Coordination of Multiple Mobile Robots

by

Muhammad Fudzail Bin Razlan

12733

Dissertation submitted in partial fulfillment of

the requirement for the

Bachelor of Engineering (Hons)

(Mechanical Engineering)

MAY 2013

Universiti Teknologi PETRONAS,
Bandar Seri Iskandar,
31750 Tronoh,
Perak, Darul Ridzuan

CERTIFICATION OF APPROVAL

Coordination of Multiple Mobile Robots

by

Muhammad Fudzail Bin Razlan

A project dissertation submitted to the
Mechanical Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(MECHANICAL ENGINEERING)

Approved by,

____
(Prof Dr Nagarajan Thirumalaiswamy)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

May 2013

CERTIFICATION OF ORIGINALITY

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

____

MUHAMMAD FUDZAIL BIN RAZLAN

ACKNOWLEDGEMENT

First and foremost, thanks to God the Almighty for the chance and blessing given to me. I would like to say thank you to my supervisor dedicated and caring supervisor, Dr. T. Nagarajan for the support, time, guidance and knowledge in supervising me that leads to the completion of my project. Special thanks to Ashwin Narayan, a friend and a great contributor especially in designing control program. Also I would like to extend my gratitude to friends who kindly support and involve directly and indirectly in my project. Lastly, I would to like to take this opportunity to give my thanks to my parents and family members, who gave encouragement to me to continue my study until completion.

ABSTRACT

In recent years, the demand for mobile robots to perform more complex tasks is increasing and a more complex maneuverability is required. Such task can be completed by utilizing a group of mobile robots, in a coordinate manner. To coordinate multiple mobile robots, formation control is required. Formation control refers to the ability to control the relative position and orientation of robots in a group, while allowing them to move as a whole. For this project, leader and follower formation control method is chosen. The objective of this project is to coordinate multiple mobile robots to perform specified trajectory while maintaining spatial distance between leader robot and followers. Three AmigoBot mobile robots are used to carry out the experiments. There are three experiments which are experiment 1; 10m straight line trajectory path, experiment 2; 6m zigzag shaped trajectory path and experiment 3; 10m straight line with obstacles trajectory path. Specified trajectory path plan are developed and given to the leader robot and the follower will move relative to the leader coordinates using the designed control program. Deviation error for the spatial distance is recorded. The mobile robots successfully maintain formation, with spatial distance deviation error less than 5%.

Contents

LIST OF FIGURE

LIST OF TABLES

CHAPTER 1:

INTRODUCTION

1.1 Background of Study

Mobile robots are defined as a system with total mobility relative to environment, perception ability and certain level of autonomy which has limited human interaction. Mobile robots are capable of maneuvering the environment with sensors such as sonar, light sensor, laser sight and others without depending on human control. Mobile robots application has expanded to various areas which include deep sea and space exploration, hazardous environment investigation and military surveillance. These areas require extensive exploration and accurate data collection as part of their research and many are beyond human body capability and perception. Mobile robots allows researcher to explore uncharted environment, mapping the area while collecting relevant data.

In recent years, the demand for mobile robots to perform more complex tasks is increasing and a more complex maneuverability is required. Such task can be completed by utilizing a group of mobile robots, in a coordinate manner. Coordination between multiple mobile robots is required to ensure that the mobile robots stay in formation and move as whole while avoiding obstacles and each others to achieve the goal. A group of robots can provide data redundancy and contribute cooperatively to perform given task with greater reliability, speed and cost reduction compared to a single mobile robots. If there are two or more mobile robots detecting an obstacle in their path, the probability of such of obstacle is present is higher as compared to a single robot.

Many problems arise from keeping the robots in formations which include the difficulty in planning, navigating and coordinating the motion of the robots while avoiding obstacles and each others to achieve the goal. Formation control requires a more elaborate control structure which not only allows the robots to navigate through the environment but also allow them to communicate with each other.

There are three main methods to formation control for multiple robots which are behavioral, virtual structure and leader-following method. This project will be focusing on the leader-following approach. In leader-follower formation approach, some of the robots act as leaders in the group while the others as followers following the leader’s movement. The followers are required to maintain a constant spatial distance with respect to the leader as the mobile robots navigate the environment. Advantages of this approach are that it is easy to understand and implement and the formation can still be maintained even if the leader is disturbed. The disadvantages is that there no explicit feedback from followers to the leader especially in dynamically changing, unknown, unstructured environments.

The main objective of this project is to coordinate multiple mobile robots to perform specified trajectory while maintaining spatial distance between leader robot and followers. Experiments will be conducted using 3 AmigoBot mobile robots. Using the leader- following approach, the multiple mobile robots will navigate the environment using specified trajectory path. At the end of the experiments, the spatial distance of the robots is recorded any deviation error will be noted.

1.2 Problem Statement

Mobile robots application has expanded to various areas which include underwater, space exploration and hazardous environment. These areas requires a more complex maneuver to complete given task and with multiple robots coordination, such maneuver can be done. Many problems arise from keeping the robots in formations which include planning, navigating and coordinating the motion of the robots while avoiding obstacles and each others to achieve the goal. The robots are required to move around the environment as whole while maintaining their formation.

1.3 Objective

1. To coordinate multiple mobile robots to perform specified trajectory while maintaining spatial distance between leader robot and followers.

1.4 Scope of Study

The study will be conducted by experiments using three AmigoBot mobile robots. Specified trajectory path plan will be given to the three robots and data on the relative position of the follower robots with respect to the leader are analyze to find any deviation error.

CHAPTER 2:

LITERATURE REVIEWS

2.1 Multiple Mobile Robots

In recent decade, mobile robots application has expanded to various areas which include underwater, space exploration and hazardous environment. The demand for mobile robots to perform more complex tasks is increasing and with that, coordination between multiple mobile robots will allow such task to be completed. A group of robots can provide data redundancy and contribute cooperatively to perform given task with greater reliability, speed and cost reduction compared to a single mobile robots (Ota, J., Arai, T., and Asama, H. 2002).

To coordinate multiple mobile robots, formation control is required. Formation control refers to the ability to control the relative position and orientation of robots in a group, while allowing them to move as a whole as described by Kuppan Chetty R.M., et al, (2011). Kuo-Yang Tu and Min-Tzung Chang Chien (n.d.) added that formation control is also how multiple mobile robots maintain formation to overcome unexpected events in crucial situation to finish task given.

2.2 Pattern Formation

According to Bahceci, E., Soysal O. and Sahin E., (2003), pattern formation refers to the coordination of a group of mobile robot to get into and maintain formation with a certain shape. There are two groups of pattern formations which are centralized pattern formation and distributed pattern formation.

In centralized pattern formation method, the group motion will be planned by a single computational unit as it oversees the others as stated by Belta C. and Kumar V., (2002). Each robot will then receive their respective movement via communication network. Centralized patter formation depends on a single central unit that oversee the entire group with communication channel between central unit and other robots as the receiver which makes it more costly, less robust to failures, and less scalable to the control of large number of robots.

Decentralized pattern formation uses local communication and sensor for each robots and it tends to be more scalable, robust and easier to build but limited in variety and precision of formations according to Balch T. and Hybinette M (2000) and Dudenhoeffer D.D. and Jones M.P., (2000). This pattern applies on multi robot system. Multi-robot system are preferred compared over a single central robot in centralized pattern due to the robustness and it can be improved by incorporating adaptation mechanisms that responds to change in environment and individual robot capabilities.

2.3 Formation Control Methods

According to Gomman J., et al, (2009), there are three main methods to formation control for multiple robots each with their own advantage and disadvantage. The three main methods are behavioral, virtual structure and leader-following.

Behavioral approach is define as a set of desired behaviors for each individual in the group, and measures them such that desirable group behavior emerges without explicit model of the subsystem or the environment. Balch T. and Arkin R.C. (1998), state that in multi robot system, behavioral approach is described as implementation of others navigational behaviors on the formation to derive control strategies for goal seeking, collision avoidance and formation maintenance. The advantage for this approach is that it is natural to derive control strategies when vehicles have multiple objectives, and a clear feedback is included through communication between group members. The disadvantages are that the group behavior cannot be explicitly defined and it is difficult to analyze the approach mathematically and guarantee the group stability according to Gomman J., et al, (2009).

For virtual structure approach, the whole formation is described as a single, virtual, structure and it acts as a unit. The governing control for each individual robot is derived by defining the motion of the virtual structure and interpreted it into the desired movement for each mobile robot as stated by Do, K.D (2007). Beard R., Lawton, J. and Hadaegh, (1999), mentioned that virtual structure has been achieved by having all members of the formation tracking assigned nodes which move into required configuration. Advantage of the virtual structure approach is that is easy to follow the coordinated behavior for the group and information can be maintained very well during maneuvers. The disadvantage is that the formation has to maintain the exact same virtual structure all the time limiting the application's potential.

In leader-follower formation approach, some of the robot act as leaders in the group while the others as followers following the leader's movement as described by Wang P.K.C., (1992). Kuppan Chetty, R.M., et al., (2011), added that the main objective of the robot acting as the follower is to position themselves relative to the leader and maintain a required distance and orientation among each others. Complex formation is possible to achieve by controlling relative positions and orientation of the leader and followers. Advantages of this approach are that it is easy to understand and implement and the formation can still be maintained even if the leader is disturbed. The disadvantages is that there no explicit feedback from followers to the leader especially in dynamically changing, unknown, unstructured environments.

2.4 Layers Architecture for Leader Follower Formation

Figure 1 shows the hybrid formation control strategy consisting of layered behavior based architecture. The lower level consists of navigation control for robots and the supervisor level is a decentralized leader-follower formation controller. Formation and navigation switch is done based on the local information and the role of robots in the group.

