

5/10/2013



ROBOT DESIGN REPORT: SEGFAULT

Ultra
ELECTRONICS



 **DALHOUSIE UNIVERSITY**
Inspiring Minds

Faculty of Engineering

Reg Peters, Faculty Advisor

I certify that the Dalhousie Robotics student team has completed significant engineering and software design challenges that are equivalent to or exceed that which would be expected in a senior year course.

Table of Contents

Table of Contents	1
1. Introduction	2
1.1. Team Overview	2
1.2. Robot Highlights.....	2
2. Design Process	3
2.1. Planning and Process Steps.....	3
2.2. Problems and Solutions.....	4
3. System Architecture.....	4
4. Intelligence Strategy	5
5. Hardware Design.....	6
5.1. Chassis.....	6
5.2. Power Train	7
5.3. Power System.....	7
5.3.1. Component Selection	8
5.3.2. Wiring	9
5.4. Sensors	9
5.5. Payload Computers	10
5.6. Safety Systems	10
6. Software Design	11
6.1. Localization	11
6.2. Vision Processing.....	12
6.3. Map Making.....	13
6.4. Path Planning.....	14
6.5. Decision Making.....	14
6.6. JAUS.....	14
7. Robot Analysis.....	15
7.1. Cost Analysis	15
7.2. Performance Analysis.....	15
8. Conclusion.....	15

1. Introduction

1.1. Team Overview

Dalhousie Robotics is a new team comprised of undergraduate and graduate students from computer science and various engineering disciplines. The project does not count for any course credit and is funded mainly by industrial and departmental sponsorship. The team is comprised of three working groups. The mechanical group designed and fabricated the robot chassis and drivetrain. The electrical group designed and wired the control circuits and power systems. The software group programmed sensor interfaces and intelligence routines. A parallel design process allowed each group to remain productive while interfacing components were still under development.

President: Dainis Nams, <i>Mechanical Engineering</i> Vice President: Kathleen Svendsen, <i>Computer Engineering</i>	
Mechanical Janis Nams <i>Mechanical Engineering</i>	Software Jeff Rouleau <i>Computer Science</i>
Jose Posada <i>Mechanical Engineering</i>	Matt MacDonald <i>Computer Science</i>
Electrical Alex Parker <i>Electrical Engineering</i>	Rober Boshra <i>Computer Science</i>
Brian McDonald <i>Electrical Engineering</i>	Rohan Bhargava <i>Computer Science</i>
Ed Barchard <i>Electrical Engineering</i>	Ross Story <i>Computer Science</i>
Irene Jantz-Lee <i>Electrical Engineering</i>	Paul Hollensen <i>Computer Science</i>
Mu He <i>Electrical Engineering</i>	Tim Pohajdak <i>Mechanical Engineering</i>

Table 1: Dal Robotics Team

Weekly group status updates and monthly team meetings ensured clear communications throughout the design process. Github was used for code source control and Dropbox was used to maintain administration documents.

1.2. Robot Highlights

As a rookie team, we have no robot from prior competitions to serve as a baseline for innovation. Having built a functional robot from scratch components in eight months, we feel that the entire vehicle is innovative! However, we would like to showcase several highlights we feel particularly distinguish Segfault:

- Easy maintenance access to any component is afforded by the quickly removable outer shell and the gas springs that automatically expose the power train by retracting the electronics bay.
- The lane-finding algorithm incorporates machine learning to individually compensate for the lens distortion present in each webcam, improving accuracy of detected white lines.

- The low center of mass, independent suspension, and four wheel drive allow for high speed manoeuvring while silicone vibration mounts protect the payload computers from damage.

2. Design Process

Having no IGVC pedigree, no hardware from previous entries, and no direct faculty guidance, the Dalhousie Robotics student team began the design process with a clean slate. The following overarching design focuses were identified at start of term as being the most important for our robot:

- 1) *Safety* – vehicle design must ensure the safety of team members and bystanders alike.
- 2) *Rules* – vehicle design must adhere to all competition rules.
- 3) *Reliability & Maintainability* – the vehicle hardware must be capable of hours of sustained operation and easily maintained in the event of malfunction.
- 4) *Modular Software* – the software must employ a centralized database with subroutines following a *publish/subscribe* paradigm in order to modularize software components.
- 5) *Simplicity* – the vehicle must be driving by end of first term and capable of autonomous motion by end of second term, and must therefore be simple enough to construct, build, and program within that time period.

The primacy of these five focuses reduced the priority of other design concerns such as weight and efficiency. The result is that Segfault is large, heavy, and has mediocre battery life. However, this sacrifice was acceptable as the ambitious construction schedule bought the team an invaluable six weeks pre-competition to test autonomy code on a completed and reliable hardware platform.

2.1. Planning and Process Steps

The team began with brainstorming sessions focused on deriving internal design requirements from the competition rules and the team's constraints such as resources and knowledge. Once the initial requirements were developed, design was divided into parallel hardware and software tracks with the major steps shown in Figure 2.

