Lawrence Technological University

Viper II

IGVC 2009 Autonomous Vehicle

Team Members

Gary Givental, Philip Munie, Shawn Ellison, Brandon Bell, Ze Cheng, Taiga Sato, Bryan Koroncey, Nathan Lucas

Faculty Advisor Statement[1]

I, Dr. CJ Chung of the Department of Math and Computer Science at Lawrence Technological University, certify that the design and development on Viper II has been significant and each team member has earned credit hours for their work.

_____ _____

Dr. CJ Chung (chung@ltu.edu, 248-204-3504) Date

[1] Co-advisor: Dr. Lisa Anneberg, Department of ECE

Table of Contents

1. Introduction

For the 2009 Intelligent Ground Vehicle Competition (IGVC), Lawrence Technological University (LTU) presents Viper II, the product of a collaborative effort among a diverse and multidisciplinary group of LTU engineering and computer science students. Viper II is an enhanced robot, with cutting-edge hardware and software updated for 2009. It represents Lawrence Tech's latest venture into developing a safe, reliable, and cost-effective autonomous vehicles that are rich in progressive technologies.

2. Innovations

2.1 Hardware Platform

2.1.1 Mechanical System

Viper II is a three wheel vehicle with front differential drive steering, a rear caster wheel, and an aluminum chassis that are all encased in a custom fiberglass mold. This design provides the most stability and maneuverability, and it allows for a zero-degree turn radius. Furthermore, the vehicle's front suspension system helps stabilize the drive-train and minimizes the shaking of the camera pole.

2.1.2 Electrical System

Viper II uses a manual electrical system that is controlled via a single electric box. There's also hardware in place (a Programmable Logic Controller (PLC) coupled with a Human Machine Interface (HMI) display) that extends the manual system to create the Electronic Diagnostic Fail-Safe System (EDFSS).

2.1.3 Software

The software for Viper II is written in the C# programming language using Visual Studio 2008 and Microsoft XNA. XNA is a set of tools with a managed runtime environment provided by Microsoft that facilitates computer game development, and it allows Viper II to quickly perform image processing on the GPU. Additionally, Viper II includes a Sensor Fusion module to bring sensor information together into one component and an easily pluggable motor control interface for its movement. Moreover, custom Local and World map software help Viper II keep track of lane, obstacle, and GPS waypoint information, and they allow it to be less reactive to the immediate environment. Furthermore, Viper II utilizes a GPS Breadcrumb system to keep track of recently visited locations, and to detect when it is heading in an incorrect direction on the course. Lastly, Viper II software is JAUS level 4 compliant.

3. Design Process

3.1 Collaborative Team Organization

2009 Team members are:

Gary Givental, MSCS – Team Captain	Philip Munie, MSCS – Autonomous Lead
Brandon Bell, MSCS – Hardware Integration Lead	Shawn Ellison, MSCS – Navigation Lead
Nathan Lucas, MSCS – JAUS Lead	Taiga Sato, BSCS
Bryan Koroncey, BSCS	Ze Cheng, BSCS

Since a large team required more communication, team members used an online discussion forum, Subversion code repository, and email to communicate effectively. Weekly meetings were scheduled to allow for status reports and to help resolve emerging issues.

3.2 Project Planning Process

The development for this year's IGVC effectively started in October 2008. **Figure 1** describes the project plan followed by the team. For this year's competition, the team met every week to go through the design plan and to perform integration testing. Since an agile, iterative development process was used, the design emerged over time as different options for the software were explored. This project plan acted as a guideline to keep the team on track, and to make sure core timelines were met.

Task	Due Date
Initial Brainstorming	2/1/2009
File Repository Setup	2/1/2009
High Level Software Design	3/1/2009
High Level Hardware Update Design	3/1/2009
Software Development	5/1/2009
E-Stop Updates	5/15/2009
System Testing	5/15/2009
Written Report	5/18/2009
Oral Presentation	6/7/2009

Figure 1: Project Tasks

3.3 Development Process

The team used agile development and set-based, concurrent methodologies for the software design in this year's competition. The set-based, concurrent development allowed the team to investigate several options for software modules and algorithms, and to find the best solution to design issues. Correspondingly, the agile development approach allowed the team to avoid the well-known pitfalls of the linear "waterfall" style approach. These techniques enabled an emergent, iterative methodology that proved to be the perfect fit and allowed for the functionality and features of the software and hardware to evolve through continuous testing and integration. The team scheduled regular weekly meetings to present status reports, to discuss progress, and to indicate any road-blocks encountered by team members during development. Frequently, students teamed up to work on a particular problem in hardware or software to get it resolved quicker, and to leverage the "pair-programming" concept.

3.4 Testing Process

A test track was setup on LTU campus, which was complete with various kinds of obstacles and curves in the lanes to approximate the challenges of the real competition. This allowed team members to collect real world data and to discover failures in software logic.

3.4.1 Hardware Testing

Viper II's electrical and mechanical systems were tested thoroughly after Viper was showcased to TARDEC, and extensive field testing was performed once the robot was re-assembled. While performing field testing, the team discovered several issues that needed to be addressed. The robot's batteries drained quickly, so a generator was obtained to keep them charged. Additionally, the GPS signal proved to be inaccurate, so the team subscribed to the Omnistar HP service to get 10-centimiter precision.

3.4.2 Software Testing

The team performed extensive unit testing, systems testing, and integration testing of the code throughout development. As new functionality was developed, it was demonstrated to the rest of the team, and, when possible, tested on the robot. Test runs were performed regularly on the test track at LTU.

3.4.3 System Integration

System integration testing was executed as quickly as possible once the development of the core components was complete. When new modules were available for general testing, they were immediately integrated into the overall system and used by all team members for testing. This modular design of the software components made it easy for new functionality to be plugged into the overall architecture. The team extensively tested the integration between the software and all the sensor hardware, as well as the motor control.

4. Hardware Design

4.1 Robot Structure

4.1.1 Chassis

Viper II is a three wheel vehicle with front differential drive steering and a rear caster wheel. This design allows for zero-turn radius maneuverability and simplified motion control. The chassis has a minimum of 4 inches of ground clearance, and this allows the robot to climb hills and ramps with an approach angle of 23 degrees. The 3-D CAD model shown in the follow **Figure 2** illustrates the design of the chassis using the ¾ inch 6063 aluminum tubing, which gives the robot the capability of carrying all of the necessary components, as well as the designated payload required for the competition. The chassis design enables Viper II to maintain its structural integrity throughout the rugged off-road course.

4.1.2 Suspension

Viper II is designed with an independent suspension that absorbs impact in the wheels from the off-road terrain. This design provides ample stability to the camera, thus allowing for a smooth video image feed. The suspension model in **Figure 3** shows the coil spring over a tube style shock that is mounted to independent control arms. This suspension style gives the robot 1 inch of travel in the shocks, and it greatly reduces the twist and roll of the chassis and components. Furthermore, the shocks are rated for 350 lbs per inch and have an adjustable rebound feature.

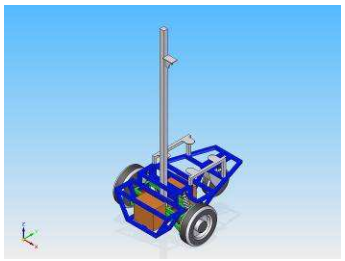

Figure 2: Chassis Model

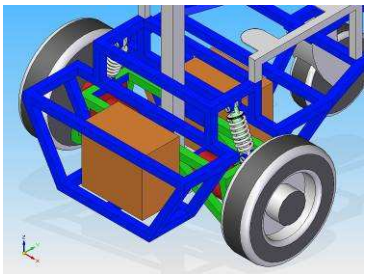

Figure 3: Front Suspension

4.1.3 Drive Train

Viper II's design uses two 24 volt, permanent magnet type model NPC-T74 high torque motors with a 20:1 gear ratio that gives the robot the ability to generate up to 1480 in-lbs (123.3 ft-lbs) of torque and allows for a maximum speed of 10.4

MPH (at 248 RPMs). Thus, these motors give Viper II plenty of power, and they eliminate the fear of premature motor failure. They ensure that Viper II can easily navigate through rough terrain, and go up a 15 degree incline.

4.1.4 Body

The body that is covering and protecting Viper II is fabricated from fiberglass, and its sleek design with smooth flowing curves and lines gives the feel of its namesake. It is designed to give easy access to the internal components through an easily removable top section, and it is fitted with electric fans to ensure that heat is removed from inside the robot. Furthermore, a tray that slides out of the right side of the fiberglass body allows access to Viper II's internal laptop computer, and it can be used without removing the top section.

4.2 Motors and Electronics

4.2.1 Motor Control

Motor control is facilitated by a Roboteq AX3500 60A/channel controller with a C# serial port interface. The AX3500 provides velocity feedback measurements through optical encoders that are mounted on the motor shafts. Additionally, the E-Stop is wired to the controller's main drive power lines via a mechanical relay that can be triggered by the top-mounted kill switch or by the wireless control system.

4.2.2 Sensors

In keeping with the theme of modularity, all of Viper II sensors (and controls) utilize RS-232 or USB 2.0 standard interfaces. **Table 1** summarizes the sensors used in Viper II.

Sensor	Function
Optical Encoders	Motor shaft position feedback for servo control
NovaTel ProPack-LB DGPS Unit	Global positioning
Digital Compass/Inclinometer	Heading, roll, and pitch information
High Dynamic Range DV Camera	Capture field image stream
Sick Laser Measurement Sensor	Provides a 180 degree polar ranging array

Table 1: Vehicle Sensor Summary

4.2.3 E-stop

Viper II is equipped with both manual and wireless (RF) remote emergency stop (E-Stop) capabilities. The Excalibur RS-310 Remote and Start Keyless Entry manufactured by Omega Research is used as the wireless E-Stop component, and its door lock function is integrated into Viper II's E-stop system. When this E-Stop is activated, it removes power from the drive train and engages both of the front wheel failsafe electromagnetic brakes.

4.2.4 Vehicle Alert System (JAUS Horn and Lights Hardware)

Viper II includes an electrical module that can switch relatively large electrical loads (using relays) to the horn and light systems via commands sent over a USB interface. It is utilized by the vehicle's alert system, and it is activated by Viper II's JAUS software.

4.3 Electrical System

4.3.1 Power System

Viper II has two parallel electrical systems: a logic control system and a manual control system. The logic control system consists of a Programmable Logic Controller (PLC) and a Human Machine Interface (HMI), and these components make up the Electronic Diagnostic Fail-Safe System (EDFSS). The manual system consists of toggle switches and fuses that allow the team to physically turn on and off each electrical component. The operating mode can each be selected by a keyed selector switch found on the main system control board.

4.3.2 Electronic Diagnostic Fail-Safe System - EDFSS

The EDFSS provides Viper II with a nearly endless range of abilities for its current design and future design modifications. It makes the use of an external touch screen display that can be programmed to control each component's power and to display status information.

4.3.2.1 External status indicator lights

During previous competitions, the team realized that easily visible power indicators for key component were needed during testing. The EDFSS solves this problem by controlling three indicator lights: the green light specifies all systems are ready, the yellow light signifies that Viper II is operational, but not all systems are responding, and the red light indicates that the system is not operational. Additional indicator lights are also available next to each manual switch on the electrical box. This functionality allows the IGVC team to quickly identify power issues during testing.

4.3.2.2 Manual Control

The manual system is a basic on/off component control. It acts as a backup electrical system if there are problems with the EDFSS system.

4.3.3 Power Source

Viper II is powered by two 12V 50 Ah AGM batteries connected in series to provide a 24V electrical system. It uses a 24/12V DC/DC converter for the onboard 12V systems and an onboard charger with 110VAC interface for restoring battery power. **Figure 4** illustrates how power for the left and right motors is fed directly into the motor controller to maximize the motor power. However, the control power for the motor controller is regulated by the E-Stop and electrical control box.

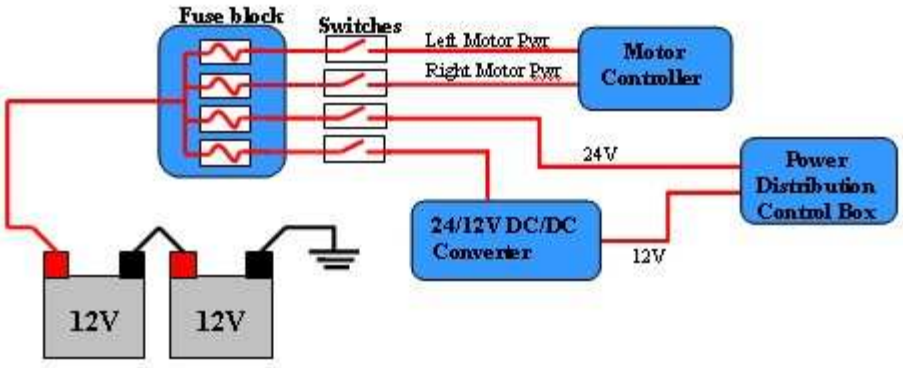

Figure 4: Power Source Schematic

4.3.3 Power Distribution

Viper II is equipped with an all-in-one electrical control box. This power distribution control box contains the systems power distribution and emergency stop printed-circuit board (PCB), JAUS PCB hardware, wireless emergency stop PCB hardware, power control switches for each component, and the main system selector switch that controls the manual and EDFSS operation modes. It is designed to be self contained and removable from Viper II, and it also offers additional testing and diagnostic abilities for system voltage and current ampere measurements.

The two main power connectors are automotive grade EDAC panel connectors, and they allow for an easy connection point for two wire harnesses, which route power to and from each hardware item and the EDFSS block. The power and communications control system schematic for Viper II vehicle is shown in **Figure 5**.

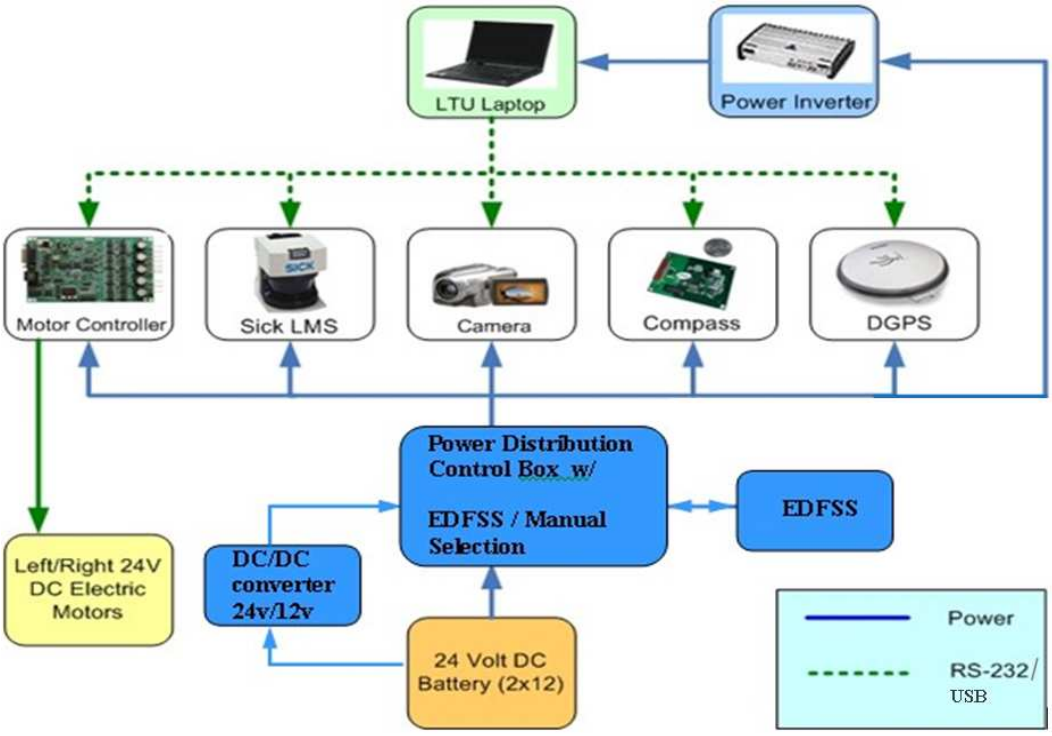

Figure 5: Power and Communications Control System Schematic

5. Software Design

5.1 Software Strategy

C# , XNA, and the Visual Studio 2008 IDE were chosen for all software development on the robot this year. C# is a powerful object oriented language, and Microsoft XNA is a video gaming toolset which allowed Viper II to utilize the processing power of GPU for image analysis. The Visual Studio IDE allowed for rapid application development with easy to build visual interfaces. **Figure 6** shows the high level software architecture design.

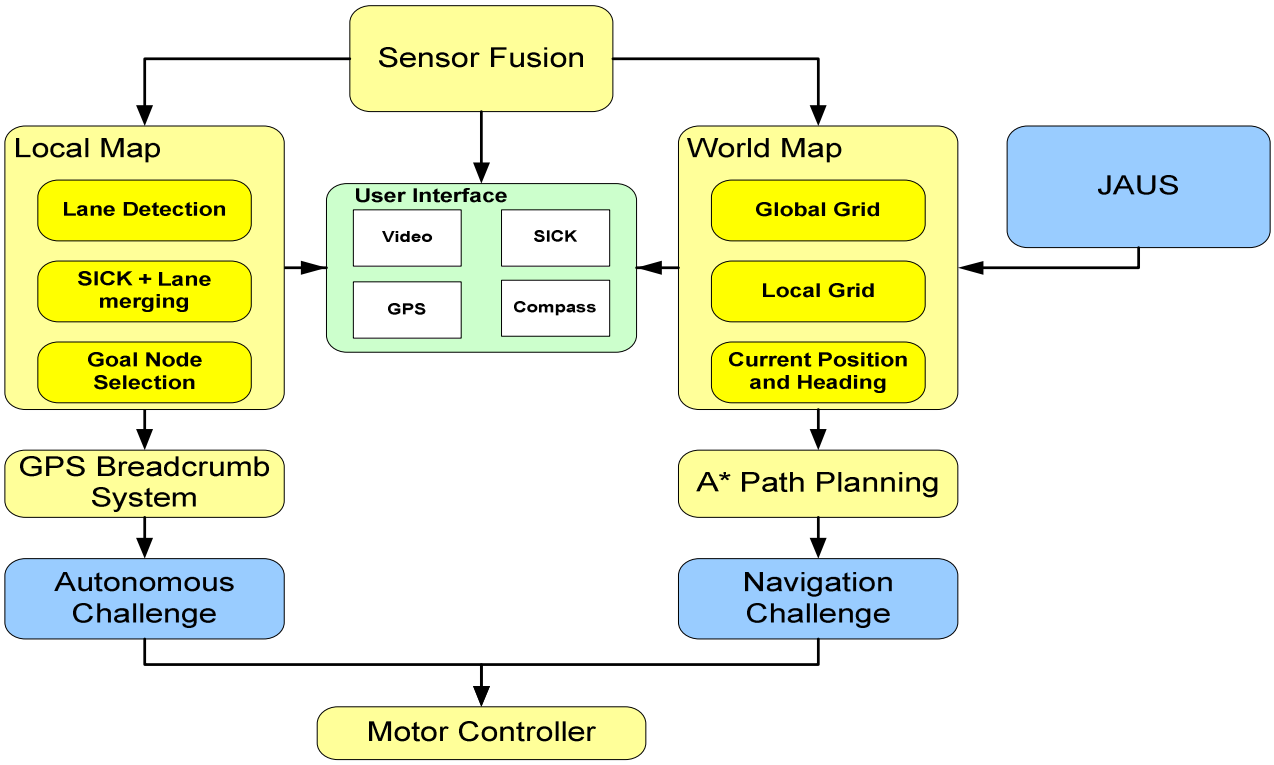

Figure 6: High Level Software Design

5.2 Sensor Fusion

For this year's competition, a Sensor Fusion Module shown in **Figure 7** was developed to collect data from all the hardware sensors, and to present this data to all subscribers in a consistent manner. This allows subscribers to access sensor data without knowing the details of hardware layer implementation.

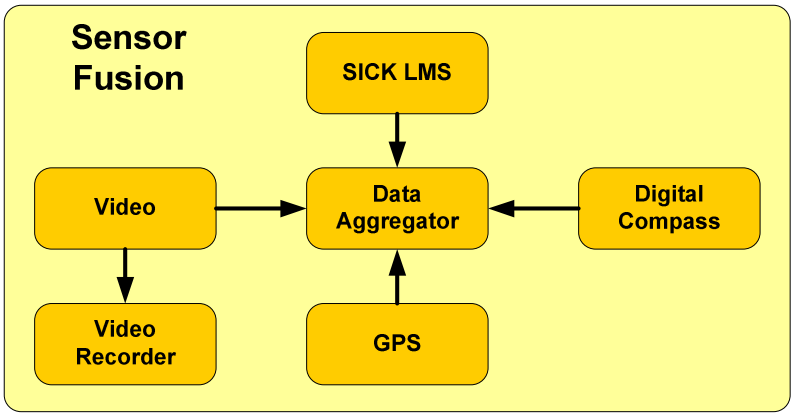

Figure 7: Sensor Fusion

5.3 World Map and Path Finding

The world map plots a geographic region to a Cartesian coordinate plane. The total size of the Cartesian map is determined by the size of the area to be represented and the desired resolution. Generally, a resolution of ½ meter has proven to work well during testing. Each point in the geographic region can be mapped to a cell in the Cartesian map, and vice versa, through the use of a conversion object. In this way, a simple A* path finding algorithm can be used to find the shortest path between any two given points.

5.4 Motor Control Module

The motor controller module was designed to be easily plugged into any of the team's projects. Once initialized, this module can accept simple operation commands for speed and direction, which it then translates into the ASCII communication strings that are sent to the motor controller via a serial connection. This module allows team members to have simple access to the hardware without knowledge of the motor controller's underlying communication protocols.

5.5 Autonomous Challenge Details

The main algorithm for the autonomous challenge consists of using computer vision to identify lane information and SICK data output to detect obstacles. To accomplish this, the camera is first calibrated so that the vision processing can map lanes onto a local map alongside the SICK data, and then a goal node selector algorithm is used to find the appropriate heading. Additionally, a GPS Breadcrumb utility, which uses GPS location information and compass headings to store a list of recently visited locations, is used to detect if the robot has become turned around. This helps the robot avoid going backwards on the course.

5.5.1 Camera Calibration

The camera calibration is responsible for synchronizing the lane data with the SICK obstacle data. It works by calculating the distance of each pixel in the vision's captured image, and it uses the SICK as the reference point. It is accomplished by measuring the distance of predefined positions on the captured image and interpolating the remaining positions on the image. A snapshot of the tool for calibration is shown in **Figure 8**.

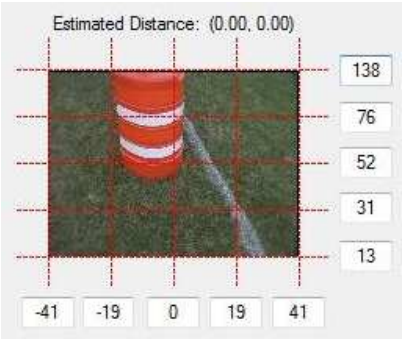

Figure 8: Camera Calibration Tool

5.5.2 SICK Data Gathering

As the SICK data is collected, it is mapped onto a local map, combined with the filtered camera image. Because of the calibration previously, the data can simply be placed in the local map without any extra processing. See **Figure 9**.

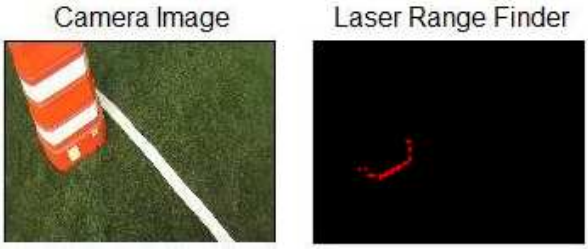

Figure 9: Example of mapping SICK data to the Local Map

5.5.3 Image Processing

For this year's competition, Viper II continues to use a local map approach for combining the vision information with obstacle data. First, SICK obstacle data is added into the local map as red pixels. Then, the camera image is captured and converted to black and white using a brightest pixel filter. This filter runs on the GPU and chooses the brightest pixel on each horizontal line. Then blob detection is used to filter out noise pixels that are not considered to be lanes. Once noise has been removed, the filtered camera image is added to the SICK data and all of the white pixels that are vertically above the red pixels are repainted as black. A Hough Transform filter is then applied to find any partial lanes in the image and fill them in. Finally, a bleeding filter is used to color any pixels vertically above red pixels as red, and above white pixels as white. Thus, the local map represents both the lane information in white pixels and the SICK obstacle data in red pixels. **Figure 10** demonstrates the steps described above.

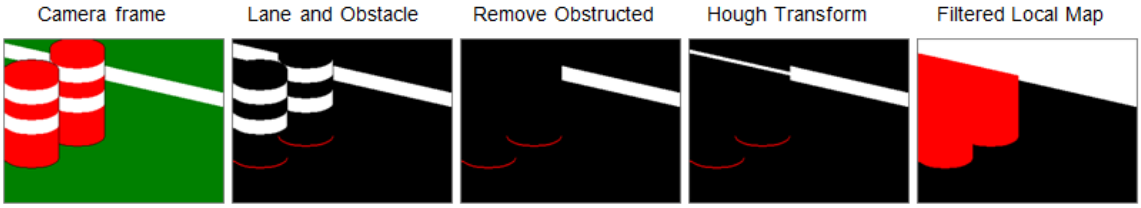

Figure 10: Image Processing Steps

5.5.4 GPU Processing

The results from previous IGVC competitions indicated that the required image processing on a laptop CPU was too strenuous for the hardware. To solve this issue, it was found that the GPU could be used as an alternative for some of the processing. The GPU is exceptional for manipulating image data and therefore is an excellent candidate for fast image processing. XNA was used to communicate with the GPU. Normally, XNA is used for video game development, but it is C# based and was easy to integrate with the existing codebase. Team Viper converted several image filtering algorithms, such as the Brightest Pixel filter, to run on the GPU to achieve dramatic performance improvements.

