UNIVERSITI TEKNOLOGI MALAYSIA

BORANG PENGESAHAN STATUS TESIS♦

JUDUL : **DC MOTOR SPEED CONTROL USING MICROCONTROLLER PIC 16F877A**

SESI PENGAJIAN : **2004/2005**

Saya : **EA AI CHOON**
(HURUF BESAR)

mengaku membenarkan tesis (PSM/~~Sarjana/Doktor Falsafah~~)* ini disimpan di Perpustakaan Universiti Teknologi Malaysia dengan syarat-syarat kegunaan seperti berikut :

1. Tesis adalah hakmilik Universiti Teknologi Malaysia.
2. Perpustakaan Universiti Teknologi Malaysia dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis in sebagai bahan pertukaran antara institusi pengajian tinggi.
4. **Sila tandakan (✓)

☐	SULIT	(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)
☐	TERHAD	(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)
☑	TIDAK TERHAD	

(TANDATANGAN PENULIS)

Alamat Tetap :

391 JALAN YUSOF GRISEK

84700 MUAR

JOHOR

Tarikh : **1 APRIL 2005**

Disahkan oleh

(TANDATANGAN PENYELIA)

EN. MOHAMAD SHUKRI BIN ABDUL MANAF

NAMA PENYELIA

Tarikh : **1 APRIL 2005**

CATATAN :
- * Potong yang tidak berkenaan.
- ** Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/ organisasi berkenaan dengan menyatakan sekali tempoh tesis ini perlu dikelaskan sebagai SULIT atau TERHAD.
- ♦ Tesis dimaksudkan sebagai tesis bagi Ijazah Doktor Falsafah dan Sarjana secara penyelidikan, atau disertasi bagi pengajian secara kerja kursus dan penyelidikan, atau Laporan Projek Sarjana Muda (PSM).

"I declare that I have read this work and in
my opinion this work is adequate in terms of scope and
quality for the purpose of awarding a Bachelor's Degree of
Electrical Engineering (Instrumentation & Control)."

Signature : .....

Name of Supervisor : En. Mohamad Shukri bin Abdul Manaf

Date : 31 MARCH 2005

DC MOTOR SPEED CONTROL USING MICROCONTROLLER PIC 16F877A

EA AI CHOON

Submitted to the Faculty of Electrical Engineering
in partial fulfillment of the requirement for the degree of
Bachelor in Electrical Engineering (Instrumentation & Control)

Faculty of Electrical Engineering
Universiti Teknologi Malaysia

MARCH 2005

"I declare that this work as the product of my own effort with the exception of excerpts cited from other works of which the sources were duly noted"

Signature	:	.....
Author's Name	:	EA AI CHOON
Date	:	1 April 2005

Dedicated, in thankful appreciation for support, encouragement and understandings to my beloved mother, father, brothers and sisters.

ACKNOWLEDGEMENT

First and foremost, I would like to express my heartily gratitude to my supervisor, Encik Mohamad Shukri bin Abdul Manaf for the guidance and enthusiasm given throughout the progress of this project.

My appreciation also goes to my family who has been so tolerant and supports me all these years. Thanks for their encouragement, love and emotional supports that they had given to me.

I would also like to thank our Control 1 Lab Assistant, Encik Shaari bin Ani and Encik Mad Rais bin Zakaria for their co-operations, guidance and helps in this project.

Nevertheless, my great appreciation dedicated to my best friends Jie, Pika, Choo Chin, Ching Chiew, Yan Chiew, Meng Yea, Say Moon and SEI member's batch 2000 and those whom involve directly or indirectly with this project. There is no such meaningful word than…..Thank You So Much.

ABSTRACT

Direct current (DC) motor has already become an important drive configuration for many applications across a wide range of powers and speeds. The ease of control and excellent performance of the DC motors will ensure that the number of applications using them will continue grow for the foreseeable future. This project is mainly concerned on DC motor speed control system by using microcontroller PIC 16F877A. It is a closed-loop real time control system, where optical encoder (built in this project) is coupled to the motor shaft to provide the feedback speed signal to controller. Pulse Width Modulation (PWM) technique is used where its signal is generated in microcontroller. Microcontroller acts as proportional (*P*) controller with K_p =1 in this study. The PWM signal will send to motor driver to vary the voltage supply to motor to maintain at constant speed. A program in Visual Basic 6.0 is developed to provide a graphic user interface (GUI) for the user to enter desired speed at computer. Besides, it also shows a graph of motor speed versus time to let the user monitor the performance of the system easily. Based on the result, the reading of optical encoder built is quite reliable. Through the project, it can be concluded that microcontroller PIC 16F877A can control motor speed at desired speed although there is a variation of load.

ABSTRAK

Motor arus terus telah menjadi satu komponen yang penting untuk aplikasi dalam julat kuasa dan kelajuan yang tinggi. Kawalan motor arus terus yang mudah dan prestasi yang baik akan menjamin motor arus terus untuk digunakan secara meluas pada masa depan. Projek ini tertumpu kepada rekaaan satu sistem kawalan kelajuan motor arus terus dengan menggunakan mikropengawal PIC 16F877A. Ia merupakan satu kawalan gelung tutup dan masa nyata sistem, di mana pengekod optic(yang dibina dalam projek ini) yang dipasang pada rotor akan menghantar isyarat kelajuan suapbalik kepada pengawal. Teknik *Pulse Width Modulation (PWM)* digunakan di mana isyarat ini dibekal oleh mikro pengawal. Mikropengawal berperanan sebagai pengawal gandaan, *P* di dalam projek ini. Isyarat *PWM* akan dihantar kepada pemacu motor untuk mengubah voltan yang dibekalkan kepada motor supaya ia dapat dikawal pada kelajuan yang ditetapkan. Satu program ditulis dalam *Visual Basic 6.0* untuk memudahkan pengguna memasukkan kelajuan yang dikehendaki di komputer dengan pengantaramuka grafik pengguna *(GUI).* Selain itu, ia juga memaparkan satu graf kelajuan motor melawan masa untuk membolehkan pengguna mengkaji prestasi sistem tersebut. Keputusan menunjukkan bahawa bacaan daripada pengekod optik yang dibina adalah boleh dipercayai. Melalui projek ini, boleh disimpulkan bahawa mikropengawal PIC 16F877A dapat mengawal kelajuan motor pada kelajuan tetap walaupun terdapat perubahan beban.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

K_E	-	A constant based on motor construction
φ	-	Magnetic flux
I_f	-	Field current
I_a	-	Armature current
R_f	-	Field resistor
L_f	-	Field inductor
R_a	-	Armature resistor
L_a	-	Armature inductor
K_v	-	Motor constant
K_t	-	Torque constant
T_d	-	Developed torque
T_L	-	Load torque
B	-	Viscous friction constant
J	-	Inertia of the motor
w	-	Motor speed
α	-	Firing angle of thyristor
t_{on}	-	Time ON of switches
T	-	Period
s	-	Standard deviation
rpm	-	Rotation per minute

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

Direct current (DC) motors have variable characteristics and are used extensively in variable-speed drives. DC motor can provide a high starting torque and it is also possible to obtain speed control over wide range. Why do we need a speed motor controller? For example, if we have a DC motor in a robot, if we just apply a constant power to each motor on a robot, then the poor robot will never be able to maintain a steady speed. It will go slower over carpet, faster over smooth flooring, slower up hill, faster down hill, etc. So, it is important to make a controller to control the speed of DC motor in desired speed.

DC motor plays a significant role in modern industrial. These are several types of applications where the load on the DC motor varies over a speed range. These applications may demand high-speed control accuracy and good dynamic responses.

In home appliances, washers, dryers and compressors are good examples. In automotive, fuel pump control, electronic steering control, engine control and electric vehicle control are good examples of these. In aerospace, there are a number of applications, like centrifuges, pumps, robotic arm controls, gyroscope controls and so on.

1.2 Objective of Project

The main core of this project is to design a speed control system of DC Motor by using microcontroller. This system will be able to control the DC motor speed at desired speed regardless the changes of load.

1.3 Scope of Project

In order to achieve the objective of the project, there are several scope had been outlined. The scope of this project includes using MPLAB IDE to program microcontroller PIC 16F877A, build hardware for the system, and interface the hardware to computer by using RS232 serial port communication. Last but not least, a graph of speed versus time is obtained by using Visual Basic 6.0 at computer to observe the performance of the system.

1.4 Outline of Thesis

This thesis consists four chapters. In first chapter, it discuss about the objective and scope of this project as long as summary of works. While Chapter 2 will discuss more on theory and literature reviews that have been done. It well discuss about types of motor, various kind of speed measurement and controllers (thyristor, phase-lock loop and PWM technique) that can be used to control the speed of the motor.

In Chapter 3, the discussion will be on the methodology hardware and software implementation of this project. The result and discussion will be presented in Chapter 4. Last but not least, Chapter 5 discusses the conclusion of this project and future work that can be done.

1.5. Summary of Works

Implementation and works of the project are summarized into the flow chart as shown in Figure 1.1. Gantt charts as shown in Figure 1.2 and Figure 1.3 show the detail of the works of the project that had been implemented in the first and second semester.

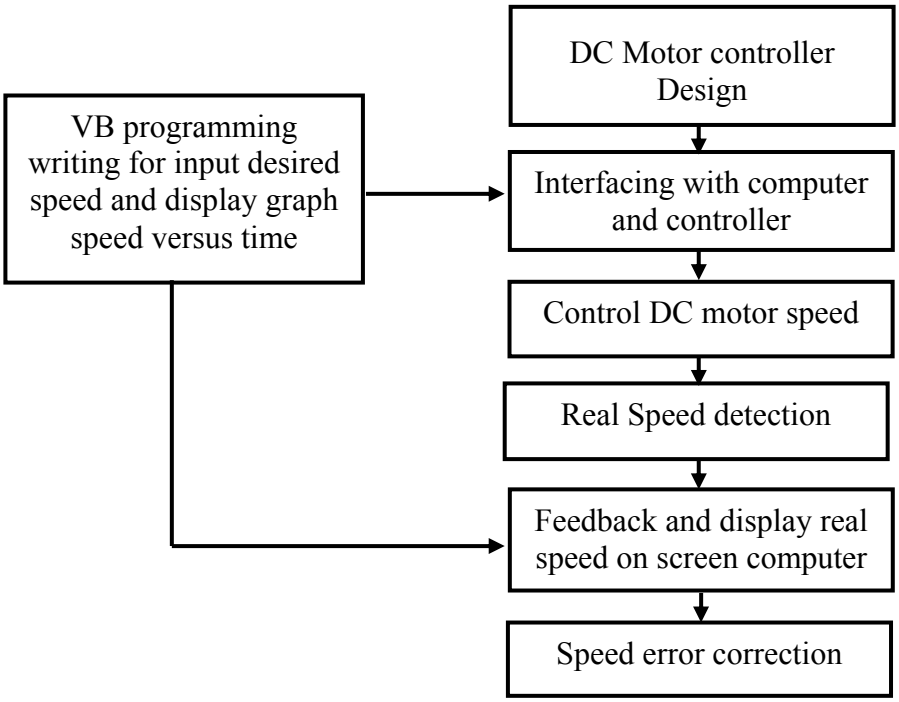


Figure 1.1 Project overview

Activities \ Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1. Brief idea								Semester Break								Study Week	Exam Week	
2. Literature and theoritical study																		
3. Study MPLAB																		
4. Hardware design																		
5. Order part of hardware																		
6. Construction of hardware																		
7. Report preparation																		
8. Presentation																		

Figure 1.2 Gantt Chart of the project schedule for semester 1

Activities \ Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1. Literature and theoritical study								Semester break									
2. Construction of hardware																	
3. Programming for plotting graph																	
4. Interfacing with computer																	
5. Modification and evaluation																	
6. Results, discussion & conclusion																	
7. Presentation																	
8. Report preparation																	

Figure 1.3 Gantt Chart of the project schedule for semester 2

CHAPTER 2

THEORY AND LITERATURE REVIEW

2.1 Introduction

This chapter includes the study of different types of DC motors, speed measurement of DC motor, model of separately excited DC motor, several types of DC motor speed controller. It also brief discuss about microcontroller, RS232 serial port, MPLAB IDE and Visual Basic 6.0.

2.2 DC Motor

There are several types of DC motors that are available. Their advantages, disadvantages, and other basic information are listed below in the Table 2.1.

Table 2.1 Advantages and disadvantages of various types of DC motor.

Type	**Advantages**	**Disadvantages**
Stepper Motor	Very precise speed and position control. High Torque at low speed.	Expensive and hard to find. Require a switching control circuit
DC Motor w/field coil	Wide range of speeds and torques. More powerful than permanent magnet motors	Require more current than permanent magnet motors, since field coil must be energized. Generally heavier than permanent magnet motors. More difficult to obtain.
DC permanent magnet motor	Small, compact, and easy to find. Very inexpensive	Generally small. Cannot vary magnetic field strength.
Gasoline (small two stroke)	Very high power/weight ratio. Provide Extremely high torque. No batteries required.	Expensive, loud, difficult to mount, very high vibration.

2.3 Speed Measurement of DC Motor

To start with this project, we need a device that will measure the speed of the motor shaft. There are several methods which can use to measure the speed of motor. Here, we will only discuss about speed measurement by using tachometer and optical encoder.

2.3.1 Speed Measurement by Using Tachometer

Tachometer is an instrument that measure speed motor based on concept of back EMF induced in motor when it is running. The EMF is voltages appear on the commutator segments caused by rotated in the magnetic field by some external force.

The magnitude of the EMF is given by [1],

$$EMF = K_E \phi N \tag{2.1}$$

where K_E = a constant based on motor construction

φ = magnetic flux

N = speed of motor (in rpm)

The actual relationship between motor speed and EMF follows and is derived from Equation 2.1,

$$N = \frac{EMF}{K_E \phi} \tag{2.2}$$

Thus, the motor speed is directly proportional to the EMF voltage ad inversely proportional to the field flux. For permanent magnet DC motor, when the EMF measured is increases, the speed of the motor is also increases with the gain. So, the speed of motor can be measured by measuring the back EMF using tachometer.

2.3.2 Speed Measurement by Using Optical Encoder

The best way to measure speed is to fit an optical encoder. This shines a beam of light from a transmitter across a small space and detects it with a receiver the other end. If a disc is placed in the space, which has slots cut into it, then the signal will only be picked up when a slot is between the transmitter and receiver. An example of a disc is shown as Figure 2.1.

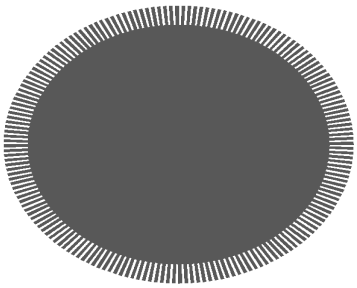


Figure 2.1 Sample disc of encoder

The encoder transmitter must be supplied with a suitable current, and the receiver biased as Figure 2.2.

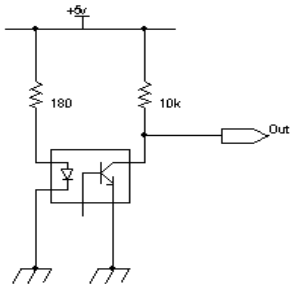


Figure 2.2 Basic schematic circuit of optical encoder

This will have an output which swings to +5v when the light is blocked, and about 0.5 volts when light is allowed to pass through the slots in the disc. The frequency of the output waveform is given by,

$$f_{out} = \frac{N \times rpm}{60} \tag{2.3}$$

where f_{out} = frequency of output waveform
rpm = speed in revolutions per minutes
N = number of slots at disc

So, from Equation 2.3, the speed of DC motor in rpm is given by,

$$rpm = \frac{f_{out} \times 60}{N} \tag{2.4}$$

2.4 Model of Separately Excited DC motor

Figure 2.3 shows a model of separately excited DC motor [1]. When a separately excited motor is excited by a field current of I_f and an armature current of I_a flows in the circuit, the motor develops a back EMF and a torque to balance the load torque at a particular speed. The I_f is independent of the I_a. Each winding are

supplied separately. Any change in the armature current has no effect on the field current. The I_f is normally much less than the I_a. The relationship of the field and armature are shown in Equation 2.5.

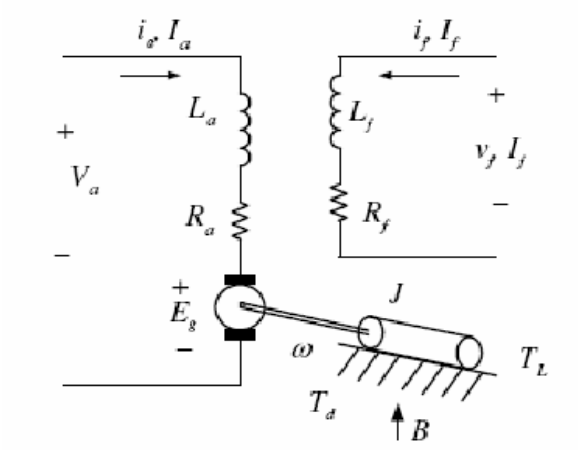


Figure 2.3 Model of separately excited DC motor

Instantaneous field current:

$$v_f = R_f i_f + L_f \frac{di_f}{dt} \tag{2.5}$$

where R_f and L_f are the field resistor and inductor respectively.

Instantaneous armature current:

$$v_a = R_a i_a + L_a \frac{di_a}{dt} + e_g \tag{2.6}$$

where R_a and L_a are the armature resistor and inductor respectively.

The motor back EMF which is also known as speed voltage is expressed as

$$e_g = K_v w i_f \tag{2.7}$$

where K_v is the motor constant (in V/A-rad/s) and w is the motor speed (rad/s).

The torque developed by the motor is

$$T_d = K_t \phi i_f \tag{2.8}$$

where (K_t=K_v) is the torque constant (in V/A-rad/s).

Sometimes it is written as:

$$T_d = K_t \phi i_a \tag{2.9}$$

For normal operation, the developed torque must be equal to the load torque plus the friction and inertia, i.e.:

$$T_d = J\frac{dw}{dt} + Bw + T_L \tag{2.10}$$

where B = viscous friction constant (N.m/rad/s)

T_L = load torque (N.m)

J = inertia of the motor (kg.m^2)

Under steady-state operations, a time derivative is zero. Assuming the motor is not saturated.

For field circuit,

$$V_f = I_f R_f \tag{2.11}$$

The back EMF is given by:

$$E_g = K_v w I_f \tag{2.12}$$

The armature circuit,

$$V_a = I_a R_a + E_g = I_a R_a + K_v w I_f \tag{2.13}$$

The motor speed can be easily derived:

$$w = \frac{V_a - I_a R_a}{K_v I_f} \tag{2.14}$$

If R_a is a small value (which is usual), or when the motor is lightly loaded, i.e. I_a is small,

$$w = \frac{V_a}{K_v I_f} \tag{2.15}$$

That is if the field current is kept constant, the speed motor speed depends on the supply voltage. These observation leads to the *application of variable DC voltage to control the speed* and torque of DC motor.

