DESIGN AND IMPLEMENTATION OF PID CONTROLLER FOR DC MOTOR USING PIC

MOHD HAFIZ BIN OMAR

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

BORANG PENGESAHAN STATUS TESIS♦

JUDUL: **INTELLIGENT NUMBER RECOGNITION**

SESI PENGAJIAN: **2009/2010**

Saya **MOHD HAFIZ BIN OMAR (851105-02-5999)**
(HURUF BESAR)

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

1. Tesis adalah hakmilik Universiti Malaysia Pahang (UMP).
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis ini 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**	

Disahkan oleh:

_____ (TANDATANGAN PENULIS)

_____ (TANDATANGAN PENYELIA)

Alamat Tetap:

F-216,LRG TIMAH 4,
TAMAN P.K.N.K,
09000 KULIM ,
KEDAH DARUL AMAN

AHMAD NOR KASRUDDIN BIN NASIR
(Nama Penyelia)

Tarikh: **24 NOVEMBER 2009**

Tarikh: **24 NOVEMBER 2009**

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 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).

DESIGN AND IMPLEMENTATION OF PID CONTROLLER FOR DC MOTOR USING PIC

MOHD HAFIZ BIN OMAR

This Thesis is Part Fulfillment of the Requirement for a Bachelor Degree of Electrical Engineering (Power System)

Faculty of Electrical & Electronic Engineering
University Malaysia Pahang

NOVEMBER 2009

DECLARATION

"I declare that this thesis entitled 'design and implementation of pid controller for dc motor using PIC' is the result of my own research except as cited in references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree",

Signature :.....

Name of candidate : Mohd Hafiz bin Omar

Date : November 24, 2009

DEDICATION

Special dedicated to my family, my friends, my fellow colleague,
and to all faculty members

For all your care, support, and believe in me.

Sincerely;
Mohd Hafiz bin Omar

ACKNOWLEDGEMENT

Alhamdulillah, the highest thank to God because with His Willingness I possible to complete the final year project. In preparing this thesis, I was in contact with many people, researchers, academicians, and practitioners. They have contributed towards my understanding and thoughts. In particular, I wish to express my sincere appreciation to my main thesis supervisor, Mr. Ahmad Nor Kasruddin Nasir, for encouragement, guidance, critics and friendship.

I would also like to thank to all UMP lecturers and electrical technicians whom had helped directly or indirectly in what so ever manner thus making this project a reality.

My special thanks for my parents, Mr. Omar bin Saad and Mrs Sariah binti Mat Ariffin and also the rest of my family, for their financial, spiritual support and pray on me throughout this project. Their blessing gave me the high-spirit and strength to face any problem occurred and to overcome them rightly.

The episode of acknowledgement would not be complete without the mention of my fellow colleagues in BEP, 2005/06 session. Finally, I would like to thank all whose direct and indirect support helped me completing my thesis in time. Only Allah can repay your kindness.

ABSTRACT

The purpose of this study is to control the speed of direct current (DC) motor with PID controller using Proportional Integral Derivative (PID). The PID Controller will be design and must be tune, so the comparison between simulation result and experimental result can be made. The scopes includes the simulation and modeling of direct current (DC) motor, implementation of Proportional Integral Derivative (PID) Controller into actual DC motor and comparison of MATLAB simulation result with the experimental result. This research was about introducing the new ability of in estimating speed and controlling the permanent magnet direct current (PMDC) motor. In this project, PID Controller will be used to control the speed of DC motor. The PID Controller will be programmed to control the speed of DC motor at certain speed level. The sensor will be used to detect the speed of motor. Then, the result from sensor is fed back to PIC to find the comparison between the desired output and measured output to get the estimating speed.

ABSTRAK

Tujuan utama kajian ini adalah untuk mengawal kelajuan *Direct Current (DC) Motor*, di mana *Proportional Integral Derivative (PID)* akan menjadi pengawal kelajuan utama dan diaplikasi menggunakan PIC. *Proportional Integral Derivative (PID)* akan direka bentuk dan harus disesuaikan nilai komponennya supaya perbezaan di antara keputusan simulasi dapat dibandingkan dengan keputusan eksperimen. Skop tugasan kajian ini termasuklah simulasi dan model *direct current (DC) motor*, perlaksanaan *Proportional Integral Derivative (PID)* ke dalam DC motor yang sebenar dan perbandingan keputusan simulasi MATLAB dengan keputusan eksperimen. Kajian ini adalah untuk memperkenalkan keupayaan baru dalam menaksir dan mengawal kelajuan *Permanent Magnet Direct Current (PMDC) motor*. Di dalam projek ini, *PID Controller* akan digunakan untuk mengawal kelajuan DC motor. *PID Controller* juga akan diprogramkan untuk mengawal kelajuan motor menggunakan PIC pada kadar kelajuan yang telah ditetapkan. Alat pengesan (*Encoder*) akan mengesan tahap kelajuan motor. Selepas itu, keputusan daripada alat pengesan akan di suap kembali kepada *PIC* untuk mencari perbandingan di antara keputusan yang kehendaki dengan keputusan sebenar untuk mencapai kelajuan yang telah ditetapkan.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS/ABBREVIATIONS

PID	-	Proportional Integral Derivative
BLDC	-	Brushless Direct Current
PIC	-	Peripheral Interface Controller
DC	-	Direct Current

CHAPTER 1

INTRODUCTION

1.1 Overview

Now a days DC motor plays a vital role in most of the industrial areas, it can be seen in most of the electronic devices. They are mainly used for the mechanical movements of physical applications such as rolling the bundle of sheets or CD drives, lifts etc. Many methods evolved to control the revolution of a motor. DC motors can be controlled either by software or directly by hardware. Software controlling needs computers which are bulky and common man cannot afford for it, so hardware controls are in use. Even in hardware if it is programmable device then it is preferred because it can be modeled according to the requirements of the user.

There are many different control schemes such as PID, Fuzzy, LQR, Integral State Feedback and some other more. For this project I implemented a PID (Proportional, Integral, and Differential) control loop to control the speed of the motors. A scaled down model of this controlling scenario is created by inducing disturbances in this scaled down model and by taking feedback from the output, we

will restore the system to a set value by using the Proportional Integral Derivative (PID) control scheme.

The PID controller calculation involves three separate parameters; the Proportional(P), the Integral(I) and Derivative(D) values. The Proportional value determines the reaction to the current error, the Integral determines the reaction based on the sum of recent errors and the Derivative determines the reaction to the rate at which the error has been changing. The weighted sum of these three actions is used to adjust the speed of the dc motor via the microcontroller. The speed of the motors is manipulated by altering the duty cycle of a PWM signal generated by the processor. The duty cycle of this signal is known as the control value. In this way, the control value is continuously updated based on the response of the motors. This ensured that the motors are moving at the desired speed despite drag, obstacles, or other unexpected track conditions.

1.2 OBJECTIVE

The objective of this project is to design a PID sub-routine in microcontroller for motor speed control. The PID algorithm written as a computer program will be embedded in a hardware device which is the microcontroller. This designed is to correct some errors on Direct Current (DC) Motor control which are:

a) DC motor steady-state error correction

b) DC motor overshoot control

c) DC motor settling time

1.3 SCOPE OF PROJECT

For this project, there are two scopes. The scope of project is dividing to software part and hardware part:

For the software part, we have:

i. Find the mathematical model of the motor
ii. Get the transfer function
iii. Design and tuning the PID controller
iv. Simulating PID control of the mathematical model of DC motor
v. Programming the PIC

For the hardware part, we have:

i. Design and construct the hardware for microcontroller unit, driver motor, encoder and serial communication.
ii. Implement of PID controller to actual DC motor.
iii. Comparison of the simulation result with the experimental result of controller performance.

1.4 Problem Statement

The problem encounter when dealing with DC motor is the lag of efficiency and losses. In order to eliminate this problem, controller is introduce to the system. There's few type of controller but in this project, PID controller is chosen as the controller for the DC motor. This is because PID controller helps get the output, where we want it in a short time, with minimal overshoot and little error.

CHAPTER 2

LITERATURE REVIEW

This chapter will review previous research which concern to dc motor system, controller algorithm, Proportional Integral Derivative (PID) and microcontroller. There are numbers of control strategy and methods in controlling the speed for dc motor which had been implemented by researchers that will be discussed.

2.1 DC Motor Characteristic

DC motors consist of one set of coils, called an armature, inside another set of coils or a set of permanent magnets, called the stator. Applying a voltage to the coils produces a torque in the armature, resulting in motion. It design to run on DC electric power which uses electrical energy and produce mechanical energy.

DC Motor Types is:

Permanent Magnet	:	No field coils at all.
Series Wound	:	The field coils are connected in series with the armature coil. Powerful and efficient at high speed, series wound motors generate the most torque for a given current. Speed varies wildly with load, and can run away under no-load conditions.
Shunt Wound	:	The field coils are connected in parallel with the armature coil. Shunt wound motors generate the least torque for a given current, but speed varies very little with load. Will not run away under no-load, but may if the field windings fail.

Compound Wound: A combination of series and shunt wound. This is an attempt to make a motor that will not run away under no load or if the field fails, yet is as efficient and powerful as a series wound motor.

2.2 State Space Equation of DC Motor:

To derive state space equation of any system, one needs to determine the inputs, output and state space variable of the system. The output and state space variable of a DC motor are speed (ω) and load current (I_L). The inputs are armature voltage (V_a) and load torque (T_L).

Table 2.1: Description of Motor Parameters

Parameter	**Description**
U_a or V_a	Input Voltage
I_a	Armature Current
L_a	Armature Inductance
R_a	Armature Resistance
J	Rotor Inertia
B_m	Friction Co-efficient
K_a	Back EMF Constant
K_T	Torque Constant or Motor Constant
T_L	Load Torque
Ω or ω_r	Rotor Speed
T_e or T_m	Motor Torque
E_a or V_{emf}	Back EMF Voltage

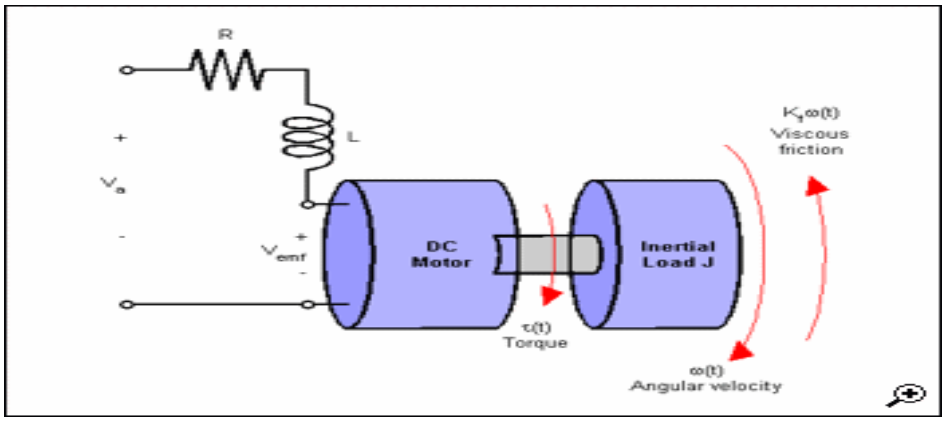

Figure 2.1 : A Simple Model of DC Motor Driving an Inertial Load

First, find the transfer functions to develop the block diagrams of the open- and closed-loop systems. These transfer function are obtained using the differential equations that describe the system dynamics.

From permanent-magnet DC motor, get equation:

$$\frac{di_a}{dt} = -\frac{r_a}{L_a} i_a - \frac{k_a}{L_a} \varpi_r + \frac{1}{L_a} u_a \quad \text{.....(1.1)}$$

Newtonian mechanics is applied to find the differential equation for mechanical system.

$$\sum \vec{T} = J\vec{\alpha} = J \frac{d\vec{\omega}}{dt} \quad \text{.....(1.2)}$$

Where J is equivalent to moment inertia, is found as sum of the moment of inertia of rotor, coupling, gear and rotating platform. The electromagnetic torque develop by permanent-magnet DC motors is found as

$$T_e = k_a i_a \quad \text{.....(1.3)}$$

Denoting the viscous friction coefficient as B_m, the viscous friction torque is

$$T_{viscous} = B_m \omega_r \quad \text{.....(1.4)}$$

The load torque is denoted as T_L. Then, making use of Newton's second law, we have

$$\frac{d\omega_r}{dt} = \frac{1}{J}(T_e - T_{viscous} - T_L) = \frac{1}{J}(k_a i_a - B_m \omega_r - T_L) \quad \text{.....(1.5)}$$

The dynamics of rotor angular displacement is [5]

$$\frac{d\theta_r}{dt} = \omega_r \quad \text{.....(1.6)}$$

To find the transfer function, the derived three first-order differential equations [5]

$$\frac{di_i}{dt} = -\frac{r_a}{L_a} i_a - \frac{k_a}{L_a} \varpi_r + \frac{1}{L_a} u_a \quad \text{.....(1.7)}$$

$$\frac{d\omega_r}{dt} = \frac{1}{J}(k_a i_a - B_m \omega_r - T_L) \quad \text{.....(1.8)}$$

and

$$\frac{d\theta_r}{dt} = \omega_r \quad \text{.....(1.9)}$$

must be rewritten in the *s*-domain.

