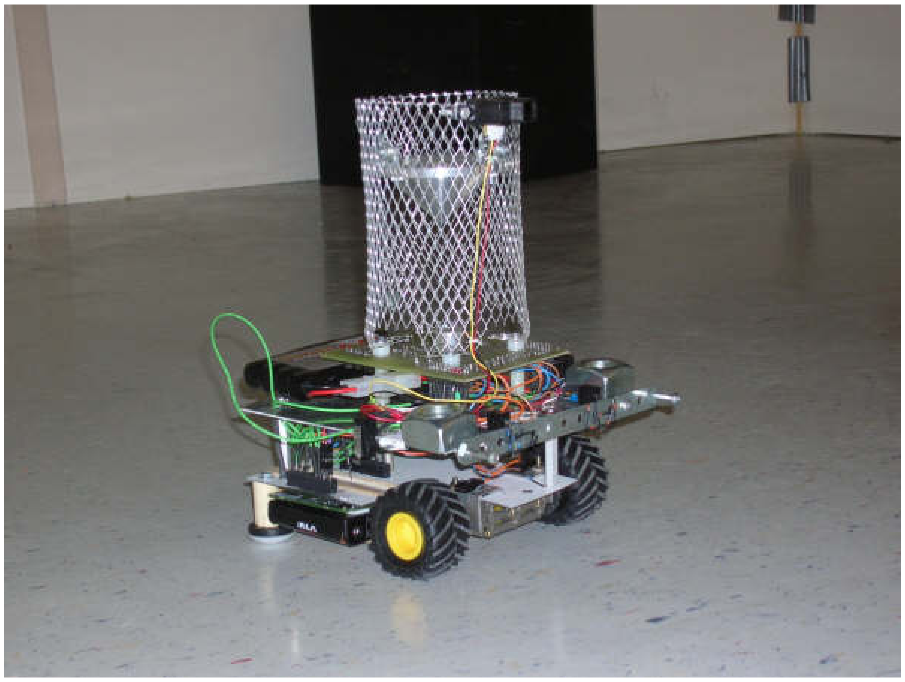
Electrical Engineering

New Mexico Institute of Mining and Technology

Mobile Node Localization and Searching Project

May 8, 2006



Prepared for: Dr. Kevin Wedeward
Dr. Aly El-Osery

Prepared by: Ivan M. Bow
Mathew E. Briggs
Phillip J. Darrow
Rhett A. Lawrence
Darla J. Le Blanc
Chris B. Whited
Mike M. Zaw

Table of Contents

Mobile Node Searching and Localization Project (May 2006)

Ivan Bow, Matthew E. Briggs, Phillip J. Darrow, Rhett A. Lawrence, Darla J. Le Blanc, Christopher B. Whited, Michael M. Zaw

***Abstract*—This paper explains a sensor network of mobile nodes with wireless capabilities. The nodes use a MicaZ microcontroller to communicate wirelessly with a base-station. The nodes localize using ultrasonic sound transmissions and Time Difference of Arrival. The task of the nodes is to locate a black box within a searching field. A random searching method is performed, and infrared sensors combined with a bumper switch system are used to locate the black box.**

***Index Terms*—localization, infrared sensing, robots, search methods, ultrasonic sensing.**

I. Introduction

Our design project is in the area of mobile sensor networks. The problem was to design a sensor network of mobile nodes which could travel an area in search of an object. The search area is a 3m x 3m walled-in arena with white walls, and the missing object is a 30cm x 30cm x 30cm black box. The nodes/robots are to be randomly placed somewhere in the searching field, and with no prior knowledge of their position they should localize themselves with respect to the field. The nodes search for the black box. After one node locates the box, the remaining nodes should also converge upon it.

A. *Design Requirements*

We were provided with 4 MicaZ microcontrollers (small microcontrollers with wireless communication abilities), an interface board to connect the MicaZ to a computer, and 3 motor kits with wheels. With these objects and a $250 budget we began our project.

