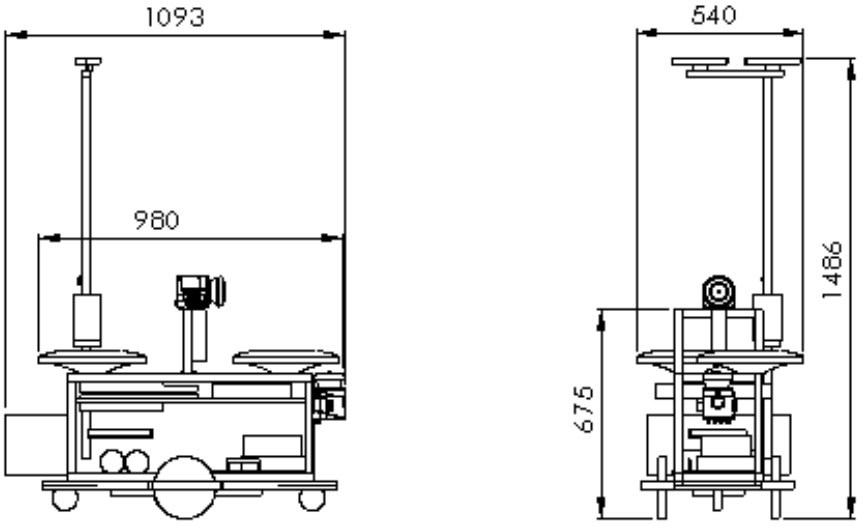
UNSW AUSTRALIA

INTELLIGENT GROUND VEHICLE COMPETITION 2015

UNSW Mechatronics Design Report



Quality Certification

I, Dr. Mark Whitty (faculty advisor), hereby certify that the design and engineering of the vehicle by the current student team has been significant and equivalent to what might be awarded credit in a senior design course.

MWhitty

Contents

1. Introduction

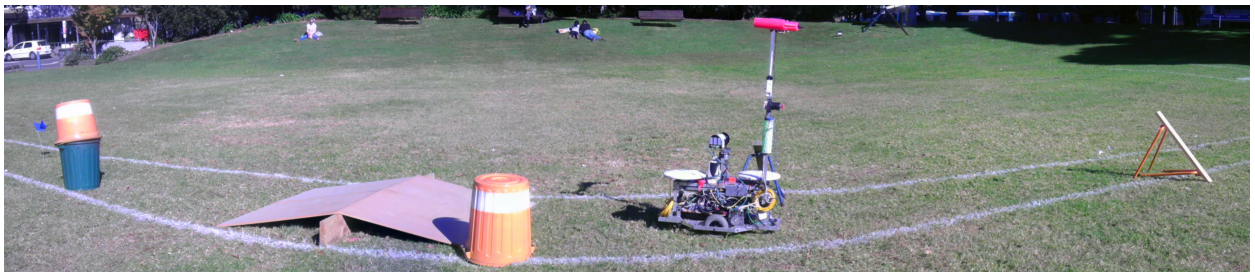


FIGURE 1.1: Prototype platform on a small course.

This report presents the UNSW Mechatronics team approach to the 2015 Intelligent Ground Vehicle Competition. Many of the members of this year's team have competed in similar competitions, and much of the development of this year's entry has been influenced by the lessons learned over the previous years. The requirement of the competition is to ensure robust and fully autonomous navigation of an unmanned ground vehicle between given GPS waypoints in the minimum time possible. The vehicle must stay within lanes and avoid randomly placed obstacles such as barrels and flags.

To ensure fulfillment of the competition requirements, the following objectives must be met and are detailed in their respective chapters.

- Provide effective team management strategies throughout the design process (Chapter 1)
- Design and procurement of mechanical and electrical components (Chapter 2)
- Design of software to provide obstacle avoidance and effective navigation (Chapter 3)
- Testing of the system to ensure robustness of design (Chapter 4)
- Provide innovative strategies to increase ease of deployment (Chapter 5)

1.1 Team Structure and Design Process

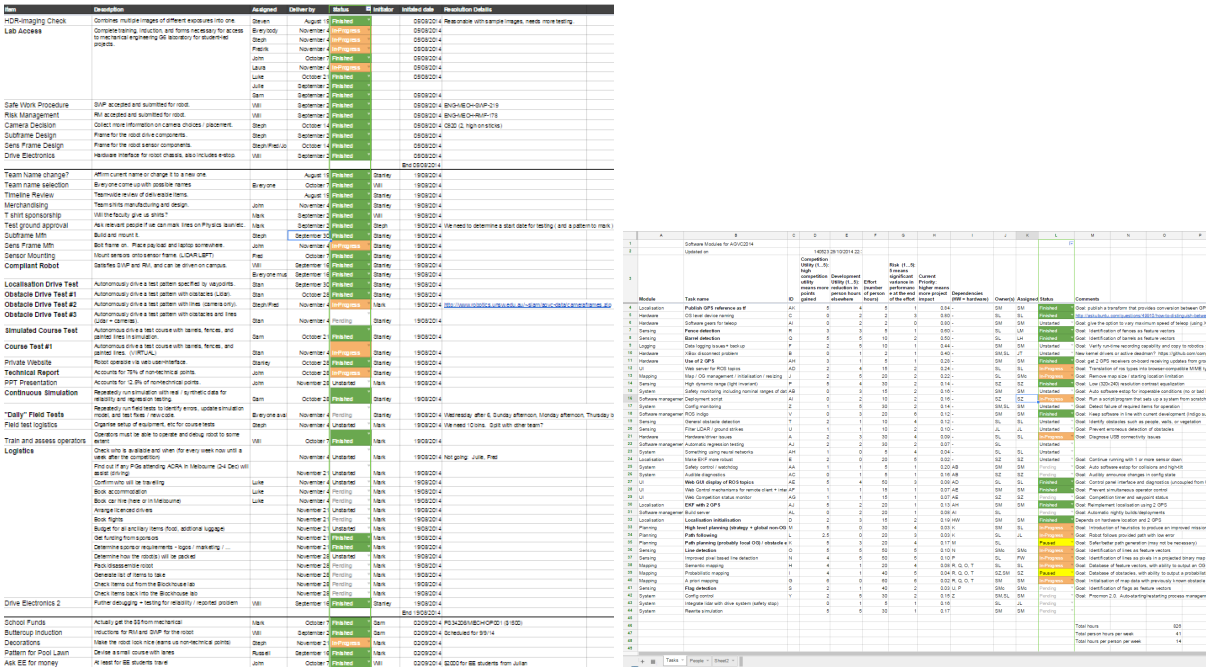
This chapter details the team's management practices, meeting structure, and design process. The team members, along with their departments, are documented in the table below. An estimated 750 person-hours was expended for this project.

