

University of West Florida IGVC Design Report

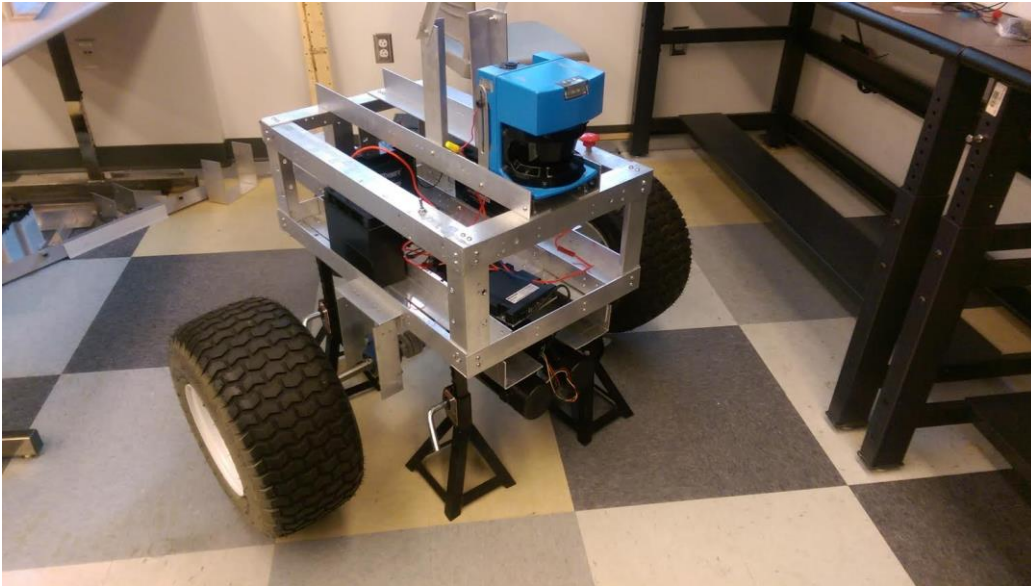
HAL

Group members

James Davis, Major: Computer Engineering, email: jed30@students.uwf.edu


Kenny Dang, Major: Electrical Engineering email: kqd1@students.uwf.edu

Kenley Tan, Major: Electrical Engineering email: kt34@students.uwf.edu



Faculty Statement:

I certify that the work done by all the students on this project is consistent with a senior design course and that the vehicle has been significantly modified for this year's competition.



Dr. Oscar Chuy Jr.
ochuy@uwf.edu

1. Team Organization

Our team consists of three members. James Davis is majoring in Computer Engineering, Kenny Dang is majoring in Electrical Engineering, and Kenley Tan is majoring in Electrical Engineering. All three of us are seniors and completed part of this project for our senior design project. James is the team leader and was the lead programmer on the team. Kenny designed and built the chassis and the electrical system. Kenley was in charge of gathering and testing all the components so that they could be placed on the robot.

2.1 Design Process

At the start of our design, we completely disassembled last year's robot and checked out the parts that were still useable. Our team agreed that there should be differential drive in order to work well outdoors and that there should be a caster in the back so that there were only two powered motors. Once this design was agreed upon by our team and our advisor then the chassis was built. Next we needed to make a power system. Our advisor suggested using a Mac mini for the main computational unit on the vehicle and that needs a consistent DC power supply. It is also not good for the operating system to be shut down by losing power. In order to circumvent this, our design uses two power supplies, one for the motors, LIDAR, and motor controller and another for the computer and router. The lidar and motor power system is 24V and the computer and router system is a 12V system. We used identical car batteries for all of the systems. In order to get a 24V output, two batteries were placed in series. Once the power system and components were mounted all the software was tested. Due to our small team size all of the team members participated in building the robot. Then the code that was written was tested.

2.2 Design Assumption

There were multiple ideas we implemented in creating a design for our vehicle. We wanted the motors to be located in the front to create a front wheel drive vehicle. This allows the wheels to create a pulling motion instead of a pushing motion which allows for better control when maneuvering, while also reducing the friction due to the pulling motion. Another idea was to have the motors located at the bottom and most of the weight at the bottom half of the vehicle. Having the weight distributed low prevents the vehicle from flipping over while turning. We assumed the vehicle would be heavy, and opted to use wheelchair motors which are used for moving heavy weight.

