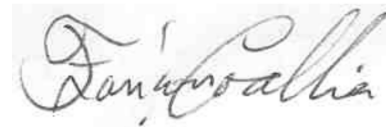


CAPRA6.2

École de technologie supérieure

Carolane Boulé, Benoit Côté-Jodoin, Philippe Delisle, Jérôme Gingras, Philippe Gorley, Iman Hassanein, Gabriel Kendressy, Quentin Lambert, Simon Landry-Pellerin, François Langelier, Thierry Maillot, Jean-Luc Montreuil, Amélie Paquette-Brisebois, Jean-René Roy, Jonathan St-Cyr, Yohan Trépanier Montpetit
François Coallier, francois.coallier@etsmtl.ca

I certify that the engineering design in the vehicle by the current student team has been significant and equivalent to what might be awarded credit in a senior design course.



Prof. François Coallier, Eng. Ph. D.
 Faculty Advisor, Capra
 École de Technologie supérieure (ÉTS)

ABSTRACT

This year the software development was based on a Test-Driven Development cycle (TDD) together with a DMAIC improvement cycle (Define, Measure, Analyze, Improve, and Control). During the design process a small group studied the various concepts, and choices were made democratically with the complete Capra organization. Since last year, real world and simulated testing was greatly improved. The result evaluations come from the sum of the test scores, with that score being weighted according to short-term and long-term objectives. One month before IGVC, Capra6.2 was able to complete the real world basic course and around 60% of the simulated advanced course.



Figure 1.
Innovation Icon

TEAM ORGANIZATION

The club was founded in 1999 by a group of students passionate about the world of robotics. Capra is a student scientific club that has as its main goal the design and implementation of autonomous ground vehicles. The team consists of engineering students from various different Bachelor Degrees and receives no help from any professors or research professionals, meaning the club is entirely directed by its members. Members of Capra work completely voluntarily, totaling to hundreds of hours without receiving any additional credits or compensation. They are students motivated uniquely by their passion to learn more practically, and accomplish in creating a product they are proud of.

Both captains of the team act as project managers. Projects are not imposed on the students, but instead they are encouraged to take on projects they feel they are best suited for. Each department discusses specific subject matter concerning their projects using a GoogleGroup, for example the process and execution of the robot's design. GitHub and Google Drive are used to centralize all the software resources.

During the departmental meetings, members discuss different feasible solutions for problems that have arisen. Each solution is then evaluated individually and the members express their doubts and concerns to then be reviewed point by point, the pros and cons, of the presented solutions. Following the discussion, decisions are taken democratically within the department to then be presented to the membership as a whole. Each department presents their decisions during general meetings, which are meetings of the entire membership to discuss problems arising within projects and solutions found to be implemented to ensure that everybody is aware of the decisions taken by each department and the overall impact it has on the overall product.

Mechanical design is done using Solidworks. Each idea, such as modifications or additions to the design, is discussed during the general meetings or on the club's mailing list within the Google group. Only a few people are tasked with the technical drawing of the vehicle during the conception phase, leaving the rest of the team time to concentrate on the construction of the drawn parts. Using a similar procedure to the mechanical department, the electrical department designs their circuits using Altium.

DESIGN STRATEGY

As any improvement process is based on the DMAIC improvement cycle (Define, Measure, Analyze, Improve, and Control), Capra is continuously using it to guide the innovations of the robot. The cycle restarts again following the tests; new objectives are defined, new measures are compiled, etc. During the design process, the software department define their needs as they relate to the design function, they present alternative measures that can be taken and ultimately the final decision is made by the membership. Capra applies the DMAIC in the following way.

Define

The first step of any design process is to define the needs to be fulfilled. Since the team participated in the 22nd edition of IGVC last year with a new robot, Capra6.2, the assessed needs and possibilities of improvement were clear. Specific objectives were defined by the needs to guide the team's work. Outlined below are examples of some of the global objectives within different DMAIC cycles of the same project, they have been organized into three distinct categories:

Reliability:

- Define the working scope of the design process until the next IGVC which is 90% software
- Use a methodology of test driven development (TDD) based on people's

