



# PRACTICAL IMPLEMENTATION OF CASCADE CONTROL APPROACH WITH PI CONTROLLERS TO CONTROL THE SPEED OF A PERMANENT MAGNET DC MOTOR

Ismaeil R. Alnaab

Department of Polymers and Petrochemicals Engineering, Oil and Gas Engineering College, Basra University for Oil and Gas, Basrah, Iraq

E-Mail: [Ismaeil.r.alnaab@buog.edu.iq](mailto:Ismaeil.r.alnaab@buog.edu.iq)

## ABSTRACT

An experimental investigation was conducted to control the speed of a permanent magnet brushed direct current motor under cascade control approach. The investigation dealt with studying the performance of the motor under open loop, current loop and speed loop conditions. The results then compared to similar investigations done by researchers. The investigation was implemented on software and hardware models with the help of MATLAB Simulink, MPLAB software and LabVIEW. PMDC motor mechanical and electrical characteristics were studied first, then they were implemented in MATLAB Simulink. A cascade control model was designed in Simulink environment and linked with the motor via H-Bridge simulation model. The cascade control has the following features, two Proportional Integral (PI) controllers, one for speed loop and the other for current loop, Integrator windup feature and some limiters. The performance of the speed control was studied successfully and motor speed was under control. The effect of tuning PI controller gains was observed. Six important tasks were focused on in this project which are Unipolar PWM generation, speed and current measurements, Integrator windup, speed transit and speed profile. The real time implementation was a success and some of the measurements were compared with a simulation model. MPLAB software was used to write the codes in C language for the cascade control loops.

**Keywords:** PMDC Motor, cascade control, LabVIEW, MATLAB simulink, encoder, tacho generator, Kp and Ki gains.

## 1. INTRODUCTION

Rotating electrical machinery is used in the majority of modern industrial equipment, whether it is a simple toy or a complicated electronic device in a missile, they all depend on a proper functioning of rotating electrical equipment. Direct current (DC) motor is a rotating electrical equipment commonly used in wide range of industrial applications and portable home appliances [1]. It is a device which converts electrical energy into mechanical energy. Therefore, controlling these types of motors is a crucial aspect that needs to be considered. There are many different approaches of controlling DC motors, such as the on-off method, pulse width modulation, back-electromotive-force (EMF) sensing and Cascade control. Through these control methods, it is possible to determine whether it is beneficial to use a particular motor for a specific application.

The aim of this project is to control the speed of a permanent magnet brushed DC motor using a cascade control approach. Speed control means intentional change of the drive speed to a value required for performing the

specific work process [2]. In other words, the purpose of speed control is to allow the speed to maintain its level when extra load torque is being added on the motor shaft. Speed control can be done either manually by the operator or by some sort of electronic device (controller). For this project, the controller is a proportional integral (PI) controller. PI-controllers are the most common form of feedback [3]. They are used in a wide range of industrial applications especially in control loops as they are similar to processors yet they are used for small applications. PI-controllers are generally used to eliminate the steady state error which is induced by the proportional controller. However, they have a negative impact on the overall stability of the system and the speed of response [4]. As the name (Proportional Integral Controller) suggests, the PI algorithm consists of two basic modes, Proportional mode and Integral mode. Therefore, it has two gains  $k_p$  and  $k_i$ . These two gains have impact on system response characteristics as shown in Table-1 below.

**Table-1.**  $k_p$  and  $k_i$  impact on system response characteristics[5][6].

Parameter	Rise time	Overshoot	Settling time	Steady-state error
Increase $K_p$	Decrease	Increase	Small Change	Decrease
Increase $K_i$	Decrease	Increase	Increase	Increase

$K_p$  and  $k_i$  values can be obtained and tuned using different techniques such as Ziegler-Nichols, the

particle swarm optimization, bode plot and trial and error [7][8].

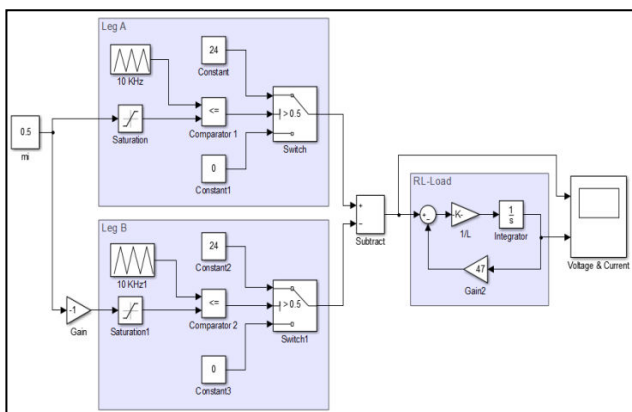


By reviewing paper [9], H. Tun and W. Aung present a similar study on a 24 volt PM brushed DC motor. In their study, they focused on a few features that a flexible inverter can provide in order to control the previous mentioned motor. In addition to that, they used the developed inverter to control brushless DC motor with PWM technique and three phase sine wave generator. They also mentioned controlling the motor with a PI controller to obtain suitable and steady responses of current and speed. However they did not focused on how to present speed and current responses and the effect of tuning the PI gains on them, as well as they never made any actual speed profile, so that they demonstrate the reliability of their work. Therefore, this paper take more steps forward and demonstrates the actual effect of PI gains on both PI controllers used in the outer and inner loops of the cascade system.

Section II shows the MATLAB simulation of open loop. Whereas section III explains the cascaded control of dc motor. Then in section IV, MATLAB simulation of inner current loop and outer speed loop is presented. After that in section V the real time implementation of inner current loop and outer speed loop is demonstrated. And finally in section VI. Speed measurement, response and profile is illustrated.

