uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Hany Geris

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Cooperative Multi-Robot Relative Positioning Using Round-Trip Ultrasonic Ranging Network in Indoor Environment

TITRE DE LA THÈSE / TITLE OF THESIS

Prof. A. Fahim

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

Prof. W. Gueaieb

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Prof. P. Payeur

Prof. T. Green

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Cooperative Multi-Robot Relative Positioning Using Round-Trip Ultrasonic Ranging Network in Indoor Environment

Hany Geris

Thesis submitted to the Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements of
MASTER OF APPLIED SCIENCE

in Electrical Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
University of Ottawa
Ottawa, Canada
June 2009

Your file Votre référence
ISBN: 978-0-494-59867-2
Our file Notre référence
ISBN: 978-0-494-59867-2

This thesis is dedicated to my Lord and Saviour, Jesus Christ, who is the true source of all knowledge, understanding, and wisdom.

It is also dedicated to my beloved wife, my two brothers, my parents, and Fr. Mikhail Fam, who have always been supportive of me and keep encouraging me from day to day.

Abstract

Research on ultrasonic ranging localization networks is being pursued for use in cooperative multi-robot systems. Many network topology variants have been developed. However, it is difficult to meaningfully compare them. This is due to the utilization of different terminologies in describing their conceptual design instead of exploiting the popular computer network terms.

In this thesis, a novel round-trip ultrasonic positioning network is developed for indoor environments. The proposed technique keeps the overall hardware cost at minimum as the synchronization between the ultrasonic transceivers does not use real time RF burst signals for ranging system time of flight coordination.

Moreover, the proposed network ensures a reasonably high position estimation accuracy. It can be used in either of two modes of operation: The communication-dependent mode which is suitable for tightly coupled tasks, and the communication-independent mode which suits the execution of loosely coupled tasks. This research work presents the topology, methodology of network operation, and computer simulations of the positioning system operation to illustrate its performance.

A prototype proposed localization network is implemented and tested. The experimental results proved the feasibility of the proposed solution and highlighted implementation issues.

Acknowledgments

I would like to express my sincere thanks to Professors Atef Fahim and Wail Gueaieb for directing my studies. Their extensive discussions and critical assessment of the work had a positive influence on the final results of this research project.

Finally, I gratefully acknowledge the support of my supervisors and the University of Ottawa, for funding this research work and my studies.

Table of Contents

List of Figures

List of Tables

Nomenclature

Δ_i	is node i echo cancellation time delay (msec) after broadcasting an ultrasonic burst, and is necessary to prevent the reception of false echoes.
t_{ij}	is the time at which node i receives the ultrasonic burst from node j.
T_{ij}	is the Time Of Flight (TOF) between node i and node j (msec.).
Δ_b	is the system blank time (msec.).
N	is the total number of mobile nodes.
M_i	is the network node i.
V_s	is the speed of sound (m/sec.).
T	is the ambient temperature (°C).
T_{max}	is the maximum TOF between any two nodes in the network (msec.).
D_{max}	is the maximum allowable distance between any two nodes in the network.
D_c	is the maximum range surrounding a broadcasting node, so that any reflected echo normalized power from any node located inside this range may be received as a false echo at that node.
μ	is the ultrasonic reflection and dissipation factor at any node.
P	is the localization cycle period.

Chapter 1
Introduction

1.1 Motivation

Multi-robot systems have widely been employed to serve in many indoor environments such as hospitals, museums, and banks. They have been used for performing different tasks such as night surveillance, and transporting heavy materials or equipments. They have been also used to replace humans in performing tasks that are dangerous such as transporting chemical or nuclear material. There is a large and growing demand for multi-robot systems. Multi-robot system may potentially have uses in the "clean room" environments required for the manufacturing in the medical, the pharmaceutical, and the electronics industries. Hospitals and other large public facilities have already demonstrated a demand for assistive cleaning and delivery of supplies by service robots. The ability to share information and tasks among robots can lead to more effective solutions to tasks of environment monitoring, surveillance, and navigation among others. Regardless of the specific application, the cooperation of the mobile robots is usually divided into three separate parts: localization, path planning and path execution. Localization is the most critical one as it is the process of determining the location of the robot with respect to fixed reference beacons or relative to other robots. Although, active research has been conducted to develop a reliable and low-cost self-localization method, cooperative localization is relatively new. This thesis presents research work carried out to develop a new approach for the cooperative multi-robot localization. This is motivated by the increasing demand on faster serving robots.

1.2 Problem Statement

Given a group of robots where each is equipped with a low-cost sensing technique, the cooperative localization problem can be defined as how this group of robots can combine their sensors' measurements to improve their individual localization performance. This approach is motivated by the fact that robots within a group can often identify each other and communicate their sensory measurements. Many localization approaches with wide ranges of accuracy and update rates have been proposed and their implementation reported in the literature. In order to compare these approaches, their performance factors must be analyzed. These factors can be briefly identified as follows;

- **Accuracy:** This term indicates the deviation of the estimated location from the real one. There are many factors that affect the localization system accuracy; the sensing technique hardware resolution, the precision of the absolute or relative position estimation method, the real time characteristics of software coded for timing or event sequencing. In view of robots motion, the accuracy is also directly coupled with the positioning method update rate relative to the robots speeds. The faster the update rate of the localization method that can provide an accurate pose estimate for each robot, the higher the accuracy of the overall system due to the decrease in uncertainty in position estimation. This coupling between the accuracy and navigation speed is primarily related to Most of the localization systems employ different data communication channels to exchange data among robots. Unfortunately, these channels may invoke a communication latency which decreases the update rate and consequently the accuracy.
- **Hardware cost:** Such a cost becomes a significant issue that could hamper the deployment of a system particularly in cases where numerous mobile robots equipped with localization mechanisms are required.

- **Environment setup:** This factor defines the required environment infrastructure necessary to support cooperative localization. An example of such a factor would be the presence of fixed beacons appropriately located in corridors and open spaces. As the number of such factors decreases, the more attractive is the localization method.

Many cooperative localization approaches for indoor environment have been developed and tailored for particular purposes. For example, a given system may be suitable for a large infrastructure setup but it may be unsuitable for high navigational speeds.

In general, there are three primary sensing technologies that have been employed for dealing with the localization problem; odometry, vision sensors, and ranging sensors. The most common odometry sensor used at present employs the wheel incremental encoders. While these low cost sensors do not require any environment setup, they suffer from the accumulation of errors due to wheel slipping that tend to increase with traveled distance and consequently adversely affect accuracy. Vision sensors like cameras have been utilized to estimate the robot position and orientation relative to landmarks in the environment. The main drawback of such a localization method is the high computational cost associated with image processing and the requirement for identifying easily recognizable landmarks and their accurate position in the environment.

The third solution involves the use of ranging sensors. The least expensive of these, and the most suitable for short to medium distance localization, are ultrasonic ranging based sensors. Ultrasonic sensors can be utilized to estimate the inter-robot distances simply by estimating the corresponding time-of-flight. The objective of this work is to develop an ultrasonic ranging network that is capable of achieving accurate multi-robot cooperative positioning at different navigational speed scale. Although the ultrasonic transmit-receive mode is more suitable for this application, it requires a transmitter-receiver synchronization mechanism. A typical method that can be used for

that purpose is the infra red (IR) or radio waves (RF). Brind'amour [1] developed a synchronization method by employing an RF burst for signaling the beginning of the ultrasonic wave burst in a real time operation mode. This additional synchronization mechanism comes at an additional cost. These synchronization signaling channels are completely different than the RF data communication channels due to the need for real time operation. As the latter can provide data exchange in non real time operation mode and in form of data packets whereas the synchronization RF channels can only convey a simple data byte representing the robot ID as in [1].

Three problems are addressed: The global synchronization of ultrasonic transmitter-receiver without using any synchronization mechanism to reduce the overall system cost, the ultrasonic sensors fast firing to increase the localization network update rate, and a time-of-flight estimation method that can achieve pose estimation accuracy in the range of few centimeters at high robot navigational speed (1-2 m/sec) in indoor environments.

1.3 Scope

The scope of this research includes the development and implementation of a localization network that employs the ultrasonic sensing technology at minimal cost. This is to be accomplished without using any additional hardware required for a sensory synchronization mechanism.

Along with the goal of reducing system cost, the proposed network should ensure a reasonable high position estimation accuracy. The system would use either of two modes of operation; the communication-dependent mode which is suitable for tightly coupled tasks, or the communication-independent mode which suits the execution of loosely coupled tasks. The proposed localization network is evaluated experimentally. In its current implementation, the localization system provides the inter-robot relative distances. An off-line multidimensional scaling mathematical

algorithm combined with the orthogonal Procrustes statistical method, is used to obtain the corresponding absolute positions.

It should be noted that the conclusion of this research work is generic in nature and is not restricted to a specific robot platform.

1.4 Summary of Contribution

The main contributions of this research project relate to the cooperative localization in the field of multi-robot systems, and more specifically:

- Development of round-trip localization protocol with two operation modes (Chapter 3);
- Numerical simulation of the proposed localization protocol (Chapter 3);
- Design and implementation of an omnidirectional ultrasonic localization module (Chapter 4);

1.5 Organization of Contents

The remainder of this thesis is organized as follows:

Chapter 2 provides a review of previous work done on mobile robot localization. The discussion focuses only on the sensors and techniques used for localization in indoor environments. Particularly, an insight is provided into the ultrasonic based localization systems by analyzing many of their possible design alternatives.

Chapter 3 details the analysis of the proposed round-trip ultrasonic positioning system. Three different variations to the proposed system are introduced to enhance the performance and increase the update rate of the network. Accuracies of these variations are compared with respect to position estimation method precision and robot navigational speed using a developed MATLAB simulator.

Chapter 4 presents the engineering and implementation design of the proposed multi-robot

localization system. The chapter is composed of two parts. The first covers the hardware design which details the system components and electrical wiring. The second part exposes the high and low level software architecture.

Chapter 5 presents the results of experiments carried out to estimate the accuracy of the proposed round-trip localization network with respect to the hardware implementation and the software implementation. An off-line multidimensional scaling method is used to translate the relative distance, estimated by the proposed round-trip localizer to absolute positions. The accuracy of the proposed localization method is evaluated experimentally.

Chapter 6 concludes with a summary of the main contributions of this work to the multi-robot localization field. Potential ideas for future work are also highlighted.

Chapter 2
Background and Literature Review

2.1 Introduction

A group of robots working collaboratively can perform complex tasks that none of them can individually. This cooperation, however, requires a global self localization method that would yield the position and bearing angle of each robot at all times relative to others in the group or to reference beacons in the environment. Many localization approaches with wide ranges of accuracy and update rates have been proposed and implemented. There is a close relationship between a chosen localization method and the achievable level of coordination. The higher the accuracy of inter-robot distances and headings that can be gleaned from the localization system, the higher the level of coordination and the complexity of the task that can be achieved.

In this chapter, a review of mobile robots localization methods are presented. The discussion focuses only on the sensors and techniques used for localization in indoor environments. Section 2.2 introduces a classification of localization systems based on different sensors and sensing techniques. Because of its high accuracy, easy deployment, low cost and low power consumption, a large variety of ultrasonic location systems have been developed and reported in the literature. In Section 2.3., an insight is provided into this wide variety by analyzing and categorizing the different design features of the ultrasonic localization systems. Through the chapter, comparisons and summary tables are offered to support the discussion. The chapter is concluded with a short synthesis providing general guidelines for my contribution.

2.2 Localization Systems Classification

Many localization systems for mobile robots have been built and used either for experimental or for commercial purposes. Among them are; Active Bats [2], Cricket [3], and RADAR [4]. These systems use different sensor techniques and localization methods ranging from diffuse infrared cellular proximity sensors, ultrasound time of flight lateration, ultrasound proximity, and RF triangulation. Several schemes may be used for classifying these systems.

2.2.1 Sensors

A sensor is a device that responds to a physical stimulus (such as heat, light, sound, pressure, magnetism, or a particular motion) and transmits a resulting signal (such as for measurement or operating a controller). Figure 2.1 shows the sensing process in terms of energy conversion. The input signal is often an analog signal while the output signal can be either analog or digital.

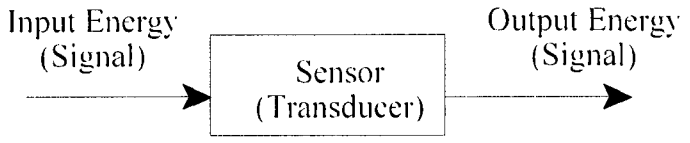


Figure 2.1: Sensing Process.

Reference [5] presents a survey that relates sensors to their respective sensing techniques. Since the interest of this work is indoor environment localization system, the following discussion will provide a brief survey about localization sensors currently used in indoor environments.

2.2.1.1 Odometry Encoders

Rotational encoders are used in wheeled vehicles to measure the angular displacement and translate it to a linear traveled distance. Starting from a known position, the present location of a vehicle can then be determined by accumulating measured values. Due to their low cost and high update rate, they have been widely used in the literature[6]-[8]. However, because of their most common use with friction based wheel traction, they are highly affected by errors caused by wheel slippage, encoder's resolution, and imperfect wheel geometry.

2.2.1.2 Inertial Navigation Sensors (INS)

Gyroscopes and accelerometers are used to measure the angular velocity and acceleration of a rotated shaft. By integrating the measured data once or twice, the position of the robot can be calculated. These sensors are self contained but suffer from the accumulation of errors caused by the integration process. The position drift rate may range from 1 to 8 cm/s, depending on the frequency of acceleration changes as detailed in [9].

2.2.1.3 Ultrasonic Sensors

Ultrasonic sensors are commonly used in indoor environments [5]. Ultrasonic sensors generate high frequency sound waves (≥40 KHz), which is beyond the human audible frequency range (20 Hz to 22 KHz). The propagation speed of the signal in the Air $V_{s,Air}$ (m/s) is given by

$$V_{s,Air} = 331+.6T \tag{2.1}$$

where T (°C) is the ambient air temperature.

As the temperature in an indoor environment is well-controlled, the speed of sound can be

considered constant. This should ideally lead to a highly accurate distance measurement if not for a number of shortcomings. Ultrasonic signals are incapable of propagating through walls and obstacles. They, rather, bounce back off such obstacles causing reflections, and add to the complexity of using such sensors in a crowded environment. This led researchers to combine ultrasonic sensors with other sensing techniques to overcome this limitation. Among such techniques are RF [3] [10], IR [11], and magnetic[12]. Ultrasonic sensors are usually utilized in either of two arrangements; pulse-echo and transmit-receive arrangements. In the pulse-echo mode, a sensor emits a sonic burst and waits to detect the reflected signal from an object. In the transmit-receive mode, the transmitter and receiver are placed at the two ends of a path the distance of which is to be estimated. A major limitation of the transmit-receive mode is that it requires a line of sight between the transmitter and the receiver.

2.2.1.4 Infrared (IR) Sensors

Infrared is an energy similar to visible light, but with a longer wavelength (750nm-1mm). Infrared energy is invisible to the human eye. The infrared energy is emitted by all objects at ordinary temperatures. It has a limitation of not propagating through walls or objects as in ultrasonic signals but it is not affected by radio or electromagnetic signals. There are two types of infrared systems; active and passive. In active systems, both objects or beacons are equipped with IR transmitters while in passive systems, objects are localized according to their natural emission of radiation. Another disadvantage of IR technology is that it is highly affected by other light sources in indoor environments such as fluorescent lighting. The accuracy of the IR technology is (0-6 inches) as reported in [5].

2.2.1.5 Radio Frequency (RF)

In indoor environments, the propagation of the RF signal is affected by many factors: multipath propagation, variations of temperature and humidity, opening and closing of doors, changes in the location of the furniture, and the presence and movement of people in the environment [14]. Current RF standards are not meant for localization. WLAN, GSM, bluetooth, and RFID can only provide low localization accuracy that is not suitable for the limited space inherent of indoor environments. Although UWB is a promising technology because of its high accuracy (6-10 cm) and its robustness against a multipath, it is not regulated yet to be used in indoor environments. Because of its low accuracy and sensitivity to dynamic changes in indoor environments, all current RF sensing based systems are not considered as good candidates for the localization problem in indoor environments. RF sensing technologies and their features are summarized in Table 2.1.

Technology	Frequency Range	Accuracy	Advantages	Disadvantages	References
WLAN (IEEE 802.11b/g)	2.4 GHz ISM band	2-100m	signal propagates through obstacles - Limited coverage.	highly affected by multipath - Highly affected with interference - low accuracy due to signal attenuation.	[15], [4], [16]
BLUETOOTH (IEEE 802.15)	2.4 GHz ISM band	2-10m			[14]
GSM	850 MHz, 1900 MHz	50-100m	Large Network Coverage - No need for new RF interfaces - Operates on licensed bands (i.e is not affected with inference).	High configuration cost.	[17]
UWB	3.1–10.6 GHz	6-10cm	Robust against multipath - Low power.	Expensive.	[18]
RFID	ISM , SRD	5cm-5m	Low power.	Low Range (1 - 2m) with passive tags.	[19],[20]

Table 2.1: RF Technologies.

2.2.1.6 Vision

Vision based localization systems have advanced considerably in the last decade. A literature review shows vision based localization systems to use different cameras such as CMOS [21] and omnidirectional ones [22]. The main disadvantage of vision systems is their need for high computational power required to deal with the complex image processing algorithms. Such computational power comes at the price of increased system cost if powerful computers are used,

or decreased update rate if low computing power is used. Furthermore, vision systems are generally dependant on the lighting conditions in the environment. In some environments, vision systems are rejected by occupants due to privacy related issues.

2.2.2 Distance and Angle Measuring Techniques

There are many techniques used to calculate the relative distances and orientation of each robot in a multi-robot configuration. The angle of arrival (AoA) technique employs an array of angularly displaced sensors with high directivity signal pattern to map relative angles between robots in multi-robot configuration. The AoA technique can be replaced efficiently by the time of arrival (ToA) technique as the latter can estimate the relative distances between robots and then infer the relative angles between the robots through trigonometry. The advantage of AoA technique over ToA is that no time synchronization is required. However, the AoA requires more complex hardware and the angle measurements degrade as the mobile robots move away from each other. Despite the need for high resolution timing to achieve an accurate pose estimation by the time of arrival technique, its computational cost is kept at a minimum compared to other techniques. The received signal strength (RSS) is another recently developed technique, it uses the RF signal strength to measure the inter-robot distances. Although the RSS technique is applicable at low cost, as it uses the common WLAN network technology, its performance is highly affected by the multipath problem. Consequently, it has to rely on complex additional algorithmic support to remedy its limitations as detailed in [23]. Table 2.2 summarizes the principal of each technique, its pros and cons, and references in literature.

Technique	Principal	Pros.	Cons.	References
Angle of Arrival(AoA)	Use array of angularly displaced sensors with high directivity signal patterns (ultrasonic, IR) to estimate and map relative angles between the neighbors.	Less computational cost - No time synchronization needed.	Requires costly antenna arrays on each node.	[24],[25]
Time of Arrival (ToA)	The time it takes for a signal to propagate from one point to another can be used to determine the distance between the points - Utilize different signals such as ultrasonic and RF signals.	Less computational cost - Time synchronization needed.	Require an additional communication channel for synchronization between transmitters and receivers - High timing resolution.	[26],[25]
Received Signal Strength (RSS)	Use a theoretical or empirical model to translate signal strength into distance.	Does not require additional hardware, it uses mostly the WLAN technology.	Very low accuracy - Highly affected with multipath effects and environmental effects (e.g building geometry and traffic).	[26],[23]

Table 2.2: Distance measuring techniques.

2.2.3 Localization Techniques

In many multi-robot applications, knowing the robot's location is of a paramount importance [27]. There are many methods used to estimate the location of every robot in multi-robot systems in indoor environments. The proximity method has the lowest computational complexity as it provides the position of a robot as "close to" a reference node rather than the metric measure,

distance and position, from that reference node. Alternatively, if multiple reference nodes are within range, the centroid between these nodes may yield a better pose estimate. The Active Badge localization system is based on proximity information acquired by ceiling-mounted infrared receivers [28]. These devices listen to unique sequences transmitted by badges worn by people. Therefore, the accuracy of the proximity method depends on the number of reference nodes in the environment . Both triangulation and trilateration have better accuracy as they use the ToA from the target robot to three reference nodes. In the multilateration technique, the range distances between the target robot and the reference nodes is determined by time difference of arrival (TDoA). Then, by solving the intersection of all hyperbolas that have their focal points at the reference points, the location of the robot can be determined [5]. Due to stochastic errors in TDoA estimations, the intersection of these hyperbolas is rarely a single point. Therefore, the location problem can be posed as an optimization problem and solved using, for example, a least squares method or an extended Kalman filter [26]. The scene analysis is another localization technique which employs the image processing of image or video frames gathered from vision sensors such as cameras. The main drawback of this technique, is its high computational complexity [32]. Table 2.3 summarizes the localization techniques and lists some of its references in the literature.

Technique	Principal	Comments	References
Proximity	The position of a target is approximated by selecting the location of the closest reference unit (e.g. beacon).	Very low accuracy depends on the no. of reference points in the environment.	[29],[30]
Triangulation	Gathers ToA measurements at the robot from at least three reference points. Then using simple geometric relationships, the location of the robot node can be calculated.	Fails if the robot is outside the triangle (geometric triangulation) - has large errors when the three reference points and the robot all lie on the same circle.	[5],[31]
Trilateration	Uses ranges to at least three references' points to find the unknown coordinates of the robot	Requires moderate computational complexity that depend on the number of reference points used for trilateration.	[5],[26]
Multilateration	Uses TDoA to calculate the distances from robot to the reference points then by solving the mathematical intersection of multiple hyperbolas, the location of robot can be estimated.		[26]
Scene Analysis	Uses image processing techniques to draw conclusions about the location of the robot.	High computation complexity.	[32]

Table 2.3: Localization techniques.

2.3 Wireless Ultrasonic Ranging Network

Because of its high accuracy, easy deployment, low cost and low power consumption, a large variety of indoor ultrasonic localization systems have been detailed in the literature. Many systems that employ the ultrasound technology were adopted in commercial systems as in Bat [2], InterSense [11], and Cricket [3]. A localization network is a wireless network consisting of spatially distributed

autonomous nodes using sensors to cooperatively estimate the pose of each node. In this section, an insight is provided into this wide variety, by analyzing and categorizing the different design features of the ultrasonic localization systems reported in [28].

2.3.1 Scalability

An important feature that identifies an ultrasonic localization network architecture is its scalability, i.e., the ability to increase the number of nodes constituting the network. In general, an ultrasonic localization network may contain two types of nodes; *reference nodes* and *mobile nodes*. Reference nodes may have fixed locations and are typically used as reference beacons [3]. Mobile nodes are typically affixed to moving targets whose locations need to be identified [33]. Increasing the number of reference nodes in the localization system adds more localization capabilities. When the localization system has no reference nodes, the mobile nodes must rely on their own sensors to navigate in the environment [5][34]. With one reference node, the localization system is only capable of detecting the presence of a mobile node as in Cricket [3], or determining the position of the mobile node relative to a fixed reference node. Although simple to implement, systems with one reference node cannot determine the orientation of each robot [35]. With two reference nodes, the coordinates of any robot can be determined by triangulation. The error range boundaries are highly affected by the location of the robot relative to the two reference nodes [35]. For 3-D positioning, more than two reference nodes are required, positioning is done by multilateration. An added computational complexity is required to solve the multilateration equations recursively [2][11]. Generally, the accuracy of 2D and 3D localization can be improved by adding more reference nodes. Table 2.4 shows a classification summary of ultrasonic ranging networks based on the number of reference nodes per site.

No. of reference nodes per site	Network features	References
0	Each robot relies on its on board sensors to calculate its relative position to other robots.	[5][34]
1	Reference node used to sense the presence of mobile nodes in its proximity or to gage its relative distance from it.	[3][35]
2	Used to locate the robot position in a 2-D space.	[35]
≥3	Using many reference nodes increases the probability of line of sight and decreases non-line-of-sight errors. If reference nodes are non-collinear, 3D position can be inferred.	[7],[11]

Table 2.4: Classification based on the number of reference nodes per site.

2.3.2 Secondary Non-Acoustic Communication Channel

In addition to the acoustic communication channels between the nodes, one or more relatively very high speed secondary data communication channels are often used. The secondary communication channels are used for communicating ID's to reference nodes to distinguish between them, or to broadcast the robot's absolute position to the reference nodes to enable the network agent to map all robots in its field [3]. This data communication channel is a typical wireless RF network which is capable of communicating data packets that can be as large as hundreds of kilobytes like in the case of conveying captured images or video frames among robots [22].

2.3.3 Network Topology

Network topology is another important feature that characterizes an ultrasonic ranging network architecture. The network topology term is traditionally associated with computer networks and it defines the organization of network elements, especially the physical and logical

interconnections between the nodes. Most of the work done in this area considers one acoustic channel per network. That is, all ultrasonic transducers in the network have the same resonant frequency. Since all the nodes in an ultrasonic ranging network can listen to the same acoustic channel, all of the developed network topologies are considered as physically connected through a bus. To evaluate the performance of network topologies, consider a network that consists of $N+1$ nodes with a maximum TOF between any two nodes being T_{max}. In this context, each node in the ultrasonic ranging network is assumed to be equipped with one transducer which may be triggered to act as ultrasonic transmitter (T) or as receiver (R).

2.3.3.1 Logical Star Topology

In this architecture, each mobile node is equipped with an ultrasonic and an RF receiver module. As shown in Figure 2.2, the node is equipped with an ultrasonic and an RF transmitter [3]. The reference node broadcasts simultaneously an RF burst signal and an acoustic signal to synchronize the reference node's ultrasonic transmitter and mobile nodes' ultrasonic receivers. It broadcasts its predefined location to the mobile nodes via an RF data communication channel. Then, each mobile robot calculates its own position based on the one way time-of-flight difference between the RF burst signal and the sonic signal. This topology is scalable to mobile nodes as no extra identification is needed at the reference node to detect new nodes. The update rate of this architecture is very fast since all mobile nodes in the range of the reference node can estimate their position relative to the reference node using one acoustic broadcast (i.e., bounded by T_{max}). Since the acoustic signal takes some time to fade out in the environment, the reference node has to wait at least that much time before another acoustic broadcast is initiated to prevent the crosstalk with the un-faded signal.

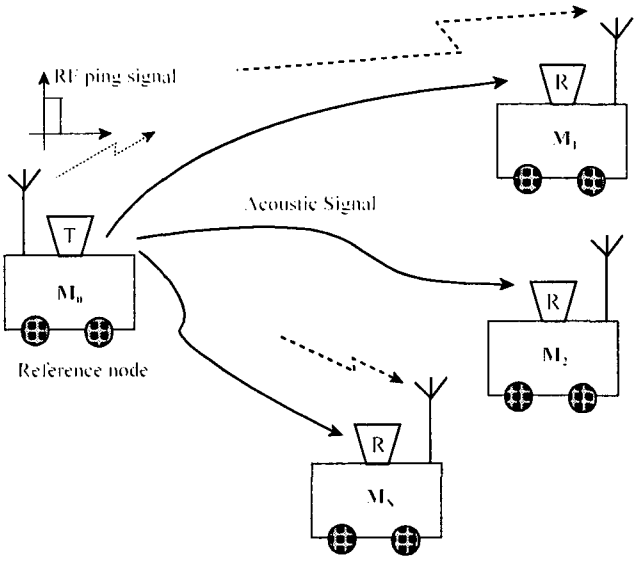


Figure 2.2: Logical Star Topology.

The overall cost for this architecture is fairly high. For a network with one reference node and N mobile nodes, the cost can be calculated as follows:

$$C_n = (N+1)(C_{US} + C_{RF}) \tag{2.2}$$

Where C_n, C_{US}, and C_{RF} are the cost of the network, the ultrasonic transducer (transmitter, receiver, or transceiver), and the RF burst transmitter/receiver respectively.

2.3.3.2 Logical Star Topology (without synchronization mechanism)

In this architecture, both the mobile nodes and the reference node are equipped with ultrasonic transceivers. The Time Division Multiple Access (TDMA) control method, as described in [37], is used to manage the acoustic channel access by all the nodes in the network. The reference node periodically broadcasts a binary coded request acoustic signal into the environment [37]. The request signal contains a binary identification (ID) of a specific mobile robot. All mobile nodes in the workspace receive this signal and decode the ID. After waiting for a predefined time delay, each

mobile node broadcasts a binary coded acoustic signal to the reference node. The reference node has a bank of filters matched to different codes which enable it to identify the source node. Then, it calculates the round-trip time-of-flight as detailed in [38]. This architecture is inexpensive compared to the star topology since it does not require an RF or IR burst communication channels. However, since it uses a round-trip TOF its update rate is slow and has a localization cycle period given by:

$$P = 2T_{\max}(N+1) \tag{2.3}$$

The added complexity resulting from the transmission of coded signals over a sonic wave and the increase in binary codes cross correlation noise, reduces the scalability of this architecture. Hori et al. [39] proposed an approach which uses the Frequency Division Multiple Access (FDMA) control method. In this approach each of the ultrasonic transmitters on mobile nodes are allowed to simultaneously fire at a unique frequency, thus sharing the acoustic communication channel bandwidth. Digital filters are hence required at the reference node to differentiate between the ultrasonic signals of the different mobile nodes. Ultrasonic transducers typically have a bandwidth of 2 kHz. This wide frequency range allows for a fair scalability without an appreciable increase in the update rate.

2.3.3.3 Fully meshed Topology

In this architecture, each node queries and estimates its distance from all other nodes in the network. Hence, the entire network has a built in redundancy since the distance between two nodes is estimated twice. This topology achieves better reliability than the star topology.

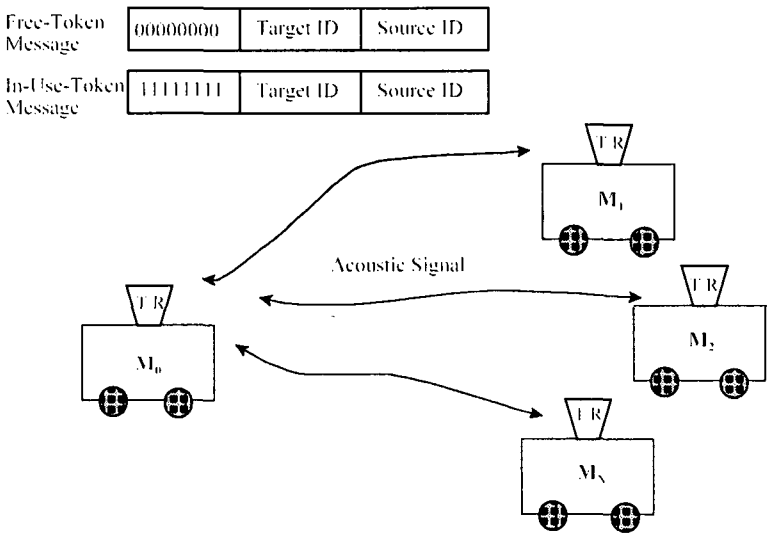


Figure 2.3: Token-ring fully meshed network topology.

It also circumvents the non-line-of-sight problems inherent to other topologies. In all of the afore discussed topologies, an extra reference node is needed for 2-D position estimation. This increases the algorithm's complexity. Because of its capability to build the Euclidean distance matrix for all network nodes, the fully meshed architecture can easily estimate the relative bearing angles between the nodes using trigonometric relations. Every node is equipped with only one ultrasonic transceiver resulting in a less expensive topology. This architecture is very similar to the token ring computer network and employs the binary coded ultrasonic signals explained earlier to identify the different nodes in the network as shown in Figure 2.3. As in the token ring network, a token message has three segments; a free or occupied token status segment, a destination ID segment, and a source ID segment. The node ID information in the destination and source segments sequence through all nodes in order. The token is passed from a message source node to a message destination node in sequence. Each node repeats the free token message and sends it to the next node in the sequence

if no measurement is required. If a node needs to measure its distance from another in the network, after receiving a free-token message it broadcasts a message consisting of three segments: in-use-token, target node code, and source node code. The node with the matching code will then reply by transmitting an acoustic response signal after a predefined time delay. A TOF of the round trip is calculated and the distance measurement is established. Due to the lack of central coordination between the nodes, the update rate is slow when compared to the other architectures. A comparative summary between the different architectures is given in Table 2.5.

