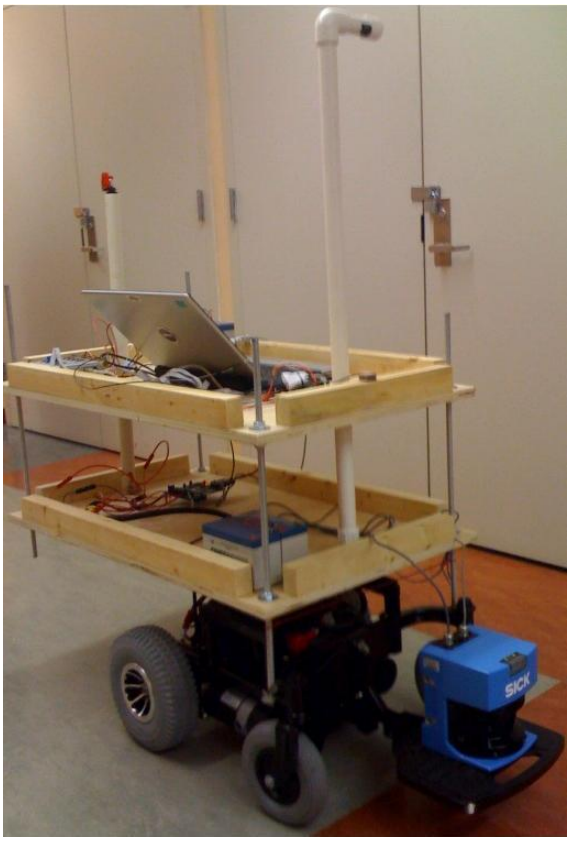
US Naval Academy Robo-Goat

2009 IGVC Design Report



Project Advisors:

Associate Professor Joel M. Esposito

Associate Professor Brad Bishop

Team Members:

MIDN J. D. Lovett, MIDN T. M. Fleming, MIDN J. P. Smith

MIDN R. P. Chandler, MIDN J. M. Lacy, MIDN M. A. Calvanico

I certify that the report and vehicle built by this team for the US Naval Academy's entry into the Intelligent Ground Vehicle Competition was all done by students; that the work was substantial (100% of design and construction done during 2008-2009 academic year); and that the work was awarded credit for a capstone senior design course.

Joel Esposito _____ Assoc. Prof. Systems Engineering

Background Information

The 2009 competition will be the first year a Naval Academy team has competed in the Intelligent Ground Vehicles Competition, so there is no local precedent set. Despite USNA having not previously competed, online design reports and schematics at the IGVC website from past competitions acted as starting blocks for our own vehicle. These reports set the precedents needed to establish a proper timetable, documentation, and testing cycle. Previous Capstone projects involving ground vehicles as well as our many system courses aided us in our design process and our implementation of the necessary sensors involved.

Objectives

The robotic vehicle must be safe, cost-effective and be able to properly perform the obstacle course. In order to be safe, the vehicle will run at limited speeds and accurately avoid obstacles. To maximize cost-effectiveness, a simple design will be implemented with low maintenance, low power consumption and high endurance. To maximize performance, the vehicle again must accurately avoid obstacles, utilize all sensors, be able to perform multiple trial runs, react quickly, be durable and rugged, easy to operate and quick to start.

The objectives for our project, listed above, came from both the competition website and from what we determined, as a group, were essential to building a successful autonomous vehicle. The following diagram shows the priority of objectives in our design process. Once the weighted objectives were determined, we came up with functions that were the driving force of our project.

OBJECTIVE	WEIGHT
Limited Speed	20
Accurate (safe)	19
Able to avoid Obstacles	18
Quick Start	15
Accurate (sensors)	15
Repeatability	13
Durable	12
Rugged	12
Quick Speed	11
Endurance	10
Simple Design	9
Easy to Operate	7
Low Maintenance	5
Low Power	3

Weights were picked based on a 1-20 scale with 20 being the highest weight and 1 being the lowest. Each objective was assigned a weight based on how important we felt it was to the overall project design and performance.

Final Design Narrative

Drive Train and Platform Design

Chassis: We decided to invest our time and money into a powered wheelchair in order to simplify the robot's drive train. With the removal of the chair, joystick control board, and the plastic frame, a powered wheelchair was nothing more than a square metal frame housing two independent brushless DC motors and a battery bank. We selected the Drive Sunfire Plus Rear Wheel Drive, which proved to be the perfect foundation for our robot. In regards to the following decision matrix, this particular wheelchair was chosen because the two motors allowed for zero turn radius and differential drive, while reaching 5 mph. Also, the 2' x 3' frame fell well within the IGVC competition rules.

