Georgia Institute of Technology

Robojackets

2006 IGVC Design Report

Table of Contents

1. Introduction and Team Organization

The primary goal of the Georgia Tech Robojackets' Intelligent Ground Vehicle Competition (IGVC) team for the 2005 – 2006 year was reorganization. Having lost upper classman to graduation, the team's focus shifted from implementing the most advanced technology to educating the new team members in fields required to excel at IGVC. Instead of a strict hierarchy of command, the Georgia Tech IGVC team experimented with allowing each member to work a broad range of tasks. Figure 1 below provides information about each team member.

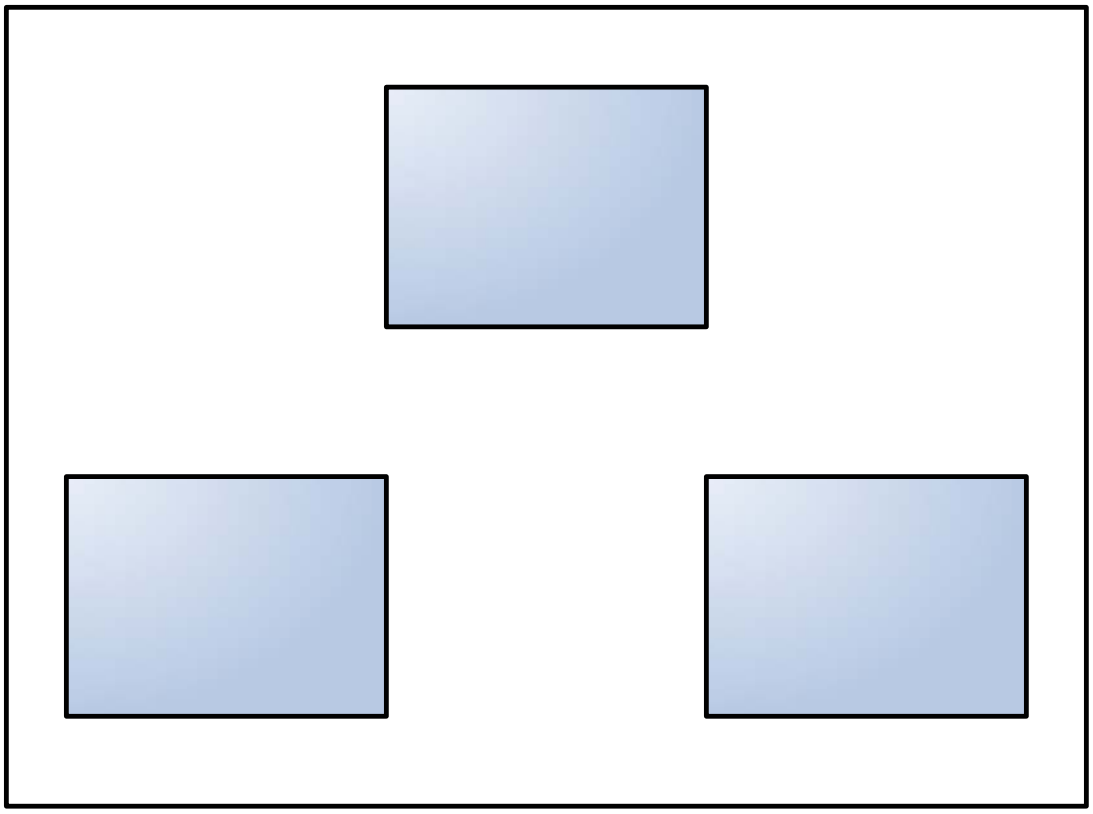
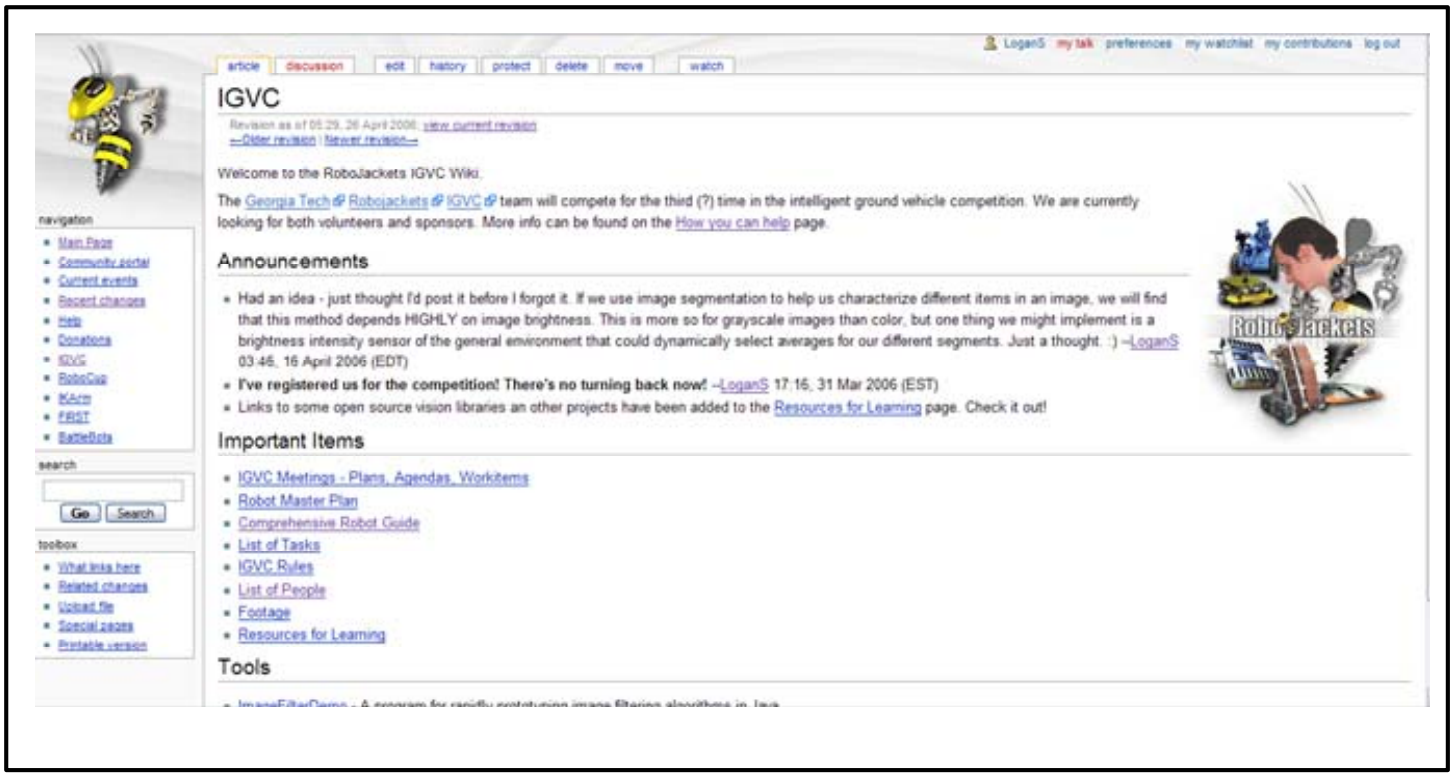


