TrueTime Simulation of Collision Avoidance in Vehicular Traffic using Coordinated Control

A Thesis submitted to the Faculty of Engineering & Technology, Jadavpur University in Partial fulfilment of the requirements for the Degree of Master of Engineering in Power Engineering

Submitted by

ATANU KHAN

*Examination Roll Number: M4POW*1615

Registration Number: 129434 *of* 2014-15

Under the Guidance of

Prof. Dr. Amitava Gupta

Faculty of Engineering and Technology

Department of Power Engineering

Jadavpur University

Kolkata-700098

2016

Department of Power Engineering
Faculty of Engineering & Technology
Jadavpur University
Kolkata-700098, India

Certificate of Recommendation

*We hereby recommend the thesis, entitled "**TrueTime Simulation of Collision Avoidance in Vehicular Traffic using Coordinated Control**" prepared under the guidance of **Prof. Dr.Amitava Gupta**, Department of Power Engineering, Jadavpur University, Saltlake Campus, Kolkata, by **Atanu Khan** (Examination Roll Number M4POW1615 and Registration Number 129434 of 2014-2015), be accepted in partial fulfillment of the requirement of the Degree of Master of Engineering in Power Engineering of Jadavpur University.*

Prof. Dr. Amitava Gupta

Department of Power Engineering,

Jadavpur University

Counter signed by:

Head

Department of Power Engineering,

Jadavpur University

Dean

Faculty of Engineering & Technology

Jadavpur University

Department of Power Engineering
Faculty of Engineering & Technology
Jadavpur University
Kolkata-700098, India

Certificate of Approval

*The foregoing thesis, entitled as "**TrueTime Simulation of Collision Avoidance in Vehicular Traffic using Coordinated Control**" is hereby approved by the committee of final examination for evaluation of thesis as a creditable study of an engineering subject carried out and presented by **Atanu Khan** (Examination Roll Number M4POW1615 and Registration Number 129434 of 2014-2015, in a manner satisfactory to warrant its acceptance as a prerequisite to the Degree of Master of Engineering in Power Engineering. It is understood that by this approval, the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve the thesis only of the purpose for which it is submitted.*

Committee of final examination for evaluation of thesis:

_____

_____

Department of Power Engineering
Faculty of Engineering & Technology
Jadavpur University
Kolkata-700098, India

Declaration of Originality and Compliance of Academic Ethics

I hereby declare that this thesis contains Literature Survey and original research work carried out by myself, as part of my Degree of Master of Engineering in Power Engineering.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name: ATANU KHAN

Exam Roll Number: M4POW 1615

Thesis Title: TrueTime Simulation of Collision Avoidance in Vehicular Traffic using Coordinated Control

Signature with Date:

ACKNOWLEDGEMENT

I express my gratitude and sincere thanks to my supervisor Prof. Dr. Amitava Gupta, Professor of the Department of Power Engineering of Jadavpur University for his constant motivation and support during the course of my thesis. I truly obligated to him for his esteemed guidance and encouragement from the beginning to the end of the thesis. I am extremely thankful to him for providing me with necessary guidance to shape up the problem and critical analysis to evolve the solution.

With gratitude and respect, I extend my profound thanks to my respected seniors Kaushik Halder, Debayan Bose, Sohon Banerjee and Soumya Dasgupta, Department of Power Engineering, Jadavpur University, Salt Lake, Kolkata, for their whole-hearted support and co-operation.

I am especially indebted to all my classmates whose constructive criticism and ideas helped me in development of my project work. And I would like to thank all whose direct and indirect support helped me completing my thesis in time.

Above all, I thank my parents, whose blessing and love have guided me all throughout my education and life.

Regards,

Atanu Khan.

Master of Engineering in Power Engineering

Examination Roll Number: M4POW1615

Reg. Number: 129434 OF 14-15

Abstract

With the rapid progress in technology, research and implementation of digital computer based modern control techniques, has gained paramount importance for the operations of any system, over the past few decades. Automatic control of unmanned vehicle is one such research domain, where the computer based control technology has found its importance. This dissertation puts forward an unique concept of automatic control in the field of unmanned vehicular technology where the purpose of the control algorithm is to prevent occurrence of a precarious circumstance of collision which may occur when the two freely moveable vehicles comes close to one another. Control algorithm adopted in this dissertation is Coordinated Control, whose objective would be to prevent the collision of two vehicles, moving freely on an undefined trajectory but on a confined space. Every vehicle has a user-specified range, preferably called *Safety Zone* around it, which is provided by employing Bluetooth device in the vehicles, and the trajectory of the vehicles has been modelled using the concept of Robot Kinematic Theory. The proposed Coordinated Controller has been designed using basic Coordinate Geometry and Pythagorean Theorem approach. The entire system with controller has been modelled in MATLAB Simulink environment. MATLAB TrueTime Toolbox has been used here for simulation of transmission and reception of data from one vehicle to another where a data element contains state vector of vehicles. The performance of the controller has been validated through the credible MATLAB-TrueTime based simulations results.

Keywords- Anti-collision, Automatic control, Coordinate Control, Robot Kinematic, TrueTime

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Literature Survey

Coordinate control is one of the most reliable techniques for automatic control. In recent research automatic control has become most popular due to the absence of human error. Automatic control is useful in various sectors such as power system **[1]-[2]**, Intelligent Control Systems**[3]**; Stochastic Control, Neuro-fuzzy Control Systems**[4]**, Automatic Control of Chemical Processes, Automotive Control Systems, Thermal System Control, Robot and Manipulator Control**[5]**, Process Control, Aerospace Control Systems**[6]**, Motion and Navigation Control, Traffic and Transport Control**[7]**, Defence and Military Systems Control, Studies on nuclear systems control**[8]**, Control analysis of Social and Human Systems, Biomedical control systems. Real-Time Systems control, Real-Time and Fault-Tolerant Systems **[9]**, Control of linear/nonlinear systems and Stability **[10]**.

Vehicular system control is contemporary research area for automatic control among the various system maintained above because the demand of automated control vehicle is rising. Due to the increase of number of man driven vehicles, the traffic jams, rough driving also increasing, that makes impulse increase of accidents. According to Willie D. Jones as reported in **[11]**, in every second a person dies by a car crash. There is various ways to control vehicle systems automatically but coordinate control is the most proficient.

Most of the automated control for the vehicle system is done by co-ordinated control (path tracking or path following). This path following or path tracking control is also one type of formation control where vehicles follow a specific track or behaviour, in other words it follows a command decided by user. There are several ways for formation control of vehicles, out of which three fundamental approaches have gained paramount importance over the past decades. They are namely:

1. Virtual structure approach

In virtual structure approach every vehicle is controlled by a virtual robot. This virtual robot designs a predefined path for each and every vehicles and it controls every vehicles to converge on its predefined path through its controller. Here, the control signals are the error of state vector of each vehicles (state vector contains present co-ordinate, orientation and steering angles for each vehicles). The main advantages of the virtual structure approach is that it is reasonably easy to prescribe the coordinated behaviour for the group of vehicles, and during the steer the formation can be maintained very well, because the virtual structure can act as a whole in a given direction with some given orientation and maintains a rigid geometric relationship among multiple vehicles. However, the potential applications are limited if the formation has to maintain the exact same virtual structure all the time, especially when the formation shape is time varying or needs to be frequently reconfigured.

Here the entire formation is considered as a rigid body. The entire structure follows a required predetermined dynamics, and the required structure motion translated to individual trajectories, for each one of vehicles structure **[12]**. A nonlinear control law was presented for formation control of a group of vehicles, using Lyapunov approach and the vehicle bicycle model **[13]**.

In **[14]** the authors present a co-ordinated path following control problem of nonholonomic vehicles, here the problem can be seen as a subtask of the general formation control problem. The solution to this problem is defined by a two layer architecture. In the inner layer, the kinematic model of the vehicle is used to solve the problem of position and orientation, with respect to a desired path. In this inner layer, the steering rate is utilized to form chained model control. And in the outer layer, the coordinated problem is solved by using a virtual vehicle to each controlled vehicle. The virtual vehicle sets predefined tasks which is required for coordination. The control input is the time derivative of the parameterized path of the virtual vehicle. Based on the kinematic model, a speed input is combined with the input to converge the controlled vehicle with the virtual vehicle.

Another similar work has been shown in **[15]** where the authors' deals with a group of car like vehicles, and each vehicle is represented by the bicycle kinematic model. Lyapunov approach has been introduced for nonlinear control law. The position errors of all vehicles with respect to the desired paths, and the coordination errors that trace deviations from the reference geometric formation are included in the Lyapunov function.

A combination of the virtual structure and path following approaches has been used to derive the formation architecture in **[16]**. A formation controller is used for the kinematic model of two-degree-of-freedom unicycle-type mobile robots. Here the approach is extended to consider the formation controller for including the physical dimensions and dynamics of the robots. Here the controller is designed in such a way that to synchronize the robot's motion, the path derivative is left as a free input and to justify this simulation results with three robots has been included to show the performance of control system. Here Lyapunov function has been used to design the controller.

The path tracking controller for an articulated vehicle (a semitrailer-like vehicle) using time scale transformation and exact linearization has been designed in **[17]**. Here the basic advantage is, during the forward and/or backward movement the proposed controller allows articulated vehicles to follow arbitrary paths consisting of arcs and lines. So here we can observe the experimental result of the 8 shaped path tracking control of the articulated vehicle, where moving backward is also presented. Here the controller has been designed as follows. First the authors define a new time scale which is identical to the distance along the desired path and using a state equation with this time scale they describe the model of the vehicle. Then the authors linearize this state equation with appropriate state coordinate transformation and feedback. Finally, for the linearized system a linear controller is designed.

Back stepping method is applied for path tracking control in **[18]**. In this paper, the authors have proposed the formation control problem of a group of wheeled mobile robots with a virtual robot. According to the requirement desired position of each robot is calculated from the virtual structure, which is determined by the virtual robot and the desired formation shape. Then, virtual orientation control input is applied and a nonlinear control algorithm is designed using backstepping technique. After that a mathematical stability is analysed adopting the cascaded system theorem. Then a strict stability analysis is done which shows the formation tracking errors are globally uniformly asymptotically convergent to zero. In the virtual structure approach there is no hierarchy in the formation, no leaders or followers, and the entire group moves as a single rigid body. A virtual structure constituted by all robots tracks forms the set of predefined trajectories and each robot tracks the corresponding trajectory generated by the formation shape.

2. Behaviour -based approach

In the behaviour based approach, different tasks are defined for each robot in the group such as obstacle avoidance, destination arrival, formation keeping etc. Each task generates a different motion command for the robot based on relative importance. The differences between different behaviour based approaches are due to the way different motion commands are combined to a single control input. The behavioural approach for multi-robot teams is described where the control strategies for goal seeking, collision avoidance and formation maintenance are achieved with the implementation of formation behaviours with other navigational behaviours. The advantage is that control strategies can be derived naturally when vehicles have multiple competing objectives, and an explicit feedback is also included through communication between neighbours.

The disadvantages are that it is difficult to analyze the approach mathematically and guarantee the group stability In competitive methods the group behaviour cannot be explicitly defined, and a supervisor is in charge of ranking priorities to different tasks, and then, only the most important task is preformed, one task at a time.

A neural architecture for learning Coordination of different behaviours in a situated agent has been described in **[19]**.Behaviour-oriented approaches define the control of an agent directly in terms of its tasks. Here key challenge is to manage the agent's ongoing tasks so that action dispute is minimized and desired levels of compliance with overall goals are achieved. The mechanisms has presented for adapting the coordination strategy through short and long-term suppression and time varying performance feedback. Finally, to demonstrate the effectiveness of this method there has preliminary experimental results for a simulated robot. Behaviour-based methods have come out as a viable alternative to traditional approaches for designing intelligent agents and autonomous robots. These systems split responsibility for making decisions to carry out an agent's mission into multiple behaviour modules, for which each dedicated to solve a distinct task. One approach to resolving this is to learn when behaviours should be activated **[20]**. It has been completely discussed by Mataric **[21]** that learning algorithms often fail to converge when presented with multiple goals, so such successes have largely been limited to single-task domains.

A novel feedback control law for coordinating motions of multiple holonomic mobile robots to conquer or enclose a target by forming troop formations has shown in **[22]**. This motion coordination is a cooperative behaviour which provides security against intruders in surveillance areas. In this control law each robot has its own coordinate system and to

accomplish this cooperative behaviour without making any collision it senses a target or intruder, other robots and obstacles. All the robots are asymptotically stabilized and they form formations enclosing a target although there has no centralized controller and each robot has local feedback that is relative position feedback. Each robot especially has a vector called "a formation vector" and formations are controllable by the vectors. There is a reactive control framework for determining the formation vectors, in which robots have reactions heuristically designed according to this cooperative behaviour. Forming formations is a task of multiple mobile robots in a sense of security against invaders, which is capturing or enclosing an invader and prevention of invasions by forming formations **[23]**.

Architecture for controlling mobile robots is described in **[24]**. Layers of control system are built to let the robot operate at increasing levels of competence. With asynchronous modules, layers are made up that communicate over low-band width channels. Each module is an instance of a fairly simple computational machine. Higher-level layers can link up the roles of lower levels by suppressing their outputs. However, when higher levels are added lower levels continue to function. The system has been used to control a mobile robot wandering around unconstrained areas; eventually it is intended to control a robot that wanders a large area using an on-board arm to perform simple tasks. There the advantages are concerning robustness, compsability and testability.

A reactive behaviour that implements formations in multi-robot teams are delivered and evaluated in **[25]**. To enable a robotic team that reaches navigational goals, avoids hazards and simultaneously remain in formation, the formation behaviours are integrated with other navigational behaviours. The behaviours has implemented in simulation, on robots in the laboratory. The technique has been incorporated with the Autonomous Robot Architecture (AuRA). The results demonstrate their appropriateness in different types of task environments and the value of various types of formations in autonomous, human-led and communications-restricted applications.

3. Leader following approach

In leader-follower approach, some robots are defined as leaders, while others act as followers. The leader's task is concerned with some group objectives, such as trajectory tracking, navigation, obstacle avoidance etc., where the followers are only required to follow the leaders. A great deal of research has been done using the leader-follower approach. An advantage of the leader following approach is that it is easy to understand and implement.

In addition, the formation can still be maintained even if the leader is distracted by some disturbances. However, the disadvantage related to this approach is that there is no explicit feedback to the formation; that is, there is no explicit feedback from the followers to the leader in this case.

A combined kinematic or torque control law has been developed for leader-follower based formation control using backstepping to accompany the dynamics of the robots and the formation with kinematic-based formation controllers**[26]**. The asymptotic stability of the entire formation is ensured using Lyapunov theory. Here kinematic controller is developed for the control strategies of single mobile robots and the idea of virtual leaders. The virtual leader is replaced with a physical mobile robot leader and the assumption of constant reference velocities is removed. A novel approach is taken to develop the dynamical controller such that the torque control inputs for the follower robots include the dynamics of its leader as well as the dynamics of the follower robot, and in this case it is considered that all robot dynamics are known. The sensory information has used to calculate velocity control inputs in both **[27]** and **[28],** in **[27]** local sensory information and in **[28]** vision based approach to leader-following has been undertaken. A modified leader follower control is introduced in **[29]** where Cartesian coordinates are used rather than polar.

A Receding-Horizon-Leader-Follower (RHLF) control framework to solve the formation problem of multiple non-holonomic mobile robots with a rapid error convergence rate has been shown in **[30]**. To maintain the desired leader-follower relationship, there is a Separation Bearing Orientation Scheme (SBOS) for two-robot formations and Separation-Separation-Orientation-Scheme (SSOS) for three robot formations in deriving the desired postures of the followers. Unlike the other leader–follower approaches in the existing literature, the orientation deviations between the leaders and followers are explicitly controlled in this framework. It enables to successfully solve formation controls when robots move backwards, which is termed as a formation backwards problem in this paper. Further, they have proposed to incorporate the receding-horizon scheme into their leader–follower controller to generate a fast convergence rate of the formation tracking errors.

The work reported in **[31]** deals with leader–follower formations of nonholonomic mobile robots, introducing a formation control strategy alternative to those existing in the literature. Suitable constraints restrict the set of leader's possible paths and admissible positions of the follower with respect to the leader. Here the proposed strategy has the characteristics that the follower position is not rigidly fixed with respect to the leader but varies in proper circle arcs centred in the leader reference frame. In the proposed leader–

follower control, the follower tracks a reference trajectory based on the leader position and predetermined formation without the need for leader's velocity and dynamics in the leader–follower formation control of multiple under actuated autonomous underwater vehicles (AUVs). This is desirable in marine robotics due to weak underwater communication and low bandwidth. A virtual vehicle is constructed such that its trajectory converges to the reference trajectory of the follower. Position tracking control is designed for the follower to track the virtual vehicle using Lyapunov and backstepping synthesis. A sliding mode control has discussed in **[32].**

The problem of modelling and controlling leader-follower formation of mobile robots is also discussed in **[33]**, hence a novel kinematics model for leader-follower robot formation has formulated based on the relative motion states between the robots and the local motion of the follower robot.

Apart from the three approaches described above for the coordinated control, there have been other approaches, such as Model predictive control which are described in **[34], [35]** and **[36].** A graph theory based solution, which uses the Laplacian eigen values of the graph for stability analysis, is presented in **[37]**. A combination of the three fundamental approaches (Virtual-structure approach, behavioural and leader-follower approach) is described in **[38]**.

1.2 Reviews on TrueTime

TrueTime is a MATLAB or Simulink-based simulator for real-time control systems that has been developed at Lund University since 1999. TrueTime facilitates co-simulation of controller task execution in real-time kernels, network transmissions, and continuous plant dynamics. Some of the features of the TrueTime simulator are –

- Simulation of complex controller timing due to code execution, task scheduling, and wired/wireless network communication.
- Possibility to write tasks as M-files or C++ functions. It is also possible to call Simulink block diagrams from within the code functions.
- Network blocks (Ethernet, CAN, TDMA, FDMA, Round Robin, Switched Ethernet, FlexRay and PROFINET).
- Wireless network block (802.11b WLAN and 802.15.4 ZigBee).
- Battery-powered devices, Dynamic Voltage Scaling, and local clocks.

Various real time represent of the systems has been done by using TrueTime simulator. The use of event triggering as well as time triggering makes TrueTime a compact simulator. It is possible to simulate the temporal behaviour of a multi-tasking real-time kernel containing controller tasks which is shown in **[39]**. The controller tasks control processes modelled as ordinary Simulink blocks. Different scheduling policies have been used, such as priority driven or deadline-driven scheduling. The effects of context switching and interrupt handling are taken into account; with TrueTime here it is also possible to simulate the timing behaviour of communication networks which are used in this networked control loops.

An extension of the Embedded Systems Modelling Language (ESMoL) tool chain that automatically synthesizes Time Triggered Architecture (TTA) based TrueTime models have been shown in **[40]**. Here time invariant Simulink models are imported into the ESMoL modelling environment where they are provided with details of the desired deployment platforms. A constraint based offline scheduler then generates the static TTA execution schedules. Finally, new TrueTime models synthesized that encapsulate all of the TTA execution semantics. Using this approach it is possible to rapidly prototype, evaluate, and modify controller designs and also their hardware platforms to better understand deployment induced performance and timing effects.

