4-18-2006

Autonomous Robotics at Trinity University (A.R.T.U.) IEEE Robotics Competition

Adam Crouch
Trinity University

Cary Wong
Trinity University

Brandi House
Trinity University

Michael Hohimer
Trinity University

1

Senior Design Final Report
Autonomous Robotics at Trinity University (A.R.T.U.)
IEEE Robotics Competition

ENGR 4381
April 18, 2006

Adam Crouch, Cary Wong,
Brandi House, and Michael Hohimer

Advisor: Michael Yockey

draft

Abstract

This senior design project sought to construct an autonomous robot such that it would be capable of competing in the 2006 IEEE Region 5 robotics competition. This required the robot to locate and distribute a red, blue, green, and yellow can to its appropriate position based on the color of the can, all within a 3 minute time limit. The purpose of this report is to describe the design goals and specifications, outline the procedures taken to design the hardware and software of the robot, and to relate all results and conclusions for the project. At the competition, the robot successfully delivered all four cans to the correct locations, avoided hitting any human workers, completed the course with a time of 2.08 minutes (125 seconds), and received the 4th position for the competition. Based on these results, the project was a success as it met all design criteria. For future improvements, the group recommends using DC motors instead of servos in the drive system to achieve faster speeds on the course, and an automated calibration procedure would also be very useful in future projects.

Executive Summary

The design goal for this senior design group was to construct an autonomous robot that would be capable of competing in the 2006 IEEE Region 5 Robotics Competition. According to the 2006 rules, the robot's main task is to find four cans in the 'incoming' rooms on the course and sort them to their appropriate 'outgoing' room on the other side of the course based on their red, blue, green, or yellow color. This task must be completed within a three minute time limit, and the fastest possible time is optimal. In the final round of the competition, 'human workers' in the form of Barbie dolls are added to the course wearing clothing that matches the color of the cans, and the robot is discouraged by time penalties from hitting these workers. More specific descriptions of the course layout and rules can be found in the Region 5 Conference Robotics Competition Rules [1].

To accomplish this task, the group first had to select the hardware components. Numerous options were evaluated before selecting each component. The Gumstix Robostix board was selected as the main processor of the robot that manages all other hardware. To locate and track the cans, the group selected the CMUcam2+, which is a camera that can also perform some basic image processing and send information to the main processor over a serial line. For the drive system, the group chose to use the combination of servo motors, digital encoders, and photoreflectors. The servos are modified to turn continuously, and they will be used to mobilize the robot. The encoders are used to determine distances and speeds, and the photoreflectors are used to follow the lines on the course and detect critical points such as intersections or black circles. Proximity sensors were selected as one form of Barbie detection, and these sensors allow the robot to sense when an object is within a certain distance from the can. Finally 9.6 V 1500 mA Ni-MH batteries are used to power the robot. Testing of each of these individual components defines their accuracy and abilities, and once integrated, final testing is performed on a mock up of the course.

The body and gripping mechanism designs are also major considerations for the success of the robot. The body shape is mostly round with extensions on the front that allow the can to fit between them. The gripper then fits into the left arm and it features a cam (a skinny solid cylinder) attached to a servo that rotates the cam on an off-centered axis. This gripper allows the can to be pressed tightly against the opposite arm and lifted slightly off the ground to avoid bumps in the course.

The most time-consuming aspect of this project was the software design, implementation, and testing. The group selected a bottom-up approach to the software design, meaning that drivers for each individual component are written first, then subprograms that use the drivers are written, and finally the main logic that combines all of these items to navigate the course. Provided example code with the processor allowed relatively quick development of the drivers, and testing was enhanced by the discovery of a serial debugger that can output readings to a terminal on a computer while the program is running. Testing on the mock-up course allows the team to optimize threshold values and routines such that the robot quickly and accurately runs the course.

The team competed with the robot on April 8th at the 2006 IEEE Region 5 Competition. The robot correctly sorted the cans in the initial runs of the competition, and it continued on to the final round with human workers on the course. The robot's final time was 125 seconds, and the team received the 4th position out of 35 national teams. This project is considered a success as the final robot met all design specifications. Further improvements to the design may include replacing the servos with DC motors for faster navigation, implementing an automated calibration procedure, and improved heat management for the system.

Table of Contents

1. Introduction

The objective of the senior design project is to construct a robot that is able to compete in the 2006 IEEE Region V Student Robotics Competition. The design group will consider the project a success if, by the end of the academic year, the group has constructed a robot capable of completing the course specified in the 2006 IEEE Region V Student Robotics Competition Rules and Course Description document as seen in Appendix A. Furthermore, the group would find it exemplary if the robot completed the second round of the competition, which introduces obstructions.

In this year's competition, competitors must construct an autonomous robot capable of recognizing and sorting different colored containers on a course, which simulates an automated warehouse. Four different colored soda cans, each located in one of four "incoming" rooms on the course, are to be identified and transported to four "outgoing" rooms. Each colored soda can must be transported to a different outgoing room based on the can's color. Black lines on the floor of the course may be used to guide the robots from room to room. Each team's robot will have two runs on the course, and the team will be awarded points based on how fast and how accurately the robot accomplishes its tasks. During each run, a time limit of three minutes is imposed.

The highest scoring teams in the first two rounds compete in a final round containing a new challenge. Barbie dolls dressed in variously colored coveralls positioned on the track simulate human workers in the automated warehouse. The robots ought to be able to distinguish between these dolls and the soda cans. Penalties are assessed for displaced or toppled dolls. The Rules and Course Description for the Robotics Competition states that each robot must have the following components:

-Drive system for motion
-Navigation sensor array to move from one room to another
-Sensing element to differentiate between different colored containers
-Manipulator to handle the containers
-Processor/software to coordinate all of the above components

Over the course of two semesters, the design group will communicate its progress by means of four formal presentations and five reports. The culmination of the group's effort will be displayed at the IEEE Region V Robotics Competition, which will be held in San Antonio, Texas on April 8, 2006.

2. Component Selection, Driver Development, and Testing

2.1 Body Design

The first thing to consider when starting the project is the body design. The group must think of a clever design that will accommodate all of its many components and allow the robot to flawlessly perform its task of transporting cans. Size constraints must also be taken into account while brainstorming design ideas. The body of the design is the platform on which every other component must be housed. It must accommodate a camera, line sensing photo reflectors, proximity sensors, a can manipulating device, a drive system, a main processor, and a power supply. Additionally, the ideal robot ought to be small, fast, and strong. The robot must be maneuverable and able to pick up soda cans. In certain circumstances, there will be obstacles on the course in the form of Barbie dolls, so the robot must not be bulky or unwieldy. Figures 1, 2, and 3 demonstrate a few body designs the group has considered.

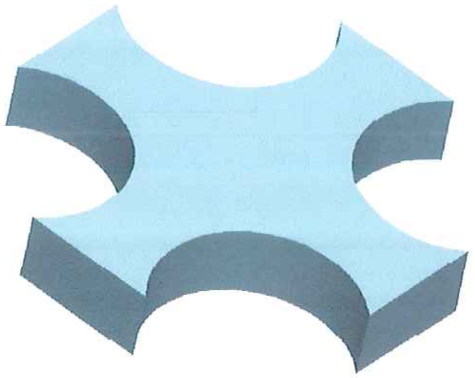


Figure 1: The Body of a 4-Can Carrying Robot

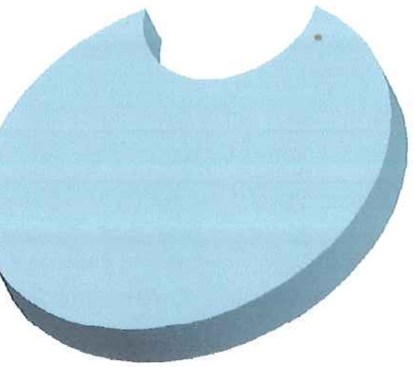


Figure 2: The Body of a Simple, Circular Design

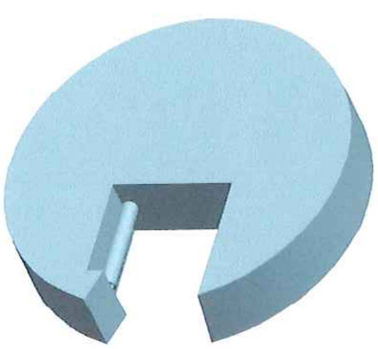


Figure 3: Single-Can Lifting Robot

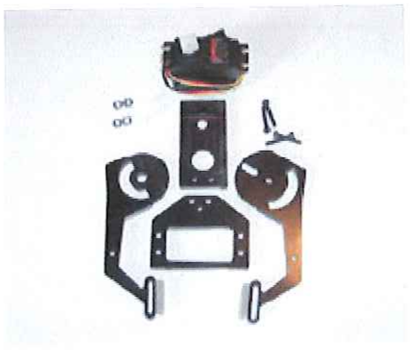


Figure 4: Can Manipulator Kit [1]

There are two basic ways the robot can move cans around the course. One design is to have the robot carry all four cans at once, limiting the distance the robot would have to travel in its quest to take the different colored cans to their designated rooms. The downfall of this design is that the robot would have to carry a lot of weight around and would have to be designed to hold four cans at once. A second design consists of a single can carrying robot which would allow the robot to move quicker but have to take many trips to move all four cans. Some preliminary state analysis proved to the group that the robot would not have to make a significant number of extra trips if it had to carry one can at a time. This analysis convinced the group to choose a single can carrying design.

Figure 1 shows a design in which the body rotates and is able to grab and drag four cans at once. This design would be useful because the robot could make the least number of trips between rooms. The downfall of this design is its mechanical complexity. The can holding compartment would have to rotate, making it very complicated mechanically as well as hard to program. Also, the robot would have to hold and drag four cans around the course, necessitating a much stronger robot than other designs would require. This is also a very expensive design, requiring stronger motors and many servos. Finally, as previously stated, a preliminary analysis of the vehicle's path along the course shows that a design such as this would not save much time over a design that loads and delivers one can at a time.

A design with a simple single-can manipulator can be seen in Figure 2. This design would be easy to implement and relatively inexpensive. One servo would be used to rotate a gripping mechanism to hold the can in its assigned place. The tradeoff to this design is that it could only carry one can at a time and would have to make more trips between rooms. Also, if not designed properly, the robot would be forced to drag the can, which would impede its motion.

The design in Figure 3 is a modification to the design seen in Figure 2, yet this design allows the robot to lift the can slightly and carry it from room to room instead of dragging it. This design makes use of a single servo with an off-centered cylinder attached to it (a cam). When the servo rotates, the cam would not only squeeze the can and grip it, but it would actually lift it off the ground. This design eliminates the resistance to motion due to the friction of a dragging can.

One final design considered involves building a robot equipped with a purchased can manipulating device, which can be seen in Figure 4. One of these kits can be obtained for $25 on the Internet [1]. This design would be more expensive and would not allow the cans to be lifted off the ground. However, it would provide a simple alternative to creating a custom can manipulator.

While each of the body design approaches is feasible, the third appears to be the most efficient design. This design not only allows for the smooth transportation of the four cans but leaves a lot of space for the components of the robot to be placed. Table 1 summarizes the advantages and disadvantages of each of these designs.

