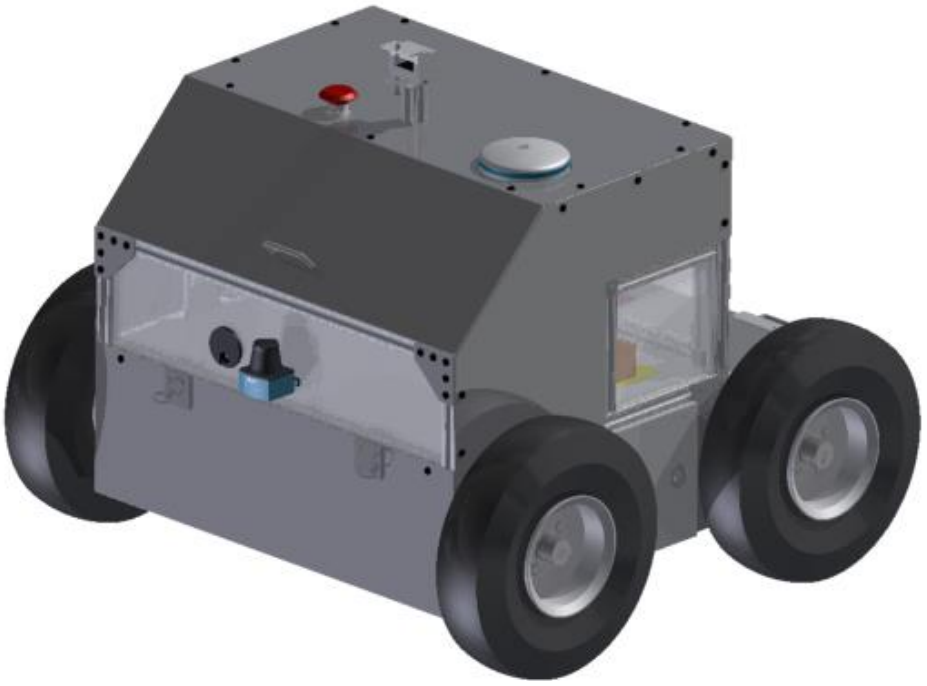
UIC COLLEGE OF ENGINEERING
UNIVERSITY OF ILLINOIS AT CHICAGO

EDT-Scipio



Chicago Engineering Design Team (EDT)

The engineering design documented in this report and implemented into this vehicle by the current student team is significant and equivalent to the credits that would be awarded in a senior design course.

Signed,

____, **Dr. Miloš Žefran,** Faculty Advisor

1. The IGVC Team

The Chicago Engineering Design Team consists of over 40 members, with the most experienced members being part of EDT's IGVC team. Figure 1 shows how the team was organized and how responsibilities were distributed. The team was divided into 3 departments: mechanical, electrical, and software. A captain was assigned to lead each department while the president and vice president were in charge of managing the three departments. Since August 2012, it is estimated that the team members invested more than 2000 man-hours in order to design and fabricate entirely new mechanical and electrical systems while greatly improving and testing the software system.

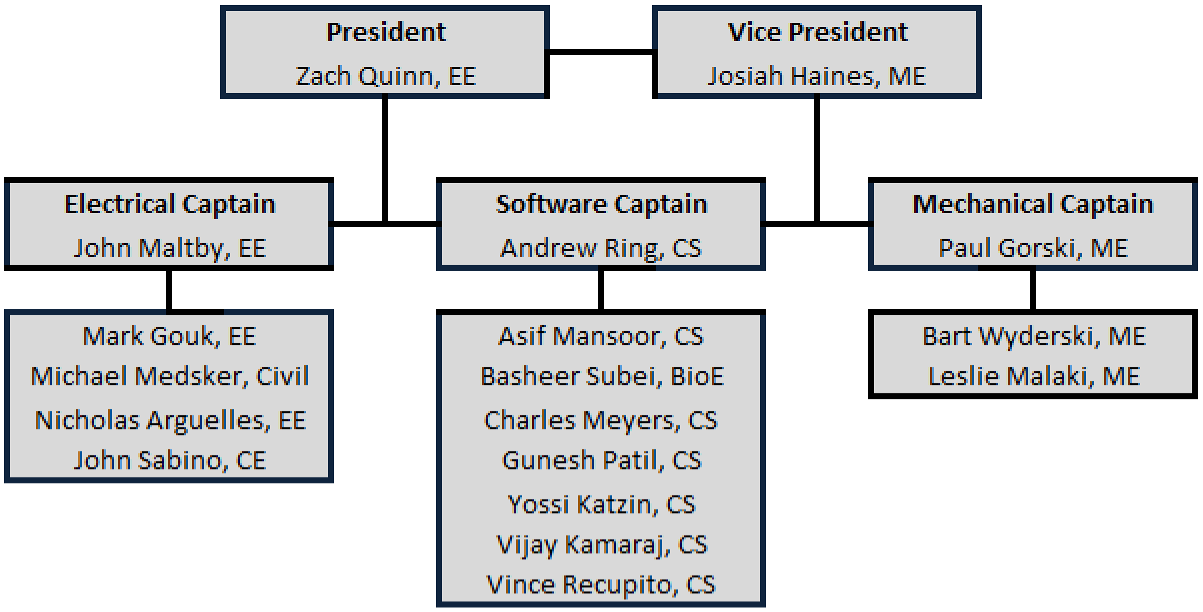


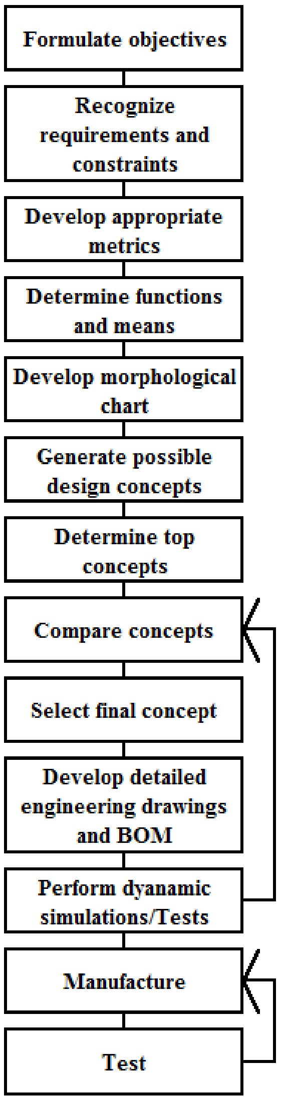
Figure 1: IGVC Team Organization

2. Design Process

Due to poor performance at the 2012 IGVC, it was made clear that the team would need to invest in a new design in order to effectively compete in 2013. The old design required frequent maintenance and the electronics could easily be wired incorrectly. Therefore, the team discussed every portion of IGVC in detail to develop a first class design. Many of the steps of the design process were borrowed from the process taught in the UIC Mechanical Engineering course called *Engineering Design and Graphics.* A representation of this process can be seen in Figure 2.