We were required to have at least 2 mobile nodes in the field with independent searching abilities. These nodes could only communicate wirelessly through the MicaZ. A Base-station located outside the field could assist the nodes in localizing and other tasks. After being placed in the field, the nodes should localize in real-time. Another project requirement was the design and implementation of a GUI (Graphical User Interface). This GUI needed to display the searching field as well as the nodes within the field. The GUI was required to update the node locations at a refresh rate of 2 seconds, and once the nodes locate the black box it should also be displayed on the GUI.

This paper analyzes the approach we took to solving this problem. Our localization method, searching approach, hardware design, and programming are all explained. A general block diagram of our overall system is included in Appendix B.

Manuscript received May 8, 2006.

II. Localization

A. *Concept*

Localization is an integral part of this project. Initially we considered localization using an RF system, but this approach is too inaccurate for use in small areas. Ultrasound was decided to be a good alternative. Upon consideration of the design requirements we decided that each node should transmit ultrasonic waves in a 360º circle. Transmitting a full 360º enables quick and reliable localization of multiple nodes in short time periods. To enable each node to transmit in this pattern, an aluminum cone was mounted directly above the transmitter as shown in Figure 1. Aluminum cones were donated to our group via the Energetic Materials Research and Testing Center Machine Shop. The transmitters were placed on each mobile node, and four ultrasonic receivers were positioned at each corner of the 3m testing area. To avoid the problem of clock synchronization our localization technique uses a Time Difference of Arrival (TDOA) method.

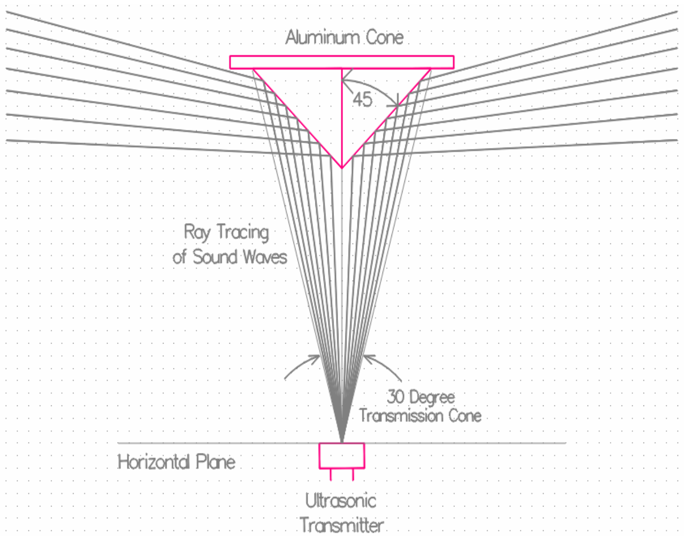


Fig.1. The cone reflects the sound waves to create a 360º transmission of ultrasound.

After researching TDOA methods [1], a non-linear, 2-Dimensional model was created in MATLAB and used to calculate each node's position. This original design used the solve function of MATLAB to take the non-linear equation and solve it. The first ultrasonic receiver to receive a signal and the two adjacent receivers calculated the position of the node. The inaccuracies in this calculation required about 20 samples per positioning to get a reading accurate to approximately 30 cm. The multiple samples necessary caused the calculations to take from 10 to 15 seconds per coordinate. These limitations did not allow this method to be used and still produce the required two second refresh rate.

B. Implementation

With this new realization in mind, the localization team began work on obtaining quicker and more accurate calculations. The original circuit for the receiver and transmitter began with an ultrasonic switch [2]. This switch does not require any information about signal strength, it only registers that a signal is received. In order for TDOA to work accurately, the sensitivity of the circuit must be very high. This high sensitivity means that the circuit can be vulnerable to outside noise. The receiver circuit has a potentiometer on a comparator to allow for sensitivity control of the design, and the circuit was compactly built to cut down on noise. Even after this, our localization output was still unreliable and would sometimes produce seemingly random numbers. The transmitter circuit also needed to be analyzed.

