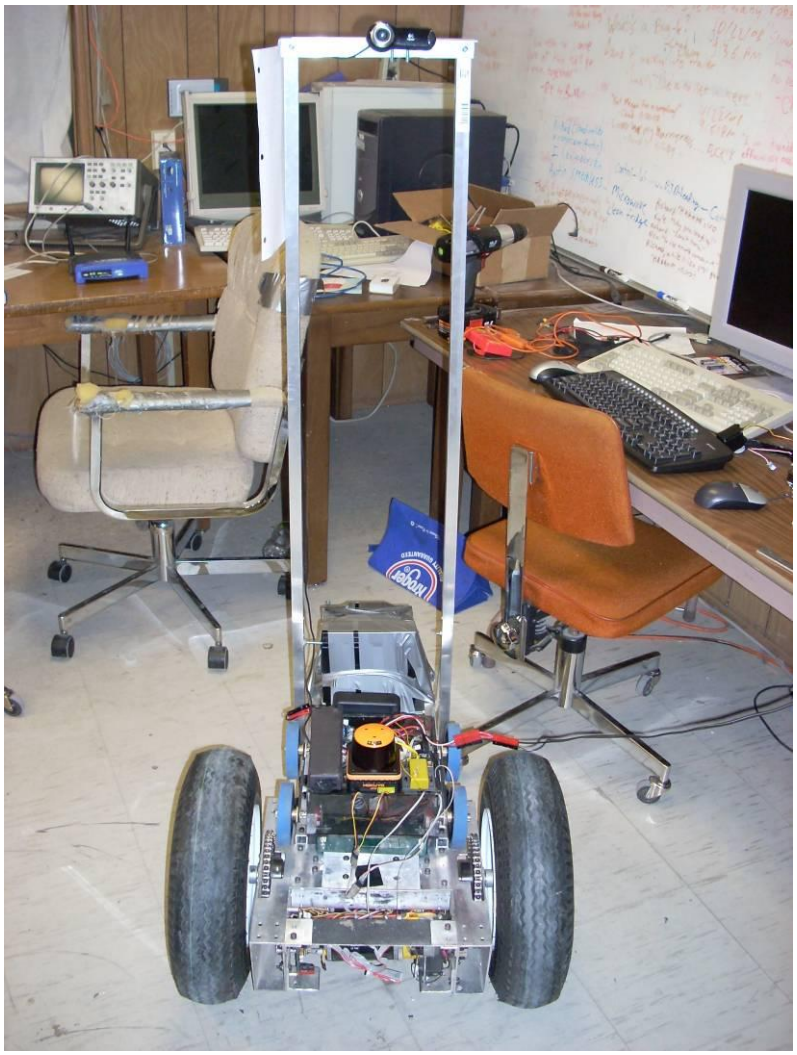
RASmanian Devil

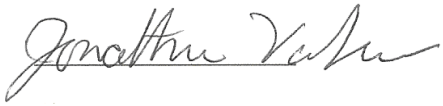


Designed by Richard McClellan, Nicu Stiurca, David Yanoshak, Alex Boehm, Steven Hall, and Juan Ortiz

Faculty Advisor Statement

I certify that the engineering design of the new vehicle, RASmanian Devil, has been significant and each team member has earned or could have earned at least two semester hour credits for their work on this project.

Signed,



1.0 INTRODUCTION

This paper describes the University of Texas at Austin's (UT-Austin) design of RASmanian Devil for the 17th annual Intelligent Ground Vehicle Competition (IGVC). This vehicle is a culmination of many hours of design and effort by voluntary student team members. RASmanian has been designed to participate in the three competitive events at IGVC. RASmanian was built as a robotic platform with portable software and integrated commercial off-the-shelf (COTS) hardware.

Student	Contribution	Major	Year	Hours
Richard McClellan	Team Lead/Integration	ME	4th	700+
Nicu Stiurca	Software/Integration	CS	2nd	500+
David Yanoshak	Electrical/Mechanical	EE	4th	150
Alex Boehm	Mechanical	ME	2nd	100
Stephen Hall	Electrical/Mechanical	EE	1st	50
Juan Ortiz	Mechanical	ASE	2nd	50
			Total (approx)	1450+

Table 1: Work Division Breakdown

2.0 DESIGN PLANNING PROCESS

The process of designing a robotics system involves a careful balance of trade-offs with complexity and hardware. Our team set many deadlines and deliverables to schedule time for testing and re-evaluation. We set a timeline to achieve hardware, portable software drivers, functional electronics and sensor integration. After multiple progress reviews of the design at set time periods, we made appropriate modifications based on available monetary resources and testing time.

2.1 Sub-Teams

The team divided into mechanical, electrical and software section sub-teams. The mechanical group focused on drive dynamics and manufacturing simplicity. The electrical group focused on power management and interconnection between all electrical systems. The software team integrated together sensor data for path planning and navigation.