5.5.5 Hough Transform

A Hough Transform helps Viper II identify lanes in the local map, but it is primarily used for filling in a lane when only a partial (dashed) one exists. See **Figure 11** for an example of the Hough Transform filling in missing data. This transform is applied twice, because the local map is split vertically. The splitting of the local map forces the detection of a lane on both sides, and this is critical in cases where both of the lanes exist on the local map at once. Since a single image does not guarantee the Hough Transform will recognize both lanes, each image must be evaluated for missing data. This lack of recognition becomes extremely apparent when the second lane's intensity is less dominant.

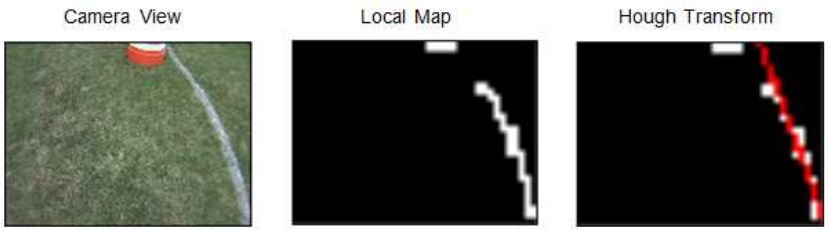

Figure 11: Hough Transform applied to dashed lane

5.5.6 Blob Detection

Blob detection and filtering allows for the elimination of noise pixels from the camera image. As the image is scanned, each pixel is examined to determine its color. Once this process is finished, each contiguous region of color in the image will have a corresponding blob object, and these blobs are useful for several reasons. First, they allow identification of all regions of color and this can help the detection algorithm filter out blobs that are too small (and thus are just noise). Secondly, blobs of "non-obstacle" space can be identified, and this ensures that the detection algorithm selects the goal node that is in the same region of the robot. Thus, the robot will not cross a lane to get to the local goal location.

5.5.7 Lane Following and Obstacle Avoidance

Viper II uses a goal node selection algorithm to find the best path through the obstacle course. This algorithm first finds the largest gap in the image, and then plots a goal node in the center of this gap. If the largest gap on the map is smaller than the Viper II, it retreats and searches for a path that it can fit through. Once the goal node is found, Viper II starts to move in the direction of this node until it processes new image data. Additionally, a new algorithm was added this year to avoid turning in the wrong direction when facing a lane on one side and obstacles on the other side of the robot.

5.5.8 Goal Node Selection

Goal node selection is an important aspect of the autonomous navigation challenge, and it relies on the local map. Once the local map is built, it is then transformed into an image, and a goal location for the robot's heading must be found. This is handled by finding the largest gap between obstacles and lanes. Ideally, this gap is at the top of the image and as far away as possible from the robot. However, if a lane bisects the image, then the gap is calculated from either the left or right side of the image. The images below demonstrate the goal node selection algorithm. The goal location is the orange dot in the images shown in **Figure 12**.

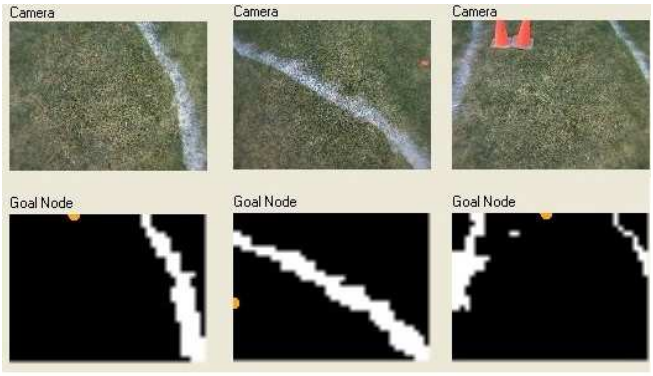

Figure 12: Goal Node Selection Example

To improve decision making when selecting a goal node, the local map is analyzed twice; the first check includes only lane information and the second check includes both lane and obstacle information. Analyzing only the lane information allows the robot to always conclude that it should turn away from the lane. This decision is followed even if there's an obstacle in the direction opposite of the lane. Even if turning causes the Viper II to face an obstacle, it will continue turning away from the lane until it sees a clear path. **Figure 13** shows the two goal node checks, and **Figure 14** shows the decision process to turn away from the lane (blue).

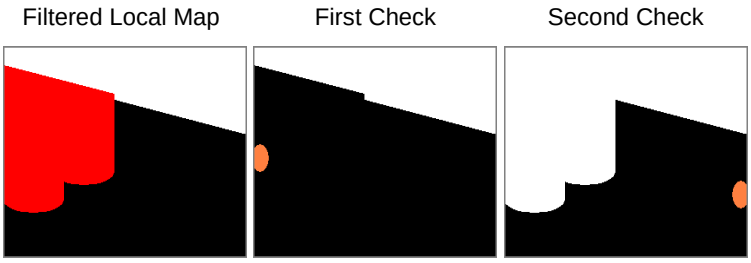

Figure 13: Two Goal Node Checks

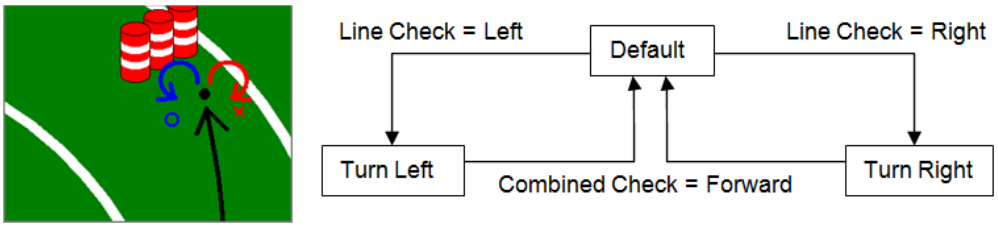

Figure 14: Direction Decision and State Diagram to Escape from a Trap

5.6 Navigation Challenge Details

The primary algorithm of the navigation challenge uses the GPS, compass, and SICK laser range finder sensors in conjunction with dead-reckoning information available from the motor controller (derived from optical encoders on the motor shafts) to move Viper II through the course to the various GPS checkpoints. The GPS waypoint locations are mapped onto a World Map. As Viper II moves through the course and discovers obstacles using the SICK, the location of the obstacles is also mapped. This allows Viper II to adjust its course and navigate around the obstacles to the next checkpoint. Additionally, this year's robot is also using a Fuzzy Controller to smooth out the robots movement through the course.

5.6.1 Waypoints Scheduling

To select the quickest route between waypoints, Team Viper classifies the problem as a Traveling Salesman Problem. For simplicity, Viper II uses a Greedy Algorithm to determine the visiting order. The obstacles detected with the SICK may change the path from the robots' position to the waypoints. Therefore, the robot will reschedule its visiting order when each sub-destination is reached.

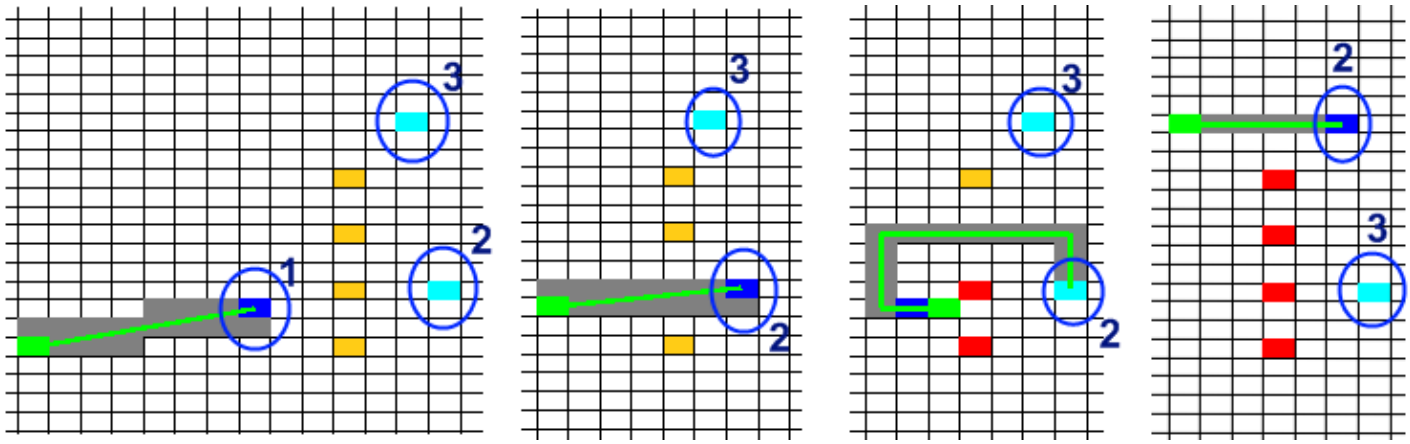

Figure 15: Waypoint scheduler

Light blue boxes represent waypoints to be visited, and dark blue is the current goal waypoint. Red boxes are detected obstacles and orange ones are not yet known.

Figure 15 demonstrates how the Greedy Waypoint Scheduling algorithm allows the Viper II to dynamically adjust its path based on newly discovered obstacle data. After the robot visits the 1st waypoint and starts heading towards the 2nd one, it discovers new obstacles. The obstacle data is mapped onto the World Map. At this point, Viper II still tries to go to Waypoint #2, but finds even more obstacles blocking its path. Finally, as it attempts to avoid obstacles, Viper II finds that it is now closer to the waypoint at the top of the map, so it schedules this waypoint as the second one to visit.

5.6.2. Fuzzy Controller

A fuzzy logic controller was created to help fix the problem with oscillation in the Viper's movement while performing obstacle avoidance in the Navigation Challenge. The controller takes the robot's current and desired direction headings as inputs, and outputs an adjusted heading. The fuzzy logic rules in the controller limit oscillation in the robot's movements by smoothing out the otherwise drastic changes in direction. A smoother path through the course helps Viper II get to all the waypoints faster.

5.7 JAUS Challenge Details

Viper's JAUS implementation has been completely redesigned for IGVC 2009. The intent was to leverage the modularity aspect of JAUS to establish a design that is easily expanded for future requirements. The implementation is comprised of a network of JAUS components working together to fulfill the JAUS Challenge requirements. A Node Manager routes all JAUS message traffic between JAUS components on Viper. All extra-nodal JAUS messages to and from Viper are translated to the Transport Layer and exchanged by a Communicator component. Figure 15 depicts the JAUS network in blue and the associated Viper modules in red.

5.7.1 Process for Learning JAUS

Development of the JAUS network benefitted greatly from experience gained preparing for and participating in last year's IGVC competition. Although the Reference Architecture was well understood by the team, a considerable amount of time was spent studying and trying to understand the new SAE specifications. The team was provided access to the JAUS Validation Tool (JVT) at http://www.usaric.org/JVT. This tool was essential for validating the team's JAUS implementation.

5.7.2 JAUS Integration into the Design

The Viper JAUS components are written in Java while the Sensor Fusion and mission modules are written in C#. A UDP inter-process link provides the interface between the JAUS components and the rest of the Viper software. This design strategy was inspired by the modular architecture of JAUS. Once the UDP link was defined, Viper's JAUS implementation was developed and tested independently of the rest of the software. **Figure 16** demonstrates the JAUS component and how it communicates to the Viper II's Sensor Fusion module.

5.7.3 Challenges Encountered

The principal challenges encountered implementing JAUS center around the new SAE requirements. The SAE specifications introduced by the 2009 IGVC Rules are vastly more complex than the Reference Architecture. The specifications establish many features new to JAUS, including modified and expanded transport requirements, a multi-layered architecture, and service oriented definitions. Specification availability, completeness, and accuracy were also challenges.

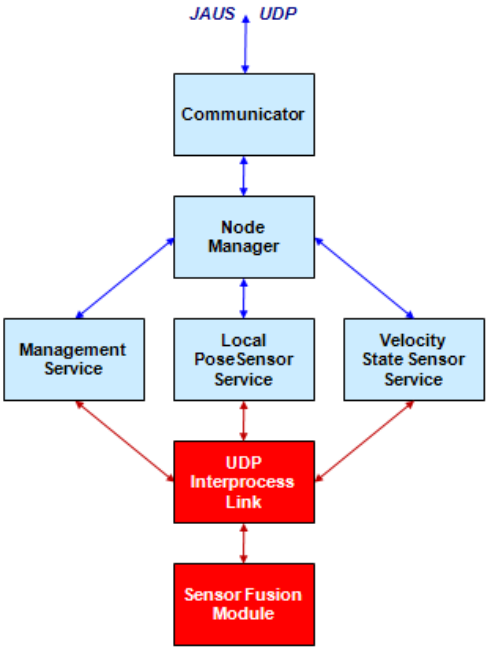

Figure 16: JAUS high level design.

6. Performance Analysis and Estimate

6.1 Safety

Viper II safety features include a manual and remote E-Stop, failsafe brakes, and a capped maximum speed of 5 mph.

6.2 Robot Performance

Vehicle performance was estimated based on the design parameters. These same attributes were measured after integration as shown in **Table 2**.

Attribute	Design Prediction
Maximum Speed	4.2 mph
Climbing Ability	2.5 inch curb
Nominal Power Consumption	500 watts
Battery Operating Time	4 hours
Waypoint accuracy (with Omnistar)	10-20 centimeters

Table 2: Performance Analysis

6.3 Reliability

Viper II utilizes dual modular power sources and a thoroughly tested drive train design that performed well in the 2008 competition. Additionally, the robot uses a rigid aluminum frame with extensive weather-proofing. Furthermore, extensive testing has been done for both the hardware and the software components. Specifically, regression and system testing was done on each module to ensure proper functionality and integration with the hardware components.

6.4 Vehicle Cost Summary

Table 3 summarizes the total material cost for the Viper II vehicle.

Component	Retail Total Cost	Team Cost
CyberPower Laptop	$3,342	$3,342
Sick LMS 291-S05 LMS	$7,000	$0
NovaTel ProPak-LB DGPS & GPS-600-LB Antenna	$4,567	$2,700
24 V NPC-T74 Motors (2)	$648	$648
Panasonic PV-GS500 digital camcorder	$800	$0
Miscellaneous Electrical Hardware	$870	$870
PNI TCM2-20 digital compass/inclinometer	$769	$0
Roboteq AX3500 Dual Channel Motor Controller	$395	$0
Main Battery 12 Volt 50 Ah AGM	$323	$323
Promariner Promite 5/5 Dual 12V Battery Charger	$110	$110
DC-to-DC Converter	$118	$118
EDFSS Hardware Components	$2,308	$780
Hollow Shaft Optical Encoder Kit (2)	$130	$130
Chassis Materials	$320	$320
Miscellaneous Hardware (nuts, bolts, etc…)	$200	$200
14" Tire & Wheel Assembly (2)	$186	$186
Rear 300 Lb Capacity Caster Wheel	$22	$22
Total	$20,931	$6,357

Table 3: Vehicle Cost Summary

7. Conclusion

Viper II continues the LTU tradition of innovation and continuous improvement. Viper II features a distinctive EDFSS system with HMI interface, as well as a backup electrical system, stable camera mount and shock absorbers, and a modular, innovative software design. All software and hardware components are easily swappable and can be upgraded in future designs. Viper II's unique design and state-of-the-art software technology set it apart from other competitors in the 17th annual Intelligent Ground Vehicle Competition.

8. References

[Ward 1994] Ward, A.C., Sobek, III, D.K., "Toyota and Set-Based Concurrent Engineering," 1994 ASME Design Theory and Methodology Conference Proceedings, Minneapolis, MN

[**http://www.jauswg.org**] The Joint Architecture for Unmanned Systems. "Reference Architecture Specification. Volume 2," 2004.

Poppendieck, M., Poppendieck, T. "Lean Software Development: An Agile Toolkit". Addison-Wesley Professional (May 18, 2003)

[**http://en.wikipedia.org/wiki/Hough_transform**] Hough Transform. March 2009.

Nilsson, N. J. , "Artificial Intelligence: A New Synthesis". Morgan Kaufmann Publishers, 1998, San Fransisco, CA.

Lawrence Technological University

Viper II

IGVC 2009 Autonomous Vehicle

Team Members

Gary Givental, Philip Munie, Shawn Ellison, Brandon Bell, Ze Cheng, Taiga Sato, Bryan Koroncey, Nathan Lucas

Faculty Advisor Statement[1]

I, Dr. CJ Chung of the Department of Math and Computer Science at Lawrence Technological University, certify that the design and development on Viper II has been significant and each team member has earned credit hours for their work.

_____ _____

Dr. CJ Chung (chung@ltu.edu, 248-204-3504) Date

[1] Co-advisor: Dr. Lisa Anneberg, Department of ECE

Table of Contents

1. Introduction

For the 2009 Intelligent Ground Vehicle Competition (IGVC), Lawrence Technological University (LTU) presents Viper II, the product of a collaborative effort among a diverse and multidisciplinary group of LTU engineering and computer science students. Viper II is an enhanced robot, with cutting-edge hardware and software updated for 2009. It represents Lawrence Tech's latest venture into developing a safe, reliable, and cost-effective autonomous vehicles that are rich in progressive technologies.

2. Innovations

2.1 Hardware Platform

2.1.1 Mechanical System

Viper II is a three wheel vehicle with front differential drive steering, a rear caster wheel, and an aluminum chassis that are all encased in a custom fiberglass mold. This design provides the most stability and maneuverability, and it allows for a zero-degree turn radius. Furthermore, the vehicle's front suspension system helps stabilize the drive-train and minimizes the shaking of the camera pole.

2.1.2 Electrical System

Viper II uses a manual electrical system that is controlled via a single electric box. There's also hardware in place (a Programmable Logic Controller (PLC) coupled with a Human Machine Interface (HMI) display) that extends the manual system to create the Electronic Diagnostic Fail-Safe System (EDFSS).

2.1.3 Software

The software for Viper II is written in the C# programming language using Visual Studio 2008 and Microsoft XNA. XNA is a set of tools with a managed runtime environment provided by Microsoft that facilitates computer game development, and it allows Viper II to quickly perform image processing on the GPU. Additionally, Viper II includes a Sensor Fusion module to bring sensor information together into one component and an easily pluggable motor control interface for its movement. Moreover, custom Local and World map software help Viper II keep track of lane, obstacle, and GPS waypoint information, and they allow it to be less reactive to the immediate environment. Furthermore, Viper II utilizes a GPS Breadcrumb system to keep track of recently visited locations, and to detect when it is heading in an incorrect direction on the course. Lastly, Viper II software is JAUS level 4 compliant.

3. Design Process

3.1 Collaborative Team Organization

2009 Team members are:

Gary Givental, MSCS – Team Captain	Philip Munie, MSCS – Autonomous Lead
Brandon Bell, MSCS – Hardware Integration Lead	Shawn Ellison, MSCS – Navigation Lead
Nathan Lucas, MSCS – JAUS Lead	Taiga Sato, BSCS
Bryan Koroncey, BSCS	Ze Cheng, BSCS

Since a large team required more communication, team members used an online discussion forum, Subversion code repository, and email to communicate effectively. Weekly meetings were scheduled to allow for status reports and to help resolve emerging issues.

3.2 Project Planning Process

The development for this year's IGVC effectively started in October 2008. **Figure 1** describes the project plan followed by the team. For this year's competition, the team met every week to go through the design plan and to perform integration testing. Since an agile, iterative development process was used, the design emerged over time as different options for the software were explored. This project plan acted as a guideline to keep the team on track, and to make sure core timelines were met.

Task	Due Date
Initial Brainstorming	2/1/2009
File Repository Setup	2/1/2009
High Level Software Design	3/1/2009
High Level Hardware Update Design	3/1/2009
Software Development	5/1/2009
E-Stop Updates	5/15/2009
System Testing	5/15/2009
Written Report	5/18/2009
Oral Presentation	6/7/2009

Figure 1: Project Tasks

3.3 Development Process

The team used agile development and set-based, concurrent methodologies for the software design in this year's competition. The set-based, concurrent development allowed the team to investigate several options for software modules and algorithms, and to find the best solution to design issues. Correspondingly, the agile development approach allowed the team to avoid the well-known pitfalls of the linear "waterfall" style approach. These techniques enabled an emergent, iterative methodology that proved to be the perfect fit and allowed for the functionality and features of the software and hardware to evolve through continuous testing and integration. The team scheduled regular weekly meetings to present status reports, to discuss progress, and to indicate any road-blocks encountered by team members during development. Frequently, students teamed up to work on a particular problem in hardware or software to get it resolved quicker, and to leverage the "pair-programming" concept.

3.4 Testing Process

A test track was setup on LTU campus, which was complete with various kinds of obstacles and curves in the lanes to approximate the challenges of the real competition. This allowed team members to collect real world data and to discover failures in software logic.

3.4.1 Hardware Testing

Viper II's electrical and mechanical systems were tested thoroughly after Viper was showcased to TARDEC, and extensive field testing was performed once the robot was re-assembled. While performing field testing, the team discovered several issues that needed to be addressed. The robot's batteries drained quickly, so a generator was obtained to keep them charged. Additionally, the GPS signal proved to be inaccurate, so the team subscribed to the Omnistar HP service to get 10-centimiter precision.

3.4.2 Software Testing

The team performed extensive unit testing, systems testing, and integration testing of the code throughout development. As new functionality was developed, it was demonstrated to the rest of the team, and, when possible, tested on the robot. Test runs were performed regularly on the test track at LTU.

3.4.3 System Integration

System integration testing was executed as quickly as possible once the development of the core components was complete. When new modules were available for general testing, they were immediately integrated into the overall system and used by all team members for testing. This modular design of the software components made it easy for new functionality to be plugged into the overall architecture. The team extensively tested the integration between the software and all the sensor hardware, as well as the motor control.

4. Hardware Design

4.1 Robot Structure

4.1.1 Chassis

Viper II is a three wheel vehicle with front differential drive steering and a rear caster wheel. This design allows for zero-turn radius maneuverability and simplified motion control. The chassis has a minimum of 4 inches of ground clearance, and this allows the robot to climb hills and ramps with an approach angle of 23 degrees. The 3-D CAD model shown in the follow **Figure 2** illustrates the design of the chassis using the ¾ inch 6063 aluminum tubing, which gives the robot the capability of carrying all of the necessary components, as well as the designated payload required for the competition. The chassis design enables Viper II to maintain its structural integrity throughout the rugged off-road course.

4.1.2 Suspension

Viper II is designed with an independent suspension that absorbs impact in the wheels from the off-road terrain. This design provides ample stability to the camera, thus allowing for a smooth video image feed. The suspension model in **Figure 3** shows the coil spring over a tube style shock that is mounted to independent control arms. This suspension style gives the robot 1 inch of travel in the shocks, and it greatly reduces the twist and roll of the chassis and components. Furthermore, the shocks are rated for 350 lbs per inch and have an adjustable rebound feature.

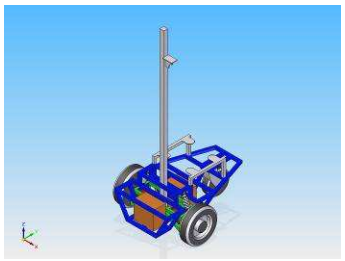

Figure 2: Chassis Model

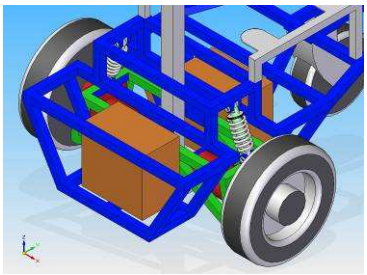

Figure 3: Front Suspension

4.1.3 Drive Train

Viper II's design uses two 24 volt, permanent magnet type model NPC-T74 high torque motors with a 20:1 gear ratio that gives the robot the ability to generate up to 1480 in-lbs (123.3 ft-lbs) of torque and allows for a maximum speed of 10.4

MPH (at 248 RPMs). Thus, these motors give Viper II plenty of power, and they eliminate the fear of premature motor failure. They ensure that Viper II can easily navigate through rough terrain, and go up a 15 degree incline.

4.1.4 Body

The body that is covering and protecting Viper II is fabricated from fiberglass, and its sleek design with smooth flowing curves and lines gives the feel of its namesake. It is designed to give easy access to the internal components through an easily removable top section, and it is fitted with electric fans to ensure that heat is removed from inside the robot. Furthermore, a tray that slides out of the right side of the fiberglass body allows access to Viper II's internal laptop computer, and it can be used without removing the top section.

4.2 Motors and Electronics

4.2.1 Motor Control

Motor control is facilitated by a Roboteq AX3500 60A/channel controller with a C# serial port interface. The AX3500 provides velocity feedback measurements through optical encoders that are mounted on the motor shafts. Additionally, the E-Stop is wired to the controller's main drive power lines via a mechanical relay that can be triggered by the top-mounted kill switch or by the wireless control system.

4.2.2 Sensors

In keeping with the theme of modularity, all of Viper II sensors (and controls) utilize RS-232 or USB 2.0 standard interfaces. **Table 1** summarizes the sensors used in Viper II.

Sensor	Function
Optical Encoders	Motor shaft position feedback for servo control
NovaTel ProPack-LB DGPS Unit	Global positioning
Digital Compass/Inclinometer	Heading, roll, and pitch information
High Dynamic Range DV Camera	Capture field image stream
Sick Laser Measurement Sensor	Provides a 180 degree polar ranging array

Table 1: Vehicle Sensor Summary

4.2.3 E-stop

Viper II is equipped with both manual and wireless (RF) remote emergency stop (E-Stop) capabilities. The Excalibur RS-310 Remote and Start Keyless Entry manufactured by Omega Research is used as the wireless E-Stop component, and its door lock function is integrated into Viper II's E-stop system. When this E-Stop is activated, it removes power from the drive train and engages both of the front wheel failsafe electromagnetic brakes.

4.2.4 Vehicle Alert System (JAUS Horn and Lights Hardware)

Viper II includes an electrical module that can switch relatively large electrical loads (using relays) to the horn and light systems via commands sent over a USB interface. It is utilized by the vehicle's alert system, and it is activated by Viper II's JAUS software.

4.3 Electrical System

4.3.1 Power System

Viper II has two parallel electrical systems: a logic control system and a manual control system. The logic control system consists of a Programmable Logic Controller (PLC) and a Human Machine Interface (HMI), and these components make up the Electronic Diagnostic Fail-Safe System (EDFSS). The manual system consists of toggle switches and fuses that allow the team to physically turn on and off each electrical component. The operating mode can each be selected by a keyed selector switch found on the main system control board.

