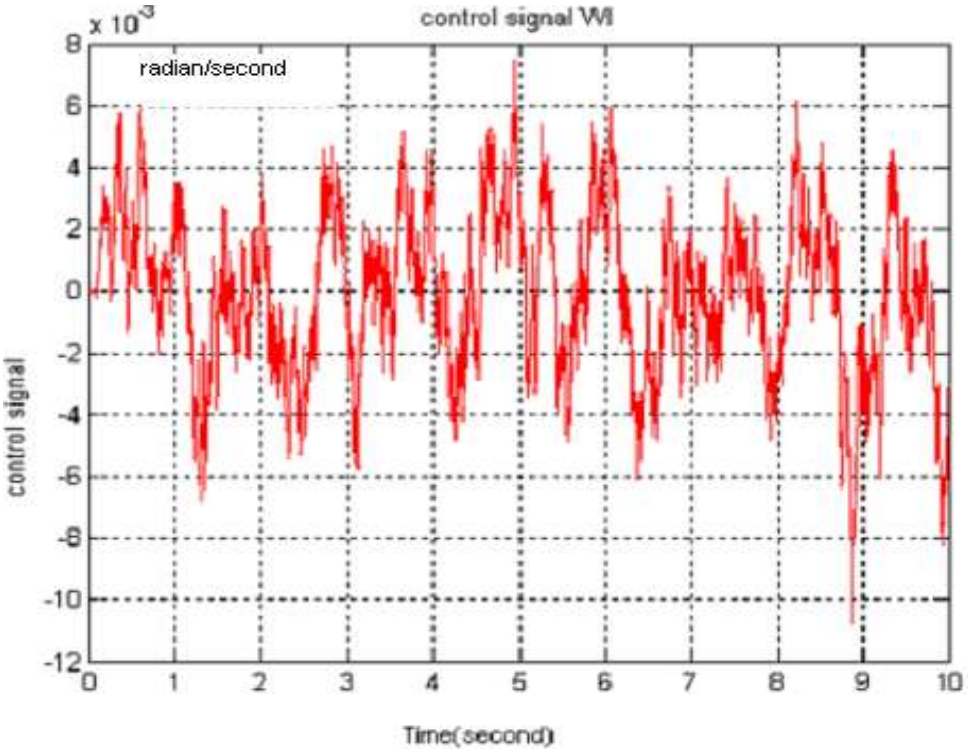


Figure 13: Control Signal for The Left Wheel when Gamma is 10,000

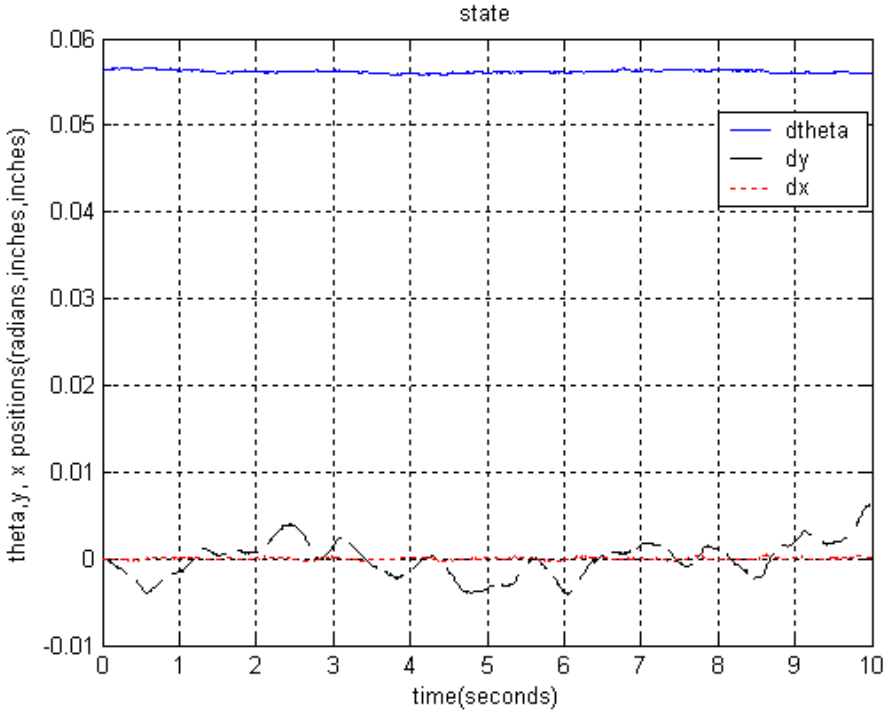


Figure 14: States of the Mobile Robot System –Gamma is 50

Figure 14 shows the states of the system under the setting where the white noise is equal to 0.01 randn which is still a good amount to test the system for robustness and performance and the step size of simulation is 0.001 seconds and the disturbance inputs are 0.01 randn. The program was simulated under different gammas. The magnitudes of the standard deviation (Std) measurements for each state involve in the system was taken, and recorded in Table II.

gamma	**Theta State radians**	**y-State radians**	**x-State radians**
8	0.000312	0.00227	0.000169
10	0.000421	0.00216	0.000170
15	0.000268	0.00309	0.000142
20	0.000400	0.00278	0.000167
30	0.000595	0.002318	0.000154
40	0.00036	0.00272	0.000164
50	0.00066	0.000375	0.000145
60	0.000187	0.00335	0.000107
100	0.000352	0.00408	0.000901
120	0.000378	0.00480	0.000242
150	0.000379	0.003823	0.000157
200	0.000264	0.00375	0.000228
10000	0.000280	0.00272	0.000184

Table II: States of Mobile Robot System

A close look to the data that is recorded in Table II shows that increasing gamma improves the performance of the system in general. Although there is no abrupt jump in magnitudes of Std values, still there is a small increase in the Std magnitude and then it goes down for all the variables of the system. The explanation for this behavior is that the non-linearties in the system cause it. Notice that the y-state is still high in magnitude compared to theta-state.

3.3.2 The Azimuth Parameter

The system is tested by changing the length (D) between the wheels (azimuth parameter). Notice that a change in this parameter affects the stability of all the parameters especially the control signal. The length in our system is 8.8 inches. The system works until 20 inches length and not less than 4 inches, but notice that 8.8 inches is a good measurement design for stability of the mobile robot system. Table III shows the relationship between the D length and the control signal.

D length (Inches)	L/R wheel Control signals Radian/second
20	unstable
15	0.0392
12	0.0415
10	0.0579
8.8	0.0691
6	0.0895
4	unstable

Table III: The D Parameter Affects the Control Signals

Table III shows the relationship between the D length and the control signals for right and left wheels. The simulation under the setting of gamma is 8, the step size of simulation is 0.001, and the time of simulation is 10 seconds. A close look at Table III shows that the control signal increased when D parameter increased until losing the capabilities of control, either when D parameter is 4 inches or when it is 20 inches. The simulation results gave us this error "Error using ==> eig" "NaN or Inf prevents convergence." This error means that the D

parameter directly affects the eignvalues of the system and increasing it or decreasing cause the system to be unstable.

3.4 P and PI Controller Implementation

A traditional planning and control system can be described as in Figure 15; the core of the system is the feedback control loop, which ensures the system's stability, robustness, and performance. The feedback turns the controller into an investigation-decision component. The planning process, however, is done off line, which is understandable because the task is usually predefined. The plan is described as a function of time, and the planner gives the desired input to the system according to the original plan. Therefore it could be considered as a memory component for storing the predefined plan. All uncertainty and unexpected events that were not considered in planning are left to the feedback control loop to handle.

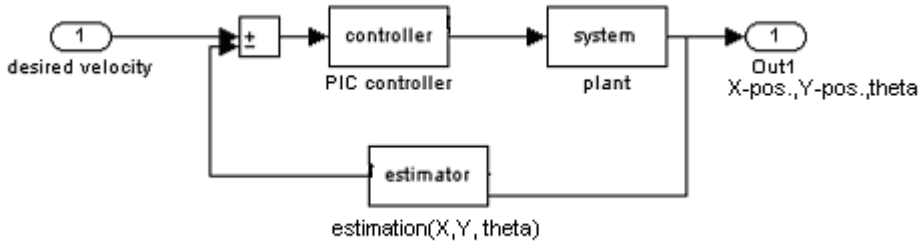


Figure 15: Traditional planner with PIC as a Controller

As seen in Figure 15, the closed loop consists of a controller such as a Microchip PIC microcontroller, a plant (system) and feedback (the sensor). The controller that is implemented in this loop is either a proportional controller, or a PI controller. The P controller has an output u proportional to its input e; that is, $u = K_p e$ where K_p is a proportionality

constant. The integral part of the PI controller is a controller that has an output u proportional to the integral of its input e, which is $u \quad K_p e \quad k_I \quad e(t)\, dt$ where K_I is an integral constant.

The proportional or P controller is the most basic controller. It is simple to implement and simple to tune. Tuning a proportional controller is straightforward; raise the gain K_p until instability appears, or we can say raise the gain K_p until the system begins to overshoot. The loss of stability is a consequence of phase loss in the loop, and the proportional gain will rise to press that limit. However, other factors also play roles, such as noise that limits the proportional gain below the stability criterion that the system demands.

3.4.1 Simulation Results of the P Controller

Figure 16 represents the simulink model for the P controller and the mobile robot system. The P gain is varied from 1 to 100. The gains 1 to 100 characterize the performance of the P controller. After the tuning process, the best result had been recorded and the step response has almost no overshoot and follows the input signal nicely.

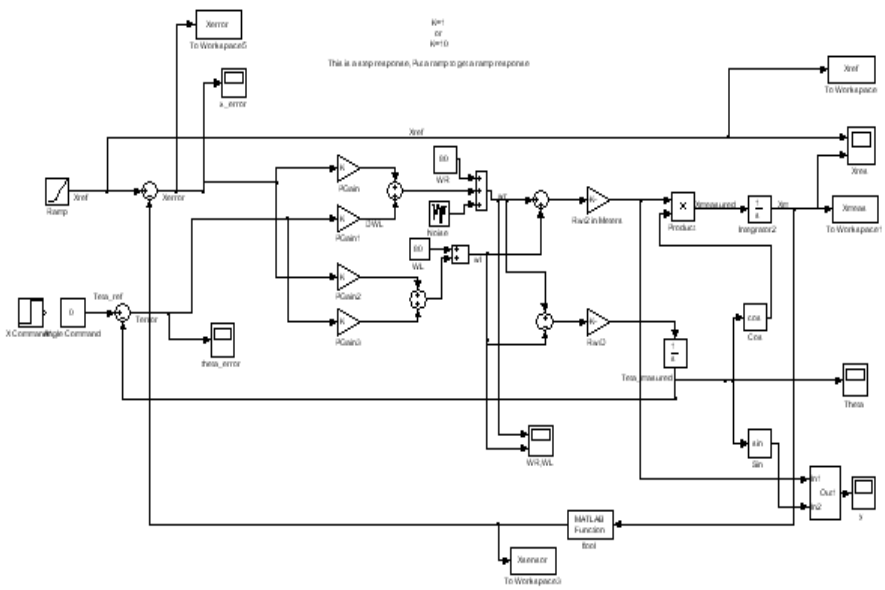


Figure 16: Simulink Diagram for P controller

Figure 16 shows the Simulink diagram for P controller. SIMULINK is a program for simulating dynamic system. It is a graphical representation for the dynamic systems. It is part of MATLAB® control tools. This diagram in Figure 16 is built with blocks to represent the dynamic system of the mobile robot, and the proportional controller. Adding white noise disturbance is to test the system stability and robustness. The blocks above represent the nonlinear system of the mobile robot. The proportional gain is changed to see how far the system can handle high gains. The mobile robot system can handle gains from 1 - 100, after that the system starts to overshoot which means the system becomes unstable.

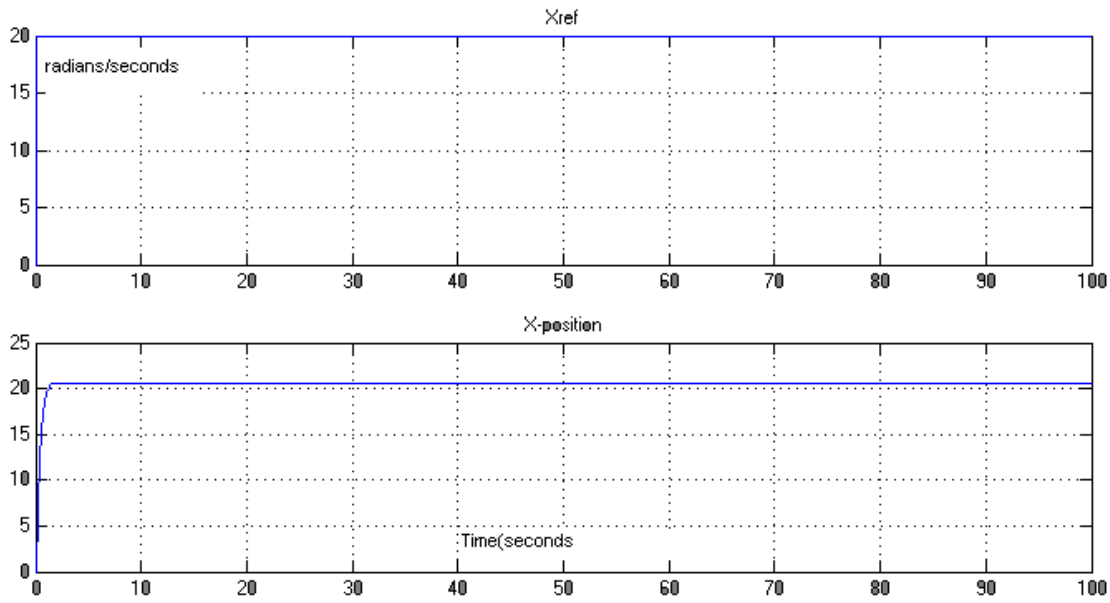


Figure 17: Reference and X-position for P controller

Figure 17 shows the X-position for the P controller when the K_p gain is 50. The proportional control is proportional to the error or the change in the measurement between the reference input and the output. The steady state error is 0.5 deviations from the set point (offset) of the proportional controller. If the gain increases to more than 100, the control loop become unstable, actually the steady state errors becomes larger and overshoot.

