Q

2014 IGVC Design Report

Team Members:
Hokchhay Tann '14, Bicky Shakya '14, Vishal Bharam '14,
Alex C. Merchen '14, Philip Cho '15, Binod Giri '15

Faculty Advisor:
Dr. David J. Ahlgren

May 10, 2014

Trinity College
HARTFORD CONNECTICUT

Faculty Statement

This is to certify that Q has undergone significant redesign in both hardware and software from last year's IGVC entry. The Q team members worked on the robot as an Independent Study project and received 0.5 credit (1.5 credit hours) per semester. This project is a significant and has led to many senior design projects in both Computer Science and Engineering.

David J Ahlgren

Dr. David J. Ahlgren
Karl W. Hallden Professor of Engineering, Trinity College

Table of Contents

1. Introduction

In this report, the Trinity Robot Study team presents its eighth iteration of Q, an autonomous ground navigating robot for the IGVC. The team consists of four seniors and two junior members. This year's goal has been to continue improving Q with great focus on the mechanical structure, vision and navigation algorithms, and revision of JAUS according to the new rules for Interoperability Profile (IOP) Challenge.

2. Innovations

The 2013 iteration of Q had a strong mechanical base, but there were problems with the placement of the GPS and an overly large turning radius. In the new design, Q's GPS sits firmly on the middle section of the robot allowing more accurate waypoints navigation. In addition, we shortened the chassis to achieve smaller turning radius and thus, better obstacle avoidance.

On the other hand, last year's image processing algorithm performed well for the basic course, but there were problems with blue and white barrels in the advanced course. The focus of vision system improvement for this year was to enhance its capabilities and run-time performance. The motor control system was also modified to accomplish a smoother and faster navigation. With the enhanced vision system, the robot can traverse the course at faster speed.

Finally, we upgraded our JAUS component to support additional messages as required by this year's Interoperability Challenge.

a. Vision System

Last year, Q's vision system underwent a major change in both hardware and software. A new Basler Scout camera was installed, and a new image processing algorithm was introduced. Although the algorithm was robust for the basic course, we ran into problems with blue and white barrels in the advanced course. This year, the focus of our work is to improve the algorithm. While the work is still under development, we have tested a functional prototype. In addition, the processing steps are also optimized allowing Q to traverse the course at a faster speed.

b. Motor Control Feedback

Previous iterations of Q had open-ended motor control loops, and Q had no means to ensure it was actually traveling at the desired speed. Q's motor controller provides built in encoder functionality, but previously, there was a time delay of approximately five seconds in reading them. This made any attempt to use the encoders in real time completely unrealistic. In 2012, to combat this problem, hardware was designed to directly read the encoder values and input them through a digital input rather than through the serial port. Two optical encoders with 180 pulse precision now give velocity feedback which can be monitored to ensure that speed constraints are adhered to. Velocity and relative position data are also used as feedback for the IOP Challenge.

c. Path Planning

In last year's version, a GPS point trail as well as heading history are recorded and stored, giving the general heading over Q's recent path. This heading is used as a heavy weight in Q's VFH cost calculation, giving Q a bias away from the path it has already taken (see Section 5.4: Intelligence). In specific situations, such as an area of dense obstacles, Q will stop recording a path history so that the general heading is not skewed by maneuvers around obstacles and the correct bias is preserved. When Q leaves the obstacle field it again beings recording a general heading. Previously to find waypoints in the GPS challenge, Q determined a heading based only upon the nearest GPS waypoint. This heading was inserted into the VFH intelligence algorithm. In 2011 also, Q has added the A* path planning algorithm, thus taking future waypoints into account. This year, major improvements were made to the existing algorithms, VFH and A*, on Q to provide a smoother path planning using data from the improved vision system, which allows Q to see much further ahead than the webcam.

d. IOP Challenge

Last year, in the JAUS Challenge (now IOP), Q was able to seamlessly interface with the Judge's COP. However, some information was missing when responding to the judge. This year, all required messages are added. In addition, the new rule for IOP requires more subsets of messages to be present.

3. Design Process

a. Team Organization

In the fall 2013, members worked together to make mechanical changes on the robot. In spring 2014, the team members were assigned into small groups based on the year's work focus. Each group was in charge of solving its main problem and reporting to the team on a weekly basis, during which brainstorming and major decision making took place by the entire team.

1. **Management**: this group was responsible for leading the team and making all final design decisions. This group also manages all logistics ant the planning for IGVC.

2. **Vision System**: this team was tasked with upgrading the vision system of Q. They were responsible for any hardware or software modifications related to this challenge.

3. **Navigation**: this group was responsible for navigation controls. It ensured the functionality and efficiency of the motor control. The team worked on both hardware and software to achieve smooth navigation.

4. **IOP**: this group was responsible for the development and testing of the IOP Implementation on Q.

Members were divided into these groups based on their expertise, interests and workload associated with each group. Most members contributed to more than just one group. All members were required to work together for a minimum of four hours per week as well as attend weekly meetings to give progress reports and discuss their projects.

Since Q is a multi-year project, key members graduate every year and knowledge that was once common to all members is lost. To combat the high turnover rate, each new member works with an experienced mentor on the team who passed on his or her expert knowledge. Furthermore, regular workshops were held on topics such as FPGA programming, Solidworks modeling, and LABVIEW coding to help new members master wide variety of skills.

b. Design Methodology

The team followed an iterative design process to improve Q. The process started with a detailed failure analysis of Q's performance in the IGVC 2012. After understanding the faults of Q and analyzing IGVC's new rules, a detailed list of requirements was created. Based on these requirements, strategies were proposed and continually tested to reach a finalized implementation.