Figure 1 also demonstrates the involvement of each team member in a number of tasks. Each member spent, on average, 400 man hours planning, designing, and implementing the project.

In order to make team communication and knowledge distribution as available as possible, a wiki server was implemented using the Wikimedia Foundation Inc. software made famous by websites such as www.wikipedia.org. The wiki encouraged the team to link each other to information, create an online guide to all of the vehicle's designs, and keep good documentation of the project. Figure 2 shows the root webpage of the Robojackets IGVC wiki.



This method of documentation and communication was so successful that it was eventually adopted by the entirety of the Robojackets organization.

2. Design Specifications

2.1. Autonomous Challenge Description

For the autonomous robot challenge, an unmanned ground vehicle must navigate an outdoor obstacle course of white lines, orange barrels, sand traps, potholes, or simulations of

such objects. The course is approximately 600 – 800 feet in length with a minimum width of six feet between obstacles and white lines. Furthermore, white lines may be solid or dashed. A variety of weather and lighting conditions is expected. Terrain consists mostly of grass and pavement with inclines of no more than 15%.

2.2. Technical Limits

All control, sensing, and power systems must be local to the vehicle. No external control is allowed. Power systems may consist of electric batteries or internal combustion.

The vehicle's maximum height, length, and width may be six feet, nine feet, and five feet. Furthermore, the vehicle may not have a length less than three feet. Space for a 20 pound payload of dimensions 18" x 8" x 8" is also required; the payload must be forward-facing.

