Redblade: Miami University's Multi-functional Autonomous Robot

Students: Robert Cole, Richard Marcus, James Morton, Chad Sobota

Advisors: Dr. Yu Morton, Dr. Peter Jamieson

Miami University

BIOGRAPHY

Robert Cole is a first year master's student in Miami University's Electrical and Computer Engineering Department. His research is in autonomous indoor navigation using an ultra-wideband radar and a robot's odometry. After graduation, Bob plans to pursue work in robotics and navigation.

Richard Marcus is a junior at Miami University studying Electrical and Computer Engineering. His research interests include GPS receivers and embedded systems. Richard plans to pursue a master's degree after completing his undergraduate work at Miami.

James Morton is a junior at Miami University studying Electrical Engineering and Computer Science at Miami University. His research interests consist of algorithmic development for computational biology applications. After completing a bachelor's degree at Miami, James plans to pursue a PhD in Computational Biology.

Chad Sobota is a fifth year senior at Miami University with a dual major in Computer and Electrical Engineering. He enjoys making his own laser projectors and is an active member in Miami's IEEE club. After graduation Chad will be a commissioned officer in the United States Air Force where he will work as a Developmental Engineer at Shriver AFB, Colorado.

Dr. Yu (Jade) Morton is a professor of Electrical Engineering at Miami University of Ohio. She received her PhD in Electrical Engineering from Penn State University and was a post-doctoral research fellow at the University of Michigan. Her current research interests are advanced GNSS receivers, ionosphere effects on GPS performances, and non-GPS RF navigation sensors.

Dr. Peter Jamieson is an assistant professor in the department of Electrical and Computer Engineering at Miami University. His PhD was granted from the University of Toronto. His research focuses on FPGA architecture and CAD.

ABSTRACT

Redblade is a multi-functional autonomous robot with two seasonal configurations which allow it to plow snow in the winter and mow grass in the summer. We are currently on the 5th generation of the Redblade platform which is an updated version of last year's platform with a new stereovision sensor and redesigned software architecture. This report presents the design and implementation of the Redblade mechanical platform, sensor components, software architecture, control algorithm, and safety systems.

INTRODUCTION

Autonomous robots capable of performing many functions with accuracy and reliability in a timely manner are highly desired in modern society. Redblade is designed as an expandable host to perform in multiple roles. It represents the next stage evolution of a multi-functional autonomous robot since it is able to compete in: autonomous navigation during the Intelligent Ground Vehicle Competition, autonomous snowplowing during the ION Autonomous Snowplow Competition, and autonomous lawn mowing during the ION Robotic Lawn Mower Competition. Redblade has been competing since the inception of both ION sponsored competitions starting with the ION Robotic Lawn Mower Competition in 2004[1] followed by the ION Autonomous Snowplow Competition in 2011[2], however this year marks the first entry of Redblade into the Intelligent Ground Vehicle Competition[3]. This paper describes Redblade's mechanical platform, sensor electronics, control algorithms, and safety mechanisms that make autonomous operation possible. Redblade's objective is to compete and win the 21st Annual Intelligent Ground Vehicle Competition.

More information, pictures, videos, and news articles can be found at www.muredblade.com[4].

TOP LEVEL REQUIREMENTS

An important step in engineering design is defining the top level requirements for the system being developed. This ensures that each necessary function that a system must perform is given the appropriate amount of consideration.

Table 1 presents a summary of the top level requirements for Redblade.

Requirement Specification	Component Used	Component Accuracy
Position Accuracy: < 22 cm	Novatel: **GPS-702-GGL**	Field test results: <16 cm
Heading Accuracy: < 2°/min	MicroStrain: **3DM-GX3-25**	Static: ± 0.5° Dynamic: ± 2.0°
Top speed: 4.03 m/s (9 mph)	RoboteQ: **AX2850** NPC Robotics: **NPC-B81HT**	N/A
E-Stop Time: < 2 sec	Remote Kill Switch: **In-house**	Field test results: <0.5 sec
E-Stop Stopping Distance: < 2 m	Mechanical Platform: **In-house**	Field test results: <1 m
Maximum Scalable Incline: 40°	N/A	N/A
Battery Life (idle): 10 hrs	Deka Solar: **8GU1-DEKA**	N/A
Battery Life (active): 4 hrs	Deka Solar: **8GU1-DEKA**	N/A

Table 1: Summary table detailing the top-level requirements for the accuracy of each component.

Additional requirements include the need for a higher amp-hour power supply, a dependable mechanical platform, enhanced control algorithm, and a CPU capable of handling an intense computational burden. These are discussed later in the report.

VEHICLE DESIGN

This section describes the overall mechanical design of Redblade. We will discuss plowing strategy, mechanical design, navigation system design, control system design, software design, and system integration.

A. MECHANICAL DESIGN

Redblade's mechanical platform consists of two tractor-style drive wheels for traction in the rear, two caster wheels for stability in the front, and an aluminum chassis with polycarbonate paneling that houses the electrical systems. An overview of the mechanical platform can be seen in Figures 1 and 2.

The primary material that was used for the chassis is 80/20 aluminum bar, chosen for its ease of construction and the large variety of materials that are able to be mounted to it [5]. Plate steel was used to mount electric motors due to the need for increased strength. Clear polycarbonate surrounds the chassis to protect the electrical components from the outside environment and allows us to easily diagnose problems. Thanks to the lightness of the aluminum frame and the Plexiglas siding, we are able to drastically reduce power consumption compared to our winter setup which uses 225 lbs of lead bricks which could simulate a heavier metal chassis.

