Modeling and Simulation of Robots Playing Football using MATLAB/SIMULINK

by

Noor Atikah Binti Mohd Fauzie

Fianl report submitted in partial fulfillment of
the requirements for the
Bachelor of Engineering (Hons)
(Electrical and Electronics Engineering)

JUNE 2008

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

SAMPLE OF CERTIFICATION OF APPROVAL

CERTIFICATION OF APPROVAL

Modelling and Simulation of Robots Playing Football using MATLAB

by

Noor Atikah Binti Mohd Fauzie

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(ELECTRICAL & ELECTRONICS ENGINEERING)

Approved by,

Herman Agustiawan

Dr. Herman Agustiawan
Associate Professor
Electrical & Electronic Engineering Department
Universiti Teknologi PETRONAS

(Dr. Herman Agustiawan)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

June 2008

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.

NOOR ATIKAH BINTI MOHD FAUZIE

ABSTRACT

Cooperating autonomous robots are characterized as intelligent systems that combine perception, reasoning, and action to perform cooperative tasks under circumstances that are insufficiently known in advance, and changing during task execution. There are various reasons to why we should build cooperative robots. They include increasing reliability and robustness through redundancy, decreasing task completion time through parallelism and decreasing cost through simpler individual robot design. Cooperative robots can be applied in various fields such as mining, construction, planetary exploration, automated manufacturing, search and rescue missions, cleanup of hazardous waste, industrial/household maintenance, nuclear power plant decommissioning, security, and surveillance. However, in this project cooperating autonomous robots are applied in terms of robots playing football. A fully autonomous robot has the ability to gain information about the environment, work for an extended period without human intervention, move either all or parts of itself throughout its operating environment without human assistance and to avoid situations that are harmful to people, property or itself. An autonomous robot may also learn or gain new capabilities like adjusting strategies for accomplishing its task(s) or adapting to changing surrounding. Therefore this project will inculcate the criteria of autonomous robots in term of robots playing football. This study will incorporate programming using MATLAB/SIMULINK, producing mathematical models and applying control analysis methods.

ACKNOWLEDGEMENTS

First and foremost, I would like to praise God the Almighty for His guidance. Though difficulties occurred, His guidance gave me the chance to still complete this challenging project successfully. Here, I would like to use this special opportunity to express our heartfelt gratitude to everyone that has contributed to the task of creating a simulation of robots playing football.

Our deepest appreciation goes to our supervisor, Dr. Herman Agustiawan who advised and guided me with moral supports throughout our project. I really acknowledge all the precious words from him and hope the moment working with him remains as valuable experience for my future undertakings. I would also like to address our thankfulness to Mr Salman as he helped me during the preparation of my project. He was very generous in helping me with the simulations and theories.

Not to forget, my highest gratitude to all the lecturers of UTP, my beloved family members, and also my friends that gave feedbacks and helped a lot through their useful ideas, advises and support.

Summing everything up, I really appreciate and pleased to all the helping hands given to me. I enjoyed this project even though it was quite tough. The experience gained throughout this project was so meaningful and I hope that my project could also be used as reference for the other students.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND OF STUDY

Robots playing football is an example of cooperating autonomous robots. Cooperating autonomous robots are characterized as intelligent systems that integrate perception, reasoning, and action to perform cooperative tasks under circumstances that are insufficiently known in advance, and changing during task execution. There are various reasons to why we should build cooperative robots. They include increasing reliability and robustness through redundancy, decreasing task completion time and decreasing cost through simpler individual robot design.

A fully autonomous robot has the ability to gain information about the environment. In this project, the robots in the team will be divided into two categories which are Leader robots and Follower robots. The follower robot and the leader robot are dependant upon each other, where the follower will obtain information from the robot and the leader will obtain and feed information to and from the follower. It will work for and extended period without human intervention, move either all or parts of itself throughout its operating environment without human assistance and to avoid situations that are harmful itself such as collision between robots. An autonomous robot may also learn or gain new capabilities like adjusting strategies for accomplishing its task(s) or adapting to changing surrounding. This capability will be part of the leader robots.

The leader robots task is to obtain the position of all the robots in the field and plan a trajectory when it is trying to position the follower and the ball. Besides obstruction, the

stadium dimension will be considered too. This is to ensure that the robot does not go out of the stadium. All the follower has to do is receive information from the leader on where to kick the ball. The leader will determine an obstruction free path for the robots.

This study will incorporate programming using MATLAB/SIMULINK, producing mathematical models and applying control analysis methods. The following diagram will show the basic layout of the game.

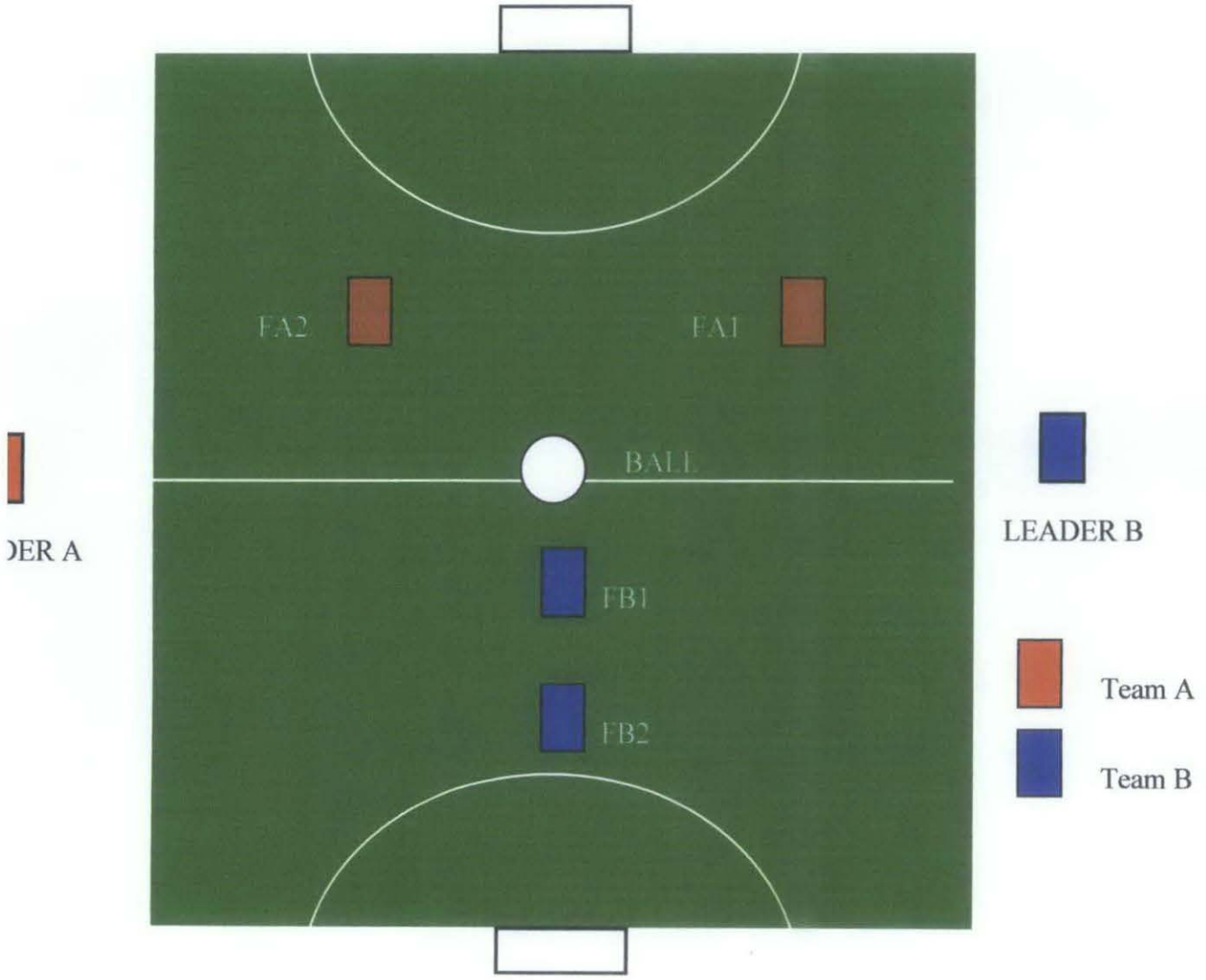

Figure 1 Basic Layout of the Game

1.1 PROBLEM STATEMENT

Robotics is gaining popularity in the world today. Football game will be used as the central topic of research because the innovations can be applied for socially significant problems and industries. This project implements various area of study which includes design principles of autonomous agents, multi-agent collaboration, strategy acquisition, path planning and communication.

1.2 OBJECTIVES AND SCOPE OF STUDY

The objectives of this project will include the following:

1. To model and simulate autonomous robots capable of playing football using the mathematical models.

2. To create a communication platform for the robots to communicate

3. To do path planning

The scope of the study will include Bluetooth communication system and control theories.

1.3 ASSUMPTIONS

The following will be the assumptions used in the project:

- The ball will only travel in a two dimensional manner.
- Collision between two teams will be allowed. However, the leader will plan to avoid collision of robots between the same team
- There will no communication lost between master and slave. What has been sent by master will be received by slave
- Communication delay will be neglected. Therefore, the robots are considered playing in real time
- The robot is considered to kick the ball once it reaches the ball destination

- The power of a kick and the ball velocity after a kick will be considered the same for all robots
- Friction between the ball and robots with the field will be neglected
- The ball could not go out of the field

CHAPTER 2
LITERATURE REVIEW

2.1 HOLONOMICITY

This project will be concentrating on nonholonomic robots. In robotics holonomicity refers to the relationship between the controllable and total degrees of freedom of a given robot. If the controllable degrees of freedom is greater than or equal to the total degrees of freedom then the robot is said to be holonomic. If the controllable degrees of freedom are less than the total degrees of freedom it is non-holonomic.

A system may be defined as holonomic if all the constraints of the system are holonomic. For a constraint to be holonomic it must be expressible as a function:

$$f(p,t) = 0 \qquad (2.1)$$

i.e. the constraint depends only on the coordinates of the system (position) and time. It does not depend on the velocity or momentum of the system.

A nonholonomic system is a system in which a return to the original internal configuration does not guarantee return to the original system position. In other words, unlike with a holonomic system, the outcome of a nonholonomic system is path-dependent, and the number of generalized coordinates required to represent a system is more than its control degrees of freedom (sometimes called differential degrees of freedom, DDOF). In addition to the motion variables corresponding to the control degrees of freedom, the history of its motion too should be known. It can be presented as a function:

$$F(p, \dot{p}) = 0 \qquad (2.2)$$

where $\dot{p}$ is velocity vector. In this project, the unicycle robots which are nonholonomic will be used.

Although it would be more logical to use the holonomic model for robots playing football, there will be a problem when the robot is implemented such as wheel slippage and difficulty in controllability. Besides that, the holonomic robot has a more complex model as compared to the nonholonomic model. Due to these reasons, I chose to use nonholonomic model.

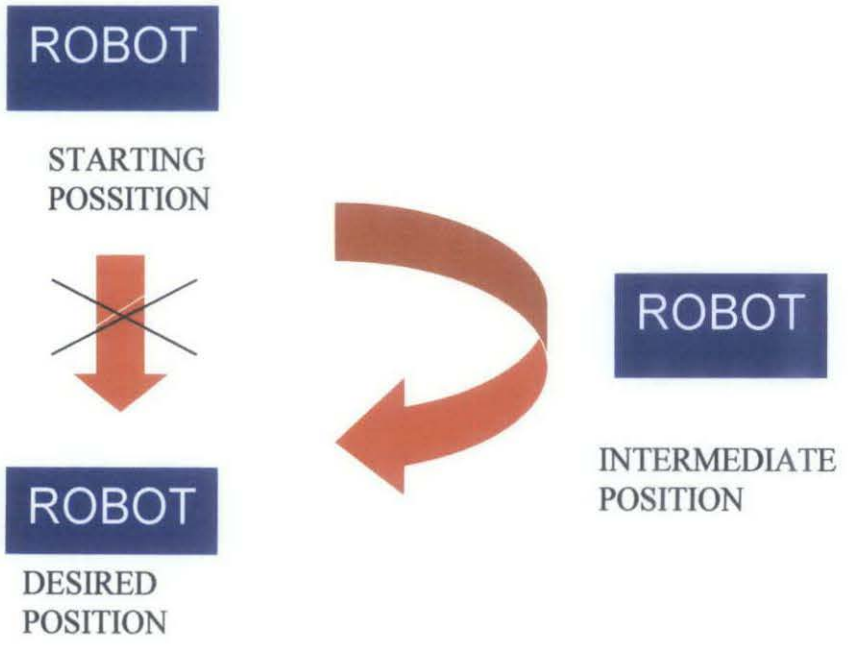

Figure 2 Non Holonomic Movement Constraints

2.2 KINEMATICS MODEL

The project uses kinematics model to position the robots. It is used because kinematics studies the position of an object with time. On the other hand, dynamics is concerned with the forces and interactions that produce or affect the motion.

In order to represent the robot's position, the robot is modeled as a rigid body on wheels, operating on a horizontal plane. The following equations are used for the follower robots position which is sent to the leader robot. The total dimensionality of this robot chassis

on the plane is three, two for position in the plane and one for orientation along the vertical axis, which is orthogonal to the plane. By robot chassis, only the rigid body of the robot will be referred, ignoring the joints and degrees of freedom internal to the robots and its wheels.