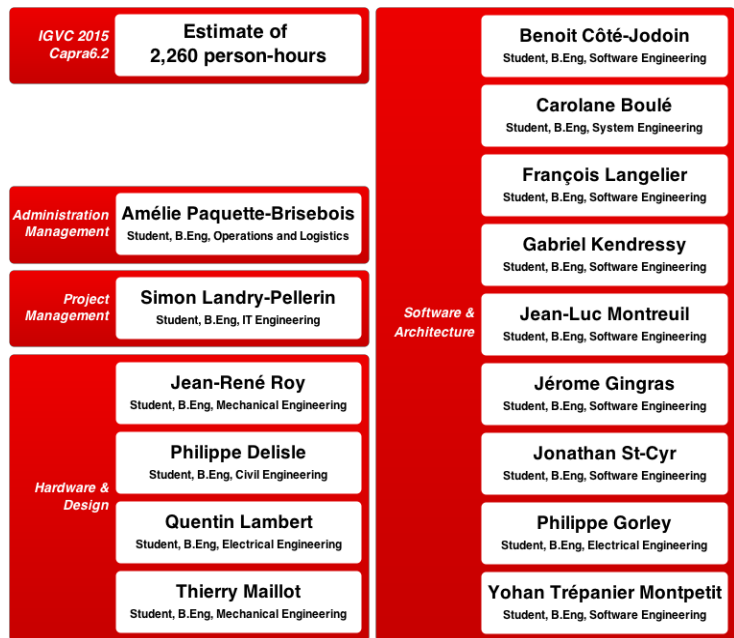


Figure 2. Team Organization

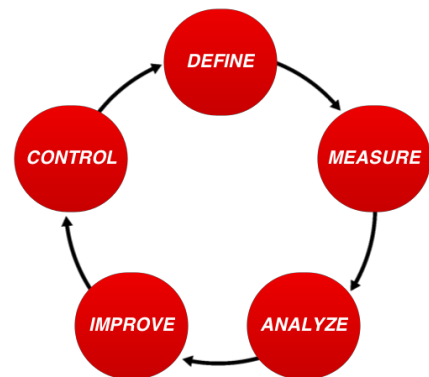


Figure 3. DMAIC Cycle

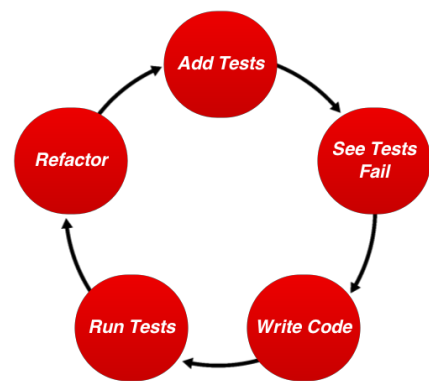


Figure 4. Test Drive Development Cycle

- experience and the data collected in previous years
- Make each component of the ROS architecture function independently by the end of 2014
- Make the whole simulation or the basic route function before march 2015
- Collect new data each weekend in new environments, similar to IGVC conditions, from March to June
- Use the collected data to conduct more precise tests
- Replicate the basic and advanced routes of the simulation environment as accurately as possible

Durability:

- Reuse the vehicle’s recordings as much as possible to avoid using the vehicle uselessly
- Eliminate structural weaknesses
- Conduct a complete check up of the vehicle after every two exterior texts
- Clean the vehicle after each test

Safety:

- Define a procedure for diagnosis of problems on the vehicle to avoid all risk of injury
- Facilitate access for maintenance
- A risk awareness session is given to each new member of the team and is mandatory

Table 1. Components Score

	Component	Score	Completion (%)
1	Camera	2	<u>1.00</u>
2	LIDAR	2	<u>1.00</u>
3	Encoders	2	<u>1.00</u>
4	Smart Motors	3	<u>1.00</u>
5	IMU	3	<u>1.00</u>
6	GPS	3	<u>1.00</u>
7	Extended Kalman Filter	20	<u>0.95</u>
8	Vision Processing	20	<u>0.90</u>
9	Mapping	30	<u>1.00</u>
10	AI / State Machine	35	<u>1.00</u>

Table 2. Courses Score

	Course	Score	Completion (%)
1	Basic (Simulation)	20	<u>1.00</u>
2	Advanced (Simulation)	50	<u>0.91</u>
3	Basic (Real World)	100	<u>0.70</u>
4	Advanced (Real World)	500	<u>0.15</u>

$$\sum_{i=1}^{NbComponents} (ComponentCompletion_i \cdot ComponentScore_i) \quad \sum_{i=1}^{NbCourses} (CourseCompletion_i \cdot CourseScore_i)$$

Figure 5. Components Scoring Formula

Figure 6. Courses Scoring Formula

Measure

Over the past year, the team has recorded measures, results, and statistical data of the vehicle’s performance. Methods to analyze the performances based on different aspects of the vehicle were already in place. This year, the team found a way to measure the software advancement based on the TDD principle. Each component and simulation of the route has a score, this score is based on the complexity, time of implementation, and the importance of the task for the realization of the team’s objectives.

The organizers of the test have to update the level of completion after each successful test. A completion of 1 (100%) represents a finished task. The total score of the project is calculated by adding the results of the two following formulas; the underlined numbers are simulated to better illustrate the formulas use. Thus, according to this table, the project would have a score of 402.5/790.

Analyze

These data sets were used to analyze the improvement or otherwise between the past iteration and the present, and accounts for future improvements to tweak achievable objectives. We also use this analysis to prioritize the short-term goals.

Improve

This is where changes were initiated. The team leaders encourage their team members to act and produce results. Experienced members taught the newer members and helped them to find and focus on the simplest and easiest solutions to issues. We found this year that the supervision offered by both DMAIC and TDD is making less experienced members more comfortable and more productive. Several little goals will always be easier to achieve than a big one.

Control

Small teams test their improvements, control any changes, record their tests, and update their scores.

CONCEPTUAL DESIGN

The architecture of Capra6.2 was separated into parts based on the functionalities needed by an autonomous vehicle to simplify the design process of the robot. Those sections are sensing, positioning, cognition, and mobility.

Sensing

The maximization of the field of perception, both for physical and visual obstacles, was the basis for the sensing strategy. The perception needs to be as reliable as possible so that the mapping is precise and accurate. It must also guarantee the safety of nearby bystanders because Capra6.2 is often used in public and interactive demonstrations. The solution is simple and efficient so as to meet the Auto-Nav challenge requirements.

Capra uses and modifies already existing ROS solutions, implementing only what is missing (e.g. a vision processing module specifically for IGVC), so as to not overcomplicate the design. The data format for the mapping uses ROS's point clouds. This setup enables the use of a standard data format for all sensors, which is in itself efficiently implemented in ROS.

The robot is designed with the various sensors in mind. For example, the body was made in such a way that the LIDAR has full access to its 270 degrees of range.

Positioning

Since the mapping depends heavily on positioning, the latter must be extremely reliable, precise and accurate. To assure that these aforementioned requirements are met, data is gathered as often as possible and passed through an Extended Kalman Filter. The use of complex components, such as an inertial measurement unit (IMU) and a global positioning system (GPS), requires strict operating procedures to both reduce the failure modes caused by human operation, and optimize their precision.

The ROS community has greatly simplified the implementation process of a positioning system; many subsystems are ready-made and publicly available. Few modifications are required to get a working positioning system up and running. The Capra robot has two motorized wheels along its center so that it can turn around on itself without skidding on the grass. The IMU must be placed outside to avoid electromagnetic interference.

Cognition

The development of a cognition system is based upon a simulated environment and tested in the field. This year, the robot's artificial intelligence (AI) is simple, stable and modular. Its flexibility is guaranteed by the incorporation of a state machine module to control which AI strategy to use when, along side the implementation of a multi-layered subsystem for the AI (detection, mapping and path planning). This offers the desired stability and simplicity. The division of the cognition system renders it more durable as depreciated modules can simply be removed or replaced.

