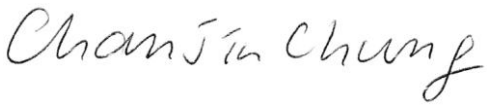
Faculty Advisor Statement

I, Dr. CJ Chung and Jonathan Ruszala of the Department of Math and Computer Science at Lawrence Technological University, certify that the design and development on the iWheels2 platform by the individuals on the design team is significant and is either for-credit or equivalent to what might be awarded credit in a senior design course.



5/16/14

Signature Date

5/16/14

Signature Date

The iWheels2 platform is an improved design evolution of previous robots. Subsystems that are significantly different from those used in previous years include:

- Addition of two sets of stereo vision cameras
- New compass sensor
- New line detection algorithm
- Improved Auto-Nav challenge code
- Improved object and line detection integration

DESIGN REPORT FOR THE 2014 INTELLIGENT GROUND VEHICLE COMPETITION

Lawrence Technological University
Students: Jorge Chagas, Gabriel Laguárdia, Gordon Stein
Faculty: Dr. CJ Chung cchung@ltu.edu, Jonathan Ruszala jruszala@ltu.edu

INTRODUCTION

This report describes the iWheels2 robotic platform, designed and built for the 2014 Intelligent Ground Vehicle Competition (IGVC). This report is organized into sections describing the design team, the design processes, and various aspects of the design, followed by performance and cost information.

TEAM ORGANIZATION

The iWheels2 design team is comprised of the following members:

Name	Course of Study	Role(s)
Gabriel Laguárdia	MSCS	Software, Mechanical, and Electrical Team Leader
Jorge Chagas	MSCS	Software, Mechanical, and Electrical
Gordon Stein	MSCS	Software
Christopher Kawatsu	MSCS	Software Advisor
Jonathan Ruszala	MSCS	Faculty Advisor, Project Manager
CJ Chung	PhD, CS	Faculty Advisor, Project Manager

iWheels2 DESIGN CONCEPTS AND GOALS

“Don't Reinvent The Wheel, Unless You Plan on Learning More About Wheels.” We believe the main goal of the IGVC is not inventing driving platforms, but introducing “intelligent” features to the driving platform. After 10 years of creating our own driving platforms since 2003, this year we decided to use an existing reliable driving platform from a wheelchair not only to save time and money, but also to focus more on software development. In addition, we plan to use the platform for creating affordable and compact intelligent wheelchairs for disabled and/or elderly people. Instead of using expensive and heavy laser scanner sensors that consume lots of power, we also decided to keep using our own low-cost stereo vision system for identifying & categorizing obstacles and detecting lanes. However, based on our 2013 failure analysis, the location, height, and angle of the stereo-camera received special attention in the design of the vehicle. Since interoperability is becoming more important design concepts for unmanned

intelligent systems, we emphasis on IOS development this year. IOS can reduce development and integration time and provide a framework for technology insertion.

iWheels2 DESIGN

Mechanical Design & Fabrication

Chassis

An Invacare motorized wheelchair was used as a chassis for our robot. The platform was chosen to be a cost effective reliable base for software development. The platform is also small and light enough to be easily transported in a small van or SUV. The original battery was not sufficient to power both the wheelchair and the added electronics/computer systems. Consequently, the original plastic battery tray was removed and a larger aluminum battery box was constructed. This allows four 12 volt 35 aH batteries to be mounted flush with the chassis. See Figure 1.

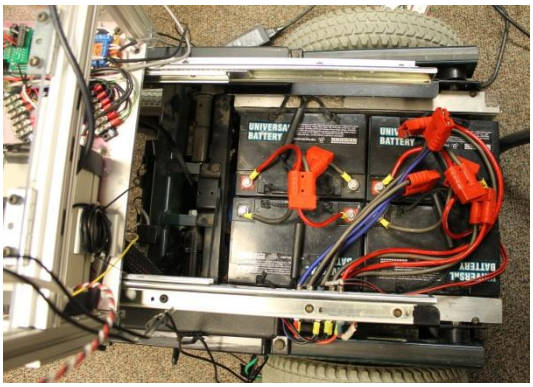


Figure 1. Battery Box

The chair was also removed and replaced by a “t-slot” aluminum case as shown in Figure 2. The case houses the computer system, power supplies, wiring, and most of the sensor interfaces. The electronics are protected from the elements by ¼ inch clear acrylic glass. Clear acrylic was chosen to allow all of the electronic systems status lights to be seen even when the case is closed. Access to the components is done via a hinged lid. The lid includes a lockable strut that provides the user with the option to lock the lid in the open position as well as provides a means of smooth closing.