Team Members	**Email**	**Department**
William Andrew	w.andrew@unsw.edu.au	Platform, Software
John Lam	john.lam@unsw.edu.au	Platform, Software
Stephanie McArthur	s.mcarthur@unsw.edu.au	Platform
Fredrik Westling	fredrik.westling@gmail.com	Platform
Samuel Marden	s.marden@unsw.edu.au	Software
Stanley Lam	s.lam@unsw.edu.au	Software

1.1.1 Time Management

The team has heavily utilized the Critical Path Method (CPM), a project management technique, when sequencing and scheduling tasks in the development of the Ground Vehicle. A worst-case time estimation approach was used for generating a work breakdown structure, ensuring that the new drive platform solution could be constructed with sufficient testing time prior to the event.

1.1.2 Task Tracking



(A) Project tasks and milestones spreadsheet. (B) Software department tasks spreadsheet.

FIGURE 1.2: Shared Google Documents for managing internal deadlines and deliverables.

To manage the status and deadlines of tasks, activities and milestones, a shared Google Documents spreadsheet was developed to track the project status. During team-wide meetings, this spreadsheet, as shown in Figure 1.2 also acted as the meeting agenda, helping the team stay focused on project tasks and responsibilities. Department tasks was also managed in a similar fashion, but contains additional cost, dependency, risk, importance and other decision metrics.

1.1.3 Project Failure Points and Resolution

A development focus for the team is to use electrically, algorithmically, and mechanically robust methods to reduce the impact of failures. There are three main areas that the team has identified as potential failure points: mechanical implementation (which includes vehicle dynamics), electrical implementation and software implementation. To ensure that the team has continuing high levels of success, the team regularly tests the vehicle beyond the operating bounds of the competition. Testing methodology is covered in Chapter 4. Specific failure points and resolutions are further covered in Chapters 2, 3, and 5.

2. Hardware Design

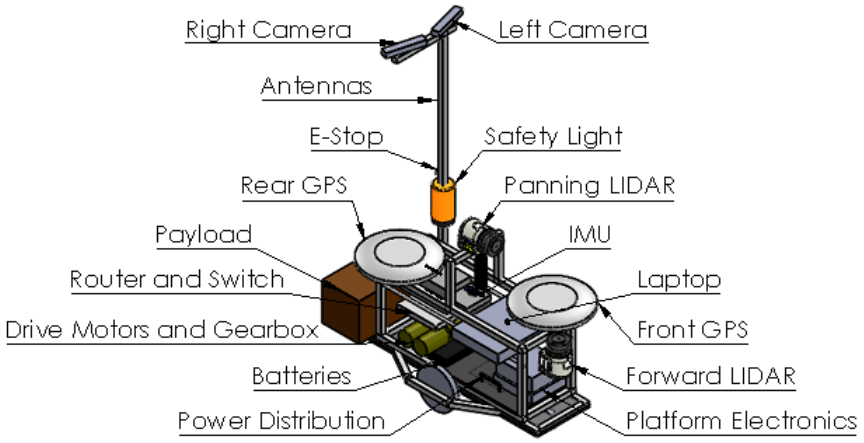


FIGURE 2.1: Platform with components labeled.

This chapter details the hardware design of the platform. Section 2.1 provides an overview of the mechanical design of the platform, Section 2.2 reviews the sensors that are used on the platform, and Section 2.3 details the electronic design. An overview of the safety, reliability, and durability considerations that have gone into the design is given in Section 2.4. A high level costing is shown in Table 2.2.

2.1 Mechanical Design

An iterative design methodology was used to construct the UGV, as the platform was used in other applications prior to the IGVC2015 competition. At each stage of assembly and design, the goal was to maintain a testable state for the next day. CAD modeling using Dassault Systèmes SolidWorks and AutoDesk Inventor was also performed in parallel to assist in the frame design and placement of components, as seen in Figure 2.1.

1" square, 1/8" thick aluminum box and 1/8" aluminum plate was selected as the primary structural materials in the assembly due to their high availability and ease-of-working. The design is fastened together primarily using high tensile steel M6 bolts and locking nuts or washers. This is in contrast to earlier prototypes which used aluminum rivets; however prolonged testing on difficult terrain caused the rivets to shear or loosen. The current design is extremely robust to rough terrain, vibration and high speed manoeuvring and has a low centre of gravity to allow rapid movement.

The design has changed from a 6WD robot for the Australian AGVC competition to 2 wheel differential drive to reduce friction whilst turning which improves the battery life and control of the robot. Three casters, two at the front and one at the back keep the platform stable, and are raised above the drive wheels in order to clear ramps and uneven terrain. We have also designed

it to disassemble into modules to make it easier to transport internationally. The drive base of the platform is powered by a two Toughbox gearboxes from AndyMark, each of which houses two coupled drive motors. With a tested 150 lb payload limit and a top speed of 7 mph, this platform will easily carry the 20 lb payload and operate above the minimum required speed of 1 mph on a 15 percent gradient.