Platform	Manufacturer	Price ($)	Turn Radius	Motors	Max Speed	Size
Sunfire Plus	**Drive**	**1530**	**0"**	**Dual DC**	**6 mph**	**24" x 35"**
Sun Runner 3	Shop Rider	1480	37"	Trans-Axle	5 mph	24" x 38"
Compass	Golden Tech	3699	19"	Dual DC	5 mph	24.5" x 38"

Platform: A platform needed to be mounted on the wheelchair in order to house the various sensor equipment, batteries, and the laptop computer. We decided a two tiered design would provide enough stability to support the equipment, while still maximizing the dimensions set by the competition constraints. The dual tiers separated the batteries and laptop from the sensors to ensure the components were not thrown off due to electromagnetic disturbances. The original design was deemed overly complicated by the woodshop carpenters, because of the hole locations and the interlocking beams. We modified the design and produced the current two tiered wooden platform currently attached to the chassis. The plywood and metal rods allowed for easy modifications to the platform's shape and height. It also made assembly and disassembly an easy process.

Driver Controller: Since the chassis had two independent DC motors, a dual H-bridge motor controller was needed to link the laptop to the motors. After weeks of fruitless manipulations to the existing joystick controller, we chose the commercial, off the shelf, Roboteq AX-1500. This particular board allowed for dual motor control, serial communications, a 30 ampere threshold, and multiple motor inputs. The control board accepted multiple inputs from the computer to drive the wheels; one input controlled the forward and reverse (translational velocity), while the other controlled the left and right motions (angular velocity). The 'driveRoboteq' function established two variables [V, W] that were called upon by other function within the master code. The translational velocity was represented by the

symbol [V] and ranged between the values [-1, 1]. The value [-1] meant the wheelchair moved in full reverse and the value [1] meant the wheelchair move in full forward. The angular velocity was represented by symbol [W] and ranged between values [-1, 1] as well. The value [-1] meant the wheelchair turned full left and the value [1] meant the wheelchair turned full right. Using a tachometer, we tested the rotations per minute and found the appropriate speed setting [3F]. Within the function, a section of code limited the speed of the wheelchair to less than 5 mph. After the joystick controller was removed, which acted like a speed governor, it was necessary to limit the speed. The Roboteq board worked using a hexadecimal value as an input. The values of hexadecimal ranged from [00] to [7F], but we limited the maximum value to a little above 50% power or [3F]. Within the laptop's master code, the 'driveRoboteq' function converted the numerical values of [V] and [W], given by the sensors, into hexadecimal by multiplying the decimal inputs by [63] and then sent them to the AX-1500, which powered the motors accordingly.

The 'initRoboteq' function established a serial port and bit rate between the laptop and the AX-1500 allowing information to pass freely. The communication port used in our system was port 6 and the required specifications were every seven data bits. The high signal was on the even parity and the terminator was set to the center. Once this was complete, the board was opened and operational.

Internal Brake System: The chassis that was chosen came with internal, electronic regenerative brakes. Prior to use, the brakes needed to be disengaged. Original this was done with the built in joystick controller, but upon removal, the brakes needed to be disengaged directly. After numerous setbacks, we realized that a direct 24 V needed to be applied directly from the batteries. There is an attached brake release, requiring two 24V inputs and one ground line. After the brakes were released, the user could control the wheels more freely without any back EMF from the AX 1500.

Manual Emergency Stop

The IGVC mandated the emergency stop be a hardwired device, so we placed a switch between the motor controller and the battery banks. This severed the power to the controller, which forced the motors to stop. The switch was mounted on PVC to reach the 2' minimum height.

Laser

Our group went through many options on the path to arriving at our final design. One of the first problems we encountered was actually figuring out what laser we wanted to use. Three different models were researched: one from Acuity Lasers, a Hokuyo and the SICK. Our decision matrix for the three different lasers is shown below:

Laser	Manufacturer	Price	Sweep	Range	Power	Weight
AR4000	Acuity Lasers	$7,500	300°	54 ft	5 V	22 oz
UTM30LX	Hokuyo	$5,590	270°	30 m	12 V	13 oz
LMS 200	**SICK**	**$5,571**	**180°**	**80 m**	**24 V**	**4.5 kg**

