NATIONAL INSTITUTE OF TECHNOLOGY, RAIPUR

G.E. ROAD, RAIPUR(C.G.)

A PROJECT REPORT

ON

DESIGNING A PID CONTROLLER FOR CONTROLLING SPEED OF A DC MOTOR USING ARDUINO

Submitted By –

Himani Sahu (20117033)

Pragati Agrawal (20117070)

Yogesh Narayan Singh (20117114)

Department of Electrical Engineering

National Institute of Technology, Raipur (C.G.)

Under the Supervision of

Dr. S. Ghosh

(Associate Professor)

Department of Electrical Engineering

National Institute of Technology

Raipur (C.G.), 492010

CERTIFICATE

This dissertation entitled project on **"Designing a PID Controller for controlling speed of a DC Motor using Arduino"** by Himani Sahu (20117033), Pragati Agrawal (20117070) & Yogesh Narayan Singh (20117114) has been carried out under my guidance and supervision for the partial fulfillment for the Degree of Bachelor of Technology in Electrical Engineering of **National Institute of Technology, Raipur Chhattisgarh**.

Guided By:

Dr. S. Ghosh

Associate Professor

Department of Electrical Engineering

NIT Raipur

Countersigned By:

Dr. (Mrs.) A. Yadav

Head Of Department

Department of Electrical Engineering

NIT Raipur

CANDIDATE DECLARATION

We declare that this written submission entitled **"Designing a PID Controller for controlling speed of a DC Motor using Arduino"** for the Bachelor of Technology in Electrical Engineering of National Institute of Technology, Raipur Chhattisgarh represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source our submission.

Himani Sahu
Roll No. – 20117033

Pragati Agrawal
Roll No. – 20117070

Yogesh Narayan Singh
Roll No. – 20117114

ACKNOWLEDGEMENT

We would like to express sincere gratitude to several individuals and our institution for supporting us throughout this project. First, we wish to express our sincere gratitude to all of our professors, for their sincere effort, enthusiasm, patience, insightful comments, helpful information, practical advice, and unceasing ideas that have helped us tremendously at all times.

A special mention to **Dr. S. Ghosh**. His immense knowledge, profound experience, and professional expertise have enabled us to complete this major project successfully. Without his support and guidance, this project would not have been possible. We could not have imagined having a better supervisor for our guidance.

We also wish to express our sincere thanks to our institute **National Institute of Technology, Raipur** as well as **Dr. (Mrs.) A. Yadav (HOD)** of Electrical Engineering, Department for allowing us this opportunity to explore and experiment with the area of our interest during the project.

Finally, last but not least, we want to extend gratitude to everyone in our batch and our teammates who were there throughout the project for any kind of support or help. It was great sharing a platform with all of you during the course of this project.

All Team Members

ABSTRACT

This project focuses on the designing of an Arduino based Proportional-Integral-Derivative (PID) controller for the precise speed control of a DC motor. The PID controller is employed to get accurate control over motor's rotational speed. The Arduino microcontroller platform is utilized as the control interface, facilitating seamless integration between the PID algorithm and the hardware components. The system's performance is evaluated through comprehensive simulations and experimental validations, demonstrating the effectiveness of the PID controller in achieving stable and responsive speed control.

The primary objective of this study was to achieve precise and stable speed regulation across a broad range of desired speeds, and hence the analysis is carried out for different values of set speed. The PID controller, implemented through the Arduino microcontroller, emerged as a highly effective solution.

In this project, the modelling of the entire control system, control and simulation has been done in MATLAB/Simulink 2023b version and as a microcontroller Arduino UNO is used.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

1.1 Background

A DC (direct current) motor is a device that converts electrical energy into mechanical motion. Its operation is based on the principle of Lorentz force, according to which - when a current-carrying conductor placed in a magnetic field it experiences a force which leads to rotational motion. The two main parts of DC motor are - rotor and stator. Stator is the stationary part whereas rotor is the rotating part. DC motors come in various types, such as brushed, brushless, series-connected, shunt-connected, and compound-connected, each designed for specific applications. Their affordability and simplicity make them valuable for research and laboratory experiments. The precise speed control they offer is a key factor in their competitiveness in modern industrial applications. Speed adjustment in DC motors can be achieved through variations in armature voltage (voltage control method) or changes in field current (flux control method). While historical methods involved introducing variable resistane in the armature or field circuit for speed control, but in recent times DC motors are commonly controlled using power electronics systems known as DC drives.

A control system refers to a network of interconnected components designed to achieve a specific system response. Arduino hardware functions as the intermediary between the computer and an external entity, such as a DC motor under control. The development of the user interface takes place within the Arduino environment. Various controller types, including P, PI, PD, PID, and fuzzy logic controllers (FLC), can be utilized to regulate DC motor speed. Regulating the speed of a DC motor is critical in industrial contexts to mitigate issues like excessive overshoot and slow rise time, preventing potential damage to machinery. To overcome this problem, in this particular project, an Arduino UNO will be employed to implement a Proportional Integral Derivative (PID) controller.

The aim of this project is to create a PID controller for the precise control of a DC motor's speed. Arduino, known as an open-source platform, forms the basis for constructing electronic projects. Arduino comprises both a tangible programmable circuit board, commonly called a microcontroller, and software known as an Integrated Development Environment (IDE). This IDE runs on a computer and is utilized for writing and uploading computer code onto the physical board. This integrated system allows for the development and implementation of diverse electronics projects.

1.2 Problem Statement

Numerous applications demand the adjustment of a DC motor's speed to optimize the functionality and overall performance of machines. Achieving this deliberately and when needed involves implementing speed control. This can be executed either manually by operators or through the utilization of automated technological devices. It's essential to note that speed control, as distinct from speed regulation, pertains to intentionally adjusting the speed rather than maintaining a constant speed despite variations in load.

There are two main types of speed controllers: armature controls and field controls. Armature controls involve changes in terminal voltage or external resistance, impacting the motor's function. Conversely, field control involves altering the magnetic flux as a means of achieving speed control.

Armature control employs various methods to vary the voltage. One approach is the utilization of armature resistance, achieved by introducing a variable resistance in series with the armature circuit. As the resistance increases, the current flowing through the circuit decreases, resulting in a reduced armature voltage drop compared to the line voltage. Consequently, the motor speed decreases proportionally to the applied voltage. This armature resistance control method is typically applied in situations where motor speed variation is needed for brief intervals rather than continuously. Additional methods of armature control include armature voltage control and shunt resistance control.

Despite being fast and well-suited for brief, specific tasks, armature-controlled motors come with certain drawbacks when weighing the pros and cons of armature control versus field control:

1. Higher initial costs: The armature control approach typically involves greater upfront expenses compared to the field control method.
2. Low energy efficiency: Armature control is commonly applied for shorter durations due to speed variation leading to substantial power wastage. This inefficiency in energy utilization makes the overall process less energy-efficient and more expensive.

In the field control method for DC motors, adjusting the motor's speed involves either weakening the field to increase speed or strengthening the field to decrease speed. To achieve speeds beyond the rated speed, several approaches can be employed. This includes introducing variable resistance in series with the field circuit, altering the reluctance of the magnetic circuit, or adjusting the applied voltage to the motor's field circuit while maintaining a constant voltage supply to the armature circuit.