Using the Laplace operator $s = \frac{d}{dt}$, one immediately obtains

$$\left(s + \frac{r_a}{L_a}\right) i_a(s) = -\frac{k_a}{L_a} \omega_r(s) + \frac{1}{L_a} u_a(s) \quad \text{.....(1.10)}$$

$$\left(s + \frac{B_m}{J}\right) \omega_r(s) = \frac{1}{J} k_a i_a(s) - \frac{1}{J} T_L(s) \quad \text{.....(1.11)}$$

$$s\theta(s) = \omega_r(s) \quad \text{.....(1.12)}$$

From equation (1.7) and (1.8), the state space equations of DC Motor are,

$$\begin{bmatrix} \frac{di}{dt} \\ \frac{d\omega}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{r_a}{L_a} & -\frac{k_a}{L_a} \\ \frac{-k_a}{J} & \frac{-B_m}{J} \end{bmatrix} \begin{bmatrix} i_a \\ \omega_r \end{bmatrix} + \begin{bmatrix} \frac{1}{L_a} \\ 0 \end{bmatrix} u_a \quad \text{.....(1.13)}$$

$$\theta = [0 \quad 1] \begin{bmatrix} 1 \\ \omega \end{bmatrix} \quad \text{.....(1.14)}$$

$$D = [0] \quad \text{.....(1.15)}$$

Thus,

$\dot{x} = Ax + Bu$

$\dot{y} = Cx + Du$

The block diagram of the permanent-magnet DC motor, as a single-input (applied armature voltage)/ single-output (rotor angular displacement) system is illustrated by Figure 2.2:

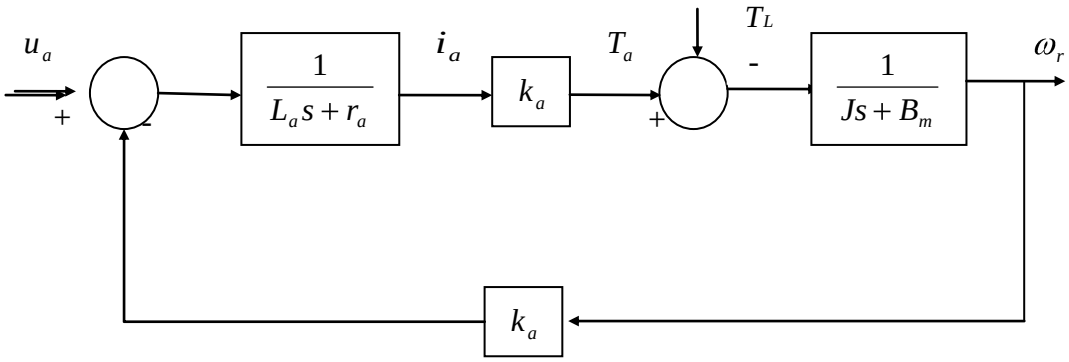

Figure 2.2 : Block diagram of the open-loop permanent-magnet DC motor

The parameters of the permanent-magnet JDH-2250 Clifton Precision motor are:

$r_a = 2.7\Omega$

$L_a = 0.004H$

$B_m = 0.0000093N - m - A^{-1}$

$k_a = 0.105V - s - rad^{-1}$

$J = 0.0001kg - m^2$

Figure 2.3 : Model of DC motor

2.3 Proportional Integral Derivative (PID)

PID controller consists of 3 sub-systems: Proportional controller, Integral (I) controller and Derivative (D) controller. These controllers can be used in different combination according to the purpose and the combinations are P, PI, PD and PID controllers. Proportional integral and derivative constant depend on each other. If we change the value of one, it will effect on other two values. So while designing PID Controllers care must be taken to get required output.

To meet with the design requirements, first the motor can only rotate at 0.1 rad/sec with an input voltage of 1 Volt. Since the most basic requirement of a motor is that it should rotate at the desired speed, the steady-state error of the motor speed should be less than 1%. The other performance requirement is that the motor must accelerate to its steady-state speed as soon as it turns on. In this case, the motor should have a settling time of 2 seconds. Since a speed faster than the reference may damage the equipment, it also need to have an overshoot of less than 5%.Using MATLAB, the original open-loop performances can be plotted as figure below.

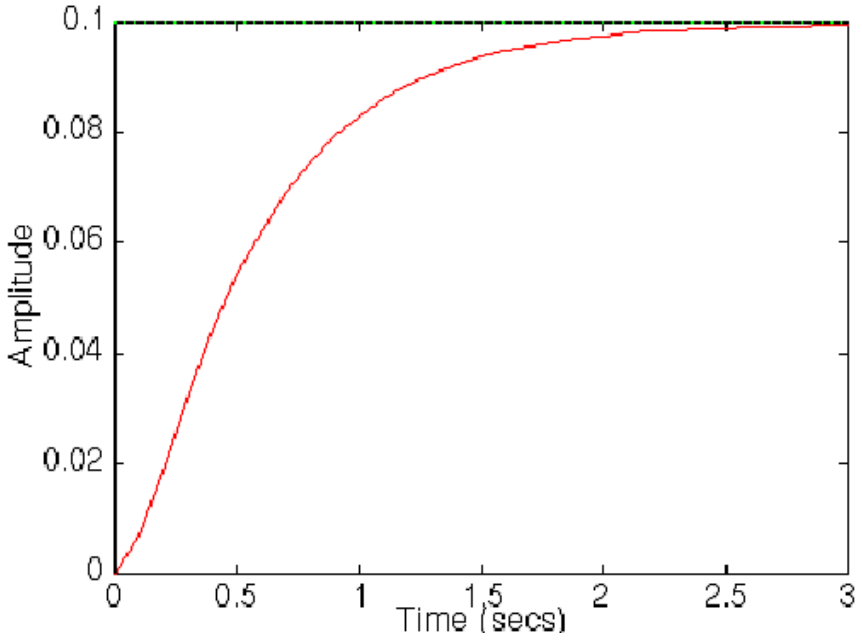

Figure 2.4 Step Response of Open Loop System

If the reference input is simulated by an unit step input, then the motor speed output should have:

1. Settling time less than 2 seconds
2. Overshoot less than 5%
3. Steady-state error less than 1% [1]

2.4 PID Implementation on DC Motor Close Loop Control

The closed-loop controller is a very common means of keeping motor speed at the required set point under varying load conditions. It is also able to keep the speed at the set point value where for example, the set point is ramping up or down at a defined rate.

In the closed loop speed controller, a signal proportional to the motor speed is fed back into the input where it is subtracted from the set point to produce an error signal. This error signal is then used to work out what the magnitude of controller output should be to make the motor run at the required set point speed. For example, if the error speed is positive, the motor is running too fast so that the controller output should be reduced and vice-versa.

If a load is applied, the motor slows down so that a positive error speed is produced. The output increases by a proportional amount to try and restore the speed. However, as the motor speed recovers, the error reduces and so therefore does the drive level. The result is that the motor speed will stabilize at some speed below the set point at which the load is balanced by the error speed times the gain. If the gain is very high so that even the smallest change in motor speed causes a significant change in drive level, the motor speed may oscillate. This basic strategy is known as “proportional control” and on its own has only limited use as it can never force the motor to run exactly at the set point speed.

The next improvement is to introduce a correction to the output which will keep adding or subtracting a small amount to the output until the motor reaches the set point, at which point no further changes are made. In fact a similar effect can be had by keeping a running total of the error speed speeds observed for instance, every 25ms and multiplying this by another gain before adding the result the proportional correction found above. This new term is based on what is effectively the integral of the error speed.

The proportional term is a fast-acting correction which will make a change in the output as quickly as the error arises. The integral takes a finite time to act but has the ability to remove all the steady-state speed error.

A further refinement uses the rate of change of error speed to apply an additional correction to the output drive. This means that a rapid motor deceleration would be counteracted by an increase in drive level for as long as the fall in speed continues. This final component is the "derivative" term and it is a useful means of increasing the short-term stability of the motor speed. A controller incorporating all three strategies is the well-known Proportional-Integral-Derivative, or "PID" controller.

Creating PID algorithm involves lots of concern in terms of the programming. The main issue on implementing PID control system is on how to program the algorithm and correctly functioning as true PID behavior. For the error calculation results, the plant variables might be bigger than Set point value and gives negative Error result. As a solution, the program must have conversion subroutine to ensure the Error result is in positive value. Another aspect to consider is the Integral Windup. Integral term is based on the sum of all previous observed error speeds. However the integral can continuous to integrate indefinitely, thus the microcontroller program must check for overflow on the resulting integral term.

For best performance, the proportional and integral gains need careful tuning. For example, too much integral gain and the control will tend to over-correct for any speed error resulting in oscillation about the set point speed.

Integral gains ensure that under steady state conditions that the motor speed almost exactly matches the set point speed. A low gain can make the controller slow to push the speed to the set point but excessive gain can cause hunting around the set point speed. In less extreme cases, it can cause overshoot where by the speed passes through the set point and then approaches the required speed from the opposite direction. Unfortunately, sufficient gain to quickly achieve the set point speed can cause overshoot and even oscillation but the other terms can be used to damp this out. Proportional gains gives fast response to sudden load changes and can reduce instability caused by high integral gain. This gain is typically many times higher than the integral gain so that relatively small deviations in speed are corrected while the integral gain slowly moves the speed to the set point. Like integral gain, when set too high, proportional gain can cause an oscillation of a few Hertz in motor speed.

There are many ways for an initial setting of the gains. One of it is to set the set point to maximum speed and with the integral and derivative gains at zero, increase the proportional gain so that the speed reaches the maximum possible before a speed oscillation sets in. Reduce the set point to zero. Repeatedly apply a step change in set point to 75% of full speed and increase the integral gain gradually until the speed starts to overshoot.

The speed should rise quickly with the step change and settle at the set point without significant overshoot. The integral gain setting will be particularly influenced by the moment of inertia of the load and some experimentation will be required. The controller is configured as a proportional-integral controller which should quickly correct speed errors without oscillation. [3]

A simulation of how PID controller works can be done through MATLAB. First, the proportional control was put to the test. By using a gain of 100, and by using MATLAB m-file, the following plot is generated.

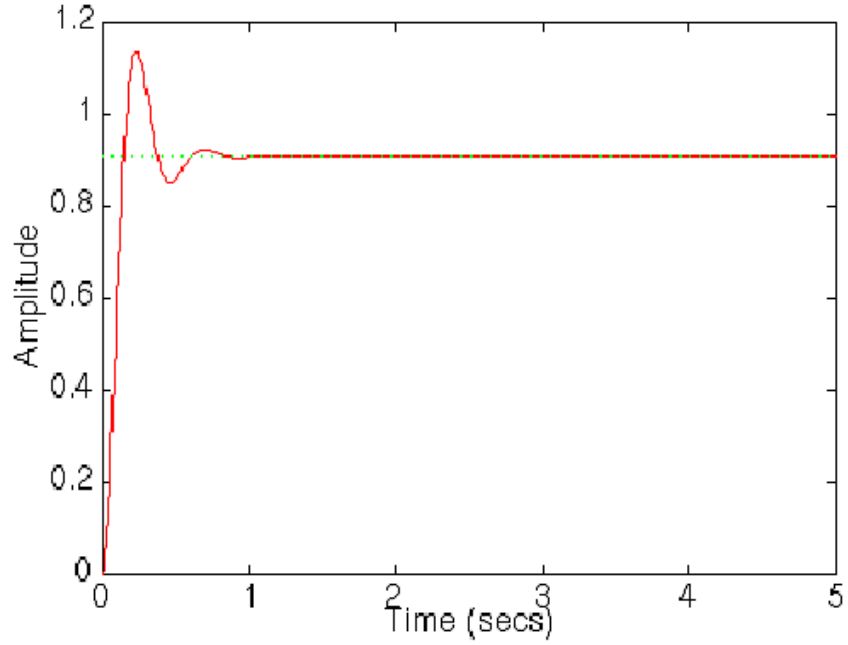

Figure 2.5 Step Response with Proportional Control

From the plot above, the steady-state error and the overshoot are too large. Adding an integral term will eliminate the steady-state error and a derivative term will reduce the overshoot. Inserting a small Ki and Kd to the system and the plot as figure below is obtained.