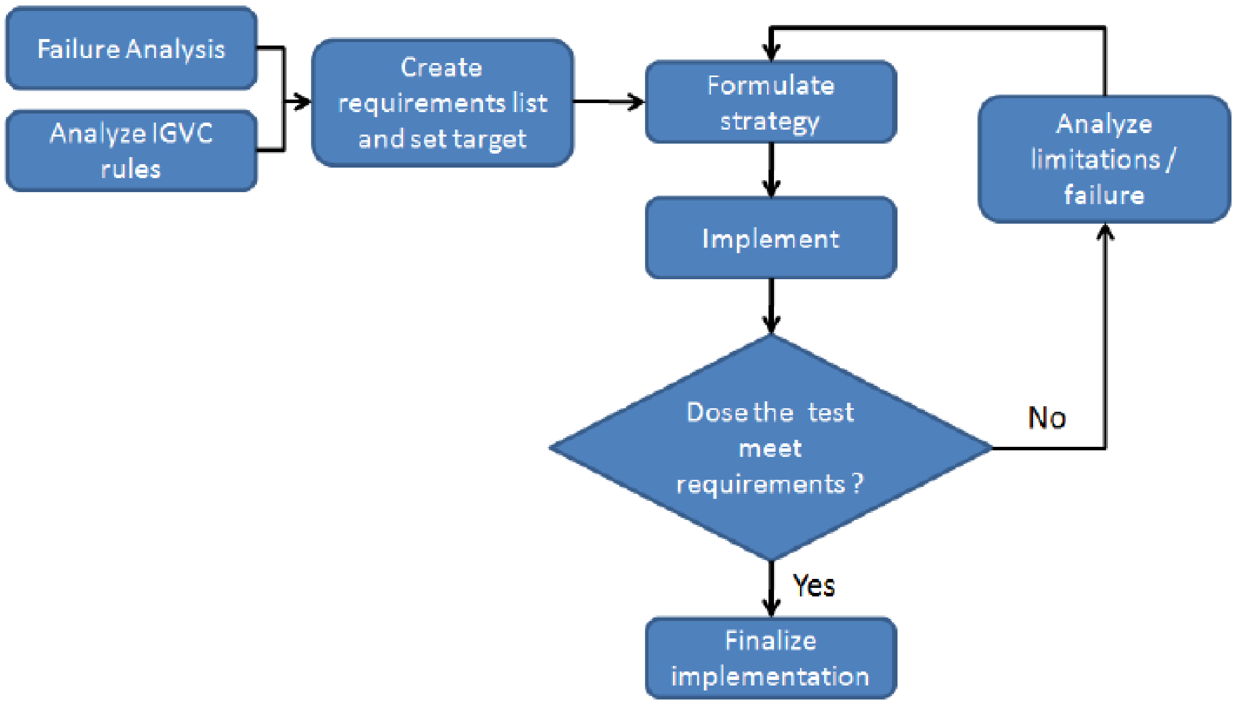
Failure Analysis
Analyze IGVC rules
Create requirements list and set target
Formulate strategy
Implement
Dose the test meet requirements ?
No
Analyze limitations / failure
Yes
Finalize implementation

Figure 1: Design Methodology

4. Hardware

a. Chassis and Drive Train

The physical platform of Q is a modified PerMobil Trax all-terrain wheelchair. This frame can support a payload of over 250 lbs [1] and has a small footprint of 40" by 26". It features a differential front wheel drive system – a pair of 500W Leroy Somer MBT1141S motors - and a pair of rear mounted casters. The motors are geared with a 25.8:1 ratio, providing 15ft-lb of torque. Additional sensor and payload mounting frames were constructed using 80/20 extruded aluminum channels. The use of these channels allowed for quick and easy component layout without compromising mechanical strength.

For previous year's design (fig. 2a), Q has a turning radius of 26 inches and a non-ideal position for the GPS. In the new design (fig. 2b), the GPS is relocated, and the turning radius is decreased to 17 inches.

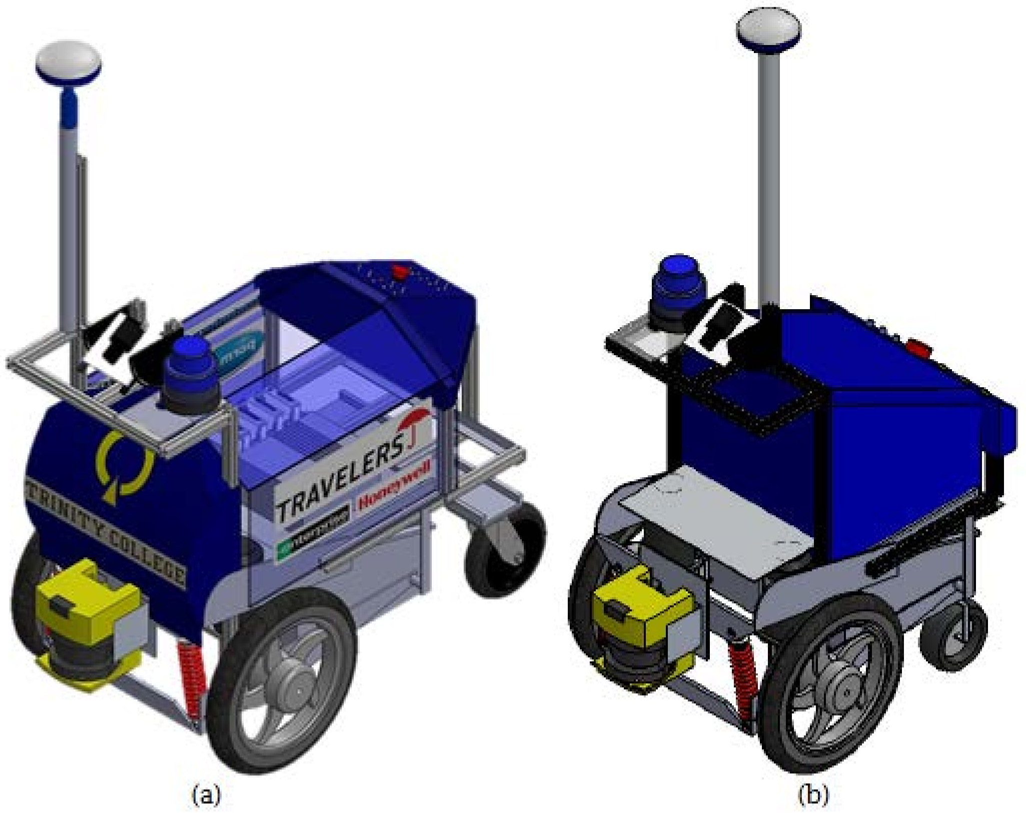

Figure 2: SolidWorks Models of Q: (a) 2013 edition, (b) 2014 edition.

b. System Integration

The cRIO is the central control unit of Q. All motor control, data collection, navigation algorithms and sensor interfacing (with the exception of vision processing and cameras) occurs on the cRIO. This communicates with the NI EVS which interfaces with the two webcams and executes image processing for lane detection, flag detection and ramp detection. The onboard Linksys WRT-54GL wireless router allows wireless monitoring and debugging of the system. The complete system diagram is as follows:

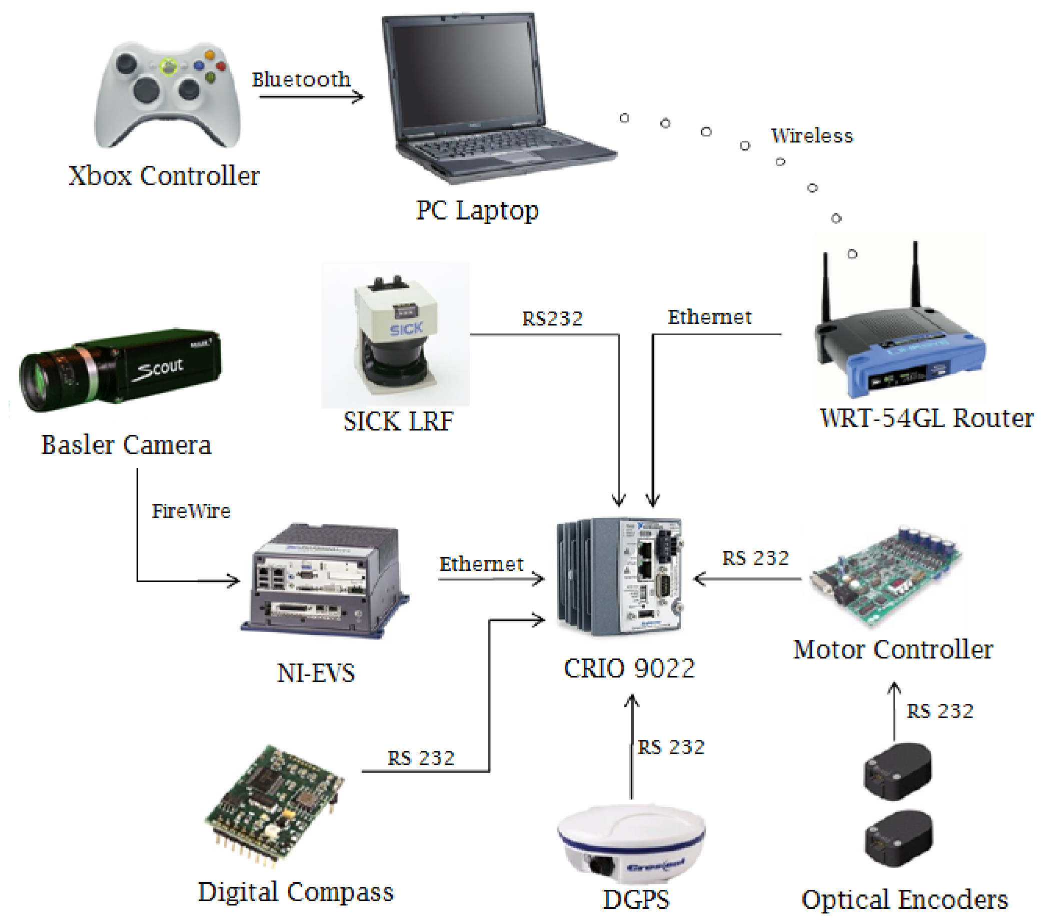

Figure 3: System Overview of Q.

c. Control Panel

The hardware control panel allows easy control of the robot without the need of a software interface. The primary control panel allows easy operation of the robot when powered on. By simply initializing, turning off the motor safety and then pushing the GO button, the user is able to start a competition run.

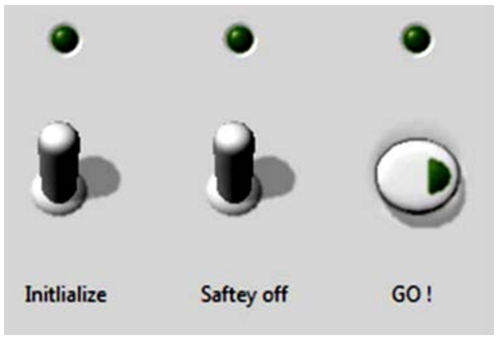

Figure 4: Main Hardware Front Panel.

An additional secondary control panel offers the user more options and also gives detailed dynamic feedback of Q's operation. Here, the user is able set competition challenge type and also reset the entire control program without the need to reboot the system. An array of LEDs provides the user with real-time status of the various sensors and the control program. An additional, tri-state LED provides battery health data.

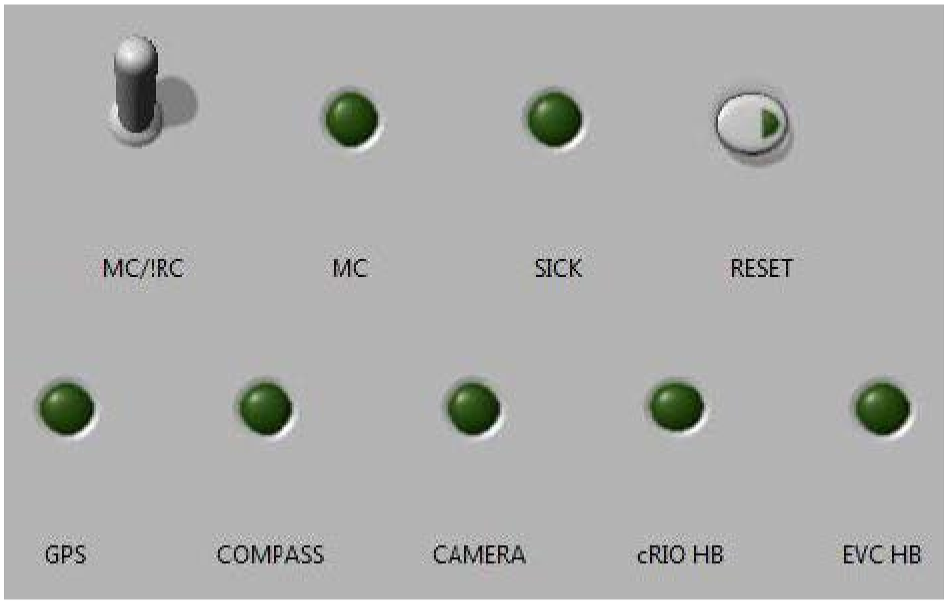

Figure 5: Secondary Hardware Front Panel.

In previous years, Q's front panel was controlled by a custom wired breadboard. Although the setup was functional, the wiring was not secured properly and very difficult to debug. This year, we upgraded the board by designing a printed circuit board (PCB) for the circuitry as shown in fig. 6. In addition, we have added a voltmeter to the front panel to monitor the robot's battery level, which was a major problem in previous years' competitions. The final design is shown in fig. 7.

