Georgia Institute of Technology
RoboJackets

Jessiii

Project Manager
Tan Gemicioglu | tgemici@gatech.edu
Software Subteam Lead
Matthew Hannay | mhannay3@gatech.edu
Mechanical Subteam Lead
Charles Li | cli651@gatech.edu
Electrical Subteam Lead
Nathanael Koh | nkoh8@gatech.edu

Members

Mechanical

Akhil Sadhu	asadhu7@gatech.edu
Elizabeth Goetz	egoetz6@gatech.edu
Tomas Osses	tomas_osses@gatech.edu
Cameron Loyd	cloyd6@gatech.edu
Daniel Kilgore	dkilgore8@gatech.edu
Alexander Shih	alexander.shih@gatech.edu

Electrical

Tyler Bierfreund	tbierfreund3@gatech.edu
Auveed Rokhsaz	auveedrokhsaz@gatech.edu
Eugene Min	emin7@gatech.edu
Christopher Semali	csemali6@gatech.edu
Andrew Roach	aroach34@gatech.edu
Indraja Chatterjee	ichatterjee9@gatech.edu

Software

Vivek Mhatre	vmhatre3@gatech.edu
Andrew Yarovoi	ayarovoi3@gatech.edu
Yosuke Yajima	yyajima@gatech.edu
Navaneet Kadaba	nkadaba3@gatech.edu
Avery Greer	agreer31@gatech.edu
Axel Diaz	adiaz86@gatech.edu
Pranathi Singareddy	psingareddy6@gatech.edu
Vikram Tholakapalli	vtholakapalli3@gatech.edu
Nicholas Leone	nleone6@gatech.edu

Faculty Advisor
Frank L. Hammond III | frank.hammond@me.gatech.edu

May 22nd, 2021

1 Who We Are

1.1 Introduction

RoboJackets is an organization in Georgia Tech that acts as an umbrella organization for several robotics teams, including ours, RoboNav. Our team is focused on the development of autonomously navigating robots, which led us to choose IGVC as a competition. Though our previous robot, Jessii, performed decently at the previous IGVC competition, there were numerous issues with it that have led us to iterate upon it for this year's competition in the hopes that these issues can be resolved. These issues include it progressing across the competition course backwards and unreliable wiring. Improvements include a weather-proofed chassis, rewired electrical system, and new path planning algorithms. With these design changes, we hope to exhaust the Jessi design space and enter IGVC with a brand new robot in future competitions.

1.2 Organization

Our team consists of three subteams: mechanical, electrical, and software. Each subteam consists of numerous members, one of which is the subteam lead, who is responsible for developing and enforcing a timeline for robot work, as well as keeping a positive work atmosphere and ensuring member retention. The subteam leads are managed by the project manager, who oversees the development of the robot as a whole.

Position	Name	Standing	Major
Project Manager	Tan Gemicioglu	Sophomore	Computer Science
Mechanical Lead	Charles Li	Junior	Aerospace Engineering
Mechanical	Cameron Loyd	Graduate	Mechanical Engineering
	Tomas Osses	Senior	Mechanical Engineering
	Daniel Kilgore	Senior	Aerospace Engineering
	Elizabeth Goetz	Senior	Mechanical Engineering
	Alexander Shih	Junior	Mechanical Engineering
	Akhil Sadhu	Junior	Mechanical Engineering
Electrical Lead	Nathanael Koh	Sophomore	Computer Engineering
Electrical	Tyler Bierfreund	Sophomore	Physics
	Auveed Rokhsaz	Sophomore	Computer Engineering
	Eugene Min	Sophomore	Computer Engineering
	Christopher Semali	Freshman	Electrical Engineering
	Indraja Chatterjee	Freshman	Computer Engineering
	Andrew Roach	Freshman	Math & Comp. Science
Software Lead	Matthew Hannay	Sophomore	Computer Engineering
Software	Yosuke Yajima	Graduate	Robotics
	Andrew Yarovoi	Senior	Mechanical Engineering
	Pranathi Singareddy	Junior	Computer Science
	Vivek Mhatre	Junior	Computer Science
	Nicholas Leone	Sophomore	Computer Engineering
	Vikram Tholakapalli	Freshman	Computer Science
	Navaneet Kadaba	Freshman	Computer Science
	Avery Greer	Freshman	Computer Science
	Axel Diaz	Freshman	Computer Science

1.3 Design Process

We started the design process for Jessiii by reflecting on Jessii's strengths and failure points from the previous competition. The mechanical platform of our robot was mostly stable, but we needed to fix slipping problems. The electrical system and software design had numerous issues, like wire management problems and delayed path planning responses. Each subteam took steps to address these issues and we managed to fix a lot of the issues that decreased our performance before. While our build cycle was longer than usual due to the cancelled 2020 IGVC, most efforts since then have focused on preparing our newer platform and testing was difficult due to COVID-19.

2 Innovations

2.1 Multi-class Semantic Segmentation

Jessiii uses a state-of-the-art U-Net Convolutional Neural Network for Image Segmentation, an architecture we used in 2019 but now improved to perform multi-class semantic segmentation. A U-Net with a pre-trained EfficientNet encoder performs multi-class semantic segmentation, classifying the pixels in each image as a line, barrel, or neither. The U-Net architecture consists of a typical contracting network supplemented by an upsampling network. The upsampling portion of the network has a large number of feature channels from the contracting portion, which helps the network propagate context information to higher resolution layers. Using transfer learning, we can improve the network's performance by fine-tuning the weights of a pre-trained EfficientNet to classify lines and barrels in images. In addition to improved performance, transfer learning allows for faster training time and less training data.