## 2. MATLAB SIMULATION OF OPEN LOOP

The H-bridge Leg A and Leg B can be represented in MATLAB Simulink as shown in Figure-1 below. The modulation index can be a constant number, 0.5 for example will give 50 % duty cycle. Each leg contains carrier frequency of 10 KHz with a magnitude varies between -1 and 1. The saturation block is to limit the modulation index within arrange of -1 to 1. The two constant blocks in each leg represent the output voltage that will supply the RL-load. The comparator is used to compare between the carrier signal and modulation index for each leg alone and the switch will do the output voltage arrangement which can be either 0 or 24 volt.



**Figure-1.** H-bridge Leg A and Leg B representation in MATLAB Simulink.

The four operational modes were explained in section (3.3), and can be summarized as follow:

**Mode A:** Leg A switch reads 0 and Leg B switch reads 0, the subtract block will produce zero volt.

**Mode B:** Leg A switch reads 24 and Leg B switch reads 0, the subtract block will produce 24 volt.

**Mode C:** Leg A switch reads 24 and Leg B switch reads 24, the subtract block will produce 0 volt.

**Mode D:** Leg A switch reads 24 and Leg B switch reads 0, the subtract block will produce 24 volt.

The RL-Load is a simple block contains a resistor and an inductor.

$$V = R * i + L \frac{di}{dt} \quad (1)$$

$$di/dt = (V - R * i)/L \quad (2)$$

Where:

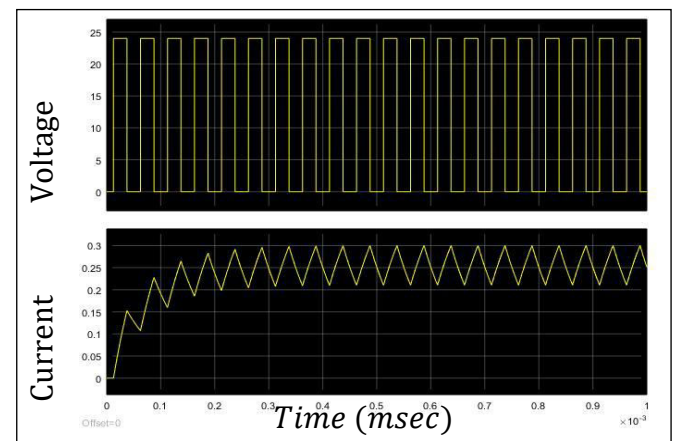
V is the RL – Load input voltage

R = 47  $\Omega$ , L = 3.3 mH,

I = current passes through the resistor,

di/dt = instantaneous rate of current change (A/sec).

The voltage and current waveforms were measured at duty cycle equals to 50%, and they are shown in Figure-2 below. For the same duty cycle, the open loop was tested in real time with RL-Load. The RL-load is shown in Figure-3. The modulation index was adjusted using the potentiometer R22 to set the duty cycle at 50%.



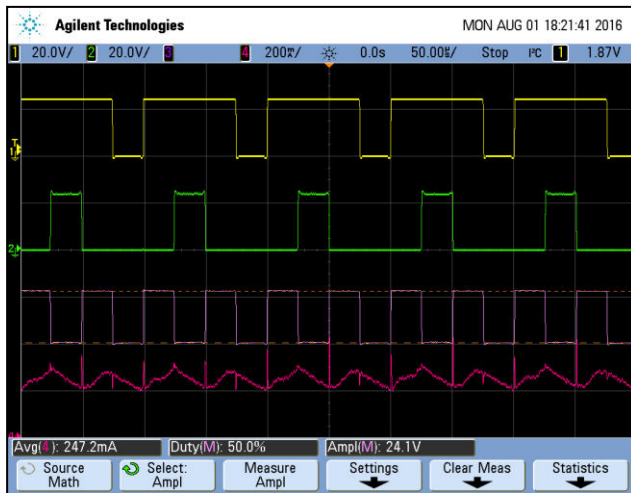
**Figure-2.** Voltage and current waveforms measured at 50% duty cycle.



**Figure-3.** RL-Load.



Figure-4 shows the output voltage waveform of each leg  $V_\alpha$  and  $V_\beta$  so as  $V_\alpha$  and current waveform for the real time implementation.

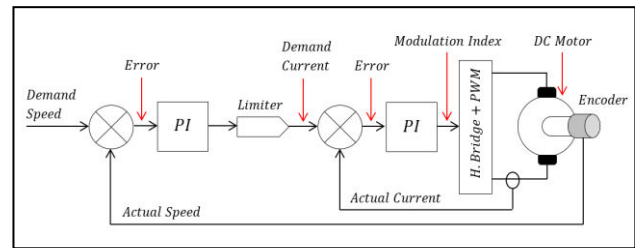


**Figure-4.** The output voltage waveform of each leg  $V_\alpha$  and  $V_\beta$  so as  $V_\alpha$  and current waveform for the real time implementation.

It is clear that MATLAB results are slightly different to real time measurements due to the percent of tolerance that the electric components have.

## 2. CASCADE CONTROL OF DC MOTOR

Cascade control is the most effective speed control of DC motors [7][10]. It is used to obtain fast output responses and high current control, as well as being suitable for low and high capacity motors [11]. Moreover, “it has a better set point tracking, better disturbance rejection and less delay time and phase lag” [12][13]. Cascade control approach consists of two loops, inner current loop and outer speed loop. The inner current loop consists of a dc motor, H-Bridge converter and proportional integral controller. The outer speed loop contains the inner current loop and another proportional integral controller. Cascade control block diagram is shown in Figure-5 below.



The motor has an encoder (speed sensor) attached to it and is used to measure the actual speed. The actual speed will be subtracted from a given demand speed to give an error which will be inserted into the outer-speed-PI controller. The output of the speed-PI controller represents the demand current. This demand current has to be limited within a specific range (motor rated current). After that it will be inserted into a summation block where the actual (measured) current will be subtracted from the demand current to give an error which will be inserted into the inner-current-PI controller. The actual current can be measured using a current sensor connected to one of the motor phases (Phase A). The output of the current-PI controller represents the modulation index which will be inserted into the PWM circuit and H-bridge converter to produce the armature voltage that will supply the motor. Within the cascade control approach, speed can be controlled so that any change happening in load torque will not affect the amount of speed produced. Cascade control will be implemented in MATLAB Simulink as well as in real time on a DC motor test rig. The programming language that was used to program the controller is C language.