The design process began by first understanding the design problem, and then formulating design objectives. After the problem was well defined and the objectives were formulated, the constraints and requirements limiting the design were recognized. The constraints included competition rules such as vehicle size, vehicle speed, and safety regulations and also internal constraints such as cost, resources, and manufacturing capabilities. In order to measure how well an objective was met, metrics were developed in order to score different aspects of the design. Metrics pertaining to the higher level objectives were weighed heavier than those pertaining to lower level objectives. In our case, the metrics carrying the most weight were cost and manufacturing capabilities. The metrics were later used in the conceptual design process to compare various design concepts against each other.

The conceptual design process began by determining all necessary functions the vehicle needed to perform. Next, all possible means to fulfill the functions were determined and inserted into a morphological chart in order to generate design concepts. A sample of the morphological chart can be seen in Figure 3. Concepts were generated from the morphological chart by making various combinations of the means. Many concepts were eliminated due to failure to satisfy the design constraints. The overall concepts list was then reduced to four top concepts which were further analyzed and compared against each other. Generic drawings and sketches were made for each concept and the overall cost and manufacturability were estimated. A comparison chart was developed and the metrics were used to score the predicted performance of each concept. The method of assigning a score to objectives involving performance was done by researching the concept and making an educated estimation. After compiling the overall scores of each concept, the concept that received the highest score was chosen as the final concept. Until the final concept was selected, all three departments worked together; however, once a final concept was selected, the departments branched out to work on the detailed design section for their respective departments.


Formulate objectives
Recognize requirements and constraints
Develop appropriate metrics
Determine functions and means
Develop morphological chart
Generate possible design concepts
Determine top concepts
Compare concepts
Select final concept
Develop detailed engineering drawings and BOM
Perform dyanamic simulations/Tests
Manufacture
Test
Figure 2: Design Process

The detailed design section consisted of, but was not limited to, developing engineering drawings, schematics, CAD models, and a bill of materials. Once the CAD models were completed, dynamic simulations and testing could be performed (where applicable). Finally, parts were ordered and manufacturing could begin. During the manufacturing process, each department performed its own independent tests of components and systems. Once the mechanical and electrical systems were manufactured, integrated testing of all 3 systems was

Functions/ Features	Mean 1	Mean 2	Mean 3	Mean 4
Translate motor rotation to wheels	Chain and Sprockets	Drive Shaft	Spur Gears	Belt and Sprockets
Detect objects	Laser Range Finder	Camera	Sonar	Infrared
Power Source	Lithium Ion Batteries	Lead Acid Batteries	Generator	Solar

Figure 3: Morphological Chart

performed. When problems were encountered, improvements were suggested and implemented where required.

3. Mechanical Design

Scipio was designed to be a reliable and stable platform so that the mechanical structure could be re-used in future years. The entire drivetrain is considered a mechanical innovation, as it is different from any prior EDT drivetrain. Scipio is split into two main sub-assemblies, the top chassis assembly and bottom chassis assembly, which are described in detail below.

3.1. Bottom Chassis Assembly

The bottom chassis was constructed of a steel tube frame, which houses the drivetrain and its components. The drivetrain is powered by two 3HP brushed DC motors, operating at 24 volts, and is a skid-steer system. The skid-steer system allows Scipio to have a zero-turn radius, which is optimal for switchbacks and dead ends. The drivetrain is comprised of two in-house manufactured gearboxes that drive a power transmission belt system. The gearbox and belt system achieves a speed reduction of 12.8 and 2.39 respectively, providing a total speed reduction of 30.63:1. Both of these systems are designed and manufactured within tolerances to achieve efficiencies greater than 93%.

The motivation behind this drivetrain was to increase the power and efficiency while reducing the amount of maintenance required. EDT robots in the past, which were chain driven, suffered performance issues and required great amounts of maintenance in order to compensate for backlash. In particular, chain based drivetrains struggled to turn the robots easily and steadily. By significantly increasing the drivetrain power from a 17.5 to 30.63 speed reduction and establishing the proper ratio between wheel centers, the robot was easily able to overcome the effects of the turning scrub. Maintenance was greatly reduced due to the strength of the belts and overall structure of the drivetrain.