2.5 DC motor Speed Controller

For precise speed control of servo system, closed-loop control is normally used. Basically, the block diagram and the flow chart of the speed control are shown in Figure 2.4 and Figure 2.5 respectively. The speed, which is sensed by analog sensing devices (e.g., tachometer) is compared with the reference speed to generate the error signal and to vary the armature voltage of the motor.

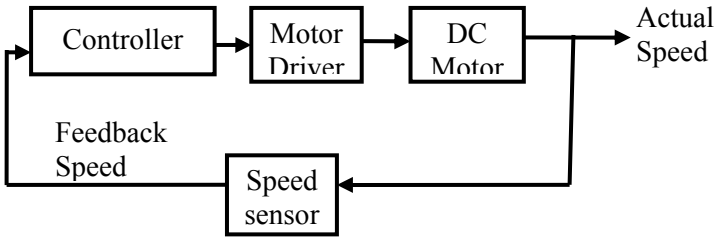


Figure 2.4 Basic block diagram for DC Motor speed control

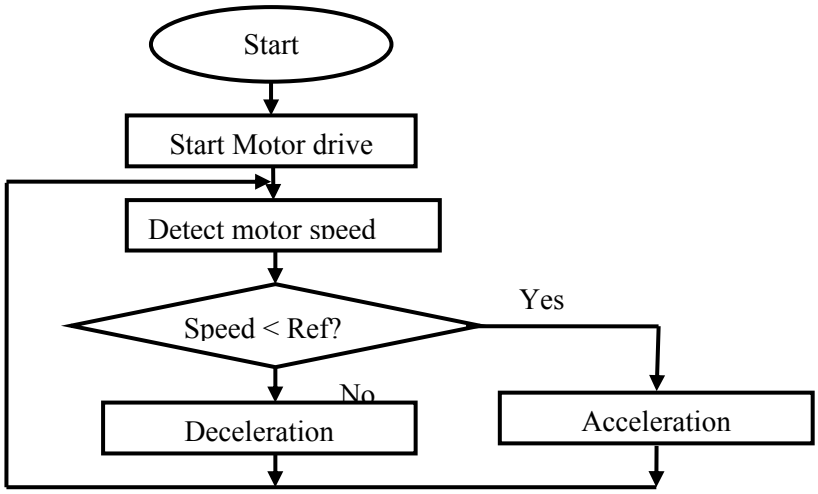


Figure 2.5 Basic flow chart of DC motor speed control

There are several controllers that can used to control the speed of the motor such as by using thyristor, phase-locked-loop control, chopper circuit, Fuzzy Logic Controller and etc. Here, we will discuss only at the speed control system by using thyristor, phase-locked loop and PWM technique.

2.5.1 Phase-Locked-Loop (PLL) Control

The block diagram of a converter-fed dc motor drive with phase-locked-loop control is shown in Figure 2.6. In a phase-locked-loop (PLL) control system, the motor speed is converted to a digital pulse train by using a speed encoder. The output of the encoder acts as the speed feedback signal of frequency f_0.

The phase detector compares the reference pulse train (or frequency) f_r with the feedback frequency f_0 and provides a pulse-width-modulated (PWM) output voltage V_e that is proportional to the difference in phases and frequencies of the reference and feedback pulse trains. The phase detector (or comparator) is available in integrated circuits. A low-pass loop filter converts the pulse train V_e to continuous dc level V_c, which varies the output of the power converter and in turn the motor speed.

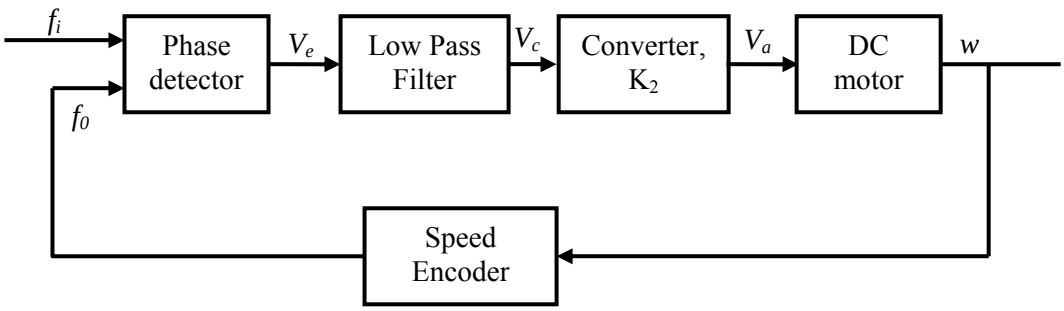


Figure 2.6 Phase-locked loop control system

When the motor runs at the same speed as the reference pulse train, the two frequencies would be synchronized (or locked) together with a phase difference. The output of the phase detector would be a constant voltage proportional to the phase difference and the steady-state motor speed would be maintained at a fixed value irrespective of the load on the motor.

Any disturbances contributing to the speed change would result in a phase difference and the output of the phase detector would respond immediately to vary the speed of the motor in such direction and magnitude as to retain the locking of the reference and feedback frequencies. The response of the phase detector is very fast.

As long as the two frequencies are locked, the speed regulation should ideally be zero.

In journals by Christopher A. Adkins and Moore respectively, it has developed a model for the components of PLL servo control system using both linear and nonlinear techniques [2], [3]. However, PLL controlled motor drives have the following shortcomings.

i. PLL-controlled motor system tend to be unstable for low-speed operation
ii. PLL-controlled motor systems have large response time.
iii. PLL controlled motor systems may get out of synchronization for an abrupt load variation.

2.5.2 Speed Control by Using Thyristor

Figure 2.8 shows the block diagram of DC motor speed control by using thyristor. The thyristor is used to supply a variable DC voltage to motor, thus it can control the speed of motor (Equation 2.15). The average output of voltage is given by

$$V_{ave} = \frac{V_m}{2\pi}(1+\cos\alpha) \qquad (2.16)$$

where V_m = peak voltage of voltage supply of thyristor and
α = firing angle of thyristor

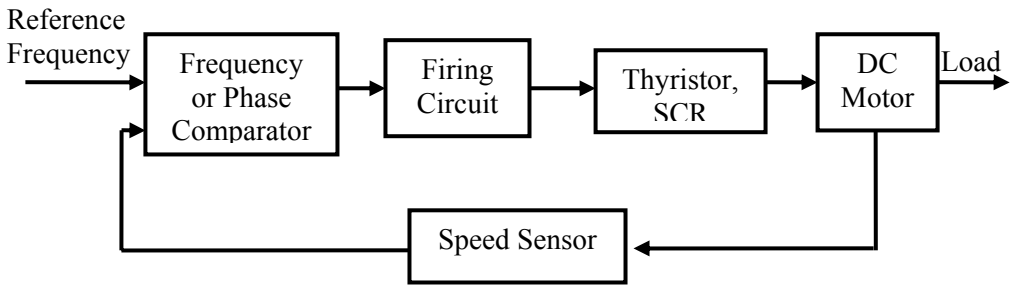


Figure 2.7 Block diagram of DC Motor speed control by using thyristor

From Equation 2.16, by controlling the firing angle, α, the average of output DC voltage can be varied. If the motor speed is low, the speed sensor frequency will be below the reference frequency. The frequency difference produces a change in the firing circuit that causes the thyristor, SCR to fire sooner (firing angle, α is reduced). There is a resulting increase in motor speed which brings the output speed back up to the value which is equal to the reference signal.

Conversely, if the speed sensor output frequency is above the reference, then the firing circuit will be modified to allow the SCR to conduct for a shorter period of time, the decrease in conduction reduces the DC motor speed.

From review, this method has been used by P.C. Sen and M.L. MacDonald in their research [4].

2.5.3 Speed Control by Using PWM and Full H Bridge Motor Drive

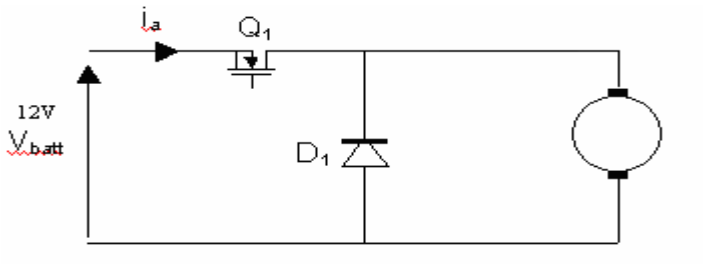


Figure 2.8 Simple motor circuit

Let us consider a simple circuit that connects a battery as power supply through a switch MOSFET (Metal-Oxide-Semiconductor Field Effect Transistor) as shown in Figure 2.8 [5]. When the switch is closed, the motor sees 12 Volts, and when it is open it sees 0 Volts. If the switch is open for the same amount of time as it is closed, the motor will see an average of 6 Volts, and will run more slowly accordingly.

This *on-off* switching is performed by power MOSFETs. A MOSFET (Metal-Oxide-Semiconductor Field Effect Transistor) is a device that can turn very large currents on and off under the control of a low signal level voltage.

The average of voltage that supply to DC motor is given by,

$$V_{ave} = \frac{t_{on}}{T} \times V_{in} \qquad (2.17)$$

where V_{ave} = average voltage supply to DC motor

t_{on} = time ON of switches

T = period of PWM

$\frac{t_{on}}{T} = DC$, duty cycle

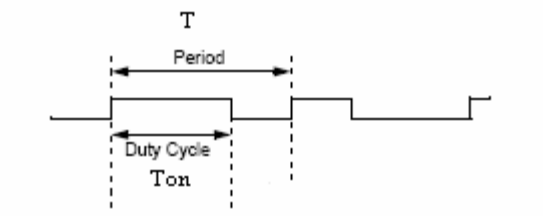


Figure 2.9 PWM signal

As the amount of time that the voltage is *on* increases compared with the amount of time that it is *off*, the average speed of the motor increases and vice versa.

The time that it takes a motor to speed up and slow down under switching conditions is depends on the inertia of the rotor (basically how heavy it is), and how much friction and load torque there is. Figure 2.10 shows the speed of a motor that is being turned on and off fairly slowly:

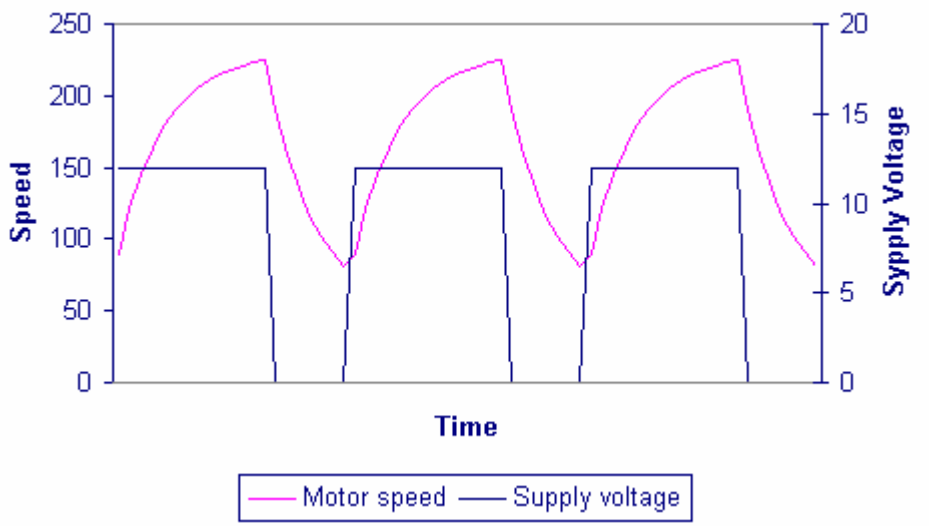


Figure 2.10 Relation of supply voltage with motor speed

We can see that the average speed is around 150 rpm, although it varies quite a bit. If the supply voltage is switched fast enough, it won't have time to change speed much, and the speed will be quite steady. This is the principle of switch mode speed control. Thus the speed is set by PWM – Pulse Width Modulation.

A full bridge circuit is shown in the diagram below. Each side of the motor can be connected either to battery positive, or to battery negative. Only one MOSFET on each side of the motor must be turned on at any one time otherwise they will short out the battery and burn out.

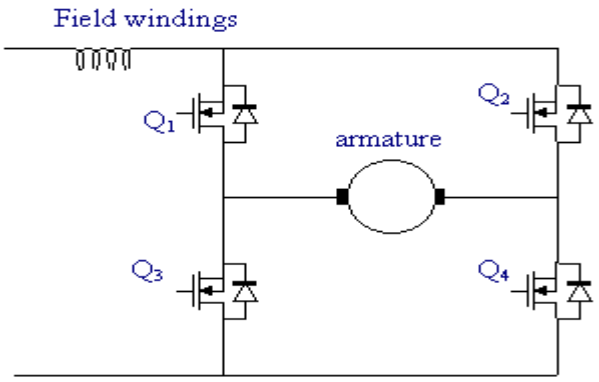


Figure 2.11 Full H bridge motor drive

To make the motor go forwards, Q4 is turned on, and Q1 has the PWM signal applied to it. Meanwhile, to make the motor go backwards, Q3 is turned on, and Q2 has the PWM signal applied to it:

From review, this method has been used by Abu Zaharin Ahmad and Mohd Nasir Taib in their study [6].

2.6 Microcontroller

Microcontrollers must contain at least two primary components – random access memory (RAM), and an instruction set. RAM is a type of internal logic unit that stores information temporarily. RAM contents disappear when the power is turned off. While RAM is used to hold any kind of data, some RAM is specialized, referred to as registers. The instruction set is a list of all commands and their corresponding functions. During operation, the microcontroller will step through a program (the firmware). Each valid instruction set and the matching internal hardware that differentiate one microcontroller from another [7].

Most microcontrollers also contain read-only memory (ROM), programmable read-only memory (PROM), or erasable programmable read-only memory (EPROM). Al1 of these memories are permanent: they retain what is programmed into them even during loss of power. They are used to store the firmware that tells the microcontroller how to operate. They are also used to store permanent lookup tables. Often these memories do not reside in the microcontroller; instead, they are contained in external ICs, and the instructions are fetched as the microcontroller runs. This enables quick and low-cost updates to the firmware by replacing the ROM.

Where would a microcontroller be without some way of communicating with the outside world? This job is left to input/output (I/O) port pins. The number of I/O pins per controllers varies greatly, plus each I/O pin can be programmed as an input or output (or even switch during the running of a program). The load (current draw)

that each pin can drive is usually low. If the output is expected to be a heavy load, then it is essential to use a driver chip or transistor buffer.

Most microcontrollers contain circuitry to generate the system clock. This square wave is the heartbeat of the microcontroller and all operations are synchronized to it. Obviously, it controls the speed at which the microcontroller functions. All that needed to complete the clock circuit would be the crystal or RC components. We can, therefore precisely select the operating speed critical to many applications.

To summarize, a microcontroller contains (in one chip) two or more of the following elements in order of importance [8]:

i. Instruction set
ii. RAM
iii. ROM,PROM or EPROM
iv. I/O ports
v. Clock generator
vi. Reset function
vii. Watchdog timer
viii. Serial port
ix. Interrupts
x. Timers
xi. Analog-to-digital converters
xii. Digital-to-analog converters

2.7 RS232 Serial Port

RS232 is a popular communications protocol for connecting modems and data acquisition devices to computers. RS232 devices can be plugged straight into the computer's serial port (also known as the COM or Comms port). Examples of

data acquisition devices include GPS receivers, electronic balances, data loggers, temperature interfaces and other measurement instruments.

A nine pin D plug has become the standard fitting for the serial ports of PCs. The pin connections used are as shown in Table 2.2. The connector on the PC has male pins, therefore the mating cable needs to terminate in a DB9/F (Female pin) connector.

Table 2.2 RS232 pin assignments (DB9 PC signal set)

Pin 1	Input	DCD	Data Carrier Detect
Pin 2	Input	RXD	Received Data
Pin 3	Output	TXD	Transmitted Data
Pin 4	Output	DTR	Data Terminal Ready
Pin 5			Signal Ground
Pin 6	Input	DSR	Data Set Ready
Pin 7	Output	RTS	Request To Send
Pin 8	Input	CTS	Clear To Send
Pin 9	Input	RI	Ring Indicator

Normal PC hardware might well run with just T_x, R_x and Ground connected, most driver software will wait forever for one of the handshaking lines to go to the correct level. Depending on the signal state it might sometimes work, other times it might not. The reliable solution is to loop back the handshake lines if they are not used [9].

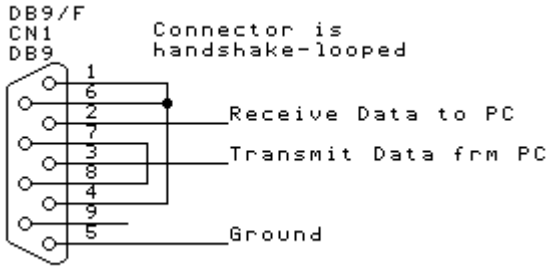


Figure 2.12 Handshake looping a PC serial connector

When the lines are handshake looped, the RTS output from the PC immediately activates the CTS input - so the PC effectively controls its own handshaking.

2.8 MPLAB IDE

MPLAB IDE is a Windows-based Integrated Development Environment for the Microchip Technology Incorporated PIC microcontroller (MCU) and dsPIC digital signal controller (DSC) families [10]. In the MPLAB IDE, we can:

i. Create source code using the built-in editor.
ii. Assemble, compile and link source code using various language tools. An assembler, linker and librarian come with MPLAB IDE. C compilers are available from Microchip and other third party vendors.
iii. Debug the executable logic by watching program flow with a simulator, such as MPLAB SIM, or in real time with an emulator, such as MPLAB IDE. Third party emulators that work with MPLAB IDE are also available.
iv. Make timing measurements.
v. View variables in Watch windows.
vi. Program firmware into devices with programmers such as PICSTART Plus or PRO MATE II.

2.9 Visual Basic 6.0

There are literally hundreds of programming languages. Each was developed to solve particular type of problem. Most traditional languages, such as BASIC, C, COBOL, FORTRAN, and Pascal are considered procedural languages. That is, the program specifies the exact sequence of all operations. Program logic determines the next instruction to execute in response to conditions and user request.

The newer language, such as C++ and VISUAL BASIC, use a different approach: object-oriented programming (OOP) and event driven programming. Microsoft refers to Visual Basic as an event driven programming language, which has many elements of an object oriented language such as Java. In the event driven model, programs are no longer procedural; they don't follow a sequential logic. So, there is no need to take control and determine the sequence of execution.

As the world turn to graphic user interface (GUI), visual basic is one of the languages that changes to accommodate the shift. Visual Basic is designed to allow the program run under the windows without the complexity generally associated with windows programming [11]. The designed screen can holds standard windows button such as command buttons, check boxes, option buttons, text boxes, and so on. Each of these windows object, operates as expected, producing a "standard" windows user interface.

Visual Basic that recently appears as one of the most popular programming language is chose. It provided standard windows object and graphic user interface that will make the program become user friendly.

CHAPTER 3

METHODOLOGY

3.1 Introduction

In this project, microcontroller will be used as the controller to control DC motor speed at desired speed. The block diagram of the system is shown in Figure 3.1. It is a closed-loop with real time control system.

