Cooperative Multi-Robot Box Pushing

New Mexico Tech
EE 382: Introduction to Design
Team 2
Spring 2007

Prepared for:
Dr. Aly El-Osery
Dr. Rene Arechiga

Prepared by:
Joseph Fernandez
John Gallegos
Jeffrey Godman
Alex Macdonell
Eric Martinez
Manuel Rivera

Abstract

The purpose of this report is to detail our investigation of multiple robot systems. Multiple robot systems are useful for performing tasks that would be difficult for a single robot to accomplish. In this case, the task is to coordinate two robots in pushing an elongated box through a pre-defined lane without leaving the boundaries. Communication must be employed between the two robots to overcome problems that occur while pushing the box through the lane. The two robots are completely autonomous and communicate wirelessly. The project is meant to demonstrate the usefulness of coordinated behavior in other important applications. The practical experience gained by the team relates to communication, task allocation, and learning.

Key terms: autonomous robots, coordinated behavior, design project, proportional control, wireless communication

Table of Contents

Table of Figures

1. Introduction

Coordinated behavior is a field of robotics in which the coordination of multiple systems achieves solutions to a given task. The primary advantage of such systems is the ability to perform or design tasks that single-agents would have difficulty accomplishing. This project coordinated behavior to move a box through a lane marked on the floor.

1.1 Junior Design Task

Our task, as specified by our instructors, is to design and build two autonomous robots which will push a hollow cardboard box down a lane. The box is four inches tall by four inches deep, and it is one meter wide. The lane is flat and measures two meters wide and four meters long. The surface of the lane is composed of a white base and boundaries marked by black electrical tape. Each robot must be no more than six inches wide by six inches tall and six inches deep. Only a three inch portion of each robot is allowed to touch the box.

The box is placed perpendicular to the sides of the lane. The robots will be placed offset from one another twelve to twenty inches behind the box. With one on each side of the box, they will also be orientated so their projected path points towards the box. Each robot may be placed on the left or right side of the box, so their relative position is not predetermined. **Figure 1** shows one possible orientation of the starting position of the robots.

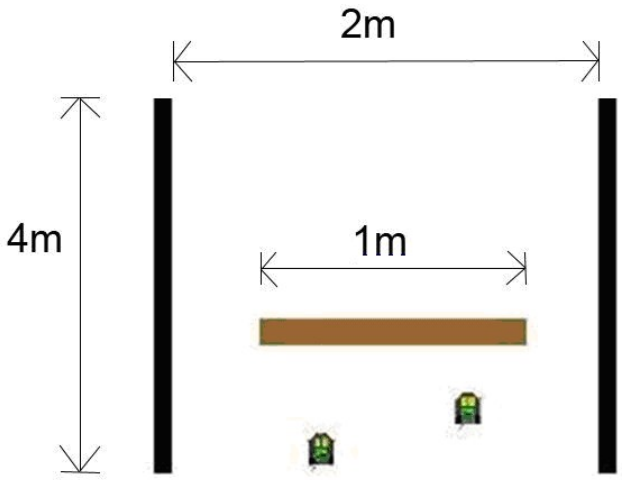


Figure 1. Initial position of robots relative to the lane and box. Actual distance away from box will vary between 12-20 inches.

We were given some hardware, software, and a $150 budget to complete the task. The most essential hardware component is the MicaZ microcontroller, manufactured by Crossbow. The microcontroller enables wireless communications between each robot and a base station computer. Each MicaZ can send and receive packets of information and can send and receive data through a serial port of a computer using a third MicaZ transceiver. The MicaZ also comes equipped with software that allows custom programming. The software program allows us to utilize the MicaZ's analog-to-digital converters, digital input and output pins, input capture, light emitting diodes, and pulse width modulation channels. Using this software, we programmed our control algorithm into each robot's MicaZ and the base station MicaZ.

We were also given two gearboxes with motors and wheels. Each gearbox has two completely independent gear sets and motors that drive each of the two wheels independently. Finally, we were provided with two H-bridge controllers that are capable of translating the MicaZ output signals to control the speed and direction of each motor.

We were tasked with designing and fabricating the rest of robot including the chassis, sensors, and electronics to be placed on the robot. In addition, we were to design a control algorithm to implement our solution. Finally, we were to make a graphical user interface (GUI) on our base station computer that would display the status of the box pushing operation and be capable of starting and stopping the control algorithm.

2. Research

We researched multi-robot systems to get an idea of their purpose and how they should be operated. We also researched different control systems to help us choose the best method for implementing our project. We researched high and low level control systems, centralized and decentralized control systems, open and closed loop systems, and proportional, integral, and differential control systems.

2.1 Multi-Robot Systems

The project has been attempted several times before, and previous attempts have yielded useful results. Mataric, Nilsson, and Simsarin [1] studied different strategies for moving a box to a set destination. They found that a two robot team succeeded not only more often than a single robot, but also was able to accomplish the task faster. In addition, they mentioned that the robots were more effective in accomplishing the task when they communicated with each other. Clearly, a multi-robot system is a good solution for the problem of pushing a large object, and is favored over a single robot solution.

Multi-robot systems are good for several other reasons. According to Aparicio and Lima [2], multi-robot systems offer robustness and adaptability not found in single robot systems. If one agent in the system is damaged or malfunctions, the task can still be accomplished with the remaining agents. This idea can be applied to maintenance as well; agents can be serviced one at a time, which results in the system having no downtime, assuming the other robots are capable of continuing the task with the absence of their counterpart. In addition, multiple robots are able to cover more ground and can specialize in doing smaller tasks that make progress towards the completion of the larger task [2]. Therefore, multi-robot systems are more likely to reliably complete large and complex tasks.

Despite the many advantages of multi-robot systems, there are a number of disadvantages that must be addressed. First of all, communication between the agents in a system is difficult computationally. According to Koes, Nourbakhsh, and Sycara [3], when multi-robot systems are making a decision, they must consider the time to travel to a location, the time to wait for other robots to arrive, and the time to complete the task. Therefore, control algorithms are very complex, as efficiency is harder to realize with more agents working on a particular problem. Radio communication adds another level of complexity to the system as transmitting circuits and communication protocols must be utilized. Therefore, while a multi-robot system may be more effective, the system requires a greater level of depth and complexity than its single-robot counterpart.

During the research process, applications were discovered which directly relate to our project. One particular example involved two larger robots used to move furniture to

specific locations. In a more practical example, robots could be used to move heavy materials at a construction site. It is likely that robots would help considerably with such a task and might increase efficiency, but robots are usually too expensive to be beneficial financially.

McLurkin proposes an application which uses coordinated robots to explore areas that may be dangerous for humans. These robots can autonomously increase the safety of make a hazardous areas. For example, an area filled with mines can be suitable for robots that are equipped with sensors designed for mine detection. The robots, once released into a minefield, would be able to detect mines without any human interaction. Once a mine is detected, the robot could communicate the location of the mine with the other robots. The other robots, using coordination algorithms and position sensors, would be able to swarm to the mine to help dismantle or mark it. The robots may talk to each other, or they may use a main robot that gives each of them directions [4]. A simple project like coordinated robotic box-pushing serves as a stepping stone to creating a more complicated and useful system, such as robotic mine-seekers.

2.2 Centralized vs. Decentralized

We decided to use a centralized control system for the high level design. Centralized control allows the base station computer to control the two robots, whereas decentralized control allows each robot to control itself without interaction from the base station [5]. Our high level control consists of directing the initial conditions and basic control of each robot using the base station. The base station will tell each robot when to go forward, stop, and turn. Also, it will send initial conditions to each robot including initial motor speeds. Our high level design will use a closed-loop control system.

2.3 Closed-Loop Control System

A closed-loop design uses a feedback control system. Feedback allows a system to determine current and future responses of a system based on present or past information. This is beneficial because it increases the accuracy and reliability of the output since the system can adjust itself to obtain the desired output. However, the feedback process makes this system more costly and complex than its open-loop counterpart [6]. An open-

loop system can only make adjustments based on pre-programmed settings and cannot account for environmental or situational changes.

We decided to implement a closed-loop control system for our robots. The position of the box and robots will change from run to run, so it is necessary to have feedback to determine the position of the box with relation to the robots and the lane. Moreover, the robots do not drive straight when all motors are set to travel at the same speed due to inherent differences in the mechanical construction of each motor. Therefore, it is critical to gather feedback data about the speed of the robot in order to make adjustments to make the robots travel in a straight line.

2.4 Proportional Control vs. Integral and Differential Control

A proportional control system uses present conditions to control the robot. Differential control uses the rates of change and error calculations to predict future conditions the system may encounter while integral control uses past states to control present states [7]. While researching each of these low level control systems, we decided that differential and integral systems are too complex and unnecessary for our project. In addition, time constraints have limited the development of our speed sensor feedback system that may have utilized differential or integral control. For these reasons, a proportional control system will utilize the data from other sensor sources for the purposes of our project.

3. Chassis Design

With research and basic design decisions complete, we began development of the actual physical system beginning with the robot chassis. The development of a reliable robot chassis is of great importance. The robot chassis often dictates the possibilities of success or failure in a design project. Options such as size, weight, maneuverability, and mounting options must be carefully considered when deciding on a platform structure.

3.1 Design Foundation and Goals

The specified dimensions of each of the robots were six inches by six inches by six inches, with only three inches of the front of the robot being allowed to contact the obstacle to be pushed. Because of these specified dimensions, a concerted effort in the

early design stages of the system required a compact design that would later allow for expansion. We did not know what physical dimensions would be required for circuitry and sensor components, so the preliminary shape of the primary platform was formulated around the dimensions of the gearbox and wheels. The main plate had to be oriented such that the gearbox could be mounted using small hardware and only two mounting points, while still allowing for the side-protruding wheels of the robot to be unobstructed by the robot platform. A reduced-width section of the rear portion of the plate and a wider main section allowed for a basic shape that would satisfy our early needs. In order to stay within the specifications determined by the professor, a tapered front-edge portion of the main plate set the "touching edge" of the robot to be within the three inch constraint. The basic foundation of the robot is shown in **Figure 2**.

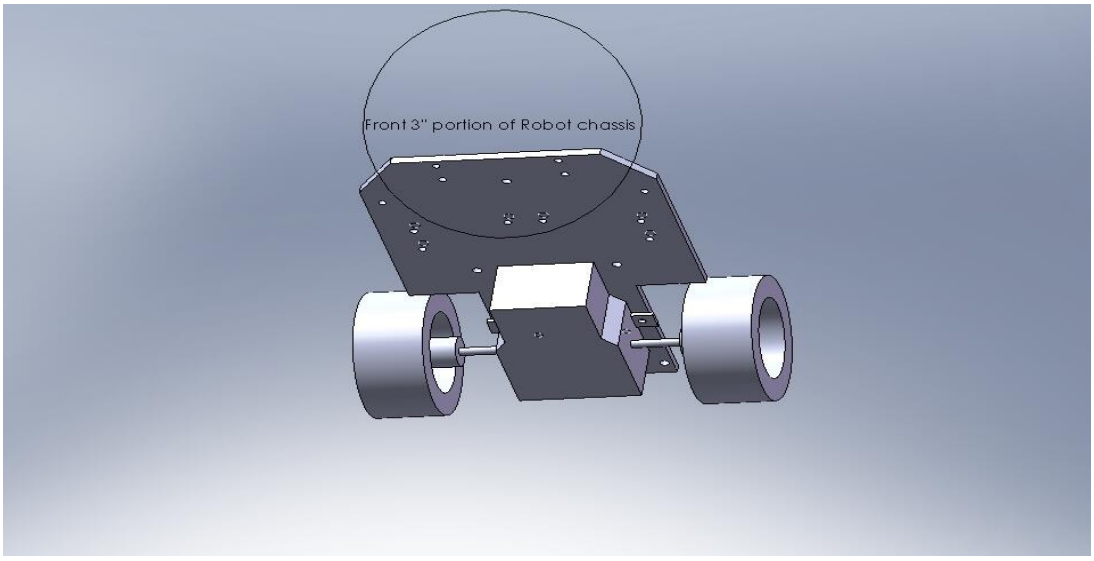


Figure 2. Front-bottom view of basic main-plate and gearbox-with-wheels assembly.

3.2 Fundamental Features

The next major feature to consider for integration into the chassis was the motor-control circuit board (H-bridge). The major consideration that was taken into account when positioning this piece was the fact that the bottom of the board must not contact the conductive main plate of the chassis. This would cause short-circuit issues and effectively would make the circuitry on the board useless. The use of standoffs, small spacers that are designed to separate or stack circuit boards, was an effective method for avoiding this problem. Standoffs not only effectively separate concentrically, vertically

aligned boards, but they also are available off of the shelf and require no additional fabrication effort. The narrow portion at the rear section of the main plate was specifically sized to align with the dimensions of the H-bridge circuit board. Additionally, a second set of standoffs could easily be mounted to the first in order to offset a mounting plate for the Mica-Z controller. This is shown in **Figure 3**. This allowed for the compact design goals that the team was aiming to achieve.

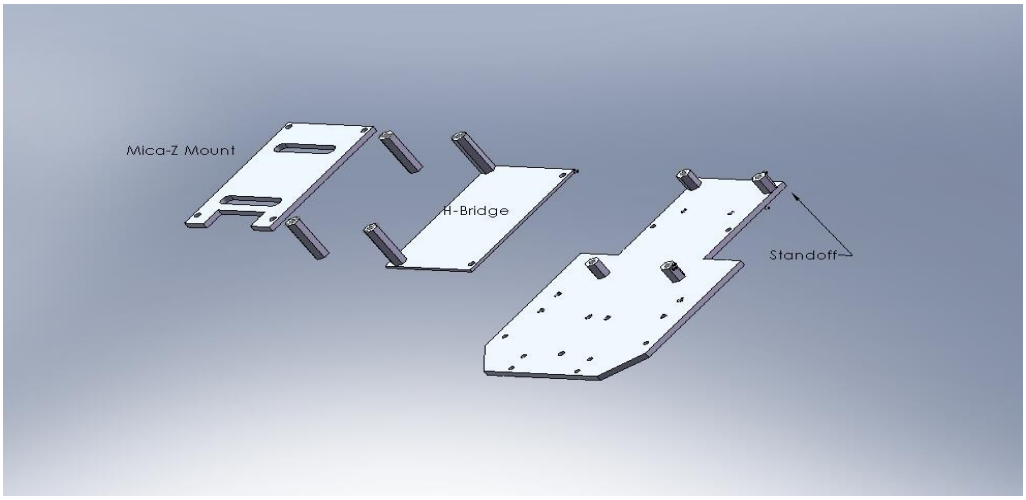


Figure 3. Exploded view of the main plate with the concentric standoffs, the H-bridge block representation, and the Mica-Z mounting plate. (Note that the standoff system is offset for illustration purposes, and will be mounted directly vertical from the main plate for final assembly).

The main robot chassis now had all of the basic design features to allow for mounting the initial components. The next goal was to allow for mobility of the unit. This required that the robot have a front bottom-support component that would allow the robot to move. The team decided against a front wheel to avoid controllability issues (the front support should not hinder movement in any direction). Because the gearbox is equipped with individually-controllable motors, speed control of each of them would allow for maneuverability. In place of a front wheel, a low-friction stub was chosen as the best front-end support method. It was mounted, bisecting the front three inch portion of the main plate, about a half inch from the front edge. The underside is shown in **Figure 4**. This configuration allowed the robot to slide in a controlled, predictable manner.

The next critical design element of the robot platform was the addition of battery clips. The clips were designed to mount the three 9-volt batteries in a manner that lowers the center of gravity (CG), and does not compromise the rigidity of the overall robot. There

are many different varieties of battery mounts, the size and shape of which were considered by our team. In order to reduce unwanted electrical noise and lower CG, the decision was made to mount the batteries on the underside of the chassis, and use individual mounts to help distribute the weight evenly. C-shaped battery mounts were the best option for our team due to cost, size, and availability considerations. The clips are shown in **Figure 4**.

