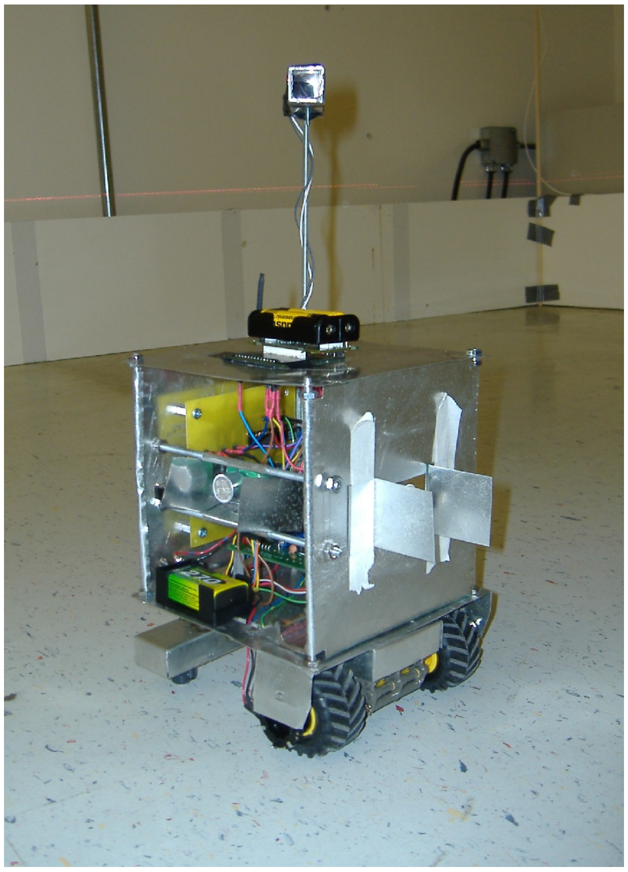
Traversing the Unknown:
The Design and Implementation of a Mobile Sensor Network



EE 382
Introduction to Design
Dr. El-Osery and Dr. Wedeward

Prepared by: Group 4
Ariel Gardner
Alexander George
Eleazar Martinez
Karina Muñoz
Philip Pinkston
Patricio Pino
Daniel Winters

Submitted on: May 8, 2006

Table of Contents

List of Appendices

All Appendices are attached as text files.

List of Figures and Tables

Abstract

The following report shows the design and implementation of a mobile sensor network. The goal was to build such a network that was autonomously capable of finding a small black box, which was placed randomly in an enclosed searching area. This task had to be completed in 3.5 months. The nodes had to search independently, localize in real-time, implement wireless communication using the micaZ microcontroller, and converge on the box when it was found.

We decided to utilize wireless communication for transmitting data from the nodes to the base station. The nodes would be responsible for data collection and motor control, while the base station would compute the calculations necessary to move the nodes the correct distances and directions. Once the robots were localized, they would go to opposite corners of the searching area, follow adjacent sides of the searching area, detecting distances to the opposing walls until a different distance was obtained. Once the box was confirmed by one robot, both robots would converge on the box.

On the day the project was due, we encountered communication errors between our nodes and base station, and were not able to demonstrate our final design. We were able to get our localization working and node locations displayed on the GUI. We simply ran out of time before we could test out our searching algorithm. If we had more time, we could have completed a fully functional mobile sensor network.

Keywords: Localization, MicaZ, Ultrasonic Sensor, Node

Introduction

The goal of this project was to build a network of mobile sensing nodes capable of detecting and confirming the location of a 30 cm by 30 cm by 30 cm black box within a closed 3 m by 3 m searching area. The requirements of the project are as follows:

1. The nodes must localize in real-time. The base station may assist the nodes in localizing. The nodes will be placed randomly in the searching area.
2. At least two nodes must be utilized, have a maximum size of 15 cm long by 15 cm wide, and have independent searching capability.
3. The network must operate wirelessly using the micaZ base station, which is interfaced with a computer.
4. A Graphical User Interface (GUI) must be used to display the locations of the nodes with a fixed reference. It must have a refresh rate of at least 2 seconds, have a start button, and display the location of the box once it's found.

The following materials have been given to each group to complete this project:

1. 4 micaZ microcontrollers
2. 1 micaZ base station
3. 2 micaZ interface boards
4. 3 motor Kits with wheels
5. $250 budget

The purpose of this project is two fold. The first objective is to give electrical engineering students the opportunity to work in teams, integrate projects, and communicate effectively. The second is to apply this project in searching for other small objects such as bombs or questionable packages.

The purpose of this report is to give engineers the ability to reproduce and troubleshoot this design. The pre-requisites for this project are as follows:

1. An in-depth knowledge of microcontrollers.
2. Analog and digital circuit design.
3. Principles in linear time-invariant systems.
4. Proficiency in C Programming

The class was divided into four groups, all of which were responsible for completing the same project. The groups were responsible for dividing up the labor between the members of the group. We divided this project into two objectives to complete: hardware design and software design. The hardware consisted of the nodes themselves, the sensors used, and the motors and gearboxes for moving the node. The software consisted of our localization algorithm, searching algorithm, GUI, and motor control.

Hardware

Designing our node started with ideas on what components and materials we should use. When we started discussing the dimensions of the chassis, the group had a hard time visualizing the design. At this point, we decided to draw scale models of all the components and chassis to better visualize what we were getting ourselves into.

Use of AutoCAD

We decided to use a computer program called AutoCAD 2004. AutoCAD is a computer aided drafting (CAD) software which allows us to draw 2-Dimensional and 3-Dimensional models of any component that we might be using in our project. By drawing all the components that we were thinking about using, we were able to see if our chassis design was going to be efficient enough for the internal components that we were using. Some of the drawings that we made in AutoCAD were the MicaZ's, H-Bridge, Ultrasonic Sensors and chassis. With this information we were able to gather what we needed to start building our node.

Frame

In the initial design we decided to maximize the size of our node. We were allowed to construct a node that had a max length and width of 15 cm by 15 cm. We started by using a two-tier node, which gave us the freedom to place components in the node without tight space restrictions. It also minimized disruptions in the performance and communication of each working component.

There were very few materials that we used in the fabrication of our frame. First of all, we used aluminum sheets for the upper and lower levels of our platform. We did this for strength and stability. All the other components, such as the base, rear axel, and walls were made of tin. Tin can be shaped and soldered easily, unlike aluminum. All the components that we built and put together were fastened by threaded rod and nuts, which made the node more stable.

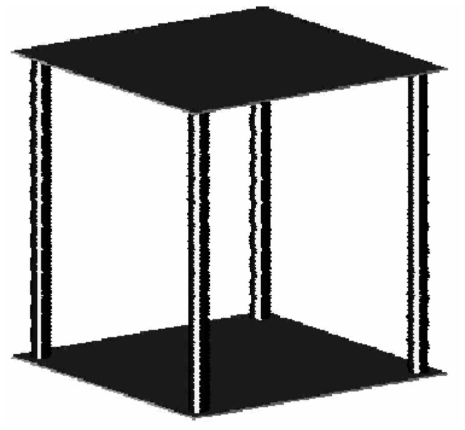


Figure 1: The Frame Design

Since we needed a place to mount the ultrasonic sensors, we constructed walls in the front and rear of the node. Mounting these walls in these areas gave our two-tier platform the stability it needed for the components that were being placed on it.

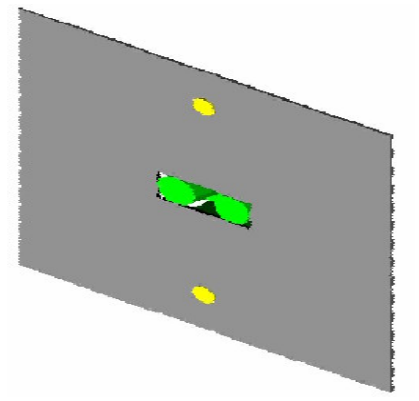


Figure 2: Front and Back Wall

Chassis and Ball Caster

Since we were using the entire area allowed, we had to modify our chassis to sit on a platform which was mounted to the gearboxes. This gave us a more stable chassis to work with.

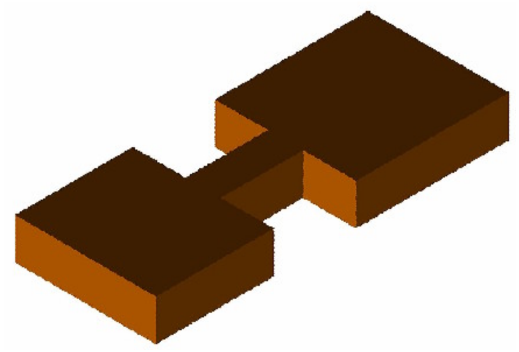


Figure 3: The Chassis

We needed a rear wheel mount to support the rear of the node. We decided to use ball casters for our rear support. These ball casters gave us the ability to turn in any direction with minimal resistance. After we knew what ball casters we were going to use, we constructed a rectangular axle 3 cm by 2 cm by 14 cm. This is where the ball casters were mounted.

