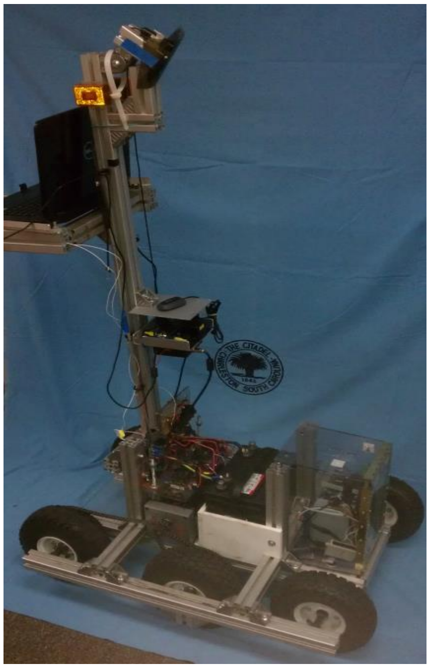
Intelligent Ground Vehicle Competition 2013

Design Report for

ZERGLING



The Citadel – The Military College of South Carolina

Charleston, SC 29409, USA
May 10, 2013

Authors: Athanasios Athanason, Nathan Cintula, Jonathan Hager, Michael Lacey

I certify that the design and engineering of the vehicle by the current student team has been significant and equivalent to what might be awarded credit in a senior design course. Changes include new data acquisition devices, introduction of a field-programmable gate array, and completely new control software in addition to numerous hardware upgrades.

Dr. Ronald Hayne, Advisor

1 Introduction

The “Zigzag Environment Routing, Ground Level Intelligent Navigational Guidance” robot (Zergling) is a 6 wheeled vehicle designed and fabricated by Citadel Electrical Engineering seniors to compete in the IGVC. Zergling is the reincarnation of CLETIS and Hawk-Eye, the previous Citadel entries into the competition. While most of the mechanical systems are the same, the software and sensors are almost entirely new and many hours were spent programming in LabView to write Virtual Instruments (VIs). Sensor integration is achieved via an AD* mapping algorithm VI, and the robot successfully navigates autonomously to GPS waypoints while avoiding lines and obstacles.

1.1 Team Organization

The Citadel Competitive Robotics team consists of four members majoring in Electrical Engineering and minoring in Computer Science. Because the team consisted of active duty Navy Officer Candidates and South Carolina Militia Cadets, balancing the demands of the design, academics, and two distinctly different sets of military obligations was a particularly unique challenge. Team organization responded dynamically according to the situational challenges that arose. Schedule competition along with the software intensive nature of the project led to our adoption of “Extreme Programming” from the Software Engineering model. We settled into two two-man teams for work: Tan Team and Gray Team. Gray Team focused heavily on the mapping, lidar, and camera components of the robot. Tan Team focused on GPS, compass, logistics, documentation, and sponsorship. Our effort spent in process management, brainstorming, and mechanical refinements was equally divided. Twice a week the full team would start with an internal status report lasting between 20 and 60 minutes to communicate updates, share challenges, and encourage detection of potential problems. This worked well to keep everyone’s efforts coordinated and to keep team synergy alive. Overall the group spent over 600 man-hours to create Zergling.

1.2 Design Process

Keeping with the extreme programming model, features were added incrementally as needed. Initial brainstorming and research took about two weeks to develop a general plan, schedule, and a notional design. The notional list of design elements included: Camera, GPS, and LIDAR, sensor-integrated to an AD* mapping algorithm using a LABVIEW programming environment. These systems were selected for various reasons. National Instruments LABVIEW was selected over the Robot Operating System (ROS) primarily due to a lack of expertise with the required LINUX operating system but also because of expected difficulties with atypical USB peripherals in LINUX. LIDAR was selected over ultrasonic sensors due to lessons learned from the experience of a previous team. A mapping algorithm was selected over the subsumption used the year before for sensor integration because we wanted to give the robot the ability to map its environment and be able to remember the location of past obstacles and plan a path, rather than being purely reactive.

Once these general ideas were developed, changes arose in response to different problems. For example, the original mapping algorithm data structure was developed using MATLAB but we found that refactoring our program to run natively in LABVIEW worked better once the LIDAR code was integrated. In order to correlate objects detected from sensors into the map data structure in a meaningful way, localization and orientation data was needed. The orientation need was met with wheel encoders and a magnetic flux gate compass. These new components then required the use of an FPGA to properly implement (previously DAQs were being used, but the sample rate did not prove to be high enough).

We investigated the use of an android device for GPS and navigation instrumentation, but eventually had to abandon that design branch due to a lack of time and expertise.

The two-man teams were very effective at producing robust code and the status meetings facilitated integration by providing a forum for negotiating code interfaces.

