Ville Robotics AutoNav Design Report

Millersville University of Pennsylvania

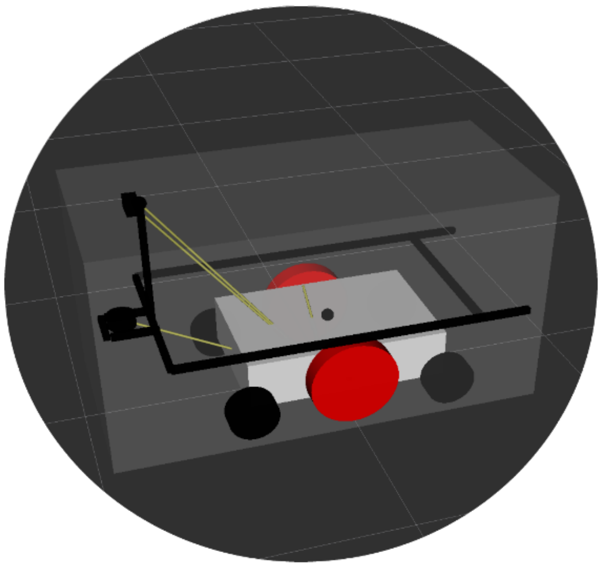

Figure 1. A.Li.E.N. 3.0 URDF Render

Team Captains and Authors

Daniel Haines | drhaines@millersville.edu

Joshua Greineder | jmgreine@millersville.edu

Faculty Advisor Names & Emails

John Haughery Ph.D., CSCE | John.Haughery@millersville.edu

John Wright Jr. Ph.D., CSTM, CLSSGB, CSCE, F.ATMAE | John.Wright@millersville.edu

Team Members' Names & Emails

Jerimiah Buck	jvbuck@millersville.edu	Robert G Kiesel	rgkiesel@millersville.edu
Kevin Constantine	ktconsta@millersville.edu	Joseph Kaskel	jmkaskel@millersville.edu
Joshua Greineder	jmgreine@millersville.edu	Benjamin Wright	brwrigh1@millersville.edu
Daniel Haines	drhaines@millersville.edu		

Faculty Advisor Name, Statement & Signature

Dr. John Wright, PhD | John.Wright@millersville.edu

I certify that the design and engineering of the A.Li.E.N. 3.0 robot has been undertaken by the team members listed above and the efforts have met the demands of a senior-level design course.

Submitted May 15, 2022

Signature: [illegible] Date: 5/14/2022

Table of Contents

INTRODUCTION

The Millersville University Mobile Robotics Team constructed two robots to compete in the IGVC competition of 2022. One of the robots (A.Li.E.N. 2.0) was developed by the majority of the robotics team, while the other robot (A.Li.E.N. 3.0) was developed by the two team captains of A.Li.E.N. 3.0 with the occasional help with mechanical and electrical development of the other members. A.Li.E.N. 3.0 has been developed to improve upon the previous robots developed for IGVC while implementing innovative and robust algorithms to autonomously traverse the entire course at the IGVC competition.

Alien 3.0 is a computer-based robot created with Robot Operating System. The challenge has several major aspects including mapping, localization, path-planning, posing, and waypoint navigation. With the given challenge and time constraints, Robot Operating System (ROS) was chosen. ROS is a framework that contains several pre-built modules which allow our engineering team to focus on innovation and implementation.

ORGANIZATION

The team consists of six members, two of whom took the lead in the research and development of A.Li.E.N. 3.0. The remaining four students aided in aspects such as manufacturing and documentation.

Both of the team captains have previous experience with research and development within their own previous projects, and their skills, passion, and aptitude greatly aided the development of A.Li.E.N. 3.0.