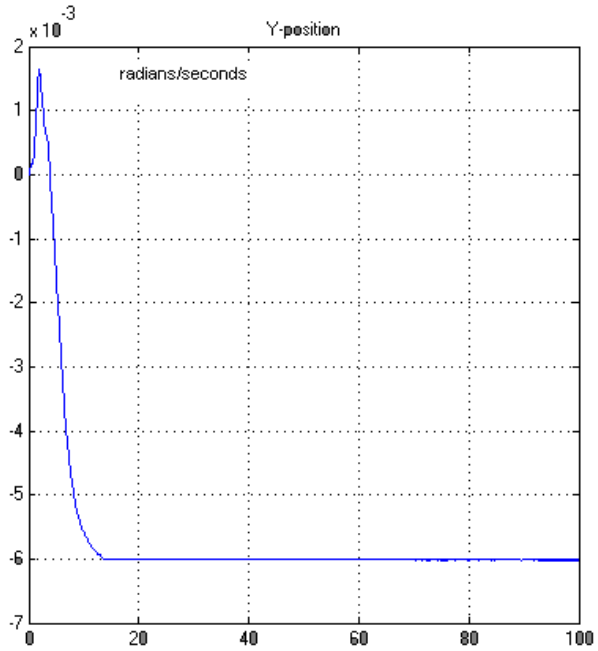


Figure 18: Y-Position for P controller

Figure 18 shows the Y-position of the P controller when the gain is 50 and Figure 19 shows theta the angle for the mobile robot system.

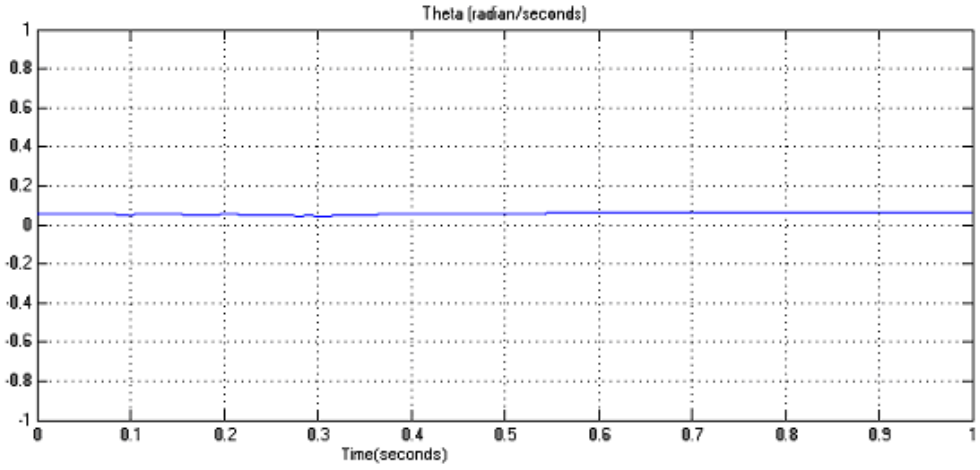


Figure 19: Theta position for P controller

Figure 20 shows the control signals for the mobile robot under P control. As seen from the figure the P controller was capable to drive the control signals for stability, but the control signals are still high.

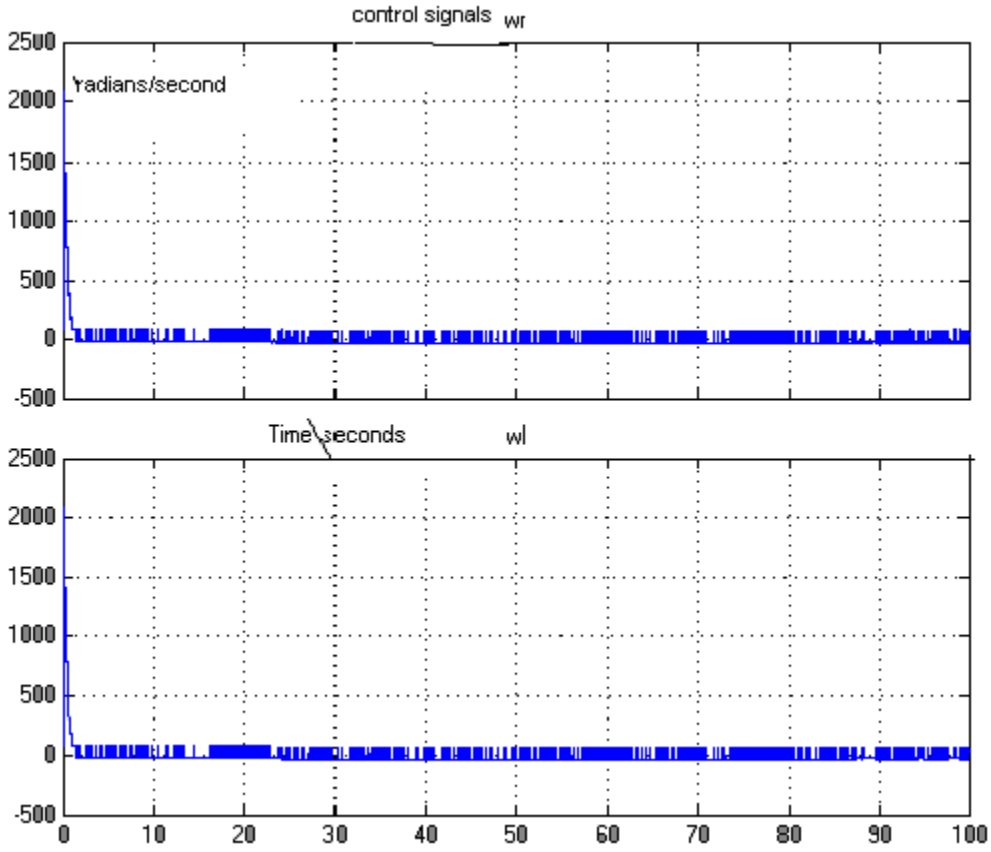


Figure 20: Control signals for P controllers

3.4.2 Simulation Results of PI Controller

The primary shortcoming of the P controller is the lack of its tolerance for high gains, which motivates adding an integral gain K_I to the controller. The integral gain will eliminate the signal drop, and generally the system reacts better with a PI controller than a P controller, because the integral gain will grow ever larger even with small errors. Integral gain provides stiffness to the signal. That means when the error occurs, the integral gain will move to correct

it. The higher the gain is, the faster the correction. Fast correction implies a "stiffer" system; in other words, higher integral gain translates to higher signal stiffness. Do not confuse the signal stiffness with the dynamic stiffness of the system. A system can be stiff under certain conditions such as low frequencies, but not stiff at high frequencies.

Design of a PI controller as shown in Figures 21 and 22 is more complicated than a P controller. The reason for adding more gains is to avoid saturation and the "wind-up" problem [23]. Tuning the PI controller was very sensitive for high K_p gains. The best results are obtained when K_p is 0.1, and the controller performs poorly when the gain is larger.

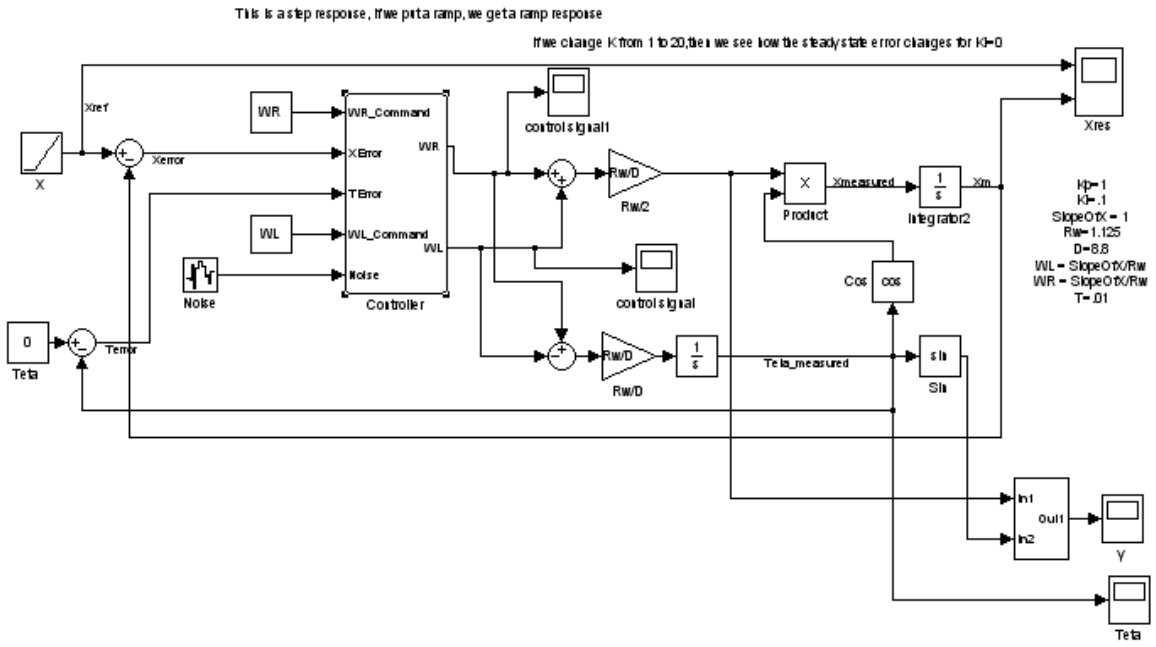


Figure 21: Simulink Diagram for PI controller

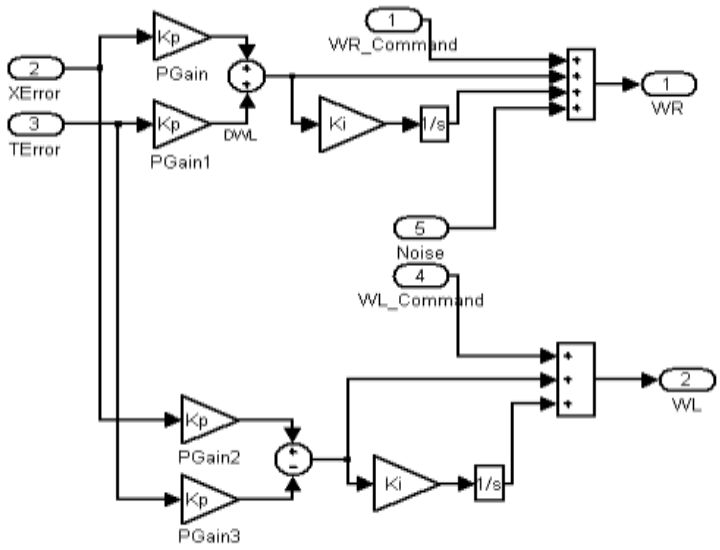


Figure 22: Simulink diagram for the integral part of PI controller

Figures 23 and 24 represent the PI Simulink controller simulation results. As seen in Figure 23 the output of X-position follows the X-reference input, which is a ramp input.

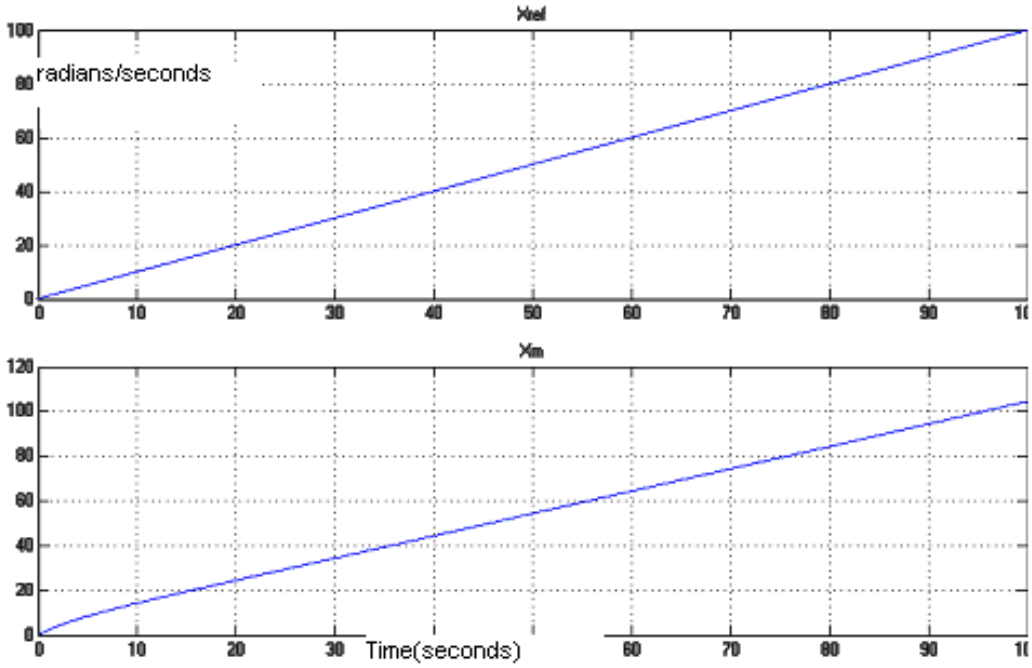


Figure 23: X-position for PI Controller

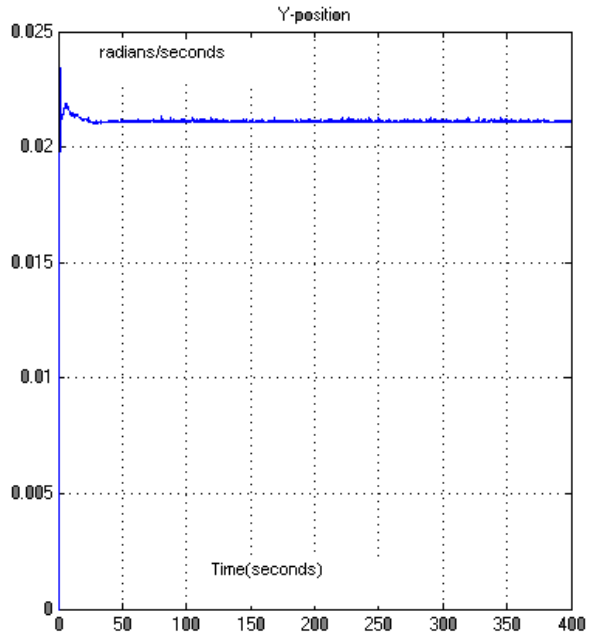


Figure 24: Y-position when the K_p gain 10 and KI gain is .1

Figure 24 shows the Y-position under the PI controller where K_p gain is 10 and KI is 0.1. As seen in Figure 24 the PI controller drive the system better than the P controller because it shows less oscillation and drives the Y-position for stability. Figure 25 shows theta, the angle for the mobile system under the PI controller. Notice that theta behaves smoothly under PI control and is close to zero, which means that the PI controller drives the mobile system in a straight line path.

