NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

Final Year Project
Report

Design & Development of a Re-Configurable NCS for Feedback Control of Diverse Processes Distributed Over an IP Network

Project Advisor	Cdr. Dr. Attaullah Memon		
Co-Advisor	Dr. Sameer Qazi		
Project Members	Adeel Jamal	EE-725	NUST-03
	Ali Ahmed	EE-724	NUST-04
	Nabeel Ahmad	EE-749	NUST-18

Pakistan Navy Engineering College
Faculty of Electronics and Power Engineering

ORIGINAL WORK DECLARATION

I hereby declare that this work has been done by my project's team and this is to certify that this UG project work titled "Design & Development of a Re-Configurable NCS for Feedback Control of Diverse Processes Distributed Over an IP Network" submitted by Mr. Adeel Jamal, Mr. Ali Ahmed, and Mr. Nabeel Ahmad is a bonafide project work carried out under my supervision and guidance and fulfilling the nature and standard required for the partial fulfillment of the degree of Bachelor of Engineering (EE). The Project work (theme and its documentation) embodied in this Project has not been copied (without any reference) from elsewhere. It is also declared that manufacturing been undertaken solely by the Group with minimal outside assistance.

_____ _____

Project advisor
Cdr. Dr. Attaullah Memon

Project co-advisor
Dr. Sameer Qazi

_____

The Dean EPE

Date: _____

ABSTRACT

The use of computer networks in a control loop has gained quite popularity in recent years due to its flexibility and cost effectiveness. Networking is widely used in the control systems of military and industrial applications such as automobiles, aircrafts and manufacturing plants.

In this project, we have built a practical setup of a Networked Control System (NCS) for distributed processes. Multiple plants can be operated in closed loop via a single centralized PID controller. What distinguishes this control system from ordinary control systems is the presence of a computer network between the controller and the plant.

Thorough simulations have been run on MATLAB after the measurement of model parameters. Controller gains are chosen after analysis of these simulations. The simulation results are also compare to experimental results.

The use of the college LAN allows for the closed loop control of multiple plants that are distributed at large distances from the main controller. This effectively reduces the wiring cost and allows greater flexibility.

The main PID controller is implemented in MATLAB. The communication over the network is done using UDP datagrams. The setup includes a dedicated development board for each plant which contains an Ethernet Shield to allow for network communication.

However, an important factor that needed to be accounted for was the presence of delays in the network. The network delays can cause the output of the plant to become unstable which can severely affect the performance of the plant and the system as a whole.

The approach, we have adopted, to cater to the adverse effects of delays involves first measuring the RTT (Round Trip Time) between the system and the plant, and then

readjusting the PID gains at runtime in such a way that the response characteristics remain within the required limits.

Our primary findings in this project were the deteriorating effects of network delays on the performance of a closed loop control system and techniques to cater to the problem of network delays to get the possible response in current conditions.

ACKNOWLEDGMENTS

We would like to express our wholesome gratitude to our Advisor Cdr. Dr. Attaullah Memon and our Co-Advisor Dr. Sameer Qazi for all the technical help and suggestions, support and motivation throughout the project titled *Design and Development of a Re-configurable Networked Control System for Feedback control of Diverse Processes distributed over an IP Network*.

We would like to express special thanks to OIC EPE Projects Lab Lt. Cdr. Mustafa Jan for the important feedback and information related to the projects.

We would also like to thank A/P Farhan Khan, A/P Ashraf Yahya, A/P Nusrat Hussain and OIC PCB Lab Sir Mukhtar for their valuable time and helpful suggestions.

Lastly, we would like to thank the staff of the ES workshop, particularly Mr. Abdul Majeed, who assisted us with mechanical structures of the plant side.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 PROBLEM DESCRIPTION

This project involves a practical setup for control of systems over networks. In this project, we control a plant connected to a controller via a computer network. This allows for long distance control and also increases flexibility in installation.

However, the delays in the network can cause trouble in the system. Delays hinder smooth communication between the controller and the plants can cause the system to become unstable.

To solve the problem of network delays, the PID controller is designed in such a way that it tunes its parameters to match that of a stable system without delays.

This ability of the re-configurable controller can ensure that the system does not lose its stability and the system performance is as good as that of a system without network.

1.2 PROJECT FEATURES

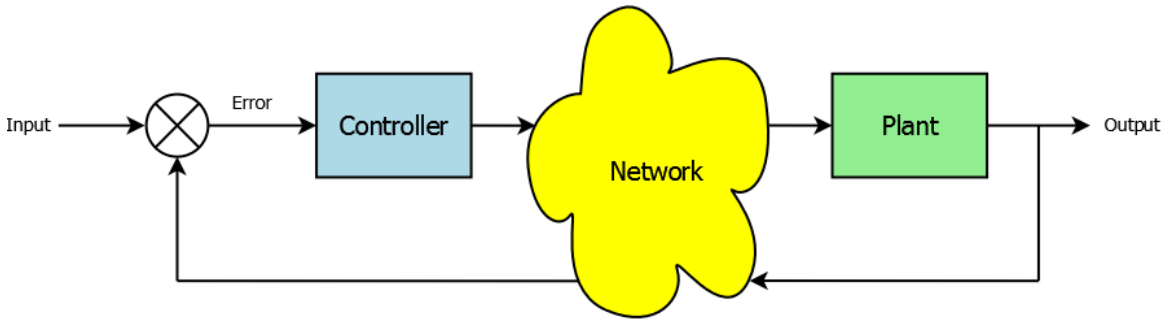


Figure 1 A closed loop control system

i. Our project's main theme is to build a networked control system with a single centralized controller that is capable of controlling plants distributed throughout a Local Area Network.

ii. The main plant is a DC motor speed control plant. This is the main scope of our work.

    As an extra effort, we are also working on two extra plants on which the work is mostly finished. These two plants are:

    - Position control of a DC motor
    - Angle control of an aero-pendulum

iii. The system on the plant side consists of a mechanical plant, a sensor, an actuator, and a custom-made DAQ board.

iv. The controller is centralized and implemented in MATLAB.

v. The presence of a network means presence of delays in the network. These delays can negatively affect the performance of the plants and make them unstable.

vi. The controller is re-configurable and able to minimize the adverse effects on network delays by adjusting its PID gains.

1.3 OVERVIEW

Now we look at the system from the perspective of a single plant.

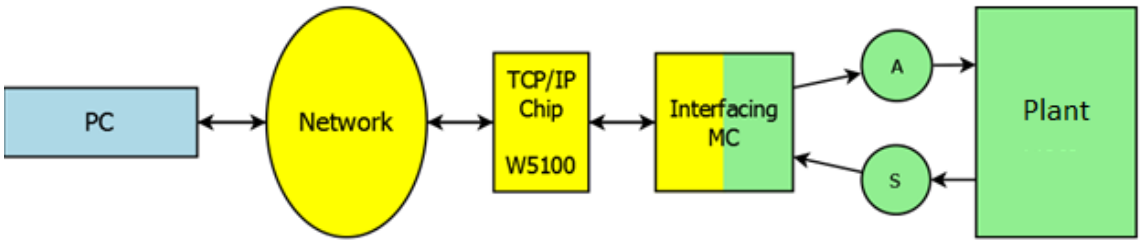


Figure 2 A more detailed view of the system

Each of the shown system components are discussed below:

1.3.1 PC (Controller)

The main controller is implemented in MATLAB. It is a discrete PI controller and is capable of changing its parameters to not only ensure system stability but also to achieve desired transient response during the presence of network delays. It is re-configurable and is capable of controlling multiple plants distributed over a network.

1.3.2 Network

The network is the communication medium between the controller and the plant. The common TCP/IP network has been used. UDP datagrams are used for communication between the controller and the plants.

1.3.3 TCP/IP chip and Interfacing MC board

The TCP/IP chip that provides network connectivity is embedded in the Ethernet Shield. The Ethernet Shield is responsible for de-encapsulating the network packet into useful information and passing it onto the microcontroller for further processing.

1.3.4 A and S

The function of the actuator (A) is to apply the signal coming from the controller. The function of the sensor (S) is to measure the current state of the project which is then sent to the controller. These devices are discussed in more detail in Chapter 3 of this document.

1.3.5 Plant

The Plant can be any physical system. We have chosen a PMDC motor as the main plant.

2 LITERATURE REVIEW

2.1 CONTROL SYSTEMS

A simple control system consists of a Controller, sensors, actuators and a plant. The actuators receive the signal from the Controller and in turn, drive the plant. The sensor measures the parameter being monitored and transmits the signal back to the Controller via a Feedback loop. The Controller compares the Feedback signal from the sensor with the set point and varies its output signal accordingly.

This is in contrast to the Open Loop system in which the Feedback loop is absent. Such a system, despite its simplicity, is highly vulnerable to disturbances.

A Closed loop system is shown below.

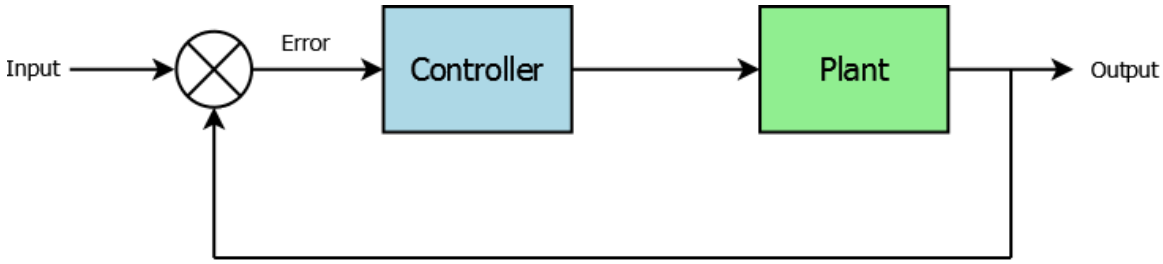


Figure 3 A closed loop control system - repeated

A closed-loop system, then, has the advantage of less sensitivity to noise and disturbances and is more accurate.

A simplistic view of a Networked Control System is given in the following diagram:

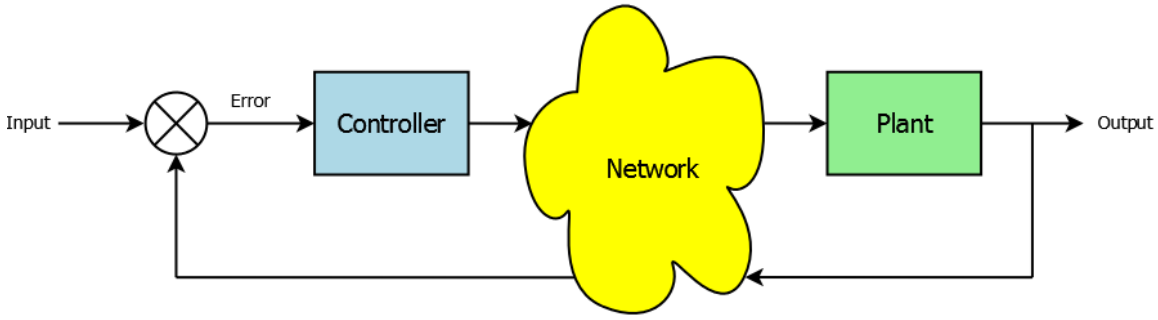


Figure 4 A simple diagram that depicts the idea of NCS

In an ordinary control system with feedback, the Controller and the actuators, sensors and the plant are connected physically through wires. While this may work for simple systems, this approach is definitely not practical for larger systems.

2.2 DISTRIBUTED CONTROL SYSTEMS

In real applications, there exist systems distributed over a large area with plants located in different places, far from the main controller. Here, communication has to be established between the controller and the plant.

An example of a distributed control system is shown on next page.

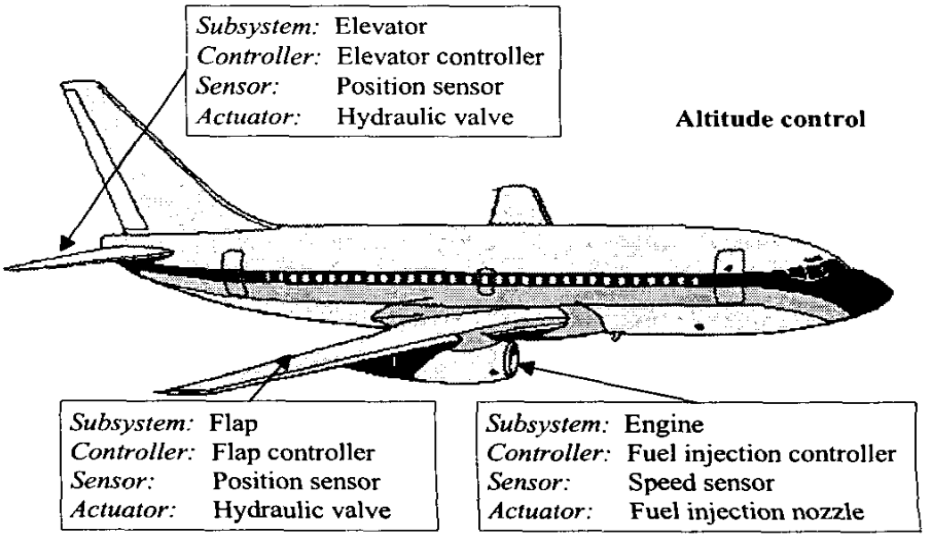


Figure 5 Distributed control systems on an aeroplane

The use of hardwired connections in such a case would do no good but add to the complexity of the system. Such systems are difficult to maintain and install.

As a solution, the Controller can be connected to the actuators, sensors and plant through a network as shown.

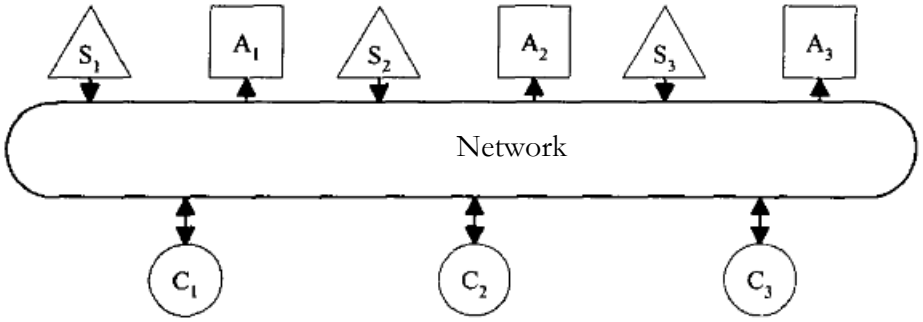
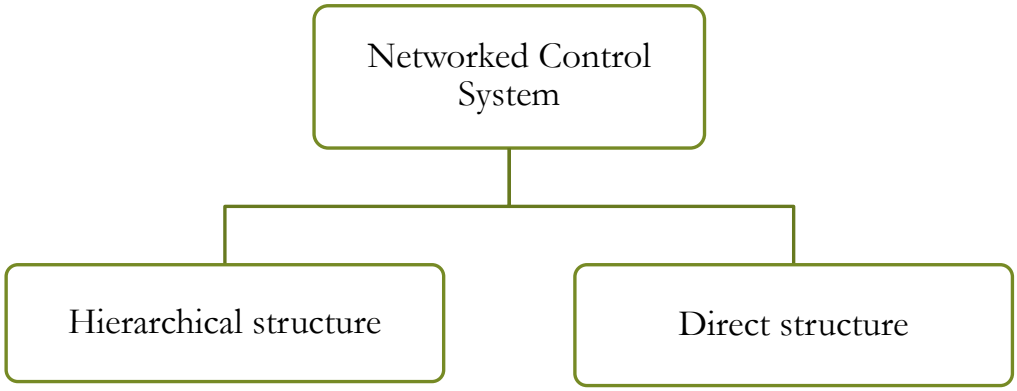


Figure 6 Topology of a distributed control system over a network

Here the control loops are closed through a real-time network. The defining feature of such a system is that control and feedback signals are exchanged among the system's components in the form of information packages through a network.

Such a system is called a Networked Control System, abbreviated as NCS.

2.3 NCS CONFIGURATIONS



2.3.1 Hierarchical structure

In this structure, there are two controllers: the main Controller and the subsystem Controller. The main Controller sends the signal through a network to the subsystem Controller. The subsystem Controller drives the Actuator and the plant. The sensor responds the subsystem Controller which transmits the signal back to the main Controller.

The Hierarchical structure is shown below:

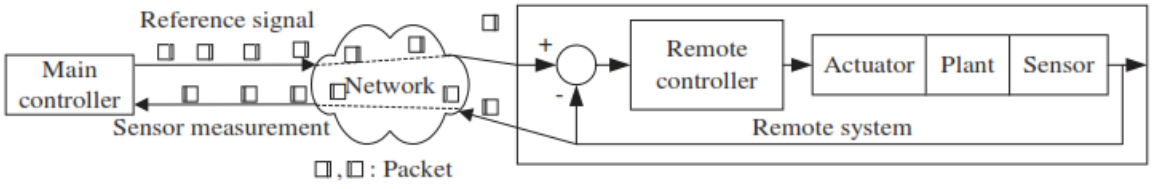


Figure 7 NCS in the hierarchical structure

2.3.2 Direct structure

Here, there is only one controller, the main Controller. The sensors and actuators are connected directly to the main Controller via a network. There is better interaction between system components and response is fast.

The Direct structure is shown below:

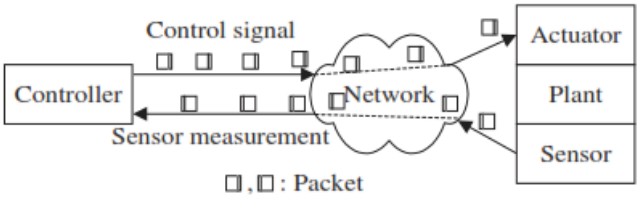


Figure 8 NCS in the direct structure

2.4 ADVANTAGES OF NCS

2.4.1 Advantages

NCS is a subject of continuous research because it promises many advantages. Some of them include:

- Execution of several tasks over long distance.
- Reduces system complexity by reducing unnecessary wiring
- Reduces overall cost of designing and implementing control systems
- Can be easily modified or upgraded with no major changes in structure
- Able to make intelligent decisions over large physical spaces through efficient sharing of data
- Great reduction in complexity of connections
- Flexible to install
- Easier to maintain
- Capability of communication among different loops for more sophisticated control

2.4.2 Applications

There are many applications of NCS and they include:

- Space and terrestrial exploration
- Domestic robots
- Hazardous environments
- Aircraft and automobiles
- Factory automation
- Nursing homes
- Remote Diagnostics and troubleshooting
- Tele-operations
- Experimental facilities

2.5 COMPUTER NETWORKS

The control and feedback signals in the system that we have developed travel through a computer network, therefore it is essential to have a thorough understanding of the theory and functionality of networks.

A computer network is a telecommunications network that allows computers to exchange data. In computer networks, networked computing devices pass data to each other along data connections. The connections between nodes are established using cable media like LAN wires. The best-known computer network is the Internet.

2.5.1 Network Layers

The in-depth understanding of Transmission and reception of information over the internet requires basic knowledge of network layers which is kind of stack across which data is passed which adds more valuable information of that layer before it is allowed to travel in the

physical layer of the network. The phenomenon is repeated both in sending and receiving digital information over the network.

The network layer provides the functional and procedural means of transferring variable-length data sequences from a source to a destination host via one or more networks, while maintaining the quality of service functions. The Open Systems Interconnection model (OSI) is a conceptual model that characterizes and standardizes the internal functions of a communication system by partitioning it into abstraction layers. Following are the main network layers briefly described.

2.5.1.1 Layer 1- Physical Layer

It defines the protocol to establish and terminate a connection between two directly connected nodes over a communications medium.

2.5.1.2 Layer 2- Data Link Layer

The data link layer provides a reliable link between two directly connected nodes, by detecting and possibly correcting errors that may occur in the physical layer. At this layer, data packets are encoded and decoded into bits. It furnishes transmission protocol knowledge and management and handles errors in the physical layer, flow control and frame synchronization. The Point-to-Point Protocol (PPP) is an example of a data link layer in the TCP/IP protocol stack.

2.5.1.3 Layer 3: Network Layer

The network layer provides the functional and procedural means of transferring variable length data sequences (called datagrams) from one node to another connected to the same network. A network is a medium to which many nodes can be connected, on which every node has an address and which permits nodes connected to it to transfer messages to other nodes connected to it.

2.5.1.4 Layer 4: Transport Layer

The transport layer provides the reliable sending of data packets between nodes (with addresses) located on a network, providing reliable data transfer services to the upper layers. An example of a transport-layer protocol in the standard Internet protocol stack is TCP, usually built on top of the IP protocol.

Some of the functions offered by the transport layer include:

- Application identification
- Client-side entity identification
- Confirmation that the entire message arrived intact
- Segmentation of data for network transport
- Control of data flow to prevent memory overruns
- Establishment and maintenance of both ends of virtual circuits
- Transmission-error detection
- Realignment of segmented data in the correct order on the receiving side
- Multiplexing or sharing of multiple sessions over a single physical link

2.5.1.5 Layer 5- Application Layer

The application layer is the OSI layer closest to the end user, which means both the OSI application layer and the user interact directly with the software application. This layer interacts with software applications that implement a communicating component. Such application programs fall outside the scope of the OSI model. Application-layer functions typically include identifying communication partners, determining resource availability, and synchronizing communication. When identifying communication partners, the application layer determines the identity and availability of communication partners for an application

with data to transmit. When determining resource availability, the application layer must decide whether sufficient network or the requested communication exists.

2.5.2 UDP

The primary protocol that is used in our system is the User Datagram Protocol (UDP), which is a part of the TCP/IP stack's Transport Layer. The other option was TCP which was discarded. The primary factor of which we have to take care is network delays. We have to send the data over the network in minimum time and TCP protocol implementation while transferring the information takes appreciable time relative to UDP but it certainly guarantees reliability. TCP is connection-oriented protocol. When a file or message is sent it will get delivered unless connections fail. If connection lost, the server will request the lost part. There is no corruption while transferring a message.

