The MCP III:
Design Report for the
2008 Intelligent Ground Vehicles Competition
91.548 Robot Design

Professor Fred G. Martin
Department of Computer Science
University of Massachusetts Lowell

Prepared by the students
of the Spring 2008 Semester graduate course
(in alphabetical order):

Puru Botla
Quoc Huynh
Robert Lamoureux
Shaun Malick
Greg Pilla
Erin Rapacki
Blake Skinner
Yan Tran
Ryan Tucker
Torbjorn Valeur

Honorary Team Members

Mark Sherman
Philip Thoren
Haiyang Zhang

Introduction

One of the most interesting parts of the Robotics field, is its ability to bring people with entirely different backgrounds, and skill sets together to work on a common problem. Though our class consisted mostly of Software Engineers, several people who had rarely done anything other than write code found themselves getting their hands greasy while attaching chains to the robot's wheels, and learning how to wire various pieces of equiptment. This is because robots challenge designers to marry the fields of mechanical engineering, electrical engineering, and software development in order to develop a reliable robot system. Any lapse in these three areas would cause an undesirable performance from the system. As a result, this class, and competition, not only caused us each to apply our skills in new, and different ways, it also gave us the opportunity to experience something completely new.

The MCP that will be competing in the IGVC this year is actually the third version of this robot. Last year, the MCPII competed in both the autonomous, and navigation challenges, with mixed results. The challenge for this year's students was to analyze the results of the last competition, discover where the previous robot's strengths, and weaknesses were, and to create a robot that would perform better.

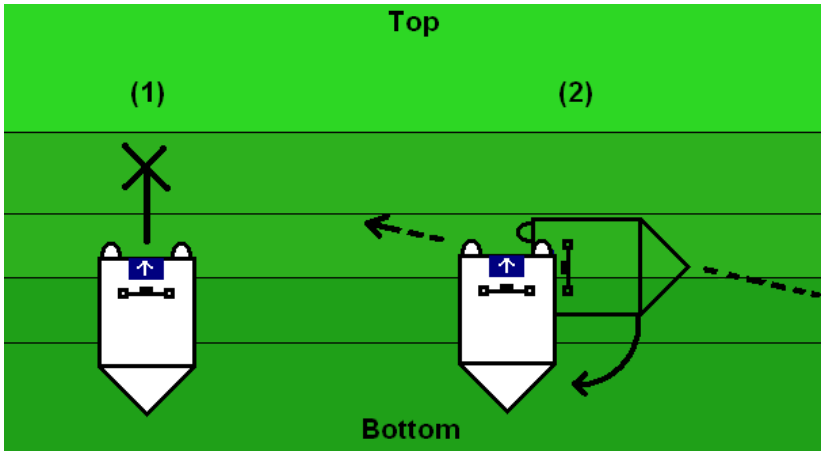
In the case of the Navigation Challenge, this was going to be a difficult task. The team of 2007 scored 6th place, reaching 4 out of the 8 waypoints, in 5 minutes and 32 seconds. However, this robot was hindered by a few things that would make improving on that score difficult, without some major modifications. On the other hand, in the Autonomous Challenge, the MCPII placed 21st, travelling only a few feet before it encountered a hole in the course, and strayed out of the boundaries. As a result of these two performances, it was obvious that a different approach was going to be needed for the 2008 IGVC, so the MCPII was recycled, and the MCPIII was born.

Design Problems with the MCP II

The design of the MCP III began with a critical assessment of the previous year’s robot entry, the MCP II. The MCP II qualified in the 2007 IGVC event, but was impeded by flaws that were revealed on the field. Since the MCP II was most successful in the navigation challenge, the problems that caused performance loss in that event were of primary concern. However, the properties of the design that allowed success in that challenge were also important to preserve. It was determined that the primary cause of performance loss was physical/mechanical in nature although it was partnered by a particular software issue which left the MCP II more vulnerable to failure. As reference, the image below displays the structure of the MCP II.

MCP II

In the navigation challenge, the MCP II was unable to ascend the hill and therefore unable to reach the final way points. The graphic and video reference below illustrates how it failed. From the critical assessment, it was determined that there were three primary factors leading to performance loss in the navigation challenge: unstable geometry, insufficient motor power, and under-tested motor control loop gains. Situation 1 was a face-on attempt to drive straight up the hill, where the MCP II simply stopped in place. Put simply, the gearing of the hub motors did not provide enough torque to handle the full incline. To avoid this situation, the robot was designed to climb the hill at and angle, reducing the incline. Situation 2 shows the caster wheel rolling uncontrollably down the hill, forcing the MCP II back into Situation 1. The unstable geometry of the frame prevented it from operating as intended. Furthermore, the motor control program loop was designed for and tested on level ground. The gains for the Proportional-Integral-Derivative (PID) control made the programs response to the hill unsatisfactory, and it was not easily reprogrammed in the field.



The MCP II failed to climb the hill, even using different approaches.

Another drawback for the MCP II was that it was too wide to avoid obstacles easily. The minimum robot platform dimension is 36” long by 24” wide. The MCP II was 36” long but it was 30” wide so it was 6” wider than it needed to be. Accessibility to the motors, batteries, and electronics was cumbersome because the top piece of plywood needed to be lifted. Although springs were added to ease the opening of the robot, the whole robot would tip over if the tower was not propped up against a chair.

