Autonomous Node Development and Implementation

New Mexico Institute of Mining and Technology
EE382 Intro to Design
Group 3

Prepared for:
Dr. Aly El-Osery
Dr. René Aréchiga

Prepared by:
Michael Jew
Brianna Klein
Jake Smith
Kristy Stong
Julie Walter
Kendrick Walter

May 7, 2007

Abstract

The basic breakdown of this project is that two autonomous robots communicate with each other and a basestation. The robots collaborate in finding a box and pushing it down a straight, white lane outlined with electrical tape. The approach to the problem included using tactile and light sensors to find the box, and another two light sensors to determine when a robot has gone outside of the lane. To control how fast the robot is moving, the speed of the motors is changed by the pulse width modulation (PWM) levels. These levels are of set values which are controlled autonomously by the basestation's algorithm. Group communication and good engineering skills provide the tools necessary to complete the project.

Table of Contents

List of Figures

I. Introduction

The objective of this project was to make two robots (mobile nodes) cooperate autonomously to find and push a box across the floor, while staying within a marked lane. The lane measured approximately 2 meters wide by 5 meters long, and was marked with black electrical tape. The lanes are comprised of boards that have been painted white to maximize the contrast between the surface and the black tape.

The robots were limited in size to 6in x 6in x 6in and the portion of the robot that comes in contact with the box could not exceed 3in. The robots would be placed on the ground facing the box so that driving straight will cause them to make contact with the box. They can be placed anywhere from 12in to 20in away from the box. Each robot would not necessarily be placed the same distance from the box. If this was the case one robot would come in contact with the box first then wait for the other robot to contact the box at which point they would proceed down the lane together. The robots began their journey down the lane via a command from the graphical user interface (GUI). Once robot motion is initiated they must proceed down the entire lane without any outside human input and without the aid of guidance beacons. All maneuvering must be taken care of by the algorithm based on feedback.

II. Hardware

A. MICAz

The MICAz, seen in Figure 1, is a 2.4 GHz wireless platform with RF capabilities that can be used for low-power sensor networks. The radio has high speed, 250 kbps, as well as hardware security. The MICAz runs TinyOS 1.1.7 which is a small, open-source operating system that was developed by Berkeley. It also has a 51-pin expansion connector that supports analog inputs, digital I/O, I2C, SPI, and UART interfaces, making it easy to connect external devices to the MICAz. The expansion connector can be seen in Figure 2.

Figure 1: MICAz

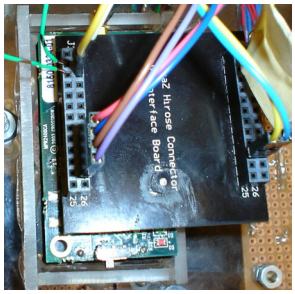

Figure 2: Expansion Connector

The MICAz microcontrollers are used to sort acquired sensor data, perform calculations, allow the nodes to communicate with each other, and execute code. Since they have built-in radio communications, it is easy for them to communicate with each other. Each of the two robots contains a MICAz. These two MICAzs are controlled by a third MICAz, called the basestation. As the robots receive different commands from the basestation, they respond by executing commands and sending packets back to the basestation. Each MICAz has input and output pins to read sensor data, as well as pulse width modulation (PWM) pins which allow the MICAz to control the speed of the motors.

B. H-Bridge

The H-Bridge enables communication between the MICAz and the gear box. Each gear box contains two motors that are independent of one another. Both the H-Bridge and motors are shown in Figure 3. The H-Bridge contains eight pins that need to be hardwired to other components on the chassis. Six pins are directly connected to the MICAz, four of which allow each of the motors to rotate either clockwise or counter-clockwise. The remaining pins are

enable pins that must be set before the wheels can be sent any PWM commands. Without setting these enable pins the motors will not respond to any PWM commands. The remaining pins are used to supply power to the H-Bridge and are connected to a common ground and to a 9 volt power supply.

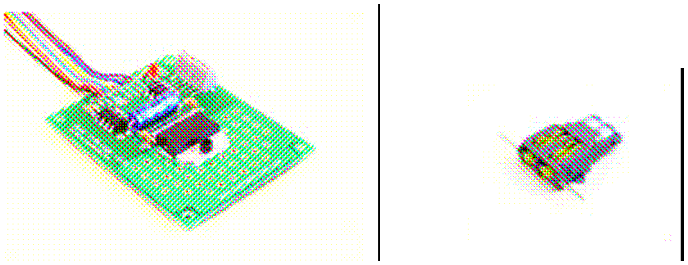

Figure 3: H-Bridge and Motor

C. Chassis

The chassis is the skeleton of the robot. As per the project specifications the size of the chassis is limited to 6in x 6in x 6in, while the portion of the chassis that comes in contact with the box may not exceed 3in. The MICAz, sensors and all other hardware must fit on the chassis. Both chassis were built out of three layers of Plexiglas, five bolts, and twenty-five nuts. See Figure 4 for a full picture. The bolts and nuts allow infinite adjustability to the height of each level which allows us to add or subtract components in a timely manner. This flexibility enabled us to continue building the robot without having all parts of the chassis completely finalized. When the chassis was connected to the gear box and wheels, the front of the robot needed to be supported so that it was parallel to the ground. To achieve this we mounted a furniture skid on a standoff to the bottom of the robot.

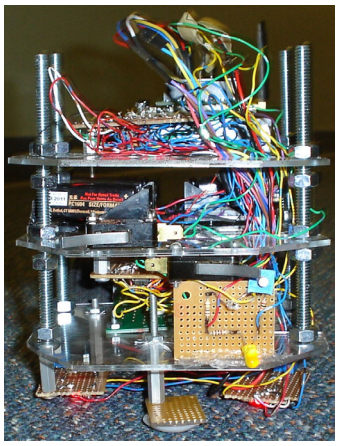

Figure 4: Completed Robot

D. Sensors

There are many sensors readily available on the market that could be utilized to detect the lines that marked the lane and the box. Many sensors were considered for detecting the lane lines and the distance to the box including camera, infrared rangefinder, laser rangefinder, ultrasonic rangefinder, and photo sensors. The first sensors we considered were cameras which, aside from being expensive, are limited by what they can see and require a lot of processing power. The second sensor type we researched was infrared rangefinders. These sensors measure angles and use them to calculate distance, but they saturate in sunlight and only work well in controlled environments. The third type of sensor we looked at was laser rangefinders. Laser rangefinders have 25mm accuracy which is very good, but the small beam width and complex data analysis were unappealing. We decided that we did not care about the distance from the robot to the box, so the rangefinders were deemed unnecessary. Another type of sensor we considered using was ultrasonic sensors which are similar to sonar. Ultrasonic is well-developed, and has the ability to find multiple items at once; however the resolution is poor and it does not work well with oblique angles. Ultrasonic sensors also presented a problem with detecting objects other than the box such as the other robot or walls, which is why we decided not to use them. Finally, we researched photo sensors and tactile sensors, both of which we ended up using for our final design. These two sensor types will be discussed below.

1. Photo Sensors

There are a few different types of photo sensors: chemical detectors with photograph plates, light detecting resistors (also known as LDR's or photo resistors), photovoltaic/solar cells, and photodiodes/phototransistors. The photo resistors decrease resistance as lumens increase. A cadmium-sulfide photo resistor can provide a range of 100-10M ohm resistance in light to dark environments. They have a broad frequency range and they are inexpensive. Photovoltaic cells, on the other hand, convert light directly as an energy source. However, they would be too large and expensive for this project. Photodiodes have low noise, low cost, compact, light weight, long lifetime, 80% efficiency, and no minimum voltage requirement. This is why we chose to go with photodiodes.

The light sensor system created for this project is made by using a photodiode, which allows current to flow through it when light is present, but permits very little current

through when shaded. Using this we fabricated sensors that output a voltage in the presence of light, and output a zero in the presence of darkness or when a line had been crossed. There is a resistor from the output of the photodiode circuit to ground which is used as a voltage-to-current converter. An LED was added to the same board as the photodiode to provide a reliable source of light where the shade of the chassis creates too much darkness. Also, a potentiometer was added to calibrate the sensors for different levels of light so that the robots can function in several different environments. An example of these light sensors can be seen in Figure 5.

Figure 5: Light Sensor

There are three light sensors per robot. One of these sensors is attached to the front of the robot to detect the shadow from the box. See Figure 7. This front sensor allows the robots to slow before impact and to detect when the tactile sensors do not have enough pressure. The other two sensors are mounted in front of the wheels on each side of the robots. See Figure 6. These sensors will detect when the robot has crossed a line.

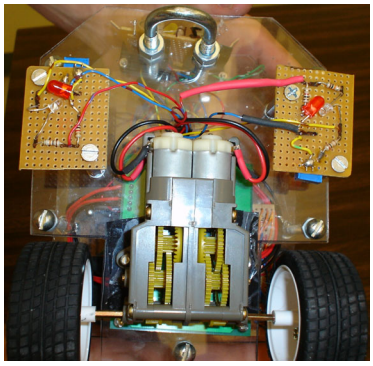

Figure 6: Bottom Sensors

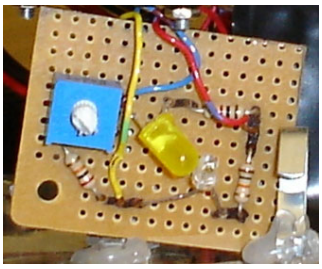

Figure 7: Front Light Sensor

The output of the light sensors did not produce enough of a voltage difference for the MICAz to recognize the difference between a high and low signal so, in order to magnify the difference, the sensor outputs were run through a comparator shown in Figure 8.

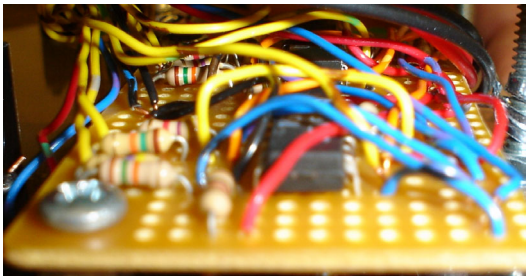

Figure 8: Comparator

2. Touch Sensors

The touch sensors are simple switches that can be in one of two states, digital high or low. There is a common pin and two output pins on each sensor. The common pin can be connected to either power or ground. The output pins are then connected to the common pin (ground) depending on the state of the switch. One output pin is connected to the common pin when the switch is open, the other when the switch is closed. The two touch sensors are mounted on the front of the middle layer of each robot, as shown in Figure 9, and are connected to a 3 volt power supply via use of a voltage divider supplied by a 9 volt battery. When the robots contact the box the switches are depressed and the MICAz receives a digital low signal.

