Project:- Building PC-Based PID Control System

Abstract

The main objective of the control systems I lab project is to build a complete PC-based control system with PID by designing an appropriate speed controller for DC motor where the control system should satisfy the following specifications which are a zero steady state error with the maximum overshoot of 10% and settling time of less than 8 seconds. Moreover, in this project, students will be able to learn how to use both the LabVIEW and DAQ software in order to implement the calibration of measurement and to control signals as well. Furthermore, students will be also able to use the oscilloscope for testing the Pulse Width Modulation Signals (PWM Signals) where the PWM signal will be created in order to generate a digital pulse train in terms of frequency using DAQmx create channel VI by selecting appropriate inputs for duty cycle, frequency, counter(s) , idle state and initial delay where the students will be able to configure the duration of the generated pulses using DAQmx timing VI. Ultimately, students will be able to tune the PID controller gains by interfacing the H-bridge and encoder with DAQ (Data Acquisition) card and LabVIEW software.

Table of Contents

List of Figures

List of Tables

Background

General Information about the DC Motor

First of all, the operation in any electric motor is based on simple electromagnetism. A current-carrying conductor generates a magnetic field; when this is then placed in an external magnetic field, it will experience a force proportional to the current in the conductor, and to the strength of the external magnetic field. The internal configuration of a DC motor is designed to harness the magnetic interaction between a current-carrying conductor and an external magnetic field to generate rotational motion. Every DC motor has six basic parts, axle, rotor (armature), stator, commutator, field magnet(s), and brushes. In most common DC motors, the external magnetic field is produced by high-strength permanent magnets. The stator is the stationary part of the motor, this includes the motor casing, as well as two or more permanent magnet pole pieces. The rotor rotates with respect to the stator. The rotor consists of windings (generally on a core), the windings being electrically connected to the commutator. The geometry of the brushes, commutator contacts, and rotor windings are such that when power is applied, the polarities of the energized winding and the stator magnet(s) are misaligned, and the rotor will rotate until it is almost aligned with the stator's field magnets. As the rotor reaches alignment, the brushes move to the next commutator contacts, and energize the next winding. Given our example two-pole motor, the rotation reverses the direction of current through the rotor winding, leading to a "flip" of the rotor's magnetic field, driving it to continue rotating. In real life, though, DC motors will always have more than two poles (three is a very common number). In particular, this avoids "dead spots" in the commutator. You can imagine how with our example two-pole motor, if the rotor is exactly at the middle of its rotation (perfectly aligned with the field magnets), it will get "stuck" there. Meanwhile, with a two-pole motor, there is a moment where the commutator shorts out the power supply (i.e., both brushes touch both commutator contacts simultaneously). This would be bad for the power supply, waste energy, and damage motor components as well. Yet another disadvantage of such a simple motor is that it would exhibit a high amount of torque "ripple" (the amount of torque it could produce is cyclic with the position of the rotor).

General Information about the H-Bridge

An H bridge is an electronic circuit that enables a voltage to be applied across a load in either direction. These circuits are often used in robotics and other applications to allow DC motors to run forwards and backwards. H bridges are available as integrated circuits, or can be built from discrete components.

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. 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.

Incremental Optical Encoder

Incremental encoder is another technique to track a rotary shaft other than Absolute Encoder, a typical applications that include incremental encoders are computer mouse, CNC machines and robot arm. Incremental encoder produces a series of

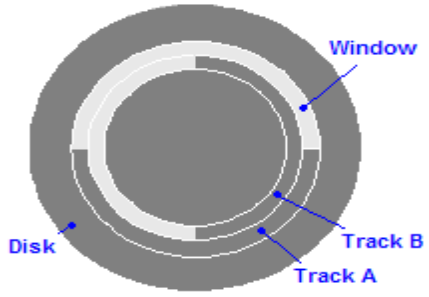

Figure 1: Incremental Optical Encoder

pulses according to the shaft rotation, and by tracking these pulses; we can know the speed, direction and position of the motor drive. You can enhance your project by adding an incremental optical encoder. As shown in above figure 3, Incremental encoder consists of disk with two tracks of slotted windows, and a couple of infrared LED and phototransistors are mounted across each track to produce pulses in response to the movement. To make it easy to illustrate, consider the disk in Fig.1, which has

one window in each track, the angle between each window is 180^{o} and the degree between the two tracks is 90^{o}. Having two tracks help know the rotation direction, as you will see later. To describe the process of determining the speed, direction and position, we will take the encoder disk, straighten it, and draw the waveforms produced by phototransistors upon disk rotation. The most common type of incremental encoder uses two output channels (A and B) to sense position. Using two code tracks with sectors positioned 90 degrees out of phase; the two output channels of the quadrature encoder indicate both position and direction of rotation. If A leads B, for example, the disk is rotating in a clockwise direction. If B leads A, then the disk is rotating in a counter-clockwise direction. By monitoring both the number of pulses and the relative phase of signals A and B, you can track both the position and direction of rotation.

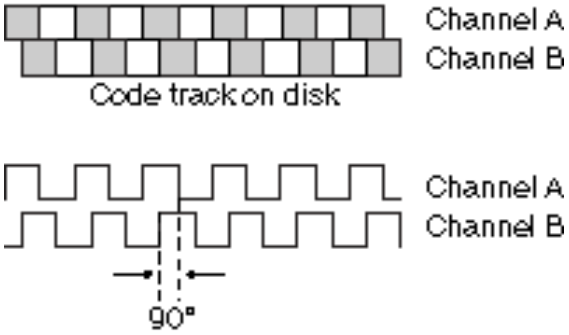

Figure 2:Incremental encoder

Some quadrature encoders also include a third output channel, called a zero or index or reference signal, which supplies a single pulse per revolution. This single pulse is used for precise determination of a reference position. There are multiple ways to determine the angular velocity and acceleration of a Quadrature Encoder. Count the number of Quadrature Encoder pulses in a fixed time interval to estimate the velocity and acceleration of the encoder, Figure 5 below demonstrates this procedure. This method is appropriate for high speed applications.