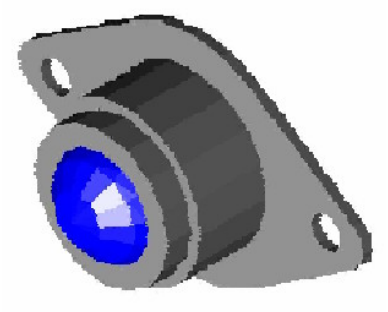


Figure 4: Ball Caster

H-bridge

The H-bridge, made by Lynxmotion, is used for motor control. It uses digital logic to control how power is distributed to the motors. It can take up to 45 V for powering the motors, but 5 V is typically used. It is typically used for controlling two motors, such that the node can move in all directions.

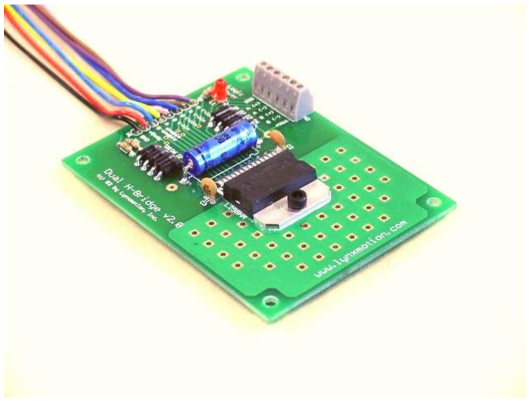


Figure 5: H-Bridge, courtesy of Lynxmotion

MicaZ

The micaZ was the microcontroller that we were required to use. We used this microcontroller because of its wireless communication capabilities. It is made by ATmega, has 52 pin connections, and acts as the “brain” of our nodes. We used an interface board to communicate with the sensors and control the motors via the H-bridge.

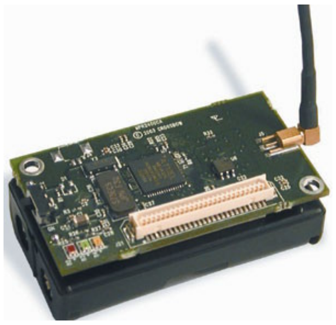


Figure 6: MicaZ, courtesy of ATmega

The base station is responsible for interfacing the micaZ with a computer. It lets the computer do the complex calculations of the sensor information collected. The base station will assist both nodes in localization and searching. It will also relay the information to the GUI.

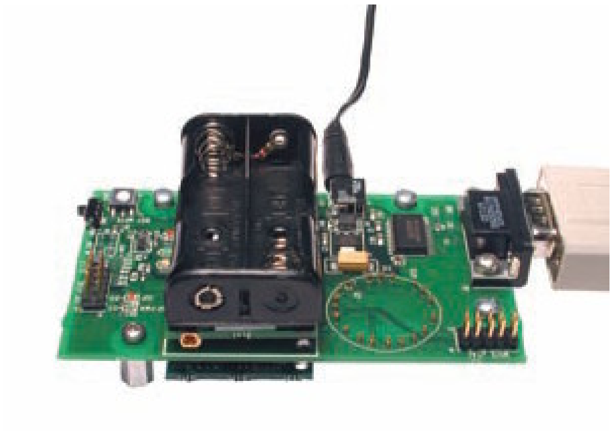


Figure 7: Base Station with micaZ and serial port connected, courtesy of ATmega

Ultrasonic Sensors

The SRF04 Ranger is a Reflective Ultrasonic Sensor (RUS) that is made by Devantech, and has a range of over 3 m. It uses sound waves to detect distances according to the following graph.

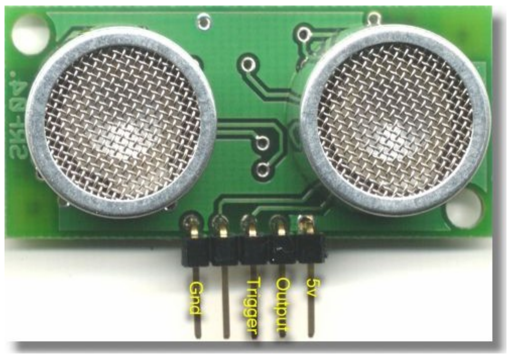


Figure 8: SRF04 Ranger Reflective Ultrasonic Sensor courtesy of Acroname

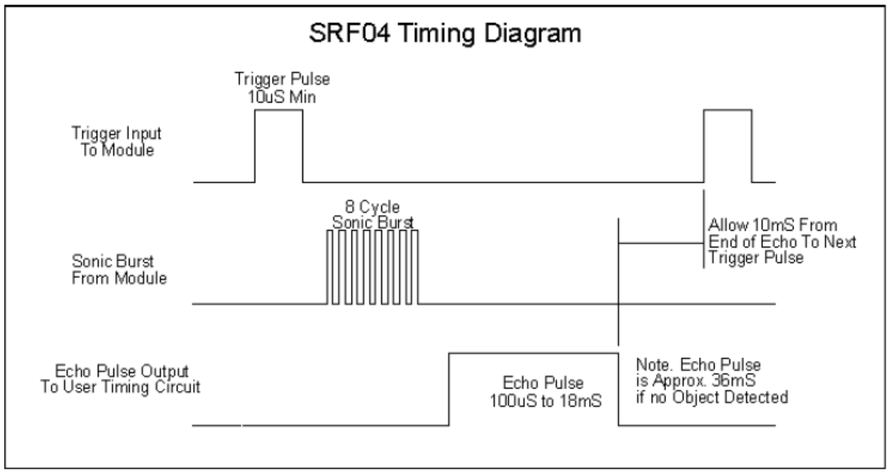

SRF04 Timing Diagram
Trigger Pulse 10uS Min
Trigger Input To Module
8 Cycle Sonic Burst
Allow 10mS From End of Echo To Next Trigger Pulse
Sonic Burst From Module
Echo Pulse Output To User Timing Circuit
Echo Pulse 100uS to 18mS
Note. Echo Pulse is Approx. 36mS if no Object Detected

Figure 9: Operation Graph for SRF04 Courtesy of Acroname

Encoders

An encoder is a sensor which converts rotary motion or position to a series of electronic pulses. The type of encoder that we used was an open center 27 mm with 15 pulses per revolution. The encoders will be mounted to the outer part of the front wheels and supplied with 3V source. As the wheel turns the encoder will oscillate between 3 and 0 volts. As the value of the encoder changes, the MicaZ will count each oscillation, and determine how far the wheels have turned.

Figure 10: Panasonic EVQ-WFP00415B Encoder.

Component Placement

Because of the RF communication that the micaZ was capable of, we mounted it upside down on the upper level of our node. This increased our signal strength and reduced noise between the micaZ and the base station.

The H-Bridge, which controlled power to the motors, was placed on the bottom level. This made the wiring of the motors and H-Bridge convenient and simple to connect. Other components that were mounted on the bottom level were two alkaline 9V batteries that connected to a 5V voltage regulator, which connected to the H-Bridge and motors.

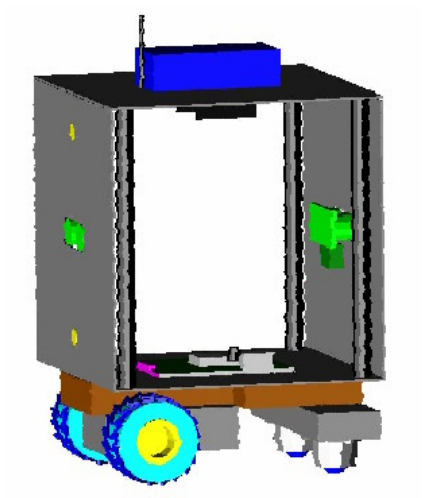


Figure 11: Initial Design of our node in AutoCAD

The last components to be mounted on our node were the two encoders. These were mounted to the outside of each front wheel, and were fastened to the frame of our node by a tin bracket. Since we did not have a circuit board in our design at the time, we placed a breadboard on the top level for simple modifications.

Final Design

Very few changes were made to the initial design. After initial testing, we discovered that making these changes made the programming simpler.

Ultrasonic Placement

The first change that we made was the 90 degree ultrasonic placement. Depending on the node, we mounted the rear ultrasonic sensor on the right or left side. This made the new orientation program more efficient. When the rear ultrasonic sensor was moved we had room to mount the circuit board to the rear wall.

Ball Casters

The casters that were initially used on the nodes did not rotate very well due to the node not having enough weight. Therefore, we decided to use a different model of ball caster that was built for the light weight of our nodes. These ball casters worked much better in the mobility of our node.

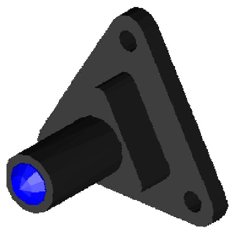


Figure 12: Final Ball Caster Used.