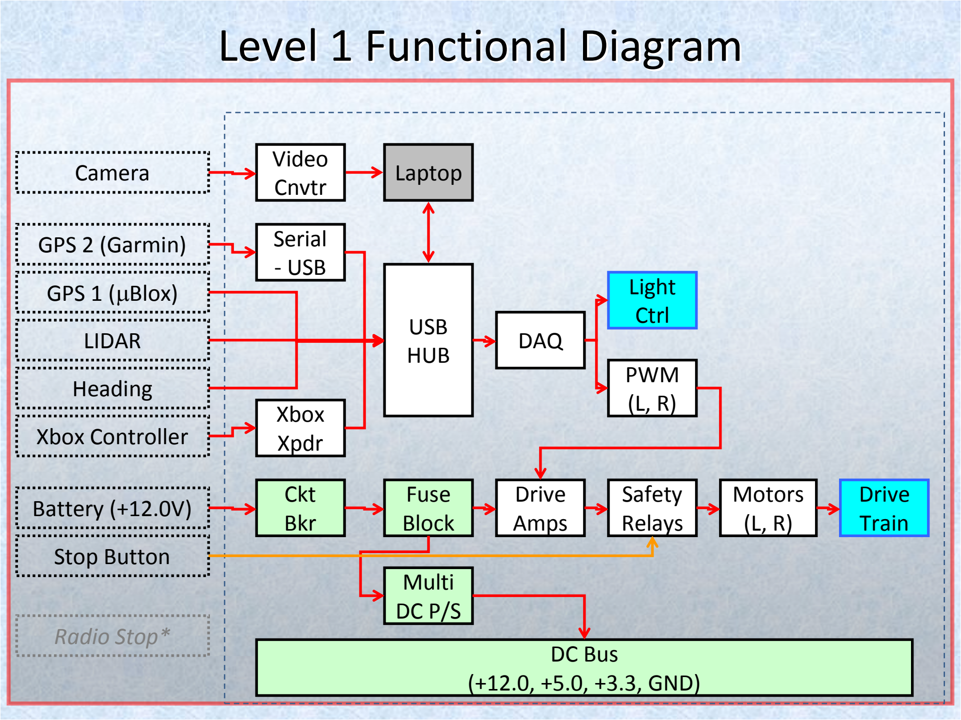


Figure 1. Initial functional diagram for robot system design

2 Innovations

Initial brainstorming began with a blank slate approach. As the notional design developed, more of the legacy components were deemed includable. Legacy systems (particularly mechanical systems) were reused where possible. A number of revisions and improvements were planned. The design elements that are reused relatively unchanged for this year's entry include: the frame, drive train, and power system. Inherited sensors were the Hokuyo URG-04 LIDAR and the PC88W R-2 camera. We reused some of the calibration data from the line detection camera code since the camera was kept in the same place. Code for the other sensors was developed by team members. We made use of the LABVIEW example library VIs where possible, but all were eventually modified to suit our purposes and integrate together. New sensors added were the uBlox 6-series GPS, Logitech C310 webcam, Devantech CMPS03 magnetic compass, and TRDA-20 wheel encoders. These sensors were included to increase the capabilities of the robot. The compass and encoders run through an FPGA, another innovation this year, and provide high quality dead reckoning data to the mapping algorithm. The webcam is an upward facing camera added along with a color recognition VI to detect flags at the end of the Auto-Nav challenge course. The largest innovation in terms of effort and code size was the mapping VI, which integrates all of the sensors and correlates real world objects to map data structure by assigning "costs" to the map nodes based on the perceived location of obstacles detected. Because the map program develops a model of the physical world as the robot travels, the path planning AD* algorithm can update as new information about the environment becomes known. As an additional feature of this approach, the map model may be stored for future use with little additional code; this may prove useful to the vehicle's operator.

3 Mechanical Design

3.1 Chassis

ZERGLING's aluminum frame was inherited from the previous two project teams. Supports are lightweight and sturdy, and the six wheel configuration provides a stable and capable platform suitable for outdoor terrain. Weather resistant components are implemented where possible and transparent acrylic paneling is used for the main housing to protect interior components.

3.2 Drive Train

The six-wheeled design was kept due to its proven ability to negotiate inclines up to a 15% gradient. Additionally the 9 tooth sprockets were used for their increased torque shown by testing from the Hawk-Eye team. 2 motors per side are used to drive a chain which in turn drives the back two wheels of each side. Back wheel drive is chosen due to the location of main weight of the robot and the payload near the aft portion of the vehicle. The front wheels are free-spinning.

4 Electrical Design

4.1 Power Distribution

A 12V Marine battery is used to supply power to the system. 12V is directly supplied to the H-bridges powering the drive train and a DC-DC step-down converter supplies a 5V bus for electronic components

4.2 Safety

Two emergency stops are implemented to kill power. One is a large red button located on the frame, and the other a wireless stop with a 200 foot operational range implemented with a DXR702 two-channel receiver. Additionally a manual control VI was written with a wireless X-box controller providing input. The manual VI is useful in transport, as well as in isolating specific systems during testing and calibration

.