2.2 Navigation Refactor

The navigation stack of Jessiii was revamped entirely to use the move_base_flex navigation framework. The decision to use move_base_flex allows us to utilize teb_local_planner and fix many issues with our previous navigation framework, move_base. teb_local_planner implements an online optimal local trajectory planner to avoid obstacles using timed elastic bands. In addition, teb_local_planner supports path planning for holonomic robots, providing flexibility if Jessiii moves from differential drive to holonomic drive. [whatever section path planning is in] provides more details on teb_local_planner.

Using the move_base_flex navigation frame allows for a custom navigation state machine and a more complex representation of the costmap, features that were previously not available using the move_base framework. A more complex costmap allows Jessiii to map the surrounding environment in three dimensions which enables handling of various obstacles and terrain. [3D lidar section] provides further detail of the new mapping scheme, and [whatever section path planning is in] describes the new custom navigation state machine implemented in the navigation stack.

2.3 Pointcloud Segmentation

Jessiii uses a new and faster approach in segmenting the set of points which the 3D LiDAR returns [1]. Using this new approach, Jessiii can estimate and extract the ground plane and obstacles much faster than our previous approach, RANSAC (Random Sample Consensus) [2]. The points left after ground plane extraction represent obstacles such as barrels. These remaining points are clustered using euclidean cluster extraction to determine the locations and number of obstacles that surround Jessiii. In addition to mapping detected obstacles, we also exploit the LiDAR's sensor model and consider angles where obstacles are not detected to infer unoccupied areas.

3 Mechanical Design

3.1 Overview

We kept the same approach when designing Jessiii as our other robots, first identifying weaknesses in previous designs, proposing a solution, modeling it in CAD, and manufacturing it.

Mechanically, the robot is composed of three components: the chassis, the cover, and the sensor tower. The chassis provides structure and support, as well as acts as a mounting point for a tray of electrical components (the electrical tray) and the drivetrain. The cover provides weatherproofing for the electrical tray, and the sensor tower acts as a mounting point for various sensors, which are connected to the electrical tray.

3.2 Structure

3.2.1 Chassis

The chassis experiences the bulk of the robot's load. This, combined with the fact that warping reduces sensor accuracy, led us to prioritize structural integrity in designing it. We performed finite element analysis (FEA) to determine warping and to verify the safety factor, 2, that we chose for it. The chassis contains the drivetrain, the batteries, and the payload, all mounted close to the ground to keep the center of gravity low.

To drive the wheels, we purchased two ElectroCraft MP-36 MobilePower Geared Motors. They have a max torque of 108.5 Nm - far greater than needed to scale 15° inclines - and a max output rotational velocity of around 115 rpm. This coupled with the 14 inch wheels lets Jessiii reach a maximum of 4.79 miles per hour - almost the maximum allowed speed for the competition. This drive system with the single caster wheel allows a fairly mobile robot with the capability of rotating 360° in place. The wheels are two feet apart from each other and three feet away from the caster wheel in order to be as small as possible without going below the minimum size constraints for the competition. This small size improves Jessiii's overall maneuverability.

Because of its modularity, the chassis was largely constructed with 8020 Inc's cross section of 6105 T5 aluminum. This was primarily chosen because the material is relatively cost efficient, has a high strength to weight ratio, and is very easy to work with. These extrusions are fastened together using the manufacturer-provided anchor fastener, T-nuts, and end fasteners, and in select situations where standard fasteners were insufficient, custom aluminum brackets and fasteners were manufactured.

The main issue with the material/manufacturer selection is that these fasteners tend to, induced by vibration, loosen with time. This was addressed by applying Medium strength heat release Loctite Threadlocker 243 to each fastener between the male and female ends, as this Loctite increases friction between them sufficiently so as to lock the fasteners in place.

3.3 Weatherproofing

Weatherproofing of the electrical tray was done through the cover, which in addition to weatherproofing, was designed to allow the covered electrical components to be easily accessed by members of the team. The cover itself is composed of two bent, powder-coated aluminum plates, one fixed on the baseplate and the other attached to drawer slides (Figure 1). These drawer slides are in turn attached to the baseplate, and allow the cover section to slide in and out. Furthermore, they can be detached completely if needed. Two access panels are built into the cover which allow for quick access to the electrical components when full detachment is not necessary.

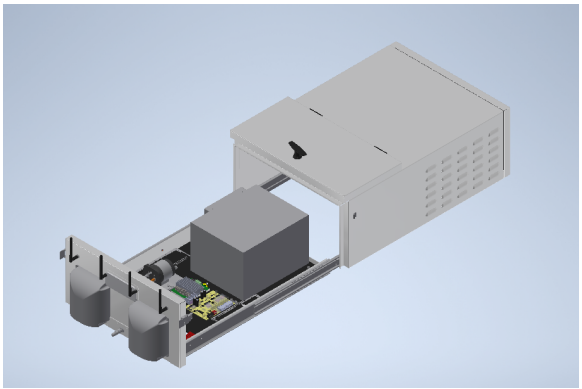

Figure 1: The electrical tray cover.

To keep the cover effective at weatherproofing the electrical components, all potentially exposed sections - such as the cover access panels and the place where the two cover sections meet - are flanged and have rubber gaskets installed within the flanges to prevent water from seeping in. The white powder coating on the cover increases the albedo and keeps the interior cool.

Proper airflow is needed to keep the onboard computer and electronics sufficiently cool. To ensure this, the cover contained holes for two Noctua fans to be inserted in and connected. The cover also contains numerous louvers. The number and size of the louvers was designed to maximise airflow; airflow is optimized when the surface area of intakes (the fans) equal the surface area of outputs (the louvers).

