Department of Computer Science & Engineering
The University of Texas at Arlington

Detailed Design Specification
CSE 4317: Senior Design II
Spring 2019

IGVC
AutoMav

Dario Ugalde
Awet Tesfamariam
Andrew Break
Amgad Alamin
Warren Smith

Revision History

Revision	Date	Author(s)	Description
0.1	02.25.2019	DU, AT, AB, AA, WS	initial draft
1.0	05.10.2019	DU, AT, AB, AA, WS	update all sections with new implementation information

CONTENTS

List of Figures

List of Tables

1 Introduction

AutoMav is an autonomous ground vehicle that will compete in the IGVC 2019 competition. AutoMav is designed to complete a lane designated course, while avoiding obstacles and navigating to way point locations. AutoMav is a modular system, with nodes being created for each subsystem, so that future teams will be able to easily extend and maintain the system.

The purpose of this document is to provide specific details regarding the implementation of the system architecture. The AutoMav system architecture is defined in the Architectural Design Specification (ADS) document. The implementation of this system meets the requirements defined in the System Requirements Specification (SRS) document. This document provides information on elements of the hardware and software of the AutoMav product. Hardware elements such as components and connections as well as software elements such as programming languages, data structures and software dependencies.

2 System Overview

The AutoMav system is separated into six modules, Central Control, External Sensors, Internal Sensors, Hardware, Navigation, and Computer Vision. Each module is responsible for a separate group of core functions within the system.

The central control unit will provide a location for system status information, as well as handling system wide commands, and a platform for all other nodes to interact with the system. External Sensors are devices that sense information about the environment. However, they are not part of the system itself. The vehicle will use a 3D camera and GPS as external sensors. The Internal Sensors consists of the Inertial Measurement Unit (IMU) to collect odometry data. The hardware subsystem deals with the components of the system that handle information exchange at the physical level. This includes any necessary signal processing, electronics, and motion controls. The navigation subsystem will be responsible for creating a path from its current location to the intended location based on the information from the external sensors and the computer vision systems. The navigation subsystem is split into two modules: Path Finding and SLAM. The computer vision subsystem is responsible for using images captured by external sensors to recognize obstructions in the vehicle's path as well as recognizing painted lane markers. Obstacle recognition will utilize a 3D camera that provides depth information to detect objects in front of the vehicle. Lane recognition will use the 2D component of the 3D camera data to search for edges in images containing painted lanes and painted potholes. The output of both computer vision nodes are used to construct a map of the vehicle's surroundings which is then provided to the path-finding subsystem.

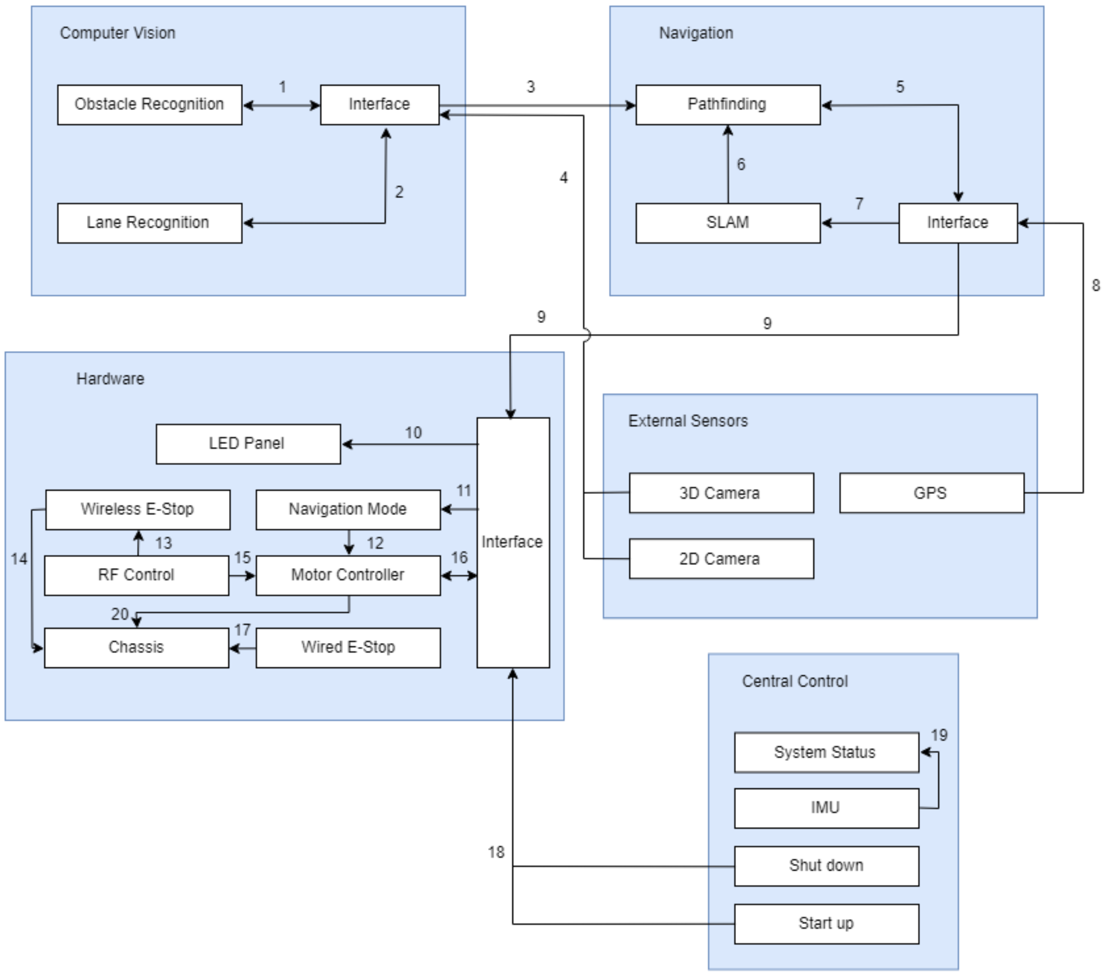


Figure 1: System architecture

3 Central Control

3.1 Layer Hardware

The central control layer will exist on an Nvidia Jetson TX2 embedded computer system. The Jetson TX2 will support software that will facilitate connectivity and data flow between central control modules. The layer also utilizes the Varience Inertial Measurement Unit (IMU) that will send data to the Jetson TX2.