Figure 2. Wireless emergency stop

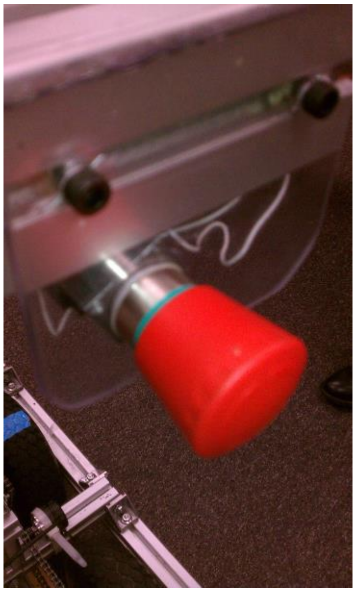


Figure 3. Manual emergency stop

4.3 Drive Subsystem

The switch to using an FPGA allowed us to eliminate two DAQs from an earlier stage in the design. However one DAQ still remains for its output capability and is used to control the drive train. A digital control signal is sent from the decision-making software and output from the DAQ to the left and right

PWM boards. This signal is then converted to analog and sent to the IFI Victor H-bridges to control the drive train. He H-bridges must occasionally be calibrated using an original VI.

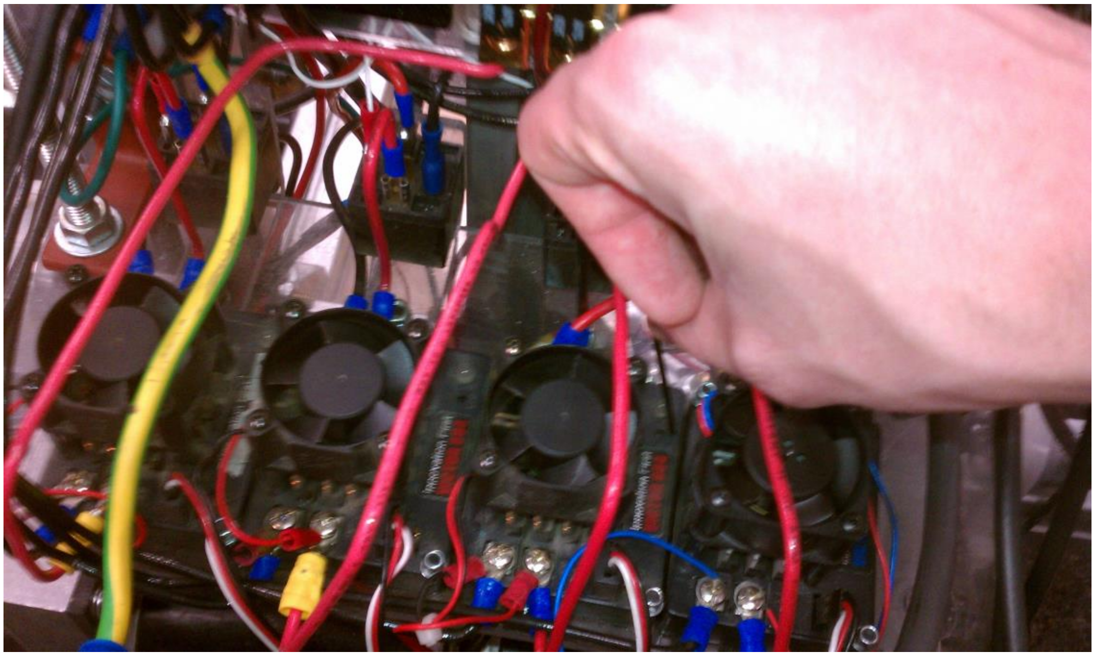


Figure 4. H-bridge calibration

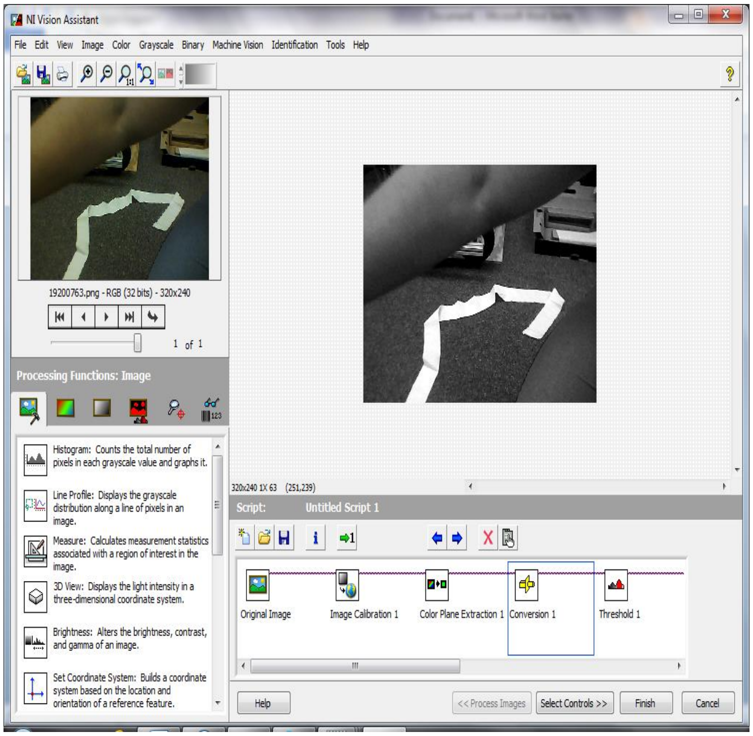
4.4 Sensors

A number of sensors are used for the robot to encounter its world. The LIDAR and line detecting camera were inherited hardware components from a previous project.

4.4.1 Camera

4.4.1.1 Line detection

The first, downward-facing, camera is used to detect lines. It continually takes images, then filters them to grayscale. Point to point mapping with a reference image of the field is used to convert the image into real world distances. We then threshold the grayscale images so as to keep only a very small band of the lighter colors. Noise reduction removes any small areas that were left over from sources of error such as a glint of light on a smooth surface. Three vertical and three horizontal divisions are used to