SMALL SCALE IMPLEMENTATION OF A ROBOTIC URBAN SEARCH AND RESCUE NETWORK

By

Steven G. Kibler

Recommended: [illegible]

[illegible]

[illegible]

Advisory Committee Chair

[illegible]

Chair, Department of Electrical and Computer Engineering

Approved: [illegible]

Dean, College of Engineering and Mines

[illegible]

Dean of the Graduate School

April 3, 2012

Date

SMALL SCALE IMPLEMENTATION OF A ROBOTIC URBAN SEARCH AND RESCUE NETWORK

By

Steven G. Kibler

Recommended: ____

____

____
Advisory Committee Chair

____
Chair, Department of Electrical and Computer Engineering

Approved: ____
Dean, College of Engineering and Mines

____
Dean of the Graduate School

____
Date

SMALL SCALE IMPLEMENTATION OF A ROBOTIC URBAN SEARCH AND RESCUE NETWORK

A

THESIS

Presented to the Faculty

of the University of Alaska Fairbanks

in Partial Fulfillment of the Requirements

for the Degree of

MASTER OF SCIENCE

By

Steven G Kibler. B.S.

Fairbanks, Alaska

May 2012

Abstract

With the advancement of robotics technologies, it is now possible to use robots for high risk jobs that have historically been accomplished by humans. One such example is the use of robots for Urban Search and Rescue (USR): finding chemical spills, fires, or human survivors in disaster areas. With the ability to include inexpensive wireless transceivers, it is possible to network numerous robots as part of a swarm that can explore an area much more expeditiously than a single robot can. With the inclusion of wireless capabilities comes the necessity to create a protocol for the communication between robots. Also necessary is the creation of an exploration protocol that allows the network of robots to explore such a building or search area in as little time as possible yet as accurately as possible.

This thesis covers the development of such a network of robots, starting with the hardware/software co-design, the individual robots' control mechanisms, and their mapping and communications protocols.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

For as long as there have been natural disasters, there has been a need to search for survivors amongst the rubble. The search for survivors can often place the search and rescue team in great peril, and injury and death among rescuers is an unfortunate possibility and does happen. More and more often we find that man-made disasters also require rescue teams to enter dangerous conditions to search for survivors. Because the environments in which these natural and man-made disasters occur are so varied, humans are the perfect machine (for now) to perform the rescue operations, because the human body is so versatile. It can traverse nearly any terrain: dry or wet, hot or cold, steep or flat, sandy or rocky or overgrown with dense trees and shrubbery. It is also capable of grasping small objects gently and moving them with great precision, as well as lifting heavy objects and carrying them for long distances. It has a great array of built-in sensors (eyes, ears, nose, etc.) that make it capable of detecting many environmental conditions that are dangerous. Most notable, though, is its mental ability. It has the unique ability to create solutions to problems that it may have never encountered before, and is not limited to performing only those tasks for which it has previously written algorithms. In computer engineering terms, it is capable of writing its own algorithms to handle nearly any condition, as those conditions become known to it. Because environments that require rescue teams are often some of the most chaotic, it is precisely that ability that makes the human being the most capable search and rescue agent. It is an incredible machine for performing such tasks. However, it comes with a great cost; the human body is fragile and cannot be easily or quickly fixed, nor can it be brought back from death.

While it is true that the human body is the best machine for search and rescue operations (at this point in time), there are many environments and situations in USR operations in which a robot could be employed to remove much or most of the danger that the human experiences. Robots are quickly becoming versatile enough to handle terrain that would have stymied their predecessors. They can now climb stairs, go over rubble fields, move over many rocky and steep terrains, can handle the heat and the cold, and can also handle moving through some amount of water. They carry sensors that actually give them a greater sensitivity to certain chemicals, such as smoke and carbon monoxide, than a human is capable of detecting. They can carry cameras that are able to detect a wider range of electromagnetic radiation (light) than a human eye can see, such as infrared light, which is important in cases where a fire could be involved. While robots do lack the human mind's creative problem solving ability and (with the exception of artificial intelligence, which is not yet as capable as the human mind) require algorithms to be written for any possible condition it may encounter, robots do have other positive advantages that humans do not. The most obvious is that robots are not capable of fear, and thus will not hesitate to enter a dangerous, life threatening area. Unlike humans, they are also highly expendable, and it is possible to put them into dangerous situations without worrying about their safety. Because they are relatively inexpensive, they can be employed in swarms, networked such that they may explore an area much faster than a group of several humans could. Finally, and very importantly, they can be equipped with communication devices that allow them to communicate with each other and more importantly with the human rescuers. This robot-to-human communication is the most important characteristic of a robotic network, as it allows the human rescuers advanced knowledge of the area into which they must go. Thus human rescuers will know before entering the building the layout of the building, including passable and impassable corridors, locations of seats of fire and of dangerous chemical spills, and the location of possible human survivors. Because a large swarm of robots may explore a building much faster than a group of humans could, the swarm could locate survivors much quicker than the humans, and after communicating those locations to the human agents, the humans may quickly move directly to those survivors to pull them out. Very soon, too, robots will be so advanced and powerful that they will be able to lift and carry - or drag - human survivors out of a danger area without the need for human rescuers.

In order to accomplish the task of search and rescue in an unknown environment, it is necessary for robots to be able to move through any manner of terrain, maintain awareness of their absolute positions within the area, and be capable of detecting walls, doors, obstacles, and human survivors. They must also be able to send all this information to human personnel operating the robot, which requires powerful radios with robust routing protocols.

1.1 Robot Requirements

Normally, USR robots would be required to handle movement through nearly all natural environments, such as deserts and forests and rocky areas, as well as damaged buildings in which walls have collapsed and left piles of rubble. In order to focus our research on the communication and mapping protocols of a USR network, we built a model building with a controlled environment, thus our robots were only required to be able to fit inside this environment and move comfortably within it. They were also required to be able to maintain their accurate position and orientation within the building, and detect the evenly spaced walls for the purpose of mapping the building. The model building is a 3 m × 3 m platform with a 16 × 16 grid of cells overlaid on it, shown in Figure 1-1. The cells are 18 cm × 18 cm. At the corner of each cell is a post with slots into which walls may be inserted, creating the ability to form any maze or building configuration. The walls themselves are 5 cm tall molded plastic pieces, painted white so that they easily reflect infrared light. Given these restrictions, the robots were required to be less than 18 cm on a side and capable of turning around inside a single cell in case the hallway is blocked.

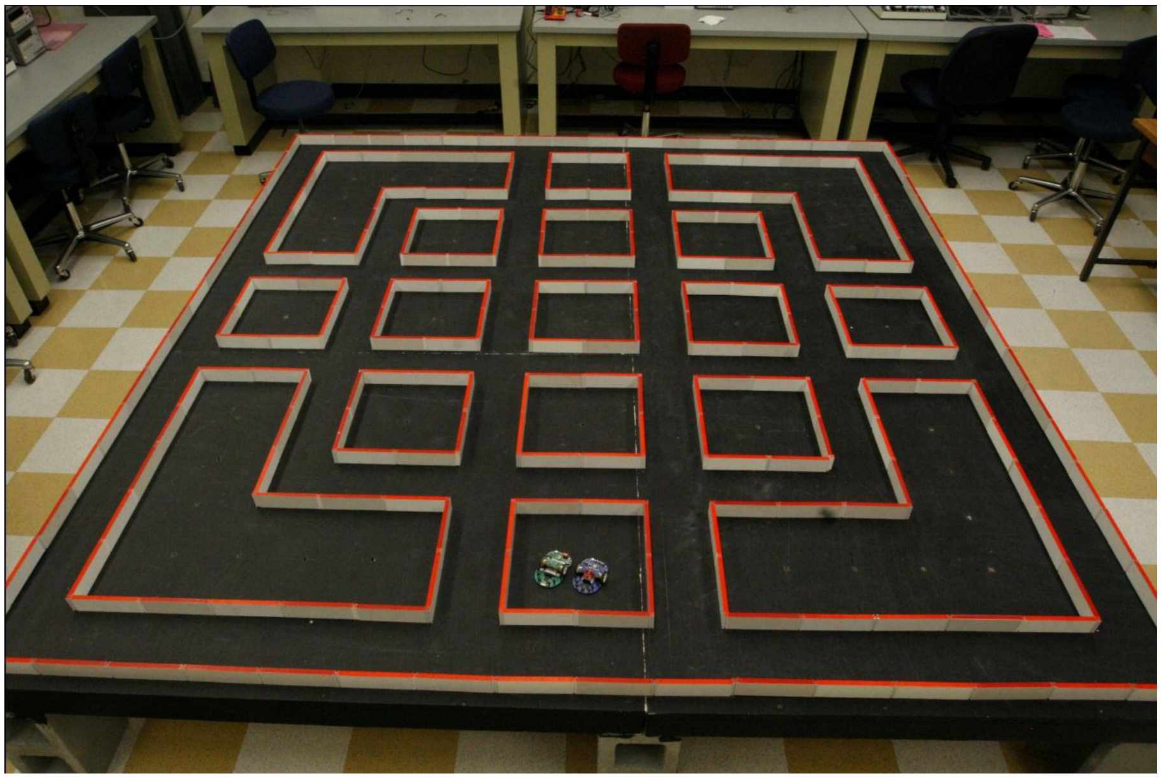


Figure 1-1: The Model Building Used for the Simulation of a USR Robotic Network

1.2 Radio Requirements

The selection of a radio for use in USR is an important issue, because moving data from the robot to the human operators is the key requirement of these robots, as is the efficiency of exploration, which determines how long it takes the robots to explore the area. Therefore the radio must be reliable, powerful enough to transmit the length of the building or area being searched, and have a data rate high enough to quickly move data from the robot to the operators. The power requirement can be mitigated in part by employing more than one robot with a multi-hopping radio protocol. With multiple robots, the distance of transmission can be significantly decreased. In most wireless sensor networks, a key issue is the power consumption of the radio, and many routing algorithms are primarily concerned with extending the lifespan of the battery to its absolute maximum. Radios used on USR networks, though, unlike stationary networks, are employed for a few hours at a time, rather than months, and in USR operations, primary concern is latency to the operator as well as efficiency of exploration. Considering this, and considering that the movements of the robots consume orders of magnitude more power than the radios, the radio selection focuses more on data rate and packet overhead than on power consumption.

1.3 Communication Network Requirements

Many protocols exist for communications in a static wireless sensor network, each with different priorities, from network lifespan (energy efficiency), latency, data throughput, security, and many others. However, because they are non-mobile, by their very nature they will not consider how efficiently a large area can be explored by only a few mobile agents. Thus, most static wireless sensor network protocols will not work in a mobile robot network without adjustment. For a robot network employed in search and rescue, the primary concern is that the data reaches the operator as quickly as possible, but a secondary and nearly as important objective is that the robots cover the entire terrain as quickly as possible. To accomplish both tasks, it is important that each robot also gets the data from all the other robots in time to prevent them from re-exploring areas already explored, and allow them to focus on moving into unexplored areas. While robot-robot communication is less time-sensitive than robot-human communication, simple flooding protocols, which send data to all other robots and humans within range of the transmission, will allow equal latency operations in both cases. Furthermore, these simple flooding protocols may actually decrease robot-human latency because they do not require knowledge of the receiving robots' near-future destinations.

1.4 Contributions of this Thesis

The major contributions of this thesis are

i) Hardware/Software co-design of a robot network for emulating a search and rescue robot network.

ii) Creation of an exploration protocol to explore a building.

iii) Creation of a communication protocol that allows the robots to share information with each other and with the human employing the network.

iv) Creation of a voting protocol that allows multiple robots to vote on the shared information to create more reliable map information.

v) Creation of a radio-limiting algorithm that allows the communications between robots to behave as if the network was employed in a full scale building, rather than a model building.

1.5 Overview of Thesis

The goal of this thesis is the creation of an urban search and rescue robot network emulator capable of testing different exploration and communication protocols in an actual robot network. In a real building, robots in opposite corners would not be able to communicate with one another due to path loss and large-scale fading effects. This network simulates this effect through software algorithms that limits the ability for robots to communicate with each other and with human users in the model. Secondary goals are the creation of an efficient exploration protocol with which the robots may explore the building, and a communication protocol with which the robots may share data about the building amongst themselves and with the human operator.

1.6 Organization of Thesis

Chapter 2 discusses some of the work that other researchers have done on the problem of robotic USR networks, mobile wireless sensor networks (MWSN) and security, search, and rescue robotics (SSRR or rescue robotics). This chapter also discusses a short description for the approach taken in this thesis, and some of the justifications for the scale reductions made for the emulator.

Chapter 3 discusses the hardware design of each of the robots, the software algorithms written for control, sensing, and communication, and finally the computer programs written to interface with the network.

Chapter 4 discusses the experiments completed to determine how the radios used for the robot networks behave in a real world environment, in this case the engineering building on the University of Alaska Fairbanks campus. With this data we justify the radio limiting software algorithm.

Chapter 5 discusses the results of the USR robot network with the simple exploration protocol and communication protocol.

Chapter 6 summarizes the main results of this thesis and discusses future work.

Chapter 2 Robotic Urban Search and Rescue

2.1 Description of USR Radio and Exploration Protocol Requirements

The field of networked robots or mobile wireless sensor networks can generally be divided into research subsections, the primary being communications, path planning, localization, and energy efficiency. Communications can be broken down further to include characteristics of the wireless channel, time synchronization, medium access and scheduling, and network routing, which also considers data throughput and latency. Communications also considers how the nodes of the network, both static and mobile, are deployed such that connections are maintained. Path planning is concerned with the movement of the nodes such that the area is sufficiently covered with sensors. Localization considers methods for determining the location of all the nodes or agents within the network, and energy efficiency looks at methods for conserving energy in the network to extend the lifetime of the network.

It is important to recognize that the four main subsections are in many ways inherently interconnected. Network performance (data throughput and latency) can be greatly affected by even small movements of the robots, where small-scale fading can drastically change the link quality of a connection. Connections can drop entirely if a robot moves around a corner or behind an obstacle (large-scale fading) or if they move too far away (path-loss). Network communication can affect the ability of the robots to share information about their environments and coordinate their movements, which could negatively affect the search time of the network by inefficiently re-covering terrain that has been previously explored. If the exact position of a robot is unknown, uncertain, or incorrect, it can cause erroneous movements that could then affect communications and cause poor energy efficiency. Also, depending on the method of localization, poor path planning can lead to insufficient data to correctly localize. Thus, while much of the research mentioned has focused on only one or two of the aforementioned topics, note that the research has an effect on more than just the focus of that paper.

2.2 Related Research

As mentioned above, for the purposes of this thesis the topic of MWSN is broken into four main characteristics: communications, path planning, localization, and energy efficiency. The last three will not be discussed in depth in this thesis; however, there are many good papers discussing these topics, which will be mentioned here briefly for the interested reader.

Several researches on energy efficiency in a MWSN have discovered that the average power consumed by each of the robots in an MWSN will decrease as more robots are added in the network [1], [2]. This is contrary to a static sensor network, in which the addition of static nodes increases the average power consumption of each node. Average power consumption is reduced in a mobile network because the robot's movements are the major source of energy consumption, and therefore, as more robots are added, each robot is required to move less, thus lowering the power consumption of each robot. Note that the total power consumption of both the static and the mobile sensor networks increases with an increase in the number of nodes.

Localization in MWSN is important mostly as a means of ensuring that the map that is being built from the collected data is accurate. If the position of the robots is incorrect, the map will be faulty and of no use to any human personnel. Accurate localization is also important to ensure correct movements for network connectivity. Some researchers use the standard hardware to dead reckon the position of their robots, such as encoders and compasses. In rare cases where the search environment is not in a GPS denied area (such as inside a building), then GPS gives near perfect position information. Some researchers, though, have been looking for ways to lower the cost of their robots by using signal strength, angle-of-arrival, or time-of-arrival information in the radio communications to triangulate the position of their robot inside the network [3], [4]. Yet another study uses a technique known as Simultaneous Localization and Mapping in which the robots use information from all their sensors (including radio maps derived from the radio's received signal strength indicator RSSI measurements) to both map and localize the robots within the area [5].

Another main area of research is that of path planning: determining where the robots should move to best accomplish whatever mission they are given. Most current path planning research is focused on fully decentralized path planning algorithms. In studies presented in [6] and [7], each robot is responsible for determining its own path through the area, based on its knowledge of the area and the position of the other robots it has obtained through the communication channel. This decentralized approach is advantageous in its redundancy, but it is not capable of making decisions concerning the network as a whole, so may not create the most efficient path plan. The other popular approach is a hybrid centralized/decentralized approach [2]. In this case, the network is divided into subnets, and in each subnet there is a single coordinator robot and multiple sensor robots. The sensor robots are required to stay within a set distance of the coordinator and send to it data about the area in which the subnet group is exploring. The coordinator robot would receive all the data and make decisions about where the group as a whole should move. In this study, there would be numerous groups comprising the entire MWSN, each making its own path planning decisions independent of what the others are doing.

Perhaps the most prominent area of research in MWSN is the communication between robots. The communication category can be broken down further into subcategories. Characterization of the wireless channel is very important, both for mobile and for static networks as it allows better mathematical approximations in simulations of the network [8]. Routing protocols determine the data throughput from source to destination, how high the latency will be, and in a mobile network, how quickly the connection will be reestablished after it is lost due to movement of the nodes [9]-[11]. Also important in the communication category, and the focus of this thesis, is the deployment of the agents or nodes within the network. Most research in this subcategory focuses on a distribution that maintains network connectivity at all times [12]-[18]. This distribution protocol will therefore distribute the nodes in such a way that connectivity is assured, and if small-scale fading, large-scale fading, or path loss cause the network to break, the distribution algorithm will quickly heal the network by moving the robot back into a connected position. In these cases though, connectivity comes at the expense of sensor coverage, which was never at 100% in any of the studies in which it was considered. This is in part due to the fact that most of the robots in an MWSN that must be always connected are forced to become relay nodes to relay data down the line, in which case their movements become tied to those of the robots that are actively exploring. This prevents expansion into corners of the search area that are out of range of the fully extended (maximum number of hops) of the network. If the network is always connected, though, data throughput will not be interrupted, which is important in a search and rescue network. However, because of the lack of full coverage, important information, such as the location of a survivor, can very well be overlooked.