A brief introduction to the TrueTime simulator and then several examples on how TrueTime can be used to simulate networked control systems are shown in **[41]**. Among the examples there are time-triggered and event-based networked controls and Ad-hoc On-Demand Distance Vector (AODV) routing in wireless ad-hoc networks are also shown. In **[42]** TrueTime toolbox (for MATLAB) has been used as a simple and easy way to realize several network types. There is a demonstration of how to setup simple TrueTime network control system with an explanation of its basic settings and parameters. In the last section they briefly compared the simulation results of motor control system which uses different network types.

In **[43]** details of all phases of adapting TrueTime simulation necessary for analysis of behaviour of WirelessHART protocol implemented for control of a system consisting of three nodes for control of DC servo motor are presented. At the end the authors have presented results of simulation for control and planning of execution tasks in WirelessHART network. **[44]** defines a complex road tunnel scenario involving multiple mobile robots navigating in a sensor network environment. Here a TrueTime simulation model of the tunnel scenario is developed. The TrueTime simulator allows concurrent simulation of the physical robots and

their environment, the software in the nodes, the network routing, the radio communication, and the ultra-sound navigation system.

For the installation of industrial field buses in new industrial applications and plants there is need of new approaches to the designing of the control system. The modern networked control systems are usual decentralized systems interconnected by industrial network cables or wirelessly **[45]**. The paper describes networked control in industrial applications and the possibility to simulate the control systems by TrueTime. Wireless Sensor networks are projected by linking existing and emerging technologies of computers, wireless radio communications systems and sophisticated sensors to be used in application. There is a description of some typical simulation methods in Wireless Sensor Networks (WSN) **[46]** where the authors present the TrueTime simulation tool and introduce key steps of WSN frame construction and node models creation and function code written. Finally, they illuminate the simulation method by an example of route research. The results show that TrueTime provides a novel tool for simulating wireless sensor networks.

TrueTime makes it possible to simulate the timely behaviour of real-time kernels executing controller tasks **[47]**. TrueTime also makes it possible to simulate simple models of network protocols and their influence on networked control loops. One example for the simulation of three phase servo motor using TrueTime simulator has shown in **[48]**.

1.3 Scope and Organization of the Dissertation

The main objective of this thesis is to construct control logic for automatic control of freely moveable vehicles, and to implement that logic so that during the movement of the vehicles they do not collide to each other. Here every vehicle will have a definite range around it, which will be obtained by using Bluetooth device. Control logic will be such that the range of the vehicles should not overlap each other to avoid collision. Basic Coordinate geometry concepts are utilized here to denote the location of every vehicle and kinematic models are constructed for the modelling of vehicles. A MATLAB SIMULINK based approach with the TrueTime toolbox is used to obtain a credible simulation.

The rest of the dissertation is organized as follows:

Chapter 2: Provides an overview on TrueTime Toolbox which gives preliminary idea about each of its models. Some examples have been explained with the simulation result for the better understanding of TrueTime Simulator.

Chapter 3: Demonstrates the mathematical modelling for the Kinematic model of the vehicles. In this section the mathematical expression for the controller is also discussed.

Chapter 4: Depicts the simulation and results for the controls of freely moveable vehicles achieve collision avoidance. In this section justification of the work done is also presented and substantiated with simulation results.

Chapter 5: Presents Conclusion to this dissertation and suggests scope for future work.

Chapter 2

TrueTime Simulator

2.1 Reason behind using TrueTime

For the simulation of coordinated control, which is discussed in this thesis there is need of frequent data exchange between the adjacent car models. So data transmitters and receivers are required for the simulation of this model. TrueTime network has the capability of simulating sending and receiving data wired or wirelessly. Apart from sending and receiving, network block simulates medium access and packet transmission in a local area network. When a node tries to transmit a message (using the primitive ttSendMsg), a triggering signal is sent to the network block on the corresponding input channel. When the simulated transmission of the message is finished, the network block sends a new triggering signal on the output channel corresponding to the receiving node. Thus data transfer can be achieved efficiently and wirelessly.

Apart from these TrueTime allows the execution time of tasks and the transmission times of messages to be modelled as constant, random, or data-dependent. Furthermore, TrueTime allows simulation of context switching and task synchronization using events or monitors.

2.2 TrueTime Toolbox

TrueTime toolbox is a MATLAB or Simulink-based simulator for real-time control systems. This toolbox helps to co-simulation of controller task execution in real-time kernels, network transmissions and continuous plant dynamics.

This toolbox provides possibility to write tasks as M-files, C++ functions or call Simulink block diagrams from within the code functions. TrueTime blocks include generally used networks as Ethernet, CAN, TDMA, FDMA, Round Robin or Switched Ethernet). In a brief description we can say that TrueTime is a small library of simulation blocks which extends usability of MATLAB/Simulink to simulate discrete network process control **[49]**.

In every TrueTime toolbox simulation scheme there should be four main parts these are TrueTime kernel (computer, I/O device or some embedded system), True Time network (Network model wired or wireless and an ultrasound network), True Time sender and receiver and a controlled process. There is also an optional part TrueTime battery (all blocks

are shown on Fig. 2.1). TrueTime kernel is responsible for I/O and network data acquisition or data processing and calculations. It can realize a control algorithm/logic and it is the "brain" of every device. It uses several simple M-files (modified by us to satisfy our needs) which handle all mentioned operations. In the kernel can be executed several independent tasks (periodic, nonperiodic) which can cooperate on the same goal.

TrueTime network or TrueTime Wireless network are blocks in which we choose desired network type and set its corresponding parameters. If we include TrueTime battery block we can set power supply for chosen TrueTime kernels. As battery is running out of power we can implement some specific behaviour to avoid collapse of our control system.

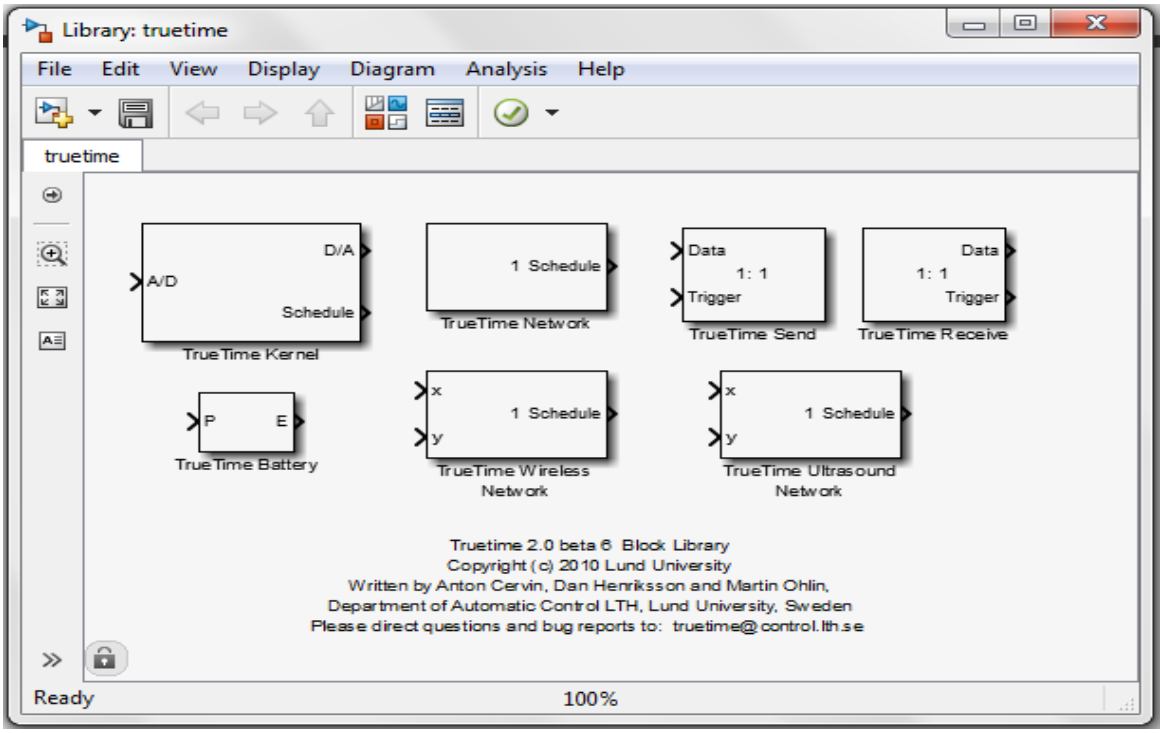

Figure 2.1 TrueTime 2.0 beta 6 Block Library [TrueTime 2.0 Software]

2.3 Description of TrueTime Block Library

As fig 2.1 shows TrueTime Block Library has six types of simulation block, each block has its own characteristics thus capability to perform. The detailed description of each block is given below:

2.3.1 TrueTime Kernel Block

The TRUETIME blocks are connected with ordinary Simulink blocks to form a real time control system. Before a simulation can be run, however, it is necessary to initialize kernel blocks and network blocks, and to create tasks, interrupt handlers, timers, events, monitors, etc for the simulation.

The execution of tasks and interrupt handlers is defined by code functions. A code function is further divided into code segments according to the execution model shown in fig 2.2. All execution of user code is done in the beginning of each code segment. The execution time of each segment should be returned by the code function. If the execution time of the first segment is 2 ms., this means that the delay from input to output for this task will be at least 2 ms. However, pre-emption from higher priority tasks may cause the delay to be longer. The second segment returns a negative execution time. This is used to indicate end of execution, i.e. that there are no more segments to execute.

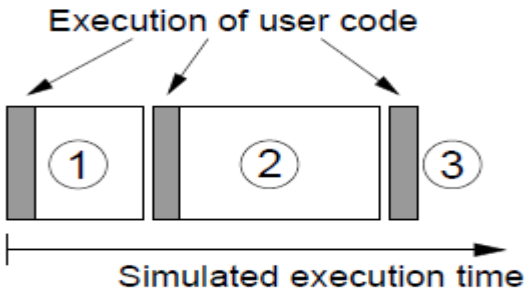

Figure 2.2 Sequence of segments executed in order by the Kernel [49]

The initialization code and the code that is executed during simulation may be written either as MATLAB M-files or as C++ code (for increased simulation speed). The name of the M-file would be same as the name of init function, which is declared on the block parameter of TrueTime Kernel block. Example of TrueTime kernel settings dialog is on the Fig. 2.3. As mentioned before it can provide control logic and data processing. Firstly it is needed to set basic parameters for this block.

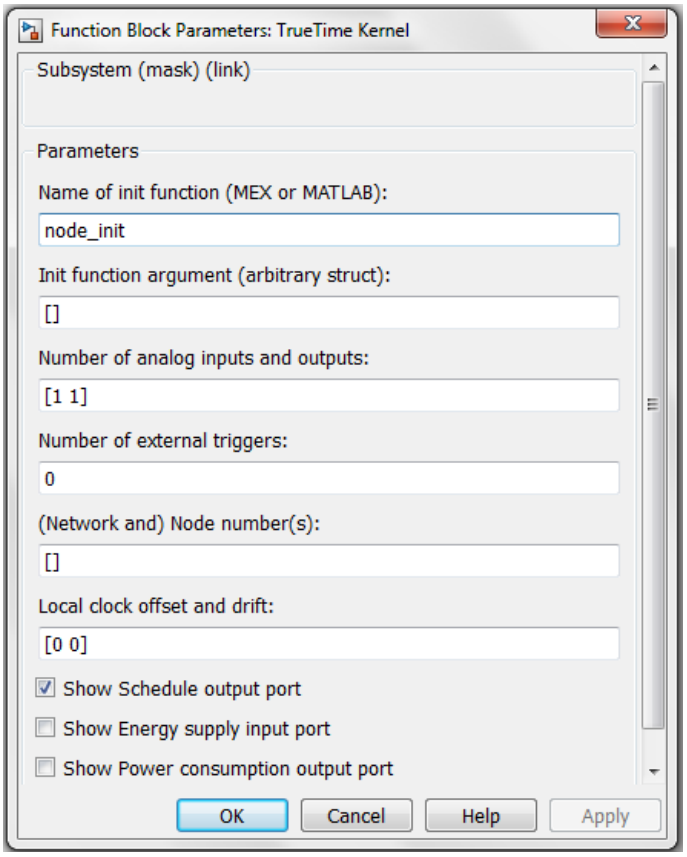

Figure 2.3 function block parameter of TrueTime Kernel block [49]

TrueTime kernel parameters are as follows:

- **Init Function:** - It defines the name of an M-file or a MEX file where the initialization code is stored.
- **Init Function Argument**: - It is an optional argument to the initialization script. This can be any MATLAB structure.
- **Number of Analog input and out:** - We need to mention here the number of input and output ports according to the requirement for any system execution.
- **Number of external triggers:** - number of external triggering required for the execution of any system.

- **Node number:** - It is the number of that block among all the TrueTime blocks.
- **Clock offset and Drift:** - Clock offset is a constant time offset from the nominal time; it sets time offset from first run of simulation. And clock drift can setup value time difference from standard time. When clock drift is set to 0.01, means that device local time will run 1% faster than real-time.
- **Energy supply input port:** - Enable this check box if the kernel should depend on a power source.

2.3.2 TrueTime Network Block

The TrueTime network block simulates medium access and packet transmission in a local area network. When a node tries to transmit a message using the primitive ttSendMsg, a triggering signal is sent to the network block on the corresponding input channel. When the simulated transmission of the message is finished, the network block sends a new triggering signal on the output channel corresponding to the receiving node. The transmitted message is put in a buffer at the receiving computer node. A message contains information about the sending and the receiving computer node, arbitrary user data which is typically measurement signals or control signals, the length of the message, and optional real-time attributes such as a priority or a deadline.

The network block is configured through the block mask dialog, using the command ttSetNetwork Parameter; it is also possible to change some parameters on source block parameter of network block which is shown in fig 2.4. The following network parameters are common to all models:

- **Network number:** - The number of the network block. The networks must be numbered from 1 and upwards. In this simulator Wired and wireless networks are not allowed to use the same number.
- **Number of nodes:** - The number of nodes that are connected to the network. This number will determine the size of the Network Send, Network Receive and Schedule input and outputs of the block.
- **Data rate (bits/s)**: - It defines the speed of the network.
- **Minimum frame size (bits)**: - A message or frame shorter than this will be padded to give the minimum length. It denotes the minimum frame size, including any overhead introduced by the protocol. That is, the minimum Ethernet frame size.

- **Loss probability (0–1)**: - The probability that a network message is lost during transmission. Lost messages will consume network bandwidth, but will never arrive at the destination.

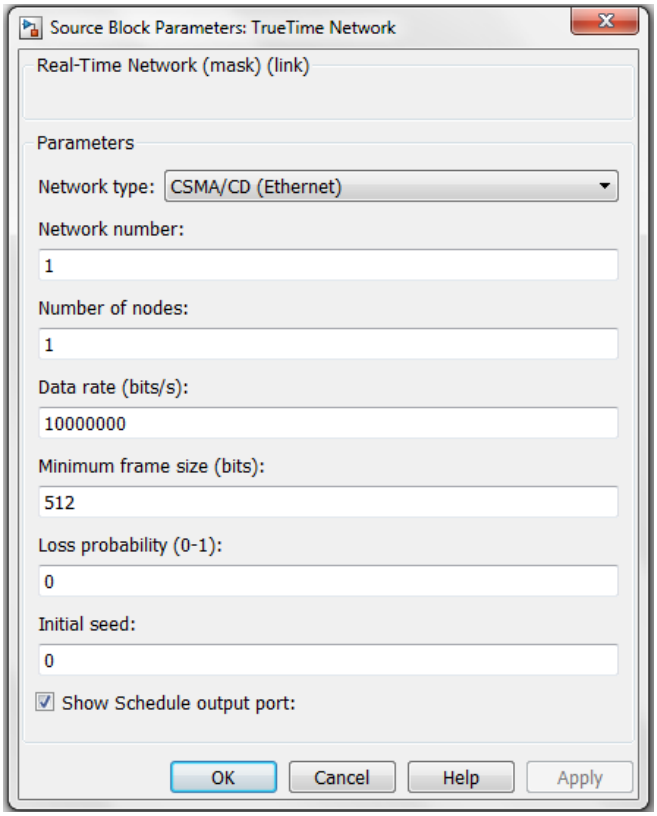

Figure 2.4 source block parameter of TrueTime network block [49]

TrueTime supports nine models of network, such as:

- CSMA/CD or Ethernet
- CSMA/AMP or CAN (Control Area Network)
- Round Robin or Token bus
- FDMA (Frequency Division Multiple Access)
- TDMA (Time Division Multiple Access)
- Switched Ethernet
- FlexRay
- PROFINET
- NCM

2.3.3 TrueTime Send and TrueTime Receive

TrueTime Send block sends a data from one node to another. A triggering pulse is required for the working of TrueTime Send block; this trigger may be type of falling, rising or either. The send block can be time-triggered or event-triggered. And TrueTime Receive block receives a data coming from one node; here triggering is optional. Block parameter for TrueTime Send and Receive has shown in fig 2.5. We can configure some parameter such as:

- **Trigger Type: -** It defines the pattern to trigger the TrueTime Send and Receive block. It is required for Send block. But for Receive block it is optional.
- **Network number: -** The number of the network block. The networks must be numbered from 1 and upwards. The network number can be same for one set of Send and Receive block.
- **Sender id:** - It is the number of sending node and must be numbered from 1 and upwards.
- **Receiver id: -** This is number of receiving node and it must be same for sender and receiver block for delivering a data from a sending node to receiving node. Sometimes Receiver source can be external for a TrueTime Send block.
- **Data length:** - It is the length of sending data in bits, which has to be send to the Receiving end.
- **Priority:** - It defines the urgency of that node or rather the requirement sequence.

