Navi Design Report
2012 Intelligent Ground Vehicle Competition

Rutgers, the State University of New Jersey
New Brunswick, NJ 08901, USA

June 8, 2012

Authors: Michael Koval, Peter Vasilnak, Elie Rosen **Advisor:** Predrag Spasojević

I certify that the design and engineering changes to Navi for 2012 IGVC has been significant and are equivalent to what might be awarded credit in a senior design course. These changes include a new chassis, an upgraded power distribution system, and improvements to Navi's localization and control algorithms.

Dr. Predrag Spasojević

1 Introduction

Navi is a three-wheeled autonomous robot that was designed, built, and programmed by members of Rutgers University's IEEE Student Branch. As the second iteration of the Navigator, Rutgers University's entry into the 2011 IGVC, Navi is the culmination of two years of planning, experience, and testing. It is specifically designed for navigating through unknown, outdoor environments such as those found in the Intelligent Ground Vehicle Competition (IGVC).

1.1 Design Process

As a group, the goal was to resolve the design problems that negatively affected the Navigator during the 2011 IGVC. These design changes were split into three major categories: mechanical, electrical, and software. Each category was assigned to a team consisting of a leader and several members, who are listed in Table 1. Despite these subdivisions, the entire group met for weekly design meetings and to discuss all major design decisions to avoid making any decisions in a vacuum. This insured that all team members had an up-to-date understanding of the robot's status and the priority of their individual tasks.

Before fabricating any components, the mechanical team would first share mechanical drawings and three-dimensional CAD renders with the whole group and solicit feedback about the feasibility of the design. Close collaboration between teams was especially important when deciding how to mount Navi's suite of sensors; making these decisions as a group allowed the team to avoid the electromagnetic interference, wiring, and occlusion issues that plagued Navi's predecessor.

Close interaction between the three teams also enabled continuous integration and testing of Navi's hardware, electronics, and software. While deciding how to integrate the two Manta G-125C cameras into the partially-finished chassis, for example, the mechanical team designed an enclosure, the electrical team figured out power/synchronization, and the software team analyzed the effect of camera placement on the computer vision algorithms. Integrating this type of close communication into the design process eliminated the long period of system integration that would otherwise follow every major design change.

1.2 Design Overview

During the initial stages of planning, members began by sharing what they observed while competing with the Navigator in the 2011 competition. Next, the group discussed the areas of the Navigator's design that were most in need of improvement and compiled the following list of focus areas to drive the redesign effort:

Team Member	Major	Role	Hours
Michael Koval	ECE 2012	Project Lead	500
Peter Vasinak	MAE 2012	Mechanical Lead	490
Elie Rosen	ECE 2013	Electronics Lead	400
Cody Schafer	ECE 2012	Software	20
Phillip Quiza	CS 2013	Software	15
Evgeniy Galustyants	ECE M.S.	Electronics	10
Nitish Thatte	MAE 2012	Mechanics	5
Wayne Chang	ECE 2014	Electronics	5
Siva Yedithi	ECE 2013	Software	3

Table 1: Members of Rutgers University's multidisciplinary IGVC 2012 team.

Component	Retail Value	Cost to Team
Hokuyo UTM-30LX Laser Rangefinder	$5,375	$0
Manta G-125C GigE Camera (×2)	$1,850	$0
Tamron M12VM412 Wide Angle Lens (×2)	$226	$226
MMP D22-376C-24V Brushed DC Motor (×2)	$1,000	$800
Stellaris Jaguar MDL-BDC24 (×2)	$218	$18
NovAtel ProPak V3 with Pinwheel Antenna	$11,000	$0
OmniSTAR HP L-Band Corrections (90 days)	$850	$0
PNI Fieldforce TCM Digital Compass	$1,100	$0
Computer with Intel QuadCore i7 Processor	$900	$900
Optima YellowTop Battery (×4)	$420	$130
Raw Materials for Chassis and Casing	$775	$685
Wheels, Sprockets, and Shock Absorbers	$265	$265
SolidConcepts SLS Printing	$1,300	$0
Miscellaneous Hardware	$100	$90
Miscellaneous Electronics	$650	$450
TOTAL	$26,029	$3,564

Table 2: Estimated cost of Navi itemized by component.

- maneuverability
- drive train reliability
- velocity algorithm performance
- power distribution
- camera hardware
- localization accuracy
- e-stop safety and reliability
- thermal management.

After compiling this list, each team decided on the design changes that they needed to make to improve the robot in each of the focus areas. For the mechanical team, this included shrinking the chassis, shifting the center of gravity, mounting fans for ventilation, and replacing the Navigator's four motors with a pair of higher-power replacements (Section 2). The electrical team outfitted the modified the chassis with an overhauled power distribution system, the dashboard, and a custom e-stop circuit for safer operation (Section 3).

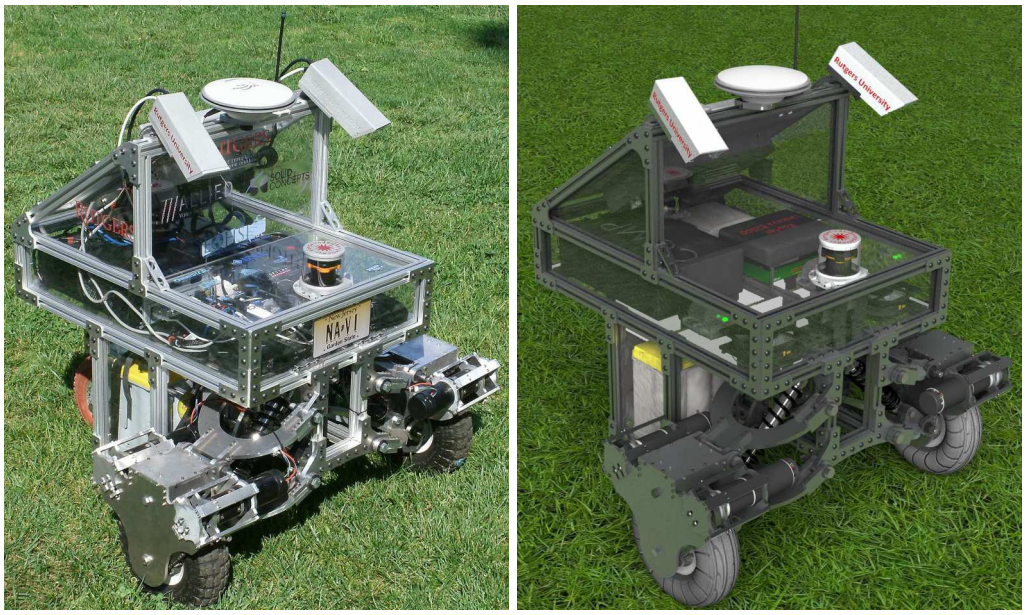
Finally, Navi received three major sensor upgrades: the Navigator's four SimpleH motor drivers were replaced with two MDL-BDC-24 motor control modules, the five commodity webcams were replaced with two AVT Manta G-125C machine vision cameras, and the unreliable, low-end inertial measurement unit was replaced with the high resolution PNI Fieldforce TCM digital compass. The software team took advantage of the new sensing capabilities by switching to a new control algorithm (Section 2.4) on the MDL-BDC-24s, improving the Kalman filter used for localization (Section 4.2), and further tuning the path planning algorithms (Section 4.4).