Table 1. Organization Table

Name	Academic Year	Club Position or Development Role	Hours	Contact
Joshua Greineder	Sophomore	Co-Project Manager Public Relations	300+	jmgreine@millersville.edu +1(717) 826-3397
Daniel Haines	Junior	Co-Project Manager	300+	drhaines@millersville.edu +1(717) 456-0326
Robert Kiesel	Senior	Senior Electrical Engineer	5+	rgkiesel@millersville.edu
Kevin Constantine	Senior	Senior Manufacturing Engineer	35+	ktconsta@millersville.edu
Ryan Martin	Senior	President		rcmarti1@millersville.edu
Elizabeth Maschke	Freshman	Treasurer		ejmaschk@millersville.edu
Dennis Nguyen	Freshman	Secretary		denguyen@millersville.edu
Joe Kaskel	Graduate	Controls Engineer	10+	jmkaskel@millersville.edu
Ben Wright	Freshman	Manufacturing Engineer Vice President	30+	brwrigh1@millersville.edu
		Total Hours:	**680+**	

DESIGN ASSUMPTIONS AND PROCESS

Going into the development of Alien 3.0, both team captains had no experience within the Robotic Operating System (ROS) and were expecting a challenge upon the start of creating Alien. To overcome this problem, both students took an online course over spring break, and conducted further research after starting the project more specific to the problems they faced. The students also considered the limited amount of time that was available during the semester until the competition and attentively balanced creating Alien along with their school and work. Both students planned accordingly for the week and progress anticipated before and after every meeting that took place.

Due to the limited amount of time available, the team decided to reuse the base of the chassis from the first version of Alien, this was to eliminate the process of building a new chassis before they were able to start developing the software. This allowed the students to start the development as soon as possible.

Initially, our engineering team began research into preexisting solutions for mapping and obstacle avoidance. We learned about the key algorithms and solutions that are common amongst many autonomous vehicles. We planned multiple phases that included setting up the base system, software modules, firmware, and drivers.

PREVIOUS ROBOTS (A.Li.E.N 1.0/2.0)

Alien was designed with innovation kept at the forefront. The 1.0 model featured a Teensy 3.2 microcontroller, boasting 32 digital I/O pins connecting various sensors and components from throughout the machine. The platform is built over a Pride Mobility Quantum wheelchair generously donated to us by Independent Home Solutions of Lancaster. The ability of the base to support a human-made it an extremely stable foundation for our design. With an OpenMV machine vision device programmed through a Python IDE, and a powerful Swift GPS unit integration was made simple and effective. However, once we got to the competition, we ran into problems with our camera picking up additional elements, and ended up switching to webcams using computer vision, with blurred vision that did not result in as many distractions. Moving forward, the technology of "gaussian blur" was discovered, and Alien 2.0 was born. Moving back to OpenMV cameras, Alien 2.0 was waterproofed and had a new GPS setup using the same teensy microcontrollers. Using gaussian blur, we were able to create distinct shapes with our imaging, greatly increasing the effectiveness of line and object detection. This new design was a clear improvement over the past, but the innovation did not stop there.

A.Li.E.N 3.0 PROCESS

With the goal of building an autonomous robot, we choose to utilize a framework with prebuilt modules. Our goal was to not only complete the competition but to learn advanced techniques for intelligently controlling robots. We began with simple goals and progressively increased the level of challenge. We did not just work towards an end goal, we worked towards many intermediate goals so we could show progress. Although most of the problems we worked through have been solved by many other teams before us, they were new to us. Through designing, piecing together, and customizing our system, we have gained an in-depth understanding of common robotics problems. This includes things like camera distortion matrices, localization, and sensor fusion. By choosing a framework we were able to focus on implementation and high-level problem-solving.The design process has been divided into 6 phases.

Phase 1

This phase primarily focused on setting up the base operating system, the ROS core system, ROS drivers and hardware interfacing. Choosing the version of each major piece was also especially important. ROS1 was specifically chosen over ROS2 for its large amount of documentation and supported packages.

Phase 2

This included ROS driver set up and hardware interfacing. Work also began on configuring the system for reliable operation. All devices were assigned static com ports. In addition, the needed ROS software modules were installed and configured.

Phase 3

Included programming and tuning the drive controller, rotary encoders, and configuring localization. Localization is a fundamental part of the entire system as it gives a reference point for multiple sensors and allows for accurate map creation. To receive odometry information we built rotary encoders. Rather than mechanically installing encoders directly to the shaft of the motor, we approached a solution that was intended to be a “quick fix.” This was to develop an encoder using hall effect sensors outside the wheel and magnets along with the wheel. The problem with this solution was that it only provided 30 ticks per rotation. This turned out to be problematic due to having a low resolution and requiring a longer sample time for accurate measurement. For this reason, we turned to hector SLAM for additional odometry data. Hector SLAM utilizes data based on the point cloud provided by the camera, and laser scan data from the LiDAR module. We then took that information and fused it with an inertial measurement unit and a GPS module, this provided accurate localization data for us to use with mapping and path planning.