Following are the principal characteristics of the UDP protocol:

- **Reliability:** UDP is connectionless protocol. When you a send a data or message, you don't know if it'll get there, it could get lost on the way. There may be corruption while transferring a message.
- **Ordered:** If you send two messages out, you don't know what order they'll arrive in i.e. no ordered
- **Lightweight:** No ordering of messages, no tracking connections, etc. This means it's a lot quicker, and the network card. Operating Systems have to do very little work to translate the data back from the packets.
- **Datagrams:** Packets are sent individually and are guaranteed to be whole if they arrive.

Examples: Domain Name System (DNS UDP port 53), streaming media applications such as IPTV or movies, Voice over IP (VoIP), Trivial File Transfer Protocol (TFTP) and online multiplayer games etc.

2.6 DELAYS IN NCS

The most important problem that is faced in NCS is that of network delays.

2.6.1 Delays in control systems

The delays affect a control system in two ways:

1. Poor system response
2. Decreased stability margin

In the following diagram, the effect of delays on a system's frequency response is shown.

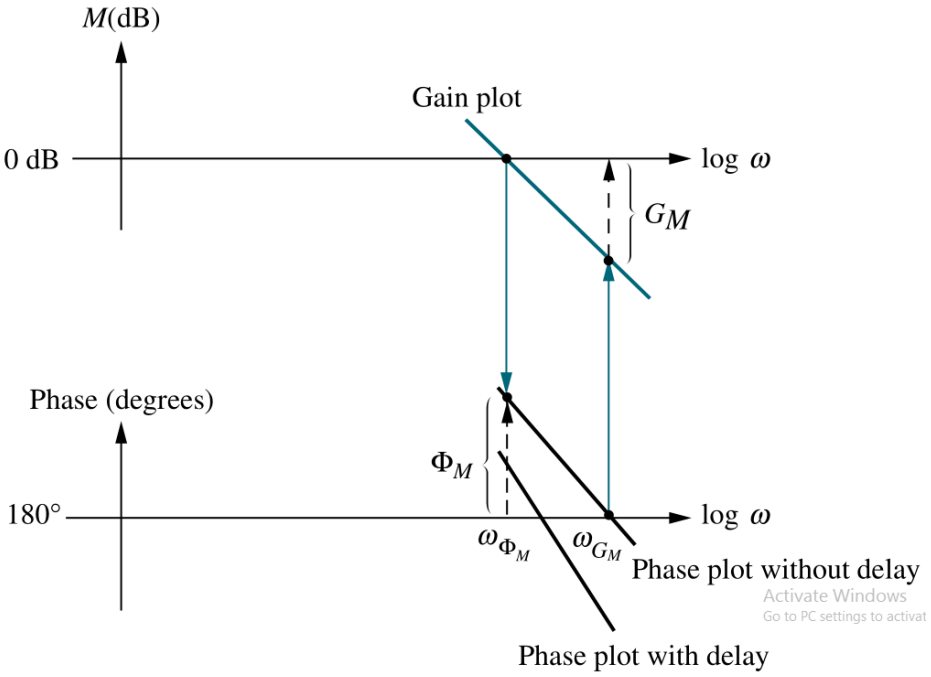


Figure 9 Effect of delays on the frequency response of a control system

The phase margin is related to the damping ratio by the following equation:

$$\emptyset_m = \tan^{-1} \frac{2\zeta^2}{\sqrt{-2\zeta^2 + \sqrt{1 + 4\zeta^4}}}$$

A reduction in phase margin leads to a reduction in the damping ratio, thus, a more oscillatory response is obtained moving the system closer to instability.

2.6.1 Delays in networks

Following types of delays are encountered in networks.

2.6.1.1 Computational delay

It is time-varying but variations are relatively small. It can be reduced and managed using real-time OS and appropriate hardware design techniques. It is normally very short and can be discarded.

2.6.1.2 Transmission delays

Their behavior depends on the network protocol.

In Cyclic Service Networks, such as PROFIBUS, the variation of delays can be periodic and deterministic if certain factors are ignored, such as Communication Errors and Clock Drift.

In Random Access Networks, which include the commonly used Ethernet and CAN networks, transmission delays are stochastic processes. Poisson distribution is sometimes used for the modeling of the delays of the global Internet.

2.6.2 Handling the delays

The problems caused by NCS can be cured by two ways:

1. Improve the network
2. Improve the controller

The focus of research on NCS is on designing various Control techniques that enable the system to maintain performance under the effect of transmission delays caused by the network.

2.6.2.1 Gain Adjustment

One of the simplest, yet most important, techniques for handling delays in NCS is automatic gain adjustment. The idea is to dynamically adjust the gain of the controller based the value of the current delay. In our project, the gains of the PID controller, which is described later in this document, will be dynamically tuned.

2.7 PID CONTROLLER

The general equation of continuous-time PID controller is;

$$u(t) = k(e(t) + \frac{1}{T_i}\int_o^t e(\tau)d(\tau) + T_d\frac{de(t)}{dt})$$

Where the error e(t), the difference between command and plant output, is the controller input, and the control variable u(t) is the controller output. The 3 parameters are K (the proportional gain), T_i (integral time), and T_d (derivative time).

Performing Laplace transform on the above equation, we get

$$G(s) = K(1 + \frac{1}{sT_i} + sT_d)$$

Another form of PID that will be discussed further is sometimes called a parallel form.

$$u(t) = K_p e(t) + K_i\int_0^t e(\tau)d(\tau) + K_d\frac{d}{dt}e(t)$$

For digital implementation, we are more interested in a Z-transform of the above equation

We get;

$$U(Z) = \left[K_p + \frac{K_i}{1 - z^{-1}} + K_d(1 - z^{-1})\right] E(z)$$

Rearranging gives

$$U(z) = \left[\frac{\left(K_p + K_i + K_d\right) + \left(-K_p - 2K_d\right)z^{-1} + K_d z^{-2}}{1 - z^{-1}}\right]$$

Define

$$K_1 = K_p + K_i + K_d$$

$$K_2 = -K_p - 2K_d$$

$$K_3 = K_d$$

The above rearranged equation can then be rewritten as

$$U(z) - z^{-1}U(z) = [K_1 + K_2 z^{-1} + K_3 z^{-2}]E(z)$$

Which then converted back to difference equation as

$$u[k] = u[k-1] + K_1 e[k] + K_2 e[k-1] + K_3 e[k-2]$$

It is a form suitable for implementation.

The following figure shows a PID controller from the perspective of time domain. The model is for continuous time, however the concept remains the same for discrete time control as well.

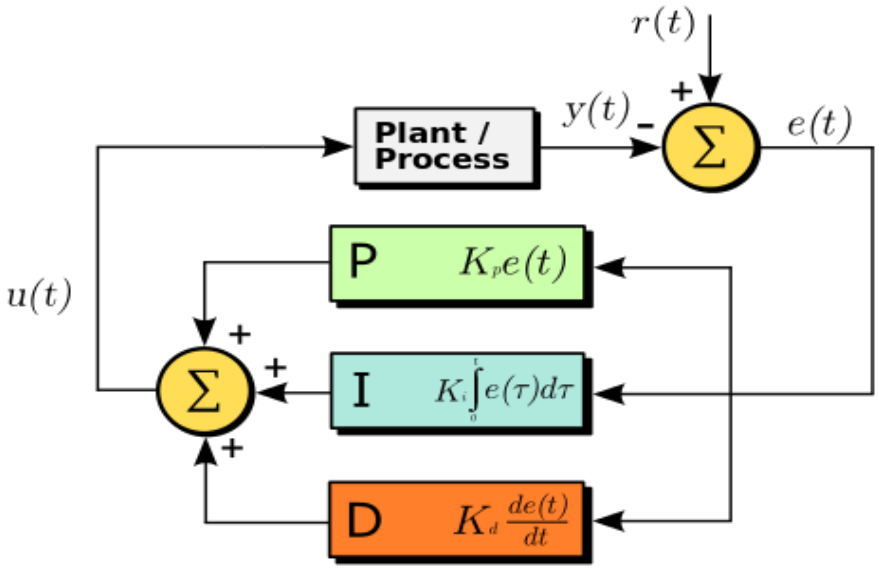


Figure 10 Diagram of a continuous time PID controller

2.7.1 Advantages of Using a PID Controller

There are two main advantage using a PID controller

2.7.1.1 Boosting of Transient Response Performance

The response of the system becomes faster by using the controller & we can set the intensity of rapidness by setting the values of gain of each P, I & D block according to our requirement. The Transient response of a system usually encompasses the following two factors.

- Peak time
- Percentage overshoot
- Settling time

2.7.1.2 Improvement in Steady-State Error

The steady state error of the system may become zero if the values of the constants are set accordingly. The figure below clearly depicts the fact that a PID compensated system's steady state error is the minimum among others.

2.8 H-BRIDGE BASED DC MOTOR DRIVES

An H bridge is an electronic circuit that enables a voltage to be applied across a load in either direction.

The term H Bridge is derived from the typical graphical representation of such a circuit. An H bridge is built with four switches (solid-state or mechanical). When the switches S1 and S4 (according to the first figure) are closed (and S2 and S3 are open) a positive voltage will be applied across the motor. By opening S1 and S4 switches and closing S2 and S3 switches, this voltage is reversed, allowing reverse operation of the motor.

Using the nomenclature above, the switches S1 and S2 should never be closed at the same time, as this would cause a short circuit on the input voltage source. The same applies to the switches S3 and S4. This condition is known as shoot-through.

If the switches S1 and S3 are simultaneously closed, then the phenomenon of Braking would result allowing minimum time for the motor's shaft to settle in the absence of terminal voltage. Similarly, In case of simultaneous closing of S2 and S4 switches.

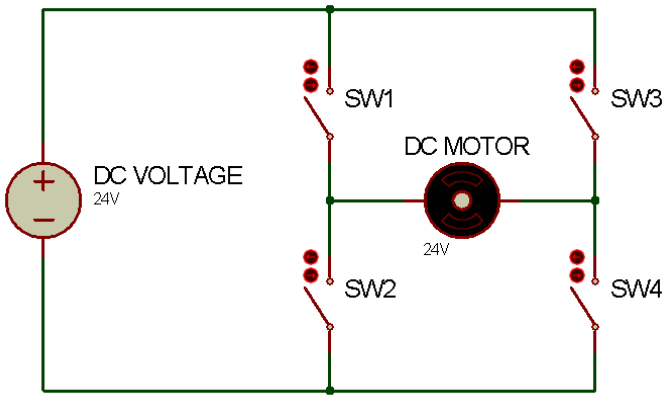


Figure 11 A generic h-bridge

The H-bridge arrangement is generally used to reverse the polarity of the motor, but can also be used to 'brake' the motor, where the motor comes to a sudden stop, as the motor's terminals are shorted, or to let the motor 'free run' to a stop, as the motor is effectively

disconnected from the circuit. The following table summarizes operation, with S1-S4 corresponding to the diagram.

Table 1 Working of an h-bridge

SW1	SW2	SW3	SW4	Direction
ON	*OFF*	*OFF*	*ON*	Clockwise
ON	*OFF*	*ON*	*OFF*	Stops
OFF	*ON*	*ON*	*OFF*	Anti-Clockwise
OFF	*ON*	*OFF*	*ON*	Stops

There are several different designs to practically implement the concept of H-Bridge circuit described above. We mostly used relays if there is a requirement to implement the H-Bridge circuit with mechanical isolating switches otherwise mostly, solid state devices like BJTs and MOSFETs are used which are also popular because of its very less switching time.

We have implemented the H-Bridge drive with the help of two complimentary pair MOSFETs connected across each leg of the H-Bridge circuit. MOSFET driver IC IR2110 is used to meet the current and voltage requirements across gate terminal of each MOSFET because the microcontroller does not have enough current sourcing capability.

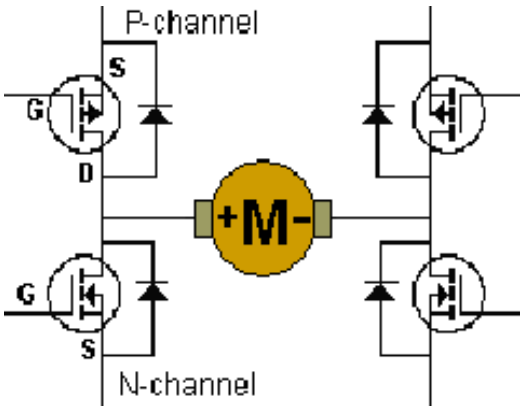


Figure 12 A MOSFET-based h-bridge

The schematic shown below consist of the configuration of MOSFET transistor described above. Here the upper MOSFETs are P-channel and lower two MOSFETs are N-channel. If Vdd and ground is applied across the H-Bridge circuit upper and lower arms respectively, then in order to move the motor in clockwise direction according to the polarity shown, we have to apply at the gate of upper left MOSFET, voltage less than Vdd by a value that is greater than the threshold value of Vgs and negative in magnitude i.e. Vgs must be negative and above the threshold value according to datasheet of transistor to make the MOSFET on. Similar logic can be applied to move the motor in anti-clockwise direction.

3 METHODOLOGY

In this chapter, we will describe each element of our system. These include both hardware and software components. The results that will be presented in the next chapter are made possible by these elements.

3.1 NETWORK

Before we go into the details of the system, we will discuss the details of the network that we have chosen to work on.

We chose to work with the popular TCP/IP protocol. This is the protocol on which the internet works. TCP/IP provides end-to-end connectivity specifying how data should be formatted, addressed, transmitted, routed and received at the destination.

To be specific, we work from the transport layer of the protocol stack. We send UDP datagrams for all the communication between the controller and the plant.

We work on an Ethernet based network. This means that Ethernet acts as the link layer for our network. However, it is still possible to work over Wi-Fi instead of Ethernet, provided that the controller and the plant are not divided by a NAT device.

3.1.1 Network delays

As is discussed in more detail later on, the delays in our main system with the Speed Control plant can only be integral multiples of 100 ms. Moreover, for this system, if the measured delay in the network is less than 5 ms in network, then it is considered as zero delay for the controller. If the measured network delay is in between 5 ms and 105 ms, then it is considered as a delay of 100 ms, and so on. For all the analysis and readings shown in this document, a custom network based on a single router has been used. For this network, the measured delays have been in the order of microseconds, and therefore 0 ms for the controller. Artificial delays have been applied in the MATLAB controller.

For the secondary systems that are discussed in the Supplementary chapter, the sample time is smaller and hence smaller delays can occur. However, their analysis/response is not the topic of discussion in this document.

More about delay measurement is discussed in section 3.2.4.

3.2 CONTROLLER

The controller is PC-based, and uses the MATLAB/Simulink software. It allows us to work on a high-level interface. This has helped us to focus on the controller by allowing for an easy and reliable method to make changes. It has also allowed us to graphically view the results in real-time, and to store these results for analysis later.

Two toolboxes are mainly used for this purpose. They are:

1. Real-time Windows Target (RTWT)
2. Simulink Coder

3.2.1 Real-time Windows Target (RTWT):

The Real-Time Windows Target (RTWT) provides a real-time engine that is central to the function of the controller. It also provides the packet input and output blocks that allow communication with the development board. The RTWT toolbox allows for hardware-in-the-loop simulation. It can provide performance approaching 500 Hz in *normal mode* and 20K Hz in *external mode*. We normally use the external mode.

3.2.2 Simulink Coder

This toolbox converts a Simulink model diagram into C code. It is needed for running RTWT in external mode.

3.2.3 Transfer Function

It has be stated previously that the system (including the Controller) works at a sample rate of 10 Hz. The Transfer Function of the used discrete PI controller is as follows:

$$P + I \cdot T_s \frac{1}{z-1}$$

Where P and I are the Proportional and Integral gains respectively, while T_s is the sample time.

3.2.4 Network delay measurement

One technique of tuning the controller gains according to network conditions is to get data of about network conditions over a long time and then make decision about controller settings based on current time.

Another technique is to measure the delays. There can again be various techniques of measuring the delays. The one that we have used is described next.

We have used the common Ping utility for getting the Round Trip Time (RTT). An m-file calls this application, which outputs a string in return. The m-file then parses this string and stores the delay value in the MATLAB workspace. The Simulink model running in hard real-time reads this variable and acts accordingly. Decision is made based on a weighted average of the measured delays. The m-file gets the delay readings at regular intervals, however it is important to note that it does not run in hard-real time like the Simulink model.

3.2.5 An example model

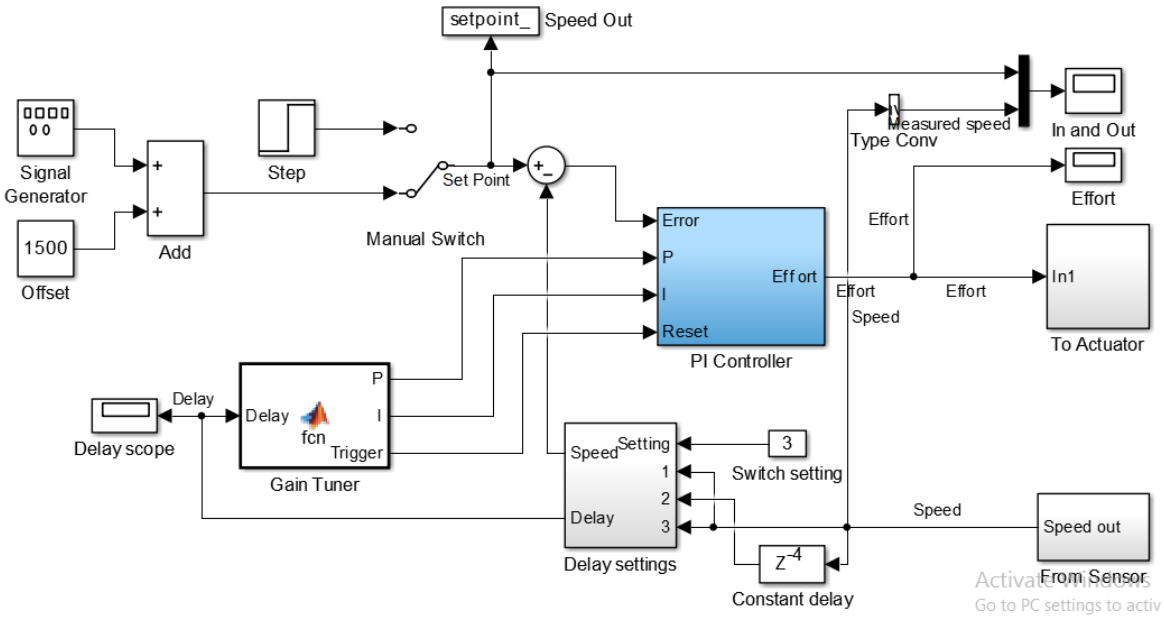


Figure 13 Screenshot of the real-time controller

3.3 PARAMETER ESTIMATION METHOD

In the mathematical and MATLAB models described in the previous two sections, there are many parameters that have been assumed as constant. For proper simulations, these parameters must be measured experimentally using some method. For this purpose, we went by the following method:

1. Record the open-loop real-time response of a system using MATLAB.
2. Get the system parameters by analyzing the recorded data.

For the second step, MATLAB provides a tool called Simulink Design Optimization.

3.4 ANALYSES USING MATLAB

The systems responses were simulated and analyzed quite thoroughly using MATLAB/Simulink. We mostly worked with Simulink models.

The Control System toolbox provided some basic blocks needed for the simulations. Moreover, the following tools/apps were very helpful for our purposes:

3.4.1 PID Tuning

This app was used to adjust gains of our PID controller for different settings of the system. When the system settings (delays in our case) were changed, the system response also changed. This tool allowed for easy adjustment of gains, so that the system response could be improved.

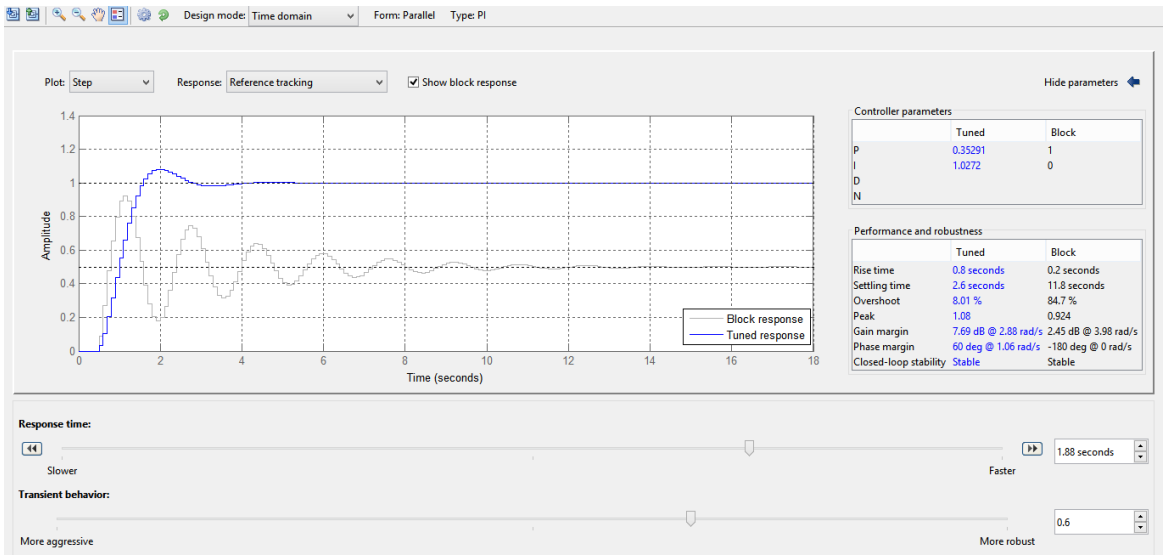


Figure 14 PID Tuning in MATLAB

3.4.2 System Identification

This tool allows the user to input a pre-recorded input/output set for a system. The order of the system is also specified by the user. The tool then estimates a suitable transfer function for the system. It is then easy to study the different characteristics of the system.

Following diagram shows the recorded responses in the left most column. Their transfer functions were then estimated using the tool.

