Oakland University IGVC 2021 - Pinguino

THE PINGUINO INTELLIGENT ROBOTICS PLATFORM

Faculty Advisor: KaC Cheok

Team Captain: McKenzie King (mckenzieking@oakland.edu)

Team Members:

Kevin Huffman khuffman@oakland.edu
Christina Salama cmsalama@oakland.edu
Alyssa Musienko amusienko@oakland.edu
Nicholas Musienko nmusienko@oakland.edu
Christopher Mackenzie cmackenzie@oakland.edu
Anthony Giannattasio agiannattasio@oakland.edu
Dan Mocnik dmocnik@oakland.edu
Jacob Monks jacobmonks@oakland.edu
Tajwar Eram tzeram@oakland.edu
Eric Chan ericchan@oakland.edu
Angel Bautista bautistaromero@oakland.edu
Lauren Gregorio lgregorio@oakland.edu

Table of Contents

Introduction

Oakland University Robotics Association is proud to enter Pinguino into the 29th Annual Intelligent Ground Vehicle Competition (IGVC). Pinguino was first debuted in the 2019 IGVC and is now resurrected for this year's competition. Pinguino includes an autonomous system developed using a wide array of sensors including GPS, stereoscopic cameras, ultrasonic distance sensors, and LiDAR. The robot's movement is directed by two bi-directional independent electric motors to allow for high speed and accurate course navigation. There are also quadrature optical encoders on both of Pinguino's motors that are used to track wheel speed and direction in real-time. All software systems, including vision processing and map-based path planning, were simulated and integrated with the Robot Operating System (ROS) environment.

Organization

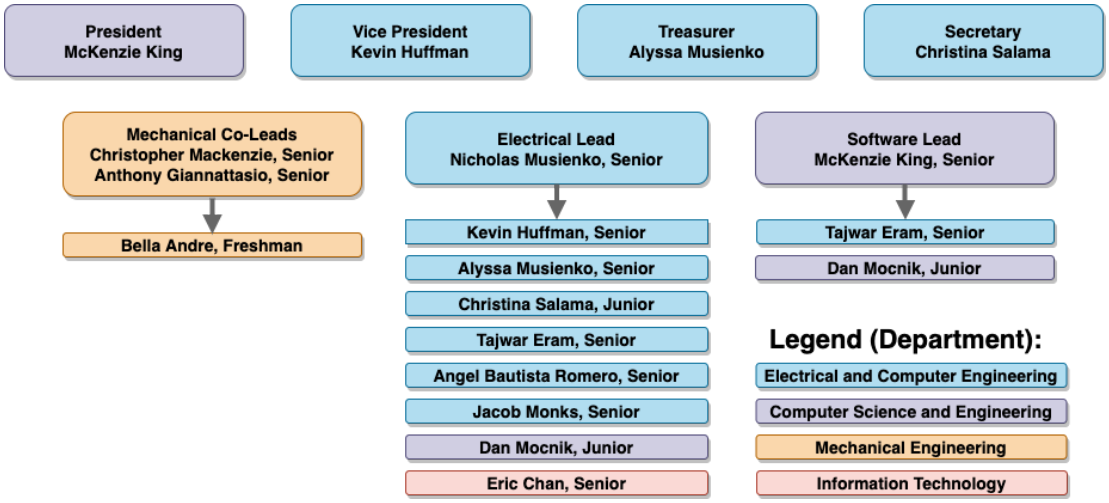

Figure 1: Oakland Robotics Association Organization Chart

Our team is divided into three subgroups: electrical, mechanical, and software. While there are separate groups for each discipline, our administrative board maintains channels of communication between them. Goals are decided by the team as a whole in a democratic process, then broken down into tasks and deadlines by the administrative board in collaboration with the group leads. Each member within these groups has their own strengths and areas in which they would like to grow and learn throughout the year. As can be seen in the organizational chart above, there is also a considerable amount of crossover between the electrical and software teams as the subject matters are often linked. This leads to a lot of diversification in technical skills and preferred topics of design.

