

**Speed Control for Linear Vehicle Following Predetermined Path**

by

**RAHIMI ZAMAN BIN ABD RASHID**

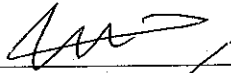
Dissertation submitted in partial fulfilment of  
the requirements for the  
Bachelor of Engineering (Hons)  
(Electrical & Electronics Engineering)

JUNE 2007

Universiti Teknologi PETRONAS  
Bandar Seri Iskandar  
31750 Tronoh  
Perak Darul Ridzuan

## CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



---

**RAHIMI ZAMAN BIN ABD RASHID**

## **ABSTRACT**

The objective of the project is to develop a controller that could control both the speed and direction of a linear vehicle following a predetermined path with the smallest error as much as possible. The project is a continuation of the previous project which concentrated on the neural network controller for steering control for defined car model. Several input paths developed in the previous projects are employed to the control system in order to verify the functionality of the controller to follow the path as close as possible and as well as for comparison purposes. The previous projects showed that the controller designed could lead the vehicle to follow the path, but the percentage error could be further improved. Due to that reason, speed control function is added to the project to improve the efficiency of the controller. MATLAB has been chosen as the platform for the simulation design and result verification. PID controller and fuzzy logic are chosen as the methods for developing the controller for the system. The controller is expected to be able to monitor and control the speed and direction of the vehicle in which being represented by a DC motor. The project is then extended in real life application through the implementation of xPC TargetBox. To accomplish the objective of the project, several tracks of methodologies are carried out during the time frame given. Several tools are required to be used through the project for testing and execution processes. The conclusion of this project would be subjected on how precise the linear vehicle follows the prescribed paths while varying the speed accordingly.

## **ACKNOWLEDGEMENT**

First of all, deepest gratitude to the Lord of the Universe, Allah SWT for His bless in various aspect to accomplish the final year project entitled “Speed Control of a Linear Vehicle Following Predetermined path”. Nothing will be happened without His permission in accomplishing this final year project.

Secondly, thousands of appreciations are dedicated to the supervisor for giving the opportunity and trust to handle the project. A big thank also goes to Mr. Rafiq, tutor for his support, idea and guidance which are really helpful throughout the semester. Acknowledgment is also given to all the technicians of EE Department especially Mdm. Siti Hawa, Mr. Azhar, Mr. Yasin and Miss Yanti for their assistance during the completion of this project

Finally, utmost appreciations are dedicated to the beloved parents for their commitment and support. They were a big supporter behind all scenes, are the best source of inspiration ever had.

## TABLE OF CONTENTS

ABSTRACT .....	iv
ACKNOWLEDGEMENT .....	v
TABLE OF CONTENTS .....	vi
LIST OF FIGURES .....	viii
CHAPTER 1 .....	1
INTRODUCTION .....	1
1.0 Background of Study .....	1
1.2 Problem Statement.....	2
1.3 Objective and Scope of Study .....	3
CHAPTER 2 .....	4
LITERATURE REVIEW .....	4
2.1 Brief Outline on Previous Works .....	4
2.2 Fuzzy Logic .....	5
2.2.1 Background of Fuzzy Logic .....	5
2.2.2 Fuzzy Inference System (FIS) .....	6
2.2.3 Fuzzification of the input variables .....	7
2.2.4 Application of the fuzzy operator.....	7
2.2.5 Apply Implication Method .....	8
2.2.6 Aggregate all outputs.....	9
2.2.7 Defuzzify .....	10
2.3 PID controller .....	11
2.4 Overview on DC motor .....	11
2.5 MATLAB Based software.....	13
2.5.1 Simulink .....	13
2.5.2 Real-Time workshop .....	14
2.5.3 xPC Target.....	16
2.5.4 xPC TargetBox .....	18
CHAPTER 3 .....	22
METHODOLOGY .....	22
3.1 Final Year Project 1 methodology.....	22
3.1.1 Literature review and research .....	23
3.1.2 DC motor transfer function development.....	23
3.1.3 Design the Controller .....	25
3.1.3.1 Construct Inputs and Outputs Data.....	26
3.1.3.2 Import Training and Testing Data into ANFIS Editor .....	27
3.1.3.3 Define ANFIS Structure & Membership Functions .....	27

3.1.3.4 Training the ANFIS .....	28
3.1.3.5 Testing the ANFIS with training data .....	29
3.1.4 Steering control transfer function development .....	29
3.1.5 System testing and modification.....	30
3.2 Final Year Project 2 methodology.....	30
3.2.1 Familiarize with xPC TargetBox.....	31
3.2.2 Modify the simulink model .....	31
3.2.3 Rapid prototyping using xPC Target.....	32
3.2.4 Design the amplifier for the project.....	32
3.2.5 xPC Target and xPC TargetBox .....	33
3.2.5.1 Host to xPC TargetBox communication.....	34
3.2.5.2 Creating a target boot disk.....	35
3.2.5.3 Booting an xPC TargetBox .....	36
3.2.5.4 Testing the installation .....	37
3.2.5.5 Entering Real-Time Workshop parameters ..	38
3.2.5.6 Building a Target application .....	41
3.2.5.7 Controlling the Target Application .....	43
3.2.5.8 Signal Tracing .....	43
3.2.5.9 Target PC and Hardware Configuration.....	45
CHAPTER 4.....	48
RESULTS AND DISCUSSION.....	48
4.1 Simulink model .....	48
4.1.1 DC Motor.....	48
4.1.2 Direction control.....	51
4.2 Controllers .....	53
4.2.1 Speed control.....	53
4.2.2 Direction control.....	54
4.3 Simulation result.....	55
4.3.1 Speed control .....	55
4.3.2 Sinusoidal path .....	56
4.3.3 Sudden change path.....	58
4.4 Running the hardware application.....	60
CHAPTER 5.....	61
CONCLUSION AND RECOMMENDATION .....	61
5.1 Conclusion.....	61
5.2 Recommendation.....	62
REFERENCES .....	63
APPENDICES.....	64
Appendix 1: DC Motor datasheet.....	64
Appendix 2: M-File for the input path.....	65
Appendix 3: DIAMOND-MM-32-AT specification sheet.....	67

## LIST OF FIGURES

FIGURE 1: Membership functions.....	7
FIGURE 2: Implication method .....	8
FIGURE 3: Aggregation step .....	10
FIGURE 4: Defuzzify step .....	11
FIGURE 5: Software design and deployment using MATLAB and Simulink .....	15
FIGURE 6: Real Time Workshop Application Areas.....	16
FIGURE 7: xPC Target and Real-Time solution .....	18
FIGURE 8: IO header pin out .....	21
FIGURE 9: Digital I/O Header.....	21
FIGURE 10: Electric circuit of the armature of the rotor.....	24
FIGURE 12: training data is imported .....	27
FIGURE 13: Membership functions is generated .....	28
FIGURE 14: ANFIS model structure .....	28
FIGURE 15: FIS Editor for the generated FIS.....	29
FIGURE 16: Rapid prototyping connection diagram.....	32
FIGURE 17: Unity Gain Buffer Amplifier.....	33
FIGURE 18: xPC Target Explorer window for communication node .....	34
FIGURE 19: Local Area Connection settings window .....	35
FIGURE 20: Creation of xPC Target boot disk window .....	36
FIGURE 21: Boot of xPC TargetBox in process .....	36
FIGURE 22: testing the installation .....	37
FIGURE 23: The modified simulink model.....	39
FIGURE 24: Configuration parameters window for real-time workshop node .....	40
FIGURE 25: Configuration parameters window for xPC Target options node .....	40
FIGURE 26: The model is built and downloaded into the xPC Target.....	41
FIGURE 27: Real-Time xPC Target Spy showing the model is built .....	42
FIGURE 28: xPC Target Explorer windows.....	42
FIGURE 29: xPc Target Explorer window .....	43
FIGURE 30: Target scope is running .....	44
FIGURE 31: Host scope is running.....	45
FIGURE 32: Connection between hardware and xPC TargetBox .....	46
FIGURE 33: Block parameter for analog output.....	47
FIGURE 34: Block parameter for analog input.....	47
FIGURE 35: Integrator blocks for the integral and rate of change .....	50
FIGURE 36: DC motor simulink model .....	50
FIGURE 37: Waveform for the output of the model .....	51
FIGURE 38: Limited integrator subsystem model.....	52
FIGURE 39: Steering control subsystem model .....	52
FIGURE 40: Simulink model.....	53
FIGURE 41: DC motor speed control simulink block .....	54
FIGURE 42: The step input result for dc motor speed control .....	55
FIGURE 43: sinus path result for steering .....	56
FIGURE 44: Error for direction control.....	57

FIGURE 45: Speed response.....57  
FIGURE 46: Sudden change path direction response .....58  
FIGURE 47: Graph of error vs. time..... 59  
FIGURE 48: Graph of speed vs. time..... 59



# CHAPTER 1

## INTRODUCTION

### 1.0 Background of Study

A lot of efforts have been contributed in recent days in developing a system that can improve car's performance especially in term of safety. The increasing interest toward the system is due to the increment of accident rate over time. Based on statistic obtained, the number of accidents happened to be incredibly increases time to time in which car and motorcycle become the main contributors for the statistic. Some the main causes to the increment of the accident rate are the careless and inability of the driver to efficiently control the car. In addition to that is the lack of safety system inside the vehicle especially for low price vehicles.

Due to those excuses, the main concern for the system is to increase the safety level for a vehicle thus improves the vehicle's performance and user's safety. A part of the system is an automated control system in which concentrates on automatic steering control system. The main concern is to enable a vehicle to follow the predetermined paths automatically with the smallest error.

With the same concerns and objectives with the previous works done by Mohamad Hanif, NHH [1], Dandre, P. [2] and Nor Hanisah Baharuddin [3], the project is introduced in order to continue the works in improving linear vehicle control system through the implementation of several methods in controller design. The previous works concentrated on developing a car model with automatic steering control. The resulted model was tested to follow the predetermined paths as accurate as possible. The designed controller used neural network and optimal preview as the base of the construction.

## 1.2 Problem Statement

Based on previous works, both theory used for the controller design has successfully produced a good self-guided vehicle controller. With the ability of neural network to predict and learn from a limited set of examples and optimal preview that able to represent well the driver's vision, the resultant self-guided vehicle really being able to follow the predetermined paths with acceptable errors. Integration of both methods in controlling the steering the model vehicle is viewed as successful tools in improving a car's performance and safety. Implication to that, the accident's rate also can be reduced thus save more lives.

However, the resultant control system designed still not very satisfying. This is due to the imprecise paths following by the model car which is represented by the amount of errors produced. Although, the model vehicle has the ability to follow the path, however the amount of errors produced could be further improved. The vehicle could not precisely follow the path especially at the large angle of corner. The errors become worse if the cruise speed of the vehicle increases.

This is because the model vehicle follows the path with the same speed although the angle of the corner is quiet high. The vehicle will not be able to take the corner precisely if the vehicle constantly moving in high speed. Although the model vehicle is equipped with both powerful methods: neural network and optimal preview for steering control, the outcome still not convincing. The vehicle still manages to moves out of track if it moves in constant high speed. Thus, the previous controller designed is still considered partly efficient and need some modifications in the design so that it can perform much better.

### **1.3 Objective and Scope of Study**

In order to improve the efficiency and performance of the model vehicle, there are several modifications that need to be done to the system. Some of the modification includes the introduction of speed control mechanism into the system. By adding the speed control mechanism, the error produced by the path following vehicle could be further reduced. The system can vary the speed of the vehicle according to the amount of angle of the steering. Thus, the precision of the vehicle's turning will be greatly improved. Therefore, it is to be the objective of the project to integrate both the steering control and speed control together in which can drive the self-guided vehicle to follow the prescribed paths with the smallest error.

The scope of the project is to design controllers that can monitor and control both the speed and direction of the vehicle according to the prescribed path. The vehicle should be able to follow the predetermined paths with acceptable error with the assistance of steering and speed control. For software part, the platform for the controller design will be MATLAB with the Simulink as the basis for construction. The vehicle will then be represented in real life application through the implementation of DC motor to represent the vehicle and xPC TargetBox for rapid prototyping.

For the controller design, different approaches are dedicated towards the project. There will be no neural network and optimal preview theory being used for the design. A new approach in which introduce the implementation of fuzzy logic and PID controller are carried out.

## **CHAPTER 2**

### **LITERATURE REVIEW**

This chapter review the basic idea of fuzzy logic and PID controller which contributes to the construction of the controller for the simulated paths and DC motor. In addition to that is an overview about permanent magnet DC motor which will be used as real time application. Beside that is an introduction to the software used in the project such as Simulink, Real-Time Workshop and xPC Target.

#### **2.1 Brief Outline on Previous Works**

This project has been conducted with reference to works by Dandre [2] on the upgraded performance of the optimal preview control with the use of neural network, Mohamad Hanif, NHH [1] on the combination of the neural network approach and of the optimal preview steering control to control a full non-linear steering model of a vehicle and Nor Hanisah Baharuddin [3] on the combination of neural network approach with optimal preview steering control to control linear vehicle.

The first reference shows comparison of weights initialized to zero, and weights obtained as optimal gains to implement single and multi-layered neural controller. The algorithm used is gradient descent, and the training method is identified as online (or incremental) training. Tracking simulations were done on linear and nonlinear models.

The second reference is about controlling a full non-linear steering model of a vehicle using the combination of both neural network controller and optimal preview steering control to increase path following accuracy. The first method, optimal preview has its particularity on the representation of the vision of the driver. The road information is viewed as part of the states and an augmented state is formed

with the states of the car. The second method, neural network concentrates on the design of the structure of a neural network controller and its training phase.

The third reference concentrates on the application of both optimal preview and neural network method to control linear vehicle as to increase path following accuracy. Different speeds were used in order to examine the capabilities of both methods in controlling the linear vehicle following the predetermined path.

## **2.2 Fuzzy Logic**

### **2.2.1 Background of Fuzzy Logic**

Few years back have witnessed a rapid growth in the number and variety of application of fuzzy logic. Those applications include from consumer product like cameras, camcorders, washing machines and microwave oven to industrial process control, medical instrumentation, decision-support systems, and portfolio selection.

Basically, fuzzy logic has two meanings. In narrow view, fuzzy logic is a logical system, which is an extension of multivalued logic. In wider view, fuzzy logic is almost synonymous with the theory of fuzzy sets, a theory which relates to classes of objects with unsharp boundaries in which membership is a matter of degree. In reference to this perspective, the narrow view can be viewed as the branch of fuzzy logic. Although in the narrow point of view, fuzzy logic still in different with traditional multivalued logic both in substances.

The basic idea of fuzzy logic is the linguistic variable that is a variable whose values are words rather than numbers. In effect, fuzzy logic can be viewed as a methodology for computing with rather than numbers. Although words is less precise than numbers, but their use is much closer to human intuition. Furthermore, computing with words exploit the tolerance for imprecision thus lowers the cost of solution.

Another basic concept of fuzzy logic which plays an important role in real life applications is that of a fuzzy if-then, or simply, fuzzy rule. In fuzzy logic, the machinery for dealing with fuzzy consequents and/or fuzzy antecedents is provided by so-called calculus of fuzzy-rules. The calculus serves as a basic for what might be called the Fuzzy Dependency and Command Language (FDCL). In this connection, the important thing to be concerned is that in most of the applications of fuzzy logic, fuzzy logic in reality is a translation of a human solution into FDCL [8].

A trend that is growing in visibility relates to the use of fuzzy logic in combination with neurocomputing and genetic algorithms. Moreover, fuzzy logic, neurocomputing and genetic algorithms may be viewed as the principal constituents of what might be called the soft-computing. Unlike the traditional method, hard-computing, soft-computing is aimed at the accommodation of pervasive imprecision of the real world. In coming years, soft-computing is likely to play an increasingly important role in the conception and design of system whose machine-IQ (MIQ) is much higher than of that constructed by conventional method [9].

### **2.2.2 Fuzzy Inference System (FIS)**

The process of developing a fuzzy logic controller is called fuzzy inference system (FIS). This system is the process of formulating the mapping from a given input to an output using fuzzy logic. The mapping then provides a basis from which decisions can be made, or patterns discerned. There are two types of fuzzy inference systems that can be implemented in the Fuzzy Logic Toolbox: *Mamdani*-type and *Sugeno*-type. These two types of inference systems vary somewhat in the way outputs are determined. The chosen type for the project is *Sugeno*-type.

The following are the explanation on each step used:

- a) Fuzzification of the input variables
- b) Application of the fuzzy operator (AND or OR) in the antecedent
- c) Implication from the antecedent to the consequent
- d) Aggregation of the consequents across the rule
- e) Defuzzification

### 2.2.3 Fuzzification of the input variables

In this step, the input variables are defined and determined the degree to which they belong to each of the appropriate fuzzy sets via membership functions.

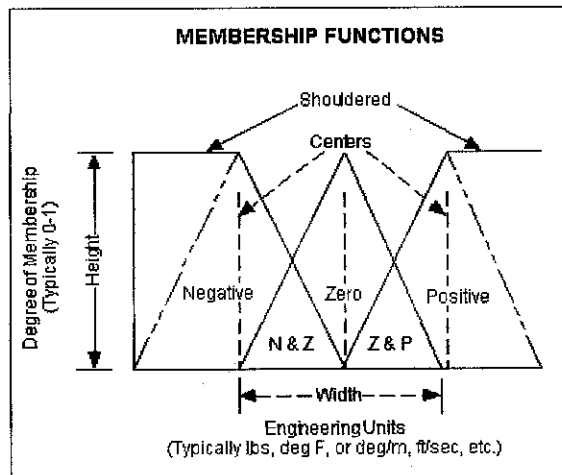


FIGURE 1: Membership functions

For the sake of the project, the inputs defined as the significant error (level) and significant integration-of-error (rate). In Fuzzy Logic Toolbox, the input is always a crisp numerical value limited to the universe of discourse of the input variable (in this case the interval between 0 and 10) and the output is a fuzzy degree of membership in the qualifying linguistic set (always the interval between 0 and 1). Fuzzification of the input amounts to either a table lookup or a function evaluation.

### 2.2.4 Application of the fuzzy operator

After the inputs are fuzzified, the degree to which each part of the antecedent is satisfied for each rule is known. If the antecedent of a given rule has more than one part, the fuzzy operator (AND or OR) is applied to represents the result of the antecedent for that rule. This number is then applied to the output function. The input to the fuzzy operator is two or more membership values from fuzzified input variables. The output is a single truth value.

The general form of fuzzy if-then rule is as follows:

*If <input1> is A and <input2> is B, then <output> is C*

### 2.2.5 Apply Implication Method

Before applying the implication method, the rule's weight needs to be determined first. Every rule has a *weight* (a number between 0 and 1), which is applied to the number given by the antecedent. Generally, this weight is 1 and thus has no effect at all on the implication process.

