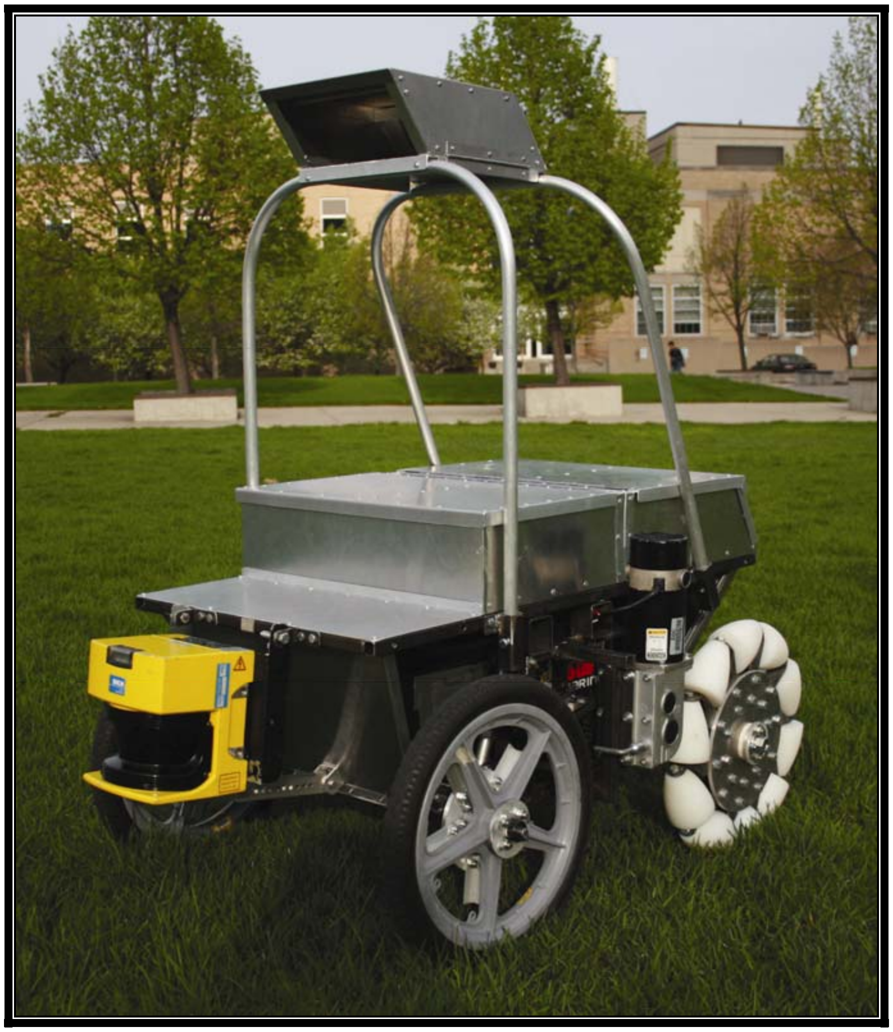
PARADROID



Presented by the
IEEE Robotics Team
University of Wisconsin-Madison

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

THE UNIVERSITY of WISCONSIN MADISON

Professor Nicola Ferrier
Department of Mechanical Engineering

1. Introduction

The University of Wisconsin-Madison is pleased to introduce Paradroid, an entirely new robotic vehicle which represents an improvement to the modularity and extendibility of our previous designs. The goal of this project was to go beyond the challenge of the competition and design a versatile, multi-purpose platform.

2. Innovations

Paradroid attains an unparalleled level of modularity and extendibility compared to competing vehicle platforms. The mechanical design of Paradroid is based on a sturdy four-wheeled platform that can accommodate a large variety of modules. This four wheel drive platform features two custom built omni-directional wheels to maximize turning efficiency and minimize rolling resistance and vibrations. Sensors and processing power are connected to the drive platform via detachable interfaces, allowing the robot to be easily configured for specialized tasks. This approach maximizes the expandability of the platform. Each compartment is accessible simultaneously, facilitating testing and debugging of various systems. Paradroid also includes independent shock absorbers on each of its four wheels to allow for smoother operation and maximum traction on rough terrain. Powerful wheelchair motors make Paradroid capable of maintaining 5 mph speed even on moderate inclines.

Paradroid also incorporates a number of power and embedded control innovations. Using an advanced motor controller, the Roboteq AX3500, Paradroid can achieve extremely tight speed control and utilizes regenerative braking to increase overall vehicle efficiency. The robust power system was custom made to efficiently power (85% nominal) the various embedded systems in a small package. This makes Paradroid's power system adaptable to platforms with different operational voltages from 8 to 30V. A power monitoring system dynamically estimates current power capacity and remaining runtime during vehicle operation. While charging, the E-stop is automatically engaged to allow safe testing of other components without danger of the vehicle moving. A commercial off-the-shelf Atmel development board forms the core of the embedded system, allowing for a substantial amount of processing to be done at the embedded level. This relieves the main computer of low level tasks like dead reckoning and sensor interfacing. The embedded system also calculates the vehicle's global position allowing the robot to be driven manually without any additional computing power on-board. An adaptive odometry algorithm was implemented utilizing yaw rate and GPS sensors to give better global position accuracy without the cost of a much more expensive differential GPS system.