2.3. Safety Specifications

A hardware based emergency stop (E-Stop) button must be located on the center rear of the vehicle, be at least two feet but not more than four feet from the ground, colored red, and at least one inch in diameter. The E-Stop button must be hardware based. Furthermore, a wireless E-Stop should be implemented with a range of at least 50 feet.

Vehicle speed may not exceed five miles per hour. Also, if the E stop is triggered, the vehicle should not move more than six feet on a 15 feet incline before coming to a halt.

3. Vehicle Design

Design of the Georgia Tech IGVC vehicle was not consolidated into a period separated from actual implementation. At the onset of the competition design and build period, the new team members met weekly to collect and study the "legacy system" left from previous Georgia Tech IGVC attempts. Once a firm understanding of previous systems was attained, new systems were developed using a selection of old devices. Approximately as much time was spent

learning about items such as data protocols, software libraries, and computer aided design (CAD) as was spent actually designing new systems. However, this method yielded interesting, innovative results.

3.1. Mechanical

The guiding principal behind mechanical design was to simplify the drive base as much as possible. To accomplish this goal, the vehicle was designed to use a two motor differential drive system. Electric motors and their respective gearboxes were connected directly to two wheels mounted at the front of the vehicle's chassis. Two non-powered wheels (casters) were attached at the back of the vehicle to enable simple, easy turning of the vehicle.

To carry all electrical data and power modules, a 3' x 2' metal box was fabricated and bolted to the chassis. A simple box was chosen primarily for its ease of fabrication and space efficiency. Also, easy access of electrical components and the laptop were factors in this design.

3.2. Electrical

3.2.1. Data

The electrical system was designed to be modular, independent, and upgradeable. Therefore, rather than purchasing a standard hobbyist robot controller, a laptop was selected to act as the primary processor of the system. A generic laptop was selected with the following specifications of interest: a Pentium IV 2.4 GHz processor, standard serial and parallel ports, an IEEE 1394 port, and two USB 2.0 ports. Debian Linux kernel v.2.6.8.1 with a KDE v.3.5 desktop environment was installed to allow for easy manipulation of the laptop's I/O systems.

The laptop's plentiful I/O ports allowed for easy connection of all sensors. Off-the-shelf components were selected when possible for their reliability and time saving benefits. A consumer grade JVC digital camcorder with an IEEE 1394 port was selected for computer vision

sensing, as digital camcorders are preconfigured to work well in a variety of lighting and environmental conditions. A SiRF III GPS Evaluation Board with RS-232 was chosen with this same philosophy of quick, reliable implementation. These off-the-shelf modules were connected to the laptop's IEEE 1394 and serial ports, respectively.

Although off-the-shelf components were valued in this project, a great need was seen for giving some modules the ability to interface with each other independent of the laptop. Thus, several custom modules were designed that use the controller area network (CAN) protocol. The CAN protocol was selected because, unlike common protocols such as I2C, it does not use a master/slave system; instead, each module's message is prioritized based on which module is trying to communicate. A CAN interface module was designed to allow the laptop to communicate with the bus through the laptop's parallel port; in addition, interfacing modules were created for joysticks, motor drivers, and sonar (it was later decided that sonar would not be used at this time, however).

