Design Report for

Zeus

Oakland University

14th International Ground Vehicle Competition

Selfridge Air National Guard Base

I, Ka C Cheok, hereby certify that the engineering design in this entry by the current student team is original and has been significant to be counted as credit in a senior design course.

Ka C Cheok
Faculty Advisor, Oakland University

I. Introduction

a. Design Methodology

The planning and design of *Zeus* began a year ago, immediately following the 13th annual IGVC. Our previous robot entry was designed to be a generic robotic platform – able to withstand enough modification and customization so that our team could experiment with different hardware and software concepts without worry of overloading the mechanical or electrical structure of the robot. Our team learned some extremely valuable lessons with this "testbed" robot during our trials at the competition, namely with how to best design propulsion systems. To this end, our team took our ideas and created a list of goals which were continuously referred to during the design and construction process:

- Simple to manufacture, maintain and operate
- Ability to operate in adverse environments such as snow and rain
- Powerful enough to maneuver thru bumpy terrain and climb steep inclines
- Implement simple solutions instead of complex ones when possible
- Replaceable batteries for continuous runtime
- Utilize simple entry and display system during runtime
- Fool-proof safety systems which ensure control during worst-case scenarios

b. Team Organization

After determining the design requirements for our robot, our team was broken into smaller groups based on knowledge and interest to further investigate, design, and implement individual components within each group. While the entire team was always made aware of the current research and status of the other groups, individual groups were given the authority to implement each component after discussing the feasibility of the underlying reasons and principals of components. The team was divided into the following groups:

Group	Team Member	Standing	Major
Software Systems	Brian Clark	Senior	Comp. Sci.
Systems Integration	Kevin Hayes	Senior	Business
Electrical & Structural	Jason Finch	Grad. Student	Elec. Eng.
	Chuwing Lo	Junior	Comp. Sci.

The task group *Software Systems* was primarily responsible for creating the core software controlling the robot, including path planning, sensor integration and obstacle avoidance. The *Electrical & Structural* task group was responsible for the design and construction of the frame, cover, and electrical systems per agreed physical requirements. Finally, the *Systems Integration* task group was responsible for acting as a liason between the other groups, making sure that the systems and components which were being designed and created would be interoperable. The entire team collaborated and created a timeline for critical events for each task group during the development and integration of the robot, with the culmination of the timeline ending in a complete system abiding to our design requirements ready for field testing. Work logs were kept individual notebooks where we noted time worked, as well as concepts and designs so that we could reference them in the future during discussions.

II. Hardware Overview

a. Frame & Construction

The basic frame of the robot is constructed out of 1-inch extruded aluminum obtained from 80/20. The aluminum is cut and fitted together using a standard joint technique. This technique consists of tapping the center of the aluminum of one piece and partially fastening a screw inside the tapped hole. This piece is then slid on to the second piece creating the joint, and affixed together using a metal plate, two screws and an eye bolt. Additional support is obtained via an angle bracket in the joint. Our team finds that using extruded aluminum is inexpensive compared to other options, and allows for the flexibility of modifications and ease of use. 1/4" Plexiglass is then used to hold the individual components in the robot, and

act as the "floors" and "ceilings" inside the robot structure. Plexiglass is measured and cut specifically for the section of the robot it will be placed in. Pieces are fit in the center groove of the extruded aluminum bars, encased on all sides. Sealant can be applied around the aluminum to ensure a watertight seal for that section. Plexiglass allows visibility thru the robot to all components inside, and can easily be drilled with holes for use in mounting our equipment. Aluminum and plexiglass are laid out in such a manner that separates the robot into two main layers. The first layer of the robot spans the entire width and length of the frame, providing the base on which everything else is based, housing all of the sensory and electrical equipment. The second layer of the frame covers the back half of the robot only and serves as a housing area for the Laptop PC which runs the software for the robot. This is then topped off with a sheet metal cover, upon which is mounted the GPS antenna, Emergency Stop button, and EasyI/O system. The covering for the robot consists of spray-painted sheet metal, affixed with weatherstripping to the outside areas of the robot. Here a combination of superglue and sealant is used to affix the sheet metal to the weatherstripping. On corners, 1" 90-degree angle plating trim was utilized to sport a sleek look. For the back portion of the second layer holding the laptop, a pull-out drawer is installed for easy access to the laptop during runtime and downtime. Given the ease-of use and cost-effectiveness of our solution, our team is extremely satisfied on the look and performance of the frame and cover.