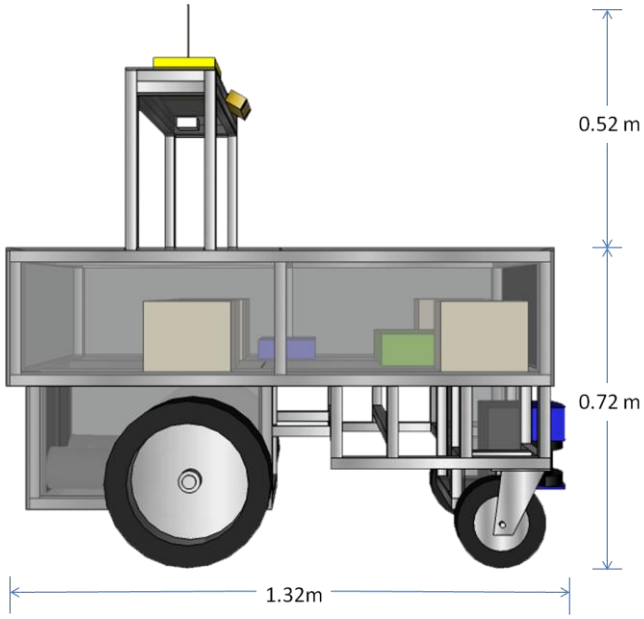

Figure 1: Side profile of Redblade with dimensions shown.

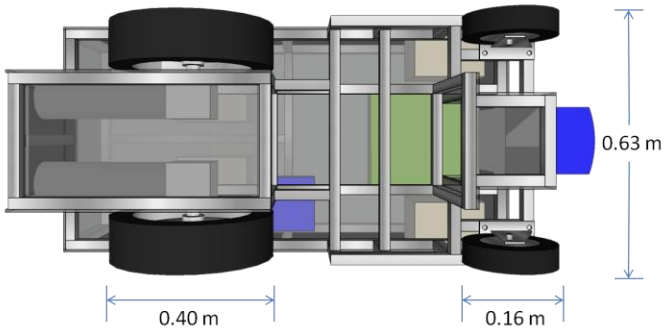

Figure 2: Bottom profile of Redblade with dimensions shown.

This year's major changes include: adding the Bumblebee 2 stereovision sensor from Point Grey[6], which is placed on our elevated sensor rack, angled downwards. In order to increase the viewing angle of our Bumblebee 2, we elevated our sensor rack an additional two feet. The Propak-v3 GPS receiver[7] is now placed directly above the wheel base to aid in the path planning algorithm. The robot has been upgraded to be driven by two NPC 24 volt electric high torque motors with 24:1 reduction gearboxes that together can pull 120 amps continuously[8].

The robot has a total of six 12 volt, 32 amp-hour gel-cell batteries[9]. Two sets are wired in series to make 24 volt sets used to power the drive motors. Since only one set is in use at a time, we can swap to the second set without having physically change out batteries and quickly swap to the fully charged set using PowerWerx[10] quick disconnects. The last two 12 volt batteries are wired in parallel for a total of 64 amp-hours and are used to power the computer, router, safety system, etc. Figure 3 shows the wiring diagram for Redblade's electrical system.

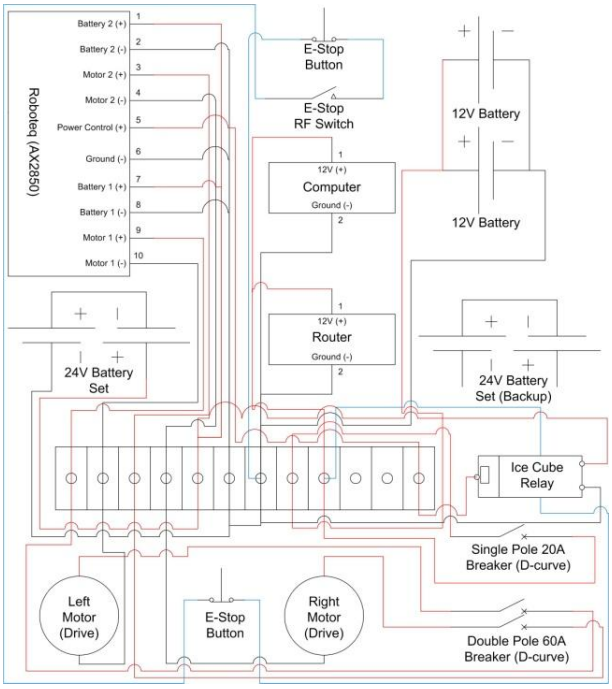

Figure 3: Wiring diagram for the Redblade power system.

B. NAVIGATION SYSTEM DESIGN

A MicroStrain 3DM-GX3-25 IMU[11] is used to determine the vehicle heading. It has an adjustable data rate to facilitate interfacing with different clients. Redblade does not use a magnetically corrected heading that is offered by this sensor. This IMU was shown to accumulate approximately 0.1^{o} of error for every minute of polling time.

The NovAtel's ProPak-V3 is a durable, triple frequency GNSS receiver that tracks GPS + GLONASS as well as L-Band and SBAS. This receiver was chosen mainly due to its OmniSTAR[12] capability, which would allow us to attain a more accurate GPS solution using OmniSTAR's virtual reference station correction information. For the GPS antenna, we choose the NovAtel GPS-702-GGL that supports L1 and L2 frequencies for GPS + GLONASS as well as support L-Band for OmniSTAR. This antenna is also durable and waterproof, has excellent multipath rejection, and has a highly stable phase center with minimum variation between the L1 and L2 centers. OmniSTAR has several subscription services from OmniSTAR VBS (sub-meter accuracy with less than 1 meter error), OmniSTAR XP (15cm accuracy), OmniSTAR G2 (10 cm accuracy with GPS + GLONASS which is great in multipath environments), and OmniSTAR HP (10 cm accuracy with GPS L1/L2 which great for high performance in open area). We chose to go with the OmniSTAR HP subscription since it would give us the most accurate gps position during the IGVC, which OmniSTAR generously gave us a free 90 day subscription (renewable) for our robotic research and design.

Two US Digital E7MS quadrature optical encoders[13] were installed on both left and right wheels of the vehicle. Each encoder sends its signal on two different channels with 90 degree offset. By using two channels it is possible to determine the direction of movement if there is no slippage. When the robot is moving forward, one channel emits a pulse before the other. The RoboteQ AX2580[14] motor controller uses these encoders in its internal feedback loop to ensure consistent speeds on both motors.