Phase 4

During we considered multiple approaches to mapping, lane following, and obstacle avoidance. With the goal of expedience and simplicity, we chose a novel solution. We decided to take the camera data, transform it, convert it, and then merge the data with the laser scan data. By doing this the camera pixel data is interpreted as walls, this prevents the robot from traversing over them and simplifies path planning. To accomplish this, we first had to remove lens distortions from the camera. A camera distortion matrix was calculated and applied. An additional matrix was applied to convert the image to a bird's eye view. This effectively translates the camera image onto the flat 2D ground plane. Everything besides the white pixels is converted to black using a filter. Up to this point the camera data has been in the form of pixels. These pixels are then converted to a point cloud. The point cloud data is then converted to laser scan data which is then merged with the LiDAR scan data. This merged laser scan data is then published to a topic in ROS and to be used for cost map generation.

Phase 5

This is still underway but will include improving the accuracy of localization data and refining sensor calibration. Throughout our work, we have prioritized showing visible progress over refinement and accuracy. We previously noted calibration issues that we will resolve in this stage. After calibration and refinement, we will be able to build an accurate map of the robot's surroundings.

Phase 6

This is a future phase focused on cost map generation, path planning, and motion planning. With an accurate map, the obstacles can be recognized. With obstacles recognized a cost map can be generated. The cost map is in the form of an occupancy grid. An occupancy grid is a collection of cells that have different values or costs. Walls and obstacles have a high cost, cells near obstacles have a small cost. Clear areas have little to no cost. With a cost map generated, multiple paths are considered and the one with the least cost is chosen. With a path planned a series of movement commands are generated with respect to the physical dimensions of the robot. With an awareness of surroundings, physical dimension, and an ability to plan movement the robot will successfully navigate the course at IGVC.

EFFECTIVE INNOVATIONS IN VEHICLE DESIGN

Issues With Encoders

To receive odometry information from the robot we needed encoders, rather than mechanically installing encoders directly to the shaft of the motor, we approached a solution that was intended to be a “quick fix.” This was to develop an encoder using hall effect sensors outside the wheel and magnets along the wheel. The problem with this solution was that it only provided 30 ticks per rotation, which turned out to be problematic. Due to having a low resolution a longer sample time was needed for accurate measurement. Due to this problem, additional localization data was derived from hector SLAM. Hector SLAM utilizes data based on the point cloud provided by the camera, and laser scan data from the LiDAR module. We then took that information and fused it with an inertial measurement unit and a GPS module, this provided accurate localization data for us to use with mapping and path planning.

Camera To Point Cloud 2

For detection of lane lines and potholes, we decided to go the route of utilizing OpenCV to process the image data received from the camera. The image underwent several filters to extract only lane lines and potholes while reducing noise as much as possible. The first filter was HSV, this was used to make color extrapolation easiest due to the separation of color and luminance, and from that, a mask was applied to remove all colors besides the color we wanted to pass through (In this case, white was passed through.) A gaussian blur filter was applied to reduce excess line noise. Following gaussian blur, an edge filter was applied to only return edges of the remaining image data. The only data remaining were the lines and potholes. From this information, we took the two lines on either side and calculated the angle of correction needed which was then sent to the robot's velocity command.

From that algorithm, we decided there were too many future integration issues. We then decided to go the route of taking the image and converting it to a point cloud to publish to a topic in ROS and to be used for cost map generation. To do this the camera uses a matrix calculated specifically for the camera we decided to use to convert the image to a bird's eye view, which allows depth to be pulled. Everything besides the white points desired are converted to black using the filter in the previous method pursued, and the remaining points are passed through a loop converting each cartesian coordinate to a position in the point cloud with a positional measurement (XYZ) in meters. The objects within the point cloud are interpreted as walls preventing the robot from traversing over them while planning the path.

DESCRIPTION OF MECHANICAL DESIGN

Weatherproofing

The previous year our team attended IGVC (2019), the team used a raincoat and tape to waterproof A.Li.E.N. 1.0. Since that approach was not robust and did not utilize an innovative concept or design, we have decided to use a pelican case to contain all non-waterproof components.