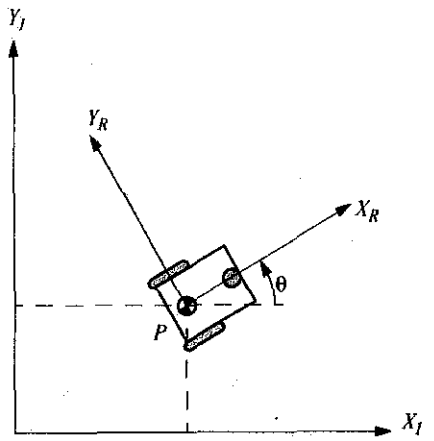

Figure 3 The global reference and the robot local reference frame

A relationship between the global reference frame and the local reference frame of the robot needs to be established to specify the position of the robot, as in Figure 3. The axes X_I and Y_I define an arbitrary inertial basis on the plane as the global reference frame from some origin O: $\{X_I, Y_I\}$. To specify the position of the robot, choose a point P on the robot chassis and is thus the robot's local reference. The position P in the global and local reference frame is specified by coordinates x and y, and the angular difference between the global and local reference frames is given by θ. We can describe the pose of these robots as a vector of these three elements. Note the use of subscript I is to clarify the basis of this pose as the global reference frame. [1]

$$\xi_I = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{2.3}$$

To describe robot motion in terms of component motions, it will be necessary to map motion along the axes of the global reference frame to motion along the axes of the robot's local reference frame. Mapping is a function of the current pose of the robot. It can be accomplished using orthogonal rotation matrix:

$$R(\theta) = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \tag{2.4}$$

This matrix can be used to map motion in the global reference frame $\{X_I, Y_I\}$ to motion in terms of the global reference frame $\{X_R, Y_R\}$. This operation is denoted by $R(\theta)\,\xi_I$ because the computation of this operation depends on the value of θ.

$$\dot{\xi}_R = R(\theta)\dot{\xi}_I \tag{2.5}$$

2.3 TRAJECTORY GENERATION

The nonholonomic nature is related to the assumption that the robot rolls without slipping. This implies the presence of a nonintergrable set of first order differential constraints on the configuration variables. From the previous model, we can equate the sates as:

$$\dot{x}_d = v_d \cos \theta_d \tag{2.6}$$

$$\dot{y}_d = v_d \sin \theta_d \tag{2.7}$$

$$\dot{\theta}_d = \omega_d \tag{2.8}$$

The subscript d represents desired, therefore x_d represents desired x position. Therefore we can know x_d, y_d, θ_d by:

$$x_a = \int v\cos\theta\, dt \tag{2.9}$$

$$y_a = \int v\sin\theta\, dt \tag{2.10}$$

$$\theta = \int \omega\, dt \tag{2.11}$$

Since nonholonomic nature is nonintergrable, differential flatness will be used instead.

Definition of differential flatness:

A system is flat if we can find a set of outputs such that all states and inputs can be determined from those outputs without integration. [6]

$$y = y(x, u, \dot{u}, \ldots, u^{(l)}) \tag{2.12}$$

$$x = x(y, \dot{y}, \ldots, y^{(q)}) \tag{2.13}$$

$$u = u(y, \dot{y}, \ldots, y^{(q)}) \tag{2.14}$$

By using differential flatness, and the same state trajectories;

$$\dot{x}_a = v_d \cos\theta_d \tag{2.15}$$

$$\dot{y}_a = v_d \sin\theta_d \tag{2.16}$$

$$\dot{\theta}_a = \omega_d \tag{2.17}$$

the input trajectory will be;

$$v_d(t) = \pm\sqrt{(\dot{x}_d^2 + \dot{y}_d^2)} \tag{2.18}$$

$$\theta_d(t) = ATAN2\left\{\dot{y}_d, \dot{x}_d\right\} \tag{2.19}$$

$$\varpi_d(t) = \frac{\ddot{y}_d \dot{x}_d - \ddot{x}_d \dot{y}_d}{\dot{x}^2 + \dot{y}^2} \tag{2.20}$$

and the output trajectories are:

$$x_d = x_d(t) \tag{2.21}$$

$$y_d = y_d(t) \tag{2.22}$$

$$t > t_0 \tag{2.23}$$

2.4 OTHER TYPES OF PATH PLANNING

The leader is required to generate a trajectory that will be used by the follower to reach the desired positions. There are two methods used which are cubic and heating functions path planning.

2.4.1 Cubic Path Planning

The first model used to generate the path is using cubic polynomial path planning. The basis of this model is the dsired velocity and cubic trajectory position formula.

$$q(t)=a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{2.23}$$

The desired velocity is given in the form:

$$\dot{q}(t)=a_1 + 2a_2 t + 3a_3 t^2 \tag{2.24}$$

Combining the cubic trajectory position formula and the desired velocity yields four equations and four unknowns.

$$q_0 = a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 \tag{2.25}$$

$$v_0 = a_1 + 2a_2 t_0 + 3a_3 t_0^2 \tag{2.26}$$

$$q_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 \tag{2.27}$$

$$v_f = a_1 + 2a_2 t_f + 3a_3 t_f^2 \tag{2.28}$$

For this project, the following are assumed:

$$v_0 = 0 \tag{2.29}$$

$$v_f = 0 \tag{2.30}$$

$$t_0 = 0 \tag{2.31}$$

Solving the equations will yield the following equations which are the equations used in the model.

$$q(t) = q_0 + 3(q_f - q_0)t^2 - 2(q_f - q_0)t^3 \quad (2.32)$$

$$\dot{q}(t) = 6(q_f - q_0)t - 6(q_f - q_0)t^2 \quad (2.33)$$

2.4.1 Path Planning using Heating Functions

The heating function plays a key role in guaranteeing asymptotic stability by sustaining motion as long as the error, e is not zero, but it also determines the transient behavior. The heating function is given by the following equation:

$$g(e,t) = \frac{\exp(k_5 e_2) - 1}{\exp(k_5 e_2) + 1} \sin t \quad k_5 > 0 \quad (2.33)$$

$$e_2 = (x_d - x_a)(-\sin\theta) + (y_d - y_a)\cos\theta \quad (2.34)$$

k_5 is a constant that determines the shape of the trajectory generated while e_2 is the state tracking error.

2.5 LEADER-FOLLOWER FORMATION

There are three approaches to formation control which are behavior-based, virtual structure formation and leader-follower formation. Majority of the algorithms implemented using behavior-based formation and virtual structure formation are implemented on robots with visual capabilities. Since the robots implemented in this project are only having communication capability, the leader-follower formation is used.

- Behavior-based formation: a distributed approach and has explicit information feedback between neighbors and focuses on peer to peer communication.
- Virtual structure formation: a centralized approach

- Leader-follower formation: one of the robots is designated as a leader and the others as followers.

The leader robot plans and follows a desired trajectory. The follower robot follows the leader robot with a desired distance. The leader is responsible for guiding the formation. The leader-follower formation comprises of two formation controllers explained in the next section.

2.5.1 Separation Bearing Controller

The separation bearing controller is used for two robots. The follower robot follows the leader while maintaining a desired relative distance and separation bearing angle with respect to the leader robot. Figure 4 shows a schematic for this formation.

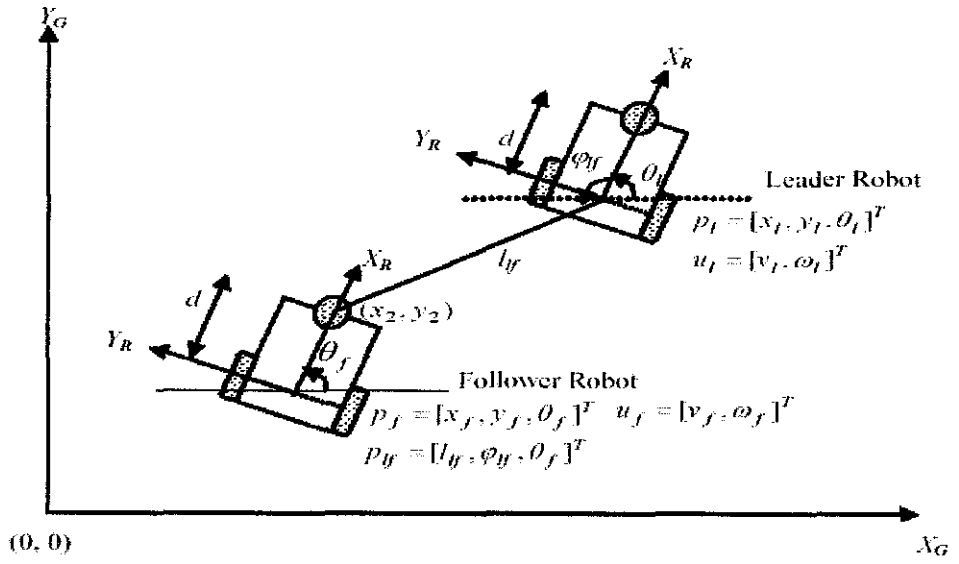

Figure 4 Separation Bearing Controller Schematic

Knowing the leader robot equation and the separation distance between the leader and follower, the follower position can be calculated.

$$x_f = x_2 + d\cos\theta_f$$
$$y_f = y_2 + d\sin\theta_f$$

The follower robot can be modeled relatively to the leader robot as $\rho_{lf} = [l_{lf}, \vartheta_{lf}, \theta_f]^T$. The new state vector, ρ_{lf} can be expressed through a transformation as $\rho_{lf} = T_{SB}(p_l, p_f)$ given by

$$l_{lf} = \sqrt{(x_l - x_f - d\cos\theta_f)^2 + (y_l - y_f - d\sin\theta_f)^2}$$
$$\varphi_{lf} = \pi - \arctan 2(y_f + d\sin\theta_f - y_l, x_l - x_f - d\cos\theta_f) - \theta_l$$

The original state vector, can be recovered throught he inverse transformation $p_f = T_{SB}^{-1}(p_l, p_{lf})$. Differentiating equation and combining it with equation, the follower robot kinematic model is obtained.

$$\dot{l}_{lf} = v_f\cos\gamma - v_l\cos\varphi_{lf} + d\omega_f\sin\gamma$$
$$\dot{\varphi}_{lf} = \frac{v_l\sin\varphi_{lf} - v_f\sin\gamma - \omega_l l_{lf} + d\omega_f\cos\gamma}{l_{lf}}$$
$$\dot{\theta}_f = \omega_f$$

where $\gamma = \theta_l - \theta_f + \vartheta_{lf}$. In order to avoid collision between the follower and the leader robots, a requirement that $l_{lf} > 2d$ must be ensured.

2.6 COMMUNICATION

There are various communication technologies in the market today. However, this project will implement only one type. Below are the comparisons for the various technologies.

Table 1 Comparisons between various communication technologies

Technology	Description
Bluetooth	• Bluetooth wireless technology is geared towards voice and data applications. • Operates under unlicensed 2.4GHz spectrum • Can operate over a distance of 10 meters or 100 meters depending on the Bluetooth device class • Peak data rate is 3Mbps • Able to penetrate solid objects • Omni directional, does not need line-of-sight positioning of connected device • Allows three modes of security • The cost of Bluetooth device is cheap
Wi-Fi	• The cost of implementation is 3 times higher as compared to Bluetooth • Power consumption is 5 times higher as compared to Blueetooth • Different 802.11 standards give different operating range, ranging from 2.4 GHz to 5.9GHz with a data rate of 54 Mbps to 11 Mbps.
WiMax	• A wireless Metropolitan Area Network (MAN) • Has a range of 50km with data rates of 70Mbps. • The original 802.16 standard operates in the 10-66 GHz frequency bands with line-of-sight environments • The 802.16a standard operates between 2 and 11 GHz and does not need line-of-sight
Infrared (IrDA)	• Used to provide wireless connectivity for devices that would normally use cables to connect. • IrDA is a point-to-point, narrow angle (30^{o}) • Ad-hoc data transmission standard designed to operate over a distance of 0 to 1 meter and at speeds of 9600 bps to 16Mbps • Not able to penetrate solid objects and has a limited data exchange applications compared to other wireless technologies
ZigBee	• Capacity of 250Kbits at 2.4 GHz, 40 Kbps at 915 MHz, and 20Kbps at 868 MHz with a range of 10-100 M. • This technology is targeting the control applications industry, which does not require high data rates, but must have low power, low cost and ease of use. • It is also low cost

Based on the comparison done, Bluetooth communication was selected as an information sharing medium among the collaborative robots.

2.7 BLUETOOTH PROFILES

A Bluetooth profile is a standard interface between Bluetooth devices. They are general behaviors whict Bluetooth enabled devices communicate with other devices. There are several profiles available in the Bluetooth protocol suite for communication among devices such as Personal Area Network (PAN) or piconet. The first Bluetooth device in the piconet is the master and the rest are slaves communicating with the master. Each device in the piconet is called a Personal Area Network Unit (PANU). There are three modes in which devices can interact with each other. They are as follow:

- **PANU-PANU**

  This mode supports at a maximum of two devices. One device will act as a master and the second device will act as a slave. Figure 4 shows two robots communicating using PANU-PANU mode.