Mobility

Security is the main concern of the mobility system; even if the robot can reach a top speed of 4.4 miles per hour (1.97 m/s), it has a hardwired limit of 2.2 mph (1 m/s), thus the reaction time is around 68 ms. The use of smart motors guarantees

a quick and detailed diagnostic of any encountered failure modes, either during testing or a competition. Only two motors are used to reduce the cumulative error from the encoders, this also benefits the path-planning module as the robot can turn on itself. The batteries have a long life, rendering them extremely ideal for tests and official events. The robot's body is as small as possible, which reduces the strain on the batteries and maximizes error tolerance.



Figure 7. Hardware Conceptual Design

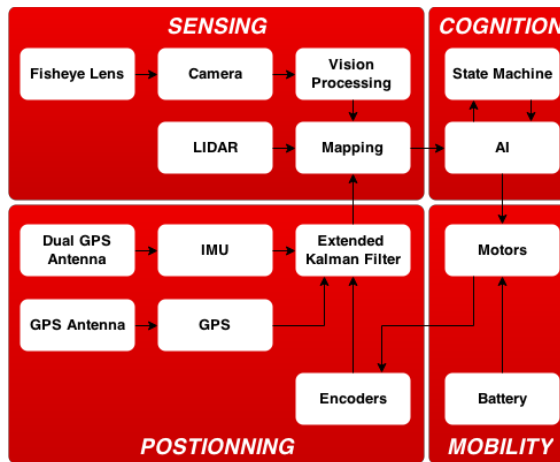


Figure 8. Software Conceptual Design

COMPONENTS OVERVIEW

Inertial Measurement Unit (IMU)

The VN300 from VectorNav Technologies uses a GPS antenna and an integrated Kalman filter to estimate the optimal positions, velocity and orientation. The use of the GPS allows us to obtain reliable measures without having to rely on the robot's dynamic or magnetic sensors. We choose this IMU for its reliability and durability. Measurement includes: 3-axis Accelerometer, 3-axis Gyroscope, 3-axis Magnetometer, Barometric Pressure and two 50-channel u-blox GPS L1 C/A GPS receivers.

Rotary Incremental Encoders With Index

Two 1000 CPR Optical Encoders from US Digital precisely measure the relative position and the velocity of the two motorized wheels of the vehicle. Those encoders are directly placed on the engine's shaft, in front of the gearboxes that have a 20:1 ratio; increasing by twenty times their precision. Each encoder has a precision of more or less 0.018 degrees, which corresponds to 0.0019 inches (0.048 mm) on the tires.

Global Positioning System (GPS)

One ProPak-V3 with 702-GG antenna from NovAtel Inc. is placed in the center of the robot. One hundred times per second, the GPS sends 3.94 inch (100 mm) precise measures with Omnistar HP. This device provides superior multi-path rejection close to the antenna, and in high multi-path environments.

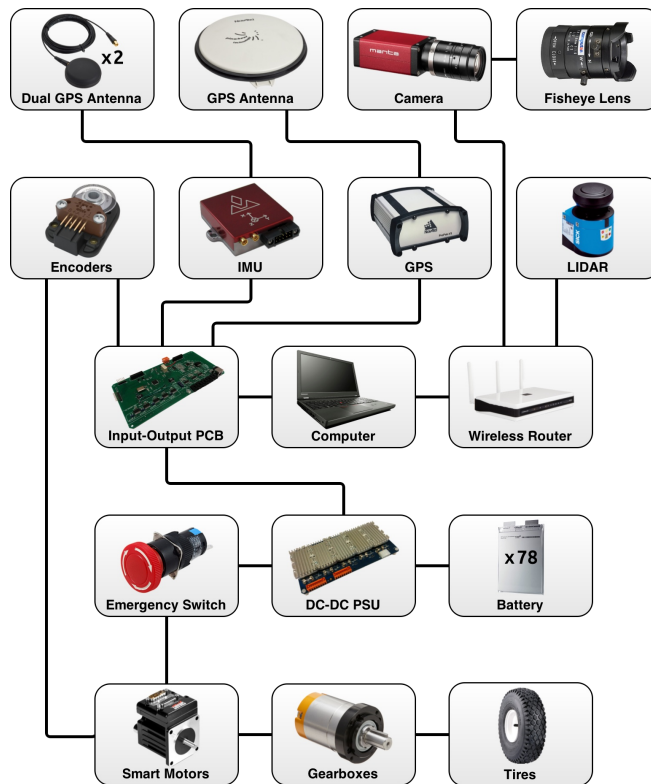


Figure 9. Components Overview

Light Detection And Ranging (LIDAR)

One *LMS100* from *SICK Group* measuring at 270 degrees, with a precision of 1.77 inches (45 mm), at more than 59 feet (18 m), at 50 Hz. The measures are taken between 1.64 and 65.62 feet (0.5 and 20 m) with an angular resolution of 0.5 degrees. Ultimately this LIDAR was chosen for its durability and reliability.

Camera & Fisheye Lens

The *Manta G-095C* from *Allied Vision* with *Fisheye Lens T2Z1816CS* from *Computar* performs considerably well, and its visibility is very impressive. It is able to provide a resolution of 1292×734 pixels at 40 Hz with a very sophisticated color correction. The fisheye lens of the camera lets us see with a field of view (FOV) of 160 degrees. It is possible to change the focus manually in order to test different strategies.

Computer

A *W540* from *Lenovo* was chosen for its durability and reliability. We have one backup in the case of a failure. Softwares are constantly synchronized between the two computers. Computer specifications are described in Table 3.

Table 3. Computer Specifications

Parameter	Value
CPU	Intel(R) Core(TM) i7-4700MQ CPU @ 2.40 GHz
RAM	16 GB @ 1600 MHz
SSD	Vertex 4 (64 GB)
SSD (Max Read)	460 MB/s
SSD (Max Write)	220 MB/s
Graphic Card	NVIDIA Quadro K1100M
Graphic Cores	384 @ 716 Mhz
Graphic Card Memory	2 GB DDR5 @ 2800 Mhz

Smart Motors

Two *SM23165D* from *Animatics Corporation* with *SL1500 20:1 Gearbox* from *Parker* are in the center of the robot in order to let the robot turn at a 0 degree turning radius. Only two engines in the center of the robot are used to avoid slipping on the grass when the robot turns. This kind of action adds an important error risk to the values of the encoders. *Capra6.2* has two motorized wheels at the front, two rear swivel wheels, and an independent suspension system. The suspension system is made of four pneumatic shocks not directly fixed to the wheels, but instead installed on a suspension arm fixed to the wheels. It makes the robot extremely dependable even where the terrain is bumpy and unequal. See Table 4 for detailed specifications.

