PARADROID

Presented by the
IEEE Robotics Team
University of Wisconsin-Madison

Faculty Advisor Statement

I certify that the engineering design of the robotic vehicle described in this report, Paradroid, has been significant and equivalent to what might be awarded credit in a senior design course.

THE UNIVERSITY of WISCONSIN MADISON

Professor Michael Zinn
Department of Mechanical Engineering

Table of Contents

1 Introduction

The University of Wisconsin-Madison IEEE Robot Team is pleased to re-introduce Paradroid to the 17th Annual Intelligent Ground Vehicle Competition. After not qualifying at last year's competition, we have spent an entire year modifying and improving Paradroid, including rebuilding several systems from the ground up. The goal of this project is to go beyond the challenge of the competition and design a versatile and adaptable platform that is useful for other applications as well.

The UW-Madison IEEE Robot Team is a group entirely of undergraduate students studying engineering and computer science and who are interested in robotics. We meet several times a week outside of class working on both Paradroid and outreach projects to teach area students about both robotics and engineering. All of our approximately 25 members are volunteers, and none of us receive course credit for this project.

2 Innovations

This year Paradroid was significantly updated from last year's design to include many new improvements. These design changes all stemmed from observations of problems with the previous design.

2.1 Mechanical Innovations

The most annoying pitfall of Paradroid's mechanical design was its height. Though it is necessary to elevate the cameras for a reasonable field of view, this height made transportation difficult. A few groups in the Madison area requested demonstrations of Paradroid's capabilities but we were unable to fit it in a car or van for transportation. For this reason, the upper frame and mounting structure was redesigned so it can be quickly and easily lowered to solve this problem.

Last year, Paradroid had only a partial suspension system that lacked adequate damping. This year we designed and manufactured our own viscous damper and coil spring suspension system. This gives much smoother operation overall due to more continuous contact with the ground on rough terrain and less oscillation after impacts.

2.2 Electrical Innovations

Paradroid's previous electrical systems had a several shortcomings. The main concern was the wireless emergency stop system, which had relatively limited range and did not operate reliably. The wireless E-stop has been completely redesigned to feature a more powerful and robust transmitter to increase the overall safety of the robot. Paradroid also has a new battery monitoring system that allows the remaining battery capacity and run-time to be determined easily. The previous system did not have a battery monitor, which led to the battery being overly discharged frequently.

2.3 Software Innovations

Because the codebase from the previous year was discarded, the software team needed to do additional design work for components based on a new framework. The team spent several weeks studying the framework's design before formally defining the required components and how they would interact. While the components themselves were mostly independent of each other, their eventual intended interaction required the team to cooperate closely, and the

actual implementation was done on a collaborative basis. At the same time, the isolation allowed components to be tested independently of each other, simplifying testing due to reduced potential conflicts. Each developer conducted some independent research in their chosen area, taking advantage of efforts by third parties to increase their own understanding and perhaps find new tricks that could be incorporated into their code. Once the various pieces neared completion, more extensive testing was done in concert to ensure they could cooperate with each other.

3 Design Process

The development process for Paradroid began in early September of 2007. The design process began with a discussion of the flaws of our performance at last year's competition. We decided that in order to address many of these shortcomings we should focus on the basics. Additionally, we realized that our robot performed well from a mechanical and electrical standpoint, but the software was not ready for the competition. We decided to return to the team-developed software platform used in previous years, which an entirely new group of team members has now almost entirely rewritten for Paradroid. The team spent an estimated 5000 hours over a period of nine months redesigning and improving Paradroid.

3.1 Team Structure

The UW-Madison IEEE Robot Team is a student organization comprised entirely of volunteers. The team mainly consists of undergraduates from various engineering disciplines and the computer science department. The team is broken up into three sub-teams: mechanical, electrical, and software, as shown in Figure 1. Leaders for each team were selected based on past involvement and experience level. An all team meeting is held approximately once a month to better facilitate communication between the sub-teams. Most major design decisions are made by a consensus of team members. When there were disagreements, all options were further researched and each party involved presented the advantages and disadvantages of their proposals to their sub-groups. If a consensus still could not be reached, the decision was put to a vote.