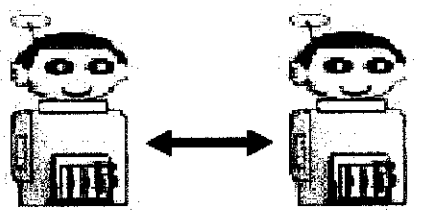

Figure 5 PANU-PANU mode

- **Group Adhoc Network**

  The Group Adhoc Network (GN) enables two or more PAN Users to communicate with each other. The GN device acts as a master and supports at the maximum of seven slaves. The GN mode for communication is shown in Figure 5.

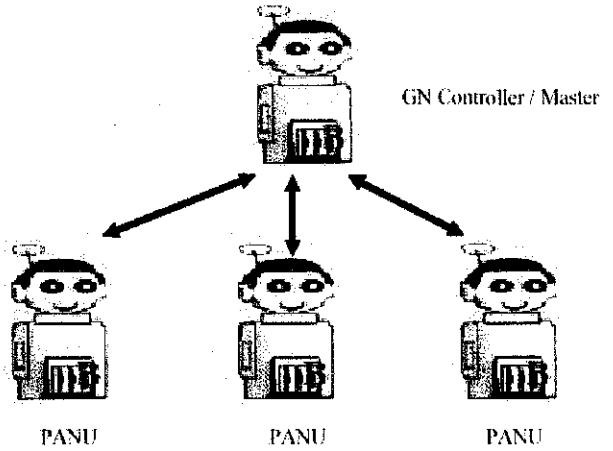

Figure 6 Group Adhoc Network Mode

- **Network Access Point**

  A Network Access Point (NAP) is a Bluetooth device that provides the service of routing network packets. A NAP can act as a bridge between Bluetooth networks anda other networks such as a Local Area Network (LAN). The figure below shows the NAP mode of communication among robots.

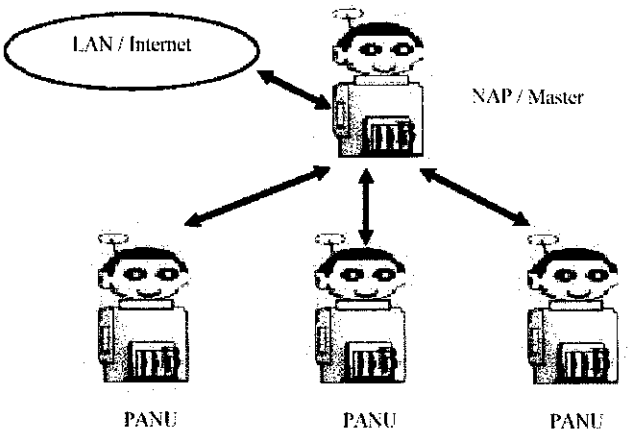

Figure 7 Network Access Point mode of communication

2.7.1 File Transfer Profile

The following roles are defined for this profile

- **Server** – The Server device is the target remote Bluetooth device that provides an object exchange server and folder browsing capability using the OBEX Folder Listing format. In addition to the interoperability requirements defined in this profile, the Server must comply with the interoperability requirements for the Server of the GOEP if not defined in the contrary.

- **Client** – The Client device initiates the operation, which pushes and pulls objects to and from the *Server*. In addition to the interoperability requirements defined in this profile, the Client must also comply with the interoperability requirements for the Client of the GOEP if not defined in the contrary. (.1)

Servers must be placed in File Transfer mode. This mode enables a Client to perform file transfer operations with the Server.

Clients provide file transfer functions to the user via a user interface. An example of a file transfer user interface is a file-tree viewer to browse folders and files. Using such a system file-tree viewer, the user can browse and manipulate files on another PC, which appears in the network view.

File Transfer Applications provide the following functions.

- Select Server
- Navigate Folders
- Pull Object
- Push Object
- Delete Object
- Create Folder

When the Select Server function is selected, an inquiry procedure will be performed to produce a list of available devices in the vicinity.

Objects are transferred from the Client to the Server using OBEX PUT, and objects are transferred from the Server to the Client using OBEX GET. Transferring files requires a single PUT or GET operation per file. Transferring folders requires transferring all the items stored in a folder, including other folders. The process of transferring a folder may require that new folders be created. The SETPATH command is used to create folders.

There are 6 OBEX operations which are used in the Object Push Profile: **Connect, Disconnect , Put , Get , Abort & SetPath.**

2.7.2 Object Push Profile

The following roles are defined for this profile:

- **Push Server** – This is the server device that provides an object exchange server. In addition to the interoperability requirements defined in this profile, the Push Server must comply with the interoperability requirements for the server of the GOEP if not defined in the contrary.

- **Push Client** – This is the client device that pushes and pulls objects to and from the Push Server. In addition to the interoperability requirements defined in this profile, the Push client must also comply with the interoperability requirements for the client of the GOEP, if not defined to the contrary.

There are three different **functions** associated with the Object Push profile:

- Object Push function
- Business Card Pull function
- Business Card Exchange function

However, for this study, only Object Push function will be considered

1. The **Object Push function** initiates the function that pushes one or more objects to a Push Server.

The function should be activated by the user. They should not be performed automatically without user interaction. When the user selects one of these functions, an inquiry procedure will be performed to produce a list of available devices in the vicinity.

The Object Push function is mandatory on both the Push Client and Push.

This feature lets a Push Client send one or more objects to a Push Server.

- **Content Format**: To achieve application level interoperability, content formats are defined for Object Push. For some applications content formats have been specified.It is highly recommended that a Push Client does not try to send objects of a format that the Push Server does not support.

- **Application Procedure**: It is mandatory for Push Servers to be able to receive multiple objects within an OBEX connection. It is not mandatory for Push Clients to be able to send multiple objects during an OBEX connection. The Push Client uses one PUT operation for each object it wants to send. It is not mandatory to support sending or receiving of multiple objects within a single PUT operation.

There are 5 OBEX operations which are used in the Object Push Profile: **Connect** , **Disconnect** , **Put** , **Get** & **Abort**.

2.8 USER DATAGRAM PROTOCOL

UDP (User Datagram Protocol) is a communications protocol that offers a limited amount of service when messages are exchanged between computers in a network that uses the Internet Protocol (IP). UDP is an alternative to the Transmission Control Protocol (TCP) and, together with IP, is sometimes referred to as UDP/IP. Like the Transmission Control Protocol, UDP uses the Internet Protocol to actually get a data unit (called a datagram) from one computer to another. Unlike TCP, however, UDP does not provide the service of dividing a message into packets (datagrams) and reassembling it at the other end. Specifically, UDP doesn't provide sequencing of the packets that the data arrives in. This means that the application program that uses UDP must be able to make sure that the entire

message has arrived and is in the right order. Network applications that want to save processing time because they have very small data units to exchange may prefer UDP to TCP.

2.9 TCP vs. UDP

The table below compares the pros and cons of UDP and TCP in terms of connection, reliability, order, weight and packet.

Table 2 Comparison between TCP and UDP

	Transmission Control Protocol (TCP)	**User Datagram Protocol (UDP)**
Connection	Connection-oriented protocol. Therefore, upon communication, it requires handshaking to set up end-to-end connection.	Connectionless protocol. There are no efforts made to set up end-to end connection. Communication is achieved by sending data in one direction, from source to destination, without checking if the receiver is still there or if it is prepared to receive the data
Reliability	Reliable. Manages message acknowledgement, retransmission and timeout. If data gets lost along the way, the server will re-request the lost part.	Unreliable. There is no concept of retransmission, acknowledgement and timeout
Order	Ordered. Messages sent must reach the receiving application by order. When it arrives in the wrong order, the TCP layer holds the later data until the earlier data can be rearranged and delivered to the application.	Not ordered. The order of messages sent could not be predicted.
Weight	Heavyweight. Requires three packets just to set up a socket, before any actual data can be sent. It handles connections, reliability and congestion control	Lightweight. There is no ordering of messages, no tracking connections, etc.
Packet	Data are read as a "stream", with nothing distinguishing where one packet ends and another begins.	Packets are sent individually and are guaranteed to be whole if they arrive. Packets have definite

	Packets may be split or merged into bigger or smaller data streams arbitrarily.	bound and no split or merge into data streams may exist.

2.10 HOW BLUETOOTH WILL WORK

2.10.1 Setup and Requirements

Bluetooth capabilities are available for the following operating systems:

- **Windows XP SP 2 and later** - Bluetooth devices that use the Microsoft Bluetooth driver included with Windows XP Service Pack 2 and later. Most Bluetooth devices come packaged with a proprietary Bluetooth driver. To use the device, the Bluetooth adapter must be using the Microsoft Bluetooth driver.
- **Windows 2000** – Bluetooth applications for a PDA target can be developed in Windows 2000. However, Bluetooth communication cannot be utilized since Windows 2000 does not have a compatible Bluetooth driver. Applications developed in Windows 2000 can be built for a target that supports Bluetooth and run on that system.
- **Pocket PC 2003** – To run Bluetooth applications on Pocket PC 2003 devices they must be using the WIDCOMM BTW-CE driver version 1.4 or later. Additional installation of Bluetooth DLLs is also required. Refer to the *Installing the WIDCOMM Bluetooth DLLs on Pocket PC 2003 Devices* section of the *Getting Started with LabVIEW PDA Module* for more information.
- **Palm OS 5.0 and later** – To run Bluetooth applications on Palm devices, you must have Palm OS 5.0 and later with installed Bluetooth hardware.

2.10.2 Creating a Bluetooth Server Application

Complete the following steps to develop a Bluetooth server application.

1. Create a Bluetooth service – Use the **Bluetooth Create Listener** function to create a Bluetooth service identified by a Bluetooth uuid. This function returns a listener ID which refers to this server. The **Bluetooth Create Listener** function also returns a reserved

Bluetooth channel that the server can use to listen for inbound connections. A Bluetooth channel is a global resource with only 30 channels available on any Bluetooth device. If no server channel is available the function returns an error.

2. Wait for incoming connection request – Use the **Bluetooth Wait on Listener** function to wait for and accept an incoming connection request from a client. This function returns a connection ID that is used to exchange data with the client.

3. Read and Write data – Use **Bluetooth Read** and **Bluetooth Write** functions to exchange data with the client.

4.Close connection – Use **Bluetooth Close Connection** function to close connection to the client and to stop listening for incoming connections.

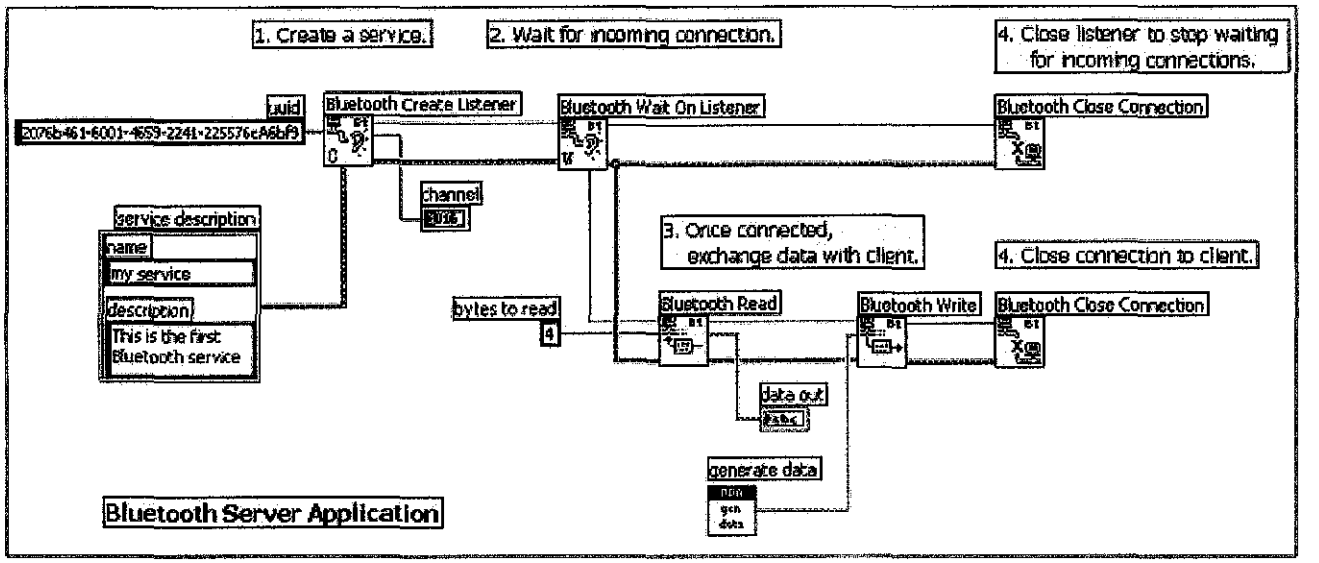

Figure 8 Bluetooth Server Application Block Diagram

10.3 Creating a Bluetooth Client Application

Complete the following steps to develop a Bluetooth client application.

1. Request a connection to Bluetooth server – Use the **Bluetooth Open Connection** function to connect to a service on a Bluetooth server. Set the channel number to zero and specify a Bluetooth uuid to identify which service to connect to. The **Bluetooth Open Connection** function performs an SDP query to make a connection to the first service found with matching uuid. Internally, the result of an SDP query is an RFCOMM channel number to connect to. The SDP query is a tool to "translate" uuid to a channel number.