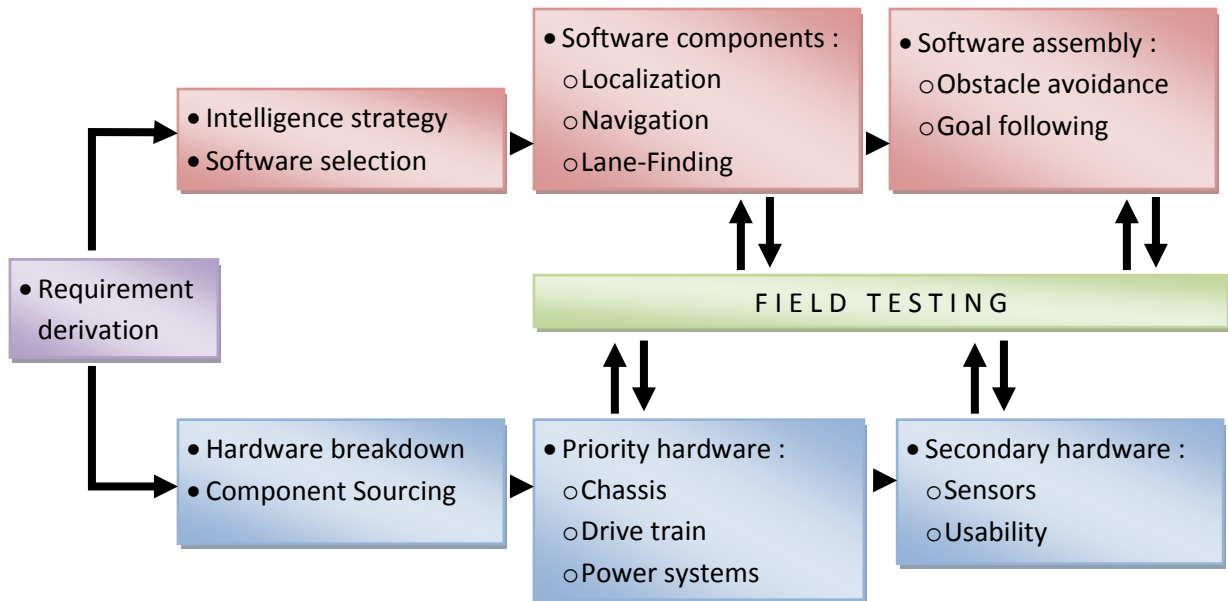


Figure 2: Design Process Steps

Basic hardware construction was accelerated in order to facilitate frequent outdoor reliability testing of all components throughout the design process.

2.2. Problems and Solutions

A variety of unexpected problems were encountered throughout the design process, with Table 2 chronicling three of the most interesting encountered.

Problem	Solution
Wheels gripping unequally on non-flat surfaces	Post-machine wheel mounts to add simple spring-pivot four wheel suspension system
Magnetic compass in inertial measurement unit (IMU) distracted by magnetic fields from DC motors	Move IMU to the robot “head” to remove from effects of motor magnetic fields
Repeated crashes of Robot Operating System (ROS) software interface to SICK LMS111 laser rangefinder	Re-write interface core and upload source to community for benefit of all ROS users

Table 2: Solutions to Design Problems

3. System Architecture

A primary focus of the early design process was to establish the system architecture. The vehicle was modularized by dividing it into vehicle platform and autonomy payload modules. As seen in Figure 3, these modules are linked only by motor control and feedback signals, allowing them to be developed in parallel and tested independently.