The entire aluminum framed case is attached to the robot chassis via industrial linear sliders. This allows the frame to slide forward, exposing the chassis and allowing for easy charging and installation of batteries. The frame is locked to the chassis using simple stainless steel cotter pins.

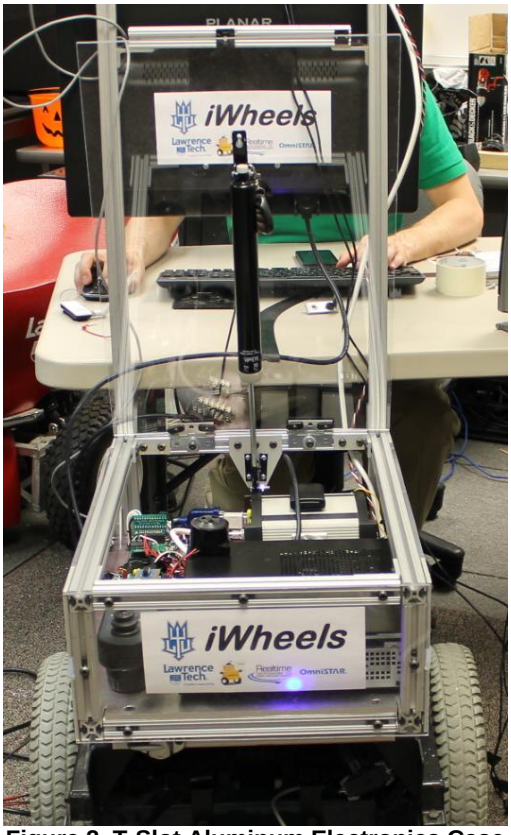


Figure 2. T-Slot Aluminum Electronics Case

Electrical Subsystems Design

The electrical design is comprised of two largely independent subsystems. The 24V subsystem powers the drive motors and motor controller. The 12V subsystem powers the processors, sensors, and controls. Each battery module has two connectors, one for each battery. These connectors are mated to both the charging system and the chassis. For the 24V subsystem, circuitry on the chassis connects the batteries in series. For the 12V subsystem, circuitry on the chassis connects the batteries in parallel. The two subsystems share a common electrical ground reference.

Motors & Motor Controller

For simplicity and cost the original equipment motors and gearboxes on the Invacare wheelchair frame were used. The motors are standard brushed DC motors with an electronic spring loaded brake. The motor controller was upgraded this year from the Invacare motor controller commanded by an Arduino to a Roboteq AX3500. The AX3500 provides H-Bridge control for each brushed motor and incorporates encoder feedback into a PID speed control system. PID parameters were tuned to give the robot fast response with minimal speed overshoot. Since the original Invacare motor control system did not have any type of speed feedback, encoders were installed on the ends of the brushed motors shafts.