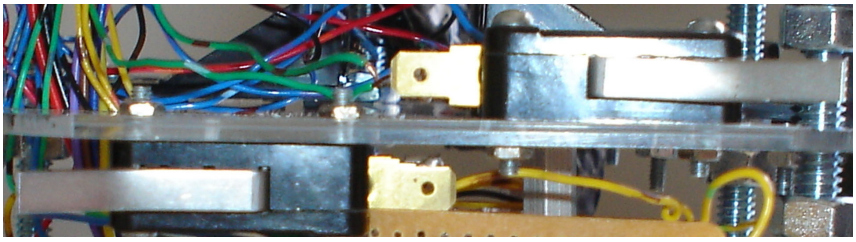

Figure 9: Touch Sensors

E. Power Supply

To power our robots we have decided to use three 9 volt batteries. Two of these batteries are used to give an 18 volt differential to the comparators, which require +/- 9 volts for operation. The third battery is used as a +9 volt supply that powers the H-Bridge and motors. Each mobile MICAz is powered by two AA batteries, while the MICAz used for the basestation receives its

power from the USB cable that links it to the computer. Each robot has a power rail circuit board where all power and ground connection initiate from. This eliminated the problems that arise from having many different grounds. Figure 10 shows all the power and ground connections.

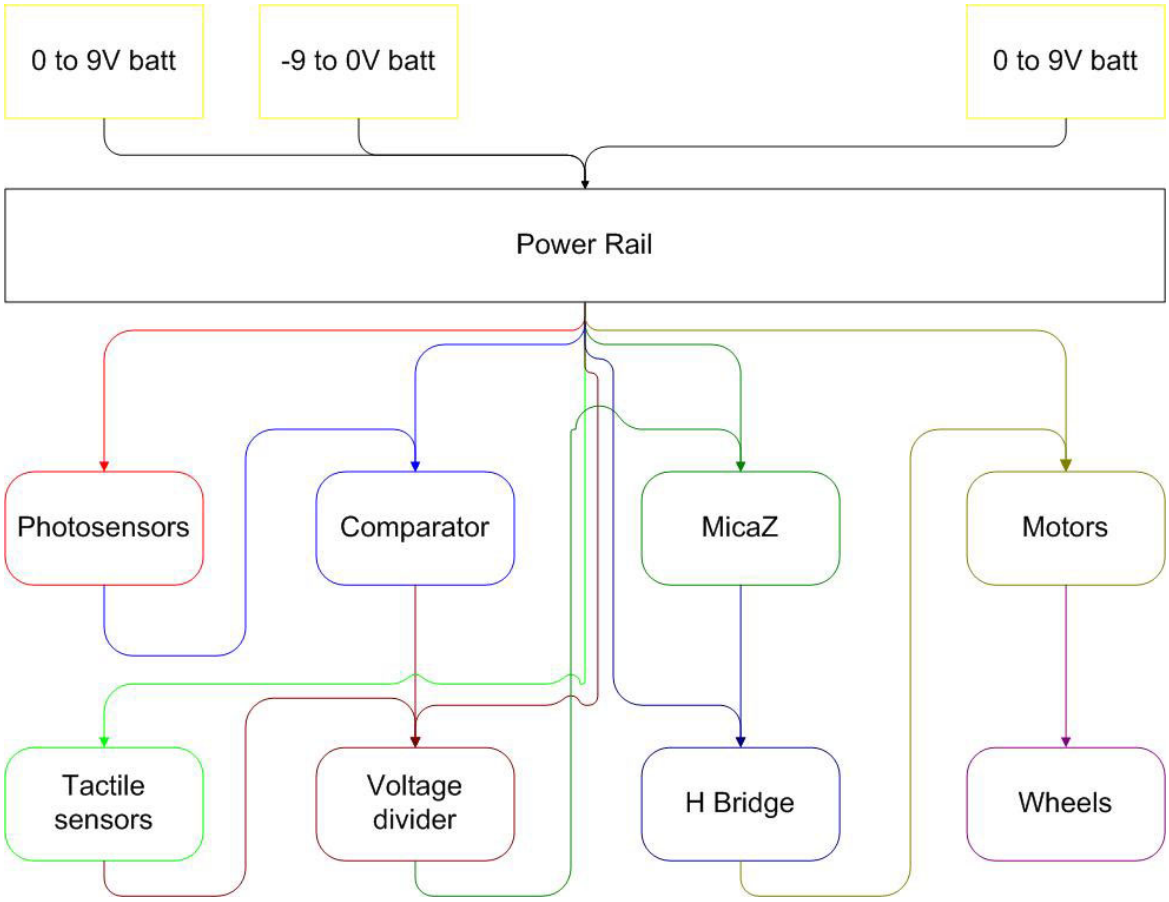

Figure 10: Power Supply

III. Software

A. Framework

The framework of the code is the backbone that all other parts are expanded from, and is crucially the first step towards software development. Part of the framework consists of a set of different packet types that can be sent and received by the MICAz. For this project, four different packet types were created. The first packet type is one that the PC sends to the basestation MICAz to control the overall state of the three MICAzs. The second packet type is one that the basestation MICAz uses to control the MICAzs on the robots. The remaining two packet types are sent by the mobile MICAzs to the basestation MICAz to inform the basestation of the status of the robot sensors and the direction and speed each robot is moving.

Another part of the framework consists of a set of basic send and receive modules. These are used in the software running on the MICAzs to allow send and receive packets of the desired type. These modules were combined into a program with basic send and receive functionality that is used to test the functionality of the different commands. This basic program was then enhanced to include sensor inputs and motor outputs, which form the basis for the software running on the MICAz. Below is a flowchart of how all the software interacts.

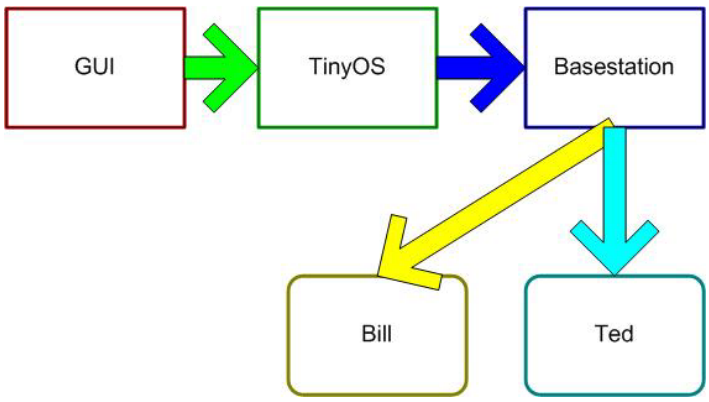

Figure 11: Software Flowchart

B. Pulse Width Modulation (PWM)

Pulse width modulation is a type of signal control that adjusts the amount of a period that a signal is a digital high. A demodulator, such as a low-pass filter, takes the percentage of high

signal and applies a voltage to the motors for the amount of time the signal is high. The longer the signal is high the longer the voltage is supplied to the motors thus increasing the speed of the motors.

PWM is used to control the speed of movement of the robots. A variety of different speeds are programmed into the robots. Initially, tests were performed in order to determine the best values to get straight movement. Both wheels on each robot are controlled with three different PWM settings that give it three different speeds. In addition, two sets of PWM values were included for turning purposes, as well as a set of values for veering that are used for the line following part of the algorithm.

When controlling the direction of the robots, two wires are needed to have two different directions and braking. With the motors wired identically, forward and reverse are achieved by setting the motor control lines to the same values. In order to turn, the robot moves forward or reverse, with one PWM value increased and the other decreased. It was discovered that if the motors turn in opposite directions, the robot does not have enough torque to move.

C. Control

1. Low Level

Low level control consists of the real time control of the motion of the robots. To process data in real time means that the processor deals with data that is collected and gives an output based on the collected inputs in a matter of time. The amount of time it takes the processor to give the output is determined by the system itself and is limited by the capabilities of the MICAz. This could range from nanoseconds to several seconds. We have our system set up to check sensor input every 50 milliseconds, and then the system sends the data to the GUI via the packet. Our low level control is implemented via closed loop control, meaning that feedback from the system is used. The feedback of our system comes from the touch and light sensors. We use the output of these sensors to determine what the robots need to do next.

2. High Level

High level control consists of the coordinated behavior of the robots to ensure that the

box moves straight down the lane. All of the high level control of our system is contained within the algorithm which will be discussed in greater detail in the next section. To make our robots work together to move the box down the lane we broke this large task into several smaller ones such as, find the box, wait for both robots to find the box, then push the box down the lane together.

D. Algorithm

The front end of the algorithm is objective based. There are five objectives that need to be addressed: listen for commands, find box, alignment, at box, push box, and wait. Each section of the front end algorithm is triggered by the feedback control. When the algorithm is started, the robots are in the listen state. From there they can be manually told to move in a certain direction or autonomously find the box. The find box section consists of moving forward and telling the robots to take inputs from the sensors which will send the execution to a different section of the algorithm. Line detection triggers the line follow portion of the algorithm. If a robot sees the line it will veer in the direction of the sensor that saw the line until the other line sensors sees the line, at which time it will begin to veer in that direction, effectively following the line for the remainder of the lane. If a node detects the box, it will wait for the other node to contact the box before it continues forward. The wait command is a simple ready state either while alone at the box. The following flow chart shows how the algorithm can get into different states and where it can go from each of those states.

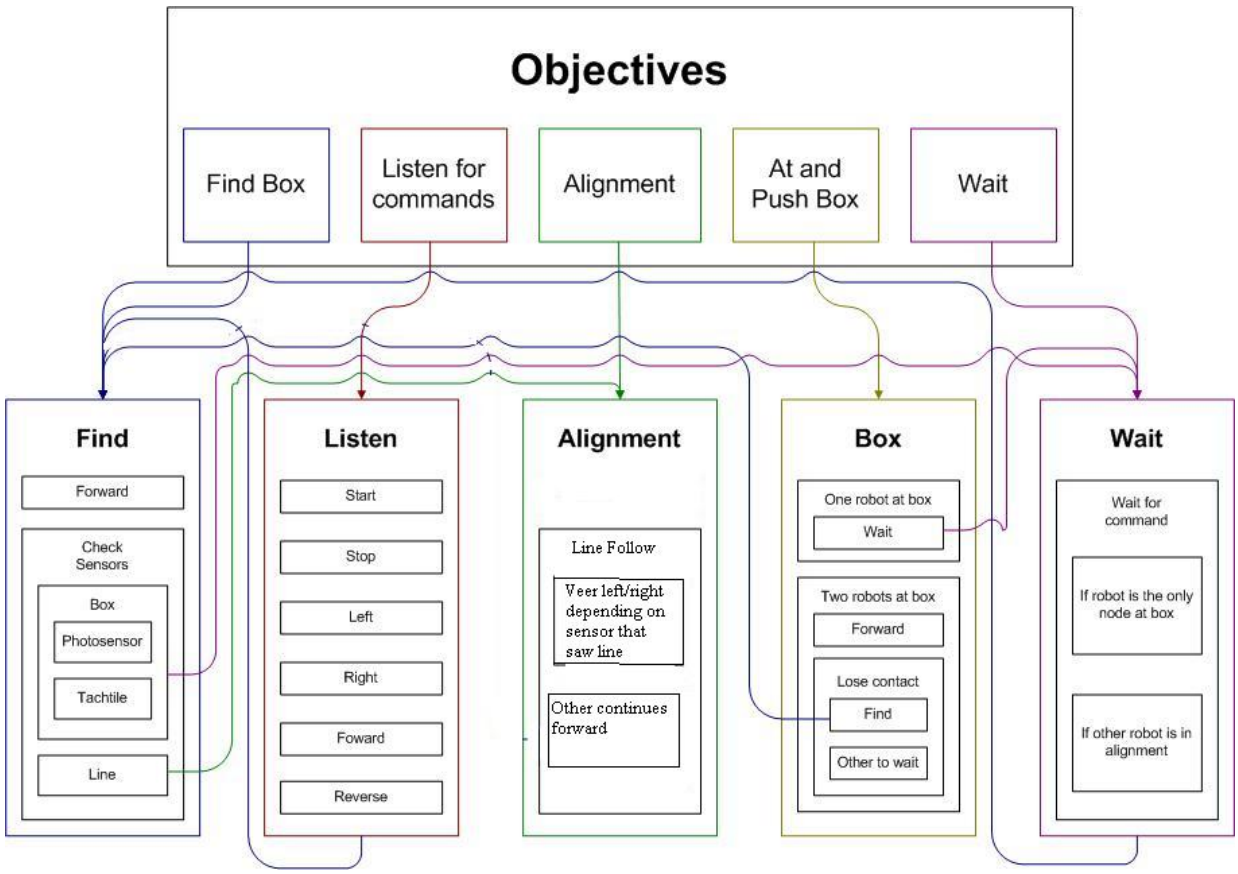

Figure 12: Algorithm Flowchart

The back end of the algorithm is where the code develops the means to jump from section to section in the front end. The feedback from the sensors is analyzed to determine where the robots are in relation to the other robot, the lane, and the box.