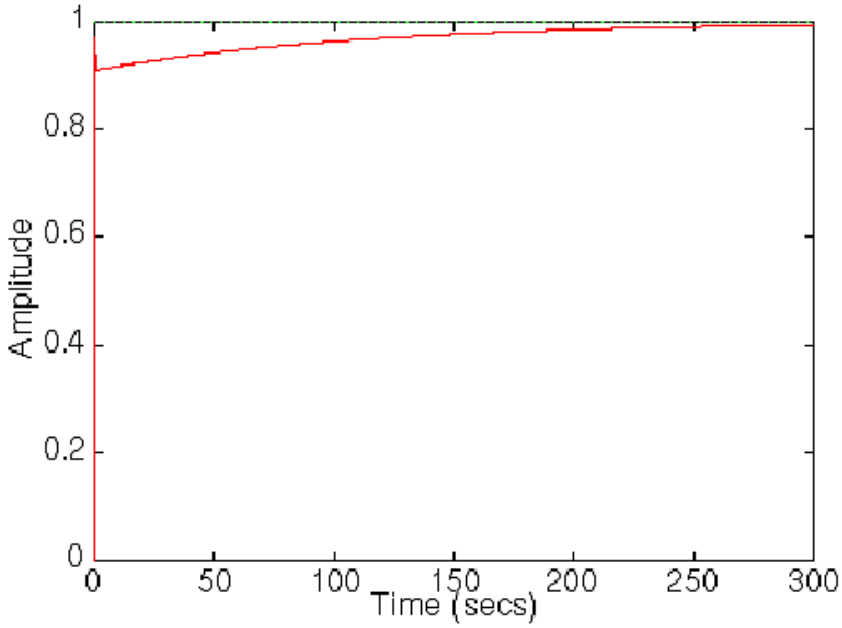

Figure 2.6 PID Control with Small Ki and Kd

From the figure above, it is seen that the settling time is too long. Increasing Ki will reduce the settling time as the figure below.

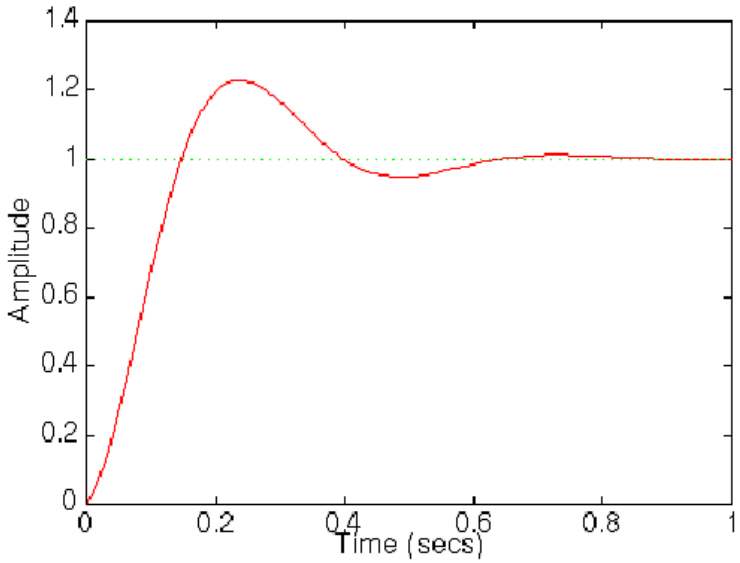

Figure 2.7 PID Control with Large Ki

From the figure above, it is seen that the response is much faster than before, but the large Ki has worsened the transient response and result in big overshoot. Increasing Kd will reduce the overshoot and figure as below is obtained. From the figure above, the design requirements has been achieved. [4]

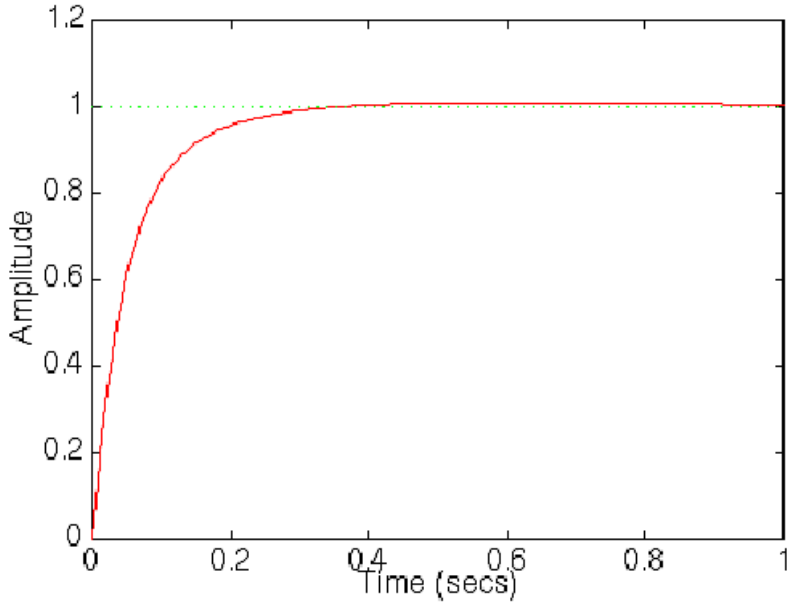

Figure 2.8 PID Control

2.5 PID Tuning

Tuning a PID is the adjustment of its control parameters to the optimum values for the desired control response. The optimum behavior of a process varies depending on the application. There are several methods for tuning a PID. The most effective methods generally involve the development of some form of process model, then choosing P, I and D based on the dynamic model parameters.

Table 2.2: Choosing a Tuning Method

Choosing a Tuning Method		
Method	Advantages	Disadavantages
Manual Tuning	No math required. Online Method	Requires experienced personnel
Ziegler-Nichols	Proven method. Online method	Process upset, some trial-and-error, very aggressive tuning
Software Tools	Consistent tuning. Online or offline method. May include valve and sensor analysis. Allow simulation before downloading	Some cost and training involved
Cohen-Coon	Good process model	Some math. Offline method. Only good for first-order processes.

2.5.1 Manual Tuning

If the system must remain online, one tuning method is to first set the I and D values to zero. Increase P until the output of the loop oscillates, and then the P should be left set to be approximately half of that value. Then increase D until any offset is correct insufficient time for the process. However, too much D will cause instability. Finally, increase I, if required, until the loop is acceptably quick to reach its reference after a load disturbance. However, too much I will cause excessive response and overshoot. A fast PID tuning usually overshoots slightly to reach the set point more

quickly; however, some systems cannot accept overshoot, in which case an over-damped closed- loop system is required, which will require a P setting significantly less than half of that of the P setting causing oscillation.[6]

Effects of increasing parameters				
Parameter	Rise Time	Overshoot	Settling Time	Steady-state error
Kp	Decrease	Increase	Small Change	Decrease
Ki	Decrease	Increase	Increase	Eliminate
Kd	Small Decrease	Decrease	Decrease	None

Table 2.3: Effect of Increasing Parameters

2.6 PIC Microcontroller

PIC is a family of Harvard architecture microcontrollers made by Microchip Technology, derived from the PIC1640 originally developed by General Instrument's Microelectronics Division. The name PIC initially referred to "Programmable Interface Controller", but shortly thereafter was renamed "Programmable Intelligent Computer" [5].

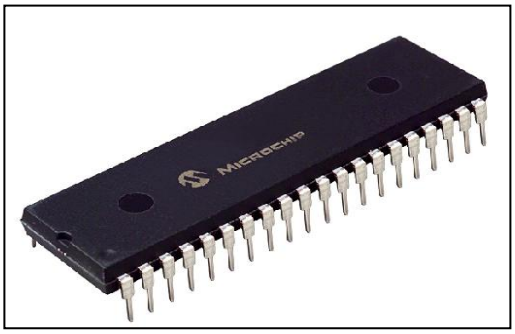

Figure 2.9: PIC

2.6.1 Origins

The original PIC was built to be used with GI's new 16-bit CPU, the CP1600. While generally a good CPU, the CP1600 had poor I/O performance, and the 8-bit PIC was developed in 1975 to improve performance of the overall system by offloading I/O tasks from the CPU. The PIC used simple microcode stored in ROM to perform its tasks, and although the term wasn't used at the time, it shares some common features with RISC designs.

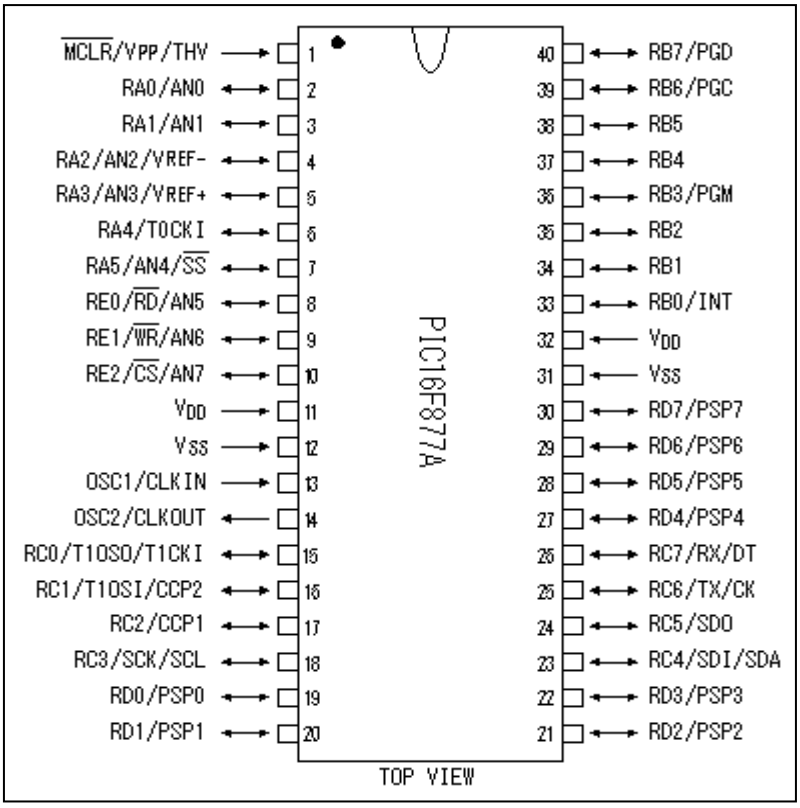

Figure 2.10: PIC16F877/877A pin

In 1985 General Instruments spun off their microelectronics division, and the new ownership cancelled almost everything — which by this time was mostly out-of-date. The PIC, however, was upgraded with EPROM to produce a programmable

channel controller, and today a huge variety of PICs are available with various on-board peripherals (serial communication modules, UARTs, motor control kernels, etc.) and program memory from 512 words to 64k words and more (a "word" is one assembly language instruction, varying from 12, 14 or 16 bits depending on the specific PIC micro family).

Microchip Technology does not use PIC as an acronym; in fact the brand name is PICmicro. It is generally regarded that PIC stands for Peripheral Interface Controller, although General Instruments' original acronym for the initial PIC1640 and PIC1650 devices was "Programmable Interface Controller". The acronym was quickly replaced with "Programmable Intelligent Computer".

The Microchip 16C84 (PIC16x84), introduced in 1993[6] was the first CPU with on-board EEPROM memory. This electrically-erasable memory made it cost less than CPUs that required a quartz "erase window" for erasing EPROM.[5]

2.6.2 PIC Microcontroller Option

A microcontroller (also MCU or µC) is a functional computer system-on-a-chip. It contains a processor core, memory, and programmable input/output peripherals. While the PIC controller chips are the combination the function of microprocessor, ROM program memory, same RAM memory and input-output interface in one single package which is economical and easy to use.