Each sensor may provide inaccurate data depending on the condition of the robot. This is discussed in more detail in the Failure Modes & Recovery Actions section.

For obstacle detection, we use a SICK Laser Measurement Sensor (LMS) 200 also known as a LIght Detection And Ranging (LIDAR) sensor. The LMS 200 is an extremely accurate 2D distance measurement sensor that can be interfaced over RS-232 or RS-422 and provides distance measurements over a 180 degree area up to 80 meters away (10 meters away for objects with only 10% reflectivity). This sensor works by beaming out a fan of eye-safe laser light off a rotating mirror and any object that breaks the fan will reflect the laser light back to the sensor, which can be calculated into a distance

measurement based on the time it takes to come back to the sensor. The LMS 200 has both a 'mm mode' where it gets back distance measurements in millimeters (with a detection range of up to 8.181 meters) and 'cm mode' where it gets back distance measurements in centimeters (with a detection range of up to 81.91 meters). It also has the ability of scanning angular range of 100° with angular resolutions of 1°, 0.5°, and 0.25° (shown in Figure 4 below) and angular range of 180° with angular resolutions of 1° and 0.5° (shown in Figure 5 below). The LMS 200 has a scanning frequency of 75 Hz and response time of 13-53 ms. The distance measurements from testing have a systematic error of +/- 15mm and statistical error (1 sigma) of 5 mm.

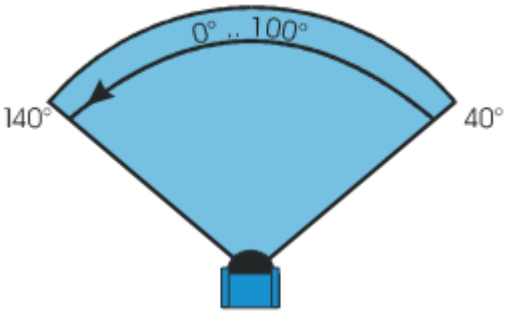

Figure 4: Measurement range 40° to 140° (view is from above, scan happens from right to left)

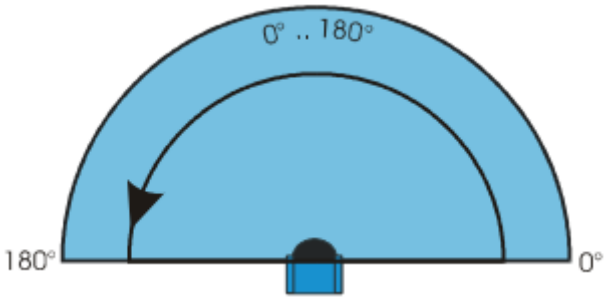

Figure 5: Measurement range 0° to 180° (view is from above, scan happens from right to left)

For our setup, we are in 'mm mode' using an angular range of 180° with an angular resolution of 0.5°, which gets us 180° vision of obstacles in front of our robot with a total of 361 different millimeter range measurements of obstacles less than 8.181 meters away from the sensor.

C. PROCESSOR & SOFTWARE DESIGN

All system processes are controlled by the onboard PC running a Linux installation. Communication with this device is accomplished via direct connection or through an on-board wireless router. A processor capable of handling a high computational load is needed. Figure 6 shows the resulting computer platform. Table 2 details the platforms specifications.

Because Redblade was required to function in a vast range of environments, weather-proofing was required to ensure safe and reliable operation. A standard hard drive contains components that are likely to freeze in low temperatures. Redblade uses a solid-state drive (SSD) to mitigate this risk as well as increasing shock resistance and speed. In addition to having better temperature endurance, the SSD is able to withstand much higher degrees of vibration and impact. The power consumption of the computer as a whole is reduced 85% from approximately 20 Watts to no more than 1.7 Watts.

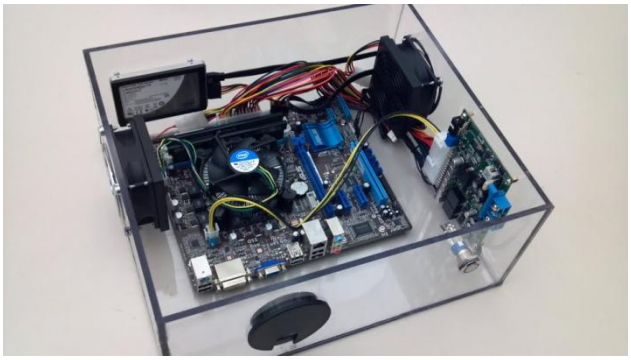

Figure 6: Redblade's computer platform in its housing. This housing can be easily removed from the vehicle if necessary.

Component	Manufacturer	Performance
CPU	Intel i7-2600K	3.4GHz quad-core
Memory	Corsair XMS	16 GB
Solid-State Drive	Intel 320 Series	80 GB

Table 2: Computer platform specifications.

Our software is written in C or C++ for speed. The measurements from all of the sensors will enter our ROS navigation node which will determine how to control the drive motors.

A more detailed representation of the component integration can be seen in Figure 7. The red boxes represent sensors while the blue boxes represent other physical components. The green boxes gather all of the information from these sensors and components and perform the communication between them and the computer.

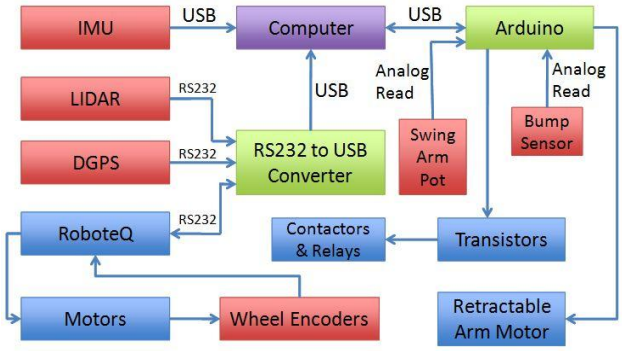

Figure 7: System integration block diagram. Note that some sensors are not used during IGVC operation.

