Complete Design of a Hardware and Software Framework for PWM/Discrete PID-Based Speed Control of a Permanent-Magnet DC Motor Without Prior Knowledge of the Motor's Parameters

Chady El Moucary, Abdallah Kassem, Walid Zakhem, Chaybane Ghabach, Roger El Khoury, and Patrick Rizk

1 Introduction

The objective of this chapter resides in presenting a comprehensive approach for the design and implementation of an effective computational tool to control the speed of a Permanent-Magnet DC (PMDC) motor.

This type of motor is widely used in innumerable industrial applications due to its rugged structure, low cost, high efficiency, and pertinent characteristics. It does not require a separate excitation coil, hence the reduced size and lower power consumption. The general block diagram that depicts the overall system is shown in Fig. 1.

The speed control is achieved using a discrete PID controller with the ability of online tuning and adjusting the parameters to a changing desired trajectory. PID controllers are also widely used for their versatile features in monitoring the shape of the tracking error according to desired mode of responses, steady-state errors, and required dynamics. They are also chosen for their ease and variety of implementation techniques and methods.

The computational tool encompasses a PIC microcontroller and a fully comprehensive, user-friendly, and resourceful interface designed using Visual Basic. VB programming offers an adaptable platform known for its appealing interfacing

C. E. Moucary (✉) · A. Kassem · W. Zakhem · C. Ghabach · R. E. Khoury · P. Rizk
Department of Electrical, Computer, and Communication Engineering, Notre Dame University–Louaize, Zouk Mosbeh, Lebanon
e-mail: celmoucary@ndu.edu.lb; akassem@ndu.edu.lb; wzakhem@ndu.edu.lb

J. M. Alja'am et al. (eds.), *Recent Trends in Computer Applications*,
https://doi.org/10.1007/978-3-319-89914-5_10

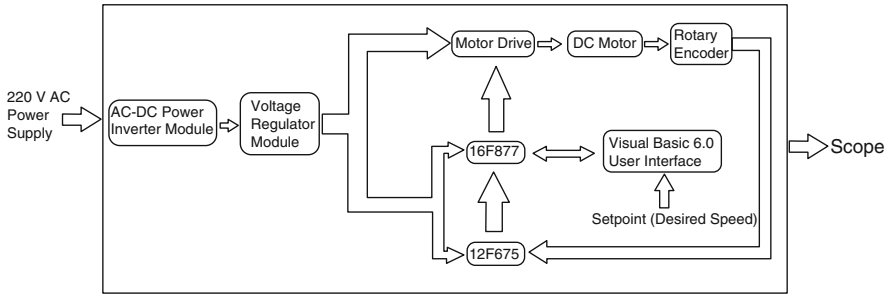


Fig. 1 Block diagram of the drive

features and portability. The GUI is fashionably designed to incorporate a broad and multipurpose portal to access, monitor, edit, and record all pertinent parameters of the entire drive.

This chapter will be divided in nine sections. Fundamentals of the speed control of a PDMC motor using PWM techniques are presented in the second section. The third section discusses the general model of the PID controller and the pertinent direct canonical forms are then presented in Sect. 4. The implementation of the PID controller is showcased in Sect. 5. Electric circuit design and pertinent schematics are then elaborated in Sect. 6, which also shows the platform used for simulation and testing purposes. The piloting software and the functionality of the GUI are overviewed in Sect. 7, followed by experimental results depicted in Sect. 8 that underline the effectiveness of the overall framework. Finally, a conclusive summary is presented in Sect. 9.

2 Pulse Width Modulation (PWM)

The foundation of modern mechanical systems lies in control systems that allow designing of apparatus that would theoretically perform according to any granularity in terms of specification requirements such as dynamic and steady-state behaviors. Control operations can be achieved in either open-loop or closed-loop approaches; the key difference is feedback. In open-loop configurations, the system acts completely on the basis of input; the output has no effect on the underlying action. Hence, no feedback is available and/or used to adjust the behavior to the desired path or outcome. Arguably the most ingenious tool in this case is the closed-loop outlook, which shares the most constituent components of the aforementioned open-loop platform but with some relevant data being passed back from some point into the control system to another preceding point. Such data is primarily used to modify/correct the response for the actual output to closely follow a desired reference trajectory in terms of many related criteria; hence, the system becomes self-adjusting. Despite the fact that open-loop systems are by far simpler to design

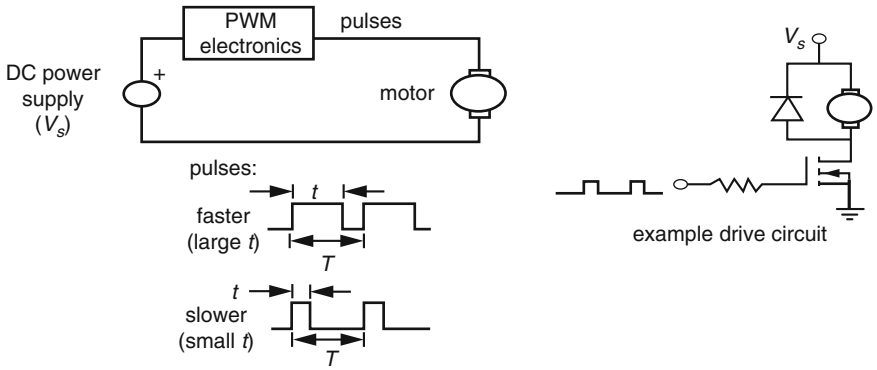


Fig. 2 Pulse Width Modulation for a DC motor

and entail low implementation cost, closed-loop systems offer a decisively more resourceful framework, and somehow become mandatory when strict requirements in terms of dynamic and steady-state operations are needed. Closed-loop speed control systems involve speed sensors such as rotary encoders that make the actual output available for feedback. Actual speed measurement values are constantly compared to the desired reference value, which is set by the user and called the *set point*. The difference between those values is then fed to an astutely designed controller that would adjust the motor input (voltage or current) to ensure effective tracking of the set point. Electronic speed controllers are of two types: linear amplifiers and pulse width modulators (PWMs). PWM controllers present the advantage of either driving bipolar power transistors rapidly between cutoff and saturation or turning FETs ON and OFF. In either case, power dissipation is small. Servo amplifiers using linear power amplification are satisfactory but produce a lot of heat, because they function in the transistor linear region. Commercial servo controllers can be achieved using linear amplifiers, but because of lower power requirements, ease of design, smaller size, and lower cost, switched amplifier designs are used [1–4].