All these design changes, combined with the cost of the components reused from the Navigator, put the total cost of Navi at approximately $26,000. Thanks to the generous support of the team's corporate sponsors, who are individually acknowledged in Section 6, Navi only cost the team $3,500 in materials. Both of these costs exclude student labor and are itemized in Table 2.

1.3 Innovations

Navi includes several major innovations that distinguish it from its competition. These include:

- **Dashboard:** Navi is controlled from centralized power distribution center known as the dashboard. As is described in Section 3, the dashboard simplifies wiring and allows the team to power-off individual



(a) Actual Robot (b) SolidWorks CAD Render

Figure 1: Comparison of Navi to its CAD render. Consisting of more than 2500 parts, Navi's CAD model is accurate down to the individual bolts.

sensors when they are not needed. It also simplifies cable management by routing all wires through the center of the robot in a "spinal cord" structure, from which wires branch off to power individual components.

- **Lane Following:** Navi uses an advanced computer vision algorithm to identify the course boundary lines. Unlike naïve techniques that rely on thresholding raw intensity values, this algorithm is unlikely to detect false positives and works under varying illumination.
- **Serviceability:** Routine service to Navi is possible without removing any of the robot's casing. This is possible because Navi's power distribution is handled via the dashboard and the internal Ethernet network is accessible through a Wifi access point and two external Ethernet ports.
- **Suspension:** Navi's front wheels are supported on a custom double wishbone suspension (see Section 2 for details). The suspension cushions Navi's delicate sensors from shocks and keeps the chassis stable even as the robot drives over uneven terrain.

2 Mechanical Design

Navi will not succeed in competition if it is not maneuverable, i.e. if it not capable of rapidly and safely changing its speed and direction. Navi's predecessor suffered from poor maneuverability because of low torque from the drive motors and the large distance between the robot's center of gravity and axis of rotation. Even worse, the Navigator's flat chassis limited the effectiveness of the sensors by occluding the cameras' fields of view and causing severe electromagnetic interference.

Navi resolves all of these problems with three major design changes: (1) upgrading to new drive motors, (2) shortening the length of the frame, and (3) switching to a wedge-shaped design to add more sensor

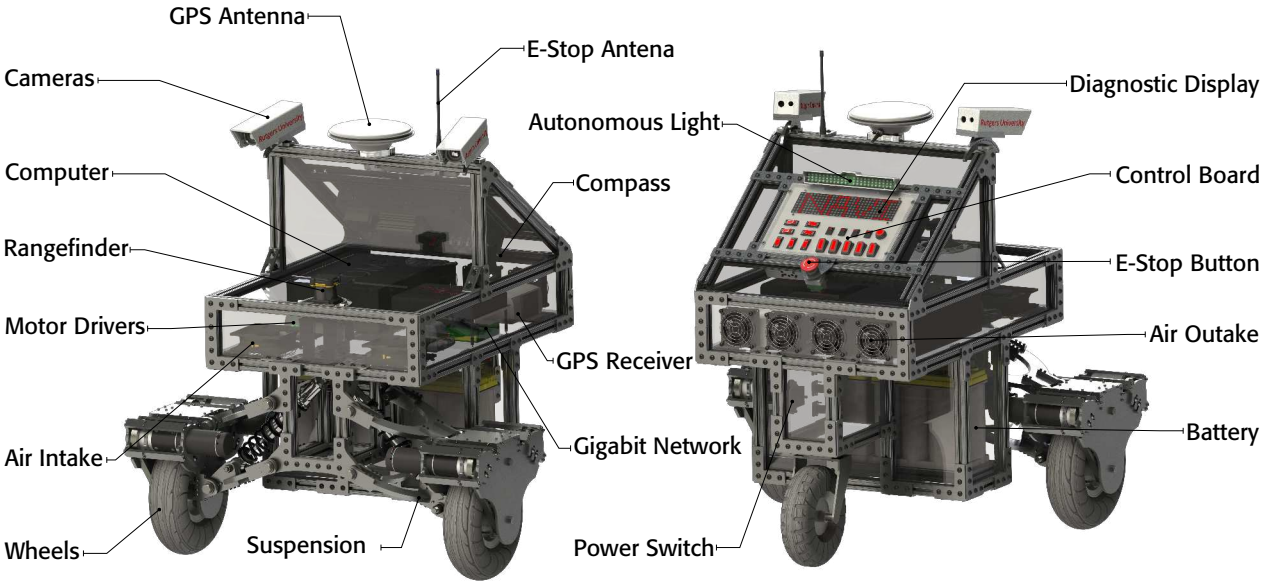


Figure 2: SolidWorks CAD render of Navi with all major components labeled.

mounting locations. These changes were possible because the mechanical team modeled the entire robot in SolidWorks, a three-dimensional CAD software package, and were able to rapidly evaluate design candidates. As Figures 1 and 2 show, Navi's CAD model consists of more than 2,500 parts and is bolt-for-bolt accurate with the physical robot.

2.1 Motor Selection

One of the most important considerations that affect Navi's speed and maneuverability are the selection of its drive motors. On the Navigator, the 10 in diameter front wheels were powered by two IG52-04-24VDC gearmotors with 12:1 planetary gearboxes and 32 CPR hall effect encoders. After the gearbox, these motors had an output speed of 285 RPM and used 1:1 gear ratio to drive the wheels. This design resulted in an underpowered drive mechanism and caused difficulty when climbing inclines and driving over uneven terrain. Even when driving on level terrain, the Navigator's low resolution encoders limited the performance of the Navigator's velocity control algorithm. Finally, driving each wheel with two motors introduced unnecessary mechanical complexity, such as idler sprockets, and made routine maintenance difficult.