Design Assumptions and Design Process

Design assumptions were made using the rules and requirements of the Intelligent Ground Vehicle Competition (IGVC). These restrictions combined with knowledge gained from the creation of previous IGVC robots set the size, weight, and height requirements that had to be met during the design process. The team also settled on a robot with two independent electric motors so the robot could easily

pivot in tight spaces while still being large enough to carry the necessary hardware, batteries, and payload to run the IGVC Course without stopping.

The design process, while being largely influenced by the requirements made in the design assumptions section, followed the classic "V-Model" which is demonstrated by the graphic shown to the right of this text. The team started with a simulated design where obvious failure points were located and resolved. Then the system was constructed which allowed for real-world issue rectification of the software, electrical, and mechanical subsystems. The process of testing for issues, correcting issues, and then retesting the system was then repeated as many times as necessary to develop a robot that the team feels is capable of safely navigating the IGVC Course.

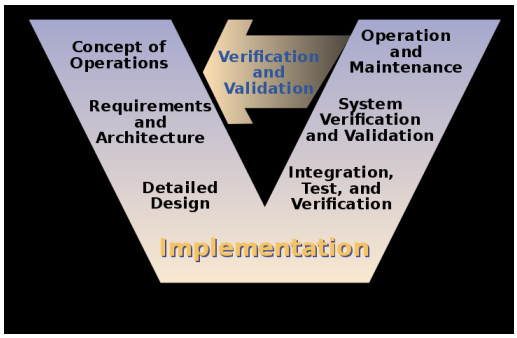

Figure 2: V-Model

Analyses were conducted on the final design to verify its theoretical feasibility to prevent any failure points once manufactured. Various tests were then performed on the system under consideration to verify its integrity, functionality, overall performance, and other key parameters before the design process was completed. If the system failed any of these tests, the cause for the failure was identified and reconsidered into a new design. The design process was repeated until the system passed.

Innovative Concepts From Other Vehicles Applied to Vehicle

For the design of Pinguino, the team opted for the implementation of various autonomous sensors including GPS, LiDAR, and stereoscopic cameras to ensure the robot can collect all of the necessary information from its surroundings to safely navigate the IGVC Course. The robot also implements an innovative embedded system design which includes an Arduino Due Microcontroller, two quadrature optical wheel encoders, a Sabertooth Motor Controller, and a Kangaroo Motion Controller. Upon initial system setup, the motion controller automatically tests the system to find the optimal values for a PID Feedback Loop. The motion controller then saves these values and acts as a "signal liaison" between the microcontroller and the Sabertooth Motor Controller. This allows for the continuous and smooth operation of the bidirectional drive system implemented on the robot without the need for vigorous calibration and system testing.

Innovative Technology Applied to Vehicle

Pinguino employs several innovative design features including the RGB LEDs surrounding the body of the robot. The lights have three modes including green for autonomous operation, yellow for drive-by-wire operation, and red for emergency stop. This design allows for the team to always be aware of the robot's operational state which allows for faster and safer system debugging. The robot also employs the use of multiple emergency safety stops including both mechanical and software stops. In the event of an emergency, the robot has multiple large red stop buttons on the outside of the chassis which can be easily activated by a bystander or the robot tipping over. The team also designed a software stop that can be controlled by an operator using a remote control.

Description of Mechanical Design

Overview

The mechanical team was directed to design and manufacture a robotic shell and all other mechanically related parts. In order to complete this goal, the team used Solidworks to design a 3D model consisting of existing and new parts, as well as sub-assemblies. New portions of the robot were designed with previous years' parts with room for improvement in mind. The overall design of Pinguino accounts for the needs of the electrical and software teams. These needs have led to strict specifications for many portions of the robot, including payload, sensor and laptop positioning, center of gravity, weight, waterproofing, and dimensional considerations.