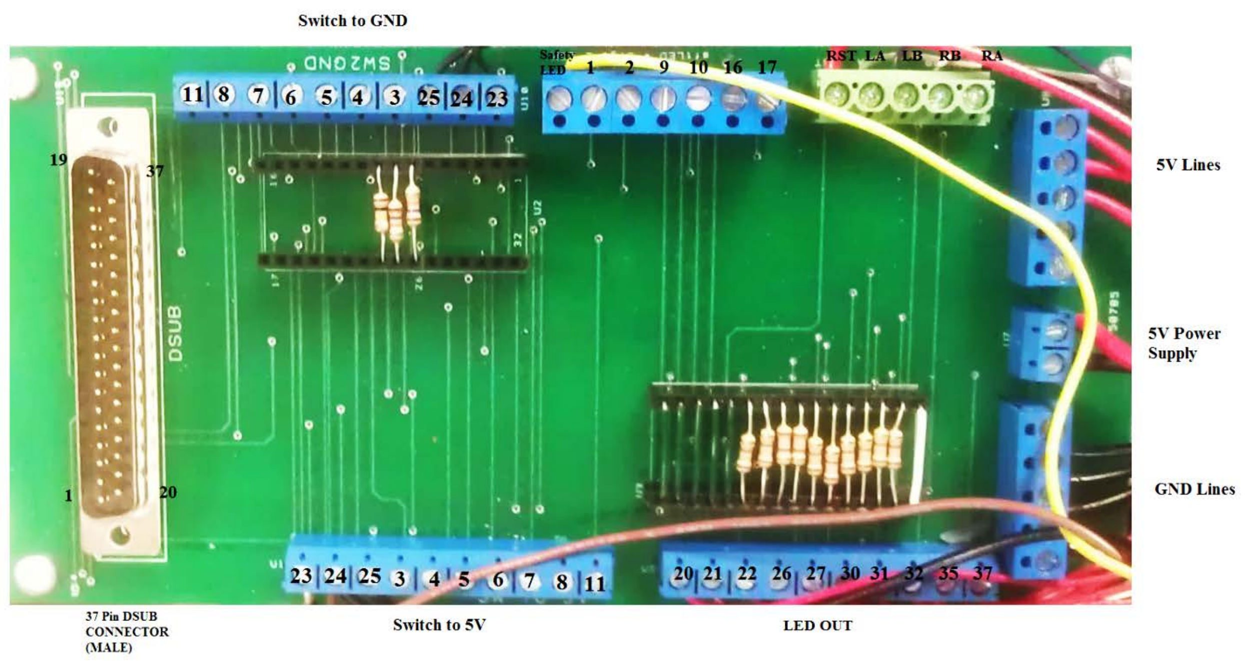

Figure 6: Printed Circuit Board (PCB) for the Front Panel Controller.

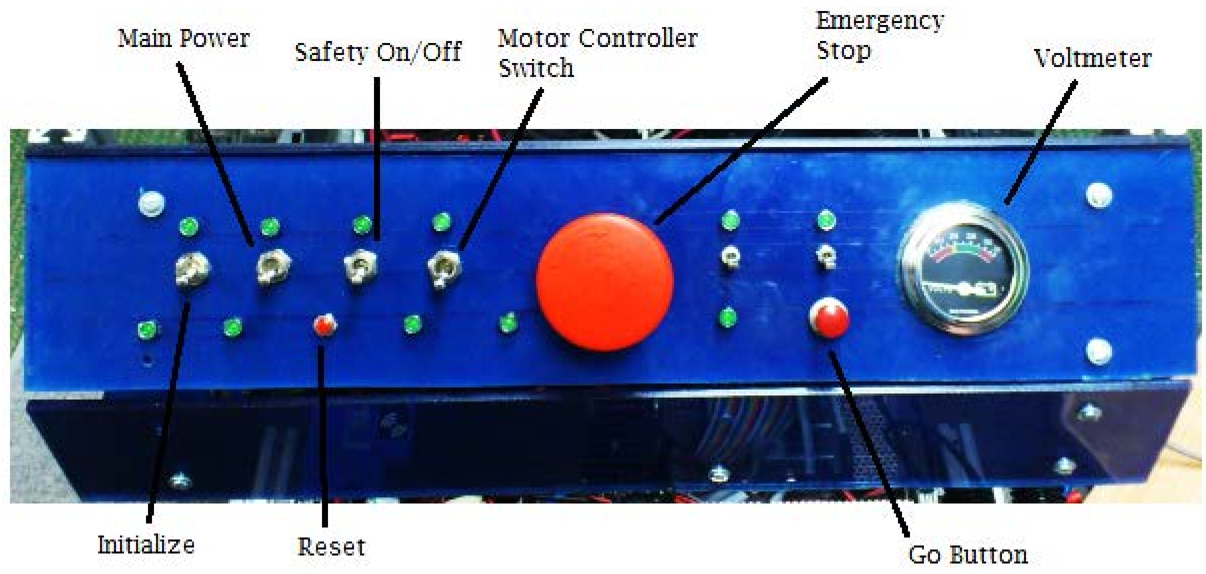

Figure 7: Q's front panel

d. Power Supply

The power system runs all electrical components (other than the motor) though a single power board. The new power board features separate DC-DC convertors for 24V, 12V and 5V power supply. They are rated at 100W, 75W and 75W respectively. In addition, Switchlock circular connectors were added for all major components to allow easy removal and reconnection.

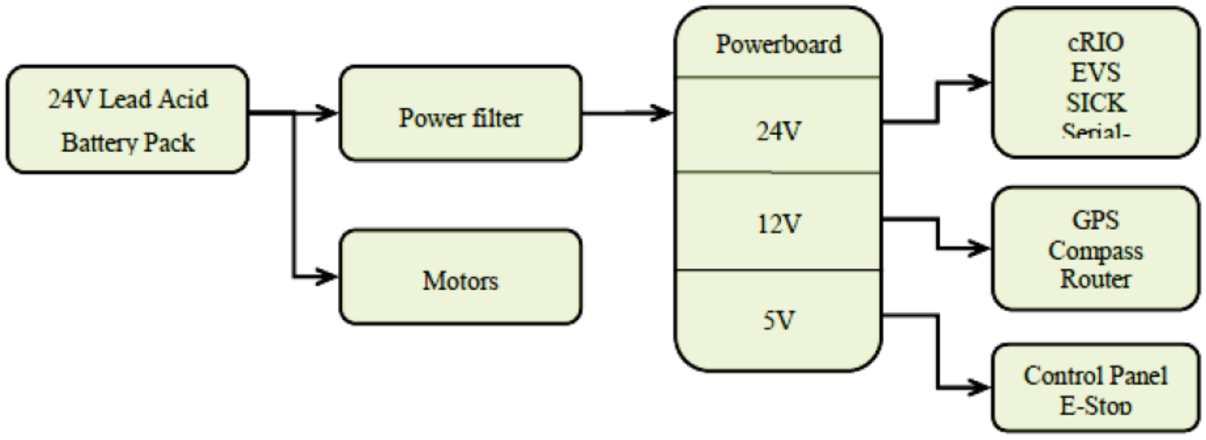

Figure 8: Power Distribution on Q

To secure sensitive electrical components from transients, high frequency feedback, and surges from the motors, a 10A Radius Power Filter model RP220-10-4.7 was added to the motor line. In addition to the filter, slow-blow fuses have been added for all major electrical components to secure them in the case of accidental short-circuits.

5. Software

In light of software performance issues in last year's competition, the team has decided to update certain sections of the code running on the Q for better reliability, flexibility and execution speed. For example, the image processing algorithm has been optimized for better frame rates and the software architecture has been radically redesigned to take advantage of the additional processing power. Additionally, new code has been written for various new system components and features. In the following sections, we will describe the design of the individual software components.

a. Software Architecture

The core of Q's control system is located in the cRIO and interfaces with most of the sensors and devices on the Q, including the motor controllers. All movement of the robot, autonomous or remote-controlled, is directed through this main software architecture, which is shown in fig. 9.