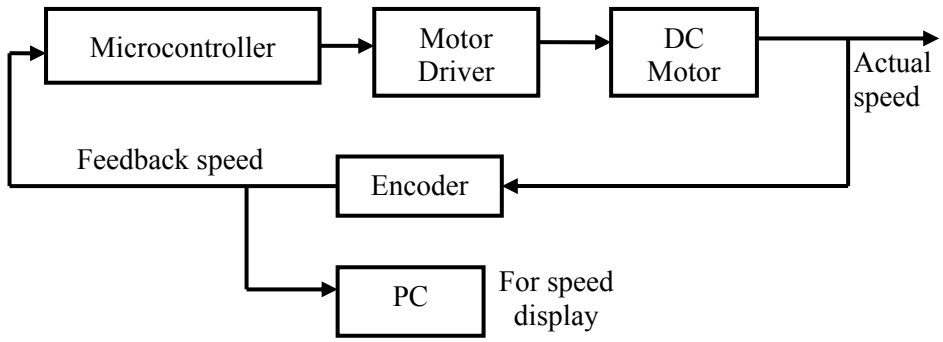


Figure 3.1 Block diagram of DC motor speed control system

The actual speed of DC motor will be measured by encoder and feedback to microcontroller. In microcontroller, it will calculate the error between the desired speed with the actual speed. The error will determine duty cycle of pulse-width-modulation (PWM) in microcontroller. Then, the duty cycle will send to DC motor driver either accelerate or decelerate DC motor to maintain it at desired speed.

Figure 3.2 shows the picture of the project. The project is divides into two parts that are software and hardware implementation. Each part of the project will discuss in the following section.

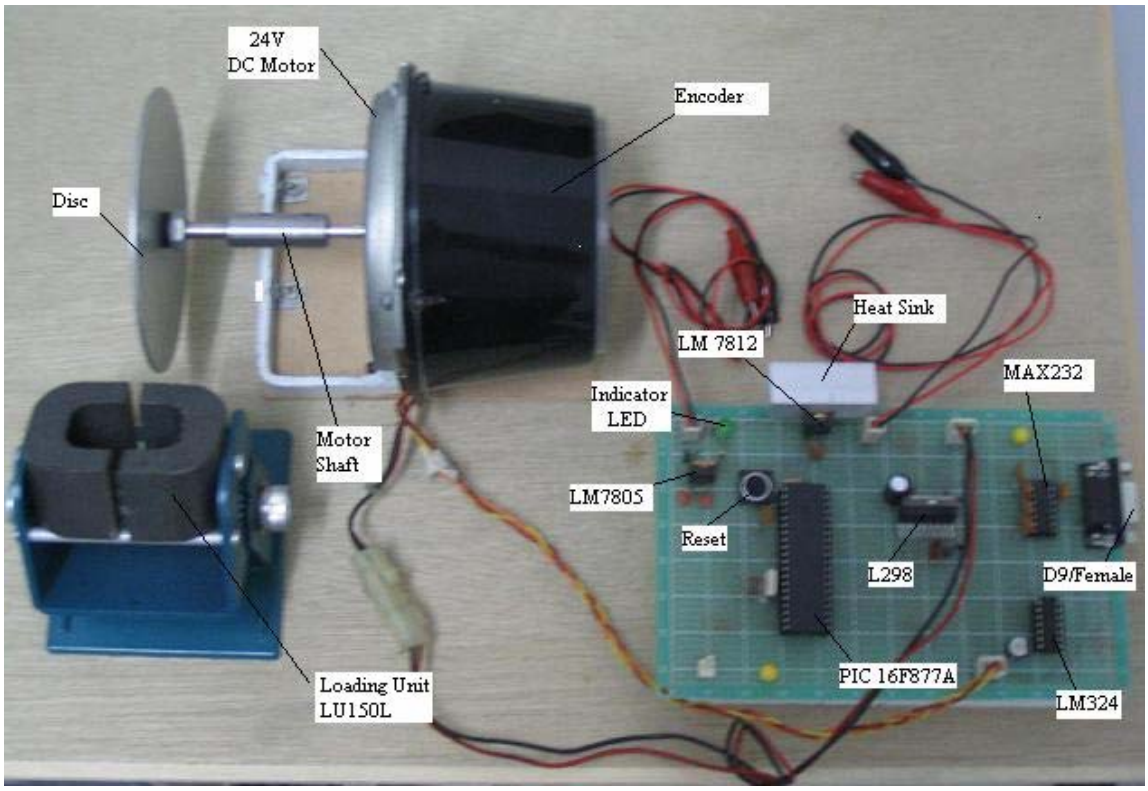


Figure 3.2 Picture of the project

3.2 Hardware Implementation

This section will discuss about components that had been used included DC motor, optical encoder, power supply 5V, microcontroller PIC 16F877A, pulse-width-modulation (PWM) and RS232 serial communication of microcontroller, DC Motor drive and loading unit LU150L

3.2.1 DC Motor

Figure 3.3 shows the DC motor that will be used in this project. It is a permanent magnet 24V DC motor. The specification of this motor is shown in Table 3.1.

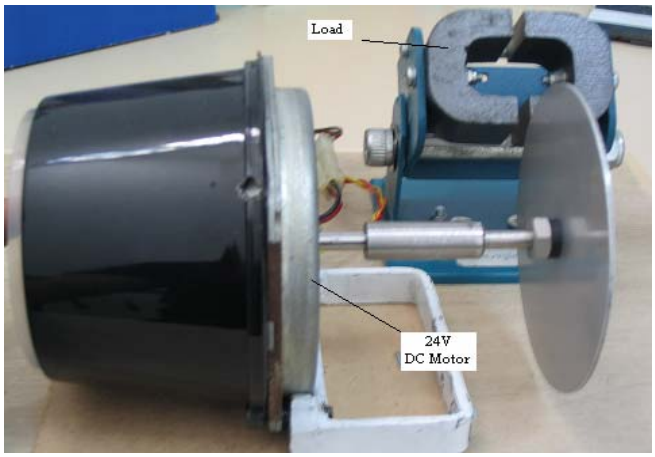


Figure 3.3 24V DC motor

Table 3.1 Specification of the motor

Manufacturer	Tohoku Ricoh CO. Ltd
Voltage rated	24V
Power	55W
Type	7K100012

3.2.2 Optical Encoder

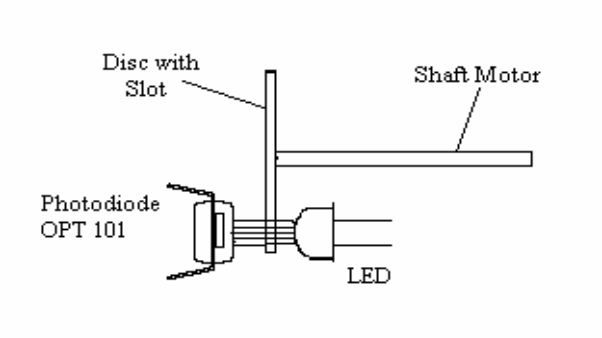
In this project, an optical encoder will be used to measure the DC motor speed. The fundamental reason for the superiority of this system is that the optical encoder used as the velocity sensor, is capable of much better performance than the generator type of tachometer (by using back EMF that discussed in section 2.2.1). When the optical disc is properly mounted on the motor shaft, it generates a frequency directly proportional to motor speed. Changes in gap, temperature, and

magnet strength simply have no effect on the output of the optical tachometer. By contrast, an analog tachometer is directly affected by all the problems listed above.

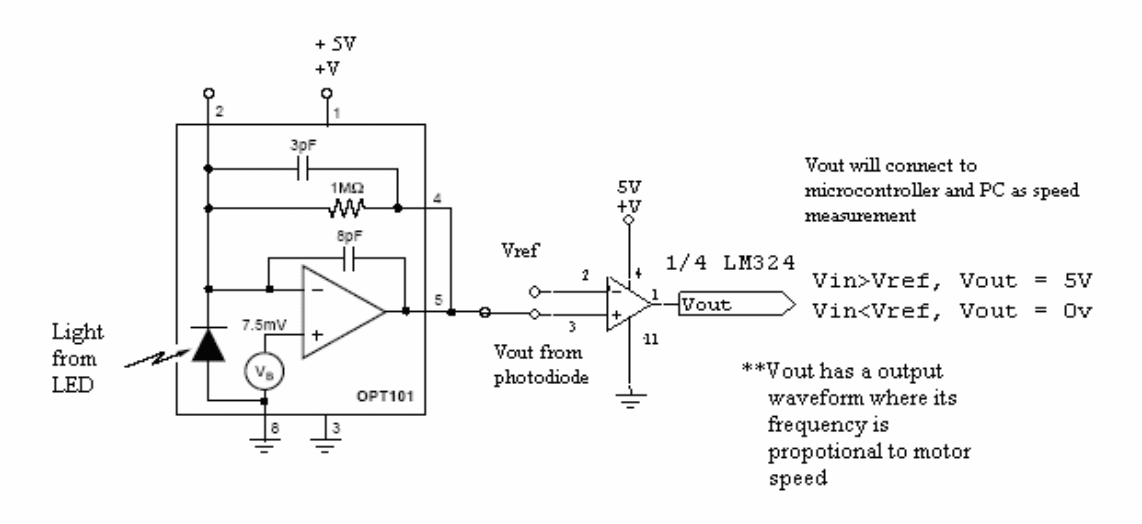
The encoder in the market is very expensive. In order to reduce the cost of the project, an optical encoder is built. Figure 3.4a and Figure 3.4b show the basic configuration and the schematic of the optical encoder in action respectively [7].

The speed accuracy over fractions of a revolution depends on the quality of the optical encoder. The DC motor has an optical disc (made by cardboard) mounted on its shaft. The disc has N radial lines on its surface. In this project, we will make four slots on the disc (N=4). This will give a resolution of 1/4 in one rotation. An LED (light emitting diode) as transmitter is put at one side of the disc and a photodiode, as receiver is fixed on the other side of the disc. Chip OPT 101 is selected as photodiode in this project.

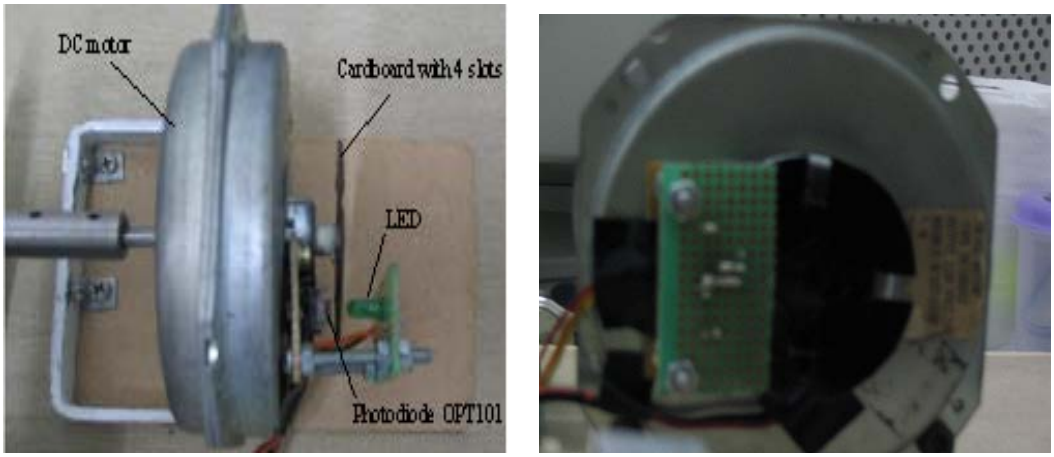
To make sure the output waveform in digital signal (High/Low) which is readable by microcontroller, Chip LM324 is used as a comparator. When the V_{out} of photodiode is less than V_{ref}, the output of LM324 will be 0V (Low) and when the V_{out} is greater than V_{ref}, the output of LM 324 will be 5V (High). The output signal form LM324 has a frequency which is given by Equation 2.3. Then, the output signal will be sent to microcontroller as representation of actual speed.



(a)



(b)



(c)

Figure 3.4 Optical encoder (a) Basic configuration (b)Schematic circuit (c)Picture

3.2.3 Power Supply +5V

Most digital logic circuits and processors need a +5 volt power supply. To use these parts we need to build a regulated +5 volt source. Usually we start with an unregulated power supply ranging from 9 volts to 24 volts DC.

To make a +5 volt power supply, we use a LM7805 voltage regulator IC (Integrated Circuit). The IC is shown below.

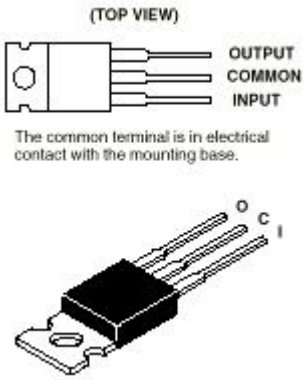


Figure 3.5 IC LM7805

Sometimes the input supply line may be noisy. To help smooth out this noise and get a better 5 volt output, capacitors is usually added to the circuit.

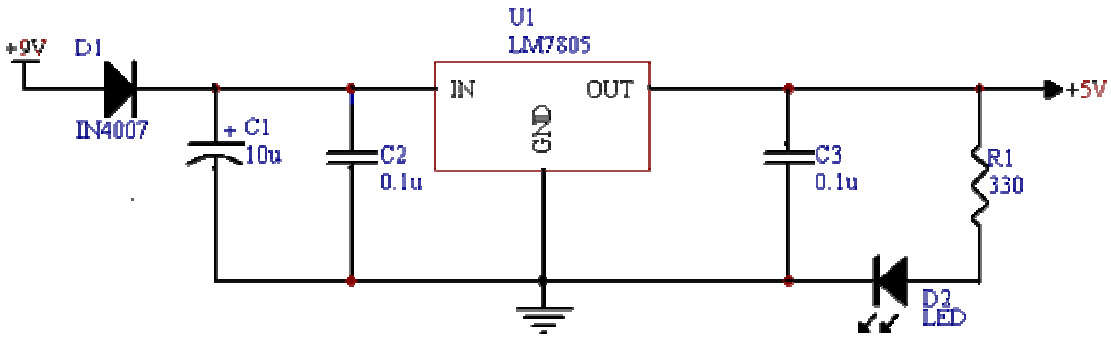


Figure 3.6 Schematic circuit of +5V power supply

3.2.4 Microcontroller PIC 16F877A

The microcontroller acts like the brain of the DC motor speed control system. The microcontroller chip that has been selected for the purpose of controlling the speed of DC motor is PIC16F877A manufactured by Microchip. This chip is selected based on several reasons [8]:

i. Its size is small and equipped with sufficient output ports without having to use a decoder or multiplexer.
ii. Its portability and low current consumption.
iii. It has PWM inside the chip itself which allow us to vary the duty cycle of DC motor drive.
iv. It is a very simple but powerful microcontroller. Users would only need to learn 35 single word instructions in order to program the chip.
v. It can be programmed and reprogrammed easily (up to 10,000,000 cycles) using the universal programmer in robotics lab.

Refer to Table 3.2 for the pin connection of PIC16F877A in DC Motor speed control system. Pins not stated in the table are not used and left hanging. Figure 3.7 shows the schematic circuit of microcontroller PIC16F877A. At the beginning, microcontroller will receive desired speed from PC through serial port. The detected motor speed from optical encoder will feedback to microcontroller through RA0 of PIC16F877A. The microcontroller will operate as it programmed (detail program at section 3.3.1) to produce a new duty cycle (from CCP2) that proportional to the error speed. Thus, average of voltage supply from DC motor drive can be varied in order to maintain the speed at desired value.

Table 3.2 Pin connection of PIC16F877A for DC motor speed control system

Pin Name	Pin No.	Description	Application
V_{DD}	11,32	Positive Supply (+5V)	Power Supply to chip
V_{SS}	12,31	Ground Reference	Ground Reference
OSC1	13	For oscillator or resonator	Connected to resonator 20MHz with 22pF
OSC2	14		
MCLR\	1	Reset Input	Always connected to +5V
RA0	2	Input/Output pin	Input of V_{out} from LM324 as speed counter
RB1	34	Input/Output pin	Output to control CW/CCW of left motor
RB2	35		
CCP2	16	Capture/Compare/PWM	Output of duty cycle(PWM) to control motor speed

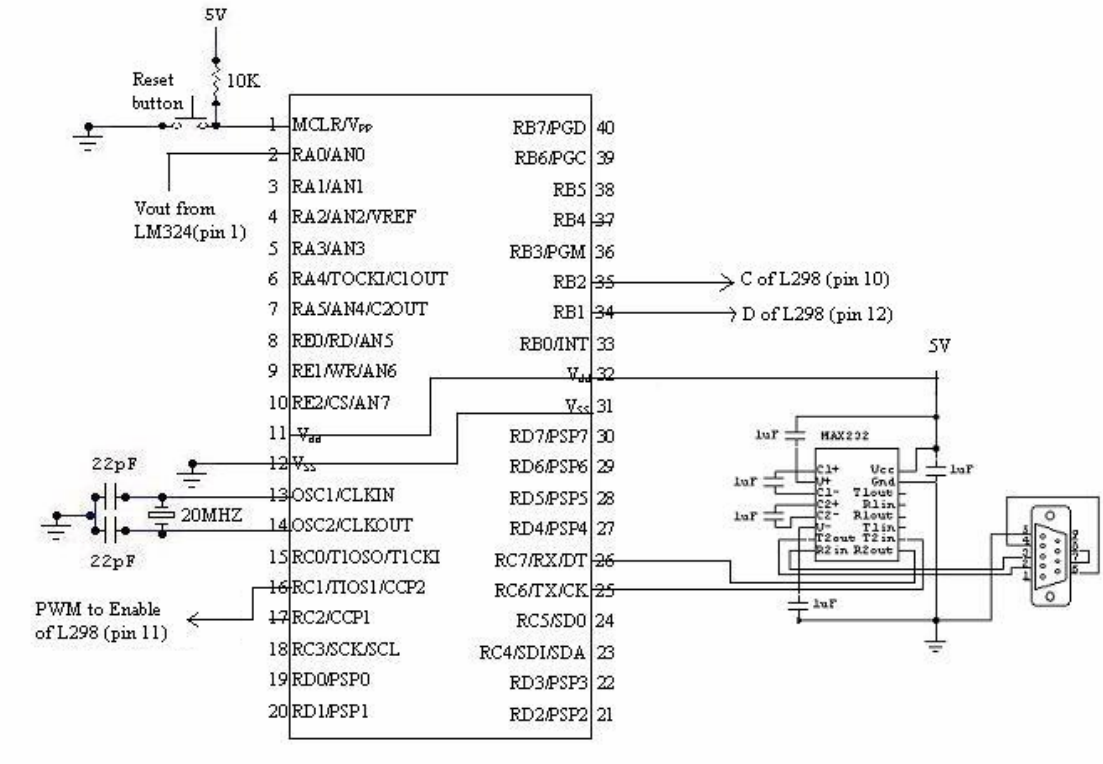


Figure 3.7 Schematic circuit of PIC16F877A

3.2.4.1 Pulse-Width-Modulation (PWM) in Microcontroller

The Pulse-Width-Modulation (PWM) in microcontroller is used to control duty cycle of DC motor drive.

