Lawrence Technological University

WASPP
5/15/2018

Team Lead

Lacy Pyrzynski lpyrzynsk@ltu.edu

Team Members

Syed Athar sathar@ltu.edu
Ryan Pizzirusso rpizzirus@ltu.edu
Chris Suchezky csuchezky@ltu.edu
Thomas Weeks tweeks@ltu.edu

"I certify that the design and engineering of WASPP by the 2017/2018 Lawrence Technological University Robotics Team has been significant and equivalent to what might be awarded credit in a senior design course".

Giscard Kfoury, PhD
Program Director
Bachelor of Science in Robotics Engineering
Email: gkfoury@ltu.edu

1. Introduction

For the past four years, seniors participating in the Blue Devil Motorsports Robotics program at Lawrence Technological University (LTU) have built an autonomous vehicle from the ground up to compete at the Intelligent Ground Vehicle Competition. The 2018 team design goals were to create a smaller, more maneuverable, accurate, and reliable competition vehicle than years past. The 2018 vehicle utilizes a 3D stereo vision camera and a series of ultrasonic sensors to perform the vehicles path planning, lane keeping, and object avoidance tasks. In addition, the vehicle also uses a compass and GPS module to aid in waypoint navigation. The vehicle was designed to be weather resistant and aesthetically pleasing. The 2018 vehicle, the WASPP, was named using the first letter of each team member's last name. See the team's mission statement and vision below.

Mission

The 2018 IGVC team's mission to build a vehicle to meet all technical specifications and autonomously perform lane detection, obstacle avoidance, and GPS waypoints tasks, in order to, successfully navigate a given obstacle course.

Vision Statement

Create a wealth of knowledge in autonomous vehicle systems and components used to thrive at competition in both vehicle performance, presentation, and written reports.

2. Organization

Each senior team member was given an overall vehicle system to manage. Through the design review process, team members presented their research findings, benchmarking, potential component selections, and system recommendations to the team. After thorough team discussion all components, vehicle design, and systems were decided on as a team. Execution of each vehicle system was led by its responsible team member. Due to the small size of the team all members worked in a cross functional manner to accomplish vehicle goals. See Table 1 below for the team's organizational chart and hours spent on the project.

Table 1: LTU Team Member Organization and Estimated Hours Spent on the 2018 IGVC Vehicle

IGVC Team Members				
Name	Role	Standing	Major	Estimated Hours
Lacy Pyrzynski	Team Lead	Senior	Mechanical	216
Syed Athar	Safety Lead	Senior	Robotics	170
Ryan Pizzirusso	Software Team Lead	Senior	Robotics	188
Chris Suchezky	Mechanical Team Lead	Senior	Mechanical & Robotics	225
Tom Weeks	Electrical Team Lead	Senior	Robotics	195

3. Vehicle Design

3.1. Design Goals

The team had several design goals, which contributed to the vehicles overall concept. The team wanted to create a highly maneuverable vehicle for increased agility on the course while performing obstacle avoidance tasks. Last year's team experienced LIDAR failure at the competition, which lead this year's team goal to incorporate a backup system in case of main environmental detection failure. The overall vehicle design goal was to fabricate a vehicle and

outfit it with components within a $3,000 budget. To meet the budget goal the team focused on making a sustainable vehicle that utilized many recycled components. For example, the team used reclaimed wood for the electronic enclosure, tires were reused from various unused LTU robots, the power distribution board, wireless emergency stop components, and batteries were all recycled from the previous IGVC team. In an effort to make the vehicle more sustainable, the team choose only 12V components to eliminate the need for 24V energy usage onboard. The team made the assumptions that they would be capable of raising the entire $3,000 needed for the vehicle and that all of the recycled components would last through the duration of the competition.

3.2. Design Process

Each vehicle system and component underwent a design review or in the case of the vehicle frame and environmental detection system, a series of design reviews. The team member responsible for the system or component first conducted independent research and benchmarking. A number of potential options or solutions were then selected and presented to the entire team. Team members facilitated a discussion until a consensus was reached to either move forward, continue researching other options, or make changes to the system design. As the vehicle was fabricated and sensors were tested there was an ongoing design process that the team adapted to in order to make changes based on actual system or component performance. This design process was less formal than the prior design reviews and resembled group discussion or trial and error testing methods.

4. Innovations

The vehicles environment detection system incorporates innovative technology applied to this year's vehicle. Previous LTU teams used a LIDAR system for obstacle avoidance and web cameras for lane keeping. This year the team invested in an innovative 3D stereo vision ZED camera to perform both obstacle avoidance and lane keeping tasks. Benchmarking of the 2017 IGVC competition vehicles that placed in the Auto-Nav competition showed that the few teams that utilized the new stereo vision cameras were successful at competition. In addition to the stereo vision camera, the team has added five ultrasonic sensors, three in the front and one on each side to serve as a redundancy object avoidance system. The ultrasonic sensors provide object detection outside of the cameras range, mainly, objects that are too close to the vehicle that the camera may have missed. The combination of the ZED camera and ultrasonic sensors comprise the innovative environment detection vehicle system.

The team's innovative vehicle frame design allows for easy access to the electronic enclosure, batteries, and payload. The acrylic access panel on the front of the vehicle allows for quick serviceability, provides impact resistance in case of object detection failure, and weather resistance protection for the electronic components inside. The innovative camera mount design matches the vehicle frame aesthetic, provides water resistant protection, and has a rugged design to minimize vibrations to the camera while traversing rough terrain.

5. Mechanical Design

WASPP has a two wheel differential drivetrain and a supporting caster wheel design. Its small size and lightweight frame design increase course maneuverability and reduce torque requirements. A blue mirror acrylic shell provides weather resistant protection and an aesthetically pleasing look to the vehicle.