b. Sensory Equipment

Our previously stated goals for the robot allowed for the efficient selection of sensory equipment which would be guiding the robot. The sensory equipment can be classified under two categories; equipment which is used to determine the robot's position (Location Sensory) and equipment which is used to detect objects in the outside world (Perception Sensory). Listed below are the specifics of the sensory equipment in both of these categories.

The term *Location Sensory* is used to describe the sensory equipment which is used to give the robot an ideal portrayal on where in the world it is currently located. This is essential for any sort of navigation where the robot needs to reach a specific destination point, be it a waypoint on a course or a building on campus. In order to give the robot the information it needs to deduce it's location, we first provide it with the raw information from a GPS receiver and a digital compass. The GPS unit we use is a NovAtel OEM4 with external antenna. This unit is programmed to send out the current latitude and longitude once a second to the Compaq PC via an RS232 serial line. The GPS system supports WAAS correction, with which an approximate raw accuracy of 1.5 meters is gained. This information is gathered along with our current direction using a Honeywell HMR3200 digital compass. This device gives our system 1-degree accuracy over 20 times a second to the Compaq PC via an RS232 serial line. We found that a compass is crucial to determining the robot's current direction, as relying on the derived direction from the GPS is not acceptable for competition performance. Finally, to complete our *Location Sensory* subsystem, the GPS data, Compass data, and odometer readings from our wheels are combined in a custom Kalman Filter which cancels out a large portion of the noise in our readings. By implementing this Kalman filter, accuracy an order of magnitude greater is obtained, compared to just the GPS system alone.

The NovAtel OEM4 GPS unit which derives latitude and longitude from the multitide of GPS satellites orbiting the Earth.

While *Location Sensory* refers to determining the position of the robot, *Perception Sensory* refers to identifying all other important and relevant objects in the vicinity of the robot. Objects such as barrels, fences, trees, lines, and people are only some of the differing things the robot must be aware of in order to perform it's navigation functions correctly. The robot needs to know what obstacles are around it on order to traverse the autonomous and navigation courses correctly and in the shortest time possible. To this end, we have employed

two different sensor systems to determine tangible objects; the SICK Laser Rangefinder and a Logitech Quickcam 4000.

This SICK Laser Rangefinder is mounted in the front of the robot and is able to easily determine if any physical objects exist in front of the robot. This unit scans a 180-degree arc in front of the robot, with individual readings every half-degree. Each reading gives us a distance to an object up to ten meters away, with each distance reading accurate to the centimeter. The SICK sends a complete set of readings every 20 milliseconds back to the Compaq PC via a RS232 line. Using this information, we plot a picture of the world directly in front of the robot according to the closest detected objects. This "picture" is then used to identify obstacles (such as barrels) in front of the robot.

The SICK Laser Rangefinder allows us to easily build a picture of the physical obstacles in front of the robot.