2.3 Proposed Solution

For USR networks, sensor coverage is the most important requirement. Expeditious movement of the data from source to human user is very important, but not at the expense of coverage. Therefore, a new paradigm must be considered for urban search and rescue, one in which sensor coverage is prioritized. The coverage does not have to be full (100%) at every point in time; rather the algorithm needs to ensure that at least one robot visits and collects data on every location of the search area. A few researchers have touched on the idea of using the mobility of the robots as part of the network connection [19]–[21]. In these cases, the direct focus was on the throughput of the network when the network was connected-in-time [22].

This thesis explores the creation of a small model network of robots that may be employed in the future to investigate the different communication and distribution protocols mentioned above.

Chapter 3 Construction of Robots for Search and Rescue Setup

3.1 Introduction

To maximize the ability of search and rescue robots to efficiently and quickly explore an area, the design must be a hardware/software co-design [23]. According to [24], hardware should be designed to maximize the performance while software should be designed to give the device flexibility to perform a variety of tasks. To accomplish the task of searching our model building, we built a fleet of four small robots, called Robotic AuTonomous Scouts (RATS), in homage to the micromouse platform from which the robots were designed. Following is a description of the hardware and the software design of the RATS robots.

3.2 Hardware

RATS are small (approximately 7.5 cm x 10 cm) two wheeled “skid-steer” style robots, shown in Figure 3-1.

Each robot is equipped with dual microcontrollers, two small motors controlled

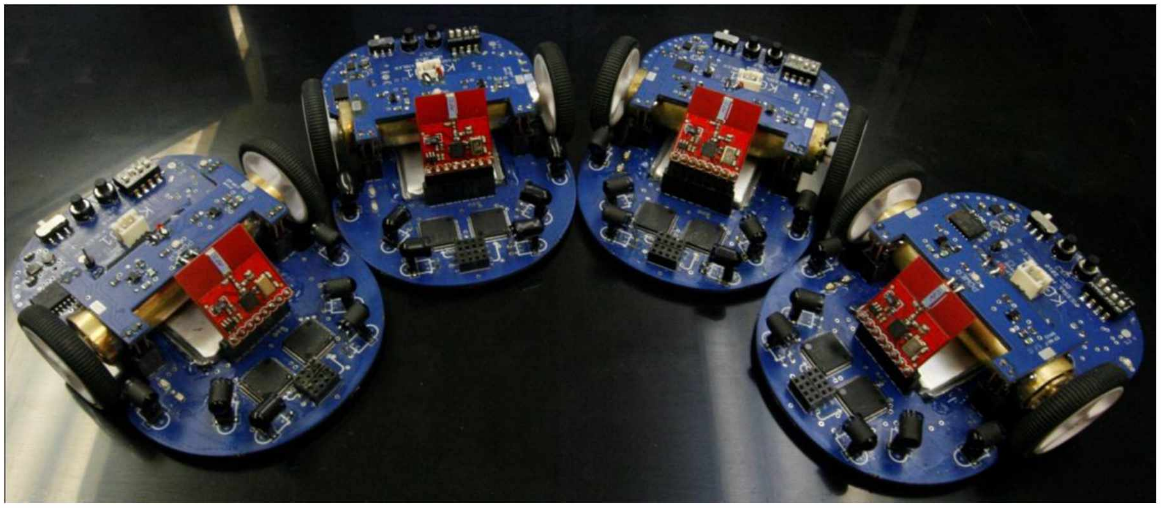


Figure 3-1: The RATS Network of Four Robots

through a dual H-bridge motor controller, a lithium-ion battery and both boost and buck regulators to give the necessary voltage levels. It has six small infrared LED emitters and matching infrared light-to-voltage sensors for detecting the walls of the building model. They also have small semi-homemade optical wheel encoders for determining how far they've moved. One final, but most important, addition to the RATS is the 2.4 GHz transceiver module built into the robots to facilitate communication amongst the network of RATS. All these features will be discussed in the following subsections.

3.2.1 Processors

In order to handle the extra workload of communicating with a network of robots and moving throughout the maze collecting data, a second processor was added, as shown in Figure 3-2.

The workload between the two processors was divided into a logical higher-function/lower-function layout. The primary processor handles all radio communications as well as the mapping algorithms for the robots. This processor also

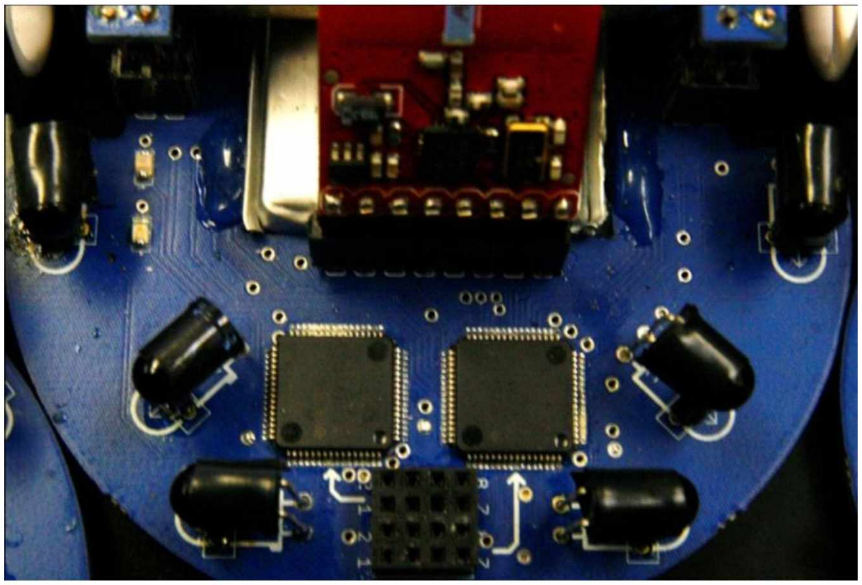


Figure 3-2: MSP430 Parallel Processors on the RATS Robots

stores all the data about the building. The secondary processor handles the control of all the hardware elements of the robot, from controlling the motors to reading the data from all the sensors. The two processors communicate via a Universal Asynchronous Receive Transmit (UART) port. The data transmitted between the two processors is very minimal; the primary sends the secondary a single 1-byte command telling it in which direction the robot should move. The secondary processor sends back a 1-byte movement-complete confirmation that includes data about the new location within the building. The RATS currently only collect data concerning the existence of any walls within the area, though eventually other sensors would need to be added, and the data from those sensors would need to be included in the confirmation packet.

3.2.2 Sensors

The RATS currently have need of only two sensor types: one for dead-reckoning its position within the building and one for determining the existence of walls at its current position. In order to determine the existence or non-existence of walls in the building, six infrared LEDs and matching infrared light-to-voltage sensors are used, shown in Figure 3-3.

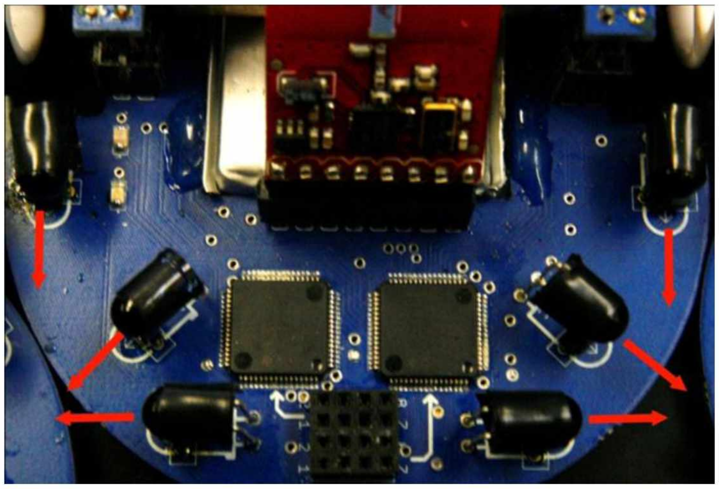


Figure 3-3: Infrared Sensors for Detecting Walls on the RATS robots

These sensors are arrayed in six positions: two pointed forward for detecting walls ahead of the robot, one pointed to each side for detecting walls to each side of the robot, and one pointed diagonally on each side for lateral and longitudinal recalibration. This recalibration is essential for helping eliminate errors in the robot's position, and therefore also in the robot's ability to correctly detect the walls. These infrared light and sensor pairs have some characteristics that must be considered in the placement of the sensors. Figure 3-4 and Figure 3-5 show the sensor output versus the distance and the angle, respectively.

Note the large flat "shelf" that constitutes the saturation region within approximately one inch of the wall and within approximately 18 degrees of perpendicular. It is because of this saturation region that the sensors are placed on the robot as far from the edge of the robot as possible, which reduces danger of entering saturation. Note also that at approximately 45 degrees from perpendicular, the slope of

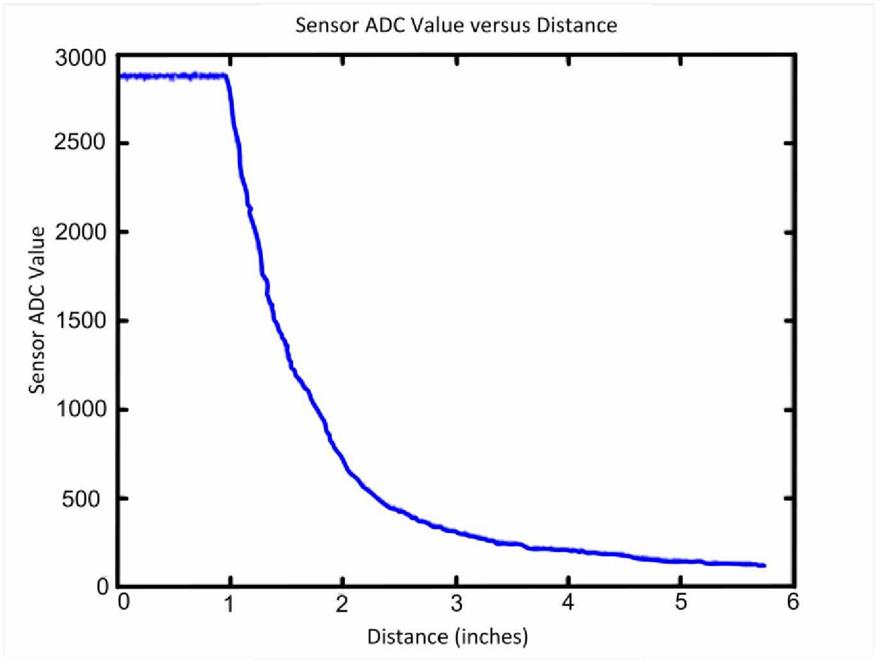


Figure 3-4: Sensor Output vs. Distance from the Wall

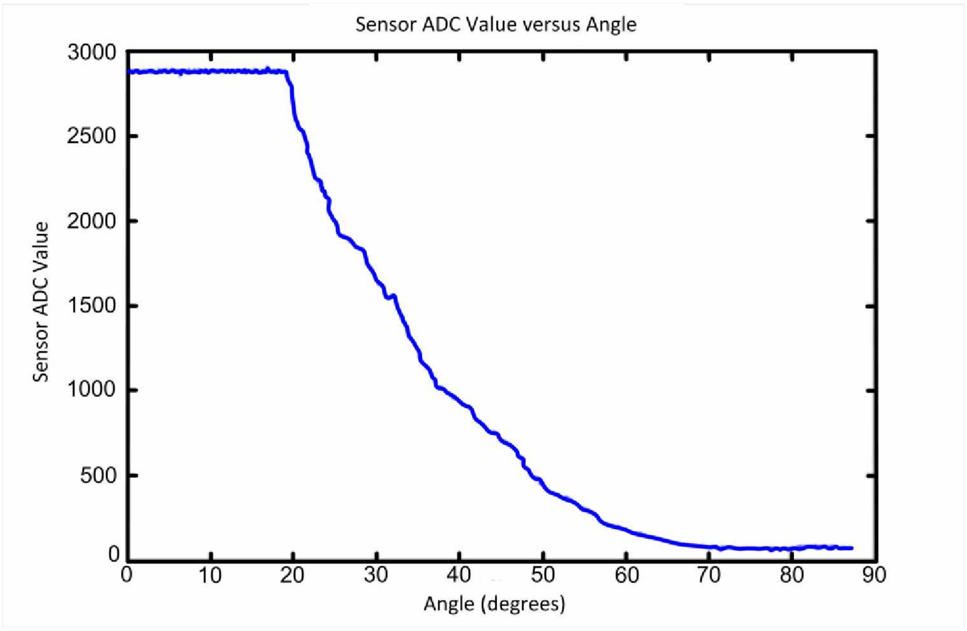


Figure 3-5: Sensor Output vs. Angle to the Wall

the curve very nearly a 1:1 ratio. This fact allows us to observe that the diagonally placed sensors' output will have equal changes in magnitude and opposite changes in direction with similar changes in the robots angle relative to the walls. Thus, these sensors are ideally placed for keeping the robot centered in the building's hallways. This concept will be explained in more detail in the section on controls.

The second type of sensor is an optical encoder for determining the degree and speed of rotation of the motors and wheels. There are two types of mainstream optical encoders used in today's robots: the first uses a solid circular disk attached to the wheel or axle with black and white pie shaped stripes on it, the white stripe reflecting the IR light back to the photo-detector and the black stripe absorbing it. The second type uses a circular device with spokes as on a bike wheel, again attached to the axle or motor shaft, wherein the spoke blocks light to the photo-detector and the space between spokes allows the light to pass through. The first type generally requires a large wheel diameter to accommodate enough stripes on the pie to allow for good encoder resolution. The second type is bulkier and requires a larger amount of space than we were willing to give it. Our solution was to use the reflective-type sensors (a side-by-side emitter/detector pair), and paint the white and black stripes on the rotor of the motor itself, shown in Figure 3-6.

This of course required a small hole to be cut into the can of the motor through which the light could reach the rotor and reflect back, and though there is a loss of magnetic flux through the new hole, it proved not to be noticeable. This allowed us to achieve twelve encoder counts per motor revolution, which multiplies to 360 at the wheel with a 30:1 gearing system. This number of encoder counts proved to be a good resolution, as we were able to have millimeter control of the robot's position in the maze, and were able to keep the wheel speeds close enough that the robot moved nearly straight at all times.

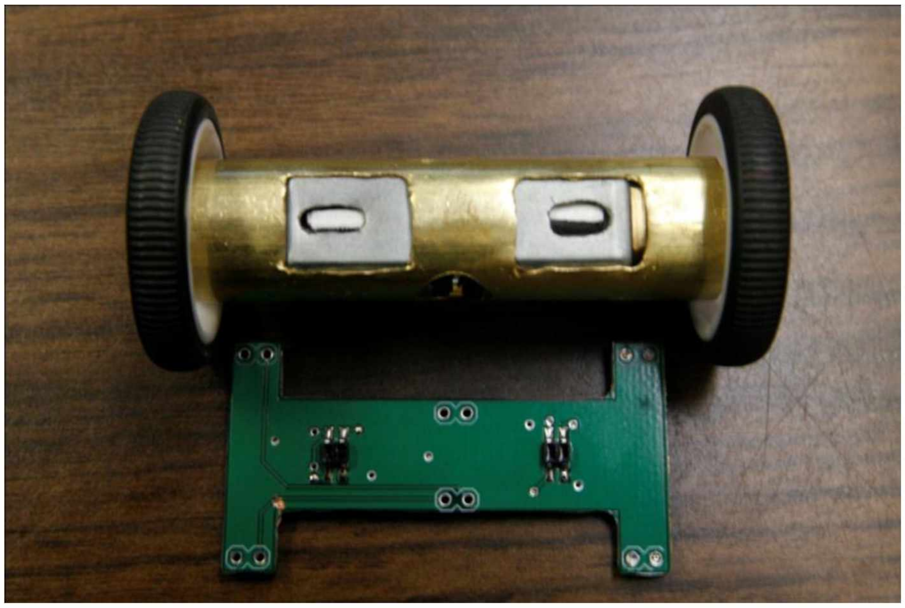


Figure 3-6: Robot's Motor with the Holes in the Motor Can and the Optical Encoders

3.2.3 Power Systems

In order to keep the robot compact and keep the center of mass centered in the robot and as close to the ground as possible, the RATS use a single 850 mAh Lithium Ion battery glued into a square cutout in the board itself. The battery is nominally at 3.7 volts, but the electronics require 3.3 volts and the motors are rated for 6 volts. Therefore, the RATS have a Texas Instruments TPS63001 [25] 3.3-volt buck regulator to power the electronics and a TPS63002 [25] 5-volt boost regulator to power the motors. Table 3-1 shows the current requirements of each of the individual parts of the RATS robots.

Table 3-1: Robot Power Requirements

Part	Qty	Average Current Draw Per Device (mA)	Duty Cycle (%)	Subtotal Average Current Draw (mA)
MSP430F2618 Microcontroller	1	10.0	16.4	1.64
MSP430F2618 Microcontroller	1	10.0	98.4	9.84
Nordic nRF24L01+ Transceiver	1	13.0	100	13.0
TPS630001 Voltage Regulator	2	0.05	100	0.10
L293DD H-Bridge	1	60	100	60.0
Gearmotor 30:1	2	100	98	196.0
TSL261R Infrared Sensor	6	1.1	100	6.60
Vishay Infrared LED	6	50	18	50.0
QRE1113 Photo reflector	2	20	100	40.0
MAX9030 Comparator	2	.035	100	0.075
Debugging LEDs	5	20	50	50.0
Power LED	1	20	100	20.0
Total				447.25

There are several things to note about the above table. First, the energy consumed by parts required for accurate robot movement (motors, motor controller, encoders, etc.) account for nearly 68% of the total energy draw. On the other hand, the parts required for radio communication account for only about 3% of the total energy draw. Noting this fact, very little attention was paid to energy efficiency in the radio algorithm. For this reason, the radio was kept in receive mode at all times (except for its brief window in which it is transmitting its own data) to ensure that all radio transmissions from other robots or the data sink are received. Because the communication protocol employs time division multiple access (TDMA), it is possible to increase efficiency by turning off the radios until the exact time that they must transmit or receive a transmission from others. Also, other radio protocols could be employed that are more energy efficient. Energy efficiency is considered outside the scope of this thesis, though, so little effort was devoted to this issue.