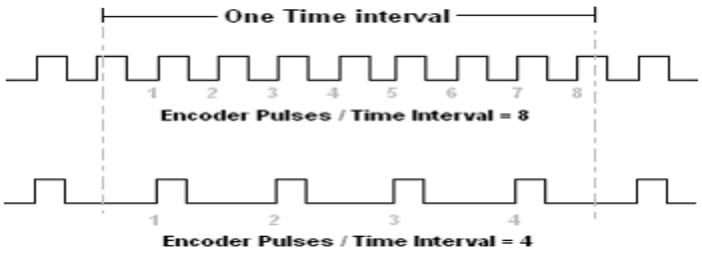

Figure 3 velocity estimation

Once the number of pulses in a fixed time interval is measured the angular velocity of the Quadrature Encoder can be calculated using the following formula:

$$Velocity = \frac{\frac{Encoder\ Pulses}{Pulses\ Per\ \mathrm{Re}volution} \times \frac{60\,\mathrm{sec}}{1\,\mathrm{min}}}{Fixed\ Time\ Interval\ (\mathrm{sec})}\ (RPM)$$

Equation 1

Where, "Encoder Pulses" is the number of quadrature encoder pulses received in the Fixed Time Interval.

Acceleration is the rate of change of velocity. The following formula can be used to estimate acceleration of the Quadrature Encoder:

$$Acceleration = \frac{\left(\frac{Encoder\ Pulses_{n} - Encoder\ Pulses_{n-1}}{Pulses\ Per\ \mathrm{Re}volution}\right)}{(Fixed\ Time\ Interval)^{2}\ (\mathrm{sec}^{2})} \times \left(\frac{60\,\mathrm{sec}}{1\,\mathrm{min}}\right)^{2}\ (RPM^{2})$$

Equation 2

Where, the numerator divided by one Fixed Time Interval represents the change in velocity. Dividing the change in velocity by one Fixed Time Interval gives the acceleration.

General Information about NI DAQ

Designed for performance, NI data acquisition devices provide high-performance I/O, industry-leading technologies, and software-driven productivity gains for your application. With patented hardware and software technologies, National Instruments offers a wide-spectrum of PC-based measurement and control solutions that deliver the flexibility and performance that your application demands. For more than 25 years, National Instruments has served as more than just an instrument vendor, but as a trusted advisor to engineers and scientists around the world.

High-Performance I/O Measurement accuracy is arguably one of the most important considerations in designing any data acquisition application. Yet equally important is the overall performance of the system, including I/O sampling rates, throughput, and latency. For most engineers and scientists, sacrificing accuracy for throughput performance or sampling rate for resolution is not an option. National Instruments wide selection of PC-based data acquisition devices have set the standard for accuracy, performance, and ease-of-use from PCI to PXI and USB to wireless.

High-Accuracy Design

Many scientists and engineers mistakenly evaluate DAQ device error by just considering the bit resolution of the DAQ device. However, the error dictated by the device resolution, or quantization error, might account for only a very small amount of the total error in your measurement result. Other types of errors, such as temperature drift, offset, gain, and non-linearity can vary drastically by hardware design. Through years of experience, NI has developed several key technologies to minimize these errors and maximize the absolute accuracy of your measurements.

Easy Sensor Connectivity with Integrated Signal Conditioning

Traditionally, measuring sensors required separate front-end signal conditioning systems cabled to a data acquisition system. New technologies and miniaturization have enabled the integration of sensor-specific signal conditioning and analog to digital conversion on the same device. NI DAQ devices with integrated signal conditioning deliver higher-accuracy measurements by eliminating error-prone cabling and connectors and reduce the number of components in a measurement system. NI has also partnered leading sensor vendors to provide easy, tool-free sensor connectivity and automatic sensor configuration with TEDS technology.

PWM Signal with NI DAQmx

Pulse width modulation (PWM) is a technique in which a series of digital pulses is used to control an analog circuit. The length and frequency of these pulses determines the total power delivered to the circuit. PWM signals are most commonly used to control DC motors, but have many other applications ranging from controlling valves or pumps to adjusting the brightness of an LED.

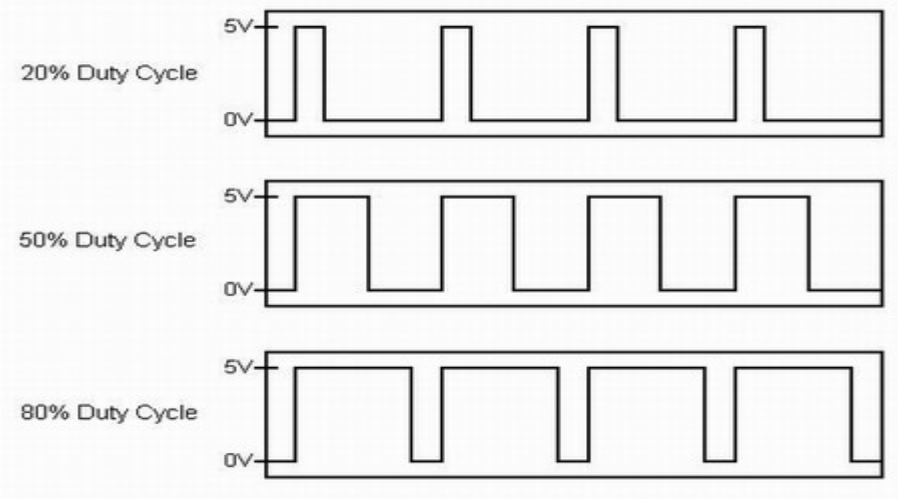

Figure 4: PWM Signals with Different Duty Cycles

The digital pulse train that makes up a PWM signal has a fixed frequency and varies the pulse width to alter the average power of the signal. The ratio of the pulse width to the period is referred to as the duty cycle of the signal. For example, if a PWM signal has a 10 ms period and its pulses are 2 ms long, that signal is said to have a 20 percent duty cycle. Figure 1 shows three PWM signals with different duty cycles. For a more thorough look into pulse width modulation, visit the link at the bottom of this page. PWM signals can be generated as a digital signal, using counters or digital output line(s), or as an analog signal, using for instance, an arbitrary waveform generator or an RF signal generator. Several National Instruments multifunction data acquisition (DAQ) devices are capable of producing PWM signals. This includes the E, S, B, M and the new X Series devices, which represent the next generation of multifunction DAQ from National Instruments. In addition, NI 660x counter/timer devices, as well as NI 653x digital devices can also be used. It's worth noting that National Instruments hardware and software also supports generation of PWM signals using digital line(s) or analog output options. NI LabVIEW is the graphical development environment for creating flexible and scalable test, measurement, and control applications rapidly and at minimal cost. With LabVIEW, engineers and scientists interface with real-world signals, analyze data for meaningful information, and share results and applications. The LabVIEW graphical development environment, combined with NI-DAQmx, gives you the tools needed to easily construct applications using counters to generate PWM signals.

In this lab, we needed the following equipment and materials:

Equipment and materials:

* DC motor (12Vdc).
* H-Bridge.
* Incremental Optical Encoder.
* LabVIEW 2010 software with Control Toolkit.
* NI DAQ system with termination box.