Wheelchair Base

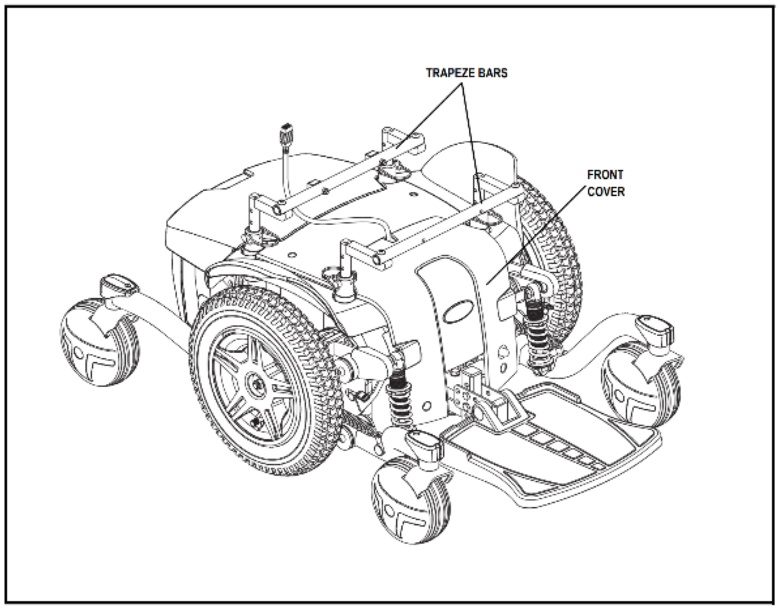

Figure 2. The Pride Mobility Quantum. 614[3]

Alien is built on a used Pride Mobility Quantum 614 wheelchair donated by Independent Home Solutions of Lancaster, Pennsylvania (As pictured in Figure 2). This was previously used by A.Li.E.N. 1.0, so we were able to use it without a use of time and money. We also decided to keep the base because of its load capacity of 300lbs and the limited velocity of just under 5mph, preventing us from surpassing the maximum speed for the competition.

Frame

The frame of the robot used to mount the sensors and other hardware was built with 80/20 T-slotted aluminum framing and sheets of polycarbonate were used to create a solid plane. These two materials were helpful in ease of mounting hardware as well as modification and dynamic development.

DESCRIPTION OF ELECTRINIC AND POWER DESIGN

Hardware Choices

Master Computer

We decided to go forward with using a MacBook Flashed with Debian Buster 10 because it was actively available to us at the start of our development, which aligned the best with our time restraint for development. The device contains a Intel Core i7-4850HQ CPU @2.30GHz x8 and has 16Gb of RAM.

Camera

For the camera, the Avater HD Webcam with 1080p and 110 FOV was chosen. The device allowed us a wide range of view to pull as much data from the robot's surroundings as possible. It does this at 30 frames per second which is high enough to retrieve live frames to send accurate object location to the robot. This webcam was easily interfaced with OpenCV as no drivers were needed and camera acted as plug and play imaging device. In addition to those benefits this camera has 1080p HD for great resolution of images to be processed and is loaded with auto brightness adjustment, preventing the issue of having as much glare reflected off the ground as possible.

LiDAR

The LiDAR device decided on was the RPLiDAR A1M8. This device was purchased due to its already developed Robotic Operating System (ROS) package, allowing the retrieval of data with A.Li.E.N. 3.0 to be as easy as possible. There was also an excess of documentation with integrating this module with ROS.

Global Positioning System (GPS)

We decided to go with a generic GPS module because of its plug and play compatibility, as well as the U-BLOX 7 Chipset used by the module allowing us to use a package designed for reading U-BLOX binary protocol.

Inertial Measurement Unit (IMU)

The IMU we decided to use for A.Li.E.N. 3.0 was the BNO055, this was chosen because of the excess of documentation and resources on this specific module. From our research, this module is an accurate solution to provide the positional data we need.

Encoder

This was custom built with hall effect sensors and an inverter chip. These components were decided on because they were available to use at the moment we needed them, and it was supposed to be a quick fix so that was the most appealing option.

Microcontroller

We needed a way to interface with the motors as well as receive information from the encoders. This was done through the use of the Teensy 3.2, the device was chosen for processing speed of 72 MHz. It was also easily available to us and we had previous experience with the microcontroller.

