MISSOURI S&T University of Science & Technology

ROBOTICS COMPETITION TEAM

"Helping Robots Help Themselves"

The Missouri University of Science and Technology Robotics Competition Team presents:

JΩTRON

Intelligent Ground Vehicle Competition IGVC

MISSOURI S&T

For entry in the 2013 Intelligent Ground Vehicle Competition

Faculty Advisor Statement:

I herby certify that design and engineering changes made to this vehicle by the current student members of the team have been significant, including major development on the software, and several hardware upgrades, and that every member has made a significant contribution that would equal or surpass that of a senior design credit.

_____

Dr. Donald Wunsch
Senior Advisor

Contents

1. Introduction

The Missouri University of Science and Technology (Missouri S&T) Robotics Competition Team is pleased to present *Jomegatron* as an entry in the 2013 Intelligent Ground Vehicle Competition. Jomegatron was named after the Missouri S&T mascot Joe Miner. The robot will be making its third appearance at the IGVC as the ninth consecutive entry from the Missouri S&T Robotics Competition Team. Over the past three years, Jomegatron has become a reliable platform for testing new code, hardware, and mechanical components. As a veteran of two IGVCs, the robot serves as a teaching tool for new members. The study of its strengths and weaknesses has proved invaluable as the team works to build better projects and train new generations of students in the field of robotics. The following report is an analysis of Jomegatron's design, a description of the significant changes made since the 2012 IGVC, and an overview of the S&T Robotics Competition Team structure.

1.1 Team Structure

The Missouri S&T Robotics Competition Team operates through the S&T Student Design and Experiential Learning Center (SDELC), which provides resources and support to all fourteen of the school's student-run design teams. The team is comprised of roughly 30 Missouri S&T undergraduates from a variety of disciplines. A full member list can be found in Appendix A. The team is run by an executive board consisting of a president, vice president, secretary, treasurer, and public relations manager. A new executive board is elected each spring and officially takes office immediately after the IGVC. The executive board appoints three division leaders to oversee the mechanical, electrical, and computing divisions of the team. The use of divisions has allowed the team to break down the wide field of robotics into more specific disciplines for members wanting to focus their interests. Figure 2.1.1 shows the overall team structure.

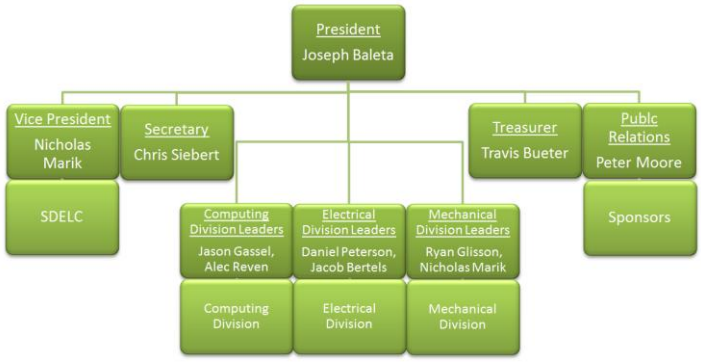


Figure 1.1.1: Robotics Competition Team Heirarchy

2. Design Process

2.1 Goals and Planning

Jomegatron's performance at the 2012 IGVC was not as strong as the previous year. This was due to software and electrical problems. Jomegatron's batteries were found to be having trouble holding a charge and one had to be replaced at competition. There were also problems with the emergency stop system that had to be fixed before the robot could qualify. But the greatest difficulties and main reasons for the year's poor performance involved the software, particularly getting the camera to work consistently and calibrating Jomegatron's response in certain situations.

The competition highlighted the robot's weaknesses, making clear what would need to be fixed in the 2012-2013 project cycle. The team had two main goals for Jomegatron at the beginning of the 2012 fall semester:

1. Create a more reliable electrical system with a larger charge capacity.
2. Rewrite navigation code to be more reliable in competition situations.

2.2 Execution

Jomegatron was upgraded using a simple six-step strategy outlined in Figure 2.2.1. Although the basic problems with the software were made known during competition, only more field testing could further refine the scope of problems and their causes. The process has as few steps as possible. The upgrading of Jomegatron had to be done quickly and effectively as the team was busy designing and manufacturing a new robot. The upgrade process is cyclical. Solutions are not minimally tested and then assumed to work entirely, but put through the entire test process again which involves trial runs in a competition setting. This is a notable difference from what was done with Jomegatron in the past. Permission was granted to paint white lines in regularly mowed fields at a public park. Construction cones and barrels were then placed within the lines and Jomegatron was sent through the obstacle course several times.

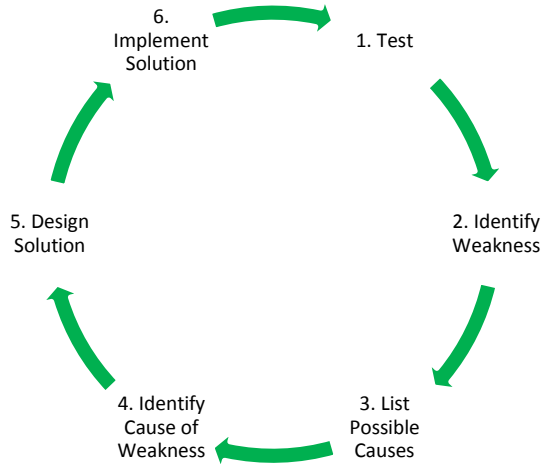


Figure 2.2.1: Problem solving strategy for upgrading Jomegatron

3. Vehicle Upgrades/Innovations

By building on the lessons of the previous year the team has created a robot fine-tuned to the needs of this competition. The problems of the 2012 model have been replaced with new capabilities and increased reliability. Underneath the Lexan shell of the 2012 model there have been major innovations in both hardware and software.