After proper weighting has been assigned to each rule, the implication method is implemented. A consequent is a fuzzy set represented by a membership function, which weights appropriately the linguistic characteristics that are attributed to it. The consequent is reshaped using a function associated with the antecedent (a single number). The input for the implication process is a single number given by the antecedent, and the output is a fuzzy set. Implication is implemented for each rule. Two built-in methods are supported, and they are the same functions that are used by the AND method: *min* (minimum), which truncates the output fuzzy set, and *prod* (product), which scales the output fuzzy set. Below is the example of implication method:

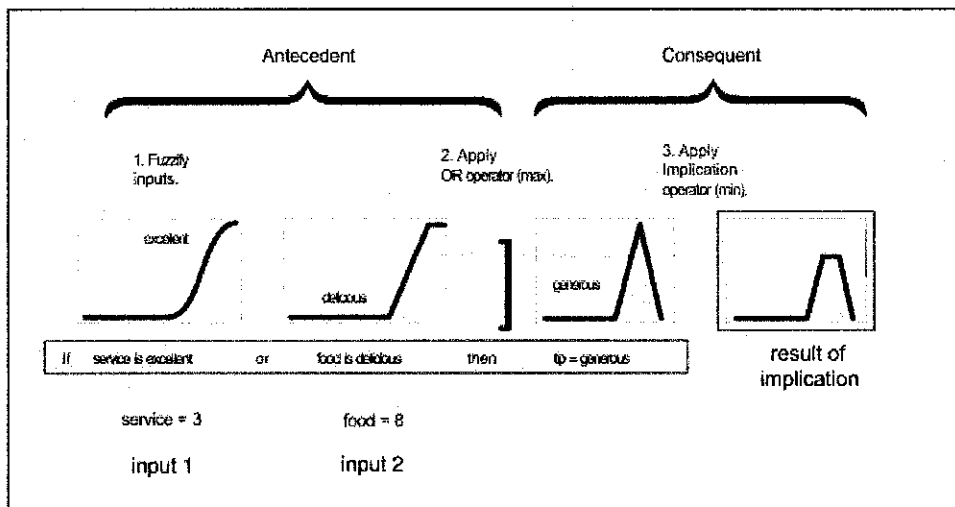


FIGURE 2: Implication method



### 2.2.6 Aggregate all outputs

Because decisions are based on the testing of all of the rules in an FIS, the rules must be combined in some manner in order to make a decision. Aggregation is the process by which fuzzy sets that represent the output of each rule are combined into a single fuzzy set. The input of the aggregation process is the list of truncated output functions returned by the implication process for each rule. The output of the aggregation process is one fuzzy set for each output variable.

As long as the aggregation method is commutative (which it always should be), then the order in which the rules are executed is unimportant. Three built-in methods are supported:

- max (maximum)
- probor (probabilistic OR)
- sum (simply the sum of each rule's output set)

In the following diagram, all three rules have been placed together to show how the output of each rule is combined, or aggregated, into a single fuzzy set whose membership function assigns a weighting for every output (tip) value.

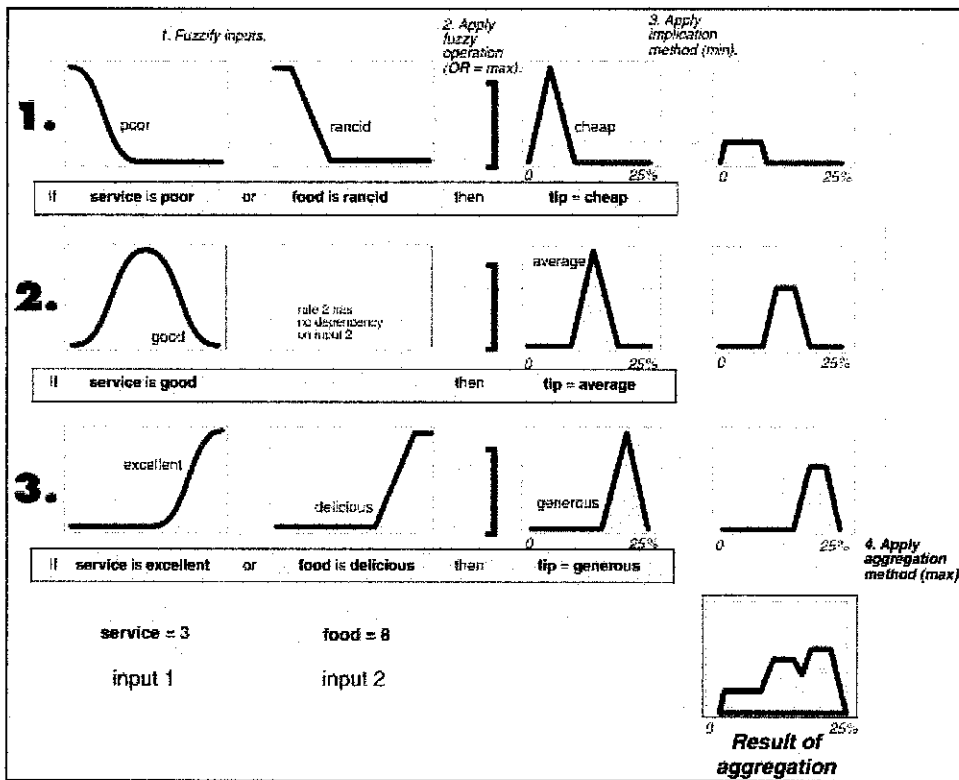


FIGURE 3: Aggregation step

### 2.2.7 Defuzzify

The input for the defuzzification process is a fuzzy set (the aggregate output fuzzy set) and the output is a single number. As much as fuzziness helps the rule evaluation during the intermediate steps, the final desired output for each variable is generally a single number. However, the aggregate of a fuzzy set encompasses a range of output values, and so must be defuzzified in order to resolve a single output value from the set.

Perhaps the most popular defuzzification method is the centroid calculation, which returns the center of area under the curve. There are five built-in methods supported: centroid, bisector, middle of maximum (the average of the maximum value of the output set), largest of maximum, and smallest of maximum.

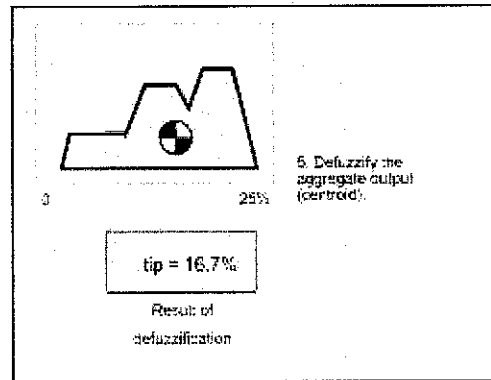


FIGURE 4: Defuzzify step

### 2.3 PID controller

The next method required for the design of the controller is PID controller concept. Basically, PID Controller is used for various control problems such as automated systems or plants. A PID Controller consists of three major parts:

- a) P- Proportional control
- b) I – Integral Control
- c) D – Derivative control

To fully understand the operation of a PID feedback controller, the three elements should be understood first. Proportional control is a pure gain adjustment acting on the error signal to provide the driving input to the process. It is used to adjust the speed of the system in the experiment. Integral control is implemented through the introduction of an integral. Integral control is used to provide the required accuracy for the control system. The derivative control is used to increase the damping of the system. It also amplifies the existing noise, which cause problems including instability.

### 2.4 Overview on DC motor

DC motor is one of the electric motor, which converts the electrical energy to mechanical energy. Its function is an opposite of generator or dynamo which converts mechanical energy to electrical energy. DC motor consists of two basic

parts: the rotating part which called armature or rotor and stationary part that includes coils of wire called the field coil or a stator.

Basic operation of DC motor is based on the concept of electromagnetism. The fundamental principle of the operation is that there is a mechanical force on current carrying wire contained within a magnetic field. The force is described by the Lorentz force law and is perpendicular to both the wire and the magnetic field. Most magnetic motors are rotary, but linear motors also exist. The rotor rotates because the wires and magnetic field are arranged so that a torque is developed about the rotor's axis. The motor contains electromagnets that are wound on a frame (armature). Correctly, the armature is that part of the motor across which the input voltage is supplied. Depending upon the design of the machine, either the rotor or the stator can serve as the armature.

Generally, the rotational speed of the motor is proportional to the voltage applied and the torque proportional to the current applied. It can be accomplished with the introduction of variable battery tapping, variable resistor, power supply or electronic controls. The effective voltage can be varied by inserting a series resistor or by an electronically controlled switching device made of thyristors, transistors, or, formerly, mercury arc rectifiers.

For this project, the DC motor used is a permanent magnet type. Generally, permanent magnet dc motor (PMDC) is a dc motor whose poles are made of permanent magnets. This type of dc motor offers a number of benefits compared with shunt dc motors in some applications. Since these motors do not require an external field circuit, they do not have the field circuit copper losses associated with shunt dc motors. Because no field windings are required, they can be smaller than corresponding shunt dc motors.

However, PMDC motors also have disadvantages. Permanent magnets can not produces as high a flux density as an externally supplied shunt field, so a PMDC will have lower induced torque  $\tau$  per ampere of armature current  $I_A$  than a shunt

motor of the same size and specifications. In addition, PMDC motors run the risk of demagnetization. Also the pole flux is just the residual flux in the permanent magnet.

Basically, PMDC motor is the same machine as a shunt dc motor, except that the flux of a PMDC motor is fixed [9]. Therefore, it is not possible to control the speed of a PMDC motor by varying the field current or flux. The only methods of speed control available for a PMDC motor are armature voltage control and armature resistance control.

## **2.5 MATLAB Based software**

### **2.5.1 Simulink**

Simulink developed by The Mathworks, is a tool modeling, simulating and analyzing multidomain dynamic system [8]. It supports both linear and nonlinear systems, modeled in continuous time, sampled time or a hybrid of the two. Its primary interface is a graphical block diagramming tool and a customizable set of block libraries. It offers a good integration with the rest of the MATLAB environment and both drive MATLAB or are coded from it.

For modeling, Simulink provides a graphical user interface (GUI) for building models as block diagrams, using click-and-drag mouse operations. With this features, the user can simply draw the model just like the usage of pencil and paper. The user does not need to formulate differential equations and difference equations in a language or program. Simulink includes a comprehensive block library of sinks, sources, linear and nonlinear components, and connectors.

Add-on products extend the Simulink environment with tools for specific modeling and design tasks and for code generation, algorithm implementation, test, and verification. Integrate with MATLAB, Simulink provides immediate access to an

extensive range of tools for algorithm development, data visualization, data analysis and access, and numerical computation.

Some of key features offered by this application include:

1. Extensive and expendable libraries of predefined blocks
2. Interactive graphical editor for assembling and managing intuitive block diagrams
3. Ability to manage complex designs by segmenting models into hierarchies of design components
4. Model explorer to navigate, create, configure, and search all signals, parameters, and properties of the model
5. Ability to interface with other simulation programs and incorporate hand-written code, including MATLAB algorithms
6. Option to run fixed- or variable-step simulations of time-varying systems interactively or through batch simulation

### **2.5.2 Real-Time workshop**

Real-Time Workshop® is an extension of capabilities of Simulink® and MATLAB® that automatically generates packages and compiles source code from Simulink models to create real-time software applications on a variety of systems including simulation acceleration, rapid prototyping, and hardware-in-the-loop testing. [8]

By providing a code generation environment for rapid prototyping and deployment, Real-Time Workshop is the foundation for production code generation capabilities. The user can interactively tune and monitor the generated code using Simulink blocks and built-in analysis capabilities, or run and interact with the code outside the MATLAB and Simulink environment.

Some of the key features include:

1. It generates ANSI/ISO C code and executables for discrete, continuous, or hybrid models
2. Uses model blocks to incrementally generate and build code for large applications
3. Supports Simulink data dictionary features for integer, floating-point, and fixed-point data
4. Generates code for single-rate, multirate, and asynchronous models
5. Supports single-tasking and multitasking operating systems and bare-board (no operating system) environments
6. Performs code optimizations that improve code execution speed

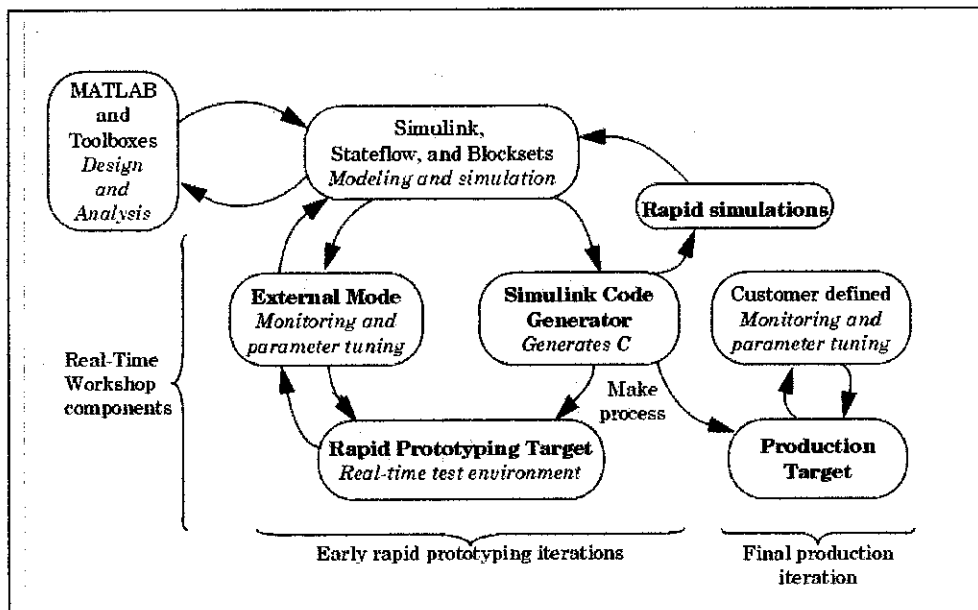


FIGURE 5: Software design and deployment using MATLAB and Simulink

Real-Time Workshop has a wide array of application areas. Most of these applications can be categorized as on of the following:

1. Rapid prototyping applications
2. Embedded applications
3. Rapid simulations

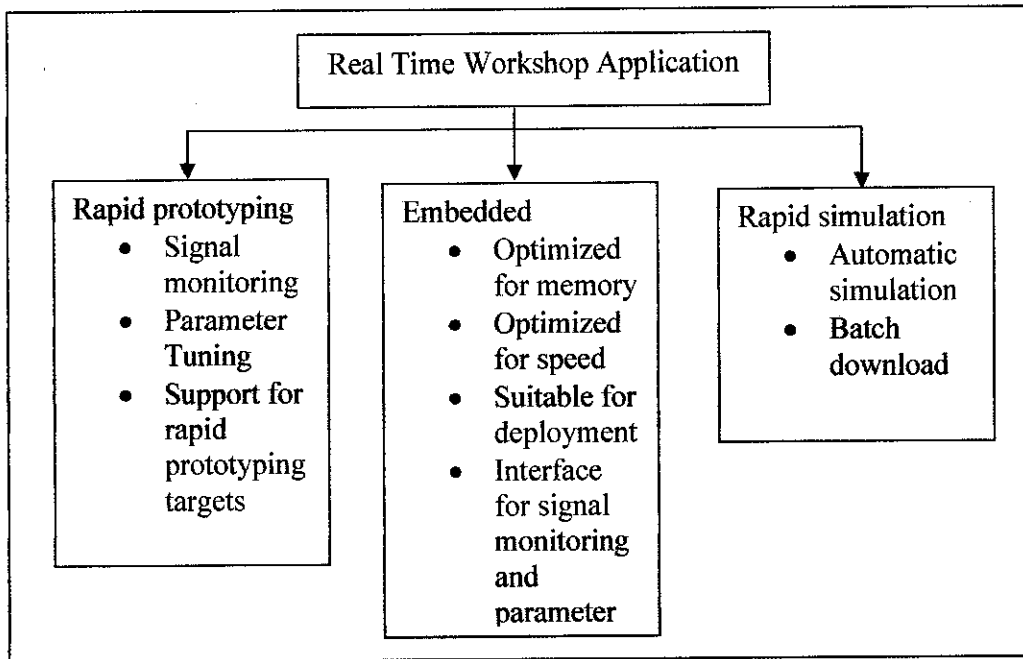


FIGURE 6: Real Time Workshop Application Areas

For the project, the preferable application of Real-Time Workshop is rapid prototyping. This type of application allows the user to:

1. Conceptualize solutions graphically in a block diagram modeling environment
2. Evaluate system performance early on--before laying out hardware, coding production software, or committing to a fixed design
3. Refine the design by rapid iteration between algorithm design and prototyping
4. Tune parameters while the real-time model runs, using Simulink in external mode as a graphical front end

### 2.5.3 xPC Target

The xPC Target is a solution for prototyping, testing, and deploying real-time systems using standard PC hardware. It is an environment that uses a target PC, separate from a host PC, for running real-time applications [8]. It provides a high performance, host-target prototyping environment that enables the user to connect



the Simulink models to physical systems and execute them in real time on PC compatible hardware.

xPC Target has the ability to work with rapid prototyping and hardware-in-the-loop simulation of control systems. It enables the user to add on I/O interface blocks to the models, automatically generate code with Real-Time Workshop and download the code to a second PC running the xPC Target real-time kernel.

