Dia-Bot: Installation Diagnostic Robot

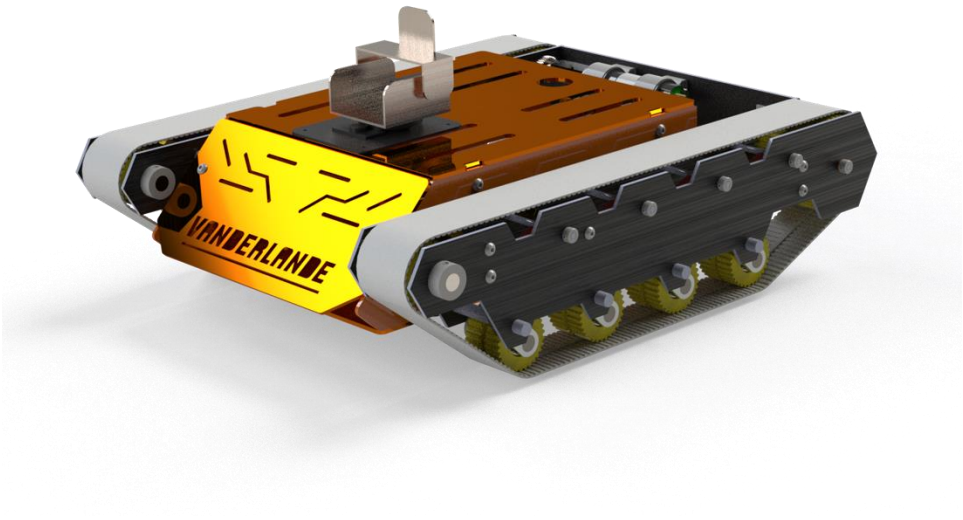


Figure 1: CAD Rendering of Dia-Bot Assembly

Interdisciplinary Capstone Final Design Report

Course Numbers: ME 4182-A, ECE 4723-A

Instructor: Dr. Amit Jariwala

Advisors: Jianxin Jiao (ME), Vijay Madisetti (ECE)

Georgia Institute of Technology

Atlanta, Georgia

Sponsor: Vanderlande Industries

Marietta, GA; Veghel, NL

Arlo.Bromley@vanderlande.com

+1 470 541 2998

Team: Operation Omega

Members: Andrew Galant, Catherine Kasper, Jason Piotter, Hunter Present, Connor Truono, Doug Walker

Primary Editors: Connor Truono, Hunter Present

Table of Contents

Executive Summary

Vanderlande produces logistic process automation for warehouses, parcels, and airports, focusing on shuttle and conveyor systems. However, when installing such large systems, the process for verifying their structure, components, and throughput can take many hours to complete manually. Because of this, Vanderlande desires a more powerful tool by which they can apply more automated and manual solutions: specifically, a diagnostic robot which moves through their setups to assist in problem detection and quality assurance.

To meet Vanderlande's needs for installation verification, the team design such a diagnostic robot, called the "Dia-Bot". This robot will aid Vanderlande by providing appropriate sensors and data collection, multiple transportation modes, and effective communication and user interface strategies to detect and flag common installation problems. This solution required solving a wide range of technical problems in mechanical, electrical, and software engineering disciplines. These include building a robust and reliable mechanical enclosure, creating a removable self-propelled movement system that can effectively traverse the obstacles of a conveyor systems, interfacing with various types of sensors, intelligently processing the sensor data, establishing real-time wireless communication channels, and exposing a proper user interface.

The objective of this Dia-Bot is to aid Vanderlande operators in identifying potential problems during a phase of their system installation. The Dia-Bot should collect various data points which may be indicative of issues: most notably visuals, acceleration, and sound, as well as a mechanism to report robot position within a given system. Real-time bot controls and sensor updates would allow operators to find and fix errors more efficiently.

Multiple design and ideation strategies were used to help determine the project scope and evaluate proper engineering methods. A function tree identified the primary features of the Dia-Bot and broke them down into smaller requirements, and a house of quality helped evaluate the priority and importance of each requirement. The team then identified the best implementations for those function using a morphological chart. A Gantt chart helped the team plan out and follow a schedule of required tasks and keep the project on track.

The final design provides proper movement via continuous treads driven by DC motors as well as a flexible suspension system. These treads can be removed to expose a flat bottom surface for ride-along mode when self-propelled movement is not desired, allowing the bot to properly experience system forces. Between the two tracks, a dampened central roll cage hosts and protects the delicate

sensors and electrical parts while providing stability with a low center of gravity. An embedded processor interface with the sensors and motors to handle input and output signals as well as expose a real-time user interface over the web to receive robot control while sending a live data feed and alerts for any detected problems.

While some products exist with similar functionalities to those described, none fit the necessary criteria for collecting the right data with robust movement systems at the proper price point. Most modified RC cars would not be able to move through Vanderlande's conveyor systems, while more advanced industrial bots are overengineered for this purpose and are therefore overly expensive. The Dia-Bot aims for the middle ground price point while providing custom hardware and software features specific for Vanderlande's use cases.

The functional robot was prototyped to validate the Dia-Bot concept. This system can traverse Vanderlande's conveyors, allows easily removable treads to expose a flat surface, collect relevant data on vibration, sound, temperature, position, and visuals for problem detection, interface with a proper GUI for real-time operator control and data feed, and process data on-board to alert users about potential problems. Key performance indicators include control response time, problem alert accuracy, positional detection accuracy, movement range and battery life, and overall ease of use. Overall, the Dia-Bot should reduce the time and improve the quality of Vanderlande's installation verification inspections.

Now that the team has created a functional prototype to validate the concept, the project's next steps include field-testing the Dia-Bot in a Vanderlande system during installation. Additionally, other potential features outside the project scope could be investigated. These include autonomous driving and navigation, computer vision integration for visual problem detection, training machine learning models to identify additional patterns, and direct communication between multiple bots for swarm functionality. This Dia-Bot prototype provides Vanderlande the platform and possibility for greater operator access and data-driven intelligence to reduce time for system installation.

Nomenclature

A. Autonomous: Referring to that which is capable of operation without direct human control
B. Dia-Bot: Title of the design project, short for “diagnostic robot”.
C. Operator: Vanderlande Industries technician who controls the Dia-Bot in real time through the system
D. Vanderlande Industries: Developer of complex package transportation solutions for warehousing, parcel, and airport industries; Corporate sponsor of the Dia-Bot design project

Glossary

A. ADC: Analog-to-Digital Converter
B. ASTM: American Society for Testing and Materials
C. CAD: Computer – Aided Design
D. GPIO: General Purpose Input/Output
E. GUI: Graphical User Interface
F. FFT: Fast Fourier Transform
G. HoQ: House of Quality
H. I^2C: Inter-Integrated Circuit
I. IMU: Inertial Measurement Unit
J. MATLAB: Matrix Laboratory
K. NIOSH: National Institute for Occupational Safety & Health
L. OSHA: Occupation Safety & Health Administration
M. POC: Point of Contact
    a. Vanderlande POCs are Arlo Bromley and Dr. Patrick Opdenbosch
N. PWM: Pulse-Width Modulation
O. RC: Remote-Controlled
P. SPI: Serial Peripheral Interface
Q. VNC: Virtual Network Computer
R. UI: User Interface

Main Body

1. Introduction & Background

Vanderlande produces automated warehousing solutions of custom sizes for various companies, focusing on efficiently storing, transporting, and retrieving immense amounts of items via shuttle and conveyor systems [1]. One of their systems can be seen in **Figure 2**. During the installation of these large racking systems, there are a number of potential issues which may be found.

Figure 2: A Vanderlande automated warehouse

The most prevalent installation error pertains to incorrect assembly of the support profiles for an individual rack. An example of a correct as well as two common incorrect support profile installations can be seen in **Figure 3** below. Vanderlande estimates that in any given racking installation, 0.5-1.0% of individual racks will have incorrectly installed support profiles. Each warehouse can easily have over 100,000 racks, and each error in racking installation can take around 5 minutes to locate and fix. This results in the potential for upwards of two weeks' worth of work to identify and correct these erroneous support profile installations. This maintenance and repair time increases the overall commissioning time for each warehouse.

Figure 3: Examples of correctly and incorrectly installed support profiles

In an effort to reduce commissioning time for a warehouse, Vanderlande desires a diagnostic robot, or Dia-Bot, that can identify errors in the installation of their automated warehouses. The robot

will be controlled to transverse around the racking systems either or its own power or riding along on the system's conveyors and shuttles and uses a kit of sensors to detect any errors in the system. Examples of such error detection include a camera system used to identify incorrect installment of support profiles, an accelerometer to detect abnormal vibrations a package would experience on the conveyors or shuttles, and a microphone used to identify abnormal decibel levels of operation. Overall, the Dia-Bot is beneficial for giving visibility to areas of the warehouse system that may otherwise be difficult or dangerous to view or reach, identify potential material handling issues before system commission, validate the speeds and throughputs of the system, and decrease commissioning time for each warehouse system which benefits both Vanderlande and their clients.

The following sections will dive deeper into similar existing products and related codes, more detailed specifications, various ideation and decision-making modes, selected design choices and solutions, and next project steps.

2. Existing Products, Prior Art, & Applicable Patents

The world of robotics is a vastly growing frontier in today's market. There are thousands of designs for various tasks ranging from recreational use to commercial industries. Automation robotics has seen a heavy impact on the supply chain industry, with robotic arms and carts being able to assemble and transport products more efficiently and accurately. The team has been aware since beginning this project that there would be a robust amount of ideology and inspiration from the market. While this aided solution brainstorming, it could also complicate the patent process.

Fortunately, Vanderlande is not looking to produce the Dia-Bot for this crowded market. They are aiming to design a solution fully for in-house use. This takes the pressure of patenting of the team's shoulders. As stated, the specific market for sensor packed transverse robots is widespread. The bookends consist of cheap toy like designs as depicted below by a generic framed RC car with exposed sensors, motors, and wiring. This design is very low cost with the typical price point falling between $50 and $250 dollars. This is juxtaposed by the top-of-the-line type constructed of high end protected sensors with advanced processors and materials as depicted by the ROSbot [2]. This top of the line can vary greatly in size and application which increases the price range. Robots of this stature can be anywhere from $2,000 to $45,000 dollars. Examples of these products can be seen in **Figure 4** below.

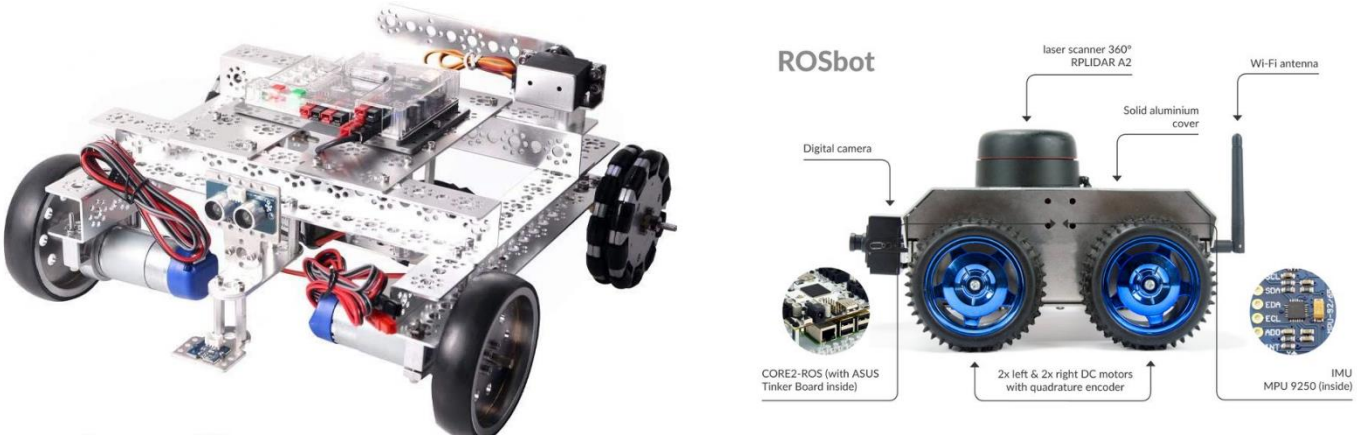


Figure 4: Examples of existing products similar in function to the Dia-Bot

3. Codes & Standards

Given that this robot must be picked up to be placed onto the warehousing system it will operate within, it is important to consider the lifting requirements. While the Occupational Safety and Health Administration (OSHA) does not have set regulations for acceptable weight to be lifting by a single person, or any other similar set regulations, it has worked with the National Institute for Occupational Safety & Health (NIOSH) to create a lifting equation. This NIOSH lifting equation gives a risk factor based on considerations of the lifting activity such as objects horizontal distance from the body, vertical location of the object to the floor, distance the object needs to be moved vertically, asymmetry angle or twisting requirement of the lift, frequency and duration of lifting activity, and the quality of the coupling or grip to the object.

While not directly within the industry of the Dia-Bot, the American Society for Testing and Materials (ASTM) have standards for the testing of response robots that fall near to some of the functionalities necessary for the Dia-Bot. ASTM standard E2566 gives a testing procedure to ascertain the visual acuity of a response robot to perform their given visual sensing task a counted number of times over a set period of time [3]. ASTM standard E2802 gives a testing procedure to determine how well a response robot is able to overcome vertical obstacles [4]. ASTM standards E2854 gives a testing procedure to measure the maximum range of wireless communication to complete tasks in line-of-sight [5]. While these tests are not required for our Dia-Bot, they are helpful testing procedures for the validation and testing of the Dia-Bot.

4. Customer Requirements & Engineering Design Specifications

A. Stakeholders

Table 1 provides a detailed analysis of the stakeholder interest in the Dia-Bot. **Figure 5** shows an overview of the stakeholder analysis in a stakeholder 2x2 chart.

Table 1: Stakeholder Analysis

Stakeholder	Interests	Impact /Effect	Importance	Influence Scope
Georgia Tech	Documentation, expo demonstration, creativity, innovation, problem definition, analysis, and design validation	Low	2	Timeline, documentation, course requirements
Primary Advisor (Dr. Jiao)	Documentation, creativity, innovation, problem definition, analysis, and design verification	Medium	2	Feedback based on knowledge and experience, and resources
Secondary/ECE Advisor (Dr. Madisetti)	Creativity, innovation, problem definition, analysis, and design verification	Low	1	Feedback based on knowledge and experience, and resources
Vanderlande Industries	Business interests, talent, ideas (i.e. support to team's mechanical engineer) and final detail design	High	2	Feedback for usability, customer needs, feasibility, and stakeholder satisfaction
Vanderlande Industries Installation Engineers (User)	Reliability, ease to use, provides meaningful information, and quickens installation verification	Low	1	Performance, analytics, feasibly, and stakeholder satisfaction
Vanderlande Industries (technical POC)	Technical Knowledge and ideas, talent, analytical approaches, prototype validation, and deliverables	High	3	Technical knowledge for feedback and feasibility, performance, and customer needs analysis
Future Design and Engineering Teams	Technical knowledge and ideas, analytical approaches, deliverables, prototype, and documentation	Low	1	Problem scope, documentation, and feasibility

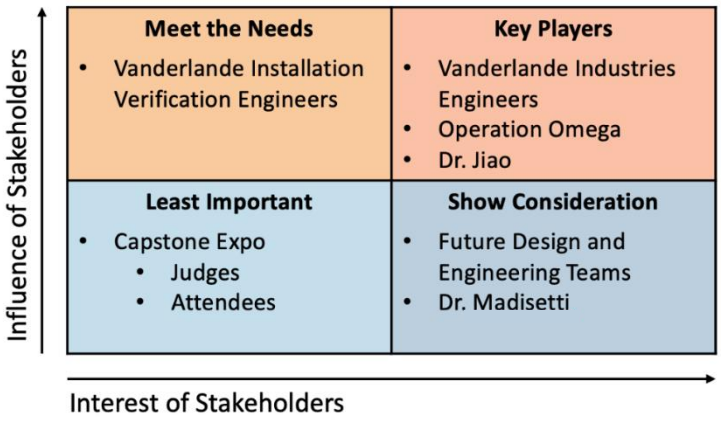


Figure 5: Stakeholder 2x2 Chart

B. Customer Requirements

Based on the team's interaction with key stakeholders, customer requirements for the Dia-Bot have been identified and divided into eight (8) categories, as shown in **Table 2** below.

Table 2: Customer Requirements

Category	**Customer Requirements (Explicit and Implicit)**
1) Function	a) Collects environment data
	b) Recognize robot's location
	c) Moves through system
2) Size	a) Fit in System: 620mm x 400mm footprint, 300mm tall
3) Weight	a) Be under 35 kg
4) Cost	a) Less than $950
5) Use	a) Easy to control
6) Power	a) Self-sustained (Battery Powered)
	b) Lasts through system navigation
7) Speed	a) Navigates faster than human verification
8) Navigation	a) Easily moved over rollers, conveyer belts, and all other surfaces

C. Evaluating Key Functions

In reference to **Table 2**, the first category, function, is the most important category here because the main goal of the robot it to report the location of errors within the large racking system. The robot needs to be more effective than the human means of verification to be worth using over the current validation techniques. The robot should be able to fit and go through the racking system (for the size and weight requirements). Additionally, the Dia-Bot must be easy to control and cordless, therefore needing self-sustaining power and wireless communication. The robot needs to be able to navigate over all the different surfaces it may encounter such as rollers, conveyer belts curves, diverts, shuttles, inclines, declines, merges, and diverges. Lastly the Dia-Bot must be cost effective. While Vanderlande Industries will not need a large quantity of these robots, the team's goal is to be able to produce a robot for no more than than $750.

D. Constraints

The robot has several constraints at this time. Notable the robot must fit inside of a 620mm x 400mm footprint with a maximum height of 300mm. Additionally the robot cannot exceed 35kg or the racking system will not be able to safety and effectively support the Dia-Bot's movements. The user must be able to see fully around the robot, which requires that the robot have a camera with the ability to rotate 360° along with vertical tilting up and down for an angle of 45° (from min to max – a minimum of 10° off level in both directions). Because the Dia-Bot must relay the racking system's environment, the robot must be accurate in its location within ten feet (±12 inches).

E. Engineering Specifications

Based on the customer requirements and our constraints, Operation Omega has identified the follow important engineering requirements. The specification sheet is shown below.

