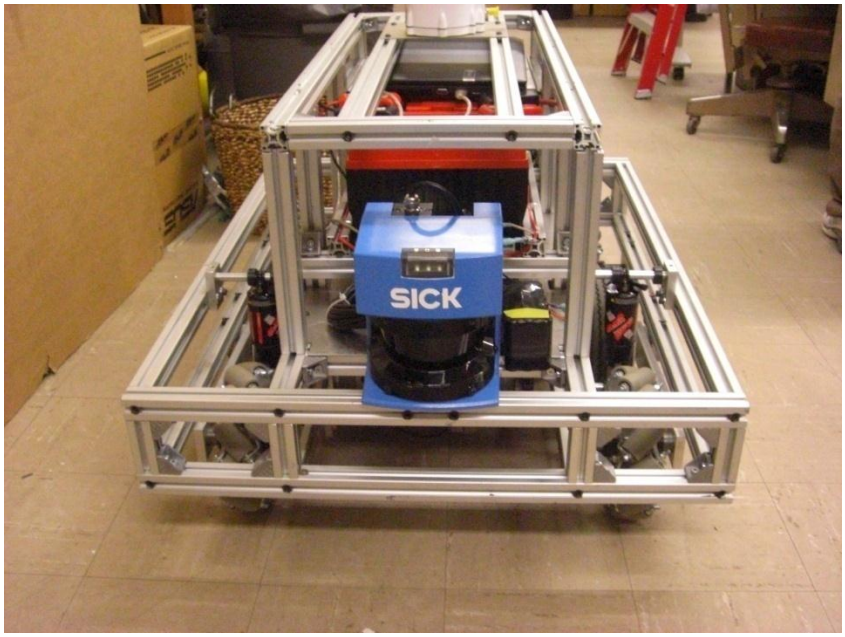
PROJECT 14



Stony Brook University
The Robot Design Team
Intelligent Ground Vehicle Competition 2011

Adam Siegel
Matthew Quigley
Kent-Diens Joseph

Faculty Advisors:
Professor Yu Zhou (Department of Mechanical Engineering)
Professor Jon P. Longtin (Department of Mechanical Engineering)

Faculty Advisor Statement

I certify that the design and creation of PROJECT 14 has been significant and is equivalent to what might be awarded credit in a senior design course.

_____

Professor Yu Zhou
Department of Mechanical Engineering
Stony Brook University

5/3/2011

_____

Date

1. Overview

PROJECT 14 was completed by three undergraduate mechanical engineering students for a senior design project. This senior design group is part of the Stony Brook Robot Design Team, which is sending a total of two robots to the 2011 IGVC. One of our members has been a part of the Robot Design Team in the past, which has competed in IGVC. Knowledge from past experiences, along with the current set of rules for the 2011 IGVC was used to design this robot.

It took nine months to design, fabricate, and test the robot. A five stage design process was implemented to help keep the task of building an autonomous vehicle under control. The first three stages of the design process, which include competition research, conceptual design and detailed design, took four months. Most of the fabrication was done in a month. This left four months for testing, debugging, and design modifications. This report will expand upon the design process. It will also portray the mechanical and electrical aspects of the hardware, as well as the design and implementation of software.

2. Design Process

A five stage design process was used to design, manufacture, and test our robot. The first step in the design process was competition research. A user needs survey and Product Design Specifications (PDS) were created to define the key requirements of the robot. At each step of the process, decisions that were made were checked with the PDS to ensure all requirements are met. Research and an parametric analysis was done on prior robots that competed in IGVC, as well as on related patents, to get a better idea of qualities that we would want in our own robot.

The second step was conceptual design. All possible designs were generated and compared by using a set of criteria. An iterative process allowed for the best design to be developed. The third step was detailed design. The classes that we have taken over the past few years have given us knowledge that was implemented into the design of the robot. Section 3.1 will describe the mechanical aspects of the detailed design phase while the electrical components will be covered in section 3.2. Each component was justified by a Component Design Specifications (CDS).

The fourth stage is centered on building the robot. Most of the fabrication was done in a four week period. Several processes were done on raw material to achieve the final product. Examples of machinery that were used include: milling machine, lathe, band saw, and rapid prototype machine. The fifth stage was to test and debug the robot. This was done systematically. First the motor controller was tested to ensure the robot could move. Sensors were progressively integrated until every sensor had been added and was working properly. This method was chosen to make debugging easier; if the robot did not respond properly, than the issue must reside in the newest addition to the system. This stage is covered in section 4.Even though each step had specific guidelines and

requirements; each task was not done independently. Often, it became necessary to return to previous steps to modify the system. Graphically, this process is shown in Figure 1 below:

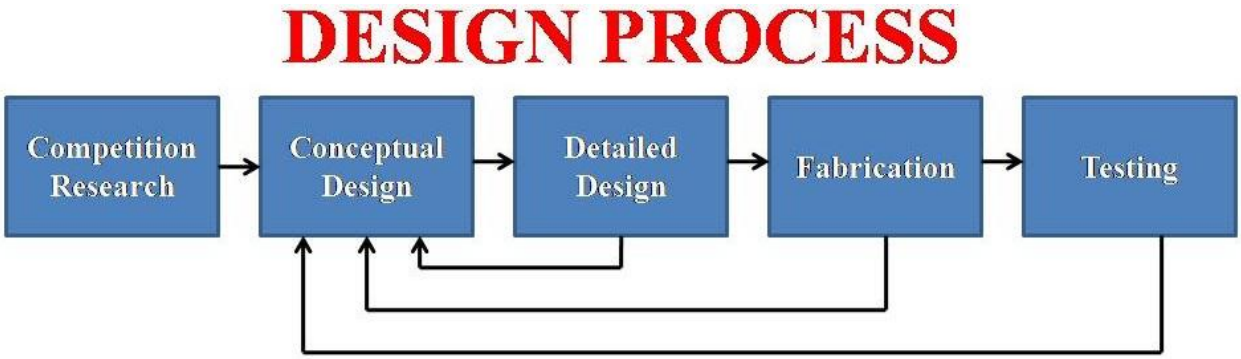


Figure 1: Design Process

3. Hardware

This section will describe the physical attributes to the system. This includes the design and analysis of the mechanical and electrical sub-systems. Attention will also be given to design innovations and safety.