3. Design Innovation

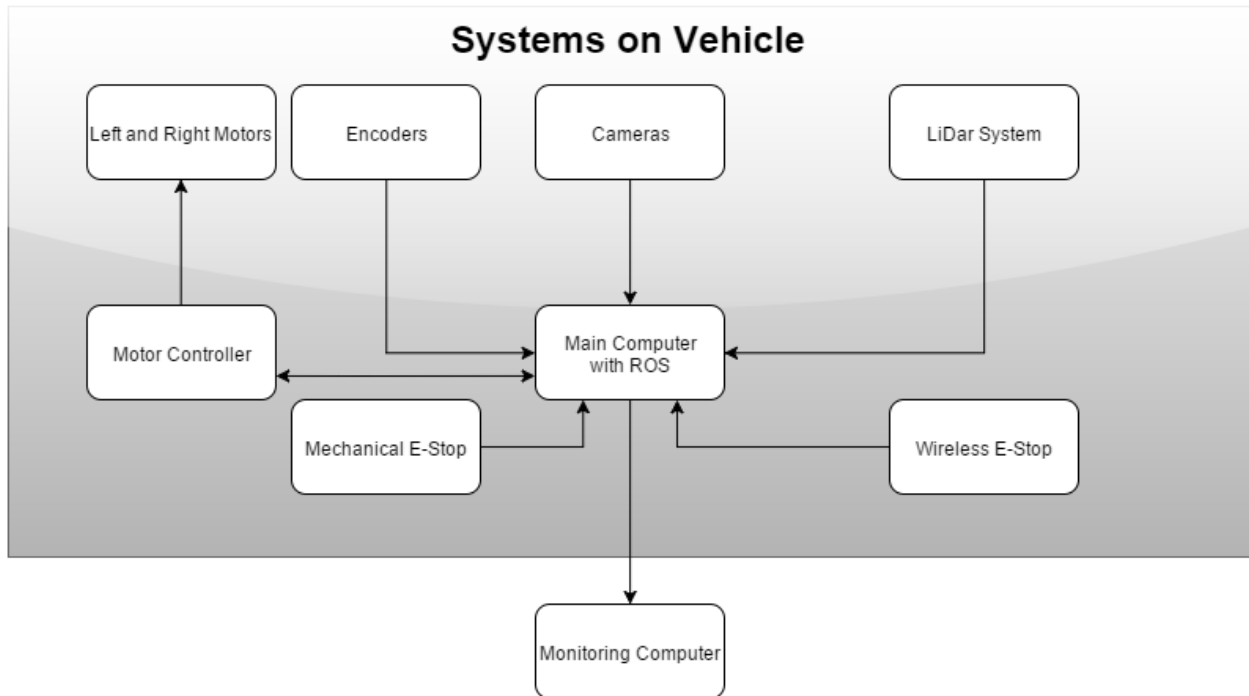
Compared to the previous year's vehicle, the current vehicle, HAL, is completely different. Instead of chain motors, we replaced the motors with electric wheelchair motors. This prevents the faults of having chains, such as the chain coming off or debris sticking into the chain. Last year's team used both a raspberry pi and an arduino with sonar sensors. Instead of using sonar sensors we opted to use a lidar instead. The lidar will give more resolution, higher rate of data acquisition, and larger range. In order to find the position of the robot high accuracy encoders were added to the vehicle. These encoders are more accurate than last year's robot's encoders in order to map its current position. The raspberry pi and pi camera were not powerful enough to process the images as fast as desired. Instead of the pi camera a usb camera is being used to gather images and a ROS node is processing the images and adding the lines to the map

for the vehicle to avoid. The vision processing node is computed on the mac mini in order for it to be fast enough to work in real time.

4. Mechanical Design

4.1 Overview

The original plan was to use the previous frame and make adjustments to the vehicle, but the team decided to completely scrapped the frame. The main reason being the difference in motors between the older model and the newer model. Even though we completely remade it, we reused most of the materials and parts from the older model.



The Functional Requirements and Specifications for the blocks are listed in Table 1-8

Table 1: Main Computer Running ROS

Requirements	Specifications
Functional	Must take in all sensor data and then make decisions based upon them
Performance	At least 20 full program cycles per second
System Interaction	Main decision making part on the robot
Operator Interaction	Operator will give GPS coordinates and put it into autonomous mode
Hardware/Software	Will have a mac mini running ubuntu with ros software

Table 2:Cameras

Requirements	Specifications
--------------	----------------

Functional	Get image data for decision making
Performance	At least 30 frames per second
System Interaction	Will return image data to the main computer for analysis
Operator Interaction	No operator interaction
Hardware/Software	Must have a base resolution of 800x600 and have at least 30 fps

Table 3:Lidar System

Requirements	Specifications
Functional	Identify obstacles in front of the vehicle
Performance	At least 20 frames per second
System Interaction	Return a stream of depth data to the main computer
Operator Interaction	No operator interaction
Hardware/Software	Have a minimum range of 10m and 180°

