Redblade: Miami University's Multi-functional Autonomous Robot

Students: Richard Marcus, James Morton, Robert Cole
Advisors: Dr. Yu Morton, Dr. Peter Jamieson
Miami University

BIOGRAPHY

Richard Marcus is a senior at Miami University studying Electrical and Computer Engineering. His research interests include signal processing and waveform design, specifically in GNSS areas. 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.

Robert Cole is a second 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 a PhD in robotics and navigation.

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

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.

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 6th generation of the Redblade platform which is an updated version of last year's platform with completely redesigned software, four-wheel drive, obstacle avoidance, and new navigation algorithms. 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 both autonomous snowplowing during the ION Autonomous Snowplow Competition[1], autonomous lawn mowing during the ION Robotic Lawn Mower Competition[2], and autonomous navigation in the Intelligent Ground Vehicle Competition[3]. Redblade has been competing since the inception of both ION competitions starting with the ION Robotic Lawn Mower Competition in 2004 followed by the ION Autonomous Snowplow Competition in 2011. This paper describes Redblade's mechanical platform, sensor electronics, software architecture, control algorithms, and safety mechanisms that make autonomous operation possible, specifically focusing on the snowplowing application. Redblade's objective is to compete and win the 4th Annual Autonomous Snowplow 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 < 15 cm	Topcon: **HiPer Lite Plus DGPS**	Static: 3mm horizontal RTK: 10mm
Heading Accuracy < 5°/min	MicroStrain: **3DM-GX3-25** US Digital: **E7P wheel encoders**	Static: ± 0.5° Dynamic: ± 2.0°
Obstacle pos. accuracy: < 0.5 m	SICK: **LMS-200 Lidar**	100° field of vision, 0.25° increments
Top speed: 1.5 m/s	RoboteQ: **AX2850** NPC Robotics: **NPC-B81HT**	Velocity Resolution: 14.25mm/s
Vehicle Dimensions: < 2 m	80/20 aluminum bar: **In-house**	N/A
E-Stop Distance: < 1m	Mechanical Platform: **In-house**	Test results: <0.5m

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.

SNOWPLOW VEHICLE DESIGN

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

A. PLOWING STRATEGY

We have implemented two different plowing strategies that take advantage of our new platform's strengths; one strategy for the single straight 'I'-shape and another for the triple straight 'I'-shape'. With four wheel drive our lateral traction has improved significantly and there is now a very small risk of the robot sliding from a lateral force on the front of the plow at the competition depth of 5-10cm of snow. Thanks to this improved traction we don't have to rely as much on our PID correcting for very large heading errors and we can angle the plow more steeply to push snow more efficiently. However, even with 7.72 horsepower from our new NPC motors[5], it is possible that plowing too much snow at once can cause the robot drift radically off-path, which is difficult to correct for. Breaking up our strategy into several passes will reduce the strain on the control algorithm.

Before the robot can move, we survey the corners of whichever field we are currently on and record these points in latitude & longitude. These points are then converted to an ENU local coordinate system using the bottom right corner point of Figure 1 as the (0,0) point (our convention, we could use any corner point). Based on the corner points, waypoints for the robot are calculated and generated within the boundaries of the surveyed field.

For the single straight 'I'-shape path a two-pass strategy is preferred because it halves the load that the motors have to push, reducing the possibility of wheels sliding. On each pass we align the inside edge of the robot over the center line to ensure that no snow is left behind. Depending on wet or icy conditions, the robot can be programmed to make more passes by joining the original path at the "Merge Point", seen in Figure 1, in order to ensure all snow is completely removed at the cost of additional time.