The MCP II placed well in the navigation challenge despite the shortcomings of the mechanical design. However, the robot performed poorly in the autonomous challenge where these flaws had little impact. This discrepancy was caused by the way the main control software was designed and tested for each challenge. The autonomous challenge code was designed in a classic fashion; each student would allocate time with the robot and test their code on one of the fields at the university. The inherent difficulty from using this method caused development cycles to take more time and limited testing. However, the control software that was used in the navigation challenge was developed by one student using the Player/Stage robotic simulation environment. Debugging and testing was very rapid, leading to more robust and successful code.

Goals and Requirements for MCP III

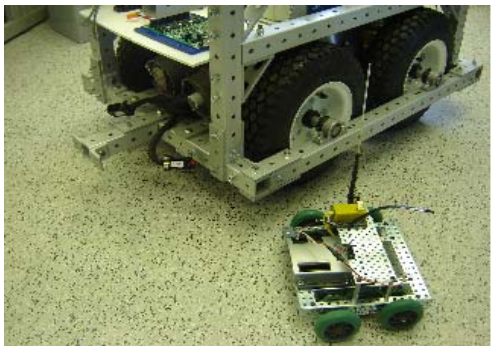
Some of designers from the previous robot design course met with current students and shared their first-hand knowledge of the performance of their robot. The apparent theme from the analysis of the MCP II was that thorough planning and testing, using as little time as possible, is vital to its success. It was determined that a new robot chassis was necessary and that it should be well prototyped before being constructed on a large scale. The electrical components were satisfactory but a detailed diagram was necessary to ensure they would have enough space in the new frame. The motors and motor controller in the MCP II were unsatisfactory and needed to be replaced. It was determined necessary to use simulation for sufficient software testing. For this reason, the MCP software was more thoroughly integrated into the Player/Stage environment. Furthermore, the Player/Stage environment was installed on multiple lab workstations and student personal computers. Other requirements were derived from meetings early in the semester to evaluate photos, reports, and knowledge of published robots and designs. The class agreed upon requirements and goals for this semester's re-design of the MCP.

- Lower the SICK laser closer to the ground.
- Make flexible & removable camera mount so we can adjust its angle.
- Adjustable camera tower so we can modify its height.
- Keep the magnetic compass away from electronics.
- Lower the overall center of gravity for the robot.
- Ventilate the electronics so they can cool faster.
- Place the camera tower in the back so it can see the front of the robot.
- Shield the SICK laser and electronics from the rain.
- Easier accessibility to the electrical panel.
- Keep the robot as narrow as possible, preferably the 24” minimum.
- Build the drive chain so the robot could turn in place.

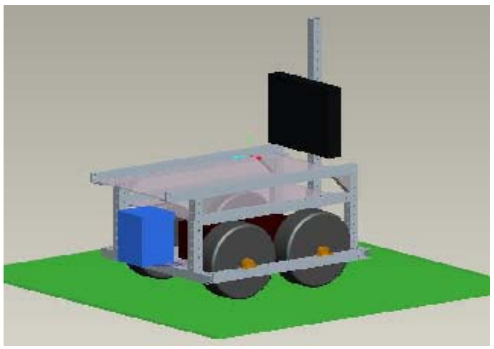
Project Innovations

The MCP III was improved upon in important ways. Although most changes or additions were not new to the field, they are innovative within the scope of the MCP Project and demonstrate improvement from previous versions of the MCP. The mechanical design of the MCP III was thoroughly prototyped using both small scale construction kits and computer aided design software. Using a VEX construction kit, it was shown that a four wheel design with a wheel base that was wider than it is long could increase robot stability without increasing the difficulty to turn in place. Sketches of possible designs for the full sized robot were made and a design was agreed upon by the class. ProEngineer CAD software allowed precise modeling of the frame and components thereby reducing errors in procurement and difficulties with fabrication. The frame was constructed from 80/20 HT

series aluminum which was chosen for its low weight and ease of assembly. Material was ordered based on the dimensions in the CAD drawings along with an over-sized length that could be resized by the mechanical engineer if changes were required. This flexibility and modularity proved invaluable in the construction process whereas it allowed for reasonable amounts of modifications or "feature creep." Below is an image of the MCP III and the VEX prototype, as well as a graphic of the ProEngineer drawing.



The VEX prototype with the MCP 3



The Pro E prototype

The motor control system of the MCP II was unsatisfactory and needed redesign or replacement. It was composed of a Dual40 motor controller and a Blackfin Handy Board which is a robotics platform used at UMass Lowell. The Handy Board ran the motor control program loop independent of the main CPU, monitored the quadrature encoders, and received serial commands from the main control code. It interfaced with the Dual40 motor controller via a daughter board constructed specifically for this purpose. This system was difficult to reconfigure in the field and often fell into a state where the Dual40 operated incorrectly. It was decided that the best option was to purchase an off-the-shelf motor controller which could perform the tasks of both the Handy Board and Dual40. The Roboteq AX3500 motor controller was chosen for the MCP III. It provides Proportional-Integral-Derivative velocity control with gains that are easier to adjust on the field and is equipped with a built in watchdog timer. Renewed confidence in the motor control system allowed greater focus in developing the main control software

Mechanical

The MCP 2 had a number of mechanical challenges. The most influential flaw is the use of a two wheeled drive system with a caster wheel. Commentary from a website for the Jerry Sanders Design Competition explains why a two wheel drive system is not ideal (http://dc.cen.uiuc.edu/guide/guide3.htm). Two wheeled robots have a tendency to tip back and fourth because only two points physically touch the ground. The MCP II had a caster wheel to allow for some stability; however, the weight may not have been equally distributed from front to back which is why it lost traction and couldn't drive straight. Castors are unstable and tend to overturn while moving and spin in place when starting. Lastly, two wheeled robots have trouble climbing ramps because of that loss of traction and thus had less pushing/driving/climbing force. It can be viewed in the subsequent youtube.com video that the robot motors stalled when on an incline. The student in the video pushed the robot with his foot in order for the robot to drive forward.

