PID CONTROLLED PATH TRACKER ROBOT

A MAJOR PROJECT REPORT
SUBMITTED IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE OF

"BACHELOR OF TECHNOLOGY"

IN

MECHANICAL ENGINEERING (MECHATRONICS)

Submitted by

BUKKA VINAY KRISHNA	**[15261A1410]**
D NAGENDRA PRASHAD	**[15261A1411]**

Under the esteemed guidance of

Mr. G. SREENIVASULU REDDY
Assistant Professor
Department of Mechanical Engineering (Mechatronics)

DEPARTMENT OF MECHANICAL ENGINEERING (MECHATRONICS)

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

Accredited by NAAC with 'A' Grade for five years, New Delhi,
Accredited by National Board of Accreditation, New Delhi.
(Affiliated to Jawaharlal Nehru Technological University, Hyderabad)
Gandipet, Hyderabad- 500 075 (T.S)
www.mgit.ac.in
APRIL 2019

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

Accredited by NAAC with 'A' Grade for five years, New Delhi,
Accredited by National Board of Accreditation, New Delhi.
(Affiliated to Jawaharlal Nehru Technological University, Hyderabad)
Gandipet, Hyderabad- 500 075 (T.S)
www.mgit.ac.in

CERTIFICATE

This is to certify that the project report entitled

PID CONTROLLED PATH TRACKER ROBOT

Submitted by

BUKKA VINAY KRISHNA	**[15261A1410]**
D NAGENDRA PRASHAD	**[15261A1411]**

In partial fulfillment for the award of the degree of Bachelor of Technology in Mechanical Engineering (Mechatronics) is a record of bonafide work carried out by them under my guidance and supervision during the academic year 2018-2019.

The results embodied in this project report have not been submitted to any other University/Institute for the award of any Degree/ Diploma.

Internal Guide

Mr. G. SREENIVASULU REDDY
Assistant Professor
Dept. of Mech.Engg., MGIT

Head of the Department

Dr. K. SUDHAKAR REDDY
Professor and head
Dept. of Mech.Engg., MGIT

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGMENT

We wish to express our sincere gratitude to my internal guide **Mr. G. Sreenivasulu Reddy, Assistant Professor, Department of Mechanical Engineering (Mechatronics), MGIT, Hyderabad** for his valuable guidance, technical support, encouragement, and constructive thoughts throughout and for helping us in completing the Project work successfully.

We wholeheartedly thank the support of **Dr. K. Sudhakar Reddy, Head of the Department, Mechanical Engineering (Mechatronics), MGIT, Hyderabad** for his timely advice and for the provision of lab facilities.

We thank the management, **Prof. K. Jaya Sankar, Principal, MGIT, Hyderabad**, the faculty and staff of M.G.I.T for the support and cooperation during our course work and project work.

Finally, we thank all the people who have directly or indirectly helped us during the course of our project work.

BUKKA VINAY KRISHNA [15261A1410]

D NAGENDRA PRASHAD [15261A1411]

ABSTRACT

A proportional-integral-derivative controller (PID) is a control loop feedback mechanism widely used in industrial control systems and a variety of other applications requiring continuously modulated control. It can be used as a means of controlling temperature, pressure, flow, and other process variables. The purpose of a PID controller is to force feedback to match a set point, such as a thermostat that forces the heating and cooling unit to turn on or off based on a set temperature. PID controllers are best used in systems which have relatively a small mass and those which react quickly to changes in the energy added to the process. The working principle behind a PID controller is that the proportional, integral and derivative terms must be individually adjusted or "tuned". Based on the difference between these values a correction factor is calculated and applied to the input.

This project aims to design a PID controlled path tracker robot with motor control. The path tracker robot is generally made to follow across a particular line or path using IR sensors. The typical line or path followers have a jerky movement due to the sudden nature of line detection and path correction. The problem of the jerky movement can be solved by PID control. By applying the PID control algorithm, the path follower can be made to move smoothly along the line. This also allows the line follower to move faster and follow its path with much pace.

TABLE OF CONTENTS

Certificate	ii
Acknowledgment	iii
Abstract	iv
Contents	v
List of Figures	vii
List of Tables	viii

CONTENTS

S.No	Chapter Name	Page No.
1.	**INTRODUCTION**	1
2.	**LITERATURE REVIEW**	3
3.	**PID CONTROLLER**	5
	3.1 PID Controller Theory	5
	3.1.1 P-Controller	6
	3.1.2 I-Controller	7
	3.1.3 D-Controller	8
	3.2 Controller Types	9
	3.2.1 P-I Controller	9
	3.2.2 P-D Controller	10
	3.2.3 P-I-D Controller	10
	3.3 PID Tuning Methods	11
	3.3.1 Manual Tuning Method	12
	3.3.2 Ziegler-Nichols Method	13
	3.3.3 Cohen-Coon Tuning Method	18
	3.4 MATLAB Tuning Method	19
4.	**HARDWARE**	25
	4.1 IR Sensor	25
	4.2 IR Sensor Array	34
	4.3 DC Motor	38

4.4 DC Geared Motor 40
4.5 Motor Driver 43
4.6 Battery 48
4.7 Lithium-Ion Battery 48
4.8 Arduino UNO 50
5. **WORKING MODEL** 52
5.1 Prototype Development 52
5.2 Block Diagram 53
5.3 Circuit Diagram 54
5.4 Working 56
5.5 Flow Diagram 61
5.6 Applications 62
6. **RESULTS AND DISCUSSION** 63
7. **CONCLUSION AND FUTURE SCOPE** 65
REFERENCES 66
APPENDIX 67

LIST OF FIGURES

Fig.No.	Name of the Figure	Page No.
3.1	PID Controller	5
3.2	P-Controller Response	7
3.3	I-Controller Response	8
3.4	PI-Controller Response	10
3.5	PD-Controller Response	10
3.6	PID-Controller Response	11
3.7	Ziegler-Nichols Method	14
3.8	Ziegler-Nichols open loop graph	17
3.9	Cohen-Coon graph	18
3.10	Unit Step Response	21
3.11	Kp = 100	21
3.12	Kp = 100, Ki = 1, Kd = 1	22
3.13	Kp = 100, Ki = 20, Kd = 1	22
3.14	Kp = 100, Ki = 100, Kd = 1	23
3.15	Kp = 100, Ki = 200, Kd = 1	23
3.16	Kp = 100, Ki = 200, Kd = 2	24
3.17	Kp = 100, Ki = 200, Kd = 10	24
4.1	IR Sensor	25
4.2	Break Beam Sensor	27
4.3	Reflectance Sensor	27
4.4	IR Transmitter	30
4.5	IR Receiver	31
4.6	IR Working Principle	32
4.7	IR Circuit	32
4.8	QTR-8RC Reflectance Sensor Array	34
4.9	Reflectance Array Circuit	35
4.10	QTR-8RC Module Pins	37
4.11	DC Motor Working Principle	39
4.12	Rotation of the Coil	39

4.13	Micro Metal Gearmotor	41
4.14	Wheels	43
4.15	H-Bridge Circuit	44
4.16	PWM Signal	45
4.17	L293D Motor Driver	46
4.18	L293D Pin Diagram	47
4.19	Batteries	49
4.20	Arduino Pin Diagram	51
5.1	Chassis	52
5.2	Fabrication of the Prototype	52

LIST OF TABLES

Table No.	**Name of the Table**	**Page No.**
3.1	PID Parameters variation with system dynamics	11
3.2	Ziegler-Nichols Closed loop parameters	16
3.3	Ziegler-Nichols Open loop parameters	17
3.4	Cohen-Coon Tuning parameters	19
4.1	Micro-Metal Gearmotor Specifications	42
6.1	Kp Response	64
6.2	Kd Response	64
6.3	Ki Response	64

1. CHAPTER
INTRODUCTION

A Closed-loop Control System, also known as a feedback control system is a control system that uses the concept of an open loop system as its forward path but has one or more feedback loops or paths between its input and its output. The reference to “feedback”, simply means that some portion of the output is returned “back” to the input to form part of the excitation of the system. Closed-loop systems are designed to automatically achieve and maintain the desired output condition by comparing it with the actual condition. It does this by generating an error signal which is the difference between the output and the reference input. In other words, a “closed-loop system” is a fully automatic control system in which its control action being dependent on the output in some way.

One of the most widely used closed-loop control strategies is the PID control system. PID controller is a Close loop system that has a feedback control system and it compares the Process variable (feedback variable) with set point and generates an error signal and according to that, it adjusts the output of the system. This process continues until this error is equal to Zero or process variable value becomes equal to a set point. This control system is a mechanism that contains three components: proportional controller (Kp), integral controller (Ki), and derivative controller (Kd). In PID control system, we need to tune the value of Kp, Ki and Kd to reach the best setting.

In this project, A detailed study of the PID controller and its tuning methods is made. The major part of the project is to develop a PID control for a path tracker (line follower) robot. Path Tracker or Line follower robot is a robot that follows along a well- designed line or path. Generally, such basic robots have large errors while following along a line like the deviation from path, wobbling, and jerks. By using PID control to such robots errors can be minimized and made equal to zero. Path tracker robots are used in industries as Automate Guided Vehicle (AGV's). The development and testing of PID controlled path tracker robot is performed with a proper tuning method.

2. CHAPTER
LITERATURE REVIEW

Various robots have been developed for line following. The various types of such robots include a two sensor robot, a three sensor robot, and a obstacle avoidance robots. Considering a two sensor line follower robot as in [1] follows a line or path with two IR sensors on either side of the chassis. The IR sensor module consists of an IR LED and a Photodiode and some other components like comparator, LED, etc. It also consists of a microcontroller to control the robot and a motor driver to control the directions of the wheels. Four possible conditions are possible in robot movement when a two sensor module is used:

- The first condition is when both the sensors don't detect the line. Both the motors rotate forward. As a result, the robot moves forward.
- The second condition is when only the left sensor detects the line which means that the robot requires to turn in the left direction. The left motor rotates backward and the right motor rotates forward. As a result, the robot turns left.
- The third condition is when only the right sensor detects the line which means that the robot requires to turn in the right direction**.** The left motor rotates forward and the right motor rotates backward. As a result, the robot turns right**.**
- The fourth condition is when both the sensors detect the line. This means that the end has come. Both the motors stop rotating. As a result, the robot stops.

Another two sensor line follower has been made as in [3] where color sensors aiming to detect the path in a faster time than the existing line follower. A line follower robot with 3 IR sensors has been made as in [2] using an Arduino 328 Microcontroller with the aim of reducing wobbling in the robot. Similarly, a robot with 5 IR sensors has been made as in

[4] using an AVR Microcontroller. Both of these papers provide a basic idea about using a PID control for a line follower robot. The efficiency of these robots in following the line is better than those in [1], [2].

A two sensor line follower robot can be designed and controlled easily. The limitations of these robots are:

1. If it overshoots the line, it cannot retrace its path again easily.
2. It only works for paths that are smooth, straight and without much turns.
3. Not suitable for high speeds.
4. Variations in the speed of the motors are not considered.
5. The Efficiency of the robot following a line is very less.

3. CHAPTER
PID CONTROLLER

3.1 PID CONTROLLER THEORY

PID is an acronym for “proportional, integral and derivative.” A PID controller is a controller that includes elements with the above three functions. In the literature survey on PID controllers, acronyms are also used at the element level: the proportional element is referred to as the “P element,” the integral element as the “I element,” and the derivative element as the “D element.” The PID controller was first placed in the market in the year 1939 and has remained the most widely used controller in process control until today. An investigation performed in Japan indicated that more than 90% of the controllers used in the process industries are PID controllers and its advanced versions.

The distinguishing feature of the PID controller is the ability to use the three control terms of proportional, integral and derivative influence on the controller output to apply accurate and optimal control. The block diagram shows the principles of how these terms are generated and applied. It shows a PID controller, which continuously calculates an error value as the difference between the desired set point and a measured process variable, and applies a correction based on proportional, integral, and derivative terms. The controller attempts to minimize the error over time by adjustment of a control variable, such as the opening of a control valve, to a new value determined by a weighted sum of the control terms.

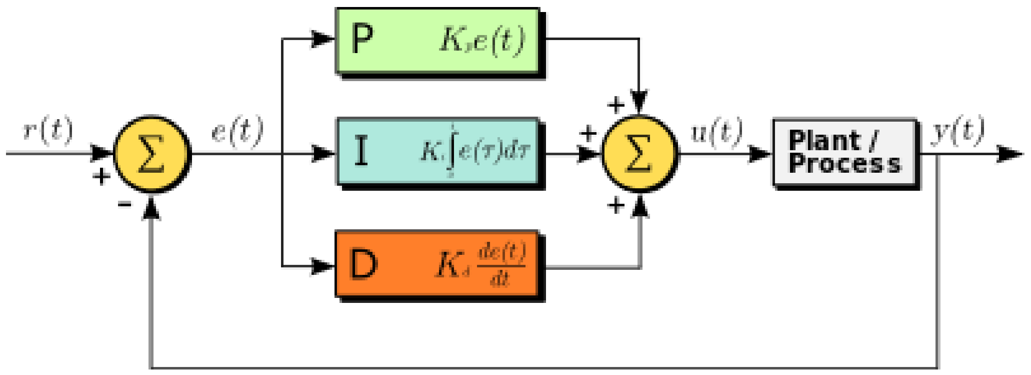


Fig:3.1 PID Controller

The mathematical form for a PID controller is given by,

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de}{dt}$$

The variable (e) represents the tracking error, the difference between the desired output (r) and the actual output (y). This error signal (e) is fed to the PID controller, and the controller computes both the derivative and the integral of this error signal with respect to time. The control signal (u) to the plant is equal to the proportional gain (K_p) times the magnitude of the error plus the integral gain (K_i) times the integral of the error plus the derivative gain (K_d) times the derivative of the error. This control signal (u) is fed to the plant and the new output (y) is obtained. The new output (y) is then fed back and compared to the reference to find the new error signal (e). The controller takes this new error signal and computes an update of the control input. This process continues while the controller is in effect.

The transfer function of a PID controller is found by taking the Laplace transform of the above Equation

$$K_p + \frac{K_I}{s} + K_d = \frac{K_d s^2 + K_p s + K_i}{s}$$

3.1.1 P-Controller

Proportional or P- controller gives an output which is proportional to current error e (t). It compares the set point with actual value. The resulting error is multiplied with a proportional constant to get the output. If the error value is zero, then this controller output is zero. This controller requires a biasing or manual reset when used alone. This is because it never reaches a steady state condition. It provides stable operation but always maintains the steady state error.

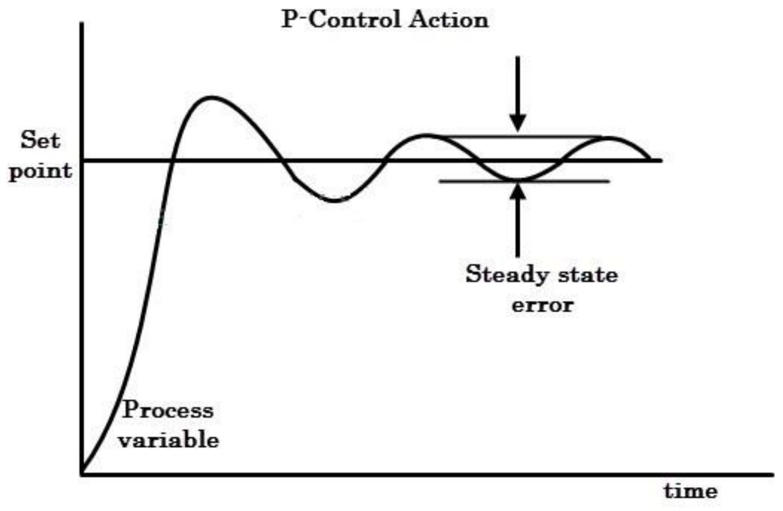


Fig: 3.2 P-Controller Response

The above figure 3.2 shows that the proportional controller has reduced both the rise time and the steady-state error, increased the overshoot, and decreased the settling time by a small amount.