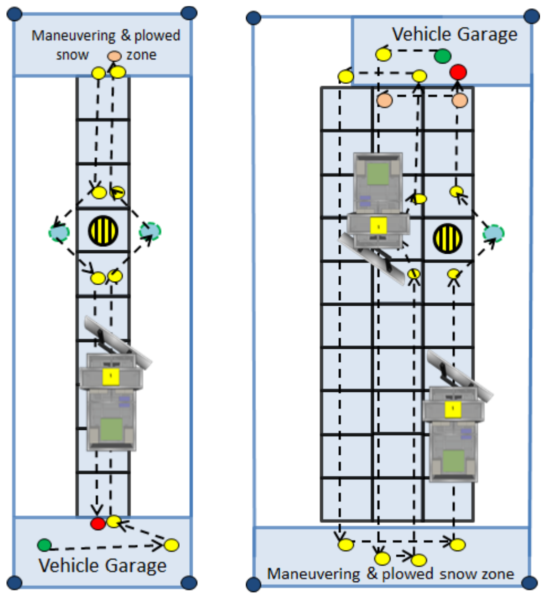
Maneuvering & plowed snow zone
Vehicle Garage
Vehicle Garage
Maneuvering & plowed snow zone

Figure 1: Single straight 'I'-shape two-pass plowing strategy (left), Triple straight 'I'-shape four-pass plowing strategy (right).

In order to avoid the obstacle in the snow field temporary waypoints are added to the front of the waypoint queue that plow around the pole. These waypoints are added whenever the obstacle is detected to be in the path between the previous waypoint and the next waypoint

Similar to our first strategy, for the triple straight ‘I’-shape path we have chosen a solution to reduce the load of snow that has to be pushed by implementing a four-pass plowing strategy, shown in Figure 1. Through our testing, we have decided to start plowing from the outside and working our way towards the middle so the robot pushes a manageable amount of snow each time.

B. SNOWPLOW VEHICLE DESIGN

Redblade’s mechanical platform consists of four snow-style drive wheels for traction in the front and rear, and an aluminum chassis with polycarbonate paneling that houses the electrical systems. An overview of the mechanical platform can be seen in Figures 2 and 3.

The primary material that was used for the chassis is 80/20 aluminum bar. It was chosen for its ease of construction and the large amount of available mounting materials[6]. Plate steel was used to mount the plow and 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.

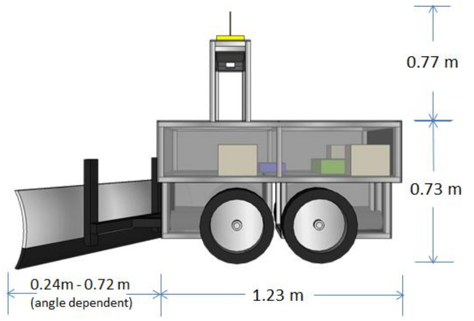

Figure 2: Side profile of Redblade with dimensions shown.

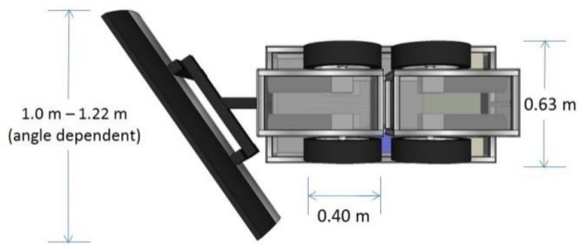

Figure 3: Bottom profile of Redblade with dimensions shown.

This year's major changes include an upgrade to the chassis to accommodate an additional set of motors. The robot is now driven by four NPC 24 volt electric high torque motors with 24:1 reduction gearboxes that can each pull up to 60 amps continuously. This upgrade to four-wheel-drive skid steering gives the robot much needed traction when pushing a heavy snow load in icy conditions.

The robot has a total of six 12 volt, 32 amp-hour gel-cell batteries [7]. Two sets are wired in series to make 24 volt sets used to power the drive motors. An extra set of fully charged batteries is always on hand and can be quickly swapped out using PowerWerx[8] 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 4 shows the wiring diagram for Redblade's electrical system.