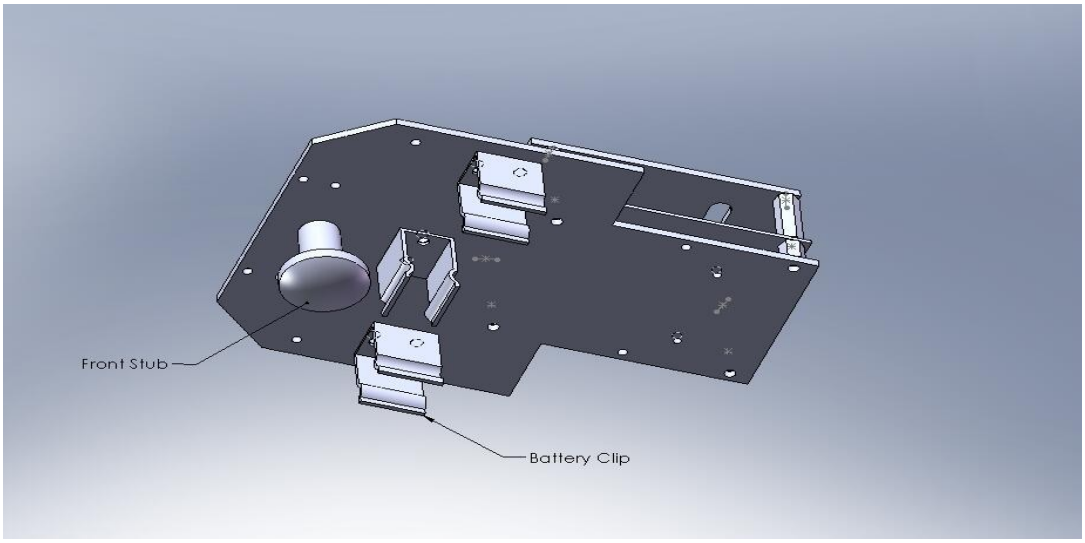


Figure 4. View of chassis underside, including the front stub and battery clips

3.3 Sensor Integration

The touch sensors mounted on the front edge of the main plate detect the presence of the box for this detection. We chose a simple double-pole, single-throw switch activated by depressing an activation arm. Double-pole, single-throw means that the switch can be either one electrical state or another, depending on the position of the activation arm. The activation arm was chosen to be long so that the sensor was sensitive enough to detect the box. Because the box is not perfectly square, we found that the arm got depressed much more easily if it was bent slightly outward; thus increasing its sensitivity.

The switches required custom fabricated L-brackets for successful upright mounting to the front of the robot. These switches are shown in **Figure 5**. The sensors were placed at the maximum distance of the robot's front plate to result in the smallest possible angle of detection

Another important sensor configuration on the robot is the pair of line detectors which change electronic logic levels when detecting a black line on a white background. In order to get maximum response time upon detection, the sensors were mounted at their maximum outward distance from the outside of the robot platform. These sensors also required custom mounting hardware in the form of variable-position L-brackets. They are shown below in **Figure 5**. These brackets allowed us to change the position of the sensor in several ways. We were able to move the sensor forward and back slightly by pivoting it around the single attachment point. We were also able to move it at varying distance from the side of the robot. Finally, the mounting style of the sensor allowed us to move it up and down, at varying distances from the floor. These adjustments allowed us to fine-tune the output of the sensor by changing its physical mounting configuration easily.

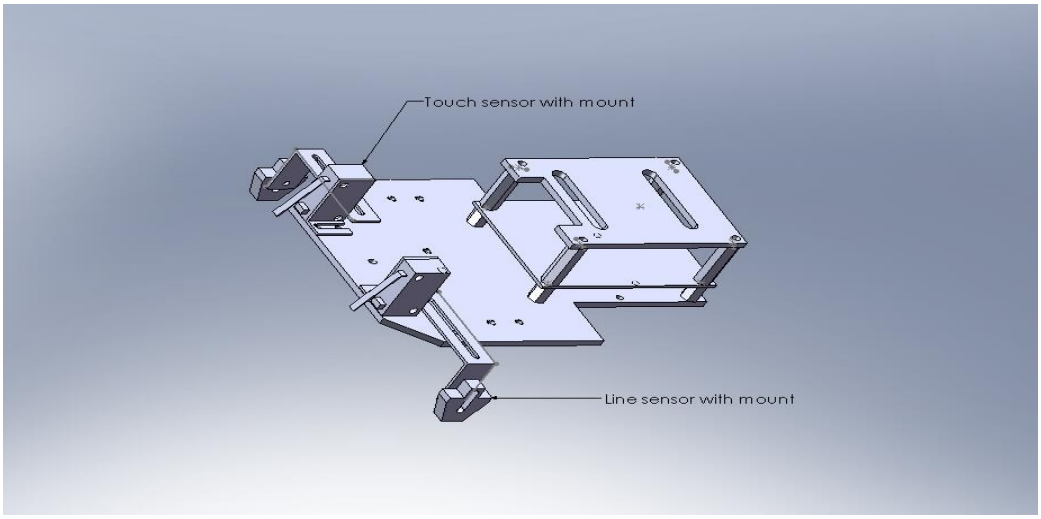


Figure 5. Sensor hardware mounted to the front and sides of the chassis. Both sets of mounting brackets include milled slots, to allow for variation of physical location of the sensors.

Finally, the most complex sensor configuration of the robot can be examined. The optical encoder assembly was designed to measure the speed of each of the wheels of the robot. Since the wheels are individually powered, they exhibit inherent differences in performance. With the same amount of power applied to each motor, we expect the output of each to vary from one another.

By interfacing the wheel speed data with the control algorithm, the power delivered to each wheel can be individually controlled, resulting in a robot that is more likely to

proceed in a straight line when it is required to do so. The sensor system includes an encoder wheel and photo-transistor-light-source pair to measure the rate at which the wheels turn. The encoder wheel is a disk with small, evenly spaced holes inside of its perimeter. The photo-transistor-light-source pair includes a small infrared light source that is directed across a small slotted piece of plastic to the transistor. An electrical signal is interrupted when the light is cut off by the solid portions of the encoder wheel that is mounted inside the slot, around the wheel axle. The optical encoders are modeled in **Figure 6**. This sensor assembly involved cutting of the factory plastic housing of the sensor and the encoder wheel to get them to fit. A custom bracket was fabricated to effectively mount the sensor to avoid interference from the wheel or axle.

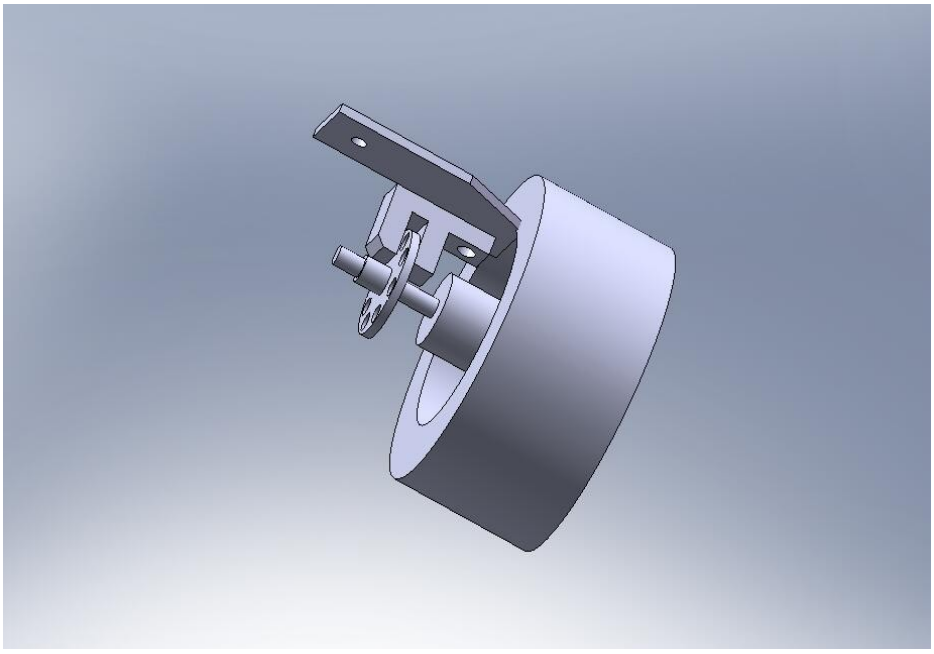


Figure 6. Encoder assembly with wheel and axle. The encoder wheel rotates with rotation of the wheel shaft, and is straddled by the photo-transistor pair.

The final chassis design is shown in **Figure 7**. More information on its fabrication and the drafting techniques used may be found in Appendix A. In addition, detailed dimensions of the robot's various parts can be found there.

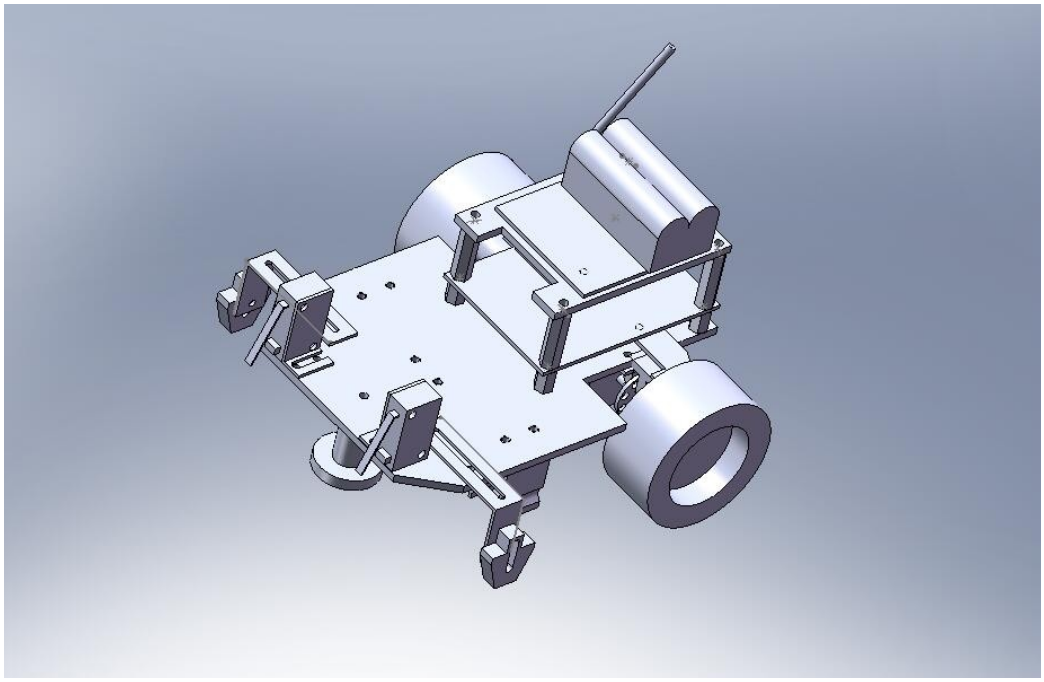


Figure 7. The completed design of the robot.

4. Electronics and Sensor Design

The following section is a detailed description of the design and implementation of all the on-board sensor systems and electronics on the robots. More detailed information, including complete circuit diagrams, part numbers, and pin connections can be found in Appendix B.

4.1 The MicaZ

We were provided with three MicaZ microcontrollers. They utilize a programming language called nesC which is a variation of the programming language. Each microcontroller also has built in communications protocols and hardware which allows for easy communication between MicaZ controllers. We were also provided with an interface board for the MicaZ which allows us to communicate with the MicaZ using our computer's serial port.

The MicaZ is powered by an on-board three volt power supply consisting of two AA batteries. The power provided can also be used by other electronic systems on the robot as long as the circuit does not draw significant current. For example, it would not be

appropriate to use the MicaZ power supply to drive the motors of the robot as the current requirements would quickly drain the batteries of the MicaZ. Therefore, we decided to design two separate power supplies for the higher current circuits of the robot.

4.2 Power supply design

Two power supplies were designed for use on the robot chassis. The first is a five volt supply devoted entirely to powering the motors on the MicaZ. Electrical motors produce a good deal of electrical noise while drawing large amounts of current which results in fluctuations in the voltage of their power supply. In addition, they draw excessively large spikes of current when they switch on and off. This makes for a noisy power supply line. The motors, therefore, were isolated from the other electronics on the board as much as possible, as we did not want this noise to interfere with our logic. Clearly, having a power supply devoted entirely to the motors was the easiest way to do this.

The motor power supply schematic can be seen in **Figure 8**. It consists of two nine volt batteries wired in parallel, so the output voltage is still approximately nine volts, but the maximum current that can be supplied is doubled. The diodes shown are simply there to force the current to flow in just one direction (so that the batteries do not charge each other). We chose rechargeable Nickel Metal Hydride batteries rated at 250 milliamp hours. With both of these batteries working together, they are essentially rated at 500 milliamp hours. The motors require about 400 milliamps if they run at full speed. However, for this project, the motors are rarely run at over half speed, meaning that the battery choice is adequate for the power demands of the robot.

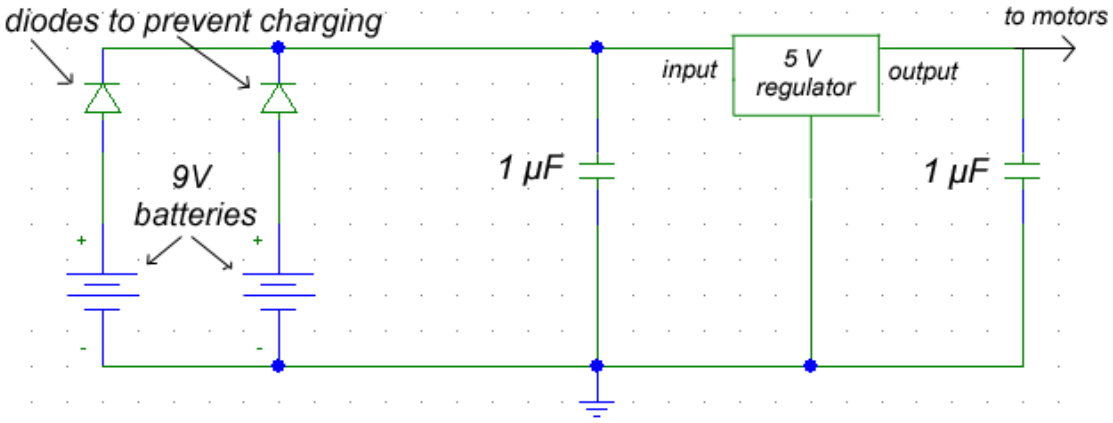


Figure 8. The motor power supply

The two batteries are hooked up to a simple five volt regulator. Because the input is nine volts, it would take considerable current draw for the output of the regulator to fall below five volts. To help add stability to the circuit, capacitors were added at the input and output of the regulator. These capacitors stay charged at nine and five volts, respectively, and help to supply extra current to the circuit in cases where the circuit draws large elements of current in a short period of time. The end result is a smoother power supply line.

Next, we built a separate five volt supply used to power the H-bridge logic circuitry and the line sensors. This circuit is another five volt regulator powered by a single nine volt battery. Because the current requirements are not as high as the motor power supply, a simple off the shelf conventional nine volt battery is used. Again, the input and output employed capacitors to even out noise on the line. The logic power supply is shown in **Figure 9**.

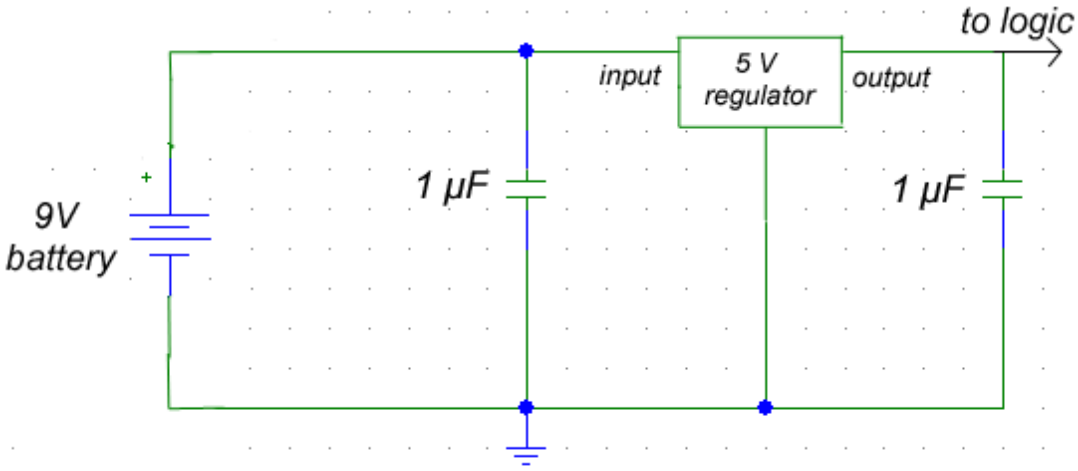


Figure 9. The logic power supply

4.3 H-bridge and Motor Control

The H-bridge is designed to control two motors. The H-bridge receives the logic signals from the MicaZ and uses them to turn the motors on and off. The unit is essentially a relay that connects the logic signals and separate motor power supply as previously described.

First, the motor leads were attached to the H-bridge in the proper configuration. Next, the motor power supply was attached so that the motors would be powered when the H-bridge switches them on. The logic was then powered using the five volt logic power supply.