Advantages of Proportional Controller: Now let us discuss some advantages of the proportional controller.

1. The Proportional controller helps in reducing the steady-state error, thus makes the system more stable.
2. The Slow response of the overdamped system can be made faster with the help of these controllers.

Disadvantages of Proportional Controller: Now disadvantages of these controllers are written as follows:

1. Due to the presence of these controllers, we get some offsets in the system.
2. Proportional controllers also increase the maximum overshoot of the system.

3.1.2 I-Controller

An Integral controller is mainly used to reduce the steady-state error of the system. The integral component integrates the error term over a period of time until the error becomes zero. This results that even a small error value will produce a high integral response. At the zero error condition, it holds the output to the final control device at its last value in order to maintain zero steady state error, but in case of P-controller, the output is zero when the error is zero. If the error is negative, the integral response or output will be decreased. The speed of response is slow (means respond slowly) when I-controller is used alone, but improves the steady-state response. By decreasing the integral gain Ki, the speed of the response is increased. So the major advantage of this integral control action is that the steady-state error due to step input reduces to zero. But simultaneously, the system response is generally slow, oscillatory and unless properly designed, sometimes even unstable.

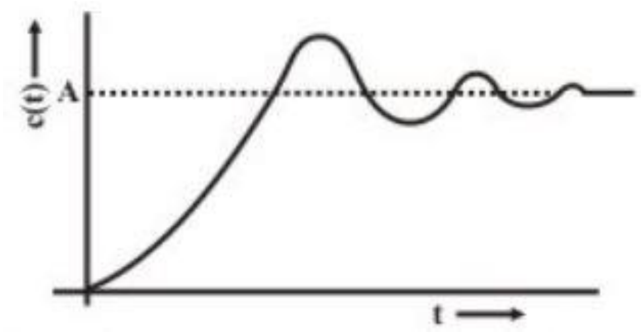


Fig: 3.3 I-Controller Response

Advantages of Integral Controller

Due to their unique ability, they can return the controlled variable back to the exact set point following a disturbance that’s why these are known as reset controllers.

Disadvantages of Integral Controller

It tends to make the system unstable because it responds slowly towards the produced error.

3.1.3 D-Controller

A derivative controller sees how fast process variable changes per unit of time and produce the output proportional to the rate of change. The derivative output is equal to the rate of change of error multiplied by a derivative constant. The D-controller is used when the processor variable starts to change at a high rate of speed. In such case, D-controller moves the final control device (such as control valves or motor) in such direction as to counteract the rapid change of a process variable. It is to be noted that D-controller alone cannot be used for any control applications. The derivative action increases the speed of the response because it gives a kick start for the output, thus anticipates the future behavior of the error. The more rapidly D-controller responds to the changes in the process variable implies that the derivative term is large (which is achieved by increasing the derivative constant or time Td).

Advantages of Derivative Controller

The major advantage of a derivative controller is that it improves the transient response of the system.

3.2 CONTROLLER TYPES

3.2.1 P-I Controller

P-I controller is mainly used to eliminate the steady state error resulting from P controller. However, in terms of the speed of the response and overall stability of the system, it has a negative impact. This controller is mostly used in areas where the speed of the system is not an issue. Since the P-I controller has no ability to predict the future errors of the system it cannot decrease the rise time and eliminate the oscillations. If applied, any amount of I guarantees set point overshoot. It is widely used for process industries for controlling variables like level, flow, pressure, etc., those do not have large time constants.

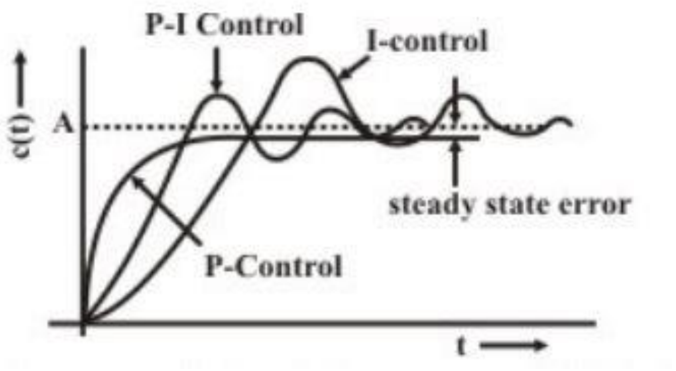


Fig: 3.4 PI-Controller Response

3.2.2 P-D Controller

The aim of using the P-D controller is to increase the stability of the system by improving control since it has an ability to predict the future error of the system response. In order to avoid the effects of the sudden change in the value of the error signal, the derivative is taken from the output response of the system variable instead of the error signal. Therefore, D mode is designed to be proportional to the change of the output variable to prevent the sudden changes occurring in the control output resulting from sudden changes in the error signal. In addition, D directly amplifies process noise therefore D-only control is not used.

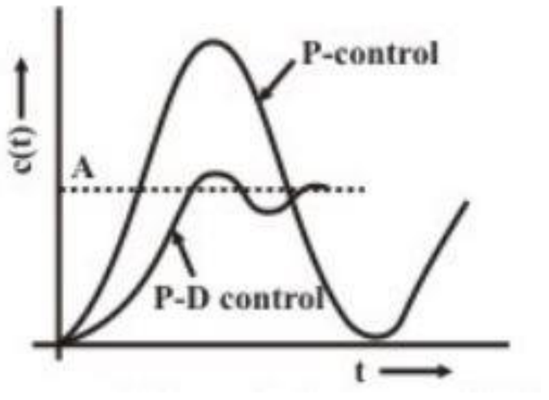


Fig: 3.5 PD-Controller Response

3.2.3 P-I-D Controller

P-I-D controller has the optimum control dynamics including zero steady-state error, fast response (short rise time), no oscillations and higher stability. The necessity of using a derivative gain component in addition to the PI controller is to eliminate the overshoot and the oscillations occurring in the output response of the system. One of the main advantages of the P-I-D controller is that it can be used with higher order processes including more than single energy storage. The PID controller is used in inertial systems with a relatively low noise level of the measuring channel. The advantage of PID is fast warm-up time, accurate set point and fast reaction to disturbances.

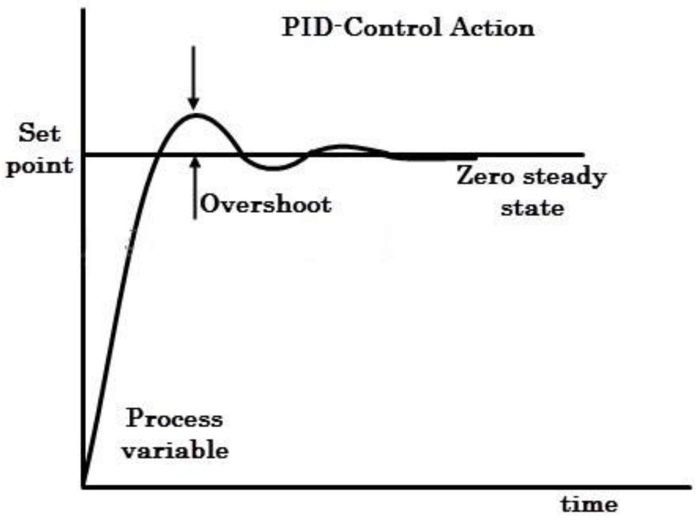


Fig: 3.6 PID-Controller Response

The PID parameters affecting the system dynamics is given in the table below

Response	Rise Time	Overshoot	Settling Time	S-S Error
K_p	Decrease	Increase	Small change	Decrease
K_i	Decrease	Decrease	Increase	Eliminate
K_d	Indefinite	Decrease	Decrease	No effect

Table: 3.1 PID parameters variation with system dynamics

3.3 PID TUNING METHODS

Tuning a control loop is arranging the control parameters to their optimum values in order to obtain a desired control response. At this point, stability is the main necessity, but beyond that, different systems lead to different behaviors and requirements and these might not be compatible with each other. In principle, P-I-D tuning seems completely easy, consisting of only 3 parameters, however, in practice; it is a difficult problem because the complex criteria at the P-I-D limit should be satisfied. P-I-D tuning is mostly a heuristic concept but the existence of many objectives to be met such as short transient, high stability makes this process harder. For example, sometimes, systems might have nonlinearity problem which means that while the parameters work properly for full load conditions, they might not work as effectively for no load conditions. Also, if the P-I-D parameters are chosen wrong, control process input might be unstable, with or without oscillation; output diverges until it reaches saturation or mechanical breakage.

For a system to operate properly, the output should be stable, and the process should not oscillate in any condition of a set point or disturbance. However, for some cases bounded oscillation condition as marginal stability can be accepted.

As an optimum behavior, a process should satisfy the regulation and command breaking requirements. These two properties define how accurately a controlled variable reaches the desired values. The most important characteristics for command breaking are rise time and settling time. For some systems where overshoot is not acceptable, to achieve the optimum behavior requires eliminating the overshoot completely and minimizing the dissipated power in order to reach a new set point.

In today's control engineering world, P-I-D is used over 95% of the control loops. Actually, if there is a control, there is P-I-D, in analog or digital forms. In order to achieve optimum solutions Kp, Ki and Kd gains are arranged according to the system characteristics.

3.3.1 Manual Tuning Method

Manual tuning is achieved by arranging the parameters according to the system response. Until the desired system response is obtained Ki, Kp and Kd are changed by observing system behavior. It is a simple method of PID controller tuning. While the system or controller is working, we can tune the controller. In this method, first we have to set Ki and Kd values to zero and increase proportional term (Kp) until the system reaches to oscillating behavior. Once it is oscillating, adjust Ki (Integral term) so that oscillations stops and finally adjust D to get a faster response.

Example (for no system oscillation): First lower the derivative and integral value to 0 and raise the proportional value 100. Then increase the integral value to 100 and slowly lower the integral value and observe the system's response. Since the system will be maintained around the set point, change the set point and verify if the system corrects in an acceptable amount of time. If not acceptable or for a quick response, continue lowering the integral value. If the system begins to oscillate again, record the integral value and raise the value to 100. After raising the integral value to 100, return to the proportional value and raise this value until oscillation ceases. Finally, lower the proportional value back to 100 and then lower the integral value slowly to a value that is 10% to 20% higher than the recorded value when oscillation started (recorded value times 1.1 or 1.2). Although the manual tuning method seems simple it requires a lot of time and experience.

3.3.2 Ziegler-Nichols Method

More than six decades ago, P-I controllers were more widely used than P-I-D controllers. Despite the fact that the P-I-D controller is faster and has no oscillation, it tends to be unstable in the condition of even small changes in the input set point or any disturbances to the process than P-I controllers. Ziegler-Nichols Method is one of the most effective methods that increase the usage of P-I-D controllers.

The Ziegler-Nichols rule is a heuristic PID tuning rule that attempts to produce good values for the three PID gain parameters:

1. *Kp* - the controller path gain
2. *Ti* - the controller's integrator time constant
3. *Td* - the controller's derivative time constant

given two measured feedback loop parameters derived from measurements:

1. the period *Tu* of the oscillation frequency at the stability limit
2. the gain margin Ku for loop stability.

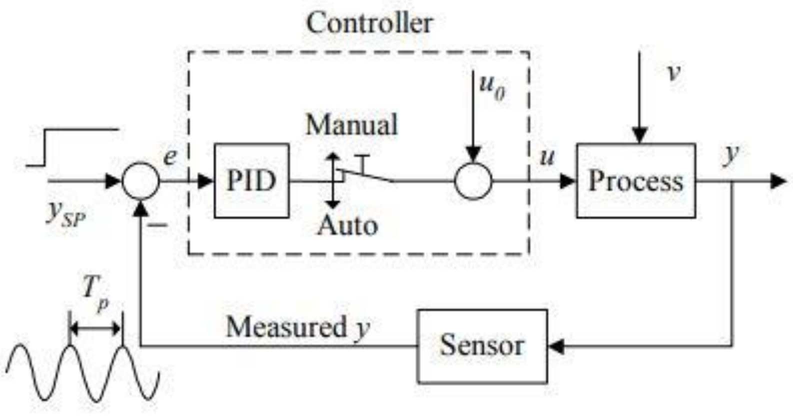


Fig: 3.7 Ziegler-Nichols Method

Ziegler-Nichols closed loop method

The main objective of the Ziegler-Nichols closed-loop method is to find the value of the proportional-only gain that causes the control loop to oscillate indefinitely at a constant value. This gain, which causes steady-state oscillations, is called the ultimate proportional gain (K_u). Another important value associated with this proportional-only control tuning method is the ultimate period (P_u). The ultimate period is the time required to complete one full oscillation once the response begins to oscillate at a constant amplitude. These two parameters, Ku and Pu, are used to find the loop-tuning constants of the controller (P, PI, or PID). To find the values of these parameters and to calculate the tuning constants, you must do the following:

1. Bring the process to (or as close to as possible) the specified operating point of the control system to ensure that the controller during the tuning is “feeling” representative process dynamic and to minimize the chance that variables during the tuning reach limits. You can bring the process to the operating point by manually adjusting the control variable, with the controller in manual mode, until the process variable is approximately equal to the set point.
2. Turn the PID controller into a P controller with gain Kp = 0 (set Ti = ∞ and Td = 0). Close the control loop by setting the controller in automatic mode.
3. Increase Kp until there are sustained oscillations in the signals in the control system, e.g. in the process measurement, after excitation of the system. (The sustained oscillations corresponds to the system being on the stability limit.) This Kp value is denoted the ultimate (or critical) gain, K_{pu}. The excitation can be a step in the set point. This step must be small, for example, 5% of the maximum set point range, so that the process is not driven too far away from the operating point where the dynamic properties of the process may be different. On the other hand, the step must not be too small, or it may be difficult to observe the oscillations due to the inevitable measurement noise. It is important that K_{pu} is found without the actuator being driven into any saturation limit (maximum or minimum value) during the oscillations. If such limits are reached, you will find that there will be sustained oscillations for any (large) value of Kp, e.g. 1000000, and the resulting Kp-value (as calculated from the Ziegler-Nichols’ formulas) is useless (the control system will probably be unstable). One way to say this is that Kpu must be the smallest Kp value that drives the control loop into sustained oscillations.

4. Measure the ultimate (or critical) period Tu of the sustained oscillations.

5. Calculate the controller parameter values according to Table, and use these parameter values in the controller.

	K_p	T_i	T_d
P controller	$0.5K_{Pu}$	∞	0
PI controller	$0.45K_{Pu}$	$\frac{T_u}{1.2}$	0
PID controller	$0.6K_{Pu}$	$\frac{T_u}{2}$	$\frac{T_u}{8}$

Table: 3.2 Ziegler-Nichols closed loop parameters

Ziegler – Nichols open-loop method

This method remains a popular technique for tuning controllers that use proportional, integral, and derivative actions. The Ziegler-Nichols open-loop method is also referred to as a process reaction method because it tests the open-loop reaction of the process to a change in the control variable output. This basic test requires that the response of the system be recorded, preferably by a plotter or computer. Once certain process response values are found, they can be plugged into the Ziegler-Nichols equation with specific multiplier constants for the gains of a controller with either P, PI, or PID actions.

