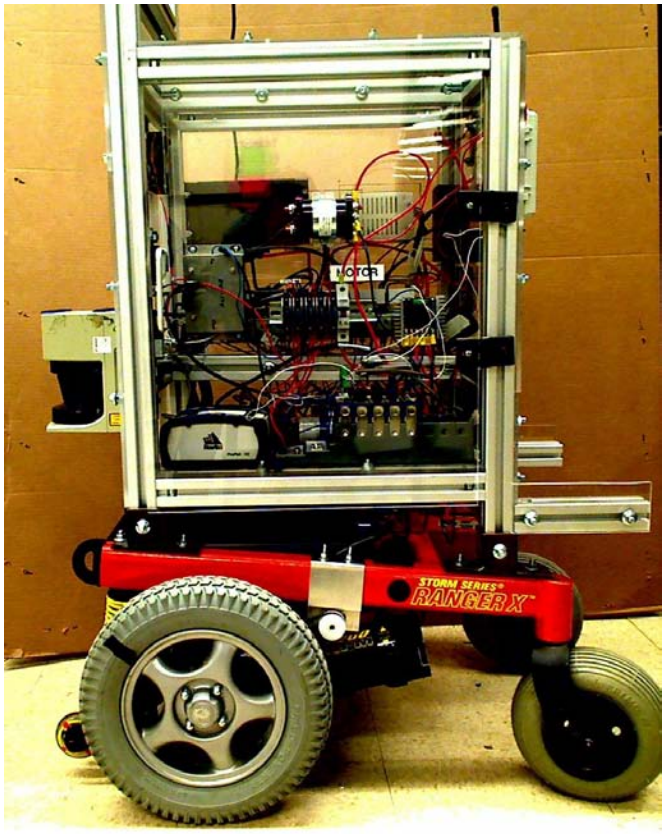
CASE WESTERN RESERVE UNIVERSITY



HARLIE

I, certify that the engineering design in the vehicle (original or changes) by the current student team has been significant and equivalent to what might be awarded credit in a senior design course.

_____

Signature Date

1. Introduction

Harlie, one of the two entries from Case Western Reserve University, is an IGVC veteran. We used Harlie in the competition last year. However, this year he is equipped with a completely redesigned software architecture as well as some new hardware components. We have also redesigned several of the emergency/safety components.

1.2 Harlie Overview

Harlie uses a National Instruments cRIO controller based system implemented on the drive train of a motorized wheel chair from Invacare Corp. The cRIO is used for sensory data acquisition from wheel/motor encoders, gyros, GPS, LIDAR, and video. It will also be used for motor control and for localization computations. Other algorithms are implemented on an onboard MAC Mini, which serves as a host and interfaces with the cRIO. Figure 1 and Figure 2 provide a functional and visual overview of the hardware.

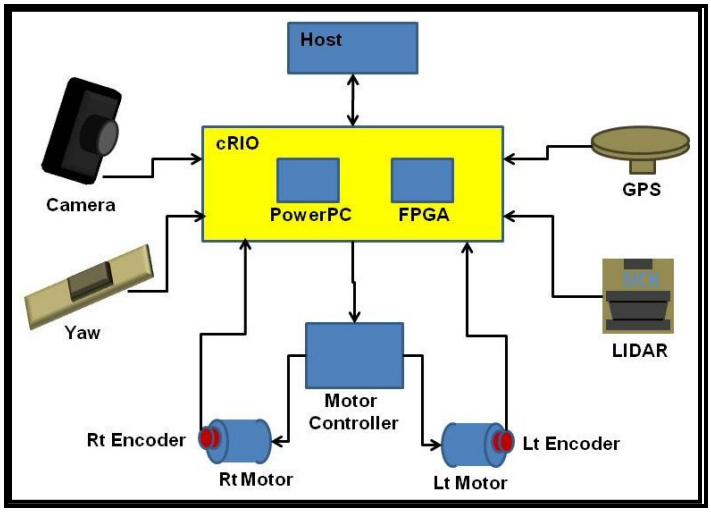


Figure 1: Block Diagram of Harlie.

