Lead-Follower Formation with Multiple Wirelessly Synchronized Autonomous Unmanned Ground Vehicles

By:
Marcos Bird
Chuong Khuc
Eric Ortega
Carlos Quiroz

Research Experience for Undergraduates
at the
University of Texas of San Antonio

August 10, 2007

Faculty advisor:
Dr. Yufang Jin

Electrical Engineering Department
School of Engineering and Computer Science
The University of Texas at San Antonio
The University of Texas Pan-American

Table of Contents

Table of Contents

List of Figures

I. Introduction

II. Trajectory Planning

III. Leader

List of Figures

IV. Follower

V. Wireless Communication Network

I. Introduction

Unmanned vehicles will have many important roles in the upcoming future. The importance of these vehicles is that they can be used in many areas such as search and rescue, exploration, surveillance, and for military use. These vehicles must be able to navigate in the dynamic environment we live in. The current state of development for autonomous vehicles of any sort is still at a very early stage. Although there have been some improvements over the last few years, UGVs are not anywhere close to being truly autonomous. They require some form of human instruction or interaction such as a predetermined path. An ideal UGV could adapt to its surroundings and learn new capabilities. The goal of program is to interest its members to research and to design an unmanned ground vehicle (UGV) that can navigate and adapt to its surroundings.

Figure 1.1: System Configuration

The Pioneer P3-AT was designated as the leader for a group of X4 Rovers developed by TotalRobots. The Lantronix WiBox was integrated on the P3-AT to provide a means of wireless communication between the robot and laptop computer, and the Designer Systems DS-FWCM was used as main communication for the X4 Rovers. The P3-AT would be programmed to lead the way to the designated path and to provide feedback to the main processor. The information such as position, velocity, and orientation would be relayed to the X4 Rovers to follow the path of the P3-AT. Obstacle tracking and avoidance, wireless communication, mapping, and programming of the leader and followers would be the tasks of the UGV team.

II. Trajectory Planning

In order for our rovers to navigate through a dynamic environment, our robots need a system to detect and track moving obstacles. The robots need to be able to avoid potentially harmful situations, adapt to new surroundings, and ideally, operate independently of human control. Figure 1 shows the path planning flow chart. The goal is to design a system to follow a collision-free path in the presence of moving obstacles within a limited sensing range.

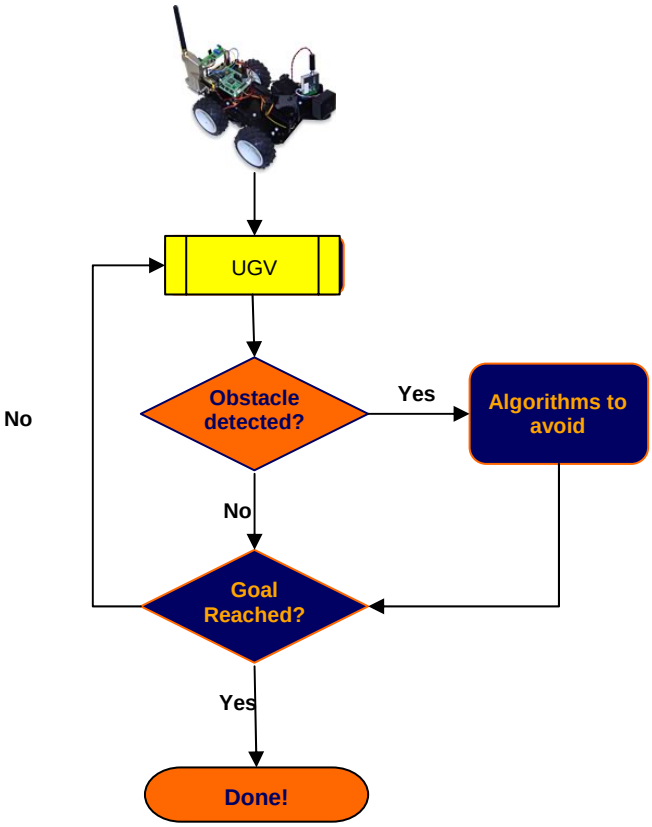


Figure 2.1: Flow Chart for Path Planning

2.1 Obstacle Avoidance

The main problem of path planning is how to deal with static and dynamic obstacles in real time. For the robots used in our research, the main sensors used are sonar. The assumption is that the sonar can used to track i-obstacles, the velocities of the robots remain constant, and that we know the size of the obstacles. To begin we use the following equations as the basis for obstacle avoidance.

(1) $x(t) = x_0 + c_1 t$

(2) $y(t) = a_0 + a_1 x(t) + a_2 x(t)^2$

(3) $[y(t) - y_{0,i}(t) - v_{i,y}t]^2 + [x(t) - x_{0,i}(t) - v_{i,x}t]^2 \geq (R + r_i)^2$

In (1), we set the x-position of the robot to move at a constant velocity c_1. For the steering function (2), a polynomial with three unknowns is chosen. $[x_{0,i}, y_{0,i}]$ are the initial positions for i-obstacles, and $[v_{i,x}, v_{i,y}]$ are the ith obstacles velocities. R is the radius around the robot to account for the size of it, and r_i is for the size of detected obstacle. To solve for these unknowns the following equations are used to evaluate (1), (2), (3) at the initial and final values.

(4) $y_0(t) = a_0 + a_1 x_0(t) + a_2 x_0(t)^2$

(5) $y_f(t) = a_0 + a_1 x_f(t) + a_2 x_f(t)^2$

(6) $x_f = x_0 + c_1 t_f$

Using substitution the above equations can be used to solve for a_1 and a_0 in terms of a_2.

(7) $b_1 = \dfrac{y_f x_0 - y_0 x_f}{x_0 - x_f}$

(8) $b_2 = \dfrac{y_f - y_0}{x_f - x_0}$

(9) $b_3 = x_0 x_f$

(10) $b_4 = x_f + x_0$

(11) $a_0 = b_1 + a_2 b_3$

(12) $a_1 = b_2 - a_2 b_4$

After obtaining a_0 and a_1 in terms of a_2, equations (11) and (12) can be substituted into the steering function (2), and equation (2) can now be substituted into the obstacle avoiding criteria equation (3).

(13) $y(t) = b_1 + b_2 x(t) + a_2[b_3 - b_4 x(t) + x(t)^2]$

(14)

$$[b_1 + b_2 x(t) + a_2(b_3 - b_4 x(t) + x(t)^2) - y_{0,i}(t) - v_{0,y}t]^2 + [x(t) - x_{0,i}(t) - v_{0,x}(t)]^2 \geq (R + r_i)^2$$

(15) $h_1 = b_1 + b_2 x(t) - y_{0,i_y}(t) - v_{0,i_y}t$

(16) $h_2 = x(t) - x_{0,i}(t) - v_{0,i}t$

(17) $h_3 = b_3 - b_4 x(t) + x(t)^2$

(18) $[a_2 h_3 + h_1]^2 + [h_2]^2 \geq (R + r_i)^2$

Equation (18) can be expanded and put into equation which can be easily solved for a_2 using the quadratic formula.

(19) $(h_3^2)a_2^2 + (2h_3h_1)a_2 + [h_1^2 + h_2^2 - (R + r_i)] \geq 0$

These values can be substituted into equation (13) to provide the new trajectory when an obstacle is predicted to collide with the robot.

2.2 Simulation

Solving the a_2 values requires the use of programming due to numerous parameters. MATLAB was the programming language of choice due to the overall ease of manipulating matrices and plotting the results. The MATLAB code for the obstacle avoidance can be found in Appendix A. In order to simplify the task, the simulation of the robot avoiding one obstacle was implemented before applying to criteria to multiple obstacles.

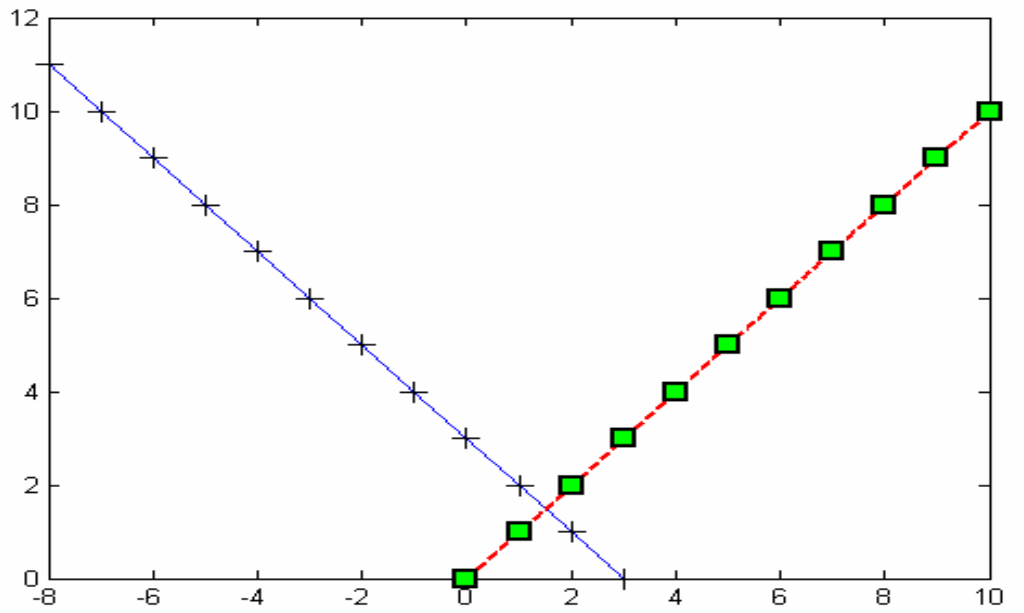


Figure 2.2: Robot(green) and Obstacle(blue) in Non-collision Course

To begin the simulation the robot is given an initial position of (0,0) and a final position of (10,10). The obstacle is given an initial position of (3,0) and a velocity of (-1,1). In Figure 2.a. the robot and obstacle are plotted in non-collision course to show how the robot will follow the given path unless it projects a possible collision.