The PIC – Logicator system is design to be used to program a range of 8, 18, 28 pin reprogrammable PIC microcontroller which provide a variety of input-output, digital input and analogue input options to suit students project uses [11].

Reprogrammable “FLASH Memory” chips have been selected as the most economical for student use. If a student needs to amend to control system as the project is evaluated and developed, the chip can simply be taken out of the product and reprogrammed with an edited version of the floe sheet [11].

The PIC devices generally feature is sleep mode (power savings), watchdog timer, various crystal or RC oscillator configurations, or an external clock [5].

2.6.3 Variants

Within a series, there are still many device variants depending on what hardware resources the chip features.

- General purpose I/O pins.
- Internal clock oscillators.
- 8/16 Bit Timers.
- Internal EEPROM Memory.
- Synchronous/Asynchronous Serial Interface USART.
- MSSP Peripheral for I²C and SPI Communications.
- Capture/Compare and PWM modules.
- Analog-to-digital converters (up to ~50 kHz).
- USB, Ethernet, CAN interfacing support.
- External memory interface.
- Integrated analog RF front ends (PIC16F639, and rfPIC).
- KEELOQ Rolling code encryption peripheral (encode/decode)
- And many more.

2.6.4 PIC Basic Pro Compiler

The PICBASIC PRO™ Compiler (or PBP) is the easiest way to program the fast and powerful Microchip Technology PICmicro microcontrollers (MCUs). PICBASIC PRO converts the BASIC programs into files that can be programmed directly into PICmicro microcontrollers (MCUs). The English-like BASIC language is much easier to read and write the quirky Microchip assembly language (likes machine language and assembly language).

The PicBasic Pro Compiler instruction set is upward compatible with the BASIC Stamp II and Pro uses BS2 syntax. Programs can be compiled and programmed directly into a PICmicro MCU, eliminating the need for a BASIC Stamp module. These programs execute much faster and may be longer than their Stamp equivalents. They may also be protected so no one can copy your code [4].

The PicBasic Pro Compiler also can create programs for any of Microchip's PICmicro microcontrollers and works with most PICmicro MCU programmers, including the elbas Serial Programmer. A printed manual and sample programs are included to get you started [3].

2.6.5 MAX232

The MAX232 is an integrated circuit that converts signals from an RS-232 serial port to signals suitable for use in TTL compatible digital logic circuits. The MAX232 is a dual driver/receiver and typically converts the RX, TX, CTS and RTS signals. The drivers provide RS-232 voltage level outputs (approx. ± 7.5 V) from a single + 5 V supply via on-chip charge pumps and external capacitors. This makes it useful for implementing RS-232 in devices that otherwise do not need any voltages outside the 0 V to + 5 V range, as power supply design does not need to be made more complicated just for driving the RS-232 in this case. The receivers reduce RS-232 inputs (which may be as high as ± 25 V), to standard 5 V TTL levels. These receivers have a typical threshold of 1.3 V, and a typical hysteresis of 0.5 V [18].

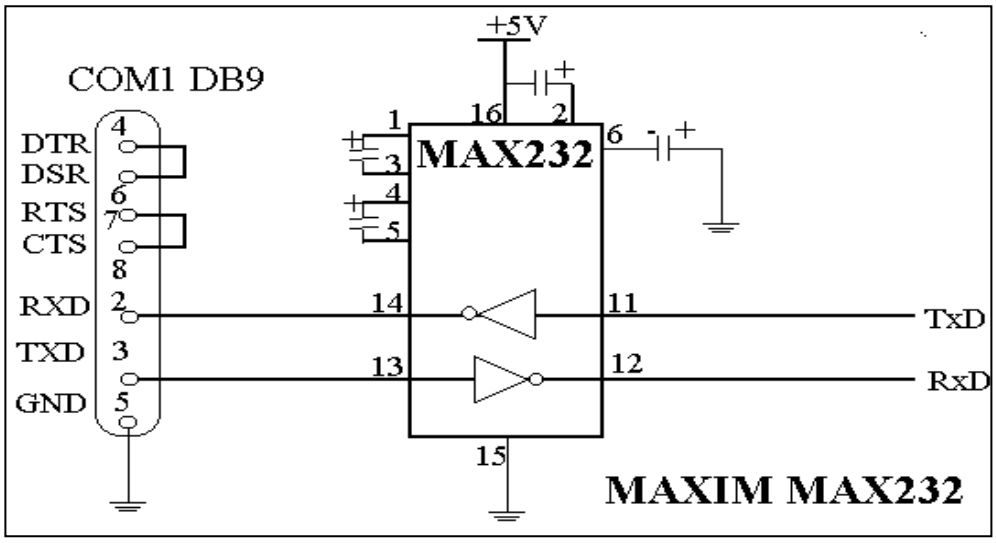

Figure 2.11: MAX232 connection to DB9

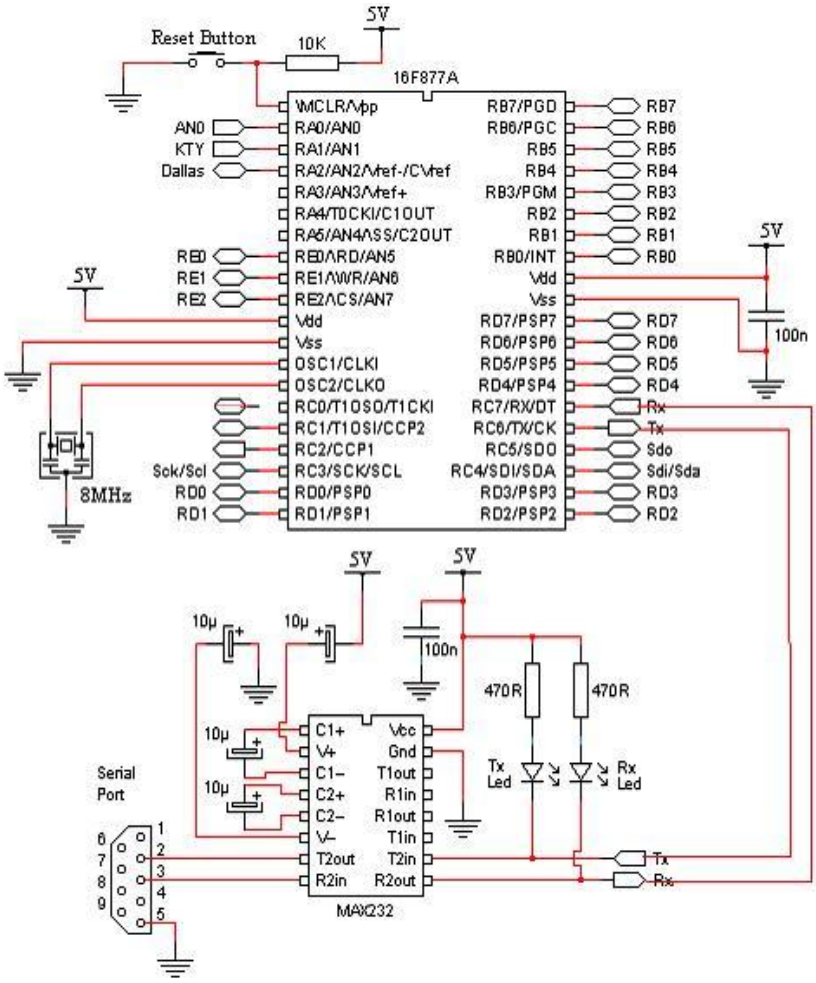

Figure 2.12: Serial connection with PIC

2.7 Implementing a PID Controller Using a PIC16 MCU

As the controller for the system, the microprocessor is chosen due to its simplicity in designing and also interfacing with other input or output devices. Below is the pin diagram of the microprocessor.

The microprocessor consists of 40 pin. At an economical price, with the addition of high endurance enhanced Flash program memory and a high speed 10-bit A/D converter. On top of these features, this family introduces design enhancements that

make these microcontrollers a logical choice for many high performance, power control and motor control applications. These special peripherals include 14-bit resolution Power Control PWM Module (PCPWM) with programmable dead time insertion Motion Feedback Module (MFM), including a 3-channel Input Capture (IC) Module and Quadrature Encoder Interface (QEI) High-speed 10-bit A/D Converter (HSADC) The PCPWM can generate up to eight complementary PWM outputs with dead-band time insertion. The MFM Quadrature Encoder Interface provides precise rotor position feedback and velocity measurement.

The PID routine is configured in a manner that makes it modular. It is intended to be plugged into an existing piece of firmware, where the PID routine is passed the 8-bit or 16-bit error value. Therefore, the actual error value is calculated outside of the PID routine. If necessary, the code could be easily modified to do this calculation within the PID routine. The PID can be configured to receive the error in one of two ways, either as a percentage with a range of 0 to 100% (8-bit), or a range of 0 to 4000 (16-bit). PID source code with the PID's variable declarations. The gains for proportional, integral and derivative all have a range of 0 to 15. For resolution purposes, the gains are scaled by a factor of 16 with an 8-bit maximum of 255. A general flow showing how the PID routine would be implemented in the main application code is presented in Figure 2. There were two methods considered for handling the signed numbers. The first method was to use signed math routines to handle all of the PID calculations. The second was to use unsigned math routines and maintain a sign bit in a status register. [7]

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter explains about what is the method that has been used to complete this project. It describes on how the project is organized and the flow of the steps in order to complete this project. The methodology is consoled of two parts, which is software and hardware. MATLAB is use for design the PID controller. In this part, this software is using for design the controller before implement it into microcontroller.

For hardware part, its start with design and construct the hardware for microcontroller unit, motor driver, encoder and serial communication. Then implement of PID controller based from the simulation result into the microcontroller. Serial communication using serial port is for real-time performance analysis.

3.2 Methodology

While doing the research and literature review for this project, relevant and important information can be obtained via surfing the internet, browsing books and journals and also with the assistance from supervisor in charge.

Several methods have to be implemented in order to ensure the success of this project. Experiment is one of the methods. By experimenting, the theories are supported and project work physically can be the evidence of the theory. Other than that, research is a good method as it can give some knowledge while doing this project. Through these researches, a lot of information can be collected and know which method will work and which will not.

In the first phase of this project, software analysis must be done to get the PID controller algorithm. Analysis had been done using MATLAB software. Analysis starts from find the mathematical model of the motor and gets the transfer function of the motor. Then design and tune PID controller using MATLAB analysis based on transfer function.

In the hardware phase of this project, the circuitry of the hardware must been done. The first stage is designing the circuitry such as, power supply circuit, DC motor driver circuit, microcontroller circuit and serial port circuit. While designing those circuits, a lot of research has been done. Basically those circuits are provided in the internet and books. Some adjustment must be done with the circuitry in order to make sure the objectives and scopes of this project are achieved. After designing the circuitry, the programs of the PIC have developed. PID algorithm must be put into the PIC. The program was written using the PICBasic Pro Compiler and it must be downloaded into the PIC.