3.1 Mechanical Design

Once the final design concepts were chosen, the mechanisms had to be designed. Most of this stage was completed in a three month period, although some modifications became necessary later on. Autodesk Inventor was used to satisfy two goals.

The first goal was to be able to model the robot to ensure that all of the components meshed with each other without any interference. Figure 2 shows a CAD drawing of the final assembly. Modeling made fabrication faster and easier. Drawings were created for each part and the final assembly was used to know where each part fit into the structure. Conflicts in part location were solved through Inventor so that everything would fit together for fabrication. The second goal was to analyze each component. This was done with Finite Element Analysis (FEA) through Inventor. FEA was done on each component designed that would be placed under loading conditions. In actuality, several steps were needed for the analysis. Loadings and constraints were placed onto the part. Next, the part was meshed. An analysis was then conducted. Special consideration was given to: the locations of stresses, displacement, and factor of safety. An example of this is shown in Figure 3. Here, the deflection of the moment arm is considered. The

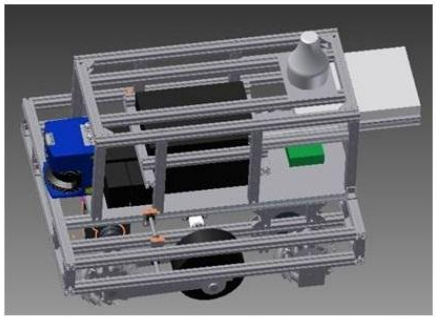


Figure 2: CAD Drawing of the Final Assembly

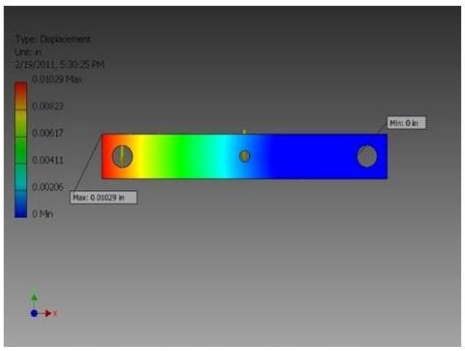


Figure 3: Deflection of Moment Arm

displacement is zero at the constrained hole and maximizes where it connects to the wheel. The maximum displacement was shown to be 0.0103 inches. The deformations are within tolerances.The mechanical sub-system is divided into four sections that will be described below.

3.1.1 Chassis

The chassis is comprised of extruded aluminum cut to various lengths. Extruded aluminum was selected due to it being versatile. The aluminum has grooves that can be bolted to other components, which makes assembling the robot easy. This makes the robot modular, allowing for modifications. The material itself is also easy to machine. The company that sells the aluminum also sells relatively expensive connectors. To alleviate this monetary issue, notches were cut into the aluminum whenever it was in contact with a groove. This decreases the degrees of freedom of the parts to one. The last degree of freedom is removed by adding a M8-1.25x35cap screw. The area near the notch is threaded and a hole is added by the groove of the reciprocating piece. The bolt is then tightened into place. This solution decreases cost and weight. The connector the manufacturer sell is used sparingly, only when there is a need for two bolts to cross paths.

3.1.2 Drivetrain

From the end of the conceptual design phase, it was determined that the robot should have 6 wheels. The drivetrain, which is shown in Figure 4, consists of two powered pneumatic wheels, placed in the center, and four passive mecanum wheels each placed at a corner. This allows the robot to turn on the spot, a key feature when maneuvering around obstacles. The mecanum wheels have rollers set to 45 degrees relative to the direction of motion, which when rolling decrease the shear stresses acting on the wheels during a turn.

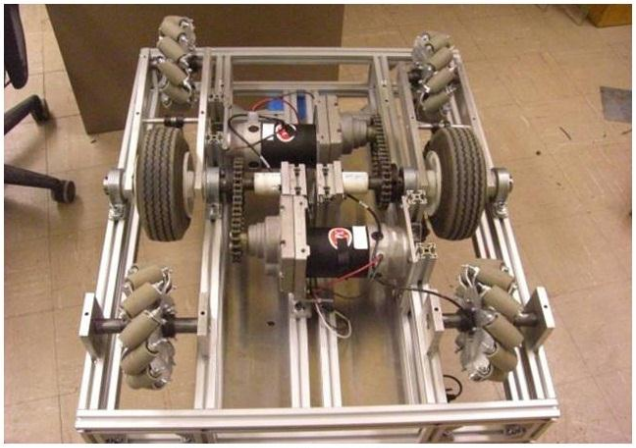


Figure 4: Project 14's Drive Train

It was also determined that the best mechanism for transferring power from the motors to the pneumatic wheels was by using roller chains and sprockets because of their robustness and their relatively low cost compared to other power transmission mechanism. Knowing the power limit of 0.8hp at 219 rpm of the motor, the diameter of 8.5 inch of the pneumatic wheel, and choosing a linear speed of 9mph, the angular speed ω_2 of the pneumatic wheel was calculated. Then, assuming a driver sprocket with N_1=23 teeth, and using the inverse relationship between angular speed ratio and the ratio of the number of teeth on the driver and driven sprocket

$$\frac{N_2}{N_1} = \frac{\omega_1}{\omega_2}$$

The driven sprocket N2 (which transmit power to the pneumatic wheel) was found to be:

$$N_2 = 23 * \frac{\omega_1}{\omega_2} = 13.995 = 14\ teeth$$

This yielded a final linear speed of 9.1mph. Using the obtained driver, driven number of teeth, corresponding angular speeds, and instructions from the Martin's Sprocket and Gear Company's website, were used to select sprockets and chain with a number of 40 (according to the ASME standards for chains and sprockets): a flat sprocket with no hub was selected for the motor, and another one that comes with a hub was selected for the pneumatic wheels.