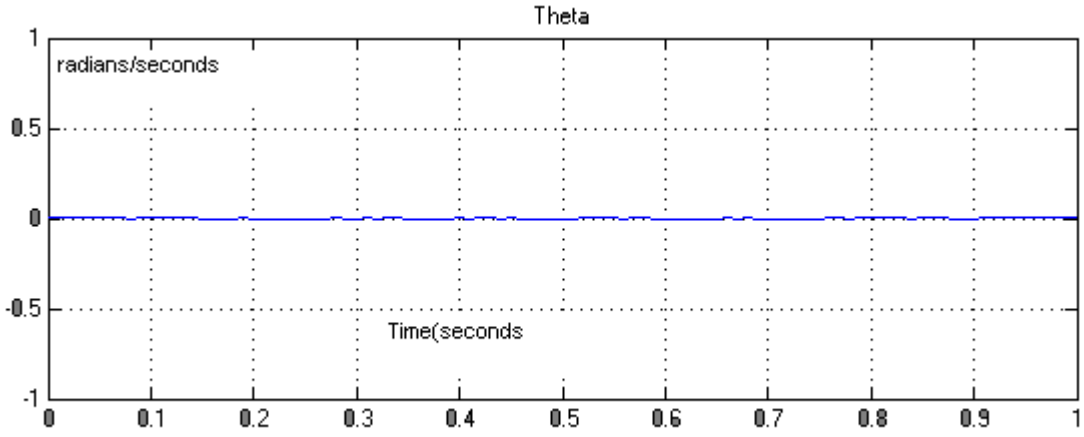


Figure 25: Theta for PI controller

In Figure 26, the system is simulated under step response input to show more clearly the steady state error for the mobile robot system under PI controller. The PI controller gains are K_p gain 10, and K_I gain 0.1.

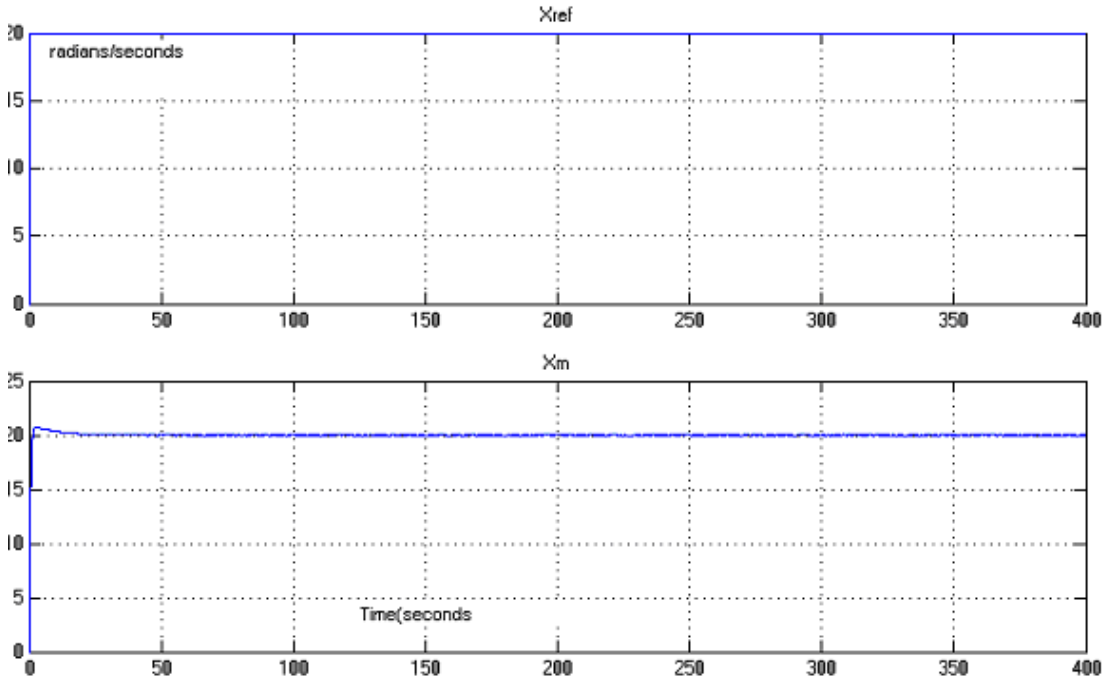


Figure 26: X-position for the PI controller the K_p gain 10 and K_I gain 0.1

After tuning the PI controller to get rid of the steady state error the K_p gain is 100 and K_I gain is 1. Figure 27 below shows the results.

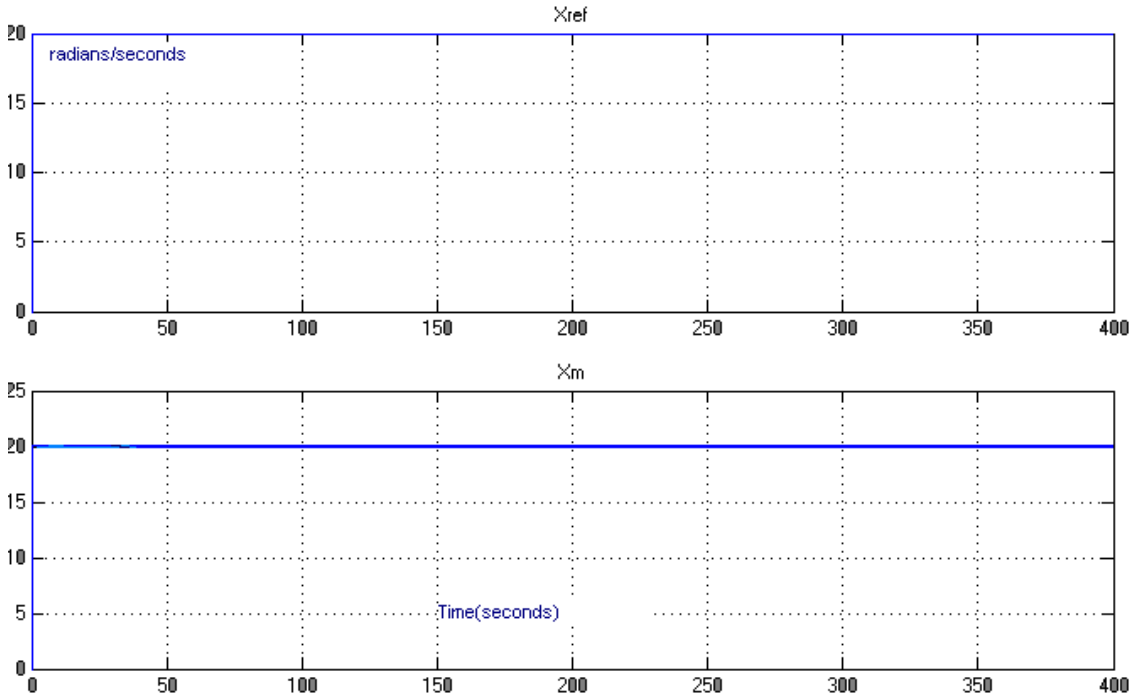


Figure 27: X-position for the PI controller the K_p gain 100 and K_I gain 1

The P controller and PI controller have been simulated with MATLAB® m-file programs. Figures 28-33 represent the simulation results of the m-file program. The m-file program is listed in Appendix A.

K_P	K_I	Y-position	Theta	Control Signals LW	Control Signals RW
1	0.1	0.0002022	0.12150	0.1158	0.1159
2	0.1	0.0008176	0.15090	0.1538	0.1509
10	0.1	8.968e-005	0.08599	0.1857	0.1857
20	0.1	5.073e-005	0.08039	0.2789	0.2792
50	0.1	2.246e-005	0.05327	0.2605	0.2607

Table IV: PI controller simulation results for mobile robot system

Table IV shows the results of PI controller simulation. The program was simulated under different K_p gains. The Standard deviations (Std) for the system variables were recorded in Table IV.

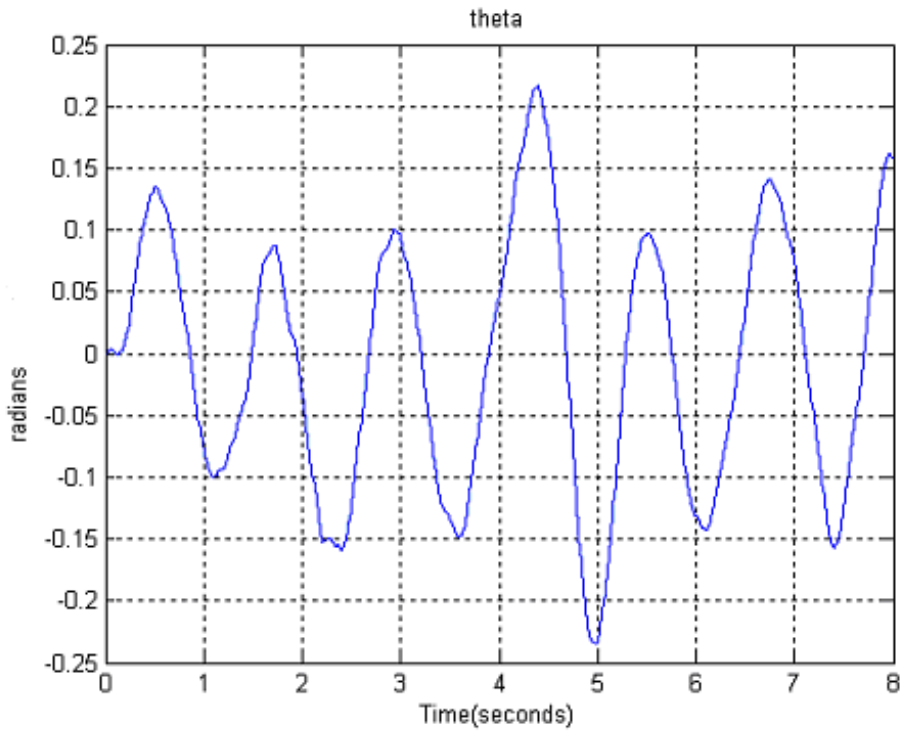


Figure 28: Theta PI controller the K_p gain is 50 and Kı gain is 1

Figure 28 shows theta for the mobile robot system where K_p gain is 50 and the Kı gain is 1. The mobile robot system was so sensitive for the Kı gain changing with the K_p changing. Figure 29 below shows the Y-position for the mobile system. The system could appear so sluggish that it doesn't move at all with tuning, but gains above show a good respond from the system and the controller drives the system for stability.

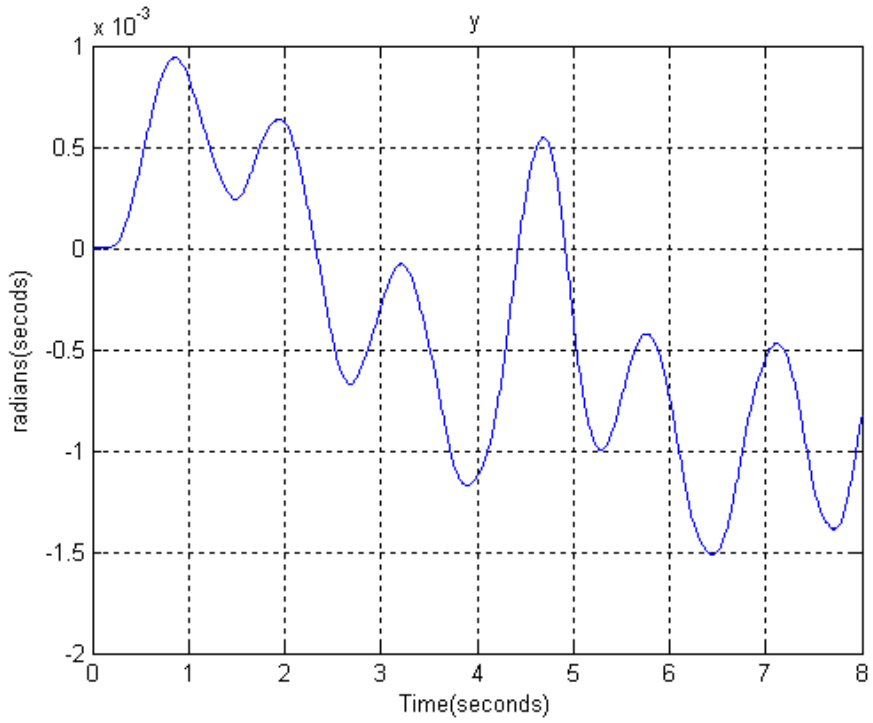


Figure 29: Y-position under PI controller

Figures 30 and 31 represent the X-reference position and the X-position for the system under PI control. Figures 30 and 31 follow each other. In other words, the controller drives the system to the desired reference. Figure 32 shows the control signals of the mobile robot. Even though we gave the same initial speed for both wheels, the PI controller failed to come out the same value, but it is very close, actually it will not be noticed physically on the mobile robot system, but comparison with the H-infinity controller results shows that the control signals for both wheels are the same.