3.2 Layer Operating System

The central control layer will utilize Ubuntu version 16.04, running on the Jetson TX2 to support software operations.

3.3 Layer Software Dependencies

The central control layer will utilize the Robot Operating System (ROS) version "Kinetic Kame", a subsystem infrastructure software and file system that will support data flow and feature implementation for the layer subsystems.

3.4 System Status

The system status subsystem is a high level monitoring process that provides the user with information about the current state of the vehicle on all layers.

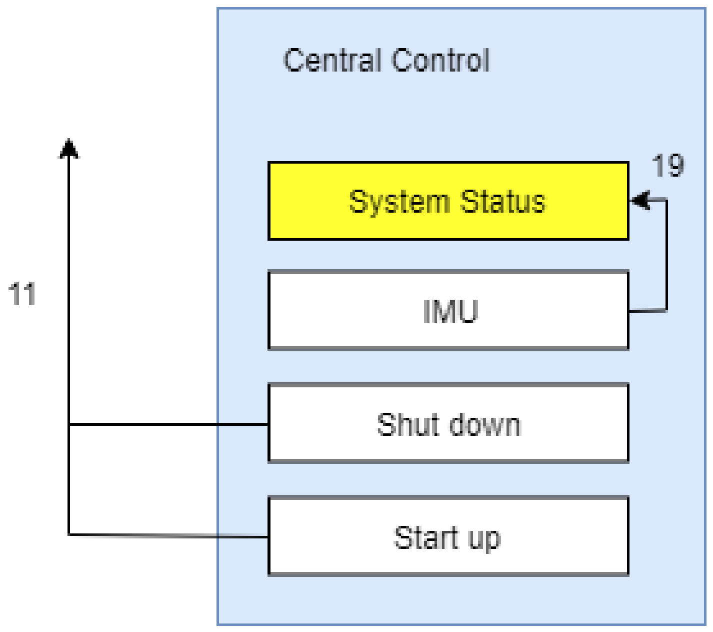


Figure 2: Central Control - System Status

3.4.1 Subsystem Hardware

The system status process will utilize the Jetson TX2 to run and retrieve data from connected components.

3.4.2 Subsystem Operating System

The system status process will run within the Ubuntu operating system.

3.4.3 Subsystem Software Dependencies

The system status process will utilize ROS to organize data flow from system components and software.

3.4.4 Subsystem Programming Languages

C++

3.5 IMU

The IMU subsystem is a hardware component that will report readings regarding the vehicle's angular velocity and linear acceleration.

3.5.1 Subsystem Hardware

The IMU subsytem will utilize the Variense IMU/AHRS Plug and Play Module. The unit features a 3-Axis Digital Accelerometer, 3 Axis Rate Gyroscope and a 3-Axis Magnetometer

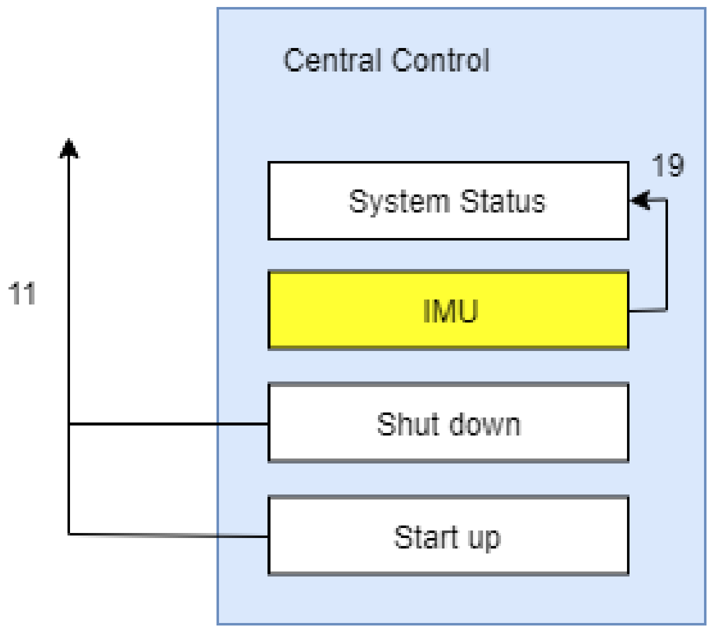


Figure 3: Central Control - IMU

3.5.2 Subsystem Data Structures

The IMU subsystem will transmit data via USB connection. The incoming data can be of several types as described in the following table:

Table 2: System Status interfaces

Name	Format	Length	Value
Accelerometers, gyroscopes, magnetometers, Euler angles			
Timestamp	Uint32	4 Bytes	(Big Endian)
x	float	4 Bytes	(Big Endian)
y	float	4 Bytes	(Big Endian)
z	float	4 Bytes	(Big Endian)
Quarternions			
Timestamp	Uint32	4 Bytes	(Big Endian)
w	float	4 Bytes	(Big Endian)
x	float	4 Bytes	(Big Endian)
y	float	4 Bytes	(Big Endian)
z	float	4 Bytes	(Big Endian)
Heading			
Timestamp	Uint32	4 Bytes	(Big Endian)
heading	float	4 Bytes	(Big Endian)

3.6 Shut Down

The shut down subsystem is a process that will safely power down all vehicle layers.