Motor Controllers: We decided to go with Jaguar motor controllers because they are full of capability and were interfaceable with pulse width modulation, which is what we are using to control the speed of our motors. The team also had a few which had been left over from previous builds allowing us sooner development time.

Data System

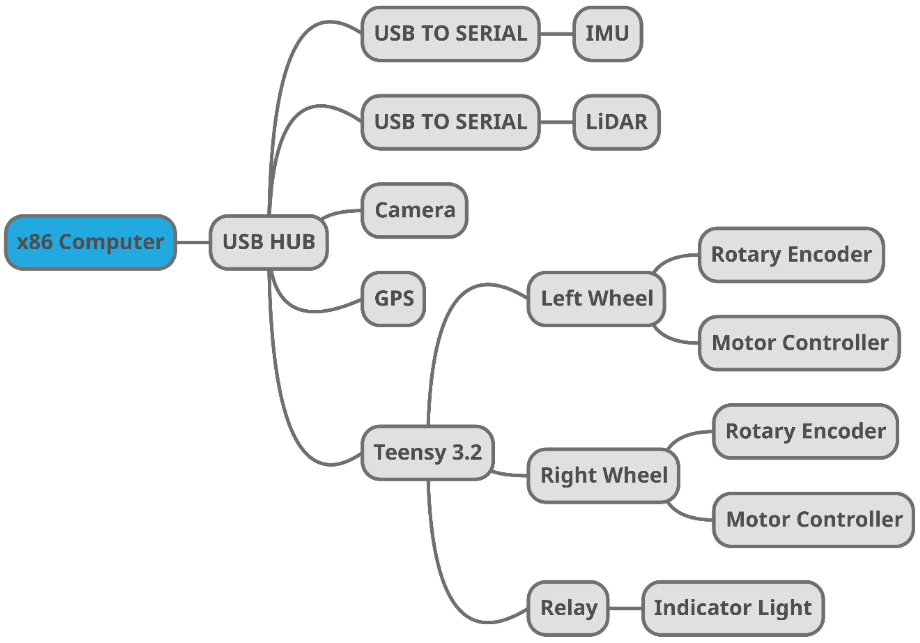

Figure 3. Data Communications Diagram

Power System

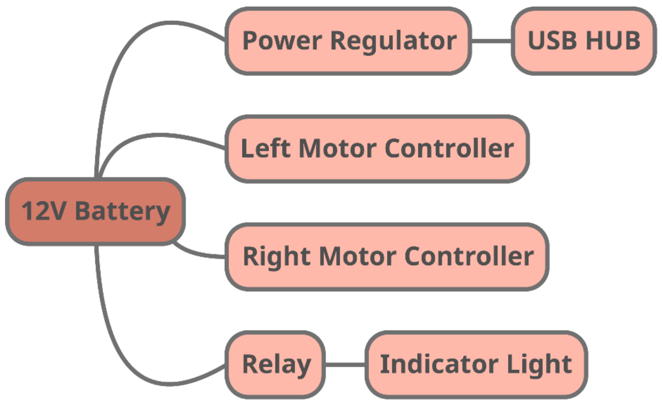

Figure 4. Electrical Diagram

DESCRIPTION OF SOFTWARE STRATEGY AND MAPPING TECHNIQUES

[redacted]

