SNOWSTORM: IGVC 2016 DESIGN REPORT

UBC Snowbots

University of British Columbia

Arjun Sethi
Jannicke Pearkes
Kirk Wong
Angy Chung
Kain Xu
Jacky Sun
Benjamin Chow
Jame Lee
Andrew Kang
Charles Cai
Edward Li
Jason Raymundo
Emma Tam
Sze-Kei Luk
Myra Niu
Shaizaib Faisal
Jay Paul
Mitchell Ang
Shahzoor Safdar
Winnie Mui
Yvonne Wu
Vivy Wang
Aly Abouzaid
Gurjeet Singh
Michelle Mak
Clarissa Gunawan
Gareth Ellis
Tracey Lui
Aaron Mishkin
Simon Jinaphant
Nick Wu
Maya Schuller
Emmanuel Sales
Eugene Shen
Vincent Yuan
Finn Hackett
Jagjot Jhajj
Jamie Ye
Aashish Karna
Anni Wang
Jeffrey Doyle
Valerian Ratu
Zara Lim
Tracy Sun
Maxim Tsai

Captain:

Arjun Sethi

snowbots.ubc@gmail.com

INTRODUCTION

Snowstorm is an upgraded robot designed and constructed by UBC Snowbots this year. *Snowstorm* has a square chassis supported by four wheels. One LIDAR sensor is used for obstacle detection, and a camera on a tower is used for computer vision. Multiple software strategies are employed to allow *Snowstorm* to navigate the IGVC course swiftly and precisely. This report will describe our team's organization, design strategy, innovations, and the mechanical, electrical and software elements of our vehicle. The report also includes a detailed cost analysis and will end with our aspirations for the upcoming competition.

TEAM ORGANIZATION

UBC Snowbots consists of UBC students from a variety of engineering departments, the faculty of computer science, and also students from the faculty of business. There are four main divisions of the team – the administration team, mechanical team, electrical team, and software team as one can see in Figure 1 below. Each team is managed by the team leads. As we are a large team, each division is further split up into sub-divisions and then into project groups. Typically 2-3 students work on a particular project. The entire team meets every week for three to four hours and project sub-divisions meet at additional times as required by their tasks.

UBC Snowbots
Team Captain: Arjun Sethi

Admin	**Mechanical**	**Electrical**	**Software**
Team Lead: Kirk Wong Activities: funding, budget, trip planning, safety	Team Leads: Jason Raymundo, Edward Li Activities: design, fabrication, repairs, mechanical safety	Team Lead: Winnie Mui Activities: firmware, electrical layout, circuitry	Team Leads Vincent Yuan, Simon J., Gareth Ellis, Finn Hackett Activities: vision, mapping, pathfinding,

Figure 1. Team Organizational Structure

DESIGN PROCESS

The team met at the beginning of the year and set goals for the two competitions we will be attending and how we can make a robot that will be able to fit the needs of both competitions. We also had a detailed discussion about the problems we faced in the previous competition and how they should be resolved. This resulted in a complete overhaul of the previous vehicle and the implementation of a new software strategy, new firmware, and new mechanical design—all of which will be discussed in detail in this report.

MECHANICAL DESIGN

The robot can be divided into three distinct modules: the drive train, housing and tower. Throughout the design process, ideas were communicated amongst our sub-divisions, and were later unified to form the robot. SolidWorks was used to design all components to confirm all design parameters. Sufficient weatherproofing has been discussed in the safety and reliability section.

 

Figure 2. Snowstorm-From Design to Reality

MATERIAL SELECTION

The robot structure is composed of four materials: aluminum 5052 sheet metal, 304 stainless steel shaft, corrugated plastic, and acrylic. All manufacturing and fabrication processes were handled in-house, utilizing rapid prototyping techniques, ie. waterjet cutting and the utilization of a pneumatic press brake. Aluminum was chosen for its machinability, strength and light weight. 304 stainless steel shafts were chosen to achieve high durability of the drivetrain. Certain parts of the robot require weatherproofing and clear acrylic was used to shield the components without obstructing the view. In cases where opaqueness and thermal shielding were desired to protect against direct sunlight, white corrugated plastic was used.

DRIVETRAIN

The drivetrain is composed of two modular sections that are connected via two large brackets and a large center belly pan. These brackets were carefully designed to handle expected loads, while keeping the overall infrastructure linked. Each drivetrain module consists of two, high wheels with accompanying motor gearbox assemblies. Given the pneumatic wheels and our experiences from the previous year, we decided we did not need additional suspension systems.