The H-bridge has several inputs that were then hooked directly to the MicaZ. The MicaZ, as mentioned before, runs on a three volt supply. This means that a high signal coming from the MicaZ is three volts and a low is zero volts. The logic on the H-bridge is five volts, which means that a high is considered to be five volts and a low signal zero volts. At first, we thought that this might cause a problem with high signals from the MicaZ being interpreted as low signals by the H-bridge. After some testing, however, we determined that the high signal from the MicaZ is sufficient to be interpreted as a high by the H-bridge circuitry. Therefore, the logic inputs to the H-bridge were hooked directly to the pin outputs of the MicaZ.

Six signals from the MicaZ were used to control each motor (three inputs for each motor). The A+ and A- leads controlled the direction of motor A (1-0 means that the motor will turn in the forward direction and 0-1 means that it will turn in the reverse direction, whereas if the inputs are the same the motor will not turn). The same configuration is used for the B+ and B- leads for motor B. The A enable and B enable leads are used to turn the motors on and off. Once the direction controls of the motors are set to forward, we left them and only changed the enable lead is of concern since our robots will never need to operate in reverse. The enable lead can be set high or low for full or no power, respectively. However, we opted to connect the enable leads to a pulse width modulated (PWM) channel on the MicaZ, so we can change the speed of each motor in the control algorithm by simply changing the duty cycle of each PWM signal.

4.4 Speed Sensors

Each robot has two wheels driven by two motors and gearboxes none of which are identical. Therefore, equal PWM signals into each motor will not guarantee that the robot will drive in a straight line. It became quite obvious from the beginning of the project that this would be an issue that requires attention. There would have to be some form of feedback to the MicaZ in order to control the robot's straight line motion.

The solution we chose involved implementing encoder wheels on each axle. The encoder wheel chosen is shown in **Figure 10**. This particular encoder wheel was inexpensive and compact which made it a great choice for our robot's chassis. It has eight holes drilled inside of its circumference. A light emitting light detecting sensor assembly was mounted to straddle the wheel. The emitter detector is wired to output a high when the light shines through the encoder wheel and a low when the encoder wheel blocks the light. Through application of a programming algorithm, the speed of the wheel can be derived from this pulse string.

Figure 10. The encoder wheel
Photo courtesy Electronics Express, www.elexp.com

We planned on using this method for speed control. However, we realized that the MicaZ only had one input designed for this type of functionality available which is the input capture channel. We therefore decided to use an XOR logic gate to detect changes from both encoders. By passing the outputs of both encoder wheels into an XOR gate, the output would transition every time either of the inputs transitioned. Therefore, assuming that the two signals will never change at the exact same time, every transition in either signal will be manifested as a transition in the output of the XOR gate. Therefore, the XOR output can be monitored by the MicaZ's input capture protocol to detect all transitions on both encoder wheels. See **Figure 11** for a sample of the XOR gate functions.

To determine which sensor changed when a transition in the XOR output is detected, the state of each sensor will be monitored separately using a separate input pin. The state of each input will be saved continually. When the input capture goes off, the current state

of each sensor will be cross checked with the previous state of each input to determine which input has changed. In this manner, we can detect not only that there was a change, but also which encoder the change came from. Then, the speed of each wheel can be calculated in software, and the PWM signals for each wheel will be modified to get the robot to drive in a straight line.

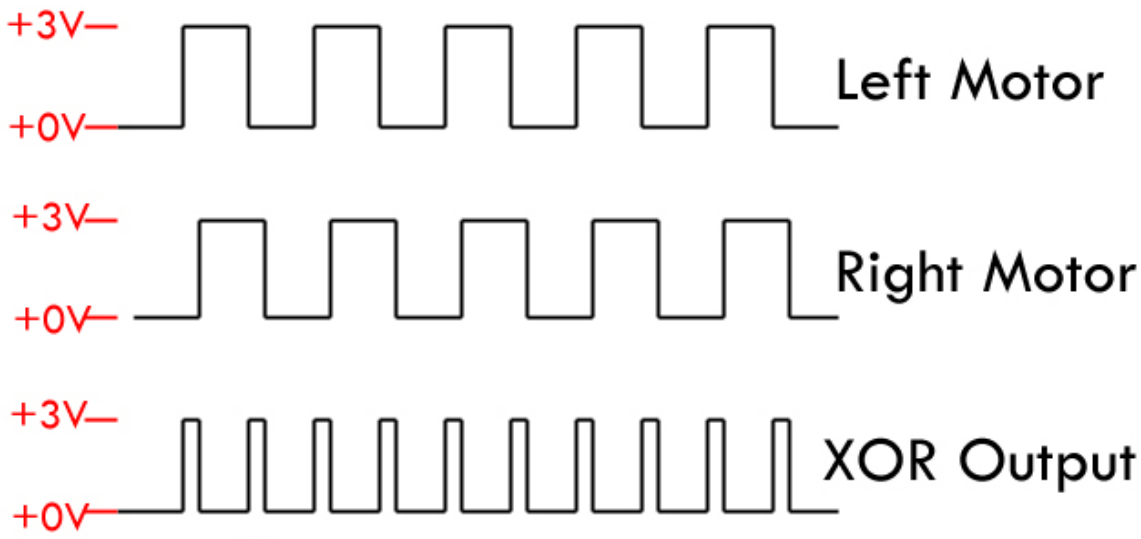


Figure 11. The XOR output of the two encoder signals

4.5 Line Sensors and Circuitry

Since the limits of the lane the robots are traveling down are marked off by black tape on either side, it is critical for each robot to be able to sense when it reaches the line. It may then turn away from the line and head back towards the center of the lane.

Each robot must be capable of sensing a line on either side. To accomplish this, both sides of each robot were equipped with a line sensor. The line sensor is a light emitting and detecting circuit, very similar to the encoder sensor described previously. We used a sensor that came pre-packaged for simplicity. The sensor and its associated circuitry is shown in **Figure 12**. The left part of the circuit feeds a sufficient amount of current from point A to point B through the diode. The sensor points down so that the diode emits infrared light at the floor. This light will reflect differently depending on the color of the floor. It will be absorbed almost completely by a black surface, but it will reflect against

a white surface. The base of the phototransistor (point C) detects the reflected light. If the light is of a sufficient level, the transistor will be properly biased, so current will flow from point D to point E. This will bring the output at the sensor low. If there is no light detected, the current from D to E will be zero and the output of the sensor will be high.

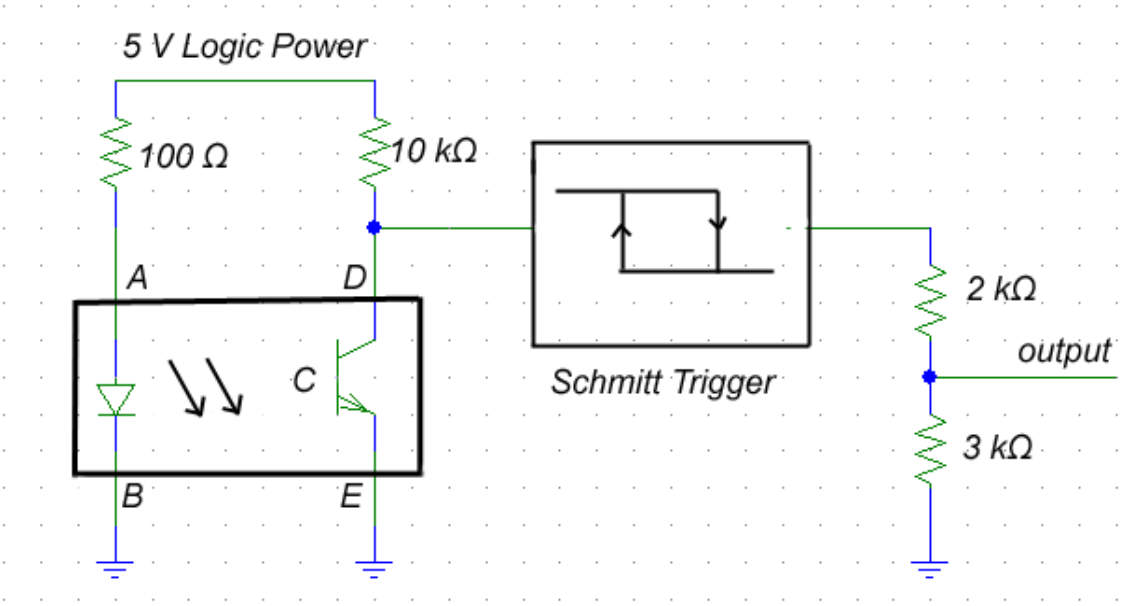


Figure 12. The line sensing circuitry. The boxed portion on the left is the line sensor and its internal components.

The sensor was tested with various supply voltages and at various distances from the floor, and the optimal operation was determined to use a five volt power supply with the sensor approximately half of a centimeter away from the floor. Even with this condition, a high was not a perfect five volts, and a low was not exactly zero volts. Therefore, we decided to run the output of the circuit through a Schmitt trigger. We used a readily available inverting Schmitt trigger chip with the thresholds of the input voltages set by default to approximately 1.9 and 2.9 volts. This was a simple way to translate the output of our sensor to a digital logic signal. At the output of our Schmitt trigger, we used a voltage divider in the ratio of 2 to 3 to step the voltage down to a 0-3 volt scale. In retrospect, we would have been better off implementing a logic chip to convert TTL zero to five volts levels to LVTTL levels of zero to three volts. We also realize that it would have been perfectly acceptable to feed the 0-5 volt output into an input on the MicaZ, as the MicaZ can handle inputs up to approximately seven volts. However, we deemed this

a bit sloppy and decided against it. As a whole, the circuit outputs a logic high if the surface below it is white and a logic low if the surface below it is black.

4.6 Pressure Sensors

Knowing how to push the box is difficult if the robots cannot sense where the box is. Therefore, we decided to install pressure switches on either end of the three inch front face of our robots. This helps not only tell whether the robot is in contact with the box, but also if the box is not flush with the front face of the robot. This situation is shown in **Figure 13**. This allows us to coordination of each robot to orient the box such that it is pushed straight down the lane.

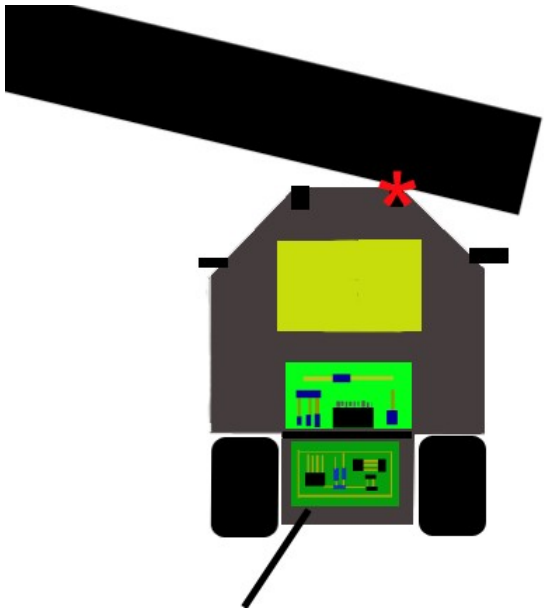


Figure 13. The robot pushing the box unevenly

The pressure switch used was a very simple pushbutton circuit. This diagram is shown in **Figure 14** with the two input leads come directly from the MicaZ – a high (three volts) and a low (zero volts). When the pushbutton is not being pressed, the output lead is a low. The output lead goes high when the button is depressed. The output lead is fed back as an input to the MicaZ, where it can be monitored.

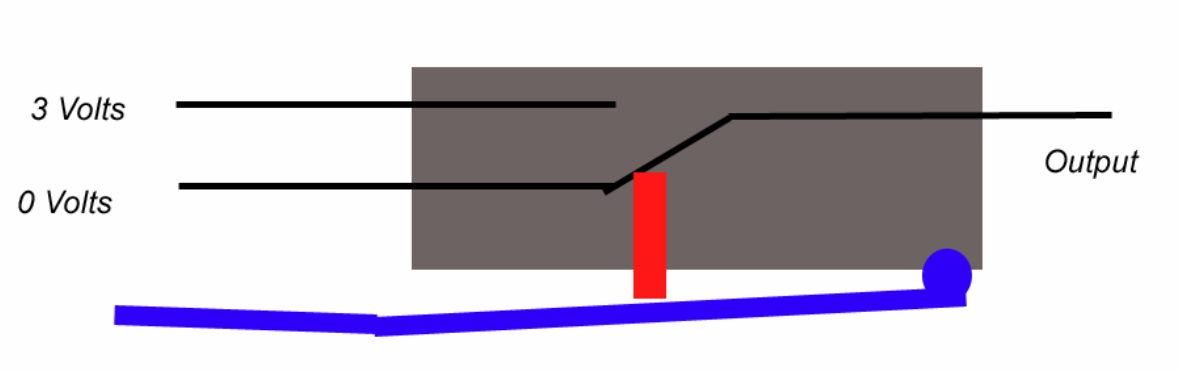


Figure 14. The pushbutton pressure switch used. The blue part is the lever arm that made contact with the box when the robot pushed it. The switch itself is normally low and switches to high when pressed.

We were not sure if the switches we obtained were debounced internally so they were tested using a logic analyzer. We observed minimal bouncing when the button was pressed or released. Therefore, we decided to put a switch debouncer on our switches. This circuit eliminated the bouncing behavior, resulting in a very smooth transition from one state to the other. The exact circuit used for the debounced circuit may be found in Appendix B.

5. Printed Circuit Board

A printed circuit board (PCB) was constructed as a central location for the electronics systems of the robot. For this project, it was not required to construct a PCB. However, doing so was beneficial. Compared to a prototyping board, a PCB saves space and is much more reliable and organized, which results in a much more robust system overall. We also decided to use pins and wiring harnesses for all connections on the board which made removal of the board for modification and maintenance more convenient. The PCB can be connected to all 33 leads in a matter of seconds using these harness assemblies.

Initially, we built our robots using all of the electronics attached to a breadboard. This became cumbersome quickly, so prototype boards were implemented. The robots functioned correctly with the prototype board approach, but were very messy and extremely difficult to troubleshoot. We therefore decided to etch PCBs for each robot. **Figure 15** shows the dramatic difference the PCB had on our robot. At the left is a robot with prototype boards, and at the right is a robot which uses a PCB and wiring harnesses.

The robot with the PCB was much neater and much easier to troubleshoot. Wire length was considered carefully to ensure that the robot was as neat as possible.

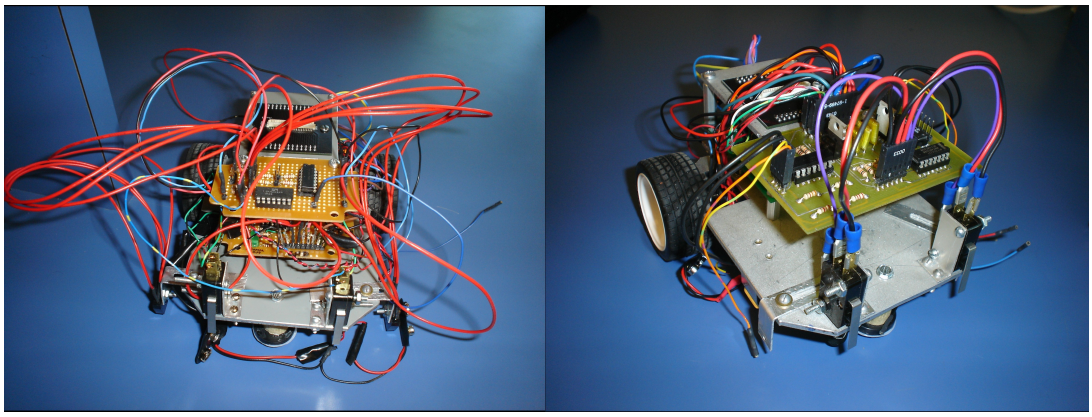


Figure 15. At left, a robot with a prototype board, and at right a robot with a printed circuit board.

We used a free online PCB design and routing software called Eagle. This program is intended for small and simple circuit designs. For our purposes, it was an excellent choice.

5.1 Designing the PCB

In the Eagle software, we first placed all of the footprints of the components onto the board design. The footprints represent the physical dimensions of the circuit elements and how they are to be soldered onto the board. The next step was to symbolically wire all of the components together. This lets the program know what components are connected together. When this was done, we have what is commonly called a rat's nest such as the one shown in **Figure 16**.

Next, the actual wire traces had to be mapped. This can be done automatically by the software program, but it is typically prone to errors. Instead, we decided to route the traces manually using Eagle. This ensured proper clearance between the traces to avoid potential shorting issues. In addition, we were able to design the board to easily interface with the rest of our electronics. The output pins and wiring harness locations were strategically placed on the board in a manner that would minimize wire length on the robot.