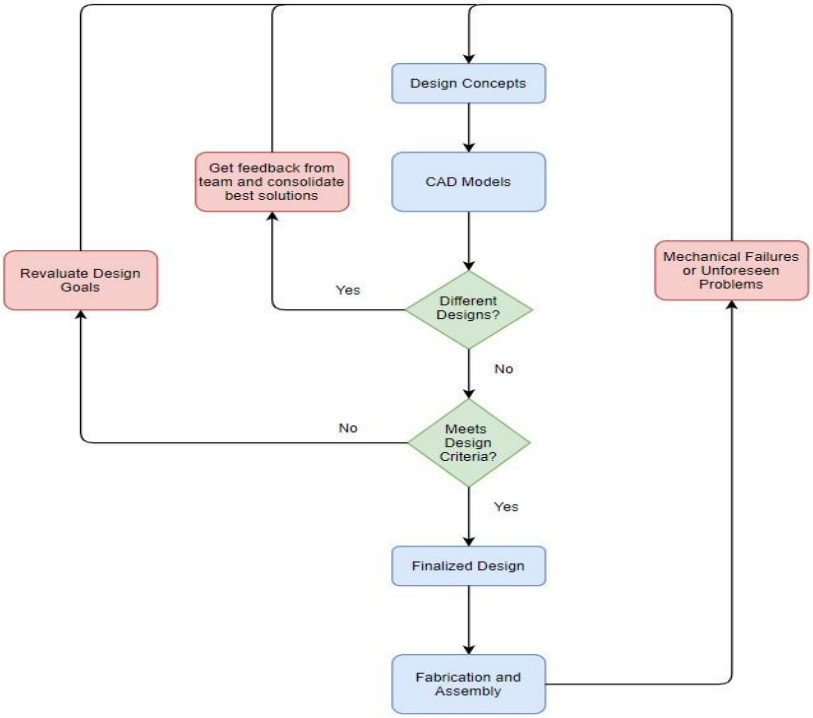

Figure 3: Mechanical Design Process Flowchart

Decision on Frame Structure, Housing, and Structure Design

Pinguino is framed by 1-inch hollow aluminum tubing. This was chosen because it is strong, light, easy to build with, and allows for modularity. Using such light material for the frame, especially the upper portion, provides more control over the weight distribution of the robot. We used this concept in coordination with placing the batteries at the base of the robot to keep the center of gravity as low as possible and as close to the center of the two drive wheels as possible. These pieces of aluminum are attached with custom steel brackets, designed and water jetted in-house by students using the campus machine shop. Pinguino's frame also has an internal housing space for electronic components. This housing space is used by two water-resistant, independent, and removable electronics boxes.