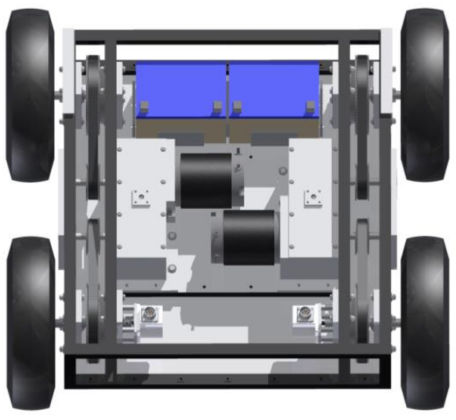


Figure 4: Bottom Chassis Assembly

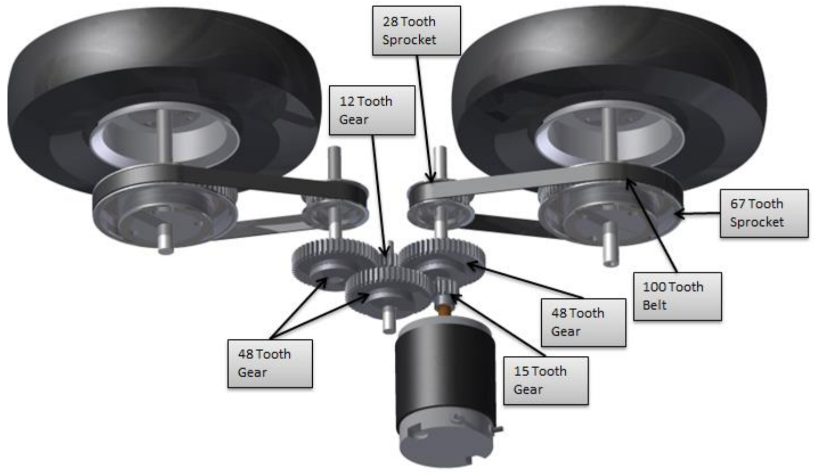


Figure 5: Drivetrain Skeleton

3.2. Top Chassis Assembly

The top chassis was designed to be modular and accessible while maintaining a clean and professional aesthetic. The top chassis stores the logic circuits, payload, and the sensors including the GPS, Laser Range Finder, webcam, and compass. The top chassis is divided into three compartments: the electrical box, laptop area, and housing area. Each area can be accessed by either a drawer or a door. The laptop platform slides outward on a drawer, while the electrical box and housing area have hinged doors for access. Aluminum T-slotted framing was used as the base structure for the top chassis, which allows for easy assembly and future modifications. Aluminum panels with rubber edge-grip seals were fastened to the T-slotted frame in order to provide protection and weather-proofing. Mating to the bottom chassis was achieved by utilizing four quick-release pins, allowing easy separation from the bottom chassis for troubleshooting or maintenance.

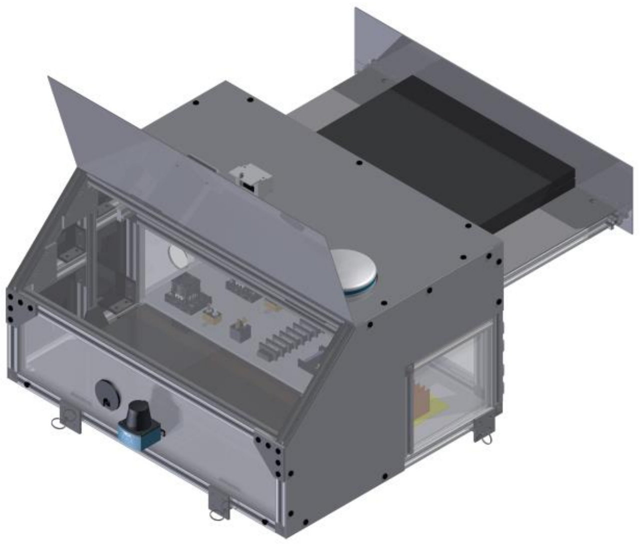


Figure 6: Top Chassis Assembly (Open)

4. Electrical Design

Scipio represents a leap forward for EDT's approach to electrical integration. The flow for the electrical system starts at the computer. The computer is fed data from a compass, GPS, laser range finder, and two incremental shaft encoders. With this data, the computer determines the navigation path and sends data to the EDCS. The Signal Multiplexer is used to switch between autonomous and remote control, as detailed below. If an invalid signal is received by the Signal Multiplexer, the robot will not be allowed to move. If a valid signal is received, the motor controllers are sent a power level and movement begins. A set of bright LEDs will flash when in autonomous mode, but will change to solid when in remote control mode. If at any point there is an emergency, pressing either the onboard E-Stop switch or the wireless E-Stop switch will disable the robot. For the first time, EDT has designed circuits as "cards" (rather than boards), which can be plugged into a

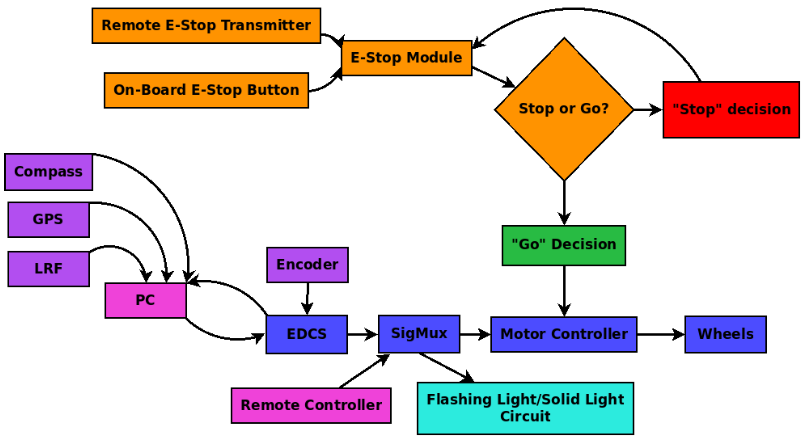


Figure 7: Electrical Flowchart

socket. This replaced the old method of wiring up boards individually, leading to the possibility of making incorrect connections. In addition, having all of the circuitry on a single logical unit greatly reduced the amount of space and wiring required for the electronics. Each of the embedded systems will enter a sleep mode when idle in order to conserve power.

4.1. Power System

Scipio is composed of a 24 volt electrical system, created by two 12 volt, 35 amp-hour, sealed lead acid batteries in series. This configuration yields 90 minutes of runtime. As in previous years, two Victor 885 motor controllers were used to throttle the power delivered to the motors. Switching regulators were used to power the sensors and the logic circuits, and were chosen for power efficiency. Moving from a desktop computer to a laptop allowed the DC-AC inverter to be removed, as well as several batteries.

4.2. Emergency Stop (E-Stop)

The Emergency Stop is designed to disable the vehicle whenever an emergency situation occurs. The E-Stop may be activated wirelessly by remote control or manually by pressing the onboard switch. The E-stop is composed of two modules: a hand held transmitter unit that wirelessly transmits a signal to the vehicle and a receiver unit which is on the vehicle. The transmitter unit has a highly visible red pushbutton switch which changes from “GO” command to “STOP” command when depressed. A “GO” signal must be received from both the onboard switch and the wireless transmitter before the vehicle will be allowed to move. If no signal is received from the wireless transmitter, the vehicle will remain stopped for safety reasons. The radio module has a range of 5 miles and uses spread spectrum technology to ensure data encryption and prevent interference or jamming from external sources, thus improving system reliability.