3.1 Electrical Upgrades

At the 2012 IGVC Jomegatron was upgraded with a new emergency stop (E-stop) control board. During the board's replacement, some upgrades were made. The E-stop board is triple redundant, as was its predecessor. The computer, remote control, and physical buttons can all be used to E-stop the robot. There is now a switch on the E-stop board so that software control can by bypassed in the event that a board failure occurs. Even if software control is bypassed, the hardware buttons still function properly. The E-stop board also serves as a power regulator, with the ability to provide 24V for the E-stop switches and 5V for the on board micro-controller. The 5V supply can also be used to drive other small boards that the team may want to add in the future.

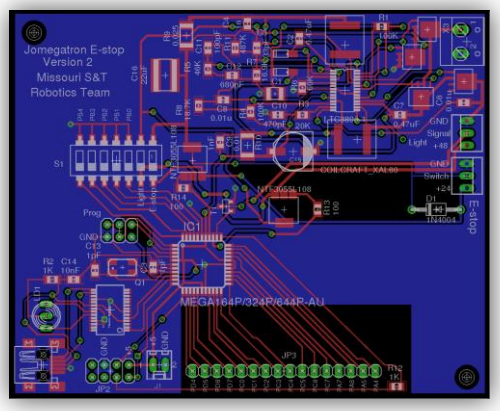


Figure 3.1.1: Custom E-Stop Board

The E-stop board has two methods of communication: serial and USB. The E-stop board includes a USB to serial converter allowing the board to be plugged directly into the computer. The second serial port also allows the board to communicate to a remote E-stop switch. The board is able to send data back to the remote with battery measurements or other information the team wishes to display.

A new GIGBYTE Intel motherboard was installed to replace the motherboard Jomegatron carried for the past two years. The motherboard was slowly dying, possibly due to the shaking experienced by the robot during travel over rough terrain. The batteries were also replaced with four new 12 volt units to power the 48 volt system. They can hold more charge than the older batteries, allowing Jomegatron to run much longer. Jomegatron now has multiple hours of drive time as opposed to just two.

3.2 Software Upgrades

The potential field navigation system was completely replaced with a new system known as A*. The A* algorithm is widely used for path finding and graph traversal, the process of plotting an efficient path between points called “nodes.” Unlike the potential field system which was designed to mimic gravity in a way, A* uses a best-first search and finds a least-cost path from a given initial node (the robot) to a goal node (the next GPS waypoint). As A* traverses the graph, it follows a path of lowest expected total distance, keeping a sorted priority queue of alternate path segments along the way. However, A* can’t see the entire map at once, which means the map and route must be continuously modified as position changes.

4.0 Mechanical Design

4.1 Frame and Shell

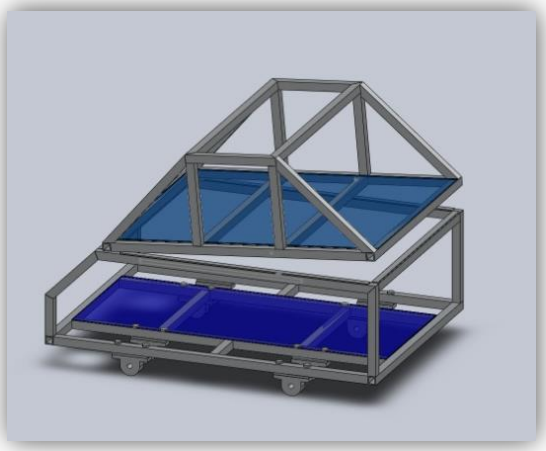


Figure 4.1.1: Solidworks Frame Design

Jomegatron’s frame is made out of square inch aluminum tubing and is 28” wide by 40” long. Aluminum was chosen for light weight while square tubing allows for easy installation of brackets and fixing components. The frame is divided into an upper and lower half. The upper half is attached to the bottom using a hinge along the front side and two hydraulic springs, much like the trunk of a car. This allows easy access to the bottom portion of the robot for placement and removal of large components. Jomegatron’s frame size was based around IGVC regulations. At the beginning of the design process, it was understood that the frame had to meet

all regulations while still being able to travel through doorways and openings at least 30” wide. The efficient placement of components inside allows the frame to carry more while still being the same size as many competitors.

The inner components are protected by a Lexan shell made of several panels. Each panel has three to four magnets epoxied to it and attaches to steel angle brackets pop-riveted along the frame. This allows for easy removal of the Lexan panels so that components within the robot can be reached in multiple ways.

Figure 4.1.2: The Lexan panels of the shell removed from the robot

4.2 Drive Train

Jomegatron was originally designed with a skid-steer drive. This system made turning far more difficult than expected and was replaced with a differential drive. Two front drive wheels steer the robot while two rear casters provide stability. This change greatly improved Jomegatron’s ability to handle various surfaces.

4.3 Camera Mount

Figure 4.2.2: Camera mount

The camera mount reaches up to a height of 5 feet in order to give the camera a top view perspective. The camera mount was designed to be very adjustable. The top bar of the mount can be unbolted and shifted forward or backward while the brackets that hold the camera allow it to be tilted at any angle. The camera mount was designed to satisfy the needs of the previous year’s software vision setup and to accomodate any changes that could be needed in the future. This adjustable design also works well with the current vision code.