We decided to use the SICK LMS 200 for two main reasons. First, the Weapons and Systems Department had it in stock, which allowed our group to save the cost of the laser. We were given an internal grant to help offset the cost of parts, but the cost of the laser would have used up over half of the funds allotted. The second reason we chose the LMS 200 over the other models was that it had the greatest range. Using the following calculation, $\left[\frac{5mi}{hr} * \frac{5280\,ft}{mi} * \frac{1hr}{3600\,\sec}\right]$, we determined that the fastest our vehicle would be traveling was 7.33 ft/s. The eighty meter range of the SICK allowed us to have plenty of look-ahead distance, which would let us find obstacles very far in advance and start avoiding them much sooner. Based on these two parameters, the LMS 200 was the best choice for our group. We did not have to wait to order the laser or to process and then ship it. This proved very helpful because we encountered some problems getting our program to receive all the data from the scans.

The laser sent its data over a serial cable, which inherently means there will be problems when trying to read this data in MATLAB. When we got the laser we were also given some previous code that read the data and produced a plot showing the scan. However, the code did not work when we ran it with our laser. We found that our buffer size was incorrect for the program. The LMS was sending more data than the program allowed for. We spent nearly three weeks trying to solve the problem before simply rewriting the code. We developed a new MATLAB script that we hoped would solve the buffer problem. Fortunately we were able to get all the data in one scan and plot the scan in a polar plot. Once we got communication between the laser and the computer, we were able to manipulate the plots and further develop the code to find obstacles in front of the laser.

We decided to mount the SICK laser onto the vehicle's footrest. Having the laser mounted on the footrest provided two distinct advantages over other mounting possibilities. The first being the fact that the laser was low to the ground and able to pick up obstacles that may have been missed if it was mounted higher on the platform. Since the obstacles at the competition can vary in size, we felt that mounting the laser low on the vehicle gave us a good look ahead and ensured that we were able to detect any obstacle on the course. The other advantage was it would remain level with the ground the vehicle was traveling. If the wheelchair traveled up an incline, the laser might have picked up the ground in front of it as an

obstacle. Mounting the laser level with the chassis solved this problem, and we yet to detect the ground as an obstacle. Mounting the laser on the foot rest was a very simple task. We drilled two holes through the rubber and aluminum and inserted two bolts into the mounting holes on the bottom of the laser. It was mounted slightly off center and to the rear of the footrest in order to avoid any collision with the laser face.

GPS/Compass

The process of selecting our GPS unit was very straight forward. We initially conducted research on handheld GPS units that provided the user with a graphical user interface. Since most of the commercial, on the self, (COTS) GPS units were fairly expensive ($100+) and we would have to find a way to hack in to them, we decided to check the in-house GPS units. After speaking with the Naval Academy's in-house developed Rabbit 3000 based Navigation/Control System. We chose this unit because it incorporated an accurate GPS (<3 meters) with an average refresh rate (0.5 Hz), a three-axis magnetometer, accelerometers, gyros, a Rabbit 3000 processor, and had a skeleton code that was used in several courses we took.

To use the GPS, we connected the unit to the laptop via serial connection and the programming port. Since the unit did not have a direct serial connection, we used an onboard TTL port with a TTL to RS-232 (serial) cable to send data to our laptop and we used a standard serial programming cable to program the Rabbit processor via the programming port and Dynamic C coding. The data that we requested from the unit came from the onboard Rabbit 3000 processor. In the present state, we only used the latitude information, longitude information, and wireless emergency stop data, but we also export the magnetometer data, number of transmissions, header, and footer.

The actual calculation data occurred in the MATLAB workspace. When the GPS, and e-stop, data was sent out of the Rabbit processor via the TTL to RS-232 cable, MATLAB read the serial port on the laptop the data was coming through and saved that as a string. Using the string read function in MATLAB, we were able to break apart the message and save the data as a variable representative of what it is, i.e. the latitude data was saved to the variable “lat”. With this data, and when desired waypoints have been loaded, it was possible to determine a desired heading for the vehicle to drive.