	Logical Star (with RF Synchronization)	Logical Star with TDMA (RF - Free)	Logical Star with FDMA (RF - Free)	Fully Meshed (Token Ring)
Cost	$(N+1)(C_{US}+C_{RF})$	$(N+1)C_{US}$	$(N+1)C_{US}$	$(N+1)C_{US}$
Update Rate (Cycles/Sec.)	$1/T_{max}$	$1/[2(N+1)T_{max}]$	$1/(2T_{max})$	$1/[N(N+1)T_{max}]$
Computational Complexity	Very Low (one way TOF estimation)	High (Matched Filter for each CDMA code)	High (Matched Filter for each frequency)	High (Matched Filter for each frequency)
Euclidean Distance Matrix *	(A)	(A)	(A)	(B)
Scalability	Highly scalable	Bounded by added complexity	Bounded by added complexity	Bounded by added complexity
Scalability Vs. Update Rate	Independent	Update Rate $\propto 1/N$	Independent	Update Rate $\propto 1/N^2$

* (A) Only the position of each mobile node, relative to the reference node, can be estimated.
(B) The position of all nodes, relative to each other, can be estimated

Table 2.5: Comparison between ultrasonic ranging network architectures.

2.4 Discussion

In light of this brief discussion, the following points are concluded:

- The best candidate for the localization problem in the indoor environments is the ultrasonic sensing technology because of its high accuracy and low cost. The computational complexity required for location estimation is also low, a feature that is very advantageous for the real-time

dynamic nature of multi-robot systems.

- The Ultra-Wide Band RF localization method is found to be a promising candidate as it has high accuracy and it overcomes the line-of-sight problem found in ultrasonic sensing based systems. However, at the time this survey was done, UWB was not approved by Industry Canada for civilian use.

- The mobile nodes of the three variants of the star topology networks discussed above have no secondary communication channels to broadcast their locations. This inherent limitation does not allow these networks to construct the inter-node Euclidean distance matrix, and hence their ability to carry out tightly coupled and coordinated tasks.

- By contrast, the fully meshed network architecture is capable of estimating all inter-node distances and angles, and hence the architecture is eminently suitable for tightly coupled and coordinated tasks. This capability is achieved at the cost of the localization cycle update rate which is of order $O(n^2)$, where n is the number of the network nodes.

Chapter 3
Round-Trip Ultrasonic Localization System

3.1 Introduction

This chapter details the analysis of the proposed round-trip ultrasonic positioning network. The proposed localization technique is applicable in any heterogeneous multi-robot system provided that each of them is equipped with the localization module that will be anatomized later in Chapter 4. There is no assumption made on either the control or dynamics structure of the robots. The round-trip fully meshed network architecture proposed in this work remedies many of the shortcomings cited in Chapter 2, namely; it allows for the construction of the inter-node Euclidian distance matrix, it does not require the highly complex matched filters required for constructing and interpreting the tokens of the fully meshed token ring network, it has a reasonably fast update rate. Furthermore, since the network does not require any real time synchronization mechanism hardware such as the RF or IR burst signals' arrangements, its hardware cost is low. Section 3.2 explains the localization network setup. Section 3.3 formulates the round-trip timing protocol equations for the communication dependent mode. Section 3.4 introduces the independent communication mode. Section 3.5 summarizes the principle of multidimensional scaling and the orthogonal Procrustes statistical method. Section 3.6 details the combined tessellation mode which enhances the performance and increases the accuracy of the localization network. Finally, Section 3.7 presents a MATLAB network simulation to compare the accuracy of each mode with respect to the position estimation method and the robot navigation speed. A discussion of the chapter follows.

3.2 Localization network setup

Figure 3.1 shows a sketch of the proposed network. With reference to the figure, the network consists of one reference node and N mobile robots (mobile nodes).

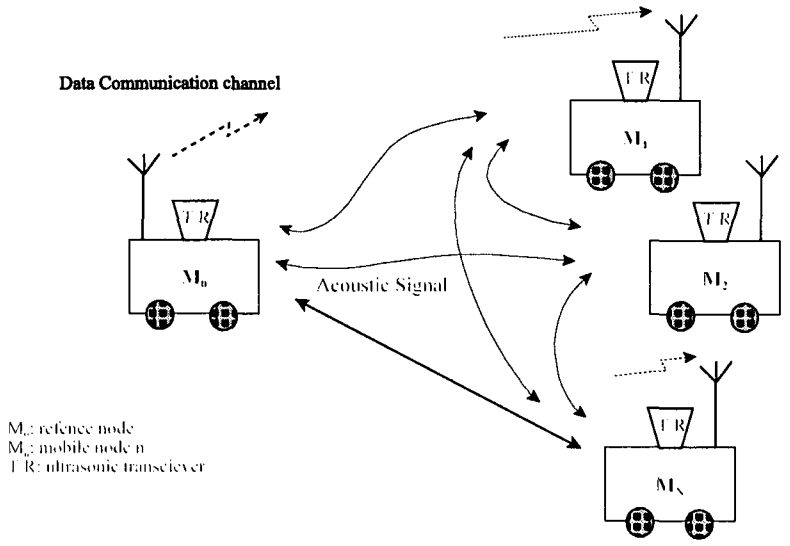


Figure 3.1: Round-trip fully meshed ultrasonic ranging network.

Each node has two communication channels;

- **Omnidirectional acoustic channel**; Each node may act as transmitter or receiver. In this case, the node may communicate bidirectionally by switching between the two roles. The omnidirectional feature is achieved by a parabolic reflector which will be detailed in Chapter 4.
- **RF Data communication channel**; This is a general purpose WI-FI wireless channel used exclusively for high-level data communication between the reference and the mobile nodes in non-real time mode.

The proposed localization network can be used in three modes: Communication dependent, communication independent, and the combined tessellation communication dependent. These modes are described in the following sections.

3.3 Communication dependent mode

In this mode, each node successively broadcasts an acoustic burst to all other nodes. Each receiving node log the time instants it receives the acoustic burst from a sender node and transmits these logged data via the RF data communication channel to the reference node on a nonpriority basis. The reference node processes the received data from all the nodes and uses it to estimate the Euclidean distance matrix.

3.3.1 Localization network parameters

The proposed localization network has mainly two parameters;

The blank time delay (Δ_b) - is defined as the time delay required to re-trigger an ultrasonic transceiver to act as transmitter or receiver.

The echo cancellation time delay (Δ_i) - is defined as the time delay applied at an end of an acoustic broadcast before the transmitting node M_i switches itself to act as receiver. This delay is necessary to guarantee that the node will not receive a reflected echo from mobile nodes within a critical range determined by a preset strength threshold. As a generic ultrasonic receiving sensory circuit is assumed, an echo is received at the ultrasonic transceiver if the signal power is higher than a preset threshold. This threshold is set just high enough to receive a directly transmitted signal from the designed maximum ranging distance of the network referred to in this context as D_{max}. Figure 3.2 portrays schematically the reflected ultrasonic signal decay.

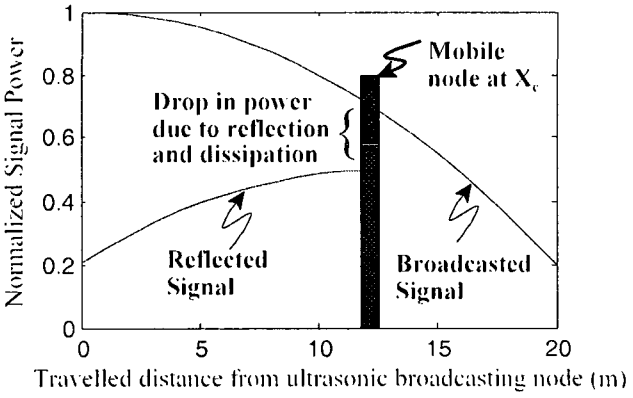


Figure 3.2: Normalized power distribution of Transmitted - Reflected ultrasonic signal.

With reference to the figure, an ultrasonic signal decays in relation to the square of the traveled distance d and is given by:

$$p(d) = 1 - a.d^2 \tag{3.1}$$

where a is the decay factor.

If the broadcasted signal bounces off an object, the drop in its power due to the reflection and dissipation is controlled by a factor μ. The critical distance from the broadcasting node D_c, at which the reflected signal may be received as a false echo is given by:

$$D_c = \sqrt{\frac{(1-\mu) - H}{(2-\mu) \cdot a}} \tag{3.2}$$

Where H is the threshold.

The echo cancellation time delay Δ_i can be approximated at runtime by:

$$\Delta_i = \begin{cases} \Delta_b & d > D_c \\ d_l / V_s & d \le D_c \end{cases} \quad (3.3)$$

Where d_l is the last estimated distance to nearest node, and V_s is the speed of sound.

3.3.2 Round-trip TOF equations

Each node in the network shown in Figure 3.1 has a timer with a microsecond resolution. All nodes' clocks need not to be synchronized at the beginning of each cycle. While this proposed architecture is valid for any ultrasonic ranging network of any size, a 4-node network will be exemplified for simplicity. Without loss of generality, consider a network of 3 mobile nodes M_1, M_2, M_3, and one reference node M_0. Figure 3.3 depicts a round trip ultrasonic ranging between two consecutive nodes M_0 and M_1. As shown in the figure, there are 3 states for any node in the network;

Transmit state - in which a node broadcasts an omnidirectional acoustic burst and is represented by a narrow pulse.

Receive state - in which the node is waiting to receive the acoustic burst from a transmitter node and its start is represented by a raising edge.

Idle state - in which the node is inactive and the start of this state is represented by a falling edge.

To simplify the estimation of the Euclidean distance matrix between the network nodes, the computational time delay is not taken into account during the localization process and would have to be considered during the system calibration. It is also assumed that there is no change in node positions during the localization cycle. If the nodes are traveling along trajectories, errors will arise since each node location is updated at a different time instance within the localization cycle. These errors will depend on the cycle update rate which is a function of the number of nodes, the trajectory,

and the route used to estimate the TOF discussed above. Further analysis of this error will be provided later in this chapter.

Initially, all mobile nodes start at different unsynchronized times and in receive state. The localization cycle is triggered when the reference node broadcasts an acoustic burst. After receiving the acoustic burst from the reference node, M_1 logs its timer value and switches to idle state for time interval Δ_b before it responds by broadcasting its burst to all other nodes. Then, M_1 sends the logged time instants to the reference node via the Wi-Fi channel for processing.

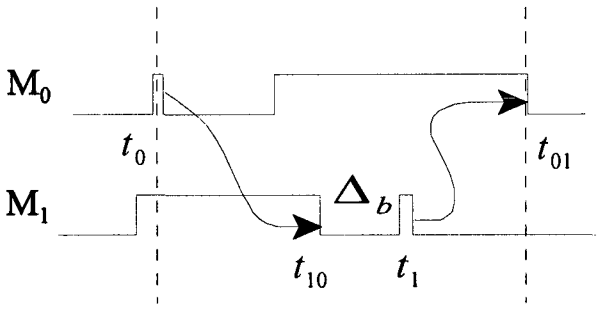


Figure 3.3: An example of round-trip TOF estimation between two consequent nodes.

As shown in Figure 3.3, the round trip TOF (T_{01} +T_{10}) is obtained by neglecting the change in position between time instants t_0 and t_1 as follows:

$$T_{01} = \frac{1}{2}\left(t_{01} - t_0 - \Delta_b\right) \tag{3.4}$$

and generally as per Figure 3.3, the TOF between any two consecutive nodes is expressed as follows:

$$T_{(n)(n-1)} = \frac{1}{2}\left(t_{(n-1)(n)} - t_n - \Delta_b\right); \quad n \geq 1 \tag{3.5}$$

Now considering the propagation of the signal between any two nodes twice removed in sequence from each other as in the case from M_0 to M_1, then from M_1 to M_2, and finally back again to M_0. The TOF between the two nodes in the example cited above can be inferred from either the above route or the round trip route between M_0 and M_2. Irrespective of the route used to calculate the TOF, the error due to nodes' trajectory depends on the displacement of the nodes that occur while they are waiting in sequence to respond to the reference node broadcast. It should be noted that M_2 does not broadcast its burst until it receives the burst signal of both M_0 and M_1.

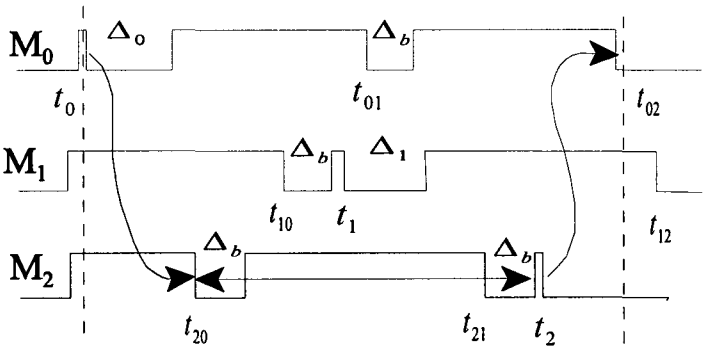


Figure 3.4: An example of round-trip TOF estimation between two nodes twice removed in order.

For the example shown in Figure 3.4, the TOF for the round trip route is given by:

$$T_{02} = \frac{1}{2}\left(t_{02} - t_0 - \left(t_2 - t_{20}\right)\right) \tag{3.6}$$

The error in localization would be due to averaging of the TOFs (t_{20} - t_0) and (t_{02} - t_2) if node's movement occurred in the time period (t_2 - t_{20}). However if the M_0, M_1, M_2, and M_0 route is considered, the TOF is given by:

$$T_{02} = t_{02} - t_0 - \Delta_b - T_{01} - T_{12} \tag{3.7}$$

In this case the localization is inferred from a single TOF (t_{02} - t_2), and is defined for the instance t_2 with respect to M_2 clock or at t_{02} with respect to M_0 clock. In general, the TOF between any two nodes twice removed from each other calculated using the round trip route, e.g., M_n and M_{n-2}, is given by:

$$T_{(n)(n-2)} = \frac{1}{2}\left(t_{(n-2)(n)} - t_{n-2} - \left(t_n - t_{(n)(n-2)}\right)\right); \quad n \geq 2 \tag{3.8}$$

Similarly, M_3 broadcasts its burst to all nodes after receiving the burst from M_0, M_1, and M_2. The round-trip TOF between two nodes thrice removed as in the case of M_3 and M_0 is exemplified using the third route as per Figure 3.5 by the propagation of the ultrasonic signal from M_0 to M_2, then from M_2 to M_3, and finally from M_3 back to M_0.

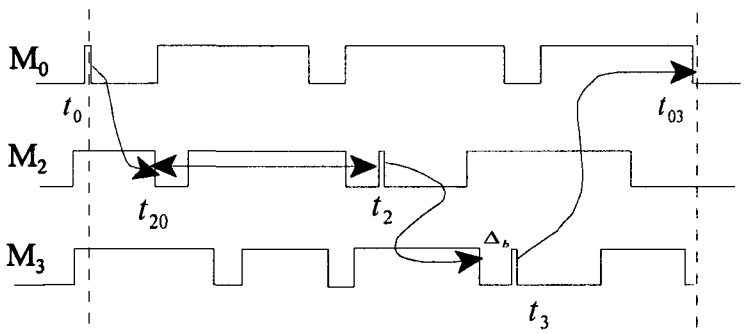


Figure 3.5: Example of round-trip TOF estimation between two nodes thrice removed in order.

The TOF in this case is expressed as follows:

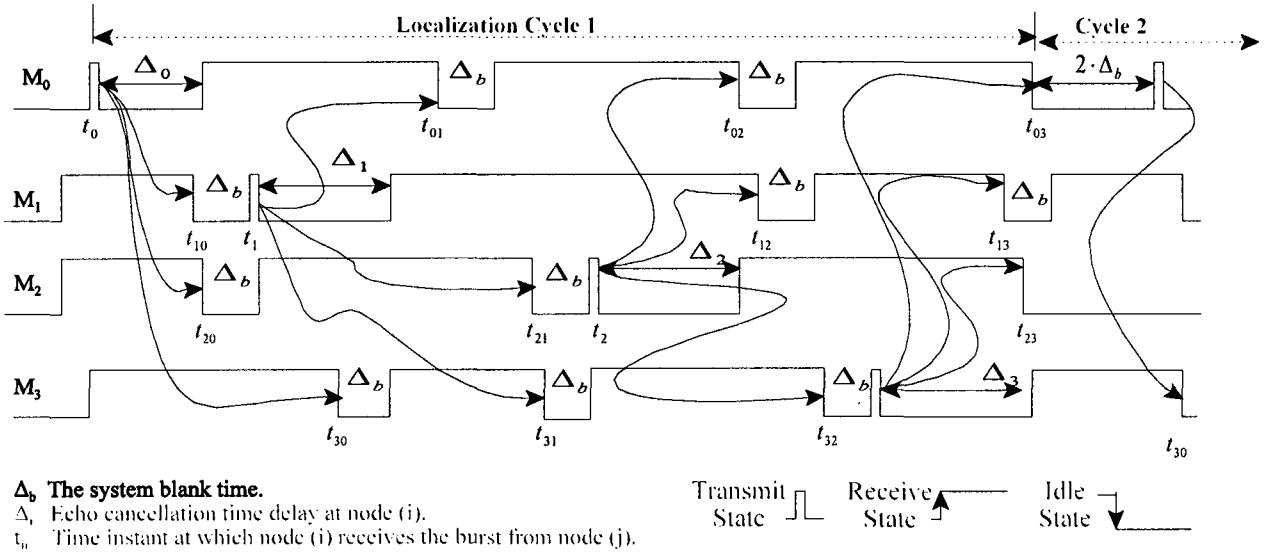
$$T_{03} = t_{03} - t_0 - \left(t_2 - t_{20}\right) - \Delta_b - T_{02} - T_{23} \tag{3.9}$$

In general, using this route, the TOF between any two nodes thrice removed from each other is given

by:

$$T_{(n-3)(n)} = t_{(n-3)(n)} - t_{n-3} - (t_{(n-1)} - t_{(n-1)(n-3)}) - \Delta_b - T_{(n-3)(n-1)} - T_{(n-1)(n)}; \quad n > 3 \text{ (3.10)}$$

Figure 3.6 depicts a single complete cycle of the proposed localization scheme and Table 3.1 summarizes all possible routes between nodes.



Δ_b **The system blank time.**
Δ_i Echo cancellation time delay at node (i).
t_{ij} Time instant at which node (i) receives the burst from node (j).

Figure 3.6: Round-trip one cycle TOF diagram.

	Route	Events Sequence*	TOF Equations
Once removed	1)	$M_0 \rightarrow M_1(t_{10}) \rightarrow M_0(t_{01})$ $M_{n-1} \rightarrow M_n(t_{(n)(n-1)}) \rightarrow M_{n-1}(t_{(n-1)(n)})$	T_{01}, T_{12}, T_{23} $T_{(n-1)(n)}$ for $n \geq 1$
twice removed	1)	$M_0 \rightarrow M_2(t_{20}) \rightarrow M_0(t_{02})$ $M_{n-2} \rightarrow M_n(t_{(n)(n-2)}) \rightarrow M_{n-2}(t_{(n-2)(n)})$	T_{02}, T_{13} $T_{(n-2)(n)}$ for $n \geq 2$
	2)	$M_0 \rightarrow M_1(t_{10}) \rightarrow M_2(t_{12}) \rightarrow M_0(t_{02})$ $M_{n-2} \rightarrow M_{n-1}(t_{(n-1)(n-2)}) \rightarrow M_n(t_{(n-1)(n-2)}) \rightarrow M_{n-2}(t_{(n-2)(n)})$	
thrice removed	1)	$M_0 \rightarrow M_3(t_{30}) \rightarrow M_0(t_{03})$	T_{03} $T_{(n-3)(n)}$ for $n \geq 3$
	2)	$M_0 \rightarrow M_1(t_{10}) \rightarrow M_3(t_{31}) \rightarrow M_0(t_{03})$	
	3)	$M_0 \rightarrow M_2(t_{20}) \rightarrow M_3(t_{32}) \rightarrow M_0(t_{03})$ $M_i \rightarrow M_n(t_{(n-1)(i)}) \rightarrow M_n(t_{(n)(n-1)}) \rightarrow M_i(t_{in})$	
	4)	$M_0 \rightarrow M_1(t_{10}) \rightarrow M_2(t_{21}) \rightarrow M_3(t_{32}) \rightarrow M_0(t_{03})$ $M_{n-3} \rightarrow M_{n-2}(t_{(n-2)(n-3)}) \rightarrow M_{n-1}(t_{(n-1)(n-2)}) \rightarrow M_n(t_{(n)(n-1)}) \rightarrow M_{n-3}(t_{(n-3)(n)})$	

*e.g. $M_0 \rightarrow M_1(t_{10})$ indicates that M_0 broadcasts the ultrasonic burst to the network and it is received at node M_1 at time t_{10} relative to node M_1

Table 3.1: Communication dependant round trip event sequences.

The Euclidean distance matrix after one localization cycle is constructed as follows:

$$D_{ij} = V_s \cdot T_{ij} \tag{3.11}$$

where V_s is the speed of sound given in Equation 2.1.

Using trigonometric relationships, the relative angles between the nodes can then be calculated. Dual solutions to the angles can be resolved by proximity comparisons to positions from the previous Euclidean distance matrix. It is important to note that system clocks on the different nodes do not need to be synchronized. This is an important feature of the proposed method. The cycle period of the proposed round-trip localization protocol can be calculated by considering the ultrasonic burst broadcasting sequence M_0 to M_1, then from M_1 to M_2, then from M_2 to M_3, and finally from M_3 back to M_0. This can be stated as follows;

$$P = (N+1)T_{\max} + N\Delta_b$$

As the number of nodes increases, the error in TOF estimation of mobile nodes, that are further away in order from the reference node, increases. The worst TOF accuracy occurs at the highest order node and the associated uncertainty time can be estimated as follows;

$$t_u = (N-2)T_{\max} + N\Delta_b \tag{3.12}$$

Where t_u is the uncertainty time. During this time, the system cannot provide any estimate about the node position and the error due to uncertainty in the robot's traveled distance is averaged in round-trip TOF estimation. As the uncertainty time increases, the individual robot position accuracy becomes more coupled to the robot's speed. The linear relation between robot's speed, position estimation accuracy, and the update rate is illustrated in Figure 3.7.

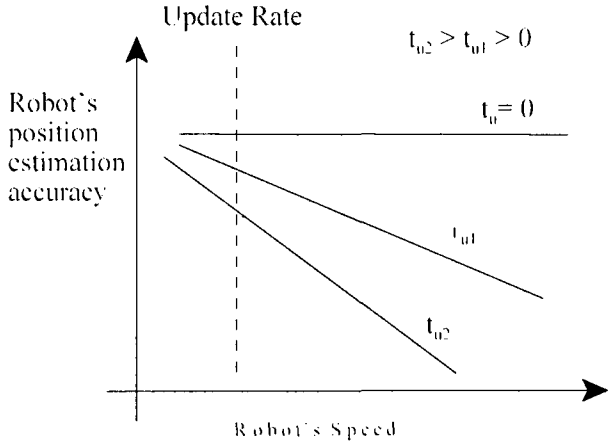


Figure 3.7: Relation between position estimation accuracy and robot's speed.

Figure 3.7 shows that at a given network update rate or number of nodes, the robot's position estimation accuracy degrades faster as the robot's speed increases. To enhance the accuracy of the system by reducing the involved uncertainty time in TOF estimation between nodes, the combined tessellation mode is developed as will be detailed in Section 3.6.

To enhance the performance of the network while ensuring crosstalk elimination, the

frequency division multiple access (FDMA) can be employed to reduce the blank time to zero. This however is obtained at the cost of added complexity in designing each frequency matching filter at the network nodes.

3.4 Communication independent mode

As discussed above, in this mode it is assumed that there is no explicit RF data communication between nodes. Only logged times at a node are used to infer the distance between nodes. As an example of the estimation of the distances at a node, consider the timing diagram of the four-node network shown in Figure 3.6. For the case of the node M_1, its localization cycle starts when it broadcasts a sonic signal at t_1. When it receives the response from M_2 at t_{12}, it can estimate the distance between them using Equation 3.5 with $n = 2$. When it receives the response from M_3 at t_{13}, it can only estimate the sum of the distances between M_2 and M_3 and between M_3 and M_1, but cannot discern the individual distances. Hence, this mode of operation is more suited for loosely coupled tasks like leader-follower ones. To perform such a task the sequence of burst transmission has to be reordered to suit the sequence in the convoy. For example, by considering the sequence shown in Figure 3.6, M_1 is able to track the movement of M_2. Similarly, M_2 can track the distance to M_3. This implies that M_2 is a leader to M_1 and M_3 is a leader to M_2. To obtain a different leader-follower sequence, the nodes' identifications have to be reordered. For instance, if M_1 is required to be the leader to M_3 and M_3 to be the leader to M_2. The nodes IDs can be changed as follows; M_1 should be changed to be M_3, M_3 is changed to be M_2, and M_2 is changed to be M_1.

3.5 System Multidimensional Scale Modeling

A 2-D system modeling method is required to provide the geometric configuration of nodes using the Euclidean distance matrix constructed from the TOF measurements gathered during a localization cycle. For that purpose, the metric multidimensional scaling (MDS) technique is considered as a good candidate. This technique, also known as the classic MDS, provides a geometric configuration of the objects in the smallest possible number of dimensions when the only known quantity is the distances between them. In [41] an analysis of this technique is presented. It primarily consists of decomposing a "dot-product" matrix B^* obtained from the Euclidean matrix using the Singular Value Decomposition (SVD). The coordinates' matrix of the objects is calculated from the eigenvectors and eigenvalues, V and A respectively. Given the Euclidean distance matrix D, the detailed steps of the technique are summarized in the following:

1) The dot-matrix B^* obtained by the double centering of D is obtained such that

$$B^* = \begin{bmatrix} b_{11}^* & b_{12}^* & \cdots & b_{1n}^* \\ b_{21}^* & b_{22}^* & \cdots & b_{2n}^* \\ \cdots & \cdots & \cdots & \cdots \\ b_{n1}^* & b_{n2}^* & \cdots & b_{nn}^* \end{bmatrix} \tag{3.13}$$

where

$$b_{ij}^* = \frac{-1}{2}\left[d_{ij} - \frac{1}{2}\sum_{j=1}^{n} d_{ij}^2 - \frac{1}{n}\sum_{i=1}^{n} d_{ij}^2 + \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}^2 \right]$$

2) The singular value decomposition of B^* is performed to give:

$$B = VAV^T \tag{3.14}$$

where V is the orthonormal matrix of the eigenvectors of B, and A is the diagonal matrix of its eigenvalues.

3) The coordinate matrix X is then obtained from the eigenvectors and eigenvalues as follows:

$$X = VA^{\frac{1}{2}} \tag{3.15}$$

It should be noted that errors in the estimated Euclidean distance matrix D affect the re-creation of the configuration of points by the classical MDS. Because the classical metric MDS has a closed-form solution, it can be performed efficiently on large matrices which supports the scalability of the proposed localization network.

The calculated points' coordinates are relative to the centroid of their geometrical shape. As the mobile nodes move, their formation centroid changes. It is then necessary to find the best translations, rotations, and reflections that map each of these coordinates into the same reference coordinate system. This is achieved by using the orthogonal Procrustes statistical method which is fully described in [42].

The Procrustes algorithm uses linear transformations to map one set of points onto another. The algorithm has been applied very successfully in diverse fields including psychology [42] and photogrammetry [43], where the alignment of related but different data sets is required. The orthogonal Procrustes problem aims to find an orthogonal transformation of a given matrix into another one in a way to minimize the sum of squares of the residual matrix. Given two matrices A

and B of size n×k, the solution to the orthogonal Procrustes problem is an orthogonal transformation T, such that the sum of squares of the residual matrix $E = AT - B$ is minimized. The least square solution must satisfy two conditions;

1) Trace$\{E^T E\}$ = Trace $\{(AT-B)^T(AT-B)\}$ = minimum

2) $T^T T = T\ T^T = 1$ (orthogonal transformation)

Both conditions can be combined into one function using Lagrangean multiplier as follows:

$$F = Trace\left\{E^T E\right\} + Trace\left\{L\left(T^T T - I\right)\right\}$$

Where L is a matrix of Lagrangean multipliers. Then to find the unknown matrix T that minimizes the residual error E entails the derivation of F with respect to T and setting the result to zero as follows:

$$\frac{\partial F}{\partial T} = 2A^T AT - 2A^T B + T(L + L^T) = 0$$

The solution to the above equation gives the orthogonal transformation T that maps the individual coordinates obtained into the reference coordinates as follows

$$T = VW^T$$

where V is the eigenvectors of $(A^T B)$, and

W is the eigenvectors of $(A^T B)(A^T B^T)$.

3.6 Combined tessellation mode

In this mode the TOF is used between successive nodes up to the point when a tessella can be formed and whose n -1 side lengths are known. In the case of the three mobile nodes and one reference node exemplified above, and as shown in Figure 3.8, the combined tessellation procedure works as follows:

1) Obtain the distances between the three vertices M_0, M_1, and M_2 by estimating them from the TOF as described in Section 3.3.

2) Use the multidimensional scaling localization (MDS) technique or trigonometric relationships combined with odometry data to establish the coordinates of M_1 and M_2 relative to the reference node M_0.

3) Establish the distance between M_3 and both M_1 and M_2 using the TOF between them as outlined in Section 3.3.

4) Again using the multidimensional scaling localization technique or trigonometric relationships, establish the coordinates of M_3 relative to M_1 and M_2.

If the network contains more nodes, the process continues along the lines outlined above. The worst case accuracy for localization between the moving node formation would be that associated with the displacement of intermediate nodes while the distance of the last side of the polygon (triangle in the example above) is being estimated. Consequently the associated time of uncertainty can be estimated as follows;

$$t_u = 2\Delta_b + T_{\max} \tag{3.16}$$

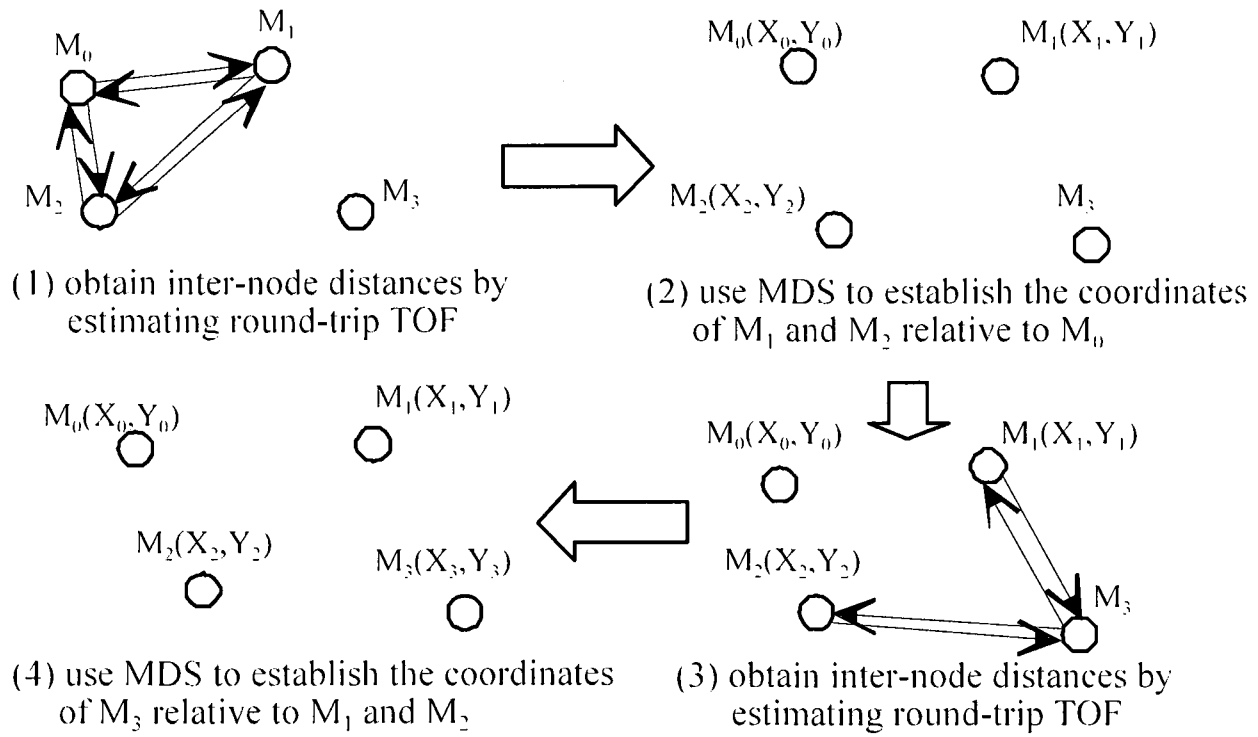


Figure 3.8: Communication dependant combined tessellation mode.