The cover on earlier iterations of Jessiii was manufactured by manually welding and painting sheets of aluminum. Because of the time cost of manufacturing it manually, combined with the fact that one of our sponsors, Protocase, offered a discount, we decided to outsource production to them. They sent us bent sheets of aluminum with white powder-coating and flanges for interfacing surfaces - ready for us to install onto the chassis of the robot.

Most of the external sensors used are sold as waterproof, but the cameras, the IMU, and the monitor required modification to make them so: the camera mounts surround the camera entirely except for openings for the lenses and wires, the IMU sits in a 3D-printed enclosure, and the monitor is encased in plexiglass.

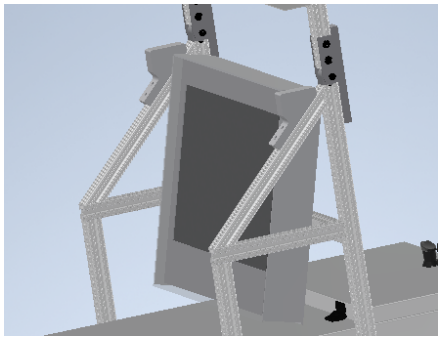

Figure 2: Sensor tower support.

3.4 Sensors

The majority of the sensors - the camera, GPS antenna, IMU, and safety lights - are mounted on the sensor tower, which like the rest of the chassis, is constructed with 8020 Inc's cross section of 6105 T5 aluminum. It is mounted to the chassis, and surrounds the cover on two sides. As seen in Figure 2, the sensor tower is held up with four beams to ensure stability.

The top sensors in Figure 3 themselves are mounted 54" from the ground, while the safety light and emergency stop button are mounted roughly 48" from the ground.

4 Electrical Design

4.1 Overview

Based on the results of the 2019 competition, the electrical design of Jessiii focused on increasing robustness and serviceability on an already functioning platform. Iterations on past designs focused on component level changes - such as changes to the E-Stop system - to increase reliability and firmware changes - such as the implementation of an RTOS to the main control firmware - to increase the performance level of the robot. A significant emphasis was placed on expanding beyond the defined IGVC electrical requirements to create a self aware platform with internal sensing capabilities. This was accomplished by creating a distributed network of custom circuit boards dedicated to obtaining, transmitting, and displaying relevant diagnostic information which connect to the rest of the system through a CAN bus. By utilizing protocols such as CAN and Ethernet, the electrical design focused on bringing automotive industry standards to a smaller scale autonomous platform. Overall, greater modularity and usability was a main focus in the iteration of previous designs and in the creation of new ones.

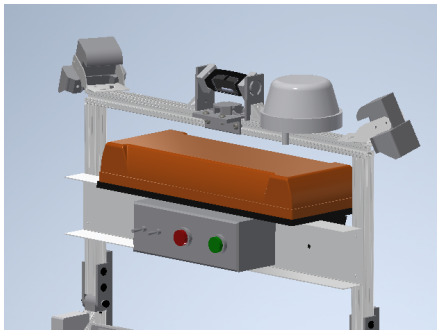

Figure 3: The LiDAR is mounted to a protruding section of the chassis.

4.2 Power Distribution

The main power of the robot is supplied using two 12V lead-acid batteries in series to create a main 24V rail. To switch the battery power, an internal disconnect rated up to 180 A is utilized. Before this disconnect, the custom hotswap circuit board (section XXX) is placed in series in order to switch the system from battery power to an external power source. The main 24V rail powers most of the larger off the shelf components such as the motors, GPS receiver, custom computer, and flood light. Using voltage regulators, the 24V rail is stepped down into 12V and 19V rails. The vast majority of components are then powered off of this 12V rail including the custom circuit boards while the monitor is powered from 19V. Other components such as the encoders are powered off of custom circuit boards which regulate power internally for smaller components.

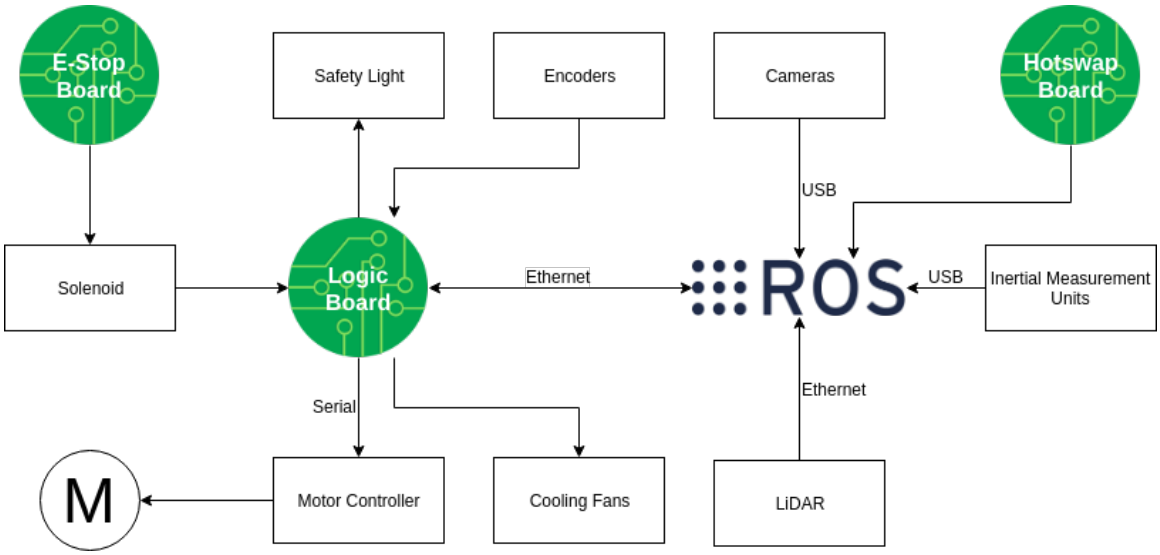

Figure 4: Overview of electrical system.