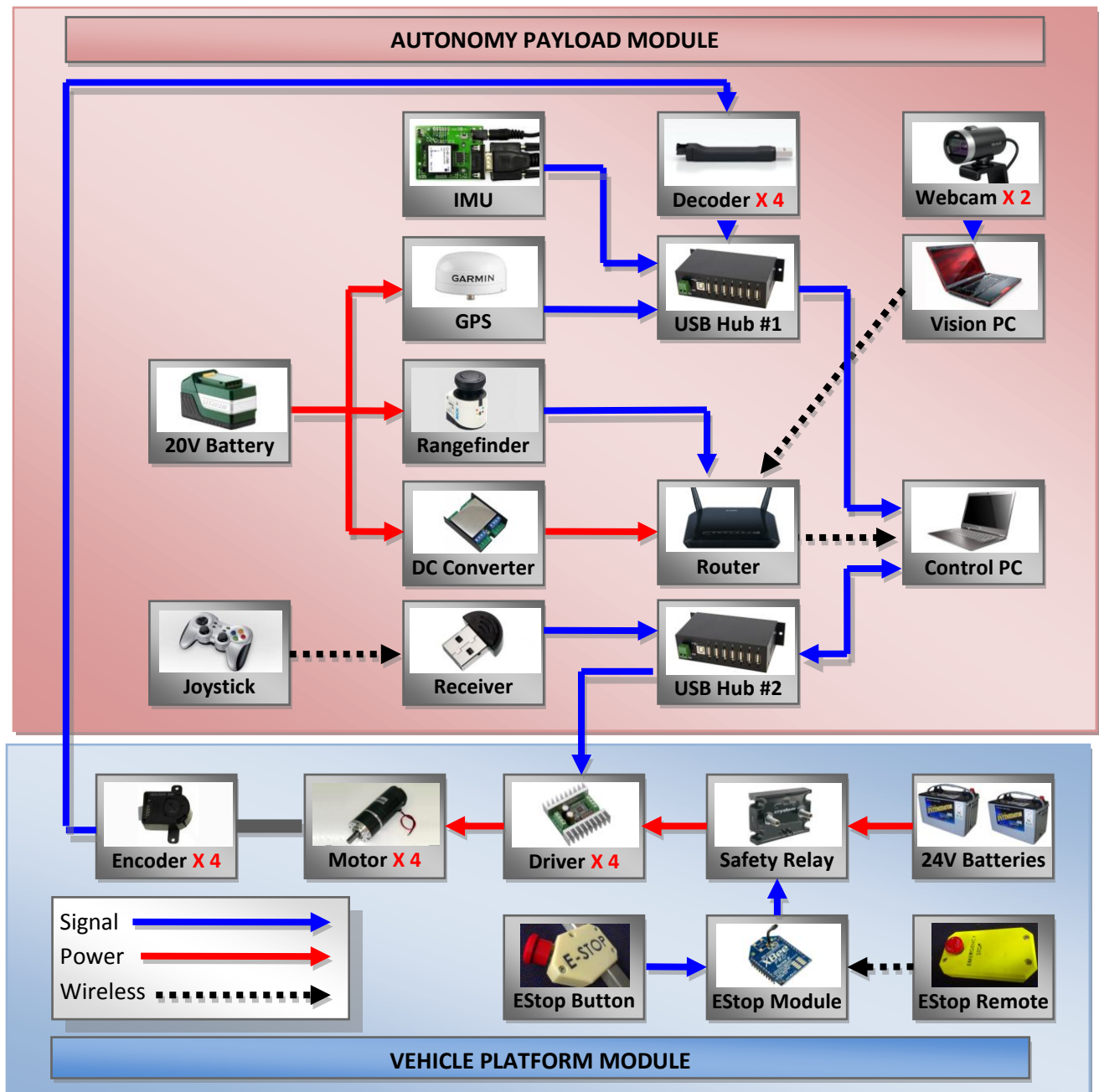


Figure 3: System Architecture

4. Intelligence Strategy

Once the overall system architecture had been determined, the next step was to devise the intelligence strategy and subdivide the software development into sections for parallel development. Figure 4 details the architecture of the intelligence strategy.

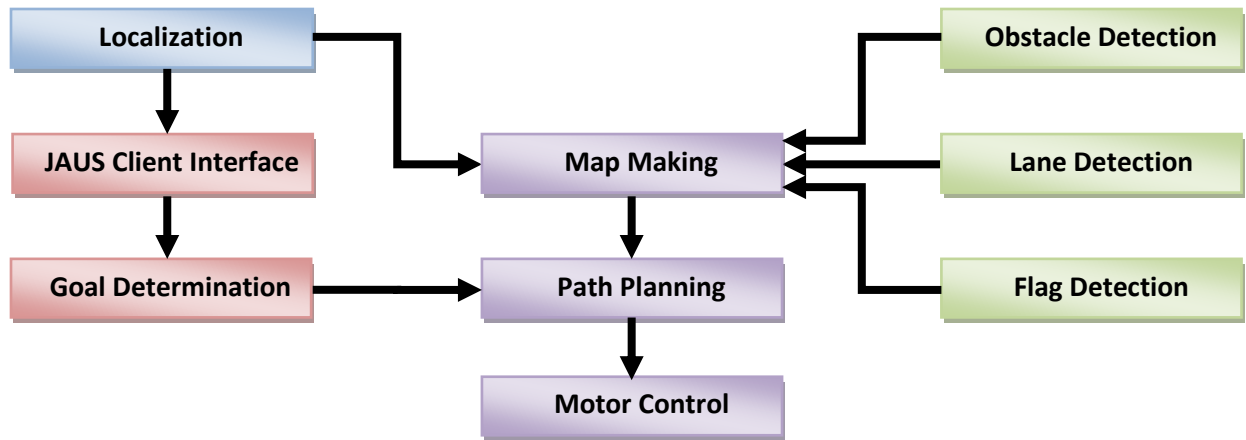


Figure 4: Intelligence Strategy

5. Hardware Design

5.1. Chassis

Segfault’s frame is a simple stacked rectangle design. The frame is fabricated from aluminum square tubing for its high strength-to-weight ratio, and is welded for rigidity. The heavier components, including batteries and motors, are housed in the bottom of the frame to keep the center of gravity low. The electronics and other sensitive components are housed in a shock-isolated