Wireless Emergency Stop

Our initial idea was to use a garage door remote as a transmitter, but this still required a mechanical receiver. We decided upon a Futaba remote control and receiver. These systems are commonly seen on remote controlled vehicles built by students. These parts were readily available and

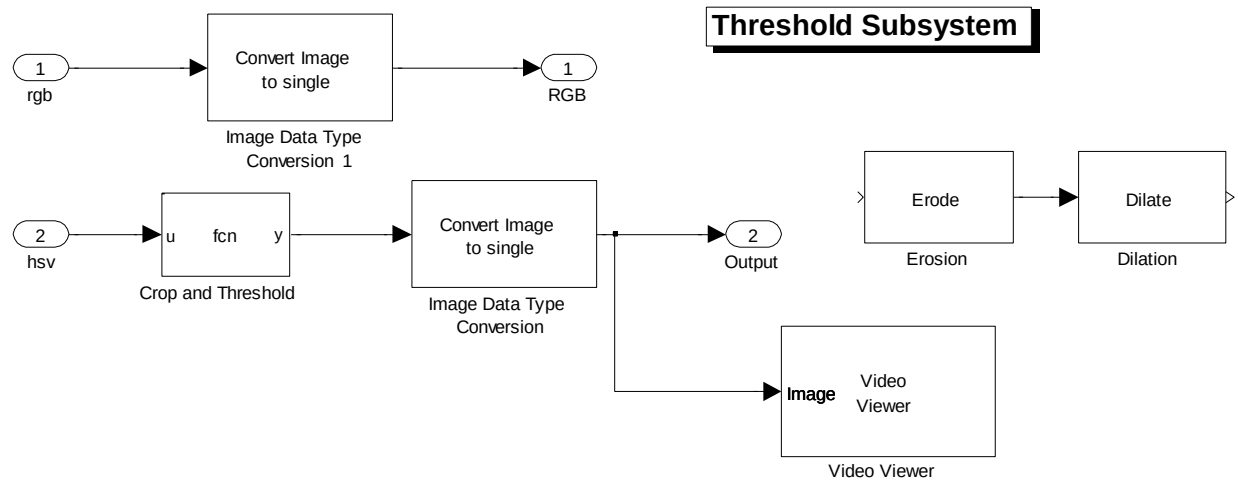
easily replaceable if they break. The Futaba receiver held 8 ports which could receive data from the remote control.

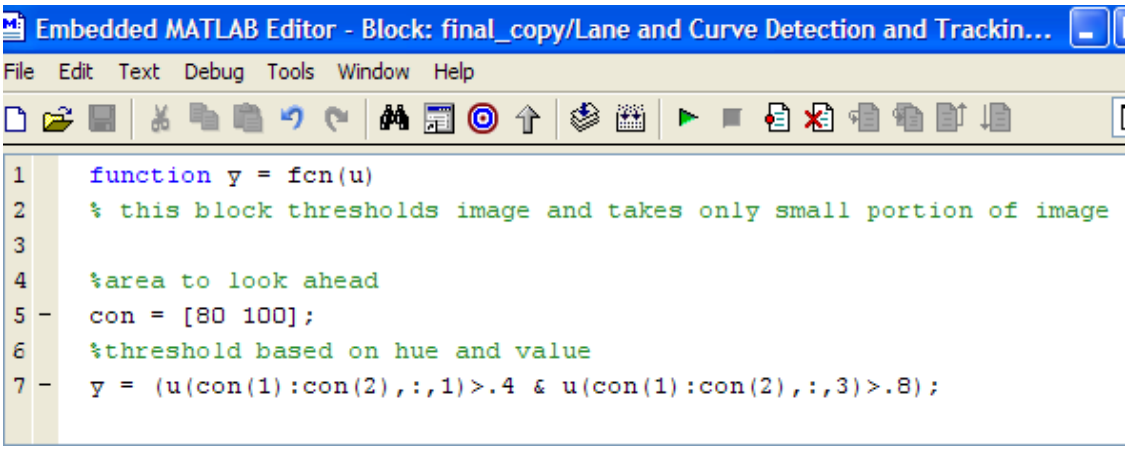
Camera

The overall goal of the vision system was to look ahead at the course and determine where to steer the vehicle. To do this, the vision system had to discern the white lanes from the grass field. Once the lanes were found, the vision system had to find the center of the lane and determine if the robot was to the left or to the right of the lane ahead so it could steer accordingly.