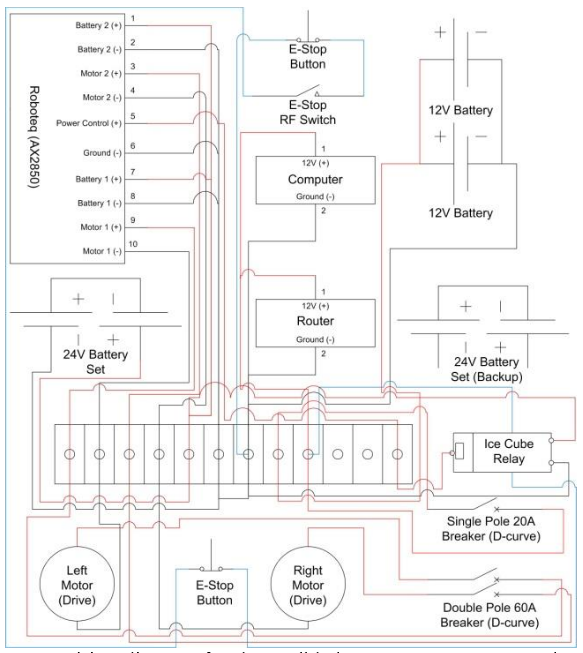

Figure 4: Wiring diagram for the Redblade power system. Note: does not reflect duplicate circuits for second set of motors.

C. NAVIGATION SYSTEM DESIGN

A MicroStrain 3DM-GX3-25 IMU[9] 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 HiPer Lite Plus system is a survey grade dual-frequency differential GPS receiver system by Topcon[10]. Field tests of the HiPer Lite Plus near Miami’s Engineering Building with masking angle at 30^o on one side of the sky shows location accuracy within 2cm as specified by the device manufacturer. The raw geodetic coordinates given by the HiPer Lite Plus receiver are converted to an ENU local coordinate system before being sent to the control algorithm. The origin of the local coordinate system is in the bottom right corner of Figure 1, while the robot’s initial heading points to the local y-axis.

Four US Digital E7MS quadrature optical encoders[11] were installed all four wheel on 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. However there is always slippage in a skid-steer design and our method for correcting for this is described later on. When the robot is moving forward, one channel emits a pulse before the other. The RoboteQ AX2580[12] 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 *Systems Integration* and *Failure Modes & Recovery Actions* sections.

D. GUIDANCE SYSTEM DESIGN

Our path plan is generated as soon as we survey the field and convert those survey points into ENU frame. A set of 7 waypoints is generated for the single-I path plan, and 14 for the triple-I. These points are created based on the measurements of the snow field given in the ASC 2014 rulebook and the dimensions of our robot so that it does not run out of bounds. Once all of these points are created, they are multiplied by a rotation matrix based on the difference in the angle of the field to ENU to put them in the ENU coordinate frame. The robot will repeatedly run through all of these waypoints for any number of iterations depending on how clean we want to make the snowfield and how much time we want to spend.

For obstacle detection, we use a SICK Laser Measurement Sensor (LMS) 200 also known as a LIght Detection And Ranging (LIDAR) sensor[13]. The LMS 200 is an extremely accurate 2D distance measurement sensor that can be interfaced over RS-232 or RS-422. 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 5 below) and angular range of 180° with angular resolutions of 1° and 0.5°. 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.

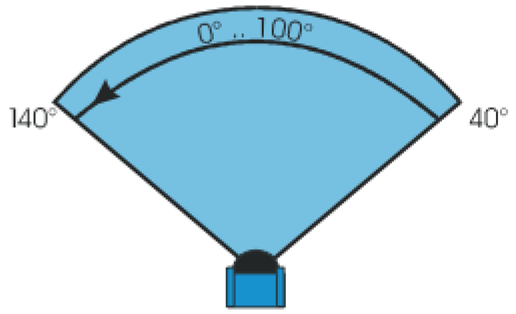

Figure 5: Measurement range 40° to 140° (view is from above, scan happens from right to left)

For our setup, we are in 'mm mode' using an angular range of 100° with an angular resolution of 0.25°, which gets us 100° vision of obstacles in front of our robot with a total of 401 different millimeter range measurements of obstacles less than 8.181 meters away from the sensor.

As the robot moves between waypoints it performs a calculation using Lidar range measurements to detect if its trajectory will intersect with radius R2 around the estimated position of the obstacle, seen in Figure 6. If the robot detects that its path will cross this area it creates four temporary waypoints that lie on the edges of the R1 and R2 circles. R1 is equal to half of the length of the robot plus a small buffer and R2 is equal to half of the width of the robot plus a small buffer. Usually point 2 is set to the right side of the pole because our plow angle performs better on that side, but it can either be set to either side if the other point would send the robot out of bounds.