Table 1: Comparison of Body Designs

Design	**Advantages**	**Disadvantages**
4-Can Carrying Robot	-shorter travel path	-complex -hard to program -expensive -must drag the cans
Single Can Dragging Robot	-simple	-must drag the cans -must drag one can at a time
Single Can Carrying Robot	-simple -does not drag the can -inexpensive	-can only carry one can at a time
Can Manipulator Kit	-simple -manipulator built to carry cans	-expensive -must drag one can at a time

Although the body design does not itself have mechanical or electrical components to be tested, it still must meet design specifications and be optimal for transportation of all hardware and can manipulation. In the second part of the competition, the robot must be able to complete the course without disturbing obstructions in the form of Barbie® dolls. For this reason, the group selected a relatively circular design. A circular design without protruding corners enables the robot to more easily enter a room, pick up or drop off a can, turn around, and exit a room without disrupting obstacles. The can manipulator will extrude slightly from the front so that the area of the robot entering the rooms is minimized, decreasing the possibility that the robot will disrupt a 'human worker'. This also allows the line sensing photo reflectors to be placed further forward on the robot, causing the line-following system to be more accurate and have more immediate reaction times. A CAD drawing of the base for this body design can be seen in Figure 5 below. The group acknowledges a slight resemblance of their *Falcon's* body to that of the *Millennium Falcon* seen in Figure 6, so some credit is due to George Lucas for creative inspiration.

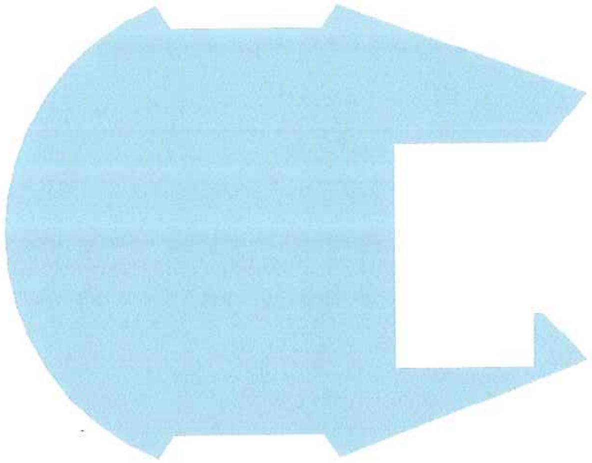


Figure 5: Pro-E drawing of *Falcon* body design

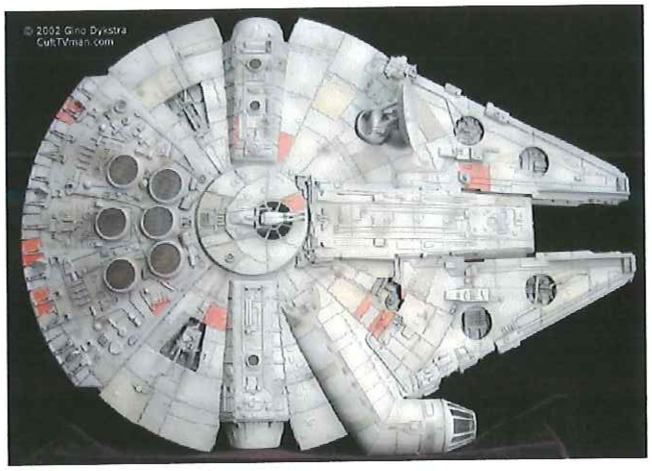


Figure 6: *Falcon* not to be confused with the *Millennium Falcon*

The wheels are positioned on the center axis of the circular design so that the Falcon can rotate any angle without subsequent translation of the body. The width of the Falcon will be less than ten inches, as the competition rules state that obstructions will be no closer than 5 inches from the black line [2]. This ensures that the Falcon will not hit any Barbie® dolls while it is following the lines on the course. The battery will be placed on the rear of the Falcon to account for counterbalancing needs when the Falcon is carrying a can. A second level may be constructed to create room for all necessary components and to provide height for the camera system without increasing the diameter.

The final design of the Falcon will be made of a lightweight metal such as aluminum. Before the final body is made, the group will first draw out the design on cardboard and model it in Pro-Engineer. Once all the dimensions are selected, a prototype will be made with plywood. This prototype allows the group to determine proper placement of all the components holes for mounting each piece. This also allows the group to optimize the design before fabrication of the final body.

2.2 Can Manipulator

One servo will be used in the design for the can manipulator. The group decided that the most effective transportation of a soda can would be to lift it off the ground instead of allowing it to drag. A unique design was developed in which a high torque servo will rotate a cylindrical gripping pad mounted on an offset axis. The gripping pad will squeeze and lift the can, pressing it against the opposite wall, which will be slanted to accommodate the lifting motion of the gripping pad. If necessary, a plate can be added to the slanted wall to provide more support. A diagram of this can manipulator design can be seen below in Figure 7.

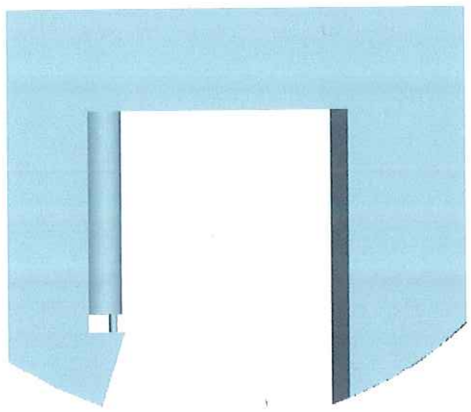


Figure 7: Drawing of Can Manipulator

This design allows enough space for the robot to position the can inside the manipulator while accounting for alignment errors. The tips of the manipulator will be funneled inward in order to better accommodate alignment errors. The design will also be dimensioned such that the cam will squeeze and lift the can up against the opposing wall. If the axis of the cam is not offset enough, it will not provide enough force to lift the can. Specifically, the servo used must provide enough torque to the cam to lift a 2.5 ounce soda can with a diameter of 2.5 inches and a height of 4.8 inches [2]. The design will also place the can in a standing position when released from the mechanism, as required by the competition rules.

Once the dimensions are fully specified, a code module will be written to activate and deactivate the manipulator at appropriate locations in the course. The servo ~~will~~ rotate the proper direction and magnitude in order to securely grasp the can and hold it while the Falcon moves, and to accomplish this task, a correlation between the pulse width and the servo rotation ~~will~~ be established.

2.3 Processor

A processor is needed to coordinate all the elements of the Falcon. This processor needs to monitor the robot's sensors and command the drive system and can manipulator to perform specific actions. There are many different processor options for our application. Some available processors are built specifically for robotics applications. These processors have much of the needed external components—such as I/O pins, user input and feedback, and voltage regulation—packaged on a single board. Examples of these types of processors are the OOPic [3] and the Ridgesoft IntelliBrain [4]. Alternately, a programmable logic controller (PLC) is capable of controlling the robot. A PLC is a self-contained system that contains a CPU, power supply, and I/O pins. The outputs of this device can be set to deliver enough power to run motors; and no external board is needed to convert logic level voltages. Finally, a general-purpose microcomputer can be used to control the robot. The microcomputer currently under consideration is the Gumstix Waysmall [5]. A Waysmall computer is a tiny yet complete Linux system. This computer would need to be connected through parallel communications to an external I/O board that can provide digital and A/D input header pins and powered output header pins. Table 2 is a feature comparison for the control systems mentioned above.

Table 2: Comparison of Processors

System	PLC	OOPic	RidgeSoft IntelliBrain	Gumstix Waysmall and Robostix
Model #	D0-06DDx-D	OOPic-R	--	200ax
Price	$199	$89	$199	$178
Inputs	20 DC	22 digital I/O, 4 of which can be A/D	7 A/D, 10 digital (expandable)	25 gen. purpose digital I/O pins, 8 A/D
Outputs	16 DC	16 servo outputs, 2 PWM outputs	2 servo ports	6 PWM servo outputs
Serial I/O	yes	no, primary serial port is dedicated to Serial Control Protocol	yes, up to 115.2K	yes, 2 serial ports (1 FF-UART, 1 HW-UART RS-232)
Power Supply	12-28 VDC, 150mA max	9 V, cell batteries; 3 Voltage regulators (separate logic and I/O power supplies)	4.5V to 9V (4 AA batteries), support for AC adaptor	Waysmall: 3.4 - 5.2 V (Li-Ion, Li-Polymer, 3-NiMH, standard 4.5 or 5.0V inputs); Robostix: 6-12 V
Memory	14.8K (words)	512 Bytes of RAM; 256 Bytes of EEPROM	RAM: 132K, Flash: 128K, EEPROM: 4Kbytes (expandable to 68Kbytes)	64MB SDRAM, 4 MB Strataflash, expandable
Speed	Contact execution: 0.6s; Typical scan: 1-2ms	2 kHz	14.7 MHz	200 MHz Intel Xscale PXA255
Programming Language	Ladder-logic	C with downloadable objects	Java ("rich RoboJDE robotics class library and easy to use user interface")	C, Java, Python
Programming Environment	Windows based ladder logic program	Windows based OOPic programming software	Windows based RoboJDE	Linux kernel 2.6; userspace includes dropbear, boa and more
Onboard user feedback	LEDs	speaker, 3 LEDs	Digital Display, 2 LEDS, buzzer	3 LEDs
Onboard user input	none	3 pushbuttons, 1 reset	Thumbwheel, Start/Stop	None
Dimensions	9"x5"x2.7"	2.5"x3"	3.2"x2.8"	Waysmall: 83x36x15 mm, Robostix: 80x35 mm

Except for the low priced OOPic, each processor alternative is comparably priced at just under $200 dollars. This is twice the estimated processor system cost in the group's budget. Aside from cost, there are a number of necessary characteristics that the robot's processor must possess.

2.3.1 General I/O

The number of I/O pins that the processor must have is dependent on the number of sensors and servos/motors that the robot will have. The group has estimated that the robot will need a processor that is capable of handling the following I/O:

- 3 PWM or analog input signals from proximity sensors
- 5 Digital input signals from the line sensor array
- 2 digital input signals from the wheel encoders
- 2 PWM output signals to the drive system
- 1 PWM output signal to the can manipulator

It is important to note that one of the image recognition system alternatives, the CMUCam2, is capable of outputting signals to drive servos. If this image recognition system is chosen, it will reduce the number of output signals that the main processor needs to provide. Assuming that the CMUCam2 is chosen, each processor alternative appears to meet the general I/O requirement. The Gumstix appears to have the best I/O capabilities.

2.3.2 Serial I/O

In addition to the I/O capabilities mentioned previously, the system will need one serial bi-directional connection to communicate with the image recognition system. Because the OOPic system does not support the serial connection needed to communicate with the image recognition system, the OOPic is no longer a processor option.

2.3.3 Programming

Writing software for the processor of the robot will account for much of the work that will be done on the project. Choosing a programming language and environment which is familiar to the members of the group is ideal in that it will reduce the number of hours spent learning new concepts and syntax. Every member of the group is familiar with the C programming language. This language is adequately sophisticated for the robot's logic; therefore it is the ideal programming language choice. Currently, the only processor that can be programmed in C is the Gumstix Waysmall.

Also, the robot's processor will need to store a reasonable amount of variables in memory; therefore, a healthy amount of RAM is important. It is currently unknown specifically how much memory will be needed, but this amount will increase dramatically if the group decides to use raw capture data from the imagining sensor system (RGB color values for select pixels of each frame of a video stream). Again, in this area, the Gumstix Waysmall is far ahead of its alternatives.

2.3.4 Processor System Conclusions

From the information in Table 2, the Gumstix Waysmall appears to be the clear winner for the robot's processor. It is the smallest of the processors and provides the needed I/O and memory to meet the group's requirements. After considering all of the mentioned factors, the Waysmall was purchased and chosen as the Falcon's processor.