Some of the key features of xPC Target include:

1. Runs applications generated from Simulink models using a real-time kernel on any PC
2. Supports any desktop PC, PC/104, CompactPCI, industrial PC, or single-board PC computer as a real-time target system
3. Achieves sample rates approaching 50 KHz, depending on processor performance level and model size
4. Supports more than 250 standard I/O boards, with an extensive I/O device driver library, including driver source code
5. Enables signal acquisition and parameter tuning from the host or target PC
6. Acquires and logs data in real time to RAM or the file system of the target PC
7. Displays data and signal traces on the host PC, target PC, or both

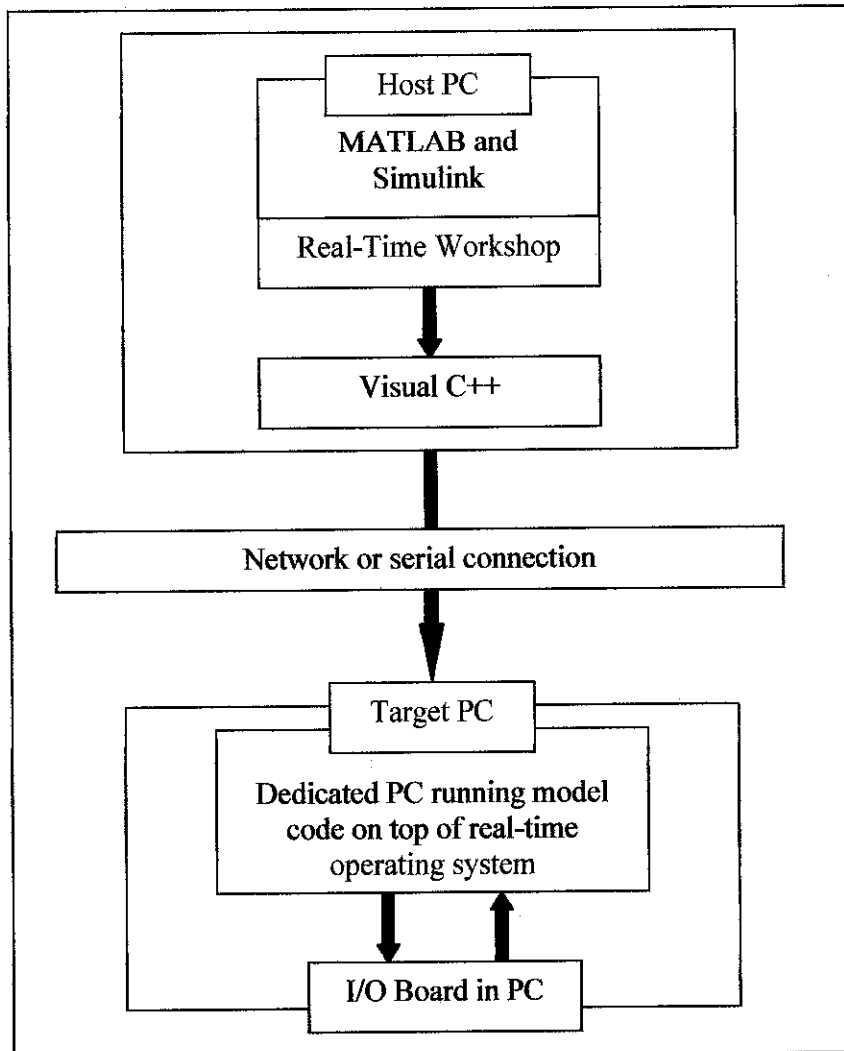


FIGURE 7: xPC Target and Real-Time solution

#### 2.5.4 xPC TargetBox

xPC TargetBox is an industrial PC system that combines performance, ruggedness and I/O expandability in a compact package specifically designed for rapid control prototyping applications [8]. One of the key features of xPC TargetBox for the project is that it supports I/O options for typical rapid prototyping requirements such as A/D, D/A, DIO, PWM, counters, timers, encoders, and CAN bus.

There are seven available I/O options, which satisfy typical rapid prototyping requirements:

- a) *xPC TargetBox IO 301* – Analog and Digital Input and Output
- b) *xPC TargetBox IO 302* – Analog Output and Digital Input and Output
- c) *xPC TargetBox IO 303* – Analog Output and Digital Input and Output
- d) *xPC TargetBox IO 304* – Digital Input and Output
- e) *xPC TargetBox IO 305* – General-Purpose Counters
- f) *xPC TargetBox IO 306* – Incremental Encoder Input
- g) *xPC TargetBox IO 308* – CAN Bus Interface

The one that will be used for the project is *xPC TargetBox IO 301* type which is DIAMOND-MM-32-AT. This IO board is a PC/104-format data acquisition board with a full set of features. Some of the features of this board:

a) Analog Inputs

- 32 input channels, may be configured as 32 single-ended, 16 differential or 16 SE + 8 DI 16bit resolution
- Programmable gain, range, and polarity on inputs
- 200,000 samples per second maximum sampling rate
- 512-sample FIFO for reduced interrupt overhead
- Autocalibration of all input ranges under software control

b) Analog Outputs

- 4 analog output channels with 12-bit resolution, 5mA max output current
- Multiple fixed full-scale output ranges, including unipolar and bipolar ranges
- Programmable full-scale range capability
- Autocalibration under software control

c) Digital IO

- 24 bidirectional lines using integrated 8255-type circuit
- Buffered I/O for enhanced current drive

- Handshaking controls enable external latching of data as well as interrupt operation
- User-configurable pull-up / pull-down resistors
- 7 auxiliary I/O lines are fixed direction with programmable functions

d) Counter/Timers and A/D Triggering

- 1 32-bit counter/timer for A/D pacer clock and interrupt operation timing
- 1 16-bit general purpose counter/timer
- Programmable input sources for each counter/timer
- External A/D triggering and gating inputs
- Multiple-board synchronization capability using A/D convert pulse out and external trigger in
- Interrupts may be generated by counter/timer

e) Miscellaneous

- Extended temperature  $-40$  to  $+85^{\circ}\text{C}$  operation
- No trimpots or user adjustments required for calibration

Diamond-MM-32-AT provides a 50-pin header on the right edge of the board labeled J3 for all I/O relating to analog functions. The terminals on the screw terminal boards are numbered sequentially while the pin numbers on the connectors are numbered alternately left and right. However, the numbers printed on each screw terminal board correspond directly to the pin numbers provided in the next figure.

J3: Analog I/O Header			
AGND	1	2	AGND
Vin 0 / 0+	3	4	Vin 16 / 0-
Vin 1 / 1+	5	6	Vin 17 / 1-
Vin 2 / 2+	7	8	Vin 18 / 2-
Vin 3 / 3+	9	10	Vin 19 / 3-
Vin 4 / 4+	11	12	Vin 20 / 4-
Vin 5 / 5+	13	14	Vin 21 / 5-
Vin 6 / 6+	15	16	Vin 22 / 6-
Vin 7 / 7+	17	18	Vin 23 / 7-
Vin 8 / 8+	19	20	Vin 24 / 8-
Vin 9 / 9+	21	22	Vin 25 / 9-
Vin 10 / 10+	23	24	Vin 26 / 10-
Vin 11 / 11+	25	26	Vin 27 / 11-
Vin 12 / 12+	27	28	Vin 28 / 12-
Vin 13 / 13+	29	30	Vin 29 / 13-
Vin 14 / 14+	31	32	Vin 30 / 14-
Vin 15 / 15+	33	34	Vin 31 / 15-
Vout 3	35	36	Vout 2
Vout 1	37	38	Vout 0
Vref Out	39	40	Agnd
A/D Convert	41	42	Ctrl 2 Out / Dout 2
Dout 1	43	44	Ctrl 0 Out / Dout 0
Extclk / Din 3	45	46	Extgate / Din 2
Gate 0 / Din 1	47	48	Clk 0 / Din 0
+5V	49	50	Dgnd

FIGURE 8: IO header pin out

Diamond-MM-32-AT provides a 34-pin header on the left edge of the board labeled J4 for the 24 8255-type digital I/O lines. The terminals on the screw terminal boards are numbered sequentially while the pin numbers on the connectors are numbered alternately left and right. However, the numbers printed on each screw terminal board correspond directly to the pin numbers provided in the next figure.

J4: Digital I/O Header			
A7	1	2	A6
A5	3	4	A4
A3	5	6	A2
A1	7	8	A0
B7	9	10	B6
B5	11	12	B4
B3	13	14	B2
B1	15	16	B0
C7	17	18	C6
C5	19	20	C4
C3	21	22	C2
C1	23	24	C0
Latch	25	26	Ack
NC	27	28	NC
NC	29	30	NC
NC	31	32	NC
+5V	33	34	Dgnd

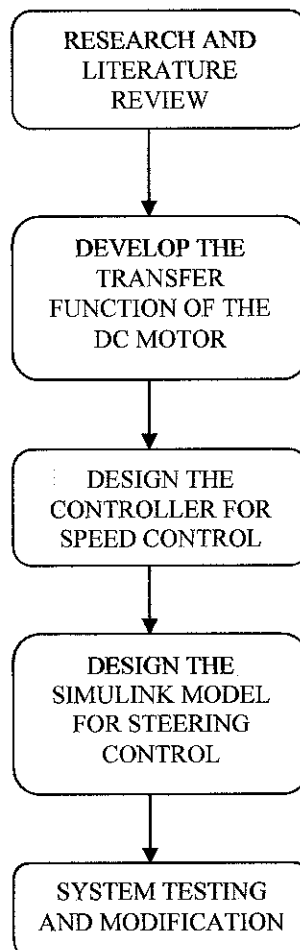
FIGURE 9: Digital I/O Header

## CHAPTER 3

### METHODOLOGY

This chapter demonstrates a list of methodology and works that have been defined and need to be carried out through out the project in order to accomplish the objective and meets the given time frame. Some of the works need to be performed and completed during the first semester and some of them need to be carried out until the end of second semester.

#### 3.1 Final Year Project 1 methodology



For the first semester of final year project, the main focus was to develop a simulink model for the system and simulated it in non-real time environment. The resulted simulink model would then be carried out to the second semester for the real-time application through the introduction of xPC TargetBox.

### **3.1.1 Literature review and research**

The first work is to conduct research and literature review regarding the background of the project and methods required for the system. All the concepts and algorithm used to design the controller need to be defined clearly before start working on the project. The research process continued with the literature review on the speed and steering control mechanism. This part is really important because the main objective of the project is to develop a controller that can control both of them. The other thing that also contributes to the project and need to be researched is the DC motor. The DC motor is important because it represents the linear vehicle and will be working together with xPC Targetbox in next semester.

### **3.1.2 DC motor transfer function development**

The next move is to concentrate on the development of speed control mechanism of the DC motor. The DC motor, which is borrowed from control lab, need to be tested and analyzed in order to get several information in constructing the transfer function. The transfer function then will be used as the “plant” that is controlled by the designed controller in the system. The tests and analyzes are performed in the machine lab with supervised by technician.

To obtain the transfer function of the motor, several methods were reviewed. The method chosen for this project is from Carnegie Mellon’s undergraduate controls lab from University of Michigan. Below is the electric circuit of the armature of the rotor:

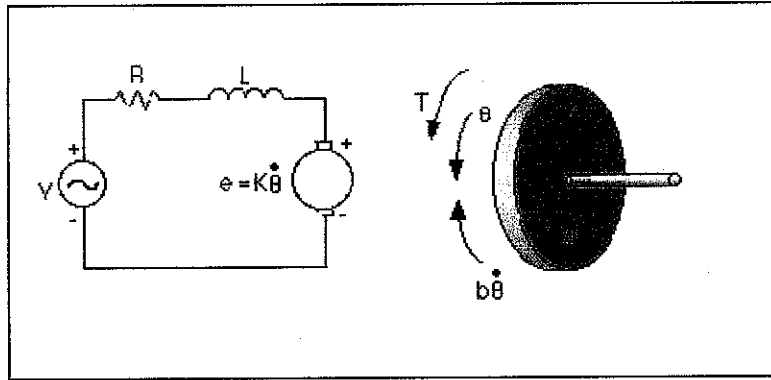


FIGURE 10: Electric circuit of the armature of the rotor

There are some values that were assumed for simplicity of the calculation and experiment:

- a) Moment of inertia of the rotor ( $J$ ) =  $0.01 \text{ kg.m}^2/\text{s}^2$
- b) Damping ratio of the mechanical system ( $b$ ) =  $0.1 \text{ Nms}$
- c) Electromotive force constant ( $K=K_e=K_t$ ) =  $0.01 \text{ Nm/Amp}$
- d) Electric resistance ( $R$ ) =  $1 \text{ Ohm}$
- e) Electric inductance ( $L$ ) =  $0.5 \text{ H}$
- f) Input ( $V$ ) = Source voltage
- g) Output ( $\theta$ ) = position of shaft

Theoretically, motor torque,  $T$  is related to the armature current,  $I$  by a constant factor  $K_t$ . The back emf,  $e$  is related to the rotational velocity by the following equations:

$$T = K_t i \quad (1)$$

$$E = K_e \dot{\theta} \quad (2)$$

The equation that can be derived based on Newton's law combined with Kirchoff's law:

$$J\ddot{\theta} + b\dot{\theta} = K_i i \quad (3)$$

$$L \frac{di}{dt} + Ri = V - K \dot{\theta} \quad (4)$$



Using Laplace transforms, the previous modelling equations can be expressed as:

$$s(Js+b) \theta(s) = KI(s) \quad (5)$$

$$(Ls+R)I(s) = V - Ks \theta(s) \quad (6)$$

By eliminating  $I(s)$ , the open loop transfer function where the rotational speed is the output and the voltage is the input can be obtained

$$\frac{\Theta}{V} = \frac{K}{(Js+b)(Ls+R)+K^2} \quad (7)$$

Substituting the values of the parameters to the equation

$$\frac{\Theta}{V} = \frac{0.01}{(0.01s+0.1)(0.5s+1)+0.01^2} \quad (8)$$

### 3.1.3 Design the Controller

Next step is the design of the controller. In the early stage, the student concentrates on speed control since it is much easier compared to path following control. The controller designed uses a lag compensator and step value as the input. The function of the controller will be extended in order to control both functions with the introduction of fuzzy logic mechanism.

The method that is used to design the fuzzy logic controller is called Sugeno-type of fuzzy inference. This method implements ANFIS or Adaptive Neuro-Fuzzy Inference System that a method for the fuzzy modeling procedure to learn information about a data set, in order to compute the membership function parameters that best allow the associated fuzzy inference system to track the given input/output data. This learning method works similarly to the neural networks. This technique allows the user to identify a system model by using test data to identify and train a FIS. As said before, it based on back propagation and least squared methods.

In order to implement the ANFIS editor, there are several steps that need to be taken as shown below:

### 3.1.3.1 Construct Inputs and Outputs Data

To start training an FIS using ANFIS, it is really important to have a training data set contains of desired inputs and outputs. For the project, the desired input is the path that needs to be followed by the steering which is represented by the sinusoidal signal. Through the implementation of PID controller, the steering is controlled to follow the path with the smallest error which gives the data for output. By implementing *To workspace* simulink block, all the input, integral of input and output is fed to the workspace as arrays.

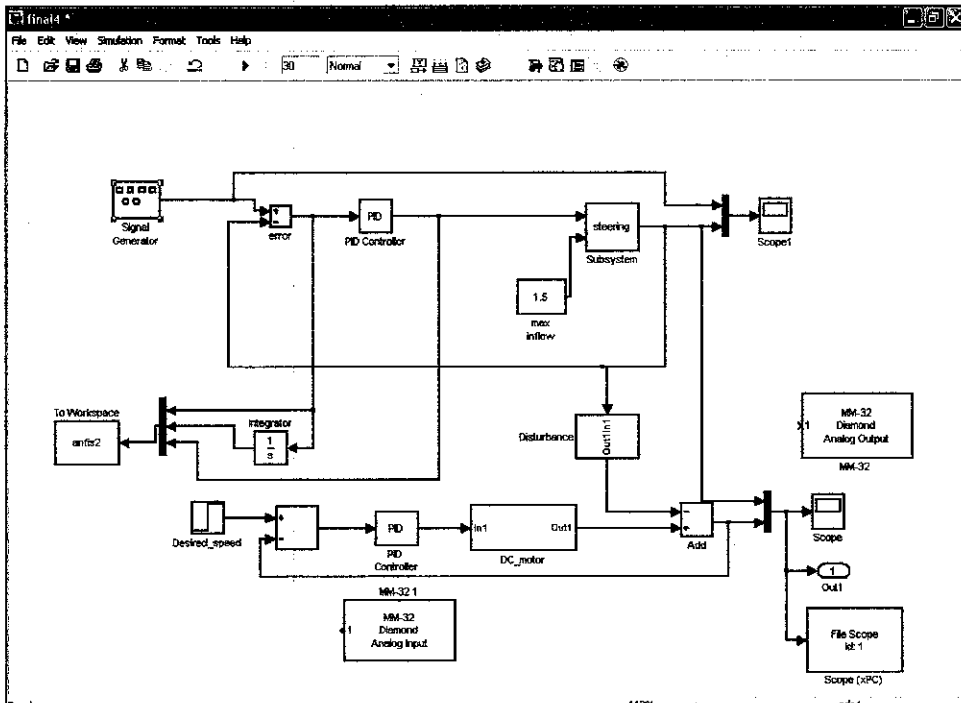


FIGURE 11: Training simulink model

### 3.1.3.2 Import Training and Testing Data into ANFIS Editor

From the ANFIS editor, the training data is imported from the workspace by selecting *Load Data From Workspace* button and typing the array title. The loaded training data is shown below:

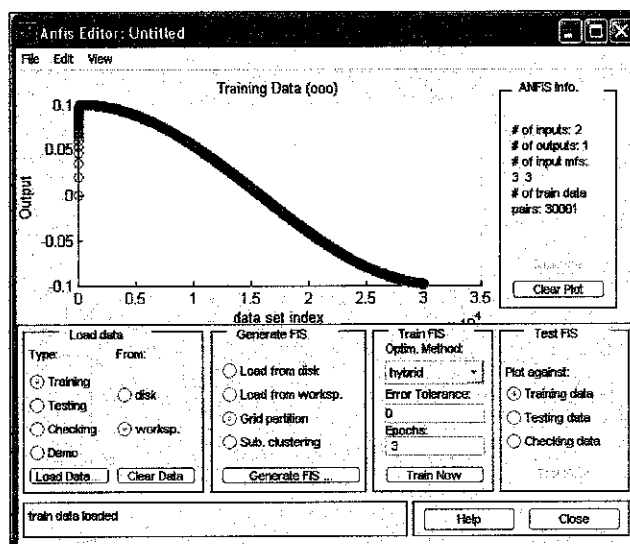


FIGURE 12: training data is imported

### 3.1.3.3 Define ANFIS Structure & Membership Functions

The FIS is generated by defining the membership functions for both inputs and output by clicking the **Generate FIS** button. After that, the model structure build can be viewed by clicking the button **Structure**

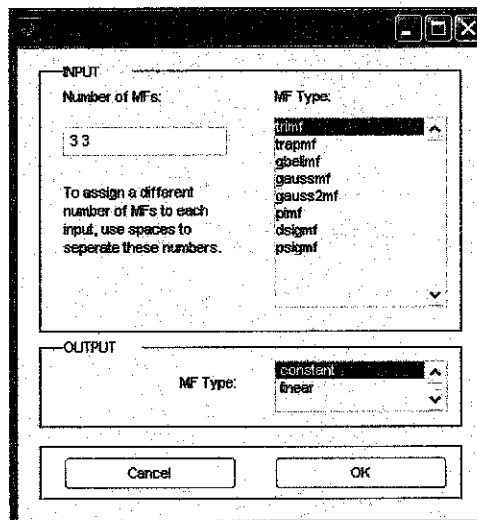


FIGURE 13: Membership functions is generated

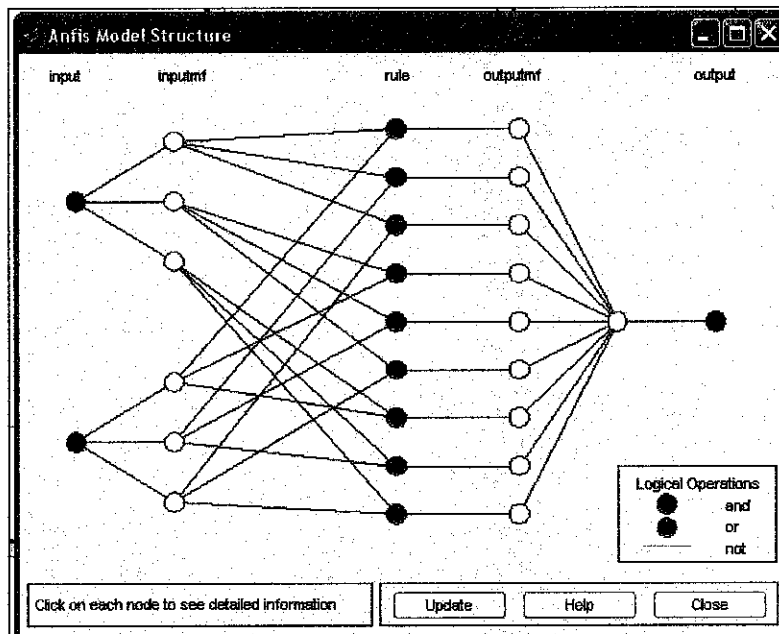


FIGURE 14: ANFIS model structure

### 3.1.3.4 Training the ANFIS

To train the FIS, there are two methods for optimization: *backpropagation* or *hybrid* (back propagation and least square) that can be chosen. The training can be started

by clicking the **Train Now** button. The user can select the amount of epochs that would be used for training.