PWM is an entirely different approach to controlling the speed of a DC motor. Power is supplied to the motor in square wave of constant voltage but varying pulse-width or duty cycle. Duty cycle refers to the percentage of one cycle during which duty cycle of a continuous train of pulses. Since the frequency is held constant while the on-off time is varied, the duty cycle of PWM is determined by the pulse width. Thus the power increases duty cycle in PWM.

The expression of duty cycle is determined by,

$$\%Dutycylcle = \frac{t_{on}}{T} \times 100\% \qquad (3.1)$$

Basically, the speed of a DC motor is a function of the input power and drive characteristics. While the area under an input pulse width train is measure of the average power available from such an input.

3.2.4.2 RS232 Serial Communication

SCI is an abbreviation for Serial Communication Interface and, as a special subsystem of microcontroller PIC16F877A. It provides RS232 serial communication with PC easily.

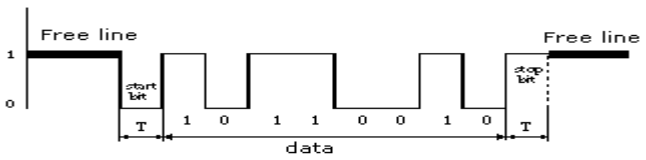


Figure 3.8 NRZ (Non Return to Zero) format data

As with hardware communication, we use standard NRZ (Non Return to Zero) format also known as 8 (9)-N-1, or 8 or 9 data bits, without parity bit and with one stop bit. Free line is defined as the status of logic one. Start of transmission - Start Bit, has the status of logic zero. The data bits follow the start bit (the first bit is the low significant bit), and after the bits we place the Stop Bit of logic one. The duration of the stop bit 'T' depends on the transmission rate and is adjusted according to the needs of the transmission. For the transmission speed of 9600 baud, T is 104μs.

In order to connect a microcontroller to a serial port on a computer, we need to adjust the level of the signals so communicating can take place. The signal level on a PC is -10V for logic zero, and +10V for logic one. Since the signal level on the microcontroller is +5V for logic one and 0V for logic zero, we need an intermediary stage that will convert the levels. One chip specially designed for this task is MAX232. This chip receives signals from -10 to +10V and converts them into 0 and 5V. The circuit for this interface is shown in the Figure 3.9

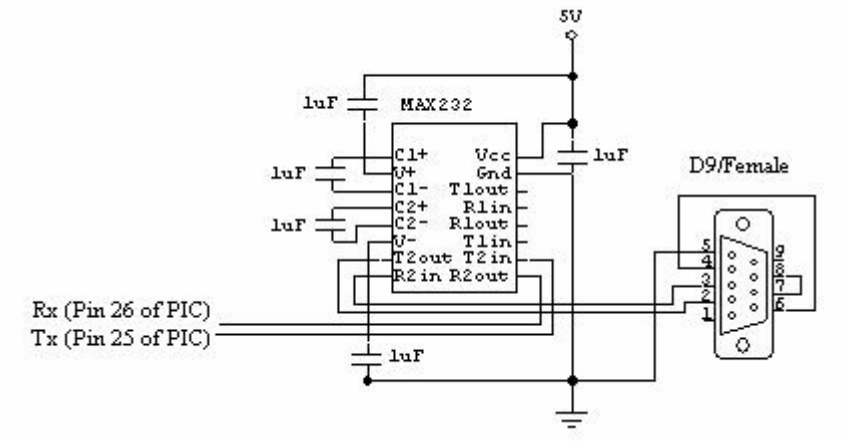


Figure 3.9 Connection between D9 Female serial port, MAX232 and PIC16F877A

3.2.5 DC Motor Drive

If a DC motor is connected directly to the voltage supply, the constant power will be supplied to the DC motor all the time. Due to the constant power to motor, the speed of motor will slow down when the load is heavier and speed up when the load is lighter. So, DC motor drive is needed where we can control the magnitude of supply voltage in order to control the speed of DC motor.

The DC motor drive that will be used in this project is a dual full bridge driver, chip L298. The operating supply voltage of chip L298 is up to 46V and the total DC current up to 4A. Table 3.3 shows function of each pin of chip L298.

The time to enable the chip L298 will be determined by the duty cycle pulse that sent from PWM in microcontroller. The average of voltage that supply to DC motor is given by Equation 2.17.

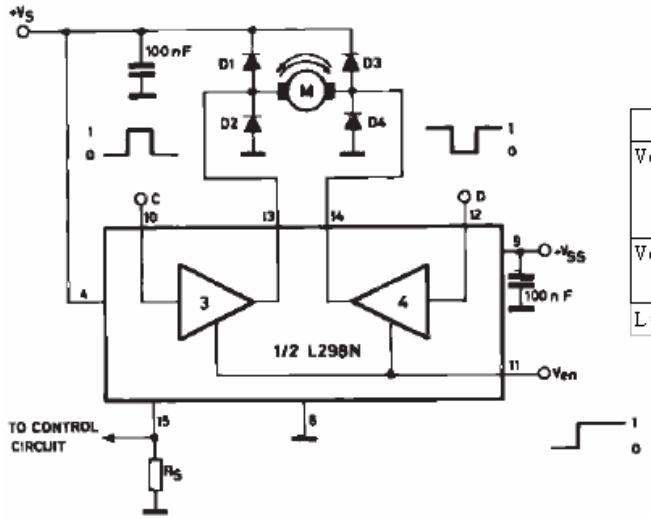
Table 3.3 Pin function of chip L298

Pin	Name	Function
1;15	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	Vs	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
6;11	Enable A; EnableB	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	GND	Ground.
9	Vss	Supply Voltage for the Logic Blocks. A100nF capacitor must be connected between this pin and ground.
10;12	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
13;14	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.

Figure 3.10 shows the schematic circuit of L298 with a 24V DC motor. To supply the voltage to motor through L298, it should be enabled by +5V at pin 6 (Enable A) or pin 11 (Enable B). If the chip is disabled, V_{ave} is zero, there is no

voltage supply to motor, the motor will stop running. In this project, the duty cycle of PWM will sent to V_{en} to drive the motor.

From table, we know that the motor will continuous running (as long as the motor is not overloaded) when the C and D of L298 have different input. When input of C (pin 10) is High (H), D (pin12) is Low (L), the motor will run in forward direction and vice versa. When both C and D have same input (H/L), the motor will stop running.



Inputs		Function
Ven = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
Ven = L	C = X ; D = X	Free Running
		Motor Stop
L = Low	H= High	X = Don't care

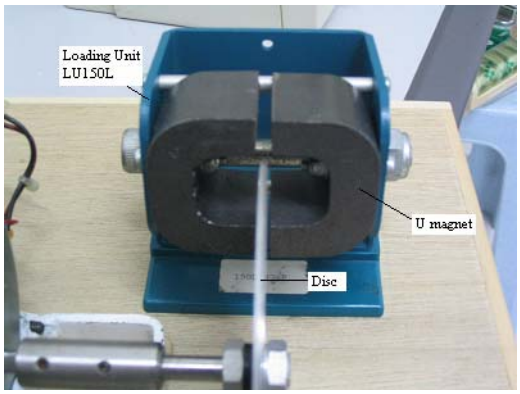
Figure 3.10 Bi-direction of DC motor speed control

So, to make sure the motor is running all the time, the different input have to sent to C and D. In this project, the motor is running all the time in forward direction by receiving C=H (RB1) and D=L (RB2) from microcontroller during the initialization of the program. (detail in part 3.3.1.1(a)).

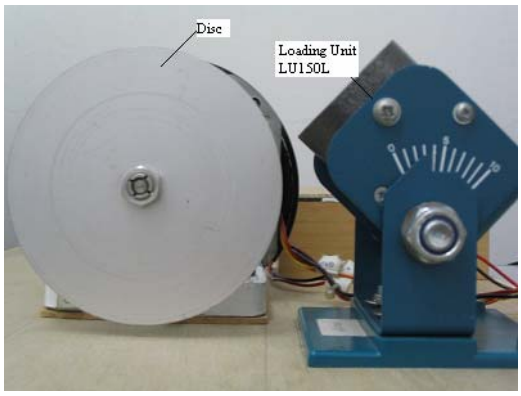
3.2.6 Loading Unit LU150L

For DC motor speed control with application of varies loads, a loading unit LU150L is used. It has two U shape magnets with a small space between it. Without LU150L, the disc that fixed at the motor shaft can run smoothly at the constant speed.

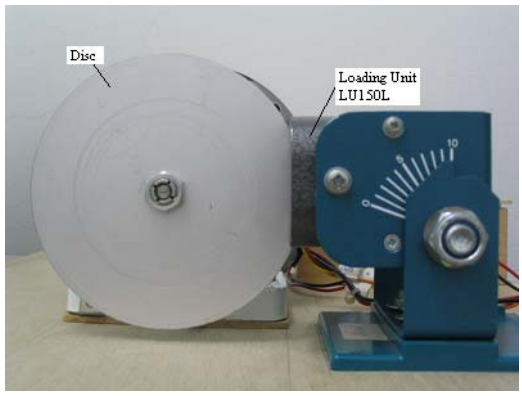
With the presence of LU150L when the motor is running, the disc will cut the magnetic flux field and it becomes a load to the motor. Thus, motor will slow down. The position of the magnets have been defined as no load, normal load and overload condition as shown in Figure 3.11. In this project, a controller has been designed to maintain speed back to the desired speed by using microcontroller.



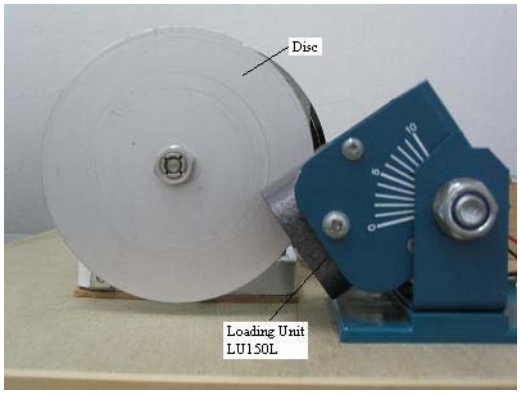
(a)



(b)



(c)



(d)

Figure 3.11 Application of vary load by using Loading Unit LU150L
(a) Loading Unit LU150L (b) No load condition (c) Normal load condition
(d) Overload condition

3.3 Software Implementation

For software implementation, MPLAB IDE is used to program microcontroller in assembly language. Besides, Visual Basic 6.0 is used for user interface purpose and for monitoring the speed response of the system.

3.3.1 Algorithm and Programming in MPLAB IDE

Microcontroller acts as brain of the whole DC motor speed control system. It will receive the desired speed from user through PC that interface with RS232 serial port. The actual speed will be compared with the desired speed and the correction will be done by microcontroller to always maintain the DC motor speed at the desired speed.

An algorithm has to be developed to make the microcontroller to read the input and respond accordingly. Therefore, the algorithm is established and represented by a flowchart in Figure 3.12 and Figure 3.13. These flowcharts are then translated into assembly language and compiled using MPLAB, the PIC16F877A software development tool. The program in assembly language can be referred in Appendix A.

There are two parts of the program which are main program and interrupt program. The microcontroller will always run the main program until there is an interrupt occurred. When microcontroller receives an interrupt flag, then it will jump to interrupt process.

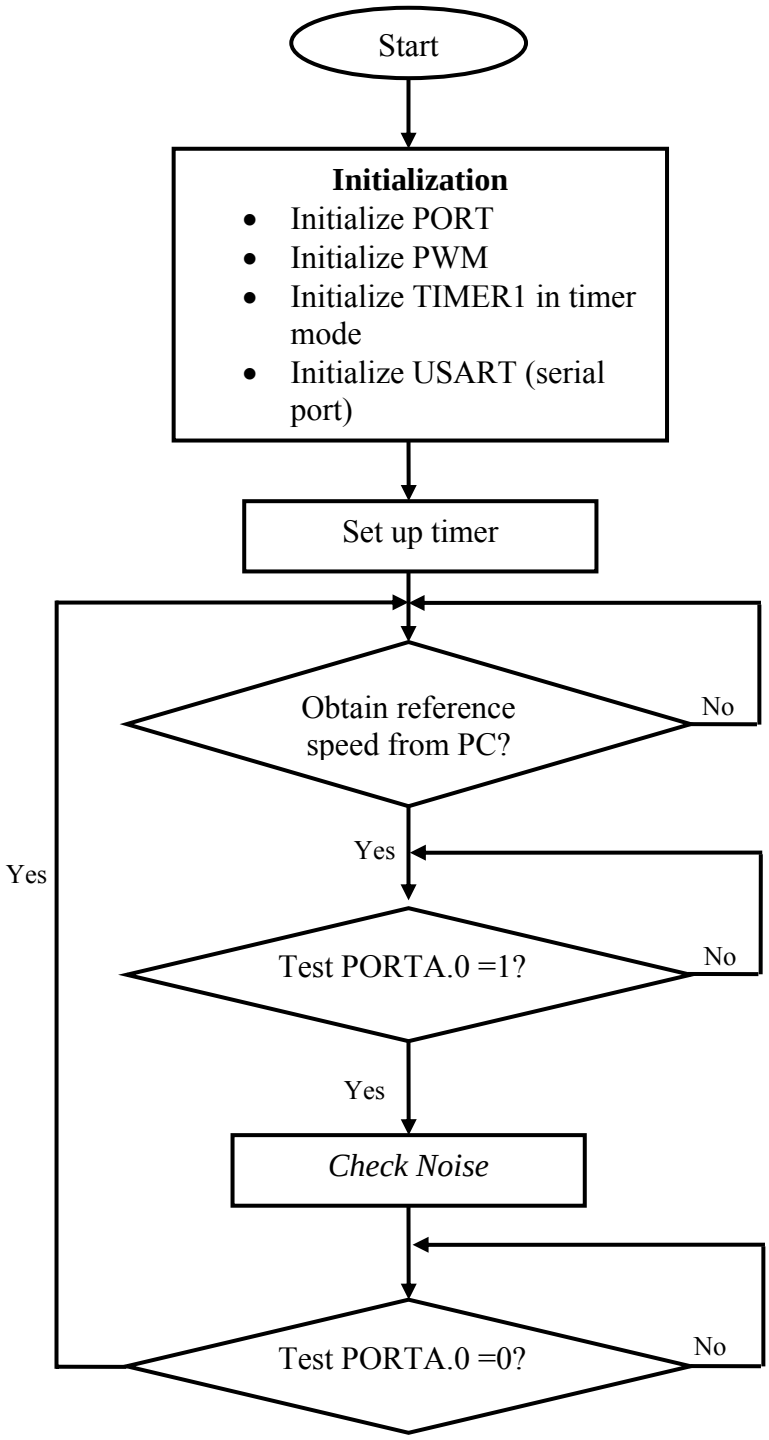


Figure 3.12 Flow chart of microcontroller's main program

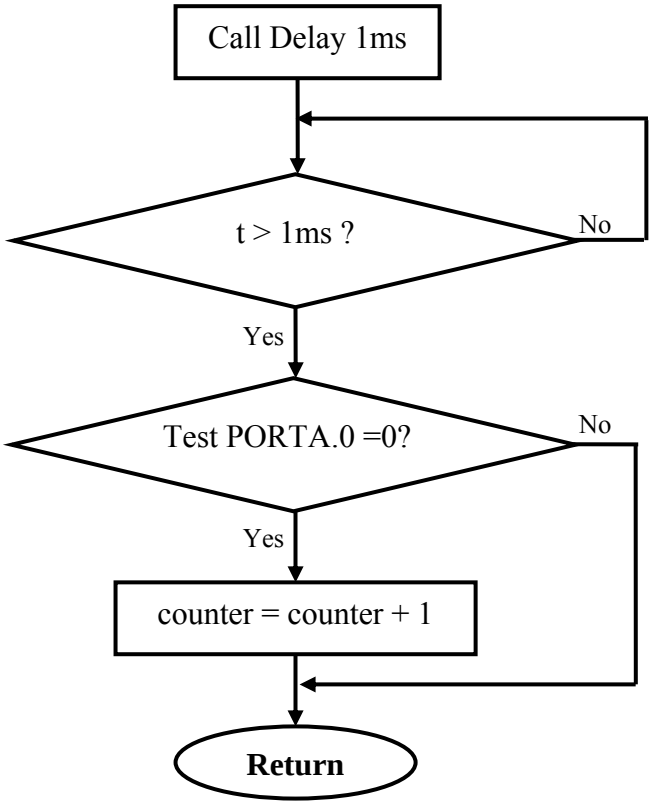


Figure 3.13 Flow chart of check noise function

3.3.1.1 Processing Explanation of Main Program

There are six main parts of main program in microcontroller. There are initialization of ports, PWM, Timer1, setup for serial port, get reference speed and check noise function.

a) Initialization of the mode of ports A ,B

In this project, we use pin 0^{th} of the Port A (RA0) as digital input where it receives input (H/L) from LM324 (pin1). A user register, X1 has been defined as speed counter for pulses receive from RA0. When the input of RA0 is High (H), it will increase speed counter X1 where X1 = X1 + 1 (if it is not noise after *check noise*, detail in section 3.3.1.1 (f)). Otherwise, the counter will remain its value.

All pins of Port B are set to output. RB1 is always set to High (H) and RB2 set to Low (L) to make the motor run in forward direction as describe in section 3.2.5.

b) Initialization of PWM

A PWM output (Figure 3.14) has a time-base (period) and a time that the output stays high (duty cycle). The frequency of the PWM is the inverse of the period (1/period).

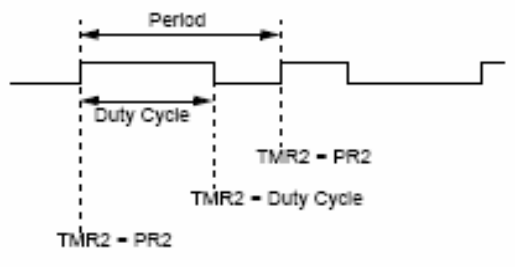


Figure 3.14 PWM output

Figure 3.15 shows a simplified block diagram of the CCP module in PWM mode.

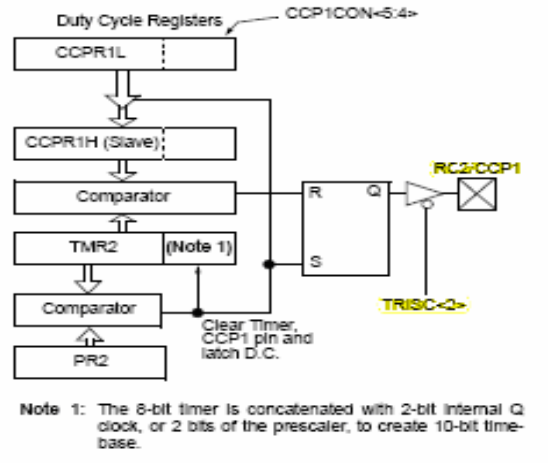


Figure 3.15 Simplified PWM block diagram

To setup for PWM operation, the following steps should be taken when configuring the CCP module for PWM operation:

i. Set the PWM period by writing to the PR2 register.

ii. Set the PWM duty cycle by writing to the CCPR2L register and CCP2CON<5:4> bits.
iii. Make the CCP1 pin an output by clearing the TRISC<2> bit.
iv. Set the TMR2 prescale value and enable Timer2 by writing to T2CON.
v. Configure the CCP2 module for PWM operation.