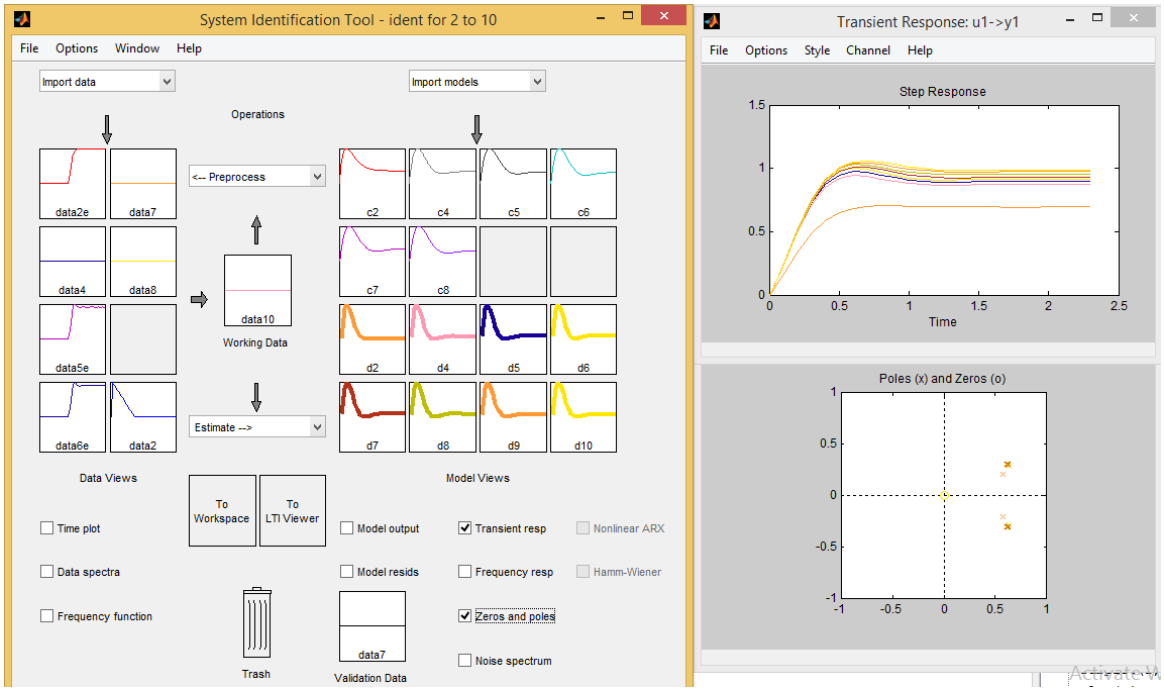


Figure 15 System Identification tool in MATLAB

3.4.3 Linear System Analysis

This tool linearizes any Simulink system. The system's input perturbation and closed-loop output are specified by the user. It provides an easy way to study a control system. It makes the generation of step response, frequency response, and pole-zero plot, relatively easy.

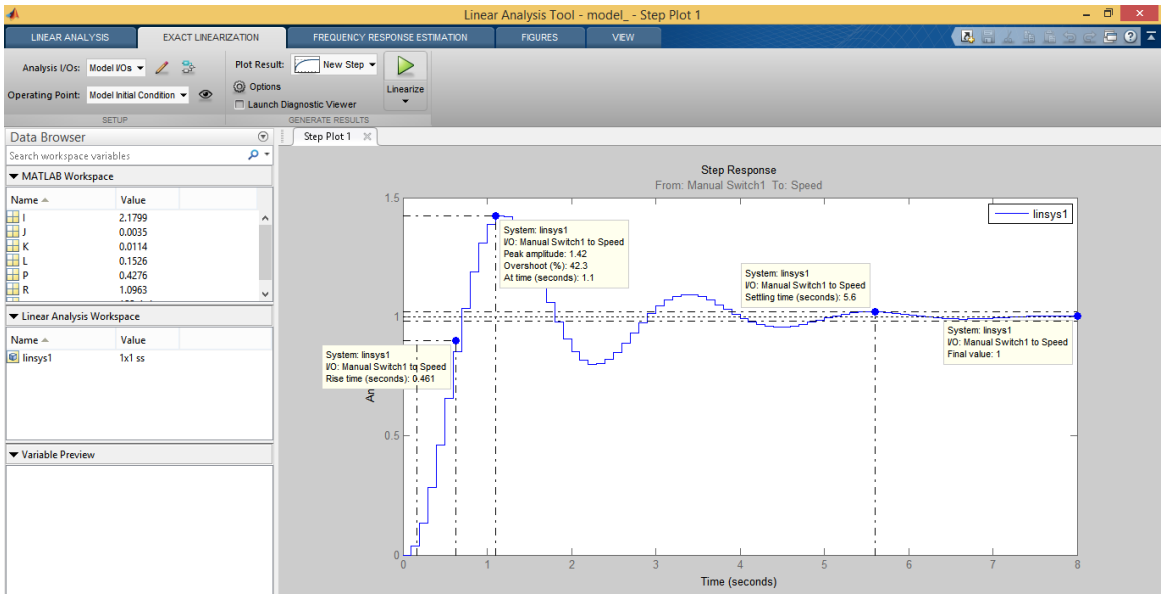


Figure 16 Linear System Analysis tool in MATLAB

3.4.4 Control System Tuning

This tool provides a very interactive way for designing controller in frequency domain.

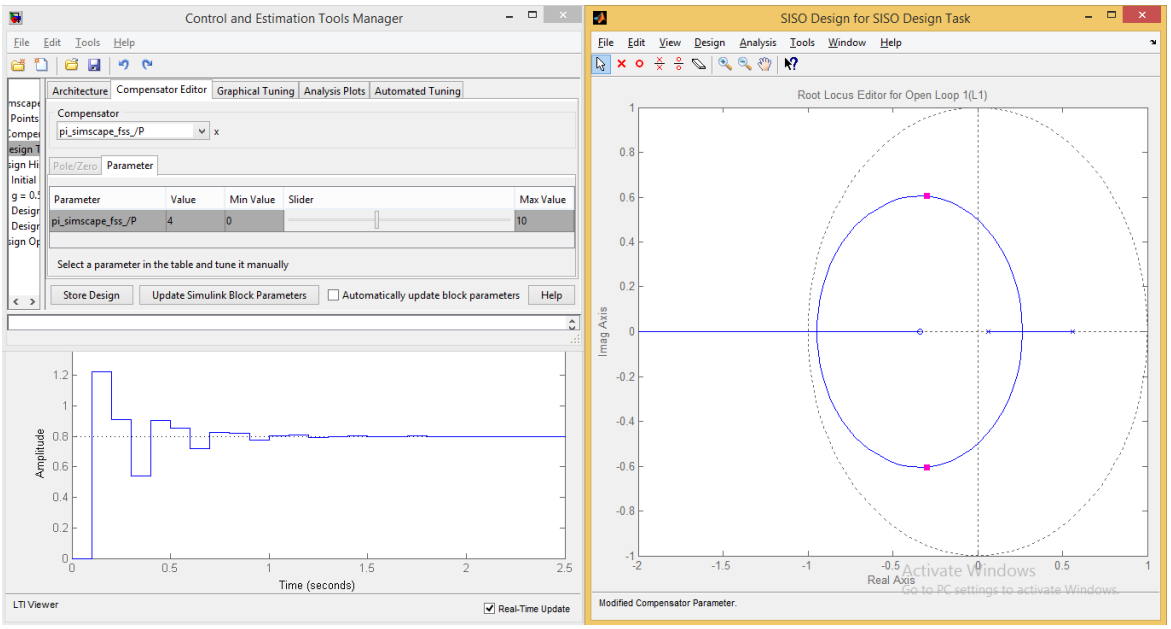


Figure 17 Control System Tuning tool in MATLAB

3.4.5 TrueTime

TrueTime is a MATLAB/Simulink-based simulator for real-time control systems. It facilitates co-simulation of controller task execution in real-time kernels, network transmissions, and continuous plant dynamics. The real-time kernel provides different scheduling techniques. It is very well-suited for simulation of packet dropouts in NCS. The software has been developed at Lund University, Sweden. We had spent a lot of time on this tool in the starting months. Lately, however, we have not found much need for this tool for our purposes due to the limited scope of our project.

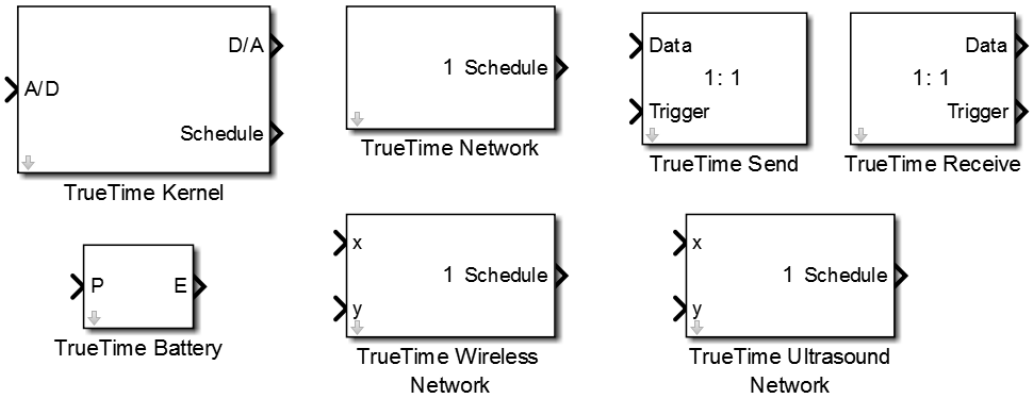


Figure 18 Simulink blocks provided by TrueTime

3.5 SPEED CONTROL PLANT

The plant side consists of two separable parts. One is the development board, and the other is the motor/sensor.

3.5.1 Development board

This single PCB hosts all of the electronic circuitry (except the sensor) needed at the plant side. Each distinct circuit is demarked on the top side of the board. These circuits are:

- Microcontroller
- Ethernet Shield
- Comparator
- Switched drive
- Power supply

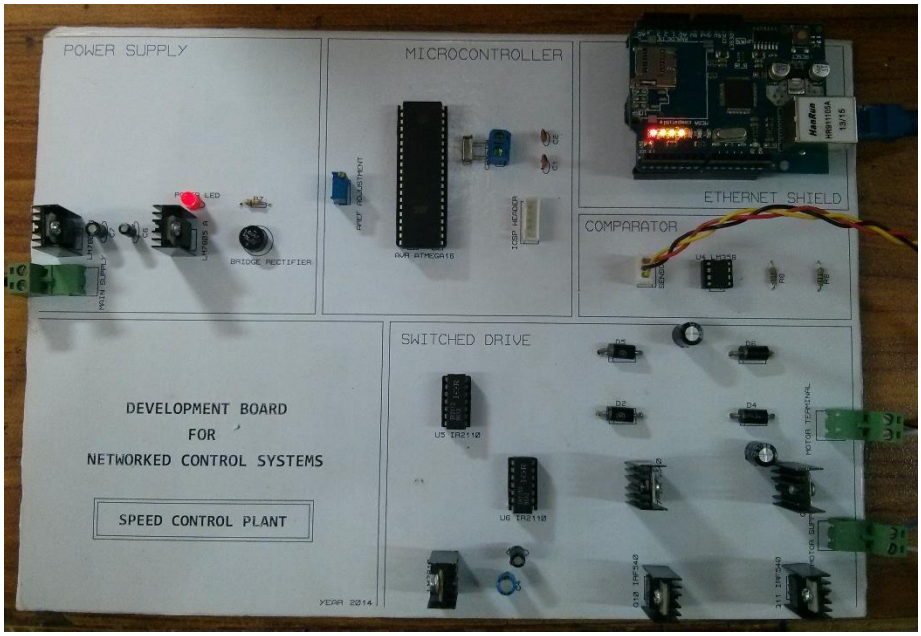


Figure 19 Development board for the speed control plant

3.5.2 Microcontroller

The microcontroller (MC) is used as an Interface between the plant and network. We have used AVR ATMega16 MC. It performs the following two functions:

1. Receive the Control Signal from the Ethernet via Hardwired TCP/IP embedded Ethernet controller i.e. w5100 chip through serial communication using SPI protocol.
2. Convert the Control Signal information to generate Pulse Width Modulated (PWM) wave with a specific duty cycle corresponding to the Control Signal information.

3.5.3 Ethernet shield

The W5100 chip by WIZnet features a hardwired TCP/IP stack and Ethernet protocols. The stack is capable of both TCP and UDP. It supports up to four simultaneous socket connections. It provides an SPI interface for communication with the MC. We have used a module called Arduino Ethernet Shield which has the chip already soldered on the PCB with other required components, such as the RJ-45 connector.

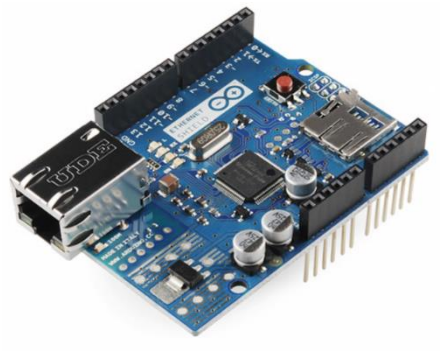


Figure 20 Arduino Ethernet Shield

3.5.4 Sensor

We have used a custom-made rotary encoder for measuring the speed of the motor shaft.

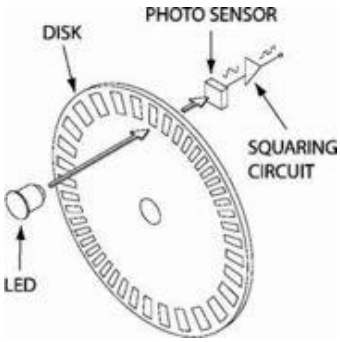


Figure 21 Diagram of an optical rotary encoder

In order to measure the speed of the motor, an optical encoder has been used. Optical Encoder is a device that converts the motion of the shaft into digital pulses. These pulses are fed to the Micro controller. The MC, then, counts the number of pulses in a given time and calculates the speed of the motor.

The Encoder consists of:

- U-shaped optocoupler
- Disk with multiple slots

The U-shaped optocoupler is an Infrared transmitter and receiver. The disk rotates between the transmitter and receiver. As light passes through the slots, it is received by the receiver and a pulse is generated.

In our case, there are 60 slots on the Encoder disk. The 60 slots ensure a simple relation between the shaft RPM and the encoder's output frequency. For example, at 2200 RPM, the encoder creates pulses at a frequency of 2200 Hz. The Encoder disk is metallic.

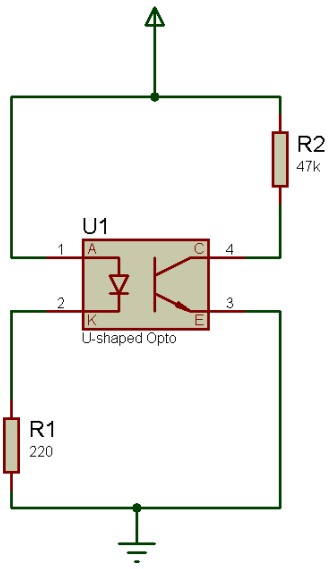


Figure 22 Schematic of the encoder circuit

The sample rate of the Speed Control plant is limited by the sensor.

3.5.5 Comparator

For the encoder, the optocoupler's rise time is not fast enough so that it can generate a wave that can be directly counted by the AVR. Therefore, the encoder output is fed into an op-amp comparator which solves this problem.

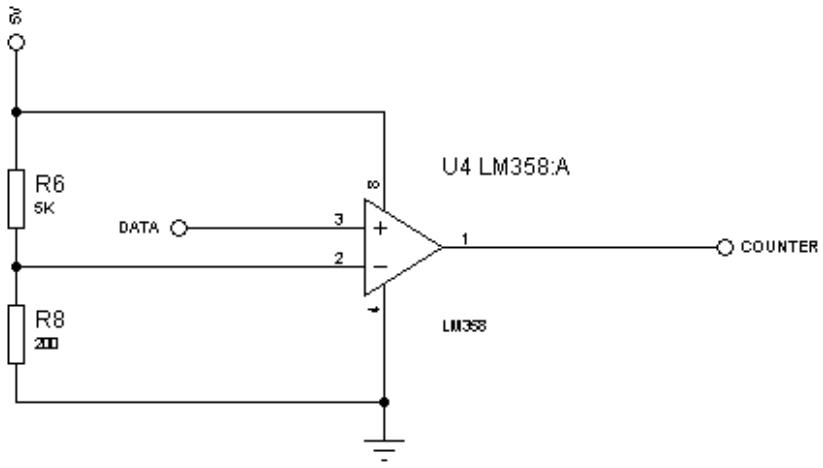


Figure 23 Schematic of the comparator circuit

3.5.6 DC motor

The DC motor used has a voltage rating of 24 V, and a power rating of about 100 W.

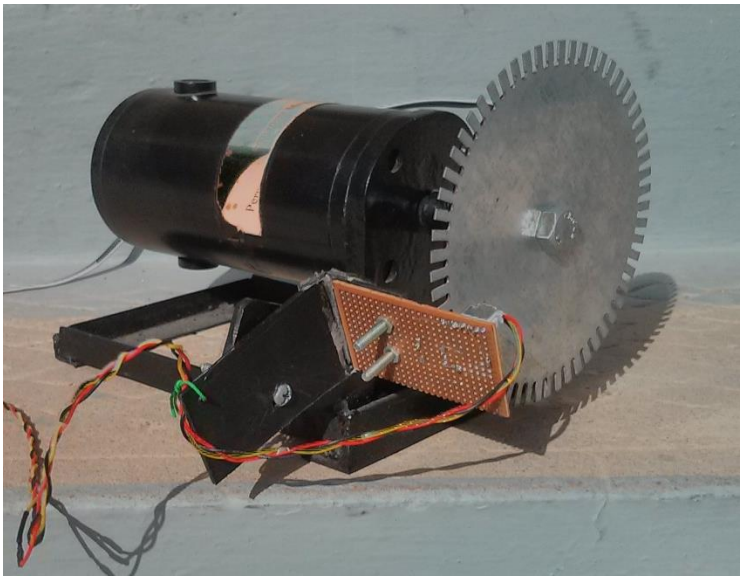


Figure 24 Picture of the speed control plant's DC motor

3.5.6.1 Mathematical modeling

A simple electromechanical model of a DC Motor is presented below:

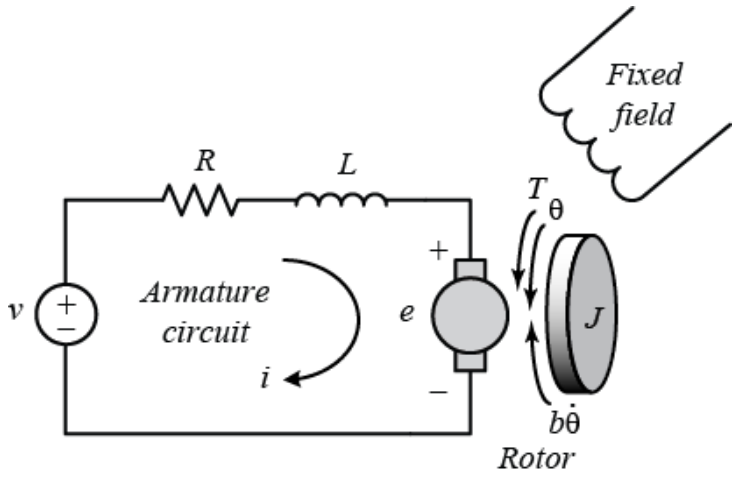


Figure 25 Electromechanical model of a DC motor

The induced torque in the motor is given by: $T = K_t i$ and the back electromotive force is given by: e = $K_e \frac{d\theta}{dt}$

Where T is the torque, K_t is the torque constant and i is the armature current, e is the back emf, K_e is the back emf constant and θ is the angular displacement of shaft.

The torque constant and the back EMF constant are essentially the same. Hence, for simplicity, we can state that $K_t = K_e = K$

The differential equations used to model the DC Motor are based according to Newton's 2nd law:

$$J\frac{d^2\theta}{dt^2} + b\frac{d\theta}{dt} = Ki \qquad (1)$$

Kirchhoff's Voltage law:

$$L\frac{di}{dt} + Ri = V - K\frac{d\theta}{dt} \qquad (2)$$

Where J is the moment of inertia, b is the viscous friction constant, L is the inductance, R is the resistance and V is the applied voltage to the motor.

Upon applying the Laplace Transform to equations (1) and (2), we get the following algebraic equations in the s-domain:

$$s(Js + b)\Theta(s) = KI(s)$$

and

$$(Ls + R)I(s) = V(s) - Ks\Theta(s)$$

Hence the open loop transfer function is given by:

$$\frac{\Theta(s)}{V(s)} = \frac{K}{(Js + b)(Ls + R) + K^2}$$

where $\Theta(s)$ is the output and $V(s)$ is the input.

3.5.6.2 MATLAB modelling

The following model, which is based on the above mathematical modelling, was used for initial simulations in MATLAB.

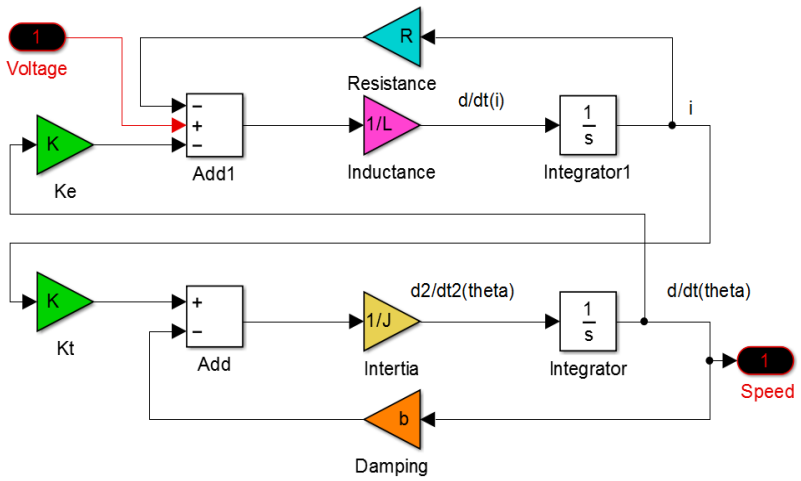


Figure 26 ODE based model of a DC motor in Simulink

Afterwards, we switched to the following model, which uses the Simscape toolbox.