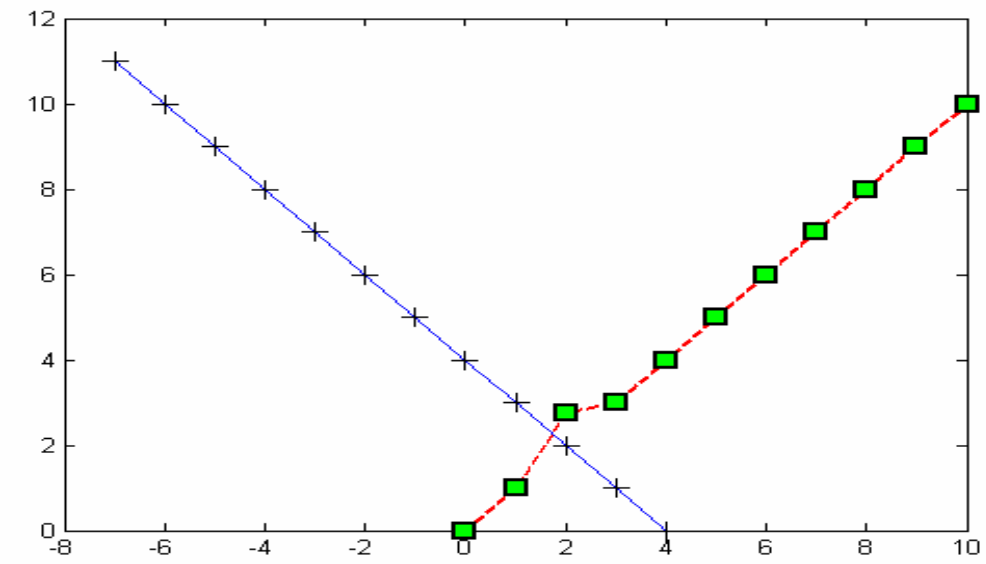


Figure 2.3: Robot(green) and Obstacle(blue) in Collision Course

In figure 2.b. the obstacle's position is changed to (4,0). This will cause a collision to occur at position (2,2). In this case, the robot applies the obstacle avoiding criteria to avoid the moving obstacle. The next process was to apply the same process for multiple obstacles.

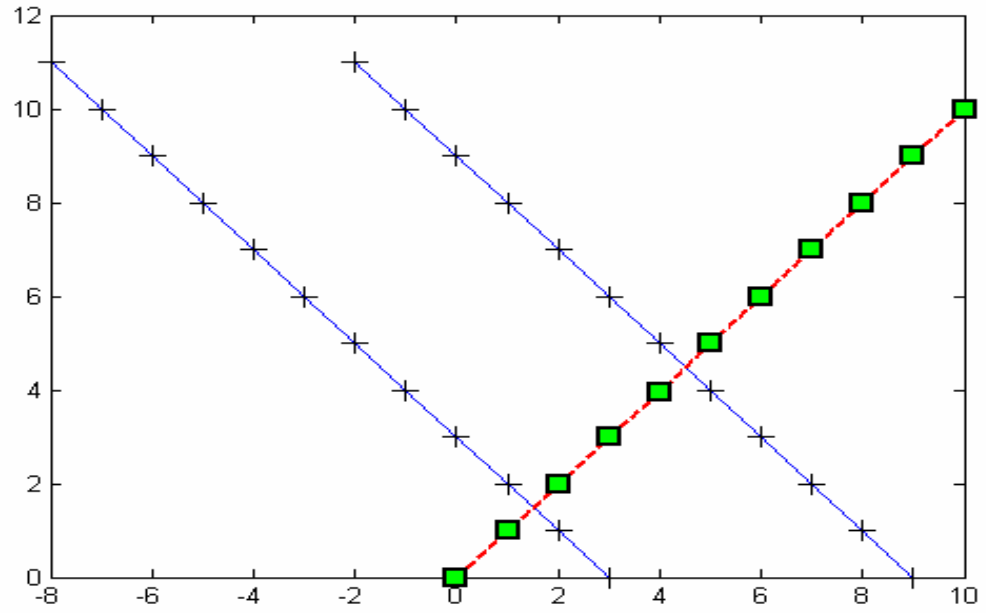


Figure 2.4: Robot(green) and Obstacles(blue) in Non-collision Course

Now the plot of two obstacles are placed in a non-collision course as shown in Figure 2.c. with initial positions (3,0) and (9,0) respectively and velocities of (-1,1).

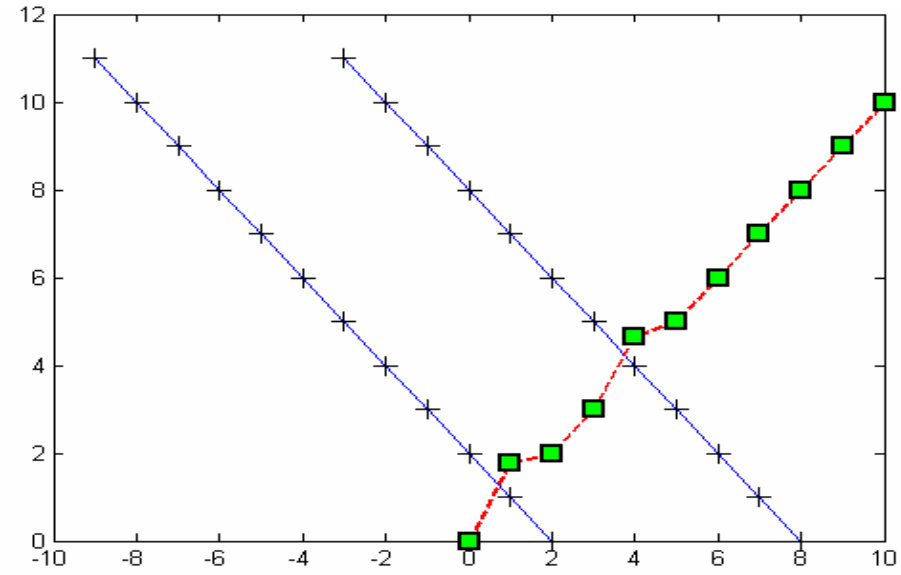


Figure 2.5: Robot(green) and Obstacles(blue) in Collision Course

The obstacle's initial positions are changed to (2,0) and (8,0) respectively, as shown in Figure 2.d. Collisions would occur at (2,2) and (4,4), but the robot applies the avoidance criteria to avoid the collisions.

2.3 Conclusion

It is shown how equations (1), (2), (3) can be applied to a system for use in dynamic obstacle avoidance and have shown the simulation results in MATLAB. There were a couple difficulties during the process. Initial simulations showed the robot avoiding obstacles that were parallel in paths that would cause a collision. This was a problem with the MATLAB code, as only the x positions were checked in determining if an obstacle was present. This was corrected by checking both the x and y positions. Another difficulty was applying the avoidance criteria to multiple obstacles. This was corrected by tracking obstacles individually. Improvements can be made in modifying the code for i-obstacles. Since our system uses sonar as the main sensor this can cause problems due to false positives, and further research is needed

III. Leader

The goal of the overall project is to have multiple robots (one Pioneer P3-AT and four X4 Rovers) travel from a point (x1,y1) to another point (x2,y2) as a networked and interactive group. To do this, leader-follower system was implemented using the P3-AT as the leader robot due to its more powerful capabilities. As the leader, the P3-AT is responsible for carrying out the following tasks: wireless communication, environment exploration, network system integration, X4 Rover detection, and simulation of some of the tasks using MATLAB. To achieve wireless

communication, the Lantronix WiBox was integrated on the P3-AT to provide a means of wireless communication between the robot and laptop computer, the central processor. Environment exploration was accomplished using the Mapper3Basic software developed by MobileRobots. Network system integration and X4 Rover detection were implemented to provide important navigation information to all the robots. Finally, a series of MATLAB simulations were done to verify, correct, and improve rover detection prior to actual implementation on the P3-AT. By accomplishing these tasks, the rest of the pack has more information about the terrain and obstacles, as well an object to track to get to the destination through a wireless network.