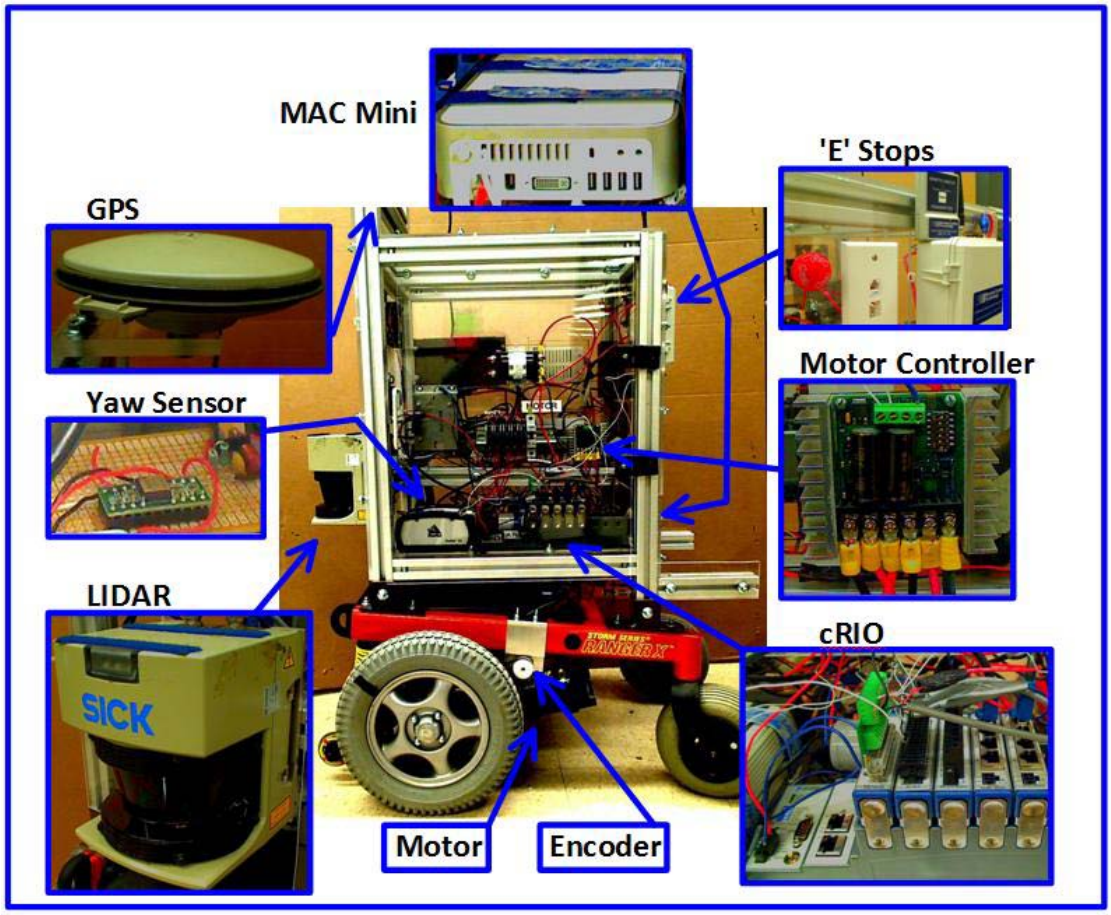


Figure 2: Visual Overview of Harlie.

2. Hardware

2.1 Drive Train

The base is from the Ranger X model wheel chair from Invacare's Storm series wheel chairs. The wheelchair base provided our team with a rugged base equipped with suspension and motors. The suspension system will help to stabilize the upper frame unit to reduce noise in sensor values, which is a desirable quality for an outdoor robot. Also, since it is a wheelchair base designed for patient care, the base and motors are

Figure 3: Wheel chair base provided by Invacare.

manufactured so that there is little if any room for failure. The motors provided by the manufacturer are also limited to a 5 mph max speed, which fits our needs perfectly.