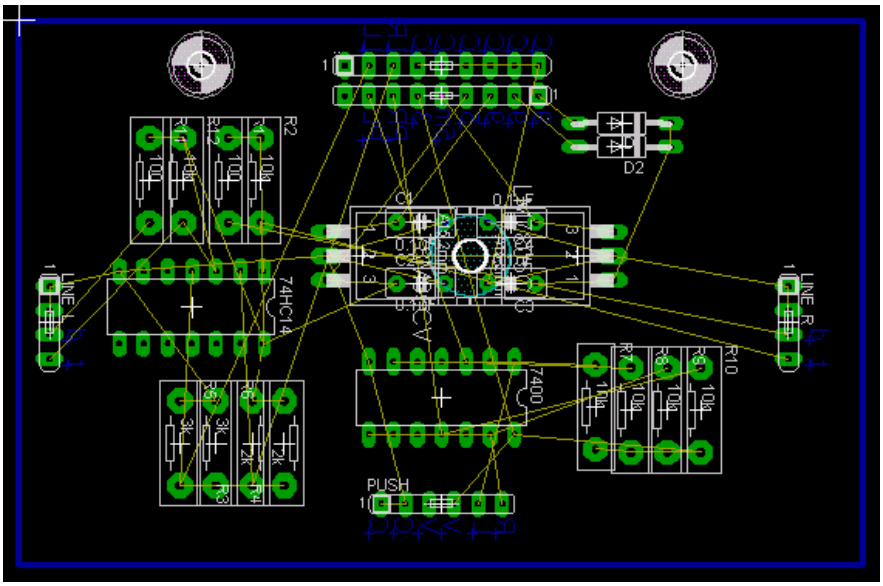


Figure 16. The footprints of all circuit components and the rat's nest of connections.

As we began our board, it was clear that we would not be able to effectively route all of the traces. Because of the board's limited space relative to the number and size of components on the board, several traces could not fit into the system. Therefore, we decided to etch a double sided board with traces on both sides. When a trace could not be continued due to intersection of its path with components or other paths, the path would be continued on the other side of the board. The two paths could then be connected using conductive paths through the board which are commonly referred to as vias. A double sided board is more difficult to implement because it is difficult to line up both sides of the layout by eye. The two layers have to match to hundredths of an inch for proper function. However, double sided boards have several noteworthy advantages. First of all, a double sided board allows more efficient use of limited space. Second of all, it is generally easier to design than a single sided board, as component placement is not as important as it is for a single sided board. A common technique in routing a double sided board is to route all of the ground wires on one side and then route the rest on the other side. This makes it easier to organize the routing and reduces the chances of a short from ground to the voltage source. The final board design can be seen in **Figure 17**.

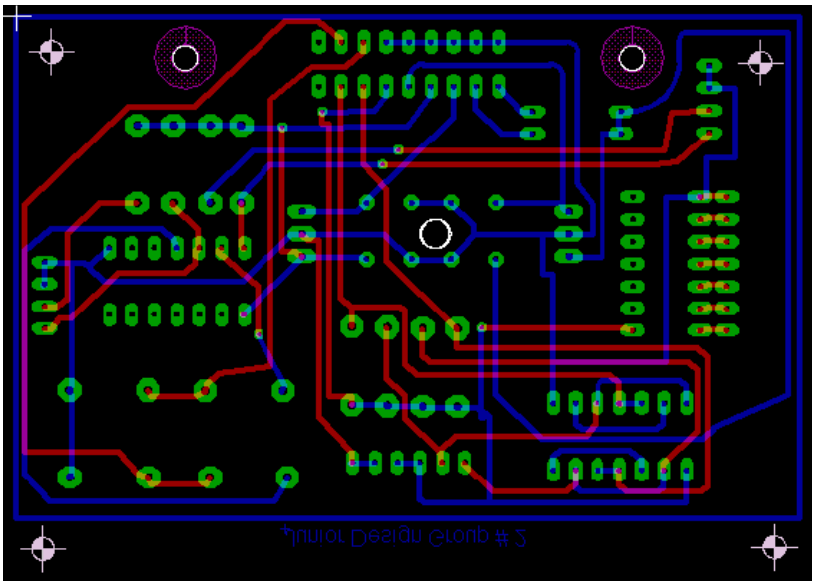


Figure 17. The final design of the board. The red traces are on the top and the blue traces are on the bottom of the board.

Finally, the board was fabricated and all of the components were soldered on. The board was drilled to allow it to be mounted to the chassis, directly next to and in front of the MicaZ. The finished product is pictured in **Figure 18.**

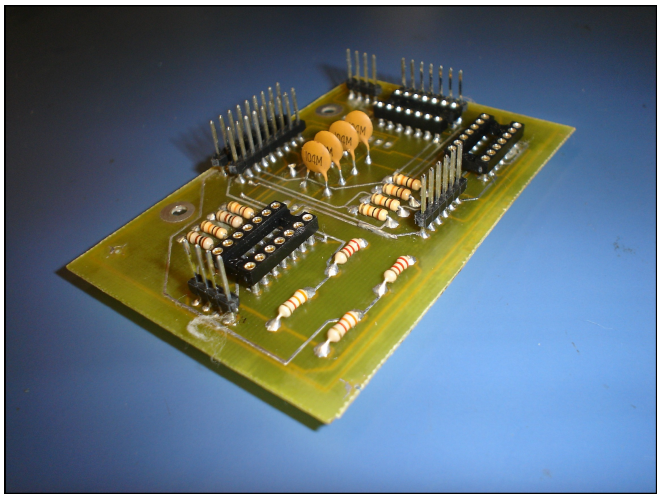


Figure 18. The final manufactured PCB.

6. Communications

Communication is a critical component of this project because cooperation between the robots is essential to the solution of our problem. We chose to implement our communications design around a centralized scheme in which the robots

communicate directly with a base station and never with each other. This scheme was chosen to make programming the control algorithm easier, as it resides completely on the base station and depends on the robots only to send it sensor data. Also, a centralized scheme allows for an easier to program and more useful graphical user interface (GUI), as all of the important information is routed to the base station where it can then be sent directly to the GUI for display. Also, any user changes made in the GUI can be relayed directly to the base station where they can be immediately incorporated into the control algorithm.

6.1 Packets for the MicaZ

Communications between the robots and the base station is accomplished using the 2.4 GHz wireless transmitter/receiver built into the MicaZ's. Packets were framed in the program for each robot and the base station using a structure that contained each piece of data required for that particular transmission. For example, the structure that defined the packets going from the robots to the base station contained variables to hold the touch sensor values, the line sensor values, the current pulse width modulation (PWM) values for the robots left and right motors, and the number of holes counted in the optical encoder assembly. The structure that defined the packets going from the base station to each robot, however, only contained variables for new PWM values for the robot and a variable to tell the robot to start or stop. The packets that were sent serially to the GUI were the largest, with variables for both robots' sensor data as well as both robots' PWM values.

In order to distinguish between the robots and the base station so that each receives only the packets meant for them, a special value in the packet called the Type ID is used. Each robot, the base station, and the computer running the GUI had its own Type ID to distinguish it from the others. For example, packets sent from the base station to Robot 1 would not be received by Robot 2, as packets meant for Robot 1 contained the Type ID for Robot 1 and not for Robot 2. Though there are other packet filtering methods available, using the Type ID worked without failure and was simple to accomplish in code.

One drawback to using the centralized communications scheme was the load put on the base station MicaZ. Because the base station was forced to receive data from both robots and the GUI as well as being forced to send data to all three of these places, the relative speed of packet receipt and dispatch could have been better. Because of this, the robots would not react instantly to changes in their sensors since they would have to wait for the base station to respond. This occasionally caused issues with a robot pushing the box when it was supposed to be stopped or not being activated in time with the other robot. This problem could have been avoided using a partially centralized and partially decentralized communications scheme. In this scheme, some of the sensor data could have been relayed between the robots for decision making while some of the data could have gone to the base station for decision making. This would have allowed for the fastest packet processing times at each node, and consequently the fastest response times for the robots to their own sensors. This method, though superior in speed, would have been significantly more complicated to program and debug, possibly causing more serious problems than the lag that the centralized scheme imposed. For this reason, we used the centralized communications scheme exclusively.

6.2 Robot Communication

The final setup for the communications code, done in the nesC programming language, is simple and straightforward. Each robot has a single receive function and a single send function, coupled with a central module that contains a continuously firing timer. The robots receive PWM values from the base station and bring them into their central module, where the values are applied on the next timer tick. Also on the timer tick, the sensor data from each robot's sensors are brought into the central module and put into a structure. This structure is then pushed to the send module, where it is sent back to the base station for processing. The base station contains three receive functions, three send functions, and a central module. The receive functions gather data from both robots and the GUI and relay them to the central module. In the central module, the control algorithm is run each timer tick and processes the data the base station has received. After decisions are made in the algorithm, new PWM values are pushed to the send functions to be sent to each robot, and all of the sensor data that the base station

initially received is sent back to the GUI for display. This is repeated continuously as the timer fires in the central module, unless the whole operation is aborted with a special value that can be set in the GUI. Base PWM values can also be set explicitly in the GUI for each robot, but ultimately, the touch and line sensors on the robots have control over what the PWM values for the robots are at any one time.

7. Graphical User Interface

We used MATLAB to create the graphical user interface for our system. According to the requirements of the project, the GUI must have a start button, and it must display the communications sent by each robot and the base station. The start button will begin the control algorithm. The GUI must display the raw packets received as well as a version of those packets that have been formatted in an easy to understand manner.

Our GUI is used to send out an initial PWM value to each motor. This will start the robot sequence, and it will begin the autonomous control of the robots. Our GUI will then display all of the data transferred between the robots and the base station such including each motor's PWM, the speed of the robots, and the sensor data. Our GUI display is shown in **Figure 19**.

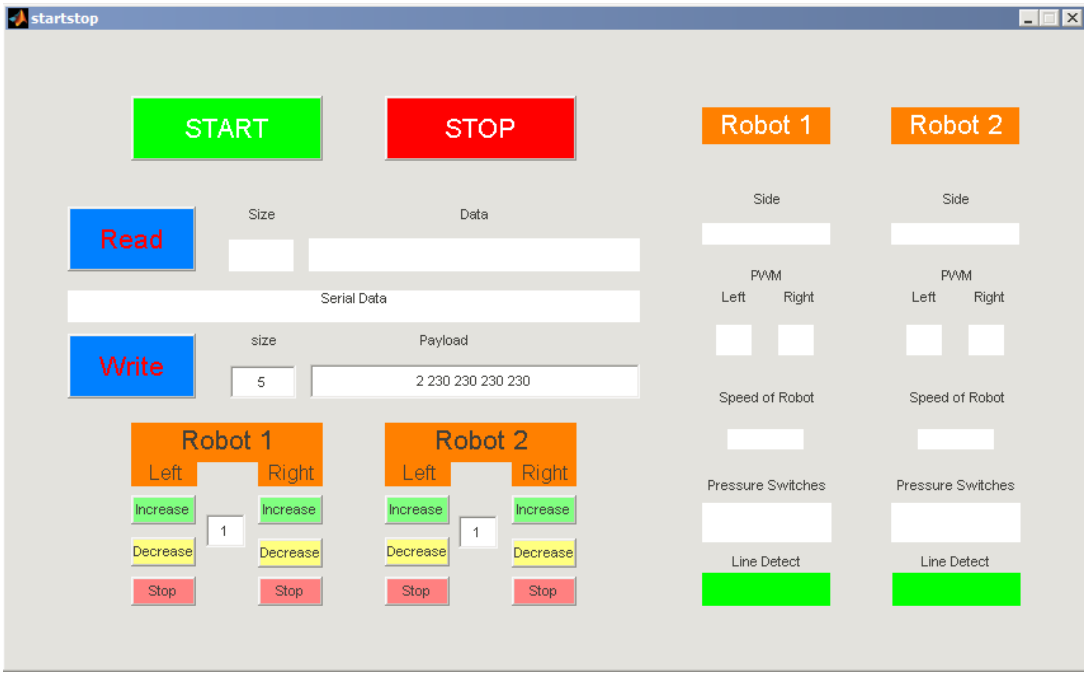


Figure 19. Picture of our graphical user interface.

For testing purposes robots, we can adjust the PWM of each motor with ease by using the buttons in the bottom left corner and the “Write” button. The “Read” button reads in serial data from the base station and displays the raw data in the white bar titled “Serial Data” and “Data”. Also, this button begins the sequence that will interpret the raw data and display it on the right side of the GUI. For each robot, the GUI will display the following: the position of the robot in relation to the other, left and right motor PWM, speed of the robot, pressure switch data on each side of the robot, and line sensor data.

The “Write” button will send a packet of five bytes. The first through last numbers are: start the PWM, robot 1 left motor PWM, robot 1 right motor PWM, robot 2 left motor PWM, robot 2 right motor PWM. The packet is sent through the USB port on a computer to the base station MicaZ. The base station MicaZ will in turn start our control algorithm, which is explained in the next section. The “Write” and “Read” MATLAB programs can be found on the attached CD.

8. Control Algorithm

The base station MicaZ controls our robots. It will read in data coming from each robot, then make decisions for each wheel on each robot. If a line is detected, it will initiate a function to turn the robots to push the box towards the center of the lane.

We were going to use the optical encoders to tell us the speed of the robot or the distance a robot has moved. Unfortunately, we were receiving the random values for both the speed and distance, so we decided not to use this data. We might have been able to fix this problem if we had more time.

8.1 High Level Control System

To start the task, each robot will move forward until it reaches the box. Since the robots will start in a staggered formation initially, they will not reach the box at the same time if they have similar speeds. Therefore, the robot that reaches the box first will stop and wait for the next robot to reach the box. Both robots will move slowly to the box to make sure it barely moves the box once it reaches it. Next, the robots will align themselves. This setup is shown in **Figure 20**.

If a robot hits the box and only one pressure switch is pressed, it will turn toward that switch until the other switch is pressed. This way, the robots will align themselves with the box to make them face straight down the lane. Next, they will move together down the lane using the initial PWM values.

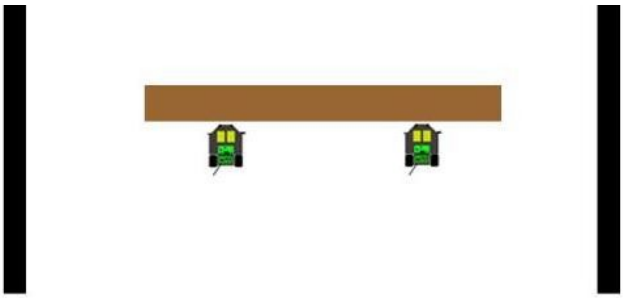


Figure 20. After each robot has moved forward and detected the box. After this, the robots should go straight forward at the same speed.

With the robots lined up with the box and one another, the base station computer will tell the robots to go forward. They will go straight forward until a pressure sensor is released. If a switch is released, the robot will turn towards the box until both switches are pressed again. To do this, the robot will increase the PWM value on the side of the robot not touching the box. Therefore, it will align with the box to keep pushing it down the lane. Unfortunately, this may cause the box to turn which will send the box and the robots towards the side of the lane. A line sensor control algorithm can then be implemented to fix this problem.

8.2 Line Detection

We can detect the sides of the lane for when the robots travel too far to one side. This situation is shown in **Figure 21**. Unfortunately, we do not know how far the robots have traveled because our optical encoder data was useless. Therefore, we must use the robots to turn the box to face towards the middle of the lane. We hope to avoid the robots reaching the edge of the land, but we must design for this occurrence regardless.

Figure 21. A robot detecting a side of the lane. This can happen on either side. If this situation occurs, our control algorithm will use integral control to turn the box to the center of the end of the lane.

Once a robot detects the black tape, it will initiate our line function. The robot will make a wide turn towards the middle while the inner robot will make a sharp turn. This way, the robots will turn the box towards the middle. Initially, we have the robots turning the box for five seconds. Hopefully, this is enough time to turn the box. If we turn the box too much, the robots will eventually detect the other side, and then the robots will repeat the process. We will change the five second turn if it is too much or too less, and this can be determined through testing.

9. Results

We have successfully completed construction of both robots. The hardware is assembled, and our chassis has remained robust. All of our sensor systems have been mounted and their functionality tested. However, we experienced some difficulty with the line sensor systems. At first, both line sensors on robot 2 were working. Robot 1, on the other hand, was returning invalid data from the line sensors. After diagnosing the board, it was determined that the problem was with a voltage divider at the output of the Schmitt trigger. For some reason, the trace leading from the divider was indicating a DC offset voltage. To fix the problem, we eliminated the voltage divider and fed the output of the Schmitt trigger (0-5 volts) directly into the MicaZ. This should not cause problems with the MicaZ, because it should be able to handle inputs up to 7 volts. This configuration seemed to solve our immediate problem. However, we continued to have wavering reliability with both robots' line sensors during the demonstration.

We also had other issues with our printed circuit boards. We eventually found that the power supply coupling capacitors were connected to the wrong nodes, resulting in a

configuration that was rapidly draining the logic battery. We re-soldered the capacitors into their correct positions and the problem was eliminated.