3.1 Objective and Assigned Tasks

In this part of the project, the focus is strictly on the P3-AT robot. The objective of this specific project is to use to powerful capabilities of the Pioneer P3-AT to autonomously lead a pack of robots from one destination to another. This specific part of the project will entail the following tasks that need to be either explored or accomplished for the successful development and implementation of the leader.

- Hardware Specifications
- P3-AT Software
- Wireless Communication
- Environment Exploration
- Network System Integration
- X4 Rover Detection

3.2 Hardware Specifications

3.2.1 Pioneer P3-AT

The Pioneer P3-AT is a powerful all-terrain robot developed by MobileRobots with a payload of 30 kg that is capable of autonomous navigation within a mapped universe.

Specifications:

- Dimensions: 50 cm x 49 cm x 26 cm
- 4 Wheels with Precise Encoders
- Microcontroller with Server Software
- 8 Front and 8 Rear Sonar Sensors

Figure 3.1: Pioneer P3-AT

3.2.2 Lantronix WiBox 2100E

The Lantronix WiBox was integrated on the P3-AT to provide a means of wireless communication between the robot and laptop computer. It was connected directly to the P3-AT through serial ports.

Specifications:

- 2 DB9 DTE Serial Ports
- Data Rates: 300 to 921,600 bps
- Characters: 7 or 8 Data Bits
- Parity: Odd, Even, None
- Stop Bits: 1 or 2
- Control Signals: RTS, CTS, DSR, DTR, DCD
- Flow Control: XON/XOFF, RTS/CTS

Figure 3.2: Lantronix WiBox

3.3 P3-AT Software

The P3-AT software was provided on a compact disc with the Pioneer P3-AT robot. The software came with open-source, C++ based, programs with extensive libraries as well as mapping software and user friendly GUIs.

3.3.1 Advanced Robotics Interface for Applications

Advanced Robotics Interface for Applications (ARIA) is a C++ based open-source development environment. Through ARIA, one can have access and modify the code for the P3-AT such as the microcontroller and accessory systems. Sensor usage, data acquisition, and synchronization examples can be found in ARIA.

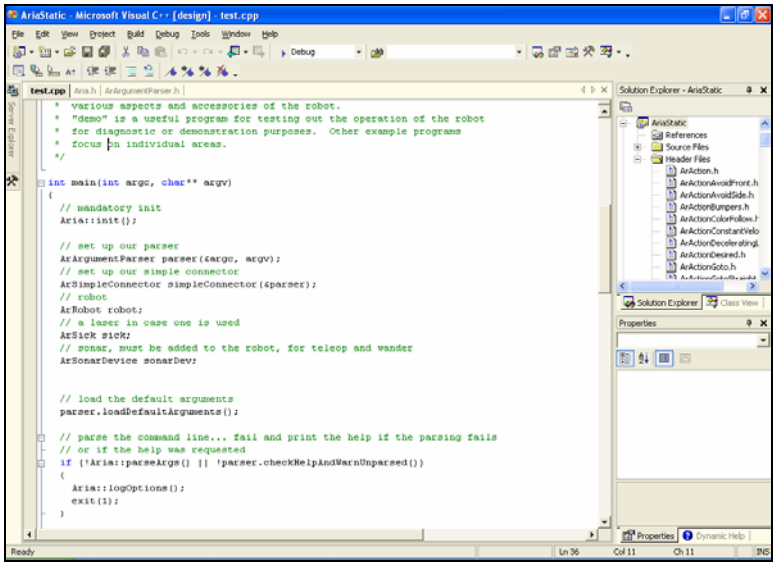


Figure 3.3: ARIA

3.3.2 Advanced Ro l Localization

Advanced Robotics Navigation and Localization (ARNL) is an extension of ARIA which allows localization to keep track of the robot. It also has navigation features which allow the robot to autonomously get to a given destination.

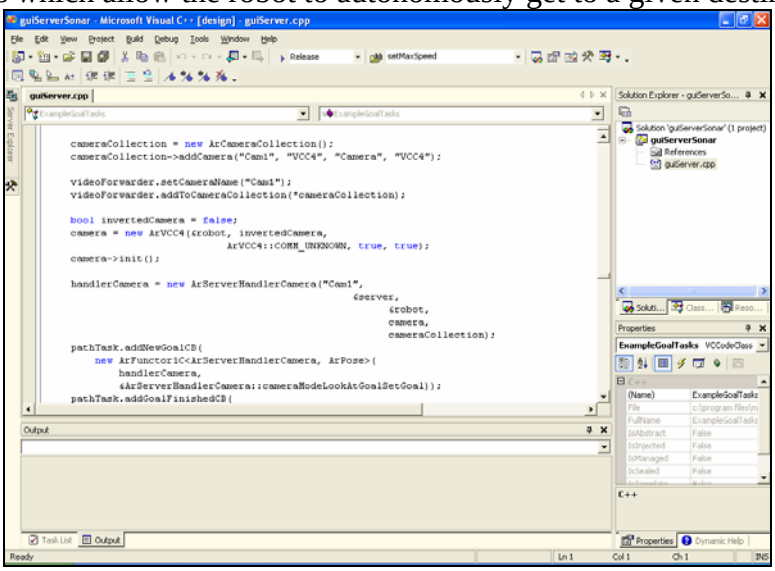


Figure 3.4: ARNL

3.3.3 Mapper3Basic

The Mapper3Basic application is a program that allows users to edit map lines and add goals, forbidden lines, home points, and other map objects. With the application, the P3-AT can use created maps to localize and navigate an environment.

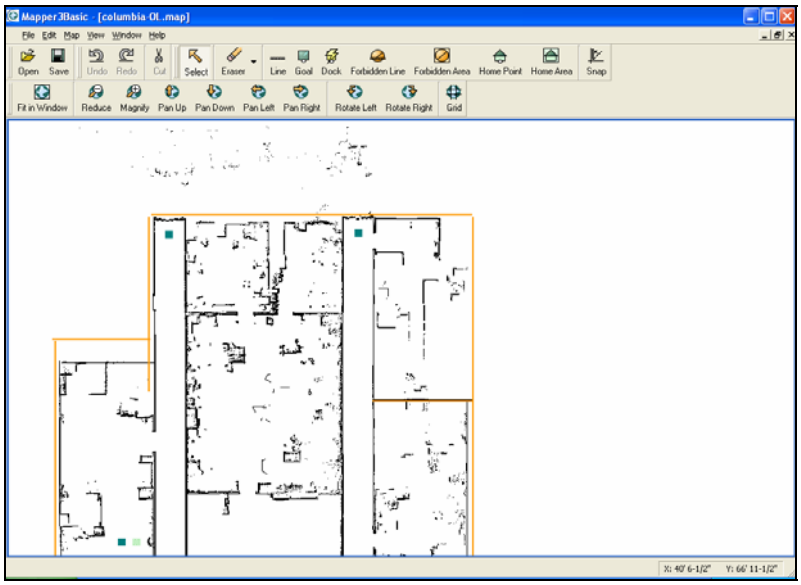


Figure 3.5: Mapper3Basic

3.3.4 MobileEyes

MobileEyes is a GUI application for remote robot control and monitoring. Connected with ARNL, MobileEyes acquires the P3-AT's position and displays it on the loaded map of an environment as well as sonar sensor readings, and other data.

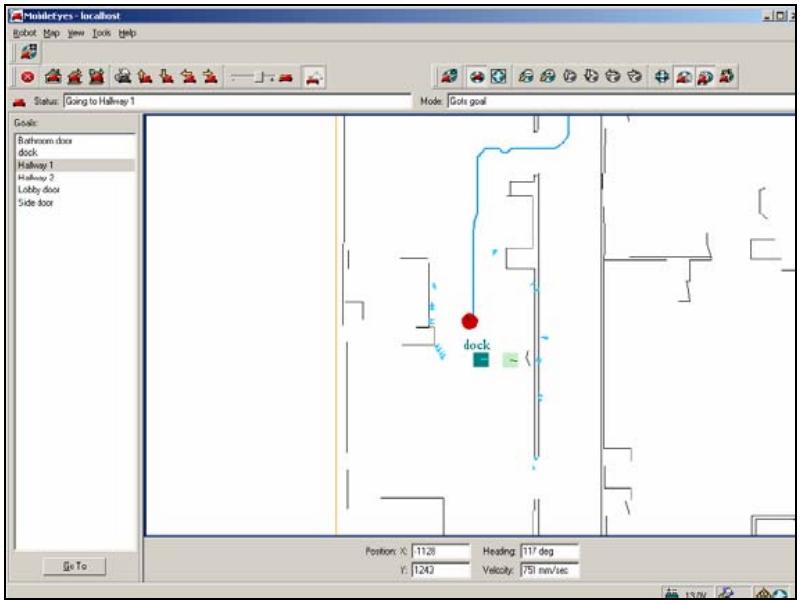


Figure 3.6: MobileEyes

3.4 Wireless Communication

As previously mentioned, the Lantronix WiBox was the peripheral used wirelessly link the Pioneer P3-AT with the laptop used as the central processor. In order to achieve this, some settings had to be configured on the WiBox to obtain communication between the two systems.