4.3 Electronics Suite

4.3.1 Hotswap Board

In order for the battery power to be reserved for actual operation, a hotswap circuit consolidated on a custom circuit board was added so that the robot could intelligently switch without interruption to an external power source. Instead of just switching power to the computer as done with the previous design, the new design is capable of switching the entire system. A LTC4416 IC controls 2 sets of MOSFETs to create a power switching circuit where the status of both power sources is monitored. When the primary source (an external power source) is present, the secondary source (battery power) is disconnected and the primary source acts as the main power input of the system.

4.3.2 Logic Board

The printed circuit board (PCB) at the center of the electrical system is the logic board and acts as a bridge between the electrical and software systems. The primary features of the logic board include maintaining the state of the robot, relaying CAN diagnostic messages, implementing a preemptive E-Stop (explained in section 4.4.2), and handling motor control logic. To maintain the state of the robot, the board reads the status of E-Stop and handles the encoder interrupts to calculate estimated wheel speeds. This information along with other status variables are relayed to the computer to be processed. Other status variables include the battery voltage, determined by the battery voltage monitor on the logic board, and the CAN messages from the diagnostic system network. This information is then later displayed on the monitor and used to inform various autonomy decisions and outputs the resulting motor commands for the logic board to execute. For added safety, a watchdog timer is implemented to automatically stop the motors if they have not seen a new command in a given period of time. This system also controls the safety light to decouple that from a potential software crash, or computer brownout.

The logic board handles the control of motors and maintains the speed using a combination of a feedforward and feedback controller

$$u = u_{feedforward} + u_{feedback} \tag{1}$$

A block diagram of the control loop is shown in Figure 5. The feedforward controller is implemented with a linear model of the motor

$$u_{feedforward} = K_v \cdot SP \tag{2}$$

where K_v is the feedforward coefficient, and SP is the set point. K_v was found by performing a linear fit on a plot of the velocity from the encoders and the motor commands. The feedback controller is implemented using a

PID controller

$$u_{feedback} = K_p e(t) + K_i \int_0^t e(t')dt' + K_d \frac{de(t)}{dt} \tag{3}$$

where K_p, K_i and K_d are the gains for the proportional, integral and derivative terms respectively, and $e(t)$ is the error term $e(t) = PV - SP$, with the process variable, which is the velocity of the motors, measured by optical encoders mounted on the wheels.

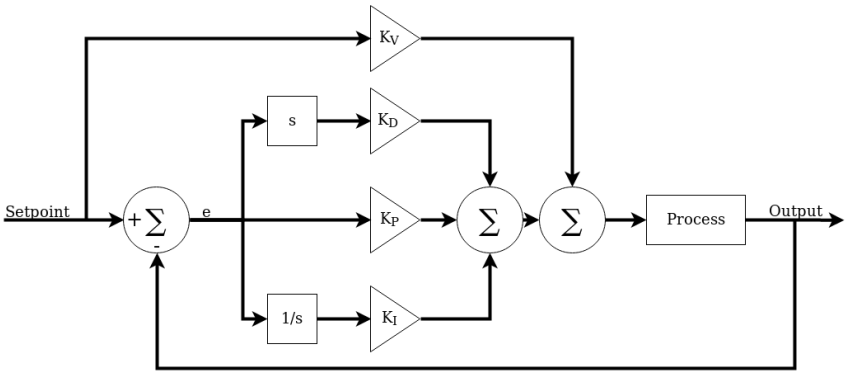

Figure 5: A block diagram of the control loop on Jessiii, with the top being the feedforward path and the bottom being the feedback PID path.

The feedforward path was added to the motor control loop in order to reduce the response time of the system, as the slow response of the system was an issue that deteriorated the ability of the robot to follow paths.

The current logic board is an iteration from the logic board on the previous robot Jessii. For this iteration, an Mbed reset circuit was added to be able to reset the Mbed from the mast when loading new firmware. In addition, extensive refactoring of the embedded firmware was done to facilitate improved debugging of the system. In addition, verification of the CAN bus, was done to support the diagnostic system. The CAN bus is a bus standard designed to allow microcontrollers and devices within a vehicle to communicate without a host computer. This feature is intended to connect all custom PCBs of the electrical system together to facilitate a more modular and robust system.

To improve real-time performance, an RTOS scheduler-based architecture was implemented in which distinct tasks were performed independently of one another. This allowed the separating the execution of low-variance tasks with tight timing requirements - such as motor control - from lower-priority diagnostics and communication tasks, reducing jitter in critical components. Mbed's OS 5 was used to provide the basic scheduling functionality.

4.4 Diagnostic System

In order to create a more intelligent platform, a diagnostic system was designed to add to the internal sensing capabilities already present on Jessiii. This system was designed to be modular in nature, by breaking functionality into several circuit boards, each with a specific function. Smaller sensing circuit boards include the fan control, current sensing, and neopixel status control. Sensor data is then processed by the main diagnostic board and relayed to the mast panel board and transmitted via RF to the remote display. With the exception of the remote display, all other circuit boards are connected via a CAN bus to create a distributed network of circuit boards to handle relevant diagnostic information. The smaller sensing boards are daisy chained together to connect to the main diagnostic board while the mast panel has its own dedicated connection. By daisy changing the smaller boards, it is possible to add and remove boards from the bus to add or change the internal sensing capabilities.