Procedures

PART A: Design Requirements

First of all, in this part of this experiment, we have to do the following steps:

- First of all, we have to design an appropriate speed controller for DC motor where the system should have zero error at steady state, maximum overshoot of 10% with a settling time of less than 8 second as shown in the figure 7 below.

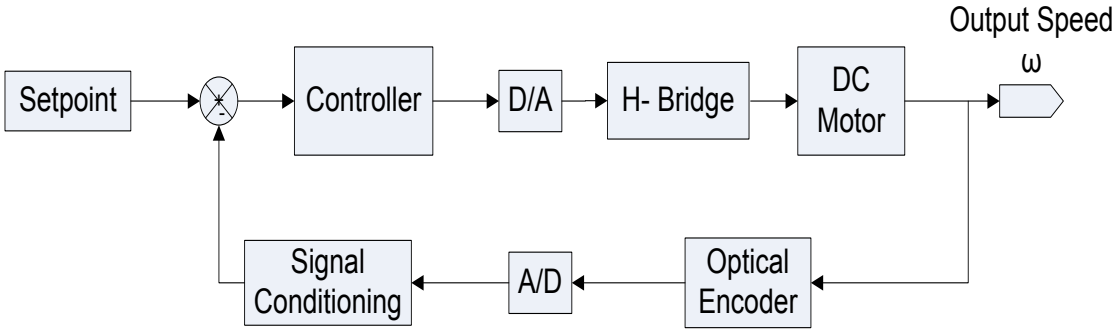

Figure 7: System Block Diagram

- Secondly, we have to start with the open loop system in order to make building the blocks easy. After that, we have to add a controller by introducing the feedback to the system in order to make our block diagram similar to the above diagram according to the basic negative feedback system.

PART B: Building LabVIEW Program

First of all, we connected our DC motor using a constant 12 Vdc power supply in order to start building the LABVIEW program. Therefore, to start working on LabVIEW software, we have to follow firstly the step of creating a PWM signal in order to produce a simple digital pulse train where the block diagram shown in the figure 2 below was built using LabVIEW to program a counter on an M Series device to create a digital pulse train signal using NI-DAQmx where each step of the program, indicated by a number at the bottom of figure 8, as mentioned below in figure 2:

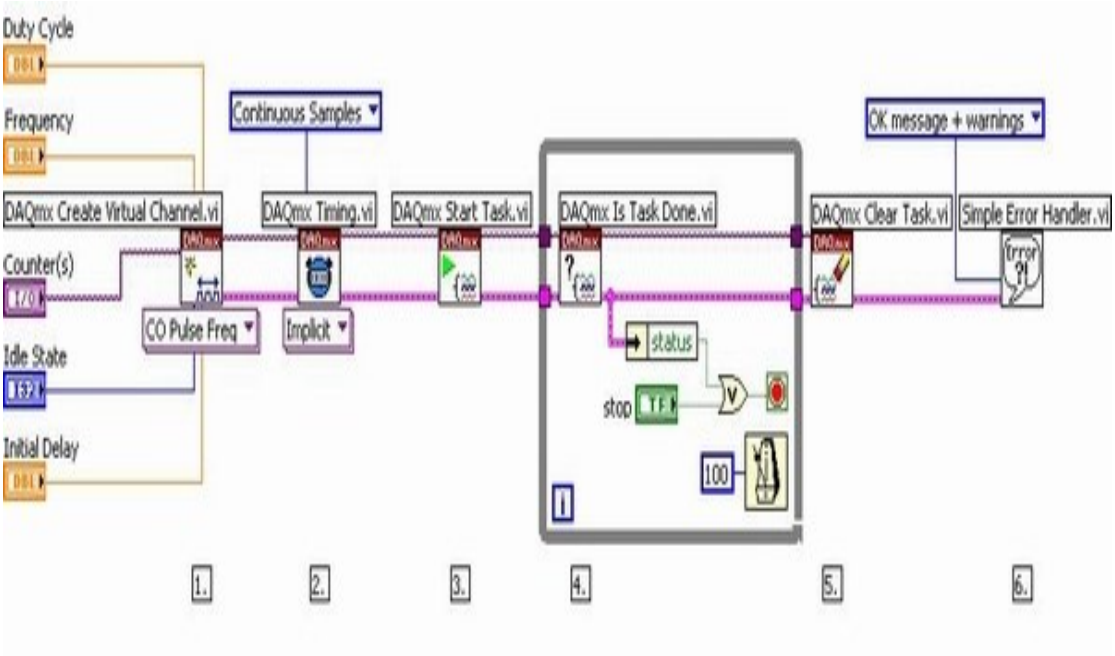

Figure 8: Creating the PMW Signal

- Furthermore, we have to follow the following steps in order to build our labVIEW program:

1. *We have to create and initialize firstly a counter output channel to produce a pulse in terms of frequency using the DAQmx Create Channel VI. Then, we select appropriate inputs for Duty Cycle, Frequency, Counter(s), Idle State and Initial Delay.*
2. *After that, we have to use the DAQmx Timing VI to configure the duration of the pulse generation. Then, we should use the implicit instance when no sample timing is needed, such as in counter tasks like pulse train generation. Furthermore, we have to select Continuous as the sample mode.*

3. *Moreover, we have to call the DAQmx Start VI in order to make VI starting the pulse train generation.*
4. *Then, we have also to loop continuously until the user presses the Stop button. After that, we have to check for errors using the DAQmx Is Task Done VI. After that, we have to add a Wait until Next Millisecond Multiple VI in order to introduce a small time delay to the loop to prevent the loop from executing as fast as possible, consuming unnecessary processor resources.*
5. *Also, we have to call the DAQmx Clear Task VI to clear the task.*
6. *Ultimately, we have to check for and display errors using the Simple Error Handler VI.*

Ultimately, using a local variable and a case structure can change the duty cycle dynamically. In order to determine this change in the duty cycle, we have to check every iteration of the loop for the duty cycle control. If it has been changed, the program sets the new duty cycle using DAQmx Write VI. On the contrary, if it has nor not been changed, the false case is being executed without updating anything.

H-Bridge

First of all, we connected the circuit with DC motor as shown in figure 9 below where the red chip is called H-Bridge. Furthermore, our H-Bridge consists of four switches (solid-state or mechanical) from S1 to S4 where as shown in figure 10 below S1 and S2 are closed as well as S2 and S3 are open. The operation of this bridge is in such a way that based on the pressing of diagonal switches, counter clockwise rotation is obtained and vice versa as shown in figure 11 and table 1 in the next page.