Paradroid's sensors were chosen for versatility and cost effectiveness. A stereovision system provides a complete solution for both depth perception and line detection. This option is more cost effective than a laser range finder and camera combination. The stereo-vision camera retails at $1800, while a laser range finder and camera combination would have cost approximately $5200. Also, the stereovision camera system makes Paradroid better suited for stealth-based applications than a laser range finder because it is a passive sensor.

Paradroid is equipped with an entirely new software package based on the open source CARMEN framework. The modularity of CARMEN allows the robot to be easily configured to incorporate data from a variety of sensors. Furthermore, the CARMEN package comes equipped with many common drivers, which allowed the software team to focus on high level concepts rather than low level integration problems. Paradroid uses Simultaneous Localization And Mapping (SLAM) algorithms to plan its course, a feat that the team has never attempted before. It adds sensory data to an environmental map and then uses this information to find the optimal path to the objective. Probability based localization algorithms correct for accumulated error in odometry data to prevent this error from invalidating the map.

The new software package also features a graphical user interface designed to be viewed over WiFi from any secure computer. This allows for remote testing and calibration as well as multiple connected operators. The GUI displays sensory data, system status, video streams and mapping information in real-time, allowing for truly remote operation. Lastly, the robot is JAUS Level 2 compatible, meaning that communications are performed over a standard military protocol. This adds to the modularity and reusability of the design and provides a means for communication with other JAUS compatible devices such as remote operator controller units. Our software goes beyond the requirements of IGVC's Level 3 JAUS Challenge by using tele-operation commands and implementing controller authorization policies to prevent multiple nodes from sending conflicting commands.

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 past IGVC entries. It was decided that in order to address some of these shortcomings, especially the mechanical ones, an entirely new robot should be built. The team spent an estimated 2000 hours over a period of nine months designing and producing Paradroid.

3.1. Team Structure

The team is a UW-Madison 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 functional sub-groups which hold multiple weekly work sessions and an all team meeting to discuss inter-disciplinary issues. Leaders for each team were chosen by past involvement, experience and seniority. Most major design decisions were 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.

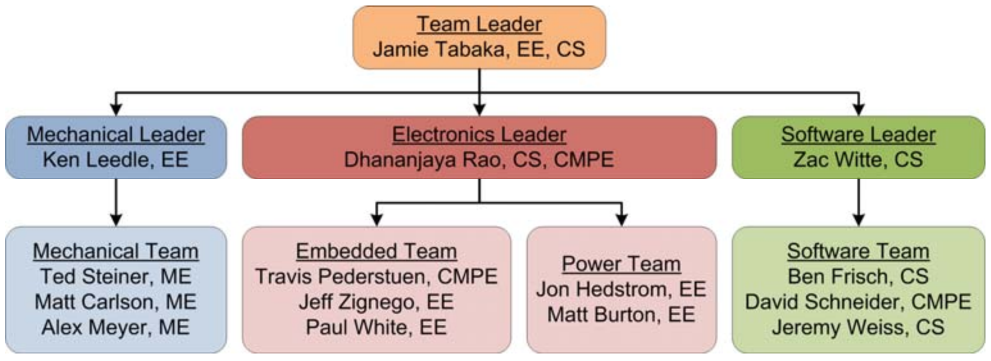


Figure 3.1: Team Organization

3.2. Project Planning

The team used the agile development methodologies for the entire project planning lifecycle. The agile methodology focuses on face-to-face interaction as the main form of communication, rather than rigid processes and tools, and encourages design flexibility as the project evolves into its final form. One of the agile core principles is creating self-documenting software over formal documentation, thereby keeping the documentation up to date at all times and not relying on quickly outdated and time consuming static documentation. The agile methodology was chosen because it is ideal for rapid development and strips out heavyweight project management practices that have failed the team in the past. Agile methodologies are also better suited to the rapidly changing circumstances a volunteer student organization faces with continuously varying monetary and human resources.

The planning process began with a brainstorming session where ideas for desired features of the robot were generated. Each feature was written on a separate note card and ranked according to priority. Each feature was then broken down into smaller tasks that could be easily accomplished within a one-to-two week period. After each period, the task's progress was evaluated and the next task was started. This process breaks seemingly daunting projects down into manageable sections which can be accomplished by even less experienced members. The advantage of the note card task tracking system was that it allowed the interdependence of different features and tasks to be understood easily. As the year progressed, the 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.