The diameter of each shaft that held the wheels had to be determined. 4340 normalized steel was chosen to be the shaft material. The torque was assumed to be 300 lbf*in even though the actual torque is estimated to be 230 lbf*in. The tensile strength, S_{ut}, is 125 kpsi. The distortion energy-Goodman criterion was used to find the shaft diameter which is shown in the following equation:

$$d = \left(\frac{16n}{\pi}\left(\frac{(4(K_f M_a)^2 + 3(K_{fs} T_a)^2)^{\frac{1}{2}}}{S_e} + \frac{(4(K_f M_m)^2 + 3(K_{fs} T_m)^2)^{\frac{1}{2}}}{S_{ut}}\right)\right)^{\frac{1}{3}} \quad (1)$$

The above equation has several unknowns. The factor of safety, n was found to be 3.3. A conservative assumption was made that an end-mill keyset ran through the entire shaft. Because of this distinction: $K_f = 2.14$.Another assumption is made that claims that midrange moments and alternating torques equal 0. In other words: $M_m = T_a = 0$. The shoulder fillet, K_{fs}, was assumed to be equaled to 3. The parameter $S_e = 0.5 k_a k_b k_c k_d k_e k_f S_{ut}$. It was found that $k_a = a(S_{ut}/1000)^b$ where a = 2.7 and b = -0.265, $k_b = 0.879 d^{-0.107}$, and $k_c = k_d = k_e = k_f = 1$.

Since the diameter the shaft is a component of one of the parameter for its own calculation, an iterative process had to occur. To speed up this process, a program was written in MATLAB such that each iteration would take little time to run through. The following result was eventually calculated:

d = 1 in.

3.1.3 Suspension

An inherent problem occurs due to the geometry of the drivetrain. The drivetrain has two rows of three wheels, with only the centered wheel being powered. The problem is that if the driven wheels lose contact with the ground, the robot won't be able to move. The robot must be able to make it up the steepest incline, which is when the robot climbs the ramp as shown in Figure 5. There is a maximum gradient of 15% which is equaled to 8.53 degrees. The figure shows the center wheel lifting off the ground.

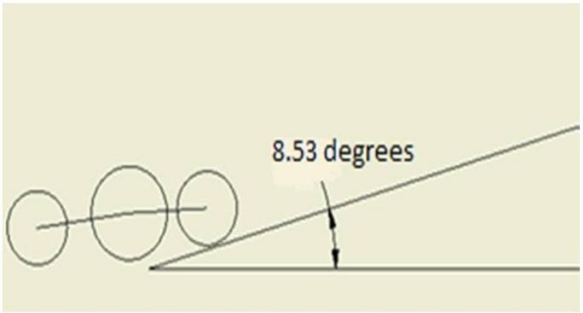


Figure 5: Drive Train Incline

To alleviate this issue, a bicycle air shock suspensionis added between the pneumatic and front mecanum wheels. These mecanum wheels are attached to the center wheel via two moment arms and are constrained to the chassis by the shocks. When the robot is on level ground, the wheels are held level, such that the center wheel contacts the ground. When the robot goes up the incline, the shock allows the front two wheels to move due to the added force acting on these wheels. This keeps the center wheel in contact with the ground at all times.

The air shock suspension that we chose are two Rockshox Monarch, as shown if figure 6. It has a maximum displacement of 2.5 inch, and the distance between the two mounting holes are 8.5 inch. The advantage of using air shocks over a spring and damper suspension is that in the air shocks the spring constant can be varied by changing the air pressure in the shock by adding or removing air in it. The air shocks also come with different turning knobs that can be used to adjust its damping.

Figure 6: Air Shock Suspension

3.1.4 Mounting Components

For most of the components, it was relatively easy to mount them to the chassis. These components required to be bolted either to groves in the extruded aluminum or into nuts that fit on the other side of holes through the extruded aluminum. There are a few exceptions. The laptop is attached to a plate by Velcro. The plate is attached to a drawer, which allows for easy access to the laptop. Another exception is with the camera. A mount out of aluminum was made to hold the camera, as shown in figure 7. The hole for the camera has an opening on the top and is slightly smaller than the diameter of the camera, which holds the camera in place. This connector can change orientation with the chassis, allowing the cameras to change its viewpoint. The encoders are located on the center shaft, where debris can damage them. To prevent this, cases out of PVC were cut. These case press around the encoders, protecting them, and yet can be pulled out, which would give access to the encoders. This is shown in figure 8.