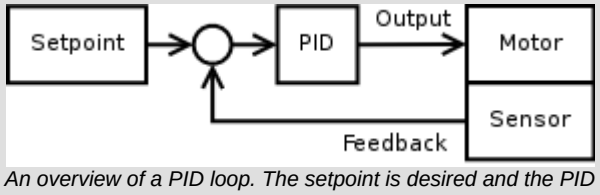
Another component we use in perception sensory is a Logitech Quickcam 4000. This webcam connects to the Compaq PC via a Universal Serial Bus(USB) connection and can provide a color 1-megapixel image 30 times a second. This webcam is placed it in the front of the robot on top of the SICK Laser Rangefinder. Using the pictures gathered from the webcam, the software on the Compaq PC can perform analysis and determine where white and yellow paint exist, regardless of ambient lighting conditions.

c. Drive Systems

After re-examining our previous IGVC entry, we decided that our current drive system was too powerful and complex to work efficiently. To correct this, *Zeus* drives itself via a simple, "master wheel" approach. The front two wheels are unpowered wheels locked in a forward position. Furthermore, the rear two wheels are joined in a "pod", are powered via a single motor, and rotate using the aid of another power motor. This drive motor is a heavy-duty 12-

volt motor from NPC Robotics, with a maximum speed of 285 RPM. The steering motor is a high-torque power window motor geared down to provide the maximum effectiveness for steering the wheel "pod". Encoders and potentiometers are attached to this wheel pod in order to give proper feedback on the actual motion of the wheel. During our field testing these motors have proved as extremely reliable and weather-resistant, and have been able to handle whatever speeds and forces were demanded of them. Each of these two motors is independently regulated by a Victor 884 Motor Controller. The Victor motor controllers control the speed by which the motors rotate, from a minimum of 3 percent to a maximum of 100 percent. The Victor motor controllers are controlled by a Pulse Width Modulation(PWM) signal which is generated by a custom processing program on a HCS12 Microcontroller board. This custom program, *CLEATUS*, was specifically designed to allow for the easy integration of PID control into any motor setup.

CLEATUS communicates to our Compaq PC using an RS232 serial line, over which we can send system variables such as our setpoints and PID constants. *CLEATUS* acquires this information and generates an adjusted PWM signal based on the setpoint and what encoders indicate the wheel is traveling at. For example, if a speed of 50% is sent to the motors, but encoder readings indicate that effective speed is only 30%, corrective action needs to be taken. *CLEATUS* will automatically adjust the PWM output to the speed regulators until it senses that the effective speed is at 50% where it belongs. This entire process occurs without any intervention on the part of our Compaq PC. The PC receives all of this information back over the RS232 serial line for use in the Kalman filter for the *Location Sensory* system. By offloading the PID calculations onto a separate controller



An overview of a PID loop. The setpoint is desired and the PID calculations take the information from the sensor to ensure accuracy.

board, the main PC can focus on intensive processing tasks such as obstacle detection instead of performing speed adjustments on the wheels.

III. Electrical Overview

a. Power Sources

Zeus is powered by one 12-volt deep-cycle marine battery from Optima. This battery is rated at 75aH and provides substantial power to the robot system. It was chosen and purchased after looking at a wide variety of battery options, and selected on the basis of the most Amp-Hours and Cold Cranking Amps(CCA) for the price. It is located in the center section of the bottom layer of the robot, and is easy to remove for transportation or storage. Additionally, it can provide a charge for approximately two hours under a full system load, and can be easily recharged by connecting battery chargers to the terminals on the electrical junction box. Charging is complete in a little over two hours, giving us a decent runtime ratio. In order to provide continuous runtime, the battery compartment was specifically designed to allow for the fast swapping of batteries, so that offline batteries could be constantly charging while fresh batteries can be placed in the robot for immediate operation.

b. Power Distribution

Electrical power is distributed thru the efforts of our main electrical junction box. Heavy-gauge wires connect from both terminals of the battery to inside the junction box. Additionally, terminals are placed on the junction box to facilitate simple charging to the battery without the disconnection of any wires. There is a master switch located next to the junction box which can break the entire electrical connection, shutting down all power for the robot.