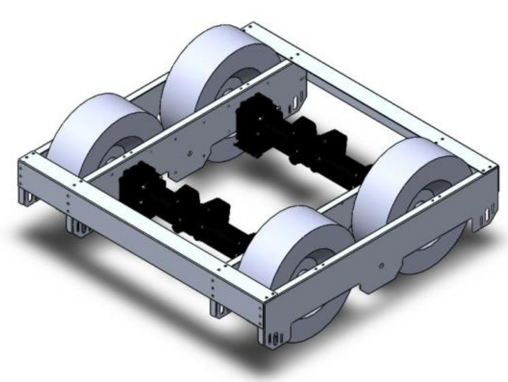

Figure 3. SolidWorks Model of the Drivetrain and Powertrain

POWERTRAIN

Each wheel is driven with a motor gearbox assembly, each with a gear ratio of 30:1. Each gearbox is connected to a 12V DC brushed motor. Calculations were made in order to assess whether the motors and accompanying gearbox would be sufficient to propel the vehicle.

HOUSING

The housing module holds robot's electrical system and the computer.

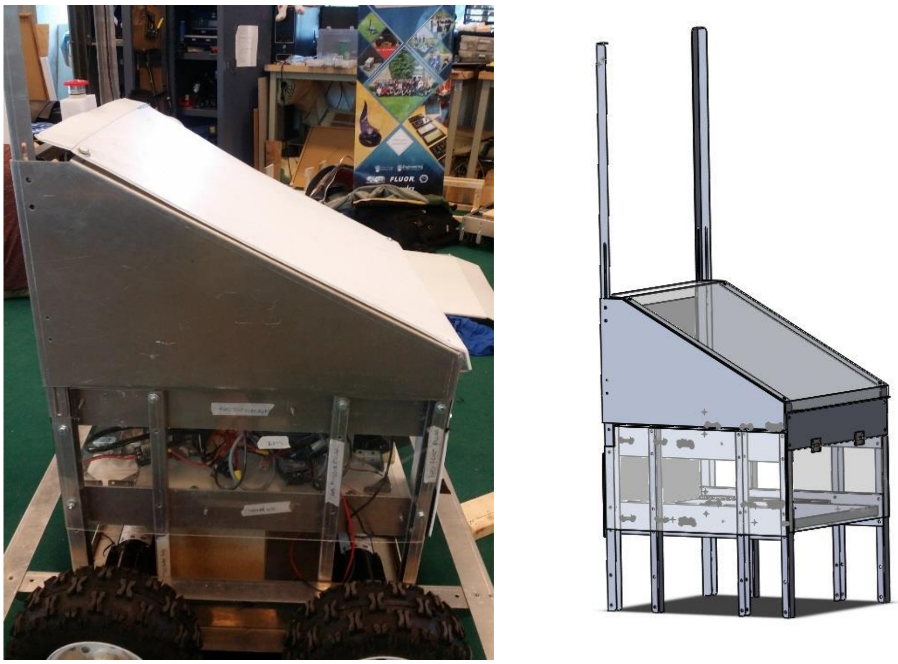

Figure 4. Housing Model and Pop-out Platforms

Since the computer has the highest demand for accessibility, the computer compartment is retractable and was designed with the programmer's seated height in mind.

Figure 5. Comfortable Seating Position for Direct Programming

The electrical compartment is accessible from the opposite end of the robot. Here, our engineers can perform electrical maintenance as needed without disrupting our programmers. To further improve the accessibility to the electrical system, we designed the compartment to be removable. Finally, since most of the human interaction to the robot occurs from the back end, we conveniently placed all of the electrical switches and payload there.

TOWER

The purpose of the tower is to optimize the position of our robot's sensors. This module holds our camera and our GPS antenna. The camera is strategically elevated in order to increase our robot's range of view. The height was also used as a way to isolate our GPS antenna from any interference coming from the other electronics. The tower's height can also be adjusted in order to allow for vision placement optimization.

We chose a strut channel as our platform to mount our sensors. Strut channels allow for easy positioning and secure mounting of sensors such as cameras and antennas. It is made out of fiberglass to eliminate any interference to the antenna that may be caused by nearby metal.