The one problem which remained with the Waysmall was its lack of onboard I/O, which is typical for robotics projects. Unlike the other systems, it is not an "all-in-one" solution. The Waysmall is not intended to be a compact, singular solution for robotic intelligence. To account for this problem, the group purchased a Robostix expansion board, which would be connected to the Waysmall through parallel communications. This expansion board would serve as an external I/O board that can provide digital and A/D input header pins and powered output header pins for the

Waysmall. This interaction between the two boards would be a great challenge that the group would be forced to overcome before proceeding.

Upon receiving both the Waysmall and the Robostix, the group determined that the Robostix was actually a very powerful processor itself. The group was unaware that the Robostix even had an onboard processor capable of running complex programs. It also included 6 PWM channels, 8 A/D ports, and 25 general I/O pins. After further research, the group concluded that the Robostix itself was sufficient enough to serve as the processor for the Falcon. This eliminated the likely problems that might arise between interfacing the Waysmall and Robostix. Figure 8 shows the general layout of the Robostix Expansion Board.

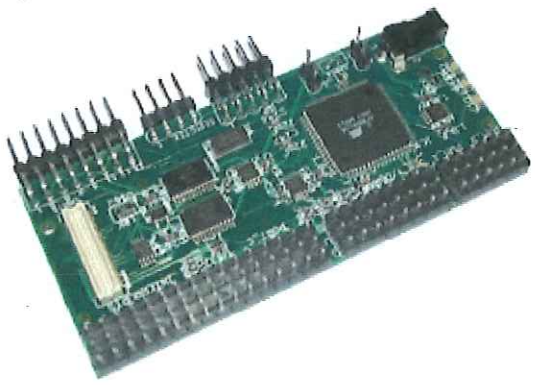


Figure 8: Robostix Expansion Board [5]

2.4 Sensors

The group designed the Falcon to make use of0 four different types of sensors. These sensors include:

- Infrared (IR) sensors for line following,
- Proximity sensors to alert the robot of obstructions,
- Wheel encoders to determine the distance the robot has traveled and the speed at which it is traveling,
- An image sensor to locate and differentiate colored cans.

Alternatives for each type of sensor are presented in the following sections.

2.4.1 Line Following

Almost ubiquitously, robotic line following is accomplished by responding to input signals from IR sensors. These sensors are diodes that are sensitive to light in the infra-red (IR) range, and produce a voltage proportional to the light that they absorb. They come in two varieties, phototransistors and photoreflectors. A phototransistor is a single photo-sensitive diode which detects IR light from an external source. In the case of this robot, an external IR diode could serve as this source. A photoreflector is an emitter/receiver pair; the emitter emits IR light and the receiver absorbs the reflected light. In a typical configuration, an array of these sensors is aimed toward the ground, approximately a centimeter away from its surface. Dark colored surfaces—such as black lines—absorb IR light, thus decreasing the light the IR sensor measures. Lighter surfaces reflect the IR light, causing the sensors to measure a higher level of light. The type and number of IR sensors are design considerations for the robot. Table 3 is a price and power consumption breakdown for various IR sensors found on the Internet.

Table 3: Line Following Options

Type	part #	Manufacturer	Price per unit	Power Dissipation
Photoreflector	P5587	Hamamatsu	\$2.25	160 mW
Photoreflector	QRB1134	Fairchild	\$1.50	200 mW
Phototransistor	L14G1	Fairchild	\$1.72	300-600 mW

Each of the IR sensors in Table 3 outputs an analog voltage which could be read by the processor using an A/D input. Alternatively, the sensor's signal could be converted to a digital signal by connecting it through a digital gate such as a NAND gate. For larger sensor arrays, digital signals are acceptable; however, for a smaller sensor array, digital signals result in jittery line tracking. A small sensor array (two sensors for example) limits the mobility of the robot: the robot has to travel slowly around sharp turns so that the sensors don't lose the line. A longer sensor array, comprised of up to five sensors, allows the robot to travel faster while still maintaining line acquisition. The group decided to purchase the Fairchild QRB1134 sensors due to their low price and relatively small power dissipation.

The line following (IR) sensors were tested using a simple circuit in isolation to ensure that the sensors perform as expected. By placing black tape on white paper, the functionality of the sensors was easily confirmed.

The line sensors were then aligned in a row, oriented downward, and attached to the underside of the Falcon's lower platform. Each sensor is read into the robot's processor using an ADC. This allowed the robot to "see" levels of grey when using the line sensors. This is useful when a line sensor is over the edge of the line it is following. Using ADCs rather than inputting each sensor signal to the processor as a single digital signal allows for much smoother line following. Software was written which inputs the value of each sensor and outputs a single 8-bit number that represents the location—from left to right—of the line under the line sensors.

The line following software component produces two 16 bit signals, average and delta, to be input into the motor software component. These output signals are based on five 8 bit input signals, which are generated by the line sensors mounted below the Falcon's frame. These line sensors are read into five of the Robostix' A/D converters. By analyzing the open-source code available to us online, the group was able to determine exactly how to read in the A/D converters. The delta output signal is calculated based on a weighted sum of the input signals. Equation 1 is the weighted sum that is used in the line following component code.

$$\text{Output} = \left[\frac{64(s2 + 2s3 + 3s4 + 4s5)}{s1 + s2 + s3 + s4 + s5}\right] - 128 \qquad \text{(Equation 1)}$$

In this equation, s1 through s5 refer to the five line sensors under the Falcon. The left most sensor is s1 and the right most sensor is s5. In theory, when the navigation line is under the right side of the sensor bank, Equation 1 will generate a large positive number. When the navigation line is under the left side of the sensor bank, Equation 1 will generate a large negative number. The range of output values for Equation 1 is -128 through +128.

The average signal is computed based on the delta signal. For values of delta close to zero, average is large. For values of delta far from zero, average is small. This allows the Falcon to slow down for sharp turns.

Though initial testing resulted in the Falcon behaving erratically, it was determined that this basic setup could be used to drive the Falcon on the lines. To improve upon this proportional control system, integral and derivative control were implemented. This served to both smoothen out the erratic and jerky behavior and keep the Falcon from overcorrecting to find the line. It improved the overall efficiency of the Falcon and kept it from losing the line or falling greatly out of position when attempting to center itself on the line.

In addition, the line sensors were used to detect intersections throughout the course. Because a weighted sum was not necessary, a much simpler configuration was used to determine when the Falcon had reached the intersection. After extensive trial and error, each possible intersection was mapped out as the line sensors would see it. For instance, on the far left side of the track, the middle sensor would read black along with one of the two outer right sensors. In a similar fashion, each intersection could be identified by the Falcon using the photoreflector sensors. In each situation, the Falcon was aware that it should be on the lookout for an intersection, allowing for their detection to go much smoother. At these intersections, control of the robot was to be passed to the encoder software component, executing actions such as traveling forward, left, or right.

2.4.2 Encoders

Encoders are the primary means by which the robot can determine the exact distance the wheels have traveled, assuming no slippage. This ability is a critical piece of the robot's navigation system. Since encoders far exceed the accuracy of other methods of determining distance (using a timer and a robot speed approximation), the focus is on encoder model selection and the different configurations available. These devices are typically used to measure the distance traveled by the robot, or in some cases, the angle the robot has turned (90 or 180 degrees for example). If this type of input sensor is necessary, then there are a few options available, as seen in Table 4.

Table 4: Encoder Options

	Description	Example	Cost
1	A motor with its own internal shaft encoder can be used. The motor outputs a signal that states the number of shaft rotations [6].	Acroname 19:1 Gearmotor with encoder	$155 each
2	A pair of small photoreflectors and black and white disks to attach the wheels, which would output a high/low for each black/white stripe [7].	Hammatsu encoder and disk combo	$32 for pair
3	A shaft encoder can be used to measure the shaft angle, speed, and direction of the motors shaft. This can be optical or through contact [8].	USDigital Optical Shaft Encoder	$49 each

The first option is very compact, and it includes the motor as well as the encoder. However, the cost is high, and with two of these motors, they may be out of the budget. The second option is very inexpensive and very simple, but it will require some additional hardware or software to count the rotations. The shaft encoder is very sophisticated, and it can provide more information than the simple photoreflector combination; however, it is also more expensive than the current encoder allowance in the budget. After evaluation by the group, option number two has been selected for use in the robot's navigation system.

The group purchased 2.55" Bolster Wheels and Quadrature Encoder Modules, as seen in Figure 9. The kit includes two wheels with the encoder stripes on the inner surface, four rubber traction bands, and two miniature photoreflectors. The wheels provided with this set had to be slightly modified to fit to the servomotors already selected.

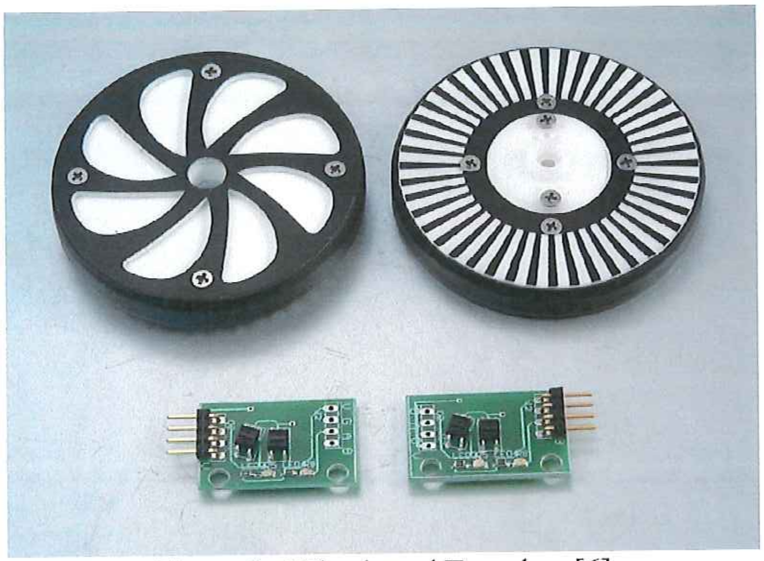


Figure 9: Wheels and Encoders [6]

In the preliminary testing, the encoders worked properly when placed at a distance of approximately 1 cm from the wheel.

The implementation of the encoder software went very smoothly. By generating interrupts at each black-white transition, the Falcon was able to keep track of the distance each wheel had traveled. This was very helpful when making turns and skipping specified distances.

Without the encoders, making turns would have been exceptionally difficult. As mentioned previously, open loop turns are extremely difficult to make when the servomotors are exhibiting unusually inconsistent behavior. With the use of encoders, the group was able to pinpoint exactly how far the servos needed to travel to accomplish the turn. For each turn, the outside encoder was used to track the distance traveled by the Falcon. For instance, left turns utilized the right encoder to determine how far the robot had turned. Exact values needed to complete the turn were determined experimentally and were different for each turn.

A similar setup was used for traveling straight. Both encoders were used simultaneously to drive both straight forward and backwards. This was useful when needing to skip an intersection or backup after dropping a can. Although the group struggled to ever completely achieve a perfectly straight line using the encoders, the encoders did provide consistency when driving the Falcon straight.

2.4.3 Image Recognition System

For the vision system on the robot, some image processing will be necessary. Can colors will have to be distinguished, the location of the can will need to be determined, and the robot will have to discern between a can and a doll. To meet these criteria, the group developed three possible options that meet the needs of the project, which can be seen in Table 5.