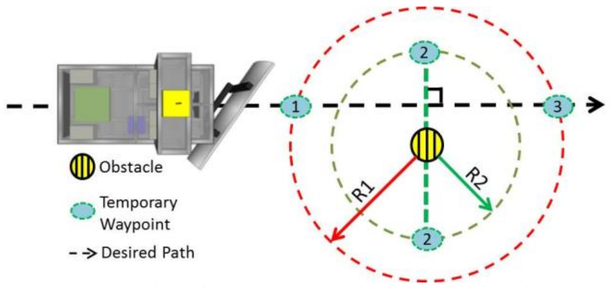

Figure 6: Generation of temporary waypoints to avoid obstacle

As long as the estimated position still lies within the path the first point will continually update with incoming position estimates. After reaching the first point the other two avoidance points are locked in and the robot travels to both of them before resuming a normal route. If the estimated position of the pole moves out of the path before it reaches the first temporary waypoint then it will resume its normal route.

E. CONTROL SYSTEM DESIGN

Redblade uses a PID control algorithm for navigation between waypoints [14]. 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 PID algorithm 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):

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

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 7 shows a diagram of this error.

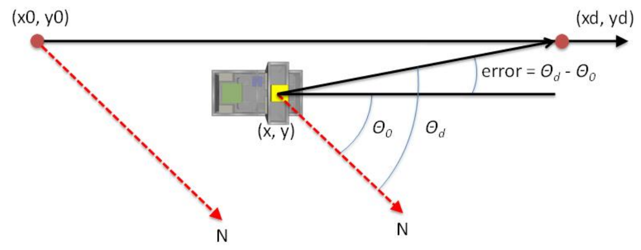

Figure 7: Diagram of how the PID error in heading is calculated.

The GPS error and IMU error are added together and are the input to the PID loop as shown in Figure 8.

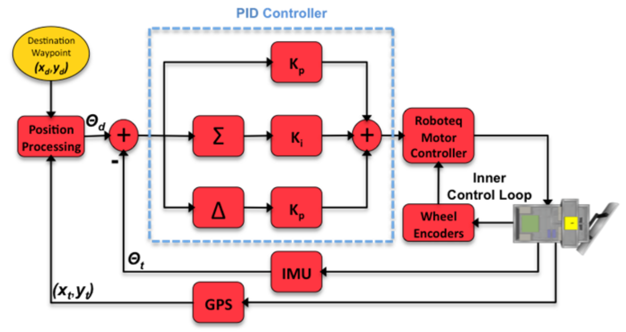

Figure 8: PID feedback loop using the first method that drives the heading error to zero.

In order to tune our PID we used the Ziegler Nichols method. To perform this heuristic first the I and D gains are set to zero, and then the proportional gain must be increased in small increments until the robot's path oscillates constantly before becoming unstable. Using that gain value (K_U) and the period of oscillation (T_U) we were then able to use the following equations to find our K_P, K_I, and K_D values:

$$K_P = 0.6 * K_U \qquad (2)$$
$$K_I = 2 * K_P / T_U \qquad (3)$$
$$K_D = 0.125 * K_P * T_U \qquad (4)$$

We considered tuning the PID manually by writing a simulator, but ultimately felt that it would be too time-consuming since the Ziegler Nichols method already worked very well even though it's not an optimal solution.

F. 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 9 shows the resulting computer platform. Table 2 details the platforms specifications.

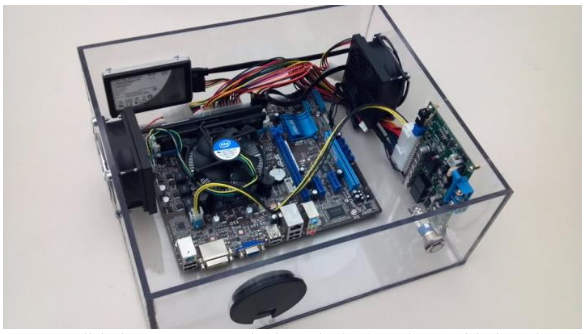

Figure 9: Redblade's computer platform in its housing. This housing can be easily removed from the vehicle if necessary.