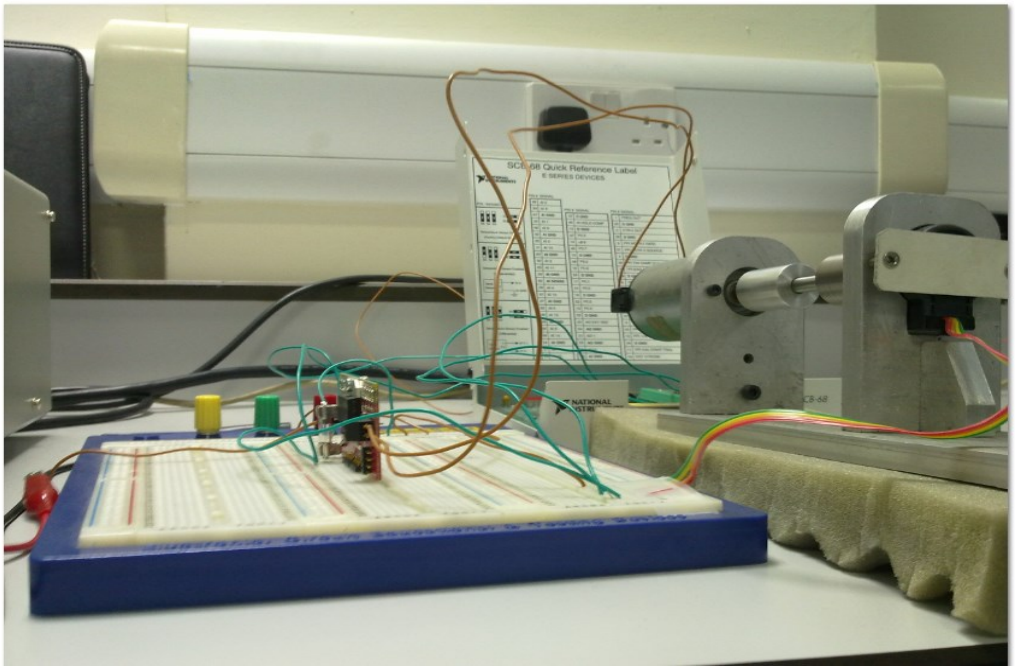

Figure 9: Circuit Connections and H-Bridge

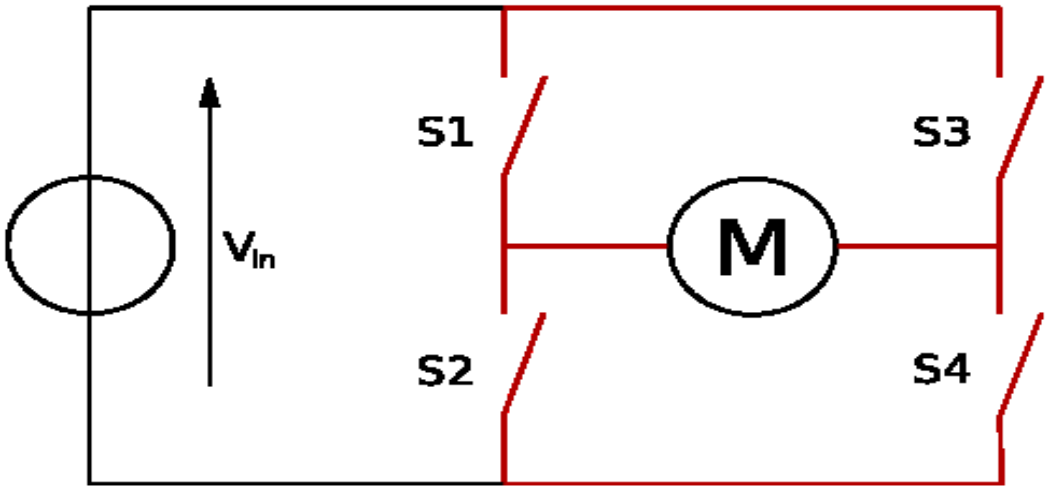

Figure 10:H-Bridge Diagram

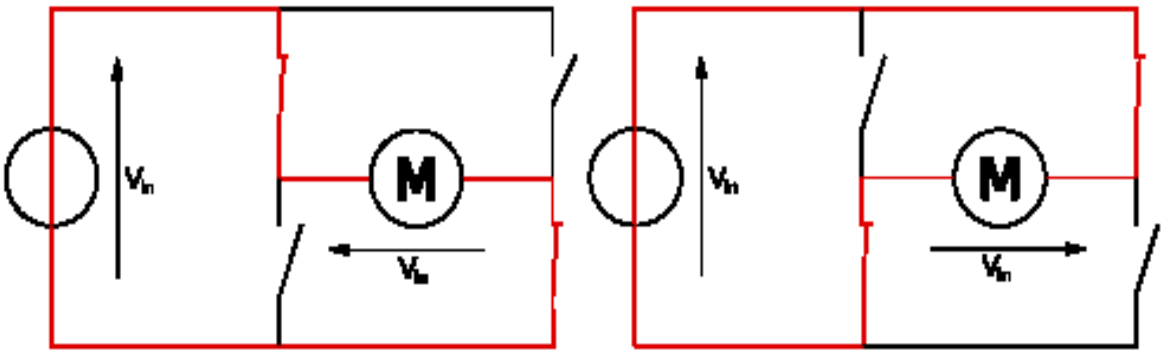

Figure 11: H-Bridge operation

S1	S2	S3	S4	Result
1	0	0	1	Motor moves right
0	1	1	0	Motor moves left
0	0	0	0	Motor free runs
0	1	0	1	Motor brakes
1	0	1	0	Motor brakes

Table 1: H-Bridge operation

In this project experiment, we used a PMW signal in order to control the average power based on duty cycle. For instance, if our duty cycle is 50% (or 0.5), it means that it is ON for 50% of the time and OFF for 50% of the time.

At the end, after building the block diagram for the PMW signal using the LabVIEW software; we connected our internal encoder, the dc motor, and the H-Bridge as shown in figure 12 below.