The transmitter is another part of this ultrasonic switch. It uses a 555 Timer to create a 40 kHz square wave to run the ultrasonic transducer [1]. The power to the entire circuit, including the 555 Timer, was cycled to create a stop and go transmit of the ultrasound. A problem we discovered with this circuit was that the 555 Timer did not always start on a 40 kHz wave. Due to the capacitance of the circuit, it could take up to 10 ms to stabilize to 40 kHz. This 10 ms period could produce errors in localization almost as large as the 3m arena.

To solve this problem, the 555 Timer was given power at all times of operation. Now a new method of cycling the transmission was needed. Since the transmit enable line from the MicaZ is only 3V max, it could not be used with a FET or BJT to switch the circuit on and off directly, since it is all run on 9-11V. To cure this problem, a BJT-FET series was created and used as a switch for the transmission line. The BJT's collector is connected to 9V via a 10k resistor, and its emitter to ground. When the BJT is off, 9V is sourced to the output, whereas a lower voltage is sourced when the BJT is on. The collector is then connected to the gate of a MOSFET and the drain of the MOSFET is connected to the transmission line right after the 1k resistor, and its source to ground. This set up allows the MOSFET to bring the voltage right after the 1k resistor to ground when it is on, regardless of whether the 555 Timer is transmitting or not. Thus, when the MicaZ transmission line is 0V, the BJT is off and the MOSFET is on, bringing the transmitter line to 0V and not transmitting. When the MicaZ sends a 3V signal to the BJT base, the BJT turns on and the gate of the MOSFET grounds, turning off the MOSFET and allowing the transmitter line to be what it likes, transmitting the 40 kHz square wave to the transducer. After this problem was solved, there was no delay difference between the transmitter and receiver. This enabled our calculations to be accurate and reliable.

C. Localization Programming

Further localization problems lied mainly in programming. The HCS12 microcontroller used its input capture pins to receive signals. When these signals were received the HCS12 stored a counter value for each received signal. Then the time difference of the counter values was taken and sent over the serial port. After a certain length of time, the HCS12 then used an output port to reset the receiver circuitry via BJT's used as switches. Once MATLAB received the values, it used a formula that takes 2 time differences and calculates the x and y coordinates for the node. The equation is shown below where Δt is the time difference of the current receiver and the initial receiver, x and y are node coordinates, x_n & y_n are the current receiver coordinates, and x_0 & y_0 are the initial receiver coordinates.

$$\Delta t = \sqrt{((x - x_n)^2 + (y - y_n)^2)} - \sqrt{((x - x_0)^2 + (y - y_0)^2)}$$

This equation is used twice to actually calculate the position of the node. The first uses the time difference and coordinates of the left adjacent receiver to the initial receiver, and the second uses the right adjacent receiver. Then MATLAB uses its 'solve' function to take the two equations and find exclusive x and y values. The problem here was that the third time difference could not be used because the equations were very exact and the third time difference would have to be exact for the equation to work, otherwise MATLAB would give an unsolvable function error. This function was still slow and was able to only give one coordinate per pulse.

To overcome this problem the 'syms' function in MATLAB was used, the equations that were used to solve for the coordinates were plugged in with its variables unsolved. Then the program produced a long and drawn out equation that could be used just the same as the original, but was only good for that one receiver. An equation was obtained using the first ultrasonic receiver to receive a pulse and the time difference in the ultrasonic receivers to the adjacent right and left corners of the box. Due to the basic symmetry of a square area, the same equation can be used by 'rotating' the 3 m area so that the second receiver is in the first receiver's position. Thus our final method evolved to use one equation no matter which receiver received first. Once the coordinates are found using this equation, they are simply shifted such that it fits our GUI assignments. This quick and linear equation was easy to implement and from this point forward the Python programming language was used for localization. It was found that for one pulse, all four receivers gave roughly the same coordinates no matter which one received first. The code uses this and with one pulse, four coordinates are found and averaged to get the best results and most accuracy.

Our receivers require a direct line of sight to the transmitter. Therefore, if one receiver was blocked, the data was ruined. To assess this problem a failsafe was implemented in the programming so that if a time difference exceeds the threshold value required for the ultrasound to almost traverse the box from corner to corner, a flag is set and the receiver number is stored. Then the program determines which receiver is opposite the blocked receiver and only uses data from that opposite receiver and the two adjacent to it. This makes the calculation a little less accurate because it does not have the average of four coordinates, but it is better than having bad data or no data at all if one receiver is blocked. If there is more than one time difference that is beyond this threshold, another flag is set and no calculations occur. The base station is given an error message informing it of bad data values.