2.2 Frame

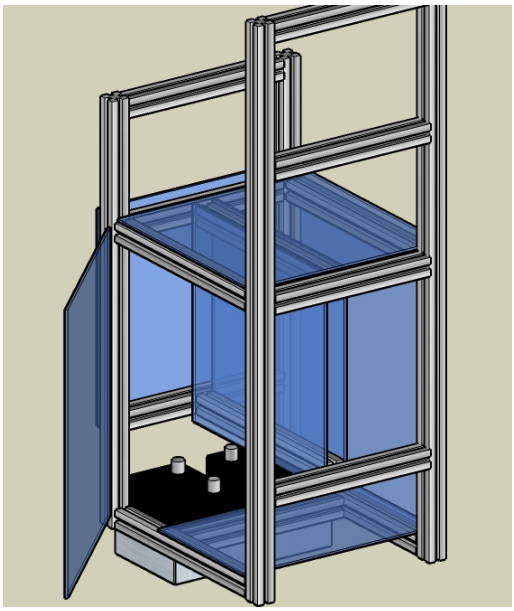


Figure 3: 3D model of frame design.

The frame was designed and built using 4 cm square aluminum framing. The frame was built to support one horizontally mounted panel and two panels vertically mounted in opposite directions. The design was chosen to maximize accessibility and surface area. Designing a method for mounting the frame to our base was one challenge our team had to overcome. As manufactured, the base was sloped down towards the casters. Using a simple mount would cause our frame to also lean at this angle. To overcome this problem our team developed custom mounts that would allow us to level the frame on top of the sloped base. Having a level frame made mounting the LIDARs easier since they were required to be mounted parallel with the ground.

2.3 Electrical System

Harlie's components have many power needs. The LIDARs require a reliable and clean 24V with a low current draw to operate quickly, but the motors require a high current at 24V. The Mac Mini computers require a special 18.5V and other sensors, such as the GPS, require only 12V. To provide this power, the wheelchair base also came with two easily replaceable batteries that are connected in series to give 24V. The base also provided an external port for charging the batteries, making recharging easy.

To generate the other power required for the robot, we used a Carnetix CNX-2140 DC-DC converter to generate 18.5V for the Mac Minis, a Samlex 24-12 DC-DC converter to generate 12V, and a custom-made 24V-24V regulator that produced regulated clean power for the LIDARs. To distribute the power to multiple components, each voltage was fed to a color coded

bus bar on separate panels. In our convention, red signifies 24V, blue indicates 12V, and black is used for all grounds. By keeping wiring and bus bars to this coding scheme, our team was able to avoid the costly error of providing sensors with too much power.

For surge protection, our team used a number of different thermal circuit breakers. We used one 120A main breaker that is also used as a main power switch. We also used two 63A breakers, each supplying power to different wheels. For individual electronics, we used small 10A breakers that integrate with the style of bus bar that we chose. A diagram of the power system is shown in Figure 5: Circuit diagram for the electrical system.

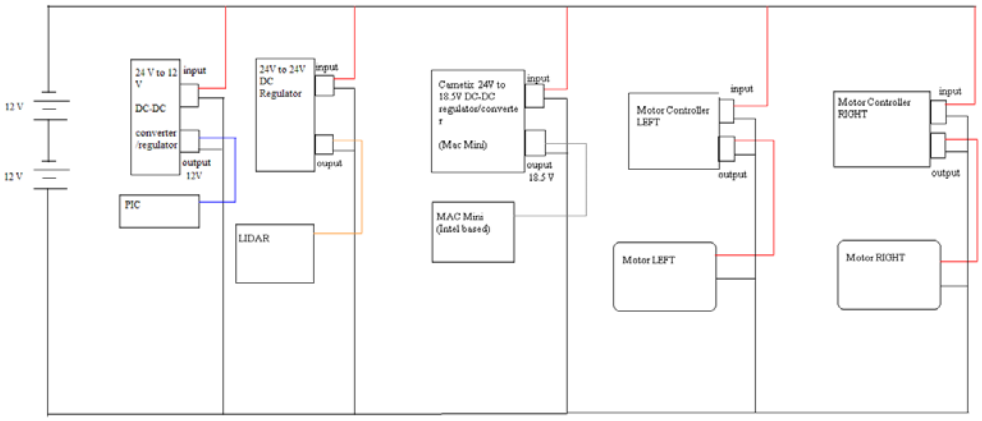


Figure 4: Circuit diagram for the electrical system

The Carnetix and Samlex power supplies were both purchased items that would give us the desired voltage and had reasonable power capabilities. The 24V-24V regulator, however, was custom-assembled by our team. At the heart of the regulator is a store bought regulator that has a range of outputs. Also, since the regulator used is a switching regulator, we needed to filter the output to make sure it was clean power.