## 3. MATLAB SIMULATION OF INNER CURRENT LOOP AND OUTER SPEED LOOP

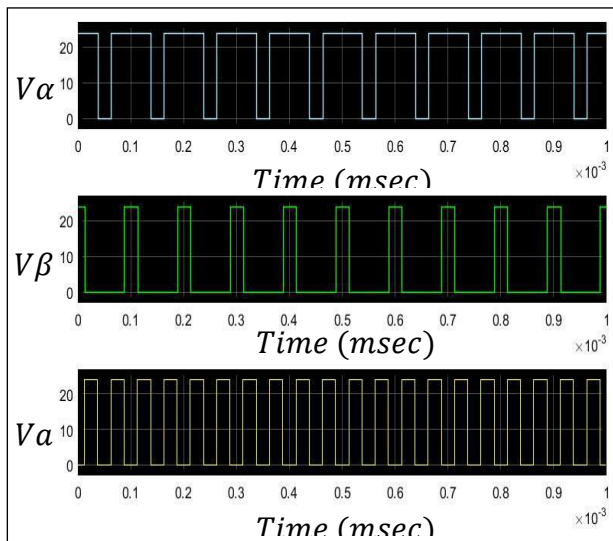
### A. Inner Current Loop

The simulation model of inner current loop contains a DC motor, H-Bridge and a PI microcontroller. The H-Bridge model is a part from what was shown in Figure-1. It contains two legs leg A and B, and each leg has one carrier signal generator block, one comparator block, one saturation block, two constant blocks and one switch block. The purpose of each block is described in Table-2 below:

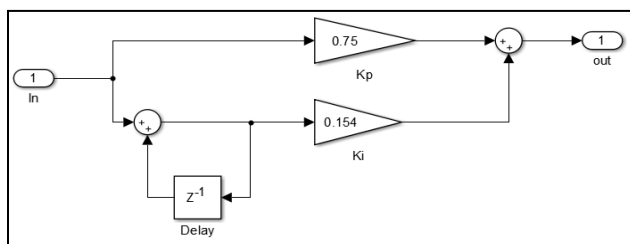
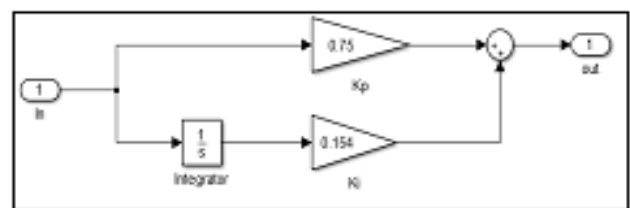
**Table-2.** Blocks and their purpose.

Block name	Purpose
Carrier signal	Produces triangle carrier signals with output values varies between 1 and -1 and a frequency of 10 KHz.
Saturation	To limit the modulation index within the range (1 upper limit and -1 lower limit)
Comparator	To compare the two signals which are (carrier signal and modulation index)
Switch	To switch between the two constant blocks which are 24 volts or zero volt depending on the comparator signal. Each time the comparator read modulation index signal is larger than carrier signal the switch closes on 24 volt, and if the modulation index is less than carrier signal the switch closes on zero volt.
Constant	Provides a constant value for example 0 volt DC or 24 volt DC.

The output of each switch is PWM pulse with a magnitude which varies between 0 and 24 volts. The voltage that will supply the motor is the subtraction of the two PWM pulses as shown in the Figure-6 below.

**Figure-6.** The output voltage waveform of each leg  $V_\alpha$  and  $V_\beta$  so as  $V_\alpha$  which is the armature voltage that supplies the motor.

The PI microcontroller model is shown in Figures (7) and (8) below.

**Figure-7.** PI microcontroller model.**Figure-8.** Continuous time representation of PI controller.

It contains two summation blocks, two gains ( $K_p$  and  $K_i$ ) and one delay block. This PI model is a digital model and can be represented by the following discrete equations:

$$Out(n) = k_p * \varepsilon(n) + k_i * x(n) \quad (3)$$

$$x(n+1) = x(n) + \varepsilon(n) \quad (4)$$

Where  $\varepsilon$  is the error input to the controller.

$k_p$  is the proportional gain = 0.75

$k_i$  is the integral gain = 0.154

The  $k_p$  and  $k_i$  were obtained using trial and error method.

The following equation represents the action of the controller:

$$Out = k_p * \varepsilon + k_i \int \varepsilon dt \quad (5)$$

The full simulation of inner current loop is shown in Figure-9 below:

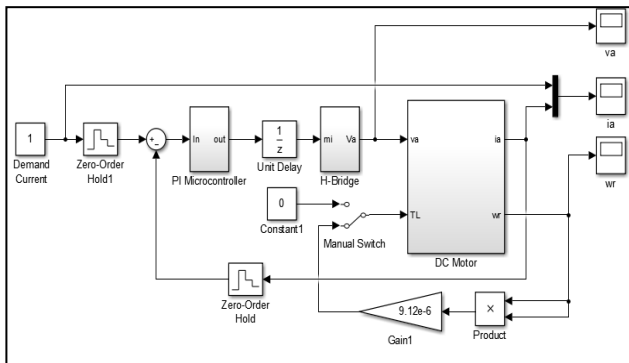


Figure-9. The full simulation of inner current loop.

The zero holders contain sampling time which is equal to 100e-6, 1/10 KHz, the purposes of these zero holders are:

- Hold its input ( $\omega, ia$ ) for a specified sample period.
- Convert its input signal ( $\omega, ia$ ) to a discrete time signal.
- Synchronizes the actual current and the demand current with the PWM carrier signal.