While planning Navi, the team's primary goal was to transition to a single brushed DC drive motor per wheel and to improve driving performance on rough terrain. When selecting new motors for Navi, the two primary design considerations were speed and power. The IGVC rules specify a maximum speed of 10 mph; however, the number of obstacles on the course and the limits of Navi's planning algorithms make reaching the maximum speed unrealistic. The mechanical and software teams deliberated and set the target speed to 5 mph. Using Navi's new 11.5 in wheels and a gear ratio of 45:15, The 5 mph target requires each wheel to be powered by a gearmotor with an output speed of 400-500 RPM. Additionally, each motor must output approximately 80 W of power to replace the pair of 40 W gearmotors. Maintaining the same power output was a cognizant decision: more than half of Navi's power consumption is from its motors (see Section 3.1) and, as such, upgrading to higher power motor would dramatically decrease Navi's battery life. By reducing the gear reduction and upgrading to more accurate encoders, Navi improved its maneuverability without

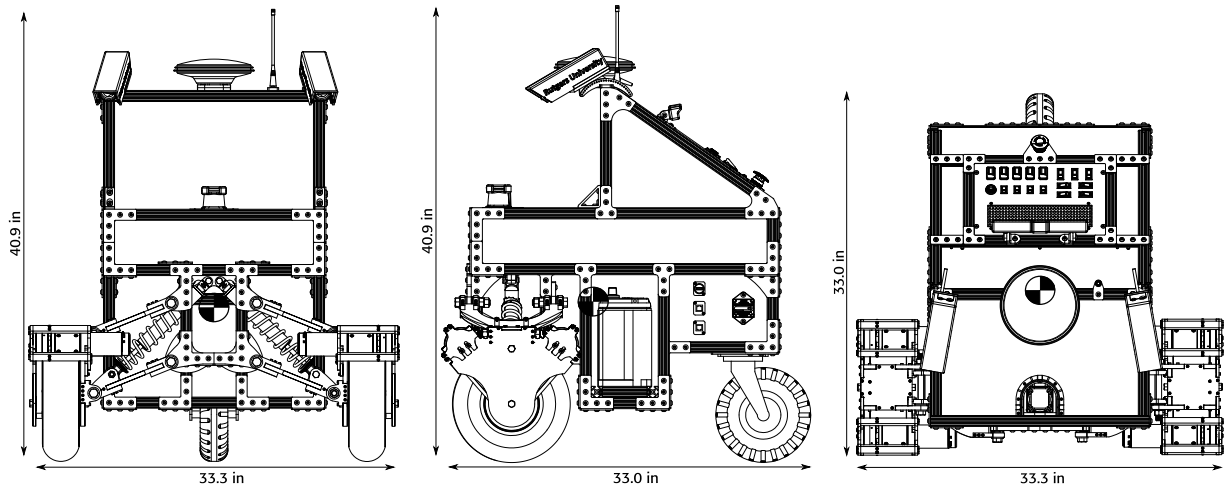


Figure 3: Mechanical drawing of Navi generated from the detailed CAD model. Note how the center of mass (denoted by a checkered circle) is located near the axis of rotation.

sacrificing energy efficiency.

From these specifications (i.e. 80 W, 400-500 RPM, high resolution quadrature encoders), the team chose Midwest Motion Product's D22-376C-12V P52-014 EU-500 gearmotors with inline 13.97:1 planetary gearboxes. The D22-376C-12V meets all of the design requirements with power output of 78 W, a top speed of 415 RPM, and an integrated 500 CPR optical encoder. With a gear ratio of 45:15 driving 11.5 in diameter wheels, Navi has a top speed of 5.67 mph and is capable of climbing slopes of exceeding a 28% grade.

2.2 Chassis Design

Second only to the drive motors, managing Navi's center of gravity was critical in achieving high maneuverability. As a three-wheeled robot with a trailing caster, Navi turns around the axis passing through the geometric center of the frame, halfway between the front wheels. Weight that is near the rear of the robot dramatically increases the robot's moment of inertia and was a serious flaw in the Navigator's design. Navi solves this problem by decreasing its length by 6 in and redistributing mass closer to the axis of rotation. As is visible in Figure 3, Navi's center of mass is just behind the front wheels and, thus, allows the robot to turn rapidly.

The shorter chassis created a space issue: Navi is not large enough for all of components to fit on a single plane. More space was added by introducing a wedge-shaped structure on top the otherwise flat chassis. The wedge houses the dashboard, cameras, GPS antenna, compass, e-stop, and wireless e-stop antenna. Besides increasing Navi's volume, the wedge distances Navi's most sensitive electronics (e.g. GPS antenna, compass) from sources electromagnetic interference, such as the drive motors, laser rangefinder, and switching voltage regulators. See Figure 2 for the locations of all of Navi's major components.

2.3 Thermal Management

Overheating can cause reliability problems and electrical faults. For this reason, Navi features a powerful air cooling system that includes two 120 mm intake fans and four 80 mm outtake fans. The intake fans are mounted on the front, bottom floor of the chassis and force cool air into the robot. This configuration pushes the air through the electronics compartment at a rate of approximately 100 cfm and forces it out the back

via the outtake fans. The airflow through the chassis was modeled in a computational fluid dynamics (CFD) simulation in SolidWorks. Figure 4 shows the result of the simulation—where bright colors indicate fast air movement—and highlights the expected front-to-back airflow.

In addition to the chassis fans, the dashboard and laser rangefinder are individually cooled. Air is forced through the dashboard by six 40 mm fans arranged in a symmetric push-pull configuration. This cools the microcontroller that is running the dot matrix display and helps dissipate heat from the large quantity of wire. The laser rangefinder is mounted on a quarter-inch, 6061 aluminum plate that acts as a mounting point for an extruded aluminum heatsink [3]. This corrected all of the overheating issues faced with Navigator. Overall, Navi's cooling system allows the robot to operate continuously in environments with high ambient temperatures: a capability that is very important because the 2012 IGVC is scheduled during the summer.