Table 3: Specification Sheet

#	Spec	Date Updated	Requirements	Responsible	Source	How Validated
General						
1	Affordable for Small Scale Production	09/26/21	Unit cost < $950	Design Team	Sponsor	Manufacturing cost analysis for the entire product

Physical Characteristics						
2	Product Weight	09/26/21	< 35kg (or <77lbs)	Mechanical Team	Sponsor	Weight measurement of full-scale prototype
3	Product Size	09/28/21	Footprint < 620mm x 400mm Height < 300mm	Mechanical Team	Sponsor	Measurement of full-scale prototype
Performance						
4	Movement Speed	09/26/21	a) Speed increments linearly	Electrical Team	Sponsor	Testing of Robot's movement
			b) Top speed ≥ 275m/hr (or ≥ 15 ft/min)	Mechanical Team	Design Team	Speed calculation from RPM from the motor
5	Navigation Surfaces	09/26/21	a) Successfully navigates over belts, rollers, curves, and diverts	Mechanical Team	Sponsor	On site Testing (or if unavailable Simulation)
			b) Successfully navigates through inclines, declines, mergers, and diverges	Mechanical Team	Sponsor	On site Testing or test system (or if unavailable Simulation)
6	Robustness	09/26/21	Withstands vibrations induced while navigating the racking system	Mechanical Team	Sponsor	Performance Assessment
7	Visibility	09/26/21	Camera provides a 360° rotation with 45° vertical tilting	Electronic Team	Sponsor	Demonstration with full-scale prototype
8	Position Tracking	09/26/21	Location Accuracy within ±10 feel	Electrical Team	Sponsor	Algorithm analysis with projection supported by field testing
Electrical						
9	Power Management	09/26/21	Uses conventional wall plug	Electrical Team	Sponsor	Verify specifications and use appropriate battery in the final product

10	Operating Time	09/26/21	> 5 hours	Electrical Team	Design Team	Verify specifications and use appropriate battery in the final product

F. Importance of Specifications

From the Customer Requirements and the Specification Requirements, the team created a House of Quality in **Figure 6** to recognize the most important engineering requirements.

			Functional Requirements									
		Direction of Improvement	▼	▼	□	▲	▲	▲	□	□	□	▲
Relative Weight	Weight	Customer Requirements	Affordable for Small Scale Production	Product Weight	Product Size	Movement Speed	Navigation Surfaces	Robustness	Vissibility	Position Tracking	Power Management	Operation Time
11%	10	Collects environment data				○		○	●	○	▽	
10%	9	Recognize robot's location								●		
11%	10	Moves through system		▽	●	●	●					
10%	9	Fits in Racking System							▽			
8%	7	Be under 35kg		●	●							
7%	6	Less that $750	●									
8%	7	Easy to control				○	○	○	○			
9%	8	Self-sustained (Battery Powered)									●	●
7%	6	Lasts through system navigation		▽		○		●		○	○	●
8%	7	Navigates faster than human verification	○			○	▽		○	○		
10%	9	Easily moved over rollers, conveyer belts, and all ofther surfaces					●					
		Importance Rating Sum (Importance x Relationship)	85,22	89,77	173,8	204,5	226,1	119,3	160,2	170,4	113,6	143,1
		Relative Weight	6%	6%	12%	14%	15%	8%	11%	11%	8%	10%

Figure 6: House of Quality

Relationships		**Weight**
Strong	●	9
Medium	○	3
Weak	▽	1

Figure 7: House of Quality Relationship Icons Weighting

The center section of the House of Quality characterizes the relationship between the customer requirements (on the left side of the table) and the functional requirements (on the top right of the

table). These relationships are weighted based on their icons according to **Figure 7** which, along with the weight of each customer requirement, is used to produce an importance rating along the bottom of the table. We have added a relative weight to each of these requirements to normalize our results.

The House of Quality analysis shows that the most important requirement for the bot will be the Navigation of the Dia-Bot over the myriad of surfaces that it will encounter. This places importance on the earlier design work for the special tread system. Next, the movement speed of the robot will be the second most important function, so that the Dia-Bot can make an improvement from Vanderlande Industries' previous system of rack installation verification. Additionally, the team will need to provide Vanderlande Industries with our estimate speed of navigation over each type of surface so that we can calculate how the Dia-Bot can improve their installation verification time. Another important functionality will be the product size, and while this is an important function, the team does not foresee this constraint as a difficult technical challenge.

In the next group of important functions lies the Dia-Bot's visibility and position tracking ability. These are vital to the robot's facilitation of information and ability to navigate and locate problems. Next in importance there is Operation Time, Robustness, Power Management, and, finally, product weight and cost.

5. Design Concept Ideation

A. Functions

Possible functions and sub-functions necessary for an effective Dia-Bot were identified using a function tree show below in **Figure 8**. These functions and subfunctions were created and evaluated based on the requirements given from Vanderlande. The function tree splits the main responsibilities of the designed robot first into electrical and mechanical components. Six distinct functions stem from those: transporter and protect internal components on the mechanical side, and data collection, data analysis, signal connection, and controlling mechanical functions on the electrical side. These functions are broken down into subsections as necessary to ensure that the Dia-Bot encompasses all its requirements.

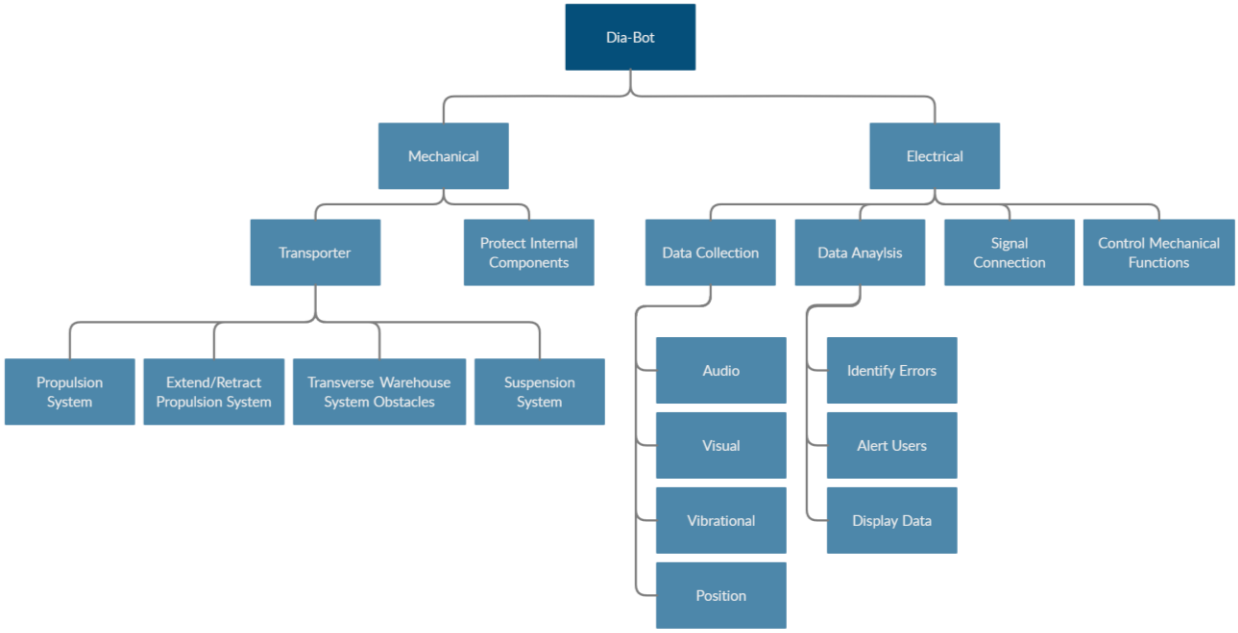


Figure 8: Dia-Bot Function Tree

Below the transporter function, the Dia-Bot must have a propulsion system that will allow it to navigate throughout its entire environment, consisting of conveyors, rollers, diverts, merges, and inclines. The Dia-Bot must also be able to retract its propulsion system to enable the 'ride along' functionality, allowing it to detect excessive vibrations. A suitable suspension system must also be selected to allow the Dia-Bot to detect vibrations coming from the environment rather than the bot itself.

Interactions between the Dia-Bot and the operate are possibly the most important function, as the bot would be useless if the operator cannot observe and verify the errors recognized. The operator should be able to control the movement and the error recognition sensors of the bot. The platform in which the bot and operator communicate is also an important consideration. Finally, the bot should be able to display real time sensor data, as well as log and report errors.

B. Morphological Chart

A morphological chart, as seen in **Figure 9**, was implemented to look more specifically at possible solutions to the subfunctions listed in the function tree. Several concepts were given for each subfunction, which allowed the team to narrow down which concepts would work best for the final

design. The team plans on revisiting the morphological chart frequently once the low fidelity prototyping phase is entered to make sure the prime concept is selected.

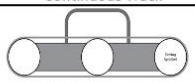
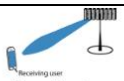
Function Grouping	Function	Concept 1	Concept 2	Concept 3
Movement	Propulsion System	 DC Motors	 AC Motors	 Servo Motors
	Retract/Extend Propulsion System	 Servo Motors	 Solenoids	 Linear Slides
	Traverse Automated Warehousing System Features	 Continuous Track	 Wheels	
	Suspension System	 No Suspension	 Shocks	
Protect Internal Components	Protect Internal Components	 Roll Cage	 Enclosed Sensor Box	
Recognize Errors	Audio Recognition	 Microphone	 Acoustic Pressure Sensor	
	Visual Recognition	 Camera	 Infrared Scanner	
	Vibrational Recognition	 Accelerometer	 IMU	
	Temperature Recognition	 Thermometer		
	Error Location Recognition	 Integrate over Accelerometer Data	 Beamforming	
Communication / User Interface	Control Movement and Sensors	 Web Access with Keyboard Controls	 Remote Control Center/Push Button Interface	 VR Interface with Haptic Glove
	Wireless Communication Platform	 Wi-Fi	 Bluetooth	 Extra Long Wires
	Error Reporting / Logging	 Sensor Threshold Limits	 Archived Data	

Figure 9: Dia-Bot Morphological Chart

Several different concepts were under consideration when it came to the movement of the Dia-Bot. To begin, we had to narrow down the propulsion system used, which could consist of DC motors, AC motors, or Servo Motors. In order to retract or extend the propulsion system, servo motors and solenoids were considered. The means of transportation, via tank treads or wheels, was also considered. Additionally, the type of suspension system was considered, with ideas ranging from no suspension, shock suspension, or single part suspension. For internal component part protection, two concepts were determined: a roll cage or an enclosed sensor box. In terms of being able to recognize and call out errors, the possible concept solutions can be seen in rows 6-10 in Figure 8. In order to control both the movement and sensors, three concepts were proposed: web access with keyboard controls, remote control center with push button interface, and virtual reality interface with haptic glove. For the wireless communication platform, three concepts were considered: Wi-Fi, Bluetooth, and extra-long wires. Finally, for error reporting/logging, the two concepts were setting sensor threshold limits that would report errors once the threshold was passed and archived data to be looked at once the bot had completed its run.

6. Concept Selection & Justification

A. Mechanical Design

The mechanical components of the Dia-Bot must work in concert to accomplish three general tasks: transport the camera and sensors across Vanderlande's system, protect the delicate electrical components during transportation, and retract its propulsion system to allow the robot to lie flat on the underside of its main body. Moreso than any other factor, the cost consideration of each component had the largest influence on the design of the Dia-Bot. Certain critical components, such as the timing belt and pulleys for the continuous track drivechain, can easily cost $600-$800. Therefore, to keep the total unit cost under the specified $950, the goal was to manufacture components when possible and purchase cheap components when necessary.

For transportation of the camera and sensors, a continuous track design and wheels were selected as the two most realistic options. Other ideas such as a drone were considered, but dismissed for the complexity, cost, reliability, and safety concerns. Each means of transportation offers its own set of advantages and disadvantages, listed in **Table 4**. Given that the robot must move through a veritable maze of conveyor surfaces, rollers, and sharp turns, a continuous track design was selected over wheels. The advantage of driving over obstacles outweighs the drawbacks to speed and maneuverability.

Table 4: Comparison of a Continuous Track Design vs. Wheels [6]

	Continuous Track	Wheels
Advantages	• Power Efficiency • Traction • Moving Over Rough Terrain • Weight Distribution • Complicated Suspension	• Lower Cost • Speed • Simplicity • Lightweight • Maneuverability
Disadvantages	• Lower Speed • Friction	• Driving Over Obstacles

Once a continuous track propulsion system was decided upon, the next step was to design the chassis. The two main considerations were whether the chassis should be symmetric and whether to include a suspension system. Making the chassis symmetric offers a few advantages for the user. The most important being that the robot would not need to rotate 180 degrees to travel backwards. In a tight environment, turning might not be physically possible. The operator can simply reverse direction and the Dia-Bot would handle the same as if it was driving forwards. However, the drawback is that such a design would be larger and, therefore, heavier. Using a suspension system would help insulate the delicate components from jostling and vibrations as the Dia-Bot traversed its environment. Once again, the trade-off is added weight and complexity. **Figure 10**, **Figure 11**, and **Figure 12** are mock-up sketches for a combination of these designs. In conversations with Vanderlande about the desired functionality of the Dia-Bot and the environment in which it would operate, they expressed a desire for the robot to be as robust and flexible as possible. To that end, a symmetric chassis with a suspension system is the strongest combination. With a finalized concept for the transportation of the camera and sensors, the next phase can begin: designing the chassis.

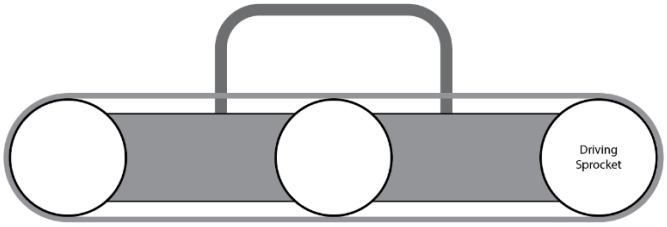


Figure 10: Symmetric Chassis Design without Suspension System

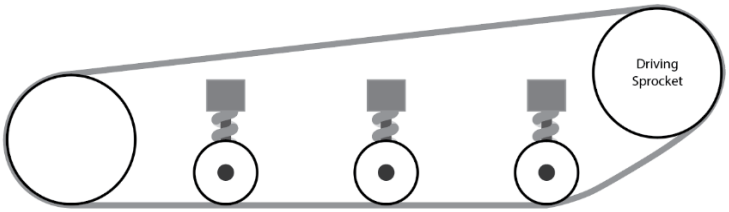


Figure 11: Asymmetric Chassis Design with Suspension System

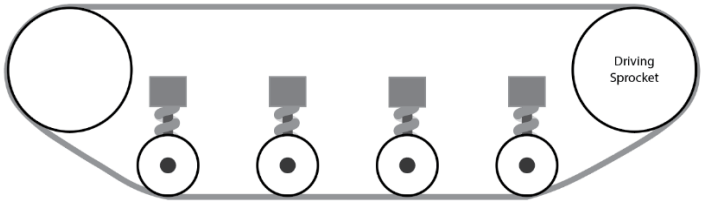


Figure 12: Symmetric Chassis Design with Suspension

Because a set of robust rubber tank treads with sprockets to use for the continuous track drive would cost roughly half the Dia-Bot's budget, two alternatives were found at a much lower price point: roller chain with flanges and timing belts. After speaking with Vanderlande, the decision was made to use the quieter timing belt. When using timing belts, there are two considerations for the design. The first being that that timing belts often run using flanged pulleys. If the flanges of the pulleys extend past the surface of the timing belt, then the Dia-Bot would essentially be running on wheels. The second consideration is that as the Dia-Bot turns, a lateral force is applied to the timing belt, pushing it off the pulleys. This phenomenon is illustrated by **Figure 13**. To overcome both considerations, a self-tracking belt, which has a centerline extrusion that meshes with a channel in the pulleys, will be used.

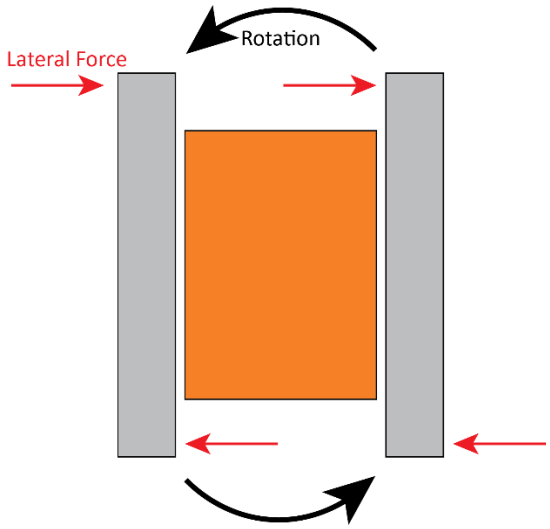


Figure 13: Visualization of Lateral Forces Acting on Timing Belt during Rotation

Designing the suspension system was a difficult task for two reasons: the system cannot extend into the cavity where the main body of the Dia-Bot will reside, and the system cannot extend up past the plane of the timing belt. Despite the difficulties, a compact modular system (FIGURE) was designed. A pulley attached to a suspension bar rotates around a fixed axle, while a RC spring shock dampens the movement. In addition to insulating the Dia-Bot's electrical components from vibrations and jostling, the suspension system serves as an auto-tensioner to ensure that the timing belt always remains taught.

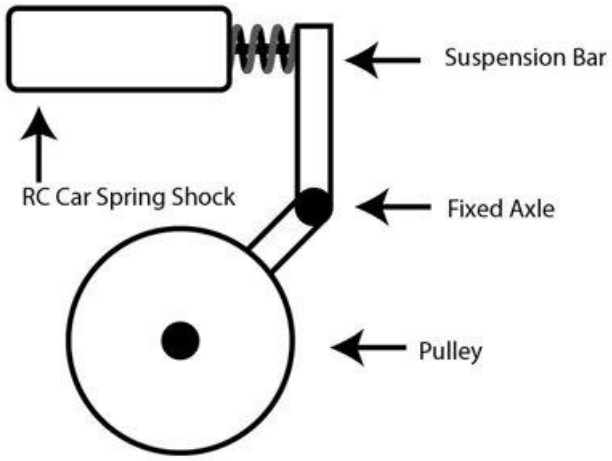


Figure 14: Suspension Subassembly Design

Retracting the propulsion system to allow the Dia-Bot to lie flat on a conveyor surface is a difficult task. The team designed the body to be interchangeable to support the new modular system. Our design is to attach the chassis to the main body with a linear slide and mechanical lock as seen in **Figures 15** and **16**. This allows a user to manually remove the main body from the chassis to use as a standalone sensor apparatus or to attach a different drive module. While it is possible to automate this process with a servo, we do not believe that the convenience outweighs the added complexity. The system will use two guide rails, shown in **Figure 17** with one attaching to each side of the chassis as well as four slides, shown in **Figure 18,** which will be integrated into the main body. Once on the rails, the body will be able to lock into position using a screw-on turn latch that will fit into a cut out slot in the chassis frame. This will inhibit motion in both the forward and backward direction while allowing for easy access for the operator.