2.2 Sensors

The following sensors are mounted on the vehicle to provide the robot with the ability to navigate autonomously:

Motor encoders CUI AMT103 2048 tick/revolution quadrature rotary encoders are fitted behind the drive wheels.

Inertial measurement unit (IMU) An XSens MTi-G IMU, which is physically small, draws very little power and has extremely low noise density: $80\mu g/\sqrt{Hz}$ and $0.03\,\mathrm{deg}/s/\sqrt{Hz}$, has been fitted to the center of the vehicle.

GNSS Two onboard Trimble NetR9 reference GNSS stations, which uses the platform's cellular connection for RTCM streaming corrections data, are mounted on top of the vehicle. Using real time kinematic (RTK) correction in this way yields a typical location accuracy of $\sigma < 10mm$, when they have a fix.

Laser scanners Two SICK LIDARs, a LMS111 and LMS151, have been mounted to the robot, which has an effective range of about 15 m with the surfaces involved in the competition. One laser is articulated for 3D sensing.

Cameras Two Logitech c920 webcams are mounted on the robot in fixed positions. Two cameras are used to provide greater visible area.

2.3 Electronic Design

Full schematics are available at: `http://snipurl.com/29xtn67`. An overview of the UGV's electrical systems is shown in Figure 2.2.

2.3.1 Communications

The primary method of communication with the UGV is via 2.4 GHz 802.11n WiFi, which is used for system management, control, testing and development. An additional long-range WiFi link is established with two Ubiquiti Picostations which can achieve more than 500 m range, line of sight. The on-board laptop, control electronics, GNSS receivers and LIDARs are all connected via Ethernet to an on-board switch, and are accessible over the WiFi link. The use of standard IP protocols and interfaces allows remote testing and trivial extension and modification of the system.

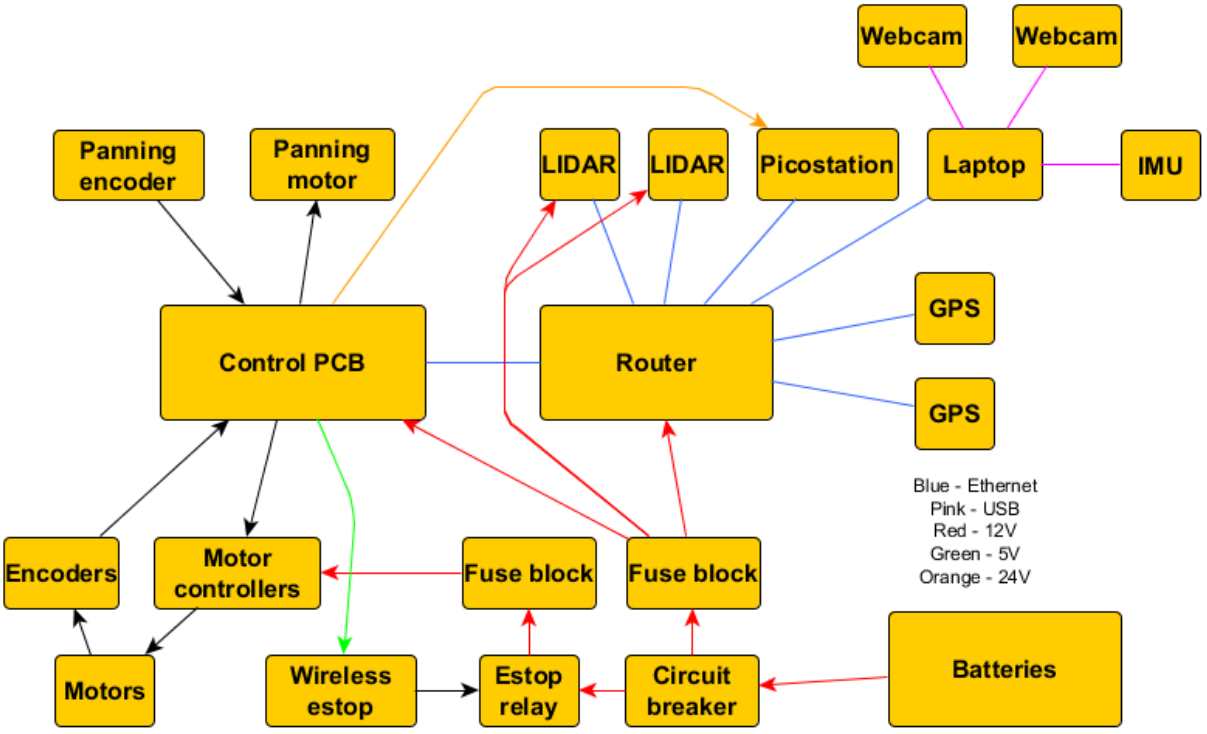


FIGURE 2.2: Overview of platform wiring

2.3.2 Power Distribution and Usage

Two standard 12 V, 18 AH SLA batteries power the UGV. They are wired in parallel to provide the main 12 V power bus which supplies the motors and most of the onboard equipment. In line with the batteries is a 100 A thermal circuit breaker followed by a two-tiered array of blade fuses for safe sub-distribution of power. Drive circuitry is separated from sensors and control circuitry, so a large relay can cut off power to the motors during an emergency stop. The four drive motors are driven by individual Talon SR motor controllers, which we have found to be extremely reliable.

Standard 120 A Anderson connectors are used for the power connections, allowing the robot to connect to an external mains supply instead of the batteries. This allows the sensors and other equipment to be run whilst stationary. This is useful during lab testing and prevents battery drain when not in use.

Power requirements for components of the system are detailed in Table 2.1.