Power is then supplied from the junction box to the several electrical circuits operating on the robot. A 24 volt circuit exists utilizing a 12V to 24V converter for the SICK laser

rangefinder. Four 12 volt circuits are also installed for the Compaq PC, *CLEATUS*, GPS/Compass, and a spare. All circuits installed in the junction box are activated using a fused toggle switch, and additionally have an LED for additional visual clarification on which circuits are active. To provide for 5 volts of power for items such as the E-STOP system, the HCS12 board running *CLEATUS* is used to step down the voltage from 12 volts. When possible, electrical wires are run along edges and thru cable organizers. In short, the power distribution is effective in it's efforts to supply power to the proper components of the robot.

IV. Software Systems

a. CPU Devices

As our robot must be fully autonomous, computer devices must be on the robot in order to perform the data acquisition, processing, and data output of the robot system without the aid of human intervention. With our entry, a Compaq R3000 laptop PC performs the data processing and calculations needed for the competitions. This device contains a Pentium 4 Processor and 512 MB of memory. Also, it sports a large fold-down display and is able to interface with any of our devices using standard Serial, Parallel, and USB port connections. An integrated wireless card is also present for networking capabilities, allowing for connections to the Internet and remote management of the laptop via other devices during field testing or operation. All software for the robot is written in C++ and is coded, compiled, and executed on this laptop. This device gives us superb battery life such that it does not need to be powered by the robot's power system, reducing main battery drain and increasing runtime.

The Compaq laptop which runs all system software and performs all data acquisition and analysis.

b. Software Layout

The design of any software system requires the precise definition of how individual components, starting with the basics of any software system: Inputs, Processing, and Outputs. While doing this, we identified components and tasks which would fall under each of these three categories. Sensory equipment (GPS, COMPASS, SICK & Webcam) provide the inputs to our computer system, and the main output of our computer system is a speed signal for each motor. Everything else in the system lies in the center under the umbrella term "Processing" which turns the raw data generated from the sensory equipment into meaningful movement of the robot itself. The following will document how we take our inputs, process them, and turn them into valid outputs for use in the competition.

The usage of C++ as our coding language allows us to use object-orientated programming to the fullest extent. By creating objects in code which correlate to physical objects on the robot, we can uniquely and conventionally assign properties and methods to all of our equipment. For instance, the COMPASS object handles all of the communication and processing regarding our Honeywell digital compass. The basis behind the object-orientated approach is that data is strictly assigned to objects and the only methods of communicating between objects are rigidly defined, preventing data loss and corruption. By knowing the communication channels which operate between objects, we are free to modify or replace the code for a specific object, so long as the inputs are processed properly and the outputs remain the same.

To implement this scheme, our team brainstormed and created a list of objects which could then create objects with. Both physical and logical devices were thought of during this process, so items such as the GPS and serial communications were considered when creating our object-orientated approach. Once this list was complete, the defined communication

channels were standardized between all of the different software objects in order to create a consistent, documented flow of data.

The display of our software system was built to be customizable so that we could display exactly what we wanted to, exactly when we wanted to. During debugging, the information needed in order to make programming changes constantly, so designing a display system to accommodate this was a priority. It allows operators and programmers to see both raw and processed data for any of the software modules in the system, as well as turning those modules on and off. The operational state of the robot can be controlled using the EasyIO system, which consists of a 4x20 LCD display and a 4x4 keypad. This enables fast data reads and one-touch debugging and operation both in the workshop and in the field.

Additionally, software support for "preferences" was enabled for quicker debugging purposes. With preferences, critical software variables inside the software system can be changed, such as thresholds, speeds and configuration values. This is done while the robot is running, so as soon as the new data is entered, the entire robotic software system is updated with the new data. This eliminates the need to stop the robot, go thru the code, change a value, recompile, and start the program back up again. Changes can be made to the operating environment in seconds rather than minutes, giving an order of magnitude more debuggin time out in the field.