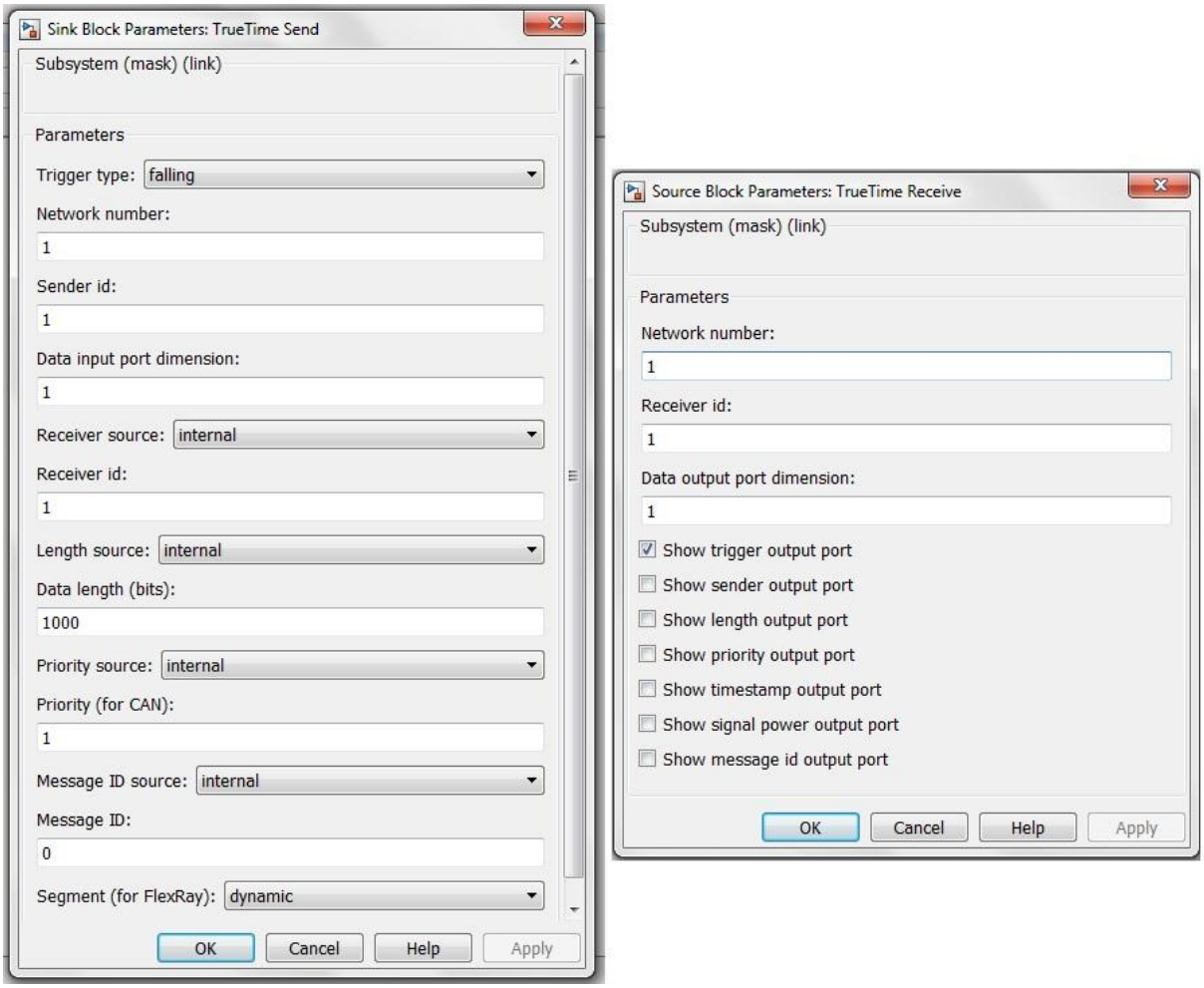

Figure 2.5 Block Parameter of TrueTime Send and TrueTime Receive [49]

2.3.4 TrueTime Wireless Network

The use of the wireless network block is similar to wired network block and it works in the same way. To take the path-loss of the radio signals into account, it has x and y inputs to specify the true location of the nodes. Some more technical details about the wireless network can be found in **[50]**. The used radio model includes here can also support for:

- Ad-hoc wireless networks.
- Isotropic antenna.
- Inability to send and receive messages at the same time.

- Path loss of radio signals modelled as $\frac{1}{d^a}$ where d is the distance in meters and a is a parameter chosen to model the environment.
- Interference from other terminals.

The wireless network block is configured through the block parameter shown in figure 2.6. Some parameters can also be set on a per node basis with the command ttSetNetwork Parameter. The following parameters are common to all models:

- **Network type: -** It determines the MAC protocol to be used. It can be three types such as 802.11b/g (WLAN), 802.15.4 (ZigBee) or NCM_WIRELESS.
- **Network number: -** The number of the network block. The networks must be numbered from 1 and upwards. Wired and wireless networks are not allowed to use the same number.
- **Number of nodes: -** The number of nodes that are connected to the network. This number will determine the size of the Send, Receive and Schedule input and outputs of the block.
- **Data rate (bits/s): -** It defines the speed of the network.
- **Minimum frame size (bits): -** This is the minimum length of message or frame; it denotes the minimum frame size, including any overhead introduced by the protocol. That is most network protocols have a fixed number of header and tail bits, so the frame must be at least size of header + size of tail long.
- **Transmit power: -** It determines how strong the radio signal will be, and thereby how long it can reach.
- **Receiver signal threshold: -** This is the threshold value for received energy; if the received energy is above this threshold, then the medium is accounted as busy.
- **Path-loss exponent: -** The path loss of the radio signal is defined as $\frac{1}{d^a}$ where d is the distance in meters and a is suitably chosen parameter to model the environment. Typically a is chosen in the interval [2 4].
- **ACK timeout: -** It is the time limit; a sending node will wait for an ACK message before concluding that the message was lost and retransmit it.

- **Retry limit: -** It is the maximum number of times a node will try to retransmit a message before giving up.
- **Error coding threshold: -** A number in the interval [0, 1] that defines the percentage of block errors in a message that the coding can handle. For example, certain coding schemes can fully reconstruct a message if it has less than 3% block errors. For the noise is all other ongoing transmissions the numbers of block errors are calculated using the signal-to-noise ratio.

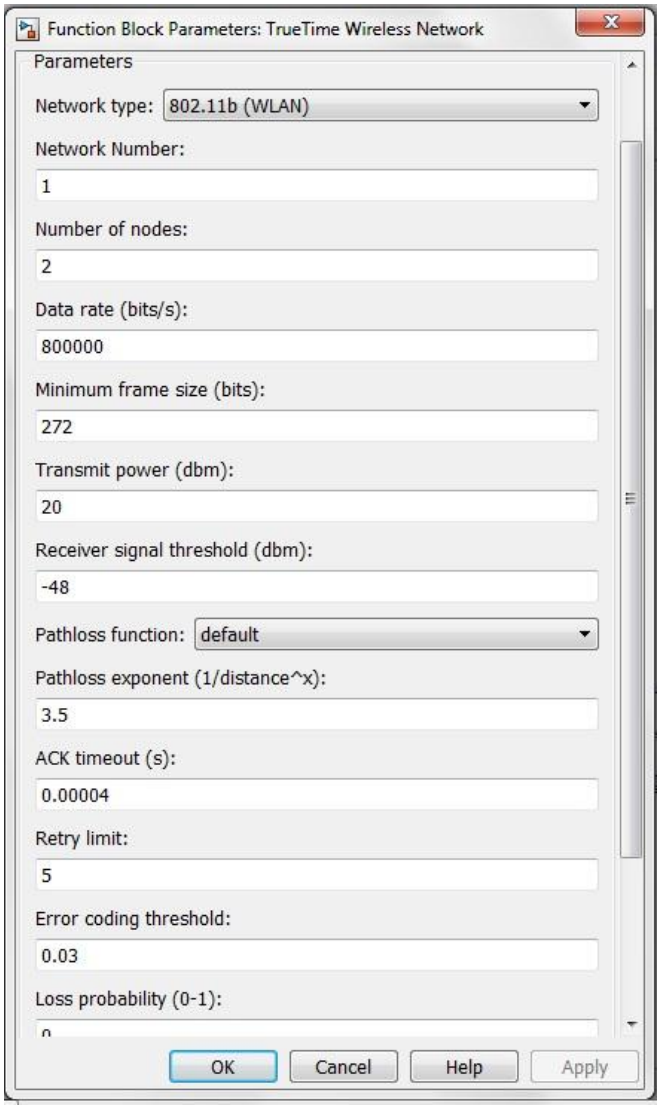

Figure 2.6 Block parameter of TrueTime Wireless Network [49]

2.3.5 TrueTime Battery

The simulation model of battery block has shown in figure 2.7. To use the battery, enable the check box in the kernel configuration mask and connect the output of the battery to the E input of the kernel block. Connect every power drain such as the P output of the kernel block, ordinary Simulink models, and the wireless network block to the P input of the battery. The battery block has one parameter, the initial power (also shown in figure 2.7), which can be set using the configuration mask. The battery uses a simple integrator model, so it can be both charged and recharged. Note that the kernel will not execute any code if it is configured to use batteries and the input P to the kernel block is zero.

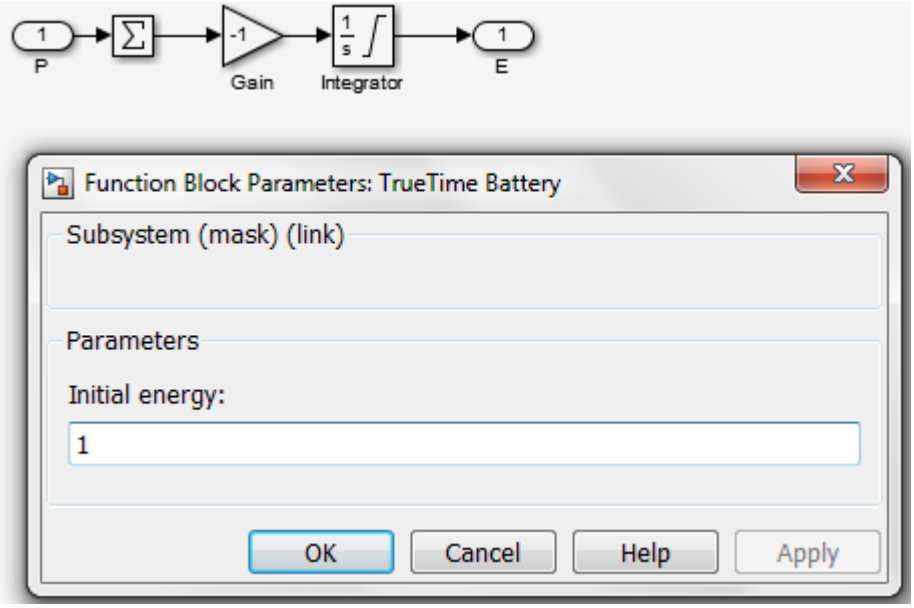

Figure 2.7 Block parameter of TrueTime Battery and Simulink model of TrueTime Battery [49]

2.4 Illustrative Examples

Here a demonstration of few examples based on the simulation models has been done from the TrueTime example directory and from **[49].**

2.4.1 Wired NCS with standalone interface block

This simulation model consists of a DC motor which is controlled by a discrete PID controller interconnected by the Ethernet fieldbus. The model is shown in figure 2.8. The discrete PID controller has the parameters k_p = 1.4, k_i = 1.2, k_d = 0.1, with sample time of 0.01 sec.

The DC motor transfer function is chosen as,-

$$G(s)=\frac{107}{s^3+11s^2+91s+108.3} \tag{2.1}$$

The simulated controlled output and reference inputs are shown in figure 2.9; and the graph for network schedule is shown in figure 2.10.

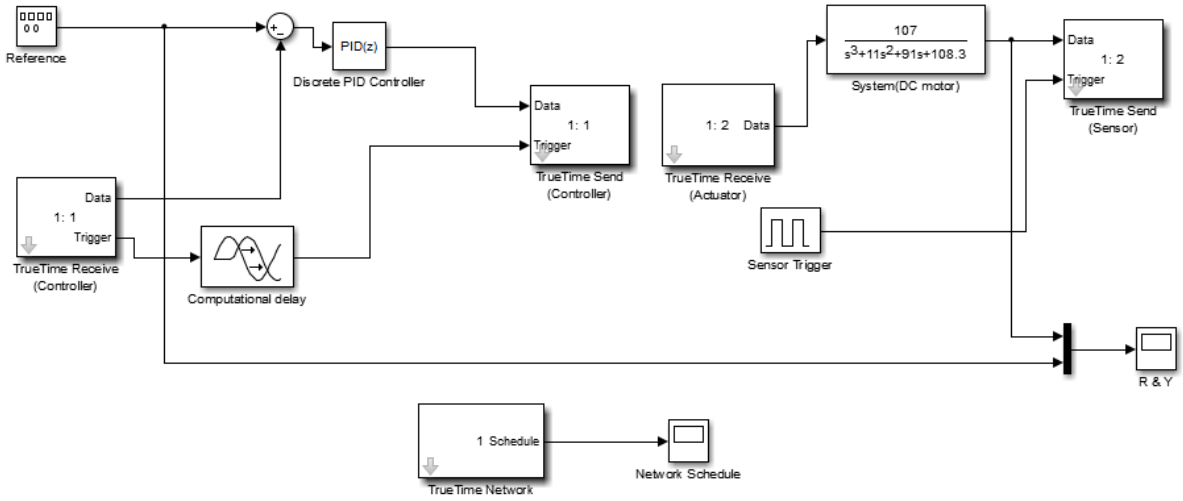

Figure 2.8 Simulation of Wired NCS model

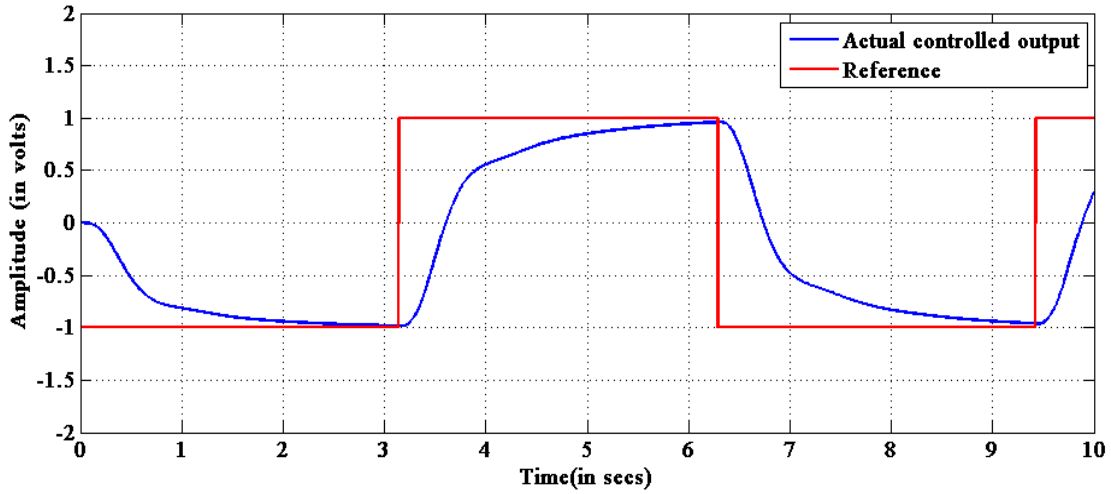

Figure 2.9 Graph for controlled output with Reference of Wired NCS model

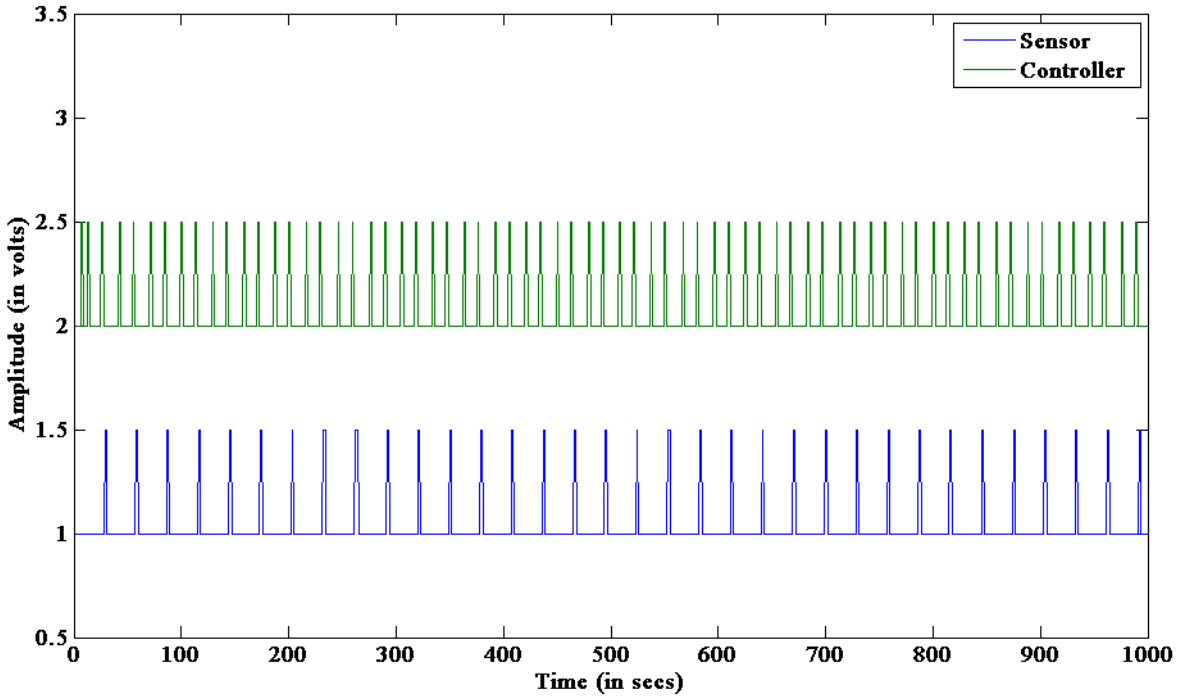

Figure 2.10 Graph for network Schedule of Wired NCS model

Here a third order system is controlled by a discrete PID controller and as evident from figure 2.9, the controlled output shows an over damped response. The settling time of the response is quite large. Such type of response is not quite preferable for a physical system, but in this section our main aim is to show the working of system which is incorporated with TrueTime simulator. Figure 2.10 shows the network response for sensor and controller. Some deviation of the signals can be seen, if the signal is *Hi* it means that the task is running. An intermediate value of the signal indicates that the task is ready but not running, whereas a *Lo* signal means that the task is idle.

2.4.2 Wireless NCS with standalone interface block

The wireless network block has two extra input ports. These ports are the x and y coordinates of the nodes to specify their location and compute the path-loss of the radio signal. Like wired control of a DC motor has done here with the help of a discrete PID controller interconnected by 802.11b (WLAN). The Simulink model has shown in figure 2.11. And graph for controlled output and Reference has shown in figure 2.12; here graph for network schedule has also shown in 2.13.

The simulation represented by Fig. 2.8 is repeated with a wired network replaced by a wireless block as presented in Fig. 2.11 and the corresponding controlled output and reference input, as well as the network schedule are represented in Fig. 2.12 and 2.13 respectively.