3. Sensors

For physical state observation, obstacle avoidance and mapping we used a multitude of sensors. They include two Firefly MV cameras, a SICK LIDAR, two wheel encoders, two motor encoders, a GPS receiver, and a yaw sensor.

3.1 Cameras

Figure 1: Firefly MV

The Firefly MV from Point Grey Research was used to capture the images. This camera, shown in Figure 5, uses a 1/3" wide-VGA CMOS sensor from Micron with a 752 x 480 resolution and a 1394a digital interface. A Senko lens was secured to the camera and adjusted to maximize the depth-of-field and provided for the widest angle. This wide angle lens results in image distortion that must be corrected and this is addressed in a later section.

National Instruments' Vision Assistant v8.6 was used to capture and manipulate the image from the Firefly MV. Vision Assistant provides a simple graphical interface to facilitate preparing filters and other algorithms to manipulate the captured images to extract the key information.

3.2 LIDAR

Figure 6: SICK LiDAR

Harlie avoids obstacles by the use of SICK Light Detection And Ranging (LiDAR) (See Fig.1). SICK is a LiDAR unit that is an accurate laser rangefinder. It uses a rotating mirror to emit an infrared beam within a viewing angle of 180 degrees sweep. With the return of the reflected beam to the LiDAR, the range is then evaluated. SICK has the capability of working in the range of 10 cm to 80 m and can provide with 500kbaud. Also, it is an active device as it operates without an external illumination.

3.3 Encoders

To aid in position localization, we use one quadrature wheel encoder for each wheel. Quadrature encoders have two output pulse trains that are out of phase with one leading or lagging the other depending on the direction of motion. The signals produced are then integrated on an NI-DAQ 6211 board in real time without the intervention of the main computer's CPU.

3.4 GPS Receiver

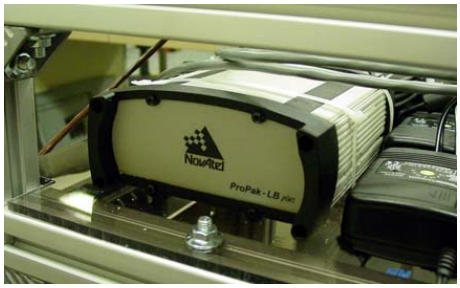


Figure 2: NovaTel DGPS

For determining our global position we used a NovaTel ProPak LB-Plus Differential GPS. We used Canadian differential signals with corrections from Omnistart. When we have a clear view of the satellites, the GPS is reported to be accurate within 10cm. Actual test data shows that the estimate is only true in open fields on a perfectly clear day. The GPS signal can be thrown off by nearby buildings, trees, puddles, and clouds. Some of these effects can actually bias the signal in a single direction which violates many assumptions made by the Kalman filter used and can cause serious problems when trying to accurately determine position.

3.5 Yaw Sensor

The yaw sensor on Harlie is an ADXRS 150 angular rate sensor. The breakout board that is on Harlie contains a setup that runs the sensor at a rate of 40Hz. There are two outputs from the ADXRS sensor, a constant 2.5 volt signal and a rate output that is represented by 12.5 mV/degree/second signal. The rate output provides the velocity of the rotation. So integrating the rate output signal will provide one with the angle. If the robot is not turning the rate output should be 2.5 volts. The current code takes the rate output and subtracts it from the 2.5 constant voltage signal. This is then multiplied by a conversion factor that is based on time and the 12.5 mV to get the change in degrees, which in turn is added to the previous value for degrees.

4. Software

The majority of our programming was done in LabVIEW and C#. LabVIEW offered real time determinacy and seamless integrations with the cRIO. However, some applications were easier much more efficiently coded in C#, such as the steering and planning algorithms. However, whenever a high level of determinacy was needed, LabVIEW was used to create FPGA code that was determinate up to 40 MHz.