As mentioned, all processing on our robot is accomplished by a Compaq Laptop PC which is responsible for coordinating all relevant hardware using the software system described above. The PC communicates with the various onboard equipment via both USB and Serial/RS232 protocols. Sensory devices are preprogrammed to automatically start sending data to the PC when power is applied to them, and *CLEATUS* automatically starts expecting data, so it is up to the PC to coordinate the incoming data, process the data into meaningful

results, and output it to the motor controlling system in a timely fashion. The independent nature of the objects allowed us to independently thread data acquisition and sending routines so we could take advantage of multithreading performance in our laptop PC. With this, we are able to independently refresh sensor information, perform path processing, and send data to the motor controller system separately at appropriate refresh rates. By performing these independent threading techniques, we can reliably achieve an efficient processing loop time. The main process loop is guaranteed to cycle at least once every 50 milliseconds, meaning that the robot is calculating what it should do over 20 times a second. We have confirmed that processing power is plentiful enough to allow for such calculations, giving us a reliable indication that the robot is constantly made aware on it's surroundings and what it should do about them.

V. Systems Integration

a. Path Planning and Obstacle Avoidance

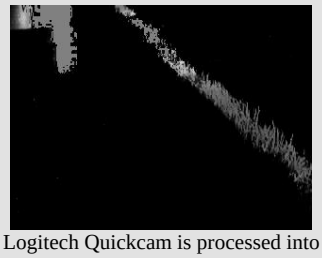
Unlike previous entries where the robot relied on passive path planning and not taking future actions into account, the current robot boasts a new, custom path planning algorithm, code named *Bolt*. *Bolt* is a destinationless implementation of the common A*STAR path planning routine. When navigating thru the obstacle course, the robot has no way of knowing where *exactly* it's destination is supposed to be – the goal is to keep moving thru the course. The one known fact about the obstacle course is that the robot must remain within the white lines. To this end, *Bolt* navigates by tracking obstacles and always aiming to identify white lines wherever possible and remaining between them at all times, present and future. This is accomplished by continuously referencing an internal memory grid which stores the physical locations of detected obstacles, such as lines and barrels. By continuously tracking the obstacles into the future, the robot navigates itself thru the course, much like a dragonfly navigates by using the sun for positioning.

While our true destination is not known in the Autonomous challenge, it is known when we perform the navigation challenge. While *Bolt* has been optimized to run by tracking objects, this feature has come in very useful when we add on a destination such as a waypoint to the system. By continuously identifying where the robot is in the internal memory grid and comparing this with information about where the next waypoint is, *Zeus* ends up "pushing" it's way thru the course, much as a swimmer pushes themselves thru water. A small subsystem identifies when a waypoint has been reached, at which point the next waypoint is added as the destination, and the robot "swims" it's way past obstacles to reach it's objective.

In order for *Bolt* to track the obstacles successfully, a way is needed to populate the internal memory grid with relevant information. Both the SICK Laser Rangefinder and the Logitech Quickcam aid in this task. The SICK can easily identify the precise location of physical obstacles in the 180-degree arc in front of the robot. Given that the width of barrels and buckets is known, a simple algorithm was written to correlate multiple "obstacles" into discrete barrels, which are continuously added and monitored to the internal memory grid.

b. Vision Processing

While avoiding the physical obstacles is key in this competition, so is remaining within the white lines. The Logitech Quickcam camera can acquire and process 20 pictures per second. During the processing phase of each cycle, the overall hue, saturation, and value(HSV) of the entire picture is calculated. By doing this, the camera can work well in almost any lighting situation – cloudy, sunny or rainy. A custom algorithm is then run on groups of pixels within each image based on the mean and standard deviation of HSV values to determine



A look at how an original picture from the Logitech Quickcam is processed into identifying probabilities for white line detection

the probabilities that they are paint sprayed on the ground. These probabilities are then transferred over to the internal memory grid, using the known angle and view of the camera to correlate a pixel with an X/Y coordinate on a grid in front of the robot. By performing these tasks, our internal memory grid referenced by our *Bolt* algorithm continuously remains updated with the most recent information available about the course for path planning and tracking purposes.