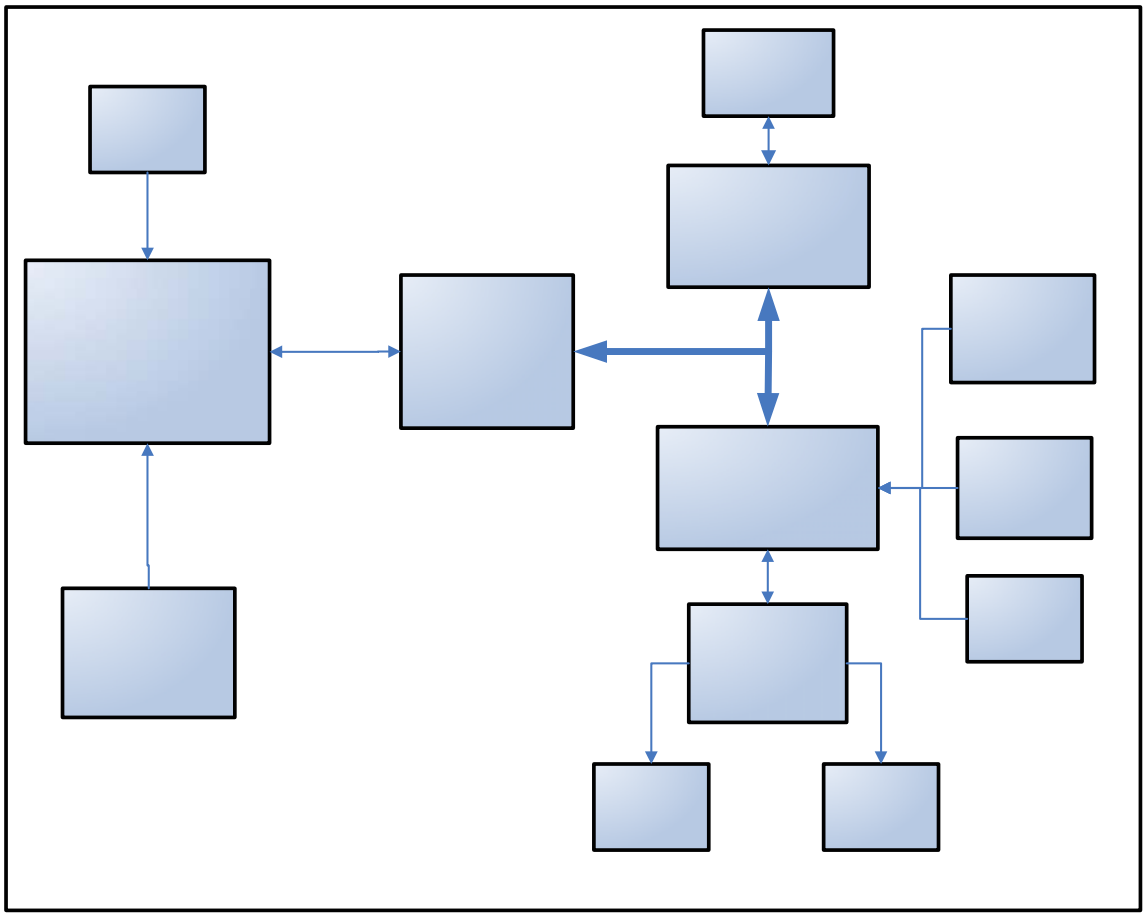
The design of the CAN interfacing modules centered on making unlike devices communicate seamlessly with any other device on the bus. Therefore, all interfacing modules were very similar; a Microchip CAN transceiver was placed at the front of the CAN bus on each module which connects to a Microchip CAN controller. The CAN controller was chosen because it abstracts nearly the entire CAN protocol, allowing designers with a minimal knowledge of CAN to use a serial peripheral interface (SPI) for bus communication. Connected to the SPI interface on the CAN controller IC was an AVR ATTINY8 MCU.

The onboard MCU was implemented to manage the specific I/O of each device. For instance, the joystick module's MCU was connected directly to the data, command, clock, and acknowledge bits of a standard Playstation 2 controller. Conversely, the motor driver module

was enabled to provide commands to motor drivers and take in data from a remote E-Stop switch, an autonomous/manual switch, and a reset switch.

All schematics and board layouts were generated by the CAD software Eagle. Once the boards were fabricated, they were built into separate boxes. Each box module interface was then connected to a different module and then to the CAN bus. These custom board designs were designed to require only small changes to allow a broad range of devices a standard means of communication.

In this year's design, for instance, the joystick controls the motors independently of the laptop. For an example of schematic and board layouts generated by Eagle, please see Appendix A. A graphical representation of all data module connections is depicted in Figure 3.



As shown by Figure 3, the laptop provides central processing for certain modules while others may operate more independently. However, future designs may enable GPS or the camcorder to communicate with the joystick or motor driver directly.

3.2.2. Power and Motor Drive

Two 12V sealed lead-acid batteries were placed in series to provide 24V. Power was then divided into sections, motor drive and data. One 24V Vantec motor driver was used to control both motors in the differential driving system. Power to the motor driver was switched by a contactor relay. The required "Big Red Button" E-Stop was positioned in the signal line of the contactor relay so that depressing the button cuts power to the motor driver. However, the remote E-Stop caused hardware in the motor driver's CAN interface to cease transmitting drive signals. Conversely, power to the data systems was forced to specific voltages by a custom designed power supply board. The board was designed to provide 11V for the JVC camcorder, 20V for the laptop, and 12V for other data modules.