4.1 PID Control

The controller we use to monitor and change our robot's velocity consists of three parts: proportional, integral, and derivative control. These different methods of control each have their own benefits and drawbacks. Proportional control helps speed up the process of reaching the desired value and will likely be the biggest proponent of the output change. If the proportional gain is too high, the resulting value will oscillate endlessly around its target value and never settle on a value. If this gain is too low, the system will not be able to effectively adjust to disturbances. Also, the proportional term alone will always have some steady-state error. This problem will be solved by the Integral control, which is designed to eliminate steady-state error and close on the target value. The drawback on the Integral term is that it increases overshoot. The Derivative control helps dampen the negative effect of the Integral term by decreasing overshoot. However, this method slows down the response of the system and may cause instabilities at higher gains. In the PID controller that we have implemented, all of these control methods are combined in an effort to increase system response time and decrease overshoot and steady-state error. In addition to a basic PID controller, we have included several extra functions for safety, convenience, and necessity that will be explained.

One of the functions we added to our PID was a slew rate limiter. This limits the amount of change that the PWM signal can undergo every loop. By checking the limits of the PWM change as set by the "R Slew Limit" control, we can limit how quickly the PWM ramps up or down. This effectively acts as an acceleration limiter for the robot and is helpful as an additive method of safety control.

One of the more important functions added to the PID was the ability the push a button (in this case the E-Stop) to flush the PID of all its current values and reinitialize it. This is especially important when dealing with integral control because if the motors are turned off and the PID is still running, the integral component will keep ramping up the PWM to reach the desired speed. The result is that your PID will command a PWM that corresponds to the max allowable value and when you turn E-Stop off and close the circuit, the robot will take off at max speed immediately. This can be very dangerous so this function is essential.

The group used National Instrument's Discrete PID (niFPGA Discrete PID.vi). It is well annotated and deals only with integers since it is housed on the FPGA. Being on the FPGA not only takes some of the processor stress off the Mac Mini in Harlie, but it allows functions like the PID to run very fast (10 ms) without sucking up excess computer resources so that other more intensive processes like Vision and PSO can be housed on the Mac Mini. This faster run speed makes the measurement of the current velocity for velocity error calculations much more accurate, thus improving the performance of the PID.

Before the new commanded PWM leaves the PID and is implemented on the motor, it goes through one last safety function. This just limits the max and minimum values of the commanded PWM. We set the limits to + or – 5000 PWM, which corresponds to a maximum robots speed of + or - ~0.8 meters per second. This is yet another safety precaution as there is really no reason for the robot to be going faster than 0.8 meters per second on any wheel in either direction.

4.2 Steering and Velocity Control

To implement an effective steering system on our robot, we chose to use a Wagon-Handle Algorithm to determine the robot's speed and heading given a particular desired path. The Wagon-Handle algorithm works by first creating a line from a starting position to a desired position, which is obtained using information from the physical state observer. The algorithm, in essence, draws a circle around the robot and calculates the two points on the circle where the line given by the PSO crosses that circle. Of these two points, the point closest to the desired waypoint is chosen and the angle between the chosen point and the y-axis of the robot is used to calculate the desired heading needed so that the robot will head towards that point. Here is an illustration of how this works:

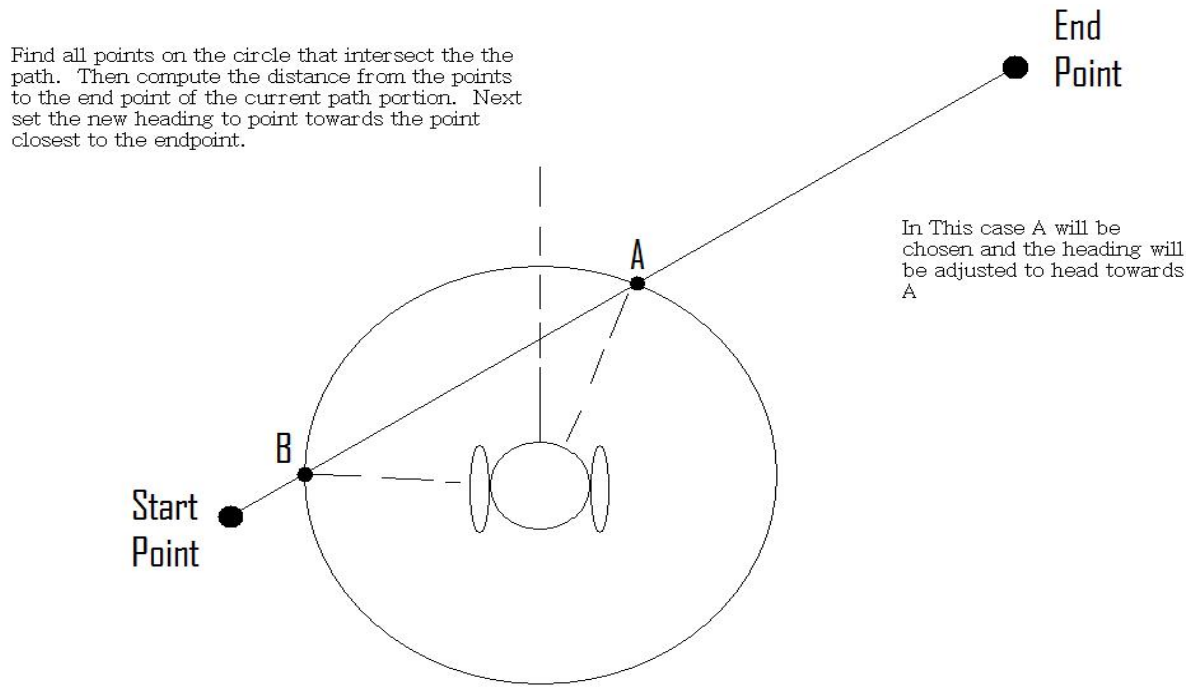


Figure 7: Basic theory behind wagon handle steering

We wrote an effective Wagon-Handle algorithm in C#. The code outputs a desired speed and heading. As a safety precaution, another program is designed to command a speed and heading of zero if it does not receive a packet of information from the Wagon-Handle Algorithm every 500ms. This is to ensure that the robot does not continue on an undesirable direction if something goes wrong. If the information from the Wagon-Handle algorithm comes through, a case structure in the program is set to "false" and the values provided by the algorithm pass through the safety net to a program that determines individual wheel speeds.

After the Wagon-handle algorithm determines what robot's speed and heading should be to continue along the predetermined path, the next step is figuring out what the individual wheel speeds need to be to obtain the proper speed and headings dictated by the wagon handle algorithm. In the steering code, the current and commanded headings are put through a simple algorithm to obtain the quickest direction to turn to achieve the desired heading. This is to ensure that the robot will not spin all the way around clockwise if it only wants to go one degree counterclockwise. As an additional security precaution, if the Lidar senses an object in the way of the robot, the commanded speed is set to zero.

After the correct heading error is determined, it is put through a simple PID structure of its own. This extra PID for the robot's heading was implemented to increase the response of the steering and eliminate steady-state error that we found caused the robot to drift when it was supposed to drive straight. The proportional, integral, and derivative gains allow the user to customize the type of response they get from the steering. We found that this structure with effective gains removed a lot of the drift and heading overshoot from the turning response while still enabling the robot to turn quickly and accurately. Once the heading error is put through the steering PID, the new heading correction is added and subtracted from the commanded speed to find the individual speeds for the left and right wheel.

After acceptable individual wheel speeds are determined, they are converted into a PWM value that corresponds to the desired speed in meters per second and sent to the PID and speed controls on the cRIO.

4.3 Planning

Harlie's planning is based on two very simple methods. First we create a 2-D array map of Harlie's world. The map is then optimized to find the ideal path to Harlie's goal and breadcrumbs are placed along the path. Second, we use the previously described wagon handle steering algorithm to find these breadcrumbs and to keep Harlie moving along the path in the desired direction.

A simple map can be made out of a comma separate value (csv) file. The area that Harlie is to navigate is broken down into a basic grid system. Every space on the grid is encoded into the file as an "X" or an "O." This csv is then imported into the code and each value of the array is stepped through and turned into a node. A node allows us to store the different attributes of the grid and allows us to run different methods on each space. A node is marked either as clear (O) or as an obstacle (X.) Once all the nodes have been created we can run a breadth first search that assigns a cost to each node based on its distance from the goal. We use this cost to calculate and plan the optimal route for Harlie to traverse the map and reach his destination.