By comparing both Equations 3.12 and 3.16, it can be noted that the combined tessellation mode enhances the system accuracy by reducing the uncertainty time and consequently the accuracy of the node's position estimation increases. Therefore, the TOF estimation accuracy becomes independent of the number of nodes.

3.7 Simulation

Simulations of the four node mobile robot system operating under the proposed communication dependent and the combined tessellation modes has been carried out using the MATLAB code given in Appendix A.1 to evaluate the performance of these approaches. For all simulations, it is assumed that transceivers broadcast omnidirectional acoustic signals, sound speed is constant at 340 m/s, and a timer with microsecond resolution is used at each node in the network.

The network is assumed to consist of three heterogenous mobile robots and one fixed reference node. The robots move in a triad formation such that; the triad centroid virtual trajectory coincides with a predefined S-curve, the inter-robot distances should be constant, and the triad heading angle relative to the tangent of the trajectory is constant as shown in Figure 3.9. The move in a formation task is a highly coupled task that requires an accurate localization method. The localization method accuracy is measured by the perfection of the triad formation shape with regards the above-mentioned constraints in addition to individual robots' positions. This is achieved by tracking the triad centroid trajectory, triad head angle, the triad opposite side, the triad heading angle, and the robots' position estimation accuracy. The triad heading angle is the angle between the line connecting the triad head and the centroid trajectory tangent. The centroid moves at a constant speed along the curve. Figures 3.9 to 3.11 show the localization errors while the mobile robot formation tracks an S-curve virtual trajectory at speeds of 0.6, 1, and 1.5 m/sec respectively for a total traveling time of 12 sec. The initial Cartesian coordinates for the 3 robots are; (3 m, 3 m), (1 m, 4.5 m), and (2 m, 0.5 m) respectively relative to the reference node located at (0,0). The maximum allowed distance between any two nodes is 20 m and the initial Euclidean distance matrix is estimated by simply running the first localization cycle while the robots are at rest. Each figure shows six plots each with respect to the x-axis coordinate of the estimated centroid location. The top plot (a) in each figure shows the y-axis coordinates of the formation centroid. Plot (b), (c), and (d) in each figure compare the error in the centroid location, the triad head angle, and the triad side opposite to the head angle for the two simulated localization methods together with those for the ideal "Logical Star with RF Synchronization network". Plot (e) shows the error in the triad heading angle. Finally, the plot (f) shows the error in each robot's coordinate position.

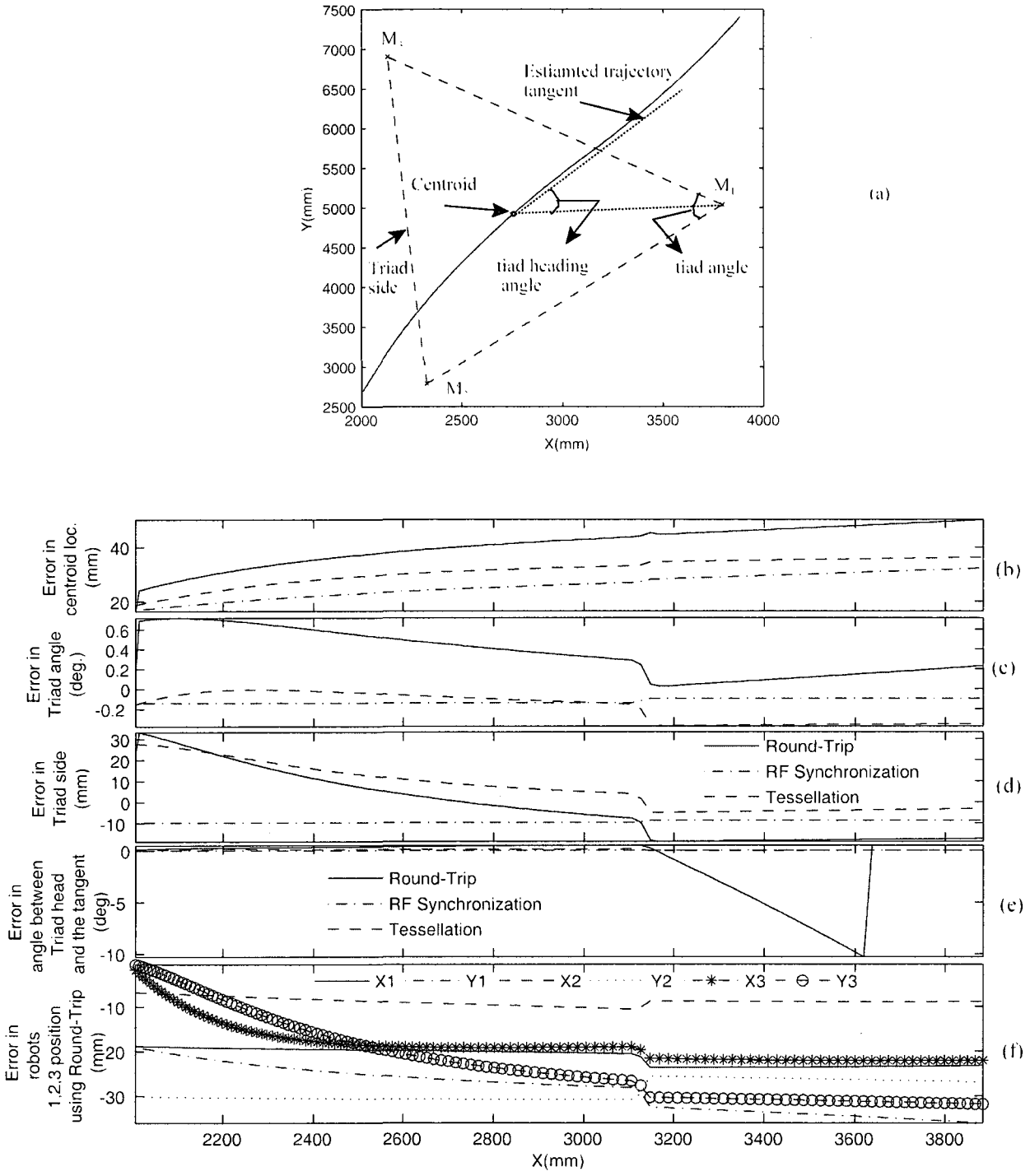


Figure 3.9: Communication dependent localization protocol performance with move in formation task (virtual speed of the triad formation = .6 m/sec).

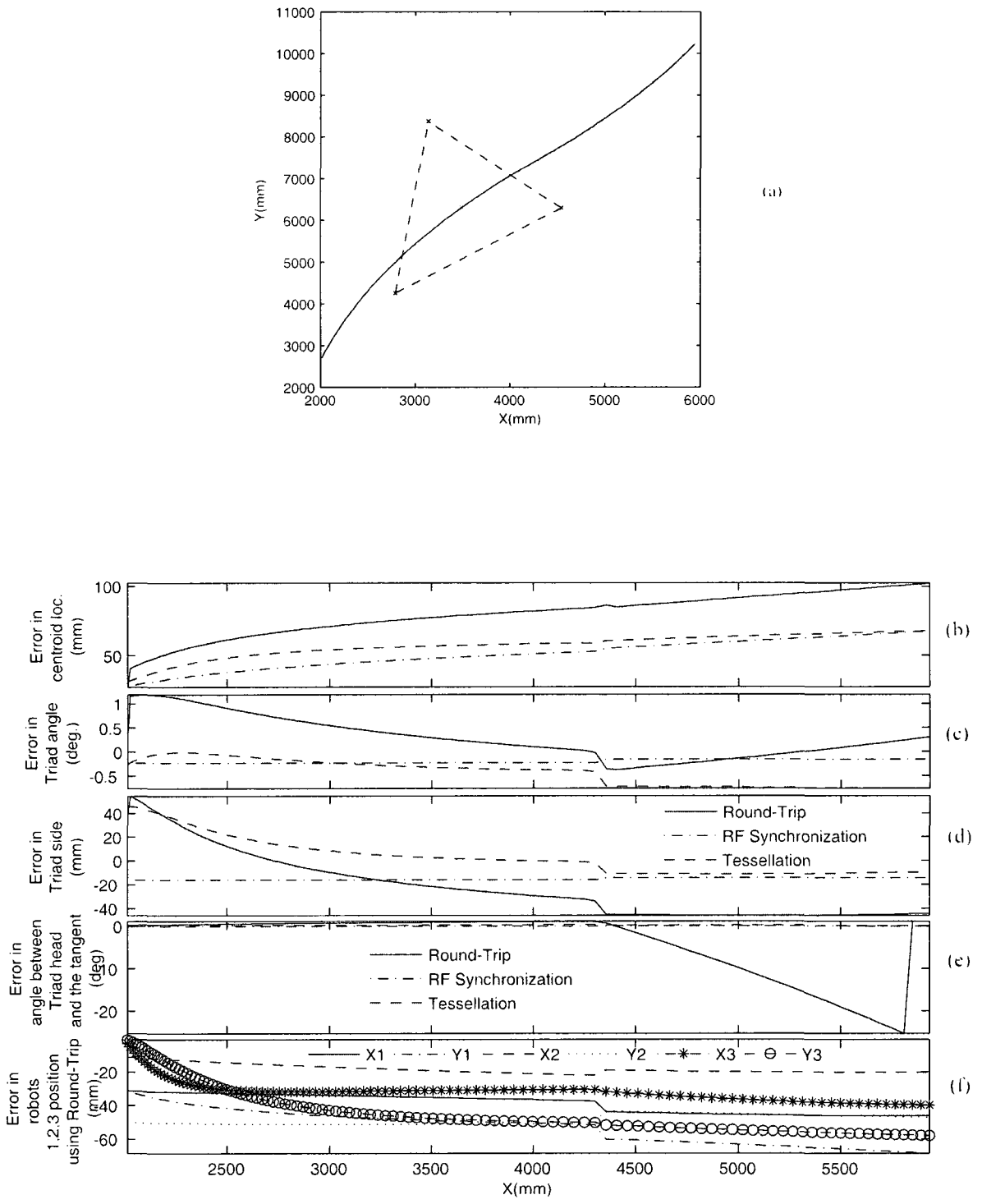


Figure 3.10: Communication dependent localization protocol performance with move in formation task (virtual speed of the triad formation = 1 m/sec).

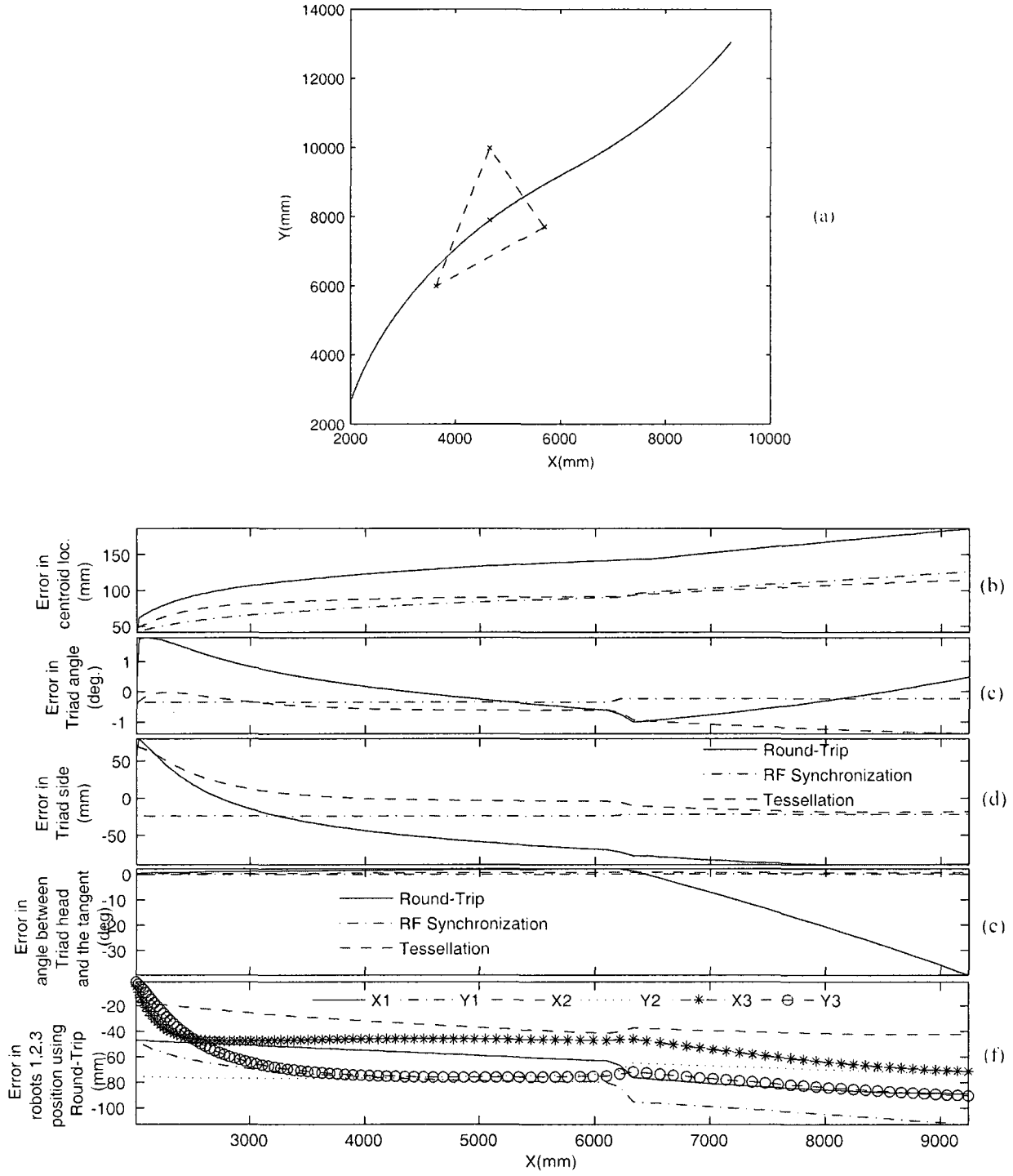


Figure 3.11: Communication dependent localization protocol performance with move in formation task (virtual speed of the triad formation = 1.5 m/sec).

With reference to Figures 3.9 to 3.11, the estimated and actual positioning trajectories in the top plot (a) appear to be coincident. This is primarily due to the scale used and the small magnitude in the error. Plot (b) in each figure shows the magnitude in the error of the centroid location as the formation navigates the track for the communication dependent, the combined tessellation mode, and the RF synchronized network. These plots show that the accuracy in centroid location estimation degrades as the nodes move away from the reference node for all cases. This is primarily due to the increase in the cycle period of the network as the TOF increases while the mobile nodes navigate away from the reference node. Therefore, the instances at which the node locations are estimated become further dislocated.

Comparing the centroid estimation error across Figures 3.9 to 3.11, it is observed that it increases linearly as the formation speed increases. This is to be expected as the TOF is linearly dependent on the inter-node distance, and the estimation cycle update rate is again linearly dependent on the TOF. The error due to the RF synchronized network is shown in the figures to be less than that for the proposed network. The difference, however, is slight and is counter balanced by the cost and complexity of the RF burst signaling hardware required for synchronization.

Both plot (c) and (d) in the figures show the error in tracking the mobile node formation through the triad head angle and its opposite side length, again for the three approaches. The plots show that the errors in tracking the formation due to the RF synchronized network is very low, and seem to be invariant as the formation moves away from the reference node. The formation tracking errors due to the two proposed approaches are higher than those of the RF synchronized network and vary as the distance between the reference node and the centroid formation change. Furthermore, the error due to the combined tessellation mode is lower than that due to the communication dependent mode. The variation in the formation tracking error is similar to that of the round-trip mode as it

inherits the same round-trip uncertainty errors in TOF estimation. The correction done in every third node absolute position by tessellation, mitigates the effect of heading away from the reference node. Although the combined tessellation mode enhances the overall system accuracy, additional computational complexity is required in performing the multidimensional scaling, transformation, and finally the tessellation.

Plot (e) in each figure shows the error in the triad heading angle. The triad heading angle should remain constant relative to the predetermined triad centroid trajectory. However, due to the error in centroid and triad head location estimation, the heading angle encounters an error. The errors in triad heading angle curves are directly coupled with the centroid and triad head locations' estimation accuracy for each method. Consequently, the RF synchronized approach achieves the best accuracy as it results in an error range which is almost zero. The accuracy in triad heading angle using the combined tessellation method results in an error that is slightly larger than that of the RF synchronized network method. Again, its curve seems to be coincident on that of the RF synchronized network because of the used scale. The communication dependent round-trip estimation method has the worst accuracy. This accuracy is highly affected by the robot's navigation speed due to the round-trip uncertainty errors in location estimation.

Finally, plot (f) shows the error in robots' coordinate position using the communication dependent method. The robots' coordinate position accuracy at a given robot's speed, degrades as the robots moves away from the reference node. This is because of the increase in localization cycle period which increases the error in position estimation due to the involved uncertainty. It can be noticed that the worst accuracy happens at the triad head node which is the furthest node from the reference node. The robots' coordinate position accuracy degrades as the navigation speed increase.

3.8 Discussion

This chapter proposes a fully meshed round-trip protocol with three possible modes of operation; communication dependent, communication independent, and combined tessellation modes. In both the communication dependent and communication independent modes, the protocol uses the round-trip ranging approach for all possible inter-node TOF estimations.

In the communication dependent mode the Euclidean distance matrix is calculated centrally at the reference node and conveyed to the mobile nodes, hence this approach is suitable for tightly coupled maneuvers. In the communication independent approach each node calculates its own Euclidean distance matrix by listening to the broadcasts from all other nodes. As such the matrix reference times may not be coordinated. Each node can maneuver independently to keep its pose in the formation, and hence this approach is suitable more for loosely coupled navigation.

The combined tessellation mode improves on the accuracy of the communication dependent approach. MATLAB simulations of the proposed approaches and of a standard RF synchronized network were carried out to evaluate the efficacy of the proposed approaches. The results show that errors due to the proposed approaches are slightly higher than those of the RF synchronized network. The higher accuracy of the RF synchronized network is offset by its higher complexity and cost.

Chapter 4
Localization System Architecture

4.1 Introduction

This chapter covers the engineering design and implementation of the proposed multi-robot localization system. The proposed system can be used in two modes: communication dependent mode by utilizing an explicit data communication channel to convey data between the mobile nodes and the reference node, and communication independent mode where nodes can only communicate implicitly via the acoustic channel. In both of these modes, the ultrasonic transmitter-receiver synchronization is achieved without using any additional hardware such as an RF or IR burst transmitter/receiver for real time coordination as detailed in Chapter 3. Two main aspects are covered; the localization module hardware and the high/low level software control architecture.

Section 4.2 describes the hardware components used in the design of the prototype localization module. The engineering design of the parabolic cone that ensures the omnidirectional broadcasting of the ultrasonic signal while avoiding ground reflections is given in detail. Section 4.3 describes the high and low-level software architecture adopted. This section also details the implementation of the two running operation modes of the proposed round-trip localization protocol. Section 4.4 ends the chapter with a discussion about possible improvements to the current design.

4.2 Hardware Architecture

A schematic of the proposed localization system infrastructure is given in Figure 4.1.

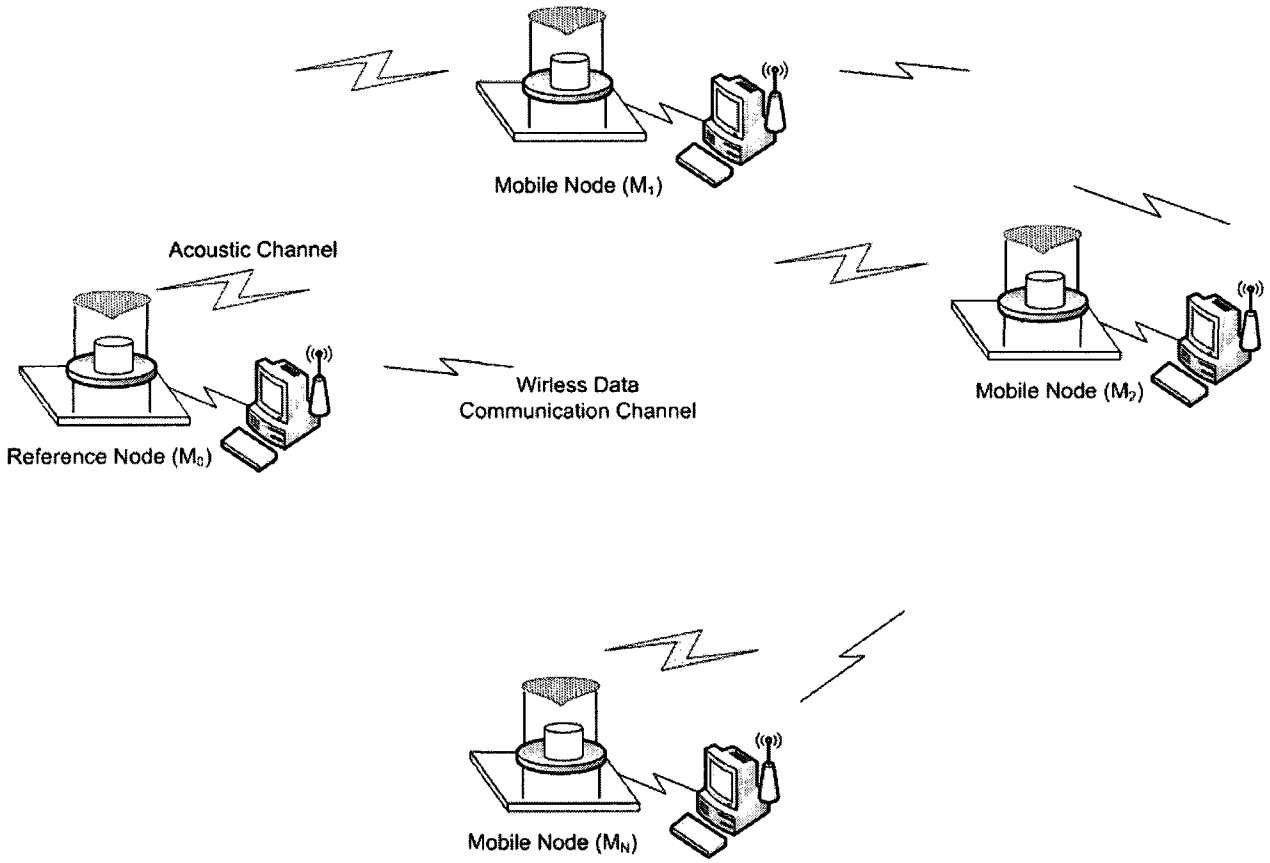


Figure 4.1: Ultrasonic localization network infrastructure.

As shown in the figure each node in the proposed system includes a wireless communication channel, an ultrasonic localization module, and a processor. The wireless communication channel is used only for data communication between the mobile nodes and the reference node.

Although, from the hardware perspective, the ultrasonic localization module used at any node is similar, the software is different as will be described later in this chapter. Figure 4.2 shows a photograph of the localization module hardware.

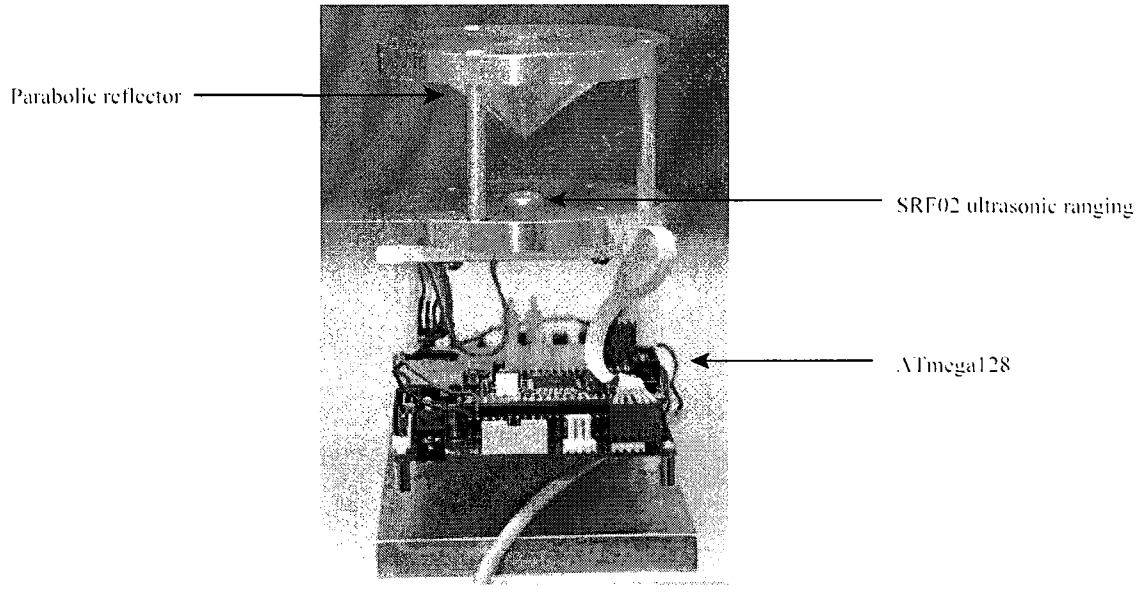


Figure 4.2: Localization module prototype.

The architecture of each node consists of four main components: a computer equipped with a wireless network adapter, an ATMEGA128 microcontroller development board, an SRF02 ultrasonic transceiver ranging module, and the parabolic cone. The parabolic cone is used as a means of broadcasting the ultrasonic wave in a thick disk pattern to achieve an omnidirectional coverage. A schematic of the node architecture can be seen in Figure 4.2. A detailed description of the used components follows.

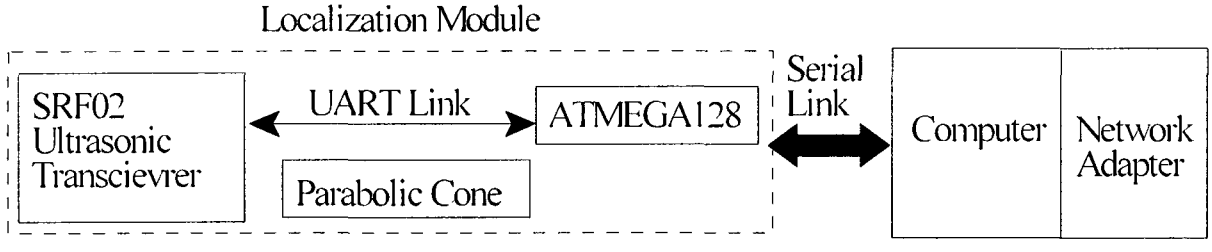


Figure 4.3: Block diagram of a network node architecture.

4.2.1 Ultrasonic transceiver driver

The ultrasonic transceiver driver consists of two components; an ET-AVR stamp development board for ATmega128 microcontroller and an SRF02 ultrasonic transceiver ranging module. The Atmega128 microcontroller is capable of performing a maximum throughput of 16 MIPS with a clock speed of 16 MHz. Most arithmetic and branch instructions are executed in a single cycle. Memory access is accomplished in 2 to 4 cycles. Notable features for this microcontroller include: 128K program space, 2K data memory, 4K non-volatile EPROM, 32 general purpose registers, two16 bit timers, two 8 bit timers, two UART ports, several internal interrupt sources including timer overflow and UART data receive, and JTAG compatibility for external debug control. The Atmega128 microcontroller is mainly employed in low-level timing control and event sequencing. The ET-AVR development board features a serial port that facilitates interfacing it to a computer.

The SRF02 ultrasonic ranging module is a single transducer ultrasonic rangefinder. Its features are: a UART interface that can be programmed to act either as transmitter or receiver, a transducer beam angle of 55°, and a maximum detected range of 11 m. The operating frequency of the ultrasonic transducer is 40 kHz. Its typical dissipation current is 4 mA at a supply voltage of 5V. The power of the emitted ultrasonic burst is about 100-150 mW. The SRF02 is used for either broadcasting an ultrasonic burst or to detect an incoming signal sent by another node. When the SRF02 receives an ultrasonic signal, it automatically returns two bytes ranging data via the UART to the microcontroller. Figure 4.3 shows a schematic diagram of a network node.