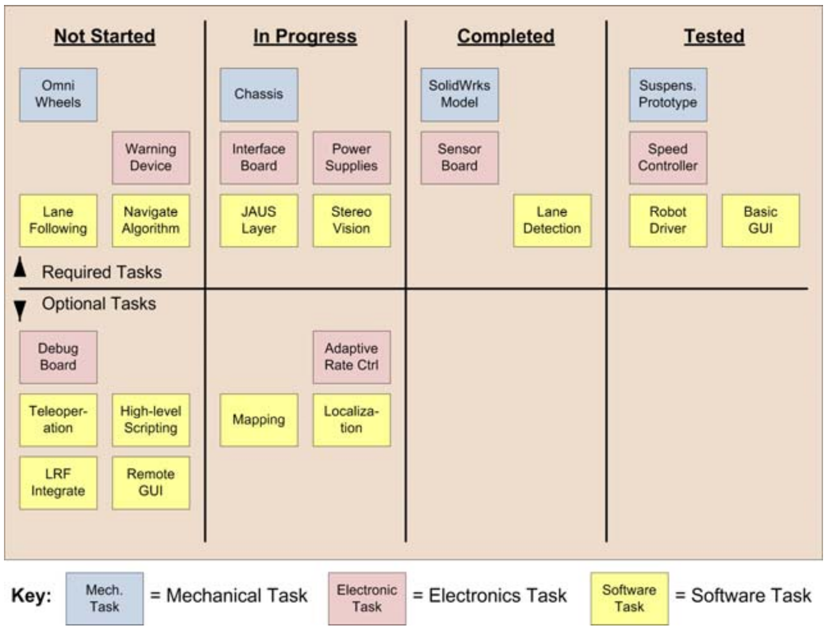


Figure 3.2: Example Project Planning Board

3.3. 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 changing scope of their project. 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 3D modeling and design software package, was used to model each component in the vehicle. By using SolidWorks, many ideas were able to be tested very quickly at a relatively low cost, and components could be tested for interferences and proper interaction before they were built. SolidWorks also helped reduce waste by giving very accurate estimates of necessary materials that would be needed. From time to time, experts 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 usually built for proof-of-concept testing. When the prototypes were shown to work, 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 power boards, which could not be prototyped as easily, were designed and manufactured with ample time for redesign if revisions were needed.

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, which reduced the need for documentation. 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 non-functioning 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, safe, accessible and efficient (in terms of both power and space) vehicle. The main goal of the mechanical design is to provide a versatile platform that software and embedded systems can be easily tested upon and interfaced with. It embraces modular design for reusability and upgradeability. The design structure consists of a drive base and separate modules for the embedded system, custom built computer, laser range finder and main sensor module. These compartments interface with each other but carry out their separate functions independently.

The drive base features independent shock absorption for each wheel to handle rough terrain. The entire robot is weather resistant from the outside but the compartments are open inside for ventilation and air circulation. Each compartment, including the batteries, can be accessed simultaneously for maximum accessibility. The vehicle was designed and manufactured completely from scratch by team members.

3.4. Drivetrain and Suspension

Paradroid is a differentially steered, four wheel drive vehicle with fully independent suspension. Both the left and right-side wheels are powered by 24V DC wheelchair motors rated at 0.9HP continuous duty. These motors provide ample power for turning and climbing ramps up to 20°, as well as curbs and small stairs. The vehicle's top speed is 5.7mph, which is limited to 5mph by the control software. This allows Paradroid to maintain 5mph speed when climbing ramps and traversing rough terrain. Differential steering was chosen due to its simplicity, as it requires fewer moving parts and the zero turning radius allows for greater control.

The front wheels are 16 inch, five-spoked pneumatic tires and the rear wheels are 16" custom-built omni-directional wheels (Figure 4.1). The combination of pneumatic tires and omni-directional 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 these wheels to transmit forward or reverse torque, as well as roll side to side with very little resistance. These wheels greatly reduce power consumption by reducing turning friction, the main drawback of four wheel drive differential steering. The large 1.4" to 3" diameter of these rollers allows them to roll over most small obstacles without difficulty or a significant reduction in ride quality.

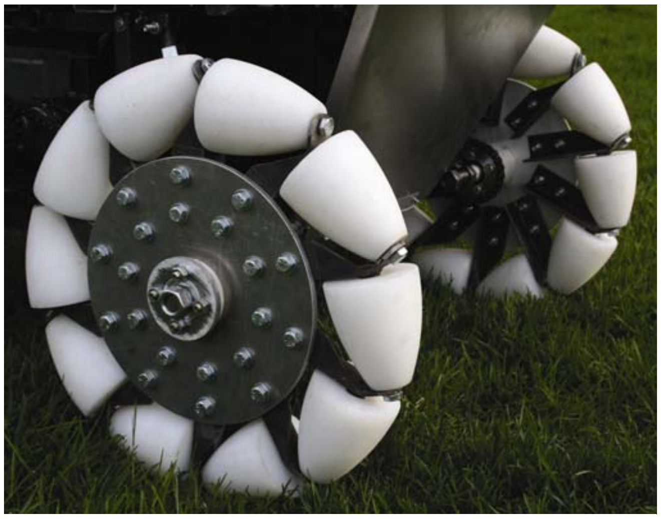


Figure 4.1: Omni-wheels

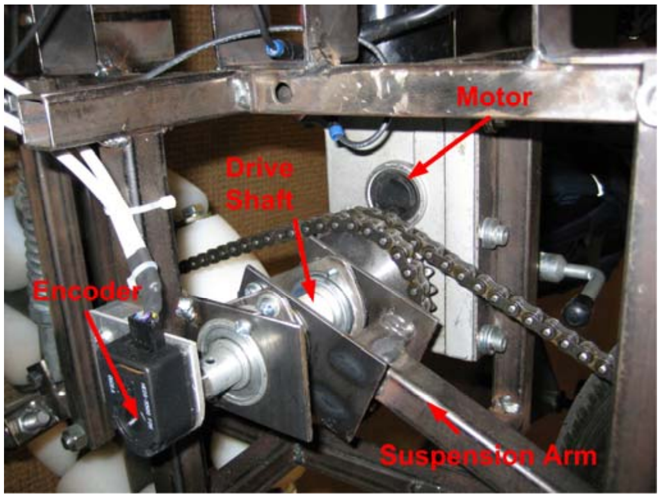


Figure 4.2: Drivetrain Components

The suspension system is designed to allow Paradroid to traverse a wide variety of terrain, from smooth to rough. The suspension is optimized for a 225 lb vehicle, which is Paradroid's operating weight with a 20lb payload. Used in conjunction with the pneumatic tires, the suspension cushions the ride for the electronic and mechanical components, reducing vibrations and fatigue. The unique coaxial drive sprocket and suspension pivot allows the drive modules to maintain chain tension throughout the entire suspension travel (Figure 4.2). The suspension arms utilize a single pivot design for simplicity and reduced cost and are long enough to have an effectively vertical path of travel.