The Unit delay is used to hold and delay its input for a specified sample period and represents the PWM cycle. Load torque can be inserted in to the motor in two ways, either by a constant block or by a loop subjected to the following formula:

$$TL = km * \omega^2 \quad (6)$$

Where  $km = 9.12 * 10^{-6} \text{ Nm s}^2$

This torque represents a load coupled to the motor shaft, for example a fan. The error that was inserted to the PI microcontroller is equal to: Demand current - Actual current ( $ia$ ). By supplying a demand current equal to 1, the following current response was achieved, Figure-10 below. The actual armature current (Green response) has an overshoot less than 18% with a settling time equals to 0.001, fast transient response.

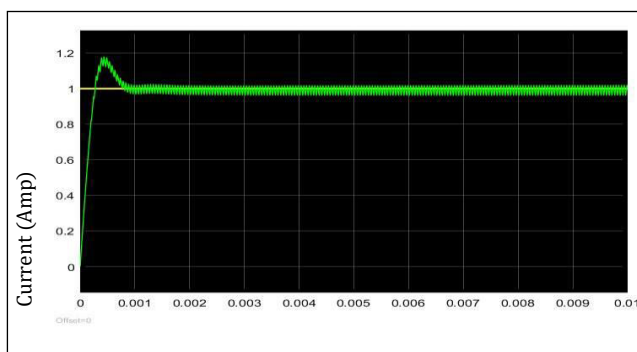


Figure-10. Output current response at demand current equals to 1 Ampere.

To check the effect of adding load torque  $TL$  on speed response, the following procedure was conducted:

The current demand was set to be 3 and the  $TL$  switched to zero. After the simulation was run, a load torque  $TL$  has been added using the manual switch at time equaled to 0.12 second, the effect of load torque on current and speed is shown in Figures (11) and (12) respectively. Since there was no speed control, the speed has been reduced and the current increased according to the following equation:

$$J \frac{d\omega}{dt} = Te - TL - B\omega \quad (7)$$

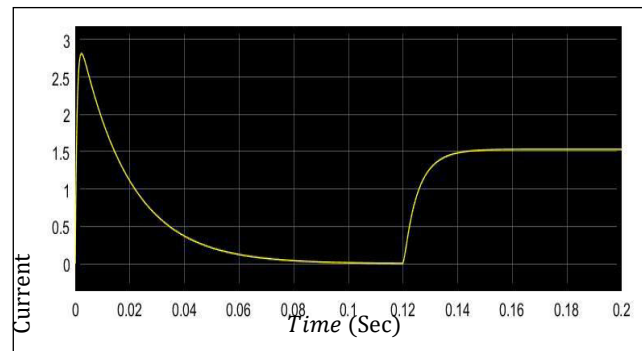


Figure-11. The effect of load torque on current.

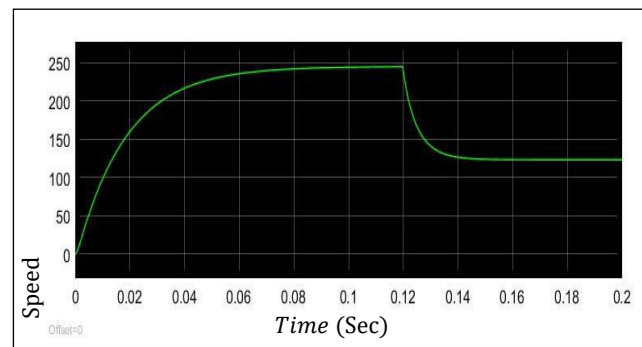


Figure-12. The effect of load torque on speed.

The main objective of the inner current loop is that the current is being controlled so as not to exceed a specific limit and burn the armature winding and thus damage the motor. As mentioned before, the PI gains  $k_p$  and  $k_i$  were obtained using a trial and error method. There are other approaches for obtaining controller gains for instance: Zigler-Nichols, the particle swarm optimization, bode plot [7][8].

## B. Outer Speed Loop

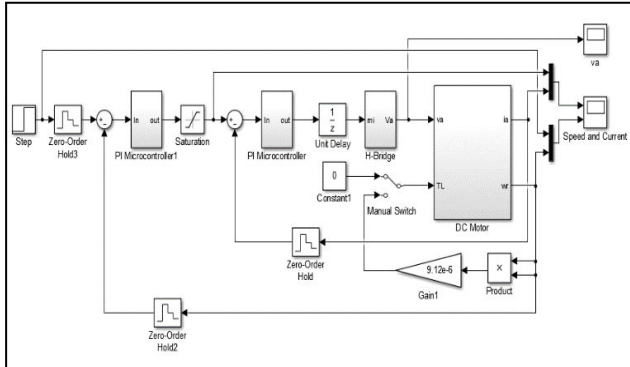
The outer speed loop consists of the inner current loop and another PI speed controller as shown in Figure-13. The PI speed controller model is the same for the current loop but it has an anti-integrator windup (Saturation Block). The purpose of integrator windup is to prevent integrator accumulating excessive output values.





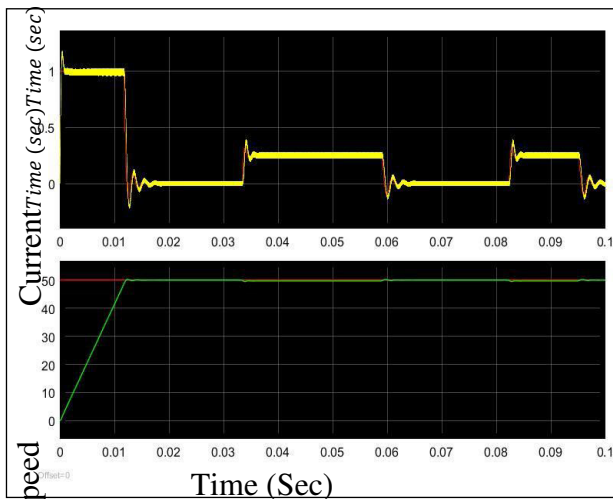
$k_p$  and  $k_i$  were obtained using trial and error and they were:

$$k_p = 0.5 \text{ and } k_i = 0.1.$$



**Figure-13.** The outer speed loop simulation in MATLAB

After setting the demand speed equal to 50, the output response of speed and current was obtained, shown in Figure-14 below, also shown is the effect of adding load torque on speed and current response. Red responses are the demand speed & current, while the yellow and green are the actual.