### 3.1.3.5 Testing the ANFIS with training data

After completing the training step, the next thing to be done is testing the FIS with the training data to verify the functionality of the FIS. Then, the resultant FIS is saved into the disk.

After that, the generated FIS is ready to be implemented to the fuzzy logic controller by typing the title in the block properties of the fuzzy logic controller

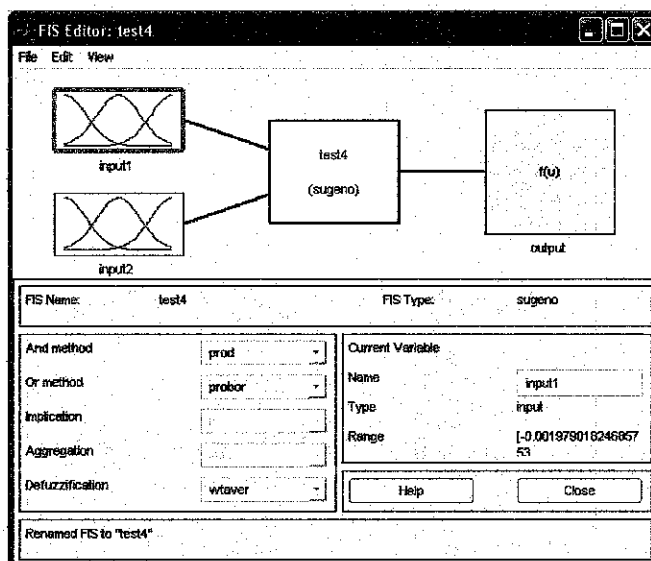


FIGURE 15: FIS Editor for the generated FIS

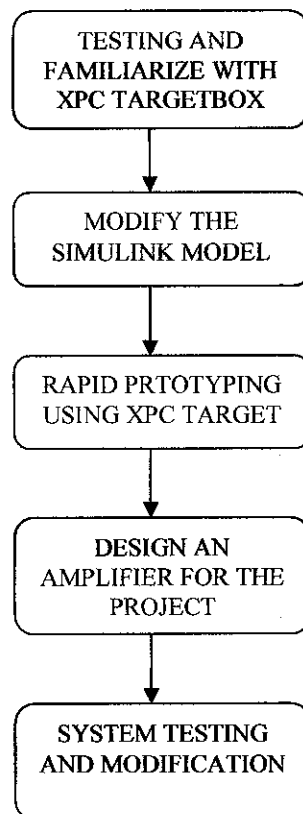
### 3.1.4 Steering control transfer function development

After completed the development of transfer function for DC motor, the progress continued with the development of steering control simulink model. With the reference from several resources, the simulink model has been developed with fuzzy logic as the basis for the controller design.

### 3.1.5 System testing and modification

Since several variables used in the experiment have been defined earlier in the previous project, so the student no needs to construct new paths for the system. The main focus of the step after designing the controller is to concentrate on injecting the input paths to the system to observe the successful rate of the project. The successful rate is defined as the path following error of the system.

### 3.2 Final Year Project 2 methodology



For the second semester of final year project, the focus of the project is subjected to the hardware or real-time application with the usage of xPC TargetBox and DC motor. The developed simulink model from the previous semester will be used and modified in order to implement it with xPC TargetBox.

### **3.2.1 Familiarize with xPC TargetBox**

In order to implement xPC TargetBox as a real-time application, several readings and testing have been made. One of the reading materials referred during the familiarization stage is the training module borrowed from supervisor. The module gives a basic understanding on how to build a simple simulink model and how to apply it in real time workshop and xPC TargetBox. Also included in the module is an overview on fuzzy logic and how to apply it in MATLAB.

Also included in the module is some explanation on Real-Time Workshop which is an important element for the project. This application has the capability to automatically generates, packages and compiles source code from Simulink models to create real-time software applications which will be implemented on xPC TargetBox.

### **3.2.2 Modify the simulink model**

In order to implement the model for xPC targetBox application, several modifications need to be done on the model. Those modifications include the introduction of additional blocks and modifications on the parameters. The main modifications are the insertion of output block and xPC Target scope block. These two blocks are important due to the xPC TargetBox requirements. The insertion of output block is important because it helps log the signal data to the Matlab workspace for analysis and later save it to a disk. The insertion of xPC Target scope block is done because xPC TargetBox does not support the standard Simulink scope block. It only supports xPC Target scope block in its application. This type of scope helps the user to automatically display the result on the target PC monitor once the target application is downloaded.

### 3.2.3 Rapid prototyping using xPC Target

On this type of development process, the code is generated from the controller and is downloaded onto the hardware (xPC TargetBox), which in turns interact with the plant. Some of the steps taken for this stage are discussed in the next section.

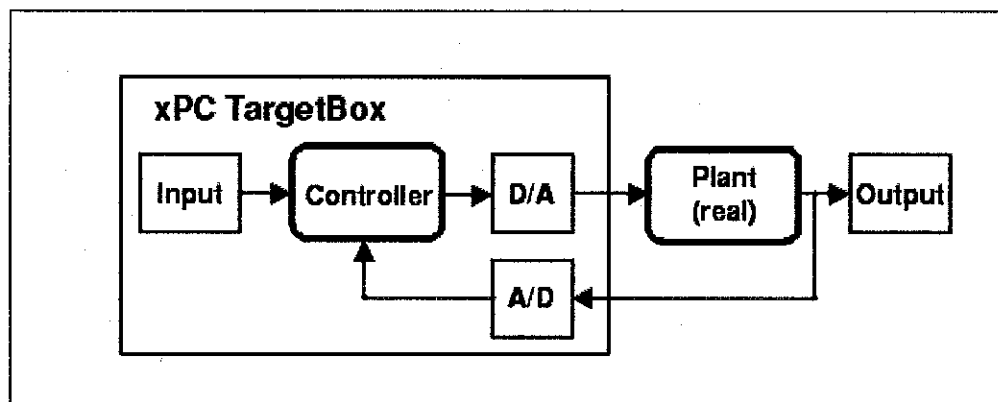


FIGURE 16: Rapid prototyping connection diagram

### 3.2.4 Design the amplifier for the project

It is important to build and design an amplifier for the project since the output from the xPC TargetBox is not enough to control the DC motor. So, several research and reading have been made in order to design a suitable amplifier for the project.

From the research, one of the solutions for amplifier problem is through the usage of op-amp which is called unity gain buffer amplifier. The circuit essentially just makes a copy at the output of the input voltage,  $V_{in}$ . It does that without drawing any current from wherever the input voltage terminal is attached. However, at the output terminal it can draw whatever amount of current the operational amplifier can supply.

Typically a buffer amplifier is used to transfer a voltage from a first circuit, having a high output impedance level, to a second circuit with a low input impedance level. The interposed buffer amplifier prevents the second circuit from loading the first circuit unacceptably and interfering with its desired operation.



The importance of this circuit does not come from any change in voltage, but from the input and output impedances of the op-amp. The input impedance of the op-amp is very high ( $M\Omega$  to  $10\ T\Omega$ ), meaning that the input of the op-amp does not load down the source or draw any current from it. Because the output impedance of the op-amp is very low, it drives the load as if it were a perfect voltage source. Both the connections to and from the buffer are therefore bridging connections, which reduce power consumption in the source, distortion from overloading, crosstalk and other electromagnetic interference.

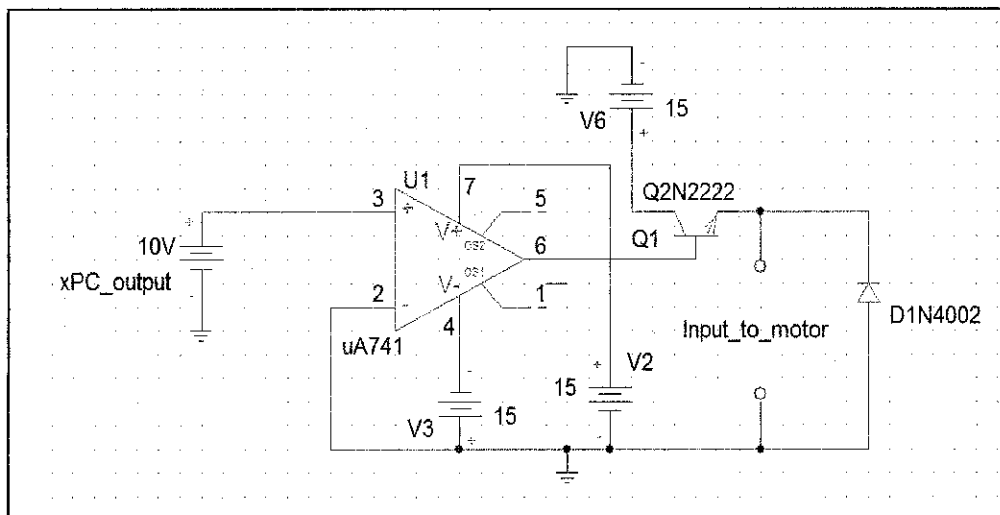


FIGURE 17: Unity Gain Buffer Amplifier

### 3.2.5 xPC Target and xPC TargetBox

For the initial stage of the semester, there are a few sequence of steps have been taken to implement the xPC TargetBox. Those include the installation of loop-back testing, connecting the hardware, initiates the communication and testing the booting process.

### 3.2.5.1 Host to xPC TargetBox communication

This process of connecting both host PC and target PC needs to be performed first before create and run a target application. There are two types of communication that can be established for both PCs which are: serial and network communication.

For the project, network communication is more preferable than serial communication due to several advantages:

1. **Higher data throughput-** network communication using Ethernet can transfer data up to 100 Mbit/second instead of the maximum data transfer rate of 115 kBaud with serial communication.
2. **Longer distances between host and target computer --** By using repeaters and gateways there are no worried over a distance communication.

To establish the communication, several configurations were done at the xPC Target Explorer window. From the window, *communication* node was chosen and the diagram below was displayed.

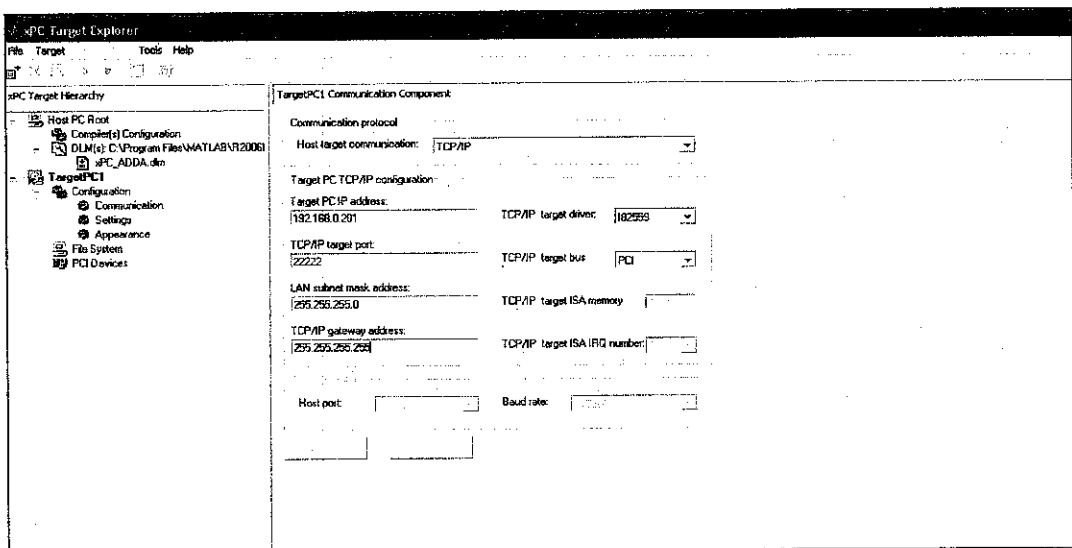


FIGURE 18: xPC Target Explorer window for communication node

The Local Area Connection (LAN) was set up as shown in the picture above. The IP Address for LAN must not be the same as IP Address for Target PC. Once the connection settings were set up, the communication is ready and established.

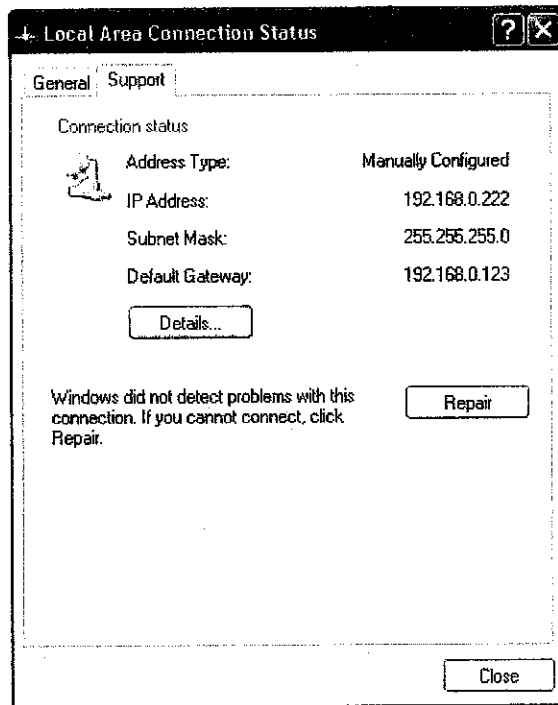


FIGURE 19: Local Area Connection settings window

### 3.2.5.2 Creating a target boot disk

xPC Target does not require any operating system to be operated on the target PC. Instead, the target PC is booted with a boot disk that contains xPC Target kernel. The boot disk eliminates the need to install software, modifying the existing software configurations or access the hard disk on the target PC. This arrangement allows the user to use the target PC as a desktop computer after finished testing real-time applications.

In order to create the target boot disk, there are several simple procedures have been carried out:

1. In the MATLAB command window, a command "*xpcsetup*" was typed to open the xPC Target Setup window.

2. The “*BootDisk*” button was clicked to pop up the below window
3. A formatted 3.5 inch floppy disk was then inserted into the host PC disk drive. The xPC Target displayed the following dialog box while creating the boot disk.

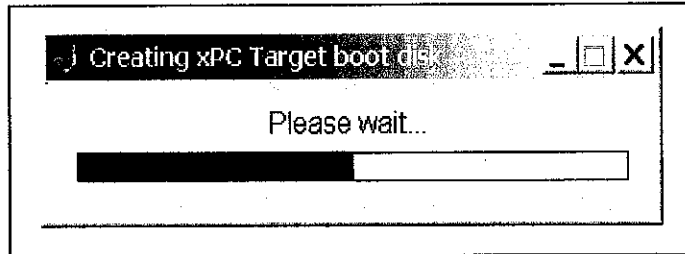


FIGURE 20: Creation of xPC Target boot disk window

4. After finished the creating process, the floppy disk was then inserted to the external disk drive connected to the xPC TargetBox.

### 3.2.5.3 Booting an xPC TargetBox

After created the target boot disk and established the network connection, the xPC TargetBox is then ready to be powered up and tested. This step was performed in purpose to check whether the xPC TargetBox can have a direct contact with host PC or not.

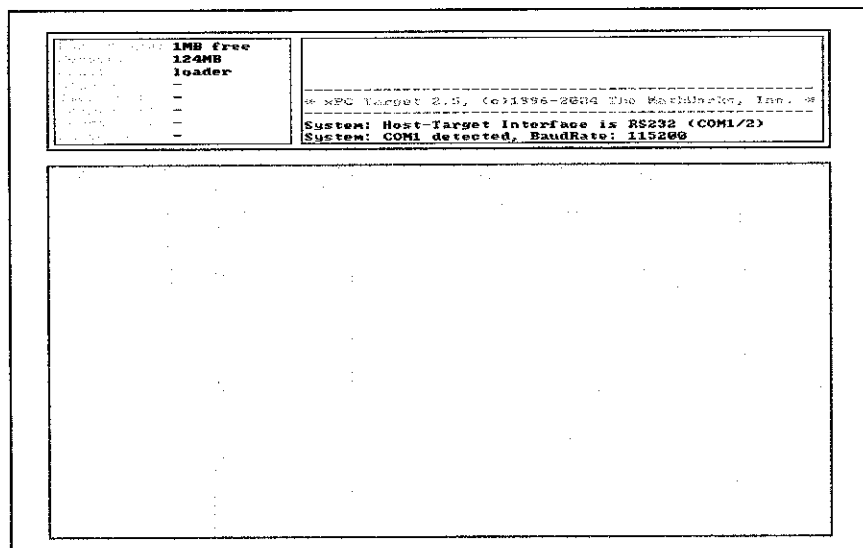


FIGURE 21: Boot of xPC TargetBox in process

As the process finished, the xPC TargetBox is then ready to accept command from host PC via the network communication.

### 3.2.5.4 Testing the installation

xPC Target uses a test script to test the entire installation. This test is used to troubleshoot connection and communication problem between host PC and target PC. This test checks both the host computer xPC Target setup and the xPC TargetBox by building, downloading, and running a simple test Simulink model.

The following were the sequence of procedures taken for this process:

1. Target boot disk was inserted into the external disk drive
2. The Reset button on the xPC TargetBox was pressed. After loading the BIOS, xPC Target boots the kernel and displays the following screen on the xPC TargetBox monitor.