Circuit Board

In order to simplify wiring for this project a printed circuit board (PCB) was designed using Eagle 4.1 and fabricated using the New Mexico Tech Electrical Engineering Tech Room facilities. This PCB served to create common grounds and voltage sources for mobile node components as well as provide pin outs for connections to the MicaZ microcontroller, the H-bridge motor control, 9V battery supplies, the orientation phototransistor, the wheel encoders, and the ultrasonic sensors. The following schematic (Figure 1) features all necessary pin out connections as well as the basic components used to interface basic functions together.

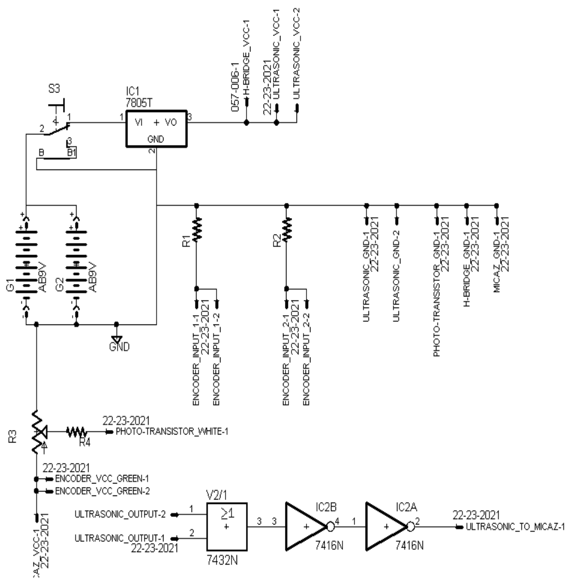


Figure 13: PCB Board Design Schematic

This schematic combines all the necessary PCB components together creating common grounds and voltage sources as well as providing pin outs for the MicaZ microcontroller, H-Bridge motor control, 9V battery external battery sources, orientation phototransistor apparatus, the wheel encoders, and the ultrasonic sensors. The components are as follows: 9V external battery sources (two battery symbols located at the center of the far left-hand side of the schematic), toggle switch (upper left-hand corner), 5V voltage regulator (top center), 250KΩ multi-turn potentiometer (lower left-hand side) and varying resistor (directly right of potentiometer), encoder current-regulating resistors (connected in parallel to the direct right of the 9V battery symbols), OR gate (square symbol at the bottom of the schematic) used to combine the readings for the two ultrasonic sensors, and the NOT gates (two bubble-ended triangular symbols located on the lower right-hand side of the schematic directly right of the OR gate symbol) used to regulate ultrasonic data voltage level to 3V.

This PCB was used to simplify wiring for the mobile node. Several external components were connected as well as nearly 20 blank pins for connection to various hardware components. The toggle switch, located at the upper left-hand corner of the above schematic, was added to make the process of powering the mobile node motors simpler as well as to conserve battery power for the system. This switch was run directly off of the 9V external battery supplies and connected to the 5V voltage regulator (top center of schematic), and therefore, controlling when the system would be operational. The voltage regulator was used to down-convert the 9V battery sources to the 5V necessary to run several hardware components, such as the H-bridge motor control and the ultrasonic sensors, which were given pin connectors run directly off of the regulator output voltage (upper right-hand corner of schematic).

The center pin of the regulator was connected to the grounding line of all the major components and hardware. Two current-regulating resistors (connected in parallel directly below and to the right of the voltage regulator) were added to limit the amount of current passing through the wheel encoders, which were given output pins on the opposite end of the regulating resistors. Directly below the 9V battery symbols and connected between the common ground and the 3V MicaZ voltage source (a pin out located at the bottom left-hand corner of the schematic) is a 250KΩ multi-turn potentiometer connected in series with a constant resistor and the phototransistor input. This potentiometer was used to adjust the sensitivity of the phototransistor to infrared light. The desired effect was to adjust the sensitivity so that the ambient light in the room would not activate the phototransistor and ruin the orientation process.

In order to utilize the ultrasonic sensors, an input capture pin on the MicaZ needed to be connected to the output of the sensors. In order to simplify programming and wiring, it was decided to only use one input capture pin for both ultrasonic sensors. The sensors were programmed to fire one at a time and the outputs of both sensors were tied together through an OR gate and two NOT gates (located at the lower right hand side of the schematic) powered by the 3V MicaZ voltage source. The OR gate was necessary to combine the outputs of the sensors into one output which could be read by the input capture pin on the MicaZ. The two NOT gates following the OR gate were used to regulate the 5V output voltage from the ultrasonic sensors down to 3V so as not to overload the MicaZ.

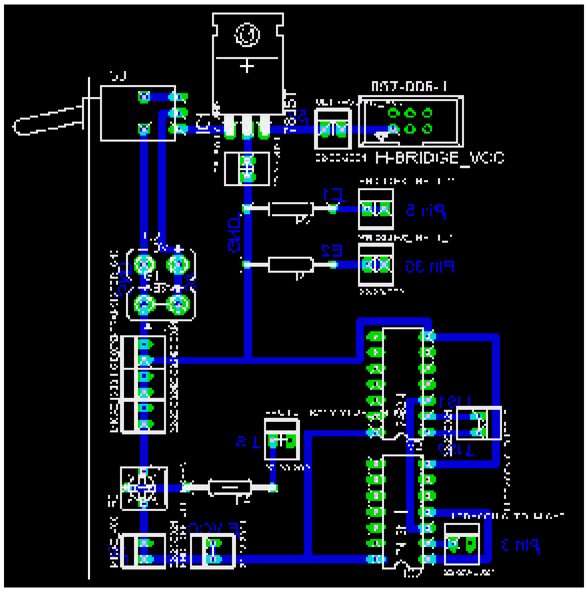


Figure 14: Component Layout on Virtual PCB

This image is the virtual PCB board design with component image overlay. The layout consists of the toggle switch being located in the upper left-hand corner (see switch-like image), the voltage regulator connected directly right of the switch, three pin headers for 5V power supply connections to the H-bridge motor control and ultrasonic sensors. Directly below the 5V voltage regulator (top center) are two grounding pins for the ultrasonic sensors. Directly below those pins are the current-regulating resistors connected in parallel from the grounding line to their respective encoder input and output pins (right of resistor images). The 9V external battery source pins are located beneath the toggle switch pins with 9V output going up to the switch as well as a ground going to the switch and the three grounding pins below for the MicaZ, H-bridge, and phototransistor. Directly below the grounding pins is the 250KΩ multi-turn potentiometer, which is connected between the 3V MicaZ source pins (bottom left-hand corner) and the constant resistor (directly right of potentiometer), which is, in turn, connected to the phototransistor input pin (right and up from constant resistor). The MicaZ 3V source pins are to the left of the two encoder power source pins and further connect to the two logic gate chips. The ground from the 9V battery sources connect to the logic chips as well (lower right-hand side of schematic). To the immediate right of the logic OR gate (top chip image in the lower right-hand corner) are the output pins from the ultrasonic sensors. The OR gate output then connects to the NOT gate which inverts the OR gate output twice and then finally outputs to the MicaZ input capture pin located directly to the right of the NOT gate chip (bottom chip in the lower right-hand corner of schematic).

Once the schematic for the PCB was designed, the components had to be arranged onto a virtual PCB board and connected by solder lines. The previous PCB design (Figure 14) shows all of the components laid out on the virtual PCB and Figure 15 has the component images removed to show only the solder lines and the drill pads.

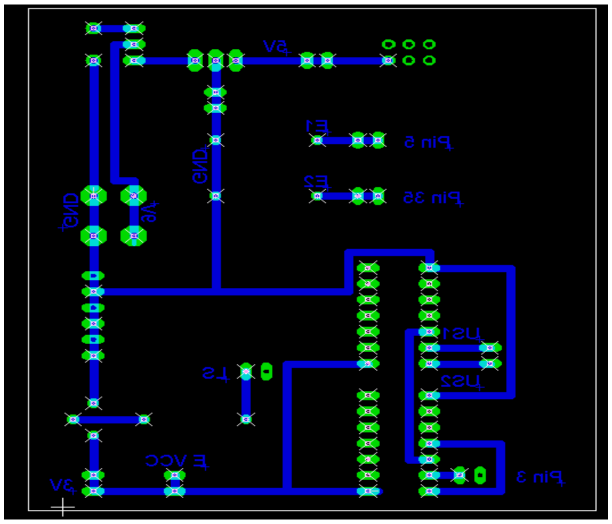


Figure 15: Virtual PCB Schematic Without Component Overlay

This image allows for easier viewing of the solder lines for the PCB. Certain pins and components are labeled for easy wiring and reference points.

The design was finalized and printed onto special blue paper, which was adhered to the copper surface of the PCB. The PCB was then immersed in ferric-chloride solution to etch away the unnecessary copper surface around the solder lines. The solder lines were then tined and holes were drilled into the pads to allow for component placement. The components were then added and soldered into place, hence completing the circuit board. Holes were then drilled into the four corners of the circuit board and it was attached to the inside of the back wall on the mobile node and wired to the hardware components.