D. SYSTEM INTEGRATION

Redblade features a three-layer system architecture that is abstracted in Figure 8. The top layer is the navigation and obstacle avoidance sensor suite. The current generation of the Redblade navigation sensor suite includes a Novatel GPS-702-GGL antenna with a ProPak-V3 receiver, a MicroStrain 3DM-GX3-25 inertial sensor, and two optical wheel encoders as part of the integrated motor drive system. The obstacle avoidance sensor suite includes our SICK Lidar and Bumbleebee 2 cameras.

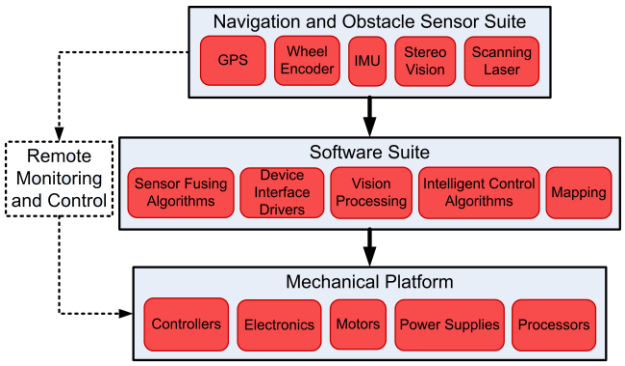

Figure 8: Three layer system architecture abstraction. Note that the remote monitoring and control is optional. The latter is disabled during autonomous operation.

The middle layer is the collection of software that provides driver functions for the sensors, sensor fusion algorithms, path planning, and vehicle motion control algorithm. The bottom layer is the mechanical platform, electronics hardware, including the motor controller, safety systems, power supplies, and processors that carry out the software functions.

Redblade utilizes the three navigation sensors (GPS, IMU, and optical wheel encoder) to determine its position, heading, and velocity (PHV). The vehicle's PHV information along with its predetermined destinations is processed by an on-board computer that implements a Proportional-Integral-Derivative (PID) control algorithm to adjust vehicle heading. Both remote and on-board emergency kill switches allow an operator to stop all robotic motion.

E. SOFTWARE STRATEGY

ROS (Robot Operating System) is an open source software suite that promises to democratize software development in the general field of robotics. ROS software consists of libraries and tools that provide hardware abstraction, device drivers, visualizers, message-passing, package management and more. Computation in ROS is segmented into a data abstraction

called a node. Each node in ROS is compiled in C++ or Python. Nodes can perform a wide variety of tasks such as sensor measurements, data processing or controller computation or data visualization. Each node has the ability to publish data to other nodes or subscribe to data streams provided by other nodes.

Each node in ROS runs on a separate CPU thread, meaning that all 8 cores on the CPU are saturated for high speed operations. Furthermore, our CPU is overclocked to 3.8GHz and 16 GB of RAM is installed to accommodate for the temporary storage for large data structures and caching. In this way, much of our data can be handled without reading and writing to disk. For further speedup, programs using OpenCV and Eigen are compiled with SSE intrinistics to enable vectorization.

NAVIGATION

A. MAPPING

Redblade uses a collection of ROS nodes that work together in order to implement SLAM (Simultaneous Localization and Mapping); *slam_gmapping*, *base_local_planner*, *base_global_planner*, and *move_base*.

When Redblade starts a run it begins creating a map with the origin of its local and global reference frames at its starting point. The local coordinate frame is centered between its wheels with the x-axis pointing towards the front and the y-axis pointing towards its left side. This local frame will move along with Redblade wherever it goes, however the origin of the global frame stays static.

As the robot moves through the course it samples point-clouds from the Bumblebee 2 and Lidar at a rate of 5 hz. In order to use these readings from a correct frame of reference Redblade uses a node based on a transform tree that shifts the coordinates of each point-cloud based on the sensors translation distance in x,y,z and rotation angle in roll, pitch, yaw from the origin point of the robot.

The Bumblebee 2 point-cloud is then pre-analyzed by a program using OpenCV to perform a Canny Edge Detection algorithm that will identify white lines, barrels, and colored flags, mark them as obstacles, and pass this new point-cloud to the *costmap_2d* node. The Lidar data is fed directly to the *costmap_2d node*.

Inside the *costmap_2d* node any data points within a designated radius of 20 feet from the center of the robot are treated as obstacles. This radius is imposed to prevent noise at the limits of our sensors' ranges from being mistaken as obstacles. These points are then used to populate a voxel 3-D grid which acts as our map, on which each point holds one of three values; unseen, obstacle present, or no obstacle present.

Of course, In order for the robot to plan a path the robot must have a goal to reach, which may be set through a configuration file that lists the different provided GPS waypoints. Once Redblade has come within a given radius of its current waypoint, it will then move on to the next point in the list until the list is exhausted. However, in order to reach each GPS waypoint Redblade must set a series of smaller points on our map that help it move around the known obstacles. In order to do this we use *base_local_planner* and *global_local_planner* nodes are able to plan the shortest trajectory path towards our goal--a GPS waypoint--that avoids the known obstacles.

Thanks to the costmaps, once the robot has observed an obstacle it will remember that for the rest of its run, which

means that Redblade will not take the same dead-end twice.

B. EXTENDED KALMAN FILTER

In our current setup, we have five sensors. Some of these sensors have very fast update rates such as the IMU and the wheel encoders; however these sensors are not very accurate by themselves. For instance, the IMU is known to have drifting problems. In order to make the best of all of the sensor measurements to obtain an accurate estimate of the current position and the positions of the surrounding obstacles, an extended Kalman Filter is used. The idea behind a Kalman Filter is any sensor, regardless of how bad it is, can be integrated into the robot to provide a better estimate of its location and surroundings. The Kalman filter contains two crucial ingredients: a dynamics model and a measurement model. The dynamics model can model the position and error covariance of the robot's physical position in the absence of sensor measurements. For our platform, this model gives information about x and position in addition to x and y velocity, heading, angular velocity, angular acceleration and IMU drift bias. The IMU is known to have an error bias that accumulates over time and in order to properly model this sensor, its error bias needs to be included in to the dynamics model. The overall dynamics model can be formulated as the following set of linear equations

$$x_k = F_k x_{k-1} + B_k u_k + w_k$$

x_k is the current state of the robot, F_k is a matrix of coefficients that relate the current state of the robot to the past state of the robot. u_k is the control input which includes motor speeds and, B_k is a matrix of coefficients that relate the current state of the robot to the control inputs. w_k is a random white Gaussian vector with a known covariance representing the noise in the robot's environment.