4.3. Embedded Drivetrain Control System (EDCS)

The EDCS is designed to interface the software controls system to the drive motors. The system consists of a module for each side of the vehicle, requiring two EDCS’s per vehicle. Each module is responsible for measuring the wheel speed and controlling the power level of the motor for its respective side. A microcontroller is the brain of each module and communicates with the software control system via RS-232 serial communication. The communications protocol between the EDCS and the computer is designed to inherently detect communication errors, leading to improved system reliability and safer operation. From a navigation standpoint, this is one of the most important systems in the robot. By checking for wheel slippage, the computer can compensate for erroneous wheel encoder readings, which aids in performing accurate localization functions.

4.4. Signal Multiplexer (SigMux)

The Signal Multiplexer allows the user to choose the manner in which Scipio is controlled. The system has four modes of operation that are easily controlled by the user through pushbutton switches located on the left side of the robot. The first mode is called safety, which causes the vehicle to remain stationary. This is the default mode, ensuring safety by not allowing the vehicle to move until a user deliberately specifies a desired mode. The second mode allows the vehicle to be exclusively controlled autonomously. The third mode allows the vehicle to be exclusively controlled by remote control. The final mode allows the vehicle to switch between autonomous and radio control. This is determined by a toggle switch on the remote control. If the robot is in remote control mode, the system ensures that a valid signal from the RC controller is present, but will put the vehicle in safety mode if there is an erroneous or absent signal.

4.5. Flashing Light Circuit

The flashing light circuit is designed to alert nearby people of the robot and indicate the mode of operation. There are four high output LEDs, one facing each side of the robot, mounted under the webcam. The flashing light circuit receives a signal from the SigMux and places itself in one of two modes. When autonomous mode is engaged by the SigMux, the LEDs blink with 1 Hz frequency, giving people adequate warning to stay clear of the robot. When not in autonomous mode, the LEDs are on but do not flash.

4.6. Sensors

Scipio has several sensors that provide data feedback to the computer software. Two incremental shaft wheel encoders are coupled directly to the front wheels to measure the exact rotational position and velocity of each side of the vehicle. Each encoder produces two pulse trains with frequencies linearly dependent on wheel speed. The encoder sends the two pulse trains to the EDCS, which then determine the wheel speed and direction of rotation. A weatherproof GPS receiver is used to help with goal planning and allows the JAUS protocol to be implemented. A new addition this year is a laser range finder mounted on the front of the robot. The laser range finder is used for object detection and replaces a previously less accurate system of sonars and a stereoscopic disparity camera. Moving from a disparity camera to a normal webcam reduced computational demands and therefore reduced the required amount of on board batteries.

4.7. Computer

This year, EDT switched the computing platform to a laptop, which allowed for a dramatic decrease in the number of batteries required on board and eliminated the need for a DC-AC inverter. A durable laptop, meant to withstand vibration and temperature shock, was selected. Replacing the mechanical drive with a solid state drive provided more resistance to vibration. The computer contains an NVidia Quadro K3000M graphics card, which allows CUDA to be utilized. CUDA is a set of API’s that allows code, such as line detection, to be quickly executed on the GPU. A secondary battery was also purchased, giving 185 watt hours of power – good for an average of 4 hours of runtime.

4.8. Electrical Innovations

There are three main electrical innovations from previous years. In Scipio, all of the major electrical components are placed together into a single, easily accessible compartment, which is beneficial from both an installation and diagnostic standpoint. For example, moving the motor controllers into the electrical compartment has greatly simplified controller calibration for the end user, and has greatly simplified the replacement process.

Another innovation was to convert all of the circuits into a motherboard-daughterboard configuration. The motherboard has seats for all five daughter cards, which can be replaced in seconds in the event of a failure. All communication between systems is accomplished through the backplane, which is completely passive. This system has greatly reduced the space required for wires and electrical components. The backplane also has programming ports so firmware can be updated

Lastly, the plan for object detection was changed from analyzing a stereoscopic image to interpreting data from a laser range finder. This allowed for a decrease in computational requirements, and therefore battery requirements. The laser range finder is also inherently more precise than image analysis because data is sent as (x,y) coordinates as opposed to being deriving (x,y) coordinates from a 3-dimensional image.

5. Software Design

Scipio runs an in house intelligent agent named Deimos. Deimos is designed to be a resilient system which plans its action based on prior experiences while maintaining a robust response to perceived changes to the surrounding environment.

To accomplish the objectives for the 2013 IGVC, the following major systems have been created:

- Mapping – Deimos will record and recall obstacles previously viewed, including physical obstacles as well as lines and flags.
- Navigation – Using the recorded map data, the shortest path to the desired destination will be found and followed.
- Wheel Controller – The speed of the robot should be maintained at the highest rate allowed by the input processing speed, while following both physical constraints and the IGVC rules.
- Sensors – Sensory data is interpreted and passed to the map in an input independent manor, decoupling the input device from logical representation.

- JAUS – Remote monitoring and interaction between robot and external system, using the JAUS protocol.

5.1. Software Architecture

Deimos is a multi-threaded system. There are two primary states the system can be in: stationary/routing in which a new path is derived, and execution, in which a set path is followed. Execution runs until either the path has been followed to its end (GPS waypoint), or an unexpected obstacle is in the path. In either case, the state returns to stationary/routing (after stopping in the case of an obstacle). The following diagram shows the high level module interaction:

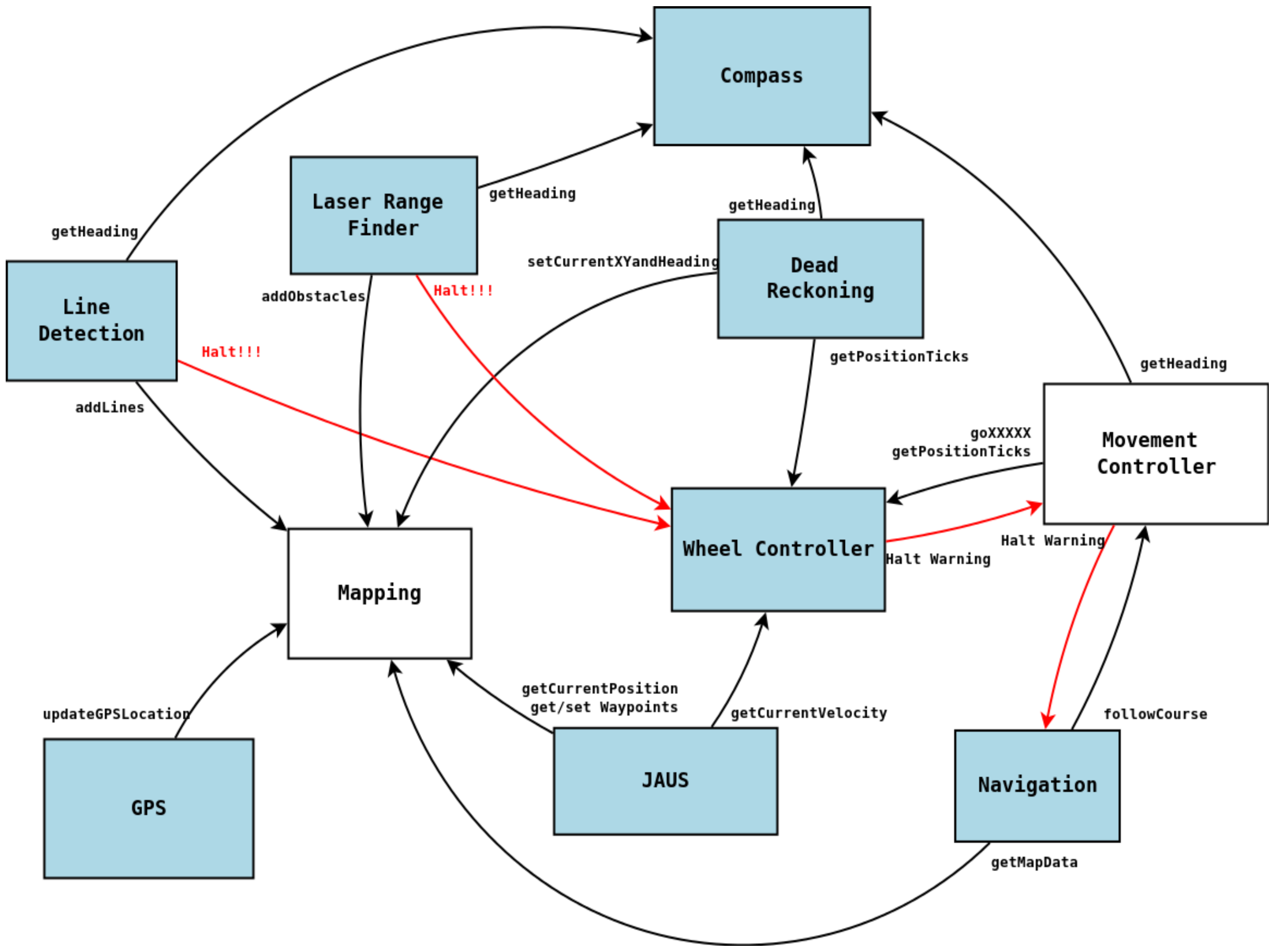


Figure 8: Deimos module and thread interaction diagram. Shaded boxes are executed on their own threads. Red lines trace a halt event's callback chain.

5.2. Languages and Libraries

Deimos is written primarily in Java, with some external C++ libraries in use. Connection between the Java and C++ code is done using JNA. A few open source libraries are used, including JAUS++, jSSC (serial communication), jUnit (framework testing), and OpenCV (CUDA processing).

5.3. Movement

A wheel control system was introduced into this year's design with the goal of enabling the movement of the robot to match specified velocities, and to accelerate and decelerate smoothly with minimal drift due to differences in the left and right motors. In order to achieve this goal, several possible methods of implementing a control system were examined. Initially, implementing a PID controller was considered; however, research led to the consideration of a fuzzy logic wheel controller. While building such a system, it was discovered that the additional overhead in translation to a fuzzy set mitigated the benefit of pre-calculating values. As such, a direct adjustment has to be performed in the final system.

The direct adjustment is in two parts, and uses the following formulas. Let v_{MAX}be the maximum velocity. v_L, which is a configurable value, is used to ensure sensory data has sufficient time to be collected and processed, so that the robot can respond to obstacles before intersecting them. Let v_L and v_R be the left and right velocities, respectively, and ρ be an acceleration factor tuned to the specific hardware. Let Δ_{vL} and Δ_{vR} be the difference between desired velocity and observed velocity for left and right side, respectively. Let $\varphi_{current}$ be the current motor set point and φ_{next} be the next motor set point. Let σ be a syncing factor, tuned to the specific hardware. Then:

$$\varphi_{next_L} = \varphi_{current_L} + \frac{\Delta_{vL}}{v_{MAX} \times \rho} \quad and \quad \varphi_{next_R} = \varphi_{current_R} + \frac{\Delta_{vR}}{v_{MAX} \times \rho}$$

Followed by the syncing:

$$\varphi_{next_L} = \varphi_{next_L}(+/-)\big||v_L| - |v_R|\big| \times \sigma \quad and \quad \varphi_{next_R} = \varphi_{next_R}(+/-)\big||v_L| - |v_R|\big| \times \sigma$$

5.4. Constraints

The control system runs at a software level as part of the agent, and not on a microcontroller. The decision to implement the control system at this level was multipart. First, the EDCS, which interfaces with the encoder and motor for the wheel, was implemented as an independent per wheel device. This existing implementation did not provide an interconnection between the two wheel controllers, and so it could not easily provide synchronization in wheel speeds to reduce drift while attempting straight line movement. Therefore, synchronization had to be done at a higher level.

5.5. Sensory

5.5.1. Laser Range Finder

A laser range finder, capable of 1mm precision, 1° angular resolution, and accurate up to 4 meters outdoors, was used for object detection on this year's robot. The data from the scans is read and sent to the Mapping module to record the presence of objects within close proximity of the robot for future navigation.