The Laser Level

To our orientation function, we needed something to orient to. Our choice was the 90 degree laser level made by Chinglin Industries. It is a class III laser product, with a 5 mw max output and 630 – 650 nm wavelength. Two of these laser levels gave us the power to cover the entire searching area effectively.

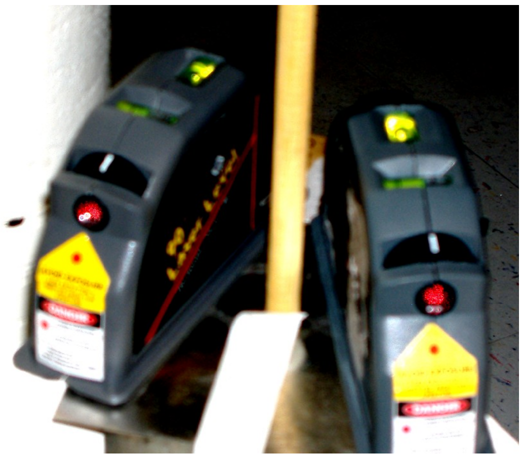
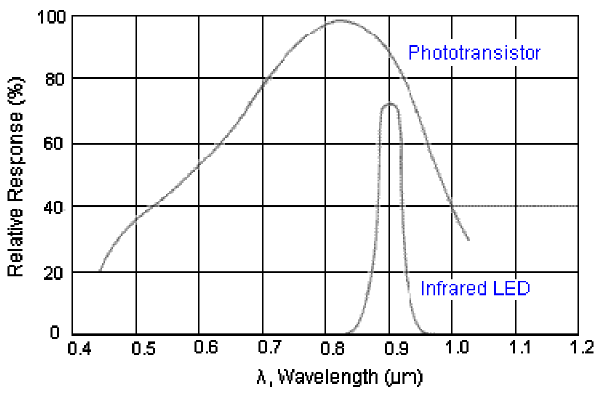


Figure 16: Laser Level Apparatus

The Cannon

To use our ultrasonic sensors to localize, we needed a reference point to start with, that so we could figure out our nodes' orientation. We decided to solve this problem by setting a laser beacon in one of the corners, and using it as a reference point. The analog to digital converter in the MicaZ was used between the resistor and the photo transistor. The voltage measured at that point is affected by ambient light, but it can be



compensated for by changing the potentiometer value. We adjusted ours to work with the lighting in the Digital Lab.

Figure 17: Transistor Response Graph
Our laser transmitted at a wavelength of 0.65 micrometers so our sensor has a relative response of about 60 %. Image courtesy of http://www.oz.net/~coilgun/theory/phototransistors.htm

One thing we had to compensate for was the level of the floor and the size of our photo transistor, which had a viewable area of only 20 mm by 20 mm. This made it very challenging to hit the transistor with our laser beam. Therefore, we designed an optical assembly that was placed in front of the photo transistor that focused the light on the photo transistor. This increased the viewing area from 20 mm to 2 cm.

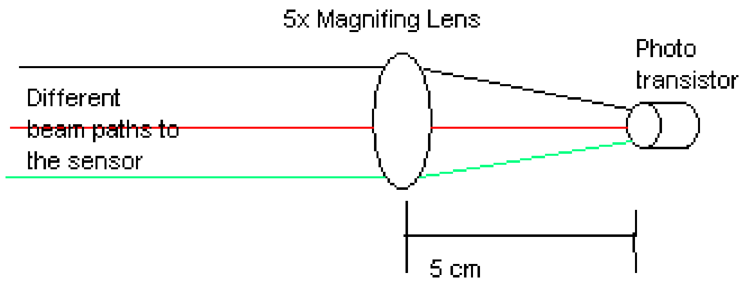


Figure 18: Lens Functionality

The above lens configuration is mounted inside an aluminum tube. We used electrical tape to seal every gap except for the lens area, such that we would reduce false alarm due to the ambient light in the room. Also, the laser beam is focused into a spot on the sensor so that more of the laser light strikes the sensor, reducing the amount of false alarm on our sensor further.

The entire assembly was about 10 cm long and 3 cm tall and 3 cm wide. It was mounted about 35 cm from the ground on a rear post. We fondly called this the “cannon”.

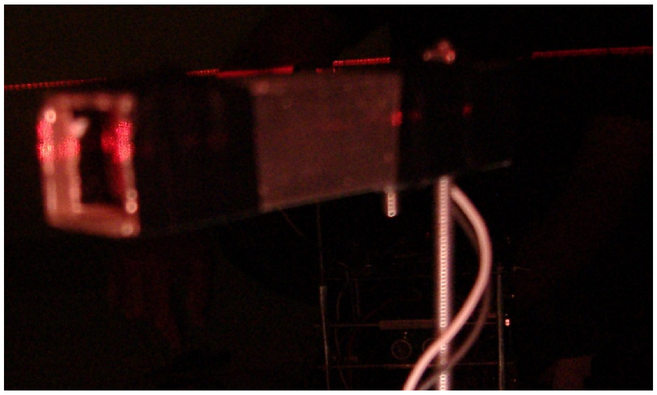


Figure 19: The Cannon in Action

We chose the laser levels over an infrared Light Emitting Diode (LED), because the intensity of the laser did not drop the voltage as fast as the infrared LED. Therefore, we can detect the laser at a much greater distance than the infrared LED.

The Overall Operation

The final design worked very well. It reduced our wiring errors greatly, and gave us a better node to do our testing on. When we had errors in the operation of our node, it was more convenient to check the hardware for errors. All of the group members approved of this design, and it worked well for us.

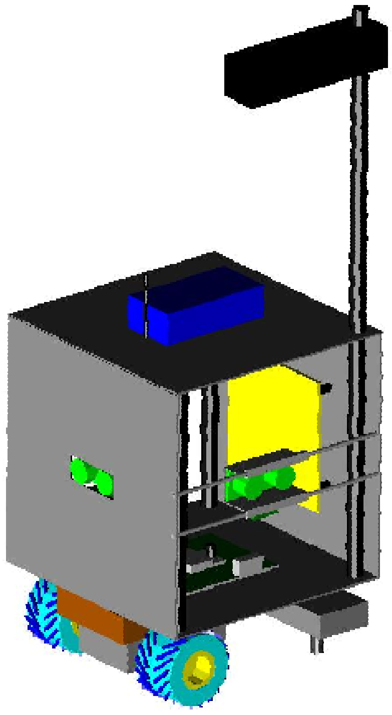


Figure 20: Final AutoCAD Drawing.

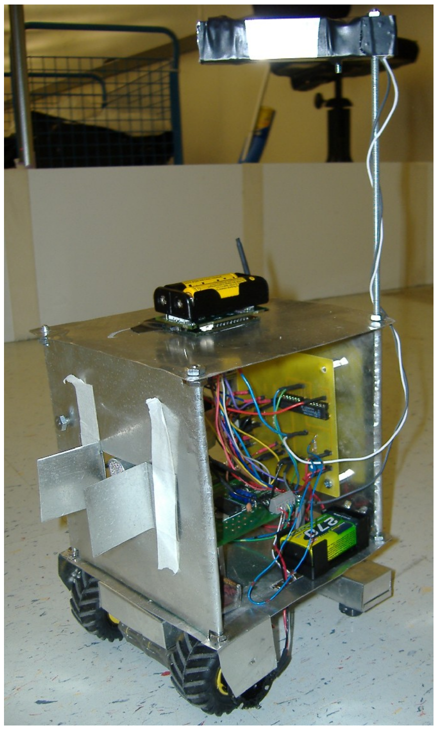


Figure 21: Final Design of Node.

Software

The hardware that we designed is useless without good programming to run the nodes. This section was by far that most difficult for the group, since we were using a programming language and operating system that we've never seen before. This is the design that we created and implemented.

Orientation

Orientation is the method by which our nodes will orient themselves with a fixed reference point. The reasoning for the "cannon" is that it gives our robots a fixed reference point to orient from. The actual algorithm for determining this orientation is discussed in the following paragraphs.

Once the nodes are placed in the searching area, the start command is sent to the nodes. The nodes start by rotating clockwise, checking the analog to digital converter's voltage lever. Once the photo transistor detects the laser the voltage drops below 1.5 V. Immediately following, the node stops rotating and starts turning counterclockwise in small iterations. After each iteration, it stops and fires the ultrasonic sensors, giving the distance to the walls. The values are compared until they increase again, and the node repeats these steps in the other direction at smaller intervals until it finds the shortest distance to the walls. This will give the orientation of the nodes.

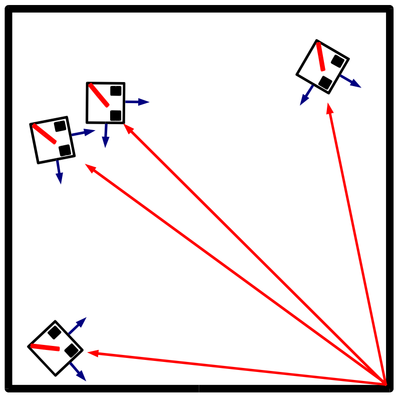


Figure 22: Orientation Method.
The node rotates until the laser is detected. The laser serves as a fixed reference point.