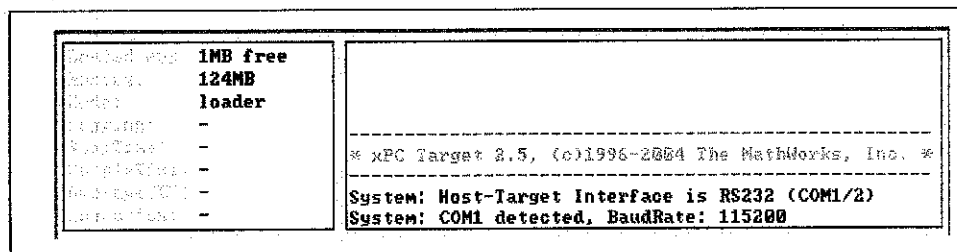


FIGURE 22: testing the installation

3. In the MATLAB command window, a command “xpctest” was typed. MATLAB runs the test script and displays messages indicating the success or failure of a test.

Since all the subtests were successful, this indicates that the host PC and target PC were properly set up the regular use.

### 3.2.5.5 Entering Real-Time Workshop parameters

Before creating a target application, there are several steps that need to be taken. This includes the creation of simulink model which have been completed during fyp1. xPC Target uses the Simulink model, Real-Time Workshop, and a third-party compiler to create the target application. In order to prepare the simulink model for the creation of target application, several procedures have been taken which were:

- a) **Adding a Simulink Outport block** - Add an Outport block to log signal data to the MATLAB workspace for analysis.
- b) **Entering Parameters for the Outport Blocks** -- Enter and select parameters in the Configuration Parameters dialog box.
- c) **Adding an xPC Target Scope Block** -- Add an xPC Target Scope block to visualize signals while running the target application.
- d) **Entering Parameters for an xPC Target Scope Block** -- Enter scope parameters in the Block Parameters dialog box before building the target application

The above procedures involving a simple modification on the simulink model in order to prepare it for the creation of target application. Below is the result of the modification:

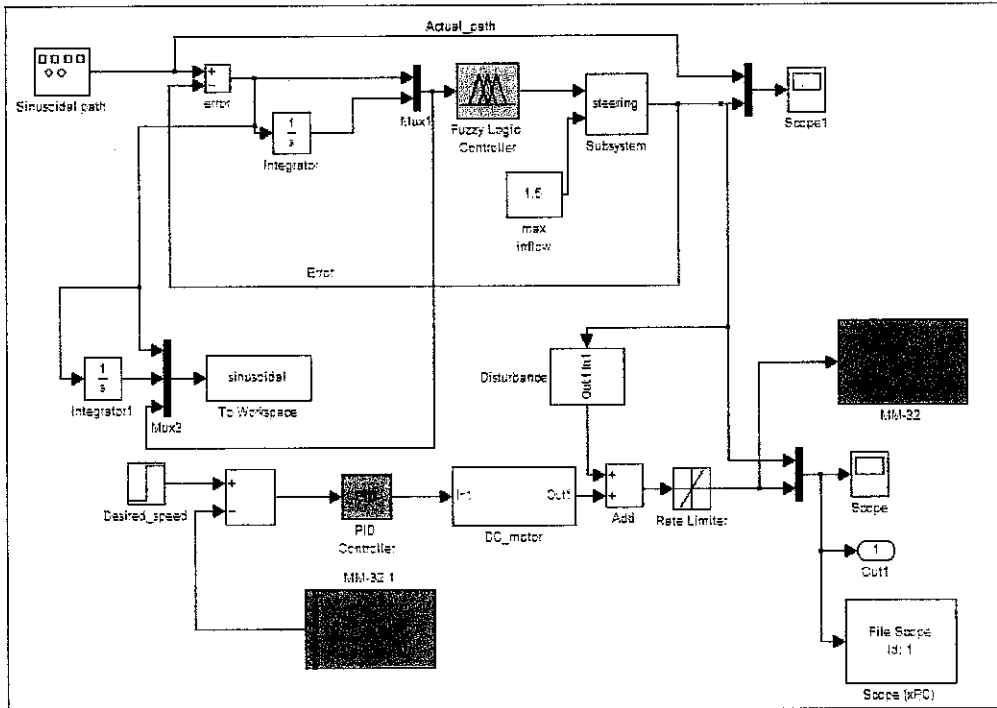


FIGURE 23: The modified simulink model

After loading the simulink model and boot the xPC TargetBox, the simulation parameter then could be entered. The following were the steps taken for this process:

1. In the Simulink window, and from the Simulation menu, Configuration Parameters was clicked. The Configuration Parameters dialog box is displayed for the model.
2. The Real-Time Workshop node was clicked. The Real-Time Workshop pane opened.
3. In the Target selection section, the Browse button at the RTW system target file list was clicked. xpctarget.tlc was chosen. The system target file xpctarget.tlc, the template makefile xpc\_default\_tmf, and the make command make\_rtw were automatically entered into the page. The xPC Target options node appeared in the left pane. The Real-Time Workshop pane looked like the figure shown.

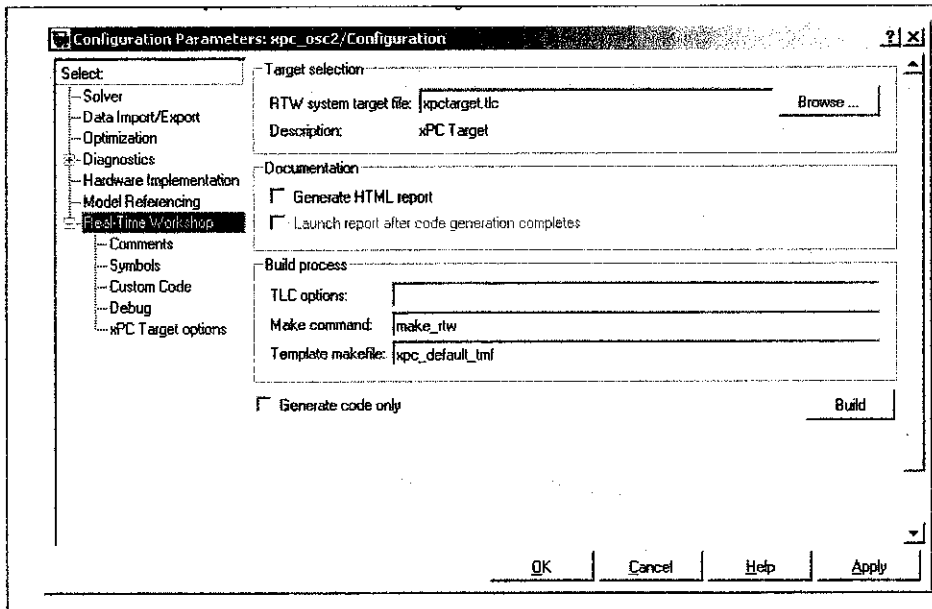


FIGURE 24: Configuration parameters window for real-time workshop node

4. In the left pane, the xPC Target options node was clicked. The associated pane was displayed.
5. The configurations were then set according to the below settings.

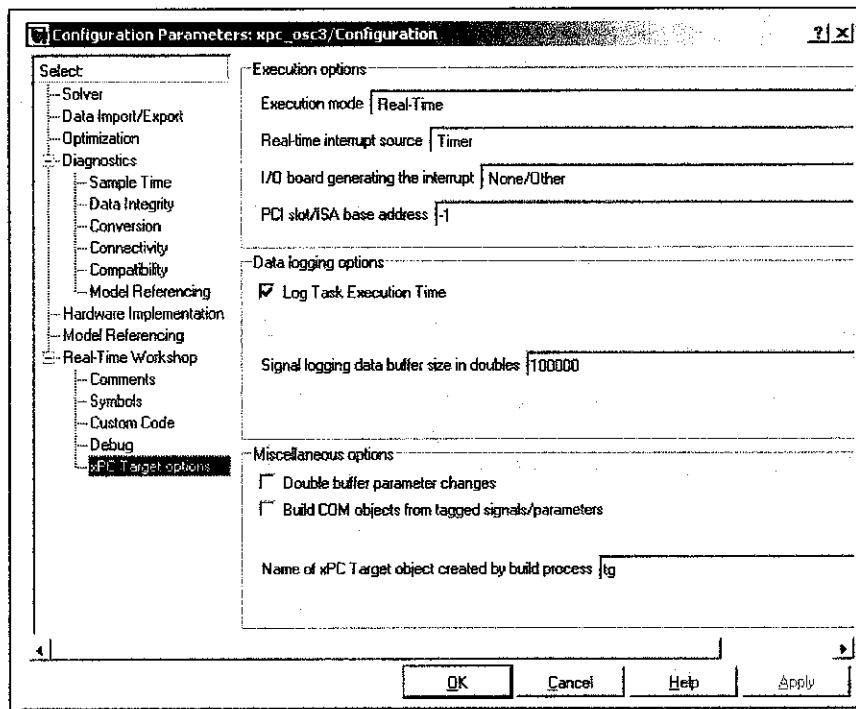


FIGURE 25: Configuration parameters window for xPC Target options node



Finally, the OK button was clicked to finish up the process. The next step after this process is to create (build) the target application.

### 3.2.5.6 Building a Target application

Once the set up is completed, the model is built and downloaded to the xPC Target by the following steps shown in figure below:

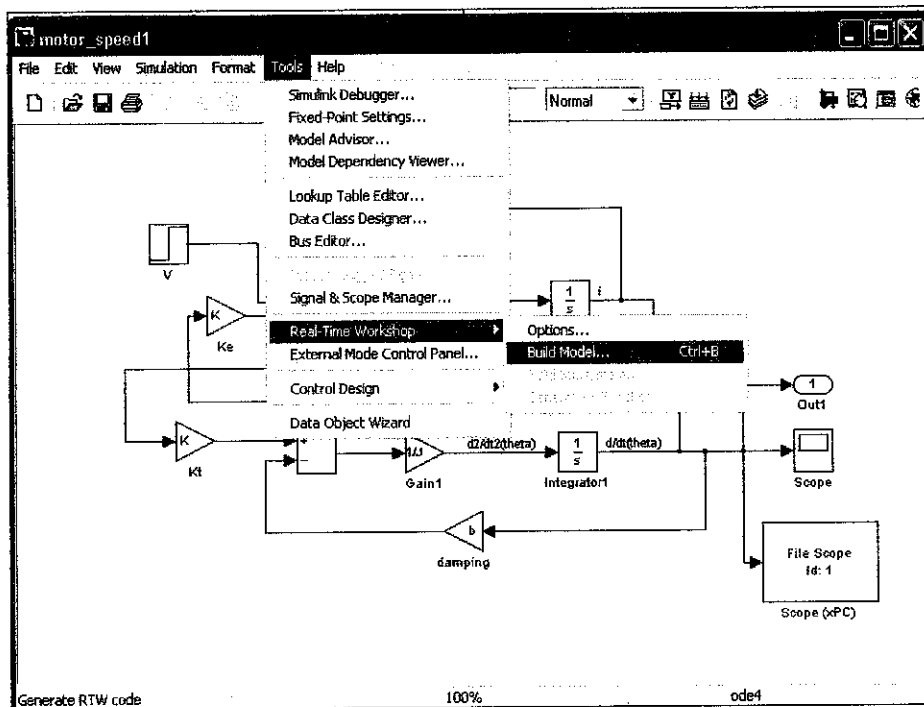


FIGURE 26: The model is built and downloaded into the xPC Target

Once the model is built, the diagram is set to run in “External” mode so that the diagram will interface with I/O hardware.

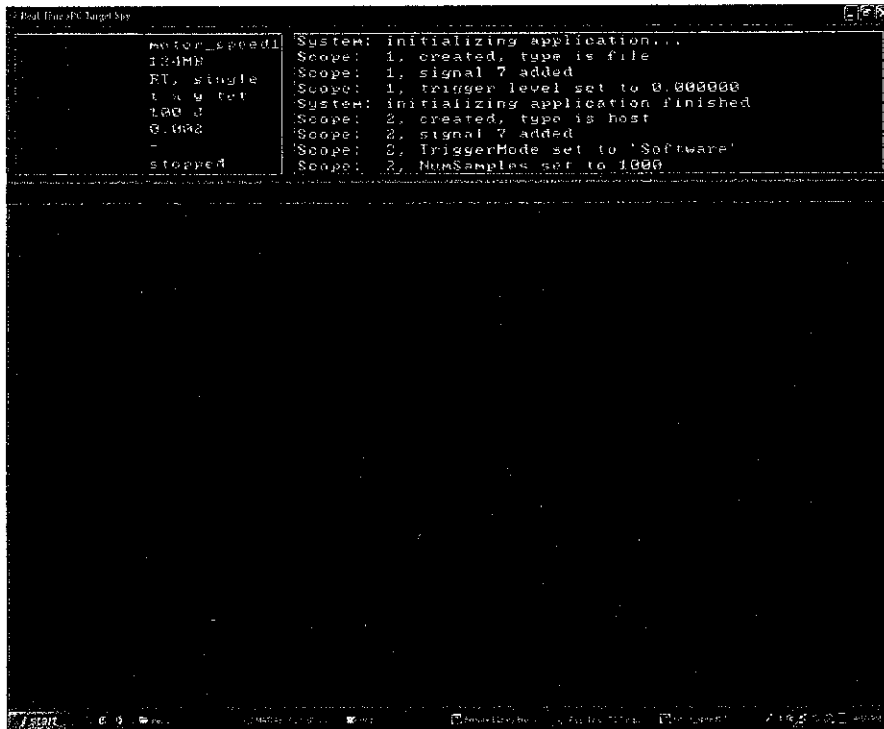


FIGURE 27: Real-Time xPC Target Spy showing the model is built

The target application can be controlled via the xPC Explorer. From the xPC explorer, the target can be connected to the host PC, run the simulation and stop the simulation.

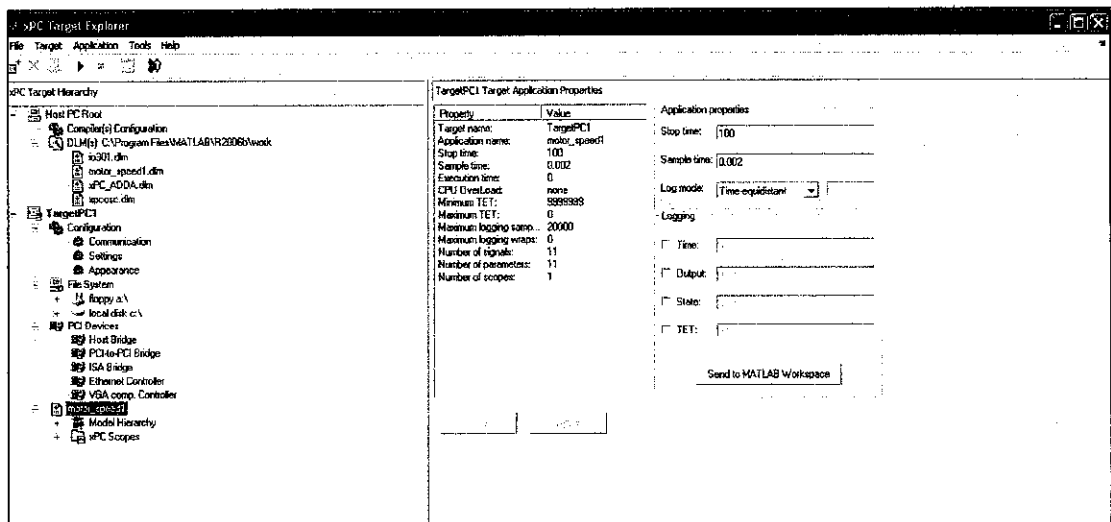


FIGURE 28: xPC Target Explorer windows

### 3.2.5.7 Controlling the Target Application

During the build process, xPC target creates a target object that represents the target application running on the xPC TargetBox. The target object is defined by a set of properties and associated methods. The target application and computer can be controlled by setting the target properties.

The selected way of controlling the target application is through xPC Explorer. From the xPC Explorer, the target can be connected to the host pc, downloaded to the target PC, run and stop the simulation and monitoring the signals.

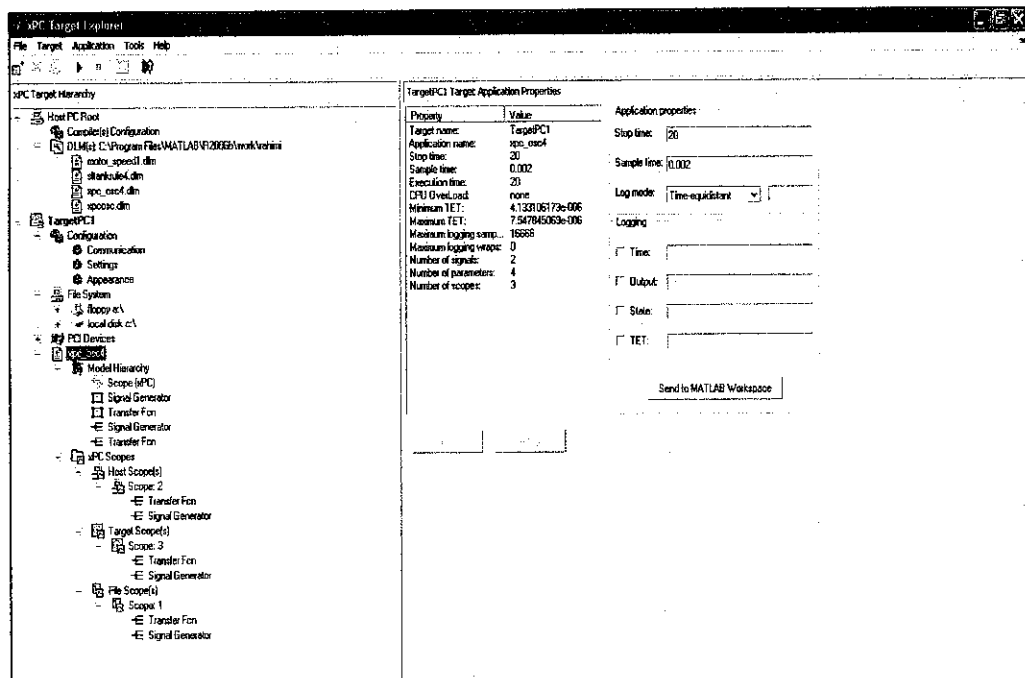


FIGURE 29: xPc Target Explorer window

### 3.2.5.8 Signal Tracing

Basically, signal tracing is the process of acquiring and visualizing signals while running a target application. In the most basic sense, this process allows the user to acquire signal data and visualize it on the target PC or upload the signal data and visualize it on the host PC while the target application is running. There are four ways of performing signal tracing which are through:

- a) xPC Target Explorer
- b) MATLAB
- c) xPC Target scope blocks
- d) Web browser

For the project, the selected way to perform signal tracing is through xPC Target Explorer. This is because it provides easier and visualized control of the signal tracing. The method allows the use of xPC Target GUI (scope dialog boxes) to create and run scopes that are displayed on the host PC. To perform this method, there are sequences of procedures that need to be followed which are:

- a) Creating the scopes
- b) Adding signals to scopes
- c) Stopping scopes

All those procedures are performed after the target application has been built and downloaded to the xPC TargetBox.