- unofficial Nav run

http://www.youtube.com/watch?v=ccmC0l5OM2A

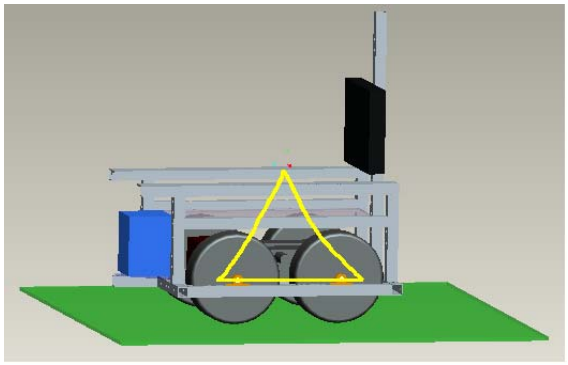
Inertia is defined as the "property of matter by which it retains its state of rest or its velocity along a straight line so long as it is not acted upon by an external force." [http://dictionary.reference.com/browse/inertia]. Inertia is the reason why a wheelchair may tip backwards if it brakes quickly or if a car has difficulty driving when it starts up a hill. The figures below show why a robot may be unstable if its center of gravity (CG) is too high.

It was suspected that the CG for the MCP II was too high. It is good technique to keep most of the weight low and centered on the robot and overall height of the MCP II chassis and battery placement may have contributed to its high inertia. Therefore, when the robot was on an incline, the inertia was acting as an opposing force to the motor's forward motion. As stated earlier, the motors were either not mechanically powerful enough to overcome this force or there may have been an electrical issue when the motors could not have all of the power they needed.

Implemented Design Changes

All of these requirements were accommodated in the MCP III design. This current robot is a four wheel drive system because it is stable, it has a lot of surface traction on the ground, and it can be controlled using two motors similar to the two wheeled robot. A four wheeled robot of the same weight should be able to out push the three wheeled robot. A wide wheel base is better than a long wheel base, as observed with the VEX robot prototype, so the wheel placement in the MCP III is a square to aid in turning ability and minimize the overall width. Wide tires were used to increase the traction and the batteries were sunk into the chassis as much as possible to maintain a favorable CG. The aim is to keep all of the heavy stuff within the triangle depicted in the figure below. This way, if the robot were to be on an incline, there won't be a large mass that hangs much in front or behind the

locations of the wheel axles. The mass would have less of an impact if kept low. Components of larger weights, which are placed in a higher location, must be centered.



Ideal placement for heaviest components in order to maintain a favorable CG.

The picture above also depicts the concept design of the MCP III. Notice that the SICK laser, the blue box, is mounted at a height of six inches instead of over a foot. This way, the SICK can better see objects that are one the plane the robot is driving on. The frame of the robot is sunk below the centers of the wheels to keep motor and battery placements low. The tower for the camera is placed in the back of the robot so it can better see the front. Lastly, the MCP III is built using levels: the lower level being for batteries/motors and a higher level for the electronics. The very top level acts as a ceiling to water proof the electronics and the laser. The cinder block payload will be placed centrally on this level.

Fabrication

The frame of the MCP II made of 80/20 HT series framing system and is 36" long by 25.5" wide. The extra width was needed in order to sink the battery below the whele hubs. The 80/20 T-slotted aluminum was not used because it is cumbersome to feed the washers into the material and large number additional holes would have to be drilled to have precise placement of components. The HT series aluminum allowed for a minimum number of holes to be drilled and changeability if the design were to be altered due to component changes. The four wheels are 12” diameter, 4” thick, wheel assemblies with inflated rubber tires and a mounted 60 tooth sprocket for a #35 chain. The wheels are mounted to the robot using steel shafts and bronze pillow blocks. Shaft collars keep the wheels in place. The motors were donated wheelchair motors off an unused power wheelchair. Velocity specifications from the wheelchair were used to determine the maximum speeds of the motors which were within range for the MCP III requirements. For this reason, another 60 tooth sprocket for a #35 chain was used to create a 1:1 ratio between the motor output and the chains so there was no gear reduction. The space between the two wheels was tight so the motor sprocket was placed in a different plane from the wheel sprockets to create space. Therefore, a double stranded #35 chain was used to connect the system. The chains would be tightened with a belt and roller chain tensioner coupled to a fifteen tooth #35 idler sprocket. The benefit of having a spring loaded tensioner is, if the wheels and chain were ever to hit an object and stall, the tensioner would be absorb the load and the motors would not stall along with the chain. Not all of the batteries were able to be placed low to the ground. A large 12V battery is on the lower level, but two more 12V batteries sit on the upper level with the electronics. These batteries are within the ideal area to maintain CG and, because of this placement; they are much closer to the motors in which they are powering.

Motor Controller

The shortcomings of the MCP II were evaluated in order to decide upon a new the motor subsystem for the MCP III. The MCP II used a Dual30 controller coupled with a Blackfin 537 Handy Board in order to perform PID and allow serial control from the PC. This setup had several serious issues that prevented the optimal performance of the motors. It was discovered that if the motor controller was not powered down properly (for example, if the kill-switch was used) it would enter into a state where it would be unable to output a proper PWM signal and cause the motors to be inoperable. Recovering from this state was difficult, and inconsistent, and demanded a better solution for the new MCP.