After the optimal route to the goal is found, we can place breadcrumbs along this path. Harlie will then attempt to follow these bread crumbs as consistently as possible, while avoiding obstacles and other obstructions that are not encoded into the map. An illustration of these concepts is summarized in the following figures.

2	2	2	2	2
2	1	1	1	2
2	1	Start	1	2
2	1	1	1	2
2	2	2	2	2

Figure 8: Perform a "bread first search" to determine the distances to the goal.

	5	1	6	
	4	Start	3	
	7	2	8	

Figure 9: The BFS is performed in this order.

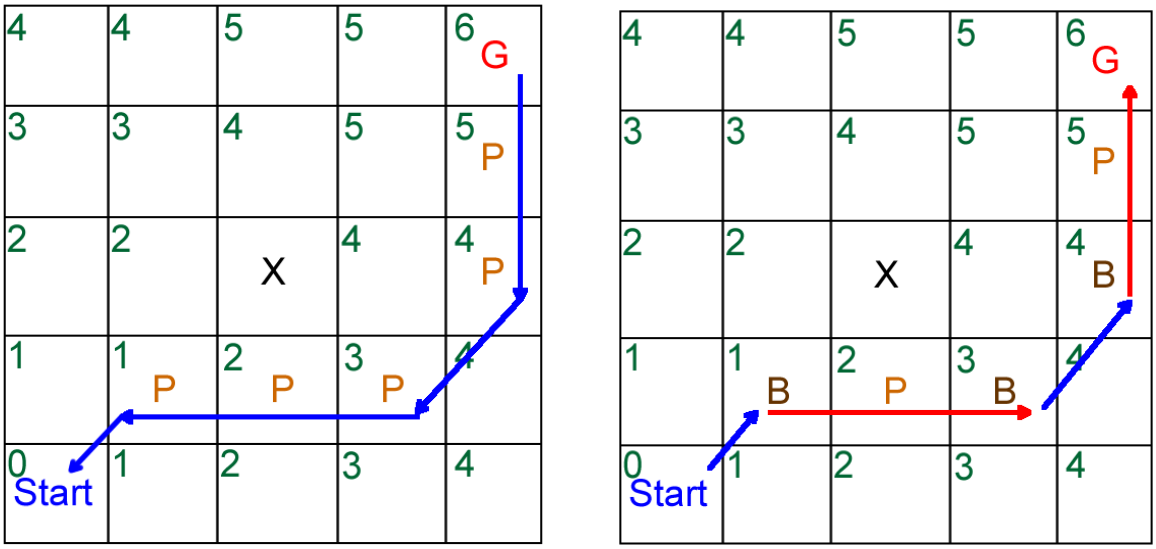


Figure 10: Left: these distances are reversed (by subtracting one each time) from the goal to the start, in the same order of preference, to provide an initial path. Right: This path is traversed and a breadcrumb is issued every time the path changes directions (ie, from north-east to east, etc.)