The PWM period is specified by writing to the PR2 register. The PWM period can be calculated using the following formula:

$$\text{PWM period} = [(PR2) + 1] * 4 * T_{OSC} *(\text{TMR2 prescale value}) \quad (3.2)$$

In this project, PR2 is set to 255(maximum value of 8 bit for maximum range), so the PWM period is then become

PWM period = (255+1)* 4 * (1/20M) * (1)
= 51.2μs
= 19.53kHz

where Fosc = 20MHz and TMR2 prescale =1

The PWM duty cycle is specified by writing to the CCPR2L register and to the CCP2CON<5:4> bits. Up to 10-bit resolution is available. The CCPR2L contains eight MSbs and the CCP2CON<5:4> contains two LSbs. This 10-bit value is represented by CCPR2L:CCP2CON<5:4>. The following equation is used to calculate the PWM duty cycle in time:

$$\text{PWM duty cycle} = (\text{CCPR2L:CCP2CON<5:4>}) * T_{OSC} * (\text{TMR2 prescale value}) \quad (3.3)$$

CCPR2L and CCP2CON<5:4> can be written to at any time, but the duty cycle value is not latched into CCPR2H until after a match between PR2 and TMR2 occurs (i.e., the period is complete).

c) Initialization of TIMER1 in Timer Mode

The Timer1 module in PIC16F877A is a 16-bit timer/counter consisting of two 8-bit registers (TMR1H and TMR1L), which are readable and writable. Figure 3.16 shows the Timer1 block diagram which operate in timer mode. The TMR1 Register pair (TMR1H:TMR1L) increments from 0000h to FFFFh (decimal from 0 to 65535) and rolls over to 0000h. The TMR1 Interrupt, if enabled, is generated on overflow, which is latched in interrupt flag bit, TMR1IF (PIR1<0>). This interrupt can be enabled/disabled by setting/clearing TMR1 interrupt enable bit, TMR1IE (PIE1<0>). Timer mode is selected by clearing the TMR1CS (T1CON<1>) bit. In this mode, the input clock to the timer is $F_{OSC}/4$.

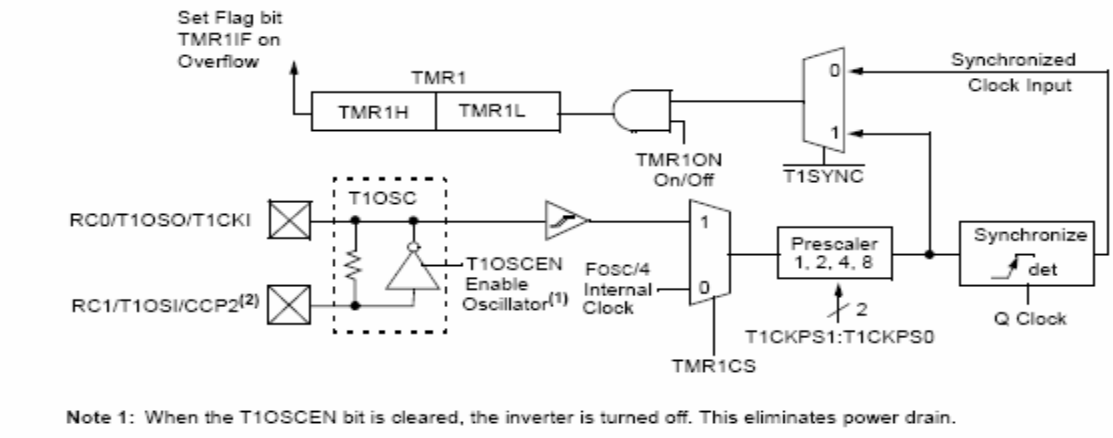


Figure 3.16 Timer1 block diagram

When Timer1 increases from 0000h to FFFFh, it takes 65535 cycle and the time consumed is given by,

$$t = (65535 - 0) * (1/(20MHz/4)) \quad ; F_{osc} = 20MHz$$
$$= 13.107 \text{ ms.}$$

In this project, the interrupt is required to occur each 0.39321s where the pulses counted in X1 will be converted to speed and correction taken. A user register, X2 is defined as gain for this purpose.

$$\text{Gain, } X2 = 0.39321s / 13.107ms$$
$$= 30$$

Besides, the interruption enable bit of CCP2 is set. Also GIE and PIE are set.

d) Setup for serial port

The Universal Synchronous Asynchronous Receiver Transmitter (USART) module is one of the two serial I/O modules. (USART is also known as a Serial Communications Interface or SCI.)

The USART is configuring in asynchronous mode. In this mode, the USART uses standard non-return-to zero (NRZ) format (one START bit, eight or nine data bits, and one STOP bit). The most common data format is 8-bits and it is used in this project. An on-chip, dedicated, 8-bit baud rate generator can be used to derive standard baud rate frequencies from the oscillator. The USART transmits and receives the LSb first. The transmitter and receiver are functionally independent, but use the same data format and baud rate.

$$\text{Baud Rate} = F_{soc}/(16(\text{SPBRG}+1)) \qquad (3.4)$$

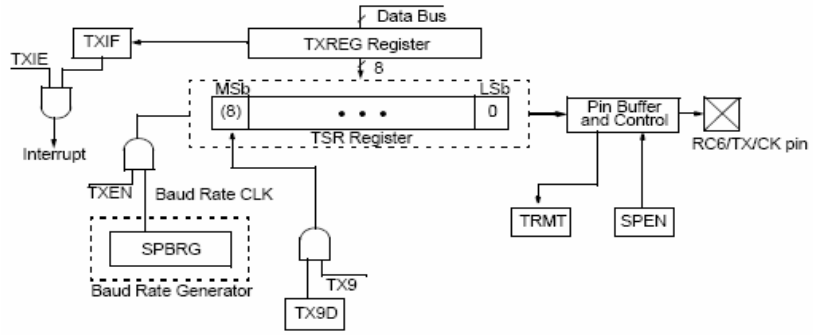


Figure 3.17 USART transmit block diagram

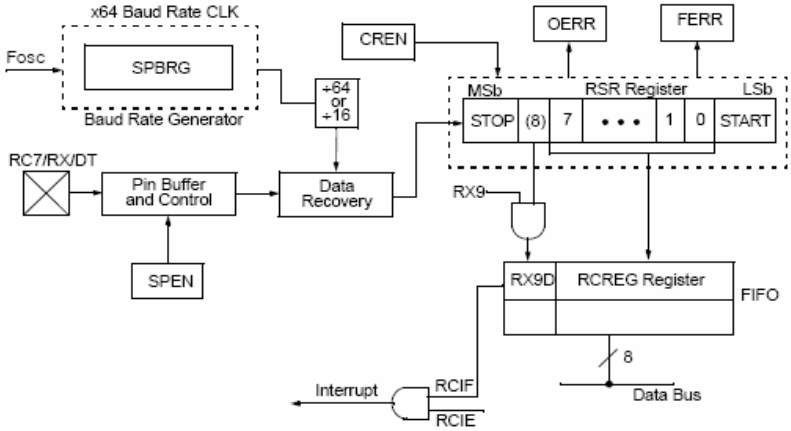


Figure 3.18 USART receive block diagram

To set baud rate at 9600bps, we have to set SPBRG as

SPBRG = (F_{soc}/ Baud Rate / 16)-1

= 20M / 9600 / 16 -1

= 129

Bit SPEN (RCSTA<7>) is set as 1 and bits TRISC<7:6> are set as 0 in order to configure pins RC6/TX/CK and RC7/RX/DT as the Universal Asynchronous Receiver Transmitter.

e) Get reference speed

At beginning of the program, microcontroller will get the reference speed from user at PC through RS232 serial communication. It will wait there until it gets the reference speed from user.

f) *Check Noise* function

From experimental result, if is found that there are some noise besides real pulses from LM324, especially when motor running at high speed. This noise can cause over reading pulses at speed counter, X1 if it is not filter out. From experiment, it is found that the noise have a bandwidth less than 1ms (bandwidth of real pulses always greater than 1ms). So, when a High (H) pulses is receive at RA0, it will hold for 1ms. If the pulse detected still High (H), it means that the received is real pulse and the X1 will increase it value with one. But if the pulse detected is Low (L), then it is noise and ignored where X1 remains its value.

3.3.1.2 Processing Explanation of Interrupt Process

Interrupt process is occurs each 0.39321s. The microcontroller will execute the program in interrupt process instead of main program that is running when it

receive the interrupt flag. Each part of interrupt process is discussed in the following section.

a) Clearing of interruption flag

The interruption is occurs every 0.39321s with CCP2. The interruption flag of CCP2 should be cleared first. When not clearing this, the following interruption occurs without waiting desired time.

b) Speed counter

At each interruption (each 0.39321s), the value of speed counter, X1 will be taken as detected pulses. From optical encoder with 4 slots that built in this project, we know that it will produces 4 pulses when motor turn for 1 round. Data from speed counter, X1 in microcontroller will be loaded to PC each 0.39321s. So, the motor speed is given by,

$$\text{detected speed} = \left(\frac{\text{detected pulses}}{N}\right) * \left(\frac{1}{t}\right) \quad \textit{round/s} \tag{3.5}$$

where N = number of slots

t = time of getting data in second

$$\text{detected speed} = \left(\frac{\text{detected pulses}}{4}\right) * \left(\frac{1}{0.39321}\right) \quad \textit{round/s}$$

$$= \left(\frac{\text{detected pulses}}{4}\right) * \left(\frac{1}{0.39321}\right) * 60 \ \textit{round/minit (RPM)} \tag{3.6}$$

c) Error speed

The detected speed will be compared to reference speed to calculate the error between them.

$$\text{Error pulses} = \text{Reference pulses} - \text{Detected pulses} \quad (3.7)$$

and

$$\text{Error speed} = \left(\frac{\text{Error pulses}}{4}\right) * \left(\frac{1}{0.39321}\right) * 60 \text{ round/min} \quad (3.8)$$

The error speed will then converted into error voltage by using Equation 4.2 from experiment with optical encoder that has been built.

$$\text{Error voltage} = \frac{\text{Error speed} + 80.89}{207.16} \quad (3.9)$$

Finally, the error voltage will be converted to duty cycle based on Equation 2.17 to determine to speed up or slow down the motor. If there is no error, the duty cycle of PWM is remained.

$$\text{Error DC} = \frac{\text{Error voltage}}{V_{in}} \quad \text{where } V_{in} = 12\text{V} \quad (3.10)$$

d) Speed down process

When the detected speed of the motor is higher than the reference speed, a duty ratio is decreased and a motor drive electric current is suppressed. The rate that the duty ratio becomes smaller is decided by error speed.

e) Speed up process

When the detected speed of the motor is lower than the reference speed, a duty ratio is increased and a motor drive electric current is increased. The rate that the duty ratio becomes bigger is decided by error speed.

f) Send detected speed to PC

To monitor the performance of the system, detected speed will sent to PC each 0.39321s as detected pulses. This value will then convert to represent the detected speed (rpm) in PC by using Visual Basic 6.0 program. (detail in section 3.2.3.1)

g) Interruption ending process

The RETFIE instruction is executed at end of the interruption processing. Before that, the counter and timer have to clear and reset.

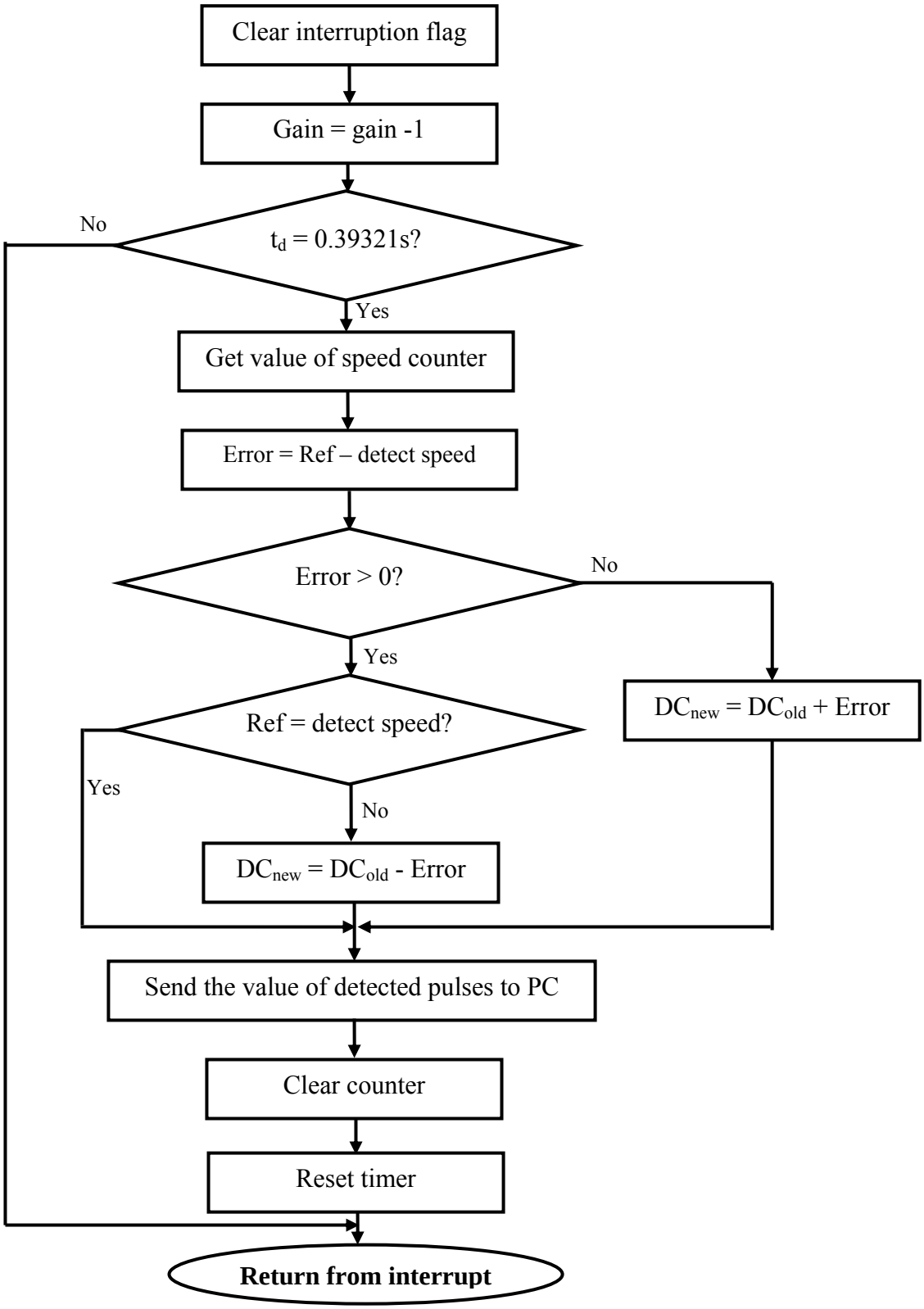


Figure 3.19 Flow chart of microcontroller's interrupt process

3.3.2 Programming in Visual Basic 6.0

In this project, a program will be developing using Visual Basic 6.0. This program is able to send data (desired speed from user) to microcontroller and plot a graph of detected speed versus time to monitor the performance of the system.

There are two forms in Visual Basic 6.0. They are introduction form and main program form which are shown in Figure 3.20 and Figure3.21 respectively.

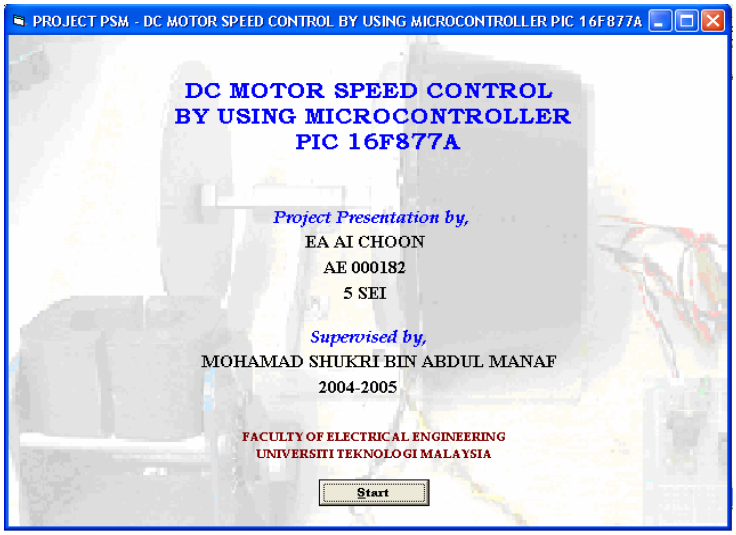


Figure 3.20 Introduction form

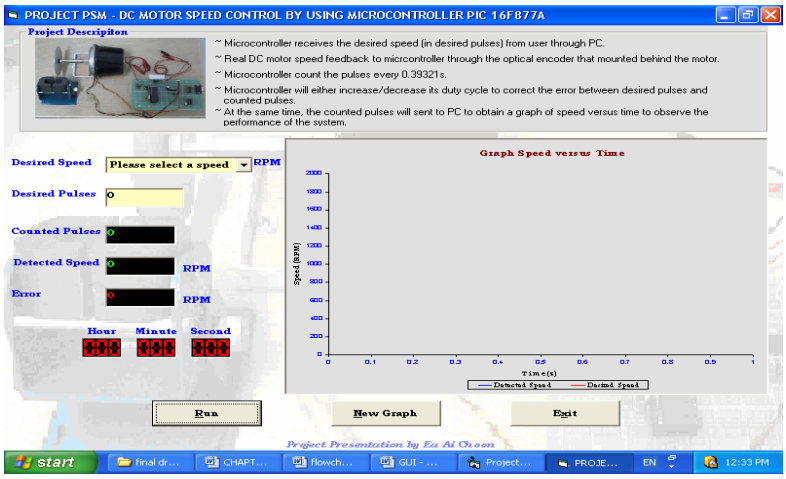


Figure 3.21 Main program form

The source code of this project is shown in Appendix B and the flow chart of the program is shown in Figure 3.22. Two main tasks of the program in Visual Basic which are reading input from user and microcontroller and plot a graph of speed versus time.

3.3.2.1 Reading Input from User and Microcontroller

To obtain desired speed from user, the user is required to select a desired speed at computer. The desired speed (rpm) will then convert into desired pulses before send to microcontroller through serial port. From Equation 3.7, desired pulses is given by,

$$\text{desired pulses} = \frac{\text{desired speed} * 4 * 0.39321}{60} \tag{3.11}$$

3.3.2.2 Plot Graph Speed versus Time

To monitor performance of the system, a graph of detected speed versus time will be obtained by using Visual Basic. The user will sent the desired speed to microcontroller by click on Run command button. It is also allowed to get a new graph by click on the New Graph command button.