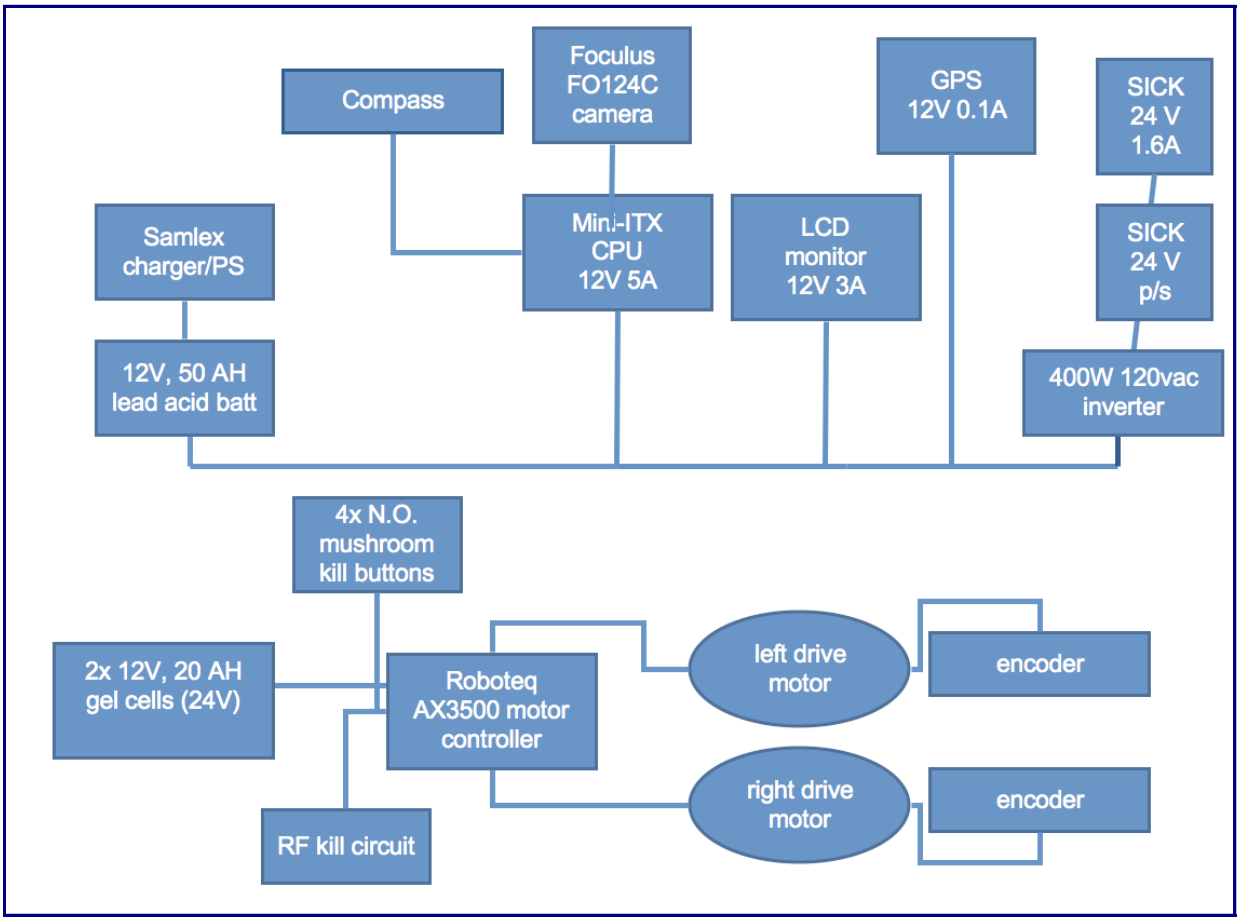
The software aspect of the motor control was also extremely limited. It did not support any form of handshaking so all the motor commands were fire and forget, which created the possibility for a significant disconnect between what the software was thinking and the robot's behavior. In addition the controller was not easily field programmable, as it was not directly linked to the main computer. This created problems on the course when the PID gains were not setup to properly handle hills, and the robot stalled.

The decisions made for a new architecture and hardware design of the MCP III allowed us to fix these limitations and also gives us a much greater degree of flexibility. It was not worth the problems to try and re-invent the wheel and so we purchased a higher-quality commercial motor controller. We chose the Robotec AX3500 series controller due to our requirements and the number of features it supported. This allowed us to have a reliable motor controller with support for serial communication (needed for the Player aspect) along with remote and analog control without any added complexity on our end. This controller was significantly easier to program out in the field and was much more reliable than our previous setup. The controller's well-defined serial protocol made it considerably easier to write a reliable driver for the competition and integration into Player. In addition, the serial control has support for a watchdog timer. Therefore, if the main computer crashes, or the connection is lost for some other reason, the robot will shut down the motors if no input is received for longer than a second.

The MCP III allows for two different control schemes. The first is via an RC remote. The robot uses an Airtronics receiver/controller combo operating on the 75MHz AM band. If the signal is lost while under RC control the AX3500 will immediately cut power to the motors and halt. The second control scheme is via Player/Stage and the serial connection. When running Player, the user can either run a brain to control the robot (as would be used in the competition) or use the playerv tool and control the robot directly through the Position2D driver. Either of the player control methods can be done either locally or via a WiFi/wired network connection.

