Hi-Techie

2010 Intelligent Ground Vehicle Competition Design Report

Michigan Technological University

Team Members

Senior Design
Matt Gauss – Captain
Mike Mott – Co-Captain
Andrew Carlson
Ameen AlMattar
David Colville
Mitch Knudson

Enterprise
Eric Springer
Scott Hickey
Jeremy Johnson
John VanDussen

Faculty Advisor:
Dr. Jeffrey Burl

Table of Contents

1. Introduction

The Michigan Technological University IGVC Senior Design Team, sponsored by Oshkosh Corporation, will compete in the Design Competition, the Autonomous Challenge, and the Navigation Challenge with the autonomous vehicle named HI-TECHIE. HI-TECHIE has a width of 0.72 m, a length of 1.12 m, a height of 1.66 m, and weighs 217 lbs without the payload. HI-TECHIE's chassis, sensor mounts, and all other component mounts were designed from scratch, and completely self-manufactured. The control algorithms for both the Autonomous and Navigation Challenge were also designed completely from scratch, making More Intelligent Little Friend a truly unique competitor.

2. Chassis Design

The goal was to design a vehicle with the smallest dimensions possible, while still keeping within size specifications, in order to maximize the area in which to maneuver on the course. The mast height is maximized, while still within specifications, in order to allow for the best camera viewing angle. Another important aspect, taken into consideration while designing the chassis, was that every component be easily accessible. Before being fabricated, every part of the vehicle was modeled and tested using UGNX. The 3D model of the vehicle is illustrated in Figure 1.

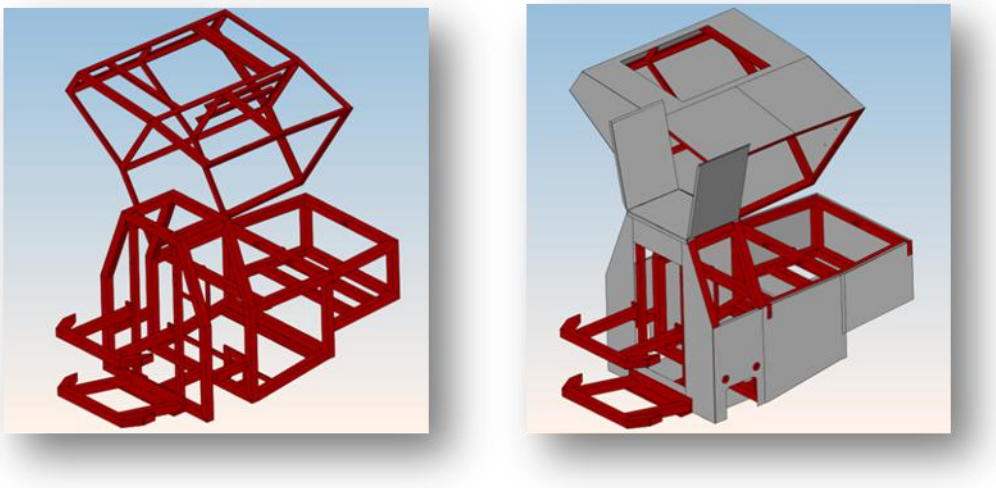


Figure : Vehicle Design with Sheeting (Right) and without Sheeting (Left)

The frame skeleton is built from one inch square aluminum 6063 tubing. Aluminum is the best material for the frame as it is light weight, strong, cost-effective, and easy to manipulate. To increase the overall strength and durability of the frame, all tubing on the frame is welded together using a MIG welder. The frame is designed to have a high safety factor in order to ensure mechanical failure will not occur. The front of the frame is built higher in order to mount the laser rangefinders at their optimal viewing positions. A bumper on the front is used to protect the laser rangefinders in the event of a collision.

Aluminum 5052 sheeting is used to cover the exterior of the vehicle frame. Aluminum sheeting is as durable as the frame and provides a protective barrier for the components housed inside. Foam seals, used at every seam of the aluminum panels, along with rubber grommets, used where wires enter the shell, help keep the components from being exposed to water or any other environmental elements.

The top housing part of the frame is hinged, allowing the 20 lb payload to be placed inside. The hinged hatch provides an efficient way to both securely load the vehicle and access every component within the vehicle. Two butterfly doors also allow for better access to the components, including the switch panel.

In order to accommodate a touch screen monitor, the top hinged portion of the chassis was extended.

3. Drive System

A swivel wheel design was chosen for the vehicle's drive system, consisting of two 12 inch drive wheels in front, each driven by a separate motor, and two 6 inch swivel wheels in back for stability. This design allows for a zero turning radius. The distance between the two drive tires was extended to improve stability and minimize tale swing. The minimum ground clearance is 11 cm, allowing the vehicle to easily traverse sand pits and clear the peaks of hills. With a limited budget and time frame, minimizing complexity and cost was vital. This, along with the fact that the swivel wheel concept is sufficient for all other aspects of concern, makes it the most appropriate choice for the drive system.

1. Drive Motors

Two NPC-T64 motors were selected for the vehicle's drive system. These motors were selected based on several criteria. The first criterion was that they needed to meet the force and speed requirements for a worst case scenario. A worst case scenario was chosen to be when the vehicle, moving at 5 mph, accelerates at ¼ the acceleration of gravity up an incline of 13 degrees. All of these variables exceed IGVC specifications to ensure adequate performance even if the vehicle weight increases. The worst case requirements were compared with the published NPC-T64 dynamometer test results, ensuring the motors would be sufficient.