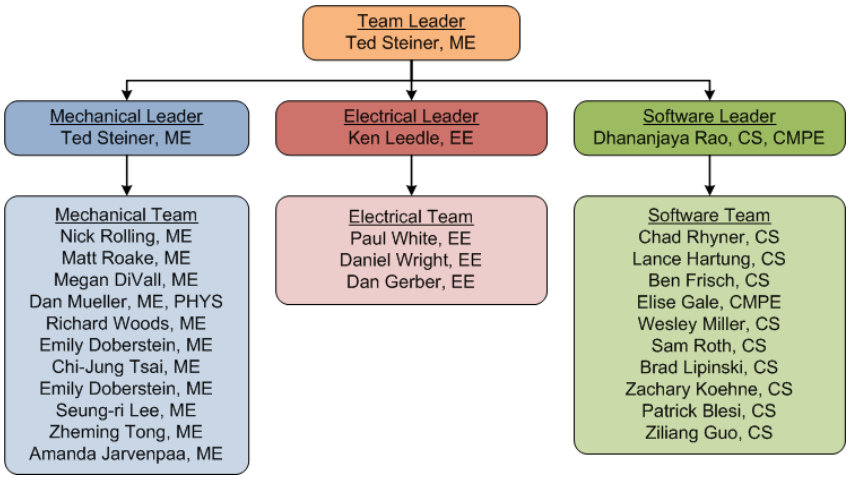

Figure 1: Team Organization

3.2 Team Development

This year the team had only four returning members, which offered an easy opportunity to restructure the team and adapt a more sustainable strategy. In previous years, active team members have been almost exclusively juniors and

seniors who have already taken several classes in relevant subject areas. A much more active approach to recruitment was taken this year, largely focusing on attracting freshmen and sophomores who could remain on the team for three to four years rather than one or two. Within a month we increased our ranks to over 30 new members and the majority of them have been active on the team the entire year. Most of these new members had little to no experience with the majority of team tasks, so we focused almost exclusively on hands on training and guided group projects for the first few months. Now, by the end of the year, we have developed many new "specialists" in topic areas ranging from drivetrains and suspensions to embedded circuit boards and vision processing.

3.3 Project Planning

The planning process began at the first meeting of the year, where Paradroid's shortcomings and inability to qualify were discussed, which led into the larger discussion of how we could improve Paradroid overall. Several new features of Paradroid were discussed and then those agreed upon were ranked according to priority and given a page on our team wiki, where their progress could be easily recorded. Each feature was broken down into smaller tasks, which broke the seemingly daunting projects down into manageable sections that could be easily accomplished by the new members to the team. As each of these small tasks was completed, the group discussed what they liked and did not like about the project before moving on to a new project.

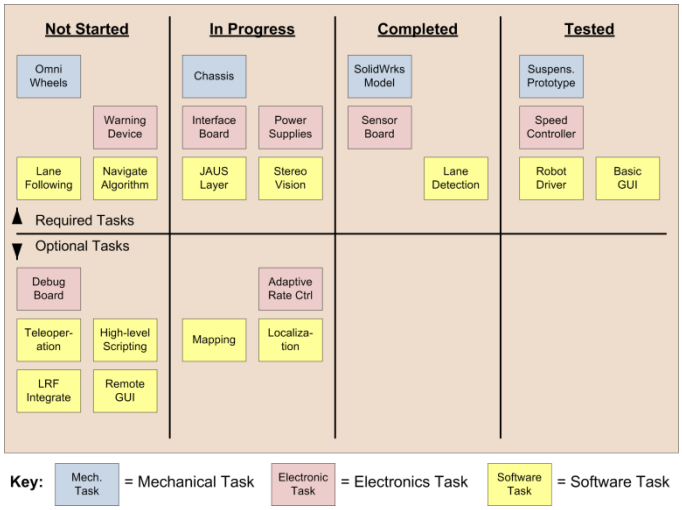

Figure 2: Sample Project Planning Board

Each of these projects was also written down on a note card, color coded by team, and posted in either the "not started," "in progress," or "completed" area of a bulletin board in the team office. The advantage of the note card task tracking system was that it allowed the interdependence of different features and tasks to be easily understood. As the year progressed, optional tasks lower in priority were whittled down to those that could be completed in the remaining time and added the most value to the robot. In the end, this tracking system worked quite well for the team because it was always up to date, as opposed previous efforts to use software programs to track our progress that were much more time intensive.

3.4 Development

The mechanical, electrical, and software sub-teams each utilized their own development processes suited to their task requirements. The mechanical team, whose work consisted of mostly hardware design, used a stricter phase-based development process. Conversely, the software team utilized the agile methodologies to allow for easier adaptation to the changing scope of their projects. The electrical team, whose projects involved both hardware and software design, used a combination of both development processes.

The mechanical team's development cycle consisted of computer-aided design, prototyping, production, and testing phases. SolidWorks, a computer aided design program widely used in industry, was used to model each component

in the vehicle. By using SolidWorks, many ideas could be visualized very quickly without cost, and components could be tested for interferences and proper interaction before they were built. From time to time, experts both on campus and at companies around the world were contacted about how to best solve a specific design issue in the most efficient and effective way. After designs were completed and tested on a computer, prototypes were often built for proof-of-concept testing. If the prototypes worked, the designs were finalized and parts were manufactured in house.

The electrical team followed a similar development process for their hardware design, using computer aided design and prototyping whenever possible. Custom boards were designed using EAGLE, a computer aided design printed circuit board layout tool. The embedded boards were prototyped using breadboards before committing the designed to a printed circuit board.