5.1. Chassis

Both steel and aluminum were considered as potential material for the vehicle chassis. While steel is much stronger than aluminum, it has a much higher cost and weight. This led the team to choose aluminum for the chassis structure in order to meet the budget and keep the total weight of the vehicle to a minimum. Hand calculations were performed early on in this stage to determine aluminum would have the necessary structural strength to comply with the requirements of our system. An FEA analysis (detailed in section 9.1.1) was completed to ensure these calculations were correct. Aluminum square tubing having a width and height of 1" and a wall thickness of .125" was chosen for the frame. The tubing was cut to size and welded in the LTU fabrication lab. The final chassis shown in Figure 1, came out to be 26 inches wide, 37 inches long, and 29 inches tall. The chassis was matte painted black for aesthetic purposes. The chassis hosts slots for batteries, mounting support for wheel assemblies, a platform for the competition payload, and a compartment for electronic equipment. Three wheels, two front drive wheels and one rear-rotating caster support the vehicle. Currently, there have been no known issues with the chassis structurally or in its size.

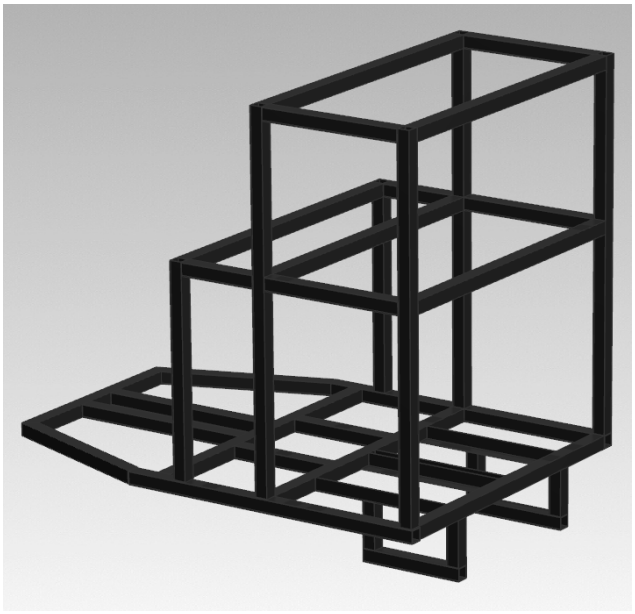

Figure 1: WASPP Chassis

5.2. Drivetrain

The drivetrain uses a three-wheeled design with two main drive wheels and a caster wheel in the rear. This configuration allows the vehicle to have a higher contact pressure with the ground verses a four-wheel design. Having a three-wheeled design ensures that even on rough terrain that all three wheels maintain contact with the ground. A Vexpro double reduction gearbox drives the two powered wheels. This gearbox uses two Vexpro CIM motors to produce an estimated torque of 406 inlbs. This motor and gearbox design were chosen due to their relatively lower cost than similar options and their torque outputs.

5.3.Suspension

The two main drive tires are made of a hard, but impact absorbent foam. These along with the tubed caster wheel act as the vehicle's suspension. The camera performance will be effected the most due to the minimal suspension incorporated to the vehicle. To counteract the vibrations from the vehicle while in motion the camera will be mounted to a rugged and sturdy shaft to minimize movement. Camera software will also be designed to filter out vibrations and account for vehicle movement over rough terrain.

5.4.Weatherproofing

The vehicle utilizes custom fitted silicone gaskets around all acrylic exterior and access panels. These gaskets will ensure that the sensitive electronics will not be exposed to moisture. The

vehicles weatherproofing will be water resistant and allow the vehicle to operate in light rain conditions.

6. Electrical Design

WASPP's electrical system supports a Jetson TK1, three Arduinos, a ZED stereo vision camera, four motors, a display screen, network switch, and safety devices. Four 12 volt batteries in parallel supply the vehicles on board power.

6.1. Power distribution system

The vehicle's on-board power system consists of (4) 12 volt batteries and a power distribution panel. The batteries are wired in parallel to produce a 12V system with a 72 Amp/Hour capacity. The charger chosen to use for recharging the batteries has 2Amp, 4Amp, and 6Amp charging options. On the 2Amp setting, the recharging process takes 18 hours to complete. For the 6Amp option, recharging is estimated to take 6 hours. Our goal is to use the 2Amp option along with proactive measures to ensure the batteries have a long lifetime. See Table 2 for the power requirements of the electrical components on board the vehicle.

Table 2: Power Requirements of Electrical Components

Component	Quantity	Max Power Consumption	Operating Voltage	Source
Jetson TK1	1	30W	12v	Distribution Board
Arduino Uno	3	0.6W	12v	Distribution Board
CIM Motors	4	337W	12v	Distribution Board
Display Screen	1	120W	120v	DC/AC Converter
Talon Motor Controllers	4	0.25W	5v	Arduino Uno
Network Switch	1	10W	5v	Distribution Board
ZED Camera	1	1.9W	5v	Jetson TK1

Maximum demand Calculations:
Max Power consumption = 1512.6W
Battery capacity = 864W/Hour
Run time = 34 minutes at maximum demand

6.2. Electronics

6.2.1. Jetson TK1

The Central Computer of WASPP is the Nvidia Jetson TK1. It is responsible for running all higher-level algorithms, including vision processing, obstacle mapping, lane mapping, and path planning. It is also responsible for coordinating and dictating all the lower microcontrollers. It runs Ubuntu 14.04 as well as OpenCV, ZED, and CMake to use and compile these algorithms. It runs on 12 volts and rarely needs more than 2.5 Amps [1]. It draws power through a 2.1x5.5 mm DC power jack.