To aid in visualizing the power connections, refer Appendix B for a block diagram of power distribution.

3.3. Algorithms

Computer vision was chosen as the primary means of sensing the environment. Several algorithms were developed to process the camera's raw line and barrel data into usable information; using this information, a navigation algorithm was developed to find a safe path through the image plane.

3.3.1. White and Orange Pixel Identification

Creation of a simple, relatively robust color identifier was the first stage of the vision design process. One of the most desirable traits of a color identifier was determined to be its

ability to accurately detect colors of objects of interest. The chief problem was creating a system with enough flexibility to allow for different lighting conditions while still detecting only the desired objects.

Orange pixel detection was simple to design. First, a red green blue (RGB) color space containing data of interest was loaded. Then, the green sample of the color space was subtracted from the red sample. The resulting data represented a measure of the orange intensity of any given subset (pixel) of the color space. Since IGVC obstacles were determined to be of a very high orange intensity, pixels were then compared against an experimentally-determined threshold. Pixels that had orange intensity values greater than the designated threshold were then passed to the barrel detection algorithm; pixels failing to meet this criterion were discarded.

White pixel detection required more processing to ensure accurate detection. White was determined to be too difficult to process by only thresholding high intensity values in the RGB color space. It was noted from experiment with the camera that many high intensity subsets of the image space appeared very close to white. In order to solve this problem, conversion to a hue saturation brightness (HSB) color space before processing was proposed. By converting the input image to HSB, intensity values of saturation (purity of the color) and brightness (intensity of the color) could be assessed. The measure of saturation was especially important, as a great deal of the uninteresting white data in images appeared somewhat grayed by mixing with other intense colors. Pixels values with saturation less than a certain threshold and brightness above a certain threshold were then considered line or pothole objects.

3.3.2. Barrel Detection

Once pixel data was processed, the next step was to detect barrels in the image. The designed algorithm iterated through each row of the image space starting in the bottom left

corner. If a certain amount of orange pixels was detected in a row (this value depended on how far the row was from the top of the image), then the first pixel in that row that registered orange was considered the bottom left of a barrel. Next, the horizontal interval of the orange pixel array was recorded. The algorithm then proceeded to scan the rows above the bottom until a specified percentage of white and orange striped had been found. Barrel data was then passed to the navigation code for final processing.

Figure 4 demonstrates the effectiveness of the barrel finding algorithm.

The yellow lines pictured in Figure 4 represent the outer bounds of the barrel data.

3.3.3. Danger Map Navigation

With a satisfactory amount of environmental data collected, the final step towards designing the vision system was to develop a navigational algorithm. In designing such an algorithm, three items were pursued: a danger map that could quantify "danger" along a given path, a system that could be adjusted quickly by changing a few parameters, and the ability to convert paths easily into drive commands.

In order to satisfy the first design item, a path system was developed. Many paths were generated at different angle intervals from each other; each path traverses image space from its origin in the bottom center of the image. Each path crossed different numbers of white pixels and barrel objects. Every time each path crossed an object of interest, a counter was incremented. At the algorithm's conclusion, the path with the fewest number of danger crossings was selected to be the vehicle's path.

This algorithm satisfied the second design specification as well. The system required no hard coding; conversely, all the system required as input was a starting angle, a stopping angle, and the number of paths for the system to use. This encapsulation allowed for easy experimentation with the best number of parameters rather than redesigning the algorithm to fit experimental data.

The third design specification was also satisfied. Each path could be characterized by two parameters – the danger value and the path's angle. By outputting the angle of the path with the lowest danger value, the system could read the angle of the path and determine proper left and right motor speeds to cause the vehicle to turn to that angle. The entire algorithm was constantly run to ensure that the vehicle constantly favored an open path.

Figure 5 shows a test of the navigation path algorithm.

Figure 5. Preliminary tests of the navigation path algorithm.

Note that a darker color indicates a higher value of danger.

3.4. Software

Software was designed in three categories: hardware, control, and graphical user interface (GUI). With the exception of firmware written for the custom CAN interface modules, all code was written for the laptop to interface with, command, and display data from the sensors. C++ was the choice language for programming the laptop; C was chosen to create module firmware.