Figure 7: Camera Mount

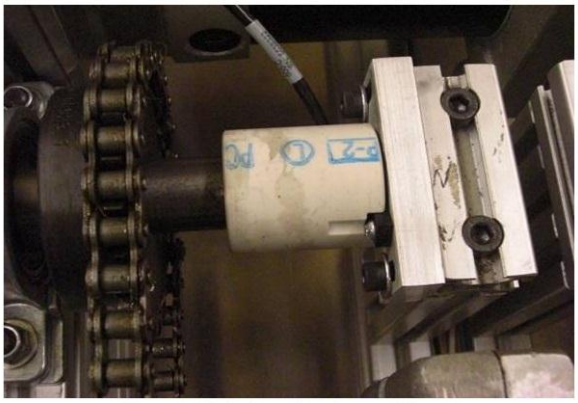


Figure 8: Protective Case Around Encoder

3.2 Electrical Design

3.2.1. Sensors

We utilize five types of onboard sensors for autonomous movement and navigation. Most importantly, we must make sure the vehicle does not run into any obstacles. The Laser Measurement System sits on the front and has a 180 degree view of the surroundings from its point of location. If an obstacle sits anywhere in front of the vehicle, it will immediately stop and reevaluate its position relative to its destination, taking into account the obstacles (see Section 4). This is the main sensor, and has priority over the others because of its ability to directly detect impediments to the vehicle's path. The GPS is used to give the CPU the vehicle's position relative to the current waypoint objective. The cameras are also essential parts of the vehicle's navigational ability; we must keep the vehicle between two white lines. Therefore, we must have all three cameras (two sides and one front) to make sure that we do not cross a while line. The front camera will also be used to recognize the red and green flags.

The compass plays a less important, but still crucial role for the correct navigation of the vehicle. The main purpose of the compass is a "safety switch". For instance, based on the LMS reading, the CPU might decide that the only clear path is that from which it came. We obviously do not want the vehicle to backtrack, so before we move, we use the compass to make sure that vehicle does not move in the direction from which it came (backtrack). The GPS keeps the vehicle on track by monitoring the global position of the vehicle at all times. It makes sure that the vehicle is headed in the right direction for the waypoint, and works together with the other sensors to make the vehicle fully autonomous.

3.2.2. Power Demands

The motors are by far the most power demanding; to operate at 0.8 hp at 219 rpm, each motor requires 28.6 amps. Therefore, we dedicated an individual power source for them[1]. Based on monetary constraints, we had to do our best with certain hardware devices. We utilize two 24V Brushless DC motors, each of which provides power to one wheel on each side of the vehicle via sprockets and a chain. In order to provide the necessary power and torque to the motors, we have two 12V 65Ah Marine batteries attached in series to a SPDT switch such that they can be grounded when not in use. When in use, the circuitry is designed such that fuse is placed between the batteries and motor controller so that we are limited to a 70A current at 24V.

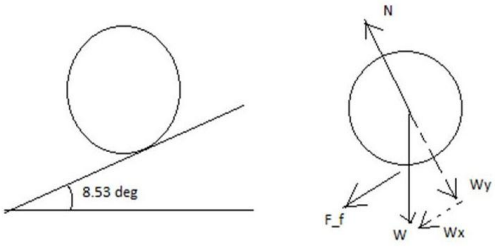


Figure 9: Pneumatic Wheel Free Body Diagram

To make sure that the motor power of 0.8hp at 219 rpm will be sufficient for the robot travel on the 8.53 degree ramp, a free-body diagram in Figure 9 was created. By performing a force balance on the wheel (assuming a distributed weight of 50lbs on each wheel, and

[1] This is soon going to change, as we plan to consolidate all of our power sources into one.

a wooden ramp), it was determined that a minimum force of 35.35lbf was necessary to keep the wheel in equilibrium on the ramp. This minimum force was then used to find the minimum torque required, which was finally used to determine the maximum angular speed, and the maximum linear speed of 8.49mph. Hence, for the robot to be able to go up the ramp it will have to do so at a speed of 8.49mph or less (which is expected).

3.3.3 Circuitry

We utilize a Sick Optics LMS 200 (Laser Measurement System) on the front of the chassis for obstacle detection. In addition, we use a Phidgets 1040 GPS for global position information. The LMS must be provided 24V at 6A (mainly for the internal heater) for normal operation, while the GPS only needs 5V. In addition, the LMS is in series with a 10A fuse.

We utilize other sensors for various reasons (see Section 3.4 for a detailed overview). Our compass and webcams are powered from our computer. As of now, we are utilizing the computer's battery to power the CPU, three webcams, and a compass. Figure 10 is a diagram of the vehicle's electrical system. Each of the fuses and switches are used to turn the various sensors on and off, and are situated on a single plate at the side rear of the chassis.