4.3.2 Electronic Diagnostic Fail-Safe System - EDFSS

The EDFSS provides Viper II with a nearly endless range of abilities for its current design and future design modifications. It makes the use of an external touch screen display that can be programmed to control each component's power and to display status information.

4.3.2.1 External status indicator lights

During previous competitions, the team realized that easily visible power indicators for key component were needed during testing. The EDFSS solves this problem by controlling three indicator lights: the green light specifies all systems are ready, the yellow light signifies that Viper II is operational, but not all systems are responding, and the red light indicates that the system is not operational. Additional indicator lights are also available next to each manual switch on the electrical box. This functionality allows the IGVC team to quickly identify power issues during testing.

4.3.2.2 Manual Control

The manual system is a basic on/off component control. It acts as a backup electrical system if there are problems with the EDFSS system.

4.3.3 Power Source

Viper II is powered by two 12V 50 Ah AGM batteries connected in series to provide a 24V electrical system. It uses a 24/12V DC/DC converter for the onboard 12V systems and an onboard charger with 110VAC interface for restoring battery power. **Figure 4** illustrates how power for the left and right motors is fed directly into the motor controller to maximize the motor power. However, the control power for the motor controller is regulated by the E-Stop and electrical control box.

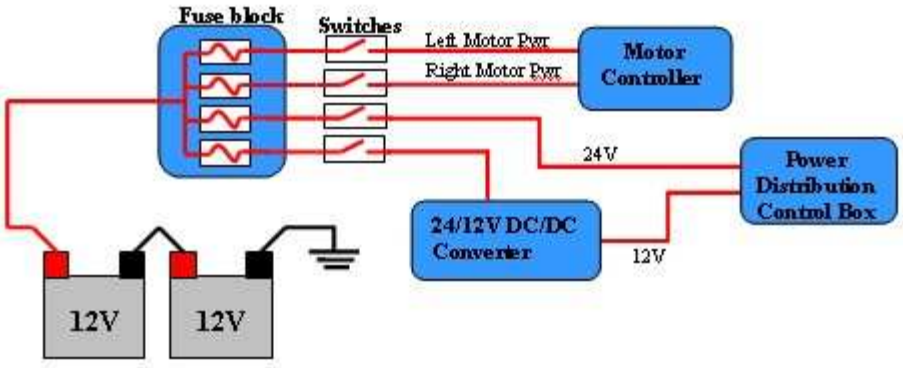

Figure 4: Power Source Schematic

4.3.3 Power Distribution

Viper II is equipped with an all-in-one electrical control box. This power distribution control box contains the systems power distribution and emergency stop printed-circuit board (PCB), JAUS PCB hardware, wireless emergency stop PCB hardware, power control switches for each component, and the main system selector switch that controls the manual and EDFSS operation modes. It is designed to be self contained and removable from Viper II, and it also offers additional testing and diagnostic abilities for system voltage and current ampere measurements.

The two main power connectors are automotive grade EDAC panel connectors, and they allow for an easy connection point for two wire harnesses, which route power to and from each hardware item and the EDFSS block. The power and communications control system schematic for Viper II vehicle is shown in **Figure 5**.

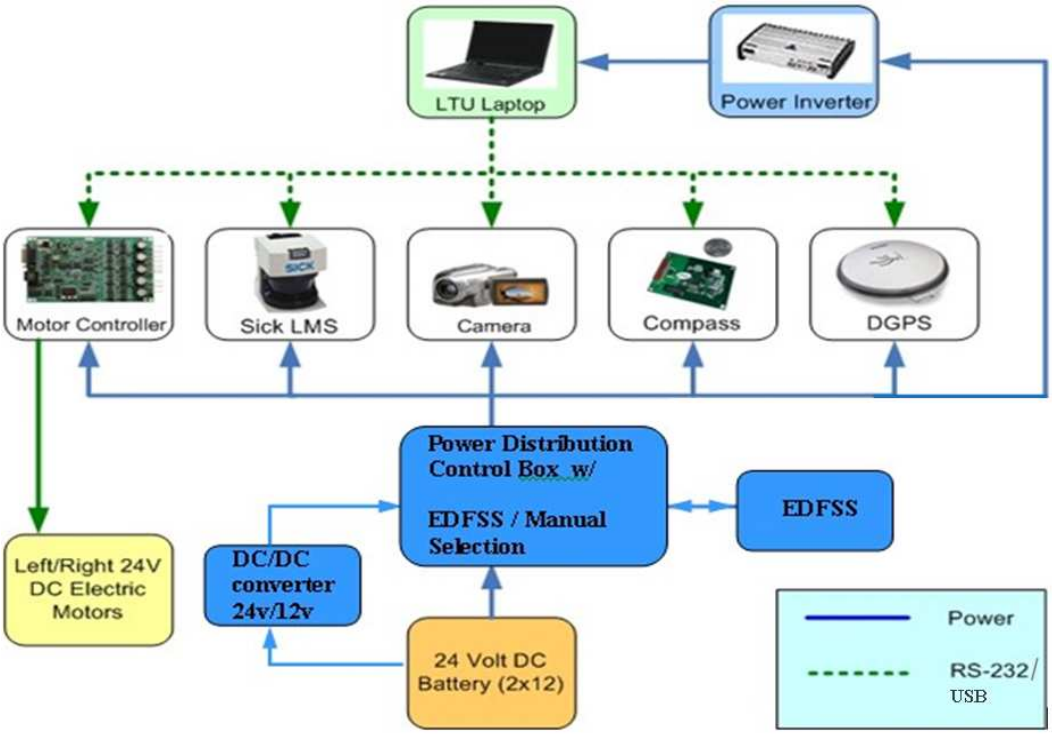

Figure 5: Power and Communications Control System Schematic

5. Software Design

5.1 Software Strategy

C# , XNA, and the Visual Studio 2008 IDE were chosen for all software development on the robot this year. C# is a powerful object oriented language, and Microsoft XNA is a video gaming toolset which allowed Viper II to utilize the processing power of GPU for image analysis. The Visual Studio IDE allowed for rapid application development with easy to build visual interfaces. **Figure 6** shows the high level software architecture design.

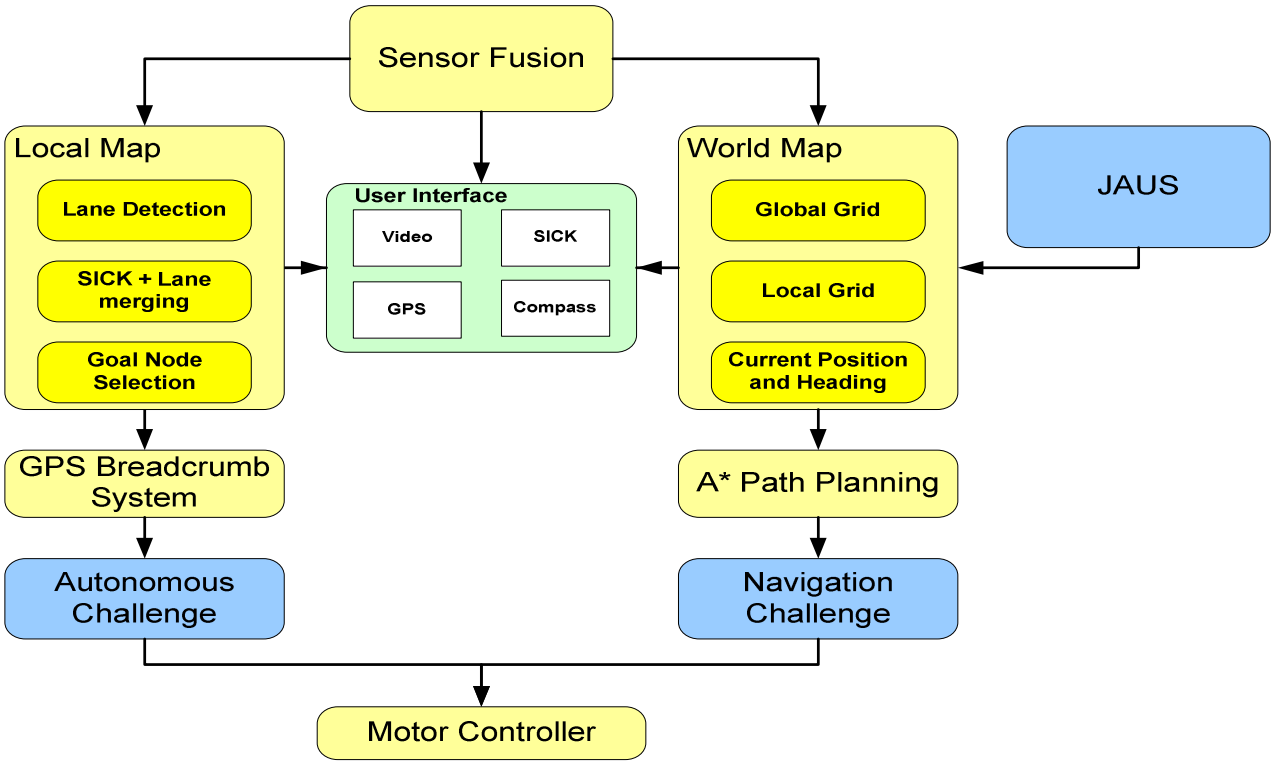

Figure 6: High Level Software Design

5.2 Sensor Fusion

For this year's competition, a Sensor Fusion Module shown in **Figure 7** was developed to collect data from all the hardware sensors, and to present this data to all subscribers in a consistent manner. This allows subscribers to access sensor data without knowing the details of hardware layer implementation.

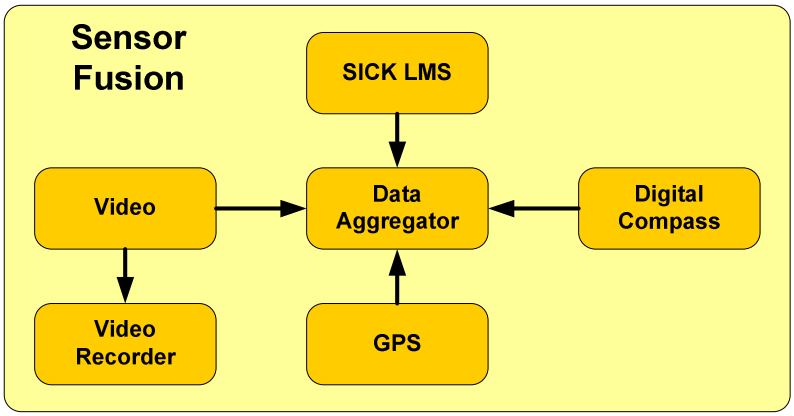

Figure 7: Sensor Fusion

5.3 World Map and Path Finding

The world map plots a geographic region to a Cartesian coordinate plane. The total size of the Cartesian map is determined by the size of the area to be represented and the desired resolution. Generally, a resolution of ½ meter has proven to work well during testing. Each point in the geographic region can be mapped to a cell in the Cartesian map, and vice versa, through the use of a conversion object. In this way, a simple A* path finding algorithm can be used to find the shortest path between any two given points.

5.4 Motor Control Module

The motor controller module was designed to be easily plugged into any of the team's projects. Once initialized, this module can accept simple operation commands for speed and direction, which it then translates into the ASCII communication strings that are sent to the motor controller via a serial connection. This module allows team members to have simple access to the hardware without knowledge of the motor controller's underlying communication protocols.

5.5 Autonomous Challenge Details

The main algorithm for the autonomous challenge consists of using computer vision to identify lane information and SICK data output to detect obstacles. To accomplish this, the camera is first calibrated so that the vision processing can map lanes onto a local map alongside the SICK data, and then a goal node selector algorithm is used to find the appropriate heading. Additionally, a GPS Breadcrumb utility, which uses GPS location information and compass headings to store a list of recently visited locations, is used to detect if the robot has become turned around. This helps the robot avoid going backwards on the course.

5.5.1 Camera Calibration

The camera calibration is responsible for synchronizing the lane data with the SICK obstacle data. It works by calculating the distance of each pixel in the vision's captured image, and it uses the SICK as the reference point. It is accomplished by measuring the distance of predefined positions on the captured image and interpolating the remaining positions on the image. A snapshot of the tool for calibration is shown in **Figure 8**.

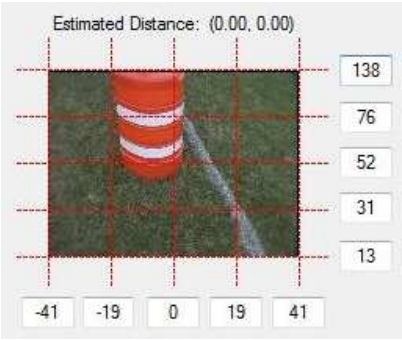

Figure 8: Camera Calibration Tool

5.5.2 SICK Data Gathering

As the SICK data is collected, it is mapped onto a local map, combined with the filtered camera image. Because of the calibration previously, the data can simply be placed in the local map without any extra processing. See **Figure 9**.

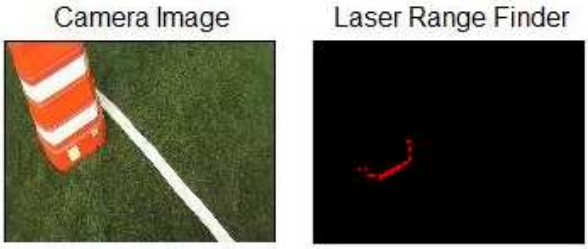

Figure 9: Example of mapping SICK data to the Local Map

5.5.3 Image Processing

For this year's competition, Viper II continues to use a local map approach for combining the vision information with obstacle data. First, SICK obstacle data is added into the local map as red pixels. Then, the camera image is captured and converted to black and white using a brightest pixel filter. This filter runs on the GPU and chooses the brightest pixel on each horizontal line. Then blob detection is used to filter out noise pixels that are not considered to be lanes. Once noise has been removed, the filtered camera image is added to the SICK data and all of the white pixels that are vertically above the red pixels are repainted as black. A Hough Transform filter is then applied to find any partial lanes in the image and fill them in. Finally, a bleeding filter is used to color any pixels vertically above red pixels as red, and above white pixels as white. Thus, the local map represents both the lane information in white pixels and the SICK obstacle data in red pixels. **Figure 10** demonstrates the steps described above.

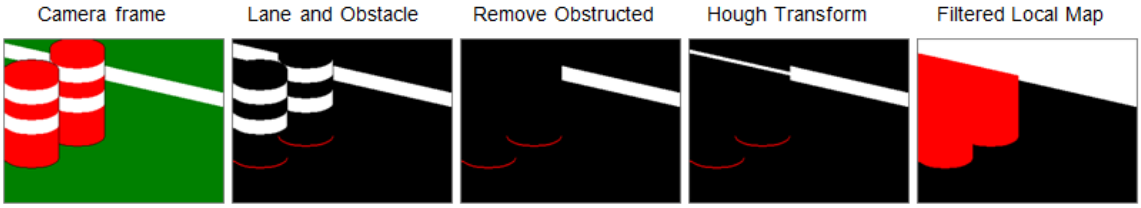

Figure 10: Image Processing Steps

5.5.4 GPU Processing

The results from previous IGVC competitions indicated that the required image processing on a laptop CPU was too strenuous for the hardware. To solve this issue, it was found that the GPU could be used as an alternative for some of the processing. The GPU is exceptional for manipulating image data and therefore is an excellent candidate for fast image processing. XNA was used to communicate with the GPU. Normally, XNA is used for video game development, but it is C# based and was easy to integrate with the existing codebase. Team Viper converted several image filtering algorithms, such as the Brightest Pixel filter, to run on the GPU to achieve dramatic performance improvements.

5.5.5 Hough Transform

A Hough Transform helps Viper II identify lanes in the local map, but it is primarily used for filling in a lane when only a partial (dashed) one exists. See **Figure 11** for an example of the Hough Transform filling in missing data. This transform is applied twice, because the local map is split vertically. The splitting of the local map forces the detection of a lane on both sides, and this is critical in cases where both of the lanes exist on the local map at once. Since a single image does not guarantee the Hough Transform will recognize both lanes, each image must be evaluated for missing data. This lack of recognition becomes extremely apparent when the second lane's intensity is less dominant.

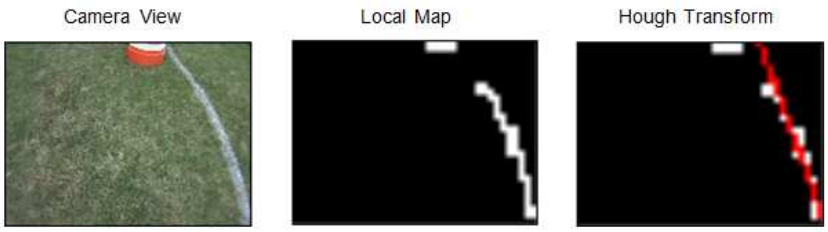

Figure 11: Hough Transform applied to dashed lane

5.5.6 Blob Detection

Blob detection and filtering allows for the elimination of noise pixels from the camera image. As the image is scanned, each pixel is examined to determine its color. Once this process is finished, each contiguous region of color in the image will have a corresponding blob object, and these blobs are useful for several reasons. First, they allow identification of all regions of color and this can help the detection algorithm filter out blobs that are too small (and thus are just noise). Secondly, blobs of "non-obstacle" space can be identified, and this ensures that the detection algorithm selects the goal node that is in the same region of the robot. Thus, the robot will not cross a lane to get to the local goal location.

5.5.7 Lane Following and Obstacle Avoidance

Viper II uses a goal node selection algorithm to find the best path through the obstacle course. This algorithm first finds the largest gap in the image, and then plots a goal node in the center of this gap. If the largest gap on the map is smaller than the Viper II, it retreats and searches for a path that it can fit through. Once the goal node is found, Viper II starts to move in the direction of this node until it processes new image data. Additionally, a new algorithm was added this year to avoid turning in the wrong direction when facing a lane on one side and obstacles on the other side of the robot.

5.5.8 Goal Node Selection

Goal node selection is an important aspect of the autonomous navigation challenge, and it relies on the local map. Once the local map is built, it is then transformed into an image, and a goal location for the robot's heading must be found. This is handled by finding the largest gap between obstacles and lanes. Ideally, this gap is at the top of the image and as far away as possible from the robot. However, if a lane bisects the image, then the gap is calculated from either the left or right side of the image. The images below demonstrate the goal node selection algorithm. The goal location is the orange dot in the images shown in **Figure 12**.

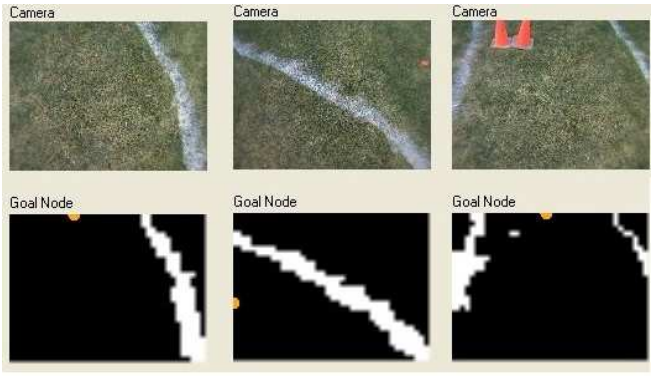

Figure 12: Goal Node Selection Example

To improve decision making when selecting a goal node, the local map is analyzed twice; the first check includes only lane information and the second check includes both lane and obstacle information. Analyzing only the lane information allows the robot to always conclude that it should turn away from the lane. This decision is followed even if there's an obstacle in the direction opposite of the lane. Even if turning causes the Viper II to face an obstacle, it will continue turning away from the lane until it sees a clear path. **Figure 13** shows the two goal node checks, and **Figure 14** shows the decision process to turn away from the lane (blue).

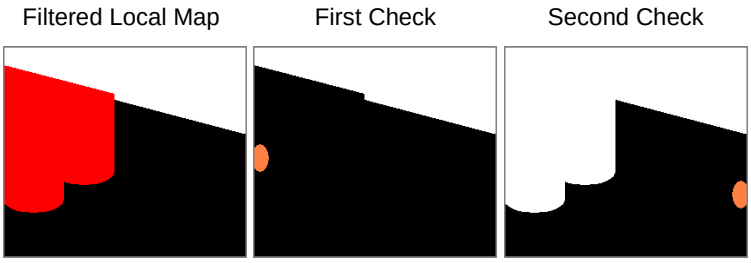

Figure 13: Two Goal Node Checks

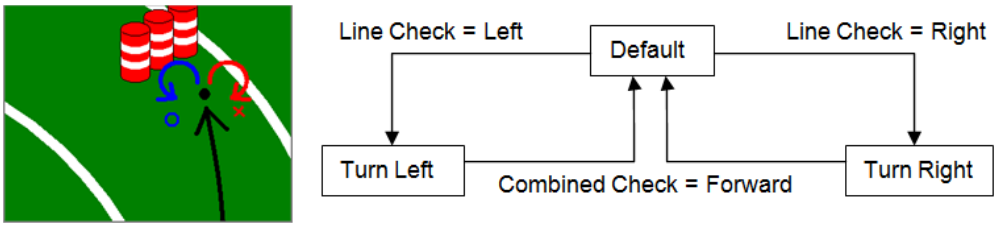

Figure 14: Direction Decision and State Diagram to Escape from a Trap

5.6 Navigation Challenge Details

The primary algorithm of the navigation challenge uses the GPS, compass, and SICK laser range finder sensors in conjunction with dead-reckoning information available from the motor controller (derived from optical encoders on the motor shafts) to move Viper II through the course to the various GPS checkpoints. The GPS waypoint locations are mapped onto a World Map. As Viper II moves through the course and discovers obstacles using the SICK, the location of the obstacles is also mapped. This allows Viper II to adjust its course and navigate around the obstacles to the next checkpoint. Additionally, this year's robot is also using a Fuzzy Controller to smooth out the robots movement through the course.

5.6.1 Waypoints Scheduling

To select the quickest route between waypoints, Team Viper classifies the problem as a Traveling Salesman Problem. For simplicity, Viper II uses a Greedy Algorithm to determine the visiting order. The obstacles detected with the SICK may change the path from the robots' position to the waypoints. Therefore, the robot will reschedule its visiting order when each sub-destination is reached.

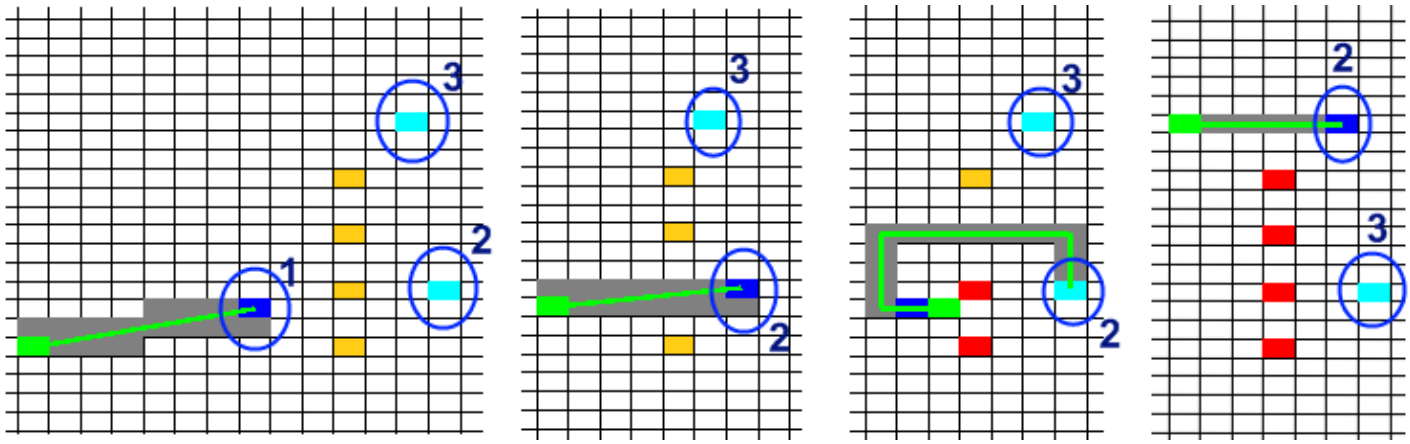

Figure 15: Waypoint scheduler

Light blue boxes represent waypoints to be visited, and dark blue is the current goal waypoint. Red boxes are detected obstacles and orange ones are not yet known.

Figure 15 demonstrates how the Greedy Waypoint Scheduling algorithm allows the Viper II to dynamically adjust its path based on newly discovered obstacle data. After the robot visits the 1st waypoint and starts heading towards the 2nd one, it discovers new obstacles. The obstacle data is mapped onto the World Map. At this point, Viper II still tries to go to Waypoint #2, but finds even more obstacles blocking its path. Finally, as it attempts to avoid obstacles, Viper II finds that it is now closer to the waypoint at the top of the map, so it schedules this waypoint as the second one to visit.

5.6.2. Fuzzy Controller

A fuzzy logic controller was created to help fix the problem with oscillation in the Viper's movement while performing obstacle avoidance in the Navigation Challenge. The controller takes the robot's current and desired direction headings as inputs, and outputs an adjusted heading. The fuzzy logic rules in the controller limit oscillation in the robot's movements by smoothing out the otherwise drastic changes in direction. A smoother path through the course helps Viper II get to all the waypoints faster.

5.7 JAUS Challenge Details

Viper's JAUS implementation has been completely redesigned for IGVC 2009. The intent was to leverage the modularity aspect of JAUS to establish a design that is easily expanded for future requirements. The implementation is comprised of a network of JAUS components working together to fulfill the JAUS Challenge requirements. A Node Manager routes all JAUS message traffic between JAUS components on Viper. All extra-nodal JAUS messages to and from Viper are translated to the Transport Layer and exchanged by a Communicator component. Figure 15 depicts the JAUS network in blue and the associated Viper modules in red.

5.7.1 Process for Learning JAUS

Development of the JAUS network benefitted greatly from experience gained preparing for and participating in last year's IGVC competition. Although the Reference Architecture was well understood by the team, a considerable amount of time was spent studying and trying to understand the new SAE specifications. The team was provided access to the JAUS Validation Tool (JVT) at http://www.usaric.org/JVT. This tool was essential for validating the team's JAUS implementation.