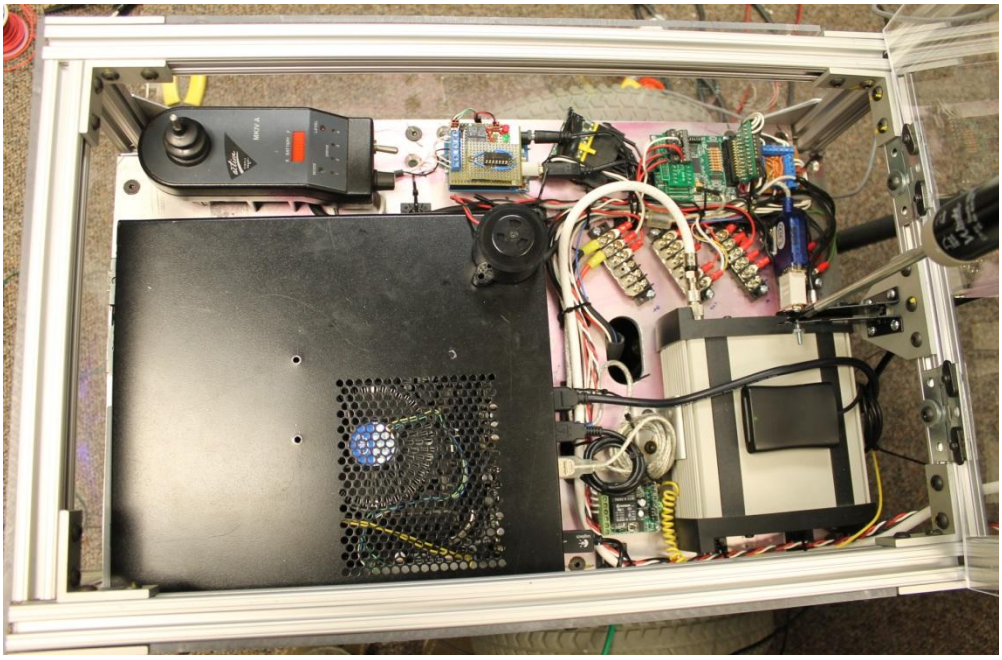


Figure 3. Inside t-slot aluminum electronics case

E-Stop

The wireless receiver has a range of over 150 ft, and can be paired with multiple wireless transmitter fobs. It incorporates a relay. The “normally closed” circuit of the relay is opened momentarily when the receiver is activated via wireless transmitter. During normal operation, both the Manual E-stop switch and the wireless transmitter circuits are closed, causing the (active-low) reset input on the motor controller to be held high. If either the manual E-stop switch or the wireless receiver relay is opened, the reset input on the motor controller is pulled low by the pull-down resistor, and the motor controller is reset to the analog input control state. The circuit includes sense connectors that allow the state of the E-stop system to be monitored by the processors. A circuit board was designed and constructed to implement this circuit. While the manual E-stop switch is open, the motor controller is held in the reset state, and the motors are de-energized. The platform is easily maneuvered by hand while in this state. The wireless receiver has a second channel that is used to produce a one-button ‘start’ signal when initiating autonomous operation.

Power Conversion & Conditioning

The 12V subsystem incorporates two DC-DC 160W converters, designed for use as motherboard power supplies. The first is used for powering the computer systems; the second provides conditioned –12V, +12V, and +5V power to sensors and controls that require regulated power.

Processing Resources

The platform is configured with two computer systems. Each system is comprised of a microATX format motherboard, Intel Core i7-2700K processor, 4GB of RAM, and 64GB solid-state drive. This particular microprocessor incorporates on-board graphics processing. When

paired with a compatible motherboard the resulting system requires no additional graphics hardware, with the result that less power and space are required for the complete system. Each processor is housed in a Cooler Master slim case measuring 12.5" x 10.3" x 2.7", a substantial reduction in space claim over previous years' entries. The integrated power supply was removed from the case (since power is supplied via DC/DC conversion) which allows the CPU fan to displace enough air to keep the system cool without needing additional cooling fans.

General Purpose I/O

The Platform incorporates an ACCES I/O model USBP-II8IDO4A general-purpose I/O board for general purpose sensing and control. This board has 8 isolated digital inputs, 2 16-bit A/D inputs, and 4 solid-state (high-side FET) relays. The A/D inputs are used to monitor the 12V and 24V battery voltage levels for the purposes of measuring remaining available power. The solid-state relays are used to switch power for the safety light and horn. Digital inputs are used to sense E-stop circuit status as well as the reception of the start signal. Communications with the I/O board is over USB 2.0. Power is supplied via the 12V conditioned power source.