It should also be noted that the specified current for the motors (100 mA) is derived from the data sheet and is the value for full-speed loaded operation. They can draw 500 mA or more during stall and acceleration. The robots stop in each cell to map the walls in that cell and make a decision about how to move. Therefore, the robots must come to a controlled stop and then accelerate out of the cell again, in which they will draw the higher current load. It is beyond the scope of this thesis to determine the average current draw of the motors as they explore the building. However, it is important to note that the above table included some simplifications.

While it is possible to compute the expected battery life given the above information, the result will not be highly accurate. As noted in [26], the battery's capacity is greatly affected by the magnitude and consistency of the current draw. Furthermore, the above table is based on values given in datasheets and is not based in actual real world operation. Therefore, equation 3-1 is only a rough estimate at best,

$$\text{Battery Life} = \frac{950\ \text{mAh}}{447.25\ \text{mA}} = 1.95\ \text{hours}. \tag{3-1}$$

While equation 3-1 implies that the battery life is almost two hours, in reality the robots are able to operate for twenty to thirty minutes, less than a quarter of the time listed in equation 3-1. Again, this is due to the fact that the battery capacity listed on the battery is based on a specific current draw.

3.2.4 Motor Drivers

The robot must control a 6 volt motor with a 3.3 volt microcontroller, so an amplifier must be included in the system. The RATS have an ST Microelectronics L293DD Dual H-Bridge Amplifier [27] for controlling speed and direction of both motors. A diagram of how the H-Bridge works is shown in Figure 3-7.

As shown in Figure 3-7, the direction of the motor is controlled by choosing which pair of inputs to toggle on the H-bridge and which to set as open. The speed is controlled by feeding a Pulse Width Modulated (PWM) signal into the amplifier. The speed of the motors can be defined by the duty cycle of the PWM cycle, or

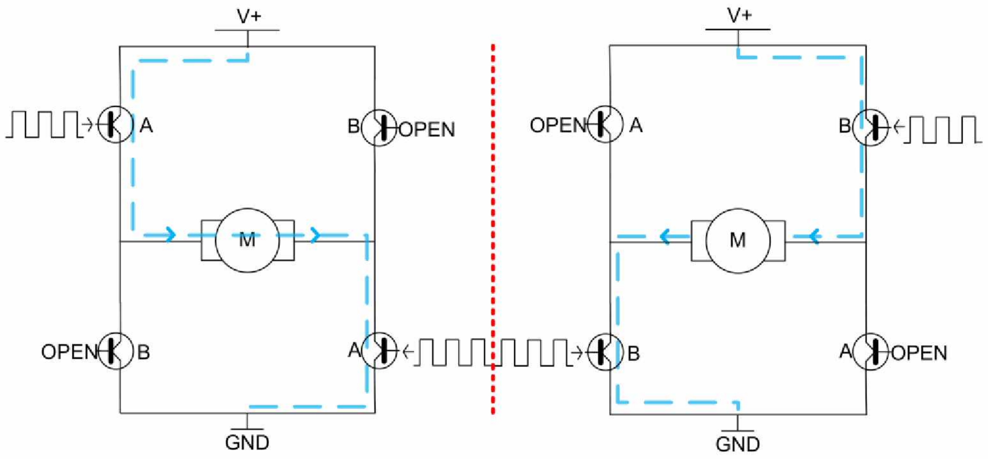


Figure 3-7: Controlling the Speed and Direction of a Motor with an H-Bridge

$$v = \frac{1}{T}\int_0^T c(t)dt = \frac{1}{T}\left(\int_0^{t_1} c(t)dt + \int_{t_1}^T c(t)dt\right). \tag{3-2}$$

In equation 3-2, v is the average voltage, T is the total period of the signal, and $c(t)$ is the PWM control signal. If we assume an ideal square wave, then the period [0, t_1] will be 6 volts (at the motor terminal) and the period [t_1, T] will be 0 volts at the motor terminal. Thus the equation becomes

$$v = \frac{1}{T}\left(\int_0^{t_1} (6v)\, dt + 0\right) = (6v)\frac{t_1}{T}. \tag{3-3}$$

From equation 3-3 we can see that the average voltage on the motor over time is directly proportional to the duty cycle ratio t_1:T. Figure 3-8 demonstrates this concept visually.

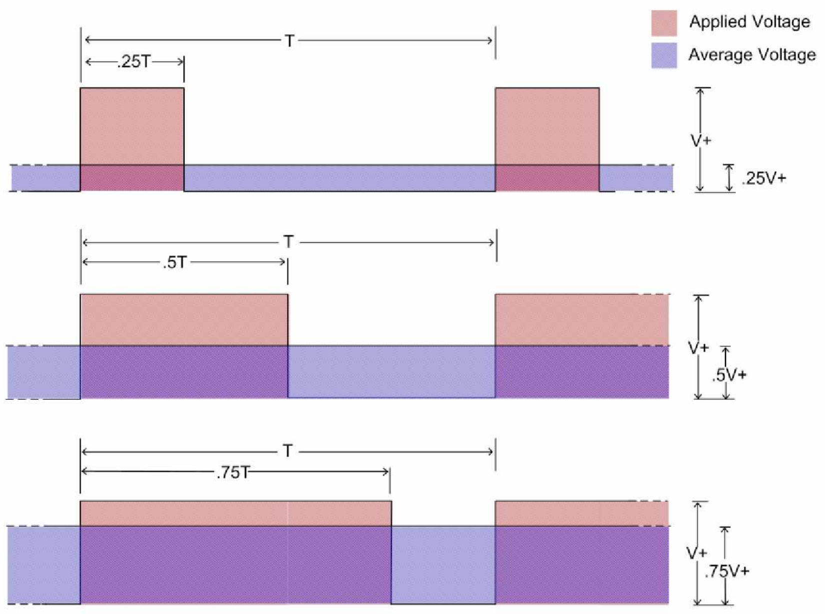


Figure 3-8: Relationship between Duty Cycle and Average Voltage

The period of the control signal is an important factor to consider in the control of the robot, as well, because each cycle of the waveform represents a single given speed to the motor. Thus, the motor's speed can only be updated in subsequent cycles of the signal. Therefore, the selection of the period of the signal is also the selection of how often the motor's speed can be changed. A signal that updates too infrequently can result in choppy motion. However, because the microcontroller's onboard timer is used, as the period decreases so do the number of steps in the duty cycle, thus resulting in larger voltage changes with each step.

3.2.5 Radio

The RATS robots have a Nordic nRF24L01+ 2.4 GHz Transceiver [28] for communicating between themselves and the data sink, pictured in Figure 3-9.

The Nordics are connected to the microcontroller via a Serial-Peripheral Interface (SPI) port, and can accept up to 32 bytes per transmission packet. The radios handle all the overhead, including the preamble, addressing, and the CRC. They also have the ability to auto-acknowledge incoming packets if desired. For the RATS network, as will be discussed later, the auto-acknowledge feature was disabled so that

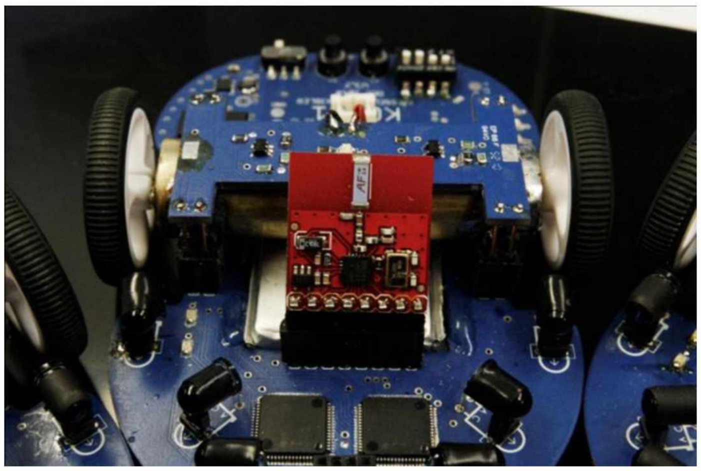


Figure 3-9: Nordic nRF24L01+ 2.4 GHz Transceiver for Wireless Communication

multiple receiving robots do not all try to acknowledge at the same time, thus creating a collision of the acknowledgement packets and rendering them useless.

3.2.6 Data Sink

On the receiving end of the communication channel is the data sink. The data sink is a custom built device with dual Nordic nRF24L01+ radios attached to a single MSP430F2618 microcontroller, shown in Figure 3-10.

For larger networks this double radio system will help in reducing congestion by separating the single network into two distinct networks, each on its own channel. The area two networks cover can overlap, or they could be tasked with exploring two distinct partitions. The dual network might also divide nodes into two distinct mission groups, such as robot scouts and human rescue personnel. For the present, only one of the two radios is being employed on the data sink, as there is only one small network in the model building.

3.2.7 Communication Snoop

Because the RATS network is an emulation of a larger-scale network in which packets can be lost due to walls and obstacles in the transmission path, the Data Sink

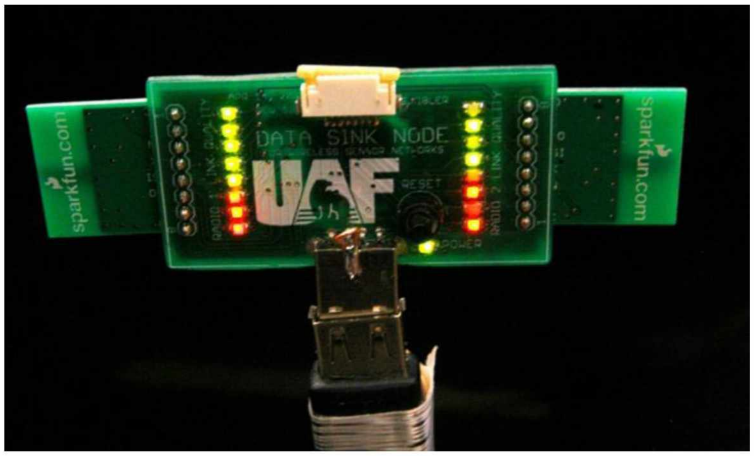


Figure 3-10: Data Sink for Relaying Data from the Wireless Medium to the Computer

mentioned above had limitations built into the software that decides which received packets from the robots may be considered in range. This will be discussed in Chapter 4 in the radio limiting section. These limitations made it impossible to visualize the actual condition of the network for testing and troubleshooting purposes. Thus a snoop device was built that was simply able to snoop on all transmissions and display all the real-time data from the network, including locations of the RATS and the information currently collected about the building. The snoop device is shown in Figure 3-11.

3.3 RATS Software Algorithms

While hardware gives the robots functionality, the software algorithms give the robots flexibility. For example, it is possible to write different exploration algorithms, so that they can be tested and the best of them selected. Similarly, different radio communication protocols can be written and compared so that the best might be found. In this section, the different software algorithms for the RATS will be discussed.

3.3.1 Control

The robot controls include reading sensors and using that data as feedback to

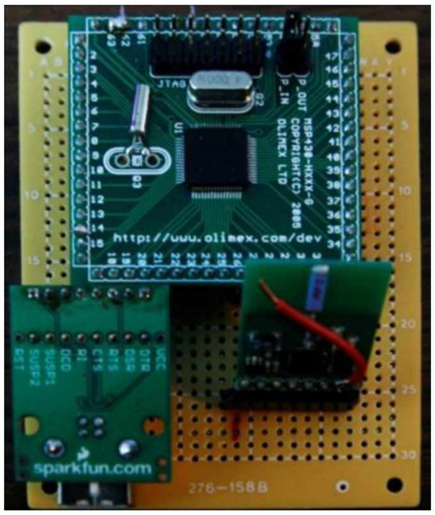


Figure 3-11: Snoop Device to Visualize Current Conditions in the Network

ensure the most accurate motor control as possible. As mentioned previously, the sensors used to assist in motor control are the encoders placed on the motors and the 45 degree infrared sensors.

The RATS use differential steering, allowing the robot to move straight or turn based on the difference in the two wheels' speeds. For the RATS to achieve consistent movements through the building's hallways, two criteria must be met. The first criterion is that the motors turn as close to the same speed as possible. When driving straight this is an obvious necessity, as it means the robot will move as straight as possible. It is also necessary in the robot's turns, though, as the robot uses pivot turns where one wheel moves forward while the other moves the exact distance backward. If the motors' speeds differ too greatly, then the turn will not be perfectly orthogonal. The second criterion is that the robot must return to a centerline course if it has been displaced. Displacement from centerline occurs frequently due to inconsistencies in the tires of the robots, warping in the floor of the model building, non-ideal controller characteristics, as well as impact with obstacles in the way. To achieve the necessary control, three feedback mechanisms are employed in the control algorithm. The first mechanism determines how closely each motor is turning to its desired speed, the second mechanism determines how closely the two motors are turning relative to each other, and the last mechanism determines the lateral error of the robot in the building corridor (whether the robot is getting too close to one of the walls).

The first feedback mechanism is the single motor feedback, which informs the motor how its actual speed relates to its given desired speed, shown in Figure 3-12.

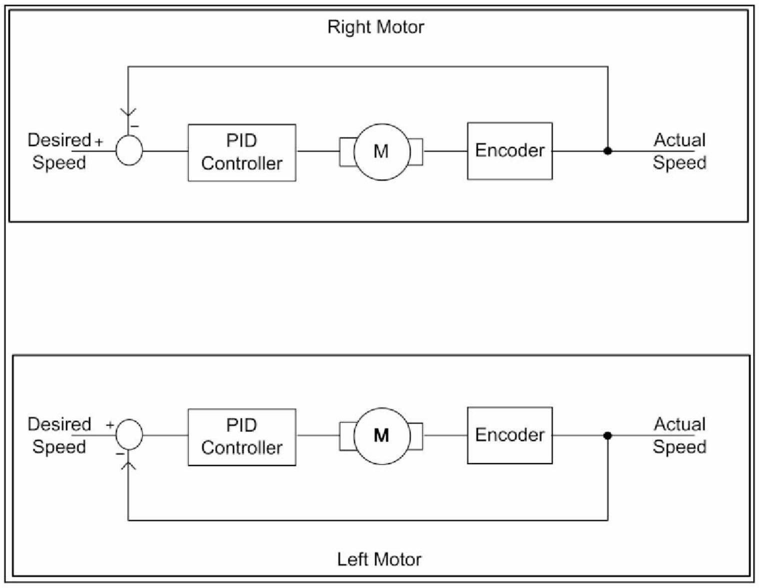


Figure 3-12: Single Motor Speed Feedback Mechanism for Motor Control

The error between the desired speed and the actual speed is obtained and fed into a Proportional, Integral, and Derivative (PID) controller, which uses the error to update the PWM signal fed to the motor. Thus if the motor is too fast, the feedback mechanism will create a negative error, and then the error is added to the current PWM signal, creating a smaller duty cycle and slowing the motor down.

The second feedback mechanism is the error between the two motors' speeds. This is a much more important feedback than the single motor's speed error, because it is the difference in the wheel speeds that cause the robot to run off course. This error is also fed into the PID controller and updates the motor speed control signal just as the single motor feedback does. However, the gain on this feedback is larger to give more weight to it than the single motor feedback. This choice is obvious because it is more important that the motors are turning at the same speed than that the robot is moving at a desired speed. Figure 3-13 shows the motor controller with the speed difference error included.

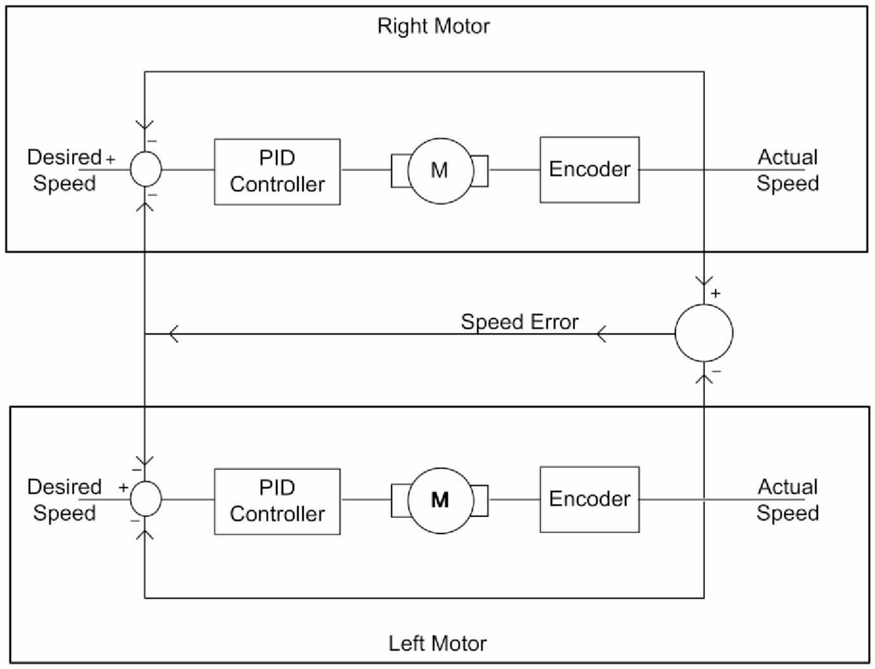


Figure 3-13: Motor Feedback with the Double Motor Error Feedback

The third and final feedback is the use of the infrared sensors placed at 45 degrees to one another. When the robot is centered in the hallway, and aligned correctly (pointing straight down the hallway), then the 45 degree IR sensors will output approximately the same value. When the robot becomes displaced or misaligned, one sensor will go high and the opposite sensor will go low, thus creating a difference or error in the signal, shown in Figure 3-14.