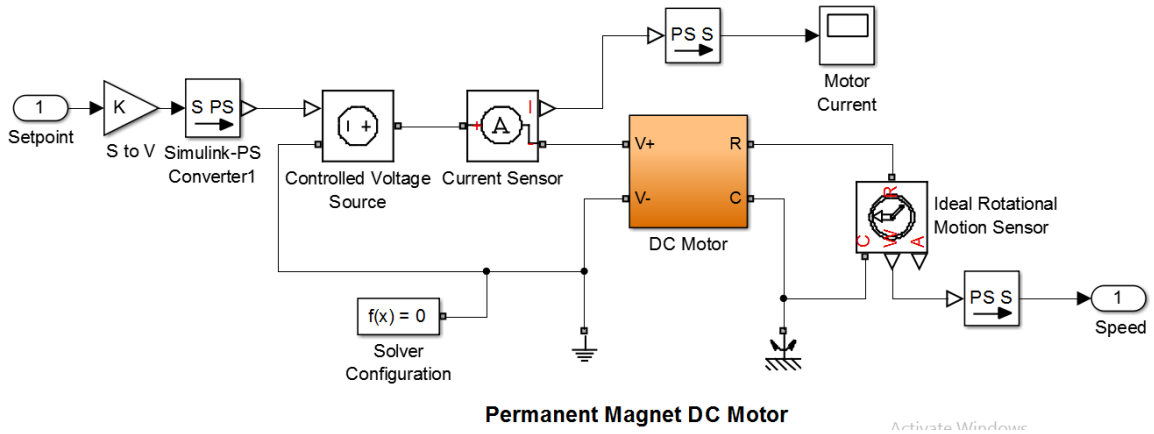


Figure 27 MATLAB model of a DC motor using the Simscape toolbox

The motor we have used has rating of about 100W, 24 Volts.

3.5.6.3 Voltage-speed response

In order to determine the speed response of the motor, we measured the speed with a tachometer at different voltage inputs. The results are provided below in tabular form.

The results when plotted showed reasonable linearity.

The following graph shows the steady state response of the motor at different voltages.

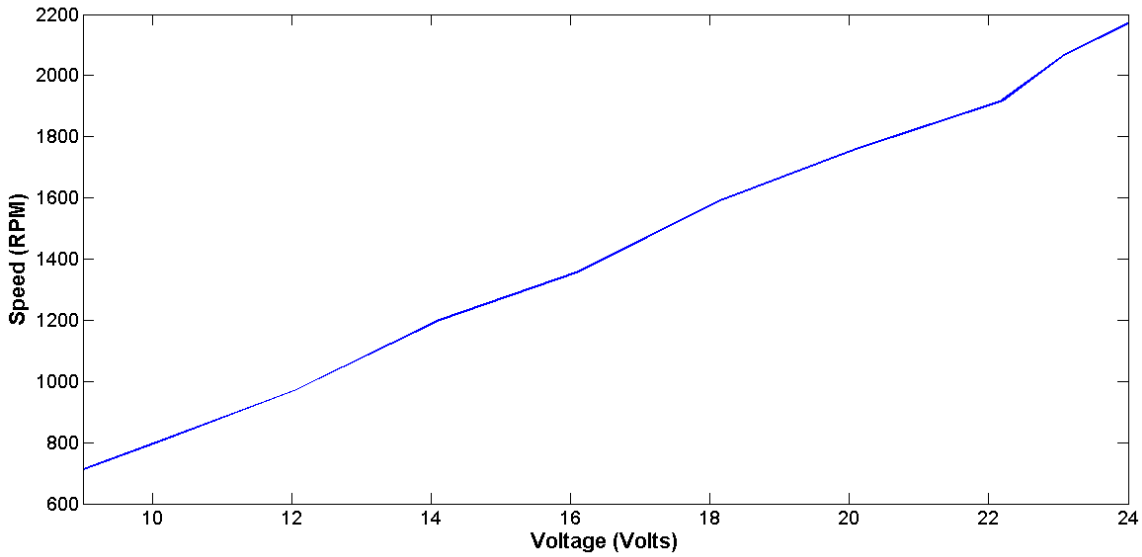


Figure 28 Voltage-speed response of the DC motor

3.5.6.4 Parameter estimation

Using the technique described in section 3.5, the parameters for the motor were estimated in MATLAB. An -open loop step response of the motor was taken with in real-time using MATLAB.

The following diagram shown how MATLAB changed the parameters in every iteration of the simulation and found the best match for our system.

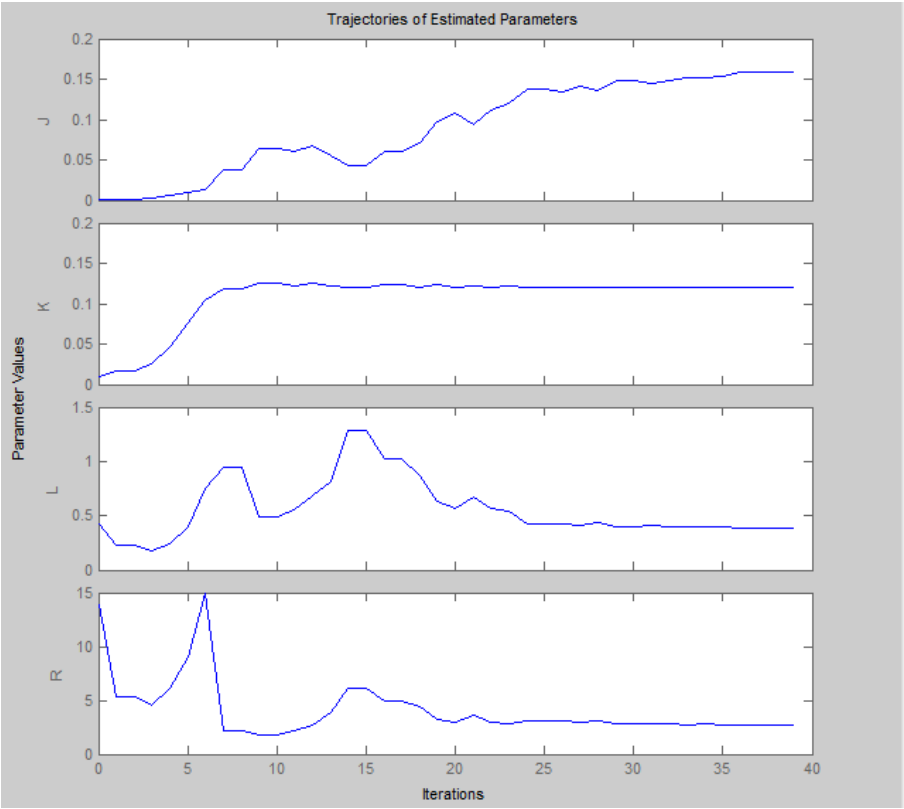


Figure 29 Trajectories of the estimated parameters of the DC motor, formed while running the Estimation

The estimated parameter values were as follows:

$$J = 0.0035$$

$$K = 0.0114$$

$$L = 0.1526$$

$$R = 1.0963$$

The accuracy of these estimated parameters can be judged from the graph in section 4.1.

3.5.7 Actuator

We have made a MOSFET based h-bridge drive, which works by the switching on and off the voltage supply to the motor. The IR2110 ICs shift the level and increase the power of the logic signal to drive the MOSFET gates. Its schematic is shown below:

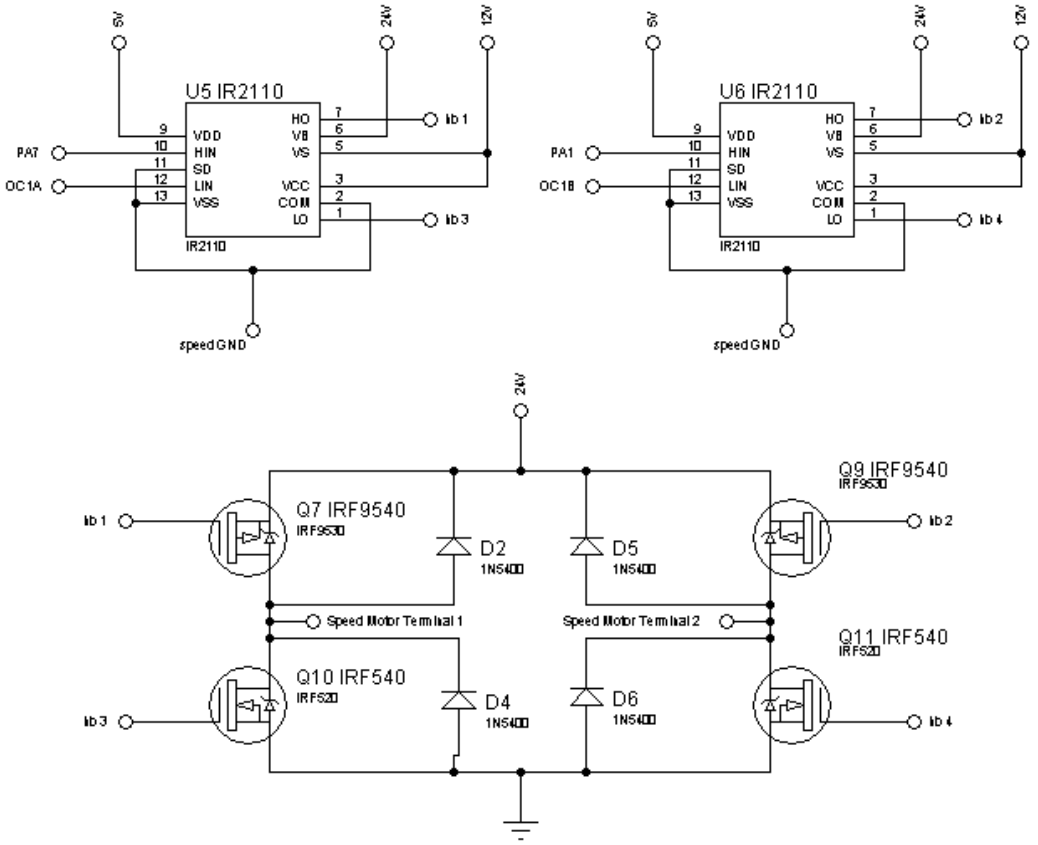


Figure 30 Schematic of the DC motor's drive

3.5.7.1 Duty-cycle vs. speed response

This graph was taken with 23.5 V applied to the switched drive's supply. It is evident from the following graph that apart from the dead band region, the response is very linear.

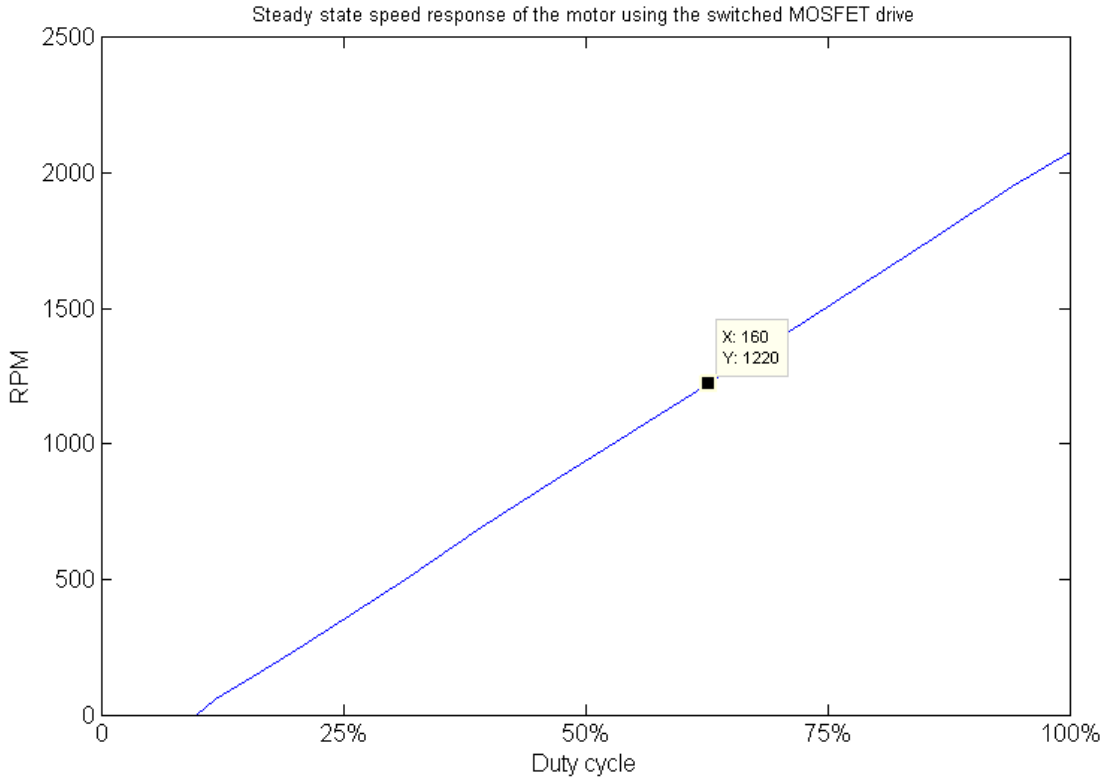


Figure 31 Duty-cycle vs. steady state speed response graph

3.6 OTHER PLANTS

As has been mentioned before, we have worked on two other plants. As they constitute an extra effort that does not come under the initially defined scope, and also because they are not fully operational at the time of writing, their details have been moved to the Supplementary chapter of this document. These plants are:

- Aero-pendulum plant, discussed in section 6.1.
- Position control plant, discussed in section 6.2

3.7 MICROCONTROLLER SOFTWARE

The code is divided into multiple files. All files except *types.h* and *main.c*, are in pairs: one *c* file and one *h* (header) file. We will briefly discuss each of these here.

First, let us look at the file listing:

- main.c
- cat_lib.h
- cat_lib.c
- lcd.h
- lcd.c
- *w5100lib* (directory)
    - dhcp.h
    - dhcp.c
    - socket.h
    - socket.c
    - spi.h
    - spi.c
    - types.h
    - w5100.h
    - w5100.c

3.7.1 main.c

All files are same for each plant except for the main file. This file uses functions defined in other files for operation. Using this approach has resulted in a more concise and readable code.

3.7.2 lcd

As the name indicates, these files contain code that was used for controlling an LCD. An LCD was used for debugging purposes during the initial phase of writing code for accessing the W5100 chip.

3.7.3 cat_lib

These files contain functions and macros that provide access to some basic features of the microcontroller, such as ADC, Timers and Counters.

3.7.4 w5100lib (directory)

This is the library that provides access to different features of the W5100 chip. It was obtained from the chip's manufacturer, WIZnet. However, it was not directly functional for our system and some changes and additions had to be made. We will discuss each part of the library using a bottom-up approach.

3.7.4.1 types.h

This file contains definitions of some data types that are used in the rest of the library. It also contains macros that can be defined according to the user's need, e.g. for the choice of data bus to be used for communication.

3.7.4.2 spi

These files contain functions that enable basic communication between the MC and the W5100 chip using the SPI bus.

3.7.4.3 w5100

It provides functions for accessing W5100 registers and memory. It uses *spi.h*.

3.7.4.4 socket

It provides standard functions to allow socket programming. It uses *w5100.h*.

3.7.4.5 dhcp

A simple and incomplete version of the DHCP-client software is implemented here. However, DHCP has not been used in later versions of the MC code in order to maintain simplicity.

The code is available on the CD submitted along with this document. The code is well commented and it is hoped that it can be understood, after some effort, by reading.

4 KEY FINDINGS AND RESULTS

In this section, the simulations and real-time responses will be shown for the Speed control plant. The results are shown with the help of both figures and numbers.

The following model has been used for simulations:

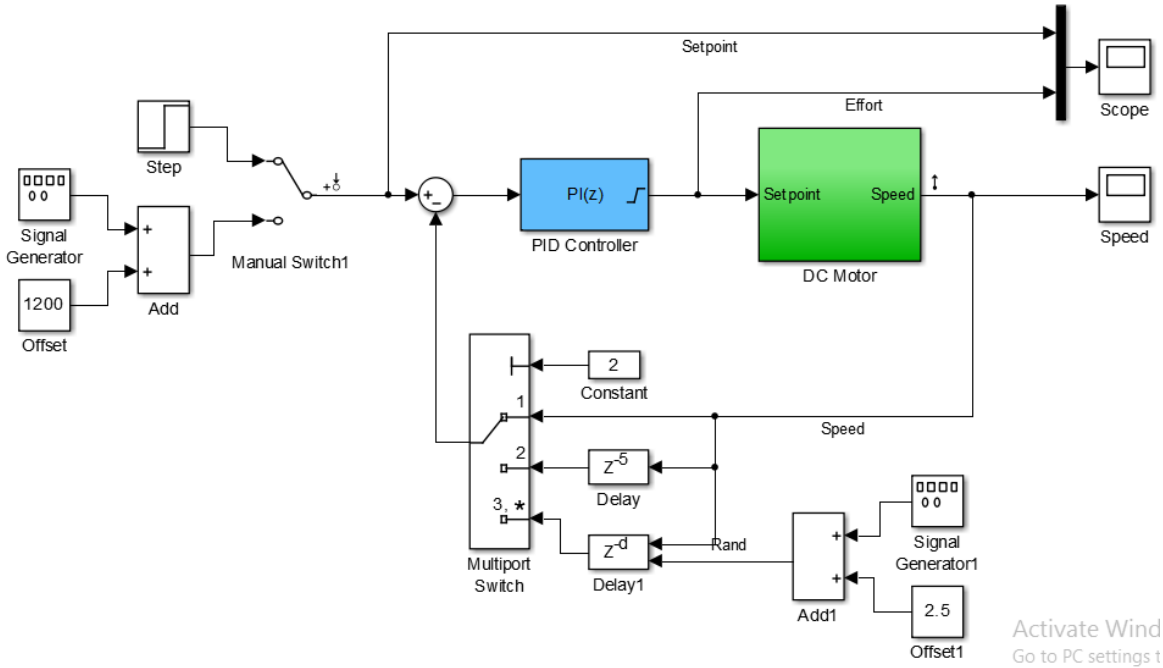


Figure 32 MATLAB model for simulations of the speed control plant

4.1 MEASURED VS. SIMULATED RESPONSES

We have discussed the Estimation of the model parameters for our motor in the previous chapter. The general technique of Parameter Estimation is discussed in section 3.3, while section 3.5.6.4 discusses the application of this technique for the DC motor.

In the following graph, open loop step responses are shown. It is apparent from the graph that the estimation has been quite successful.

The unit of speed is RPM, and this unit will be followed throughout this document.

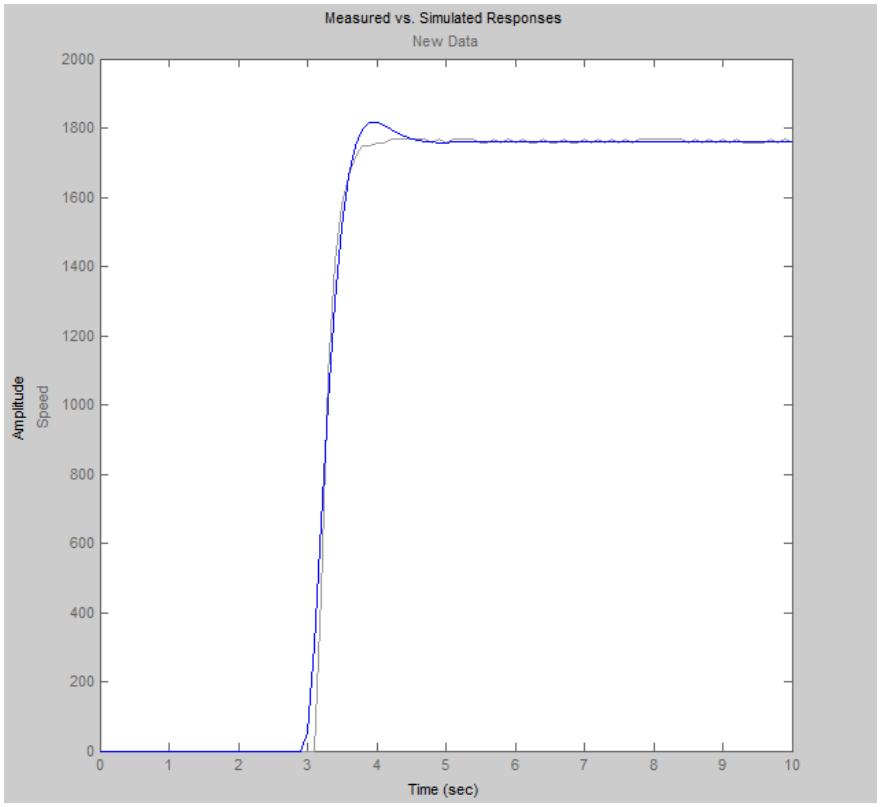


Figure 33 Measured vs. simulated response

For the simulated response, the characteristic parameters are as follows:

Rise time = 0.495 s

Overshoot = 2.47 %

Settling time = 1.19 s

Error = 0

For the real-time (RT) measured response, the parameters are almost the same except for overshoot. The overshoot is zero for the RT response.

This anomaly is caused due the fact that in the simulated response, two phenomena that appear in real systems have not been taken into account. They are:

1. Dead region, in which the motor does not run.
2. Friction, which is also the main cause of the dead band.
3. Voltage limitations of both the Motor and the Drive. (Saturation.)

Another reason is that the sensor is not ideal and only gives an average speed measured over an internal of 95 milliseconds.

4.2 CLOSED-LOOP CONTROL

4.2.1 Simulation

Using the method described in section 3.6.1, different gain settings were found for the system. We have worked on a PI controller, each set of gains contains two parameters: P and I. A complete list of these gain settings is given in section 4.5.1.

For the graph that follows, no delays were *applied*. The gains are as follows:

$$I = 0.42758$$

$$P = 2.1799$$

The following graph shows the closed loop step response of the modeled system.

The input signal is colored blue, while the output signal is colored red. The controlled parameter is speed and the units are in revolutions per minute (RPM).