2.3.3 Control board

The control PCB (Figure 2.3) is our own design and interfaces between the control software and all low level electronic systems: drive motors, safety light, panning laser, emergency stop and battery management. It also provides 24 V and 5 V rails from high efficiency buck and boost converters to the components that need them. It is housed in an IP67 polycarbonate enclosure with milspec Ethernet jack to prevent damage and water or dust ingress.

Component	**Power**	**Current and Voltage**	**Source**
2 × SICK LMS111	18 W	2 × 0.75 A at 12 V	Platform
XSens MTi-G IMU	0.4 W	80 mA at 5 V	Laptop
Wireless Router	12 W	1 A at 12 V	Platform
2 × Logitech C920	4.5 W	2 × 450 mA at 5 V	Laptop
Trimble GPS Receiver	3.8 W	-	Internal
Control Electronics	1.5 W	125 mA at 12 V	Laptop
Safety Light	6 W	500 mA at 12 V	Platform
Laptop	35 W	-	Internal
Picostation WiFi Bridge	8 W	0.33 A at 24 V	Platform
4 × Motors	720 W	4 × 15 A at 12 V	Platform

TABLE 2.1: Power requirements of the components that make up the platform.

The control board uses a dsPIC microcontroller (MCU) connected to an Ethernet PHY. The MCU runs a TCP/IP stack with a web server for diagnostics and a UDP server for control and status messages. One of the foremost goals in the design of the UGV is flexibility and ease of development. By using a network interface to the control electronics, instead of USB or similar, developers can connect to the network, immediately check the status of the platform, independently develop and test firmware and software without requiring a physical connection to the robot.

True quadrature decoding and hardware glitch filtering of the wheel encoders ensures low noise odometry data for the drive control loop. The battery voltage is constantly monitored and operators are warned if the battery runs low. The control board includes input filtering to mitigate against EMI from the motors and transient voltage dips and spikes.

A custom UDP protocol interfaces the control board to its driver on the laptop. This protocol is timestamped and checksummed to ensure safe and in-order processing of packets. The protocol is asynchronous, which interfaces well with the software architecture described in Chapter 3.

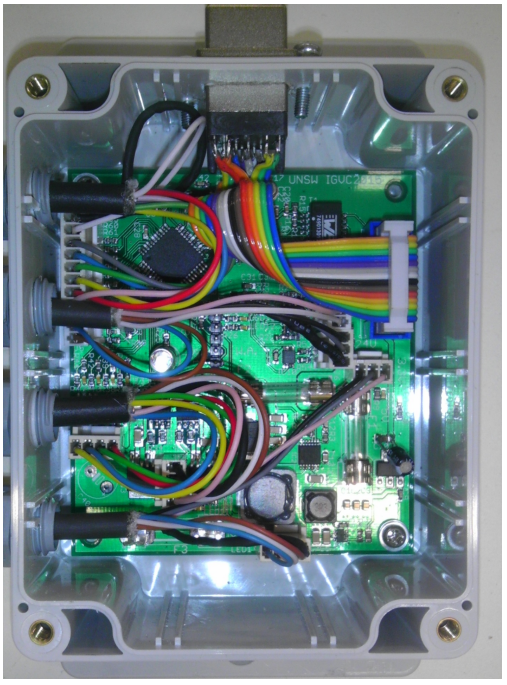


FIGURE 2.3: Control board in sealed polycarbonate enclosure (cover removed)

2.3.4 Emergency stops

The UGV features two emergency stops, in compliance with IGVC rules. There is an on board emergency stop at the back of the robot and a fail safe wireless emergency stop operable at a range of up to 100 m. Both devices are wired in series and either one will bring the UGV to a quick and complete stop. The wireless emergency stop is implemented with nRF24L01+ ICs which utilise a simple but robust handshaking protocol to mitigate interference and fail safe. Most of the time the transmitter enters a low power state, drawing only about 1 uA. This allows us to gain in excess of 1000 hours of battery life from two AAA batteries.

2.4 Safety, Reliability and Durability

To ensure the safety of people working on or around the UGV, safe working procedure, and risk management forms were produced: `http://snipurl.com/29dm898`.

Component	**Cost**	**Cost to Team**	**Type**
Custom platform	$1,000	$1,000	Platform
SICK LMS111	$4,500	Loaned	Sensor
XSens MTi-G	$2,300	Loaned	Sensor
Wireless router	$80	$80	Assembly
Ethernet switch	$8	$8	Assembly
2 x Logitech c920	$150	$150	Sensor
2 x Trimble NetR9	$41,000	Loaned	Sensor
Frame and fasteners	$300	$300	Assembly
Custom electronics	$100	$100	Assembly
Laptop	$850	$850	Assembly
Total	$50,310	$2,480	

TABLE 2.2: Costing for the project, if all of the components were bought new. The majority (95 %) of the cost is sensors.

At previous competitions, the team experienced reliability issues with USB connections, in terms of connector durability and signal integrity. Moving to an Ethernet based controller has greatly improved safety and reliability of the platform. As Ethernet is higher voltage swing and fully isolated, it negates signal integrity issues. The robust and standard packet model ensures data integrity, wide compatibility and easy development.

All sensors and their drivers are able to cope with momentary connection loss and data corruption caused by connector vibration, EMI or other data integrity issues. This has required modification and bug fixes in existing open source drivers and careful component placement.

The control board uses a watchdog timer and careful fail-safe reset handlers to ensure that any firmware bugs do not impact on the safety of the vehicle.

All structural members are fastened using high tensile capscrews, shakeproof washers and tapped holes or locking nuts. Foam rubber has been placed in some joints, to protect sensors and damp vibration. Extensive testing has proved the robot to be highly resilient to vibration and impact; no damage has been caused driving at high speed over very rough terrain. Fluted plastic board is used to form a shell around the vehicle, shielding it from rain and splashes.