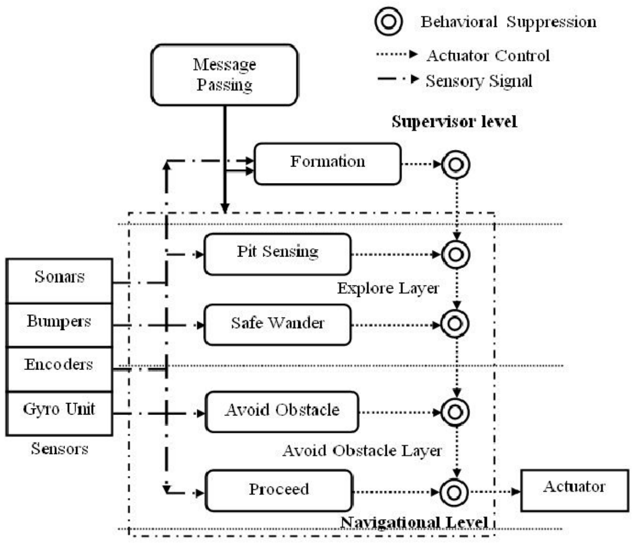

Figure 1 Layered Formation Control Architecture

Navigation capabilities of the robots is achieved at lower level which consist of two layers which are the explore layer and avoid obstacle layer. The higher level formation control consists of supervisor layer required for higher level operation such as formation and communication.

In leader-follower formation approach, the robot acting as leader of the group navigate the environment with the lower level behavioral architecture. At this point, the functionality of the system is divided into simpler task or behaviors that are manipulated sequentially and transmit its relative position and orientations through higher level behavior of message passing. The followers execute the formation while the leader navigates independently.

Detail of some of the behaviors according to Kuppan Chetty R.M, et al, (2011) is as follows:

a) Heading: This behavior process provides approximate heading values for the safe wandering and obstacle avoidance behaviors by processing the positioning data, providing the robot current position and orientation at any time in two dimensional workspace.

b) Avoid Obstacle: By using the sensors, this behavior allows the robots to avoid obstacle without colliding and to navigate through the environment. This behavior is initiated when the robot sensor detects an obstacle and it will manipulate the wheel's translational and rotational velocity.

c) Safe-wandering: With piece wise constant velocity, this behavior guide the robot through the environment by turning left or right at regular intervals at set angle. This allows the robot to explore the environment thoroughly and look for the goal.

d) Message passing: According to Hu H., et al (1998) and James L.C. and Patrick R., (1993), message passing behavior allows the robots to exchange information such as position, orientation and velocity by using the explicit socket communication capability through wireless links. In leader-following formation approach, this behavior provides the necessary interaction between the leader and follower by allowing the leader to current tasks or behavior to the follower and the follower will navigate through the environment according to the leader's information and also allows leader to know the follower's current position.

e) Formation: This behavior manipulates the required wheel velocities of the follower when the leader path is known to maintain the follower position relative to the leader with specified separation and orientation.

The proposed leader-follower formation control approach was experimented using two Pioneer P3DX mobile robots research platform and it was concluded that the dynamic switching between the behaviors and robot helps the follower to trade their roles with the leader to avoid obstacle in their path while maintaining the desired formation.

2.5 Kinematic Model

Figure below shows a system modeling of two robots in a leader-follower formation (Wang Z, et. al., 2012).

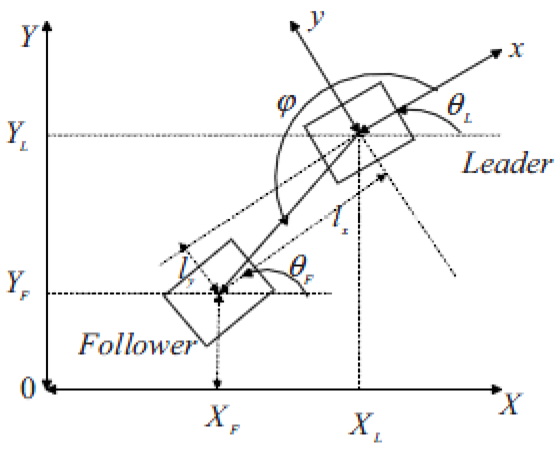

Figure 2 Leader-Following Formation of Two Robots

Let L be leader and F as the follower. For the follower, the desired position is defined as follows:

$$L_X = -(X_L - X_F)\cos(\theta_L) - (Y_L - Y_F)\sin(\theta_\iota) \quad (1)$$

$$L_Y = -(X_L - X_F)\sin(\theta_L) - (Y_L - Y_F)\cos(\theta_\iota)$$

LX and LY are the followers relative position along X and Y direction respectively. XL, YL and XF, YF are the global positions of the leader and follower respectively. The orientation angles are represented by θL and θF. ι and φ are the spatial distance with respect to the leader. To get the desired result ι_D and φ_D for this project, it is required to control the ι and φ at which ‘D’ denotes the desired result..

Since:

$$L_{XD} = L_D \cos(\emptyset_D) \tag{2}$$

$$L_{YD} = L_D \sin(\emptyset_D)$$

Then:

$$\dot{L}_{XD} = \dot{L}_D \cos(\varphi_D) - L_D \dot{\emptyset}_D \, sin(\emptyset_D) \tag{3}$$

$$\dot{L}_{YD} = \dot{L}_D \sin(\varphi_D) - L_D \dot{\emptyset}_D \, cos\emptyset_D)$$

Define:

$$e_F = \begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} L_{XD} - L_X \\ L_{YD} - L_Y \\ \theta_F - \theta_L \end{bmatrix} \tag{4}$$

From equation (1):

$$\dot{L}_X = L_Y w_L + v_F \cos(\theta_e) - v_L \tag{5}$$

$$\dot{L}_Y = -L_X w_L + v_F \sin(\theta_e)$$

If the desired distance, L_D, between the leader and follower is kept constant, then:

$$\begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\theta}_e \end{bmatrix} = \begin{bmatrix} y_e w_L - v_F \cos(\theta_e) + f_1 \\ -x_e w_L - v_F \sin(\theta_e) + f_2 \\ w_L - w_F \end{bmatrix} \tag{6}$$

In which:

$$f_1 = -L_D \, \emptyset_D \sin(\emptyset_D) - w_L L_D \sin(\emptyset_D) + v_L \tag{7}$$

$$f_2 = L_D \, \emptyset_D \cos(\emptyset_D) + w_L L_D \cos(\emptyset_D)$$

CHAPTER 3:

METHODOLOGY

3.1 Project Flow

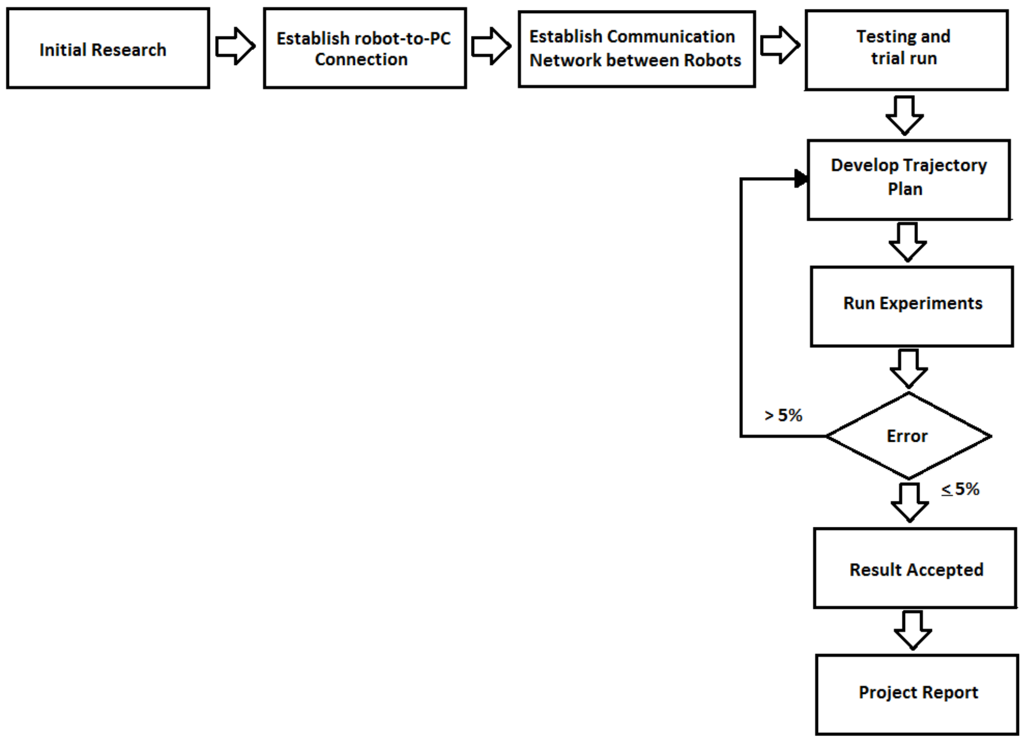

Figure 3 Project Flow

At the early stage of the project, study will be done on the multiple mobile robots coordination and formation. The research is conducted to acquire better understanding on the subject through literature reviews, journals reading, and internet research. Robot-to-PC connection will be established with client-server relationship. The next step is to connect the three AmigoBot mobile robots using the wireless network. After forming connection between the three robots, testing and trial run is done to test the working principle and maneuverability. Trajectory plan will be developed and uploaded to the robots. Experiment will be done using various trajectory plans and set environments. The relative position and orientation of the follower robots with respect to the leader will be recorded and if the error is more than 5% the result will not be accepted. Modification will be done until the result is acceptable.