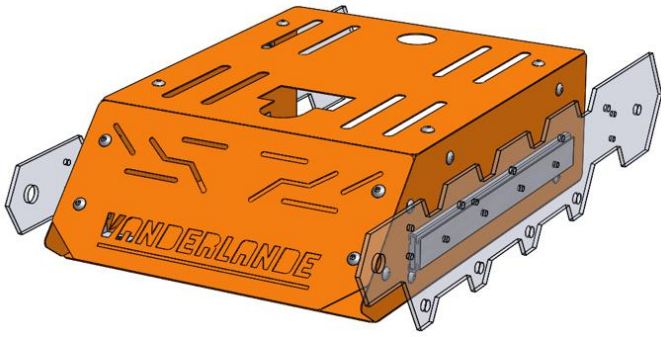


Figure 15: Modular Body Design

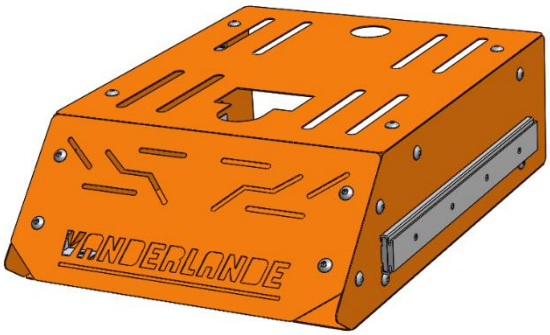


Figure 16: Modular Body Design Side Profile

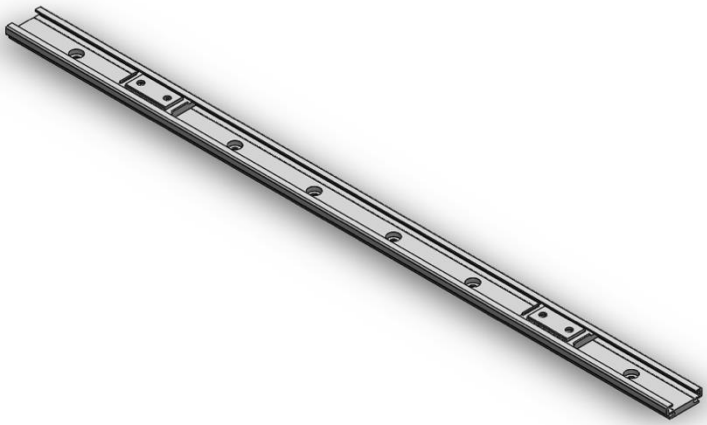


Figure 17: Chassis Guide Rail

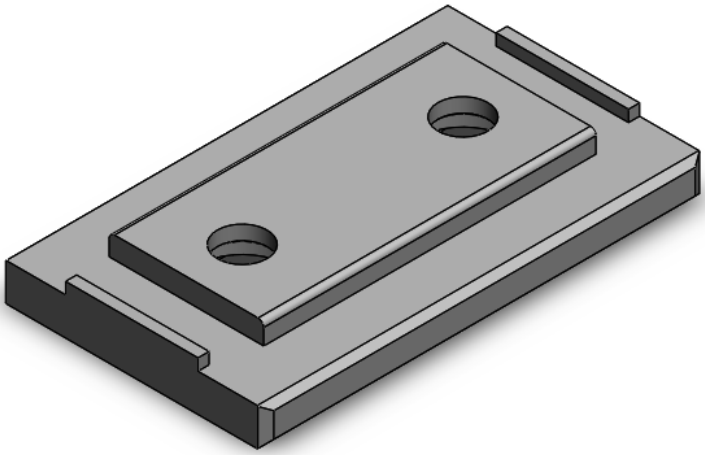


Figure 18: Main Body Slide Attachment

B. Electrical Concept

The electrical subsystem must support the robot's ability to move through the system, send real-time data to and receive controls from the operator, and track its position. To accomplish these functions, the team must recognize the requirements necessary such as a mobile power source, a rechargeable power source, movement generators (for the treads and camera), a light source for the camera, a user interface with the ability to control the features in real time, and an embedded processing center on the robot. As we consider the electrical subsystem's requirements, we made an evaluation matrix for the location positioning, movement generation, processing center, power source, real-time access to Dia-Bot data, and user interface. Below are the six evaluation matrices used to formulate a design for each of these critical tasks.

Table 5: Electrical Components Evaluation Table

Mandatory Criteria	Option 1	Option 2	Option 3
Location Positioning	GPS	On Dia-Bot algorithm with navigation and system map	Raw position data using telemetry
Movement Generation	DC Motors 	AC Motors 	Servo Motors 
Real Time Access to Dia-Bot Data	Wi-Fi 	Bluetooth 	Extra Long Wires 
Processing Center	Arduino UNO Rev 3 	Raspberry Pi 4 	Mbed (LPC1768 Cortex-M3) 
Power Source	Rechargeable: Lithium-ion battery 	Rechargeable: Lead Acid or SLA Batteries 	Non-rechargeable: Alkaline Cell Batteries (PP3) 
User Interface	Web server access by HTTP commands	Remote control with push button interface	Direct MCU control with VNC Server & Viewer

Referring to **Table 5**, this section discussed the options selected and why those are the best fit for the Dia-Bot. For Location Positioning, an affordable GPS unit will not yield the accurate position data necessary for the robot and from experience, raw telemetry data is known to be extremely inaccurate. As a result, the Dia-Bot will be using a positioning algorithm with navigation and a system map to determine the location. For movement generation, DC motors are light enough for the Dia-Bot's requirements and will provide enough torque and RPM. Therefore, these are the best option for the Dia-Bot. Next, Wi-Fi was chosen as the best way to access real time information from the Dia-Bot. Due to the large amount of data needing to be moved over large areas, Wi-Fi is the optimal connection technology, since Bluetooth has a very limited range of connectivity and significantly lower data throughout rates [7]. For the Dia-Bot's processing center, the team has experience using all three microcontrollers listed and believes that the Raspberry Pi 4 will best address the robot's Wi-Fi requirement because these microcontrollers have an on-board W-Fi chip [8]. This choice was made in tandem with the method of exposing the user interface. Vanderlande has expressed that a User Interface through a user's computer would be most cost effective, affordable, and malleable for future work, ruling out the remote control. In addition to the Wi-Fi module, the Raspbery Pi 4 allows direct remote access with VNC Server [9], a program which allow system control over any Wi-Fi-connected device running VNC Viewer. Since VNC is easy to set up and allows full access to the system, a graphical user interface could be developed most efficiently to directly control the Raspberry Pi's camera and GPIO pins.

The team has identified the following potential risks: extra setup steps for connecting the Wi-Fi module, poor camera output quality, and inaccurate positional calculations. For initial setup of the Dia-Bot in a new location, the Raspberry Pi must be accessed directly to first connect to the local Wi-Fi network. While Raspbian OS allow users to connect to Wi-Fi with a similar GUI to any normal desktop operating system, this connectivity process will likely require access via an external keyboard and monitor connection. For environments where there is poor signal connectivity, VNC will reduce the frame rate of image transfer while retaining a high-quality GUI and camera image. Lastly, the team had identified potential alternative solutions for position tracking in the event of difficulty with the on-board algorithm. These alternatives include high-end GPS modules or other tracking tags, while also interfacing with a map of the system in-use. Since these modules can be expensive, the custom algortithm was pursued first. This is discussed further in the project conclusions and future work section.

A block diagram displaying electrical components has been provided (**Figure 19**) to ensure that all necessary components are considered for the Dia-Bot. The team has used the following updated block diagram to complete a voltage analysis based on the load of each item on the voltage source and the Pi. The load analysis supports our current design, and our system should be able to sustain itself without too much draw on the system or any need to achquire new parts. The updated block diagram shows all the necessary logical, voltage and ground connections for our system.

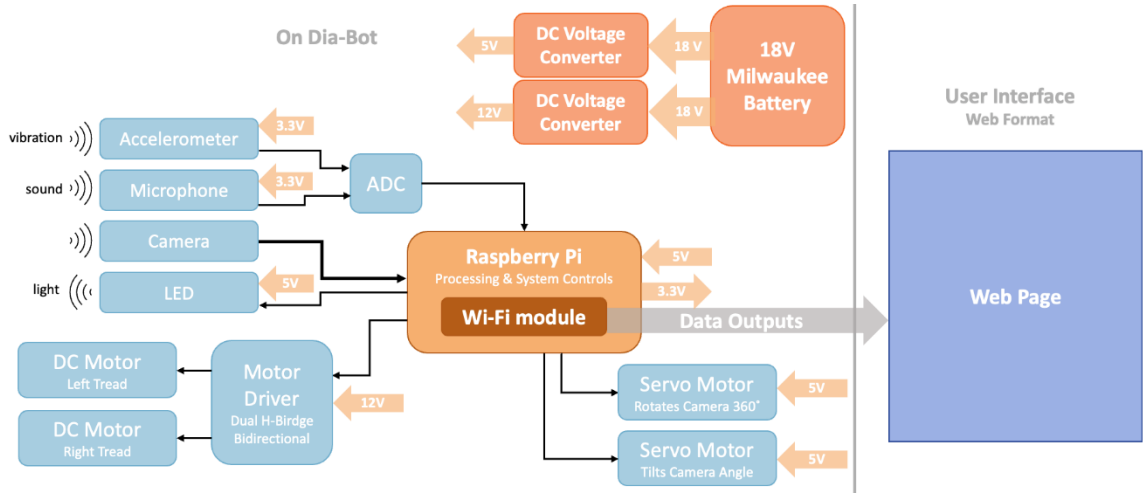


Figure 19: Electrical Block Diagram

C. Software and User Interface Desgn

The team hosted a User Interface design workshop with the client, Vanderlande Industries, to ensure that all the data and information passed to the operator is helpful, necessary, and comprehensive. In this design session, the team presented a preliminarily mockup of the user interface (**Figure 20**) and discuss the user groups who may interact with each of the data points collected, the purpose of the data collected, and the ease of use for the operator. This led to the revised user interface seen in **Figure 21**1521. This design workshop gave the electrical team members of Operation Omega the ability to see hwo the software end of the system may interact with the mechatronic end, including the mechanical outputs of the motors, the light of the LED, and others which are largely a part of the embedded system side of the Dia-Bot.

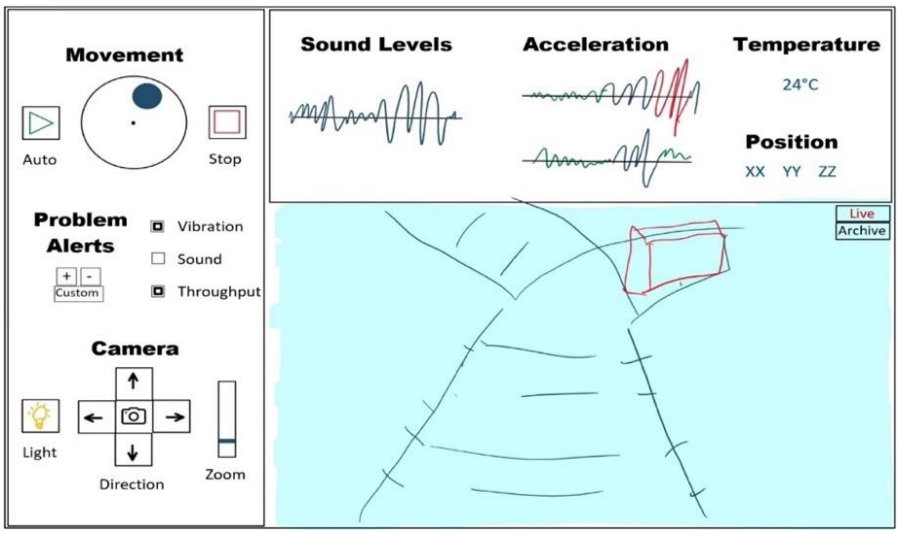


Figure 20: Preliminary User Interface Mockup

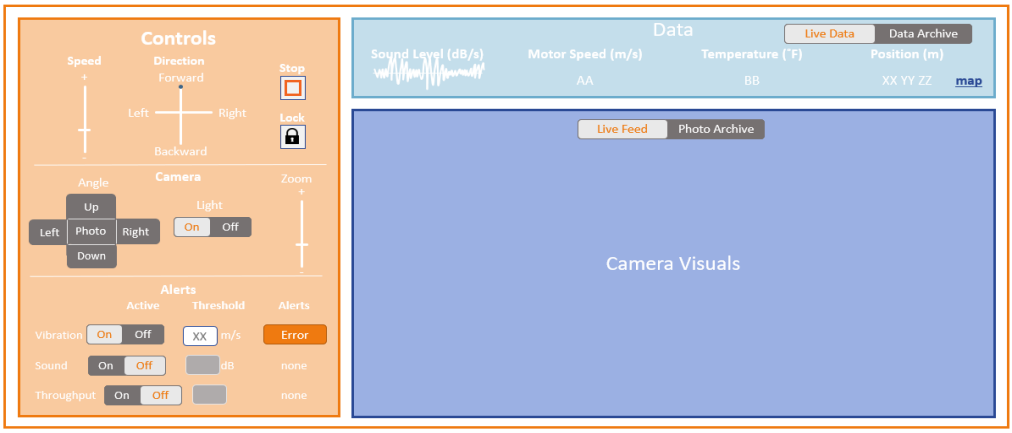


Figure 21*15*: Revised User Interface

As building of the electrical sub-system began, the electrical team used randomized generated data values to create **Figure 221622**. In the initial user interface prototype, the client can see live charts of sound levels, virbration, temperature, and position, above the real time camera visuals. On the left of

the image in **Figure 221622** is a control panel by which the operator may control the movement of the robot, camera, and light, as well as setting alert trackers and viewing their notification status.

Figure 22*16*: Initial Prototype User Interface

D. Software-Mechanical Interaction

As described earlier this section, the Dia-Bot moves via two treads, one on either side, each individually controlled by its own DC motor. An operator controls the movement of these two treads based on the eight-way direction buttons and speed slider exposed by the user interface. Since an operator may be controlling this robot with only a laptop, a discrete direction control scheme must be implemented to accommodate users with a keyboard and no touchscreen. Due to these constraints, the following control scheme is used to move the Dia-Bot with the eight directional buttons plus a center button for a short brake. **Figure** shows how each of the nine control buttons maps to a different speed setting for each tread.

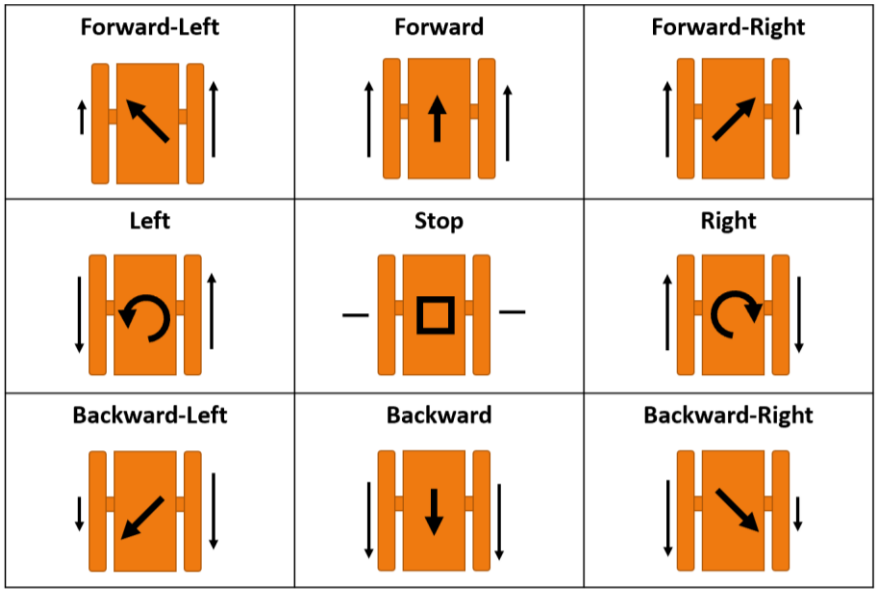


Figure 23: The nine buttons which control the Dia-Bot's movement and their corresponding speed setting for each tread.

7. Industrial Design

A. Logo and Color Scheme

Some of the main Industrial Design considerations are unity among Georgia Tech, Vanderlande Industries, and future design teams, cohesion of a final product from the user interface to the robot, ownership recognition, and practicality of the robot. To ensure unity among Georgia Tech and Vanderlande Industries, the team has been using a combination of the Vanderlande and Georgia Tech logos stacked on top of each other (**Figure 24**). To ensure that future groups feel a sense of ownership of this project, this combination logo in general does incorporate their identity as members of the Georgia Tech community and their growing relationship with Vanderlande Industries. This combination logo supports Vanderlande's ownership and Georgia Tech's innovation of the final product. While the combination logo is easily recognizable, it does not help distinguish the robot from other projects done between Georgia Tech and Vanderlande. Visually, the combination logo does recognize the hierarchy of Vanderlande as the owner, users, and proprietors of the robot while much of the innovation and support is provided by Georgia Tech teams. The Language of the combination logo is awkward, with two fonts stacked together, but without them the project would lose brand recognition.

Figure 2417: Vanderlande Industries and Georgia Tech Combination Logo

Because the robot will be owned and operated by Vanderlande Industries, all deliverables will use the Vanderlande Industries Color Scheme as seen in **Figure 25**18**25Figure 25**18. The primary color is Vanderlande Orange (Pantone 158). Using this color scheme shows respect for their contributions to Georgia Tech and helps support their corporate cohesion for the Dia-Bot. Lastly, target demographic research is unnecessary as the robot will be used by operators and most of the time the robot will not be visible because it will be traversing non-visible areas withing racking systems.