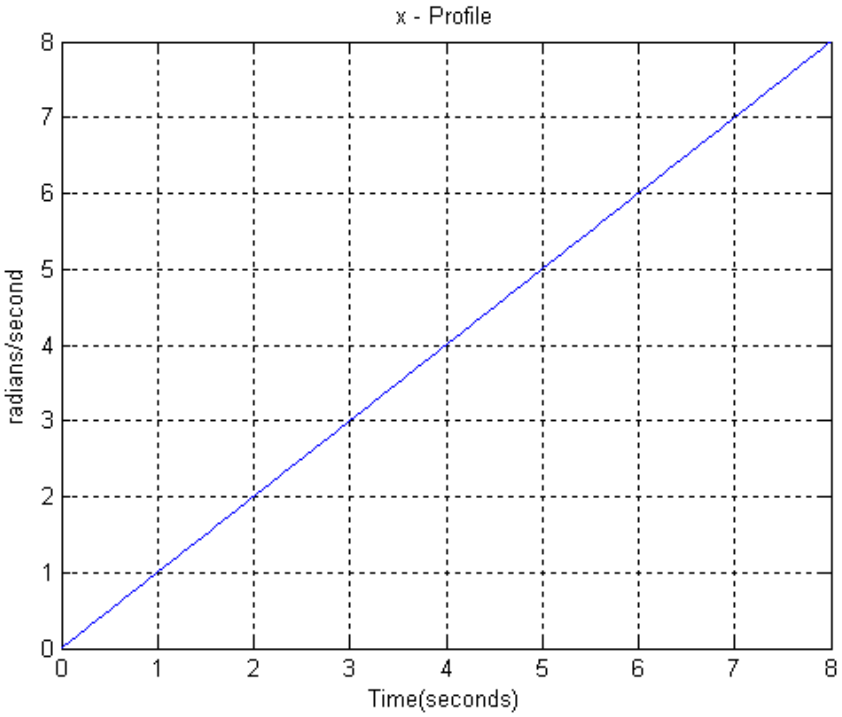


Figure 30: X-reference for X-position

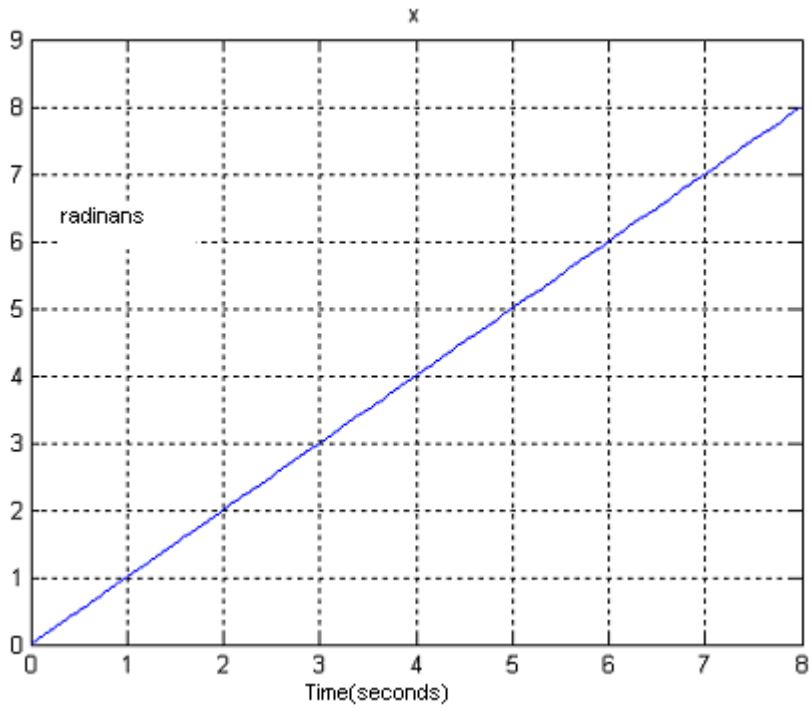


Figure 31: X-position controlled by PI controller

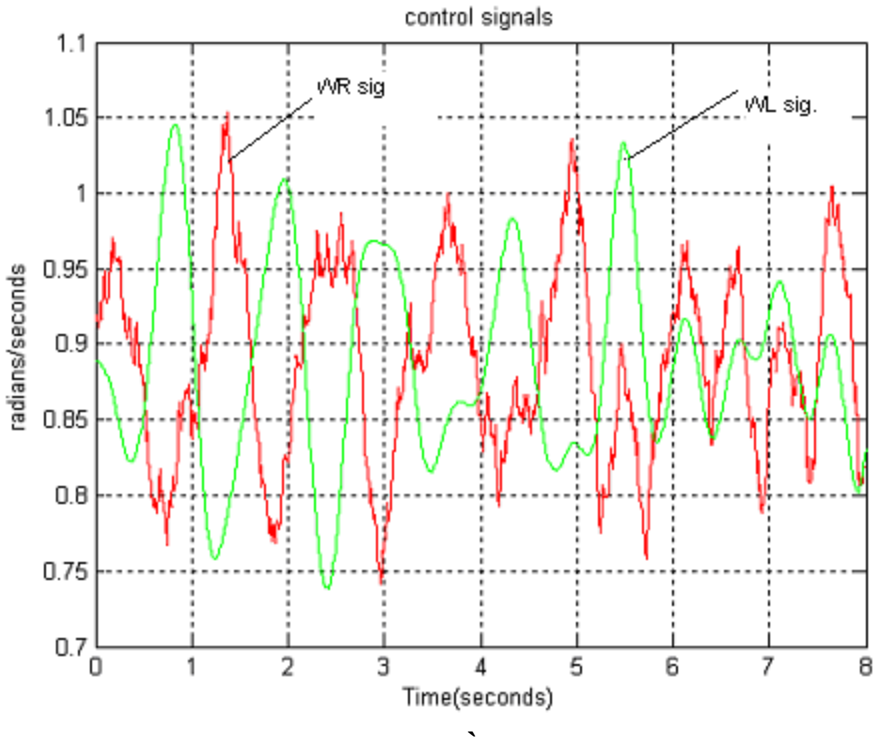


Figure 32: Control Signals for PI Controller

3.5 Differences between the H-infinity and P and PI Controllers

The performance of the H-infinity controller was better than the PI controller. But looking at the control signals for both controllers in Figures 18 and 19 for H-infinity controller and Figure 32 for PI controller, shows that the H-infinity controller signals perform better than the PI controller. This means that the H-infinity control drive the system smoothly to the desired trajectory and faster. Looking closely at the results in Table I, and Table IV, shows that the H-infinity controller fulfills the requirement of the control better than PI controller. Some values are shown in Table V below to show how much smaller the H-infinity control signals are compared to the PI controller. Comparing all the system variables shows that H-infinity controller reached the target faster than PI controller. For example the Std value for PI is 1.444 while for H-infinity it is 1.64 which means that the slope for the H-infinity is higher.

H-infinity Control Signal RW Radian/sec.	PI Control Signals LW	PI Control Signals RW
0.00303	0.1158	0.1159
0.00273	0.1538	0.1509
0.00242	0.1857	0.1857
0.00273	0.2789	0.2792
0.00242	0.2605	0.2607

Table V: Comparison between Control Signals for PI, H-infinity controllers

The H-infinity controller may be preferred for sophisticated systems that demand more accuracy and faster performance, such as a robotic surgeon. The PI controller seems more applicable to industrial systems for cheaper control signals and for quantity (not quality) products.

CHAPTER IV

MOBILE ROBOT CONSTRUCTION

The physical design of the mobile robot "ROCK" is very effective and simple. It is a combination of various physical (hardware) and computational (software) elements that integrate the subsystems of the mobile robot to work in one unit. In terms of hardware components, ROCK is constructed mainly from these components: microcontroller (PIC16F877), two stepper motors, ultrasonic sensor and LCD (liquid crystal display). Assembly programming is designed to run the mobile robot tasks and control its behavior. Section 4.1 describes the microcontroller (PIC16F877) and its features. Section 4.2 discusses the stepper motor circuit and how it is interfaced with the microcontroller. Section 4.3 explains the ultrasonic sensor circuit. Section 4.4 explores the LCD circuit. The final design of the mobile robot system is in Section 4.5. Section 4.5.1 explains the software behind ROCK, and Section 4.5.2 talks about the problems on the mobile robot design.

4.1 Microcontroller PIC16F877

One of the most popular and easy to use microcontrollers today is the Microchip "PIC micro®" microcontroller. The Peripheral Interface Controller (PIC) has a lot of variations, each designed to be optimal in different applications. These variations consist of a number of memory configurations, different I/O pin arrangements, amount of support hardware required, packaging and available peripheral functions. The primary role for these microcontrollers is to provide inexpensive, programmable logic control and interfacing to

external devices. Thus, they typically are not required to provide highly complex functions. They are capable of providing surprisingly sophisticated control of different applications. When I say that the devices are inexpensive, I mean that they range in cost from $1 to $20 each depending on complexity, internal and external features, and the quantity purchased. All of these features increase the flexibility of the device considerably and make developing applications easier. The brain of the mobile robot is the PIC16F877 microcontroller. Figure 33 shows the pin diagram for the PIC16F877 [33].

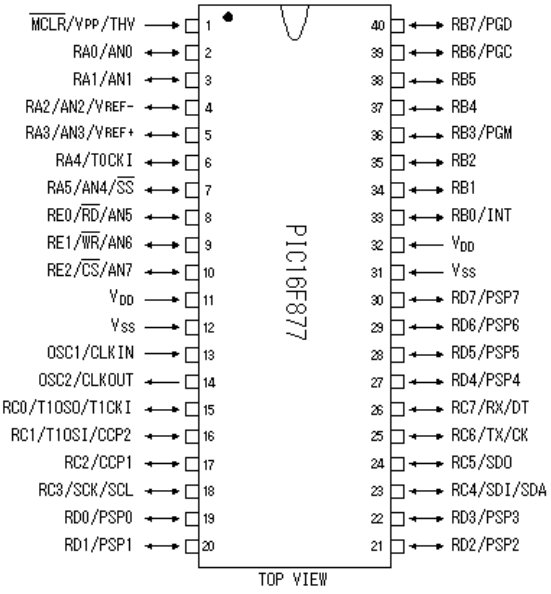


Figure 33: Top View of PIC16F877

The PIC16F877 has a lot of features as summarized in Table VI. Also the PIC has an eight level deep hardware stack, direct, indirect and relative addressing modes, single 5 V in-circuit serial programming capability, read/write to program memory, low power consumption, and high sink/source current which is 25 mA per pin.

MAX Operating Frequency	20MHz
FLASH Program Memory (14-bit words)	8K
Data Memory (bytes)	368
EEPROM Data Memory (bytes)	256
I/O Ports	RA0-5 (6) RB0-7 (8) RC0-7 (8) RD0-7 (8) RE0-2 (3)
Timers	3
CCP	2
Serial Communications	MSSP, USART
Parallel Communications	PSP
10-bit Analog-to-Digital Module	8 Channels
Instruction Set	35 Instructions
Pins (DIP)	40 Pins
interrupt capability	14 sources

Table VI: PIC16F877 Features

Figure 34: PIC 16F877 Chip Package

Figure 34 shows the PIC16F877 chip [33]. Each processor may have different versions. These versions are identified by letters after the part number.

Microchip's MPLAB integrated development environment (IDE) is development tool for creating the projects that fulfill the applications we need. It is capable of integrating all of the software development tasks which are provided in a source code level. MPLAB IDE provides functions that allow us to:

Create and edit source files

Group files into projects

Debug source code

Debug executable logic using the simulator or emulator

Figure 35 shows the setup of the mobile robot system and MPLAB and also shows the PIC16F877 development kit.

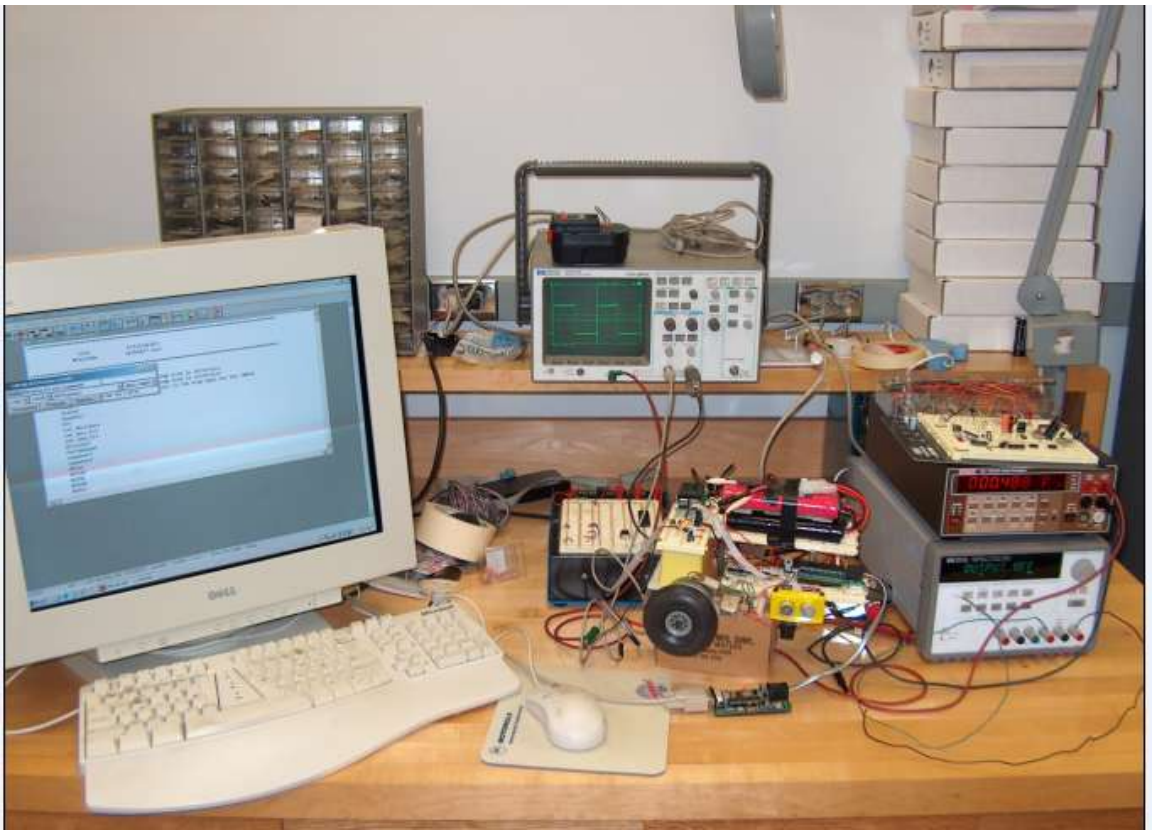


Figure 35: Mobile Robot Setup for Programming

We create our source code on MPLAB, then compile it and download it to the microcontroller chip through an RS-232 serial cable connected between the computer and the PIC16F877development kit. To enable in-circuit debugging, the debug code residing in the microcontroller in the MPLAB ICD module is programmed into the target PIC16F877. The code is an MPASM assembler module that will be programmed into the PIC16F877 through the MPLAB ICD header automatically by the MPLAB IDE. The demo board is provided for demonstration and evaluation of the PIC16F877 without a target application board. It can be connected to the MPLAB ICD module directly or via the MPLAB ICD header. The PIC16F877 can be unplugged from the header and plugged directly into the demo board for stand-alone operation.