Localization

Localization is the method by which an object determines its location within a given area respective to available parameters. Originally, two methods of node localization were considered for the project: Radio Frequency (RF) Wave Triangulation and Reflective Ultrasonic Wave Triangulation. After considerable testing that yielded inconsistent data, the RF Wave Triangulation method was abandoned.

Due to time constraints and the desire to reduce the design complications, the decision was made to pass over the Reflective Ultrasonic Wave Triangulation method and adopt the simpler Reflective Ultrasonic Range Finding method. This allowed the same ultrasonic sensors used to implement the searching algorithm to be concurrently used for node localization.

Radio Frequency Wave Triangulation

Radio Frequency (RF) Wave Triangulation utilizes a range of electromagnetic waves known as radio waves to transmit data wirelessly. For this project, the localization method would consist of three MicaZ radio transmitters located around the perimeters of the search area and two mobile nodes in the field acting as radio receivers. Two of the

MicaZ transmitters would act as nothing more then radio signal transmitters which, when given the command by the base station, would transmit a radio signal to be received by the mobile nodes in the field. Prior to the beacon transmission, the base station would inform the mobile nodes of the need to localize. This information would also contain a time from which to mobile nodes should begin a timer to measure the time difference between the beacon transmission and the node receipt of that transmission.

The mobile node would then transmit this time difference back to the base station, which would use that information to calculate the node's distance from the fired beacon. This would be done three times, with the base station acting as the last beacon. After gathering this data, the base station would be able to accurately place the icon representing a particular mobile node onto the GUI as well as transmit this information to the respective mobile node.

This method of localization is very difficult to achieve in small-enclosed areas. Radio waves travel at the speed of light (approx. $3 * 10^8$ m/s) and are therefore prone to refract off of nearby surfaces. This refraction can yield incorrect time difference values hence decreasing the overall integrity of the system. For a period of three weeks tests were run between a transmitting beacon and a receiver. Unfortunately, the resolution of this method was approximately 20 – 30 cm, which was greater than the 10 cm resolution we were expected to achieve.

The lack of consistent and accurate data forced the group to redirect localization focus onto the ultrasonic sensors that were being used to implement the searching algorithm. The alternate option of Reflective Ultrasonic Wave Triangulation was abandoned due to cost, time restraints, and complexity of the system in favor of the less complex and inexpensive method of Reflective Ultrasonic Range Finding.

Reflective Ultrasonic Range Finding

Reflective Ultrasonic Range Finding is a method of localization, which utilizes ultrasonic sound waves to determine the distance from the reflective ultrasonic sensor to an object. The sensor fires an ultrasonic wave, which reflects off of nearby objects and returns to the sensor. Visually, an object, which the sensor detects, is seen as either a rise or a fall in the voltage level of the receiving pin. The ultrasonic sensors used for this project had a maximum range of 3 m and a minimum range of 3 cm.

The MicaZ microcontroller used a timer to measure the time difference between the firing of the ultrasonic wave by the sensor and the return of the first wave to the receiver. The time difference measured by the MicaZ timer is the amount of time the ultrasonic wave takes to travel round-trip from the sensor to the object and back. For calculation purposes, the one-way trip time difference is needed; therefore the measured time is scaled by ½. This information is then transmitted to the base station, which calculates the distance from the mobile node to the detected object.

The localization algorithm implemented for this project was chosen due to the ability to easily integrate the algorithm with the desired search algorithm and the cost effectiveness of the dual functionality the ultrasonic sensors would serve. The most important aspect of localization is reference points. In order to implement a successful localization algorithm at least one unvarying reference point must be present. This allows the system to determine locality relative to a stable variable within the environment.

In order to most effectively integrate the desired coordinated searching algorithm with the reflective ultrasonic range finding localization, a basic X-Y axis for the search area needed to be established. As mentioned above, the orientation algorithm that was implemented allowed the mobile nodes to easily move to their respective corners. Once the nodes were properly oriented the localization would commence. The ultrasonic sensors would measure the time differences to the two closest walls (as seen in Figure 1) and transmit that information wirelessly back to the base station.

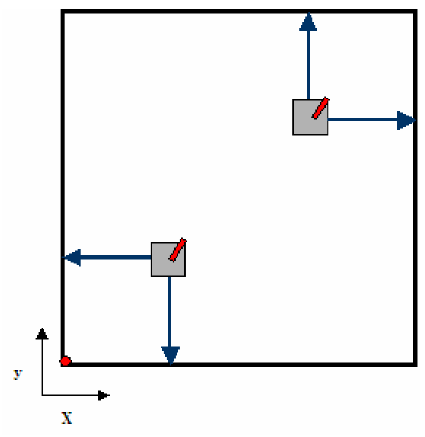


Figure 23: Post-Orientation Distance Measurement
After orienting to the laser in the lower right-hand corner via the cannon (marked in red) the nodes (gray boxes) take distance measurements (blue arrows) to the orientation walls.

The base station would then calculate the distance to the detected object and the number of encoder clicks the wheels needed to turn to place the mobile node within 25 cm of that object:

$$clicks = \left[\left(\frac{time * 343}{576}\right) - 25\right] * \left(\frac{15}{16}\right) \qquad (1),$$

where clicks is the number of encoder clicks needed to reach within 25 cm of the detected object, time is the one-way trip time difference, 343 is the speed of sound in meters per second, 576 is the speed of the MicaZ clock scaled down by a factor of 100 (the number of centimeters in a meter), the 25 represents the 25 cm distance that the node is to remain away from the wall, 15 is the number of encoder clicks per revolution of the wheel, and 16 is the circumference of the wheel in centimeters.

This information is transmitted back to the mobile node, which then moves either forward or backward along the y-axis the designated number of encoder clicks. The node is then turned 90º to face the closest y-axis wall (as in Figure 2) and moves either forward or backward along the x-axis the desired number of encoder clicks.

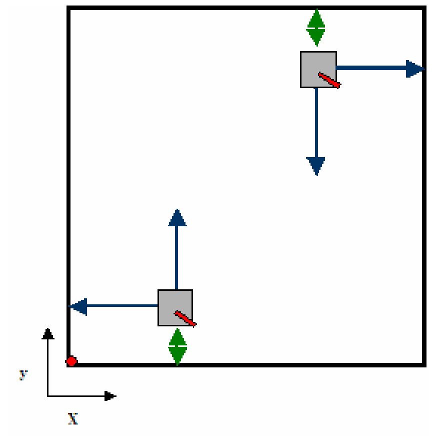


Figure 24: Y-Axis Movement of the Mobile Nodes
The nodes move either up or down along the y-axis to within 25 cm (green arrows) of the x-orientation wall.

Once the node has moved the desired number of clicks in the x- and y-axes, the node re-enters the orient phase. This is used to ensure that the node is within the desired 25 cm distance from its respective orientation wall. These re-orient and move phases will continue until the node has positioned itself within the desired range. Upon proper distancing, the node will rotate 180º to position itself to begin the coordinated search (see Figure 3).

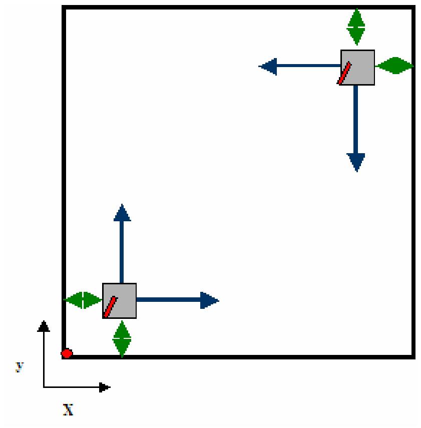


Figure 25: Start of Coordinated Search Position
Upon moving to within 25 cm of both orientation walls, the node performs a 180º turn and readies for the commencement of the searching algorithm. One ultrasonic faces into the playing field while its partner is positioned at the front of the node facing the far wall toward which the node is migrating.

In the case where the black box is positioned between a mobile node and its orientation wall (see Figure 4), the movement in both the x- and y-axes directions will prevent the node from orienting to the box (see Figure 5).

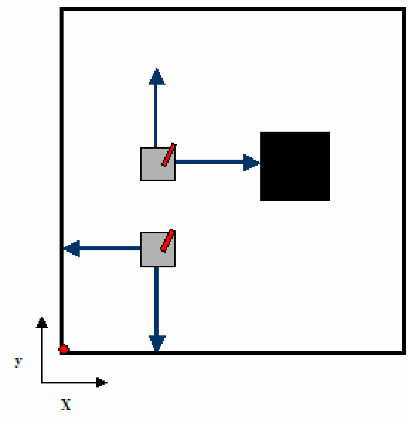


Figure 26: Worst-Case Orientation Scenario
Should the situation arise that the mobile node, once oriented to the laser, first sees the black box as one of its orienting walls, there are no long term effects on the localization or search algorithm.