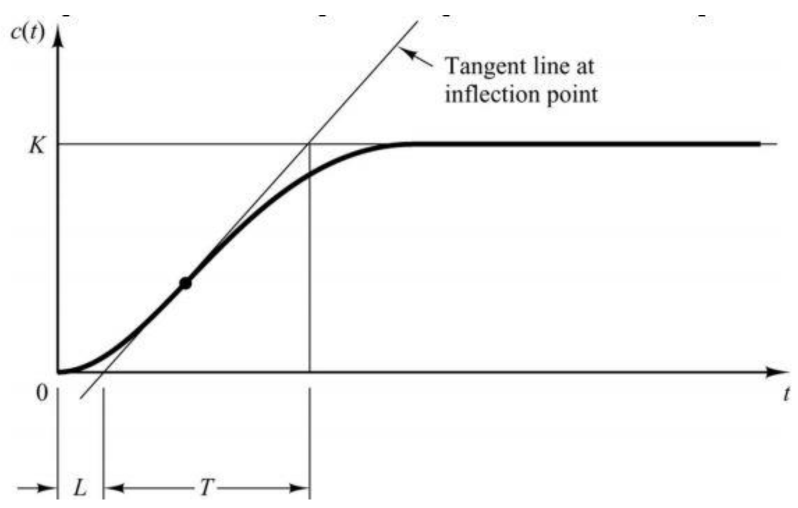


Fig: 3.8 Ziegler-Nichols open loop graph

The S-shaped reaction curve can be characterized by two constants, delay time L and time constant T, which are determined by drawing a tangent line at the inflection point of the curve and finding the intersections of the tangent line with the time axis and the steady-state level line. Using the table below the tuning parameters are calculated.

	K_p	T_i	T_d
P controller	$\frac{T}{L}$	∞	0
PI controller	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID controller	$1.2\frac{T}{L}$	$2L$	$0.5L$

Table: 3.3 Ziegler-Nichols open loop parameters

Advantages:

- It is an easy experiment; only need to change the P controller.
- Includes the dynamics of the whole process, which gives a more accurate picture of how the system is behaving.

Disadvantages:

- The Experiment can be time-consuming.

- It can venture into unstable regions while testing the P controller, which could cause the system to become out of control.
- For some cases, it might result in aggressive gain and overshoot.

3.3.3 Cohen-Coon Tuning Method:

This tuning method has been discovered almost after a decade after the Ziegler-Nichols method. Cohen-Coon tuning requires three parameters which are obtained from the reaction curve as in the Figure.

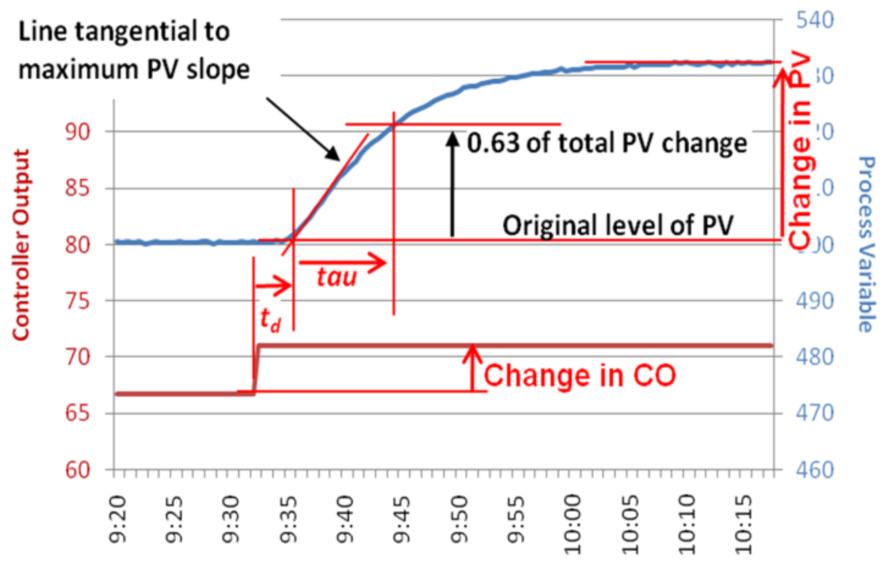


Fig: 3.9 Cohen-Coon graph

The controller is manually placed and after the process is settled out a few percent of the change is made in the controller output (CO) and waited for the process variable (PV) to settle out at a new value. As observed from the graph, process gain (gp) is calculated as follow:

$$gp = \frac{\Delta \mathrm{pv}}{\Delta \mathrm{co}} (in\ \%)$$

The maximum slope at the inflection point on the PV response curve is found and drawn a tangential line. t_d (dead time) is measured as taking the time difference between the change in CO and the intersection of the tangential line and the original PV level. As a final parameter τ (time constant) as the time difference between intersection at the end of the dead time and the PV reaching 63% of its total change. After converting the time variables into the same units and applying a couple of tests to find a similar result, these three variables are used to define new control parameters using the table

	Controller Gain	Integral Time	Derivative Time
P Controller:	$K_C = \frac{1.03}{g_p}\left(\frac{\tau}{t_d}+0.34\right)$		
PI Controller:	$K_C = \frac{0.9}{g_p}\left(\frac{\tau}{t_d}+0.092\right)$	$T_I = 3.33t_d \frac{\tau+0.092t_d}{\tau+2.22t_d}$	
PD Controller:	$K_C = \frac{1.24}{g_p}\left(\frac{\tau}{t_d}+0.129\right)$		$T_D = 0.27t_d \frac{\tau-0.324t_d}{\tau+0.129t_d}$
PID Controller: (Noninteracting)	$K_C = \frac{1.35}{g_p}\left(\frac{\tau}{t_d}+0.185\right)$	$T_I = 2.5t_d \frac{\tau+0.185t_d}{\tau+0.611t_d}$	$T_D = 0.37t_d \frac{\tau}{\tau+0.185t_d}$

Table: 3.4 Cohen-Coon tuning parameters

3.4 MATLAB Tuning Method

The P, I, D parameters can be also tuned using the software. One such software is MATLAB. With the help of MATLAB, we can tune the gains by using the transfer function of any model. We can also create a Simulink model to get the values.

With the help of transfer function of a DC Motor we will tune the gains and find the best possible values for a given step response. The transfer function for a DC Motor is given by,

$$P(s) = \frac{\dot{\theta}(s)}{V(s)} = \frac{K}{(Js+b)(Ls+R)+K^2} \quad \left[\frac{rad/sec}{V}\right]$$

Where j = moment of inertia of motor, b = motor viscous friction constant, V = voltage source, $\dot{\theta}(s)$ = rotational speed of the shaft, R = resistance, L = inductance, K = constant, s = Laplace variable.

The transfer function for a PID controller is given by,

$$C(s) = K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s}$$

With the help of the following MATLAB commands we can tune the gains,

```
J = 0.01;

b = 0.1;

K = 0.01;

R = 1;

L = 0.5;

s = tf('s');

P = K/((J*s+b)*(L*s+R)+K^2); % transfer function of motor

Kp = 100;

Ki = 200;

Kd = 1;

C = pid(Kp,Ki,Kd);

sys_cl = feedback(C*P,1);

t = 0:0.01:4;

step (sys_cl,t);

title ('Step Response with PID control')
```

The following graphs show the tuning of dc motor with help of the transfer function,