If the channel number associated with the service is known in advance, use the channel number instead of zero. Specifying a nonzero channel number bypasses the internal SDP query operation thus reducing the amount of time it takes to connect to the service. If the channel number is nonzero, the uuid input parameter is ignored.

2. Read and Write data – Use **Bluetooth Read** and **Bluetooth Write** functions to exchange data with the server.

3. Close connection – Use **Bluetooth Close Connection** function to close connection to the server.

The block diagram of a typical Bluetooth client application looks similar to Figure 2.

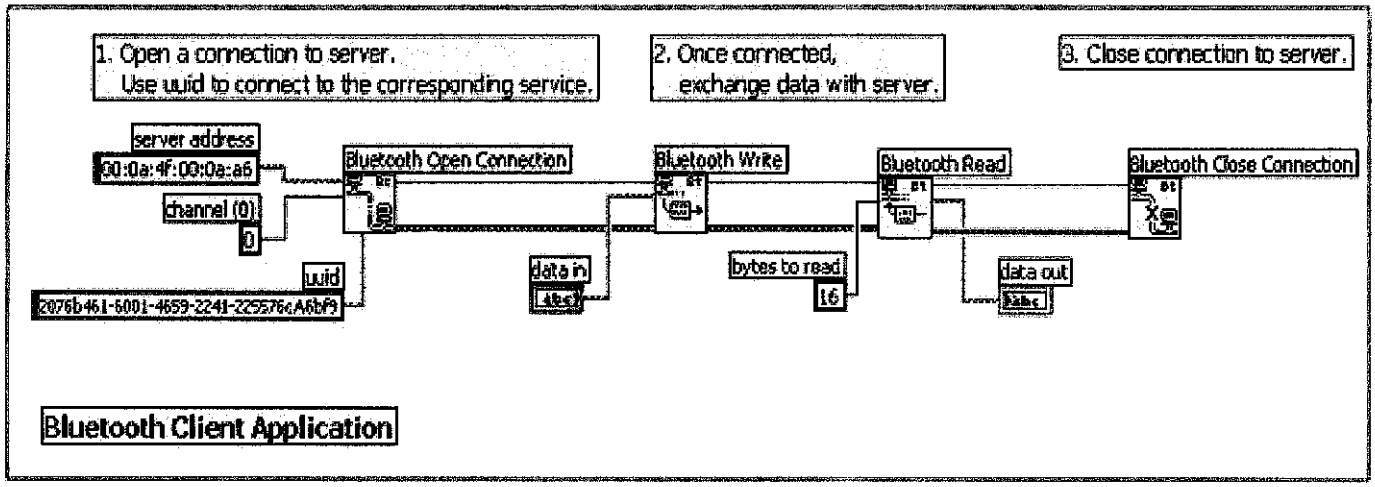

Figure 9 Bluetooth Client Application Block Diagram

2.10.4 Finding Nearby Bluetooth Devices

Use the **Bluetooth Discover** function to search for Bluetooth devices that are within the permissible range. The function returns a list of device addresses and names. These device addresses can then be used by a client to connect to a specific Bluetooth server. The optional input parameter, **time limit (ms)**, specifies the length of Bluetooth inquiry. The default value is 10 ms and the maximum value is 30 s. If **time limit** is less than or equal to zero, the function returns a list of installed local Bluetooth devices.

Note: Bluetooth discovery is a slow operation because of the communication involved. A Bluetooth device address is a fixed address that is usually printed on the actual device and you can query the address from the device control setting. This address is unique to each device. If you know the Bluetooth address of the specific device you want to connect to, you can skip the discovery process and use the **Bluetooth Open Connection** function to connect to the device directly.

CHAPTER 3

METHODOLOGY

The robots are divided into two categories, which are Leader robots and Follower robots. The following will be regarding the leader and follower robot roles and how communication will be done.

3.1 LEADER ROBOTS

As we can see from Figure 1, each team will have one leader robot. The leader robots will ask for the position of all of its follower robots as well as the opponent team follower robots. It has to know the opponent team's positions in order to avoid collision and plan an obstruction free trajectory. It also has to plan the trajectory based on the field layout to avoid the robots from going out of the field.

Besides that, the leader will keep updating the position of the ball. The leader will not be in the field playing with the other follower robots. It will be outside of the game and it will play the role of the playmaker. This means that it is responsible to plan the strategy of its team. It will also know the dimension of the goal post in order to instruct the robots to score a goal

3.2 FOLLOWER ROBOTS

The follower robot will be the football players. It will receive information from the leader robot on its movement trajectory, the position of the ball and the direction it is suppose to kick the ball. All the follower robots have to do is follow instructions by the leader robots.

3.3 COMMUNICATION

After comparing the various technologies that have been mentioned in part 2.4, the Bluetooth technology is the best technology for this project. This is because it is low cost, able to communicate within the field range and it does not nee line of sight. Therefore, all communications between follower robots, leader robots and ball will be done using Bluetooth. It will be set up using Master Slave communication where the Leader will be the Master and the Follower will be the Slave. A Master Bluetooth can support up to eight active devices (PICONET) and 255 inactive devices. The master can bring to active any inactive devices at any time.

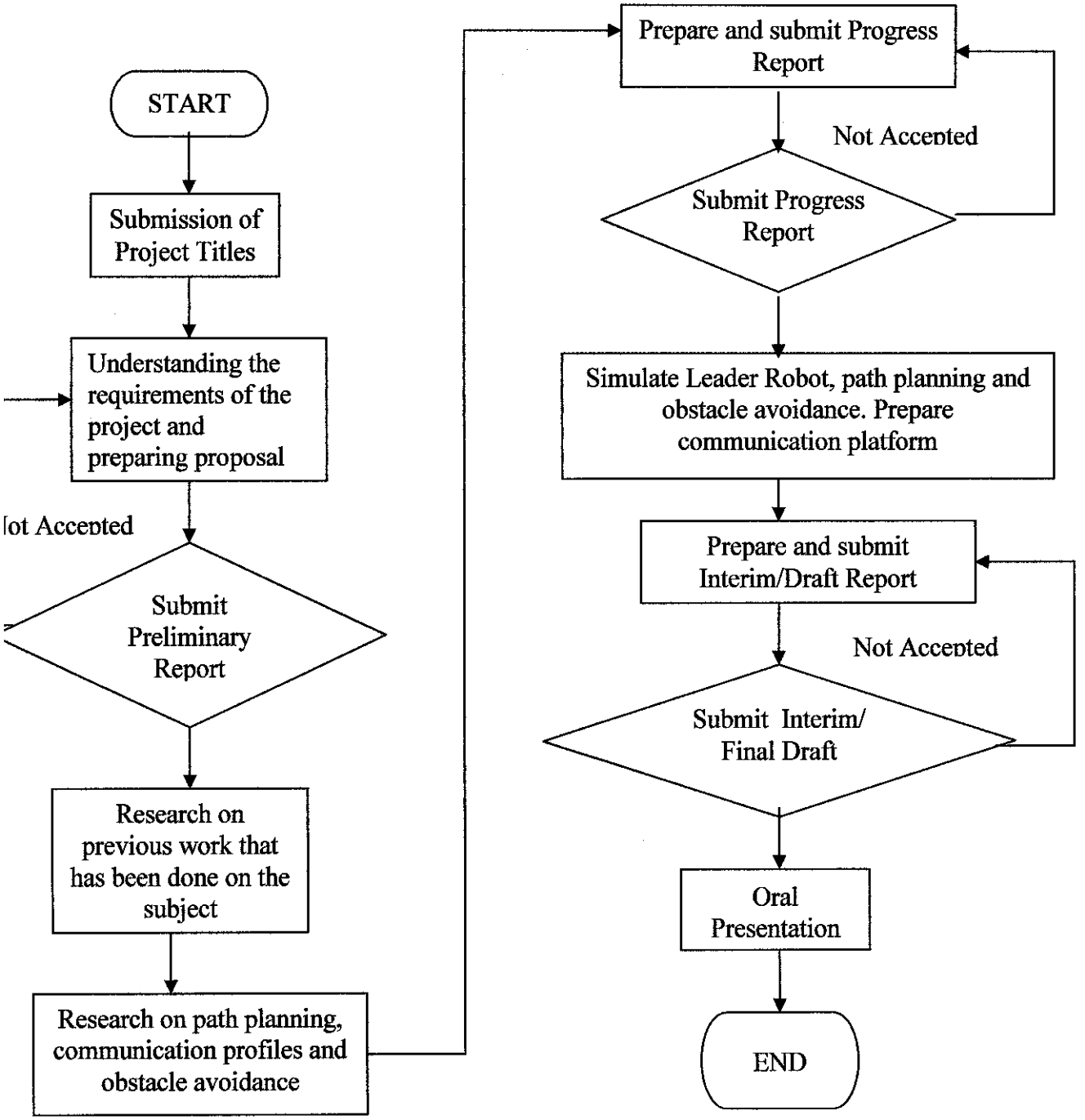

Figure 10 Project flow chart

The following figure will show the overview of steps that will be taken to accomplish this project. In other words, the figure shows the planned steps of the project. The project will start with modeling the follower robots, ball and leader robots. After that, the project will concentrate on path planning and trajectories tracking. This will be placed in the Leader robot for it to instruct the other follower robots on where to go. Before simulating in MATLAB, the communication between the leader and follower will be set up using Bluetooth. The next step is simulation in MATLAB/SIMULINK. Once all the path and trajectories movement does not collide between the robots, the author can begin with the strategy of each team.

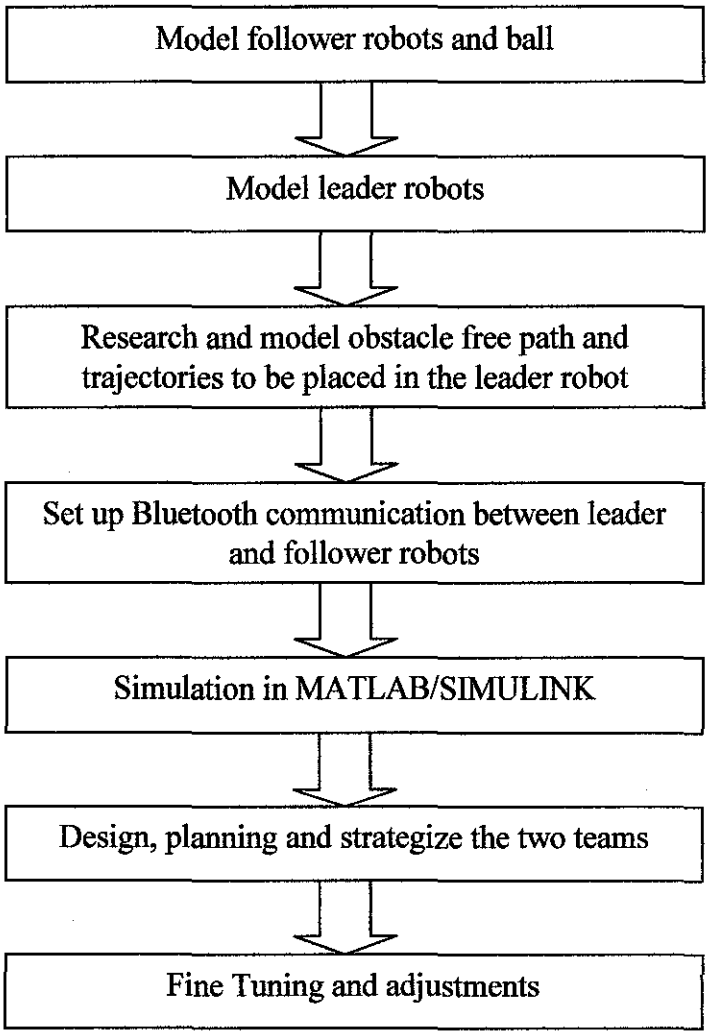

Figure 11 Project steps chart

CHAPTER 4

RESULTS AND DISCUSSION

4.1 DELIVERABLES

The author is required to deliver the following by the end of this project:

1. A simulation of the field, showing the dynamic movements of the robot while the football game is in play
2. Simulated robots playing football with at least two robots playing
3. A communication path for the robots to communicate within the team
4. Path planning and trajectory tracking for each team
5. Game strategy for each team

4.2 RESULTS

In this section all the results obtained will be shown and discussed.

4.2.1 Field Generations

The field that will be used in the game is created using MATLAB. The field is created based on the standard field provided by Fédération Internationale de Football Association (FIFA). However, the field has been scaled down on the scale of 1:5 and it has been simplified. The figure below is the dimensions that have been scaled down.