Figure 2 depicts the principle of a PWM amplifier. A DC power supply voltage is rapidly switched at a fixed frequency f between two values (e.g., ON and OFF). This frequency is often in excess of 1 kHz. The high value is held during a variable pulse width t within the fixed period T where $T = 1/f$.

The resulting asymmetric waveform has a duty cycle defined as the ratio between the ON time and the period of the waveform, usually specified as a percentage:

$$\text{Duty cycle} = \frac{t}{T} \times 100 \tag{1}$$

As the duty cycle is changed (by the controller), the average current through the motor changes, causing changes in speed and torque at the output. It is primarily the duty cycle, and not the value of the power supply voltage, that is used to control the

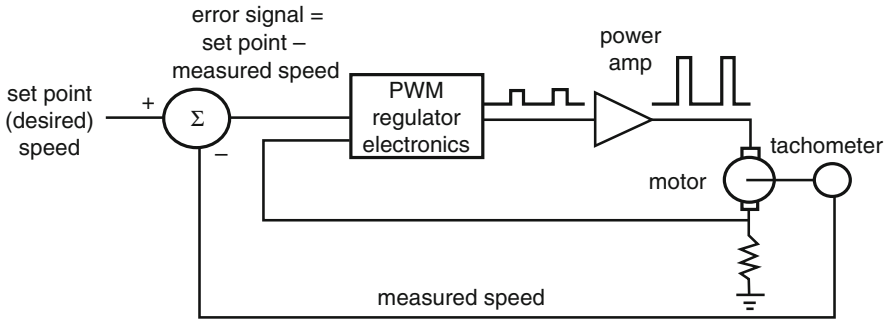


Fig. 3 PWM speed control

speed of the motor. The block diagram of a PWM speed feedback control system for a DC motor is shown in Fig. 3. A voltage tachometer produces an output linearly related to the motor speed. This is compared to the desired speed set point (another voltage that can be manually set or computer controlled). The error and the motor current are sensed by a pulse-width-modulation regulator that produces a width-modulated square wave as an output. This signal is amplified to a level appropriate to drive the motor.

In a PWM motor controller, the armature voltage switches rapidly, and the current through the motor is affected by the motor inductance and resistance. Since the switching speed is high, the resulting current through the motor has a small fluctuation around an average value. As the duty cycle grows larger, the average current grows larger and the motor speed increases.

3 PID Controller

The proportional integral derivative or the PID is one of the most commonly used controllers nowadays. The control signal is generated from three terms: a term that is proportional to the error, a term that is integral to the error, and a term that is derivative of the error [5–8]. The sum of these three terms will serve as control signal for the PWM block. The PID controller can be modeled by the block diagram shown in Fig. 4.

The components of a PID system are:

- Proportional term K_P depends on the present error. This term defines the speed of change in the output.
- Integral term K_i is the accumulation of past errors. It aids in reaching with a faster response the steady state and eliminates the residual steady-state error that results from the pure use of a proportional controller.
- Derivative term K_d is the prediction of future errors. This term of control is used to decrease the magnitude of the overshoot.

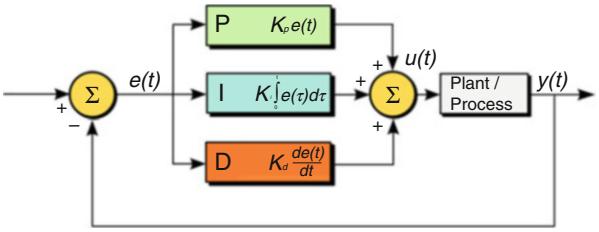


Fig. 4 PID block diagram

Analyzing the block diagram, we obtain the following equation:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)\mathrm{d}t + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t} \tag{2}$$

where $K_i = \frac{K_p}{T_i}$ and $K_d = K_p T_d$; therefore, we can write the transfer function of a continuous-time PID as

$$\frac{U(s)}{E(s)} = K_p + \frac{K_p}{T_i s} + K_p T_d s \tag{3}$$

The discrete form of PID controller can also be derived by finding the z-transform of equation (3).

Therefore, we obtain:

$$\frac{U(z)}{E(z)} = K_p \left[1 + \frac{T}{T_i \left(1 - z^{-1}\right)} + T_d \frac{\left(1 - z^{-1}\right)}{T} \right] \tag{4}$$

Then

$$\begin{aligned} u(kT) = {} & u\left(KT - T\right) + K_p \left[e(kT) - e\left(kT - T\right)\right] \\ & + \frac{K_p T_d}{T} \left[e(kT) - 2e\left(kT - T\right) - e\left(kT - 2T\right)\right] \end{aligned} \tag{5}$$

The PID controller is accurately tuned using the Ziegler–Nichols approach [9–13]. The closed-loop tuning algorithm based on plant closed-loop tests is as follows:

1. *Disable any derivative and integral action in the controller and leave only the proportional action.*
2. *Carry out a set-point step test and observe the system response.*
3. *Repeat the set-point test with increased (or decreased) controller gain until a stable oscillation is achieved; this gain is called the ultimate gain* $\boldsymbol{K_u}$.
4. *Read the period of the steady oscillation and let this be* $\boldsymbol{T_u}$.

Table 1 Effect of PID parameters on the tracking error

Parameter	Rise time	Overshoot	Settling time	Steady-state error	Stability
K_p	Decrease	Increase	Small change	Decrease	Degrade
K_i	Decrease	Increase	Increase	Eliminate	Degrade
K_d	Minor change	Decrease	Decrease	No effect in theory	Improve if small

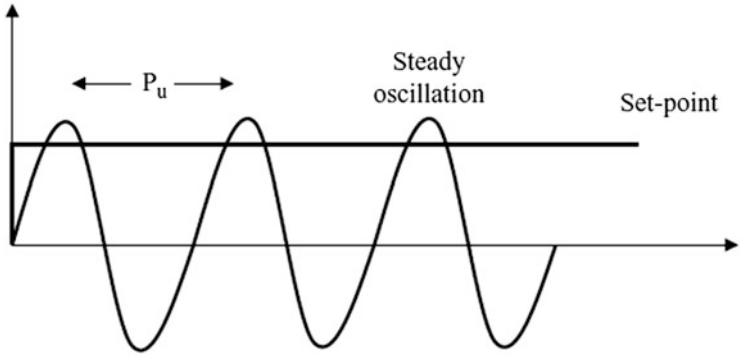


Fig. 5 Ziegler–Nichols closed loop

5. *Calculate the controller parameters according to the following formulas*: $\boldsymbol{K_P = 0.45K_u}$, $\boldsymbol{T_i = T_u/1.2}$ *in case of PI controller, and* $\boldsymbol{K_P = 0.6K_u}$, $\boldsymbol{T_i = T_u/2}$, $\boldsymbol{T_d = T_u/8}$ *in the case of the PID controller.*

Table 1 summarizes the effect of changing the parameters of the PID controller (Fig. 5).