Safety Light

We modified an inexpensive 'emergency beacon' to incorporate a standard socket for an automotive turn-signal lamp. The turn signal lamp has two circuits: one controls a low-intensity 'parking lamp' indicator, and the other a high-intensity 'turn lamp' indicator. The 'parking' indicator circuit is hard-wired into the platform power circuit so that it is illuminated at all times that power is available on the platform. The 'turn' indicator flashes under software control during autonomous operation. This is accomplished via a solid-state relay on the general-purpose I/O board. The lamp selected is LED-based and exhibits high-reliability, superior brightness, 360-degree visibility, and low power consumption attributes.

GPS

Platform position is obtained via a NovaTel ProPak-LB GPS receiver that incorporates differential corrections obtained via the OmniStar service for sub-meter accuracy in positioning. Communications with the GPS receiver is via RS-232. Power is from the 12V conditioned power source.

Electronic Compass

Platform heading is obtained via a Sparton GEDC-6 electronic compass, selected for its precision, configurability, and update rate. Communications with the electronic compass is via a USB to serial adaptor. The compass can return roll, pitch, and yaw and incorporates tilt compensation for more accurate measurements.

Web Cameras for Stereo Vision System

The platform incorporates six Microsoft LifeCam Studio cameras. Communication with the cameras is managed through Emgu CV, which is a C# wrapper for the OpenCV libraries which talks to the camera through USB 2.0.

WiFi Network Adapter

When required for the JAUS portion of the competition, the platform uses a Wireless USB Network Adapter. Power and communications are via USB 2.0.

PLATFORM SOFTWARE

Platform processing components use Windows 7 Professional 64 bit as the operating system. Other than device drivers and device libraries provided by device vendors, platform software was developed using the C# language. The C# language was selected for its combination of ease of development, and excellent peripheral support specifically for the off the shelf web cameras which we use for our stereo vision.

Within the software, a number of sub-modules were created, corresponding to major areas of processing. Interfaces were established between groups where appropriate, and development preceded largely in parallel using agile software development techniques. Configuration management was via a secure Subversion server. Using the Visual Studio development environment, along with a subversion client, coordination of code between individuals and sub-groups was a trivial exercise. The sub-groups are: Common Classes, Device Interfaces, Vision Processing & Obstacle Detection, IOS, Global Path Planning, Local Path Planning, and Simulation.

Common Classes

A number of common classes were developed for use by all developers where appropriate. This includes common-use patterns, data structures, dimensional, and geometric classes.

Two coordinate systems are used during processing. The 'local' coordinate system moves with the platform. Its origin is where the platform's natural axis of rotation intersects the ground plane. The positive Z axis points forward, the positive X axis points to the right, and the positive Y axis points up from the ground. The 'global' coordinate system is fixed, with its origin located at some convenient point (a fixed latitude and longitude). In the global coordinate system, the positive X axis points north, the positive Y axis points east, and the positive Z axis points into the ground. Classes were developed that allow straightforward mapping between latitude/longitude and the global coordinate system, as well as between global and local coordinate systems.

Device Interfaces

Device-specific classes were developed. These classes interact with the devices over various communications interfaces, and convert between device-specific values and engineering units. All of our device data is managed through a basic wrapper class which provided the appropriate methods to interact with each device. Each device interface runs on its own individual thread allowing the device to be automatically restarted if an error occurs while not interfering with any of the other modules.

Vision Processing & Obstacle Detection

Vision processing relies on a stereo vision system which was developed by the Vision sub-team using Emgu, a C# wrapper for the OpenCV library. The stereo vision system requires many stages which are accomplished using Emgu functions. First, the cameras must be calibrated using a series of chessboard images. The location of the chessboard corners for the left and right camera images is found to the nearest pixel and further refined using a sub pixel search function. The locations of the chessboard corners are then used to calculate camera parameters for the left

and right cameras in addition to the transformation which relates geometric location of the two cameras. Once these calibration parameters are known the next step is to rectify the images. Rectifying the images guarantees that corresponding pixels in the left and right images will be in the same vertical pixel row. An example rectified image can be seen in Figure 5. Once the images are rectified, a semi global block matching algorithm is used to find the location difference between corresponding pixels in the left and right rectified images. An example disparity image is shown in Figure 5. The three dimensional coordinates can be determined using the disparity image and Emgu's ReprojectImageTo3D function. This function relies on the following matrix