The measurement model gives information about reliable the sensor readings are. For our platform, we have five different measurement equations – one for each sensor. In the original Kalman filter equations, all of the equations in the dynamics and sensor models must be linear. However, since both the LIDAR and stereovision cameras are used as range sensors for SLAM, and range equations are nonlinear, a linear approximation must be made for these sensor measurements – a common technique used in the Extended Kalman Filter. The overall measurement model can be represented by the following set of linear equations.

$$z_k = H_k x_k + v_k$$

z_k is the current set of measurements and H_k is the a matrix of coefficients that relate the current robots state to the current set of measurements. w_k is a random white Gaussian vector with a known covariance representing the noise in the sensors measurements.

Using the dynamics and sensor models, the propagation equations can be readily applied as follows:

$$\hat{x}_k^- = F_k \hat{x}_{k-1}^- + B_k u_{k-1}^-$$
$$\hat{P}_k^- = F_k \hat{P}_{k-1}^- F_k^T + Q_k$$

Where $\hat{x}_k^-$ signifies the estimate of the current robot's state without measurements, $\hat{P}_k^-$ is the error covariance representing the uncertainty involved in estimating the robots current state and $Q_k$is the covariance of the w_k. These propagation equations are critical for calculating the estimated position and uncertainty in the absence of measurements. The presence of measurements, the update equations can be applied as follows:

$$K_k = \hat{P}_k^- H_k^T (H_k \hat{P}_k^- H_k^T + R_k)^{-1}$$
$$\hat{x}_k^+ = \hat{x}_k^- + K_k(z_k - H_k \hat{x}_k^-)$$
$$\hat{P}_k^+ = (I - K_k H_k)\hat{P}_k^-$$

Where $\hat{x}_k^+$ denotes the robot's state and $\hat{P}_k^+$ denotes the robots uncertainty after an update. It has been shown that this newly update readings always improve upon existing sensor readings.

C. STEREO IMAGE PROCESSING

To handle obstacle avoidance and line detection, Point Gray's the Bumblebee2 is used. These stereo cameras provide images at a frame rate of 20fps at with 0.8MP at 1032 by 776 pixel resolution. The Bumblebee 2 cameras can generate 3-D point clouds with color information and are accurate up to 5 meters. In order to integrate these cameras into ROS, these cameras need to be calibrated. Camera calibration consists of tuning two sets of parameters: intrinsic parameters and extrinsic parameters. Intrinsic parameters take into consideration the distortion of the cameras. Extrinsic consider the physical position of the cameras relative to each other necessary for conducting stereo rectification. Stereo rectification is used to generate point clouds that also give depth information about each pixel seen by the Bumblebee.

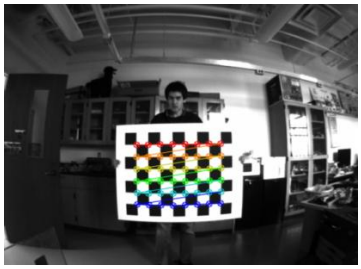

Figure 9: Camera calibration conducted using the ROS camera calibration program.

The ROS camera calibration program collects pictures simultaneously with both cameras and identifies the corners of the checkerboard in order to calculate these camera calibration parameters. Once the cameras are calibrated, images retrieved from the Bumblebee2 are sent to the ROS stereo image pipeline where undistorted images, disparity images and point clouds are generated.

Figure 10: The left and middle images are raw images from the Bumblebee2. The right image is the disparity image generated by the ROS stereo image pipeline

The point clouds from the ROS stereo image pipeline are then used for obstacle detection and line detection. The Canny Edge Detection algorithm is applied on the color information provided by the point clouds for position estimation of the obstacles and white boundary lines.

D. CONTROL SYSTEM DESIGN

Redblade uses a PID control algorithm for navigation between waypoints[15]. This algorithm adjusts wheel speeds based on present and past errors. We have two methods of defining the "error" of our robot. The first method drives heading error to zero and the second drives the distance from a line to zero. We are in the process of evaluating the performance of both approaches.

The first method starts by accepting a waypoint vector as its input. This waypoint (xd, yd) will be the destination waypoint for this method. (x0, y0) is the starting point. Both of these waypoints are defined in a local ENU reference frame with the origin being where our robot began. At any point during its travel between these two waypoints, its position (x, y) can be found with the GPS, and its heading (Θ_0) can be found with the IMU. Using this current position (x, y) and the destination (xd, yd), the desired heading (Θ_d) can be calculated using equation (1):

$$\theta_d = \arctan\left(\frac{xd - x}{yd - y}\right) \quad (1)$$

The difference between Θ_d and Θ_0 serves as the error input to the PID loop. When the KP, KI, and KD coefficients are selected correctly, they create a signal

which drives the motors and minimizes this error. Figure 11 shows a diagram of this error.

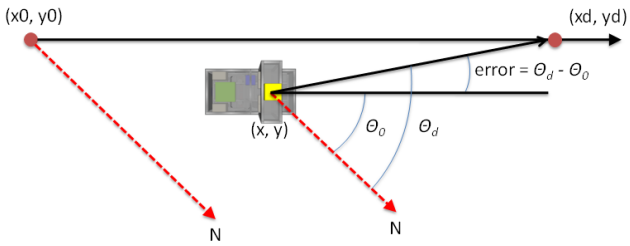

Figure 11: Diagram of how the PID error in heading is calculated.

Figure 12 below shows a block diagram of the PID feedback loop using this method.