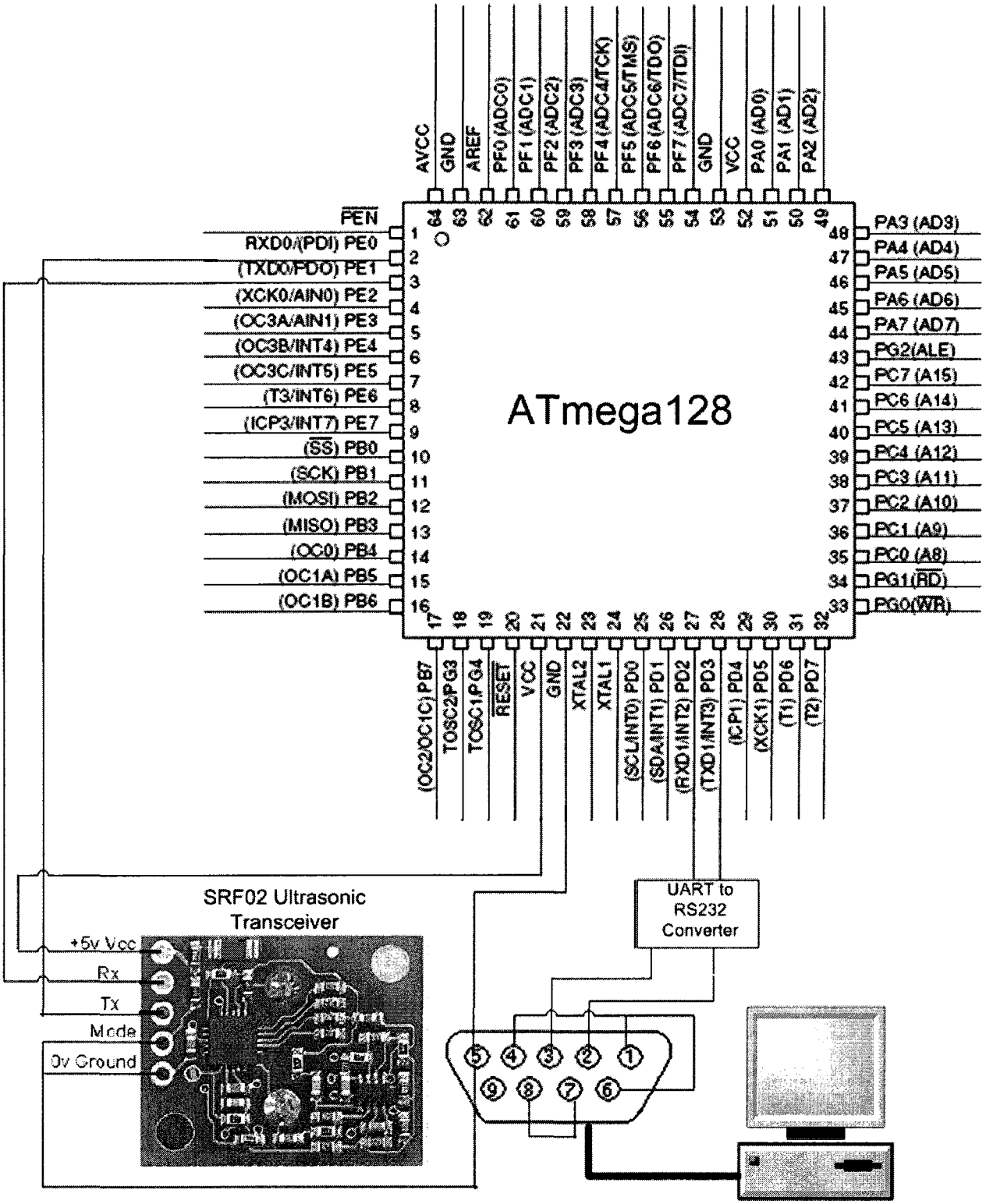
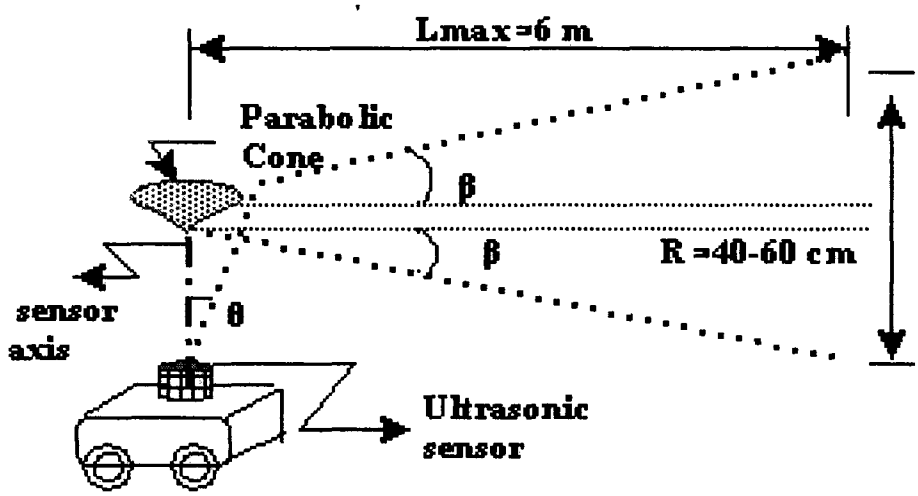


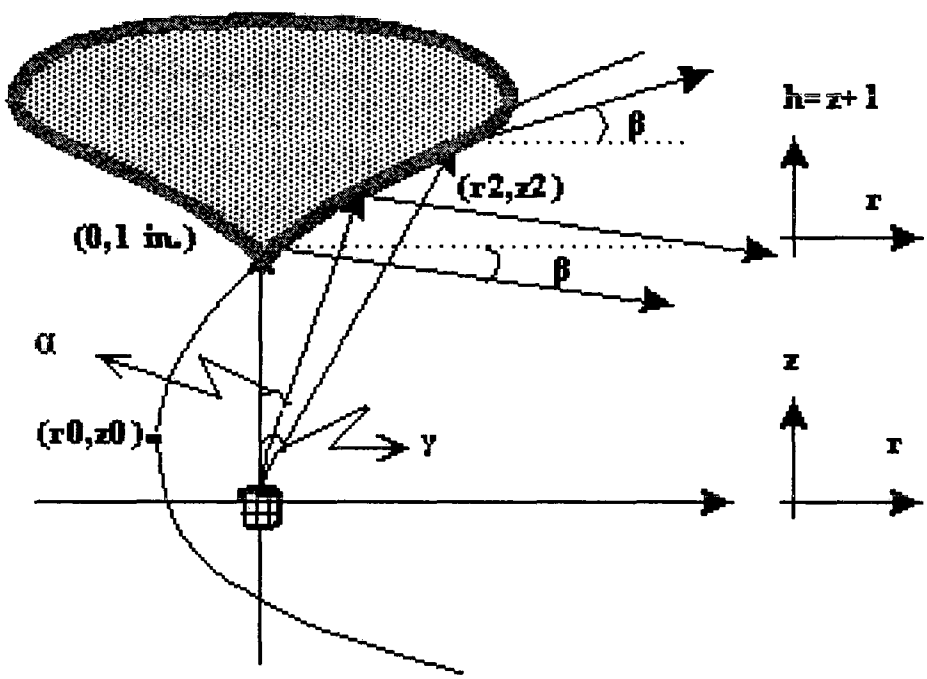
Figure 4.4: Localization module prototype schematic diagram

4.2.2 Parabolic cone

To obtain omnidirectionally spread of the ultrasonic waves emitted from the transducer, many researchers used a straight reflector cone as in [1]. Others suggest to use parabolic cones. While reflecting the ultrasonic wave from the focal point of the parabolic cone seems to be a good idea to achieve an omnidirectional broadcast in the form of a disk, the problem of ensuring that the broadcast from each robot impinges on the cones of all the other robots arises and is exasperated as the inter-robot distances increase. This is due to the increased chance of cones of different robots misalignment as they travel away from each other. Two solutions are envisaged to remedy the problem; (i) design a cone with sufficient height to ensure an omnidirectional disk of at least 40 to 60 cm in thickness, and (ii) place the ultrasonic sensor carefully away from the focal point to obtain a desired vertical reflection range as shown in Figure 4.4. With reference to the figure, it is desired that the ultrasonic beams emanating at an angle less or equal to α° would reflect downwards at an angle β° such that its bounce back from the ground would not reach another node 6 m away as shown in Figure 4.5 (a). Furthermore, the emitted ultrasonic beam between γ° - α° would be reflected within the bounds of an angle of $\pm\ \beta^\circ$ to obtain a disk spread as per Figure 4.5 (b). Considering the weight and the bulky size of the cone, the first solution is not practical. To implement the second solution, the relation between the beam width, intensity level and directivity of the ultrasonic lobe becomes critical and have to be taken into consideration as follows.



(a) Desired omnidirectional ultrasonic beam spread parameters.



(b) Parabolic reflector design parameters.

Figure 4.5: Parabolic reflector; (a) Desired omnidirectional ultrasonic beam spread parameters, (b) Parabolic reflector design parameters.

For a circular ultrasonic transducer source, the directional factor $H(\theta)$ is given by [40];

$$20\log(H(\theta)) = 20\log\left(\left|\frac{2J_1(x)}{x}\right|\right) = 10\log\left(\left|\frac{I_r(\theta)}{I_{ax}}\right|\right), \quad (4.1)$$

$J_1(x)$ is a Bessel's function and is given by;

$$J_1(x) = \frac{x}{2} - \frac{2x^3}{2(4^2)} + \frac{3x^5}{2(4^2)(6^2)} + \ldots,$$

$$x = ka\sin(\theta),$$

$$ka = \frac{2\pi af}{C}$$

$$C = C_0\sqrt{1 + \frac{T}{273}}$$

Where θ is the angle measured normal to the face of the ultrasonic transduce so that $\theta = 0$ on-axis, $I_r(\theta)$ is the emitted ultrasonic beam intensity at an angle θ measured relative to the sensor axis ($\theta =$ 0 on-axis), $I_{ax}(\theta)$ is the ultrasonic beam intensity on-axis ($\theta = 0$), a is the radius of the ultrasonic transducer, f is the operating frequency, C is the sound speed at temperature T, and C_0 is the sound speed at temperature 0°C.

The ultrasonic intensity of each beam, emitted from the sensor at angle θ measured relative to the axis of the cylindrical sensor decreases monotonically as θ increases, which follows from Equation

4.1.

The Beam Angle (γ) is defined as the total angle between the points at which the sound intensity of the main beam reduces to half its peak value on both sides of the on-axis peak, commonly known as 3 dB points. From the transducer's manufacturer data sheet; a = 5 mm, T = 25 °C, and f = 40Khz. By substituting these parameters in the speed of sound equation, the sound speed is found to be 335.55 m/s, and consequently the beam angle $\gamma = 27.1°$.

Choosing $\alpha = 14°$, the drop in the ultrasonic intensity is given by Equation 4.1 as follows:

$$20\log H(\theta) = 20\log\left|1 - \frac{(ka\sin\alpha)^2}{8}\right| = 20\log\left|1 - \frac{(3.745\sin\alpha)^2}{8}\right| \qquad (4.2)$$
$$= -0.92\ dB$$

This implies that the beam in the range γ - α has a considerable sound intensity to be reflected by the cone.

The general form of the parabola equation is given by;

$$(z + z_0)^2 = 4p(r + r_0)$$

Assuming that the tip of the parabolic reflector is to be fixed at a height of one inch from the center of the transducer surface, the tangent of the parabola curve at that point is derived as

$$\frac{dz}{dr} = \tan\left(45 - \frac{\beta}{2}\right) = \frac{4p}{2(1 + z_0)}$$

Where z_0 is given by,

$$z_0 = \frac{2p}{\tan\left(45 - \frac{\beta}{2}\right)} - 1 \tag{4.3}$$

Since point (0,1) lies on the parabola section, r_0 can be expressed as

$$r_0 = \frac{(1+z_0)^2}{4p} = \frac{\left(\frac{2p}{\tan(45-\frac{\beta}{2})}\right)^2}{4p} = \frac{p}{\left(\tan\left(45-\frac{\beta}{2}\right)\right)^2} \tag{4.4}$$

The tangent to the parabola curve at (r_2, z_2) can be expressed as

$$\frac{dz}{dr} = \tan\left(45 + \frac{\beta-\alpha}{2}\right) = \frac{2p}{(z_2+z_0)}$$

where

$$z_2 = 1 + 2p\left[\frac{1}{\tan\left(45+\frac{\beta-\alpha}{2}\right)} - \frac{1}{\tan\left(45-\frac{\beta}{2}\right)}\right] \tag{4.5}$$

and

$$r_2 = \frac{p}{\left(\tan\left(45+\frac{\beta-\alpha}{2}\right)\right)^2} - \frac{p}{\left(\tan\left(45-\frac{\beta}{2}\right)\right)^2} \tag{4.6}$$

whereas

$$\tan(\alpha) = \frac{r_2}{z_2} \tag{4.7}$$

Equations 4.3 to 4.7 provide all the design parameters for the parabola. For the purpose of experimental implementation, the following values are used $\beta = 2.5°$, $\alpha = 14°$. So, the parabola equation for the cone section with these values is given by;

$$(z+.64124)^2 = 3.14228(r+.85723)$$

For prototyping purposes, let $z = h + 1$, where $h = 0$ is at the tip of the cone. The final parabola equation is then given by:

$$(h+1.64124)^2 = 3.14228(r+.85723)$$

Where $0 \leq h \leq 1$ inch and $0 \leq r \leq 1.36$ inch.

It is very important to emphasize that the above detailed design of the parabolic reflector is an original contribution in this thesis.

Figure 4.6 shows two rays reflected from the top and bottom points of the designed cone. Since the reflection can occur at any point on the parabolic surface, the fluctuation in estimating the time of flight is expected. The maximum difference in path length (L) between the shown two extreme reflected ultrasonic rays can be calculated as follows;

$$L = \sqrt{(2H)^2 + r^2} - H \tag{4.8}$$

where r is the radius of the cone base and H is the height of the cone.

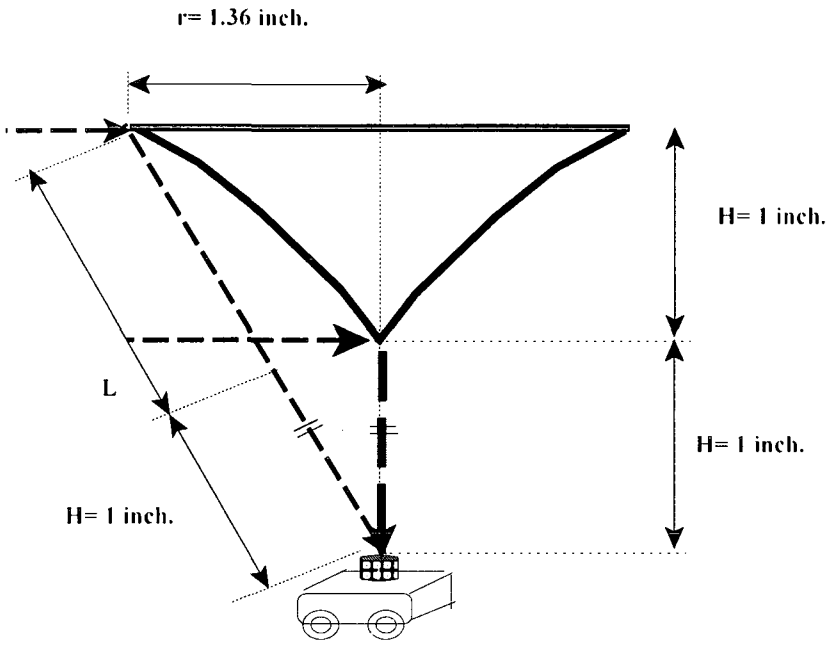


Figure 4.6: Parabolic reflector multipath reflection problem.

Substituting in Equation 4.8 with the chosen design parameters (r = 1.36 inch, H = 1 inch), results in L = 1.42 inch at each node.

4.3 Software Architecture

The implementation of the proposed round-trip protocol is abstracted into two distinct layers. The low-level timing layer is dedicated to event sequencing and time logging at each node while the high-level network layer is responsible for data communication and integrity. The software architecture for both the mobile nodes and the reference node is illustrated in Figure 4.7.

4.3.1 Low-level layer

As shown in Figure 4.7, the lower level consists of three basic modules; an event sequencer, an RS232 low-level driver interface, and an SRF02 ultrasonic transceiver driver. All modules are implemented and coded in Assembly language and then downloaded to the ATmega128 microcontroller. The low-level firmware is coded in a way to ease its deployment on any node by simply identifying two parameters, the node ID and the total number of nodes in the network. There is no difference between the developed low-level code for the reference node and any other mobile node. This low-level Assembly code can be found in Appendix A.

The RS232 low-level driver supports full-duplex serial communication between the ATmega128 and the computer. Due to the limitations of the ATmega128 UART port, as it does not support any serial communication protocol, the RS232 port is configured to work at a baud rate of 2400 bps. This low baud rate yields an error rate which is almost zero and increases the reliability of the serial communication channel. This channel is used primarily to send the logged accumulated timer instants to the computer's high-level layer which coordinates the wireless communication with the reference node. The 4-byte timer has a resolution of 4 μsec.

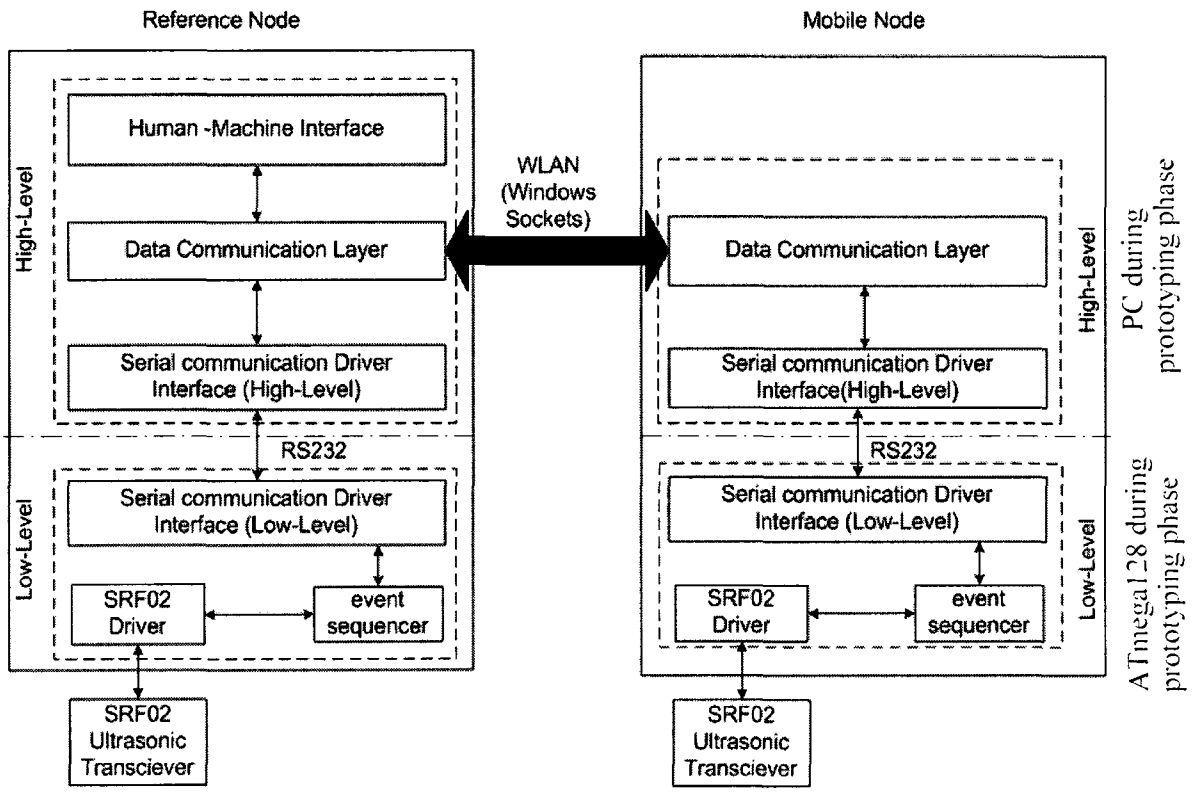


Figure 4.7: Software architecture.

The SRF02 ultrasonic transceiver driver has two functions. First, it interfaces the SRF02 to the ATmega128 via the UART port which is configured at 9600 bps baud rate with no parity check, and 2 stop bits. Second, it can be triggered to act either as a transmitter or a receiver. The receiving mode of SRF02 is limited to be no longer than 65 msec as after that time interval, it will be switched back to be in idle state as given in the manufacturer specifications sheet. If a burst is received, the SRF02 automatically returns two bytes that contain the ranging data to the ATmega128. Since the developed protocol requires a continuous receiving mode that may be longer than 65 msec, another part of the driver is implemented to overcome the SRF02 receiving mode limitation. It checks the returned two-byte value at the end of each ranging mode. If these two bytes are not zero, then a

received burst event is triggered and the accumulated timer value is logged. Otherwise, the SRF02 will be re-initiated to act as a receiver. Due to the possibility of missing a broadcasted ultrasonic burst from a sender node during the re-initialization phase of the SRF02 to act as receiver, another timer 's interrupt is used. The timer overflow interrupt is configured to be triggered periodically every time interval that is less than 65 msec to check the SRF02 status. In case there was no burst signal received, the re-initialization of the SRF02 receiving mode is triggered. Although, this timer's interrupt ensures the continuous SRF02 receiving mode, there is a statistically very remote possibility to miss the received ultrasonic burst at the same time when the reinitialization command is sent to the SRF02.

4.3.2 High-level layer

The high-level layer is developed using the embedded visual basic programming language. The software program can be found in Appendix A. As detailed in Chapter 3, in the communication dependant network mode, all of the time instants gathered at each node are sent to the reference node. Consequently, a high-level data communication layer is developed at each mobile node to send and receive data from the low-level layer to the reference node via Windows sockets. Exceptionally, at the reference node, the high-level communication layer exchanges data with the low-level layer via the high-level serial communication layer. However, the high-level programming was needed for the prototyping phase, the high-level software code can be implemented on the ATmega128 for an operational system.

At the high-level data communication layer, each mobile node is associated with a different user datagram protocol (UDP) Windows socket. Although the UDP sockets do not guarantee reliability, they ensure fast communication which matches the time sensitive nature of this

application. As all data at the end of each localization cycle is received asynchronously from different nodes, a high-level overflow timer event is triggered periodically every time interval that depends on the network latency to check the receiving of all gathered data. Then, the estimation of the Euclidean distance matrix is performed according to Equation 3.11. The new calculated time delays are estimated afterwards and are sent first to the mobile nodes via the data communication channel. A short time delay is introduced before sending the time delay value to the reference node low-level layer via the serial port to ensure that all the other mobile nodes are already triggered to act as receivers before starting a new localization cycle.

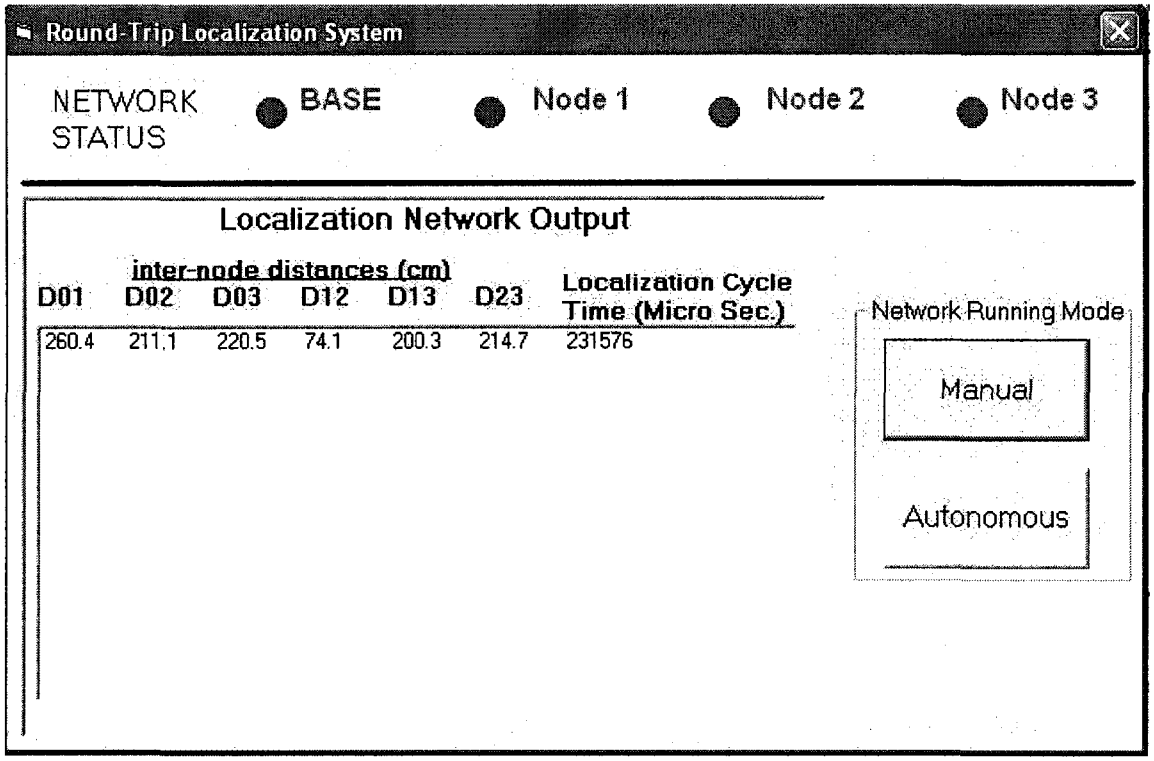


Figure 4.8: Human-machine interface.

A human-machine interface is developed to track the output of the system. The graphical interface is divided into three main areas as shown in Figure 4.8: a network status indicator, a

network operation mode selection, and a localization network output.

Network Status Indicator - Is used to indicate the status of the high-level communication network layer. Each indicator becomes solid green when a bidirectional socket communication is established between a mobile node and the reference node.

Network Operation Mode Selector - The system can be initiated to operate in two modes; manual or autonomous. In the manual mode, the system executes one localization cycle by pressing the "Manual" button. The manual running mode is used in debugging and calibration of the system as will be detailed in Chapter 5. In the autonomous mode, the system executes the localization protocol cyclically without the intervention of the user.

Localization Network Output - This dialogue shows the inter-node distances in cm and the localization cycle time period in μsec at the end of each localization cycle.

4.4 Discussion

The overall localization system described is the first prototype developed for the proposed localization network. Although the overall hardware and software modules offer relatively good performance as it will be shown in Chapter 5, some improvements could be made in a future release.

Ultrasonic Transceiver Ranging Circuit - The currently used digitally-controlled SRF02 module has a critical limitation of a maximum 65 msec receiving mode operation. That is, each time the sensor is triggered to act as a receiver, it returns to idle status if no signal is received within 65 msec time frame. This does not suit the asynchronous behavior of this application. An alternative would be to use an ultrasonic transducer that can be driven directly by the ATmega128 microcontroller. This alternative would also offer the flexibility to control the maximum threshold

applied to detect a received burst and consequently reduce the time delay as detailed in Section 3.3.1.

High-Level Computer Network Layer - The network layer and high-level processing are currently employed for data communication and Euclidean distance matrix estimation on the reference node. This influences the internal processing time delay at each node which also increases the localization cycle time period and consequently reduces the distance matrix update rate. This can be enhanced by using a serial communication adapter that can be interfaced directly to the ATmega128. All the Euclidean distance matrix calculations are simple enough to be carried out efficiently by the ATmega128. The fine resolution and high accuracy that characterize the proposed method would not be affected by these enhancements.

Chapter 5
Experimental Results

5.1 Introduction

In this chapter the analysis of the results of the experiments carried out to test the accuracy of the proposed localization protocol with respect to the hardware resolution and the timing coordination software. The localization method proposed in Chapter 3 was tested using a network of localization modules whose design is detailed in Chapter 4.

Section 5.2 presents the experimental setup including the environment description, the network architecture, and the methodology adopted to conduct the experiments. Section 5.3 details the system calibration procedure. The core of the chapter, Section 5.4, presents two comprehensive tests that were conducted to evaluate the efficiency of the proposed localization method. Finally, Section 5.6 discusses the results with a particular emphasis on error analysis.

5.2 Experimental Setup

The network architecture, shown in Figure 5.1, is employed for operating the experimental setup. The network consists of four localization nodes which are assumed to be three mobile nodes and one fixed reference node. Each node consists of one localization module interfaced to a computer via a serial bus. The hardware localization module of each node is shown in Figure 4.2. All four computers are connected to an ethernet network (i.e., the Electrical Engineering department network) which is adopted to provide a data communication channel between each mobile node and the reference node. This can be substituted by any wireless

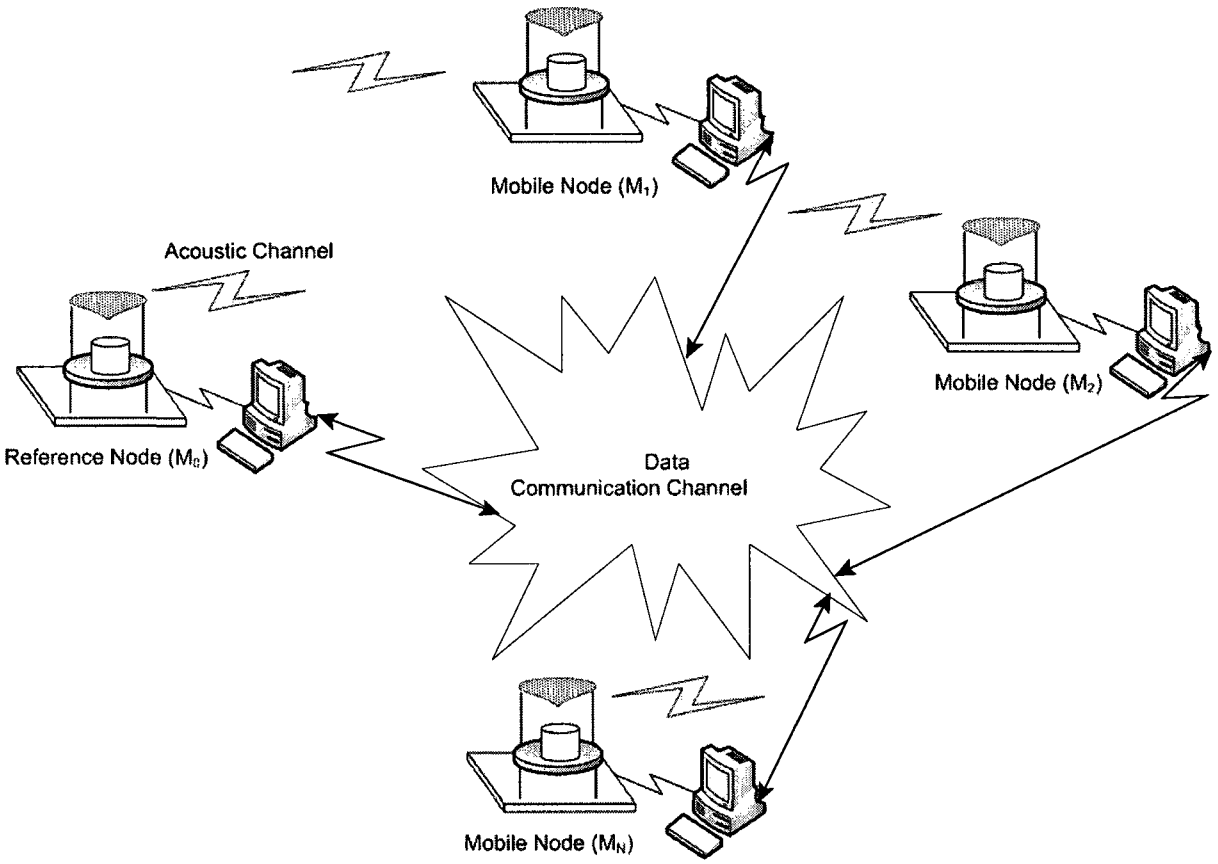


Figure 5.1: The network architecture for the experimental setup.

communication channel if needed. The lab area where the localization experiments were conducted is 4m x 4m. The room temperature is assumed to be constant and the speed of sound calibration is carried out as will be detailed in Section 5.3.

5.3 System Calibration

There are three system parameters that should be determined as explained in Chapter 3; the speed of sound, the blank time, and the time delay due to the node's internal processing. All of these parameters are measured only once when deploying the system in a new environment. Both, the speed of sound and blank time calibrations were conducted offline, whereas the node's internal processing time delay parameter was estimated online. The procedures for measuring these

parameters are outlined below.

5.3.1 Speed of Sound

Within a building, the ambient conditions such as temperature and humidity are subject to temporal and spatial variations. Temporal variations take place at different time scales, e.g. the time during the day or season of the year. Spatial variations occur due to heaters or air conditioners, or direct sunlight in certain parts of a room. According to the utilized technology, these variations can have an impact on the accuracy of the localization system. The relation between speed of sound and the ambient temperature is linear as depicted in Equation 2.1. In the proposed system, it is assumed that no spatial variation in ambient temperature existed since the indoor environment is air-conditioned. Thus, only a simple system calibration is required to determine the speed of sound to accommodate the temporal variations. This can be done when the localization system is either online or offline.

In an online system calibration, temperature measurements can be taken through an onboard temperature sensor of the robots at the beginning of each localization cycle. This ensures high accuracy and increases the system's autonomy. On the other hand, it increases the hardware cost required at each node.

The offline system calibration is performed by facing the SRF02 module to a reflector that is perpendicular to the center of the SRF02 at a distance larger than 25 cm. When ranging is completed, the SRF02 returns to the microcontroller two bytes representing the range result in centimeters. Then, by firing the SRF02 in real ranging mode using 0x53 hexadecimal command. The returned two bytes are representing the same range result, obtained by, but in micro-seconds. The speed of sound can be calculated as follows:

$$V_s = \frac{r_{cm} \times 10^{-2}}{r_{\mu \sec} \times 10^{-6}} \tag{5.1}$$

where r_{cm} is the range result in cm and $r_{\mu sec}$ is the range result in microseconds.

Although the offline speed of sound calibration is not as accurate as the online method, it has an advantage that no extra hardware is needed. This keeps the hardware cost at a minimum.

5.3.2 Blank Time

As mentioned in Section 3.3.1, the blank time value should be enough to re-initiate an ultrasonic transceiver to act as a transmitter or receiver. However, due to the limitation of the SRF02 transceiver; the acoustic threshold level cannot be changed by the user. As detailed in Chapter 3, the variable threshold is necessary in order to filter out the reflected signals from surrounding obstacles and other robots. Consequently, the blank time is re-defined here to be the time delay required at any node in order not to receive a false echo from the surrounding objects. The blank time needs to be calibrated such that a signal reflected from the farthest objects from the broadcasting node fades out.

This calibration is carried out simply by measuring the longest distance (D_{max}) between any two points in the environment. Then the blank time (Δ_b)can be calculated as

$$\Delta_b = \frac{2 \cdot D_{\max}}{V_s} \tag{5.2}$$

Since the experimental environment was 4mx4m, a blank time of 35 msec was found to be sufficient for the transmitted ultrasonic signal to dissipate without being received by any node. It is important to emphasize that this large blank time value was due to hardware limitations. However,

using an ultrasonic circuit with a variable threshold capability, a blank time that is long enough to reconfigure the transceiver circuit to act as transmitter or receiver from an idle state is estimated to be in a range of a few microseconds.

5.3.3 Internal Node Processing Delay

The internal node delay calibration needs to be carried out while the system is online. This time depends on the delay in the low-level interrupts handling process and the execution time of the low-level code. The low-level interrupt handling delay was measured to be in the range of zero to 16 microseconds.

The online calibration was carried out by running the proposed round-trip human-machine interface detailed in Section 4.3.2 in the debug mode. As detailed in Section 3.3.2, there are six TOF estimations representing all possible inter-node distances during one localization cycle. For each TOF equation, a calibration is performed by subtracting a time delay to minimize the error in the corresponding inter-node distance. This step is repeated at all nodes.

5.4 Experiments

This section presents the two separate experiments conducted to evaluate the accuracy of the proposed localization protocol with regard to the timing software and the node's hardware. For that purpose, one node is kept at a fixed location and three other nodes were moved manually. The nodes' locations at discrete points are recorded along with the four-node formation centroid. The human-machine interface manual mode described in Section 4.3.2 is used to track the system output at the end of each localization cycle. The communication dependent mode TOF equations detailed in Section 3.3.2 were used to determine the inter-node TOF estimation.