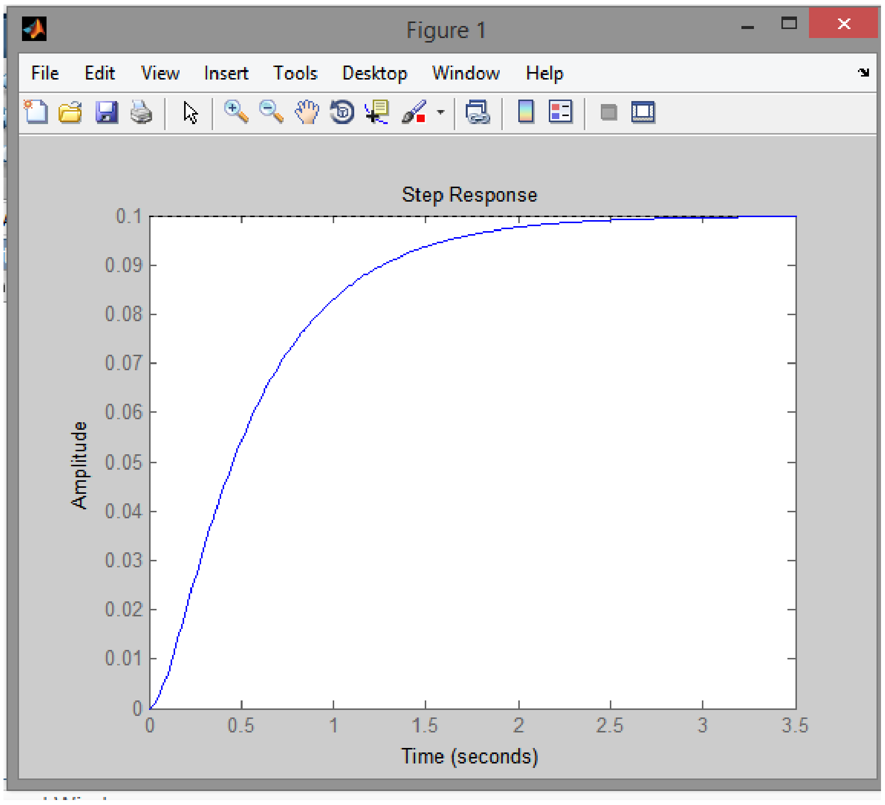


Fig: 3.10 Unit step response

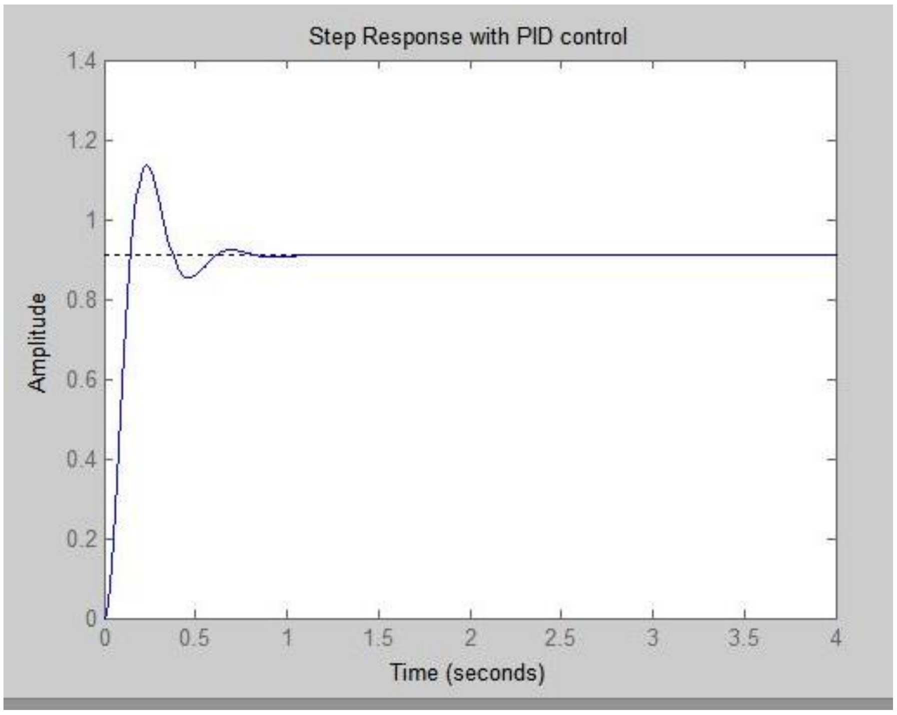


Fig: 3.11 Kp = 100

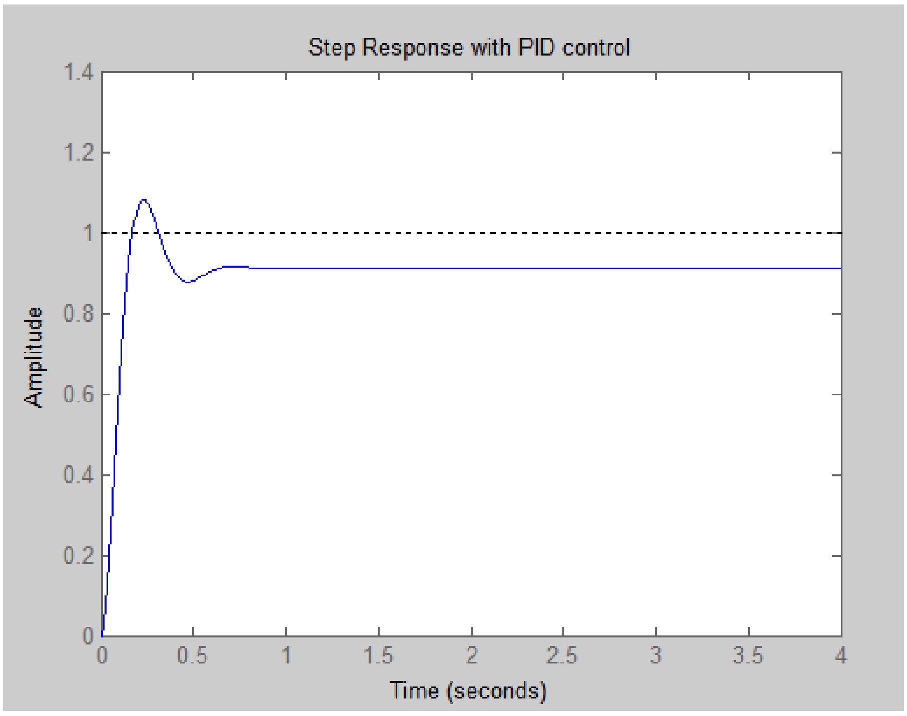


Fig: 3.12 Kp = 100, Ki = 1, Kd = 1

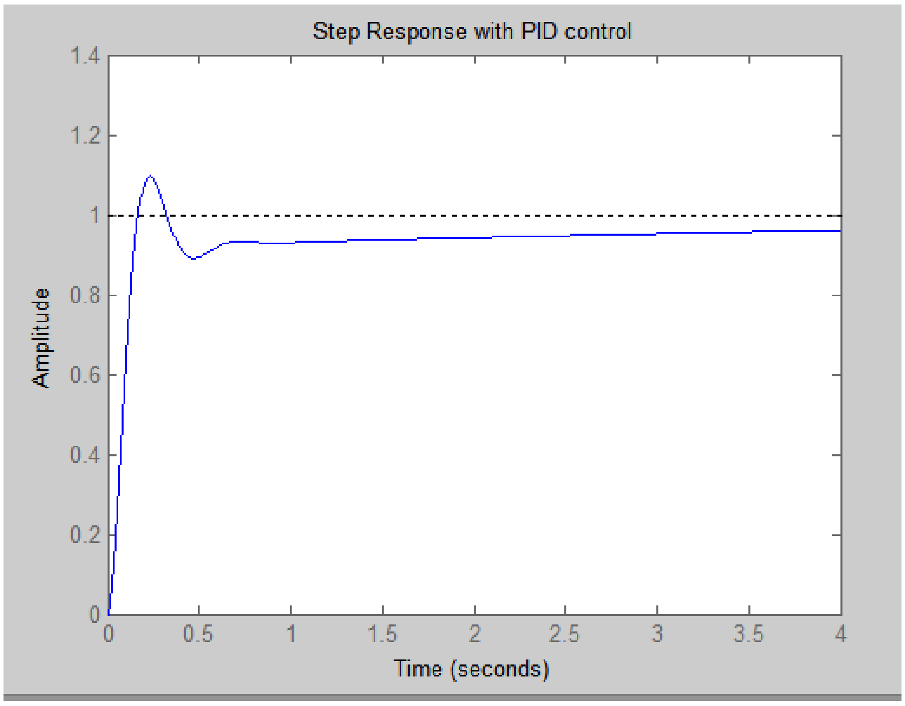


Fig: 3.13 Kp = 100, Ki = 20, Kd = 1

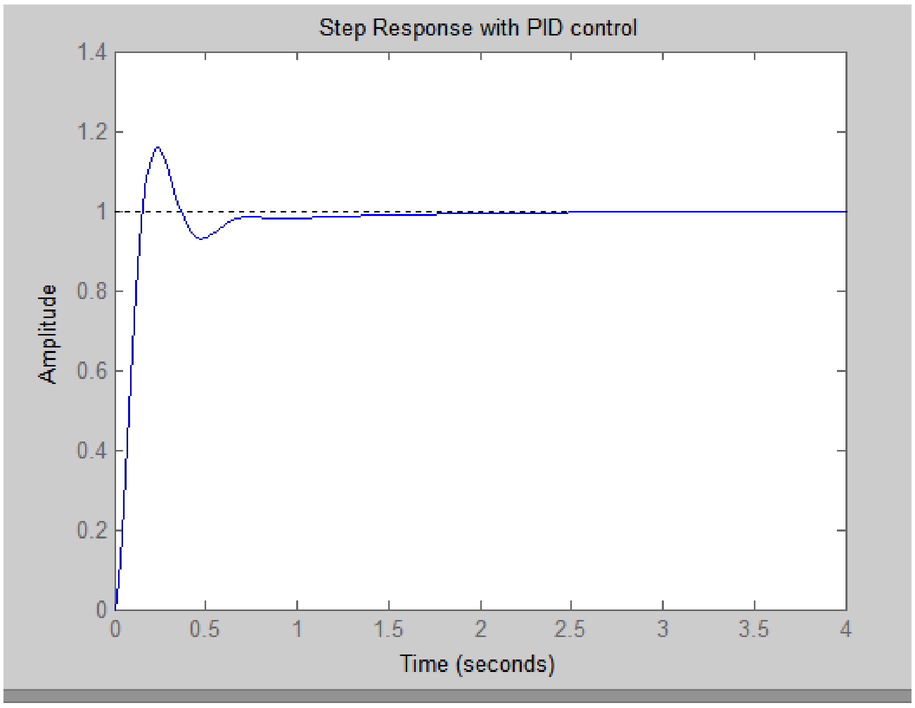


Fig: 3.14 Kp = 100, Ki = 100, Kd = 1

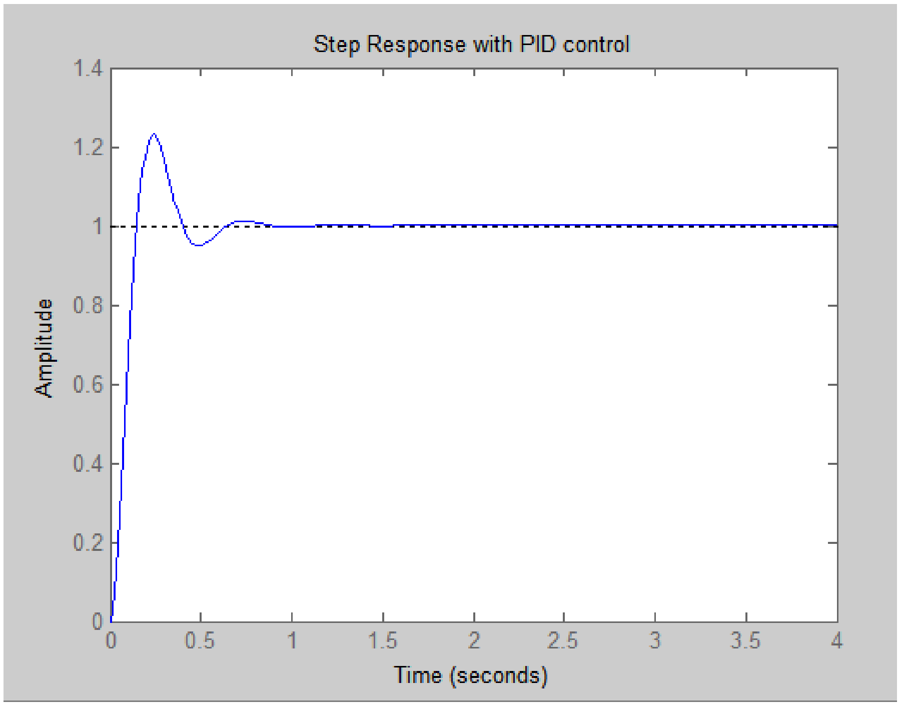


Fig: 3.15   Kp = 100, Ki = 200, Kd = 1

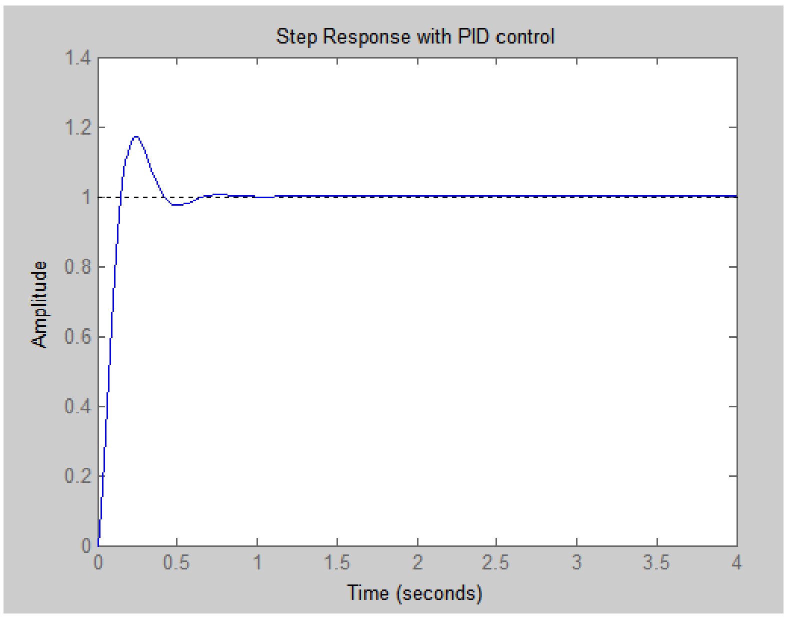


Fig: 3.16 Kp = 100, Ki = 200, Kd = 2

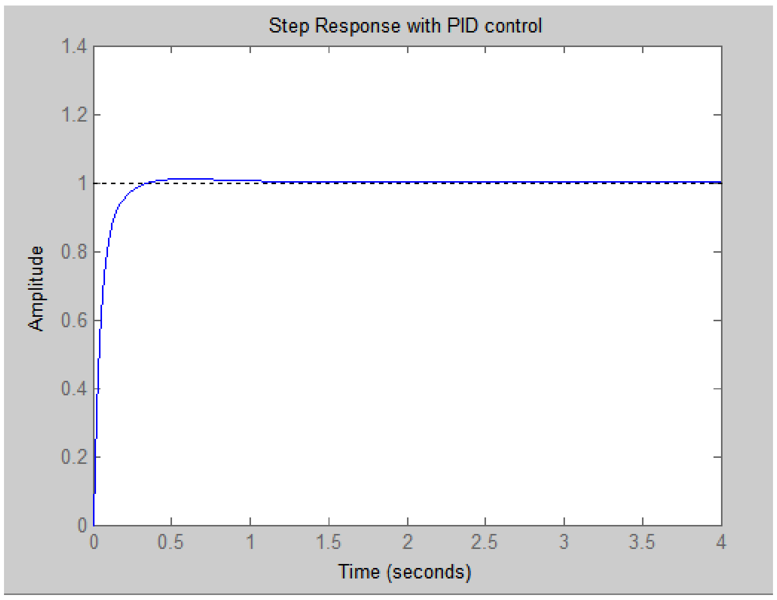


Fig: 3.17 Kp = 100, Ki = 200, Kd = 10

From the above Fig 3.10 to Fig 3.17 following has been observed:

1. After the addition of proportional constant the error is corrected but the steady state error and overshoot are high.
2. With the addition of integral constant the steady state error is reduced to almost zero but overshoot has been increased.
3. The derivative constant helped in decreasing the overshoot and system is stable.

4. CHAPTER

HARDWARE

4.1 IR SENSOR

Infrared waves are not visible to the human eye. In the electromagnetic spectrum, infrared radiation can be found between the visible and microwave regions. The infrared waves typically have wavelengths between 0.75 and 1000µm. The infrared spectrum can be split into near IR, mid IR and far IR. The wavelength region from 0.75 to 3µm is known as the near infrared region. The region between 3 and 6µm is known as the mid-infrared region, and infrared radiation which has a wavelength greater higher than 6µm is known as far infrared.

An infrared sensor is an electronic device that emits in order to sense some aspects of the surroundings. It does this by either emitting or detecting infrared radiation. Infrared sensors are also capable of measuring the heat being emitted by an object and detecting motion. An infrared sensor can be thought of as a camera that briefly remembers how an area's infrared radiation appears.

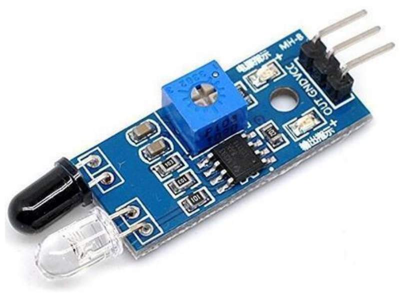


Fig: 4.1 IR Sensor

Types of IR sensors

IR sensors can be classified into two types based on the presence of IR source:

I. Active Infrared Sensor

II. Passive Infrared Sensor

Active Infrared Sensor

Active infrared sensors employ both infrared source and infrared detectors. They operate by transmitting energy from either a light emitting diode (LED) or a laser diode. An LED is used for a non-imaging active IR detector, and a laser diode is used for an imaging active IR detector. In these types of IR sensors, the LED or laser diode illuminates the target, and the reflected energy is focused onto a detector. Photoelectric cells, Photodiode or phototransistors are generally used as detectors. The measured data is then processed using various signal-processing algorithms to extract the desired information. Active IR detectors provide count, presence, speed, and occupancy data in both night and day operations. The laser diode type can also be used for target classification because it provides target profile and shape data. The types of active IR sensors are as following:

A) Break Beam Sensors -These types of Active IR sensors have emitter and receiver placed in such a way that the IR emitted by the emitter falls directly into the receiver. During the operation, the IR beam is emitted continuously towards the receiver. The flow of IR can be interrupted by placing an object between the emitter and receiver. If the IR is transmitted but altered then the receiver generates output based on the change in radiation. Similarly, if the radiation is completely blocked the receiver can detect it and provide the desired output. For example: let's

Consider a Break beam sensor and a conveyer belt as shown in the figure. When an opaque object interrupts the IR flow the receiver doesn't receive any signal thus the conveyer belt stops.

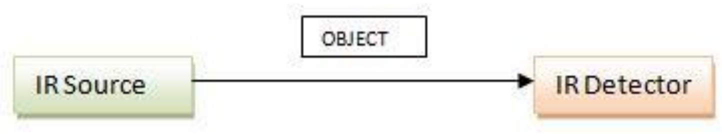


Fig: 4.2 Break Beam Sensor

B) Reflectance Sensors

This type of sensors house both an IR source and an IR detector in a single housing in such a way that light from emitter LED bounces off an external object and is reflected into a detector. The Amount of light reflected into the detector depends upon the reflectivity of the surface. This principle is used in intrusion detection, object detection (measure the presence of an object in the sensor's FOV), barcode, and surface feature detection (detecting features painted, taped, or otherwise marked onto the floor), wall tracking (detecting distance from the wall), etc. It can also be used to scan a defined area; the transmitter emits a beam of light into the scan zone, the reflected light is used to detect a change in the reflected light thereby scanning the desired zone.

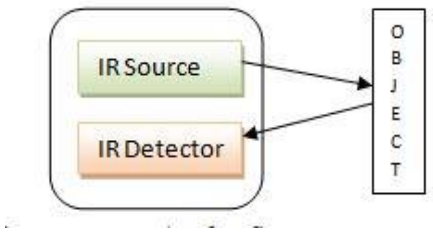


Fig: 4.3 Reflectance Sensor

Passive Infrared Sensor

Passive infrared sensors are basically Infrared detectors. Passive infrared sensors do not use any infrared source and detect energy emitted by obstacles in the field of view. It does not emit any energy of its own for the purposes of detection. Passive infrared systems can detect presence, occupancy, and count. Types of Passive Infrared sensors are:

A) Thermal Infrared sensor

Thermal type sensors have no wavelength dependence. They use the infrared energy as heat and their photosensitivity is independent of wavelength. Thermal detectors don’t require cooling but have disadvantages that response time is slow & detection time is low. Types of Thermal IR detectors are:

B) Thermocouple-Thermopile

A detector that converts temperature into an electrical signal is commonly known as a thermocouple. The junction of dissimilar metals generates a voltage potential, which is directly proportional to the temperature. This junction can be made into multiple junctions to improve sensitivity. Such a configuration is called a thermopile. The active or ‘Hot’ junctions are blackened to efficiently absorb radiation. The reference or ‘Cold’ junctions are maintained at the ambient temperature of the detector. The absorption of radiation by the blackened area causes a rise in temperature in the ‘hot’ junctions as compared to the ‘cold’ junctions of the thermopile. This difference in temperature across the thermocouple junction causes the detector to generate a positive voltage. If the active or ‘hot’ junction were to cool to a temperature less than the reference or ‘cold’ junction the voltage output would be negative. These detectors have a relatively slow response time, but offer the advantages of DC stability, requiring no bias, and responding to all wavelengths.

C) Bolometer

A bolometer is a simple thermal or total power detector. A bolometer changes resistance when incident infrared radiation interacts with the detector. This thermally sensitive semiconductor is made of a sintered metal oxide material. It has a high-temperature coefficient of resistance. It essentially consists of two main elements: a sensitive thermometer and an absorptive element and a heat sink. An Absorber is connected by a weak thermal link to a heat sink (at temperature T0). Incoming energy increases the temperature of the absorptive element above that of a heat sink and the rise in temperature is measured by a thermometer.

D) Pyroelectric Detector

Pyroelectric detectors use PZT having a pyroelectric effect, a high resistor and a low noise FET, hermetically sealed in a package. Pyroelectric materials are crystals, such as lithium tantalate, which exhibit spontaneous polarization, or a concentrated electric charge that is temperature dependent. PZT is spontaneously polarized in a dark state. As infrared radiation strikes the detector surface, the change in temperature causes a current to flow. This results in the change of polarization state which is reflected in terms of voltage change at the output. This detector exhibits good sensitivity and good response to a wide range of wavelengths and does not require cooling of the detector. It is the most commonly used detector for gas monitors.

E) Quantum Infrared Sensor

Quantum Infrared Sensor provides higher detection performance and faster response speed. Their photosensitivity is dependent on wavelength. Quantum detectors have to be cooled in order to obtain accurate measurements.

There are five basic elements used in a typical infrared detection system: an infrared source, a transmission medium, an optical component, infrared detectors or receivers, and signal processing. Infrared lasers and Infrared LED’s of a specific wavelength can be used as infrared sources. The three main types of media used for infrared transmission are a vacuum, atmosphere, and optical fibers. Optical components are used to focus the infrared radiation or to limit the spectral response.

IR Transmitter

Infrared Transmitter is a light emitting diode (LED) which emits infrared radiations. Hence, they are called IR LED's. Even though an IR LED looks like a normal LED, the radiation emitted by it is invisible to the human eye. There are different types of infrared transmitters depending on their wavelengths, output power and response time. A simple infrared transmitter can be constructed using an infrared LED, a current limiting resistor and a power supply. When operated at a supply of 5V, the IR transmitter consumes about 3 to 5 mA of current. Infrared transmitters can be modulated to produce a particular frequency of infrared light. IR transmitters can be found in several applications. Some applications require infrared heat and the best infrared source is an infrared transmitter. When infrared emitters are used with Quartz, solar cells can be made.

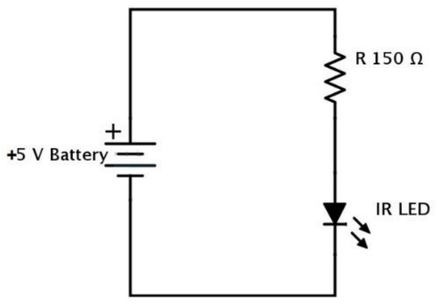


Fig: 4.4 IR Transmitter

IR Receiver

Infrared receivers are also called as infrared sensors as they detect the radiation from an IR transmitter. IR receivers come in the form of photodiodes and phototransistors. Infrared Photodiodes are different from normal photodiodes as they detect only infrared radiation. Different types of IR receivers exist based on the wavelength, voltage, package, etc. When used in an infrared transmitter-receiver combination, the wavelength of the receiver should match with that of the transmitter. It consists of an IR phototransistor, a diode, a MOSFET, a potentiometer and an LED. When the phototransistor receives any infrared radiation, current flows through it and MOSFET turns on. This, in turn, lights up the LED which acts as a load. The potentiometer is used to control the sensitivity of the phototransistor.