Field-controlled DC motors are favoured by motor operators and manufacturers due to their ease of use and hassle-free operation. However, there are situations where alternative motor control methods might be more effective. Some drawbacks of field control include:

1. Speed limitations: Field-controlled DC motors are restricted to operating above the normal speed. If your application demands adjustments below the standard speed, opting for an armature-controlled method might be more suitable. Additionally, higher speeds in field control can lead to reduced torque.
2. Reduced stability: While the field control method allows for achieving speeds higher than the norm, its overall range may be constrained by a lack of stability. Operating with a weaker field might only permit safe operation at specific speeds, limiting the flexibility of the motor.

Due to these disadvantages conventional speed control methods are not preferred these days. Therefore, implementing a computer-based controller offers a more cost-effective and time-saving alternative. The advantages associated with these controllers includes: Precise speed control, electrical protection for motor and associated mechanical components, consistent speed maintenance even with changing load, provides dynamic response to changing system’s demand and increased efficiency.

1.2 Objectives of the Project

The objectives of this project are:

i. To design discrete model of a DC Motor in MATLAB Software.

ii. To design and implement a PID controller system for precise speed control of a DC motor using Arduino UNO.

THEORY

2.1 DC Motor

DC motors are widely used in the industry control area for a long time due to its various advantages. For example, high starting torque characteristic, high response performance and ease to be linearly controlled.

Various types of DC motor are: brushless, brushed, shunt, series and compound.

DC motors find a wide array of applications due to their various types, each offering specific advantages. In residential settings, diminutive DC motors power tools, toys, and a range of household appliances. Retail environments often deploy DC motors in conveyors for precise movement control and in turntables for smooth rotations. In industrial contexts, large DC motors play crucial roles in applications like braking systems and reversing mechanisms. Each type of DC motor serves distinct purposes, making them versatile solutions for different needs and industries.

DC motors are well-regarded for their versatile control capabilities, allowing adjustments to speed, torque, and even the direction of rotation. They find widespread use in speed control applications due to their cost-effectiveness, excellent drive performance, and minimal maintenance requirements. DC motors offer the added advantage of providing high starting torque at low speeds, and their speed can be controlled over a broad range.

The equivalent electrical circuit of a DC motor is shown in Figure below. It can be symbolized by a voltage source (V) connected to the armature coil. The induced voltage (E), resulting from the rotation of the electrical coil within the fixed flux lines of the permanent magnets, is commonly known as back electromotive force (back emf).

The following equations are applicable for all DC motors –

$$E = K_e \Phi \omega_m \quad \text{(eq.1)}$$

$$V = E + I_a R_a \quad \text{(eq.2)}$$

$$T = K_e \Phi I_a \quad \text{(eq.3)}$$

Where,

Φ = flux per pole;

I_a = armature current;

V = armature voltage;

R_a = Resistance of the armature circuit;

ω_m = Speed of the motor;

T = Torque developed by the motor,

K_e = Motor Constant

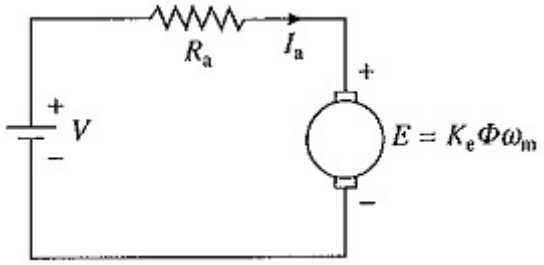


Fig 2.1: Equivalent Circuit of DC Motor

In the past, the rheostat armature control method was commonly employed for regulating the speed of low-power DC motors. Nevertheless, the advent of static power converters, renowned for their enhanced controllability, cost-effectiveness, increased efficiency, and higher current-carrying capacities, marked a significant shift in the landscape of electrical drives. The desired torque-speed characteristics can be achieved by using conventional proportional-integral-derivative (PID) controllers. As PID controllers require exact mathematical modelling, the performance of the system changes in case of any parameter variation.

DC motors are often subjected to speed control for a variety of reasons, each depending on the particular application. The motor's speed must be regulated to meet the demands of the system it powers in many situations. Energy conservation is another motivation for controlling the speed of a DC motor, running a motor at maximum speed all the time can be wasteful and expensive, consuming significant amounts of energy, by matching the motor's speed to the system's requirements, energy usage can be minimized, resulting in lower operating costs.

The block diagram of a DC motor is illustrated in Figure: 2.2, as depicted below. In DC motor, the electrical input, comprising the supply voltagev(E) and current (I), is provided through the electrical port (input port). The mechanical output, involving torque (T) and speed (ω), is then obtained from the mechanical port (output).

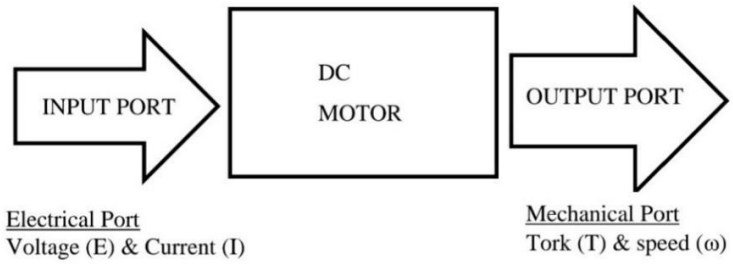


Figure 2.2: Block Diagram of DC Motor

In a DC motor, power is directly supplied to the armature from a DC source, whereas an induction motor receives power induced in the rotating device. DC motors find applications in scenarios where speed control is a key requirement. The standout feature of DC motors is their capacity for speed control. Through the manipulation of the applied voltage using a rheostat (variable resistor), the speed can be adjusted smoothly from zero to the motor's maximum revolutions per minute (rpm).

2.2 Transfer function of a simple DC Motor

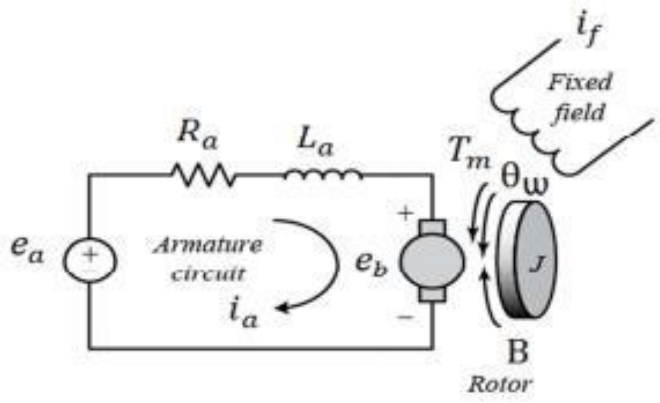


Fig. 2.3: Equivalent circuit of a DC Motor with R_a, L_a and e

Figure 2.3 depicts the equivalent circuit of DC Motor with armature resistance (R_a), self-inductance (L_a), induced emf (e).

The terminal voltage (V_t) can be determined as,

$$V_t = E_b + R_a \cdot I_a + L_a \cdot \frac{dI_a}{dt} \qquad \ldots(1)$$

In steady state condition, armature current and rate of armature current change both are zero. Hence, equation (1) becomes:

$$V_t = E_b + R_a \cdot I_a \qquad \ldots(2)$$

Air gap power in terms of electromagnetic speed and torque is written as,

$$P_a = \omega_m T = E_b \cdot I_a \qquad \ldots(3)$$

Where,

ω_m is the speed of motor,

T is the torque,

And I_a is the armature current.