5.0 Electrical Design

Jomegatron features an electrical system with improvements based on several previous years of experience. This system is designed to serve the team for years to come without a foreseeable need of service or significant changes that often were required in the past.

5.1 Sensors

- *Vision Sensors*

Jomegatron uses a single Point Grey Firefly MV camera. This camera operates at a resolution of 0.3 megapixels and is able to supply VGA (640x480) images at 30FPS over a standard IEEE 1394a "Firewire" connection. Standardization of 1394 cameras provides access to all internal setting registers for camera configuration. A removable lens with a 2.2mm focal length provides a 130-degree field of view.

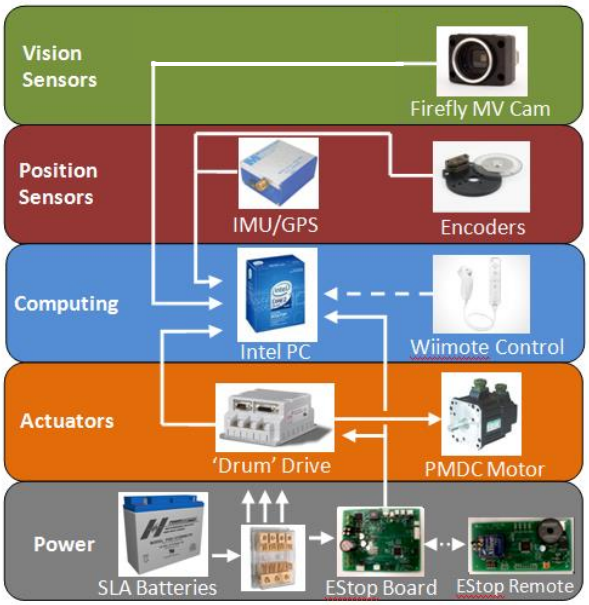


Figure 5.1.1: Simplified System Diagram

- *Position Sensors*

The robot utilizes a Microbotics MIDG-II INS/GPS as its primary position / pose sensor. This device includes a WAAS compliant GPS, a 3-axis accelerometer, a 3-axis rate gyro, and a 3-axis magnetometer. The device is capable of integrating positional information through an on-board Kalman filter and sending revised position / pose information via a serial interface at 50 hertz.

Additional positional information can be derived from the motor controllers, which maintain a running position based on the wheel encoders. This derived positional information is relatively accurate at short time scales, but tends to drift over time due to wheel slippage. The combination of the GPS for long-term absolute accuracy, accelerometers for intermediate accuracy, and wheel encoders for short-term accuracy is used to determine the most probable position at any point in time.

5.2 Computing

Jomegatron carries a full desktop computer to handle all of the vision, mapping, and navigation tasks the team requires for competition. The computer has an Intel quad core processor and a Nvidia GT 430 graphics card (GPU) to aid in vision processing. The graphics card gives the team the ability to perform parallel processing on images, providing a large speed increase. The computer uses a solid state drive to store data allowing memory to be safely accessed while the robot is in motion.

To aid in software development and to give the operator more feedback about the state of the robot, Jomegatron carries a "software workstation" complete with monitor, keyboard, and mouse. This allows the programming team to make changes to the software directly on the robot. This also allows them to view realtime video from the camera and change control parameters on the field. A second monitor on the front of the robot is used to display the feed from the camera or any other image of the operator's choosing. This is used for debugging as well as a promotional tool at public events.

The computer provides more than enough processing power to compute the team's complex vision algorithms. The GPU gives the team the ability to perform parallel processing on images, providing a large increase in speed. The computer can

be accessed through the robot's wireless network or via the workstation. The computer runs Ubuntu Linux, which allows software to be edited, compiled, and tested onboard. The onboard wireless router is configured to connect to external wireless networks, allowing software changes to be pushed to the team's GitHub repository.

5.3 Actuators

Jomegatron uses two permanent magnet synchronous motors to drive the main wheels. The motors are controlled using "Drum" motor controllers from Elmo Motion Control. Each motor is capable of supplying 900W of power, which gives Jomegatron high maneuverability even on rough terrain. Permanent magnet machines are known for their high torque output, meaning that Jomegatron won't stall under load. The motors are rated for 100 volts each, as are the motor controllers. This leaves room to upgrade Jomegatron's power system in the future if the team feels that such an upgrade is necessary.

5.4 Power

Jomegatron has a custom power supply to run all of the computer components on a 48 volt system. The power supply runs directly off four brand new batteries with no converters necessary. This can power monitors, the computer, the camera, speakers, and a router used to remotely access the robot. The power supply is capable of supplying up to 500 watts, which leaves the team a lot of room to add more computer components if they feel it is necessary in the future.

6.0 Software Strategy

Jomegatron's software was programmed in C++ and designed around ROS (Robot Operating System). ROS provides a dynamic and robust transport layer for the robot. The system allows code modules to be linked at runtime, making it easy to edit or replace a single module without the user being required to comprehend the program as a whole.