Component	Manufacturer	Performance
CPU	Intel i7-2600K	3.4GHz quad-core
Memory	Corsair XMS	4GB
Solid-State Drive	Intel 320 Series	80GB

Table 2: Computer platform 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. In addition to having better temperature endurance, the SSD is able to withstand much higher degrees of vibration and impact. Power consumption is reduced 85% from approximately 20 Watts to no more than 1.7 Watts.

The software is mostly written in C or C++ for speed, although there are some small scripts written in Python that are not computationally expensive, as well as some testing scripts in Matlab.

One of the largest changes to Redblade this year was the introduction of the Robot Operating System (ROS), which is designed specifically for managing data passing between varying collections of sensors in a robotics system. All measurements are taken from sensors by their respective drivers which time-stamp and publish that data in different buffers called topics. Processes that need to read these measurements can subscribe to these topics and grab the most recent sensor measurement at any given instant in time. The abilities to communicate easily between any process and synchronize sensor measurements allow our solutions to be much more accurate. A visual representation of this procedure is shown in Figure 10.

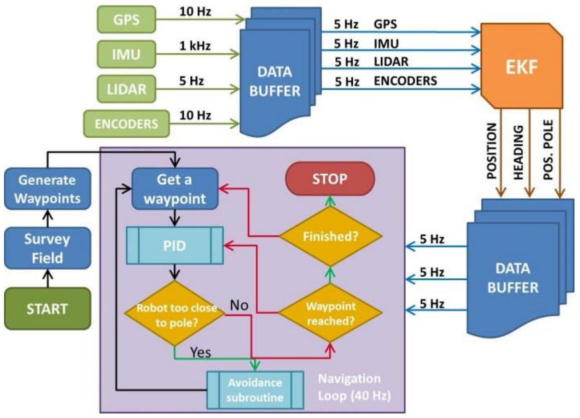

Figure 10: System flow chart demonstrating how messages are passed between software modules and hardware components.

ROS provides suites like rviz for real-time visualization of sensor data, which we use for viewing and quickly debugging Lidar data. Tools like ros-bags are available for easily recording data from any number of sensors that you can “replay” in order to test multiple algorithms on the same set of real-world data.

G. SYSTEM INTEGRATION

Redblade features a three-layer system architecture that is abstracted in Figure 11. The top layer is the navigation and obstacle avoidance sensor suite. The current generation of the Redblade navigation sensor suite includes a Topcon HiPer Lite Plus GPS receiver, a MicroStrain 3DM-GX3-25 inertial sensor, and four optical wheel encoders as part of the integrated motor drive system.

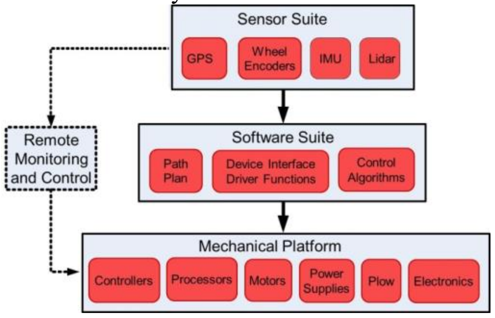

Figure 11: Three layer system architecture abstraction. Note that the remote monitoring and control is optional. The latter is disabled during autonomous operation.

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 vehicles PHV information along with its predetermined destinations are processed by an on-board computer that implements an Extended Kalman Filter to improve our PHV before giving the information to the Proportional-Integral-Derivative (PID) control algorithm to adjust vehicle heading.

Sensors like the IMU and wheel encoders have very fast update rates, but they are not very accurate by themselves. For instance, the heading reported by the IMU will drift over long periods of time, about 2^o per minute. 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 an EKF is to model the error of different information sources and combine readings from these sensors to obtain a better estimate of the orientation and position then given by any of the individual sensors.