D. Localization Hardware

After the program was finalized, the system needed to be fine tuned to work to the best of its ability. To get the best signal quality and accurate localization, the aluminum cones need to be exactly centered over the transmitter and level. The first method to mount the cones on the chassis used four rods to hold the cone above the transmitter. Upon testing, we discovered that the rods could block the signal if they directly lined up with one of the receiver's in the corner of the 3m field. That receiver would not receive a straight signal, but rather a reflection, giving bad data values. For this reason, the final design uses a wire mesh (or gutter guard from ACE Hardware) to hold the cone above the transmitter. A problem with this design is that the mesh is not as sold as the rods, and the cone can move slightly around the transmitter. It is required to level and center the cone by eye, which introduces inaccuracies.

Another hardware issue we encountered was mounting the receiver transducers. We began by using foam taped into the corners of the field, but this was very unstable. Also during final testing we discovered that when the black box was placed in the field it blocked our signals. So for the final demonstration, we chose dowel rods to hold the transducers about 91 cm above the floor and pointed down at a 30º angle. This increased our line of sight between the transmitters and receivers, but introduced a 3-Dimensional factor to our calculations. Even if the node was in the exact corner, it still had to travel about 70 cm to the receiver. Thus, as the nodes got closer to the corners, our localization was less accurate. We used a linear equation to correct for this. This made the center areas near the walls less accurate, due to the fact that the linear equation pushed the limits outside the field. This makes the nodes appear at the edge of the box even if they are not quite there yet. Time constraints prohibited us from fully accounting for the 3-Dimensional difference and with more time, better calculations could be used.

E. Localization Results

In the end, the accuracy for an immobile node could be 3-5 cm (before we raised the receivers and introduced a 3-D calculation error). For moving nodes we obtained 10-20 cm accuracy. Once the moving node stopped, the accuracy pinpointed once again to an accurate position. When displayed on the GUI, a moving node's position could be tracked. A stationary node would display at almost the exact same position after each refresh, unless an error was introduced into the receivers such that one of the data points was bad or thrown out. The scalability of this method of localization is only limited by the reception of the receivers. However this design could be implemented into RF systems for larger scale. If a larger area is used, the only changes needed in the equations are to use the correct size of field and to update the array of positions for the receivers. The equation could even be calculated in such a way that the positions of the receivers are scaleable with one equation. For our purposes it was more efficient to use our current equation due to the unchanging size of the field.

III. Sensors

Aside from the ultrasonic sensors for localization, we used three different types of sensors on our mobile nodes. These were infrared ranger sensors, open center encoders, and push button switches.

A. Infrared Sensors

After localization, our primary concern was object location and obstacle avoidance. For this we needed sensors. We considered many possibilities such as photo sensors, ultrasonic rangers, laser range finders, and infrared sensors. We dismissed the use of photo diodes because they do not have the range we require. Ultrasonic rangers could not be used to measure distance because they would interfere with our localization method. Laser rangers were too expensive for our project. Our final decision was to choose infrared (IR) sensors. The infrared sensors, GP2Y0A02YK, we picked have a measuring range from 20 cm to 150 cm. They provide precise detection of distance and output an analog voltage which could be directly connected to the MicaZ with no additional circuitry needed. Also, the price for each unit was well within our budget. We used these IR sensors for obstacle avoidance and object detection. They integrated nicely onto our mobile nodes and worked quite well.