2.2 Design Sequence

Our design sequence begins with high level block diagram of sensors, electronics, software and mechanical ideas. Many design decisions involved choosing commercially-off-the-shelf (COTS) hardware to achieve a layer of abstraction from low level devices. After establishing a sufficient roadmap of the

various sub-systems, the sequence begins an iterative loop of evaluating an integrated sub-section and redesigning when necessary.

Our approach differs from a more traditional planning which requires more simulation and analysis of each decision. Our organizational structure and time budget required us to accelerate the analysis phase. To demonstrate proof-of-concept, we create a series of rapid prototypes to eventually achieve a functional design to utilize in our final system. Our quick design strategy allows us to have a working robot throughout many design stages. A functional prototype improved our time in the overall system evaluation.

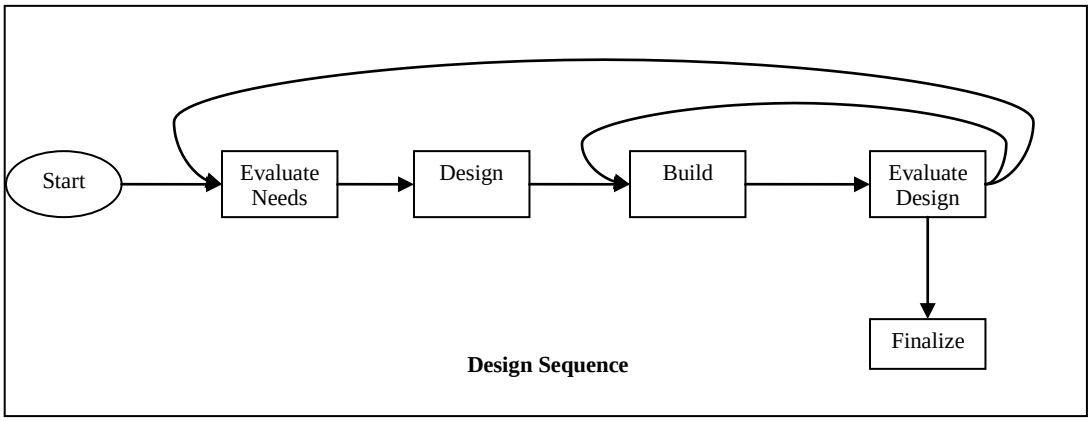


Figure 1: Process Flowchart

The UT-Austin team is composed of voluntary members of the IEEE Robotics & Automation Society called “RAS” in abbreviation. There are five undergraduates participating in the contest. Overall work hours were approximated into the following breakdown below

3.0 ELECTRICAL SYSTEM

RAS Devil's electrical system can be broken up into four major systems: power, control, sensor, and communication systems. Each system is closely tied to the others so that we can have full control over how our robot performs.

3.1 Power System

Three components make up the power system on the robot: the batteries, motor controller, and the control system power supply. RAS Devil is powered by a pair of 12 V lead acid batteries run in series to produce 24 V. Each battery can hold 17 Ah of charge while having the ability to provide several hundred amps of current when needed. We are using a Sabertooth Dual 25A motor controller to control our two DC drive motors which have the ability to charge the batteries when the motors are back-driven. Our control system is powered by a picoPSU power supply. The picoPSU can provide 125W of power and has the ability to trigger the the startup and shutdown sequence of the computer's motherboard. Figure 2 shows our power system block diagram.