Figure 25*18*: Vanderlande Industries Color Scheme

B. Mechanical Design Considerations

Incorporating visual elements representative of Vanderlande and Georgia Tech into the physical build of the Dia-Bot further solidify the unity between the two institutions. To show Vanderlade's ownership of the robot, the main body of the Dia-Bot was painted Vanderlande Orange. Smaller components were more subtly designed to indicate Georgia Tech's involvement. The baseplates for the chassis (**Figure31**) have a hexagional, honeycomb aestethic, representative of Georgia Tech's Yellow Jackets mascot. The idle pulleys for the drive chain were printed out of yellow filament, once again indicative of Georgia Tech's Yellow Jackets. Having the largest component on the Dia-Bot painted to match Vanderlande Industries color scheme shows ownership, while having more subtle components allude to Georgia Tech shows partnership.

8. Engineering Analyses and Experiments

A. Mechanical Systems

Engineering Analysis for the chassis proved to be difficult. From the CAD model, it was clear that all the components would be able to fit together dimensionally. However, an engineering analysis to determine whether the purchased parts were strong enough to withstand the expected forces was

impossible to perform before purchase. As a consequence of the necessity of sourcing cheap parts, many of the components, such as the RC spring shocks and the aluminum composite material for the baseplate, did not have their specific material properties listed. Rather, the RC spring shocks came with extra internal springs advertised as 'low', 'medium' and 'high' stiffness, while the aluminum composite material was described as lightweight, stiff, and strong.

Without the ability to perform finite element analysis on the components to determine failure points and because the budget precluded multiple iterations of the prototype, the decision was made to overengineer the chassis such that it could withstand forces an order of magnitude greater than expected. This was proven during prototype testing.

For the modular body attachment system, a static load assessment was completed during the research for design materials. With a maximum possible load of 77 lbs, the Dia-Bot needed a durable and strong attachment system to meet the team's requirements. The sourced parts for the slide and rail system have a static load maximum of 110lbs. These parts were selected based on their ability to support the maximum load as well as give the system a minimum safety factor of 1.42 based on our approximations and free body models.

When determining the proper motor to use in the propulsion system, the main specifications that must be designed around are the torque and rotational velocity. Specifically, the stall torque requirement and desired RPM are vital specifications. There has been extensive research done into proper stall torque required for robots operating propulsion systems based on DC motors. The rule of thumb used is that the stall torque should be greater than the weight of the robot times the radius of the driving wheel [10]. The selected motor has a stall torque rating of 39.6 in/lbs. With two motors and driving sprocket diameter of 2", these motors are able to propel a system of up to 39.6 lbs using this model. The motor produces rotation of up to 410 RPM according to its gear ratio, which equates to a top speed of 4.88 mph, which is acceptable for the use case of the Dia-Bot.

B. Positional Tracking

A crucial part of the algorithm for tracking Dia-Bot position most accurately involves interfacing with Vanderlande's systems. Any package moving through the system will interact with various photo-eyes for proper material tracking. Since these photo spots will not move among within each system, the Dia-Bot's flagging by a photo-eye offers regular updates of positional location from a known source, which is used for proper calibration. Due to the inability to work in a system during the project timeline nor the ability to source a photo-eye on its own, proper position algorithm implementation and testing

fell outside of this scope for concept validation. This is discussed further in the 15. Conclusions: Project Deliverables & Future Work section, along with other viable solutions for future exploration.

9. Final Design, Mockup, and Prototype

A. Mechanical

A timing belt that meets are of the necessary criteria for the continuous track drivechain was sourced from BRECOFlex (**Figure 26**). The timing belt has a self-centering track which meshes with grooves in the pulleys to ensure that it remains securely in place as the Dia-Bot turns. A sketch of the timing belt and pulleys, shown in **Figure *19*27**, was drawn up to ensure that the perimeter of the design matches the perimeter of the 1100mm timing belt.

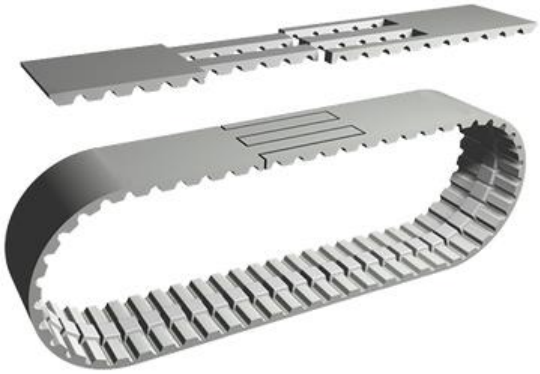


Figure 26: Self-Centering Timing Belt for Tank Tread Drive

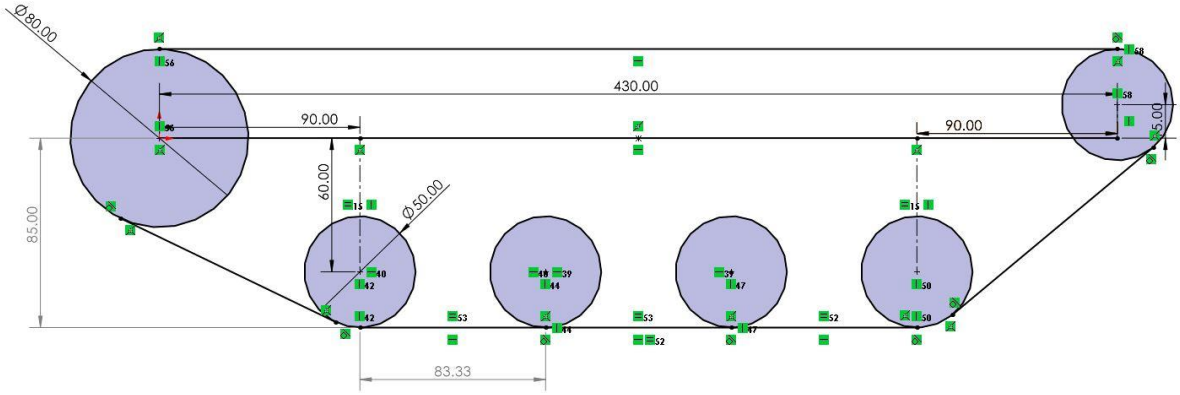


Figure 19: Layout Sketch of Chosen Chassis Design

BRECOFlex offers pulleys with a variety of customizations for their timing belts; however, each individual component has a price tag of a few hundred dollars. To alleviate this issue, CAD models of the

pulleys were generated for 3D-Printing. The 50mm 30 tooth idle pulley (**Figure *20*28**) has a large through hole in the middle for a roller bearing. The 80mm 50 tooth drive pulley (**Figure *20*29**) was designed such that an axle hub can be adhered to each face with four screws.

Figure 20: 30 Tooth Custom Pulley and Roller Bearing

Figure 21: 50 Tooth Custom Drive Pulley

From the suspension system design of (FIGURE), a cheap, compact, and modular suspension system was developed. Seen in **Figure 30**, a 3D-printed idle pulley revolves on an aluminum suspension bar around a fixed steel axle, with a RC Car Spring Shock providing sufficient dampening. In total the design will use eight suspension subassemblies.

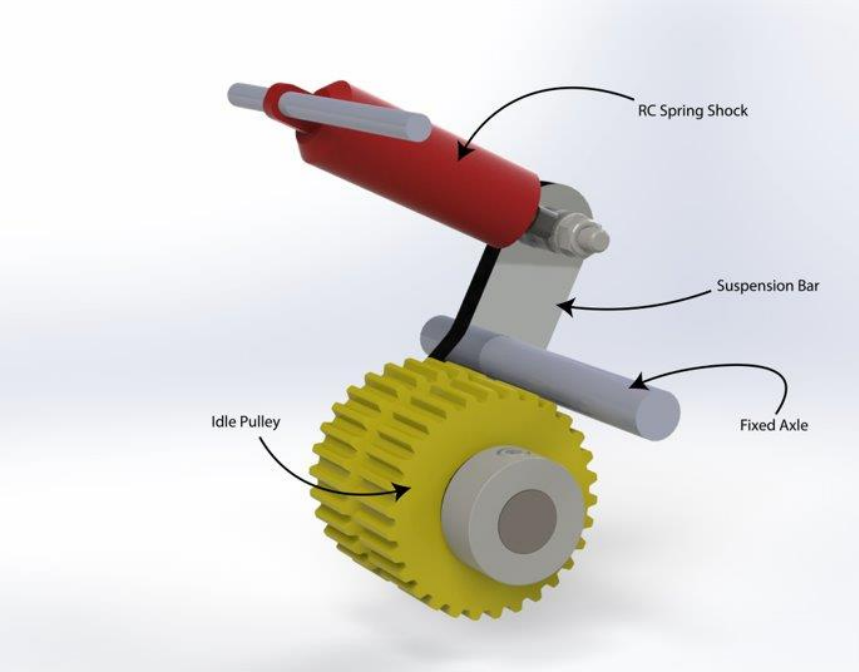


Figure 30: Suspension Subassembly

A baseplate for the chassis was developed with through holes to mount all other components (**Figure31**). The baseplate is made from an aluminum composite material that is lightweight and sufficiently stiff. Other necessary components such as axles, hubs and bushing were sourced and then a CAD assembly for one side of the chassis was constructed, shown in

Figure with components labeled. The entire CAD Model for the chassis with transparent baseplates is visualized in **Figure**

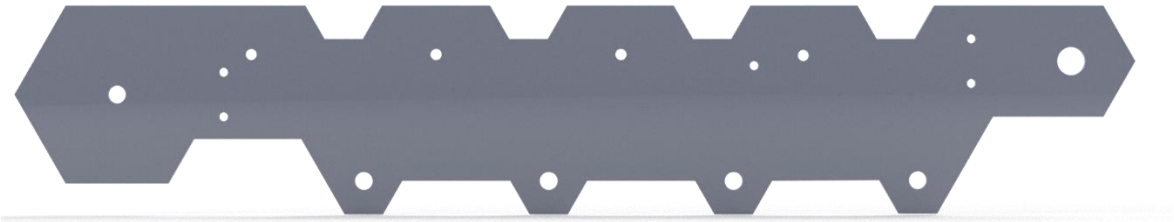


Figure 31: Chassis Baseplate

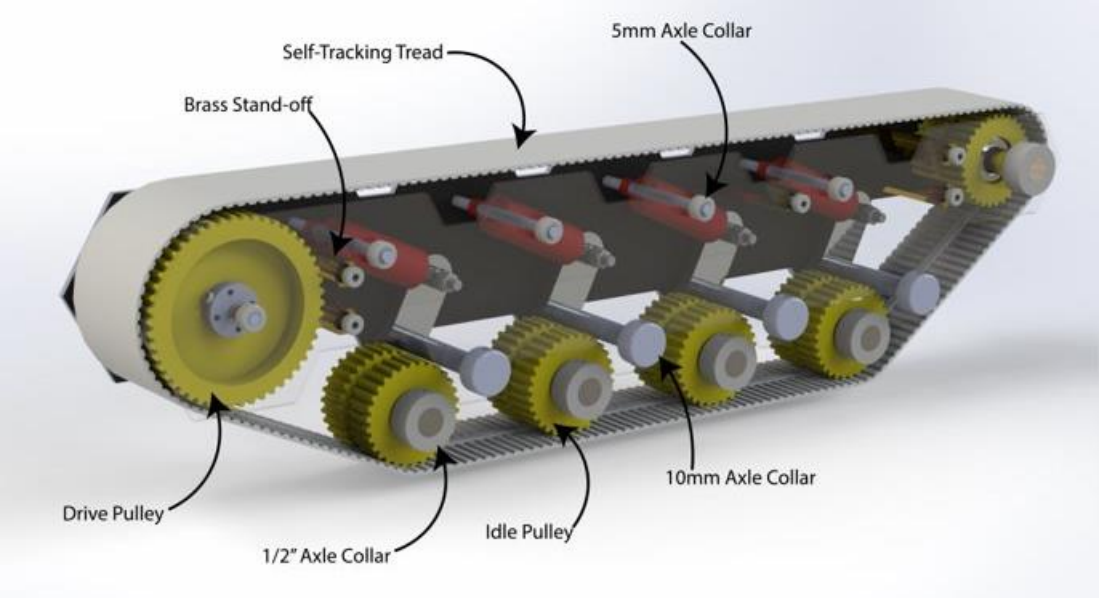


Figure 32*:* CAD Model of a Single Side of the Chassis Design

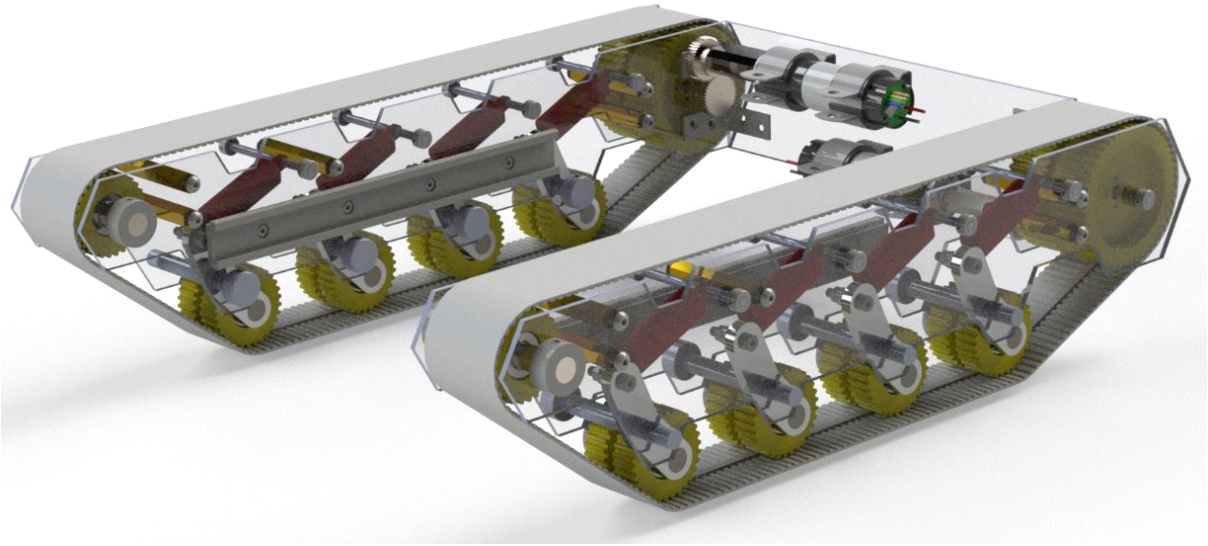


Figure 33: Full CAD Assembly of Chassis and Motors

With the design finalized and parts sourced, a complete prototype of the chassis and suspension system was constructed. To keep costs down, many of the components were manufactured using Georgia Tech's makerspaces. Once all the components were either purchased or manufactured, the

prototype chassis was assembled. A detailed assembly instructions can be found in the Fabrication Package. From left to right in (**Figure 30**), three stages of construction are shown: suspension system installed, single side constructed, entire assembly of chassis.

Figure 34: Construction of Chassis Prototype

As seen in **Figure 35**, the Dia-Bot uses a dual motor design to individually drive the right and left sides of the tank treads. This is necessary for the use of differencial steering in which nine directions of motion are achieved by altering the forward or reverse speeds of each side of the tank treads. The motors are set up in a 1:1 gear ratio from the motors to the axle of the driving sprocket instead of axial connections in order to avoid interferences between the two motors. Given the radius of the driving sprocket, the motors produce enough torque to drive a Dia-Bot of up to 40 lbs. at a speed of roughly 4.88 mph.

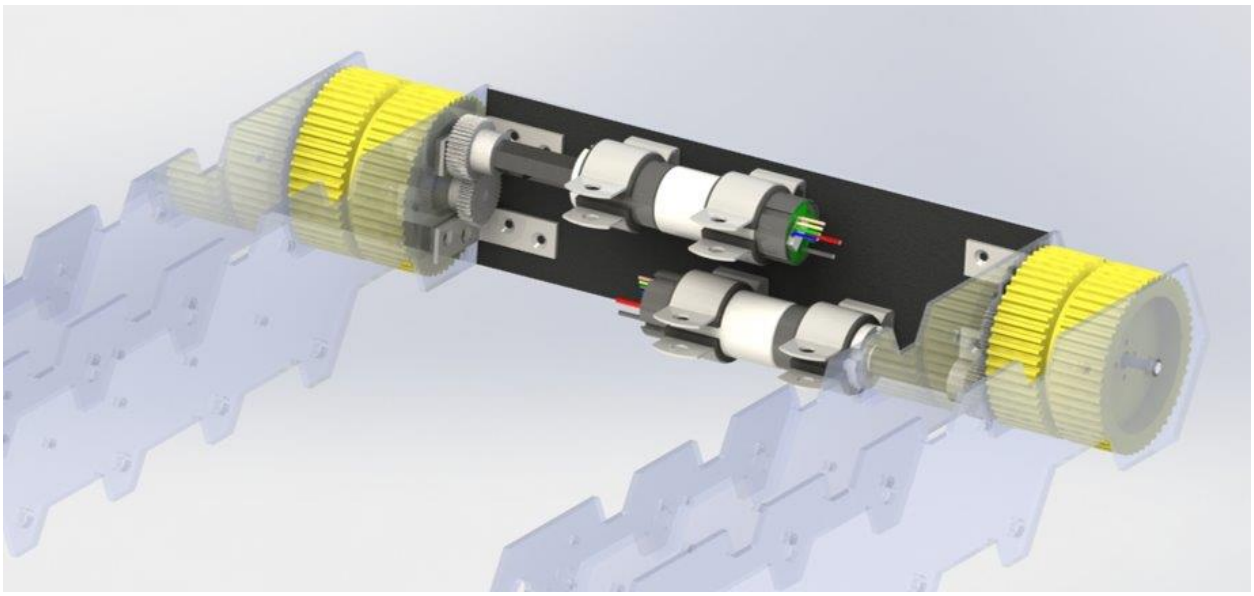


Figure 22: Detailed view of the Dia-Bot motor system

The main body was designed such that none of the mounting holes for the linear slides would interfere with the mounting holes for the chassis baseplate. In order to attach each individual panel to another, mounting holes for corner brackets were added, making sure to not interfere with linear slides. The final necessary design feature involved the camera, which was placed near the edge of the top panel. The camera components were given a clearance fit hole such that it could easily slide into the correct position without damaging the wiring. Once the camera position was finalized, the team decided to angle the front panel such that the camera could view any obstacles directly in front of the Dia-Bot. Once all the necessary features had been designed, additional slots were added to the front and top panels, allowing the operator to add additional electrical components if need be. These slots, paired with the lack of a rear panel, allowed the Dia-Bot to reduce any possible Faraday Cage effects produced from the Raspberry Pi and allowed adequate air flow to cool the electrical components.