B. Encoders

The second type of sensors we used were open center encoders. We looked at a few types of encoders before we chose to use open center, P12319-ND, encoders. Some of our earlier considerations were optical encoders and surface mount magnetic binary encoders. These were dismissed due to hardware mounting difficulties. We intended to use the open center encoders to keep track of the nodes' movements and assist in updating the GUI at the required refresh rate without constantly using the ultrasonic localization. These encoders have two output channels, phase A and B, and these particular encoders have 15 pulses resolution. By comparing the two phases, A and B, and which gets triggered first, we can determine the direction of the nodes' movement, forward or backward. The encoders are mounted on each wheel. They track of each wheel's speed, and theoretically we could synchronize the speed of both wheels using pulse width modulation. They are also intended to keep track of which direction the node turns as well as the angle of the turn the node made. In practice, the encoders did not work very well with our particular situation. The slick tiled floor of the arena caused the wheels to often spin/slip without actually moving the robot. This is another area of the project that could have been improved on with more time. In the end, we did not use the data from the encoders.

C. *Push Button Switches*

The third type of sensors we used were push button switches. We used these switches primarily for object detection. Since the IR sensors' readings are unreliable from 0 to 20 cm, we needed something to compensate for that region. We mounted these switches on bumpers as a fail safe device for object detection. Each node had four switches, two in front and one on each side of the robot. Their outputs were all connected together and sent through an RC circuit for debouncing. These switches worked well with our mobile nodes. Also, the switches and IR sensors provided some redundancy in objection detection.

IV. Programming

The programming team for this project had the job of writing all the glue that made all the hardware subsystems interact to meet the project's two main goals localization and searching. To make the code as readable as possible and to keep make modular coding a reality it was chosen to adopt an object oriented software design. This meant creating objects and therefore classes to support said objects. Programming can broken up into several parts namely MicaZ programming, HCS12 programming, low-level GUI, and high-level GUI.

First the MicaZ programming must be done under the Tinyos architecture and therefore in the language nesC. This new architecture and subsequent language required quite a learning curve to be comfortable with programming for it. In addition, applying the aforementioned object oriented programming model presented some challenges. The programming for the MicaZ can be further divided into loosely defined modules. The first and lowest module was named Drive. This module as the name suggests controlled all the micocontroller code which governed the driving the node around. This involved writing code which implemented PWM and at the same time as sending the correct signals on the control lines to the H-Bridge. This module was quite easy to implement as the strategy was adopted to use the underlying Atmel layer to directly access pins and other important registers. The most difficult challenge for this module was figuring out the interface description code for the Tinyos compiler.

The next MicaZ component which was implemented was called Node. This is what could best be described as a main object. Therefore this object called all the methods mentioned above in the discussion of the Drive module. This module unlike the Drive module needed to take advantage of many of Tinyos's features. Node could be further divided in two major components communications and data acquisition. The communications for this module relied on the GenericComm interface provided by TinyOs. The communications for the node programming actually plays one of the most integral roles in driving the operation of the nodes. This is because the Node module actually implements an event driven design model. This means that if no commands are sent then the object does nothing. The model limits some features that might be handy such as an automatic bumper kill switch. However, given the timetable for our project and the necessity for easy debugging and quick changes it was decided that pure event driven was fine to complete the goals of this project. The sensor code was relatively simple as there were existing examples for all the various types of input that were taken in.

Next, the HCS12 programming was quite simple as all it did was capture time differences and send them to the computer via serial UART (Universal Asynchronous Receiver Transmitter). First the there needs to be a justification as to why to use the HCS12 rather than rely entirely on the MicaZ as an exclusive base-station. This was done due to the on board hardware that the HCS12 includes but the MicaZ lacks. With the input capture pins on the MicaZ there is no latch to hold the values so the ISR must be relatively fast. By itself this is not necessarily a problem, except that the localization needs as accurate times as possible. In addition, during periods of heavy communication via the radio the ISR may not be as predicable as before and therefore the probability of missing the aforementioned deadline is increased. In summary, to implement the data acquisition the input capture functions of the HCS12 were used. Then to transmit the data simple Debug12 printfs were used.

Next, the low-level GUI implemented all the hardware interfacing needed for this project. The two pieces of hardware which communicated with the computer were the MicaZ base-station and the HCS12. The HCS12 was quite easy to implement as the data was packed into parseable strings which were then fed to the high-level GUI. The MicaZ base-station was a bit more complicated as it required the packing and unpacking of packets. This required quite a bit of error checking as well as a good amount of research to get the packets absolutely correct. Note that to communicate with a node this module was passed an address so that the address for the nodes were absolutely an abstract id not particularly hard coded in any fashion.