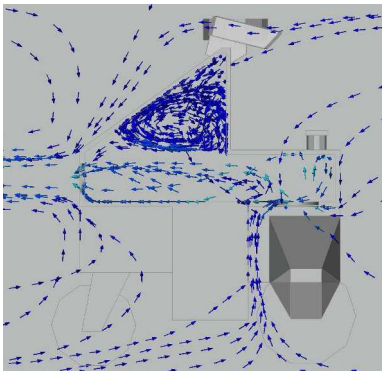


Figure 4: Airflow simulation.

2.4 Controls

Each of Navi's drive motors are controlled separate by Texas Instruments MDL-BDC-24 motor control modules. The MDL-BDC-24 is capable of sourcing 40 A of continuous current, uses indexed quadrature encoders for feedback, and manages all low-level control on an embedded ARM processor [15]. Navi takes advantage of this by off-loading each wheel's PID velocity controller to its corresponding motor control module. This eliminates the need for a custom control loop and guarantees that the controller's real-time constraints are satisfied without installing a real-time operating system on the on-board computer.

Firmware running on each motor control module's embedded processor runs a high-frequency proportional-integral-derivative (PID) controller and attempts to maintain a velocity set point using quadrature encoders for feedback [15]. While a PID velocity control loop is included in the standard MDL-BDC-24 firmware, it was still necessary for the team to tune the proportional (K_p), integral (K_i), and derivative (K_d) gains for Navi's hardware configuration. The gains were found with the Ziegler-Nichols (ZN) heuristic method, which is done by setting all of the gains to zero and slowly increasing K_p until the wheel velocity begins to oscillate. Suppose the oscillation began at $K_p = K_u$ and has a period T. Then, gains were assigned as follows: $K_p = 0.45K_u = 1.00$, $K_i = 1.2K_p/T = 0.02$, and $K_d = 0.00$. Gains obtained by the ZN method approximate a "quarter wave decay" and perform well at rejecting disturbances.

The derivative term is intentionally omitted (i.e. $K_d = 0$) because the system recovered sufficiently quickly after a load disturbance using only a PI controller. Furthermore, the standard PID controller was modified to limit Navi's instantaneous acceleration through velocity ramping. Limiting acceleration reduces wheel slippage on grass, limits stress on the wheel assembly, and minimizes the magnitude of the large current transients generated during rapid changes in velocity.

3 Electrical Design

Navi's entire electrical system was revamped for improved serviceability and safety as one of the focus areas for the 2012 competition. Following the redesign, Navi's electrical system is split into three subsystems: power distribution, communication, and safety. Navi's power distribution system (Section 3.1) is the heart

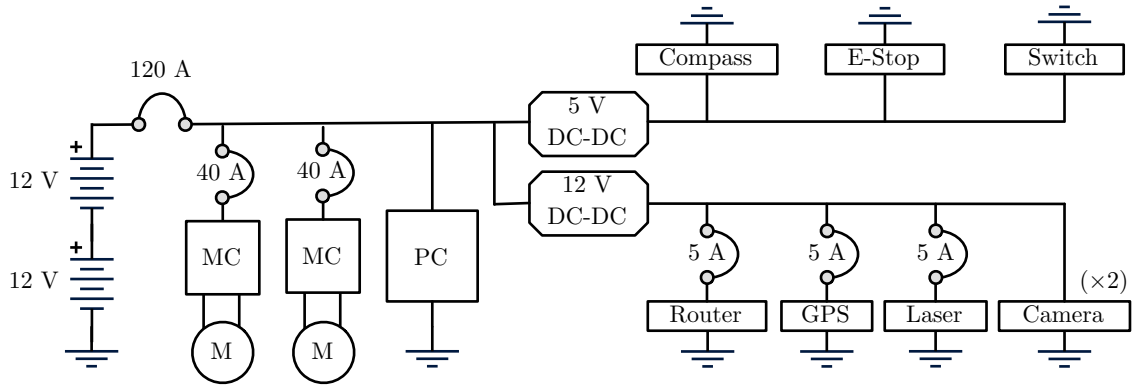


Figure 5: High level wiring diagram for Navi. Thermal circuit breakers are labeled with their current and some come components are omitted for clarity.

of the robot, and includes the dashboard, voltage regulators, and the wiring necessary to power the rest of the electronics. Once the sensors are powered, they rely on Navi's communication networks (Section 3.3) to transport data to the on-board computer. Finally, the safety systems (Section 3.4), such as the e-stop, emergency light, and dot matrix display, are vital for safe operation near humans.

3.1 Power Distribution

Navi is powered from two 12 V lead acid, Optima YellowTop batteries wired in series, which provide a total capacity of 35 A·h at 24 V. The batteries directly power the brushed DC drive motors, but are too high voltage for the other electronics. Therefore, Navi is equipped with a 360 W 24-to-12 V switching regulator, a 50 W 24-to-5 V switching regulator, and a 250 W automotive ATX power supply to power the on-board computer. All three of these regulators have 85%+ efficiency and electrically isolate the input and output grounds. Isolating the 5 V, 12 V, and 24 V grounds prevent the large current transients produced by the drive motors from affecting any of the delicate electronics attached to the lower voltage subsystems. See Figure 5 for a wiring diagram that includes all of Navi's major components.

All three voltages are distributed via distribution blocks inside the dashboard, making the dashboard the nexus of power distribution. Navi's electronics, including the voltage regulators, are plugged into the back of the dashboard using standard Molex connectors. Depending on the component, the corresponding dashboard port may be internally wired to a switch, connected in series with a thermal circuit breaker, protected by an in-line fuse, or directly attached to a distribution rail. This central source of power distribution reduces the likelihood of an inadvertent short, makes it easy to power individual components off, and simplifies the addition of new electronics.

When driving with all components powered on, Navi consumes 425 W of power and its batteries lasts for two hours. Navi's battery life is doubled to four hours if the motors are idle and can be further increased by powering-off sensors from the dashboard. Additionally, Navi is always accompanied by a second, identical battery pack that charges while the robot is running and can be swapped in as needed. Using two sets of batteries increase Navi's effective operating time to four hours under load, which is more than sufficient for a typical IGVC navigation run and long enough to preform exhaustive outdoor testing. For indoor testing, Navi is capable of being powered by a 24 V AC-DC power supply.