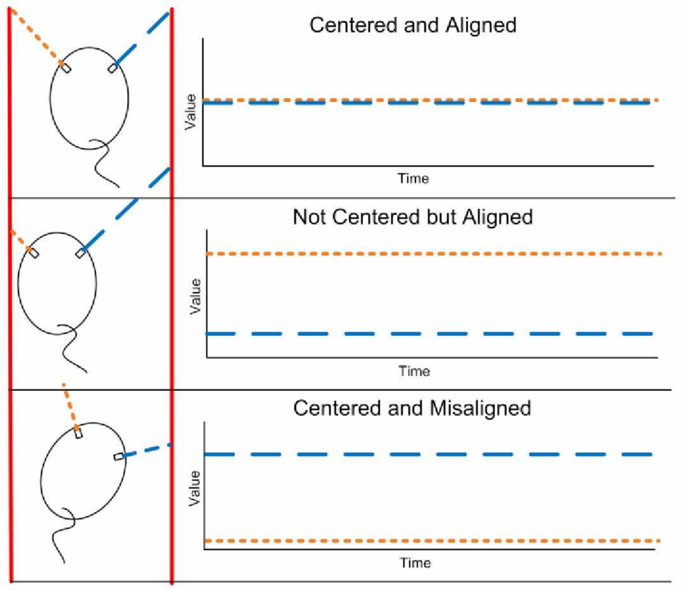


Figure 3-14: Diagonal Sensor Outputs in Three Unique Conditions

When this signal is also fed into the PID controller, the motors can be adjusted based on this error, causing the robot to realign itself to the centerline. Figure 3-15 shows the motor controller with the IR sensor feedback included.

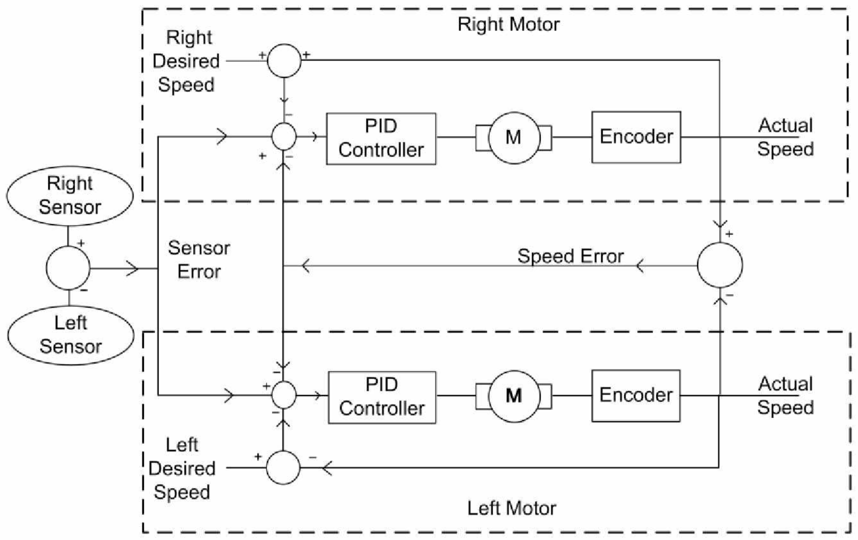


Figure 3-15: Motor Controller with IR Sensor Feedback Included

3.3.2 Mapping

The RATS must be able to explore the building quickly and efficiently, which requires the robots to keep updated information about their location and make good decisions about how to move through the building. These movement decisions are based on the current location of the robot, the desired goal of the robot, and the robot’s knowledge about the building at the time the decision is made. The mapping algorithm does this in three steps. It first divides the building floor plan into a grid with X and Y coordinates that stores all the wall information about the building. The algorithm then assigns the robot a goal cell within that grid, and makes note of its current position within the grid. The goal cell assignments are such that the robots are spread as evenly as possible within the building as well as ensuring that all areas are mapped as quickly as possible. The third step is to use the popular flood fill algorithm to make the movement decisions for the robot [29]. The flood fill algorithm works by assigning each grid cell a value that represents the number of cells necessary to move from that cell to the goal cell, which is set at a value of zero. After mapping the walls at its current location, and receiving building data from the other robots through the radio link, each robot will recalculate all the flood fill values based on the new wall information. After the flood fill values have been updated, the robot’s movement decision is made by moving to the lowest valued neighboring cell. Each time the robot moves, the process begins again: it maps the walls, recalculates all the flood fill values, and makes another movement.

When the robot has arrived at its goal square, it calculates a new goal square and the flood fill and movement process begins again. There are several methods for calculating new goal squares. The first is to randomly select goal cells. The second method is to analyze the building to determine where the robots have not been and assign cells from within that area. The third option is to find the closest unmapped cell and mark that as the goal cell. In each case, the cell is reviewed to ensure that the robots are not unnecessarily revisiting cells. Results from each of these mapping methods will be compared in Chapter 5.

3.3.3 Radio Communication

The radio communication must allow the data concerning each cell to move quickly through the network. It is important that the robots get all the data quickly so they can avoid revisiting cells unnecessarily. One efficient method of ensuring data gets to all the robots as quickly as possible is to have the robots transmit all their building data to all their in-range neighbors, as shown in Figure 3-16.

Their in-range neighbors will extract the data about the building and place the data in their memory alongside the data they have previously collected by their own sensors and from previously received transmissions. The neighbors, or receiving nodes, will then transmit this collection of building data, in their appointed time slot, to all their neighbors, shown in Figure 3-17.

Thus, this protocol is best considered a multi-hopping naïve flooding protocol, because it transmits to all its neighbors without bias, and the data a robot collects about

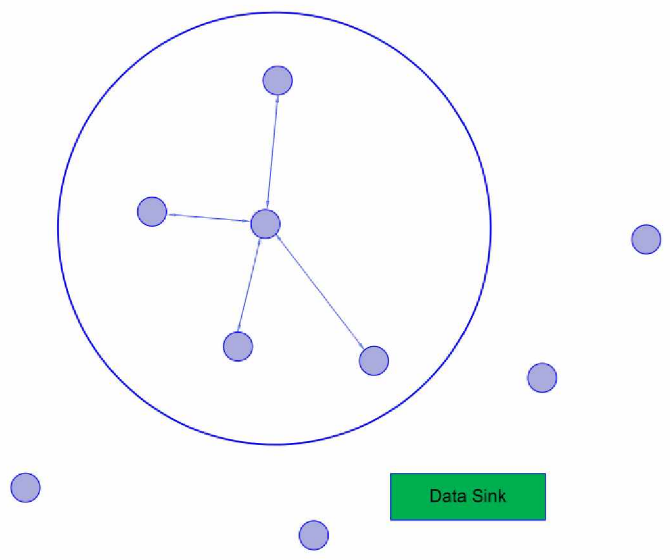


Figure 3-16: Flooding Protocol Example where all Nodes in Range Accept Data

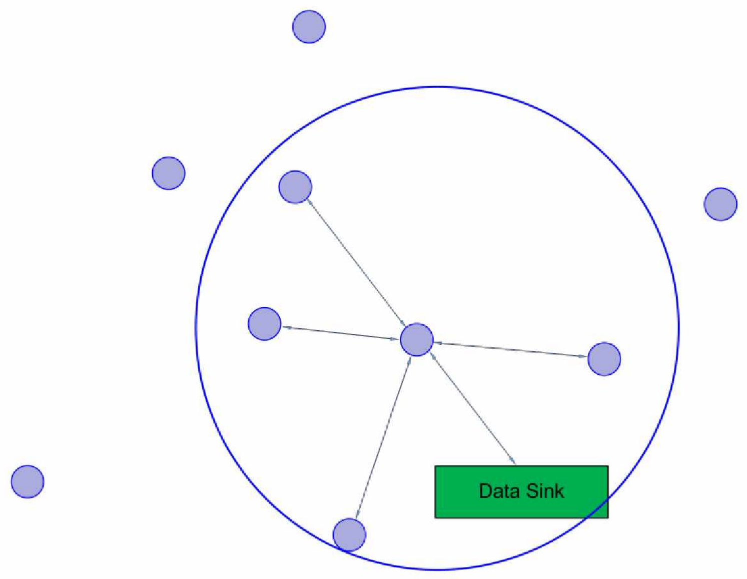


Figure 3-17: Second Step in Flooding Protocol

the building can hop to a robot (n-1) hops away (where n is the number of robots in the network), by being collected and retransmitted by the neighboring robots. Because this protocol distributes the data faster than direct-routing protocols, it allows for more efficient exploration of the building because the robots know which areas have been explored and which have not. By knowing this data the robots can move only toward areas that no robot has yet seen.

Another advantage of a flooding protocol is that routing tables are unnecessary. In an extremely dynamic network, where the nodes are constantly moving, maintaining routing tables (in a proactive network), or creating them (in a reactive network), requires extra transmissions containing “hello there” messages to establish routes to all other nodes before the data can be sent. This overhead is time consuming and could result in the robots making inefficient movement decisions if data has not arrived before the movement decision needs to be made. Also, using routing tables in a dynamic environment could cause delays in the data reaching the human users when a robot that has been included in a route suddenly moves out of range and the routing table becomes invalid. As mentioned in [8], even small movements of the robots can result in large drops in signal strength. This break in the link requires reestablishing a new route, which takes time. By flooding the network, it is possible that a robot that is not in range of the human users at the time of an important discovery (and would thus be ignored by a protocol requiring routing tables) could after only one or two small moves be the first robot to establish a connection with the data sink. For this reason, flooding protocols have an advantage in USR networks in which reducing the latency of the data movement is of the highest priority.

In order to prevent collision with other robots' transmissions, the radio network is set up with a time division multiple access (TDMA) protocol. Each robot has a specific time slot in which it must transmit. In its assigned slot, the robot must send all its collected information about the entire building. It sends the entire building because it may not know which robots are in range of the transmission (robots can move into and out of range very quickly), or what data the other robots may have. Therefore, to avoid the time consuming overhead of back and forth transmissions to request specific data from one another, each robot simply sends data concerning the entire building. For each cell in the building, the robot must send three bytes, which contain all the information necessary for the robots to vote on the condition of that cell. This will be discussed further in the section on voting below. In addition to the building data, each robot must send information concerning its own location and heading so that the other robots may decide whether it is in range (discussed in chapter 4), as well as for future work when robots will decide on cells to explore based on the location of known neighbors. Each packet the robot sends contains information on 8 of the 256 cells in the model building, and because the payload size for the Nordic radios is limited to 32 bytes, in order to send data about the entire building, each robot requires 32 of these 32 byte packets. The information in each packet is shown in Table 3-2.

Table 3-2: Radio Payload Layout

Byte #	Information Contained
1	Packet Type
2	Robot ID Number
3	Current Cell (X + 16*Y)
4	Current Heading
5	Row Index
6 – 29	Row Data
30,31	Free

Information concerning bytes 6 through 29 ("Row Data") will be discussed in the section on voting below. The Nordic radios take the 32 bytes given to them and insert them into a packet with a preamble, an address, a control field, and a CRC field as shown in Table 3-3.

Table 3-3: Nordic Radio Packet layout

Field Name	Number of Bytes
Preamble	1
Address	5
Packet Control Field	1.125 (9 bits)
Payload	32
CRC	2

Thus, each of these individual packets requires 329 bits to transmit. At 250 kbits/s (which is the slowest data rate at which the Nordic radios transmit), the time it takes to transmit each packet is

$$T_{packet} = \frac{329 \text{ bits}}{250 \text{ kbits/s}} = 1.315 \text{ ms}. \tag{3-4}$$

Each robot transmits 32 of these 1.315 ms packets in each time slot. Therefore, each time slot needs to be at least 42 ms long. In order to have tolerance of clock drift, each slot was made longer, approximately 250 ms long. Note that the robots must move between cells before collecting new data, a process that requires approximately 2 seconds per robot per cell. In a network of four robots and a data sink (which is also forwarding data back out to the robots), a time slot of 250 ms gives each robot an opportunity of transmitting approximately twice per movement. Transmitting twice per movement is redundant, so the robot will only transmit in the first slot after entering a new cell and collecting new data. The second slot will be left empty. However, it is advantageous to keep the second slot, rather than lengthening the slots to 500 ms, because the robots' movements are not in sync, and by keeping the slot width short, it allows those robots who finish their movement later an opportunity to transmit later, and every robot will still receive their neighbors' data within the span of one movement, as shown in Figure 3-18.

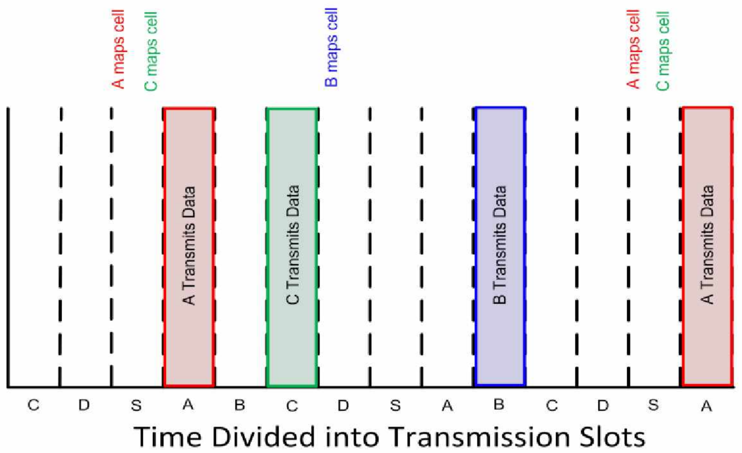


Figure 3-18: Transmission Slots for Multi-hop Flooding Protocol

3.3.4 Voting Algorithms.

Disaster areas are capable of shifting while a search is in progress. Unstable walls might collapse, blocking off a hallway that was previously passable, and doors may open that were previously closed. Furthermore, one or more of the robots' sensors may return incorrect data about the building's walls. This could happen for a number of reasons, including a faulty sensor, or operation of the sensor in a climate for which it is not rated. A good example of this is the use of laser distance sensors in a smoky or dusty environment, which significantly reduces their accuracy and range. A significant advantage of having multiple robots exploring the area is that they may vote on the condition of the building. This allows a higher level of confidence in the data that the human personnel are receiving and reviewing. To be a valid vote, at least three robots must vote on the cell. Each of the 4 walls (North, East, South, and West) is voted on independently of the other walls in the cell. The vote occurs with the following bitwise Boolean logic equation

$$walls = (X_1 \bullet X_2) + (X_1 \bullet X_3) + (X_2 \bullet X_3). \tag{3-5}$$

In the above equation, *walls* represents the final voted value of the cell's walls, while X_1, X_2, and X_3 represent the individual robots' vote on the walls, the value that they collected with their sensors. In this case, so long as any two of the robots agree about the state of a particular wall, then that wall state will be accepted as valid. This can be seen in Figure 3-19, where the bottom cell shows the actual state of the wall, and the cells above show the cell as each robot mapped it.

By having three robots vote on each wall, we should end up with a valid state on the walls. An exception would occur when a wall collapses after the first two robots visit a particular section of hallway. When the third robot moves through, it registers the new, actual state of the hallway, but because it is the only one of the three that has

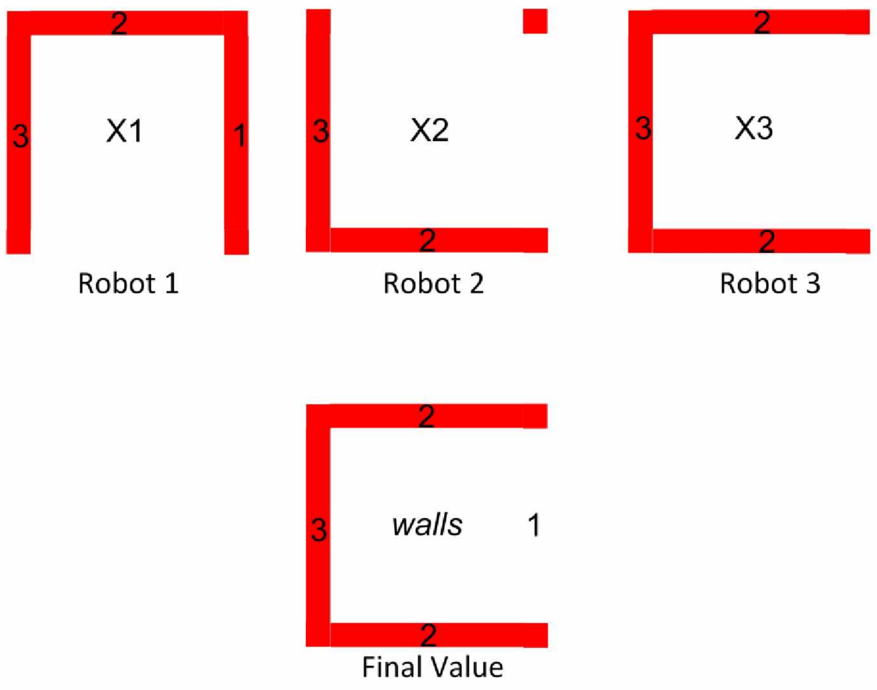


Figure 3-19: Voting Algorithm Catches Errors in the Individual Robot's Maps

seen the absence of the wall, the vote still registers a wall in that position. To track areas where this has occurred, the location of errors can be found using the bitwise Boolean logic equation

$$error = \overline{\left((X_1 \bullet X_2 \bullet X_3) + \overline{(X_1 + X_2 + X_3)}\right)}. \tag{3-6}$$

In this case, *error* will contain a 1 for any bit in which the three robots do not agree, and a 0 where the three robots do agree, as shown in Figure 3-20.