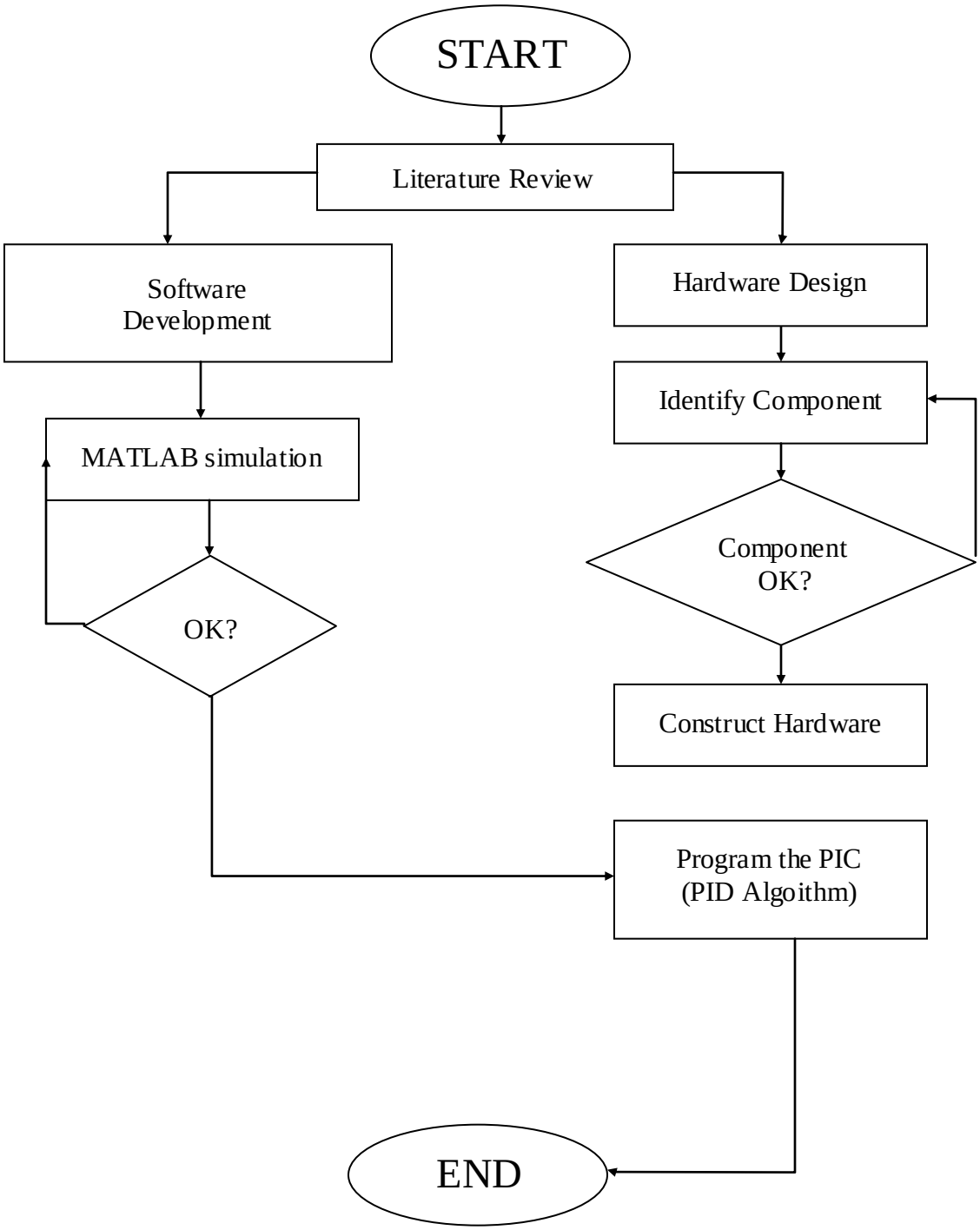

Figure 3.1: Flowchart of the project

After the software and hardware part had been done, the communication between personal computer (PC) and hardware had been create. This communication is for real-time performance analysis. Figure 3.1 shows the flow chart of the project.

3.3 Hardware part

In this system, the PID controller is designed using PIC microcontroller 16F877. This microcontroller provide motion feedback module that is useful in designing a close loop control system. Furthermore the microcontroller also provide up to 4 PWM channels that allow the user to control more motor. To provide feedback to the microcontroller, a quadrature encoder is used. The quadrature encoder will provide the actual speed references to allow the microcontroller to calculate the error.

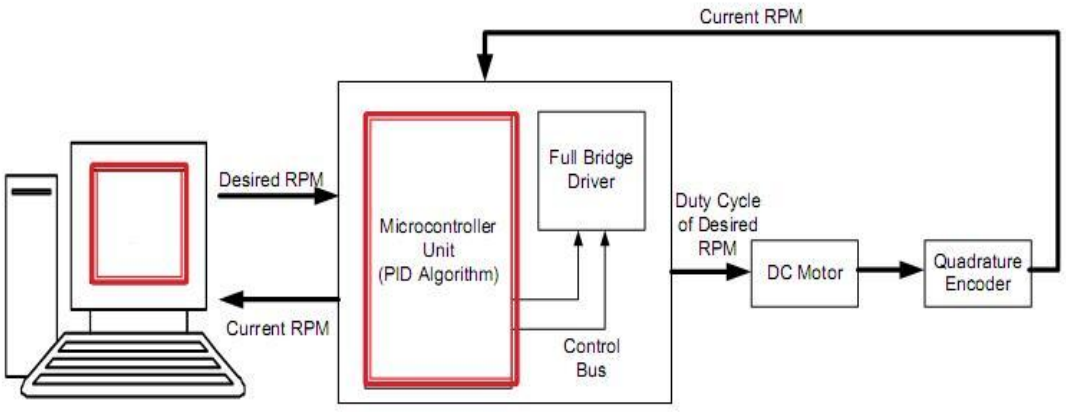

Figure 3.2: Hardware Design

From the figure above, it should not be confused that the computer does not work as a controller, instead it just a monitoring device that allow the user to monitor the performance. The computer is connected to the microcontroller using RS-232 serial data communication. The motor driver works as an actuator in providing the desired duty cycle to the motor.

3.3.1 Hardware Installation

For the hardware design, the first circuit design is power supply circuit. It is to supply the input voltage to the project; the supply output voltage must fix to 5V to support the PIC, temperature sensor and MAX232 IC. To achieve the 5V output voltage the 7805 voltage regulator IC had used in this project, it also to make input supply larger than the output voltage. This module is important to this project because it can prevent damage to the PIC and MAX232 IC if users give the higher input supply to device. The schematic diagram for power supply module is like in Figure 3.2.

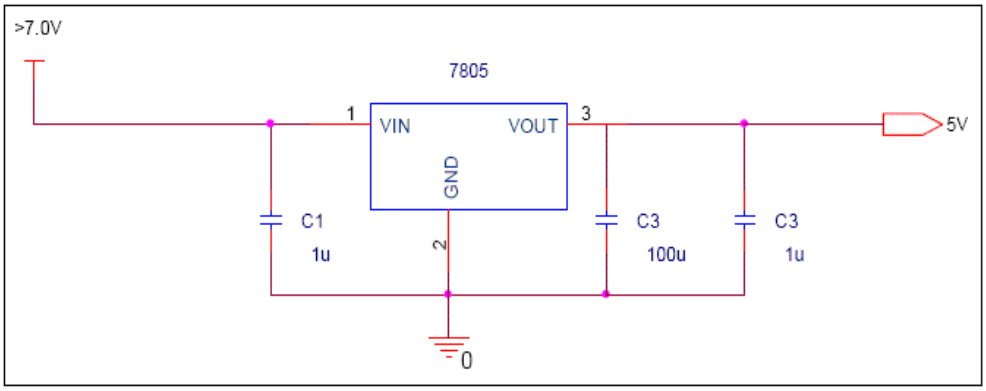

Figure 3.3: Power supply modules

The second part circuit design is the connection from the communication port, it is the DB9 connection from the personal computer (PC) to the device. The DB9 pin is shown in Table 3.1 below and the figure of RS232 communication port is shown in Figure 3.4. The connection from personal computer (PC) to device is only on pin 2, 3 and pin 5. So in this project only three pin are used for connection in the serial port communication; one for the signal ground, one for transmitting data and one for receiving data. The connection between the RS232 with MAX232 and the PIC circuit has shown in Figure 3.4.

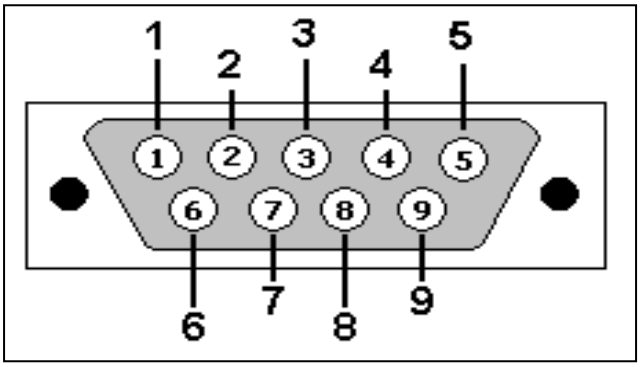

Figure 3.4: Pins and signal associated with the 9-pin connector

Table 3.1: Serial port pins and signal assignments

Pin No.	Name	Dir	Notes/Description
1	DCD	IN	Data Carrier Detect. Raised by DCE when modem synchronized.
2	RD	IN	Receive Data (a.k.a RxD, Rx). Arriving data from DCE.
3	TD	OUT	Transmit Data (a.k.a TxD, Tx). Sending data from DTE.
4	DTR	OUT	Data Terminal Ready. Raised by DTE when powered on. In auto-answer mode raised only when RI arrives from DCE.
5	SGND	-	Ground
6	DSR	IN	Data Set Ready. Raised by DCE to indicate ready.
7	RTS	OUT	Request To Send. Raised by DTE when it wishes to send. Expects CTS from DCE.
8	CTS	IN	Clear To Send. Raised by DCE in response to RTS from DTE.
9	RI	IN	Ring Indicator. Set when incoming ring detected - used for auto-answer application. DTE raised DTR to answer.

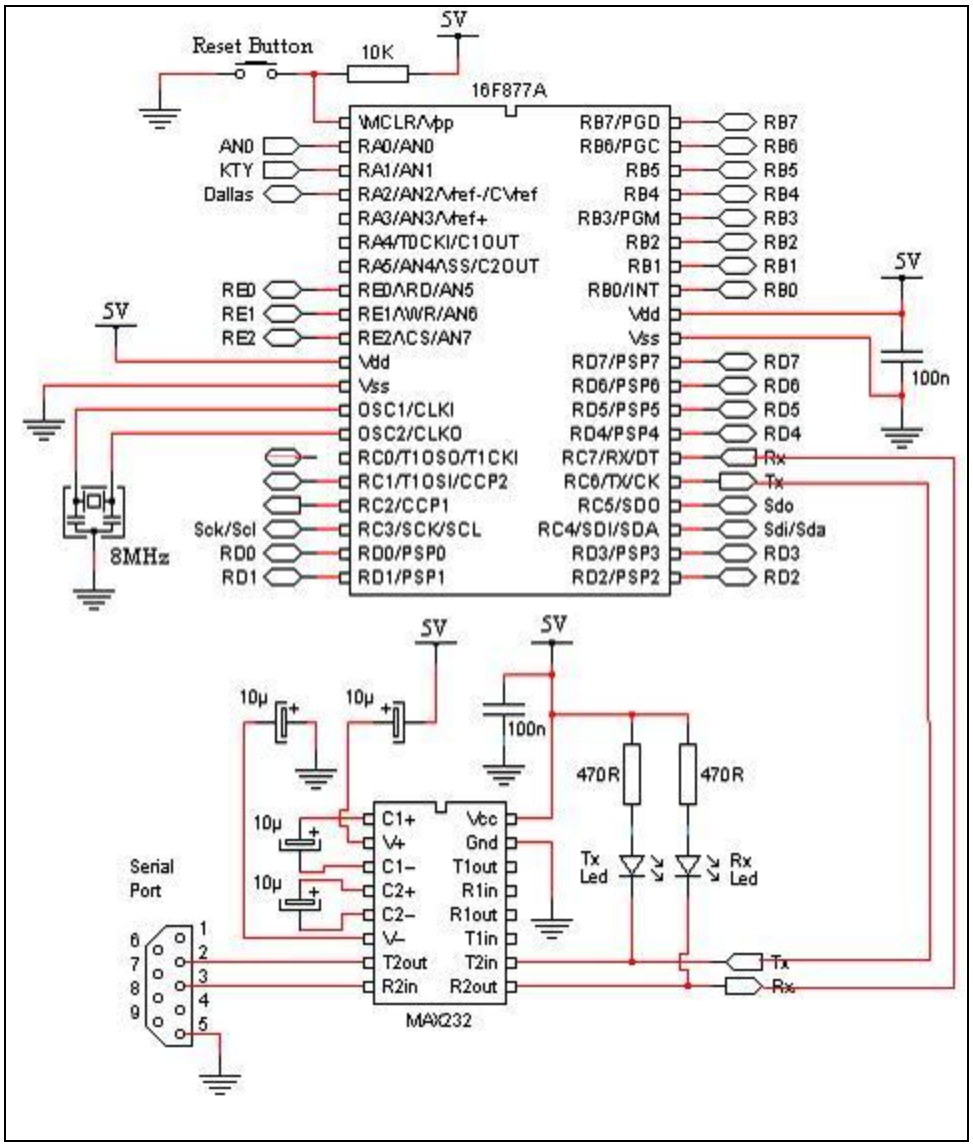

Figure 3.5: Serial port connection to PIC

In this project the output data from MAX232 is send directly to PIC at PORTC.7 and the input data from MAX232 is received directly to PIC at PORTC.6. The connection is depending on the PIC programming that has been developed. The value of oscillator use in the circuit diagram also must be same with the define one in the PIC programming to avoid instability.