The Ziegler–Nichols rules are then applied to obtain initial controller design followed by design iteration and refinement. By using the Ziegler–Nichols formulas we have:

$K_P = 0.6, K_U = 53.13, K_I = 1.2, K_U/T_U = 1280.2$, and $K_D = 0.6, K_U\ T_U/8 = 55$. Those parameters yielded a settling time of about 2.4 s and a percentage overshoot of approximately 60%.

The Ziegler method based on assumed forms of the process presents decisive advantages for it allows achieving a speed control of the motor without prior knowledge of its parameters. This method shall procure the system with the required constants $a0$, $a1$, $a2$, $b1$, $b2$ that will be used as preset values for the compensator. The physical meaning of these constants and their relationship to the PID parameters are explained in the following section.

4 Direct Canonical Forms

The strategy of a numerical control structure begins with a precise model of the process to be controlled. Then a control algorithm is developed, which will ensure the required system response. The loop is closed by using a digital computer as the controller. The computer implements the control procedure in order to obtain the desired response. Different approaches do have dissimilar computational efficiencies, dissimilar sensitivities to parameter errors, and dissimilar programming techniques are needed in each case. As such, various approaches are available for implementation. To name a few, cascaded structures, parallel structures, second-order structures, and direct structures could be used for building the controller. In fact, two different types of direct structures can be considered: the direct noncanonical structure and the direct canonical structure [14–17].

The direct canonical structure is selected since it presents significant advantages over other structures such as memory size and efficiency as it requires a smaller number of memory locations and the number of delay elements is fixed.

In direct structure, the coefficients a_j and b_j appear as multipliers. By considering that b_0=1, for the discrete compensator, therefore, we can express

$$D(z) = \frac{U(z)}{E(z)} = \frac{\sum_{j=0}^{n} a_j z^{-j}}{1 + \sum_{j=0}^{n} b_j z^{-j}} \tag{6}$$

Let us introduce now a new variable R(z) such that

$$\frac{U(z)}{R(z)} \frac{R(z)}{E(z)} = \frac{\sum_{j=0}^{n} a_j z^{-j}}{\sum_{j=0}^{n} b_j z^{-j}} \tag{7}$$

or

$$\frac{U(z)}{R(z)} = \sum_{j=0}^{n} a_j z^{-j} \text{ and } \frac{E(z)}{R(z)} = \sum_{j=0}^{n} b_j z^{-j} \tag{8}$$

Assume that the transfer function of a digital controller is

$$R(z) = E(z) - \sum_{j=1}^{n} b_j z^{-j} R(z) \tag{9}$$

$$\text{And } U(z) = \sum_{j=0}^{n} a_j z^{-j} R(z) \tag{10}$$

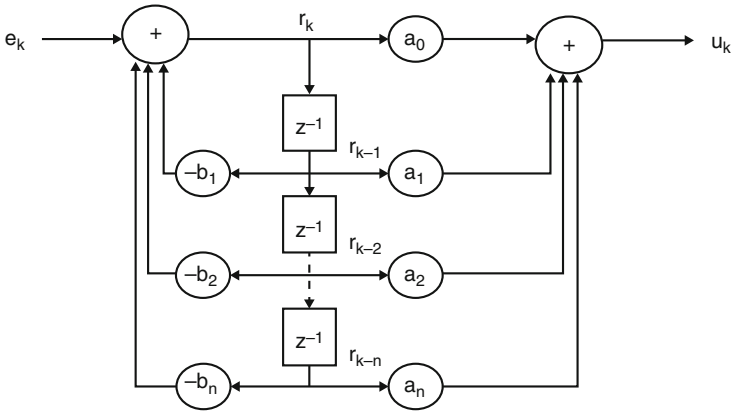


Fig. 6 Block diagram implementation discrete compensator

The equations above can be written in time domain

$$r_k = e_k - \sum_{j=1}^{n} b_j r_{k-j} \tag{11}$$

$$u_k = \sum_{j=0}^{n} a_j r_{k-j} \tag{12}$$

Those equations define the direct form, and the block diagram of implementation is shown in Fig. 6. The controller is made up of delays, adders, and multipliers.

5 Controller Implementation

The z-transform of the PID controller was derived before, and is reproduced here for convenience:

$$D(z) = K_p + \frac{K_p T}{T_i \left(1 - z^{-1}\right)} + \frac{K_p T_d \left(1 - z^{-1}\right)}{T} \tag{13}$$

An alternative implementation of the PID would be to find a second-order transfer function for $D(z)$ and then use the direct structure to implement it. The equation can be written as

$$D(z) = \frac{K_p \left(1 - z^{-1}\right) + \frac{K_p T}{T_i} + \left(\frac{K_p T_d}{T}\right) \left(1 - z^{-1}\right)^2}{1 - z^{-1}} \tag{14}$$