Table 5: Vision System Options

	Description	Example	Cost
1	A camera attached to a printed circuit board that does limited image processing in the device. Serial communications used to communicate with CPU [9].	CMUcam2	$200
2	A Smart Camera that can do advanced image processing within its body. Small size, but complicated software and communications [10].	HawkEye 1610	$5,500
3	Cameras available in department (through Dr. Nickels), that do not have image processing as a feature. Additional hardware and software would be necessary for the image processing [10].	Marlin F-046C	Free + cost of image processing HW and SW

As seen in the table above, each system has its advantages and disadvantages. However, the Smart Camera option is not within the budget of the project and is over-qualified for the limited image processing needed in the design. The cameras that are available in the department are free and readily available; however, the complications involved with programming all image processing from scratch may require more time than is available before the competition. The most favorable option is currently the CMUcam2 camera, which can be seen below in Figure 10. All the image processing requirements will be met with this device, and the cost is within the project's budget. Due to availability restrictions, the CMUcam2+, which is essentially the same camera, was chosen by the group.

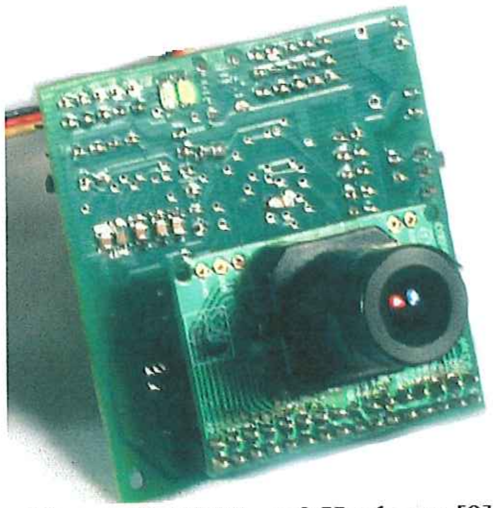


Figure 10: CMUcam2 Hardware [9]

The CMUcam2+ was selected and purchased early to allow for maximum testing time prior to the competition. Initial testing involved powering the device and verifying that the correct LEDs were activated. This particular device has a graphical user interface (GUI) provided by Carnegie Mellon University (CMU) that allows for rapid development with this device. In addition to the GUI, CMU also provides the extensive *CMUcam2 Vision Sensor User Guide* [11] and the *CMUcam2 Graphical User Interface Overview* [12]. The user guide includes testing procedures for the camera to ensure that it communicates properly with a personal computer out of the box. This system can first be accessed through a serial terminal, in which the user sends commands to the camera. For example, typing GV and 'enter' in the terminal causes the camera to send its version number to the serial terminal.

Once the device responded properly through the terminal, the camera was ready to be tested using the GUI. This java program allows the user to send raw images taken by the camera to a PC then use some of the data in the picture (such as RGB values) to perform basic image processing. This functionality is essentially a visual simulation in which the user is able to 'see' from the perspective of the camera as seen in Figure 11. Within the GUI the user can track a certain 'color blob', a solid color region of similar RGB values, and its centroid, check histograms for RGB levels, change servo settings if pan or tilt has been implemented, or track the motion of an object. The use of the GUI allowed the group to better understand exactly what information would be useful to aid the Falcon in locating and identifying cans.

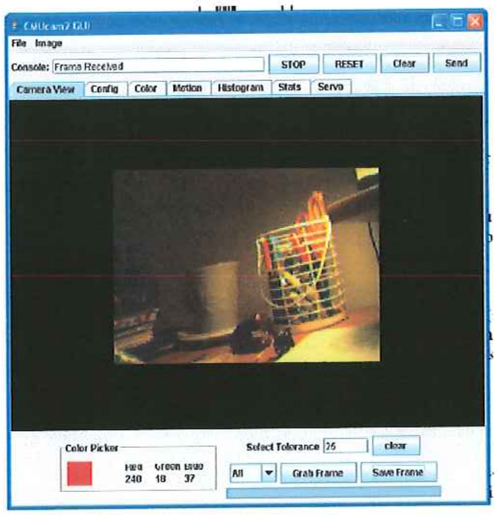


Figure 11: Screen shot of GUI [11]

From the beginning, the group has required that the vision system be capable of performing two major functions. The first of these is the color determination of the can. 'Track Color', or TC, is a simple command used to accomplish this. Using the 'Track Color' command, the camera receives a minimum and a maximum RGB value to track, and it locates the largest 'blob' of color that fits within the specified range. This region of color is then assigned a rectangular bounding box, and the centroid is determined. The camera then serially returns the coordinates of the centroid, the coordinates of the bounding box, the number of pixels present in the RGB range, and a confidence value back to the processor. These values are streamed, so as each frame refreshes, a new bounding box and centroid are sent. This information was parsed from the packet sent by the camera, storing only the necessary information. It was determined that this command is very effective in determining the color of a can, if the group is able to specify the correct RGB ranges for each can color. This task was a difficult one to accomplish due to the specifications handed to us by the rules of the competition and many inconsistencies in the lighting of the cans.

The same command, 'Track Color', was also used to accomplish the second task needed from the camera. The vision system must be capable of locating and tracking the can. The X and Y centroids, which are returned to the processor once the command is sent to the camera, were used to accomplish this task. Using the X centroid, delta was determined for the servos as to center the Falcon on the can. As previously mentioned, new values were streamed once the camera refreshes, serving as feedback and continuously directing the Falcon towards the can. As done in the line following driver, proportional, integral, and derivative control were used to achieve the most controlled and efficient tracking of the can. The Y coordinate was used to control the speed of the Falcon. As the Falcon got closer and closer to the can, the Y centroid of the can gets lower on the camera screen. As it is closer, the Falcon slows down to avoid a fast collision with the can.

The camera was also used in the Challenge Round of the competition to distinguish between the Barbie and the cans. A Barbie is a much taller and slimmer object, with a much higher Y centroid on the camera screen. On the other hand, the can is relatively short, thus yielding a much lower Y centroid. Using this distinction, the Falcon was able to distinguish whether the object in view was either a Barbie or a can. The Falcon would then know which 'blob' to drive towards and which to ignore.

A buffer was used to receive the camera response after each command. The device driver for the camera was then responsible for parsing the data and extracting only the necessary information, which was the centroid and pixel count. Using these values, the falcon is able to identify the color of a can and track it until the Falcon is in position to pick it up.

2.4.4 Proximity Sensors

Proximity sensors are used to detect nearby obstructions that the robot might encounter, such as Barbie dolls that will be placed on the course in the final round of competition. These sensors have many characteristics that must be considered, such as how the sensor should measure distance, how far the sensor should be able to detect an obstruction, and what type of signal it should output to convey its results to the processor. These are the three most relevant considerations when finding the right proximity sensor for a project.

Sonar pulse and infrared detection are two commonly used methods for detecting the distance to an object. Distance is measured using sonar by transmitting an ultrasonic pulse from the unit. The distance-to-target is then determined by measuring the time required for the echo return. IR detectors work in a slightly different manner, emitting a pulse of IR light. This light travels away from the emitter, according to the field of view, and either hits an object or just keeps going. If the light reflects off an object, it returns to the detector and creates a triangle between the point of reflection, the emitter, and the detector. The angles are then used to determine the distance. Either type could work successfully, but a few more factors need to be considered first.

Table 6 outlines the important characteristics of a variety of proximity sensors.

Table 6: Proximity Sensor Options

Detector	Output Type	Range	Estimated Cost
GP2D02	Byte value read serially from device	10cm - 80cm	$46.00
GP2D12	Analog value (0V - ~3V) based on distance measured	10cm - 80cm	$23.00
R93-SRF04	Variable pulse width based on distance from object	3cm – 3m	$25.00

After considering the above information, the GP2D12 was chosen for its low price, acceptable sensitivity range, and analog output. The analog output seems to be the most practical and easiest to work with.

Basic testing was completed to confirm the correct operation of the proximity sensors once they arrived. The sensors exhibited a behavior exactly as expected. The output is an analog signal which varies logarithmically with the object distance. Objects close to 10 cm form the sensor produce voltages around 3 V, while no object produces a voltage of 3 V.

The use of proximity sensors evolved over time as the Falcon was being developed. Originally, the proximity sensors were to serve two purposes on the robot: obstruction detection (simulated human workers) and can placement. The group initially intended for the obstruction detection sensors to utilize interrupts, allowing the Falcon to sense an object while turning and rotate in the opposite direction. A separate proximity sensor, which was to be placed directly under the camera in the center of the robot, was to be dedicated to positioning the Falcon at the correct distance to lift the can.

After careful consideration by the group, it was decided that it was unnecessary to use interrupts to detect obstacles for the Falcon. This technique would require much more complex maneuvers by the Falcon. For instance, the Falcon would need to be capable of detecting an object while turning and force itself to turn a specified distance in the opposite direction. Knowing exactly how far to turn would require the use of the encoders and/or other hardware components. As an alternative solution, the group chose to simply poll the sensors as the Falcon approached the can. By doing this, the Falcon would be aware of any obstacle to either side and would know which way to turn to evade the object. While considering the competition rules carefully, the group determined that this was an adequate technique in detecting any possible obstruction to the Falcon. This required no additional hardware and required only that the Falcon poll the sensors often enough as to not miss detecting an object.

As the project progressed, the can placement proximity sensor was deemed unnecessary. As a substitute, the group opted to use a simple switch to position the can appropriately for pick-up.

2.4.5 Can Placement Switch

The switch chosen for this function provides maximum simplicity. The switch was connected to the Robostix using a general digital I/O port, as opposed to the analog input and external circuitry needed for the proximity sensor. As the Falcon is approaching the can, the Falcon is constantly polling the switch to determine if the can has been found. When the can comes into contact with the switch, the Falcon is informed to proceed with picking up the can. Testing revealed that this technique worked flawlessly in hardware, software, and simulation.

2.5 Drive System

When choosing a drive system for the robot, there were many different options. Foremost, the group had to decide whether to use motors or modified servos. Since both options are feasible, some design constraints were considered. The robot must be able to go forward and backward at varying speeds and must be able to stop very quickly. The robot must also be easy to control and be built within the groups allotted budget. Furthermore, the motors would have to be fast, yet have enough torque to transport the weight of the robot and its cargo. All of these factors were taken into account in order to find a design that met these requirements

First, there are many different types of motors which have different specifications. AC and DC motors are both readily available; however, a portable battery supplies a DC voltage, so DC drive components are preferable. Furthermore, there are many variations of DC motors. For this project, general purpose low power motors are acceptable. A plastic high power gear box kit can be purchased for approximately $15 to $30 online [13]. These kits give the user the opportunity to create specialized gear ratios. Ordinary low power motors range in price from $1 to $30 depending on their various voltages, power ratings, and gear ratios [13]. These motors are very simple and reliable provided the user operates them within their specified limitations.

Another type of DC motor that can be used is the stepper motor. These motors use a set of activated solenoids to rotate the drive shaft. These motors can be found for about $3 to $50, depending on the specifications of the motor [13]. While the stepper motors can provide valuable information about distance traveled, they also draw too much power from a portable battery.

An alternative to using DC motors for the drive system would be using modified servos. Servos normally just rotate a fixed angle and stop within 360 degrees, but they can easily be modified to run continually forward or backward similar to a DC motor. This use of a servo is commonly used in small robots. Servos can be purchased for $10 to $250 depending on their specifications [14].

The group selected two GWS Micro 2BBMG "Mighty Micro" Ball-bearing Metal Gear Servo Motors that will be used to drive the robot once they are modified for continuous rotation. A picture of this servo can be seen below in Figure 12 along with the technical specifications in Table 7 [13]. These motors were first inspected to see if the group had in fact received the correct model and that they were not damaged in any way. They were also tested to see that they do in fact work when power and a pulse width were supplied. Initial tests indicated that the servomotors were in working order.