With all the subassemblies manufactured, the complete prototype for the mechanical portion of the Dia-Bot was assembled (**Figure 36**). A complete bill of materials for this design can be found in Appendix B.

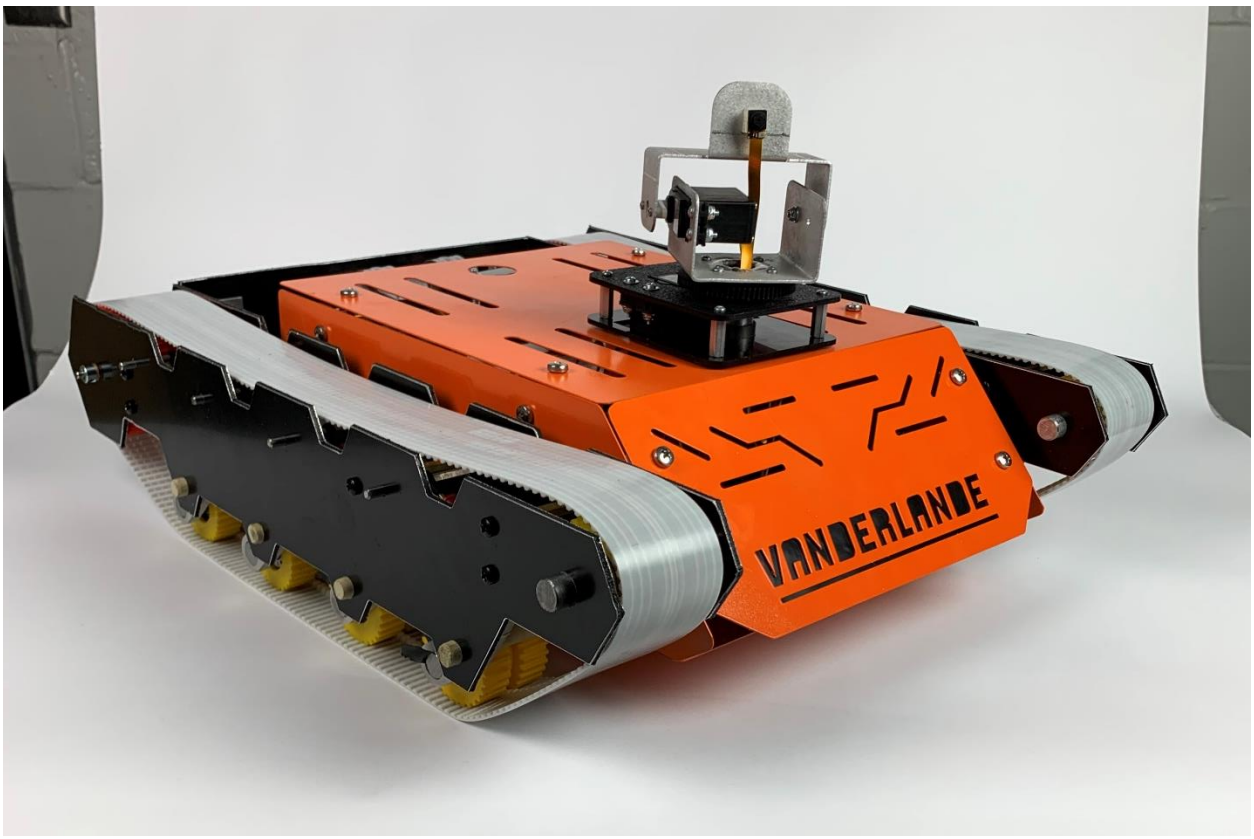


Figure 36: Complete Prototype of the Mechanical Side of the Dia-Bot

B. Electrical

For the Electrical Sub-System, the Dia-Bot is made up of a Raspberry Pi 4, an accelerometer, a microhpne, an Analog-to-Digitial Converter (ADC), a camera, an LED Ring, 2 Gearable DC Motors, a Dual H-Birdge Bidirectional Motor Driver, 2 Servo Motores, an 18V Milwaukee Battery, an 18V to 5V DC Voltage Converter, and an 18V to 12V DC Voltage Converter. All these electrical components are connected based on the Electrical Block Diagram shown in **Figure 37**. The Individual Pin assignments for each module can be seen in **Table 6**: Electrical Pin Assignments by Module. Each of the components in this system are all also correlated to a GPIO Pin on the Raspberry Pi 4, the universal ground, and/or the output of one of the DC Voltage converters. Lastly in **Figure 38248**, the pin assignmnets from the

Raspberry Pi 4 can be seen, which are assigned in the code for the Dia-Bot. With this information one could easily recreate and build the Dia-Bot's electrical components.

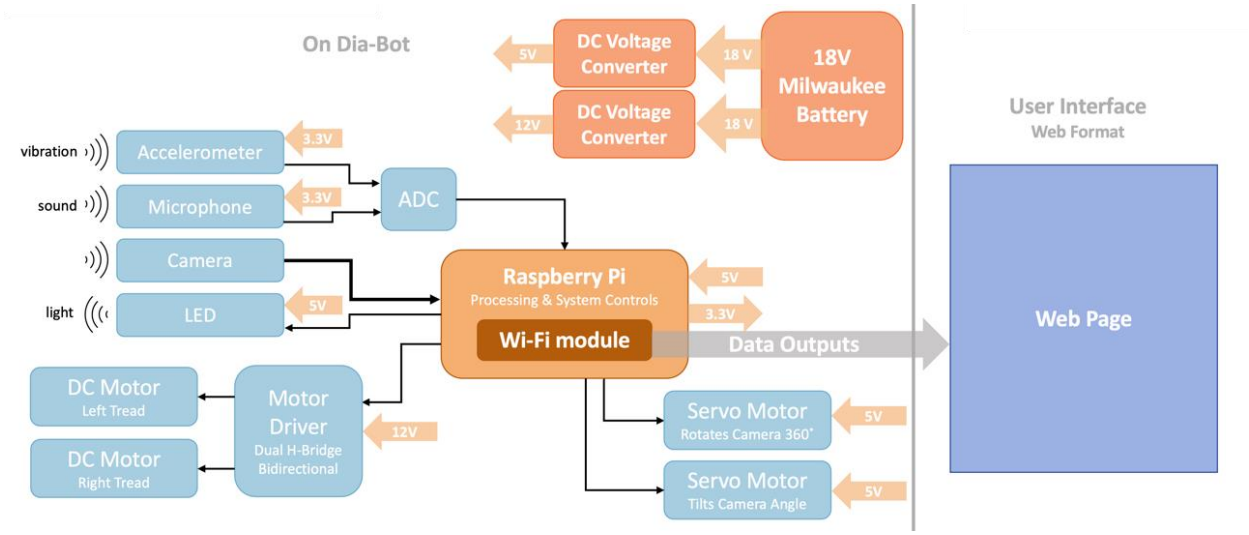


Figure 37*23*: Final Electrical Block Diagram

Table 6: Electrical Pin Assignments by Module

Module	Pin	Pi Pin or Intermediary
Microphone	V in	5V
	Data out	ADC in 1
	Ground	Gnd
Temperature Sensor	V in	5V
	Data out	ADC in 0
	Ground	Gnd
Camera Motor Tilt (HS-422 Servo)	Black wire	Gnd
	Red wire	5V
	Yellow wire	GPIO 12 (PWM)
Camera Motor Pan (HS-785 Sail Winch Servo)	Black wire	Gnd
	Red wire	5V
	Yellow wire	GPIO 13 (PWM)
Accelerometer	V in	3.3 V
	SDA	GPIO 3
	SCL	GPIO 2
	Ground	Gnd

Module	Pin	Pi Pin or Intermediary
LED Ring (NeoPixel Ring x12)	V in	5V (through a diode)
	Data In	GPIO 18 (PWM)
	Ground	Gnd
Camera	Camera Bus ribbon cable	Camera connection built into Pi
ADC: Analog to Digital Converter (MCP3002)	Pin 1: CS (Chip Select)	GPIO 8 (Chip Enable)
	Pin 2: Channel 0	Temp data out
	Pin 3: Channel 1	Mic data out
	Pin 4: Ground	Gnd
	Pin 5: Data In	GPIO 10 (MOSI)
	Pin 6: Data Out	GPIO 9 (MISO)
	Pin 7: Clock	GPIO 11 (CLK)
	Pin 8: V in	3.3 V

Module	Pin	Pi Pin or Intermediary
Dual H-Bridge	Motor A +	Right Motor + (red wire)
	Motor A -	Right Motor – (black wire)
	Motor B +	Left Motor + (red wire)
	Motor B -	Left Motor – (black wire)
	DC Motor +	12 V from DC Converter
	DC Motor -	Gnd
	Ground	Gnd
	In1	GPIO 23
	Enable A (ena)	GPIO 25
	In2	GPIO 24
	In3	GPIO 0
	Enable B (enb)	GPIO 5
	In4	GPIO 6
	5V, CSA, CSB	floating

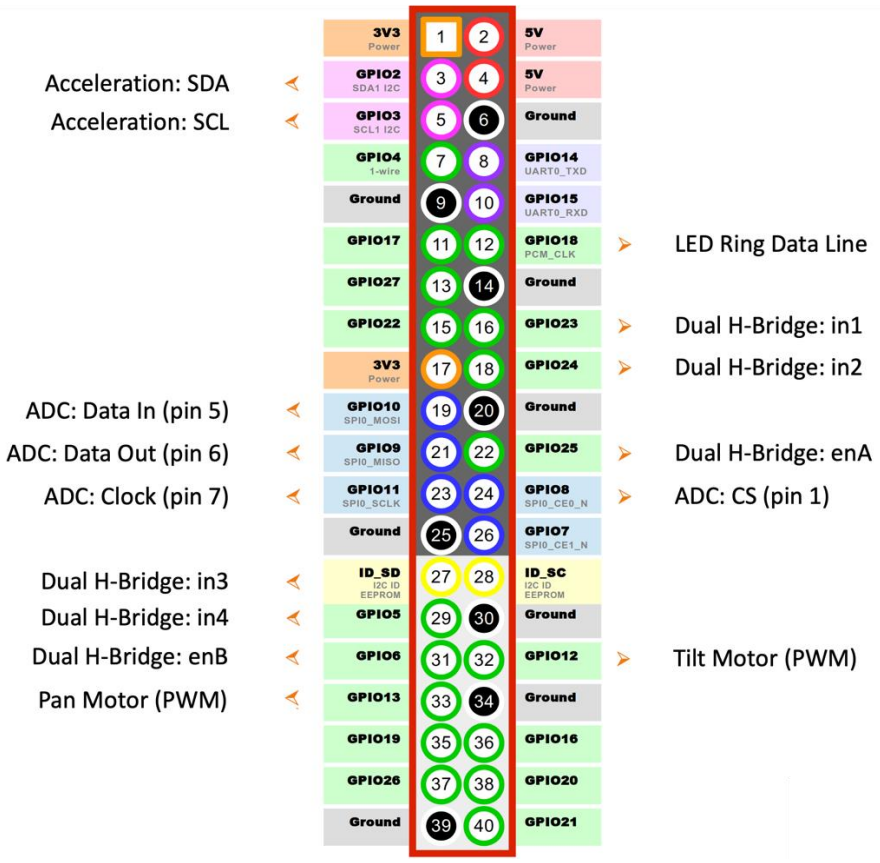


Figure 38*24*: Electrical Pin Assignments by the Raspberry Pi 4

In **Table 7**, there is a complete Bill of Materials for the electrical subsystem. With these components there are a few things to note. First during the testing of the electrical modules, the team discovered the 18V Milwaukee Battery provided 20.6V. This large voltage difference was a concern for the team as several our parts such as the 18V to 5V converters were rated for a maximum of 21V. The electrical team slow built and tested the electrical subsystem a few our electrical parts popped because of this voltage difference: such as the motor driver and two 18V to 5V converters. As a result, we have selected more robust electrical components for the final design and for demos we used small external power supplies to power items that required less than 12V.

Table 7: Electrical System Bill of Materials

Electrical Item	Detailed Name	Vendor	Part Number	Price	Number Needed
Accelerometer	SparkFun Triple Axis Accelerometer Breakout – LIS3DH	Sparkfun	13963	$5	1

Microphone	Electret Microphone Amplifier – MAX4466 with Adjustable Gain	Adafruit	1063	$6	1
ADC	Analog to Digital Converter – MCP3002	Sparkfun	0751	$2	1
LED	NeoPixel Ring – 12 x 5050 RGB LED with Integrated Drivers	Adafruit	1643	$7	1
Camera	Spy Camera for Raspberry Pi	Adafruit	1937	$39	
DC Motor	Johnson Electric Gearmotor and Output Shaft	AndyMark	AM-4230	$45	2
Motor Driver	DRV8876 H-Bridge Motor Driver with Integrated Current Sense and Regulation	Digi-Key or Texas Instruments	DRV8876EVM	$50	1
Servo Motor Pan	HS-785 Sail Winch Servo	SuperDriod Robots	TD-033-000	$60	1
Servo Motor Tilt	HS-422 Servo	SuperDriod Robots	TD-009-000	$15	1
Camera System	Camera 360 Pan and Tilt System - Standard	SuperDriod Robots	WC-003-300	$140	1
18V Milwaukee Battery	FirstPower 6.0Ah 18V M18 Replacement Battery	Amazon	N/A	$65*	1
Battery Adapter	M18 Power Wheels Adapter M18 Battery Adapter for Milwaukee M18 Battery Adapter	Amazon	N/A	$12	1
20V to 12V Converter	DROK DC 20V-72V Step Down to DC 12V 20A Voltage Regulator	Amazon	N/A	$18	1
20V to 5V Converter	Automotive 12V 24V to 5V 1A 2A 3A Converter Step Down Buck Power Supply	Ali Express by IdealPlusing	NP-DTD1224S51 and NP-DTD1224S53	$6	2 (one at 1A and one at 3A)
Raspberry Pi 4	Raspberry Pi 4 Model B with 2+GB of RAM	Adafruit	4295	$30	1
16 GB Micro SD Card	SanDisk Ultra SDSQUNS-016G-GN3MN 16GB 80MB/s UHS-I Class 10 mircoSDHC Card	Amazon	N/A	$6	1

* Not necessary for our sponsors to purchase because these are already available on site.

The design we have present our client with after our prototyping fulfills our sponsor's our sponsors main requirements of movement generation, temperature data, sound data, a 360° rotation with tilt, live camera feed, and the ability to see in dark environments all being powered by an 18V Milwaukee Battery.

C. Software

i. Graphical User Interface

The final Dia-Bot software was written using Python (specifically Python3, as version 2 will not work) and can be found on this GitHub page. To create a graphical user interface in Python, the Tkinter ("Tk Interface") library was used [11]. The raw user interface (without the camera preview overlayed) is seen in **Figure 35** below. Three main UI frames were built to host the controls, data display, and visual feed. In the Controls frame, traditional Tkinter UI elements allowed user interaction: *Label* (text), *Button*, *Slider*, *Entry* (text box), *Checkbutton*, and *OptionMenu* (drop-down menu) instances were included. The Data pane contains graphs of the live data which require Tkinter to interact with the *matplotlib* library. After the program suffered performance issues caused by clearing and re-plotting the graph every time it updates (which is every two seconds), an alternate solution had to be found. For plots which constantly update, the *matplotlib.animation* library was leveraged, which allowed for a faster method of plotting data which constantly updates. This utilizes a double-ended queue, or *deque*, data structure to hold the data values. The deque is beneficial in that appending new data points is a fast operation, and an upper bound on the number of data points can be set. Throwing out old data in the graph's cache ensure program performance does not falter over time by keeping too much data in memory.

Finally, the video pane only contains a *Label* with an *ImageTk.PhotoImage* as its display content. This image is always rendered on the Tkinter interface, regardless of video connection. This is because the most efficient method of rendering the camera view is to overlay the image on top of a separate interface, bypassing Tkinter entirely. However, the camera window is still managed by the Dia-Bot software via the *PiCamera* library. Upon startup of the GUI, the camera preview is defined by the program to fit entirely over the video pane. The camera preview is then displayed on the screen for use by the operator using the GPU, leaving CPU resources free to continue running the rest of the program at full capacity [12].

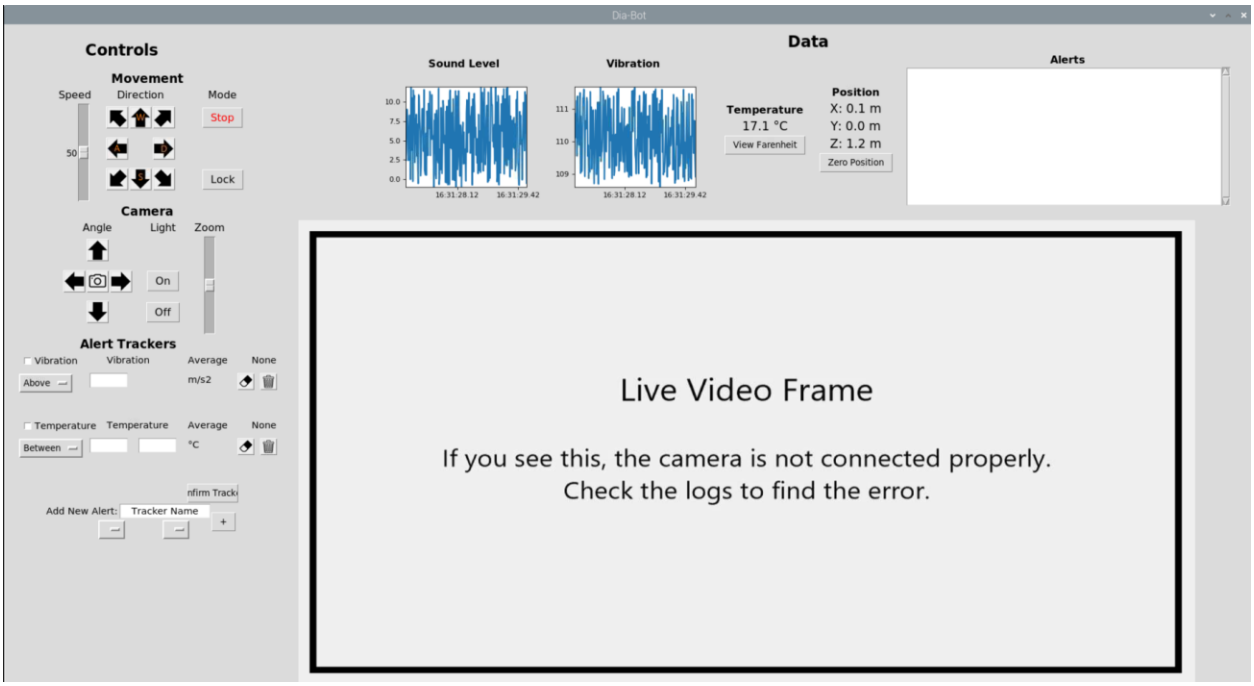


Figure 39: Basic GUI view shown without video overlay

ii. Software Operation

Beyond the basic user-GUI interaction, many other software sections had to be built to properly support all Dia-Bot and interface needs. In order to properly abstract and streamline the software design process, Python's classes were leveraged to build an object-oriented program structure [13]. This let each code block focus only on the functionality required of it, abstracting the details of lower-level processes.