1. Sensors

The sensors are read and stored by the robot MICAzs. There are five sensors per robot and each is stored in a bit of the packet. Bits 0 and 1 are used for touch sensors. Bits 4, 5, and 6 are used for the optical sensors. There are a total of 32 different states the sensors can be in and the algorithm considers all of these states.

2. Robots

The robots are isolated from the control algorithm. They read the sensor inputs and control the motors. In order to put the sensor data in the packet, each sensor is checked

sequentially and its state is saved in the sensor variable of the packet. The robots listen to the basestation for commands for which direction to go and what the speed of the motors should be. The robots configure the PWM and control lines of the motors to match the commands being sent by the basestation.

3. Basestation

The basestation takes the sensor data from the two robots and uses it to determine when to change states. The basestation considers the robots "at the box" when the front optical sensor or either of the touch sensors has been triggered. When one of the robots reaches the box, it stops and waits for the other robot to reach the box. When both robots are at the box, the basestation tells them both to move forward. It watches for the line sensors to trigger and then starts the line following portion of the algorithm.

The last few states are dubbed "panic" states. These are states that the sensors should never get into but are included in case some of the hardware on the robot fails. All panic states stop the robots.

E. GUI

Two different pieces of software were tested to find a program that could be used to create a graphical user interface (GUI) for the project. The first piece of software tested was MATLAB but we were unable to correctly read from or write to the port that the MICAz was connected to. The second piece of software tried was Labview and we were able to get this to read from and write to the serial port very easily. Writing our own graphical interface for the project was not an option because the necessary programming experience to create a graphical program was not available and the project did not leave enough time for the team to work out how to utilize this form of programming.

After choosing Labview as the tool to be used to create the user interface, learning how to implement different functionality was achieved through experimentation and reading the documentation included with the program. The current form of the graphical user interface includes manual and automatic control of the robots. With automatic control enabled, the

control algorithm on the basestation and robots decides what needs to be done next. Then basestation sends status information back to the PC and the GUI displays the status of the robots on the screen. In manual control mode, the GUI is able to individually control the direction and speed of the two robots and still receives status information back from the robots. In both modes the sensor status is displayed as a set of lights that change color based on whether each of the sensors is triggered. Movement status is also displayed to the user. The sensor and movement status is found by parsing a packet the basestation MICAz sends the PC. Figure 13 shows what the GUI looks like to the user.

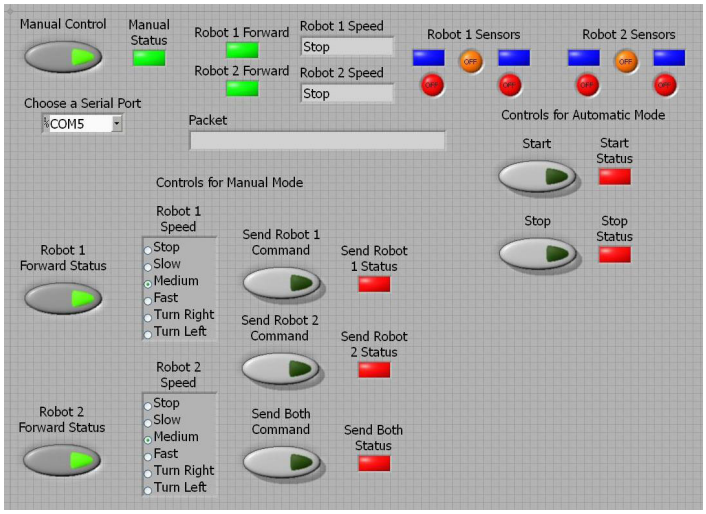

Figure 13: GUI Screenshot

IV.Hardware and Software Integration

Hardware can be defined as parts that physically make up the robot as a whole. Examples of hardware are wheels, the gearbox, bolts, nuts, Plexiglas, and sensors. Software can be defined as the programs that control the hardware. Examples of software are the GUI and all of the code for the sensors and the motors.

To make the hardware respond to the commands of the software there needed to be an element of coordination. The signals from the hardware such as the touch and light sensors give their output in the form of either 2 volts or 0 volts. The outputs of the sensors were then fed into the MICAz, which was programmed to recognize a 2 volt signal as a digital high and a 0 volt signal as a digital low.

The pulse width modulation controls the amount of time voltage is supplied to the motors and thus the speed that the motors go. The value that the PWM is set to controls the duty cycle of the signal. The duty cycle refers to the amount of time relative to an entire period that the signal is high, which in turn feeds voltage to the motor. For example, a 50% duty cycle feeds voltage to the motor for half a period and no voltage the other half of the period. The higher the duty cycle percentage, the faster the robots will move. The motors in our system respond the PWM commands from the MICAz by utilizing the H-Bridge, which was discussed in a previous section. Below is a diagram of all the interconnections of the hardware and the MICAz.

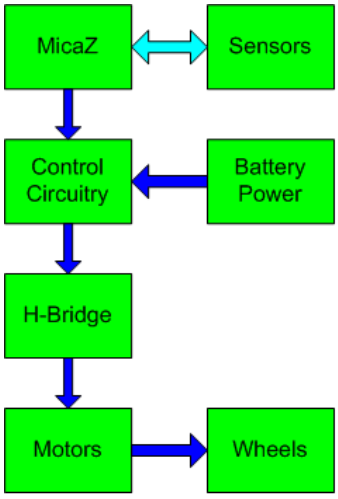

Figure 14: Software/Hardware Integration

V. Data and Results

A. Problems and Solutions

1. Third Wheel

The front support, or third wheel, of the robot has been the Achilles heel of this project for the group. Several different designs came and went before our final design of a furniture skid mounted on a standoff was implemented.

Originally, the third wheel was a piece of gauge wire that was bent and then melted into the bottom layer of the chassis. The idea behind it was to level the entire chassis so the robot would travel in a more predictable manner. Due to stability concerns we changed the design of the first scoot. To ensure that the scoot would be more stable two holes were drilled into the bottom layer of the chassis. Then a threaded u-bolt was inserted into the bottom layer of the chassis. Once the chassis was level, nuts were used to tighten the scoot into place. This new design allowed adjustment to the angle at which the front touch sensors came into contact with the box. Also, the new scoot could withstand more friction and overall provided a much stronger and more stable robot. Unfortunately, once the final surface was available to use for testing this scoot became a problem. The running surface was painted with a soft latex paint which the u-bolt sank into. The robots did not have enough power to push the robot out of the indentation the u-bolt created and thus could not move down the lane at all. To fix this problem we decided to change the scoot to something that rolled. Several solutions were tried such as putting washers on the present u-bolt and ball casters. The washers did not stay in place, and allowed the u-bolt to fall back to the surface, and the ball casters did not have enough friction to deal with the play in the gear box. Finally, skateboard trucks that had been removed from miniature toy skateboards were mounted PC-3 board then mounted this board to the robot in much the same way that the u-bolt was mounted. This also proved to be a problem as it did not allow for turning. The last and final design that was implemented was a plastic furniture skid that was mounted on a standoff to the bottom of the robot. This had enough friction to ignore the play in the gearbox, but enough freedom to allow the robots to turn.

2. Light Sensors

One problem that was encountered with the light sensors was that in the shadow and the difference between high and low signals was very small. To compensate for this, a comparator was added to the circuit to magnify the difference in high and low signals from the sensor outputs. Also, the sensors were so sensitive that the shadow of the robot sometimes triggered false line sightings. To fix this problem potentiometers were added to adjust the sensors for different amounts of ambient light.

3. Chassis

Whenever the robots ran we noticed that the nuts that were used to support the different levels of the robots had a tendency to vibrate down the bolt pillars. While this did not affect the working of our robot, it was an annoyance to have to move the nuts back into place every time we stopped the robots. To fix this we decided to use thread locker, which turned out to be a huge mistake. The next day we realized that the thread locker had decimated our robots. We believe that when the thread locker dried it expanded against the nuts and Plexiglas, causing the Plexiglas to crack and shatter in any place that made contact with it. While the sensors and MICAz remained undamaged, neither chassis was salvageable. This resulted in a twelve hour marathon re-build of both robots from the ground up. With the new chassis we decided to let the vibration of the nuts be an annoyance rather than risk another rebuild.

B. Budget

For this project each group was allocated $150. As a group it was decided that we needed to work on conserving the money by buying inexpensive items and building the parts ourselves. The largest part of our total costs has come from batteries, totaling over $50. As a result, the team has spent around $117 and will be completing the project well under budget. A table displaying the total budget can be found in figure 15 below.

COSTS

Description	Unit Cost	Units	Total Cost	Running Cost
Push Buttons	$0.50	2	$1.00	$1.00
Battery Holders	$0.83	2	$1.66	$2.66
Wire Caps	$0.05	10	$0.50	$3.16
Push Buttons	$0.50	2	$1.00	$4.16
Phototransistors	$0.50	12	$6.00	$10.16
Comparators	$1.00	4	$4.00	$14.16
Quadrature Detector	$0.80	2	$1.60	$15.76
9V Batter Cap	$0.30	4	$1.20	$16.96
Battery Holders	$1.50	1	$1.50	$18.46
Potentiometer	$1.75	2	$3.50	$21.96
LocTite	$4.28	1	$4.28	$26.24
Hot Wheels	$0.97	1	$0.97	$27.21
Matchbox	$2.98	2	$5.96	$33.17
Skateboards	$7.58	1	$7.58	$40.75
Miscellaneous*	$16.92	1	$16.92	$57.67
Last Minute Fixes**	$1.15	1	$1.15	$58.82
Perforation Board	$1.00	5	$5.00	$63.82
Battery 9V	$1.86	29	$53.94	$117.76
TOTAL				$117.76

*Includes bolts, nuts, and Plexiglas
**Includes comparator, LED, and crimp pins

Figure 15: Final Budget

C. Improvements / Future Improvements

1. GUI Improvement

The robot code framework includes the functionality to operate each robot individually in automatic mode or both together. This functionality could be included in the user interface for testing purposes. An example of how to use this mode of the robots is when testing different box finding and lane tracking algorithms. Setting one robot in automatic mode allows the user to control the direction and speed of the other robot and allows the user to manually find the box. This has the advantage of stress testing any box finding and pushing algorithm because a human operator can force the robots into situations the algorithm will normally never encounter.