Table 7 – Manufacturer's Specifications of the Drive Servos [13]

Dimensions:	1.1 x 0.55 x 1.17 in. 28.0 x 14.0 x 29.8 mm
Weight:	28 Grams / 0.98 oz.
Ball Bearings:	Yes, 2BB
Metal Gears:	Yes
Torque (4.8V):	75 oz.in.
Transit Time (4.8V):	0.17 sec./60°
Torque (6.0V):	89 oz.in.
Transit Time (6.0V):	0.14 sec./60°

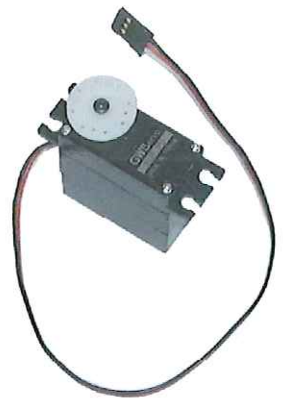


Figure 12: "Mighty Micro" Drive Servo

The next step was writing code to find the zero point of the servo motors. This code simply sent the motors a signal, which in turn rotated the motor shaft. The pulse width of the signal was modified to find the exact pulse width that yielded no movement in the motors. This was done independently for both servos. This no-rotation pulse width was the basis of driving the motors. Varying the pulse width from this position will cause the servos to rotate in either direction and at varying speeds.

The motor software component was designed to control the speed and direction of the Falcon based on two input signals, average and delta, and two pulse-width modulated (PWM) output signals, one for each servomotor. Average is a 16-bit signed integer, which increases or decreases the speed of both servomotors simultaneously. Negative values of average cause the servomotors to spin in reverse. Delta is a 16-bit signed integer that, when given a positive value, increases the speed of one servomotor while decreasing the speed of the other. Greater values of delta cause the Falcon to make sharper turns than smaller values. Positive values of delta cause the Falcon to turn to the right; negative values cause the Falcon to turn left.

Because of component variability in the servomotors, each servomotor responds differently to the same input signal. This means that the Falcon cannot be driven using open-loop control. Using average and delta input signals, as opposed to independent signals for each servomotor, allows the direction and speed of the robot to be more easily controlled based on feedback from the line sensors, encoders or the vision system. The servos rely on feedback from sensors.

After extensive trial and error testing, it was determined that the motor software component was working adequately. To make this assessment, the average and delta signals were assigned to the values of two A/D converters on the Robostix board. Potentiometers were connected to each A/D channel and, using these pots, the Falcon was "driven" forward, back, left, and right at various speeds. Favorable results were obtained, allowing the group to move forward in developing code.

Various adjustments were made throughout the testing process to further improve this system. For instance, the servos needed to be "re-zeroed" after a month or so of testing to keep the servos from rotating when the pulse signal is directing it to remain still. Also, it was important to keep in mind that each servo is unique in its response to identical pulses. For example, when making a 90° right turn, the delta signal must be higher than when trying to make the same left turn. This is due to the non-uniformity in servomotors. Exact average and delta values were determined by running the Falcon over and over again on the mock course.

2.6 Power System

The selection of a battery was one of the last selections made, for it depended greatly on the different components chosen for the robot. There are a variety of types of batteries available, each possessing its own advantages and disadvantages. Four of the most relevant battery types are Nickel Metal Hydride (NiMh), Lithium Ion (LiOn), Nickel Cadmium (NiCad) and Sealed Lead Acid (SLA).

Each battery type is slightly different than the others, with major differences occurring in price, cycle life, and energy density. Table 8 gives a brief outline of the differences between the different battery types, highlighting their major differences.

Table 8: Comparison of Battery Types [15]

Battery Type	Energy Density [W/kg]	Cycle Life	Cost	Environmental Impact
Nickel Metal Hydride (NiMh)	50 to 80	300-500	$60 (7.2V)	Relatively low toxicity, recycle
Lithium Ion (LiOn)	100 to 150	300-500	$100 (7.2V)	Low toxicity, relatively harmless
Nickel Cadmium (NiCad)	35 to 57	1500	$50 (7.2V)	Highly toxic, harmful
Sealed Lead Acid (SLA)	25 to 35	200-300	$25 (6V)	Toxic lead and acids, harmful

Overall, any one of the four battery types could potentially be used. Each battery has its own unique advantages and disadvantages. For this project, many of these factors have little impact on the group's decision. For instance, cycle life may be important in an industrial setting; however, the group had no intention of recharging the battery more than 10-20 times. As a result, the group chose two 9.6 V, 1600mAh NiMh battery packs, which are commonly used in RC cars. These batteries are lightweight, rechargeable, relatively inexpensive, and provide more than enough current to power the robot for extended periods of time.

The selection of the power system came only after each component had been fully tested. It was determined that the Robostix and components, including all of the sensors and servos, drew a maximum current of approximately 850 mA. To power these components, a single battery pack was dedicated to this circuitry, providing more than enough current and power. The servos themselves, which were expected to draw the most current, drew a maximum of around 400-500 mA. To account for these high current components, a separate battery was also used here. None of the components required a voltage higher than 9.6 V, making this battery selection the ideal selection for the Falcon. While one battery could have possibly been sufficient in powering the Falcon entirely, it was determined that buying extra batteries was a much safer solution.

The power system seemed to be reliable throughout the testing of the Falcon. Wall power was used extensively for most of the testing, prior to purchasing the batteries. To extend the group's testing capabilities, an extra set of two batteries was purchased. An RC battery tester also allowed the group to monitor the status of the batteries both after charging and throughout testing.

2.7 Prototype Falcon

The first step the group took in creating a prototype Falcon was to discuss and model what the robot will look like based on the functions it has to carry out. Here the group discussed design ideas, chose the best one, and sketched it. A model of the Falcon created in Pro-Engineer can be seen in Figure 13 below. A more precise sketch was then drawn to scale on a piece of cardboard and included all the components in their appropriate places.

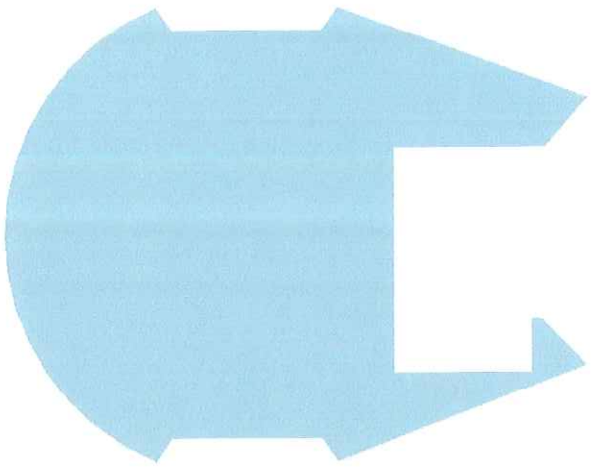


Figure 13 : The prototype Falcon body design

Once the design was sketched, it was then physically constructed. The design originally called for a metal body, but the group decided that it would be easier and more cost efficient to build the prototype out of ¼" plywood. The wood was then cut with its dimensions matching the cardboard diagram.

The servos were mounted under the Falcon using aluminum brackets as seen in Figure 14 below. Once the servos were securely fastened, the wheels were screwed onto the servos. Next, a metal bracket was constructed to house the line following photoreflectors. The bracket, which can be seen in Figure 15, was then attached to the underside of the Falcon. Once mounted, the 5 line following photoreflectors were bolted into place on the bracket. Bolts were used so that the position of the photoreflectors could be adjusted. This enables the group to put them closer or farther from the ground depending on their ability to read the black lines necessary for navigation.

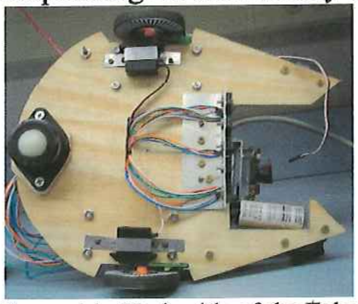


Figure 14: Underside of the Falcon

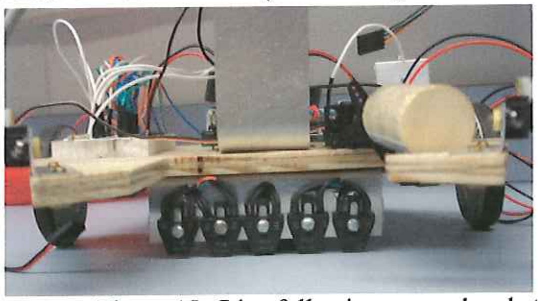


Figure 15: Line following sensor bracket

The next component mounted was the rear caster. The caster was mounted under the rear of the Falcon as illustrated in Figure 14. After the caster was in place, aluminum brackets were constructed for the proximity sensors. These proximity sensors, shown in Figure 16, were then mounted onto the front tips of the Falcon.

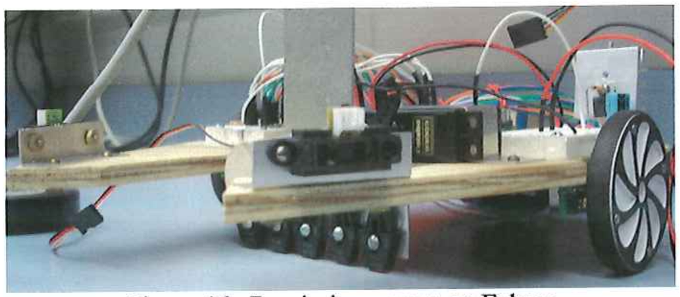


Figure 16: Proximity sensors on Falcon

A sliver was cut into the Falcon for the encoders to mount vertically to the side of the Falcon. Since the body of the Falcon is all wood, there is no fear of short circuiting the encoder printed-circuit-board (PCB). Using this technique, the group was able to mount the encoders close enough to the wheels for the sensors to work. This also positioned the encoders under the body of the robot, blocking out a most of the ambient lighting.

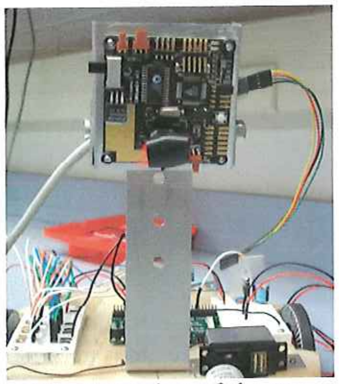


Figure 17: Front view of the camera mount

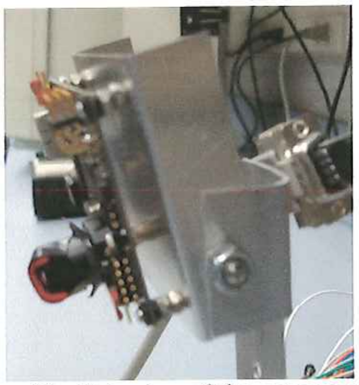


Figure 18: Side view of the camera mount

The camera mount, which can be seen in Figures 17 and 18 above, was built with the option of having the camera at varying heights and angles, one combination of which will be chosen after the camera testing is completed.

After finishing the camera mount, additional circuitry was added to the robot. Two small protoboards were mounted to the top of the Falcon. The group's processor was also temporarily fastened between the protoboards. A slit was then drilled into the Falcon so that all the wires from the photoreflectors and the drive servos under the Falcon could be routed to the protoboards and processor on top of the Falcon. All of this circuitry can be seen in Figure 19 below.