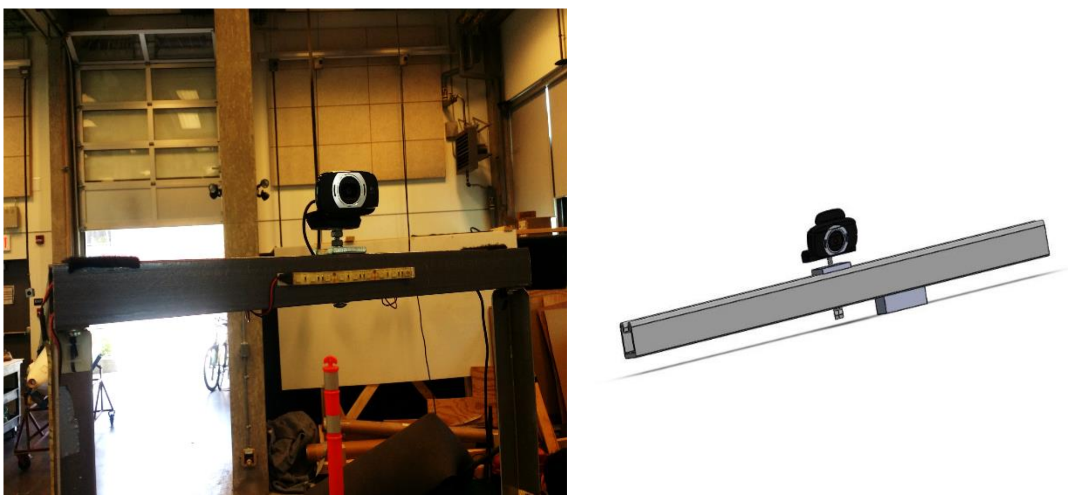

Figure 6. Tower Module with Camera

RAMP CLIMBING CAPABILITY

Given that the gradient is to not exceed 15% (~8.5 degrees), and knowing the characteristics of our robot, some simple calculations we performed in order to assess whether or not our robot can traverse through inclines found in the Auto-Nav course. The following table lists our values used in the calculations.

Table 1. Ramp Climbing Analysis

Parameter	**Value**
Weight of Vehicle, m	60.7 kg
Gear ratio	30
Max motor output	0.6 Nm

Number of motors	4
Wheel radius	0.17 m
Coefficient of Friction, mu	0.35

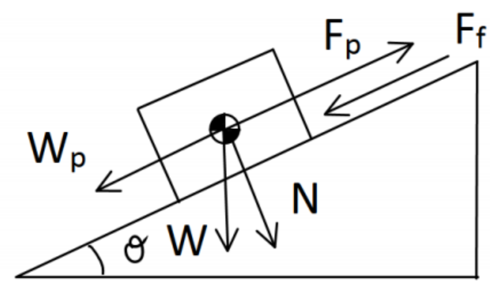

Figure 7. Free Body Diagram

By using Newton's laws of motion and using a free body diagram, we can calculate the forces.

From our calculations, we have a pushing force of 423.5 N, and a static weight of 88 N along the incline. This means that our vehicle is capable of keeping itself static on an 8.5 degree incline. The static friction force is 206 N. On maximum incline, our vehicle can propel itself since it can overcome static friction and weight.

SPEED

Given that the maximum speed 5 mph and minimum speed 1 mph in the competition and the characteristics of the robot, we can calculate whether our robot speed is within the restriction.

Table 2. Speed Analysis

Parameter	Value
Max motor RPM	56
Gear ratio	30
Max motor output	0.6 Nm
Number of motors	4
Wheel radius	0.17 m

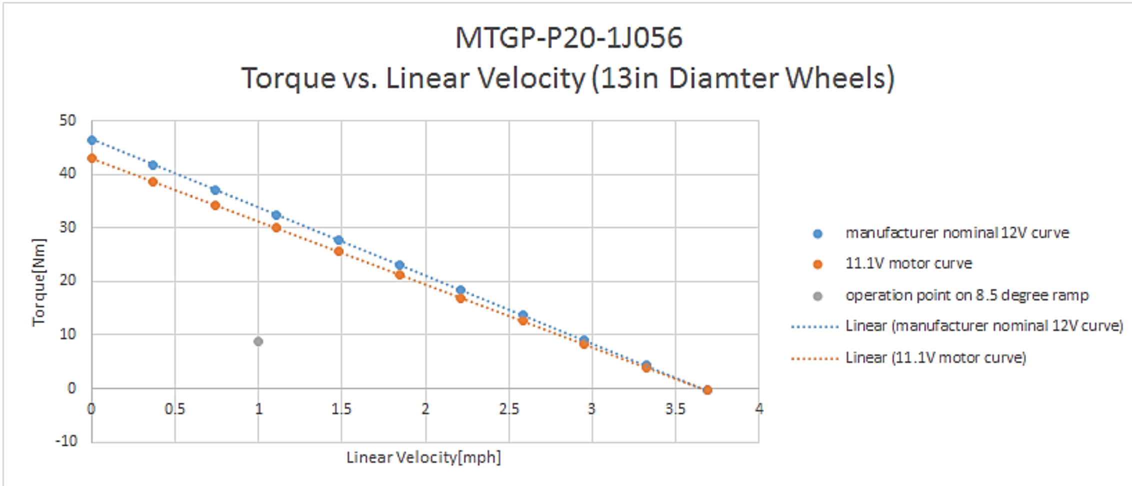

Figure 8. Free Body Diagram

Based on specification data point we found for the motor and converted to the speed of the robot, we concluded that the maximum speed of our vehicle is 3.6 mph, and thus within the speed limit.