Each side of the drivetrain is a self-contained module, allowing them to be individually tuned and tweaked, as well as mounted on virtually any platform with 5 mounting holes. The drivetrain also incorporates parking brakes which are activated any time the robot is not purposely moving. In the unlikely event of an electrical failure, the parking brakes will automatically engage, bringing Paradroid to a halt from full speed in less than 12" of travel. Under normal operation regenerative braking is used to bring the vehicle to a complete stop, for example, in the event of emergency stop activation. This method allows the kinetic energy of the vehicle to be stored back in the batteries while bringing the vehicle to a stop in less than 18" from full speed.

3.5. Chassis

Paradroid's chassis consists of a minimalist steel skeleton to support all modules. The core of the frame is a simple rectangle on the top and bottom; everything else on the robot is modular. This design allows any individual component to be augmented independently any other component, while securely holding all components of the robot to the frame. The steel frame provides superior strength and durability, while also being resilient to impacts and reducing vibrations. It also reduces costs by having fewer load bearing members, and low cost sheet metal covers the outside and protects against the environment and minor impacts.

The chassis has a minimum of 7 inches of ground clearance, but under normal operation 8 inches of ground clearance is maintained. The battery compartment is located in the center bottom of the vehicle to provide a low center of gravity and improve wheel traction. Paradroid uses two 12V 75Ah deep cycle marine batteries for over 3 hours of continuous run time under normal use. Each battery can be exchanged with a fully charged battery in less than 30 seconds to maximize run time. Also, a charging port integrated into the side of the robot allows the batteries to be charged inside the vehicle for convenience.

3.6. Embedded Compartment

The embedded compartment is located in the middle of the "deck" on top of the chassis. This compartment houses the motor controller, power supplies and boards for interfacing with all of the robot's sensors. The open bottom allows air to circulate between this compartment and the battery compartment and makes routing wires and cables much simpler. All of the components are located on an organized subpanel, which can be easily removed to be worked on. The entire compartment can also be easily removed via four screws.

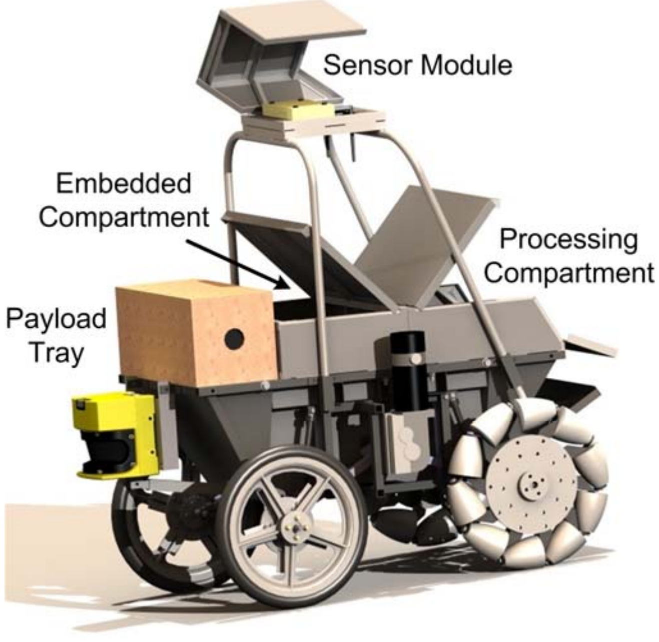


Figure 4.3: Mechanical Compartments

3.7. Processing Compartment

The processing compartment is situated on the rear of the deck for easy access. It is designed to house either a laptop or a custom built computer. It is designed similarly to the embedded compartment, giving it protection from impacts and the environment. This compartment has two doors that, in combination with the open bottom, allow unobstructed access to the computer. The rear door can double as a keyboard tray when using the custom built computer. In addition, the flat lid provides an ideal location for a laptop to be placed to interface with the internal computer.

3.8. Sensor Module

The main sensor module is mounted directly above the embedded and processing compartments and houses the majority of the sensors on the robot. It is designed with the same frame and covering style as the embedded and processing compartments. The stereovision camera, GPS sensor, digital compass, yaw rate sensor, status panel, emergency stop and wireless router are housed in the sensor module. This compartment is directly above the pivoting axis of the robot in order to keep the camera and sensors in a fixed reference during turns.

The entire outer shell of the sensor module is attached with a hinge and can be easily lifted out of the way, allowing easy access to all compartments housed within. The status panel displays the status of every major component of the embedded system and is placed in a convenient location to be easy to view. The wireless router allows the robot to communicate with an operator control unit or another laptop, even as it is driving. The usual height of the module is 4 feet off the ground, which keeps Paradroid's profile low and inconspicuous. In addition, Paradroid has the option of using a laser range finder which can be attached to a mount in front with 3 pins.