The software team carried out much of its development using pair programming techniques. This reduced the amount of debugging needed and resulted in easier to read code. Pairs worked on individual components and unit tests for the components. When unit tests passed, they moved on to testing their components with other components. The software team also focused on producing working revisions of software whenever possible. The use of a modular software framework made this relatively easy, because nonfunctioning components could be kept in the root of the versioning repository without being included in a build.

4 Mechanical Design

Paradroid was designed to be a rugged, reliable, and safe unmanned vehicle. The main goal of the mechanical design is to provide a versatile platform that is capable of traversing all obstacles in the competition and that the software and embedded systems can be easily tested upon and interfaced with. It embraces modular design for reusability and upgradeability, with several independent compartments that can be accessed simultaneously. Team members designed and manufactured all of the parts using SolidWorks and the student shop on campus.

4.1 Drivetrain

Paradroid is a differentially steered, four wheel drive robot powered by two 24V DC wheelchair motors rated at 0.9HP continuous duty. These two motors each power two wheels with chains, and allow Paradroid to reach a maximum speed of 5.7mph, which is limited to 5mph by the control software. This allows Paradroid to maintain fast, yet controllable speeds on rough terrain.

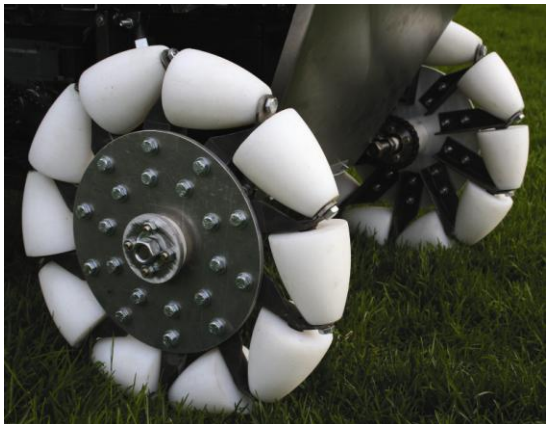

Figure 3: Omni-directional wheels reduce lateral friction for more efficient operation.

The front wheels are 16-inch pneumatic tires and the rear wheels are 16" custom-built omni-directional wheels. The combination of pneumatic tires and omni-wheels allows Paradroid to combine the traction and terrain handling abilities of a four-wheel drive vehicle with the turning efficiency of a two-wheel drive vehicle. Each omni-wheel consists of 10 circumferential lateral rollers that allow the wheels to transmit torque while also rolling side to side, and the large 1.4" to 3"

variable diameter of these rollers keeps the overall shape as circular as possible. This wheel configuration greatly reduces power consumption by reducing turning friction, the main drawback with a typical four wheel drive differential steering. As shown in Table 1, using two omni-wheels reduces the power consumption in zero-radius turns by 73 percent, allowing the vehicle to run longer on an equivalent battery.

Table 1: Comparison of omni and standard wheel power requirements

Wheel	Movement Type	Torque	Power
Omni-Wheel	Zero Radius Turn	38 N-m	480 W
Standard Wheel	Zero Radius Turn	143 N-m	1810 W

4.2 Suspension

To reduce undesirable oscillation this year, Paradroid's suspension was entirely redesigned and rebuilt. The original suspension consisted of rod cylinder assembly with two nested springs. One spring was longer with a low spring constant; the second was a shorter spring with a higher spring constant to add additional response when the cylinders approached their maximum extension. This original system had poor response and return when acted upon by force inputs. The poor response resulted in some oscillation, affecting the camera input during operation.

Through dynamic system analysis, theoretical models of a new suspension were made. One model ignored the spring constant of the wheel, as would be found in Paradroid's Omni-Wheels, while another took into account the natural springiness of the pneumatic tire. Both models also contained the theoretical spring and damping effects we would be adding to the suspension. Compression tests were done on the springs and plotted using Excel to find the spring coefficient that will be acting in the suspension. Using the quarter car model, spring coefficients, and the equations determined from the system analysis, MatLab was used to run a simulation with force impulse and step inputs. Figure 4, below, shows an example response for one such model. From these models, we were able to estimate the amount of damping necessary to achieve a smoother, quicker response than the original suspension.

SolidWorks designs were made for the new suspension, incorporating seals to allow the cylinder to be filled with

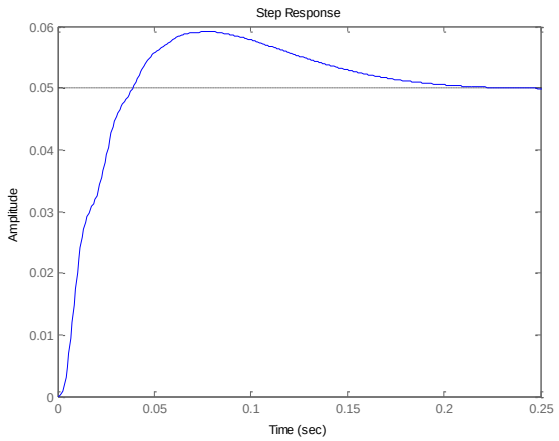

Figure 4: MatLab computer model of suspension response to a step input.