6.2.2. ZED Stereo Camera

In order to meet the two main functionality requirements of the competition (lane following and obstacle avoidance), an Environment Detection System (EDS) was needed. The ZED stereo vision camera is used as the primary sensor for lane following and obstacle avoidance within the

EDS. Compared to the alternative of using a combination of LIDAR and web cameras, the ZED's 110 degree viewing angle allows the system to simultaneously find lanes in the grass and detect barrels. The ZED camera is designed to work with the Jetson TK1 and includes the libraries necessary for accessing the images and depth map.

6.2.3. Ultrasonic Sensors

A combination of the ZED stereo camera and ultrasonic sensors comprise the EDS. The ZED camera has a minimum range of 0.5m therefore; an object within the range of 0 to 0.5m will not be detected by the ZED. To avoid missed objects, a redundant obstacle avoidance system or backup system, comprised of five HC_SR04 ultrasonic sensors was designed to detect objects within this close range. Testing proved that the sensors could detect objects up to 4 meters away from the vehicle and within an angle of 13 degrees. Based on the measuring angle of 13 degrees, three sensors were mounted on the front of the vehicle and one on each side to maximize the redundant system coverage area. All five ultrasonic sensors are connected to an Arduino Mega, which communicates with the Jetson TK1 via the network switch.

6.2.4. GPS and Compass

The team selected the Adafruit Ultimate GPS breakout board for the vehicles GPS module. This module was selected for its high number of channels and external antenna connector. The high number of available channels will reduce the initial fix time with satellites and the external antenna will aid in the GPS modules satellite tracking. The GPS module was also selected for its high accuracy relative to price. Without using a base station, it was a challenge to find a GPS module with consistent accuracy within the two-meter waypoint competition requirements within the team's budget. With thorough testing, the team aims to establish a typical offset that can be used to accurately navigate to waypoints at competition. An Adafruit Absolute Orientation Sensor was selected by the team to serve as the vehicles compass. The compass will provide vehicle-heading information for path planning and GPS waypoint navigation in No-Man's Land. This component was selected for it is ability to provide heading output information in the Euler Vector form as well as the in Quaternion form. The path planning algorithms are designed to use the three axis orientation data based on a 360° sphere from the Euler Vector form. However, the component provides design flexibility for future use or problem solving. Both components will be connected to an Arduino for data collection and processing. The Arduino will then transmit latitude, longitude, and heading data to the Jetson TK1 to be used in the vehicles path planning algorithms. See Figure 2 for the GPS and compass connection with the Arduino.

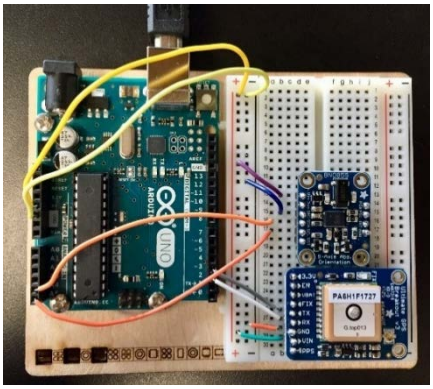

Figure 2: Vehicle GPS and Compass wiring to the Arduino

6.2.5. Network Switch

The Jetson TK1 and Arduinos communicate with each-other through TCP/IP protocols. The Arduinos accomplish this using Keyestudio W5100 Ethernet Shields. Each of these shields can take 12 V. The shields get their power from the Arduino they are connected to. Each of these devices is connected to an un-managed Tenda S108 8-port Fast Ethernet Switch through CAT6 cables which runs on 5 volts as shown in Figure 3.