Finally, the High-level GUI allowed for interesting and user interactive features to be implemented with ease. One of the objects contained within the high-level GUI category is the Node class. This class of objects implements all the features available through the corresponding MicaZ module. This includes drive functions and sensor readings. The main purpose of this class is to allow the random searching algorithm to be written in high level human readable format. An example of this is to move an instantiated "Node node1" forward. All one has to do is simply call the forward method in the following manner:"node1.forward(...)." Using this interface the random search algorithm can be easily implemented. The concept behind our searching algorithm is that when the node determines it is within 20cm of an object the program accurately calculates the nodes position. If the node position is within 30cm of a wall then the sensed object must be a wall. The node then turns a random angle and continues under the same parameters. However, if the node encounters an object and determines that is not less than 30cm to a wall then that object must be the missing black box. A small correction subroutine is needed to check to make sure the box that it found is not really another node. This is done by simply calculating the remaining nodes positions and comparing to the found box.

In addition, the graphics for the GUI are implemented in a suite known as TkInter or Tk. This allows the normally command line driven python to turn into a graphical interface that any windows user would be comfortable using.

In conclusion, the programming for this project was not as difficult as some might think. However, without the use of standard design strategies, the actual coding could turn from well structured and easy to write into the ever feared spaghetti code that is found in many failed programming projects.

V. Chassis

The components of our chassis were included in a two level structure. The first level held our gear boxes, MicaZ, and H-Bridge while our second level held our bumper switches, printed circuit board, the main 9.6V battery, and our aluminum cone. Our chassis allowed for easy expandability and room for large changes in the design. However, it had difficulties with flat tires, wheel slippage, and gear box speed variations between the two tires. In future use, the provided wheels should be replaced by heavy duty wheels which can hold more weight and grip more easily. Instructions for the assembly of our chassis with necessary parts are included in Appendix A.

VI. Power

Our chassis required 3 different power sources. A 9.6V unregulated voltage, a 5V regulated voltage, and the 3 volts needed for the MicaZ. Our power system includes a 9.6 volt nickel metal hydride battery. This battery is rechargeable in 6 hours and provides 1300 milliamp hours. The 9.6 volts are regulated down to 5 volts as shown in the voltage regulator schematic of Appendix C.

The unregulated 9.6 volts was used for the ultrasonic transmitter and the transmit enable circuitry. The infrared sensor, the bumper switches, the encoders, and the H-Bridge all required the 5 volt input. The 9.6 volts was regulated down to 5 volts using a 7805 voltage regulator with a heat sink.

The MicaZ microcontroller runs off of its own 2 AA batteries which in series provided 3 volts. The life expectancy of the AA never came into play as they never required replacement. Basic AA batteries usually provide about 1800-2600 milliamp hours of playtime. If the MicaZ allowed for a 3 volt input, a regulated 3 volt input would be the preferred supply voltage for the microcontroller. By regulating the 9.6 volts down to 3 volts for the MicaZ, we would be capable of running the entire chassis off of one battery.

Depending on the components being tested, the actual lifetime for the 9.6 volt battery was around 2-3 hours for our design. These numbers could be enhanced by using a low dropout voltage regulator.

VII. Printed Circuit Board

A. Board Fabrication

As each node evolved, circuitry was added to enable the various sensors and components to function properly. This resulted in a large amount of wires and components needing placement on each node. As a solution to this "rats" nest it was decided to fabricate a PCB board. This decision was made since moving to a PCB board eliminated any "rats" nest wiring, created a professional look, saved space on each node, and aided in the placement of components. In order to realize a PCB board the following steps had to be performed:

1) Design

The first step in creating the PCB board was the design process. First the desired circuits were analyzed for best placement. These circuits include the voltage regulator, transmitter, bumper switch, and the encoders. Once all the circuitry was tested for full functionality drawings were created of possible component layouts on various board sizes/board layers. Components from the following circuits were placed in this design: A board size of 3" wide by 4" tall was chosen. This size allowed for easy placement on each node while allowing extra space for the 9.6V battery and any unseen additions. It was initially decided to use a single sided board for ease of manufacturing. It became apparent thereafter that due to the number of components and nets that a double sided design would be necessary.