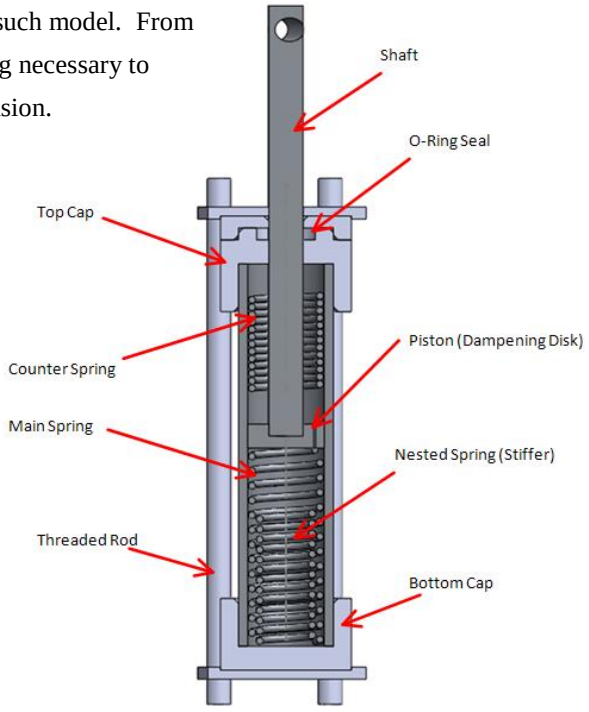

Figure 5: Schematic of the new suspension element, including a spring and damper system

fluid. A nested spring system incorporates multiple levels of spring coefficients, and the cylinder is filled with oil for damping. Inside the cylinder is a moving, piston-like assembly with small holes in the face. Fluid moving through these holes causes viscous damping. An additional spring was added above the piston disk to return the suspension to an equilibrium position faster. The greatest challenge with this project was sealing the sliding rod attaching to the frame, as this seal must be precise to prevent leaking and excessive stress on the shaft.

4.3 Chassis and Body

Paradroid's construction consists of modular compartments built around a sturdy steel frame, which provides strength and durability while also being resilient to impacts and vibrations. Sheet metal covers the outside providing inexpensive and lightweight protection. The compartmentalized design allows for easy access, removal, and augmentation of the four main compartments: embedded, processing, battery, and sensors.

The embedded compartment, located in the middle of the deck, houses the motor controller, power supplies and boards for interfacing with all of the robot's sensors. The processing compartment is located on the rear of the deck. This compartment has two doors that allow unobstructed access to the onboard custom-built computer. The open bottoms of these two compartments allow air to circulate between them and the battery compartment and make routing wires and cables much simpler.

The battery compartment is located in the center bottom of the vehicle to provide a low center of gravity and balanced wheel traction. A charging port integrated into the rear of the robot allows the batteries to be easily charged while inside the vehicle. The sensor module above the robot houses the stereovision camera, GPS sensor, digital compass, yaw rate sensor, emergency stop, and wireless router. The last sensor, a laser range finder, is attached underneath the front of the robot.

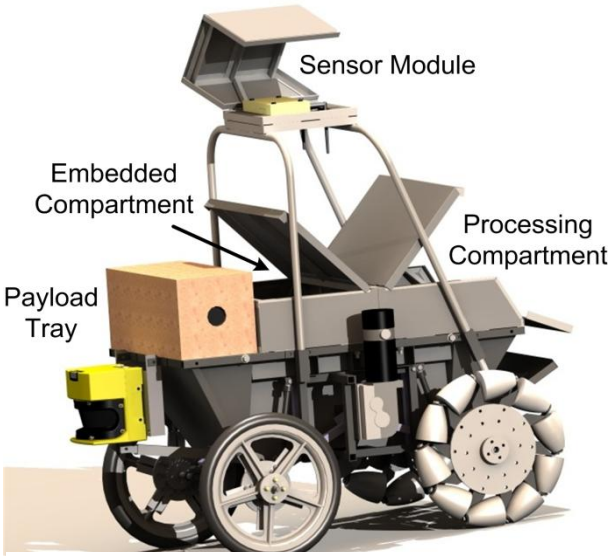

Figure 6: Compartmentalized body design allows simultaneous modification of separate modules and supports future upgrades.

The riser frame, used to hold the sensor module, was redesigned this year to make transportation easier. The sensor module stands 4 feet off the ground. A fixed riser frame made it difficult to transport because the robot was too tall to fit in a van, such as those we take to area schools for demonstrations and to the competition, itself. With the new collapsible design, the sensor module can now be folded down within minutes so that it sits on the front of the robot. This new design provides an opportunity for easy future upgrades, such as running the robot in "short" mode.