First, hardware code was written to prepare the laptop's I/O for the purpose of acting as a robot controller. By using Debian Linux, the only software design required for general I/O was to create code that input and interpreted data directly from Linux device libraries. Specifically for digital video, the Linux digital video library Kino was used to simplify and optimize intake

from the IEEE 1394 port. Software was also created to interface with the CAN bus via an SPI connection through the laptop's parallel port.

Once I/O was established on the laptop, a control thread was created. The control thread's purpose was to process all data from sensors, implement the algorithms discussed in section 3.3., and to send drive commands to the CAN bus. Furthermore, basic GPS interpretation code was written into the control thread for future use.

To encapsulate all of the processes on the laptop, a GUI was written using the QT C++ graphics library. The interface was designed to allow vehicle operators a rapid method for getting updates about the vehicle's status, changing parameters between course attempts, and running diagnostics. Figure 6 demonstrates the "main" screen of the GUI.

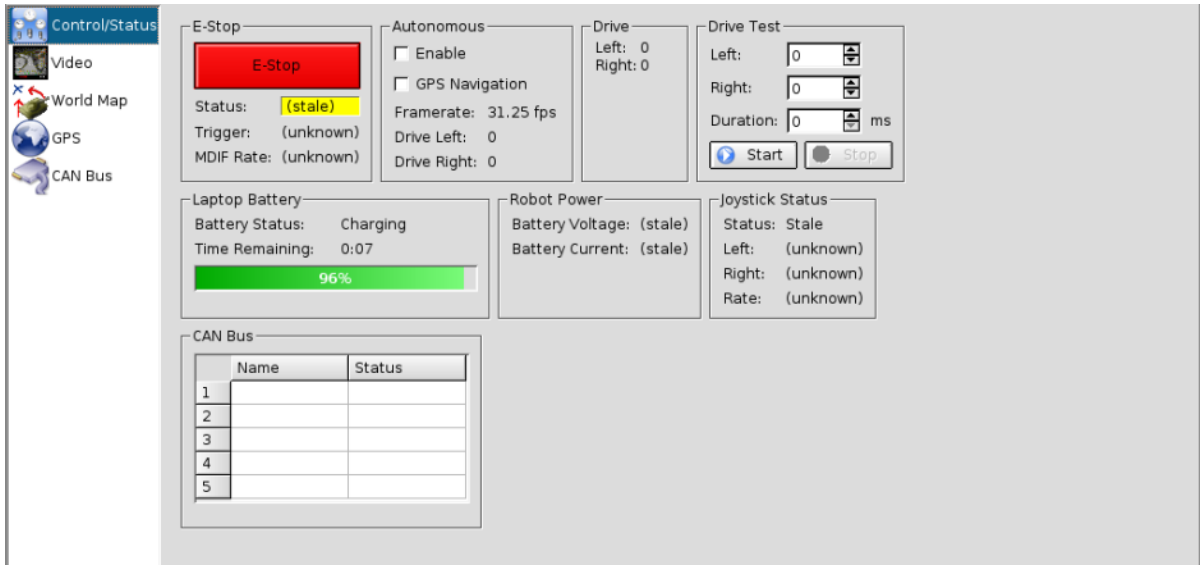


Figure 6. The default page of the vehicle control GUI.

4. Predicted Performance

The vehicle's performance is still under heavy evaluation at the writing of this paper. However, predictions on performance can be made. The vehicle's top speed approaches five mph. Ramp climbing ability is very possible mechanically; also, the navigation algorithm performs well with the solid white lines of the ramps. System reaction times can be adjusted precisely by changing the sampling rate of the navigation algorithms. Battery life, with motors constantly running under the load of the vehicle, lasts approximately six hours.

At competition, the vehicle is expected to navigate the course if the bounding lines remain visible. Based on camera tilt, the vehicle can see objects of interest at a distance of approximately four meters. The vehicle should be able to avoid traps and dead ends, as its navigation algorithm can search "deep" into the image plane to find truly clear paths. Objects such as potholes are treated as lines and should be avoided. GPS navigation most likely will not be ready for competition, however.

A few trial runs have shown that the vehicle is capable of finding its way between and around barrels. It relies heavily on the bounding lines to keep it on track, however. Preliminary results have been encouraging, but further testing is required for certainty.