divide the screen into seven sectors, and a best Hough Edge Rake is performed in each quadrant to sweep for edges. The start and end points of each edge are then found and 30 points are approximated in between to fit a line which is then added onto the appropriate map squares.



Histogram: Counts the total number of pixels in each grayscale value and graphs it.

Line Profile: Displays the grayscale distribution along a line of pixels in an image.

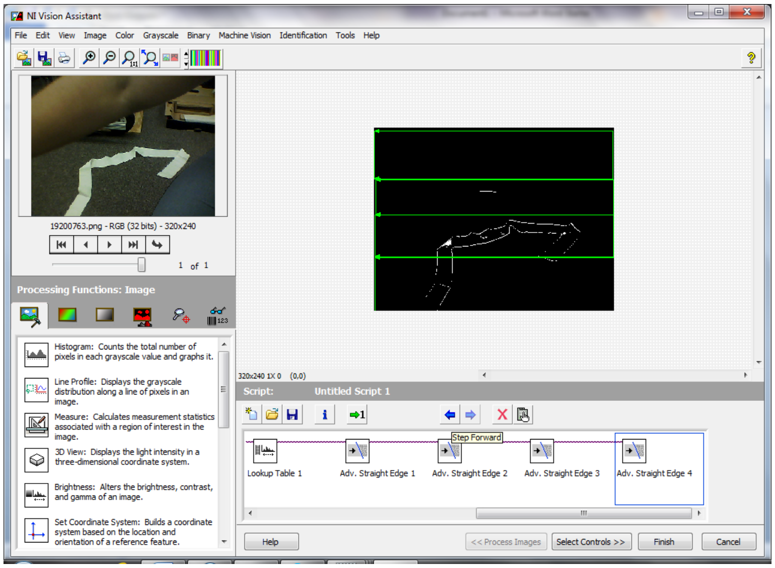
Measure: Calculates measurement statistics associated with a region of interest in the image.

3D View: Displays the light intensity in a three-dimensional coordinate system.

Brightness: Alters the brightness, contrast, and gamma of an image.

Set Coordinate System: Builds a coordinate system based on the location and orientation of a reference feature.

Figure 5. Initial image acquisition and grayscale filter



Histogram: Counts the total number of pixels in each grayscale value and graphs it.

Line Profile: Displays the grayscale distribution along a line of pixels in an image.

Measure: Calculates measurement statistics associated with a region of interest in the image.

3D View: Displays the light intensity in a three-dimensional coordinate system.

Brightness: Alters the brightness, contrast, and gamma of an image.

Set Coordinate System: Builds a coordinate system based on the location and orientation of a reference feature.

Figure 6. Filtering and noise reduction.