3.9. Payload Tray

The front of the deck is used as the payload tray. This versatile tray can accommodate payloads of various sizes and shapes weighing up to 50lbs. The location of this bay distributes more weight on the pneumatic tires to reduce vibration and increase traction. This is an ideal location for a sensory payload, which in a military or commercial application might be the primary purpose of the robot.

5. Power System Design

The power system for Paradroid was designed with the intention of making it both efficient and robust. Two deep cycle marine batteries supply 24 volts, which is distributed using a central distribution board that monitors the flow of current to each subsystem. The distribution board implements current integration and error reducing algorithms to estimate power consumption and extrapolate remaining battery life. Versatile and efficient flyback switching regulators supply 24, 12, or 5 volts depending on need, and an on-board custom charger is used to recharge the batteries. The power supplies are extremely rugged and able to handle conditions such as under-volt dropout with hysteresis, over-voltage, over-current, reverse polarity, thermal overload and electrostatic discharge. They have extremely wide input ranges from 8 to 30 volts, allowing the embedded and computing system to be extremely modular between other systems as well as be supplied from the primary batteries without causing problems during motor current spikes. Depending on load and temperature, the relative efficiencies of the converters are 84% to 95%.

The emergency stop cuts the power to the motors with a normally-open relay. It can be triggered by an exterior switch, a wireless signal, or by a signal from the interior charger which prevents the robot from accidental movement during debugging.

As an optional feature on the robot, a laser range finder (LRF) can be mounted to provide additional spacial information. The LRF now incorporates its own power supply with an input range from 8 to 30 volts. This allows the LRF to be used with a variety of power sources and easily attached to different systems.

Top-Level System (24 volts)	**Sub-System**	**Component**	**Max Power (mW)**	**Typical Power (mW)**
Embedded Power Supply	12 volt output	Efficiency: 84% @1.5A	36,000	
		Atmel NGW100	300	100
		Soekris Wireless Router	12,000	3,360
		Garmin HVS17 GPS	100	
		Videre Stereo Camera	2,000	
	5 volt output	Efficiency : 86% @1A	7,500	
		Interface Board	50	
		Status Panel	400	
		Warning Light	500	
LRF (optional)	24 volt supply	Efficiency: 89% @.8A	30,000	
		Sick PLS101 LRF	30,000	20,000
DC-DC ATX Supply		Efficiency : 95% @110W	5,800	
		MicroATX Mainboard	15,000	
		Core2 Processor	95,000	
Roboteq AX3500 Motor Controller		Efficiency: 90% @2.9KW	322,000	
	H-Bridge Output	Motors	2,880,000	400,000

Table 5.1: Power Requirements

6. Electronics Design

An embedded system was designed to interface the sensors to the processing unit and provide low-level processing of sensory data. One of the primary goals of the system was user-friendliness, both in terms of using the embedded system and interfacing with a processing unit. To this end, the system also had to be flexible in accepting a variety of connected devices using multiple communication protocols. Components were selected to minimize power requirements and cost. All components of the embedded system have been tested rigorously to ensure reliability, and the design allows for multiple connection options to individual devices.

The daughter boards include a status console and the motor control board. The status console displays vital information about each system on the robot and can be configured through the embedded system to display several variables. The information displayed includes battery levels, power indicators for various systems and an LCD diagnostic screen.

Also included on-board is a wireless router to enable remote access to the vehicle's internal systems. This allows multiple people simultaneous access to different parts of the software and embedded systems, via remote debugging software like the remote GUI. The router is also meant to be the primary connection point for JAUS communications.

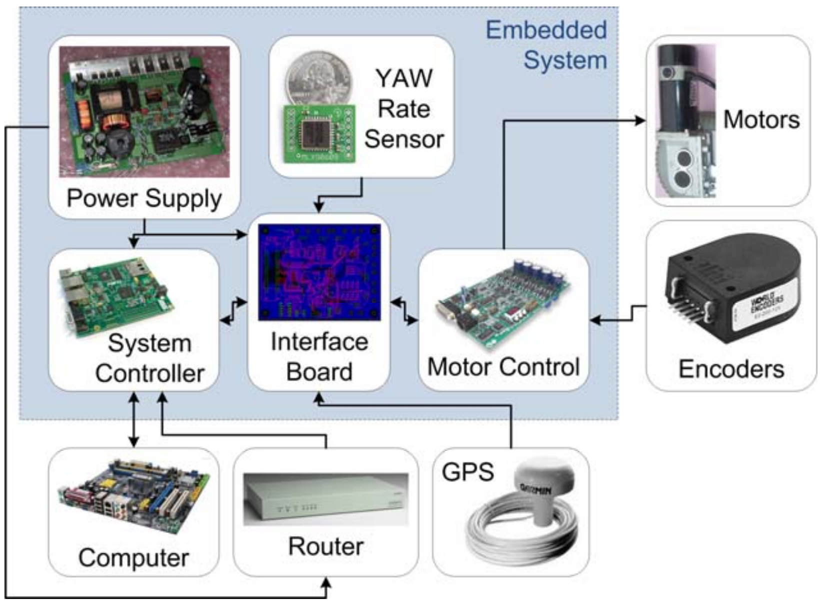


Figure 6.1: Embedded Systems Diagram

3.10. Vehicle Control

3.10.1. Embedded System Controller

The embedded system is based off an Atmel NGW100 development board which was chosen for its cost effectiveness, ease of use and commercial off-the-shelf (COTS) availability. The Atmel line distributes free C compilers, allowing for the lowest cost development possible. The development board has an Ethernet interface that is used for communications with the processing unit while RS232, I2C and SPI are used for communications with additional daughter boards. The board features a low-power AVR32 processor running at 130MHz with on-board flash memory and a SD slot for memory expansion. This allows the board to run Linux, which greatly simplifies development and usage of the system.