2. Autonomy Improvement

The degree to which is robots can operate independent of the basestation could be

improved. One option is to make one of the robots the master and let it control the movement of the other robot. Another option is to disintegrate the control algorithm. Disintegrating the control algorithm is advantageous because it allows the robots to operate without a computer monitoring them. However, it suffers from a drastic increase in robot code which has the possibility of overwhelming the microcontroller's processing capabilities.

3. Line Sensor Improvement

While the line sensors work well, they do have the disadvantage of needing constant recalibration in order to function properly. The sensors are very close to the ground and sometimes the position of the LEDs causes the photo sensors to never see the lines. To fix this we could possibly buy line sensors instead of using the ones we made ourselves. This would be more expensive, but less hassle. Another possibility is that we could find a way to securely fix the position of the LEDs and photo sensors in such a way that they won't move. Another way to improve the line sensors would be to have printed circuit boards rather than perforation board. This would make the circuit very neat and reduce the risk of wires becoming unsoldered. It would also have helped if the comparators were not directly soldered to the board, but rather connected through a socket. This way, if a chip got destroyed, it would be much easier and less time consuming to replace.

D. Work and Manpower

In order for our group to work efficiently as a team we decided to try and break the project into several main parts. The parts consist of chassis design, tactile sensors, photo sensors, GUI, algorithm, budget, and research papers. Many of our group members had a hand in several different parts of main project. A basic summary of what we were responsible is presented below:

Michael Jew: algorithm, GUI, software

Brianna Klein: photo sensors, comparator, research papers

Jake Smith: chassis design, photo sensors, budget, and research papers

Kristy Stong: photo sensors, comparator, research papers

Kendrick Walter: chassis design, tactile sensors, GUI

Julie Walter: chassis design, tactile sensors, GUI

This is a basic guide to where each individual spent the majority of their time. The above list doesn't include other minor tasks that needed to be done. Though we each had individual tasks, we had meetings on almost a daily basis and set up goals that we would try to meet by the end of the day. By dividing our manpower we were able to accomplish much on any given day.

E. Testing and Results

Each individual part was tested before it was put on the chassis. The tactile sensors were tested by seeing if they would trigger when they came in contact with the box. The photo sensors were tested in a similar manner to ensure that they would in fact see the difference between the line and the lane. After the photo sensors were built, they were tested by running them over light and dark areas on the floor. This proved to be successful and was incorporated in our chassis design.

Once each individual part was tested we began to mount them on the chassis. Once all of the components were added to the chassis we began doing the same tests but with all of the components working together by using the GUI to command the robots.

The testing of the photo sensors was a multi-stage process. First, they were tested in direct light, with no shading. In this case, 5V to 0V difference was present from the output of the photo sensor. Then they were tested in the shade created by the robot itself and it was found that the difference had diminished by quite a bit. To compensate for this comparator circuits to increase the difference between high and low to 9V to 0V were added. The 9V was too high for the MICAz, so a voltage divider was added to make the difference from 3V to 0V. Once the sensors were mounted on the robots, many minor adjustments needed to be made in order for the sensors work correctly. A potentiometer was also added to the light sensors was a potentiometer to increase the voltage output so that the sensors could be adjusted to several different levels of light. The adjustments needed mostly consisted of changing the position of the LEDs until we got the proper output from the comparators.

Another important test was for motor control. In our design we decided against using feedback

in order to ensure that our robot would travel in a straight line; because of this, we had to test different PWM values. The motors in the gear box travel at different speeds even though the same PWM values were used. We tested several different PWM values via a process of trial and error to find the values that made the robots travel suitably straight. We found three sets of values for each robot to give three different speeds of forward travel. We also used one of these sets for a single reverse speed. We also tested several different values to make the robots execute right and left turns. Turning is implemented by setting one wheel faster than the other which causes the robot to turn toward the side of the slow wheel. Again we used a process of trial and error to find values that gave a suitable turning radius. This process proved to be rather inconsistent. As the batteries lost voltage, the values that previously made the robots go straight didn't necessarily make them go straight anymore. To compensate for this we implemented the line following portion of the algorithm.

The GUI was tested by making sure it read the packets being sent back by the mobile MICAzs and that it displayed the appropriate sensor data. We tested this by manually tripping sensors to make sure the GUI responded correctly.

F. Timeline

Below is a timeline that shows the dates we planned to have certain components of the project completed by and the date we actually completed those components.

Landmark	Expected Completion date	Actual Completion date	Status
Shipping List	2/28/2007	2/28/2007	On time
Communication between basestation And platform	3/8/2007	Late February	Early
Initial GUI	3/8/2007	3/8/2007	On time
PWM Functional	3/8/2007	Early April	Late
Sensors mounted	3/22/2007	Early March	Early
Sensors program	3/22/2007	Early March	Early
Complete and working Program	4/15/2007	5/3/2007	Late
Complete project	Late April	5/3/2007	Late*

*While the project was finished after our expected date, it was still finished by the deadline.

Figure 16: Timeline

VI.Conclusion

Over the course of the semester and the project our group took away several important lessons. We learned the importance of open communication among group members, how to work as a group, the importance of good documentation and standardization, the importance of adaptability, and good engineering practices in general.

Each class period started with a group meeting where we discussed the status of everything we were working on, and what needed to be done by the end of the day, and what else needed to be started that particular day. These meetings really opened up the lines of communication; every group member always knew the status of everything that was going on, even of the components they were not working on.

Good documentation on our part basically saved our project in the last week before demos. When our robots were destroyed by the thread locker, we simply reverted back to the measured drawings of the robot that we made with Inventor to help us rebuild the robots from the ground up. We also had all the circuit diagrams for everything on the robot so when it came time to reconnect wires that got pulled out from being jostled around during the rebuild it was not a huge ordeal to find where they went. This is also where the standardization of wire color for power, ground, light sensor output, and touch sensor output was a huge help. Each type of wire had its specific color, so we were able to quickly locate where it was to be connected.

The lesson of adaptability was one that we learned the hard way. We had designed parts of our robots so specifically that they only worked on the tile or the boards before they were painted. This was what led to so many different implementations of the third wheel. We ended up spending time replacing the third wheel that we had designated testing the algorithm and making any adjustments. Instead, we spent countless hours the days before the demo doing this on our own time.
Even though the group had several crises the final week we managed to pull it all together in the end to have functioning robots and algorithm.