5.7.2 JAUS Integration into the Design

The Viper JAUS components are written in Java while the Sensor Fusion and mission modules are written in C#. A UDP inter-process link provides the interface between the JAUS components and the rest of the Viper software. This design strategy was inspired by the modular architecture of JAUS. Once the UDP link was defined, Viper's JAUS implementation was developed and tested independently of the rest of the software. **Figure 16** demonstrates the JAUS component and how it communicates to the Viper II's Sensor Fusion module.

5.7.3 Challenges Encountered

The principal challenges encountered implementing JAUS center around the new SAE requirements. The SAE specifications introduced by the 2009 IGVC Rules are vastly more complex than the Reference Architecture. The specifications establish many features new to JAUS, including modified and expanded transport requirements, a multi-layered architecture, and service oriented definitions. Specification availability, completeness, and accuracy were also challenges.

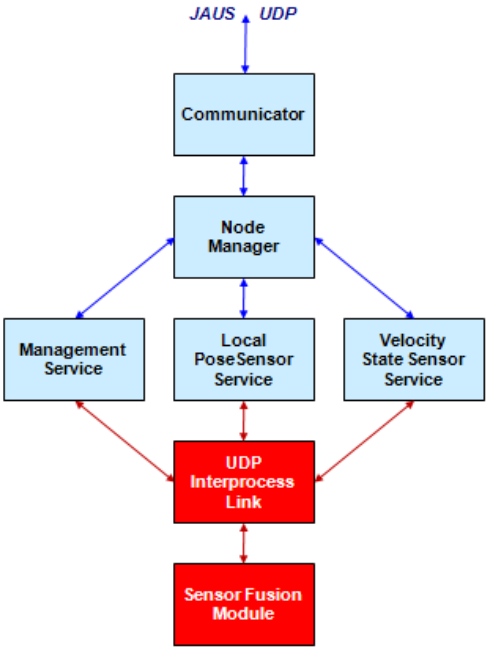

Figure 16: JAUS high level design.

6. Performance Analysis and Estimate

6.1 Safety

Viper II safety features include a manual and remote E-Stop, failsafe brakes, and a capped maximum speed of 5 mph.

6.2 Robot Performance

Vehicle performance was estimated based on the design parameters. These same attributes were measured after integration as shown in **Table 2**.

Attribute	Design Prediction
Maximum Speed	4.2 mph
Climbing Ability	2.5 inch curb
Nominal Power Consumption	500 watts
Battery Operating Time	4 hours
Waypoint accuracy (with Omnistar)	10-20 centimeters

Table 2: Performance Analysis

6.3 Reliability

Viper II utilizes dual modular power sources and a thoroughly tested drive train design that performed well in the 2008 competition. Additionally, the robot uses a rigid aluminum frame with extensive weather-proofing. Furthermore, extensive testing has been done for both the hardware and the software components. Specifically, regression and system testing was done on each module to ensure proper functionality and integration with the hardware components.

6.4 Vehicle Cost Summary

Table 3 summarizes the total material cost for the Viper II vehicle.

Component	Retail Total Cost	Team Cost
CyberPower Laptop	$3,342	$3,342
Sick LMS 291-S05 LMS	$7,000	$0
NovaTel ProPak-LB DGPS & GPS-600-LB Antenna	$4,567	$2,700
24 V NPC-T74 Motors (2)	$648	$648
Panasonic PV-GS500 digital camcorder	$800	$0
Miscellaneous Electrical Hardware	$870	$870
PNI TCM2-20 digital compass/inclinometer	$769	$0
Roboteq AX3500 Dual Channel Motor Controller	$395	$0
Main Battery 12 Volt 50 Ah AGM	$323	$323
Promariner Promite 5/5 Dual 12V Battery Charger	$110	$110
DC-to-DC Converter	$118	$118
EDFSS Hardware Components	$2,308	$780
Hollow Shaft Optical Encoder Kit (2)	$130	$130
Chassis Materials	$320	$320
Miscellaneous Hardware (nuts, bolts, etc…)	$200	$200
14" Tire & Wheel Assembly (2)	$186	$186
Rear 300 Lb Capacity Caster Wheel	$22	$22
Total	$20,931	$6,357

Table 3: Vehicle Cost Summary

7. Conclusion

Viper II continues the LTU tradition of innovation and continuous improvement. Viper II features a distinctive EDFSS system with HMI interface, as well as a backup electrical system, stable camera mount and shock absorbers, and a modular, innovative software design. All software and hardware components are easily swappable and can be upgraded in future designs. Viper II's unique design and state-of-the-art software technology set it apart from other competitors in the 17th annual Intelligent Ground Vehicle Competition.

8. References

[Ward 1994] Ward, A.C., Sobek, III, D.K., "Toyota and Set-Based Concurrent Engineering," 1994 ASME Design Theory and Methodology Conference Proceedings, Minneapolis, MN

[**http://www.jauswg.org**] The Joint Architecture for Unmanned Systems. "Reference Architecture Specification. Volume 2," 2004.

Poppendieck, M., Poppendieck, T. "Lean Software Development: An Agile Toolkit". Addison-Wesley Professional (May 18, 2003)

[**http://en.wikipedia.org/wiki/Hough_transform**] Hough Transform. March 2009.

Nilsson, N. J. , "Artificial Intelligence: A New Synthesis". Morgan Kaufmann Publishers, 1998, San Fransisco, CA.

Lawrence Technological University

Viper II

IGVC 2009 Autonomous Vehicle

Team Members

Gary Givental, Philip Munie, Shawn Ellison, Brandon Bell, Ze Cheng, Taiga Sato, Bryan Koroncey, Nathan Lucas

Faculty Advisor Statement[1]

I, Dr. CJ Chung of the Department of Math and Computer Science at Lawrence Technological University, certify that the design and development on Viper II has been significant and each team member has earned credit hours for their work.

_____ _____

Dr. CJ Chung (chung@ltu.edu, 248-204-3504) Date

[1] Co-advisor: Dr. Lisa Anneberg, Department of ECE

Table of Contents

1. Introduction

For the 2009 Intelligent Ground Vehicle Competition (IGVC), Lawrence Technological University (LTU) presents Viper II, the product of a collaborative effort among a diverse and multidisciplinary group of LTU engineering and computer science students. Viper II is an enhanced robot, with cutting-edge hardware and software updated for 2009. It represents Lawrence Tech's latest venture into developing a safe, reliable, and cost-effective autonomous vehicles that are rich in progressive technologies.

2. Innovations

2.1 Hardware Platform

2.1.1 Mechanical System

Viper II is a three wheel vehicle with front differential drive steering, a rear caster wheel, and an aluminum chassis that are all encased in a custom fiberglass mold. This design provides the most stability and maneuverability, and it allows for a zero-degree turn radius. Furthermore, the vehicle's front suspension system helps stabilize the drive-train and minimizes the shaking of the camera pole.

2.1.2 Electrical System

Viper II uses a manual electrical system that is controlled via a single electric box. There's also hardware in place (a Programmable Logic Controller (PLC) coupled with a Human Machine Interface (HMI) display) that extends the manual system to create the Electronic Diagnostic Fail-Safe System (EDFSS).

2.1.3 Software

The software for Viper II is written in the C# programming language using Visual Studio 2008 and Microsoft XNA. XNA is a set of tools with a managed runtime environment provided by Microsoft that facilitates computer game development, and it allows Viper II to quickly perform image processing on the GPU. Additionally, Viper II includes a Sensor Fusion module to bring sensor information together into one component and an easily pluggable motor control interface for its movement. Moreover, custom Local and World map software help Viper II keep track of lane, obstacle, and GPS waypoint information, and they allow it to be less reactive to the immediate environment. Furthermore, Viper II utilizes a GPS Breadcrumb system to keep track of recently visited locations, and to detect when it is heading in an incorrect direction on the course. Lastly, Viper II software is JAUS level 4 compliant.

3. Design Process

3.1 Collaborative Team Organization

2009 Team members are:

Gary Givental, MSCS – Team Captain	Philip Munie, MSCS – Autonomous Lead
Brandon Bell, MSCS – Hardware Integration Lead	Shawn Ellison, MSCS – Navigation Lead
Nathan Lucas, MSCS – JAUS Lead	Taiga Sato, BSCS
Bryan Koroncey, BSCS	Ze Cheng, BSCS

Since a large team required more communication, team members used an online discussion forum, Subversion code repository, and email to communicate effectively. Weekly meetings were scheduled to allow for status reports and to help resolve emerging issues.

3.2 Project Planning Process

The development for this year's IGVC effectively started in October 2008. **Figure 1** describes the project plan followed by the team. For this year's competition, the team met every week to go through the design plan and to perform integration testing. Since an agile, iterative development process was used, the design emerged over time as different options for the software were explored. This project plan acted as a guideline to keep the team on track, and to make sure core timelines were met.

Task	Due Date
Initial Brainstorming	2/1/2009
File Repository Setup	2/1/2009
High Level Software Design	3/1/2009
High Level Hardware Update Design	3/1/2009
Software Development	5/1/2009
E-Stop Updates	5/15/2009
System Testing	5/15/2009
Written Report	5/18/2009
Oral Presentation	6/7/2009

Figure 1: Project Tasks

3.3 Development Process

The team used agile development and set-based, concurrent methodologies for the software design in this year's competition. The set-based, concurrent development allowed the team to investigate several options for software modules and algorithms, and to find the best solution to design issues. Correspondingly, the agile development approach allowed the team to avoid the well-known pitfalls of the linear "waterfall" style approach. These techniques enabled an emergent, iterative methodology that proved to be the perfect fit and allowed for the functionality and features of the software and hardware to evolve through continuous testing and integration. The team scheduled regular weekly meetings to present status reports, to discuss progress, and to indicate any road-blocks encountered by team members during development. Frequently, students teamed up to work on a particular problem in hardware or software to get it resolved quicker, and to leverage the "pair-programming" concept.

3.4 Testing Process

A test track was setup on LTU campus, which was complete with various kinds of obstacles and curves in the lanes to approximate the challenges of the real competition. This allowed team members to collect real world data and to discover failures in software logic.

3.4.1 Hardware Testing

Viper II's electrical and mechanical systems were tested thoroughly after Viper was showcased to TARDEC, and extensive field testing was performed once the robot was re-assembled. While performing field testing, the team discovered several issues that needed to be addressed. The robot's batteries drained quickly, so a generator was obtained to keep them charged. Additionally, the GPS signal proved to be inaccurate, so the team subscribed to the Omnistar HP service to get 10-centimiter precision.

3.4.2 Software Testing

The team performed extensive unit testing, systems testing, and integration testing of the code throughout development. As new functionality was developed, it was demonstrated to the rest of the team, and, when possible, tested on the robot. Test runs were performed regularly on the test track at LTU.

3.4.3 System Integration

System integration testing was executed as quickly as possible once the development of the core components was complete. When new modules were available for general testing, they were immediately integrated into the overall system and used by all team members for testing. This modular design of the software components made it easy for new functionality to be plugged into the overall architecture. The team extensively tested the integration between the software and all the sensor hardware, as well as the motor control.

4. Hardware Design

4.1 Robot Structure

4.1.1 Chassis

Viper II is a three wheel vehicle with front differential drive steering and a rear caster wheel. This design allows for zero-turn radius maneuverability and simplified motion control. The chassis has a minimum of 4 inches of ground clearance, and this allows the robot to climb hills and ramps with an approach angle of 23 degrees. The 3-D CAD model shown in the follow **Figure 2** illustrates the design of the chassis using the ¾ inch 6063 aluminum tubing, which gives the robot the capability of carrying all of the necessary components, as well as the designated payload required for the competition. The chassis design enables Viper II to maintain its structural integrity throughout the rugged off-road course.

4.1.2 Suspension

Viper II is designed with an independent suspension that absorbs impact in the wheels from the off-road terrain. This design provides ample stability to the camera, thus allowing for a smooth video image feed. The suspension model in **Figure 3** shows the coil spring over a tube style shock that is mounted to independent control arms. This suspension style gives the robot 1 inch of travel in the shocks, and it greatly reduces the twist and roll of the chassis and components. Furthermore, the shocks are rated for 350 lbs per inch and have an adjustable rebound feature.

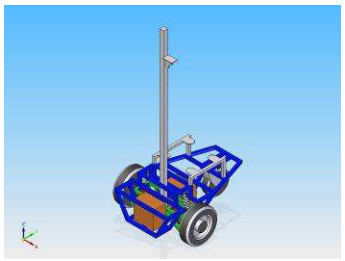

Figure 2: Chassis Model

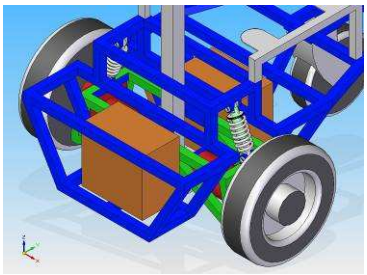

Figure 3: Front Suspension

4.1.3 Drive Train

Viper II's design uses two 24 volt, permanent magnet type model NPC-T74 high torque motors with a 20:1 gear ratio that gives the robot the ability to generate up to 1480 in-lbs (123.3 ft-lbs) of torque and allows for a maximum speed of 10.4

MPH (at 248 RPMs). Thus, these motors give Viper II plenty of power, and they eliminate the fear of premature motor failure. They ensure that Viper II can easily navigate through rough terrain, and go up a 15 degree incline.

4.1.4 Body

The body that is covering and protecting Viper II is fabricated from fiberglass, and its sleek design with smooth flowing curves and lines gives the feel of its namesake. It is designed to give easy access to the internal components through an easily removable top section, and it is fitted with electric fans to ensure that heat is removed from inside the robot. Furthermore, a tray that slides out of the right side of the fiberglass body allows access to Viper II's internal laptop computer, and it can be used without removing the top section.

4.2 Motors and Electronics

4.2.1 Motor Control

Motor control is facilitated by a Roboteq AX3500 60A/channel controller with a C# serial port interface. The AX3500 provides velocity feedback measurements through optical encoders that are mounted on the motor shafts. Additionally, the E-Stop is wired to the controller's main drive power lines via a mechanical relay that can be triggered by the top-mounted kill switch or by the wireless control system.

4.2.2 Sensors

In keeping with the theme of modularity, all of Viper II sensors (and controls) utilize RS-232 or USB 2.0 standard interfaces. **Table 1** summarizes the sensors used in Viper II.

Sensor	Function
Optical Encoders	Motor shaft position feedback for servo control
NovaTel ProPack-LB DGPS Unit	Global positioning
Digital Compass/Inclinometer	Heading, roll, and pitch information
High Dynamic Range DV Camera	Capture field image stream
Sick Laser Measurement Sensor	Provides a 180 degree polar ranging array

Table 1: Vehicle Sensor Summary

4.2.3 E-stop

Viper II is equipped with both manual and wireless (RF) remote emergency stop (E-Stop) capabilities. The Excalibur RS-310 Remote and Start Keyless Entry manufactured by Omega Research is used as the wireless E-Stop component, and its door lock function is integrated into Viper II's E-stop system. When this E-Stop is activated, it removes power from the drive train and engages both of the front wheel failsafe electromagnetic brakes.

4.2.4 Vehicle Alert System (JAUS Horn and Lights Hardware)

Viper II includes an electrical module that can switch relatively large electrical loads (using relays) to the horn and light systems via commands sent over a USB interface. It is utilized by the vehicle's alert system, and it is activated by Viper II's JAUS software.

4.3 Electrical System

4.3.1 Power System

Viper II has two parallel electrical systems: a logic control system and a manual control system. The logic control system consists of a Programmable Logic Controller (PLC) and a Human Machine Interface (HMI), and these components make up the Electronic Diagnostic Fail-Safe System (EDFSS). The manual system consists of toggle switches and fuses that allow the team to physically turn on and off each electrical component. The operating mode can each be selected by a keyed selector switch found on the main system control board.

4.3.2 Electronic Diagnostic Fail-Safe System - EDFSS

The EDFSS provides Viper II with a nearly endless range of abilities for its current design and future design modifications. It makes the use of an external touch screen display that can be programmed to control each component's power and to display status information.

4.3.2.1 External status indicator lights

During previous competitions, the team realized that easily visible power indicators for key component were needed during testing. The EDFSS solves this problem by controlling three indicator lights: the green light specifies all systems are ready, the yellow light signifies that Viper II is operational, but not all systems are responding, and the red light indicates that the system is not operational. Additional indicator lights are also available next to each manual switch on the electrical box. This functionality allows the IGVC team to quickly identify power issues during testing.

4.3.2.2 Manual Control

The manual system is a basic on/off component control. It acts as a backup electrical system if there are problems with the EDFSS system.

4.3.3 Power Source

Viper II is powered by two 12V 50 Ah AGM batteries connected in series to provide a 24V electrical system. It uses a 24/12V DC/DC converter for the onboard 12V systems and an onboard charger with 110VAC interface for restoring battery power. **Figure 4** illustrates how power for the left and right motors is fed directly into the motor controller to maximize the motor power. However, the control power for the motor controller is regulated by the E-Stop and electrical control box.

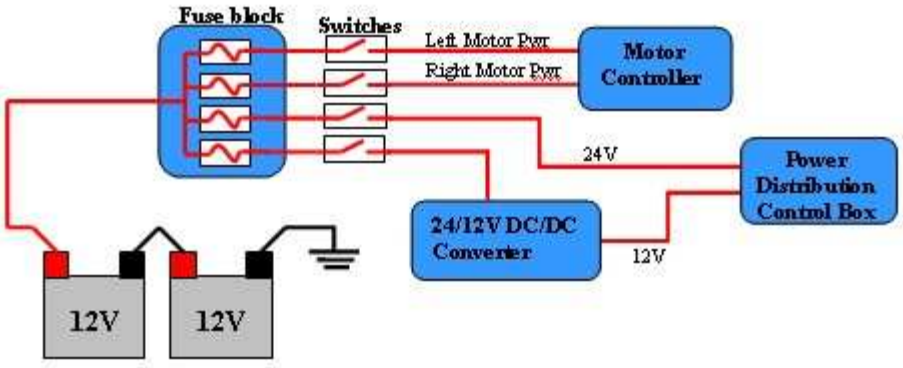

Figure 4: Power Source Schematic

4.3.3 Power Distribution

Viper II is equipped with an all-in-one electrical control box. This power distribution control box contains the systems power distribution and emergency stop printed-circuit board (PCB), JAUS PCB hardware, wireless emergency stop PCB hardware, power control switches for each component, and the main system selector switch that controls the manual and EDFSS operation modes. It is designed to be self contained and removable from Viper II, and it also offers additional testing and diagnostic abilities for system voltage and current ampere measurements.

The two main power connectors are automotive grade EDAC panel connectors, and they allow for an easy connection point for two wire harnesses, which route power to and from each hardware item and the EDFSS block. The power and communications control system schematic for Viper II vehicle is shown in **Figure 5**.

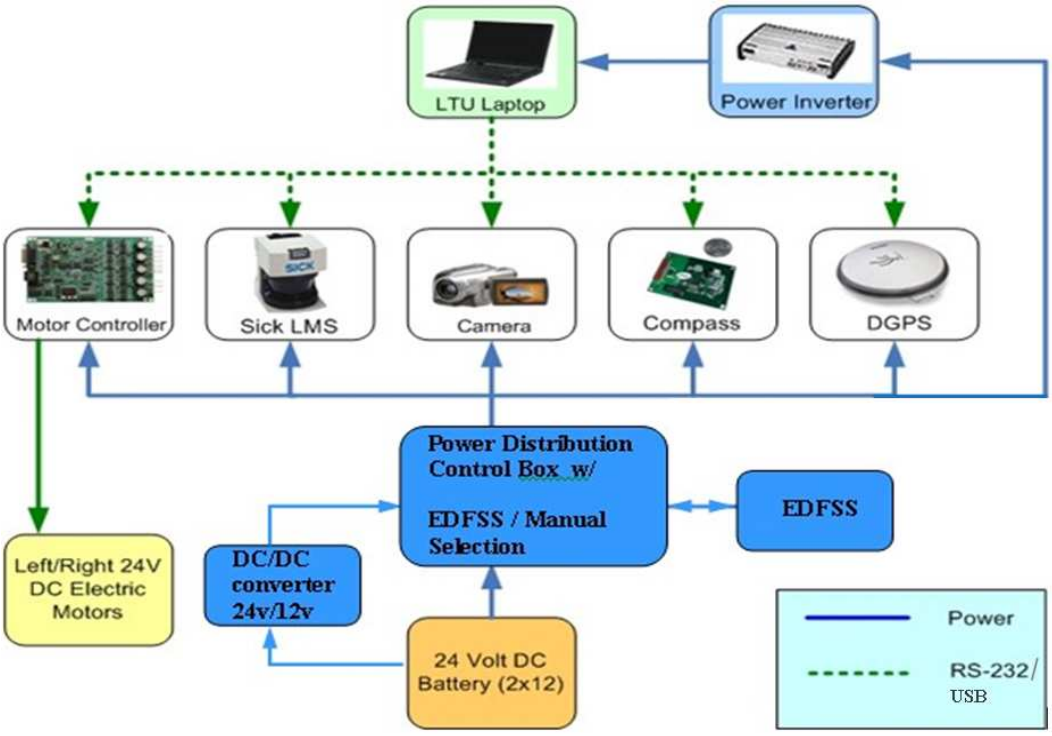

Figure 5: Power and Communications Control System Schematic

5. Software Design

5.1 Software Strategy

C# , XNA, and the Visual Studio 2008 IDE were chosen for all software development on the robot this year. C# is a powerful object oriented language, and Microsoft XNA is a video gaming toolset which allowed Viper II to utilize the processing power of GPU for image analysis. The Visual Studio IDE allowed for rapid application development with easy to build visual interfaces. **Figure 6** shows the high level software architecture design.

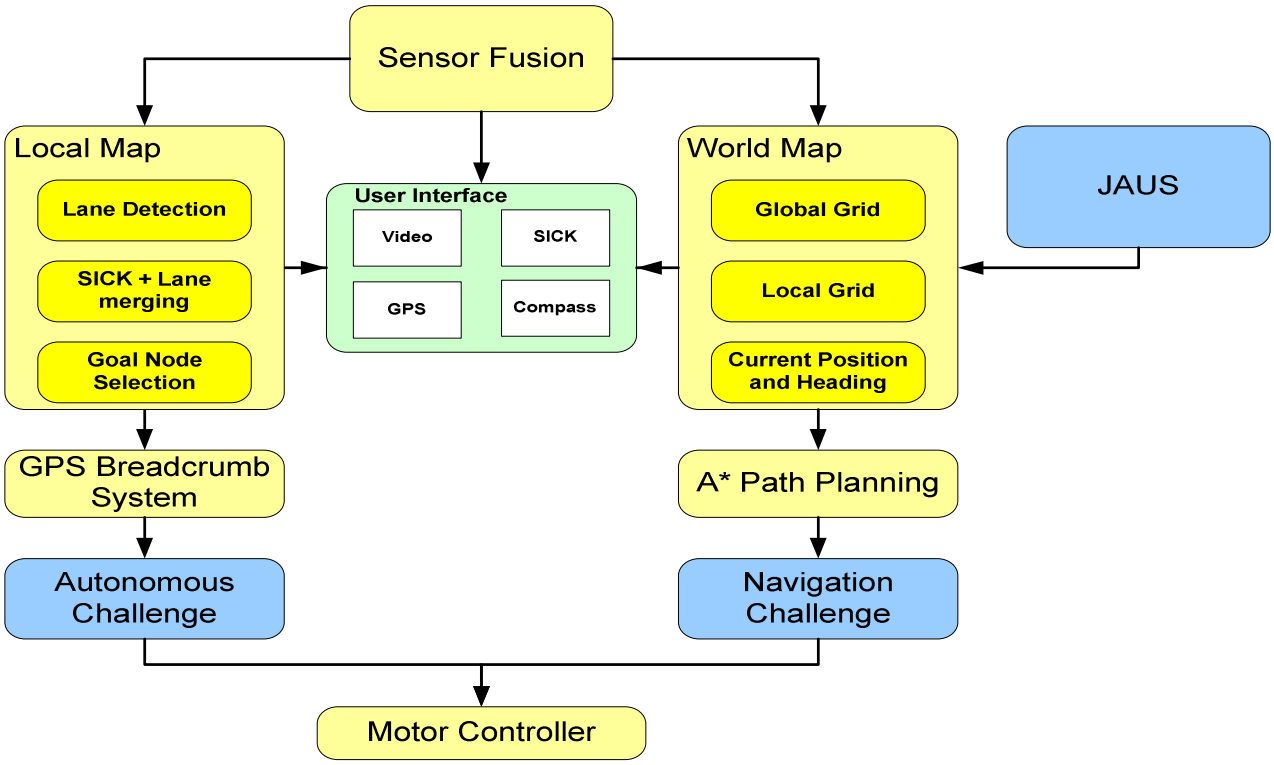

Figure 6: High Level Software Design

5.2 Sensor Fusion

For this year's competition, a Sensor Fusion Module shown in **Figure 7** was developed to collect data from all the hardware sensors, and to present this data to all subscribers in a consistent manner. This allows subscribers to access sensor data without knowing the details of hardware layer implementation.

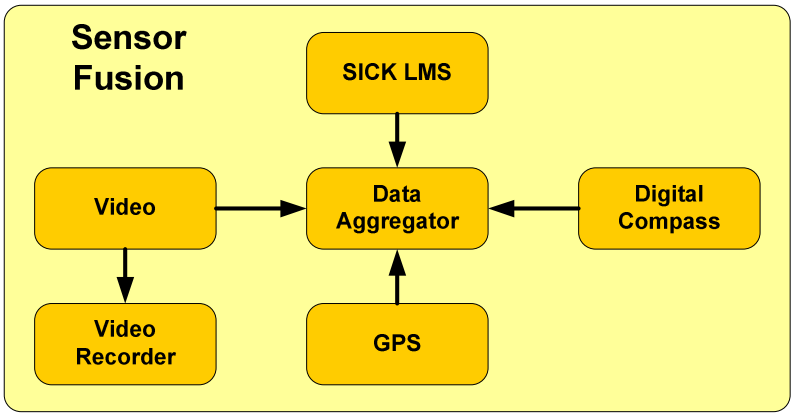

Figure 7: Sensor Fusion

5.3 World Map and Path Finding

The world map plots a geographic region to a Cartesian coordinate plane. The total size of the Cartesian map is determined by the size of the area to be represented and the desired resolution. Generally, a resolution of ½ meter has proven to work well during testing. Each point in the geographic region can be mapped to a cell in the Cartesian map, and vice versa, through the use of a conversion object. In this way, a simple A* path finding algorithm can be used to find the shortest path between any two given points.