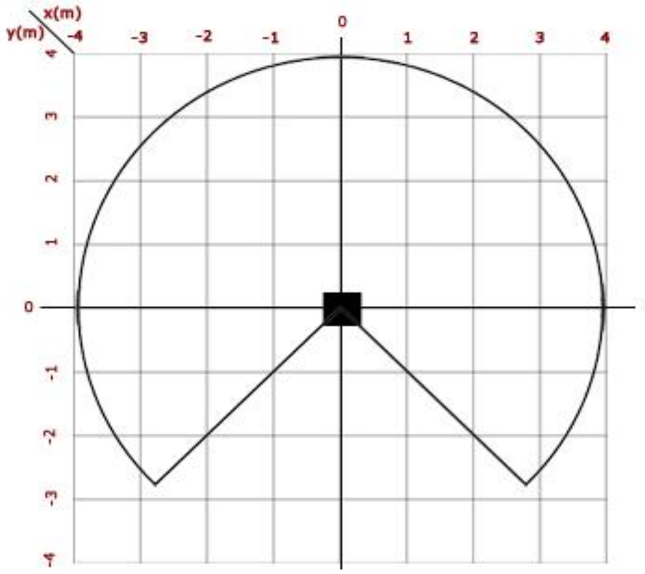


Figure 9: Laser Range Finder - Viewing Range

5.5.2. GPS/Compass

The data loaded from the GPS is accurate to within 0.6m. Upon polling current location by GPS, the conversion equation between GPS waypoints and map coordinates is recalculated. This means that if any slippage occurs when tracking Scipio's location in the map, the GPS waypoint location is self-corrected.

5.6. Line Detection

Line detection is performed using images captured from a HD webcam. Polling of the image is performed in C++, and the resulting image is processed using OpenCV's CUDA library, using the Hough Lines algorithm. The image coordinates are then mapped into a 3-dimensional real coordinate space. This can be done geometrically given the height and angle of the camera. Currently, it is assumed that the robot is level and that the ground in front of it is a level plane.

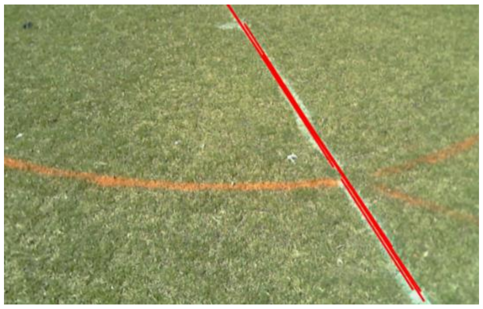


Figure 10: Line detection example

5.7. Dead Reckoning

Dead reckoning is used to maintain a knowledge of the current location of the robot in map-space. While this type of system typically suffers from slight drift over time, this is an expected characteristic and is countered by the self-correcting nature of the mapping design. Due to the inherent slippage of wheels while rotating in place with a differential drive, the compass is used to provide heading information. Distance traveled is calculated as follows.

Let R_w be the radius of the wheels, θ the angle of heading from the compass, T_L and T_R be the number of pulses counted by the encoders on the left and right wheels, respectively, and ρ be the number of pulses per revolution. The equations to calculate Δx and Δy, the changes to the robots location in map-space since the last update are:

$$\Delta x = R_w \cos(\theta)\,(T_L + T_R)\frac{\pi}{\rho} \quad and \quad \Delta y = R_w \sin(\theta)\,(T_L + T_R)\frac{\pi}{\rho}$$

The updates are incrementally passed to the Mapping module, which maintains the current location in map-space of Scipio.

5.8. Mapping

The mapping system in Deimos uses the blackboard model. In the blackboard model, various systems post and receive information to a central repository or "blackboard." In Deimos, the Mapping module serves as this blackboard. Various threads add data to the map as it is generated and current data is polled from the map as needed. The map holds the following information:

- Object and line locations in a coordinate space defined with respect to the location of the robot when Deimos is initialized.
- Locations in the map not yet viewed are marked as such to allow navigation processing optimizations
- Current location of the robot in map-space
- List of waypoints in GPS coordinates. These are mapped to map-space coordinates when requested.

5.9. Coordinate Spaces

The map-space functions at two logical levels. On the surface, map coordinates are (x,y) pairs where x is the number of meters north of the origin (location of initialization), and y is the number of meters east of the origin. This allows all coordinate passing to be in a standardized format decoupled from the data structure used to store the map. Internally a map exists as a two dimensional array, of which each value stores the current best known state information for the block of space it represents. The array data can represent lines, probability of an objects presence (low, medium, or high), flags, empty space, unknown, etc.

Translation between coordinate spaces functions as follows. Let ω be the number of indices per meter (the resolution of the map). Let x_{MAX} and y_{MAX} be the maximum distances from the origin point that the map will cover. Let x_m and y_m be coordinates in map-space (meters). Let x_i and y_i be coordinates in internal structure coordinates.

$$x_i(x_m) = \omega(x_m + x_{MAX}) \quad and \quad y_i(y_m) = \omega(y_m + y_{MAX})$$

$$x_m(x_i) = \frac{x_i}{\omega} - x_{MAX} \quad and \quad y_m(y_i) = \frac{y_i}{\omega} - y_{MAX}$$

5.10. Robot Point Location

To increase the efficiency of the navigation search algorithm, the map treats the robot as a point particle. To achieve this without causing collisions with obstacles, each object in the map casts a "shadow" around its true location. This shadow effect causes objects on the map to effectively swell in size to compensate for the size of the robot.

5.11. Self-Correction

A key feature of the map is that it is self-correcting. To achieve this, when an area is viewed multiple times, such as by the laser range finder, all values which were previously set by the laser range finder within the viewing area are first decreased in confidence (for example from high to medium). Next, the new values are placed, with shadow, at high confidence. This causes values to diminish if the area is viewed again and the object is not in the same location. This can result from obstacles (such as people) moving or through the expected drift of accuracy of current location in map-space. When revisiting an area, if drift has occurred, the objects will degrade and be replaced by their corrected location points.

If objects are not located close to each other, a distortion effect may be created due to drift; however, a path through the area will be maintained (if it exists in real space). The exact distances and angles of the path may be likewise distorted, so as the path is followed it will be corrected with respect to the drift, maintaining the ability to route through distorted areas.