The motors are designed for continuous use in rugged environments and weather conditions, including light rain and high humidity, which are all possible conditions. The motors feature a built in gear reduction that eliminates the need for buying or constructing a gearbox, thus saving time and money.

The NPC-T64 motors also simplify encoder and wheel mounting. Each motor only requires an adapter plate in order to mount a wheel. A tapped hole in the armature of each motor comes standard, requiring only a threaded shaft to interface with the encoders. The motor also has an extended flange on the rear of the motor that provides protection to the encoders during installation and operation of the motors.

2. Motor Controller

The Roboteq AX2850 motor controller is used to interface the motors with the software. It has two channels that control both drive motors, eliminating the need for two separate controllers. It also has inputs for motor

encoders, allowing for closed loop motor control. The controller has both RS-232 and R/C inputs allowing for both autonomous computer control and manual radio control. The manual radio control feature of the vehicle, while not permitted during competition, is useful in that it allows for easy transportation of the vehicle as well as being helpful for simulation and testing. Another benefit of the AX2850 motor controller is that it has a built in shutdown feature. This, in combination with the radio control functionality, is useful for fulfilling the remote E-stop requirement. Other features include current limiting, motor voltage sensing, and amperage sensing [1].

A custom designed and self-fabricated breakout board is used to interface the motor controller, control computer, and R/C receiver to allow for the ability to switch between radio control and autonomous computer control by means of a toggle switch.

4. Navigational System

The navigational system is the autonomous aspect of the vehicle, controlling the vehicle based on what it senses. The navigation system consists of two laser rangefinders for detecting physical objects, or obstacles. One rangefinder is used for obstacles above ground height and the other detects potholes and changes in terrain. Two cameras are used to detect nonphysical objects, such as lane boundary lines and simulated potholes. A computer with the Windows operating system uses a program, called RoboRealm, to gather data from the cameras and perform the image processing in order to determine the positions of lines and simulated potholes. This data along with the laser rangefinder data, GPS data, and encoder data is sent to a computer with the Linux operating system. The Linux PC uses a program, called Player, to interface the sensor data and the motor controller with a control program implemented in C++. This control program uses the sensor data to control the movement of the autonomous vehicle, navigating the course while avoiding all objects. The hardware connections of the navigational system are illustrated in Figure 2.

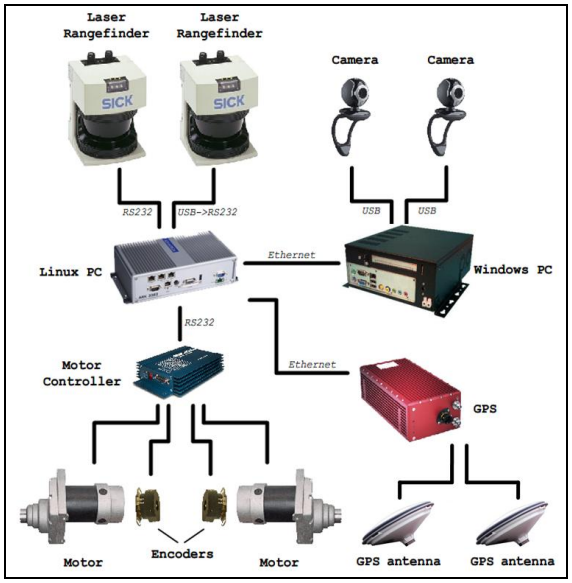


Figure 2: Navigational System Hardware Connections

1. Sensors

1.1. Obstacle Detection

The autonomous vehicle uses two SICK LMS-291 laser rangefinders, mounted on the front of the vehicle, for obstacle detection. Each rangefinder is accurate to within 1 cm with a 180 degree planar field of view. Each rangefinder uses an angular resolution of 0.5 degrees, in order to detect 4 cm wide objects, such as sign posts, about 4.5 m away. The first rangefinder is mounted such that the scanning plane is parallel to the ground and is used to detect obstacles 20 cm above ground level at a maximum range of 8 m. The second rangefinder is mounted above the first, with its scanning plane at a 15 degree angle from the horizontal, in order to detect real potholes and terrain changes. With this downward angle, potholes and terrain changes straight in front of the vehicle can be detected approximately 1.47 m away.

1.2. Lane Boundary and Simulated Pothole Detection

Two Logitech QuickCam Communicate Deluxe webcams are used to detect lane boundaries and simulated potholes. These cameras are small, inexpensive, and adjust well to light changes [2].

The Logitech QuickCam is a color camera, allowing for more options when performing image processing techniques. Through testing and simulation, the camera's video resolution of 640×480 pixels at 30 fps was determined to be more than sufficient. Only a resolution of 320×240 at 20 fps, for each camera, is needed in order to determine the positions of lines and simulated potholes 15 times a second. This means the vehicle, at 5 mph, is able to acquire all object positions every 15 cm, giving the vehicle approximately 3.35 m or 1.5 seconds to react when a line or pothole appears 3.5 m ahead of the vehicle.

Two cameras are used instead of one to increase the field of view. The two cameras are positioned next to one another, facing outwards. This not only increases the field of view, it also allows for separate focus on individual lane boundary lines. The actual field of view of the two cameras is shown in Figure 3, where the coordinate (0,0) is the position of the laser rangefinders on the vehicle.