Table 4: Wireless/Mechanical E-Stop

Requirements	Specifications
Functional	Stops the robot
Performance	Must stop the robot immediately at least 100m away
System Interaction	Cuts power to the robot
Operator Interaction	Presses large red button
Hardware/Software	Mechanical: Cut power, Wireless: Software interrupt

Table 5: Motor Controller

Requirements	Specifications
Functional	Controls the motors on the robot
Performance	Must maintain velocity and directions of the motor
System Interaction	Receives the pulse width modulation from the main computer and returns encoder values for tracking distances.
Operator Interaction	No operator interaction
Hardware/Software	Brushless motor variant of the RoboteQ motor controller

Table 6: Left and Right Motor

Requirements	Specifications
Functional	Moves the robot
Performance	Must be capable of going at least 1 mph and no more the 5 mph
System Interaction	Receives the current and voltage from the motor controller
Operator Interaction	No operator interaction
Hardware/Software	Brushless motor used in electric wheelchairs

Table 7: Encoders

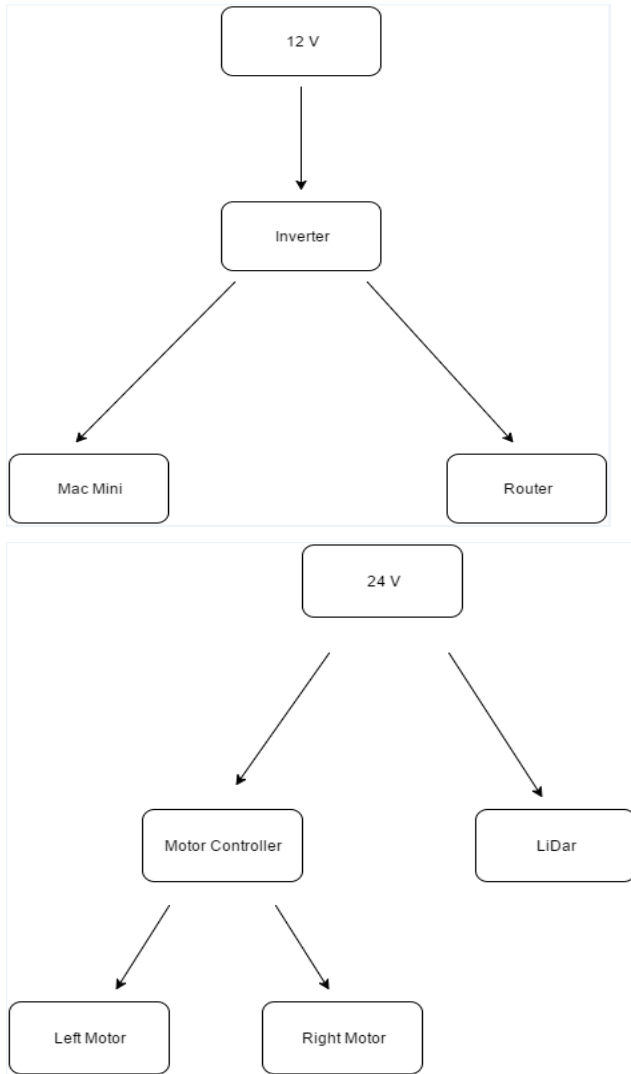
Requirements	Specifications
Functional	Returns the rotation of the wheels
Performance	Must be capable of returning correct data at up to 8000 rpm (max of the motors)
System Interaction	Sends data to the motor controller
Operator Interaction	No operator interaction
Hardware/Software	Must have 8mm shaft and be capable of running from 10-24V

4.2 Mechanical design

We opted to have our vehicle frame to be a rectangular shape. This will allow for the electronic components to be easily placed into the frame. The vehicle will have three wheels, two motor controlled wheels and one caster wheel. As mention before, we are using wheelchair motors,DG-158, for our design. Not will these motors provide more torque than our previous motors, they also come with encoders. Having encoders will help immensely in any motor based robot, allowing for easy speed and position control.

5. Electronic and Power Design

HAL's power consist of two different power systems. Two 12V batteries are connected in series to created a 24V system and a single 12V battery creates the second system. Many of the electronic components were either 24V or 12V. Having the two power system extremely convenient when testing. To prevent the vehicle from going to an undesired location such as a curb we had to quickly shut down the system. Instead of shutting down the whole system, we could just shut down the motors and LiDar and keep the computer on. This keeps our work saved while stopping the vehicle. Our two power systems are shown in the below figures.