5.4 Motor Control Module

The motor controller module was designed to be easily plugged into any of the team's projects. Once initialized, this module can accept simple operation commands for speed and direction, which it then translates into the ASCII communication strings that are sent to the motor controller via a serial connection. This module allows team members to have simple access to the hardware without knowledge of the motor controller's underlying communication protocols.

5.5 Autonomous Challenge Details

The main algorithm for the autonomous challenge consists of using computer vision to identify lane information and SICK data output to detect obstacles. To accomplish this, the camera is first calibrated so that the vision processing can map lanes onto a local map alongside the SICK data, and then a goal node selector algorithm is used to find the appropriate heading. Additionally, a GPS Breadcrumb utility, which uses GPS location information and compass headings to store a list of recently visited locations, is used to detect if the robot has become turned around. This helps the robot avoid going backwards on the course.

5.5.1 Camera Calibration

The camera calibration is responsible for synchronizing the lane data with the SICK obstacle data. It works by calculating the distance of each pixel in the vision's captured image, and it uses the SICK as the reference point. It is accomplished by measuring the distance of predefined positions on the captured image and interpolating the remaining positions on the image. A snapshot of the tool for calibration is shown in **Figure 8**.

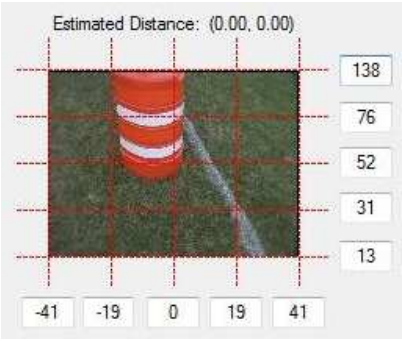

Figure 8: Camera Calibration Tool

5.5.2 SICK Data Gathering

As the SICK data is collected, it is mapped onto a local map, combined with the filtered camera image. Because of the calibration previously, the data can simply be placed in the local map without any extra processing. See **Figure 9**.

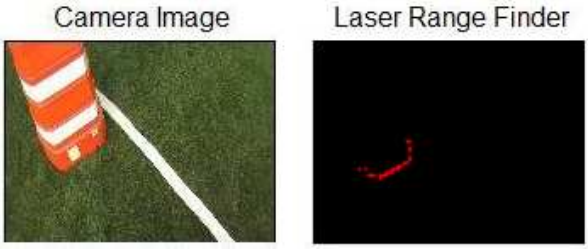

Figure 9: Example of mapping SICK data to the Local Map

5.5.3 Image Processing

For this year's competition, Viper II continues to use a local map approach for combining the vision information with obstacle data. First, SICK obstacle data is added into the local map as red pixels. Then, the camera image is captured and converted to black and white using a brightest pixel filter. This filter runs on the GPU and chooses the brightest pixel on each horizontal line. Then blob detection is used to filter out noise pixels that are not considered to be lanes. Once noise has been removed, the filtered camera image is added to the SICK data and all of the white pixels that are vertically above the red pixels are repainted as black. A Hough Transform filter is then applied to find any partial lanes in the image and fill them in. Finally, a bleeding filter is used to color any pixels vertically above red pixels as red, and above white pixels as white. Thus, the local map represents both the lane information in white pixels and the SICK obstacle data in red pixels. **Figure 10** demonstrates the steps described above.

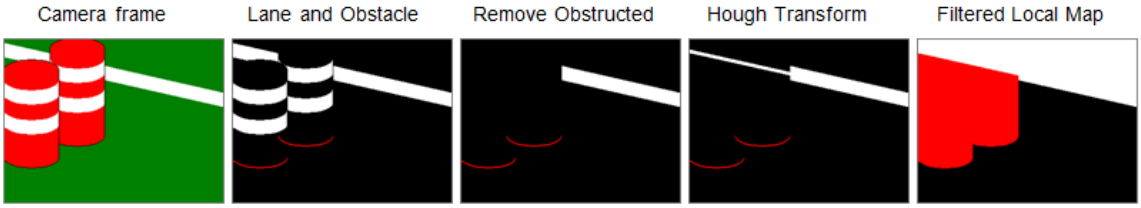

Figure 10: Image Processing Steps

5.5.4 GPU Processing

The results from previous IGVC competitions indicated that the required image processing on a laptop CPU was too strenuous for the hardware. To solve this issue, it was found that the GPU could be used as an alternative for some of the processing. The GPU is exceptional for manipulating image data and therefore is an excellent candidate for fast image processing. XNA was used to communicate with the GPU. Normally, XNA is used for video game development, but it is C# based and was easy to integrate with the existing codebase. Team Viper converted several image filtering algorithms, such as the Brightest Pixel filter, to run on the GPU to achieve dramatic performance improvements.

5.5.5 Hough Transform

A Hough Transform helps Viper II identify lanes in the local map, but it is primarily used for filling in a lane when only a partial (dashed) one exists. See **Figure 11** for an example of the Hough Transform filling in missing data. This transform is applied twice, because the local map is split vertically. The splitting of the local map forces the detection of a lane on both sides, and this is critical in cases where both of the lanes exist on the local map at once. Since a single image does not guarantee the Hough Transform will recognize both lanes, each image must be evaluated for missing data. This lack of recognition becomes extremely apparent when the second lane's intensity is less dominant.

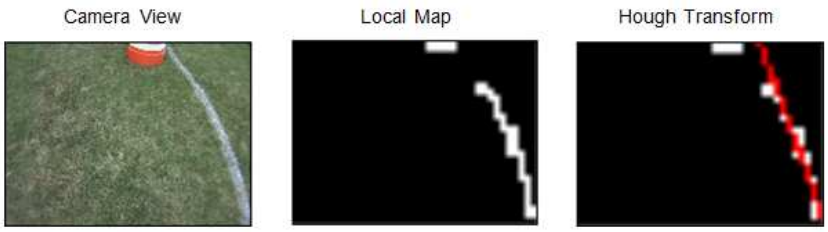

Figure 11: Hough Transform applied to dashed lane

5.5.6 Blob Detection

Blob detection and filtering allows for the elimination of noise pixels from the camera image. As the image is scanned, each pixel is examined to determine its color. Once this process is finished, each contiguous region of color in the image will have a corresponding blob object, and these blobs are useful for several reasons. First, they allow identification of all regions of color and this can help the detection algorithm filter out blobs that are too small (and thus are just noise). Secondly, blobs of "non-obstacle" space can be identified, and this ensures that the detection algorithm selects the goal node that is in the same region of the robot. Thus, the robot will not cross a lane to get to the local goal location.

5.5.7 Lane Following and Obstacle Avoidance

Viper II uses a goal node selection algorithm to find the best path through the obstacle course. This algorithm first finds the largest gap in the image, and then plots a goal node in the center of this gap. If the largest gap on the map is smaller than the Viper II, it retreats and searches for a path that it can fit through. Once the goal node is found, Viper II starts to move in the direction of this node until it processes new image data. Additionally, a new algorithm was added this year to avoid turning in the wrong direction when facing a lane on one side and obstacles on the other side of the robot.

5.5.8 Goal Node Selection

Goal node selection is an important aspect of the autonomous navigation challenge, and it relies on the local map. Once the local map is built, it is then transformed into an image, and a goal location for the robot's heading must be found. This is handled by finding the largest gap between obstacles and lanes. Ideally, this gap is at the top of the image and as far away as possible from the robot. However, if a lane bisects the image, then the gap is calculated from either the left or right side of the image. The images below demonstrate the goal node selection algorithm. The goal location is the orange dot in the images shown in **Figure 12**.

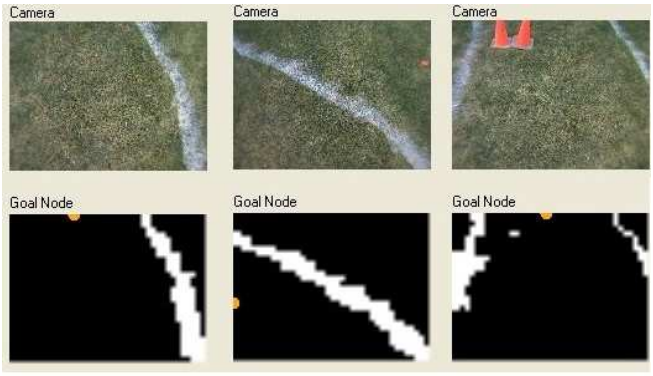

Figure 12: Goal Node Selection Example

To improve decision making when selecting a goal node, the local map is analyzed twice; the first check includes only lane information and the second check includes both lane and obstacle information. Analyzing only the lane information allows the robot to always conclude that it should turn away from the lane. This decision is followed even if there's an obstacle in the direction opposite of the lane. Even if turning causes the Viper II to face an obstacle, it will continue turning away from the lane until it sees a clear path. **Figure 13** shows the two goal node checks, and **Figure 14** shows the decision process to turn away from the lane (blue).

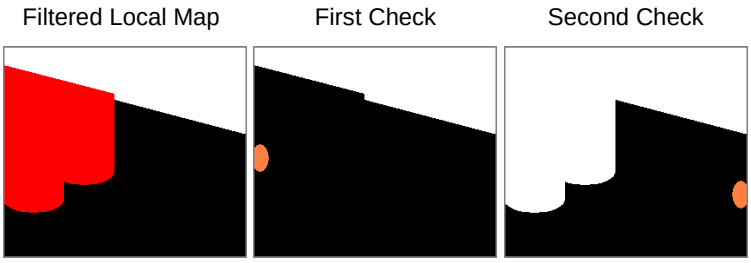

Figure 13: Two Goal Node Checks

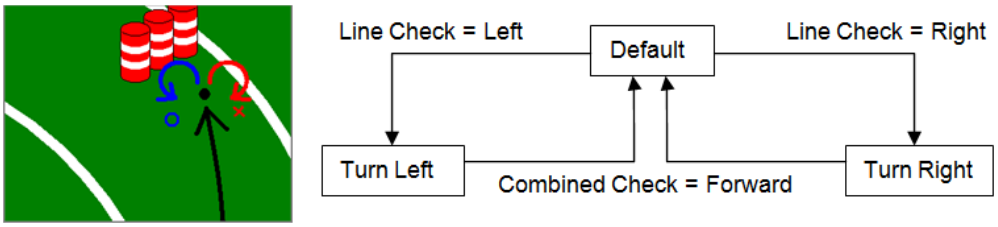

Figure 14: Direction Decision and State Diagram to Escape from a Trap

5.6 Navigation Challenge Details

The primary algorithm of the navigation challenge uses the GPS, compass, and SICK laser range finder sensors in conjunction with dead-reckoning information available from the motor controller (derived from optical encoders on the motor shafts) to move Viper II through the course to the various GPS checkpoints. The GPS waypoint locations are mapped onto a World Map. As Viper II moves through the course and discovers obstacles using the SICK, the location of the obstacles is also mapped. This allows Viper II to adjust its course and navigate around the obstacles to the next checkpoint. Additionally, this year's robot is also using a Fuzzy Controller to smooth out the robots movement through the course.

5.6.1 Waypoints Scheduling

To select the quickest route between waypoints, Team Viper classifies the problem as a Traveling Salesman Problem. For simplicity, Viper II uses a Greedy Algorithm to determine the visiting order. The obstacles detected with the SICK may change the path from the robots' position to the waypoints. Therefore, the robot will reschedule its visiting order when each sub-destination is reached.

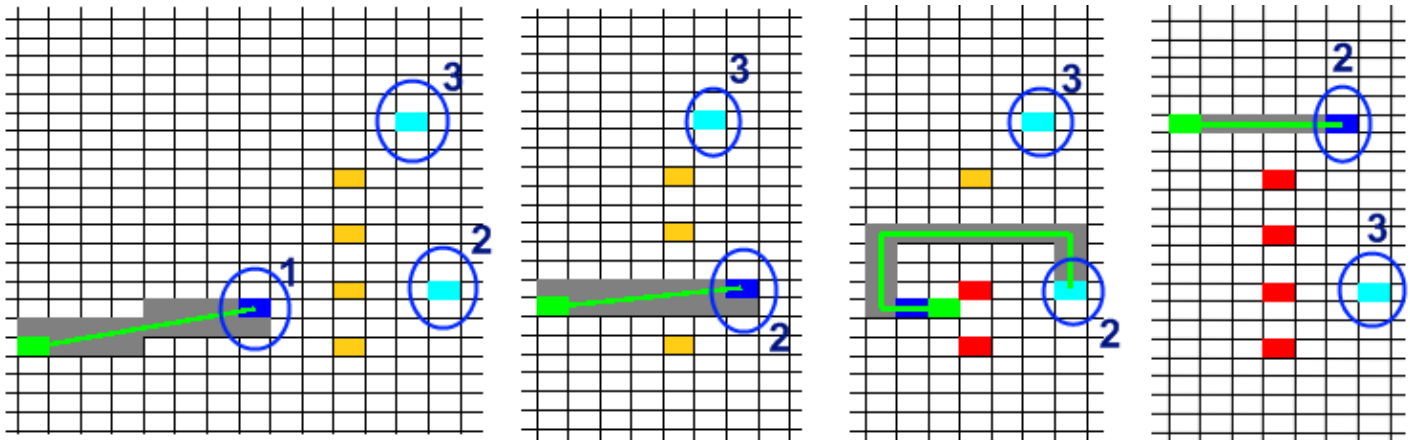

Figure 15: Waypoint scheduler

Light blue boxes represent waypoints to be visited, and dark blue is the current goal waypoint. Red boxes are detected obstacles and orange ones are not yet known.

Figure 15 demonstrates how the Greedy Waypoint Scheduling algorithm allows the Viper II to dynamically adjust its path based on newly discovered obstacle data. After the robot visits the 1st waypoint and starts heading towards the 2nd one, it discovers new obstacles. The obstacle data is mapped onto the World Map. At this point, Viper II still tries to go to Waypoint #2, but finds even more obstacles blocking its path. Finally, as it attempts to avoid obstacles, Viper II finds that it is now closer to the waypoint at the top of the map, so it schedules this waypoint as the second one to visit.

5.6.2. Fuzzy Controller

A fuzzy logic controller was created to help fix the problem with oscillation in the Viper's movement while performing obstacle avoidance in the Navigation Challenge. The controller takes the robot's current and desired direction headings as inputs, and outputs an adjusted heading. The fuzzy logic rules in the controller limit oscillation in the robot's movements by smoothing out the otherwise drastic changes in direction. A smoother path through the course helps Viper II get to all the waypoints faster.

5.7 JAUS Challenge Details

Viper's JAUS implementation has been completely redesigned for IGVC 2009. The intent was to leverage the modularity aspect of JAUS to establish a design that is easily expanded for future requirements. The implementation is comprised of a network of JAUS components working together to fulfill the JAUS Challenge requirements. A Node Manager routes all JAUS message traffic between JAUS components on Viper. All extra-nodal JAUS messages to and from Viper are translated to the Transport Layer and exchanged by a Communicator component. Figure 15 depicts the JAUS network in blue and the associated Viper modules in red.

5.7.1 Process for Learning JAUS

Development of the JAUS network benefitted greatly from experience gained preparing for and participating in last year's IGVC competition. Although the Reference Architecture was well understood by the team, a considerable amount of time was spent studying and trying to understand the new SAE specifications. The team was provided access to the JAUS Validation Tool (JVT) at http://www.usaric.org/JVT. This tool was essential for validating the team's JAUS implementation.

5.7.2 JAUS Integration into the Design

The Viper JAUS components are written in Java while the Sensor Fusion and mission modules are written in C#. A UDP inter-process link provides the interface between the JAUS components and the rest of the Viper software. This design strategy was inspired by the modular architecture of JAUS. Once the UDP link was defined, Viper's JAUS implementation was developed and tested independently of the rest of the software. **Figure 16** demonstrates the JAUS component and how it communicates to the Viper II's Sensor Fusion module.

5.7.3 Challenges Encountered

The principal challenges encountered implementing JAUS center around the new SAE requirements. The SAE specifications introduced by the 2009 IGVC Rules are vastly more complex than the Reference Architecture. The specifications establish many features new to JAUS, including modified and expanded transport requirements, a multi-layered architecture, and service oriented definitions. Specification availability, completeness, and accuracy were also challenges.

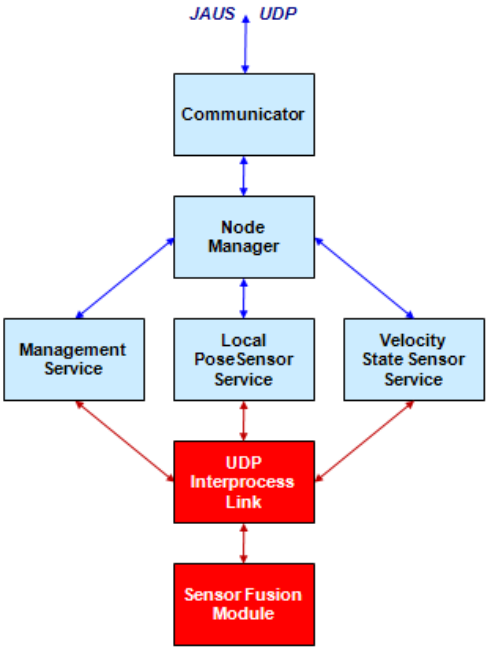

Figure 16: JAUS high level design.

6. Performance Analysis and Estimate

6.1 Safety

Viper II safety features include a manual and remote E-Stop, failsafe brakes, and a capped maximum speed of 5 mph.

6.2 Robot Performance

Vehicle performance was estimated based on the design parameters. These same attributes were measured after integration as shown in **Table 2**.

Attribute	Design Prediction
Maximum Speed	4.2 mph
Climbing Ability	2.5 inch curb
Nominal Power Consumption	500 watts
Battery Operating Time	4 hours
Waypoint accuracy (with Omnistar)	10-20 centimeters

Table 2: Performance Analysis

6.3 Reliability

Viper II utilizes dual modular power sources and a thoroughly tested drive train design that performed well in the 2008 competition. Additionally, the robot uses a rigid aluminum frame with extensive weather-proofing. Furthermore, extensive testing has been done for both the hardware and the software components. Specifically, regression and system testing was done on each module to ensure proper functionality and integration with the hardware components.

6.4 Vehicle Cost Summary

Table 3 summarizes the total material cost for the Viper II vehicle.

Component	Retail Total Cost	Team Cost
CyberPower Laptop	$3,342	$3,342
Sick LMS 291-S05 LMS	$7,000	$0
NovaTel ProPak-LB DGPS & GPS-600-LB Antenna	$4,567	$2,700
24 V NPC-T74 Motors (2)	$648	$648
Panasonic PV-GS500 digital camcorder	$800	$0
Miscellaneous Electrical Hardware	$870	$870
PNI TCM2-20 digital compass/inclinometer	$769	$0
Roboteq AX3500 Dual Channel Motor Controller	$395	$0
Main Battery 12 Volt 50 Ah AGM	$323	$323
Promariner Promite 5/5 Dual 12V Battery Charger	$110	$110
DC-to-DC Converter	$118	$118
EDFSS Hardware Components	$2,308	$780
Hollow Shaft Optical Encoder Kit (2)	$130	$130
Chassis Materials	$320	$320
Miscellaneous Hardware (nuts, bolts, etc…)	$200	$200
14" Tire & Wheel Assembly (2)	$186	$186
Rear 300 Lb Capacity Caster Wheel	$22	$22
Total	$20,931	$6,357

Table 3: Vehicle Cost Summary

7. Conclusion

Viper II continues the LTU tradition of innovation and continuous improvement. Viper II features a distinctive EDFSS system with HMI interface, as well as a backup electrical system, stable camera mount and shock absorbers, and a modular, innovative software design. All software and hardware components are easily swappable and can be upgraded in future designs. Viper II's unique design and state-of-the-art software technology set it apart from other competitors in the 17th annual Intelligent Ground Vehicle Competition.

8. References

[Ward 1994] Ward, A.C., Sobek, III, D.K., "Toyota and Set-Based Concurrent Engineering," 1994 ASME Design Theory and Methodology Conference Proceedings, Minneapolis, MN

[**http://www.jauswg.org**] The Joint Architecture for Unmanned Systems. "Reference Architecture Specification. Volume 2," 2004.

Poppendieck, M., Poppendieck, T. "Lean Software Development: An Agile Toolkit". Addison-Wesley Professional (May 18, 2003)

[**http://en.wikipedia.org/wiki/Hough_transform**] Hough Transform. March 2009.

Nilsson, N. J. , "Artificial Intelligence: A New Synthesis". Morgan Kaufmann Publishers, 1998, San Fransisco, CA.

Lawrence Technological University

Viper II

IGVC 2009 Autonomous Vehicle

Team Members

Gary Givental, Philip Munie, Shawn Ellison, Brandon Bell, Ze Cheng, Taiga Sato, Bryan Koroncey, Nathan Lucas

Faculty Advisor Statement[1]

I, Dr. CJ Chung of the Department of Math and Computer Science at Lawrence Technological University, certify that the design and development on Viper II has been significant and each team member has earned credit hours for their work.

_____ _____

Dr. CJ Chung (chung@ltu.edu, 248-204-3504) Date

[1] Co-advisor: Dr. Lisa Anneberg, Department of ECE

Table of Contents

1. Introduction

For the 2009 Intelligent Ground Vehicle Competition (IGVC), Lawrence Technological University (LTU) presents Viper II, the product of a collaborative effort among a diverse and multidisciplinary group of LTU engineering and computer science students. Viper II is an enhanced robot, with cutting-edge hardware and software updated for 2009. It represents Lawrence Tech's latest venture into developing a safe, reliable, and cost-effective autonomous vehicles that are rich in progressive technologies.

2. Innovations

2.1 Hardware Platform

2.1.1 Mechanical System

Viper II is a three wheel vehicle with front differential drive steering, a rear caster wheel, and an aluminum chassis that are all encased in a custom fiberglass mold. This design provides the most stability and maneuverability, and it allows for a zero-degree turn radius. Furthermore, the vehicle's front suspension system helps stabilize the drive-train and minimizes the shaking of the camera pole.

2.1.2 Electrical System

Viper II uses a manual electrical system that is controlled via a single electric box. There's also hardware in place (a Programmable Logic Controller (PLC) coupled with a Human Machine Interface (HMI) display) that extends the manual system to create the Electronic Diagnostic Fail-Safe System (EDFSS).

2.1.3 Software

The software for Viper II is written in the C# programming language using Visual Studio 2008 and Microsoft XNA. XNA is a set of tools with a managed runtime environment provided by Microsoft that facilitates computer game development, and it allows Viper II to quickly perform image processing on the GPU. Additionally, Viper II includes a Sensor Fusion module to bring sensor information together into one component and an easily pluggable motor control interface for its movement. Moreover, custom Local and World map software help Viper II keep track of lane, obstacle, and GPS waypoint information, and they allow it to be less reactive to the immediate environment. Furthermore, Viper II utilizes a GPS Breadcrumb system to keep track of recently visited locations, and to detect when it is heading in an incorrect direction on the course. Lastly, Viper II software is JAUS level 4 compliant.

3. Design Process

3.1 Collaborative Team Organization

2009 Team members are:

Gary Givental, MSCS – Team Captain	Philip Munie, MSCS – Autonomous Lead
Brandon Bell, MSCS – Hardware Integration Lead	Shawn Ellison, MSCS – Navigation Lead
Nathan Lucas, MSCS – JAUS Lead	Taiga Sato, BSCS
Bryan Koroncey, BSCS	Ze Cheng, BSCS

Since a large team required more communication, team members used an online discussion forum, Subversion code repository, and email to communicate effectively. Weekly meetings were scheduled to allow for status reports and to help resolve emerging issues.

3.2 Project Planning Process

The development for this year's IGVC effectively started in October 2008. **Figure 1** describes the project plan followed by the team. For this year's competition, the team met every week to go through the design plan and to perform integration testing. Since an agile, iterative development process was used, the design emerged over time as different options for the software were explored. This project plan acted as a guideline to keep the team on track, and to make sure core timelines were met.

Task	Due Date
Initial Brainstorming	2/1/2009
File Repository Setup	2/1/2009
High Level Software Design	3/1/2009
High Level Hardware Update Design	3/1/2009
Software Development	5/1/2009
E-Stop Updates	5/15/2009
System Testing	5/15/2009
Written Report	5/18/2009
Oral Presentation	6/7/2009

Figure 1: Project Tasks

3.3 Development Process

The team used agile development and set-based, concurrent methodologies for the software design in this year's competition. The set-based, concurrent development allowed the team to investigate several options for software modules and algorithms, and to find the best solution to design issues. Correspondingly, the agile development approach allowed the team to avoid the well-known pitfalls of the linear "waterfall" style approach. These techniques enabled an emergent, iterative methodology that proved to be the perfect fit and allowed for the functionality and features of the software and hardware to evolve through continuous testing and integration. The team scheduled regular weekly meetings to present status reports, to discuss progress, and to indicate any road-blocks encountered by team members during development. Frequently, students teamed up to work on a particular problem in hardware or software to get it resolved quicker, and to leverage the "pair-programming" concept.

3.4 Testing Process

A test track was setup on LTU campus, which was complete with various kinds of obstacles and curves in the lanes to approximate the challenges of the real competition. This allowed team members to collect real world data and to discover failures in software logic.

3.4.1 Hardware Testing

Viper II's electrical and mechanical systems were tested thoroughly after Viper was showcased to TARDEC, and extensive field testing was performed once the robot was re-assembled. While performing field testing, the team discovered several issues that needed to be addressed. The robot's batteries drained quickly, so a generator was obtained to keep them charged. Additionally, the GPS signal proved to be inaccurate, so the team subscribed to the Omnistar HP service to get 10-centimiter precision.