$$Q = \begin{bmatrix} 1 & 0 & 0 & -C_x \\ 0 & 1 & 0 & -C_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -1/T_x & (C_x - C_x')/T_x \end{bmatrix}$$

where (*Cx*, Cy) is the principle pixel in the left rectified image, *f* is the focal length of the left camera in pixels, *C'x* is the *x* coordinate of the principle pixel in the right camera and *Tx* is the spacing between the two cameras. The value of the *Q* matrix is obtained during the calibration process; however, some of the values must be modified in order for the function to work. The disparity image actually returns the difference in pixels multiplied by 16 so the -1/*Tx* term must be divided by 16. Additionally, the 1/*Tx* term has the wrong sign which results in large nonlinear errors in the resulting 3D coordinates. Therefore the -1/*Tx* term is multiplied by -1/16 to obtain the correct results from the ReprojectImageTo3D function. Using these settings the depth measurement provided by this function is accurate to roughly 1/10 inch from 2 to 10 feet away from the stereo cameras.

The vision system distinguishes between three types of obstacles. During obstacle detection the 3D camera coordinates are rotated so that the xz plane is parallel to the ground. This is done by measuring the height of the cameras and the distance to the principle point on the left rectified image. For each obstacle type the vision system provides a list of rotated (x, z) points where obstacles are present.

In total, there will be three sets of stereo vision, one located on the back top side, and two located in both frontal left hand side and frontal right hand side, proportioning a more reliable and elaborative vision system control in which obstacles can be detected in a precise way by activating the side cameras from each respective side when the vehicle has to change.

Obstacle Detection

Obstructions are detected by finding all points where the rotated y distance from the plane is greater than 10 inches. For each pixel in the disparity image, 3D coordinates are found and rotated. The (x, z) coordinates are then passed to the local path planning.

Surface Feature Detection

Line detection is done by filtering the HSV components of the images obtained by the cameras. By adjusting the range of Hue, Saturation and Value (perceived luminance in relation to the saturation) of the pixels, it is possible to isolate both the white line in the grass, the red and the blue flags.

After filtering, three black and white images are obtained, the white dots correspond either to the lines or to the flags, according to the analyzed image. The 3D coordinates for each pixel are

found by considering the floor a plane surface, and converting the pixel coordinates into the car coordinates.

To make this correlation, the height of the camera, its angle and its field of view were measured. Pixels in the superior left camera identified as lines can be seen in Figure 6. Flags are detected by changing the filters to the desired colors, and by changing the height of the analyzed plan during the calculations to the flags height.