4.2 Stepper Motors Circuit

A stepper motor is a unique type of DC motor that rotates in fixed steps of a certain number of degrees. Step size can range from 0.9° to 90°. The basic stepper motor consists of a rotor and stator. Generally the rotor is a permanent magnet, and the stator is made up of electromagnets (field poles). The rotor will move (or step) to align itself with an energized field magnet. If the magnets are energized one after the other around the circle, the rotor can be made to move in a complete rotation. Stepper motors are particularly useful in control applications because the controller can know the exact position of the motor shaft without the need of position sensors. This is done by simply counting the number of steps taken from a known reference position. Step size is determined by the number of rotor and stator poles, and there is no cumulative error, which means that the angle error does not increase

regardless of the number of steps taken. In fact, most stepper motor systems operate under open loop control. That is, the controller sends the motor a determined number of step commands and assumes the motor goes to the right place. Steppers have inherently low velocity and therefore are frequently used without gear reductions. There are three types of stepper motors: permanent magnet, variable reluctance, and hybrid. All types perform the same basic function, but some differences among them may be important in some applications.

Stepper motors can't deliver as much running torque as standard DC motors of the same size and weight. A typical 12 volt, medium sized stepper motor as shown in Figure 36 (which is the same kind of motors as used in the mobile robot ROCK) may have a running torque of only 25 oz-in, while DC motors may have a running torque three times more. For open loop control to work, the motor must actually step once each time it is commanded. If the load is too great, the motor may not have enough torque to make the step. In such a case, the rotor would probably rotate a little when the step pulse was applied but then fall back to its original position. This is called stalling.

Figure 36 shows one of Rock's stepper motors [48]. It is 55M 048D Series stepper motor with a 7.5° step angle or 48 steps per revolution, 4.8 watts input power per winding, and 12 volts DC operating voltage. It is a four phase unipolar stepper motor.

Figure 36: Mobile Robot Stepper Motor

The term four-phase unipolar is used to describe Rocks stepper motors because the motor has four field coils that can be energized independently, and the term unipolar is applied because the current always travels in he same direction through the coils. The unipolar stepper motors usually have 5 or 6 wires as shown in the schematic in Figure 37. In Figure 36, the stepper motor has 6 wires, with a center tap on each of two windings that is typically wired to the positive supply. Motor winding number 1 is distributed between the top and bottom stator pole, while the motor winding 2 is distributed between the left and right motor poles. The rotor is a permanent magnet with 6 poles, 3 south and 3 north, arranged around its circumference. As shown in Figure 37 [44], the current flowing from the center tap of winding 1 to terminal *a* causes the top stator pole to be a north pole while the bottom stator pole is a south pole. This attracts the rotor into the position shown. If the power to winding 1 removed and winding 2 energized, the rotor will turn 30 degrees, or one step. For higher angular resolutions, the rotor must have proportionally more poles. The 30

degree per step motor in Figure 36 is one of most common permanent magnet motor designs, although 15 and 7.5 degree per step motors are also widely available.

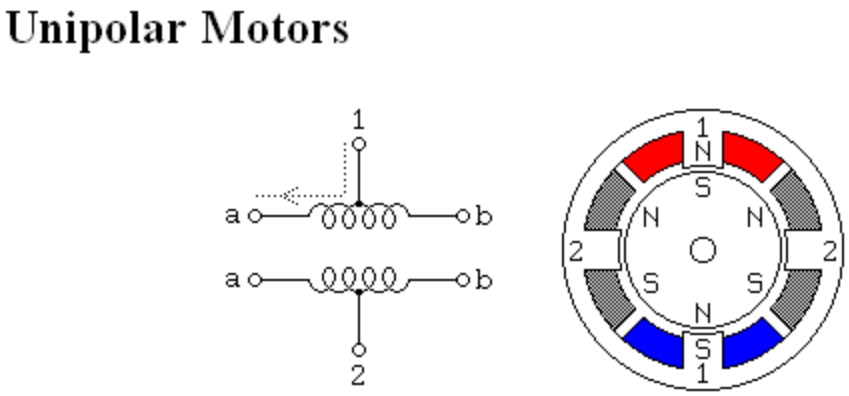


Figure 37: Unipolar Stepper Motor

To rotate the motor continuously, power should be applied to the two windings in sequence. To make the motor step, power is applied to each coil in turn. The four windings have to be energized in the right sequence. Stepper motors have three stepping methods: wave, two phase, and half step. This is because there are three basic patterns of energizing the coils to make them move. The last two are the most efficient.

The stepper motor driver UN5804 Allegro Chip in Figure 38 [31] is one of those marvelous devices that replace a handful of discrete components. The driver will operate the mobile robot motors at up to 35 V and 1.25 A. The step input is to pin 11 and the direction command goes to pin 14. Pins 9 and 10 control one phase and half step operation respectively.

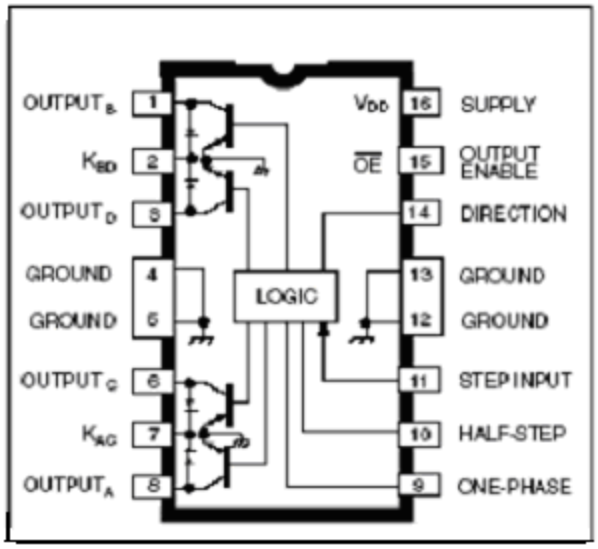


Figure 38: UCN5804 Stepper Motor Driver

The outputs will advance one sequence position on the high-to-low transition of the STEP INPUT pulse. The Direction pin determines the rotation sequence of the outputs. Figure 39 [31] shows the unipolar motor connected to the driver UN5804 stepper motor. The sequence is connected to pins 1, 3, 6, and 8 and each pin is connected with a diode to allow an escape path for current in the coil when the transistor is turned off. Pin 14 is the direction pin and pins 2 and 7 are connected to the power supply of the motor while pin 16 is connected to the 5 V power supply.

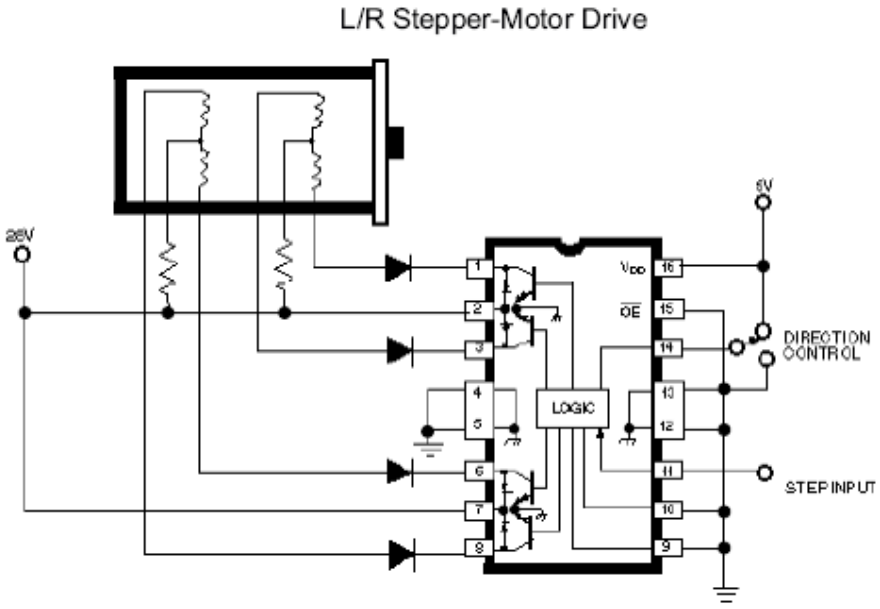


Figure 39: Stepper Motor Translator/Driver

Table VII [31] shows the motor response to the two-phase drive sequence, a dual-excitation mode where two adjacent phases are energized at the same time to give more torque. The mobile robot was driven for a while in this mode, and then we started facing power consumption problems and torque deficiency problems. Then we changed the mode of operation to a half-step drive sequence where the operation alternates between single-and dual excitation modes yielding eight half steps per cycle. This is shown in Table VIII [31].

TWO-PHASE DRIVE SEQUENCE

Half Step = L, One Phase = L				
Step	**A**	**B**	**C**	**D**
POR	ON	OFF	OFF	ON
1	ON	OFF	OFF	ON
2	ON	ON	OFF	OFF
3	OFF	ON	ON	OFF
4	OFF	OFF	ON	ON

Table VII: Two-Phase Driver Sequence

HALF-STEP DRIVE SEQUENCE

Half Step = H, One Phase = L				
Step	A	B	C	D
POR	ON	OFF	OFF	OFF
1	ON	OFF	OFF	OFF
2	ON	ON	OFF	OFF
3	OFF	ON	OFF	OFF
4	OFF	ON	ON	OFF
5	OFF	OFF	ON	OFF
6	OFF	OFF	ON	ON
7	OFF	OFF	OFF	ON
8	ON	OFF	OFF	ON

Table VIII: Half-Step Drive Sequence

Figure 40 shows the timing diagram for different modes of operation for the unipolar motor that the mobile robot was built with [31]. It shows one phase excitation, half step, and two phase. As it is mentioned before we tried all modes to drive the stepper motor to get the best torque and resolution and the final motor drive is half step mode.

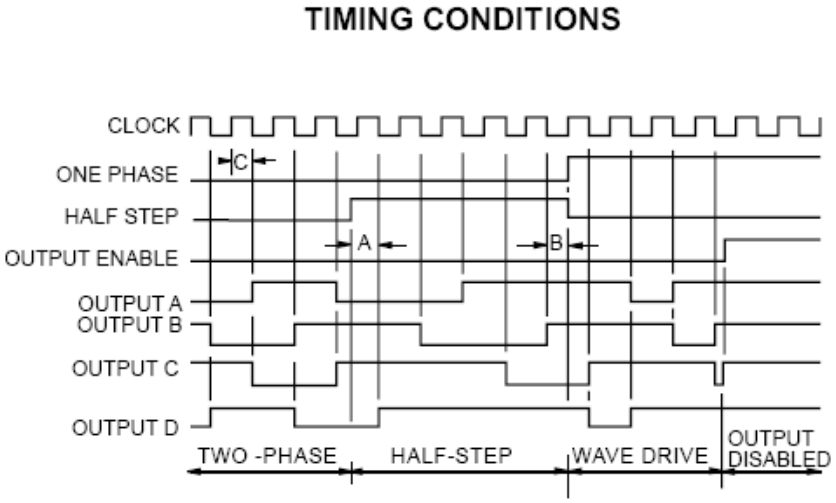


Figure 40: Timing Pulses for Different Modes

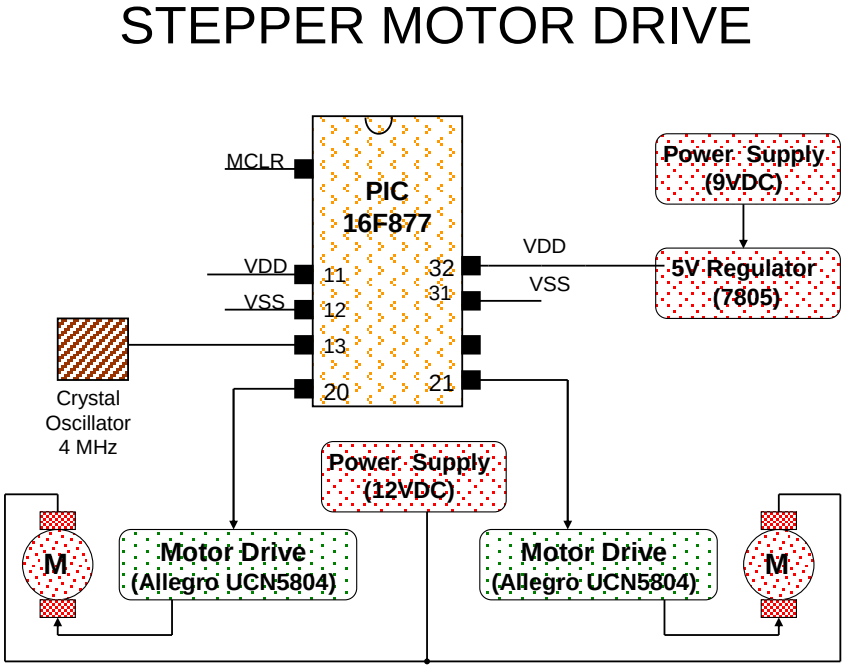


Figure 41: Stepper Motor Circuit

Figure 41 shows the stepper motors of the mobile robot and the PIC16F877 microcontroller. The PIC is connected with the stepper motor drive chip Allegro UCN5804, each motor has one chip, and at the same time it is connected with a 9 V power supply that is regulated through a 5 V regulator 7805 and external oscillator.