A simplified overview of the current software architecture may be found in Figure 6.0.1. The software stack is designed to map the environment and navigate to GPS (Global Positioning System) waypoints using the input from a single monocular camera. The software provides a GUI (Graphical User Interface) to display debugging data and allows users to provide input via the GUI or a Wiimote wireless controller. The team uses Git revision

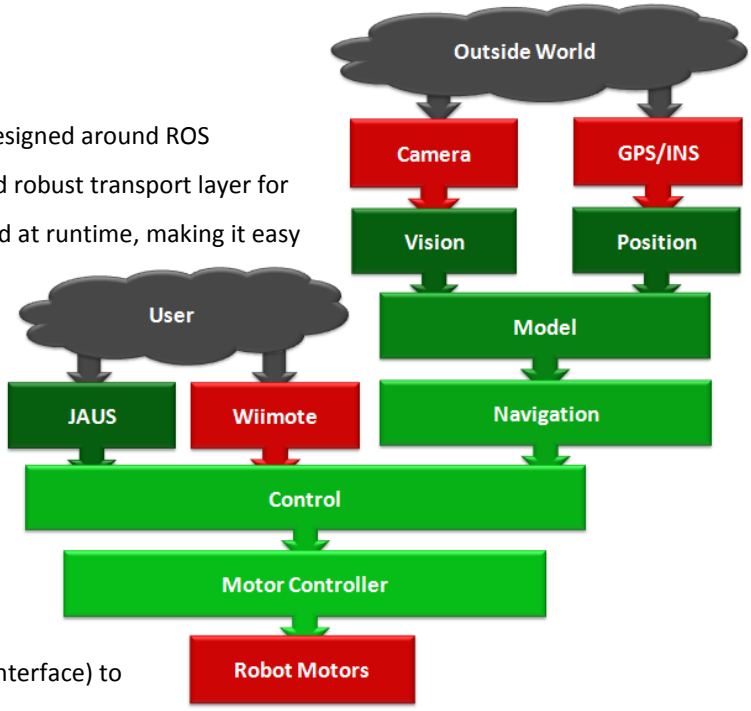


Figure 6.0.1: Software Architecture

control software to track changes. All software is available under the Open Source GPL v3 license. The software may be found in the team's GitHub repository at https://github.com/MST-Robotics/Jomegatron_IGVC.

6.1 Vision

The primary sensor of Jomegatron is a 640x480 resolution wide angle camera. The first step of the vision pipeline is to identify obstacles. The team has developed two primary methods for finding obstacles which are described below. Both methods attempt to identify obstacles based on their color and output an image marking all of the obstacles within each frame.

- *Per Pixel Based*

The per-pixel based method of image segmentation identifies obstacles based on their color characteristics. The user specifies the colors of various obstacles in the frame as well as the color of the grass. The software creates normal distribution curves based on the input color chromaticity and hue components. When images are published from the camera driver every pixel in the image is given a probability of being an obstacle based on where it lies on the distribution curves. The module publishes a grayscale image defining each pixel by the probability that it is an obstacle.

- *Gradient Based*

The gradient based method of image segmentation attempts to first segment the image into regions of continuous color and then uses the statistics of all the pixels in a region to determine obstacles. To do this, the module creates runs of pixels in the X and Y directions that have a consistent change in gradient. The module looks at the second derivative of the image to determine the start and stop of runs. The runs are then linked together into regions defining areas with similar gradients. The statistics of all the pixels in these regions are then compared to the statistics of the training obstacles to determine obstacles within the image. The module publishes a binary image defining the pixels that make up all obstacles within the image. The modules may be launched separately or may be used together with their outputs combined. The blue and red flags of the competition are handled by creating virtual walls to the right of the red flags and left of the blue flags after performing segmentation. Once an image has been found with all of the obstacles marked, a homographic transform is applied to the image. The homographic transform attempts to create a bird's eye view of the area around the robot, correlating obstacles on the

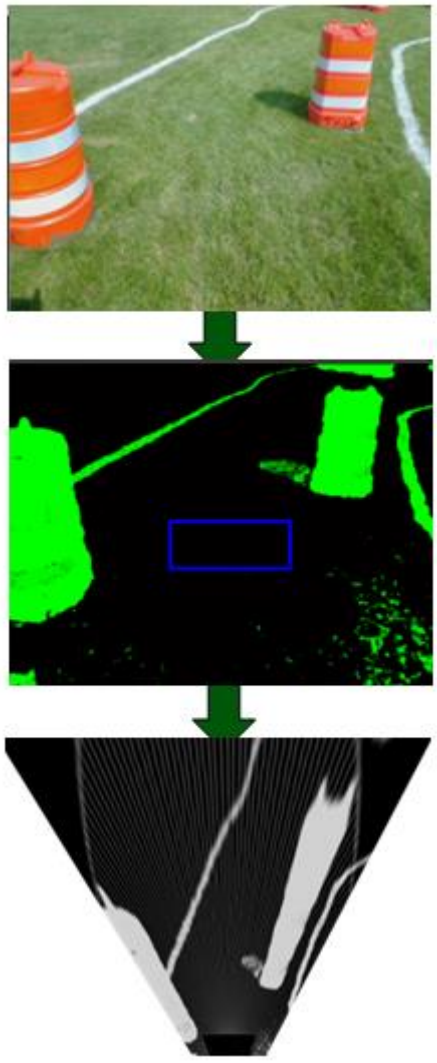


Figure 6.1.1: Image Pipeline

ground plain to their positions in the world. Ray-casting is then performed on the transformed image to give an array containing the distance to the closest obstacle along each angle. Both the homographic image and ray-cast are published. The various stages of the image pipeline may be seen in Figure 6.1.1.