Component	Voltage (V)	η	Power (W)	Load (W)
MMP D22-376C-24V Motor (×2)	24	1.00	210.0	210.0
On-board Computer	24	0.95	144.0	151.6
AVT Manta G-125C (×2)	12	0.87	7.2	8.3
Hokuyo Laser Rangefinder	12	0.87	8.4	9.7
Linksys Wireless N Router	12	0.87	12.0	13.8
Novatel ProPak V3 DGPS	12	0.87	2.8	3.3
Cooling Fans	12	0.87	13.1	15.0
Autonomous Light	12	0.87	6.0	6.9
8-port TrendNET Switch	5	0.90	3.5	3.9
PNI Fieldforce TCM	5	0.90	2.5	2.8
TOTAL			409.5	425.3

Table 3: Power consumption of Navi broken down by component. All components that are powered through DC-DC regulators are penalized by the inverse of their regulator's efficiency, η.

3.2 Sensors

Navi's sensors fall into two main categories: localization and perception. For localization, Navi's primary sensor is the NovAtel ProPak V3 differential GPS with OmniSTAR HP corrections. Once the GPS has been initialized and HP corrections become available, the sensor is accurate to an RMS error of 10 cm and updates at 2 Hz [9, 10]. As a high-end GPS receiver, the NovAtel ProPack V3 handles the conversion from longitude and latitude to UTM coordinates and provides a real-time estimate of the sensor's uncertainty as northing and easting standard deviations in meters [9]. By having an estimate of uncertainty, it is straightforward to incorporate GPS updates into the Kalman filter, as described in Section 4.2.

While the NovAtel ProPak V3 is accurate, it does not provide a good estimate of the robot's heading or velocity at low speeds. Navi gets most of its heading estimates from the Fieldforce TCM tilt-corrected digital compass. After the compass is calibrated for hard iron and soft iron distortions, the Fieldforce TCM measures the robot's heading with 0.3° degree accuracy at a sample rate of 25 Hz [11]. This accuracy is achievable because of the compass' calibration routines and on-chip finite impulse response (FIR) filter mitigate the effect of noise. Like the GPS, the compass takes absolute measurements and its measurements do not drift over time.

Using its GPS and compass, Navi is able to accurately measure its position and orientation over long periods of time. Disappointingly, the location estimate provided by the GPS updates far too slowly to be used by the local planner. Navi improves the sample rate of its the location estimate using a 500 CPR optical quadrature encoder attached to each drive motor to estimate the wheel's relative motion. Accounting for the 13.97:1 planetary gearbox reduction and the 45:15 sprocket reduction, 500 CPR on the motor shaft is equivalent to a resolution of 20,955 CPR on the wheel or 0.05 mm of movement [8]. Thus, the encoders are excellent for measuring instantaneous velocity and for estimating the robot's motion over short periods of time, but are integrated measurements and accumulate unbounded error. To fuse the short term with the long term pose estimates, all three localization sensors feed into a Kalman filter (Section 4.2), giving Navi a highly accurate estimate of its position in real-time.

For perception, Navi uses one Hokuyo UTM-30LX laser rangefinder and two AVT Manta G-125C digital cameras. The laser rangefinder is located near the front of the robot and is capable of perceiving objects in a 250° field of view at a distance of up to 30 ft [3]. With an accuracy of 50 mm and a sample rate of 40 Hz [3], the Hokuyo UTM-30LX is Navi's primary sensor for detecting barrels, cones, and barricades. However, the rangefinder is not capable of detecting the course lanes and two-dimensional obstacles, such as

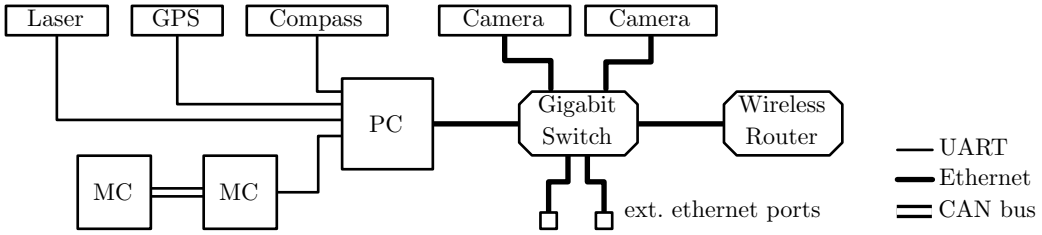


Figure 6: Navi's on-board communication networks, including UART, Ethernet, and CAN bus. Note how one of the MDL-BDC-24 motor control (MC) modules acts as a bridge between UART and CAN.

flags and potholes. Those obstacles are identified using two Manta G-125C digital cameras. The cameras are each equipped with a wide-angle, 70°-field-of-view [14] and are oriented to watch different regions of nearby ground. Together, the pair of cameras can detect obstacles on the ground at distances between 0.5 m to 12 m in a 125° field of view after being processed by the computer vision algorithms described in Section 4.3.

3.3 Communication

All of Navi's sensors must communicate with the on-board computer. At the simplest level, the PNI Fieldforce TCM digital compass, Hokuyo UTM-30LX laser rangefinder, and NovAtel ProPak V3 GPS are wired directly to the computer using USB-to-serial adapters and communicate using proprietary UART protocols [3, 9, 11]. The left MDL-BDC-24 motor controller module is also connected to the computer using a USB-to-serial adapter and communicates over UART, but acts differently: it serves as a bridge between the computer and to the on-board CAN bus [15]. Both motor control modules are individually addressable and communicate over the CAN network. Using a UART bridge reduces the effective bandwidth of CAN network to 115 kbps—substantially lower than the 1 Mbps supported by CAN—but is more than sufficient for controlling Navi's two motors [15].

Navi features an on-board gigabit Ethernet network including an 8-port TrendNET gigabit switch and a Linksys 802.11n wireless router. The wireless router allows both Navi to easily compete in the JAUS challenge and enables team members to easily control the robot while outdoors. If wireless speeds are insufficient, e.g. while transferring large amounts of data, there are two permanently-mounted Ethernet ports on the side of Navi used to obtain wired access to the internal network. Additionally, the on-board gigabit network is used to stream data from the two Manta G-125C GigE cameras to the computer for processing. This connection takes advantage of the network's support for 9 kB "jumbo frames," which dramatically reduces network latency and reduces the computer's overhead from the high network load [1].

3.4 Safety

As with any large mechanical project, an attention to safety must be considered throughout the entire design and building process. Navi's safety starts with the use of polarized Anderson connectors to ensure that a 24 V power source cannot be connected incorrectly or cause a short circuit. Next, the custom made dashboard incorporates thermal circuit breakers to ensure that excess current cannot be drawn by the motors if they stall and reduces the risk of an electrical fire in the case of a major fault. For the same reason, all wires, connectors, and switches are used within their rated parameters and include a significant safety factor.