The motor program in the appendix explains the procedure for controlling the stepper motors of the mobile robot Rock. The program generates a pulse through Timer 2 and feeds the pulse from pin 19 on the PIC which is PORT D 0 to the stepper motors drive chip UCN5804 at pin 11. To control the direction for both motors we change PORTD 1 and PORTD 2 from low bit status to high bit status to change the rotation sequence of the outputs. To control the Output Enable bits the program used PORTD 3 and PORTC 4 to change the bits status from high to low which means the motors turn off or on.

After controlling the sensor of the mobile robot and the LCD and integrating the system as a whole unit, the program discovers the need for more timers if it needs to control the speed of the motors and implement the proportional controller on the mobile robot. Due to this reason the mobile robot generates the step input for one motor from an outside Timer which is a 555 chip. The 555 timer circuit is shown in Figure 42

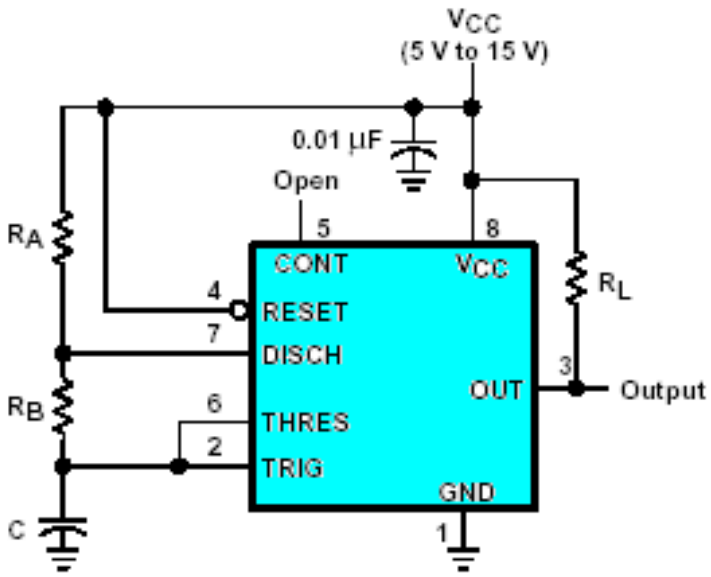


Figure 42: Timer 555 Circuit

4.3 Ultrasonic Sensor

Figure 43 below shows the ultrasonic sensor SRF04 in operation [33]. An ultrasonic burst of energy is emitted from the transducer. This is known as a ping. The sound waves travel until reflected off of an object. The echoed sound wave then returns to the transducer. The echo may be of smaller amplitude, but the carrier frequency should be the same as the ping, and if there is an external timer to record the time of flight (the time that the sound waves take to travel to and from the object), the time can be converted to distance when

considering the speed of sound in air. As the transmitter sound waves propagate from the transducer, they spread over a range in the shape of a cone of angle . The range of is 45° on both sides of the sound wave.

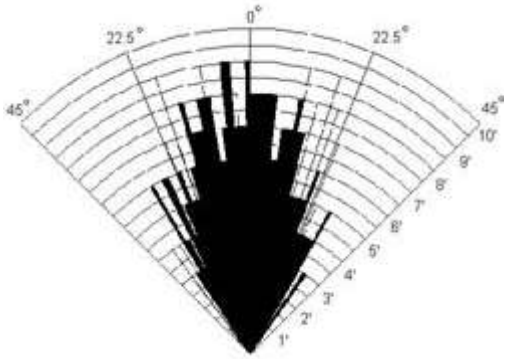


Figure 43: The Ping Wave for Ultrasonic Sensor

The Devantech SRF04 Ultrasonic Range Finder is unique in that it has a separate transmitter and receiver. This allows for the smallest minimum detectable distance. The frequency of the ping is 40 KHz. The reason for a 40 KHz frequency sound wave is to reduce the chances of false echoes. An advantage of the SRF04 is that the frequency generating circuit is integrated into the printed circuit board supporting the transducers. This minimizes the amount of wiring needed to utilize the sensor. The SRF04 offers precise ranging information from roughly 3 centimeter to 3 meters. This range and minimal power requirements, 5 volts, make this an ideal ranger for robotics applications.

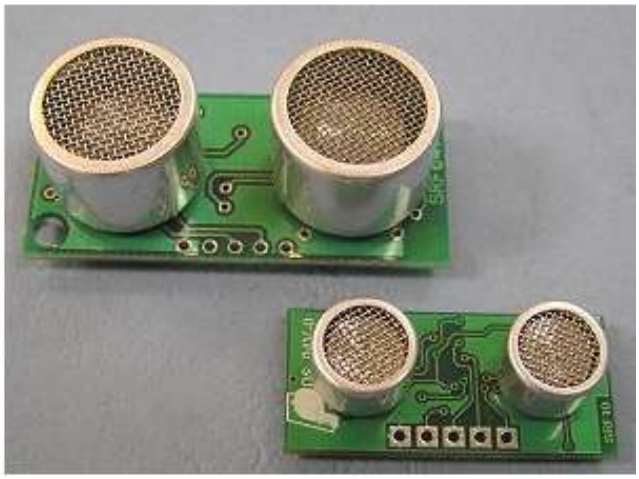


Figure 44: Ultrasonic Sensor transducers

Figure 44 shows the top layer of the sensor board with transmitter and receiver transducers [33].

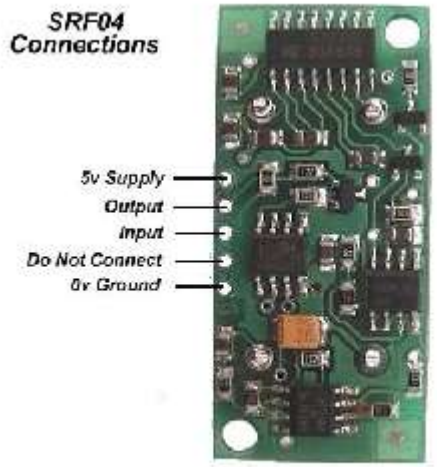


Figure 45: The Pin Connections

Figure 45 [33] shows the pin connections for the sensor, the SRF04 requires four connections to operate. First is the power and ground, the sensor SRF04 requires a 5V power supply capable of handling roughly 50 mA of continuous output. The remaining two wires

are the trigger pulse in and the echo pulse out. These two lines shall be connected directly to the PIC16F877 PORTB 3 and 4, which are pins 36 and 37 in the PIC.

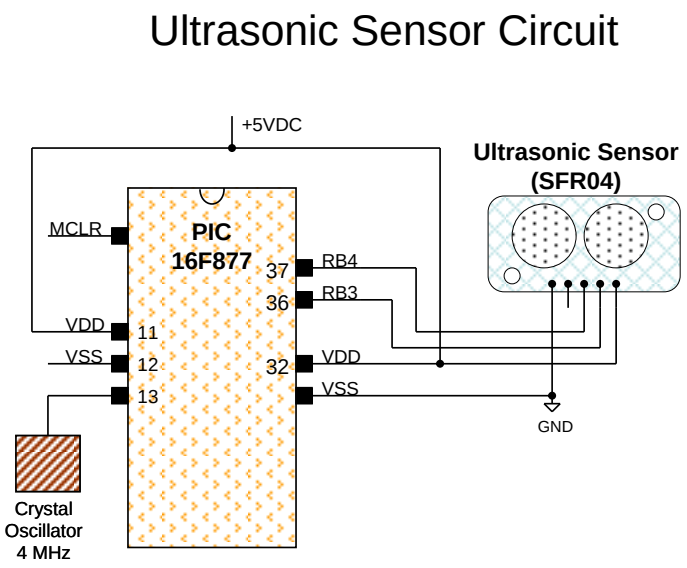


Figure 46: The Ultrasonic Sensor Circuit

Figure 46 depicts the basic schematic of how one should go about connecting the SRF04 sensor to the PIC16F877 microcontroller and oscillator. The basic operation between these I/O lines is as follows. The echo pulse line will output the measured distance in the form of a varied length square wave form upon the initial pulsing of the input trigger line. In other words, we should trigger the sonar sensor to send out an echo pulse to detect object distance. There are a couple of requirements for the input trigger and output pulse generated by the SRF04. The input line should be held logic low and then brought high for a minimum of 10 usec to initiate the sonic pulse. In order to generate the trigger pulse with the PIC, an output high pulse must be generated via a code pulse. Another subroutine is required for the 10 usec delay for the trigger pulse, and another 10 usec delay is required for the 8 cycles

between the falling edge of the trigger and the rising edge of the echo. The echo pulse is generated on the falling edge of the input trigger. The ranger's receiver circuitry is held in a short blanking interval for a minimum of 100 usec to avoid noise from the initial ping and then it is enabled to listen for the echo. The echo line is low until the receiver circuitry is enabled. The falling edge of the echo line signals either echo detection or a time-out detection which signals that there is no object or the object is out of sensing range (3 cm 3 meters). Figure 47 shows the timing specifications for the ultrasonic sensor SRF04 [33].

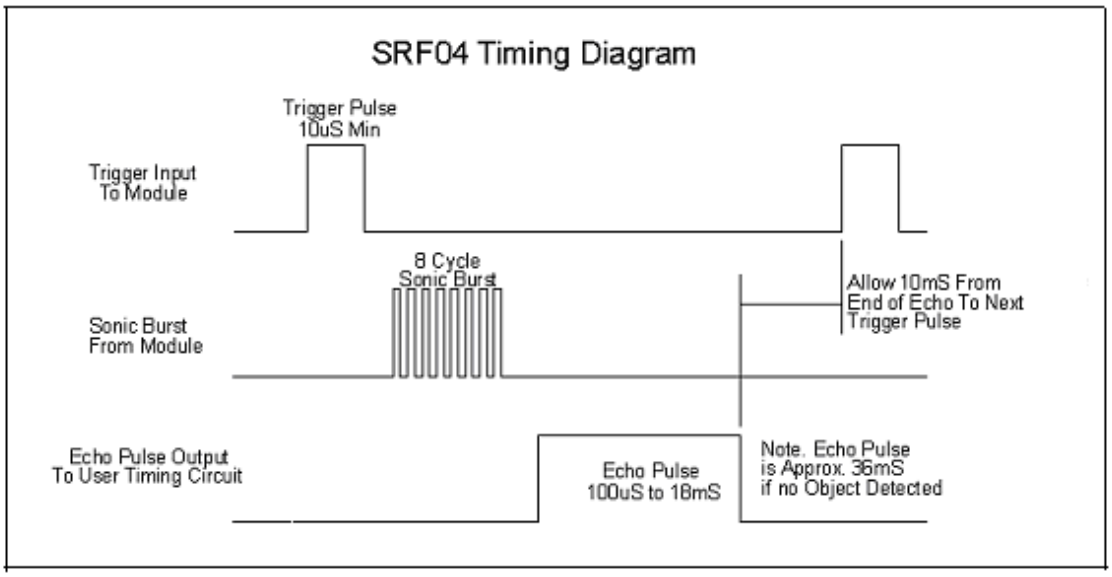


Figure 47: Timing Specification

For safe measure, the sensor program makes sure the trigger pulse would not trigger again for another 5 msec. These delays were also coded into the PIC microcontroller program. For measuring distance the PIC can be coded to start timing on the falling edge of the trigger input and end timing on the falling edge of the echo line. This duration determines the distance to the first object the echo is received from. The Ultrasonic Sensor program is found in the appendix.

4.4 The LCD0821 Display

The LCD0821 is designed as the display unit for the mobile robot (ROCK). It is connected with the PIC16F877 microcontroller to show the measurement distance. The LCD0821 display has the following features:

8 colum by 2 line text display

Built-in font with provision for up to 8 user-defined characters

Speeds from 1200 bps to 19.22 kbps over RS-232

Communicate over RS-232 or I^2C

Use up to 16 modules on the same 2-wire I^2C interface

Software controlled contrast

Backlight with configurable time-out stetting up to 180 minutes

One general purpose output for a variety of applications

Horizontal or vertical bar graphs

Variable power options: +5 V, +7 V to +15 V

Extended temperature option.

Figure 48 shows the electrical connection for each application [35]. A close look at the PCB board, it shows that the board has 2 pins for general purpose connection, 3 pins for power and I^2C communications, and 4 pins for RS-232/power, and jumpers to set the RS-232 speed and I^2C address. These jumpers J1 and J2 are installed to give a speed of 19200 bps in our case. The screen of the display which is shown in Figure 49 is backlit for low-light situations. Backlighting may be turned on or off under program control. Contrast is adjustable to compensate for differing conditions and viewing angles.