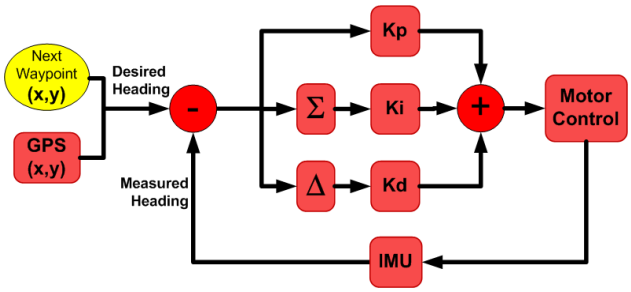

Figure 12: PID feedback loop using the first method that drives the heading error to zero.

The motor controller model is designed to take the update rates of all our sensors and controllers into account. Based on our wheelbase width, acceleration speeds, motor controller update intervals, and PID update interval we were able to construct a model of our system. It wasn't a perfect system, but the overall response of the system to input was captured and it was with this model that we tuned out PID controller. Figure 13 below shows a run where we gave the same inputs to our actual robot and our motor controller model. To test out our PID controller algorithm, we developed a simulator for error analysis. The next plot shows the calculated position from the simulator and the actual position.

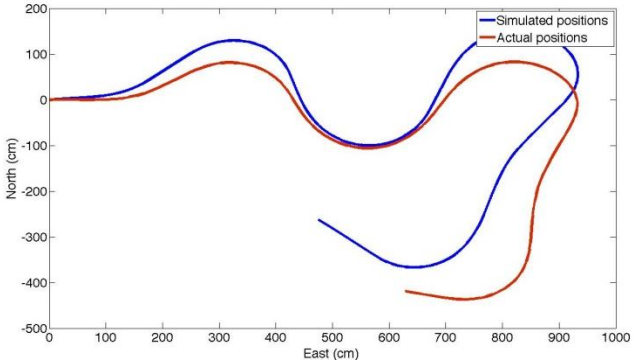

Figure 13: Plot of the calculated position from the simulator versus the actual position.

This simulator was then used to try different PID constants to evaluate their performance. In software, the robot was given the task of traveling ~40 meters along a straight line. It was given some initial error, and random noise was also added during the run to simulate inconsistencies that it will encounter in the field. Shown in Figure 14 is the algorithm that we used to find optimal PID constants. Two vectors are input into the algorithm, the PID vector, which in our case starts at [0, 0, 0], and the dp vector, which is initialized to [1, 1, 1]. By changing the PID constants one at a time by the value in the dp vector associated with that constant (respectively), we can evaluate the performance, and adjust this new set of constants. After this run, the method returns a number which represents the average error (the system can be tuned using either of the aforementioned methods of defining "error"). It is a hill climbing algorithm, so it may converge on a local maximum instead of a global maximum. To counter this problem, we ran the algorithm below multiple times with the PID and dp vector being initialized to random numbers.

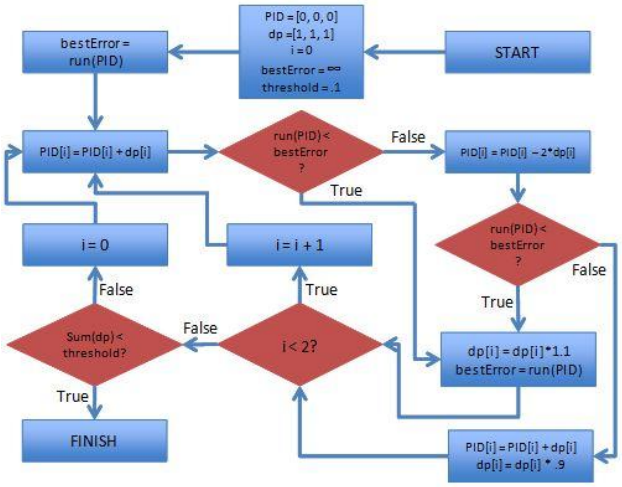

Figure 14: Algorithm used to find the optimal PID constants.

SAFETY SYSTEM

The safety system ensures that the robot ceases operation and comes to a complete stop within 2 meters. The emergency safety system on Redblade stops all motion in approximately 0.5 seconds and in less than 1 meters. This was in worst case testing from full speed to a complete stop on grass. It is accomplished by engaging one of three emergency kill switches. Two physical kill switches reside on either end of the vehicle (labeled as "A" in Figure 15), while the third switch is a remote handled by the user (labeled as "B" in Figure 15). The safety system circuitry is shown in Figure 16. Note that the switches are wired in series to allow a single switch to cause a complete stop of all motion.

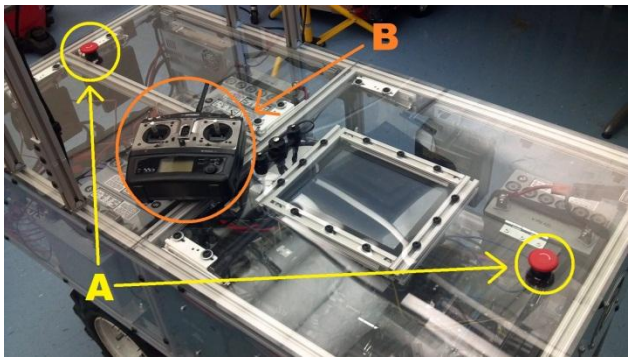

Figure 15: Redblade safety system includes two physical kill switches (A) and a remote kill switch (B). Note that the remote actuates a third physical switch on board the vehicle (not shown).

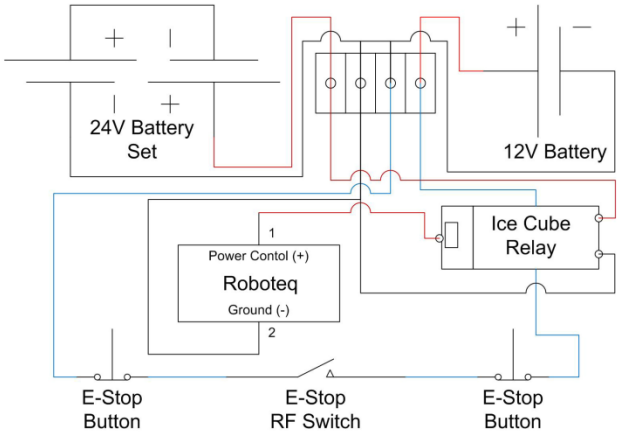

Figure 16: Circuit diagram of Redblade's safety system.