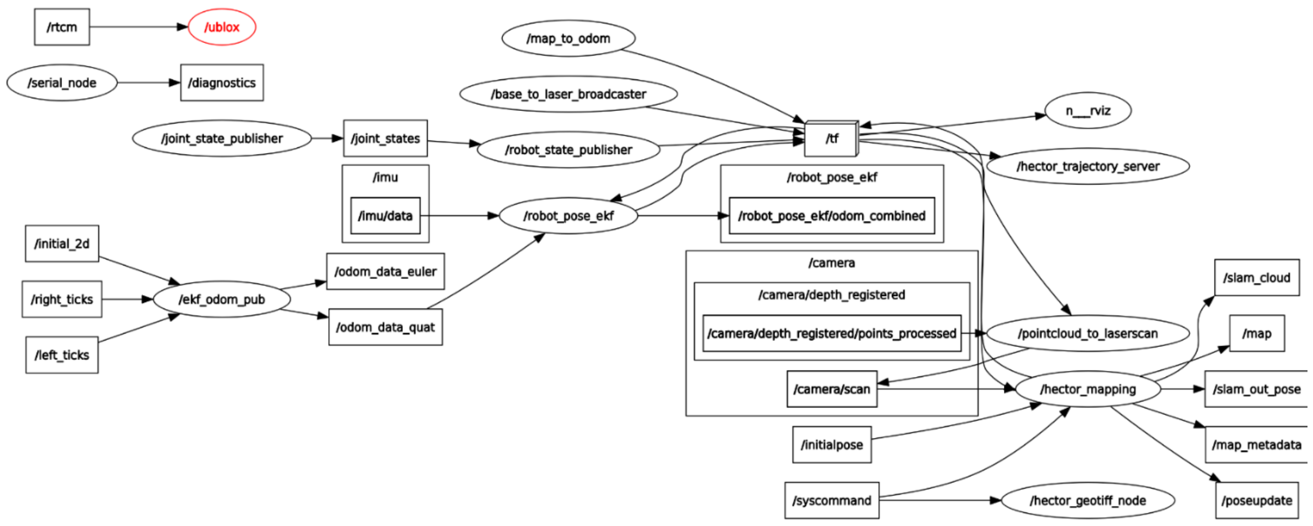

Figure 5. System Wide Computational Graph

SIMULATIONS EMPLOYED

Rviz

While we did not necessarily have a simulated environment, we utilized RVIZ to visualize all the environmental data and positional robot data. This was to ensure accurate information was being published by individual sensors and accelerated troubleshooting while the development was being undergone, as well as a means to demonstrate what the robot was doing to others.

PERFORMANCE TESTING TO DATE

At our current point, we have just implemented GPS navigation, with that integrated with the rest of the system we only need to implement object avoidance/path planning algorithms. The original plan was for A.Li.E.N. 3.0 to compete in IGVC 2023 because the Robotic Operating System (ROS) was new to everyone building the robot. However, since the team made more progress than expected, they made the decision to register A.Li.E.N. 3.0.

Initial performance assessments

ROS1 – ROS2

We decided to create this robot using Robot Operating System 1(ROS1) instead of ROS2 due to the amount of documentation and project resources available to us on the internet, however, in our future development, we would like to use ROS2 because as we have discovered, the documentation is mostly out of date, and since ROS2 is newer and still being developed we theorize it being a better more permanent solution.

Chassis

The chassis we are using is not the most efficient chassis, in the future we would like to use a smaller rounder robot, allowing it to navigate through obstacles easier.

Members Allocated

This project has been a huge learning experience. We have had to learn a great deal of information, and still have a lot to learn. However, if we were to have more people working alongside us then we would be able to get more things done, allowing us to make a more efficient and precise robot while meeting the deadline. Due to this we want to get more people educated in ROS to assist us in the future.

REFERENCES

Not alphabetical, goes in order in which these were presented in the paper. i.e. 1-10

https://www.pridemobility.com/pdf/owners_manuals/us_jazzy/us_quantum_614_om.pdf

Addison A 2021. How to set up the Ros Navigation Stack on a robot *Automatic Addison*

Carroll M, Foote T, Khandelwal P, O'Quin J, Perko E, Purvis M, Rockey C, Tossell K and Macenski S 2019 Ros-geographic-info/geographic_info: Ros packages for geographic information *GitHub*

DiCola T BNO055 absolute orientation sensor with Raspberry Pi & Beaglebone Black *Adafruit Learning System*

Ferguson M 2020. Robot & Chisel *Outdoor Global Localization · Robot & Chisel*

Kumar V Kumarrobotics/ublox: A driver for ublox GPS *GitHub*

Moore T 2020. robot_localization *ros.org*

Renard E 2022. Learn Robot Operating System (ROS) for robust & Scalable Robot Apps *Udemy*

Ryeng R 1963. Remove spurious small islands of noise in an image - python opencv *Stack Overflow*

Shaoul Y 2022. A full autonomous stack, a tutorial: ROS + raspberry pi + arduino + slam *Yorai*

Zhang X, Lai J, Xu D, Li H and Fu M 2020. 2D lidar-based slam and path planning for indoor rescue using Mobile Robots *Journal of Advanced Transportation*