The output of the NAND gates used in the debounced switch was not consistent with the output of the pressure switches. To eliminate the discrepancy, we decided to bypass the debounced circuit, feeding the output of the switch directly into the MicaZ. The errors typically caused by a bounced switch are of no concern to us because of the sampling rate of the MicaZ is not fast enough to detect the switch bouncing. Even if the MicaZ saw bouncing on the output of the switch, it would eventually detect the steady state of the switch output and the control algorithm will very quickly react by entering the correct state. In retrospect, the debounced switch was probably a complication to the electronics system that should not have been included to begin with.

Our communications system was fully functional with the exception of some lost packets. The robots communicated with each other effectively, and all of the correct information was displayed on our graphical user interface. Due to the nature of our testing room and the MicaZs from other groups, we detected some wireless interference. The most common symptom was that one robot would respond to the start command before the other one would. The loss of packets also happened a few times while the robots were pushing the box. In these cases, the box was pushed incorrectly and was directed towards the side of the lane. To fix this problem, the base station MicaZ could be placed closer to the robots, or we could try to use a different channel.

Although we could only test our control algorithm for a few weeks, the final version proved to be quite efficient. Our preliminary version of the control algorithm featured exclusive use of the pushbutton switches on the robots. This algorithm was very effective, and occasionally the robots were able to push the box all the way down the lane by using only the feedback from the pressure switches. The feedback from the switches allowed each robot to continuously orient itself such that both pressure switches were pressed. This effectively keeps the robots moving in a straight line. In addition, the robots were programmed to change speed with respect to each other when certain states were encountered. The robots were able to occasionally straighten a crooked box by using this speed control determined by pressure switch data. However, a crooked box did not always activate the switches in the way expected, so sometimes these corrective states

were not encountered. We noticed that the robots would occasionally not correct for a crooked box or even orient themselves slightly off and push the box in a straight line directly out of the lane. We decided that we needed to incorporate our line sensors to avoid leaving the lane and to give us a more robust control algorithm.

We tested the robots by allowing them to push the box towards a line and determining how long they would have to turn in order to get the box back into the lane. We then implemented code to accomplish this when a line was detected. We had the robot that detected the line make a wide turn away from the line, and we had the robot on the other side slow down, making a shallow turn. We continued this operation for predetermined numbers of timer ticks in order to achieve the correct turn. We made sure that the robots were able to turn enough to get back into the lane, but not so much that they would go directly to the other side of the lane. This code has been tested and works well in conjunction with our pressure switch algorithm. The robots will push the box relatively straight using only pressure switch feedback, and if they detect a line, they will turn away from it and will continue pushing the box straight. We tested this extensively and had the robots pushing the box down the lane with consistent success. When it came to the demonstration, however, we had several malfunctioning line sensors. However, we were still able to demonstrate the functionality of our design because one line sensor was still functioning properly. In conclusion, the algorithm we have developed works very well, though our hardware limitations caused wavering reliability in the system's overall performance.

We intended on having optical encoder feedback, but we had problems interpreting the data from the encoders. They were working well physically when we tested them, but working with the data in the code was cumbersome. Correct feedback data would have allowed us to turn the robots down the lane more accurately, though in retrospect they were not necessary for accomplishing the task at hand.

10. Conclusions

Our robot features a strong chassis, well designed circuitry, an efficient printed circuit board, a reliable communication system, and an effective control algorithm. Our team

worked well together. The team worked hard, and we put in extra time and effort to complete our tasks to get the final project functioning.

The project has been a significant learning opportunity. Because we decided to build our robot from scratch, a couple of us had to learn how to use the tools in the machine shop. We also had to apply several concepts learned in previous classes for designing our electronics. None of us had ever used wireless communications or the nesC language, so we had to learn this language to make our robots communicate with each other in addition to understanding wireless and serial communications. We also had to learn how to create a graphical user interface using Matlab, and operation of drafting software to develop final mechanical schematics of the robot. Most importantly, we all had to learn how to work together effectively to complete our task.

If we were to repeat the project, we would do several things differently. First of all, we would try to develop more software code earlier in the semester. At the beginning of the project, we devoted most of our manpower to electronics and chassis design. This was good, but it limited our programming ability, as the nesC programming language is difficult to pick up quickly. In retrospect, sacrificing some time in chassis development may have been worth it to have an extra person on code development. It was a trade-off that we had to decide on at the beginning of the project. In the end, our decision was probably a good one, as we had several issues with the hardware and electronics that took teamwork and collaboration to diagnose, modify, and test.

We think that we could have accomplished the task better if we had made the electronics simpler. Our system as a whole was designed to be simple. However, it could have been simplified by removing circuits such as voltage dividers and debounced switches. Our sensors worked great in the testing stages, but had issues once they were integrated onto the robot. By simplifying the design, we could have simplified the troubleshooting stages and could have had a more reliable system. In addition, we should have focused more on power consumption and limiting unnecessary current flow, as this seemed to drain our batteries. It probably would have been wise to consider more efficient voltage regulators such as switching regulators, and different battery systems as well.

The biggest thing we learned about was component integration. It was common to have all of our systems working individually, but when we put them together they did not. Next time, we would try to integrate our systems together as soon as possible to avoid issues that prevent algorithm development and testing from taking place.

There are a number of things that we would do the same if we were to do it over again. Our control algorithm and communications protocol was designed very well. We coded a little at a time and continually tested what we had. Therefore, it was always easy to get the code to compile and function. Making small changes to the code led to faster and better results. We also think that our chassis design was very good. It was strong and reliable, and it was heavy enough to give the wheels adequate traction. Our robot is also very agile and can turn sharply if needed. We also had electronics that were simple to work with, even though we had problems with them. Our issues would have been much more complicated if we had designed more complicated sensor systems, which we avoided by designing them in an entirely digital format that made working with them quite easy.

This project served as a very good insight to coordinated behavior. Our robots were able to accomplish a task that a single robot would have a very hard time accomplishing by itself. This project was a good study of an applied multiple robot system and all of its necessary individual components.

11. Budget

Part:	Quantity	Price of each	Total
Batteries			
Rechargeable Batteries 4 9v Plus Charger	1	$22.89	$22.89
Duracell Procell 9v Batteries	8	$2.00	$16.00
Duracell Procell 9v Batteries (Norton)	8	$1.86	$14.88
Duracell Procell AA Batteries	4	$2.00	$8.00
Chassis			
Scrap Metal	1	$2.00	$2.00
Battery Holders	6	$0.35	$2.10
Circuitry			
Pin Headers	2	$1.00	$2.00
Male to Female Wiring Kit	1	$3.00	$3.00
Female to Female Wiring Kit	1	$2.00	$2.00
Female Pin Crimp Connectors	96	$0.15	$14.40
Female Pin Crimp Connectors (Norton)	39	$0.05	$1.95
Pin Housing (9 Hole)	4	$1.00	$4.00
Pin Housing (4 Hole)	4	$0.75	$3.00
Pin Housing (6 Hole)	2	$0.75	$1.50
Crimp Spades	13	$0.12	$1.56
5v Regulators	5	$0.50	$2.50
NAND Gate	1	$0.50	$0.50
XOR	1	$0.50	$0.50
Schmitt Trigger	3	$0.50	$1.50
14 Pin Sockets	9	$0.50	$4.50
1N4001 Diode	8	$0.10	$0.80
Sensors			
Optical Encoder Wheels	8	$0.95	$7.60
Encoder Sensor Assembly	4	$5.00	$20.00
Sensor Reflector Object OPB704	5	$3.48	$17.40
Rocker Snap Switches	4	$1.00	$4.00
Prototyping Board	2	$1.00	$2.00
		Total:	$160.58

12. Power Budget

3V MicaZ power

MicaZ: 0.036W

Encoder Wheel Sensors: 0.0295W each x2 = 0.0590W

Total: 0.0950W

Batteries: 2 AA @ 650mAH each → in series, so 650mAH:

20.5 hours of operation.

5V Logic Power Supply

H-Bridge Logic: 0.25W

Line Sensors: 0.1672W each x2 = 0.3344W

Total: 0.5844W

Battery: 1 conventional 9V, @600mAH:

5.1 hours of operation.

5V Motor Supply

Motors: 250mA average: 1.25W

Total: 1.25W

Batteries: 2 9V @ 250mAH each → equivalent to 1 9V @ 500mAH:

2 hours of operation.

References

[1] M.J. Mataric, M. Nilsson, and K.T. Simsarin. (1995, August). Cooperative Multi-Robot Box Pushing. *IEEE/RSJ International Conference on Intelligent Robots and Systems* [Online]. Volume 3, pp. 3556-3561. Available: http://coblitz.codeen.org:3125/citeseer.ist.psu.edu/cache/papers/cs/216/ftp:zSzzSzftp.usc.eduzSzpubzSznn_roboticszSzpaperszSzautonomous.robotszSz95zSziros95.pdf/mataric95cooperative.pdf

[2] P. Aparicio, and P. Lima. (1999, September). Autonomous Distributed Control of a Population of Cooperative Robots. Proceedings of SIRS99 [Online]. Available: http://welcome.isr.ist.utl.pt/img/pdfs/656_iclab99-5.pdf

[3] M. Koes, I. Nourbakhsh, and K. Sycara. (2005, July). Heterogeneous Multi-Robot Coordination with Spatial and Temporal Constraints. Proceedings of the Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovating Applications of Artificial Intelligence Conference [Online]. pp. 1292-1297. Available: http://www.cs.cmu.edu/~mberna/publications/koes-aaai05.pdf

[4] J. McLurkin. (1996) Using Cooperative Robots for Explosive Ordance Disposal. IEEE Explore [Electronic Journal]. Available: oscar.csail.mit.edu/~eugene/education/courses/6.836/eod-paper.pdf

[5] Y. Cao, A. S. Fukunaga, A. B. Kahng, and F. Meng, “Cooperative Mobile Robotics: Antecedents and Directions,” IEEE/TSJ International Conference on Intelligent Robots and Systems, Yokohama, Japan, 1995.

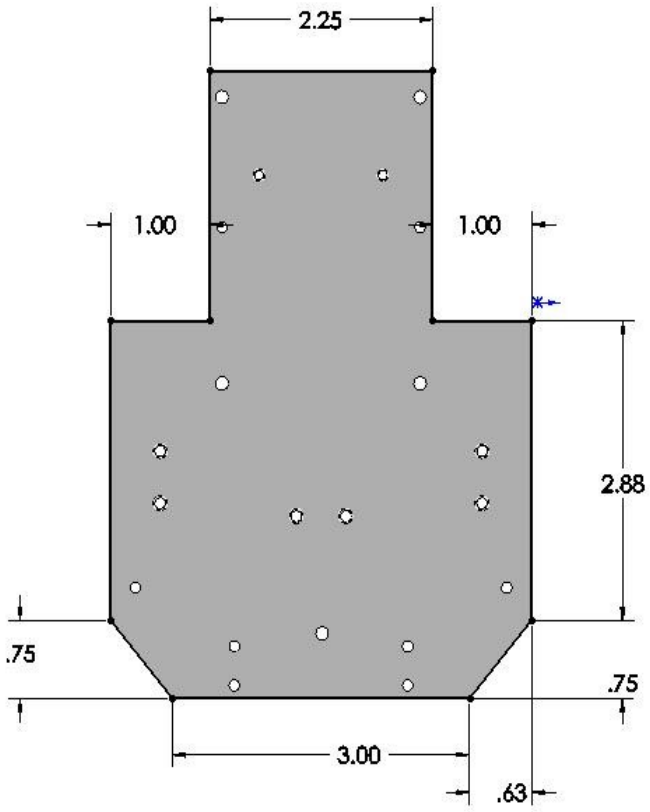
[6] C. L. Phillips and J. M. Parr, Signals, Systems, and Transforms, Prentice Hall, Upper Saddle River, New Jersey, 1995.

[7] V. J. Bucek, Control Systems: Continuous and Discrete, Prentice Hall, Englewood Cliffs, New Jersey, 1989.

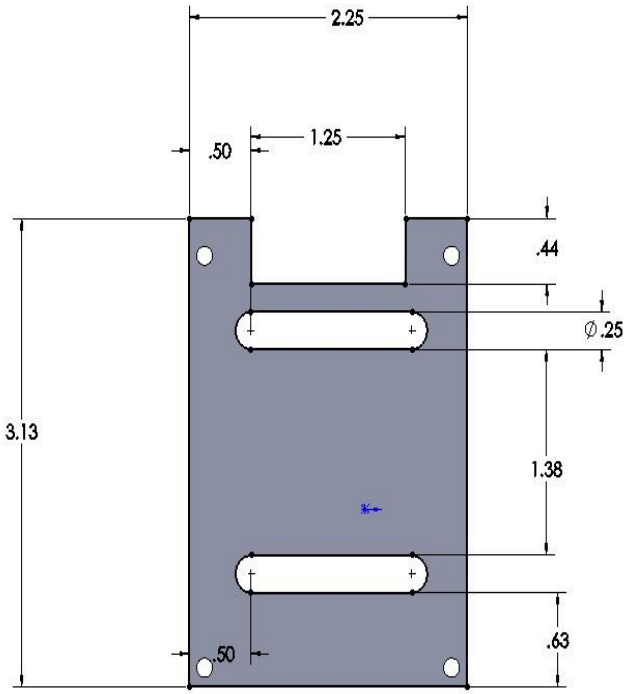
Appendix A – Chassis

Part	Manufacturer	Manufacturer Part Number
MicaZ	Crossbow	MOTE-KIT2400CB
Gearbox	LynxMotion	DMG-01
Wheels	LynxMotion	STS-01
H-bridge	LynxMotion	DHB-01

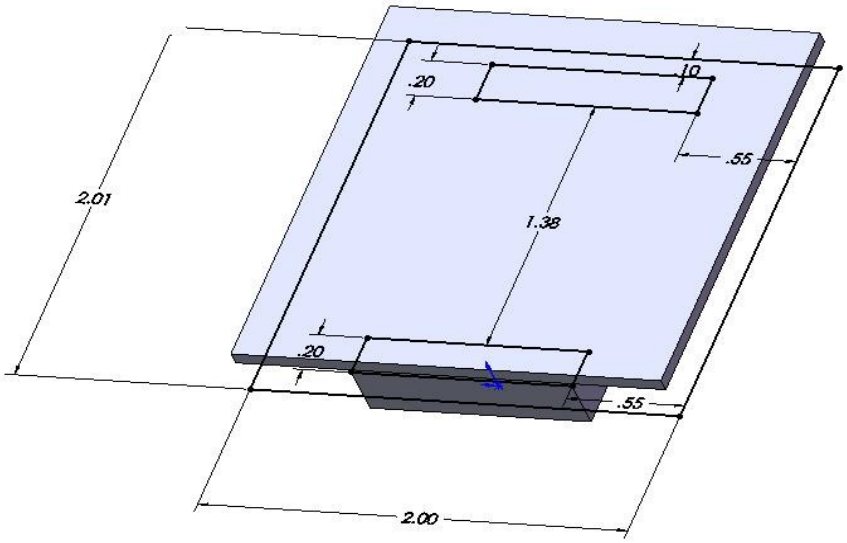
What follows are all of the parts we fabricated and/or modified.
All dimensions shown are in inches.



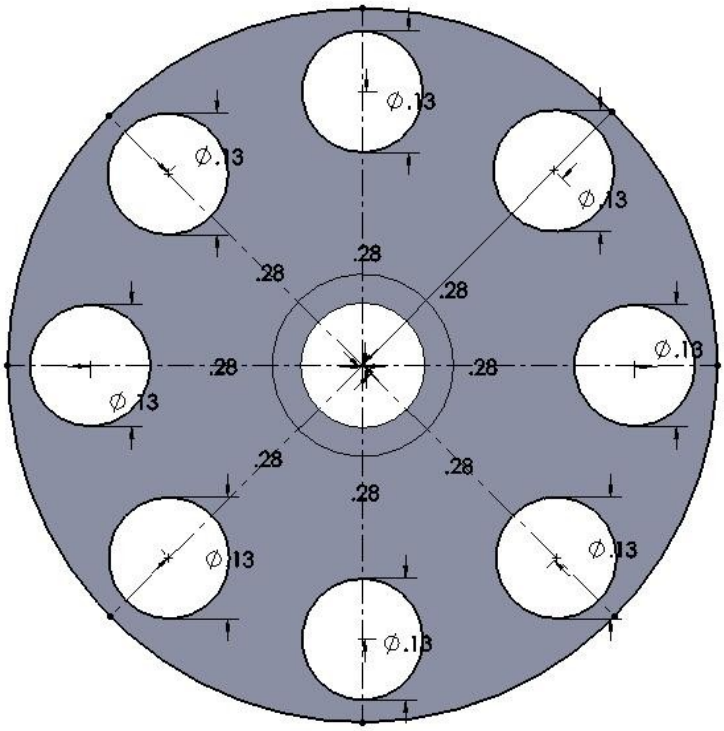
Chassis main plate.