3.2 Gantt Chart and Project Milestones

Table 1 Gantt Chart For January 2013

No.	Detail/ Week	1	2	3	4	5	6	7		8	9	10	11	12	13	14
1	Selection of Project Topic	🟦	🟦						MID SEM BREAK							
2	Initial Research		🟦	🟦	🟦	🟦										
3	Submission of Extended Proposal Defence						●									
4	Proposal Defence									🟦	🟦					
5	Establish Robot-to-PC connection											🟦	🟦	🟦		
6	Develop Trajectory Plan and Control Program													🟦	🟦	🟦
7	Testing and Trial Run															
															●	
8	Submission of Interim Draft Report															
																●
9	Submission of Interim Report															

●	Suggested milestone
🟦	Process

Table 2 Gantt Chart For May 2013

No.	**Detail/ Week**	**1**	**2**	**3**	**4**	**5**	**6**	**7**		**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
1	Develop Trajectory Plan and Control Program	🟦	🟦	🟦	🟦	🟦	🟦	🟦		🟦	🟦	🟦	🟦	🟦			
2	Testing and Trial Run		🟦	🟦	🟦	🟦	🟦	🟦									
3	Experiment and Data Collection							🟦									
4	Submission of Progress Report									●							
5	Experiments, Data Collection, Report writing									🟦	🟦	🟦	🟦	🟦			
6	Pre-SEDEX												●				
7	Submission of Draft Report													●			
8	Submission of Dissertation (soft bound)														●		
9	Submission of Technical Paper														●		
10	Oral Presentation								MID SEM BREAK							●	
11	Submission of Project Dissertation (Hard Bound)																●

● Suggested milestone

🟦 Process

3.3 Software Required

Software involved includes Microsoft Visual Studio Express 2010, Python Server 2.7 MobileSim and advance robot interface for application (ARIA) library. Microsoft Visual Studio Express 2010 software is used to develop and compile control program by using action group available in ARIA library to dynamically control robot's velocity, heading, relative heading, and other motion parameters either through simple low-level commands or through its high-level Actions infrastructure. Python Server 2.7 allows communication between leader and follower robots by navigating information transfer via wireless network. MobileSim software is for simulating mobile robots platforms and their environments and for debugging and experimentation of the compiled command structure.

3.4 Hardware Required

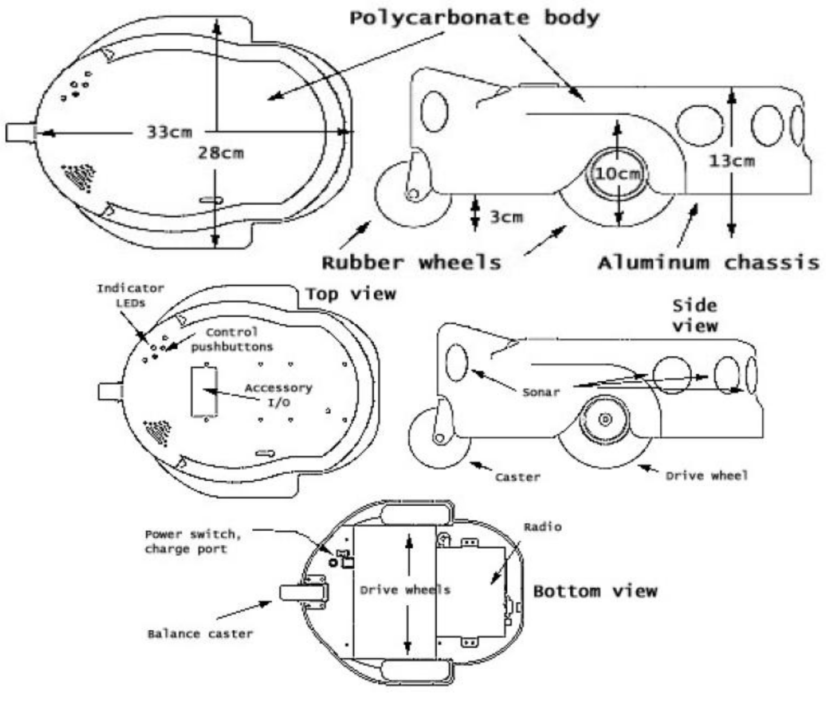

Figure 4 AmigBot Mobile Robot

AmigoBot mobile robots track their position and orientation based on dead-reckoning from wheel motion derived from encoder readings. The robot maintains its internal coordinate position in platform-dependent units, but reports the values in platform-independent millimeters and angular units in the standard SIP (X, Y, heading). Registration between external and internal coordinates deteriorates rapidly with movement due to gearbox play, wheel imbalance and slippage, and many other real world factors. The dead reckoning ability of the robot can only be relied on for just a short range on the order of a few meters and one or two revolutions, depending on the surface. Carpets tend to be worse than hard floors.. On start-up, the robot is at the origin (0, 0, 0), pointing along the positive X-axis at 0 degrees. Figure 5, shows the axis direction for the AmigoBot. For the ease of understanding, the coordinates obtain for results will be change such that positive x-axis is in the right direction and positive y-axis is in the up direction.

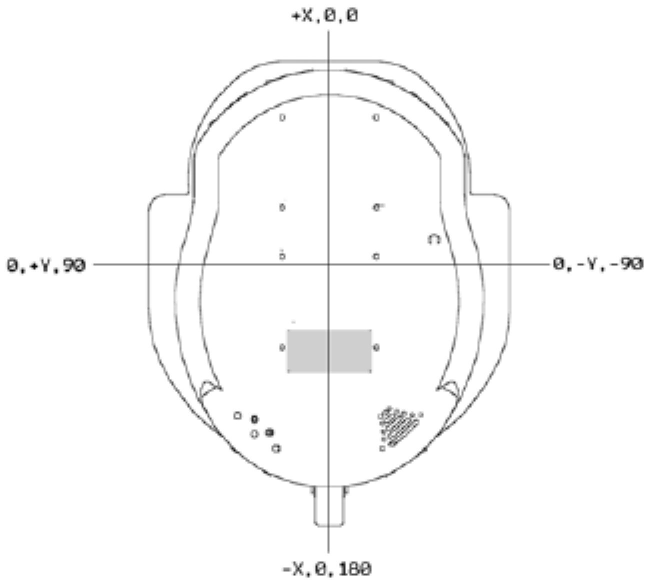

Figure 5 AmigoBot axis direction

The project will be conducted by experiments using 3 AmigoBot mobile robots from the manufacturer Adept Mobile Robots. The following are given specification of the mobile robot:

- Weight: 3.6 kg
- High maneuverability: 750 mm/sec translation speed, 300 degree/second rotation, turns in place
- High-impact resistant polycarbonate body and lightweight aluminum chassis
- Driven by 12 VDC motor
- Eight range finding SONAR, 360 degree coverage
- Built in tracking system
- Wireless Ethernet-to-Serial accessory

3.5 Experiments Description

To test the coordination method for multiple mobile robots for this project, experiments will be conducted by using three AmigoBot mobile robots with similar specification. One robot assigned ‘leader mode’ while the other two robots is in ‘follower mode’.. Each robot will have individual onboard PC connected to the robot using a 9 pin serial-to-USB tether cable and each PC will be connected to each other via a wireless communication network using wireless router.

In Leader mode the robot directly drives to the coordinates while transmitting its position to the Python server. In follower mode the robots will connect to the Python server to receive continuous updates on the position of the leader robot. It will then continuously change the goal of to match the path of the leader. The follower mode cannot be run without the server and a leader running at the same time. The leader mode can however, be run independently.

At the start of the experiment, each robot will be position such that the leader robot will be place in front while the 2 follower more than 30 cm at the back to avoid each other obstacle avoidance range. Distance between the robots is measured before starting and will be measured again at the end to find formation error that may occurred.

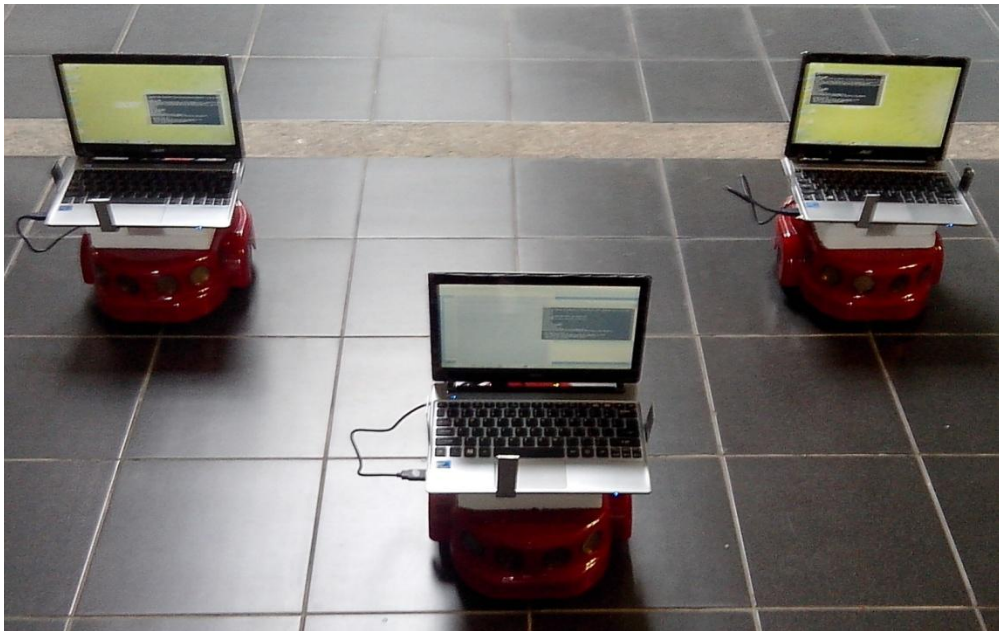

Figure 6 Starting Position for Experiments

Three experiments will be conducted using different trajectory path in coordinating multiple mobile robots. In the first experiment, the robots will move together in a straight line at constant speed until it reaches the desired coordinates. Each coordinates is represented in millimeter unit. The leader is required to move to coordinate (10000, 0). Figure 7 shows the trajectory path for the first experiment:

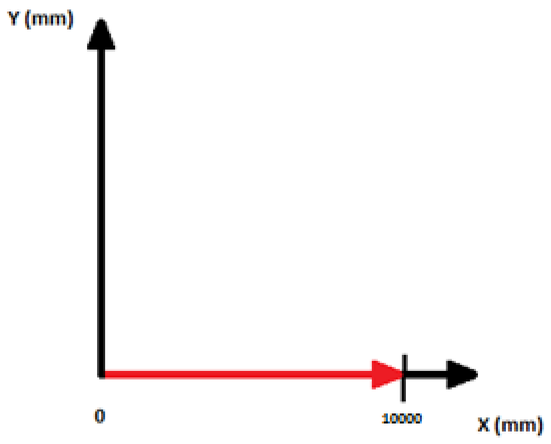

Figure 7 Experiment 1

In the second experiment, the mobile robots are required to complete a more complex trajectory path. The leader robot will be assigned to move in zigzag trajectory by going through the following coordinates, (1500, 1500), (3000,-1500), (4500, 1500) and (6000, 1500) as shown in figure 8.