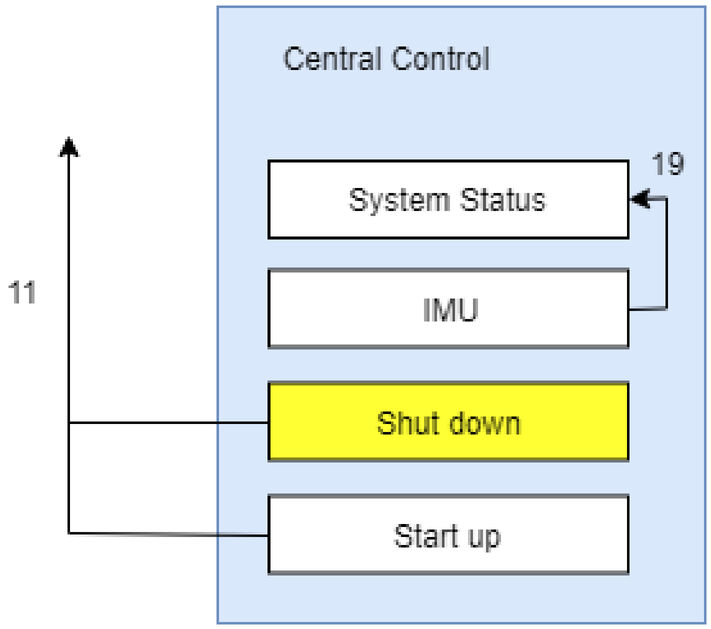


Figure 4: Central Control - Shut Down

3.6.1 Subsystem Software Dependencies

The shutdown process will utilize ROS to instruct the shut down of vehicle modules.

3.6.2 Subsystem Programming Languages

C++

3.7 Start Up

The start up subsystem is a process that will initiate all vehicle subsystems and verify that they are responding properly.

3.7.1 Subsystem Software Dependencies

The start up process will utilize ROS to initiate all vehicle subsystems.

3.7.2 Subsystem Programming Languages

C++

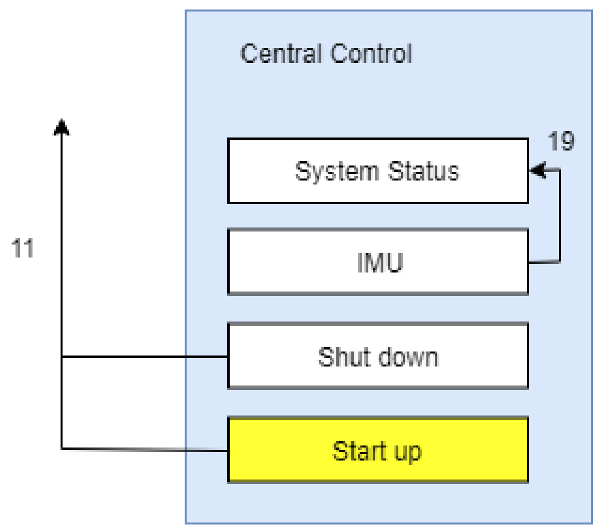


Figure 5: Central Control - Start Up

4 External Sensors

4.1 Layer Hardware

Physical components in this layer include the Intel Real Sense 3D camera and the GPS module.

4.2 GPS

This piece of hardware is used to feed in a coordinate location that will be used by system to determine its current location.

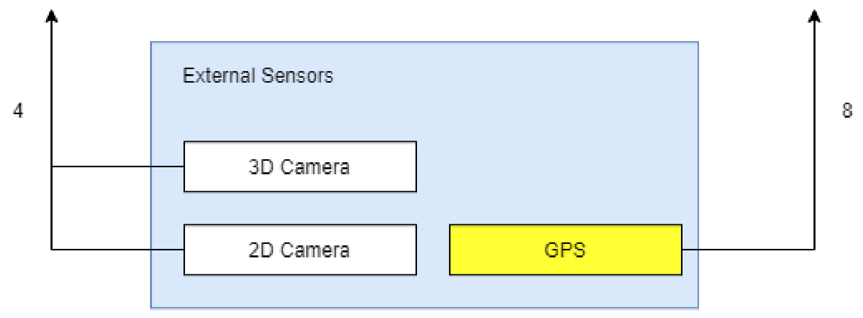


Figure 6: External Sensors - GPS

4.2.1 Subsystem Hardware

The physical GPS sensor.

4.3 3D Camera

The Intel Real Sense is a piece of hardware that will provide images of the system's surroundings to determine the proximity of objects in the system's environment.

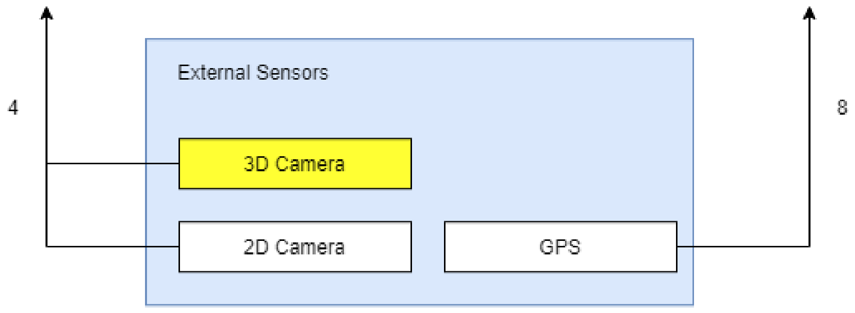


Figure 7: External Sensors - 3D Camera

4.3.1 Subsystem Hardware

The Intel Real Sense camera that uses a 3D point cloud to build images of the system's surroundings.