The most immediately noticanle example of this separation with interaction occurs between the top-level GUI and the Raspberry Pi peripherals. In *PiInterface.py*, all camera and GPIO processes are handled, and many push buttons on the GUI call directly into these functions. For instance, Dia-Bot movement occurs once the user presses a directional button. Two digital signals for each DC motor will be set as HIGH or LOW depending on the direction, and a PWM pin duty cycle is set to correspond to the motor speed. Once the button is released, the PWM duty cycle is reset zero, waiting for the next movement input. Camera tilt and pan servo motor direction are also directly controlled by the duty cycle of a PWM signal, and each button press increases or decreases the duty cycle until the bounds are reached. The LED ring can be turned on or off by interfacing with the *neopixel* library through one GPIO pin [14].

Beyond control outputs, sensor data is also collected via *PiInterface* and the GPIO pins. The accelerometer communicates via the I2C bus and is read using the *adafruit_lsm303_accel* library [15].

Finally, the sound and temperature sensors provide analog outputs, but the Raspberry Pi does not have GPIO pins capable of analog inputs. Therefore, the software interfaces with these sensors through an ADC over the Raspberry Pi's SPI bus. The camera is handled separately, as it uses the Raspberry Pi's camera bus instead of the GPIO pins and therefore has its own separate software stack. However, as discussed in the GUI section above, *PiInterface* can still display a preview window and can command the camera to take and save a still picture.

Proper handling of the data streams takes three separate classes of operation for each input data type, titled *DataCollection*, *DataProcessing*, and *DataDisplay*. The *DataCollection* class actually has two separate instantiations for different means of accumulating data points: reading data directly from the sensors, then reading and appending those values to the current data array. The purpose behind this is discussed further in the next subsection, *iii. Multiprocessing Architecture*. The *DataProcessing* object periodically receives new data points appended from its *DataCollection* parent class, calculates the processing metrics (average, max, min, FFT frequency, and FFT magnitude), sends those metrics to the *AlertTrackers* (discussed below), and finally forwards those data points to *DataDisplay* for visual display. Each *DataDisplay* object is initialized with the proper GUI components: *matplotlib.animation* graphs for sound level and vibration display, and text labels and buttons for temperature and position. When these objects receive their periodic data updates from their accompanying *DataProcessing* objects, the corresponding displays are updated. For temperature and position visuals, the display strings are re-written to show the latest live values, since a full graph of these metrics is not necessary. For the vibration and sound displays, each new data point is appended to the deque data structure, then a graph is redrawn. These graphs are set to display the last three seconds of data and update every two seconds.

The final and most distinctive function of the Dia-Bot software is of the *AlertTrackers*, notifying the operators whenever potential problems are discovered in the input data. These alert trackers are designed to be flexible and completety customizable to the current metrics of interest to the operator. For example, operators may want to be notified whenever the instaneous acceleration is measured above a certain threshold. They can then create a new Alert Tracker in the GUI to display when the maximum values of the vibration metric are above that threshold. Tracker options include each data type (vibration, sound level, and temperature), all processing metrics listed above (average, max, min, frequency, and magnitude), and threshold ranges of above, below, or between values. Additionally, whenever a tracker is tripped, a printout of the alert is written to the GUI and the raw data, position, and an image are all saved to the storage system.

Once the user has set any trackers, these *AlertTrackers* receive periodic updates (about every 2.5 seconds) from the corresponding *DataProcessing* instance, detailing each of the metrics calculated over the last round of data collection. The desiged values are compared to the thresholds set for that data type. If the value is within the range, the GUI updates to display a new alert has occurred in two places: on the *AlertTracker* frame itself in the Controls pane, and in the Alerts printout in the Data pane for more detailed information. Additionally, the context of the new *Alert* is sent to a separate *FileIO* class, which writes the raw data to a CSV file in permanent storage. For greater operator understanding, an image is also taken by the camera and saved to the alert's new directory, and the current position is written, to assist the operators in investigating the potential error.

iii. Multiprocessing Architecture

Multiprocessing is a computer's ability to run multiple "threads" of operation at the same time. Due to the large amount of real-time processing required by the Dia-Bot software, utilizing multiple cores for parallel processing is crucial for ensuring proper software responsiveness. The Raspberry Pi 4 can four threads in parallel on its multicore Broadcom SoC [16]. Therefore, the software must spin up and handle multiple external processes, which will be scheduled by the Raspbian operating system to share time on each of those cores.

While both terms "thread" and "process" generally mean refer to a program, or a sequence of operations, running on a computer, there are crucial differences in their Python implementation which greatly impact software performance. Python has two corresponding libraries for implementing parallel computing: *threading* and *multiprocessing*. The *multiprocessing* library creates new processes that can be run on different processor cores simultaneously [17]. However, new threads made with the *threading* are not run at the same time as the original thread. Instead, execution is swapped back and forth between different threads at different times, all on the same process [18]. There are advantages to using *threading*, however: any objects and variables created on the same process are shared between different threads (i.e., all threads run in the same "context"). Another important note is that external processes created via *multiprocessing* can contain multiple *threading* threads. In order to build the most effective and efficient Dia-Bot software, both *multiprocessing* and *threading* were leveraged.

The diagram below (**Figure 40**Error! Reference source not found.) displays how both *multiprocessing* and *threading* are incorporated into the Dia-Bot software to properly utilize all processor cores. Additional threads and processes are all handled in *Threads.py*. Since variables in different processes exist in different data contexts, the *multiprocessing.Queue* data structure is used to allow communicate between them. While the *Tkinter* GUI runs on the main process, two other

processes are created per data type (sound, vibration, temperature) for a total of seven processes distributed over up to four processing cores. One process for each data type consists of a *DataCollection* object, and its sole purpose is to loop at the data type's given sampling rate and read the real-time data from the sensors. New data points are sent to a *data queue*, eventually to be retrieved by the *DataProcessing* instance in another process. This thread runs in its own process without any other threads competing for processing time in this process due to its time-sensitive nature.

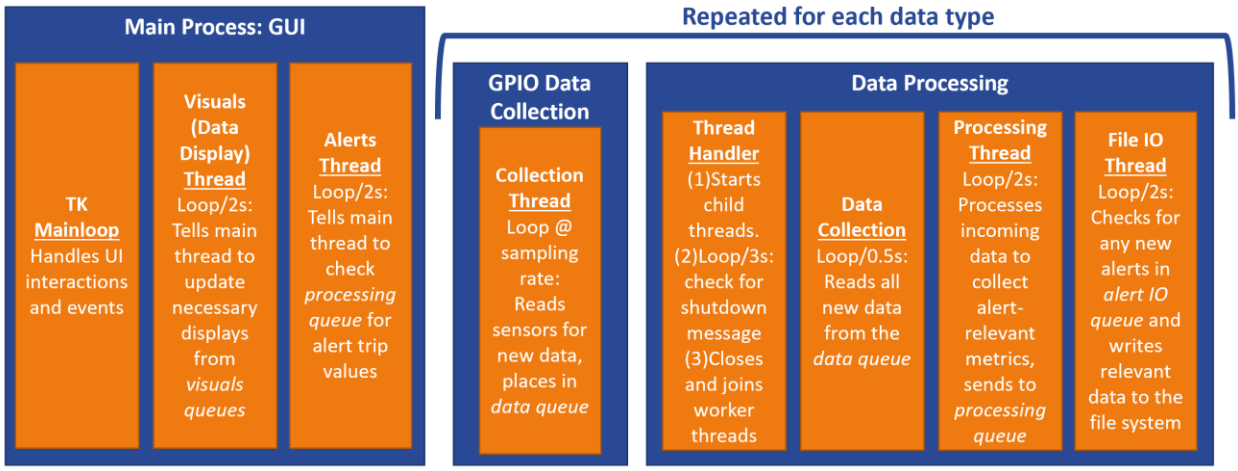


Figure 40: Multiprocessing and Threading Architecture

The other process spun up for each data type is for data processing. Multiple threads are created within each of these processes and thus share the execution time allotted to the process. This strategy is feasible since these tasks do not require a fast response time like GPIO data collection, instead focusing on data throughput. For proper handling of shutdown upon closing the program, a "handler" thread checks every three seconds for a message that indicates the program ending, and once it is received, sets a flag to stop additional processes. The other collection method of the *DataCollection* class is utilized in this process, as new messages in the *data queue* are read into this process's context and appended to the full data array for data processing twice per second. Every two seconds, the five processing metrics are recalculated by the processing thread and sent via the *processing queue* to the *AlertTracker* objects. Raw data is sent from this thread to a *visuals queue* for updating data display, to be discussed shortly. Finally, the FileIO thread checks for new *Alerts* sent to the *alert IO queue* and writes the data, image, and position to a file in storage.

Two additional threads are spun up in the main GUI process. While most clock time is spent in this process handling user input and data display, extra threads are needed to periodically interrupt normal Tkinter operation. The first of these functions is to manually update the display text for the temperature and position viewers. Every two seconds, the *visual queue* is checked for data updates, and

the text is updated accordingly. The other function is to receive updated data metrics from the *processing queue* and check if any *AlertTrackers* would be tripped. While many Python libraries do not integrate nicely with *multiprocessing*, Tkinter is especially sensitive to splitting objects among multiple processes. No objects on the GUI can be directly updated from another process, which creates the need for periodic interrupts by addition threads on the main process. By this method, the GUI is updated within its main process. These difficulties with *multiprocessing*, determining how to process and send data across many different threads, was the toughest challenge to overcome by the software development team.

D. System Architecture

The control and information flow for the final Dia-Bot design is shown in the system architecture diagram below (**Figure *2541***). This displays how each major component of the system interacts with other components and what those interactions involve. The diagram is broken into two sections based on localization and logical flow. A remote operator is detailed on the left, while the Dia-Bot system itself is displayed on the right, with information travelling over Wi-Fi creates the bridge between these major operating areas.

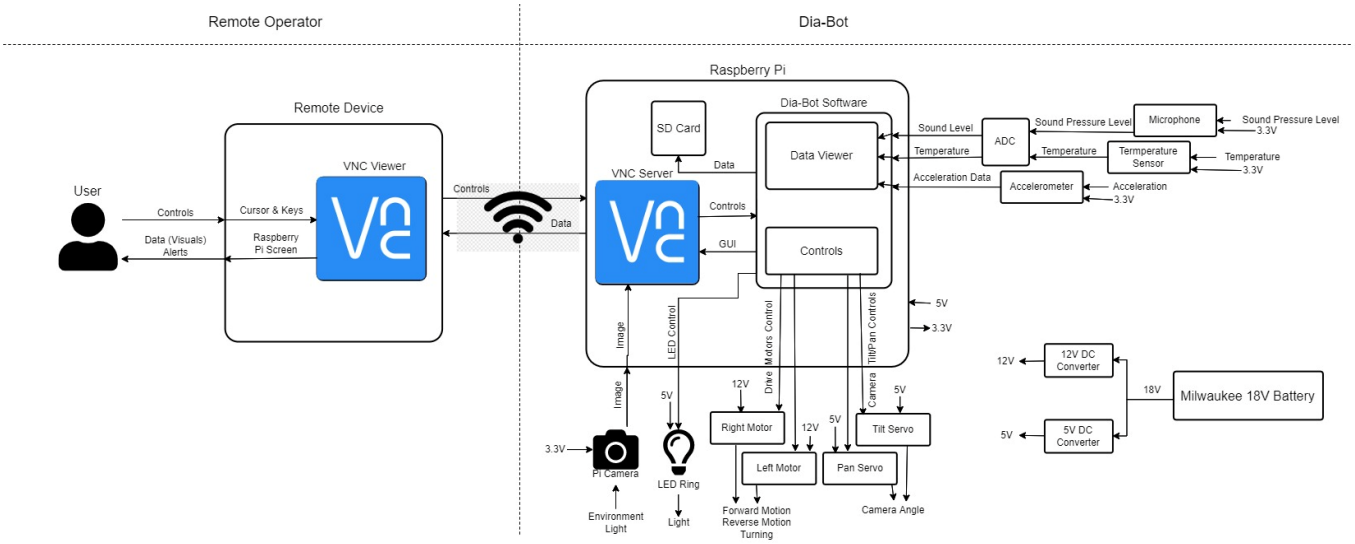


Figure 2541: System Architecture diagrams displays information flow

For the user's section, a Vanderlande operator primarily uses his or her own device to interact with a Dia-Bot. Any computer than can connect to Wi-Fi and run the VNC Viewer software is capable of Dia-Bot interaction, with the right credentials. As a user opens a VNC window, they see the exact display output from the Raspberry Pi controller as the Dia-Bot's data output. When the user clicks on UI artifacts and sends key presses, these raw commands are sent by VNC to the Raspberry Pi. These commands and

screen data are sent between VNC Viewer on the user's computer and VNC Server on the Raspberry Pi over a Wi-Fi connection.

On the other end, the Dia-Bot's Raspberry Pi receives these cursor and keyboard controls from VNC for direct control of the computing system itself. These are used to interact with the Dia-Bot software, hosted by the Python script *DiaBotGUI.py*. When the user presses a button (or key) corresponding to movement or camera angle, these control signals are sent over the GPIO ports to the proper motors. Additionally, an LED around the camera can be toggled on and off in this matter. This software also receives data inputs from each sensor (sound level, acceleration, and temperature) via GPIO ports, displaying the live data feed in graphs on the GUI. A video feed from the camera is rendered directly onto the screen in the software-defined video pane. All of this screen information for the GUI and video feed are sent back to the operator's device by VNC over the Wi-Fi connection. The Dia-Bot software also writes raw data and image files to a persistent storage device (i.e., SD card) when it is relevant. Finally, the input and output voltages are displayed for the Milwaukee battery, DC converters, Raspberry Pi, sensors, and motors, completing the electrical architecture requirements.

10. Manufacturing

A. Mechanical

The focus of the manufacturing design of this Dia-bot was cost. The vision of our client is to produce roughly 10 of these bots with minimal capital investment. The chassis frame was manufactured through a water jet machine. This process would require the purchase of the necessary machinery to be repeated, however, this can be contracted to a third party to minimize capital cost. The internal components of the chassis and suspension are parts that are available for commercial purchase. In the interest of cost, our team 3D printed the most expensive parts as our production lab had these additive manufacturing resources available to us. In addition, the motors and motor mounting system were all created using OEM parts, with exception for the backing piece connecting the two sides of chassis. This piece was created using the same composite sheet material as the chassis and only requires hole drilling operations which can be included in the water jet process of the chassis.

For the chassis components, the aluminum suspension bars were cut out using a waterjet machine (**Figure 42)**. The two drive pulleys and ten idle pulleys were 3D printed out of ABS using Georgia Tech's many printers around campus. The aluminum composite material of the baseplate deteriorated in the waterjet cutter, so instead it was cut out using a CNC machine in a metal shop. The various axles and other components were cut down to size using a vertical bandsaw. Unfortunately, the high strength

steel 10 mm axles used in the suspension system were unable to be cut. Wooden dowls were used as a cheap, last-minute substitute and were a surprisingly effective replacement.

Figure 4226: Waterjet of Suspension Bar and 3D Printing of Drive Pulleys

The main body housing was machined through a similar water jet process that the chassis was fabricated from. This allowed the team to input their custom body design that improved airflow cooling and decreased signal interruption. This process mimicked that of the chassis to limit excess machinery needed for the construction of the Dia-bot parts. The modular attachment system was comprised of locally sourced commercial components. This allowed the team to have a standard part to deliver to the client in the fabrication package to increase repeatability and simplicity of the bot. These decisions led the team to keep costs low will delivering high quality and durable components.

B. Electrical

On the electrical side there are few manufacturing details. During manufacturing all components should be tested individually, soldered, electrical connections covered with heat sink, electrical tape or other protectors, and then inserted into the body of the Dia-Bot with the necessary standoffs, zip-ties, and velvo. More spectfic details have been noted in 12. Risk Assessment, Safety, and Liability, specifically the B. Electrical section.

For the electrical design, there is no need for different material selection. We do need to place the microcontrollers with exposed bottoms – the Raspberry Pi and the motor driver – on standoffs, but all other electrical components can be secured to the Dia-Bot's body with Velcro. The electrical team additionally placed electrical tape on the bottom on the Raspberry Pi and the motor driver.

The main design considerations for the electrical system come with layout with the three following constraints. First, the camera bus from the Raspberry Pi is only about 10 inches long and must be pushed through the small hole in the Camera System. As a result of this cable length, the Raspberry Pi must be placed close to the front center of the Dia-Bot to make sure this electrical connection can reach and does not get pulled as the camera mmoves through is range of motion. Next, the gear motor leads are about 13 inches and due to how the back of the Dia-Bot is exposed the motor driver must be placed in the back of the dia-bot to make sure these electrical wires with high current are not pulled on as the robot navitages. Lastly, the 18V Milwaukee Battery should be easily accessed to be replaced. Therefore, the 18V Milwaukee Battery needs to be placed in the back of the Dia-Bot and should be in the middle to balance the load on each tread because the battery is heavier than most other electrical components. All other electrical conponents can be placed with the remaining space and cable connections can easily be lengthed.