An external frame provides the platform, sensors and payload protection against damage from collisions and rollovers.

3. Software Design

This chapter provides an overview of the software design and integration that provides the UGV with the ability to perform autonomous navigation.

The software systems that work together in order to provide this functionality are:

- **Localization:** determining the location of the vehicle in the world.
- **Mapping:** determining what the environment around the vehicle looks like, and where it is safe to drive.
- **Navigation:** determining how to drive and control the vehicle in order to reach the destination.

Each software system is composed of several processes that each has a singular purpose. Interprocess communication is handled with Robot Operating System (ROS). ROS also provides a number of standardised message formats to disseminate data for processing, allowing a high degree of modularity. Numerous open-source drivers have already been developed for the ROS architecture, allowing us to trivially integrate new sensors into our software systems. However in many cases we have needed to improve the performance or robustness of these drivers to produce a reliable and responsive system. Although a number of open source ROS software packages and algorithms are already available, we have found these to deliver unsatisfactory performance and so have developed our own systems which fit into the same flexible architecture.

ROS uses socket based communications, allowing rapid and extensible modular development and debugging across the platform's internal network and WiFi. This model also removes the need for an user level event based system to poll for changes, as required in shared memory systems, and has lower-latency than message brokering systems [1]. For system management, integrated monitoring and control software was developed to watchdog process and system states, and is covered in Chapter 5. A CPU performance budget is used to ensure that the vehicle can detect and replan in under 250 ms, while offering over a 3 hour single-charge run time.

3.1 Localisation

The vehicle performs localization using an Extended Kalman Filter (EKF), which fuses the sensor data calculated by the IMU (estimates angular velocity), motor encoders (estimates linear velocity) and GPS (estimate position in the world).

One of the key concerns of the approach to localization used in a recent competition was that the vehicle needed to be driven around for approximately 30 seconds in order for an accurate estimate of the platform's heading to be obtained. This problem has been solved through the addition of a second GPS unit; by combining measurements from the two GPS sensors fixed at different points on the vehicle, heading determination is achieved [2].

3.2 Mapping

An overview of the approach to mapping in this year's entry is illustrated in Figure 3.1 below.

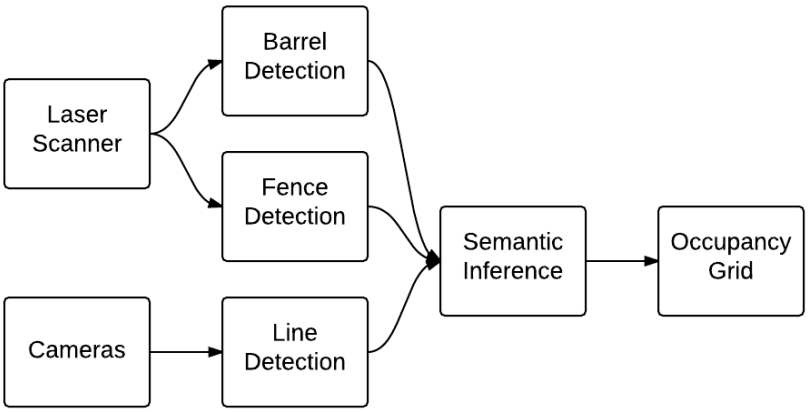


FIGURE 3.1: Overview of the approach to mapping.

The overall approach to mapping utilizes an occupancy grid to formulate a map of the environment that determines whether or not certain areas of the areas occupied by obstacles or free space. This occupancy grid is filled in by observations of the barrels, fences, and lines, as detailed in Sections 3.2.1 and 3.2.2.

A key innovation in this approach is in the use of semantic identification of obstacles within the sensor data that is collected. This means that individual barrels are detected in the laser data, and similarly lines are detected in the camera data. The advantages of this approach are two-fold. Firstly, this has resulted in a significant reduction in the amount of spurious data that is added into the map, such as areas of grass that appear quite white, or laser scans that hit the ground. The second advantage is targeted at an issue that arose in a previous year's competition, which was that under certain lighting conditions (such as direct sunlight) barrels would appear to be white, meaning that they would be detected as being lines, and erroneously added into the map. As explained in further detail in Section 3.2.3, by combining the information collected from the laser and camera, we are able to avoid this problem.

3.2.1 Obstacle Detection

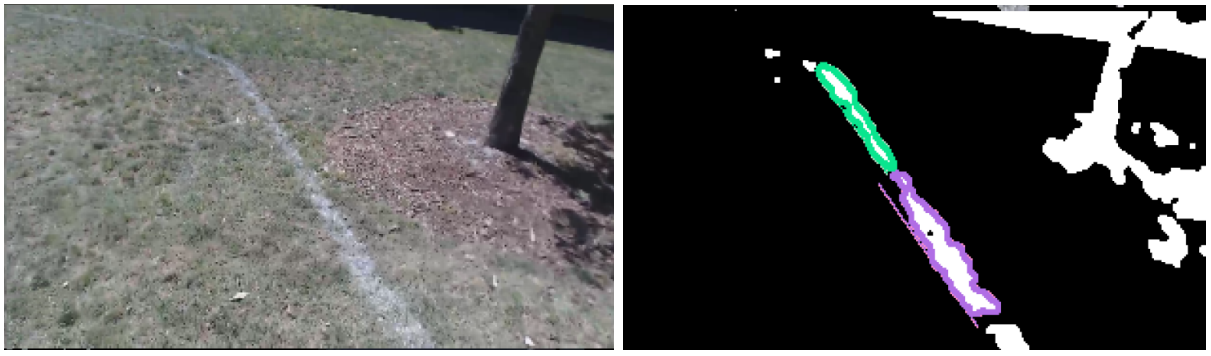
Obstacles are detected using computational geometry to identify geometric objects from laserscan data as discrete entities. A two-stage process allows barrel and fence detection to be performed simultaneously. Collected laser data is firstly segmented into regions of points in close proximity to one another. Regions are then classified into course elements such as a barrel, fence, ramp, or the ground. As mentioned previously, the advantage of this process is the absence of erroneous laser data during map generation. This process is illustrated in Figure 3.4 below.