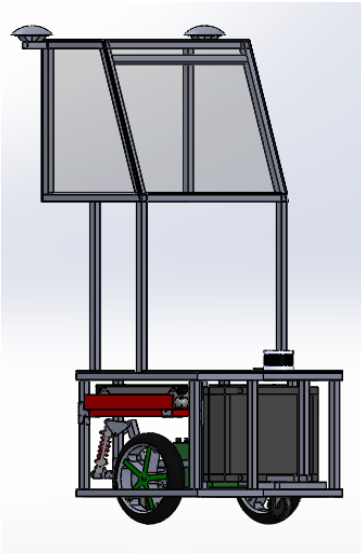

Figure 4: Full Assembly of Pinguino

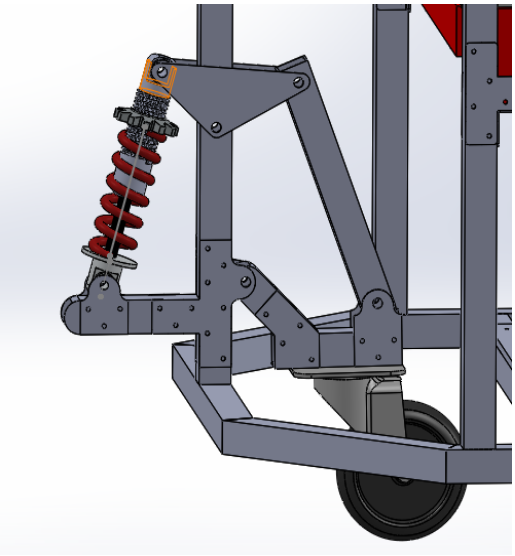

Figure 5: Side View of Suspension System

Suspension

Pinguino's robust suspension utilizes a hinged mechanical spring system. This custom suspension system prevents potentially large fore-aft rocking and even possible tipping as the robot traverses the outdoor course. This design consists of a single strut, aluminum/bracket connectors, and a swiveling caster wheel to increase efficiency and decrease complexity compared to previous models. A spacer was also added to the strut to adjust preload and vary the stiffness of the springs to accommodate specific terrain and/or weight of the vehicle. The softer spring allows for a smoother ride.

Weather Resistance

Pinguino is made with careful consideration of the elements since it must traverse an outdoor course in all weather conditions. This is why it has plexiglass body panels surrounding all important components, especially those close to the ground as the wheels can kick up mud, dust, dirt, and water in adverse weather conditions. While body panels do not keep *everything* out of the components, a vast majority of the electronics in Pinguino are housed in two electrical boxes. These boxes are sealed with epoxy on static edges and have rubber gaskets on any spots that can move such as the access lid. All electrical connections on these boxes are IP67 rated water-resistant to ensure our electronics are secure.

New Improvements

Over the past year, Pinguino has been tested and improved for several key weak points in weatherproofing and vibration damping. These improvements include new waterproof epoxy at fixed joints around the main compartment to make a better seal from the weather, flexible rubber gaskets installed for joints between panels that needed to be accessed frequently, and velcro strips at places where mechanical parts frequently slid across or rattled against one another to improve accessibility and mitigate unwanted vibration.

Description of Electronic and Power Design

Overview

The electrical and electronic system powering Pinguino was equipped and designed by dividing the system into two electrical boxes. One box contains the components for the embedded system while the other electrical box contains components used to process information from the robot's sensors. The overall design also took into consideration the ease of access for electrical components while testing and debugging the various systems.