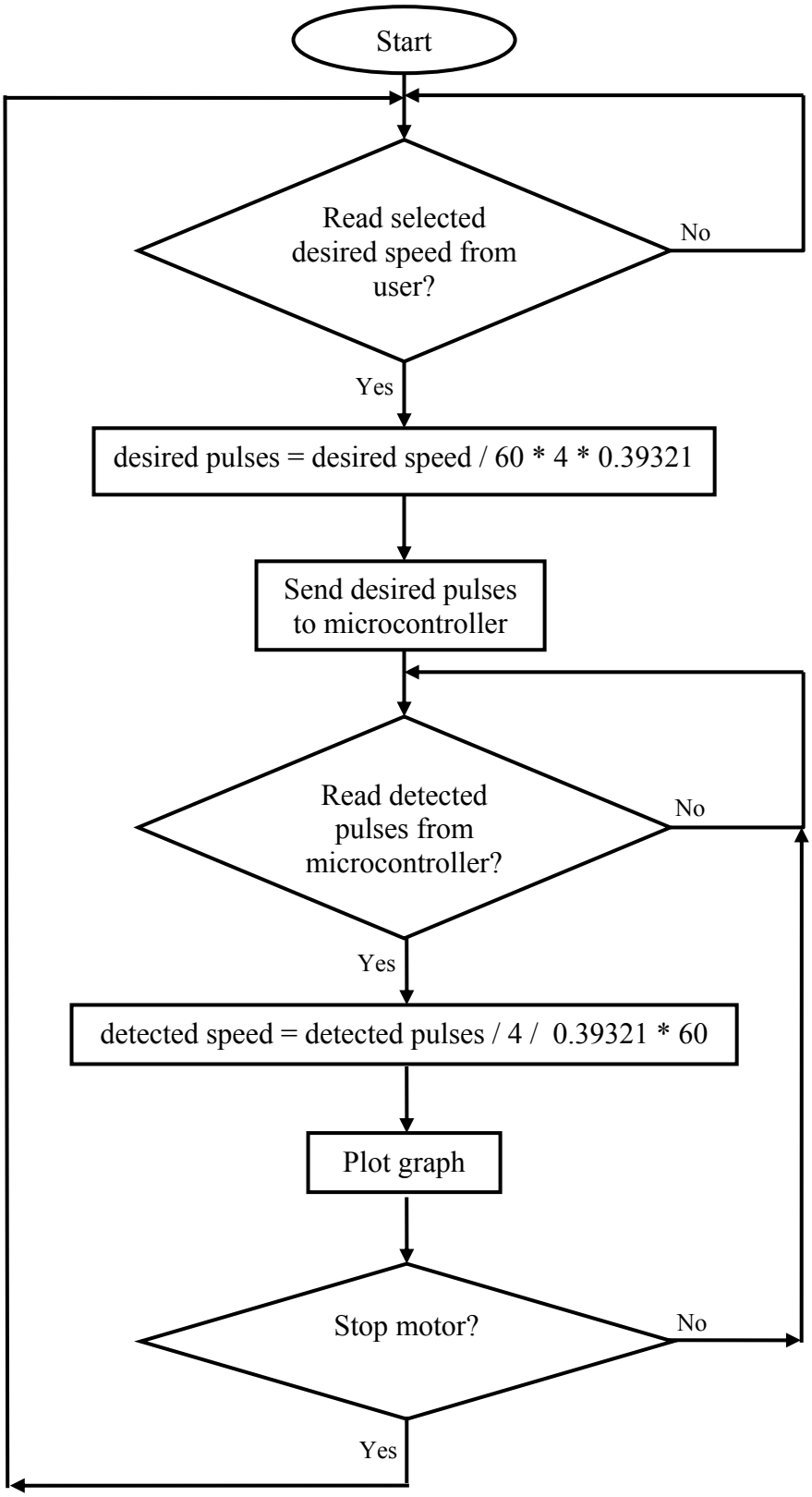


Figure 3.22 Flow chart of program in Visual Basic 6.0

CHAPTER 4

RESULT AND DISCUSSION

4.1 Introduction

Some experiments had been conducted for the project. First and foremost, an experiment is conducted to find out the relationship between voltage supply and motor speed. Then, data collection is done at each speed for DC motor speed control system to observe performance of the system. Last but not least, an analysis on the dead time of the system is made.

4.2 Experiment: Determine Relationship of Voltage Supply and Motor Speed

An experiment is conducted to determine the relationship between voltage supply and speed [14]. The procedures and the result will be discussed in following sections.

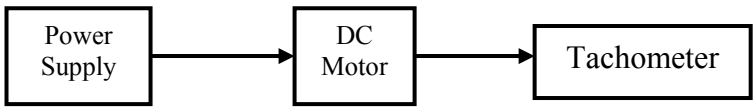


Figure 4.1 Experiment with tachometer

4.2.1 Procedures

1) The circuit was connected as Figure 4.1.
2) Voltage of 0.5V was supplied to motor.
3) Value of rpm at tachometer was recorded in Table 4.1.
4) The voltage increased in steps of 0.5V until 16V and step 3 was repeated.

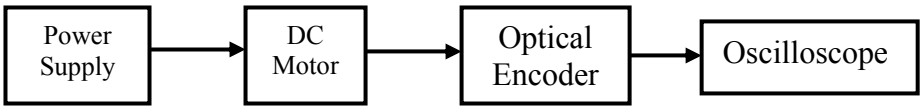


Figure 4.2 Experiment with optical encoder

5) Circuit as Figure 4.2 was completed by changing tachometer with the optical encoder that has been built. The output waveform from optical encoder was connected to oscilloscope.
6) Step 2 was repeated.
7) The readings for the frequency of oscilloscope were recorded.

$\left(\text{rpm} = \frac{f_{out} \times 60}{N} \right)$, N=4.

8) Step 4 was repeated.
9) A graph of rpm-voltage was plotted for both tachometer and optical encoder.
10) Based on graphs, both result was compared and the accuracy of optical encoder that has been built is found.

4.2.2 Experimental Result Analysis

From experiment, the data was recorded in Table 4.1. A graph of speed versus voltage supply by using tachometer and optical encoder is shown in Figure 4.3.

Table 4.1 Relationship of voltage supply and motor speed

Voltage Supply (V)	Speed (rpm_1) **Tachometer**	Speed (rpm_2) **Optical Encoder**	$\%\ \text{Error} = \frac{rpm_1 - rpm_2}{rpm_1} \times 100\%$
0.0	0	0.00	0.00
0.5	0	0.00	0.00
1.0	100	99.60	0.40
1.5	240	230.06	4.14
2.0	330	315.79	4.31
2.5	410	420.21	2.49
3.0	550	553.51	0.64
3.5	640	655.02	2.35
4.0	770	767.26	0.36
4.5	860	854.70	0.62
5.0	970	940.44	3.05
5.5	1090	1048.95	3.77
6.0	1200	1164.60	2.95
6.5	1280	1258.39	1.69
7.0	1400	1358.70	2.95
7.5	1490	1467.71	1.50
8.0	1590	1552.80	2.34
8.5	1700	1648.35	3.04
9.0	1820	1794.26	1.41
9.5	1920	1870.32	2.59
10.0	2030	1968.50	3.03
10.5	2140	2074.69	3.05
11.0	2230	2228.83	0.05
11.5	2340	2272.73	2.87
12.0	2480	2400.00	3.23
12.5	2570	2516.78	2.07
13.0	2690	2617.80	2.68
13.5	2810	2702.70	3.82
14.0	2890	2840.91	1.70
14.5	2990	2958.58	1.05
15.0	3130	3048.78	2.59
15.5	3230	3131.52	3.05
16.0	3370	3232.76	4.07

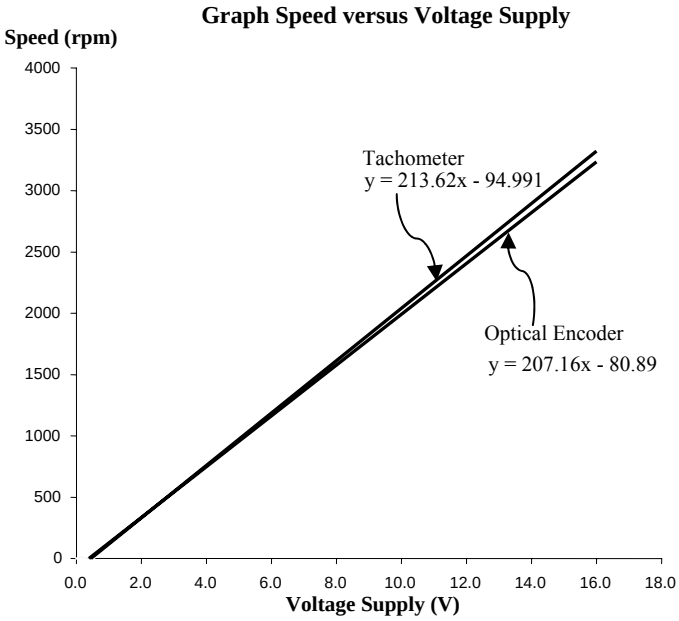


Figure 4.3 Graph of speed versus voltage supply

The accuracy of the optical encoder can be checked by calculate its standard deviation, s for percentage of error by using Equation 4.1,

$$s = \sqrt{\frac{\sum_i (x_i - \bar{x})^2}{n-1}} \qquad (4.1)$$

s = 1.2801

The standard deviation percentage of error is quite small, so it can be concluded that the reading of the optical encoder for speed measurement is quite reliable which is given by,

$$\textbf{rpm = 207.16V -80.89} \qquad (4.2)$$

where rpm = motor speed

V = voltage supply to motor

4.3 DC Motor Speed Control Result

Microcontroller acts as proportional (*P*) controller in the DC motor speed control system. At each speed, the result was collected by applying normal load, overload and then suddenly the load was as no load condition (Refer Figure 3.11). The performance of the system at each speed is shown in Figure 4.4 to Figure 4.10 respectively.