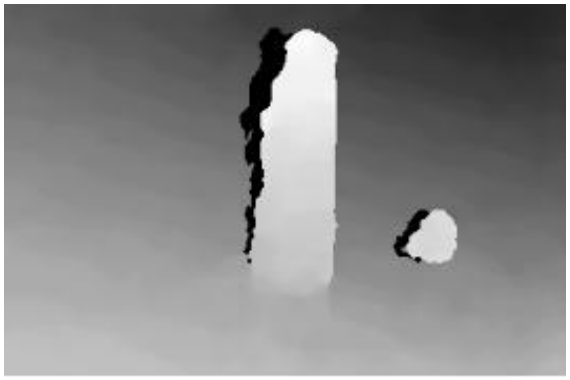


Figure 4. Disparity Image

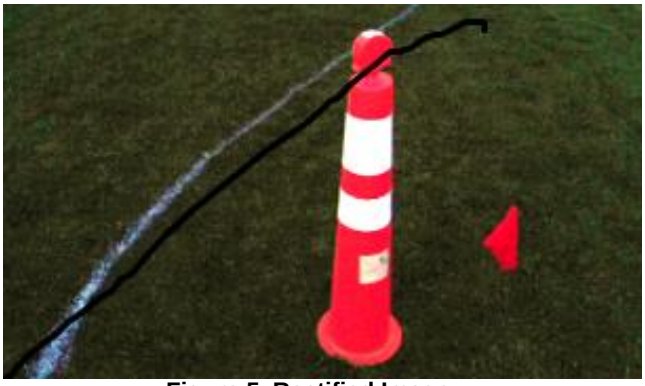


Figure 5. Rectified Image

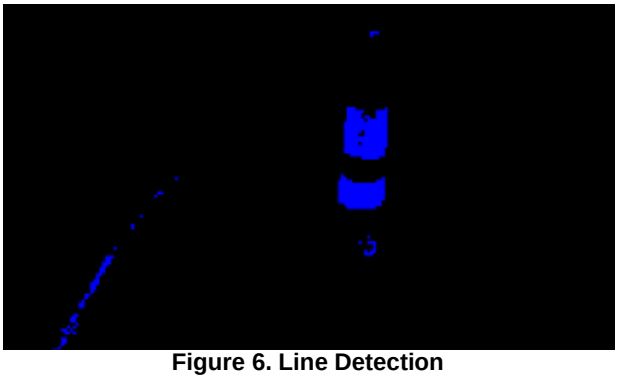


Figure 6. Line Detection

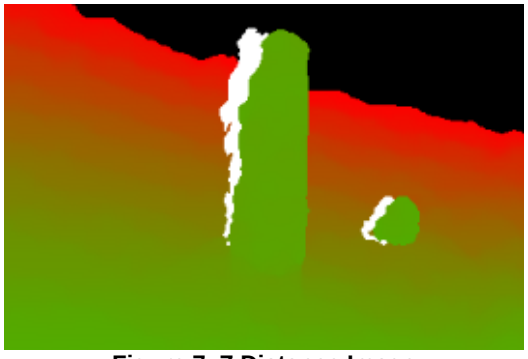


Figure 7. Z Distance Image

Local Path Planning

The local planning system receives (x, z) coordinates for obstructions, lines, and flags from the vision system. The local path planner tests a set of rectangles to see if they contain obstacles. Each rectangle starts at the front of the robot and extends about 5 meters ahead. The rectangles are placed at five different angles relative to the robot's heading (straight, hard left, soft left, hard right, and soft right). The rectangle containing no obstacles with a heading which is closest to the next GPS waypoint is chosen as the desired driving direction. In the case where all rectangles contain obstacles, the robot will turn in place until an obstacle free rectangle appears.

IOP

For this year's competition, the IOP component was based on the JAUS component from the previous year. The component has 3 major subcomponent groups: the JUDP Message Handler, the System Topology Definition, and Services. Each of the subcomponent groups is abstracted from the others so they are easily configurable and expandable for future competitions. Interfaces flow in a "trickle down" way so each group has access to only the necessary features from the level above it.

The JUDP Message Handler is concentrated around sending, receiving, and validating IOP messages from a UDP Packet. The JUDP Message Handler has 3 main features: a connectionless UDP socket, prioritized transmit and receive queues, and an address registry. Mailboxes are indexed by IOP IDs and are layered by priority.

The address registry contains a dynamic list of the available entities on the system. When valid messages are received, the source IP Address and Port of the message are tracked and placed in the registry, indexed by Source IOP ID. The address registry only tracks the IP and Port for messages on the first receipt from a new Source IOP ID. If the IP and Port are known for an entity on the system prior to receipt of a message from that system, the address can be registered manually. Additionally, any address in the registry may be removed and updated.

The prioritized queues allow a non-blocking interface for other subcomponent groups in the IOP component to send and receive messages. Each mailbox is layered by priority, so that higher priority messages are always handled first, regardless of when they were received / transmitted. Before entering either queue, messages are checked for validity. Any non-valid message is not queued. To transmit messages, a subcomponent only needs to supply a valid IOP Message with the appropriate Destination IOP ID. The ID will be queried against the address registry, and if an address if found, transmission attempts begin. Messages will be retransmitted either until they are transmitted successfully, or until a (configurable) maximum number of attempts is reached. To get a message from the receive queue, subcomponents must supply a valid IOP ID. This id corresponds to the Destination IOP ID of the received message.