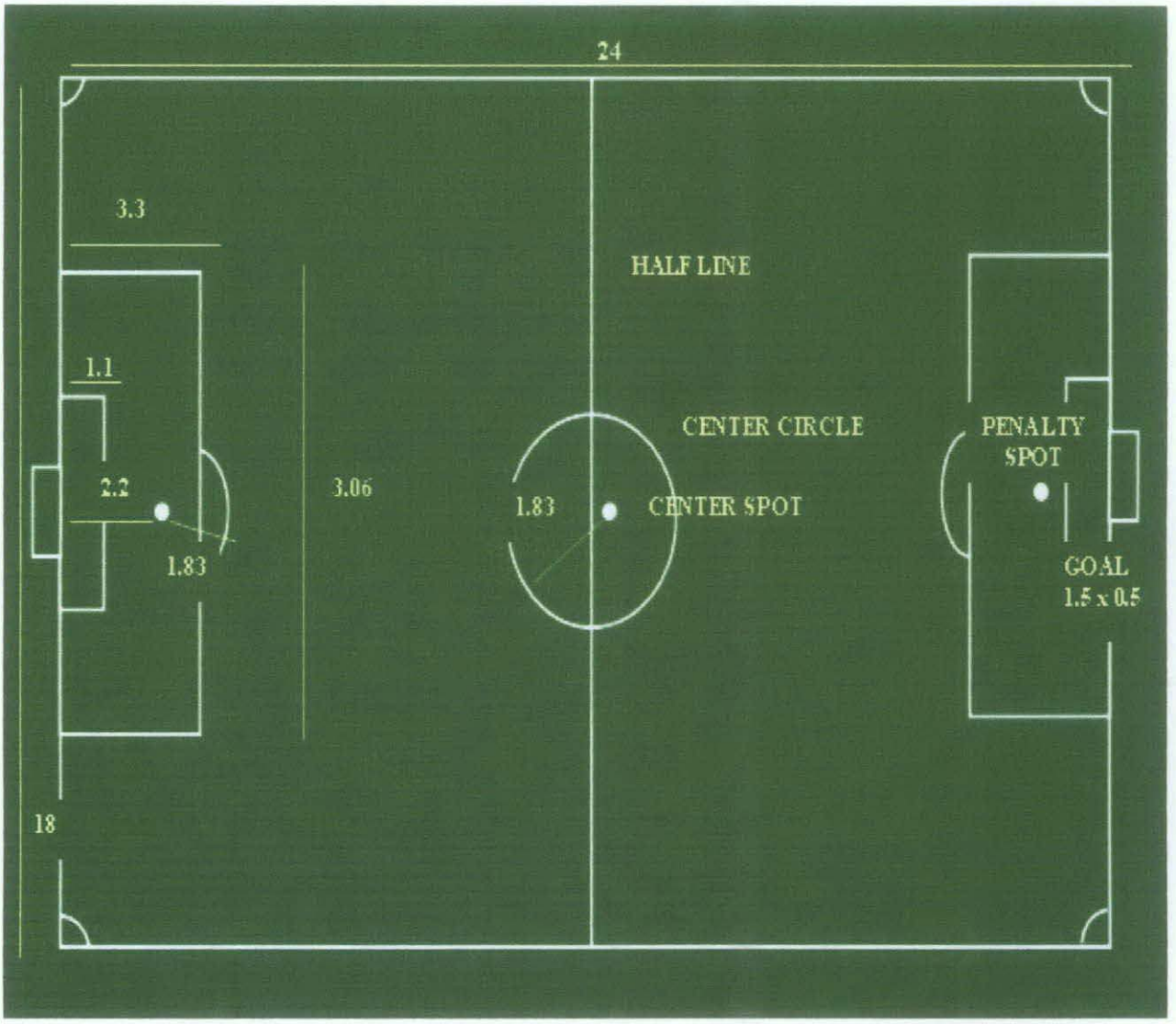

Figure 12 Field dimension based on a standard field provided by FIFA

The M-File that is used to create the field is attached in Appendix A. The field dimension is very important as it will be used to plan the trajectories of the robot. The robot is planned in a manner that the robot could not go out of the field dimension. Besides that, the goal dimension will be used as a target for the robot to score.

The M-File is then saved and run, which will produce the figure below. The figure below will represent the field that will be used during the game. The author has chosen to use a 2 dimension field. This is done because she assumes that the ball could not travel in the air (ball could not fly). This is the basis for the goal post design.

There are two teams on each side on the field. The lighter color circles will be Team A and the darker colored circles are Team B. the small white circle in the middle of the field is the ball. However the graphics are not dynamic yet at this stage. It will show movements once the communications have been successful. At the end of each side of the field, there are goal posts for the robot to score the ball. As compared to real field dimension given above, it is much simpler because this is the basic of the game. The complexity will be added step by step.

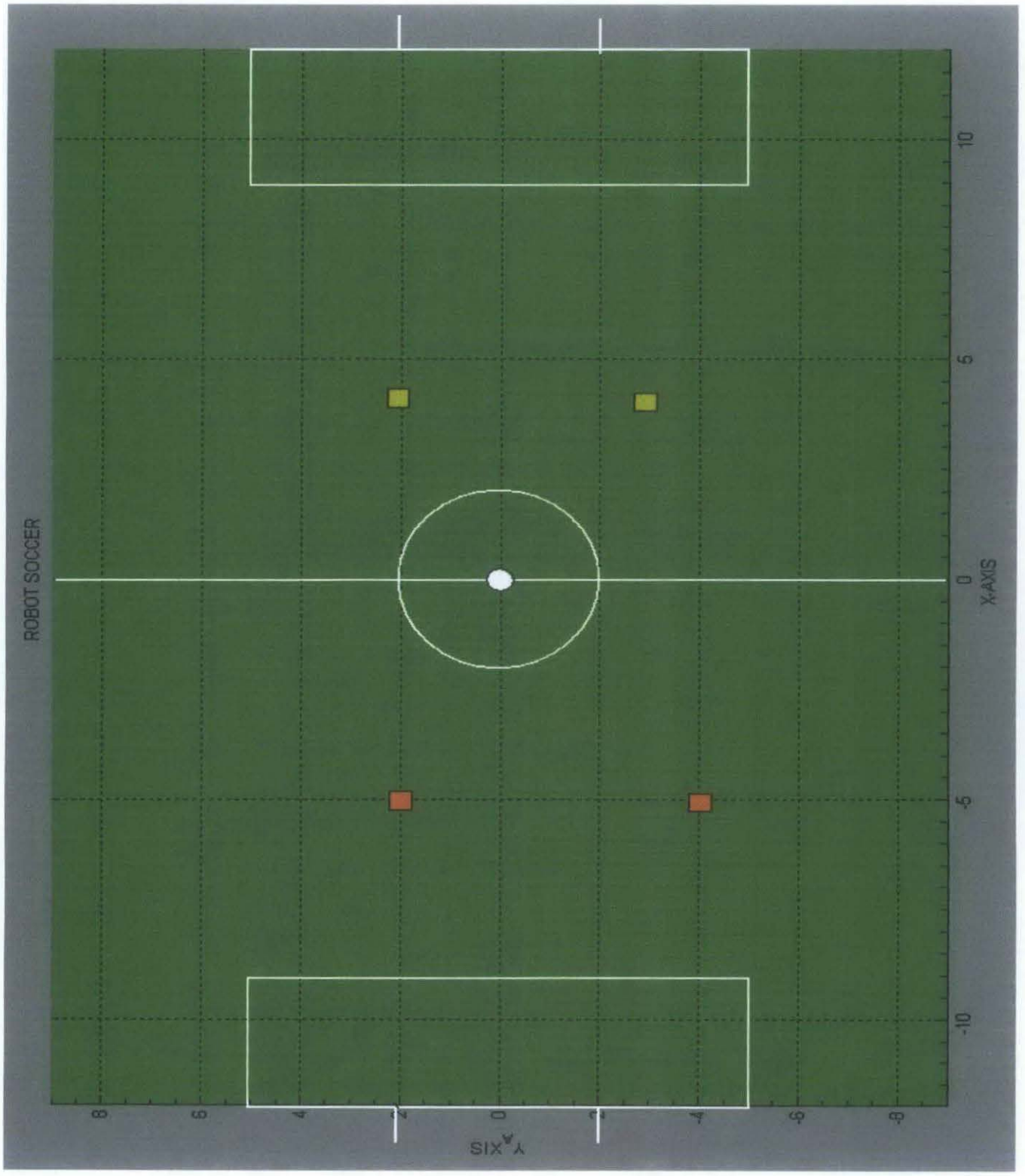

Figure 13 Field produced using MATLAB

4.2.2 Leader Model

The figure below, Figure 11 shows the mathematical model used for the movement of the robot. As u can see from the figure, the model is divided into two main sections; the feed forward controller and the kinematics model.

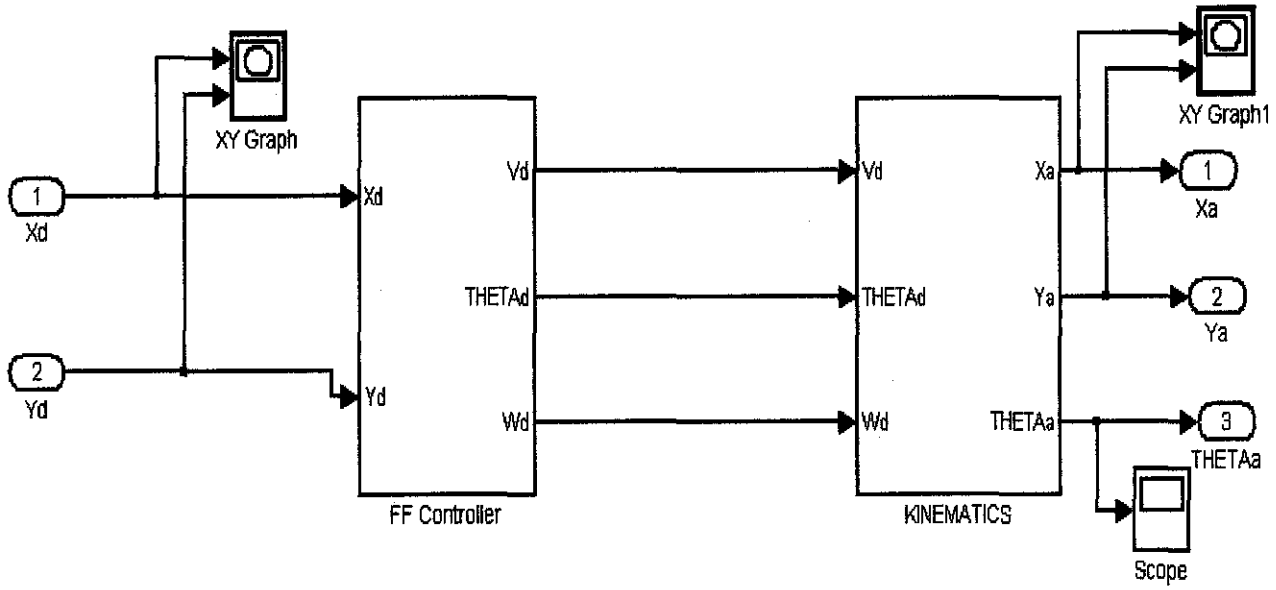

Figure 14 Leader model

Figure 12 shows the components inside the feed forward controller. The Vd(t) block calculates the desired velocity of the robot while Thetad(t) block calculates the theta of the robot. It is represented by the following formula, Equation 2.18, 2.19 and 2.20. The subsystem block is actually calculating the omega of the model which will then be used to calculate the actual Theta of the robot.

$$v_d(t) = \pm\sqrt{(\dot{x}_d^2 + \dot{y}_d^2)} \tag{2.18}$$

$$\theta_d(t) = ATAN2\{\dot{y}_d, \dot{x}_d\} \tag{2.19}$$

$$\omega_d(t) = {y_d'' x_d' - x_d'' y_d'} \Big/ {x'^2 + y'^2} \tag{2.20}$$

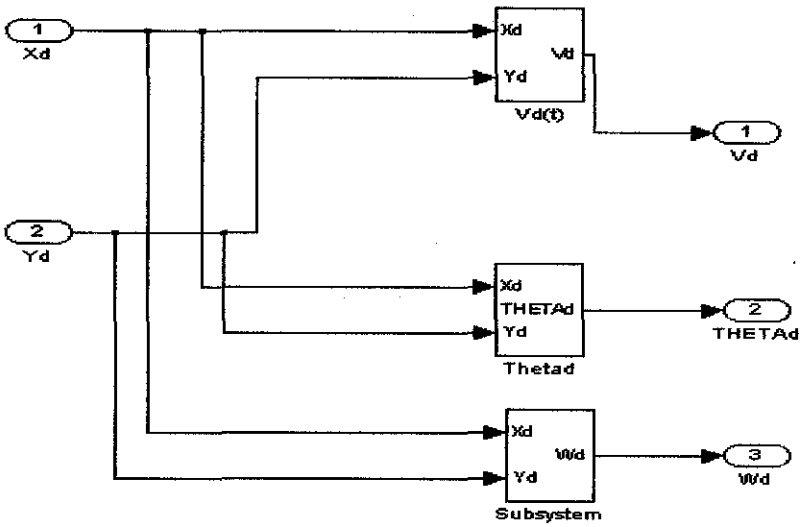

Figure 15 Feed forward controller model

The kinematics model is represented by Figure 13. The block X and Y accordingly calculates the actual X and Y positions by integrating the following formula, (Equation 2.15, 2.16, and 2.17)

$$\dot{x}_a = v_d \cos\theta_d \tag{2.15}$$

$$\dot{y}_a = v_d \sin\theta_d \tag{2.16}$$

$$\dot{\theta}_a = \omega_d \tag{2.17}$$

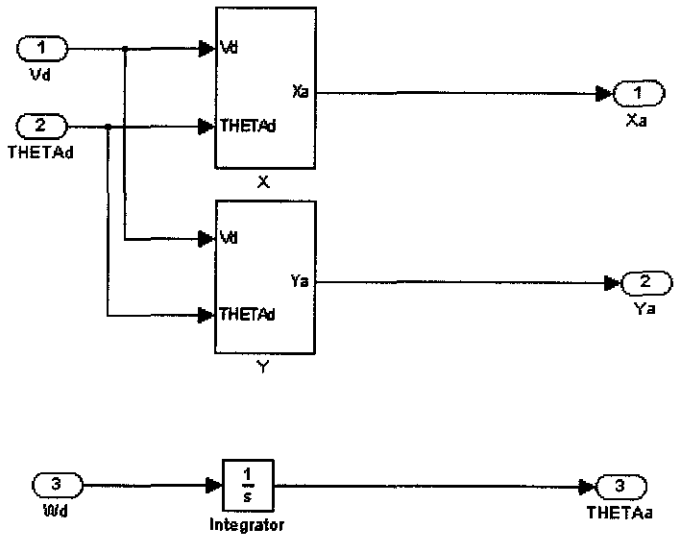

Figure 16 Kinematics Model

4.2.2.1 Slope Trajectory

After simulating the model using a ramp for X_d and Y_d function, the following graph shown in Figure 14 is obtained. It shows the movement of the robot from point (0,0) to (0,1). By comparing the graphs, it can be seen that the movement error of actual and desired is minor.