First to achieve the goal of the vision system, we had to choose a camera to use. We kept in mind the specific metrics, including specifically a simple design and accuracy of the sensors. We decided to use in house parts to achieve the simple design metric. We were given two options: a D Link camera and a Creative Live Notebook Ultra Camera. We had some experience with the D Link camera in previous systems courses and we knew it was not as accurate as we would like. The Creative Live Notebook Ultra boasted a 1.3 megapixel resolution and an ultra wide viewing angle of 85 degrees, which was be ideal for seeing both of the lanes. We decided to use the Creative camera and were very pleased with the results. It was accurate, performed the functions we needed and even came with software to adjust brightness and other relevant camera settings. Additionally, this camera used a USB interface to communicate with the laptop. The cost of the camera is $70.00 from newegg.com. Once the camera was selected we looked at ways of mounting the camera to the frame of the vehicle. We decided to go with PVC pipe for several reasons. The PVC pipe was solid and durable. It was also moveable and not glued, which allows us to adjust the angle of the camera whenever necessary. One disadvantage to our setup was the shakiness whenever the vehicle moved. This however was not necessarily attributed to the PVC pipe, but more to the size of the hole, the PVC pipe went through, being too big. This hole can be filled to reduce the shakiness of the structure.

With all the hardware set up, we needed to come with accurate code to take the hardware and come up with relevant data to steer the vehicle. To do this we used MatLab Simulink to program our vision system. The following is pseudocode for how the vision system works *(1) Take Image; (2) Threshold and Crop; (3) Morphology; (4) Detect Lane; (*5*) Find Center of Lane (6) Output Where to Aim.* The main work of this system was done in the Threshold Subsystem, the Lane Detection Subsystem, and the Find Lane Center (which was embedded MatLab code).





This code looks only at rows 80-100 of the input image (which is used to look ahead in the course a specific distance). Then it thresholds based on, in this case, the hue and value numbers. This was embedded in code, because it was more versatile and easier to change the numbers for thresholding. It was also easier to switch thresholding by hue, saturation, and value. To threshold by these numbers, we had to take a picture and convert it to HSV. Once in HSV, we looked at the individual Hue, Saturation, and Value images to decide which to threshold by. We used the MatLab 'imtool' function to get specific values for thresholding. We found that on different days we might need to threshold by hue, whereas others we had to threshold by saturation, but as long as the lighting stayed somewhat constant over the day it would threshold very effectively. Also, we originally had morphology in our simulink diagram to reduce noise from bad thresholding. However, in the final model we took out the morphology, because we were able to threshold effectively.

For the lane detection subsystem, we took several different approaches. At first, we had bad videos that were unable to threshold very effectively, so we decided to try blob analysis and get the centroids of the blobs, which we assumed to be the lanes. There was a problem with this in that if the lanes were broken and not full blobs we were unable to get the lane data consistently. After this we moved to a new approach called pixel searching. We wrote embedded MatLab code that looked at specific rows and searched from left to right for the lane pixels in the image. The problem with this approach was that if there was any noise whatsoever and the program runs into it, the program assumes the noise pixel was the lane and ruins the whole approximation of the lane. This was unacceptable, so we finally moved to using Hough transforms. Hough transforms worked by finding where large pixel collections laid in the image to correlate to a straight line. We originally thought to use this approach, but decided against it because there was too much noise and we thought it did not work well. Once we got good thresholding images, we tried the Hough transform and it worked very well.

The final component of the vision system was the lane center determination. We needed to find the center of the lanes after finding them through the Hough transform. To do this we wrote embedded MatLab code to calculate the center of the lane and tell the vehicle where to aim for. The main problem this program had to address was how many lanes the lane detection system found. We knew there were possible cases where only one lane was found or even no lanes were found. This program addressed these cases in the following manner. If one lane was found and the midpoint of the lane was on the left half of the screen, then we assumed the left lane. If the midpoint was in the right side then we assumed the right lane. We will have to address the possibility of no lanes in the future, because as of now the program will just aim straight, when in reality, we will probably need to make the robot rotate 360 degrees. Additionally, since we have no video data from actual competitions, we think that our single lane decision making process might need to be changed. Until we actually run through the competition course, we will not know how to tweak the system, but we should be able to make a more effective decision when we have the data.

Subsystem Interfaces

Wheelchair

- Battery Bank: two 12 volt batteries supply 24 volts in series.
- Internal motor brakes require a continuous 24 volts to disengage.
- Each motor requires 9 ampere.

AX-1500 Controller

- 30 ampere maximum limit.
- Connected RS-232 serial connector from control board to laptop.

- The 24 volts from the battery bank are applied directly to the AX-1500 to power the board and the motors.
- The control board must be initialized in MATLAB from an open serial port. This is done using 'initRoboteq'.

Manual Emergency Stop

- The ground pin needed to be removed from the switch, in order to prevent the AX-1500 from sending a back EMF through the wires and blowing a fuse.