6.0 Software

6.1 Computational Devices

The main computational computer is a mac mini that has a 2.6Ghz intel processor and intel HD graphics. This computer runs the robot operating system and aggregates all the sensor data. The robot will also have a roboteq motor controller and an arduino. The arduino runs an attached gps module and constantly reports the robot's current latitude and longitude. The roboteq motor controller runs a script that constantly publishes both wheels current speed. The motor controller also takes commands from the mac mini to change the velocities of the wheels.

6.2 Software Strategy and Control Decisions

The software of the robot relies on the Robot Operating System . ROS is a collection of tools for linux that contain drivers, communication protocols, and tools for working with robots. Our robot is running the ROS navigation stack. This stack requires a planar laser (the SICK LMS200 in our case) a base that can take data in a special ROS data format, and accurate odometry and transform data. In order to get the motor controller to take the special messages a ROS driver had

to be written. A custom driver was written in Python to publish the current speed of the robot to ROS and to take in the base messages and send the motor controller the new desired speed. This allows the navigation stack to autonomously control the speed and direction of the vehicle. The odometry data required its own special node. By using the speed of the motors that is being constantly published the distance from the initial point of the robot is calculated. This distance is recorded and the current distance the robot has moved is published to the stack. In order for the stack to correctly use the planar laser data a transform must move the laser data to the center of the robot. This is done using a static transform node that constantly transmits the position of the laser based upon the position of the robot. The waypoint will be determined by gps coordinates recorded from the sensor and the ones given by the judge.

6.3 Mapping

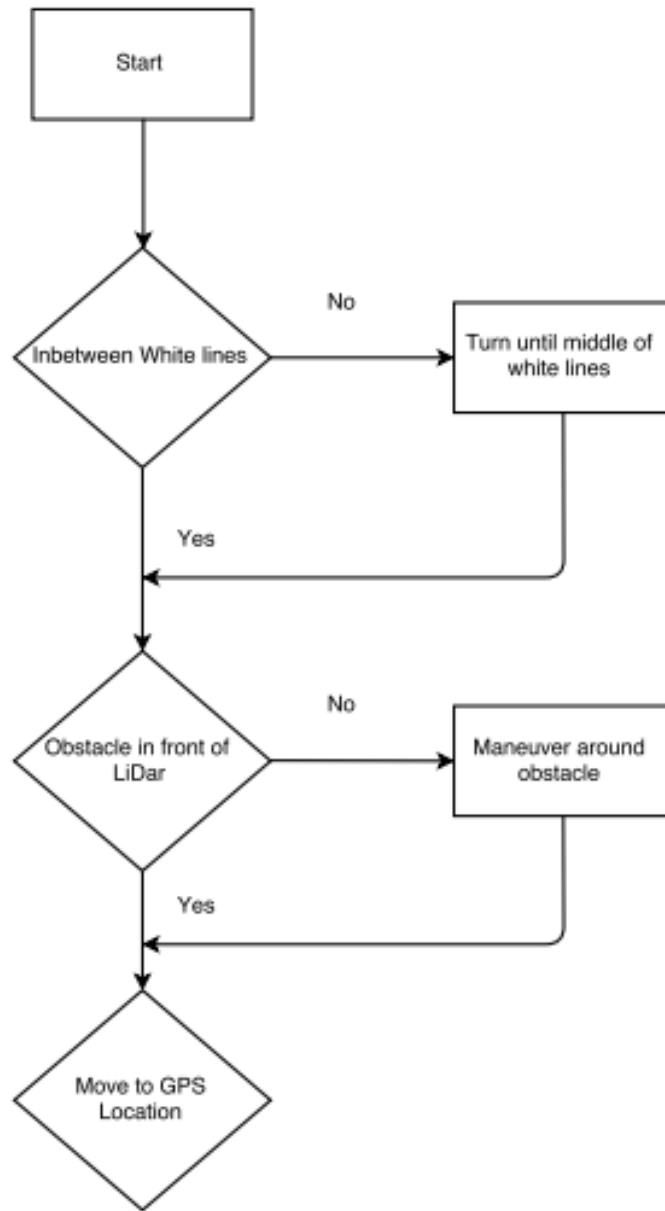
The ROS navigation stack uses two separate maps. There is a local cost map and a global cost map. The global costmap is recorded prior to navigation by recording all the odometry and laser data from the other ROS nodes. Once this global cost map is recorded then it is loaded upon initialization into the mapping server. Once a waypoint is given the global planner determines a path to the waypoint and sends the robot on its way. Once the robot is moving then a local map is created. The local map is created for immediate reactions on the way of the global costmap. The obstacles are detected using point cloud data from a camera and the planar laser.