The third circuit design is driver circuit. This circuit directly connects to PIC at PORTC.1 for PWM, PORTB.0 and PORTB.1 for control the DC motor run in forward and reverse direction. This driver circuit will injected with PWM from the PIC as input. The driver circuit connection has shown in Figure 3.5.

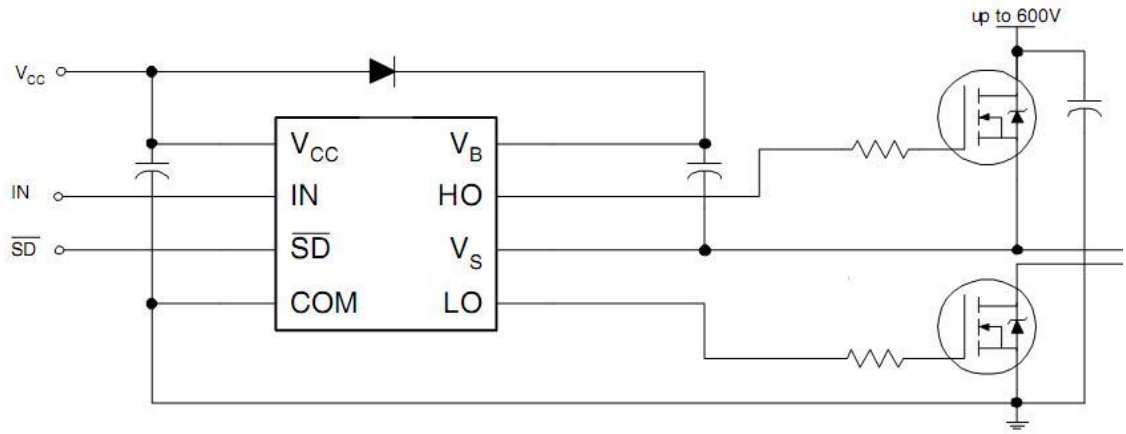

Figure 3.6: Driver circuit using IR 2109

3.4 Encoder configuration

The feedback module consists of a quadrature encoder and a flexible coupling. The quadrature encoder will enable the system to acquire the feedback and later performing the required operations to effectively use the information coming from the encoder. The two quadrature encoder output signals channel A and channel B. The position counter can be used either for position or speed measurement. To measure motor position, we must know the relationship between the displacement and the number of phase pulses we get from the encoder. This relation can be known in advance, or can be measured during initialization by accumulating the total count for the maximum allowed displacement by using the formula below.

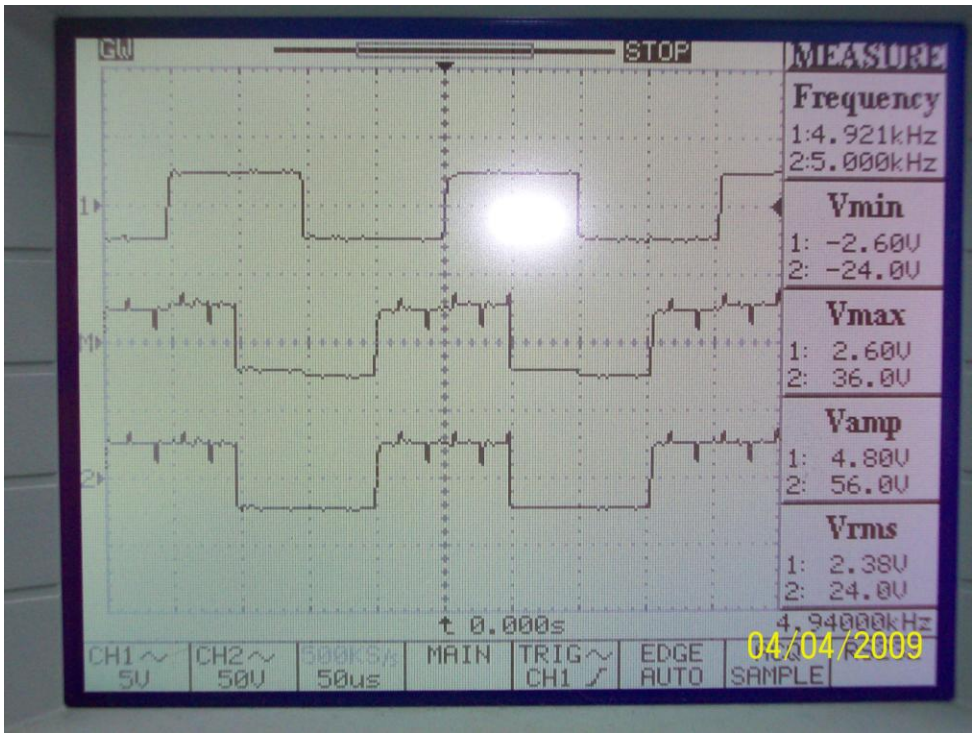

Figure 3.7 Sample of Output from Encoder

To calculate the angular velocity under a fixed time interval, the value of encoder pulses and pulses per revolution of the encoder need to be known. Mathematically, it can be derived from the formula below. By running the motor at full speed and record the number of pulses under 1 second sampling time, the value of pulses per revolution can be obtained. Above is the sample of output from the encoder.

$$Velocity = \frac{\frac{Encoder\ Pulse}{Pulse\ Per\ Revolution} \times \frac{60sec}{1min}}{Time\ Interval(sec)}\ (RPM)$$

Formula 1 : Velocity formula(RPM)

3.4.1 Pulse Width Modulation

Pulse-width modulation (PWM) or duty-cycle variation methods are commonly used in speed control of DC motors. The duty cycle is defined as the percentage of digital 'high' to digital 'low' plus digital 'high' pulse-width during a PWM period.

The average DC voltage value for 0% duty cycle is zero; with 25% duty cycle the average value is 7.5V (25% of 30V). With 50% duty cycle the average value is 15V, and if the duty cycle is 75%, the average voltage is 22.5V and so on. The maximum duty cycle can be 100%, which is equivalent to a DC waveform. Thus by varying the pulse-width, we can vary the average voltage across a DC motor and hence its speed.

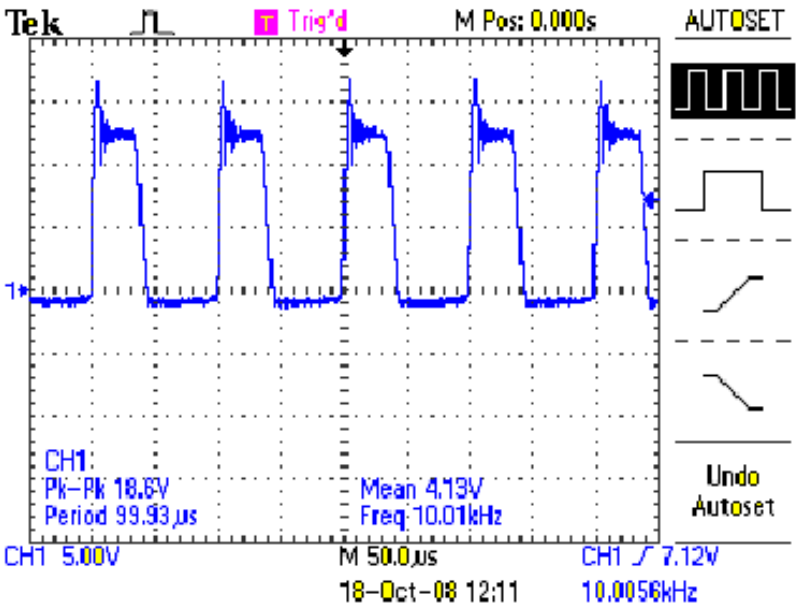

Figure 3.8 : Sample of a PWM Waveform

.

3.5 PID Algorithms

PID algorithms consist of three parameters which are Proportional, Integral and Derivative terms. All this three terms later on added to create an output which will be inserted into the plant or in this case the motor. Previously, it is known that the motor run on a generated PWM from the microcontroller and this PWM waveform is controlled through its duty cycle. The duty cycle plays an important role in the whole system. The PID output itself will be inserted together with the duty cycle to create an adjustment so that the motor can be brought back to its desired speed. To do this, the system need to know the rated speed, or the maximum speed the motor can handle. This rated speed value is used with the desired speed value so that it can be converted into a duty cycle.

Duty Cycle = (Desired Speed / Rated Speed) x 255

This formula is converted into a PICBasic language as below:

```
Duty = Setpoint * 255
Duty = Duty/Rated Speed
HPWM 2,Duty,10000
```

Once the motor already run on the desired speed, it will be left to the PID algorithm to correct its speed by adjusting the output of PID into the duty cycle.

In opening and closing the communication port the command fclose (SerPIC) is use to disconnect a serial port object from the device. The baud rate from MATLAB must be set same with the baud rate in PIC before it can transmit and receive the data. For example if baud rate in MATLAB is 9600bps, so the baud rate in PIC also 9600bps.

3.5.1 Error Calculation

The error calculation is basically the difference between the desired speed or setpoint and the actual speed of the motor. The actual speed from encoder will be used as Current speed.In the programming, a subroutine is used to calculate the error.

```
Error = Setpoint - actual RPM
```

However, under certain condition, the actual speed might be bigger than the set point. Thus, the controller must be made to ensure that it know the sign of the error. This can be done by checking the highest bit of the error variable.

```
Error = ABS error
IF Error.15 = 1 THEN error = -error
```

Furthermore, it is important to use the absolute value off error in all the calculations later on.

3.5.2 Proportional Terms

Proportional parameter is simply the multiplication between the proportional gain, Kp with the Error.

```
P = Error * Kp
IF Error.15 = 1 THEN P=-P
```

3.5.3 Integral Terms

Unlike proportional control, which looks at the present error, integral control looks at past errors. This is the accumulative error (sum of all past errors) which is used to calculate the integral term, but at fixed time intervals. By using a program, the program simply records the value of E at fixed time interval of T (sampling time). Since Integral terms looks at past errors, the new integral term is obtained by adding the old integral term with accumulated errors which has been multiplied by the integral gain. However, to prevent integral windup, a limit must be used for calculating the accumulated errors to avoid the accumulated errors to keep on adding.

```
IF Err.15 =1 THEN Err_2 = -Err_2
Ei = Ei + Err 2
Sign = Ei.15
Ei_2 = (ABS Ei) * Ki
IF Sign = 1 THEN Ei_2 = -Ei_2
I = I + Ei_2
Sign = I.15
I = ABS I
I = I MIN 100
IF Sign = 1 THEN I = -I
```

In the code snippet above, the current error, Ei_2 will be added to accumulated error, Ei. The accumulated error is then multiplied by Ki. Later on the result of the multiplication is added with the I terms.