We also added a red led safety beacon to the top of the robot which goes solid red whenever the vehicle power is turned on. This led light goes from solid to flashing whenever the robot enters into autonomous mode and goes back to solid when exited.

FAILURE MODES & RECOVERY ACTIONS

This section will describe the failure modes and recovery actions that may arise during vehicle operation. Each mode and the corresponding recovery action is identified and explained below.

A. UNRELIABLE GPS SOLUTION

The possibility of high masking angle of buildings surrounding a competition site is potentially hazardous to the GPS system. Poor dilution of precision and the higher multipath of the environment can cause the receiver to lose carrier phase lock on one or more satellites. This can compromise the expected centimeter level accuracy of the system.

To solve this problem, the robot switches over to odometry. Our odometry solution uses the optical wheel encoders and the IMU to obtain a relative position. The number of clicks received from the wheel encoders can be

directly translated into distance. Our algorithm receives a reading from our left and right wheel encoders at 10 Hz and uses the following formulas to calculate position in our ENU reference frame. The heading used is the heading measured by our IMU.

distanceTraveled=(leftDistance+rightDistance)/2; (2)

*newEastPosition+=distanceTraveled*sin(heading);* (3)

*newNorthPosition+=distanceTraveled*cos(heading);* (4)

Since this is a relative positioning solution, errors compound over time. This is a backup solution, and through testing we found the error from our true position during a GPS outage isn't large enough to cause any significant problems with our robot's navigation. Figure 17 shows a graph of the position of the robot along an arbitrary run. Figure 18 shows the error (compared to the GPS) of the odometry-calculated position along that run. The largest error is no more than 0.5m and it occurred briefly when the robot is about 18m away from the initial position.

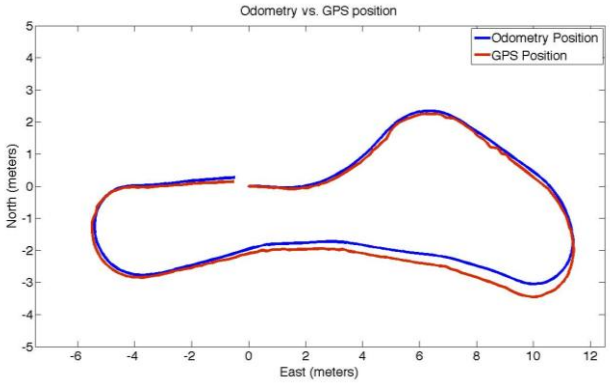

Figure 17: Plot of the GPS position versus the odometry position along an arbitrary run.

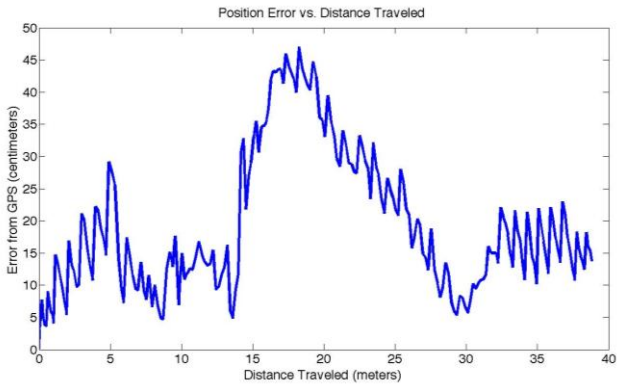

Figure 18: Plot of the error between the odometry position versus the known GPS position over the distance of an arbitrary run.

B. IMU DRIFT

A problem we had with the IMU was the gyroscope's drift over time. We've accounted for nearly all of the drift by using the zero-velocity update method. Before Redblade begins its run, it averages the bias of the IMU while it's in a static environment, and this average bias is then subtracted from each new reading. When the robot is navigating between two waypoints, we can dynamically calibrate the IMU drift by comparing the IMU readings to the vehicle heading computed from its path vector.

C. VEHICLE SLIPPAGE

Depending on the consistency of the terrain, it is possible to incur such load on the wheels to cause them to slip. This can result in heading changes and negatively impacts performance, especially if our GPS solution has failed and we are relying on odometry. We can detect slippage by comparing changes in distance reported by the GPS and odometry. To do this, we keep a list of previous positions and calculate the change in distance between the current position and a position measured one second before from both the odometry and GPS. If the difference between these two calculated distances is greater than an experimentally determined threshold, then we know that the wheels are slipping.

D. CURRENT OVERLOAD

When attempting to climb a very steep hill the amount of current requested by the drive motors maybe higher than the current rating of the wires which carry the power to the motors. This overload situation is handled first by a current limiting parameter in the configuration of the motor controller. This is set to 120 amps to prevent the motors' current carrying wires from overheating and causing potential damage to the wires or the motors.

Two 60-amp circuit breakers were also installed as a form of redundancy. These breakers are D-curve "slow blow" because electric motors can have an inrush current several times larger than their maximum sustainable current[16]. This slow blow capability allows the breakers to safeguard the drive system from any damaging overloads, but still allows for the high initial currents indicative of electric motors.

E. SPEED CONTROL

Redblade is capable of traveling up to a maximum speed of 9 mph, which is within the safety requirements of the competition. There are two methods used to ensure that the vehicle does not move with extreme or erratic speeds. The first is a software limit on the driver that communicates with the RoboteQ motor controller. This limit does not allow motor speed values to be sent to the controller if they will cause excessive speed. Secondly, the path planner that generates movement vectors is limited to requesting a maximum speed at any given interval. If the software receives a velocity that exceeds the speed limit, it decreases wheel speed proportionally to the amount of excess reported speed.

COMMERCIALIZATION & IMPLEMENTATION

Table 3 shows a detailed breakdown of the primary costs associated with the build of Redblade.