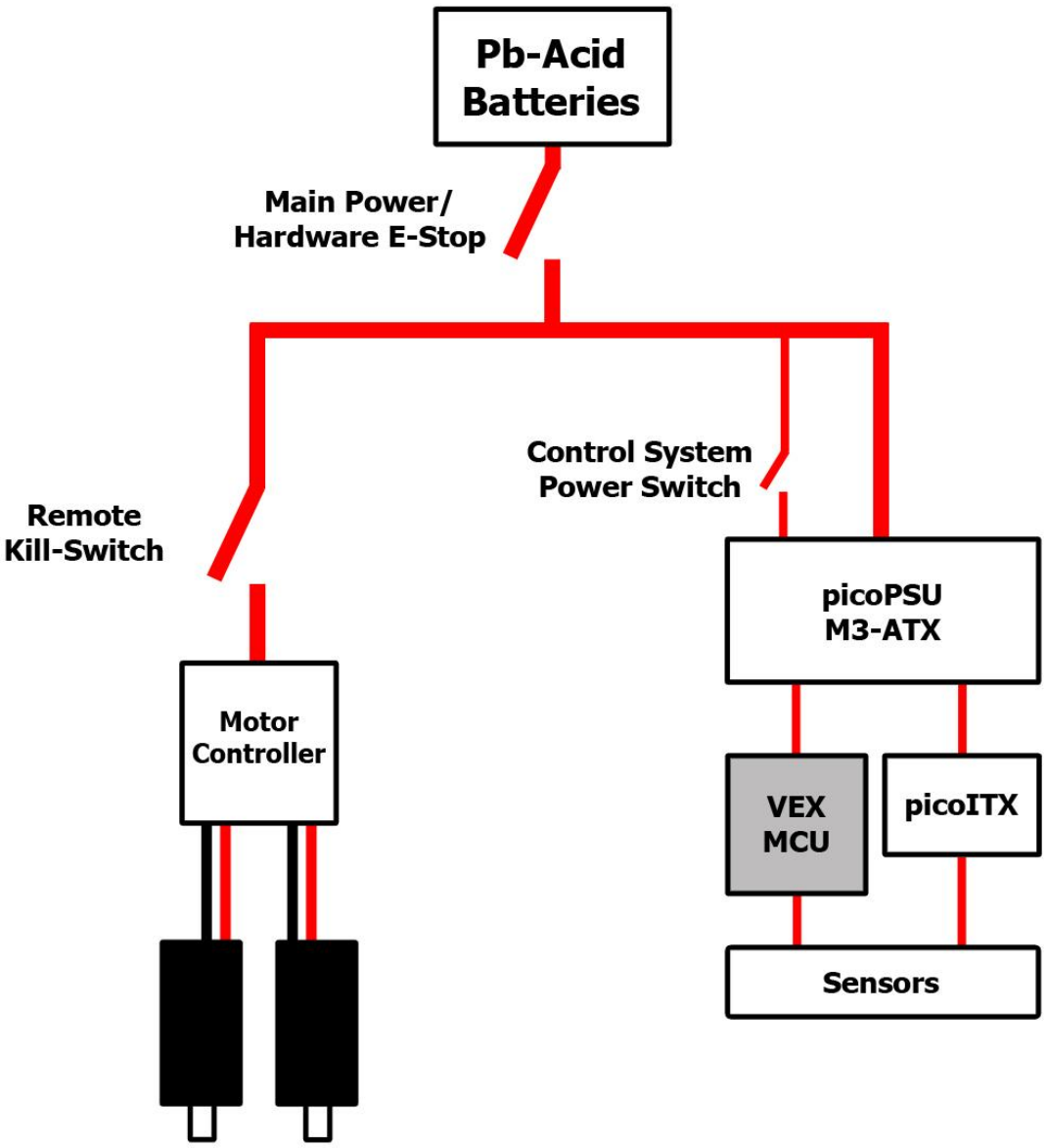


Figure 2: RAS Devil power diagram. Ground connections not shown.

The batteries directly power the motor controller and the picoPSU. When the control system power switch is closed, the picoPSU powers the VEX microcontroller and the picoITX. All the robot's sensors draw their power from either the microcontroller or the picoITX. The safety benefits of the three switches in the power system will further be discussed in section 7.0, Safety.

Monitoring the state of our power system is very important. We can gather valuable information about the state of the robot by monitoring the battery voltage and the current draw from the motors. The battery voltage is monitored by the microcontroller through a simple voltage divider that reduces the battery voltage by a factor of three so the microcontroller ADC can measure it. Motor current is also measured

by Allegro current sensors, whos ouput is measured by the microcontroller as well.

Overall our power system is very simple. In years past we have had very complicated wiring, multiple separate battery sources, and many extra and unnecessary power switches. Having such a complicated power system lead to failures in our previous robots that prevented the robots from operating correctly, even before the code could malfunction itself. We hope to improve the reliability of our robot's power system, which will also improve our overall robot's success.

3.2 Controller

From past experience, our team wanted a control system which would be reliable, fast, and easy to access remotely. In our previous robots, we have used old laptop computers which did not survive the beatings of an outdoor robot, and were not easily repairable or upgradable. For this reason, we chose to use a picoITX motherboard with a VIA C7 1GHz processor. It is lightweight and very compact, so we did not have any trouble making room for it on the robot. We also purchased a 1GB DD2 RAM card and a 30GB solid state SATA hard drive, which is much more robust than a conventional hard drive as it is not subject to mechanical failure.

A Vex Robotics controller unit is used for controlling the motors and encoders. The unit contains two microcontrollers, which operator in a master/slave configuration. The slave microcontroller handles all low level interrupts necessary for reading the encoders, PWM signal generation, and serial communication and is preprogrammed. The master microcontroller provides the interface through which the user can easily interface with the sensors and serial port. Communication with the microcontroller is handled through the RS232 port on the motherboard. The motherboard sends commands to reset or get the latest encoder values, or set the robot speed. The microcontroller responds first with an echo of the command received, and then in the case of the encoders, the encoder values.

The Vex Robotics controller also has a built in RC receiver, making it very easy to drive the robot with a hand-held remote control unit without having to turn on the computer. A simple toggle switch connected to one of the digital I/O pins toggles the signal source between the RC transmitter and the picoITX.