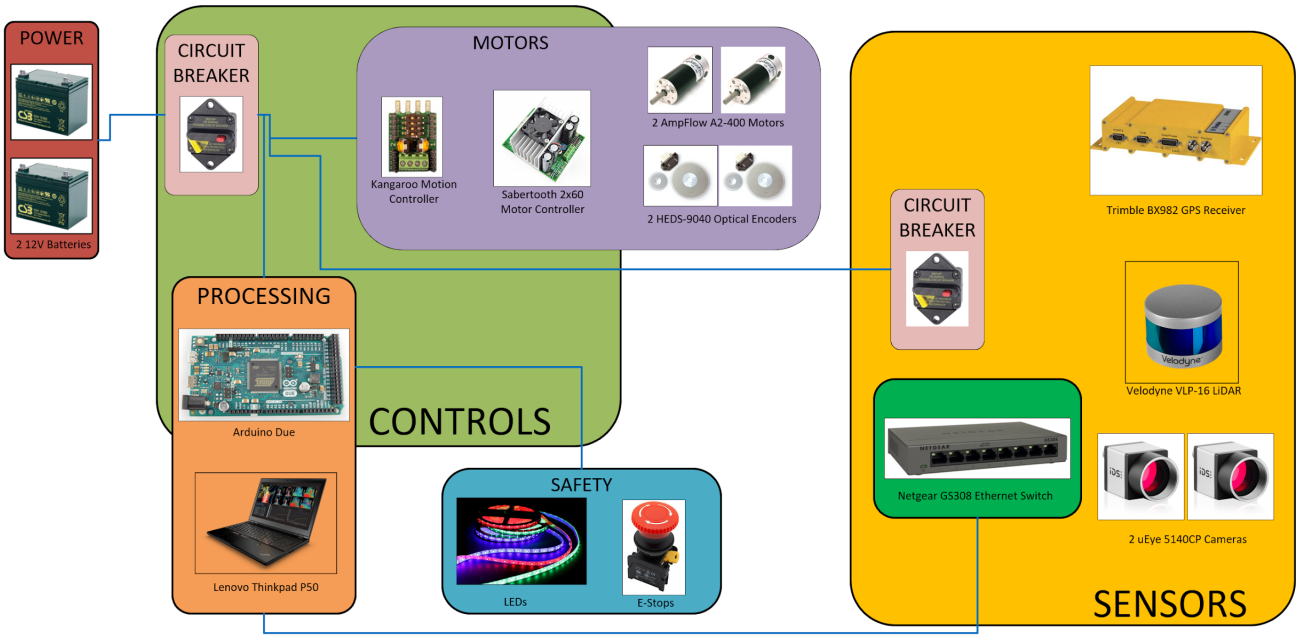

Figure 6: Diagram of Electrical System

Power Distribution System

Pinguino is powered by two 12V lead-acid batteries wired in series to provide 24V power. The batteries provide an operating time of approximately 1 hour and take 6 hours to recharge from dead. Various sensors require 12V, and to get this two DC-DC buck converters are used to provide power within each e-box. Powering the cameras involves Power over Ethernet, in which the PoE injector requires 48V to operate. Generating this 48V requires a DC-DC boost converter to step the voltage up from 24V. The entry point for each electrical box is a circuit breaker rated for the load of the box, 150A to the control e-box and 60A to the sensor electrical box. Within each electrical box, a fuse box exists to ensure components do not get damaged, with separate critical parts getting separate appropriate fuses. Because electrical components exist within normally sealed electrical boxes, connectors are necessary to allow power to transfer across the boundaries of the boxes. The chosen connectors are rated to allow the appropriate power and are rated at IP67. Voltage divider circuits are used to reduce voltage where necessary.

Electronics Suite Description

As with the twin electrical box setup, the electronics can be divided up into two major groups, controls and sensors. The control system was designed by Oakland University students around the Arduino Due microcontroller and a Sabertooth 2x60 motor controller. To ensure that the motors spin correctly, optical quadrature encoders are mounted to the shaft of each motor, their outputs getting routed to a Kangaroo motion controller. The Kangaroo is a self-tuning closed-loop PID controller designed specifically for communication with the Sabertooth, which allows for position and speed readings from the motors. Powered off the Arduino Due is a BNO055 IMU, for more precise measurements of movement. An RC receiver is also powered off the Arduino Due, and the RC transmitter acts as both a

user input for the robot and as the emergency stop. Also included is a perfboard containing several custom circuits. An LED driver, battery voltage sensing circuit, logic step-down circuit, and software e-stop implementation are all custom-made for this application and are found on this perfboard.

The sensor system was designed via collaboration between the electrical and software subteams. An Ethernet switch collects the data from the GPS, LiDAR, and cameras for transmission to the computer.

Computer. The computer acting as the core of the robot is a Lenovo Thinkpad P50. This specific model boasts a 2.8-3.7GHz Intel Xeon Quad-Core Processor, 16GB of RAM, a 256GB SSD, and an NVIDIA Quadro M2000M GPU. This laptop has the processing power necessary to collect the present data and run the appropriate computations.

Figure 5: GPS Unit

GPS. The chosen GPS for the platform is the Trimble BX982 GPS receiver. It operates at 50Hz and has an accuracy of up to 8mm. For this GPS, there are two antennas, allowing for more accurate readings.

LiDAR. The Velodyne VLP16 LiDAR was chosen for its 16 scanlines and its accuracy of 3cm. It has an angular resolution of 2°, with a 360° field of view and a rotation rate of between 5 and 20Hz. It publishes its data over Ethernet and is capable of generating up to 0.3 million points per second. These specifications are plenty powerful for the given application and provide enough data to determine appropriate behavior.

Figure 6: LiDAR

Camera. Two uEye 5140CP color cameras are used in this platform. Each camera is capable of PoE operation, allowing both power and data to travel over a single Ethernet cable each. Each camera is capable of 88 frames per second operation with a 1280 by 1024 resolution.