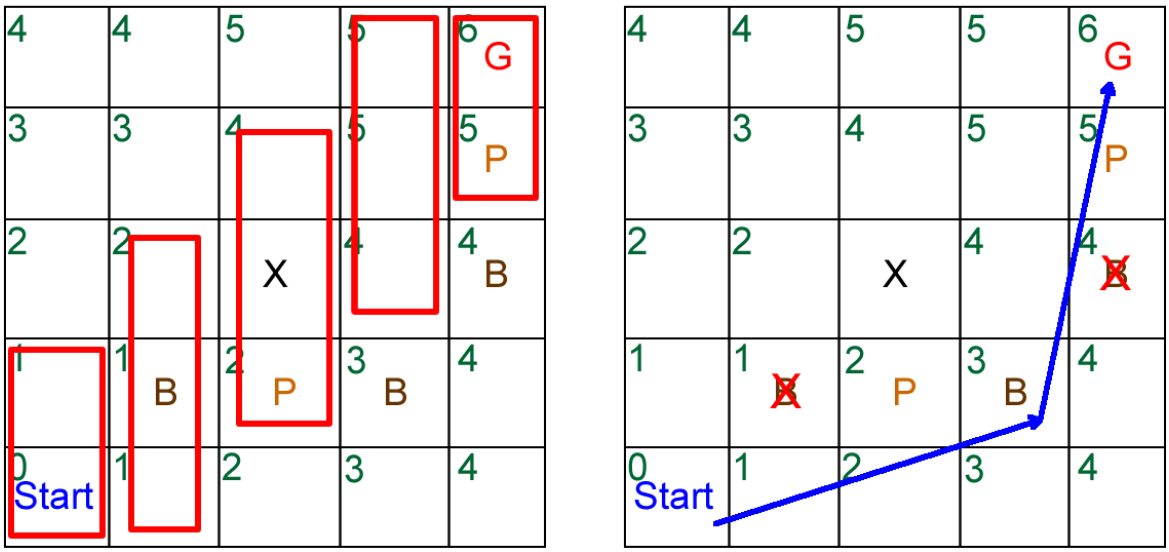


Figure 11: Left: Now these breadcrumbs are examined to see if there are any straight line paths between any two. This is done in the following order: Start to Goal, Start to Goal-1, Start to Goal-2, …, Start + 1 to Goal, Start + 1 to Goal – 1… This algorithm examines a sliding window of configurable length (in this case the length is 3) that expands from the original breadcrumb in question to its maximum size, and then contracts and maintains centered as it nears the final breadcrumb in question. Right: After all of the breadcrumbs are eliminated, the path looks like this, which is always is a most (or tied for most) efficient path.

4.4 Physical State Observer

Physical state is one of the most difficult problems in robotics. A good estimation of the position of the robot is vital to almost every other higher level system on the robot. We use a GPS solution that on a clear day is accurate within 10cm. However due to atmospheric effects, water on the ground, and nearby buildings, GPS is not always very accurate.

Kinematic modeling and state estimation can be added to pure GPS to provide some sort of feedback such that the state is reasonable for a longer period of time. Wheel encoders can be used to measure the kinematic movement of the system. In general, the best kinematic models under ideal conditions of minimal wheel slip are only accurate between 5% and 10% of the total

distance traveled. Kinematic estimation is still considered open loop – but is much better than dead reckoning alone.

Since a variety of sensors are available for use in state estimation and each sensor has its own downfall, the Kalman Filter can be used to probabilistically determine the most accurate state estimate. Moreover, the Kalman filter will estimate the state recursively and iteratively in real time – constantly driving the uncertainty of the solution downward. The Kalman Filter takes inputs from wheel encoders, GPS, and laser odometery allowing us to determine our position within 20-30 centimeters. With proper analysis and tuning of the Kalman Filter gains, we hope to get our estimation down to within 10 centimeters.

5. Expected Behaviors

5.1 Speed

The speed of the wheel chair motors as defined by the manufacturer is only up to 5mph. To make sure that our controllers do not exceed the limit, we verified the restriction. We also implemented several safety precautions in both hardware and software that disable the motors if a speed threshold is exceeded.

5.2 Battery Life

The typical battery life for the robot with all of its sensors running is around two hours. This duration should be long enough for each heat of the competition.

5.3 Complex Obstacles

Using the planning algorithm in conjunction with our obstacle avoidance systems, the robot will get to its goal if a path exists. We do not anticipate that complex objects such as switch-backs, center islands, and dead ends will cause any problems during planning; however, testing and verification have not yet been completed.

6. Parts

Part Description	Retail Price	Cost to team
Wheelchair base	Est. $3000	$0
Bosch Aluminum Framing	$500	$500
Power Converters	$345	$345
Electrical Components	$770	$770
Victor Motor Controller	$200(x2)	$400
SICK LIDARs	~$6000	$6,000
Mac Mini Computers	$800(x2)	$1600
Wheel Encoder	$150(x4)	$600
RS422-USB Converter	$80(x2)	$160
FireFly Camera	$350(x2)	$700
Camera Lens	$150(x3)	$450
NI-DAQ 6211	$800(x2)	$0
NovaTel ProPak DGPS	$5490	$2700
Total	$24,835	$14,225

6. Case Team - Harlie

David Thorndike – Electrical Engineering and Computer Science

Eric Perko – EECS

Ben Ballard – EECS

Kyle Levine – EECS

Chad Rockey - EECS

Matthew Klein – Aerospace and Mechanical Engineering