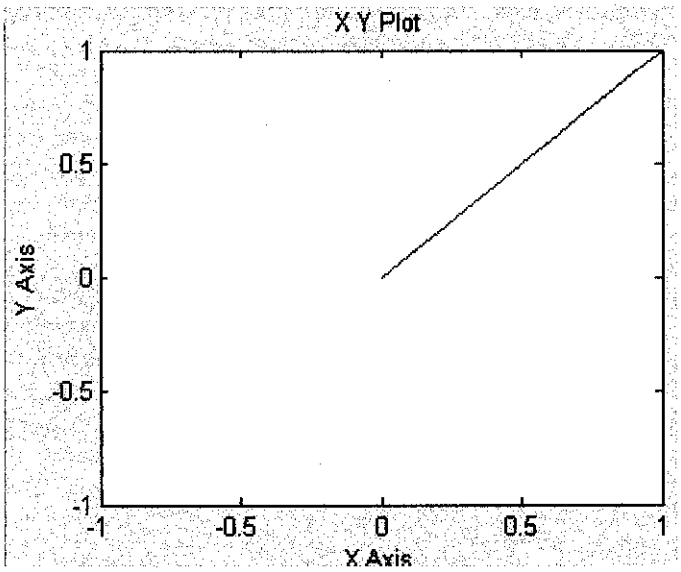

Figure 17 Desired X and Y positions

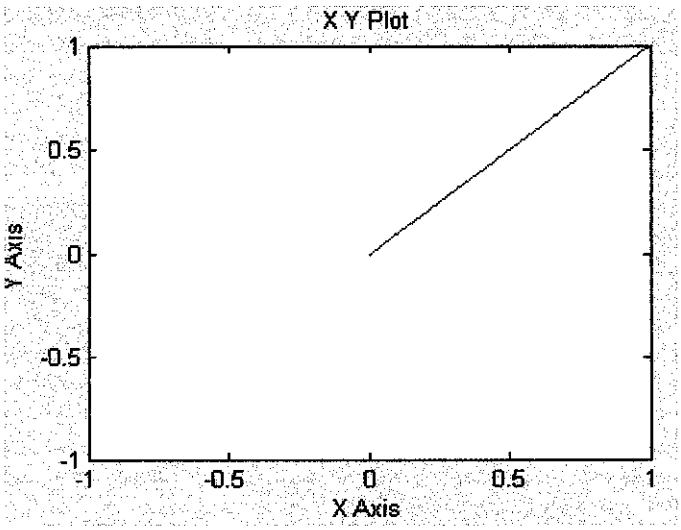

Figure 18 Actual X and Y position

4.2.2.2 Eight Trajectory

The model is simulated using the following parameters. Figure 18 shows the graph obtained.

$$Xd = 4 \sin (t/20)$$

$$Yd = 4 \sin (t/40)$$

$$T = 2*pi*(40)=252$$

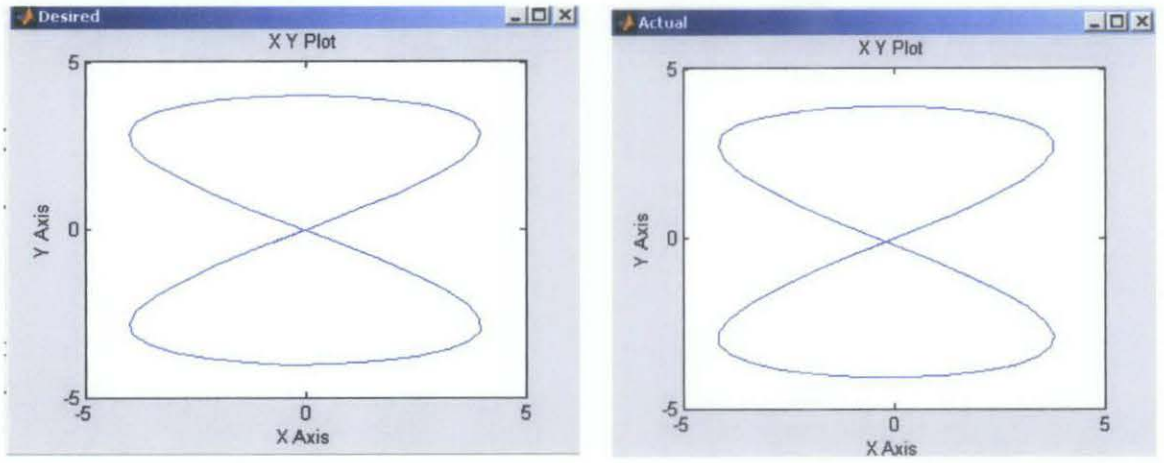

Figure 19 Desired and actual eight trajectory

After simulating the model for T=252ms, the trajectory obtained is an 8-figure. The desired and actual has an almost same value.

4.2.2.3 Straight Line Trajectory

The model is again simulated, but now it is using the following parameters:

$$Xd = 0$$

$$Yd = t$$

$$T = 15$$

The Figure 20 is obtained. The graph shows that the desired trajectory begins at (0,0) and it is set to move to (0,15) following a straight line. The Actual trajectory obtained shows the exact movement as required by the desired trajectory.

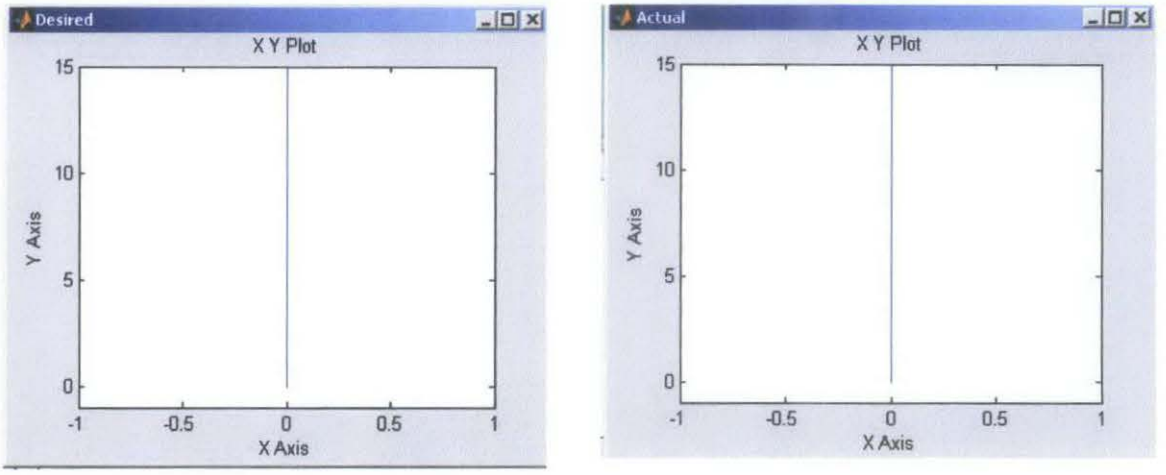

Figure 20 Desired and Actual straight line trajectory

4.3 LEADER-FOLLOWER FORMATION

4.3.1 Separation-Bearing Controller

For the Leader-follower formation, the separation-bearing controller is used. Figure 21 shows the model developed using the equation:

$$v_f = \rho - d\omega_f \tan\gamma$$

$$\omega_f = \frac{\cos\gamma}{d}(k_a l_{lf}(\vartheta_{lf}^d - \vartheta_{lf}) - v\sin\vartheta_{lf} + l_{lf}\omega_l + \rho\sin\gamma$$

$$\rho = \frac{k_b(l_{lf}^d - l_{lf}) + v_1\cos\vartheta_{lf}}{\cos\gamma}$$

$$\gamma = \vartheta_{lf} + \theta_l - \theta_f$$

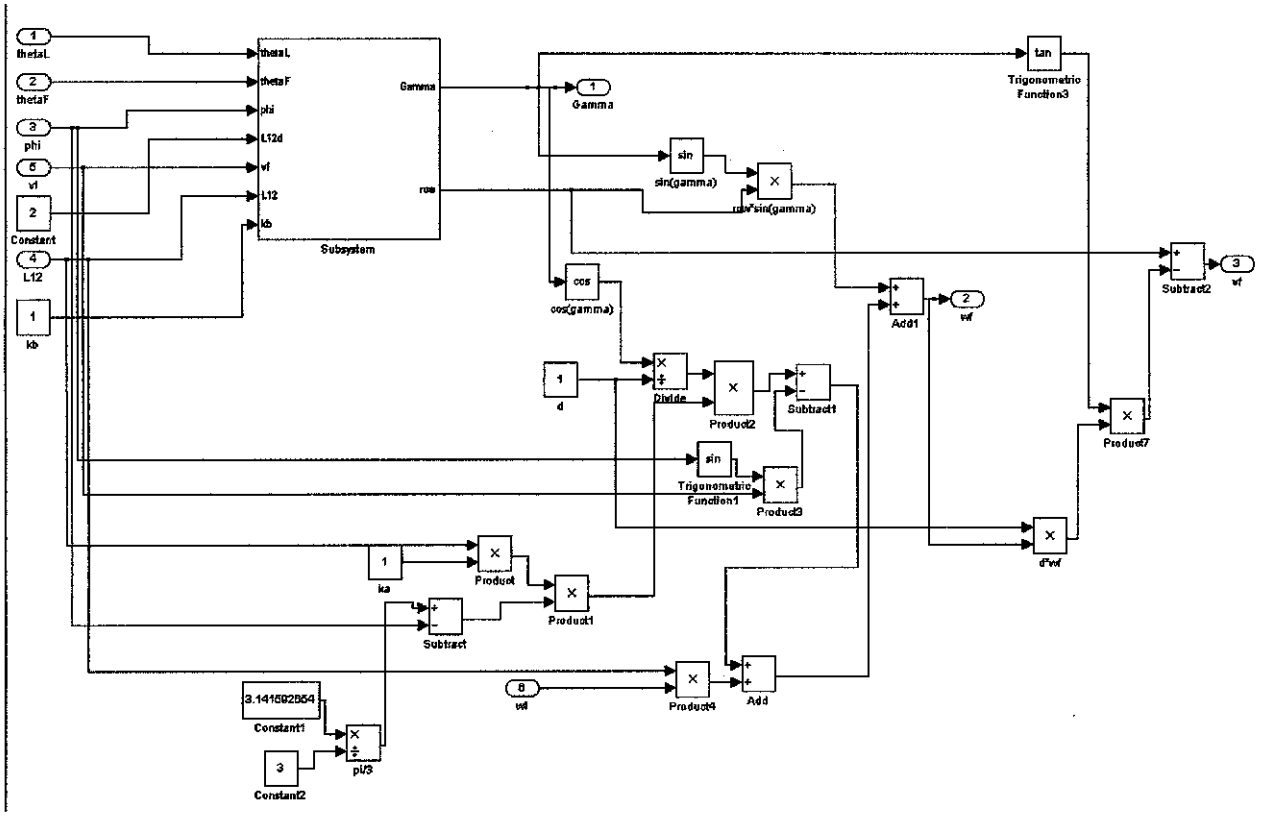

Figure 21 Separation-Bearing Controller Model

To simulate the separation-bearing controller, the following assumptions are made,

$$l_{lf}^{d} = 2$$

$$\vartheta_{lf}^{d} = \frac{\pi}{3}$$

$$k_a = k_b = 1$$

$$d = 1$$

and the figure 22 is obtained

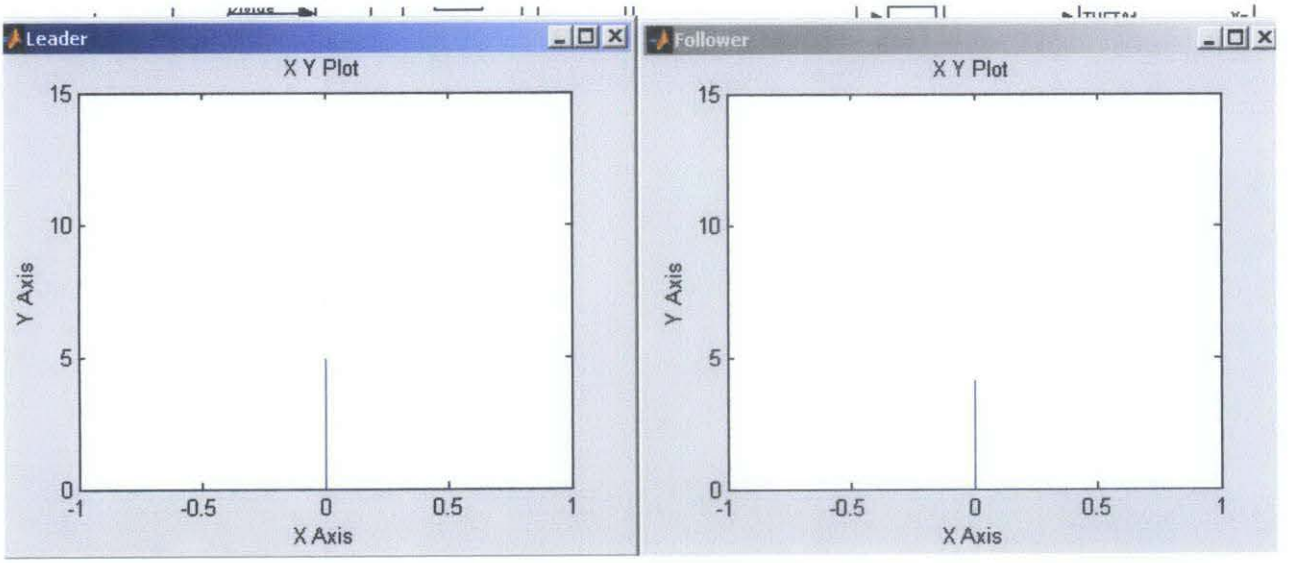

Figure 22 Leader-Follower movement graph

4.4 OTHER ROBOT PATH PLANNING

The leader robot is robot is required to generate a path that will be followed by the follower. There are two methods to achieve this. They are by using cubic and heating functions path planning.