5. Cost of Project

Table 1. Estimate of project costs

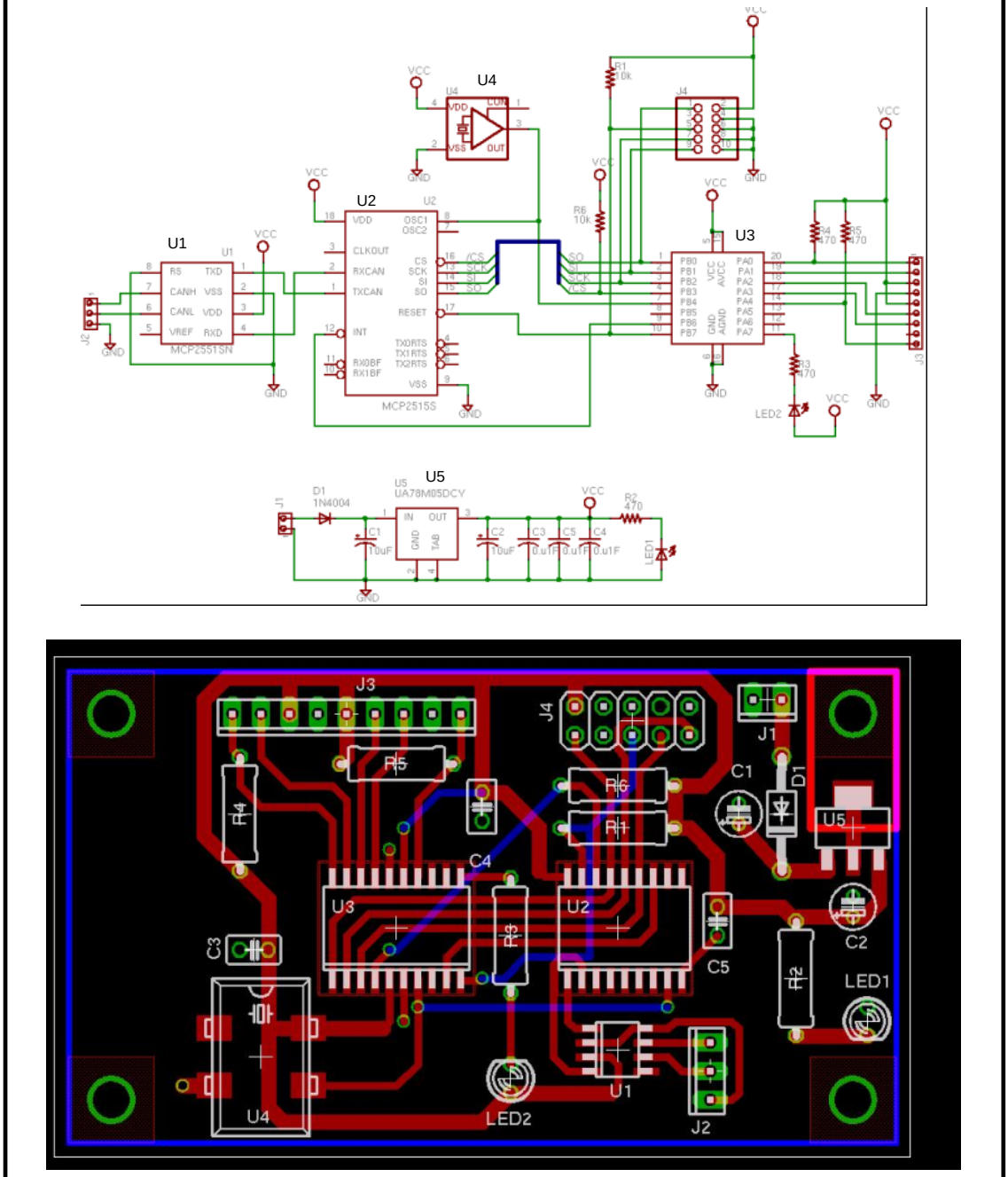
Item	Cost
Camera	$450.00
GPS	$120.00
Chassis	$600.00
Batteries	$600.00
Motor Drivers	$200.00
CAN Modules	$300.00
Laptop	$1,000.00
Total	$3,270.00

COURSE EQUIVALENCY

I, _____, certify that the engineering design in the vehicle (original or changes) by the current student team has been significant and equivalent to what might be awarded credit in a senior design course.