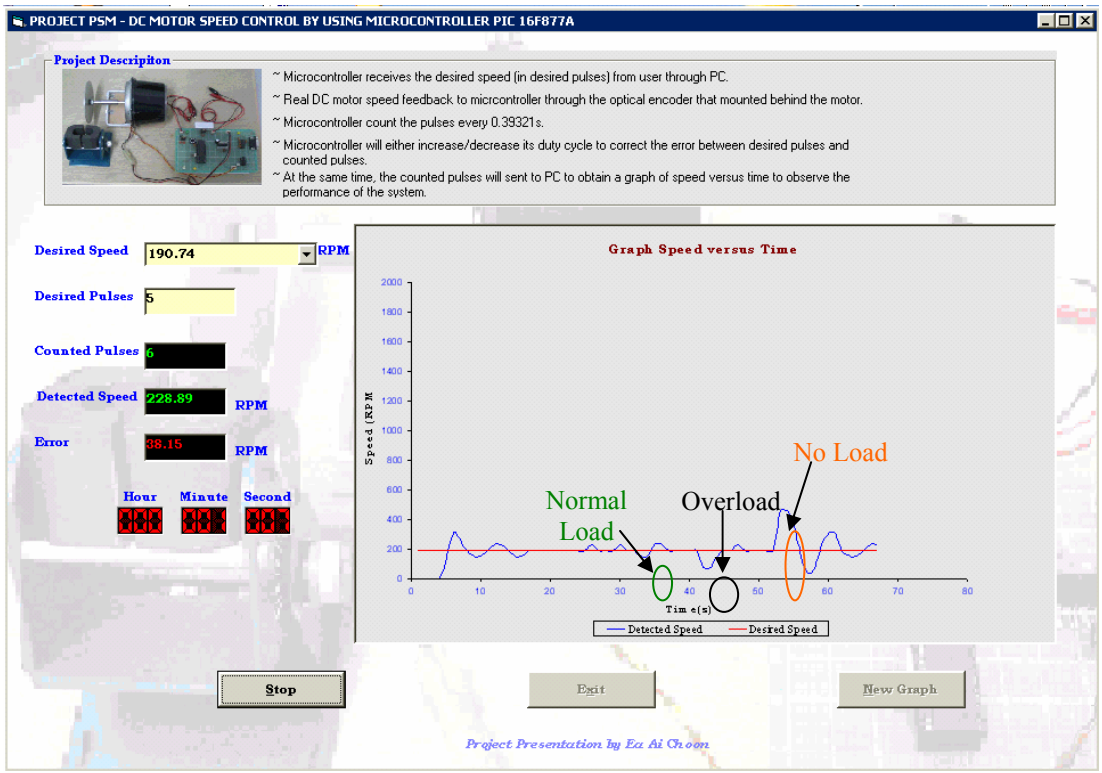


Figure 4.4 DC motor running at speed 190.74 rpm

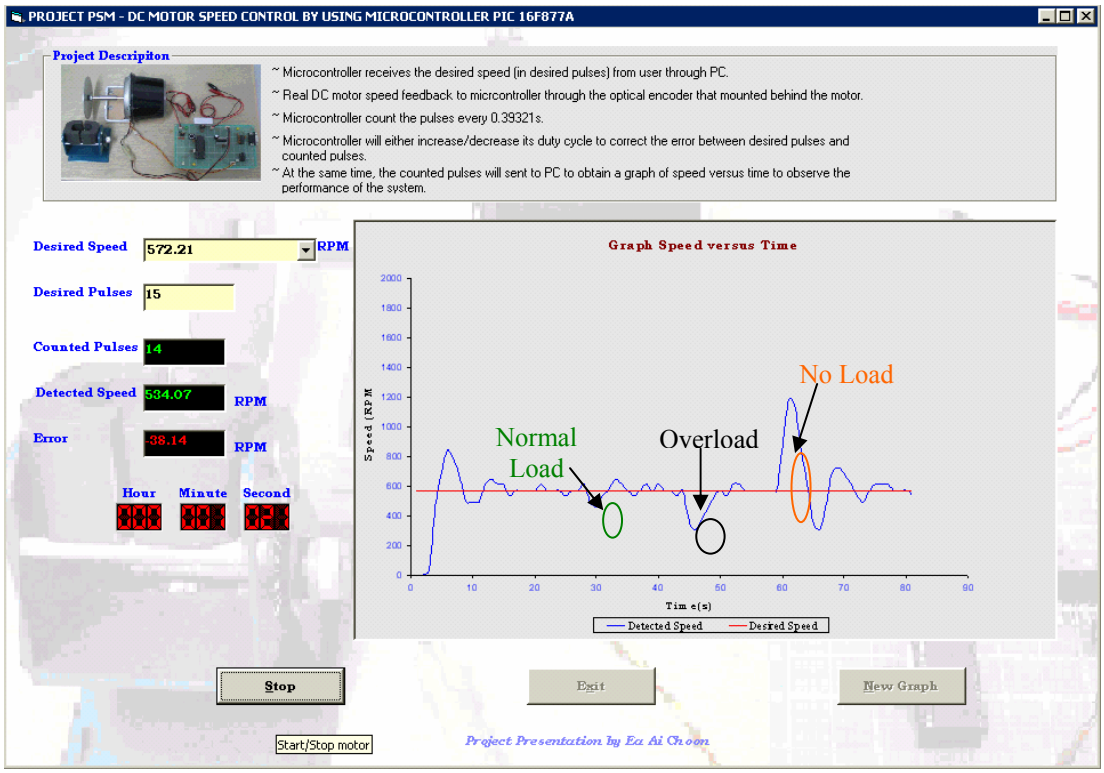


Figure 4.5 DC motor running at speed 572.21 rpm

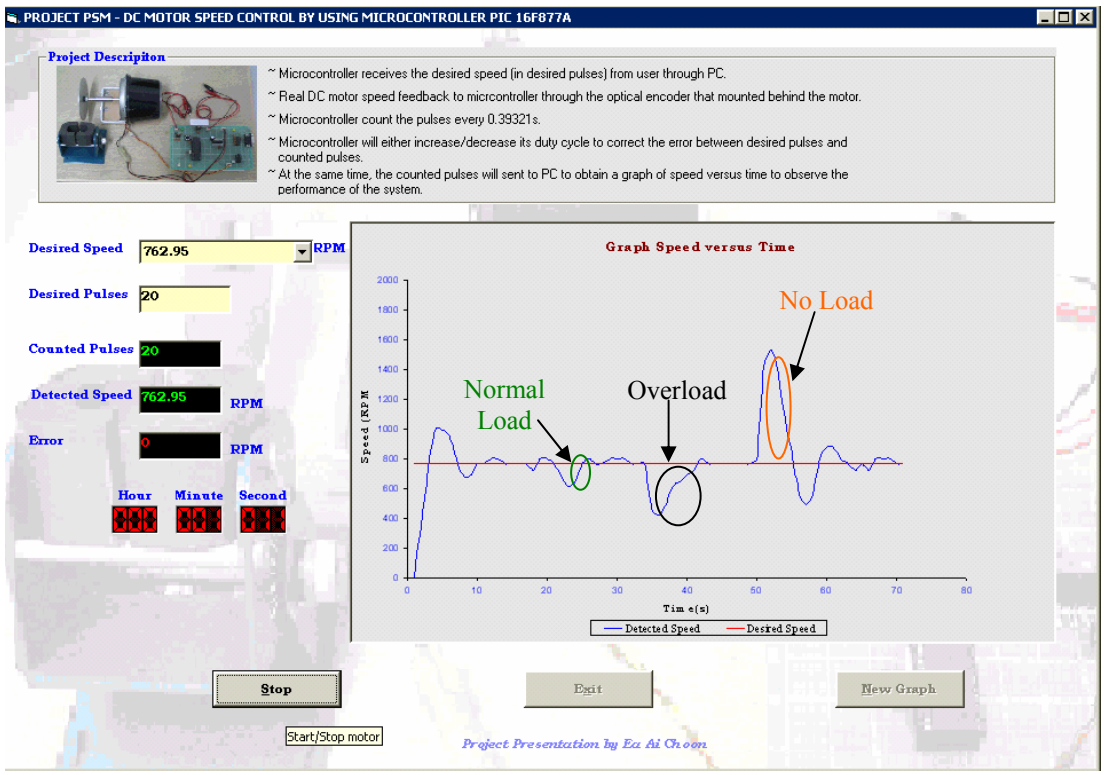


Figure 4.6 DC motor running at speed 762.95 rpm

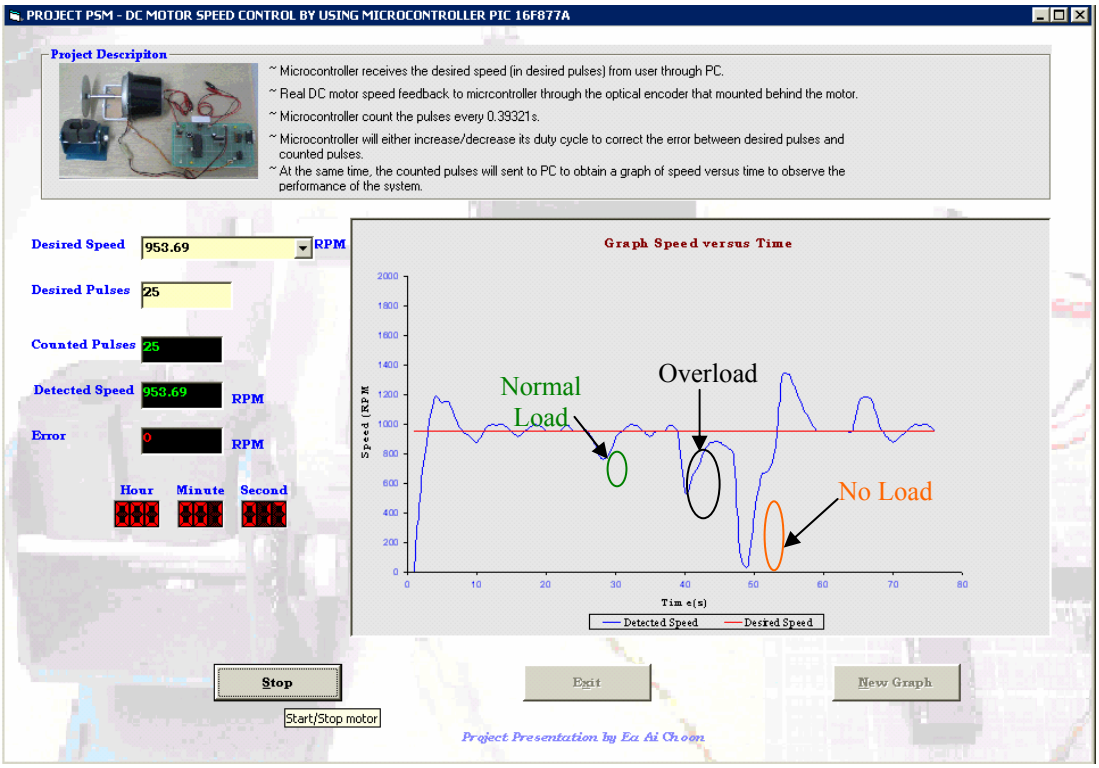


Figure 4.7 DC motor running at speed 953.69 rpm

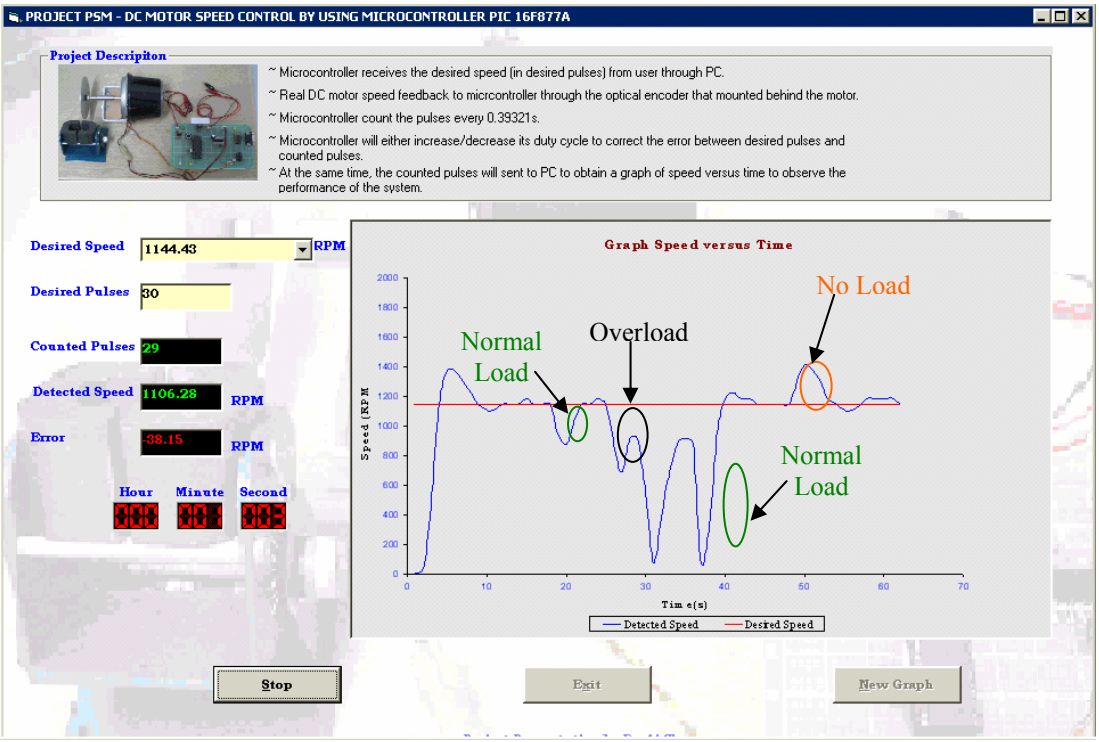


Figure 4.8 DC motor running at speed 1144.43 rpm

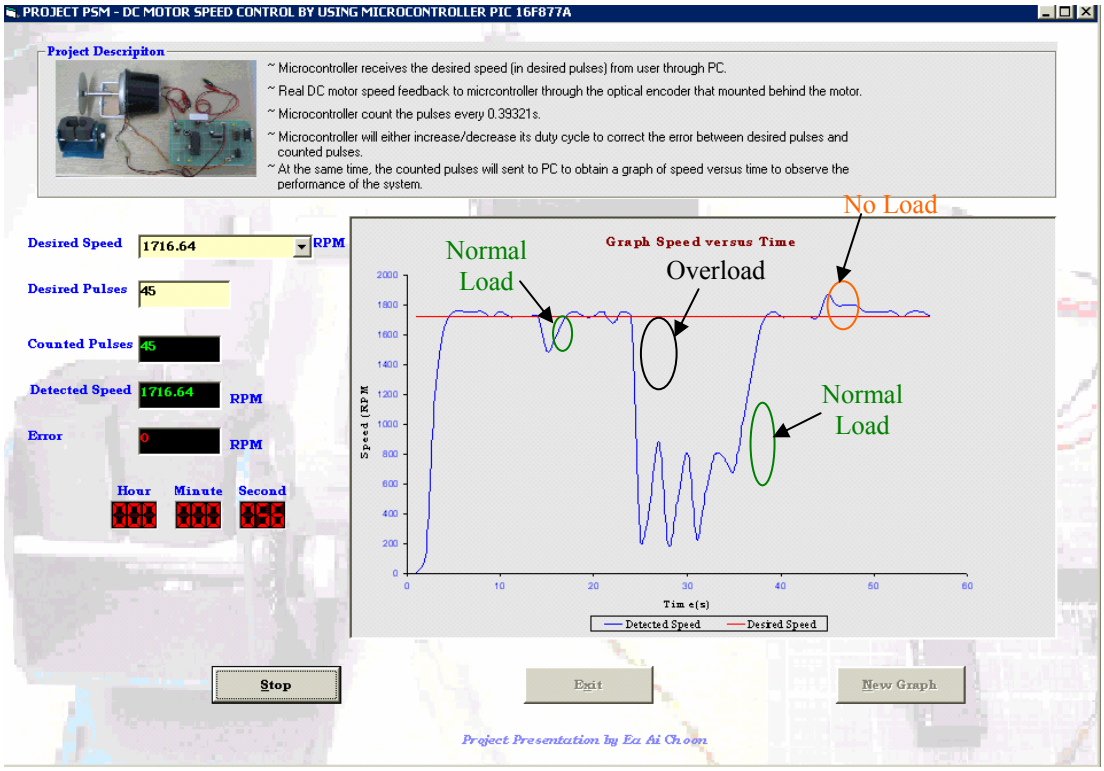


Figure 4.9 DC motor running at speed 1716.64 rpm

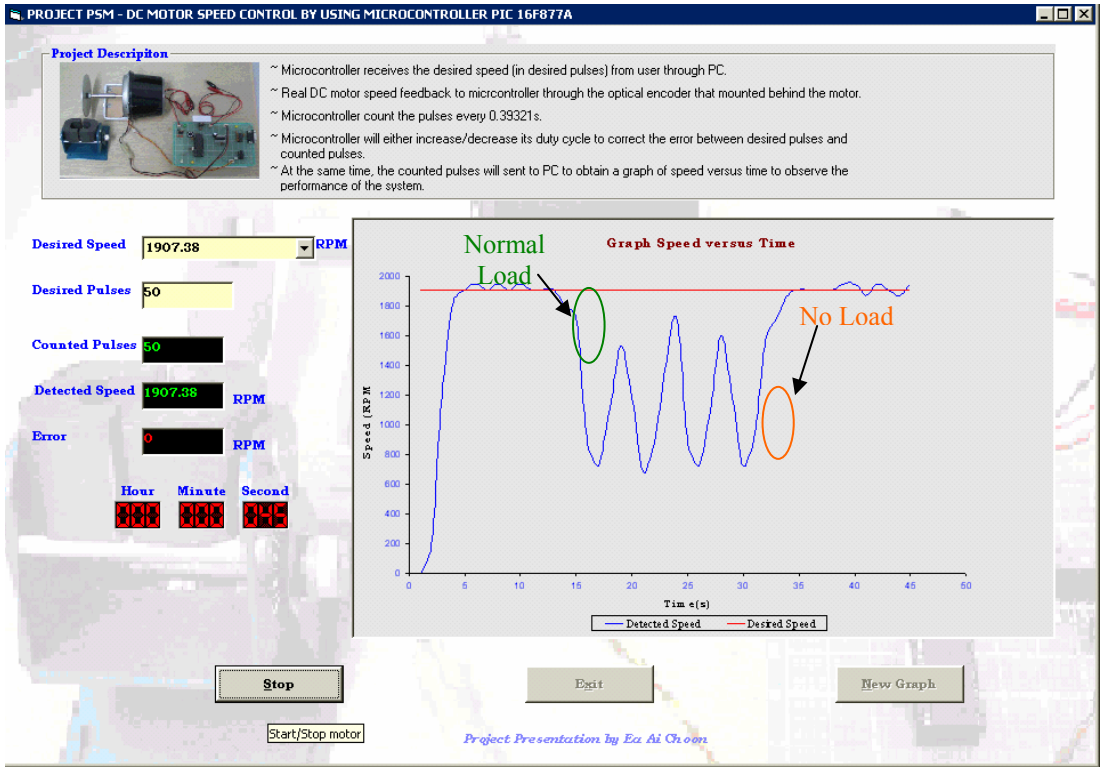
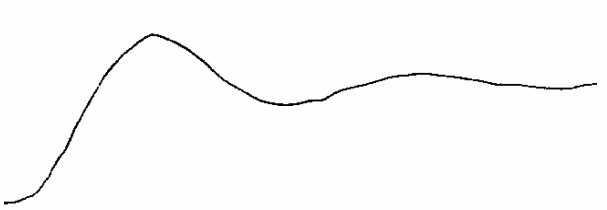


Figure 4.10 DC motor running at speed 1907.38 rpm

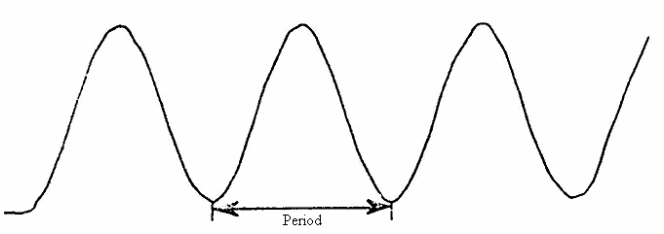
4.3.1 DC Motor Speed Control Result Analysis

For *P* controller, as the controller proportional gain, K_p is increased, the response to set point changes becomes more oscillatory, commonly called underdamped (Figure 4.14a). In this project, the speed response will give an underdamped response when K_p=1.

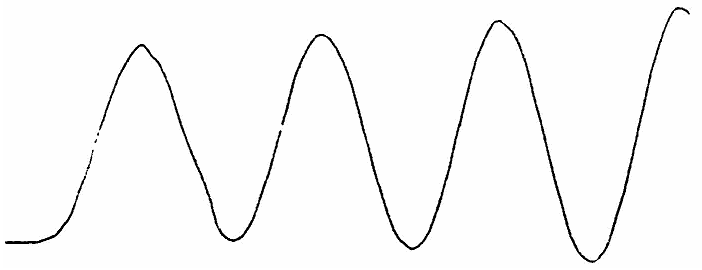
At some greater gain, the response of the control loop becomes a steady-state oscillation (Figure 4.11b). The system is called "marginally stable". If the gain is increased past the point where steady oscillation is observed, the control loop will become unstable and the oscillations will increase in amplitude (Figure 4.11c) [15].



(a)



(b)



(c)

Figure 4.11 Types of oscillatory response: (a) underdamped (b) sustained oscillation (c) unstable.

From graph speed versus time for speed from 190.74 rpm to 762.95 rpm, the controller is able to control the speed at their desired speed when applying normal load and overload. Without controller, the motor will slow down or maybe die out.

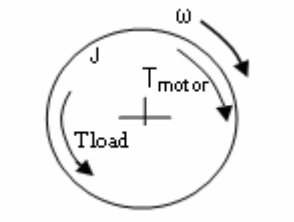


Figure 4.12 Free body diagram of the disc

$$T_{motor} - T_{load} = J\omega \qquad (4.3)$$

where J = moment of inertia of the wheel about the axis of rotation

T_{load} = torque induced by load

T_{motor} = torque induced by voltage supply

ω = speed motor

Figure 4.12 shows a free body diagram of the disc [16]. Based on Equation 4.3, if the load is removed suddenly where $T_{load} = 0$, the motor will speed up before T_{motor} motor is changed (as seen in graph in no load condition). With controller, the motor is able to maintain back to the desired speed in a period of time.

For motor speed in the range of 953.69 rpm to 1716.64 rpm, the controller can control at normal load but not in overload condition. In overload condition, the speed response is oscillating until the load is removed. It is because T_{motor} induced by the voltage supply (maximum 12V) is not enough to overcome the T_{load} of overload condition within these speed.

Due to voltage limitation, the controller is unable to control the motor speed although just apply with normal load for motor speed which is higher than 1907.38 rpm. The motor speed response will become oscillate and unstable.

At each speed, there is a error speed of ±38.15 rpm when at the steady state. This is due to over one count or miss one count pulse of the speed counter that sent

by the optical encoder. Based on Equation 3.4, for one count pulse,N =1, the speed is given by,

$$1 \text{ count speed} = \frac{1}{4} * \frac{1}{0.39321} * 60$$

$$= 38.15 \text{ rpm}$$

4.4 Dead Time Analysis

Dead time is the time of values of the quantity being measured for which it gives no reading [17]. The motor that does not respond at very low input voltage supply due to frictional forces exhibits nonlinearity. There is a dead time at each speed. Figure 4.13 shows a graph of dead time versus motor speed.

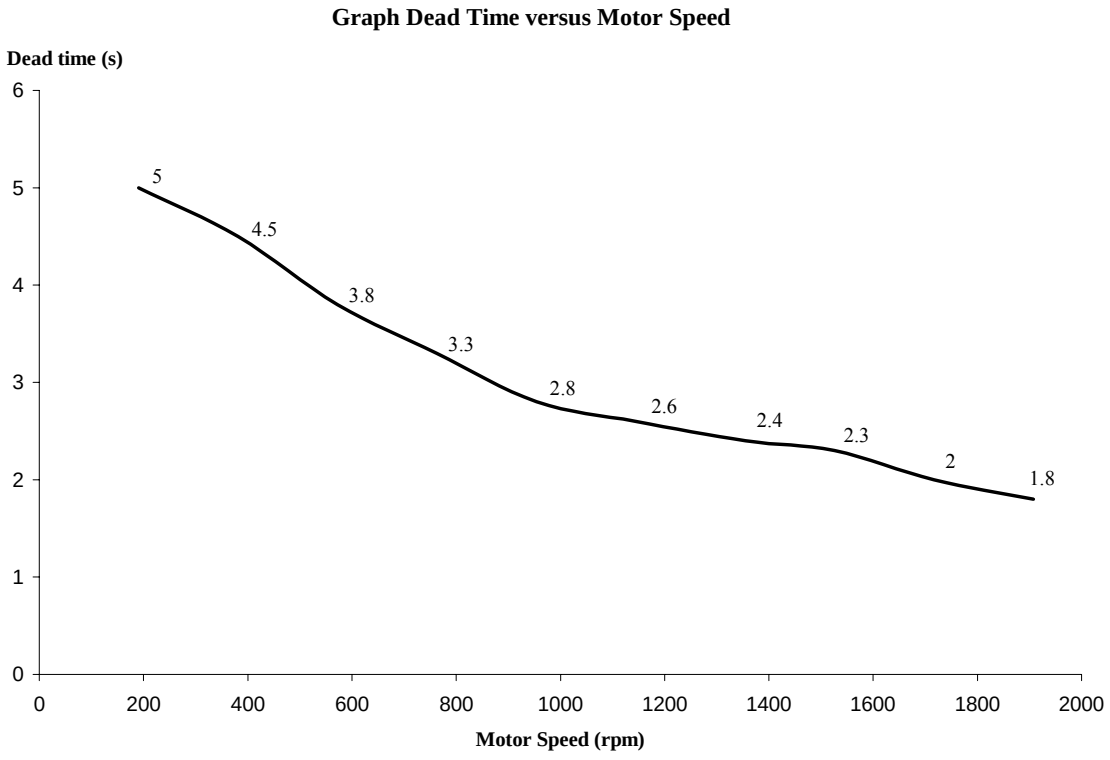


Figure 4.13 Graph of dead time versus motor speed

From Figure 4.13, the dead time of the system is decreasing as the motor speed is increasing. This is because for higher speed, the error speed at starting up condition will be higher. So, more voltage will supply to motor to induce more T_{motor} to overcome frictional forces. Thus, the motor will start running earlier and the dead time become smaller.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

Recent developments in science and technology provide a wide range scope of applications of high performance DC motor drives in area such as rolling mills, chemical process, electric trains, robotic manipulators and the home electric appliances require speed controllers to perform tasks. DC motors have speed control capabilities, which means that speed, torque and even direction of rotation can be changed at anytime to meet new condition.

The goal of this project is to design a DC motor speed control system by using microcontroller PIC16F877A. It is a closed-loop real time control system. The controller will maintain the speed at desired speed when there is a variation of load. By varying the PWM signal from microcontroller (*P* controller) to the motor driver, motor speed can be controlled back to desired value easily.

For this project, by applying $K_p = 1$ to *P* controller in microcontroller, the speed response become underdamped response. If $K_p < 1$, the speed response is not satisfied. At some greater gain, the speed response of the control loop becomes a steady-state oscillation. If the gain is increased past the point where steady oscillation is observed, the control loop will become unstable and the oscillations will increase in amplitude. The motor will suddenly speed up and it will damage the motor.

In conclusion, with the K_p =1 for *P* controller at microcontroller PIC 16F877A, the motor speed response can be maintained at desired value although there is a variation of load. The objective of the project is successfully fulfilled.