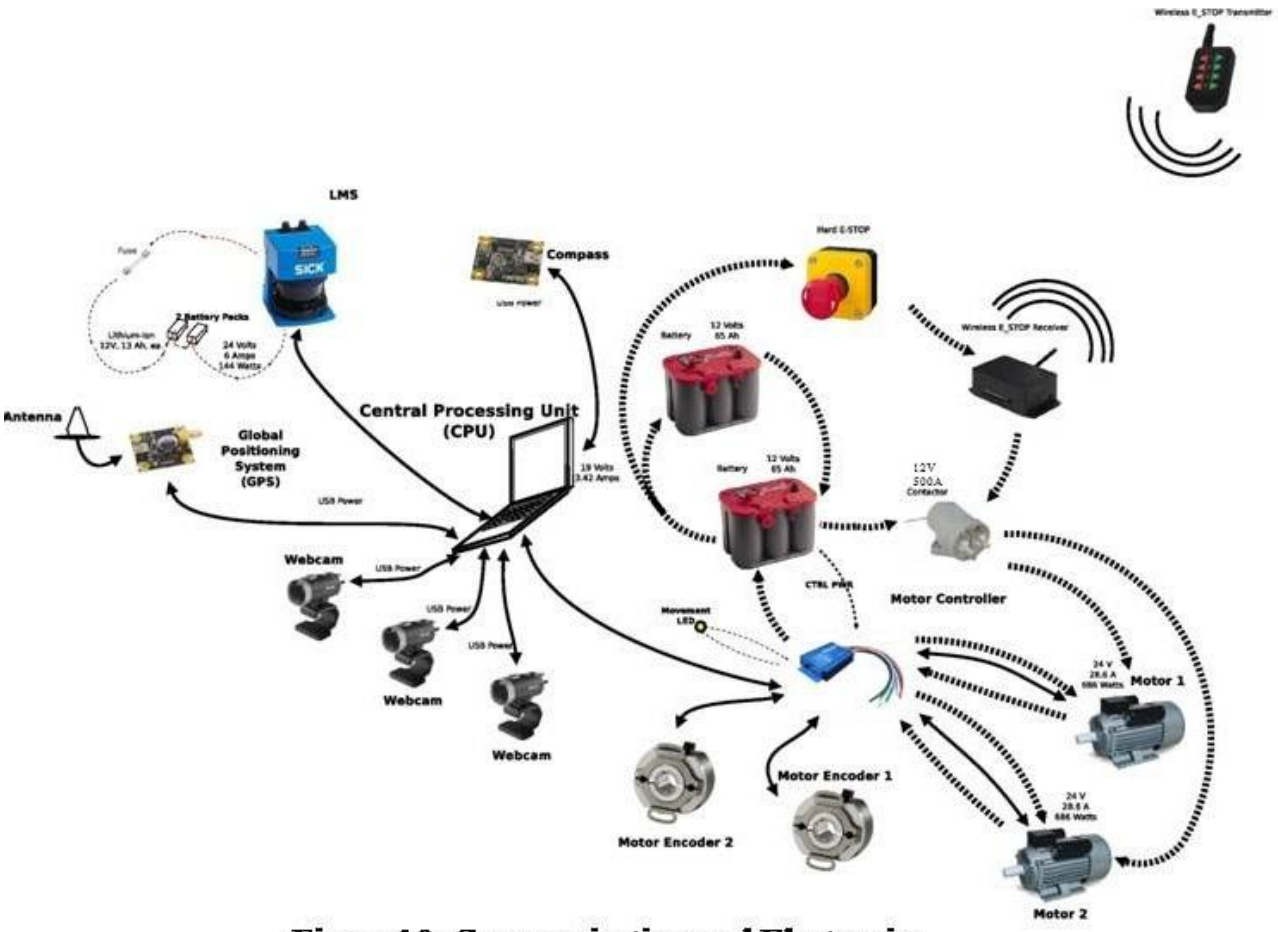


Figure 10: Communication and Electronics

The circuit for the emergency stop (E-stop) and the wireless E-stop consists of: the regular emergency push button switch, a wireless receiver which has a latching relay in it, and a contactor which is normally open and would only close when activated. This can be seen in Figure 11.

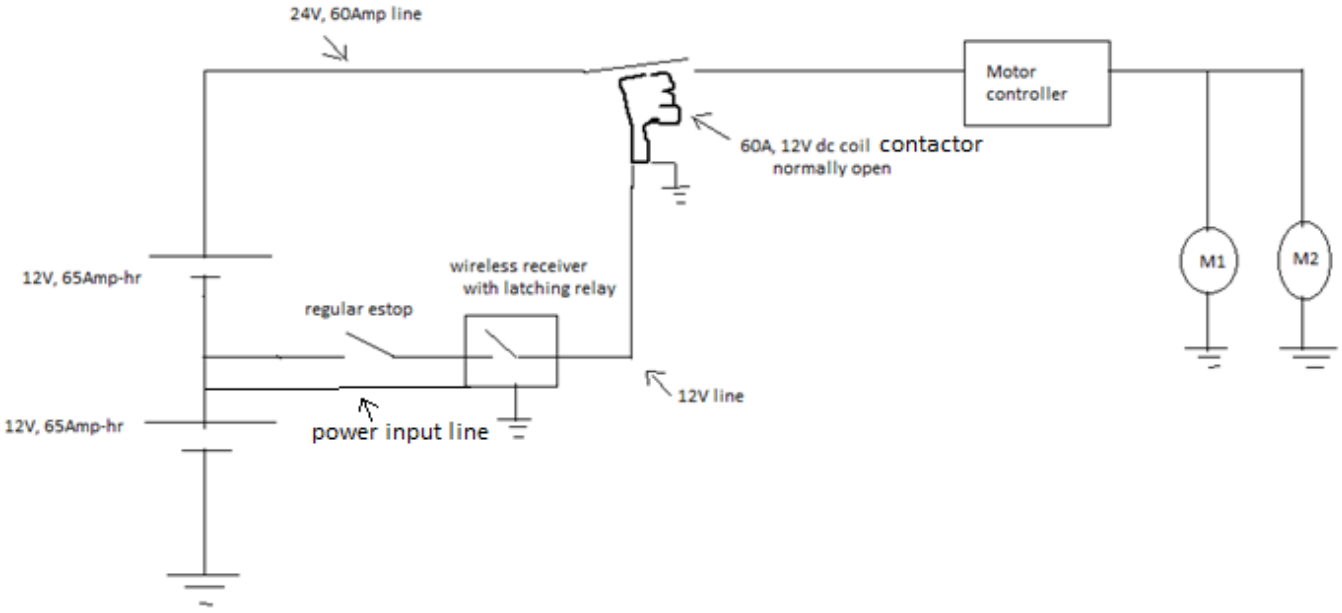


Figure 11- Circuit Diagram for the regular and wireless E-stop switches.

When the robot is running, both the regular E-stop and the wireless receiver relay are closed which allows 12V electricity to activate the contactor with minimum specifications of 60A 12V coil, and to close it. When either the regular E-stop or the wireless receiver relay are opened (by either pushing the regular E-stop, or activating the wireless receiver with its transmitter), this causes the contactor to open the line connecting the batteries to the motor controller and thus causes the robot to stop, as power is immediately cut to the motors. Similarly, if either the regular E-stop or the wireless relay was to be knocked off the robot while it's running, this would create an opening in the circuit thus deactivating the contactor and thus the robot would stop. A 500 Amp, 12V dc coil Kilovac Lev200 series contactor, a Linx Technologies receiver and transmitter, and a round E-stop button with a 469V DC and 10 A , were used to create the circuit.