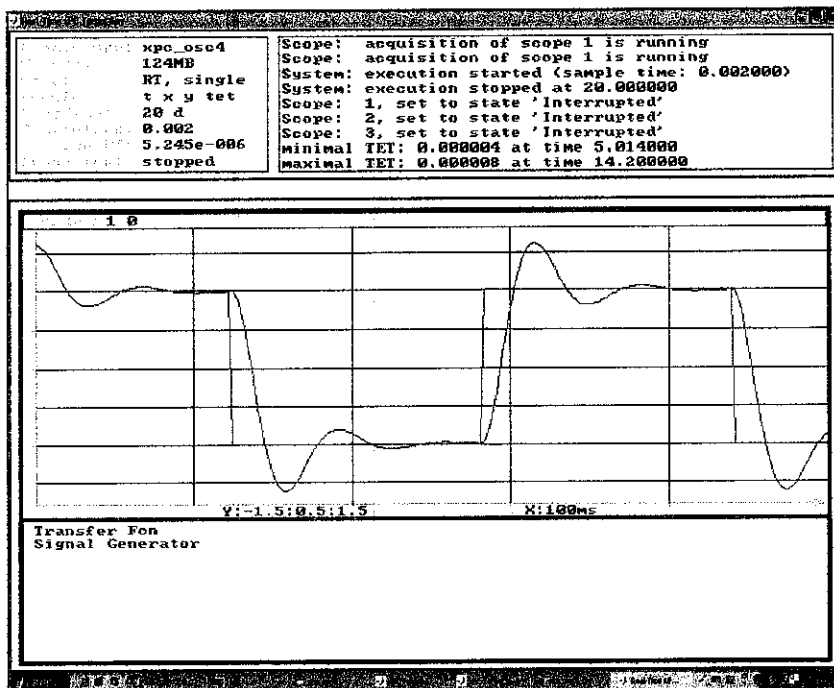


FIGURE 30: Target scope is running



FIGURE 31: Host scope is running

### 3.2.5.9 Target PC and Hardware Configuration

Since the speed control mechanism is carried out in closed loop control, there will a feedback component that used to measure and supply the output speed. The output speed is then compared with the desired speed to get the error. The controller will then control the motor to reduce the amount of error.

For hardware implementation, the DC motor used will drive a generator to produce the corresponding voltage to be fed back to the system. In between the DC motor and the xPC TargetBox, there will be a power amplifier to drive the motor according to the input given. The power amplifier is connected to the xPC TargetBox through the analog output channel while the generator output is connected to the analog input channel. Below is the configuration of the system used to run the rapid control prototyping.

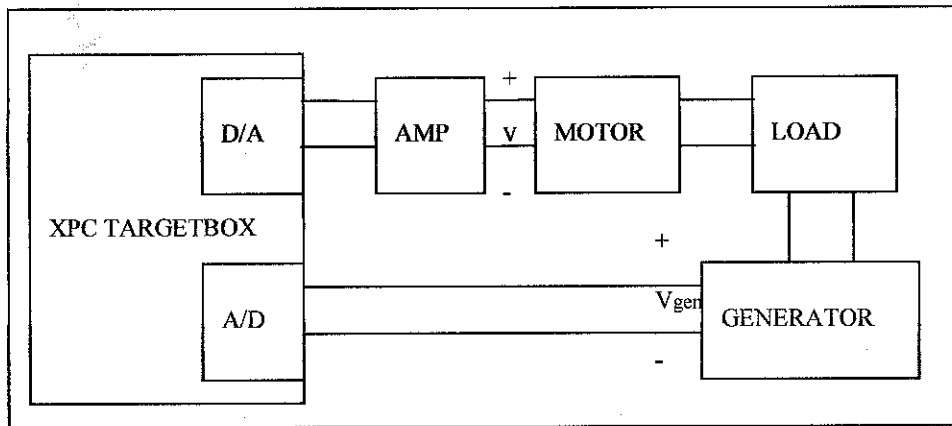


FIGURE 32: Connection between hardware and xPC TargetBox

In order to implement the real time rapid prototyping with generator application, there is several modifications that were done on the simulink model. A block of generator with several driver blocks are used in which correspond to the specific I/O hardware board of the xPC TargetBox. The generator will be connected to the DC motor to produce a voltage corresponding to the speed produced by the DC motor. The generator will then feed the xPC TargetBox the current speed so that it will be calculated and processed through the controller in software environment.

Special blocks are used to produce output voltage from xPC TargetBox. Those blocks are selected based on I/O board used for the project which is DIAMOND-MM-32. Those driver blocks are important in order to get the relationship between the hardware part and the software part. The drivers used include:

- a) Diamond-MM-32-AT Analog Input (A/D, IO 301)
- b) Diamond-MM-32-AT Analog Output (D/A, IO 301)

The analog voltage input and output functions on the Diamond MM-32 board are used to control the motor and read the generator voltage, respectively. For the parameters of the driver blocks, there are several settings that need to be done in order to get them working properly. Some of the parameters that need to be considered for the sake of the hardware requirements are the channel vector

(channels used), range vector, sample time (set to match the fixed-step size) and also the base address (as set in hardware).

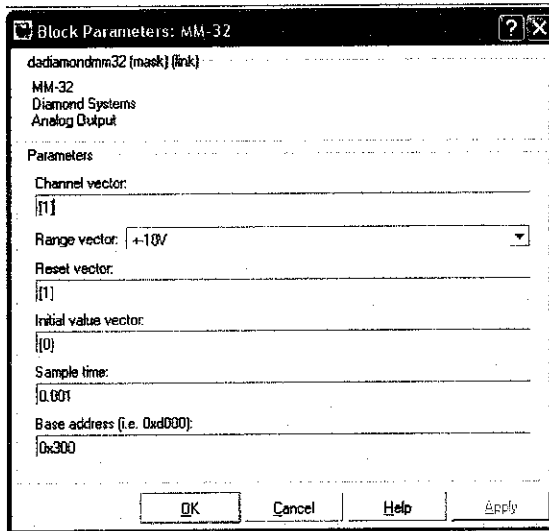


FIGURE 33: Block parameter for analog output

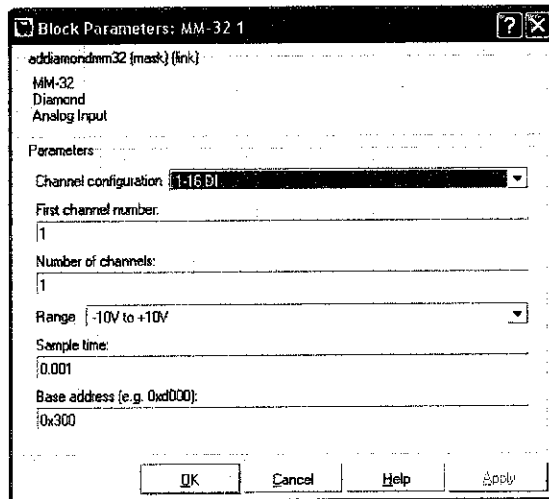


FIGURE 34: Block parameter for analog input

## **CHAPTER 4**

### **RESULTS AND DISCUSSION**

The main objective of the project is to design a speed control mechanism with the ability to follow a predefined path with the smallest error. The resulted design is being applied for simulation only during first semester. The simulink model is then being applied for real-time application with the application of DC motor and xPC TargetBox.

Since rapid prototyping is chosen for the project, there is no real controller being applied for the project. There is a prototype controller running in real-time on an xPC TargetBox connected to a real plant. The hardware application is introduced to the speed control only. The movement of the DC motor represents the speed of the linear vehicle while following the path.

Some of variables have been defined in the previous projects which include the construction of the training path. The paths were designed on MATLAB. Those simulated paths would be implemented to the control system in order to examine the functional of the controller in following the injected path as close as possible. There are two types of paths defined which are: sinus shape and sudden change of direction.

#### **4.1 Simulink model**

##### **4.1.1 DC Motor**

Since the main concern of the project is to implement the xPC Targetbox, so the only thing that will be used to represent vehicle is the DC motor not the car model



defined previous project. The DC motor is represented in form of transfer function in MATLAB simulink which is represented as below:

$$\frac{\dot{\theta}}{V} = \frac{K}{(Js + b)(Ls + R) + K^2} \quad (9)$$

After obtaining the transfer function of the DC motor (plant), then, proceed with creating the simulink model for the DC motor speed control using lag compensator. In order to create the model, several equations need to be reviewed. For the motor torque,  $T$ , it is related to the armature current,  $i$ , by a constant factor  $K_t$  and the back emf,  $e$ , is related to the rotational velocity by the following equations:

$$\begin{aligned} T &= K_t i \\ e &= K_e \frac{d\theta}{dt} \end{aligned} \quad (10)$$

In building the block, the system would be modelled by summing the torque acting on the rotor inertia and integrating the acceleration to give the velocity. In addition, Kirchhoff's law would be applied to the armature circuit. In modelling the system, the first thing to do is to model the integrals of the rotational acceleration and of the rate of change of armature current.

$$\begin{aligned} \int \frac{d^2\theta}{dt^2} &= \frac{d\theta}{dt} \\ \int \frac{di}{dt} &= i \end{aligned} \quad (11)$$

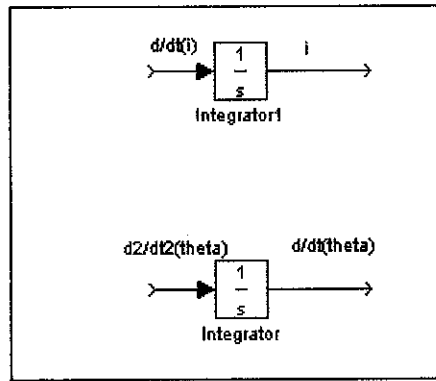


FIGURE 35: Integrator blocks for the integral and rate of change

After that, the modelling work is started by applying the following equations:

$$\begin{aligned}
 J \frac{d^2\theta}{dt^2} &= T - b \frac{d\theta}{dt} \implies \frac{d^2\theta}{dt^2} = \frac{1}{J} \left( K_t i - b \frac{d\theta}{dt} \right) \\
 L \frac{di}{dt} &= -Ri + V - e \implies \frac{di}{dt} = \frac{1}{L} \left( -Ri + V - K_e \frac{d\theta}{dt} \right)
 \end{aligned}
 \tag{11}$$

By using the simulink block provided by MATLAB, below is the resultant system for the DC motor.

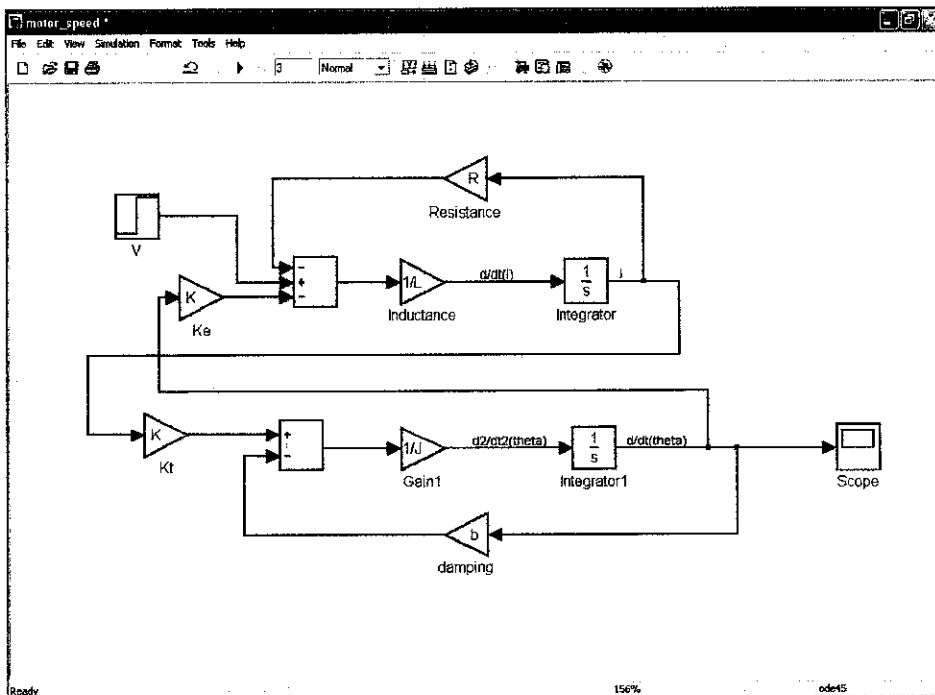


FIGURE 36: DC motor simulink model

A step input is applied to observe the operation of the operation of DC motor. A sinusoidal input is not suitable for the DC motor since it needs a direct current input. The resultant waveform is shown in the next figure.

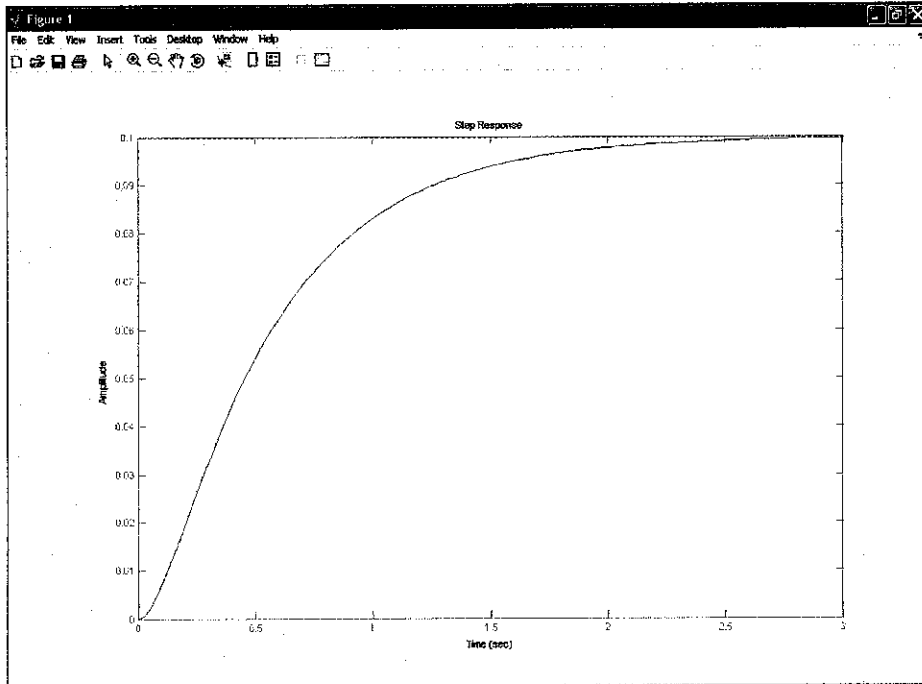


FIGURE 37: Waveform for the output of the model

#### 4.1.2 Direction control

For the direction simulink model, a block named as limited integrator is used as the main part of the system. It is a closed loop system which gets feed from the controller. The output of the system will then be multiplied with the source flow (amount of steering turn) which is set to 1.5 maximum. Limited integrator is used in order not to saturate the auxiliary steering actuator which is set earlier. This low frequency limited integrator implementation also allows the driver to take care of low frequency steering and disturbance rejection tasks.

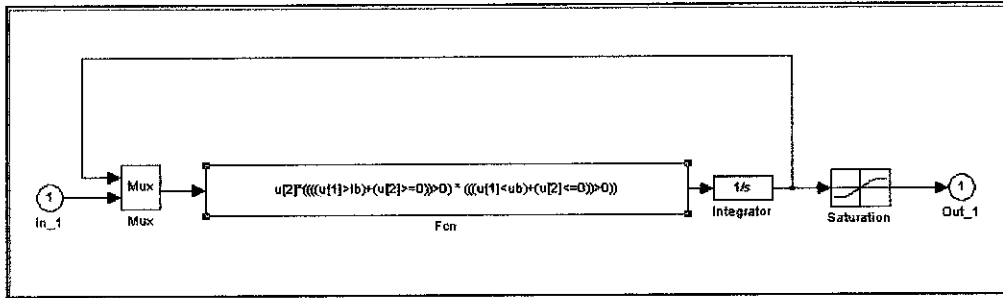


FIGURE 38: Limited integrator subsystem model

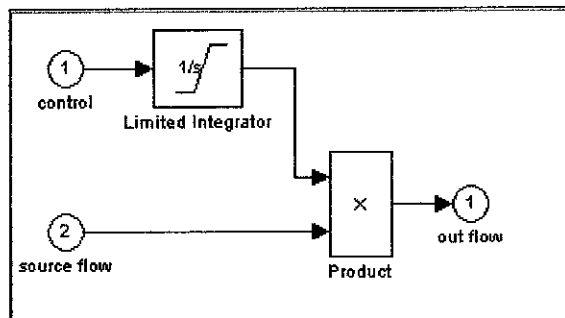


FIGURE 39: Steering control subsystem model

Together with the speed control mechanism, both blocks are combined to create a full system that can control both the speed and direction. With the introduction of PID controller for the speed control and fuzzy logic for the direction control, both systems are connected and get input from a signal generator to represent the path.

The output from the direction control will be subjected to the disturbance for speed control system. The speed control system which is set to 5 units will control the speed in reflect to the disturbance. When there is a corner, the direction control system will lead the steering accordingly. The result of the steering change will then reduce the speed of the motor.

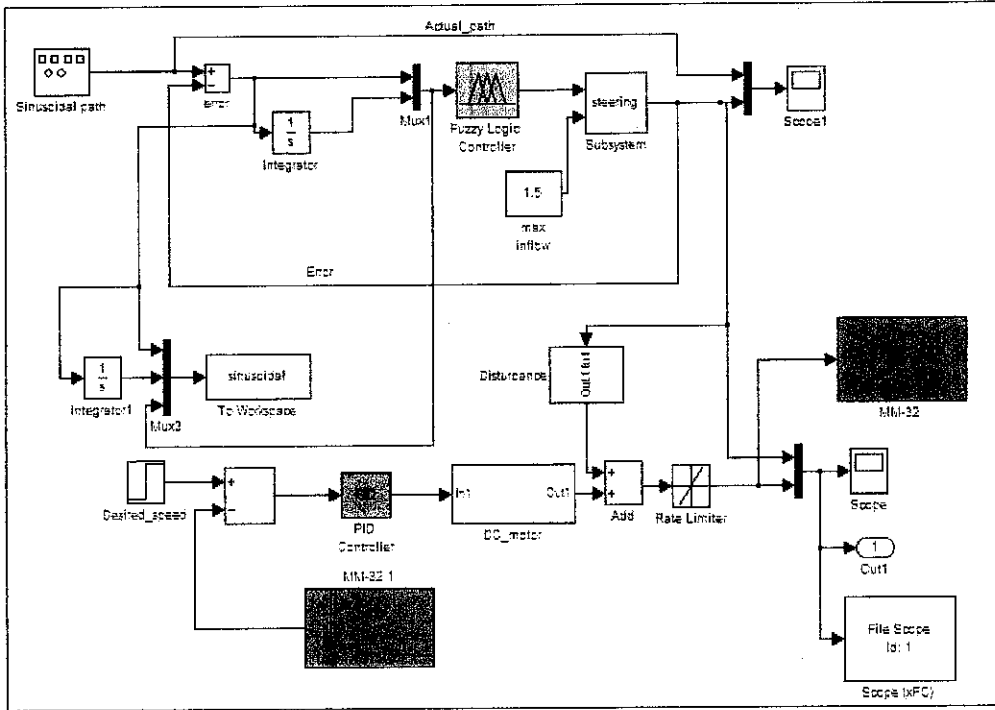


FIGURE 40: Simulink model

## 4.2 Controllers

### 4.2.1 Speed control

For the controller designing step, a lag compensator was chosen with the following transfer function.

$$50 \frac{s + 1}{s + 0.01} \quad (12)$$

By using the simulink block available in the MATLAB features, the following is simulink model for the DC motor speed control using the lag compensator.