For each of the two experiments, the blank time was kept constant (35 msec). The high-level timer that was discussed in Section 4.3.2 had to be triggered every 80 msec due to the latency of the ethernet data network. Six localization cycles were run while the nodes were at a standstill positions and the average of all estimations was taken to be the system's output. Then, the three mobile nodes were moved manually to a different position. The estimated Euclidean distance matrix at each standstill position on the trajectory is fed into the multidimensional scale modeling algorithm. The resultant set of estimated coordinates of each mobile node is orthogonally transformed such that all the estimated coordinates are referred to the coordinate system of the first point on the trajectory of movement as described earlier in Section 3.5. The actual inter-node distances were measured at each position and fed into the MDS. Then, the actual nodes' coordinates were referred to the coordinate system of the first point on the trajectory by applying the Procrustes orthogonal transformation. The actual and estimated data are compared and the error in each node position is plotted, as shown in Figures 5.4-5.6 for the first experiment and Figures 5.8-5.10 for the second experiment. Figure 5.2 shows a photograph of the experimental setup.

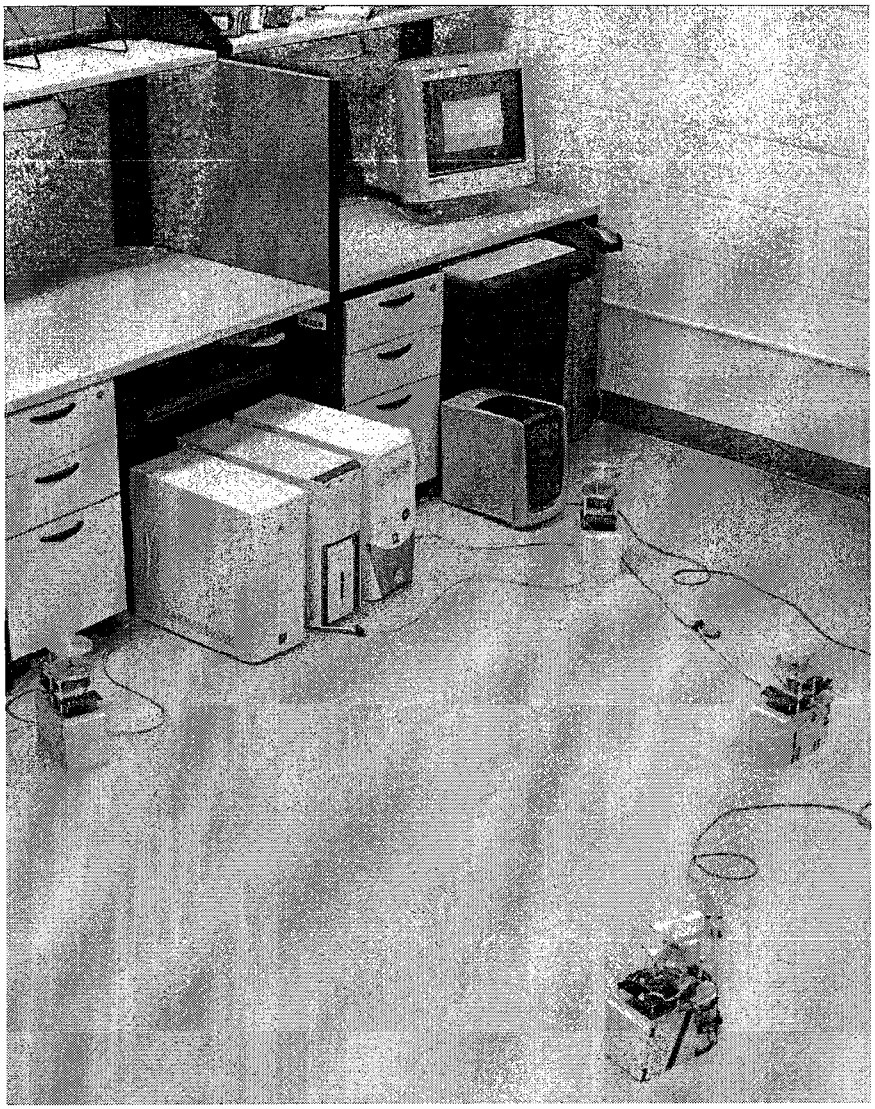


Figure 5.2: The experimental setup.

5.4.1 Experiment #1

Figure 5.3 shows a window in the planar domain whose origin is the location of the reference node. The initial coordinates of the three mobile nodes along with their centroid are shown by circles. Five discrete locations of the centroid along its motion trajectory, along with the time instances at these locations are also shown by squares. The total traveling time is 1.4 sec. The three

circular points show the initial position (at t = 0) of the triad formation (M_1, M_2, and M_3).

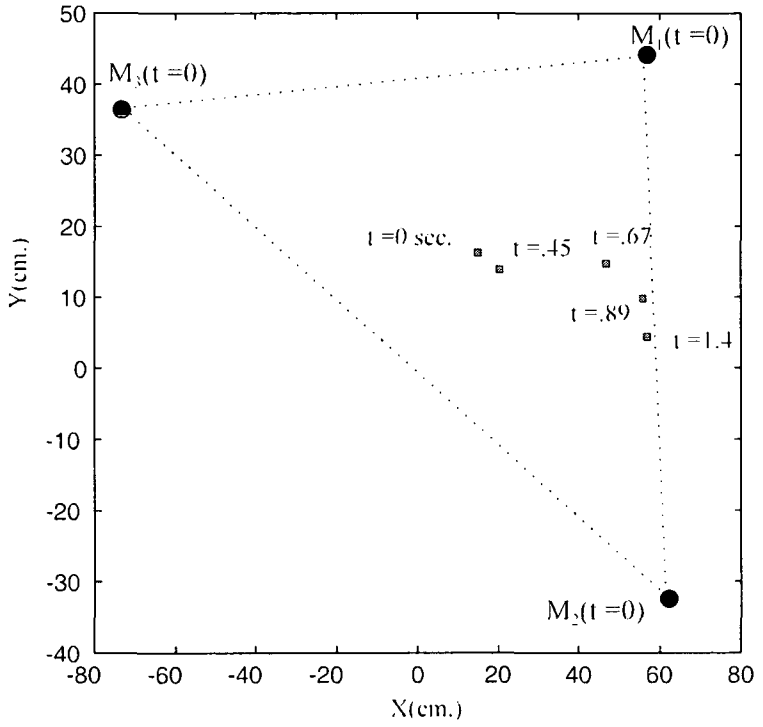


Figure 5.3: Estimated centroid trajectory - Experiment #1.

The results of Experiment #1 are presented in Figures 5.4 to 5.6. The figures show the estimated polar coordinates for the three nodes corresponding to the time instances indicated beside each formation centroid marked on Figure 5.3. Among the three mobile nodes, the longest traveled distance from the base is 111 cm, and is attributed to M_1. This can be calculated from the polar coordinates shown in Figures 5.4 to 5.6. The figures also show that the range of the radial coordinate error in the system is ± 4 cm and the range of the angular coordinate error is ± 8°.

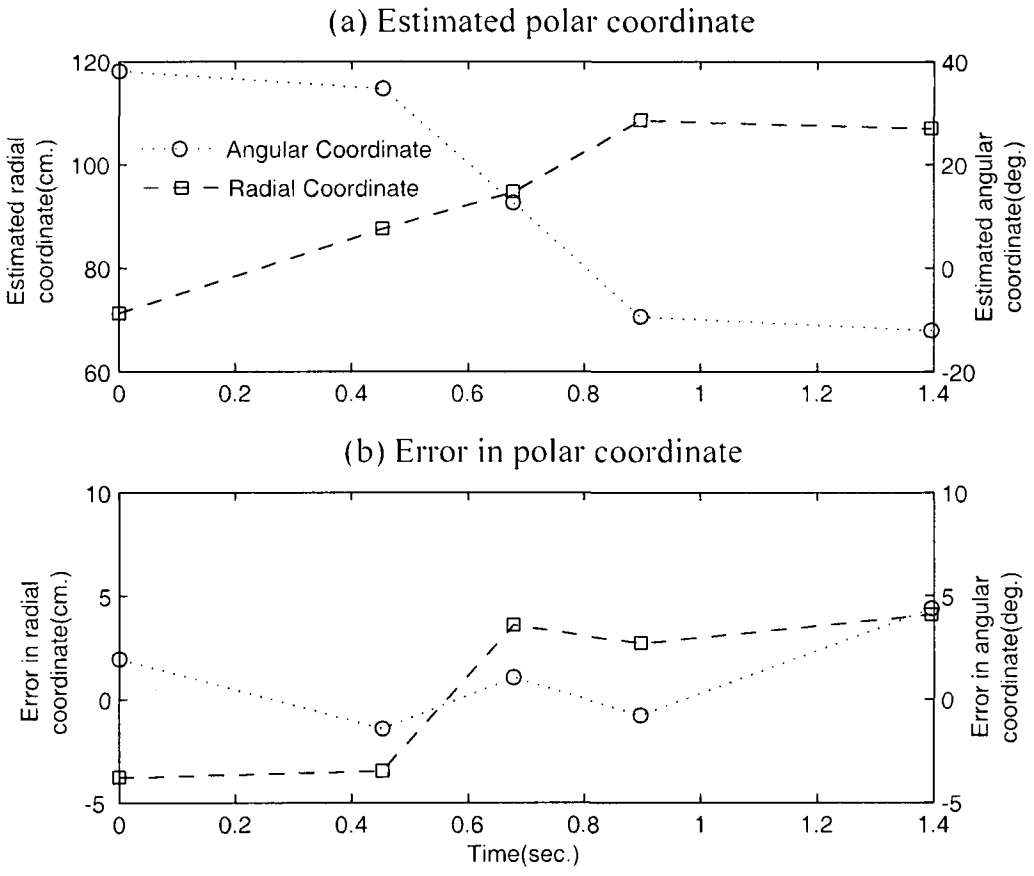


Figure 5.4: Robot #1 - Experiment #1; (a) Estimated polar radial coordinate (radial, angular), (b) Error in estimated polar coordinate (radial, angular).

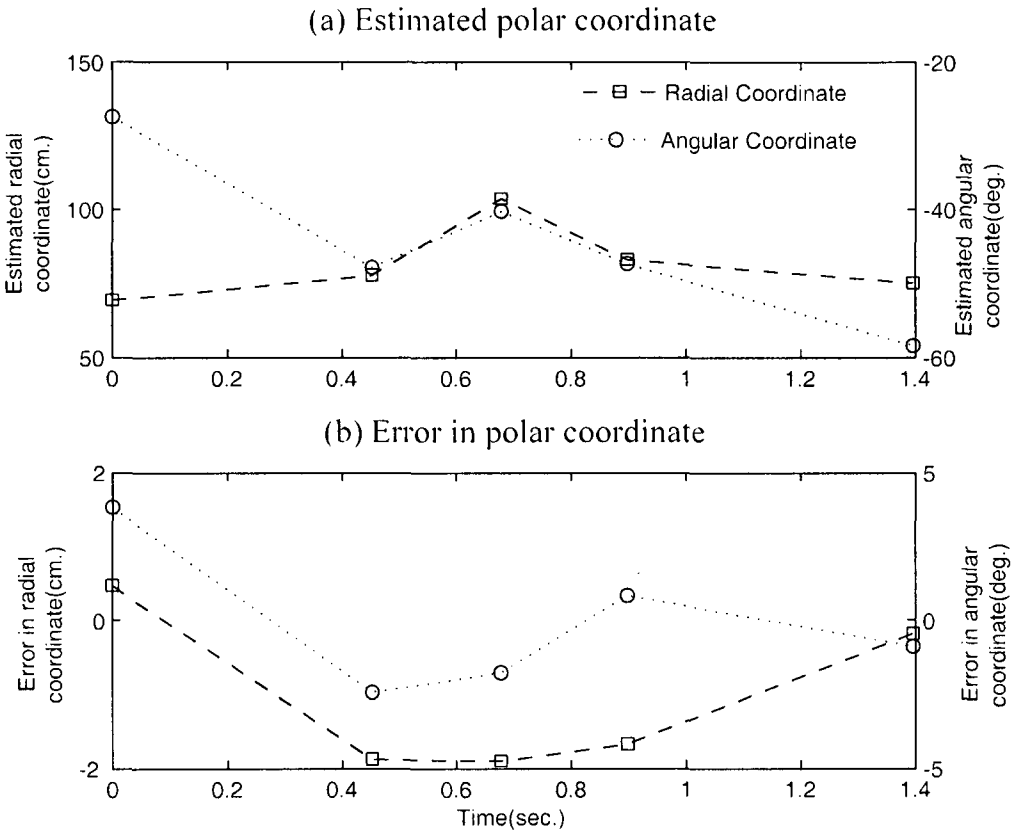


Figure 5.5: Robot #2 - Experiment #1; (a) Estimated polar radial coordinate (radial, angular), (b) Error in estimated polar coordinate (radial, angular).

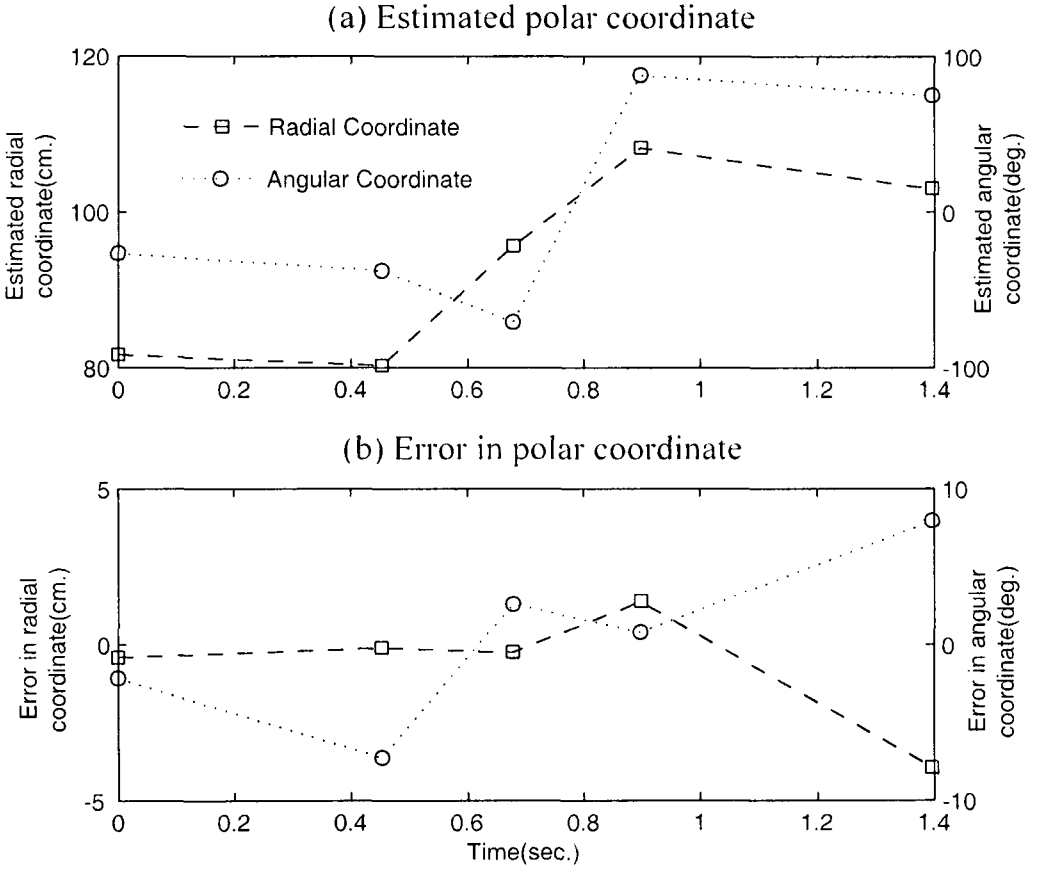


Figure 5.6: Robot #3 - Experiment #1; (a) Estimated polar radial coordinate (radial, angular), (b) Error in estimated polar coordinate (radial, angular).

With regard the hardware resolution, the reflection from the parabolic cone surface that acts as a omnidirectional reflector and is designed in Section 4.2.2 has a substantial effect on the achieved accuracy. Since the actual distance between any two nodes is measured between the center points of their cones, the propagation of ultrasonic waves due to the reflection from any point located within the route length (L) shown in Figure 4.6, yields an error in the TOF estimation. The maximum route length (L) was calculated in Section 4.2.2 to be 3.6 cm (1.42 inches). This factor results in a constant error and it is hard to compensate for since the reflection point can be anywhere on the

parabolic surface curvature within L.

Although the low-level timing software was calibrated, the fluctuations in interrupt handling processing time has an effect on the accuracy. These fluctuations range from 0 to 16 microseconds. Therefore, it results in another bias in the error which can be as high as 0.5 cm.

5.4.2 Experiment #2

The variation in low-level interrupts handling time is enhanced by the rearrangement of interrupts priorities at the low-level coding. This reduced it to be almost in the range of 4 microseconds before running experiment 2. Figure 5.7 shows a window in the planar domain whose origin is the location of the reference node. The initial coordinates of the three mobile nodes along with their centroid are shown. The trajectory of the formation movement is changed to be towards the left direction (opposing that of Experiment 1). Five discrete locations of the centroid along with its motion trajectory and their respective time instances are also shown. The total traveling time is 1.14 sec. The three circular points show the initial position (at t = 0) of the triad formation (M_1, M_2, and M_3).

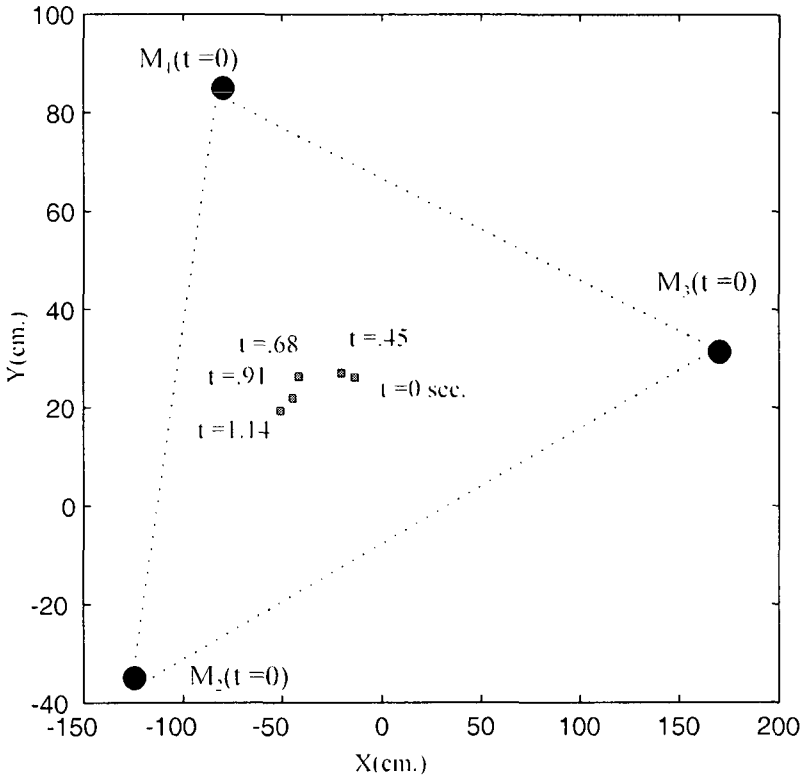


Figure 5.7: Estimated formation centroid trajectory-Experiment #2.

The results of Experiment #2 are presented in Figures 5.8 to 5.10. The figures show the estimated polar coordinates for the three nodes corresponding to the time instances indicated beside each formation centroid marked on Figure 5.5. Among the three mobile nodes, the longest traveled distance from the reference node is 172 cm, and is attributed to M_3. This can be calculated from the polar coordinates shown in Figures 5.6 to 5.8. The figures also show that the range of the radial coordinate error in the system is ± 3.5 cm and the range of the angular coordinate error is ± 6°. It is noted that there is no improvement in the system's accuracy. The error is still biased due to the

multipath reflection of the parabolic reflector problem detailed above. The accuracy of the used designed cone is recorded and left for further investigation.

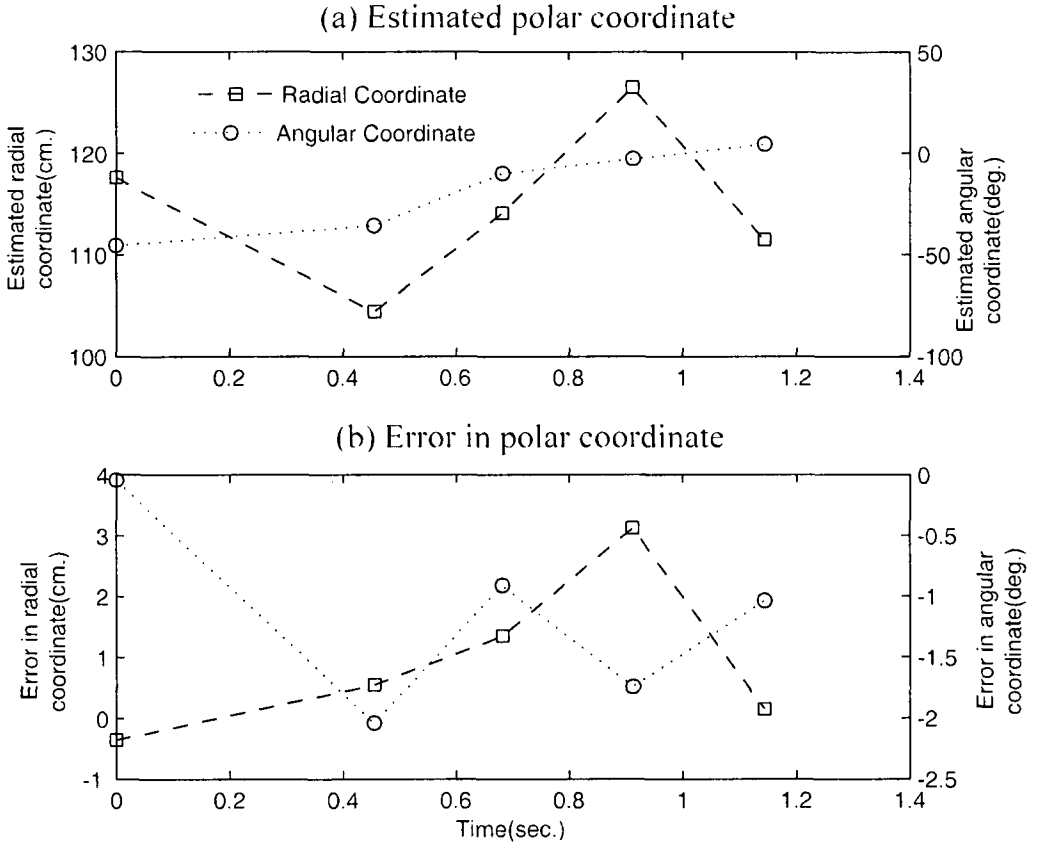


Figure 5.8: Robot #1 - Experiment #2; (a) Estimated polar radial coordinate (radial, angular), (b) Error in estimated polar coordinate (radial, angular).

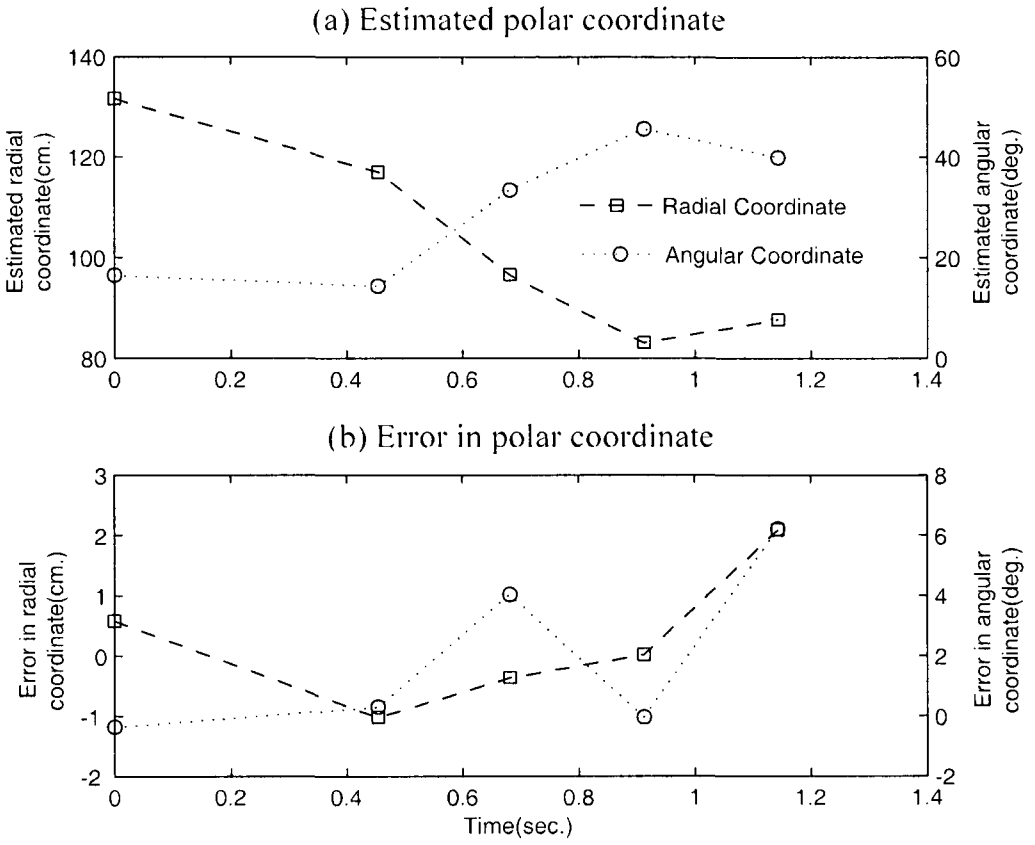


Figure 5.9: Robot #2 - Experiment #2; (a) Estimated polar radial coordinate (radial, angular), (b) Error in estimated polar coordinate (radial, angular).

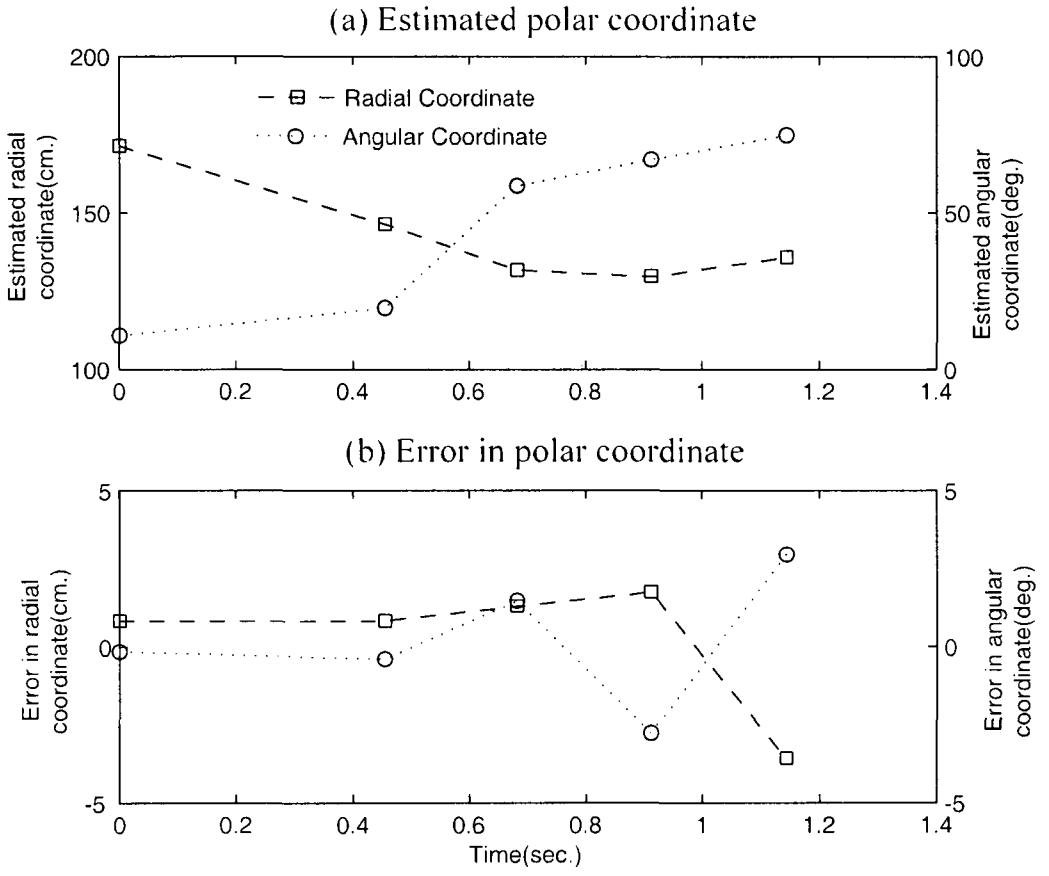


Figure 5.10: Robot #3 - Experiment #2; (a) Estimated polar radial coordinate (radial, angular), (b) Error in estimated polar coordinate (radial, angular).

These experiments were aimed at revealing the accuracy of the proposed localization network to the exclusion of errors resulting from the speed of the platforms. Errors due to the mobility of the platforms are directly related to the acquisition cycle period, the relative speed, and heading of the mobile robots. The experimental setup employed the SRF02 transceiver with its limited range circuit that does not allow control over the receiving circuit threshold. This was a primary reason to increase the blank time value from a range of few microseconds to 35 msec which does not match the proposed design as stated in Section 3.3.1. However, these experiments prove the functionality of

the round-trip localization network and its advantage of low cost. Moreover, the multidimensional scale modeling together with the statistical orthogonal Procrustes method were shown to provide an accurate conversion from relative positioning to absolute positioning. This approach leads the way for employing a low-cost sensing technology in localization problems without the need for fixed beacons in the environment.

5.5 Discussion

This chapter has presented a set of experiments that demonstrate how the proposed localization system parameters' calibration affects the system's accuracy and consistency. The results show that the accuracy is limited by the parabolic cone design which contributes to a bias in the error of the TOF estimations. However these experiments show that the system is not affected by false echoes from surrounding objects in the environment. This is because the reflected signal's power is less than the direct broadcasted signal from a source node and it will not be picked up by the receiver node. It is believed that using a ranging circuit as described in Section 3.3.1 will enhance the overall system's tracking capability.

In addition, the multidimensional scale modeling and the statistical orthogonal Procrustes method were implemented and tested. The results showed the power of the MDS modeling in converting the relative positioning into absolute positioning which can incrementally provide the heading angle of each robot.

Chapter 6
Conclusions

6.1 Concluding Remarks

This research addresses the problem of cooperative localization of multi-robot systems in indoor environments. The goal of the work was to develop a localization system able to provide accurate inter-robot distances at fairly high navigation speed. The developed solution's cost and its associated environmental infrastructure requirements had to be minimized. The chapter summarizes the major achievements of the work.

6.1.1 Round-Trip localization protocol

The work resulted in the development of a fully meshed round-trip ultrasonic ranging network. The proposed network analysis and MATLAB simulation results have been presented in Chapter 3. The method uses the inter-robot round trip TOF estimations to build the Euclidean distance matrix. The ranging network can be used in either of two modes of operation; communication-dependent and communication-independent. Experimental verification of the proposed localization network was carried out through an in-house developed prototype ultrasonic localization module. The hardware and software engineering and design of the ultrasonic localization module has been presented in Chapter 4.

In the communication-dependent mode the Euclidean distance matrix is calculated centrally at the reference node and conveyed to the mobile nodes. Hence this approach is suitable for tightly coupled maneuvers. It was found by means of simulation that the accuracy of this mode decreases

as the mobile robots move away from a fixed reference node. The combined tessellation approach has been used to improve the accuracy of the communication-dependent approach. In this technique, the TOF is used between successive nodes up to the point when a tessella can be formed. The worst case accuracy for localization between the moving node formation would be that associated with the displacement of intermediate nodes while the distance of the last side of the triangle is being estimated.

In the communication-independent mode, each node calculates its own Euclidean distance matrix by listening to the broadcasts from all the other nodes. As such, the matrix reference times may not be coordinated. Each node can maneuver independently to keep its pose in the formation, and hence this approach is suitable more for loosely coupled navigation.