6.2 Position

The position module is in charge of maintaining an accurate account of the robot's position in the world. The module subscribes to the position information being published by the GPS/INS unit and the wheel odometer. The software combines all position information using a Kalman filter to maintain the most accurate position. The module is also in charge of maintaining a list of GPS waypoints. The waypoints may be loaded from a file or may be input by the user via the team's GUI or using the JAUS (Joint Architecture for Unmanned Systems) protocol. The module decides the robot's current target based on priority. If two waypoints are given the same priority, such as those in no-man's land, the program will choose the closest. The user may set time limits on each priority to be sure the robot has enough time to finish the course.

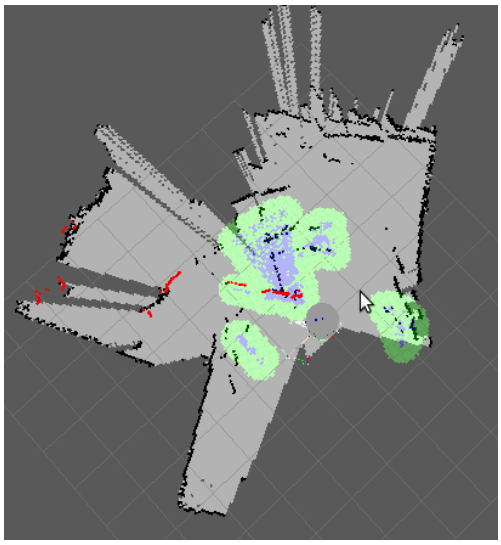


Figure 6.3.1: Map of the Team's Lab

6.3 Model

The model module attempts to create an accurate map of the world. The team's current method uses the gmapping stack which may be found in the ROS repository. The gmapping stack uses a SLAM (Simulations Location and Mapping) algorithm to map the environment. The module subscribes to the ray-cast output by the vision pipeline and the combined position. The software places local obstacle information onto the global map by using the input position and tracking features from frame to frame. The software uses the tracking features to create a more accurate position and aid in future mapping. The module outputs the corrected position as well as a local and global map of obstacles.

6.4 Navigation

The navigation module is responsible for deciding the movement of the robot based on the obstacle map and the current target. The team has two methods of determining the robot's movement which are described below. The methods output desired forward and rotational velocities for the platform.

- *ROS Navigation stack*

Jomegatron's software was designed to be compliant with the ROS navigation stack. The navigation stack looks at the local obstacle map around the robot and determines the path needed to avoid close obstacles. The software then looks at the global map and attempts to find a path that will lead to the next waypoint given by the position node or by a user.

- *A* Algorithm*

As mentioned in Section 3.2, the A* algorithm replaces the potential field model as the main form of robot navigation. The potential field model caused Jomegatron several problems at the last IGVC. It became apparent that either the potential field code would have to be drastically changed or completely replaced. The latter was done in the hope that the new A* system would provide the robot with more direction when dealing with short-term destinations.

6.5 JAUS

The JAUS module was designed by the team to convert JAUS messages into ROS messages. The module was designed to be as general as possible and can easily be used on other robots. The software supports all of the JAUS capabilities and allows the user to pull information from the software and input controls.

6.6 Control

The control node has the final control over what velocity commands are sent to the hardware interface module. The node has several modes of operations that decide the behavior of the robot. The module starts in standby mode and waits for a Wiimote controller to be connect or for the JAUS node to take control. Once the user connects, the user will have the ability to place the robot into either user controlled mode or autonomous mode. In autonomous mode the node will pass the velocity commands published by the navigation software through to the motor controller node, giving the software control over the robot. In user controlled mode the software will interface with the Wiimote or JAUS module and compute velocity commands based on user inputs. The software also launches a ROS tool named RViz. RViz is a visualizer that allows the user to view combined information about the robot's inputs in a three-dimensional virtual environment. The output of the RViz display may be seen in Figure 10. The display is customizable at run time so users may view any debugging information that is being published. Users are also able to subscribe to this data over a network allowing for remote operation. The node interfaces with a text-to-speech library to provide feedback about the current state of the robot.

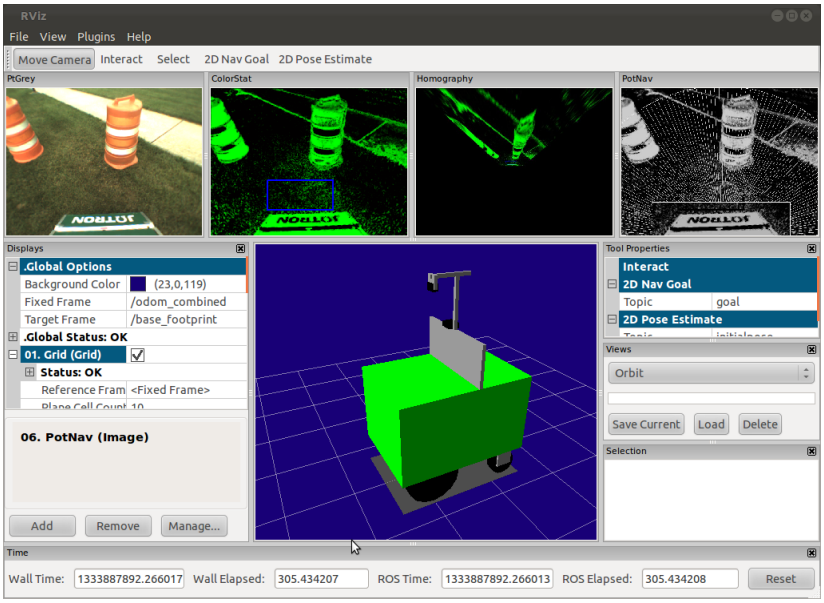


Figure 6.6.1: RViz Robot Dashboard

6.7 Hardware Interface

The hardware interface node converts the desired robot velocity output into wheel velocities. The software uses the computed wheel velocities to create serial commands which are then sent to the motor controllers. The module reads back the encoder information and publishes the wheel odometer. The software also interfaces with the e-stop board, giving the state of the robot and control over the safety light.