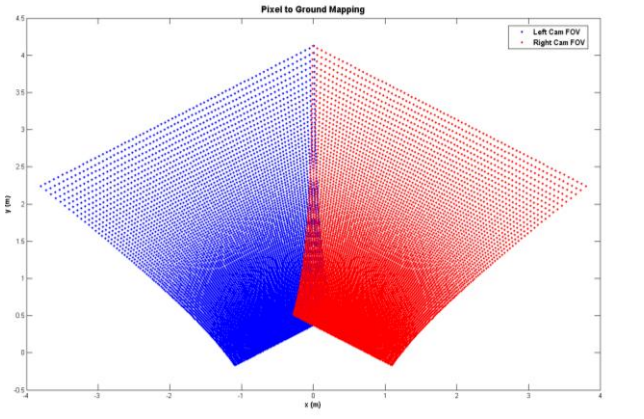


Figure 3: Two Camera Field of View

1.3. Waypoint Tracking

The autonomous vehicle is using the RT3052 Inertial and GPS Navigation System made by OXTS to track waypoints for the Navigation Challenge Competition. This device not only has a dual antenna GPS receiver accurate to about one meter, it also includes three angular rate sensors and three servo-grade accelerometers, used to correct the GPS data [3]. On average, the unit was found to be accurate to less than 0.5 m. The dual antenna setup is used in order to acquire the vehicle's compass heading along with its global position. Also, the unit is easily interfaced with the Linux PC via Ethernet.

1.4. Closed Loop Speed Control

Closed loop speed control is essential to efficiently navigate the courses, as well as being an acceptable method for governing the vehicle's top speed. Two Quantum Devices QR-12 rotary encoders were selected for closed loop motor control. The QR-12 features quadrature output, which allows the motor controller to obtain both the rotational speed and the direction of the motors. The small size of the encoder allows for mounting inside the cast flange of the motors, protecting it from damage and debris. The encoder is powered by the Roboteq AX2850 controller with a line input of 5 volts dc and draws a low current of 65mA. Each encoder outputs 500 pulses per revolution, which equates to a resolution of 1.92mm per pulse [4].

2. Image Processing

The image processing is used for detecting white and yellow lines (lane boundaries), and for detecting two foot diameter white circles (simulated potholes). Because image processing is computation intensive, a panel mount Small PC computer, with a 2.00GHz dual core processor, 2GB of RAM, and a 4GB solid state hard drive, is used. The software being used for image processing is RoboRealm, which is only compatible with Windows. RoboRealm is a powerful robotic vision software application with an interface that makes simulating and testing image processing techniques time efficient. Another benefit to using RoboRealm is that the RoboRealm Server API specifies a socket based communication protocol, allowing other programs to command RoboRealm [5]. This makes it easy to integrate the image processing data with other essential sensor data, in order to control the vehicle.

2.1. Distinguishing White and Yellow from Other Colors

Lines and simulated potholes are not solid, especially when painted on grass, thus color boundaries (intersection of two different colors) are not clearly defined. This, along with sun glare (bright spots) and shadows, makes picking out white/yellow lines and white potholes a challenge. Experimenting with RoboRealm's vast amount of image processing techniques, the following techniques have been found to be useful:

1) Kuwahara Filter – This filter softens the current image and attempts to preserve edges by replacing each pixel with the mean color of neighboring 3x3 pixel groups that have the least variance [6]. The filter gives a more cartoon-like image, making color boundaries more apparent.

2) Scale – This technique is used to reduce the image to a specified size. In doing so, a specified area of the original frame gets reduced to a single pixel, taking the color equal to the mean color in the original frame.

Not only does this improve the processing time by reducing the amount of data, it also flattens or smoothes out the image, reducing the noise within the image.

3) Normalize Color – This removes the pixels without a lot of color, pixels that appear grayer in color. Before this technique is used, a filter is used to make all colors, other than white and yellow, appear gray.

After these techniques and others are performed, thresholding is performed to obtain a binary image, which is then cleared of noise using different blob filters.

Once the resulting image consists of only white blobs, representing where white and yellow is distinguished from other colors, lines (boundary lines) and circles (simulated potholes) need to be extracted from the white blobs in order to determine the positioning of these objects.

2.2. Detecting Lines

In order to detect lines within an image, the Hough Transform is used. The coordinates of the endpoints making up each line are obtained from RoboRealm using an original C++ program. The C++ program then performs camera perspective corrections to the line endpoints in order to determine their actual coordinates (with respect to the vehicle). Once the acquired lines are sorted through, keeping only unique lines, these lines are sent via Ethernet to the Linux PC. These values can then be used in the path finding algorithm.

In the case that no lines are present, the path finding algorithm uses the previous found lines. If only one line is present, a second "dummy line" is created parallel to the known line and at an average lane distance from the known line. These two techniques are important when dealing with dashed lines and imperfect data.

2.3. Detecting Simulated Potholes

Detecting simulated potholes involves locating circles in the resulting binary image. An image processing technique in RoboRealm, called Circles, finds circles with a specified minimum and maximum radius. The same C++ program used to find lines is used to obtain the pixel positions of the circles (simulated potholes) from RoboRealm. Once again, by performing perspective corrections, the actual position values, (x,y) coordinates, are sent via Ethernet to the Linux PC to be used in the path finding algorithm. It was also found that by performing this technique before looking for lines, any potholes found can be removed from the image, making the Hough Transform more efficient.