4.3.2 Subsystem Software Dependencies

Using ROS 3D image processing libraries to make use of this hardware.
Using BaseCam Electronics SimpleBCG GUI version 2.2b2 to calibrate gimbal mount.

4.3.3 Subsystem Data Processing

The point cloud is processed by the Rtabmap ROS package.

4.4 2D Camera

The Intel Real Sense camera collects 2D camera data, which is used to detect 2D features of the system's surroundings such as lane markings.

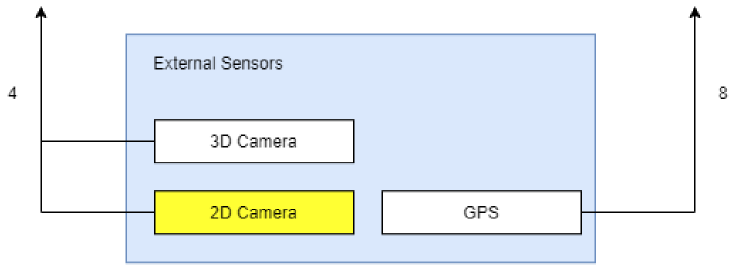


Figure 8: External Sensors - 2D Camera

4.4.1 Subsystem Hardware

The Intel Real Sense 3D camera

4.4.2 Subsystem Data Processing

Camera data will be processed using the ROS packages for the OpenCV library.

5 Hardware

5.1 Layer Hardware

This layer encompasses physical components of the overall system including, the E-Stops, the Navigation Mode switch, the motor controller, the chassis.

5.2 RF Control

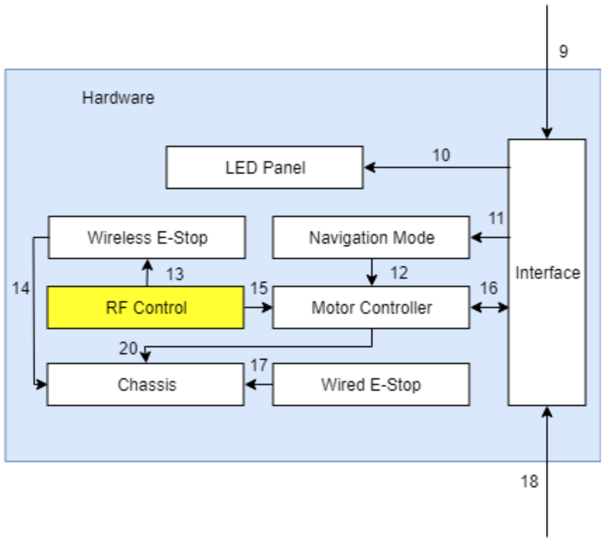


Figure 9: Hardware - RF Control

5.2.1 Subsystem Hardware

This component will take wireless control signals that will be used to operate the system while in manual mode.

5.2.2 Subsystem Software Dependencies

Manual control inputs are handled by the teleop-twist-keyboard ROS package. Wireless control is currently handled over SSH through an on-board router.

5.2.3 Subsystem Programming Languages

C++

5.3 Navigation Mode

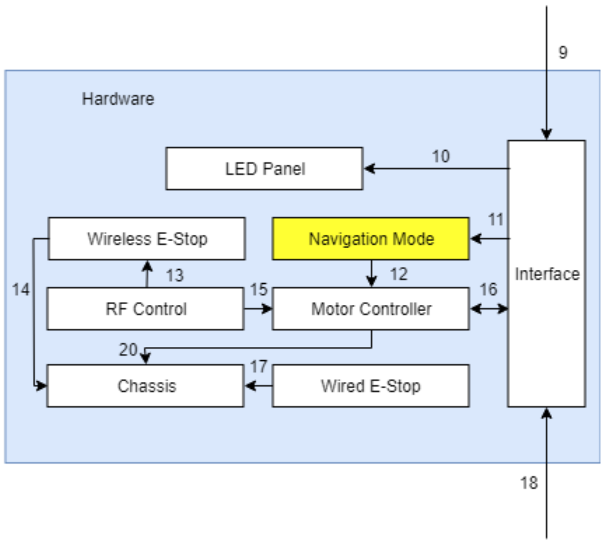


Figure 10: Hardware - Navigation Mode

5.3.1 Subsystem Hardware

A physical switch that will need to be enabled to all the system to enter Navigation state.

5.3.2 Subsystem Programming Languages

C++

5.4 Motor Controller

This component uses inputs from the TX2 to send control signals to the motors connected to the wheels to drive the system.