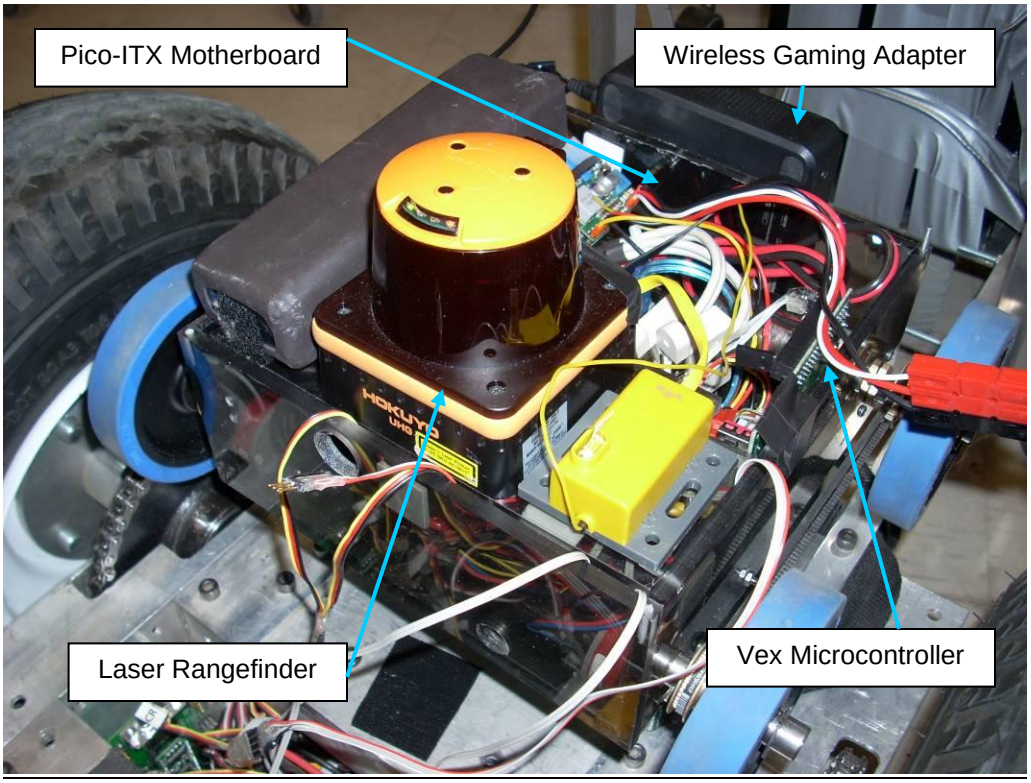


Figure 3: Electronics System

3.3 Wireless Communication

To make our system easier to develop from remote workstations, and eliminate the need for a keyboard, mouse, and monitor connected to the robot, we wanted to have wireless communication. Initially, an EDIMAX EW-7711UAn Wireless USB Adapter was used to connect the picoITX to a Linksys WRT54G router so that we could remotely connect to the robot from another computer connected to the same router. This solution worked, but was not ideal because the Wireless Adapter had a very limited range and when it disconnected from the router, it would not automatically reconnect. This required hooking up a monitor, mouse, and keyboard to the robot every time we needed to reconnect to the wireless network. To fix this issue, the robot now utilizes a Linksys WGA600N Wireless Gaming Adapter, connected directly to the picoITX ethernet port. The gaming adapter is setup to automatically connect to the main router, and also provides a much greater range for the robot to travel which is very helpful.

3.4 Sensors

Choosing the right sensors was key to making a system that worked well as a whole. Significant effort was spent examining each datasheet of each sensor before its purchase to ensure that it would work the way we desired.

3.4.1 Laser Rangefinder

Our team chose to purchase a Hokuyo UHG-08LX scanning laser rangefinder for its 8m range, and relatively low price tag compared to the more common SICK rangefinders. With 1mm trace resolution, a 270 degree field of view, 0.36 degree angular resolution, and 15Hz scan rate, it was determined to be suitable for obstacle avoidance in this competition.

3.4.2 Camera

Line detection is done with a 2MP Logitech Quickcam Pro 9000, capable of up to 1600x1200 pixels and up to 30 frames per second. It also features an autofocus system which eliminates all manual tuning, which was one issue we had in previous years.

Figure 4: Logitech Quickcam Pro 9000

3.4.3 Quadrature Encoders

Our instantaneous localization algorithm is done using odometry, which integrates values from two quadrature encoders (Grayhill 63R256), which are directly coupled to the left and right wheel shafts. The casing of the encoder is completed enclosed allowing it to work in any lighting without calibration, and also increasing durability. By looking at both the rising and falling edges of each signal, we can obtain 512 increments per revolution. With 16in diameter wheels, movement of the robot is measured with 2.5mm resolution.

3.4.4 IMU

The IMU is a Microstrain 3DMG IMU. The 3DMG uses tri-axial accelerometers, magnetometers and angular rate sensors to provide accurate roll, pitch, and heading information. The IMU provides data at rate of 75 Hz with a precision of 0.01 degrees [2].

3.4.5 GPS

We have tested two different GPS receivers on the robot. Eventually we decided on the Garmin 72 GPS, which gives WAAS capability and a 3 meter accuracy [1]. We communicate with the GPS using a RS-232 serial link. The GPS has handheld features such as embedded button controls allowing the user to view all information on the GPS for quick debugging.