Laser

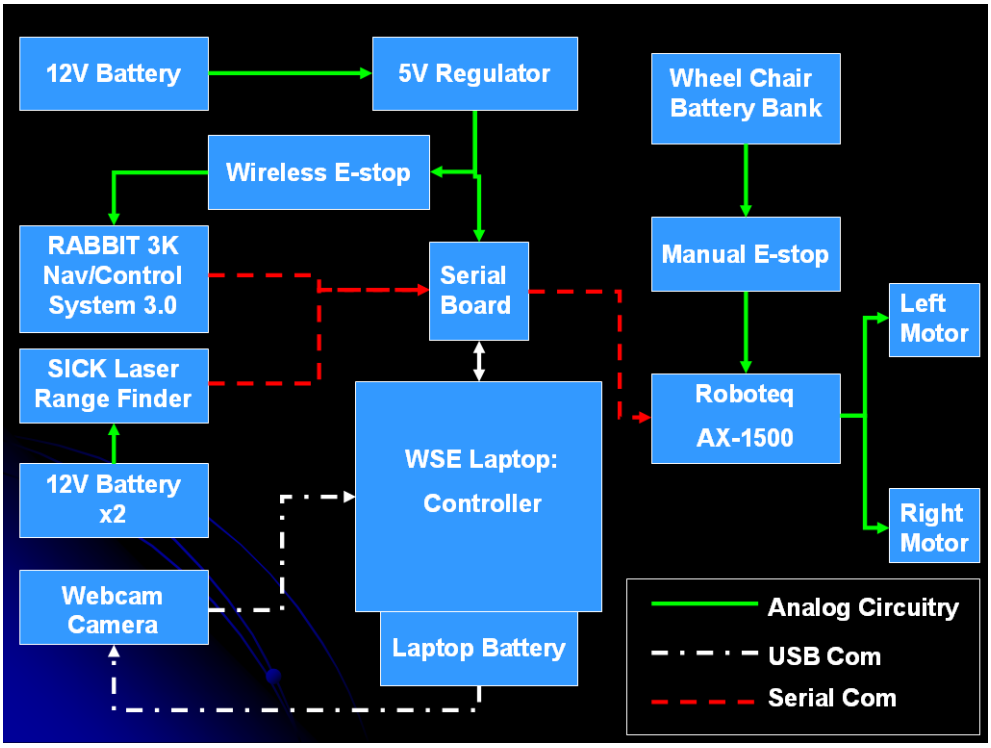
- Requires customized RS232 serial cable.
- Requires customized power cable and complimentary (±12) volt power supply.

GPS/Compass

- The GPS unit was originally powered by a wall outlet, which was stepped down to a regulated 3.3V, but to make the component mobile, we used a regulated 5V from a 12V battery to power the board.

Wireless Emergency Stop

- This system worked by sending a "high" voltage (binary 1) or "low" voltage (binary 0) to the Rabbit processor via pin PD7.



Design Evaluation

Drive Train and Platform Design

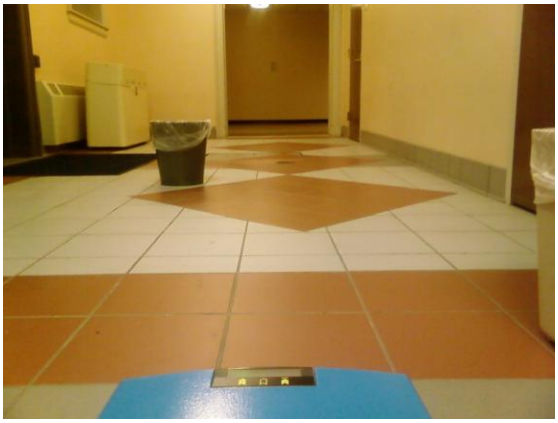
Objective	1	2	3	4	5
Quick Speed	< 1 mph	1.5 mph	2.5 mph	3 mph	**4.5 mph**
Easy to Operate	> 10 min setup	7 – 10 min setup	5 – 7 min setup	**3 – 5 min setup**	< 3 min setup
Simple Design	complete custom build	several custom parts	**extensive external sources**	little help from outside	built with COTS parts
Low Maintenance	1 hour in shop for 1 hour run	1 hour in shop for 2 hour run	**1 hour in shop for 5 hour run**	1 hour in shop for 7 hour run	1 hour in shop for 10 hour run
Quick Start	> 60sec start time	45 – 60sec start time	30 – 45sec start time	**26 – 30sec start time**	15 – 25sec start time
Durable	complete destruction	major repairs	minor repairs	**minor damage, no work**	no structural damage
Rugged	flat terrain only	slight grades	hilly	**small rocks**	all terrain

We tested the motor controller extensively to ensure each of the two motor functions were capable of meeting the team's overarching functions. For the most part, the motor functions worked well, but occasionally received errors from the various subsystems due to the clear delay between iterations of [V] and [W]. Our vehicle design met all the constraints and standards set forth by the IGVC, but there were a few technical and functional errors. The platform was 1" too narrow, hence, not meeting the 2' width minimum. This will need to be address in regards to future modifications to the overall design. Greater gauge of wire is recommended as well as more fuses to prevent mishaps.