3. Vehicle Control Software

After extensive research and the implementation of a few well known methods of path finding, such as the Potential Fields Method and the Graph Search Method, it was determined that these methods were insufficient in accomplishing the goals of the competition [7] [8]. Thus, for the sake of both originality and efficiency, new path finding algorithms were implemented for both the Autonomous Challenge and the Navigation Challenge. For both competitions, the new methods rely on calculating path lengths, where a path length is defined as the straight distance in a given direction that the vehicle can safely travel before it contacts an object. Both control algorithms

were implemented in C++ and run on an ARK-3380 imbedded computer with a Pentium M 1.33GHz processor, 1 GB of RAM, and a 4 GB solid state hard drive.

3.1. Autonomous Challenge

The Autonomous Challenge searches path lengths, in the field of view of the lower laser rangefinder, at increments equal to the angle between each distance reading of the laser rangefinder. Thus, if the rangefinder collects 361 data points, the maximum possible number of path lengths to search is 361. The vehicle then uses the greatest path length from the search to determine its direction and speed, where the direction of the path is the direction to travel and the length of the path is used to determine the speed at which to travel. This is illustrated in Figure 4, where the greatest path length is shown in green and labeled Path Length 2.

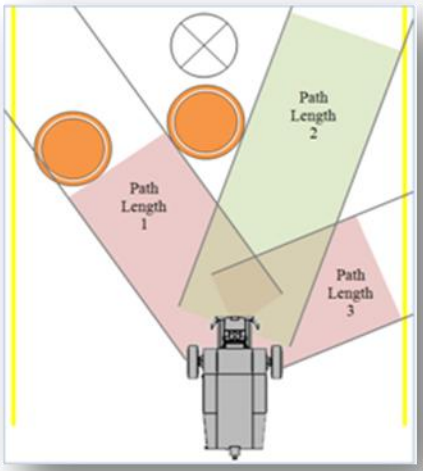


Figure 4: Using Path Length for Autonomous Challenge

A big improvement in the autonomous algorithms is the use of an electronic compass. The compass allows the robot to understand its general direction. As the robot comes to white lines, the current waypoint is moved further down the course. If the robot were to approach a line almost horizontal, it would follow the old waypoint heading and force itself through the course. The waypoint is only updated when the slope of the white lines it sees is greater than one; this is to prevent it from updating the waypoint with horizontal lines. The compass heading allows the robot to know where it came from and prevent it from turning around on the course.

3.2. Navigation Challenge

The Navigation Challenge uses the path length concept in a similar way, except there are no lines to direct the vehicle. Instead, two "dummy lines" are imposed parallel with the heading to the current GPS waypoint, simulating that of the autonomous course. The lines move as the vehicle moves, directing the vehicle toward the waypoint. In order to deter the vehicle from heading the opposite way down the imposed lane (away from the waypoint), when the vehicle searches for a path length at an angle pointing away from the waypoint, it limits the length of the path length found. This can be imagined as imposing a half circle on the back end of the imposed lane, as illustrated in

Figure 5. This way the vehicle can still head away from the waypoint if need be, as in the case when trying to find the opening in the fence.

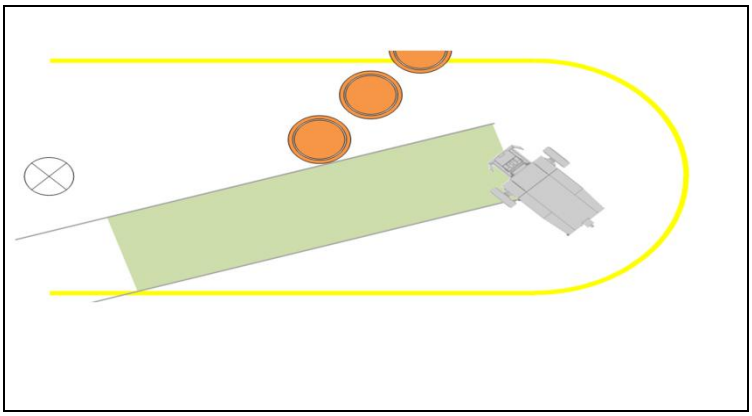


Figure 5: Using Path Length for Navigation Challenge

For the Navigation Challenge algorithm, the path length search begins at the angle of the heading to the current waypoint. If this path length is clear then no other path lengths are searched and the vehicle heads straight for the waypoint. If the path length is not clear then path lengths are searched outward from the heading angle, alternating left and right, for the greatest path length. Similar to the Autonomous Challenge algorithm, the direction of the path is the direction to travel and the length of the path is used to determine the speed at which to travel. The difference is that because the lines are imaginary they can be ignored for determining the length of the greatest path length found. In this way, the direction of the greatest path length found does not change, and the length is determined only by obstacles. Thus, the vehicle's speed is only based on obstacles. The vehicle continues heading toward the current waypoint until the distance between it and the waypoint is less than a specified amount. Once the specified distance is achieved, the vehicle continues to the next waypoint.

In order to keep the vehicle from leaving the bounds of the course, if the vehicle is heading toward a boundary of the course and is within a specified distance of the boundary, the vehicle performs a 180 degree turn, in place.