Angle-stamped scans from the panning LIDAR are processed individually to approximately identify complex obstacles which are not clearly present in the single horizontal plane of the front-facing LIDAR. This method is faster than identifying objects in a 3D point cloud and compensates for the vehicles' rapid movement during the course of the pan.

3.2.2 Line and Pothole Detection

Robust and fast line detection is achieved by the use of a Bayesian particle filtering image processing methodology. A number of particles are distributed across the image, and the likelihood of a particle being on a line or pothole is obtained from an structure-specific objective function utilizing a trained Bayesian classifier [3]. The variance of the best solutions is used as a confidence measure to eliminate false positives.

This process is illustrated in Figure 3.2 below.



(A) Image collected from camera. (B) Line pixels and lines detected.

FIGURE 3.2: Line detection.

Note that two cameras have been mounted on the vehicle at slightly different locations in order to provide a greater field of view, allowing the robot to see lines next to itself.

3.2.3 Semantic Inference

As mentioned previously, the detected barrels, lines and fences are combined together in a further post-processing stage in order to eliminate erroneous detection of lines. The basic concept behind this is that at any point in time, we should not be able to observe lines behind a physical obstacle, as vision should be occluded by that obstacle. Unclassified persistent objects, such as flags, are also treated as obstacles to be avoided. Flag directionality can be identified with image processing.

3.3 Navigation

The autonomous navigation performed by the vehicle has a hierarchical structure with three levels; goal management, path planning, and path following. These components are detailed in the following sections.

3.3.1 Goal Management

A high-level waypoint module is responsible for managing the progress on the list of objectives that the vehicle must navigate through. As soon as the current objective is completed, the next waypoint is issued to the path planning subsystem.

3.3.2 Path Planning

The goal of path planning is to determine a suitably optimal path through the environment that will allow the robot to safely reach a destination waypoint from its current position, while minimizing an objective function involving items such as distance and risk.

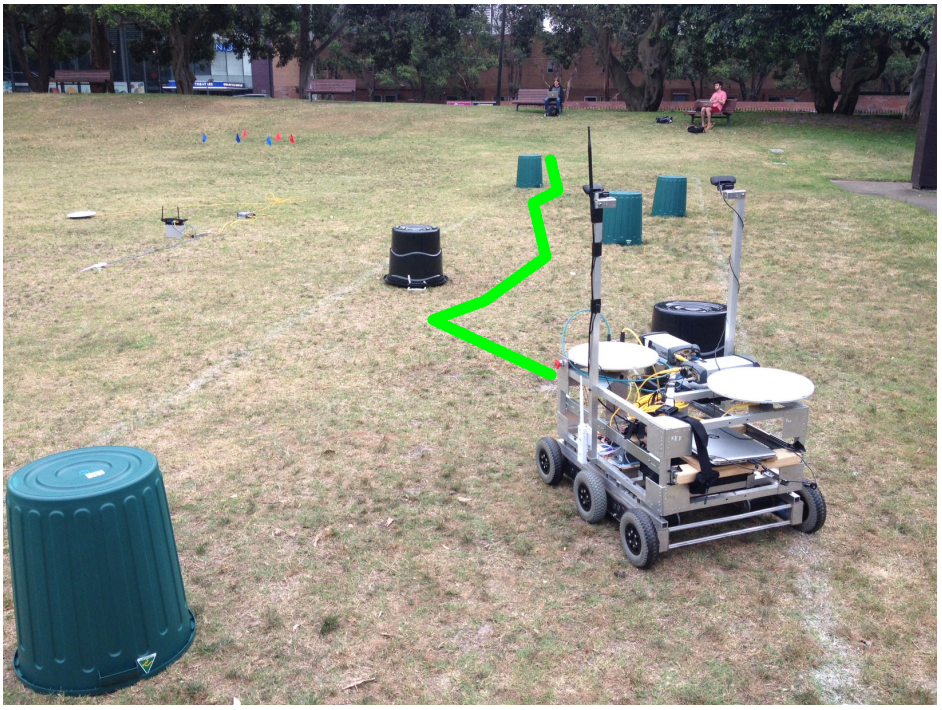


FIGURE 3.3: A photo of a miniature course with lines, barrels and flags.

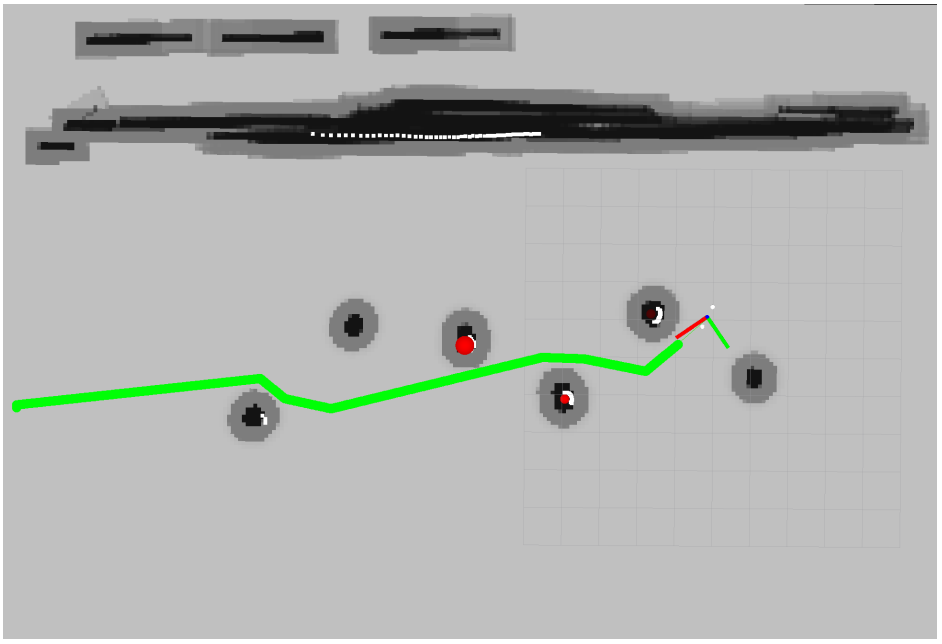


FIGURE 3.4: Sensor data, costmap, detected hazards, and path plan.