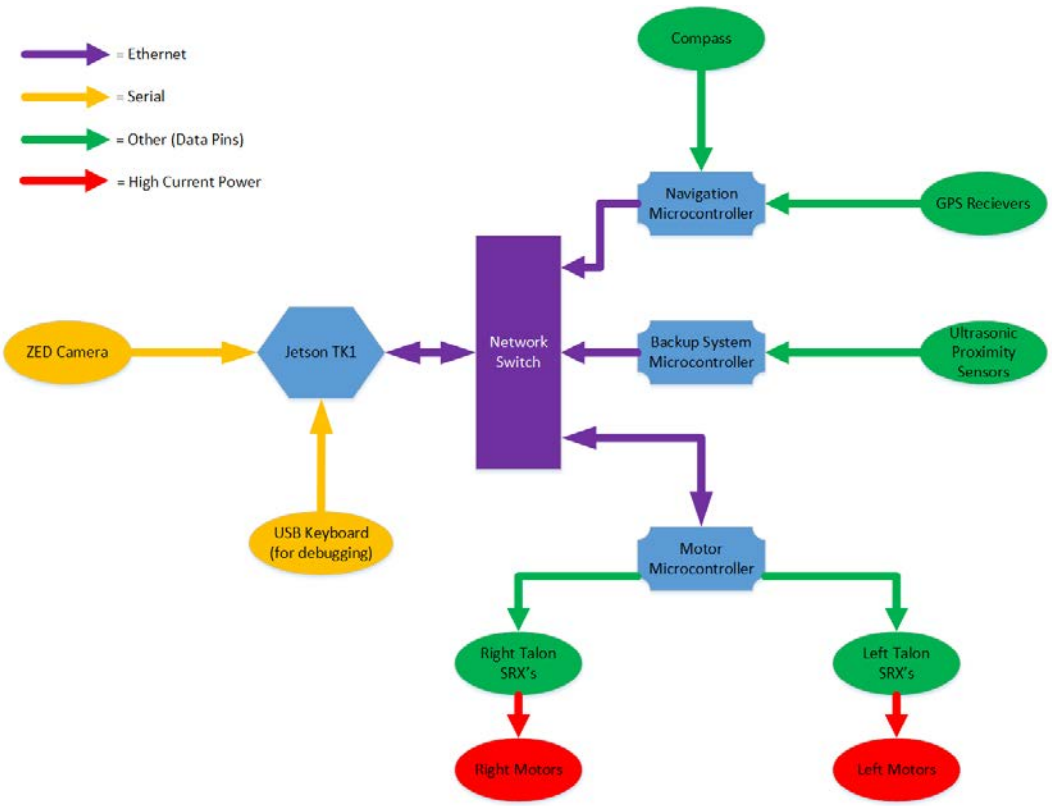

Figure 3: Wiring diagram for data wires

6.3. Safety Devices

WASPP uses an LED strip light, which encircles the top edge of the electronics enclosure as its safety indication light. When the vehicle is in powered on, the light will be solid and it will begin flashing when the vehicle is in its autonomous mode. The colored strip light is weatherproof and could also be used as a communication tool regarding the states the vehicle is in within its autonomous mode. For example, a blue flashing light could indicate ultrasonic thresholds reached and a red flashing light could indicated an obstacle detected by the ZED camera.

The vehicle also is equipped with an on board emergency stop button, positioned rear center on the top of the electronics enclosure. In addition, a wireless emergency stop button can be used up to 100m away from the vehicle.

7. Software Design

The computers onboard WASPP are the Nvidia Jetson TK1 central computer (TK1), an Arduino Uno for motor control (MCA), an Arduino Uno for navigation (NA), and an Arduino Mega for reading the redundancy object avoidance system proximity sensors (ROASA). The data each computer sends and receives is displayed in Figure 4. The programming languages used are C++ for the TK1 and Arduino IDE for the Arduinos.

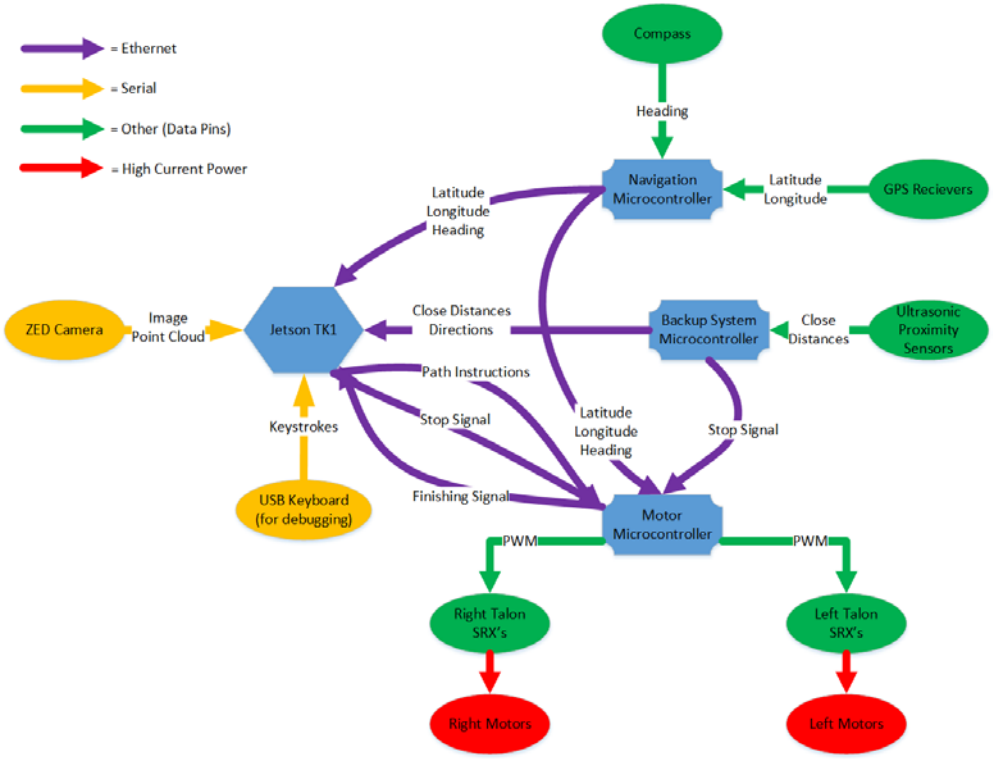

Figure 4: Data Transmitted Between Devices

7.1. Jetson TK1

7.1.1. Point_Map.h

The processes run in the TK1 are built around the Point_Map, a custom C++ class. It stores all points that have a detected obstacle or lane marking, as well as the current goal, in relation to the robot. These points are stored as another custom C++ class, mapPoint. They are stored as Cartesian coordinates in feet with WASPP being (0, 0), forward direction being the Y-axis.

7.1.2. Avoidance Mapping

"Avoidances" are obstacles and lane markings. Any data relating to them that is used by higher level processes come from the ZED depth camera.

Obstacles are mapped using the Point Cloud provided by the ZED API. The point cloud is given in the form of (X, Y, Z), which represent an objects horizontal position, vertical position, and distance respectively, in relation to the camera. This Data is rotated to account for the camera being angled downwards. Then, the (X, Z) values are added to the Point_Map if their corresponding Y value falls within a certain range (to avoid adding ground points).