Objective	1	2	3	4	5
Obstacle	No avoid	Avoid 25%	Avoid 50%	Avoid 75%	**Avoid 100%**

Avoidance					
Accurate (Sensors)	Unreliable readings	Can detect 25% of objects	Can detect 50% of objects	Can detect 75% of objects	**Can detect 100% of objects**
Durable in collision	Complete destruction	Major damage	Structural damage	Minor cosmetic damage	**No damage**

Laser The final design performed well. As stated earlier, our final design was very simple. It was able to detect and avoid 100% of the obstacles. If the laser detected an obstacle, it would find a way around the obstacle, and if it could not, then it would stop the vehicle. The way we demonstrated the use of the laser was by setting up a test course and having the vehicle maneuver through the course. The left image is the course from the laser's point of view, while the right image is an overall perspective.



The vehicle was able to move in the hall without hitting the obstacles. Overall we were satisfied with this design aspect. If given more time, a more robust code could be written. Now, with the current code, there is oscillation in the system because there is a delay from when the laser scans to when it sends a command to the motors, to when it takes another scan.

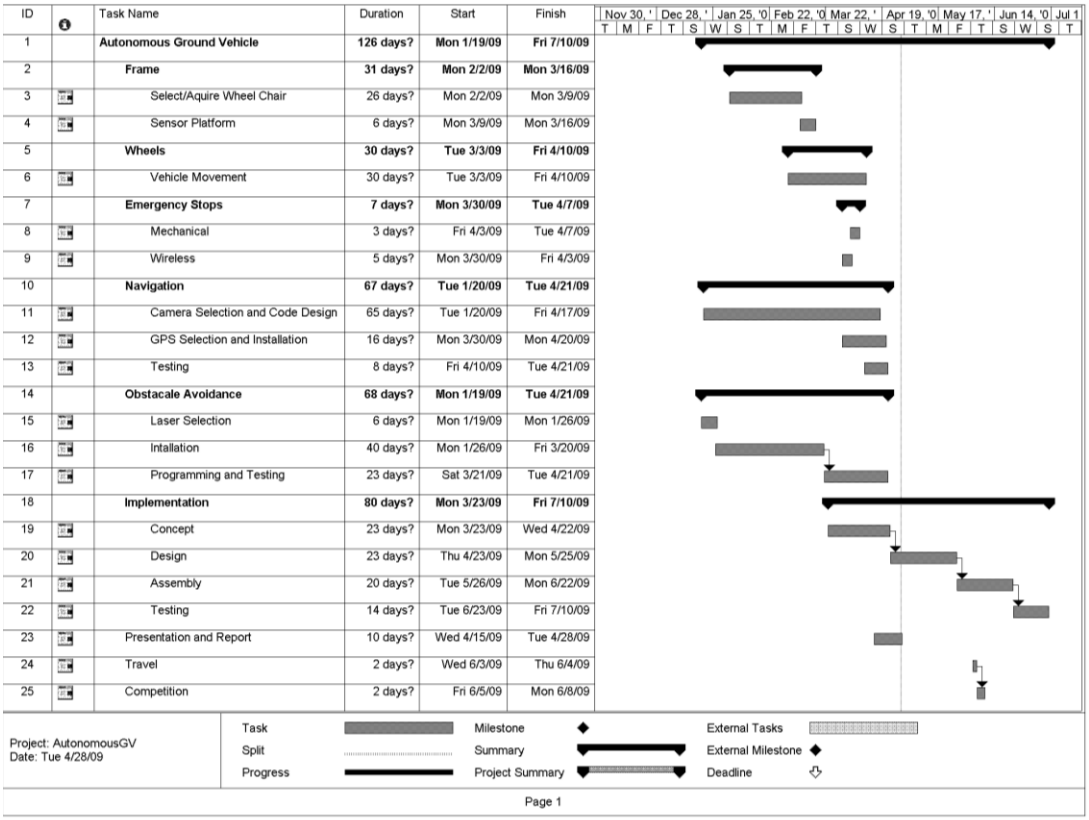
GPS/Compass

To test the functionality of this system, the GPS unit should be taken out to a large field with a secondary, hand-help GPS system. Fixes should be taken with both systems and compared. If they match, the computational algorithm and motor control should be tested by entering desired waypoints and actually setting the vehicle in motion to see if it reaches those points.