Both the communication-dependent and the combined tessellation approaches were simulated in MATLAB. The performance of each method was compared to that of the traditional RF-based synchronization ultrasonic localization system. The results show that errors due to the proposed approaches are slightly higher than those of the RF synchronized network.

6.1.2 Parabolic reflector

For experimental purposes, a reflector was required to provide an omnidirectional spread of the ultrasonic signals emitted from a single transducer. For that purpose, the ultrasonic beam characteristics are considered. The parabolic reflector was designed to favor a predetermined disk alike ultrasonic signals. The design was carried out to avoid ground reflections within a range from the transmitting node. Moreover, the relation between the ultrasonic beam intensity and the angle of reflection relative to the tangent of the parabola curvature was stated. Section 4.2.2 details the

engineering design of the parabolic reflector.

6.2 Summary of Contributions

This research contributes to the general body of knowledge in the area of multi-robot cooperative localization. The three major contributions of this thesis are outlined below.

1) The development of an accurate and low cost cooperative localization network for multi-robot systems operating at high navigational speeds in indoor environments.
2) The localization network can operate in either of two modes; a communication-dependent mode suitable for tightly coupled tasks, and a less costly communication-independent mode suitable for loosely coupled tasks like leader-follower navigation.
3) The development of a prototype localization module that features a parabolic cone reflector for an omnidirectional broadcast of the ultrasonic ranging signal. The engineering design of the parabolic cone was detailed in Chapter 4. It relates the ultrasonic beam characteristics and reflection pattern to the design parameters of the cone.

6.3 Future Work

The thesis addresses the development and prototyping of a cooperative multi-robot localization method at low cost and acceptable accuracy. Experiments carried out on the prototype helped identify areas where modifications are necessary to improve the accuracy of the system and render its performance more robust and suitable for industrial deployment. These areas are summarized below.

6.3.1 Ultrasonic Transceiver Ranging Circuit

The currently used digitally-controlled SRF02 module has two critical limitations; it allows a maximum duration of 65 msec in receiving mode and it does not allow to control the maximum threshold applied to detect a received burst. This does not suit the asynchronous operation of the proposed method. An alternative solution is to develop an ultrasonic transducer prototype ranging circuit that can be driven directly by the ATmega128 microcontroller. This would offer more flexibility and would allow control over the threshold applied to detect a received burst and consequently reduces the time delay as detailed in Chapter 3.

Another improvement in the network, that can further shorten the localization cycle period, would be the employment of the FDMA access control method. In this approach all the nodes are allowed to simultaneously fire at a unique frequency, thus sharing the acoustic communication channel bandwidth. This, however, would require digital filters to be implemented at the reference node in order to differentiate the ultrasonic signals of the different mobile nodes. Since most ultrasonic transducers have a bandwidth of 2 kHz, a fairly large network with adequate differentiation can be achieved.

6.3.2 Parabolic reflector

The multipath ultrasonic reflection due to the curvature surface of the cone was recorded in Section 4.2.2. This problem biased the error by 3.6 cm based on the experimental results in Section 5.4.1. The cone height is found to be the main source of the added error. Although, reducing the cone height seems to be the direct solution to reduce that bias, other geometrical shapes may be considered for further analysis.

References

[1] F. Brind'amour, "Navigation Sensor for Collaborative Mobile Robots",Ph.D. Thesis, Ottawa, Canada, 2005.

[2] M. Addlesee, R. Curwen, S. Hodges, J. Newman, P. Steggles, A. Ward, and A. Hopper, "Implementing a Sentient Computing System," IEEE Computer Magazine, vol. 34, no.8, pp.50-56, 2001.

[3] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The Cricket Location Support System," in Proc. of the Sixth Int'l Conf. Mobile Computing and Networking (MobiCom), pp. 32–43, Aug. 2000.

[4] P. Bahl and V. N. Padmanabhan, "RADAR: An In building RF-based User Location and Tracking System," in Proc. of the IEEE Conf. Computer Comm. (InfoCom), vol. 2, pp. 775-784, Mar. 2000.

[5] J. Borenstein, H. R. Everett, and L. Feng., "Where am I? Sensors and methods for mobile robot positioning. Technical report," University of Michigan, MI, USA, April 1996.

[6] F. Chenavrier and J.L.Crowley, "Position estimation for a mobile robot using vision and odometry," in Proc. of the IEEE International Conference on Robotics and Automation, pp. 2588-2592, Nice, May 1992.

[7] C. Ming Wang, "Location estimation and uncertainty analysis for mobile robots," in Proc. Intl. Conference on Robotics and Automation, pp.1230-1235, Philadelphia, April 1988.

[8] K.S. Chong, and L. Kleeman, "Accurate odometry and error modeling for a mobile robot," in Proc. of the Intl. Conference on Robotics and Automation, vol. 4, pp. 2783–2788, Albuquerque, 1997.

[9] B. Barshan and H. Durrant-Whyte, "Inertial navigation systems for mobile robots," in IEEE Transactions on Robotics and Automation, vol. 11, pp. 328 – 342, June 1995.

[10] K. Hwang, D. Kim, D. Lee, T. Kuc, "A Simple Ultrasonic GPS System for Indoor Mobile Robot System using Kalman Filtering," in SICE-ICASE Intl. Joint Conference, pp. 2915-2918, Busan, Oct. 2006.

[11] E. Foxlin, M. Harrington, and G. Pfeifer, "Constellation: A Wide-Range Wireless Motion-Tracking System for Augmented Reality and Virtual Set Applications," in Proc. of the ACM SIGGRAPH Conference, pp. 371-378, Orlando, 1998.

[12] M. Mataric, "Environment Learning Using a Distributed Representation," in Proc. of the IEEE Int'l Conference Robotics and Automation (ICRA 90), pp. 402-406, Los Alamitos, 1990.

[13] M. Hazas, J. Scott, and J. Krumm, "Location-aware computing comes of age," IEEE Computer Magazine, vol. 37, no.2, pp.95–97, Feb. 2004.

[14] T. Fernandez, J. Rodas , C. Escudero, and D. Iglesia, "Bluetooth Sensor Network Positioning System with Dynamic Calibration," IEEE ISWCS, pp. 45-49, Trondheim, 2007.

[15] "Ekahau positioning system", www.ekahau.com, 2008.

[16] M. Youssef. A. Agrawala. and A. U. Shankar, "WLAN Location Determination via Clustering and Probability Distributions," in Proc. of the IEEE 1st. Intl. Conf. on Pervasive Computing and Communications, pp. 143- 150, March. 2003.

[17] V. Otsason, A. Varshavsky, A. LaMarca, "GSM Indoor Localization," in Pervasive and Mobile Computing Journal, vol. 3, no. 6, pp. 698-720, Dec. 2007.

[18] R. Christ and R. Lavigne, "Radio frequency-based personnel location systems," in Proc. of

the IEEE 34th Annual Intl. Carnahan Conference on Security Technology, pp. 141-150, 2000.

[19] G. Jin, X. Lu, and M. Park, "An Indoor Localization Mechanism Using Active RFID Tag," in Proc. of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, 2006.

[20] Y. Zhao, and Y. Liu, "VIRE: Active RFID-based Localization Using Virtual Reference Elimination," in Proc. of the Intl. Conference on Parallel Processing, pp.56-56, Sept. 2007.

[21] A. Rynn, W. A. Malik, and S. Lee, "Sensor based localization for multiple mobile robots using virtual links," in IEEE Intl. Conference on Intelligent Robots and Systems, vol. 2, pp. 1771- 1776, Oct. 2003.

[22] J. Spletzer, A.K. Das, R. Fierro, C. J. Taylor, V. Kummar, and J. P. Ostrowski, "Cooperative Localization and Control for Multi-Robot Manipulation," in Proc. of the IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, pp. 631- 638, Maui, HI, USA, 2001.

[23] E. Elnahrawy, X. Li, and R. P. Martin, "The limits of localization using signal strength: A comparative study," in the 1st. IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON), pp. 406- 414, Santa Clara, CA, USA, Oct. 2004.

[24] D. Nicelescu and B. Nath, "Ad hoc positioning (APS) using AOA," in Proc. of the IEEE Global Commuincation Conference, pp. 2926-2931, Nov. 2001.

[25] A. R. Jimenez, F. Seco, R. Ceres, and L. Calderon, "Absolute Localization using Active Beacons: A survey and IAI-CSIC contributions," Institute for Industrial Automation, CSIC Madrid, 2004.

[26] A. Savvides, C. Han, and M. Srivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in Proc. of the 7th. ACM Annual International Conference on Mobile

Computing and Networking (MobiCom), pp. 166-179, July 2001.

[27] J. M. Rabaey, M. J. Ammer, J. L. da Silva, D. Patel, and S. Roundy, "Picorodio supports ad hoc ultra-low power wireless networking," IEEE Computer Magazine, vol. 33, no. 7, pp. 42–48, July 2000.

[28] E. Dijk, "Indoor Ultrasonic Position Estimation Using a Single Base Station",Ph.D. Thesis, Eindhoven, Netherlands, 2004.

[29] D. Hallaway, T. Hollerer, and S. Feiner, "Coarse, inexpensive, infrared tracking for wearable computing". In 7th IEEE International Symposium on Wearable Computers (ISWC), pp. 69–78, White Plains, NY, USA, Oct. 2003.

[30] R. Want, A. Hopper, V. Falcao, and J. Gibbons, " The active badge location system," in ACM Transactions on Office Information Systems (TOIS), vol. 10, no. 1, pp. 91–102, Jan. 1992.

[31] C. Cohen and F. Koss, "A Comprehensive Study of Three Object Triangulation," in Proc. of the 1993 SPIE Conference on Mobile Robots, pp.95-106, Boston, Nov. 1993.

[32] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, and S. Shafer, "Multi-camera multi-person tracking for easy living," in Proc. of the Third IEEE International Workshop on Visual Surveillance (VS'2000), pp. 3–10, IEEE Computer Society, 2000.

[33] L. Navarro-Serment, R. Grabowski, C.J.J. Paredis, P.K. Khosla, "Modularity in Small Distributed Robots," in Proc. of the SPIE conference on Sensor Fusion and Decentralized Control in Robotic Systems II, pp. 297-306, Boston, MA, Sep.1999.

[34] J. Borenstein, D. Wehe, L. Feng, and Y. Koren, "Mobile robot navigation in narrow aisles with ultrasonic sensors," in Proc. of the ANS 6th Topical Meeting on Robotics and Remote

Systems, Monterey, California, Feb. 1995.

[35] S.Kim and Y.Kim, "Robot localization using Ultrasonic Sensors," in Proc. of the IEEE/RSl intel. Conference on intelligent Robots and Systems, pp. 3762 - 3766, Sendal, Japan,2004.

[36] C. Randell and H. Muller, "Low Cost Indoor Positioning System," in Proc. of the Intl. Conference on Ubiquitous Computing (UbiComp), pp. 42-48, Atlanta, Georgia, USA, 2001.

[37] F. Seco, A. R. Jiménez, "Multiple Access Interference (MAI) cancellation for CDMA ultrasonic systems," in 3rd Workshop on Biomimetic Ultrasound, Alcalá de Henares, March 2005.

[38] T. Karalar, J. Rabaey, "An RF ToF Based Ranging Implementation for Sensor Networks," in IEEE Intl. Communications Conference, vol. 7, pp. 3347-3352, Istanbul, Turkey, June 2006.

[39] T. Hori, Y. Nishida, T. Kanade, K. Akiyama, "Improving sampling rate with multiplexed ultrasonic emitters," in IEEE Intl. Conference on Systems, Man and Cybernetics, vol.5, pp. 4522-4527, Oct. 2003.

[40] L. Beranek, "ACOUSTICS," Mcgraw-Hill Electical and Electronic Engineering Series 1954.

[41] I. Borg and P. Groenen, "Modern Multidimensional Scaling, Theory and Application," Springer Series in Statistics 2005.

[42] J. Gower, "Generalized Procrustes Analysis," in Journal of Psychometrika, vol. 40, no. 1, pp. 33-51, 1975.

[43] M Akca,, "Generalized Procrustes Analysis and its Applications in Photogrammetry," in Intl. Colloquium at Photogrammetry and Remote Sensing Group of IGP- ETH Zurich, Zurich, Switzerland, July 2003.

Appendix A

Source Code Listing

This appendix includes the computer programs developed within the framework of implementation this research. It includes the network simulator developed using *Matlab® version 7.0.1*, the low-level embedded firmware programmed in Assembly language using the ATmega 128 instruction set, and the high-level Human-Machine interface code developed using Visual Basic 6.

Listings

A.1 Matlab Network Simulator

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%File Name:   roundtripsim.m
%Description:  Round-trip localization network simulator
%Author:      Hany Geris, University of Ottawa
%Date:        June 2008
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
TIME=9000;
Max_Distance=20; % max. distance in the network is 20m
% Calculating the critical distance
max_power=1; min_threshold=.2;% threhold to receive a signal from 20 m
away
% p(x)=max. power - a*x^2 where is x is the distance in m
a=(max_power-min_threshold)/((Max_Distance)^2);
mu=.3 ;% reflection and dissipation factor
Critical_Distance=sqrt(((1-mu)*max_power-min_threshold)/(a*(2-mu)));
% number of robots
m = 3;
% virtual centroid speed (mm/msec)
rs=1.5;
% Assume the Sound speed is (340 m/s) == (340 mm/msec)
ss= 340.0;
cycles=1000; % to track no. of localization cycles
% radius of circluar trajectory of the centroid
radius=12000 ;
sample=50;
t=0:sample:TIME;
% blank time 5 msec.
Blank_Time=5;
% Triangle initial position
x1_act(1)=3000;y1_act(1)=3000;
x2_act(1)=1000;y2_act(1)=4500;
x3_act(1)=2000;y3_act(1)=500;
% calculating the initial location of centroid
cx_act(1)=(x1_act(1)+x2_act(1)+x3_act(1))/3;
cy_act(1)=(y1_act(1)+y2_act(1)+y3_act(1))/3;
% calculating the relative distances and angles between points and centroid
% by applying the law of cosines
r1=sqrt((y1_act-cy_act(1))^2+(x1_act-cx_act(1))^2);
r2=sqrt((y2_act-cy_act(1))^2+(x2_act-cx_act(1))^2);
r3=sqrt((y3_act-cy_act(1))^2+(x3_act-cx_act(1))^2);
Alpha_2=acos((r1^2+r2^2-((y2_act(1)-y1_act(1))^2+(x2_act(1)-x1_act(1))
^2))/(2*r1*r2));
Alpha_3=acos((r1^2+r3^2-((y3_act(1)-y1_act(1))^2+(x3_act(1)-x1_act(1))
^2))/(2*r1*r3));
% calcualting the center of the circualr trajectory of the centroid
ry=0;
rx=cx_act(1)+sqrt((radius)^2-(cy_act(1))^2);
% calculating the initial angle of the tangent to the circular trajectory
% of the centroid
Theta= pi/2+atan(cy_act(1)/(rx-cx_act(1)));
% calculating the Orientation of the initial position of the 3 robots
% (i.e. angle of Centroid-x1) in degrees
if x1_act(1)>= cx_act(1)
  Theta_1(1)=atan((y1_act(1)-cy_act(1))/(x1_act(1)-cx_act(1)));
else
  Theta_1(1)=(pi+atan((y1_act(1)-cy_act(1))/(x1_act(1)-cx_act(1))));
end
tangent(1)=atan((rx-cx_act(1))/(cy_act(1)-ry));
counter1=0;
for tt=0:sample:TIME
  counter1=counter1+1;
  if tt < (3/5)*TIME
                    cx_act(counter1) =
cx_act(1)+radius*sin(Theta)-radius*sin(rs*tt/radius+Theta);
                    cy_act(counter1) =
```

```
cy_act(1)+radius*cos(Theta)-radius*cos(rs*tt/radius+Theta);
   end
   if tt == (3/5)*TIME
      cx_init=cx_act(counter1-1);
      cy_init=cy_act(counter1-1);
      time_init=tt-sample;
      if rx > cx_init
         Theta=atan(( rx-cx_init)/(cy_init));
      else
         Theta=-atan(( cx_init-rx)/ (cy_init));
      end
   end
   if tt >= (3/5)*TIME
                              c x _ a c t ( c o u n t e r 1 ) =
cx_init-radius*sin(Theta)+radius*sin(rs*(tt-time_init)/radius+Theta);
                              c y _ a c t ( c o u n t e r 1 ) =
cy_init+radius*cos(Theta)-radius*cos(rs*(tt-time_init)/radius+Theta);
   end
end
%calculating the center of the touching circle
rx2=2*cx_init-rx;
ry2=2*cy_init-ry;
Theta_1(1)= atan((y1_act(1)-cy_act(1))/(x1_act(1)-cx_act(1)));
tangent(1)=atan((rx-cx_act(1))/(cy_act(1)-ry));
Const_theta=Theta_1(1)-tangent(1);
for i=2:length(t)
   if i<=(3/5)*(TIME/sample)
      tangent(i)=atan((rx-cx_act(i))/(cy_act(i)-ry));
      Theta_1(i)=Const_theta+tangent(i);
   else
      tangent(i)=atan((rx2-cx_act(i))/(cy_act(i)-ry2));
      Theta_1(i)=Const_theta+tangent(i);
   end
end
% % testing the constant orientation relative to the curve
difference=Theta_1-tangent;
% Calculating the other two points coordinates of the triangle
for i=2:length(t)
   x1_act(i)=cx_act(i)+r1*cos(Theta_1(i));
   y1_act(i)=cy_act(i)+r1*sin(Theta_1(i));
   x3_act(i)=cx_act(i)+r3*cos(Theta_1(i)-Alpha_3);
   y3_act(i)=cy_act(i)+r3*sin(Theta_1(i)-Alpha_3);
   x2_act(i)=cx_act(i)+r2*cos(Theta_1(i)+Alpha_2);
   y2_act(i)=cy_act(i)+r2*sin(Theta_1(i)+Alpha_2);
end
% The next part of the simulator is dealing with the estimation of the
% formation coordinates using the proposed round-trip protocol
% Building the Distance Matrix between all nodes
xcomp(1)=0;xcomp(2)=x1_act(1);xcomp(3)=x2_act(1);xcomp(4)=x3_act(1);
ycomp(1)=0;ycomp(2)=y1_act(1);ycomp(3)=y2_act(1);ycomp(4)=y3_act(1);
Distance_Matrix1=zeros(m+1,m+1);
Distance_Matrix2=zeros(m+1,m+1);
timers1=zeros(9,m+1);
timers2=zeros(9,m+1);
timers1(1,1) = input('The Base starts to send burst at time (msec): ');
for i=2:m+1;
   timers1(1,i) = input('Enter the time at robot to start receive (msec): ');
   timers2(1,i)=timers1(1,i);
end
temp=zeros(length(t),1);
x_act=[temp x1_act' x2_act' x3_act'];
y_act=[temp y1_act' y2_act' y3_act'];
% Estimated log matrix
est1_log=zeros(5*cycles,13);
% Estimated log matrix
est2_log=zeros(6*cycles,13);
% Building the Distance Matrix between all nodes
for i=1:m+1
   for k=1:m+1

Distance_Matrix1(i,k)=sqrt(((xcomp(i)-xcomp(k))^2)+((ycomp(i)-ycomp(k
))^2));
      Distance_Matrix2(i,k)=Distance_Matrix1(i,k);
   end
```

```
end
counter1=0;
for y=1:cycles
  sender=1; % at the start of each cycle, set the sender flag to be the
  % reference node
  step=1; % reset the steps flag to follow the steps in one cycle
 % Now Let's Calculate Delta Time Matrix (msec)using the Sound speed and
  % also the Max Waiting time based on the initial Distance Matrix
  Delta=zeros(m+1,m+1);
  for i=1:m+1
    % searching the Distance Matrix for each sender (column)for the max.
    % distance between it and other nodes
    Max_Distance=0.0;
    Min_Distance=100000000;
    for k=1:m+1
      % look for the far node from each sensder (column)
      if ge(Distance_Matrix1(k,i),Max_Distance)
        Max_Distance=Distance_Matrix1(k,i);
        order_of_further_robot=k;
      end
      if lt(Distance_Matrix1(k,i),Min_Distance)&& (k~=i)
        Min_Distance=Distance_Matrix1(k,i);
        order_of_nearest_robot=k;
      end
    end
    if (Min_Distance/1000)> Critical_Distance
      Delta(i,i)=Blank_Time;
    else
      Delta(i,i)=2*Min_Distance/ss;
    end
    for k=1:m+1
      if k~=i
        if k==i+1
          Delta(k,i)=Blank_Time;
        else
          if (i==m+1)&& (k==1)
            Delta(k,i)=Blank_Time;
          else
            Delta(k,i)=Blank_Time;
          end
        end
      end
    end
  end
  while step <= 7
    for i=2:m+1
      if i~=sender
        if y ==1
          % first cycle will be done while robots are rest
          % so, the actual position will be the initial position
          xcomp(i)=x_act(1,i);
          ycomp(i)=y_act(1,i);
        else
          % according the time interval, Find the related
          % actual position by parsing the lookup table at the
          % starting of each step marked by the start of broadcasting
          % by any node
          index=1+floor(timers1(step,sender)/sample);

xcomp(i)=(x_act(index+1,i)-x_act(index,i))*(timers1(step,sender)-((index-1
)*sample))/sample+x_act(index,i);

ycomp(i)=(y_act(index+1,i)-y_act(index,i))*(timers1(step,sender)-((index-1
)*sample))/sample+y_act(index,i);
          end
        end
      end
      % now the sender will broadcast the ultrasonic signal to the
      % network and the instants of receiving the ultrasonic signal can
      % be calculated as follows
      for i=1:m+1
        if i~=sender
                                timers1(step+1,i)=  timers1(step,sender)  +
sqrt((xcomp(i)-xcomp(sender))^2+(ycomp(i)-ycomp(sender))^2)/ss;
```

```
      else
        % the sender timer is not updated for that time sample
        timers1 (step+1,sender)=0;
      end
    end
    % lets Add delta time to each robot now
    for i=1:m+1
      if i==sender
       timers1(step+2,sender)=timers1(step,sender)+Delta(sender,sender);
      else
        timers1(step+2,i)=timers1(step+1,i)+Delta(i,sender);
      end
    end
    step=step+2;
    sender=sender+1;
end
% The Total Cycle Time
Total_Time=timers1(step,1);
counter1=counter1+1;
% calculating the RF-Synchronized inter-node distances
est1_log(counter1,8)=ss*(timers1(4,1)-timers1(3,2)) ; % D01
est1_log(counter1,9)=ss*(timers1(6,1)-timers1(5,3)) ; % D02
est1_log(counter1,10)=ss*(timers1(8,1)-timers1(7,4)) ; % D03
est1_log(counter1,11)=ss*(timers1(6,2)-timers1(5,3)) ; % D12
est1_log(counter1,12)=ss*(timers1(8,2)-timers1(7,4)) ; % D13
est1_log(counter1,13)=ss*(timers1(8,3)-timers1(7,4)) ; % D23
% Based on the 6 instants at which ultrasonic broadcast is received from
% the sender node, the estimation can be done as follows
% estimate T01 ( 0->1->0 )
if counter1 >10
   est1_log(counter1,1)=timers1(4,1);%+est_log(counter1-1,1);
else
   est1_log(counter1,1)=timers1(4,1);
end
est1_log(counter1,2)=(timers1(4,1)-timers1(1,1)-Delta(2,1))/2;
for i=3:7
   if counter1==1
      est1_log(counter1,i)=0;
   else
      est1_log(counter1,i)=est1_log(counter1-1,i);
   end
end
% estimate T12 ( 1->2->1 )
counter1=counter1+1;
if counter1 >10
   est1_log(counter1,1)=timers1(6,2);%+est_log(counter1-2,1);
else
   est1_log(counter1,1)=timers1(6,2);
end
est1_log(counter1,5)=(timers1(6,2)-timers1(3,2)-Delta(3,2))/2;
for i=2:7
   if i ~=5
      if counter1==1
         est1_log(counter1,i)=0;
      else
         est1_log(counter1,i)=est1_log(counter1-1,i);
      end
   end
end
%estimate T02 ( 0->2->0 )
counter1=counter1+1;
if counter1 >10
   est1_log(counter1,1)=timers1(6,1);%+est_log(counter1-3,1);
else
   est1_log(counter1,1)=timers1(6,1);
end

est1_log(counter1,3)=(timers1(6,1)-timers1(1,1)-(timers1(5,3)-timers1(2,3)
))/2;
   est1_log(counter1,2)=est1_log(counter1-1,2);
   for i=4:7
      if counter1==1
         est1_log(counter1,i)=0;
      else
```

```
      est1_log(counter1,i)=est1_log(counter1-1,i);
    end
  end
  % estimate T23 ( 2->3->2 )
  counter1=counter1+1;
  if counter1 >10
    est1_log(counter1,1)=timers1(8,3);%+est1_log(counter1-4,1);
  else
    est1_log(counter1,1)=timers1(8,3);
  end
  est1_log(counter1,7)=(timers1(8,3)-timers1(5,3)-Delta(4,3))/2;
  for i=2:7
    if (i ~=7)
      if counter1==1
        est1_log(counter1,i)=0;
      else
        est1_log(counter1,i)=est1_log(counter1-1,i);
      end
    end
  end
  %estimate T13 ( 1->3->1 )
  counter1=counter1+1;
  if counter1 > 10
    est1_log(counter1,1)=timers1(8,2);%+est1_log(counter1-5,1);
  else
    est1_log(counter1,1)=timers1(8,2);
  end

est1_log(counter1,6)=(timers1(8,2)-timers1(3,2)-(timers1(7,4)-timers1(4,4)
))/2;
  for i=2:5
    est1_log(counter1,i)=est1_log(counter1-1,i);
  end
  est1_log(counter1,7)=est1_log(counter1-1,7);
  % estimate T03 based on the triangle
  index=1+floor(est1_log(counter1,1)/sample);
  if (counter1<=5)
    x1_actual=x_act(1,2);
    y1_actual=y_act(1,2);
    x2_actual=x_act(1,3);
    y2_actual=y_act(1,3);
    x3_actual=x_act(1,4);
    y3_actual=y_act(1,4);
  else

x1_actual=(x_act(index+1,2)-x_act(index,2))*(est1_log(counter1,1)-((inde
x-1)*sample))/sample+x_act(index,2);

y1_actual=(y_act(index+1,2)-y_act(index,2))*(est1_log(counter1,1)-((inde
x-1)*sample))/sample+y_act(index,2);

x2_actual=(x_act(index+1,3)-x_act(index,3))*(est1_log(counter1,1)-((inde
x-1)*sample))/sample+x_act(index,3);

y2_actual=(y_act(index+1,3)-y_act(index,3))*(est1_log(counter1,1)-((inde
x-1)*sample))/sample+y_act(index,3);

x3_actual=(x_act(index+1,4)-x_act(index,4))*(est1_log(counter1,1)-((inde
x-1)*sample))/sample+x_act(index,4);

y3_actual=(y_act(index+1,4)-y_act(index,4))*(est1_log(counter1,1)-((inde
x-1)*sample))/sample+y_act(index,4);
  end

x1_est=ss*(est1_log(counter1,2))*(x1_actual/sqrt(x1_actual^2+y1_actual^2));

y1_est=ss*(est1_log(counter1,2))*(y1_actual/sqrt(x1_actual^2+y1_actual^2));

x2_est=ss*(est1_log(counter1,3))*(x2_actual/sqrt(x2_actual^2+y2_actual^2));

y2_est=ss*(est1_log(counter1,3))*(y2_actual/sqrt(x2_actual^2+y2_actual^2));
  d=sqrt((x2_est-x1_est)^2+(y2_est-y1_est)^2);
  a=((ss*est1_log(counter1,7))^2 - (ss*est1_log(counter1,6))^2 + d^2)/(2*d)
;
```

```
h= sqrt((ss*est1_log(counter1,7))^2 - a^2);
dx = x1_est - x2_est;
dy = y1_est - y2_est;
intersection_x = x2_est + (dx * a/d);
intersection_y = y2_est + (dy * a/d);
x31_est = intersection_x + h*( y1_est - y2_est )/d ;
x32_est = intersection_x - h*( y1_est - y2_est )/d ;
y31_est = intersection_y + h*( x1_est - x2_est )/d ;
y32_est = intersection_y - h*( x1_est - x2_est )/d ;
if (y3_actual-y31_est) > (y3_actual-y32_est)
   y3_est=y31_est;
else
   y3_est=y32_est;
end
if (x3_actual-x31_est) > (x3_actual-x32_est)
   x3_est=x31_est;
else
   x3_est=x32_est;
end
est1_log(counter1,4)=(sqrt(x3_est^2+y3_est^2))/ss;
for i=1:4
   if y==1
      timers1(1,i)=0;
   else
      timers1(1,i)=timers1(9,i);
   end
end
Distance_Matrix1(1,2)=ss*est1_log(counter1,2);
Distance_Matrix1(2,1)=Distance_Matrix1(1,2);
Distance_Matrix1(1,3)=ss*est1_log(counter1,3);
Distance_Matrix1(3,1)=Distance_Matrix1(1,3);
Distance_Matrix(1,4)=ss*est1_log(counter1,4);
Distance_Matrix1(4,1)=Distance_Matrix1(1,4);
Distance_Matrix1(2,3)=ss*est1_log(counter1,5);
Distance_Matrix1(3,2)=Distance_Matrix1(2,3);
Distance_Matrix1(2,4)=ss*est1_log(counter1,6);
Distance_Matrix1(4,2)=Distance_Matrix1(2,4);
   Distance_Matrix1(3,4)=ss*est1_log(counter1,7);
   Distance_Matrix1(4,3)=Distance_Matrix1(3,4);
   if Total_Time >=(TIME-400)
      break;
   end
end
counter2=0;
for y=1:cycles
   sender=1;
   step=1;
   Delta=zeros(m+1,m+1);
   for i=1:m+1
      Max_Distance=0.0;
      Min_Distance=100000000;
      for k=1:m+1
         if ge(Distance_Matrix2(k,i),Max_Distance)
            Max_Distance=Distance_Matrix2(k,i);
            order_of_further_robot=k;
         end
         if lt(Distance_Matrix2(k,i),Min_Distance)&& (k~=i)
            Min_Distance=Distance_Matrix2(k,i);
            order_of_nearest_robot=k;
         end
      end
      if (Min_Distance/1000)> Critical_Distance
         Delta(i,i)=Blank_Time;
      else
         Delta(i,i)=2*Min_Distance/ss;
      end
      for k=1:m+1
         if k~=i
            if k==i+1
               Delta(k,i)=Blank_Time;
            else
               if (i==m+1)&& (k==1)
                  Delta(k,i)=Blank_Time;
               else
```

```
            Delta(k,i)=Blank_Time;
          end
        end
      end
    end
  end
  while step <= 7
    for i=2:m+1
      if i~=sender
        if y ==1
          xcomp(i)=x_act(1,i);
          ycomp(i)=y_act(1,i);
        else
          index=1+floor(timers2(step,sender)/sample);

xcomp(i)=(x_act(index+1,i)-x_act(index,i))*(timers2(step,sender)-((index-1
)*sample))/sample+x_act(index,i);

ycomp(i)=(y_act(index+1,i)-y_act(index,i))*(timers2(step,sender)-((index-1
)*sample))/sample+y_act(index,i);
        end
      end
    end
    for i=1:m+1
      if i~=sender
                              timers2(step+1,i)= timers2(step,sender) +
sqrt((xcomp(i)-xcomp(sender))^2+(ycomp(i)-ycomp(sender))^2)/ss;
      else
        timers2 (step+1,sender)=0;
      end
    end
    for i=1:m+1
      if i==sender
       timers2(step+2,sender)=timers2(step,sender)+Delta(sender,sender);
      else
        timers2(step+2,i)=timers2(step+1,i)+Delta(i,sender);
      end
    end
  end
  step=step+2;
  sender=sender+1;
end
% The Total Cycle Time
Total_Time=timers2(step,1);
counter2=counter2+1;
% calculating the RF-Synchronized inter-node distances
est2_log(counter2,8)=ss*(timers2(4,1)-timers2(3,2)) ; % D01
est2_log(counter2,9)=ss*(timers2(6,1)-timers2(5,3)) ; % D02
est2_log(counter2,10)=ss*(timers2(8,1)-timers2(7,4)) ; % D03
est2_log(counter2,11)=ss*(timers2(6,2)-timers2(5,3)) ; % D12
est2_log(counter2,12)=ss*(timers2(8,2)-timers2(7,4)) ; % D13
est2_log(counter2,13)=ss*(timers2(8,3)-timers2(7,4)) ; % D23
% Based on the 6 instants at which ultrasonic broadcast is received from
% the sender node, the estimation can be done as follows
% estimate T01 ( 0->1->0 )
est2_log(counter2,1)=timers2(4,1);
est2_log(counter2,2)=(timers2(4,1)-timers2(1,1)-Delta(2,1))/2;
for i=3:7
  if counter2==1
    est2_log(counter2,i)=0;
  else
    est2_log(counter2,i)=est2_log(counter2-1,i);
  end
end
% estimate T12 ( 1->2->1 )
counter2=counter2+1;
est2_log(counter2,1)=timers2(6,2);
est2_log(counter2,5)=(timers2(6,2)-timers2(3,2)-Delta(3,2))/2;
for i=2:7
  if i ~=5
    if counter2==1
      est2_log(counter2,i)=0;
    else
      est2_log(counter2,i)=est2_log(counter2-1,i);
    end
```