VII. Appendices

A. Appendix A: Circuit Diagrams

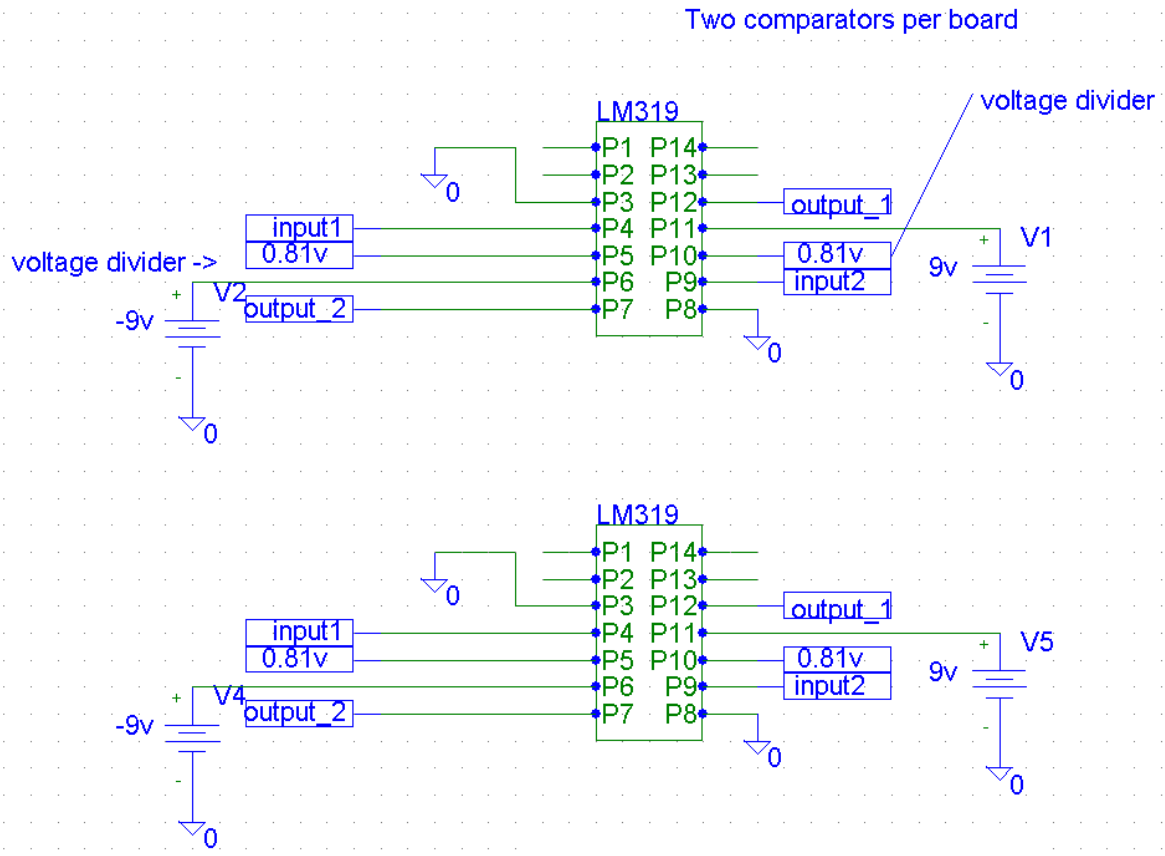

Figure 17: Comparator Circuit

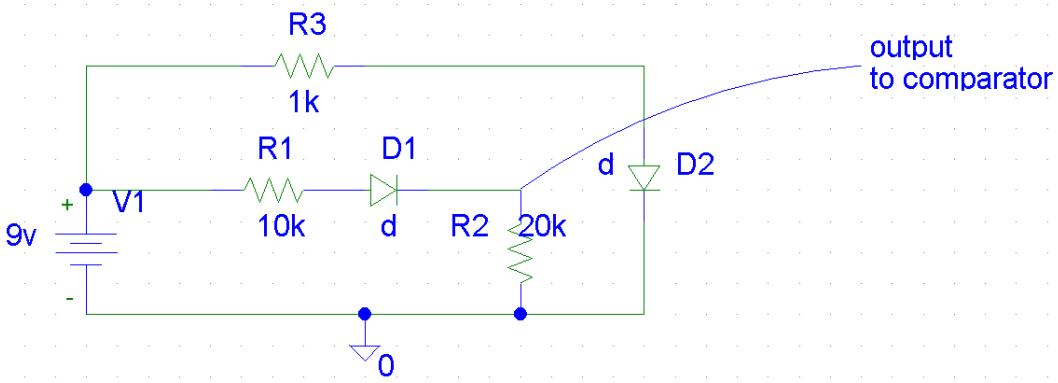

Figure 18: Light Sensors Circuit

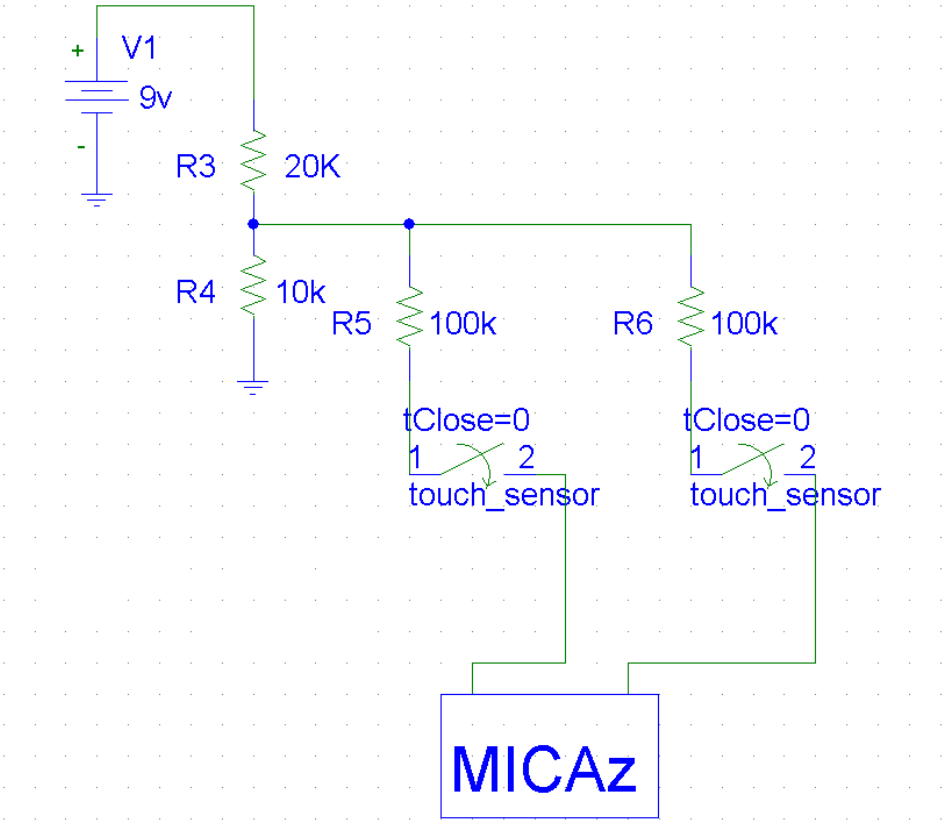

Figure 19: Touch Sensors Circuit

B. Appendix B: Inventor Drawings

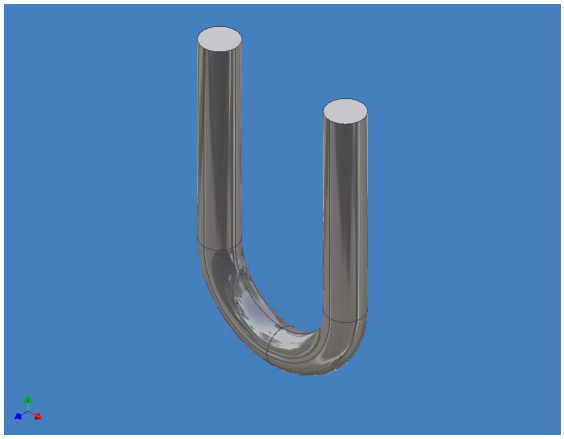

Figure 20: Original Third Wheel

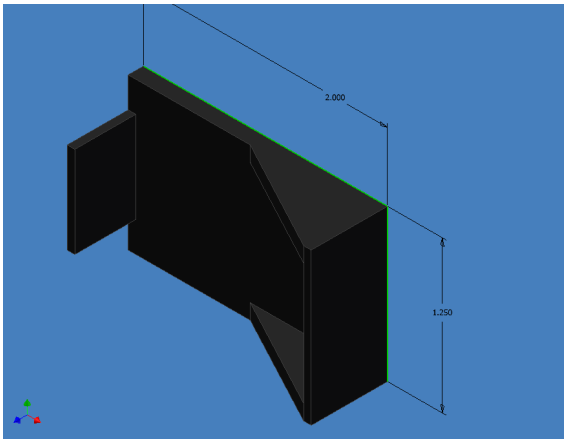

Figure 21: Battery Holder

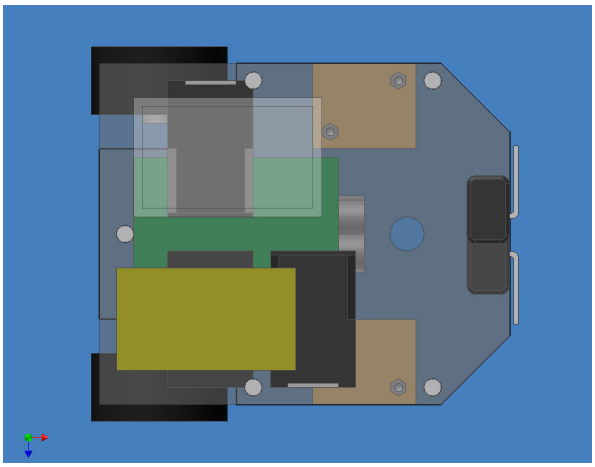

Figure 22: Robot Top View

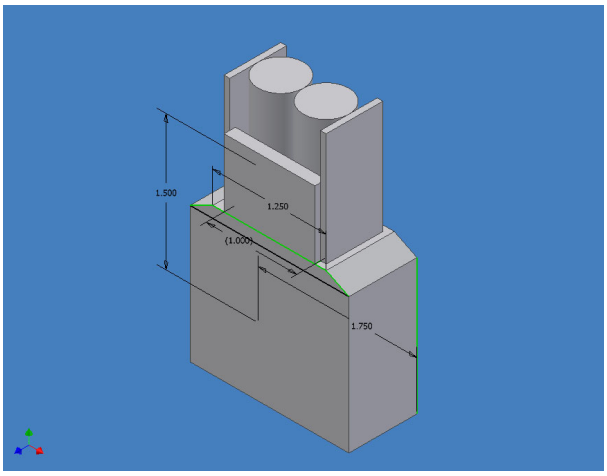

Figure 23: Motors

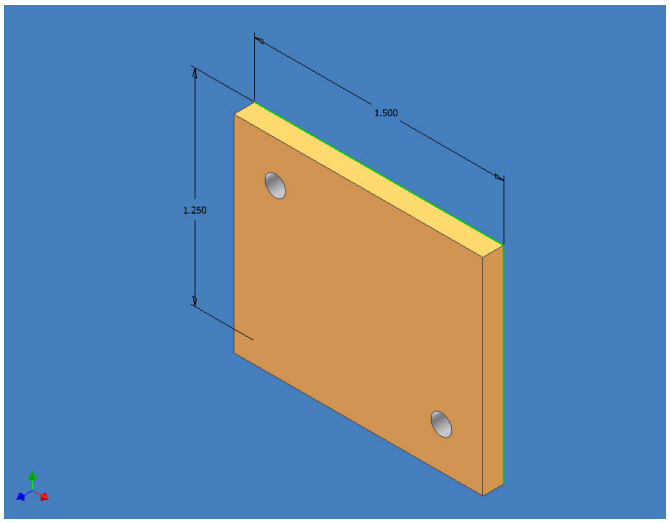

Figure 24: Light Sensor Board

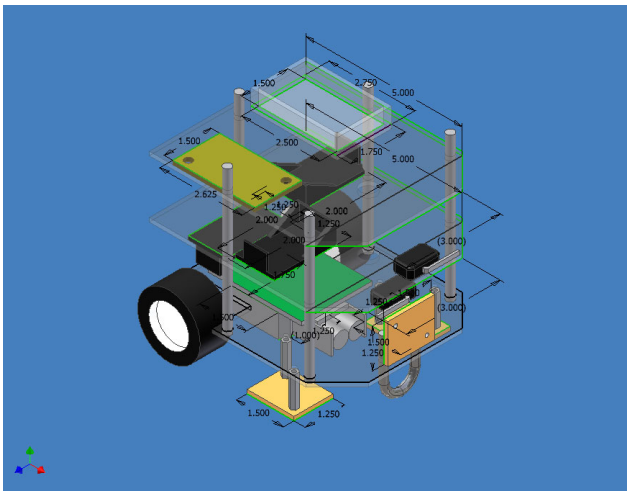