**Figure-14.** Current and speed responses at demand speed equal to 50.

It is clear from Figure-14 above that the speed has been slightly affected when the load torque was added to the motor, while the effect on current is observable.

#### 4. REAL TIME IMPLEMENTATION OF INNER CURRENT LOOP AND OUTER SPEED LOOP

##### A. Inner Current Loop

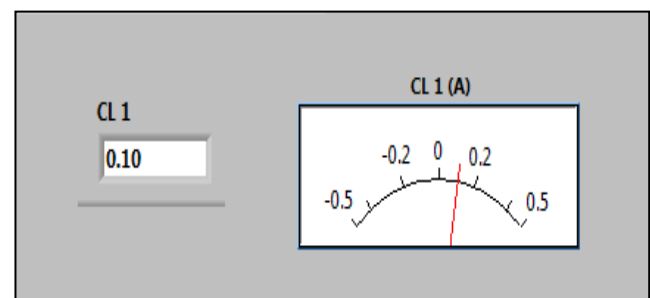
With the inner current loop, current can be controlled using a feedback loop and a PI controller. In this step, the PWM pulses were no longer controlled using a potentiometer, they were fixed depending on a specific

amount of current given. The aim of inner loop is to control the amount of current that supplies the load. One of the benefits that cascade control has is that the inner loop can be set up and tested first before the outer loop being engaged.

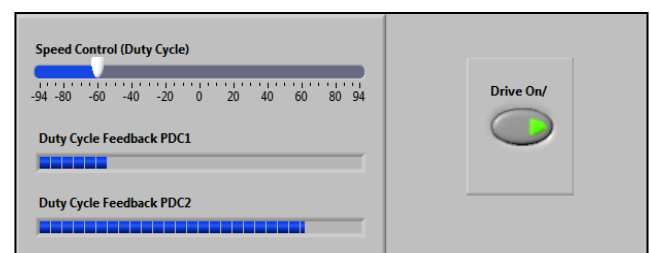
With the current loop control, a demand current was inserted in the code as a fixed value and the actual current was measured using a current sensor. An error was calculated by subtracting the actual current from the demand and then inserted into the PI controller. The inner current loop code has to be inserted within the `void_ISR_ADCInterrupt(void)` which is the analog digital conversion interrupt within this code, the  $k_p$  and  $k_i$  were defined as well as the demand current. The error was calculated and inserted to the calculation of modulation index which is the output of the current loop. The current integrator was limited to prevent the integrator windup, and the modulation index was also limited and then it was used to produce Leg A and Leg B PWM pulses.

The current can be changed by inserting different values each time and running the program again. And with DC motor load the direction of rotation can be changed by adding a negative sign to the demand current value.

LabVIEW software was used to monitor the current in two ways, by digital display and indicator display. Figure-15 below shows the two types of display. A further step was conducted to control the demand current using LabVIEW instead of inserting it as a fixed value. The control board on LabVIEW is shown in Figure-16.



**Figure-15.** Digital display on left and indicator display on right.



**Figure-16.** Demand current control board.

By moving the white arrow to the right or left, the value of duty cycle will be changed. Therefore, sscanf function was used to transfer any change occurring in LabVIEW (duty cycle channel) to decimal values so that c



program can understand, and the rest of the code does the function of producing PWM pulses. With DC motor load, the direction of rotation can be changed by the same arrow, if it moves in the positive direction it produces positive current and if it moves in the negative it produces negative current and therefore it changes the direction. Drive On/ is a button used to enable LabVIEW control of demand current.

A test was conducted to investigate the open loop behavior in real time implementation "from speed loop point of view" with constant current demand. At this test the machine was running open loop as there was no speed control. A power resistor was connected with the load machine after setting the test motor on a fixed demand current. This test was to see the effect of adding load torque on the speed of the motor.

After applying different amounts of power resistor to the motor the following curves of speed versus resistor and current were gained in Figures (17) and (18) below.

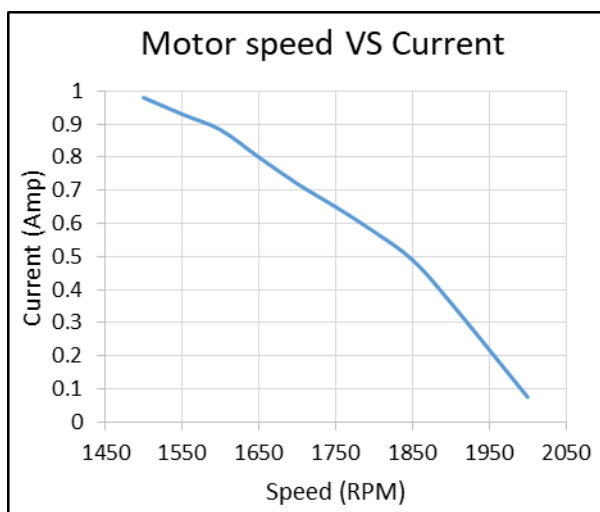


Figure-17. Motor speed versus Current.