Table 4. Smart Motors Specification

Parameter	Value
Continuous Torque	74 oz-in (0.52 Nm)
Peak Torque	118 oz-in (0.84 Nm)
Nominal Continuous Power	204 Watts
No Load Speed	5,200 RPM
Tires diameter	12 inches (304.8 mm)
Gearbox ratio	20:1

Battery

A battery system of 78 4.2V 5Ah LiFePo4 cells from *Amita Technologies Inc.* there are 6 of these cells connected in series to form a single battery, leading to a total of 13 batteries. Each of the 13 batteries are plugged in parallel for a total supply of 54.6 V. The battery system was completely assembled by the *Capra* team.

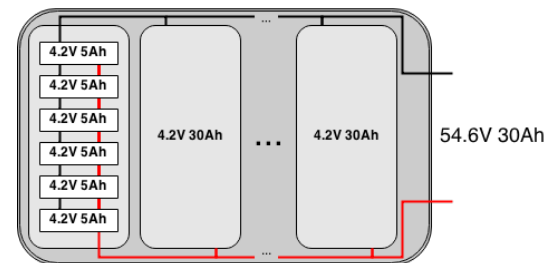


Figure 10. Battery Schematic

Input-Output PCB & DC-DC PSU

Both are custom made and are described in the Electric Design section.

ELECTRIC DESIGN

Power Distribution

The battery is directly connected to the motors and the electrical box where inside the power lines are connected to a

power supply. The power supply has been designed by Capra’s team and is based on the Vicor Corp micro DC-DC converter. The power supply produces different voltage rails to supply each sensor; generating 5Vdc, 9Vdc, 12Vdc, 19.5vdc and 24vdc. Each rail is fused at the input and the output.

The power supply is connected to the control panel through a power cable, and to the other PCBs through a mezzanine card connector. The control panel splits the power line to every sensor/actuator. The control panel allows the user to disable/enable any sensor/actuator individually either manually with a switch or using a software command. A power-on LED indicates the state of each sensor/actuator.

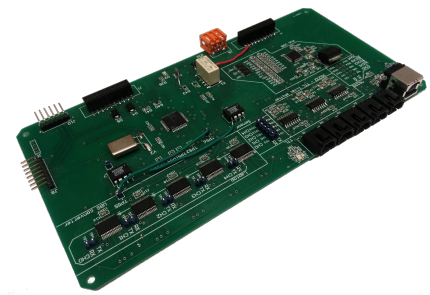


Figure 11. Input-output PCB

Control Panel

The control panel acts as the user entry point to the electrical system of Capra6.2. It provides manual and software control for the power distribution. It also monitors information such as battery voltage, and vehicle current serial port to the mezzanine card connector. Its consumption, internal temperature, problem indicator, and individual sensors power-on LED. Finally, an ATC fuse (automotive model) for each power rail is present on the control panel.

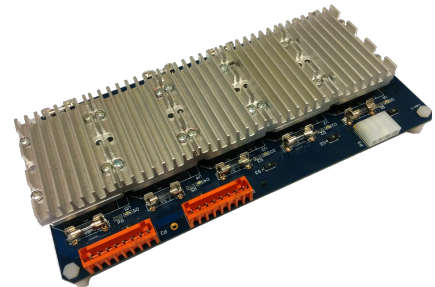


Figure 12. DC-DC PSU

This circuit board is a four layer PCB based on a PIC24FV32KA304 microcontroller; it has been completely produced by the Capra team. The use of such a circuit considerably simplifies the maintenance of the vehicle as a result of the easy-connect configuration and the monitoring features. Each device has a unique connector that avoids human errors and results in clean wiring.

Input-output PCB & DC-DC PSU

Capra’s electrical team has designed a printed circuit board that allows easy scaling of the system. This circuit has two main safety related features: it provides five RS232 ports over one USB cable, and it allows input-output PCB to take control over a RS232 line as a middle man.

SYSTEM INTEGRATION

System integration is what makes Capra6.2 a robot functioning as a whole rather than many modules working individually. The centerpiece of Capra6.2’s software system integration is the ROS architecture that manages the communication between all the nodes of the system. The software team used this integration to connect all the sub-systems together, be they pre-existing nodes within ROS or new ones designed by them.

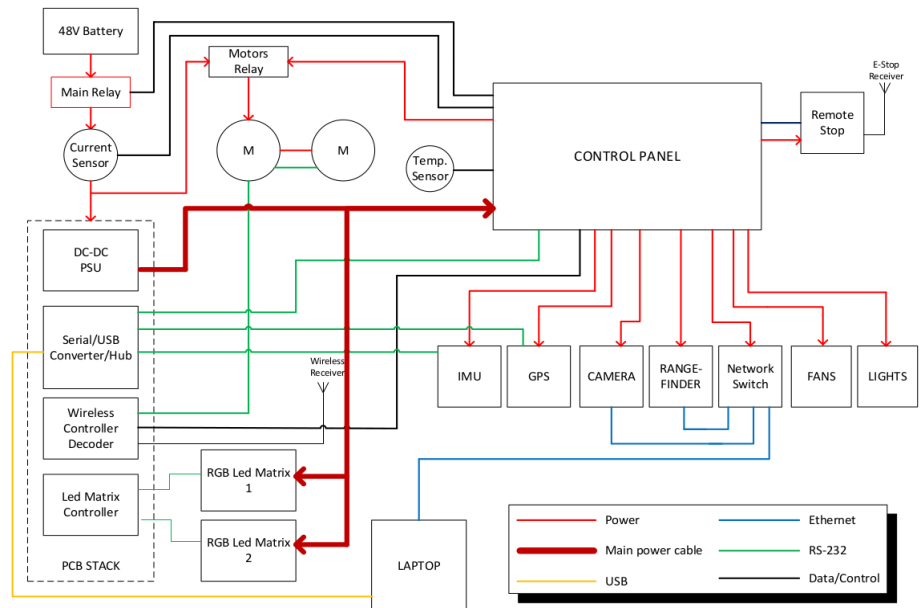


Figure 13. Electrical System

The node controlling the Animatics SmartMotor is a new part of ROS added by Capra this year. It is a conversion of pre-

existing Java code to Python for an optimized integration with ROS, and is able to send velocity commands to the motors while simultaneously reading the encoders' position. The team also developed various new nodes that add functionalities and interact with different parts of pre-existing ones.