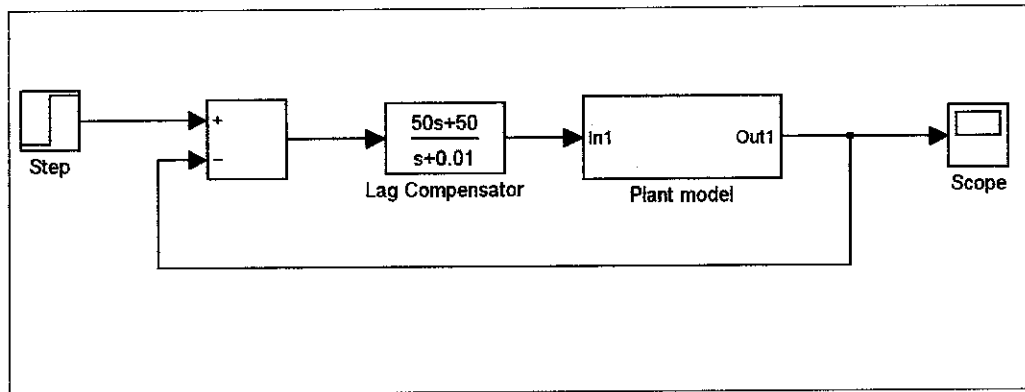


FIGURE 41: DC motor speed control simulink block

#### 4.2.2 Direction control

In designing the fuzzy logic controller, MATLAB toolbox is used as the platform for the controller design. The input paths are divided into two parts which are error and integral which have five rules and conditions. Direction is chosen as the output of the controller and defined in term of nine.

The rules for the system need to be defined in the rule editor in which determining the succession of the controller. The rules decided for the controller is as follows:

1. If Error is negative and integral is small then direction is very very left
2. If Error is negative and integral is medium then direction is very left
3. If Error is negative and integral is large then direction is left
4. If Error is okay and integral is small then direction is little left
5. If Error is okay and integral is medium then direction is okay
6. If Error is okay and integral is large then direction is little right
7. If Error is positive and integral is small then direction is right
8. If Error is positive and integral is medium then direction is very right
9. If Error is positive and integral is large then direction is very very right

The controller is designed based on those nine rules defined earlier. With two types of input: error and integral, the rules will subject the control to the output which is the direction.

### 4.3 Simulation result

#### 4.3.1 Speed control

The resultant output of the DC motor speed control system:

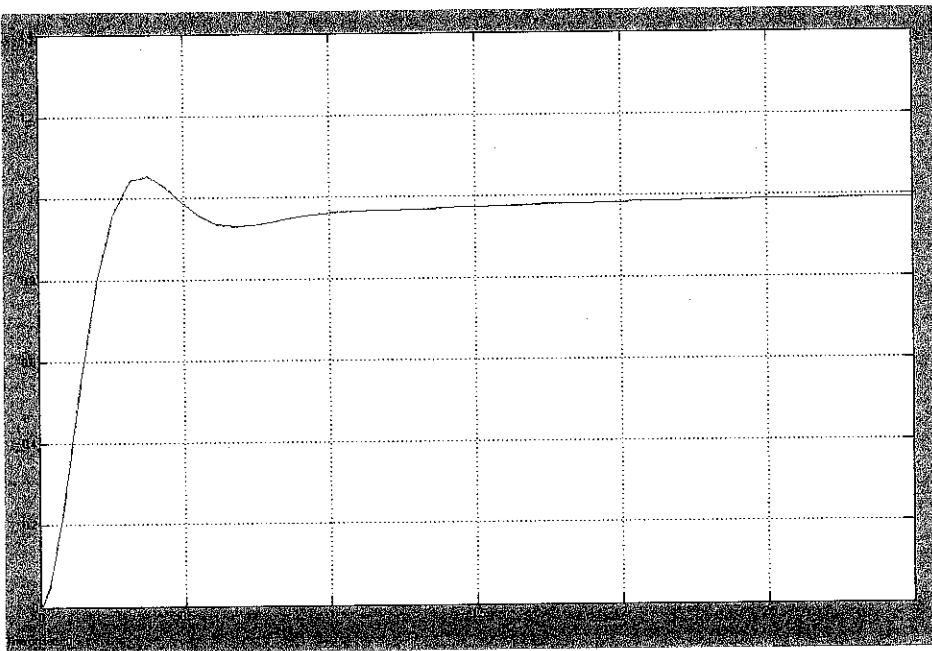


FIGURE 42: The step input result for dc motor speed control

From the result obtained; the resultant motor speed control produced the desired overshoot and steady state error value. The compensator used will be extended for the input path and also for direction control through the implementation of fuzzy logic.

### 4.3.2 Sinusoidal path

The path is applied in order to verify the functionality of the controller. The duration of the testing is set to be 30s. Based on the result, the direction control can follow the path and the error is acceptable.

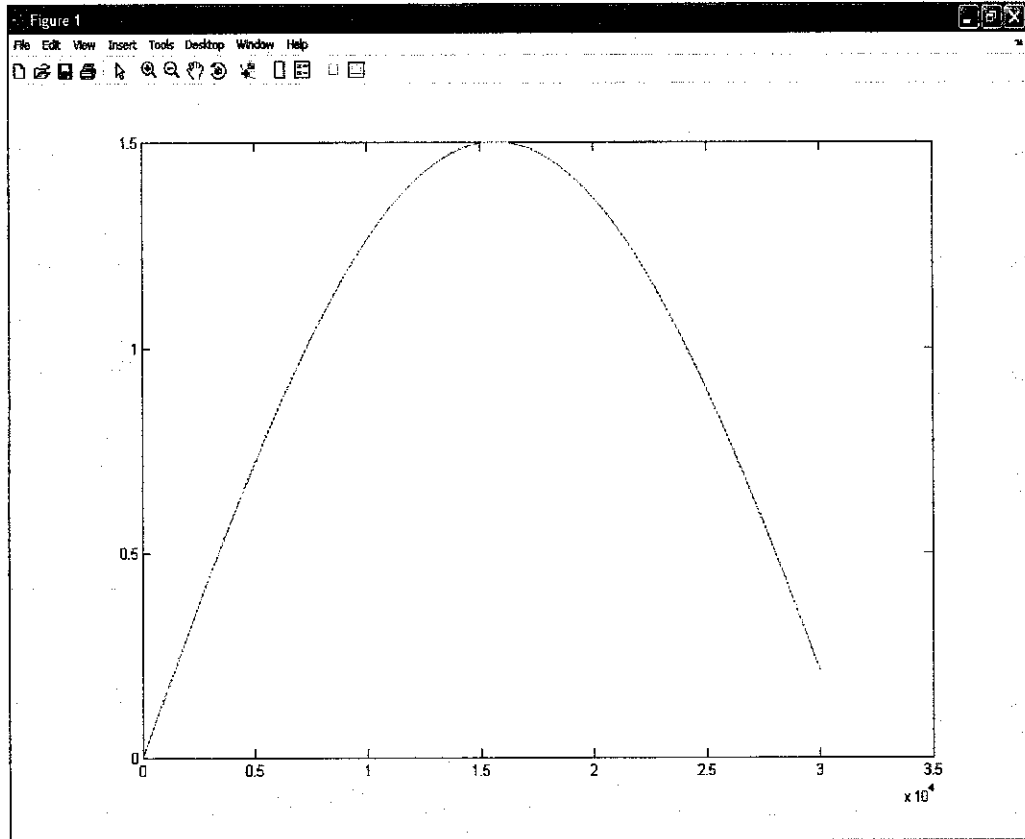


FIGURE 43: sinus path result for steering

From the figure 43, the green line is the output of the direction control while the blue one is the desired output or the input path. As shown, the steering should response to the applied path as soon as is it generated. So, it is shown that the system is able to follow the input path with small error.



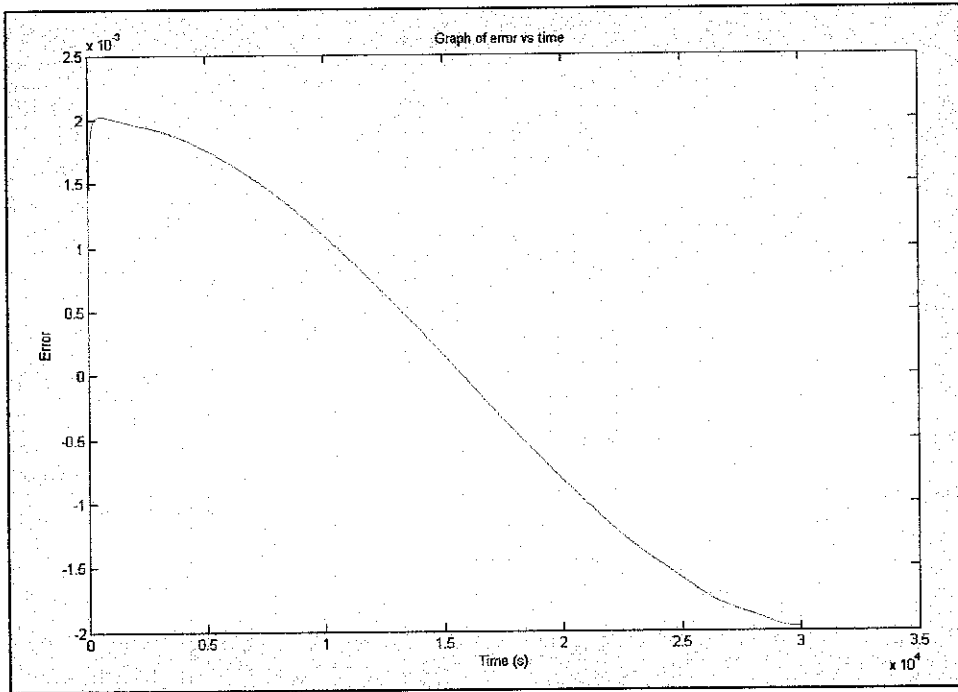


FIGURE 44: Error for direction control

From the graph 44, the error produces from the path following mechanism is very low. The amount of the maximum error is about 0.25% and it is reducing over time.

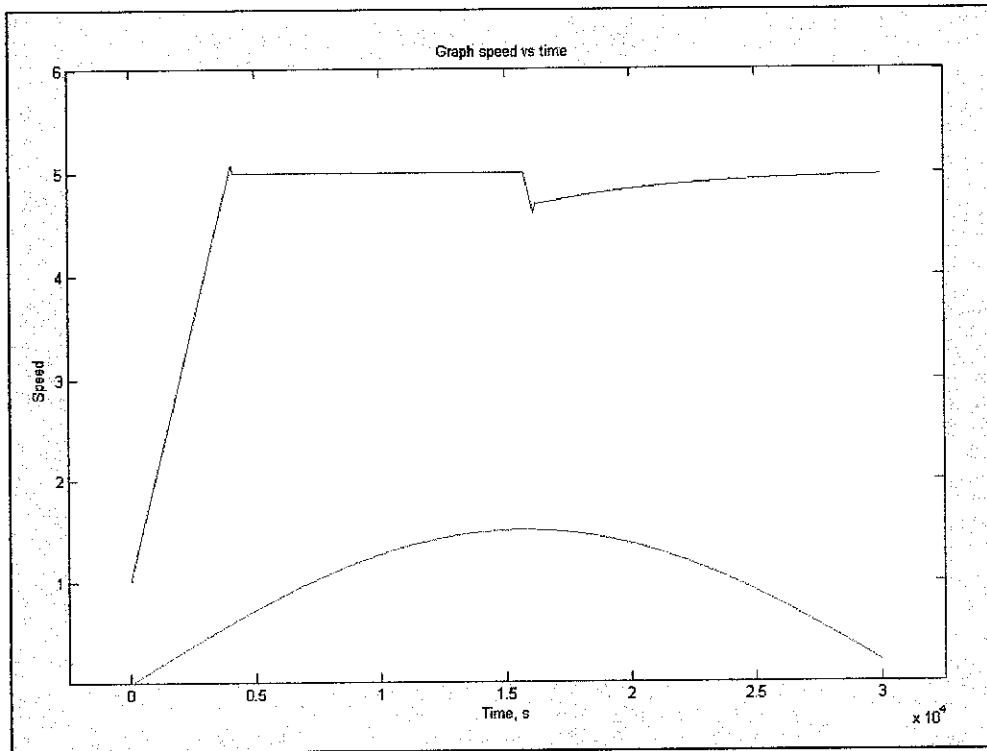


FIGURE 45: Speed response

From the response shown in figure 45, the direction response is represented by the green line, while the corresponding speed response is represented by red line. The speed is reduced according to the shape of the path. At the turning point of the path, the speed is reduced to a lower speed of about 10% lower than the constant speed. The system manages to reduce the speed before the turning point which is about 3 to 4 second before the vehicle takes the corner (in real life condition).

#### 4.3.3` Sudden change path

This type of path contains two turning points which would examine the capability of the controller to follow the path. The duration of the path is set to 30s which is the same as sinusoidal path. From the result obtained, the system manages to follow the path with an acceptable amount of error.

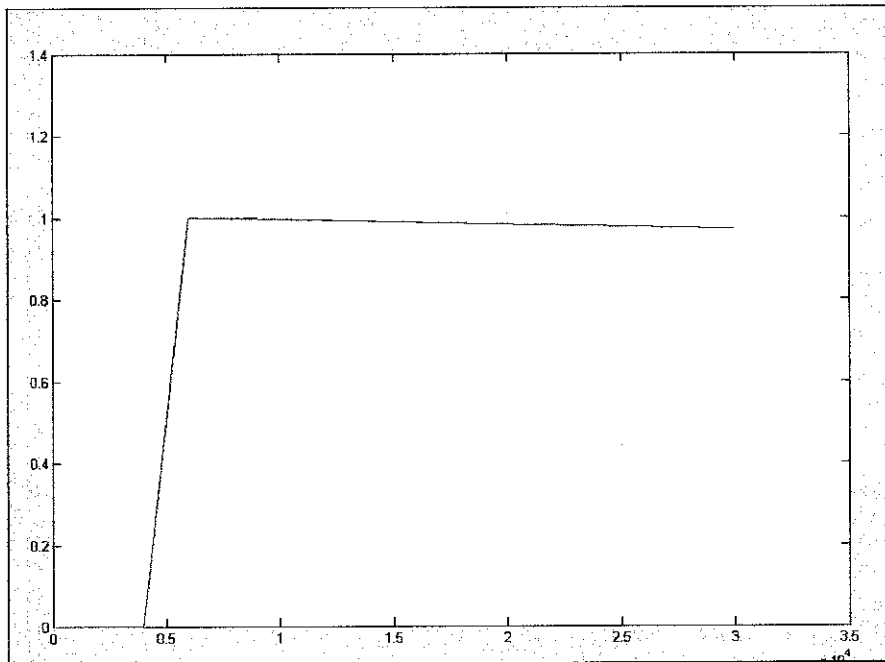


FIGURE 46: Sudden change path direction response

From figure 46, the red line is the actual path while the green line is the output of the direction control. As shown in the figure, the output of the direction control follows the shape of the input path as accurate as possible.

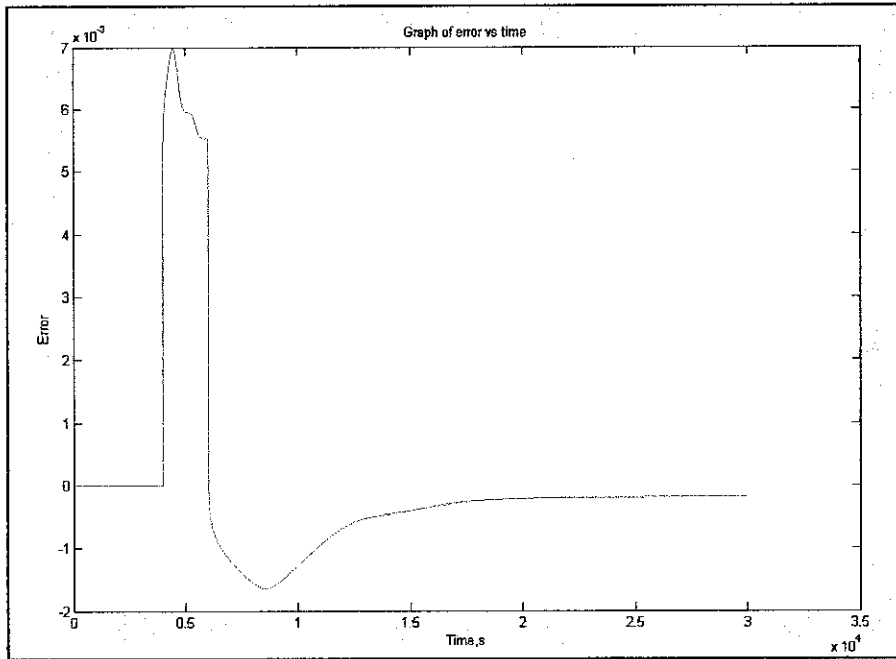


FIGURE 47: Graph of error vs. time

Based on the figure 47, the amount of error produced in direction control reduces as the time increases. The maximum amount of error produced for this type of path is 0.007 which is near to zero. So, the system can be said to have successfully follow the path with an acceptable amount of error.