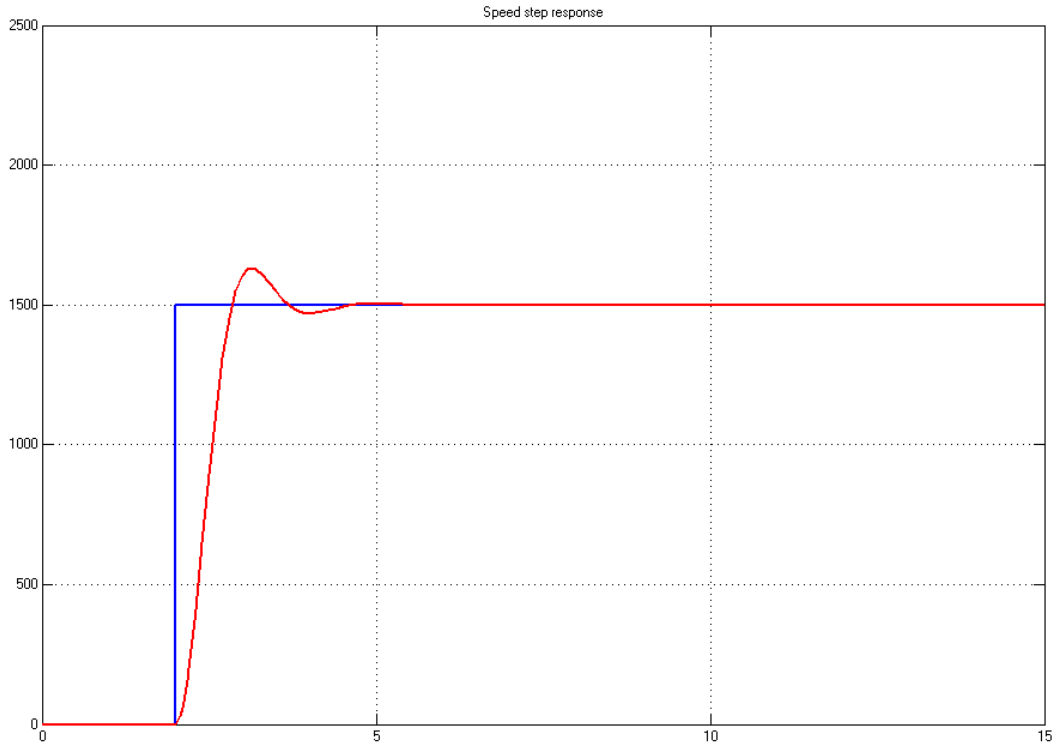


Figure 34 Simulation with tuned controller

Rise time = 0.495s

Overshoot = 2.47%

Settling time = 1.19s

Error = 0

4.2.2 Measured response

For the graph that follows, no delays were applied. The gains are as follows:

$$I = 0.42758$$

$$P = 2.1799$$

The input signal is colored blue, while the output signal is colored red. The controller parameter is speed and the units are in revolutions per minute (RPM).

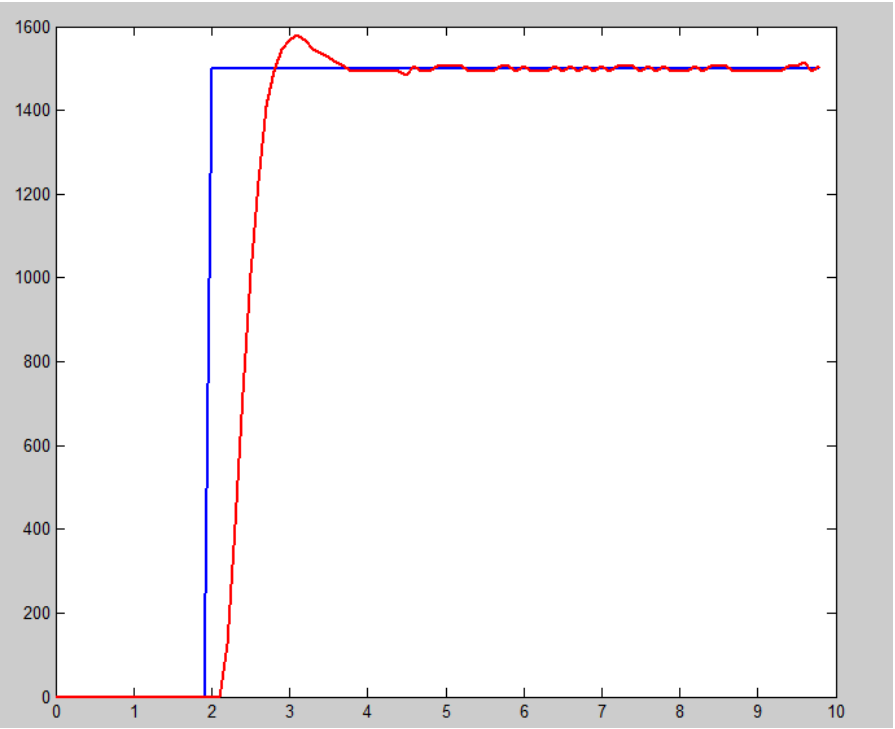


Figure 35 Real-time response with tune controller

Overshoot = 5%

Peak time = 1.1 s

Settling time = 1.5s

Error = 0

4.3 SIMULATIONS WITH DELAYS AND SAME CONTROLLER GAINS

Now unit step response are shown for different delays in the feedback loop. The response is deteriorated with increasing delays.

The controller sample rate is 10 Hz. Therefore, the delay can either be 0 ms, or 100 ms, or 200 ms, and so on.

The controller gain settings for all the responses shown in this section are:

$$P = 0.42758$$

$$I = 2.1799$$

4.3.1 Delay = 100ms

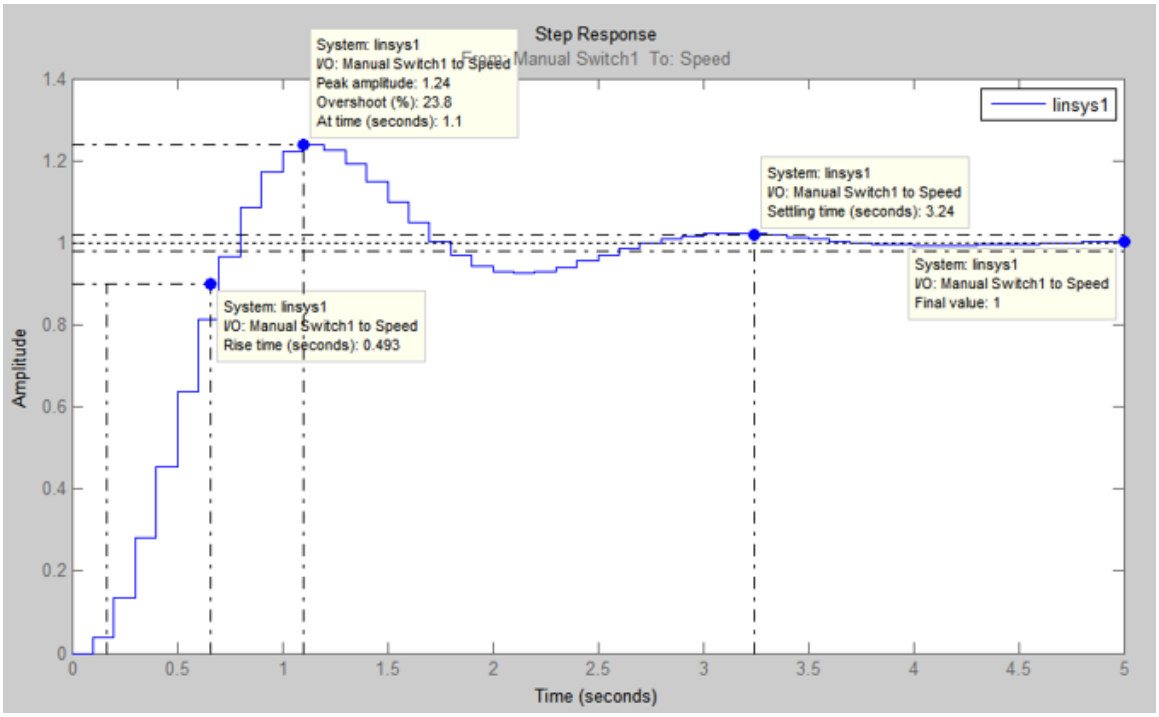


Figure 36 Effect of delays in simulation - 1

4.3.2 Delay = 200ms

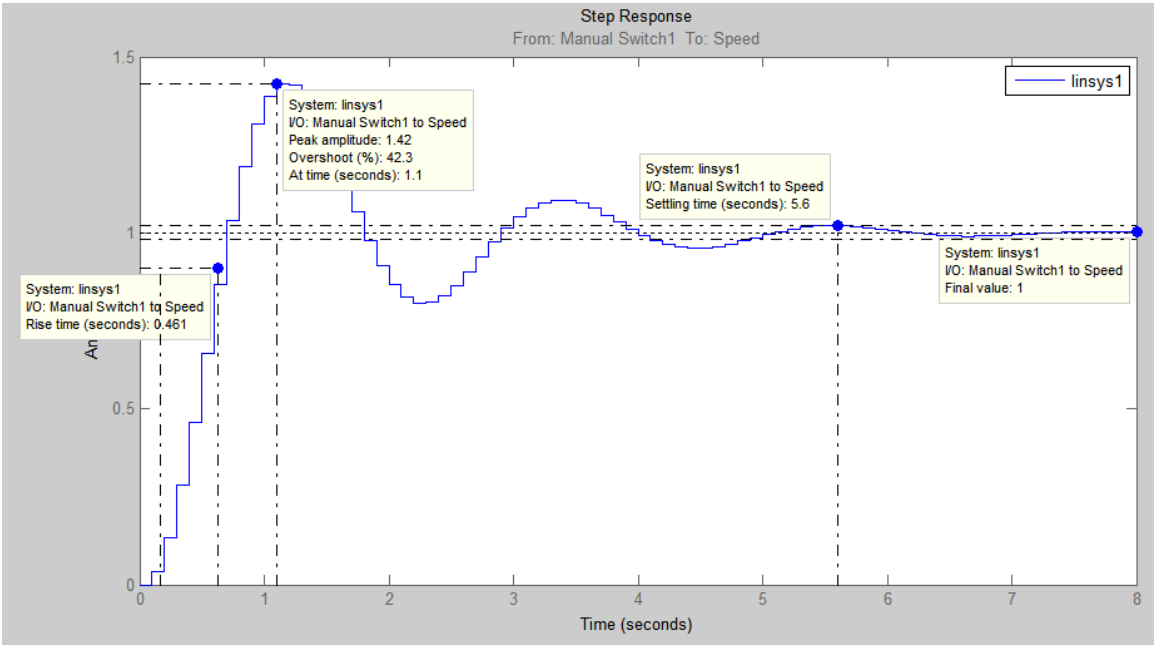


Figure 37 Effect of delays in simulation - 2

4.3.3 Delay = 300ms

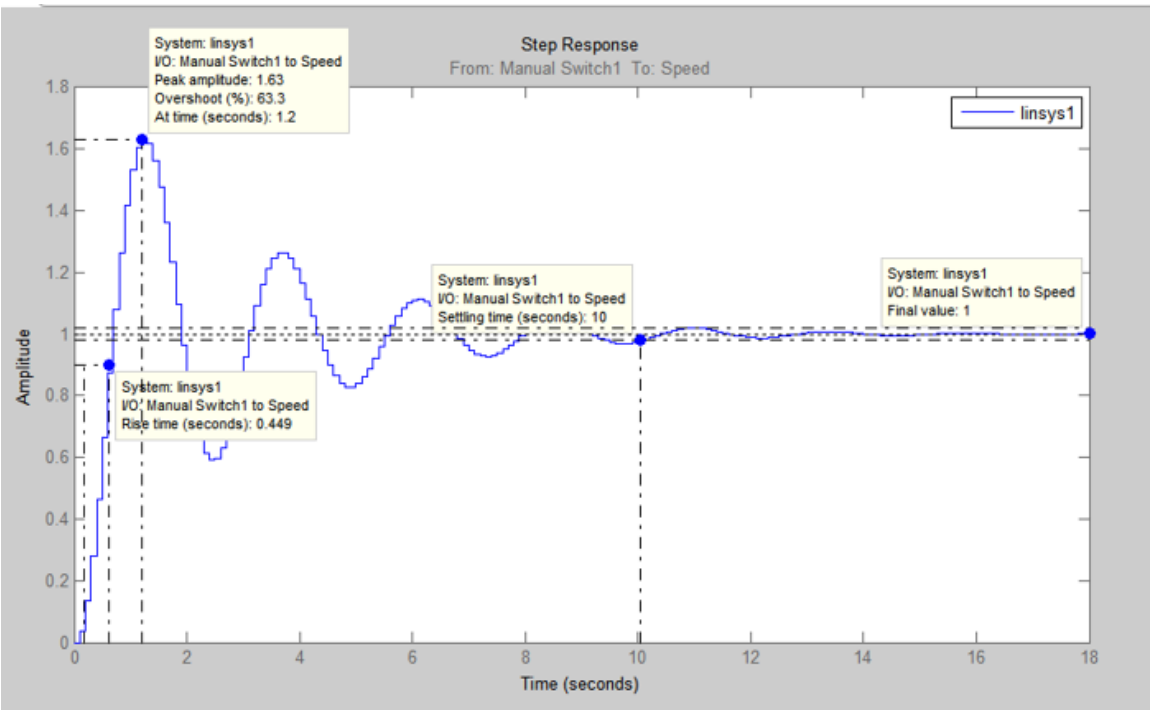


Figure 38 Effect of delays in simulation - 3

4.3.4 Delay = 400ms

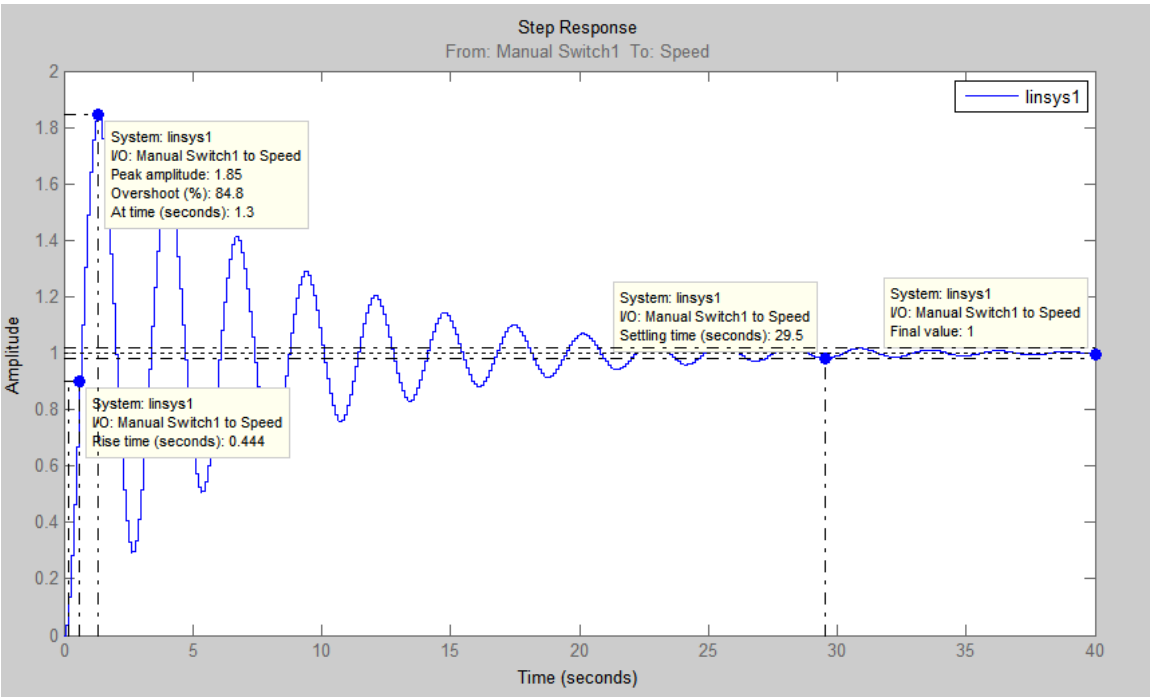


Figure 39 Effect of delays in simulation - 4

4.3.5 Delay = 500ms

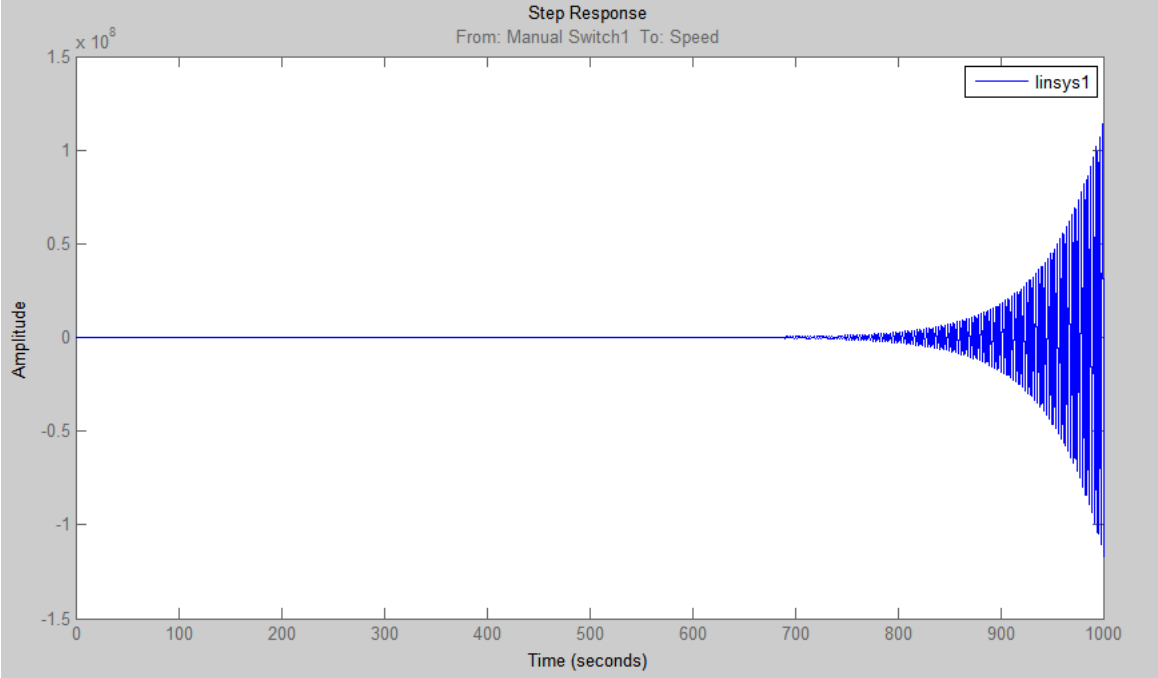


Figure 40 Effect of delays in simulation – 5

The system is unstable under current settings of delay and controller gains.

4.4 REAL-TIME RESPONSES WITH DELAYS

In this section, we will show the step responses of the real system after delays have been applied. One response is given for the controller gain at which the response is best without any delay in the system. Then the controller gain is changed which was found during simulations using the PID Tuning tool. The reader can see an appreciable difference in the system's performance after tuning the controller.

4.4.1 Gain-Delay relation

Table 2 Gain-Delay relation

DELAY (MS)	I	P
0	2.1799	0.4276
100	1.6541	0.3659
200	1.4039	0.3839
300	1.1608	0.3241
400	1.0272	0.3529
500	0.8967	0.3321

NOTE: THE INPUT SIGNAL IS COLORED BLUE, WHILE THE OUTPUT SIGNAL IS COLORED RED. THE CONTROLLED PARAMETER IS SPEED AND THE UNITS ARE IN REVOLUTIONS PER MINUTE (RPM). THIS IS THE CASE WITH ALL THE GRAPHS IN THIS SECTION.