Power and Electronics

The MCP III is powered by two battery supplies. It was mechanically and electrically beneficial to separate the 12V battery for the electronics and the 24V system into separate locations on the robot because the batteries are located in proximity to the components they are powering. A single 12V, 50 AH lead acid battery operates all of the logic and electronics devices. A series of two 12V, 20 AH gel cells (producing 24v) operates the motors. The 12V battery is placed beneath the computer components and the 24V battery is placed close to the motor controller and motors to minimize the length of wire that would be required. The figure below shows the electrical arrangement.



Block diagram of MCP III electrical system.

Nominal current draw for the logic system is about 12A. This includes the main CPU (a mini-ITX board, 5A), the LCD monitor (2A), and the SICK laser (4A). The SICK laser itself draws 1.8A at 24v; it is powered via a 12 VDC-to-120 VAC inverter, which is followed by the manufacturer's 48W, 120 VAC-to-24 VDC converter.

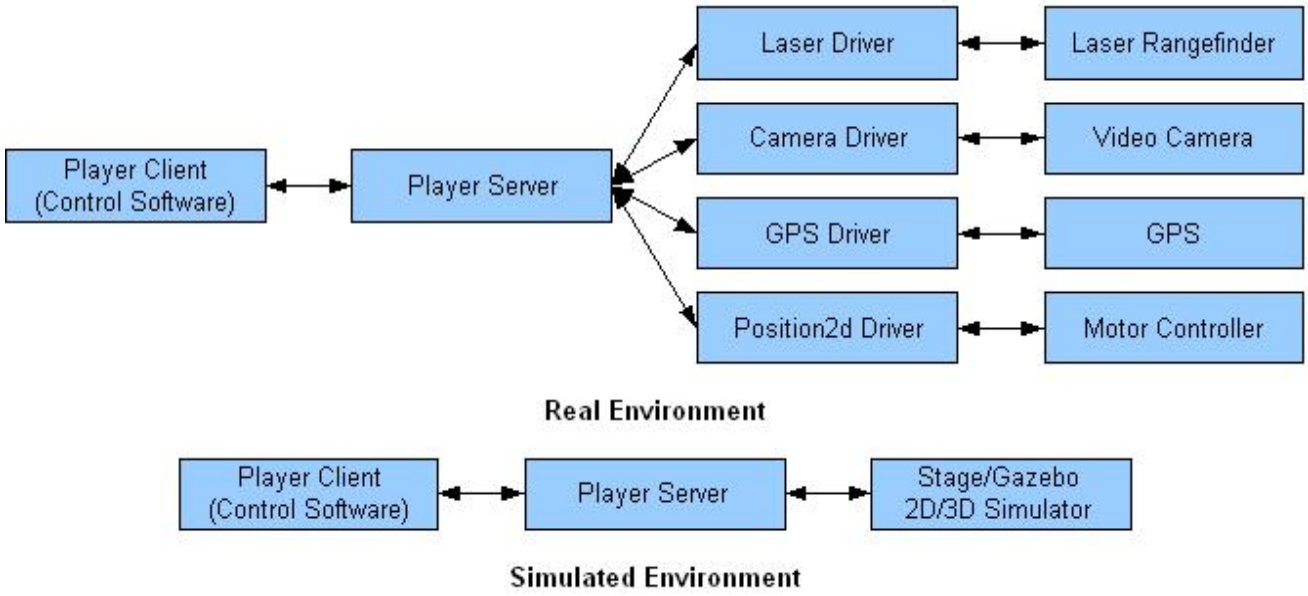
The logic battery is recharged by a dedicated Samlex 12v, 45 am battery charger, which is mounted on the robot. The charger itself can operate as a power supply while the battery is connected, allowing run-while-charging operation. We thus calculate 4 hours of logic runtime away from charge, and given the pattern of usage at the contest event, continuous operation of the logic system.

The motor controller, a Roboteq AX3500, is powered by a dedicated 24v, two-battery supply. It operates two wheelchair motors with quadrature encoders. Four physical "mushroom button" kill switches and one RF kill are wired into the Roboteq's hardware kill circuit.

Software Innovations

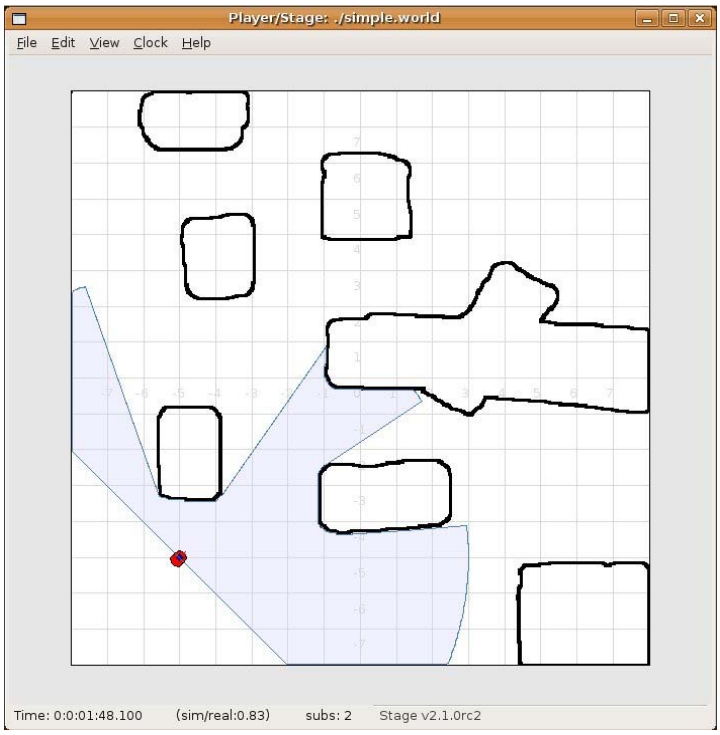
The control software for the MCP was developed and tested using the open source software tools Player, Stage, and Gazebo. Player provides an abstraction layer to interface with the robot's hardware devices. The control software communicates with the standard device interfaces provided by Player without knowing the details of the hardware implementation. This abstraction serves two purposes: first, the same control software can be run on different robots with different hardware implementations; and second, the control software can be tested with simulated hardware devices. Stage and Gazebo provide 2D