3.3 Design Innovations

One of the main innovations that is shown from this robot is with the unique drivetrain. If pneumatic tires were used instead of mecanum wheels, whenever the robot went to turn, shearing forces would be acting on the wheels that are in the corners. This would happen due to rigid body motion. The wheels would be forced to move in a direction that is constrained (not in the direction of motion allowable from the rotation along the shaft). A diagram of the designed drivetrain is shown in Figure 12. The rollers are positioned such that they can roll along the path of the robot while it turns, which is symbolized by the circle. The four mecanum wheels use the 45 degree rollers along with the two pneumatic wheels to support the weight of the robot as well as allowing it to turn in place.

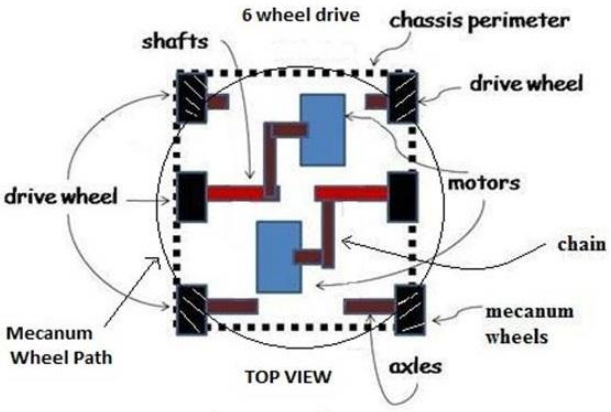


Figure 12: Drive Train

Another design innovation was with the suspension. Through testing, it was determined that the vibrations of the robot on both grass and pavement were negligible. This removes a common reason to have a suspension, and is why we felt that a suspension on each wheel was not necessary. Having air shocks on only the front two wheels is an easy solution to ensure the robot can climb the ramp. This solution requires minimal design work as well as minimal cost. The air shocks that were selected have three parameters that can be adjusted. This allows for the dynamic motion of the robot up a ramp, or any other incline, to be controlled.

3.4 Safety

Safety has been a priority of the robot. One issue was that having the wheels outside of the chassis could be a safety hazard to anyone standing nearby. To protect people, as well as the drivetrain, the chassis was built around all of the components. To prevent people from contacting wires that have current flowing through them, certain precautions were taken: electrical tape was used to seal any exposed connections and plastic covers were used for the terminals of the lead-acid batteries. In addition the fact that the round head E-stop button and wireless switch are connected in series enables the robot to be stopped by activating either switch. Also if either one of these switches were to be detached from the robot as a result of an accident, then this would create an open circuit and also force the robot to stop.

4 Software

Our software is an in-house developed program written for the specific purpose of teaching the vehicle how to respond to different scenarios. Written in C++ and designed to be executed on a Linux platform, our program utilizes many open-source libraries (OpenCV Image Processing Library, SICK LIDAR C++ Toolbox, and the standard Phidgets Library). In addition, we take advantage of advanced parallel programming techniques in order

to complete its task. The software provides a means of interfacing the electrical and mechanical subsystems by providing the vehicle with logic and structure. The CPU itself is a laptop computer, which contains an Intel Dual-Core 2 GHz processor P6100 processor with 3 gigabytes of memory. Just as any normal program runs, our program will run on one processor. However, our program splits up tasks by utilizing multiple processes and multithreading. For example, the computer itself recognizes multiple CPUs, which gives an intuitive and unambiguous way to split up work via forking multiple processes. This will be especially helpful for image processing, as normal vehicle operation can continue while images are being rendered and processed (see below) on different CPUs. Each sensor will be connected via a USB port to the computer.

When the program starts, it immediately splits up its tasks. As stated in Section 3.3, there are five main sensors; the motor controller, the LMS, the cameras, the GPS, and the compass. Therefore, we separate jobs as shown in Figure 13 by creating separate processes for each of the sensors. This is more efficient due to the nature of the sensors. For example, in order to detect the lines, we create a video stream and filter through the frames as they come in to look for the color white.