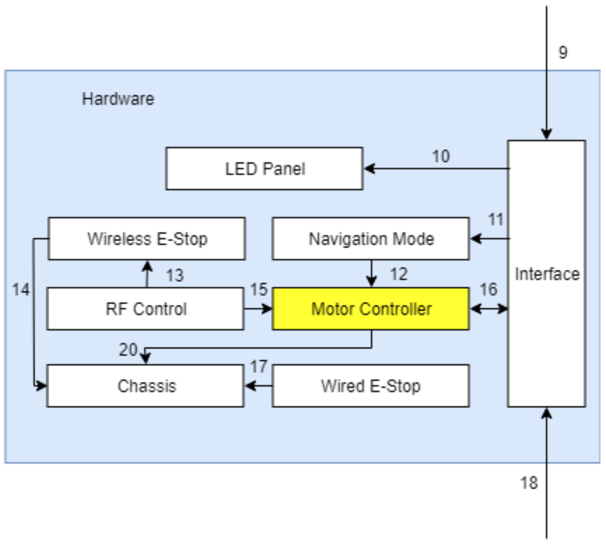


Figure 11: Hardware - Motor controller

5.4.1 Subsystem Hardware

RobotEq DC Motor Controller to send control signals to the system motors.

5.4.2 Subsystem Software Dependencies

A custom ROS node sends commands to the motor controller based on input commands from the manual control node.

5.4.3 Subsystem Programming Languages

C++

5.5 LED Panel

This component is an array of LEDs that will communicate to people around the system the current status of the system, such as if it is current in Navigation Mode or Manual Mode.

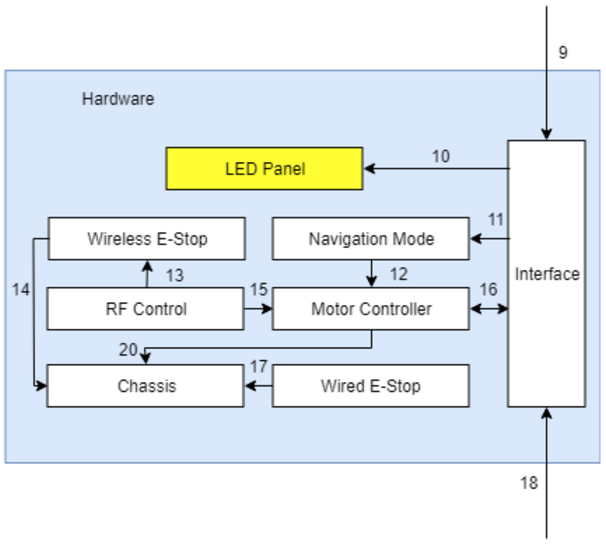


Figure 12: Hardware - LED Panel

5.5.1 Subsystem Hardware

LED array which should be bright enough to be seen clearly during the day.

5.5.2 Subsystem Programming Languages

C++

5.6 Wired E-Stop

A physical button on the chassis that will disconnect the entire system from power.

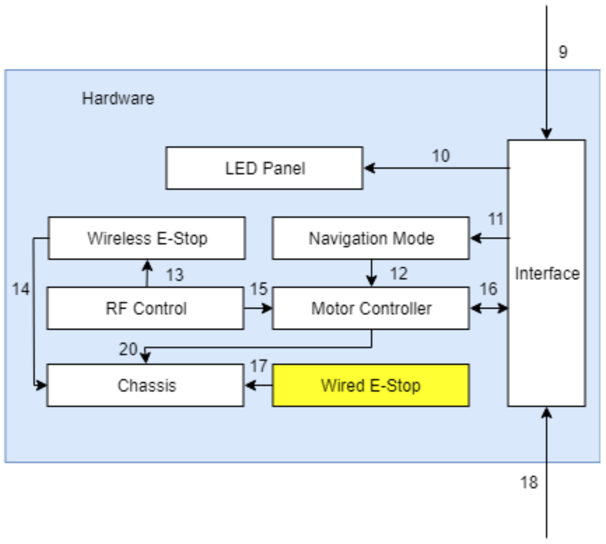


Figure 13: Hardware - Wired E-Stop

5.6.1 Subsystem Hardware

A physical switch connected directly to input voltage from the batteries.

5.7 Chassis

An electric wheelchair base with separated payload and component housings.

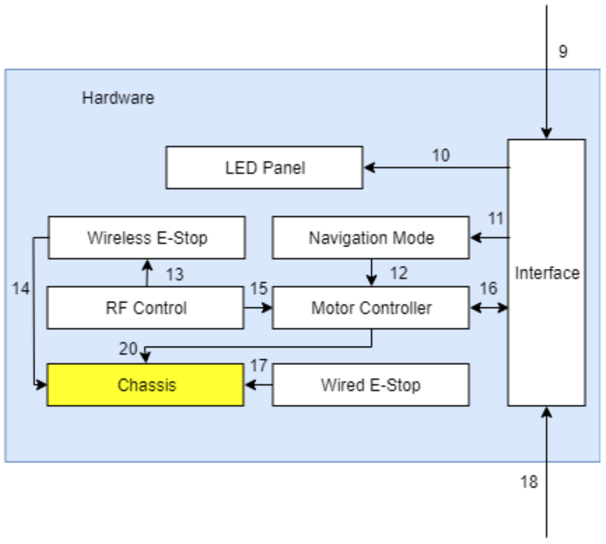


Figure 14: Hardware - Chassis

5.7.1 Subsystem Hardware

The chassis frame is constructed from 1 inch guided aluminum rods and the outside of the chassis made of clear acrylic.

5.8 Interface

Physical connections to other systems used send data between components.