and 3D simulation environments. They allow virtual worlds to be set up, and allow virtual robots to interact with the environments by simulating the hardware interfaces. The diagram below shows the communication structure of Player/Stage/Gazebo.

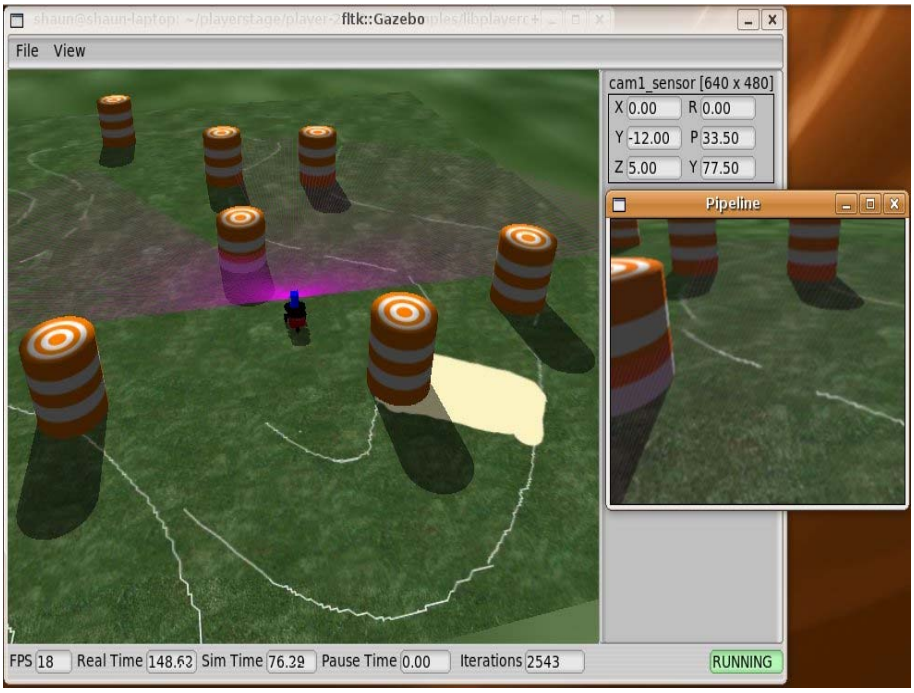


Player provides standard interfaces for a number of hardware devices, and for the MCP the following interfaces were used: laser range finder, video camera, GPS, and position2d. Player uses device drivers for interfacing with the actual hardware devices. Player came with drivers for the SICK laser, video camera, and GPS, and a position2d driver was developed for the MCP's motor controller.

The primary advantages of using the simulation environment were that testing could be started before the robot's hardware was completed, and testing could be performed by multiple developers independently and simultaneously. Stage was particularly useful for the navigation challenge. A virtual 2D world was created with obstacles and way points, and Stage provided the simulated laser and GPS sensor inputs and simulated the robot motion through the position2d interface. The following screenshot shows a simulated test environment using Stage.



Similarly, Gazebo was used to test the software for the autonomous challenge. A virtual world was created with painted lines and obstacles, and Gazebo provided the simulated laser and camera inputs. The following screenshot shows a simulated test environment with Gazebo, including the output from the simulated camera device.



Phission is a vision processing software platform developed by Philip Thoren at UMass Lowell. It can take inputs from V4L video source, or from player server (with customized video capture module). Either a physical camera or a Gazebo simulated camera can be connected to the player server. So the client side control program doesn't need changes when you switch between physical and simulated camera.

There are plenty of vision filters available from Phission platform—either functions provided by Phission, or 3rd party modules wrapped by Phission into a common interface. These features have greatly simplified the vision processing required by autonomous challenge. For example, to add an edge detection function, or Hough transformation filter, we only need to create the related C++ objects with desired initializing parameters, and add them to the vision pipeline.

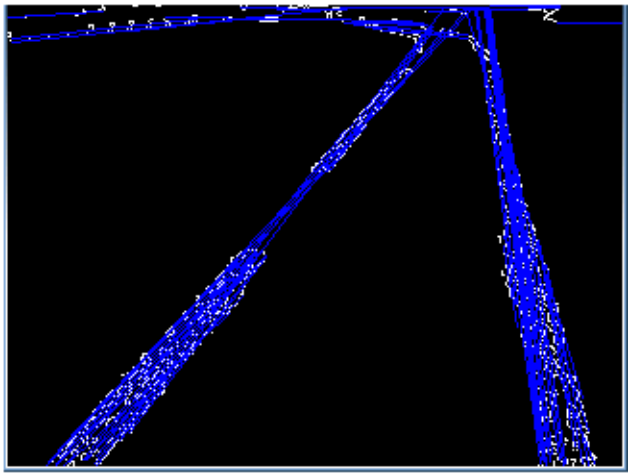
Software

To follow the lines on the autonomous challenge course, we use a set of filters available from Phission platform. White lanes in the frame form the brightest spots in the frame. This information, suggested by Andrew Bacha in his design document is used to detect lines for the robot. The color input frame is converted into "gray" scale by choosing blue channel, which provides better contrast between the green grass white lines on the IGVC field. The reason for this is that the grass will display as slightly blue, while the lines will have no blue in them at all. The "gray" frame is then divided into two vertical halves, and brightest pixel is found for each row and column in both halves using the "brightest pixel" algorithm. The "Hough Transform" is then applied to the brightest pixels, to detect significant lines in the frame.