c. Safety & Emergency Control

The Emergency-Stop (ESTOP) system is critical in order to maintain proper control over the robot. With our current ESTOP system, there is a custom relay board sitting between *CLEATUS* and the Victor motor controllers. The Victor motor controllers will cease to operate if they do not detect a PWM signal within a very narrow range. Knowing this, a system was designed such that a GO signal had to be present at all times from both the wired and wireless ESTOP systems. Two relays control this signal, and if the GO signal from either the manual pushbutton or the wireless controller goes away, the relays snap open, halting the PWM signal from going to the motor controllers and stopping the robot. This default-off scenario ensures that the robot will only operate under safe conditions. If the ESTOP system trips and the robot stops, simply reapplying the GO signal will not make the robot move. A separate RESET switch has been placed on the robot such that additional measures need to be taken so that the robot does not inadvertently slip in and out of a working condition.

VI. Expected Performance

a. Speed

While the theoretical speed of our robot given the RPMs of our motors and our wheel size is around 12 miles per hour, under controlled testing we can max out our speeds at around 8

miles per hour. Knowing this, we set the maximum duty cycle within *CLEATUS* to be 50% of the expected full-speed signal that the Victor motor controller accepts. This hardware limitation prevents the settings of speeds which may be harmful. Furthermore, odometer readings of the wheels tell *CLEATUS* precisely the speed the wheel is spinning. If it is detected that the wheels are spinning over 4.5 MPH, the system will immediately reduce the output to the wheels in order to maintain a safe environment.

b. Climbing Ability

The maneuvering and climbing abilities of *Zeus* have been tested throughout the rolling terrain of Oakland University. The robot can navigate 25-degree inclines both up and down with absolutely no loss in control.

c. Reaction Times and Battery Life

Because of the independent threaded nature of our programming, the main event loop can cycle no matter what the refresh rates of our equipment are. The main process loop cycles once every 100ms, providing quick reactions to any obstacles or objects which are in the vicinity of the robot. While CPU utilization can reach 80% at times, our cycle time still remains valid. With regards to battery life, we can run the robot for 2 hours in the field before recharging needs to take place. This is with a full system load and processing taking place. Due to the modular nature of the battery compartment, additional power can be immediately applied in the form of a fresh batter for a continuous runtime scenario.

d. Obstacle Detection

Obstacle detection is based around how far the *Perception Sensory* systems can operate. The SICK can identify objects at a distance of 30 feet, and the Logitech Quickcam was angled and positioned to give the same amount of distance. With this in mind, our system can identify and classify barrels, trees, people, and paint marked on the ground successfully at a

range up to 10 meters. This can then easily fill up the internal memory grid of where obstacles lie so that the *Bolt* system can track and navigate thru the course successfully.

e. Navigational Accuracy

Given that the location of the robot is essential for the path planning to take place, the *Location Sensory* system has been designed to maximize accuracy. The GPS can give the robot a raw WAAS-corrected signal which in practice gives us an accuracy on where the robot is in the world by roughly 180 cm. By combining this information, the direction the robot is heading via the digital compass, and the odometer readings from the wheel sensors into a Kalman filter, an accuracy of approximately 20 centimeters has been tested – an order or magnitude more accurate than simply using a GPS system alone. Using this, *Bolt* can successfully plot obstacles and navigate itself to predetermined points with ease.

APPENDIX A: Cost Estimate

The following is a breakdown of the costs involved in building this robot, including materials and equipment.

Extruded Aluminum	$200
Plexiglass	$50
Brackets & Screws	$20
Castors & Tires	$80
Electrical Wires & Enclosures	$70
Battery	$185
NPC Robotics Motors	$155
Victor Motor Controllers	$149 each
Honeywell HMR3200 Compass	$175
NovAtel OEM4 GPS	$3,000
SICK LMS200-6	$4,000
Logitech Quickcam 4000	$89
MiniDragon+ HCS12	$85
Compaq Presario R3000 Laptop	$900
TOTAL	**$9,307**