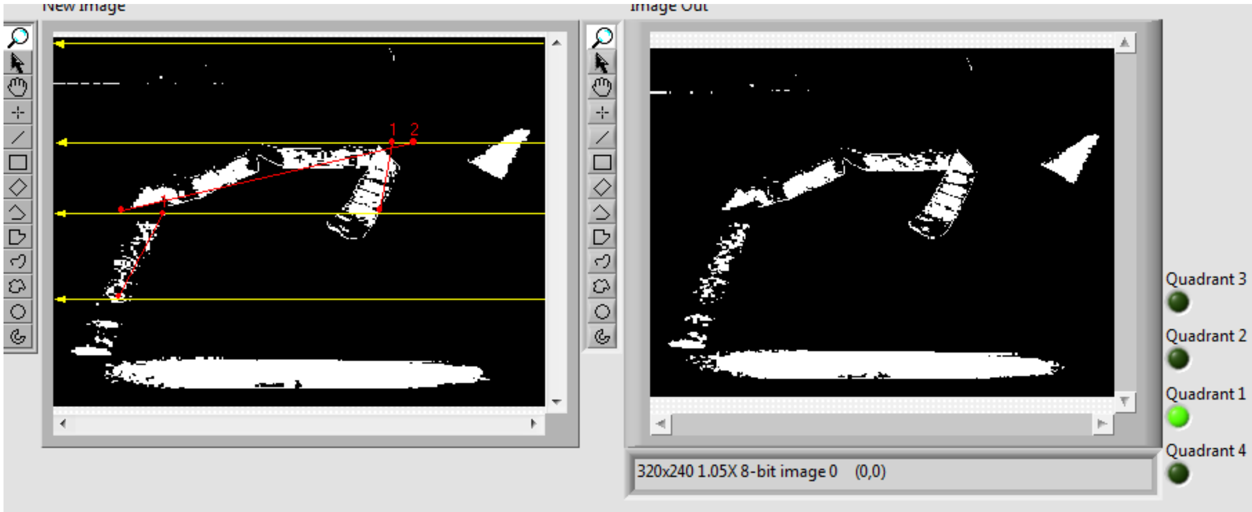


Figure 7. Hough Edge Rake and resulting line approximations

4.4.1.2 Flag Detection

A second, upward-facing, camera is used to detect flags for the end of the Auto-Nav challenge. Most of the color is removed from the image except the desired red or blue. Random spaces are removed via averaging and then a search for a cluster of color chains is performed. Since the camera is pointed straight out instead of at an angle like the line detection camera, no point to point mapping is necessary. The screen is divided simply into quadrants corresponding to far left, left, right, and far right of the robot's centerline. Location of the flags relative to the position of the robot is then used to determine which channel to navigate to at the end of the course.

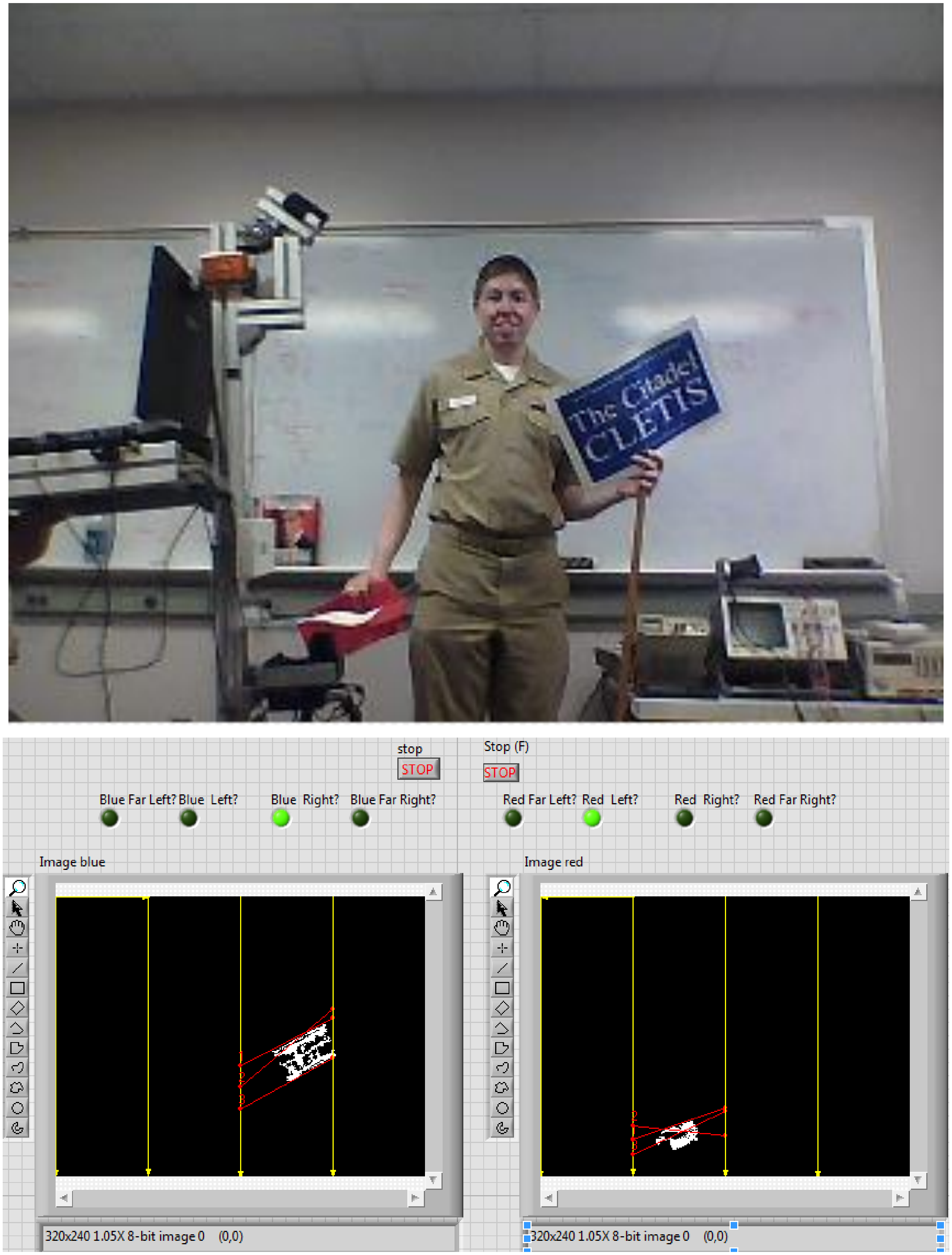
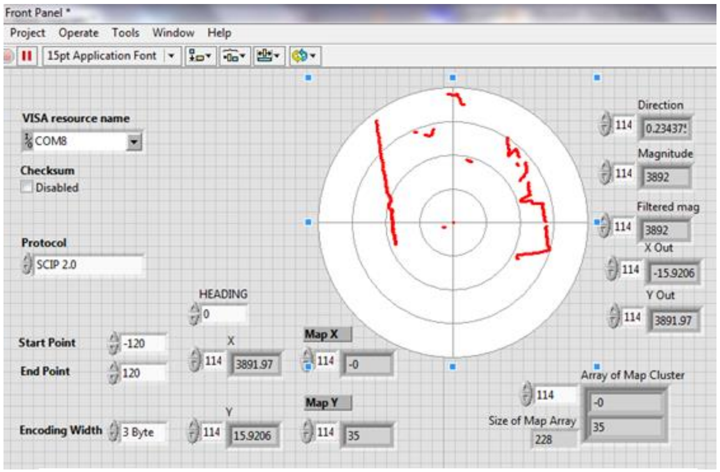