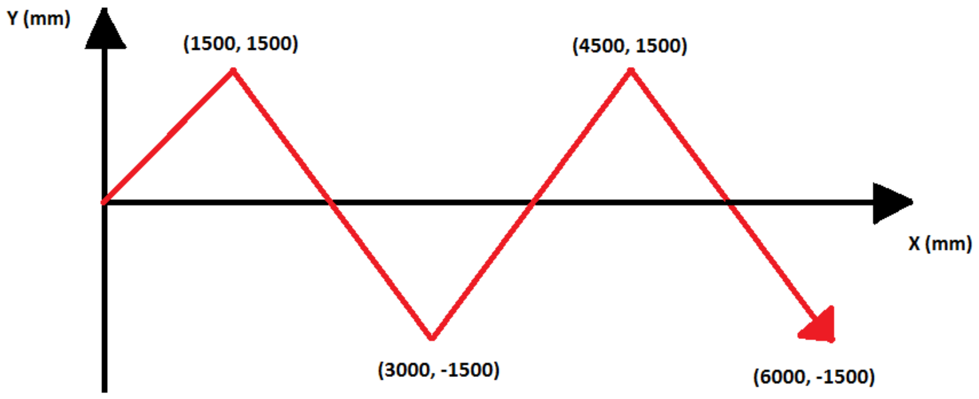

Figure 8 Experiment 2

For the third experiment, the leader robot is tasked to travel in a straight path similar to the first experiment but with obstacles to avoid along the way. The experiment measures the ability of the multiple robots to avoid obstacles while maintaining formation along the set path. 3 obstacles will be placed as shown in figure 6. The leader is expected to avoid the obstacles and followers are to follow the leader trajectory while avoiding obstacle on their respective path. The obstacles is placed at (4500, 0), (6000, 700) and (6000, -700). Figure 9 shows the obstacles placement for experiment 3.

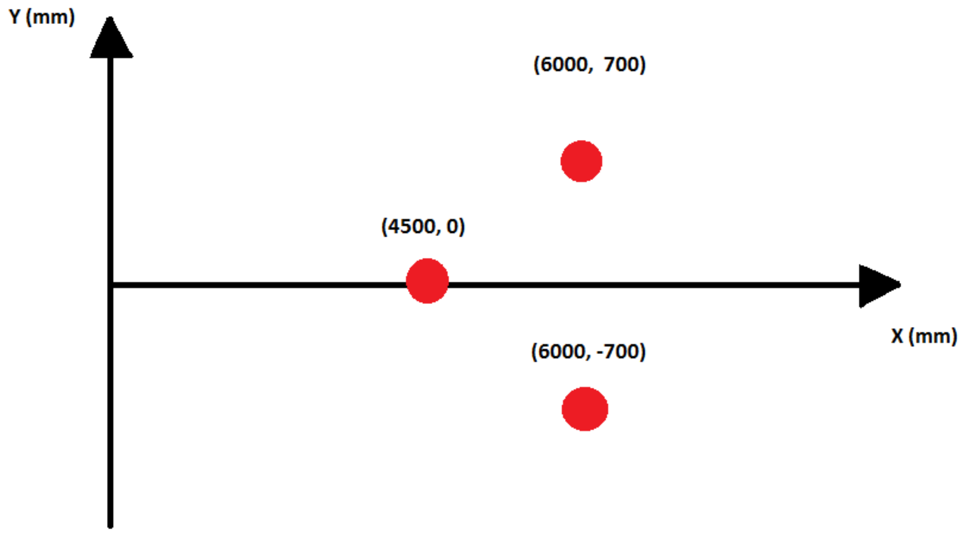

Figure 9 Experiment 3

In all three experiments, the data on the spatial distance between the leader and follower is recorded. Error calculation will be done by comparing starting and end spatial distance using the robots relative coordinates. Modification will be made if the error is more than 5%.

CHAPTER 4:

RESULT AND DISCUSSION

4.1 Control Program

A control program is design to fulfill experiment requirements include performing given task and collecting required data. The control program is developed using Microsoft Visual C++ express utilizing ARIA library. ARIA library contain various action groups and among them are ones that allow communication between robots and onboard PC and controlling the robot's motor and sensors. Following are the main function for the designed control program:

- Build robot-to-PC and PC-to-server connection
- Assign 'leader mode' and 'follower mode' to respective robots
- Goal setting by defining required X and Y coordinates
- Transmit and receive coordinate information and differentiate between leader and follower coordinates
- Behavior control which include stall recovery, pit sensing and obstacle detection and avoidance
- Trajectory plan for more complex trajectory path

The program utilizes Python Server to allow message passing from the leader to the followers. Server receive current position information from all the robots in real time at a rate of 100 milliseconds per cycle and select only the coordinates from the leader and transmit it to follower's onboard PC and updates the follower current goal. This process repeats itself until the leader reaches the target. The server can be installed in either one leader or follower PC. Figure 10 shows a diagram depicting robot-PC and PC-server relation.

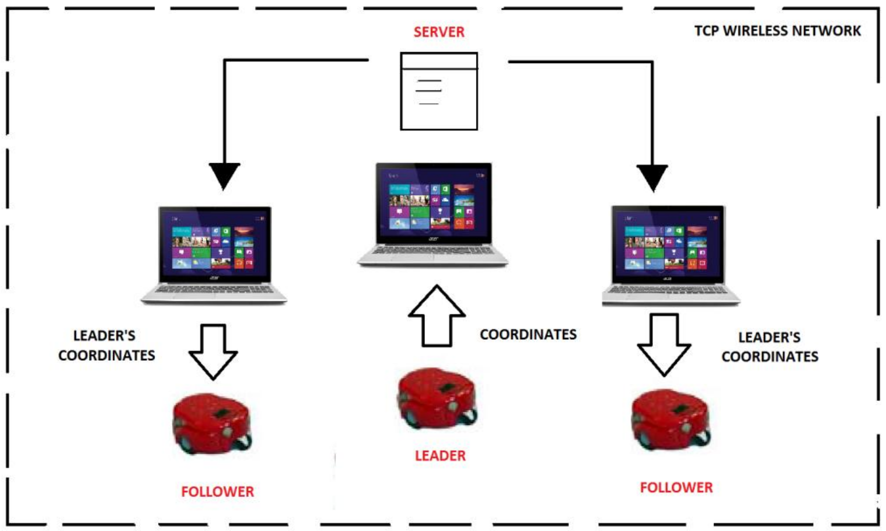

Figure 10 Diagram On Information Transfer

4.2 Main Control Structure

In leader mode, the first priority is stall recovery to ensure that the robot will move after being stopped until it reaches desired destination. The next priority is to stop when the bumper touches an obstacle to avoid any physical damage. Next is to avoid obstacles from a distance with avoid front command line. Avoid obstacles action is divided into short range and long range. Finally, after ensuring safety, the leader priority is to reach destination. Additional priority is set to check for server connection. Below is leader mode C++ code:

```
leader = new ArActionGroup(&robot);
leader->addAction(new ArActionStallRecover, 100);
leader->addAction(new ArActionBumpers, 75);
leader->addAction(new ArActionAvoidFront("Avoid Front Near", 225, 0), 50);
leader->addAction(new ArActionAvoidFront, 45);
leader->addAction(&GoTo, 43);
leader->addAction(new ActionCom(&data_link, &robot),44);
```

In follower mode, the first priority is to connect to the server to obtain leader robot current coordinates and update goal. Other priority is the same as in leader mode. Below is the C++ code for follower:

```
follow = new ArActionGroup(&robot);
follow->addAction(new ActionCom(&data_link, &robot), 100);
follow->addAction(new ArActionStallRecover, 99);
follow->addAction(new ArActionBumpers, 75);
follow->addAction(new ArActionAvoidFront("Avoid Front Near", 225, 0), 50);
follow->addAction(new ArActionAvoidFront, 45);
follow->addAction(&GoTo, 43);
```

For a more complex trajectory, the control structure is similar but it requires proper planning in determining direction, coordinates and the shape of the trajectory path. The shape of trajectory is based on coordinate where the direction changes. The robots will continue to reach the next coordinate after reaching its destination until it reaches the goal. The C++ code is given below:

```
drawZigZag->addAction(new ArActionStallRecover, 100);
drawZigZag->addAction(new ArActionBumpers, 75);
drawZigZag->addAction(new ArActionAvoidFront("Avoid Front Near", 225, 0), 50);
drawZigZag->addAction(new ArActionAvoidFront, 45);
drawZigZag->addAction(new ArActionGoto("Side 1", ArPose(1500, 1500), 100, 400,
150, 7), 44);
drawZigZag->addAction(new ArActionGoto("Side 2", ArPose(-1500, 3000), 100, 400,
150, 7), 43);
drawZigZag->addAction(new ArActionGoto("Side 3", ArPose(1500, 4500), 100, 400,
150, 7), 42);
drawZigZag->addAction(new ArActionGoto("Side 4", ArPose(-1500, 6000), 100, 400,
150, 7), 41);
```

Other control structure includes robot-to-PC and PC-to-server connection, coordinates information transfer and command key setting.

4.3 Results and Discussion

The formation control structure designed is independent of the robot capability to maintain trajectory while unaffected by terrain condition. AmigoBot mobile robots are for research purpose and not suitable to do an actual field work thus the results may differ depending on type of mobile robot.

The results and discussion for all 3 experiments will be based on coordinate data acquired instead of actual position of the robots in the environment. This due to the fact that the mobile robots tends diverge from its path because of the terrain condition. Moving either too fast or too slow tends to exacerbate the absolute position errors. The robots dead reckoning capability is a means of tying together sensor readings taken over a short period of time.