Software Architecture

Capra's software architecture is built on top of the Robot Operating System (ROS). It provides a reliable communication model between all subsystems, an efficient distributed architecture, and an extensive set of debugging tools to build a software platform. Subsystems are divided into nodes, which can be developed independently one from another and provide a perfect integration with the team's DMAIC processes. Every device on the robot has its own set of nodes that are independent from each other. This makes the code easier to maintain as each node acts as a software that receives and publishes messages (i.e.: raw sensor data, the robot position, etc.) to and from other nodes.

Robot Positioning

Great improvements were made when it comes to position the robot in a virtual environment. With the help of an extended Kalman filter, from the robot_pose_ekf ROS package, the robot positioning subsystem can now accurately determine the robot's position. The data from the wheel encoders, the Novatel ProPak v3 GPS, and the VectorNav VN-300 IMU, are merged in the filter and the linear quadratic estimation algorithm recursively calculates a statistically optimal estimate of the robot's position. The 0.5-degree accuracy of the IMU, together with the 5 cm precision of the GPS receiver and the 0.02 degree precision of the rotary encoders allow an incredibly precise estimation. With the Kalman filter, the robot's position will always drift slowly towards the measurements and will attenuate the effect of erroneous data or bad sensor calibration.

Waypoint Navigation

GPS Waypoints are loaded and converted to Cartesian coordinates as soon as Capra6.2 is launched. The current position of the vehicle is accounted for and considered as the origin of the grid in which it will move. Since the GPS receiver's accuracy is very high, we can trust that converting the coordinates only once, at the start, will yield reliable results. Trials in a real-world environment confirmed this theory. One by one, the Cartesian waypoints will be sent to the robot and it then will calculate a plan to reach them according to strategies described in the Software Strategy section.

COGNITION

Obstacle Detection

Obstacles are detected with our Sick LMS100 LIDAR. The physical layout of the robot allows the sensor to detect obstacles within 270 degrees around the robot at 50 frames per second, with a precision of 0.5 degrees that is even able to detect the poles of the flags of the auto-nav challenge. The raw data is filtered to remove optical distortions and sent to the mapping subsystem.

Lane & Flag Detection

In the last few years, in association with other student clubs at École de Technologie Supérieure, Club Capra was able to develop a common vision server named SeaGoatVision. This year, SeaGoatVision was encapsulated in a ROS node to enable perfect compatibility with the rest of the system.

The camera output is up to 40 images per second, with a 720p resolution. With the help of a fisheye lens, the camera has a field of view of 160 degrees, and the robot can easily compete against other robots that use two cameras. Using a single camera makes the vision system simpler since there is no need for hardware or software synchronization between multiple optical devices.

The goal set by the team this year was to be able to have an artificial intelligence that could take decisions at a rate of 20 Hz. In order to achieve this goal, the vision software had to be reworked entirely. With the help of multiple filters to sanitize the output from the camera, the line detection software can now output lines at a rate of 20 Hz. Besides the optimization of the filters, the execution of the line detection algorithm was moved to the GPU in order to let the AI use as much of the CPU as possible.

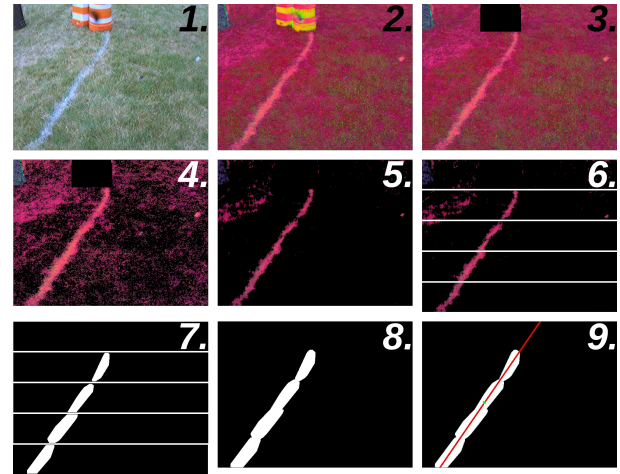


Figure 14. Image Processing

To detect the lines and the flags, SeaGoatVision uses a set of filters based on OpenCV algorithms to perform the following actions:

- Resize the image to maximize performance Figure 14. Image processing
- Remove the fisheye distortion of the lens
- Apply a mask to hide the robot from the camera image
- Apply color filtering
 - Detection of white lines
 - Detection of flags
- Adjust perspective to adequately position the elements in the 3D environment
- Publish the images

To detect the flags, we simply apply a HSV color threshold to the image since the colors of the flag (red and blue) have hue values that are quite different from the green grass.

The first step of the vision strategy is to convert the image in HSV; some obstacles are then removed using their typical orange colors to detect them. A filter removes the grass from the image using predefined hue parameters to output a binary black and white image. Some particle of grass may still be visible, so a particle filter is applied to remove the smallest bits.

The next step is to split the image in 5 rows. A convex hull is applied on each of these rows to make the detected grass areas bigger. Finally, the rows are put back together and a dilate function is applied to connect them.

This process allows Capra6.2 to accurately detect lines. The filter must be calibrated if the lighting changes significantly, but otherwise this system is quite robust in a stable environment.