4.4.2 Delay = 100 ms

The following response was obtained at the following controller settings:

$$P = 0.42758$$

$$I = 2.1799$$

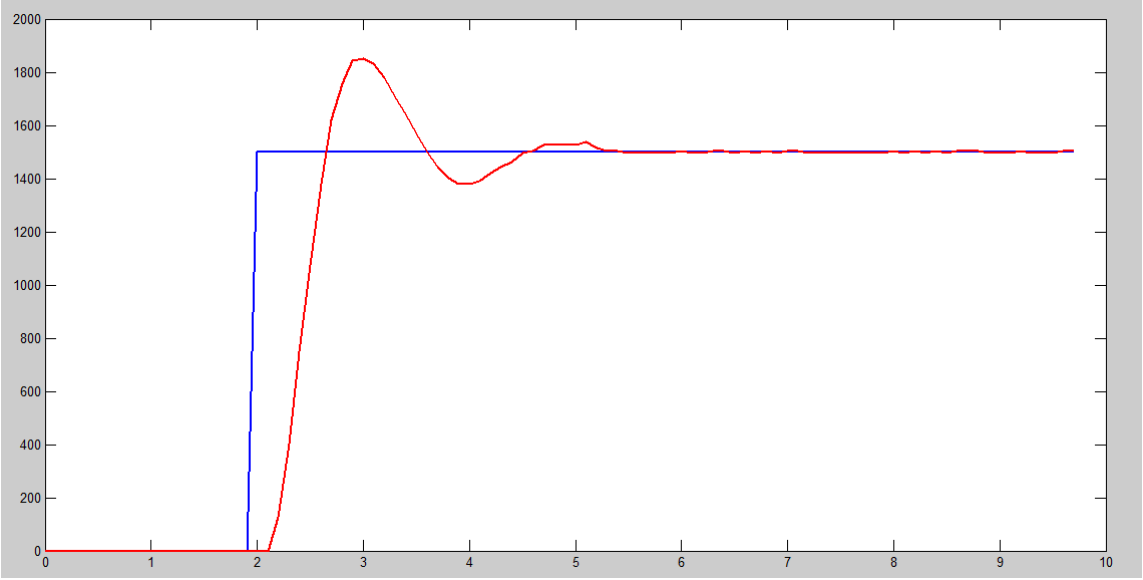


Figure 41 Delays in RT, un-tuned - 1

Overshoot = 23%

Peak time = 0.9 s

Settling time = 3s

Now, the controller was tuned to the following gains:

$$I = 1.6541$$

$$P = 0.3659$$

And the following response was obtained:

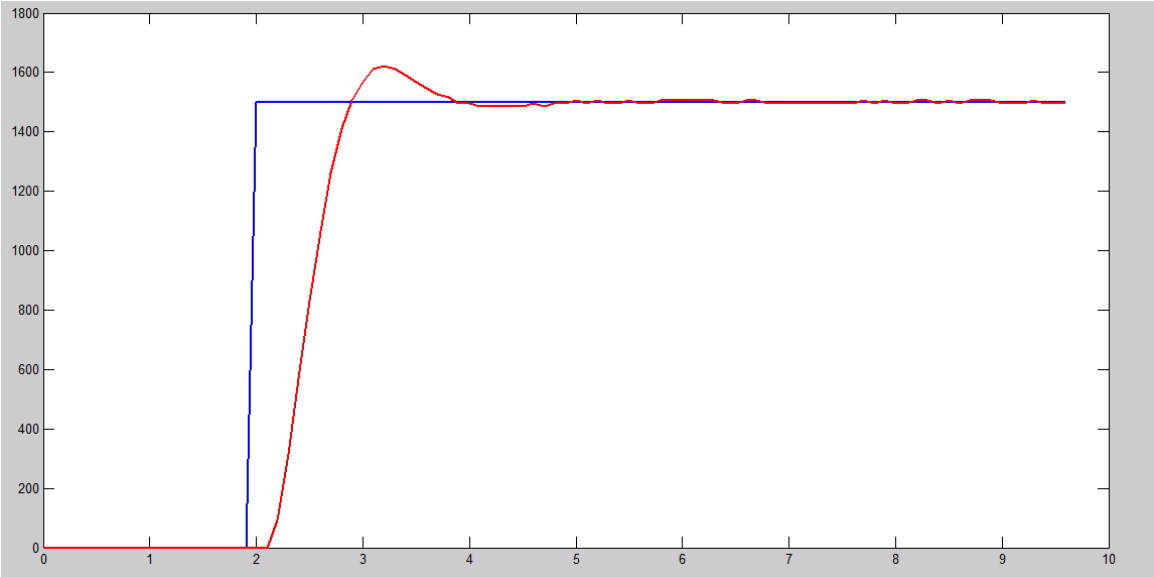


Figure 42 Delays in RT, tuned – 1

Overshoot = 4.7 %

Peak time = 1.1 s

Settling time = 1.8 s

4.4.3 Delay = 200 ms

The following response was taken at the following controller settings:

$$P = 0.42758$$

$$I = 2.1799$$

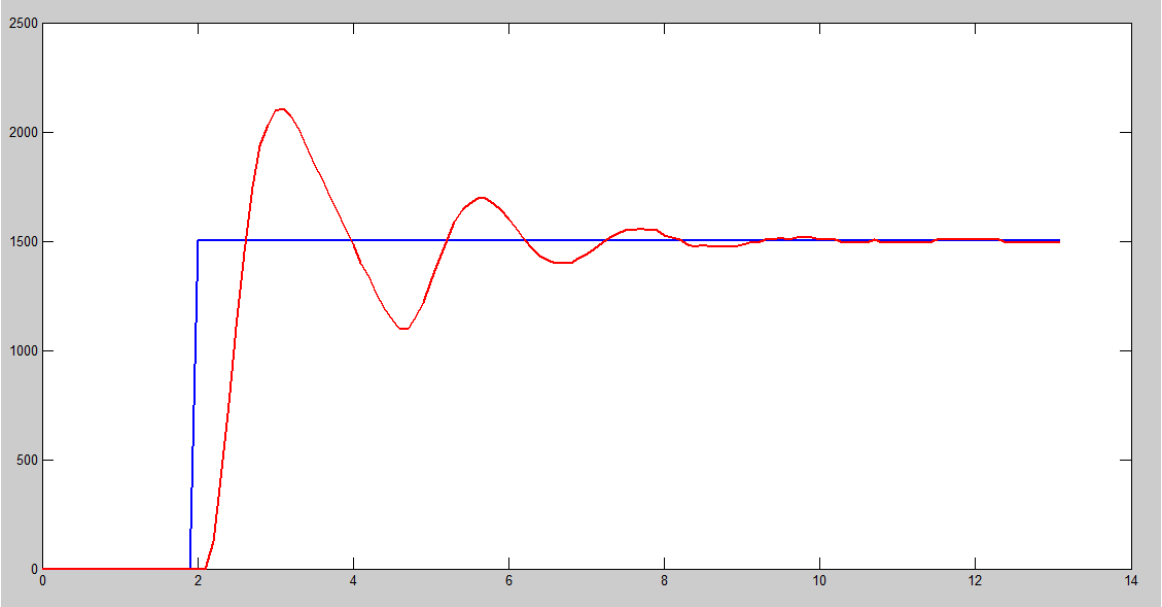


Figure 43 Delays in RT, un-tuned - 2

Overshoot = 40 %

Peak time = 1 s

Settling time = 7 s

Now the controller was tuned to the following gains:

$$I = 1.4039$$

$$P = 0.3839$$

And the following response was obtained:

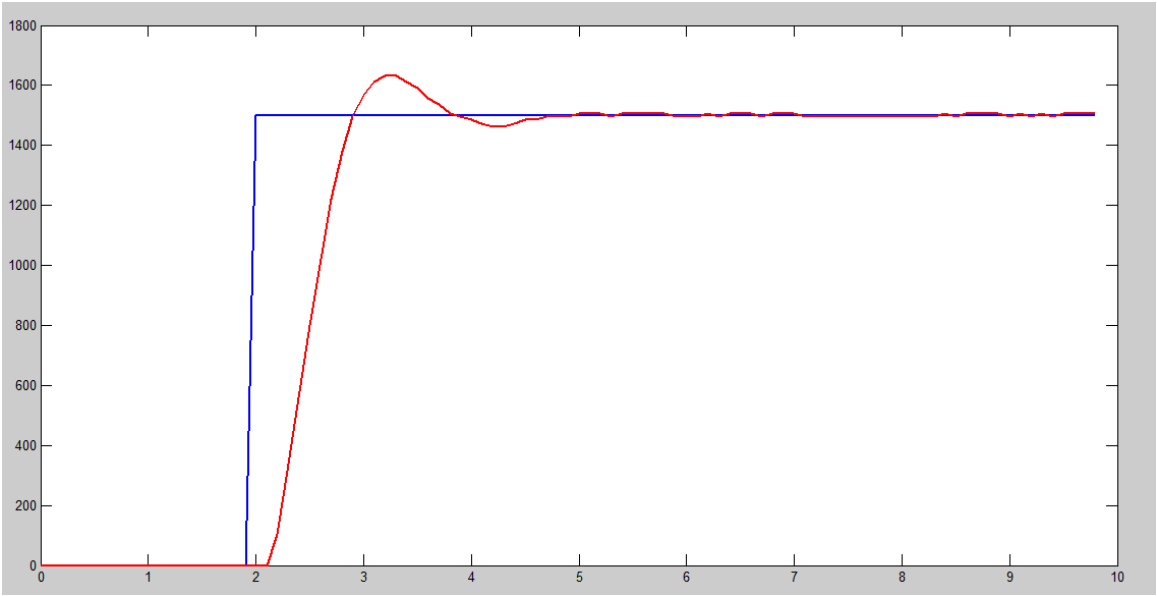


Figure 44 Delays in RT, tuned – 2

Overshoot = 5.3 %

Peak time = 1.2 s

Settling time = 2.2 s

4.4.4 Delay = 300 ms

The following response was taken at the following controller settings:

$$P = 0.42758$$

$$I = 2.1799$$

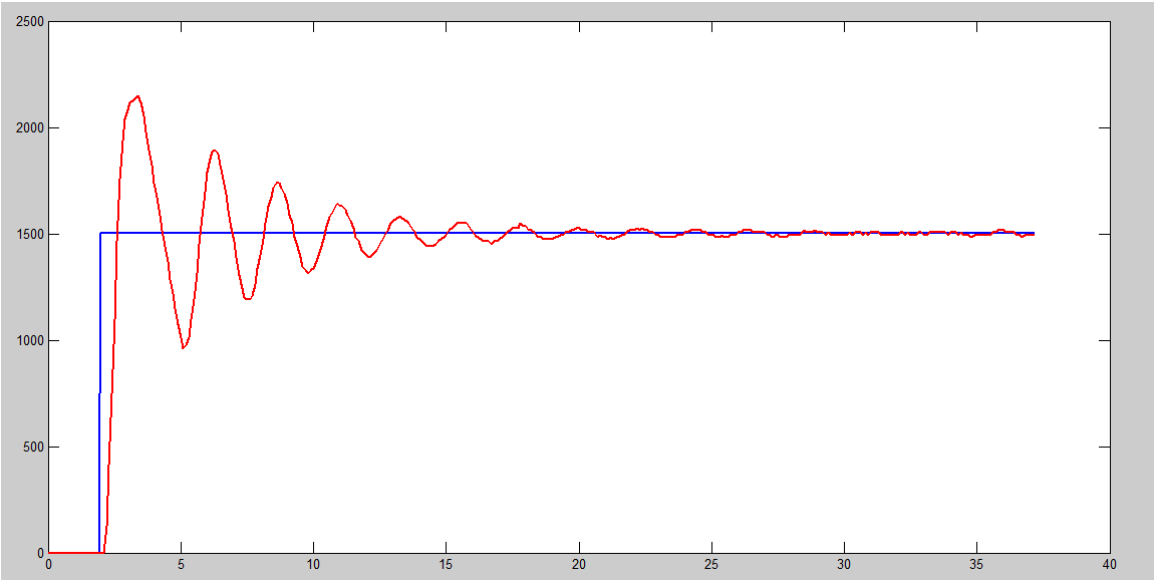


Figure 45 Delays in RT, un-tuned – 3

Overshoot = 42.5 %

Peak time = 1.2 s

Settling time = 19 s

Now the controller was tuned to the following gains:

$$I = 1.1608$$

$$P = 0.3241$$

And the following response was obtained:

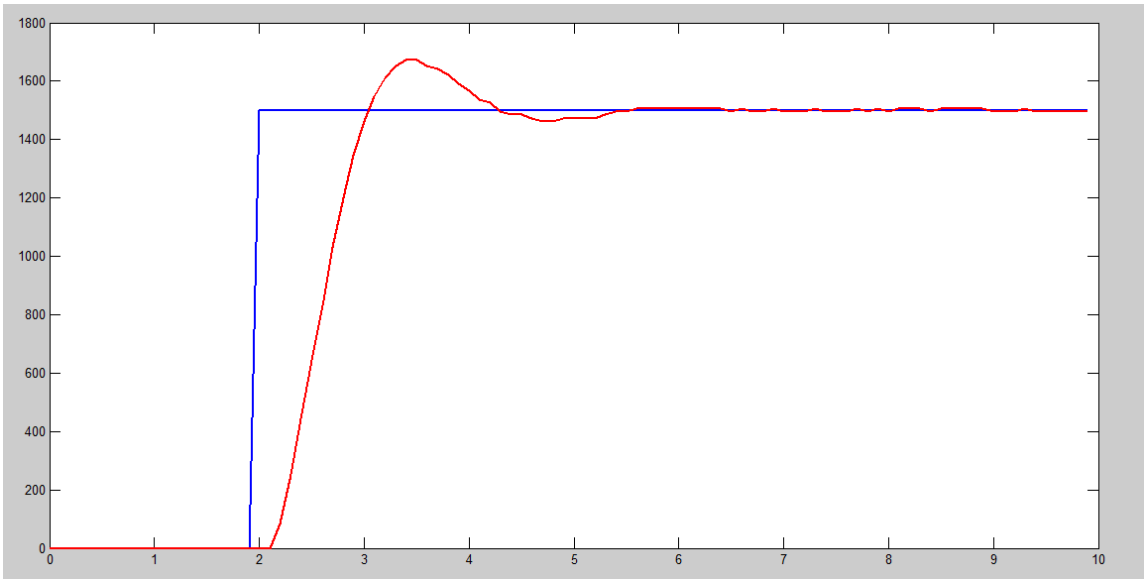


Figure 46 Delays in RT, tuned – 3

Overshoot = 11.6 %

Peak time = 1.4 s

Settling time = 3.3 s

4.4.5 Delay = 400 ms

The following response was taken at the following controller settings:

$$P = 0.42758$$

$$I = 2.1799$$

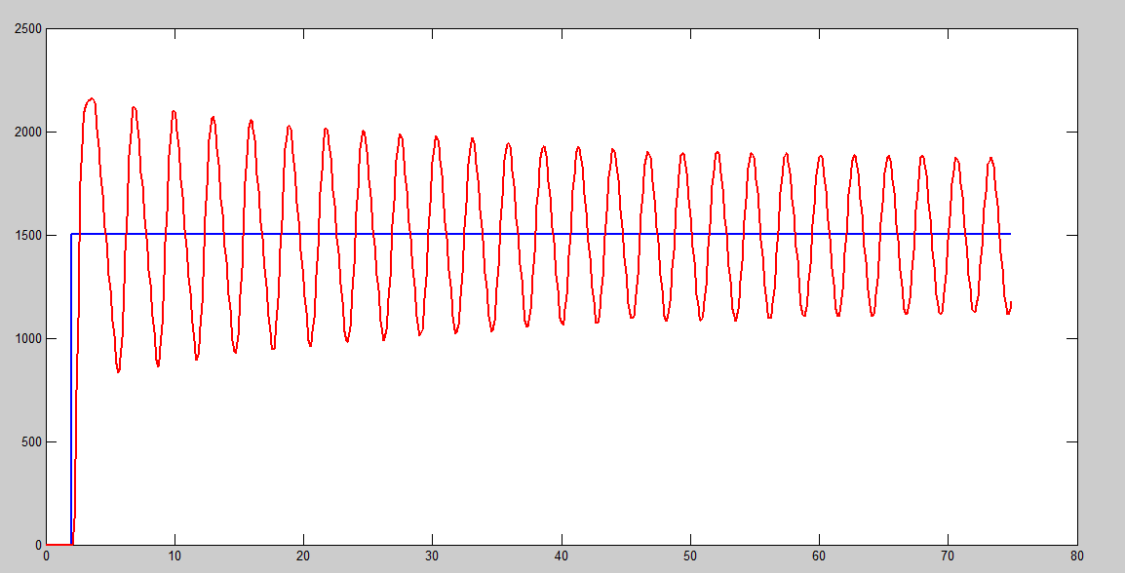


Figure 47 Delays in RT, un-tuned - 4

Overshoot = 47 %

Peak time = 1.5 s

Now the controller was tuned to the following gains:

$$I = 1.0272$$

$$P = 0.3529$$

And the following response was obtained:

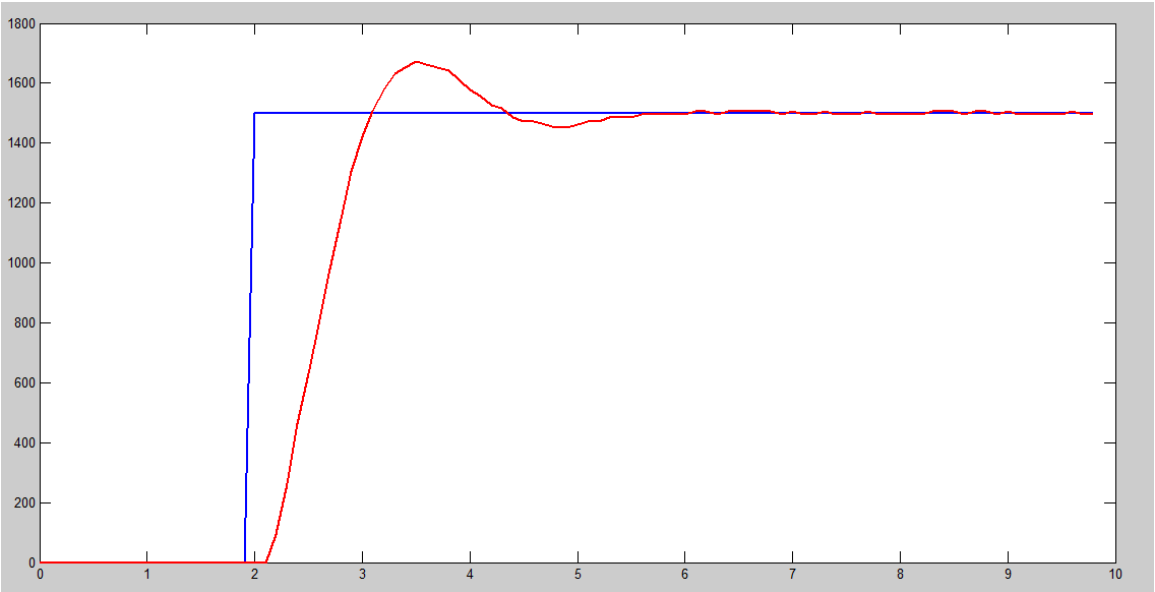


Figure 48 Delays in RT, tuned – 4

Overshoot = 11.6 %

Peak time = 1.4 s

Settling time = 3.3 s

4.4.6 Comparison

A comparison of the system responses shown above is now given in tabular form.

Table 3 Comparison of measured responses with delays

Delay	**100 ms**		**200 ms**		**300 ms**		**400 ms**	
	Un-tuned	**Tuned**	**Un-tuned**	**Tuned**	**Un-tuned**	**Tuned**	**Un-tuned**	**Tuned**
Overshoot	23%	4.7%	40%	5.3%	42.5%	11.6%	47%	11.6%
Peak Time	0.9 s	1.1 s	1 s	1.2 s	1.2 s	1.4 s	1.5 s	1.4 s
Settling Time	3 s	1.8 s	7 s	2.2 s	19 s	3.3 s	Very long	3.3 s

4.5 VARIABLE DELAYS

In the above responses, the delays have been constant. However, in an NCS, the delays vary with time. Now, we will consider this condition. We will first look at the effect of the change of delays on the system response. Then we will see how a more intelligent controller can recover the system performance by automatically tuning the controller gains at runtime. This smart controller makes decisions based on measured delay values.

4.5.1 Scheme of delay variation

For all of our analysis and measurements, we have used artificial delays applied in the MATLAB controller. The variable delays that are discussed in this section follow the following scheme:

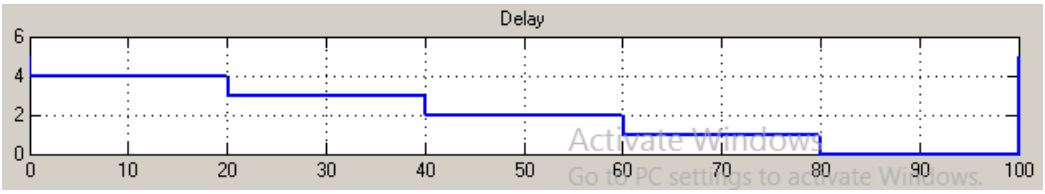


Figure 49 Scheme of delay variation

In the above figure, the number on the y-axis indicates the number of sample times by which all samples are being delayed. The sample time of our system is 100 ms, so the delays vary from 400 ms to 0 ms. The x-axis shows time in seconds.

We will show the responses of two waveforms now.

NOTE: FOR ALL THE FIGURES THAT FOLLOW IN SECTION 4.5, THE INPUT SIGNAL IS COLORED BLUE, WHILE THE OUTPUT SIGNAL IS COLORED RED. THE CONTROLLED PARAMETER IS SPEED AND THE UNITS ARE IN REVOLUTIONS PER MINUTE (RPM).

4.5.2 Sawtooth waveform

The frequency of the waveform is 0.2 Hz.

4.5.2.1 Untuned

The following figures shows the response with delays varying according to the scheme discussed in section 4.5.1. The controller gains are:

$$P = 0.42758$$

$$I = 2.1799$$

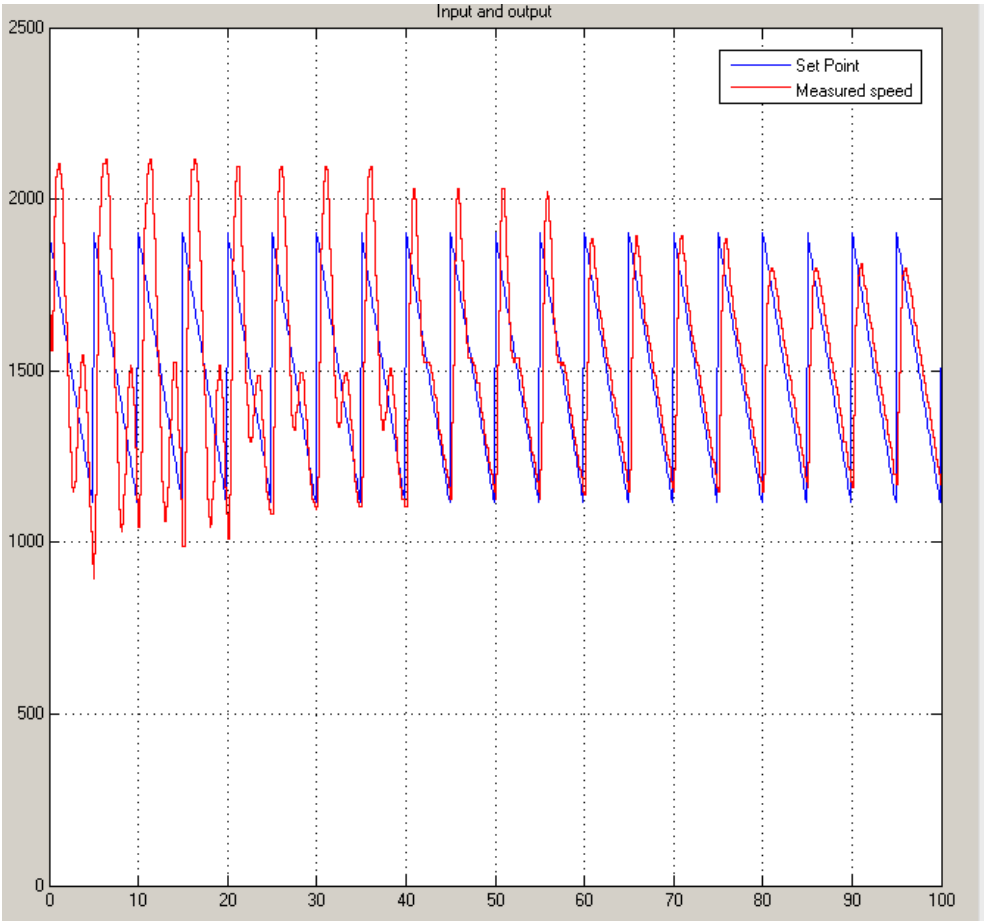


Figure 50 Sawtooth with variable delays – untuned

4.5.2.2 Automatically tuned

Now, we will see how the controller improves the response of the system by automatically changing the controller gains at runtime. The gain-delay relation shown in section 4.4.1 is followed here.