Knowledge of where the errors exist is important because it allows human personnel to focus their efforts on the troubled locations, and it also allows them to track whether a robot's sensor has broken. If one robot is consistently incorrect, then it is likely its sensor is damaged and it can be removed from the network and repaired. If

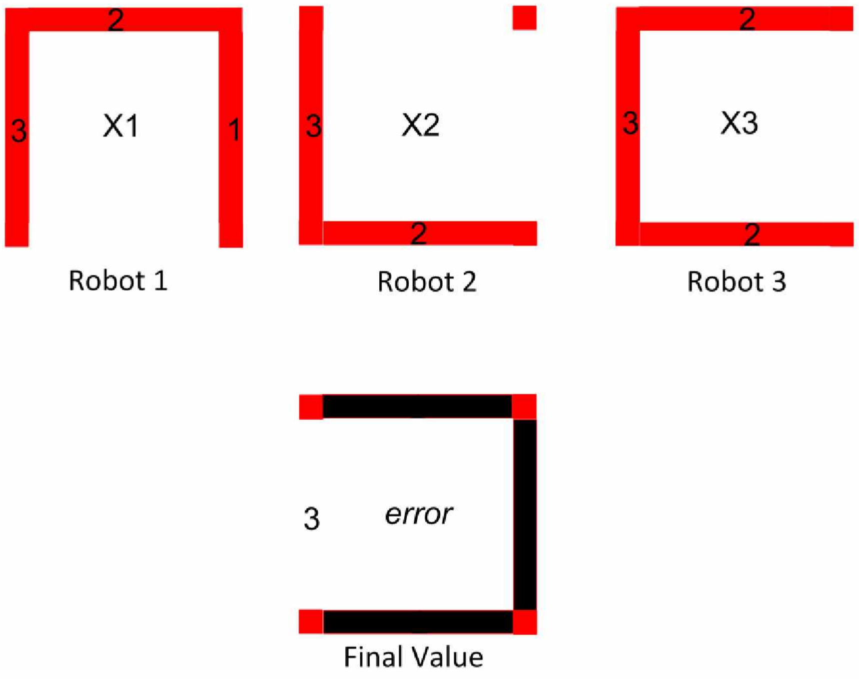


Figure 3-20: Image Demonstrating Error Equation Results

desired another robot can be sent to the location of the errors to verify the condition of the wall.

In order for voting to occur, each robot requires knowledge of the data gathered by other robots. Thus, it is required that the information contained in the radio packet contain the data needed for the vote. Because it is necessary to have three unique robots voting on the data, it is also important to send the robot's ID along with its wall data to ensure that the same robot doesn't get two votes per wall. In our small network of four robots, we send three bytes of data per cell, shown in Table 3-4.

Table 3-4: Voting Data for One Cell Contained Inside the Radio Packet

Byte 1	Robot ID 1	Wall Data 1
Byte 2	Robot ID 2	Wall Data 2
Byte 3	Robot ID 3	Wall Data 3

Each robot, as well as the data sink and the snoop, upon receiving a packet of information, removes the information about the walls shown above, stores it in memory, and then uses the above equations to vote on the walls. Unfortunately, this method does limit the scalability of the network as a whole, as the Robot IDs are given 4 bits in the radio packet. This means the maximum size of the network is

$$n = 2^4 = 16. \tag{3.7}$$

Fifteen robots (plus one data sink) is a fairly good number for most average office-style buildings, as will be shown in Chapter 5. If more robots are required, though, we can either open up the second channel of the data sink and have two distinct, separate networks, or change the protocol to have each packet use 6 bytes per cell, which gives each robot ID 8 bits, allowing for up to 255 robots.

3.4 Computer Visualization Programs

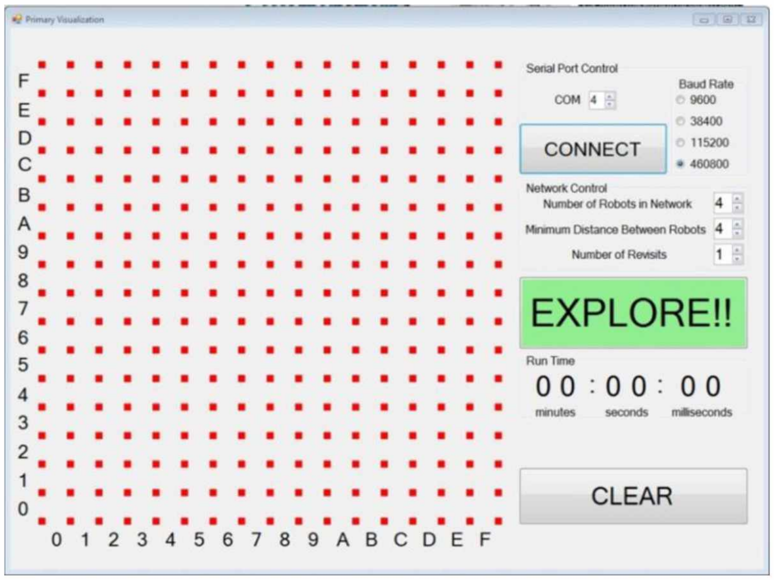
It is necessary to display the data that is collected by the robots in an easy to read and clear manner. Simply displaying the raw data would not be sufficient, as it would take too long to pick out the data that is important. It was necessary to create programs that would visualize the data in a clear and logical layout. There are two similar, but importantly distinct programs that were created for the data sink and the snoop so that that data could be visualized.

3.4.1 Primary Computer Display

The primary program was written for visualization of the data sink's data. The data sink is constrained with the radio distance limiting algorithms (explained in Chapter 4), so its purpose was to display only that data collected from robots within range of the data sink, virtually located at the “entrance” to the model building in the lower left corner. This allows the user to see what they would see were they controlling a real network of real robots in which their radios were being naturally distance limited by the building itself. This visualization, written in visual C#, was also created to control the main network of robots. With it, the human user can interface with the robots, telling them to start exploring or to stop exploring, and notifying each robot how many robots are in the network. This program also allows the user to determine how spread out the robots need to be, by setting the “minimum distance between robots” selection. The higher that number, the farther apart the robots have to stay. This program also allows the user to track how much time is required for the robots to explore the building. This primary display is shown in its starting state in Figure 3-21. It is shown after four robots have mapped the cells once (no voting), with two uncorrected errors in Figure 3-22.

3.4.2 Secondary Computer Display

The second display program was built as a way of assisting the debugging stage.

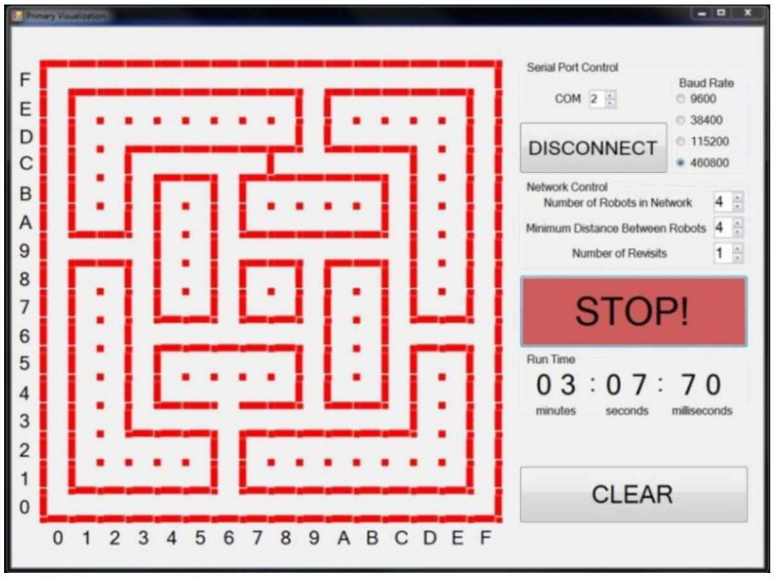


Serial Port Control
COM 4
Baud Rate
( ) 9600
( ) 38400
( ) 115200
(●) 460800

Network Control
Number of Robots in Network 4
Minimum Distance Between Robots 4
Number of Revisits 1

Run Time
00 : 00 : 00
minutes seconds milliseconds

Figure 3-21: Primary Computer Based Network Controller and Visualization Program


Primary Visualization

Serial Port Control

COM 2

Baud Rate: 9600, 38400, 115200, [✓ 460800]

Network Control

Number of Robots in Network 4

Minimum Distance Between Robots 4

Number of Revisits 1

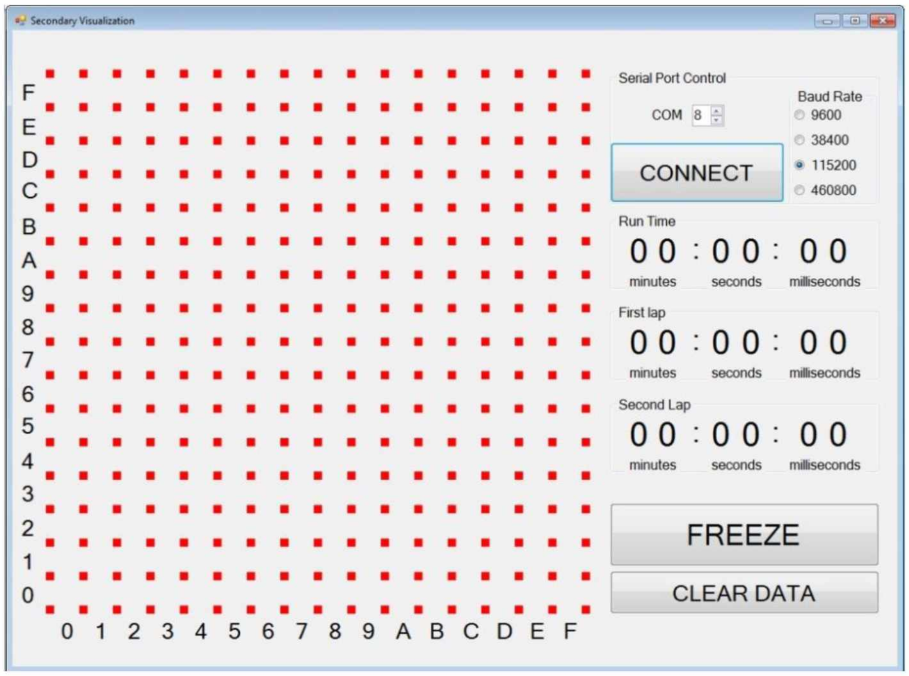
Run Time

03 : 07 : 70

minutes seconds milliseconds

Figure 3-22: Primary Visualization Showing Full Building, Two Errors

Because the primary program does not show all the data collected by the robots as they are collecting it (it is radio-limited), it became necessary to produce this visualization that would display all information that is useful for debugging. In addition to showing all the data as it was being discovered, this visualization also displays the current location of the robots, whether the robots are active or in a non-transmitting state (locked up), and it shows the times required for the robots to map each building cell one time, two times, and three times. With this program we can also “freeze” the current state of the exploration to study particular points in the exploration. This secondary program connects to the snoop device, which does not actively transmit into the network. Thus this secondary program is only for debugging purposes and has no active role in the network. The secondary program is shown in its initial state in Figure 3-23. It is shown after the building has been mapped one time in Figure 3-24.



Serial Port Control

COM 8

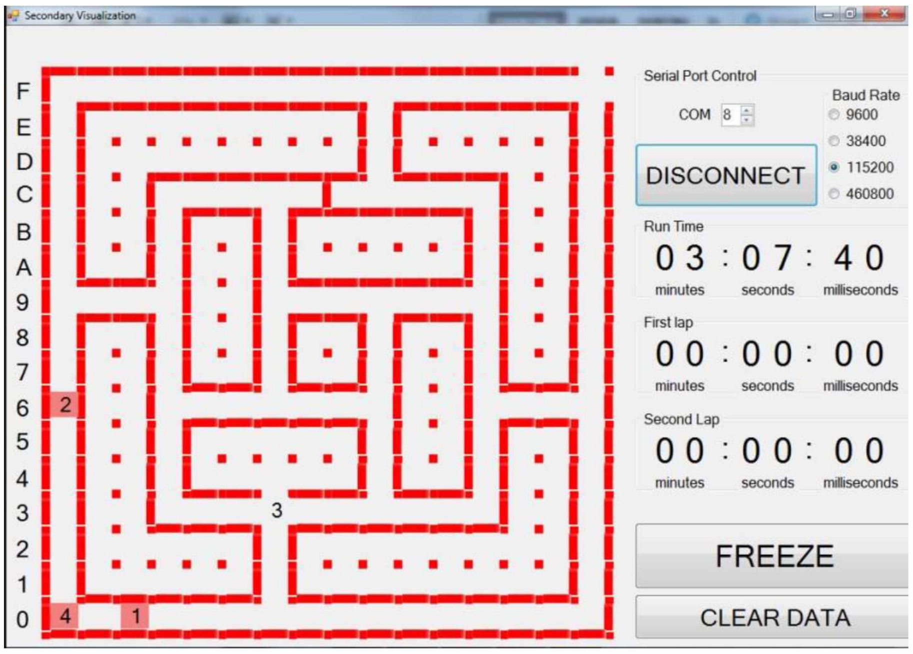
Baud Rate
( ) 9600
( ) 38400
[✓ 115200]
( ) 460800

Run Time
00 : 00 : 00
minutes seconds milliseconds

First lap
00 : 00 : 00
minutes seconds milliseconds

Second Lap
00 : 00 : 00
minutes seconds milliseconds

Figure 3-23: Snoop Program for Simulation Analysis



Serial Port Control

COM 8

Baud Rate
( ) 9600
( ) 38400
[✓ 115200]
( ) 460800

Run Time
03 : 07 : 40
minutes seconds milliseconds

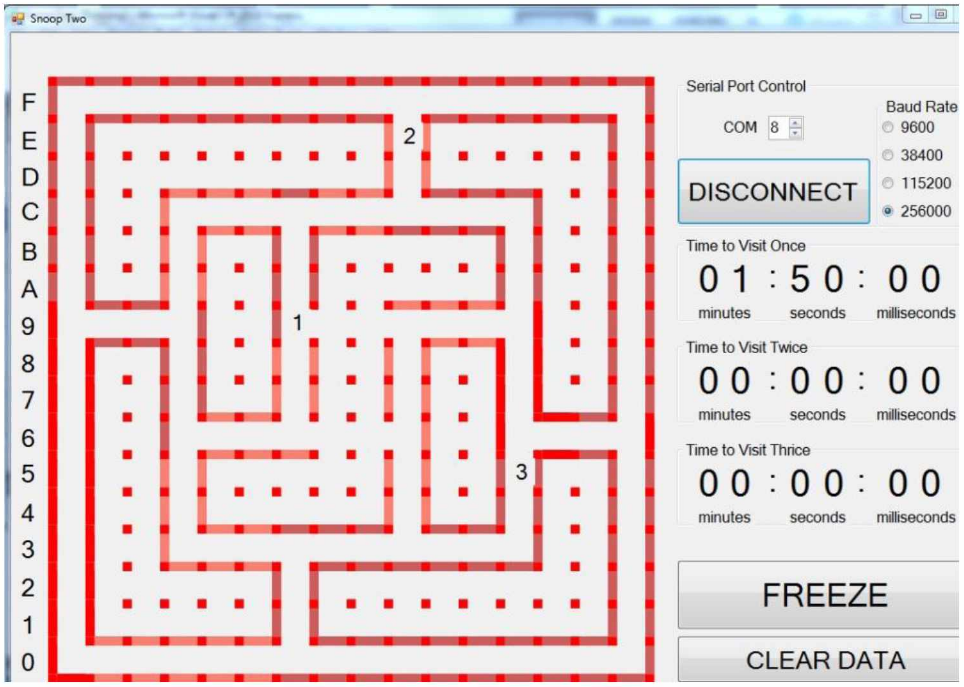
First lap
00 : 00 : 00
minutes seconds milliseconds

Second Lap
00 : 00 : 00
minutes seconds milliseconds

Figure 3-24: Secondary Visualization Showing the Full Maze and Robot Locations

3.4.3 Display of Explored Areas in the Visualization Programs

In order to see how effectively the robots are moving through the building, it became necessary to distinguish the number of times the cells have been visited. This was accomplished by having the walls turn a darker red each time they are visited, up to a maximum of three times. The walls are the background grey color if they have not been visited yet, then grow darker red each visit. This tool allows the user to analyze the search pattern for areas that are not being adequately visited, and in the future controls will be added for commanding robots to the locations that are less visited. The four wall colors can be seen in Figure 3-25.



COM 8

Baud Rate
( ) 9600
( ) 38400
( ) 115200
(●) 256000

Time to Visit Once
01 : 50 : 00
minutes seconds milliseconds

Time to Visit Twice
00 : 00 : 00
minutes seconds milliseconds

Time to Visit Thrice
00 : 00 : 00
minutes seconds milliseconds

Figure 3-25: Display of the Snoop Visualization Showing all Four States of the Walls

Chapter 4 – Conditions and Assumptions

4.1 Introduction

Inside a building or a confined space, geographical obstacles such as corners or bends in hallways and closed doors, and effects such as multipath interference can all significantly reduce a radio's transmission distance in the building. As mentioned previously, the Nordic radios we are using are capable of transmitting up to approximately 40 - 50 meters indoors, whereas the model building that is used to test the robot network is only 3 meters long. Because of this, unlike a full scale robot network exploring a full scale building, the radio is capable of transmitting throughout the entire model building. In order to adequately simulate a full scale robotic network, it was necessary to limit the distance the radio signal could transmit. The radios are capable of transmitting down to -18 dBm, which is still powerful enough to cover the entire model. Therefore, it was necessary to limit the radio transmission distance in software. Before the software radio limiting could be implemented, however, it was necessary to first obtain data about how the radios actually behaved in a real building. Section 4.2 discusses these tests.

4.2 Radio Limiting Hallway Test Results, and Subsequent Assumptions

The radios are capable of transmitting at four different power settings (-18, -12, -6, and 0 dBm) and three different data rates (250 kbits/s, 1 Mbits/s, and 2 Mbits/s). The tests were conducted in two hallways in the engineering building here at the University of Alaska Fairbanks. The first hallway has two bends in it that allow us to test how the radios perform around corners, as well as testing along a moderately long straight stretch. As shown in Figure 4-1, the first leg of the hallway is approximately 22 meters long and 1.4 meters wide, which is an average sized building hallway.

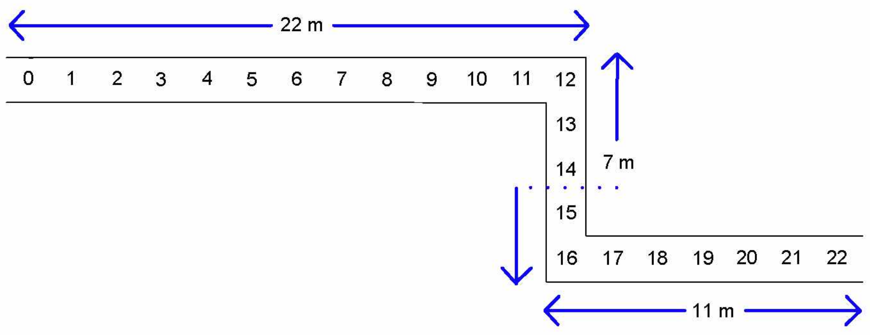


Figure 4-1: Hallway of Initial Radio Performance Test

The hallways were discretized into "cells" of approximately 1.8 meters in length. During the first test, radio node #1 was placed in the '0' cell and radio node #2 was moved one cell at a time, and in each cell the radios were tested at all three data rates for each of the four power levels. This test is bidirectional; it requires a packet to go out to node #2, which retransmits it back to node #1. The transmission is considered successful only if node #1 receives back the packet it just sent. Each test comprised of sending 3000 packets, and the amount of successful packets was recorded for each power level and data rate. The results of the first test are shown in Table 4-1.