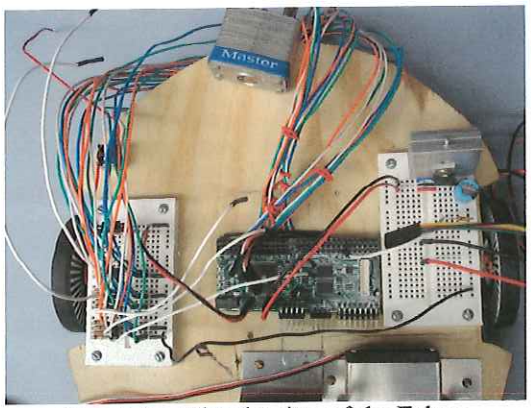


Figure 19: The circuitry of the Falcon

The next component added to the Falcon prototype was the can manipulating device. The group's design consisted of a cam mounted to a servo. The servo was mounted on the top of the Falcon in its allocated position using an aluminum bracket. A wooden dowel with a 1" diameter was used as the cam. The wooden dowel was cut to its designated length and nailed to the servo mount in an off-centered fashion creating the cam. The servo mount was attached to the servo creating a can manipulator that was ready to pick up and drop off cans as illustrated in Figures 20 and 21 below.

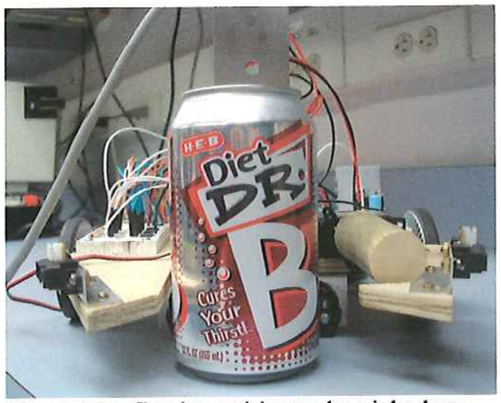


Figure 20: Can in position to be picked up

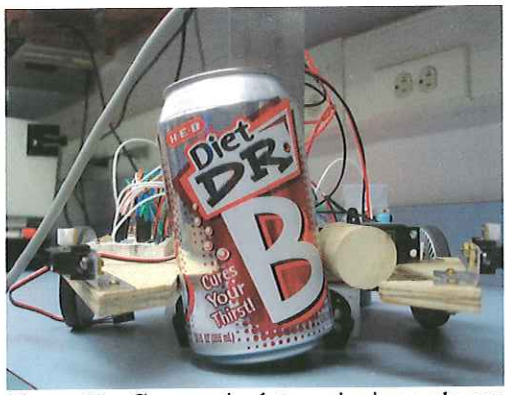


Figure 21: Can manipulator gripping soda can

Finally, the power system was added to the Falcon. The two batteries were mounted to the rear of the Falcon using Velcro. The batteries were then wired to a switch which will be used to turn the Falcon on and off.

As the construction of the prototype was almost complete, the Falcon needed a place to begin testing itself, both in hardware and software. For this reason, a mock-up of the competition course was constructed. The construction of the course required two pieces of 4x8 ft plywood, a bucket of black paint, a bucket of white paint, and some paint brushes. The two plywood pieces were painted white. Then the black paint was used in conjunction with black electrical tape to create an exact duplicate of the course that the Falcon will be competing on in April. This will allow the group to test the Falcon on an actual course, enabling the group to perfect the Falcon's physical design and the programs that will be controlling it.

After great consideration, it was determined that the prototype Falcon possessed every characteristic needed to be used as the competition robot. All of the hardware components had been properly placed and needed no further improvements. The Falcon was undamaged and had been successful throughout the testing process. Furthermore, it was determined that any variation in the new body might lead to inconsistencies in the performance of the Falcon. As a result, the group determined that it was not necessary to create a new Falcon to use in competition.

3. High-Level Software Development and Testing

3.1 Subprograms

Once each of the device drivers were implemented and functioning properly, subprograms were constructed to perform different simple tasks that the Falcon would complete. As an example, GET CAN is a subprogram which is called once the switch is set, indicating that the can is in place to be grabbed. In this subprogram, the Falcon activates the can manipulator, picking up the can. It then turns around 180° and travels back to the line until the line sensors are centered on the line. This subprogram requires the use of the can manipulator, encoders (the 180° turn), servos, and line following sensors. Once each of the drivers has been successfully coded, a subprogram like this one requires that the Robostix coordinate the different actions to be completed by each component. The subprograms were divided in a fashion which allowed for universal use. For example, GET CAN could be called at any time by the main program when picking up a can. There are no alternate subprograms, like GET CAN BLUE or GET CAN RED.

Other subprograms were written to perform actions such as dropping off cans, making turns (90°, 180°), traveling to the cans and so forth. Because the group insisted upon using modular programming techniques throughout, the construction of these subprograms was simplified greatly, as it is relatively easy to follow from an outsider's point of view. To view any and all of the subprograms, refer to the file Subprograms.c file, which is included on the attached CD.

Because the subprograms were so important in the overall functioning of the Falcon, extensive testing was completed to ensure that each subprogram was properly functioning. By testing each device driver prior to constructing the subprograms, the group was able to more accurately pinpoint the source of various coding problems. Serial communication was used in conjunction with an interface called Kermit to monitor and debug the code as it ran on the Falcon. The group was able to print various statements onto the monitor within the code to follow the program and monitor the status of different variables. This was very helpful in pinpointing problems and coding errors. Each subprogram was thoroughly tested before moving on to the next subprogram and the main logic.

3.2 Main Logic

The purpose of the main logic is to map the path of the Falcon around the course based on the various placements of the cans. The logic should consistently work for each and every possible can placement scenario. When first designing the main logic, the group considered a few directions to go. The first solution the group considered was devising a series of case statements to determine the path of the Falcon. This would involve a series of very involved case statements, which would limit the future possibilities of expanding the capabilities of the Falcon. Alternatively, the group decided to be more original and create a logic scheme which would be much more robust and much less convoluted.

This logic scheme does not define the path that each scenario would call for; conversely, the path is calculated. In this system, each room is labeled 0→3, with the incoming and outgoing rooms across from each other having the same value. The red room is 0, the blue room is 1, the green room is 2, and the yellow room is3. When a blue can is detected and picked up, the Falcon then knows that the can belongs in room 1. The Falcon is also always aware of the room number it is currently in. The path is then calculated by subtracting the destination room from the current room. For instance, if the blue can was found in room 0, then the path becomes -1 (0-1=-1). The path values -3→3 all have instructions indicating exactly how to travel to the destination room. This method provides an easy way to determine the path necessary to drop off the can.

Once the can has been dropped off, the main logic then determines which room, with a can, is the closest. This is done using a similar subtraction method. Different variables indicate to the Falcon which rooms have already been cleared and which rooms remain full. For a more in depth look at the main logic, please refer to the file Main.c, which is included on the attached CD.

It is important to note that the main logic is very robust and could be expanded to include an infinite amount of rooms. By using a mathematical approach to determining the path, the group has avoided unnecessary case statements, which grow exponentially when more rooms and cans are added to the course. After extensive testing, the main logic appeared to direct the Falcon in the right direction 100% of the time. The only errors reported can be attributed to the inconsistencies in the can color determination.

3.3 Optimization

Once it appeared that the subprograms and main logic were performing adequately, it was time to optimize the Falcon to complete the course most efficiently. Many small optimization fixes were implemented throughout the development of the drivers, subprograms, and main logic. Additionally, the group made final touches on the subprograms to ensure that the Falcon would complete the course in many trouble areas.

While the group attempted to increase the speed of the Falcon by increasing the voltage provide to the servos, the Falcon actually became much more inconsistent and had problems dissipating the heat provided by this increased voltage.

Other attempts to optimize the Falcon were aimed at improving any problem areas that the Falcon had exhibited. For instance, on a few of the 90° turns, the Falcon was either over-rotating or under-rotating, based on the point of detection of the intersection and the angle at which the Falcon came into the intersection. To fix this, the group decided to abandon using only the encoders to make the turns. Instead, the encoders were used in conjunction with the line following sensors. The Falcon was instructed to turn until it found the line again. This fixed the occasional problem where the Falcon would miss the line.

Another programming technique allowed the group to fine tune the Falcon much more easily. Every constant, including those which determined the driving speed, turning angles, and delay periods, were lumped together in the header file. It was very simple to make minor adjustments to the speed and turns by merely accessing the header file and changing a single number. Digging through the programs would have increased the complexity of making such a minor change. While these changes were only minor, they allowed the group to have a great deal of control over these very important variables.

4. Results: The Competition

On Saturday, April 8, 2006 the A.R.T.U. design group and the Falcon won 4th place out of 37 registered teams in the IEEE Region V Robotics Competition. The Falcon was one of three robots to successfully complete the course in both the regular round and challenge round.

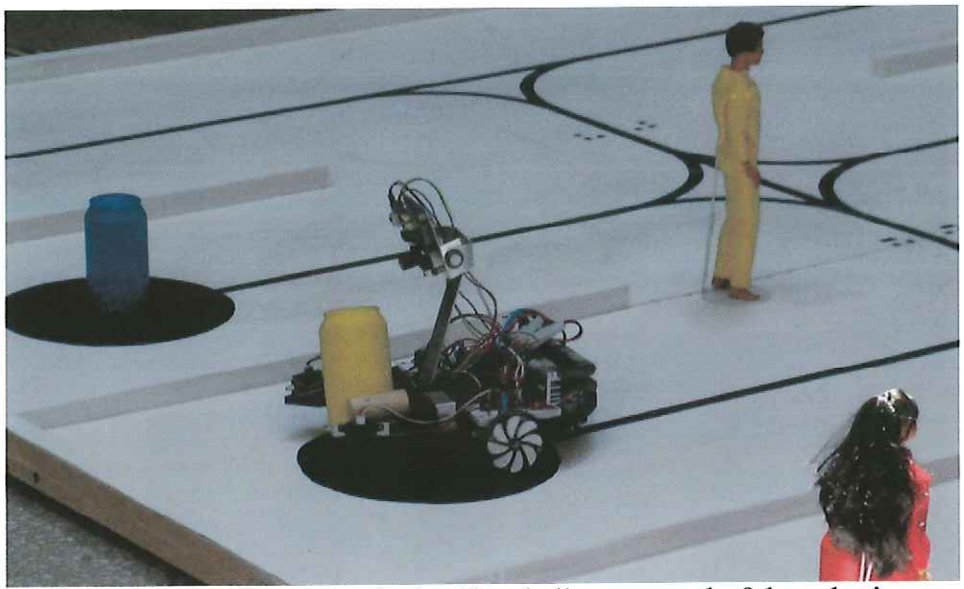


Figure 22: The Falcon loading a can during the challenge round of the robotics competition

4.1 Regular Round

In each round, the Falcon was allowed two runs of the course. In the regular round, the Falcon failed to make a wide enough turn at an intersection half way through completing the course. The design team had never observed this behavior in testing. After analyzing the video recording of the run, the team decided to modify the line-sensor intersection detection software to allow for more cases in which an intersection would be identified. During the second run of the first round of the course, the Falcon successfully delivered all of the cans to their respective room in 125 seconds. This time put A.R.T.U. into 6th place in the regular round of the course. Only the top six teams successfully completed the regular round of the competition, and each of those teams (including A.R.T.U.) advanced to the challenge round.