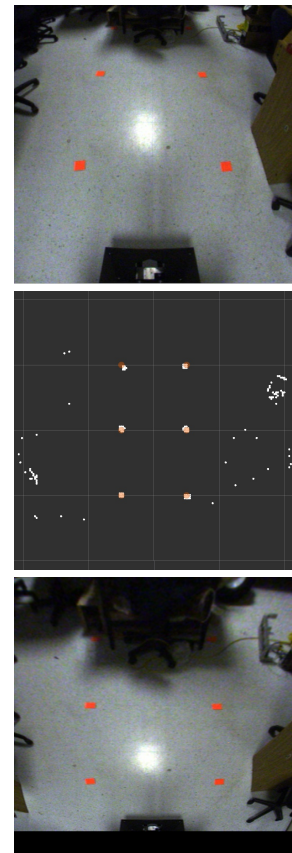


Figure 15. Image Processing

Vision Mapping

To position the various elements detected through an optical device in the robot's virtual environment, the team implemented a new calibration system. The idea was to design a system that would be simple to use and calibrate. Thus was developed a ROS node that shows the camera image to the user and allows for dynamic modification of the parameters of a



perspective transform to align real-world points with virtual ones. Figure 15 shows the result of this process where a user has aligned 6 real-world points (white) with their virtual equivalent (orange). The parameters of the transformations are then saved and the filter is applied to any input image.

The whole process is very simple to do, taking from 2 to 3 minutes, and the filter only has to be recalibrated if the orientation of the camera changes.

Mapping



This year, Capra overhauled the way its robot maps its environment now using the probabilistic 3D mapping library Octomap, based on the octree data structure.



To navigate through its environment, Capra6.2 uses its sensors to create an accurate map of its surroundings. The map generation is based on the ROS package `move_base`. Two different maps are used; the first map, called the local map, is built with instantaneous data from the sensors and the persistence of these data is very low. Almost as soon as the sensors get updated, the map is refreshed to contain only the most up-to-date information. This allows the vehicle to plan a precise short-term plan meanwhile reliably avoiding obstacles using the up-to-date data.



The disadvantage with the local map is that it does not take into account the position of the robot and is therefore useless for long-term uses. This is why a second map, called the global map, is generated as the robot moves along its environment. The map is built by taking into account the current position and orientation of the robot and the currently detected obstacles and lines. The data is added to the map using the probabilistic model of Octomap. The model accounts for sensor noise and the map is dynamically resized as the robot explores its surroundings. The data structure allows the map to be stored efficiently and reused for a later run. .

All the data uses point cloud formats and, even though the robot currently only supports 2D sensors, the maps could easily be transformed to 3D representations of the environment.

Both maps apply inflation to its input data. The inflation varies according to the type of the obstacle, the type of the map, and follows the following rules (table 5). By using that strategy, the robot plans its global path to avoid any obstacle or line, while simultaneously planning its local path by considering it can pass quite close to lines. The difference between the line and physical obstacle inflations allows the robot to avoid obstacles at all cost while keeping close to the lines. This optimization was designed to increase the performance for the IGVC challenges, since the robot may touch lines, but not obstacles..

Path Planning

To generate its path, Capra6.2 uses 2 strategies, basing itself on the `move_base` ROS package Figure 17 It starts by generating a global plan using the global map built with high inflation values. This plans uses a dijkstra algorithm to find the shortest path to the next waypoint. The path passes relatively far from any currently known obstacle, line, flag, or pothole

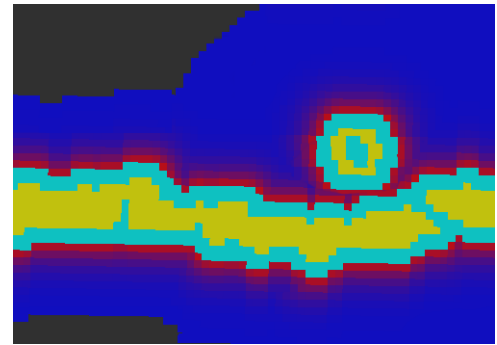


Figure 16. Inflated Map

Table 5. Obstacles Inflation

	Line	Value
Global map	High inflation	Very high inflation
Local map	Very low inflation	Low inflation

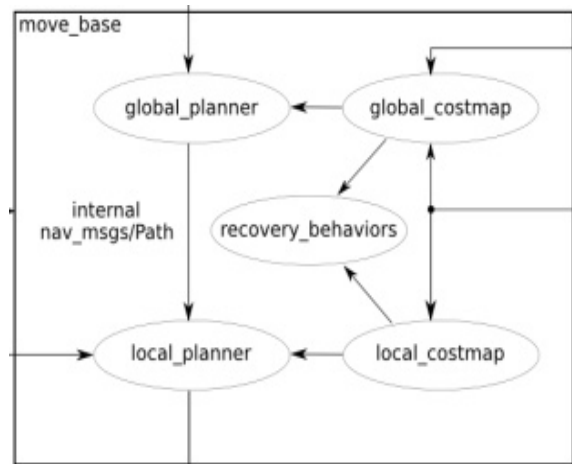


Figure 17. `move_base` ROS Package

and is recalculated every 2 seconds to account for newly discovered obstacles. The second step is to generate a local path that will calculate the velocities for the next 1.5 seconds and is recalculated at a frequency of 20 Hz. The local plan is configured with three heuristics (named in order of importance):

1. Avoid obstacles
2. Follow the global plan
3. Reach the goal

The obstacle avoidance is based on instantaneous data from the sensors instead of on a pre-generated map to achieve very robust results. During the process, if the robot gets stuck and is unable to move, it starts cycling through its recovery behaviors, pivoting on itself until it finds a way out.

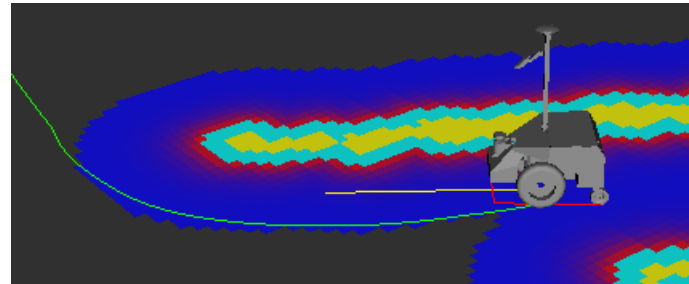


Figure 18. Path Planning

Figure 18 shows both plans as the vehicle plans its way through an obstacle course. The green line represents the global path while the short yellow line represents the local path.