Finally, Navi includes a custom, hardware-based emergency stop system. The e-stop insures that Navi can always be stopped by pressing the large emergency stop button located on the rear of the vehicle or by

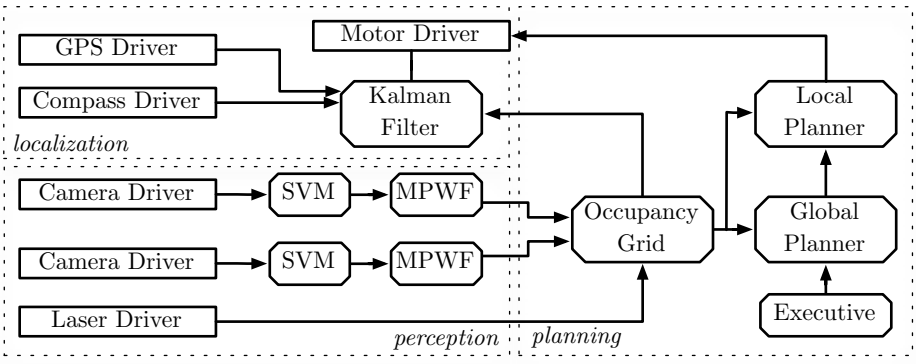


Figure 7: High level overview of Navi's software architecture. Each box represents one or more ROS nodes and each edge represents one or more topics. Nodes that directly interface with hardware are denoted as rectangles.

activating the wireless remote from distances up to 1000 ft away. Special attention was given to the e-stop circuitry to ensure that Navi becomes inoperable if the e-stop board loses power. Additionally, the e-stop is built entirely from discrete CMOS logic and does not depend on the functionality of any software.

4 Software Design

There are three major components to the Navis software architecture: localization, perception, and planning. *Localization* (Section 4.2) is the process of estimating the robot's absolute pose—position and orientation—in a fixed coordinate frame and is crucial for finding the waypoints. Next, *perception* (Sections 4.3 and 4.4) involves identifying the obstacles in Navi's path, such barrels, barricades, and lane boundaries. Together with localization, perception allows Navi to build a map of its environment. Lastly, Navi's *planning* algorithm (Section 4.4) uses a map of the robot's environment to generate an obstacle-free path between Navi's position and the location of the next goal. Navi then converts the path into motor velocities using a trajectory-based local planner.

Each of these architectural components is actually composed of many smaller units, called *nodes*, and are unified using the Robot Operating System (ROS) [12]. Nodes run in separate processes and communicate via the publish-subscribe model using a small number of standard messages over a TCP/IP protocol. This flexibility gives Navi's software the ability to scale across multiple CPU cores and multiple computers that are connected to the internal gigabit Ethernet network. The ROS architecture provided the team with a wealth of pre-written software, such as the local path planner, OpenCV, Point Cloud Library (PCL) [13], the Gazebo simulator [6], and the RViz visualization utility. In turn, the team has made several contributions to the open source community, including a driver for PNI compasses, control software for the MDL-BDC-24, and the custom Kalman filter used for outdoor localization.

4.1 Simulation

All of Navi's high level software has been evaluated in the simulated environment provided by the open source Gazebo simulator [6]. This is possible because of Gazebo's close integration with ROS: the same Unified Robot Description Format (URDF) file that defines the locations of Navi's coordinate frames also defines the

parameters necessary to simulate the robot in Gazebo. The URDF file is annotated with simulated sensors, including wheel encoders, a GPS, a compass, a laser rangefinder, and two cameras. All of the parameters to these sensors have been tuned to approximate those actually on the robot and have been modified to introduce sensor-specific noise that follows a realistic noise model (e.g. the simulated encoders suffer from wheel slip).

The Gazebo simulation takes advantage of ROS' publish-subscribe model by publishing and subscribing to to the same topics as the actual hardware drivers [12]. This makes the simulator a drop-in replacement for the physical robot and allows all of Navi's high level algorithms to run in simulation with no changes. This was especially beneficial in developing and tuning Navi's localization, perception, and planning algorithms because they could be developed while the mechanical and electrical changes were still underway. Finally, the simulator provides a full-featured test bed for evaluating how Navi's planning algorithms respond to complex situations, such as switchbacks, center islands, and dead ends, that are difficult to exhaustively test on the actual robot.

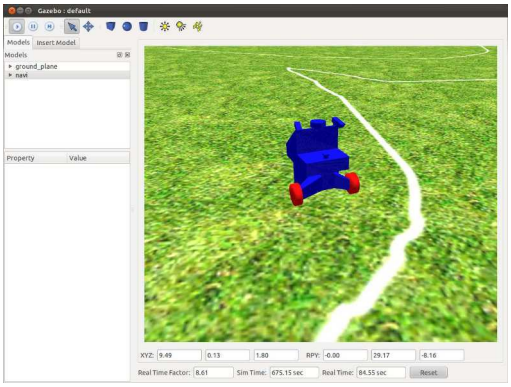


Figure 8: Gazebo Simulation

4.2 Localization

For Navi to drive between GPS waypoints, it first must *localize*, or determine its position and orientation, relative to those waypoints. This is challenging because Navi's three methods of estimating it's pose—the GPS, compass, and wheel encoders—have dramatically different properties. The GPS measures the robot's absolute position with a high degree of accuracy (approximately 10 cm RMS error), but suffers from an low sample rate of 2 Hz and provides no information about orientation. Complementary, the compass provides high resolution (20 Hz) orientation data, but gives no estimate of the robot's position. Finally, Navi's wheel encoders provide instantaneous feedback about changes in position and orientation, but must be integrated to obtain a pose estimate. This causes the odometry estimate provided by the encoders to drift over time and accumulate unbounded error. Together, these properties make the encoders vital for short term estimates, but useless for long-term localization.

Treating Navi's actual pose, $x = \langle x, y, \theta \rangle$, as the robot's state, each sensor reading of the GPS and compass can be thought of a *measurement* of the latent state corrupted by noise. Similarly, the wheel odometry can be thought of as a *prediction* of how the robot has moved since the last sensor measurement, also corrupted by noise. Assuming that these sources of noise are zero-mean, have known covariance, and are multivariate Gaussian, then the Kalman filter will optimally estimate the robot's state over time [16]. The GPS and compass covariance matrices, Σ_{gps} and Σ_{comp}, are assumed to be diagonal and are updated in real-time using the error estimates provided by the sensors. The covariance matrix for the wheel encoders, Σ_{enc}, is assumed to be diagonal, proportional to velocity, and was empirically estimated.