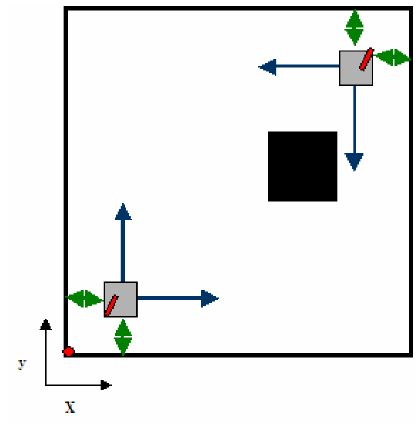


Figure 27: Start of Coordinated Search in Worse-Case Orientation Scenario
Once the node moves both directions, the black box will no longer act as an orienting wall. The node will instead continue re-orienting and moving to the correct orientation walls.

Localization during the coordinated search is accomplished using two methods: ultrasonic distance readings and number of encoder clicks traveled. As previously mentioned, the reflective ultrasonic sensors serve to locate the black box and to localize the node. During the search algorithm, the mobile nodes move along a wall in either the x or y direction. One ultrasonic sensor is positioned to fire into the search area while its partner is positioned at the front of the node facing the far wall toward which the node is migrating. The front sensor will be fired to determine the distance forward the node has moved.

The searching field sensor reading will serve to not only detect anomalies in the field, but also to give the amount of drift the node is experiencing during movement. The side-to-side drift of the node is not substantial enough to cause the base station to raise a false alarm for finding the box. Therefore, a slight drift along the x- or y-axis as the node moves would be detected by the base station, and allow accurate positioning of the icon on the GUI screen.

There are pros and cons for every localization algorithm that could be implemented for this project. For this particular algorithm pros include dual use of ultrasonic sensors for the coordinated search and node localization, simple linear distance calculations and integration with desired searching algorithm, and the assurance that the black box can only be in the scanned search area and not between the node and its guiding wall.

Cons include the necessity of re-orienting the node whenever a rotation is made and the difficultly of compensating for x- or y-axis drift due to unsynchronized motor control. Another unfortunate occurrence, which caused some problems with node movement, was the tendency for the ultrasonic to return very large inaccurate time differences to the base station resulting in very large distance readings which would cause the node to attempt move a position beyond the constraints of the search arena. This dilemma could be fixed by implementing an if statement into the programming in the base station, which would disregard distance values that exceeded the bounds of the search area.

Search Algorithm

Since we were already using the ultrasonics to localize, we decided to use them for our search algorithm as well. Our localization depended on the nodes being perpendicular to the walls, so we designed our search algorithm in order to take advantage of this.

The Plan

Once the nodes are oriented they should be at opposite corners of the box. They will then do a 180-degree turn so that the front sensors of both nodes are pointing to the same corner and that side sensors of both nodes are also pointing to the same corner. Both nodes will then move forward. As they move forward, they will stop about every

15 cm and scan the area with the side ultrasonic. The nodes will send the information back to the base station, which will perform calculations to determine whether the box is seen or not.

Distinguishing Between the Wall and the Box

If the orientation goes as expected, both nodes will be 25cm from one wall and 25 cm from the other after they do the 180-degree turn (See Figure #). When the nodes scan with the side ultrasonic as they move forward, they send a value that translates to a distance of about 250-270cm from the wall (See Figure #). This distance assumes that the error of the ultrasonics is less than 10cm. During testing we did not get any consistent values of error that were greater than 10cm.

If the ultrasonics read a value that's less than what the base station knows it should receive, there is a possibility that something (in this case, the box) is obstructing the view of the ultrasonics.

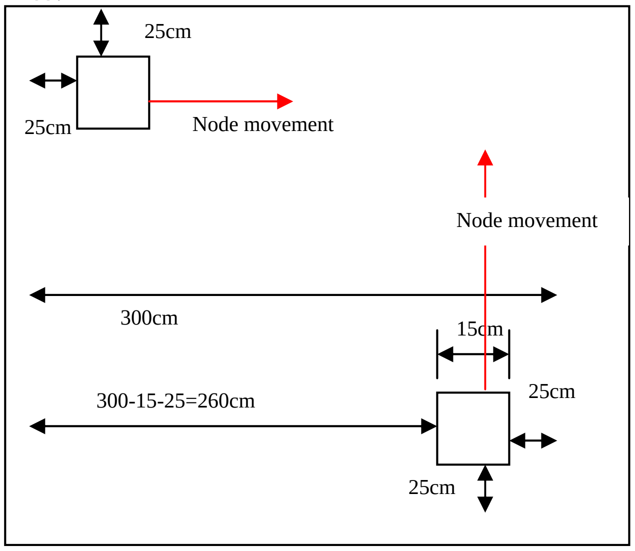


Figure 28: Initial Search Position

Verification

When the ultrasonics read a value less than expected, the base-station will then tell the nodes to move 5 cm (instead of 15 cm) forward and re-scan. The second scan is to make sure that the readings received were not errors. If the ultrasonics did see the box, there is no way that these readings will be much different since the box is 30 cm long. If those two readings are close to the same, then the node will be asked to keep moving forward in 5 cm increments and re-scan.

When the measurements read by the ultrasonic sensor revert to the expected value in between 250 and 270 cm, the box will have been passed. The 5 cm increments are convenient because they give the base-station a better resolution of where the box ends. When the end position of the box is determined, the base station will send this information to the GUI and the GUI will then draw the box location on the screen.

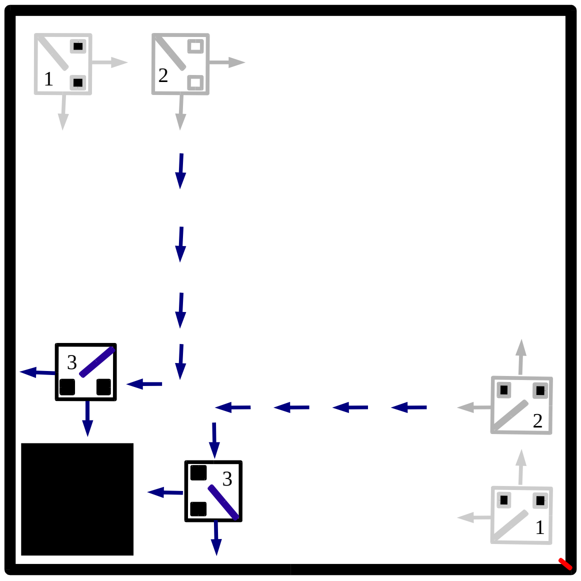


Figure 29: Verification of the Box

Once one node has found the box, the base-station will determine the position of the box. It will then transmit the distances and the directions that the other node has to travel in order to move toward the box. Immediately following, the other node will approach the box. When the other node reaches the box, the mission will be complete.

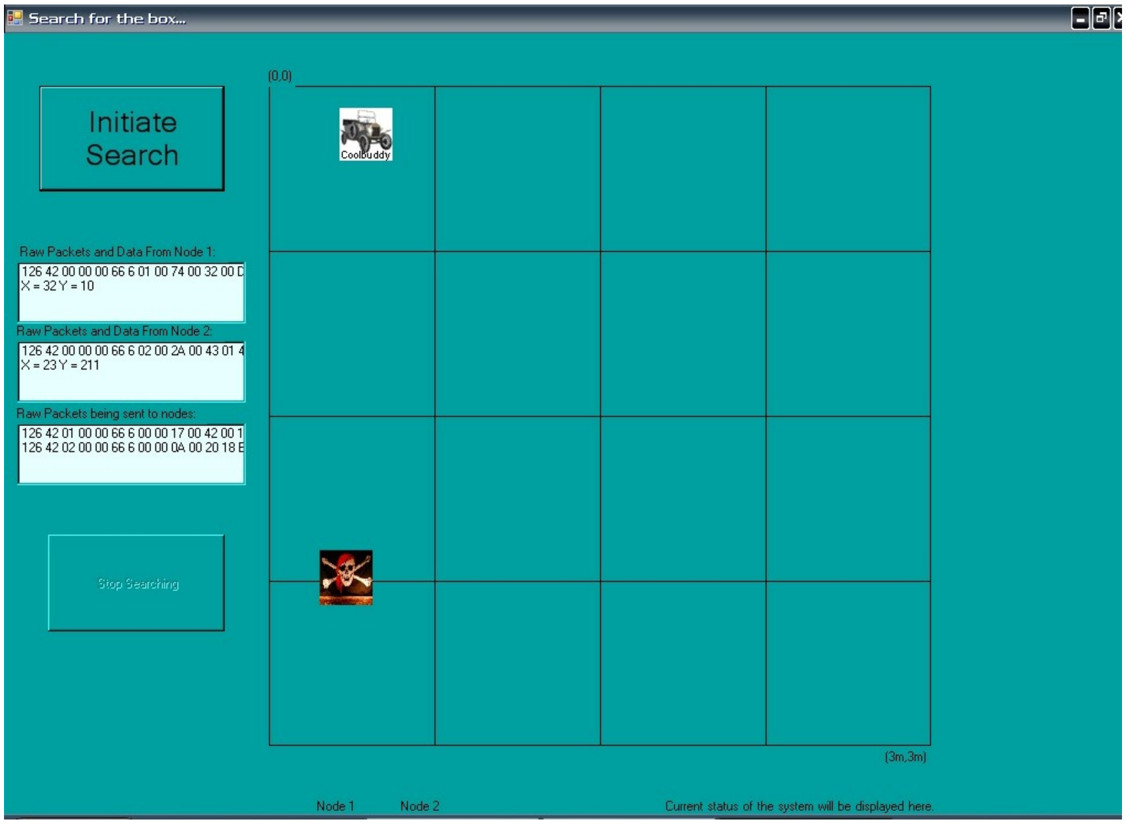
Pros & Cons

Our search algorithm is very simple to implement. The base-station does all the calculations necessary and gives the nodes directions on what to do. The directions are very straightforward and include move forward and scan. Money was saved due to the fact that the ultrasonic sensors were used for localization as well as for searching.