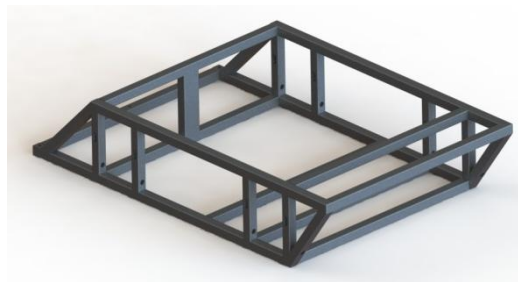


Figure 5: Bare Chassis



Figure 6: Gas Springs

compartment above the aluminum superstructure. The electronics compartment hinges upward, supported on gas pistons, providing easy access to the components below. The frame is clad in aluminum paneling, and a custom acrylic rain guard covers the electronics compartment to protect all components from weather. Cameras, GPS receiver, and inertial measurement unit are housed in Segfault’s “heat” atop an aluminium square tubing mast under an independent rain guard.

5.2. Power Train

The vehicle is driven by four independent brushed DC motors fitted with 15:1 reduction planetary gearboxes. The motors are rigidly coupled to the four drive wheels, providing independent control of each

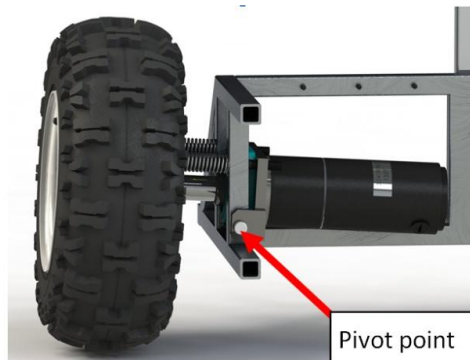


Figure 7: Spring Suspension

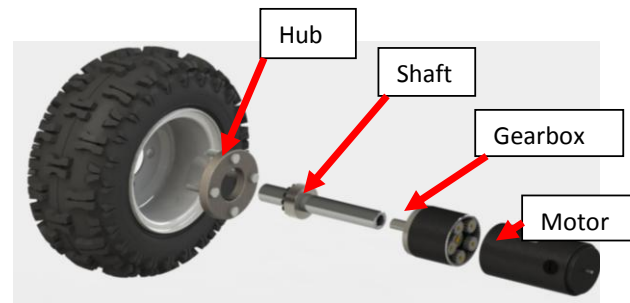


Figure 8: Wheel Assembly

wheel. The use of four motors is also important to ensure our heavy vehicle can handle slopes and rough terrain without bogging down. Each motor-wheel assembly is mounted to the chassis on a pivot joint, and balanced against a pair of suspension springs. This allows each wheel to pivot up or down to ensure all wheels maintain traction over uneven terrain.

5.3. Power System

The power and signal flow of the robot's power systems are shown in Figure 9. Yellow indicates power from the 20V electronics battery, red is power from 24V main batteries, and blue is signal. The maximum current and power ratings are shown; the components were selected so as to not exceed these maximums.

There are two main power sources: 20V is used for sensors (see Section 5.4), while 24V is used for high-power items power such as the inverter and the motors. This separates the sensors from the motors' inductive load.

As a safety feature a relay controls the main 24V power output (see Section 5.6). Wired to this is a set of 24V fans which is always on when the robot is powered. The 12V digital input is controlled by the payload computer, blinking head-mounted LEDs when the machine is autonomous mode.