WiBox Settings:

- Channel 1
- Baud rate: 9600 bps
- Flow: None
- Remote IP Address: 129.115.1.100
- Port Number: 8101

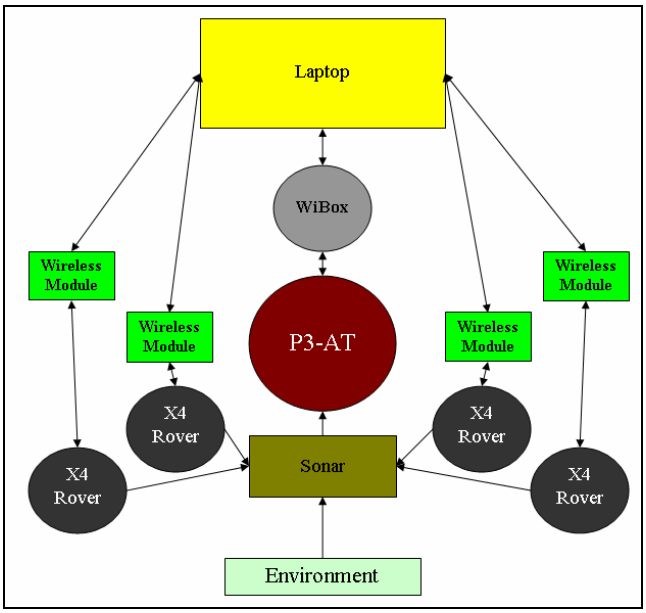


Figure 3.7: Wireless/Sensor Block Diagram

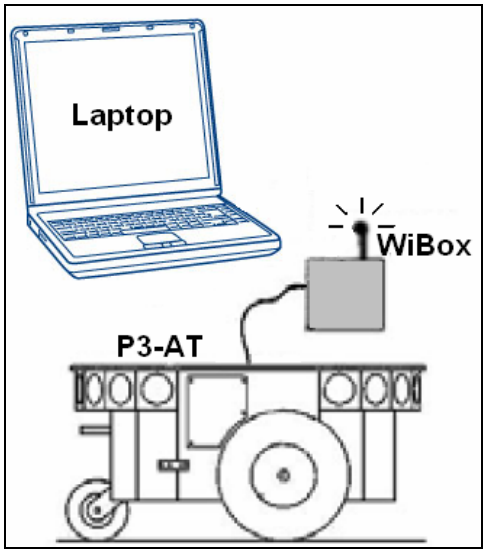
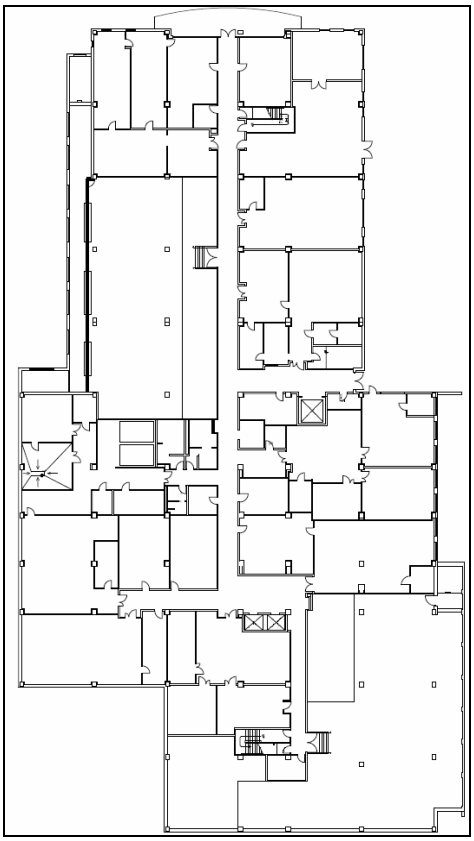


Figure 3.8: Wireless Configuration

3.5 Environment Exploration

In order for the P3-AT to be able to localize and navigate from one point to another, it must have a map of the environment created using the Mapper3Basic application. In this case, the research laboratory is located in the UTSA Bioscience and Engineering Building sublevel. Therefore, the testing environment became the larger hallway passages and laboratory room of the UTSA BSE Sublevel floor. In order to accurately recreate this environment in Mapper3Basic, the architectural schematics (floor plans) were used to obtain detailed information regarding the measurements of the hallways, passages, and rooms. This process allows the P3-AT to autonomously navigate and explore the specified environment. As a result of incorporating this software, the P3-AT is limited to the size, scale, and detail put into the map representation of the real world environment.



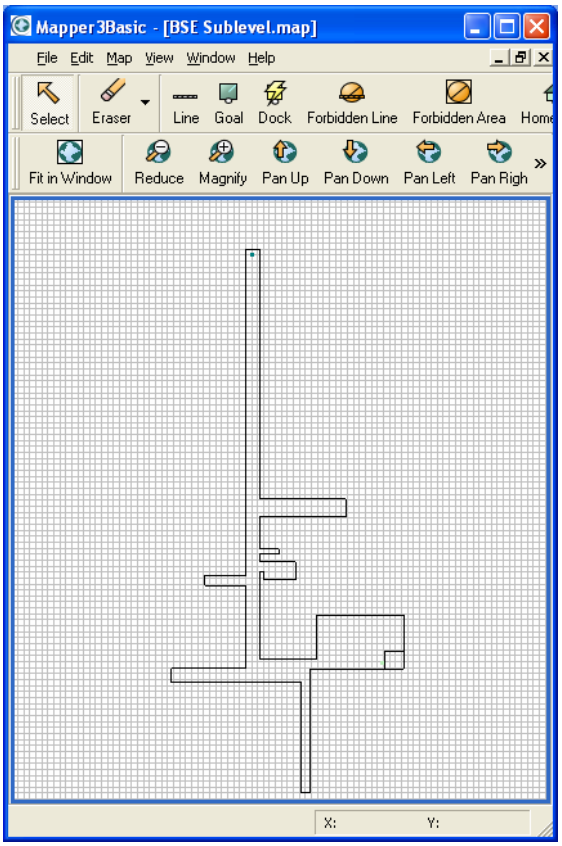


Figure 3.9: BSE Sublevel Floor Plan

Figure 3.10: Mapper3Basic Sublevel Map

3.6 Network System Integration

The system integration aspect using the P3-AT was designed to aid the significantly less powerful X4 Rovers with navigation. Unlike the P3-AT, the X4 Rovers have small memory storage for programming, relatively inaccurate encoders for the wheels, only two sonar sensors, and move at a much slower rate than the P3-AT. Therefore, the P3-AT's resources were used to provide crucial information to the Rovers through a text file accessible to any robot on the wireless network. The text file is constantly overwritten to provide the latest information regarding navigation.

Data Found in the Text File:

- P3-AT's Position: (x, y)
- P3-AT's Orientation: (Th)
- P3-AT's Velocity: (v)
- X4's Position*: (x_r, y_r)

* The returned position is of the X4 Rover closest to the P3-AT.

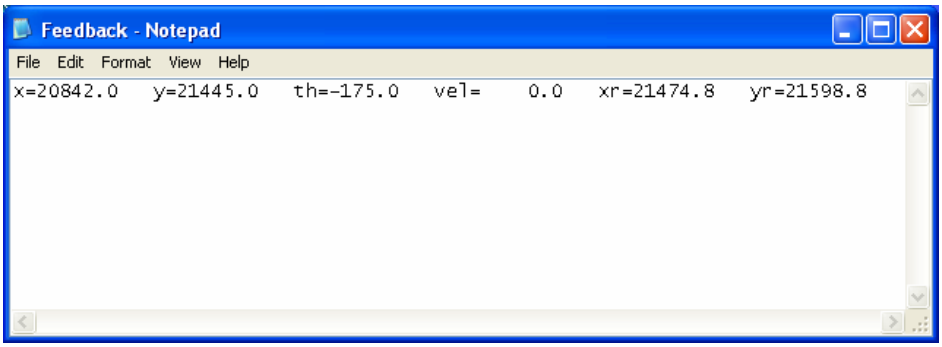


Figure 3.11: Text File Structure

3.7 X4 Rover Detection

Obtaining the position X4 rovers and including it in the previously mentioned text file greatly aids the rovers with navigation. Therefore, the sonar sensors on the P3-AT were used to accomplish this task.

3.7.1 Rover Detection Problem

Since the P3-AT uses sonar to detect any object in its near proximity, it was difficult to distinguish between random objects and the X4 rovers. The sonar sensors will detect any obstacle, object, or person within a five feet radius. Detecting which, if any, of these readings was a rover became a difficult task.