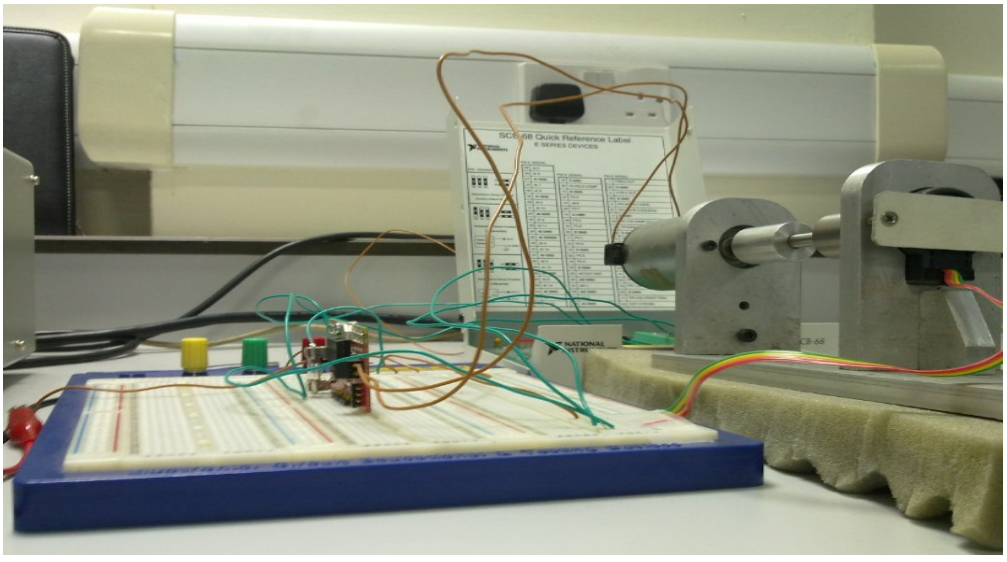

Figure 12: Encoder, DC Motor and H-Bridge Components Connection

Furthermore, the internal encoder have four terminals as shown in figure 13 below, which will be connected to the DAQ. Then, we connected two terminals to the DAQ counter 1 and we connected the other two terminals to +5Vdc. The incremental encoder uses two output channels (A and B) to sense position. After that, using two code tracks with sectors positioned 90 degrees out of phase; the two output channels of the quadratic encoder indicate both position and direction of rotation. If A leads B, for instance, the disk is rotating in a clockwise direction. If B leads A, then the disk will be rotating in a counter-clockwise direction. Moreover, we monitored both the number of pulses and the relative phase of signals A and B, to be able to track both the position and direction of rotation. Ultimately, after connecting the components, we used labVIEW to design a control system for the DC motor. Then, from the figure 8, we can notice the speed of the dc motor as the duty cycle of the PMW changes. The DAQ is connected to the internal encoder to read the speed where the speed read is converted to rad/sec by multiplying by two pi (3.14) and dividing by 60. The first division by 1000 in the block diagram represents the resolution of the encoder. With this block diagram the open loop response of the system can be observed. Table 2 mentions the response of the system with different values of the duty cycle of the PWM, and figure 14 shows the open loop system response.

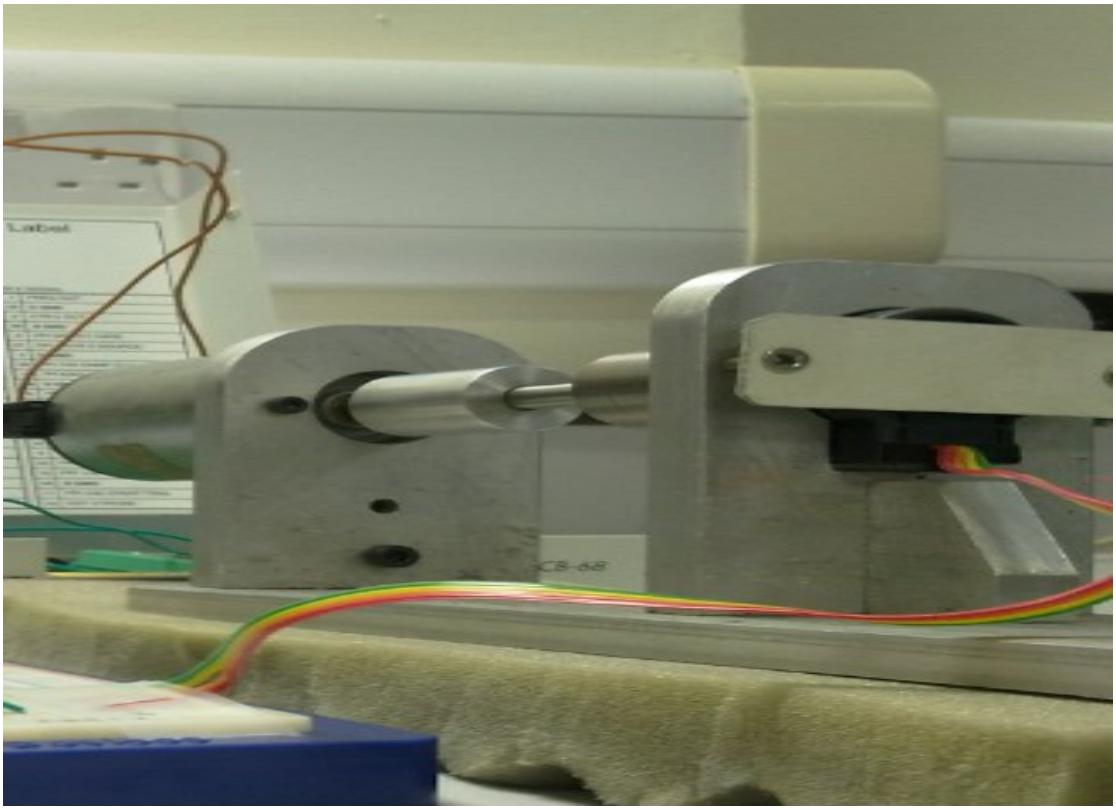

Figure 13: Encoder Connection

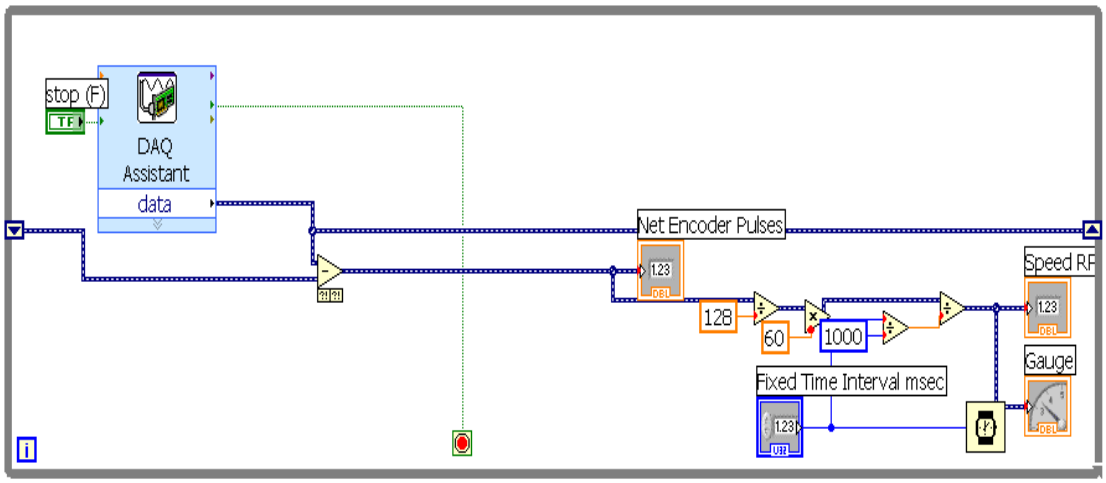

Figure 14: Open Loop, Motor Speed measurement block diagram

Duty Cycle	*Speed (RPM)*	*Speed (Rad/Sec)*
0.3	*375*	*39.01*
0.6	*1079*	*113.1*
0.7	*1285*	*135.0*
0.8	*1968*	*206.1*