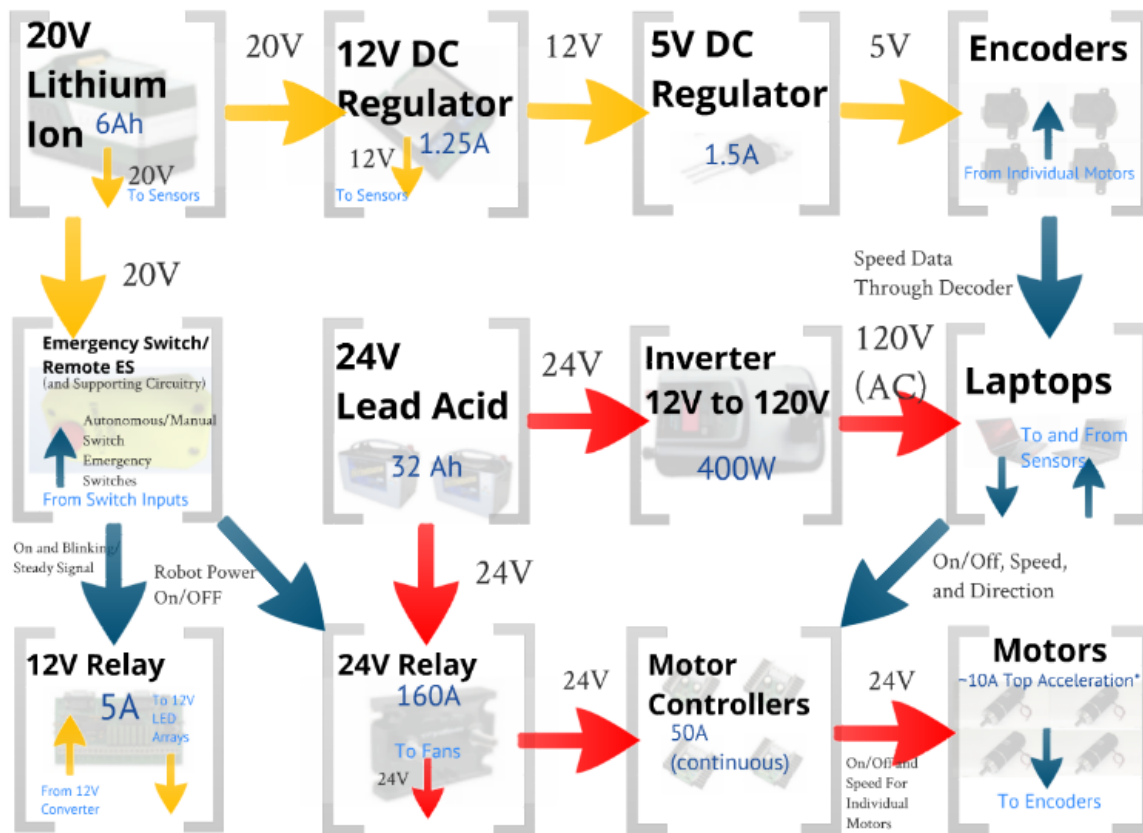


Figure 9: Top Level Power Schematic.

5.3.1. Component Selection

The power system was designed around several main components: the batteries, the motor drivers, and the routing and safety circuitry. The batteries were selected based on criteria found in Table 3.

Lithium Ion	Lead Acid
Half the weight of lead acid	The cheapest option
Easier to misuse – eg. improper charging	Robust (deep cycle)
Chosen for supplementary sensor power	Chosen for main vehicle power

Table 3: Battery Selection Criteria

The 8AU1H sealed lead acid batteries have 32 Amp-hours, 500 recharge cycles (assuming 50% discharge per use), and have handles for reliable loading into the robot. Custom designed keyed connectors allow for easy connection and prevent reverse polarity connections. The batteries have been tested for a minimum run time of one hour, and have a set of backups for easy swap-outs.





The team is using SyRen 50 motor controllers by Dimension Engineering. These compact, 50A (continuous) controllers were chosen because they have built in overcurrent protection, regenerative braking, and can handle PWM (Pulse Width Modulation.)

5.3.2. Wiring

Segfault has two sets of din rails mounted terminals on a multiple layer electronics stack. This innovative solution means devices can be added with ease, the circuit can be changed dynamically, and circuit components and wiring are more efficiently placed. Three layers of electronics are placed on clear lexan for ease of troubleshooting. The lower layer services the motors and controllers with 24V. The middle layer has terminals for sensors, laptops, and emergency stop circuitry; it houses a small 20V lithium ion battery. There are also 12V and 5V regulators to handle the needs of some of the sensors. A 400W inverter is provided to power the laptops through extended testing periods.

5.4. Sensors

Segfault's five primary sensor types are detailed in Table 4:

Sensor	Technical Details		Implementation in Segfault
 <p>Encoder</p>	<p>Measures: Wheel position</p> <p>Quantity: 4</p> <p>Location: Motor encoder shafts</p> <p>Accuracy: 0.05 degrees</p> <p>Manufacturer: Avago Technologies</p> <p>Part Number: HEDS-5600#A06</p> <p>Type: Optical Quadrature</p>	<p>The encoders measure wheel position from which velocity is computed. This is used in two applications. First, each motor's velocity is input to a speed feedback PID controller which converts each of Segfault's basic DC motors into velocity-controlled servomotors. Secondly, the wheel velocity values are used as an odometry input to the localization routine's Kalman filter.</p>	
 <p>Inertial Measurement Unit (IMU)</p>	<p>Measures: 3 axis orientation</p> <p>Quantity: 1</p> <p>Location: Head</p> <p>Accuracy: 2 degrees – Pitch & roll 5 degrees - Yaw</p> <p>Part Number: UM6</p> <p>Manufacturer: CH Robotics</p> <p>Type: MEMS sensors</p>	<p>The IMU is primarily used as a compass, with its yaw output being used to convert between the GPS and robot reference frames in the Kalman filter. Its pitch and roll measurements are also used to detect "ramp" conditions on the course.</p>	
 <p>Global Positioning System (GPS)</p>	<p>Measures: Global position</p> <p>Quantity: 1</p> <p>Location: Head</p> <p>Accuracy: 3 m 95%</p> <p>Part Number: 19HVS</p> <p>Manufacturer: Garmin</p>	<p>The WAAS-enabled GPS is used as a global position input to the kalman filter. It has less precision than the IMU and encoders, but it does not suffer drift and provides bounded error measurements throughout the entire course.</p>	
 <p>Laser Rangefinder</p>	<p>Measures: Range to obstacles</p> <p>Quantity: 1</p> <p>Location: Chassis Front</p> <p>Accuracy: 30mm</p> <p>Part Number: LMS111</p> <p>Manufacturer: SICK</p>	<p>The weatherproof 2D scanning laser rangefinder is Segfault's primary obstacle detection sensor. With a max range of 20m at 270° angular span, the LMS111 is well-suited to detecting obstacles far before Segfault approaches them.</p>	