Figure 7: uEye Camera

Safety Devices and Integration

Safety is a key consideration in the design of Pinguino. Each electrical system on Pinguino has a dedicated circuit breaker that prevents excessive current from being drawn by critical components and provides a physical disconnect for power when working on the robot. Fuses help serve a similar purpose for components like the motors and sensors, preventing damage to these components in the event of an excess current draw. There are two physical Emergency Stop buttons located on the sides of Pinguino, both of which are wired into relays that control the power delivery to the main motors, as well as a software stop relay controlled by a switch on the remote controller. These three e-stop systems combined help ensure that in the event an unexpected behavior is observed during operation, power delivery to the motors can be stopped physically on the robot as well as remotely from one hundred feet away if necessary. The top speed of the motors has been restricted to five miles per hour to comply with the rules set by the IGVC and lateral acceleration limits have also been set in order to prevent Pinguino from tipping over. In the event that Pinguino does not receive an input from the controller after a duration of 25 milliseconds, the electronics systems on Pinguino will automatically shut down in order to prevent unintended user inputs from being registered in the event of a controller malfunction. The current power status of Pinguino can easily be viewed through a series of color-coded LEDs and can be used to identify if the robot is currently restricted by a hardware or software stop, whether the robot is being controlled remotely or autonomously, and when the batteries are low. This battery level indicator light is a critical

part of the safety design for Pinguino due to the fact that some components may begin to behave unpredictably in the event they do not receive the proper voltages needed to operate normally.

Table 1: LED Indicator Legend

Color				Function	
Green	Yellow	Red	Blue	Solid	Blinking
Normal Operation State	Software e-Stop Engaged	Hardware e-Stop Engaged	Low Battery	Remote Operated	Autonomous Operation

Description of Software Strategy and Mapping Techniques

Overview

The many facets of Pinguino's software design allow it to navigate the road-like IGVC course with great precision. Data is taken in through our perception suite, processed through a variety of methods, and communicated to our lower-level software and controls described in the electrical design portions above.

Obstacle Detection and Avoidance

Localization and Mapping

For this topic, Pinguino employs a Velodyne VLP16 LiDAR, a Trimble BX982 Global Positioning System (GPS), and a Bosch BNO055 Inertial Measurement Unit (IMU).

LiDAR. LiDAR is the source of cone detection in our software suite. We use the Point Cloud Library (PCL) to form a Point Cloud Dictionary (PCD) from the point cloud constructed by the individual data points provided by the VLP16. The first step in processing the large amount of raw data is to remove the planes from the dataset. This removes a large amount of unnecessary data so that subsequent processing is significantly faster. Next, cylindrical segmentation is used to find cone-shaped objects in the remaining data. Those shapes are then separated from the rest of the data and added to an array of barrel locations and sizes.

GPS. Our GPS is primarily used to confirm wheel speed and localize the robot on the course in collaboration with the IMU. Pinguino's GPS utilizes the benefits of dual antennas to obtain heading at an accuracy of 0.09 degrees. It also provides a position with 8mm of accuracy.

IMU. Our IMU provides us with the orientation and linear acceleration of the robot. After it is sent to the PC and transformed into vectors, it is fused with the GPS. This allows for the calculation and estimation of current and future poses for planning and recovery purposes.

Figure 7: Hough Transform Line Overlay

Vision and Lane Keeping

Our vision system consists of two uEye cameras. These cameras are mounted high up on the robot, close to the primary framing structure, and under sloped body panels for a variety of reasons. The paneled shielding offers resistance from rain and the proximity to the frame keeps the cameras as stable as possible and well-calibrated with the localization of the robot since the cameras are as stationary as possible. The mounting location gives us as wide a field of view as possible. Both cameras are calibrated each day with our in-house calibration tool that guides our software users through an OpenCV-based calibration procedure. This allows us to remove any lens distortion and keep our map generation as accurate as possible. To detect the lines, we use a Probabilistic Hough Transform. They are then added to the cost map by doing the following steps. Lines are first converted to the robot coordinate system and overlaid onto the point cloud. Collinear and semi-random points are chosen from this and fit over the point cloud in the field coordinate system, the base frame, in order to have them in reference to the barrels. Our lane-keep function is not designed to go *between* lines, but instead to avoid them in general and fall back on GPS for direction as needed. This design consideration was made so that if the robot can only detect one side of a lane due to a wide lane or a heavy yaw angle, it will still be able to navigate.