4.0 MECHANICAL SYSTEM

Before designing this year's mechanical system, our first step was to carefully review the strengths and weaknesses of last year's robot. At the actual competition, our primary strengths were reliability, and robustness, but our ability to navigate was complicated by the fact that we used skid steering. At a slow speed, which is what the software called for, the response from the motors was far from linear and since we lacked a velocity control system, the system was jerky. As a result, we designed this year's drive system to operate at a lower maximum speed for more torque. We also went to a two wheel drive system with a caster in order to drastically reduce the resistance to turning, thus making it easier to control and maneuver around obstacles.

Since the motors were much faster than we wanted the wheels to spin, gearboxes were required. We chose to use the gearboxes out of a Dewalt drill for their reliability, and coupled them with some 2.5" CIM motors, which we had available from a previous competition. Two Dewalt/CIM assemblies were made for each side, and were linked to the main driveshaft using sprocket and chain. Our motors free spin at 5310 rpm with 343 oz-in torque. This is reduced by 60:1 via the 12:1 gearbox and the 5:1 sprocket reduction. With two 16" driven trailer tires, we should move approximately 4.3 mph with a pushing force forward of 400 lbs, much more powerful than the pushing force of 200lbs last year. With these specifications we will meet the max speed limit of the vehicle with an ample amount of torque for climbing the fifteen percent grade ramp. In addition to the torque, the two 16" wheels have high traction to make the climb on the ramp easier.

For mounting structures, we decided on a polycarbonate frame to house the majority of the electronics. Last year we had everything in a box underneath the payload, so this year we made everything above the payload for easier access. For the speed controllers and power distribution, we built a small polycarbonate compartment underneath the robot. The electronics mounting system was created to keep the center of gravity relatively low. As a result, the payload, speed controllers, breaker panel, microcontroller, and scanning laser range inders all had to be mounted relatively close to the bottom of the chassis to keep it from tipping over. The laptop was also mounted above the payload to allow for easy access while testing the system. From the base of the robot, two vertical aluminum square extrusions were mounted. Along the entire length of the square extrusions, mounting holes were drilled so the range finders can be easily attached at different heights. The range finders themselves are mounted on a piece of flat aluminum bent at 22.5 degree angles. Since the sonars have a range of 45 degrees, this bending strategy gives us double coverage on obstacles 8 feet from the front of the vehicle with a total range of 180 degrees. The IMU, Camera, and GPS had to be mounted high for optimum sensor performance, so we put a tower above the two front drive wheels for the mounting of these three components.