The size of the board and the number of layers affected the localization ability of each node. In order to achieve accurate ultrasonic dispersion the aluminum cone had be directly above the transducer. This was accomplished with the utilization of computer aided measuring. The ultrasonic transducer was placed in the center of the board. Given the size of the aluminum cone, it was determined that mounting holes needed to be placed at a distance of 1.3774 inches from center every 90 degrees. At each of the measured points a pad was placed that would later be drilled out to allow for hardware installation.

For input/output (I/O) operations 2 twenty-pin, female headers are used. These connectors are placed on the edge of the board near the bottom. The input connector serves as an interface between the sensors and the board. The output connector serves as an interface between the MicaZ and the board. Both connectors contained power pins in order to enable later expansion.

Next, the remaining components were placed in the design around the mounting holes and transducer. This included voltage regulation, bumper switches, and encoders. All circuits that contained the same nets were placed in close proximity to each other. At this point the circuit layout was created using Altium Protel DXP 2004 SP2. Trace widths were set to be 25 thou (1 thou = 1/1000 inch). Power trace widths were set to 30 thou. Electrical clearance was set to 15 thou. The software and equipment used to realize the design was property of the National Radio Astronomy Observatory [3]. When the layout was complete, gerber files were created that were then changed to bitmaps (See Appendix C). The final board can be observed in Appendix D. The bitmaps are necessary for step 2, printing.

2) Printing

The second step in the board fabrication was the transfer of the design from the computer to blue paper. Using the electrical engineering department's facilities the bitmaps were printed on blue paper. The print job was scaled to 3" by 4" to fit the board size. Each bitmap was converted to grayscale and had the contrast increased. This allowed for the best transfer to paper. The bitmap files printed were "JuniorDesignBottomLayer.bmp" and "JuniorDesignTopLayer.bmp". (See Appendix C). It is important to note that one bitmap is mirrored. This enables the two layers to be aligned when transferred.

3) Design Transfer

Three pieces of 3" by 4" PCB board were then cut. Using an SOS pad, any oxidation was removed. The board was then cleaned. Heat was applied to transfer the design to one side of the board. Once transferred three holes were drilled in known locations in order to align the two sides. The second side was aligned with the known hole locations using wires. Once aligned the second side was transferred. At this point the board was inspected for any transfer defects. The first board that was transferred was a learning experience. All of the traces near the edge of the board failed to transfer. Simply drawing in the missing traces with a Sharpie fixed this problem but left an unappealing final result. The second and third boards did not encounter any errors in the transfer process. When the transfer process was complete the boards were etched.

4) Etching

Etching is a process in which excess copper is removed from the PCB. This is done with a ferric chloride chemical bath. All of the places where a design has transferred are safe from the chemical reaction leaving only the desired circuitry remaining. It was quickly learned that since the board is double sided, it must be regularly turned. This process took approximately 25 minutes per board.

5) Tinning

After the etching stage all that is left on the board are copper traces. In order to ensure functionality and longevity each trace was then tinned with solder. Tinning was performed at 700 degrees Fahrenheit.

6) Drilling

Holes were drilled in the tinned board to prepare for component placement. For all resistors, capacitors, and vias, a size #69 drill bit was used. All power pads, connector pads, regulator pads, and FET pads were drilled with a 1.15 size bit. The first board was drilled without any pad preparation and resulted in many inaccurate but functional pads. The two successive boards were drilled with hole preparation and were more successful. It was noticed that with a two sided board if the drill was applied too fast the pad on the other side would be lifted off the board. This was fixed after the first occurrence

7) Component Placement

All components were then placed on the board using the bitmap "JuniorDesignTopOverlay.bmp" as a reference (See Appendix C). All resistors, capacitors, and BJT's were easy to place. The FET, voltage regulator, 555 timer, Schmitt trigger, and connectors were more difficult due to inaccuracy in hole drilling. The functionality was never a problem with the difficult component placement. This difficulty was consistent through the fabrication of all three boards. The ultrasonic transducer was placed on the opposite side of the board than the rest of the components in order to allow for unobstructed transmission. One error was encountered with capacitor C8. It has the value of 680pF. This value was not available through the electrical engineering facilities. In order to accommodate for this value two capacitors were used in parallel, one on each side of the board.