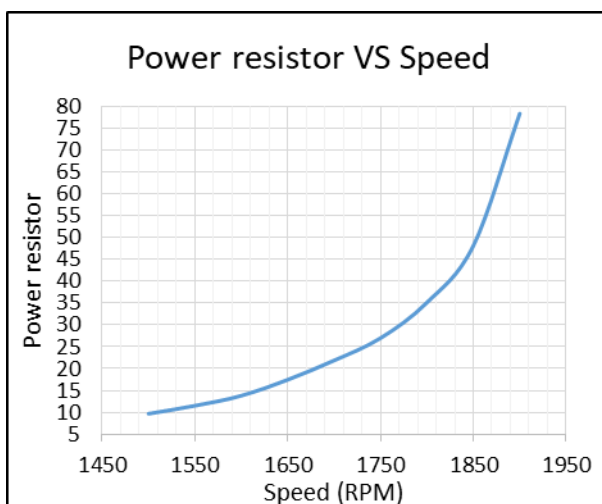


Figure-18. Motor speed versus Power Resistor.

Due to the parallel connection between the power resistor and load machine, the equivalent resistor will be low and hence the current becomes high when the magnitude of power resistor is reduced as shown in Figures (17) and (18) above. When the load torque (TL) increased, the current increased and due to the fact that the mechanical equation of the motor contains (Bw) which is the friction multiplied by the speed, the speed goes down to equalize the equation, taken into consideration that TE and J were constants, in equation (7).

$$R_{\text{equivalent}} = \frac{R_1 \cdot R_2}{R_1 + R_2} \quad (8)$$

$$V = i \cdot R \quad (9)$$

$$T = kt\psi \cdot i \quad (10)$$

### B. Outer Speed Loop

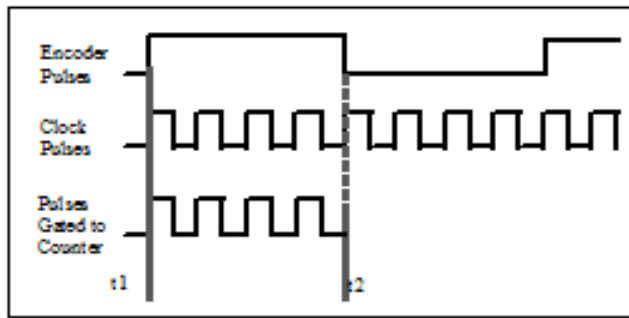
The second feedback of the cascade control system is the speed feedback. It is called outer loop because this loop contains the current loop (inner loop) inside. The loop aim is to control the speed with PI controller and thus to avoid speed drop when extra load torque is being applied on the motor. The actual speed was measured using an Encoder sensor attached to the motor shaft. A demand speed was inserted in the C code, and an error was calculated by subtracting the actual speed from the demand speed. This error is inserted to PI controller with *kp* and *ki* gains and the output of the controller is the demand current. Therefore, the demand current does not need to be identified as a constant anymore nor inserted via LabVIEW panel.

Speed integrator and demand current were both limited to prevent windup. Note: the outer speed loop code has to be inserted within the void `_ISR_ADCInterrupt(void)` which is the analog digital conversion interrupt.

## SPEED MEASUREMENT, RESPONSE AND PROFILE

### A. Speed Measurement

An encoder was used to measure the position and speed of a motor shaft. This instrument has a glass disk with 500 black marks etched on its surface. It also has a light emitting diode (LED) and photosensitive transistor, the LED produces light directed through the glass disk to a transistor. As the black marks pass through the light beam, pulses are produced by the transistor. Figure-19 shows one encoder pulse and how the pulses are used to measure the speed.



**Figure-19.** Speed measurement process.

The PIC controller has an oscillator inside it, this oscillator produces pulses with 67.8 Nano second per pulse. With the function “void \_ISR\_IC1Interrupt” the number of oscillator pulses that pass through one encoder pulse can be calculated.

The variable rpm that is shown in the code represents the number of oscillator pulses that pass through one encoder pulse and can be calculated by measuring t1 and t2. In the case of t1 is bigger than t2 a constant PR3 can be added, which is equal to 0xffff “load period register”. For example if t2 equals 0x0006 and t1 equals 0xfffa, t1 is bigger than t2 and to prevent this, 0xffff is added to give a positive result. To measure the speed in revolution per minute (RPM) which represents the actual speed for the speed loop, the following calculation has to be made:

Each oscillator pulse has 67.8 Nano second time period.  
rpm gives the number of oscillator pulses.

Therefore one encoder pulse time period can be calculated as follow:

$$\text{encoder pulse time period} = \text{rpm} * 67.8 * 10^{-9} \text{ sec} \quad (11)$$

The glass disk has 500 black marks which means 500 pulses.

Therefore one revolution time period can be calculated as follow:

$$\text{time in seconds for one revolution of encoder disk} = \text{rpm} * 67.8 * 10^{-9} * 500 \text{ sec} \quad (12)$$

$$\text{to get the speed in revolution per second} = \frac{1}{\text{rpm} * 67.8 * 10^{-9} * 500} \quad (13)$$

$$\text{Speed in RPM} = \frac{60}{\text{rpm} * 67.8 * 10^{-9} * 500} \quad (14)$$

Another device called Tachogenerator was used for measuring the actual speed values of a motor shaft that is coupled to it. A tachogenerator with a precise design produces precise voltages proportional to shaft speed. By measuring the generated voltages, it is easy to determine the amount of rotational speed. Tachogenerator can also detect the direction of rotation, as the polarity of the voltage changes with the change in direction. Figure-20

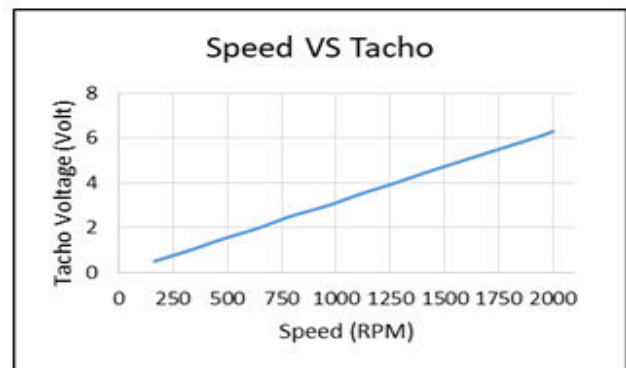
shows a similar type of tachogenerator to the one that was used in this project.