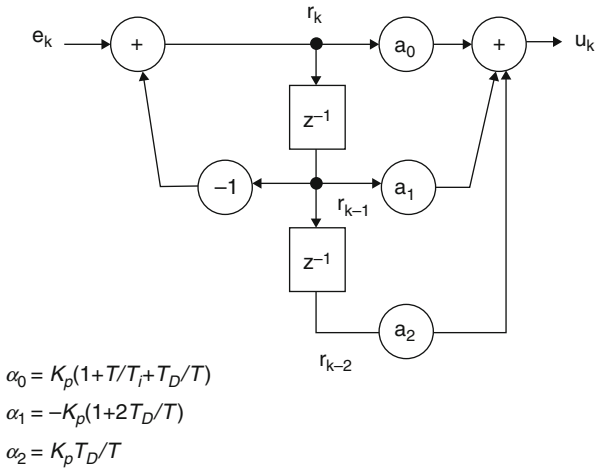


Fig. 7 PID implementation—direct canonical structure

Therefore,

$$D(z) = \frac{K_p + \frac{K_p T}{T_i} + \frac{K_p T_d}{T} - \left(K_p + \frac{2K_p T_d}{T}\right) z^{-1} + z^{-2}\left(K_p T_d / T\right)}{1 - z^{-1}}, \tag{15}$$

which is of the form $\frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}}$,where

$$a_0 = K_p\left(1 + \frac{T}{T_i} + \frac{T_d}{T}\right), a_1 = -K_p\left(1 + \frac{2T_d}{T}\right), a_2 = \frac{K_p T_d}{T}, b_1 = -1, b_2 = 0. \tag{16}$$

Figure 7 shows the PID implementation as direct canonical structure.

Considering again the velocity form of PID, and replacing the kT simply by subscript k, we can write

$$u_k = u_{k-1} + \left[K_p + \frac{K_p T}{T_i} + \frac{K_p T_d}{T}\right] e_k - \left[K_p + \frac{2K_p T_d}{T}\right] e_{k-1} + \frac{K_P T_d}{T} e_{k-2} \tag{17}$$

Alternatively, we can write (17) in a simpler form as

$$u_k = u_{k-1} + a e_k + b e_{k-1} + c e_{k-2} \tag{18}$$

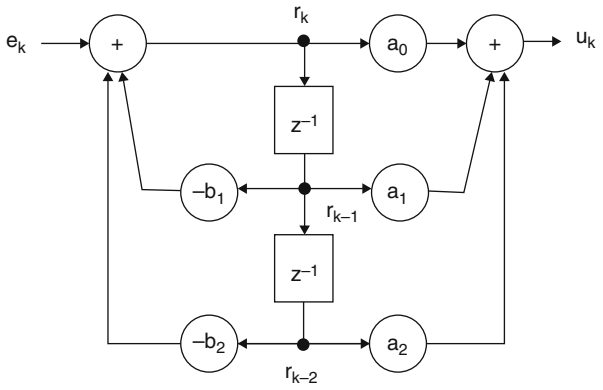


Fig. 8 Second-order module implementations

where

$$a = K_p + \frac{K_p T}{T_i} + \frac{K_p T_d}{T}, b = -\left[K_p + \frac{2K_p T_d}{T}\right], c = \frac{K_p T_d}{T} \tag{19}$$

By taking the z-transform we obtain

$$D(z) = \frac{U(z)}{E(z)} = \frac{a + bz^{-1} + cz^{-2}}{1 - z^{-1}} \tag{20}$$

Notice that if only proportional plus integral (PI) action is required, the derivative constant T_d can be set to zero and the PI equation becomes $D(z) = \frac{U(z)}{E(z)} = \frac{a_1 + bz^{-1}}{1 - z^{-1}}$ with $a = K_p + \frac{K_p T}{T_i}$ and $b = K_p$.

The second-order module is shown in Fig. 8. This module serves as our final model of the discrete controller. It ensures an enhanced response in presence of disturbances.

Where

$$Q(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}} \tag{21}$$

The difference equations describing such a module are

$$r_k = e_k - b_1 r_{k-1} - b_2 r_{k-2} \tag{22}$$

$$u_k = a_0 r_k + a_1 r_{k-1} + a_2 r_{k-2} \tag{23}$$

If we let

$$M_1 = -b_1 r_{k-1} - b_2 r_{k-2} \text{and } M_2 = a_1 r_{k-1} + a_2 r_{k-2} \tag{24}$$

Then these equations for the second-order module become

$$r_k = e_k + M_1 \tag{25}$$

$$u_k = a_0 r_k + M_2 \tag{26}$$

6 Circuit Design and Schematics

In this section, we shall discuss the electric circuit implementation of the system using a bottom-up approach. The function of each component will be analyzed, then the overall operation is summarized. Figure 9 depicts the complete schematics of the drive.

In order to minimize the number of wires needed, we implemented a voltage regulator module, shown in Fig. 10, which controls various components. Along with the 30 V power supply, a 5 V DC source is needed in order to supply the PIC16F877A as well as PIC12F675. Moreover, a 12 V voltage source is required to supply the power MOSFET transistor.

The PIC16F877A has 40 pins and five ports. It uses reduced instruction set RISC (35-instruction set); it can support four different types of oscillators—the crystal