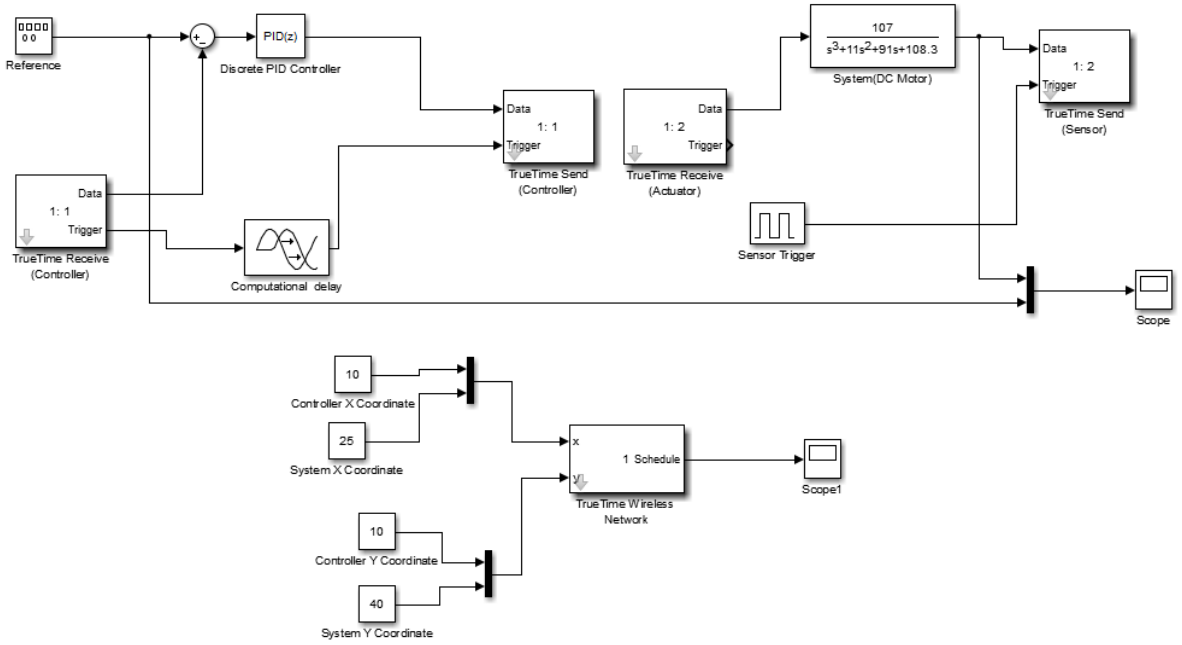

Figure 2.11 Simulation of Wireless NCS model

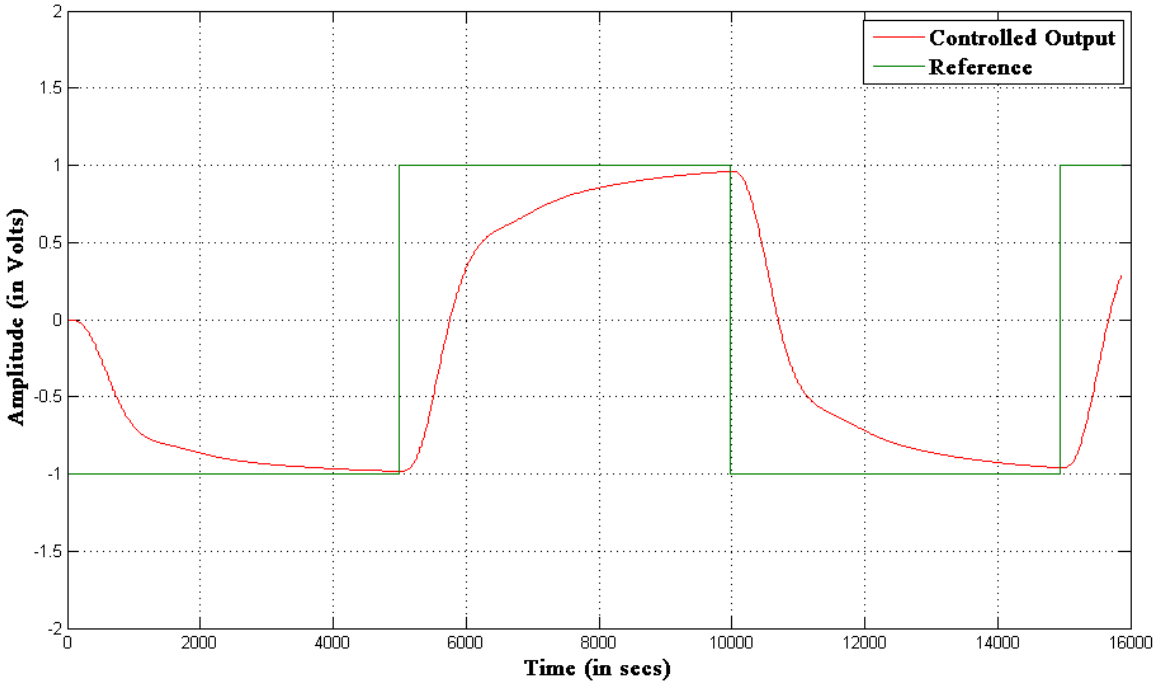

Figure 2.12 Controlled Output and Reference Graph for Wireless NCS model

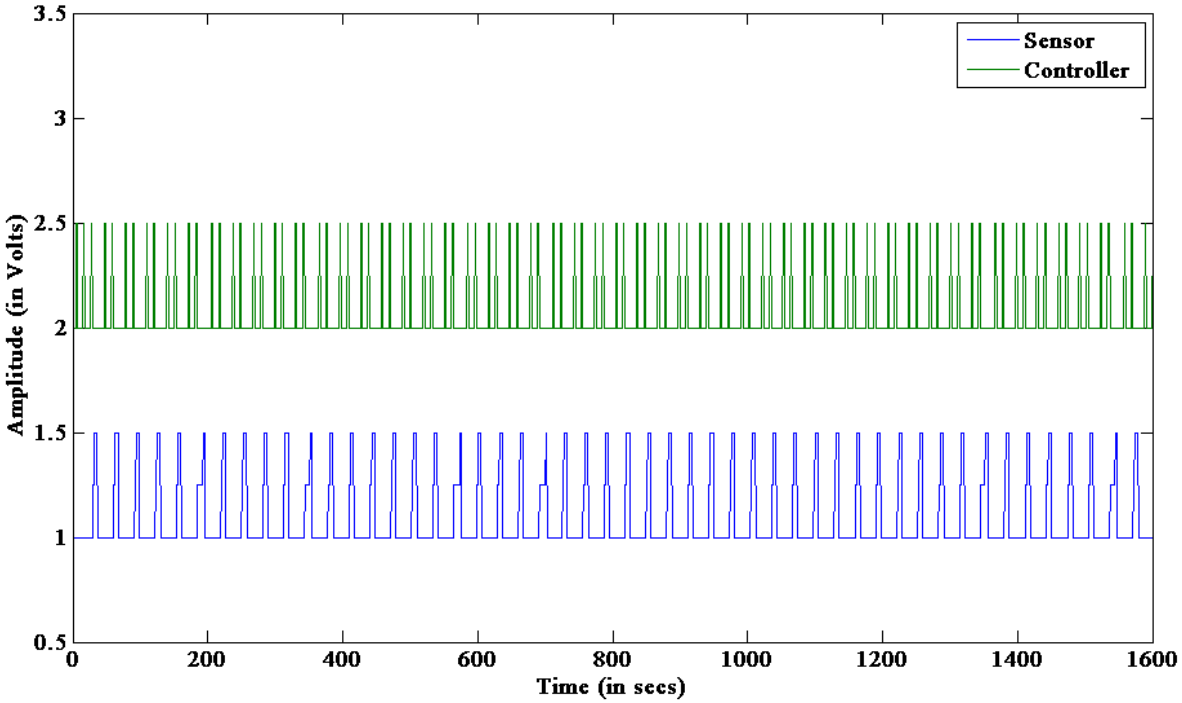

Figure 2.13 Network Schedule for Wireless NCS model

Here the same system is controlled by a discrete PID controller and due to selected parameter it can be realised from figure 2.12 that like previous here the controlled output is an over damped response. The settling time of the response is quite large. Figure 2.13 shows the network response for sensor and controller. Here deviation of the signals can be seen which are quite different. Due to wireless transmission some delay in execution of tasks can be seen.

2.4.3 Wired NCS Simulation with Standalone and Kernel Block

In this simulation a kernel block is used as a controller and standalone interface blocks as the sensor and actuator. Here also controlled system is a DC motor controlled by PID controller interconnected by Ethernet; shown in figure 2.14. Due to use of kernel block as a controller there is an *initialization code* and a *task code* for the kernel block (Controller); presented in APPENDIX – I. The system and the PID controller remain same as in the previous simulation and the controlled output corresponding to a reference input variation is presented in figure 2.15, and the corresponding Network Schedule in figure 2.16.

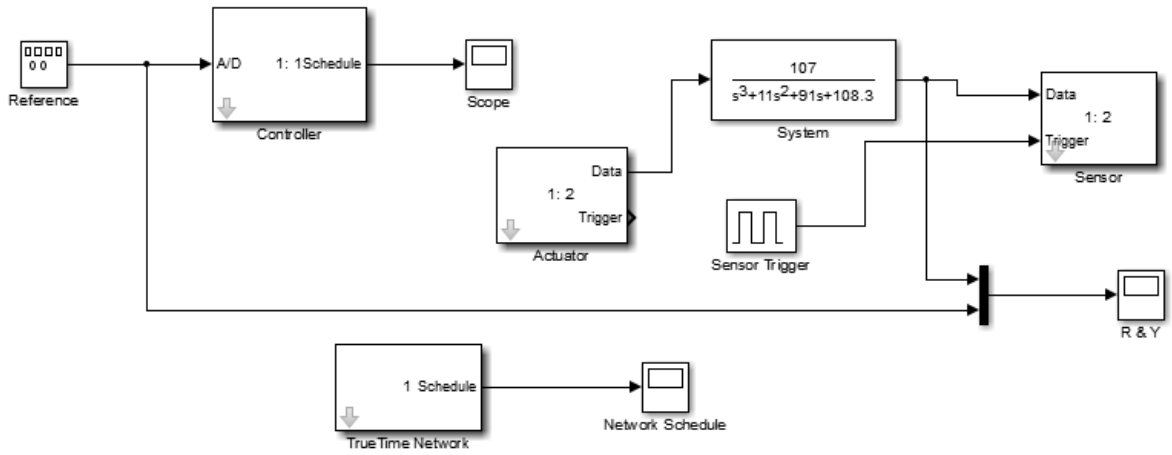

Figure 2.14 wired NCS model with a Kernel Block

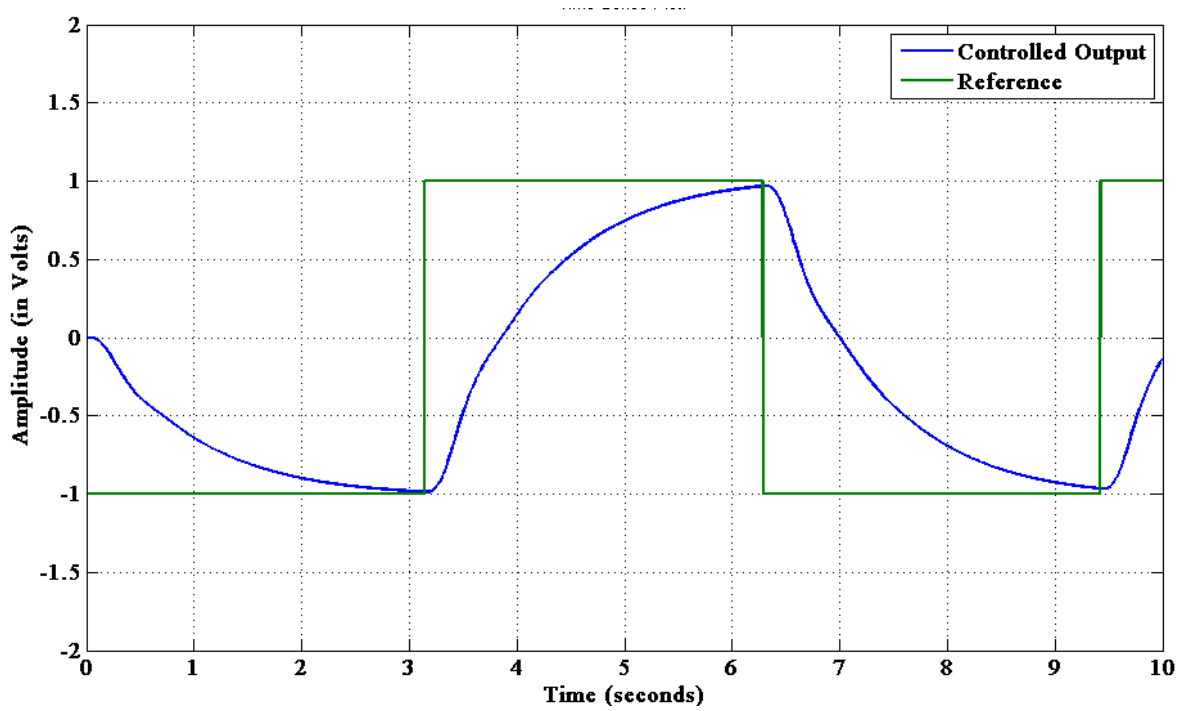

Figure 2.15 Reference and Controlled Output for Wired NCS model with Kernel

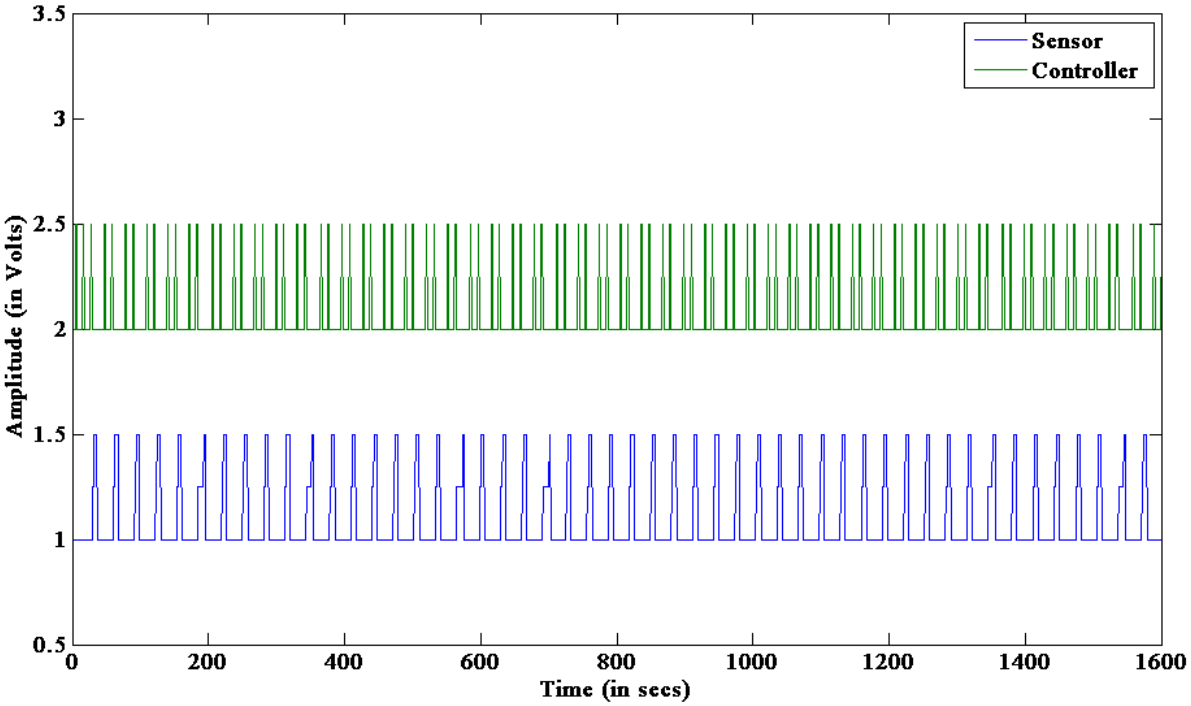

Figure 2.16 Network Schedule for wired NCS model with Kernel

It is seen that the time response of the simulated system in this case resembles those in the previous cases.

2.4.4 Wireless control of a first order system using Kernel block for Sensor, Actuator and Controller

In this simulation one First Order system is used and it is controlled by PID controller. The properties of controller, actuator and sensor are incorporated with TrueTime Kernel Blocks. The Simulink block diagram is shown in figure 2.17. The initialization and task codes for the Actuator, Controller and Sensor are given in APPENDIX-I. The transfer function for the system chosen is

$$G(s) = \frac{1}{s+1} \tag{2.2}$$

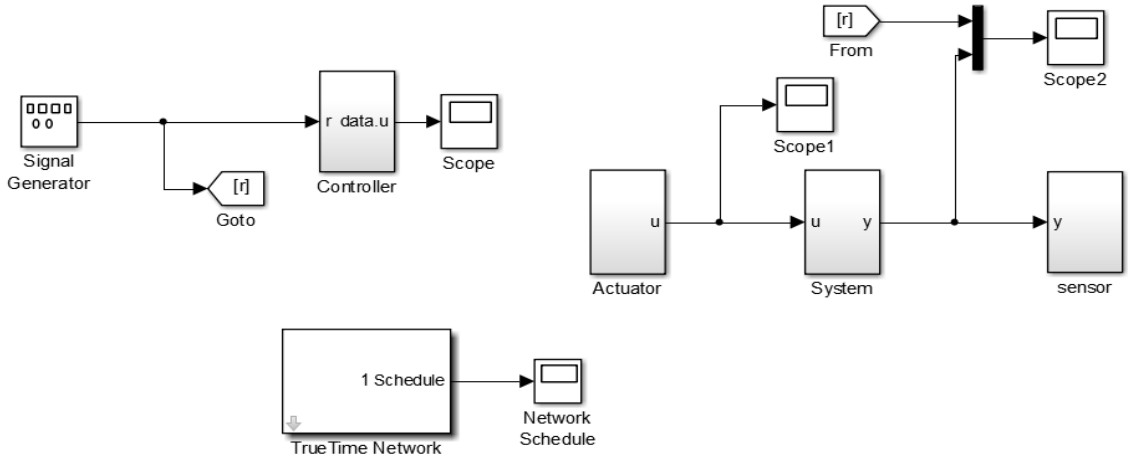

Figure 2.17 Simulation for wireless control of first order system

Taking PID controller parameters as $k_p = 100$, $k_i = 1$, $k_d = 0.049$; the controlled output with reference has shown in figure 2.18, Actuator output is presented in figure 2.19, and Network Schedule graph in 2.20.