**Figure-20.** Tachogenerator.

### B. Speed Response Transient

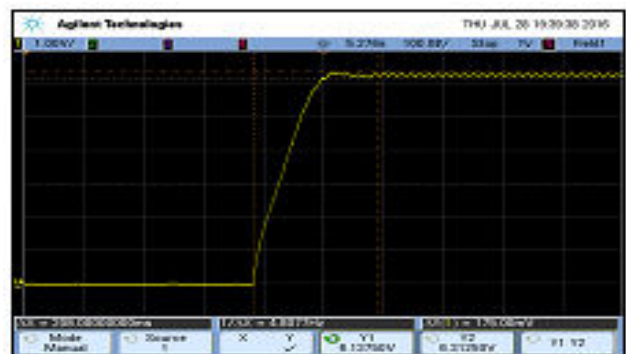
The tacho generator that is attached to the load motor of the test rig produce voltages that have a direct relationship with the speed. Figure-21 below shows the relationship between tacho voltage and speed.



**Figure-21.** Tacho voltage and speed relationship

From this relationship, speed response can be easily determined and recorded using an oscilloscope. The speed response was first recorded to modify  $k_p$  and  $k_i$  “PI gains” of the speed loop.

The speed response at the instant of switching on the power supply is shown in Figure-22.



**Figure-22.** Speed response at the instant of switching on the power.





A power resistor was attached to the load machine to help modifying speed response characteristics such as overshoot and settling time. The power resistor was connected in parallel with the load so as to reduce the equivalent resistor and thus increase the current according to ohms law. Figure-23 shows how the power resistor was connected with the test rig.

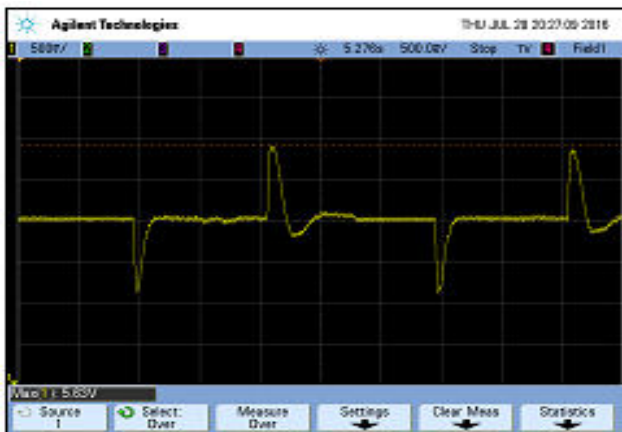


**Figure-23.** Power resistor and motor test rig connection.

The following procedure was conducted to observe the effect of modifying  $k_p$  and  $k_i$ :

Firstly, the c code was compiled to run at a constant speed of 1500 RPM by applying a demand speed equals to 1500, then the power resistor was set at its minimum value which gave the highest current that the motor can draw which was limited to 1 amp. The power resistor reading by digital voltmeter (DVM) was 10.4 ohm, and the current reading was 0.94 amp.

Secondly, the power resistor was connected and disconnected from the load more than once during a 5 second period. Figure-24 shows the speed transient response during the 5 second period.



**Figure-24.** Speed transient response during the 5 second period with  $k_p = 2.114$  and  $k_i = 0.001$

It is clear from the figure that the speed response was effected during the plugging in and out of the resistor. The PI gains at this test were  $k_p = 2.114$  and  $k_i =$

0.001 and the tacho voltage reading was 4.77 volts before connecting the power resistor and became 5.63 at the instant of connecting, then settling at 4.77 volts after a small period of time. The overshoot percentage was calculated as follow:

$$PO = \frac{M_{pt} - f_v}{f_v} * 100\% \quad (15)$$

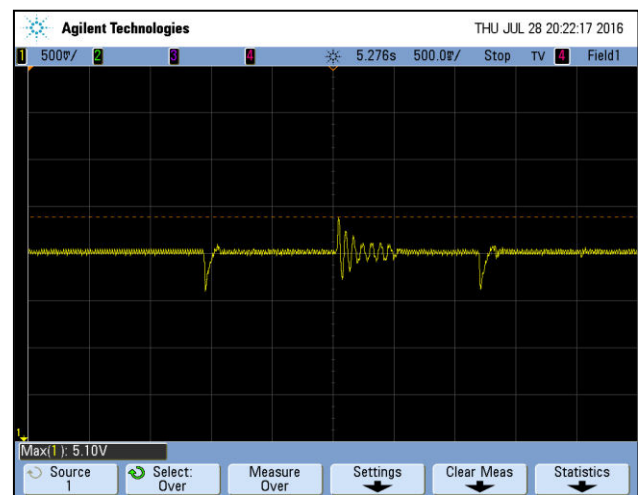
$M_{pt}$  is the peak value of the time response.

$f_v$  is the final value of the response.

$$\text{OverShoot percentage} = \frac{5.63 - 4.77}{4.77} 100\%$$

$$\text{Overshoot percentage} = 18\%$$

Thirdly, further tuning was applied on  $k_p$  and  $k_i$  to minimize the overshoot percentage, the new PI gains are:  $k_p = 2.214$  and  $k_i = 0.02$ . Figure-25 shows the transient response of speed after tuning.