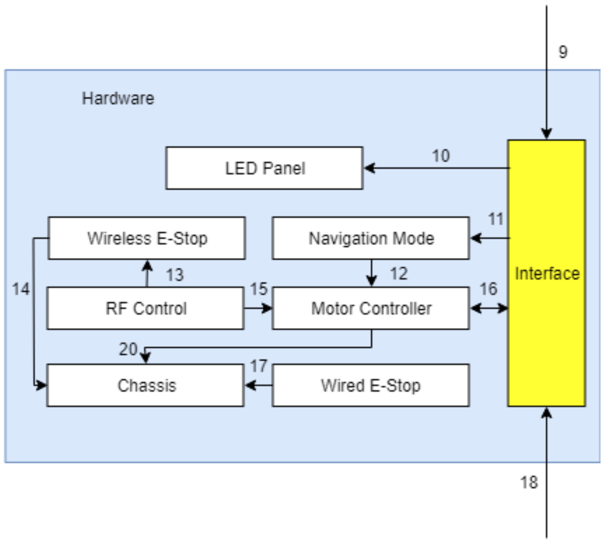


Figure 15: Hardware - Interface

5.8.1 Subsystem Hardware

Consists of the physical connections between components within the subsystems.

6 Navigation

The Navigation layer will utilize information provided by the computer vision and external sensors layers to construct a viable path for the vehicle to reach its destination. The Navigation layer will construct and update a path from its current GPS location to a destination in GPS coordinates. The layer will map out obstacles reported by the computer vision layer and will modify the path to navigate around them. This will be accomplished utilizing the Robot Operating System (ROS) and accompanying path-finding and movement control libraries.

6.1 Layer Software Dependencies

The navigation layer will utilize ROS to facilitate data flow between layers and subsystems.

6.2 SLAM

The simultaneous localization and mapping (SLAM) subsystem is a process that will utilize data from the camera about surrounding obstacles and update a map of the vehicle's surroundings.

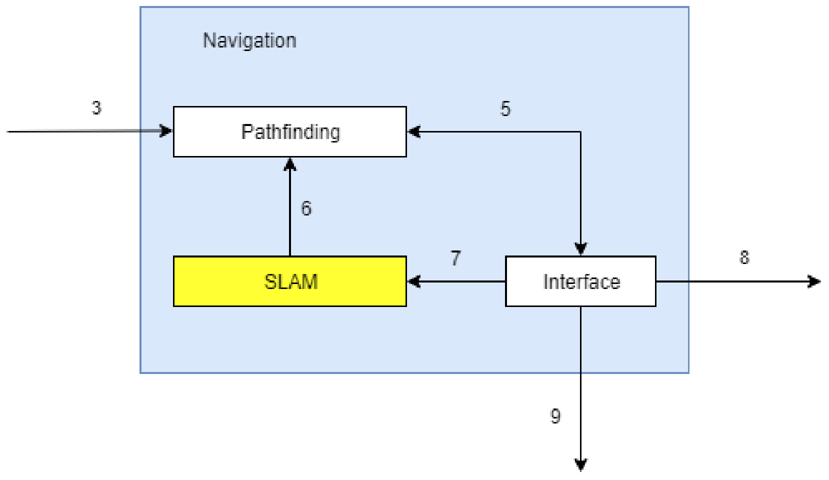


Figure 16: Navigation - SLAM

6.2.1 Subsystem Software Dependencies

The SLAM subsystem will use the ROS Rtabmap package to construct an occupancy grid based on 3D image data.

6.2.2 Subsystem Programming Languages

C++

6.3 Pathfinding

The pathfinding subsystem is a process that will utilize data from the SLAM and GPS unit to calculate a path for the vehicle to follow in order to reach its destination. Autonomous movement will be executed according to this path.

6.3.1 Subsystem Software Dependencies

The pathfinding subsystem will utilize the ROS MoveBase package along with the map generated by the Rtabmap package.

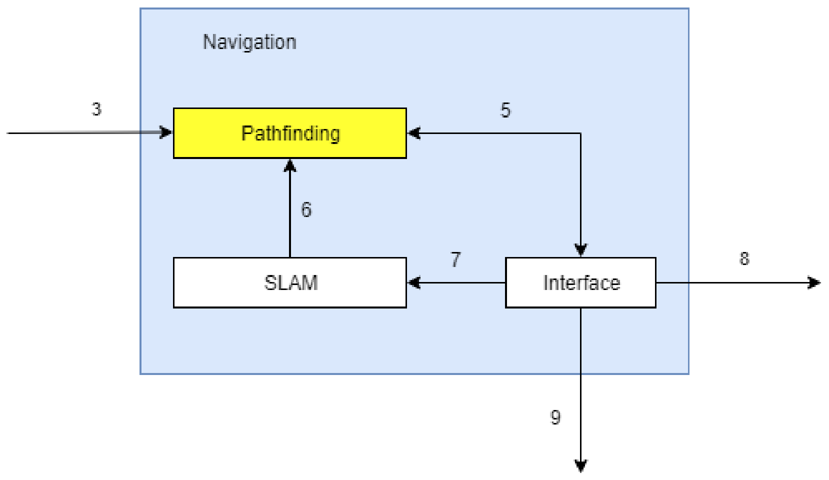


Figure 17: Navigation - Pathfinding

6.3.2 Subsystem Programming Languages

C++

6.4 Interface

The navigation interface subsystem is a process that facilitates data flow from other layers of the system to the navigation subsystems.