Since Gaussian distributions are self-conjugate, the Kalman filter is able to track the robot's pose simply by repeated application of Bayes rule. When a new GPS or compass sample is available, the *measurement step* incorporates the new information into the state estimate using the update rule,

$$x_{t+1} = x_t + K(z - Hx_t)$$
$$\Sigma_{t+1} = (I - KH)\Sigma_t,$$

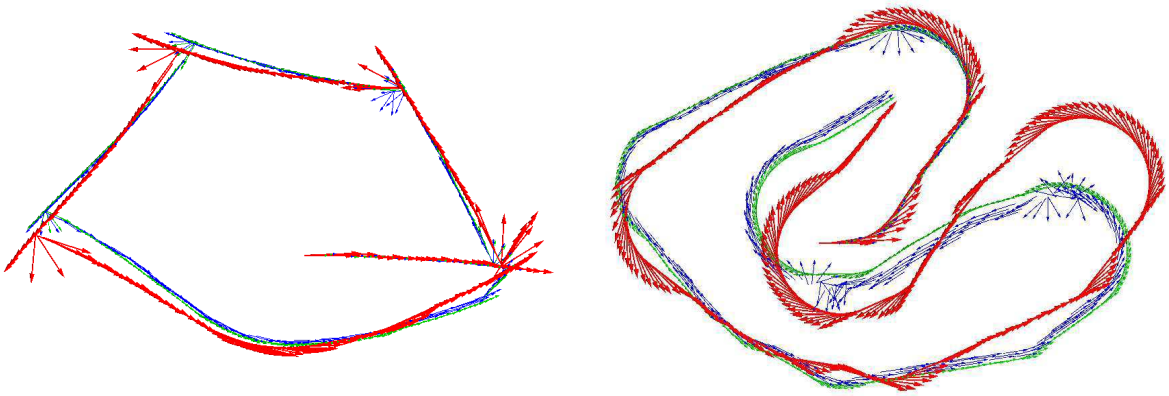


Figure 9: Comparison between position estimates provided by the wheel encoders (red), Kalman filter (blue), and ground truth (green) in two Gazebo simulations. Note how the output of Kalman filter closely approximates ground truth.

where $K_t = xH^T(H\Sigma_t H^T + \Sigma_z)^{-1}$ is the Kalman gain, H is the sensor specific measurement matrix, and z is the measurement [16]. Similarly, the *prediction step* incorporates a new odometry sample into the state estimate using the update rule,

$$x_{t+1} = x + u$$
$$\Sigma_{t+1} = \Sigma_t + \Sigma_u.$$

Note that the Kalman filter only needs to store the mean and variance of the last pose estimate. When updating the estimate, the measurement step may only decrease and the predict step may only increase the variance of the estimate. Intuitively, this is similar to using the encoders for short-term estimates and correcting the long-term behavior using the GPS and compass.

The actual performance of this Kalman filter was evaluated evaluated in a Gazebo simulation. In the simulation, the GPS was corrupted by two-dimensional Gaussian noise with 0.15 m standard deviation, the compass was corrupted by Gaussian noise with 3° degree standard deviation, and the wheel velocities measured by the encoders were subject to noise with standard deviation equal to 10% of the velocity. As Figure 9 shows, the Kalman filter eliminates the drift that is visible in the encoder data while retaining their high sample rate.

4.3 Lane Following

Before Navi can follow the course lanes, it must first identify the boundary lines in the images provided by its two cameras. When Navi receives a pair of images from the cameras, it begins processing them in parallel by classifying each pixel as 'line color' or 'not line color' to produce a pair of binary images. These binary images are processed by matched pulse width filters and are refined using non-maximal suppression to remove regions of the wrong shape. Finally, the remaining pixels are projected onto the ground plane and added to Navi's occupancy grid.

After exploring a large amount of training data, it became clear that thresholding the intensity of individual pixels was far too unreliable to use in competition. Instead, each pixel is characterized by the hue and saturation values of the 5×5 pixel window centered at the pixel of interest. The contents of this window are compressed to a two-dimensional histogram of hue and saturation values with three bins along each

(a) Original Image (b) SVM Classification (c) MPWF Response

Figure 10: Intermediate stages in the lane detection algorithm. The original image is binarized by the SVM, then false positives are rejected by the MPWF.

dimension, resulting in a nine-dimensional feature vector. These features are then classified by a radial basis function support vector machine (SVM) trained on a large corpus of manually-labeled training data. Since the training data includes images captured at many different illumination levels and the features are largely illumination-invariant, this classifier generalizes well to different lighting conditions.

Unfortunately, it is impossible to differentiate between lines and other bright regions in the image using only color information. After the source images have been binarized by the SVM, they are processed by a *matched pulse width filter* (MPWF) that removes white regions that are of incorrect width. This is made difficult because of the effects of perspective projection: while the lines are a constant width in real units, their perceived size varies inversely with distance [4, 5]. The MPWF uses the pinhole camera model to construct two digital filters tuned to the appropriate widths: a row filter to detect vertical lines and a column filter to detect horizontal lines. Each filter consists of a centered, positive *pulse* of width w flanked by two negative *supports* of width αw. Normalized to sum to zero, each filter has a zero response to uniform noise and responds most strongly to white regions of the image with the appropriate width.

Once processed by the color classifier and matched pulse-width filter, the lines have been sharpened and the false-positives have been removed. Finally, non-maximal suppression is used to find the center of each line and the resulting points are thresholded by their intensities. Any remaining points are projected onto the ground plane and are added as obstacles on the map used for navigation. At this point, following the lanes requires no special navigation code: Navi's planner will automatically avoid the lines and prefer to remain near the center of the lane.

4.4 Path Planning

Navi's *navigation executive* takes the pose estimate from the localization and the occupancy grid from our mapping system and determines the order in which the robot should visit the goal GPS waypoints. On first receiving the list of waypoints, the executive converts the latitude and longitude coordinates into $\langle x, y \rangle$ displacements from the robots start position. the executive then chooses the order of the goals that minimizes the cost of the round trip between the waypoints by assuming that the robot can travel in straight line between waypoints. At each step, the executive greedily chooses the next closest goal and informs the path planner so that it can determine a detailed motion plan.