Figure 8. Initial image and resulting color detection of blue (bottom left) and red (bottom right)

4.4.2 Lidar

The Hokuyo URG LIDAR populates an array of values in LABVIEW which corresponds to a 2D representation of the distance and direction of obstacles. Using heading input from the navigation package as a reference, the array of polar values is converted to x and y coordinates used to populate the path-finding map.



VISA resource name COM8
Checksum Disabled
Protocol SCIP 2.0
HEADING 0
Start Point -120
End Point 120
Encoding Width 3 Byte
X 114 3891.97
Y 114 15.9206
Map X 114 -0
Map Y 114 35
Direction 114 0.23437
Magnitude 114 3892
Filtered mag 114 3892
X Out 114 -15.9206
Y Out 114 3891.97
Size of Map Array 228
Array of Map Cluster 114 -0 35

Figure 9. LIDAR VI front panel view. Polar values can be seen converted into Cartesian coordinates.

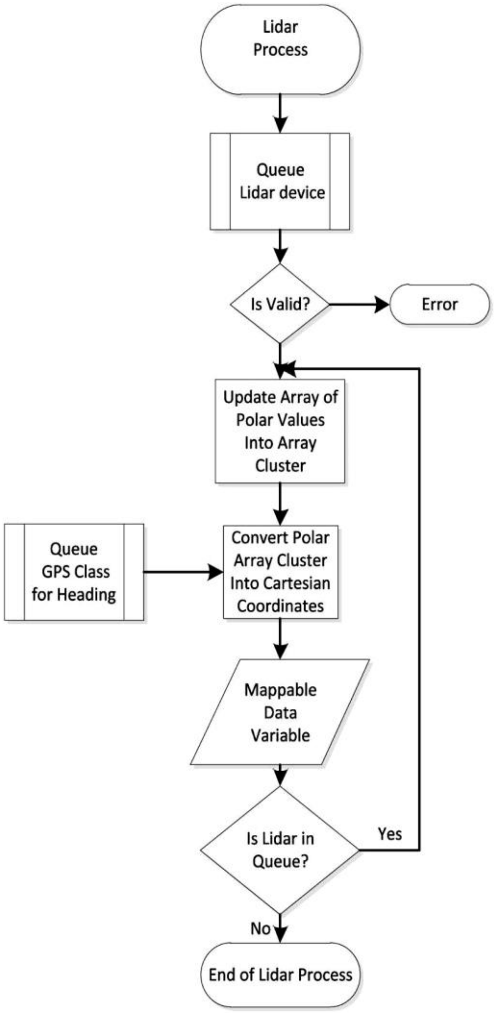


Figure 10. LIDAR decision flowchart

4.4.3 GPS

The uBlox 6-series GPS is used to take a GPS reading when the map is initialized, and can be called to double check the robot’s position if needed. By taking the initial reading while the robot is stationary we found the highest level of accuracy for generating the map and the relative position of the waypoints. The robot is then able to accurately navigate to the waypoints on the map via use of its other navigational sensors.