6.4 Plan for Path Following

The vehicle has a threshold of 0.2m radially in which objects cannot be. If an object is found within that radius then the vehicle will back up and spin until an appropriate path is found. Other than this failure state the robot continues on the path communicated to it by the global and local planners. These planners send vectors out and compute the least cost path for the robot to complete. We have set the commands given out to be for differential drive robots so that Any path given to the robot is completed as a series of x and z movements. Our vehicle cannot accomplish direct movement in the y plane.

6.5 System integration plan and Signal Processing

The sensor information from the LiDar and camera is sent to the main computer that is running ROS, Robot Operating System. Based on the information gathered from the sensors, the main computer will make decisions to travel in the most convenient path. The decisions are made from a set of priorities that the team feels are most important going to least important. The priorities from most important to least important are white line detection, object detection, and gps location; this is shown in the below figure.



7. Failure

1. Moving directly towards the sun or shiny object disorientates the vehicle.
2. Unable to find flat or/and narrow objects.
3. The vehicle fails at avoiding obstacles in rapid succession.

8. Simulations

8.1 Technical Details of Implementation

8.3.1 Theory and Calculations

- Power consumed by robot and prospective parts
 - Assuming 150 lb, 1.5 m/s^2 , 1.34 m/s
 - Force = $ma = (68.03 * 1.5) = 102.045 \text{ N}$
 - Power = $(102.045)(1.34) = 136.74 \text{ W}$
- Necessary run time and battery amount
 - $900 \text{ ft} = 0.170455 \text{ miles}$. $0.170455 \text{ miles}/1\text{mph} = 10.2277 \text{ mins}$
 - Run time
 - Components 9.456 A
 - Battery 19 Ah
 - Runtime: $19/9.456 = 2 \text{ h}$
- Wheel diameter and minimum angular velocity
 - wheel diameter = 17 in
- Maximum weight and max weight with load
 - $250 \text{ lb}-300 \text{ lb}$
 - load approximately 20 lb
- Straight away and turn speed
 - we want to finish in under 10 mins
 - $F = c * W$
 - $c = .35$
 - $w = mg$

9. Performance

9.1 Lidar

The lidar works as planned and produces the correct data and distances. This data can be visualized using rviz in ROS. The only issue with the lidar is the boot time. The boot time however is short enough that it is not an issue within the confines of the competition.

9.2 Motors and Feedback Controllers

The motor controller accurately moves the motors at the specified rate. The motor controller also receives commands over serial and will execute the correct command. All data that is sent from the motor controller reaches the computer in the correct time as well. There is only one serious issue with the motor controller. When the motors get too much resistance then the controller shuts down that part of the drive in order to save the motor from pulling too many amps and burning up the motor. This should not be a huge issue in the competition because if the vehicle hits obstacles it is grounds for disqualification.

9.3 Arduino and Mac Mini

The arduino and computer work completely as expected. There has never been an issue with either. They also communicate with each other as programmed.

10. Current Performance

Currently the vehicle will avoid obstacles and travel to a specified waypoint on a given map. The vehicle requires that a map of the surrounding area be created before it can properly navigate to the specified location. The location must also exist on the map for it to navigate to the coordinates correctly. Within the rules of the competition mapping of the competition area is allowed. As long as the map is created correctly then the robot works well and can navigate to its goal without much issue.

11.0 Parts List and Budget

Item	Function	MSRP	Quantity	Cost	Actual Paid	Notes
Sick LMS500	LiDar	5000	1	5000	0	Donated
6061 90 deg Aluminum	Fram	71.92	1	71.92	0	Previous Team
DG-158 with E2-500-315-NE-D-G-1	Motors	450	2	900	900	Optical encoder included
Microsoft Lifecam Studio	Camera	99.95	2	199.9	500	
VPN1513 GPS smart module	GPS	34.99	1	34.99	0	Previous Team
Martin Wheels 958-2TR-I	Wheels	52.29	2	104.58	0	Previous Team
Crossbow	Compass	Unknown	1	Unknown	0	Donated
Mac Mini	Main Computer	1600	1	1503.93	1503.93	
12V Everstart Powersport	Battery	51.88	2	103.76	103.76	
RoboteQ MDC2460	Motor Controller	175	1	175	175	
Back Castor	Back Wheel	5	1	5	5	
RS 232	Cable	10	1	10	10	
USB to DB 15	Cable	5	1	5	5	
Encoder Cable + Transition board	Cable	25.00	1	25.00	25.00	Included with motor controller
TP-link N300 router	Router	24.91	1	24.91	24.91	
CON-LC5	Connector	3.15	2	6.30	6.30	
Total				8294.58	3483.14	