5.2 Problems

Although the controller can function as we expected, but the performance is slightly sluggish where it takes about 2 or 3 second to react properly when there is a disturbance especially at low speed. This is what we need to overcome in order to achieve quick control of motor speed smoothly.

Besides, there is a constraint with the optical encoder that had been built. Based on Equation 3.4, if we can increase the resolution of optical encoder by increasing its number of slots, then the sensitivity of optical encoder can be increased. So, the time for getting data and for microcontroller take correct action will be reducing where $t < 0.39321$s. As a result, the controller can react faster when there is a disturbance.

However, due to the size of photodiode which is quite big, the number of slots at cardboard that can be used is only four. Otherwise, the light of the LED cannot be blocked by the cardboard and photodiode will always give High (H) output. Then, the speed sensor is fail.

5.3 Recommendation

The performance of the system is slightly sluggish. For future works, some recommendations have been listed based on the problems in order to improve the performance.

i. Mathematical modeling of motor response

Mathematical model can be obtained from the graph of motor speed response. Then, from the mathematical model, it can be simulated using software such as Matlab to get the improved motor speed response by using controller packages such as PID controller, Fuzzy Logic Controller and others. Besides, it will reduce the total hardware complexity and cost at the same time.

ii. Hardware Improvement

Use infra red (IR) as transmitter and receiver for optical encoder for more narrow light. So, these will allow us to increase the resolution of optical encoder by increasing the number of slots. Thus, it will reduce the steady state error (based on Equation 3.4). Besides, time for getting data and for controller to take action also can be reduced. So, the motor speed response will become better.

iii. Software Improvement

Use fuzzy logic microcontroller which combine the idea of fuzzy logic in microcontroller to obtain a DC motor speed control system with excellent regulation and high robustness.

REFERENCES

1. Muhammad H. Rashid. *Power Electronics Circuits, Devices and Applications*. 3rd edition. United States of America: Prentice Hall. 2004.

2. Christopher A. Adkins and Michael A. Marra, Modeling of a Phase-Locked Loop Servo Controller with Encoder Feedback. *IEEE Spectrum,* August 1999. 51-56.

3. Moore, A.W. Phase-Locked Loops for Motor-Speed Control. *IEEE Spectrum*, April 1973. 61-67.

4. P. C. Sen and M. L. MacDonald. Thyristorized DC Drives with Regenerative Braking and Speed Reversal. *IEEE Transactions on Energy Conversion,* 1978, Vol. IECI-25, No. 4: 347-354.

5. http://homepages.which.net/paul.hills/SpeedControl/SpeedControllersBody.html

6. Abu Zaharin Ahmad and Mohd Nasir Taib. A study On the DC Motor Speed Control by Using Back-EMF Voltage. *AsiaSENSE SENSOR*, 2003, pg. 359-364

7. Iovine John. *PIC Microcontroller Project Book.* 2nd Edition. Singapore: Mc Graw-Hill. 121-123; 2000.

8. Lawrence A. Duarte. *The Microcontroller Beginner's Handbook.* 2nd Edition. United States of America: Prompt Publication. 3-5; 1998.

9. http://www.airborn.com.au/rs232.html

10. MPLAB IDE, Simulator, Editor User's Guide

11. Julia Case Bradley, Anita C. Millspaugh. *Programming in Visual Basic 6.0*. Version 6. New York: McGraw-Hill/Irwin. 2002

12. http://www.seattlerobotics.org/encoder

13. http://www.microchip.com

14. *Sistem Kawalan Halaju*, Kertas Kerja Makmal Kawalan I

15. Sjhinskey, FG. *Process Control Systems*. 2nd Edition, Singapore: McGraw-Hill Book Company, 2003.

16. Paraskevopoulos, P.N. *Modern Control Engineering.* New York: Marcel Dekker, Inc. 2002.

17. Norman S. Nise. *Control Systems Engineering*. 2nd Edition. Redwood City, California: The Benjamin/Cummings Publishing Company, Inc. 1995.

APPENDIX A

Program in Microcontroller PIC 16F877A for DC Motor Speed Control

```
;=================================================================
; File name       : psm.asm
; Project         : Dc Motor Speed Control by Using Microcontroller PIC16F877A
; Programmer      : EA AI CHOON
; Date            : 27/1/2005
; Supervisor      : MOHAMAD SHUKRI ABDUL MANAF
;=================================================================
        list            p=16f877A       ; list directive to define processor

        ERRORLEVEL-302
        ERRORLEVEL-305
        #include        <p16f877A.inc>          ; processor specific variable definitions

        __CONFIG _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON &
_HS_OSC & _WRT_OFF & _LVP_ON & _CPD_OFF

;============== HARDWARE CONNECTION============================
;       Crystal - 20MHZ
;       Capacitor       - 22pF
;       RB1(PIN34) & RB2 (PIN35) - CONNECT TO L298(H,L)
;       CCP2 (PIN16) - CONNECT TO ENABLE L298
;       RA0 as speed counter
;=================================================================

;===============MACRO CHANGE BANK ============================
BANK0           MACRO                   ;Change to BANK0
                BCF             STATUS,RP0
                BCF             STATUS,RP1
                ENDM
BANK1           MACRO                   ;Change to BANK1
                BSF             STATUS,RP0
                BCF             STATUS,RP1
                ENDM
;=================================================================
;Set up initialize value and cblock for the variables needed.

BAUD_RATE       EQU     0X81 ;Set Baud Rate 9600 for 20MHZ for USART
                             Communication

CBLOCK      0X20
D1                  ;For Delay
D2                  ;For Delay
D3                  ;For Delay
X1                  ;For Counter
X2                  ;For Gain
REFERENCE           ;For Reference Speed
CHANGE              ;For error correction
ENDC
;-----------------------------------------------------------------------------------------------
```

```
                ORG             0x00
                GOTO            MAIN

                ORG             0x04
                GOTO            INT                     ;Interrupt Process

;-------------------------------------------------------------------------------------------------------

;============== MAIN PROGRAM ===================================

MAIN            BANK0
                CALL            INIT            ; Intialization
STANDBY         CALL            SERIALSCAN      ; Get data for reference speed
                MOVF            REFERENCE,W
                SUBLW           0X00
                BTFSC           STATUS,Z        ; Reference speed = 0?
                GOTO            STANDBY         ; Yes. Get data again
                MOVLW           D'30'           ; Set Timer1 as 30*13.107ms = 0.39321
                                                 where  X2 as gain
                MOVWF           X2


;------------------------------ RA0 - AS SPEED COUNTER  ----------------------------------------
LOOP            BTFSS           PORTA,0
                GOTO            $-1
                CALL            CHECKNOISE
                BTFSC           PORTA,0
                GOTO            $-1
                BTFSS           PIR1,TMR1IF
                GOTO            LOOP

;Check Noise function

CHECKNOISE      CALL            DELAY1   ;If bandwith is less than 1ms, then it is noise
                BTFSC           PORTA,0
                INCF            X1,1            ;Not noise, increase counter
                RETURN
;================================================================

;############################INITIALIZATION#################################

INIT            BANK0
                CLRF            PORTA
                CLRF            PORTB
                CLRF            PORTC
                CLRF            PORTD
                CLRF            X1
                CLRF            X2
                CLRF            REFERENCE
```

```
;---Initialize PORT----------------------------
                BANK1
                MOVLW       0x06
                MOVWF       ADCON1      ;PortA as digital input
                MOVLW       0xFF        ;PortA as input
                MOVWF       TRISA

                MOVLW       0x00        ;Set PortB as output
                MOVWF       TRISB
                BANK0
                BSF         PORTB,1     ;RB1,RB2(pin 34 and 35) sambung ke
                                        pin 10 dan 12 of L298
                BCF         PORTB,2


;----Initialize PWM-----------------------------
                BANK1
                MOVLW       0xFF        ;PWM Setup: Period KHZ(19.152KHZ)
                MOVWF       PR2
                BCF         TRISC,1
                BCF         TRISC,2

                BANK0
                CLRF        TMR2
                MOVLW       0X00        ;Duty Cycle = 0%
                MOVWF       CCPR2L
                MOVLW       0x04        ;ON TMR2, PRESCALE = 1
                MOVWF       T2CON
                MOVLW       0x0C
                MOVWF       CCP2CON     ;PWM Mode

;-----Set TIMER1 AS TIMER MODE------------------
                BANK0
                MOVLW       0X00
                MOVWF       TMR1H       ;Timer= 0000-FFFFH =
                                        ;       65535*1/(20M/4)=13.107ms
                MOVLW       0X00
                MOVWF       TMR1L
                MOVLW       b'00000001' ;Pre=1:1 TMR1=Int TMR1=ON
                MOVWF       T1CON

                BANK1
                MOVLW       b'00000001' ;TMR1IE=1
                MOVWF       PIE1
                BANK0
                MOVLW       b'11000000' ;GIE=1, PEIE=1
                MOVWF       INTCON

;------Set up Serial Port----------------------
SERIAL_SETUP:   BANK1
                MOVLW       0XC0        ;RC6&7->Input,Others output
                IORWF       TRISC,F     ;Keep in file register

                MOVLW       BAUD_RATE
                MOVWF       SPBRG
```

```
                    MOVLW       0X24
                    MOVWF       TXSTA       ;Enable transmission & high baud rate

                    BANK0
                    MOVLW       0X90
                    MOVWF       RCSTA   ;Enable srial port  & continuous reception

                    RETURN

;########################## END OF INITIALIZATION############################


;^^^^^^^^^^^^^^^^^^^^^^^^^ SUBROUTIONE^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

;-----------Correction of Overrun Error----------

OVERRUN_ERROR   BCF         RCSTA,CREN  ;Disable continuous reception
                BSF         RCSTA,CREN  ;Enable continuous reception
                RETURN

;-----------Correction of Framming Error----------

FERR_ERROR      MOVF        RCREG,W             ;Discard Framming Error
                RETURN


;----------Send the detect speed to PC-------------

SERIAL_TRANSMIT:  BTFSS     PIR1,TXIF       ;Check if data TXREG is transfer
                                            to TSR ->TXREG is empty
                  GOTO      $-1
                  MOVWF     TXREG
                  RETURN

;-----------Get data for reference speed --------------------

SERIALSCAN      CALL        OVERRUN_ERROR  ;Correction of Overrun Error
                CALL        FERR_ERROR      ;Correction of Framming Error
                BTFSS       PIR1,RCIF           ;Check if data receive
                GOTO        $-1                 ;Wait until new data

                MOVF        RCREG,W             ;Get received data to W
                MOVWF       REFERENCE
                RETURN

;------------Short Delay----------------------------------------

DELAY1          MOVLW       D'5'            ;PAUSE FOR ABOUT 1mS
                MOVWF       D3
                MOVLW       D'9'
                MOVWF       D2
                MOVLW       D'36'
                MOVWF       D1
                DECFSZ      D1
                GOTO        $-1
```

```
                    DECFSZ      D2
                    GOTO        $-5
                    DECFSZ      D3
                    GOTO        $-9
                    RETURN

DELAY2              MOVLW       D'255'          ;PAUSE FOR ABOUT 1mS
                    MOVWF       D3
                    MOVLW       D'255'
                    MOVWF       D2
                    MOVLW       D'36'
                    MOVWF       D1
                    DECFSZ      D1
                    GOTO        $-1
                    DECFSZ      D2
                    GOTO        $-5
                    DECFSZ      D3
                    GOTO        $-9
                    RETURN

;^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^


;********************** INTERRUPT  PROCESS*****************************

INT                 CLRF        PIR1            ; Clear interruption flag
                    DECFSZ      X2,F            ; Haven't reach 0.799527s
                    GOTO        CONTINUE        ; Continue counter

GETDATA             MOVF        X1,W            ; Get value of counter,detect speed
                    SUBWF       REFERENCE,W     ; Ref speed - Detect speed
                    MOVWF       CHANGE          ; CHANGE = error correction

                    BTFSC       STATUS,C        ; Reference < Detect ?
                    GOTO        GREATER         ; No. Jump to > or = check

LESS                MOVF        CHANGE,W
                    ADDWF       CCPR2L,F        ; Add duty cycle with error correction
                    GOTO        SHOWSPEED

GREATER             BTFSC       STATUS,Z        ; Reference = Detect?
                    GOTO        SHOWSPEED       ; Yes, no need correction.

                    MOVF        CHANGE,W        ; Detect speed > ref speed, decrease
                                                duty cycle with error correction
                    SUBLW       0XFF; Convert the negative value to positive value
                    SUBWF       CCPR2L,F


SHOWSPEED           MOVF        X1,W
                    CALL        SERIAL_TRANSMIT
                    CALL        OVERRUN_ERROR   ;Correction of Overrun Error
                    CALL        FERR_ERROR      ;Correction of Framming Error

                    CLRF        X1              ;Reset Counter
```

```
                    MOVLW     D'30'              ;Reset Timer
                    MOVWF     X2

                    CALL      SERIALSCAN     ; Get data for reference speed
                    MOVF      REFERENCE,W
                    SUBLW     0X00
                    BTFSS     STATUS,Z           ; Reference speed = 0?
                    GOTO      CONTINUE           ; No, continue the process

STOPMOTOR           MOVLW     0X00               ; Yes. Stop motor
                    MOVWF     CCPR2L
                    CALL      DELAY2

CONTINUE            RETFIE

;****************** END OF INTERRUPT PROCESS************************


                    END

;-----------------------------------END OF PROGRAM-----------------------------------------------
```

APPENDIX B

Source Code of Visual Basic 6.0 for DC Motor Speed Control Using Microcontroller PIC 16F877A

```
Option Explicit
   Dim pulse As Integer              'Varible for desired_pulse
   Dim str As Variant                'Count pulse received from microcontroller
   Dim speed As Double               'Real speed
   Dim desired_speed As Double
   Dim Error_speed As Double
   Dim objExcel As Excel.Application   ' Setup for OLE graph
   Dim wExcel As Excel.Workbook
   Dim xlchart As Excel.Chart
   Dim j As Integer


Private Sub cmdExit_Click()

   objExcel.Visible = True       'Exit the form and show the graph in Microsoft Excel
   End

End Sub


Private Sub cmdNew_Click()

 'Clear the graph
   Workbooks("Speed2").Sheets("Sheet1").Range("A1:E1000") = ""
   j = 2
   OLE_Speed.Update
   Counter1.Value = 0
   Counter2.Value = 0
   Counter3.Value = 0

End Sub


Private Sub cmdRun_Stop_Click()

 If cmdRun_Stop.Caption = "&Run" Then

      desired_speed = Val(cboSpeed.Text)                  ' Get desired speed from user
      pulse = Round(desired_speed / 60 * 4 * 0.39321) 'Convert the desired speed
                                                          into desired pulse
```

```
 If Val(cboSpeed.Text) > 0 Then          ' If receive desired speed from user
        MSComm1.Output = Chr$(pulse)    ' Send desired count pulse to
                                          microcontroller
        lblPulse.Caption = pulse
        Counter1.TimerEnabled = True    ' Start the counter as timer
        cboSpeed.Locked = True          ' Lock the desired speed selection
        cmdRun_Stop.Caption = "&Stop"
        cmdExit.Enabled = False         ' Disable the command of Exit and New Graph
        cmdNew.Enabled = False

      Else
        MsgBox "You must select a desired speed first!", vbOKOnly, "Invalid Data"
        ' If no desired speed receive from user
        MSComm1.Output = Chr$(0)
        cmdExit.Enabled = True
        cmdNew.Enabled = True
      End If

   ElseIf cmdRun_Stop.Caption = "&Stop" Then
      Counter1.TimerEnabled = False                 ' Stop the timer
      cmdRun_Stop.Caption = "&Run"
      pulse = 0
      MSComm1.Output = Chr$(0)          ' Send speed = 0 rpm to stop the motor
      cboSpeed.Locked = False
      cmdExit.Enabled = True
      cmdNew.Enabled = True
   End If

End Sub


Private Sub Form_Load()

'MSComm setup

   With MSComm1
      .CommPort = 2                  ' Use CommPort2 to communicate with
                                       microcontroller
      .Settings = "9600,N,8,1"       ' Baud rate 9600, none parity, 8 data bits, 1 stop bit
      .InBufferSize = 1024           ' Receiver buffer = 1024 bytes
      .OutBufferSize = 1024          ' Transmitter buffer = 1024 bytes
      .DTREnable = True              ' Enable the Data Terminal Ready signal
      .EOFEnable = False             ' Disable the End of File type
      .Handshaking = comNone         ' Disable all network handshaking
      .InputLen = 1                  ' Read all the charater in buffer
      .InputMode = comInputModeText ' Set the incoming messages to ve ASCII text
                                          characters
      .NullDiscard = False           ' Discard bytes that are all zero's
      .RThreshold = 1                ' Set the Receive Oncomm event to occur after 1
                                       byte of data have been received
```

```
    .RTSEnable = True          ' Enable the request to send data
    .SThreshold = 1            ' Set the Transmitter Oncomm event to occur after 1
                                 byte of data have been received
    .PortOpen = True           ' Open CommPort
  End With


'Setup for excel file

  Set objExcel = GetObject("", "Excel.Application")
  Set wExcel = objExcel.Workbooks.Open("C:\Speed2.xls") ' File Speed2 as storage
  objExcel.Visible = False
  j = 2

  With Workbooks("Speed2").Sheets("Sheet1")
       .Cells(1, 1) = 0
       .Cells(2, 1) = 0
  End With

  OLE_Speed.Update

End Sub


Private Sub MSComm1_OnComm()

  If MSComm1.CommEvent = comEvReceive Then
  ' This is used when data is received
    str = Asc(MSComm1.Input)          'Get the counter pulse from microcontroller
    lblCount_pulse.Caption = str
    speed = Round(str / 4 / 0.39321 * 60, 2) ' Convert counter pulse into speed(rpm)
    Error_speed = Round((speed - desired_speed), 2)   ' Calculate the error between
                                                        detected speed with the
                                                        desired speed
    lblSpeed.Caption = speed
    lblError.Caption = Error_speed
    MSComm1.Output = Chr$(pulse) 'Always send desired pulse to microcontroller
  End If


  If Counter1.Value > 0 Then
    If (Counter3.Value * 60 * 60) + (Counter2.Value * 60) + Counter1.Value >
Workbooks("Speed2").Sheets("Sheet1").Cells(j - 1, 1).Value Then

      'Plotting graph
      With Workbooks("Speed2").Sheets("Sheet1")
        .Cells(j, 1) = (Counter3.Value * 60 * 60) + (Counter2.Value * 60) +
                       Counter1.Value
        .Cells(j, 2) = Format(speed, "##.##")
        .Cells(j, 3) = desired_speed
```

```
        End With

        j = j + 1
        OLE_Speed.Update    ' Update the graph
        End If

    End If

    'Timer : Counter1 as second, Counter2 as minute, and Counter3 as hour

    If Counter1.Value = 60 Then

        Counter2.Value = Counter2.Value + 1
        Counter1.Value = 0
        Counter1.TimerEnabled = True
        If Counter2.Value = 60 Then
            Counter3.Value = Counter3.Value + 1
            Counter2.Value = 0
        End If

    End If

End Sub
```