A disadvantage of the search algorithm is that it can only be used when the exact dimensions of the search area are known. Also, due to the fact that the ultrasonic sensors have a limited range, the searching area can not be larger than a certain value in order for the search algorithm to work.

Graphical User Interface

The goal of the GUI is to display the position of the mobile nodes with a refresh rate of no more then 2 seconds. Once the nodes have located the black box it’s location must be displayed. To accomplish this task, our group chose to write the GUI in Visual C# 2005 using the .Net version 2.0 framework. This language was chosen for ease and quickness of implementation. Visual C# provides a simple solution to a managed application, while the .Net framework made serial port communication much easier to implement. The GUI consisted of a standard form layout with two button controls, three list box controls, and a custom user interface control.



Raw Packets and Data From Node 1:

```
126 42 00 00 00 66 6 01 00 74 00 32 00 D
X = 32 Y = 10
```

Raw Packets and Data From Node 2:

```
126 42 00 00 00 66 6 02 00 2A 00 43 01 4
X = 23 Y = 211
```

Raw Packets being sent to nodes:

```
126 42 01 00 00 66 6 00 00 17 00 42 00 1
126 42 02 00 00 66 6 00 00 0A 00 20 18 E
```

Figure 30: The GUI Display with the position of both nodes shown.

The GUI consists of two main classes. The main form class (Form1.cs) and the packet handling class (PacketHandler.cs). The main form class is responsible for all of the user interface objects and all corresponding tasks, such as the button clicks, displaying data, and serial port I/O. This class is also extended by the custom user interface control (GridDisplay.cs). The custom control creates a scaled model of the searching area and provides an interface for the other classes to give it data to display the nodes positions, and eventually the position of the black box.

Form1.cs

This file is declares the main form class that handles the initialization and construction of the main form/user interface of the program. It defines all the user interface controls and any properties associated with them, for example, the definition of the start button. The size, location, color, text, and scope of visibility of all the user interface controls are set during the forms initialization.

Serial port functionality is also defined in this class. A new serial port is created and its attributes set. The serial port class functions by using events to signal the program when data is received or other such events occur. This requires that a function be defined to handle the events that are generated during the use of the serial port. In this case, the only event that we care about is the data received event, so a function was written to be used as a serial data received event handler. The only difficulty in dealing with the serial port is the class which deals with it functions on its own thread. Therefore, when events occur the functions to handle them cannot simply be called, because they would do the work on the wrong thread and the results would not be seen. To deal with this in C#, the function must be invoked.

GridDisplay.cs

This is the custom user interface control that extends the main form class to implement the display of the searching area. This custom control contains the code to generate the grid that is displayed to represent the field as well as the initialization code for picture box controls. These were used to represent the nodes and the box. By making these components of the class public, they are easily accessed by other classes and the position of them can be changed as the actual nodes position changes.

PacketHandler.cs

This class was constructed to disassemble the incoming packets from the serial port and handle them accordingly. An instance of this class is created upon the main forms initialization. Contained in this function is one public function, PacketSeparator, which is called when the serial port data received event is triggered. In this function, the raw data from the packet is added to a corresponding list box on the main form.

Motor Control

The most basic requirement of a mobile node network is the ability to control and operate the motors of that network. At the onset of the project the decision was made to use generic digital signals as inputs to the mobile node motor systems. The application of these signals was allocated to a MicaZ timer. The gear boxes were built to allow only the lowest speed setting the motors could output. The simple digital signals would supply a constant voltage to the motor causing the node to move at its full-speed setting. For initial tests, which required the node to move a limited distance in either direction and turn a desired degree, the generic digital signals worked well. Once localization testing began, the motor control choice began to prove problematic. The motors would consume a vast amount of power quickly, causing the movement programming to become inconsistent after a short period of time. After careful consideration, the decision to add encoder sensors to the wheels was made, in the hopes that the encoders would better regulate the movement of the wheels. Unfortunately, the encoders did not satisfactorily solve the drifting and inaccuracy problems. The continued to experience relatively high levels of motor power consumption and the ability to turn was unreliable at best. This raised further concern about the choice of motor control, being that the localization, orientation, and search algorithms required highly accurate turning and movement of the node. Finally, after much debate and troubleshooting, it was decided that the motor control must be done using pulse width modulation (PWM) methods.

Initial Encoder Usage

The wheel encoder sensors were added in an attempt to force the rotation of the mobile node wheels to become independent of the battery supply power. The encoders were attached to both front wheels of the node and connected to continuously poled analog-to-digital (A/D) converter ports on the MicaZ. The A/D converter was used to record the number of encoder clicks a particular wheel had turned. The values from the converters were then compared and the program would regulate the rotation of the wheels so that the encoder clicks of both were equal. For instance, should one wheel stop after it has completed the designated number of encoder clicks, the other wheel would continue to rotate until it also has reached the desired number of toggles. The comparisons of these values were accomplished on a continuous basis so as not to interfere with the input capture events being triggered by the ultrasonic sensors. A further precaution was taken to ensure that the ultrasonic sensors and encoders were not capable of competing for computation time. This precaution was accomplished in the form of programming the node to prevent any action besides movement inside the Drive function.

Direction Control

The mobile node network required free movement capabilities within the search area. This is achieved through the implementation of directional motor control. For this project, the code (seen in Motor_ControlM_old.nc) for all directions was programmed in essentially the same fashion. The variables were initialized, registers set, and the motors were supplied with the necessary power for movement. A continuous loop would be executed until the wheels had completed the designated number of encoder toggles. Within this loop, the program would continually check the turning status of the left wheel, via the encoder, and increment the stored 'clicks' value within the register. The same status check was then performed using the right wheel encoder. These incremented values would then be compared to the desired values. Once an encoder had reached the desired 'clicks' value, the power supply to that wheel would be cut off and the program would wait for the remaining wheel to complete its rotation.

During the initial test phases of the project, this code presented no problems. The node would drive short distances and turn relatively accurately unless the battery supply was low. For a time, the drifting experienced due to inconsistent powering of the motors, was acceptable, due to the fact that the new encoders would force the node to face in the original direction despite the non-linear characteristics experienced during driving.

Drifting Errors

During travel, the mobile nodes would experience non-linear movement characteristics due to inconsistent powering of the motors. The node would drift excessively during long-term movement causing the ultrasonic resolutions to be outside the desired 10 cm range. The search algorithm, which required highly accurate readings, would read these discrepancies as the black box the nodes were attempting to locate. The drifting tendencies drastically increased the probability of false detection, therefore it was decided, rather late in the project, to implement PWM motor control instead of the generic digital signals used previously.

Use of Pulse Width Modulation

In order to regulate the supply power to the motors and hence force linear travel characteristics from the mobile node motors, Pulse Width Modulation (PWM) was adopted for motor control. With the wheel encoders operational and regulating the rotation of the wheels and the PWM controlling the motor speed, each motor could implement control theory for drift tendency corrections. Unfortunately, while the PWM was implemented the encoder-based motor control program was also active, causing the node to no longer straighten out after traveling a long distance. Instead, one wheel was being shut off, while the other was left running. This problem required many hours of troubleshooting and modifications to the code.

Troubleshooting

The 'spinning wheel' behavior exhibited by the motor system after the implementation of PWM was occasionally caused by incorrect wiring of the encoder lines but, more often than not, the problem was on the program side.

Initial troubleshooting was aimed at determining if the encoders were toggling at a rate faster than the MicaZ clock could read. The speed of the motors was reduced, but, unfortunately, this had no effect on the motor behavior. The conclusion was reached that the motor speed was not the issue because the 2.4 GHz processor speed of the MicaZ caused the wheel encoder values to be checked once every 5 μs. The motors would have to move at 1000 m/s to miss this routine check. Another option for the error was the possibility that the MicaZ was receiving interrupts from another component during the motor control function. An interrupt requires a large amount of CPU time and hold precedence over all other actions for the microcontroller. It was deduced that the only possible source of an external interrupt was the RF radio transmitter and receiver on the MicaZ. Our code was sending RF messages sparingly, but signals from other groups could have been interfering with our system. Upon receiving a signal or packet the MicaZ CPU forces all register data onto the stack, and executes the information within the interrupting packet. Not all packets apply to all microcontrollers but, as the microcontroller must receive the packet then analyze it for group number and destination information, an interrupt could occur due to stray packets. Despite efforts, the source of the encoder read error was never determined though the 'spinning wheel' behavior was solved using the encoders.