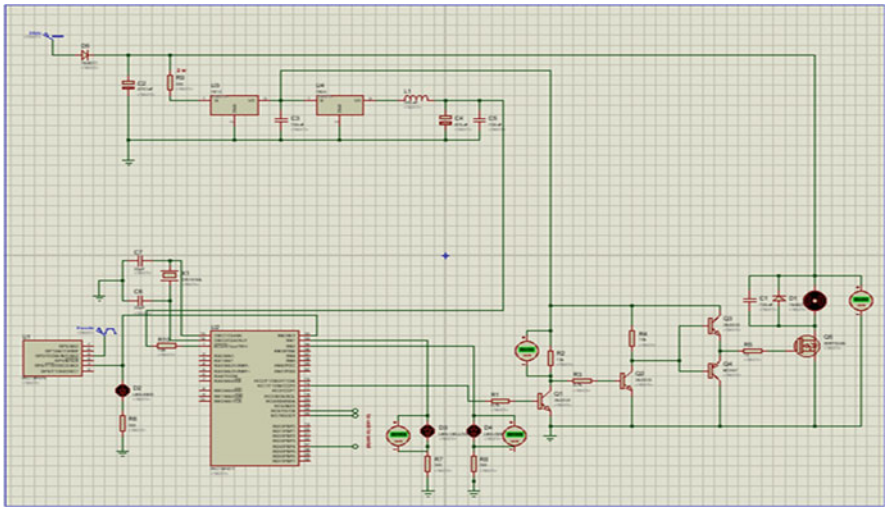


Fig. 9 Circuit of DC motor control using PID

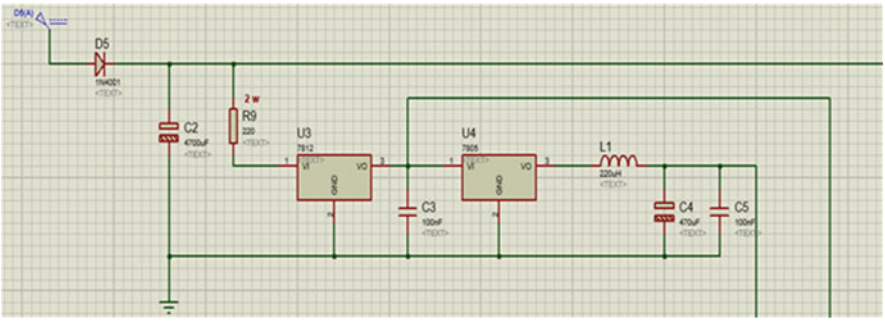


Fig. 10 Voltage regulator module

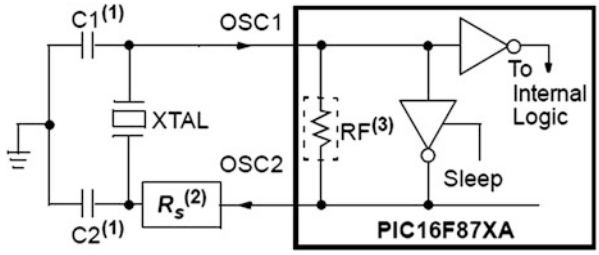


Fig. 11 Crystal oscillator

oscillator 770(XT) was adopted. Figure 11 shows the choice of the capacitors of 13 pF each in order to generate a frequency of 4 MHz, and since most instructions require four cycles, the operating frequency of the PIC is 1 MHz. Moreover, it has three different timers and can accommodate up to 15 different sources of interrupts. Only three interrupts are required: INTERRUPT_TIMER0, INTERRUPT_RC, and INTERRUPT_RB0. These three interrupt service routines are assigned to Timer0, USART (universal asynchronous transmitter receiver), and pin RB0, respectively.

INTERRUPT_TIMER0 subroutine is set each 100 ms, the flag TO1F will be set and thus, the interrupt service routine is called. This routine will read the actual speed of the motor and will place it in a variable called ACTUAL_SPEED.

To measure the speed of the motor, a rotary encoder was used; each round per second (or rps) will generate 400 pulses. The rotary encoder is connected to the input of the PIC12F675 (GP2) as per Fig. 12. By doing so, all pulses are saved and no data is lost. The microcontroller divides the number of pulses obtained from the rotary encoder by 40, that is, the number of pulses is now 400/40. Thus, each one round from the motor's shaft is now equivalent to 10 pulses. This relationship summarizes the measuring procedure of the speed as a linear relationship for the calculation of the equivalent number of pulses for x number of physical pulses.

To make sure that we tackled the issue of speed measuring (all pulses are counted) let us consider a high-speed scenario. The rated speed of the motor is 3300 rpm, which will generate (3300 $\times$ 400)/60 or 22,000 pulses/s. This number

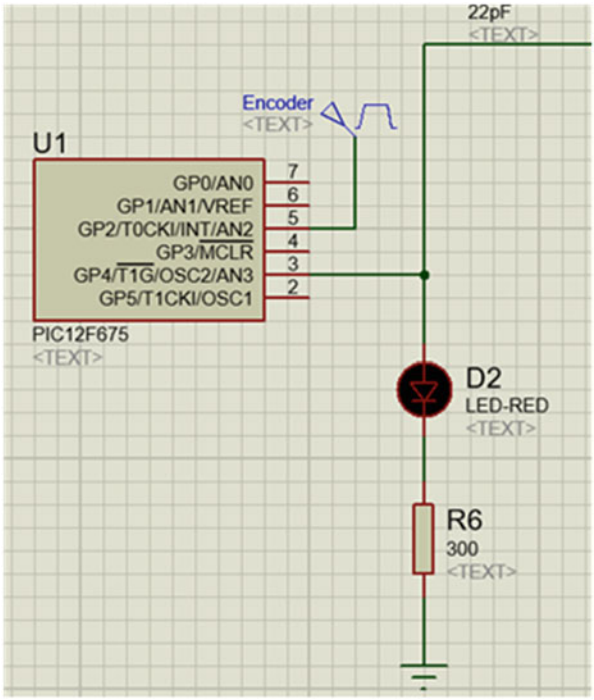


Fig. 12 PIC12F675 module

will be divided by 40 and thus, if the motor is running at its maximum speed (3300 rpm or 55 rps) the total number of pulses generated will be 550 pulses per second. However, the PIC16F877A is an 8-bit microcontroller and hence, could not accommodate the entire range of numbers issued by the pulse counter. Consequently, an interrupt timer0 is set at each 100 ms and hence, instead of reading 550 pulses per second at rated speed, only 55 pulses per 100 ms will be read. Because this is the maximum number of pulses that can be generated; therefore, if the motor is running at any other given speed, the number of pulses will all be saved without any loss since an 8-bit register can save up to $2^8 - 1$ or 255.

Figure 13 shows a Totem Pole driver circuit configuration chosen over the H-Bridge driver due to its high abilities for fast switching as well as for having a low cost and being easily implementable.

Figure 14 is a standard rectifier circuit that is simulated using National Instruments Multisim.

Figure 15 shows the 3D models of the PCB circuit generated using Proteus.

Figure 16 shows the actual image of the PCB with the components installed in it.