Table 2: Changing the speed with changing the duty cycle

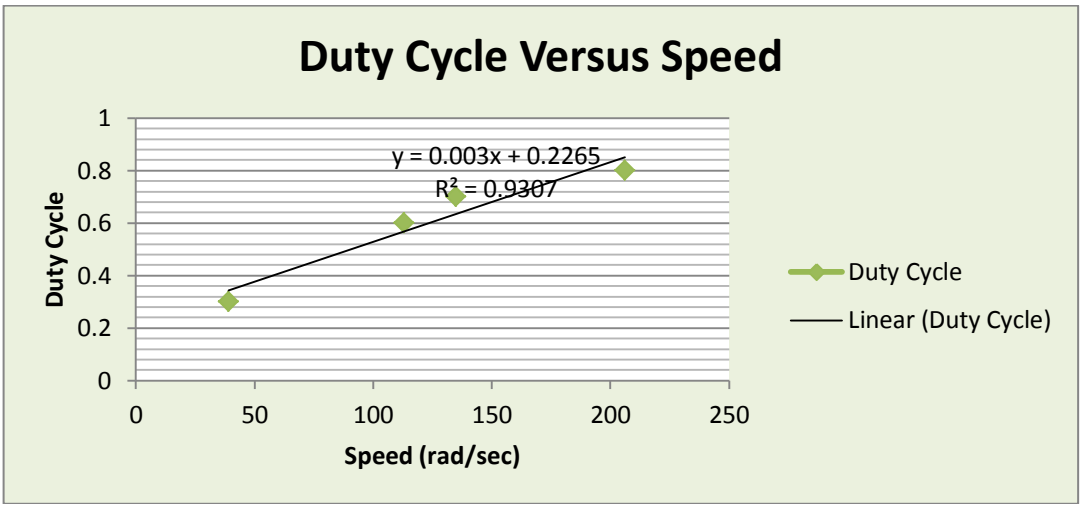

Figure 15: Open Loop System Response

First of all, we normalized the speed of the motor after building the circuit as shown in above figures by testing the motor for maximum speed by making duty cycle of 0.9. Then, a constant was introduced in the software along with the divider block in order to divide by this maximum speed. Furthermore, we designed a controller that satisfies the specification; zero steady state error, Maximum overshoot of 10%. Settling time of less than 8 seconds as shown in the figure 10 in the next page where it shows our designed closed loop system using LabVIEW block diagram with feedback.

Furthermore, the speed of the motor in rad/sec is filtered first using a moving average filter of order 2 and then fed to the controller input after normalizing. The filter is needed to make the signal smoother. The case structure is used as a reference set point to test the closed loop system. Moreover, if the structure is true the reference is 140 rad/sec and if it is false the reference changes to 90 rad/sec. This way the system oscillating between these two values can be observed. The output of the case structure is normalized by dividing by the highest value which is 21 and then fed to controller. Then the output of the controller is fed to the input by replacing the duty cycle with the output of the controller to connect the two circuits and make a closed loop control. After that the output filtered signal and the reference signal are merged and sent to waveform chart to observe the behavior of the system as well as to tune the controller to achieve the required specifications. Ultimately, we tuned the controller until we got the desired output where we observed our system by choosing low proportional gain in order to get a very low system response and steady state error as well. Then, we used the integral control gain by tuning both the proportional and integral gains by trial and error in order to get a zero steady state error and high rise time.

At the end, The proportional, integral and derivative settings were adjusted as follows to obtain the desired output, where Ti and Td are in minutes:

KP = 1.000

Ti = 0.090

Td = 0.000

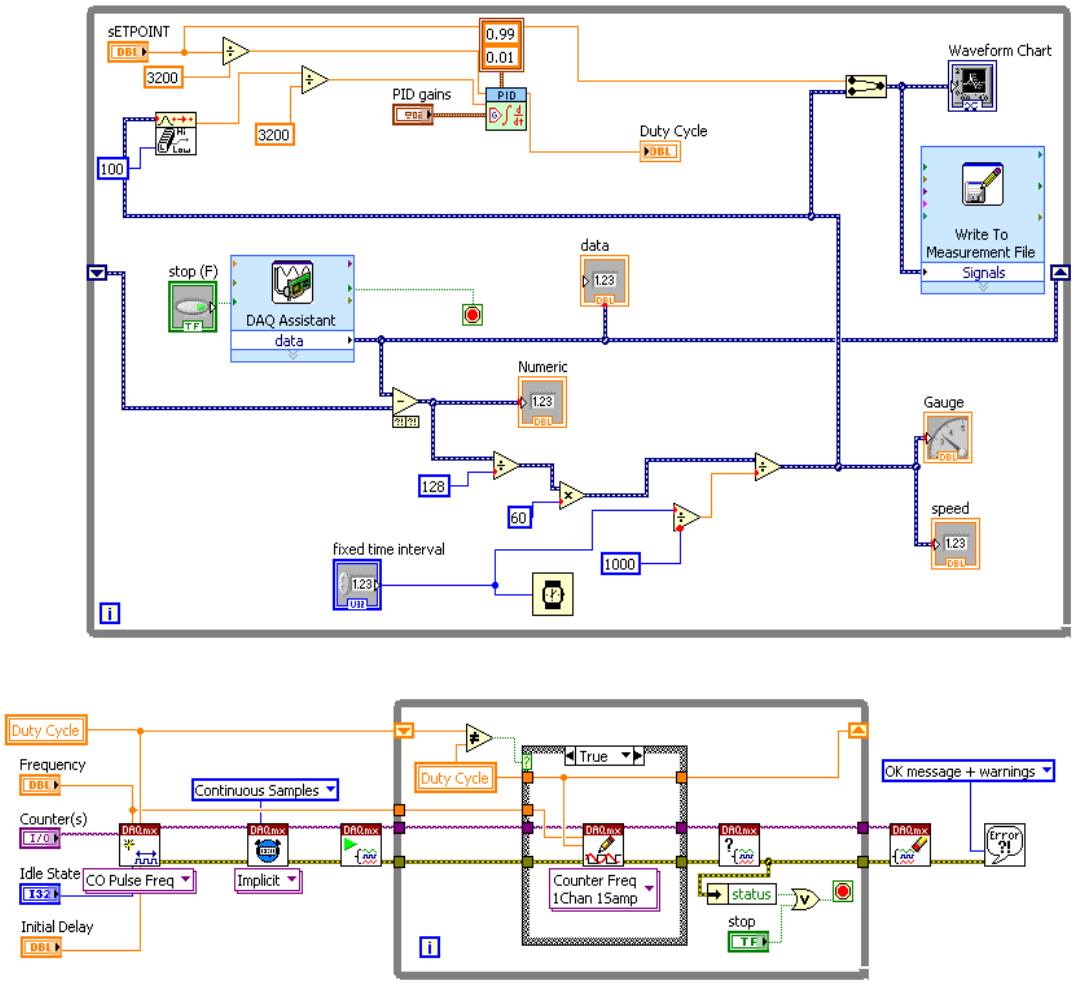

Figure 16: LabVIEW Block Diagram for Closed Loop System

Figure 20 shows the system response graphically using the data acquired in the next page.