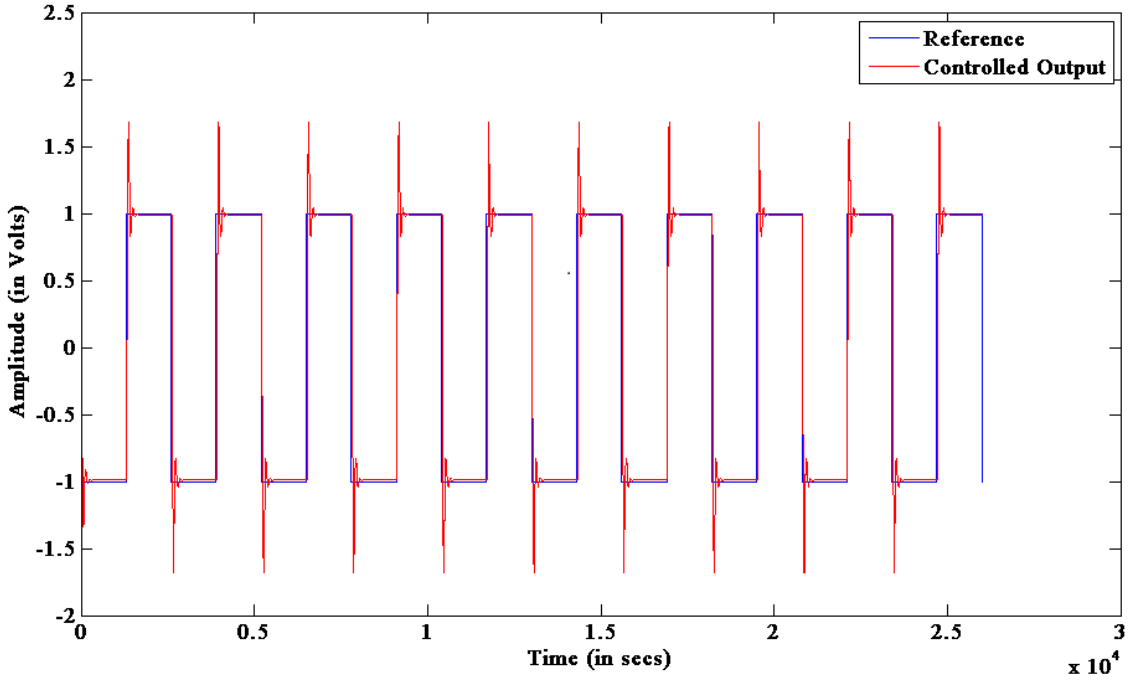

Figure 2.18 Reference and Controlled Output of first order system

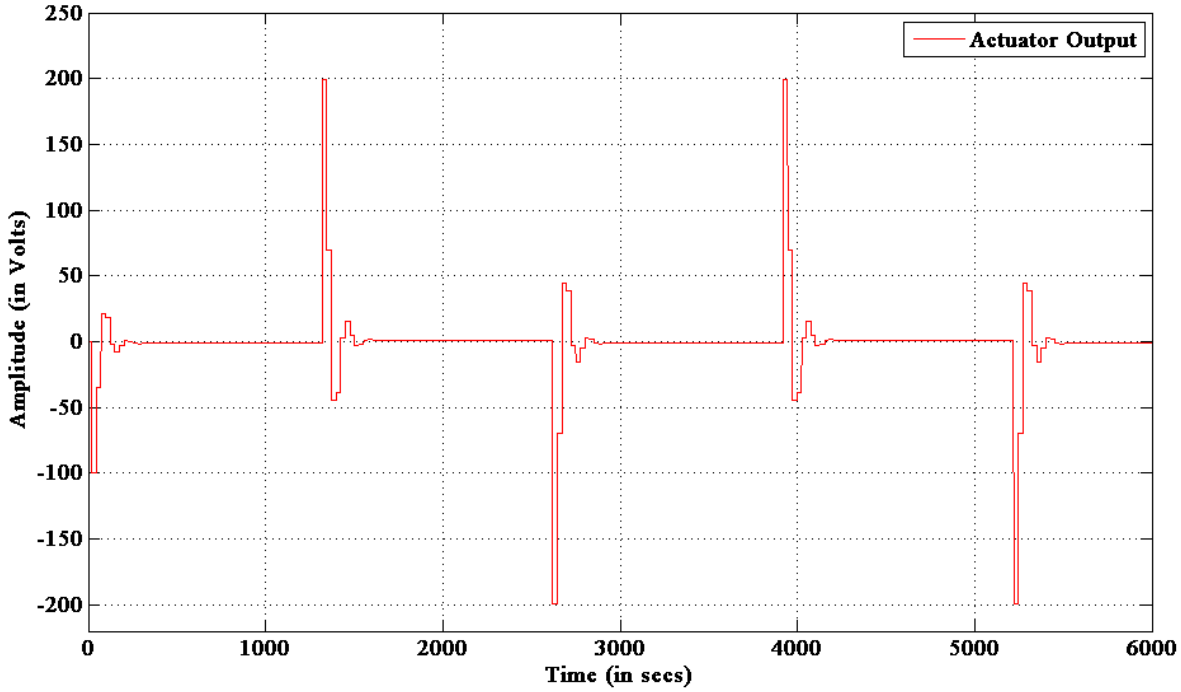

Figure 2.19 Output graph of Actuator

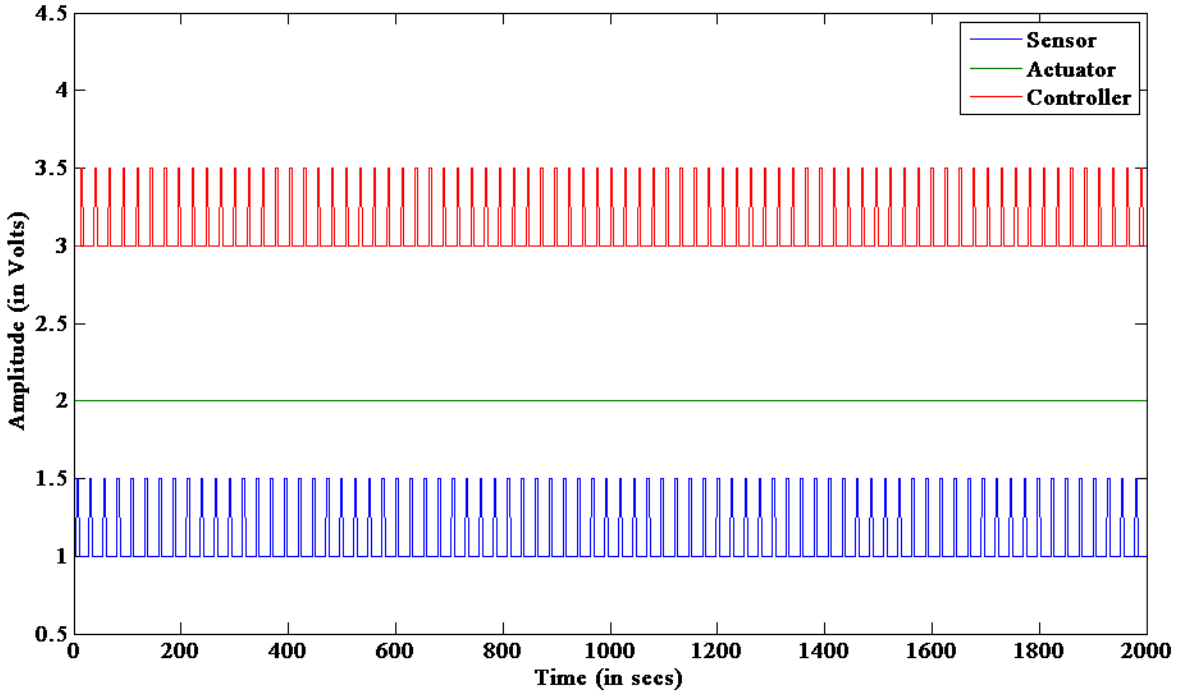

Figure 2.20 Network Schedule graph

Figure 2.20 shows the network schedule output for sensor and controller. Here some delay in response can be seen due to the wireless communication.

Chapter 3

Coordinated Control

3.1 Problem Formulation

Automatic control for vehicular system can be done in various ways (such as using obstacle sensors, lighting sensors etc.) but coordinate control is the one which is most popular. Various ways for coordinate control has been discussed in **Chapter-1**. In coordinate control, Formation control or path following control is done; but there are some constraints such as:-

1. Vehicles have to follow a predefined path constructed by the virtual robot in the virtual-structure approach.
2. Vehicles are bound to do the predefined tasks in behaviour approach.
3. Vehicles have to follow the leader and the leader performs predefined tasks in Leader-follower approach.
4. For graph theory based solution vehicle trajectory is predefined.

From the above discussion it is clear that approaches used here makes vehicles to follow a predefined trajectory or do predefined tasks. So the problem arises here is then what will happen to the freely moveable vehicles, that is the control of vehicles which does not follow any trajectory or any predefined tasks?

So now a real problem is automatic control of vehicles which does not follow any predefined path or behaviour using a co-ordinated approach.

3.2 Proposed Solution

The solution proposed in this dissertation is inspired by a natural phenomenon viz.- "*motion of similarly charged particles in a confined space*" for the co-ordinated control of freely moveable vehicles to achieve collision avoidance. According to this theory if we randomly place some similarly charged particles in a confined space, they move freely without causing any collision with each other. This happens because each particle is similarly charged, so when they come close to each other they start repelling, and the distance between them from which they starts repelling depends upon their charge. The movement of two positively charged particles A and B is shown in figure 3.1. At first they are free to move whichever direction they want to but when they come close to each other they starts repelling. And the range from where they will start repelling depends upon their charge.

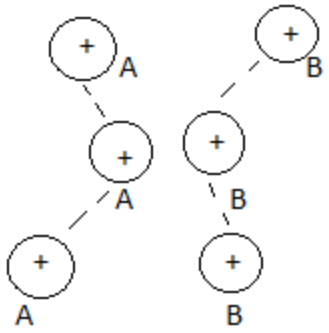

Figure 3.1 movement of positively charged particles

3.3 Methodology

As proposed we can assume here that each vehicle is a positively charged particle. It is further assumed that each car has a Bluetooth device. This Bluetooth device will have some range around the vehicle; the range will depends upon the power of Bluetooth device. So by varying the power of the device we can vary the range. Now they can sense each other when they come close to each other through Bluetooth transceiver system. One condition is shown figure 3.2 where two vehicles are moving freely having some Bluetooth transceiver range around them.

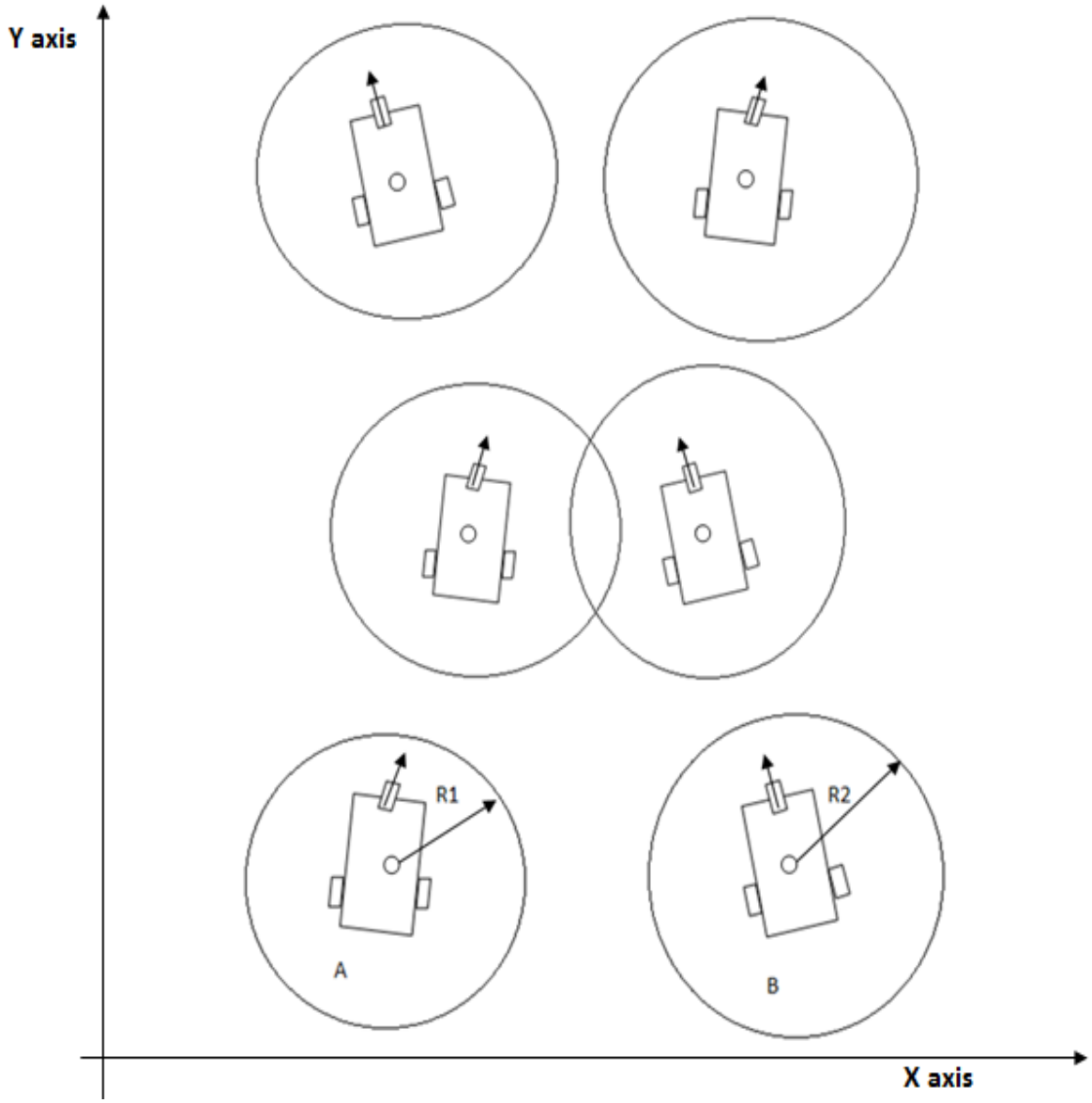

Figure 3.2 Movement of vehicles A and B

Here vehicles A and B are moving freely in a confined place; we can assume that they are moving along X-Y plane having a coordinate. Let coordinate for A is (x_1, y_1) and B is (x_2, y_2). Both vehicles have Bluetooth device we assume that they are placed in the centre of mass of the vehicle; A has a range of R_1 around it, similarly B has a range of R_2. Control logic implemented in this dissertation is such that, when R_1 and R_2 overlap each other the orientation of the vehicles will be reversed. That is if a vehicle is oriented at an angle θ along X axis during its free move, it will become $-\theta$ when an overlap is detected.

3.4 Mathematical Modelling

Mathematical modelling of vehicles has to be done for automatic control. There is two ways of modelling a vehicle:-

1. Dynamic model
2. Kinematic model

In dynamic modelling we have to consider vehicle body and all the manipulators connected to the vehicle. For dynamic modelling we have to calculate total torque and force acting due to vehicle and its manipulators. Torque and force can be calculated by considering Lagrangian mechanics **[51]**. Lagrangian equation can be derived as

$$L = K - P \tag{3.1}$$

where, K is total kinetic energy for vehicle and manipulators
and P is total potential energy for vehicle and manipulators

From the above equation force and torque can be calculated as following (for i^{th} manipulator).

Force of i^{th} manipulator is;

$$F_i = \frac{\partial}{\partial t}\left(\frac{\partial L}{\partial \dot{x}_i}\right) - \frac{\partial L}{\partial x_i} \tag{3.2}$$

Torque of i^{th} manipulator is;

$$T_i = \frac{\partial}{\partial t}\left(\frac{\partial L}{\partial \dot{\theta}_i}\right) - \frac{\partial L}{\partial \theta_i} \tag{3.3}$$

where, x and θ are system variables.

Then modelling of the vehicle is obtained by representing the torque and force equations in state space model.

Kinematics is the mathematics of motion which does not consider the forces that affect the motion. The science of kinematics is the studies of position, velocity, acceleration,

and all higher order derivatives of the position variables with respect to time or any other variables. Hence, the study of the kinematics of manipulators refers to all the geometrical and time-based properties of the motion. So the kinematics of vehicles deals with:-

- The geometric relationships of that vehicle system.
- The relationship between control parameters and the behaviour of system in state space.

In kinematic model we consider the vehicle as a rigid body and the effect of the manipulators is not considered here. For geometric calculations we take the coordinate point at the centre of mass of the vehicle. Figure 3.3 shows a vehicle with an orientation of θ placed in global frame; moving frame of the vehicle is also shown.

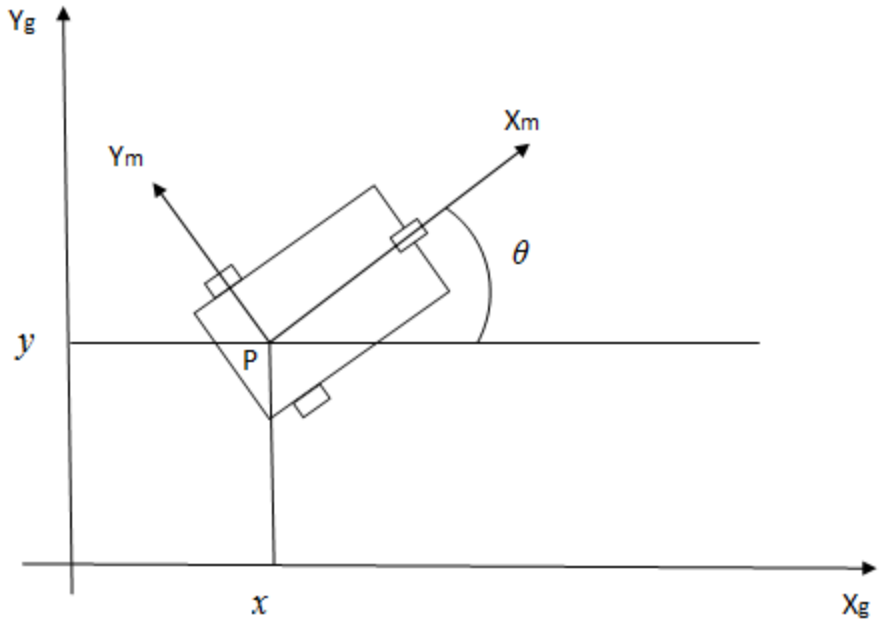

Figure 3.3 Vehicle placed on Global frame with orientation θ

As the figure shows the vehicle is placed on global frame (Xg, Yg); vehicles centre of mass lies on point P and its orientation is θ with respect to global frame. So moving frame (Xm, Ym) will be lagging at an angle θ to the global frame. Here coordinate of P is (x, y); now let us assume that the vehicle is moving with a linear velocity v and an orientation of θ. So the position and orientation vector for the vehicle and the rotation matrix expressing the orientation of the global frame with respect to the moving frame will be (here rotation is about Z axis)

$$\xi = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \tag{3.4}$$

$$R(\theta) = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.5}$$

And due to the orientation the linear velocity along X and Y axis will be respectively

$$v_x = v\cos\theta \tag{3.6}$$

$$v_y = v\sin\theta \tag{3.7}$$

The kinematic equation for the vehicle will be

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{pmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.8}$$

From equation (6) we can get

$$\begin{aligned} \dot{x} &= \frac{dx}{dt} = v * \cos\theta \\ \dot{y} &= \frac{dy}{dt} = v * \sin\theta \\ \dot{\theta} &= \frac{d\theta}{dt} = w \end{aligned} \tag{3.9}$$

Where v is linear velocity; θ is angular displacement and w is angular velocity.

Figure 3.3 shows the vehicle is in coordinate position $\mathrm{P}(x, y)$; if the vehicle moves with a linear velocity v and an orientation θ, its position will change **[52]**. The way of changing and thus next expected position is shown in figure 3.4.

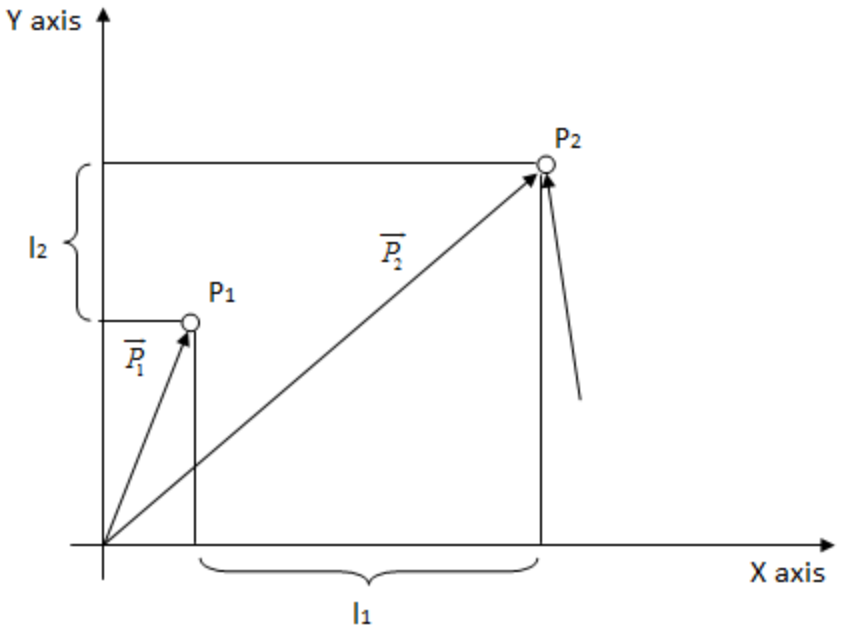

Figure 3.4 vector P_1 rotated and translated to form P_2

In figure 3.4 a case is shown where a vehicle is at point P_1 (x, y) which is denoted by vector $\vec{P_1}$; due to θ rotation about Z axis vector $\vec{P_1}$ rotates θ anti-clock wise and due to its linear velocity v it moves l_1 along X axis and l_2 along Y axis; so its new coordinate is P_2 (x_1, y_1), which is denoted by vector $\vec{P_2}$.