Moreover, based on the friction calculated in the climbing ability section, we can calculate the maximum torque the motor needs to provide on an 8.5 degree ramp to be 35 Nm. Therefore, the operation point of the robot running on an 8.5 degree ramp is represented as the grey dot in the plot, meaning that our robot can safely climb the hill.

ELECTRICAL DESIGN

The mandate for this year's electrical division was to ensure a reliable and safe electrical system. Safety features include fuses to prevent overcurrent, and voltage monitors to prevent over/under-charging the batteries.

POWER DISTRIBUTION SYSTEM

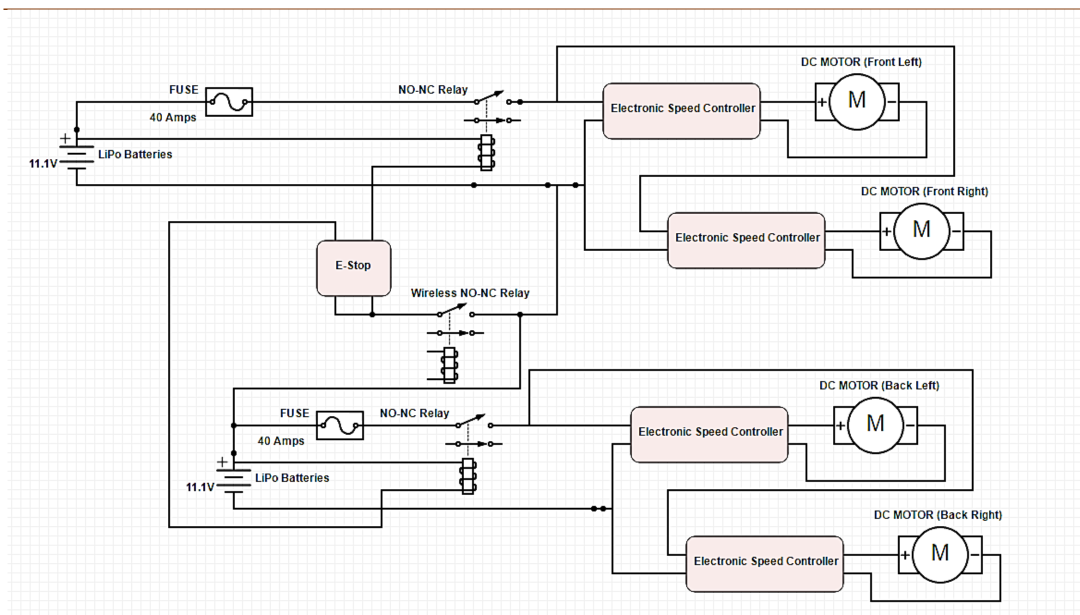

Figure 9: Circuit diagram of our power system.

The power for the four motors onboard *Snowstorm* is provided by four lithium polymer (LiPo) batteries capable of providing 11.1V for 7500mAH. The circuit is secured down to decrease the amount of loose wires and to ensure reliable connections.

The motors used are DC motors. To ensure that the voltage drop that occurs with the drainage of the lithium polymer batteries does not cause the robot to veer one way, the front two motors are connected to two lithium polymer batteries, while the back two motors are connected to other lithium polymer batteries. There are two separate circuits with a mutual ground as we wanted to decrease the maximum current running through our wires for safety reasons. We ensured that each component in the circuit that power from the motors would go through would be capable of handling at least 41 amps. The motors were rated at 15.70 amps full-load amperage each, leading to 31.4 amps being our maximum current draw. To ensure our robot would be safe, we checked each component to ensure that each component of our circuit that handled the high current would be able to withstand at least 40 amps, and put in 30 amp fuses.

The battery is connected in series with a relay and an E-stop. When the E-stop is pressed, power to the relay is cut, which will cut power to the motors. There is a wireless relay that will cut power to the E-stop relay. The two relays function as an AND gate -- unless both the E-stop is released, and the wireless relay is turned on, the robot's motors will not get any power and will be unable to move. Both these relays are normally open - ensuring that if the relay fails, the robot will no longer run. There are also multiple fuses in our circuit to prevent short-circuiting and ensure that the current levels will not exceed the maximum we can handle. We used a fuse block so that burnt out fuses can easily be replaced.