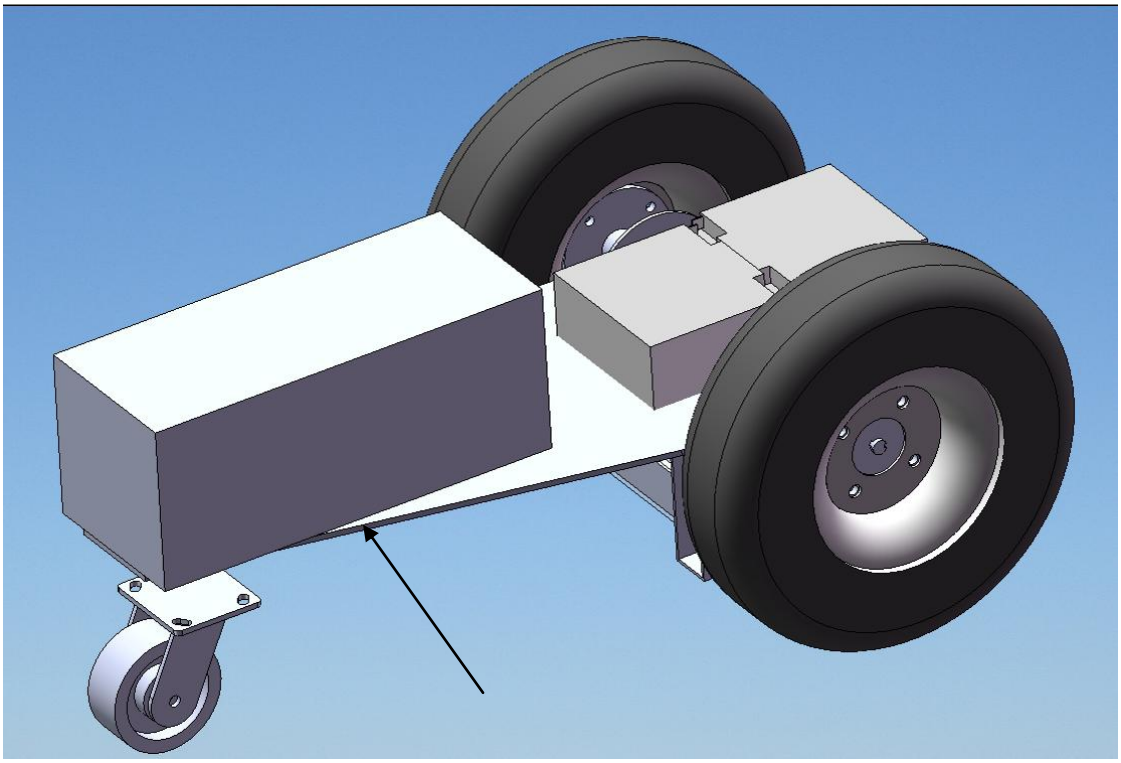


Figure 5: Solidworks System Level Model

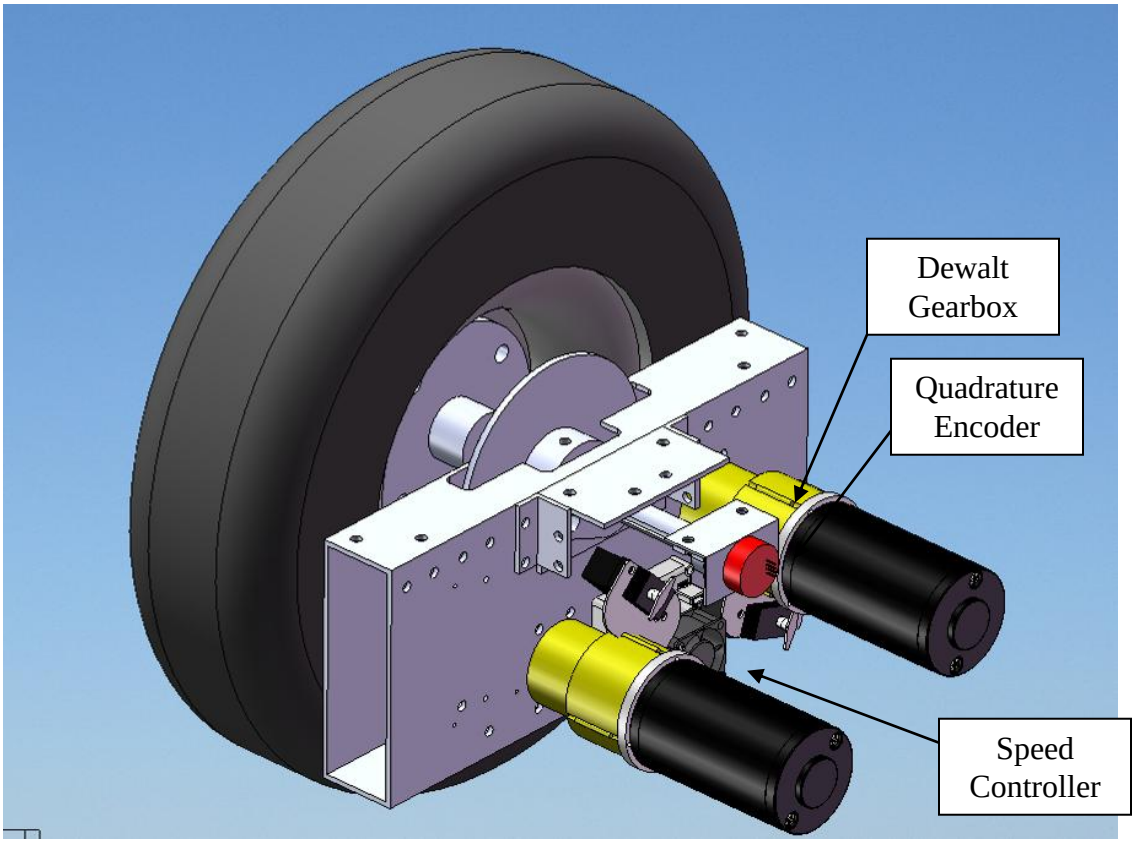


Figure 6: Solidworks Drivetrain System

5.0 SOFTWARE

The main software architecture is built using the OpenGL Utility Toolkit, or GLUT. This API was chosen because it fuses computer industry standard OpenGL with a mature, flexible, and simple framework for designing event-driven programs. Since the ultimate goal is autonomy, the simplicity of GLUT is not restrictive because our User Interface need not be overly complex. That said, we use OpenGL to visualize the robot's sensory input, and designing the application as event-driven aids debugging by enabling easy on-line calibration of the robot. Specifically, we set up a menu, keyboard, and mouse callbacks to control the robot's hardware (eg, motors) directly, start/stop data aquisition, calibrate variables used by our Simultaneous Localization and Mapping (SLAM), vision, and navigation algorithms, and the display of sensor inputs and all relevant debug data. Furthermore, we take advantage of GLUT's robust event processing loop to schedule data aquisition and processing in a seemingly multi-threaded, asynchronous fashion without having to deal with any of the pitfalls, gotchas, and intricacies of creating and managing multiple threads. Whenever the GLUT program is not busy with handling user input or displaying sensor and debug info (these actions take very little time and CPU power), it polls the sensors for new data and uses it to update the robot's perceived state.