Therefore, electromagnetic torque can be written as,

$$T = \frac{E_b \cdot I_a}{\omega_m} \qquad \ldots(4)$$

The back emf (E_b) is given by:

$$E_b = K_b \cdot \omega_m \qquad \ldots(5)$$

Therefore,

$$T = K_b I_a \qquad \ldots(6)$$

Moment of inertia *J* of DC motor, viscous friction coefficient B and an acceleration torque T_a is provided by:

$$T_a = T - T_l = J \cdot \frac{d\omega_m}{dt} + B \cdot \omega_m \qquad \ldots(7)$$

Where, T_l is load torque. Eq. (1) and (6) describe dynamics of DC Motor with load condition. With Laplace Transform for Eq. (1) & Eq. (6), gives

$$I_a(s) = \frac{V(s) - K_b\omega_m(s)}{R_a + sL_a} \quad \ldots(8)$$

$$\omega_m(s) = \frac{k_b I_a(s) - T_l(s)}{B + sJ} \quad \ldots(9)$$

The open-loop transfer function of DC motor's speed is expressed as:

$$\frac{\theta}{V} = \frac{K}{(sJ+B)+(Ls+R)+K^2} \quad \ldots(10)$$

2.3 PID Controller

PID controllers find widespread use in various industrial applications, with approximately 95 percent of closed-loop operations in the industry relying on them. PID stands for Proportional, Integral, and Derivative, denoting that all three controllers are integrated in such a way to generate a control signal.

This controller evaluates a measured value derived from a process, typically associated with an industrial or system output, by comparing it to a predetermined reference set point, representing the desired value specified by the operator. The resulting error signal, which signifies the difference between the target value and the measured process value, is then utilized to compute a new input to manipulate the process. This manipulation aims to realign the measured process value with the user/operator's desired value.

The PID controller, renowned for its capability to adjust process outputs based on the past values and rate of change of the error signal, contributes to a more precise, accurate, and stable operational state. PID controllers offer flexibility in adjustment or tuning to suit specific applications, making them highly regarded. Through the fine-tuning of three parameters (K_p, K_i, and K_d) within the PID controller algorithm, the controller can effectively administer control actions tailored to the requirements of a particular process. The response of the controller algorithm is characterized by its sensitivity to errors, the extent of overshooting beyond the set point, and the level of system oscillation.

For this project, the adoption of a PID controller is suggested because of its simplicity, robustness, ability to offer closed-loop response characteristics, and its effectiveness in regulating the time-domain behaviour of various types of plants. The PID controller integrates proportional, integral, and derivative terms, with each of these terms customizable by the user. To enhance control precision, these terms require adjustment, a process referred to as PID tuning.

The proportional, integral, and derivative terms are added together to calculate the output of the PID controller. By defining u(t) as the controller output, the general form of a PID controller as a continuous function of time is given by:

$$u(t) = K_p \cdot e(t) + K_i \int_0^t e(t)dt + K_d \cdot \frac{de}{dt} \quad \ldots(11)$$

where,

u(t) = control signal(input to the plant i.e. DC Motor in our case)

K_p = Proportional Gain,

K_i = integral Gain,

$\int_0^t e(t)dt$ is a summation of all past error over time,

K_d = derivative Gain,

$\frac{de}{dt}$ is rate of change of error.

The PID controller comprises three essential terms: the proportional term, integral term, and derivative term. The proportional term adjusts the controller output based on the magnitude of the error, the integral term works to eliminate steady-state errors in the control system, thereby enhancing steady-state response, and the derivative term predicts the error trend, thereby improving the transient response of the system. These functionalities suffice for the majority of control processes. Due to its simple structure, the PID controller stands out as the most widely employed control method in various industrial applications. The key to optimal control performance lies in appropriately adjusting the proportional gain (K_p), integral gain (K_i), and differential gain (K_d) within the PID controller.

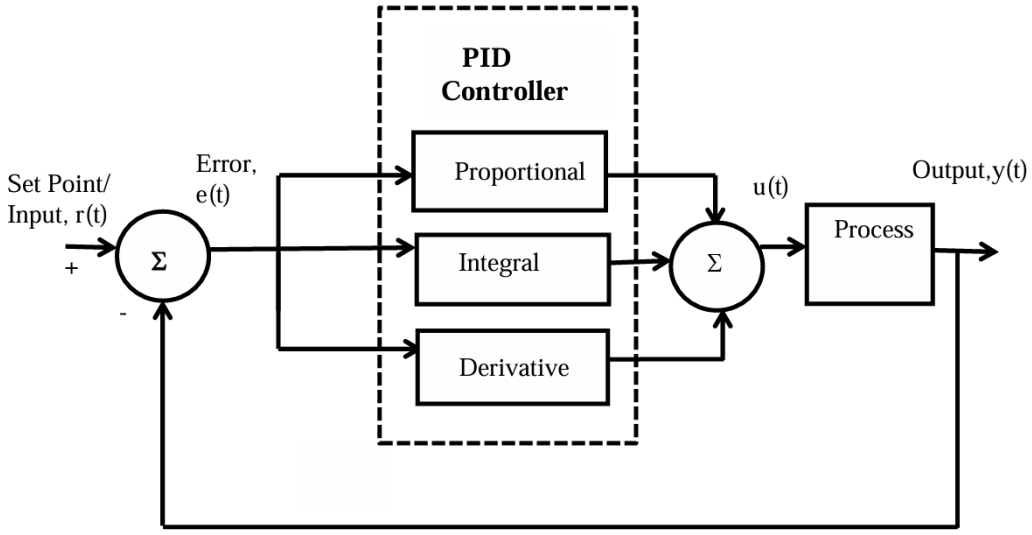


Fig. 2.4: General block diagram of a PID Controller

In a fundamental control system, the controller assesses the measured value in relation to a set point or reference voltage, generating an error value. Subsequently, the error signal directs the application of the necessary corrective action. The PID controller's parameters, namely Kp, Ki, and Kd, are adjustable, allowing for manipulation to yield diverse response curves from a motor controller.

$$e(t) = r(t) - y(t) \qquad \ldots(12)$$

where:

r(t) is a set point or reference value

y(t) is a measured value or process variable

2.4 Arduino UNO Microcontroller

The Arduino Uno stands as a widely embraced open-source microcontroller board within the Arduino ecosystem, celebrated for its adaptability in electronic project development and prototyping.

At its core lies the ATmega328P microcontroller, a product of Atmel (now under Microchip Technology), renowned for its high performance and low power consumption. The board consists of 14 digital input/output pins, with 6 supporting PWM output, along with 6 analog inputs. Operating at a clock speed of 16 MHz, it provides a robust foundation for diverse applications. With 32 KB of flash memory for program storage, 2 KB of SRAM, and 1 KB of EEPROM, the Arduino Uno offers substantial memory resources. Its USB connectivity allows seamless interaction with a computer, facilitating programming and serial communication. The board can be powered either through USB or an external supply within the voltage range of 7 to 12 volts. Programming is streamlined through the Arduino IDE, a user-friendly platform encompassing code development, compilation, and uploading using C/C++.

Furthermore, the Arduino Uno is emblematic of open-source hardware, embodying transparency and accessibility by providing users with the freedom to access design files and specifications for modification. Its simplicity and versatility render the Arduino Uno a preferred choice for a wide spectrum of projects, catering to the needs of hobbyists, students, and professionals, ranging from basic LED applications to intricate robotics and automation endeavours.