The path planning makes use of the occupancy grid generated at the mapping stage. Celebrated graph-based planning algorithms can then be employed to find the shortest path between waypoints on the map. A dynamic programming approach applied by the team utilizes evolutionary particle swarm optimization to find a fast and smooth path and speed profile through the map [4]. An attraction heuristic from a modified version of A* is applied to take advantage of the large areas of open space present in the IGVC environment. An example of the paths generated through the initial path planning stages is shown in Figures 3.3 and 3.4.

Just knowing the path to goal is insufficient, the robot must also be able to follow this path accurately if it is to reach the goal. The approach to path following is detailed in the following section.

3.3.3 Path Following

The following of the paths generated in the previous stage of the navigation process is governed by the application of sliding mode control, which is commonly used when controlling systems with nonlinear dynamics [5]. To support high speed maneuvering, a electronic stability control (ESC) system was added which uses the difference between the vehicle's actual behavior and the commanded behavior to determine when and where counter-torque should be applied to stabilize the vehicle. This ESC system counters the effect of body roll, which can lift the inside tire off the ground causing loss of traction during fast turns.

3.3.4 Drive control

A low level control loop for the vehicle's drive wheels runs on the laptop at 100 Hz or more. The low latency and low overhead of the BSD socket architecture in conjunction with our UDP message protocol makes it possible to achieve high performance, stable and real time control in software without a real time . Typical round trip control loop latencies over the network from the laptop to the control board are on the order of 1 millisecond or better. Having a high level controller implementation allows straightforward testing, development and adaptive tuning without cumbersome firmware modification.

4. Testing

The chapter outlines the team's approach to the testing of autonomous navigation of the platform. This testing involved three stages; simulation, small-scale real-world testing, and large-scale real-world testing. The following sections detail each of these stages.

4.1 Simulation

A new simulation environment has been engineered specifically for development and testing for IGVC. This simulation has a number of features that have been critical to the development of our teams solution:

- **Rapid testing:** the simulation is capable of being run at up to five times faster than real-time and in parallel. To facilitate repeated and rapid testing, the team has rented several cloud computer instances to run many iterations of the simulation simultaneously.
- **Logging and log analysis:** as mentioned above, we are able to rapidly iterate over our simulation. Unfortunately this results in too much data than can be easily analyzed manually, and so we have developed several tools to automatic analyze and collect statistics regarding the performance in a simulated run of the competition. These statistics, which include average speed, localization error, and proximity to obstacles, allow for quick tuning and verification of parameters to determine which combination of these parameters optimizes the performance of the system as a whole.

Unfortunately, a simulation can only come so close to replicating real-world performance. One of the shortcomings to our preparation for last year's competition was that we felt that we did not conduct enough real-world testing. As such, there has been a much heavier focus this year on conducting such tests, as detailed in the following sections.

4.2 Small-Scale Real-World Testing

In order to verify and test the performance of the system in the real-world, the team purchased a number of correctly-sized bins and organized the marking of an approximately 60 foot long lane of white lines on the UNSW campus. This setup, seen previously in Figure 3.3, has been used to analyze the performance of the integrated system with actual lines and barrels.

4.3 Large-Scale Real-World Testing

It was not practical or feasible to be able to mark out enough lines or acquire enough barrels to be able to recreate the entire IGVC course. However, the team still felt that it was necessary that

we do a full-scale test of the system with the actual platform in the real world, rather than in simulation.

In order to accomplish this, a tool was developed that allowed the user to mark out a virtual competition course in the real-world. This tool is shown in Figure 4.1 below.

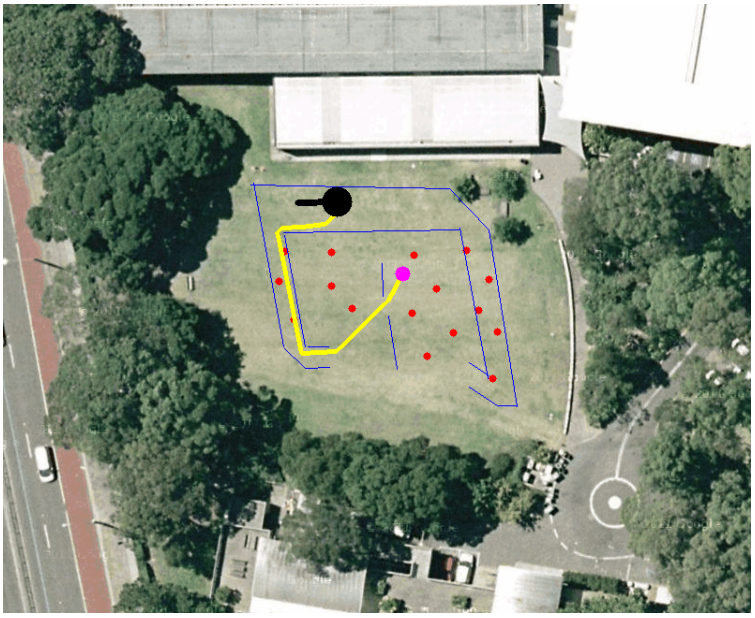


FIGURE 4.1: Large-scale testing was facilitated through a tool that allowed virtual courses to be marked out in the real world.