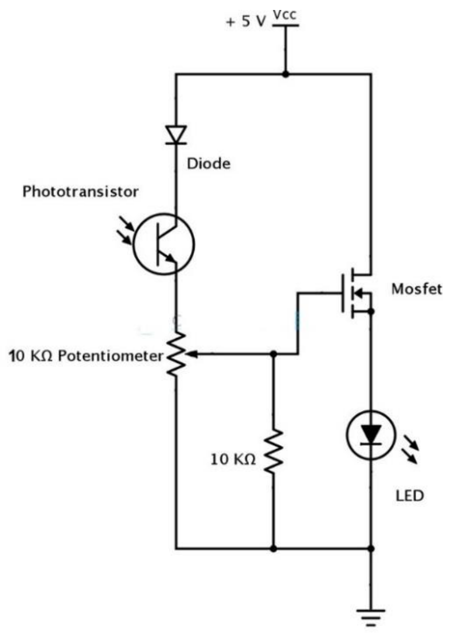


Fig: 4.5 IR Receiver

Working Principle

The principle of an IR sensor working as an Object Detection Sensor can be explained using the following figure 4.6. An IR sensor consists of an IR LED and an IR Photodiode; together they are called Photo – Coupler or Opto – Coupler. When the IR transmitter emits radiation, it reaches the object and some of the radiation reflects back to the IR receiver. Based on the intensity of the reception by the IR receiver, the output of the sensor is defined.

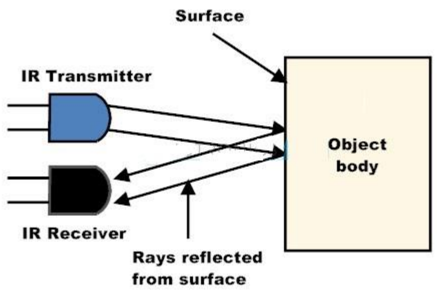


Fig: 4.6 IR Working Principle

IR Sensor Circuit

It consists of an IR LED, a photodiode, a potentiometer, an IC Operational amplifier, and an LED. IR LED emits infrared light. The Photodiode detects the infrared light. An IC Op – Amp is used as a voltage comparator. The potentiometer is used to calibrate the output of the sensor according to the requirement.

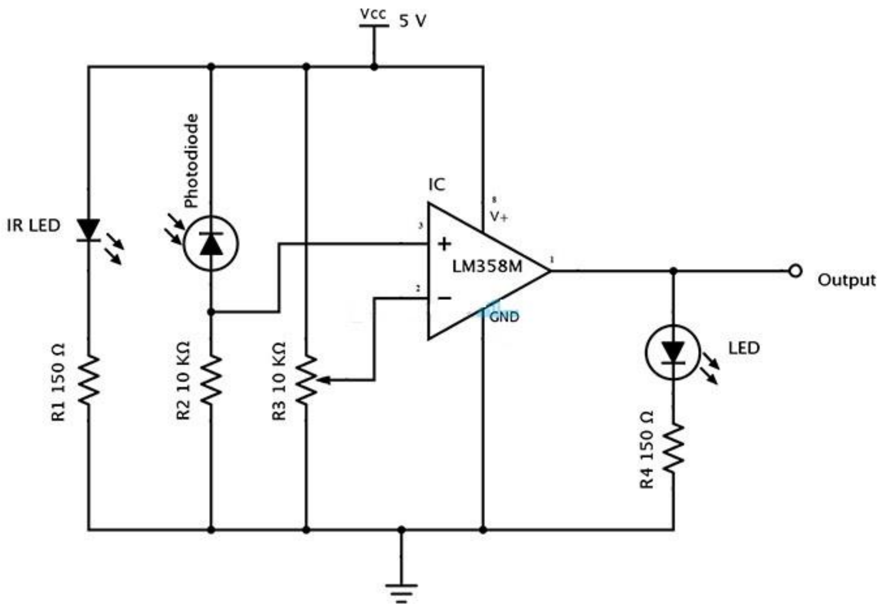


Fig: 4.7 IR Circuit

When the light emitted by the IR LED is incident on the photodiode after hitting an object, the resistance of the photodiode falls down from a huge value. One of the inputs of the op – amp is at a threshold value set by the potentiometer. The other input to the op-amp is from the photodiode's series resistor. When the incident radiation is more on the photodiode, the voltage drop across the series resistor will be high. In the IC, both the threshold voltage and the voltage across the series resistor are compared. If the voltage across the resistor series to photodiode is greater than that of the threshold voltage, the output of the IC Op – Amp is high. As the output of the IC is connected to an LED, it lightens up. The threshold voltage can be adjusted by adjusting the potentiometer depending on the environmental conditions.

The positioning of the IR LED and the IR Receiver is an important factor. When the IR LED is held directly in front of the IR receiver, this setup is called Direct Incidence. In this case, almost the entire radiation from the IR LED will fall on the IR receiver. Hence there is a line of sight communication between the infrared transmitter and the receiver. If an object falls in this line, it obstructs the radiation from reaching the receiver either by reflecting the radiation or absorbing the radiation.

Distinguishing Between Black and White Colors

It is universal that black color absorbs the entire radiation incident on it and white color reflects the entire radiation incident on it. Based on this principle, the second positioning of the sensor couple can be made. The IR LED and the photodiode is placed side by side. When the IR transmitter emits infrared radiation, since there is no direct line of contact between the transmitter and receiver, the emitted radiation must reflect back to the photodiode after hitting any object. The surface of the object can be divided into two types: reflective surface and non-reflective surface. If the surface of the object is reflective in nature i.e. it is white or other light color, most of the radiation incident on it will get reflected back and reaches the photodiode. Depending on the intensity of the radiation reflected back, current flows in the photodiode.

If the surface of the object is non-reflective in nature i.e. it is black or other dark color, it absorbs almost all the radiation incident on it. As there is no reflected radiation, there is no radiation incident on the photodiode and the resistance of the photodiode remains higher allowing no current to flow. This situation is similar to there being no object at all.

4.2 IR SENSOR ARRAY

An IR Sensor Array consists of pairs of IR LED's and photodiodes. Generally, the pair count ranges from 5 to 8. In this project, Pololu QTR-8RC reflectance array is used.

QTR-8RC Reflectance Sensor array

This sensor module has 8 IR LED/phototransistor pairs mounted on a 0.375" pitch, making it a great detector for a line-following robot. Pairs of LEDs are arranged in series to halve current consumption, and a MOSFET allows the LEDs to be turned off for additional sensing or power-savings options. Each sensor provides a separate digital I/O-measurable output**.**

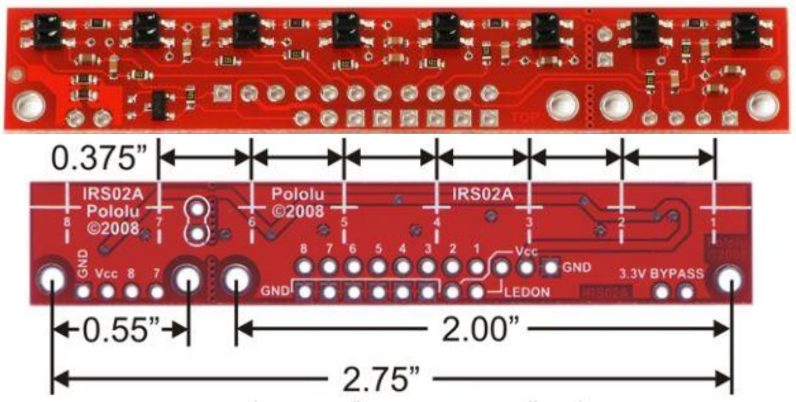


Fig: 4.8 QTR-8RC Reflectance Sensor Array

Description

The QTR-8RC reflectance sensor array is intended as a line sensor, but it can be used as a general-purpose proximity or reflectance sensor. The module is a convenient carrier for eight IR emitter and receiver (phototransistor) pairs evenly spaced at intervals of 0.375" (9.525 mm). To use a sensor, you must first charge the output node by applying a voltage to its OUT pin. You can then read the reflectance by withdrawing that externally applied voltage on the OUT pin and timing how long it takes the output voltage to decay due to the integrated phototransistor. Shorter decay time is an indication of greater reflection. This measurement approach has several advantages, especially when coupled with the ability of the QTR-8RC module to turn off LED power:

- No analog-to-digital converter (ADC) is required.
- Improved sensitivity over voltage-divider analog output.
- The Parallel reading of multiple sensors is possible with most microcontrollers.
- Parallel reading allows optimized use of LED power enable option.

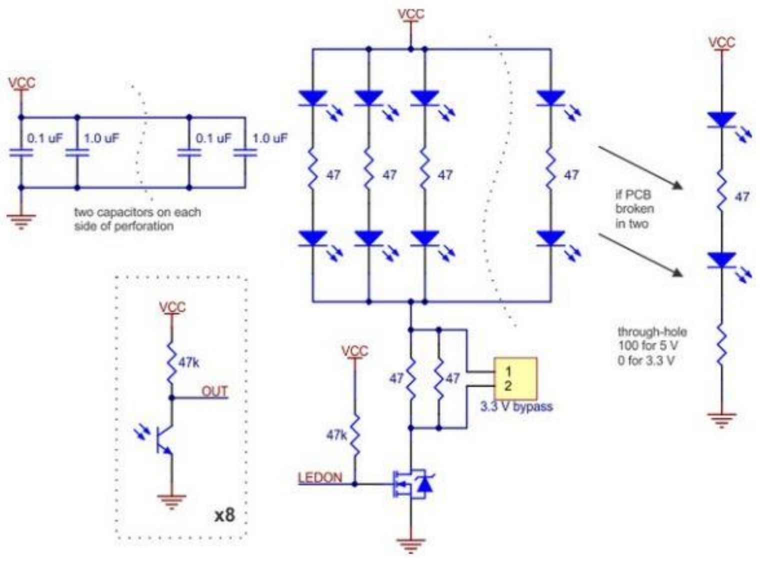

VCC
0.1 uF
1.0 uF
0.1 uF
1.0 uF
two capacitors on each side of perforation
47
if PCB broken in two
through-hole 100 for 5 V 0 for 3.3 V
47k
OUT
x8
LEDON
3.3 V bypass

Fig: 4.9 Reflectance array circuit

The outputs are all independent, but the LEDs are arranged in pairs to halve current consumption. The LEDs are controlled by a MOSFET with a gate normally pulled high, allowing the LEDs to be turned off by setting the MOSFET gate to a low voltage. Turning the LEDs off might be advantageous for limiting power consumption when the sensors are not in use or for varying the effective brightness of the LEDs through PWM control. This sensor was designed to be used with the board parallel to the surface being sensed. The LED current-limiting resistors for 5 V operation are arranged in two stages; this allows a simple bypass of one stage to enable operation at 3.3 V. The LED current is approximately 20–25 mA, making the total board consumption just under 100 mA.

Specifications

- Dimensions: 2.95" x 0.5" x 0.125" (without header pins installed).
- Operating voltage: 3.3-5.0 V.
- Supply current: 100 mA.
- Output format: 8 digital I/O-compatible signals that can be read as a timed high pulse.
- Optimal sensing distance: 0.125" (3 mm).
- Maximum recommended sensing distance: 0.375" (9.5 mm).
- Weight without header pins: 3.09 g.

Interfacing the QTR-8RC Outputs to Digital I/O Lines

The QTR-8RC module has eight identical sensor outputs that, like the Parallax QTI, require a digital I/O line capable of driving the output line high and then measuring the time for the output voltage to decay. The typical sequence for reading a sensor is:

1. Turn on IR LEDs (optional).
2. Set the I/O line to an output and drive it high.
3. Allow at least 10 µs for the sensor output to rise.
4. Make the I/O line an input (high impedance).
5. Measure the time for the voltage to decay by waiting for the I/O line to go low.
6. Turn off IR LEDs (optional).

Module Connections

The QTR-8RC reflectance sensor array is designed to provide some connection flexibility. The pins are standard 0.25 cm spacing and arranged to support connection using 11*2 header strip. Several pins appear in multiple locations, but you can leave duplicate connection points disconnected. For example, you do not need to connect ground to all six ground pins along the lower edge of the board.

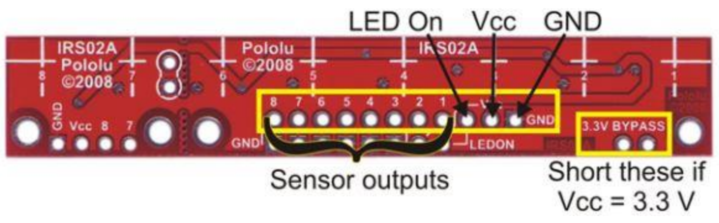


Fig: 4.10 QTR-8RC Module Pins

The Vcc and GND pins are where the sensor array receives its power. Your Vcc connection must be between 3.3 and 5 V and must be able to supply at least 100 mA. The sensor’s outputs will be relative to GND. The module is calibrated so that its IR LEDs will achieve optimal brightness when Vcc is 5 V; using a lower Vcc will decrease the LED brightness or cause them to turn off completely. You can compensate for this by shorting the two 3.3V BYPASS pins together, which bypasses one stage of the LED current-limiting resistors and increases LED brightness.

4.3 DC MOTOR

The DC motor is a device that converts electrical energy into mechanical energy. The DC motor consist of a rotating armature in the form of an electromagnet. A rotary switch known as commutator reversing the direction of the electric current twice every cycle to flow through the armature so that the poles of the electromagnet push and pull against the permanent magnets on the outside of the motor. The armature electromagnet passes the poles of the permanent magnets, since using the poles, the commutator reversing the polarity of the armature electromagnet. During that instant switch of polarity, inertia actuates the classical motor going in the proper direction. DC motors are most easy for controlling. One dc motor has two signals for its operation. Reversing the polarity of the power supply across it can change the direction required. Speed can be varied by varying the voltage across the motor.

Working Principle of a DC Motor

The principle upon which a dc motor works is very simple. If a current carrying conductor is placed in a magnetic field, mechanical force is experienced on the conductor, the direction of which is given by Fleming's left hand rule (also called motor rule) and hence the conductor moves in the direction of the force. The magnitude of the mechanical force experienced on the conductor is given by

$$F = BI_c l_c \; newtons$$

Where B is the field strength is teslas (Wb/m^2), I_c is the current flowing through the conductor in amperes and l_c is the length of conductor in meters.

Before understanding the working of DC motor first, we have to know about their construction. The armature and stator are the two main parts of the DC motor. The armature is the rotating part, and the stator is their stationary part. The armature coil is connected to the DC supply.

The armature coil consists the commutators and brushes. The commutators convert the AC induces in the armature into DC and brushes transfer the current from rotating part of the motor to the stationary external load. The armature is placed between the north and the South Pole of the permanent or electromagnet. For simplicity, consider that the armature has only one coil which is placed between the magnetic field. When the DC supply is given to the armature coil the current starts flowing through it. This current develops their own field around the coil. The Figure shows the field induces around the coil.

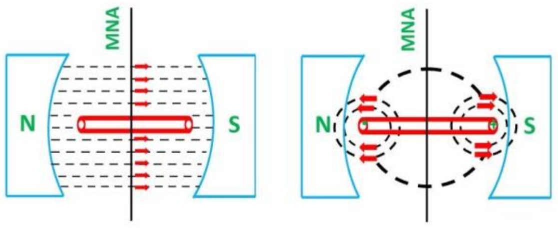


Fig: 4.11 DC Motor Working Principle

By the interaction of the fields (produces by the coil and the magnet), the resultant field develops across the conductor. The resultant field tends to regain its original position, i.e. in the axis of the main field. The field exerts the force at the ends of the conductor, and thus the coil starts rotating.