It is clear from above discussion that when a vehicle a state vector of $[x, y, \theta]^T$ is moving on plane with a velocity and orientation; it will have two transformation such that

- Rotational
- Translational

As the vehicle is moving in ground so its rotation will be about Z axis; here the rotational matrix will be (for rotating angle θ)

$$R(\mathrm{Z}, \theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.10}$$

And Translational matrix will be

$$T(l_1, l_2) = \begin{pmatrix} 1 & 0 & l_1 \\ 0 & 1 & l_2 \\ 0 & 0 & 1 \end{pmatrix} \quad (3.11)$$

So if the vehicle is at $P_1(x, y)$; when its rotation about Z axis is θ and translation along X axis is l_1, along Y axis is l_2; then new position $P_2(x_1, y_1)$ can be obtained by

$$P_2(x_1, y_1) = R(Z, \theta) \times T(l_1, l_2) \times P_1(x, y) \quad (3.12)$$

This can be represented in state space model **[52].**

$$\begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & l_1 \\ \sin\theta & \cos\theta & 0 & l_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 0 \\ 1 \end{pmatrix} \quad (3.13)$$

From equation (11) we can get the following equations

$$x_1 = x\cos\theta - y\sin\theta + l_1 \quad (3.14)$$

$$y_1 = x\sin\theta + y\cos\theta + l_2 \quad (3.15)$$

These equations have been used for developing the Simulink models.

3.5 Controller Approach

The co-ordinated controller development is based on the following assumptions **[53]-**

1. Every vehicle has a circular safety area which is obtained by Bluetooth transceiver system. The centre of this communication area is the centre of mass of the vehicle; in this case the radius of communication area for vehicle A is R_1, and R_2 for vehicle B.

2. Every vehicle always transmits its state ($[x, y, \theta]^T$) in its communication area; similarly it can receive the state of others state, if they are in communication area.
3. Initially each vehicle starts from a location which is outside of others communication area.

In this work we have considered two vehicles freely moving in a confined space, so both of them should have a certain coordinate; and as Bluetooth device is placed in the vehicles, there would be a range around that coordinate. The distance between the two vehicles can be calculated by geometric calculation in terms of the co-ordinates of each vehicle. Figure 3.5 shows a condition having two vehicles in a confined space.

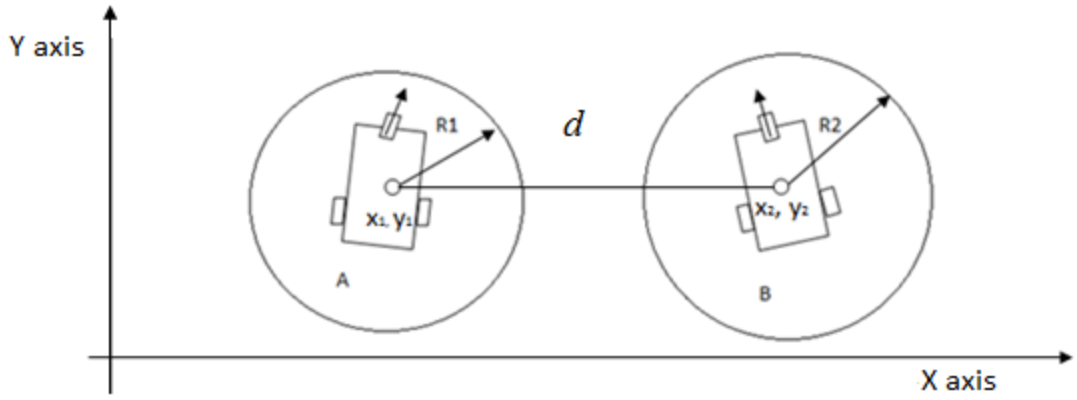

Figure 3.5 Geometry of two vehicles

As figure 3.5 shows in our work there are two vehicles, A and B. Vehicle A having coordinate (x_1, y_1) and Bluetooth range of R_1 , and B with coordinate (x_2, y_2) and Bluetooth range of R_2 are moving in X-Y plane; so the distance between A and B will be-

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{3.16}$$

To avoid collision d should be always greater than sum of range of two vehicles. i.e.

$$d - (R_1 + R_2) \geq \partial \tag{3.17}$$

where, ∂ has very small positive value.

Now controller is placed on vehicle A (local controller) with the logic based on equation (3.15) to control vehicle A. Vehicle A has a state vector $[x, y, \theta]^T$. When controller of vehicle A receives zero or negative value of ∂; controller will turn orientation θ into $-\theta$; thus new state vector will be $[x, y, -\theta]^T$. That means if the orientation was anti-clock wise (θ) along X axis, now orientation will be clock wise ($-\theta$). Thus vehicle will avoid collision automatically with the help of controller. The new position for the vehicles can be simulated using equation set (3.10)-(3.13).

In the next chapter, a TrueTime bases simulation is presented for a two vehicle case.

Chapter 4

Result and Discussion

4.1 Simulink model and result for vehicles movement without controller

In this dissertation we have designed a Simulink model to represent automatic control of vehicles; Simulink blocks are used to design the vehicle model and controller. For data transmission and receiving, TrueTime blocks are used. The simulation of the vehicular models without controller is been shown in figure 4.1.

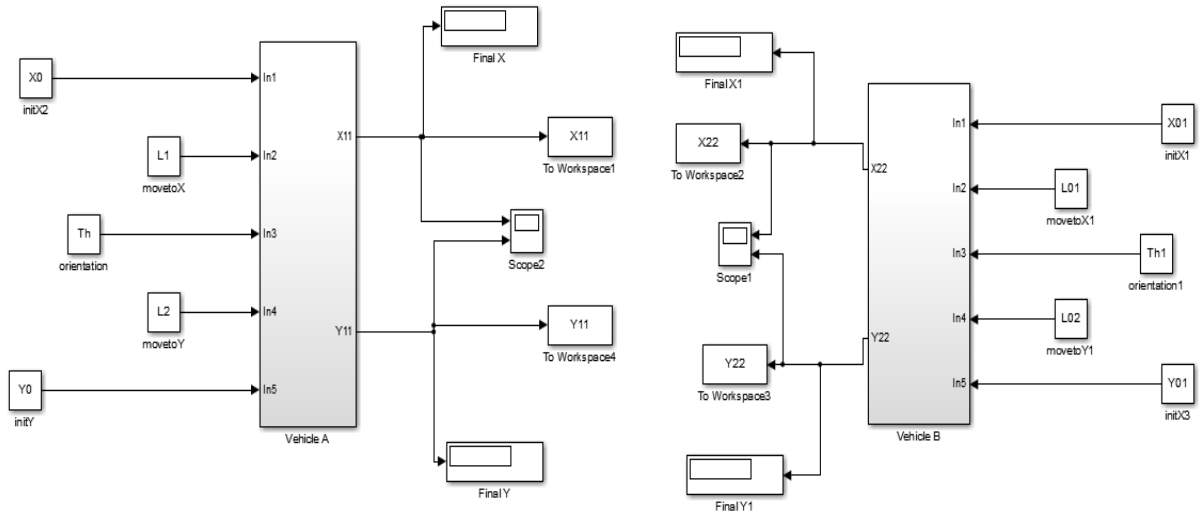

Figure 4.1 Simulation of Vehicles A and B without controller

With the following parameter of vehicle A and vehicle B we can get the trajectory of vehicles as shown in figure 4.2. MATLAB coding for this simulation is shown in APPENDIX-II. The simulation parameters are represented in Table 4.1.

Table 4.1 Initial parameters for vehicles A and B without controller

Parameters for Vehicle A	
Starting coordinate	x_0 =-130 y_0 =120
Fixed orientation	θ =-1.9° (degree)
Translational length along X and Y axis	l_1 =2.5 meter l_2 = 3 meter
Parameters for vehicle B	
Starting coordinate	x_0 =-130 y_0 =20
Fixed orientation	θ =-1.1° (degree)
Translational length along X and Y axis	l_1 =2.5 meter l_2 =3meter

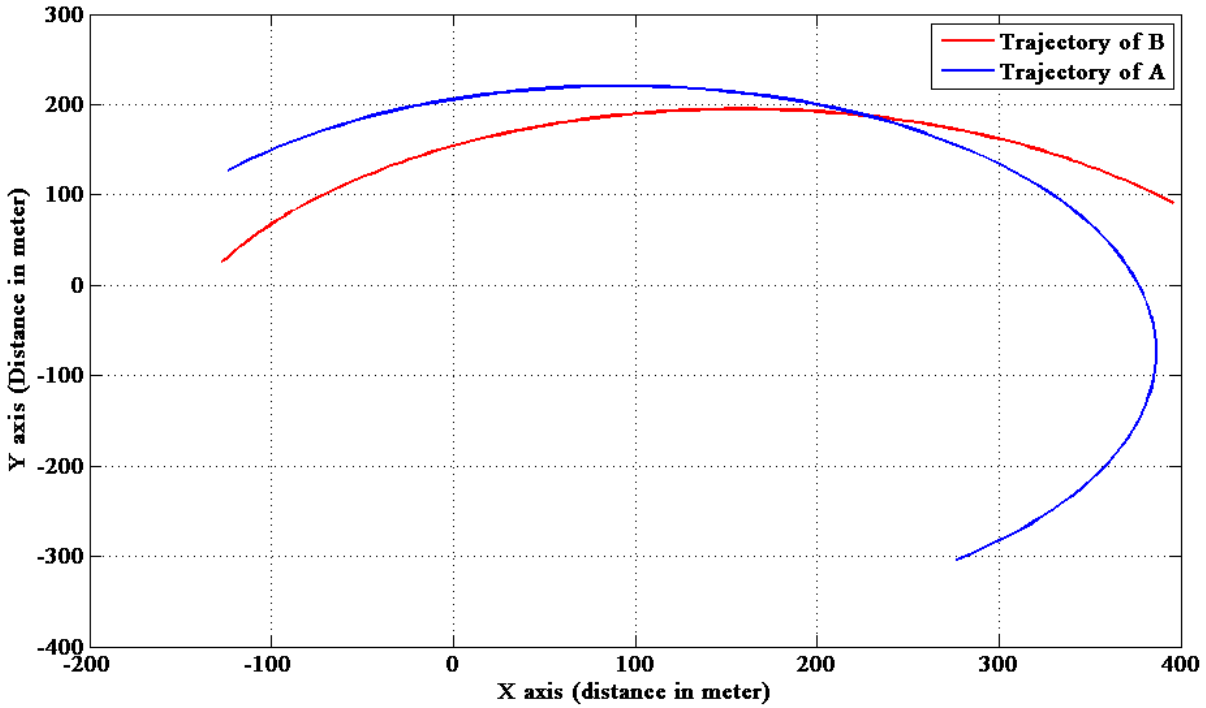

Figure 4.2 Trajectory of Vehicles A and B

Figure 4.3 shows clearly that after travelling of 38 seconds two vehicles are going to collide (in this case step size of the simulation is 1 second). To make this condition we have chosen following parameters.

A second simulation case is shown. The parameters are represented in Table 4.2

Table 4.2 Initial parameters for vehicles A and B with a condition

Parameters for Vehicle A	
Starting coordinate	x_0 =-130 y_0 =55
Fixed orientation	θ =0º (degree), for first 30 second θ =-1.9º (degree), rest of the time
Translational length along X and Y axis	l_1 =2.5 meter l_2 = 0 meter
Parameters for vehicle B	
Starting coordinate	x_0 =-130 y_0 =30
Fixed orientation	θ =0º (degree), for first 30 second θ =-2.45º (degree), rest of the time
Translational length along X and Y axis	l_1 =2 meter l_2 =0meter

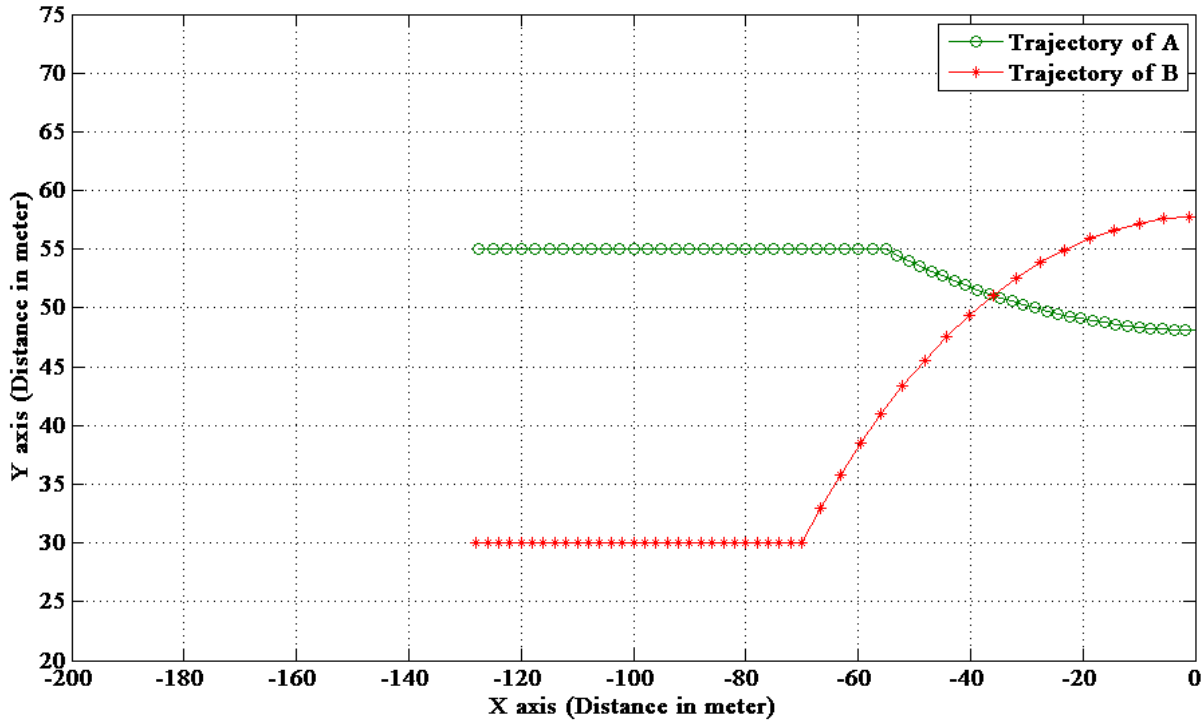

Figure 4.3 Trajectory of vehicles A and B with condition

Figure 4.2 shows the trajectory of two vehicles having a definite linear velocity and orientation. As the linear velocity and orientation is fixed throughout the journey the trajectories of the vehicles are like a continuous curve. The linear velocity and orientation can be different at any point of the journey for these freely moveable vehicles, and then the trajectory should be according that condition. So the occurrence of the collision between the vehicles is optional, it depends on their trajectory and their presence at a time, if the situation is such that two vehicles are in same coordinate at the same time so definitely there should be a collision. From figure 4.2 it is clear that the trajectory of the two vehicles meets at a certain point, but it cannot be unclouded from this figure that the meeting point coordinates are achieved by both of the vehicles at the same instant. So we could not be sure from figure 4.2 that there is a definite collision or not.

In the second condition the parameter are chosen such that by analysing the output result, shown in figure 4.3, there is a certain collision between two vehicles. Figure 4.3 shows that if two vehicles start their journey at the same time, after 38 second there is an indisputable collision between two vehicles (here the step size of the simulation is 1 second), because at 39^{th} second of their journey the coordinate is same for the two vehicles.

4.2 Simulink model and result for vehicles movement with Controller

A local controller is included for avoiding the collision between the vehicles which could occur during their free move. The simulation model describing vehicles with the controller is shown in figure 4.4. MATLAB coding for this simulation is shown in APPENDIX-II.

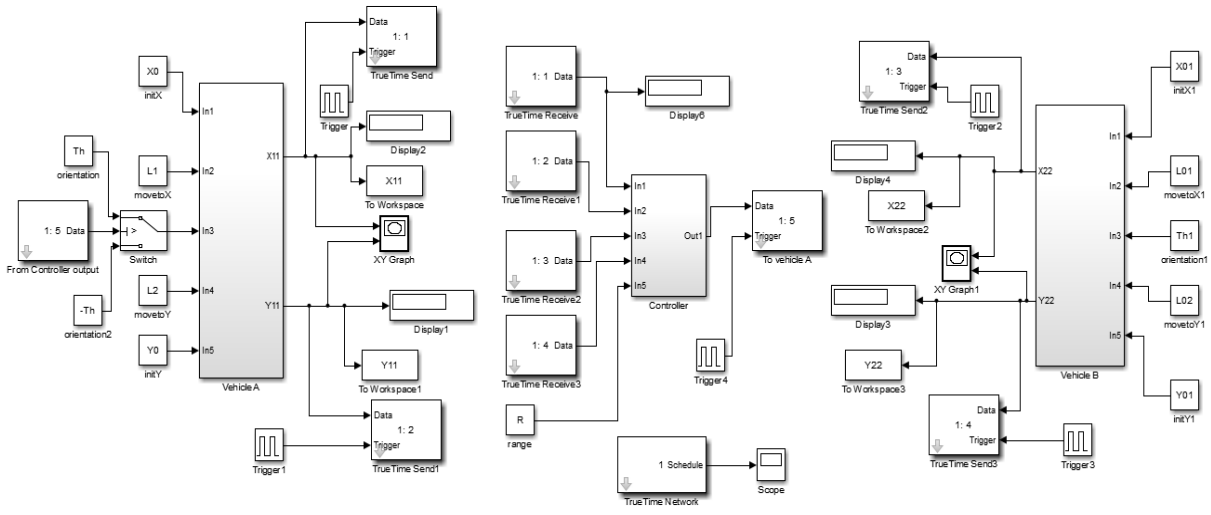

Figure 4.4 Simulink model of vehicles A and B with controller

Figure 4.4 shows the Simulink model where controller is used only for vehicle A, with the following parameter shown in Table 4.3, the Simulink result is shown in figure 4.5. The simulation parameters are chosen as described in Table 4.2, where a collision was shown to occur in absence of a co-ordinated controller.