The Kalman filter contains two main parts: 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

$$x_k = F_k x_{k-1} + B_k u_k + w_k \tag{5}$$

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 three different measurement equations –one for the IMU, wheel encoder and GPS measurements. Since the measurement equations for GPS and IMU 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.

$$z_k = H_k x_k + v_k \tag{6}$$

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:

$$\hat{x}_k^- = F_k \hat{x}_{k-1}^- + B_k u_{k-1}^- \tag{7}$$
$$\hat{P}_k^- = F_k \hat{P}_{k-1}^- F_k^T + Q_k \tag{8}$$

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:

$$K_k = \hat{P}_k^- H_k^T (H_k \hat{P}_k^- H_k^T + R_k)^{-1} \tag{9}$$
$$\hat{x}_k^+ = \hat{x}_k^- + K_k (z_k - H_k \hat{x}_k^-) \tag{10}$$
$$\hat{P}_k^+ = (I - K_k H_k) \hat{P}_k^- \tag{11}$$

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.

SAFETY SYSTEM

The safety system ensures that the robot ceases operation and comes to a complete stop within 3 meters. The emergency safety system on Redblade stops all motion in approximately 0.5 seconds and in less than 1.5 meters. This was in worst case testing from full speed to a complete stop on an icy surface. It is accomplished by engaging one of three emergency kill switches. Two physical kill switches reside on either end of the vehicle, while the third switch is a remote handled by the user. The safety system circuitry is shown in Figure 12. Note that the switches are wired in series to allow a single switch to cause a complete stop of all motion.

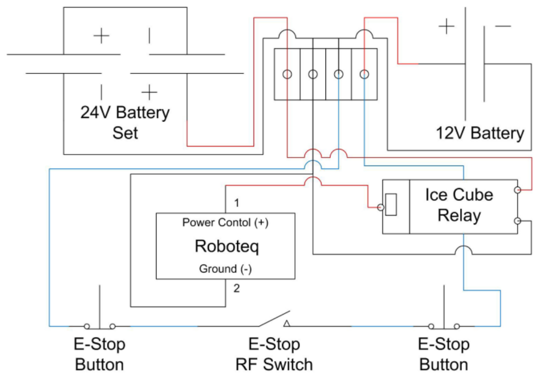

Figure 12: Circuit diagram of Redblade's safety system.

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. DENIED GPS SOLUTION

The high masking angle of buildings surrounding the competition site is potentially hazardous to the DGPS system. Poor DOPs 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 EKF will discard any GPS measurements that exceed the covariance we set and rely solely on measurements from the wheel encoders and IMU to get a relative position until the GPS is picked back up again. 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 5 Hz and uses the following formulas to calculate position in our ENU reference frame. The heading used is the heading measured by our IMU.

Dist_Traveled = (leftDistance+rightDistance)/2 (12)
*newEastPosition+= Dist_Traveled *sin(heading)* (13)
*newNorthPosition+= Dist_Traveled *cos(heading)* (14)

Since this is a relative positioning solution, errors compound over time. Through tests where we denied the GPS position to the EKF for periods of up to ten seconds we found the error from our estimate to the true position to be typically less than 0.5 m during a GPS outage. This error isn't large enough to cause any significant problems with our robot's navigation and it is unlikely we will be denied GPS for so long. Figure 13 shows a graph of the position of the robot along an arbitrary run. Figure 14 shows the error of the odometry-calculated position along that run. The largest error is no more than 0.7 meters.

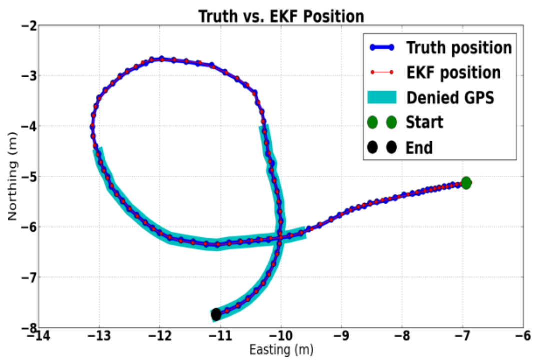

Figure 13: Plot of the GPS position versus the odometry position along a run where GPS solutions were denied for 10 second blocks.