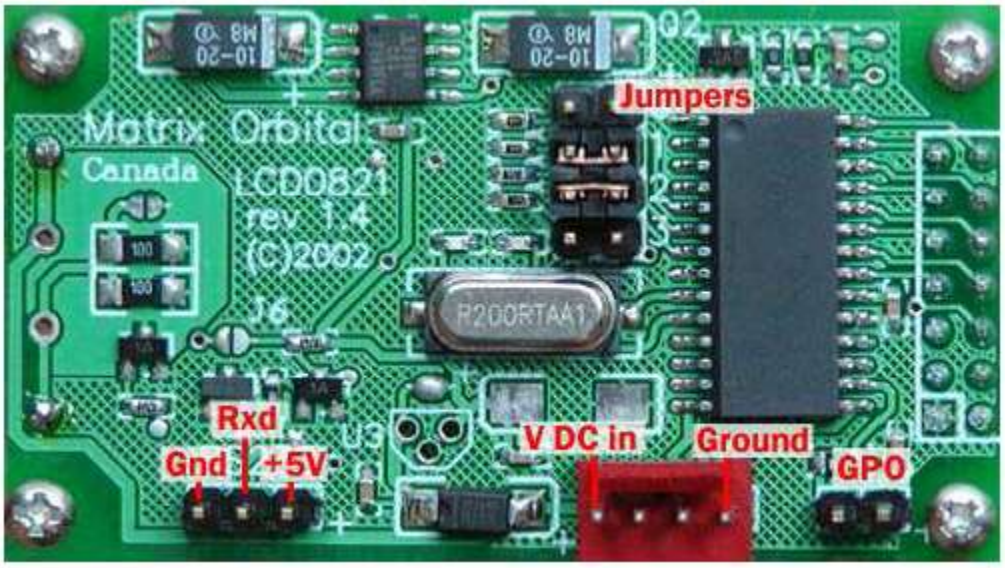


Figure 48: PCB for the LCD connection Pins

Figure 49: Front image for the LCD0821

The LCD0821 display is connected to the PIC with a Max202 transceiver chip whose role is to transmit the ASCII code from the PIC to the LCD.

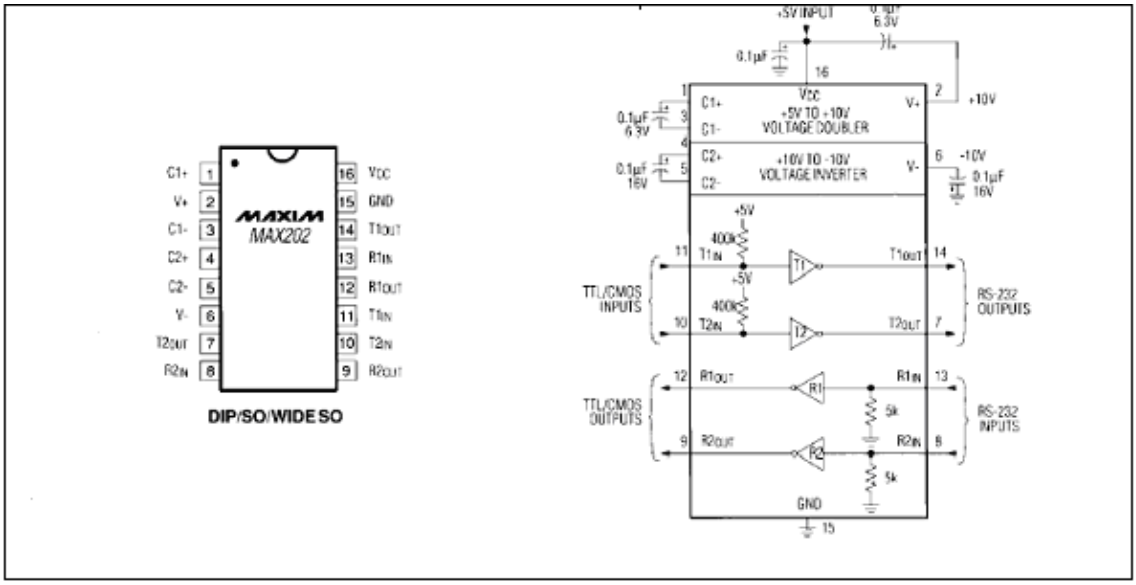


Figure 50: Max202 Pin Configuration and the Operating Circuit

Figure 50 shows the Max202 pin configuration and the operating circuit [35]. The Max202 consists of three sections: charge-pump voltage converters, drivers (transmitters), and receivers. In our application we use the Max202 to receive the serial data from the PIC and transmit to the LCD. The Universal Synchronous Asynchronous Receiver Transmitter (USART) module is one of the two serial I/O modules that the PIC has. The USART is also known as a serial communications interface or SCI. The USART can be configured in the following modes: asynchronous (full duplex), synchronous master (half duplex) and synchronous slave (half duplex). In this mode, the USART uses standard non-return-to-zero (NRZ) format (one start bit, eight or nine data bits, and one stop bit). The most common format data is 8 bits; this is what Rock's LCD display implements. The USART transmits the data using the PIC's transmit (serial) shift register (TRS). The shift register obtains its data from the read/write transmit register TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the STOP bit has been transmitted from the previous load. Once the TXREG register transfer the data to the TSR register then it becomes

empty and the TXIF (PIR1<4>) flag bit will be set. This interrupt can be enable/disabled by setting/clearing enable bit TXIE. The actual transmission will not occur until the TXREG register has been loaded with data and the baud rate generator (BRG) has produced a clock shift.

Figure 51 shows the circuit for the LCD connected with the PIC. Pin 25 in the PIC (RC6/TX) is connected to the input of the MAX202 pin 11, while the output of the MAX202 (pin 12) is connected to the serial input on the LCD. The program for LCD display is in the appendix.

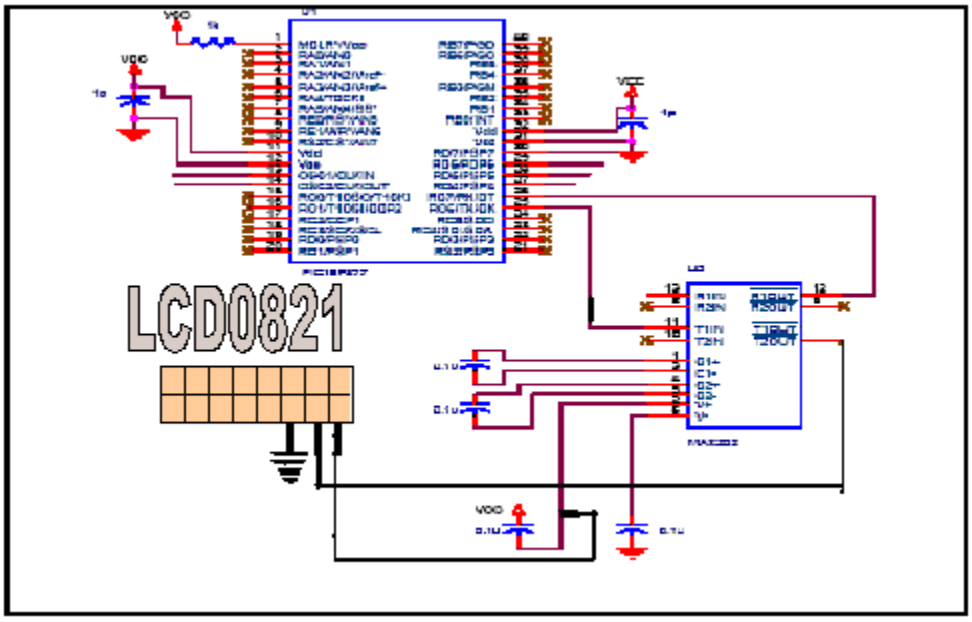


Figure 51: LCD Display Circuit

4.5 The Mobile Robot System (ROCK)

The main objective of this part of the thesis is to build a mobile robot system at the cheapest price possible. The total cost of the mobile robot assumes that the final design will not require more than a few hundred dollars to be complete. Table IX shows the complete

list of components that used to build ROCK and a rough estimation of the cost. The total cost of the mobile robot was $424.60. Compared to other mobile robots with its quality and functionality, ROCK is a very cheap mobile robot. If we add the cost of the components damaged and replaced during development, the total price of the mobile robot comes to $574.60.

Part	Price of the Unit $	
Two Stepper Motors	8.90	29.60
Two Wheels	8	16
Plastic Chasse	8	8
Ultrasonic Sensor	44	44
LCD Display	59	59
Two Allegro chips	8	16
Maxim Max202	8	8
Microchip PIC 16F877	10	10
PIC 16F877 Development Ki	150	150
Timer 555 Chip	6	6
Capacitors, Diodes, Resistors	5	5
Crystal Oscillator	3	3
Four Breadboard	8	32
Two Batteries 7.2 Volts	8	16
9V battery, Package of Two	6	6
Connector	6	6
Miscellaneous	0	10
Total price		**424.60**
Damaged and Replaced Components		**150.00**
Total price		**574.60**

Table IX: Mobile Robot Cost

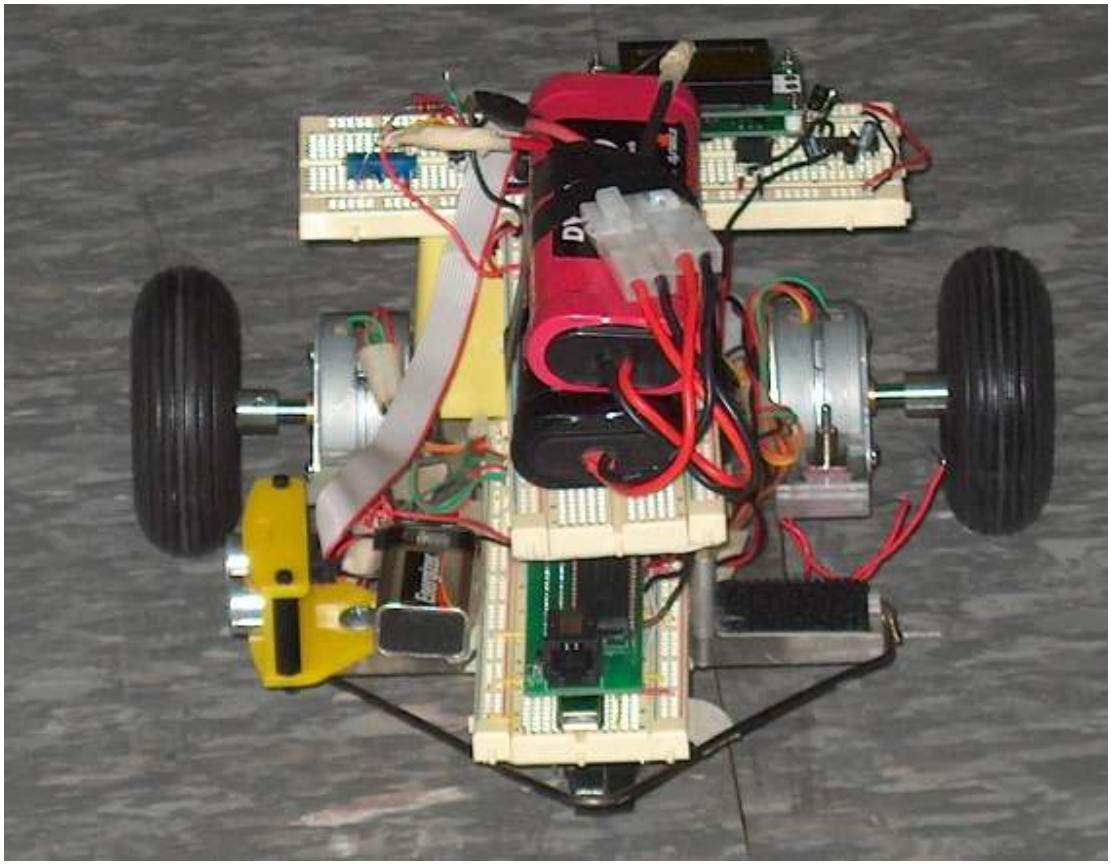


Figure 52: Hardware of Mobile Robot

Figure 52 shows the hardware of the mobile robot ROCK.

4.5.1 The Software behind ROCK

The software behind the robot behavior is very simple and direct. It depends on a closed feedback loop, monitoring the distance from the wall by the ultrasonic sensor and correcting the measured distance by a control algorithm to make the mobile robot follow the wall in a straight line. Figure 53 shows the flowchart of the software for ROCK.