5 Electronics Design

Paradroid's electrical systems are designed to provide a simple interface between the main processor and the mechanical systems, delivering efficiency and expandability without compromising features or safety. The electrical systems handle all low-level sensor interfacing as well as providing power to all system components. The

main processor interfaces with the core of the embedded system, a 32-bit Atmel Linux processor, over a TCP/IP connection, enabling high-speed data transfer and providing maximum flexibility. The 32-bit Atmel processor relays commands from the main processor to peripheral boards and provides sensor data and system status updates to the main processor. The embedded Linux processor has a number of interfaces for maximum flexibility between different peripheral devices, including I2C, SPI, RS-232, RS-485, and discrete digital I/O.

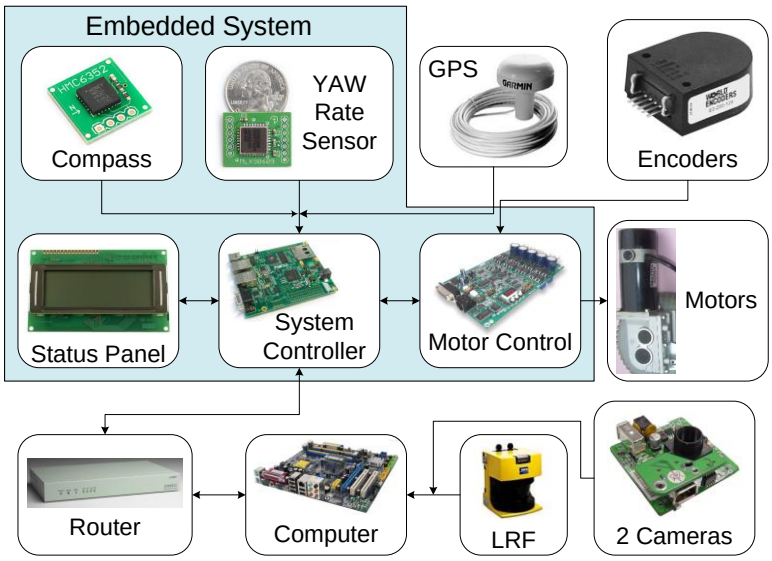

Figure 7: Electrical system diagram

5.1 Battery Monitoring System and Status Panel

Paradroid features a new battery monitoring system that continuously analyzes the vehicle's power usage and can display current battery capacity and estimate the vehicle's remaining run time. This information is displayed on a character LCD screen for easy monitoring, along with other important system parameters from the embedded Linux processor. The status panel also has five green/red light emitting diodes that display the go/no-go status of processes on the embedded Linux processor.

5.2 Electrical Safety Features

Paradroid has a new wireless emergency stop system with improved range and reliability compared to the old system. The new remote E-stop has a 1" emergency stop switch and 1000' effective range. The E-stop directly shuts off power to the motors, bringing the vehicle to a complete stop from full speed in less than two feet of travel. In the unlikely event of a major electrical failure, electromagnetic parking brakes automatically engage to bring the vehicle to a halt in less than 1 foot of travel. Paradroid also has a newly designed warning light system that is more robust than its predecessor was. The warning light system can also operate an optional 110dB air-horn for both visual and audible warnings. Both the warning light and air-horn are controllable through JAUS commands.

5.3 Sensors

The embedded Linux processor gathers data from the Global Positioning System (GPS) receiver, compass, yaw-rate sensor, and wheel encoders and then relays that information to the main processor. The main processor then uses a

Kalman filter to determine the robot's position and heading. The yaw rate sensor provides accurate feedback during turns and corrects for odometry errors and compass lag for accurate heading determination. The GPS receiver is accurate to within 0.5 meters. Corrections to the GPS position using wheel encoder data and vehicle heading provide internal position accuracy within 10cm.

Other sensors on the robot include two Unibrain Fire-i Digital Board Cameras and a SICK PLS101-112 laser range finder that interface directly with the main processor. The cameras provide a color 640x480 VGA picture at up to 30fps over a IEEE1394 (Firewire) interface. The cameras have a field of view of 42.25 degrees, which provides an effective sensing range of 5 meters using vision data. The laser range finder provides a 180° single-plane sweep of the area in front of the robot with 0.5° angular and approximately 70 mm radial resolution over an RS-232 connection. The laser range finder is restricted to only sensing obstacles within 8 meters of the robot.

5.4 Main Processor

Paradroid uses a mini-ATX computer as its main processor because it provides much more processing power than an equivalently priced laptop. This allows Paradroid to quickly analyze sensor data and react to changes in its environment. Paradroid's typical reaction time is about 200 milliseconds, which corresponds to a vehicle movement of 17.6 inches at 5 miles per hour. The main processor is a custom-built computer outfitted with a 2.5GHz Intel Core 2 Quad Processor, 4GB DDR2 RAM and 12GB of solid-state memory. Solid-state memory was chosen for its increased robustness over conventional hard drives. Additional data can be stored on USB flash drives during testing. The main computer has a removable 15" LCD monitor and mini-keyboard for adjustment on the fly. However, a wireless router provides a remote interface to operate the robot, so the detachable display is only used during testing and debugging.