	Measures:	Stereo visual feed	Two webcams are used as a stereo video source in lieu of a costly industrial stereo camera. To preserve stereo calibration the cameras are rigidly mounted to a jig, which swivels to allow easy adjustment of the field-of-view.
	Quantity:	2	
	Location:	Head	
	Accuracy:	720p live feed	
	Part Number:	LifeCam Cinema	
	Manufacturer:	Microsoft	

Table 4: Sensors

5.5. Payload Computers

Two computers were used for our robot, connected via an onboard Ethernet router. The main control computer is an Acer S3 ultrabook chosen for its light weight and good battery life. This is connected to two 7-port USB hubs for sensor interface. The vision processing computer is a Toshiba Qosmio gaming laptop, chosen for its quad-core I7 processor, 16GB of RAM, and 336 core GTX 670M graphics processing unit. The CUDA interface on the GPU is used to optimize the frame rate of Segfault's vision processing algorithms. Both computers feature solid state drives that will resist any residual vibration transmitted through the silicon vibration mounts.

5.6. Safety Systems

The vehicle power system was built with reliability and safety in mind. A solid state relay serves as the link between the batteries and motor drivers. When in its default OFF position, the drives are de-energized, making it impossible for the robot to move on its own. When the vehicle is on, the relay is held in its ON position by a dedicated microcontroller. An easily accessible pushbutton (Figure 10) is mounted three feet above the ground on the back of the robot, and a wireless radio link button can be controlled by an operator. When either of these devices is tripped, the signal to the relay is cut, whereby it reverts to its off state and severs the 24V main power lines. Operation of the robot is locked out until a manual rearm is performed by toggling the power button on the robot's chassis, ensuring the robot is unable to move until it is deemed safe by team members. This system is fail-safe; any failure in or power loss to the microcontroller will simply de-energize the relay and cut system power.

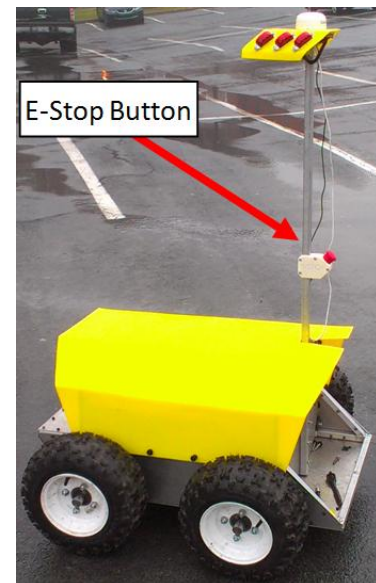


Figure 10: Vehicle E-Stop



Figure 11: Remote E-Stop

A set of 900MHz XBee Pro radios are used to form the transmitter-receiver component of the control system. Each XBee radio is shipped with a unique address identifier which has been utilized in combination with a unique network ID and full 128-bit AES encryption to ensure that our radios communicate only with each other. This ensures no other XBee radios present in the area are able to present a possible threat to our safety system.

Once the transmitter and receiver are energized and have connected, the transmitter regularly sends a small packet of data to the receiver. When the emergency stop button on the transmitter (Figure 11) is actuated, power is severed to its radio, and packet transmission halts. When the receiver misses a predetermined number of these packets the solid state contactor on the robot is turned OFF. This is a fail-safe design: for example, if the transmitter batteries die, the robot moves out of range, or the radio modems fail the robot will cease movement, ensuring we remain in control at all times. Once a "trip" condition is detected, our microcontroller on the robot prevents re-energizing of the motor drives until manually rearmed by a human.

The emergency stop pushbutton located on the robot's mast is in series with the microprocessor's control signal to the solid state contactor. When the button is actuated, the contacts are forced open, ensuring that even a fault in the electrical safety hardware may be overridden, halting Segfault.

6. Software Design

6.1. Localization

Segfault's localization stack was written in python and designed around the Robot Operating System (ROS). ROS provides standard operating system services such as hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. An overview of the localization system is shown in Figure 12. We feed encoder values, IMU, GPS and visual odometry from the camera into an Extended Kalman Filter (EKF) which gives us the robot's pose. Then the robot pose is fed into Adaptive Monte Carlo

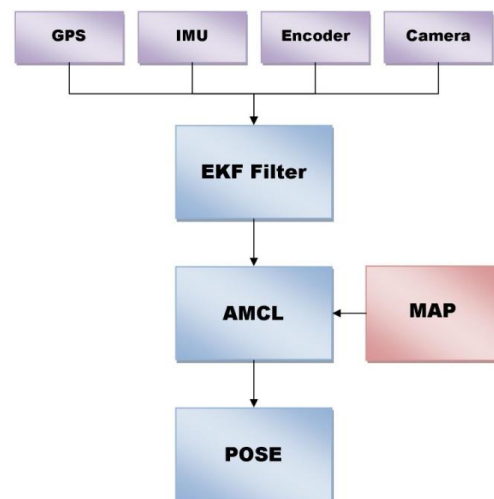


Figure 12: Localization Flow

Localization (AMCL) which uses particle filters to track the pose of the robot against a known map. The performance of the EKF is primarily dependent upon the covariance of its inputs. The GPS and encoders were given more importance for the translational movement whereas the IMU was given priority for detection of orientation. Visual odometry was given equal importance for translational as well rotational movement. Testing showed that the robot could consistently localize itself within 1.0m.