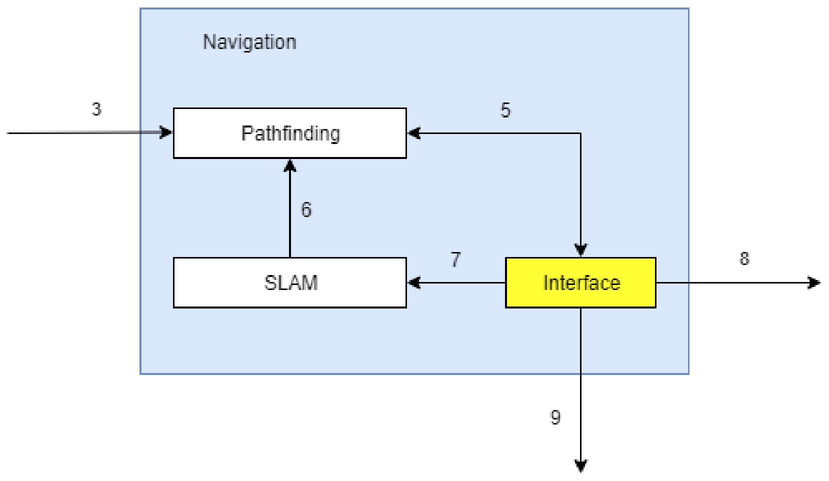


Figure 18: Navigation - Interface

6.4.1 Subsystem Software Dependencies

The navigation interface will be supported by the ROS data flow infrastructure.

6.4.2 Subsystem Programming Languages

C++

7 Computer Vision

The computer vision layer will utilize 2D and 3D camera data to accomplish object recognition and lane detection.

7.1 Layer Hardware

The computer vision layer will take in data collected by the Intel RealSense 3D camera. Computer vision image processing will use the Nvidia Pascal GPU included with the Nvidia Jetson TX2.

7.2 Layer Software Dependencies

The computer vision subsystem will use the OpenCV packages for ROS to execute image processing.

7.3 Obstacle Recognition

The obstacle recognition subsystem utilizes the 3D point cloud data from the 3D camera to determine spaces that are not obstacles and continuously builds an occupancy grid of its surroundings.

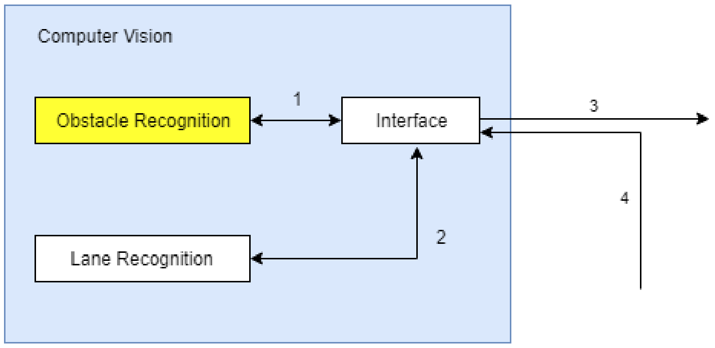


Figure 19: Computer Vision - Obstacle Recognition

7.3.1 Subsystem Software Dependencies

The obstacle recognition subsystem utilizes the rtabmap ROS package to generate a map of obstacles using 3D camera data.

7.3.2 Subsystem Programming Languages

C++

7.4 Lane Recognition

The lane recognition subsystem use computer vision image processing to detect edges that are defined by painted lane markings of the course.

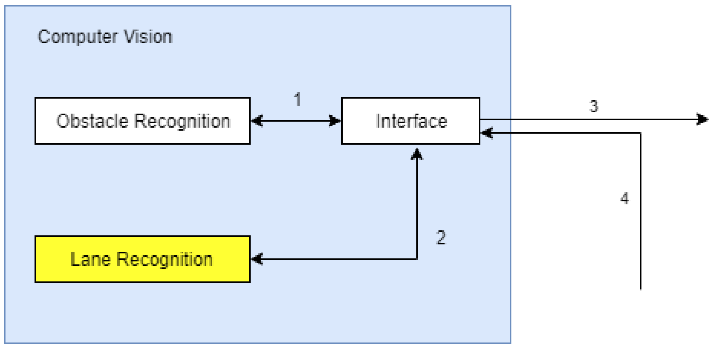


Figure 20: Computer Vision - Lane Recognition

7.4.1 Subsystem Software Dependencies

The lane recognition subsystem uses the ROS OpenCV packages for image processing.

7.4.2 Subsystem Programming Languages

C++

7.5 Interface

The interface subsystem is a process that facilitates the handling of data from the camera sensors and the output of the computer vision layer to other layers.

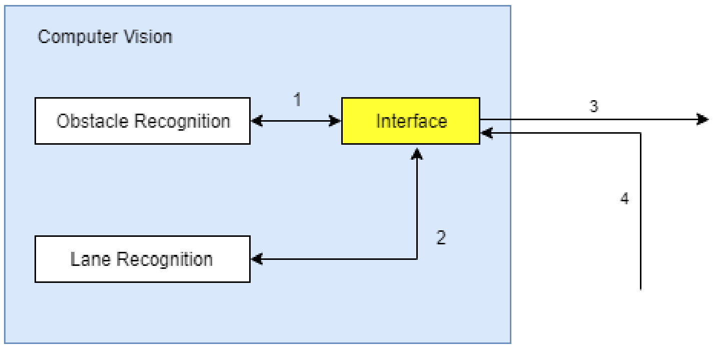


Figure 21: Computer Vision - Interface

7.5.1 Subsystem Software Dependencies

The computer vision interface will be supported by the ROS data flow infrastructure.