5.5 Power System

Paradroid's power system is designed to maximize vehicle run time. Power is derived from two 12V deep-cycle lead acid batteries that form a 24V nominal battery pack with 75AH capacity. This battery system provides power for up to three hours of operation under normal conditions and up to ten hours in standby mode. The long battery life and integrated charging port allow Paradroid to run nearly continuously. In addition, depleted batteries can be replaced in less than one minute to maximize run time. Power conversion using DC-DC converters provides 24V, 12V, and 5V power to the various systems on the robot at 85-90% efficiency. These converters incorporate over voltage, under voltage, short-circuit, and electrostatic discharge protection, making the power system robust under a wide variety of conditions. Paradroid also has an optional 40W headlight for operating the vehicle in low-light conditions.

Table 2: Power System Requirements

	Normal Operating Conditions			Worst-Case Conditions		
Device	Volts	Amps	Watts	Volts	Amps	Watts
Atmel NGW100	12	0.25	3	12	0.25	3
Linksys WRT54G Router	12	0.4	4.8	12	0.4	4.8
Garmin HVS17 GPS	12	0.1	1.2	12	0.1	1.2

Fire-i Digital Board Cameras (2)	12	0.25	3	12	0.25	3
Compass and Yaw Rate Sensors	5	0.025	0.125	5	0.025	0.125
Sick PLS101 LRF	24	0.8	19.2	24	1	24
Main Processor DC-DC supply	24	3	72	24	4.5	108
Wheelchair Motors (2)	24	10	240	24	120	2880
		Total Amps	Total Watts		Total Amps	Total Watts
		14.825	**343.325**		**126.525**	**3024.125**

6 Software Design

The primary goal of our software is to be a training platform for new and inexperienced team members, while meeting performance requirements. As such, our general approach was to bring together simple strategies to handle the various aspects of autonomous navigation, as opposed to creating a complex system that "does it all". With this in mind, we continued development of the Robotics Simulation and Control Lab (RSCL) framework, originally designed by the team in 2005. This platform had been abandoned in favor of the CARMEN toolkit from Carnegie Mellon in 2008, but we decided to revert to RSCL primarily because it was written in Java instead of C++, which significantly lowered the entry barrier for new students. This year, RSCL was almost completely rewritten to support remote sensors and effectors, as well as implement greater parallelization and more robust mapping and localization algorithms. Code was designed from the bottom-up, with focus on testing each input or output component for performance and reliability before integrating into the overall framework.

6.1 Structure

The software architecture was designed to be modular and adaptable, allowing interchanging of components and porting to new platforms with minimal effort. On the robot, the software is split into two layers - a set of platform-specific hardware drivers running on the embedded system, and the high-level, platform-independent processing and decision-making software (RSCL). Communication between the hardware controllers and RSCL is done over TCP/IP in a client-server model. This gives the ability to running the two halves on completely separate systems if the need arises.

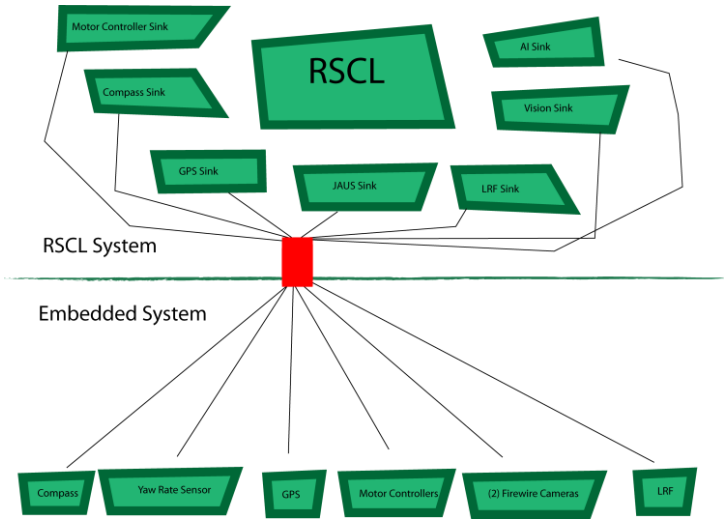

Figure 8: RSCL Diagram

Components within RSCL are organized into Sources (data providers such as sensors) and Sinks (data consumers such as motor controllers), so the software can be run with any combination of sensors and/or effectors. Each component runs independently in a separate thread to take advantage of parallel processing. Since some components are more important to the vehicle than others are (motor control takes precedence over vision

processing, for example), each thread is assigned a priority level which controls frequency of execution. This architecture greatly reduces inter-dependencies and allowed us to work on various components in parallel. Our decision-making algorithms also plug into the framework in the same way and act as the "glue" between the Sources and Sinks.