The Arduino Uno microcontroller board based on the ATmega328 will be used for this project.

The physical hardware of Arduino Microcontroller i.e., the front view (fig. 2.5) and back view (fig. 2.6) is depicted below.

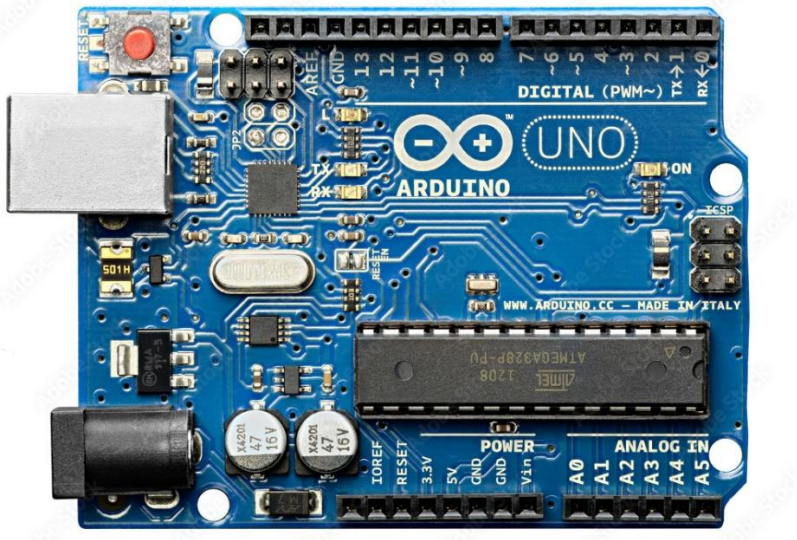


Fig. 2.5: Arduino UNO (FRONT)

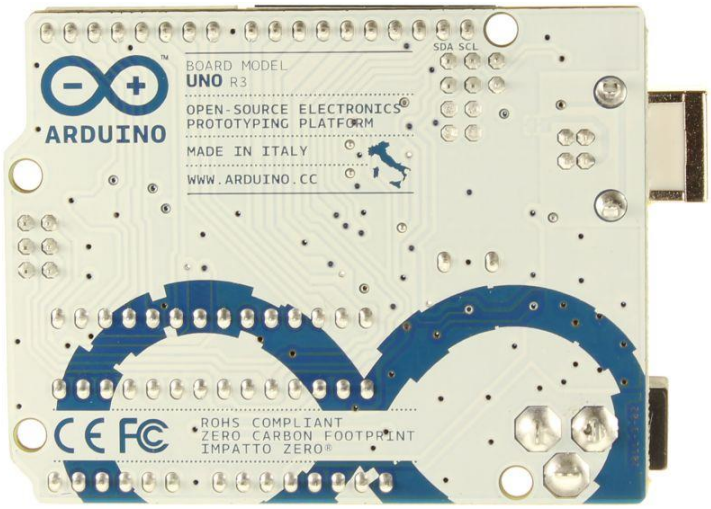


Fig. 2.6: Arduino UNO (REAR)

Advantages of Arduino include:

1. Cost-Effective: Arduino embedded devices offer a cost-effective solution in comparison to other microcontroller options.

2. Cross-Platform Compatibility: Arduino is versatile, and hence operates seamlessly on Windows, Linux etc. unlike many microcontroller systems limited to specific platforms.

3. User-Friendly Programming Environment: The Arduino programming environment is designed to be straightforward and easily understandable, making it particularly accessible for beginners.

4. Open Source: means it is published with accessible source code. This open nature provides users with readily available information and insights from experienced programmers.

2.5 MATLAB Simulink software

MATLAB, short for MATrix LABoratory, is a high-level programming language and computing environment developed by MathWorks. It is widely used in various fields, including engineering, for numerical computation, data analysis, algorithm development, and visualization.

MATLAB is used for several reasons:

1. **Numerical Analysis and Simulation:** MATLAB provides a rich set of mathematical functions and tools that are crucial for numerical analysis. Electrical engineers can use MATLAB to model and simulate complex electrical systems, analyze the behaviour of circuits, and solve equations involving electrical parameters.
2. **Signal Processing:** MATLAB includes specialized toolboxes for signal processing, making it an invaluable tool for electrical engineers working with signals and systems. It facilitates tasks such as filtering, spectral analysis, and the design of digital signal processing algorithms.
3. **Control System Design:** MATLAB is widely employed in the design and analysis of control systems. Engineers can model and simulate control systems, analyze their stability, and design controllers using MATLAB's control system toolbox.

4. **Image Processing:** For electrical engineers working with image data, MATLAB's image processing toolbox offers a range of functions for tasks such as image enhancement, segmentation, and feature extraction.
5. **Communication Systems:** MATLAB provides tools for designing and simulating communication systems. Engineers can model various modulation schemes, analyze the performance of communication systems, and design error correction codes.
6. **Machine Learning and Data Analysis**: With the increasing use of machine learning in engineering applications, MATLAB's machine learning toolbox allows electrical engineers to implement and train models for tasks such as pattern recognition and predictive analytics.
7. **Graphical Visualization:** MATLAB's visualization capabilities are crucial for engineers to analyze and present their data effectively. Engineers can create 2D and 3D plots, graphs, and other visual representations to better understand their results.
8. **Efficient Programming:** MATLAB's syntax is designed to be user-friendly and intuitive, enabling engineers to quickly prototype and implement algorithms. This efficiency is especially important in the iterative process of designing and refining electrical systems.

In summary, MATLAB is widely used by electrical engineers because it provides a comprehensive set of tools for numerical computation, simulation, and data analysis, making it a versatile platform for tackling a broad spectrum of challenges in the field of electrical engineering.

METHODOLOGY

3.1 Dynamic Model of a DC Motor

This part presents the dynamic model of closed loop armature-controlled DC motor speed control system. The simulation of armature-controlled DC motor has been performed on Simulink/MATLAB software.

Table 3.1: Specification of DC Motor

Parameters	Values and Units
Rated Power (P)	5 HP
Rated Armature Voltage	240 V
Armature Resistance (R_a)	2.518 Ω
Armature Inductance (L_a)	0.028 H
Field Resistance (R_f)	281.3 Ω
Field Inductance (L_f)	156 H
Back EMF Constant (K_b)	0.0924
Motor Constant (K_t)	0.0924
Friction coefficient of motor (B)	0.0005 Nm*s/rad
Moment of inertia of motor (J)	0.003 kg-m^2
Rated Speed	500 rpm
Rated Field voltage	300 V

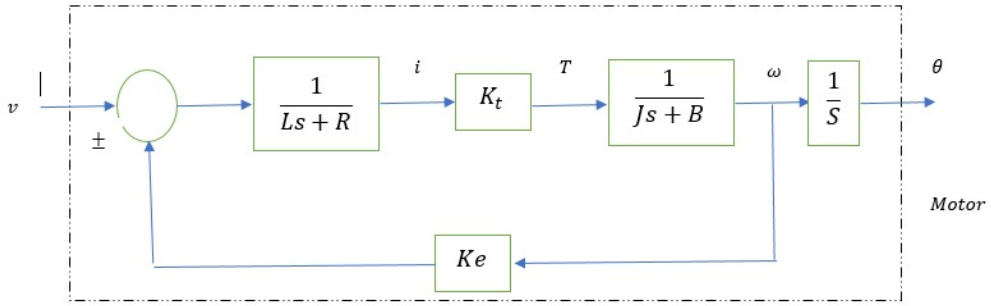


Fig. 3.1: Block diagram of Armature controlled DC Motor

Overall transfer function of the given DC Motor is,

$$G(s) = \frac{0.0924}{8.49\times 10^{-7}s^2 + 0.00585s + 0.01729} \qquad ...(13)$$

Above equation shows the s-domain model of a DC Motor but to analyze a motor using Arduino we need the discretised model.