Final usage of Encoders and PWM

In a final attempt to accurately implement the encoder and PWM motor controls, the program was altered to allow reading from only one encoder to judge distance traveled or turned by the node. The drifting tendencies were still present although the PWM did help to reduce the effects to acceptable resolution ranges. This alternate code forced both wheels to stop at the same time, ceasing the 'spinning wheel' behavior.

Despite the original benefits of the 'same distance' encoder programming, this new program worked better with long distances and did not usually require an end-of-movement facing adjustment. The PWM was used to accurately adjust the motor speeds concurrently helping the node to travel in a more or less straight line.

The drift tendencies of the nodes seemed to be caused by two separate things. The first type of drift was constant and likely due to the physical properties of each node, such as off center weight and the distance of the wheels from the gearbox. This drift was correctable with the constant change in the motor speeds. The second type came in bursts which were likely due to slipping wheels. Corrections were made for wheel slippage, such as adding weight to the node and cleaning the floor. With these efforts, the node seemed to go straighter, though there will still small amounts of drift.

In the end PWM and encoders were used for motor control though not to implement feedback. While feedback to cause the node to go straight would have proved beneficial, without the ability to accurately read from both encoders it would have been impossible. The final implementation of the motor control, as seen in the code, was certainly an improvement over the original test-drive motor control code. Therefore, the motor control a success despite the inability to fulfill expectations.

Final Budget

Quantity	Item	Description	Cost	Total
1	SPDT	Toggle Switch	$0.50	$0.50
2	LM7805	5V Regulator	$0.50	$1.00
2	WM2903-ND	C-Grid Connector	$0.50	$1.00
2	WM4903-ND	C-Grid PCB Header	$1.00	$2.00
46	WM2510-ND	C-Grid Crimp Pin	$0.08	$3.68
3	CRMT254	250K Precision Pot	$1.00	$3.00
1	SPST	Switch	$0.50	$0.50
1	1x40	Pin header	$1.00	$1.00
40	A3003-ND	Crimp Pin	$0.15	$6.00
1	Sink	Heat sink	$0.50	$0.50
2	9V	9V Battery	$2.00	$4.00
2	Heat shrink sm	Small heat shrink tubing	$0.25	$0.50
4		Encoders	$6.34	$25.36
2	SR04	Ultrasonic Sensors	$24.62	$49.24
2		Set of 2 Ball Casters	$10.00	$20.00
4	9V	9V Rechargeable Battery	$8.95	$35.80
6	9V	9V Battery	$1.86	$11.16
2		Photo Transistors	$4.00	$8.00
2		Laser Levels	$10.00	$20.00
2		Magnifying Glass	$4.12	$8.24
1		Sheet of 4 ' x 4 ' Aluminum	$4.00	$4.00
		Screws, bolts, etc.		$5.00
		Tin		$10.00
Grand Total				**$220.48**
Total Money Left				**$29.52**
Actual Money Left				**$107.44**

Table 1: Final Budget

Note: The ***Actual Money Left*** does not include the following:
2 encoders
1 set of 2 ball casters
2 Photo Transistors
2 Laser Levels
2 Magnifying glass
Sheet of 4’ x 4’ Aluminum
Tin
Screws, bolts, etc.

These items are not included because they were either donated or already owned by group members.

Power Budget

2 motors	- 14.60	W
2 ultrasonic sensors	- .30	W
H-Bridge	- .18	W
MicaZ	- .156	W
TOTAL POWER	- 15.34	W

Table 2: Power Budget

2 AA batteries give 2 mA-hr
2 9V Rechargeable NiCd give 270 mA-hr

Using current, power, voltage and time equations if the node is run continuously:
Total life of batteries ≈ 30 min.

Conclusion

The conclusion will state how far we progressed with this project, what didn't get completed, and what we learned from this project.

What Works

We were able to get only one node working, but on this node orientation and localization were working just like we wanted them to. The GUI displayed the location of the nodes with reference to the laser level placed in the corner. The motor control and PWM work. The robot is light, yet robust.

What Needs Work

On the day the project was due, the communication system was having errors with packets. We tried to figure out why this was happening, and we concluded that it was interference coming from one of the other groups.

Because of this, we did not get to test our searching algorithm. If we would have had about 2 more weeks, we would have had our sensor network searching and finding the black box.

What We Learned

A semester is not as much time as we thought it would be. It took the group a long time to learn and become efficient at nesC, which took away time from testing and debugging. We spent far too long troubleshooting errors that we thought were software, yet discovered it was the hardware or vice versa. TinyOS did not give the group much of an opportunity to effectively debug our nodes.

We learned the key to project success is working well both individually and in a group. The group was extremely close to having a working sensor network, and we would have loved to see it in action. The group worked well together overall, and we made good progress throughout the semester. This course did a good job in preparing us for Senior Design and projects that we will encounter for the rest of our lives.

References

[1] G. Benet, F. Blanes, J.E. Simó, P. Pérez. (2002, March 27). Using infrared sensors for distance measurement in mobile robots. [Online]. Available: http://www.ingentaconnect.com/content/els/09218890/2002/00000040/00000004/art00271

[2] J. Gaspar, G. Lacey, J.S. Victor, N. Winters. (Date?). Omni-directional Vision for Robot Navigation. [Online]. Available: http://www.lkl.ac.uk/niall/00-Omnidir.pdf

[3] M.O. Franz, B. Scholkopf, H.A. Mallot, H.H. Bulthoff. (Date?). Learning View Graphs for Robot Navigation. [Online]. Available: http://www.kluweronline.com/article.asp%3FPIPS%3D154954%26PDF%3D1

[4] A. Ohya, A. Kosaka A. Kak. (Date?). Vision-Based Navigation of Mobile Robot with Obstacle Avoidance by Single Camera Vision and Ultrasonic Sensing. [Online]. Available: http://www.roboken.esys.tsukuba.ac.jp/~ohya/pdf/iros97-oya.pdf

[5] **http://www.solarbotics.net/library/pieces/assy_sensors_prox.html**

[6] http://www.cs.brown.edu/~tld/courses/cs148/01/sensing/sensing.htm

[7] http://voronoi.sbp.ri.cmu.edu/papers/ICRA2002Latimer.pdf

[8] http://en.wikipedia.org/wiki/Image_processing

[9]http://www.machinedesign.com/ASP/viewSelectedArticle.asp?strArticleId=59848&strSite=MDSite

[10] http://www.hobbyengineering.com/SectionS.html

[11] G. Bishop and G. Welch. (2004, April 5). An Introduction to the Kalman Filter. [Online]. Available: http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf

[12] W. Burgard, D. Fox, and S. Thrun. (1998, March 19). Active Markov Localization for Mobile Robots. [Online]. Available: http://citeseer.ist.psu.edu/cache/papers/cs/499/http:zSzzSzwww-cgi.cs.cmu.eduzSzafszSzcs.cmu.eduzSzuserzSzthrunzSzpublic_htmlzSzpaperszSzfox.ras_act_local.pdf/fox98active.pdf

[13] W. Burgard, F. Dellaert, D. Fox, S. Thrun. (1999). Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. *AAAI/IAAI* [Online]. AAAI/IAAI. Available: http://citeseer.ist.psu.edu/cache/papers/cs/5834/http:zSzzSzwww.ri.cmu.eduzSzpub_fileszSzpub1zSzfox_dieter_1999_1zSzfox_dieter_1999_1.pdf/fox99monte.pdf

[14] W. Burgard, F. Dellaert, D. Fox, and S. Thrun. (2000, April 20). Robust Monte Carlo Localization for Mobile Robots. *Artificial Intelligence*. [Online]. Artificial Intelligence 128, pp. 99-141. Available: http://citeseer.ist.psu.edu/cache/papers/cs/26304/http:zSzzSzwww.informatik.uni-

freiburg.dezSz~burgardzSzabstractszSz..zSzpostscriptszSzrobustMonteCarlo.pdf/thrun01robust.pdf

[15] M. Castelonovi, A. Sgorbissa, and R. Zaccaria. Markov-Localization through Color Features Comparison. [Online]. Available: http://www.robotics.laboratorium.dist.unige.it/corso_robotica/Projects/pub01/MLTCFC_cr_1.0.pdf

[16] Y. Chen, J. T. Kwok, J. J. Pan, and Q. Yang. (2005, August). Accurate and Low-cost Location Estimation Using Kernels. [Online]. Available: http://www.ijcai.org/papers/1059.pdf

[17] P. K. Ganesh. Kalman-Filter Based Robot Localization. [Online]. Available: http://iiit.net/research/irl/kalman.html