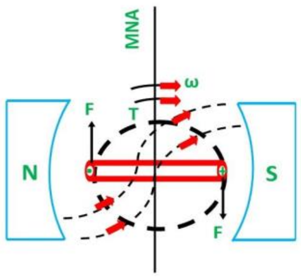


Fig: 4.12 Rotation of the coil

4.4 DC GEARED MOTOR

A gear motor is a specific type of electrical motor that is designed to produce high torque while maintaining a low horsepower, or low speed, motor output. A gear motor can be either an AC (alternating current) or a DC (direct current) electric motor. Most gear motors have an output of 1,200 to 3,600 revolutions per minute (RPMs). These types of motors also have two different speed specifications: normal speed and the stall-speed torque specifications.

Gear motors are primarily used to reduce speed in a series of gears, which in turn creates more torque. This is accomplished by an integrated series of gears or a gearbox being attached to the main motor rotor and shaft via a second reduction shaft. The second shaft is then connected to the series of gears or gearbox to create what is known as a series of reduction gears. Gear motors are commonly used in devices such as can openers, garage door openers, washing machine time control knobs, and even electric alarm clocks. Common commercial applications of a gear motor include hospital beds, commercial jacks, and cranes.

In this project, A Micro Metal Gearmotor is used.

Micro Metal Gearmotor

With a cross section measuring only 10 × 12 mm (0.39″ × 0.47″), these small brushed DC gearmotors are available in a wide range of gear ratios—from 5:1 up to 1000:1—and with five different motors: high-power 6 V and 12 V motors with long-life carbon brushes (HPCB), and high-power (HP), medium power (MP), and low power (LP) 6 V motors with shorter-life precious metal brushes. With the exception of 1000:1 gear ratio versions, all of the micro metal gearmotors have the same physical dimensions (though the terminal size and separation on the HPCB versions differ slightly from the other versions), so it is generally easy to swap one version for another if your design requirements change.

In this project, 12V-600RPM micro metal gearmotor is being used. This N20-12V-600 RPM Micro Metal Gear Motor has a small volume, torsion big and all metal gear, durable, not easy to wear. Great replacement for the rusty or damaged DC geared speed reduce motor on the machine. Widely used on Boat, Car, Electric Bicycle, Fan, Home Appliance. The Motor is of lightweight, high torque, and low RPM. It is equipped with gearbox assembly so as to increase the torque of the motor. It has a cross-section of 10 × 12 mm, and the D-shaped gearbox output shaft is 9 mm long and 3 mm in diameter. It has a very small size so as fit in complex spaces of small-scale applications. One can connect this Micro Gear Motor to wheels to drive them from one place to another while carrying high loads.

Fig: 4.13 Micro Metal Gearmotor

Features

- This is a DC Mini Metal Gearmotor, ideal for making a robot.
- Lightweight, high torque and low RPM.
- Fine craftsmanship, durable, not easy to wear.
- With excellent stall characteristics, can climb hills easily.

- You can also easily mount a wheel on the motor's output shaft.

Specifications

Model No.	N20-12V-600
No load Current	0.06A
Rated Torque(kg-cm)	0.18
Stall Current	0.75A
Stall Torque(kg-cm)	0.65
Rated Operating Voltage	12V
Operating Voltage Range	6V-12V
No Load Speed	600RPM
Shaft Type	D-type, Single side
Shaft Diameter(mm)	3
Weight(gm)	12
Output Axial Length(mm)	10
Screw Size	M3

Table: 4.1 Micro Metal Gearmotor Specifications

Wheels

The white plastic wheels have rubber tires measuring **42 mm (1.65″)** in diameter. These rugged wheels are designed to press-fit securely onto the 3mm D output shafts of our micro metal gearmotors and 15.5D metal gearmotors. These gearmotors should be mounted onto the side of the hub with the protruding teeth, as shown on the right. The output shaft will slide into the socket easily at first but will achieve a snug fit when pressed through to the other edge of the hub.

Fig: 4.14 Wheels

Specifications

- Size – 42*19 mm
- Weight – 19gm
- Shaft Diameter – 3mm

4.5 MOTOR DRIVER

Motor Driver circuits are current amplifiers. They act as a bridge between the controller and the motor in a motor drive. Motor drivers are made from discrete components which are integrated inside an IC. The input to the motor driver IC or motor driver circuit is a low current signal. The function of the circuit is to convert the low current signal to a high current signal. This high current signal is then given to the motor. The motor can be a brushless DC motor, brushed DC motor, stepper motor, or any other DC motor, etc. Motor Drivers are also used to control motors in autonomous circuits. Motor Drivers contain H-Bridge circuit which is the main part in controlling the direction of motors.

H – Bridge

H Bridge is a simple electronic circuit which enables us to apply voltage to load in either direction. It is commonly used in robotics applications to control DC Motors. By using H Bridge we can run DC Motor in clockwise or anticlockwise directions. An H-Bridge circuit contains four switching elements, transistors or MOSFETs, with the motor at the center forming an H-like configuration. By activating two particular switches at the same time we can change the direction of the current flow, thus change the rotation direction of the motor. When switches S1 and S4 are switched on, the motor runs in clockwise direction. When S2 and S3 are switched on, the motor runs in anticlockwise direction.

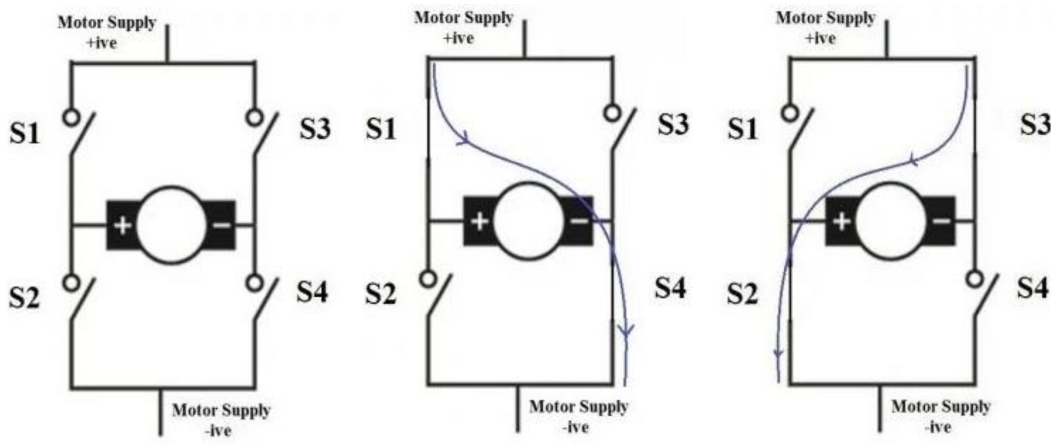


Fig: 4.15 H-Bridge Circuit

Pulse Width Modulation

Pulse Width Modulation (PWM) is a commonly used technique for generally controlling DC power to an electrical device, made practical by modern electronic power switches. The average value of current supplied to the load is controlled by the switch position and duration of its state. If the On period of the switch is longer compared to its off period, the load receives comparatively higher power. Thus the PWM switching frequency has to be faster. PWM, as it applies to motor control, is a way of delivering energy through a succession of pulses rather than a continuously varying (analog) signal. By increasing or decreasing pulse width, the controller regulates energy flow to the motor shaft. The motor's own inductance acts like a filter, storing energy during the "ON" cycle while releasing it at a rate corresponding to the input or reference signal. In other words, energy flows into the load with not much of the switching frequency, but at the reference frequency.

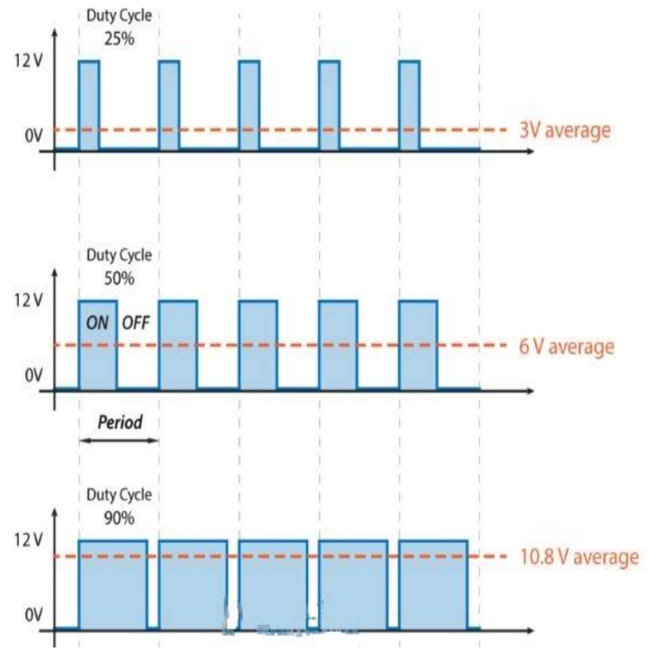


Fig: 4.16 PWM Signal

The average voltage depends on the duty cycle, or the amount of time the signal is ON versus the amount of time the signal is OFF in a single period of time.

L293D Motor Driver

L293D is a dual H-bridge motor driver integrated circuit. It contains two inbuilt H-bridge driver circuits. In its common mode of operation, two DC motors can be driven simultaneously, both in forward and reverse direction. The L293D IC receives signals from the microprocessor and transmits the relative signal to the motors. It has two voltage pins, one of which is used to draw current for the working of the L293D and the other is used to apply voltage to the motors. The L293D switches its output signal according to the input received from the microprocessor. For Example: If the microprocessor sends a 1(digital high) to the Input Pin of L293D, then the L293D transmits a 1(digital high) to the motor from its Output Pin. An important thing to note is that the L293D simply transmits the signal it receives. It does not change the signal in any case.

Fig: 4.17 L293D Motor Driver

The L293D is a 16 pin IC, with eight pins, on each side, dedicated to the controlling of a motor. There are 2 INPUT pins, 2 OUTPUT pins and 1 ENABLE pin for each motor. L293D consists of two H-bridge. H-bridge is the simplest circuit for controlling a low current rated motor.

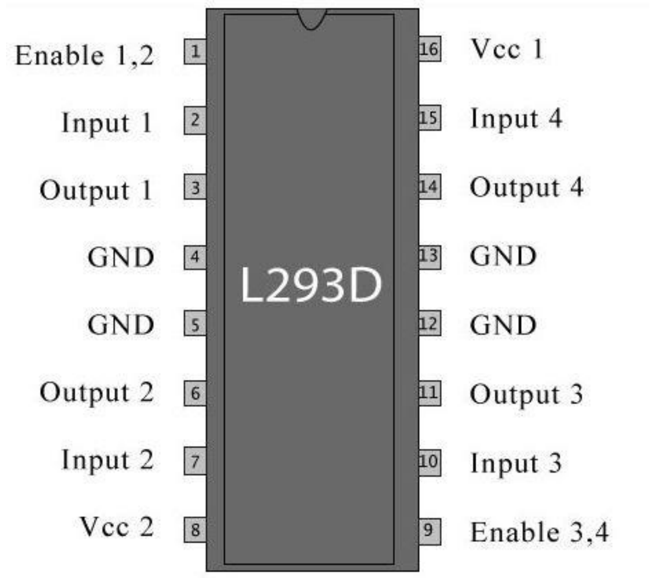


Fig: 4.18 L293D Pin Diagram

- Pin-1 (Enable 1-2): When the enable pin is high, then the left part of the IC will work otherwise it won't work. This pin is also called as a master control pin.
- Pin-2 (Input-1): When the input pin is high, then the flow of current will be through output 1.
- Pin-3 (Output-1): This output-1 pin must be connected to one of the terminals of the motor.
- Pin4 &5: These pins are ground pins.
- Pin-6 (Output-2): This pin must be connected to one of the terminals of the motor.
- Pin-7 (Input-2): When this pin is HIGH then the flow of current will be through output 2.
- Pin-8 (Vcc2): This is the voltage pin which is used to supply voltage to the motor.
- Pin-16 (Vss): This pin is the power source to the integrated circuit.
- Pin-15 (Input-4): When this pin is high, then the flow of current will be through output-4.
- Pin-14 (Output-4): This pin must be connected to one of the terminals of the motor.

- Pin-12 & 13: These pins are ground pins.
- Pin-11 (Output-3): This pin must be connected to one of the terminals of the motor.
- Pin-10 (Input-3): When this pin is high, then the flow of current will through output-3.
- Pin-9 (Enable3-4): When this pin is high, then the right part of the IC will work & when it is low the right part of the IC won't work. This pin is also called as a master control pin for the right part of the IC.

4.6 BATTERY

As the device needs to be wireless, batteries are a viable option. These batteries could be either single-use or rechargeable. Characteristics that must be taken into account include battery size, cell voltage, energy capacity (measured in milliamp-hours [mAh]), rate of discharge (in storage), risks, and cost. In this project, Li-Ion batteries and a 9v cell has been used.

4.7 LITHIUM-ION BATTERY

Lithium-ion batteries (LIB) are a family of rechargeable batteries having high energy density and commonly used in consumer electronics. Unlike the disposable lithium primary battery, a LIB uses an intercalated lithium compound instead of metallic lithium as its electrode. Usually, LIBs are significantly lighter than other kinds of rechargeable batteries of similar size. LIBs are heavily used in portable electronics. These batteries can be commonly found in Laptop, IPod, Power bank, and cell phones. This term is also known as a Li-Ion.

In this project 3 Li-Ion cells of 3.7V, 2200mAh is being used to power the motors.
A 9V Zinc Chloride battery is used to power the Arduino board.

Fig: 4.19 Batteries

Advantages

- A typical LIB is able to store 150 watt-hours electricity per kg of battery, compared to 100 watt-hours electricity in a nickel-metal hydride (NiMH) battery, and only 25 watt-hours electricity in a lead-acid battery.
- LIBs hold a charge well. They usually lose approximately 5% of their charge each month, against a 20% monthly loss for NiMH batteries.
- LIBs do not require complete discharge prior to recharging.
- LIBs are able to handle more charge/discharge cycles.

Disadvantages

- LIBs start to degrade the moment they leave the factory. They usually last for only two to three years from the date of manufacture, regardless of whether used or unused.
- LIBs are highly sensitive to higher temperatures. Higher temperature leads to a much faster degradation rate than normal.
- If a LIB is fully discharged, it gets totally damaged.
- LIBs are comparatively expensive.

4.8 ARDUINO UNO

Arduino microcontrollers are specifically advertised as being good choices to prototype devices; they provide an IDE (Integrated Development Environment) which can use any C code; many components also provide code libraries to work specifically with the Arduino. According to the technical specifications, the Arduino Uno board requires an input voltage of 6-20 volts, which is regulated down to a usable voltage. This means the chosen power source for the Arduino can vary as much as required for the other components. The Uno contains 6 analog I/O pins, which are required to read data from the ultrasonic sensors and to control the stepper motor. 6 pins are required for the device, as 4 are used for the ultrasonic sensors and 2 are required to control the stepper motor. The analog pins output is upto 50 mA of current, which is more than enough for the ultrasonic sensors. There are also 14 digital I/O pins, which are required to control the wireless transmission system.

Features

- Arduino Uno comes with USB interface i.e. USB port is added on the board to develop serial communication with the computer.
- Atmega328 microcontroller is placed on the board that comes with a number of features like timers, counters, interrupts, PWM, CPU, I/O pins and based on a 16MHz clock that helps in producing more frequency and number of instructions per cycle.
- It is an open source platform where anyone can modify and optimize the board based on the number of instructions and tasks they want to achieve.
- This board comes with a built-in regulation feature which keeps the voltage under control when the device is connected to the external device.

- Reset pin is added on the board that reset the whole board and takes the running program in the initial stage. This pin is useful when board the hangs up in the middle of the running program; pushing this pin will clear everything up in the program and starts the program right from the beginning.
- 13KB of flash memory is used to store the number of instructions in the form of code.
- Only 5 V is required to turn the board on, which can be achieved directly using USB port or external adopter, however, it can support external power source up to 12 V which can be regulated and limit to 5 V or 3.3 V based on the requirement of the project.