A problem was found when testing this algorithm that would cause the vehicle to oscillate back and forth when an obstacle was directly between the vehicle and the current target waypoint. This was due to the fact that the GPS data is not as accurate as the laser rangefinder data. Since the GPS is only accurate to about 0.5 m and the rangefinder is accurate to about 0.01 m, the imposed lane seems to jump around with respect to the obstacles found. This means that at one instant the greatest path length may be to the left of the object, and then at the next instant it may be to the right of the object. This error is accentuated by the fact that the algorithm searches for path lengths by alternating left and right. The solution was to save the compass heading of the greatest path length found, and use that heading, as if it were the way to the waypoint, for a certain distance of travel before using the heading of the actual waypoint. While the algorithm is using this "heading lock" the imposed lane width is also decreased, keeping the vehicle from veering too far from the current waypoint.

3.3. Correcting Path Direction

Due to slight miscalculation and the imperfect precision of motor control, it is possible for the vehicle to get too close to an object as it passes by. This results in the same path length for an entire sweep. In order to avoid this problem, our object avoidance algorithm uses a couple of different values. When the closest object to the robot is more than 1.5 meters away, the algorithms believe the width of the robot is 1.5 meters. Now when the robot calculates the longest path, it believes it is larger than actual so the robot will veer further away from objects. When the robot gets into tight places, it chooses the actual width of the robot. A width of .85 meters is used to determine the longest path. This allows the robot to be safe when possible, but has the ability to squeeze between narrow objects.

3.4. Dynamic Speed Control

Another innovative aspect of both control algorithms is the implementation, from scratch, of dynamic vehicle speed control. Basically, the vehicle changes speeds based on the angle and length of the best path length it found. The greater the path length and the less turning required, the greater the speed the vehicle travels at. Also, the sharper the turn, the greater the speed at which the vehicle turns. The motor controller takes both a straight velocity and an angular velocity from the control algorithm in order to command the motors. The max straight velocity is calculated using Equation 1.

$$\mathbf{MaxStraightVelocity = LengthofPath * AngleToTurn} \quad (1)$$

The straight velocity that we send to the motors depends on the current speed and the next calculated speed. If the new speed calculated is faster than the robot is moving now, the robot slowly accelerates to the new speed. If the new speed calculated is slower than the current robot's speed, the robot slows down to the speed quickly. Controlling the acceleration and deceleration allows us to have a large max velocity, but use it only when there are no objects around. Formula 2 shows how the control works. X is a number between 0 and 1, where the larger X is the faster it will accelerate or decelerate to the calculated max velocity. Therefore when we want to accelerate we use a small X value and when we want to decelerate we use a bigger value of X.

$$\mathbf{StraightVelocity=CurrentSpeed*(1-X) + MaxStraightVelocity *X} \quad (2)$$

Using Equation 3, the angular velocity is calculated based on the difference between the angle of the best path length and the straight angle, similar to the straight velocity. This results in a negative angular velocity for a left turn and a positive angular velocity for a right turn.

$$\mathbf{AngularVelocity=MaxAngularVelocity\times BestPathLengthAngleStraightAngle-1} \quad (3)$$

The dynamic speed control allows the vehicle to maximize its speed during the competition, thus minimizing the time to complete the courses, by increasing its speed when it is safe and decreasing it when it must be more careful.

3.5. Algorithm Speed and Reaction Time

Both the Autonomous and Navigation algorithm loop through the processes of gathering data from the sensors, finding the best path length, and commanding the motors appropriately. One loop through these processes is considered a cycle of the algorithm. The speed and reaction time for both algorithms is based on the time it takes to complete a cycle. The worst case in finding the best path length is when every path length must be searched. In this case, the number of path lengths to search is equal to the number of data points from the laser rangefinder (361). For every path length searched, every data point of the laser rangefinder must also be searched to determine whether or not the data point is in the path. Thus, 361^2 checks will need to be performed in order to determine the closest data point to the vehicle in the path. Note that the distance to this point is the path length. Although this was implemented and determined acceptable in terms of reaction time, it was improved.

To improve the speed of the algorithms, we processed our data differently. We found simplifying our data into one set of rangefinder data made our calculations easier and faster. Figure 6 shows how white lines from the cameras and barrels from the rangefinders are viewed. Instead of keeping lines and barrels separate, they all return a distance to one set of data. The final set of data is used to determine our longest path the robot can take.

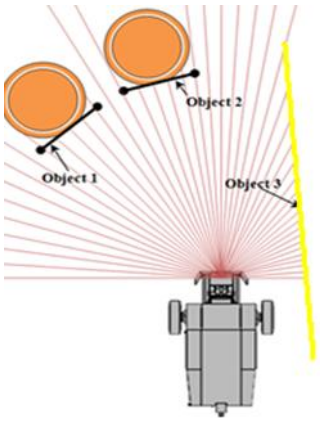


Figure 6: Converting All Data To Rangefinder Data

The average run time for one cycle of the Autonomous algorithm is 108 ms. This means that if the vehicle is traveling at 5 mph, the vehicle corrects/updates its path of travel every 0.24 m or 9.3 times a second. The average run time for one cycle of the Navigation algorithm is 67 ms. Thus, the vehicle corrects/updates its path of travel every 0.15 m or14.9 times a second.

5. Hardware and Software Integration

A large part of any embedded software project is writing code to interface with the hardware peripherals. This task was accomplished using Player. Player is an open source robotic hardware abstraction layer (HAL), which provides a clean and well-defined API to access and control the vehicle's hardware by defining standard "interfaces" as illustrated in Figure 7 [9]. These interfaces hide the differences in the underlying hardware from the control software.