The System Topology Definition subcomponent is responsible for defining IOP Subsystems on the IOP System. There are 3 major components used to achieve this goal: IOP Subsystem Definitions, IOP Node Definitions, and IOP Component Definitions. These definitions correlate tightly with the definitions found in SAE AS5710. A generic topology definition can be visualized by Figure 6 (page 16) of SAE AS5710.

The component definition consists of a (node-wise) unique unsigned byte id and a list of IOP Services (described later). The node definition consists of a (subsystem-wise) unique unsigned byte id and a list of component definitions. Finally, subsystem definitions consist of a (system-wide) unique unsigned short integer id and a list of node definitions. A subclass of the subsystem definition type is created for each subsystem in the IOP Component. Each such subclass is responsible for defining its nodes and components and the relation between them. The subclass is also responsible for defining any interfaces needed by services its components may contain. Lastly, the subsystem definition must also define how the subsystem is to behave, such as: when and how the receive queue is queried; when the received messages are passed to nodes (and subsequently components and services); and any other special tasks that need to be done.

Similar to the System Topology Definition subcomponent, the Service Definition subcomponent is modeled closely after definitions found in SAE AS5710 as well as SAE AS6009. Services define a set of related functionality in the entity. All services must contain a URN, a major version, and a minor version. Each service also contains a list of IOP Message handlers, which are indexed by IOP Message IDs. When a message is passed into a service, the Service interprets the message, performs necessary actions within the entity, and sends a response (if required). Unlike the System Topology Definition subcomponent, however, a service definition will remain the same from entity to entity. Instances of each desired service definition are created and placed in a component definition. To interact with the entity, as well as other services, interfaces are passed to each service as needed.

It is with these service definitions that events are created to handle status changes. The robot can be in several various stages including: Ready, Standby, and Shutdown. When a state change is detected (whether internally or from a message), an event is raised that calls various function pointers that have been registered.

IOP Simulation and Test

As with any component in a system, there needs to be a way to test it against its requirements. IOP is no different. Similar to past years, a COP was created to test our IOP implementation. However, instead of implementing the COP on top of the IOP code in C#, it was abstracted to Javascript / HTML5 for the Interface. Because HTML5 only allows client to client side communication in the form of WebSockets (RFC 6455), which are built on top of TCP, a WebSocket Pass-through module was created in Python. The COP was designed outside of the competition IOP system to allow for reusability for future years and flexibility in system setup. The COP uses the Google Maps API to find coordinates. Figure 8 below is a screen shot of the COP.