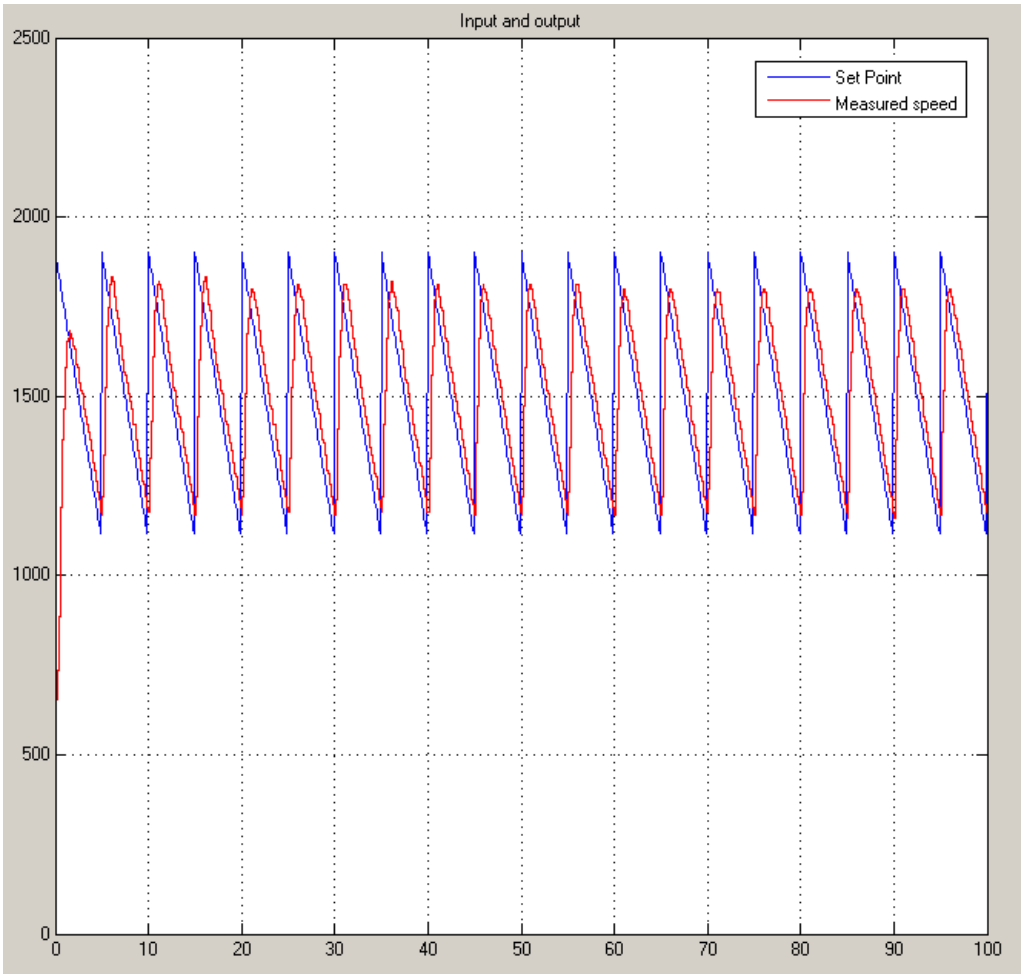


Figure 51 Sawtooth with variable delays – automatically tuned

4.5.3 Square waveform

In this section, the responses of a square wave will be shown. The frequency of the waveform is again 0.2 Hz.

4.5.3.1 Untuned

The controller gains are again:

$$P = 0.42758$$

$$I = 2.1799$$

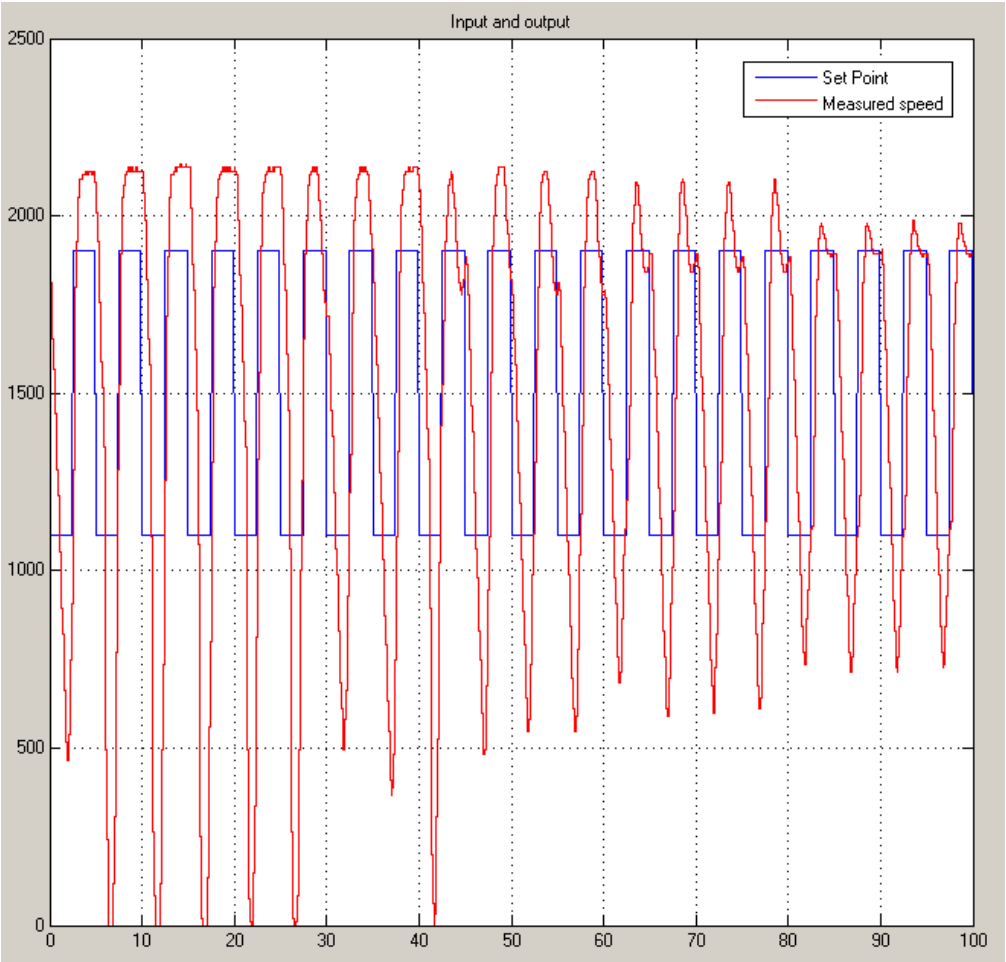


Figure 52 Square wave with variable delays – untuned

4.5.3.2 Automatically tuned

Like section 4.5.2.2, we will now see how the controller improves the response of the system by automatically changing the controller gains at runtime. The gain-delay relation shown in section 4.4.1 is followed here.

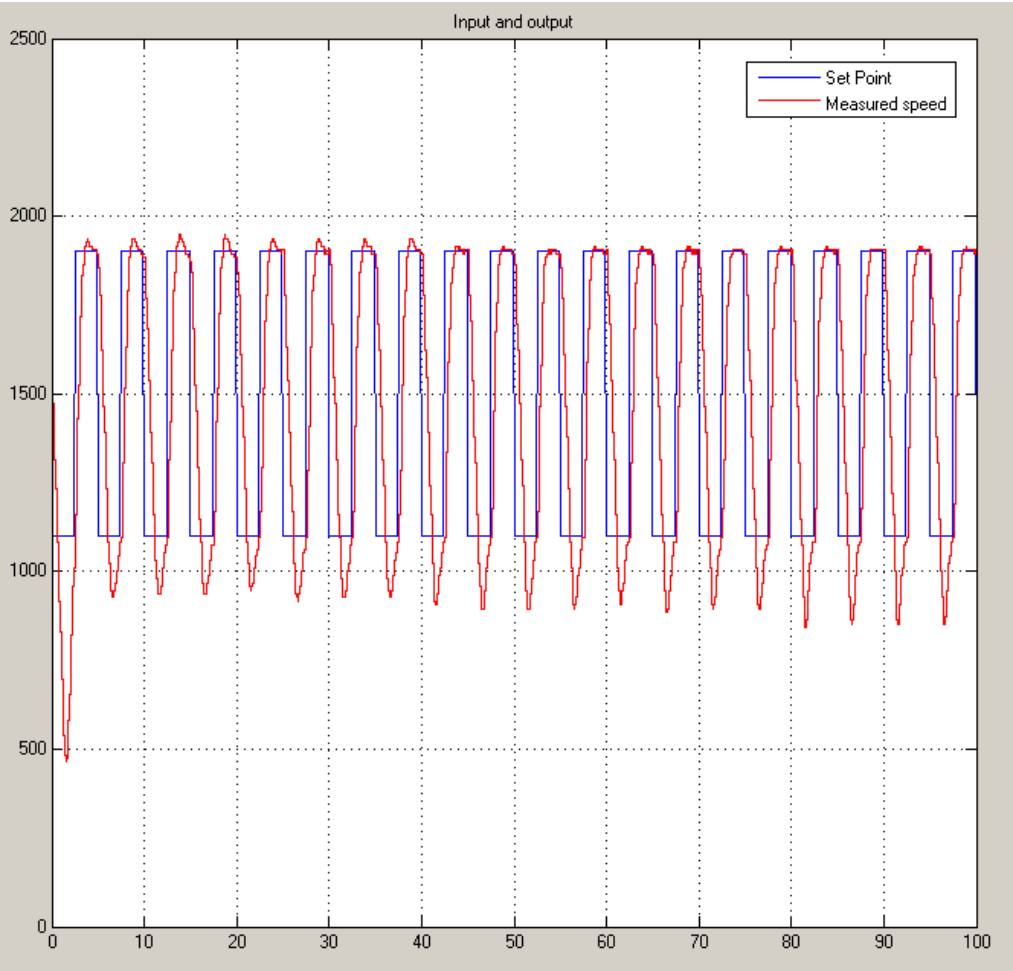


Figure 53 Square wave with variable delays - automatically tuned

4.5.3.3 Automatically tuned with resets

In a PI controller, the integrator stores a value which is equal to the weighted sum of the system errors and the controller's Integral gain. In the previously discussed automatically tuned controllers, this value is not automatically cleared when the gains are changed. We will now look at the response of the controller that clears the integrator's value whenever the delay (and hence the controller gains) are changed.

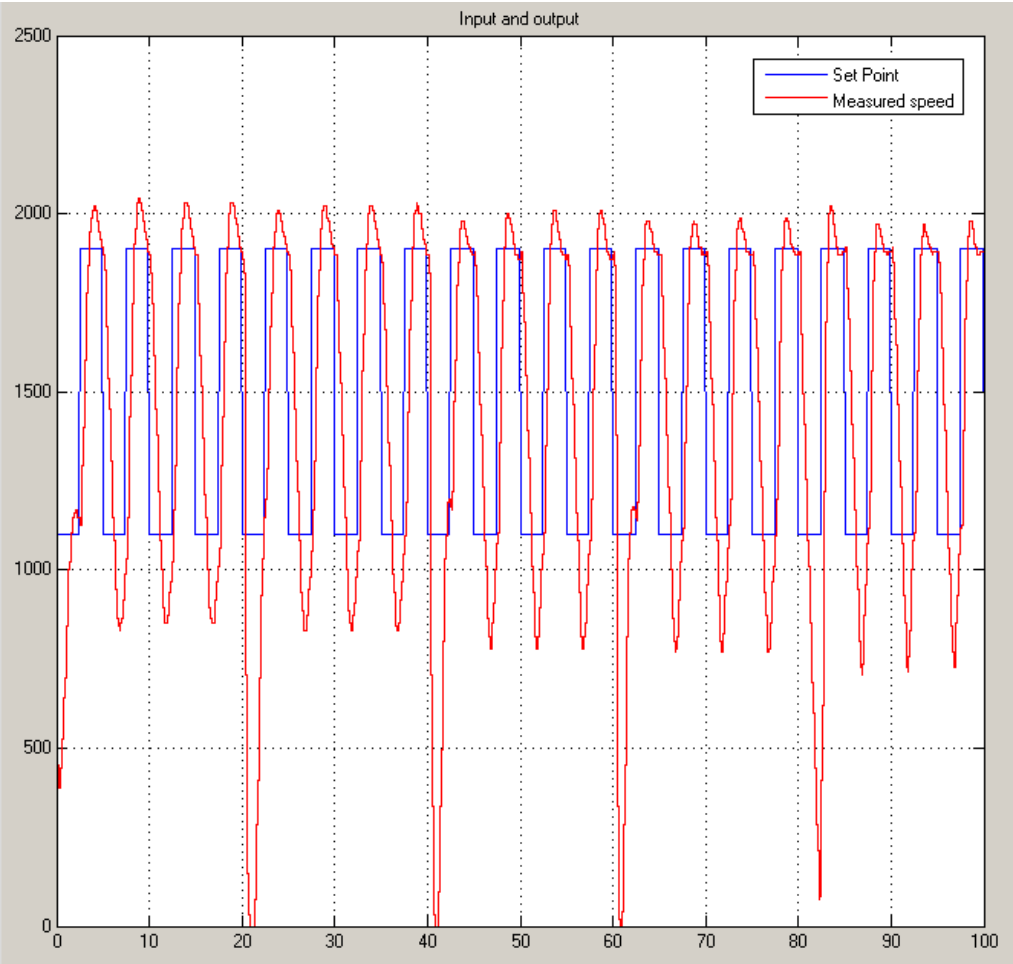


Figure 54 Square wave with variable delays - automatically tuned with resets

It can be seen in the above figure that whenever the controller is reset, it results in spikes.

The system response appears to be better without resets for our case. However, that would vary from system to system.

5 CONCLUSION AND RECOMMENDATIONS

5.1 CONCLUSION

The most important issue in Networked Control Systems (NCS), as discussed before, is of that of delays. These delays can severely affect the performance of a system with dire consequences if the system is being used for a critical purpose.

However, the controller in this project has been designed in such a way that it caters to the problems caused by delays by changing its PID gains.

A change in the PID gains directly affects the transient behavior and steady state response of the system. We have seen the general trend that as the values of the P and I gains are increased, the response of the system becomes more aggressive. At the same time, the stability margin of the system is reduced. The optimal values of the P and I gains is greater for shorter delays and smaller for longer delays.

The controller can never match the performance of a hard wired closed loop system without a network. But this should not be seen as a limitation of a NCS controller. The controller not only ensures that the stability margin is within its limit but also provides the best possible response in the in the presence of network delays.

5.2 RECOMMENDATIONS

The setup that we have built can support further UG or PG projects.

The topic of packet loss can be studied in a future project. The same real-time setup can be used for this purpose. For the purpose of simulations, the TrueTime software discussed above is a very good choice.

We have worked on three different plants that can be almost readily used in future projects about control systems, even if they are not related to NCS.

6 SUPPLEMENTARY

6.1 AERO-PENDULUM PLANT

In this plant, the angular position of the rod an aero-pendulum is controlled. The rod moves with the help of thrust provided by two small propeller fan which use coreless DC motors.

A model for the pendulum with the propeller is shown below.

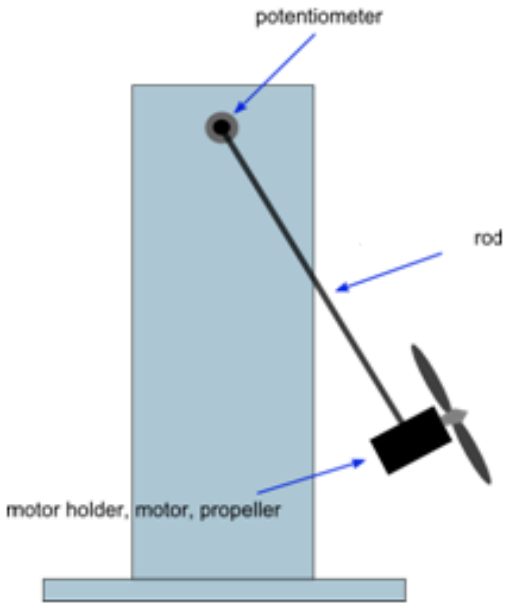


Figure 55 Simple diagram of an aero-pendulum

6.1.1 Mechanical Design

Now, we come to our construction of the aero-pendulum. The base is 1ft x 1ft. The height of the frame is 2.5 feet and the length of the rod is 1.5 ft.

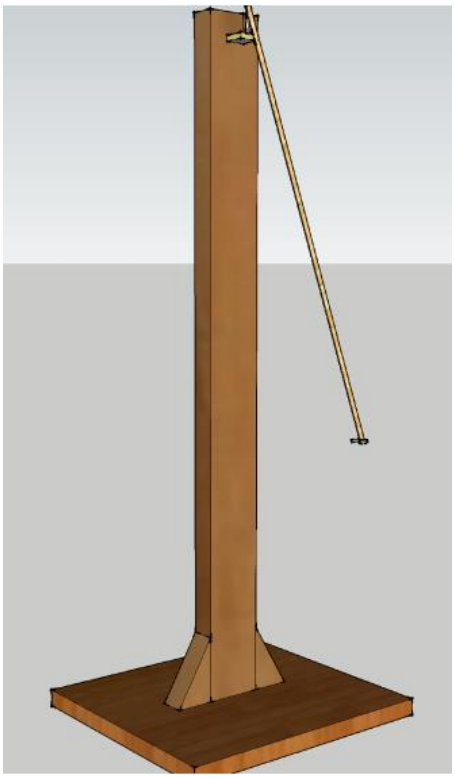


Figure 56 Application diagram of the aero-pendulum

Figure 57 Picture of the aero-pendulum

6.1.2 Development board

This single PCB hosts all of the electronic circuitry needed at the plant side. Each distinct circuit is demarked on the top side of the board. These circuits are:

- Microcontroller
- Ethernet Shield
- Switched drive
- Power supply

Several components used in the Development board have already been discussed in the Speed Control Plant (sections 3.3.2 and 3.3.3). Hence, they need not be discussed again.

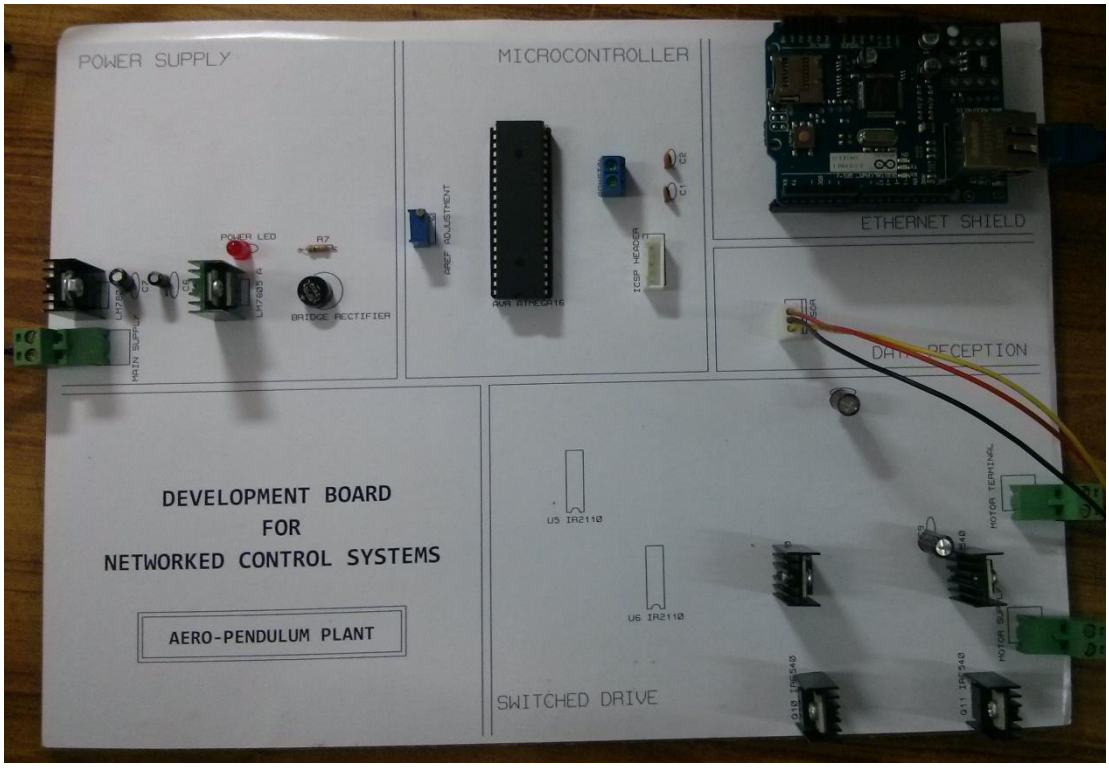


Figure 58 Development board for the aero-pendulum plant

6.1.3 Sensor

The angular displacement is measured with the help of a precision potentiometer. The pendulum rod is connected to the shaft of a multi-turn potentiometer with very little friction.

As the rod is displaced, the shaft of the potentiometer rotates and there it registers a change in voltage.

The change in voltage is used to measure the angular displacement of the rod. The voltage signal from the pot is fed to the Microcontroller ADC which processes the data to determine the change in angular displacement.