All of the above-detailed circuits were tested throughout simulation to check the readiness of the framework and its compliance to the system's specification requirements before any experiment is carried out. This step is crucial in order to

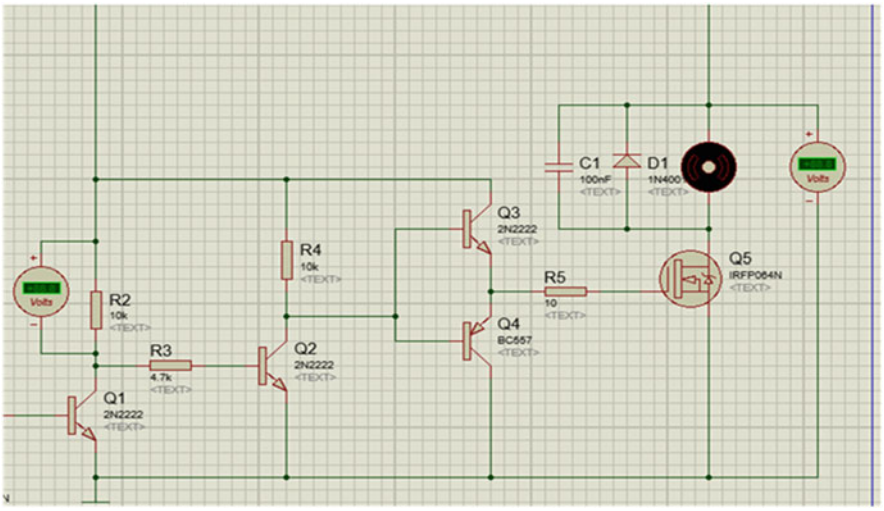


Fig. 13 The drive motor module

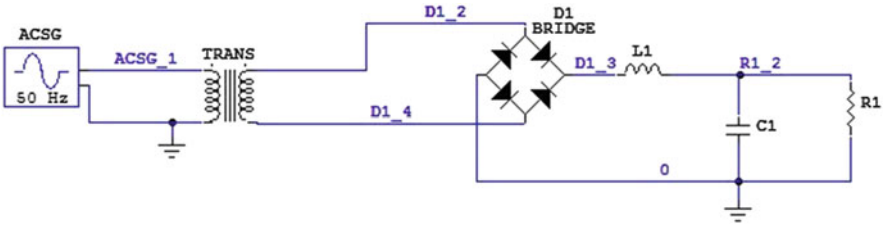


Fig. 14 Power supply module

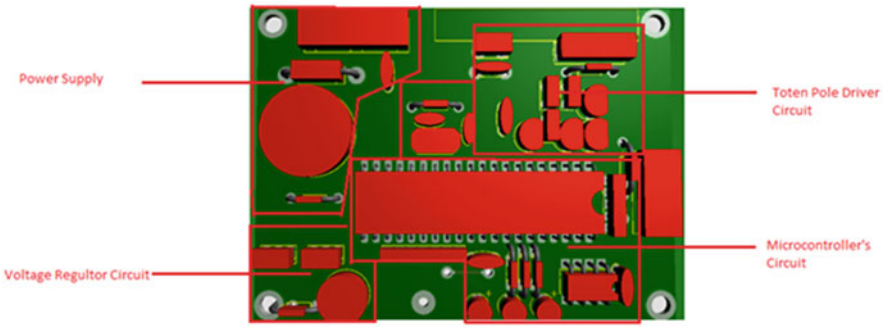


Fig. 15 PCB layout

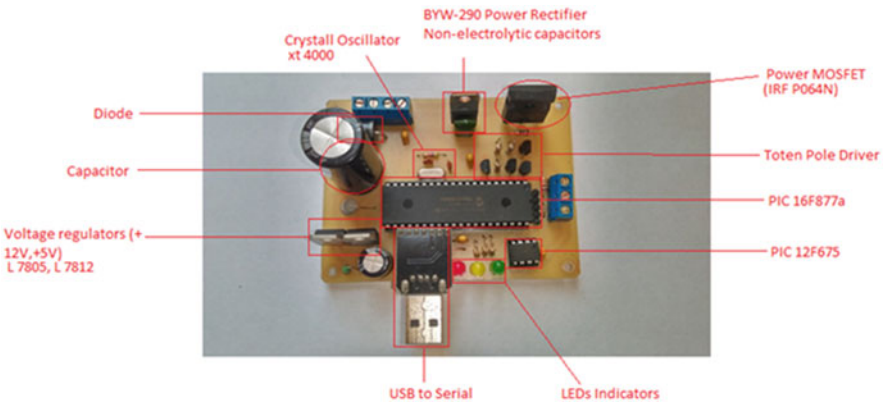


Fig. 16 PCB with components installed (top view)

prevent any possible damage to the motor and/or the electric components. The next step is to develop the pilot software to be used for the control scheme. A resourceful GUI is presented in the following section that offers a versatile and effective tool, as will be shown.

7 Software Design

In this section, the main pilot software is presented. It is developed in Visual Basic 6, which is an integrated development environment (IDE) that was developed by Microsoft. A Graphical User Interface was developed to entail the user a friendly yet versatile and effective tool for parameter calibration and control. The GUI is depicted in Fig. 17. It is divided into several parts: the controller port interface with the main computer, the compensator parameters, and the display properties, as showcased in Figs. 18 and 19.

To obtain the most optimum response in terms of overshoot as well as settling time, a "Preset" button is available to set the values of $a0$, $a1$, $a2$, $b1$, $b2$ to 1.3, 0.5, 0.2, 0.1, and 0.6, respectively. In fact, these values were obtained after extensive PID tuning.

Figure 20 below showcased a closed look at the compensator structure.

Figure 21 depicts where the online pertinent parameters are calculated and displayed. It also exhibits the capacity of saving results and data in excel files with all relevant parameters for offline analysis.

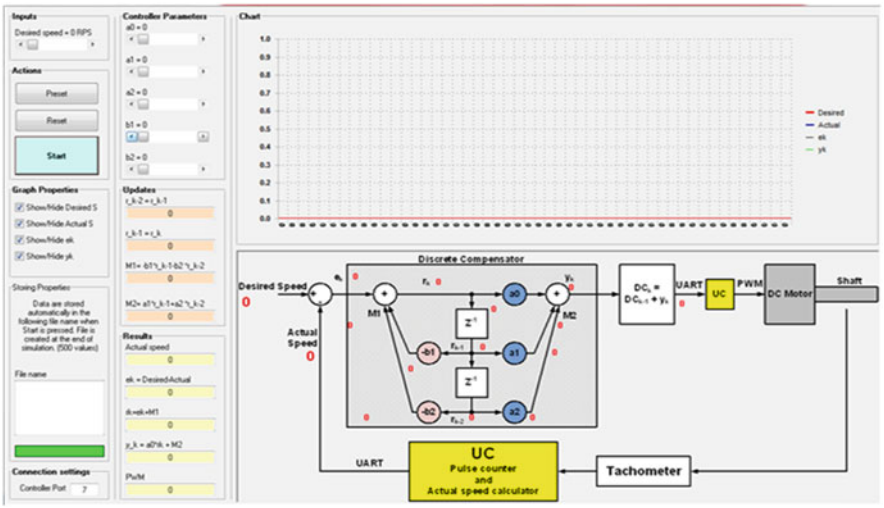
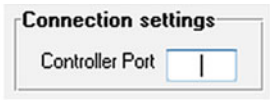
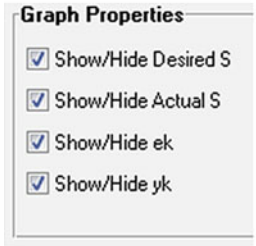