Lane markings are mapped using a raw camera feed and OpenCV. After converting the ZED image to an OpenCV one, it is run through HSV based filters to filter out shadows, dead grass, dirt, and other possible discolorations to make sure the lane markings are the brightest parts of the image. After this, the program thresholds the image to make sure that lane markings are pure white while anything else is black. The filtered pixels are then related to the camera's point-cloud. The point-cloud info is stored in a similar manner to an image, simply storing X, Y, and Z values instead of Blue, Green, and Red. In this scenario, OpenCV uses HSV based filters to determine if the pixel is of a lane marking or not. If it could be, it then checks the corresponding "pixel" in the point cloud. If that pixels "true-Y" value (distance from ground) is too high, the program filters it

out and moves on to the next pixel. Otherwise, it stores the point's X and Z values, converts them to feet, and adds it to the Point_Map (as X and Y, respectively).

7.1.3. Goal Mapping

Upon program start, two lists of GPS Waypoints are inputted and stored. One list stores the waypoints at the transitions from Lane to No-Man's-Land and vice-versa, while the other has all the waypoints within No-Man's-Land. Each stored waypoint also has a flag to indicate if it has already been visited to prevent backtracking.

WASPP starts in a Lane portion. As it moves along the lane, it checks its GPS position. If it gets within a short range of one of the transition waypoints, it marks it as visited and switches to No-Man's-Land state. Later, while WASPP is in No-Man's-Land it will reach the other transition waypoint. When it does, it will switch back to a Lane Following state. The only things that change in these transitions are how the current goal is selected and how much of a path should be executed (more on that in section 7.1.4). Otherwise, the path planner is un-changed.

In No-Man's-Land, the current goal is based on the given GPS waypoints. When choosing a waypoint, it reads its the current GPS position and compass heading. It then analyzes the distances to all un-visited, non-transition waypoints and uses the closest. If there are no more, it will use the un-visited transition (one was already visited to get into No-Man's-Land). WASPP then determines the distance in feet and its own heading to get an angle, creating polar coordinates [2] [3]. These polar coordinates are converted to Cartesian coordinates and added to the Point_Map.

In Lanes, the current goal is based on the surrounding lane markings. WASPP reads the detected lane marking points (ignoring obstacles for now) and finds the farthest one that has "line-of-sight" to itself. It then places the goal there and moves it closer unit-by-unit until there are no avoidances within a fixed range of the goal.

7.1.4. Path Planner

The Path Planner is the same in both No-Man's-Land and Lane portions. It is unique in that it starts with a straight line from start to goal and works from there. It is a greedy algorithm because it is more concerned with quickly finding a reasonably short path, rather than the shortest possible path. It also assumes that the robot will not need to go backwards or approach the goal from behind, both of which are reasonable assumptions for this competition. It works by starting with a straight line from start to goal, which will be referred to as the "Primary B-Line". It then finds the closest obstacle to the robot that is within a "buffer distance" of that line (buffer distance is distance from the robots effective center that is needed to clear any lanes and obstacles). If there are none, WASPP will simply go towards it. Otherwise, it will find the closest clear point to either side of the obstacle and find the closest obstacle to that point between it and the robot. It will repeat this until a clear path is found going from the robot to one of those original points. The other path is then forgotten. This process is demonstrated in Figure 5.

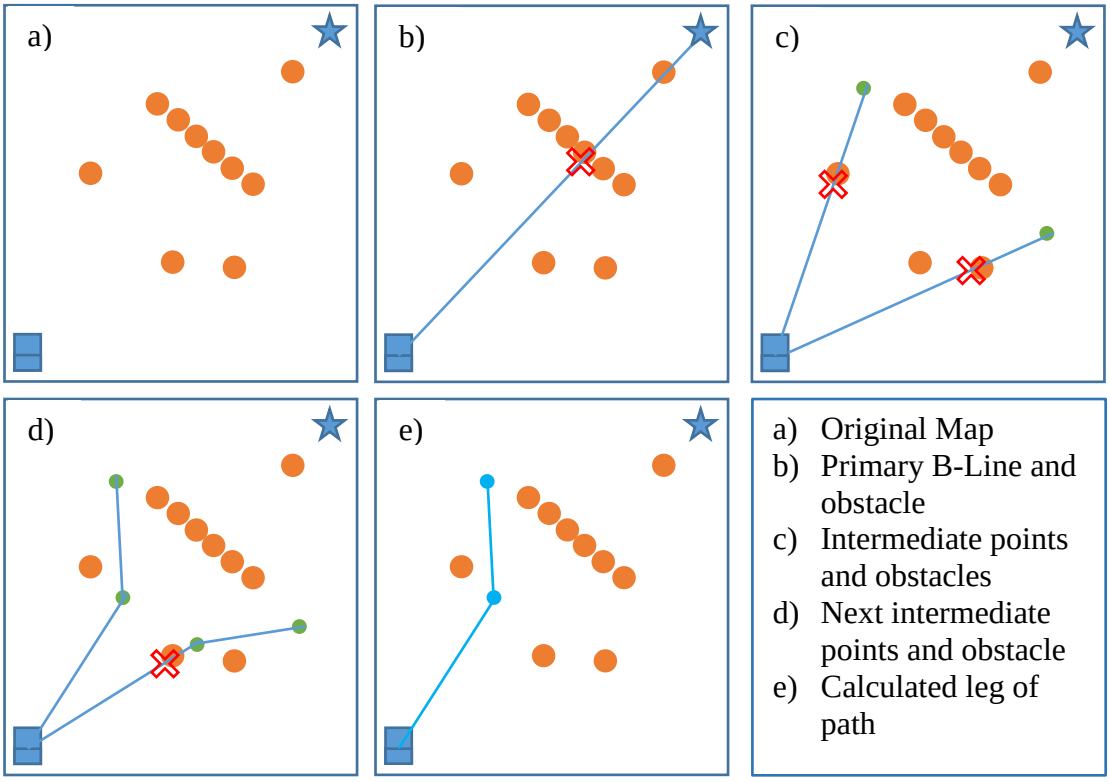

Figure 5: Pictorial Depiction of Path Planning

At this point one of two things will occur. If the goal is to find a complete path (such as in simulations and testing), the planner will repeat this process starting from the end of that path. It will continue repeating until the goal is reached. However, in practice the robot will find a path around that first obstacle, send it to be executed, and go idle until one of three things occur:

1) The MCA sends a signal that it completed its current path
2) An avoidance is detected as coming within a certain range of the robot on the Point_Map
3) An obstacle is detected as coming within a certain range of the robot by the ROASA

When any of these happen, the program sends a stopping signal to the MCA (2 or 3) and begins the process again. It does this because it will not be able to perceive the field beyond the first obstacle with much reliability. Therefore, it is faster to find the first leg of the path and wait until it is in a position to better perceive the field before doing the next. If it is in lane following mode, it will only execute to the first waypoint in the path.

The MCA will not be able to do much if the path is presented as a sequence of waypoints, therefore, the TK1 needs to translate them into a series of instructions. These instructions take the form of rotations and translations. A rotation is how much the robot needs to turn counter-clockwise to be facing the next point and a translation is how far the robot needs to travel after the rotation to reach the next point. This works by creating a value to store the robots current heading (initialized at zero). Then, for each point in the path, it finds the polar coordinates of the next point from the current. It then subtracts the current heading from that angle to get rotation, uses the radius as translation, and sets the current heading as the angle.

The socket functions in C++ and the Arduino requires the data to be represented as a char (or byte) array. This array will be a sequence of char arrays. Each of those arrays will store one byte for a byte integer to keep the order of the instructions, one byte for a character to signify if the instruction is a rotation or translation, and 4 bytes for a double to signify the angle to turn or the

distance to travel. The array will look like this: $b_1^{int}, b_1^{char}, b_1^{double\,1}, \dots, b_1^{double\,4}, b_2^{int}, b_2^{char}, b_2^{double\,1}, \dots, b_2^{double\,4}, \dots$
The conversions will be performed using unions.

7.2. Navigation Arduino

The Navigation Arduino (NA) reads serial data from an Adafruit Ultimate GPS Breakout board and an Adafruit Absolute Orientation Sensor (compass) using pre-installed libraries. The compass gives orientation in Euler Angles, but only the first one is needed to get the vehicles heading. Each sub-array of the char array contains a character to signify if the value is Latitude, Longitude, or Heading. Then 4 bytes are used to store a double that represents the value. The array looks like this: $b_1^{char}, b_1^{double\,1}, \dots, b_1^{double\,4}, b_2^{char}, b_2^{double\,1}, \dots, b_2^{double\,4}, \dots$ The conversion are performed using unions. This data is sent to the TK1 for goal calculation and to the MCA for feedback.

7.3. Redundancy System Arduino

The Redundancy Object Avoidance System Arduino (ROASA) reads distances from five ultrasonic proximity sensors. It then transmits the as a char array in the form $b_1^{int}, b_1^{double\,1}, \dots, b_1^{double\,4}, b_1^{int}, b_1^{double\,1}, \dots, b_1^{double\,4}$, where the integer signifies what sensor is giving the reading, and the double is the actual reading. This data is sent to the TK1 for processing. It also checks if anything comes within a certain range, sending a signal to the MCA if anything does.

7.4. Motor Control Arduino

The Motor Control Arduino (MCA) can control the speed and direction of the each wheel by sending a PWM signal to Talon SRX Motor controllers.

7.4.1. Path Execution

The MCA receives decodes, and stores the instructions received by the TK1. It then executes each one in turn. For rotations, it reads the current heading from the NA and calculates the desired heading. It then turns in place until the current heading matches the desired. For translations, it reads the current GPS location from the NA and calculates the desired GPS location along the same heading. It then travels forward until it is at (or close to) the desired location. As WASPP is traveling forward, it reads the change in current heading over time from the NA. It adjusts the speed of the wheels to keep the change as close to zero as possible (to keep heading straight). Once the path is finished, it sends a signal to the TK1 to let it know it needs to create a new path.

7.4.2. “Stop” Signals

Sometimes, a path will need to be abandoned due to an un-accounted for obstacle or lane marking. In these cases, the MCA will get a “Stop” signal, which will tell it to stop the motors and abandon its current path, waiting for a new one. At the same time, the TK1 will begin planning a new path that factors in the new information. This signal could come from the TK1, triggered when it sees something on the Point_Map that is too close. The signal could also come from the ROASA, if an obstacle comes within a certain range.

8. Failure Modes and Failure Points

8.1. Vehicle Failure Modes

8.1.1. Software

Lane mapping can be a challenge because it relies on the camera's ability to tell lane markings from anything else. While it is possible to create a filter that removes most noise from the used image there is the possibility of something (dead grass, a shiny patch, etc.) being confused for a marking. There is also the possibility of the filter accidentally removing portions of a lane marking. In these cases, the resolution is to fine-tune the filter. The data update rate could also be increased, so the noise isn't around long enough to cause lasting problems.

8.1.2. Path Planning

As mentioned in section 7.1.4, the path planner is built on the assumption that WASPP will not need to go backwards or approach the goal from the back. While this is unlikely to occur in the competition, it is still a potential problem for the robot to face. There is also the possibility of a complete blockage between two points trapping the planner in an infinite loop. In either of these cases, a possible solution would be to add a watchdog that tells WASPP to reposition and try again if it takes too long to find a path.