Discretization involves the transformation of a continuous system, described by functions or equations that change smoothly over time, into a discrete system. This process is essential for platforms like Arduino, which operate on discrete-time systems, necessitating mathematical models suitable for numerical implementation and assessment. Control functions, depicting the relationship between inputs and outputs in continuous time, are adapted through discretization to enable their effective representation and execution on digital platforms like Arduino.

The bilinear transform is defined as:

$$s = \frac{a(z-1)}{(z+1)}$$

Where a is usually given as 2/T and T is Sampling time. We are taking sampling time = 0.2 sec.

On using the aforementioned transformation on equation 13, we get,

$$\frac{1}{Ls+B} = \frac{z+1}{2.546z+2.49}$$

$$\frac{1}{Js+B} = \frac{z+1}{0.02z+0.01}$$

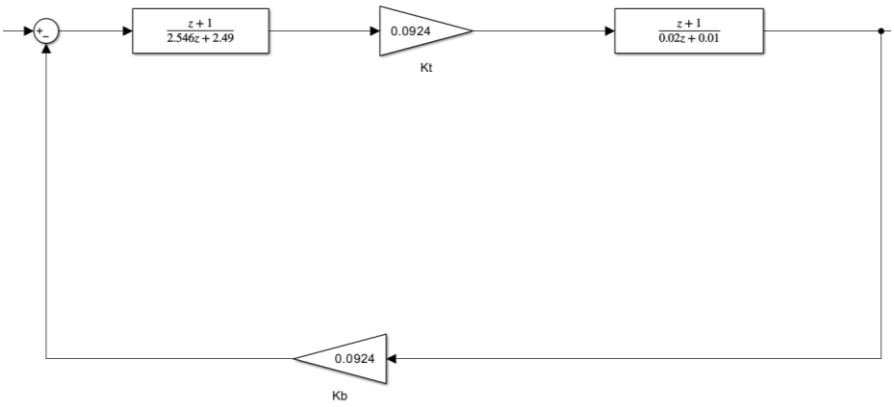


Fig. 3.2: Discrete model of given DC Motor

3.2 Tuning of PID Control Parameters

For the tuning of PID controller Ziegler–Nichols method is used. In this method initially integral and differential gains are kept zero and then the proportional gain is increased until the system is unstable. The value of K_p at the point of instability is called K_{max} and the frequency of oscillation is f_0[4].

From the open loop step response of system following set points are noted –

$$t_a = 1 \text{ second},$$

$$t_b = 1.306 \text{ second},$$

and DC Gain is, $\beta = 5.344$

therefore, $T = 0.67(t_b - t_a) = 0.205,$

$$\propto = 1.3t_a - 0.29t_b = 0.920$$

The values of K_p, K_i and K_d are given by to the formulas –

$$K_p = 1.2\frac{T}{\beta\alpha} = 0.05,$$

$$K_i = \frac{k_p}{2\propto} = 0.009,$$

But by trial and error we get final value of $K_i = 0.9$

$$K_d = \frac{K_p \propto}{2} = 0.069,$$

This method provides a starting point for PID tuning, and therefore after further adjustments based on the specific characteristics and requirements of the control system the final values of K_p, K_i and K_d are determined. Hence the final values of K_p, K_i and K_d by manual tuning are 0.05, 0.009 and 0.069 respectively.

3.3 PID Controller design in Arduino

As PID Controller uses the current error (Proportional), its integral and derivative to generate an output. Implementation of this controller aims to manage the speed of a DC Motor using Arduino. Therefore, to achieve this we'll employ the discrete version of the eq.(11). This involves replacing the integral with the summation and rewriting the error variation in a discrete way as follows:

P: Instantaneous Error = setpoint – input;

I: Cumulative Error = error + error * elapsedTime;

D: Rate of Error = (error – errorLastCalculated)/elapsedTime;

To achieve the desired output, multiply each term by its associated K value and then combine them. The proportional component works to reduce the rise time, the integral component eliminates steady-state error, and the derivative component minimizes overshoots and ringing.

3.4 Analysis of PID Controller –

To check that PID controller is working before connecting it to the DC Motor model, it is tested by giving an array of random speed values and then calculating the output value, for that we made some changes in the existing code, and instead of giving an actual input to the Arduino block an array consisting of random speed values is given as input and then the behaviour of controller is studied.

Code Implementation:

```
float dt = 0.1;
float integral = 0;
float prev_error = 0;
float output = 0;

float kp = 0.05;
float ki = 0.9;
float kd = 0.069;

float desiredSpeed = 400;
float actualSpeed = 0;

int motorInputs[] = {368,436,389,418,423,398,426,372,438,360};
int currentIndex = 0;

void setup() {
  Serial.begin(9600);
}

void loop() {

  actualSpeed = motorInputs[currentIndex];


  float error = desiredSpeed - actualSpeed;


  output = pid(error);




  Serial.print("Desired Speed: ");
  Serial.print(desiredSpeed);
  Serial.print(" RPM, Actual Speed: ");
  Serial.print(actualSpeed);
  Serial.print(" RPM, Output: ");
  Serial.println(output);


  currentIndex++;


  if (currentIndex >= sizeof(motorInputs) / sizeof(motorInputs[0])) {
    currentIndex = 0;
  }

  delay(100);
}

float pid(float error) {
  float proportional = error;
  integral = integral + error * dt;
  float derivative = (error - prev_error) / dt;
  prev_error = error;

  float pidOutput = (kp * proportional) + (ki * integral) + (kd * derivative);


  pidOutput = max(min(pidOutput, -255), 255);

  return pidOutput;
}
```

Explanation:

1. **PID Controller Parameters:**
   K_p (Proportional Gain): 0.05
   K_i (Integral Gain): 0.9

K_d (Derivative Gain): 0.069

2. **Speed Control Parameters:**
   desiredSpeed: The desired speed of the motor (set to 400 RPM)
   actualSpeed: The current speed of the motor (initialized to 0 RPM)
3. **Motor Input Array:**
   motorInputs: An array representing the motor speed values at different time steps.
4. **Control Loop in loop() Function:**
   The loop reads the current motor speed from the array (actualSpeed).
   Calculates the error (desiredSpeed - actualSpeed).
   Calls the pid() function to compute the PID output (output).
   Prints the desired speed, actual speed, and PID output to the Serial Monitor.
   Updates the array index for the next iteration.
5. **PID Controller in pid() Function:**
   Calculates the proportional, integral, and derivative components of the PID controller.
   Sums up these components with their respective gains to get the PID output.
   Constrains the PID output to a specified range (-255 to 255 in this case).
   Returns the computed PID output.
6. **Serial Communication:** The Serial Monitor is used to display information about the desired speed, actual speed, and PID output.
7. **Delay:** There's a delay of 100 milliseconds at the end of each loop iteration.

In order to ensure that the controller works efficiently with a wide range of set speed and actual speeds, the output of controller is analysed for different set speed values like 200,300,400,450 and 500.