4.4.4 FPGA

Data from the TRDA-20 wheel encoders and the Devantech CMPS03 magnetic compass is processed by a Xilinx Spartan 3E FPGA board. The encoders send out two square waves indicating the speed and direction of the drive train. The compass sends out a varying PWM signal based on its relative position. The advantage of the FPGA over a DAQ is its high sample rate due to the 50MHz onboard clock, and its ability to perform basic arithmetic to transform the pulse signals into usable data for the main LABVIEW VI. The FPGA was programmed in LABVIEW to perform the conversion arithmetic based on the product specs. The LABVIEW VI was then compiled into VHDL using a Xilinx compiler in order to program the FPGA.

4.4.4.1 Compass

The Devantech CMPS03 magnetic compass is used as a reference in the construction of the map and determining the robot’s position. We measure the output pulse width using the FPGA and convert to orientation in degrees based on the specification table. A calibration button was added to allow us to reset the cardinal directions based on where the robot is transported to in order to correct for error between magnetic and true north.

4.4.4.2 Wheel Encoders

Twin TRDA-20N Series incremental encoders are used to calculate the robot’s position with dead reckoning. The encoders translate rotation into linear distance travelled after taking into account gear ratio and tire circumference.

4.5 Central processor

Due to concerns with the robot’s ability to process video feeds and the need to run LabView, we upgraded the laptop to an ASUS G75VW-DH72B with an Intel Core i7-3630QM 2.4 GHz processor and a solid state drive. Previous issues with the system slowing down while processing video have been eliminated.

5 Software

5.1 Operational Control

5.1.1 Autonomous Mode

5.1.1.1 Navigation

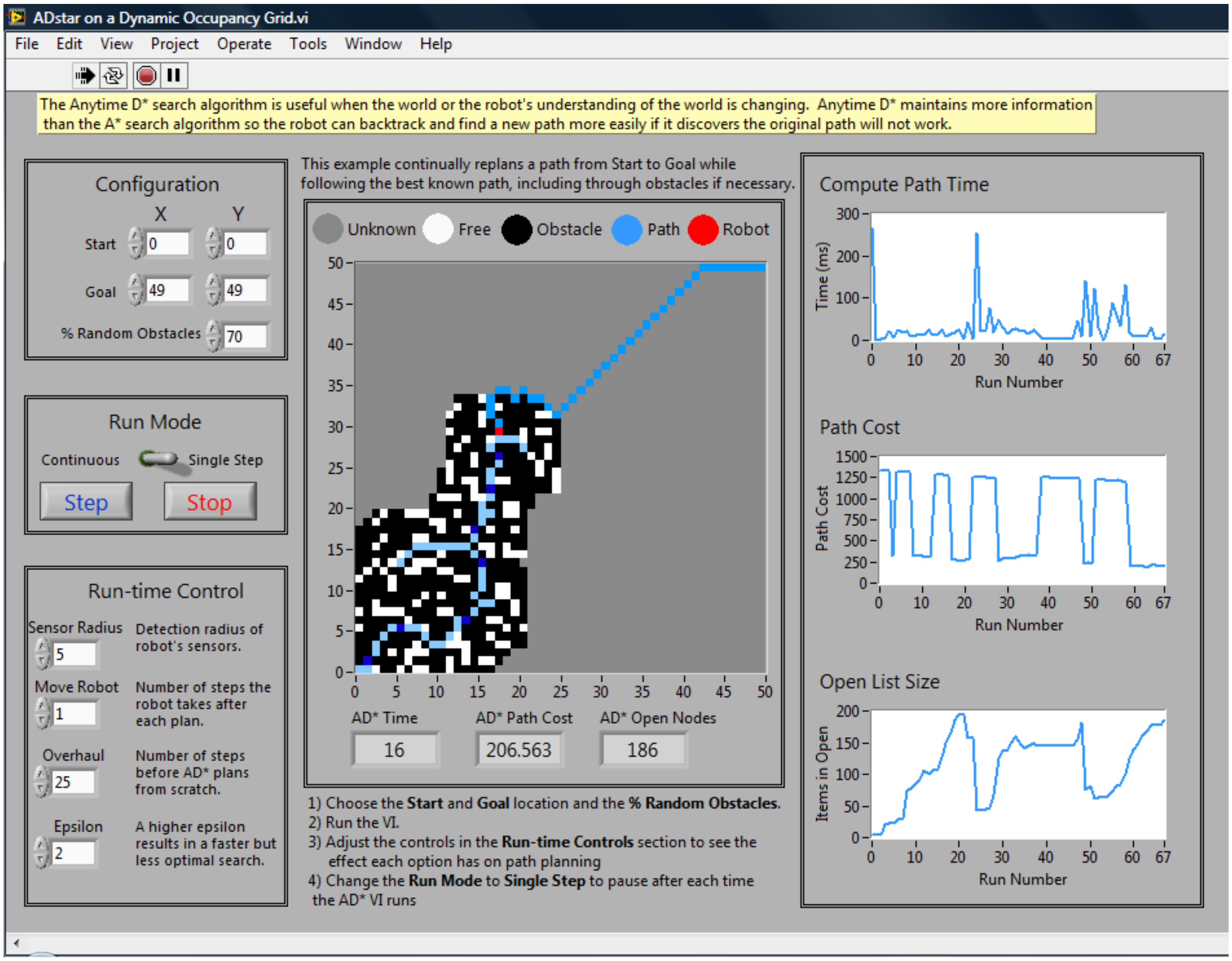
Navigation is very important to the vehicle for both the purposes of the challenge and for its decision making process. A GPS system is used to be able to find the various waypoints on the challenge course, and encoders and a magnetic compass are used to provide additional information to the decision-making VI.

