AUTONOMOUS PACKAGE DELIVERY ROBOT

Intelligent Path Following with ROS and LiDAR

INTRODUCTION

This goal of the Autonomous Package Delivery Robot (APDR) is to automate campus package deliveries. The key design aspects of this project include a web user interface and a variety of sensors to navigate between locations. These include IMU, LiDAR, and GPS. The mapping and navigation algorithms are provided by the Robot Operating System (ROS). The frame, motors, and batteries are recycled from an old electric wheelchair.

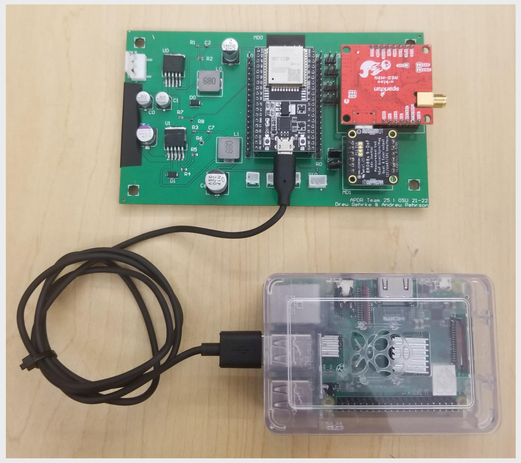
DESIGN REQUIREMENTS

- **Path Following:** The system will follow a predefined path between an origin and destination.
- **Object Reaction:** The system will safely traverse around stationary objects in its path.
- **Lock Box:** Packages will be transported in a secure container that can be unlocked by an authorized user.
- **Data streamed to Website:** The admin web interface will display a live feed of information about the robot's status.
- **Data steamed from Website:** Notifications about robot and delivery status will be sent to administrators and clients through the website.
- **Emergency Stop:** The system will shut down within 500ms of the emergency button or collision sensor activating.
- **Edge Detection:** The system will determine the bounds of pathways and maintain a safe distance from the edges.
- **Battery Monitoring:** The system will measure the voltage of the onboard batteries within an accuracy of 100mV.

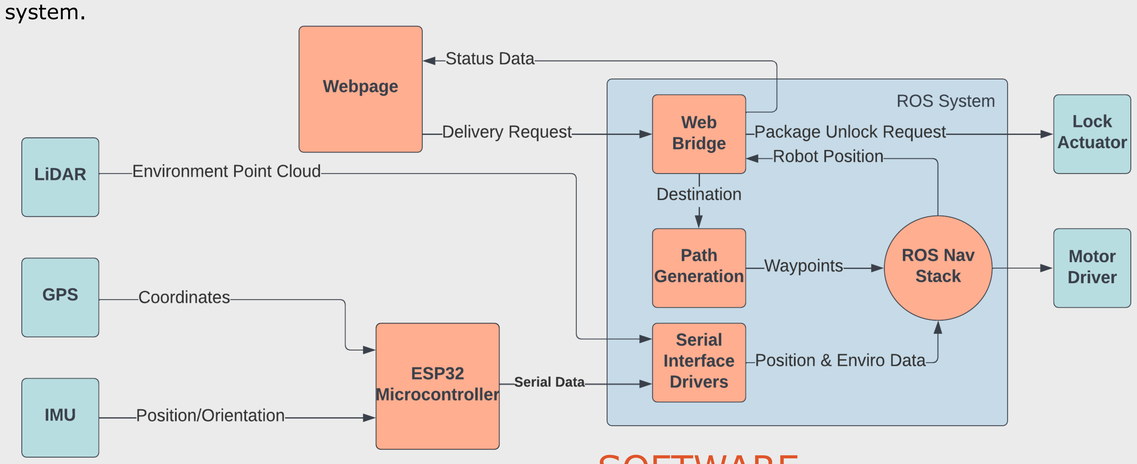
Oregon State University

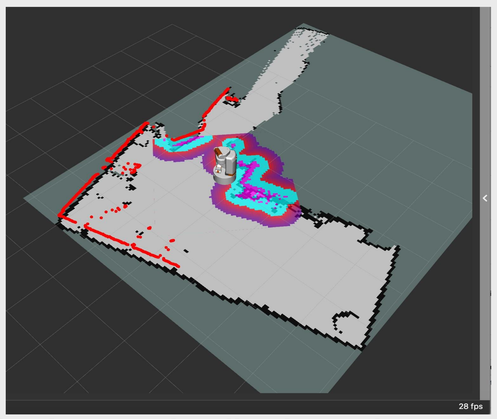
HARDWARE

- **GPS:** Global Position System. Latitude and longitude of robot's current position. Used to navigate between waypoints.
- **IMU:** Inertial Measurement Unit. Measures accelerations in 6 degrees. Used to give the robot's quaternion orientation.
- **LiDAR:** Light Detection and Ranging. Used to remotely detect solid objects within the surroundings of the robot.
- **ESP32:** Microcontroller used as the main interface between the ROS2 system and the hardware. All firmware was written in C++.
- **R. Pi:** Raspberry Pi 4. Responsible for all ROS functionality and web interfacing. Serves as the core computational hardware for the system.



PCB with ESP32, GPS, and IMU connected to R. Pi





Simultaneous localization and mapping (SLAM) with costmap generation

SOFTWARE

- **WEB INTERFACE:** The site is hosted on Amazon webservers and uses Amazon Amplify to process users for login.
- **WEB BACKEND:** The site is using React and node.js with the Web Bridge to process data from the system to the site or from the site to the system using WebSockets.
- **ROS NAVIGATION:** Provides interface between sensor hardware, real world localization and mapping algorithms. Used to produce and generate paths between origin and destination positions for the robot.
- **SENSOR INTERFACE:** Utilizing the ESP32, many hardware interfaces were serialized and transmitted to ROS on the Raspberry Pi. GPS data and inertial measurement data were interpreted by through this program

APDR TEAM MEMBERS

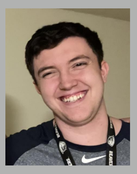
Tyrone Stagner
(stagnert@oregonstate.edu)

Throughout this project Tyrone oversaw building the website, integrating ROS Bridge Suite with the website, setting up ROS2 on the Raspberry Pi 4, and creating the GitHub and Wiki page.

Nathan Searles
(searlesn@oregonstate.edu)

Nathan was responsible for the implementation of optical sensors and the configuration of the ROS Navigation stack. This included setting up a network of multiple python nodes to perform a SLAM algorithm.

Drew Gehrke
(gehrkean@oregonstate.edu)

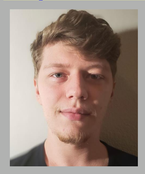


Drew was responsible for the lock box, sensor inputs, and PCB design. Sensors included the GPS and IMU. The PCB, designed in Altium Designer, is the interconnect for the ESP32 and sensors.

Nicholas McBee
(mcbeen@oregonstate.edu)

Nick headed up the power management sub-system for this project. Nick also configured the ROS Navigation stack and created a Python script for sending waypoints.

Andrew Pehrson
https://Pehrsona.com



Throughout this project Andrew oversaw integrating the hardware peripherals, like the ESP32 and motor controller, into ROS2. He also designed and fabricated the enclosure.

OUR PROJECT SHOWCASE