Also, to verify the action of controller we used the same code in MATLAB to create a functional block and analyzed its response for various values of set speed in similar manner and actual speed as input.

Then the behaviour of both the controller(in Arduino UNO and MATLAB) is compared in order to analyze that the control output is same or not.

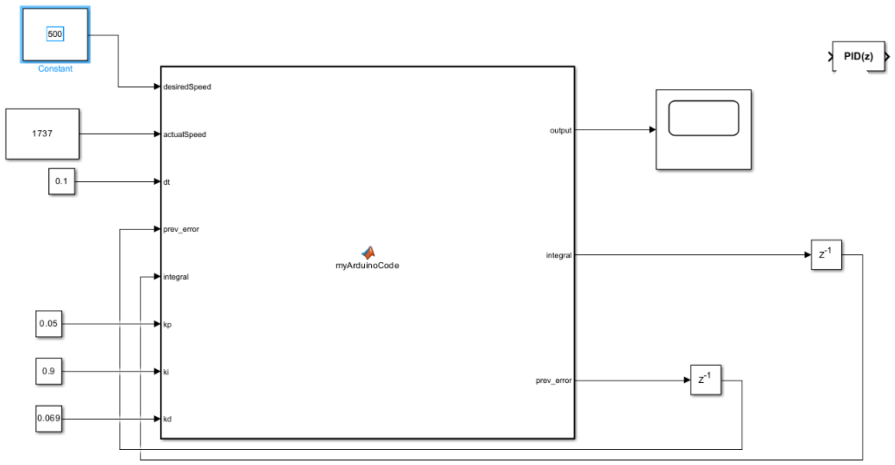


Fig 3.3: MATLAB implementation of PID controller

Comparison of responses from Arduino and MATLAB:

For the set speed of 400 rpm the response of Arduino is as follows:

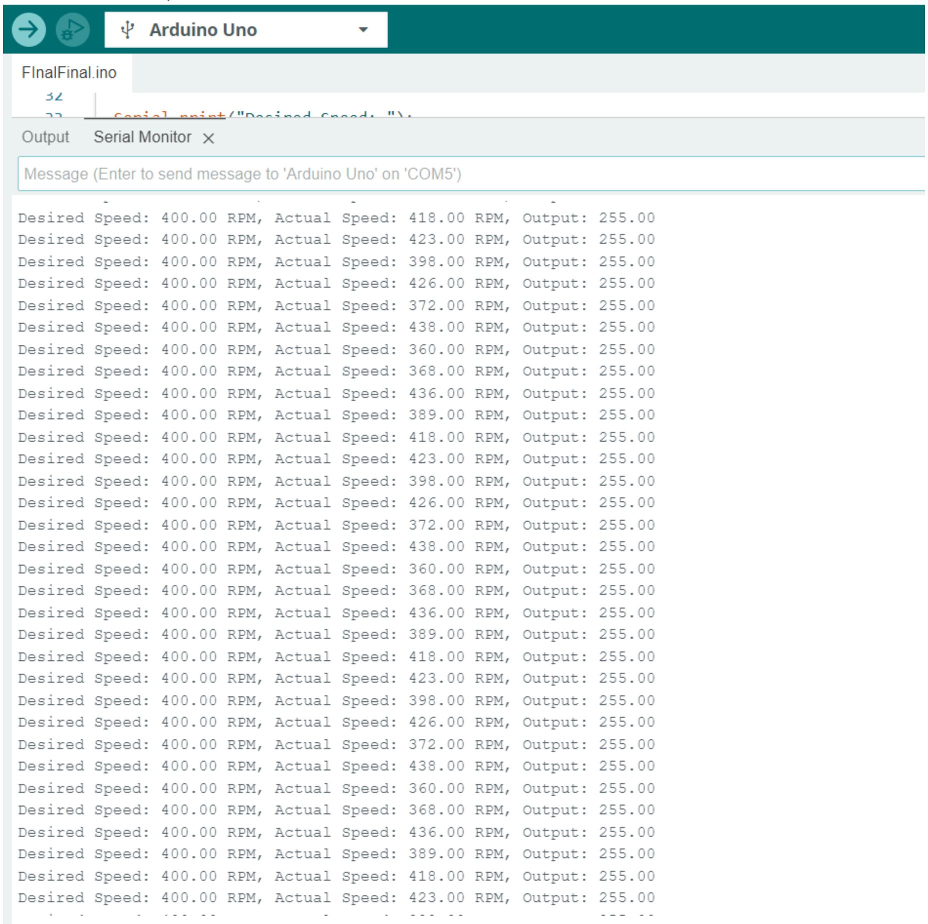


Fig. 3.4: Serial Monitor's snapshot from Arduino for set speed of 400 rpm

For the set speed of 400 rpm the response of MATLAB PID Controller block is as follows:

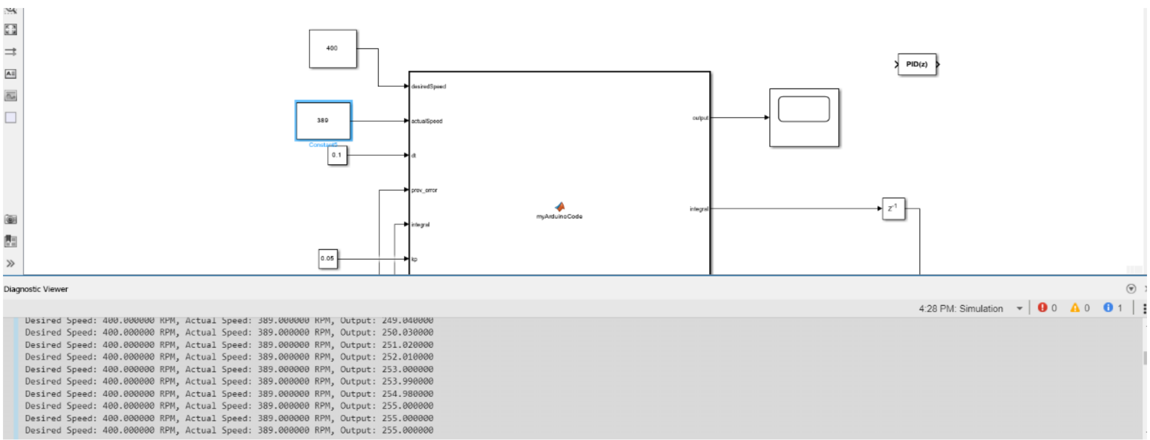


Fig. 3.5: MATLAB PID Controller's response

3.4 Block diagram of the Project –

By joining both the PID Controller and DC Motor Model together we get the final system as:

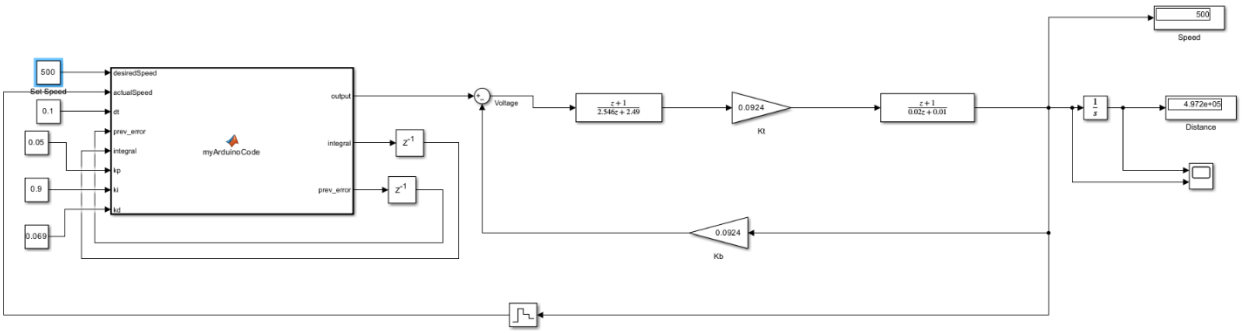
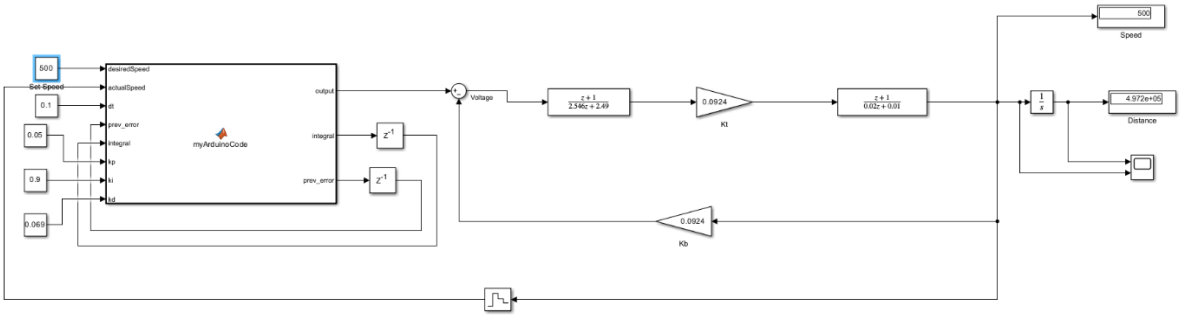


Fig. 3.6: Block diagram of the project

after integration with the DC Motor model system behaviour is observed carefully for the same values of set speed. The closed loop response for various values of set speed is given below:



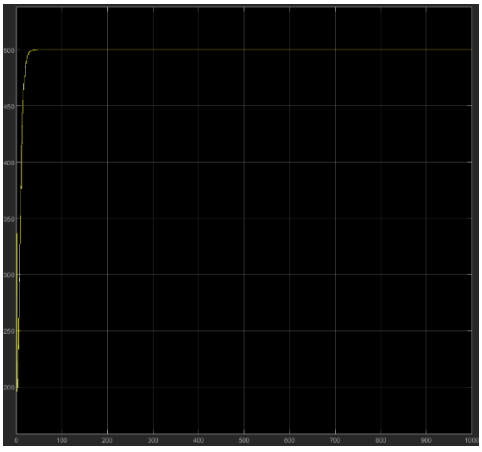
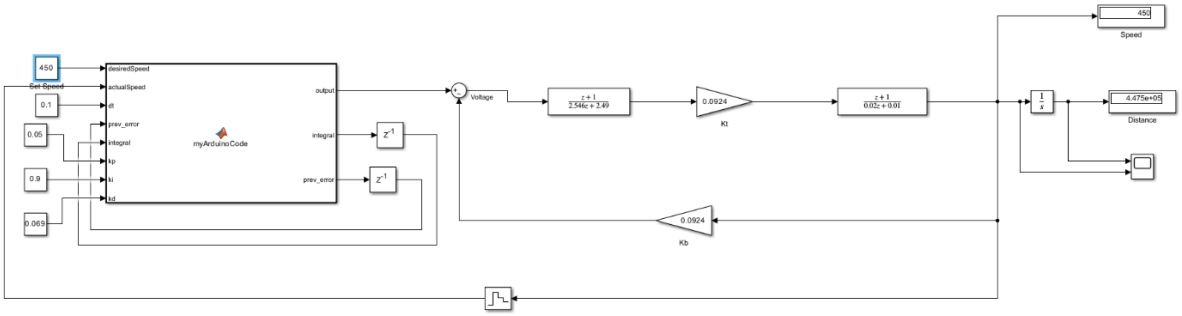


Fig. 3.7: Closed loop response of the system when desired speed is set to 500 RPM



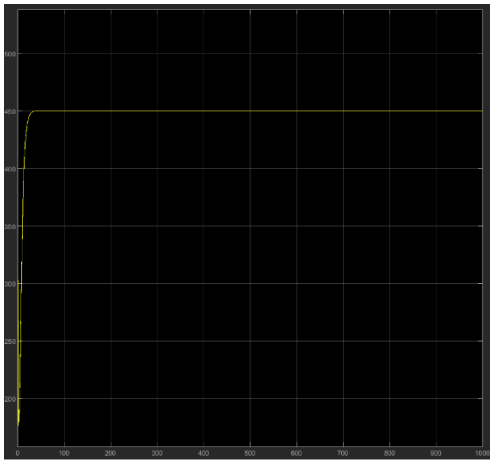
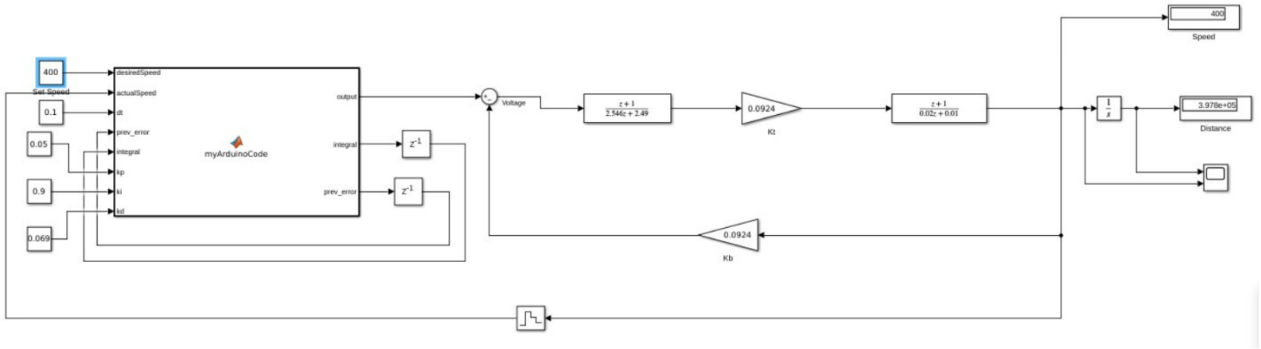


Fig. 3.8: Closed loop response of the system when desired speed is set to 450 RPM