Sensor Fusion

For sensor fusion, Pinguino uses an Extended Kalman Filter (EKF). It is used to fuse our location data sources, the GPS and IMU. The inputs are converted into an odometry message and given to the EKF for processing. Once the data is processed, it is synchronized and recombined as an output to a new node.

Software Strategy and Path Planning

Pinguino's path planning system consists of a trajectory rollout sampling algorithm. It is able to take in sample trajectories starting from the robot's current position and score them based on safety, or cost, and efficiency in achieving the goal.

Map Generation

The software consists of a global and local costmap. The global costmap is a static map of the areas of the field that the robot has seen. The local costmap is dynamic, meaning that it can adapt based on the changed surroundings of the course. If a judge is in the area around the robot on the course and is

detected to be moving, those changes will adapt on the local costmap, but not the global costmap unless updated by another portion of the software.

Goal Selection and Path Planning

The path planning algorithm used is Dijkstra's algorithm. This will populate the costmap and continue updating it based on the local trajectory planning to the overall global costmap.

Additional Creative Concepts

The portion of our software that we find the most creative is the Probabilistic Hough Transform. We implemented it through openCV and once the borders of the lines are detected, we have a visualization program to show us the lines overlaid onto the camera image. This allows us to accurately tune the parameters behind the transform and get the best line detection for whatever lighting, weather, or field conditions we may be facing.

Description of Failure Modes, Failure Points and Resolutions

Vehicle Failure Modes and Resolutions

In the event that the robot becomes stuck and cannot determine any valid paths to take, the robot has procedures in place to make sure that it is able to recover from this situation. The primary method for accomplishing this is done by clearing the local cost map and populating it with new sensor data. This allows for any errors to be corrected and enables the path planner to create a new path with the newer map. An example of a situation where this method can be useful in case of a misplaced or erroneous obstacle on the map which prevents the goal from being achieved.

Vehicle Failure Points and Resolutions

Pinguino takes advantage of several hardware failure point resolutions. In case of any type of malfunction, the robot is equipped with both software and hardware e-stops (emergency stops). There are two physical e-stop buttons on each side of the robot in easy-to-access locations, and the software e-stop can be triggered by a switch on the remote controller. Relays that are connected to the motors will disengage and prevent power from being applied to them if the controller is failing or turned off, or if any of the e-stops are engaged. If either of the motors were to stall and pull more current than they are allotted from the power distribution system, the 60A in-line will blow in order to protect the system. Alternatively, the circuit breaker can trip which will disconnect the electrical box entirely. If any bad motor commands are issued by the Arduino or the signal from the Arduino reaches the Sabertooth with some interference, a unique value is issued with each motor command which is compared to the command itself. If a discrepancy is found, that command will be thrown out and disregarded. If an electrical component is behaving irregularly due to power issues, the Arduino can provide some data to help locate the source of the issue. A voltage sensing circuit that is present in the electrical box can be used to determine if a battery needs to be replaced with one that is fully charged. The Arduino can also be used to determine if any of the e-stops are engaged.

Loose wires could also have the potential to get caught in any moving parts. This was remedied by using different types of cable management, such as zip ties or cable sleeving. If the robot were to structurally fail, aluminum structural members can be easily drilled out, replaced, and riveted back together.

All Failure Prevention Strategy

To ensure that the number of failure points and modes are reduced, the robot has been well-maintained, all of the connections were double-checked thoroughly to guarantee that they are strong, correctly mated, and still have all associated waterproofing intact. Pinguino was designed with modularity in mind, so if a component or sensor is failing, it can be removed and replaced easily. This modularity also allows the robot to be better tuned to the environment in which it is being used, which further minimizes potential failures.