3.4.2 Software Testing

The team performed extensive unit testing, systems testing, and integration testing of the code throughout development. As new functionality was developed, it was demonstrated to the rest of the team, and, when possible, tested on the robot. Test runs were performed regularly on the test track at LTU.

3.4.3 System Integration

System integration testing was executed as quickly as possible once the development of the core components was complete. When new modules were available for general testing, they were immediately integrated into the overall system and used by all team members for testing. This modular design of the software components made it easy for new functionality to be plugged into the overall architecture. The team extensively tested the integration between the software and all the sensor hardware, as well as the motor control.

4. Hardware Design

4.1 Robot Structure

4.1.1 Chassis

Viper II is a three wheel vehicle with front differential drive steering and a rear caster wheel. This design allows for zero-turn radius maneuverability and simplified motion control. The chassis has a minimum of 4 inches of ground clearance, and this allows the robot to climb hills and ramps with an approach angle of 23 degrees. The 3-D CAD model shown in the follow **Figure 2** illustrates the design of the chassis using the ¾ inch 6063 aluminum tubing, which gives the robot the capability of carrying all of the necessary components, as well as the designated payload required for the competition. The chassis design enables Viper II to maintain its structural integrity throughout the rugged off-road course.

4.1.2 Suspension

Viper II is designed with an independent suspension that absorbs impact in the wheels from the off-road terrain. This design provides ample stability to the camera, thus allowing for a smooth video image feed. The suspension model in **Figure 3** shows the coil spring over a tube style shock that is mounted to independent control arms. This suspension style gives the robot 1 inch of travel in the shocks, and it greatly reduces the twist and roll of the chassis and components. Furthermore, the shocks are rated for 350 lbs per inch and have an adjustable rebound feature.

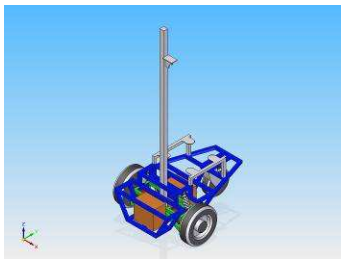

Figure 2: Chassis Model

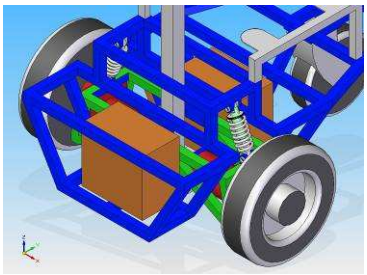

Figure 3: Front Suspension

4.1.3 Drive Train

Viper II's design uses two 24 volt, permanent magnet type model NPC-T74 high torque motors with a 20:1 gear ratio that gives the robot the ability to generate up to 1480 in-lbs (123.3 ft-lbs) of torque and allows for a maximum speed of 10.4

MPH (at 248 RPMs). Thus, these motors give Viper II plenty of power, and they eliminate the fear of premature motor failure. They ensure that Viper II can easily navigate through rough terrain, and go up a 15 degree incline.

4.1.4 Body

The body that is covering and protecting Viper II is fabricated from fiberglass, and its sleek design with smooth flowing curves and lines gives the feel of its namesake. It is designed to give easy access to the internal components through an easily removable top section, and it is fitted with electric fans to ensure that heat is removed from inside the robot. Furthermore, a tray that slides out of the right side of the fiberglass body allows access to Viper II's internal laptop computer, and it can be used without removing the top section.

4.2 Motors and Electronics

4.2.1 Motor Control

Motor control is facilitated by a Roboteq AX3500 60A/channel controller with a C# serial port interface. The AX3500 provides velocity feedback measurements through optical encoders that are mounted on the motor shafts. Additionally, the E-Stop is wired to the controller's main drive power lines via a mechanical relay that can be triggered by the top-mounted kill switch or by the wireless control system.

4.2.2 Sensors

In keeping with the theme of modularity, all of Viper II sensors (and controls) utilize RS-232 or USB 2.0 standard interfaces. **Table 1** summarizes the sensors used in Viper II.

Sensor	Function
Optical Encoders	Motor shaft position feedback for servo control
NovaTel ProPack-LB DGPS Unit	Global positioning
Digital Compass/Inclinometer	Heading, roll, and pitch information
High Dynamic Range DV Camera	Capture field image stream
Sick Laser Measurement Sensor	Provides a 180 degree polar ranging array

Table 1: Vehicle Sensor Summary

4.2.3 E-stop

Viper II is equipped with both manual and wireless (RF) remote emergency stop (E-Stop) capabilities. The Excalibur RS-310 Remote and Start Keyless Entry manufactured by Omega Research is used as the wireless E-Stop component, and its door lock function is integrated into Viper II's E-stop system. When this E-Stop is activated, it removes power from the drive train and engages both of the front wheel failsafe electromagnetic brakes.

4.2.4 Vehicle Alert System (JAUS Horn and Lights Hardware)

Viper II includes an electrical module that can switch relatively large electrical loads (using relays) to the horn and light systems via commands sent over a USB interface. It is utilized by the vehicle's alert system, and it is activated by Viper II's JAUS software.

4.3 Electrical System

4.3.1 Power System

Viper II has two parallel electrical systems: a logic control system and a manual control system. The logic control system consists of a Programmable Logic Controller (PLC) and a Human Machine Interface (HMI), and these components make up the Electronic Diagnostic Fail-Safe System (EDFSS). The manual system consists of toggle switches and fuses that allow the team to physically turn on and off each electrical component. The operating mode can each be selected by a keyed selector switch found on the main system control board.

4.3.2 Electronic Diagnostic Fail-Safe System - EDFSS

The EDFSS provides Viper II with a nearly endless range of abilities for its current design and future design modifications. It makes the use of an external touch screen display that can be programmed to control each component's power and to display status information.

4.3.2.1 External status indicator lights

During previous competitions, the team realized that easily visible power indicators for key component were needed during testing. The EDFSS solves this problem by controlling three indicator lights: the green light specifies all systems are ready, the yellow light signifies that Viper II is operational, but not all systems are responding, and the red light indicates that the system is not operational. Additional indicator lights are also available next to each manual switch on the electrical box. This functionality allows the IGVC team to quickly identify power issues during testing.

4.3.2.2 Manual Control

The manual system is a basic on/off component control. It acts as a backup electrical system if there are problems with the EDFSS system.

4.3.3 Power Source

Viper II is powered by two 12V 50 Ah AGM batteries connected in series to provide a 24V electrical system. It uses a 24/12V DC/DC converter for the onboard 12V systems and an onboard charger with 110VAC interface for restoring battery power. **Figure 4** illustrates how power for the left and right motors is fed directly into the motor controller to maximize the motor power. However, the control power for the motor controller is regulated by the E-Stop and electrical control box.

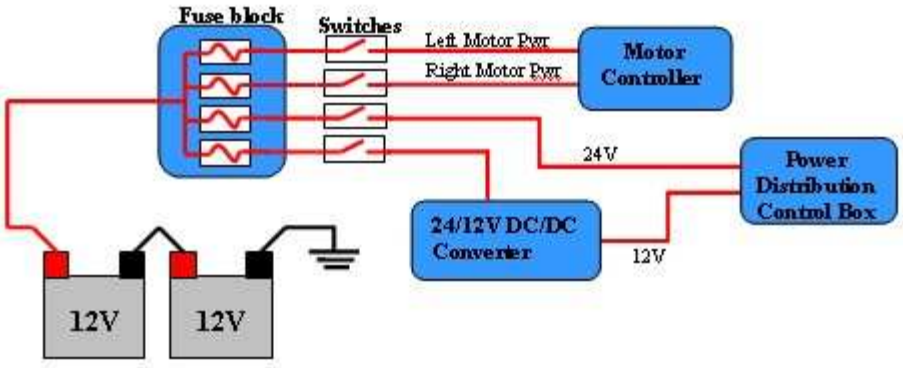

Figure 4: Power Source Schematic

4.3.3 Power Distribution

Viper II is equipped with an all-in-one electrical control box. This power distribution control box contains the systems power distribution and emergency stop printed-circuit board (PCB), JAUS PCB hardware, wireless emergency stop PCB hardware, power control switches for each component, and the main system selector switch that controls the manual and EDFSS operation modes. It is designed to be self contained and removable from Viper II, and it also offers additional testing and diagnostic abilities for system voltage and current ampere measurements.

The two main power connectors are automotive grade EDAC panel connectors, and they allow for an easy connection point for two wire harnesses, which route power to and from each hardware item and the EDFSS block. The power and communications control system schematic for Viper II vehicle is shown in **Figure 5**.

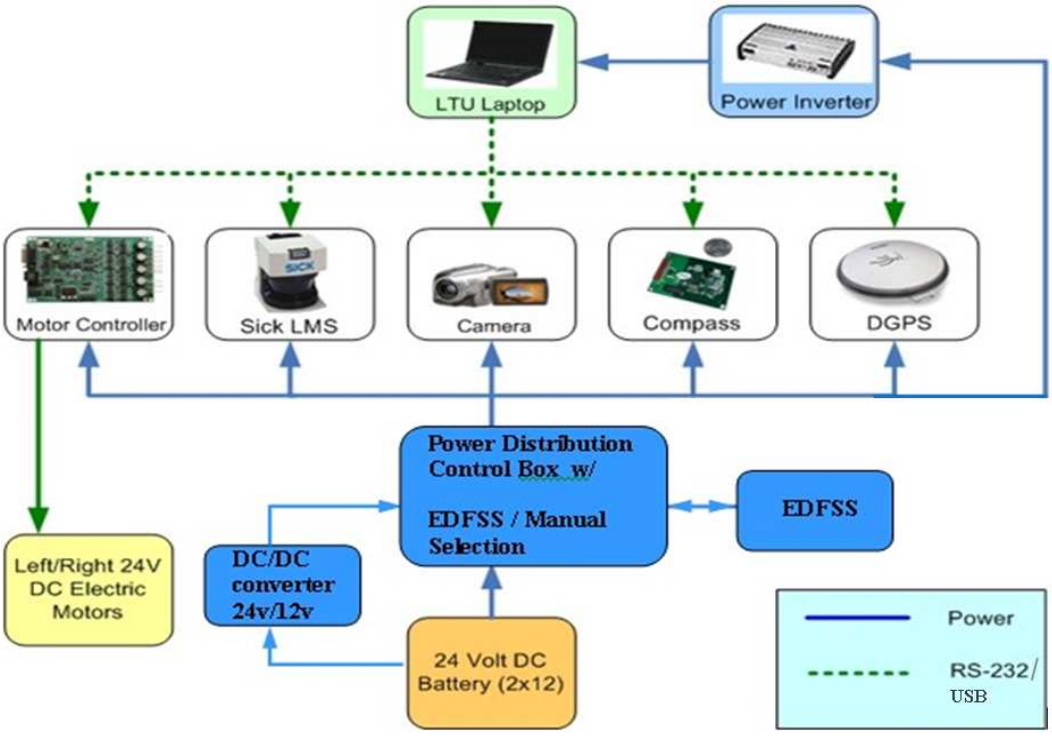

Figure 5: Power and Communications Control System Schematic

5. Software Design

5.1 Software Strategy

C# , XNA, and the Visual Studio 2008 IDE were chosen for all software development on the robot this year. C# is a powerful object oriented language, and Microsoft XNA is a video gaming toolset which allowed Viper II to utilize the processing power of GPU for image analysis. The Visual Studio IDE allowed for rapid application development with easy to build visual interfaces. **Figure 6** shows the high level software architecture design.

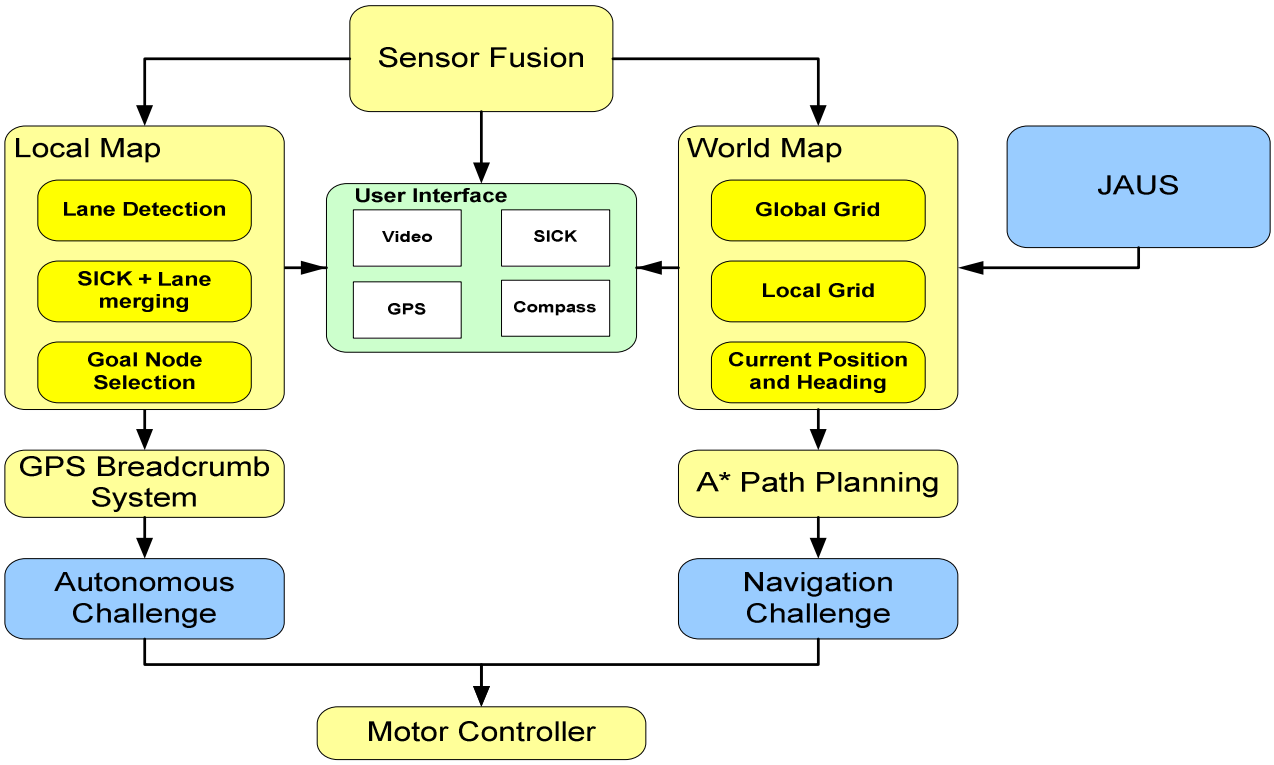

Figure 6: High Level Software Design

5.2 Sensor Fusion

For this year's competition, a Sensor Fusion Module shown in **Figure 7** was developed to collect data from all the hardware sensors, and to present this data to all subscribers in a consistent manner. This allows subscribers to access sensor data without knowing the details of hardware layer implementation.

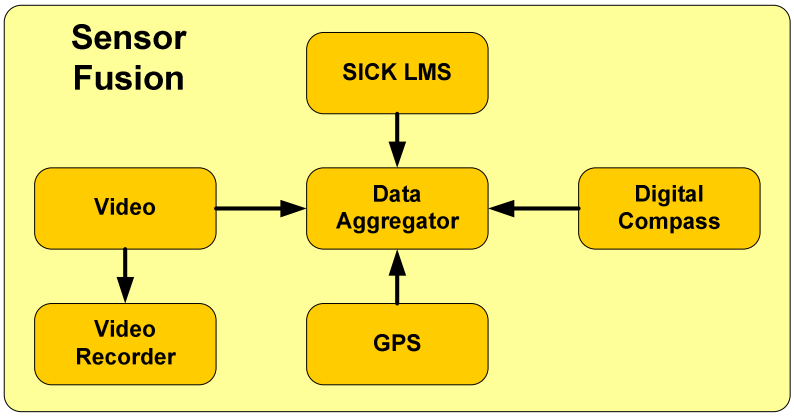

Figure 7: Sensor Fusion

5.3 World Map and Path Finding

The world map plots a geographic region to a Cartesian coordinate plane. The total size of the Cartesian map is determined by the size of the area to be represented and the desired resolution. Generally, a resolution of ½ meter has proven to work well during testing. Each point in the geographic region can be mapped to a cell in the Cartesian map, and vice versa, through the use of a conversion object. In this way, a simple A* path finding algorithm can be used to find the shortest path between any two given points.

5.4 Motor Control Module

The motor controller module was designed to be easily plugged into any of the team's projects. Once initialized, this module can accept simple operation commands for speed and direction, which it then translates into the ASCII communication strings that are sent to the motor controller via a serial connection. This module allows team members to have simple access to the hardware without knowledge of the motor controller's underlying communication protocols.

5.5 Autonomous Challenge Details

The main algorithm for the autonomous challenge consists of using computer vision to identify lane information and SICK data output to detect obstacles. To accomplish this, the camera is first calibrated so that the vision processing can map lanes onto a local map alongside the SICK data, and then a goal node selector algorithm is used to find the appropriate heading. Additionally, a GPS Breadcrumb utility, which uses GPS location information and compass headings to store a list of recently visited locations, is used to detect if the robot has become turned around. This helps the robot avoid going backwards on the course.

5.5.1 Camera Calibration

The camera calibration is responsible for synchronizing the lane data with the SICK obstacle data. It works by calculating the distance of each pixel in the vision's captured image, and it uses the SICK as the reference point. It is accomplished by measuring the distance of predefined positions on the captured image and interpolating the remaining positions on the image. A snapshot of the tool for calibration is shown in **Figure 8**.

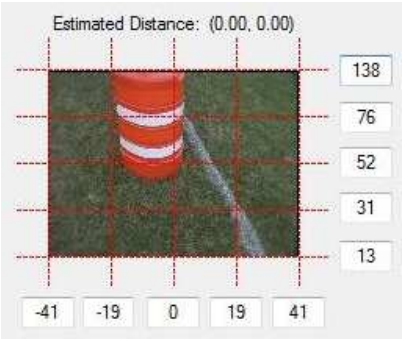

Figure 8: Camera Calibration Tool

5.5.2 SICK Data Gathering

As the SICK data is collected, it is mapped onto a local map, combined with the filtered camera image. Because of the calibration previously, the data can simply be placed in the local map without any extra processing. See **Figure 9**.

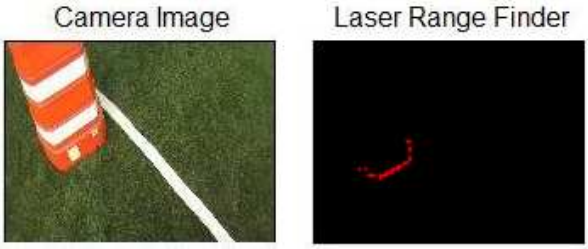

Figure 9: Example of mapping SICK data to the Local Map

5.5.3 Image Processing

For this year's competition, Viper II continues to use a local map approach for combining the vision information with obstacle data. First, SICK obstacle data is added into the local map as red pixels. Then, the camera image is captured and converted to black and white using a brightest pixel filter. This filter runs on the GPU and chooses the brightest pixel on each horizontal line. Then blob detection is used to filter out noise pixels that are not considered to be lanes. Once noise has been removed, the filtered camera image is added to the SICK data and all of the white pixels that are vertically above the red pixels are repainted as black. A Hough Transform filter is then applied to find any partial lanes in the image and fill them in. Finally, a bleeding filter is used to color any pixels vertically above red pixels as red, and above white pixels as white. Thus, the local map represents both the lane information in white pixels and the SICK obstacle data in red pixels. **Figure 10** demonstrates the steps described above.

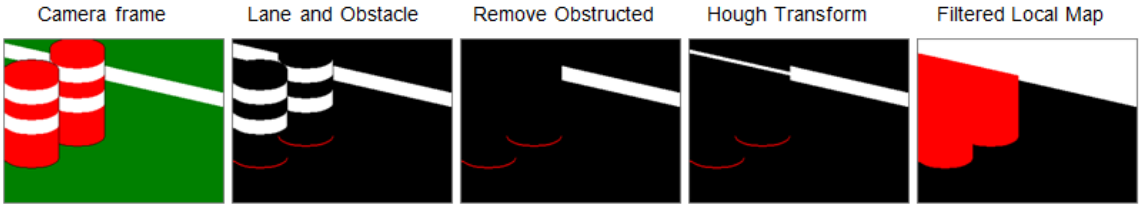

Figure 10: Image Processing Steps

5.5.4 GPU Processing

The results from previous IGVC competitions indicated that the required image processing on a laptop CPU was too strenuous for the hardware. To solve this issue, it was found that the GPU could be used as an alternative for some of the processing. The GPU is exceptional for manipulating image data and therefore is an excellent candidate for fast image processing. XNA was used to communicate with the GPU. Normally, XNA is used for video game development, but it is C# based and was easy to integrate with the existing codebase. Team Viper converted several image filtering algorithms, such as the Brightest Pixel filter, to run on the GPU to achieve dramatic performance improvements.

5.5.5 Hough Transform

A Hough Transform helps Viper II identify lanes in the local map, but it is primarily used for filling in a lane when only a partial (dashed) one exists. See **Figure 11** for an example of the Hough Transform filling in missing data. This transform is applied twice, because the local map is split vertically. The splitting of the local map forces the detection of a lane on both sides, and this is critical in cases where both of the lanes exist on the local map at once. Since a single image does not guarantee the Hough Transform will recognize both lanes, each image must be evaluated for missing data. This lack of recognition becomes extremely apparent when the second lane's intensity is less dominant.

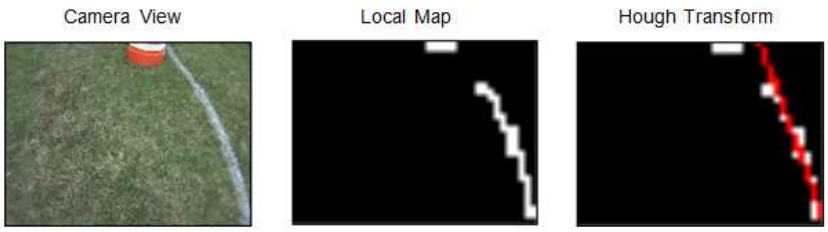

Figure 11: Hough Transform applied to dashed lane

5.5.6 Blob Detection

Blob detection and filtering allows for the elimination of noise pixels from the camera image. As the image is scanned, each pixel is examined to determine its color. Once this process is finished, each contiguous region of color in the image will have a corresponding blob object, and these blobs are useful for several reasons. First, they allow identification of all regions of color and this can help the detection algorithm filter out blobs that are too small (and thus are just noise). Secondly, blobs of "non-obstacle" space can be identified, and this ensures that the detection algorithm selects the goal node that is in the same region of the robot. Thus, the robot will not cross a lane to get to the local goal location.

5.5.7 Lane Following and Obstacle Avoidance

Viper II uses a goal node selection algorithm to find the best path through the obstacle course. This algorithm first finds the largest gap in the image, and then plots a goal node in the center of this gap. If the largest gap on the map is smaller than the Viper II, it retreats and searches for a path that it can fit through. Once the goal node is found, Viper II starts to move in the direction of this node until it processes new image data. Additionally, a new algorithm was added this year to avoid turning in the wrong direction when facing a lane on one side and obstacles on the other side of the robot.

5.5.8 Goal Node Selection

Goal node selection is an important aspect of the autonomous navigation challenge, and it relies on the local map. Once the local map is built, it is then transformed into an image, and a goal location for the robot's heading must be found. This is handled by finding the largest gap between obstacles and lanes. Ideally, this gap is at the top of the image and as far away as possible from the robot. However, if a lane bisects the image, then the gap is calculated from either the left or right side of the image. The images below demonstrate the goal node selection algorithm. The goal location is the orange dot in the images shown in **Figure 12**.

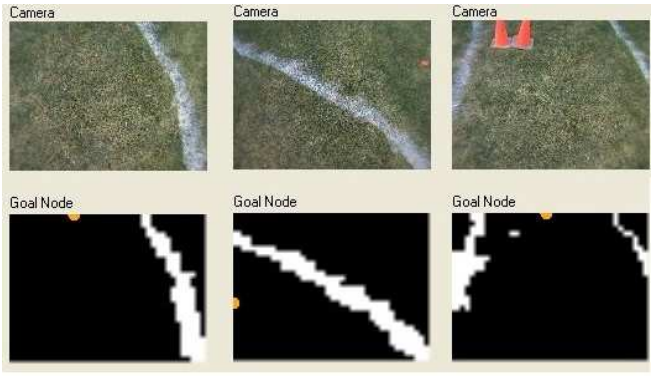

Figure 12: Goal Node Selection Example

To improve decision making when selecting a goal node, the local map is analyzed twice; the first check includes only lane information and the second check includes both lane and obstacle information. Analyzing only the lane information allows the robot to always conclude that it should turn away from the lane. This decision is followed even if there's an obstacle in the direction opposite of the lane. Even if turning causes the Viper II to face an obstacle, it will continue turning away from the lane until it sees a clear path. **Figure 13** shows the two goal node checks, and **Figure 14** shows the decision process to turn away from the lane (blue).

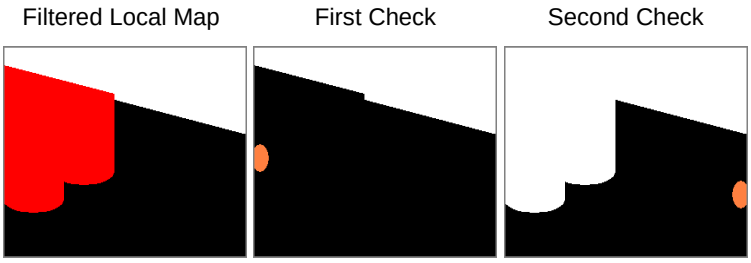

Figure 13: Two Goal Node Checks

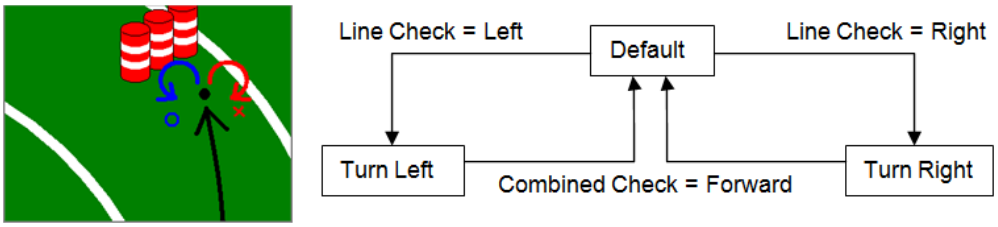

Figure 14: Direction Decision and State Diagram to Escape from a Trap

5.6 Navigation Challenge Details

The primary algorithm of the navigation challenge uses the GPS, compass, and SICK laser range finder sensors in conjunction with dead-reckoning information available from the motor controller (derived from optical encoders on the motor shafts) to move Viper II through the course to the various GPS checkpoints. The GPS waypoint locations are mapped onto a World Map. As Viper II moves through the course and discovers obstacles using the SICK, the location of the obstacles is also mapped. This allows Viper II to adjust its course and navigate around the obstacles to the next checkpoint. Additionally, this year's robot is also using a Fuzzy Controller to smooth out the robots movement through the course.