Two additional 11.1V LiPo batteries connected in series power the LIDAR.

Our power system includes an external battery for the laptop to allow for extended use. Our camera, microcontrollers, and other small sensors are powered through a USB connection to the USB hub connected to the laptop.

OPERATING LIFE

As our robot has three separate electrical systems powered by three distinct power sources, the operating life of the entire system is limited by the one with the shortest operating life. In this situation, the limiting factor is the time the motors can run. Since we have four motors powered by four batteries, this essentially works out to one battery powering one motor. Each battery is rated 7500mAh, while the motors are rated 15.70 full-load amperage (FLA). However, this is FLA, and the motors don't draw nearly as much current since we aren't carrying a large load. We also do not run the motors at full speed, and generally send a pulse-width modulated signal with a duration of between 50-70%.

The running load amperage (RLA) is estimated to be 75% of the FLA, which is approximately 11.78amps. Since we were using LiPo batteries, we are only able to safely use 70% of the battery's power capacity, which is 5.25Ah. This leads to 27 minutes of operating life. However, since we are not running the robot non-stop at full speed, and instead running it at approximately 70% it has an operating life of 38 minutes per charge.

EMERGENCY STOP SYSTEM

As stated above, our emergency stop system on *Snowstorm* is implemented by using a 1NO+1NC emergency push button located at the center of the robot's width and at a height of approximately 3.5ft from the ground. Pressing this will cut power to the relay that controls power going to the motors. Connected to the circuit that powers the E-stop relay is a wireless one which must also be closed before the robot will run. Both our emergency stop switch and our wireless switch are hardware-based and do not involve any software to stop the vehicle.

FIRMWARE

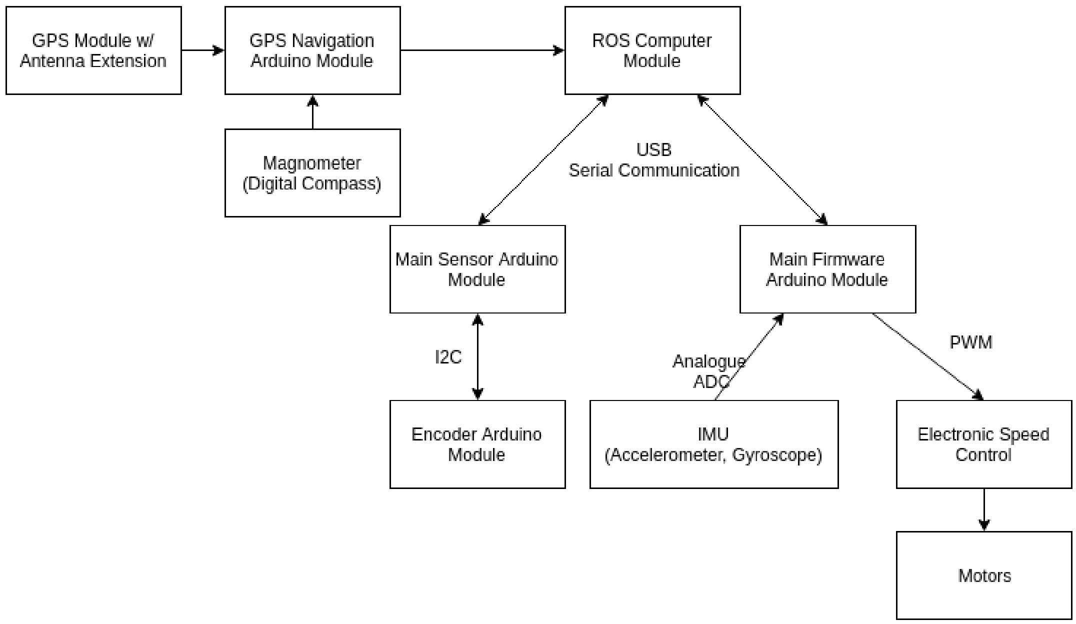

Figure 10: Snowstorm Firmware Setup

An Arduino Mega microcontroller is used for controlling and communicating with the electrical systems.

The vehicle has three main states: autonomous, remote controlled, and stopped, which can be changed through the radio controller. When it is set to autonomous mode, serial communication is used to send a twist signal from the laptop to the microcontroller. Under remote control, a wireless communication signal is sent from the radio controller to the microcontroller's receiver.