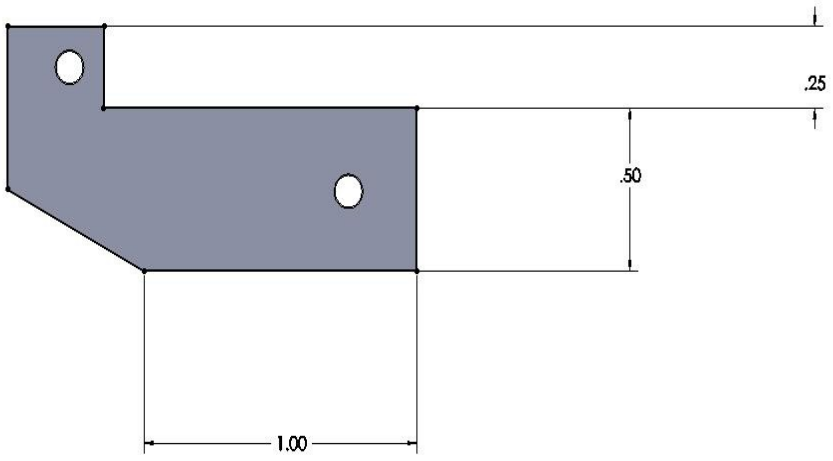
MicaZ mounting plate.



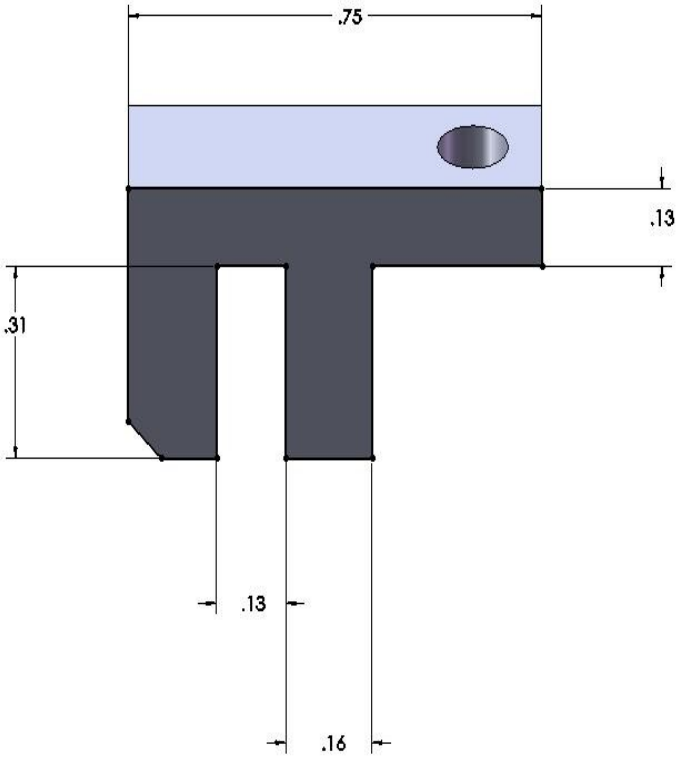
MicaZ socket.



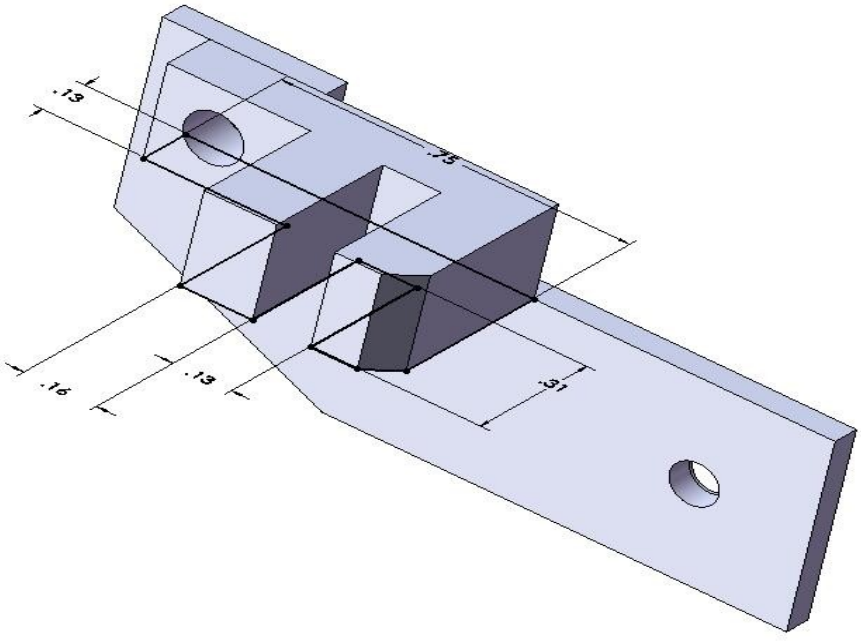
Encoder wheel side view, (Electronics Express 08DISC).



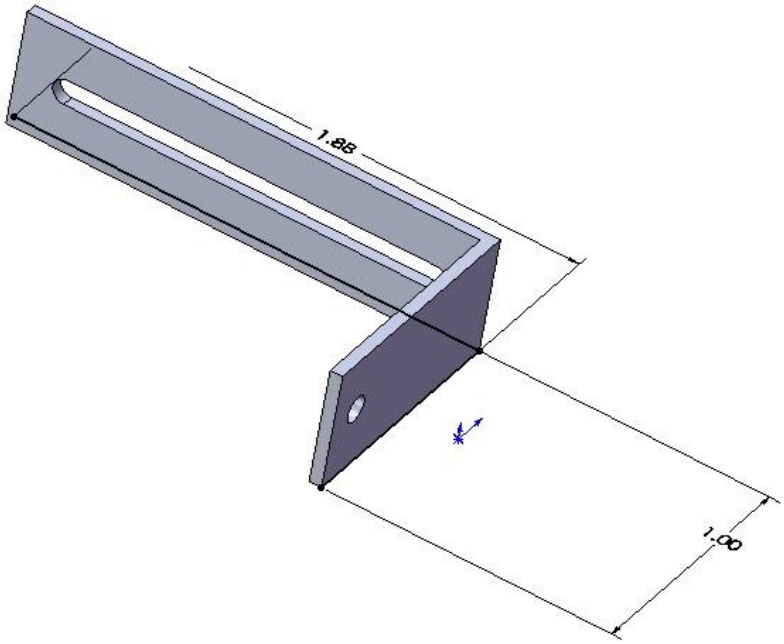
Encoder assembly mounting plate.



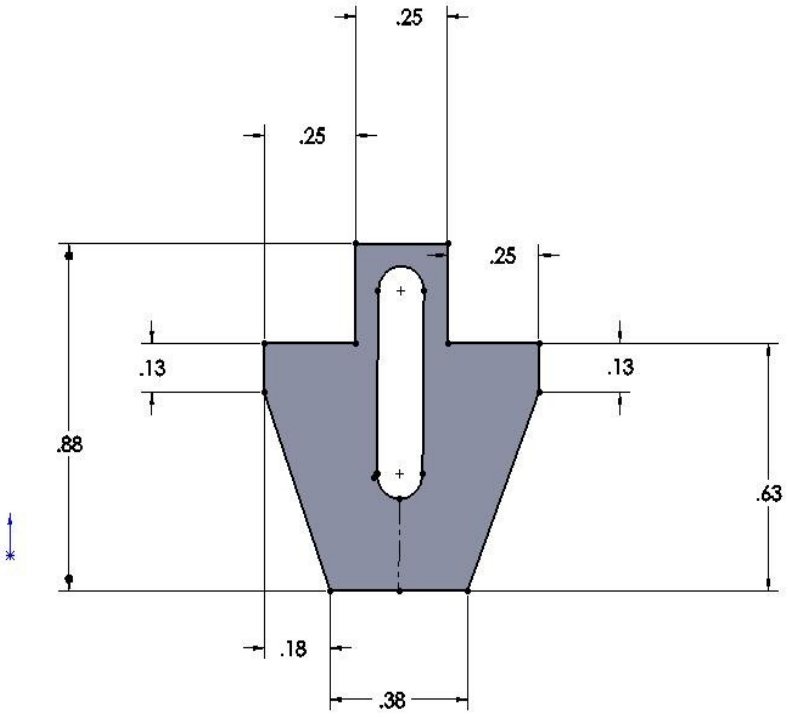
Modified H21LTB sensor.



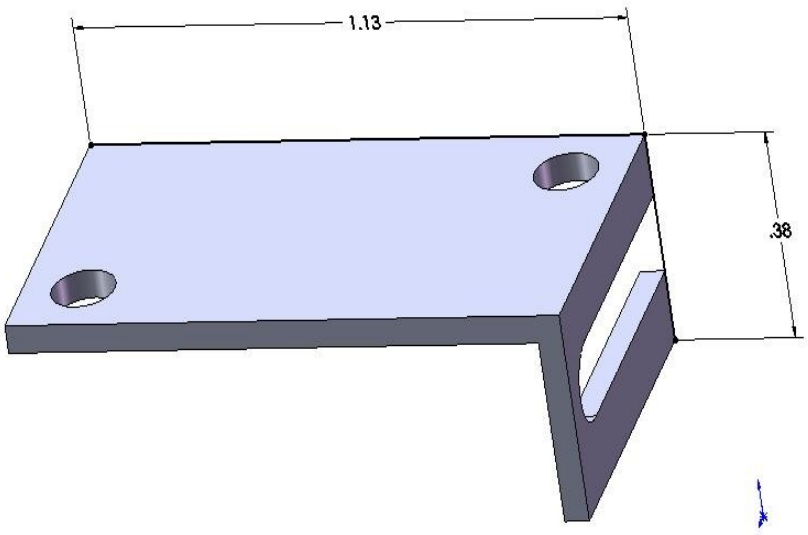
Encoder assembly.



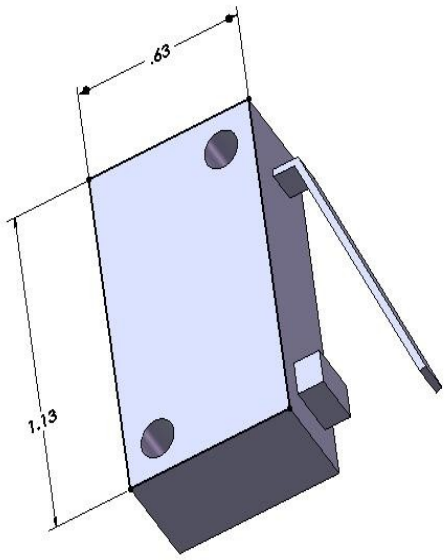
Line sensor bracket.



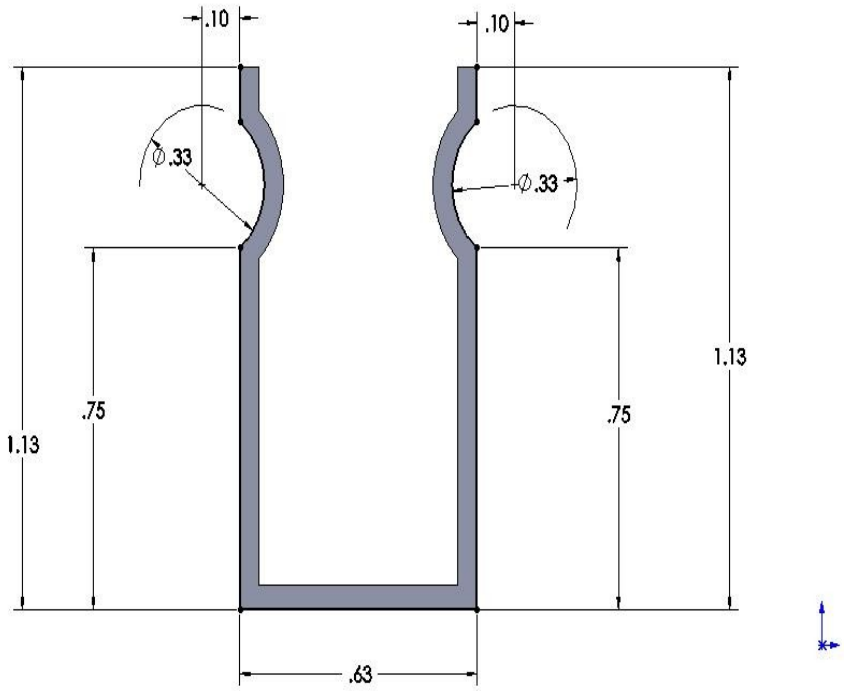
Line sensor, OPB704.



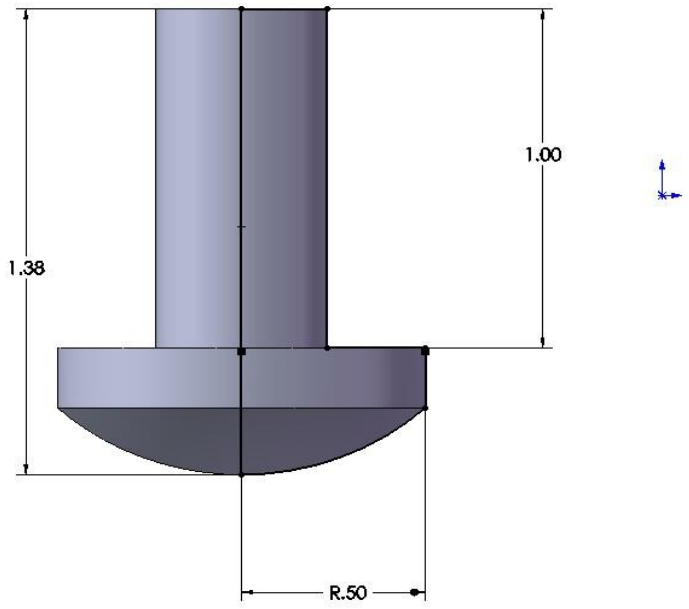
Push button bracket.



Push button sensor, V3L-1108-D8.



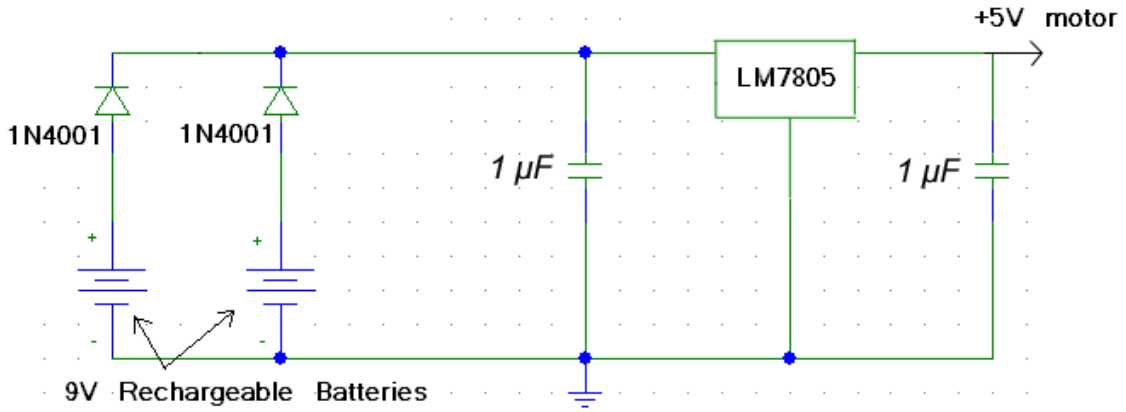
Battery clip.



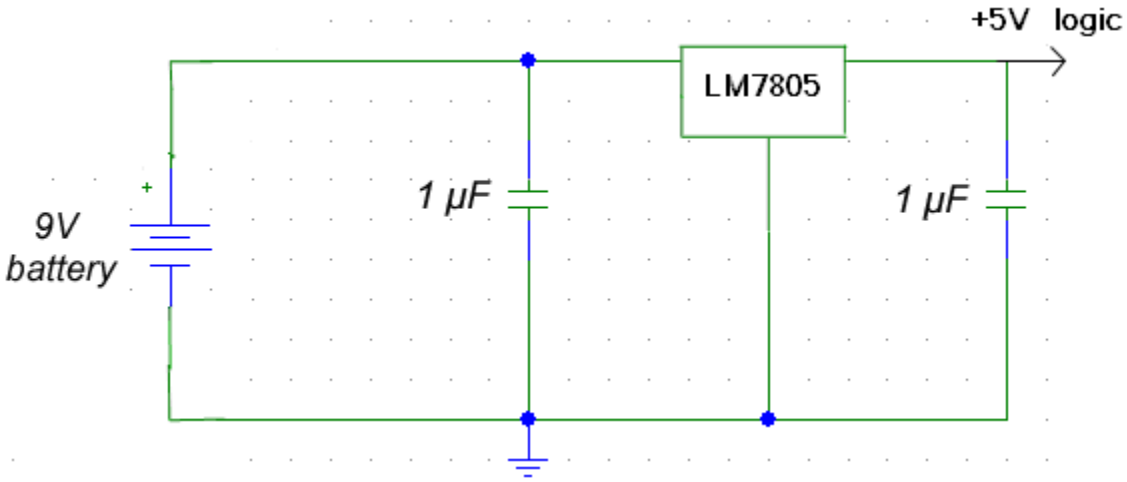
Front stub.