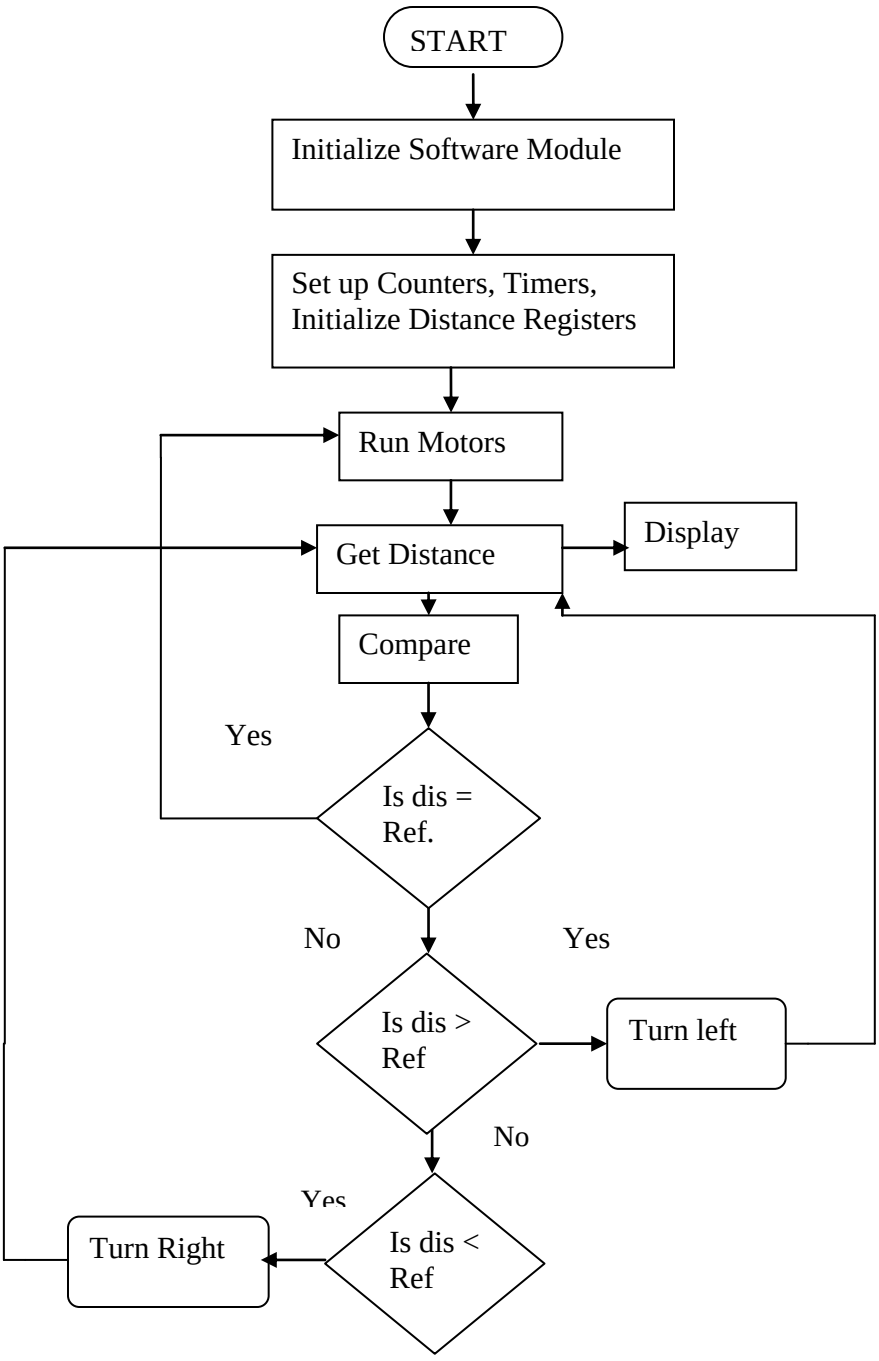


Figure 53: Flowchart of Mobile Robot Software

In Figure 53, the ultrasonic sensor measures the distance between the mobile robot and the wall. If the measurement distance in higher than the reference distance, the mobile robot should get closer to the wall which means turn right. If the measurement distance is lower then the reference distance, then the mobile robot should go farther from the wall which means turn left.

The mobile robot is programmed to drive with its left side near a wall, following the wall with a constant distance from it. The program orients ROCK exactly parallel to the wall, and then drives it straight ahead. Although this solution looks very simple, it has a lot of obstacles facing the mobile robot navigation. Mechanical failures and sensor source limitation contribute to the failure of this simple approach.

4.5.2 Problems with the Mobile Robot Design

In general, the most robot failure is not mechanical or electrical failure, but rather involves the software that controls the robot. For example, if the robot should follow the wall, and its sensor did not trigger, the robot would become stuck, and lost. In this situation, the robot is not physically stuck, but rather it is "mentally stuck" Its control program does not account for this kind of failure and does not provide another way or solution for the robot to accomplish its task. Many robots fail in this way. But in ROCK the situation is different. The limitation of the hardware puts more emphasis on the software and it is very difficult to develop software that is able to deal with the limitation of the hardware.

The straight line algorithm fails to drive the mobile robot for a lot of reasons. In Figure 54 (the mobile system diagram) shows one of the wheels is controlled by an external

timer (555 chip) while the other wheel is controlled by the PIC through Timer 2. Feedback control is implemented on one wheel. The other wheel is supplied with a constant speed through the 555 Timer. But practically, the timer changes the pulse periods with the heat of the chip, which means there is constantly imbalance between the two wheels.

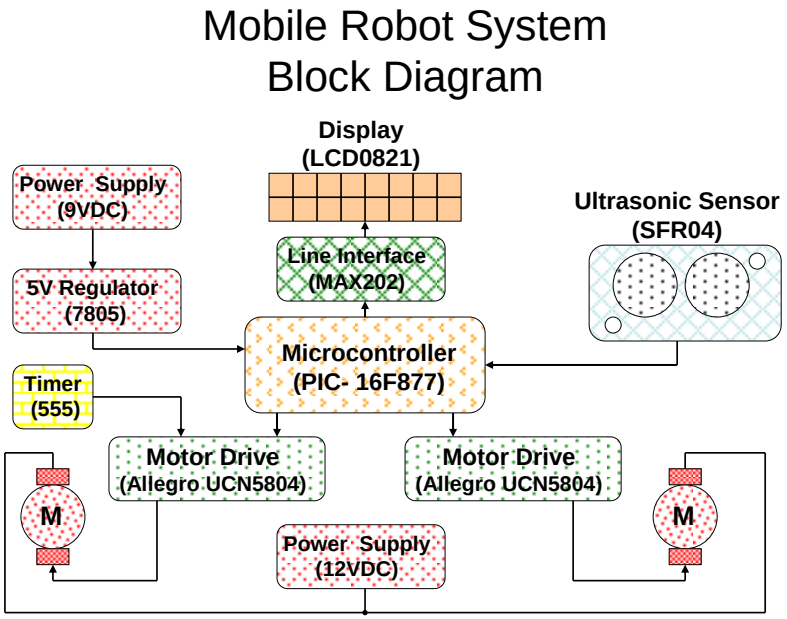


Figure 54: Diagram of Mobile Robot System

Even though someone will argue this algorithm should work and accomplish its goal, if the mobile robot attempt to drive straight, the robot will actually veer a bit toward the wall, then after driving the robot for a bit, the control algorithm would drive the robot too close to the wall, or too far from the wall. In ROCK's case, the Timer2 rate of change is not linear, which forces the feedback algorithm to constantly correct the change.

Another difficulty is the accuracy of the ultrasonic sensor reading for the distance to the wall. Sometimes the sensor becomes blind due to its assembly on the mobile robot. To solve this problem, calibration of the sensor is required to get the best reference distance for

good robot performance. The best distance is found to be 43 cm, where the sensor shows a constant reading, and this where ROCK's software initializes the reference distance.

Another problem which affects the robot is the lack of a power source to feed all the systems, especially the two stepper motors. The two stepper motors consume 1.2 amperes, which a lot of current to keep feeding the system. This current drain quickly depletes the battery power. Also, the motors can not handle the required load torque for the robot. If the stepper motors has been changed for more powerful ones, the new motors need more power and then we constantly need to charge the batteries to test the mobile robot on the floor. The lack of power adversely affects all the other parts of the system.

CHAPTER V

CONCLUDING REMARKS

5.1 Conclusions

Two main objectives of this thesis were achieved. The first objective was to provide a practical approach for robotic technology development, to explore the basic principles of sensory systems, data acquisition systems, and actuation systems. The second objective was to apply modern control technologies such as control and compare with traditional controllers such as proportional and proportional integral controllers.

Therefore, the theoretical aspect was established where the nonlinear and the linear system models were developed mathematically and simulated with MATLAB®. An H-infinity, *P*, and *PI* controller were simulated and compared. The second aspect of this thesis was the practical part where a mobile robot was built as an embedded system, which includes integration of hardware and software in its design and operation. The hardware includes the physical parts of the system, which include a microcontroller (PIC16F877), two stepper motors, an ultrasonic sensor and an LCD. Assembly programming language is designed to run the mobile robot tasks and control its behavior.

The main purpose of this thesis is to gain practical experience with control schemes and embedded systems interfaces. Trouble shooting of electrical circuits, designing mechanical systems for a mobile robot, and implementing software on the system, is required to make it function. Exposure to the problems that accompany this project expanded the practical experiences and made any electrical engineer gain a lot of insight on his/her career path.

5.2 Future Work

This thesis creates possibilities for a lot of research and applications, because mobile robotics is a fertile research area that deals with the control of both autonomous and semi-autonomous vehicles. What sets mobile robotics apart from other research areas is its multidisciplinary nature. For a robot to behave in a large-scale environment requires dealing with the incremental acquisition of knowledge, the estimation of positional error, the ability to recognize important or familiar objects or places and respond in real time, and the coordination of all these functionalities in concert.

Following are some improvements that could follow this thesis that are recommended for better understanding and control of mobile robotics.

1. Implement H-infinity control on the mobile robot system digitally to study the performance and the robustness of this technique.

2. To achieve this complicated task, it is better to use a different microcontroller that can handle more complicated tasks and execute software in parallel rather than sequentially.

3. Use a higher programming language than assembly such as C or C++ to execute more complicated functions that the mobile robotics system requires.

4. This kind of research touches a lot of engineering areas such as mechanical, electrical, software, and even computer science. This kind of project should be a group project requiring the cooperation of all the engineering departments to achieve a practical system that works and can grow as the requirements demand.

BIBLIOGRAPHY

[1] Gao Zhiqiang, "From linear to nonlinear control means: a practical Progression," ISA Transactions, vol. 41, no. 2, pp. 177-89, April 2002.

[2] Gao Zhiqiang, Huang Yi, and Han Jingqing, "An alternative Paradigm For Control System Design," Proc. of the 40th IEEE Conference on Decision and Control, Orlando, Florida, pp. 4578-4585, December 2001.

[3] Gao Zhiqiang and Nonnenmarcher Wilfred, "Fuzzy Logic Control of an- Industrial Indexing Motion Application," Department of Electrical and Computer Engineering, Cleveland State University, Cleveland, Ohio.

[4] Y. Hou, Z. Gao, F. Jiang and B. T. Boulter, "Active Disturbance Rejection Control for Web Tension Regulation," Department of Electrical and Computer Engineering, Cleveland State University, Cleveland, Ohio.

[5] Simon Dan, "Analyzing Control system robustness," Department of Electrical and Computer Engineering, Cleveland State University, Ohio

[6] Burl Jeffrey B., "Linear Optimal Control," Addison-Wesley, 1999.

[7] Bijoy K. Ghosh, Hing Xi, and Tarn T.J., "Control in Robotics and Automation," Academic Press, 1999.

[8] Burns William C., and Worthington Janet, "Practical Robotics," Prentice Hall, 1998.

[9] Jones Joseph L., Flynn Anita M., and Seiger Bruce A., "Mobile Robots," A K Peters, 1999.

[10] Wise Edwin, "Applied Robotics," PROMPT pub. 2000.

[11] Miller Merl K., Winkless Nelson, and Bosworth Joe, "The Personal Robot Navigator," Robot Press, 2000.

[12] Bolton W., "Mechatronics," LONGMAN, 1999.

[13] McComb Gordan, "The Robot Builder's Bonanza," McGraw-Hill, 2001.

[14] Spong Mark W., Lewis F.L., and Abdallah C.T., "Robot Control Dynamics," Motion Planning and Analysis, IEEE Press, 1996.

[15] Tsui Chia-Chi, "Robust Control System Design," Marcel Dekker,Inc., 1997.

[16] Jang J.-S.R., Sun C.T., and Mizutani E., "Neuro-Fuzzy and Soft Computing," Matlab Curriculum Series, 1997.

[17] Godjevac Jelen, "Neuro-Fuzzy Controller," LAM II, 1996.

[18] Benson, David. "Easy Step'n." Version 1.0. Square 1 Electronics. 2001.

[19] Jones, Joseph, "Mobile Robots: Inspiration to Implementation," A.K. Peters. 1990.

[20] Kozyrev, Yu. "Industrial Robots," Mir Publishers. 1985

[21] Durf, Richard, "Modern Control Systems," Addison Wesley. 1998.

[22] Chen, Chi-tsung. "Linear System Theory and Design," Oxford University Press. 1999.

[23] Valavanis, Kimon and Saridis, George. "Intelligent Robotic Systems: Theory, Design, and Applications," Kluwer Academic Publisher. 1992.

[24] Gibilisco, Stan. "Artificial Intelligence," McGraw-Hill. 1994.

[25] Gosh, Bijoy and Taran, T.J. "Control in Robotics and Automation," Academic Press. 199

[26] Lee, Mark. "Intelligent Robotics," Halsted Press. 1989.

[27] Koren, Yoram. "Robotics for Engineers," McGraw-Hill. 1985.

[28] Nadiu, Desineni. "Optimal Control Systems," Dover Publications. 2000.

[29] Stengel, Robert. "Optimal Control and Estimation," Dover Publications. 1982.

[30] Ellis, George. "Control System: Design Guide," Academic Press. 2000.

[31] Nehmzow, Ulrich. "Mobile Robotics," Springer. 2003.

[32] Burns, William and Worthington, Janet. "Practical Robotics; System, Interfacing and Applications," Prentice-Hall. 1986.

[33] Wise, Edwin. "Applied Robotics," Prompt. 1999.

[34] Dudek, Gregory and Jenkin, Michael. "Computational Principles of Mobile Robotics," Cambridge University. 2000.

[35] Martin, Fred. "Robotic Explorations," Prentice-Hall. 2001.

[36] Kilian, Christopher. Modern Control Technology. West Publishing Company. 1996.

[37] Potter, Tony and Guild, Ivon. "Robotics," Usborne Publishing Ltd. 1993.

[38] Burks, Arthur. "Robotics and Free Minds," The College of Literature, Science, and the Arts the University of Michigan. 1982.

[39] Bone, Jan. "Opportunities in Robotics Careers," VGM Career Horizons. 1993.

[40] Branwyn, Gareth. "Absolute Beginner's Guide to Building Robots," Que Publishing. 2004.

[41] Katzen, Sid. "The Quintessential PIC Microcontroller," Springer-Verlag. 2001.

[42] Kenjo, Takashi and Sugawara, Akira. "Stepping Motors and their Microprocessor Controls," Oxford Science Publications. 1994.

[43] Acarnley, Paul. "Stepping Motors: A guide to Theory and Practice," Institution of Electrical Engineers. 2002.

[44] Dorst Leo, Michiel Van Lambalgen, Fran Voorbraak, "Reasoning with Uncertainty in Robotics," Spriger. 1995.

[45] Vladimir J. Lumelsky, Michael S. Shur, and Sigurd Wagner, "Sensitive Skin," IEEE Sensors Journal, Vol. 1, NO.1, June 2001.