The robot's *path planner* is an adaptation of the standard path planner provided in ROS navigation stack [7]. This path planner receives the goal from the navigation executive and creates a coarse plan using a graph-based *global path planner*. Ignoring the robot's dynamics, the global path planner models the robot as

a point that can move in any direction and inflates obstacle in the occupancy map by half the width of the robot plus a small safety factor. By treating each unoccupied cell in the occupancy grid as a node in a graph and the adjacency of cells as edges, the global path planner uses the A* search algorithm to efficiently plan a rough path through the occupancy grid. This coarse plan is then fed into a local planner, which is active throughout the robot's journey along the path.

Accepting the output of the global path planner as input, the *local path planner* is responsible for directly issuing velocity commands to Navi's drive system. The local path planner takes into account the robot's physical size and dynamics to plan locally optimal paths and quickly reacts to new sensor data. The local planner works by tracking and planning on a 10×10 m local occupancy grid around the robot rather than the global map. It then uses the *dynamic window approach* (DWA) [2] to search for safe velocity commands within that window. The DWA works by sampling all possible velocity commands, constrained by the robot's acceleration and velocity limits, and simulates each using knowledge of the robot's dynamics. Each candidate command is scored based on how closely its forward-simulated trajectory follows the global path, proximity to the centerline of the course, proximity to obstacles, and the robot's speed. Paths that cause the robot to hit an obstacle or exit the course are immediately discarded and the optimal command is sent to the motor driver. Navi runs the local path planner at 20 Hz and samples 500 trajectories for each iteration.

Combining the global and local path planners allows Navi to efficiently deal with the most difficult terrain. The global path planner produces a near-optimal path given accurate knowledge of the environment, but is too computationally expensive to quickly react to new sensor data. The local is capable of quickly reacting to new sensor data, but can easily get caught on local minima without the global path planner specifying a long-term goal. This is especially apparent in courses with U-shaped center islands: the robot thrashes about because no set of commands both moves it closer to the goal and further from obstacles. With the global path planner active, the robot plans a new path that leads out of the trap. Furthermore, this planning combination allows Navi to deal with the dashed lines without explicitly modeling the dashes. Since the local planner heavily favors trajectories that are near the course centerline, the Navi will equally avoid solid and broken lines.

5 Conclusion

Navi is an all-terrain intelligent ground vehicle that is capable of autonomously navigating through an outdoor obstacle course. The robot's upgraded drive motors and custom double wishbone suspension enable the robot to travel at speeds of up to 5.6 mph and climb slopes with a 28% grade. Under these conditions, Navi's battery pack lasts for two hours and can be quickly swapped for indefinite runtime. The fine grain control afforded by the dashboard allows operators to power individual sensors off to further extend the robot's life during extended testing. When operating autonomously, Navi's sophisticated software stack allows the robot to localize itself with 10 cm accuracy, perceive obstacles at a distance of 30 m, and identify boundary lines at a distance of 12 m. Navi's local planner continually incorporates new sensor data and is capable of responding to obstacles in less than 50 ms. The innovations described in this paper represent two years of work by members of Rutgers University's IEEE Student Branch and will make Navi successful in the 2012 Intelligent Ground Vehicle Competition.

6 Acknowledgments

Rutgers University's IGVC team would like to express our gratitude to everyone that has helped us over the past two years of competing in IGVC. First and foremost, we would like to thank Rutgers University's Engineering Governing Council, the Computational Biomedicine Imaging and Modeling Center, the Rutgers Laboratory for Real-Life Reinforcement Learning, and the departments of Electrical, Industrial, and Mechanical Engineering for the continual support.

We would have never been able to build Navi without the generous support of our sponsors: 80/20, Allied Vision Technologies, Environmental and Occupational Health Sciences Institute, ESD, Github, IEEE, the Knotts Company, Linx Technologies, Midwest Motion Products, National Instruments, NovAtel, OCZ Technology, OmniSTAR, Optima Batteries, PNI Sensors Corporation, Red Bull, Solid Concepts, Sticker You, and Texas Instruments. Finally, we would like to thank Dr. Kristin Dana, Dr. Stuart Shalat, Dr. Predrag Spasojević, Owen Healy, Joe Lippencott, John Petrowski, Steve Orbine, Joseph Vanderveer, and John Scafidi for their help.

References

[1] Allied Vision Technologies. *AVT GigE Vision Cameras Technical Manual*, Jun. 2011.

[2] D. Fox, W. Burgard, and S. Thrun. The Dynamic Window Approach to Collision Avoidance. *Robotics & Automation Magazine, IEEE*, pages 23–33, 1997.

[3] Hokuyo Automatic. *Hokuyo UTM-30LX Specification*, May 2009.

[4] A. Huang. *Lane Estimation for Autonomous Vehicles Using Vision and LIDAR*. PhD thesis, Massachusetts Institute of Technology, 2010.

[5] A. Huang, D. Moore, M. Antone, E. Olson, and S. Teller. Multi-Sensor Lane Finding in Urban Road Networks. *Proceedings of Robotics: Science and Systems, Zurich, Switzerland*, 2008.

[6] N. Koenig and A. Howard. Design and Use Paradigms for Gazebo, an Open Source Multi-Robot Simulator. In *Intelligent Robots and Systems*, pages 2149–2154, 2004.

[7] E. Marder-eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige. The Office Marathon: Robust Navigation in an Indoor Office Environment. *System*, 1998.

[8] Midwest Motion Products. *MMP D22-376C-24V Specification*.

[9] NovAtel. *NovAtel ProPak-V3 Product Sheet*, Apr. 2011.

[10] OmniSTAR. *OmniSTAR DGNSS Coverage Information Sheet*.

[11] PNI Sensors Corporation. *PNI Fieldforce TCM User Manual*, Jun. 2011.

[12] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. ROS: An Open-Source Robot Operating System. In *ICRA Workshop on Open Source Software*, 2009.

[13] R.B. Rusu and S. Cousins. 3D is Here: Point Cloud Library (PCL). In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–4, 2011.

[14] Tamron. *M12VM412 Specification*.

[15] Texas Instruments. *Stellaris Brushed DC Motor Control Reference Design Kit User Guide*, Jan. 2010.

[16] S. Thrun, W. Burgard, and D. Fox. *Probablistic Robotics*. MIT Press, Cambridge, MA, 2006.