Component	Cost	Projected Market Cost
80/20 Frame	$1,500	$825
Wireless router	$50	$15
E7P Optical Encoders	$96	$28
Polycarbonate	$360	$180
Batteries	$660	$330
Sheet of Steel	$30	$17
ASUS Mobo/Intel i7/SSD/4GB RAM	$675	$200
RoboteQ DC motor controller	$495	$146
NPC Robotics 24V right angle motors	$900	$450
MicroStrain IMU 3DM-GX1	$1,500	$443
Novatel GPS-702-GGL	$1,200	$354
Misc (wire, bolts, fasteners, etc.)	$200	$110
ProPak-V3	$12,000	$7,085
OmniSTAR subscription	$0	variable
Bumblebee 2	$2,500	$738
Total Cost to Manufacturer		$10,921
Market Cost		$16,381
Profit to Manufacturer		$5,460

Table 3: The primary costs associated with the build of Redblade.

The table above shows the total cost to produce Redblade in a market environment assuming a manufacturer can obtain parts for roughly 50% of their retail prices. The costs also take into account a depreciation of 10% per year for electronic components and a 10% increase in cost per year for metals (over a period of 5 years).

A commercially available Redblade unit could be available with a permanently installed base station in lieu of an OmniSTAR subscription if it is planned to be used as a snowplow or lawnmower. This method would require a one-time survey of the property with the detachable GPS receiver in order to define the operating boundaries. If a movable tripod is used to hold the base station, a 30-

minute recalibration is required every time the tripod is moved.

CONCLUSIONS & RECOMMENDATIONS

This iteration of Redblade is the most robust platform to date. It is able to function autonomously as a snowplow, a lawnmower, and a navigating robot. This ability has been achieved through a navigation sensor suite, including a GPS receiver, IMU, wheel encoders, stereo vision, and laser scanner and an in-house mechanical platform. Several failure modes have been taken into consideration and recovery actions have been implemented to ensure robust performance.

The next goal for our team is to compete in the 2014 ION Autonomous Snowplow Competition in Saint Paul, Minnesota. This challenge will require an improved mechanical platform for winter weather. Obstacle detection and avoidance will not be necessary but the Bumblebee 2 and Lidar may be used to in place of the unreliable GPS solution in a downtown environment. In its current state described by this report, Redblade stands to serve as a reliable platform for autonomous multi-functional operations to come in the future.

The more long-term impact of this project is the valuable learning experience gained by the students working on the team. Students learned trouble shooting, managing deadlines under a tight schedule, and interfacing with parts and supply sources. They also learned specialized technical skills through this complicated project that required interfacing multiple components. Additionally, Redblade has been an excellent outreach and promotional platform not only for Miami University's Engineering programs, but also for STEM and navigation education in general.

ACKNOWLEDGMENTS

The Redblade team would like to thank the United States Army Tank Automotive Research, Development and Engineering Center (TARDEC), the Association for Unmanned Vehicle Systems International (AUVSI), Oakland University, and all other sponsors for holding the competition. The team received the funding support from Miami University Office for Advancement of Research and Scholarship, School of Engineering and Applied Science Dean's Office, the Department of Electrical and Computer Engineering, and the Department of Computer Science and Systems Analysis. Additionally, the team appreciates the technical guidance and support from Mr. Jeff Peterson from Miami University as well as the efforts of Mrs. Michele Lea for her administrative support. MicroStrain provided an educational discount for the 3DM-GX3-25 IMU used in the project.

REFERENCES

[1] McNally, B., M. Stutzman, C. Korando, J. Macasek, C. Mantz, S. Miller, Y. Morton, S. Campbell, J. Leonard, "The Miami Red Blade: An Autonomous Lawn Mower," Proc. 2004 ION Annual Meeting, pp. 538-542.

[2] R. Wolfarth, S. Taylor, A. Wibowo, B. Williams, Y. Morton, P. Jamieson, "Redblade: Miami University's Multi-Functional Autonomous Robot," 2011 ION International GNSS Conference.

[3] IGVC, "The 21th Annual Intelligent Ground Vehicle Competition," 2013, retrieved from http://www. igvc.org.

[4] "Redblade: Miami University's Multi-Functional Autonomous Robot," 2013, retrieved from http://www.muredblade.com.

[5] 80/20 Inc., "T-Slotted Framing," 80/20 Inc., 2005, retrieved from http://www.8020.net/T-Slot-1.asp.

[6] Point Grey, “Register Reference for Point Grey Digital Cameras”, 2013, retrieved from http://ww2.ptgrey.com/stereo-vision/bumblebee-2.

[7] ProPak-V3 Receiver, “ProPak-V3™ Triple-Frequency GNSS Receiver Overview”, NovAtel, 2013, retrieved from http://www.novatel.com/ products/gnss-receivers/enclosures/propak-v3/.

[8] NPC-B81, “NPC-B81HT High Torque Geared Motor,” The Robot Market Place, 2013, retrieved from http://www. robotmarketplace.com.

[9] Deka, "Solar Photovoltaic Batteries," Deka East Penn Manufacturing Co. Inc., retrieved from www.dekebatteries.com.

[10] PowerWerx, "Anderson Powerpole & SB Multipole Series Sets," PowerWerx, retrieved from http://www.powerwerx.com.

[11] MicroStrain, "Technical Product Overview: 3DM-GX3-25," MicroStrain, retrieved from http://microstrain.com.

[12] OmniSTAR, “Leader in Differential GNSS Solutions Worldwide,” 2013, retrieved from http://www. omnistar.com.

[13] US Digital, “E7P OEM Optical Kit Encoder”, US Digital, 2012, retrieved from http://www.usdigital.com.

[14] RoboteQ, Inc., “AX2550 AX2850 Dial Channel High Power Digital Motor Controller User's Manual,” 2007, RoboteQ, retrieved from www.roboteq.com.

[15] Wikipedia contributors, “PID Controller,” 2012, Wikipedia, The Free Encyclopedia, retrieved from http://en.wikipedia.org/wiki/PID_controller.

[16] Wikipedia contributors, “Inrush Current,” 2011, Wikipedia, The Free Encyclopedia, retrieved from http://en.wikipedia.org/wiki/Inrush_current.