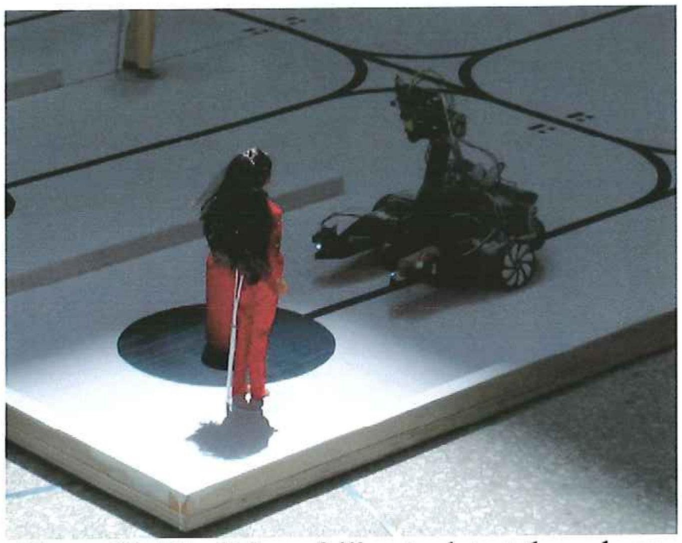


Figure 23: The Falcon failing to detect the red can

4.2 Challenge Round

In the Falcon's first attempt of the challenge round, high intensity light from the skylight above the challenge course interfered with the CMUCam2 aboard the Falcon. This caused the Falcon to fail to detect the red can. Figure 23 shows the Falcon failing to detect the red can during this run. In the second run of the challenge round, a member of the audience blocked the interfering sunlight and the Falcon successfully completed the course in 125 seconds. In both challenge round attempts, the falcon successfully avoided the Barbie® dolls added to the course. The arrangement of cans and dolls for the challenge round was kept the same for each team and is shown in Figure 24. The Barbie® dolls, which are represented by diamonds, were placed far enough behind the cans that the proximity sensors did not trigger and no obstruction avoidance was necessary.

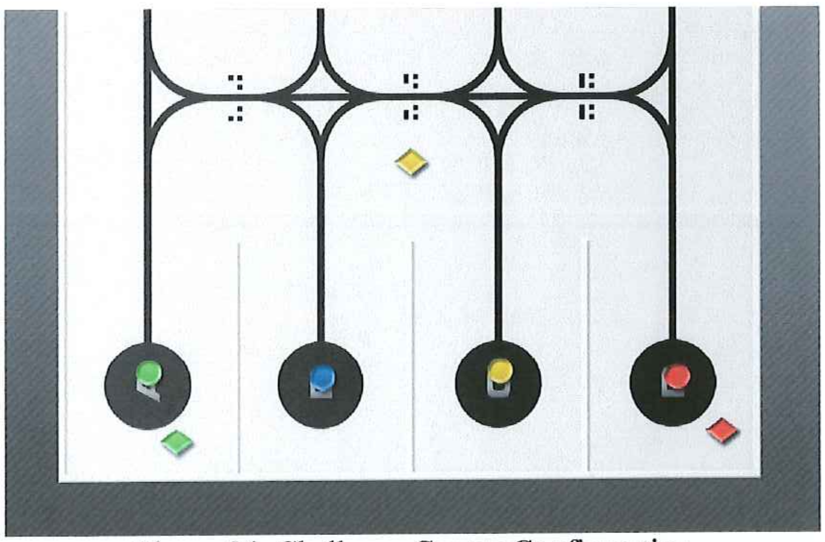


Figure 24: Challenge Course Configuration

5. Conclusions and Recommendations

The Falcon competed in the 2006 Region 5 Conference and performed very well against other teams from around the nation. The design fulfilled all criteria, and the final time of 2.08 minutes was well under the 3 minute limit. All design criteria was met, and the project was a complete success. Compared to other robots at the competition, the Falcon was fairly slow but very accurate. The group noted several future improvements that would enhance the robots performance.

The most obvious improvement to the design would be to replace the servos with DC motors, as they have much greater potential for faster speeds. Very few other robots used servos in their drive systems, and the team realized that while servos are less power-consuming than motors, speed was the most important aspect in the competition. The switch to DC motors would require modifications to the hardware as well as the software, but it should be constrained to the driver level, due to the modular design of the software. A valuable lesson learned is the importance of modular programming, as it allows one problem to be fixed without causing errors in other systems.

Another significant improvement to the design would be to have an automatic calibration of the robot. If the encoders could be used to steady and zero the drive sensors, and if the camera had a method of automatically white-balancing itself, the robot could be much more self-sufficient. Also, the regulators and camera seemed to be negatively affected by poor heat dissipation. The longer the camera and robot ran, the less accurate it would become. Perhaps a fan or better quality heat sinks could improve the design.

The group also learned many valuable lessons from mistakes that could be avoided in the future. One suggestion is to double check all competition specifications to ensure that the test course and robot are properly constructed. Similarly, it is not always a wise decision to rely on open-source documentation for programs. The group chose an open-source program to program the Robostix that claimed that it supported the programming cable in use. However, this was incorrect, and a regulator on the board was destroyed, causing the group to need to reorder the processor. This leads to the other suggestion of ordering parts early and ordering spare parts if the budget will allow it. The dead processor caused a large delay in the software development, and a spare would have been very helpful. The group also learned that hardware sometimes may have odd behaviors that have to be accounted for, so being aware of the limitations of the hardware allows the software to be more robust. Error correction is vital for getting the robot back on course if anything unexpected occurs, although it is sometimes impossible to correct for every circumstance that may occur.

These lessons were often discovered through numerous hours debugging software and will be very valuable to each group member as they progress in their careers as engineers. The purpose of senior design projects is not only to practice technical skills, but also to give students lessons on the reality of errors and mistakes that will probably be encountered in their futures. This design project was a great success, not only in the status earned at the competition, but also in the knowledge gained by each group member that will assist them in all future endeavors.

6. References

[1] Robodyssey Systems, LLC, "Robotic Parts and Mechanical," http://robodyssey.com/.

[2] Bartlett, Jonathan. *2006 IEEE Region 5 Student Robotics Competition Rules and Course Description.* http://www.2006ieeer5conference.com

[3] Savage Innovations, "OOPic The Hardware Object," http://www.oopic.com/.

[4] Ridgesoft, "IntelliBrain," http://www.ridgesoft.com/intellibrain/intellibrain.htm.

[5] Gumstix, "Waysmall Computer Systems," http://www.gumstix.com/spexwaysmalls.html

[6] Acroname Easier Robotics, http://www.acroname.com/

[7] RoboticsConnection.com, http://www.roboticsconnection.com/

[8] Allied Vision Technologies, "AVT Marlin," http://www.kamery.sk/avt.html

[9] The Robotics Institute at Carnegie Mellon University, "The CMUCam2," http://www.cs.cmu.edu/~cmucam/cmucam2/

[10] RVSI Acuity CiMatrix, "Machine Vision Products," http://www.rvsi.net/HawkEye%201610.html

[11] The Robotics Institute at Carnegie Mellon University, "CMUcam2 Vision Sensor User Guide," http://www.cs.cmu.edu/~cmucam/cmucam2/downloads.html

[12] The Robotics Institute at Carnegie Mellon University, "CMUcam2 Graphical User Interface Overview," http://www.cs.cmu.edu/~cmucam/cmucam2/downloads.html

[13] Jameco Robot Store, "Motors and Mechanical Hardware," http://www.robotstore.com/catalog/list.asp?cid=22

[14] Futaba, "Servos," http://www.futaba-rc.com/servos/

[15] Rose, Alexander, "Batteries," http://www.longnow.org/rhino/BatteryPrimer.htm

Appendix A: User's Guide

1. Place the Falcon inside of 'Red' Room (shown as Room 1 below in Figure A.1), and center the line sensors over the line as seen in Figure A.2.

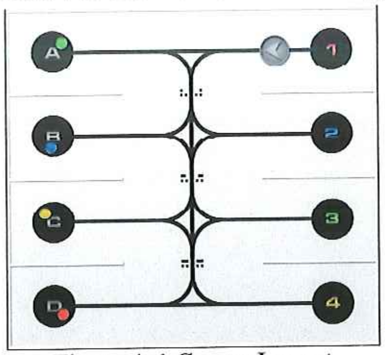


Figure A.1 Course Layout

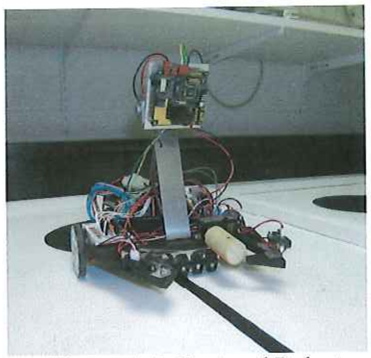


Figure A.2 Centered Robot

2. Place one bright green, blue, red, or yellow can in each of the rooms on the opposite side of the course, shown above in Figure A.1 as rooms A, B, C, and D. The order of the can placement does not matter, the robot should be able to sort any arrangement.

3. Hold medium grey calibration sheet in front of the camera at an angle of about 45 degrees, as seen in Figure A.3, while performing step 4.

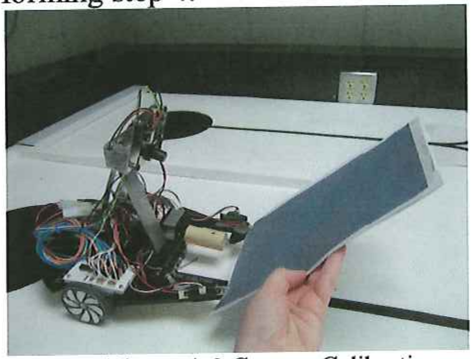


Figure A.3 Camera Calibration

4. Turn on camera power, board power (blue wires), and motor power (orange wires) in that order. See Figures A.4 and A.5 for reference.

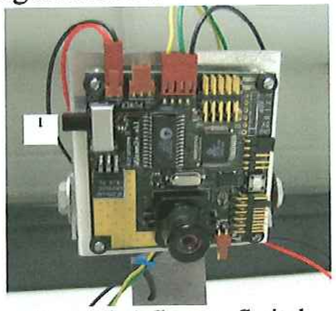


Figure A.4 Camera Switch

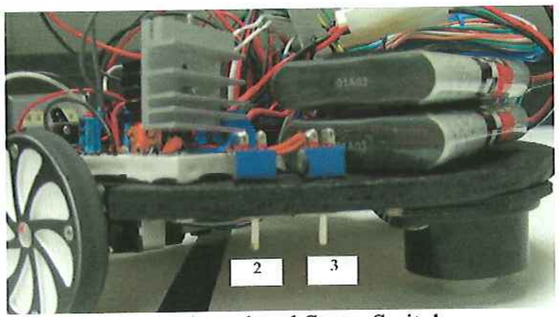


Figure A.5 Board and Servo Switches

5. Wait for the blue LED to turn on and then off before removing the grey sheet. The Falcon will move on his own, so stand back to remove shadows and watch it sort the cans.