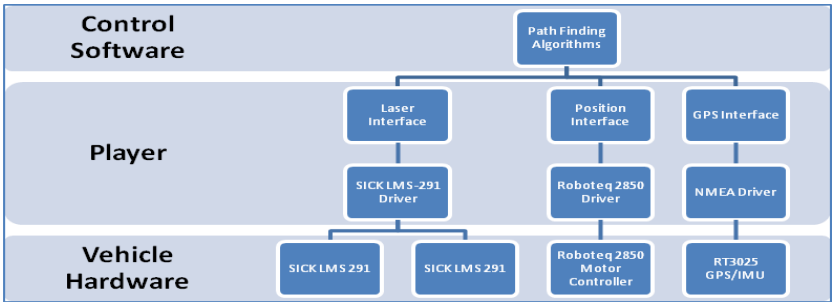


Figure 7: Hardware Abstraction Layers

Drivers then implement each of these interfaces and communicate with the underlying hardware devices. A configuration file is loaded into Player to define the available hardware and interfaces.

The GPS, laser rangefinder, and motor controller drivers are included in Player. However, the motor controller driver needed to be improved, as the original provided only a minimal level of functionality. Support for position reporting from the encoders, writing flash configuration parameters, and motor power reporting were all added. Player is a network server, which means multiple clients can access the robot hardware over the network. This capability enables remote logging and control of the robot remotely from any computer. Thus, sensor and control software data can be collected and analyzed during the testing of the autonomous vehicle. Player also includes a full suite of simulation software (including 3D simulation). This simulation software generates sensor data, which is provided to the control algorithms through the Player server. Thus, from the perspective of the control algorithm, there is no difference between the simulation and the actual test runs, making simulation a useful tool for testing.

6. Power

Two Tempest TR45 12 volt sealed lead acid batteries were chosen for the power supply, as they are the most inexpensive option while still meeting and exceeding the specifications. Two were required, wired in series, to obtain a terminal voltage of 24V. For the 12V components we installed a 12-24V converter. The maximum power draw and the operating voltage for each component are shown in Table 1.

Table 1: Power and Voltage Draw from Individual Components

Component	Max Power Required (W)	Voltage (V)
Motors	1200 per motor	24
Motor Controller/Encoders	5	24
Laser Range Finders	20 each	24
GPS	20	12
Computers	54 each	24
Monitor	20	12

Each battery has a 20 hr rating of 50 amp-hours and weighs 33 lbs [10]. At maximum power draw, the batteries are estimated to last 30 minutes. With actual testing, the average run time was found to be approximately 4 hours. This allows for adequate testing time and more than adequate run time during competition.

The vehicle also features an onboard charger that charges both batteries simultaneously. The approximate charging time at maximum current is 5 hours.

When the robot was received at the beginning of last semester, the wiring was very clustered together. The robot's motor controller was damaged beyond repair. The wiring was the same color and each individual devise used the chassis as a ground. This would normally be acceptable, but it was found that the motor controller did not have a control ground, and for this case, it was attempting to use the motor ground and the housing of the controller as a ground also. This was causing the replacement motor controller to fail instantly.

We solved the problem by starting from scratch on the wiring of the robot, as shown in Appendix B. We stripped the robot down to bare chassis and added components individually. When the new wiring was installed, it was done so that the wires could be easily traced by a documented color coding system. While running all of the 24 volt positive wires, individual ground wires were also installed for each devise.

The previous protection system of the robot was composed of 5A circuit breakers that used a standard off then on reset routine. We replaced the circuit breakers with a Cooper-Bussmann 15600 Series automotive fuse block with a ground pad. This allowed us to use one main conductor to supply the fuse block then distribute out after the fuse block. This improves simplicity because each battery only has two conductors attached to it, rather than 7 as before. Because they were using the circuit breakers for switching also, we decide to use standard single pole, single throw toggle switches rated at 15A. This forced the team to realize when there was a problem, instead of just cycling the circuit breakers allowing them to reset, a fuse must be replaced.

The robot uses both 12 and 24 volts for the different devices. The way the previous team dealt with this is they used only one of the batteries to supply power to the 12 volt devices. Because we went to a centrally distributed fuse block, only 24 volts was supplied to the devices. To solve this problem, we installed 24-12volt voltage converters on the 12 volt devices. This allowed us to standardize the voltage on the robot to 24 volts. Another benefit of the voltage converters is the fact that both batteries share the load, where as last year, one battery was taking more load than the other.

7. Safety

The vehicle has several safety features. A mechanical Emergency stop or E-stop, in the form of a red push button, is located at the rear of the vehicle. Pushing the E-stop disconnects power to a solenoid, which in turn kills the power to the motor controller. A remote E-stop, controlled by a hand held R/C transmitter, will also shutdown the Roboteq motor controller, thus stopping the motors. With the implementation of the GUI on our linux PC we've also implemented a very large electronic stop button on the touch screen.

8. Team Organization

The MTU IGVC Senior Design team consists of six undergraduates with very little or no experience in robotics. With little experience, many tasks were shared among multiple team members for best results. A summary of the team members and the major tasks they accomplished are provided in Table 2. Despite the inexperience, the 1500 hours put in over two semesters is believed to make More Intelligent Little Friend a strong competitor.