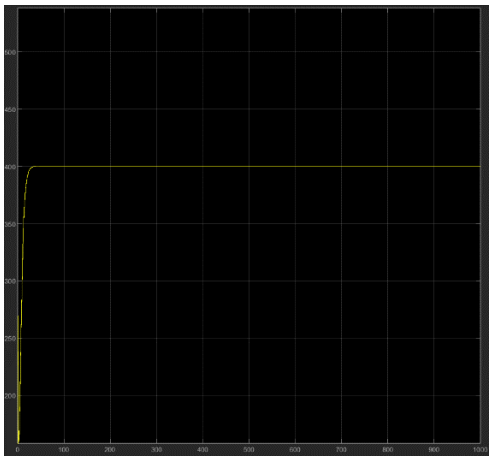
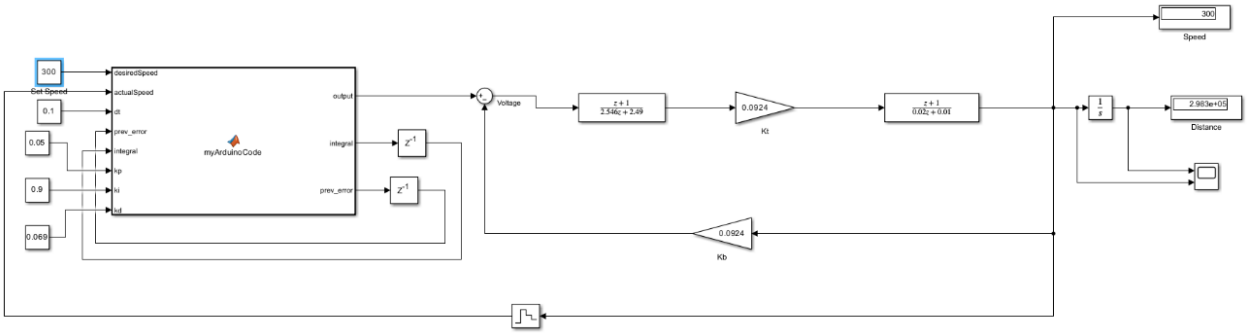


Fig.3.9: Closed loop response of the system when desired speed is set at 400 RPM



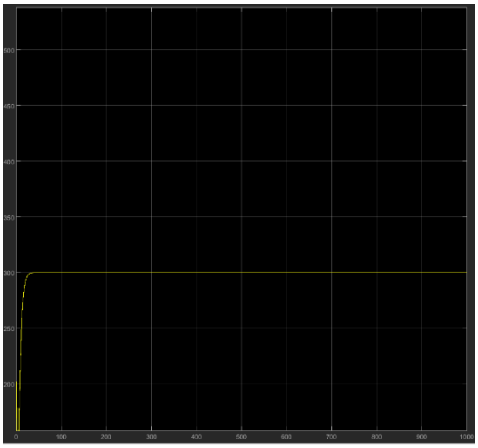
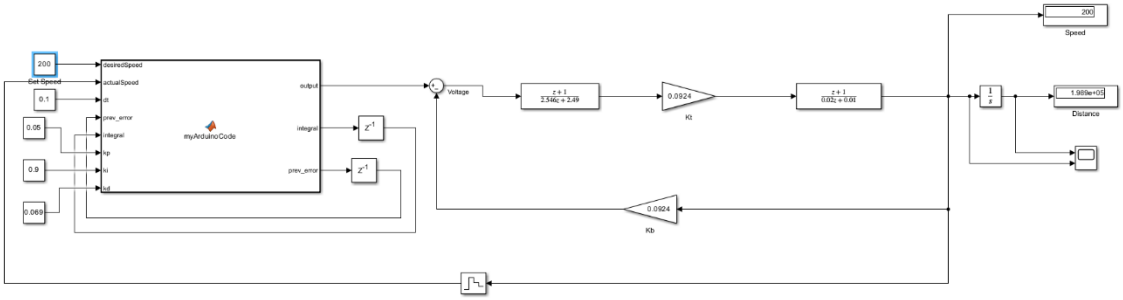


Fig. 3.10: Closed loop response of the system when desired speed is set to 300 RPM



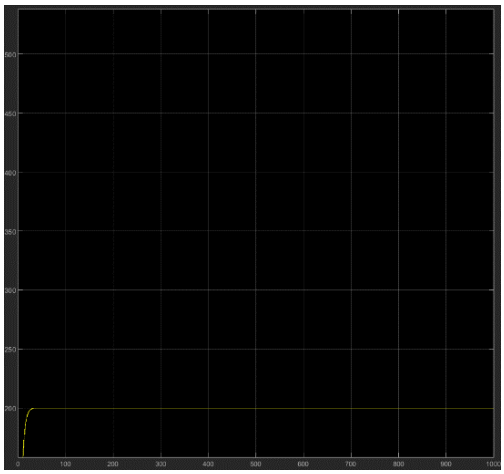


Fig. 3.11: Closed loop response of the system when desired speed is set to 200 RPM

RESULTS AND CONCLUSION

The experimental investigation into DC motor speed control utilizing a proportional-integral-derivative (PID) controller on the Arduino platform has yielded comprehensive and encouraging results. The primary objective of this study was to achieve precise and stable speed regulation across a broad range of desired speeds, and the PID controller, implemented through the Arduino microcontroller, emerged as a highly effective solution. The PID controller effectively minimized steady-state error, enabling the motor to reach and maintain the setpoint with minimal oscillations.

Throughout the process, the system consistently demonstrated an impressive ability to respond promptly to changes in the setpoint. This responsiveness was crucial, especially in applications where quick and precise speed changes are required.

In conclusion, the use of a PID controller for DC motor speed control in conjunction with Arduino proved to be a robust and efficient solution. The proportional, integral, and derivative terms collectively contributed to a well-balanced control system, addressing both transient and steady-state responses. This project highlights the versatility and effectiveness of Arduino in implementing advanced control algorithms for real-world applications, paving the way for further enhancements in motor control systems.

REFERENCES

[1] S. Balmurugan, A. Umarani, “Study of discrete PID Controller for DC Motor speed control using MATLAB”, Vol. 1, Department of electronics and Instrumentation Engineering, K.L.N. College of Engineering, Tamil Nadu, 2020.

[2] N. B. Berahim, “Development of PID Voltage Control for DC Motor using Arduino”, Universiti Tun Hussein Onn Malaysia, 2014.

[3] A. P. Singh, “Speed Control of DC Motor using Pid Controller Based on MATLAB”, Vol. 4, 2013.

[3] “PID controller implementation using Arduino”. microcontrollerslab.com. [Online]. Available: https://microcontrollerslab.com/pid-controller-implementation-using-arduino/.

[4] “Ziegler Nichols Method”. ScienceDirect.com. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/ziegler-nichols-method.

[5] M. M. Sabir, J. A. Khan, “Optimal Design of PID Controller for the Speed Control of DC Motor by Using Metaheuristic Techniques”. [Online]. Available: https://www.hindawi.com/journals/aans/2014/126317/.

[6] “PID Controller Basics & Tutorial: PID Implementation in Arduino”. Arrow.com. [Online]. Available: https://www.arrow.com/en/research-and-events/articles/pid-controller-basics-and-tutorial-pid-implementation-in-arduino.

[7] R. Singhal, S. Padhee, G. Kaur, “Design of Fractional Order PID Controller for Speed Control of DC Motor”, 2012.

[8] M. S. Microcontroller, "Low-cost embedded solution for PID controllers of DC

motors", no. 15, pp. 1178–1183, 2009.

[9] K. H. Ang, G. Chong, S. Member, and Y. Li, "PID Control System Analysis, Design, and Technology", vol. 13, no. 4, pp. 559–576, 2005.

[10] G. Huang and S. Lee, "PC-based PID speed control in DC motor", 2008 Int. Conf.

Audio, Lang. Image Process., pp. 400–407, Jul. 2008.

[11] F. Directions, "PID Control System Analysis and Design", no. February 2006.

[12] H. Faugel and V. Bobkov, "Open source hard- and software: Using Arduino boards

to keep old hardware running", Fusion Eng. Des., vol. 88, no. 6–8, pp. 1276–1279,

Oct. 2013.