5.6.1 Waypoints Scheduling

To select the quickest route between waypoints, Team Viper classifies the problem as a Traveling Salesman Problem. For simplicity, Viper II uses a Greedy Algorithm to determine the visiting order. The obstacles detected with the SICK may change the path from the robots' position to the waypoints. Therefore, the robot will reschedule its visiting order when each sub-destination is reached.

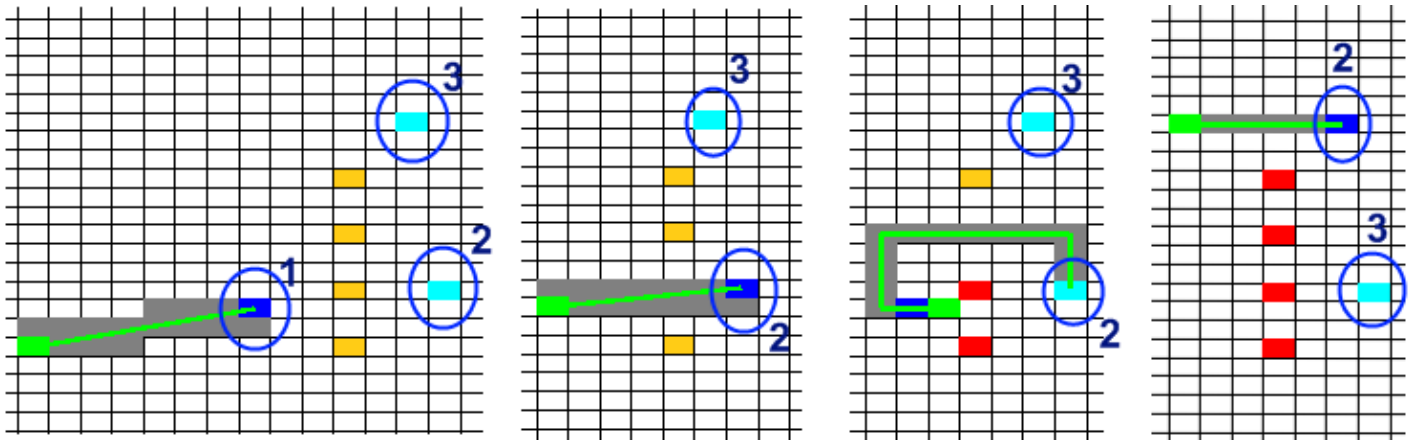

Figure 15: Waypoint scheduler

Light blue boxes represent waypoints to be visited, and dark blue is the current goal waypoint. Red boxes are detected obstacles and orange ones are not yet known.

Figure 15 demonstrates how the Greedy Waypoint Scheduling algorithm allows the Viper II to dynamically adjust its path based on newly discovered obstacle data. After the robot visits the 1st waypoint and starts heading towards the 2nd one, it discovers new obstacles. The obstacle data is mapped onto the World Map. At this point, Viper II still tries to go to Waypoint #2, but finds even more obstacles blocking its path. Finally, as it attempts to avoid obstacles, Viper II finds that it is now closer to the waypoint at the top of the map, so it schedules this waypoint as the second one to visit.

5.6.2. Fuzzy Controller

A fuzzy logic controller was created to help fix the problem with oscillation in the Viper's movement while performing obstacle avoidance in the Navigation Challenge. The controller takes the robot's current and desired direction headings as inputs, and outputs an adjusted heading. The fuzzy logic rules in the controller limit oscillation in the robot's movements by smoothing out the otherwise drastic changes in direction. A smoother path through the course helps Viper II get to all the waypoints faster.

5.7 JAUS Challenge Details

Viper's JAUS implementation has been completely redesigned for IGVC 2009. The intent was to leverage the modularity aspect of JAUS to establish a design that is easily expanded for future requirements. The implementation is comprised of a network of JAUS components working together to fulfill the JAUS Challenge requirements. A Node Manager routes all JAUS message traffic between JAUS components on Viper. All extra-nodal JAUS messages to and from Viper are translated to the Transport Layer and exchanged by a Communicator component. Figure 15 depicts the JAUS network in blue and the associated Viper modules in red.

5.7.1 Process for Learning JAUS

Development of the JAUS network benefitted greatly from experience gained preparing for and participating in last year's IGVC competition. Although the Reference Architecture was well understood by the team, a considerable amount of time was spent studying and trying to understand the new SAE specifications. The team was provided access to the JAUS Validation Tool (JVT) at http://www.usaric.org/JVT. This tool was essential for validating the team's JAUS implementation.

5.7.2 JAUS Integration into the Design

The Viper JAUS components are written in Java while the Sensor Fusion and mission modules are written in C#. A UDP inter-process link provides the interface between the JAUS components and the rest of the Viper software. This design strategy was inspired by the modular architecture of JAUS. Once the UDP link was defined, Viper's JAUS implementation was developed and tested independently of the rest of the software. **Figure 16** demonstrates the JAUS component and how it communicates to the Viper II's Sensor Fusion module.

5.7.3 Challenges Encountered

The principal challenges encountered implementing JAUS center around the new SAE requirements. The SAE specifications introduced by the 2009 IGVC Rules are vastly more complex than the Reference Architecture. The specifications establish many features new to JAUS, including modified and expanded transport requirements, a multi-layered architecture, and service oriented definitions. Specification availability, completeness, and accuracy were also challenges.

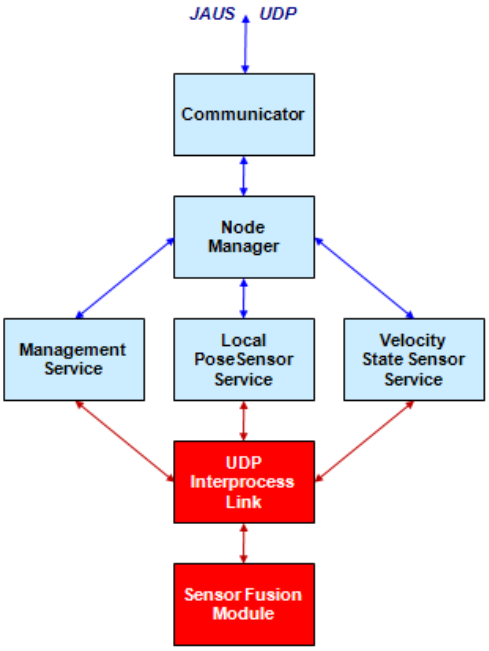

Figure 16: JAUS high level design.

6. Performance Analysis and Estimate

6.1 Safety

Viper II safety features include a manual and remote E-Stop, failsafe brakes, and a capped maximum speed of 5 mph.

6.2 Robot Performance

Vehicle performance was estimated based on the design parameters. These same attributes were measured after integration as shown in **Table 2**.

Attribute	Design Prediction
Maximum Speed	4.2 mph
Climbing Ability	2.5 inch curb
Nominal Power Consumption	500 watts
Battery Operating Time	4 hours
Waypoint accuracy (with Omnistar)	10-20 centimeters

Table 2: Performance Analysis

6.3 Reliability

Viper II utilizes dual modular power sources and a thoroughly tested drive train design that performed well in the 2008 competition. Additionally, the robot uses a rigid aluminum frame with extensive weather-proofing. Furthermore, extensive testing has been done for both the hardware and the software components. Specifically, regression and system testing was done on each module to ensure proper functionality and integration with the hardware components.

6.4 Vehicle Cost Summary

Table 3 summarizes the total material cost for the Viper II vehicle.

Component	Retail Total Cost	Team Cost
CyberPower Laptop	$3,342	$3,342
Sick LMS 291-S05 LMS	$7,000	$0
NovaTel ProPak-LB DGPS & GPS-600-LB Antenna	$4,567	$2,700
24 V NPC-T74 Motors (2)	$648	$648
Panasonic PV-GS500 digital camcorder	$800	$0
Miscellaneous Electrical Hardware	$870	$870
PNI TCM2-20 digital compass/inclinometer	$769	$0
Roboteq AX3500 Dual Channel Motor Controller	$395	$0
Main Battery 12 Volt 50 Ah AGM	$323	$323
Promariner Promite 5/5 Dual 12V Battery Charger	$110	$110
DC-to-DC Converter	$118	$118
EDFSS Hardware Components	$2,308	$780
Hollow Shaft Optical Encoder Kit (2)	$130	$130
Chassis Materials	$320	$320
Miscellaneous Hardware (nuts, bolts, etc…)	$200	$200
14" Tire & Wheel Assembly (2)	$186	$186
Rear 300 Lb Capacity Caster Wheel	$22	$22
Total	$20,931	$6,357

Table 3: Vehicle Cost Summary

7. Conclusion

Viper II continues the LTU tradition of innovation and continuous improvement. Viper II features a distinctive EDFSS system with HMI interface, as well as a backup electrical system, stable camera mount and shock absorbers, and a modular, innovative software design. All software and hardware components are easily swappable and can be upgraded in future designs. Viper II's unique design and state-of-the-art software technology set it apart from other competitors in the 17th annual Intelligent Ground Vehicle Competition.

8. References

[Ward 1994] Ward, A.C., Sobek, III, D.K., "Toyota and Set-Based Concurrent Engineering," 1994 ASME Design Theory and Methodology Conference Proceedings, Minneapolis, MN

[**http://www.jauswg.org**] The Joint Architecture for Unmanned Systems. "Reference Architecture Specification. Volume 2," 2004.

Poppendieck, M., Poppendieck, T. "Lean Software Development: An Agile Toolkit". Addison-Wesley Professional (May 18, 2003)

[**http://en.wikipedia.org/wiki/Hough_transform**] Hough Transform. March 2009.

Nilsson, N. J. , "Artificial Intelligence: A New Synthesis". Morgan Kaufmann Publishers, 1998, San Fransisco, CA.

Lawrence Technological University

Viper II

IGVC 2009 Autonomous Vehicle

Team Members

Gary Givental, Philip Munie, Shawn Ellison, Brandon Bell, Ze Cheng, Taiga Sato, Bryan Koroncey, Nathan Lucas

Faculty Advisor Statement[1]

I, Dr. CJ Chung of the Department of Math and Computer Science at Lawrence Technological University, certify that the design and development on Viper II has been significant and each team member has earned credit hours for their work.

_____ _____

Dr. CJ Chung (chung@ltu.edu, 248-204-3504) Date

[1] Co-advisor: Dr. Lisa Anneberg, Department of ECE

Table of Contents

1. Introduction

For the 2009 Intelligent Ground Vehicle Competition (IGVC), Lawrence Technological University (LTU) presents Viper II, the product of a collaborative effort among a diverse and multidisciplinary group of LTU engineering and computer science students. Viper II is an enhanced robot, with cutting-edge hardware and software updated for 2009. It represents Lawrence Tech's latest venture into developing a safe, reliable, and cost-effective autonomous vehicles that are rich in progressive technologies.

2. Innovations

2.1 Hardware Platform

2.1.1 Mechanical System

Viper II is a three wheel vehicle with front differential drive steering, a rear caster wheel, and an aluminum chassis that are all encased in a custom fiberglass mold. This design provides the most stability and maneuverability, and it allows for a zero-degree turn radius. Furthermore, the vehicle's front suspension system helps stabilize the drive-train and minimizes the shaking of the camera pole.

2.1.2 Electrical System

Viper II uses a manual electrical system that is controlled via a single electric box. There's also hardware in place (a Programmable Logic Controller (PLC) coupled with a Human Machine Interface (HMI) display) that extends the manual system to create the Electronic Diagnostic Fail-Safe System (EDFSS).

2.1.3 Software

The software for Viper II is written in the C# programming language using Visual Studio 2008 and Microsoft XNA. XNA is a set of tools with a managed runtime environment provided by Microsoft that facilitates computer game development, and it allows Viper II to quickly perform image processing on the GPU. Additionally, Viper II includes a Sensor Fusion module to bring sensor information together into one component and an easily pluggable motor control interface for its movement. Moreover, custom Local and World map software help Viper II keep track of lane, obstacle, and GPS waypoint information, and they allow it to be less reactive to the immediate environment. Furthermore, Viper II utilizes a GPS Breadcrumb system to keep track of recently visited locations, and to detect when it is heading in an incorrect direction on the course. Lastly, Viper II software is JAUS level 4 compliant.

3. Design Process

3.1 Collaborative Team Organization

2009 Team members are:

Gary Givental, MSCS – Team Captain	Philip Munie, MSCS – Autonomous Lead
Brandon Bell, MSCS – Hardware Integration Lead	Shawn Ellison, MSCS – Navigation Lead
Nathan Lucas, MSCS – JAUS Lead	Taiga Sato, BSCS
Bryan Koroncey, BSCS	Ze Cheng, BSCS

Since a large team required more communication, team members used an online discussion forum, Subversion code repository, and email to communicate effectively. Weekly meetings were scheduled to allow for status reports and to help resolve emerging issues.

3.2 Project Planning Process

The development for this year's IGVC effectively started in October 2008. **Figure 1** describes the project plan followed by the team. For this year's competition, the team met every week to go through the design plan and to perform integration testing. Since an agile, iterative development process was used, the design emerged over time as different options for the software were explored. This project plan acted as a guideline to keep the team on track, and to make sure core timelines were met.

Task	Due Date
Initial Brainstorming	2/1/2009
File Repository Setup	2/1/2009
High Level Software Design	3/1/2009
High Level Hardware Update Design	3/1/2009
Software Development	5/1/2009
E-Stop Updates	5/15/2009
System Testing	5/15/2009
Written Report	5/18/2009
Oral Presentation	6/7/2009

Figure 1: Project Tasks

3.3 Development Process

The team used agile development and set-based, concurrent methodologies for the software design in this year's competition. The set-based, concurrent development allowed the team to investigate several options for software modules and algorithms, and to find the best solution to design issues. Correspondingly, the agile development approach allowed the team to avoid the well-known pitfalls of the linear "waterfall" style approach. These techniques enabled an emergent, iterative methodology that proved to be the perfect fit and allowed for the functionality and features of the software and hardware to evolve through continuous testing and integration. The team scheduled regular weekly meetings to present status reports, to discuss progress, and to indicate any road-blocks encountered by team members during development. Frequently, students teamed up to work on a particular problem in hardware or software to get it resolved quicker, and to leverage the "pair-programming" concept.

3.4 Testing Process

A test track was setup on LTU campus, which was complete with various kinds of obstacles and curves in the lanes to approximate the challenges of the real competition. This allowed team members to collect real world data and to discover failures in software logic.

3.4.1 Hardware Testing

Viper II's electrical and mechanical systems were tested thoroughly after Viper was showcased to TARDEC, and extensive field testing was performed once the robot was re-assembled. While performing field testing, the team discovered several issues that needed to be addressed. The robot's batteries drained quickly, so a generator was obtained to keep them charged. Additionally, the GPS signal proved to be inaccurate, so the team subscribed to the Omnistar HP service to get 10-centimiter precision.

3.4.2 Software Testing

The team performed extensive unit testing, systems testing, and integration testing of the code throughout development. As new functionality was developed, it was demonstrated to the rest of the team, and, when possible, tested on the robot. Test runs were performed regularly on the test track at LTU.

3.4.3 System Integration

System integration testing was executed as quickly as possible once the development of the core components was complete. When new modules were available for general testing, they were immediately integrated into the overall system and used by all team members for testing. This modular design of the software components made it easy for new functionality to be plugged into the overall architecture. The team extensively tested the integration between the software and all the sensor hardware, as well as the motor control.

4. Hardware Design

4.1 Robot Structure

4.1.1 Chassis

Viper II is a three wheel vehicle with front differential drive steering and a rear caster wheel. This design allows for zero-turn radius maneuverability and simplified motion control. The chassis has a minimum of 4 inches of ground clearance, and this allows the robot to climb hills and ramps with an approach angle of 23 degrees. The 3-D CAD model shown in the follow **Figure 2** illustrates the design of the chassis using the ¾ inch 6063 aluminum tubing, which gives the robot the capability of carrying all of the necessary components, as well as the designated payload required for the competition. The chassis design enables Viper II to maintain its structural integrity throughout the rugged off-road course.

4.1.2 Suspension

Viper II is designed with an independent suspension that absorbs impact in the wheels from the off-road terrain. This design provides ample stability to the camera, thus allowing for a smooth video image feed. The suspension model in **Figure 3** shows the coil spring over a tube style shock that is mounted to independent control arms. This suspension style gives the robot 1 inch of travel in the shocks, and it greatly reduces the twist and roll of the chassis and components. Furthermore, the shocks are rated for 350 lbs per inch and have an adjustable rebound feature.

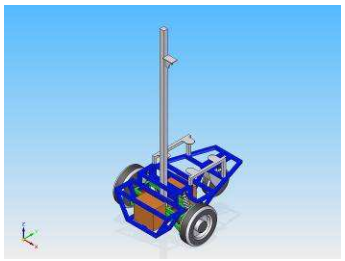

Figure 2: Chassis Model

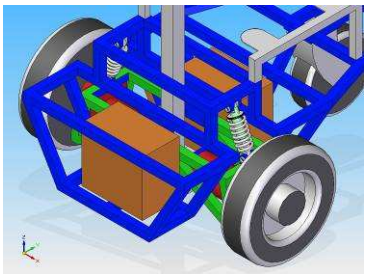

Figure 3: Front Suspension

4.1.3 Drive Train

Viper II's design uses two 24 volt, permanent magnet type model NPC-T74 high torque motors with a 20:1 gear ratio that gives the robot the ability to generate up to 1480 in-lbs (123.3 ft-lbs) of torque and allows for a maximum speed of 10.4

MPH (at 248 RPMs). Thus, these motors give Viper II plenty of power, and they eliminate the fear of premature motor failure. They ensure that Viper II can easily navigate through rough terrain, and go up a 15 degree incline.

4.1.4 Body

The body that is covering and protecting Viper II is fabricated from fiberglass, and its sleek design with smooth flowing curves and lines gives the feel of its namesake. It is designed to give easy access to the internal components through an easily removable top section, and it is fitted with electric fans to ensure that heat is removed from inside the robot. Furthermore, a tray that slides out of the right side of the fiberglass body allows access to Viper II's internal laptop computer, and it can be used without removing the top section.

4.2 Motors and Electronics

4.2.1 Motor Control

Motor control is facilitated by a Roboteq AX3500 60A/channel controller with a C# serial port interface. The AX3500 provides velocity feedback measurements through optical encoders that are mounted on the motor shafts. Additionally, the E-Stop is wired to the controller's main drive power lines via a mechanical relay that can be triggered by the top-mounted kill switch or by the wireless control system.

4.2.2 Sensors

In keeping with the theme of modularity, all of Viper II sensors (and controls) utilize RS-232 or USB 2.0 standard interfaces. **Table 1** summarizes the sensors used in Viper II.

Sensor	Function
Optical Encoders	Motor shaft position feedback for servo control
NovaTel ProPack-LB DGPS Unit	Global positioning
Digital Compass/Inclinometer	Heading, roll, and pitch information
High Dynamic Range DV Camera	Capture field image stream
Sick Laser Measurement Sensor	Provides a 180 degree polar ranging array

Table 1: Vehicle Sensor Summary

4.2.3 E-stop

Viper II is equipped with both manual and wireless (RF) remote emergency stop (E-Stop) capabilities. The Excalibur RS-310 Remote and Start Keyless Entry manufactured by Omega Research is used as the wireless E-Stop component, and its door lock function is integrated into Viper II's E-stop system. When this E-Stop is activated, it removes power from the drive train and engages both of the front wheel failsafe electromagnetic brakes.

4.2.4 Vehicle Alert System (JAUS Horn and Lights Hardware)

Viper II includes an electrical module that can switch relatively large electrical loads (using relays) to the horn and light systems via commands sent over a USB interface. It is utilized by the vehicle's alert system, and it is activated by Viper II's JAUS software.

4.3 Electrical System

4.3.1 Power System

Viper II has two parallel electrical systems: a logic control system and a manual control system. The logic control system consists of a Programmable Logic Controller (PLC) and a Human Machine Interface (HMI), and these components make up the Electronic Diagnostic Fail-Safe System (EDFSS). The manual system consists of toggle switches and fuses that allow the team to physically turn on and off each electrical component. The operating mode can each be selected by a keyed selector switch found on the main system control board.

4.3.2 Electronic Diagnostic Fail-Safe System - EDFSS

The EDFSS provides Viper II with a nearly endless range of abilities for its current design and future design modifications. It makes the use of an external touch screen display that can be programmed to control each component's power and to display status information.

4.3.2.1 External status indicator lights

During previous competitions, the team realized that easily visible power indicators for key component were needed during testing. The EDFSS solves this problem by controlling three indicator lights: the green light specifies all systems are ready, the yellow light signifies that Viper II is operational, but not all systems are responding, and the red light indicates that the system is not operational. Additional indicator lights are also available next to each manual switch on the electrical box. This functionality allows the IGVC team to quickly identify power issues during testing.

4.3.2.2 Manual Control

The manual system is a basic on/off component control. It acts as a backup electrical system if there are problems with the EDFSS system.

4.3.3 Power Source

Viper II is powered by two 12V 50 Ah AGM batteries connected in series to provide a 24V electrical system. It uses a 24/12V DC/DC converter for the onboard 12V systems and an onboard charger with 110VAC interface for restoring battery power. **Figure 4** illustrates how power for the left and right motors is fed directly into the motor controller to maximize the motor power. However, the control power for the motor controller is regulated by the E-Stop and electrical control box.

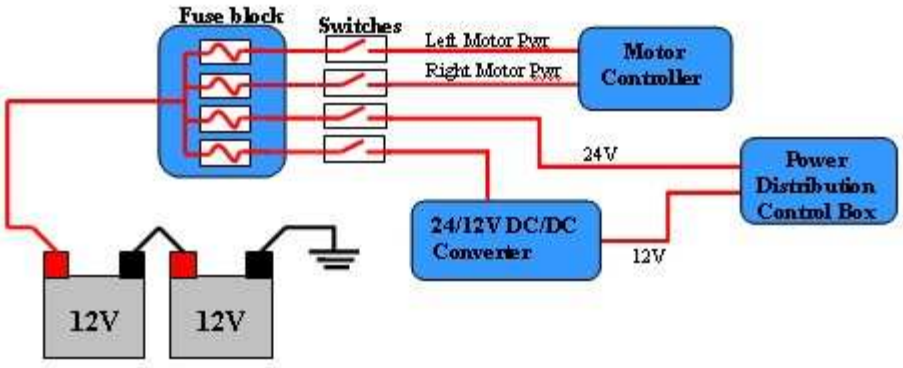

Figure 4: Power Source Schematic

4.3.3 Power Distribution

Viper II is equipped with an all-in-one electrical control box. This power distribution control box contains the systems power distribution and emergency stop printed-circuit board (PCB), JAUS PCB hardware, wireless emergency stop PCB hardware, power control switches for each component, and the main system selector switch that controls the manual and EDFSS operation modes. It is designed to be self contained and removable from Viper II, and it also offers additional testing and diagnostic abilities for system voltage and current ampere measurements.

The two main power connectors are automotive grade EDAC panel connectors, and they allow for an easy connection point for two wire harnesses, which route power to and from each hardware item and the EDFSS block. The power and communications control system schematic for Viper II vehicle is shown in **Figure 5**.

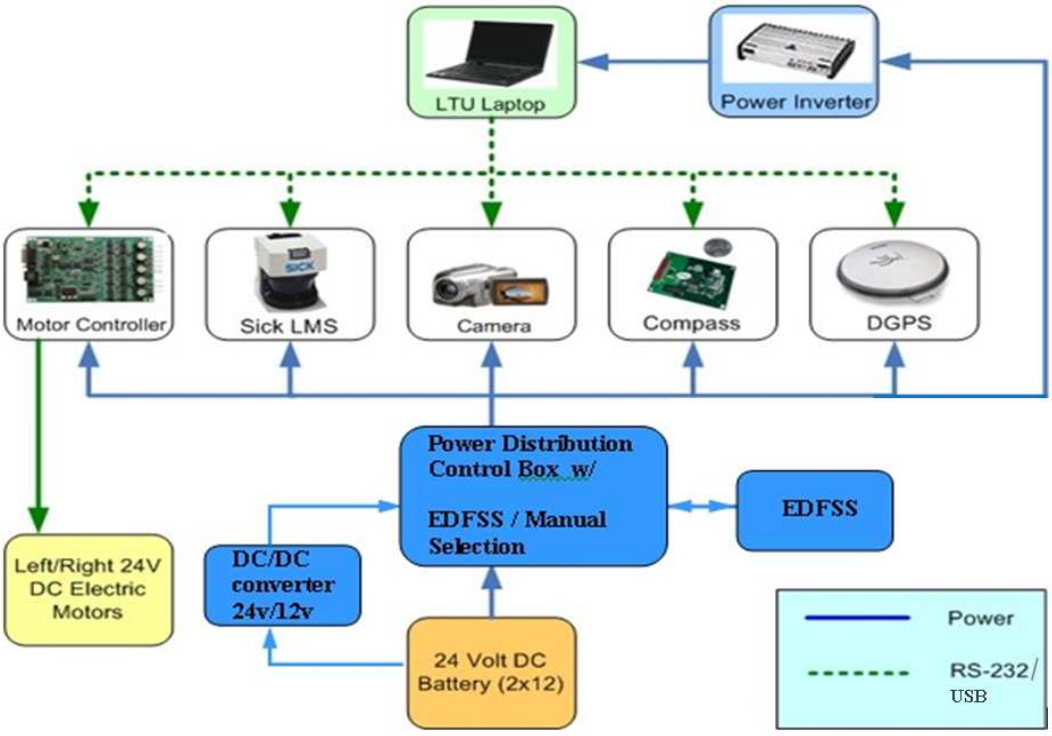

Figure 5: Power and Communications Control System Schematic

5. Software Design

5.1 Software Strategy

C# , XNA, and the Visual Studio 2008 IDE were chosen for all software development on the robot this year. C# is a powerful object oriented language, and Microsoft XNA is a video gaming toolset which allowed Viper II to utilize the processing power of GPU for image analysis. The Visual Studio IDE allowed for rapid application development with easy to build visual interfaces. **Figure 6** shows the high level software architecture design.