C. Software

For proper software setup, a micro-SD card must first be flashed with the proper Raspbian operating system version. The only OS version on which the Dia-Bot has been tested is Raspbian 10, named Buster. At the time of this document's creation, the newest version is Bullseye, version 11. However, this version will not work with the Dia-Bot software due to the camera interface. Bullseye removes support for the *picamera* software stack, which is used by the Dia-Bot, in favor of the newer *libcamera* software [19]. But *libcamera*, at the time of this project, is only supported via the command line and not via Python. Therefore, the team used Buster and the *picamera* software for its Python interface with the Dia-Bot software.

For the dia-bot software to run properly several packages need to be installed. The packages that need to be installed include *numpy*, *matplotlib*, *pillow*, *mcp3xxx* (we will use *mcp3002*), *rpi.gpio*, *lsm303-accel*, *neopixel*, and *blinka*. The necessary package installation commands are all shown in **Figure** . If any of the commands in **Figure** error, the operator should add "sudo" to the front of the command to resolve the error. Some commands already need sudo to run properly and have that added to the command.

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install python3-pip

python3 -m pip install --upgrade pillow

pip install matplotlib

pip3 install "numpy == 1.15.0" --user

pip3 install adafruit-circuitpython-mcp3xxx

sudo apt-get install rpi.gpio

sudo pip3 install adafruit-circuitpython-lsm303-accel

sudo pip3 install rpi_ws281x adafruit-circuitpython-neopixel
sudo python3 -m pip install --force-reinstall adafruit-blinka
```

Figure 43: Package Installation Commands

One way to save time is by storing the disk image of an SD card that has been properly configured. This way, operators can simply flash the image to a new SD card for immediate use in a new Dia-Bot module.

Initial setup of the Raspberry Pi 4 module for Wi-Fi and VNC connection must be completed using an external monitor, keyboard, and mouse. After booting up the system, connect to the Wi-Fi network and begin exposing a VNC Server connection. At this point, any computer running VNC Viewer can connect to the Dia-Bot and control its Raspberry Pi module by using its IP address and signing in with the given credentials. It is recommended to remember the identity of each Raspberry Pi through VNC, so users can still connect to the same module even when the IP address changes. More setup details and tutorials can be found on the RealVNC website.

Once internet connection is established, retrieve the software from the Dia-Bot GitHub page and clone the main branch into any directory. Open the Terminal, run the command *sudo pigpiod* to enable GPIO connection (every time the system is restarted), navigate to the directory into which the code was cloned, and finally into the Dia-Bot_Prototype folder. Before running the software over VNC, ensure that the *Enable direct capture mode* option is enabled, shown below in **Figure 44***27***44**, in order to see an on-screen camera preview during operation. Finally, begin the GUI with elevated privileges (this is necessary for the LED library) by running *sudo python3 DiaBotGUI.py* to begin remote Dia-Bot control. Any alerts and images captured during operation are saved in the *Dia-Bot_Prototype/Alerts* and *Dia-Bot_Prototype/Photos* subdirectories, respectively,

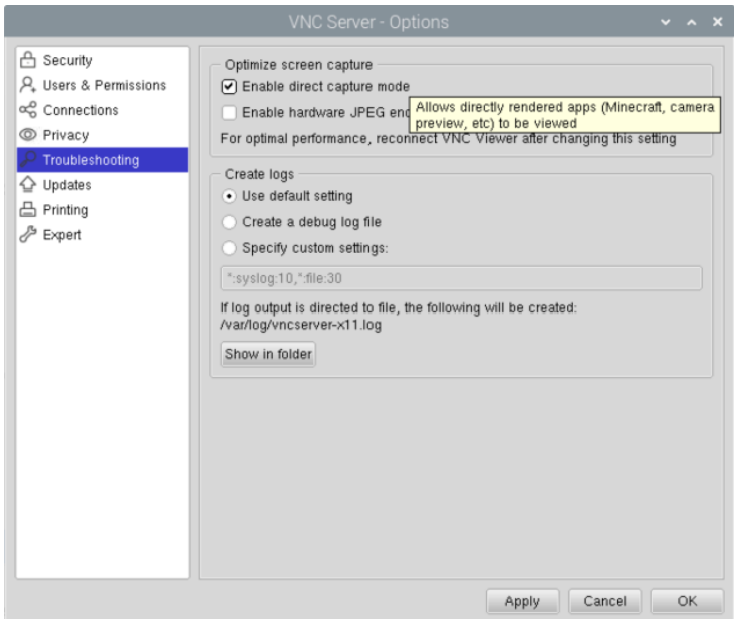


Figure 4427: VNC Direct Capture Mode Option

11. Societal, Environmental, and Sustainability Considerations

One aspect of societal considerations is evaluating how people's lives are impacted by the Dia-Bot. Since the Dia-Bot is initially intended to be controlled by and send data outputs to Vanderlande operators, this will not remove any jobs from the work force. The Dia-Bot will instead accelerate and augment the operators' decision-making process for evaluating potential installation issues. Operators will thus need to be trained to use the Dia-Bot properly and effectively. However, due to the intuitive nature desired for the user interface, this training should take minimal time. Proper instructions and debugging tips for known use case issues will be easily accessible via the user interface by leveraging existing programs on the Raspberry Pi. These considerations ensure the most positive social impact for the Dia-Bot.

Other considerations have been made when deciding on the power supply system. Vanderlande operators generally install conveyor systems using Milwaukee power tools, which use their own rechargeable 18V batteries. Due to the numerous advantages, this battery will therefore be used as the Dia-Bot's power supply. This type of battery is already readily avaible to the client during times when the Dia-Bot would be used, simplifying power management for operators, as these will already be constantly used and recharged during work cycles. Additionally, reusing Vanderlande's existing batteries ensures that Dia-Bot production consumes fewer materials.

To assess positive and negative impacts on society, a Social Impact Assessment (SIA) was performed for the Dia-Bot. The outlines for the assessment can be seen below in Tables 6-7, wherein **Table 8** defines the goal and cope of the assessment and **Table 9** performs inventory analysis.

Table 8: Goal and Scope of Social Impact Assessment

Objective of Assessment	Design Function	Functional Unit	Lifecycle Stages Considered	Associated Activities
Assess social impacts of robot	Remotely detect installation errors of Vanderlande automated warehouses	A single Dia-Bot Unit	End of Life	Recycling, Disposal
			Use	Vanderlande Engineer/Operator use in warehouse
			Manufacturing	Material forming, product assembly
			Production	Sourcing of (raw) materials

Table 9: Inventory Analysis Section Summary of Social Impact Assessment.

Product Lifecycle Stage	Stakeholder Group	Social Impact Category	Impact Indicators
End of Life	Consumer	End of Life Responsibility	% of recyclable parts of the Dia-Bot that are successfully recycled rather than disposed
Use	Consumers	Feedback Mechanisms	Increased efficiency in installation error detection
			Decreased maintenance time during system validation
		Technology Development	Increased Autonomy of Dia-Bot
Manufacturing	Workers	Health and Safety	Number of OSHA violations
			Reported Accidents
Production	Value-chain actors	Social Responsibility	Presence of explicit code of conduct

There are many possible positive impacts on human well being resulting from the Dia-Bot. Many electronic components used for the Dia-Bot require more complex assembly. This complexity drives the need for high skill labor which carries more sustainable wages. Additionally, electronic components require a vast array of raw materials which also must be processed, creating more employment opportunities. Furthermore, electronics are a global market, promoting trade across multinational value-chain actors each in their own local communities. The Dia-Bot will increase maneuverability into the warehousing systems where previously an operator may have had difficult or unsafe methods of entering. This eliminates potential safety hazards for the user.

There are also possible negative impacts on human well being as a result of the Dia-Bot. While electronic components require a vast array of raw materials and more complex production, it is not uncommon for the collection of raw materials and production to be performed in less developed areas. In these less developed areas, wages may be unsustainably low for the workers, or conditions harsh or unsafe. Some of the raw materials and natural resources used in the creation of these electronic components may be draining to the economy of less developed areas, or cause pollution in the areas where the materials are extracted in unregulated manners for profit's sake.

With concern for poor labor practices in the supply chain of electronic components is present, it will be sensible to select vendor companies that have proven records of proper labor practices. Any vendor with potential for child labor, forced labor, environmentally hazardous, or in any other way poor labor practices should not be considered as a value-chain actor for the Dia-Bot. It will also be prudent to select materials for the Dia-Bot to be as recyclable or as reusable as possible so that there is a minized footprint of disposed materials.

The production, manufacturing, use, and end of life of the Dia-Bot were all stages of the life cycle of the Dia-Bot considered in the SIA. These four life cycle stages incorporate different stakeholder groups of which the design of the Dia-Bot will affect. As such each has the potential to impact the society around this product. Consumers were selected as stakeholders because the use case of the product is designed with the operator as the consumer in mind. Workers and value-chain actors were considered stakeholders because of the positive and negative societal impacts possible discussed above. The social impact categories and impact indicators were selected based on known working conditions in expansive warehouse systems and labor conditions in countries that produce lower cost electronic components.

12. Risk Assessment, Safety, and Liability

A. Mechanical

While no product is remiss of risk, the Dia-Bot is designed in such a way where few risk factors remain. Because of the remote operation of the Dia-Bot, there is little necessary human interaction that can lead to bodily harm. Even so, some potential risks to human safety would be blunt trauma due to dropping the robot onto an appendage, being pinched by mechanical components, or experience a shock from the electrical components inside the device. Each of these risks can be viewed as a remote or improbable likelihood. In order to avoid the operator dropping the device, the chasis is designed using a rigid material in a form that provides plenty of grip. The risk of a finger being pinched by the device, such as between pulleys of the tank tread drive chain is mitigated by the spacing between these pulleys being larger than a finger. The risk of small electrical shock is decreased using proper wiring, connection, and soldering practices between electrical components.

In terms of risk to the Dia-Bot itself, there are also only small levels of risk. Possible instances of risk of damage to the robot include traversing off warehousing conveying surfaces, clipping racking system parts while being shuttled, and lateral forces misplacing the mechanical parts of the chassis. The Dia-Bot was programmed with a hard stop locking feature that the user can employ if the system nears any ledge to reduce risk of falling off the conveying surfaces. The Dia-Bot fits well within a bounding box of dimensions to ensure that when it rides on the shuttle no part of the Dia-Bot may come into contact with any unforeseen surfaces in the racking systems. Finally, to reduce the risk of chassis components being misaligned due to lateral forces, the tank treads of the Dia-Bot are equipped with a self-centering groove. These risks are also considered remote to improbable likelihood, as well as only of negligible consequence to operator and device.

B. Electrical

As with all electrical systems with high voltages safety is paramount. As our system uses an 18V power source, it is important to not that this voltage is not lethal, but it can produce a burn if not handled correctly. First, as electrical components arrive, they would be tested with Multimeters to ensure their output currents and voltages meet the datasheet specifications. As we did this, we found that the 18V battery produced 20.6V. As a result, our final design has a Bill of Materials with more robust parts to ensure the electrical parts will last over time.

First because we are dealing with high current and to stop a possible wiring mistake in the system it is important to have a on-off switch connected to the power supply. Our system had one installed internally so that the bot does not accidentally turn itself off while nagivating an area. This

power switch should be soldered between the high voltage output of the power source and the 12V and 5V buck converters. Each of the buck converters should be tested with a multimeyer and an DC Voltage external power supply which are easily found the Van Leer labs at Georgia Tech. Soldering should be done by people with experience soldering and while obversing all safety standards such as fans and safety glasses. After all soldering for electrical connections have been made all the exposed wire or solder needs to be heat shriked and/or covered with electrical tape.

Once all connecting wire are solder and properly covered the testing of electrical components can move from external power sources to the sub-system's 18V Milaukee Battery and the buck converters are tested for accuracy. After testing the power subsystem, each of the electrical components should be tested indivually before being compiled together. The software team's github has a "Test Code" folder, on the test_code branch, which contains code to test the LED Ring, Accelerometer, Microphone, ADC, Camera, and Motors individually

After all components are tested, the electrical system can be assembled and tested. When the electrical system in verified, it must then be installed into the body of the dia-bot. When installing the electrical systems, the Raspberry Pi 4 and the motor driver must be placed on standoffs can the dia-bot's body is metal and the exposed connections on these mircocontrollers will be disturbed. Additionally, to ensure components not on standoffs do not shake loose they should be either Velcro-ed, taped, glued, or other wise secured to the diabot frame. Lastly loose wires should be secure with electrical tape or vip ties.

While the electrical sub-system presents inheriant risks many of the precautions for medium voltage systems are well documented online and the dia-bot is not a risk to users.

13. Patent Claims and Commercialization

The solution that our team has provided to Vanderlande is one the client intends to keep fully in-house. Their vision is to fabricate roughly ten of these bots to aid an installation team when they are creating new systems. At this moment, there is no intention of making this product commercially available. For these reasons, our team did not pursue patent claims. The team was also able to avoid interference with existing patents or prior art by keeping this project in house.

14. Team Member Contributions

While team progress has consistently been made in group efforts, either the team as a whole or the mechanical or computer and electrical teams, everyone has played roles in the progress of the project. The mechanical team, consisting of Andrew, Jason, Hunter, and Douglas, has focused on the

development of the movement functions and requirements of the Dia-Bot. The computer and electrical team, consisting of Catherine and Connor, worked in parallel to advance the controls, communication, and user interface portions of the Dia-Bot's necessary functionality. A more detailed breakdown of tasks is as follows.

- Andrew conducted the research into the market's prior art and patents. By working off of the groups brainstorming and function requirements, he was able to produce information on related robots in the field currently. This led to easier decisions for the Dia-Bot and a good baseline for the group to branch off of. He created the design for the attachment of the modular drive system by sourcing the needed materials integrating them with the main body design team.