4.4.1 Main Diagnostic Board

The main diagnostic board acts as the central hub of the diagnostic system by connecting all components of the diagnostic system together as well as other existing circuit boards via a CAN bus. The CAN bus is achieved by implementing the MCP2551 CAN transceiver and the MCP2515 CAN controller. Dedicated connections for the E-Stop, logic, and mast panel boards exist, where the logic board connection can be used to transmit all CAN information to the computer. A singular connection to all of the small sensing boards is included which includes the CAN signals as well as power to all the smaller boards. The board implements the CC1101 RF module running at 915 MHz to transmit information to the remote display. Information can also be stored on a SD card.

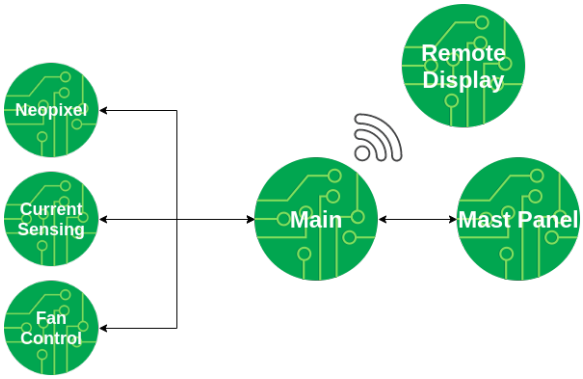

Figure 6: Diagnostic system organization.

4.4.2 Sensing Boards

The smaller sensing boards include the fan control, current sensing, and neopixel control boards. The fan control board allows for feedback control of the front fans based on the internal temperature of the electronics tray. The temperature is determined by an onboard temperature sensor as well as a temperature probe. The current sensing board connects to an Aim Dynamics' AIMH021 hall effect current sensor to determine the overall current draw of the system. The neopixel control board implements underglow underneath the robot chassis where color presets correspond to certain states such as E-Stop status.

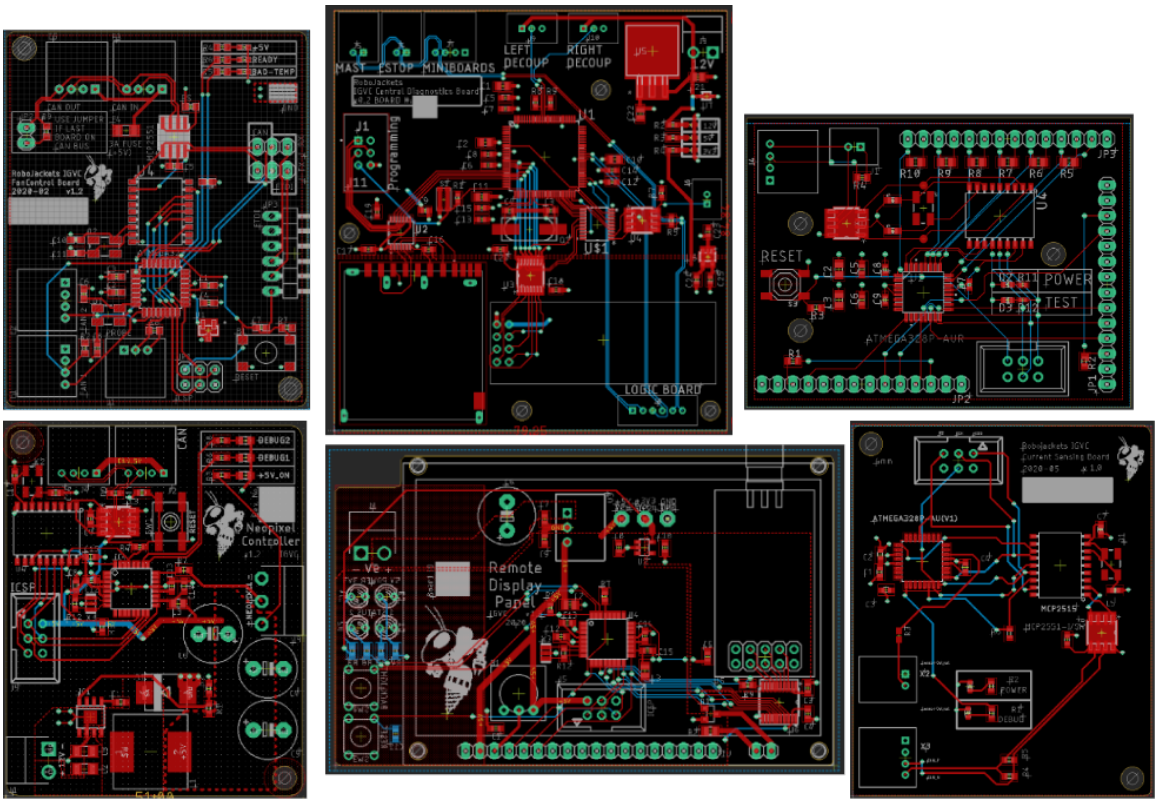

Figure 7: Top Right to Left: Fan control, main diagnostic, mast panel. Bottom Right to Left: Neopixel, remote display, current sensing.

4.4.3 Display Boards

The display boards include the mast panel board and the remote display. The mast panel board mounts next to the button box and consists of a LCD screen and debug LEDs to display various status variables. The remote display board consists of an LCD display, the same CC1101 RF module as was on the main board, and

a battery connection. While they display similar information, the mast panel board is mainly utilized when in close proximity to the robot, such as during programming, whereas the remote display is used while the robot is in operation.