4.4.1 Cubic Path Planning

The cubic polynomial function position model is as the following, Figure 16.

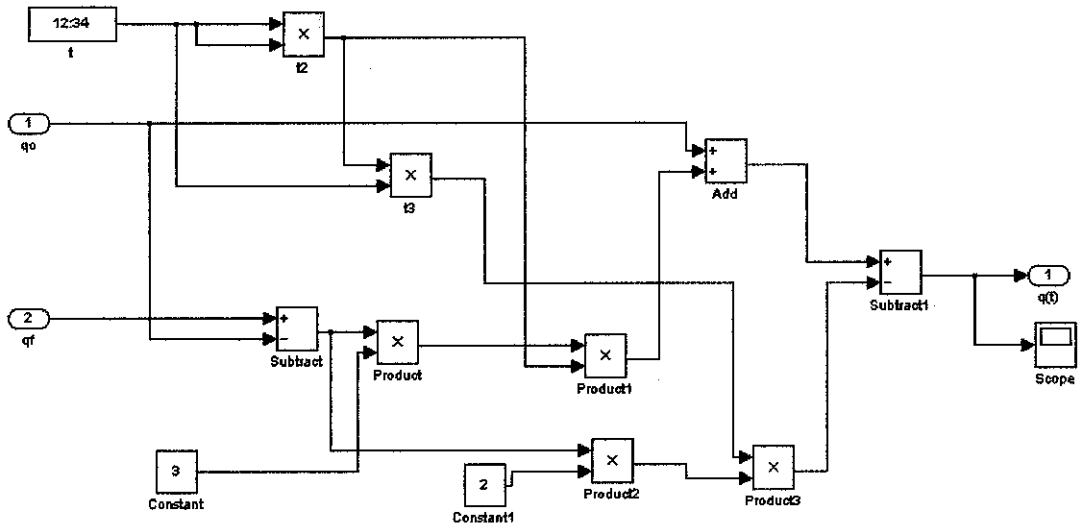

Figure 23 Cubic path planning model

After simulating the model using a sin wave as the input the following graph represented by Figure 17 is obtained at the output. Figure 17 shows that the robot path generated is not smooth.

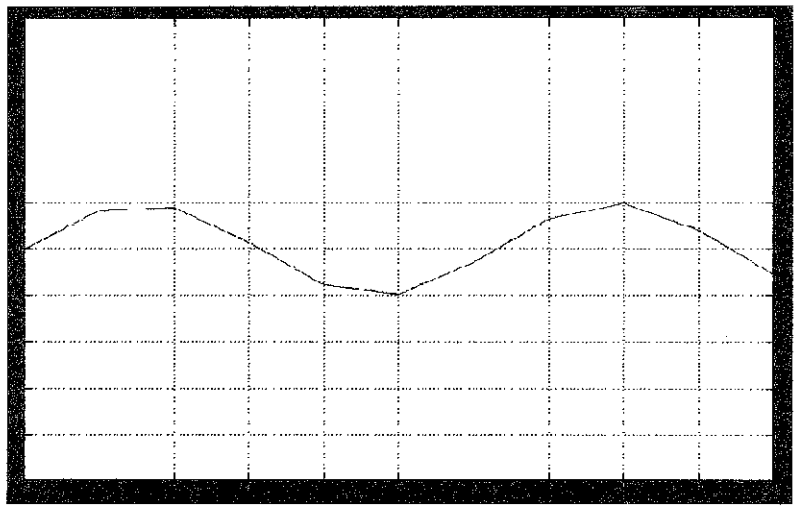

Figure 24 Cubic path planning output graph

4.4.2 Path Planning using Heating Functions

The following model shows the model developed for path planning using Heating Functions. The model also uses the desired X and Y positions used by the follower. This is to ensure that the positions are the same.

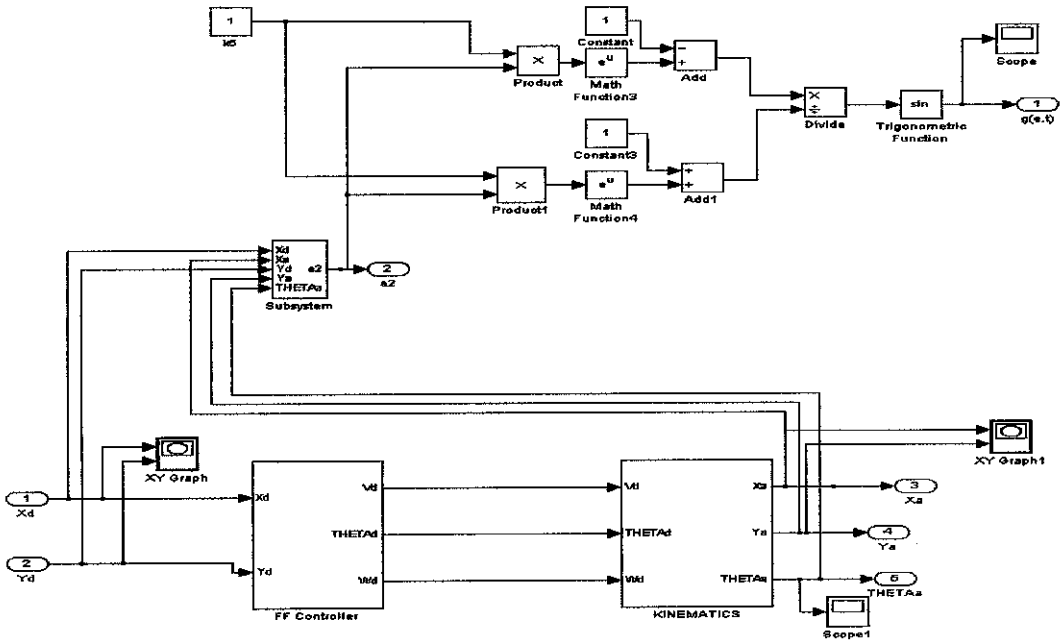

Figure 25 Heating Function Model

Figure 19 shows the curve of heating functions given by Equation 2.33 and 2.34. The simulation was done using ramp function and k used is 1

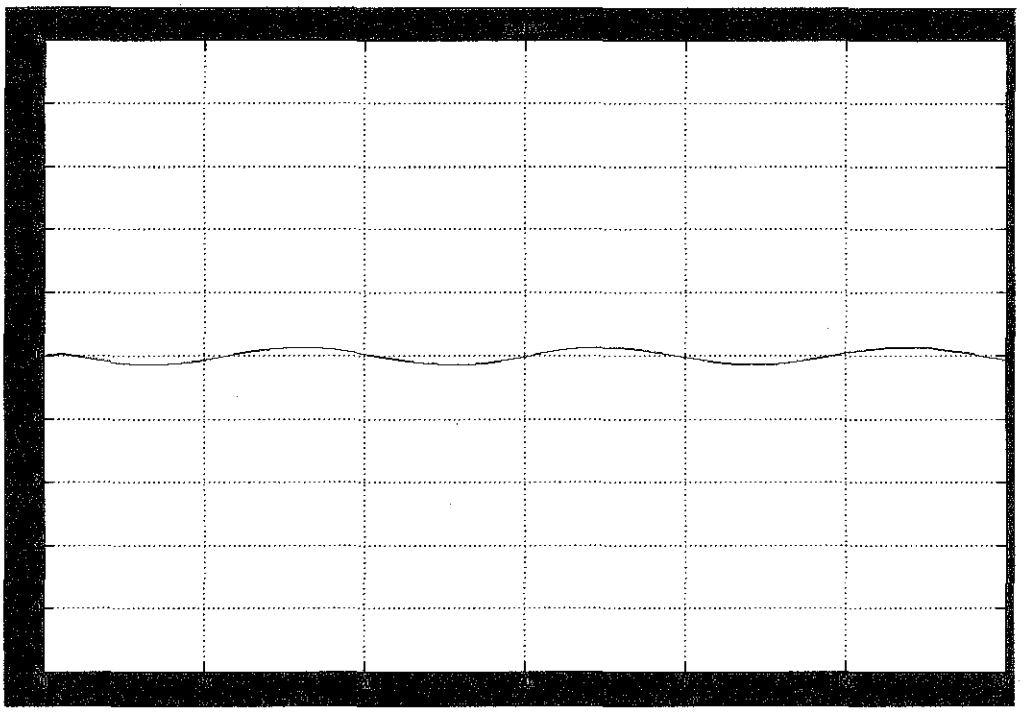

Figure 26 Heating function output trajectory

4.5 BLUETOOTH COMMUNICATION

4.5.1 User Datagram Source Code

After considering User Datagram Protocol against Transmission Control Protocol, it is decide that the project will use UDP to make the game run smoother. The source code is included in Appendix B.

4.6 DISCUSSION

This section will discuss the results obtained in the previous section. It will compare and evaluate which methods are best used and what the does it mean by the output of the graphs.

4.6.1 Mathematical Model for Robots Movement

In Figure 11, the whole follower model is shown. It consists of the feedforward controller and the kinematics model.

The first block explained is the feed forward controller. The feed forward controller will feed the desired positions and theta which will be used to calculate the velocity and w. This model will reside in the Leader Robot. In the figure, there is a block which states compare to zero. This comparator will compare its input to zero and it will multiply it by a small value. This is done to avoid it to be at zero as it will be undefined in the integral part.

The second block is the Kinematics Model. The kinematics model is used to move the robot and it resides in the follower robots. Once the follower robot receives the velocity and w from the leader robot, it will calculate the actual x and y position and move to that particular position. This position will be compared to the initial desired position to ensure that the robot moves to the correct position.

4.6.2 Slope Trajectory

The graph shows that the robot is moving linearly to the desired position. The robot moves from position (0,0) to (0,1). The desired and actual position graph shows that the positions are almost the same with minor errors. The graph also shows that there is very little disposition. The Figure 27 shows the error obtained.

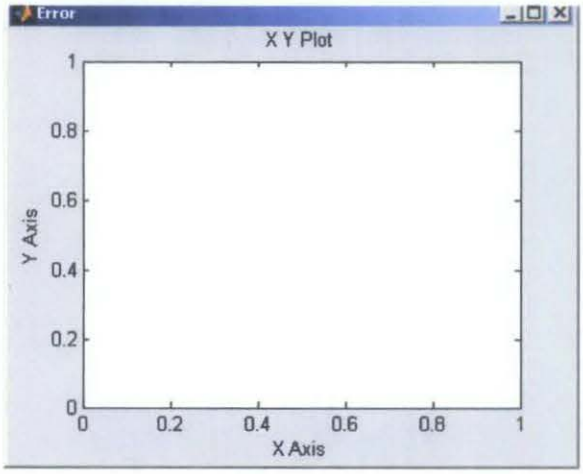

Figure 27 Error for slope trajectory

From the figure, we can see that there is no error at all. Therefore the actual and desired graph is exactly the same.

4.6.3 Eight Trajectory

Comparing the desired and actual of the eight trajectory it looks almost the same. However, according to the error graph shown below, it shows that there are minor disposition. There are at a maximum of less than 0.4 disposition in X and Y direction.