3.10.2. Motor Speed Controller

A Roboteq AX3500 motor controller interfaces between the motors and the embedded system. This offers the advantage of a quick and reliable setup. The AX3500 uses a simple ASCII-based command and control protocol over RS-232, eliminating the need for highly-customized communication protocols.

3.10.3. Adaptive Odometry System

Rather than employing an expensive high-precision GPS navigation system, we developed a hybrid navigation system that uses data from a low-cost GPS unit, wheel encoders and a yaw rate sensor. We initially considered using a digital compass as well, but decided against it due to the susceptibility of the compass to external influences such as large metallic structures or even magnetic fields generated by the vehicle itself. We expected the greatest error in odometry to occur when the vehicle was turning, as wheel slippage would lead to incorrect readings from the encoders. To mitigate this error, we installed a yaw rate sensor to obtain precise information about the vehicle's lateral movements. This "dead-reckoning" approach to navigation is not be sufficient by itself, however, since errors accumulate over time. Hence, GPS data is used to correct the odometry data. The GPS unit does not suffer from cumulative errors, but isn't as accurate or precise. The combination of the two systems results in an ideal low cost and precise navigation solution.

3.11. User Interface

3.11.1. Status Panel

The status panel integrates information from various components of the vehicle into a single, easily accessible display. The panel provides a central location from which to manage various parts of the vehicle, as well as enabling quick detection of problems when they arise. The panel features power control switches for individual subsystems, a Go/No-Go LED array that indicates operational status and an LCD for accessing advanced functions.

3.11.2. Warning Device

The warning device was designed to be highly visible at all times, using an array of Super Bright LEDs arranged in a circle for 360-degree visibility. The warning light allows for indication of vehicle operation mode (autonomous vs. manual) as well as indication of major faults if they occur. The system was designed in a modular fashion, so it can be scaled to different sizes relatively easily. Power to the LEDs can be controlled via software, enabling significant power savings when the vehicle is operating in optimum lighting conditions.

3.12. Main Processing

All other processing is performed by an on-board mini-ATX computer. A mini-ATX computer was chosen over a laptop because it is able to provide more processing power. This allows for more complex processing of sensory data as well as faster reaction times. Furthermore, because the graphical interface for the software can be run from any computer wirelessly, an on-board monitor was not needed. The custom mini-ATX computer is outfitted with a 2.5GHz Quad Core processor, 4GB DDR2 RAM and 8GB of solid state memory. Solid state memory allows the computer to handle whatever small vibrations make it through the suspension. A DC-DC power supply allows it to run directly off the primary 24V battery supply.

7. Software Design

Paradroid runs an entirely new software package which was designed around the processing requirements of intelligent vehicles. The software rests on the framework of the open source Robot Navigation Toolkit, CARMEN, which encourages modularity and interoperability between logical components. CARMEN includes drivers for the SICK laser range finder, the GPS and a generic robot driver. It also includes common high level modules such as a simulator, a path planner and GUI controls. These served as stepping stones from which we built custom modules to serve our purposes. The ability to swap out different versions of these modules helped the development and testing process a great deal.

Carmen organizes logical functions into separate processes that communicate using an inter-process communication server (IPC) called "central". It passes messages over TCP sockets, hence each process can be run either locally or remotely. Persistent parameters are stored in a configuration file and passed up to any module subscribing to a parameter. These parameters can be changed in real-time and the new value is pushed to any process using that data. The Paradroid base module interfaces with Paradroid's embedded systems via JAUS messages to drive the robot and collect sensory data. High level modules including navigation, vision processing and mapping run as separate processes, taking advantage of the computer's four cores.

Paradroid can run exclusively on the stereovision camera for obstacle detection, line detection and mapping. It can also use laser range finder data in place of the stereovision depth map data if greater accuracy is required. All sensorary data is aggregated on an environmental grid map after passing through a probability-based localization process to correct for incremental odometry error. All sensor and map information is displayed in real-time on a remote graphical user interface application written in the Qt windowing library for cross-platform compatibility.

3.13. Vision Processing

3.13.1. Stereo Vision and Obstacle Detection

The Videre Stereo-On-Chip (STOC) system is optimized to perform real-time pixel correspondence and disparity selection on-board, freeing the main processing computer from these intensive computations. The camera provides a depth map of the areas in front of it in the form of a video stream. A 3D reconstruction is generated from the depth map by projecting each pixel point back into 3D space with an exponentially increasing size of granularity to ensure that areas closer to the robot are represented in more detail. From the 3D representation a ground plane is estimated and subtracted from the data so that only obstacles above ground remain. The remaining obstructions are rotated and flattened along the z-axis into 2D space, then scaled to fit the resolution of the world map. The result is a 2D occupancy grid, which is passed to the mapping module and added to the world view.