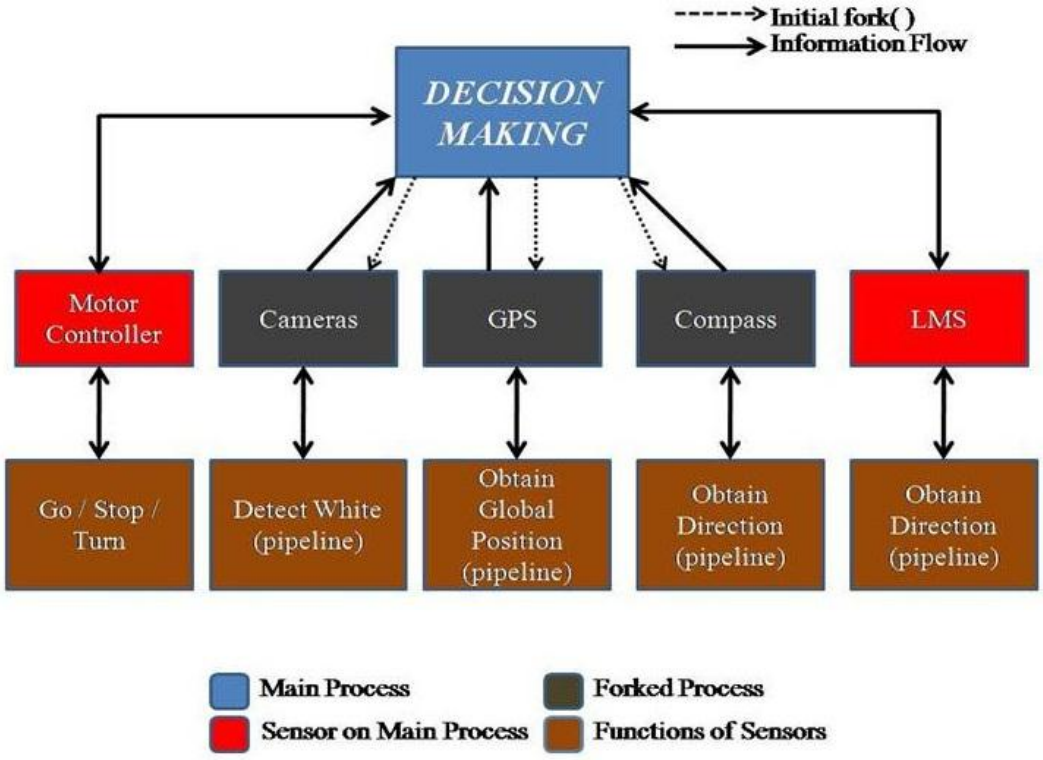


Figure 13: Vehicle's Decision Making Process

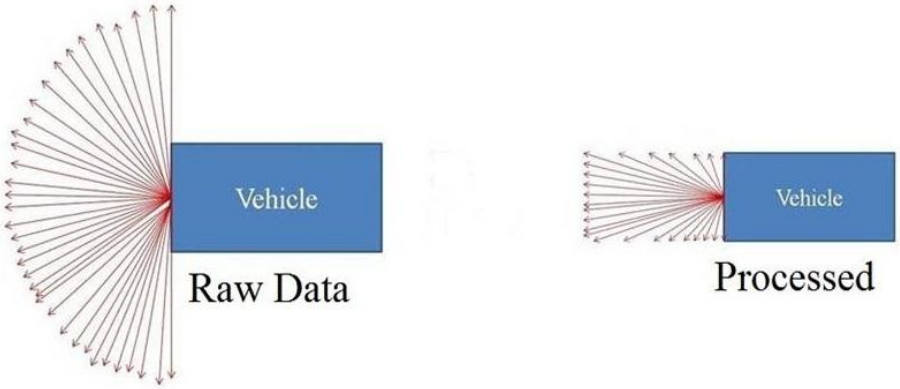
We create a certain threshold for the color white based on the distance the camera sits from the ground (for example, if 5% of the image is white, we must have reached a line). As it is too computationally expensive to start and end the video feed each time we want to look for a line, we run the camera process simultaneously. We keep the video stream *open* on our camera process so that it can continuously monitor the images. The main decision

making process continuously polls the cameras' sub-processes. Once the threshold is reached, it triggers an interrupt in the main program by piping to it. The main program sees the piped message from the camera and responds accordingly. Therefore, the main program takes care of the decision making, and is interrupted when a white line is detected. A screenshot from the white line detection process is shown in Figure 14

(a) (b)

Figure 14: (a) Raw Image from Camera, (b) Filtered Image with 5% White Detected

In addition to this communication between processes, we also utilize the OpenMP API for shared memory multiprocessing. This is important for various functions, and is most easily understood when analyzing the data obtained from the LMS. As noted before, the raw data from the LMS is a 360 element vector (one distance reading, every half degree). However, we are not interested in an object that may be on the side of the LMS, only one which is impeding its path (in front of it). Therefore, we multiply each of the distances by the sine and cosine of the angle to find the distance away something is in front of it (see Figure15)



(a) (b)

Figure 15: (a) Raw Data from LMS, (b) Processed Data from LMS

Although not a very expansive calculation by many standards, we can still speed it up by doing it in parallel via OpenMP. We utilize OpenMP for all calculations that involve at least a mediocre amount of computation. For instance, we may also implement this API in the camera functions, as identifying white involves looping through

pixels, which can be done nicely in parallel. Therefore, when any one of the processes is not working to its full capacity, the main program will send it work.

We are utilizing a Phidgets compass in order to ensure our vehicle does not turn around when it sees a clear path behind it (where it came from). Although the compass data is streamed on a separate process, we only receive data in the form of a magnetic field reading. We must somehow convert this to a direction (0 to 360 degrees, with 0/360 degrees being magnetic north). We do this by calibration, which involves turning the vehicle in circles while we obtain magnetic field readings. This is shown in Figure 16 The red line is the raw data after the vehicle spins around three times. When this is done, we take one of the full revolutions and scale the magnetic field such that one revolution corresponds to numbers between 0 and 360 degrees. This scaled value is shown in the figure as the green line. After the compass is calibrated, these readings are stores in an array. Therefore, when streaming readings, the compass can interpolate a magnetic field value and match it up with a number between 0 and 360. Finally, when turning, we can check the direction in which the vehicle wants to turn with the compass value to ensure that the direction in which the vehicle would like to proceed is not the same in which it came.