6.2. Vision Processing

Our image processing pipeline for line detection is depicted in Figure 13. OpenCV calibration functions are used to acquire the intrinsic camera properties, such as radial distortion. This generates a pixel mapping that can be used to correct for these factors, rendering the image closer to the pinhole camera ideal.

By placing a small collection of objects at known locations with respect to the camera, we obtain a sample of metric-to-pixel mappings which we can use to acquire precise estimates of the camera height and angle. This allows us to map the image into the ground plane, which we assume to be locally flat.

Mapping into ground coordinates allows us to easily combine camera data with the other sensors. Another advantage is that lanes now appear equally wide regardless of image location. This allows us to use specifically tuned edge detection filters. False positives may still occur on obstacles, but these will be filtered out using laser and stereo data. The lane point cloud is published as obstacles in ROS, visualized here using ROS rviz. This point cloud is fed into the mapping and navigation functions.

Additionally, vision is used to find flags. Stereo camera inputs are parsed for red or blue blob objects, and a disparity depth map is used to determine distance of flag from cameras. As seen in Figure 14, this algorithm was developed in cluttered indoor environments: the field will have fewer distractions leading to improved results.

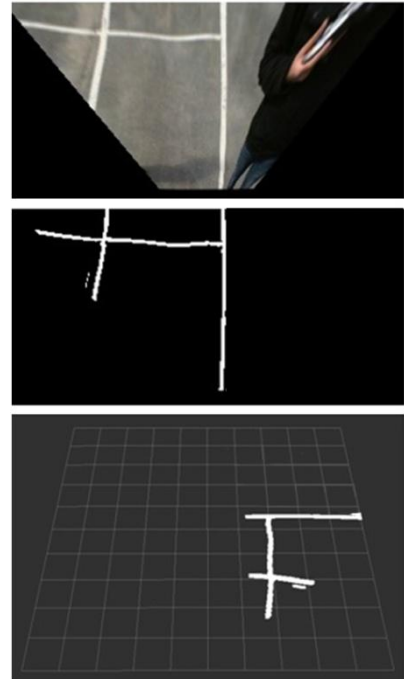


Figure 13: Lane Processing Flow



Figure 14: Flag Detection and Localization

6.3. Map Making

Segfault utilizes a map for producing the required paths for navigating through the track. This map is generated by a SLAM (Simultaneous Localization and Mapping) algorithm called GMapping which is already implemented within ROS. GMapping uses a particle filter to build a map, in real-time, of detected obstacles and walls. The algorithm inputs pose estimates generated by the localization EKF, laser scanner data, and point cloud data from the stereo vision system to map the track's chalk lines.

While not containing detailed information about small and moving obstacles, the map will have a general layout of the obstacle course and is dynamically corrected. The map is mainly concerned with localizing Segfault with respect to the GPS goals, the chalk lines and detectable static obstacles. A sample map made by GMapping is shown below in Figure 15.

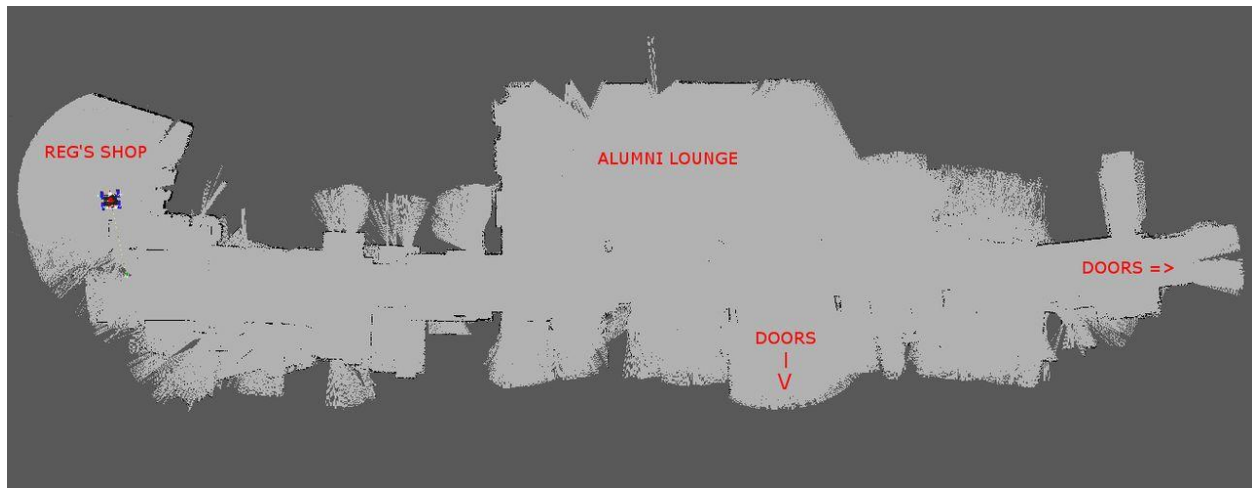


Figure 15: Map of the A,B,C,D Building Complex in Sexton Campus Generated using GMapping