Our goal in designing the main software architecture was to create a system with maximum performance and reliability, but also to make our software as intuitive and as modular as possible. To that end, our design is based on a parallel architecture, where each component runs relatively independent of other components. Our implementation of this architecture uses parallel loops, each containing code pertaining to one particular component of the robot. For instance, each of the sensors interfaced with the cRIO and the motor controllers is controlled by a separate loop. The sensor loops continuously update data buffers, while the motor controller loop waits for active commands from the main control loop. This is to ensure that there is absolutely no undesired movement of the robot.

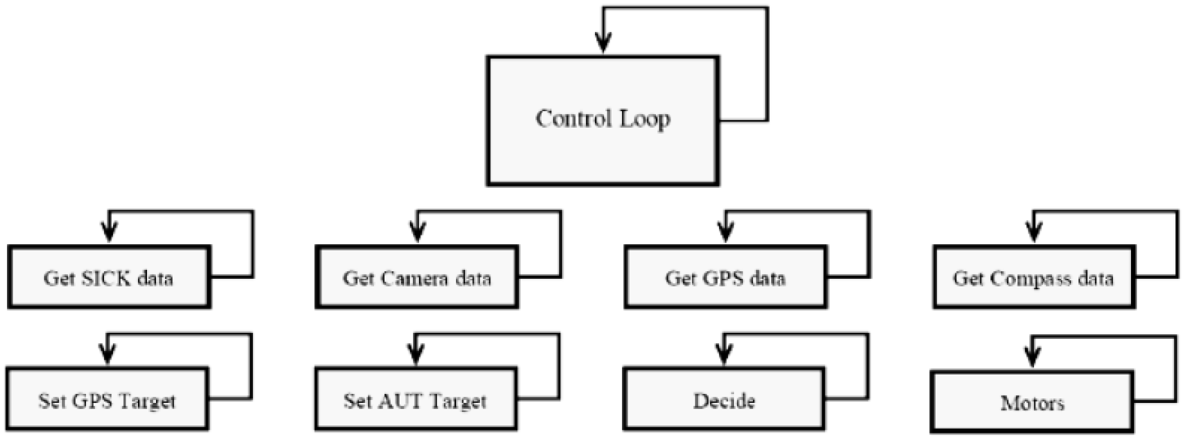

Figure 9: Parallel Software Architecture on Q.

The main control loop directs data flow between loops, thereby controlling the behavior of the system. For example, when in the „remote control" state, motor commands received from the tablet PC are sent to the motor controller loop, and while in the „autonomous" state, the motor commands are computed by the VFH loop. Communication with the tablet PC and the Embedded Vision System are again achieved by using separate UDP loops.

The parallel architecture has several advantages, for instance, it allows the overall throughput of the system to be much higher, allowing for smoother control of the robot. The reliability of the software seems to have also improved, as we have seen far less system crashes than last year. Furthermore, the new architecture has proved to be much more intuitive than its predecessor, since the overall structure is much more natural to understand.

Having parallel loops has also allowed us to modularize the code and work on separate parts individually, increasing efficiency and reducing the amount of time spent on piecing the various sections of code together.

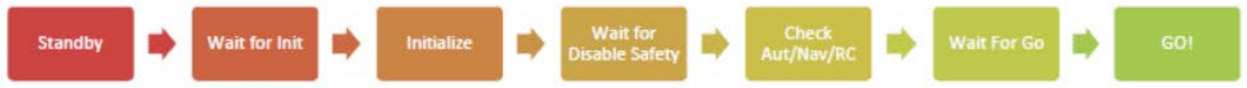

Figure 10: Control Loop State Machine

b. Software Interfacing

The camera is connected directly to the EVS's FireWire ports. LabVIEW provides library functions for capturing images from the camera. The GPS, compass, and SICK LIDAR interface with the cRIO through the FPGA. Each of them connects to one of the four ports on the cRIO RS232 module, which is directly interfaced with the FPGA. The code to read the compass data is written entirely in the FPGA. The FPGA examines the bytes coming through the serial port until it sees the line feed character. At this point it begins reading the heading, pitch and roll, which are sent as "heading'', "pitch'', "roll''. One example would be 100,15,33$<$LF$>$ for a heading of 100 degrees, a pitch of 15 degrees, and a roll of 33 degrees. Each number is extracted as a string, then converted to an integer data type, and passed along to the real time controller.

The code to interpret data from the GPS runs on the real-time controller in the Compact RIO. The FPGA is programmed to act as a serial port. The GPS outputs longitude and latitude following the NMEA format. All latitude and longitude data are preceded by the string \$GPGGA, so the GPS interpreter searches for that string, and then parses the following string for the latitude and longitude, which are converted to double floating point values and transmitted to the Compact RIO.

To read data from the SICK LIDAR, some initialization commands must be sent. Once these commands are sent, the SICK sends 360 bytes preceded by a 6 byte header. Those 360 bytes are 180 16 bit distance values in centimeters - one value for each degree, starting at -90 degrees from the heading and going until 90 degrees from the vehicle heading. The FPGA code searches for the 6 byte header and then reads each pair of bytes into some of the dedicated memory in the FPGA.

c. Communication

There are several components in the system that must communicate for the purpose of debugging, monitoring, parallel processing, and remote control of the robot. Fig. 3 shows how all the processing components in the system are connected.

The EVS and cRIO each have alternate Ethernet ports which are used for direct communication between the cRIO and the EVS. The primary Ethernet ports are used for the wired connection from the EVS and cRIO to the Linksys wireless router. The router is used for wireless debugging, development, and monitoring of the robot. A LabVIEW application for real time debugging was created that displays the exact information Q is using to navigate, such as current GPS location, current heading, target heading, SICK array, compass reading, a waypoint checklist, battery voltage, sensor initialization, and the control loop state machine state.

d. Intelligence Algorithms: VFH+

A modified form of the Vector Field Histogram (VFH) algorithm was used for obstacle avoidance. Using a SICK LIDAR sensor, a compass, and camera data, a polar histogram representation of the obstacles is created and paths are planned accordingly. The algorithm utilizes a cost function which considers the target direction, wheel orientation and previous direction. When there is more than one opening between obstacles, this cost function is used to select the best candidate direction. The opening with the lowest cost is chosen. The result is a heading which represents the best path. Several improvements from previous years were instantiated. The scan, which covers 180 degrees, was split into 180 separate sectors, rather than 30, for path consideration for better resolution. Detected obstacles were widened to account for the robot radius and turning ability. Finally, the SICK data has been split into three distance thresholds, with the nearest threshold (at one meter) receiving priority in motor functions. The further two thresholds, at two and three meters, are considered in path planning. They influence the cost calculation in the same manner as the first threshold, with robot direction and target heading taken into account, but have less weight than the threshold at one meter.