7.0 System Integration Plan

Jomegatron's control software was developed by several student members of the team, all working on different modules and levels. Early in the design process, emphasis was placed on higher-level module functional descriptions and interface specifications. After the interfaces were designed and the desired functionality achieved, sub-teams were free to start coding the internals of each module.

Testing was performed at both the module level using test drivers and at the system level using lab and hallway operational tests. Once acceptable behaviors were reliably demonstrated in the lab environment, outdoor operational tests were conducted on a local field designed to replicate the IGVC course.

As several team members participated in past IGVC events, the team was able to re-create all of the challenging features typically found at the IGVC, including solid and dashed white painted lines, various densities of grass / dirt, shadows, sun glare, ramps, cones of various types and colors, snow fencing, plank saw-horses, switchbacks, center islands, dead ends, traps, potholes, and sand pits.

During the spring semester, the team scheduled numerous outdoor tests, each of which focused on a particular set of issues. After each test, the testing sub-team reviewed the results, took notes, and made plans to address any deficient performance observed.

8.0 Performance Expectations

The robustness of Jomegatrons algorithms have been proven multiple times in simulations. The new A* navigation code is much more reliable than the potential field model. The team's predictions along with the design's demonstrated values can be found in Table 8.0.1.

Table 8.0.1: Performance Comparison Table

Characteristic	Design Goal	Demonstrated in Field Test
Max Speed	5 MPH (2.237 M/s)	3.2 M/s (limited to 2.2 M/s in motor controller firmware)
Ramp Climbing Ability	15 degrees	22 degrees
Reaction time - processing rate (sense-think-act loop)	4 hertz	7 hertz
Battery Life	1 hour	1.6 hours
Distance at which obstacles are detected	• Web Cameras : ○ 4 M forward ○ 3 M side – looking ○ 5 M Diagonal • Stereo Camera: ○ 10 M forward ○ 60 degree FOV	• Web Cameras : ○ 4.5 M ○ 3.2 M ○ 5.52 M • Stereo Camera: ○ 12 M ○ 65 degrees
Accuracy of arrival at way points	2 M	1.5 M
Microbotics INS/GPS	$0	$5,710
Wireless Router	$60	$60
Totals	***$60***	***$5,770***

8.1 Complex Obstacles

The control software detects and handles the following special situations. Specific detection / handling methods are described below:

- *Switchbacks*

When a switchback situation is encountered, Jomegatron will seek the path of least resistance. When no such path is obvious within 190 degrees of the front view, Jomegatron will turn 180 degrees and examine the rear environment for a potential exit path. The limited obstacle model memory will discourage Jomegatron from repeatedly taking the same path.

- *Dead Ends*

Jomegatron retains a short-range memory of objects visited in the past few dozen cycles. If Jomegatron encounters a dead end, it will rotate 180 degrees (as in the switchback case above) to look for a more promising path.

- *Traps*

To negotiate traps, Jomegatron employs a method similar to that used to detect and navigate out of dead ends.

- *Potholes*

Jomegatron will avoid all potholes provided they are a sufficiently different color than the grass.

- *Dashed Lane Lines*

Jomegatron's particle-based vision produces notable artifacts when it encounters a partial lane line, allowing it to detect it as a dash and not an opening. These artifacts are detected and subsequently avoided to prevent the robot from entering the opening.

9.0 Safety

9.1 Emergency Stop

With all of the power Jomegatron can supply, there needs to be a way to stop the robot in case the operator loses control. Jomegatron is equipped with a triple redundant emergency stop (E-stop) system. There are two buttons located at hand height on the main frame of the robot. If either button is pressed, the motor controllers shut off and relays switch the motor power lines into a bank of resistors to bring the robot to a quick and easy stop. In addition to the buttons, an AVR micro-controller takes commands from the computer and from a remote control to stop the robot remotely. The computer must reset a count on the micro-controller every second to keep the robot moving. This prevents the robot from running off in the event of a computer glitch or total computer failure. Finally, all software E-stops may be bypassed so that the robot can be driven directly in case of E-stop board failure. The buttons are hardwired, however, and cannot be bypassed so that the robot can be stopped for sure in this way.

Power tracking and E-Stop systems can be handled by an independent AVR microcontroller, ensuring that these critical systems continue to function in the event of a computer failure or when the robot is in a low-power state. Power tracking via current and voltage sensors enables the robot to provide an accurate estimate of its remaining battery life and give warnings when battery levels are dangerously low.

The Wii gaming console's Wiimote has proven effective at controlling the robot for the past two years. The team can link the Wiimote, drive manually to testing areas, then place the Wiimote in standby mode for autonomous testing. Jomegatron's hardware limits its speed to just less than five miles per hour, and the fuses installed on the motors ensure that they receive no more than forty amps. The robot is also programmed to stop upon the loss of a Wiimote or Wireless E-Stop connection or in the event of a crashed program. If the Motors module has not received a request in the last 3 seconds, it will safely stop and turn off the motors. This prevents a single module crash from causing a runaway situation.