The signals are then processed by the microcontroller. The microcontroller sends Pulse Width Modulation (PWM) signals to the Electronic Speed Controls (ESCs) which control the four wheels independently, thus moving the robot in its desired direction.

Encoders are used to calculate the speed of the robot. Due to the high resolution of our encoders, the encoder output is first processed by an Arduino Uno, and then using I2C protocol, the position is sent to the microcontroller. In addition, the microcontroller also receives information from the compass, and remote control.

The compass and encoder count are combined to calculate the velocity of the vehicle, which is then sent back to the computer to be processed as feedback.

An accelerometer and gyroscope module was also included in our system to provide us with the capability of error correction in navigation when needed.

LED ALERT SYSTEM

Green LED strips are mounted on the front, back, and sides of the tower to ensure that the safety light can be easily viewed from all directions. A separate circuit utilizing discrete logic drives the LED's operation modes.

SOFTWARE STRATEGY

The software has been developed in C++, using Robotic Operating System (ROS), Open Computer Vision (OpenCV) and Open Simultaneous Localization and Mapping Libraries (OpenSLAM) libraries.

The software system receives three fundamental inputs: GPS information from the GPS, LIDAR scans from the LIDAR, and video streams from the cameras. The data from these systems is used to construct a map of the environment, over which pathfinding algorithms can then be run. Once the path has been computed, a velocity vector is computed for the robot.

VISION SYSTEM

The vision system of the robot uses three camera inputs which from which the shape of the path painted on the field is constructed. This shape is then transformed into a bird's eye view of the field for use in the path-finding and mapping algorithms.

CAMERA

A Logitech C615 HD Webcam is being used to collect data for the vision system. This camera were chosen due to its economical price and 74° field of view.

FILTERING

The goal of the filtering process is to extract the white lines painted on the grass from the surrounding environment. This is achieved by first applying a Gaussian blur to the image to remove high frequency noise. The image is then transformed into HSV colour space and separated into Hue, Saturation and Value channels. A threshold is then applied to the Hue channel to extract the white from the background.

To mask the white appearing on obstacles such as cones another filter is applied in parallel to the image. The orange from the cones is filtered and a mask is applied around the region of interest.

The two binary images are then combined using a logical AND to remove the interference from white lines appearing on obstacles.

Figure 11: Filtered Image

BIRD'S EYE VIEW PROJECTION

For the bird's eye view transformation, the input video is converted into a bird's eye view so that AI can map the vision information. The program takes the four corners of the plane of the ground in the input video and transforms them to the corners of the output video.

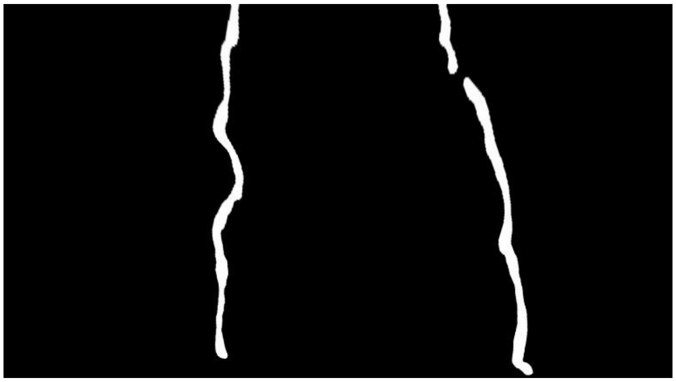
Figure 12: Bird's Eye View

LIDAR:

The SICK LMS 291-S14 LIDAR uses a rotating laser beam to measure distances to obstacles by analyzing the time of flight of the reflected beam. The LIDAR is mounted in the front of the robot and has a scanning angle of 180 degrees and a range of 30m.

The ROS node for LIDAR runs concurrently with the sicktoolbox_wrapper, which translates raw data from the LIDAR into useful data that can be read by the LIDAR node. The LIDAR node then takes the translated data and publishes the data to the Mapping node where the LIDAR data can be combined and processed along with other data.

For this year's object detection, the same LIDAR model, SICK LMS 291-S14 was used again. This decision was made based off of past performance and budget consideration. Though the LIDAR module did not change, the application of collected data has been adjusted to suite our needs better via improved mapping algorithms.

GPS

Commercially available GPS modules were obtained for this year's competition with 2 qualities in consideration: Differential GPS capabilities and antenna extension capabilities.

From previous experience, single point precision with on board antennas provided us with a 10m accuracy based off of longitudinal and latitudinal fluctuation from collected data. Hypothesis of fluctuation was based off of further research into antenna propagation and comparison of other on-hand GPS modules.