3.5.4 Derivative Terms

The derivative term works on the present errors to forecast a future response of the system. The derivative term makes an adjustment based on the rate at which the Plant output is changing from its Setpoint. A notable characteristic in this type of control is when the error is constant, or at the maximum limit, the effect is minimal. To get the derivative term, the previous error is subtracted from the current error and multiplied by the derivative gain, Kd. Then the program must save the current error so that at next time, it will be the old error.

```
D = (Error-LastError)
D = ABS D * Kd
IF D.15 = 1 THEN D = -D
LastError = Error
IF Error.15 =1 THEN LastError = -LastError
```

3.5.5 PID Output

The PID output is calculated after the proportional, integral and derivative terms have been determined. It is done by adding the current motor duty cycle with the PID. This result will later be inserted in the duty cycle variable of the motor.

```
Duty = actual RPM * 255/Maximum Motor RPM
Duty = Duty + (P + I + D)
Duty = Duty MIN 255
HPWM 2,Duty,10000
```

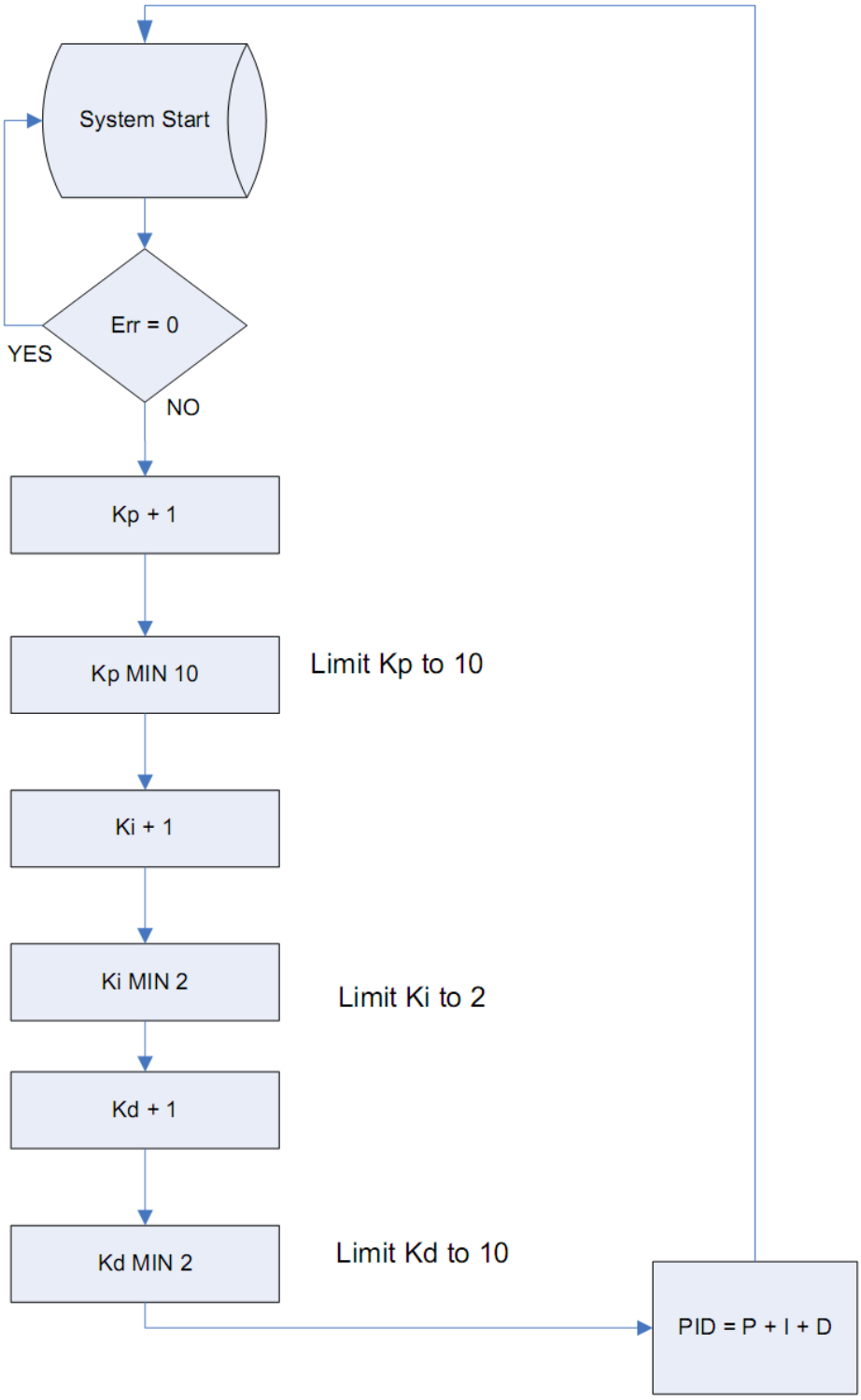

Figure 3.9 Flowchart of the Adaptive PID Algorithm Implemented In the MCU

3.6 Build PIC Programming

There are many way to program the PIC either the user can use LDmicro, PICBasic, PICBasic Pro or assembly language. The LDmicro use ladder diagram approach like PLC while PICBasic Pro Compiler is English-like BASIC language and mush easier to read and write than the quirky Microchip assembly language. Both PICBasic and PICBasic Pro language are very similar to the standard BASIC language but they have some modified and some additional instruction specifically for microcontroller programming. In this project the PICBasic Pro Compiler is used to create the programming. The Table 3.5 is shown the comparison of PICBasic and PICBasic Pro language.

Table 3.2: Comparison of PICBasic and PICBasic Pro

PICBasic	PICBasic Pro
Low cost	Higher cost
Limited to first 2K of program space	No program space limit
Interrupt service routine in assembly language	Interrupt service routine can be in assembly language or in PICBasic Pro
Peek and Poke used to access register	Register can be accessed directly by specifying
Some commands can be used only for PORTB, PORTC or GPIO	Commands can be used fir all ports
Clock speed 4MHz	Any clock speed up to 20MHz
Most 14 bit PIC microcontroller supported	All PIC microcontroller, including 12 bit ones are supported
More code space in memory	5-10% less code space in memory
More difficult to learn and less powerful	Easier to learn and more powerful
No LCD commands	Special LCD control commands (LCDOUT, LCDIN)
No hardware serial communication commands	Special hardware serial communication commands (HSERIN, HSEROUT)
No PWM commands	Special PWM commands for the microcontrollers that have built in PWM circuit (HPWM)
No Select-Case command	Select-Case command for multi-way

	selection
No program memory read-write commands	Commands to read and write program memory locations (READCODE, WRITECODE)
No One-wire device interface	One-wire device interface commands (OWIN, OWOUT)
No USB commands	USB commands for microcontroller that have built in USB circuit (USBIN, USBOUT)
No X-10 remote control commands	X-10 remote control commands (XIN, XOUT)
No A/D commands	A/D commands for microcontrollers that have built in A/D converter (ADCIN)

The data from PIC in decimal form is send to MATLAB, so the MATLAB is program need to read the data also in decimal form. The communication between MATLAB and PIC is in standard asynchronous format where the device uses its own internal clock resulting in bytes that are transferred at arbitrary times. The baud rate is specifying according to MATLAB. Some standard baud rates are listed in Table 3.6. For PIC programming, 9600bps is used which same with the MATLAB.

The input data at PIC that transmit from MATLAB GUI is set to PORTC.0 before it run certain program to control the DC motor. Here is the example to program the stepper motor run in clockwise and anticlockwise direction. If PIC sends data ‘001’, so the MATLAB will perform case 001 according the programming.

Table 3.3: List of standard baud rate

Baud Rate	Bits 0 - 12
300	3313
600	1646
1200	813
2400	396
4800	188
9600*	84
19200*	32

To program the PIC, make sure the oscillator that has been defined in programming is similar to the hardware in order to avoid instability during the transmitting and receiving data. The SERIN2 command in the program support many different data modifier which may be mixed and matches freely within single SERIN2 statement to provide various input formatting. The modifier support is shown in Table 3.7. The number 84 on "Serin2 SerI, 84, [dec3 B0]" command is refer to baud rate that equal to 9600bps according Table 3.6.

Table 3.4: Modifier support by SERIN2 command

Modifier	Operation
BIN{1..16}	Receive binary digits
DEC{1..5}	Receive decimal digits
HEX{1..4}	Receive upper case hexadecimal digits
SKIP n	Skip n received characters
STR ArrayVar\n{\c}	Receive string of n characters optionally ended in character c
WAIT ( )	Wait for sequence of characters
WAITSTR ArrayVar{\n}	Wait for character string

CHAPTER 4

RESULT AND DISCUSSION

4.1 Introduction

This chapter will discuss all the result obtained and the limitation of this project. All discussions concentrate on the result and performance from simulation in MATLAB. The discussions based on the simulation results from the MATLAB analysis by using MATLAB SIMULINK.

4.2 Simulation in MATLAB

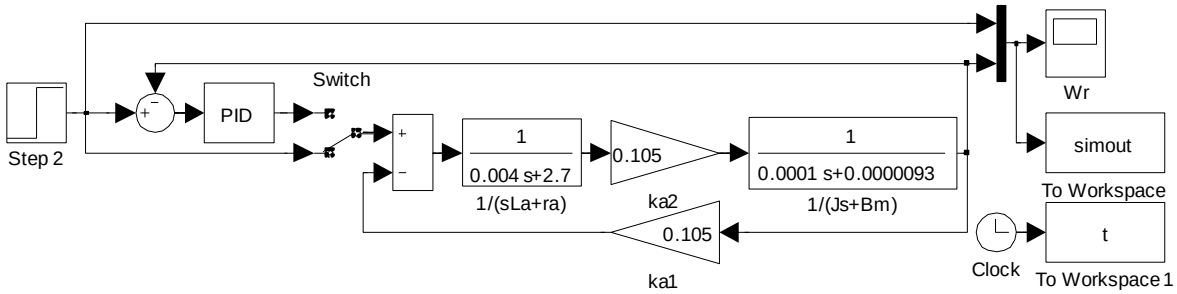

Figure 4.1: Block diagram for simulation in MATLAB

The parameter for step time is 10, then for the initial value is 3. The final value and sample time is 10 and 0.5. The parameter for all proportional, integral and derivative is defined by using try and error method. To see the response more clearly, the result taken is not just the PID mode but also proportional mode, proportional + integral mode and proportional + derivative mode. From all this response the analysis will became easier since we can see which response is better.

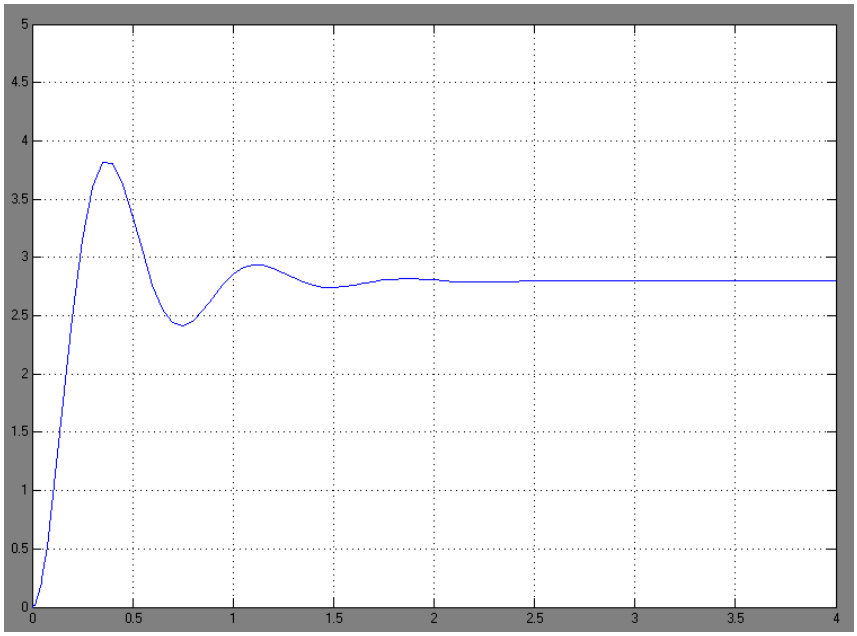

Figure 4.2: Response in MATLAB – Proportional mode

Figure 4.1 shows the step response if only the P (Proportional) controller is applied in the SIMULINK in Figure 3.3. A proportional controller allows tighter control of the process variable because its output can take any value between fully on and fully off, depending on the magnitude of the error signal [16]. With proportional band, the controller output is proportional to the error or a change in measurement and offset (deviation from set-point) is present. This steady state error is the difference between the attained value of the controller and the required value. In this SIMULINK, the gain for P is 110 and the set point is 3. As we can see, there is overshoot at 3.8 and it over the

target value which is 3. In the end the system doesn't settle out any quicker than it would have with lower gain, but there is more overshoot. If we kept increasing the gain we would eventually reach a point where the system just oscillated around the target and never settled out-the system would be unstable.