6.2 Graphical User Interface

Paradroid's graphical user interface provides an intuitive method of controlling the robot and testing vital systems remotely. It is designed to be run as a stand-alone package on any computer connected to Paradroid's on-board wireless network. Any number of users can monitor its systems simultaneously. The GUI displays the status of vital systems such as the GPS, compass, cameras, LRF, and motor controller. Each major software component can be started alone or in concert with other systems for increased flexibility during testing. In addition, the GUI supports a manual control mode in which the speed and direction of the robot can be set by a remote user.

6.3 Obstacle & Lane Detection

Line and obstacle detection is done based on data provided by the laser range finder and the camera. This data is layer processed onto a global map, which is then used for navigation. Data from each sensor "Source" is run through a processor before being integrated by the mapping component. During this processing, a confidence rating is assigned to the output that is then used to resolve conflicting information from two or more sensors.

6.3.1 Laser Range Finder

The simplest and most reliable sensor, the LRF simply provides an array of distances to detected objects in a 180-degree arc. For our purposes, the range is set to ten meters, although the device is capable of measurements 50 meters away. The data provided is used to identify obstacles by overlaying it with the map generated by the vision system.

6.3.2 Vision

The team originally started development with an FPGA-based Stereo-On-a-Chip (STOC) camera from Videre Design, but damage during testing required falling back on dual FireWire cameras from our previous robot, ReWIRED. We continued using the SVS stereo software bundled with the STOC camera to replicate the depth map using our standard cameras. While doing the processing on a general-purpose computer CPU is slower, the processor in Paradroid proved sufficient for the task, and more importantly, the depth map produced was identical to that produced by the STOC.

Our vision processing relies primarily upon color differences between various objects as well as the ground. During testing, we found that this approach is sensitive to ambient light and worked poorly in rapidly changing lighting conditions. To account for this an automatic recalibration algorithm was implemented to compute the new thresholds used for processing. To determine whether the ground color range needed to be calibrated, sample pixels are averaged from two locations that were highly unlikely to contain obstacles. If the averages shift significantly from one frame to the next, recalibration is requested on the next frame, which reduces the risk of calibrating based on a bad frame. The recalibration process consists of creating a histogram of every pixel in the frame and taking sample averages from the same two locations. The algorithm then scans outward along the histogram values until it finds a local minimum to each side, and sets those values as the minimum and maximum values for ground. When this process is completed for an image, it is handed to other components that run concurrently to handle line and obstacle detection.

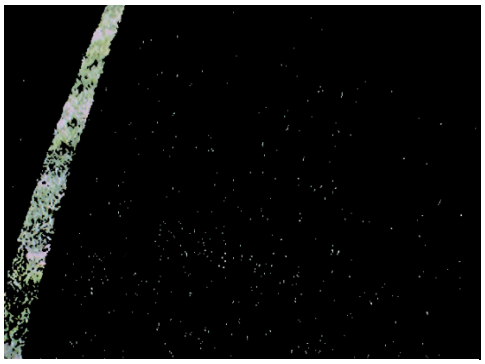

Figure 10: Depth Map Line Detection

Figure 9: Coordinates Given by Algorithm

Figure 12: Obstacle Detection Sample

Figure 11: Depth Map for Obstacle

Line detection is performed by taking the converted HSL image and filtering out the ground color range as well as any physical obstacles such as barrels. Obstacle filtering is based on data from the LRF. The remaining pixels represent lines or other discolorations on the ground.

Obstacle detection is handled similarly. The code uses an incomplete depth map calculated from images taken from each camera and filters out any data from the image that is mapped to the expected ground values. To fill in depth data from areas where there is not enough texture to compute a reliable value, the algorithm looks at the HSL image for areas different from the expected ground color range. It also removes pixels from the image where it determines there is an obstacle, to avoid processing the same object twice.

6.4 Navigation

6.4.1 Autonomous Movement

The mapping component in RSCL creates a map of obstacles, which is used by the path planner to determine the best course for the robot. All obstacles are stored with global coordinates, which are determined based on the robot's position at the time of obstacle detection. We have developed two user-selectable storage formats for the obstacle map. The first is an occupancy grid, and the second is a priority queue of obstacles, sorted by distance away from the robot. The first method is simpler and offers better performance and accuracy, but does not scale well in sparse environments. The second method uses storage space much more efficiently, but accesses are slower. Between the two, RSCL is able to operate in a wider range of conditions than if it were tied to a single representation.

Navigation is performed using a series of goals - a fixed point ahead of the robot for the Autonomous Challenge and waypoints in the Navigation Challenge. Since all navigation and obstacle avoidance by the robot is done via global coordinates, it is critical that the exact position of the robot be known at all times, and that the values do not drift over time. Unfortunately a high-precision differential correction subscription for the GPS is beyond our budget, so we implemented localization using wheel encoder and inertial measurement unit (IMU) data to improve precision. The GPS, in turn, corrects the drift and jitter present in the other sensors.