10.0 Cost in Dollars/Hours

Jomegatron's design and build process began during the fall semester of 2009 and was completed in spring of 2011. Jomegatron's total financial cost is show below in Table 10.0.0.

Component	Cost to Team	Retail Value
Frame	$500	$500
Motors	$1,300	$100
Gear Boxes	$300	$300
Shell	$250	$250
Wheels	$120	$120
Misc. Hardware	$100	$100
Misc. Electrical	$200	$200
Elmo Motor Controllers	$4,000	$4,000
Batteries(Old)	$260	$260
Batteries(New) (2013)	$130	$130
Charger	$200	$200
Power Supply	$300	$300
Point Grey Camera	$700	$700
Videre Stereo Camera	$1,450	$1,450
Motherboard (2013)	$69	$69
Computer	$800	$800
Blue-tooth Control	$0	$65
Microbotics INS/GPS	$0	$5,710
Totals	***$10,679***	***$15,254***

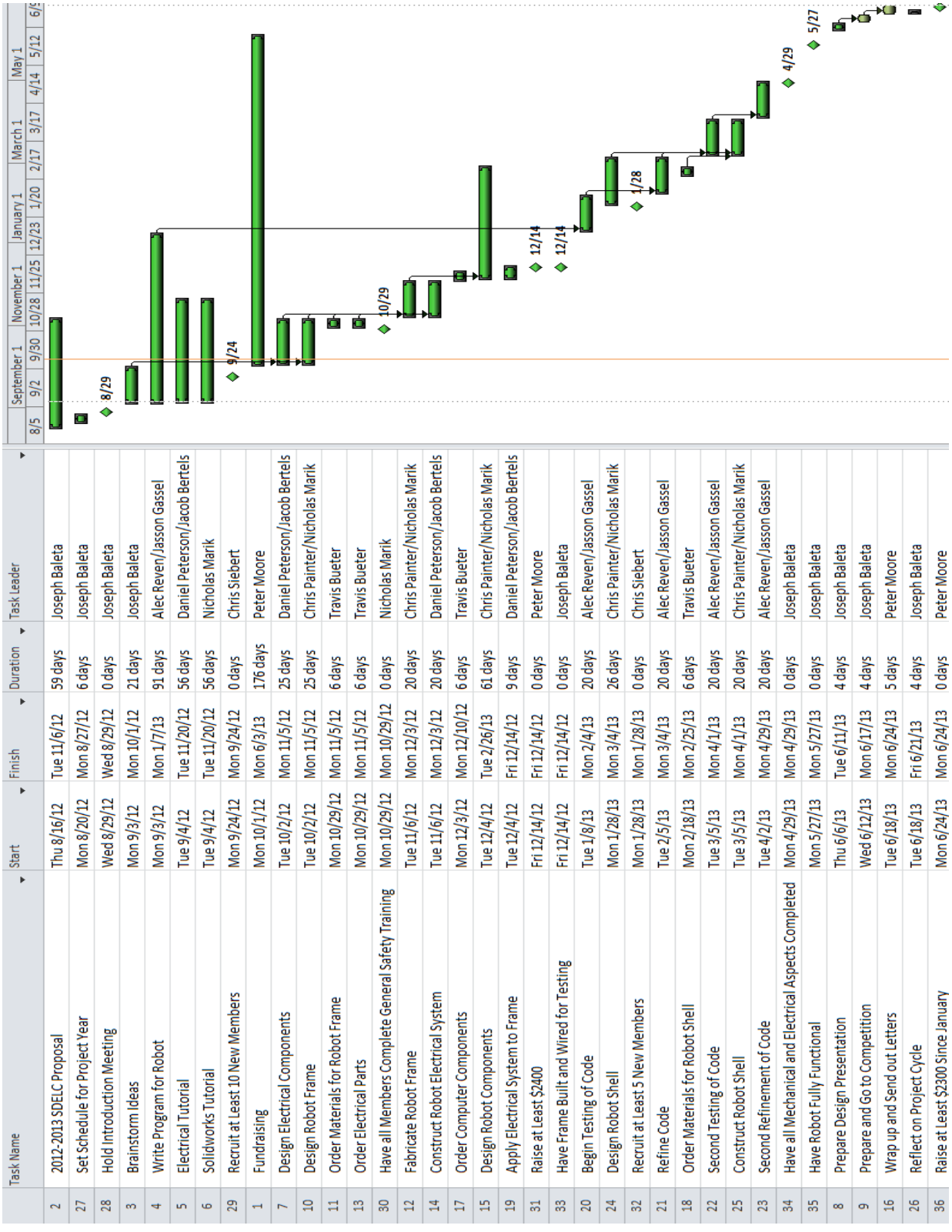
Table 10.0.0: Cost Analysis

Jomegatron's construction and programming required a large amount of time. The first two years of its development cost about 2000 man-hours. Since the team was also focused on designing and building a completely new robot, less time was spent this year on Jomegatron as opposed to the past two. Overall, around 2400 man-hours have been spent on the robot over its three years of existence.