PIN Diagram

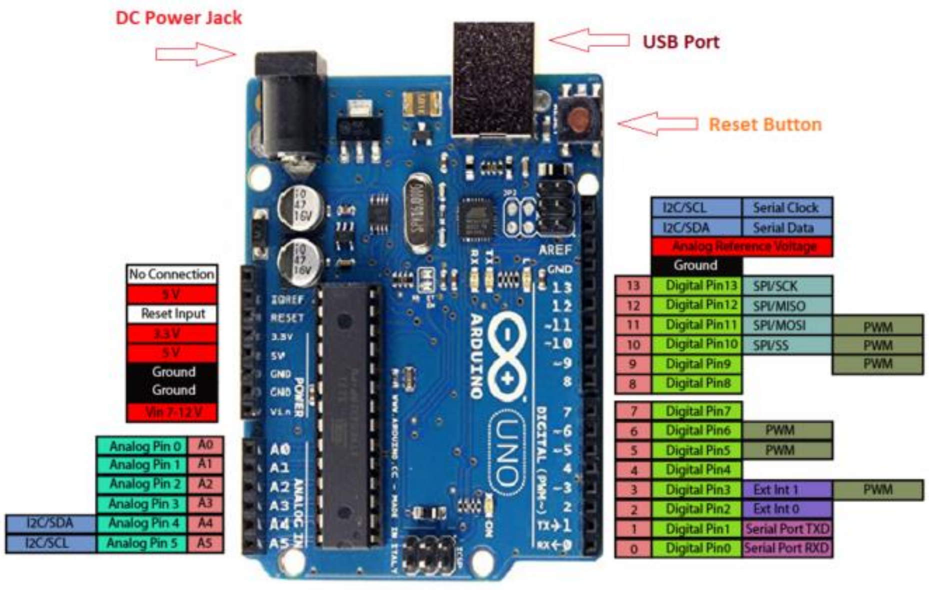


Fig: 4.20 Arduino Uno Pin Diagram

5. CHAPTER
WORKING MODEL

5.1 PROTOTYPE DEVELOPMENT

The first part of the prototype is the chassis. The chassis is of two parts. The base part is a round piece of 1/8" (3 mm) acrylic with a 5" (127 mm) diameter and cutouts for building a differential-drive robot chassis. The chassis has assorted general-purpose holes and slots that support many configurations of sensors and other robot components. The top part is a square wooden piece of dimensions 100mm.

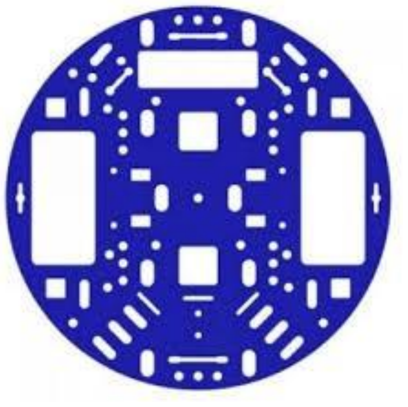


Fig: 5.1 Chassis

Both the parts are fixed together with the help of nuts and bolts. The bolts are of M3 type and 50mm long. The other parts are also fixed onto the chassis with help of nuts and bolts. The prototype after fabrication looks like in the figure below

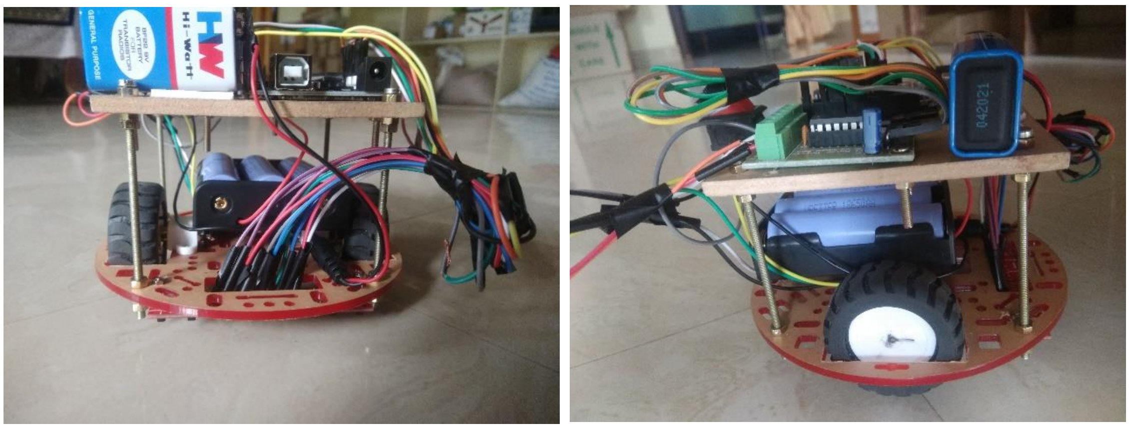
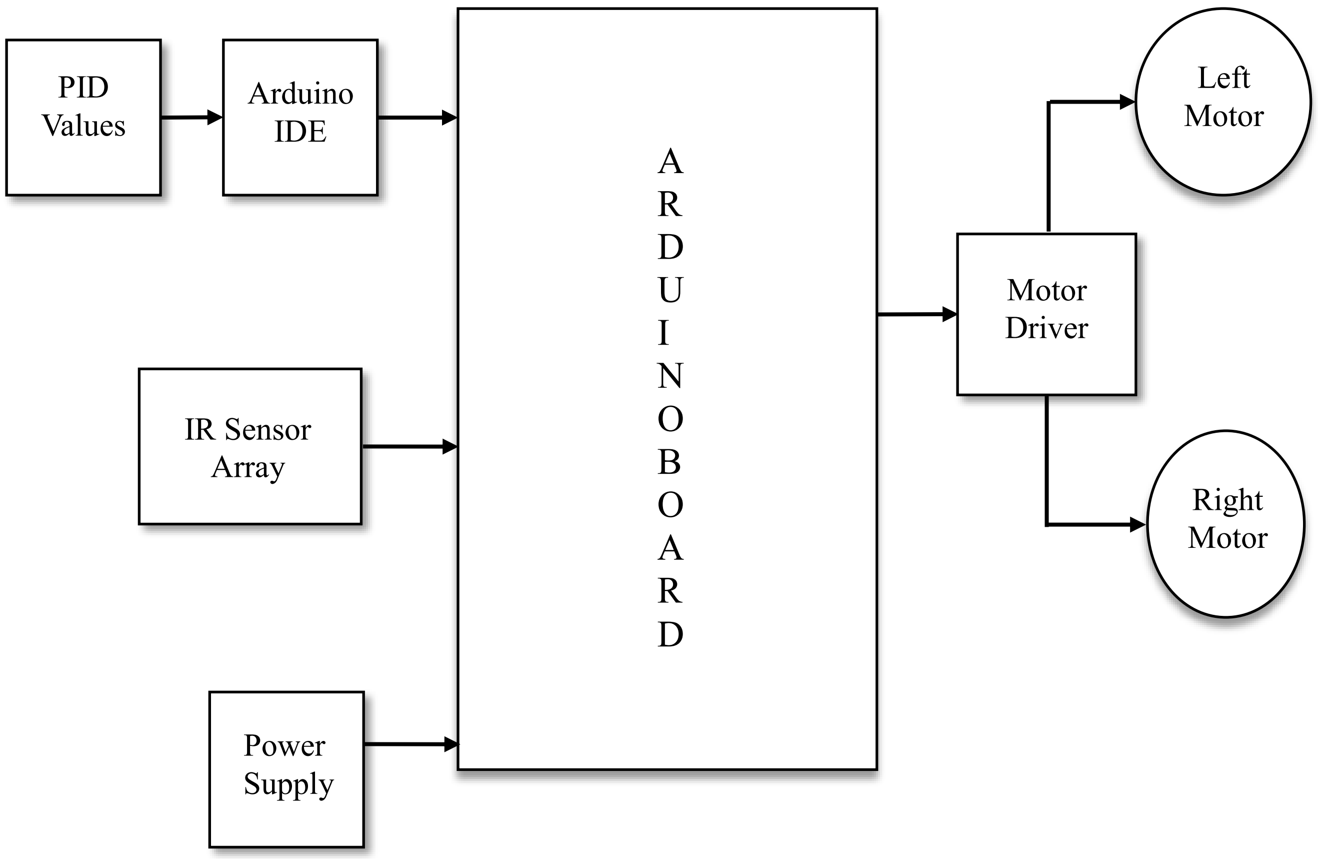
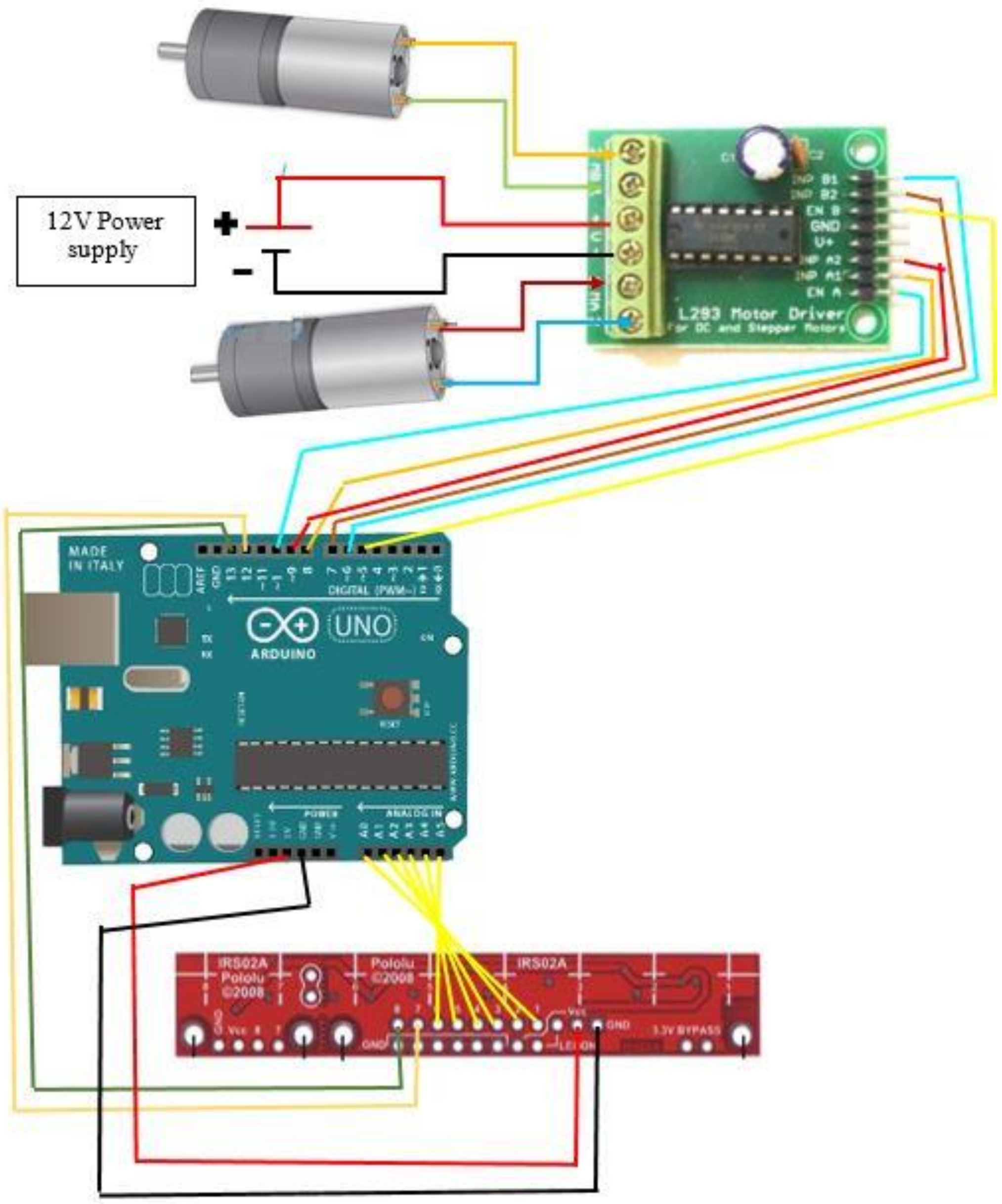


Fig: 5.2 Fabrication of the Prototype

5.2 BLOCK DIAGRAM



5.3 CIRCUIT DIAGRAM



Arduino to Motor Driver connections

Arduino digital Pin 8 – Motor Driver A1

Arduino digital Pin 9 – Motor Driver A2

Arduino digital Pin 10 – Motor Driver ENA

Arduino digital Pin 6 – Motor Driver B1

Arduino digital Pin 7 – Motor Driver B2

Arduino digital Pin 5 – Motor Driver ENB

Arduino to QTR-8RC connections

Arduino Analog Pin A0 – Sensor 1

Arduino Analog Pin A1 – Sensor 2

Arduino Analog Pin A2 – Sensor 3

Arduino Analog Pin A3 – Sensor 4

Arduino Analog Pin A4 – Sensor 5

Arduino Analog Pin A5 – Sensor 6

Arduino Digital Pin 12 – Sensor 7

Arduino Digital Pin 13 – Sensor 8

Arduino 5V – Vcc

Arduino GND- GND

5.4 WORKING

All the parts are fabricated or fixed onto the chassis. Connections between various components of the system are given according to the circuit diagram. The goal is to make the Robot follow the black line on the white surface. For it to follow the line with fewer errors and in a smooth manner we use PID Algorithm. It uses an IR sensor to detect the change in the color of the place it is on. If it is a black, the sensor is on the line. If it is white, it is on the background.

The QTR-8RC sensor needs to be calibrated first. During this phase, each reflectance sensor should be exposed to the lightest and darkest readings they will encounter. The sensors should slide across the line during the calibration phase so that each sensor can get a reading of how dark the line is and how light the ground is. Improper calibration will result in poor readings. A value of 1000 is read if it detects a black line and 0 is read if it detects a white line. The line position command returns a value from 0 to 7000 based on the sensor's position over the black line. The estimate is made using a weighted average of the sensor indices multiplied by 1000, so that a return value of 0 indicates that the line is directly below sensor 0 (or was last seen by sensor 0 before being lost), a return value of 1000 indicates that the line is directly below sensor 1, 2000 indicates that it's below sensor 2. Intermediate values indicate that the line is between two sensors.

First, we will start off by describing the normal behavior of a robot while following a line. While the robot moves over the line it may come into the following conditions, if the line is detected by the Left Sensor, then the robot is programmed to go towards Left, if it is detected by the Right sensor, then it goes towards right and similarly for center position of the line it will go straight. The disadvantage in using this is that the robot may wobble a lot, and if going too fast then it may lose control and stop following the line. When we introduce PID into the motion of Robot then it takes the following factors into consideration: the speed with which the robot is moving from side to side is the robot-centered over the line and how long the robot is not centered on the line. By introducing these, we can get a line following robot which will be very smooth in its motion and can follow it at high speeds as well.

Target Position (Set-Point)

It is the required position for the robot to be in. For a line following robot, it is over the center of the line. It is set to zero. It is basically the mean position/the desired position. It is set as a set point in the program, and all the positions of the robot will be with respect to the target position.

Measured Position

This is a value depending upon the position of the robot with respect to the line. It can either be positive or negative. It involves calculating the distance between the current position of the robot and the target position. It can either be to the right or to the left of the target position and based on that it is given a positive or negative value.

Error

This is the difference between the Target Position and the Measured Position. So the more far away the robot will be from the Target position, the more the value of the error. The magnitude of error depends on which side of the line the robot is in. This also has positive and negative values.