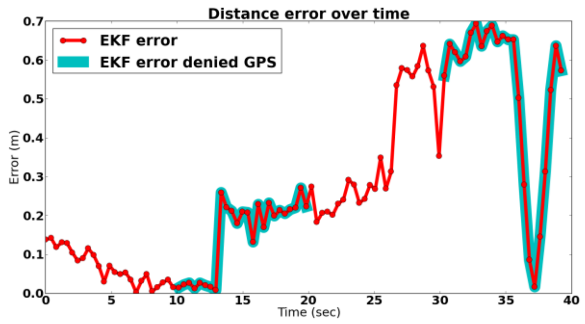

Figure 14: Plot of the error between the odometry position versus the known GPS position over the time of a run where GPS solutions were denied for 10 second blocks.

B. VEHICLE SLIPPAGE

Depending on the consistency of the snow being plowed, it is possible to incur such load on the plow as to cause the vehicles wheels to slip. This can result in heading changes and negatively impacts performance. 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. When this happens, we reverse the robot a certain distance and then continue forward at a faster speed.

C. CURRENT OVERLOAD

With a heavy snow load 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 50-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[15]. 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.

D. SPEED CONTROL

Redblade is capable of traveling much faster than the competition rules allow. There are two methods used to ensure that the vehicle does not exceed competition 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. Velocity measurements obtained from the GPS receiver are the second method of speed control. If the software receives a velocity that exceeds the speed limit, it decreases wheel speed proportionally to the amount of excess reported speed.

VEHICLE DESIGN CHALLENGES

A. ROBOT OPERATING SYSTEM (ROS)

In the spring of 2013 team Redblade began converting all existing software and hardware drivers to be compatible with ROS, which is a “meta-operating system” that abstracts control of hardware and communication between different processes. The majority of our software was transferred into the protocol that ROS uses, although some hardware drivers that we use were already created by the ROS community, but even those were modified.

The learning curve for ROS can be very steep initially but the community is very helpful and responsive to questions and issues. Although there was a large initial investment creating our drivers to work with ROS features, we are now able to develop much more quickly.

B. OBSTACLE AVOIDANCE & LIDAR

New to all teams this year is the obstacle in the middle of the snowfield. Because we have competed in other robotics competitions, like the ION Robotic Lawn Mower Competition and the IGVC, we are familiar with detecting and avoiding obstacles. Although we are comfortable using the Lidar and reading its data, we have never used it in a winter application before.

Our major challenges in using the Lidar came from errors due to snow flying around in the air. Because of this added noise we were having trouble clustering enough points together to detect the obstacle. In order to rectify this we use a moving average filter over several frames of Lidar data in order to filter out any snowflakes.

Another challenging issue we have had is that the position of the pole can drift up to 0.5 meters away when we are traveling quickly at extreme angles ($> 45^o$) relative to the pole. This is caused by the fact that the robot is moving during the Lidar scan which gives a bias to the ranges. Thankfully we never travel at such an extreme angle during the competition so we don't expect to see such extreme drifting. However, a small bit of error still remains due to movement and to compensate for that we average the estimated position of the pole over time.

C. FOUR-WHEEL DRIVE

Last year Redblade's biggest mechanical problem was traction. Our robot was designed with caster wheels in the back instead of another set of NPC B81HT motors. Because of the angle of the plow, the robot experiences a great amount of lateral force on its front, which caused it to rotate around its set of front wheels, introducing huge heading error that the PID had to correct for. With our new set of motors we do not only have the ability to push more snow, but we have significantly greater lateral traction in the back, keeping the robot on a straight path when snow pushes on the angled plow.

However, this solution did not come without a price. This extra set of motors required its own Roboteq and power supply, meaning we had to use our back up set of 24v batteries that we keep inside the robot as a normal supply to the second pair of motors. Additionally the robot draws much more current than normal because it is now skid-steer, meaning that the wheels must skid when turning since all of our axels are rigid. One problem that we rarely deal with is mechanical failure due to stress, but recently during normal testing on dry concrete a motor shaft sheared off cleanly. When testing on wet, icy, or snowy concrete the skid-steer behaves much more predictably with less stress on the motors and we do not expect to see this problem when competing.