5.1 S.L.A.M.

The robot continuously scans its surroundings using a laser rangefinder in order to update an occupancy grid to be used by the navigation code. It does so in conjunction with encoders and a compass which serve to provide a pose estimation. After this data is received from each hardware device, it is all passed to another function which performs simultaneous localization and mapping to update the occupancy grid. In order to obtain the initial pose estimate, the following algorithm is used:

LeftDistance = ((LeftEncoder - LastLeftEncoder))/TICKS_PER_METER RightDistance = ((RightEncoder - LastRightEncoder))/TICKS_PER_METER Distance = (LeftDistance + RightDistance)/2.0
Theta = Theta - (RightDistance - LeftDistance)/WHEEL_BASE
Y = Y + Distance*cos(Theta)
X = X + Distance*sin(Theta)

This algorithm will work for any robot with a tank style drive system. TICKS_PER_METER and WHEEL_BASE are two constants dependent upon the size of the specific robot being used. Although encoders are good for computing relative heading in the short term, they are unreliable in the long term due to wheel slippage, limited encoder precision, finite computational precision, and any other factors that introduce error. Therefore, we periodically re-evaluate our absolute heading using a digital compass to prevent buildup of error over the run of the robot.

Once the robot's pose is determined, the occupancy grid is adjusted. Each cell in the grid represents a 10cm by 10cm square in the world and contains a value representing the probability that the cell is occupied. Each cell is initialized with a value of zero for unknown, and ranges between positive and negative infinity. When the rangefinder traces are superimposed on the occupancy grid, the value of the cell containing the endpoint is decremented to represent a decreased probability that the cell is passable, and every cell between the robot and endpoint is incremented to represent an increased probability that those cells are passable. The amount that the probability is adjusted for each cell depends on how much of the cell is traversed by the trace: the probability is adjusted more for cells that the beam passes straight through the middle than for cells whose corner is barely grazed by the trace. From a graphical perspective, all grid cells start out gray and gradually turn black if the cell is occupied, or white if the cell is empty.

5.2 Vision

The vision algorithm uses an open source computer vision library developed by Intel called OpenCV. The OpenCV process is for acquiring and processing the image from the webcam. We used C++ and the OpenCV library to process the image and extract information about hazards. We tried to remove as many user defined parameters as possible and eliminate false positives due to noise while at the same time keeping our algorithm as robust as possible.

The vision algorithm first gets the raw image from the camera and rescales the image size to 160x120 pixels. This resolution was empirically chosen because the processing time and information losses were deemed acceptable. The algorithm converts the image to grayscale, splits the image into two sub-images and begins the search for lines in each of the two half images.

The line detection algorithm uses the OpenCV implementation of the Hough transform to find the three most prominent lines in each image. If no prominent lines are in the image, the Hough transform will often detect lines around the square edges of the image or detect non-stationary lines due to noise, in which case the data is ignored. Note that the Hough transform is much more robust than any methods that binarize an image using a user defined intensity threshold because there are no parameters to be adjusted.

The line extraction algorithm then converts the three most prominent lines into robot centric coordinates using an inverse perspective transformation. This transformation allows us to represent the detected lines in a robot centric frame of reference, which is of more value for control purposes than a camera centric frame of reference.

To filter spurious data, the angle and perpendicular distances of the three lines are found and the lines that have the most common properties are averaged. This filter, combined with the filter that ignores edge

lines, greatly reduces the number of false positives the algorithm encounters. Shown below in Figure 8 is a distorted grass/concrete sample image which displays the final Hough green line and binary image output of the brightness threshold.

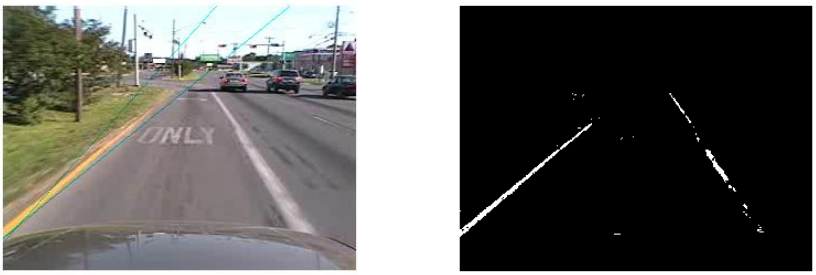


Figure 7: Vision Processing Image on road

We also tested in an all concrete driving scenarios to prove the algorithm will be robust for a wide variety of terrain. Notice how the grass line is much easier to distinguish than the highway line. As a stable testing platform we implemented “OpenView Vision”, a Labview-OpenCV testing environment that quickly test and modify parameters on the fly [11].

5.3 Navigation and Obstacle Avoidance

Since the map of the course and placement of obstacles is not available ahead of time, it is impossible for the robot to plan its path from start to goal. Taking advantage of the fact that the road does not fork, the robot's goal is simply "go forward". As the robot advances through the course and obstacles are discovered and marked on the occupancy grid, the navigation code uses a simple bugging algorithm to circumvent cells that are impassable. For the navigation challenge, keeping track of which way is "forward" can be tricky as the road turns or if the robot is forced to backtrack after encountering a dead end because of obstacle placement. The main heuristic for determining which way is "forward" is to remember the traversed path and to avoid going towards regions that have already been explored. In the aforementioned case when the robot is forced to backtrack even as the heuristic "forward" into the dead end, the backtracking marks the path to the dead end as having been explored a second time, so the robot becomes more reluctant to fall into the same trap again since it avoids paths it has already taken. Eventually after sufficient backtracking, the robot comes to a point where it can go forward again without being forced into the dead end a second time, so it gets back on track towards the goal by exploring new regions of the course.