The graph plot represents the path taken by the robots as seen from top view. As the robots move to reach specified goal, real time coordinate for all the robots is captured using python server. Coordinate information is used to plot the graph. Error calculation will be done based on the coordinate data instead of measuring it in actual environment.

4.3.1 Experiment 1

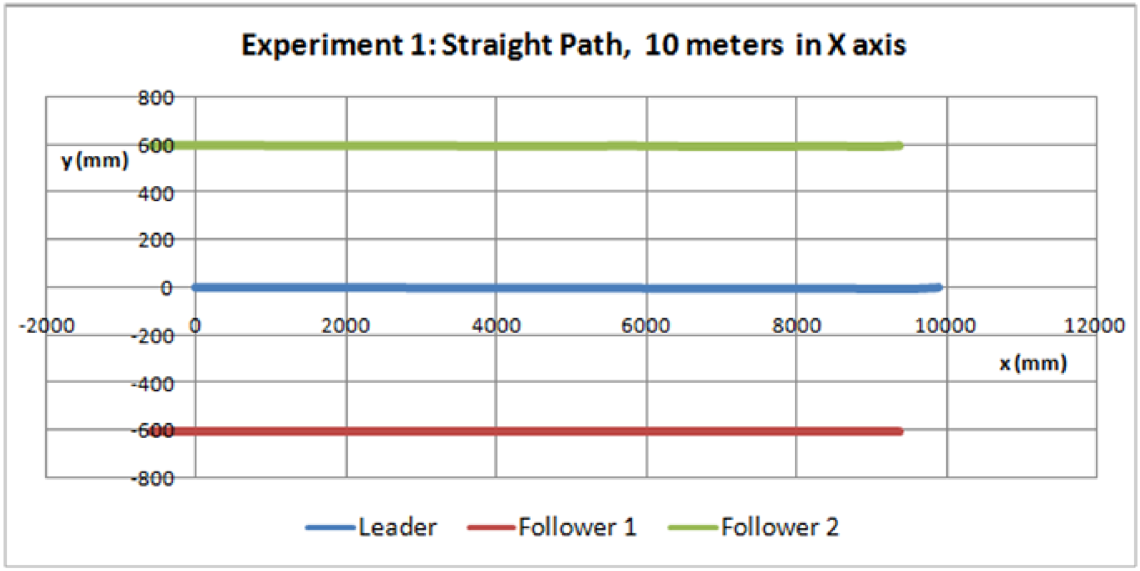

Figure 11 Experiment 1 Result

Figure 11 shows a graph plot showing the result for experiment 1. Based on the figure, leader robots travel in a straight path with little to no deviation. Both followers manage to follow the leader without any difficulties. This experiment proves that the control program designed is working and capable of message passing to coordinate the mobile robots.

Table 3 Experiment 1: Coordinate Data

No	Leader		Follower 1		Follower 2	
	x	y	x	y	x	y
1	0	0	-600	-600	-600	600
2	378	-1	-514	-600	-514	599
3	767	-1	-111	-601	-132	601
4	1246	-1	384	-601	357	600
5	1695	-1	838	-601	810	600
6	2085	-1	1229	-602	1201	599
7	2445	-1	1589	-603	1560	599
8	2805	-1	1949	-603	1919	599
9	3149	-2	2339	-603	2309	599
10	3583	-2	2728	-603	2698	598
11	3973	-2	3117	-602	3088	598
12	4362	-2	3508	-604	3477	598
13	4752	-2	3896	-603	3867	597
14	5141	-2	4287	-601	4256	597
15	5561	-2	4706	-602	4676	598
16	5950	-2	5095	-601	5065	597
17	6370	-3	5515	-602	5485	598
18	6759	-3	5904	-603	5874	598
19	7119	-3	6264	-603	6234	597
20	7509	-3	6654	-603	6623	596
21	7982	-3	7073	-603	7043	596
22	8317	-3	7463	-603	7433	595
23	8706	-3	7852	-603	7822	596
24	9096	-4	8241	-603	8212	597
25	9486	-4	8630	-603	8600	597
26	9869	-1	9020	-603	8990	595
27	9900	0	9377	-601	9377	599

4.3.2 Experiment 2

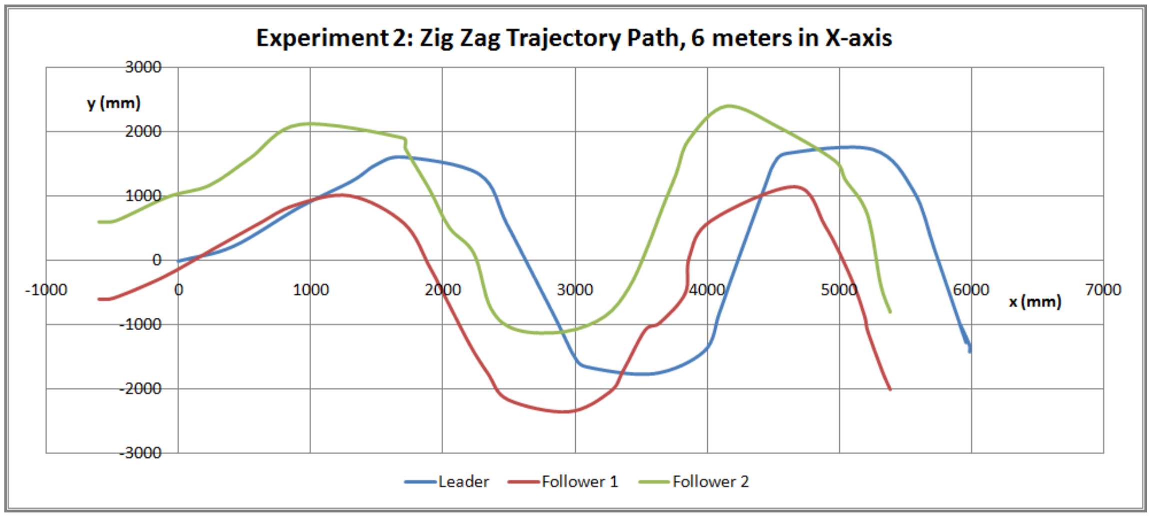

Figure 12 Experiment 2 Result

The result for experiment 2 is shown in figure 12. For this experiment, the leader successfully reaches each coordinates points, forming a zigzag pattern until it reaches the end coordinates. The follower robots still follow leader robot's movement but not as smooth as the previous experiment. From (0, 0) until (1500. 1500) Follower 1 and Follower 2 follow the leader without much trouble. After the first turn at (1500, 1500) until (3000, -1500), Follower 2 is struggling to find the correct path but eventually manages maintain the original trajectory path as shown in figure 13. Similar incident happened from (3000, -1500) to (4500, 1500) but this time to Follower 1. This is shown in figure 14. A possible explanation for this incidence is that, at the turning point, distance between leader and follower closes in causing follower to detect the leader as an obstacle. The follower slows down to avoid collision. As the leader continue moving the stall recovery command become first priority as the obstacle is no more. Possible solution to prevent such incidence from occurring includes increasing the initial spatial distance, decrease the obstacle avoidance range of the follower, adjust follower velocity to prevent it from getting too near and increase leader velocity.