Table 4.3 Initial parameters for vehicles A and B with controller in one vehicle

Parameters for Vehicle A	
Starting coordinate	x_0 =-130 y_0 =55
Fixed orientation without controller action	θ =0º (degree), for first 30 second θ =2.5º (degree), rest of the time
Fixed orientation with controller action	θ =-2.5º (degree), rest of the time
Translational length along X and Y axis	l_1 =2 meter l_2 = 0 meter
Transmission radius	R_1 =5 meter
Parameters for vehicle B	
Starting coordinate	x_0 =-130 y_0 =35
Fixed orientation without controller action	θ =0º (degree), for first 30 second θ =-2.45º (degree), rest of the time
Translational length along X and Y axis	l_1 =2 meter l_2 =0meter
Transmission radius	R_2 =5 meter

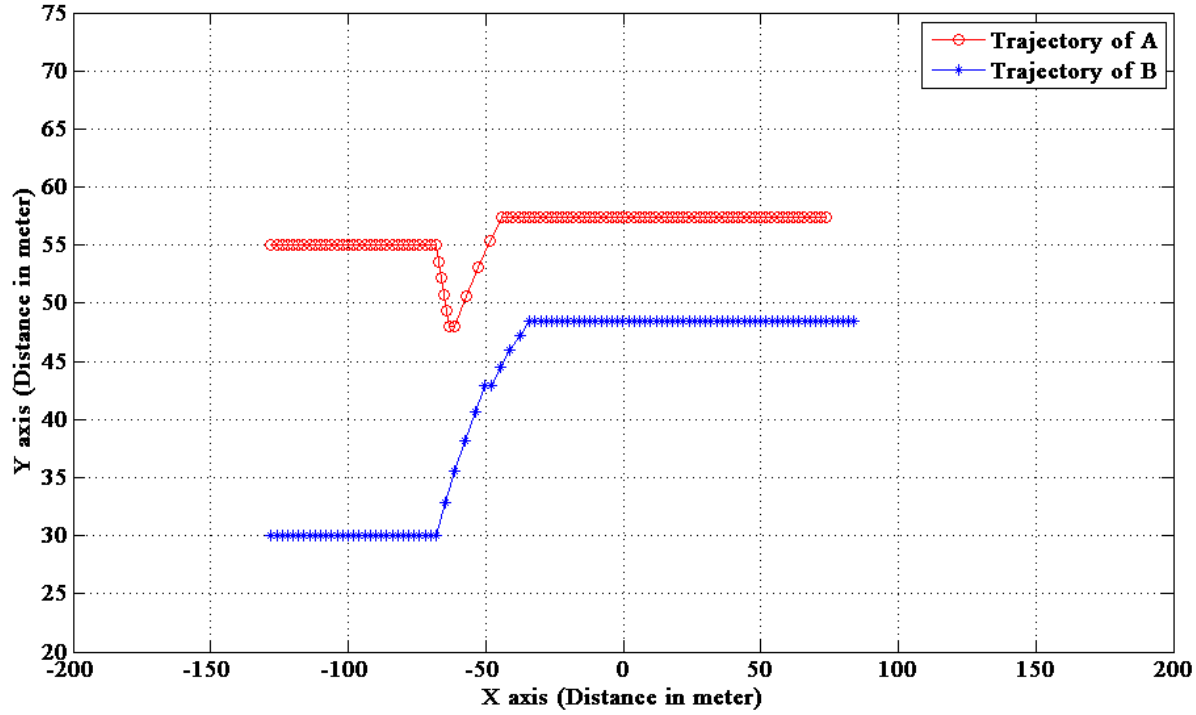

Figure 4.5 Trajectory of vehicles A and B with controller

A comparison of vehicles trajectory, with and without controller is shown in figure 4.6

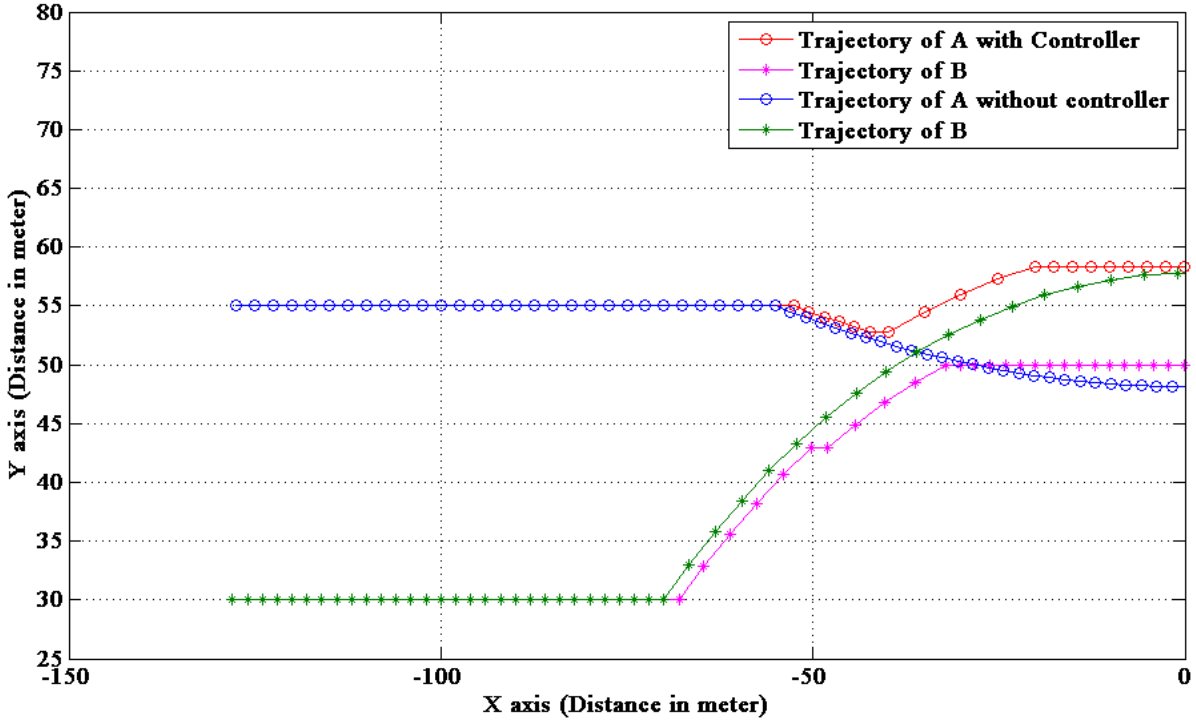

Figure 4.6 Trajectories of vehicles A and B with and without Controller

With the selection of certain parameters there is a situation (shown in figure 4.5) where two vehicles are going to collide, but ceased by the help of controller action. Here controller is placed on one vehicle, and the controller logic is such that commence of its action depends upon the distance between the two vehicles at a time. From the state vector, controller is aware of the coordinate of the vehicle in which it is placed, and the other vehicle in case of other vehicle is within its range. From the coordinate information controller can calculate the distance between two vehicles by employing Pythagorean Theorem. If this distance is good enough to keep two vehicles out of their range, there should be no collision. This condition can be shown in figure 4.5 (for first few seconds of journey). According to the selected parameter the clearance distance between two vehicles would be 10 meter (as the radius of range of vehicles are selected as $R_1 = 5$ meter and $R_2 = 5$ meter) for which should will be no activity of controller.

After few seconds distance between two vehicles is less than 10 meter, so the controller starts to litigate. As a result of that the orientation of the vehicle (in which controller is placed) is reversed for computing the next coordinate of its trajectory. That is how controller compelled the vehicles to preserve a certain distance between them. Due to reversed orientation the change of track of vehicle can be witnessed from figure 4.5.

A comparison between the trajectory of vehicles which can be prevail with and without employing controller is exhibited in figure 4.6. From this figure we can experience the deviation of trajectories which a vehicle should adopt when there is controller on the vehicle and a sealed prospect of collision. Thus figure 4.6 depicts the automatic control for the freely moveable vehicles to deflect the collision.

4.3 Simulink model and result for vehicles movement with Controller output affecting both the vehicles

The simulation results presented in this section represents a case where a local controller is included in both the vehicles for avoiding the collision between the vehicles which could occur during their free move. The simulation model describing vehicles with the controller action taking place on both the vehicles is shown in figure 4.7. MATLAB coding for this simulation is shown in APPENDIX-II.

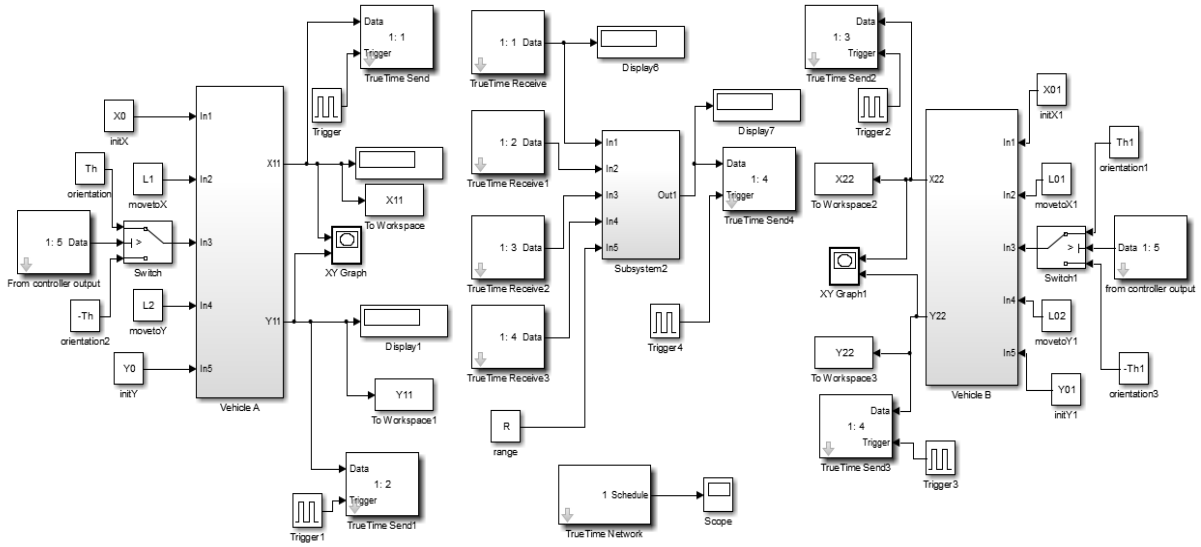

Figure 4.7 Simulink model of vehicles A and B where controller action takes place on both vehicles

The controller designed in this dissertation is a local controller, so there should be two different controllers for both the vehicles. The parameters used for this simulation are shown in Table 4.4. The Simulink result is shown in figure 4.8.

Table 4.4 initial parameter for vehicles A and B with controller in both vehicles

Parameters for Vehicle A	
Starting coordinate	x_0 =-130 y_0 =55
Fixed orientation without controller action	θ =0º (degree), for first 30 second θ =2.5º (degree), rest of the time
Fixed orientation with controller action	θ =-2.5º (degree), rest of the time
Translational length along X and Y axis	l_1 =2.5 meter l_2 = 0 meter
Transmission radius	R_1 =10 meter

Parameters for vehicle B	
Starting coordinate	x_0 =-130 y_0 =30
Fixed orientation without controller action	θ =0º (degree), for first 30 second θ =-2.45º (degree), rest of the time
Fixed orientation with controller action	θ =2.5º (degree), rest of the time
Translational length along X and Y axis	l_1 =2 meter l_2 =0meter
Transmission radius	R_2 =10 meter

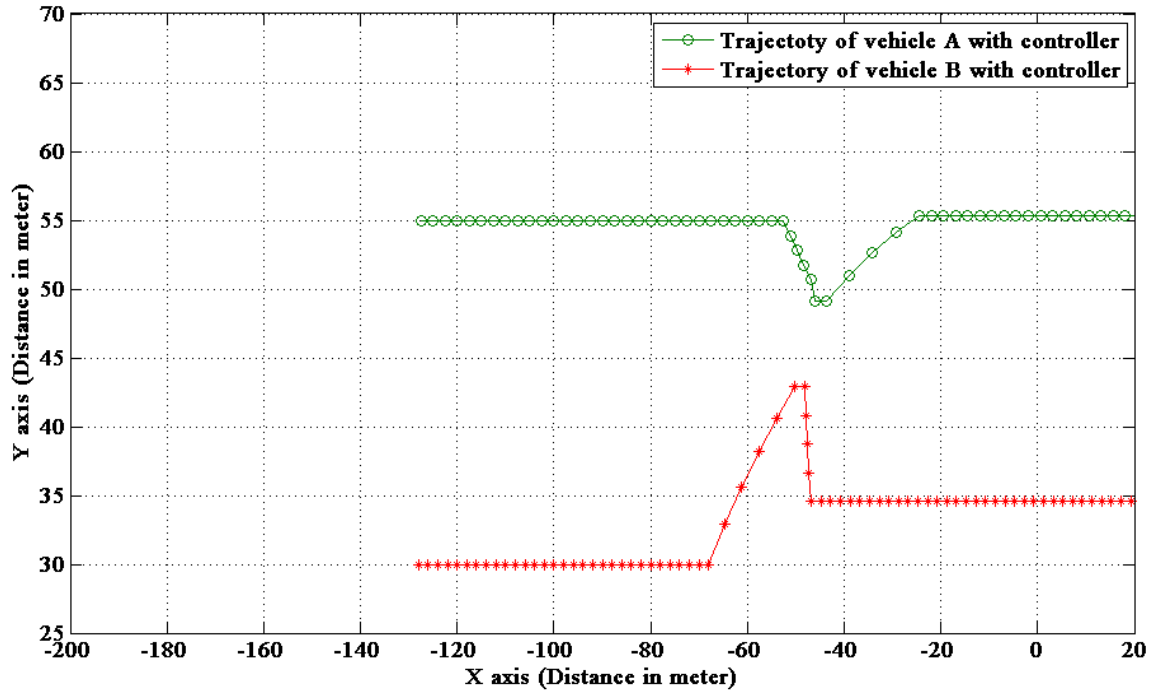

Figure 4.8 Trajectories of vehicles A and B with controller action in both vehicles

A comparison of vehicles trajectory, without controller and with controller action taking place in both vehicles is shown in figure 4.9.

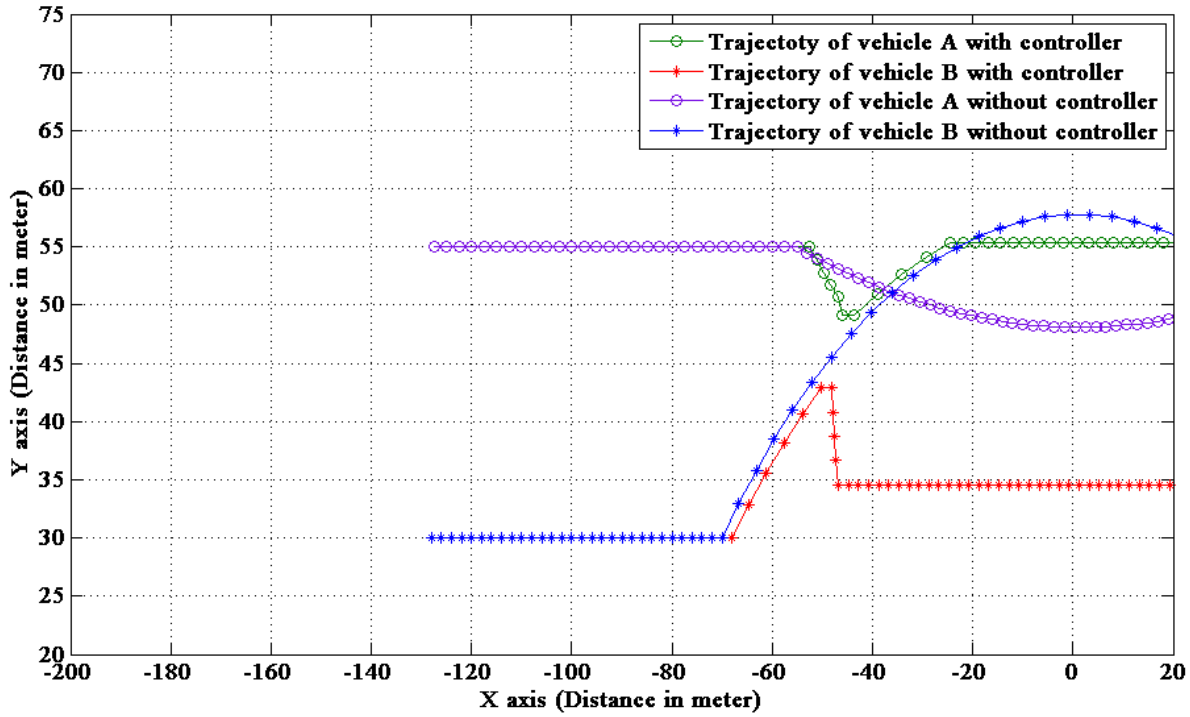

Figure 4.9 Comparison between vehicles trajectories with and without controller for both vehicles

The selection of certain parameters makes a situation (shown in figure 4.8) where two vehicles are going to collide, but avoided by the help of controller action which affects both the vehicles. Here the controller output is fed to both the vehicles, and the controller logic is same as explained in the previous experiment. In this simulation according to the selected parameters the clearance distance between two vehicles would be 20 meter (as the radius of range of vehicles are selected as R_1 = 10 meter and R_2 = 10 meter) for which definitely there is no activity of controller.

After few seconds distance between two vehicles is less than 20 meters, so the controller starts to litigate. As a result of that the orientation of the vehicles (in this case controller affects both the vehicles) are reversed for computing the next coordinate of its trajectories. That is how controller compelled the vehicles to preserve a certain distance between them. Due to reversed orientation, the change of track of vehicle can be witnessed from figure 4.8.

A comparison between the trajectory of vehicles which can be prevail with and without employing controller is exhibited in figure 4.9. From this figure we can experience the deviation of trajectories which both the vehicles should adopt when the controller is litigate for controlling both the vehicles and a sealed prospect of collision. Thus figure 4.9 depicts the automatic control of freely moveable vehicles to deflect the collision by the controlled action of both the vehicles.

Chapter 5

Conclusion and Future Scope

5.1 Conclusion

Automatic control for two freely moveable vehicles has been employed in this thesis. From the theoretical analysis and MATLAB Simulink result it is unclouded that if the controller is embedded with the vehicles, the possibility of collision between the vehicles is averted. The Kinematic model for the vehicles has been developed here. The MATLAB Simulink result depicted here to show the trajectory of the vehicles with the specific parameters.

Introduction of TrueTime Toolbox are being discussed here which has been validate with some examples. Different orders of systems are being controlled by the discrete PID controller with the help of TrueTime Simulator. The Simulink result depicts the system performance as they should be.

5.2 Future Scope

From the overall study covered by the dissertation we can provided some new idea for future study which can give this work a pulse. Such as

1) A controller can be designed which will be able to control any number of vehicles around it (i.e. a generalised control logic), with the help of provided theory.
2) The input parameters can be applied as variable, i.e. at every point of the journey the linear velocity and orientation can be different for the vehicles.
3) The controller can be designed such that controller action is designated to change any of the input parameters, i.e. to avoid collision controller can change the linear velocity or the orientation of the vehicles.
4) Practical implementation of this designed simulation will be a great realisation for this proposed theory.

Reference

1. A. Dahbi, M. Hachemi "Control of a Wind Turbine Based on PMSG and Connected to the Grid". September 2012 International Review of Automatic Control;Sep2012, Vol. 5 Issue 5, p553

2. D. V. V. V. CH. Mouli, K. Dhanvanthri "Analysis & Design of Closed Loop Control Using PID to Enhance Voltage Stability for STATCOM". September 2012 International Review of Automatic Control;Sep2012, Vol. 5 Issue 5, p560

3. Petros A. Ioannou, C.C. Chien "Autonomous Intelligent Cruise Control". IEEE Transactions on Vehicular Technology, Vol. 42, No. 4, November 1993

4. Jana Paulusova, Maria Dubravska "Neuro-Fuzzy Predictive Control". 2015 International Conference on Process Control (PC) June 9–12, 2015, Strbske Pleso, Slovakia.

5. Alain Liegeois "Automatic Supervisory Control of the Configuration and Behaviour of Multibody Mechanisms" IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMc-7. No. 12, December 1977.

6. WL Garrard, JM Jordan "Design of nonlinear automatic flight control systems" Automatica, Vol. 13, No. 5, 1977, p. 497-505.