7.5.2 Subsystem Programming Languages

C++

8 Appendix A

8.1 CAD Designs

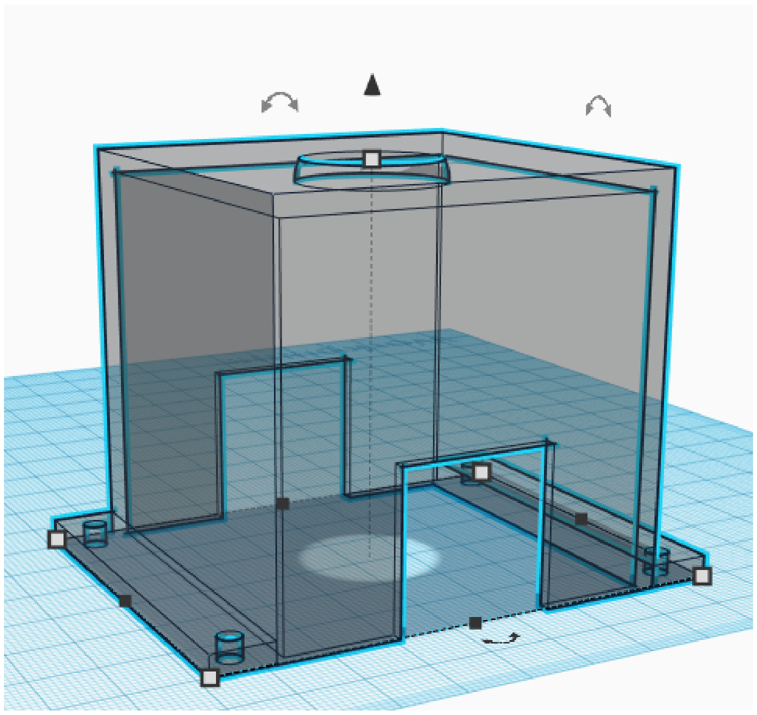
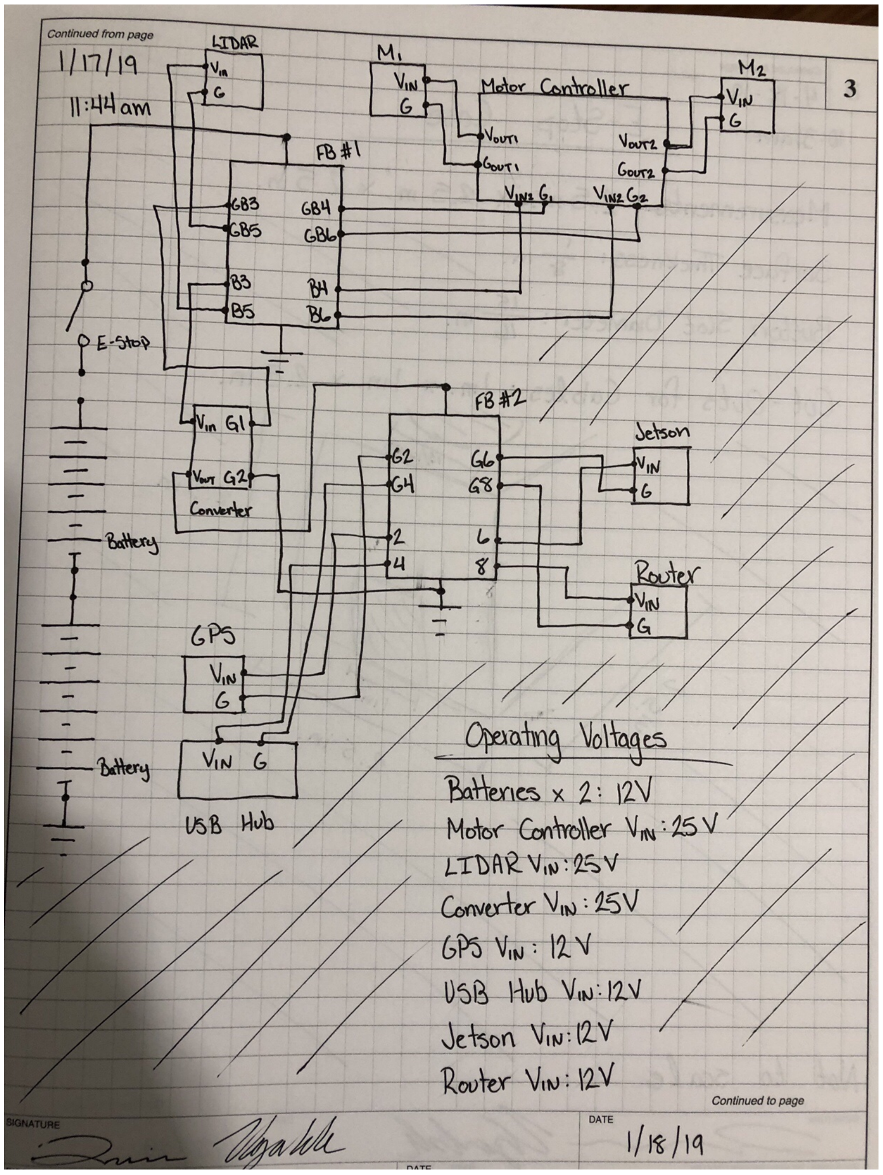


Figure 22: E-Stop CAD design

8.2 Wiring Diagrams



Operating Voltages

Batteries x 2: 12V

Motor Controller V_{IN}: 25 V

LIDAR V_{IN}: 25V

Converter V_{IN}: 25V

GPS V_{IN}: 12 V

USB Hub V_{IN}: 12V

Jetson V_{IN}: 12V

Router V_{IN}: 12V

Figure 23: AutoMav Wiring Diagram

References

[1] J. Lane and A. Kosinski, "The 27th Annual Intelligent Vehicle Competition (IGVC) and Self-Drive: Official Competition Details, Rules and Format," IGVC, 26 Nov. 2018. [Online]. Available: http://www.igvc.org. [Accessed: 25 Feb. 2019].