5.12. Navigation

Navigation checks Mapping for current GPS waypoints. If GPS waypoints are found, it attempts to route to one. To achieve this, the internal map grid is viewed and a modified breadth first search is performed. If points are encountered which are unknown (not previously visited), the search does not extend into that space. Instead, a straight line is substituted to the destination (or the closest known point on the line). Once a path to the target has been identified, it may contain "jagged" lines if a diagonal path should be followed.

To correct for these "jagged" paths and reduce the overall time to navigate, the path is smoothed. Smoothing is accomplished by moving each vertex on the path closer to its neighbors along the path, and at each step checking if the resulting new path segment encounters an obstacle or line. This is repeated along the length of the path until a series of straight line segments are formed (along any angle). This path is then passed to the movement controller, a module which attempts to follow a given path until the end is reached or an obstacle is unexpectedly found in the path.

By following this method, the navigation is generalized, which removes the need for specialized lane following. Spaces containing lines or obstacles are both treated as not crossable, i.e. occupied.

5.13. JAUS

Initially, JAUS implementation used OpenJAUS. After a significant investment in time and submitting many bug fixes for this framework, a switch was made to Jaus++. Jaus++ provides a more reliable framework for the code to run on. Both systems are in C++, and connection to the core Java Deimos code is made using JNA.

5.14. Software Innovations

Deimos represents a significant advancement in intelligent agent development by Chicago EDT compared to prior years. This includes the following:

- A control system has been added to allow smooth movement, with wheel synchronization to avoid rotational drift.
- A map is now maintained which differentiates between types of objects, allowing for different update policies by type of obstruction (ex. physical object vs. painted line).
- Navigational routing intelligently uses map data to avoid unnecessary routing computation by recognizing areas which have not yet been seen.
- JAUS is now implemented for the first time at Chicago EDT.
- The use of a laser range finder allows for significantly more accurate object detection, and reduces the computational cost associated with cleaning up noise in a disparity image, as well as image processing in general.

6. System Safety

Ensuring that the vehicle was safe was of high importance. The key features that ensure that the vehicle is safe for all users and observers are listed below.

- All exposed edges and corners have been smoothed to prevent injury
- All drivetrain components are properly enclosed and protected
- A flashing light alerts bystanders
- The SigMux checks for valid signals to ensure no random or spontaneous movement when in RC mode
- If no wireless E-Stop signal is detected in autonomous mode, the vehicle is automatically deactivated
- Speed is mechanically limited to 5.98 mph at 24 Volts

7. Performance Analysis

The performance analysis section includes the predicted performance of the vehicle versus the actual performance of the vehicle when tested.

7.1. Vehicle speed

With 14 inch diameter wheels, a speed reduction of 30.63:1, and motors with an output of 4400 rpm at 24 volts, the maximum theoretical speed of Scipio is 5.98 mph (assuming no losses due to loading). Testing showed an average maximum speed of 5.59 mph, which is within 6.6% of the calculated speed.

7.2. Ramp climbing ability

Due to Scipio's low center of mass and powerful drivetrain, it was expected that it could climb an incline of 40 degrees with ease. Testing showed that Scipio was able to easily climb a 45 degree incline, which is significantly greater than any incline on the IGVC course.

7.3. Reaction times

Scipio's reaction time is limited by a combination of collecting sensory data and then processing the data. The EDCS's are able to accept a new power level every 50 ms. Line detection, on average, takes less than 10ms, and the laser range finder data is read every 20 ms. The map can be updated with laser range finder data almost instantly. Therefore, the worst case scenario is when the EDCS's respond to data from the laser range finder, giving a 70ms reaction time.

7.4. Battery life

Scipio can operate for a maximum of 90 minutes before the drive motor batteries must be replaced. The laptop battery lasts an average of 4 hours. The goal was to achieve 2 hours of runtime.

7.5. Distance at which obstacles are detected

According to the data sheet for the SICK TIM-310 laser range finder, objects can be detected at distances up to 4 meters indoors and 3+ meters outdoors. Testing revealed that the laser range finder was able to detect objects as distant as 3.96 meters in direct sunlight, exceeding the expectations.

7.6. Dealing with complex obstacles

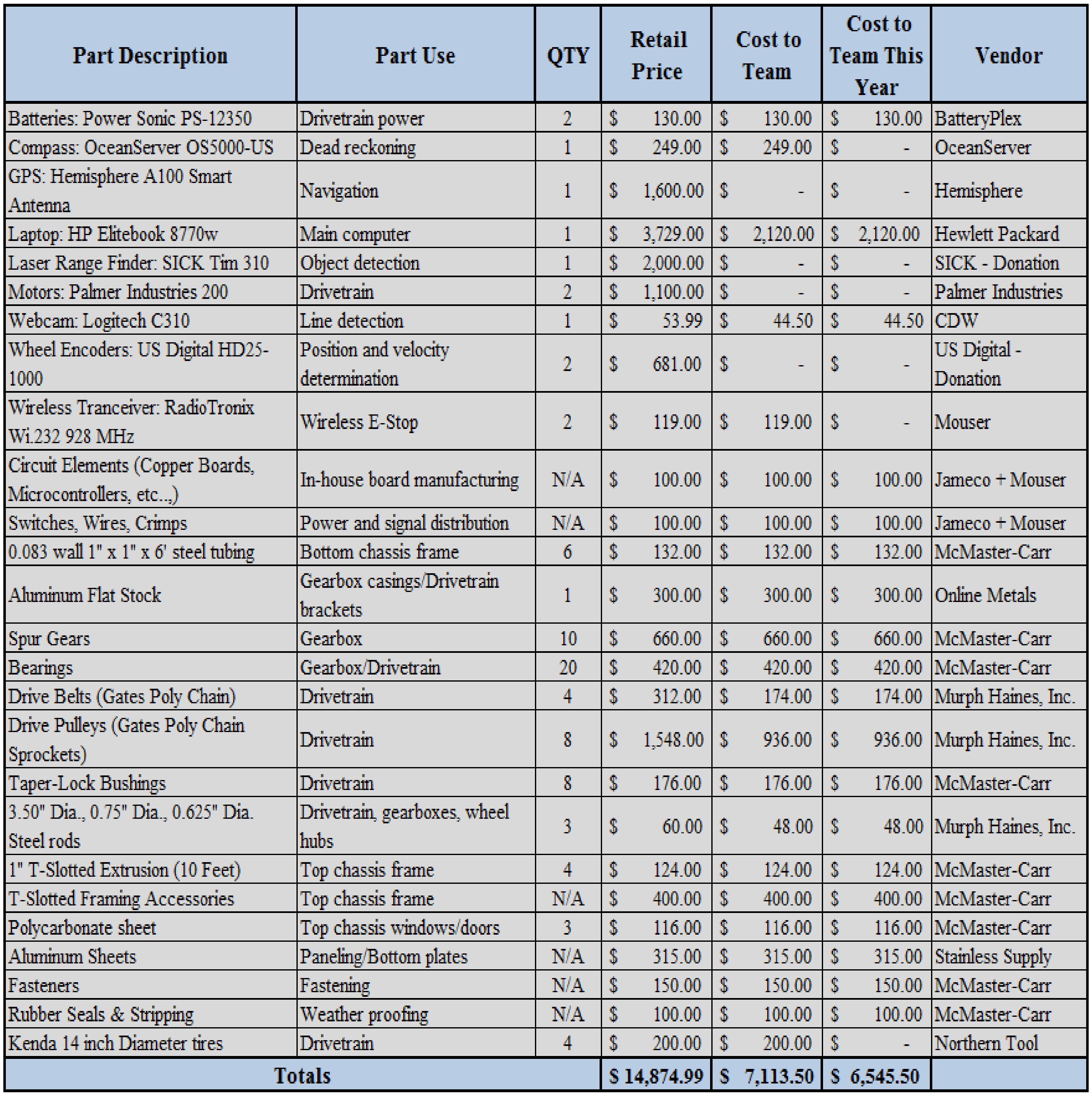
Due to the generalized mapping and navigation system employed by Deimos, complex obstacles, such as switch backs, are handled intrinsically by the routing algorithm. This is an improvement over systems such as line following, which require additional logic to handle these special cases. Being able to turn on point allows Scipio to easily follow the commands to maneuver around complex obstacles.