Fig. 17 Comprehensive GUI interface

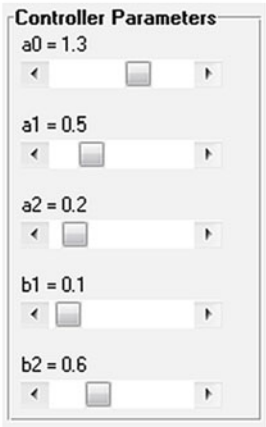


Check the communication port if properly set and connected

Set the desired speed



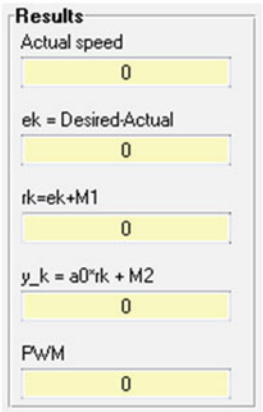
Set what is needed to be drawn on the graph



Set the controller parameters.

Fig. 18 Display properties and compensator parameters

A "Start" button is available to initiate the system.



The calculated results will be shown

Fig. 19 Compensator options and tuning

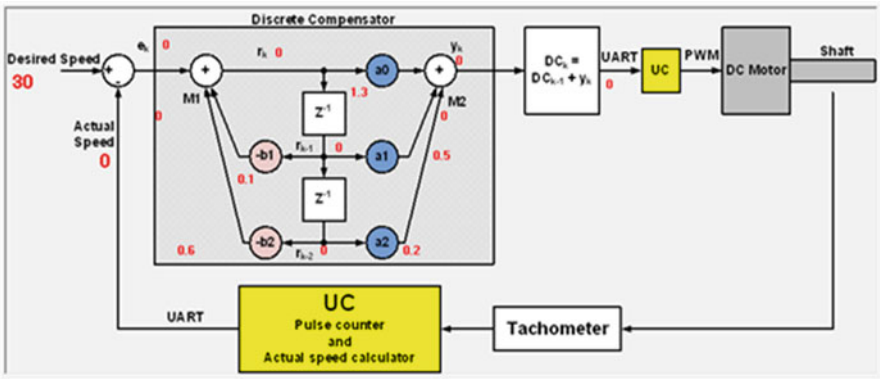
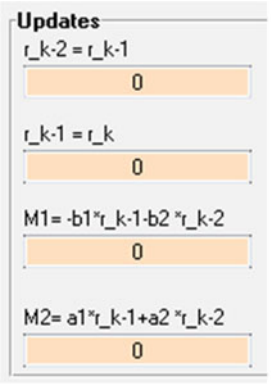


Fig. 20 Discrete compensator

8 Experimental Results

Several experimental tests were carried out that demonstrated the effectiveness of the overall framework in terms of the dynamic and steady-state behavior of the system. Typical situations are presented below. The first one involves a speed control with no load disturbances. Figure 22 shows the motor's response to a step in speed set at 30 rps. The behavior is quite satisfactory and overshoot is noted as previewed by the compensator's parameters.

The second typical situation involves the application of a sudden load or disturbance at different instants. Figure 23 clearly shows that the compensator was able to handle the speed control by keeping track of the desired speed after slight



Internal calculation of the system in order to generate the corresponding PWM.

The results of the experiment are saved in a excel file in which the user can check after the simulation finishes (500 samples)

Fig. 21 Online computation of related parameters; saving results

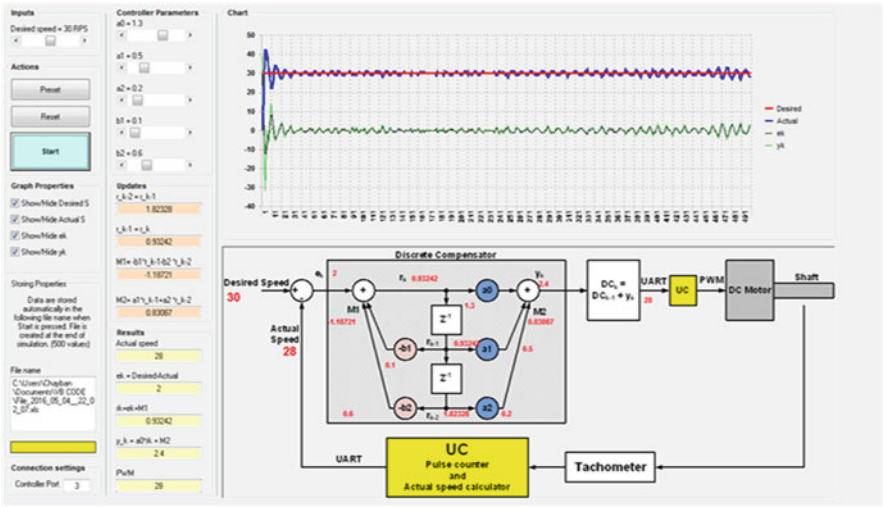


Fig. 22 Speed control (no load applied)