Appendix B: Budget

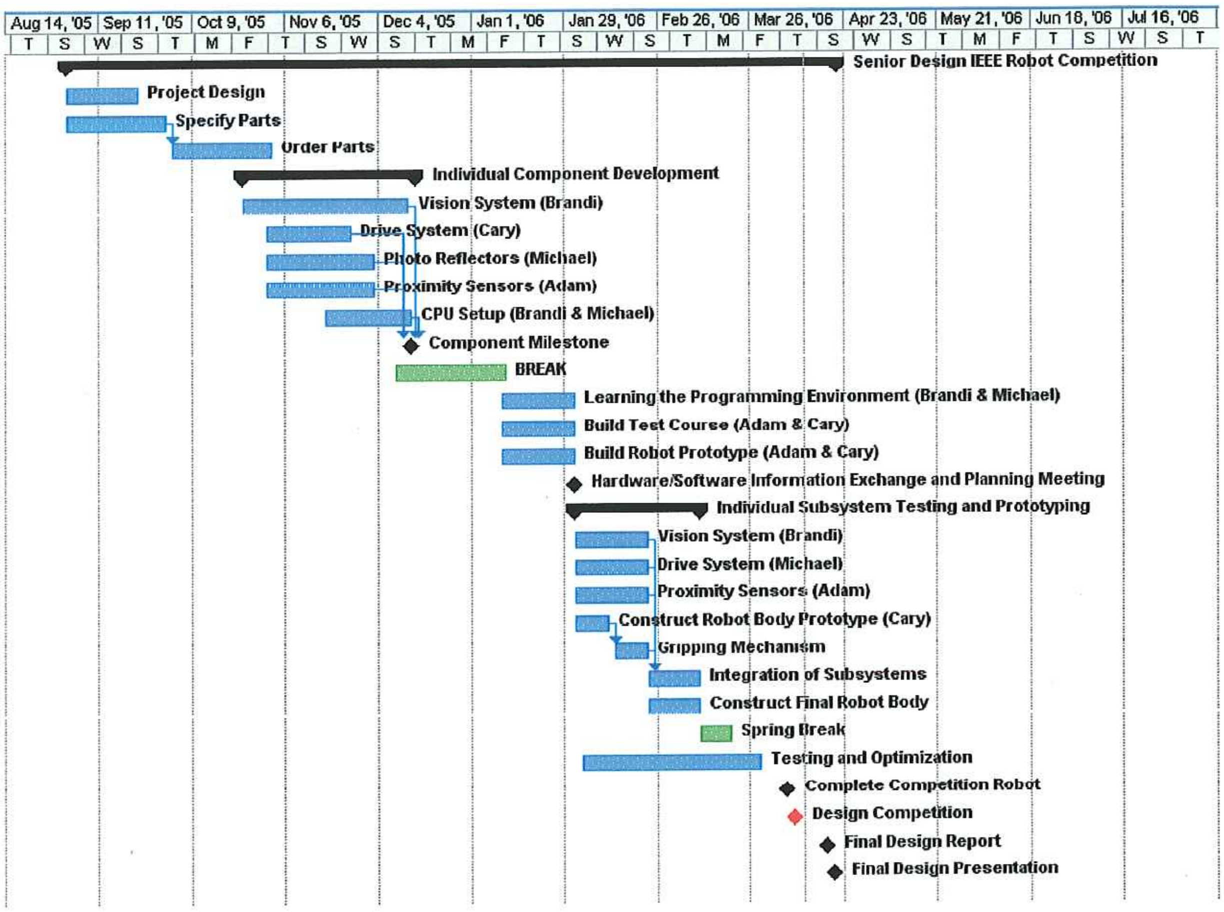
Table B.1 Wages for Team Members

	Hours (1st quarter)	Wage (1st quarter)	Hours (2nd quarter)	Wage (2nd quarter)	Hours (3rd quarter)	Wage (3rd quarter)	Hours (4th quarter)	Wage (4th quarter)	
Cary Wong	56	$30/hr	32	$20/hr	66.5	$20/hr	43	$20/hr	$4,510
Michael Hohimer	72	$20/hr	51	$20/hr	148	$30/hr	106	$20/hr	$9,020
Adam Crouch	59	$20/hr	43.5	$30/hr	76.7	$20/hr	76.8	$20/hr	$5,555
Brandi House	61	$20/hr	53.5	$20/hr	114.5	$20/hr	101	$30/hr	$7,610
Total									**$26,695**

Table B.2 Purchases for Robot

Store	**Item**	**Anticipated Quantity**	**Anticipated Costs**	**Actual Quantity**	**Costs Incurred**
Acroname	CMUcam2+	1	$200.00	1	$175.95
	TTL to Serial Converter	1	-	1	$23.95
Mark III	Fairchild QRB1134 IR Photoreflector	7	$20.00	7	$10.50
	3-pin JST Cable for Sharp Sensors (12 inch)	3	-	3	$3.30
	Sharp GP2D120 Distance Measuring Sensor	3	$35.00	3	$24.75
	Mighty Micro Ball-bearing-Metal Gear Servo Motor	2	$100.00	2	$38.00
Robotics Connection	2.55" Botster Robot Wheels and Encoder Module Combo	1	$50.00	2	$69.67
Gumstix	Waysmall	1	$120.00	1	$129.00
	Robostix	1	$80.00	3	$176.28
Home Depot	Mock Up Course Supplies	-	$60.00	-	$62.78
Radio Shack	Batteries and Accessories	1	$60.00	4	$98.38
Michael's	Can Manipulator	1	$50.00	1	$3.23
	Miscellaneous	-	-	-	$48.96
	Body	1	$125.00	1	$0.00
		Total	**$900.00**		**$864.75**

Appendix C: Timeline



Appendix D: Division of Labor

Cary Wong was responsible for primarily hardware selection and body construction. Optimizing the weight of the body and spacing of components resulted in a compact design. He constructed the mock-up course for testing, and he also made component upgrades and adjustments to maintain the structural integrity of the Falcon. He contributed a total of 197.5 hours to this project.

Brandi House was mainly responsible for writing and debugging software. Her most significant contributions were in the developments for the camera and image recognition. She contributed a total of 330 hours to the project.

Michael Hohimer made significant contributions in software development, understanding the embedded architecture, and navigating the Linux programming environment. His specialty was mainly the drive system, where he implemented PID control for line-following and camera tracking. His time contribution was a mere 377 hours.

Adam Crouch worked in both hardware and software, though much of his time was spent developing the main logic and proximity sensor Barbie detection code. He worked with all major software components to make improvements and debug problems. He contributed 256 hours to the project.

Appendix E: Design Concepts

1. Establishment of Design Specifications and Criteria

   The design specifications and criteria are set forth in the 2006 IEEE Region V Student Robotics Competition Rules and Course Description document (revision 3). This document specifies the scope of the project, as well as the specifications for both the robot and the competition course.

2. Analysis

   Prior to constructing the robot in its entirety, analysis will be completed on the various subsystems of the robot by repeated testing and simulation. Each component's advertised specifications will be tested independently. Software drivers for each system can only be tested with the involvement of the processor. This analysis will enable the group to have a firm understanding of the various subsystems prior to integration. Once all the subsystems perform acceptably on their own, they will be integrated with each other through the central processor of the robot.

3. Synthesis

   The subsystems will be integrated together based on the knowledge gained through analyzing the individual subsystems and their communication abilities. While synthesis of the robot's physical components is important, the individual software that controls each one of these systems must also be integrated in a non-destructive fashion. For instance, sub-systems that use the same processor resources can not run simultaneously, and the processor resources must be freed before another sub-systems can use them.

4. Health and Safety

   According to the rules of the competition, the robot will receive time penalties for harming the 'human workers' on the course. The robot must not damage the course. If a judge decides it is unsafe to the course and the people around it, the robot will be disqualified. In an industrial setting, the health and safety of human workers in the proximity ought to be of the utmost concern. Health and safety must be taken into account in component selection such as batteries, body material, and can manipulator.

5. Social, Political and Environmental Considerations

   The course simulates an automated warehouse, and if this scenario were to become a reality, many jobs could be replaced. This has been an increasing concern for manual workers, as robotics continues to become more sophisticated and readily available. The robot must be able to coexist along with coworkers, human or robot. Component selection, such as battery choice, must take into account environmental regulations.

6. Construction

The construction of the robot will encompass integrating various electrical subsystems together. This will occur only after each subsystem has been analyzed and understood. A body will be shaped that will house the many components of the robot. This body must also be optimally designed for the given design specifications. The body construction would need to be easily repeatable if this robot were to be manufactured for industrial purposes. Refer to Section 2.1 for a more thorough discussion on the body design.

7. Testing

Much of the testing of the robot will be done on the various subsystems of the robot. Each subsystem must be evaluated to better understand its functionality and characteristics. In addition to testing the mentioned subsystems, extensive testing will be necessary in the actual programming used to control the central processor in the robot. Oscilloscope traces will also be used to make sure the proper signals are being sent to and received from each component. Although computer simulation of the robot's systems would be helpful, the development tools were infeasible to implement with the given budget and time constraints of the project.

A mock-up course will be built that exactly matches the design competition specs provided. The robot will then undergo a series of tests to determine whether or not it will be able to perform as expected. Each of the subsystems will again be critically monitored to determine their performance. Other aspects that will be closely watched are the speed and efficiency of the robots movements as well as the structural integrity of the body itself. This testing will indicate the strengths and weaknesses of the robot and lead to the modification and optimization of the design. Final testing will occur at the competition site using the official cans and course. This permits the group to calibrate the camera to account for any discrepancies.

8. Evaluation

As mentioned above, each component will be evaluated to determine its function and capabilities. Each subsystem must be capable of performing its assigned task effectively. In addition to evaluating the individual systems, the robot will be evaluated in its entirety. Doing this will allow the group to determine whether or not the robot will be able to perform well enough to not only meet the criteria of the competition, but subjugate the rest of the competition with despotic impunity.

9. Communication

Over the course of two semesters, the design group will communicate its progress by means of four formal presentations and five reports. The culmination of the group's effort will be displayed at the IEEE Region V Robotics Competition which will be held in San Antonio, Texas on April 8, 2006.

10. Mathematical Modeling

During system integration, calibration may be necessary. To compensate for errors, mathematical modeling may be useful. Mathematical algorithms may be necessary in path modeling and route optimization. This will enable the robot to carry out its mission quickly and efficiently. Dedicated software for each component makes use of mathematical equations that reduce sensor data into usable inputs to the robot's main program.

11. Chemical, Electrical and Mechanical Engineering Analogs

The line following system and the image recognition system both take advantage of digital PID feedback control that can be modeled using a typical spring mass damper analog.

12. Optimization

Optimization will occur both in the programming and final testing stages. Through testing, the programming code will be optimized to enable the robot to think efficiently and quickly. Also, once the robot is in its final stages, a great deal of optimization to the robot's software will occur based on results from testing on the mock course.

13. Ethics

This design project revolves around a competition. Social responsibility implies that the group ought to behave ethically and not attempt to sabotage others' projects or be involved with any similar deviant behaviors. The internet provides the group with many useful programming and implementation examples that must be referenced if used. Additionally, the group needs to investigate copyright and intellectual property limitations on the use of these resources.

14. Aesthetics

This robot does not have to be aesthetically pleasing; it just has to be functional. However, time permitting, the group would like to have the robot look its best on the competition day. In this situation, functionality takes priority to aesthetics. For instance, the robot must be assembled in an organized fashion in order to allow expedient access to components for maintenance purposes or optimizations. However, the addition of original stencils enhanced the personability of the Falcon.

15. Robust Design

The robot must not fall apart during the competition if it hits a small bump, as was an obvious oversight by several other design teams at the competition. The robot must also be able to correct itself if it deviates from the course. All the components must also endure the rigors of frequent testing procedures. The fidelity of the sensor inputs to the processor requires that these sensors must be aligned properly. Therefore, they must be attached such that they will not misalign at any point during the competition. The body must be able to endure any unexpected turbulence during either operation or transportation. It is also important that all electrical connections are as secure as possible while still modifiable up to