To provide accurate planning of local waypoints within this year's mapping system, a custom IMU was incorporated in order to provide compensation data for an Extended Kalman Filter.

NETWORKING

The networking module's purpose is to ensure JAUS compliance of the system, and to map from required JAUS commands to internal commands using the ROS framework.

The system is implemented in the Python programming language using the asyncio framework in order to meet two implementation goals: ease of troubleshooting and debugging, and to provide as simple and easy to understand implementation as possible.

In order to achieve these goals, the pytest unit testing framework was used to test both the functionality of individual modules and the overall compliance of the system using simulated network events.

Elements of the JAUS standard that have been implemented so far include:

- JUDP transport: the low-level transport specification

- Liveness: a basic heartbeat pulse
- Events: request scheduled reports from other services
- Management: set emergency shutoff, query platform state
- Access Control: request exclusive control over the platform
- Discovery: query the name and components of the platform
- List Manager: manage a linked list structure in the platform’s memory

DESCRIPTION OF MAPPING TECHNIQUE

The main purpose of our mapping system is to generate a local environment map that is used to update an internal global environment map. The local map generation will be implemented in our custom ROS mapping node. The mapping node will then pass the processed local map to the *slam_gmapping* node, which will superimpose the local map onto the global map, and permit pathfinding algorithms to run over it. This entire process is looped through until the ROS system is aborted.

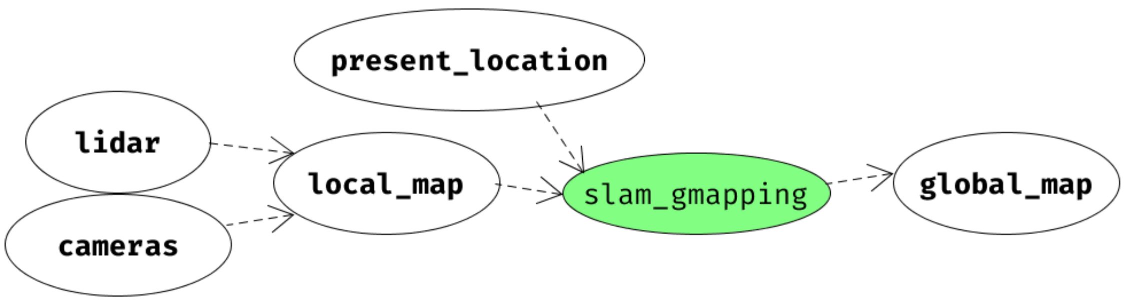

Figure 13: An Overview of the Mapping System: The Green Nodes Represent ROS Nodes.

GMAPPING:

Our mapping system uses the *slam_gmapping* ROS node, which utilizes the *SLAM (Simultaneous Localization and Mapping)* method. This node primarily relies on data from the LIDAR instead of odometry, as the LIDAR’s fast update rate can provide more accurate and precise results. The node publishes the gathered data in the form of an *occupancy grid*, along with useful localization data such as position and orientation. In addition, the *slam_gmapping* node provides us with the ability to customize the map resolution and update thresholds, making it a versatile system.

LOCAL MAP GENERATION

The main local map generation is implemented in several custom ROS mapping nodes that process and combine local environment data from the cameras and LIDAR. The scans and images from the Lidar and cameras respectively, are converted into PointClouds, and are then merged into a local map. The portions of the local map generated by the camera and LIDAR are updated as new scans or images become available.

LOCAL TO GLOBAL MAP

The path-finding algorithm requires a global map, regardless of how incomplete the map may be. The algorithm also needs to know which area of the global map is being updated to avoid recalculating the path over the entire global map. During every ROS loop iteration, the local map received from the mapping node is mapped onto a global map that is initialized in the *slam_gmapping* node. This operation is performed by transforming the local map matrix into the orientation and position on the global map with the assistance from the GPS compass as well as relative position changes based on mapping data from the *laser_scan_matcher* node. Upon consolidating this information, the map is then ready for the path-finding algorithm to update its current path.

SYSTEMS INTEGRATION

PATH FINDING ALGORITHM

Popular algorithms, such as Dijkstra and D*Lite, are capable of determining the shortest path in an obstacle filled environment; however, these algorithms are only suitable for a fully mapped environment where nothing will change during the path calculation. Our situation required an algorithm that can determine an initial path based on the limited data provided by the LIDAR and vision system, and update its resultant path as the robot traverses the environment and discovers more obstacles. For this reason we have chosen to implement the D*Lite algorithm.