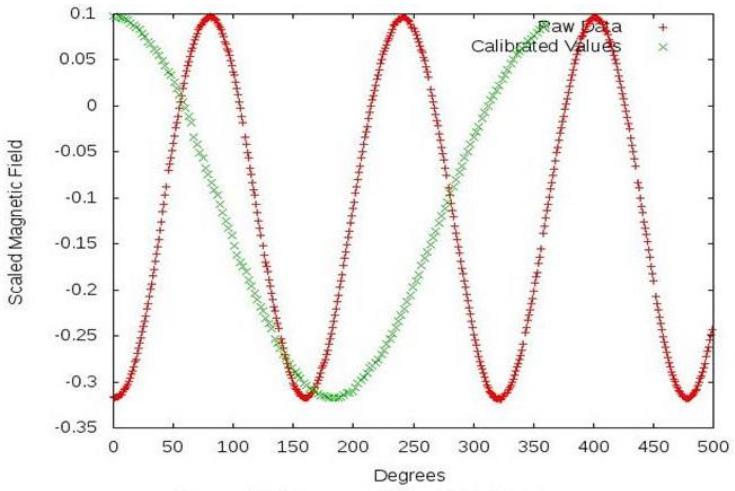


Figure 16: Raw and Calibrated Data from Compass

Our lower-level programming involved interfacing with numerous sensors via a serial drive, and was typically done using the standard C++ method of building communication classes. Our final higher-level algorithm however involves utilizing the feedback from all of the sensors to develop a hierarchical decision making program (Figure 17). It involves using feedback from all of the separate sensor processes to make decisions on where to go. The very high-level pseudo code associated with the vehicles decision making process involves a list of things to check and do, along with the priority (which is checked first) based on the need to stay away from obstacles and lines.

```
Fork processes
        Initiate all processes
        While not at destination:
                Look for obstacles
                        Avoid obstacles (priority=1)
                Look for Lines
                        Avoid lines (priority=2)
                Get compass reading
                Move towards open path
                Re-obtain global position
        Kill all processes
End
```

Figure 17: High Level Algorithm

5 Testing

There have been many changes in the design of the program after testing. With this said, there will be many more changes as we continue testing, as nothing is truly perfect. Initially, we did not have separate processes controlling each sensor, and main had to run on one thread as it controlled everything. This was not very efficient,

as the cameras data stream had to be opened and closed after every frame was take. In addition, our preliminary code was structured poorly, as objects were created and destroyed inside every loop in order to get readings from certain sensors (such as the LMS). Needless to say, this was very inefficient and was soon changed. The solution was trivial, as pointers just needed to be passed to the functions that needed to be initialized and uninitialized only in the main program, and not in the functions themselves. After much testing, we finally found a nice balance between putting every sensor on its own process and passing pointers between functions in the main program so that things did not have to be reinitialized after every call.

With this, we try not to make code execution time the limiting factor. Code is executed quickly; however receiving messages from the sensors is the most limiting factor. For instance it takes approximately 300 milliseconds in order to obtain and process the readings from the LMS in the way we would like. Therefore, if we run at full speed (10 miles per hour), the vehicle can move a distance of approximately 1 ½ meters before another readings is taken and processed. However, we do not run faster than 5 miles per hour, and can still move ¾ of a meter before a reading is taken and processed. Therefore, we set a buffer region depending on the speed. We scan the readings and look for a reading less than the buffer reading; if one is found, the vehicle is stopped and its position is revaluated. We continue to change and test, we would like to bring this excess movement before complete stop to a minimum.

As the most limiting sensor, we can only run as fast as the LIDAR allows. In terms of runtime, the LIDAR batteries must be changed periodically (about every few hours if running full time). However, the motors will be supplied power for months at a time, as we are using two very high capacity batteries. With this, we are constrained in time by the LIDAR and laptop batteries. This is why we are currently making changes to run both the laptop and LIDAR off of the motor's lead acid batteries. If we don't, of these must either be replaced or recharged within a couple hours' times. It would make most sense to run the laptop and LMS off of the motor's batteries' because this would allow for an extended period of time without needing recharge. However, before we can do this, we need more testing of exactly how much current is used during normal operation, at different speeds, and up ramps, etc.

6 Cost

Cost was a key factor in determining the design and fabrication of this robot. The table below contains a detailed list of the parts that were used in this robot.

Table 1: Materials Attained for Robot

Part Name	Make/manufacturer	quantity	buying	in house manufacturing	Cost
flat plate sprocket	Martin's sprocket and Gears	2	yes	no	2 x 14.36
bored to size sprocket	Martin's sprocket and Gears	2	yes	no	2 x 30.37
#40 chain	Applied.com	1	yes	no	19.90
odyssey battery	Odyssey	2	no	no	donated
Lithium ion battery	Tenergy	1	no	no	donated
pneumatic wheels	N.A.	2	no	no	donated
Master Links	Applied.com	2	yes	no	2 x 2.11
mecanum wheels	AndyMark	4	yes	no	336.00 (set of 4)
Alumnum (2024,6061)	N.A.	1	no	yes	donated
dc motors	NPC	2	no	no	donated
Suspension Shocks	Xtreme Bike and Sports	2	yes	no	390.00
mounted ball bearings	Grainger	4	yes	no	4 x 17.66
key stock	McMaster Carr	2	yes	no	2 x 7.24
shaft collar	McMaster Carr	8	yes	no	8 x 1.44
ball bearing	McMaster Carr	8	yes	no	8 x 10.92
1 inch steel shaft	Onlinemetals.com	1	yes	no	67.1
extruded aluminum	Airline Fluid Power	N.A.	yes	no	159.20
drawer slider	McMaster Carr	1	yes	no	29.19
Plastic	McMaster Carr	1	yes	yes	15.89
cap screw	MSC	1	yes	no	21.80
motor controller	RoboteQ	1	yes	no	495.00
USB to Serial	ByteRunner Technologies	3	yes	no	57.45
Webcam	Microsoft	3	yes	no	3 x 60.25
LIDAR	Sick Optics	1	no	no	donated
Encoders		2	no	no	donated
Compass	Phhidgets Inc.	1	yes	no	150.00
GPS	Raven	1	no	no	donated
Laptop	HP	1	yes	no	463.99
encoder module	Robotshop	1	yes	no	125.00

TOTAL COST: $2806.98

7 Conclusion

PROJECT 14 was designed, built, and tested to compete in the 2011 Intelligent Ground Vehicle Competition. Physically, our robot was designed to be robust, and yet also mobile enough to be able to navigate around tight corners that may be present during the competition. Sensors were implemented to take the environment and turn it into a series of digital signals that the robot could use to find and negotiate a path through obstacles while staying within line boundaries. Through IGVC we have gained knowledge on the challenges and awards that come from working on a problem as complex as navigation of an autonomous vehicle.