Skid steer and the errors it introduces have also caused us to have to remodel several parts of our software including our odometry calculations, PID, and motor controllers.

D. EXTENDED KALMAN FILTER

One of the challenges of developing this EKF was obtaining the dynamics and measurement models. The calculation of the robot's position given the heading of the robot and the robot's velocity requires polar to Cartesian transformations, which are not linear equations. To circumvent this problem, we developed linear equations to approximate this transformation using linearization.

Another problem we faced was accounting for IMU bias. The IMU only gives an angular velocity which can give information about the heading in the robot's local coordinate fram. However, we need to know the robot's orientation in terms of North and East. To estimate this bias, we created another linear equation in the EKF. For each measurement update, the bias will be estimated based on the GPS position and the IMU angular velocity.

Theoretically, the orientation and the position of the robot can be even further improved if we account for the Lidar readings and the pole positions in the EKF. Another EKF was developed to account for these extra sensors to estimate the position of the poles in addition to the position and orientation of the robot. However, due to time constraints, we couldn't thoroughly test this EKF and didn't incorporate it into the final design.

COMMERCIALIZATION & IMPLEMENTATION

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

Component	Cost	Projected Mark Cost**
80/20 Frame	$1,500	$825
Wireless router	$50	$15
E7P Optical Encoders	$192	$56
Polycarbonate	$360	$180
Batteries	$660	$330
Sheet of Steel (x2)	$60	$34
Computer Hardware	$675	$200
RoboteQ motor controller (x2)	$990	$292
NPC Robotics 24V right angle motors (x4)	$1,800	$900
MicroStrain IMU 3DM-GX-25	$1,500	$443
Topcon HiPer Lite Plus System	$25,000	$7,381
Misc (wire, bolts, fasteners, etc.)	$200	$110
SICK LMS-200 LIDAR	$5,000	$1,476
Totals	**$37,987**	**$12,242**
Market Cost After Mark-up	**$18,363**	
Profit to Manufacturer	**$6,121**	

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 would be available with a permanently installed base station, only requiring 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 15-minute recalibration is required every time the tripod is moved.

The onboard software will automatically calculate an optimized path plan with adjustable settings like *path overlap*. Additionally, OmniSTAR[16] (a virtual base station) subscriptions are available to operate the robot without the base station, which have position accuracy within 10 centimeters compared to the 2 centimeter accuracy of the local base station. Based on our testing, any navigation solution accurate within 15 cm would be sufficient for our EKF to correct reliably.

CONCLUSIONS & RECOMMENDATIONS

This iteration of Redblade is the most robust platform to date. It is able to function autonomously as a snowplow and also a lawnmower. This ability has been achieved through a navigation sensor suite, including a DGPS receiver, IMU, and wheel encoders, an Extended Kalman filter, a PID-based control algorithm, 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 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 for Miami University's Engineering programs. The Redblade team members contributed to numerous outreach activities both on campus.

ACKNOWLEDGMENTS

The Redblade team would like to thank the Institute of Navigation Satellite Division for sponsoring the ASC and for the ION North Star for organizing the competition. The team received funding support from Miami University Office for Advancement of Research and Scholarship, School of Engineering and Applied Science Dean's Office, and the Department of Electrical and Computer Engineering. 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. We would also like to sincerely thank all of our new sponsors. The Topcon HiPer Lite Plus RTK system and Topcon B110 receivers were acquired through a sponsorship with Topcon Industries. The wheels and tires were generously provided by another sponsor, Kauffman Tire. A third sponsor, NPC Robotics Inc., gave us four B81HT motors and a cash donation of one thousand dollars. We are also grateful for the labor that Miami University's instrumentation lab has donated in order to make us expensive custom fittings.

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

[6] 80/20 Inc., "T-Slotted Framing," 80/20 Inc., 2005, retrieved from http://www.8020.net/T-Slot-1.asp.

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

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

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

[10] Topcon, "HiPer Lite +," Topcon, 2010, retrieved from www.topconpositioning.com.

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

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

[13] SICK Inc, "LMS200 Laser Measurement Systems," 2012, SICK Inc, retrieved from www.sick.com.

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

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

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