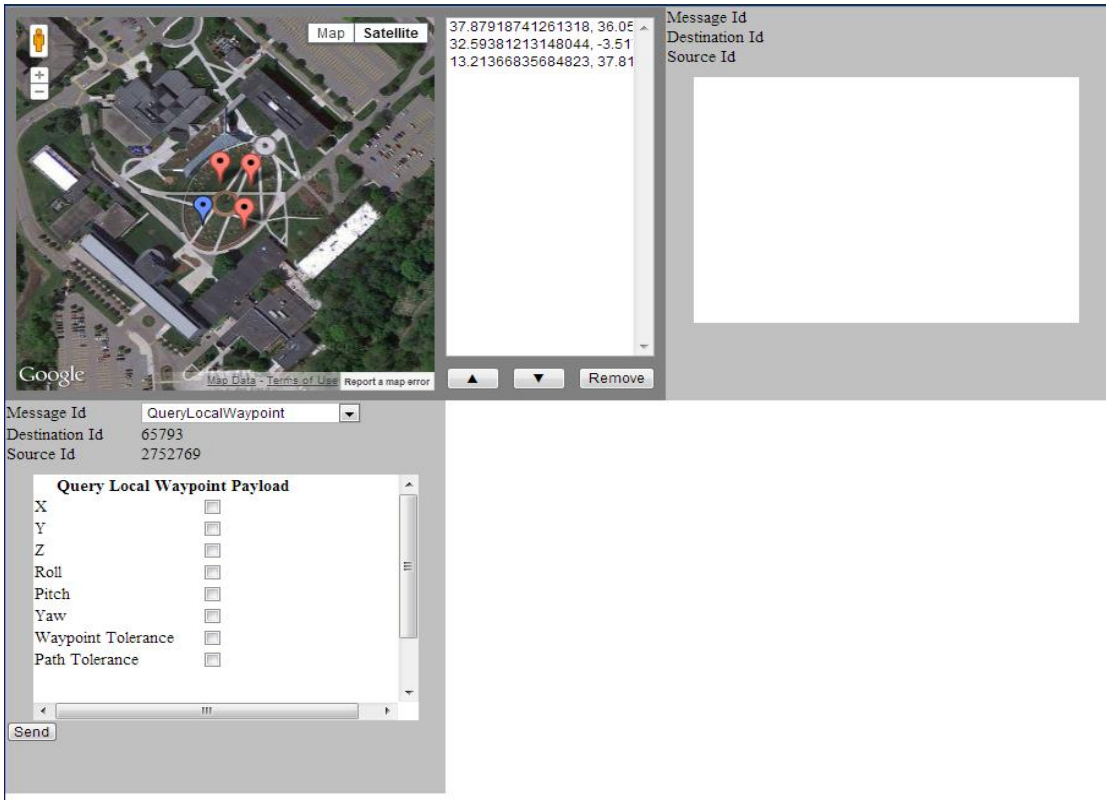


Figure 8. COP User Interface

PREDICTED PERFORMANCE

The following points describe predicted performance and the methods used to determine these numbers.

Speed – The top speed of the robot is governed by the maximum RPM of the robots brushed DC motors. This top speed is further reduced by the integrated gear reduction which increases the effective torque and provides greater mobility on soft surfaces like grass. The speed is controlled by an integrated motor controller which receives commands from the PC. Top speed is approximately eight miles per hour.

Ramp climbing – Propulsion system design and component selection were undertaken with a goal of performing at top speed on a 15% gradient, the specified maximum under IGVC rules. Performance to this goal has been verified in trials.

Reaction times – The vision system achieves a sustained throughput of 15 frames per second for each camera. Based on an analysis of latency in acquisition, processing, and communications paths, it is estimated that an obstacle presented within the field of effect will affect motor speed in 75 +/- 25 milliseconds.

Battery Life – Battery life is highly dependent upon the operational environment. Under continuous load and with a full charge, the 24V battery module life is estimated at 2 hours. The 12V battery life is estimated at 5 hours under full processing and sensor load.

Obstacle Detection Distance – This is configurable via parameters to the stereo vision processing software. Detection is presently limited to 6 meters.

Complex Obstacle Negotiation – Switchbacks and traps are handled as a natural consequence of the path planning algorithm, described in the software design section.

Navigational Accuracy – The geolocation equipment used is capable of sub-meter accuracy when used with satellite- or earth-based augmentation. The GPS sensor is capable of employing satellite-based augmentation and is presently configured to use the OmniSTAR service for differential corrections, which after initial settling will generally achieves a standard deviation of 0.3 meters or less from actual.

Table 1. Cost Data (Dollars)

Mechanical / Propulsion		Sensors		Processing / Electrical	
Invacare wheelchair	Free	Web Cameras	120	Computer Systems	1500
Batteries	340	GPS System	2,700	Power Supplies	150
Base-Mount Drawer Slide	70	Compass with Tilt Compensation	Donated	E-Stop System	40
Miscellaneous Hardware	300			Safety Light	40
				Misc. Electrical	200
				TOTAL	$5,609

Table 2. Labor Data (Man Hours)

Mechanical		Electrical		Software	
Design	2	Design	10	Device Interfaces	10
Fabrication	15	Component Selection	5	IOS-Specific	350
Assembly	15	Integration	40	Auto-Navigation	40
				Integration & Testing	40

Sub Total	32	Sub Total	55	Sub Total	440
				TOTAL	527