PID Constants

There are 3 constants used in this algorithm. These are the Proportional Constant (Kp), Integral Constant (Ki) and the Derivative Constant (Kd). The values of these constants are set beforehand by the programmer. The values of these constants depend on various things ranging from the weight of the robot to the size of it. The robot dimensions and the actuation system also play an important role in determining the value of these constants. In this project, we are using a manual tuning method to find these gains or constants. The manual tuning method is as follows:

- Start with Kp, Ki, and Kd equaling to 0 and work with Kp first. Try setting Kp to a value of 1 and observe the robot. The goal is to get the robot to follow the line even if it is very wobbly. If the robot overshoots and loses the line, reduce the Kp value. If the robot cannot navigate a turn or seems sluggish, increase the Kp value.
- Once the robot is able to somewhat follow the line, assign a value of 1 to Kd (skip Ki for the moment). Try increasing this value until you see a lesser amount of wobbling.
- Once the robot is fairly stable at following the line, assign a value of 0.5 to 1.0 to Ki. If the Ki value is too high, the robot will jerk left and right quickly. If it is too low, you won't see any perceivable difference. Since Integral is cumulative, the Ki value has a significant impact. You may end up adjusting it by .01 increments.
- Once the robot is following the line with good accuracy, you can increase the speed and see if it still is able to follow the line. Speed affects the PID controller and will require retuning as the speed changes.

PID Implementation in the Robot

For this robot, Arduino is used as the microcontroller board, with an ATMega328 on board. We initialized the various constants such as Kp, Ki, and Kd. We also gave the maximum speed to the wheels for the straight condition. Also, initialize the last error, which is the previous error value & other variables such as integral, derivative and proportional.

The error is calculated as follows,

$$error = position - 3500$$

Where position is the position of the sensor above the line, the set point is taken as 3500 since it is the position between 4 and 5 sensor which is the center of the robot.

A higher error value tells us that the robot is farther away from the line than a lower error value. We then set the integral value as the sum of current integral value & the error. The derivative value was updated to the sum of current derivative value & the last error. We then calculated PID Value as the sum of the product of error value & the K_p constant, product of Ki constant and integral and the product of Kd constant & the derivative. We then give the output to the left wheel and the right wheel as a PWM (Pulse Width Modulation) output, which directly controls the speed of the motor.

$$rightMotorSpeed\ =\ rightBaseSpeed\ +\ PIDvalue;$$

$$leftMotorSpeed\ =\ leftBaseSpeed\ -\ PIDvalue;$$

The advantage of using more number of sensors is that it causes the robot to go faster at curves and follow it more efficiently. The reason to this is that at curves the corner sensors will sense the line and proportionally are able to come back to the line, whereas for lesser number of sensors the robot may just come off the line.

Calculations for motor speed

The motor speeds will be constantly changing based upon the position and constants. For example,

If position is 4000 i.e the black line is above the 5 sensor. Let's assume the values of constants as $K_p = 0.03$, $K_d = 2$, $K_i = 0$. The PID value is

$$I = 0, Last\ error = 450$$

$$error = 4000 – 3500 = 500$$

$$P = 500$$

$$D = 500 - 450 = 50$$

$$I = I + P = 500$$

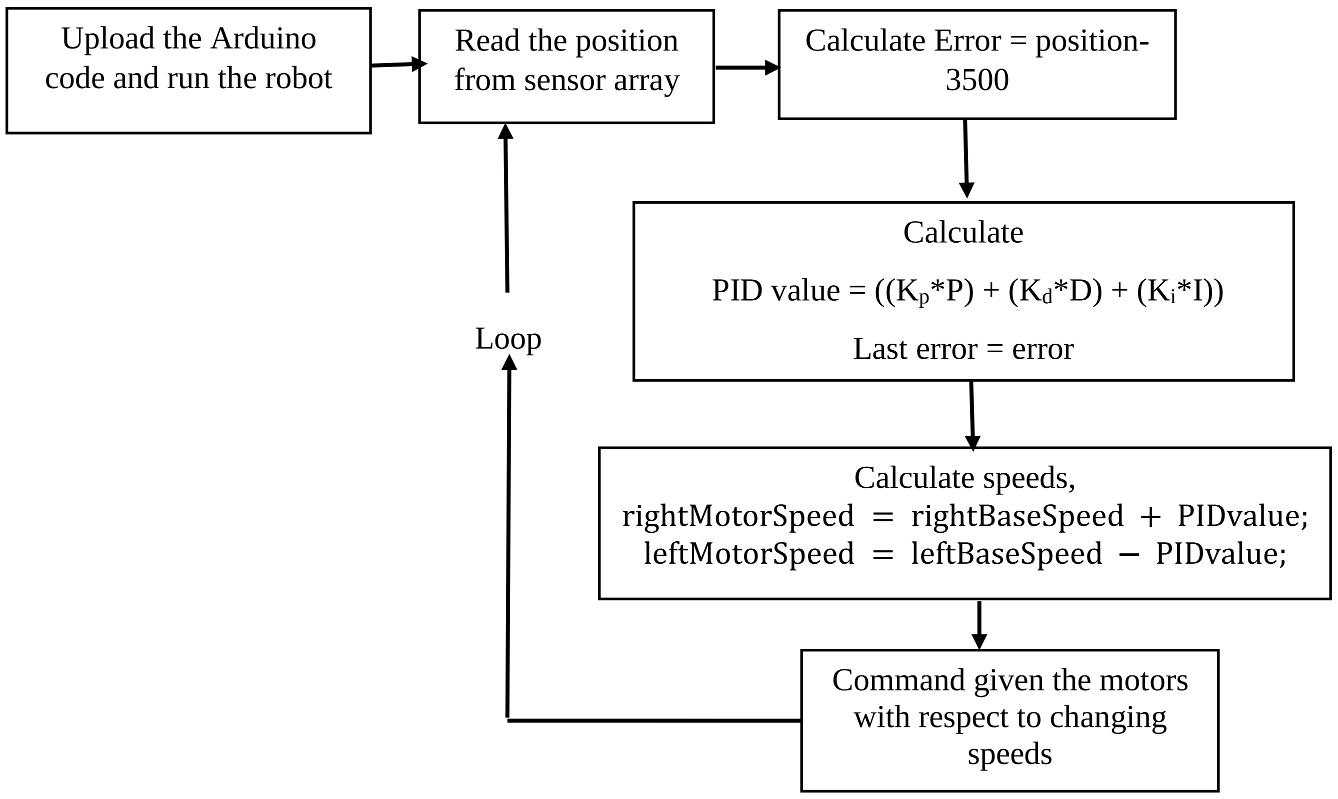
$$PIDvalue = ((0.03*500) + (2*50) + (0*500)) = 115$$

$$rightMotorSpeed\ =\ rightBaseSpeed\ +\ PIDvalue = 180 + 115 = 295$$

$$leftMotorSpeed\ =\ leftBaseSpeed\ -\ PIDvalue; = 180 - 115 = 65$$

In this manner, the motor speeds keep changing according to the positions and the robot moves accordingly at different turns.

5.5 FLOW DIAGRAM


Upload the Arduino code and run the robot
Read the position from sensor array
Calculate Error = position-3500
Calculate
PID value = ((K$_p$*P) + (K$_d$*D) + (K$_i$*I))
Last error = error
Calculate speeds,
rightMotorSpeed = rightBaseSpeed + PIDvalue;
leftMotorSpeed = leftBaseSpeed − PIDvalue;
Command given the motors with respect to changing speeds
Loop

5.6 APPLICATIONS

- PID controller can be used to control temperature, motion, speed, and position.
- Line follower robots are used in the industry to carry parcels or materials from one place to another utilizing the crane system.
- Line follower robots can be used in home for floor cleaning etc.
- In hotels, they are being used for the transfer of things from one place to another following a straight path.
- These can be used in public places like shopping malls, museums, etc to provide path guidance.
- These PID line follower robots can also be used for agriculture and health care purposes.

6. CHAPTER
RESULTS AND DISCUSSION

The PID controlled manual tuning has been performed for the path tracker robot, thus acquiring the constants (Kp, Kd, Ki). It is also found how these constants are dependent on each other in a PID controller. Based on the position an error is produced and the Arduino determines what PID value is. The PID value in this project is used to control two motors that are attached to the robot. After getting PID value, Arduino will send certain PWM signal pattern to the motor driver (L293D). Then the motor driver amplifies the PWM signal to both motors to control the speed.

In this project, various runs or tests have been performed to observe how the performance of the robot changes with varying Kp, Kd, Ki. Through these tests, the following results are presented,

- The Kp value will tell us how the robot will respond to the line, vary the value of Kp from 0 to its best value that produce its best responsivity to the lines. When Kp value is 0, it shows that the robot doesn’t do anything regarding to line. But when it is changed, it is observed that the responsivity is changed and finally we find its best response at a value 0.08.
- The Kd value variation is conducted to change the performance of the turning of the robot. In this case, Kd value in this project will help to increase the speed of the motors when having turns. It can be concluded that giving Kd value will change the performance of turning. But there is a limit when the speed of turning is exceeding, it will make the robot out of reach from the line while turning.
- The addition of Ki value has caused the robot to overshoot much because of variations in the PID values. So, the Ki value is 0 in this project.

The following tables show the response of the robot for variations in the constants. The robot is moved on a track of length 2m. The best possible response of the robot is found at value Kp = 0.08, Kd = 3, Ki = 0.

Kp	Time(sec)	Response
0	-	Nothing changes
1	17	More overshoot, Line detection problem, jerky movement
0.5	16	More overshoot, Line detection problem, jerky movement
0.3	15	More overshoot, Line detection problem, jerky movement
0.1	14.5	More overshoot, Line detection problem, jerky movement
0.03	14.8	Less overshoot, jerks, more time
0.04	14.5	jerky movement, overshoot
0.06	13.8	Jerky movement but follows the line
0.08	13.3	Follows line

Table: 6.1 Kp response

Kd	Time(sec)	Response
0	-	Nothing changes
1	13	No jerks but overshoots more
2	12.5	Follows line smoothly but overshoots at turns
4	13.2	Overshoots more and no jerks
3	11.8	Follows line perfectly

Table: 6.2 Kd response

Ki	Time(sec)	Response
0	-	Nothing changes
1	16	More overshoots and does not follow the line

Table: 6.3 Ki response

7. CHAPTER
CONCLUSION AND FUTURE SCOPE

CONCLUSION

Before using the PID, the problem faced was that the robot did not follow the line smoothly. It wasn’t efficiently following the path, so to improve this PID logic is implemented. It is started with implementing the proportional part and code was written for it. After running the robot it was noticed that while following the wobbling of the robot was lesser than before, but it was still a big issue. The line following was further improved after adding the differential part to the code, which reduced the overshooting of the robot while trying to return to the line.

While implementing PID, a major problem faced was in setting up the constants for the Proportional, Integral and Differential. Firstly, it started by setting each value as 1 and then made the changes by either increasing it or decreasing it and seeing the changes in the behavior of the robot. The constants had to be changed to many times before arriving at the perfect values.

FUTURE SCOPE

The Future scope of this project is to find out a better tuning method to tune the PID controller. This can be done by using the Ziegler Nichols method. This can be done by finding out the transfer function of the process and tuning the value of the proportional gain up to the extent when the output starts oscillating. Noting down the gain and time period of oscillations, the value of the integral and derivative gain can also be determined.

The manual tuning can also be improved by Bluetooth control which helps in sending the constants through an app without uploading the code again and again.

REFERENCES

[1]. “Arduino Line follower Robot”. Internet: https://www.electronicshub.org/arduino-line-follower-robot/

[2]. Anirudh Sunil Nath. “Implementation of PID Control to Reduce Wobbling in a Line Following Robot”, IJRET, Volume: 2 Issue: 10, Oct 2013.

[3]. Ebiesuwa O. O. “Line Follower Robot Using A Sophisticated Sensor Approach”, IJRET, Volume: 2 Issue: 7, July 2013.

[4]. PID for Line Following. (2007*)* “PID for Line following”. [Internet]. Available at http://www.chibots.org/index.php?q=node/339.

[5]. Hari Om Bansal, Rajamayyoor Sharma, P. R. Shreeraman. “PID Controller Tuning Techniques: A Review”, JCET, Volume: 2 Issue: 4, 2012.

[6]. “Experimental Tuning of PID Controllers”. [Internet]. Available at http://techteach.no/fag/tmpp250/v06/pidcontrol/pid_tuning

[7]. “Introduction: PID Controller Design”. [Internet]. Available at http://ctms. engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID.

[8]. Araki M. “Control systems, Robotics, And Automation-Vol 2- PID Control”. Ency-clopedia of Life support systems.

[9]. “PID Controller Tuning”. [Internet]. Available at http://shodhganga.inflibnet.ac.in/ bitstream/10603/108862/12/12_chapter%202.pdf

[10]. “Motor Driver IC”. [Internet]. Available at https://2019.robotix.in/tutorial/ auto/motor_driver/

[11]. “QTRSensors Library”. [Internet].Available at https:// http://pololu.github.io/ qtr-sensors-arduino/

APPENDIX

Arduino Code

```
#include <QTRSensors.h>

#define Kp 0.08// start with this slowly by increasing from 1. If at 1 it oveshoots decrease
the value
#define Kd 3 // adjust this for varying speeds to reduce woobling
#define Ki 0
#define rightMaxSpeed 220 // max speed of the robot
#define leftMaxSpeed 220 // max speed of the robot
#define rightBaseSpeed 180 // this is the speed at which the motors should spin when the
robot is perfectly on the line
#define leftBaseSpeed 180  // this is the speed at which the motors should spin when the
robot is perfectly on the line
#define rightMotor1 8
#define rightMotor2 9
#define rightMotorPWM 10
#define leftMotor1 6
#define leftMotor2 7
#define leftMotorPWM 5


QTRSensors qtr;
const uint8_t SensorCount = 8;
uint16_t sensorValues[SensorCount];

void setup()
{
  pinMode(rightMotor1, OUTPUT);
  pinMode(rightMotor2, OUTPUT);
  pinMode(rightMotorPWM, OUTPUT);
  pinMode(leftMotor1, OUTPUT);
  pinMode(leftMotor2, OUTPUT);
  pinMode(leftMotorPWM, OUTPUT);


   qtr.setTypeRC();
qtr.setSensorPins((const uint8_t[]){A0,A1,A2,A3,A4,A5,12,13}, SensorCount);
//qtr.setEmitterPin(2);
```

```
// 2.5 ms RC read timeout (default) * 10 reads per calibrate() call
  // = ~25 ms per calibrate() call.
  // Call calibrate() 400 times to make calibration take about 10 seconds.
  for (uint16_t i = 0; i < 400; i++)
  {
    qtr.calibrate();
  }
  digitalWrite(LED_BUILTIN, LOW);

Serial.begin(9600);
delay(4000); // wait for 4s to position the bot before entering the main loop

  }

int lastError = 0;
int I=0;
int P;
int D;

void loop()
{
 uint16_t position = qtr.readLineBlack(sensorValues);
 int error = position - 3500;
 Serial.println(error);
  P = error;
 D = error - lastError;
 I = I+P;

  int PIDvalue = (Kp * P)+(Kd *D)+ (Ki *I)  ;
  lastError = error;

  int rightMotorSpeed = rightBaseSpeed + PIDvalue;
  int leftMotorSpeed = leftBaseSpeed - PIDvalue;

    if (rightMotorSpeed > rightMaxSpeed ) rightMotorSpeed = rightMaxSpeed; // prevent
the motor from going beyond max speed
  if (leftMotorSpeed > leftMaxSpeed ) leftMotorSpeed = leftMaxSpeed; // prevent the
motor from going beyond max speed
  if (rightMotorSpeed < 0) rightMotorSpeed = 0; // keep the motor speed positive
  if (leftMotorSpeed < 0) leftMotorSpeed = 0; // keep the motor speed positive

   {
```

```
 digitalWrite(rightMotor1, HIGH);
 digitalWrite(rightMotor2, LOW);
 analogWrite(rightMotorPWM, rightMotorSpeed);

 digitalWrite(leftMotor1, HIGH);
 digitalWrite(leftMotor2, LOW);
 analogWrite(leftMotorPWM, leftMotorSpeed);

}
}
```