Table 4-1: Results of Radio Reception Test 1

Power (dBm)	-18			-12			-6			0		
Data Rate (Mbits/s)	2	1	0.25	2	1	0.25	2	1	0.25	2	1	0.25
1.8	2980	2976	2979	2986	2989	3000	2971	2979	2983	3000	3000	3000
3.6	2817	2927	2935	2964	2971	2961	2767	2822	2937	2976	2981	2991
5.4	2274	2587	2871	2633	2556	2924	1313	2351	2902	2966	2961	2944
7.2	299	1158	2748	1853	2580	2878	32	24	2283	2853	2875	2938
9	1504	2644	1360	1152	2476	2544	513	22	2749	2946	2955	2900
10.8	2423	2535	2385	2827	2943	2921	2459	2390	2862	2951	2957	2948
12.6	21	2413	2640	2654	2746	2866	1361	519	2733	2894	2951	2912
14.4	715	11	2511	1826	2614	2810	0	36	2457	2765	2899	2902
16.2	2413	2701	2829	2798	2944	2915	2613	2510	2666	2933	2955	2931
18	430	5	2523	2535	2713	2847	8	0	1085	2732	2882	2871
19.8	0	0	2312	1267	2527	2745	0	59	2518	2726	2760	2926
21.6	1736	2440	2698	2710	2751	2882	354	2408	2765	2933	2964	2938
23.4	50	1891	2563	2701	2721	2870	24	862	2542	2904	2960	2937
25.2	0	0	2325	1740	2574	2852	0	0	2469	2790	2902	2921
27	0	0	1907	300	2103	2578	0	0	1922	2697	2724	2910
28.8	0	0	52	0	718	2419	0	0	2150	2619	2781	2914
30.6	0	0	0	0	0	0	0	0	0	43	18	2301
32.4	0	0	0	0	0	0	0	0	0	0	0	1427
34.2	0	0	0	0	0	2	0	0	0	0	4	2384
36	0	0	0	0	0	0	0	0	0	0	0	1300
37.8	0	0	0	0	0	0	0	0	0	0	0	5
39.6	0	0	0	0	0	0	0	0	0	0	0	0

In order to assist in deciphering the data, plots of the data are shown in Figure 4-2, Figure 4-3, Figure 4-4, and Figure 4-5.

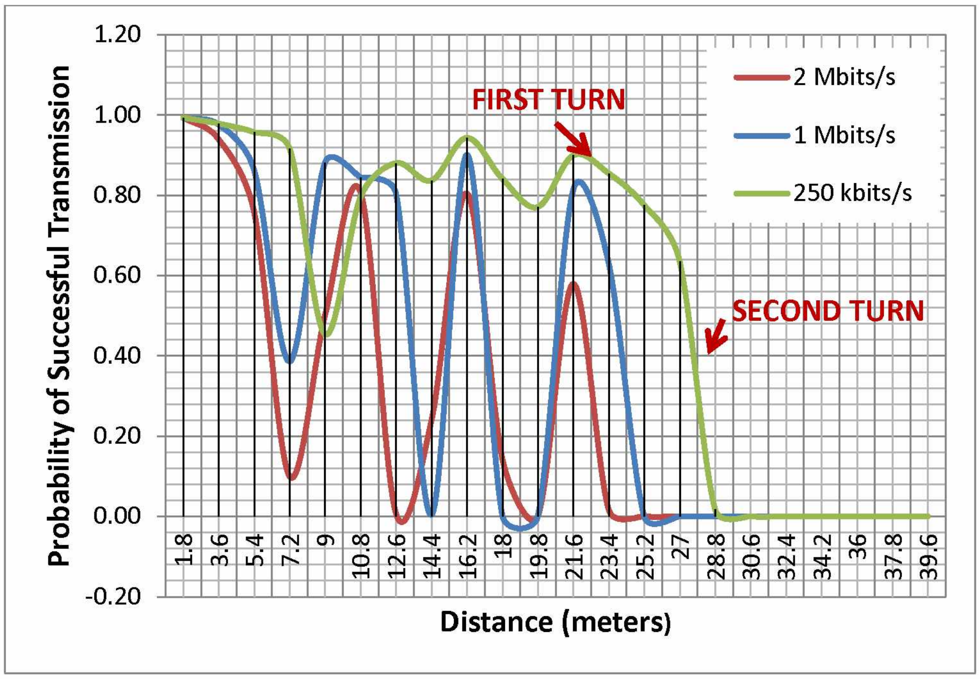


Figure 4-2: First Radio Reception Test at -18 dBm Power Level

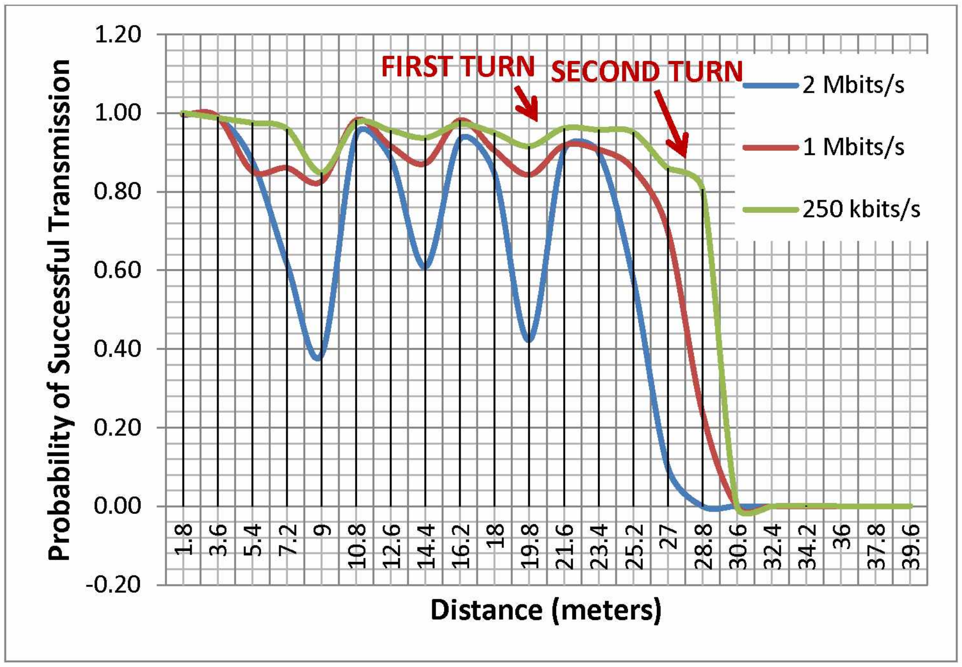


Figure 4-3: First Radio Reception Test at -12 dBm Power Level

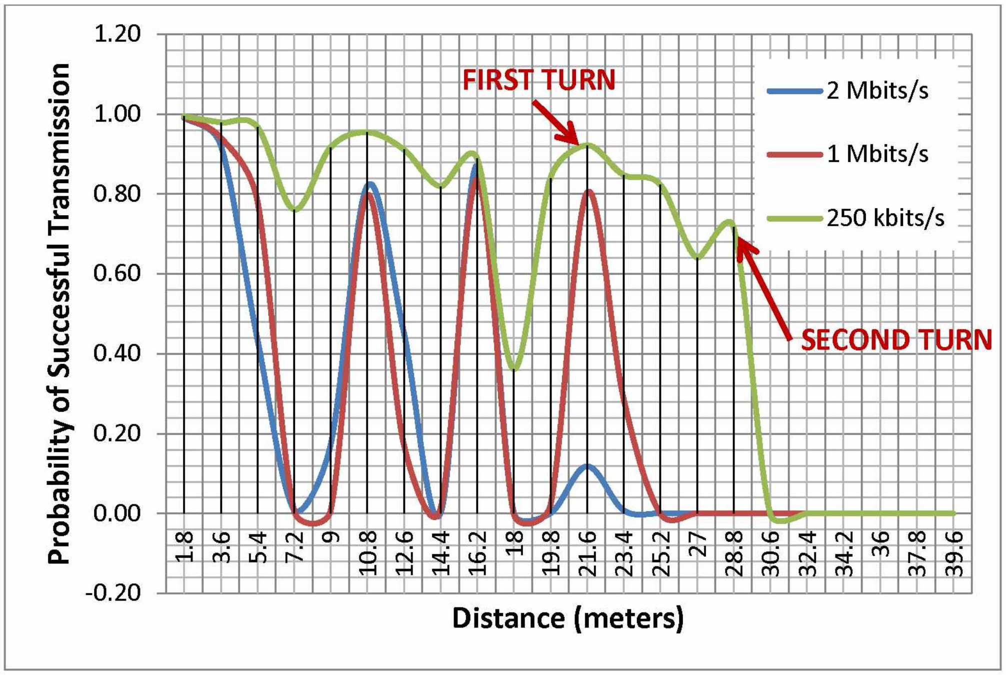


Figure 4-4: First Radio Reception Test at -6 dBm Power Level

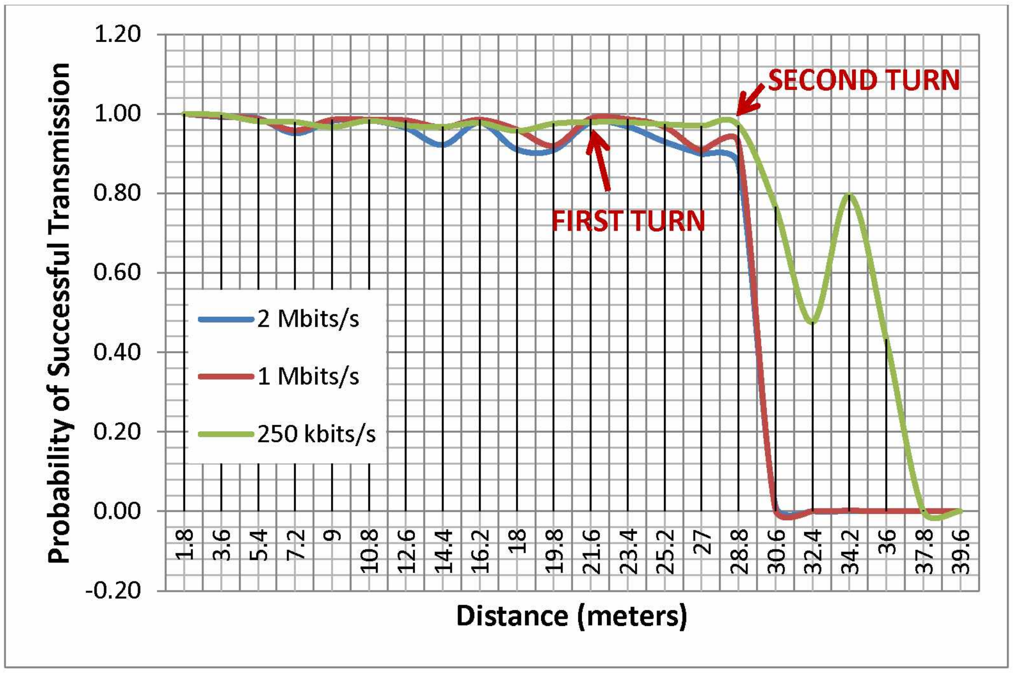


Figure 4-5: First Radio Reception Test at 0 dBm Power Level

The data shows that only the 0 dBm power setting provides a consistently high enough probability of successful transmission to be reliable in a USR network. It also shows that the lower data rate of 250 kbits/s provides higher probability of success than the other data rates. Therefore, for the network of robots, the radio power is set at 0 dBm and the data rate is set to 250 kbits/s. The data also provides evidence that no matter what the power level or the data rate, the signal does not survive much beyond the second corner at 28.8 meters. Only at the highest power level and the lowest data rate does it survive more than about 2 meters beyond it, and even then it has a decreasing probability of success. This will become important in the determination of the software radio limiting.

It was also important to see how far the radio transmission went after only a single corner. To do this, a hallway was chosen with only a single corner halfway along its length. Another hallway in the engineering building was selected to test a single corner condition. The hallway is drawn in Figure 4-6.

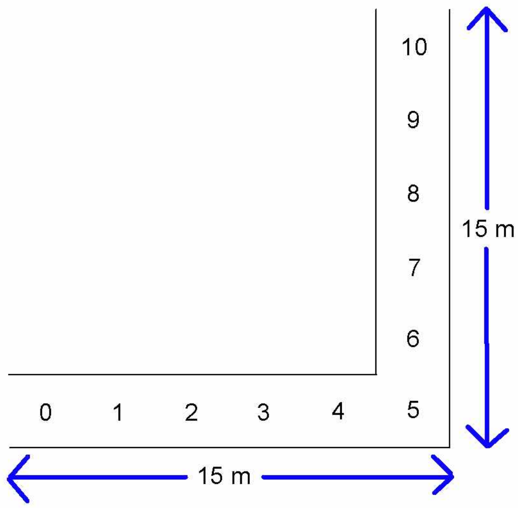


Figure 4-6: Hallway Used for the Second Radio Performance Test

Because the first test showed that the bottom three power levels had a much lower probability of success than 0 dBm, only the 0 dBm results are displayed for this second test. The test was still completed at all three data rates. The results of this test are shown in Table 4-2, and plotted in Figure 4-7.

Table 4-2: Results of Second Hallway Radio Performance Test

Distance (meters)	2 Mbits/s	1 Mbits/s	250 kbits/s
3.0	3000	3000	3000
6.0	3000	3000	3000
9.0	2991	2991	3000
12.0	2860	2775	3000
15.0 (corner)	2485	2644	2996
18.0	1	103	1976
24.0	23	684	2579
27.0	0	182	2269
30.0	0	0	4

Based on the above data, there are three observations that can be made about the radio transmissions: the radio signal will travel the length of most building hallways without any loss of signal (provided the hallway is not more than 100 meters long, which is the radio's outside transmission distance, and no significant obstacles exist in the hallway), the radio signal will carry a distance of approximately 15 meters after a single bend with some loss in signal, and the radio signal will drop out completely shortly after making a second corner.

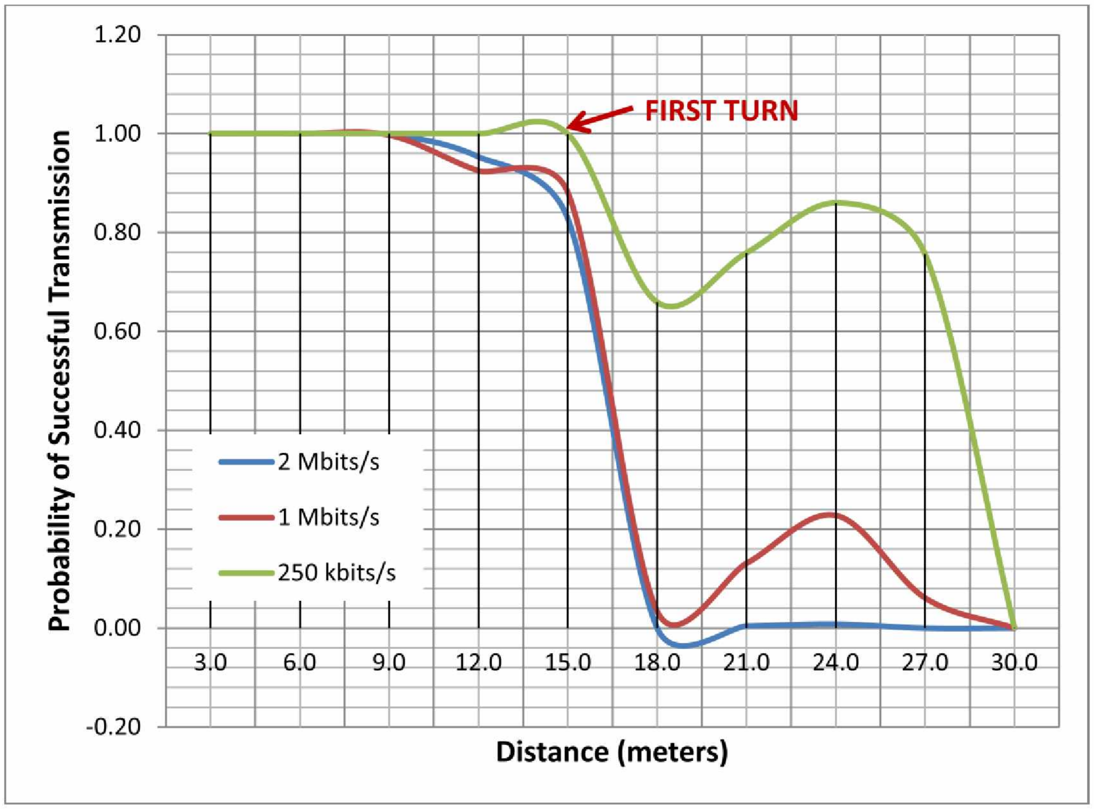


Figure 4-7: Second Radio Reception Test at 0 dBm Power Level

To assist in making a simple radio limiting algorithm, a few idealizations were made. First, any probability of successful transmission above 50% was boosted to be considered always fully successful, and any below 50% was considered to be always unsuccessful. Second, each bend or corner in the transmission path will attenuate signal by a third of its full strength. Also, the assumption was made that each cell in the model building was 3 meters in distance, making the building approximately 45 meters to a side, which is approximately the length of an average building. Given these idealizations and assumptions, the equation for simulated radio limiting is

$$\text{Probability of Success} = 16 - C_S - (5 * C_T). \tag{4-1}$$

In equation 4-1, C_S and C_T represent a single "movement" of the transmission, the former is a single straight move from one cell to another, and the latter is a turn. If the equation returns a probability of success of any number 1 through 15, then the transmission will be considered successful and the receiving node will accept the transmission. If however, the equation returns a probability of success of 0, then the transmission is ignored by the receiving node. This equation allows transmissions to travel the entire length of the building provided no turns are made. If a single turn is made, the transmission may only travel 2/3 the distance of the building. If two turns are made, then the transmission may travel forward only 1/3 the length of the building. All three cases can be seen in Figure 4-8. In these figures, the empty blue circle in the