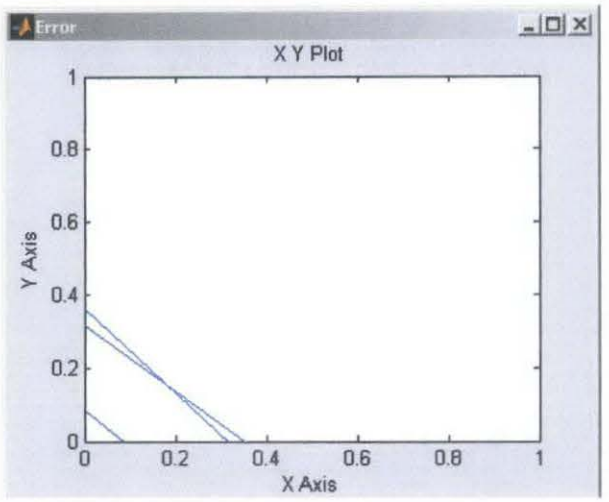

Figure 28 Eight trajectory Error Graph

4.6.4 Straight Line Trajectory

The graph obtained from the simulation of the straight line model shows that the trajectory is generated from and initial position (0,0) to a final position (0,15). Both the actual and desired has achieved the same initial and final positions. However, Figure 29 will show the exact comparison of the desired and actual trajectory.

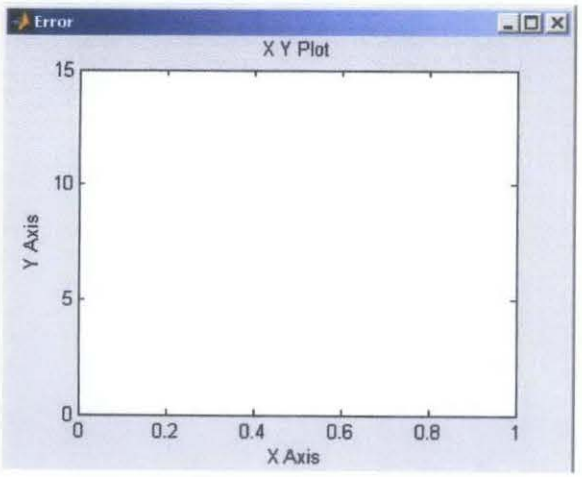

Figure 29 Error of Straight Line Trajectory

The figure shows that there are no errors in the trajectory obtained.

4.6.5 Leader-Follower Formation

The figure obtained from the separation-bearing controller model shows that the leader moves from an initial position of (0,0) to a final position of (0,5) for a simulation time of T=5. however, using the same simulation time, the follower moves from and initial pposition of (0,0) toa final position of (0,4). There is a distance of 1 which is the distance set to avoid collision between the leader and follower. This shows that the follower is following the leader.

4.6.6 Cubic Path Planning Output Graph

From the graph, we can see that the graph is not a smooth curve. This is due to derivative of acceleration, called jerk. Impulsive jerk may reduce tracking accuracy.

4.6.7 Path Planning using Heating Functions.

The heating functions graph shows that a smooth trajectory is developed as compared to the cubic path planning. After comparing both of these methods, path planning using heating functions will be used in the game. This is because this model gives a smooth trajectory and it also incorporates the nonholomic constraints in producing the desired trajectory.

4.6.8 Bluetooth Communication

The Bluetooth communication platform for the leader and follower to communicate has not been achieved. This is because each Bluetooth dongle has proprietary software such as IVT Bluesoleil which is used to communicate with the computer's operating system. The proprietary software however is unable to communicate directly with MATLAB because MATLAB does not have the features that support Bluetooth. Therefore, for future work it is recommended that a program is

developed to act as an intermediate between the proprietary software and MATLAB using JAVA or C language.

Since the communication platform is not achieved, all data are sent from the leader to the follower using hardwire for simulation purposes.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

As a conclusion, the project has achieved the objective of creating the follower and leader models. It has also incorporated the theory of collaborative autonomous agents. However, the project has not achieved the communication platform desired in order to show the required game.

For future development of the game, the game could use holonomic models and compare the results obtained. Besides that, the game could try to remove the assumptions mentioned in Chapter 1 one by one such as the ball could only move in two dimension and try to make it able to move in 3 dimensions. By removing the assumptions, it will make the game more real and complex. The ultimate objective in robotics is to make it as humanly as possible by eliminating the errors.

It is recommended that the project is divided in to two main areas for future development. The first area would be the simulation regarding the robots and the second area would be setting up the communication platform.

The robots could also try to be developed using other type of controller such as the fully linearized dynamic controller which might drive all the errors to zero.

Finally, to improve on the communication platform, a program has to be developed using JAVA or C to communicate between MATLAB and Bluetooth's proprietary software.

REFERENCES

1. Roland Siegwart and Illah R. Nourbakhsh 2004, Introduction to Mobile Autonomous Robots.

2. Wikipedia, The Free Encyclopedia , 4 May 2007 http://en.wikipedia.org/wiki/Nonholonomic_system

3. Richard M. Murray, Muruhan Rathinam and Wilem Sluis 1995, Differential Flatness of Mechanical Control Systems: A Catalog of Prototype Systems

4. A. De Luca, G. Oriolo and C. Samson, Feedback Control of Nonholonomic Car-Like Robot

5. M. Van Nieuwstadt, M. RAthinam and R. M. Murray a995, Differential Flatness and Absolute Equivalence of Nonlinear Control Systems

6. http://me598.wikidot.com/an-introduction-to-differential-flatness

7. Robert Morrow, Bluetooth Operation and Use

8. James B. DAbney and Thomas L. Harman, Mastering Simulink

9. Wikipedia, The Free Encyclopedia , 14 August 2007 <http://en.wikipedia.org/wiki/Kinematics>

APPENDICES

APPENDIX A: M-FILE FOR CREATING FIELD

```
function ROBOSOCCER

%% Create figure
figure1 = figure('Color',[0.502 0.502 0.502]);

%% Create axes
axes1 = axes(...
  'Color',[0 0.498 0],...
  'XGrid','on',...
  'XMinorTick','on',...
  'YGrid','on',...
  'YMinorTick','on',...
  'Parent',figure1);
axis(axes1,[-12 12 -9 9]);
title(axes1,'ROBOT SOCCER');
xlabel(axes1,'X-AXIS');
ylabel(axes1,'Y_AXIS');
hold(axes1,'all');

%% Create annotation
annotation1 = annotation(...
  figure1,'line',...
  [0.5175 0.5175],[0.9211 0.1107],...
  'LineWidth',2,...
  'Color',[1 1 1]);

%% Create ellipse
centre = annotation(figure1,...
  'ellipse',[0.4527 0.4279 0.1307 0.1829],...
```

```
 'LineWidth',2,...
 'EdgeColor',[1 1 1]);

%% Create rectangle
D_A = annotation(figure1,...
 'rectangle',[0.1296 0.2903 0.09671 0.4547],...
 'LineWidth',2,...
 'EdgeColor',[1 1 1]);

%% Create rectangle
D_B = annotation(figure1,...
 'rectangle',[0.8076 0.2919 0.09874 0.453],...
 'LineWidth',2,...
 'EdgeColor',[1 1 1]);

%% Create annotation
CENTRE_LINE= annotation(figure1,'line',[0.1286 0.1336],[0.6091 0.6141]);

%% Create annotation
GOAL_A2 = annotation(...
 figure1,'line',...
 [0.1029 0.1286],[0.6107 0.6107],...
 'LineWidth',2,...
 'Color',[1 1 1]);

%% Create annotation
GOAL_A1 = annotation(...
 figure1,'line',...
 [0.1029 0.1286],[0.4262 0.4262],...
 'LineWidth',2,...
 'Color',[1 1 1]);
```

```
%% Create annotation
GOAL_B2 = annotation(...
  figure1,'line',...
  [0.9064 0.9321],[0.6107 0.6107],...
  'LineWidth',2,...
  'Color',[1 1 1]);

%% Create annotation
GOAL_B1 = annotation(...
  figure1,'line',...
  [0.9033 0.929],[0.4279 0.4279],...
  'LineWidth',2,...
  'Color',[1 1 1]);

%% Create ellipse
ball = annotation(figure1,...
  'ellipse',[0.5093 0.505 0.01646 0.02461],...
  'FaceColor',[1 1 1]);

%% Create rectangle
A1 = annotation(figure1,...
  'rectangle',[0.3488 0.5973 0.01337 0.02013],...
  'FaceColor',[0.8471 0.1608 0]);

%% Create rectangle
B1 = annotation(figure1,...
  'rectangle',[0.644 0.6007 0.01337 0.02013],...
  'FaceColor',[0.749 0.749 0]);

%% Create rectangle
```

```
B2 = annotation(figure1,...
  'rectangle',[0.6409 0.3758 0.01337 0.02013],...
  'FaceColor',[0.749 0.749 0]);

%% Create rectangle
A2 = annotation(figure1,...
  'rectangle',[0.3477 0.3255 0.01337 0.02013],...
  'FaceColor',[0.8471 0.1608 0]);
```

APPENDIX B: UDP SOURCE CODE

For the server:

```
#include <winsock.h>
#pragma comment(lib,"ws2_32.lib")

int main()
{
        WSADATA wsaData;
        SOCKET RecvSocket;
        sockaddr_in RecvAddr;
        int Port = 2345;
        char RecvBuf[1024];
        int  BufLen = 1024;
        sockaddr_in SenderAddr;
        int SenderAddrSize = sizeof(SenderAddr);
        WSAStartup(MAKEWORD(2,2), &wsaData);
        RecvSocket = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
        RecvAddr.sin_family = AF_INET;
        RecvAddr.sin_port = htons(Port);
        RecvAddr.sin_addr.s_addr = INADDR_ANY;
        bind(RecvSocket, (SOCKADDR *) &RecvAddr, sizeof(RecvAddr));
        recvfrom(RecvSocket,RecvBuf, BufLen,0,(SOCKADDR
*)&SenderAddr,&SenderAddrSize);
        printf("%s\n",RecvBuf);
        closesocket(RecvSocket);
        WSACleanup();
}
```

For the client:

```
#include <winsock.h>
#pragma comment(lib,"ws2_32.lib")

int main()
{
        WSADATA wsaData;
        SOCKET SendSocket;
        sockaddr_in RecvAddr;
        int Port = 2345;
        char ip[] = "127.0.0.1";
        char SendBuf[] = "hello";
        WSAStartup(MAKEWORD(2,2), &wsaData);
        SendSocket = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
        RecvAddr.sin_family = AF_INET;
        RecvAddr.sin_port = htons(Port);
        RecvAddr.sin_addr.s_addr = inet_addr(ip);
        sendto(SendSocket,SendBuf,strlen(SendBuf),0,(SOCKADDR *)
&RecvAddr,sizeof(RecvAddr));
        WSACleanup();
}
```

APPENDIX C: FYP 1 GANTTCHART

NAME: NOOR ATIKAH BINTI MOHD FAUZIE
MATRICS NUMBER: 5604
FINAL YEAR PROJECT 1 GANTT CAHRT
SUPERVISOR: DR. HERMAN AGUSTIAWAN

TASK	JULY				AUGUST				SEPTEMBER				OCTOBER					NOVEMBER			
			1	2	3	4	5	6	7	MID SEM	8	9	10	11	12	13	14	15	16		
TITLE SELECTION AND APPROVAL			■	■																	
MODEL BALL AND FOLLOWER ROBOTS				■	■	■															
MODEL AND SIMULATE LEADER ROBOTS					■	■	■	■	■	■	■	■	■	■	■						
SUBMISSION OF PRELIMNARY REPORT						■															
PATH AND TRAJECTORIES PLANNING						■	■	■	■	■	■	■	■	■	■						
*SEMINAR 1								■													
BLUETOOTH SETUP AND STRATEGY									■	■	■	■	■	■	■						
SUBMISSION OF PROGRESS REPORT											■										
*SEMINAR 2													■								
SUBMISSION OF DRAFT REPORT														■							
FINE TUNING AND ADJUSTMENTS													■	■	■	■					
SUBMISSION OF INTERIM REPORT																	■				
ORAL PRESENTATION																		■			

*Subject to change

NAME: NOOR ATIKAH BINTI MOHD FAUZIE
MATRICS NUMBER: 5604
FINAL YEAR PROJECT 1 GANTT CAHRT
SUPERVISOR: DR. HERMAN AGUSTIAWAN

TASK	JAN		3	4	FEB 1	2	3	4	MARCH 1	MID SEM	3	4	APRIL 1	2	3	4	MAY 1	EXAM	JUNE 1		
TITLE SELECTION AND APPROVAL			■	■																	
MODEL BALL AND FOLLOWER ROBOTS				■	■	■															
MODEL AND SIMULATE LEADER ROBOTS					■	■	■	■	■	■	■	■	■	■	■						
SUBMISSION OF PRELIMNARY REPORT						■															
PATH AND TRAJECTORIES PLANNING						■	■	■	■	■	■	■	■	■	■						
SUBMISSION OF PROGRESS REPORT 1							■														
BLUETOOTH SETUP AND STRATEGY									■	■	■	■	■	■	■	■					
SUBMISSION OF PROGRESS REPORT 1											■										
SUBMISSION OF DRAFT REPORT													■								
*SEMINAR 1															■						
FINE TUNING AND ADJUSTMENTS													■	■	■	■	■				
SUBMISSION OF FINAL REPORT																■	■				
SUBMISSION OF TECHNICAL REPORT																■	■				
ORAL PRESENTATION																			■		
SUBMISSION OF INTERIM REPORT																			■		

*Subject to change