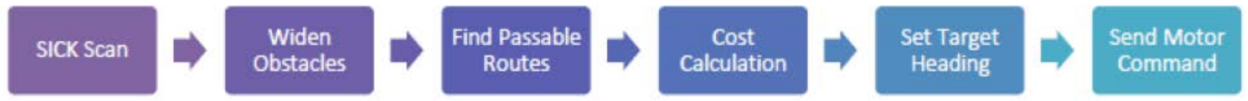

Figure 11: VFH Loop

Special VFH States

Several VFH states have been added to Q outside the “normal” state described above. These states are triggered by specific signatures detected by Q‟s sensors. Examples of this would be when the ramp is detected by the SICK and Ping sensors, or when slalom gates are found from the cameras. When the ramp is detected, Q will give lowest cost to the path of the steepest gradient, so that lateral motion on the ramp is limited. When slalom flags are detected, a high cost will be placed on paths outside of the slalom gates.

Smart Path History

Q now uses a smart path history to calculate a target heading to help navigate and avoid turning around. Q uses GPS co-ordinates taken at time intervals to calculate and store past headings. A weighted average of this history gives a general heading that roughly follows the direction of the course. This assists Q in determining the general direction of the on track. This is illustrated in fig. 12. However, using a simple averaged path history fails while traversing a complex chicane. While traversing a chicane, Q may be required to turn at a direction that is radically different than the general direction of the course. Using heading readings from inside the chicane to calculate an averaged path history may result in a target direction that leads Q into a dead end. Such a scenario is identified in fig. 13.

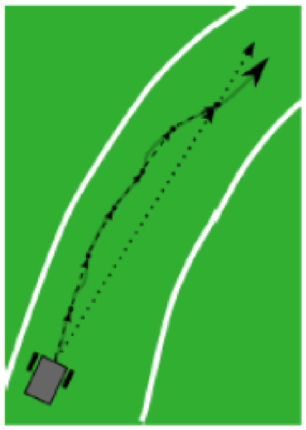
Figure 12: Usage of path history to determine general direction of the course

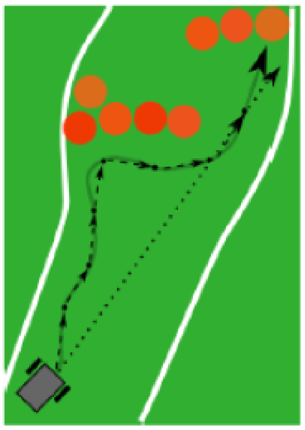
Figure 13: General path taken by Q with simple path history. Here, Q is driven to a corner

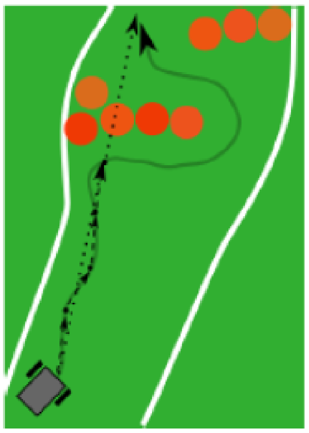
Figure 14. General path taken by Q with environment aware path history. Here, Q successfully traverse the chicane

To overcome the challenge of traversing a chicane, Q uses an environment-aware path history. Q uses the concept of an obstacle field to select an alternate path. Here, an obstacle field is defined as an environment where there is a large density of obstacles such as in a chicane. When Q is not in an obstacle field it continuously logs path history to calculate a new target heading. But when Q encounters an obstacle field, data collection for the path history is temporarily suspended and the target heading calculated before entering the obstacle field is held constant. This helps Q maintain a sense of direction even while traversing a chicane. Fig. 14 illustrates how, this algorithm helps Q traverse a chicane.

e. Waypoint Navigation

The autonomous and navigation algorithms were merged. This was accomplished by switching from the autonomous algorithm to the navigation algorithm after the first GPS waypoint is reached. When the robot is in the navigation portion of the obstacle field, Q creates an creates an X-Y coordinate system with its current location as the origin and plots each waypoint with coordinates based on the distance of the waypoint from Q's original location. Q then moves toward the closest waypoint by assigning a very low cost to the heading of the target waypoint in the VFH cost calculation. When an obstacle is encountered, Q plots the obstacle on the X-Y graph and routes a path around it. When the waypoint is reached, Q approaches the next closest waypoint with obstacles that have been plotted considered in shortest distance calculations. If the closest waypoint changes when Q is navigating to a waypoint (i.e. going around a wall and increasing distance from the original waypoint) then Q will pursue the closer waypoint having plotted obstacles along the way. The GPS readings are accurate to within one meter.

f. Image Processing

The Basler camera has been positioned for maximal viewing range with bottom of the image closest to the robot. The field of view is shown in fig. 15.

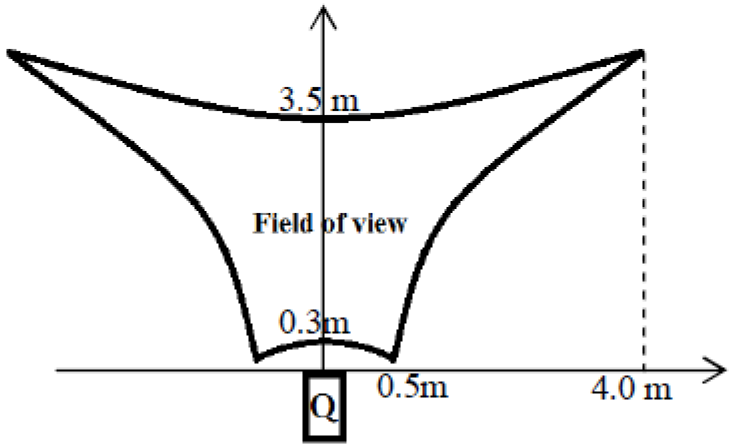

Figure 15: Q's Camera Field of View

A new vision algorithm was developed to achieve a more effective, consistent performance despite varying lighting conditions. Fig. 16 illustrates the processing algorithm for each acquired image. The original image (Fig. 16a) had a RGB color map, and because the background was mostly grass with high intensity in the green plane, a blue plane extraction eliminated most of the grass and distinguished the white lines (Fig. 16b). X1.5 lookup table and X0.5 were then applied to the blue plane consecutively to further enhance the contrast (Fig. 16c). The green plane of the original image (Fig. 16d) was then subtracted from the blue plane to eliminate noisy reflections from the grass (Fig. 16e). LabVIEW's auto threshold by metric was then applied to the image, generating the binary image (Fig. 16f). Hough Transform was then used to reconstruct the white line from the image, such that any noise left over was automatically removed (Fig. 16g).