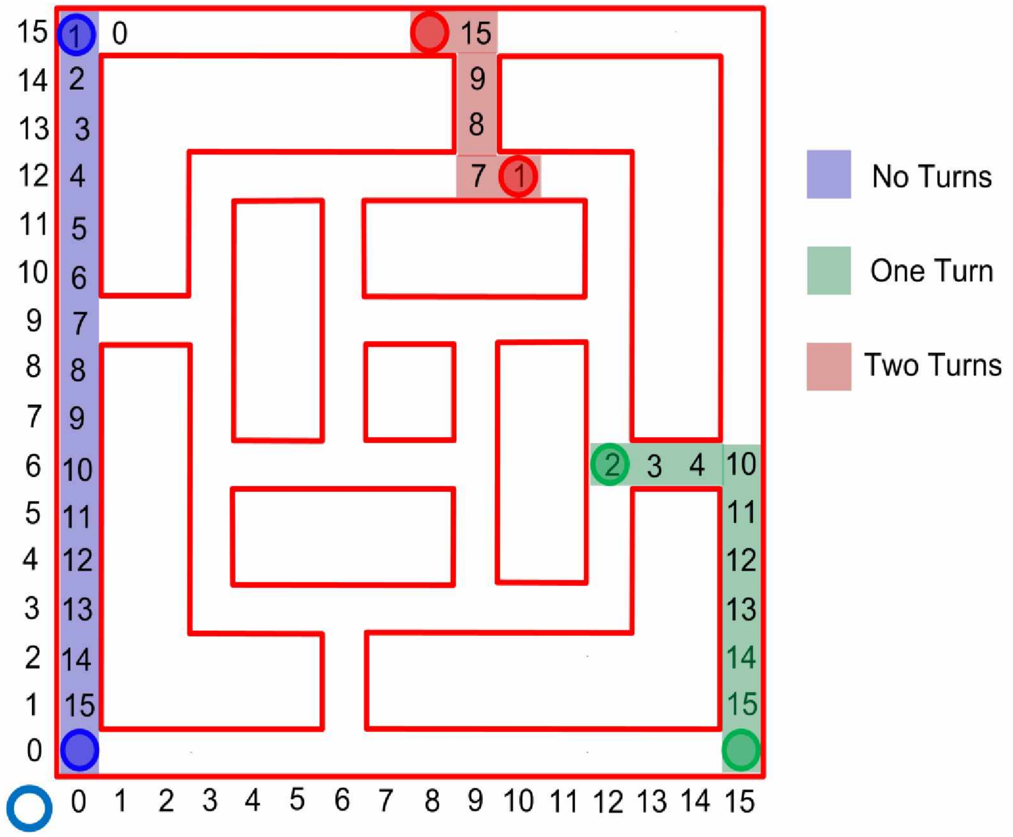


Figure 4-8: Examples of the Radio Limiting Algorithm

lower left corner represents the data sink, and the solid blue, green, and red circles represent the transmitting node, and the blue areas represent the areas in which the signal is taken as successful. The numbers are derived directly from equation 4-1.

In a real world environment, there is no calculation necessary to determine whether two nodes are in range of each other; they will either hear the other's transmission well enough to accept it or they will not. To simulate radio limiting as closely as possible, the algorithm should take very little time to determine whether the transmitting node is in range. In order to accomplish this, the radio limiting is performed by looking up the cell in a look-up table. The cell address of the transmitting node is the index to the look-up table, and all cells within range of the transmitting node will be found within the table. With this method, each robot can look up the cell within 130 µs. While this is not an instantaneous decision as it would be in a real world situation, it is sufficient for our emulation, as it does not seem to cause any negative problems in our radio protocol when running the emulation.

4.3 Sensor Interference and Subsequent Requirement for Proper Spacing

Even though the robot network employs voting mechanisms to prevent mapping errors, particularly those caused by sensor interference between robots, it is still a good idea to prevent errors from occurring by preventing sensor interference to begin with. In order to do this, it is important to determine the range of sensor coverage for each of the robots. By knowing the sensor coverage of the robots, it is possible to determine how far apart the robots must be in order to prevent interference. In order to test this, two robots were placed facing each other, and the strength of their infrared sensor signals were read between the two. The results of this test are shown Table 4-3. The numbers represent the output of the MSP430's onboard 12-bit ADC. Note that with these sensors, saturation occurs at approximately 3000.

Table 4-3: Sensor Coverage Test Results

Distance	Sensor	First Test	Second Test	Third Test	Average
No Robots in View	Right	22	22	21	22
	Left	24	25	25	25
5 Cells Away	Right	464	462	464	463
	Left	578	569	568	572
4 Cells Away	Right	800	792	788	793
	Left	824	817	809	817
3 Cells Away	Right	1244	1239	1238	1240
	Left	1218	1214	1211	1214
2 Cells Away	Right	2313	2312	2315	2313
	Left	1707	1711	1710	1709
1 Cell Away	Right	2954	2953	2953	2953
	Left	2827	2829	2829	2828

The threshold for recognizing a wall in the model building is 1000. Anything above this value, the robots will say a wall is there, and anything less than that the robots will say there is no wall. Therefore we can see that the robots should maintain a distance of at least 4 cells to prevent false-positive errors in the map due to sensor interference.

Chapter 5 Results of Implementation

5.1 Introduction to Results

In this chapter the results of all the tests completed with the robots in the model building will be reviewed and discussed. These tests include the results with and without voting protocols, as well as the results of the latency test for the reactive distribution protocol.

5.1.1 Results of Network Activity without Using Voting Protocols

In order to obtain data about the cost in time of the voting protocols, the network was run without voting. Voting protocols are especially essential in a highly dynamic environment, or an environment that creates high noise levels in the robots' sensors. Thus operators of these USR networks might opt against using the voting protocols if they have a significant impact in the time to search the building. First it must be known whether they do have any impact. Thus, the network was first tested without the protocols, and data was collected concerning the time it took to explore and map the building, as well as the number of errors that exist in the map. Table 5-1 shows the results of the time tests, and Table 5-2 shows the errors that result from mapping without voting.

Table 5-1: Time to Explore and Map Building for Different Numbers of Robots

Number of Robots	Run 1 (Sec)	Run 2 (Sec)	Run 3 (Sec)	Run 4 (Sec)	Run 5 (Sec)	Average (Sec)
1	307	313	309	312	307	309
2	172	172	172	171	174	172
3	163	158	165	162	162	162
4	95	96	95	95	97	96

Table 5-2: Number of Detection Errors versus Number of Robots

Number of Robots	Run 1	Run 2	Run 3	Run 4	Run 5	Average
1	0	0	0	0	0	0
2	0	1	0	0	0	0.2
3	2	0	1	1	0	0.8
4	1	2	1	2	2	1.6

Figure 5-1 shows the relative search time versus the number of robots in the building, and Figure 5-2 shows the average number of errors versus the number of robots in the building.

The time results are interesting in that adding the third robot had much less of an effect on the search time as adding the second and fourth robots. One possible explanation for this is the increase in robot-robot interactions. The robots are not capable of passing each other in the hallways, so when two come head-to-head, one or

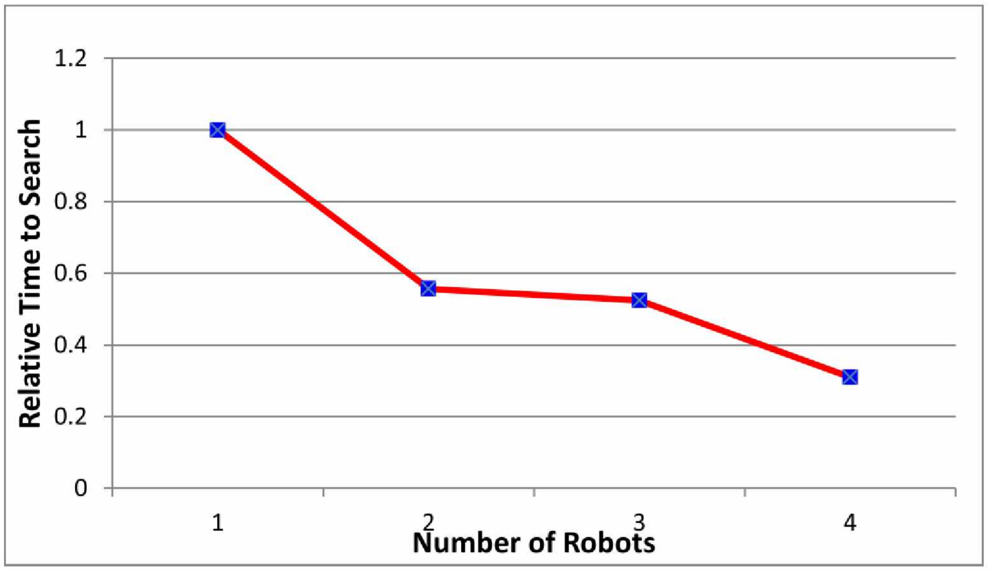


Figure 5-1: Relative Search Time as a Function of Network Size

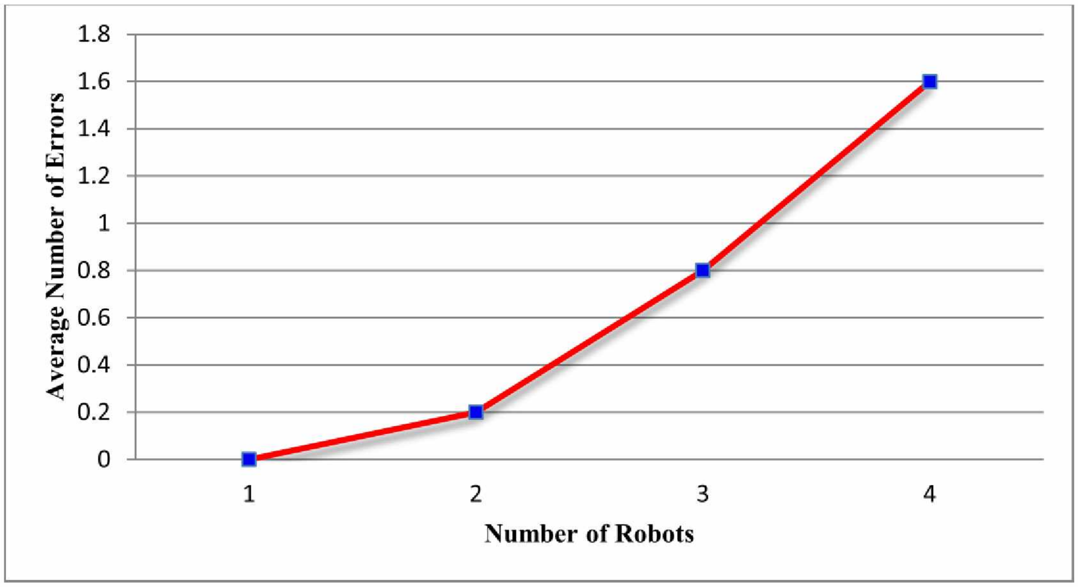


Figure 5-2: Number of Detection Errors as a Function of Network Size

both robots must turn around and find an alternate route to their goal square. With three robots in the network the number of interactions increased. Also, because of the exploration protocol that was used, three robots are able to divide the building much less efficiently, with more overlap, than two or three robots could. As the results show, using four robots in the building increases these robot-robot interactions as well, but the extra robot still allows the robots to significantly decrease the exploration time.

It should be pointed out that these tests were done with only four robots, and therefore do not reflect the behavior of a full scale network. Future work will include simulations that will show the results as network and building sizes increase up to and beyond what would be considered normal sizes.

Other than the somewhat unexpected result with a three-robot network, the overall results support the theory that increasing the network size decreases the exploration time. This decrease in time comes at the expense of accuracy; as network size increases, errors increase. The problem of errors is solved by allowing each robot to vote on the walls. This idea is explored in the next section.

5.1.2 Results of Network Activity While Using Voting Protocols

After testing how the network performed without the voting protocols in place, it was important to explore how the voting protocols affected performance. Therefore, the tests were performed again, with the voting protocols operating. In the case without voting, it was only necessary to visit all the cells in the building once. Exploration beyond that point would not result in a change in the map, even if the condition of the building changed (for example, if a wall collapsed). Therefore, if there was more than one robot in the network, each robot would not have to visit every cell. With the voting protocols in place, though, each cell would need to be visited at least three times. Depending on the number of robots in the network and the method of movement, a cell could be visited by one robot three times, or by three different robots once each. Unfortunately, just before the tests could be completed, one of the four robots broke, and the results could not be recorded with four robots in the network. However, the results obtained with up to three robots should be sufficient to draw a few conclusions. For the time being, it was decided that each robot was only able to have a single vote on each wall. This was partly due to the desire to keep the radio packets small, and time stamps were not inserted into the data being sent. Without time stamps it is not possible to allow the robots to have more than one vote, because the voting algorithm would need to distinguish between the two or three votes of a single robot. The decision was also made because it created reliability in the voting mechanism in the case that one robot had a broken sensor. If that robot were allowed more than one vote then the map of the building that the user receives could be very inconsistent. In the future, time stamps will be included and tests will be performed in which the robots do get more than a single vote, and comparisons will be made to the results in this section.

Because each robot currently is allowed only one vote, though, for a network of one robot there is little need for it to explore the building more than once. Similarly, two robots need only explore each cell twice (once each). As it is set now, three or more robots are required to accurately vote on a wall, so the voting algorithm really only works with at least three robots in the network. However, data was collected with both one and two robot networks so that comparisons could be made with the data collected without the voting protocol. Table 5-3 displays the results of the network exploration test with voting.

Table 5-3: Time Results (in Seconds) of Voting Protocol Testing

Network Size	First Vote		Second Vote		Third Vote	
	Snoop	Primary	Snoop	Primary	Snoop	Primary
1	299	299				
2	190	235	341	351		
3	183	202	284	328	392	398

Times in Table 5-3 represent the time in seconds that the visualization programs recorded a completed vote of the building. For each vote sequence, the numbers were recorded for both the snoop visualization, which shows when the robots have actually mapped the building, and for the primary visualization, which represents when the users at the entrance would have received that information. For example, it took the three robots 183 seconds to fully map the building the first time, but the human user would not have seen the completed building until 202 seconds. Also, it took 284 seconds to fully map the building a second time, but the human user would not have gotten the information from the second vote until 328 seconds. Including both times demonstrates that latency exists in the network when the robots are not required to maintain connectivity in the communication channels. This concept will be explored further in future work.

Comparing the network with voting and the network without voting, and considering only the time that it takes for the robots to explore the building the first time through, it is apparent that the voting algorithm itself has little impact on the exploration time. The results are plotted in Figure 5-3 and it is clear that the performance between the two networks is similar.

Another result that is important to consider with the voting protocol is the increase in time that is required by the network to explore the building each subsequent time. The time required for each visit level (first visit, second visit, etc.) is shown in Figure 5-4.