Professor:_____ Date:_____

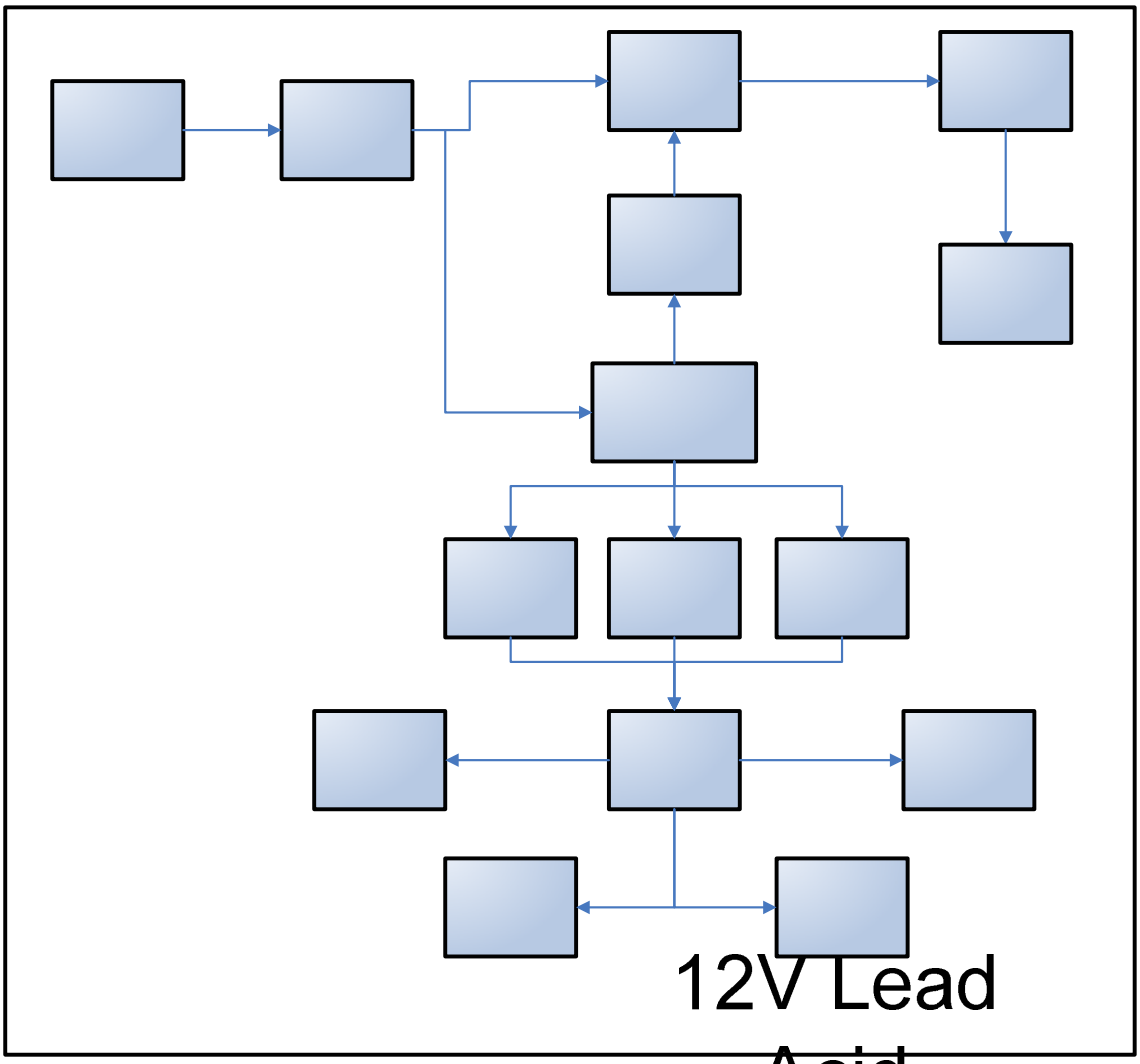
Appendix A: Example CAN Schematic and Board Layout



Legend
U1. CAN Transceiver
U2. CAN Controller
U3. Microcontroller
U4. Crystal Oscillator
U5. 5V Regulator IC

Figure A1. Example EAGLE schematic and board layout using the joystick CAN interface.

Appendix B: Power Distribution Block Diagram



12V Lead Acid Battery

1