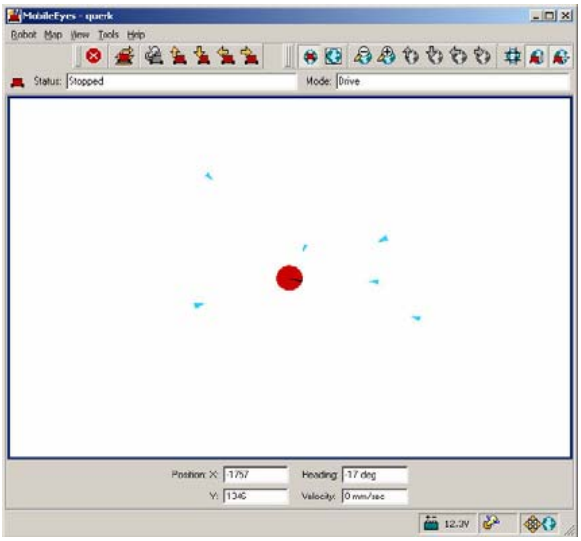


Figure 3.12: Sonar Readings

3.7.2 Rover Detection Solution

To solve this problem, an area with adjustable variables was specified to prevent sonar readings from random objects to be confused with the X4 rovers. This rover detection area is located directly behind the P3-AT and only sonar readings within this area are considered for determining the X4 position. Any readings outside of this area are disregarded as possible X4 rovers but are still used for navigation and obstacle avoidance purposes. If there are no sonar readings within the area, then the text file returns an "OoR" (Out of Range) string rather than a numerical value for the X4 position.

$$x1 = range \times \cos(theta - \pi/2)/2 + x\_ref \qquad x2 = range \times \cos(theta - \pi/2)/2 + x\_ref2$$
$$y1 = range \times \sin(theta - \pi/2)/2 + y\_ref \qquad y2 = range \times \sin(theta - \pi/2)/2 + y\_ref2$$
$$x3 = range \times \cos(theta + \pi/2)/2 + x\_ref2 \qquad x4 = range \times \cos(theta + \pi/2)/2 + x\_ref$$
$$y3 = range \times \sin(theta + \pi/2)/2 + y\_ref2 \qquad y4 = range \times \sin(theta + \pi/2)/2 - y\_ref$$

Figure 3.13: Rover Detection Area Equations

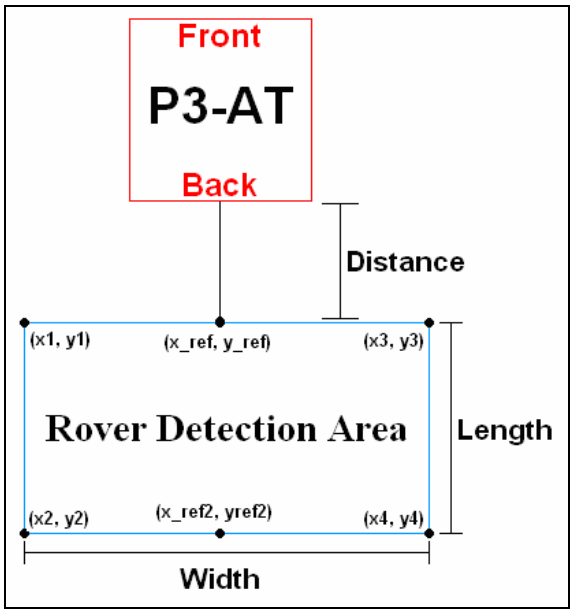


Figure 3.14: Rover Detection Area

3.7.3 MATLAB Simulation

Multiple MatLab simulations were done using this concept before it was implemented on the P3-AT's program code. A MATLAB simulation of the rover detection area can be found below.

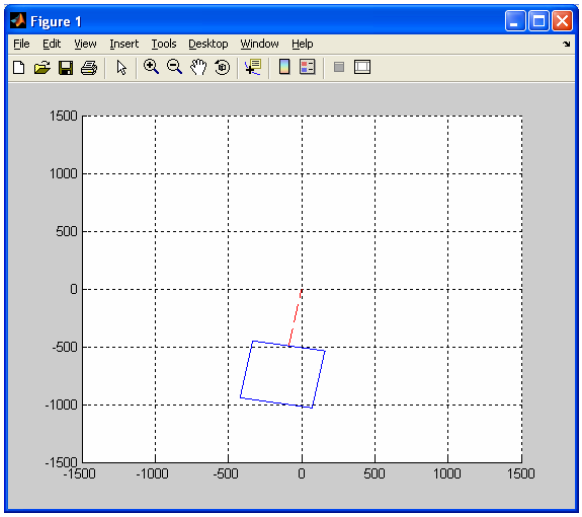


Figure 3.15: MATLAB Simlation

3.8 Conclusion

After all of the previous tasks were implemented on the Pioneer P3-AT, the robot was able to successfully navigate the UTSA Bioscience and Engineering sublevel by specifying a designated position. While autonomously navigating the building using a created map, the P3-AT also sends back feedback information to the X4 Rovers to help them navigate the floor as well. The accomplished tasks can be used in the future and improved upon to further research autonomous swarm navigation

IV. Follower

In order to make use of a leader-follower system using only a single navigational rover and multiple cheaper rovers, some means must be employed in order to allow the small rovers to take advantage of the main rover's path planning. One proposed means of solving this problem is using a technique known as object tracking, in which, opposite to obstacle avoidance, a moving obstacle is tracked or followed. In order to develop such a system, a basic fuzzy logic system could be developed, however, we anticipate much noise from the hardware given that our object detection sensors are ultrasonic rangefinders. Thus, we need a system which will filter out large amounts of noise. One well known filter with this capability is the Kalman filter. Therefore, we propose to use a Kalman filter for object tracking in the follower rovers.

4.1 Configuration

As previously stated, the main rover which will was used as the leader is the Pioneer P3-AT all terrain robot, equipped with 16 ultrasonic rangefinders, 4 dc motors with encoders, and onboard microcontroller. Also, since its path planning software, ARIA, is run from an external computer, the P3-AT is equipped with a serial wireless communication device known as a WiBox. For the follower part of the formation, 4 X4 rovers were used, each equipped with onboard OOPic-R microcontroller, 2 ultrasonic rangefinders, magnetic compass, and fast wireless communication module. Because of the limited capabilities of the microcontroller, the object tracking algorithm was run using an external, central control computer, the same computer used to run ARIA for the leader. A fifth wireless module was attached via serial port to the computer to allow communication with the X4 rovers. Finally, MATLAB was selected as the program in which the algorithm would be run because of its built-in serial communication capabilities.

4.2 Object Tracking (Kalman Filter)

Based on the assumption that our hardware will be relatively noisy due to the characteristics of sonar, the Kalman filter was selected as the best candidate for the object tracking algorithm. Developed originally by Rudolph E. Kalman, the Kalman filter is a very effective recursive filter designed to use the previous state and current noisy measurements of a dynamic system to estimate the current state of that system. The filter uses a two-step method to arrive at the new estimate. First, the current state of the system is predicted using the previous state of the system. Secondly, the prediction from the first step is refined using noisy measurements of the current state of the system. This generates a more accurate estimate of the actual state of the system. The actual formulas for the filter can be seen in below.

Predict:

state: $\mathbf{x}_{k|k-1} = \mathbf{F}_k\mathbf{x}_{k-1|k-1}$

variation: $\mathbf{P}_{k|k-1} = \mathbf{F}_k\mathbf{P}_{k-1|k-1}\mathbf{F}_k^T+\mathbf{Q}_k$

Update:

Measurement residual: $\mathbf{y}_k = \mathbf{z}_k-\mathbf{H}_k\mathbf{x}_{k|k-1}$

Residual variation: $\mathbf{S}k = \mathbf{H}k\mathbf{P}k|k\text{-}1\mathbf{H}kT+\mathbf{R}k$

Optimal Kalman Gain: $\mathbf{K}k = \mathbf{P}k|k\text{-}1\mathbf{H}kT\mathbf{S}k\text{-}1$

State estimate: $\mathbf{x}k|k = \mathbf{x}k|k\text{-}1+\mathbf{K}kyk$

Variation estimate: $\mathbf{P}k|k = (\boldsymbol{I}\text{-}\mathbf{K}k\mathbf{H}k)\mathbf{P}k|k\text{-}1$

4.3 Simulation

After writing a simple version of a Kalman filter to be used as an object tracker, simulation was done by feeding the filter a set of simulated sensor values, generated by adding random noise at each step to a predetermined system model. It is considered a simple filter because Qk and Rk, system and measurement noise, are set as constant values. Results of simulation can be seen in Figure 3a.