7. W Wen "A dynamic and automatic traffic light control expert system for solving the road congestion problem" Expert System with Application, Volume 34, Issue 4, May,2008, pages 2370-2381.

8. E Irving, C Miossec, J Tassart "Towards efficient full automatic operation of the PWR steam generator with water level adaptive control" British Nuclear Energy Society, London; 430 p.

9. S Yin, H Luo, S.X .Ding "Real-time implementation of fault-tolerant control systems with performance optimization" IEEE Transactions on Industrial Electronics, vol. 61, No. 5, May, 2014
10. Miroslav Krstic, Ioannis Kanellakopoulos, Petar V. Kokotovic "Nonlinear Design of Adaptive Controllers for Linear Systems" IEEE Transactions on Automatic Control, vol. 39, no. 4, April 1994.
11. Willie D. Jones "Keeping Cars from Crashing" IEEE Spectrum, September 2001, pp 40-45.
12. R. Lenain, B. Thuilot, C. Cariou, and P. Martinet, "High accuracy path tracking for vehicles in presence of sliding: Application to farm vehicle automatic guidance for agricultural tasks," Autonomous Robots, vol. 21, no. 1, pp. 79–97, 2006.
13. R. Ronen and S. Arogeti, "Coordinated path following control for a group of car like vehicles," in 2012 12th International Conference on Control Automation Robotics Vision (ICARCV), Dec 2012, pp. 719–724
14. Itai Arad, Shai Arogeti, Rami Ronen "Coordinated Path Following Control for a Group of Car-like Vehicles with an Application to Intelligent Transportation System" 2014 13th International Conference on Control, Automation, Robotics & Vision Marina Bay Sands, Singapore, 10-12th December 2014 (ICARCV 2014)
15. Rami Ronen, Shai Arogeti "Coordinated Path Following Control for a Group of Car-like Vehicles" 2012 12th International Conference on Control, Automation, Robotics & Vision Guangzhou, China, 5-7th December 2012 (ICARCV 2012).
16. Jawhar Ghommam, Hasan Mehrjerdi , Maarouf Saad , Faical Mnif "Formation path following control of unicycle-type mobile robots" Robotics and Autonomous Systems 58 (2010) 727-736.
17. Mitsuji Sampei, Takeshi Tamura, Tadaharu Kobayashi, and Nobuhiro Shibui "Arbitrary Path Tracking Control of Articulated Vehicles Using Nonlinear Control Theory" IEEE Transactions on Control Systems Technology, vol. 3, no. I , March 1995.

18. LEI Chen, MA Baoli "A Nonlinear Formation Control of Wheeled Mobile Robots with Virtual Structure Approach" Proceedings of the 34th Chinese Control Conference July 28-30, 2015, Hangzhou, China.

19. Joel Hoff, George Bekey "An Architecture for Behaviour Coordination Learning" - Neural Networks, 1995. Proceedings., IEEE International conference.

20. P. Maes, R.A. Brooks, "Learning to Coordinate Behaviours," AAAI, Boston, MA, pp. 796-802,1990.

21. M.J. Mataric, "Reward Functions for Accelerated Learning," Proceedings of the Eleventh International Conference on Machine Learning, 1994.

22. Hiroaki Yamaguchi " A Cooperative Hunting Behaviour by Mobile Robot Troops" Proceedings of the 1998 IEEE International Conference on Robotics & Automation Leuven, Belgium May 1998.

23. H. Yamaguchi, "Adaptive formation control for distributed autonomous mobile robot groups," in Proc. 1997 IEEE Int. Conf. on Robotics and Automation, Albuquerque, NM, USA, pp. 2300-2305, April 1997.

24. Rodney A. Brooks "A Robust Layered Control System For A Mobile Robot" IEEE Journal of Robotics and Automation, vol. RA-2, no. I , March 1986.

25. Tucker Balch, Ronald C. Arkin "Behaviour-based Formation Control for Multi-robot Teams" IEEE Transactions on Robotics and Automation, vol. XX, no. Y, 1999.

26. Jagannathan Sarangapani , Travis Dierks "Control of Nonholonomic Mobile Robot Formations: Backstepping Kinematics into Dynamics" 16th IEEE International Conference on Control Applications Part of IEEE Multi-conference on Systems and Control Singapore, 1-3 October 2007.

27. G. L. Mariottini, G. Pappas, D. Prattichizzo, and K. Daniilidis, "Vision-based Localization of Leader-Follower Formations," Proc. Of the IEEE Conference on Decision and Control and , pp 635-640, Dec.2005.

28. Jaydev P. Desai, Jim Ostrowski, and Vijay Kumar, "Controlling Formations of Multiple Mobile Robots", Proc. IEEE International Conference on Robotics and Automation, pp. 2864-2869, Leuven,Belgium, May 1998.

29. X. Li, J. Xiao, and Z. Cai, "Backstepping Based Multiple Mobile Robots Formation Control,"Proc. IEEE International Conference on Intelligent Robots and Systems, pp 887-892, August 2005.

30. Jian Chen, Dong Sun, Jie Yang, Haoyao Chen "Leader–Follower Formation Control of Multiple Non-holonomic Mobile Robots Incorporating a Receding-horizon Scheme" The International Journal of Robotics Research OnlineFirst, published on May 28, 2009.

31. Luca Consolini, Fabio Morbidi, Domenico Prattichizzo, Mario Tosques "Leader–follower formation control of nonholonomic mobile robots with input constraints" Elsevier Automatica 44 (2008) 1343–1349.

32. J. Sanchez and R. Fierro, "Sliding mode control for robot formations," in Proc. IEEE International Symposium on Intelligent Control, 2003, pp. 438–443.

33. LIU Shi-Cai, TAN Da-Long, LIU Guang-Jun "Robust Leader-follower Formation Control of Mobile Robots Based on a Second Order Kinematics Model" Acta Automatica sinica, Vol-33, No-9, September, 2007.

34. Zhao Weihua, Tiauw Hiong Go and Eicher Low "Formation Flight Control Using Model Predictive Approach" 47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition 5 - 8 January 2009, Orlando, Florida.

35. S. Joe Qin, Thomas A. Badgwell "A survey of industrial model predictive control technology" Control Engineering Practice 11 (2003) 733–764.

36. V. Manikonda, P. Arambel, M. Gopinathan, R. Mehra, and F. Hadaegh,"A model predictive control-based approach for spacecraft formation keeping and attitude control," in Proceedings of the 1999 American Control Conference, vol. 6, 1999, pp. 4258–4262 vol.6.

37. J. Ghommam, H. Mehrjerdi, M. Saad, and F. Mnif, "Formation path following control of unicycle-type mobile robots," Robotics and Autonomous Systems, vol. 58, no. 5, pp. 727–736, 2010.

38. R. Beard, J. Lawton, and F. Hadaegh, "A coordination architecture for spacecraft formation control," IEEE Transactions on Control Systems Technology, vol. 9, no. 6, pp. 777–790, 2001.

39. Dan Henriksson, Anton Cervin, Karl-Erik Arzen "TrueTime: Simulation of control loops under shared computer resourses" 15th Triennial World Congress, Barcelona, Spain.

40. G. Hemingway, J. Porter, N. Kottenstette, H. Nine, C. vanBuskirk, G. Karsai, J. Sztipanovits "Automated Synthesis of Time-Triggered Architecture-based TrueTime Models for Platform Effects Simulation and Analysis" .

41. Anton Cervin, Martin Ohlin, Dan Henriksson "Simulation of Networked Control Systems using Truetime" Department of Automatic Control LTH Lund University, Sweden.

42. T. Chvostek, A. Kratky, M. Foltin "Simulation of network using TrueTime Toolbox" Slovak Research and Development Agency, Grant No. APVV-20-023505.

43. Miroslav Kostadinovic, Mile Stojcev, Zlatko Bundalo, Dusanka Bundalo "Simulation model of DC servo motor control" 14th International Power Electronics and Motion Control Conference, EPE-PEMC 2010.

44. K.E. Arzen, M. Ohlin, A. Cervin, P. Alriksson, D. Henriksson "Holistic Simulation of Mobile Robot and Sensor Network Applications Using TrueTime" Proceedings of the European Control Conference , Kos, Greece, July 2-5, 2007.

45. L. Farkas, J. Hnat "Simulation of networked control system using TrueTime" Slovak Research and Development Agency under the contract APVV-0337-37.

46. Houjun Wang, Bin Yan, Xiaojia Zhou，Li Li, Benliang Li "Route Simulation Research of Wireless Sensor Networks based on Truetime" National Natural Science Foundation of China (ID: 60673011), PhD. Foundation of China (ID: 20070614018).

47. Dan Henriksson, Anton Cervin, Karl-Erik Arzen "TRUETIME: Real-time Control System Simulation with MATLAB/Simulink" Department of Automatic Control Lund Institute of Technology Box 118, SE-221 00 Lund, Sweden.

48. Josep Guardia, Pau Marti, Manel Velasco and Rosa Castane "Enabling Feedback Scheduling in TrueTime" Research Report ESAII-RR-06-05 Automatic Control Department Technical University of Catalonia, March 2006.

49. Anton Cervin, Dan Henriksson, Martin Ohlin "TRUETIME 2.0 beta, Reference Manual" Department of Automatic Control, Lund University, June 2010.

50. Ohlin, M. (2006): "Feedback Linux scheduling and a simulation tool for wireless control." Licentiate Thesis, ISRN LUTFD2/TFRT- -3240- -SE, Department of Automatic Control, Lund University, Sweden.
51. Saeed B. Niku "Introduction to Robotics" Pearson Education Inc. Second edition, 2011.
52. John J. Craig "Introduction to Robotics" Pearson Education Inc., 2005.
53. K. D. Do "Formation Tracking Control of Unicycle-Type Mobile Robots With Limited Sensing Range" IEEE Transactions on Control system Technology, Vol. 16, No. 3, May 2008.

APPENDIX – I

2.4.3 Codes for Kernel Block

Initialization code

```
function controller_init(arg)
ttInitKernel('prioFP')
% create task data
data.MPn = 0;
data.MIn = 0;
data.MDn = 0;
data.MX = 0;
data.h =  0.010;
data.K = 1.2;
data.Ti = 1.1;
data.Td = 0.4;
data.yold = 0;
data.u = 0;
%periodic controller task
period = 0.01;
ttCreatePeriodicTask('controller_task', period, 'controller_code', data);
```

Controller Code

```
function [exectime, data] = controller_code(seg, data)
switch seg
    case 1
        y = ttGetMsg;    %obtain sensor value

        if isempty(y)
            disp('Error in controller: no msg received');
            y = 0;
        end
        r = ttAnalogIn(1);  % Read referance value
        data.MPn = data.K * (r-y);
        data.MIn = data.K * data.h/data.Ti*(r-y)+ data.MX;
        data.MDn = data.K * data.Td/data.h * (data.yold-y);
        data.u = data.MPn + data.MIn + data.MDn;
        data.MX = data.MIn;
        data.yold = y;
        exectime = 0.0005;
    case 2
        ttSendMsg (2, data.u, 100);
        exectime = -1;
end
```

2.4.4 Codes for wireless control of first order system

Actuator Initialization

```
function actuator_init
ttInitKernel('prioDM');
period = 0.01;
```

```
deadline = period;
offset = 0.0;
data.beta = 0.5;
prio = 2;
data.u = 0;
data.exectime = 0.005;
deadline = 10;
ttCreateTask('actuator_task',deadline,'actuator_code'); %for an aperiodic
task
% Network handler
prio = 1.0;
data = 'actuator_task';
ttCreateHandler('network_handler', prio, 'nwhandler_code', data);
ttAttachNetworkHandler('network_handler')
```

Actuator Task

```
function [exectime, data] = actuator_code(seg, data)
%  persistent u
switch seg
    case 1
       data.u = ttGetMsg;
       exectime = 0.0005;
    otherwise
        if ~ isempty(data.u)
            ttAnalogOut(1, data.u)
        else
            disp('Error: actuator received empty message!')
        end
        exectime = -1; %finished
end
```

Sensor Initialization

```
function sensor_init
ttInitKernel('prioDM');
period = 0.01;
deadline = period;
offset = 0.0;
data.beta = 0.5;
prio = 2;
data.u = 0;
data.exectime = 0.005;
ttCreatePeriodicTask('sensor_task',offset, period,'sensor_code');

% Network handler
prio = 1.0;
data = 'actuator_task';
ttCreateHandler('network_handler', prio, 'nwhandler_code', data);
ttAttachNetworkHandler('network_handler')
```

Sensor Task

```
function [exectime, data] = sensor_code(seg, data)
persistent y
switch seg
    case 1
```

```
        y = ttAnalogIn(1);
       exectime = 0.0005;
       if isempty(y)
            disp('Error in sensor:no msg received!');
             y = 0;
        end
    case 2
        ttSendMsg(3,y, 80); % send 80 bit data to node 3, controller
        exectime = 0.0004;
    case 3
        exectime = -1;
end
```

Network Handler Code

```
function [exectime, data] = nwhandler_code(seg, data)

ttCreateJob(data)
exectime = -1;
```

Controller Initialization

```
function controller_init
ttInitKernel('prioDM');
period = 0.01;
deadline = period;
offset = 0.0;
prio = 3;
data.u = 0;
data.exectime = 0.01;
data.K = 48;
data.Ki = 1.1;
data.Kd = 0.05;
data.beta =0.5;
data.h = 1;
data.N = 10;
data.u = 0;
data.Iold = 0;
data.Dold = 0;
data.yold = 0;
% data.rchan = 1;
% data.ychan = 2;
% data.uchan = 1;
ttCreateTask('controller_task', period,'controller_code', data); % to
create an aperiodic task
% Create and attach network interrupt handler
% prio = 1.0;
data = 'controller_task';
ttCreateHandler('network_handler', 1, 'nwhandler_code', data)
ttAttachNetworkHandler('network_handler')
ttNoSchedule('network_handler');
```

Controller Task

```
function [exectime, data] = controller_code(seg, data)
%   persistent y
switch seg
    case 1
        y = ttGetMsg;

        if isempty(y)
            disp('Error in controller:no msg received!');
            y = 0;
        end
        % read reference or interference
       r = ttAnalogIn(1);
        P = data.K*(r-y);
        I = data.Iold;
        data.u = P + I;
        exectime = 0.0005;
    case 2
        ttSendMsg(2, data.u, 80);%send 80 bit data to node 2, actuator
        exectime = -1;
end
```

APPENDIX-II

Code for Trajectory of vehicles without controller

```
clc;
X = -130;
Y = 120;
L1 = 2.5;
L2 = 3;
theta = ((pi/180)*(-1.9));

for i=1:100
X(i+1) = (X(i)*cos(theta))-(Y(i)*sin(theta))+(L1);
Y(i+1) = (X(i)*sin(theta))+(Y(i)*cos(theta))+(L2);
A(:,i) = X(i+1);
B(:,i) = Y(i+1);
end
plot(A,B)
hold all

U = -130;
V = 20;
thetai = ((pi/180)*(-1.1));

for n=1:100
U(n+1) = (U(n)*cos(thetai))-(V(n)*sin(thetai))+(L1);
V(n+1) = (U(n)*sin(thetai))+(V(n)*cos(thetai))+(L2);
C(:,n) = U(n+1);
D(:,n) = V(n+1);
end
plot(C,D)
```

Code for Trajectory of vehicles with condition

```
clc;
X = -130;
Y = 55;
L11 = 2.5;
L12 = 0;
L21 = 2;
L22 = 0;
% theta = ((pi/180)*(0));
U = -130;
V = 30;
% thetai = ((pi/180)*(0));
for i=1:100
    if i<31
        theta(i) = 0;
        thetai(i) =0;
    else
        thetai(i) = ((pi/180)*(-2.45));
        theta(i) = ((pi/180)*(.51));

%     else
%         theta(i) = 0;
%         thetai(i) =0;
    end
```

```
X(i+1) = (X(i)*cos(theta(i)))-(Y(i)*sin(theta(i)))+(L11);
Y(i+1) = (X(i)*sin(theta(i)))+(Y(i)*cos(theta(i)))+(L12);
A(:,i) = X(i+1);
B(:,i) = Y(i+1);
U(i+1) = (U(i)*cos(thetai(i)))-(V(i)*sin(thetai(i)))+(L21);
V(i+1) = (U(i)*sin(thetai(i)))+(V(i)*cos(thetai(i)))+(L22);
C(:,i) = U(i+1);
D(:,i) = V(i+1);
end
plot(A,B,'-o')
hold on
plot(C,D,'-*')
axis([-200 200 0 65])
```

Code of vehicle Trajectory with controller in one vehicle

```
clc;
X = -130;
Y = 55;
L1 = 2;
L2 = 0;
% theta = ((pi/180)*(0));
Range=10;
U = -130;
V = 30;
% thetai = ((pi/180)*(0));
for i=1:100
    if i<31
        theta(i) = 0;
        thetai(i) =0;


    else
        theta(i) = ((pi/180)*(2.5));
        thetai(i) =((pi/180)*(-2.45));
    end

X(i+1) = (X(i)*cos(theta(i)))-(Y(i)*sin(theta(i)))+(L1);
Y(i+1) = (X(i)*sin(theta(i)))+(Y(i)*cos(theta(i)))+(L2);
A(:,i) = X(i+1);
B(:,i) = Y(i+1);
U(i+1) = (U(i)*cos(thetai(i)))-(V(i)*sin(thetai(i)))+(L1);
V(i+1) = (U(i)*sin(thetai(i)))+(V(i)*cos(thetai(i)))+(L2);
C(:,i) = U(i+1);
D(:,i) = V(i+1);
E(i) = (sqrt(((A(i)-C(i))^2)+((B(i)-D(i))^2))) - Range;
    if  E(i) <= 0.01
        theta(i) = -((pi/180)*(-2.5));
    end
end
plot(A,B,'-or')
hold on
plot(C,D,'-*r')
axis([-200 200 0 65])
```

Code of vehicle Trajectory with controller controlling both the vehicle

```
clc;
X = -130;
Y = 55;
L1 = 2.5;
L2 = 0;
L11 =2;
L22 =0;
% theta = ((pi/180)*(0));
Range=20;
U = -130;
V = 30;
% thetai = ((pi/180)*(0));
for i=1:100
    if i<31
        theta(i) = 0;
        thetai(i) =0;

    else
      thetai(i) = ((pi/180)*(-2.45));
      theta(i) = ((pi/180)*(2.5));
    end

X(i+1) = (X(i)*cos(theta(i)))-(Y(i)*sin(theta(i)))+(L1);
Y(i+1) = (X(i)*sin(theta(i)))+(Y(i)*cos(theta(i)))+(L2);
A(:,i) = X(i+1);
B(:,i) = Y(i+1);
U(i+1) = (U(i)*cos(thetai(i)))-(V(i)*sin(thetai(i)))+(L11);
V(i+1) = (U(i)*sin(thetai(i)))+(V(i)*cos(thetai(i)))+(L22);
C(:,i) = U(i+1);
D(:,i) = V(i+1);
E(i) = (sqrt(((A(i)-C(i))^2)+((B(i)-D(i))^2))) - Range;
    if  E(i) <= 0.01
        theta(i) = -((pi/180)*(-2.5));
        thetai(i) = ((pi/180)*(2.45));
    end
end
plot(A,B,'-or')
  hold on
 plot(C,D,'-*r')
 hold on
axis([-200 200 0 65])
```