Once this is finished, the vision system is integrated with the SICK laser's obstacle detection data. When the SICK laser detects any obstacles close to the robot, the vision system applies a preprocessing step, to detect these obstacles. The gray frame is passed through a Canny edge detection algorithm to find all the edges in the frame. Then, it applies a contour detection algorithm

to find objects with a certain area of the frame. The area of pixels occupied by these contours is then removed from the gray input frame by making these pixels black. This pre-processing step filters out objects in the frame which may contain bright points in the frame, and enhances the likelihood of detecting the lines accurately.

As shown in the picture below, the picture on the left is the raw input from the camera, and the picture on the right is the processed picture. The white points are the points on the edge, and the blue lines are the outputs from Hough transformation.

As you can see from the picture above, the broken lines are linked by blue lines. By choosing parameters passed into the Hough transformation, it can link broken lines automatically when the gap is within the specified threshold.

We receive an array of lines from the Hough transform. Each of the line are represented as (x1, y1) and (x2, y2), these are the coordinates of the starting/ending points of the line. We compute the center points and slopes of the lines. Lines on the left and right side of the picture are treated differently. When it can see lines on both sides, the robot tries to go to the center of them. But if only one line is visible, we let the robot follow the line direction but keep a certain distance.

GPS Navigation

For the Navigation Challenge, our robot has been equipped with a Garmin brand GPS receiver, that it uses to determine the locations of the various way points, relative to the location of the robot. However, this particular device provided its own unique set of challenges, both from a mechanical, and software standpoint.

The GPS receiver uses a custom connection to interface with the devices it is normally used for. However, this was inadequate for our purposes. In order to get the receiver to communicate with a computer, and provide meaningful data to the robot, an adapter needed to be found, to allow it to connect through a serial port to the robot's brain. Since this is a custom connection, these adapters don't exist, so it was necessary for last year's team to create one. In order to make the receiver even remotely reliable, a card needed to be created, with ports for the wire coming out of the receiver, and a serial port to connect to the computer. As this was not figured out until the time of the competition was approaching, the card was created in a hurry, and has since begun to fall apart. As a result, a new card needed to be made, for this year's competition.

Programmatically, the GPS problem seems simple, on the surface. All that is really necessary is to use a compass, and the GPS to determine the heading the robot needs to take, and the distance it needs to travel to reach each way point. However, given the time constraints of the challenge, going through each way point in a random, or even hard coded order is insufficient. This

problem ends up turning into a Travelling Salesman problem, without the luxury of being able to attempt all paths, in order to figure out which is the fastest.

As a result, this program settles for a path that is "fast enough." When the challenge is started, the robot simply picks the way point that is closest to it, and begins travelling towards it. This is done by tying into the already written obstacle avoidance routines, and changing the robot's goal from "go forward," to "go towards this point." Once at the first way point, it removes it from the list of way points it needs to travel to, and moves to the next closest one. This cycle repeats until the robot comes into contact with an object that it determines to be a fence. At this point, it attempts to orient itself so that it is travelling perpendicular to the fence, and determines if there are any way points "behind" it. If there are, these way points are given a higher priority, and are travelled to first. Once all way points on one side of the fence are reached, it crosses over to the other side of the fence, and repeats the process.

Obstacle Avoidance Routines

The basic logic of the program used for the Autonomous Challenge is an obstacle avoidance program.

Its sphere of interest is the 180 degree, semi-circular area surrounding the forward section of the robot.

The robot’s turning rate and velocity are determined by a vector that is influenced by how obstacles are positioned in relation to the robot. When no obstacles are present, the robot drives with a vector that points forward and with full velocity. Obstacles on the port side of the robot adds a perpendicular vector pointing to the robot's starboard side. Obstacles on the starboard side of the robot adds a perpendicular vector that points to the robots port side. The velocity of each perpendicular vector is determined by two factors:

1. The proximity of the obstacle. The closer the obstacle is, the greater the velocity for the corresponding vector.
2. The bearing of the obstacle. If the robot is driving forward, obstacles directly in front of the robot will be a greater hazard than obstacles on the starboard and port sides. Thus obstacles directly in front of the robot will have vectors with velocities that are higher than obstacles on the port and starboard sides.

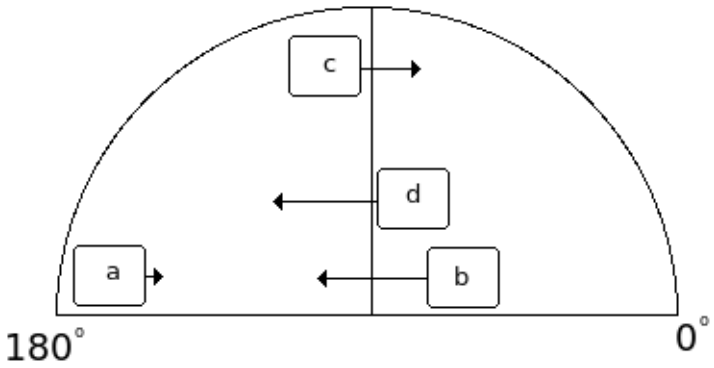
It makes no distinction between white lines or physical obstacles.