8.1.3. Communication

It is possible that the double – to – char array – to double conversions used for the communications will alter the data, such as dropping the decimal point. This could be fixed by multiplying a decimal number by 10^n so that it is an integer (as in nothing after the decimal) and sending n as well so the recipient can divide by 10^n. It is also possible that the recipient will read the bytes on an offset, as in it misses the first byte and reads the second as the first, third as the second, etc. While this is a big concern using USB serial, it is unlikely to be a problem since TCP/IP protocols are being used.

8.2. Vehicle Failure Points

8.2.1. Electronic

A GPS module point of failure could be the external antenna point of connection. The u.FL connector for the external active antenna is very small and delicate. If the GPS module and antenna are not fixed on the vehicle, the connectors could easily become disconnected or damaged during testing. To prevent this point of failure the internal antenna will solely be used on the GPS module until the GPS and antenna can become permanently fixed on the vehicle in a safe and weather resistant location.

If the ZED stereo camera fails during competition, then the redundancy obstacle avoidance system could take over obstacle avoidance tasks. However, a web camera would need to be added for lane detection.

The on-board display screen uses a 5 amp fuse in the DC to AC converter for its power. This fuse is a purposely made as a weak point in the event of a current overdraw. As a resolution, if the fuse blows it is easily accessible for maintenance and replacement. In addition, all electronics on-board are protected by fuses in the main power distribution board. These fuses are easily accessible behind the vehicle's front access panel. Replacement fuses will be brought to competition in case of a failure.

8.2.2. Electrical

The power distribution board has 12 open slots available that are not in use. If a terminal connection is damaged, then the connection point can be moved to another open slot. For protection, the power distribution board uses temperature controlled fuses. The fuse will trip if too much current is drawn and will reset after a 2-3 minutes of cool-down time.

The 12V battery bank on-board made is separated into two sections. These sections are made of two individual batteries and are replaced regularly for charging. If a set fails during competition, the failed section can easily be changed out and the vehicle can continue.

8.2.3. Mechanical

To date no failures have occurred on the vehicle from a mechanical standpoint. In the case of parts breaking or failing, the team is prepared to bring as many tools and equipment to the competition as necessary.

8.2.4. Structural

The vehicle frame is made of welded aluminum square tubing. There is potential for welds to crack or corrode. To prevent corrosion, a corrosion resistant paint was used on the chassis. To prevent cracks in the welds, high stress points have full welds around the joint with no gaps.

8.3. All Failure Prevention Strategy

Most failures in the software will likely come from inaccurate mapping because of incorrect calculations. In the lane marking detection code, this could come from incorrect measurements of the camera's position or image. In the object avoidance code, this could come from the ZED camera not functioning properly in the competition conditions or a misunderstanding of how the depth field is presented. Our team has come up with a redundant obstacle avoidance system that uses five ultrasonic proximity sensors for such scenarios where the ZED camera has failed to detect an object.

8.4.Vehicle Safety Design Concepts

The vehicle was required to have an onboard mechanical estop. This estop button is placed on the top of the vehicle which cuts the power off from batteries to the rest of the Drivetrain System. By cutting power to only the Drivetrain System ensures that the vehicle comes to a complete stop and does not affect any of the other onboard electrical components. The Wireless estop is also attached solely to the drivetrain system circuitry so that it does not disable any of our sensors or communication devices. To indicate whether the vehicle is in autonomous or manual mode, the vehicle is equipped with an addressable LED strip as our safety light. It uses 5 volts and gives an aesthetically pleasing look to the vehicle.

9. Simulations

9.1. FEA Analysis

An FEA analysis was completed on the aluminum tubing used in the chassis of the vehicle. A worst case scenario was defined as being twice the vehicle weight (300 lbs.) distributed across one aluminum tube being 28 inches and having fixed ends. As seen in Figure 6, the max deformation is 0.0335 inches. This is well within the necessary structural integrity of the chassis.

9.2.Path Planning

The use of the Point_Map makes it easy to test the path planner in a simulated environment. To do so, a Point_Map was created with obstacle and lane-marking points entered into it manually. A goal point was also added manually or using the lane goal mapping method. Then the path planner was run using that map as it would be on the final robot. This was instrumental in debugging the planner.

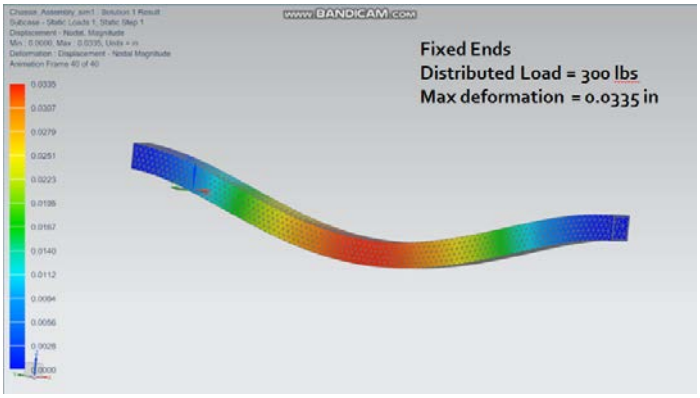

Figure 6: FEA Analysis

10. Testing

A test area at LTU was created to mimic the competition course at Oakland University. The test course was created to incorporate shadows, dead grass, sinusoidal curves, and straight aways. See for an overhead view of the test course.

Figure 7: LTU Test Course

Preliminary drivetrain testing revealed that the set screws we chose would come lose under vibration. These screw mounted the wheel hubs to the output shaft of our gearboxes. To correct this we drilled pockets in the shaft and applied thread locker to the set screws.