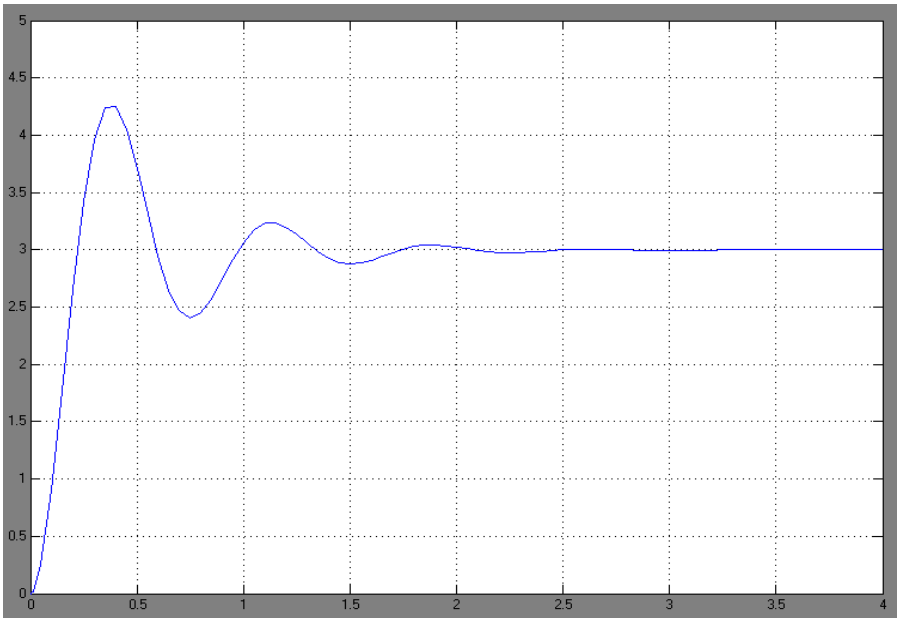

Figure 4.3: Response in MATLAB – Proportional + Integral mode

Figure 4.2 above shows when the P (Proportional) and I (Integral) gain is applied in the SIMULINK. To eliminate the offset error, the controller needs to change its output until the process variable error is zero. Integral action gives the controller a large gain at low frequencies that results in eliminating offset and "beating down" load disturbances. It also can reduce the final error in a system. Summing even a small error over time produces a drive signal large enough to move the system toward a smaller error. In this system, the gain for Integral is 20. In this graph we can see that the steady state error still occur but it is now at the set point input which is 3. However the overshoot became greater than before which is at 4.3. The rise time is also faster than just using the proportional controller.

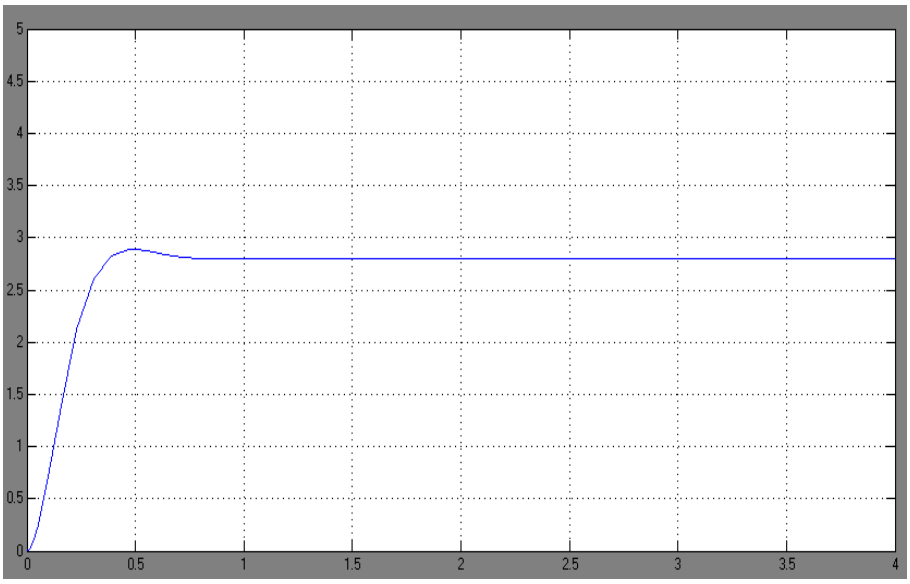

Figure 4.4: Response in MATLAB – Proportional + Derivative mode

In Figure 4.3 we can see that the response became better than the PI controller. This SIMULINK is actually the Proportional plus Derivative controller. The gain for Proportional is 110 and for the Derivative is 10. Derivative action can stabilize loops since it adds phase lead and it act as an anticipator. That is why the settling time now is near the set point and the steady state error became lesser than the PI controller.

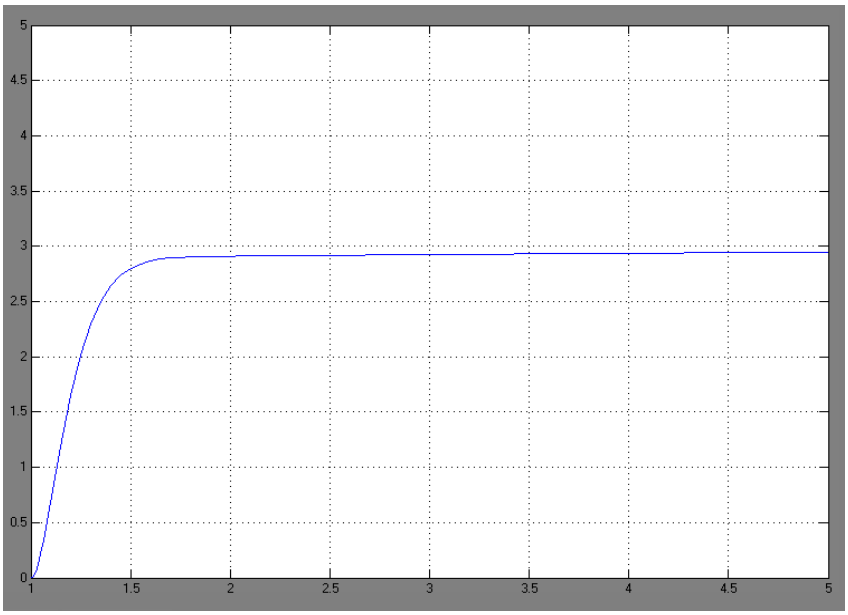

Figure 4.5: Response in MATLAB – PID mode

Lastly, the Figure 4.4 show the step response if the entire element in PID controller is applied in the system. As the proportional gain is increased, the controller responds faster. If the proportional gain is too high, the controller may become unstable and oscillate. The integral gain acts as a stabilizer. It provides power even if the error is zero. The Derivative controller counteracts the KP and KI terms when the output changes quickly. This helps reduce overshoot and ringing and it has no effect on final error. In this SIMULINK, the gain for Proportional is 100, Integral is 20 and the gain for Derivative is 15. The step response for this system became smooth and the response is better than if we use P, PI, and PD. There is no overshoot; the rise time is faster and less error. The settling time is exactly at the set point which is 3.

4.3 Discussion

Before starting a motion-control project, the system's requirements are need to determine as a whole factor in the cost of both the motor and the control system. Also, consider the own comfort level with more complex technologies. In this project, the technology would be the PID controller and PIC. It is possible to mathematically calculate the PID coefficients and accurately predict the machine performance because of the PID tuning. However, in order to do this the transfer function of the machine being controlled must be accurately mathematically modeled.

In simulation using MATLAB the modeling for motor is actually from reference book. This is because, there is no information or specifications given from the supplier about this dc motor. Even though the response from this simulation can be used to determine which mode has a better response and the parameter from this simulation can also be applied into the real system.

For programming in PIC, the first problem that occurred is at the PID algorithm into the PIC. A few weeks is taken to inserted PID algorithm in the program. After joining the PIC forum and using try and error method, this project successfully done.

Another limitation in this project is failed to get interface between hardware and software (MATLAB). This is necessary for get the real-time performance analysis of the implemented control system. If the interfacing the hardware with the MATLAB successful, some improvement can be achieve. As we know MATLAB is language of technical computing. By using MATLAB, we are easy to make an analysis. If the interfacing successful, this hardware can be replaced the DAQ card.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

The design and implementation of DC motor application in MATLAB GUI has been presented in this project. The DC motor application is able to control by using GUI via serial port communication. The development of the hardware and PIC16F877A programming using Microcode Studio was done after detail studies and analysis. Through of the development of this project it has concluded that the PID controller can be implementing using PIC for the DC motor

5.2 Future Recommendation

For future recommendations, in order to improve this project, other features such as GUI control can be added to control the motor speed. Besides that, the other type of motor such as AC motor can be added to be controlled through MATLAB GUI. For the interface part, the RS232 cable can be improved by replacing it using universal serial bus (USB), infrared, Bluetooth or wireless communication. By using universal serial bus (USB) infrared, Bluetooth or wireless communication the system can be control without cable and it make the system free to be used anywhere. Furthermore there are many things that we can develop such as rotation or speed of the motor that can be measured in MATLAB GUI.

Furthermore, with its simple and user friendly control panel the PID Controller can meet the demand of such controller in the industry. By coming with its own Control Panel, the PID Controller can be practically used on any system that needs the performance of its DC Motor to be monitored and analyzed

5.3 Costing and Commercialization

The design of the PID Controller comes with the approach of simplicity and also user friendly. The controller uses PIC 16F877 from the 16F family series of Microchip's PIC which already widely used in application regarding motor control. By using Printed Circuit Board (PCB) and also Surface-Mounted technology (SMT), it gives better performance under shake and vibration conditions and also robust design by its protective plastic container. Their small and lightweight designs also allow the controller to be more mobile while offering less work space it its usage. The optional RS-232 interface feature also can provide full remote operation of the controller. For commercialization purpose, the price of the unit is created based on the

equipment or component used in the controller. This price list of components used in important if the product need to be commercialized in the future. The complete list of price is as below.

Table 5.1: Approximation cost of component

No	Components	Specifications	Price / unit	Quantity	Estimation Cost
1	PIC	16F877	RM 25.00	1	RM 25.00
2	IC base	40 pin	RM 0.20	1	RM 0.20
3	MAX	232	RM 4.00	1	RM 4.00
4	DB9		RM 0.60	1	RM 0.60
5	Heat sink		RM 0.70	4	RM 0.70
6	Strip board	10" X 4"	RM 5.00	1	RM 5.00
7	Wire wrap		RM 40.00	1	RM 40.00
8	Regulator	7805	RM 2.00	1	RM 2.00
9	Capacitor	1uF	RM 0.24	2	RM 0.48
10	Capacitor	100uF	RM 0.48	3	RM 1.44
11	Ribbon cable		RM 2.00	1	RM 2.00
12	Capacitor	22pF	RM 0.08	3	RM 0.24
13	Reset switch		RM 0.60	1	RM 0.60
14	IC base	(16 pin)	RM 0.16	1	RM 0.16
15	Crystal	20MHz	RM 1.90	1	RM 1.90
16	Capacitor	100nF	RM 0.40	2	RM 0.40
17	Capacitor	10uF	RM 0.48	4	RM 1.92
18	Resistor	470Ω	RM 0.04	2	RM 0.08
19	Resistor	10KΩ	RM 0.04	1	RM 0.04
20	Resistor	3.3KΩ	RM 0.04	2	RM 0.08
21	Resistor	22Ω	RM 0.04	5	RM 0.20
22	Resistor	75Ω	RM 0.04	1	RM 0.04
23	Diode	N4002	RM0.10	2	RM 0.20
24	PCB header	40 ways	RM 0.80	10	RM 8.00
25	Diod	IN4007	RM0.15	4	RM0.15
26	Mosfet	IRF740	RM3.00	4	RM12.00
27	IC mosfet driver	IR2109	RM5.00	2	RM10.00
28	Zener Diode	4148	RM0.15	4	RM0.15
29	LED		RM 0.05	2	RM 0.10
30	DC motor	Litton Clifton	Provided	1	Provided
TOTAL					**RM 119.72**

The price above really meets the low-cost approach in the design of the controller. It will allow the controller to be sold at a cheaper price comparing to most of the PID Controller in the market. Furthermore, since the controller application is computer based, it should have a significant advantage over other PID controller in the market that relies on any other approach. This computer based application also allows the performance analysis to be made easily.

REFERENCES

(i) Articles in internet

[1] http://en.wikipedia.org/wiki/Brushless_DC_electric_motor. (February 18, 2008)

[2] http://en.wikipedia.org/wiki/Neural_network. (February 18, 2008)

[3] http://ai-depot.com/articles/evolutionary-PID-Controller (February 18, 2008)

[4] http://en.wikipedia.org/wiki/PID (February 18, 2008)

[5] http://www.imagesco.com/microcontroller/picbasic-pro-compiler.html(24 February 2009)

[6] http://www.alldatasheet.com (July 29, 2008)

[7] http://www.microchip.com (Mac 2008)

(ii) Books

[7] William Palm III (2005), "*Introduction to MATLAB 7 for Engineers*", McGraw Hill.

[8] Sergey E. Lyshevski (2000), Electromechanical System, Electric Machines, and Applied Mechatronics, CRC Press LCC.

(iii) Software

[9] Matlab 7.1 software, 'PID Toolbox', Mathworks

APPENDIX

Snapshots of the Hardware with the DC Motor