By using this tool, we have been able to mark out, virtually, a full-sized obstacle course that the robot is able to attempt to navigate.

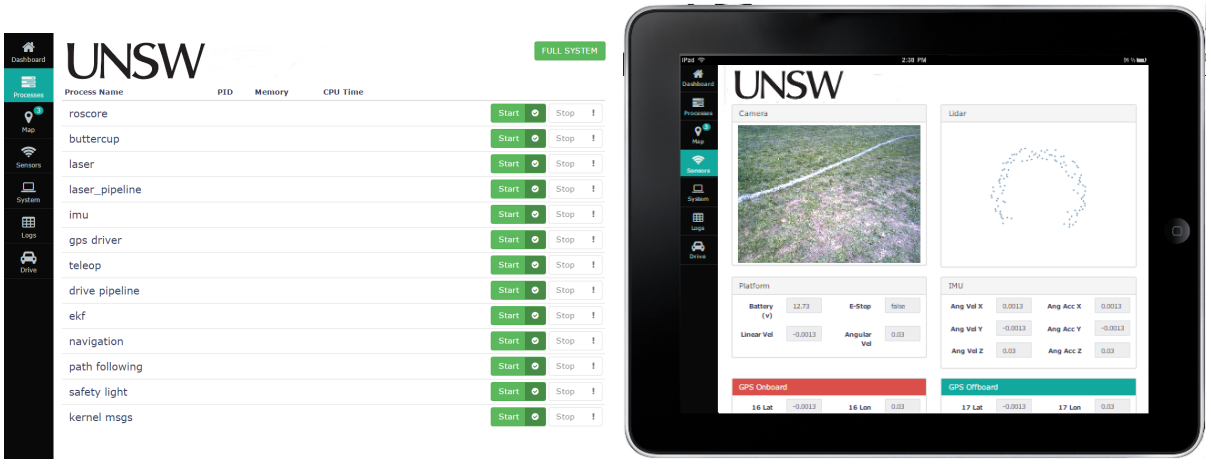
4.4 Failure Testing

Extremes such as vehicle rollover, low-battery operation, connectivity loss, power loss, loss of sensors, and abnormal process termination are intentionally induced during testing to ensure that the entire system itself is fault tolerant, such that it recovers automatically, or that safeguards and monitoring systems are in place to prevent or to alert the operator of system errors.

As a policy, to guard against intermittent component or system availability, automatic recovery from disconnections through distributed stateless design and fast reinitialization has been implemented wherever possible. In addition, almost all open-source low-level device drivers and software used by the team has been reworked to operate at the team's specifications.

As the vehicle dynamics for the platform change with every iteration, a potential for rollovers and vehicle spin due to loss of vehicle stability was identified during design and testing. To resolve this problem, an electronic stability control system was added to regulate vehicle dynamics such that it was always within safe operating bounds.

5. Integration Management



(A) Process control interface.

(B) Sensor feedback on a tablet.

FIGURE 5.1: Browser-based remote monitoring and control system for the ground vehicle.

Unfortunately, most robotic platforms that are developed for purposes such as tasks like IGVC require an extraordinarily in-depth knowledge of the inner workings of the system to be able to operate it. At best, many systems will only provide an untrained user with remote operation using a joystick. For commercial viability, users of autonomous systems need the ability to remotely monitor the system without having an intricate understanding of the implementation details. For this problem, one of the innovations this year is the introduction of an intuitive web-based user interface that is operable on any network-enabled mobile device (phone or tablet), laptop or desktop.

By offering a web-based interface (as shown in Figure 5.1), a variety of tools such as process controls; waypointing, map and database visualization and editing; sensor feedback; audible system health monitoring; data logging and playback; and touch-based drive command systems are available. This simplification of several components such as the startup, diagnostics and drive system, enhances the vehicle's overall deployability and usability, and complements existing command-line processes or other advanced full-feature visualization and simulation capabilities. In a casual usability test, users reported ease of use with the new graphical system than the previous system.

6. Conclusion

In summary, the team has proposed a custom-built UGV based on commercially available components. The design also adapts existing hardware, software, and algorithms to autonomously perform localisation, mapping, and path planning. The system we have developed is highly modular and based on standardised and proven interfaces to allow easy and rapid addition of new sensors and capabilities. A significant amount of improvements have been made based on the lessons that we learned from the Australian AGVC competition, as well as based on the significant amount testing that has been conducted by the team throughout the year.

Overall, the theme of development has been to have a versatile and expandable system with a robust back-end with a user-friendly front-end. This has resulted in a system that has been proven through extensive simulation and real-world testing to be able to reliably complete the challenges that IGVC poses, whilst still being able to be controlled by anyone, anywhere in the world.

Bibliography

[1] G. Hohpe and B. Woolf, *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2004.

[2] L. Lau, P. Cross, and M. Steen, "Flight Tests of Error-Bounded Heading and Pitch Determination with Two GPS Receivers," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, pp. 388–404, Jan. 2012.

[3] M. J. Jones and J. M. Rehg, "Statistical color models with application to skin detection," *International Journal of Computer Vision*, vol. 46, no. 1, pp. 81–96, 2002.

[4] V. Miranda and N. Fonseca, "Epso-evolutionary particle swarm optimization, a new algorithm with applications in power systems," in *Transmission and Distribution Conference and Exhibition 2002: Asia Pacific. IEEE/PES*, vol. 2, pp. 745–750 vol.2, Oct 2002.

[5] J. Taghia, S. Lam, and J. Katupitiya, "Path following control of an off-road track vehicle towing a steerable driven implement," in *2015 International Conference on Advanced Intelligent Mechatronics*, IEEE/ASME, 2015.