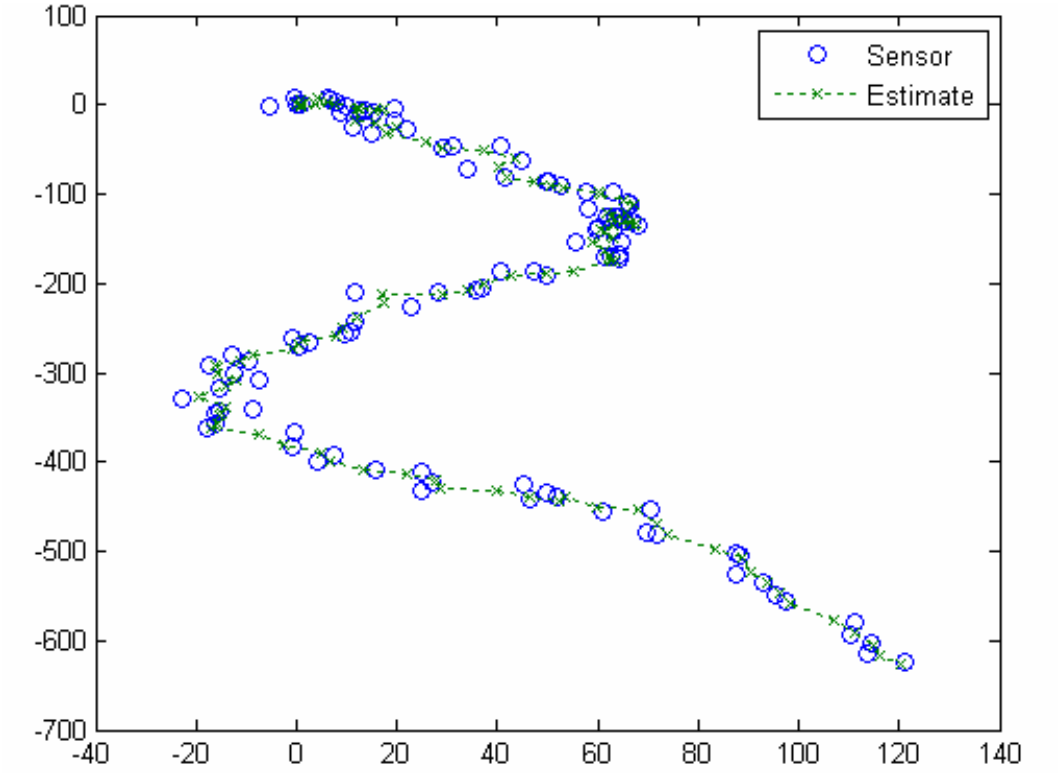


Figure 4.1: Kalman Filter with Constant Noise

As the above simulation shows, if the noise values that are set are not the same as the actual values of the noise in the system, the tracker will not track as smoothly the overall path of the object. In an attempt deal with this, a fuzzy logic controller was developed using the MATLAB FID editor. Using this controller, the noise values are changed based on the difference between the predicted location of the object and the measured location based on sensor feedback: the larger the difference, the larger the noise, and vice versa. Using this new setup and the same sensor values from the previous simulation, a new was done. Results for this can be seen below in Figure 3b.

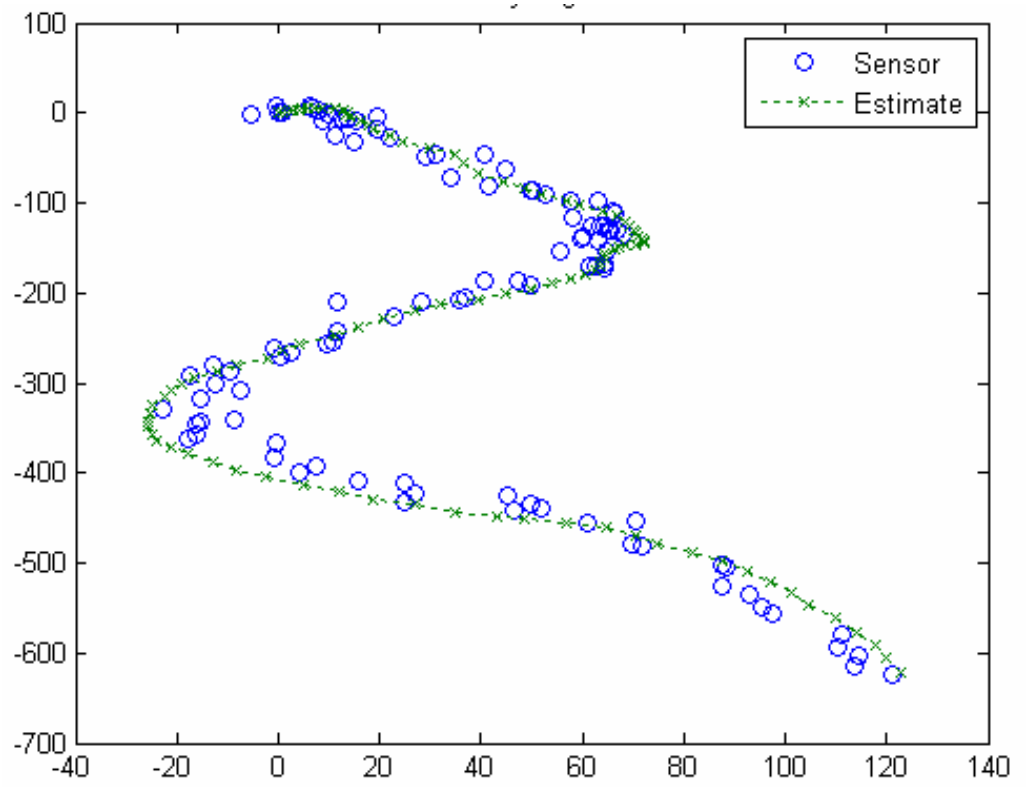


Figure 4.2: Kalman Filter with Fuzzy Controller on Noise

This modification can be seen to help smooth out the object tracking, effectively filtering out most of the noise. This improved performance over the constant noise method was a significant factor in the decision-making process on which algorithm to use.

4.4 Implementation

The algorithm selected to be used, a Kalman filter with a fuzzy noise controller, was much too complicated for the X4 rover's on-board microcontroller to handle by itself. Thus, the first step of implementation was to write code expressly to facilitate wireless send/receive from the central computer to the follower rovers. This is when the decision was made to make use of MATLAB for the main tracking algorithm because of ease of use of the serial functions as opposed to those in C or C++.

4.4.1 Wireless Serial in MATLAB

In order to setup the system to use the fast wireless communication module, a crossover cable as described in the TotalRobots website was constructed. Initially, attempts were made to use the software that comes with the module in order to transmit and receive, but we finally had to write our own MATLAB code instead. As stated in the data sheet for the wireless module, there is a specific format for the data stream to be sent wirelessly, and the values sent are binary, so we made

use of the fread and fwrite commands in MATLAB, with the data stream being represented as a single-row matrix. Also, the serial port was configured inside MATLAB to have the specs indicated by the data sheets on the wireless, including setting baud rate to 19200. Code for this can be seen in Appendix C. This took care of the central computer wireless communication.

4.4.2 Rover OOPic Microcontroller

Next, code was written for the OOPic microcontroller to allow it to respond to the command data that we intended to send it, namely compass heading and time of motion. It also had to allow for transmission of the command data and sensor data between the rover and the central computer. The way the final flow of the code for this program worked is as follows:

(i) Read sensor data (compass, sonar) and transmit to computer.
(ii) Wait for computer to respond with calculation done command and command data
(iii) Match heading in command data
(iv) Move forward for received amount of time
(v) Send "done" to computer
(vi) Repeat a-e

Completion of motion or calculation was indicated using the lead bit transmitted by the wireless. Specific numbers were used to indicated each of the different completed tasks (i.e. done moving, send 2).

4.4.3 Motion Control

Third, we had to develop code that would translate a desired position to move to given in MATLAB into heading and time of motion to transmit to the rover wirelessly. Also, the method had to consider the fact that the rover would be reading compass headings, which use a different angle chart (0-360 clockwise) as opposed to standard polar coordinates (0-360 counter-clockwise, shifted 90 degrees from compass heading). Furthermore, the code we developed allowed for a relative frame of reference to be set in the case that the user wishes to specify this at the beginning of the autonomous run.

4.4.4 Kalman Filter

Our fourth task in order to make the system operational was to port the Kalman filter from simulation to allow for use in the real world. The filter from simulation was based on all the sensor data being currently stored in a matrix, but we needed to make it into a single-step filter. This was done by developing a Kalman function which takes in the previous state, previous variance $\mathbf{P}_k$, and the current sensor readings, and outputs present state and present variance. This would allow

us to incorporate it into our other code very easily. Also, the fuzzy noise controller was included and required no modification or porting.

4.4.5 System Integration

Finally, all of these different functions and task were integrated together to develop a fully functional object tracking system with a fuzzy noise controller. The overall flowchart for the entire system can be seen in Figure 3c.