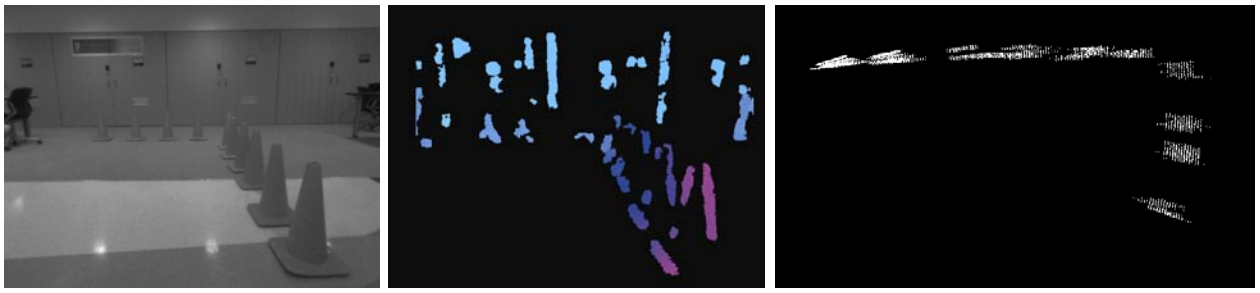


Figure 7.1: (a) Monochrome Camera Output (b) Stereo Camera Depth Map Output (c) Resulting World Map

3.13.2. Line Detection

Line detection employs a color filtering pre-processes to isolate lines based on hue and saturation values from the stereo camera color image stream. The Otsu algorithm is used to perform automatic thresholding before binarizing the data. The Hough transform is used to pick out lane boundaries from obstacles and noise. Next, Monte-carlo methods are used to match cubic polynomials to the binary image to interpolate dotted lines or noisy environments and provide line detection. Lines are stored in vector format and projected on to the grid map as constraints for path planning. Throughout the entire process, any data which is deemed too different from the previous line estimate is discarded to prevent single errors in processing from being added to the line vector.

3.14. World Mapping and Navigation

Paradroid maps the obstacles and lines it detects onto a persistent two-dimensional probability grid map anchored and scaled to true GPS coordinates. The mapping module can optionally run in corrective mode, which decrements the probability of occupancy on grid cells that have no obstructions, repairing erroneous data previously mapped. The map is layered so that stereovision, LRF and line detection data can be stored separately. This enables more intelligent reasoning on the accuracy of the data and allows for selective error correction. The mapping module is designed to work dynamically in any size environment and resizes as the robot approaches the boundary of the current map.

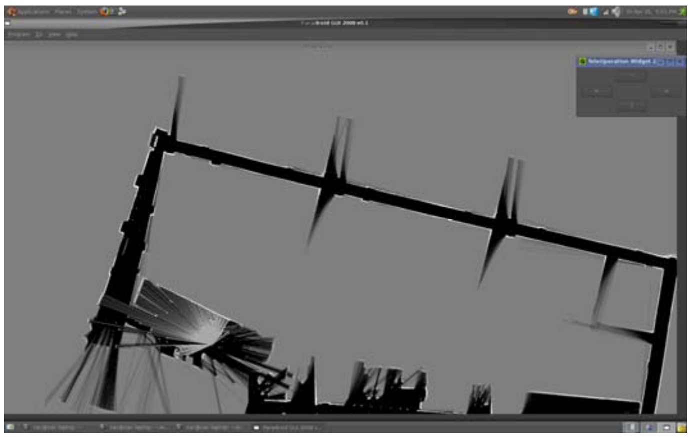


Figure 7.2: Probability Map of a Building Level

Because of the large amount of data the map stores, a 90x40 meter course would require about 8MB of memory in raw format. For better data passing efficiency, the map is compressed using zlib. An impressive 99.5% compression rate is achieved because of consistency in data, but the processing power used for compression can become an issue as the map grows in size. To address this problem, a fragmented updating policy is used to only send data that has changed since the last update.

Mapping alone is subject to the accumulated error in odometry data. To correct for this, the sensor data goes through a localization process on each update. Simultaneous Localization And Mapping (SLAM) has always been a significantly more difficult problem in mobile robots than simply localization or mapping alone. Typical SLAM algorithms are impractical for real-time applications with limited processing power like ours, or they require a feature graph with known correspondences. Our unique implementation was inspired in part by algorithms presented in Sebastian Thrun's book *Probabilistic Robotics*, which we fundamentally modified to work within our constraints. To negate accumulated error the most probable position is taken and mapped given the entire history of odometry and sensor data by integrating out past poses from the posterior data. To solve this using Bayes' Rule requires hypothetical sensor data to be generated for each probable pose so we limit the set of considered possibilities using a number of factors. These factors include measuring the difference between the reported pose, previous pose and the overall performance of the process.

3.15. Challenge Strategies

Paradroid uses a predictive algorithm which relies on the world map to navigate both challenges. The predictive algorithm determines the safest path to a given location based on cost calculations for possible paths. The cost is proportional to the probability of encountering an obstacle on the world map.

For the Navigation Challenge, the desired location is always a GPS waypoint. Which waypoint the software chooses is based on cost calculations done for all waypoints. Initially, when the majority of the world map is unknown, the robot chooses the closest waypoint. If a large enough obstacle is detected, such as a fence, the robot may choose a farther waypoint which is estimated to be closer to drive to.

For the Autonomous Challenge, the robot also drives towards a GPS location. This location is chosen according to the location of the robot and estimations of where the lane is on the world map. The location is continually updated as the robot moves farther down the lane, and hence it is essentially a "carrot on a stick" leading the robot forward. The previous locations of the robot are tracked to allow calculations of which direction is forward in the lane and to prevent the robot from backtracking.