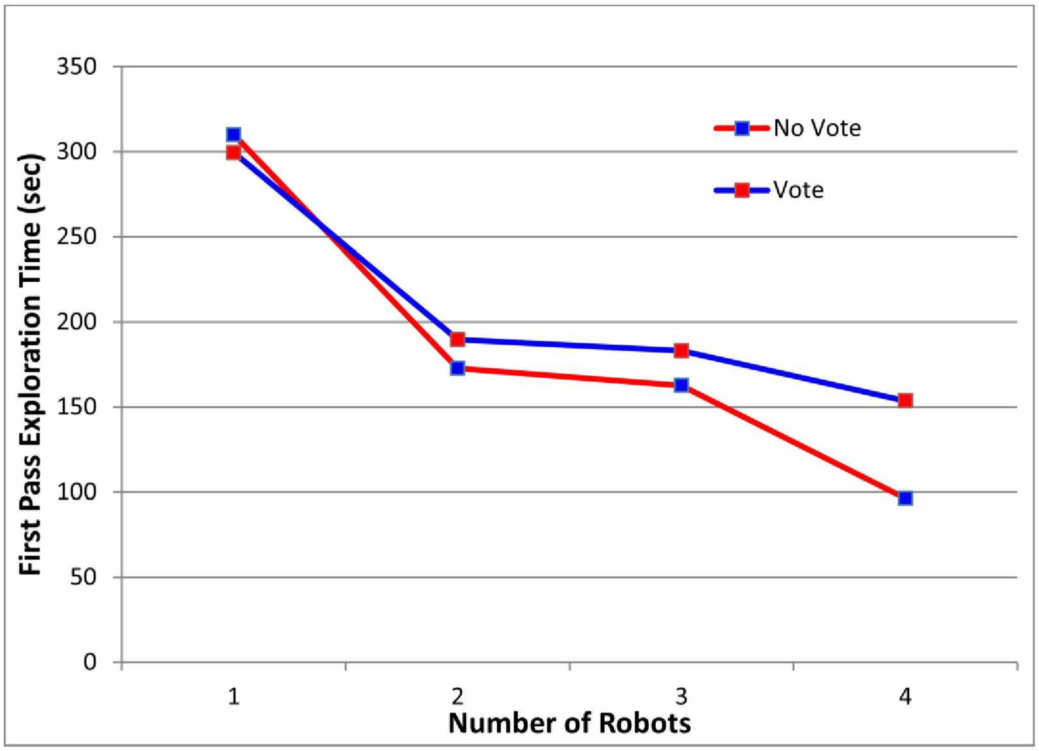


Figure 5-3: Comparison of Required Time to Map the Building One Time

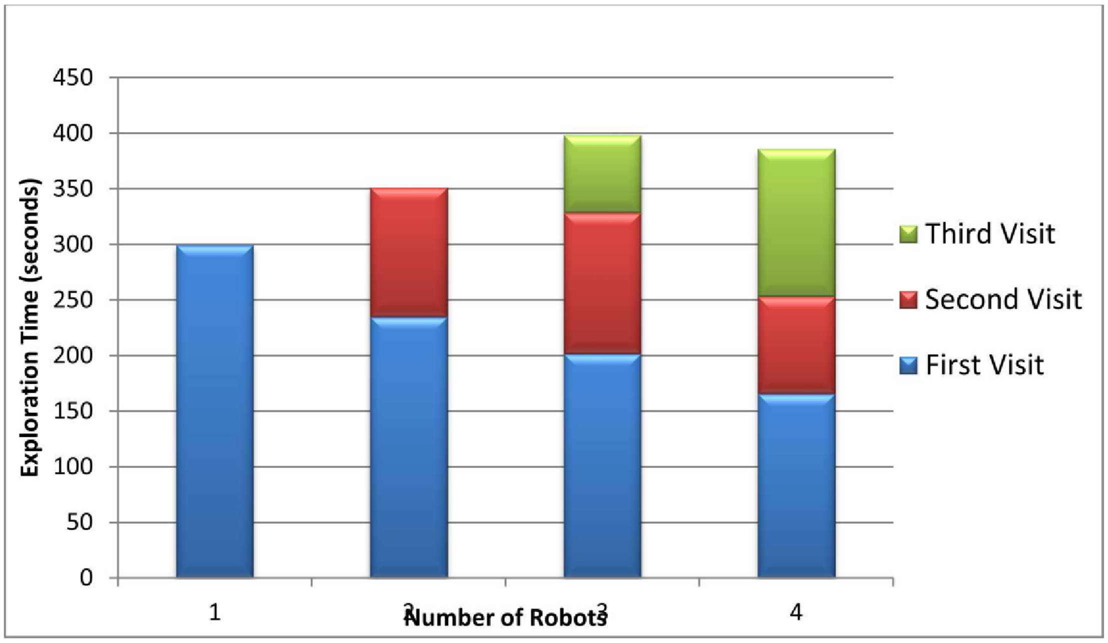
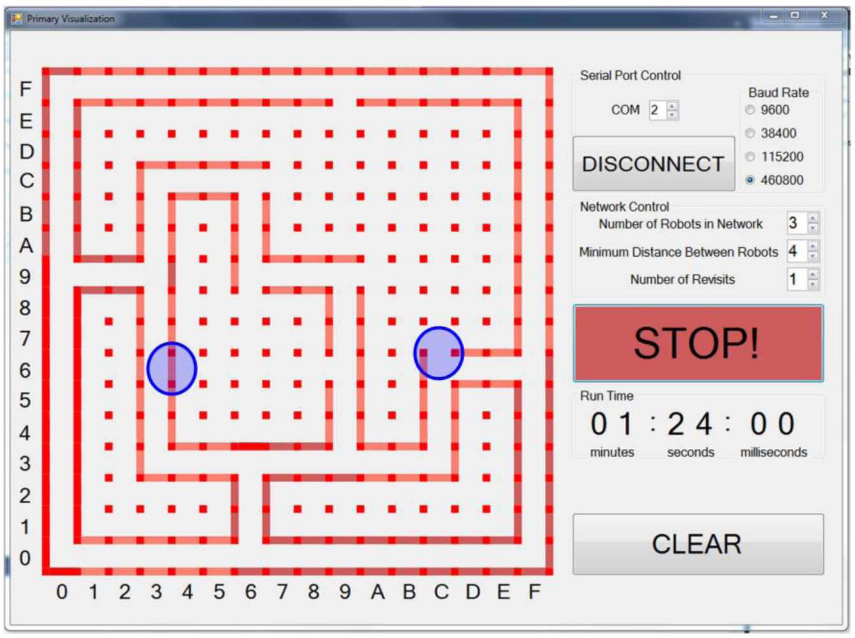


Figure 5-4: Relative Exploration Time for Each Visit Level

Note in Figure 5-4 that each subsequent visit time is shorter than the visit time before it. This is because in their first attempt at exploring the building, the robots will have to move through previously explored cells to be able to get to unexplored cells. Thus some areas in the building may be visited the required three times before another area is visited the first time. This can be seen in Figure 3-21, where some corridors have been visited at least three times, by three different robots, whereas some corridors have yet to be visited at all. This overlapping exploration along with the decrease in exploration time that comes with the addition of more robots also accounts for the fact that it takes only 33% more time for three robots to visit the maze the full three times than it does for a single robot to visit the maze once.

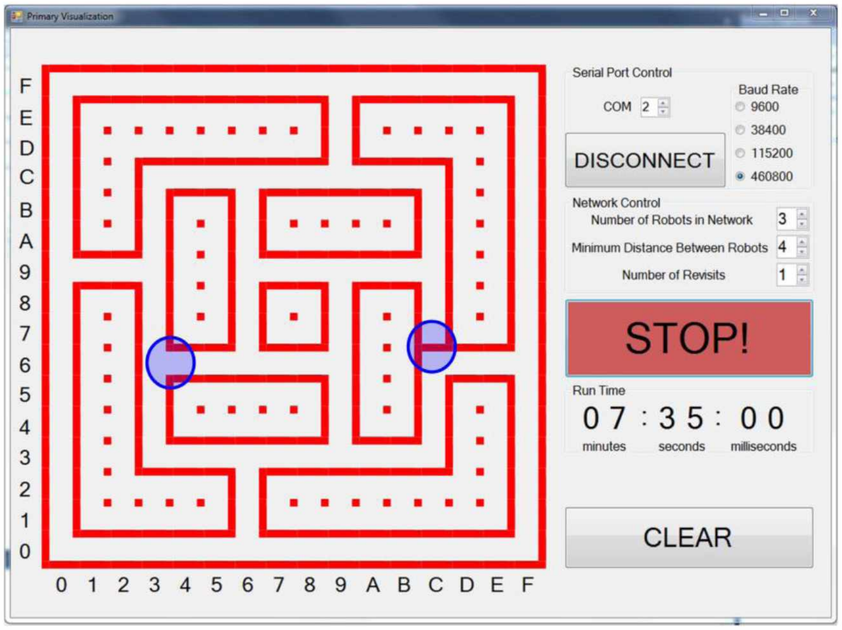
Based on the above data, it is sufficient to say that increasing the number of robots has a positive effect on the time that they require to explore a building, whether voting or not voting on the walls. Voting does not have an appreciable effect on the exploration time. It does, however, have an appreciable, positive effect on the number of errors in the building. During all tests in which voting was employed the final map was perfectly correct, even when walls were added and removed in an attempt to trick the network. Figure 5-5 and Figure 5-6 demonstrate the effectiveness of the voting algorithm. Figure 5-5 shows the visualization during the initial exploration, and though the network has not recognized it yet, there are two errors in the map, shown under the blue circles. The left circle highlights a false positive (a wall displayed in the map where in fact there is no wall) and the right circle highlights a false negative (a missing wall in the map where a wall actually exists).

Figure 5-6 shows the map after the robots have visited the building the required three times and have voted on every wall. Note that the errors have been corrected, and under the left circle, where the false-positive wall had been is now an empty corridor, and under the right circle where the false-negative space had been is now the wall that actually existed in the building. It is apparent that voting is working.



Serial Port Control
COM 2
Baud Rate
9600
38400
115200
[✓ 460800]
Network Control
Number of Robots in Network 3
Minimum Distance Between Robots 4
Number of Revisits 1
Run Time
01 : 24 : 00
minutes seconds milliseconds

Figure 5-5: Initial Exploration Run in Which Errors Were Created



Serial Port Control

COM 2

Baud Rate

9600 38400 115200 [✓ 460800]

Network Control

Number of Robots in Network 3

Minimum Distance Between Robots 4

Number of Revisits 1

Run Time

07 : 35 : 00

minutes seconds milliseconds

Figure 5-6: Final Result of Exploration Where Errors Were Corrected

The results achieved with the voting algorithm suggest that it is worth using every robot in the network to act as exploration members with ability to sense and vote on their surroundings. The ability for the network to vote on the map has only a small increase in the time of exploration of the building, yet eliminates most of the errors that exist in the map.

Chapter 6 Conclusion

6.1 Conclusion

Urban search and rescue robots are quickly becoming advanced enough to be able to replace humans, thus removing human rescuers from danger. This thesis has covered the construction of a small model robot network, in which a naïve flooding protocol and a flood fill based exploration protocol was employed to map a model building as expeditiously as possible. The ability of this network, using the mentioned protocols, was tested to discover how search time and map errors were affected by network size.

From the research accomplished with this emulated network, some notable advantages and disadvantages were observed. With increasing network sizes, exploration time decreases, which is a logical result. However increasing the number of robots also increases the interactions between robots, and the sensor interference that results from those interactions creates errors in the map. Without mechanisms for detecting and correcting those errors, increasing network size also increases the number of errors in the map of the building. However, when voting protocols are employed, those errors can be corrected with little extra mapping time. Increasing network size does increase the amount of data sharing between robots, which decreases latency and increases throughput in the network. However the extra robots also create more data to be moved in the network, and depending on the amount of data each robot collects and pushes through the network, there could be an inflection point at which adding more robots into the network decreases network performance. Finally, one important disadvantage to larger network sizes is cost. Unfortunately the cost of the network is an important consideration, and more robots will cost more money. In the end, the number of robots that should be employed will be a factor of cost and the size of the area to be explored. Adding more robots, though, at least to some point, is going to be advantageous in a USR robot network.

6.2 Future Work

Future work will primarily focus on exploration and testing of the concept of reactive distribution algorithms that allow fixed network sizes to explore buildings larger than they could otherwise explore with a proactive algorithm, and compare the results to tests of proactive distribution algorithms under similar conditions. With the robots physically built and working it will be possible to make these comparisons in an emulation device, but that will necessarily be a small version. The tests will also be done in simulation that would allow us to test network sizes and building sizes that would be out of the scope of a real world condition.

Urban search and rescue robots will always require human handlers, people who can operate or command the robots and who will collect and analyze the data. However, as mentioned in the introduction, it will be a long time before robots have the capabilities that humans have, and thus the most optimum search and rescue team will be a heterogeneous mix of robots and humans, working together to perform the search and rescue tasks. The first step to this end will be to add a remotely controlled unit into the current model network, and explore how the robots will handle a member that is not communicating its exact position and orientation, as would be the case with a human agent. The human agent will also be given the ability to control all the other robots in the network, which would be expected in a full size network.

Ultimately the final stage would be the construction of a full scale network exploring an actual building with humans integrated into the network. At this scale, the humans would carry a device with an odometer for localization, sensors for detecting chemicals or seats of fire, and a radio for communicating this data to the robots in the network and any human operators outside the building. The humans would also carry a wrist device that would allow them to see the data collected about the building and the location of the robots. This same device would allow the humans to command the robots inside the building in order to better assist in the exploration of the building.

Chapter 7 References

[1] M.K. McLelland and V. Emamian, "Application of Cooperative Wireless Networks in Reducing Power Consumption of Robots Exploring a Planet," in *IEEE Int. Conf. System of Systems Engineering*, San Antonio, TX, 2007, pp. 1-4.

[2] J. Chen and L. Li-Ren, "Path planning protocol for collaborative multi-robot systems," in *2005 IEEE Int. Sym. Computational Intelligence in Robotics and Automation*, Espoo, Finland, pp. 721- 726.

[3] K.A. Luthy, E. Grant and T.C. Henderson, "Leveraging RSSI for Robotic Repair of Disconnected Wireless Sensor Networks," in *2007 IEEE Int. Conf. Robotics and Automation*, Roma, Italy, pp.3659-3664.

[4] Siyao Fu, Zeng-Guang Hou and Guosheng Yang, "An indoor navigation system for autonomous mobile robot using wireless sensor network," in *2009 Int. Conf. Networking, Sensing and Control*, Okayama, Japan, pp.227-232.

[5] Xinkai Kuai, Kehu Yang, Siyao Fu, Rui Zheng, Guosheng Yang, "Simultaneous localization and mapping (SLAM) for indoor autonomous mobile robot navigation in wireless sensor networks," in *2010 Int. Conf. Networking, Sensing and Control (ICNSC)*, Chicago, IL, pp.128-132.

[6] C.M. Clark, S.M. Rock and J.C. Latombe, "Motion planning for multiple mobile robots using dynamic networks," in *2003 IEEE Int. Conf. Robotics and Automation*, Taipei, Taiwan, pp. 4222- 4227.

[7] Sujin Park, Jin Hong Jung and Seong-Lyun Kim, "Cooperative path-finding of multi-robots with wireless multihop communications," in *6th Int. Symp. Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks and Workshops*, Berlin, Germany, 2008, pp.673-678.

[8] Y. Mostofi, A. Gonzalez-Ruiz, A. Gaffarkhah and Ding Li, "Characterization and modeling of wireless channels for networked robotic and control systems - a

comprehensive overview," in *2009 Int. Conf. Intelligent Robots and Systems*, St. Louis, MO, pp.4849-4854.

[9] F. Zeiger, N. Kraemer and K. Schilling, "Commanding mobile robots via wireless ad-hoc networks — A comparison of four ad-hoc routing protocol implementations," in *IEEE Int. Conf. Robotics and Automation*, Pasadena, CA, 2008, pp.590-595.

[10] Zhigang WANG, Lichuan LIU and MengChu ZHOU. (2005, Dec.). Protocols and Applications of Ad-hoc Robot Wireless Communication Networks: An Overview. *Int. J. Intelligent Control and Systems*. [Online]. *10(4)*, pp.296-303. Available: http://www.asmemesa.org/IJICS/files/31/5-Wang-296-303.pdf

[11] K. Guan, D. Imbrenda, R. Ghanadan and J. Hsu, "Distributed sensing and communications in tactical robotic networks," in *Military Communications Conference*, San Diego, CA, 2008, pp.1-7.

[12] H. Sugiyama, T. Tsujioka and M. Murata, "Collaborative movement of rescue robots for reliable and effective networking in disaster area," in *2005 Int. Conf. Collaborative Computing: Networking, Applications and Worksharing*, San Jose, CA, pp.7.

[13] H.C.H Chiu, B. Ryu, Hua Zhu, P. Szekely, R. Maheswaran, C. Rogers, A. Galstyan, B. Salemi, M. Rubenstein, Wei-Min Shen; , "TENTACLES: Self-configuring robotic radio networks in unknown environments," in *Int. Conf. Intelligent Robots and Systems*, St. Louis, MO, 2009, pp.1383-1388.

[14] A. Simonetto, P. Scerri and K. Sycara, "A mobile network for mobile sensors," in *2008 11th Int. Conf. Information Fusion*, Cologne, Germany, pp.1-8.

[15] T.C. Chieuh, R. Krishnan, P. De and I.H. Chiang, “A networked robot system for wireless network emulation,” in *1st Int. Conf. Robot communication and coordination*, Piscataway, NJ, 2007, pp. 1-8.

[16] A. Birk, S. Schwertfeger, and K. Pathak, “A networking framework for teleoperation in safety, security, and rescue robotics,” *Wireless Communications, IEEE, vol. 16, no. 1*, pp.6-13, Feb, 2009.

[17] M.F. Mysorewala, D.O. Popa, V. Giordano and F.L. Lewis, "Deployment Algorithms and In-Door Experimental Vehicles for Studying Mobile Wireless Sensor Networks," in *Seventh ACIS Int. Conf. Software Engineering, Artıficial Intelligence, Networking, and Parallel/Distributed Computing*, Las Vegas, NV, 2006, pp.290-298.

[18] N. Correll, J. Bachrach, D. Vickery and D. Rus, "Ad-hoc wireless network coverage with networked robots that cannot localize," in *IEEE Int. Conf. Robotics and Automation*, Kobe, Japan, 2009, pp.3878-3885.

[19] A.A. Somasundara, A. Kansal, D.D. Jea, D. Estrin and M.B. Srivastava, “Controllably mobile infrastructure for low energy embedded networks,” *IEEE Trans. Mobile Computing*, vol. *5, no. 8*, pp.958-973, Aug, 2006.

[20] R. Suzuki, Y. Tobe and K. Sezaki, "Session Control Protocol Exploiting Controlled Mobility in Multi-Robot Sensor Networks," in *32nd IEEE Conf. Local Computer Networks*, Dublin, Ireland, 2007, pp.395-402.

[21] N.S.V. Rao, Qishi Wu; S.S. Iyengar and A. Manickam, "Connectivity-through-time protocols for dynamic wireless networks to support mobile robot teams," in *IEEE Int. Conf. Robotics and Automation*, Taipei, Taiwan, 2003, vol.2, pp. 1653- 1658.

[22] S. Radhakrishnan, G. Racherla, C.N. Sekharan, N.S.V. Rao and S.G. Batsell, "DST-A routing protocol for ad hoc networks using distributed spanning trees," in *Wireless Communications and Networking Conf.*, 1999, vol.3, pp.1543-1547.

[23] M. Chiodo, P. Giusto, A. Jurecska, H.C. Hsieh, A. Sangiovanni-Vincentelli and L. Lavagno, “Hardware-Software Co-Design of Embedded Systems,” *IEEE Micro*, vol.14, no. 4, pp. 26-36, Aug, 1994.

[24] The Donald O. Pederson Center for Electronic Systems Design. (n.d.). [Online]. Available: http://embedded.eecs.berkeley.edu/Research/hsc/abstract.html [March 2012].

[25] Texas Instruments. (n.d.). [Online]. Available: http://www.ti.com/products/tps63000 [March 2012].

[26] D. Raskovic and D. Giessel, "Dynamic Voltage and Frequency Scaling for On-Demand Performance and Availability of Biomedical Embedded Systems," *IEEE Transactions on Information Technology in Biomedicine*, vol. 13, no. 6, pp. 903-909, 2009.

[27] STMicroelectronics. (n.d.). [Online]. Available: http://www.st.com/internet/analog/product/63141.jsp [March 2012].

[28] Nordic Semiconductor. (n.d.). [Online]. Available: http://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01P [March 2012].

[29] S. Mishra and P Bande, "Maze Solving Algorithms for Micro Mouse," *2008 Int. Symp. Signal Image Technology and Internet Based Systems,* Inderprastha Eng.Coll., Ghaziabad, 2008, pp. 86-93.