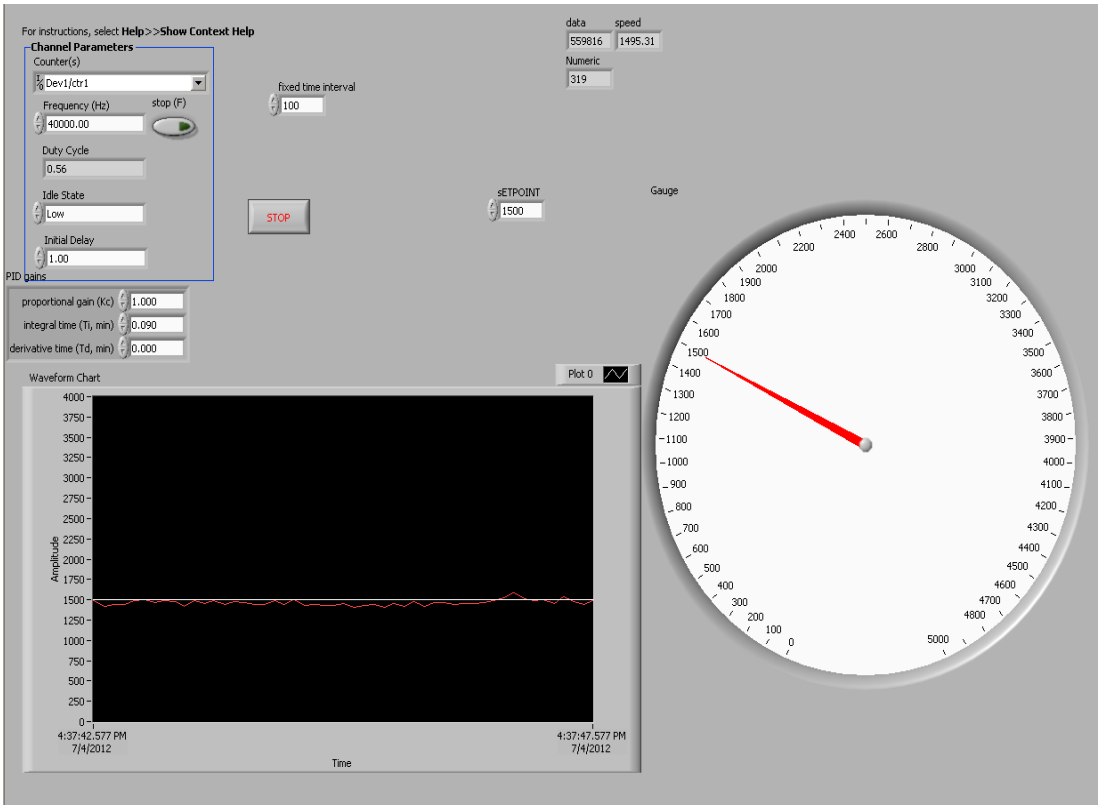

Figure 17: System Response

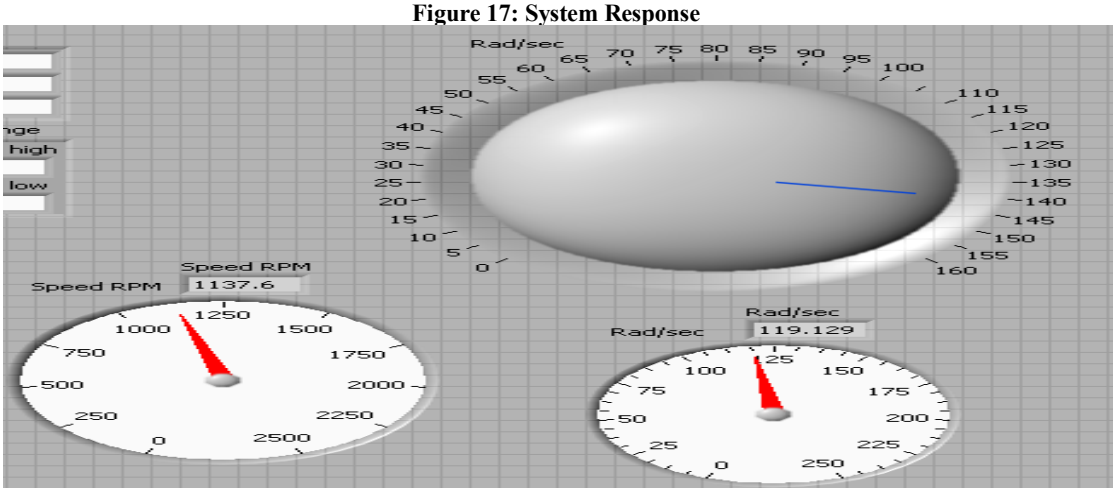

Figure 18: System Response with Different Reference Value

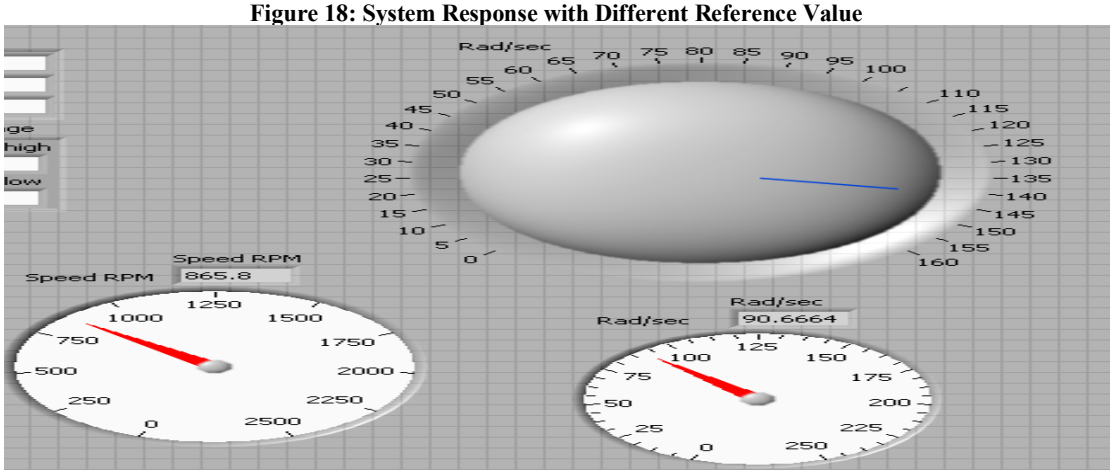

Figure 19: System Response with Another Different Reference Value

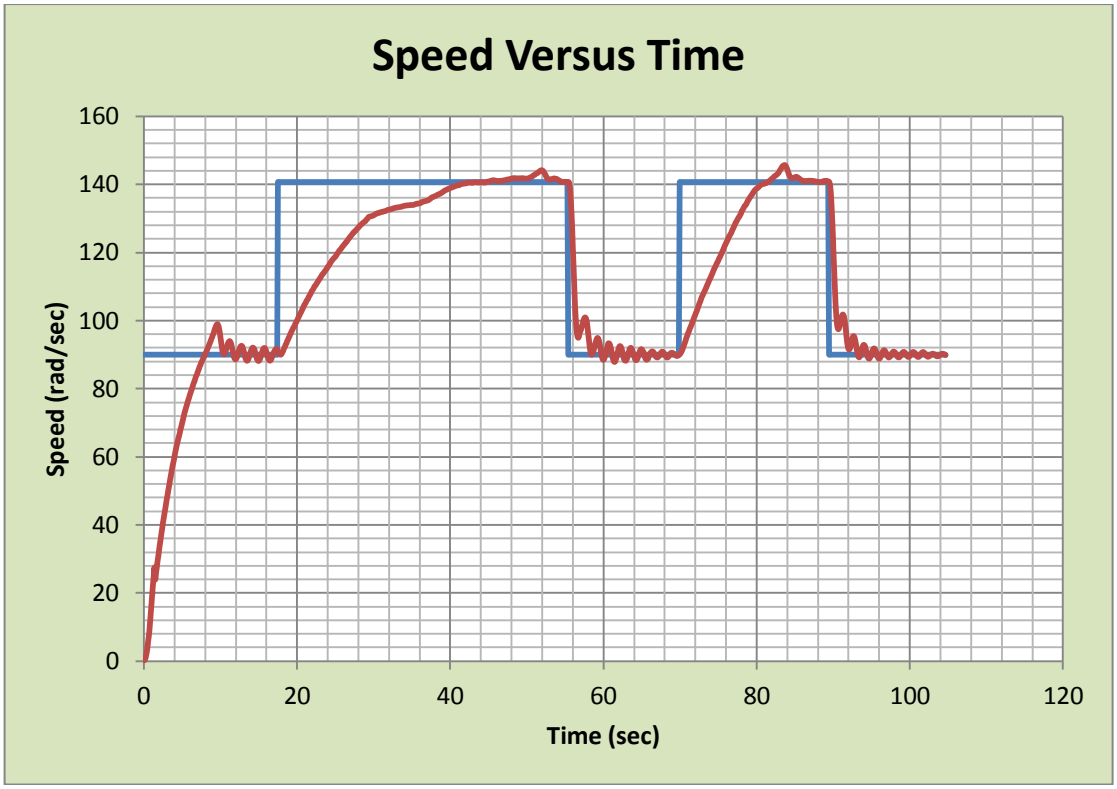

Figure 20: The Graphical System Response

From the above figure, The red plot in figure 14 shows the actual speed of the motor when the PI controller were used and the blue plot shows the reference speed. From the figure we can see that the steady state error is about zero and the response to sudden changes is fast.

Conclusion & Discussion

The main aim of this project is to control the speed of an H-bridge motor using a template (or schematic) for the PWM generation. As start, we started by enhancing our knowledge of the LABVIEW pogram though using a testig signal sent by the function generator and detect it by the LABVIEW. We also used the osciliscope to view the signals and observe the results. After enhancing our skills with the LABVIEW program, we started our project. The criteria that was given is zero steady state, less than 10% overshoot and a setling time less than 8 seconds. Furthermore, we used the incremental encoder with two output channel in order to convert the signal coming from the motor to a suitable format for the computer and also convert the signal coming from the computer to one that is in a suitable fromat. As a result, LABVIEW showed us the response of the motor by changing the setpoint. It showed the calculated speed respictive to time. After doing all the connections and testing the connectivity, we started tunning the PID controller. Different combinations where tried and we got the optimal output. In conclusion, the project was very helpful in explain the functionality of the PID controller and how does it react to different gains. Finally, it also improved our skills using the LABVIEW. Moreover, we were introduced to the concept PWM generation and we were understood how it controls the analog motor through the duty cycles. At the end, the project objectives were achived.

References

Mr. El-Tahir, W. (2012). *Control Systems I Lab LabVIEW I handout.* Retrieved on July 5th, 2012, from Blackboard Website: https://iLearn.aus.edu.

Mr. El-Tahir, W. (2012). *Control Systems I Lab LabVIEW II handout.* Retrieved on July 5th, 2012, from Blackboard Website: https://iLearn.aus.edu.

Mr. El-Tahir, W. (2012). *Control Systems I Lab LabVIEW III handout.* Retrieved on July 5th, 2012, from Blackboard Website: https://iLearn.aus.edu.

Mr. El-Tahir, W. (2012). *Control Systems I Lab LabVIEW IV handout.* Retrieved on July 5th, 2012, from Blackboard Website: https://iLearn.aus.edu.

Mr. El-Tahir, W. (2012). *Control Systems I Lab LabVIEW V handout.* Retrieved on July 5th, 2012, from Blackboard Website: https://iLearn.aus.edu.

Retrieved on July 7th, 2012, from: http://en.wikipedia.org/wiki/H_bridge.