There are two sensor modules that run asynchronously from the main program: one that processes laser range finder data for detecting physical obstacles and one that processes camera data for detecting lines. The laser range finder module will make its data directly available to the main program as a two dimensional map without modification. The camera module, however, must process video frames, detect white lines, and map those lines into the same two dimensional map format. This way, the main program interprets the camera modules data as obstacle data.

Every time the main program attempts to determine a direction to drive towards, it makes a 180 degree scan of maps from both sensor modules. At each degree, it takes the minimum of the obstacle range values from each map.

Drawing 1: Example scan of forward area surrounding the robot.

The arrows show the velocity vectors that are created as a result of objects bearing and range.



Drawing 1 shows an example of what the main program may see when it scans all the map data.

Objects “a” and “c” are on the port side and will exert a vector that is perpendicular to the robot's forward direction that points toward the starboard side. Object “a” is closer to the starboard side of the robot than object “c”, so the robot has less of a chance of colliding with it. Thus the vector generated by object “a” will have a smaller velocity than object “c”. “Objects “a” and “b” have similar bearings relative to the robot, but object “b” has a much closer range, so it will exert a much greater velocity in its vector towards the port side as opposed to the velocity vector from object “a” that points to the starboard side.

Overall, the sum of the vectors from objects “b” and “d” will have a velocity magnitude that is greater than the velocity magnitude from the sum of the vectors from objects “a” and “c”. Thus, sum of all vectors will cause the robot's final vector to point somewhere towards the port side.

Position 2D Driver

While Player’s position2d interface provides a powerful and intuitive motion control scheme, no drivers existed that would allow Player to communicate with our robot's motor control system. As a result we could test our algorithms in simulated environments, but the same code could not be used in the physical robot without time-consuming modifications. To resolve this issue, we built a custom position2d driver to meet our specific needs. This driver utilizes Player's support for user-created plug-ins.

The Player control code can provide position2d commands in several formats: "car-like", motor positioning, absolute heading, and forward and angular velocities. The format most compatible with our configuration is a pair of values: forward velocity in meters per second, and angular velocity in radians per second. Our driver accepts this input and from the robot's wheel radius and axle length computes the rotational speeds the robot's left and right wheels must maintain to match this course. These rotational speeds are then translated into power values the represent percentages of the robot's top speed and transmitted to the left and right motor controllers.

The driver also accounts for hardware limitations. If for any reason the driver receives a position command that would require one of the motors to perform beyond a threshold level, the driver will scale down both forward and angular the velocities by the closest possible ratio that allows the robot to remain on course without stressing the motors.

Conclusion

Based on the results seen last year, it was decided that a different approach was needed, and with an entirely new team participating this year, this is exactly what happened. An entirely new robot was built. Nearly all of the software was rewritten, even some of the seemingly basic pieces. The robot was wired differently, and even the methods of testing our robot's performance changed. Because of this, even now, several weeks before the beginning of the competition, our team is confident that great improvements have been made.

The reason for this is that our robot has become simple. Rather than building a robot that relies on chance, and the hope that more complicated solutions will stay out of their own way, this robot was designed to be robust, and adaptable, while still following all the rules required by the various challenges. As a result, the frame of the robot has become smaller, and more agile. The code controlling the robot has become more general, and easier to work with, and our tests allow us to make quick, and accurate changes, without needing to worry if the robot's batteries have enough power, or if we have a field set up to test on.

Hopefully next year will allow us to expand on these solutions even further, allowing us to deal with the problems set forth in the IGVC better. However, for this year, and for this competition, our robot should easily surpass last year's performance in both competitions. With any luck, our performance this year will cause next year's class to face a much harder challenge. Perhaps we will even be challenging them to try to finish higher than first place.

Parts List

Description	Price	
New Parts		
SEC12-15 Samlex 12V 15A 3 Bank 3 Stager Charger	$149.00	
UB12500-45977 12 Volt 50 AH UB12500 12V AH	$153.00	
UPG 12 Volt Minder	$30.00	
Roboteq AX3500BP motor controller	$445.00	
80/20 HT Framing System	$495.73	
Misc Nuts & Bolts	$100.99	
Chain & Drive Parts	$353.83	
(2) Invacare 1095852 Wheelchair Motors	$1,390.00	*
(4) Wheels and Inner Tubes	$221.16	
Lexan Mounting Plates	$15.00	
	Subtotal	$2,709.59
Old Parts		
SICK LMS-200 Laser Ranger	$6,000.00	*
Foculus FO124C Firewire Camera	$1,000.00	*
Mini-ITX Mainboard & Pentium M Processor	$500.00	*
Dell LCD Monitor	$100.00	*
Garmin GPS16-HVS	$100.00	
(2) 12V Lead Acid Batteries Powering Motors	$140.00	
80/20 Building System and Steel Tubing for MCP 2	$400.00	
Doorbell for Wireless E-Stop	$14.00	
Misc electronic Components	$300.00	
	Subtotal	$8,554.00
	Total	$11,263.59
* Donated Parts		