6.4.2 Lane Following

Lane following is simply accomplished by marking the lane boundary lines as another type of obstacle, and using the obstacle-avoidance algorithm to remain within the lane. In the case of dashed lines, the vision processor interpolates existing lines to fill in the gaps. As a secondary measure, in the Autonomous Challenge, the LRF is used to detect possible gaps in the lines by flagging any area free of obstacles for 10m as potentially unsafe.

7 JAUS Integration

Given that RSCL is a much simpler architecture than specified by JAUS, we decided to create an abstraction layer between RSCL and JAUS. All communication within RSCL still uses our custom protocols, as these are much more efficient given the limited code space and execution speed on some of our embedded system components. Instead, we represent most of RSCL as components and group them into the appropriate nodes. The appropriate IDs were also assigned so that commands targeted at specific JAUS nodes or components are passed on to the correct RSCL system. Since RSCL did not possess an existing remote command and control protocol, we adopted JAUS for all information transfer between the robot and a remote interface.

We based our JAUS implementation off the OpenJAUS project. Development of their Java implementation has unfortunately been discontinued and the latest version of JAUS supported was 3.2. We worked around this by extending the code to support version 3.3 as well as future implementations.

8 Cost Summary

Ideally, the team would design and manufacture all components on the robot for the experience it would provide. However, several components are too expensive to make in small quantities, require access to specialized

equipment, or are simply beyond the level of undergraduate work. These components, such as motherboards, motors, the GPS, and others, were purchased, saving us both time and money. The vast majority of the components on Paradroid were designed and manufactured by the team, including the suspension, frame, power supplies, and battery monitor. The vast majority of the software is written entirely by team members. In many cases, code originates from various open source projects and is updated or improved upon. After the competition, these improvements will be returned to their respective projects so that others may benefit from our improvements as well.

Table 3: Estimated part costs for the project, including those purchased prior to this academic year.

System	Item	Qty	Cost	Our Cost
Mechanical	1008 Low Carbon Steel / 6061 Aluminum	1	$550	$300
	Misc Hardware	1	$200	$200
	UHMW Polyethylene (6 ft)	-	$98	$98
	Pneumatic Tires / Bearings	4	$105	$105
	Wheelchair Motors	2	$210	$210
	#35 Roller Chain / Sprockets	4	$92	$92
Computer	Main Board - Foxconn G33M-S MicroATX	1	$95	$95
	Processor - Intel Q9300 Quad Core	1	$280	$280
	Memory - 4GB DDR2 800	1	$61	$61
Vehicle Control	System Controller - Atmel NGW100	1	$89	$89
	Interface Board – PCB & Parts	1	$50	$50
	Motor Controller - Roboteq AX3500	1	$395	$395
	Wireless Router - Linksys WRT54G	1	$50	$0
	Wire and Interface Hardware	-	$100	$45
Sensors	Fire-i Digital Board Cameras	2	$222	$0
	Quadrature Shaft Encoders - HEDS-5600	2	$120	$0
	GPS - Garmin 17HVS	1	$112	$0
	Yaw Rate Sensor - MLX90609	1	$60	$0
Power	Embedded Power Supply – PCB & Parts	1	$30	$20
	ATX Power Supply - M4-ATX 250W	1	$100	$100
	Batteries – 75Ah 12V Deep Cycle Lead-Acid	2	$120	$120
	Battery Monitor/Status Panel	1	$80	$80
Total			**$3,341**	**$2,340**

9 Performance and Conclusion

Table 4: Summary of predicted and observed performance.

Performance Parameter	Prediction	Result
Top Speed	5.0 mph	5.7 mph
Ramp Climbing	15°	21.4°
Curb Climbing	3"	6"
Reaction Time	0.1 sec	0.1 sec
Battery Life	2 hours	3 hours
Detection Distance	15 feet	17 feet
Waypoint Accuracy	0.5m	0.5m

The relatively high top speed and improved independent suspension system allow Paradroid to move fluidly over a variety of terrain at any speed up to 5mph. The combination of high-power motors and 4-wheel drive allows Paradroid to climb slopes and curbs with ease, while the omni-directional wheels make it easy and efficient to maneuver. Paradroid quickly reacts to obstacles due to the powerful on-board computer and optimized localization and path planning algorithms. The large lead-acid batteries provide ample life for testing and can last all day under intermittent use.

Paradroid was originally designed with both military and commercial applications in mind, and with the hope of advancing the field of unmanned ground vehicles. In that same spirit, Paradroid was redesigned and improved this year to be a strong competitor in the 2009 Intelligent Ground Vehicle Competition. Paradroid's modularity, versatility, and efficiency have proven it an ideal platform for autonomous vehicle research.