Appendix B - Electronics

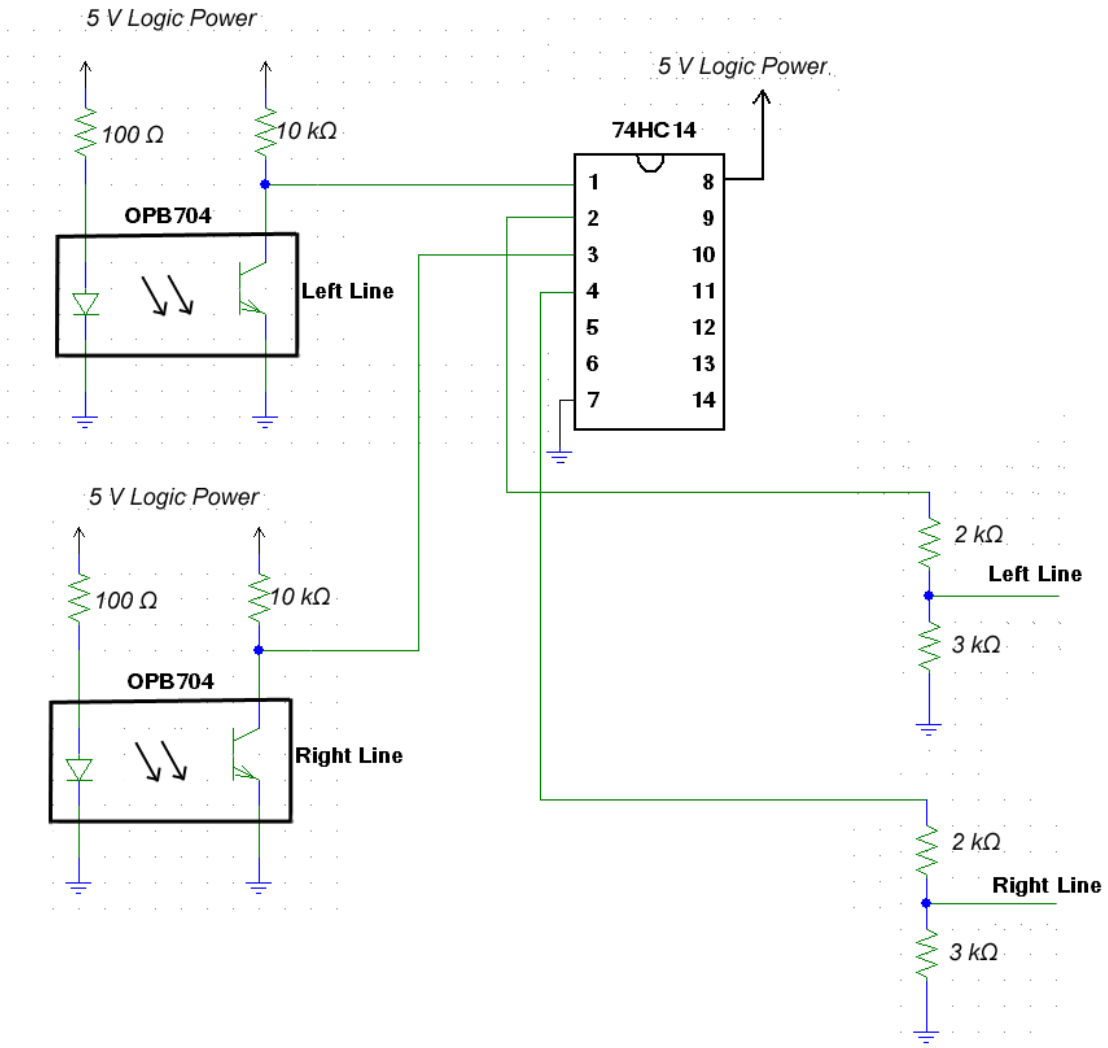
5V Motor Supply:



5V Logic Power Supply:

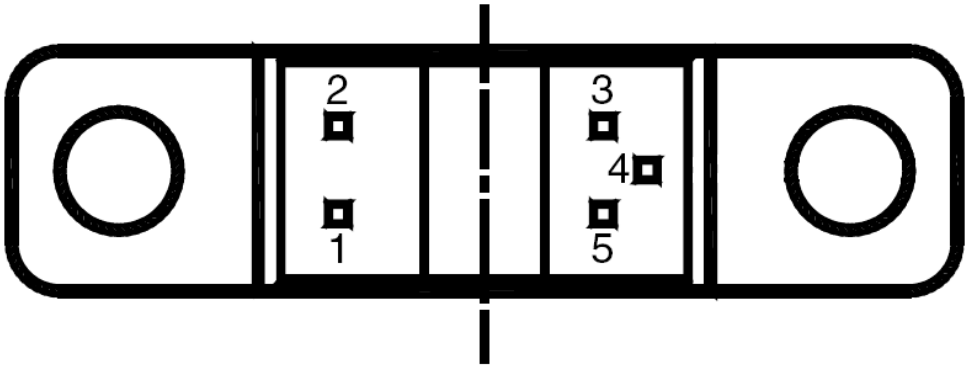


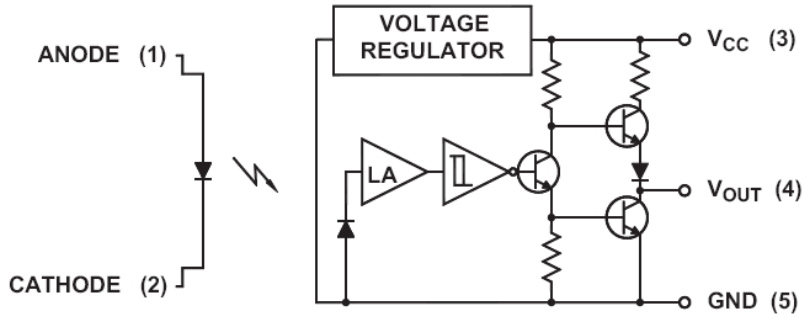
Line Sensor Circuitry:



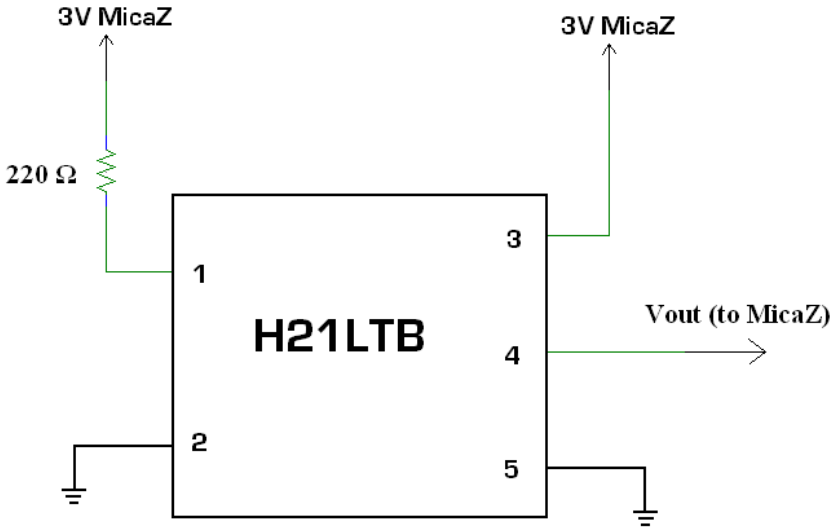
Encoder Wheel Sensors

This is the pinout of the H21LTB optical encoder sensor as well as the internals of the sensor. Diagrams are courtesy the Fairchild Semiconductor specifications sheet.





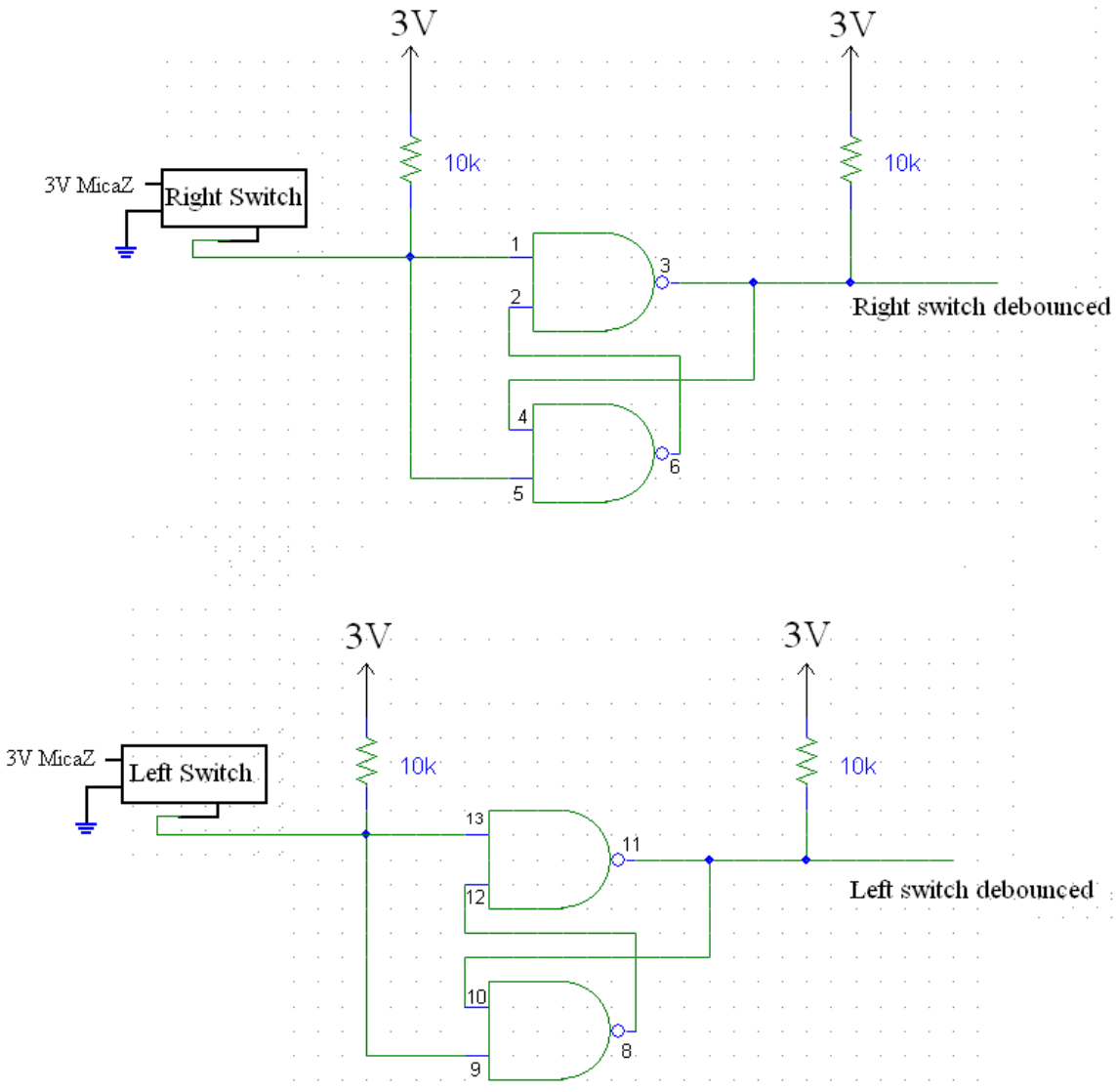
We used the following circuit to wiring up the sensors to the robot.



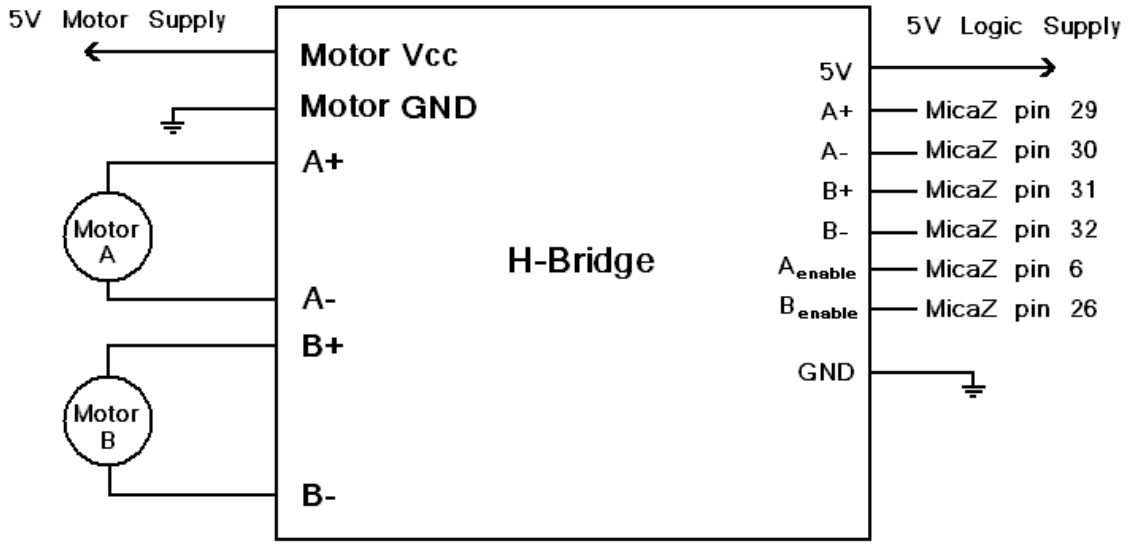
Pressure Sensors

The pressure sensors consisted of two pushbuttons, V3L-1108-D8.

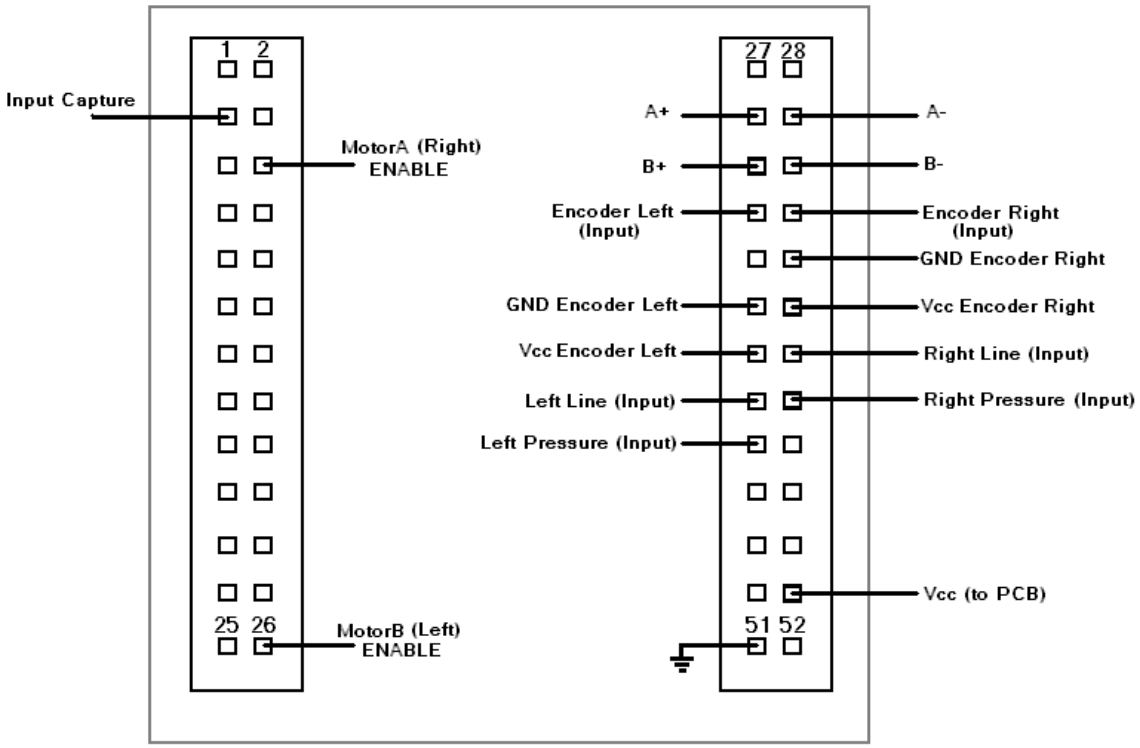
The logic chip used was a 74HC00 NAND chip, with pin 14 hooked to 3V and pin 7 hooked to ground. All power in this circuit was taken from the MicaZ 3V supply.