Figure 25: Robot Measured

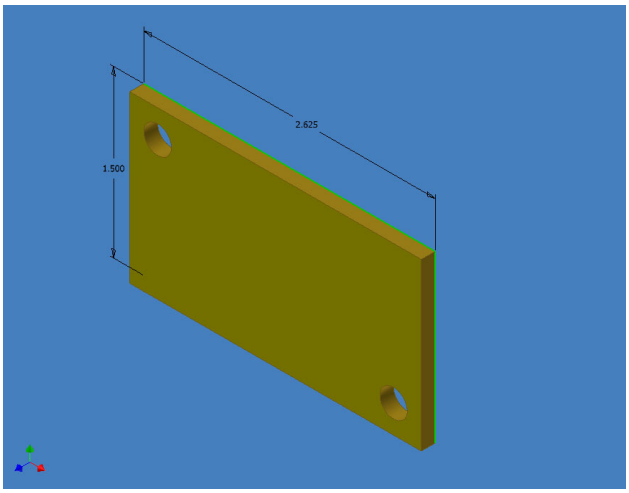

Figure 26: Comparator Board

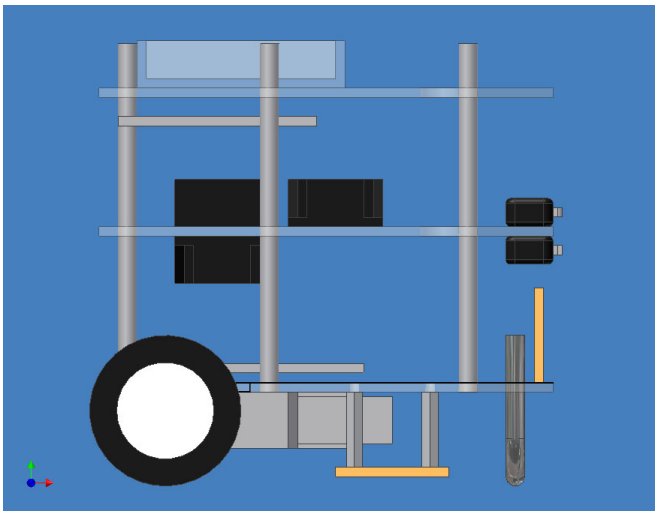

Figure 27: Robot Side View

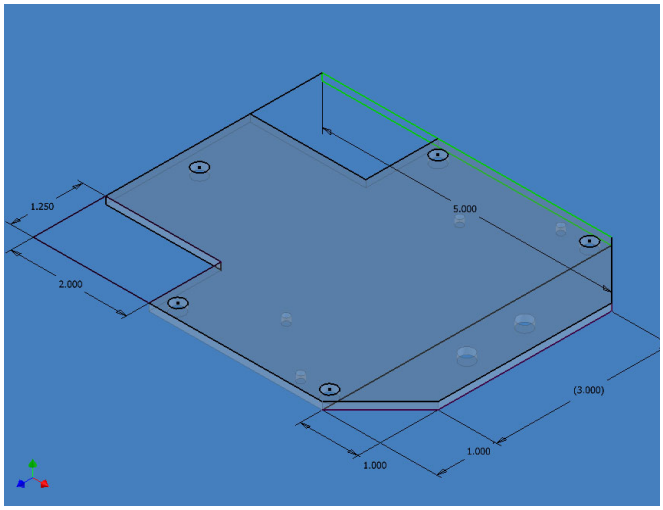

Figure 28: Bottom Plate Measured

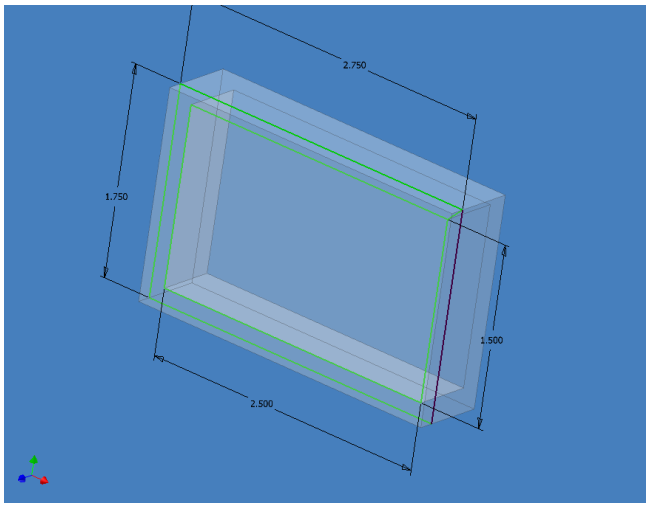

Figure 29: MICAz Holder

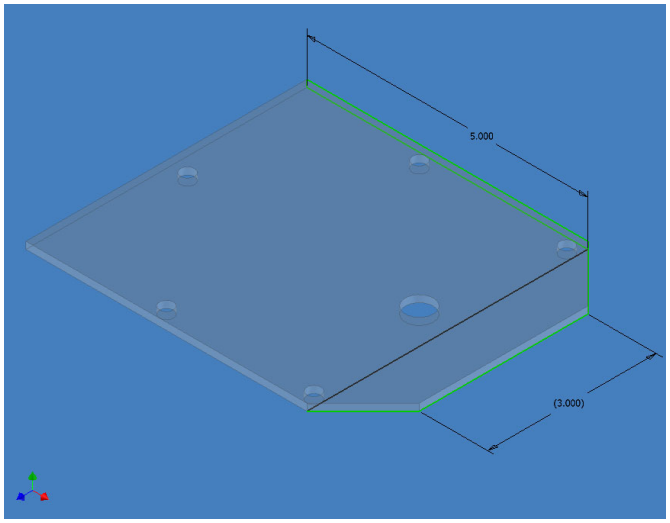

Figure 30: Middle/Top Plates Measured

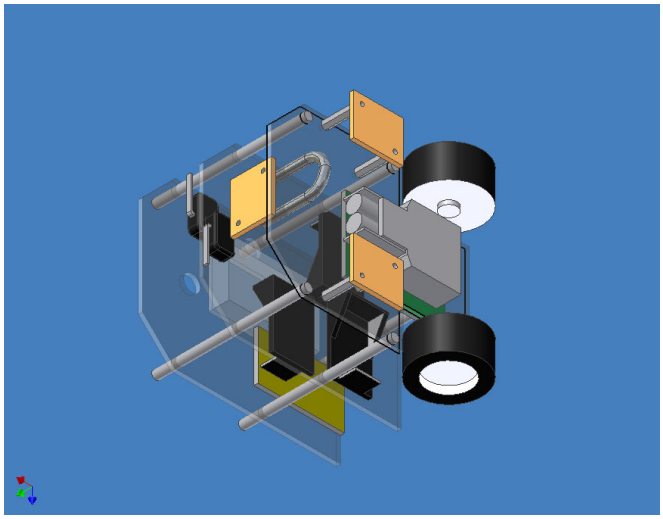

Figure 31: Assembled Robot Bottom View

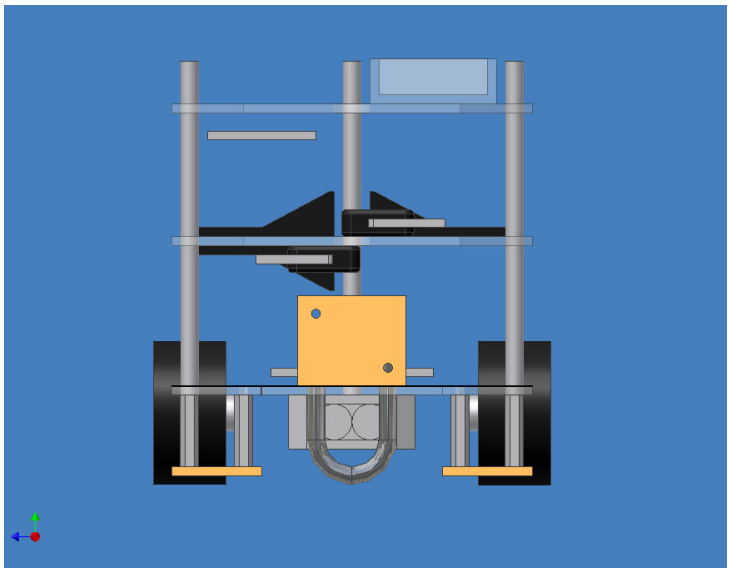

Figure 32: Assembled Robot Front View

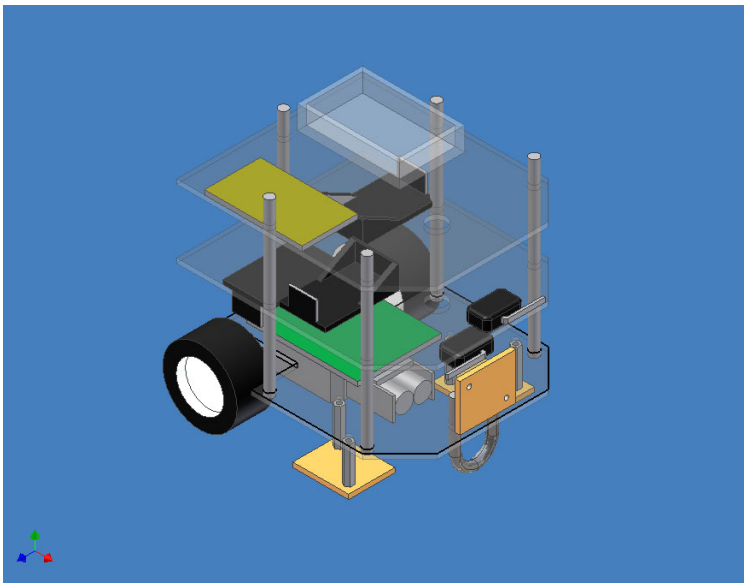

Figure 33: Assembled Robot Isometric View

C. Appendix C: Subsystem block diagrams