6.1.4 Propellers

Two RC Propellers have been used to provide thrust to the pendulum rod. Each Propeller is driven at 3.6 Volts and has a maximum speed of 45000 rpm.

6.1.5 Mathematical Modeling

The equation modeling the forces on the pendulum rod is:

$$mL\frac{d^2\Theta}{dt^2}$$

$$= ku - mgLsin\Theta - c\frac{d\Theta}{dt} \qquad (1)$$

Where m is the mass of the rod, L is the length of the rod and Θ is the angular displacement of the rod.

k is the thrust coefficient, g is the acceleration due to gravity and c is the damping coefficient.

At steady state,

$$\frac{d^2\Theta_{ss}}{dt^2} = \frac{d\Theta_{ss}}{dt} = 0$$

Hence, $sin\Theta_{ss} = \frac{k}{mg}u_{ss}$

And the open-loop transfer function is found to be

$$\frac{\Theta_{ss}}{u_{ss}} = \frac{kL}{mL^2.s^2 + c.s}$$

6.1.6 MATLAB modeling

Using the above modeling, the following model was made in MATLAB/Simulink.

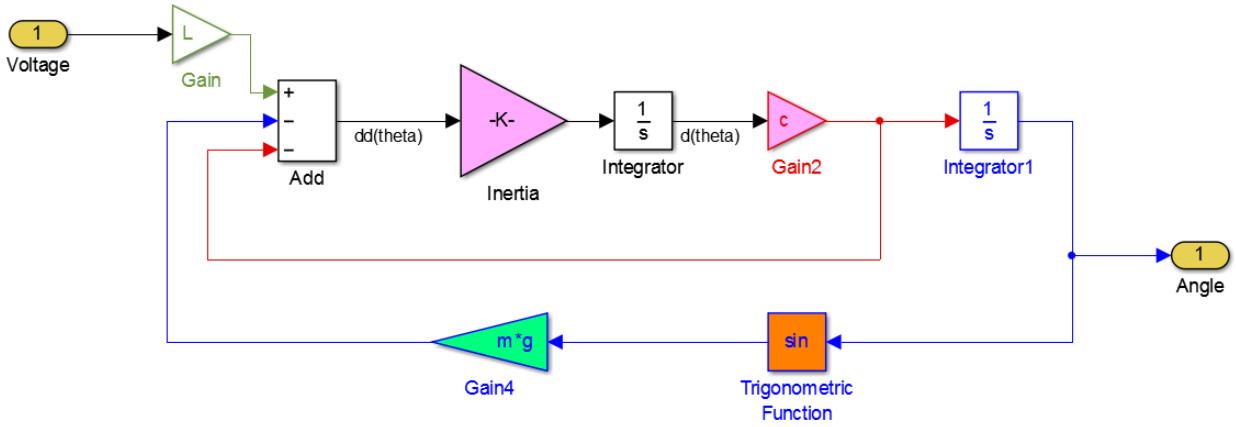


Figure 59 MATLAB model of the pendulum plant

The input signal is Voltage.

The feedback signal is voltage corresponding to the angular position of the potentiometer.

6.1.7 Actuator

An h-bridge based switched drive is used as the actuator. Its schematic is shown below:

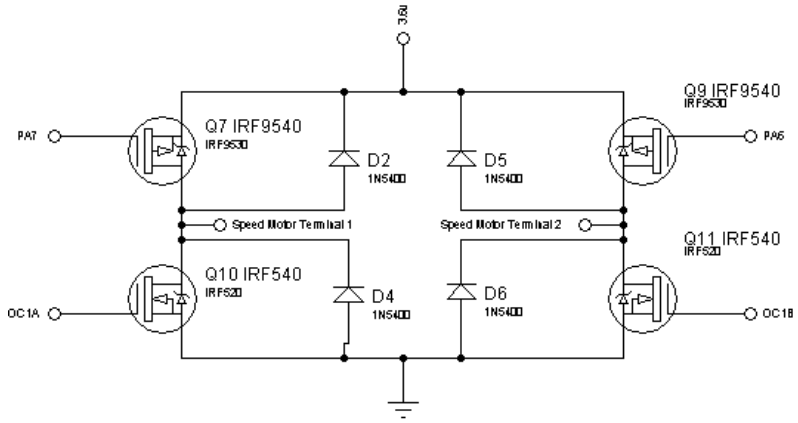


Figure 60 Schematic of the drive for the aero-pendulum's propeller

6.2 POSITION CONTROL PLANT

In this plant, the angular position of the shaft of a permanent magnet DC motor is controlled.

6.2.1 Mechanical design

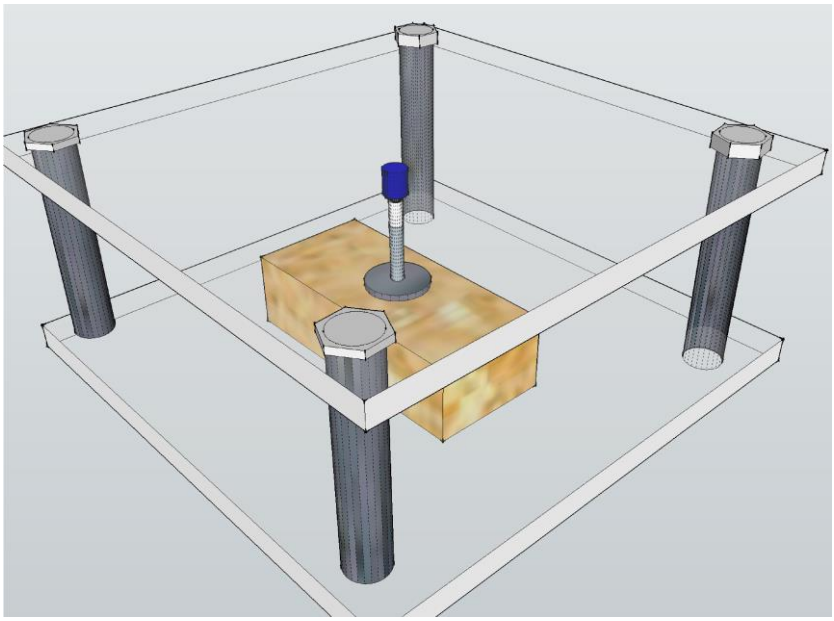


Figure 61 Application diagram of the position control plant

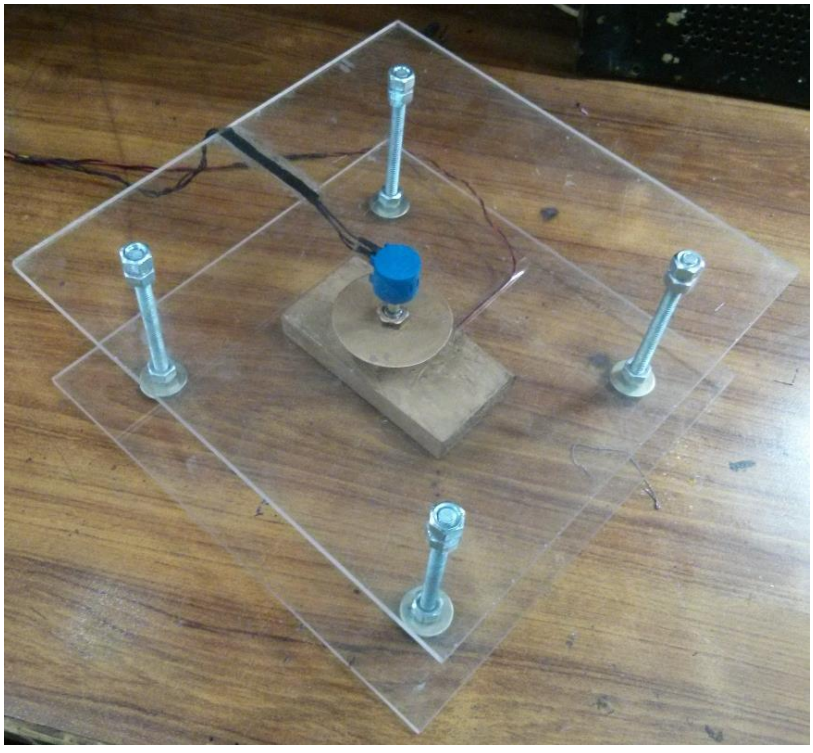


Figure 62 Mechanical structure of the position control plant

6.2.2 Development board

A board has been fabricated for this plant as well, which is similar to the first two boards.

6.2.3 Modelling

The mathematical and MATLAB modelling of the system is the same as that of the speed control plant as discussed in Chapter 3.

6.2.4 Linear Drive

An op-amp based linear drive is to be used for this plant. The drive is tested and it working fine. It consists of two parts. The first one is a digital to analog converter (DAC). The output of the DAC is fed to a MOSFET based linear drive that is controlled by an op-amp using a closed loop feedback system. Moreover, switching MOSFETs are used to change the direction of the motor.

6.2.4.1 DAC

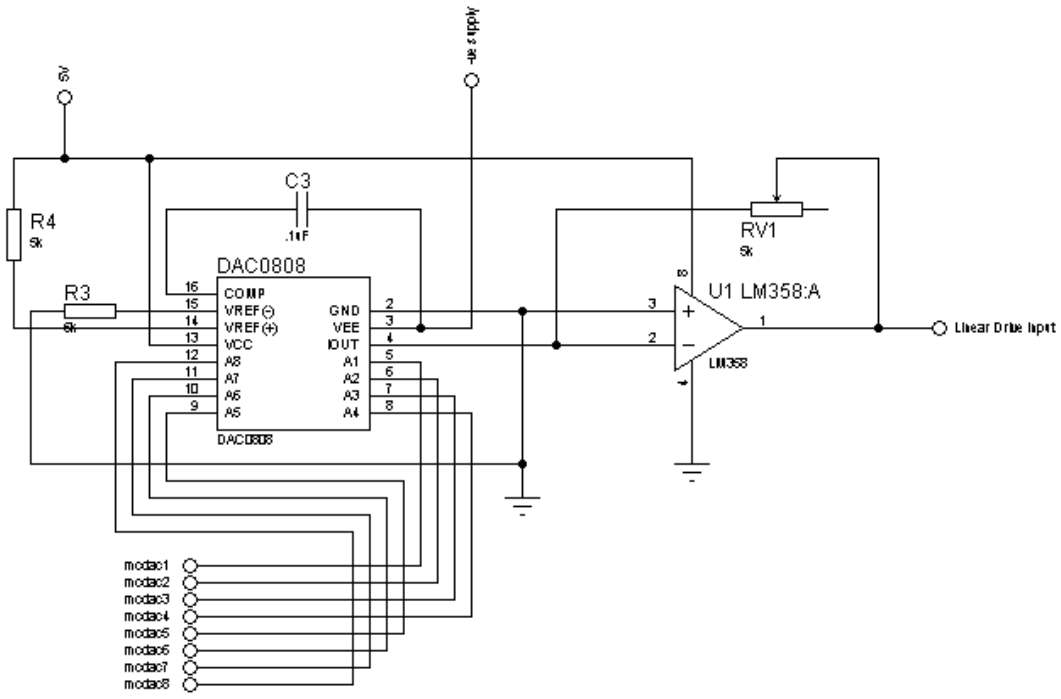


Figure 63 Schematic of the Digital to Analog controller (DAC)

6.2.4.2 Op-amp and MOSFET based regulator

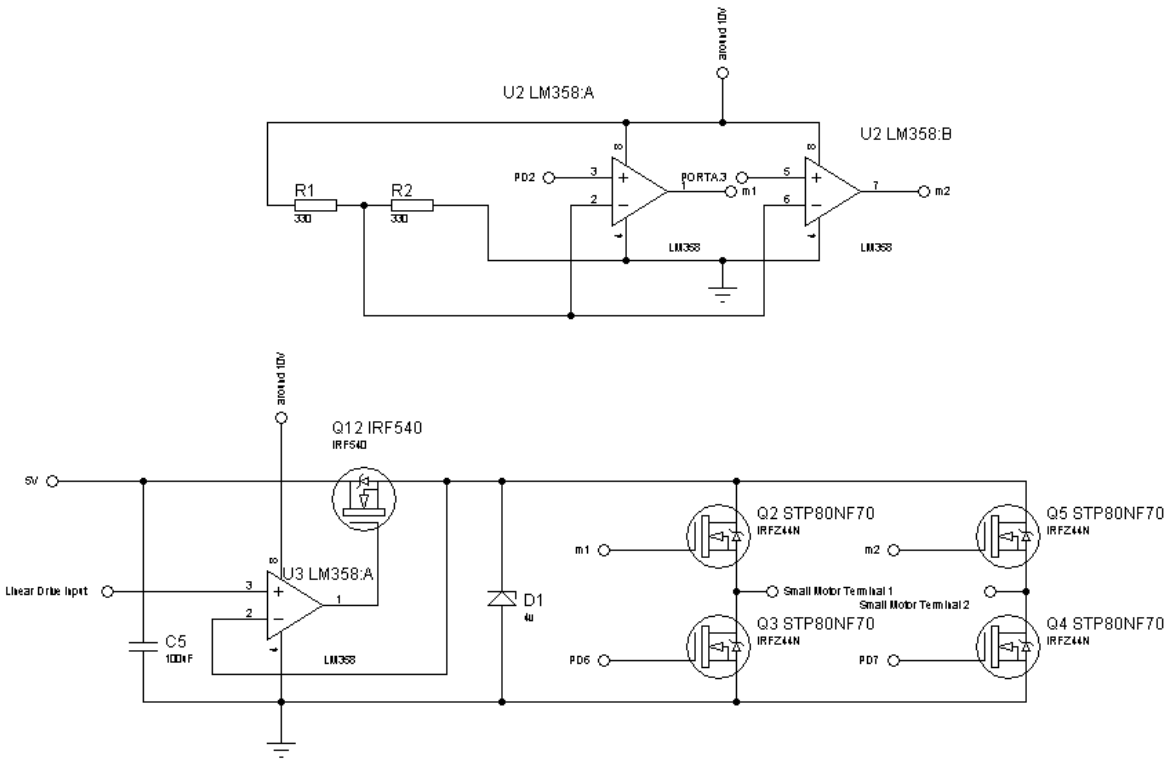


Figure 64 Schematic of linear drive/regulator

6.3 PCB LAYOUT

Layout of the development board is as follows. Some jumpers have also been used which are not shown here.

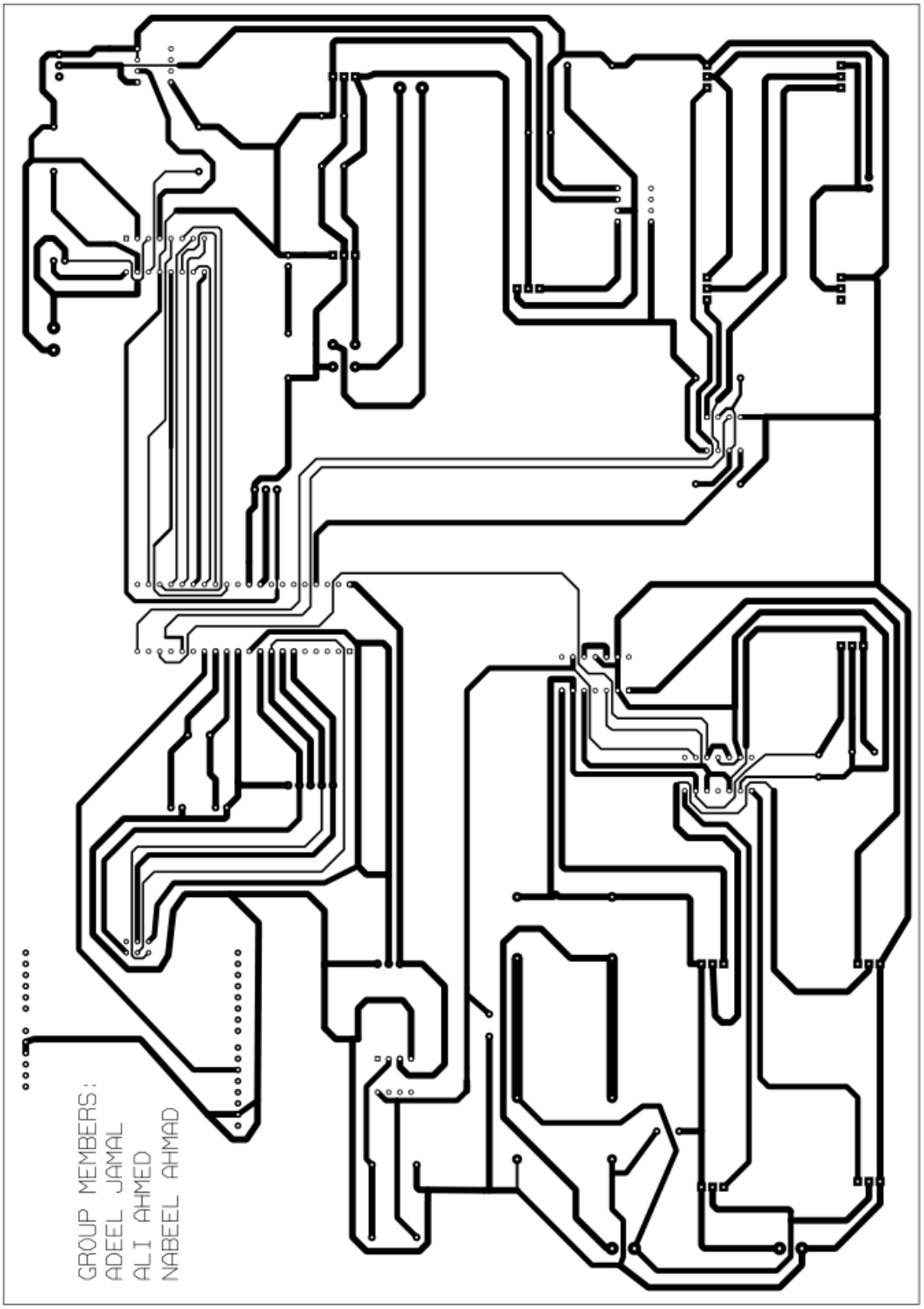


Figure 65 Layout of the PCB – view from the bottom

6.4 GANTT CHART

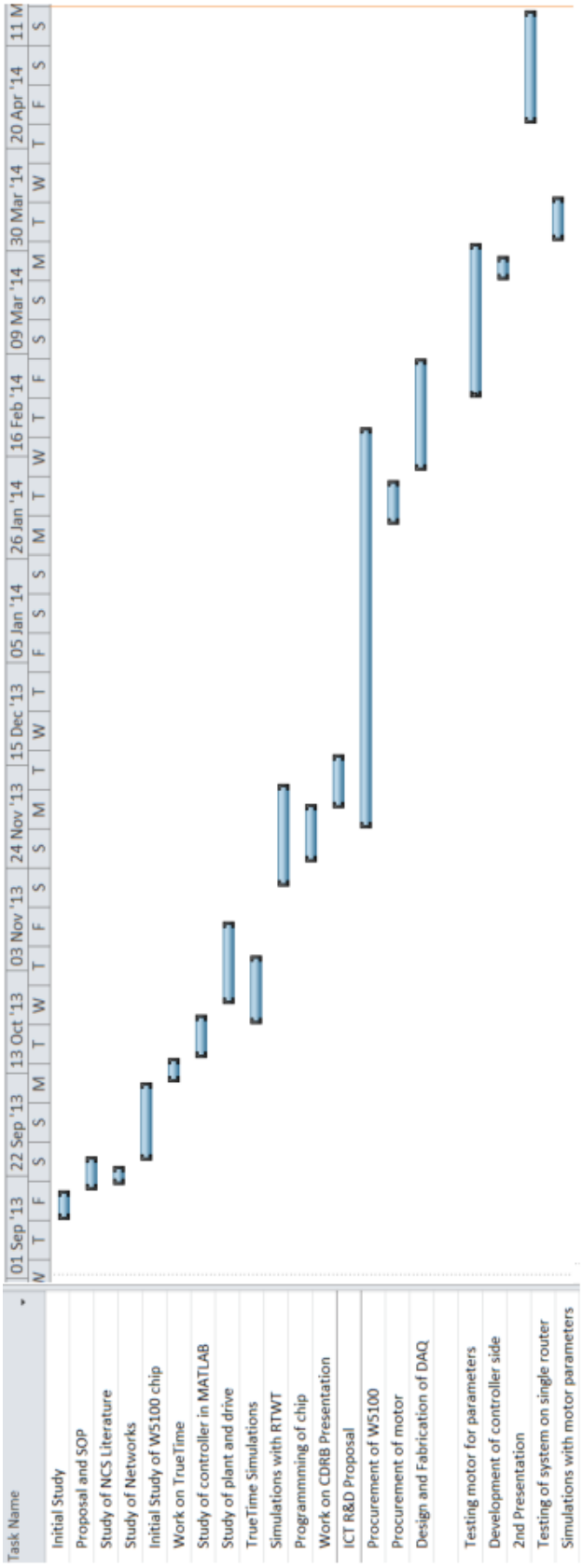


Figure 66 Gantt chart

7 REFERENCES

1) Control Systems Engineering, 6e, Norman S. Nice
2) Computer Networking, 6e, James F. Kurose, Keith W. Ross
3) Control methodologies in networked control systems, Yodyium Tipsuwan, Chow, 2003
4) Gain Scheduling for Networked Control System, A dissertation by Yodyium Tipsuwan, 2003
5) TCP/IP Essentials for Embedded Programmers, Jean J. Labrosse
6) TrueTime: Simulation of Networked and Embedded Control Systems, Anton Cervin
7) Study of Immune PID Networked Control System based on TrueTime, Daogang Peng
8) How Does Control Timing Affect Performance? Analysis and Simulation of Timing Using Jitterbug and TrueTime, Cervin, Henriksson, Lincoln, Eker, and Årzén
9) What is Network Latency and Why Does It Matter? by O3b Networks
10) Control System Cyber Vulnerabilities and Potential mitigation of risk for utilities, 2010, Juniper Networks, Inc.
11) RFC 2131: Dynamic Host Configuration Protocol, IETF, http://tools.ietf.org/html/rfc2131