6.0 PREDICTIONS

Our low-cost drive train and configurable electronics will provide a reasonable opportunity for success at the 2009 competition. The motors and gear box in this drive train allow a maximum speed of around 5 MPH. However, as the robot was designed for outdoor use, we predict that we should be able to carry

the payload up 15% grades with no problem. We predict that the 24V lead acid battery will provide six to eight hours of battery life.

Our camera can see objects in excess of 20 feet depending on the ambient brightness and the contrast, but our sonars have a maximum range of 10 feet. Our GPS has a best accuracy of 2 meters, but in practice on a clear day, far from buildings, we achieve an accuracy of 3-5 meters. Our waypoint navigation is bounded by our GPS navigation.

7.0 SAFETY

Safety has been a primary concern in many decisions throughout the design process of the robot. Human interaction with the robot is the main focus of robot safety. A testing operator will be present with an electrical safety stop (E-stop) whenever the robot is in operation. This hardware E-Stop is located directly after the batteries, see figure #. It is meant to cut power to every system of the robot in the event of a serious problem or emergency. A less extreme option for disabling the robot is in our remote kill-switch. The remote kill-switch is wired so that it only cuts power to the motor controllers, disabling the robot's movement. This allows the user to disable the robot's motors while keeping the control system running, which is very useful for testing and debugging. In addition to the hardware E-stop and the remote kill-switch, we designed the software to immediately stop robot operation if our testing communication link is lost. In the early construction phase, we incorporated safety rules such as eliminating sharp edges, adding bumpers and covering exposed wires. All the high-current power wires use Anderson PowerPole connectors to avoid short circuiting batteries. In general, the robot has been designed for human interaction to ensure public safety around the RAS Devil.

8.0 COST

Quantity	Part	Retail Price	Our Price
1	VIA EPIA N 10000 Pico-ITX Motherboard, 1GB RAM, 30GB SATA Solid State Drive	$350	$0
1	M3-ATX PicoPSU Form Factor Intelligent Vehicle DC-DC Power Supply	$80	$0
1	Microstrain 3DM-G IMU	$1,495	$0
1	Logitech QuickCam Pro 9000	$90	$90
1	Garmin GPS 72 Unit	$110	$110
1	Hokuyo UHG-08LX Laser Rangefinder	$3950	$0
2	GrayHill 63R Encoders	$60	$60
2	IFI Victor 884 Speed Controller	$350	$350
1	Vex Microcontroller	$150	$150
1	Linksys WGA600N Wireless Gaming Adapter	$80	$80

1	Linksys WRT54G Wireless Router	$70	$70
4	CIM Motors	$120	$120
4	Dewalt Hand Drills (for gearboxes)	$220	$220
2	16” Trailer Tires	$70	$70
1	Aluminum for frame	$150	$150
1	Polycarbonate for Electronics Housing	$50	$50
1	Sprockets and Chain	$100	$100
Total		$7,495	$1,620

Table 2: System Level Budget

9.0 CONCLUSION

RASManian is a culmination of effort of the RAS team from the University of Texas at Austin. As a third year team into the competition, our contributions to software infrastructure and low-cost electronic design are quite portable. In addition, our versatility and flexibility in software design and multiple mechanical testing platforms will be a defining aspect of our presence at the IGVC competition.

10.0 REFERENCES

[1] Garmin 72 GPS datasheet , https://buy.garmin.com/shop/shop.do?pID=214

[2] 3DMG IMU datasheet, http://www.microstrain.com/3dm.aspx

[3] Grayhill Encoders http://lgrws01.grayhill.com/web/images/ProductImages/I-37-41.pdf

[4] Logitech QuickCam Orbit Spec Sheet, http://www.logitech.com/index.cfm/products/details/US/EN,CRID=2204,CONTENTID=10628 [5] Acroname Sonar SR04 spec Sheet, http://acroname.com/robotics/parts/R93-SRF04.html

[6] D. Fox, W. Burgard, S. Thrun, “The Dynamic Window Approach to Collision Avoidance” IEEE Robotics & Automation Magazine, 4(1), March 1997.

[7] C. Schlegel, “Fast Local Obstacle Avoidance under Kinematic and Dynamic Constraints for a Mobile Robot” Proceedings of the 1998 IEEE-RSJ Intl. Conference on Intelligent Robots and Systems Victoria, B.C., Canada, October 1998.

[8] M. Yguel, O. Aycard, C. Laugier, “Efficient GPU-based Construction of Occupancy Grids Using several Laser Range-finders” Proceedings of the IEEE-RSJ Intl. Conference on Intelligent Robots and Systems, 2006.

[9] S. LaValle, *Planning Algorithms*, Cambridge University Press, 2006

[10] S. Thrun, W. Burgard, D. Fox, *Probabilistic Robotics*, 2005.

[11] OpenView Vision Code, http://ras.ece.utexas.edu/learning.php