While developing this system, the greatest difficulty was getting accurate, real-time data from the GPS unit. As has been determined, the "buffer" size setting in MATLAB to read the serial port was initially set to read 512 characters but our message was only approximately 57 characters. Since the

buffer was set too large, multiple GPS messages would fill the buffer and after the first message was read, old data would still be sitting in the buffer. To solve this problem, we redefined the buffer size in MATLAB to fit exactly one message. Another method to solve this problem would be to "empty" the buffer after each reading. This would alleviate the problem of multiple messages being in line since only one message would be read before the buffer were emptied. For the time being, our solution works well.

Gantt Chart



ID	Task Name	Duration	Start	Finish
1	**Autonomous Ground Vehicle**	**126 days?**	**Mon 1/19/09**	**Fri 7/10/09**
2	**Frame**	**31 days?**	**Mon 2/2/09**	**Mon 3/16/09**
3	Select/Aquire Wheel Chair	26 days?	Mon 2/2/09	Mon 3/9/09
4	Sensor Platform	6 days?	Mon 3/9/09	Mon 3/16/09
5	**Wheels**	**30 days?**	**Tue 3/3/09**	**Fri 4/10/09**
6	Vehicle Movement	30 days?	Tue 3/3/09	Fri 4/10/09
7	**Emergency Stops**	**7 days?**	**Mon 3/30/09**	**Tue 4/7/09**
8	Mechanical	3 days?	Fri 4/3/09	Tue 4/7/09
9	Wireless	5 days?	Mon 3/30/09	Fri 4/3/09
10	**Navigation**	**67 days?**	**Tue 1/20/09**	**Tue 4/21/09**
11	Camera Selection and Code Design	65 days?	Tue 1/20/09	Fri 4/17/09
12	GPS Selection and Installation	16 days?	Mon 3/30/09	Mon 4/20/09
13	Testing	8 days?	Fri 4/10/09	Tue 4/21/09
14	**Obstacale Avoidance**	**68 days?**	**Mon 1/19/09**	**Tue 4/21/09**
15	Laser Selection	6 days?	Mon 1/19/09	Mon 1/26/09
16	Intallation	40 days?	Mon 1/26/09	Fri 3/20/09
17	Programming and Testing	23 days?	Sat 3/21/09	Tue 4/21/09
18	**Implementation**	**80 days?**	**Mon 3/23/09**	**Fri 7/10/09**
19	Concept	23 days?	Mon 3/23/09	Wed 4/22/09
20	Design	23 days?	Thu 4/23/09	Mon 5/25/09
21	Assembly	20 days?	Tue 5/26/09	Mon 6/22/09
22	Testing	14 days?	Tue 6/23/09	Fri 7/10/09
23	Presentation and Report	10 days?	Wed 4/15/09	Tue 4/28/09
24	Travel	2 days?	Wed 6/3/09	Thu 6/4/09
25	Competition	2 days?	Fri 6/5/09	Mon 6/8/09

Cost and Labor

Drive Sun fire Plus Rear Wheel Drive Wheelchair - $1530.00

NSC Semi-Rugged Laptop - $1475.00

2 – Roboteq Ax1500 Motor Control Board - $275.00 ea.

8 – 30amp fuses - $2.00 ea.

RABBIT 3K Navigation Board - $400

Emergency Stop - $100

Misc. Parts - $10

Total: $4081.00

Student Labor - $26/hr * 126hr = $3,276

($3,276 * 6students) = $19,656

Advisor/Tech Support - $50/hr * 4hr = $200

Total: $19,856

References

A. www.IGVC.org (Intelligent Ground Vehicle Competition Website) - This is where our group was able to pull specs and information regarding the competition.

B. http://www.electricvehiclesusa.com/ - A site that has various parts needed to build an electric vehicle ranging from frames to controllers

C. http://www.coolrobots.com/builders/newbie.html - This site is dedicated to the discussion and assistance of new and upcoming mobile robot builders. With discussion and links to useful information.

D. http://www.goldenmotor.com/ - A site that has various motors and drive systems, sorted by torque and voltage requirements.