COMMAND FOR ROBOT MOVEMENT

Once D*Lite finds the current optimal path to the goal waypoint, a new linear and angular velocity will be sent to *Snowstorm* to direct it. To determine *Snowstorm*'s new linear and angular velocity, a local path (essentially the first portion of the optimal path) will be estimated and smoothened out to mimic a more realistic vehicle movement.

ATTENTION GIVEN TO SAFETY, RELIABILITY, DURABILITY AND FAILURE MODES

Failure Points Identification and Resolution: The following potential failure points of our robot were identified and a corresponding solution was found to minimize the possibility of failure or mitigate effects.

System was designed to be normally open to account for any failure in the stop system.

OVERCURRENT TO POWER TRANSMISSION CABLES

The power supplied to the brushed DC motors is monitored by voltage and current meters and is controlled by PWM from the ESC's. In case overcurrent arises due to a programming error going undetected, fuses are placed at the bottleneck wire gauge at its rated ampacity.

OPERATE IN RAINY CONDITION

Since the housing of our robot is made with sheet metal held together with bolts, water may seep into the gap between the sheet metal and bolts. In order to prevent this, silicon will be inserted to cover the gap after assembling the robot for the competition. This will prevent possible damage to the electrical equipment. Furthermore, the steel shafts were replaced with 304 stainless steel ones, to prevent rusting.

STALL MOTORS

Although preventing a stalled motor beforehand is difficult, we can reduce the severity of this occurrence by minimizing the amount of time for which the motor is stalled. This is achieved by constantly polling the rotary encoder when power is supplied to the motors.

BATTERY MONITORING

The voltage of the LiPo batteries is monitored by the firmware using an inline LiPo battery monitor. This monitor automatically stops the robot if the voltage in any of the battery packs goes below a threshold voltage of 3.3V.

CHALLENGING OBSTACLES

In the case of a challenging obstacle (dead end or island) *Snowstorm* is programmed to slowly back up and try again. Upon a third failed attempt the robot will slowly start turning to find a different path.

INNOVATIVE CONCEPTS

We encountered major challenges with the merging of video streams and LIDAR data. This was eventually accomplished by conversion of both data streams to an intermediate format. Pointclouds were chosen, as they are computationally efficient to merge, and are also usable for localization and mapping, reducing the need for further data conversion.

For JAUS, the development of the networking module led to the development of a generic binary data parsing library which aims to make possible declarative parsing of arbitrary data structures, especially those like JAUS messages which mix bit fields with variable-length arrays. We hope that this pushes us further towards our ease of use and understanding criterion, since it removes the usual requirement for ad-hoc parsing code and instead allows other program logic to simply see objects with named attributes.

COST ANALYSIS

The estimated overall cost of *Snowstorm* was as follows.

Table 3: Snowstorm Cost Breakdown

Cost Analysis	
Item	**Cost (CAD)**
LIDAR Sensor	$ 3,000.00
Housing (machining and assembly)	$ 1,200.00
Laptop	$ 1,149.00
Drive train	$ 1,000.00
Lithium ion batteries +charger	$ 700.00
Snowplow	$ 280.00
Microcontroller	$ 250.00
Misc. items: wiring, velcro etc	$ 200.00
GPS, encoders, sensors	$ 100.00
Remote and Receiver	$ 55.00
Total:	$ 7,879.00

However, this does not correspond to the costs incurred by the team this year. Snowbots received a discount on the battery and GPS from sponsors earlier, and already had the LIDAR sensor.

PERFORMANCE TILL DATE

So far, *Snowstorm* has been a very promising in terms of firmware and mechanical components. Remote control testing of the robot on long grass and slopes have shown it to be robust and fast. Individual software components such as LIDAR and vision have also worked well by themselves on the robot. We also successfully merged LIDAR and vision inputs. We have had success with JAUS- all the core events, liveness and transport tests have passed.

Right now, we are in the process of integrating all sensor input to run the robot fully autonomously.

CONCLUSION

Over 40 members of UBC Snowbots worked on the design and construction of *Snowstorm* this year. The entire robot has been re-built on the software and the mechanical levels.

The mechanical design greatly emphasizes adaptability, accessibility and innovation through its modular design and unique features. The electrical design emphasizes both reliability and safety. It uses feedback systems to monitor voltages, currents, and position to minimize the damage that could be caused by over-discharged batteries and stalled motors. Safety mechanisms such as the E-stop and fuses are also in place in case of a failure. Both the mechanical and electrical systems have been extensively tested and are performing very well. This year's software strategy is sophisticated and customizable, allowing the robot to utilize a number of different strategies to complete the Auto-Nav challenge. The team is looking forward to bringing *Snowstorm* to this year's IGVC and excited to see the results of the competition.