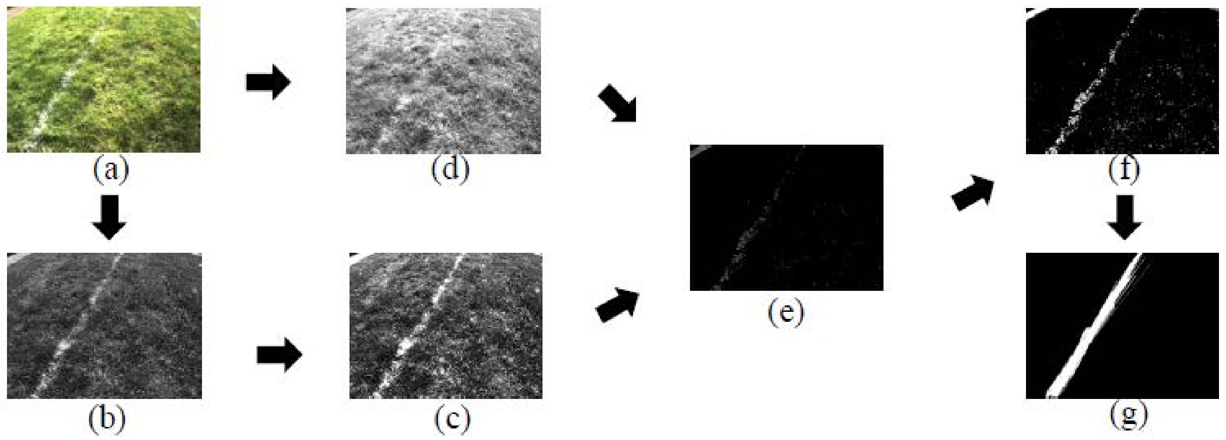

Figure 16: Illustration of the image processing algorithm employed on Q for the 2013 IGVC: (a) Original image, (b) Blue plane extraction, (c) Blue plane ^1.5, (d) Green plane extraction, (e) (c) – (d), (f) Threshold by metric, (g) Hough Transform.

The processed image then underwent a calibration procedure which analyzed the foreground object (the white line) and created a 2D occupancy plot (fig. 17d). The calibration mask (fig. 17a) contained real-world coordinates of where each white circle was relative to Q. This information is used to transform input image into a corrected image (fig. 7c). The resulting plot data (fig. 17d) is sent to cRIO to be combined with the SICK data for Vector Field Histogram.

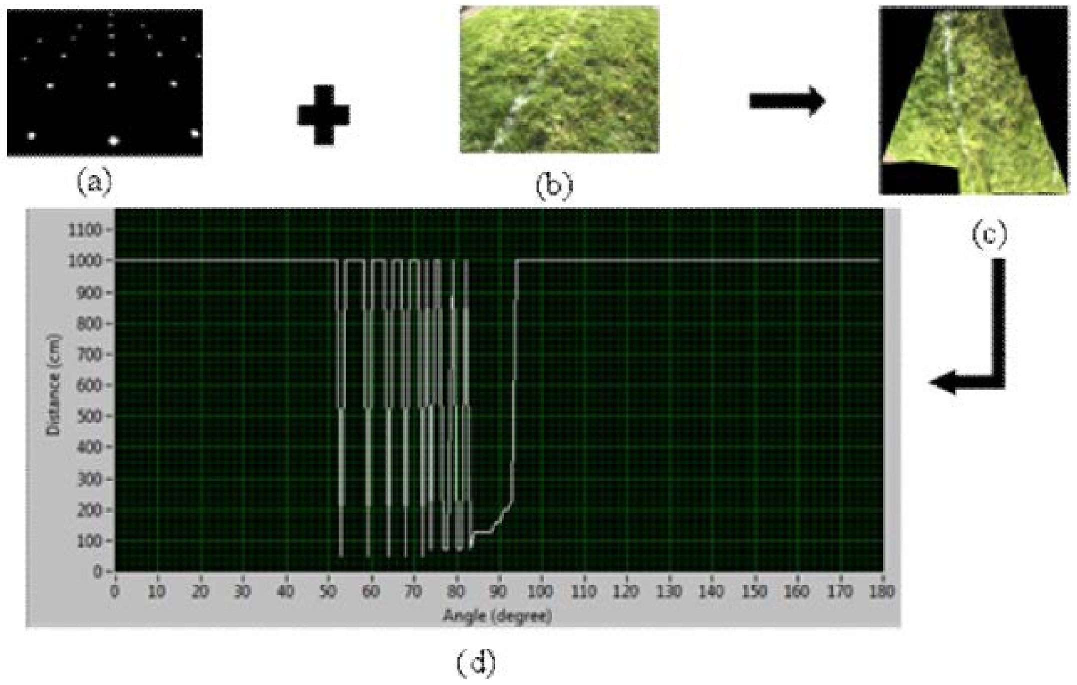

Figure 17: Image calibration process (a). calibration mask, (b). input image (hough transform image), (c). corrected image, (d). plot of obstacle distance versus angle.

The throughput of the vision system has limited the reaction time of the robot. The algorithm has undergone pipelining. The various subtasks in the algorithm now run parallel to each other thereby reducing the execution time and increasing throughput. Portion of the code has also been ported to the FPGA of the EVS, thus decreasing processing time.

In addition, new software was written to check the brightness of each input image by calculating the mean and standard deviation of the grayscale level on the blue plane. The result is then compared to a lookup table. If the image is determine to be too bright or too dark, the gain of the camera is decreased or increased respectively. Otherwise, no change is made to the gain.

g. Motor Control

The lack of proper speed control in the previous year led Q to rapidly decelerate or accelerate when approaching or moving away from obstacles. In order to combat the issue of rapid speed change, we implemented a step-wise speed control algorithm. This algorithm converted raw SICK data into actual distance measurements and enabled Q move at 5mph when more than 3m away from the nearest obstacle. For distances between 1 to 3m, Q's speed decreased as a linear function, resulting in gradual deceleration as opposed to a sudden halt when approaching an obstacle.

6. Interoperability Profiles Challenge (IOP)

a. Previous JAUS Implementation on Q

Our robot features a single JAUS component to handle all the required JAUS services. Even though JAUS supports mechanisms for communication between different components of the robot, we decided not to adopt those mechanisms. Building JAUS components on top of the cRIO, the vision system, and sensors would have added significant overhead in performance without discernable benefits. Instead, we have a separate JAUS component that provides a JAUS-compliant interface to the remote client. An internal communication mechanism is used among the components that is more cost efficient.

The JAUS component consists of five parallel loops that communicate via queues as shown in fig. 18. LabVIEW® supports queues that are safe to use in multiple threads of execution. We adopt a common parallel pattern called Producer-Consumer, where one loop puts messages to a queue while another loop fetches them from the queue.