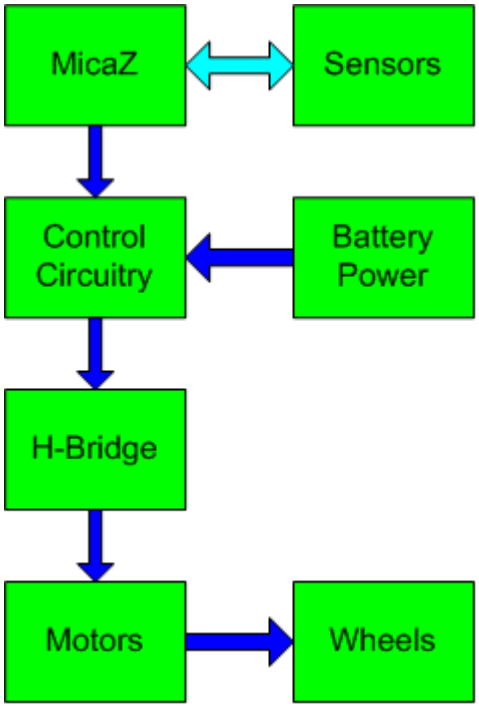

Figure 34: Interconnections of Components

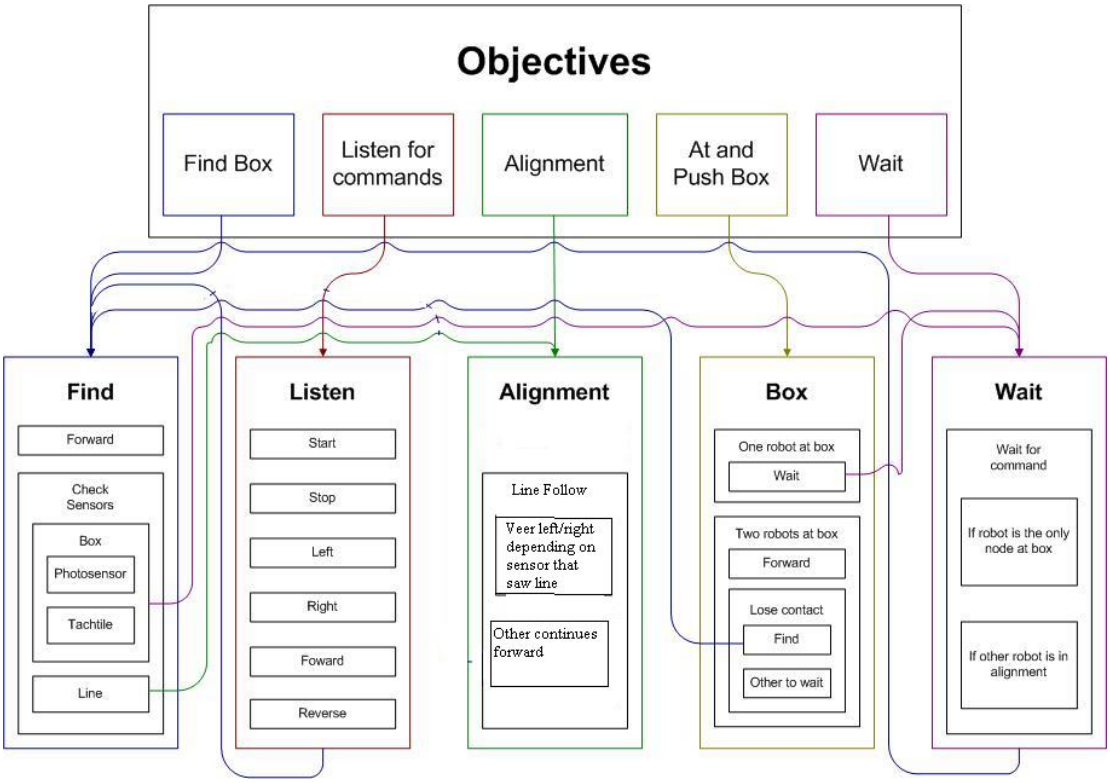

Figure 35: Algorithm Block Diagram

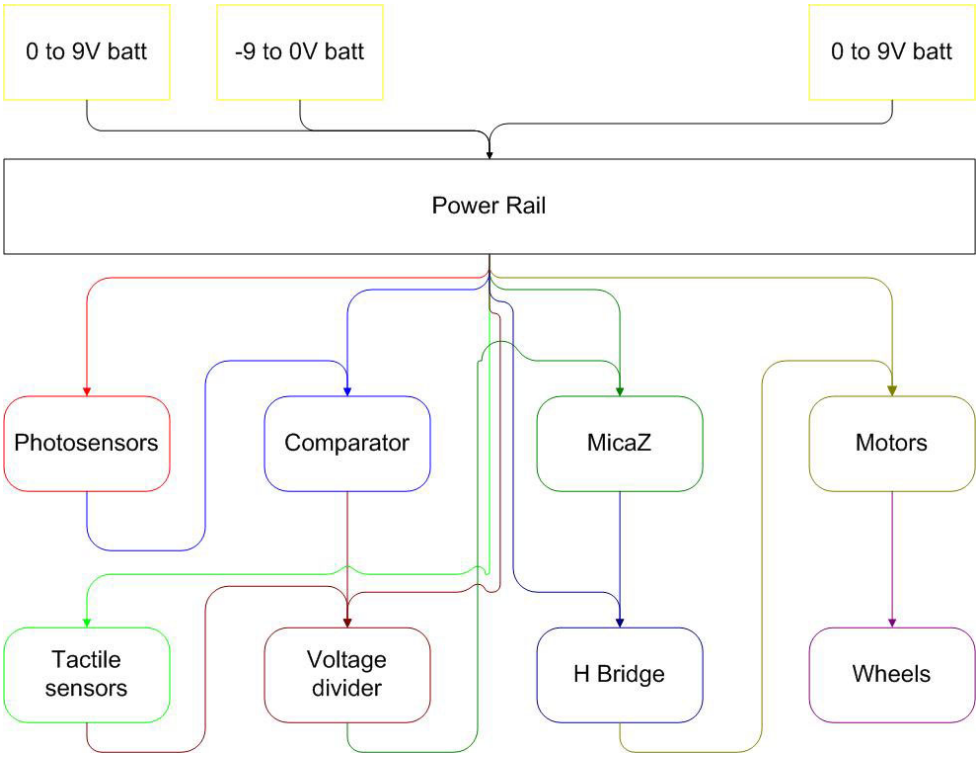

Figure 36: Power Connections Block Diagram

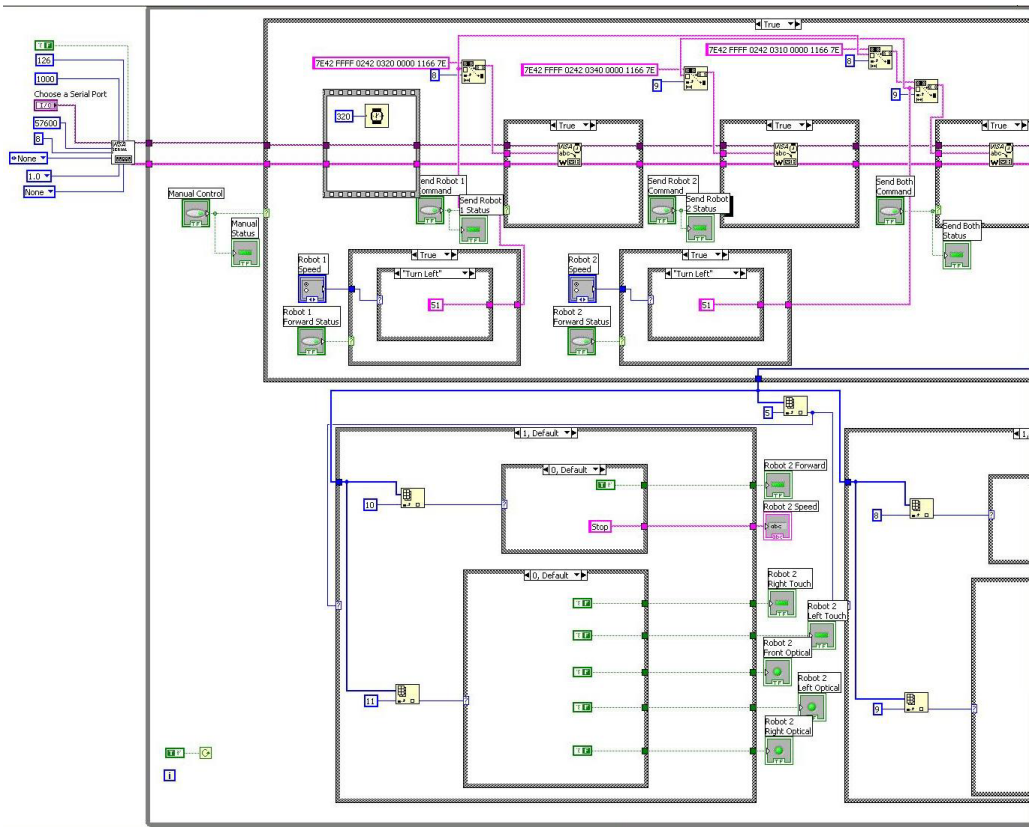

Figure 37: GUI Backend

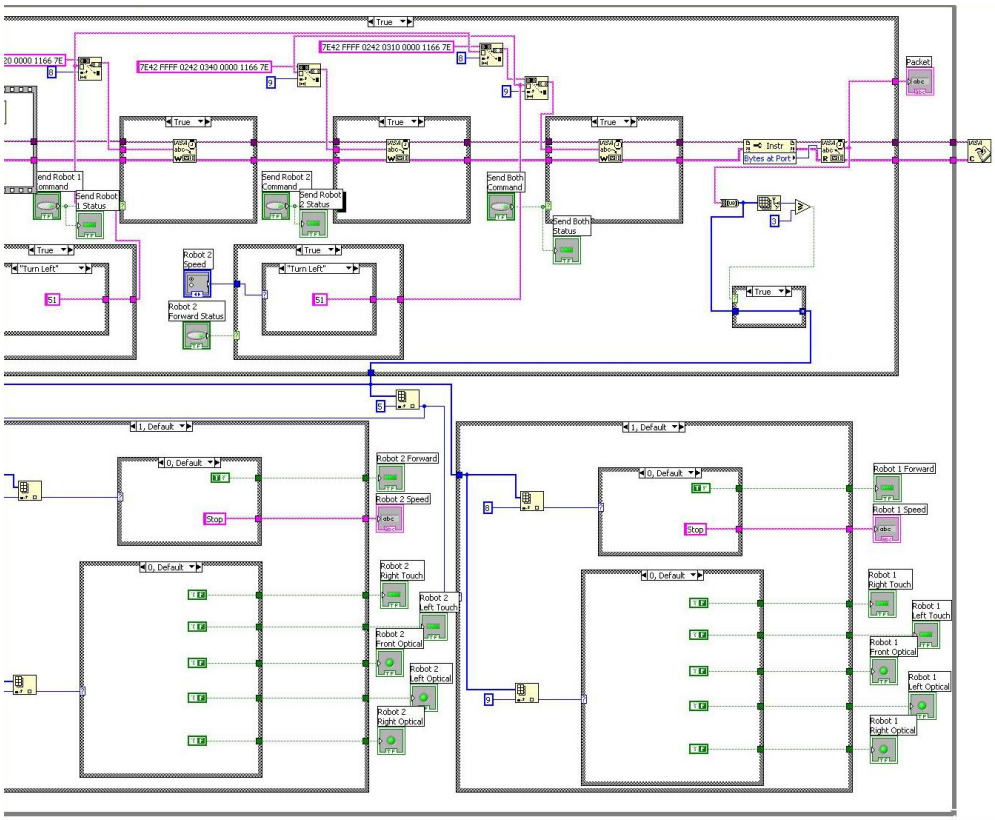

Figure 38: GUI Backend 2

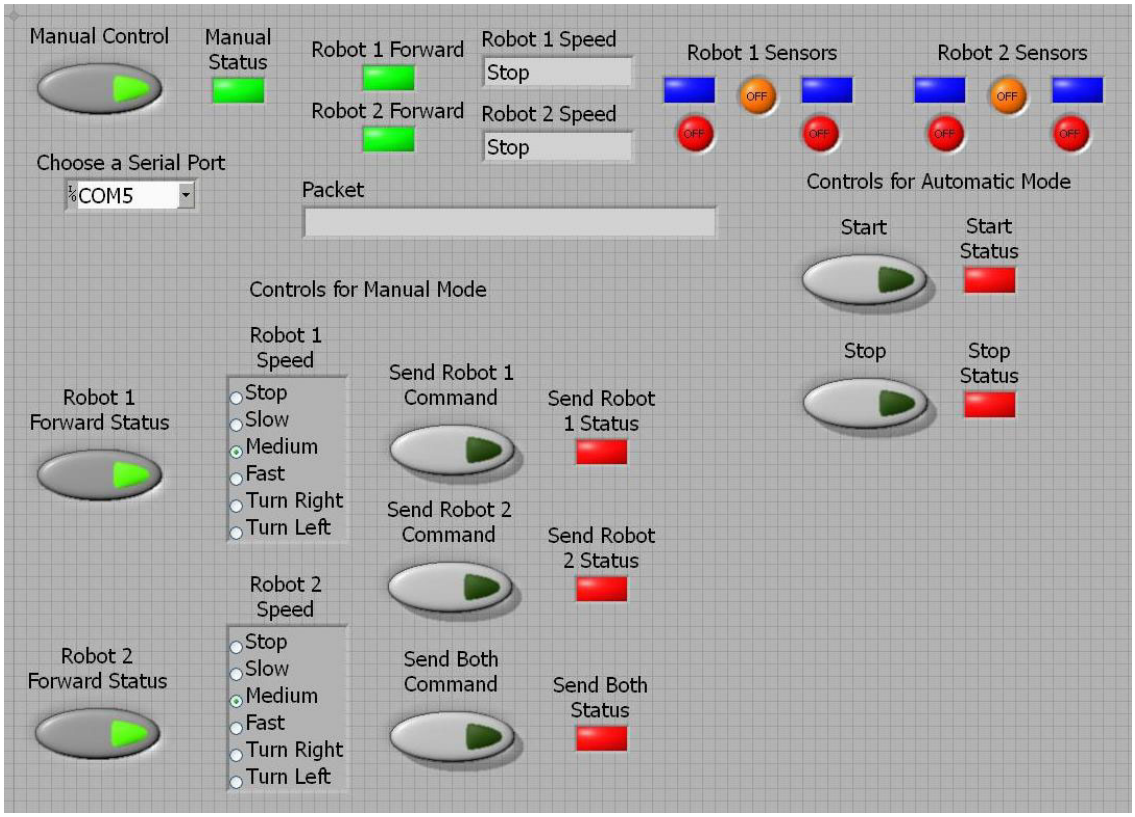

Figure 39: GUI Display

D. Previous Robot Designs

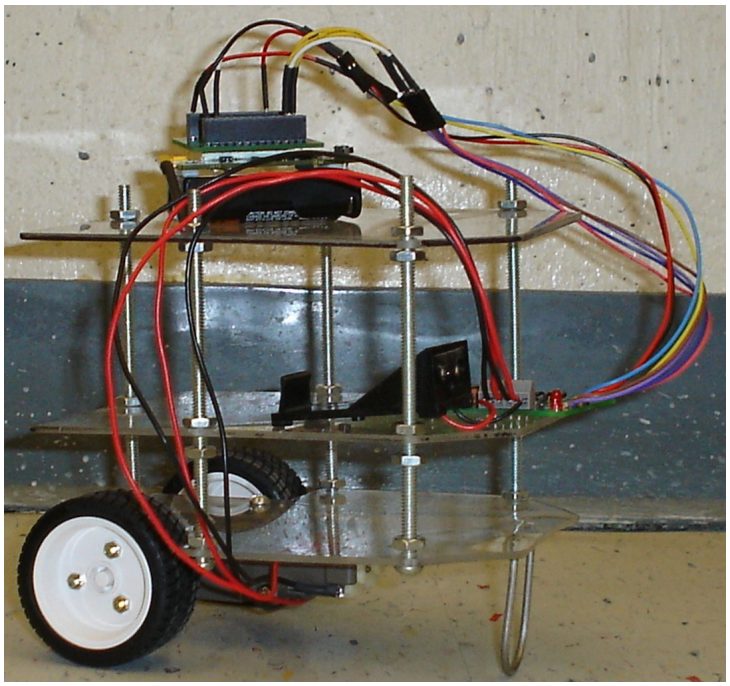

Figure 40: Prototype Third Wheel

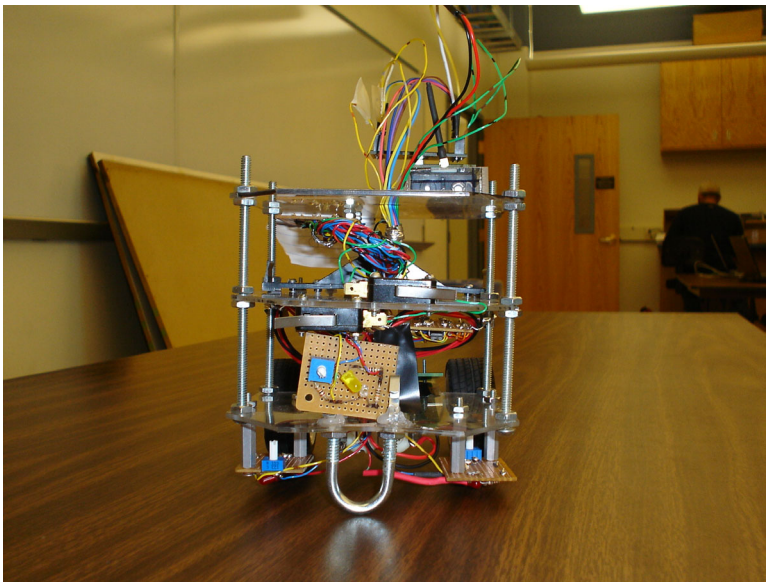

Figure 41: U-Bolt Third Wheel

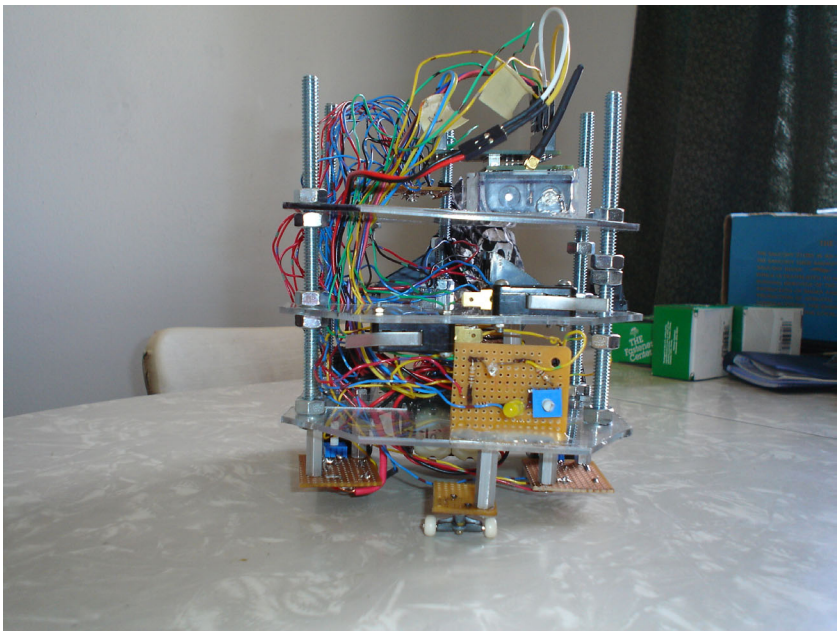

Figure 42: Skateboard Wheels Third Wheel