3.16. Remote GUI

Paradroid's graphical user interface is a vital component giving meaningful feedback from processing components for testing and debugging. It is designed to be run as a stand-alone package on any computer connected to Paradroid's on-board wireless network. Hence, any number of users can monitor its systems simultaneously. JAUS serves as the primary communication medium. However, some data such as the grid map and parameter editor speak directly with CARMEN since JAUS has no defined message structure specific to that information. The configurable GUI provides a workspace for widget windows displaying information from one or more systems or offering command capability. Exclusive control over the robot is negotiated over JAUS and the GUI reflects the state visually.

8. JAUS Challenge

Paradroid implements Level 3 of the JAUS Challenge using the open source OpenJAUS libraries. The team's first attempts at the JAUS challenge began two years ago with a java framework and were implemented from scratch. We read the reference architecture documents and consulted with Brian Prodoehl, a former UW-Madison student and active developer in the JAUS community. We learned how difficult it is to correctly implement the broad and extensive standard to any practical state of usefulness so this year we investigated OpenJAUS. The stable, public release of OpenJAUS follows the RA3.2 specifications, which are now one year obsolete and their implementation was in an awkward state using c, c++ and java for different parts. We contacted the development team and expressed our desire to help test the new 3.3 branch which recently entered a closed beta phase. Since April of 2008 we have participated in weekly phone conferences and exchanged emails with the group of roughly 15 other beta testers on various bugs and insufficiencies. The motivation for our participation was to avoid investing time modifying OpenJAUS 3.2 and to achieve a much higher level of functionality than we would if we tried to implement each component ourselves to the minimum requirements outlined in the IGVC rules.

As OpenJAUS 3.3 is still in beta, we did have difficulties getting everything to work. The lack of documentation on how to interface with the node manager and message parsing functions slowed down development by a great deal. It forced us to explore and understand the library source code in order to use it.

Besides implementing the Level 3 Challenge, we also implemented JAUS Level 2 to communicate between the processing unit and the embedded system as well as the remote GUI. A Level 3 implementation was considered, but it was decided that it would not be efficient for our purposes to design individual electronic adapters to convert each purchased sensor into a JAUS Level 3 component as defined in the reference architecture. Instead, the embedded system acts as a data grabber for all of our sensors, so that the data can be packaged into JAUS messages and sent up to the processing unit. From the perspective of the computer, this strategy makes sensors appear to be Level 3 components. Although we are not to the point of replacing all embedded communication to external entities with JAUS messages, we have the capacity and intention of doing so. This opens the possibility of operating the vehicle wirelessly with a JAUS operator controller unit, without a computing payload. Although not useful for the IGVC challenges, this could be a cost-saving feature in other robotics applications.

9. Cost

Cost was always a consideration in the design of Paradroid. The convenience of using consumer off-the-shelf (COTS) components was balanced with the cost-effectiveness of designing systems from a low level by using a combination of the two approaches. Systems were designed in-house when it was decided that they were relatively easy to design and maintain and that the design could be put to use on other vehicles in the future, such as in the case of the power supplies. Systems were purchased when it was decided that the work of implementing them would be more effort and maintenance than the cost associated with purchasing the system, such as the stereovision camera system and the CARMEN software framework.

System	Item	Qty	Retail Cost	Our Cost
Mechanical	1008 Low Carbon Steel / 6061 Aluminum	-	$550	$300
	Misc Hardware	-	$200	$200
	Pneumatic Tires, Bearings & Omni-wheel Materials	-	$203	$203
	Wheelchair Motors	2	$210	$210
	#35 Roller Chain / Sprockets	4	$92	$92
Computer	MicroATX Motherboard, Processor & Memory	1	$436	$436
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 - Soekris net4826	1	$172	$0
	Wire and Interface Hardware	-	$100	$45
Sensors	Stereo Vision Camera – Videre STOC	1	$1800	$1200
	Quadrature Shaft Encoders - HEDS-5600	2	$120	$0
	GPS - Garmin 17HVS	1	$112	$0
	Yaw Rate Sensor - MLX90609	1	$60	$60

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
Total			**$4839**	**$3520**

Table 9.1: Bill of Materials

10. Predicted Performance

Paradroid was designed to have superior performance in virtually every category. The increased top speed and independent suspension system allow Paradroid to move fluidly over a variety of terrain at any speed up to 5mph. The combination of high power wheelchair motors and 4 wheel drive allows Paradroid to climb steep slopes and curbs with ease. The omni-directional wheels make it easy and efficient to maneuver. Paradroid reacts to obstacles nearly three times faster than its predecessors 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.

Performance Parameter	**Prediction**	**Result**
Top Speed	5.0 mph	5.7 mph
Ramp Climbing	15°	20°
Curb Climbing	3"	6"
Reaction Time	0.1 sec	1/30 sec
Battery Life	2 hours	3 hours
Detection Distance	15 feet	17 feet
Waypoint Accuracy	0.5m	0.5m

Table 10.1: Predicted and Observed Performance Parameters

11. Conclusion

Paradroid was designed to be a strong competitor in the 2008 IGVC competition. Paradroid's modularity, versatility and efficiency make it an ideal platform for autonomous vehicle research. It was designed with military and commercial applications in mind, with the hope of advancing the field of unmanned ground vehicles.