GPS and the magnetic compass are used during initialization to correlate the robot's model map to the geographical competition space. As the robot travels on the course, additional information from the wheel encoders is used to accurately and precisely populate the returns from the LIDAR and camera into the model map. A uBlox 6-series GPS chip was chosen to provide GPS data over other models because of its high level of accuracy, ease of implementation in the LABVIEW environment, and attractive cost. A Devantech CMPS03 magnetic compass was chosen for integration for the same reasons.

In order to provide direct positioning information to the mapping VI, TRDA-20N Series incremental encoders were attached to the drive train. Each encoder provides two digital signals which are used to identify the amount of rotation from the encoder shaft. After taking into account the gear ratio and circumference of the tires, the rotation values from the encoders are translated into linear ground distance traveled. Additionally, the speed of each encoder is monitored during operation. Since an encoder is installed on both the left and right tracks of the vehicle, separate values for the displacement of the left and right side are recorded. These values are then used in a dead reckoning computation to determine the vehicle's current location.

5.1.1.2 Mapping

An AD* mapping and path-finding algorithm is used to make decisions based on sensor input. Information on obstacles is used to assign costs to areas on the map around the robot's position. Each cell of the map corresponds to 0.365223 feet of real-world space. A best fit path is determined to avoid the obstacles and information is used to send the appropriate signal to the motors. AD* mapping was investigated by the CLETIS team, but never implemented.



The Anytime D* search algorithm is useful when the world or the robot's understanding of the world is changing. Anytime D* maintains more information than the A* search algorithm so the robot can backtrack and find a new path more easily if it discovers the original path will not work.

Configuration

	X	Y
Start	0	0
Goal	49	49

% Random Obstacles: 70

Run Mode

Continuous ○ Single Step

Run-time Control

Setting	Value	Description
Sensor Radius	5	Detection radius of robot's sensors.
Move Robot	1	Number of steps the robot takes after each plan.
Overhaul	25	Number of steps before AD* plans from scratch.
Epsilon	2	A higher epsilon results in a faster but less optimal search.

This example continually replans a path from Start to Goal while following the best known path, including through obstacles if necessary.

AD* Time	AD* Path Cost	AD* Open Nodes
16	206.563	186

1) Choose the **Start** and **Goal** location and the **% Random Obstacles**.
2) Run the VI.
3) Adjust the controls in the **Run-time Controls** section to see the effect each option has on path planning
4) Change the **Run Mode** to **Single Step** to pause after each time the AD* VI runs

Figure 11. Mapping algorithm VI front panel view. A random map has been generated. The blue path remains stored in memory.

5.1.2 Local Control

For ease of transport an additional VI is written for manual control using a wireless X-box 360 controller and receiver. The triggers and left thumbstick are used to vary the voltages provided to the drive train and control steering.

6 Conclusion

The robot is an improvement on previous designs largely due to the work on sensor integration through the mapping VI, which was a significant challenge. The robot is well equipped for competition on the Auto-Nav course and therefore a good test platform for the challenges confronting autonomous land systems. For future years we would like to see a re-examination of the chassis and drive train, and effective implementation of JAUS protocols for vehicle communication.

7 Acknowledgements

The authors gratefully acknowledge the financial contributions of The Citadel Department of Electrical Engineering, Google, and SAIC, as well as the expertise and encouragement of Dr. Ronald Hayne, Dr. Mark McKinney, Dr. Johnston Peeples, and Mr. Darrell Crawford, and the team members of CLETIS and Hawk-eye, particularly Mr. Kendall Nowocin and Mr. Jason McLemore for their advice and previous work on the project.

SAIC®
From Science to Solutions

Google

THE CITADEL
SCHOOL OF ENGINEERING