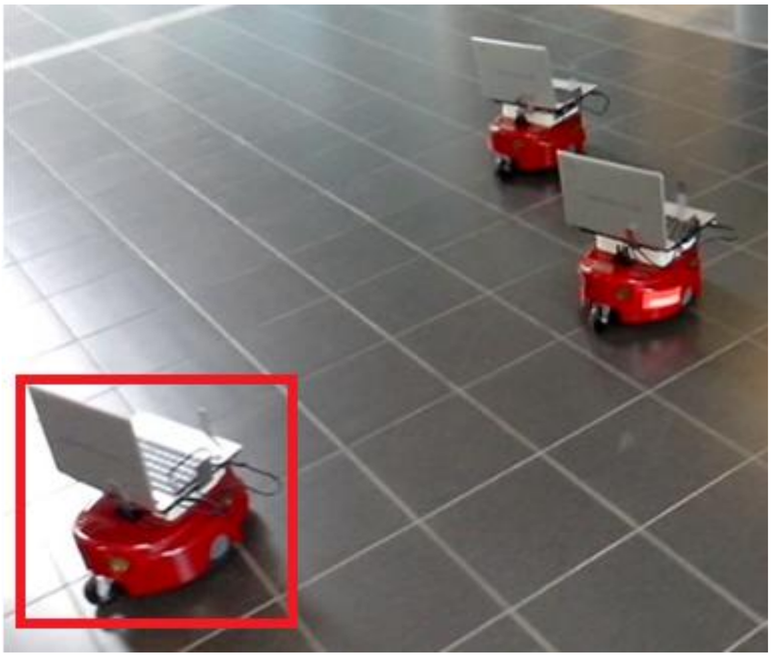

Figure 13 Experiment 2: Follower 2, lagging behind

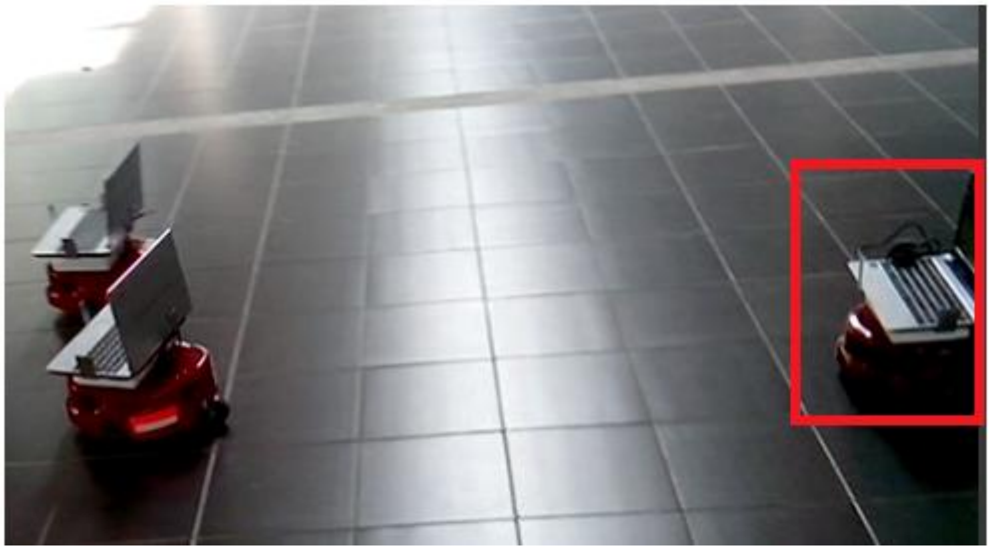

Figure 14 Experiment 2: Follower 1, lagging behind

Table 4 Experiment 2: Coordinate Data

No.	Leader 1		Follower 1		Follower 2	
1	0	0	-600	-600	-600	600
2	414	227	-490	-584	-465	625
3	914	834	-115	-260	-78	992
4	1312	1240	267	180	235	1169
5	1492	1490	606	567	539	1588
6	1698	1614	878	855	920	2116
7	2275	1349	1295	1003	1698	1906
8	2483	582	1703	580	1716	1742
9	2702	-289	1901	-105	1902	1091
10	2871	-966	2037	-646	2047	515
11	2993	-1473	2203	-1305	2235	107
12	3081	-1637	2341	-1755	2363	-746
13	3590	-1748	2494	-2160	2563	-1080
14	3976	-1400	2948	-2346	2928	-1102
15	4090	-813	3262	-2038	3225	-866
16	4175	-312	3367	-1693	3406	-424
17	4303	416	3529	-1080	3537	160
18	4444	1208	3640	-967	3661	826
19	4498	1488	3833	-507	3763	1344
20	4593	1667	3863	52	3861	1884
21	5254	1731	4029	618	4144	2400
22	5561	1084	4663	1142	4544	2066
23	5715	199	4893	524	4963	1552
24	5846	-589	5086	-274	5042	1256
25	5956	-1257	5188	-863	5209	717
26	5909	-972	5211	-1089	5309	-372
27	5983	-1307	5321	-1714	5381	-793
28	5983	-1407	5382	-2001	5381	-801

4.3.3 Experiment 3

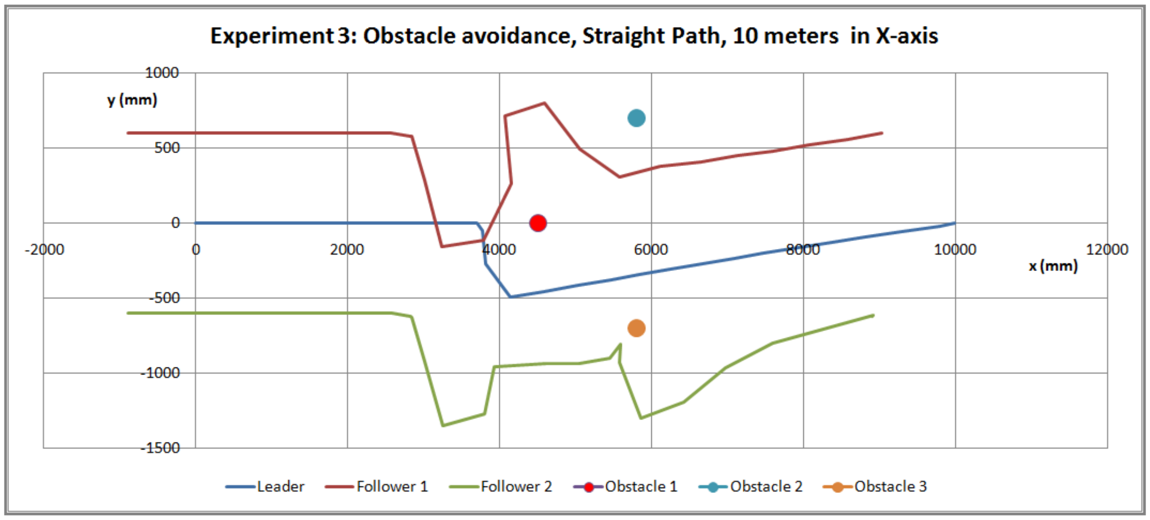

Figure 15 Experiment 3 Result

Figure 15 shows the result for the third experiment. The obstacle is placed is such away so that it will test follower’s ability to follow leader as the leader avoid obstacles and when followers meet an obstacle in their own path as it follow the leader. Based on the figure above, it shows that all three robots manage to reach specified goal while avoiding obstacles in their respective path. Leader moves in a straight line and when it detect Obstacle 1, it stops as avoid obstacle command line becomes main priority and changes orientation pointing away from obstacles as shown in figure 16. Stall recovery command takes over to move away from the obstacle. When the path is clear from any obstacle, the main priority will be to reach specified goal.

For both followers, even without any obstacle in their path, they will still follow the leader as it avoids Obstacle 1. Follower 1 and Follower 2 will then keep following the leader until they detect obstacles on their respective path. Follower 1 detects Obstacle 1 from a distance as it follows Leader using avoid front command line and change its course. Follower 1 repeats the same action to avoid obstacle 2 as shown in figure 17. Follower 2 did not detect Obstacle 3 from distance. The cause may due to Follower 2 moving too fast or the sensor is lagging. Figure 18 show that follower 2 still manages to avoid the obstacle by bumper command action. When Follower 2 touches the obstacle, it will stop and try to find a different path to avoid the obstacle. When both followers path are cleared, they will continue to move according to leader.

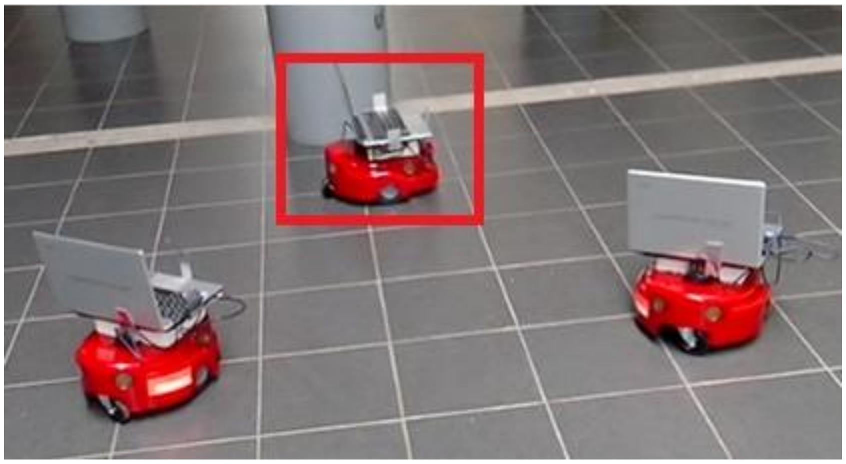
Figure 16 Experiment 3: Leader Avoid Obstacle 1

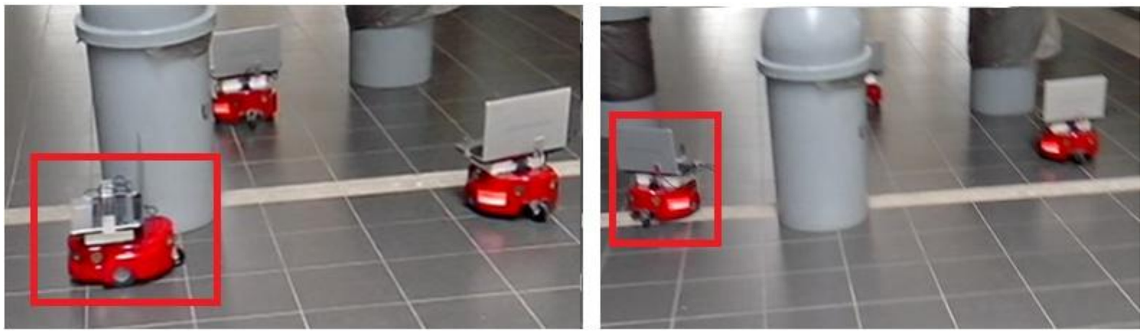
Figure 17 Experiment 3: Follower 1 Avoid Obstacle1 and 2

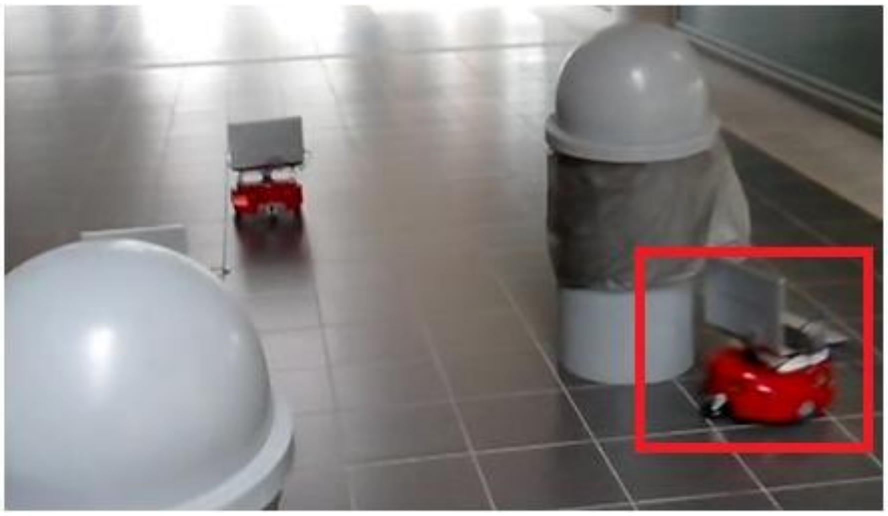
Figure 18 Experiment3: Follower 2 Avoid Obstacle 3

Table 5 Experiment 3: Coordinate Data

No.	Leader 1		Follower 1		Follower 2	
	x	y	x	y	x	y
1	0	0	-900	600	-900	-600
2	439	-1	-788	600	-765	-601
3	888	-1	-302	599	-283	-601
4	1397	-1	139	599	211	-601
5	1787	-1	585	599	594	-602
6	2177	-1	967	599	977	-602
7	2596	-2	1360	598	1383	-601
8	2985	-2	1772	598	1779	-602
9	3375	-2	2160	598	2172	-603
10	3708	-1	2578	598	2588	-603
11	3773	-47	2823	579	2826	-621
12	3815	-275	2838	572	2839	-628
13	4141	-494	3015	285	3013	-920
14	4576	-457	3239	-157	3256	-1350
15	5024	-417	3792	-114	3805	-1274
16	5470	-378	4154	263	3927	-960
17	5831	-341	4067	713	4611	-936
18	6248	-305	4584	796	5036	-938
19	6665	-270	5049	491	5455	-897
20	7082	-235	5577	304	5597	-810
21	7469	-202	6114	377	5585	-926
22	7916	-164	6648	405	5866	-1299
23	8364	-130	7130	444	6428	-1193
24	8810	-95	7590	479	6981	-966
25	9287	-58	8071	518	7598	-803
26	9794	-18	8579	556	8923	-616
27	9991	-1	9035	594	8911	-619

4.4 Error Calculation

Error calculation is done using the coordinate information obtained. Table 6 shows the error calculation done for all 3 experiments.