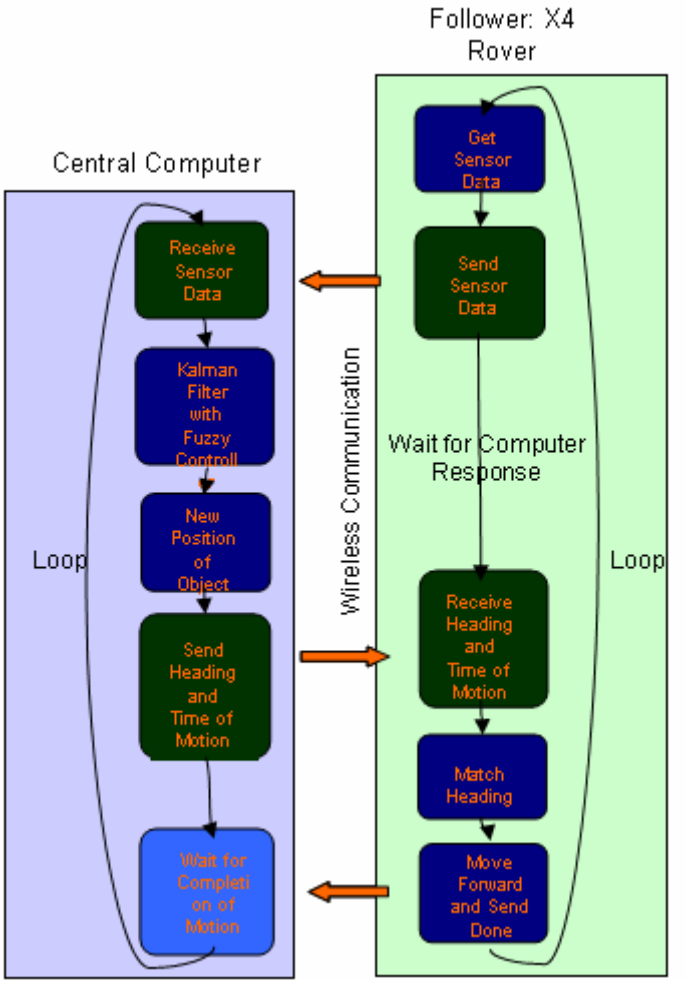


Figure 4.3: Object Tracking System Flowchart

4.5 Conclusion

Thus, we have proven the effectiveness of using object tracking via Kalman filtering as a means to implement the follower side of a lead-follower formation, as we were able to implement this concept using inexpensive followers constructed from hobbyist-level hardware. Also, the use of a fuzzy controller to set the noise matrices in the Kalman filter is a viable and useful means of cutting down on inaccuracies generated from using noisy hardware and sensors. Finally, improvements that could be made in our follower system would be better sensors for tracking the leader, such as video cameras. This is because

our system currently uses sonar, which have no way of determining whether or not the object detected is the object to be tracked.

V. Wireless Communication Network

5.1 Overview of the Wireless Network

The UGV's wireless network is composed of several Fast Wireless Communication Modules (FWCM). The FWCM is a 433 MHz RF communication module manufactured by Designer System Inc. Each FWCM can support bi-directional wireless communication with 100% data accuracy over a distance of up to 300 meters. The module is designed to work with both the OOPic Microcontroller through I^2C connection and the computer through RS232 serial communication. When configured for normal mode, a FWCM network can have at most 255 modules with each one assigned a unique address. The FWCM can also operate in broadcast mode or autonomous mode, but these modes are not used in this research project. The power range for this module is either 6-16 V from external power supply or 5 V from I^2C connection.

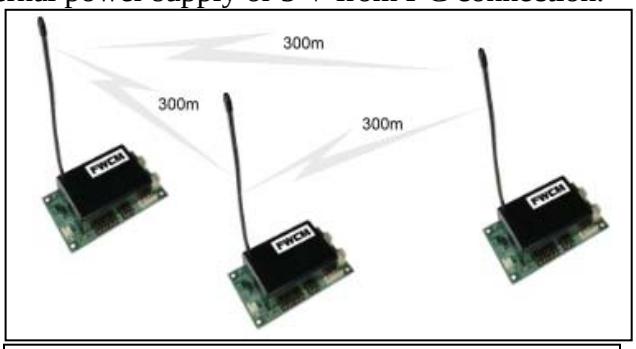


Figure 5.1: FWCM Network

5.2 Hardware Connection

The FWCM must be assigned a unique I^2C bus address by setting the jumpers in the ADDRESS section. The default value is 00 when both jumpers are left on. In this research, the default values will be used for all modules. However, if there is a magnetometer on board, the default value cannot be used because it will conflict with the magnetometer. To configure the FWCM to a different address, follow the table below.

Address XX	A0	A1
00	ON	ON
01	OFF	ON
10	ON	OFF
11	OFF	OFF

Figure 5.2: FWCM Local Address Table

Then connect the two jumpers across the SDA and SCL header pins in the PULL UP section. This will provide enough source current for I^2C communication with either the computer or the OOPic. The PRG pins of the OOPic will be connected to the FWCM's I2C Output. The following schematic shows the links.

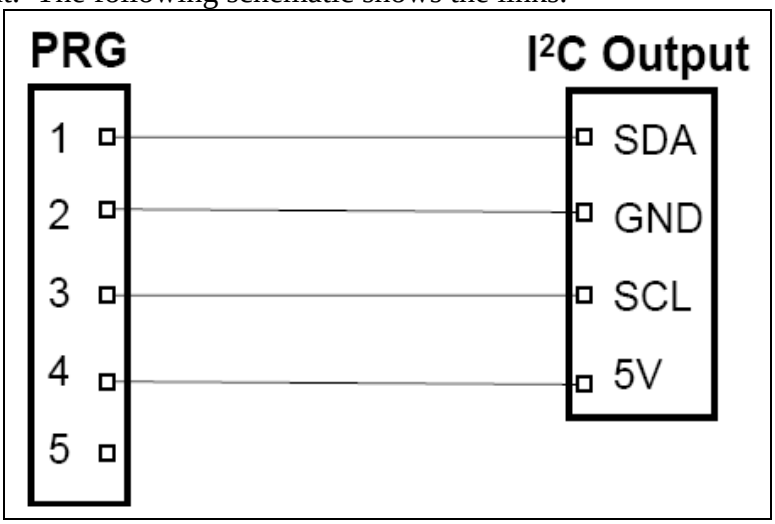


Figure 5.3: I2C Connection (OOPic-R)

To connect the FWCM to the computer, a serial cable must be used. The schematic for the serial communication is shown below.

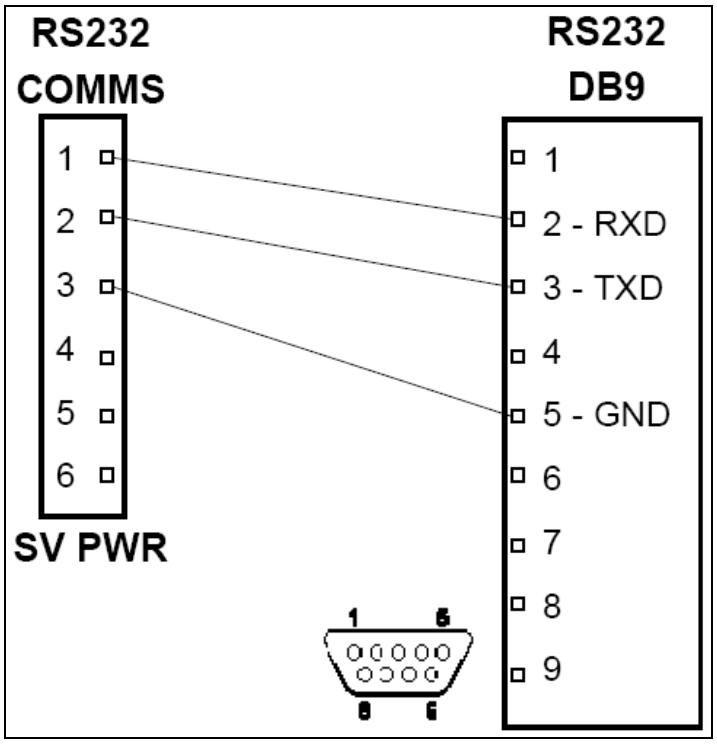


Figure 5.4: RS232 Connection from Computer

5.3 Software Implementation

A FWCM connected to an OOPic must be set up before use by specifying I^2C address, number of bytes to read or write, I^2C addressing mode, increment option and local node address. Most of these steps can be skipped for the module connected to the computer, except the I^2C address and the local node address. The list of write/read registers is described in the data sheet of the FWCM.

The serial communication between the computer and FWCM must be set up as followed:

19200 baud rate
8 data bits
1 stop bits
No parity bit
No handshaking (if configurable)

MATLAB is employed to handle serial communication. Because a read can be executed after a write, a write to a non existent is performed to prevent corrupting data in other remote FWCM. Two operating schemes are used to implement a wireless network: parallel and sequential network.