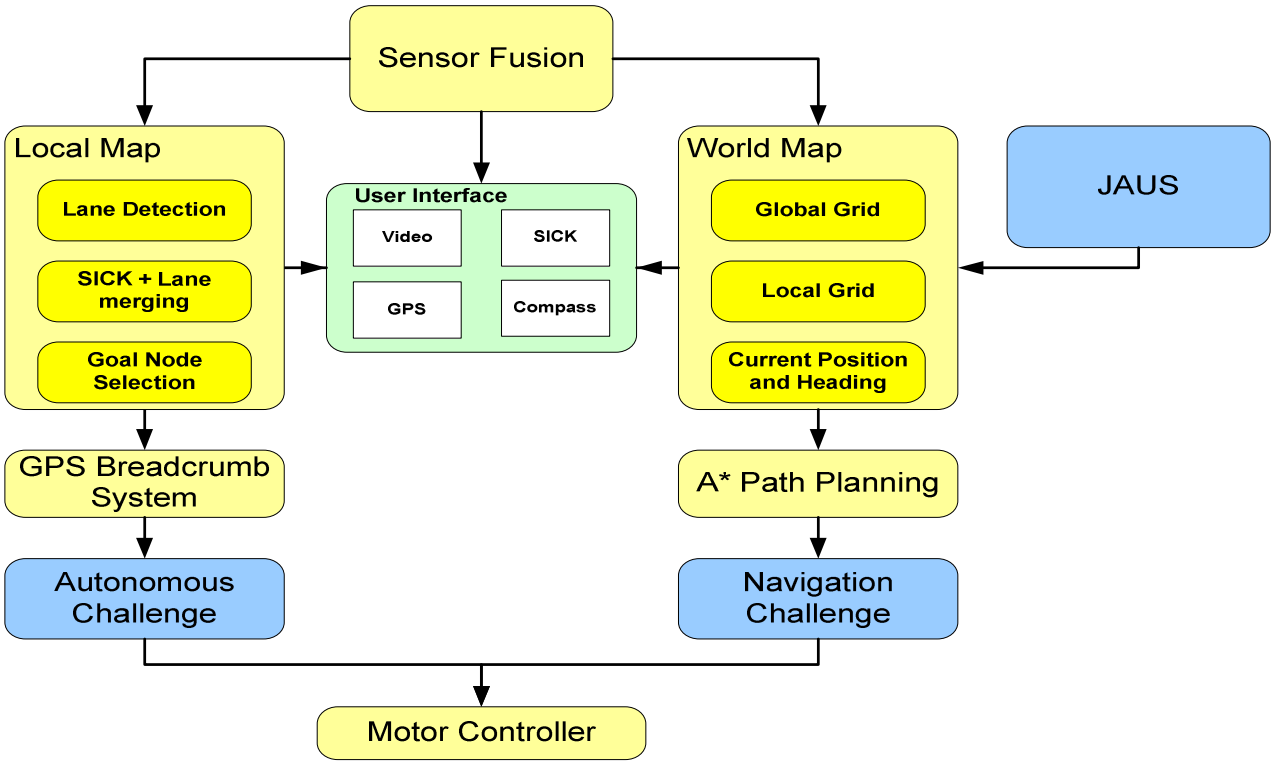

Figure 6: High Level Software Design

5.2 Sensor Fusion

For this year's competition, a Sensor Fusion Module shown in **Figure 7** was developed to collect data from all the hardware sensors, and to present this data to all subscribers in a consistent manner. This allows subscribers to access sensor data without knowing the details of hardware layer implementation.

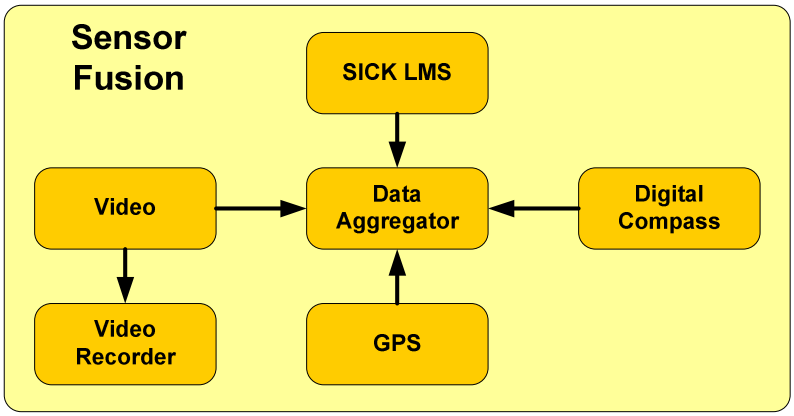

Figure 7: Sensor Fusion

5.3 World Map and Path Finding

The world map plots a geographic region to a Cartesian coordinate plane. The total size of the Cartesian map is determined by the size of the area to be represented and the desired resolution. Generally, a resolution of ½ meter has proven to work well during testing. Each point in the geographic region can be mapped to a cell in the Cartesian map, and vice versa, through the use of a conversion object. In this way, a simple A* path finding algorithm can be used to find the shortest path between any two given points.

5.4 Motor Control Module

The motor controller module was designed to be easily plugged into any of the team's projects. Once initialized, this module can accept simple operation commands for speed and direction, which it then translates into the ASCII communication strings that are sent to the motor controller via a serial connection. This module allows team members to have simple access to the hardware without knowledge of the motor controller's underlying communication protocols.

5.5 Autonomous Challenge Details

The main algorithm for the autonomous challenge consists of using computer vision to identify lane information and SICK data output to detect obstacles. To accomplish this, the camera is first calibrated so that the vision processing can map lanes onto a local map alongside the SICK data, and then a goal node selector algorithm is used to find the appropriate heading. Additionally, a GPS Breadcrumb utility, which uses GPS location information and compass headings to store a list of recently visited locations, is used to detect if the robot has become turned around. This helps the robot avoid going backwards on the course.

5.5.1 Camera Calibration

The camera calibration is responsible for synchronizing the lane data with the SICK obstacle data. It works by calculating the distance of each pixel in the vision's captured image, and it uses the SICK as the reference point. It is accomplished by measuring the distance of predefined positions on the captured image and interpolating the remaining positions on the image. A snapshot of the tool for calibration is shown in **Figure 8**.

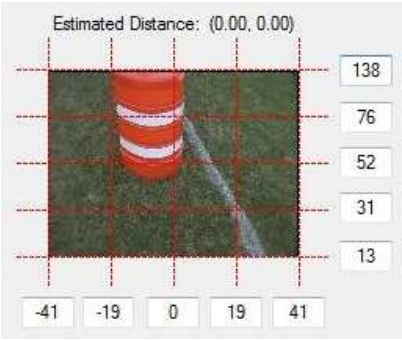

Figure 8: Camera Calibration Tool

5.5.2 SICK Data Gathering

As the SICK data is collected, it is mapped onto a local map, combined with the filtered camera image. Because of the calibration previously, the data can simply be placed in the local map without any extra processing. See **Figure 9**.

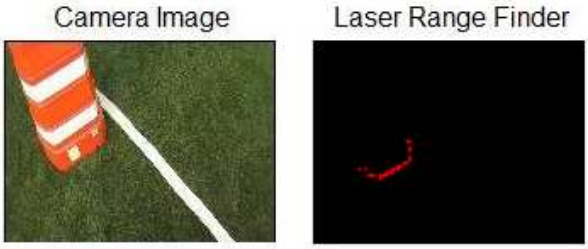

Figure 9: Example of mapping SICK data to the Local Map

5.5.3 Image Processing

For this year's competition, Viper II continues to use a local map approach for combining the vision information with obstacle data. First, SICK obstacle data is added into the local map as red pixels. Then, the camera image is captured and converted to black and white using a brightest pixel filter. This filter runs on the GPU and chooses the brightest pixel on each horizontal line. Then blob detection is used to filter out noise pixels that are not considered to be lanes. Once noise has been removed, the filtered camera image is added to the SICK data and all of the white pixels that are vertically above the red pixels are repainted as black. A Hough Transform filter is then applied to find any partial lanes in the image and fill them in. Finally, a bleeding filter is used to color any pixels vertically above red pixels as red, and above white pixels as white. Thus, the local map represents both the lane information in white pixels and the SICK obstacle data in red pixels. **Figure 10** demonstrates the steps described above.

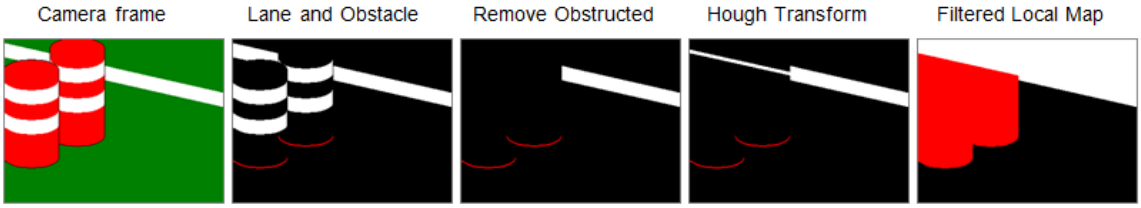

Figure 10: Image Processing Steps

5.5.4 GPU Processing

The results from previous IGVC competitions indicated that the required image processing on a laptop CPU was too strenuous for the hardware. To solve this issue, it was found that the GPU could be used as an alternative for some of the processing. The GPU is exceptional for manipulating image data and therefore is an excellent candidate for fast image processing. XNA was used to communicate with the GPU. Normally, XNA is used for video game development, but it is C# based and was easy to integrate with the existing codebase. Team Viper converted several image filtering algorithms, such as the Brightest Pixel filter, to run on the GPU to achieve dramatic performance improvements.

5.5.5 Hough Transform

A Hough Transform helps Viper II identify lanes in the local map, but it is primarily used for filling in a lane when only a partial (dashed) one exists. See **Figure 11** for an example of the Hough Transform filling in missing data. This transform is applied twice, because the local map is split vertically. The splitting of the local map forces the detection of a lane on both sides, and this is critical in cases where both of the lanes exist on the local map at once. Since a single image does not guarantee the Hough Transform will recognize both lanes, each image must be evaluated for missing data. This lack of recognition becomes extremely apparent when the second lane's intensity is less dominant.

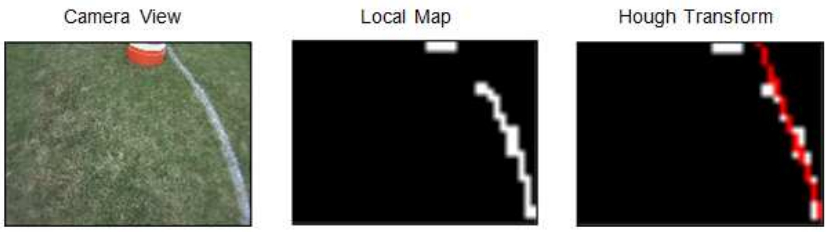

Figure 11: Hough Transform applied to dashed lane

5.5.6 Blob Detection

Blob detection and filtering allows for the elimination of noise pixels from the camera image. As the image is scanned, each pixel is examined to determine its color. Once this process is finished, each contiguous region of color in the image will have a corresponding blob object, and these blobs are useful for several reasons. First, they allow identification of all regions of color and this can help the detection algorithm filter out blobs that are too small (and thus are just noise). Secondly, blobs of "non-obstacle" space can be identified, and this ensures that the detection algorithm selects the goal node that is in the same region of the robot. Thus, the robot will not cross a lane to get to the local goal location.

5.5.7 Lane Following and Obstacle Avoidance

Viper II uses a goal node selection algorithm to find the best path through the obstacle course. This algorithm first finds the largest gap in the image, and then plots a goal node in the center of this gap. If the largest gap on the map is smaller than the Viper II, it retreats and searches for a path that it can fit through. Once the goal node is found, Viper II starts to move in the direction of this node until it processes new image data. Additionally, a new algorithm was added this year to avoid turning in the wrong direction when facing a lane on one side and obstacles on the other side of the robot.

5.5.8 Goal Node Selection

Goal node selection is an important aspect of the autonomous navigation challenge, and it relies on the local map. Once the local map is built, it is then transformed into an image, and a goal location for the robot's heading must be found. This is handled by finding the largest gap between obstacles and lanes. Ideally, this gap is at the top of the image and as far away as possible from the robot. However, if a lane bisects the image, then the gap is calculated from either the left or right side of the image. The images below demonstrate the goal node selection algorithm. The goal location is the orange dot in the images shown in **Figure 12**.

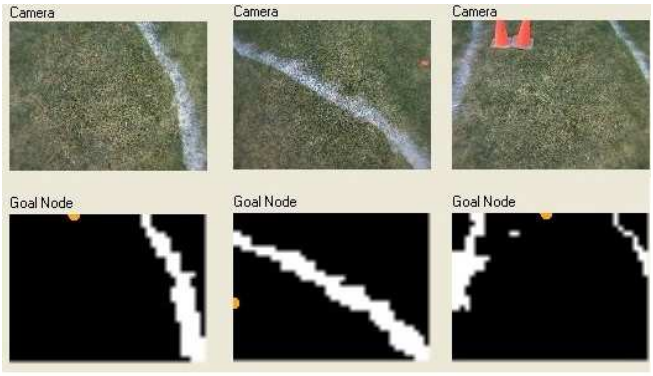

Figure 12: Goal Node Selection Example

To improve decision making when selecting a goal node, the local map is analyzed twice; the first check includes only lane information and the second check includes both lane and obstacle information. Analyzing only the lane information allows the robot to always conclude that it should turn away from the lane. This decision is followed even if there's an obstacle in the direction opposite of the lane. Even if turning causes the Viper II to face an obstacle, it will continue turning away from the lane until it sees a clear path. **Figure 13** shows the two goal node checks, and **Figure 14** shows the decision process to turn away from the lane (blue).

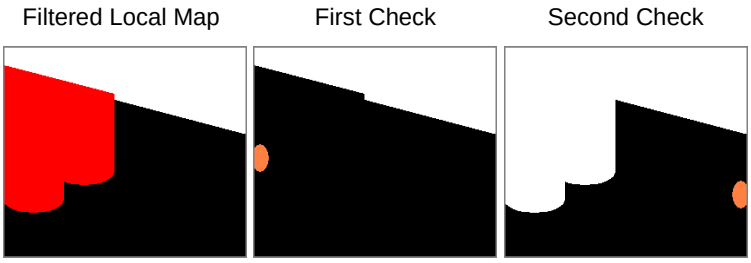

Figure 13: Two Goal Node Checks

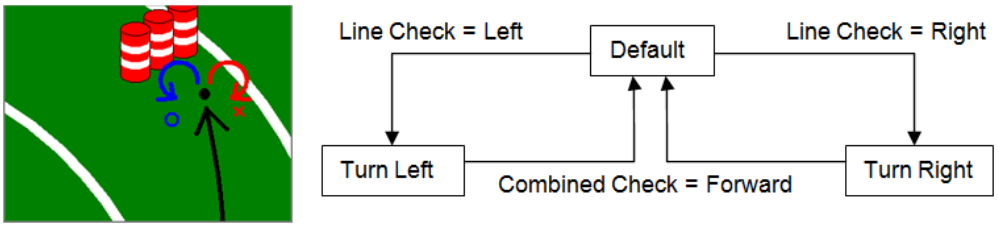

Figure 14: Direction Decision and State Diagram to Escape from a Trap

5.6 Navigation Challenge Details

The primary algorithm of the navigation challenge uses the GPS, compass, and SICK laser range finder sensors in conjunction with dead-reckoning information available from the motor controller (derived from optical encoders on the motor shafts) to move Viper II through the course to the various GPS checkpoints. The GPS waypoint locations are mapped onto a World Map. As Viper II moves through the course and discovers obstacles using the SICK, the location of the obstacles is also mapped. This allows Viper II to adjust its course and navigate around the obstacles to the next checkpoint. Additionally, this year's robot is also using a Fuzzy Controller to smooth out the robots movement through the course.

5.6.1 Waypoints Scheduling

To select the quickest route between waypoints, Team Viper classifies the problem as a Traveling Salesman Problem. For simplicity, Viper II uses a Greedy Algorithm to determine the visiting order. The obstacles detected with the SICK may change the path from the robots' position to the waypoints. Therefore, the robot will reschedule its visiting order when each sub-destination is reached.

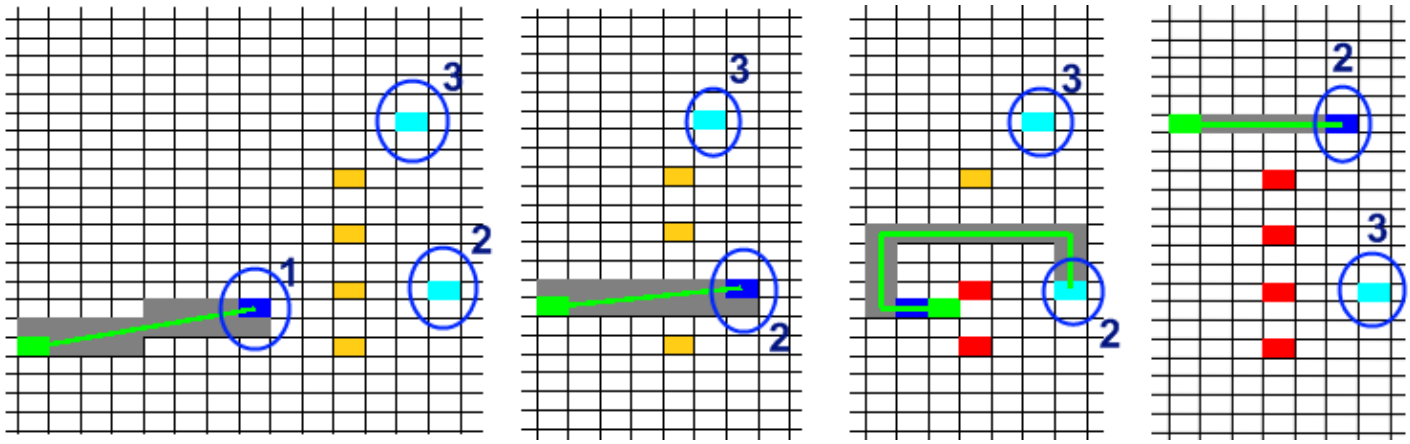

Figure 15: Waypoint scheduler

Light blue boxes represent waypoints to be visited, and dark blue is the current goal waypoint. Red boxes are detected obstacles and orange ones are not yet known.

Figure 15 demonstrates how the Greedy Waypoint Scheduling algorithm allows the Viper II to dynamically adjust its path based on newly discovered obstacle data. After the robot visits the 1st waypoint and starts heading towards the 2nd one, it discovers new obstacles. The obstacle data is mapped onto the World Map. At this point, Viper II still tries to go to Waypoint #2, but finds even more obstacles blocking its path. Finally, as it attempts to avoid obstacles, Viper II finds that it is now closer to the waypoint at the top of the map, so it schedules this waypoint as the second one to visit.

5.6.2. Fuzzy Controller

A fuzzy logic controller was created to help fix the problem with oscillation in the Viper's movement while performing obstacle avoidance in the Navigation Challenge. The controller takes the robot's current and desired direction headings as inputs, and outputs an adjusted heading. The fuzzy logic rules in the controller limit oscillation in the robot's movements by smoothing out the otherwise drastic changes in direction. A smoother path through the course helps Viper II get to all the waypoints faster.

5.7 JAUS Challenge Details

Viper's JAUS implementation has been completely redesigned for IGVC 2009. The intent was to leverage the modularity aspect of JAUS to establish a design that is easily expanded for future requirements. The implementation is comprised of a network of JAUS components working together to fulfill the JAUS Challenge requirements. A Node Manager routes all JAUS message traffic between JAUS components on Viper. All extra-nodal JAUS messages to and from Viper are translated to the Transport Layer and exchanged by a Communicator component. Figure 15 depicts the JAUS network in blue and the associated Viper modules in red.

5.7.1 Process for Learning JAUS

Development of the JAUS network benefitted greatly from experience gained preparing for and participating in last year's IGVC competition. Although the Reference Architecture was well understood by the team, a considerable amount of time was spent studying and trying to understand the new SAE specifications. The team was provided access to the JAUS Validation Tool (JVT) at http://www.usaric.org/JVT. This tool was essential for validating the team's JAUS implementation.

5.7.2 JAUS Integration into the Design

The Viper JAUS components are written in Java while the Sensor Fusion and mission modules are written in C#. A UDP inter-process link provides the interface between the JAUS components and the rest of the Viper software. This design strategy was inspired by the modular architecture of JAUS. Once the UDP link was defined, Viper's JAUS implementation was developed and tested independently of the rest of the software. **Figure 16** demonstrates the JAUS component and how it communicates to the Viper II's Sensor Fusion module.

5.7.3 Challenges Encountered

The principal challenges encountered implementing JAUS center around the new SAE requirements. The SAE specifications introduced by the 2009 IGVC Rules are vastly more complex than the Reference Architecture. The specifications establish many features new to JAUS, including modified and expanded transport requirements, a multi-layered architecture, and service oriented definitions. Specification availability, completeness, and accuracy were also challenges.

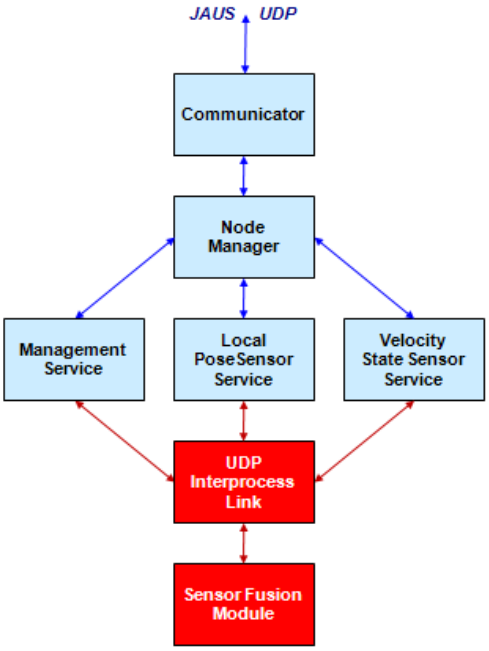

Figure 16: JAUS high level design.

6. Performance Analysis and Estimate

6.1 Safety

Viper II safety features include a manual and remote E-Stop, failsafe brakes, and a capped maximum speed of 5 mph.

6.2 Robot Performance

Vehicle performance was estimated based on the design parameters. These same attributes were measured after integration as shown in **Table 2**.

Attribute	Design Prediction
Maximum Speed	4.2 mph
Climbing Ability	2.5 inch curb
Nominal Power Consumption	500 watts
Battery Operating Time	4 hours
Waypoint accuracy (with Omnistar)	10-20 centimeters

Table 2: Performance Analysis

6.3 Reliability

Viper II utilizes dual modular power sources and a thoroughly tested drive train design that performed well in the 2008 competition. Additionally, the robot uses a rigid aluminum frame with extensive weather-proofing. Furthermore, extensive testing has been done for both the hardware and the software components. Specifically, regression and system testing was done on each module to ensure proper functionality and integration with the hardware components.

6.4 Vehicle Cost Summary

Table 3 summarizes the total material cost for the Viper II vehicle.

Component	Retail Total Cost	Team Cost
CyberPower Laptop	$3,342	$3,342
Sick LMS 291-S05 LMS	$7,000	$0
NovaTel ProPak-LB DGPS & GPS-600-LB Antenna	$4,567	$2,700
24 V NPC-T74 Motors (2)	$648	$648
Panasonic PV-GS500 digital camcorder	$800	$0
Miscellaneous Electrical Hardware	$870	$870
PNI TCM2-20 digital compass/inclinometer	$769	$0
Roboteq AX3500 Dual Channel Motor Controller	$395	$0
Main Battery 12 Volt 50 Ah AGM	$323	$323
Promariner Promite 5/5 Dual 12V Battery Charger	$110	$110
DC-to-DC Converter	$118	$118
EDFSS Hardware Components	$2,308	$780
Hollow Shaft Optical Encoder Kit (2)	$130	$130
Chassis Materials	$320	$320
Miscellaneous Hardware (nuts, bolts, etc…)	$200	$200
14" Tire & Wheel Assembly (2)	$186	$186
Rear 300 Lb Capacity Caster Wheel	$22	$22
Total	$20,931	$6,357

Table 3: Vehicle Cost Summary

7. Conclusion

Viper II continues the LTU tradition of innovation and continuous improvement. Viper II features a distinctive EDFSS system with HMI interface, as well as a backup electrical system, stable camera mount and shock absorbers, and a modular, innovative software design. All software and hardware components are easily swappable and can be upgraded in future designs. Viper II's unique design and state-of-the-art software technology set it apart from other competitors in the 17th annual Intelligent Ground Vehicle Competition.

8. References

[Ward 1994] Ward, A.C., Sobek, III, D.K., "Toyota and Set-Based Concurrent Engineering," 1994 ASME Design Theory and Methodology Conference Proceedings, Minneapolis, MN

[**http://www.jauswg.org**] The Joint Architecture for Unmanned Systems. "Reference Architecture Specification. Volume 2," 2004.

Poppendieck, M., Poppendieck, T. "Lean Software Development: An Agile Toolkit". Addison-Wesley Professional (May 18, 2003)

[**http://en.wikipedia.org/wiki/Hough_transform**] Hough Transform. March 2009.

Nilsson, N. J. , "Artificial Intelligence: A New Synthesis". Morgan Kaufmann Publishers, 1998, San Fransisco, CA.