Table 2: Team Member Contributions

Matt Gauss (Captain)	*Control Algorithms, Navigation Implementation*
Mike Mott	*Code Debugging, Documentation, Hardware Installation*
Dave Coleville	*Electrical Wiring, Documentation*
Andy Carlson	*Electrical Wiring, Documentation*
Mitch Knudson	*Chassis Modification, Solid Modeling*
Ameen Almatter	*GPS Communication, Algorithm Testing*
Eric Springer	*GUI, Algorithm Testing*
Scott Hickey	*GUI, Algorithm Testing*
Jeremy Johnson	*GUI, Electronic Compass*
John VanDussen	*Part Catalogue, Documentation*

9. Design Process

This is the second year our robot is entering this competition, many of the physical parts of the robot haven't changed much. The team last year designed and fabricated almost the entire robot, so our design task this year was to improve it. With little knowledge in robotics to work from, most of the first semester involved learning how to operate the robot and figuring out what design changes would be most beneficial. Many of the vehicle's main components were donated thanks to our sponsors. The design of the original vehicle was built around these components any new components required were purchased with this year's budget. A list of the donated and purchased components is found in Appendix B.

A decision matrix was used to determine what design changes and additional components would most benefit the project. Selected design improvements include adding a touch screen to run and change code more easily without opening the robot, rewiring the robot for better organization, robustness, and to include a fuse block, and creating new algorithms for the navigation and autonomous challenges.

The first semester was primarily spent trying to get the robot to move. We quickly figured out that the motor controller we inherited was burned out. While we were waiting for a replacement we began to prepare the chassis for the monitor, a new enclosure for the breakout board was designed and fabricated, and the robot was re-wired.

During the second semester, we finally got everything running properly so we could start testing the robot. After a few testing sessions ,the task of revising the programming was split into three tasks. The autonomous code was rewritten, the navigation code was being troubleshooted, and a GUI was developed for use on the Linux computer. The rest of the team worked on any piece of documentation, finalization, or fabrication that needed to be done.

The first major problem encountered was the replacement motor controller. After installing our new motor controller, the robot still failed to function. Assuming that we had received a faulty controller we asked for a replacement. Two controllers later we discovered there was some faulty wiring involved. After thoroughly re-wiring everything connected to the motor controller, we finally got the robot to move. The only other significant issue was the GPS system. After exhaustively troubleshooting the navigation code, a heading from the GPS could not be obtained. To solve this problem, a relatively inexpensive electronic compass was implemented.

A large amount of time was spent making sure there was adequate documentation this year to prevent any future teams from having the kind of problems experienced by this year's team. So even though the documentation isn't vital to have a working robot , it is important to make sure any future teams can concentrate on improving thedesign, instead of having to figure out what was done.

With all major problems overcome, the Michigan Technological University IGVC team looks forward to competing in the 2010 Intelligent Ground Vehicle Competition with a truly innovative vehicle, Hi-Techie.

10.Resources

[1] Roboteq Inc., "Product Information AX2850," Roboteq. Available: http://www.roboteq.com/ax2550-folder.html. [Accessed: Nov. 16, 2008].

[2] "QuickCam Communicate STX," 2008. [Online]. Available: http://www.logitech.com/index.cfm/webcam_communications/webcams/devices/352&cl=US,EN. [Accessed: Oct. 12, 2008].

[3] Oxford Technical Solutions, "RT3052," *OXTS Inertial+GPS*, 2008. [Online]. Availible: http://www.oxts.co.uk/default.asp?pageRef=96. [Accessed: Nov. 15, 2008].

[4] Quantum Devices Inc., "QR12 (1.22) Diameter Optical Encoder," *Quantum Devices*. [Online]. Available: http://www.quantumdev.com/products/optical_encoders/qr12.html. [Accessed: Nov. 16. 2008].

[5] *RoboRealm Vision for Machines*, RoboRealm, © 2008. [Online]. Available: http://www.roborealm.com/help/index.php. [Accessed: Nov. 11, 2008]

[6] IMAP-VISION, "Kuwahara." [Online]. Available: http://www.ph.tn.tudelft.nl/~imap/library/wouter/kuwahara.html. [Accessed: Nov. 11, 2008]

[7] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," IEEE Transactions on Systems, vol. 19, no. 5, pp. 1180-1182, Sep. 1989.

[8] Georgia Institute of Technology, "RoboJackets IGVC Design Report," *igvc.org*, pp. 11-12, June 2007. [Online]. Available: http://www.igvc.org/design/reports/dr175.pdf. [Accessed: Nov. 15, 2008]