4.4.4 Sensors

Jessiii utilizes several different sensors for obstacle detection and localization. The Velodyne Puck VLP-16, a 3D LiDAR, is used for primary obstacle detection. Three Logitech c920 1080p cameras on the top of the robot are used for detecting lines. For localization, Jessiii uses the Hemisphere R330 receiver with an A21 antenna for the GPS. This year, we have chosen to use two Hillcrest Labs FSM-9 IMUs: one on the mast to minimize electromagnetic interference for magnetometer readings, the other between the motors to minimize lever arm effect on accelerometer and gyroscope readings. Jessiii also contains two CUI AMT103-V optical encoders, one located on each motor. The optical disks within them each contain 48 ticks per revolution. They communicate directly with the Mbed via the logic board and provide the current speed of each motor.

4.5 Custom Computer

Jessiii's main computer has remained unchanged since the 2019 IGVC. Jessiii's computer contains an Intel Core i7-8700 CPU, a NVIDIA GeForce GTX 1060 GPU, and 32GB of RAM.

4.6 Safety Devices

4.6.1 Wireless E-Stop

The wireless E-Stop module is another custom designed PCB on the robot. For this iteration, while the functionality of the system remains with all E-Stop circuitry on the PCB, the design was changed to integrate with the greater diagnostic system, improve reliability of the wireless E-Stop, and improve user experience. The previous design relied on having two sister boards only differing in what components were populated and the loaded program, whereas this iteration separated both the transceiver and receiver design into two separate PCBs to create a more compact design for both. The receiver board is able to trigger the 24V signal used in the E-Stop, connect to the diagnostic CAN network, and connect to an RF antenna mounted on the front of the robot. The transceiver board is able to connect to the receiver through the same RF module and placed within a 3D printed case with an external push button to trigger the wireless E-Stop and external LEDs to indicate the status to the user. Similarly to the previous iteration, an ATmega328p microcontroller is responsible for generating and receiving E-Stop messages and processing CAN messages. The wireless component is handled with a new RFM95W LoRa module running at 900 MHz for greater range and reliability and interfaces with the ATmega328p through SPI.

Figure 8: Left, RF transceiver. Right, RF receiver.

In order to integrate with the mechanical E-Stop, each system does not override each other. If either physical switch is enabled or the remote is enabled, the robot will enter the E-Stop state. This ensures that in order to run the robot, both the mechanical and wireless E-stops are in agreement that it is safe to run the robot.

4.6.2 Preemptive E-Stop

Jessiii is equipped with a virtual bumper system composed of three optical distance sensors. Each optical distance sensor, a LiDAR-Lite v3, is situated under the Velodyne Puck 3D LiDAR with an angular separation of 30 degrees between each LiDAR-Lite v3. The virtual bumper system is a fail-safe redundant system that provides obstacle detection on top of the software's navigation directives. Each LIDAR-Lite v3 communicates with the Logic Board over an I2C bus. On startup, the mbed powers on each LiDAR-Lite, one at a time, and changes the default I2C address of the LiDAR-Lite to a unique address. To obtain a distance measurement, the mbed addresses each LiDAR-Lite by its unique I2C address. If one of the LiDAR-Lites detects an obstacle Jessiii cannot avoid, the virtual bumper system triggers the E-Stop.

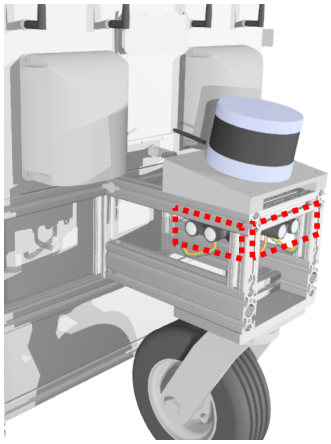

Figure 9: The location of the optical sensors for the preemptive E-Stop system.

5 Software Strategy

5.1 Overview

This year, we've continued to use the Robot Operating System (ROS) framework, a message-passing robotics middleware, for the software stack. Using this framework, the software system can be roughly divided into five groups of nodes: Sensors, Localization, Perception, Navigation and Output.

We perform multiclass segmentation on three cameras aimed at different angles to identify lanes, potholes and barrels. In addition, we construct a heightmap using a 3D LiDAR and perform a traversability analysis using this information. This is passed to the navigation module and used to construct a probabilistic traversability map. Using this map, we perform path planning and time optimal trajectory planning to generate velocities for each motor.

5.2 Obstacle Detection

We perform obstacle detection by combining redundant information from our vision-based and LiDAR-based perception. The detected obstacles are then avoided by marking them as intraversable in the traversability map, allowing our navigation module to navigate around them.

5.2.1 Vision-Based Perception

This year, we improved upon our CNN based image segmentation for lines and potholes from the last competition by changing the task to multiclass segmentation of lines, potholes and barrels. This allows us to better model occlusion of lines by other objects such as barrels when integrating this information into the traversability map.

We also improved upon the model architecture used. We use an encoder taken from a pre-trained EfficientNet (CITE) as a feature extractor as opposed to using the raw image directly, which allows us to take advantage of pre-existing datasets such as ImageNet.

We take the result of the CNN, a pixel wise per-class probability vector, and project the resulting detected lines and potholes onto the ground plane by making a flat ground plane assumption. While this assumption is not correct, probabilistic mapping and increasing the covariance on further points due to projection error helps to correct for errors introduced by this assumption.

5.2.2 3D LiDAR

We also make several improvements to our lidar-perception strategy. While our previous approach of directly using the point cloud after filtering for the ground plane worked well for a flat plane, it did not work for when the ground plane was not flat nor for the case of a ramp.