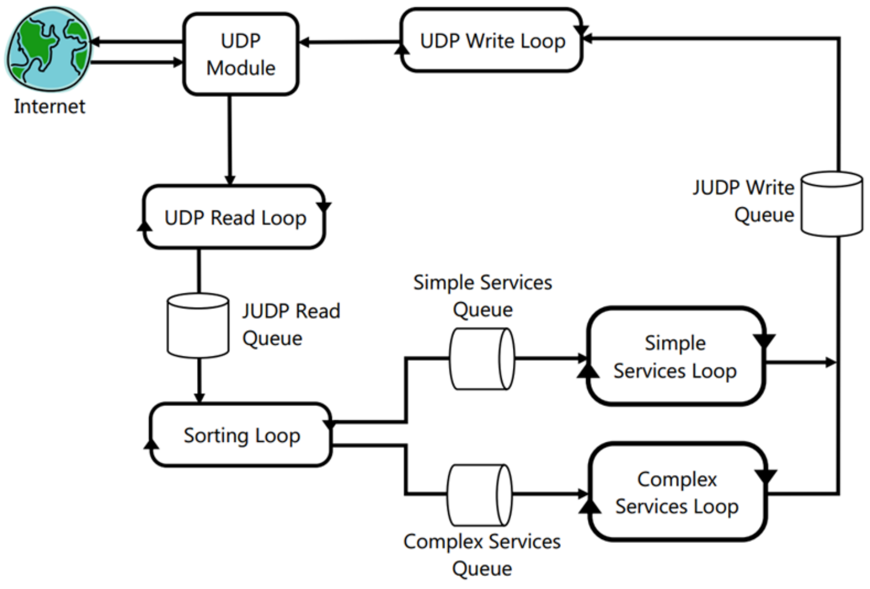

Figure 18: Parallel Implementation of JAUS on Q

We classify JAUS messages as either simple or complex. Complex messages require the loop to keep track of multiple overlapping states at once. Rather than implementing the finite-state machine ourselves, we make used of the Statechart Module provided by LabVIEW®. The module lets us simulate states containing substates.

The JAUS component also comes with a set of subroutines that either encodes or decodes JAUS messages. The Statechart depend on those subroutines. We recently made a complete documentation of the subroutines so that our successors may understand their uses.

b. Upgrades to IOP

Last year, only a subset of messages had to be present. We upgraded our JAUS component to support additional messages as required by this year's Interoperability Challenge.

Most of the additional messages are part of services that our JAUS component already provides. Others form a new service called Events. The Events service regularly notifies the remote client of such information as velocity, local position, status, and heartbeat. The client specifies those pieces of information it wants and the interval at which it want to be notified. The JAUS component is to send response messages at the specified time interval. This particular service is specified in the SAE JAUS Profiling Rules but was not required by last year's competition. We added a global events manager that periodically generates response messages.

We had a problem with reporting of velocity and local position last year. The real-time loop reading signals from the motor encoder made an incorrect conversion of velocity units. We revised the loop so that we have accurate velocity state readings.

Adding new messages and making revisions required extensive testing. We wrote a modular testing framework. The framework lets us generate any string of messages we want and send them to either the real JAUS component or a simulated one. It takes minimal work to support a new class of messages because the code for sending and receiving messages is separate from the code for generating and decoding them. All it takes to add a class of messages is two small modules to generate and decode these messages. On an incidental note, the framework also helped us discover bugs and undefined behavior from the existing code base.

7. Safety

Safety has been given the utmost priority in the design of Q both for the electrical system and the mechanical system. Wires of gauge 10 were used to connect power sources to the motors and filter, and 16 gauge wires were used elsewhere. A circuit breaker was used for the entire electrical system. Slow-blow fuses were instantiated into the connections to each component from the power board to ensure that individual electronic devices did not receive too much power. A single phase dual stage power line filter inserted to prevent transient current from the motors.

Three main motor safety measures have been implemented. They include the motor control board FETs, the physical emergency-stop, and the wireless emergency-stop. The motor control board FETs are now controlled by the program to disconnect power from the motor whenever the system is in an idle state. The physical and wireless e-stop features also work by immediately cutting power from the motor controllers as soon as one of them is triggered.

a. Wireless Emergency Stop

The wireless Emergency Stop is controlled by a Seco-Larm SK-919TD1S-UP transmitter that uses 315 MHz RF transmission with a one-channel receiver. The range was extended to 150 feet by installing antennae on Q and on the transmitter.

8. Vehicle Cost

Component List	Price (US$)	Cost Incurred (US$)
Basler Camera + Wide Angle lens	900	900
Crescent A100 DGPS	2000	0
Caster Assembly	150	150
Chassis	650	0
Encoders	100	100
Honeywell Compass (HMR-3300)	750	0
Motors	1000	0
NI cRIO-9022 Real-Time Controller	3199	3199
NI cRIO-9133 Reconfigurable Chassis	1999	1999
NI EVS-1464RT Embedded Vision System	4499	4049
SICK LMS291 Laser Range Finder	4000	4000
Power Supply Board	150	0
Encoded Receiver/Transmission Set	77	77
Remote Control	180	0
Roboteq AX3500 Motor Controller	400	300
Wiring/Electrical	50	50
Parallax Ping Ultrasonic Sensor	30	30
Total	20134	14854

9. Concluding Remark

The changes made to Q hopefully will lead to improved autonomous performance. Our goal has been to expand the range of situations that Q can handle when navigating so that each run will more likely be successful. We look forward to participating at the 2014 IGVC.

10. SPONSORS

- Enterprise Rent-A-Car
- Hemisphere GPS
- Honeywell International Inc.
- National Instruments
- PerMobil Corporation
- Travelers Insurance
- Trinity College

11. References

[1]. “USA - Products - Support - Product Support - Permobil." Power Wheelchairs - Permobil.Web. 14 May 2010. <http://www.permobil.com/USA/Products/Support/Manuals/drivers/>.

[2]. Ulrich, Iwan, and Johann Borenstein. "VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots." Proceedings: 1998 IEEE International Conference on Robotics and Automation, May 16-20, 1998, Katholieke Universiteit Leuven, Leuven, Belgium. IEEE International Conference On Robotics and Automation, Leuven, Belgium. Piscataway, NJ: Robotics and Automation Society, 1998. 1572-577. Print.

[3]. Tann, Hokchhay, et al. "New vision system and navigation algorithm for an autonomous ground vehicle." *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2013.

[4]. Wright, Adam, Orko Momin, Young Ho Shin, Rahul Shakya, Kumud Nepal, and David Ahlgren. "Application of a Distributed Systems Architecture for Increased Speed in Image Processing on an Autonomous Ground Vehicle (Proceedings Paper)." Intelligent Robots and Computer Vision XXVII: Algorithms and Techniques. Proc. of IS&T/SPIE Electronic Imaging, San Jose. Vol. 7539. San Jose: SPIE, 2010. Print.