Lane Following

The strategy to follow the lanes of the IGVC Auto-Nav challenge is the result of the process described in the previous sections. As white lines are detected, they are added to the local maps, the global map, and the global path. This allows the robot to plan a path to the next waypoint every 2 seconds, allowing the local planner to send velocity commands in such a way as to avoid obstacles and lines; eventually leading to the objective.

To be able to follow lanes with dashed lines, the robot will extend the detected lines to cover the gaps between them if they end abruptly. The generated path will then automatically circle around them.

SOFTWARE STRATEGY

The artificial intelligence subsystem of the robot is based on a state machine that manages the path planning algorithms, the waypoint navigation parameters and accuracy. Figure 19 describes how this state machine evolves as the robot progresses in the course.

The parameters are adapted to optimize the robot's behavior and decision-making process. The precision with which the robot will attempt to reach waypoints is dynamically adjusted to reflect the importance of the waypoint (IGVC vs generated waypoint from the previous run). The speed of the vehicle is also modified since the no man's land generally requires less precise

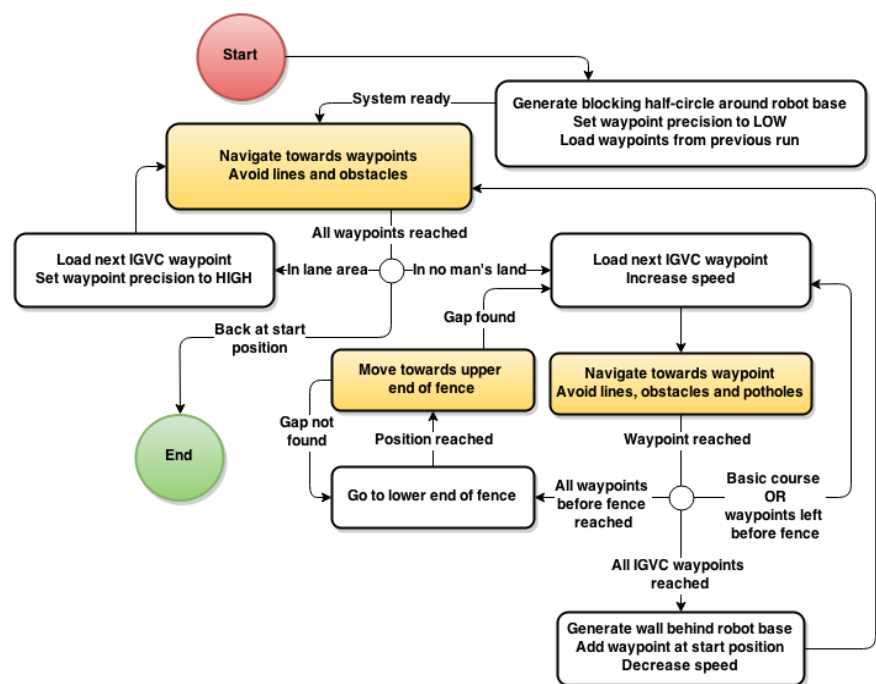


Figure 19. Decision-Making

movements. At the start of the run and when the last IGVC waypoint is reached, virtual obstacles are automatically generated to help guide the robot. A half-circle will force it to start by going in the right direction and a simulated wall will close the way as the robot reaches the last IGVC waypoint, effectively forcing it to come back to the starting point through the lane.

The yellow states represent states where the robot uses the path planning algorithms to reach a goal. This is where the cognitive process takes place, iteratively calculating a path to reach the current waypoint at a rate of 20 Hz.

For high speed operations, various parameters are adjusted in the following way:

1. Speed is increased
2. Obstacle detection range is increased
3. Waypoint precision is slightly reduced
4. Camera angle is adapted to see further
5. The resolution of the input camera image is slightly reduced to allow a quicker analysis
6. Path planning parameters are adjusted to avoid dangerous mechanical behaviors

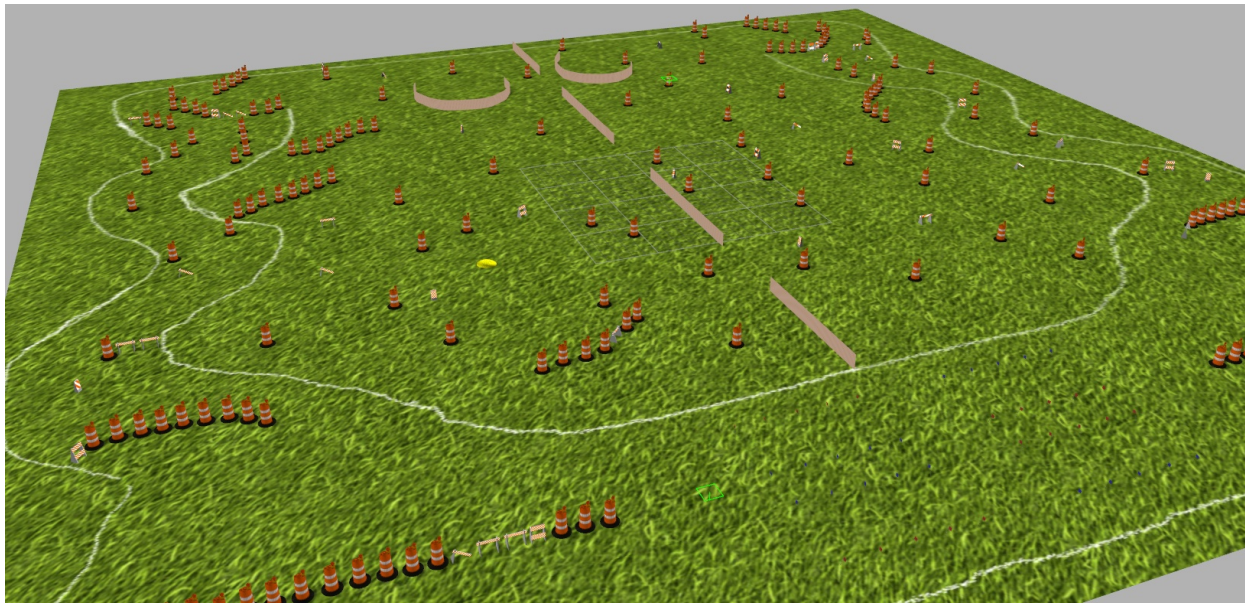


Figure 20. Simulated Advanced Course

Simulations



To test software strategies and analyze the decision-making process of Capra6.2, the team decided to use the ROS simulator, Gazebo, and configure it to accurately replicate the sensor data perceived by the robot in a competition environment. Two complete courses were designed, for the basic and the advanced challenges, and include lanes, obstacles, waypoints, potholes and even alternating fence openings, see Figure 20.