Testing

Pinguino's software was tested by utilizing it in simulations to determine usability, performance, and to get an idea of any potential failure modes. Critical hardware systems, such as motors, sensors, and power distribution were assembled and built outside of the robot for testing purposes. Once the viability of these components was proven, they were then redesigned to comply with the amount of space and form factors available. Once Pinguino was fully assembled, each subsystem underwent testing in a student-created version of the actual IGVC course.

Simulations Employed

Simulations in Virtual Environment

The tools that were used to simulate Pinguino include Gazebo, which is a powerful robot simulation tool used to test the effectiveness of the robot and performance in an environment similar to that of an IGVC course. Another tool we have employed is Rviz, which is used to visualize data that is being processed internally within the robot, such as sensor data, transformation trees, costmaps, or the current path plan. Gazebo and Rviz were used in collaboration with each other to test and troubleshoot components of the robot's systems individually, as well as the entire robot.

Theoretical Concepts in Simulations

The theoretical concepts that are provided by the visualization from Rviz are very helpful in creating and tuning the robot's navigation stack. Rviz allows us to view the relationships between the sensors of the robot and the ground plane, and visualize the data in relation to the robot as well. This makes us able to optimize the placement and configuration of our sensor suite in order to obtain optimal inputs for data processing.

Performance Testing to Date

With all the testing completed so far, the gathered values are shown in the table below. Pinguino's maximum speed was calculated from the RPM of the motors while under a nominal load, and the diameter of the wheels. The robot's ramp climbing ability was calculated with the amount of known torque generated and the weight of the robot. However, the actual value has differed from our calculated value. Battery life was considered by summing the maximum current draw from the robot electrical system, then dividing it by the capacity of the batteries in their current configuration. Finally, Pingino's reaction time and maximum obstacle detection window are simply determined by the limitations of our LiDAR, since it is the sensor that is in charge of detecting objects, and has the slowest refresh rate of all the sensors within Pinguino.

Table 2: System Parameters as Calculated from Known and Obtained Values

Value	Obtained By	Predicted Performance
Speed	Calculated	4.8 mph
Ramp Climbing Ability	Calculated Experimental	19° ~ 30°
Battery Life	Calculated (Worst Case) Calculated (Best Case)	~ 0.5 hr ~ 1 hr
Reaction Time	Limited by rate of LiDAR data	~ 50 ms
Maximum Obstacle Detection	Limited by LiDAR specifications	100m

Initial Performance Assessments

Pinguino's initial performance shows significant improvement from past years. The opportunity for growth of our team in this past unprecedented year was not left to waste. Pinguino is coming back stronger than ever.

Cost Report

Table 3: Model, Cost, and Number of Items Required.

Part	Model	Quantity	Price Per Unit	Cost Total	Cost to Team
LiDAR	Velodyne VLP-16	1	$8,000	$8,000	$0
GPS	Trimble BX982	1	$5,000	$5,000	$0
Laptop	Lenovo P50	1	$1,749	$1,749	$1,749
Network Switch	Netgear GigE	1	$73	$73	$73
Microcontroller	Arduino Due	1	$40	$40	$40
PCB Printing	Custom	1	$25	$25	$25
IMU	BNO055	1	$35	$35	$35
Battery	Lead-Acid	2	$210	$420	$420
Motion Controller	Kangaroo x2	1	$51	$51	$51
Wheel Encoders	E5 Optical Encoders	2	$62	$124	$124

Cameras	uEye 5140CP	2	$313	$626	$0
Camera Lens	Edmund TECHSPEC UC	2	$168	$336	$0
Motor Controller	Sabertooth 2x60	1	$190	$190	$190
Mechanical	Raw Materials		$200	$200	$200
Electrical	Assorted Materials		$300	$300	$300
			Total Cost	**$17,169**	**$3,207**