Table 6 Error Calculation Results

Experiment	Mobile Robot	Initial Spatial Distance (mm)	End Spatial Distance (mm)	Error Calculated (%)
1	Follower 1	848.5	796.7	6 .1
	Follower 2	848.5	795.2	6 .2
2	Follower 1	848.5	848.5	0.4
	Follower 2	845.0	854.2	0.7
3	Follower 1	1081.7	1126.0	4.0
	Follower 2	1081.7	1244.3	15.0
			Average	**5.4**

For experiment 1, the average error calculated is 6.15 %. For experiment 2, 0.55 % and 9.5 % for experiment 3. The average error calculated for all three experiment 5.4% which is still acceptable error for this project. Error for experiment 1 is quite high due to the ineffectiveness of the mobile robots to travel a long distance. The same goes for experiment 3 but with higher calculated error as the obstacles interfere with the follower’s trajectory and possibly making it deviate from its original designated position. For experiment 2, the calculated error is lower since zigzag shaped trajectory path requires the robots to travel in short distances to reach each coordinate point which is more for AmigoBot mobile robots.

CHAPTER 5:

CONCLUSION AND RECOMMENDATION

Based on the experiment results, coordination between three mobile robots is successful done for experiment 1; 10m straight line trajectory path, experiment 2; 6m zigzag shaped trajectory path and experiment 3; 10m straight line with obstacles trajectory path. The designed control programming allows information transfer and behavior control which are vital to coordinate the mobile robots. From the three experiments, the follower follows the leader trajectory and successfully maintains spatial distance and orientation with an average error of 5.4 %. For future experiment, it is recommended that the designed trajectory path is in short distance but if long distance is required, it is preferred to divide it into several coordinates point reduce margin of error. To increase the effectiveness of the designed control program, additional function can be added such that to control translational and rotational velocity, area mapping, line tracing and object recognition.

REFERENCES

1. Balch T. and Hybinette M., 2000, "Behavior-based coordination of large-scale robot formations,". *Proceedings. Fourth International Conference on MultiAgent Systems*. < http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=858476>
2. Bahceci, E., Soysal, O., and Sahin, E., October 2003, *A Review: Pattern Formation and Adaptation in Multi-Robot Systems*. CMU-RI-TR-03-43, Robotics Institute Carnegie Mellon University Pittsburgh, Pennsylvania 15213. < http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1013622>
3. Belta C. and Kumar V. 2002. "Trajectory design for formations of robots by kinetic energy shaping". *In Proceedings. ICRA '02. IEEE International Conference on Robotics and Automation.*
4. Crowly, J. L. and Reigner, P., 1993 "Asynchronous Control of Rotation and Translation for a Robot Vehicle," International Journal of Robotics and Autonomous Systems, Vol. 10, pp. 243-251.
5. Do, K.D., 2007, *"*Formation tracking control of unicycle-type mobile robots," in: Proc. of Robotics and Automation Conf., Roma. < http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4209441>
6. Dudenhoeffer D.D. and Jones M.P. 2000, "A formation behavior for large-scale micro-robot force deployment," *Simulation Conference Proceedings*.
7. Ghommam, J., Saad, M., and Mnif, F. 2008, "Formation path following control of unicycle-type mobile robots*", IEEE International Conference on Robotics and Automation Pasadena*, CA, USA. <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4543495>
8. Hu H. et al., 1998 "Coordination of Multiple Mobile Robots via Communication," *In Proceedings of SPIE '98 Mobile Robots XIII Conference*, Boston, pp 94-103.
9. J.P. Desai, J. Ostrowski, V. Kumar, 1998. "Controlling formations of multiple mobile robots," *In IEEE International Conference on Robotics and Automation*, pp. 2864–2869.
10. Vyas, K, Metsis, V. and Makedon, F., n.d., "ARIA: Getting Started Quickly," 416 Yates Street, 250 Nedderman Hall, Arlington. TX76019, USA

11. Kuppan Chetty, RM., Singaperumal, M. and Nagarajan, T., 2011, “Planning and Control Formation in Multiple Mobile Robots,” *International Journal of Advanced Robotics System*. ISSN 1729-8806.
12. Kuppan Chetty, RM., Singaperumal, M. Karsiti N.B and Nagarajan, T., n.d., “ State Based Modelling and Control of a Multi Robot Systems using SIMULINK
13. M. Breivik, T. Fossen, 2006, “Guided formation control for wheeled mobile robots,” In9th IEEE Conference on Control, Automation, Robotics and Vision, Singapore, pp. 1_7.
14. Ota, J., Arai, T., and Asama, H. 2002, “Cooperative transport by multiple mobile robots in unknown static environments associated with real-time task assignment,” *In Proceedings. ICRA '02. IEEE International Conference on Robotics and Automation*.
< http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1067997>
15. R. Beard, J. Lawton, and Hadaegh, 1999, “A coordination architecture for spacecraft formation control”, *IEEE Transactions on Control Systems Technology* , pp. 777-790.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=960341
16. T. Balch and R.C. Arkin, 1998, “Behavior-based formation control for multirobot teams,” *IEEE Transactions on Robotics and Automation*, pp. 926-939. < ieeexplore.ieee.org/xpl/freeabs_all.jsp?&arnumber=736776>
17. T. D. Barfoot and C. M. Clark,2004. *Motion Planning for Formations of Mobile Robots*, Robot. Auton. Syst., vol. 46, pp. 65-78.
18. Wang, P.K.C.,1992 “Navigation strategies for multiple autonomous robots moving in formation,” *Journal of Robotic Systems 8 (2),* pp. 177-195.
< http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=637948>
19. Wang Z., Mao Y., Chen G., and Chen Q., 2012, “Leader Follower and Communication Based Formation Control of Multi-Robots,” *In Proceedings of the 10th World Congress on Intelligent Control and Automation*
20. Wu J., and Jiang Z., 2009, “Formation Control of Multiple Mobile Robots Via Switching Strategy,” *In International Journal of Information and System Sciences Volume 5, number 2,* pg 210-218