In order to address this issue, we change our approach to instead map the 3D environment around Jessiii with a heightmap and perform traversability analysis by calculating local gradients of the heightmap. In doing so, we are able to handle barrels, uneven terrain and ramps in a single unified framework. The heightmap is calculated using a per-cell kalman filter on the height, and the heightmap is smoothed in a post processing phase to remove small holes that may result in incorrect gradients.

5.3 Path Planning

When a waypoint is sent to the navigation server, the goal is sent to move_base_flex's get_path client. This returns a global path to the waypoint. If the waypoint goal has "fix_goal_orientation" set to true, then the navigation server will correct the orientation of the last segment of the path to be in-line with the previous segment. This is important when the path is sent to move_base_flex's exe_path, which generates a local path and executes it. If the orientation is not fixed, then exe_path will approach the waypoint from a strange angle instead of naturally approaching from the front. After a certain period of time, the path is replanned by sending the goal to get_path again, which afterwards sends a new goal to exe_path. This repeats until the goal is reached.

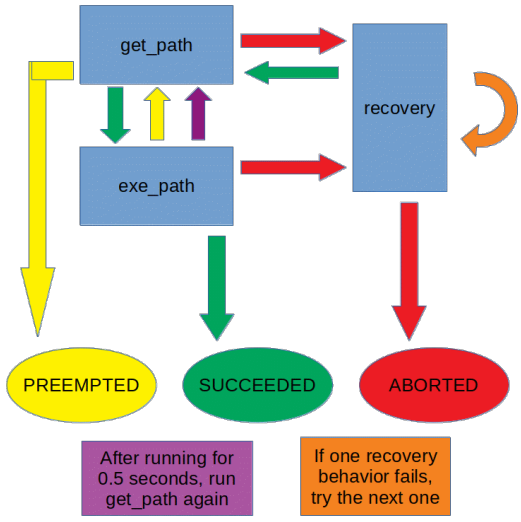

Figure 10: Diagram of path planning pipeline.

If at any point exe_path fails or if the robot is stuck in the same position for an extended period of time, recovery behavior is activated. The current recovery behavior is back_up_recovery, which makes the robot back up to a certain distance or until it is about to bump into something behind it. Afterwards, the goal is sent to get_path again.

Our local paths are generated by the teb_local_planner, which executes the exe_path section of move_base_flex. This path planning approach generates timed elastic bands with a limited distance towards a point on the global path. Timed elastic bands allow the robot to stay a safe distance away from obstacles while still moving in the correct direction. Afterwards, motor velocities are generated based on the local path and sent to the firmware.

5.4 Localization

An essential part of global map generation and navigation is knowing the position and orientation of the robot in some global reference frame (aka. localization). This is achieved by collecting an assortment of sensor information and fusing them together to calculate the most probable state of the robot. Our implementation accomplishes this by building a factor graph and performing trajectory smoothing using an open source factor graph optimization library called GTSAM [3].

Factor graphs are bipartite graphs with vertices consisting of factors and variables. The variables represent the hidden random variables such as the robot's position and orientation, while the factors represent probabilistic constraints on the variables imposed by sensor measurements or prior knowledge of the system.

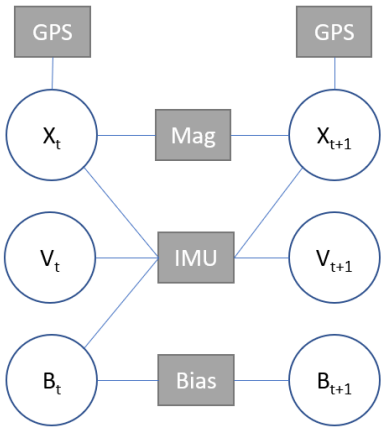

Figure 11: Graph of localization system.

Our localization implementation uses a state-of-the-art approach [4] from GTSAM to pre-integrate multiple IMU measurements into a single relative motion constraint and incorporate it as a factor in the factor graph, constraining the previous pose, velocity, and IMU bias, along with the current pose and velocity. Magnetometer readings from the top IMU and measurements from the GPS are also incorpo-

rated as unary constraints on the current robot orientation and the robot position respectively. A bias factor was also implemented to allow for online IMU bias estimation.

One of the advantages of using factor graphs and GTSAM, instead of alternatives such as an Extended Kalman Filter, is that factor graphs allow for a sparse representation of the localization problem, which is exploited by GTSAM to increase computational efficiency. This allows for more frequent state estimations and improved performance. Factor graphs are also more flexible, making it easier to add new sensor measurements and constraints to the localization problem.

5.5 Map Generation

There are four layers used for costmap generation: traversability_layer, line_layer, inflation_layer, and rolling_layer. All these layers except rolling_layer are on the global costmap, and rolling_layer is on the local costmap.

Traversability_layer uses elevation_mapping, a library that generates an elevation grid_map using the lidar sensor. The library uses kalman filters and covariances to accurately predict the height at each cell. Then, we use grid_map filters to smooth out the elevation map and then calculate the gradients and slopes at each point. If the slope is too high, the costmap layer marks that cell as lethal.

Line_layer uses neural networks and computer vision to determine where the white lines are. Lines from the segmented image output of the neural network are mapped to the cost map using pinhole projection. Areas with white lines have a high cost, while regions without lines have a low cost.

Inflation_layer comes with the costmap_2d library and "inflates" the costmap. It surrounds lethal cost cells with more, according to the size of the robot. If the robot cannot be in a cell without overlapping a lethal cell, it will be marked. Afterwards, cells within a certain radius of lethal cost cells are given a small cost, to encourage the global path planner to avoid going near obstacles when possible.