Appendix A: 2012-2013 Member Roster

Name	Position	Name	Division	Name	Division
Dr. Donald Wunsch	Main Advisor	Samuel Pester	Mechanical	Emily Hernandez	Electrical
Dr. Douglas Bristow	Technical Advisor	Michael Lester	Computing	Jessica Greathouse	Electrical
Joseph O. Baleta	President	Ashley Painter	Mechanical	Josh Davis	Electrical
Nicholas Marik	Vice President	Jason Jarquio	Mechanical	Matthew Anderson	Computing
Travis Bueter	Treasurer	Dustin Pieper	Electrical	Alan Gallegol	Computing
Christopher Siebert	Secretary	William Roussin	Electrical	Dzung Tran	Mechanical
Peter Moore	Public Relations	Ryan Glosemeyer	Electrical		
Alec Reven	Computing Lead	Zach Kreuer	Mechanical		
Jason Gassel	Computing Lead	Alex White	Mechanical		
Daniel Peterson	Electrical Lead	Sheldon Harper	Mechanical		
Jacob Bertels	Electrical Lead	Valentine Mbah	Mechanical		
Ryan Glisson	Mechanical Lead	Rory Hayes	Computing		

Appendix B: 2012-2013 Schedule



	Task Name	Start	Finish	Duration	Task Leader
2	2012-2013 SDELC Proposal	Thu 8/16/12	Tue 11/6/12	59 days	Joseph Baleta
27	Set Schedule for Project Year	Mon 8/20/12	Mon 8/27/12	6 days	Joseph Baleta
28	Hold Introduction Meeting	Wed 8/29/12	Wed 8/29/12	0 days	Joseph Baleta
3	Brainstorm Ideas	Mon 9/3/12	Mon 10/1/12	21 days	Joseph Baleta
4	Write Program for Robot	Mon 9/3/12	Mon 1/7/13	91 days	Alec Reven/Jasson Gassel
5	Electrical Tutorial	Tue 9/4/12	Tue 11/20/12	56 days	Daniel Peterson/Jacob Bertels
6	Solidworks Tutorial	Tue 9/4/12	Tue 11/20/12	56 days	Nicholas Marik
29	Recruit at Least 10 New Members	Mon 9/24/12	Mon 9/24/12	0 days	Chris Siebert
1	Fundraising	Mon 10/1/12	Mon 6/3/13	176 days	Peter Moore
7	Design Electrical Components	Tue 10/2/12	Mon 11/5/12	25 days	Daniel Peterson/Jacob Bertels
10	Design Robot Frame	Tue 10/2/12	Mon 11/5/12	25 days	Chris Painter/Nicholas Marik
11	Order Materials for Robot Frame	Mon 10/29/12	Mon 11/5/12	6 days	Travis Bueter
13	Order Electrical Parts	Mon 10/29/12	Mon 11/5/12	6 days	Travis Bueter
30	Have all Members Complete General Safety Training	Mon 10/29/12	Mon 10/29/12	0 days	Nicholas Marik
12	Fabricate Robot Frame	Tue 11/6/12	Mon 12/3/12	20 days	Chris Painter/Nicholas Marik
14	Construct Robot Electrical System	Tue 11/6/12	Mon 12/3/12	20 days	Daniel Peterson/Jacob Bertels
17	Order Computer Components	Mon 12/3/12	Mon 12/10/12	6 days	Travis Bueter
15	Design Robot Components	Tue 12/4/12	Tue 2/26/13	61 days	Chris Painter/Nicholas Marik
19	Apply Electrical System to Frame	Tue 12/4/12	Fri 12/14/12	9 days	Daniel Peterson/Jacob Bertels
31	Raise at Least $2400	Fri 12/14/12	Fri 12/14/12	0 days	Peter Moore
33	Have Frame Built and Wired for Testing	Fri 12/14/12	Fri 12/14/12	0 days	Joseph Baleta
20	Begin Testing of Code	Tue 1/8/13	Mon 2/4/13	20 days	Alec Reven/Jasson Gassel
24	Design Robot Shell	Mon 1/28/13	Mon 3/4/13	26 days	Chris Painter/Nicholas Marik
32	Recruit at Least 5 New Members	Mon 1/28/13	Mon 1/28/13	0 days	Chris Siebert
21	Refine Code	Tue 2/5/13	Mon 3/4/13	20 days	Alec Reven/Jasson Gassel
18	Order Materials for Robot Shell	Mon 2/18/13	Mon 2/25/13	6 days	Travis Bueter
22	Second Testing of Code	Tue 3/5/13	Mon 4/1/13	20 days	Alec Reven/Jasson Gassel
25	Construct Robot Shell	Tue 3/5/13	Mon 4/1/13	20 days	Chris Painter/Nicholas Marik
23	Second Refinement of Code	Tue 4/2/13	Mon 4/29/13	20 days	Alec Reven/Jasson Gassel
34	Have all Mechanical and Electrical Aspects Completed	Mon 4/29/13	Mon 4/29/13	0 days	Joseph Baleta
35	Have Robot Fully Functional	Mon 5/27/13	Mon 5/27/13	0 days	Joseph Baleta
8	Prepare Design Presentation	Thu 6/6/13	Tue 6/11/13	4 days	Joseph Baleta
9	Prepare and Go to Competition	Wed 6/12/13	Mon 6/17/13	4 days	Joseph Baleta
16	Wrap up and Send out Letters	Tue 6/18/13	Mon 6/24/13	5 days	Peter Moore
26	Reflect on Project Cycle	Tue 6/18/13	Fri 6/21/13	4 days	Joseph Baleta
36	Raise at Least $2300 Since January	Mon 6/24/13	Mon 6/24/13	0 days	Peter Moore