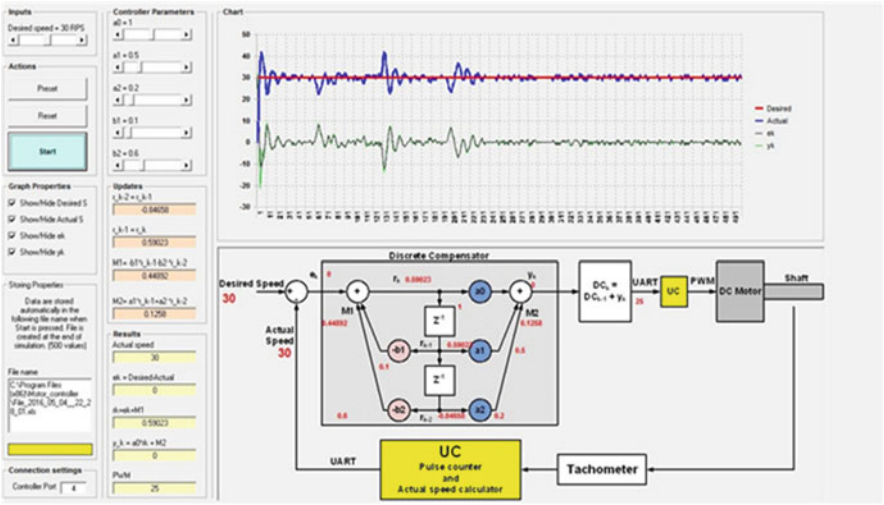


Fig. 23 Speed control with load disturbance

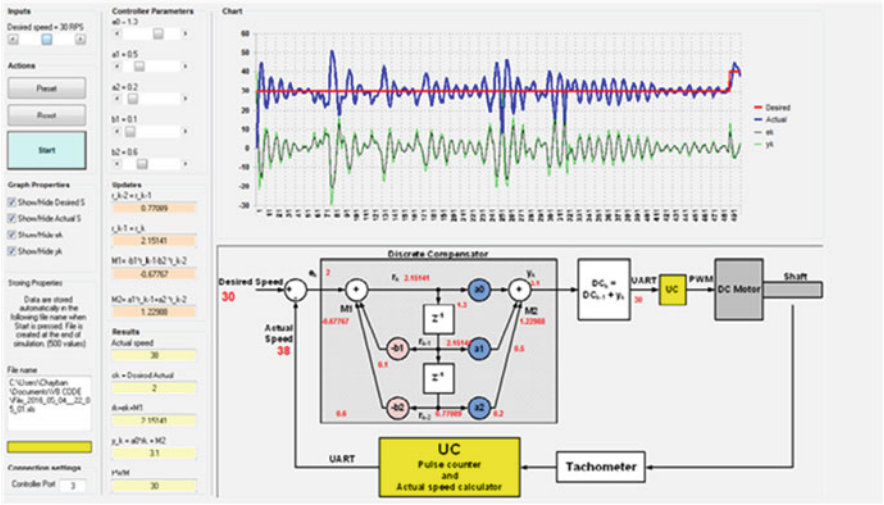


Fig. 24 Speed control (heavy load)

deviation and oscillations. In fact, new PWM values were generated in order to adjust the speed; those values are shown online via the GUI and then saved with the related speed curves.

Figure 24 shows a heavier load was applied to the shafts of the motor. The motor would still run at the desired speed. However, since the applied load was heavy (the shafts were almost about to stop as per the above graph), greater oscillations are observed but the motor did not halt and the average speed was maintained.

9 Conclusion

In this chapter, we developed a comprehensive design of a hardware and software framework for PWM/Discrete PID-based speed control of a Permanent-Magnet DC Motor without prior knowledge of the motor's parameters. Ziegler–Nichols approach associated with Direct Canonical Forms theory allowed tuning a discrete PID compensator to achieve speed control that is robust to load torque disturbances and able to meet requirements in terms of steady-state error, time response, and percent overshoot. Simulation and experimental results showcased the effectiveness of the design that is low cost and somehow simple to implement.

References

1. Buja, G.S. and Kazmierkowski, M.P., 2004. Direct torque control of PWM inverter-fed AC motors-a survey. IEEE Transactions on industrial electronics, 51(4), pp.744–757.
2. Ogasawara, S., Akagi, H. and Nabae, A., 1990. A novel PWM scheme of voltage source inverters based on space vector theory. Archiv für Elektrotechnik, 74(1), pp.33–41.
3. Kazmierkowski, M.P. and Malesani, L., 1998. Current control techniques for three-phase voltage-source PWM converters: A survey. IEEE Transactions on industrial electronics, 45(5), pp.691–703.
4. Öztürk, N., Kaplan, O. and Çelik, E., 2017. Zero-current switching technique for constant voltage constant frequency sinusoidal PWM inverter. Electrical Engineering, pp.1–11.
5. Leon, J.I., Kouro, S., Franquelo, L.G., Rodriguez, J. and Wu, B., 2016. The essential role and the continuous evolution of modulation techniques for voltage-source inverters in the past, present, and future power electronics. IEEE Transactions on Industrial Electronics, 63(5), pp.2688–2701.
6. Johnson, M.A. and Moradi, M.H., 2005. PID control. Springer-Verlag London Limited.
7. Hamamci, S.E., 2008. Stabilization using fractional-order PI and PID controllers. Nonlinear Dynamics, 51(1), pp.329–343.
8. Vilanova, R. and Visioli, A., 2012. PID control in the third millennium. London: Springer.
9. Lin, C.L. and Jan, H.Y., 2005. An approach to solving for multi-objective optimization problem with application to linear motor control design. Control and intelligent systems, 33(2), pp.75–86.
10. Hendy, H., Rui, X., Zhou, Q. and Khalil, M., 2014. Controller parameters tuning based on transfer matrix method for multibody systems. Advances in Mechanical Engineering, 6, p.957684.
11. Mudi, R.K. and Pal, N.R., 1999. A robust self-tuning scheme for PI-and PD-type fuzzy controllers. IEEE Transactions on fuzzy systems, 7(1), pp.2–16.
12. Åström, K.J. and Hägglund, T., 2004. Revisiting the Ziegler–Nichols step response method for PID control. Journal of process control, 14(6), pp.635–650.
13. Åström, K.J. and Hägglund, T., 1984. Automatic tuning of simple regulators with specifications on phase and amplitude margins. Automatica, 20(5), pp.645–651.
14. Barbosa, R.S., Machado, J.T. and Ferreira, I.M., 2004. Tuning of PID controllers based on Bode's ideal transfer function. Nonlinear dynamics, 38(1), pp.305–321.
15. Åström, K.J. and Hägglund, T., 2006. PID control. IEEE CONTROL SYSTEMS MAGAZINE, 1066(033X/06).

16. Chemuturi, Murali. Requirements Engineering and Management for Software Development Projects. New York: Springer, 2013.
17. Carriegos, M. and Hermida-Alonso, J.A., 2003. Canonical forms for single input linear systems. Systems & control letters, 49(2), pp.99–110.