Rolling_layer subscribes to the global costmap, and iterates over cells within a square area around the robot. If the cells are lethal, then it is copied over. This way it acts like a zoomed in version of the global costmap.

6 Failure Points

6.1 Connectivity and Wiring

While we improved our wiring layout and organization on the electrical tray for Jessiii, we still face connection issues with some of our sensors. Unfortunately, COVID-19 has made it difficult to test and repair such problems and continued use of the Jessi platform has worn down some of our equipment. Our move to a new robot platform and a more organized electrical layout plan from the start of design will help address this issue.

6.2 False Positives in Neural Network

The new multi-class segmentation model produces some false positives when used on the output of the side cameras due to the cameras' unique mounting angle. These false positives may result in certain sections of the map falsely labeled as lines and thus impassable. To address false positives a pretrained Efficient Net model is used as the backbone of the model. However, additional side camera training data is needed to further train the model and address this failure point.

7 Simulations

Jessiii was simulated in Gazebo, a 3D robot simulator which was critical to the software testing of the robot. Jessiii's model is detailed in a Universal Robotics Description Format file, allowing accurate, dynamic interactions within the simulation. Different sensors on Jessiii, like the LiDAR and vision were simulated separately using their respective libraries.

This year, in addition to a qualification course for testing, we designed a full-scale AutoNav course to make sure Jessiii can navigate itself around the challenges we encounter. This course consisted of a section that reflected

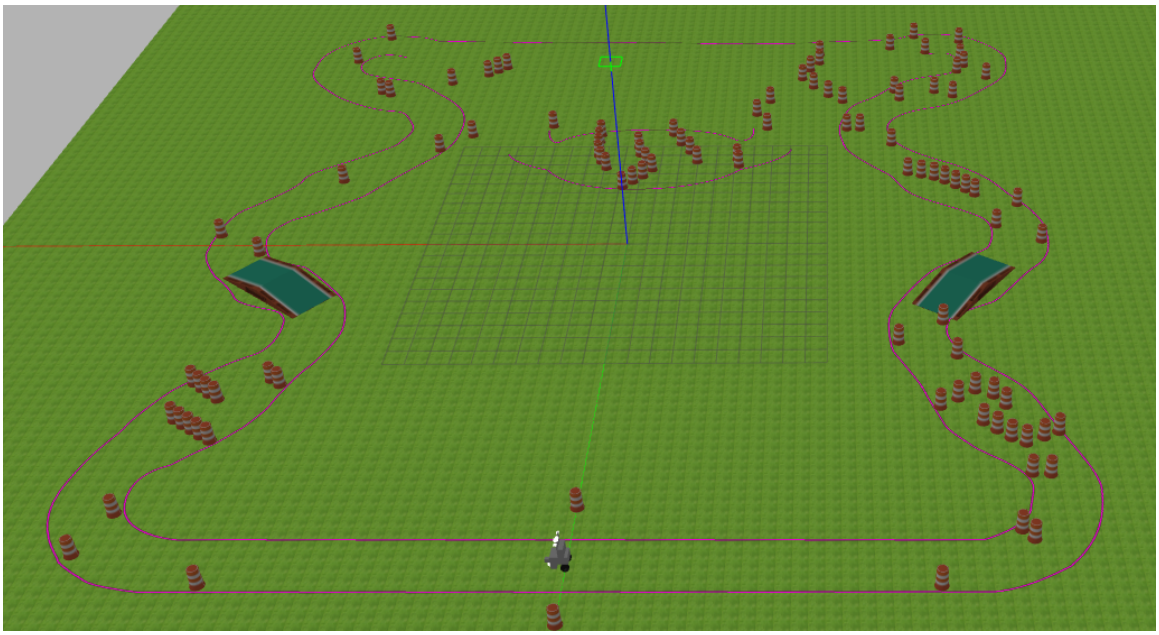

Figure 12: The simulated AutoNav course in Gazebo.

the track that was used in the previous competition, a No-Man's Land as well as a custom-designed path of edge cases that would be difficult to navigate around.

8 Performance Testing

In order to test our software stack, we created a testing library that allows us to check whether nodes are sending and receiving the correct messages. Previously, checking whether a subscriber had received the correct message was hard to do without physically interacting with our robot. To create proper tests without physical testing, we created a mock publisher and subscriber, sent messages through the publisher and subscriber, and asserted that the messages were correctly received. Using the templated class in the testing library, tests on any node are able to be created quickly and efficiently without physically interacting with our robot.

9 Initial Performance Assessments

Max Speed	4.79 mph
Acceleration	$5\,\mathrm{m\,s^{-2}}$
Ramp Climbing	25 degrees
Battery Life	6 hours standby 2 hours with motors running

10 References

[1] M. Himmelsbach, F. v. Hundelshausen and H. -. Wuensche, "Fast segmentation of 3D point clouds for ground vehicles," 2010 IEEE Intelligent Vehicles Symposium, 2010, pp. 560-565, doi: 10.1109/IVS.2010.5548059.

[2] Martin A. Fischler & Robert C. Bolles (June 1981). "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography" Comm. ACM. 24 (6): 381395. doi:10.1145/358669.358692.

[2] F. Dellaert, "Factor Graphs and GTSAM: A Hands-on Introduction," Georgia Institute of Technology, Technical Report, Sep. 2012. Accessed: May 22, 2021. [Online]. Available: https://smartech.gatech.edu/handle/1853/45226

[4] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "IMU Preintegration on Manifold for Efficient Visual-Inertial Maximum-a-Posteriori Estimation," 2015, Accessed: May 22, 2021. [Online]. Available: https://smartech.gatech.e