```
    end
  end
  %estimate T02 ( 0->2->0 )
  counter2=counter2+1;
  est2_log(counter2,1)=timers2(6,1);

est2_log(counter2,3)=(timers2(6,1)-timers2(1,1)-(timers2(5,3)-timers2(2,3)
))/2;
  % re-estimate T01 as follows ( 0->1->2->0 )

est2_log(counter2,2)=timers2(6,1)-timers2(1,1)-Delta(2,1)-Delta(3,2)-est2_
log(counter2,3)-est2_log(counter2-1,5);
  for i=4:7
    if counter2==1
      est2_log(counter2,i)=0;
    else
      est2_log(counter2,i)=est2_log(counter2-1,i);
    end
  end
  % estimate T23 ( 2->3->2 )
  counter2=counter2+1;
  est2_log(counter2,1)=timers2(8,3);
  est2_log(counter2,7)=(timers2(8,3)-timers2(5,3)-Delta(4,3))/2;
  for i=2:7
    if (i ~=7)
      if counter2==1
        est2_log(counter2,i)=0;
      else
        est2_log(counter2,i)=est2_log(counter2-1,i);
      end
    end
  end
  %estimate T13 ( 1->3->1 )
  counter2=counter2+1;
  est2_log(counter2,1)=timers2(8,2);

est2_log(counter2,6)=(timers2(8,2)-timers2(3,2)-(timers2(7,4)-timers2(4,4)
))/2;
  %re-estimate T12 ( 1->2->3->1 )

est2_log(counter2,5)=timers2(8,2)-timers2(3,2)-Delta(3,2)-Delta(4,3)-est2_
log(counter2,6)-est2_log(counter2-1,7);
  for i=2:4
    est2_log(counter2,i)=est2_log(counter2-1,i);
  end
  est2_log(counter2,7)=est2_log(counter2-1,7);
  %estimate T03 ( 0->2->3->0 )
  counter2=counter2+1;
  est2_log(counter2,1)=timers2(8,1);
        est2_log(counter2,4)=timers2(8,1)-
timers2(1,1)-(timers2(5,3)-timers2(2,3))-est2_log(counter2-1,3)-est2_log(c
ounter2-1,7)-Delta(4,3);
  % re-estimate T01 ( 0->1->2->3->0 )

est2_log(counter2,2)=timers2(8,1)-timers2(1,1)-Delta(2,1)-Delta(3,2)-Delta
(4,3)-est2_log(counter2,4)-est2_log(counter2-1,5)-est2_log(counter2-1,7);
  if counter2==1
    est2_log(counter2,3)=0;
  else
    est2_log(counter2,3)=est2_log(counter2-1,3);
  end
  for i=5:7
    if counter2==1
      est2_log(counter2,i)=0;
    else
      est2_log(counter2,i)=est2_log(counter2-1,i);
    end
  end
   % Now Prepare for next cycle by only copying the last timers2 matrix
values
  % to be the initial values and also calculate the Distance_Matrix2
  for i=1:4
    if y==1
      timers2(1,i)=0;
```

```
      else
        timers2(1,i)=timers2(9,i);
      end
    end
    Distance_Matrix2(1,2)=ss*est2_log(counter2,2);
    Distance_Matrix2(2,1)=Distance_Matrix2(1,2);
    Distance_Matrix2(1,3)=ss*est2_log(counter2,3);
    Distance_Matrix2(3,1)=Distance_Matrix2(1,3);
    Distance_Matrix2(1,4)=ss*est2_log(counter2,4);
    Distance_Matrix2(4,1)=Distance_Matrix2(1,4);
    Distance_Matrix2(2,3)=ss*est2_log(counter2,5);
    Distance_Matrix2(3,2)=Distance_Matrix2(2,3);
    Distance_Matrix2(2,4)=ss*est2_log(counter2,6);
    Distance_Matrix2(4,2)=Distance_Matrix2(2,4);
    Distance_Matrix2(3,4)=ss*est2_log(counter2,7);
    Distance_Matrix2(4,3)=Distance_Matrix2(3,4);
    if Total_Time >=(TIME-400)
      break;
    end
end
%for eact time instant logged in the est1_log , Find the related actual
%position and angle
results1=zeros(counter1,28);
for i=1:counter1
    index=1+floor(est1_log(i,1)/sample);
    if (i<=5)
      x1_actual=x_act(1,2);
      y1_actual=y_act(1,2);
      x2_actual=x_act(1,3);
      y2_actual=y_act(1,3);
      x3_actual=x_act(1,4);
      y3_actual=y_act(1,4);
    else

x1_actual=(x_act(index+1,2)-x_act(index,2))*(est1_log(i,1)-((index-1)*sa
mple))/sample+x_act(index,2);

y1_actual=(y_act(index+1,2)-y_act(index,2))*(est1_log(i,1)-((index-1)*sa
mple))/sample+y_act(index,2);

x2_actual=(x_act(index+1,3)-x_act(index,3))*(est1_log(i,1)-((index-1)*sa
mple))/sample+x_act(index,3);

y2_actual=(y_act(index+1,3)-y_act(index,3))*(est1_log(i,1)-((index-1)*sa
mple))/sample+y_act(index,3);

x3_actual=(x_act(index+1,4)-x_act(index,4))*(est1_log(i,1)-((index-1)*sa
mple))/sample+x_act(index,4);

y3_actual=(y_act(index+1,4)-y_act(index,4))*(est1_log(i,1)-((index-1)*sa
mple))/sample+y_act(index,4);
    end
    x1_est=ss*(est1_log(i,2))*(x1_actual/sqrt(x1_actual^2+y1_actual^2));
    y1_est=ss*(est1_log(i,2))*(y1_actual/sqrt(x1_actual^2+y1_actual^2));
    x2_est=ss*(est1_log(i,3))*(x2_actual/sqrt(x2_actual^2+y2_actual^2));
    y2_est=ss*(est1_log(i,3))*(y2_actual/sqrt(x2_actual^2+y2_actual^2));
    x3_est=ss*(est1_log(i,4))*(x3_actual/sqrt(x3_actual^2+y3_actual^2));
    y3_est=ss*(est1_log(i,4))*(y3_actual/sqrt(x3_actual^2+y3_actual^2));
    results1(i,1)=est1_log(i,1);
    results1(i,2)=x1_actual;
    results1(i,3)=y1_actual;
    results1(i,4)=x1_est;
    results1(i,5)=y1_est;
    results1(i,6)=x2_actual;
    results1(i,7)=y2_actual;
    results1(i,8)=x2_est;
    results1(i,9)=y2_est;
    results1(i,10)=x3_actual;
    results1(i,11)=y3_actual;
    results1(i,12)=x3_est;
    results1(i,13)=y3_est;
    results1(i,14)=sqrt((x2_actual-x1_actual)^2+(y2_actual-y1_actual)^2); %
actual D12
    results1(i,15)=ss*est1_log(i,5); %est. D12
```

```
   results1(i,16)=sqrt((x3_actual-x1_actual)^2+(y3_actual-y1_actual)^2); %
actual D13
   results1(i,17)=ss*est1_log(i,6); %est. D13
   results1(i,18)=sqrt((x2_actual-x3_actual)^2+(y2_actual-y3_actual)^2); %
actual D23
   results1(i,19)=ss*est1_log(i,7); %est. D23
   results1(i,20)=(x3_est+x2_est+x1_est)/3; % estimated Cx
   results1(i,21)=(y3_est+y2_est+y1_est)/3; % estimated Cy
   % calculating the Error in Head angle
   if index<=(3/5)*(TIME/sample)
      Theta_r1= atan((y1_est-results1(i,21))/(x1_est-results1(i,20)));
      tangent_r1 =atan((rx-results1(i,20))/(results1(i,21)-ry));
   else
      Theta_r1= atan((y1_est-results1(i,21))/(x1_est-results1(i,20)));
      tangent_r1 =atan((rx2-results1(i,20))/(results1(i,21)-ry2));
   end
    results1(i,28)=difference(1)-(Theta_r1-tangent_r1); % error in the angle
between head and tangent to the trajectory
   results1(i,22)=(x3_actual+x2_actual+x1_actual)/3; % actual Cx
   results1(i,23)=(y3_actual+y2_actual+y1_actual)/3; % actual Cy

results1(i,24)=sqrt((results1(i,22)-results1(i,20))^2+(results1(i,23)-results1(i
,21))^2); % error in centroid
   a_act=sqrt((x2_actual-x1_actual)^2+(y2_actual-y1_actual)^2);
   a_est=sqrt((x2_est-x1_est)^2+(y2_est-y1_est)^2);
   b_act=sqrt((x3_actual-x1_actual)^2+(y3_actual-y1_actual)^2);
   b_est=sqrt((x3_est-x1_est)^2+(y3_est-y1_est)^2);
   c_act=sqrt((x3_actual-x2_actual)^2+(y3_actual-y2_actual)^2);
   c_est=sqrt((x3_est-x2_est)^2+(y3_est-y2_est)^2);
   actual_angle=acos((a_act^2+b_act^2-c_act^2)/(2*a_act*b_act));
   if (i<=5)
      est_angle=0;
   else
      est_angle=acos((a_est^2+b_est^2-c_est^2)/(2*a_est*b_est));
   end
   results1(i,25)=(actual_angle-est_angle)*(180/pi); % error in triangle angle
in degrees
   results1(i,26)=c_act-c_est ; % error in opposite side of the triangle in mm
   results1(i,27)=actual_angle;
end
%building the second results matrix
results2=zeros(counter2,28);
for i=1:counter2
   index=1+floor(est2_log(i,1)/sample);
   if (i<=6)
      x1_actual=x_act(1,2);
      y1_actual=y_act(1,2);
      x2_actual=x_act(1,3);
      y2_actual=y_act(1,3);
      x3_actual=x_act(1,4);
      y3_actual=y_act(1,4);
   else

x1_actual=(x_act(index+1,2)-x_act(index,2))*(est2_log(i,1)-((index-1)*sa
mple))/sample+x_act(index,2);

y1_actual=(y_act(index+1,2)-y_act(index,2))*(est2_log(i,1)-((index-1)*sa
mple))/sample+y_act(index,2);

x2_actual=(x_act(index+1,3)-x_act(index,3))*(est2_log(i,1)-((index-1)*sa
mple))/sample+x_act(index,3);

y2_actual=(y_act(index+1,3)-y_act(index,3))*(est2_log(i,1)-((index-1)*sa
mple))/sample+y_act(index,3);

x3_actual=(x_act(index+1,4)-x_act(index,4))*(est2_log(i,1)-((index-1)*sa
mple))/sample+x_act(index,4);

y3_actual=(y_act(index+1,4)-y_act(index,4))*(est2_log(i,1)-((index-1)*sa
mple))/sample+y_act(index,4);
   end
   x1_est=ss*(est2_log(i,2))*(x1_actual/sqrt(x1_actual^2+y1_actual^2));
   y1_est=ss*(est2_log(i,2))*(y1_actual/sqrt(x1_actual^2+y1_actual^2));
   x2_est=ss*(est2_log(i,3))*(x2_actual/sqrt(x2_actual^2+y2_actual^2));
```

```
y2_est=ss*(est2_log(i,3))*(y2_actual/sqrt(x2_actual^2+y2_actual^2));
x3_est=ss*(est2_log(i,4))*(x3_actual/sqrt(x3_actual^2+y3_actual^2));
y3_est=ss*(est2_log(i,4))*(y3_actual/sqrt(x3_actual^2+y3_actual^2));
results2(i,1)=est2_log(i,1);
results2(i,2)=x1_actual;
results2(i,3)=y1_actual;
results2(i,4)=x1_est;
results2(i,5)=y1_est;
results2(i,6)=x2_actual;
results2(i,7)=y2_actual;
results2(i,8)=x2_est;
results2(i,9)=y2_est;
results2(i,10)=x3_actual;
results2(i,11)=y3_actual;
results2(i,12)=x3_est;
results2(i,13)=y3_est;
results2(i,14)=sqrt((x2_actual-x1_actual)^2+(y2_actual-y1_actual)^2); %
actual D12
results2(i,15)=ss*est2_log(i,5); %est. D12
results2(i,16)=sqrt((x3_actual-x1_actual)^2+(y3_actual-y1_actual)^2); %
actual D13
results2(i,17)=ss*est2_log(i,6); %est. D13
results2(i,18)=sqrt((x2_actual-x3_actual)^2+(y2_actual-y3_actual)^2); %
actual D23
results2(i,19)=ss*est2_log(i,7); %est. D23
results2(i,20)=(x3_est+x2_est+x1_est)/3; % estimated Cx
results2(i,21)=(y3_est+y2_est+y1_est)/3; % estimated Cy
results2(i,22)=(x3_actual+x2_actual+x1_actual)/3; % actual Cx
results2(i,23)=(y3_actual+y2_actual+y1_actual)/3; % actual Cy
% calculating the Error in Head angle
if index<=(3/5)*(TIME/sample)
   Theta_r2= atan((y1_est-results2(i,21))/(x1_est-results2(i,20)));
   tangent_r2 =atan((rx-results2(i,20))/(results2(i,21)-ry));
else
   Theta_r2= atan((y1_est-results2(i,21))/(x1_est-results2(i,20)));
   tangent_r2 =atan((rx2-results2(i,20))/(results2(i,21)-ry2));
end
results2(i,28)=difference(1)-(Theta_r2-tangent_r2);

results2(i,24)=sqrt((results2(i,22)-results2(i,20))^2+(results2(i,23)-results2(i
,21))^2); % error in centroid
a_act=sqrt((x2_actual-x1_actual)^2+(y2_actual-y1_actual)^2);
a_est=sqrt((x2_est-x1_est)^2+(y2_est-y1_est)^2);
b_act=sqrt((x3_actual-x1_actual)^2+(y3_actual-y1_actual)^2);
b_est=sqrt((x3_est-x1_est)^2+(y3_est-y1_est)^2);
c_act=sqrt((x3_actual-x2_actual)^2+(y3_actual-y2_actual)^2);
c_est=sqrt((x3_est-x2_est)^2+(y3_est-y2_est)^2);
actual_angle=acos((a_act^2+b_act^2-c_act^2)/(2*a_act*b_act));
if (i<=6)
   est_angle=0;
else
   est_angle=acos((a_est^2+b_est^2-c_est^2)/(2*a_est*b_est));
end
results2(i,25)=(actual_angle-est_angle)*(180/pi); % error in triangle angle
in degrees
results2(i,26)=c_act-c_est ; % error in opposite side of the triangle in mm
results2(i,27)=actual_angle;
end
% averaging errors
index=0;
counter2=floor(counter2/6);
counter2=(counter2-1)*6;
for i=7:6:counter2
   index=index+1;
   av_error2(index,1)=results2(i+5,1); % cycle time
   av_error2(index,2)=results2(i+5,22); % Actual Cx
   av_error2(index,3)=results2(i+5,23); % Actual Cy

av_error2(index,18)=sqrt((results2(i+5,10)-results2(i+5,6))^2+(results2(i+5
,11)-results2(i+5,7))^2)-est2_log(i,13); % error in triad side if
RF-Synchronization

av_error2(index,19)=(180/pi)*(results2(i+5,27)-acos((est2_log(i,11)^2+est
2_log(i,12)^2-est2_log(i,13)^2)/(2*est2_log(i,11)*est2_log(i,12)))); % error
```

```
in triad side if RF-Synchronization

x1_est_RF=(est2_log(i,8))*(results2(i+5,2)/sqrt(results2(i+5,2)^2+results2(
i+5,3)^2));

y1_est_RF=(est2_log(i,8))*(results2(i+5,3)/sqrt(results2(i+5,2)^2+results2(
i+5,3)^2));

x2_est_RF=(est2_log(i,9))*(results2(i+5,6)/sqrt(results2(i+5,6)^2+results2(
i+5,7)^2));

y2_est_RF=(est2_log(i,9))*(results2(i+5,7)/sqrt(results2(i+5,6)^2+results2(
i+5,7)^2));

x3_est_RF=(est2_log(i,10))*(results2(i+5,10)/sqrt(results2(i+5,10)^2+resul
ts2(i+5,11)^2));

y3_est_RF=(est2_log(i,10))*(results2(i+5,11)/sqrt(results2(i+5,10)^2+resul
ts2(i+5,11)^2));
   Cx_EST_RF= (x1_est_RF+x2_est_RF+x3_est_RF)/3;
   Cy_EST_RF= (y1_est_RF+y2_est_RF+y3_est_RF)/3;

av_error2(index,20)=sqrt((results2(i+5,22)-Cx_EST_RF)^2+(results2(i+5,2
3)-Cy_EST_RF)^2); % error in centroid

e1=0;e2=0;e3=0;av_x1=0;av_y1=0;av_x2=0;av_y2=0;av_x3=0;av_y3=0;a
v_angle=0;
   for j=i:i+5
      av_angle=av_angle+results2(j,28);
      av_x1=av_x1+results2(j,2);
      av_y1=av_y1+results2(j,3);
      av_x2=av_x2+results2(j,6);
      av_y2=av_y2+results2(j,7);
      av_x3=av_x3+results2(j,10);
      av_y3=av_y3+results2(j,11);
      e1=e1+results2(j,24);
      e2=e2+results2(j,25);
      e3=e3+results2(j,26);
   end
   av_error2(index,23)=(180/pi)*results2(i+5,28)/6; % error in angle
   av_error2(index,8)=av_x1/6;
   av_error2(index,9)=av_y1/6;
   av_error2(index,12)=av_x2/6;
   av_error2(index,13)=av_y2/6;
   av_error2(index,15)=av_x3/6;
   av_error2(index,16)=av_y3/6;
   av_error2(index,4)=e1/6;
   av_error2(index,5)=e2/6;
   av_error2(index,6)=e3/6;
   if index==1
                                    av_error2(index,7)=
(sqrt((av_error2(index,2)-cx_act(1))^2+(av_error2(index,3)-cy_act(1))^2))/
av_error2(index,1);
                                    av_error2(index,10)=
(sqrt((av_error2(index,8)-x1_act(1))^2+(av_error2(index,9)-y1_act(1))^2))/
av_error2(index,1);
                                    av_error2(index,14)=
(sqrt((av_error2(index,12)-x2_act(1))^2+(av_error2(index,13)-y2_act(1))^2
))/av_error2(index,1);
                                    av_error2(index,17)=
(sqrt((av_error2(index,15)-x3_act(1))^2+(av_error2(index,16)-y3_act(1))^2
))/av_error2(index,1);
   else

av_error2(index,7)=(sqrt((results2(i+5,22)-av_error2(index-1,2))^2+(results
2(i+5,23)-av_error2(index-1,3))^2))/(av_error2(index,1)-av_error2(index-1,
1));

av_error2(index,10)=(sqrt((av_error2(index,8)-av_error2(index-1,8))^2+(a
v_error2(index,9)-av_error2(index-1,9))^2))/(av_error2(index,1)-av_error2
(index-1,1));

av_error2(index,14)=(sqrt((av_error2(index,12)-av_error2(index-1,12))^2+
(av_error2(index,13)-av_error2(index-1,13))^2))/(av_error2(index,1)-av_er
```

```
ror2(index-1,1));

av_error2(index,17)=(sqrt((av_error2(index,15)-av_error2(index-1,15))^2+
(av_error2(index,16)-av_error2(index-1,16))^2))/(av_error2(index,1)-av_er
ror2(index-1,1));
  end
end
index=0;
counter1=floor(counter1/5);
counter1=(counter1-1)*5;
for i=6:5:counter1
  index=index+1;
  av_error1(index,1)=results1(i+4,1); % cycle time
  av_error1(index,23)=(180/pi)*results1(i+4,28); % error in angle
  av_error1(index,2)=results1(i+4,22); % Actual Cx
  av_error1(index,3)=results1(i+4,23); % Actual Cy

av_error1(index,18)=sqrt((results1(i+4,10)-results1(i+4,6))^2+(results1(i+4
,11)-results1(i+4,7))^2)-est1_log(i,13); % error in triad side if
RF-Synchronization

av_error1(index,19)=(180/pi)*(results1(i+5,27)-acos((est1_log(i,11)^2+est
1_log(i,12)^2-est1_log(i,13)^2)/(2*est1_log(i,11)*est1_log(i,12)))); % error
in triad side if RF-Synchronization

x1_est_RF=(est1_log(i,8))*(results1(i+4,2)/sqrt(results1(i+4,2)^2+results1(
i+4,3)^2));

y1_est_RF=(est1_log(i,8))*(results1(i+4,3)/sqrt(results1(i+4,2)^2+results1(
i+4,3)^2));

x2_est_RF=(est1_log(i,9))*(results1(i+4,6)/sqrt(results1(i+4,6)^2+results1(
i+4,7)^2));

y2_est_RF=(est1_log(i,9))*(results1(i+4,7)/sqrt(results1(i+4,6)^2+results1(
i+4,7)^2));

x3_est_RF=(est1_log(i,10))*(results1(i+4,10)/sqrt(results1(i+4,10)^2+resul
ts1(i+4,11)^2));

y3_est_RF=(est1_log(i,10))*(results1(i+4,11)/sqrt(results1(i+4,10)^2+resul
ts1(i+4,11)^2));
  Cx_EST_RF= (x1_est_RF+x2_est_RF+x3_est_RF)/3;
  Cy_EST_RF= (y1_est_RF+y2_est_RF+y3_est_RF)/3;
  if index<=(3/5)*(TIME/sample)
    Theta_r3= atan((y1_est_RF-Cy_EST_RF)/(x1_est_RF-Cx_EST_RF));
    tangent_r3 =atan((rx-Cx_EST_RF)/(Cy_EST_RF-ry));
  else
    Theta_r3= atan((y1_est_RF-Cy_EST_RF)/(x1_est_RF-Cx_EST_RF));
    tangent_r3 =atan((rx2-Cx_EST_RF)/(Cy_EST_RF-ry2));
  end
   av_error2(index,21)=(180/pi)*(difference(1)-(Theta_r3-tangent_r3)); %
error in angle

av_error1(index,20)=sqrt((results1(i+4,22)-Cx_EST_RF)^2+(results1(i+4,2
3)-Cy_EST_RF)^2); % error in centroid
  av_error1(index,8)=results1(i+4,2);
  av_error1(index,9)=results1(i+4,3);
  av_error1(index,12)=results1(i+4,6);
  av_error1(index,13)=results1(i+4,7);
  av_error1(index,15)=results1(i+4,10);
  av_error1(index,16)=results1(i+4,11);
  av_error1(index,4)=results1(i+4,24);
  av_error1(index,5)=results1(i+4,25);
  av_error1(index,6)=results1(i+4,26);
  av_error1(index,25)=results1(i+4,4)-results1(i+4,2); % Error in x1 (round
trip)
  av_error1(index,26)=results1(i+4,5)-results1(i+4,3); % Error in y1 (round
trip)
  av_error1(index,27)=results1(i+4,8)-results1(i+4,6); % Error in x2 (round
trip)
  av_error1(index,28)=results1(i+4,9)-results1(i+4,7); % Error in y2 (round
trip)
 av_error1(index,29)=results1(i+4,12)-results1(i+4,10); % Error in x3 (round
```

```
trip)
  av_error1(index,30)=results1(i+4,13)-results1(i+4,11);%Error in y3 (round
trip)
   if index==1
                                av_error1(index,7)=
(sqrt((av_error1(index,2)-cx_act(1))^2+(av_error1(index,3)-cy_act(1))^2))/
av_error1(index,1);
                                av_error1(index,10)=
(sqrt((av_error1(index,8)-x1_act(1))^2+(av_error1(index,9)-y1_act(1))^2))/
av_error1(index,1);
                                av_error1(index,14)=
(sqrt((av_error1(index,12)-x2_act(1))^2+(av_error1(index,13)-y2_act(1))^2
))/av_error1(index,1);
                                av_error1(index,17)=
(sqrt((av_error1(index,15)-x3_act(1))^2+(av_error1(index,16)-y3_act(1))^2
))/av_error1(index,1);
   else

av_error1(index,7)=(sqrt((results1(i+4,22)-av_error1(index-1,2))^2+(results
1(i+4,23)-av_error1(index-1,3))^2))/(av_error1(index,1)-av_error1(index-1,
1));

av_error1(index,10)=(sqrt((av_error1(index,8)-av_error1(index-1,8))^2+(a
v_error1(index,9)-av_error1(index-1,9))^2))/(av_error1(index,1)-av_error1
(index-1,1));

av_error1(index,14)=(sqrt((av_error1(index,12)-av_error1(index-1,12))^2+
(av_error1(index,13)-av_error1(index-1,13))^2))/(av_error1(index,1)-av_er
ror1(index-1,1));

av_error1(index,17)=(sqrt((av_error1(index,15)-av_error1(index-1,15))^2+
(av_error1(index,16)-av_error1(index-1,16))^2))/(av_error1(index,1)-av_er
ror1(index-1,1));
   end
end
subplot(6,1,1); plot(av_error2(:,2),av_error2(:,3));ylabel('Y(mm)');axis tight
;hold on;
subplot(6,1,2);plot(av_error2(:,2),av_error2(:,4));hold
on;plot(av_error2(:,2),av_error2(:,20),'-.');hold
on;plot(av_error2(:,2),av_error1(:,4),'--');ylabel({'Error in';'centroid
loc.(mm)'});axis tight ;hold on;
subplot(6,1,3);plot(av_error2(:,2),av_error2(:,5));hold
on;plot(av_error2(:,2),av_error2(:,19),'-.');hold
on;plot(av_error2(:,2),av_error1(:,5),'--');ylabel({'Error in';'Triad
angle(deg.)'});axis tight ;hold on;
subplot(6,1,4);plot(av_error2(:,2),av_error2(:,6));hold
on;plot(av_error2(:,2),av_error2(:,18),'-.');hold
on;plot(av_error2(:,2),av_error1(:,6),'--');xlabel('X(mm)');ylabel({'Error
in';'Triad side(mm)'});axis tight;legend('Round-Trip','RF
Synchronization','Tessellation',0);legend('boxoff');hold on;
subplot(6,1,5);plot(av_error2(:,2),av_error2(:,21));hold
on;plot(av_error2(:,2),av_error2(:,23),'-.');hold
on;plot(av_error2(:,2),av_error1(:,23),'--');xlabel('X(mm)');ylabel({'Error
in';'angle between Triad head and the tangent(deg)'});axis
tight;legend('Round-Trip','RF
Synchronization','Tessellation',0);legend('boxoff');hold on;
subplot(6,1,6);plot(av_error2(:,2),av_error1(:,25));hold
on;plot(av_error2(:,2),av_error1(:,26),'-.');hold
on;plot(av_error2(:,2),av_error1(:,27),'--');hold
on;plot(av_error2(:,2),av_error1(:,28),':');hold
on;plot(av_error2(:,2),av_error1(:,29),'-.*');hold
on;plot(av_error2(:,2),av_error1(:,30),'--o');xlabel('X(mm)');ylabel({'Error
in';'robots 1,2,3 position using Round-Trip(mm)'});axis
tight;legend('X1','Y1','X2','Y2','X3','Y3',0);legend('boxoff');hold on;

%plotting actual planned triangluar formation and actual centroid
for i=80:150:length(t)
        subplot(6,1,1); plot ([x1_act(i) x2_act(i)],[y1_act(i)
y2_act(i)],'--x','MarkerSize',4);hold on;
        subplot(6,1,1);plot ([x3_act(i) x2_act(i)],[y3_act(i)
y2_act(i)],'--x','MarkerSize',4);
        subplot(6,1,1);plot ([x1_act(i) x3_act(i)],[y1_act(i)
y3_act(i)],'--x','MarkerSize',4);
  temp_cx=(x1_act(i)+x2_act(i)+x3_act(i))/3;
```

```
 temp_cy=(y1_act(i)+y2_act(i)+y3_act(i))/3;
          subplot(6,1,1);plot   ([x1_act(i)   temp_cx],[y1_act(i)
temp_cy],'o','MarkerSize',3);
end
```

A.2 Low-Level Embedded Firmware

```
;*****************************************************
;File Name:         robot.asm
;Description:       Low-Level Localization Node Firmware
;Author:            Hany Geris
;Date:              February 2008
;Microcontroller:   ATmega128
;*****************************************************
.nolist
.include "m128def.inc"
.list
; thesis
;*******************
; Define Register
;*******************
.def            STEP_COUNT              = R0
                ; To track the no. of receieved times (3/cycle)
.def            ROBOT_ID                = R1
                ; ROBOT ID (Base :0 , Robot 1:1 ,....)
.def            NO_OF_ROBOTS            = R2
                ; identifies the no. of robots in the network (doesn't
                ; include the BASE)
.def            STATUS                          = R3
                ; Used to track the status of the robot in any cycle
.def            DELAY_COMPLETED = R19
                ; To monitor if delay is completed
.def            RECV_COMPLETED  = R29
                ; To monitor if Recv. of BURST is completed
.def            TIMER_SENT_FLAG = R29
                ; To monitor if Recv. of BURST is completed
.def            DELTA_LOW               = R14
.def            DELTA_HIGH              = R15
.def            TEMP                    = R16
.def            TEMP_TIMER0             = R17
.def            TEMP_TIMER1             = R18
.def            DELAY_TIME_HIGH     = R20
.def            DELAY_TIME_LOW      = R21
.def            Timer_1             = R22 ; in x 1000 microsec.
.def            Timer_2             = R23 ; in x 256 x1000 microsec.
.def            Timer_3             = R24 ; in x 256 x 256 x1000 microsec
.def            RCV_COUNTER= R25
.def            DATA1               = R26
.def            DATA0               = R27
.def            SREG_R              = R28
.def            COUNTER             = R7
.def            TOTAL_COUNT= R8
;*************************
;Define Robot/Network Constants
;*************************
.equ    NODE_ID         =1      ;robot (node) ID
.equ    NO_OF_NODES  =3         ;no. of robots (doesn't include base)
.equ    SENSOR_ID   =0          ;Physical Sensor Address
.equ    FAKE_COMMAND =0x5A          ;Fake command to sensor
.equ    SEND_BURST_COMMAND =0x5C
;*************************
;Define Time Constants
;*************************
.equ    BLANK_TIME_LOW      = 0xB8 ; Blank Time =35000 microsec.
.equ    BLANK_TIME_HIGH     =0x88
;*************************
; Define Messages Codes used in the Network Protocol
;*************************
.equ    REQUEST_DELTA       =       0x01
;*******************
; Main Program
;*******************
.CSEG
        .ORG 0 rjmp RESET
        nop
```

```
        .ORG    $0020   RJMP    TIMER0OVF
        nop
        .ORG    OVF1addr        RJMP    TIMER1OVF
; Handle the TIMER 1 time out
        nop
        .ORG    0X0024          RJMP    UART0_RECEIEVE1
; used to handle the recv. Interrupt of uart 0 connected to sensor
        nop
        .ORG    0X003C          RJMP    UART1_RECEIEVE1
; used to handle the recv. Interrupt of uart 1 connected to PC
                        nop