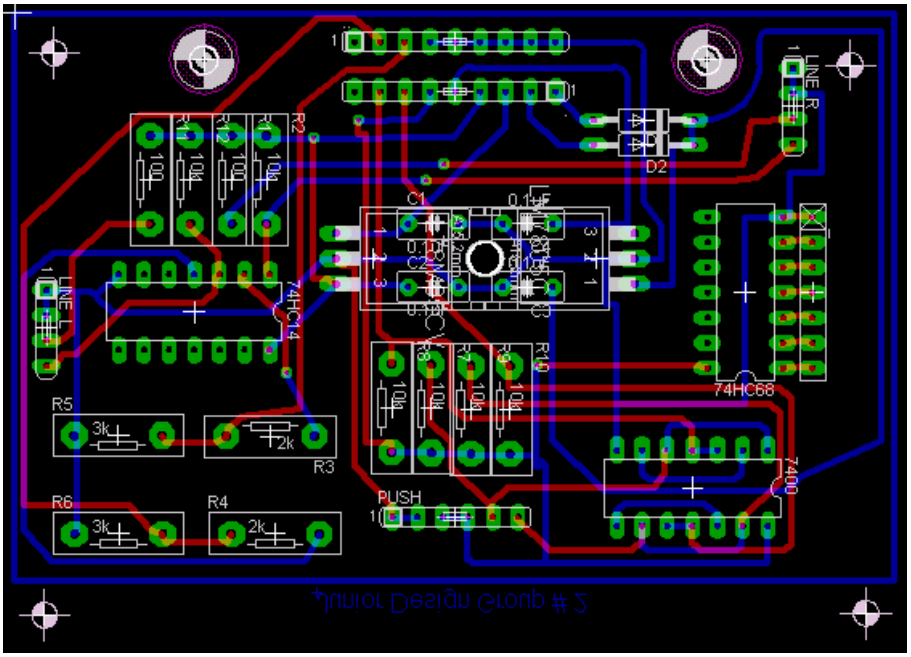
H-Bridge Wiring Schematic:



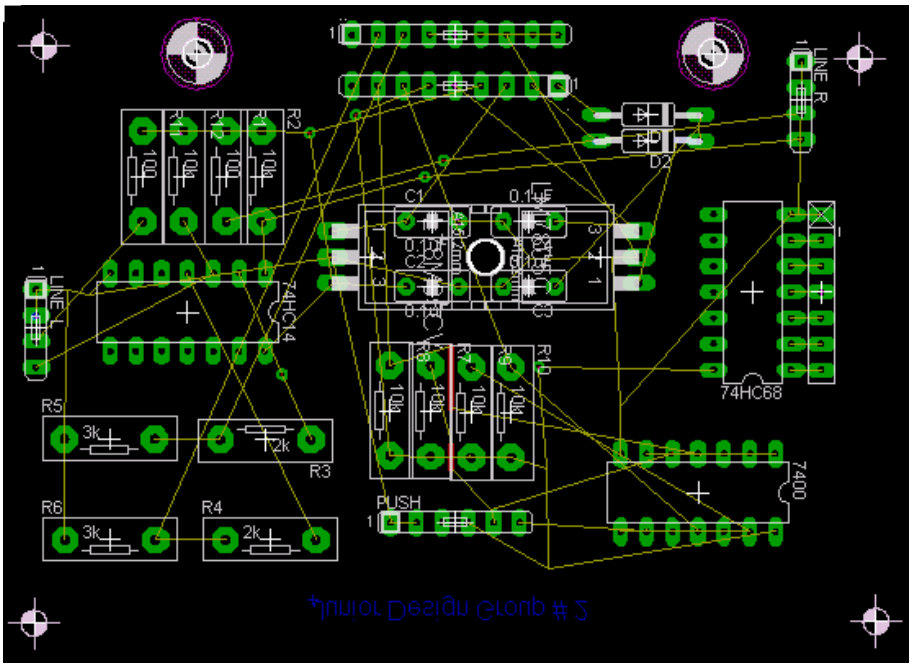
MicaZ pin out diagram. This picture represents physical pinout for MicaZ socket.



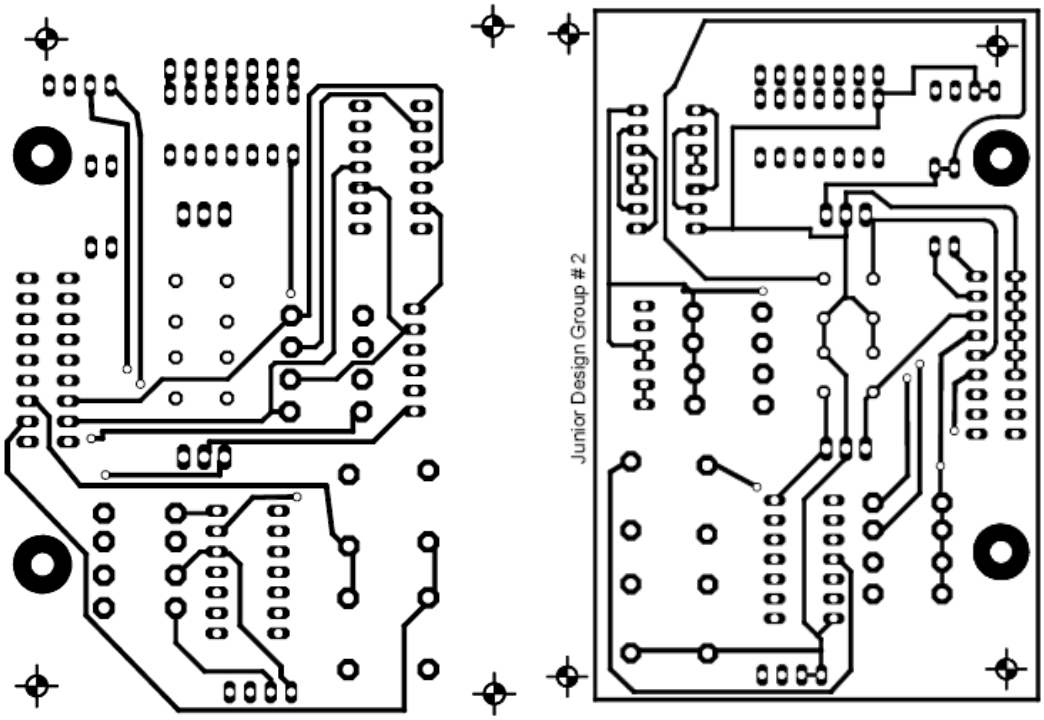
Appendix C - Printed Circuit Board



PCB completely routed showing footprints and top and bottom layers



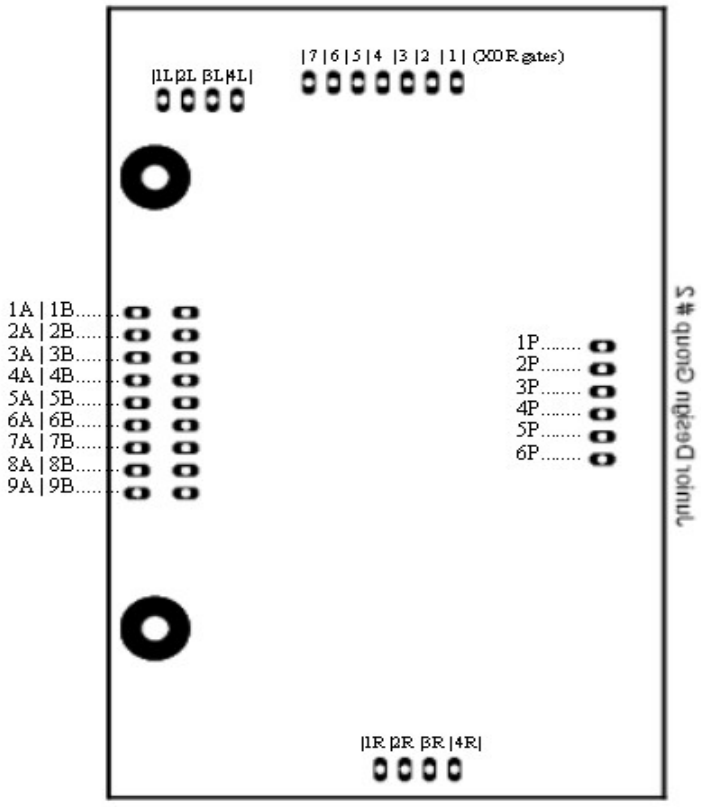
PCB with footprints and rats nest

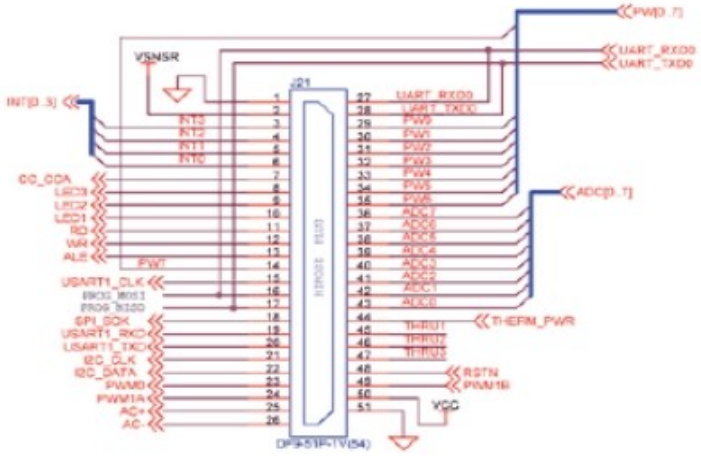


Printed top layer Printed bottom layer

Pin #	Description	Destination
1A	GND	Battery
2A	GND	Battery
3A	GND	Battery
4A	GND	H-Bridge Logic
5A	GND	H-Bridge Motor Supply
6A	GND	MicaZ pin # 51
7A	Line Sensor-Left	MicaZ pin # 41
8A	Line Sensor-Right	MIcaZ pin # 40
9A	N/A	
1B	9-volts in	Battery
2B	9-volts in	Battery
3B	9-volts in	Battery
4B	5-volts out	H-Bridge
5B	5-volts out	Motor Supply
6B	3-volts in	MicaZ pin # 50
7B	Push Button-Right	MIcaZ pin # 42
8B	Push Button-Left	MIcaZ pin # 43
9B	N/A	
1P	Left box sensor	push button switch
2P	Right box sensor	push button switch
3P	3-volts	push button switch
4P	3-volts	push button switch
5P	GND	push button switch
6P	GND	push button switch
1 (XOR)	Corresponds to the pin #s of the 74HC68 therefore pin # 7=GND	from Right encoder
2 (XOR)		from Left encoder
3 (XOR)		MicaZ input capture
4 (XOR)		
5 (XOR)		
6 (XOR)		
7 (XOR)		
MicaZ pin # 29	+/-(A/B) enable forward, reverse, or breaking functions of motors	A+ on H-Bridge
MicaZ pin # 30		A- on H-Bridge
MicaZ pin # 31		B+ on H-Bridge
MicaZ pin # 32		B- on H-Bridge
MicaZ pin # 33	Enables power to motor	Enable Left on H-Bridge
MicaZ pin # 34		Enable Right on H-Bridge

Overhead view of PCB





Appendix D – Matlab Code

Included on the attached CD is the MATLAB code used in our GUI. The CD contains all of the MATLAB files we used in the development of the GUI. Important files are detailed in the following table.

File Name	Description
micazwrite.m	The "Write" command will write binary numbers to the base station MicaZ to start the control algorithm.
start.m	The "Start" function writes the initial values to the base station MicaZ, and it continually reads from the serial port until the "stop" command is activated (stop button is pressed.)
read.m	The "Read" command will read incoming data from the base station computer. This data is mainly sensor data from the robots.
interpretdata.m	This file has the "interpret data" command that will display sensor data on the GUI.

Appendix E - File Descriptions for Base Station and Robot Programs

All the code can be found on the attached CD.

Base Station Program:

File Name	Description
Base.nc	Main interface interconnection definition file for the program.
DataToUART.nc	Interface interconnection definition file for the DataToUART Module.
DataToUARTM.nc	This Module contains basic code to send a packet over the serial line. Accessed via the MsgOutputUART interface. Data is sent in the form of the MyMsg5 structure.
MsgOutput.nc	Interface definition file for the MsgOutput interface which is no longer used.
MsgOutputR1.nc	Interface definition file for the MsgOutputR1 interface which brings Robot 1's data from the ReceiveData module to PWMCounter, the main module.
MsgOutputR1Out.nc	Interface definition file for the MsgOutputR1Out interface which brings data for Robot 1 from PWMCounter, the main module, to PWMIntToRfm, the send data module.
MsgOutputR2.nc	Interface definition file for the MsgOutputR2 interface which brings Robot 2's data from the ReceiveData module to PWMCounter, the main module.
MsgOutputR2Out.nc	Interface definition file for the MsgOutputR2Out interface which brings data for Robot 2 from PWMCounter, the main module, to PWMIntToRfm, the send data module.
MsgOutputUART.nc	Interface definition file for the MsgOutputUART interface which brings data for the GUI from PWMCounter, the main module, to DataToUART, the send data (over serial) module.
MsgOutputUARTInput.nc	Interface definition file for the MsgOutputUARTInput interface which brings data from the GUI through the ReceiveData module to PWMCounter, the main module.
MyMsg.h	This file contains code to define the MyMsg structure, which contains data that is transmitted from the Base Station to Robot 1 using a Type ID of 4.
MyMsg2.h	This file contains code to define the MyMsg2 structure, which contains data that is sent from the GUI to the Base Station using a Type ID of 3.
MyMsg3.h	This file contains code to define the MyMsg3 structure, which contains data that is sent from the Robots to the Base Station using Type IDs of 10 (Robot 1) and 11 (Robot 2).

MyMsg4.h	This file contains code to define the MyMsg4 structure, which contains data that is transmitted from the Base Station to Robot 2 using a Type ID of 12.
MyMsg5.h	This file contains code to define the MyMsg5 structure, which contains data that is transmitted from the Base Station to the GUI using a Type ID of 255.
PWMCounter.nc	This file contains the code for the main module, which integrates the timer and all of the interfaces with the control algorithm code. Data from the ReceiveData module is brought in and used in the Check() task to determine output data. Output data is then stuffed into the appropriate structure and sent over an interface to the appropriate Send module. See code comments in this file for details.
PWMIntToRfm.nc	Interface interconnection definition file for the PWMIntToRfm Module.
PWMIntToRfmM.nc	This Module contains basic code to send a packet over wireless. Accessed via both the MsgOutputR1Out and the MsgOutputR2Out interfaces. Data is sent in the form of the MyMsg (Robot 1) and MyMsg4 (Robot 2) structures. Data being sent is separated by Type ID so that individual packets can be sent specifically to each robot.
ReceiveData.nc	Interface interconnection definition file for the ReceiveData Module.
ReceiveDataM.nc	This Module contains basic code to receive data from any source (in this case serial or wireless). Data that is received is separated by Type ID so that the source of the data is known. Data is then channeled to the PWMCounter module using the MsgOutputUARTInput, MsgOutputR1, and MsgOutputR2 interfaces in the form of the MyMsg2 or MyMsg3 structures.

Robot Programs (Robot 1 and Robot 2 have identical code):

File Name	**Description**
MsgOutput.nc	Interface definition file for the MsgOutput interface which brings data in from the Base Station through the PWMRfmToInt module to RobotTimerMain, the main module.
MsgOutputOut.nc	Interface definition file for the MsgOutputOut interface which brings data for the Base Station from RobotTimerMain, the main module, to PWMIntToRfm, the send data module.
MyMsg.h	This file contains code to define the MyMsg structure, which contains data that is being received from the Base Station.
MyMsgOut.h	This file contains code to define the MyMsgOut structure,

	which contains data that is being sent to the Base Station.
PWMIntToRfm.nc	Interface interconnection definition file for the PWMIntToRfm Module.
PWMIntToRfmM.nc	This Module contains basic code to send a packet over wireless. Accessed via the MsgOutputOut interface. Data is sent in the form of the MyMsgOut structure.
PWMRfmToInt.nc	Interface interconnection definition file for the PWMRfmToInt Module.
PWMRfmToIntM.nc	This Module contains basic code to receive data from the wireless. Data is channeled to the RobotTimerMain module using the MsgOutput interface in the form of the MyMsg structure.
Robot*.nc (* indicates number for robot [1 or 2], but each file will be identical)	Main interface interconnection definition file for the program.
RobotTimerMain.nc	This file contains the code for the main module, which integrates the timer and all of the interfaces with the sensor data collection code. Data from the PWMRfmToInt module is brought in and used to apply new PWM values to the Robot. Sensor data is collected on the timer, is packed into the MyMsgOut structure, and then is sent to the PWMIntToRfm module via the MsgOutputOut interface. See code comments in this file for details.