6.4. Path Planning

In the Figure 16 we divide our path planning process into a global and local plan. The global planner takes a GPS goal and tries to find a path. We have used Dijkstra's algorithm to compute a path; it was chosen largely due to the ability to produce the shortest path to the goal efficiently. A* sometimes spent more time on searching and also requires more memory in comparison. In practice we found that A* results in sub-optimal paths whereas Dijkstra's is more likely to produce an optimal path to the goal for our sensors and robot control. The local planner for obstacle avoidance follows the global path and tries to avoid unknown obstacles by controlling the velocity of the robot.

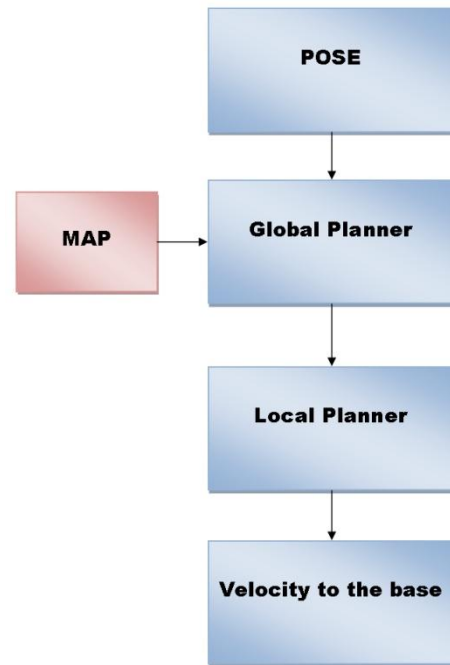


Figure 16: Path Planning Algorithm

6.5. Decision Making

Robot decision making is controlled by a series of modes.

The modes depend on the robot progress to date and its goals. In the primary robot mode, it travels through lanes avoiding white lines and obstacles to reach a goal location as determined by a GPS waypoint. The robot possesses secondary behaviours to account for less frequent situations:

- The robot detects nearby red or blue colored flags. Flag locations are used to allow the robot to drive with the red flags on the left and blue on the right per the course rules and objectives
- The robot detects inclined orientation and sets an “on-ramp” condition where it discards laser and visual sensor data made while the robot is at an angle while crossing the ramp.

6.6. JAUS

To parse and translate from the XML based messages passed by JAUS as defined in the JSIDL, we created a ROS node with JAUS++ integration. JAUS++ is an open source C++ library for JAUS integration and handles message encoding and decoding as well as discovery and handshaking for connection to the OCU. When position or velocity JAUS queries are received the appropriate ROS topic is polled and the reply is encoded in JSIDL by JAUS++. Received JAUS goals are published using a ROS ActionClient interfacing with the navigation stack, which already provides goal status and a reply is sent to the OCU upon completion.

7. Robot Analysis

7.1. Cost Analysis

As we had no prior hardware, Dal Robotics had to fund the construction of an entire vehicle. We deeply appreciate the many industrial sponsors that provided monetary donations, discounted products, and free services. A cost breakdown is provided in Table 5.

Component	Value	Cost	Comments
Frame/Body	\$626.96	\$626.96	Mariner Forge sponsorship
Motor Systems	\$1826.79	\$1826.79	
Power System	\$1038.37	\$1038.37	
Acer Laptop	\$700	\$692.40	
Qosmio Laptop	\$1837.57	\$1337.57	Toshiba sponsorship
Radios	\$184.81	\$184.81	
SICK Laser	\$6583.00	\$4937.00	Omnitech sponsorship
GPS	\$241.43	\$241.43	
Other Costs	\$3622.65	\$3622.65	Misc. Hardware
Total	16653.98	14507.98	

Table 5: Robot Cost Breakdown

7.2. Performance Analysis

Segfault's performance in key parameters is detailed in Table 6.

Parameter	Design	Tested	Units
Speed	3.0	2.4	m/s
Ramp	15	20	degrees
Battery Life	1	1	hours
Obstacle distance detect	10	18	m
GPS	2.0	1.5	m
Reaction time	5	10	Hz

Table 6: Robot Design and Tested Performance

8. Conclusion

The Dalhousie Robotics team is excited to join 2013 IGVC with our debut robot Segfault. We would like to thank our sponsors for their valuable contributions of funding and equipment, as well as the IGVC for hosting the competition.