RESET:  LDI             TEMP,LOW(RAMEND)
        ;Init Stack Pointer
        OUT             SPL,TEMP
        LDI             TEMP,HIGH(RAMEND)
        OUT             SPH,TEMP
;Program Start:
        ldi     TEMP, NODE_ID
        ; set the robot id here
        mov     ROBOT_ID , TEMP
        ldi     TEMP, NO_OF_NODES
        mov     NO_OF_ROBOTS,TEMP
; Initiate the UART 0 to connect to the sensor (frame format: 8data, 2stop
; bit, no parity)
        clr     TEMP
        LDI     TEMP,(1<<RXEN0)|(1<<TXEN0)|(1<<RXCIE0)
        out     UCSR0B,TEMP
        clr     TEMP
        LDI     TEMP ,(1<<UCSZ01)|(1<<UCSZ00)|(1<<USBS0)
        STS     UCSR0C,TEMP
        clr     TEMP
        ;Set baud rate 9600 bps
        STS     UBRR0H,TEMP
        LDI     TEMP,103
        out     UBRR0L,TEMP
        clr     TEMP
        LDI     TEMP,(1<<RXEN1)|(1<<TXEN1)|(1<<RXCIE1)
        ;Enable receiver ,transmitter, Receieve interrupt
        STS     UCSR1B,TEMP
        clr     TEMP
        LDI     TEMP ,(1<<UCSZ11)|(1<<UCSZ10)
        STS     UCSR1C,TEMP
        LDI     TEMP,3                  ;Set baud rate 2400 bps
        STS     UBRR1H,TEMP
        LDI     TEMP,64
        STS     UBRR1L,TEMP
        LDI     TEMP ,(1<<U2X1)
        STS     UCSR1A,TEMP
; initiate the long timer (TIMER 0) that accumulates with time
; using 8 bit TIMER (0) with clock 16MHZ and prescaler (64) means every ;
clock=4 microsec.
; if we reload the timer with a value 6 so it will count 250 * 4 ; microsec.=1000
micrsec=1msec.
        in      TEMP_TIMER1, TIMSK
        ldi     TEMP_TIMER1, (1<<TOIE1)|(1<<TOIE0)
        out     TIMSK, TEMP_TIMER1
INIT_CYCLE:     ldi     zl, low (Timer_Log)
                ldi     zh, high(Timer_Log)
                clr     RCV_COUNTER
                sei
                ldi     DATA1,REQUEST_DELTA
                call    SEND_MSG_2_BASE
; wait to receive the delta value and that happen when RCV_COUNTER =1
WAIT_2_RCV_DELTA:
                cpi     RCV_COUNTER, 0x01
                brne    WAIT_2_RCV_DELTA
                ldi     TEMP_TIMER0, 0x05
                out     TCNT0 , TEMP_TIMER0
                clr     Timer_1
                clr     Timer_2
                clr     Timer_3
                ; start the ticking of the accumulative timer (0)
        in      TEMP_TIMER0, TCCR0
```

```
        ldi     TEMP_TIMER0,(1<<CS02)
        out     TCCR0,TEMP_TIMER0
        Mov     STATUS, ROBOT_ID
        Mov     STEP_COUNT , NO_OF_ROBOTS
        inc     STEP_COUNT
STEP:
        mov     TEMP, STEP_COUNT
        cpi     TEMP, 0
        brne    NOT_YET
        rjmp    INIT_CYCLE
NOT_YET:
        mov     TEMP, STATUS
        cpi     TEMP, 0
        brne    RECV_STATE
TRANS_STATE: call   LOG_TIMER
             ldi    DATA0 ,SENSOR_ID
             call   SEND_MSG_2_SENSOR
             ldi    DATA0 , SEND_BURST_COMMAND
             call   SEND_MSG_2_SENSOR
CHECK_CURRENT_STEP:
             mov    TEMP_TIMER1, STEP_COUNT
             dec    TEMP_TIMER1
             cpi    TEMP_TIMER1,0
             brne    CHECK_STATUS
             clr    TIMER_SENT_FLAG
             call   SEND_TIMER_READINGS
WAIT1:
             cpi    TIMER_SENT_FLAG , 0xFF
             brne   WAIT1
             nop
             nop
             nop
             nop
             nop
             nop
             nop
             nop
             nop
             nop
             nop
             ; upload the delay timer by the BLANK Time value
             ldi    DELAY_TIME_HIGH, 0xFF
             ldi    DELAY_TIME_LOW, 0xFF
             ; reset the DELAY_COMPLETED flag
             clr    DELAY_COMPLETED
             call   LOAD_DELAY_TIMER
WAIT33:
             cpi    DELAY_COMPLETED , 0xFF
             brne   WAIT33
             nop
             nop
             nop
             nop
             nop
             nop
        ; As that was the last step in the cycle so branch to the init_cycle
             RJMP INIT_CYCLE

RECV_STATE:  ; it is a Receiver Node
             clr    RECV_COMPLETED
             ; start receiving
             ldi    DATA0 , SENSOR_ID
             call   SEND_MSG_2_SENSOR  ; send sensor id
             ldi    DATA0 , FAKE_COMMAND
             call   SEND_MSG_2_SENSOR
             ; send FAKE Recv command to the sensor
WAIT_RECV:   cpi    RECV_COMPLETED , 0xFF
             brne   WAIT_RECV
             nop
             nop
                    nop
                    nop
                    nop
                    nop
```

```
                        nop
                        brne    WAIT_RECV
; log the Accumulated Timer 0 at this point and send it to the BASE
                        call    LOG_TIMER
                        RJMP CHECK_CURRENT_STEP
CHECK_STATUS:
                        mov     TEMP_TIMER1, STATUS
                        cpi     TEMP_TIMER1, 0
                        brne    CHECK_NEXT_STATUS
; each node after transmitting once, it needs to be switched to be receiver
; for no. of times that can be calculated as follows:
; NO_OF_ROBOTS -ROBOT_ID --> STATUS
                        mov     STATUS , NO_OF_ROBOTS
                        sub     STATUS , ROBOT_ID
                        inc     STATUS
                        RJMP    RV_STATE
CHECK_NEXT_STATUS:
                        mov     TEMP_TIMER1, STATUS
                        dec     TEMP_TIMER1
                        cpi     TEMP_TIMER1, 0
                        brne    RV_STATE
                        mov DELAY_TIME_HIGH, DELTA_HIGH
                        mov DELAY_TIME_LOW, DELTA_LOW
                        clr DELAY_COMPLETED
                        call LOAD_DELAY_TIMER
WAIT22:                 cpi DELAY_COMPLETED , 0xFF
                        brne WAIT22
                        nop
                        nop
                        nop
                        nop
                        nop
                        ; Delay is completed and continue looping
                RJMP    CONTINUE
RV_STATE:       ldi     DELAY_TIME_HIGH, BLANK_TIME_HIGH
                ldi     DELAY_TIME_LOW, BLANK_TIME_LOW
                clr     DELAY_COMPLETED
                call    LOAD_DELAY_TIMER
WAIT3:          cpi     DELAY_COMPLETED , 0xFF
                brne    WAIT3
                nop
                nop
                nop
                nop
CONTINUE:       dec     STEP_COUNT
                dec     STATUS
                rjmp    STEP
END:            RJMP    END
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; UART 0 RECV INTERRUPT HANDLING (Connected to Sensor)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
UART0_RECEIEVE1:
                in      SREG_R, SREG
                clr     TEMP
                SBIS    UCSR0A,RXC0
                RJMP UART0_RECEIEVE1
                ;Receieve the MSB of the range reading
                IN      TEMP,UDR0
                CPI     TEMP, 0
                BRNE RETURN_OK
UART0_RECEIEVE2:
                SBIS    UCSR0A,RXC0
                RJMP    UART0_RECEIEVE2
                ;Receieve the LSB of the range reading
                IN      TEMP,UDR0
                CPI     TEMP , 0
                BRNE    RETURN_OK
                ; if it returns zero , jmp to resend FAKE recv command
                rjmp    RESEND
RETURN_OK:
                ser     RECV_COMPLETED
                ; log the TIMER (0) reading (code to be added here)
                rjmp    RETURN
RESEND:         ldi     DATA0 , SENSOR_ID
```

```
TX_BYTE00:
                SBIS    UCSR0A,UDRE0
                RJMP    TX_BYTE00
                out     UDR0,DATA0
                ldi     DATA0, FAKE_COMMAND
TX_BYTE01:
                SBIS    UCSR0A,UDRE0
                RJMP    TX_BYTE01
                out     UDR0,DATA0
RETURN:         out     SREG,SREG_R
                reti
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; UART 1 RECV INTERRUPT HANDLING (Connected to PC)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
UART1_RECEIEVE1:
                in      SREG_R,SREG
                LDS     TEMP,UCSR1A
                SBRS    TEMP,RXC1
                RJMP    UART1_RECEIEVE1
DELTA1:         LDS     DELTA_LOW,UDR1
                ;look for the MSB
UART1_RECEIEVE2:
                LDS     TEMP,UCSR1A
                SBRS    TEMP,RXC1
                RJMP    UART1_RECEIEVE2
DELTA2:         LDS     DELTA_HIGH,UDR1
                inc     RCV_COUNTER
BACK:           out     SREG,SREG_R
                reti
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; ACCUMULATIVE TIMER INTERRUPT (TIMER 0)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

TIMER0OVF:      sei
                in      SREG_R,SREG
                ; stop ticking
                ldi     TEMP_TIMER0,(0<<CS02)
                out     TCCR0,TEMP_TIMER0
                ; reload the initial value of the timer
                ldi     TEMP_TIMER0, 0x05
                out     TCNT0 , TEMP_TIMER0
                ldi     TEMP_TIMER0,0x01
                clc
                add     Timer_1,TEMP_TIMER0
                clr     TEMP_TIMER0
                adc     Timer_2,TEMP_TIMER0
                adc     Timer_3,TEMP_TIMER0
                ; start ticking again
                ldi     TEMP_TIMER0,(1<<CS02)
                out     TCCR0,TEMP_TIMER0
                out     SREG,SREG_R
                reti
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Delay TIMER INTERRUPT (TIMER 1)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TIMER1OVF:      sei
                in      SREG_R,SREG
                clr     TEMP
                out     TCCR1B, TEMP
                in      TEMP_TIMER1, TIMSK
                ldi     TEMP_TIMER1, (0<<TOIE1)|(1<<TOIE0)
                out     TIMSK, TEMP_TIMER1
                ser     DELAY_COMPLETED
                out     SREG,SREG_R
                reti
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;sub to Load the Delay Timer with the appropiate Delay Time
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
LOAD_DELAY_TIMER:

                        clc
                        lsr     DELAY_TIME_HIGH
                        ror     DELAY_TIME_LOW
                        clc
```

```
                lsr     DELAY_TIME_HIGH
                ror     DELAY_TIME_LOW
                clc
                com     DELAY_TIME_LOW
                com     DELAY_TIME_HIGH
                ldi     TEMP, 1
                add     DELAY_TIME_LOW , TEMP
                clr     TEMP
                adc     DELAY_TIME_HIGH , TEMP
                out     TCNT1H , DELAY_TIME_HIGH
                out     TCNT1L , DELAY_TIME_LOW
                ; start Timer 1 to tick
                in      TEMP_TIMER1, TIMSK
ldi     TEMP_TIMER1, (1<<TOIE1)|(1<<TOIE0)
                out     TIMSK, TEMP_TIMER1
                in              TEMP_TIMER1, TCCR1B
ldi     TEMP_TIMER1,(1<<CS11)|(1<<CS10)
                out     TCCR1B,TEMP_TIMER1
                ret
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; sub to log the Accumulated timer 0
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
LOG_TIMER:      ; stop ticking
                clr     TEMP_TIMER0
                ldi     TEMP_TIMER0,(0<<CS02)
                out     TCCR0,TEMP_TIMER0
                in      DATA1,TCNT0
                st      z+, DATA1
                st      z+,Timer_1
                st      z+,Timer_2
                st      z+,Timer_3
                ; start ticking
                clr TEMP_TIMER0
        ldi     TEMP_TIMER0,(1<<CS02)
        out     TCCR0,TEMP_TIMER0
        ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; sub to send the TIMER LOG to the BASE at the end of each cycle
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SEND_TIMER_READINGS:
                ldi     zl, low (Timer_Log)
                ldi     zh, high(Timer_Log)
                clr     COUNTER
                clr     TOTAL_COUNT
                mov     TOTAL_COUNT, NO_OF_ROBOTS
                inc     TOTAL_COUNT
                clc
                rol     TOTAL_COUNT
                clc
                rol     TOTAL_COUNT
REPEAT:         cp      COUNTER, TOTAL_COUNT
                breq    DONE
                LD      DATA1,z+
                cpi     DATA1, 255
                brcc     SEND_255
                cpi     DATA1,128
                brcc    SEND_GE_127
                call    SEND_MSG_2_BASE
                ldi     DATA1,0
                call    SEND_MSG_2_BASE
                ldi     DATA1,0
                call    SEND_MSG_2_BASE
                jmp     GO
SEND_255:       ldi     DATA1, 1
                call    SEND_MSG_2_BASE
                ldi     DATA1,127
                call    SEND_MSG_2_BASE
                ldi     DATA1,127
                call    SEND_MSG_2_BASE
                jmp     GO
SEND_GE_127:
                subi    DATA1,127
                call    SEND_MSG_2_BASE
```

```
                ldi     DATA1,127
                call    SEND_MSG_2_BASE
                ldi     DATA1,0
                call    SEND_MSG_2_BASE
GO:             inc     COUNTER
                rjmp    REPEAT
DONE:           ser     TIMER_SENT_FLAG
                ret
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; sub to send one byte to SENSOR
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SEND_MSG_2_SENSOR:
        TX_BYTE0:       SBIS    UCSR0A,UDRE0
                        RJMP    TX_BYTE0
                        out     UDR0,DATA0
                        ret
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Sub to send one byte to the BASE
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SEND_MSG_2_BASE:
        TX_BYTE1:       LDS     TEMP,UCSR1A
                        SBRS    TEMP,UDRE1
                        RJMP    TX_BYTE1
                        STS     UDR1,DATA1
                        ret
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
.dseg
Timer_Log: .byte  100
```

A.3 High-Level Human-Machine Interface

```
Option Explicit
Private Declare Sub CopyMemory Lib "kernel32" _
Alias "RtlMoveMemory" (Destination As Any, Source As Any, ByVal _
Length As Integer)
Dim SocketCounter As Integer
Public flag As Boolean
Public cyclic As Boolean
Public delay_flag As Boolean
Private Sub Command1_Click()
cyclic = False
flag = True
End Sub

Private Sub Command2_Click()
flag = False
cyclic = True
End Sub

Private Sub Form_Load()
Dim i As Integer, j As Integer
Dim max_distance(No_Of_Robots + 1) As Double
flag = False
cyclic = False
'initialize the serial port
MSComm1.CommPort = 1
MSComm1.RThreshold = 1
MSComm1.InputLen = 1    ' use 'on comm' event processing
MSComm1.Settings = "2400,n,8,1"        ' baud, parity, data bits, stop bits
MSComm1.SThreshold = 1                 ' allows us to track Tx LED
MSComm1.InputMode = comInputModeBinary
MSComm1.PortOpen = True
' initialize the socket 1, 2, 3
Winsock1.Protocol = sckUDPProtocol
Winsock1.LocalPort = "1223"
Winsock1.Bind Winsock1.LocalPort

Winsock2.Protocol = sckUDPProtocol
Winsock2.LocalPort = "1224"
Winsock2.Bind Winsock2.LocalPort
Winsock3.Protocol = sckUDPProtocol
Winsock3.LocalPort = "1225"
Winsock3.Bind Winsock3.LocalPort
Timer2.Enabled = False
status(0).FillColor = &H8000&
For i = 1 To No_Of_Robots
 status(i).FillColor = &HC0FFC0
Next
Erase counter, ready, timers, timers_index, distance, old_distance, delta,
old_delta, distance_tracker
previous_max = 450
For i = 0 To No_Of_Robots
   For j = 0 To No_Of_Robots
      If j = i Then
         delta(j, i) = Blank_time + (previous_max / sound_speed)
         distance(j, i) = 0
      Else
         delta(j, i) = Blank_time
         distance(j, i) = 0
      End If
   Next
Next
For i = 0 To No_Of_Robots
   counter(i) = 0 'initiate the counter of each robot initially
   ready(i) = 0 'initiate the ready to calculate the Distance FLAG array
Next
Track_Cycles = 0
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
Winsock1.Close
Winsock2.Close
Winsock3.Close
End Sub

Private Sub MSComm1_OnComm()
Dim recv_temp() As Byte
Dim str As String
Dim j As Integer, temp0 As Integer, temp As Integer, i As Integer
Dim dat0_out(1) As Byte
Dim bytArr() As Byte
Dim rr(1) As Byte
rr(0) = 0
Dim temp_long0 As Long

Select Case MSComm1.CommEvent
Case comEvReceive
   recv_temp = MSComm1.Input
   temp_long0 = recv_temp(0)
   If ready(0) = 0 Then
      If counter(0) = 0 Then
        recv_queue(0) = recv_temp(0)
      Else
        If counter(0) <= No_Of_Robots + 1 Then
        ' check which byte in each phase for each robot (index)
         timers_index(0) = timers_index(0) + 1
         If timers_index(0) = 5 Then
           timers_index(0) = 1
           counter(0) = counter(0) + 1 'move to next phase
         End If
        ' if it is equal 1 so, recv_byte * 4
        'if it is equal 2 so, recv_byte * 1000 microsec
        'if it is equal 3 so, recv_byte *256*1000
        'if it is equal 4 so , recv_byte *256*256*1000
        If timers_index(0) = 1 Then
           timers(0, counter(0)) = timers(0, counter(0)) + temp_long0 * 4
           'send it back for hand shake
           MSComm1.Output = rr()
        End If
If timers_index(0) = 2 Then
          timers(0, counter(0)) = timers(0, counter(0)) + temp_long0 * 1000
          'send it back for hand shake
          MSComm1.Output = rr()
End If
If timers_index(0) = 3 Then
   timers(0, counter(0)) = timers(0, counter(0)) + temp_long0 * 256 * 1000
   'send it back for hand shake
    MSComm1.Output = rr()
End If
    If timers_index(0) = 4 Then
        timers(0, counter(0)) = timers(0, counter(0)) + temp_long0 * 256 *
                256 * 1000
         'send it back for hand shake
         MSComm1.Output = rr()
     End If
     If (timers_index(0) * counter(0)) = (4 * (No_Of_Robots + 1)) Then
'reaching this point implies that we already received all readings for the
'BASE after completing one cycle
      ready(0) = 1
       End If
      End If
    End If
 Else
   ' buffer the delta request received byte to check it later after finishing last
   ' cycle
   recv_queue(0) = recv_temp(0)
  End If
 End Select
End Sub

Private Sub Timer1_Timer()
Dim max_distance As Double
Dim i As Integer, temp As Integer, j As Integer
```

```
Dim dat0_out(1) As Byte
Dim bytArr() As Byte
Dim compare As Boolean
Dim s As String
' check if all bytes are recieved to start the calculation of the DISTANCE
' and Delta Arrays
temp = 0
For i = 0 To No_Of_Robots
   temp = temp + ready(i)
Next
If (temp = No_Of_Robots + 1) Then
   ' add code here to calculate the Distance matrix
   ' as a test to verify the validity of this code only the base and one robot
   distance(0, 1) = sound_speed * ((timers(0, 2) - timers(0, 1) - (timers(1, 2)
- timers(1, 1)))) / 2 - 209
   distance(1, 2) = sound_speed * ((timers(1, 3) - timers(1, 2) - (timers(2, 3)
- timers(2, 2)))) / 2 - 231
   distance(2, 3) = sound_speed * ((timers(2, 4) - timers(2, 3) - (timers(3, 4)
- timers(3, 3)))) / 2 - 209
   distance(0, 2) = sound_speed * ((timers(0, 3) - timers(0, 1) - (timers(2, 3)
- timers(2, 1))) / 2) - 226
   distance(1, 3) = sound_speed * ((timers(1, 4) - timers(1, 2) - (timers(3, 4)
- timers(3, 2))) / 2) - 208
   distance(0, 3) = sound_speed * (timers(0, 4) - timers(0, 1) - (timers(3, 4) -
timers(3, 1))) / 2 - 195
   'distance(0, 3) = sound_speed * (timers(0, 4) - timers(0, 1) - (timers(3, 4) -
timers(3, 3)) - (timers(2, 3) - timers(2, 1))) - distance(2, 3) - distance(0, 2) -
712
   distance(1, 0) = distance(0, 1)
   distance(2, 0) = distance(0, 2)
   distance(3, 0) = distance(0, 3)
   distance(2, 1) = distance(1, 2)
   distance(3, 1) = distance(1, 3)
   distance(3, 2) = distance(2, 3)
   distance_tracker(Track_Cycles, 1) = distance(0, 1)
   distance_tracker(Track_Cycles, 2) = distance(0, 2)
   distance_tracker(Track_Cycles, 3) = distance(0, 3)
   distance_tracker(Track_Cycles, 4) = distance(1, 2)
   distance_tracker(Track_Cycles, 5) = distance(1, 3)
   distance_tracker(Track_Cycles, 6) = distance(2, 3)

   For i = 0 To No_Of_Robots
   ' for each Robot i ,we will look for the far distance between i-1  and all
    ' robots
    max_distance = 0
    For j = 0 To No_Of_Robots
      If max_distance < distance(i, j) Then
        max_distance = distance(i, j)
      End If
    Next
    If max_distance > 450 Then
       max_distance = 250
    ElseIf max_distance < 0 Then
         max_distance = 250
    End If
    For j = 0 To No_Of_Robots
      If j = i Then
         delta(j, i) = 214 + Blank_time + (300 / sound_speed)
         distance(j, i) = 0
      Else
         delta(j, i) = 214 + Blank_time
         distance(j, i) = 0
      End If
    Next
   Next
   'display the estimated distances between nodes in list box
     s = Math.Round(distance_tracker(Track_Cycles, 1), 1) & vbTab &
Math.Round(distance_tracker(Track_Cycles, 2), 1) & vbTab &
Math.Round(distance_tracker(Track_Cycles, 3), 1) & vbTab &
Math.Round(distance_tracker(Track_Cycles, 4), 1) & vbTab &
Math.Round(distance_tracker(Track_Cycles, 5), 1) & vbTab &
Math.Round(distance_tracker(Track_Cycles, 6), 1)
   List1.AddItem (s)
   Track_Cycles = Track_Cycles + 1
```

```
   If (Track_Cycles = 1000) Then
    Track_Cycles = 0
    End If
   If cyclic = False Then
     compare = True
   Else
     compare = False
   End If
   Do Until flag = compare
    DoEvents
    Loop
    If cyclic = False Then
        flag = False
   End If
   ' disable timer 1 inorder to keep the sequence
   Timer1.Enabled = False
   'introduce delay before checkin all nodes recieved delta requests
    delay_flag = False
    Timer2.Interval = 70
    Timer2.Enabled = True
    Do While (Not delay_flag)
         DoEvents
    Loop
    Timer2.Enabled = False
   ' now check if all delta requests are recieved or not
    temp = 0
For i = 0 To No_Of_Robots
    temp = temp + recv_queue(i)
Next
temp = No_Of_Robots + 1
If (temp = No_Of_Robots + 1) Then
  'reset the timers matrix
   For i = 0 To No_Of_Robots
      For j = 0 To No_Of_Robots + 1
        timers(i, j) = 0
      Next
      recv_queue(i) = 0
 Next
For i = 1 To No_Of_Robots
    'parse the delta to send it as 2 consecutive bytes
    bytArr = IntToByteArray(delta(i, i))
    'send the DELTA via the socket to the robot
    If i = 1 Then
      Winsock1.SendData bytArr
    End If
    If i = 2 Then
      Winsock2.SendData bytArr
    End If
    If i = 3 Then
      Winsock3.SendData bytArr
    End If
    counter(i) = 1 ' prepare for the next recieving phase
    timers_index(i) = 0 ' reset the index for 4 bytes timer reading
    ready(i) = 0
Next
    'introduce delay before initiating the base
    delay_flag = False
    Timer2.Interval = 100
    Timer2.Enabled = True
    Do While (Not delay_flag)
      DoEvents
    Loop
    Timer2.Enabled = False
    recv_queue(0) = 0
    'parse the delta to send it as 2 consecutive bytes
     bytArr = IntToByteArray(delta(0, 0))
     ' send the DELTA via the serial port to the BASE
     dat0_out(0) = bytArr(0)
     dat0_out(1) = bytArr(1)
     MSComm1.Output = dat0_out()
     counter(0) = 1 ' prepare for the next recieving phase
     timers_index(0) = 0 ' reset the index for 4 bytes timer reading
     ready(0) = 0
End If
```

```
End If
If Track_Cycles = 0 Then
 Timer1.Enabled = False
 temp = 0
 For i = 0 To No_Of_Robots
  temp = temp + recv_queue(i)
 Next
 If (temp = No_Of_Robots + 1) Then
  For i = 1 To No_Of_Robots
     recv_queue(i) = 0
    'parse the delta to send it as 2 consecutive bytes
     bytArr = IntToByteArray(delta(i, i))
   'send the DELTA via the socket to the robot
    If i = 1 Then
      Winsock1.SendData bytArr
    End If
    If i = 2 Then
      Winsock2.SendData bytArr
    End If
    If i = 3 Then
      Winsock3.SendData bytArr
    End If
    counter(i) = 1 ' prepare for the next recieving phase
    timers_index(i) = 0 ' reset the index for 4 bytes timer reading
  Next
   'introduce delay before initiating the base
   delay_flag = False
   Timer2.Interval = 100
   Timer2.Enabled = True
   Do While (Not delay_flag)
     DoEvents
   Loop
   Timer2.Enabled = False
   recv_queue(0) = 0
  'parse the delta to send it as 2 consecutive bytes
   bytArr = IntToByteArray(delta(0, 0))
   ' send the DELTA via the serial port to the BASE
    dat0_out(0) = bytArr(0)
    dat0_out(1) = bytArr(1)
    MSComm1.Output = dat0_out()
    counter(0) = 1 ' prepare for the next recieving phase
    timers_index(0) = 0 ' reset the index for 4 bytes timer reading
  End If
End If
Timer1.Enabled = True
End Sub

Private Sub Timer2_Timer()
  delay_flag = True
End Sub

Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
'This is being trigger every time new data arrive
Dim recv_byte As Byte
Dim dat_in() As Byte, dat_out(1) As Byte
Dim bytArr() As Byte, rr(1) As Byte
Dim temp_long1 As Long
status(1).FillColor = &HC000&
Winsock1.GetData dat_in 'writes the new data in our string dat ( string format
)
recv_byte = dat_in(0)
temp_long1 = recv_byte
If ready(1) = 0 Then
    If counter(1) = 0 Then
      recv_queue(1) = recv_byte
    Else
    If counter(1) <= No_Of_Robots + 1 Then
    ' check which byte in each phase for each robot (index)
      timers_index(1) = timers_index(1) + 1
      If timers_index(1) = 5 Then
        timers_index(1) = 1
        counter(1) = counter(1) + 1 'move to next phase
      End If
      ' if it is equal 1 so, recv_byte * 4
```

```
        'if it is equal 2 so, recv_byte * 1000 microsec
        'if it is equal 3 so, recv_byte *256*1000
        'if it is equal 4 so , recv_byte *256*256*1000
        If timers_index(1) = 1 Then
           timers(1, counter(1)) = timers(1, counter(1)) + temp_long1 * 4
        End If
        If timers_index(1) = 2 Then
          timers(1, counter(1)) = timers(1, counter(1)) + temp_long1 * 1000
        End If
        If timers_index(1) = 3 Then
  timers(1, counter(1)) = timers(1, counter(1)) + temp_long1 * 256 * 1000
        End If
        If timers_index(1) = 4 Then
           timers(1, counter(1)) = timers(1, counter(1)) + temp_long1 * 256
                                                          * 256 * 1000
        End If
        If (timers_index(1) * counter(1)) = (4 * (No_Of_Robots + 1)) Then
           ready(1) = 1
        End If
     End If
   End If
Else
 recv_queue(1) = recv_byte
End If

End Sub

Public Function IntToByteArray(ByVal long_num As Long) As Byte()
Dim ByteArray(0 To 1) As Byte
CopyMemory ByteArray(0), ByVal VarPtr(long_num), 2
IntToByteArray = ByteArray
End Function

Private Sub Winsock1_Error(ByVal Number As Integer, Description As
String, ByVal Scode As Long, ByVal Source As String, ByVal HelpFile As
String, ByVal HelpContext As Long, CancelDisplay As Boolean)
Dim dd As Integer
dd = 1
End Sub

Private Sub Winsock2_DataArrival(ByVal bytesTotal As Long)
'This is being trigger every time new data arrive
Dim recv_byte As Byte
Dim dat_in() As Byte, dat_out(1) As Byte
Dim bytArr() As Byte, rr(1) As Byte
Dim temp_long1 As Long
status(2).FillColor = &HC000&
Winsock2.GetData dat_in 'writes the new data in our string dat ( string format
)
recv_byte = dat_in(0)
temp_long1 = recv_byte
If ready(2) = 0 Then
     If counter(2) = 0 Then
        recv_queue(2) = recv_byte
     Else
     If counter(2) <= No_Of_Robots + 1 Then
     ' check which byte in each phase for each robot (index)
        timers_index(2) = timers_index(2) + 1
        If timers_index(2) = 5 Then
           timers_index(2) = 1
           counter(2) = counter(2) + 1 'move to next phase
        End If
        ' if it is equal 1 so, recv_byte * 4
        'if it is equal 2 so, recv_byte * 1000 microsec
        'if it is equal 3 so, recv_byte *256*1000
        'if it is equal 4 so , recv_byte *256*256*1000
        If timers_index(2) = 1 Then
           timers(2, counter(2)) = timers(2, counter(2)) + temp_long1 * 4
        End If
        If timers_index(2) = 2 Then
          timers(2, counter(2)) = timers(2, counter(2)) + temp_long1 * 1000
        End If
        If timers_index(2) = 3 Then
  timers(2, counter(2)) = timers(2, counter(2)) + temp_long1 * 256 * 1000
```

```
        End If
        If timers_index(2) = 4 Then
timers(2, counter(2)) = timers(2, counter(2)) + temp_long1 * 256 * 256 *
                                                                1000

        End If
        If (timers_index(2) * counter(2)) = (4 * (No_Of_Robots + 1)) Then
          ready(2) = 1
        End If
      End If
    End If
Else
   ' buffer the delta request received byte to check it later after finishing last
cycle
  recv_queue(2) = recv_byte
End If
End Sub

Private Sub Winsock2_Error(ByVal Number As Integer, Description As
String, ByVal Scode As Long, ByVal Source As String, ByVal HelpFile As
String, ByVal HelpContext As Long, CancelDisplay As Boolean)
Dim dd As Integer
dd = 1
End Sub

Private Sub Winsock3_DataArrival(ByVal bytesTotal As Long)
'This is being trigger every time new data arrive
Dim recv_byte As Byte
Dim dat_in() As Byte, dat_out(1) As Byte
Dim bytArr() As Byte, rr(1) As Byte
Dim temp_long1 As Long
status(3).FillColor = &HC000&
Winsock3.GetData dat_in 'writes the new data in our string dat ( string format
)
recv_byte = dat_in(0)
temp_long1 = recv_byte
If ready(3) = 0 Then
  If counter(3) = 0 Then
    recv_queue(3) = recv_byte
  Else
  If counter(3) <= No_Of_Robots + 1 Then
  ' check which byte in each phase for each robot (index)
    timers_index(3) = timers_index(3) + 1
    If timers_index(3) = 5 Then
      timers_index(3) = 1
      counter(3) = counter(3) + 1 'move to next phase
    End If
    ' if it is equal 1 so, recv_byte * 4
    'if it is equal 2 so, recv_byte * 1000 microsec
    'if it is equal 3 so, recv_byte *256*1000
    'if it is equal 4 so , recv_byte *256*256*1000
    If timers_index(3) = 1 Then
      timers(3, counter(3)) = timers(3, counter(3)) + temp_long1 * 4
    End If
    If timers_index(3) = 2 Then
      timers(3, counter(3)) = timers(3, counter(3)) + temp_long1 * 1000
    End If
    If timers_index(3) = 3 Then
timers(3, counter(3)) = timers(3, counter(3)) + temp_long1 * 256 * 1000
    End If
    If timers_index(3) = 4 Then
      timers(3, counter(3)) = timers(3, counter(3)) + temp_long1 * 256
                                                    * 256 * 1000
    End If
    If (timers_index(3) * counter(3)) = (4 * (No_Of_Robots + 1)) Then
      ready(3) = 1
    End If
  End If
 End If
Else
  recv_queue(3) = recv_byte
End If
End Sub
Private Sub Winsock3_Error(ByVal Number As Integer, Description As
```

```
String, ByVal Scode As Long, ByVal Source As String, ByVal HelpFile As
String, ByVal HelpContext As Long, CancelDisplay As Boolean)
Dim dd As Integer
dd = 1
End Sub
```