7.7. Positional accuracy

The accuracy of the Hemisphere GPS is +/- 0.6 meters. The software team has programmed the robot to navigate to within 1 meter of a waypoint. With these two factors, the expected accuracy of arrival at navigation waypoints is 0 to 1.6 meters.

8. Bill of Materials

The Bill of Materials section includes a breakdown of the all the major components and parts used to manufacture Scipio along with cost and vendor information.



Part Description	Part Use	QTY	Retail Price	Cost to Team	Cost to Team This Year	Vendor
Batteries: Power Sonic PS-12350	Drivetrain power	2	$ 130.00	$ 130.00	$ 130.00	BatteryPlex
Compass: OceanServer OS5000-US	Dead reckoning	1	$ 249.00	$ 249.00	$ -	OceanServer
GPS: Hemisphere A100 Smart Antenna	Navigation	1	$ 1,600.00	$ -	$ -	Hemisphere
Laptop: HP Elitebook 8770w	Main computer	1	$ 3,729.00	$ 2,120.00	$ 2,120.00	Hewlett Packard
Laser Range Finder: SICK Tim 310	Object detection	1	$ 2,000.00	$ -	$ -	SICK - Donation
Motors: Palmer Industries 200	Drivetrain	2	$ 1,100.00	$ -	$ -	Palmer Industries
Webcam: Logitech C310	Line detection	1	$ 53.99	$ 44.50	$ 44.50	CDW
Wheel Encoders: US Digital HD25-1000	Position and velocity determination	2	$ 681.00	$ -	$ -	US Digital - Donation
Wireless Tranceiver: RadioTronix Wi.232 928 MHz	Wireless E-Stop	2	$ 119.00	$ 119.00	$ -	Mouser
Circuit Elements (Copper Boards, Microcontrollers, etc..,)	In-house board manufacturing	N/A	$ 100.00	$ 100.00	$ 100.00	Jameco + Mouser
Switches, Wires, Crimps	Power and signal distribution	N/A	$ 100.00	$ 100.00	$ 100.00	Jameco + Mouser
0.083 wall 1" x 1" x 6' steel tubing	Bottom chassis frame	6	$ 132.00	$ 132.00	$ 132.00	McMaster-Carr
Aluminum Flat Stock	Gearbox casings/Drivetrain brackets	1	$ 300.00	$ 300.00	$ 300.00	Online Metals
Spur Gears	Gearbox	10	$ 660.00	$ 660.00	$ 660.00	McMaster-Carr
Bearings	Gearbox/Drivetrain	20	$ 420.00	$ 420.00	$ 420.00	McMaster-Carr
Drive Belts (Gates Poly Chain)	Drivetrain	4	$ 312.00	$ 174.00	$ 174.00	Murph Haines, Inc.
Drive Pulleys (Gates Poly Chain Sprockets)	Drivetrain	8	$ 1,548.00	$ 936.00	$ 936.00	Murph Haines, Inc.
Taper-Lock Bushings	Drivetrain	8	$ 176.00	$ 176.00	$ 176.00	McMaster-Carr
3.50" Dia., 0.75" Dia., 0.625" Dia. Steel rods	Drivetrain, gearboxes, wheel hubs	3	$ 60.00	$ 48.00	$ 48.00	Murph Haines, Inc.
1" T-Slotted Extrusion (10 Feet)	Top chassis frame	4	$ 124.00	$ 124.00	$ 124.00	McMaster-Carr
T-Slotted Framing Accessories	Top chassis frame	N/A	$ 400.00	$ 400.00	$ 400.00	McMaster-Carr
Polycarbonate sheet	Top chassis windows/doors	3	$ 116.00	$ 116.00	$ 116.00	McMaster-Carr
Aluminum Sheets	Paneling/Bottom plates	N/A	$ 315.00	$ 315.00	$ 315.00	Stainless Supply
Fasteners	Fastening	N/A	$ 150.00	$ 150.00	$ 150.00	McMaster-Carr
Rubber Seals & Stripping	Weather proofing	N/A	$ 100.00	$ 100.00	$ 100.00	McMaster-Carr
Kenda 14 inch Diameter tires	Drivetrain	4	$ 200.00	$ 200.00	$ -	Northern Tool
Totals			**$ 14,874.99**	**$ 7,113.50**	**$ 6,545.50**	

Figure 11: Bill of Materials