- Catherine has worked organize the team by creating presentation (in line with Vanderlande Industries' formatting), meeting notes, and the reports format. On a more technical note, she prepared the User Interface design workshop, the Electrical Block Diagram, the Specification Sheet, House of Quality, Customer Requirements, Industrial Design, and Stakeholder Analysis. In building the prototype, sdered, tested, assembled, and installed the electrical sub-system.
- Jason worked to create a clear and detailed function tree that incorporates the entire scope of functions of the Dia-Bot as well as doing research into the codes and standards relating to the solution. Furthermore, Jason did extensive research into different types of motors and drivetrain systems to be used in the propulsion of the continuous track system. In parallel, he researched the torque and RPM requirements of the output of the motor to ensure proper power transfer and speed of the Dia-Bot. Lastly, Jason completed the social impact analysis (SIA) to investigate the potential positive and negative social repercussions of the Dia-Bot.
- Hunter has spent much of his time designing the various components of the chassis, sourcing affordable parts for each subsystem, designing custom parts when needed, and then building out the chassis in CAD Software.
- Connor primarily architected, prototyped, and developed the Dia-Bot software for user interface, robot control, and data processing. In this, he established methods to wirelessly connect to and communicate with the bot via the Raspberry Pi. He also helped select and source some electrical parts, aiding in the early electrical system design. Additionally, he helped define and narrow the Dia-Bot scope and features, most notably by providing initial drafts of the function tree, user interface design and contents, and system architecture. He has also helped

add formatting and final editing for team reports and presentations. Finally, he created the web page used for submitting the ECE deliverables.

- Douglas worked primarily on the Design Concept Ideation portion, where he reviewed and revised both the Function Tree and Morphological Chart. He provided an extensive evaluation of each of the Dia-Bot's functions, as well as the concepts that led to the final design components. Additionally, he designed the main body and made sure it meshed with the conincident features. Finally, he created the CAD drawings for multiple custom made parts.

15. Conclusions: Project Deliverables & Future Work

Operation Omega produced a Dia-Bot prototype for concept validation in line with sponsor specifications. The initial steps of diagnostic robot ideation and discussion helped the team define a proper design scope for providing a solution to the problems that Vanderlande is describing. The team's Dia-Bot designs include proper movement modes, appropriate data collection, and an accessible software user interface for robot control and data visualization, all of which allow Vanderlande engineers to discover issues during the installation of their shuttle and conveyor systems.

The final physical Dia-Bot prototype is shown below in **Figure 45**Error! Reference source not found.. This featues a tough metal but vented central body for housing delicate electrical components, providing proper cooling air flow, and avoiding the Faraday cage effect on the Wi-Fi module. For bot motion, continuous tracks line a suspension system lined to traverse common conveyor obstacles. These treads can be easily removed, leaving only the body with a flat surface when ride-along mode is desired. On top, a camera mounted to a tilt-pan control system allows a 360° horizontal view with vertical tilting. This entire design measures at 520mm x 420mm x 265mm with a weight of around 14kg, which are within the given limits.

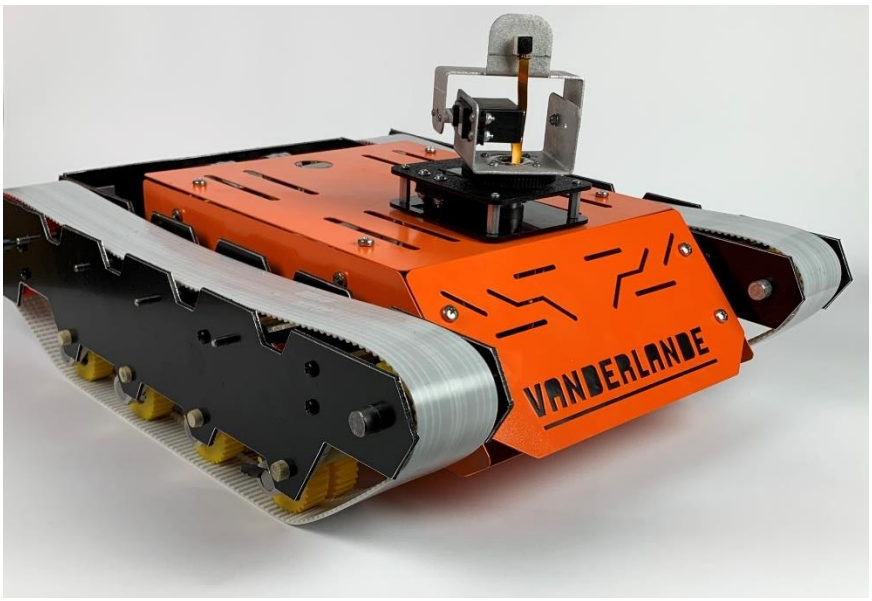


Figure 45: Dia-Bot Physical Prototype

The Dia-Bot's accompanying software and GUI is displayed below in **Figure 6**. This software exposes the necessary features for Vanderlande operators to control the Dia-Bot and receive its data. The controls pane on the left side lets the user control the movement speed and direction, camera tilt and pan, and the alert trackers. These alert trackers control the metrics and thresholds of each data stream for which the users are notified, helping operators identify where certain issues need attention. The results of these are alerts are displayed on the right side of the data pane on the top-right of the interface. The data and video panes also stream real-time information received by the Dia-Bot, including camera visuals, soud level and vibration graphs, and temperature and position values.

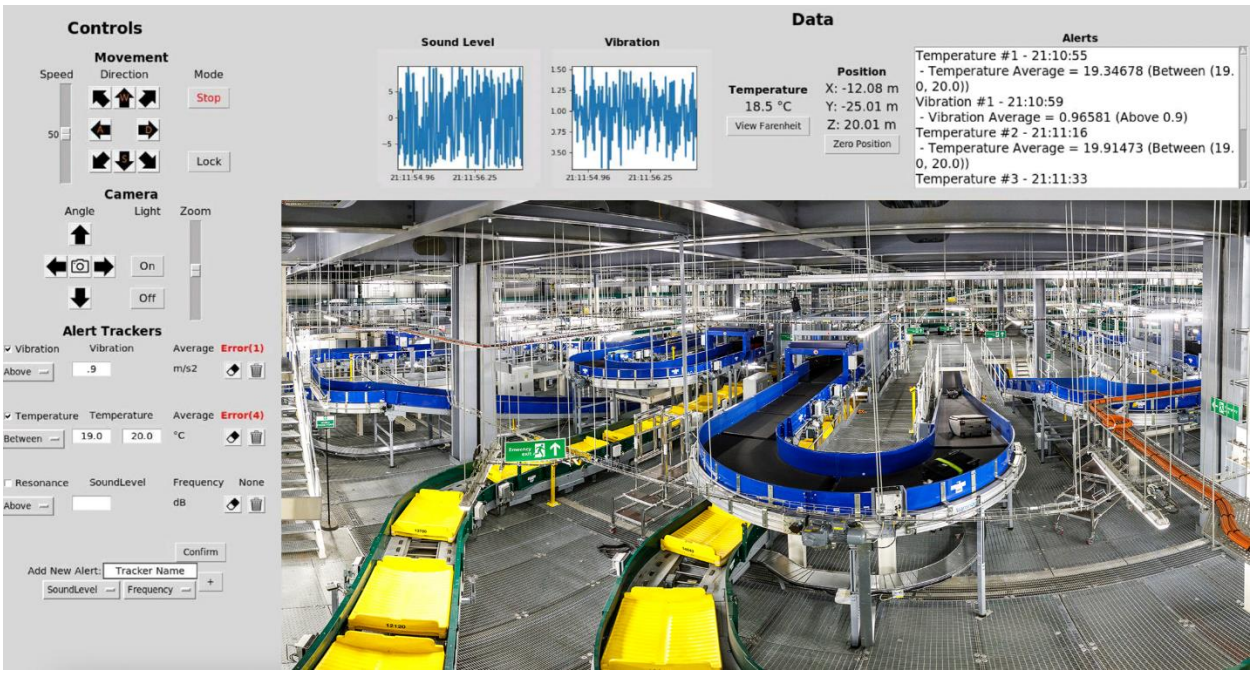


Figure 46: Dia-Bot Software and User Interface

In addition to prototyping the Dia-Bot, the team has devised a set of potential stretch goals and further recommendations for Vanderlande. One such goal is to improve the position tracking and display to track the bot's movement through a specific system by interfacing with its CAD model instead of only providing a 3D coordinate. Additional position tracking strategies have been identified which can further improve accuracy, such as a module from Pozyx which fell outside of our budget [20]. While manual movement and ride-along mode are useful transportation methods, another useful feature is an automatic movement mode – either by a pre-defined route or by letting the bot roam on its own. To further automate installation verification, computer vision techniques could be integrated to automatically analyze the camera's feed and identify problems visually. Additionally, establishing modes of communication between multiple Dia-Bots may allow for quicker and more advanced verification algorithms. Finally, even though the primary purpose of the Dia-Bot is to identify issues, methods may be discovered and implemented over time to fix certain problems without the need for an operator.

Due to the timelines of Vanderlande installing systems for their clients, the team did not get a chance to test the Dia-Bot in a proper shuttle or conveyor environment. Vanderlande operators would then carry out this field-testing step, sending the Dia-Bot through a new system to track the bot's movement and detect possible problems. This would allow proper calibration for collecting positional data within a complex conveyor space and identifying different vibrational, sound, and temperature patterns which may denote potential issues.

With the design scope and initial solutions have been defined, the team began prototyping and design simulation. The mechanical team has sourced OEM parts for the propulsion and suspension systems as well designd and will manufacture the structures of the chassis specifically used to house and protect the internal electrical components. For embedded systems engineering, the prototyping steps included breadboarding a simple setup to control simple servo and DC motors while reading IMU and camera data. The software team is creating a functional user interface on a Raspberry Pi using to control the prototype system. These initial steps to simulate a final design are crucial in ensuring the feasibility of the design solution and creating initial Dia-Bot functionality.

Figure 47 below shows the timeline for the project. Mechanical design simulations have been completed and predict a functional and structurally sound design. The team has sourced materials and completed production of each subsystem: movent, enclosure, power, electrical, and software, as described throughout the document. Finally, the full Dia-Bot was be assembled prior to display at the Georgia Tech Capstone Design expo.

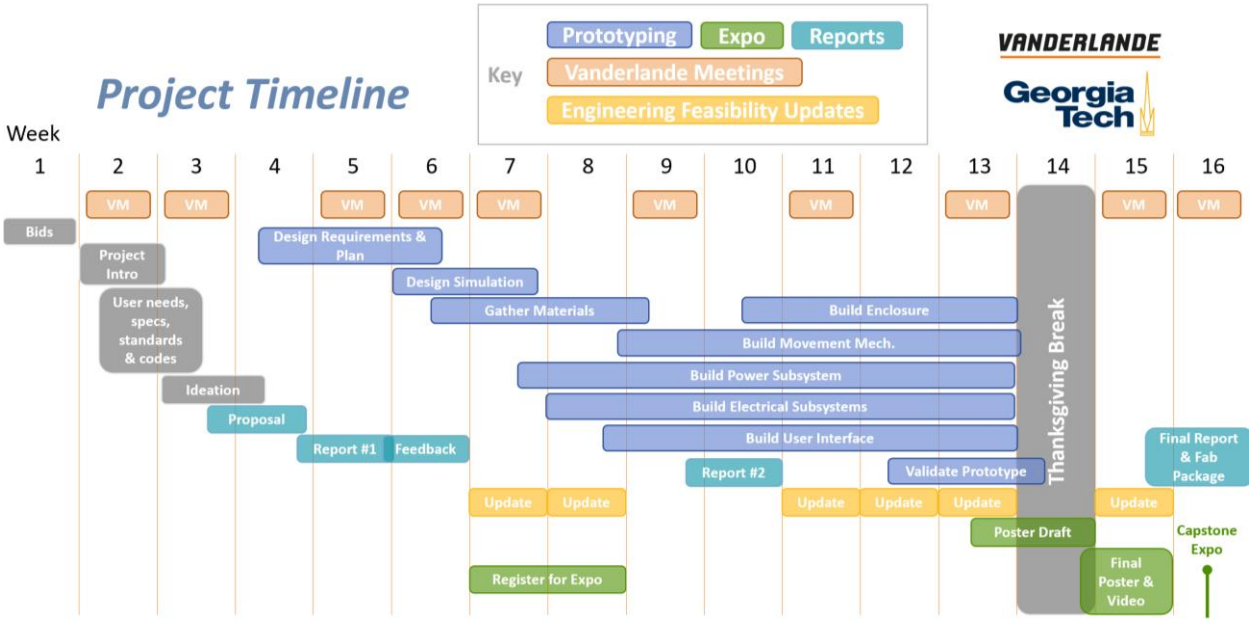


Figure 287: Project timeline, including design protyping, feasibility updates, reports, and expo tasks

The team has held weekly or as-needed update meetings with Arlo Bromley and Dr. Patrick Opdenbosch of Vanderlande Industries conducted via Microsoft Teams. Additional email communication for quicker and more urgent questions has been in use. Additionally, the team has met in a studio

session each week with primary and ME advisor Dr. Jianxin Jiao and has sent weekly update emails to Dr. Whit Smith and Dr. Vijay Madisetti for ECE advising.

References / Citations

1. “About Vanderlande and acquisitions - Vanderlande industries,” *Vanderlande*, 21-Jul-2021. [Online]. Available: https://www.vanderlande.com/about-vanderlande/. [Accessed: 26-Sep-2021].
2. “Robotyka Może Być prostsza z PRODUKTAMI HUSARION,” *FORBOT*, 04-Aug-2017. [Online]. Available: https://forbot.pl/blog/robotyka-moze-byc-prostsza-z-produktami-husarion-id21590. [Accessed: 26-Sep-2021].
3. “Standard Test Method for Evauating Response Robot Sensing: Visual Acuity,’ *ASTM Compass*, 01-Sep-2017 [Online]. Available: https://compass.astm.org/document/?contentcode=ASTM%7CE2566-17A%7Cen-US [Accessed: 26-Sep-2021]
4. “Standard Test Method for Evauating Response Robot Mobility Using Variable Hurdle Obstacles,’ *ASTM Compass*, 01-Mar-2021 [Online]. Available: https://compass.astm.org/document/?contentcode=ASTM%7CE2802_E2802M-21E01%7Cen-US [Accessed: 26-Sep-2021]
5. “Standard Test Method for Evaluating Response Robot Radio Communications Line-of-Sight Range,’ *ASTM Compass*, 01-Jan-2021 [Online]. Available: https://compass.astm.org/document/?contentcode=ASTM%7CE2854_E2854M-21%7Cen-US [Accessed: 26-Sep-2021]
6. “Wheels vs tracks: Advantages and disadvantages,” *Lite Trax*, 01-Mar-2018. [Online]. Available: https://litetrax.com/wheels-vs-tracks-advantages-disadvantages/. [Accessed: 26-Sep-2021].
7. J. Arellano, “Bluetooth vs. Wi-Fi for IOT: Which is better?,” *Very Possible*, 19-Jul-2021. [Online]. Available: https://www.verypossible.com/insights/bluetooth-vs.-wi-fi-for-iot-which-is-better. [Accessed: 26-Oct-2021].
8. “RaspberryPi models comparison: Comparison tables,” *SocialCompare*, 05-Sep-2021. [Online]. Available: https://socialcompare.com/en/comparison/raspberrypi-models-comparison. [Accessed: 23-Sep-2021].
9. “VNC Connect Features,” RealVNC, 2021. [Online]. Available: https://www.realvnc.com/en/connect/features/ [Accessed: 2-Dec-2021].
10. J. Palmisano “Actuators - DC Motors Tutorial,” *Society of Robots*, 2014. [Online]. Available: https://www.societyofrobots.com/actuators_dcmotors.shtml. [Accessed 20-Oct-2021].
11. “tkinter - Python interface to Tcl/Tk,” Python Software Foundation, 2001. [Online]. Available: https://docs.python.org/3/library/tkinter.html [Accessed: 2-Dec-2021].
12. D. Jones “Camera Hardware,” Picamera, 2013. [Online]. Available: https://picamera.readthedocs.io/en/release-1.13/fov.html [Accessed: 2-Dec-2021].
13. “Classes,” Python Software Foundation, 2001. [Online]. Available: https://docs.python.org/3/tutorial/classes.html [Accessed: 2-Dec-2021].
14. S. Shawcroft & D.P. George, “Adafruit CircuitPython NeoPixel,” CircuitPython, 2017. [Online]. Available: https://circuitpython.readthedocs.io/projects/neopixel/en/latest/ [Accessed 2-Dec-2021].

15. “Adafruit_LSM303_Accel,” Github, 2021. [Online]. Available: https://github.com/adafruit/Adafruit_LSM303_Accel [Accessed 2-Dec-2021].
16. “Raspberry Pi 4 Tech Specs,” Raspberry Pi, 2021 [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/ [Accessed 2-Dec-2021].
17. “multiprocessing - Process-based parallelism,” Python, 2001. [Online]. Available: https://docs.python.org/3/library/multiprocessing.html [Accessed: 2-Dec-2021].
18. “threading - Thread-based parallelism,” Python, 2001. [Online]. Available: https://docs.python.org/3/library/threading.html [Accessed: 2-Dec-2021].
19. “Introducing the Raspberry Pi Cameras,” Raspberry Pi, 2012 [Online]. Available: https://www.raspberrypi.com/documentation/accessories/camera.html#introducing-the-raspberry-pi-cameras [Accessed 2-Dec-2021].
20. “Hardware: Developer tag: Pozyx,” Hardware | Developer tag. [Online]. Available: https://www.pozyx.io/products/hardware/tags/developer-tag. [Accessed: 09-Dec-2021].

Appendices

Appendix A: Electrical Bill of Materials

Dia-Bot Electrical Bill of Materials				
Name	**Part Number**	**Cost (Per)**	**Quantity**	**Link**
Raspberry Pi 4	4295 from Adafruit	$ 35.00	1	Buy a Raspberry Pi 4 Model B – Raspberry Pi
Raspberry Pi Spy Camera	1937 from Adafruit	$ 39.95	1	Spy Camera for Raspberry Pi : ID 1937 : $39.95 : Adafruit Industries, Unique & fun DIY electronics and kits
Camera Tilt-Pan System	WC-003-300 from SuperDroid	$ 218.58	1	Camera 360 Pan and Tilt System - Standard (superdroidrobots.com)
DC Motors (Drive)	AM-4230	$ 45.00	2	https://www.andymark.com/products/johnson-electric-gearmotor-and-output-shaft
Accelerometer	13963 from SparkFun	$ 4.00	1	SparkFun Triple Axis Accelerometer Breakout - LIS3DH - SEN-13963 - SparkFun Electronics
H-Bridge Driver	L9110H	$ 1.50	2	https://www.adafruit.com/product/4489
H-Bridge Breakout	TB6612	$ 4.95	1	https://www.adafruit.com/product/2448
Temperature sensor	MAX31820	$ 1.95	1	https://www.sparkfun.com/products/14049
Microphone (Electret)	MAX4466	$ 6.95	1	https://www.sparkfun.com/products/12758
Buck Converter: 18V-5V	MPM3610	$ 9.95	1	https://www.sparkfun.com/products/18375
Buck Converter: 18V-12V	WG8-40S1203	$ 15.99	1	Amazon.com: DC Voltage Reducer Converter DC 8V-40V to 12V 3A 36W Automatic Step Down Up Voltage Regulator Power Converter Waterproof Module Transformer for Golf Cart Club Car : Electronics
Milwaukee 18V Battery	M18	$ 30.99	1	For Milwaukee 18V Battery Replacement | M18 3.0Ah Li-Ion Battery — Vanon-Batteries-Store (vanonbatteries.com)
LED Ring	1643 from Adafruit	$ 7.50	1	NeoPixel Ring - 12 x 5050 RGB LED with Integrated Drivers : ID 1643 : $7.50 : Adafruit Industries, Unique & fun DIY electronics and kits
Total:	-	$ 468.81	-	-

Appendix B: Mechanical Bill of Materials

Dia-Bot Mechanical Bill of Materials					
Name	**ID**	**Part Number**	**Cost (Per)**	**Quantity**	**Link**
1045 Carbon Steel Keyed Rotary Shaft, 10x300mm	1	1439K311	$24.21	2	https://www.mcmaster.com/rotary-shafts/keyed-rotary-shafts-5/system-of-measurement~metric/
Bronze Sleeve Bearings, 10mm	2	2938T763	$2.19	24	https://www.mcmaster.com/bushings/bearings-3/high-load-oil-embedded-flanged-sleeve-bearings/system-of-measurement~metric/
Shaft Collar, 10mm	3	G0318906	$2.73	24	https://www.zoro.com/ruland-manufacturing-shaft-collar-set-screw-1pc-10mm-steel-msc-10-f/i/G0318906/
Thick Aluminum Composite ACM Black Sheet 24in x 72in	4		$65.67	1	https://www.homedepot.com/p/Falken-Design-24-in-x-72-in-x-1-8-in-Thick-Aluminum-Composite-ACM-Black-Sheet-Falken-Design-ACM-BK-1-8-2472/308670300?MERCH=REC-_-pip_alternatives-_-308670312-_-308670300-_-N&
M4 x 60mm Female Threaded Brass Hex Standoff Pillar Spacer Nut 10pcs	5		$9.99	1	https://www.amazon.com/Female-Threaded-Standoff-Pillar-Spacer/dp/B0177VG4Q8
Yeah Racing 90mm Desert Lizard Two Stage Internal Spring Shock	6	YEA-DDL-090RD	$24.99	4	https://www.rcplanet.com/yeah-racing-90mm-desert-lizard-two-stage-internal-spring-shock-2-red-yea-ddl-090rd/p-tuqesxwqtm2xweue

Brecoflex Self-Tracking Timing Belt Two Pack	7	50 TK5K6/1100 V	$188.94	1	https://www.brecoflex.com/products/timing-belts/self-tracking-series/
3mmx3mm Machine Key Stock	8	92288A715	$4.14	1	https://www.mcmaster.com/machine-keys/machine-key-stock-5/system-of-measurement~metric/
½" Roller Bearing	9	25015T24	$7.24	8	https://www.mcmaster.com/bearings/wheel-axles-bearings-and-reducer-bushings/
Low-Profile Sleeve Bearing Carriage for 27 mm Wide Rail	10	6723K11	6.77	4	https://www.mcmaster.com/6723K11/
27 mm Wide Guide Rail for Low-Profile Sleeve Bearing Carriage	11	6723K2	17.50	2	https://www.mcmaster.com/6723K2/
Tetrix Max Hub and Axle Pack	12	W43004	$89.00	1	