For battery testing, the vehicle was operated using the batteries in the grassy test course. In this area, light duty maneuvers were performed with the vehicle. After testing for 2 hours, the batteries were measured for their remaining capacity. It was determined that after this 2-hour test, the batteries were depleted by 40 percent.

GPS module testing was conducted in the quad at LTU and around campus. To test the accuracy of the GPS, an easy to identify location on google maps was chosen, typically the corner of a sidewalk. The GPS module was then placed on the exact location and latitude and longitude coordinates were record after the sensor stabilized, outputting the same data consistently. The Arduino output coordinates had to then be converted from degrees and decimal minutes to the decimal degree format google maps uses. Once the data was converted the distance between coordinate points was evaluated. According to google maps, the accuracy of the GPS ranged from 2.7-5.1 meters. It is hard to decipher exactly if the inaccuracy is from the GPS module or from the coordinate error associated with google maps. To minimize the GPS at competition the GPS module will be placed directly on the known waypoint in the qualification course. The given waypoint coordinates and the GPS coordinates will be compared to determine an offset. This offset will then be integrated in to the vehicles path planning for No Man's Land.

Camera testing revealed that the original method of mapping lane markings using a trigonometric method to the determine the position of the white markings on the ground was not the most efficient. While most of the equations were sound, it was reliant on there being a fixed relationship between the location of the pixel in the image, and the angle from the camera's center. For most cameras, this is taking the pixel location (from center) and multiplying it by $\frac{view\ angle}{image\ size}$.

However, this is not the case with the ZED camera due to its 110° viewing angle. With testing, the team determined that it is more efficient to relate the pixel to the camera's point-cloud. See Figure 8 for a sample of the ZED stereo vision camera's depth map, 2D and 3D image outputs during testing.

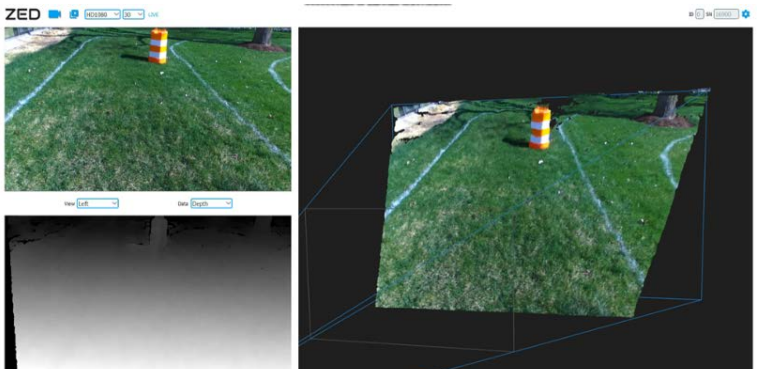

Figure 8: Stereo Camera Output

11. Budget

The team has currently spent approximately $2,262 on vehicle supplies. This is well within the team's $3,000 design goal. Through a crowdfunding platform and corporate outreach, the team was able to successfully raise $3,334. These funds cover the total vehicle cost and leave money remaining in the IGVC account to help next year's team get started on the 2019 IGVC vehicle. See Table 3, for purchased a list of purchased components and costs.

Table 3: Purchased Vehicle Component Costs

Purchased			
Component	QTY	Cost/Unit	Total Cost
Aluminum Tubing	NA	NA	$ 163.05
Drive Motors	4	$ 27.99	$ 111.96
Motor Gearbox	2	$ 59.99	$ 119.98
Motor Controller	4	$ 89.99	$ 359.96
Jetson TK1	1	$ 129.99	$ 129.99
Network Switch	1	$ 8.99	$ 8.99
ZED Camera	1	$ 469.00	$ 469.00
GPS	1	$ 37.30	$ 37.30
IMU	1	$ 34.95	$ 34.95
Ultrasonic Sensors	5	$ 1.99	$ 9.95
Antenna	1	$ 14.99	$ 14.99
Spray Paint	7	$ 3.48	$ 24.36
Competition Registration Fee	1	$ 300.00	$ 300.00
Arduino Ethernet Shields	3	$ 9.99	$ 29.97
Ethernet Cables	4	$ 2.60	$ 10.40
Battery Quick-Connectors	3	$ 4.99	$ 14.97
Small Power Connectors	1	$ 7.99	$ 7.99
Hardware (50 pieces)	1	$ 20.00	$ 20.00
Circuit Breakers	1	$ 34.44	$ 34.44
Volt Meter	1	$ 2.50	$ 2.50
Bus Bars	2	$ 5.00	$ 10.00
Mini Breadboards	1	$ 4.00	$ 4.00
Blue Mirror Acrylic	1	$ 270.00	$ 270.00
Sponsor Stickers	7	$ 7.00	$ 49.00
LED Strip Light	1	$ 12.00	$ 12.00
Sealant	1	$ 12.00	$ 12.00
		Total	$ 2,261.75

References

[1] NVIDIA, "Q&A Jetson TK1 FAQ," 1 5 2014. [Online]. Available: http://developer.download.nvidia.com/embedded/jetson/TK1/docs/Jetson_TK1_FAQ_2014May01_V2.pdf.

[2] P. Lewis, "How to Convert GPS Coordinates to Feet," Sciencing, 24 April 2017. [Online]. Available: https://sciencing.com/convert-gps-coordinates-feet-7695232.html.

[3] A. Upadhyay, "Formula to Find Bearing or Heading angle between two points: Latitude Longitude," GIS Map, July 2015. [Online]. Available: https://www.igismap.com/formula-to-find-bearing-or-heading-angle-between-two-points-latitude-longitude/.