8) Testing

Once a board was fully populated it was tested for functionality. This was done by first performing a continuity test to check for short circuits and open circuits. Each of the three boards was verified to have continuity. Power was then applied to each board to ensure correct operation. Using a multimeter the power was checked at each +5V net and +9V net. This included all power pins on the input and output connector. All boards were verified working. At this point the ultrasonic transducer had to be tuned to the correct frequency for optimal operation. Each board was tuned to 40 kHz by adjusting the 100 kΩ precision pot and observing the output on an oscilloscope. Tuning was the last required act on each board. The boards were now ready to be mounted on the chassis.

VIII. Conclusion

A. Design Strengths

The strengths of this design can be summarized in four main topics; localization method, modular software design, integrated circuit board, and expandability.

1) Localization Method

The dominant strength in this design was the ability to adapt a TDOA localization method from the wall used radio frequency spectrum to the less used ultrasonic range. By making this adaptation this design is easily scalable from a small 3 meter by 3 meter box to a playing field that is much larger. Along with the scalability of the localization method, TDOA also allows there to be very little communication between the base-station and the mobile sensors. The reason for this is due to the lack of synchronization that is needed between the base-station and the mobile sensors.

2) *Modular Software Design*

With the modular software design, troubleshooting of the individual parts of the robot are made substantially easier. Each sensor circuit can be tested very early in the design phase with it own module software. This allows for trouble shooting of hardware and software before integration occurs. Once a robot is completed, troubleshooting of the individual parts can still be accomplished using the same modules.

3) *Integrated Circuit Board*

By using an integrated circuit board that contains all of the circuitry need for the entire mobile sensor the design is easily reproducible. Also having all of the signals on one circuit card assembly signal integrity is easily controlled. An integrated circuit board is essential to having a robust and reproducible design.

4) *Expandability*

When trying to produce a working prototype mobile sensor one thing is constant, and that is change. For this reason it is imperative to have a platform that is easily expanded. The expandability also allows for changes in customer requirements without taking a large hit in cost or production time.

B. *Second Generation Recommendations*

The following are a list of recommendations for a second generation build of the mobile sensor. These recommendations are intended to make the reliability, ease of production, ease of hardware debugging, and functionality better.

1) *Chassis Weight Distribution*

To eliminate tire slippage the weight distribution of the chassis should be improved. Recommendations are to move more of the weight over the wheels of the chassis. This could be done by moving the battery mount to the front of the chassis while still keeping the weight of the cone close to the wheel base.

By improving the weight distribution of the chassis the problem of wheel slipping can be improved. This will in turn improve the usefulness of the wheel encoders. The more accurate these encoders are the easier it is for the program to keep track of where the mobile sensor is.

2) *Adding Test Points to PCB*

The hardware for this design is sometimes difficult to trouble shoot because most of the parts are not accessible to test probes. To make it easier to access, test points should be added to the sides of the integrated circuit board.

3) *Using Surface Mount Components*

By changing from through hole parts to surface mount parts, both reliability and ease of hardware debugging will be improved. By going to surface mount parts and moving them to the front of the board (same side as transducer) access to signals with test probes will be easier.

Surface mount parts will also allow for better reliability. The lower profile parts reduce the chance of breaking components when handling the board is reduced.

4) *Implementation of Wiring Harness*

To make for easier integration and improved signal integrity, the use of a wiring harness is highly recommended.

References

[1] "Time Difference of Arrival Principles" May 2006, http://www.era.cz/en/tdoa.shtml

[2] "Ultrasonic Switch," May 2006, http://www.reconnsworld.com/ir_ultrasonic_ultraswitch.html

[3] http://www.aoc.nrao.edu/gold/