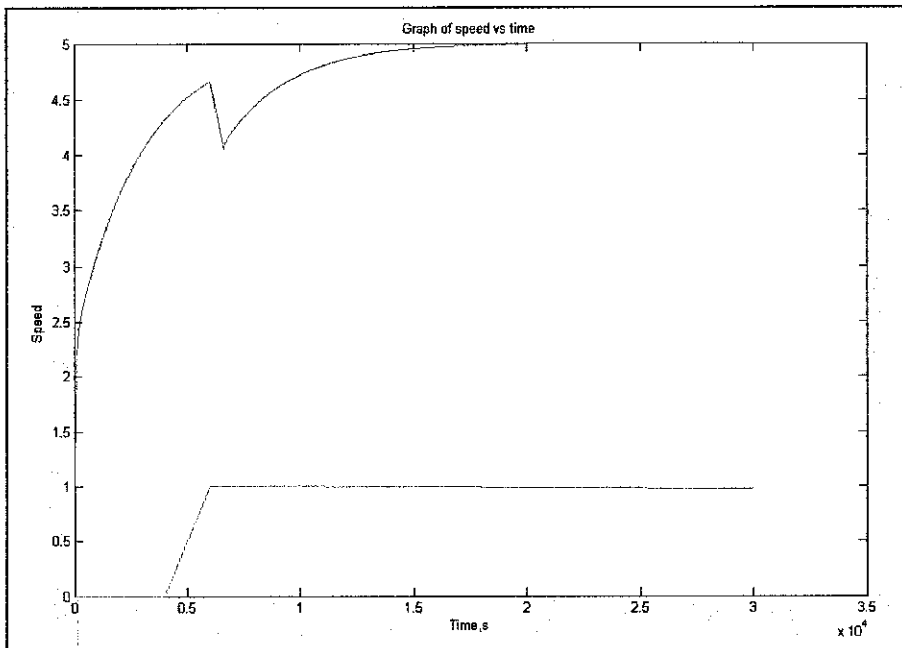


FIGURE 48: Graph of speed vs. time

Based on figure 48, the red line represents the output response from the speed control while the green line is the output from the direction control. The speed is set to a constant value. The system is expected to control the speed according to the disturbance from the direction control. During the first corner, the speed is still low, so there is no dropping of speed occurred. For the second corner, the speed reduction is about 10% from the previous speed.

#### **4.4 Running the hardware application**

From the testing, the simulink model successfully produces an analog voltage ( $\pm 10V$ ) through the screw terminal board. Since there is only one channel set for the screw terminal board, so only pin 36 produces the analog voltage output. However, this type of output could not be used to drive the DC motor. This is due to the low amount of current produced by the I/O board. The amount of current produced is approximately 5 mA, out of expectation from the DC motor that requires about 1.6 Ampere of current. The low amount of current occurred because the signal is directly converted from digital medium to the analogue medium.

So by constructing a unity gain buffer amplifier, the current from the xPC TargetBox can be amplified to the required value needed by the motor. Since the current is enough to drive the motor and the generator, then the analogue input block can be used as the feedback to the system by connecting it to the generator.

The motor is expected to run at a constant speed as soon as the system is turned on. When there is a disturbance from the direction control, the speed would reduce to a lower speed corresponding to the output response from the speed control system. When the disturbance is gone, the speed would come back to the constant speed set earlier.

## **CHAPTER 5**

### **CONCLUSION AND RECOMMENDATION**

#### **5.1 Conclusion**

The project examines the capability of fuzzy logic and PID controller in constituting the controller for the linear vehicle. Both methods have been reviewed in order to develop a controller for both steering and speed of the vehicle. The linear vehicle is represented by a DC motor and xPC Targetbox will be used to apply the system in real life condition. The input for the system is a sinusoidal path and sudden change path. Those paths have been selected in order to examine the successful rate of the vehicle achieve the objective. The successful rate is defined on how the vehicle follows the path as precise as possible and able to vary the speed according to the pattern of the path with small error. The only concern is to develop the controllers for the linear vehicle to control both the speed and steering. An amplifier is developed in order to amplify the current out from the xPC targetBox to drive the motor.

## **5.2 Recommendation**

There is a room of improvements that could be applied to the designed control system. The direction control simulink model is not a complete model since the controller only controls the steering to follow the path. A lot of justifications and calculations need to be carried out to get a better simulink model especially for direction control. The integration between the hardware and software also needs further working to get the best out of real time application. Below are several suggestions that could be carried out in order to improve the project:

1. Introduction of linear car model to the direction control system. This could further improve the system for real time application
2. Application of optimal preview method and optimal controller in order to amplify the ability of the direction control to follow the path. The optimal preview could help the system by portraying the driver's vision.
3. Improve the relationship between the direction and speed control. The relationship needs to be added more features and considerations along with the application of optimal preview.
4. More input paths need to be applied to the system in order to increase the flexibility of the system.
5. More hardware application should be implemented such as for direction control. Other hardware than DC motor should be considered for hardware application.

## REFERENCES

- [1] Mohamad Hanif, NHH 2003, Employing Batch Training Through Neural Network for Path Following, MSc. Thesis, Imperial College London.
- [2] Dandre, P. 2003, Learning Path Following Control of An Automobile, MSc. Thesis, Imperial College London
- [3] Nor Hanisah Baharuddin, 2005, Neural Network Controller for High Speed Vehicle Following Predetermined Path. Final Year Project, Electrical & Electronics Dept., Univ. Teknologi PETRONAS.
- [4] Taquiuddin Mohd Nasir 2006, Modelling and simulation of series parallel hybrid electrical vehicle. Final Year Project, Electrical & Electronics Dept, Univ. Teknologi PETRONAS
- [5] Citing Internet sources URL  
<http://auto.howstuffworks.com/automatic-transmission8.htm>
- [6] Citing Internet Sources URL  
[http://en.wikipedia.org/wiki/PID\\_controller](http://en.wikipedia.org/wiki/PID_controller)
- [7] Citing Internet Sources URL  
<http://www.answers.com/fuzzy%20logic>
- [8] Citing Internet Sources URL  
<http://www.mathworks.com/products/fuzzylogic/>
- [9] Stephen J.Chapman 2005. *Electric Machinery Fundamentals*. BAE SYSTEMS: Australia

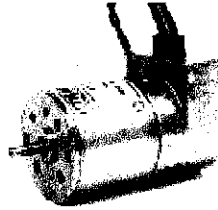
# APPENDICES

## Appendix 1: DC Motor datasheet



9232S003

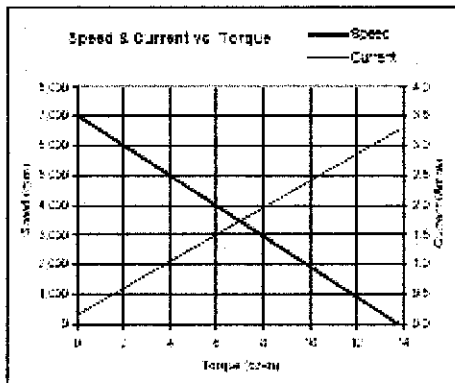
Lo-Cog® DC Servo Motor



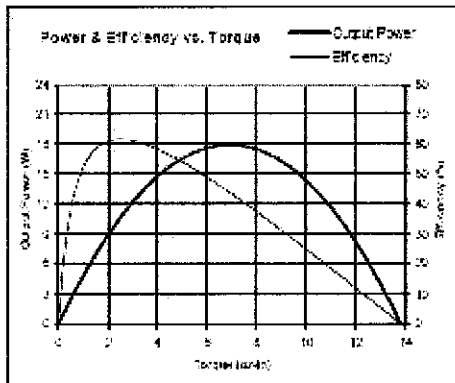
Assembly Data	Symbol	Units	Value
Reference Voltage	E	V	24
No-Load Speed	$\omega_{NL}$	rpm (rad/s)	7,016 (336)
Continuous Torque (Max.) <sup>1</sup>	$T_C$	oz-in (Nm)	2.4 (0.17E-01)
Peak Torque (Max.) <sup>2</sup>	$T_{pk}$	oz-in (Nm)	14 (0.97E-01)
Weight	$W_M$	oz (g)	10 (288)
Motor Data			
Torque Constant	$K_t$	oz-in/A (Nm/A)	4.40 (0.11E-02)
Back-EMF Constant	$K_b$	V/rpm (V/rad/s)	3.35 (0.11E-02)
Resistance	R	$\Omega$	2.38
Inductance	L	mH	4.64
No-Load Current	$I_{NL}$	A	0.15
Peak Current (Max.) <sup>2</sup>	$I_p$	A	3.25
Motor Constant	$K_M$	oz-in/A (Nm/A)	1.62 (0.14E-02)
Friction Torque	$T_f$	oz-in (Nm)	0.50 (0.00E-03)
Rotor Inertia	$J_M$	oz-in <sup>2</sup> (kg-m <sup>2</sup> )	2.7E-04 (0.00E-06)
Electrical Time Constant	$\tau_e$	ms	0.68
Mechanical Time Constant	$\tau_M$	ms	14.4
Viscous Damping	D	oz-in/rpm (Nm/s)	0.027 (0.00E-06)
Damping Constant	$K_D$	oz-in/rpm (Nm/s)	1.8 (0.00E-04)
Maximum Winding Temperature	$T_{Max}$	°F (°C)	311 (155)
Thermal Impedance	$R_{th}$	°F/Watt (°C/Watt)	72.5 (22.7)
Thermal Time Constant	$\tau_{th}$	min	7.0
Gearbox Data			
Encoder Data			
Channels			3
Resolution		PPR	500

1 - Speed at max. winding temperature at 25°C, ambient without load & 1.2 - Theoretical values supplied for reference only

- ### Included Features
- 2-Pole Stator
  - Ceramic Magnets
  - Heavy-Gauge Steel Housing
  - Al-Glob. Armature
  - Sticon Steel Laminations
  - Stainless Steel Shaft
  - Copper-Graphite Brushes
  - Diamond-Turned Commutator
  - Motor-Ball Bearings
- ### Customization Options
- Alternate Winding
  - Sealed or Ball Bearings
  - Modified Output Shaft
  - Custom Cable Assembly
  - Special Brushes
  - EMI/RFI Suppression
  - Spur or Planetary Gearbox
  - Special Lubricant
  - Optional Encoder
  - Fail-Safe Brake



All values are nominal. Specifications subject to change without notice. Graphs are shown for reference only.



© 2007 Pittman.



## Appendix 2: M-File for the input path

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% trainingcircuits %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
disp('which type of circuit do you want?')
disp(' 1 circle of 1km lengh')
disp(' 2 sinus shape (default)')
disp(' 3 lane change ')
disp(' 4 sudden change of direction')
disp(' 5 smooth random path')

i=input(' ');
if isempty(i), i=2; disp('Using sinus (default)'), end
T=1;
u=20;

if i==1
% circle circuit
R=250;
xr1=[R:-u*T:-R];
xr2=[-R+u*T:u*T:R];
xref=[xr1 xr2];
yr1=sqrt(R^2-xr1.^2);
yr2=-sqrt(R^2-xr2.^2);
yref=[yr1 yr2];
plot(yref,xref);
end

if i==2
% sinus shape
xref=[0:u*T:900];
yref=50*sin(xref/100);
plot(yref,xref);
end

if i==3
%lane change
xr1=[0:u*T:50-u*T];
xr2=[50:u*T:50+60];
xr3=[110+u*T:u*T:300];
xref=[ xr1 xr2 xr3];
yr1=0*xr1;
yr2=2-2*cos((pi/60)*xr2-50*pi/60);
yr3=4*ones(size(xr3));
yref=[yr1 yr2 yr3];
plot(yref,xref);
end

if i==4
%sudden change of direction
xr1=[0:u*T:60-u*T];
xr2=[60:u*T:200];
xref=[xr1 xr2];
yr1 = 0*xr1;
SIZE_OF_xr2=size(xr2);
yr2 = 0.5*0.25*[1:SIZE_OF_xr2(1,2)]*0.5359';
yref = [yr1 yr2];
plot(yref,xref);
```

```
end
```

```
if i==5  
%smooth random path  
K=900/(u*T)+1;  
xref=[0:u*T:900];  
[Bfilter, Afilter] = butter(5, 0.007);  
roadn=10*rand(K,1);  
roadn = 40*(roadn-5);  
yref = filter(Bfilter, Afilter, roadn)';  
plot(yref,xref);  
end
```

## Appendix 3: DIAMOND-MM-32-AT specification sheet



SPECIFICATIONS	
<b>ANALOG INPUTS</b>	
Number of inputs	32 single-ended, 16 differential, or 16 SE + 8 D; user selectable
A/D resolution	16 bits (1/65,536 of full scale)
Bipolar ranges	$\pm 10\%$ , $\pm 5\%$ , $\pm 2.5\%$ , $\pm 1.25\%$ , all 6.25V
Unipolar ranges	0-10V, 0-5V, 0-2.5V, 0-1.25V
Input bias current	100pA max
Nonlinearity	$\pm 1LSB$ , no missing codes
Conversion rate	200,000 samples/sec, max
Conversion trigger	software trigger, internal power clock, or external TTL signal
AD FIFO	512 samples, programmable threshold
Calibration	Automatic; values stored in EEPROM
<b>ANALOG OUTPUTS</b>	
Number of outputs	4
D/A resolution	12 bits (1/4096 of full scale)
Output ranges	$\pm 5\%$ , $\pm 10\%$ , 0-5V, 0-10V, programmable
Output current	$\pm 5mA$ max per channel
Settling time	6 $\mu$ s max to 0.01%
Relative accuracy	$\pm 1LSB$
Nonlinearity	$\pm 1LSB$ , monotonic
Reset	All channels reset to 0V
Calibration	Automatic; values stored in EEPROM
<b>DIGITAL I/O</b>	
No. of I/O	24 programmable I/O
Input voltage	Logic 0: 0.0V min, 0.8V max Logic 1: 2.0V MIN, 5.0V max
Input current	$\pm 1\mu$ A max
Output voltage	Logic 0: 0.0V min, 0.35V max Logic 1: 2.4V min, 5.0V max
Output current	Logic 0: 64mA max per line Logic 1: 15mA max per line
<b>COUNTER/TIMERS</b>	
A/D Power clock	32-bits (2 42054 counters cascaded)
Clock source	10MHz on-board clock or external signal
General purpose	16-bits (1 42054 counter)
<b>GENERAL</b>	
Power supply	$\pm 5VDC \pm 10\%$ ; 200mA typ
Operating temp.	-40 to +85°C
Weight	5.4oz / 96g

The Diamond-MM-32-AT is the undisputed world leader in PC/104 analog I/O. No other A/D board can match its combination of feature density, configuration flexibility, and advanced technology.

The 32 analog input channels reduce overall system size and cost for high channel count applications. A unique variable input configuration feature lets you configure the inputs for 32 single-ended, 16 differential, or a combination of 16 single-ended and 8 differential.

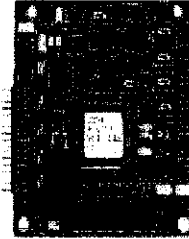
Nine analog input ranges, from a wide  $\pm 10V$  down to 0-1.25V, cover the greatest number of input requirements. A 512-sample FIFO with programmable threshold lets you reach maximum A/D speed without missing samples. With external triggering you can synchronize the A/D converter to external signals and events.

The four analog outputs can be configured in four different fixed output ranges as well as a programmable range anywhere from 1V to 10V with 1mV accuracy. Each output can drive up to 5mA.

The advanced auto-calibration circuit calibrates both the analog inputs and outputs under software control. It provides individual precise adjustments for each analog input range to maximize accuracy across all configurations. Calibration takes just seconds and can be performed as often as desired.

The board contains an integrated 82C55-type digital I/O circuit with 32-bit ports. Each port features configurable direction. The digital I/O lines have user-configurable pull-up / pull-down resistors and latching / handshaking capability. Each output line can sink up to 64mA in logic 0 state or drive up to 15mA in logic 1 state.

A 32-bit counter/timer is provided for programming the A/D sample rate. A second 16-bit counter/timer can be programmed to generate waveforms, count pulses and events, or generate interrupts at programmed rates. The counter clock source can be selected from the on-board 10MHz oscillator or an external signal.



- ◆ 32 analog inputs, 16-bit A/D
- ◆ 200KHz maximum sampling rate
- ◆ Multi-channel scan sampling with interrupts and FIFO support
- ◆ Programmable input ranges
- ◆ Unipolar/bipolar and single-ended/differential inputs
- ◆ 4 analog outputs, 12-bit D/A
- ◆ Multi-range auto-calibration of A/D and D/A
- ◆ 24 digital I/O with latching capability and enhanced output current
- ◆ 512-sample FIFO with programmable threshold
- ◆ Countertimers for A/D control and general use
- ◆ +5V power supply
- ◆ -40 to +85°C operation
- ◆ FREE Universal Driver software included

### ORDERING GUIDE

DM-32-AT 32 16-bit A/D, 200KHz,  
4 12 bit D/A

For cables and accessories, see pages 44-47.

### DIGITAL I/O HEADER

IO	1	2	3	4
IO1	1	2	3	4
IO2	5	6	7	8
IO3	9	10	11	12
IO4	13	14	15	16
IO5	17	18	19	20
IO6	21	22	23	24
IO7	25	26	27	28
IO8	29	30	31	32
IO9	33	34	35	36
IO10	37	38	39	40
IO11	41	42	43	44
IO12	45	46	47	48
IO13	49	50	51	52
IO14	53	54	55	56
IO15	57	58	59	60
IO16	61	62	63	64
IO17	65	66	67	68
IO18	69	70	71	72
IO19	73	74	75	76
IO20	77	78	79	80
IO21	81	82	83	84
IO22	85	86	87	88
IO23	89	90	91	92
IO24	93	94	95	96

### NOTE:

The styling and package with dimensions page 22 with the Diamond-MM-32-AT product also applies to Diamond-MM-32AT.

### ANALOG I/O HEADER

ANALOG	1	2	ANALOG
AN1	1	2	AN32
AN3	3	4	AN30
AN5	5	6	AN28
AN7	7	8	AN26
AN9	9	10	AN24
AN11	11	12	AN22
AN13	13	14	AN20
AN15	15	16	AN18
AN17	17	18	AN16
AN19	19	20	AN14
AN21	21	22	AN12
AN23	23	24	AN10
AN25	25	26	AN8
AN27	27	28	AN6
AN29	29	30	AN4
AN31	31	32	AN2
AN33	33	34	AN0
AN35	35	36	AN0
AN37	37	38	AN0
AN39	39	40	AN0
AN41	41	42	AN0
AN43	43	44	AN0
AN45	45	46	AN0
AN47	47	48	AN0
AN49	49	50	AN0
AN51	51	52	AN0
AN53	53	54	AN0
AN55	55	56	AN0
AN57	57	58	AN0
AN59	59	60	AN0
AN61	61	62	AN0
AN63	63	64	AN0
AN65	65	66	AN0
AN67	67	68	AN0
AN69	69	70	AN0
AN71	71	72	AN0
AN73	73	74	AN0
AN75	75	76	AN0
AN77	77	78	AN0
AN79	79	80	AN0
AN81	81	82	AN0
AN83	83	84	AN0
AN85	85	86	AN0
AN87	87	88	AN0
AN89	89	90	AN0
AN91	91	92	AN0
AN93	93	94	AN0
AN95	95	96	AN0

### DIAMOND-MM-32 BLOCK DIAGRAM