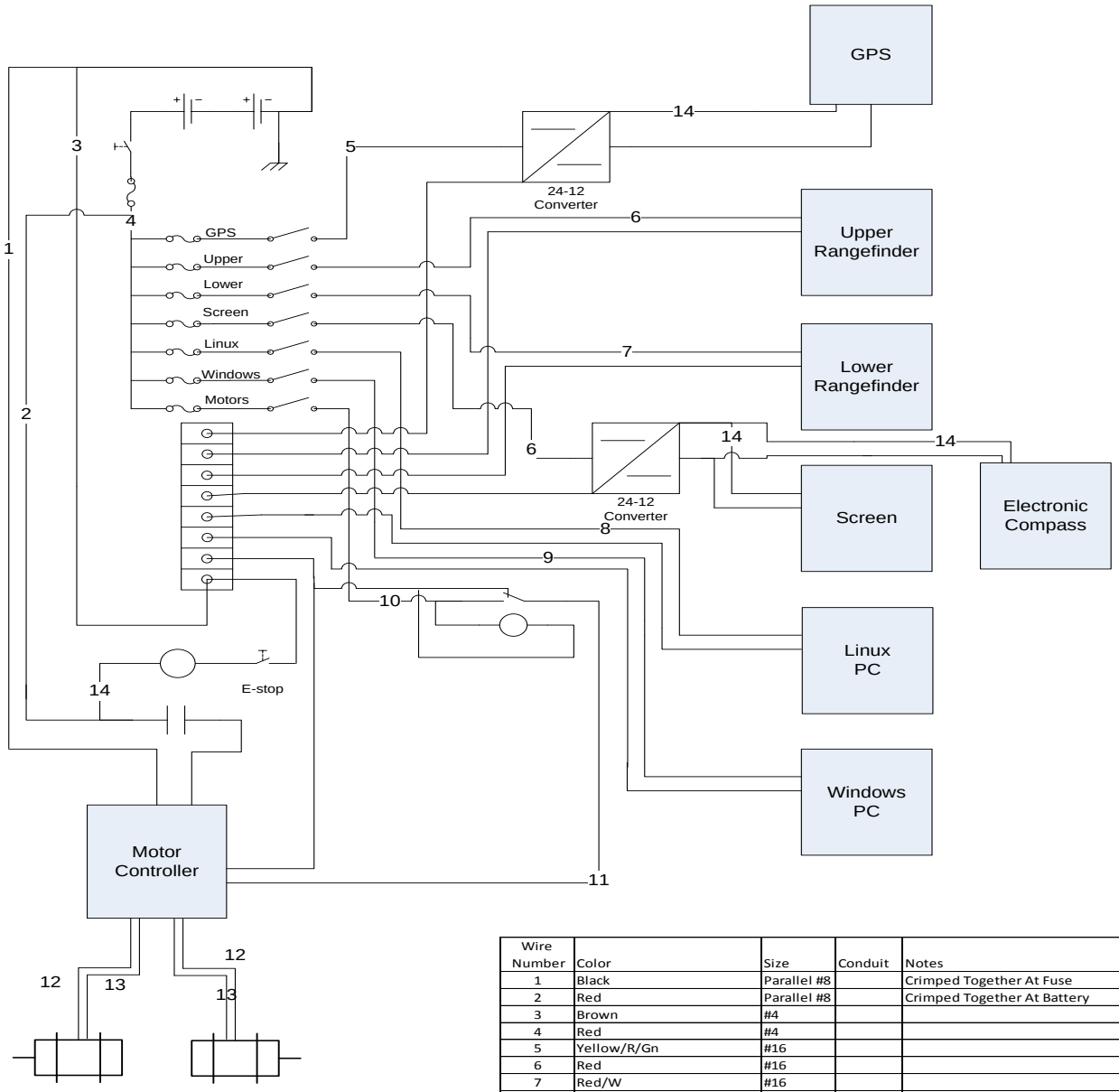
[9] B. Gerkey, "The Player Robot Device Interface," *Player*, Sept. 2005. [Online]. Availible: http://playerstage.sourceforge.net/doc/Player-2.1.0/player/index.html. [Accessed: Nov. 16, 2008].

[10] Universal Power Group Inc., "Data Sheet for UB12500," *UPG*. [Online]. Available: http://www.universalpowergroup.com/batteries/sla.aspx. [Accessed: Nov. 16. 2008].

[11] "American Wire Gauge table and AWG Electrical Current Load Limits," 2003. [Online]. Available: http://www.powerstream.com/Wire_Size.htm. [Accessed: Nov. 2, 2008].

[12] "Blue Sea Systems – ANL Fuses," 2006. [Online]. Available: http://bluesea.com/productline/overview/135. [Accessed: Nov. 15, 2008].

Appendix A



Wire Number	Color	Size	Conduit	Notes
1	Black	Parallel #8		Crimped Together At Fuse
2	Red	Parallel #8		Crimped Together At Battery
3	Brown	#4		
4	Red	#4		
5	Yellow/R/Gn	#16		
6	Red	#16		
7	Red/W	#16		
8	Yellow/Bl	#16		
9	Blue	#16		
10	Pink	#16		
11	Yellow spliced with Orange	#16		Soldered in conduit
12	Green	#4		
13	White	#4		
14	Orange	#16		Signifies 12 Volt Supply, Anderson connectors in different orientation to prevent mistaken connection.
Unmarked	Black	#16		Ground wire, travels with live pair.

Appendix B

Last Year's Contribution	***Team Cost***
4 Tires	*$111*
2 Laser Range Finders (Donated)	*$0*
2 Computers (Donated)	*$0*
Motor Controller	*$637*
2 Motors	*$596*
Encoders (Donated)	*$0*
2 12V Sealed Lead Acid Batteries	*$233*
Battery Charger	*$216*
2 Web Cameras (1Donated)	*$80*
Miscellaneous Hardware	*$400*
Dual Antenna GPS/IMU (Donated)	*$0*
Chassis Material	*$409*
R/C Transmitter	*$150*
Total Year 1 Cost	*$2832*
This Year's Contribution	
Touch Screen Monitor	*$976.36*
Electronic Compass	*$360.00*
KVM Switch	*$34.35*
2 24-12 volt Converters	*$60.00*
Various Connectors and Components (Donated)	*$0*
Total Year 2 Cost	*$1430.71*
Total Ongoing Project Cost	*$4262.71*