APPENDIX

Control Program C++ Code

```
#include "Aria.h"
#include <stdio.h>
#include <iostream>
#define SERVER_IP "127.0.0.1"
#define DEFAULT_PORT "27015"
#define DEFAULT_BUFLEN 22
#include <string.h>
#include <sstream>

using namespace std;
//Variable that determines if the robot is in leader mode or not
char *L = "L";
char *F = "F";
char *leader_flag = L;
int modeFlag = 1;
int switchFlag = 0;

//Target coordinates when the robot is in follower mode. This keeps changing
double X_goal = 2500;
double Y_goal = 2500;
char * pch; char *end;
char *s; char *s2;
ArActionGoto GoTo("drive to target", ArPose(X_goal, Y_goal));

//Target coordinates for when the robot is leading. This does not change.
double leader_goalX;
double leader_goalY;

//Side length of the square for the drawSquare action group
double side_length = 10000;

//Network data
char *sendbuf = "Bot1 ACK";
char recvbuf[DEFAULT_BUFLEN];
int iResult;
int recvbuflen = DEFAULT_BUFLEN;
///Strings that will contain the conversions
string xcord; string ycord;

//Function to convert from number to string
template <typename T>
  string NumberToString ( T Number1, T Number2, T Number3 )
  {
     ostringstream ss;
     if(modeFlag==0) ss << Number1<<","<<Number2<<","<<Number3<<".";
     if(modeFlag==1) ss <<"L"<<Number1<<","<<Number2<<","<<Number3<<".";
     return ss.str();
  }
```

```
void updateGoal(string str)
{
    size_t found1 = str.find(",");
    X_goal = (double)atoi((str.substr(0, found1)).c_str());
    size_t found2 = str.find(",", found1+1);
    Y_goal = (double)atoi((str.substr(found1+1,found2-found1)).c_str());
        if(modeFlag==0)
        {
                GoTo.setGoal(ArPose(X_goal,Y_goal));
        }
        if(modeFlag==1)
        {
                GoTo.setGoal(ArPose(leader_goalX, leader_goalY));
        }
}

//ArAction classes
class ActionCom : public ArAction
{
public:
        //Constructor
        ActionCom(ArTcpConnection *data_link, ArRobot *robot);
        //Destructor. Does not need to do anything
        virtual ~ActionCom(void) {};
        //Called by the action resolver to obtain the action's requested behaviour
        virtual ArActionDesired *fire(ArActionDesired currentDesired);
        //Store the robot pointer and it's ArSonarDevice object, or deactivate this action
        if there is no sonar
        virtual void setRobot(ArRobot *robot);
        int x_goal, y_goal;
protected:
        //the sonar device object is obtained from the robot by setRobot()

        ArRangeDevice *mySonar;
        ArTcpConnection *tcpptr;
        ArRobot *robotptr;
        int x_pos; int y_pos; int heading;
        double obsDist, angle;
        /*  Our current desired action: fire() modifies this object and returns to the
          action resolver a pointer this object. This object s kept as a class member
          so that it persists after fire() returns (otherwise fire() would have to
          create a new object each invocation but would never be able to delete that
          object).
        ArActionDesired myDesired;
};

ActionCom::ActionCom(ArTcpConnection *data_link, ArRobot *robot) : ArAction("Go")
{
        //This is the constructor. Some data setup is needed
        data_link->write("Red Leader Checking in!", DEFAULT_BUFLEN);
        //Set an internal pointer to point to the TCP connection
        tcpptr = data_link;
        //Set an internal pointer to the robot
        robotptr = robot;
        //Get the curent pose of the robot.
        x_pos = (robotptr->getX());
        y_pos = (robotptr->getY());
        heading = robotptr->getCompass();
        printf("Sending data to server.");
}

void ActionCom::setRobot(ArRobot *robot)
{
        ArAction::setRobot(robot);
        mySonar = robot->findRangeDevice("sonar");
        if(robot==NULL)
        {
                ArLog::log(ArLog::Terse,  "ActionCom:  Warning!  I  found  no  sonar.
Deactivating");
                deactivate();
        }
}

/*
        This file is the whole point of the action.
        currentDesired is the combined desired action from other actions
        previously processed by the action resolver. In this case, we're
        not interested in that, we will set our desired forward velocity
        in the myDesired member and return it.

        Note that myDesired must be a class member since this method will
        return a pointer to myDesired to the caller. If we had declared

        the desired action as a local variable in this method the pointer we
        returned would be invalid after this method returned.

*/

ArActionDesired *ActionCom::fire(ArActionDesired currentDesired)
{
        //Get the current pose of the robot
        x_pos = robotptr->getX();
        y_pos = robotptr->getY();
        heading = robotptr->getCompass();
        obsDist = (robotptr->checkRangeDevicesCurrentPolar(-70, 70, &angle) - robotptr-
>getRobotRadius());
        if(obsDist < 450) switchFlag = 1;
        //Send the pose as a string over the TCP connection
        tcpptr->write(NumberToString(x_pos, y_pos, heading).c_str(), DEFAULT_BUFLEN);
        //Get the position of the leader
        tcpptr->read(recvbuf, DEFAULT_BUFLEN, 5);
        printf("Received: %s¥n", recvbuf);
        //Convert the message to a string
        string str = (string)recvbuf;
        //cout<<"¥nString: "<<str<<"¥n";
        //Update the goal of the robot
        if(recvbuf[0] == 'S') modeFlag ^= 1;
        updateGoal(str);
        return &myDesired;
}


ArActionGroup *teleop;
ArActionGroup *follow;
ArActionGroup *leader;
ArActionGroup *drawZigZag;

// Activate the wander action group. activateExlcusive() causes
// all other active action groups to be deactivated../
void followMode(void)
{
    follow->activateExclusive();
    printf("¥n== Follow Mode ==¥n");
    printf("     The robot will now follow the leader around¥n     Press 't' to switch to
teleop mode.¥n    Press escape to exit.¥n");
}

//Activate the leader mode of the robot.
void leaderMode(void)
{
        leader->activateExclusive();
        printf("¥n== Leader Mode ==¥n");
        printf("¥nThe robot will now act as a leader in cooperative navigation.¥n");
}

//Activate the actionGroup that makes the robot draw a square.
void drawZigZagMode(void)
{
        drawZigZag->activateExclusive();
        printf("¥n== Draw ZigZag Mode ==¥n");
        printf("¥nThe robot will now move in a ZigZag path¥n");
}

int main(int argc, char** argv)
{
    Aria::init();
    ArArgumentParser argParser(&argc, argv);
        char* server = argParser.checkParameterArgument("-rs");
        char* mode_flag = argParser.checkParameterArgument("-mode");
        if(!mode_flag) modeFlag = 1;
        else if(strcmp(mode_flag, "F")) modeFlag = 0;
        else if(strcmp(mode_flag, "L")) modeFlag = 1;
        else leader_flag = mode_flag;
        if(!server) server = "localhost";
        char* x_togo = (argParser.checkParameterArgument("-X"));
        char* y_togo = (argParser.checkParameterArgument("-Y"));
        if(x_togo) X_goal = (double)(atoi(x_togo));
        if(y_togo) Y_goal = (double)(atoi(y_togo));
        if(x_togo) leader_goalX = (double)(atoi(x_togo));
        if(y_togo) leader_goalY = (double)(atoi(y_togo));
        cout<<"¥nX: "<<X_goal<<" Y: "<<Y_goal<<"¥n";
        GoTo.setGoal(ArPose(X_goal, Y_goal));
    ArSimpleConnector con(&argParser);
    ArRobot robot;
    ArSonarDevice sonar;
        //Initialize the TCP connection
        ArTcpConnection data_link;
        //Connect to the server.
        //Replace 127.0.0.1 with the IP address of the server.
        data_link.open(server, 27015);
        //WSAStartup(MAKEWORD(2,2), &wsaData);
    argParser.loadDefaultArguments();
    if(!Aria::parseArgs() || !argParser.checkHelpAndWarnUnparsed())
    {
        Aria::logOptions();
        return 1;
    }
    /* - the action group for follow actions: */
    follow = new ArActionGroup(&robot);

    //Transmit location and receive goal
    follow->addAction(new ActionCom(&data_link, &robot), 100);
    // if we're stalled we want to back up and recover
    follow->addAction(new ArActionStallRecover, 99);
    // react to bumpers
    follow->addAction(new ArActionBumpers, 75);
    // turn to avoid things closer to us
    follow->addAction(new ArActionAvoidFront("Avoid Front Near", 225, 0), 50);
    // turn avoid things further away
    follow->addAction(new ArActionAvoidFront, 45);
    // keep moving
    follow->addAction(&GoTo, 43);



    //The action group for the leader mode operation
    leader = new ArActionGroup(&robot);

    //Highest priority for recovering froma stall.
    leader->addAction(new ArActionStallRecover, 100);
    //Next highest priiority for responding to bumpers.
    leader->addAction(new ArActionBumpers, 75);
    //Avoiding obstacles detected by the sonar that are close by.
    leader->addAction(new ArActionAvoidFront("Avoid Front Near", 225, 0), 50);
    //Avoiding obstacles that are far away.
    leader->addAction(new ArActionAvoidFront, 45);
```

```
    //Going to the goal.
    leader->addAction(&GoTo, 43);
    //Connection test
    leader->addAction(new ActionCom(&data_link, &robot),44);


    //The action group for moving the robot in a square
    drawZigZag = new ArActionGroup(&robot);

    drawZigZag->addAction(new ArActionStallRecover, 100);
    //Next highest priiority for responding to bumpers.
    drawZigZag->addAction(new ArActionBumpers, 75);
    //Avoiding obstacles detected by the sonar that are close by.
    drawZigZag->addAction(new ArActionAvoidFront("Avoid Front Near", 225, 0), 50);
    //Avoiding obstacles that are far away.
    drawZigZag->addAction(new ArActionAvoidFront, 45);
    //Sides of ZigZag pattern
    drawZigZag->addAction(new ArActionGoto("Side 1", ArPose(1500, 1500), 100, 400, 150,
7), 44);
    drawZigZag->addAction(new ArActionGoto("Side 2", ArPose(-1500, 3000), 100, 400, 150,
7), 43);
    drawZigZag->addAction(new ArActionGoto("Side 3", ArPose(1500, 4500), 100, 400, 150,
7), 42);
    drawZigZag->addAction(new ArActionGoto("Side 4", ArPose(-1500, 6000), 100, 400, 150,
7), 41);


    /* - use key commands to switch modes, and use keyboard
     *   and joystick as inputs for teleoperation actions. */

    // create key handler if Aria does not already have one
    ArKeyHandler *keyHandler = Aria::getKeyHandler();
    if (keyHandler == NULL)
    {
        keyHandler = new ArKeyHandler;
        Aria::setKeyHandler(keyHandler);
       robot.attachKeyHandler(keyHandler);
    }

    // set the callbacks

    ArGlobalFunctor followCB(&followMode);
    ArGlobalFunctor leaderCB(&leaderMode);
    ArGlobalFunctor squareCB(&drawZigZagMode);

    keyHandler->addKeyHandler('F', &followCB);
    keyHandler->addKeyHandler('f', &followCB);

    keyHandler->addKeyHandler('l', &leaderCB);
    keyHandler->addKeyHandler('L', &leaderCB);

    keyHandler->addKeyHandler('s', &ZigZagCB);
    keyHandler->addKeyHandler('S', &ZigZagCB);


    //Setting up the robot as the client

    /* - connect to the robot, then enter teleoperation mode.  */

    robot.addRangeDevice(&sonar);
    if(!con.connectRobot(&robot))
    {
        ArLog::log(ArLog::Terse, "actionGroupExample: Could not connect to the robot.");
       Aria::exit(1);
    }

    robot.enableMotors();
       robot.run(true);

    Aria::exit(0);
       data_link.close();
}
```

<u>Python server script</u>

```
#Importing all the libraries necessary

#for socket programming.

import socket

import select

import sys


#Setting up the TCP connection

host = ''

port = int(raw_input("Enter port Number (Try 27015 or 27016): "))

backlog = 5

size = 512

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

s.bind((host, port))

#Listen for incoming connection on the port given by user

s.listen(backlog)

count = 1

addrlist=['192.168.0.103', '127.0.0.1']

raw_data = []

leaderPos = '0,0,0.'

i=0


#Open a file to write the data to.

datafile = open("exp1.txt", 'a')
```

```
#Cycle through the connections from the robots.
#This loop keeps running as long as there are connections
#coming from the robots.
read_list = [s]
while True:
    readable, writable, error = select.select(read_list, [], [],0)
    for sock in readable:
        if sock is s:
            client, address = sock.accept()
            read_list.append(client)
            print "Connection from", address
        else:
            data = sock.recv(size)
            #write data to file
            datafile.write("%s\n" %(data))
            if(data[0]=="L"):
                leaderPos = data
            print data
            if ((data)):
                sock.sendall(leaderPos)
            else:
                sock.close()
                read_list.remove(sock)
#close the file
datafile.close()
```