1. Parallel:

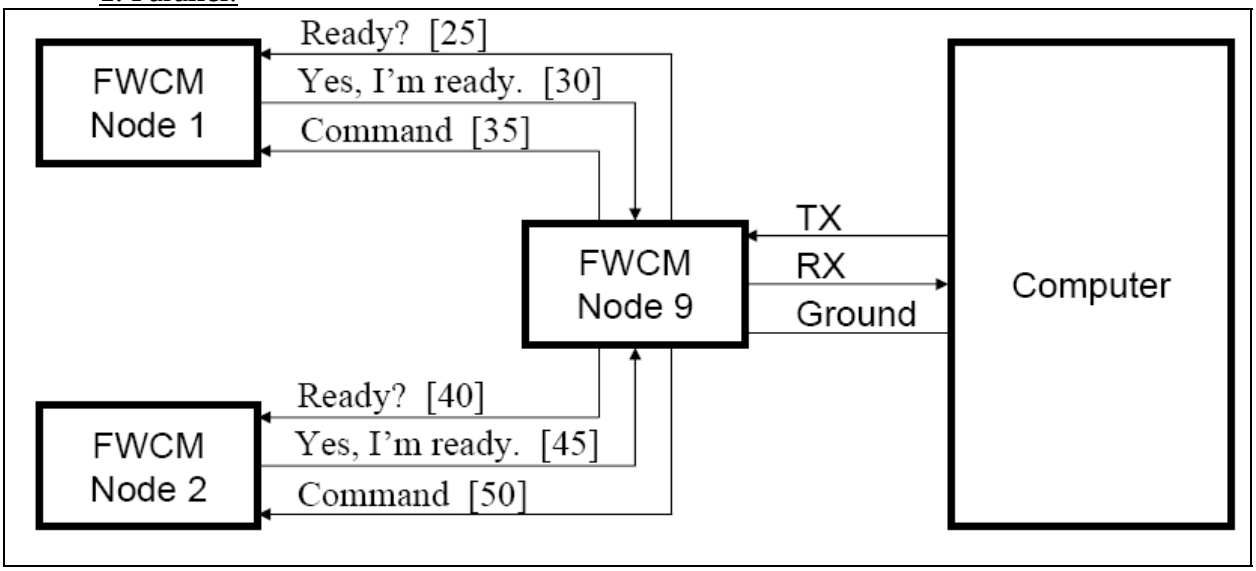


Figure 5.5: Parallel Configuration

In this network the computer will keep polling every FWCM. If a FWCM is busy, the computer will come back later. If not, the computer will record the position sent by that FWCM and give it a command. The stream of data is 9 byte long with 1 signal byte and 8 command and data bytes. The data is imported into a text file. The time out is set at 2 seconds.

Each rover will not talk to each other in this network because it can communicate directly with the computer. This network is very efficient because the rover can execute the command on its own after it talks to the computer. However, there are some potential status errors which mean when the computer mistakenly think that an idle rover is still busy executing its last command. An idle rover can be skipped at most twice.

b. Sequential:

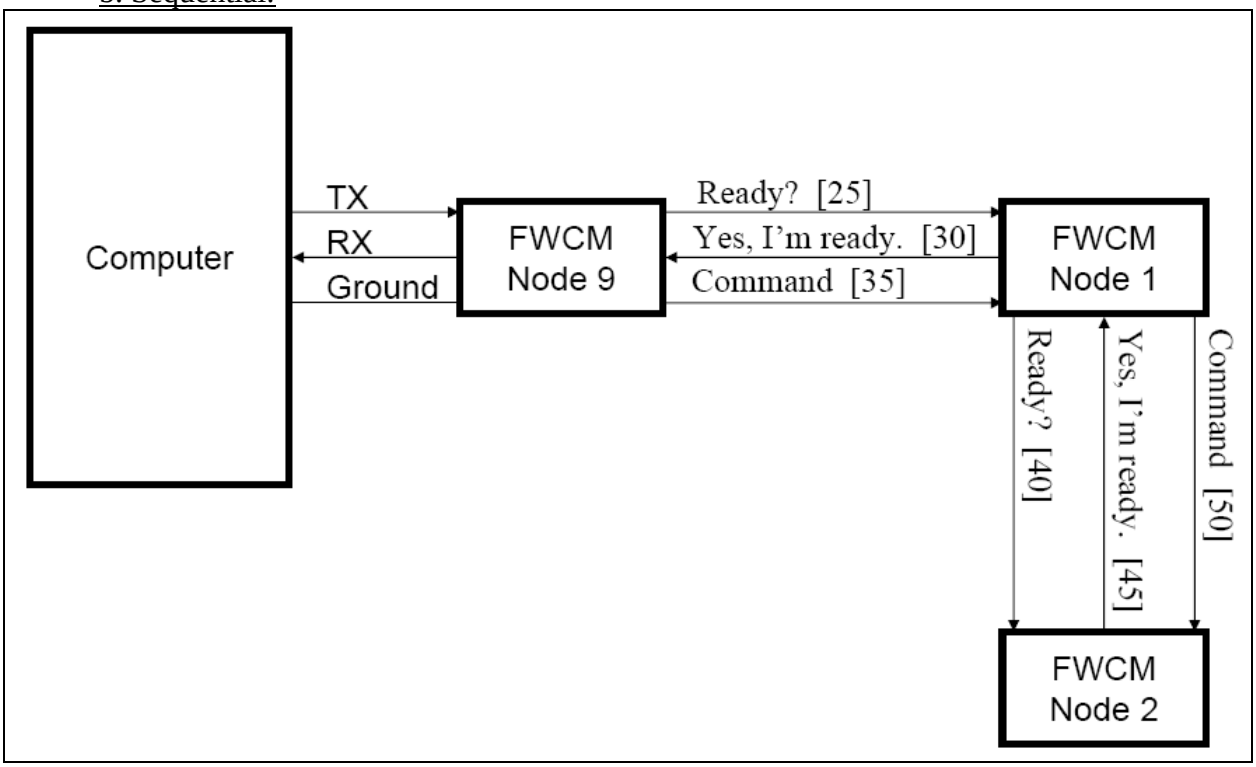


Figure 5.6: Sequential Configuration

In this network the rover carrying FWCM 1 will act as a bridge between the computer and FWCM 2. FWCM 1 will receive command for itself and FWCM 2. Then it will wait until FWCM 2 finishes its job and send the command to FWCM 2. This network is not as efficient as parallel network but it has 2 advantages over the parallel network: no status error and double range of operation.

5.4 Conclusion

The two schemes of wireless communication are tested and verified thoroughly. The parallel network is more efficient but suffers potential status error. On the other hand, the sequential network, though lack efficiency, performs without any error and poses no limitations on operation range. Generally, the two networks are very reliable due to the data verification feature of the wireless modules and the hand-shaking procedures implemented.

However, at this point both schemes operate separately. The programmer has to choose either one to apply to the network. Therefore, future improvements are recommended to combine these two schemes into a single network. To implement this idea, the computer will first assume that all FWCM are in the operation range and apply the parallel scheme. Then if any FWCM does not reply to the computer for at least a number of times (determined by the programmer), the computer will switch to sequential and try to make contact with that FWCM.

VI. Summary

Overall, we have proven that a lead-follower system is a viable and helpful possibility as a means to field a group of fully autonomous ground vehicles. First, we have shown a viable algorithm to allow a robot to avoid a moving obstacle. Secondly, we demonstrated the implementation of a path planning system for the lead rover as well as ability to locate the following rover and data logging of the current state of the lead rover, such as heading, speed, and location. Thirdly, we have developed an object tracking algorithm and a system setup to allow execution of the algorithm from a central computer and wirelessly communicate with the smaller rovers. Finally, a reliable wireless network was designed and implemented incorporating handshaking techniques, with an alternate mode of operation allowing for relayed transmission.

VII. References

"Fast RS232 Serial & I2C Wireless Communications Module." TotalRobots. 2007. Designer Systems. 22 June 2007 <http://www.totalrobots.com/pdfs/DS-FWCM_V1.03.pdf>.

"Kalman Filter." Wikipedia. 8 July 2007. 20 June 2007 <http://en.wikipedia.org/wiki/Kalman_filter>.

Kobayashi, Kazuyuki, Ka C. Cheok, Kajiro Watanabe, and Fumio Munekata. "Accurate Differential Global Positioning System Via Fuzzy Logic Kalman Filter Sensor Fusion Technique." IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS 45 (1998). IEEExplore. San Antonio. 1 Aug. 2007. Keyword: Kalman Filter and Fuzzy Logic.

Lantronix, "WiBox2100E User Guide Revision F", Lantronix Inc., June 2006

MobileRobots, "Pioneer 3 Operations Manual Version 4", MobileRobots Inc., January 2007

MobileRobots, "SonARNL with MobileEyes Installation and Operations Manual Version 2.1", MobileRobots Inc., January 2007

Pike, John. "Man Portable Robotic System (MPRS/URBOT)." GlobalSecurity. 27 Apr. 2005. 30 July 2007 <http://www.globalsecurity.org/military/systems/ground/mprs.htm>.

"Unmanned Ground Vehicle." Wikipedia. 23 July 2007. 30 July 2007 <http://en.wikipedia.org/wiki/UGV>.

Xiong, Zhilan, Yanling Hao, Jinchen Wei, and Lijuan Li. Fuzzy Adaptive Kalman Filter for Marine INS/GPS Navigation. Proceedings of the IEEE. International Conference on Mechatronics, July 2005. IEEE, 2005.

Z. Qu, J. Wang, C. E. Plaisted, "A New Analytical Solution to Mobile Robot Trajectory Generation in the Presence of Moving Obstacles," *IEEE Transactions on Robotics, vol. 20, pp. 978-993, December 2004.*