Debugging

To efficiently debug the robotic platform, the Capra team developed and integrated various tools:

- The ROS visualization tool, Rviz, is used to inspect and analyze the robot. Many tools were developed to allow an extensive use of Rviz to debug any aspect of the robot, including sensor data, maps, decision plans, waypoints, etc.
- The distributed architecture of the systems allows any team member to connect remotely to the robot using its internal router to run diagnostic tests or analyze runtime data.
- The control panel and LED panels show information about the current state of the robot without requiring any

external software.

FAILURE POINTS IDENTIFICATION & RESOLUTION METHODS

Table 5. Obstacles Inflation

Failure Point	Failure Mode
Camera	no power, no output, output refresh rate too low, output too bright, output too dark, output field of view too small, output colors not optimal for detection
LIDAR	no power, no output, always detects an obstacle
GPS	no power, no output, output refresh rate too low, output precision too low
IMU	no power, no output, erroneous output
Encoder	no power, no output, erroneous output
Motor	no power, not rotating, rotating too fast, rotating too slow, fails to rotate at a prescribed time, fails to cease rotating at a prescribed time
Computer	no power, not booting correctly, no hardware inputs

During the competition, if there is no power to a component, the cables, the DC-DC PSU and the battery will be checked, in that order. Once the defective component is identified, it is replaced by a backup. If the component is not transmitting information, the connection between it and the software, the physical connection, the cables, the input/output card, and the software configuration are checked in that order. If needed, the components or configurations are changed.

If the camera emits too bright or too dark of an image, or if the field of view is too small, the iris or the focus is manually changed to suit the situation. If the camera emits an image with non-optimal colors for the detection of obstacles, the color correction, exposure, and white balance are tweaked in the camera's configuration.

If the LIDAR always detects an obstacle, no matter its position, its line of sight will be checked for obstacles (e.g. part of the robot), its position is also verified as the sensor must be parallel to the ground.

If the GPS is lacking precision or accuracy, the connection to the OmniSTAR service will be verified first, then the filtering of data, and finally its configuration. If needed, the filters will be adjusted, the configuration changed, or the data refresh rate lowered. If the latter is too slow, the configuration will be tweaked or the GPS will be switched with its backup.

If the IMU emits erroneous data, the antenna's connection will be checked, the data filtering will be verified to see if it's coherent, verification that no external interference might distort the data, and then the configuration would be reviewed. If needed, the filtering or the configuration can be adjusted.

If an encoder is emitting false information, a well-established connection to the laptop will be checked and checks will be made to all other components to assure the error is not being cause by them. If needed, the encoder will be switched with a spare.

If a smart motor is not spinning, the logs will be verified to see that the motors are supplied with current. If they are spinning too fast or too slowly, the logs and encoders will be checked. The communication between the motors and laptop will be checked to see if it is stable. If the motors fail to rotate or fail to stop at a prescribed time, the logs will be checked, as well as the communication. In all cases, an attempt to recreate the problem off course will be made. More often than not, the logs give information as to the next step to follow in order to solve the problem

If the laptop does not boot properly, the hard drive may need to be replaced by a backup drive which has recently been updated with a functional copy of the system, and as always the logs must be checked. If the laptop does not receive data from one or more hardware components, the input/output card's USB connection to the laptop will be checked, the communication with the camera and range finder will be tested, and the system logs and electronic box will also be

checked. The worst-case scenario, the laptop itself will be switched with another that has a working copy of the system.

A more exhaustive list of procedures than those listed in this report is available. It was built quickly thanks to multiple outside tests that are very well documented. This tool allows for rapid diagnostics and problem solving, whether the problem is software-related or hardware-related. In addition, each component has its driver in the form of an ROS node. To make problem solving quicker during a failure, the diagnostic is made on these nodes.

There is at least one backup for each of Capra6.2's components. Only a few do not have one for budgetary reasons. Strict procedures are in place to reduce the risk of breakage or failure, for example during transport and maintenance. These components are the Fisheye Lens, the LIDAR, the IMU and the Dual GPS Antenna. The latter are expensive because they have excellent durability and reliability.

OVERALL SYSTEM PERFORMANCE & COST ESTIMATION

Table 6. Overall System Performance

Parameter	Theoretical	Trial Data
Top speed	4.4 mph	~3.35 mph
Ramp vlimbing ability (15° slope)	3.8 mph	~2.79 mph
Reaction time	50 ms	~68 ms
Battery life	5 hrs	~4 hrs
Physical obstacles detection range	20m @ 270°	~20m @ 270°
Visual obstacles detection range	10m @ 160°	~10m @ 160°
Effectiveness in dealing with switchbacks	99%	~99%
Effectiveness in dealing with center islands	99%	~98%
Effectiveness in dealing with deadends	90%	~86%
Effectiveness in dealing with traps	99%	~95%
Effectiveness in dealing with potholes	90%	~83%

Table 7. Cost Estimation

Component	Retail price (USD)	Cost to team for 2014-2015 (USD)
Building materials	3,000.00	0.00
Batteries	2,500.00	0.00
Wheels & Tires	350.00	0.00
Smart Motors	5,500.00	0.00
LIDAR	2,700.00	0.00
GPS	20,000.00	0.00
IMU	5,000.00	0.00
Camera	1,000.00	0.00
Electrical Compoents	2,500.00	0.00
Computers	5,400.00	0.00
TOTAL	47,950.00	0.00