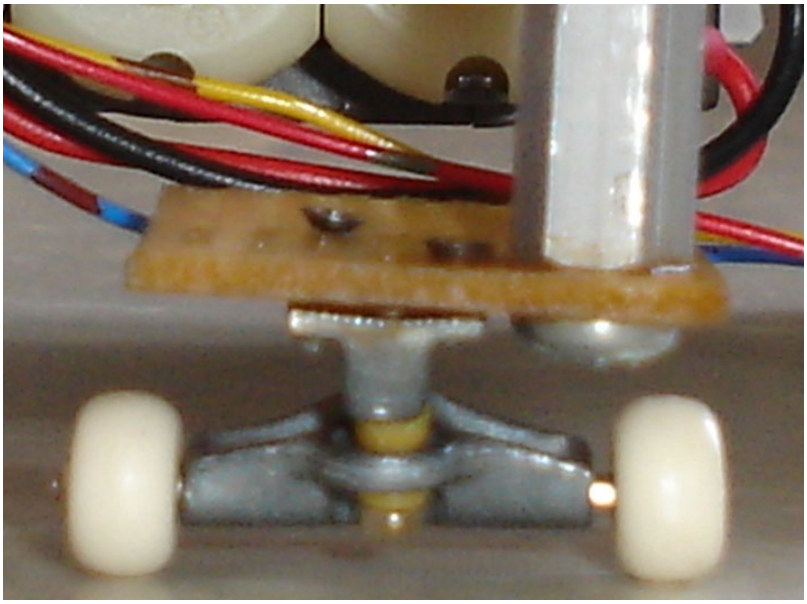

Figure 43: Skateboard Wheels Close up

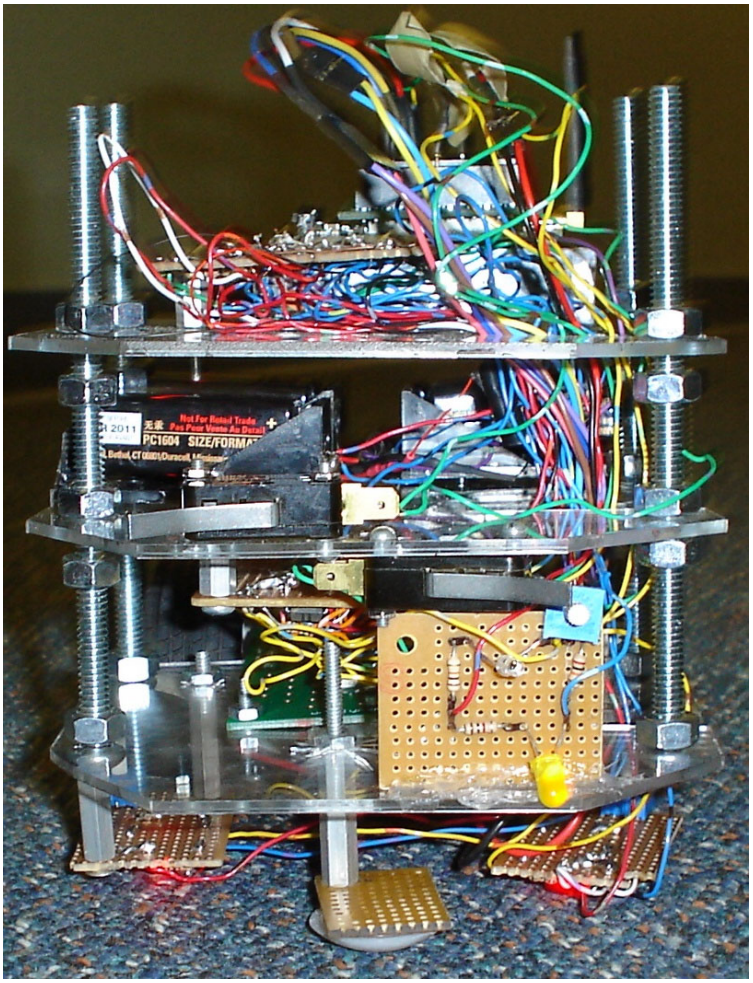

Figure 44: Final Robot

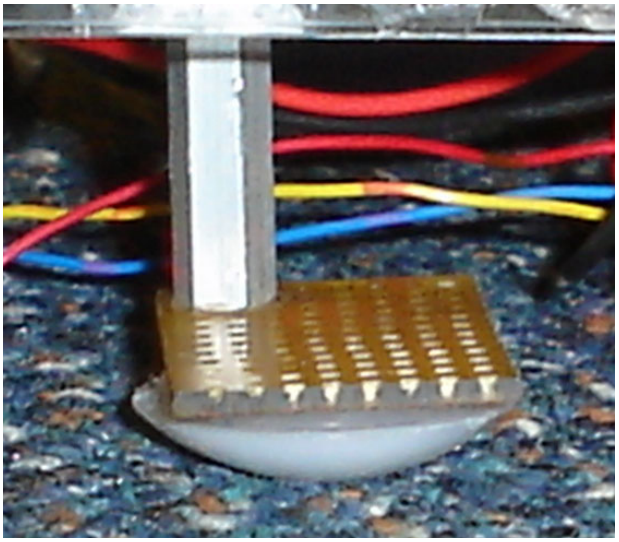

Figure 45: Final Third Wheel Close up

VIII. References

Atmel Microcontroller Documentation:
http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf

General MICAz, H-Bridge, Motor pictures:
http://www.ee.nmt.edu/~elosery/spring_2007/ee382/introduction2007.pdf

MICAz Datasheet
http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf

nesC Language Reference:
http://nescc.sourceforge.net/papers/nesc-ref.pdf

Tiny OS Documentation:
http://www.tinyos.net/dist-1.1.0/snapshot-1.1.1Nov2003cvs/doc/