**Figure-25.** Speed transient response during the 5 second period with  $k_p = 2.214$  and  $k_i = 0.02$

The overshoot percentage was calculated as follow:

$$\text{OverShoot} = \frac{5.1 - 4.77}{4.77} * 100\%$$

$$\text{Overshoot} = 7\%$$

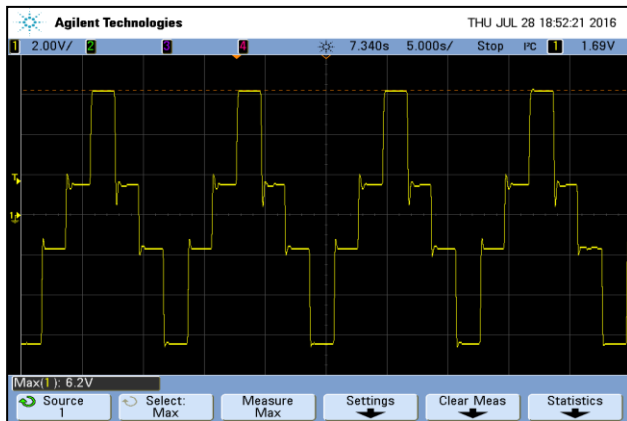
### C. Speed Profile

For more advanced applications speed needs to be changed repeatedly within a specific time period. With such speed control a specific speed profile can be done easily and via the tachogenerator the speed profile can be monitored. C code was used to produce two different speeds 500 and 2000 RPM within two directions of rotation for a 6 second time period, after the 6 seconds ends the counter resets and starts again.

When the power was switched on the motor was spinning in clockwise direction with 500 RPM, after one second the speed increased to 2000 RPM and after another second the speed reduced to 500 RPM, then it changed the



direction and so on. Figure-26 below shows the speed profile which was recorded by an oscilloscope via measuring tachogenerator voltage.



**Figure-26.** Speed profile

(1.52 → 6.2 → 1.52 → -1.52 → -6.2 → -1.52) volt  
(500 → 2000 → 500 → -500 → -2000 → -500) RPM

## CONCLUSIONS

This project was undertaken to investigate the performance of a permanent magnet brushed DC motor under cascade control conditions. The procedure was conducted in two phases, MATLAB Simulation phase and Real Time implementation phase. Each phase studied some of the cascade systems circuit characteristics, such as, open loop, current loop and speed loop. Six main targets were focused on which are Unipolar PWM generation, speed measurement using an encoder, current sensor calibration, speed transient, speed profile and integrator windup. A lot of measurements were taken during the procedure, most of them were focused on the output responses of speed and current as well as the PWM pulses produced by the H-bridge converter. Cascade control has proved its ability to drive direct current motors efficiently, the motor was tested under load and no load conditions as well as at blocking mode condition. It was realised that the real time implementation is important to understand the motor behavior under different circumstances. MATLAB Simulink is crucial to understand how the system behaves in state space representation, however it might not give similar results to real time work. The most obvious finding to emerge from this study is that controlling a DC motor is essential due to its necessity in modern technology for instance robots.

## REFERENCES

- [1] H. Taha and I. Alnaab. 2019. Designs of PMSMs with Inner and Outer Rotors for Electric Bicycle Applications. *Kurdistan Journal of Applied Research*. 4(1): 20-25.
- [2] A. S. A. Elhamid. 2012. Cascade Control System of Direct Current Motor. *World Applied Sciences Journal*. 18(12): 1680-1688.
- [3] N. Bacac, V. Slukic, M. Puškarić, B. Stih, E. Kamenar and S. Zelenika. Comparison of different DC motor positioning control algorithms. 2014. In 37<sup>th</sup> Int. Convention on Information and Communication Technology., Electronics and Microelectronics., MIPRO. pp. 1654-1659.
- [4] M. Kushwah and A. Patra. 2014. Tuning PID Controller for Speed Control of DC Motor Using Soft Computing Techniques-A Review. *Advances in Electronic and Electric Engineering, AEEE*. 4: 141-148.
- [5] M. Namazov. 2010. DC motor position control using fuzzy proportional-derivative controllers with different defuzzification methods. *Turkish J. Fuzzy Syst., TJFS*. 1(1): 36-54.
- [6] A. P. Singh, U. Narayan and A. Verma. 2003. Speed Control of DC Motor using Pid Controller Based on Matlab. *Innovative Systems Design and Engineering*. 3(9): 1209-1220.
- [7] M. N. Kamarudin and S. M. Rozali. 2008. Simulink implementation of digital cascade control DC motor model - a didactic approach. In 2<sup>nd</sup> IEEE Int. Conf. on Power and Energy., PECon 08., Johor Baharu., Malaysia. pp. 1043-1048.
- [8] R. G. Kanojiya and P. M. Meshram. 2012. Optimal tuning of PI controller for speed control of DC motor drive using particle swarm optimization. 2012 Int. Conf. on Advances in Power Conversion and Energy Technologies (APCET), Mylavaram, Andhra Pradesh. pp. 1-6.
- [9] H. M. Tun and W. Aung. 2014. Analysis of Control System for A 24V PM Brushed DC Motor Fitted with an Encoder by Supplying H-Bridge Converter. *Bahria University Journal of Information & Communication Technologies*. 7: 54-67.
- [10] J. Scott, J. McLeish and W. H. Round. 2009. Speed control with low armature loss for very small sensorless brushed DC motors. *IEEE Trans. Ind. Electron.* 56(4): 1223-1229.
- [11] L. S. Patel and K. C. Dave. 2011. Cascade control Technique for D. C. Motor Speed Control. In *Proc. of*



the Int. Conf. on Science and Engineering., ICSE. pp. 599-603.

- [12] K. Anagha, C. Ranjith, P. Anand, D. Rahul and A. S. Anusha. 2014. CASCADE SPEED CONTROL OF DC MOTOR. International Journal of Electrical, Electronics and Data Communication. 2: 78-81.
- [13] R. Bhavina, N. Jamliya and K. Vashishtha. 2013. Cascade Control of Dc Motor With Advance Controller. International Journal of Electrical, Electronics and Data Communication. 2: 18-20.