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

Detailed Design Specification
CSE 4317: Senior Design II
Spring 2020

Dream Team
Autonomous Lawnmower

Ulysses Aguilar
Phu Nguyen
Alex Ho
Jerry Olds
Aderinsola Oladaiye

Revision History

Revision	Date	Author(s)	Description
0.1	2.14.2020	UA, JO, AH, PN, OA	document creation
0.2	2.24.2020	UA, JO, AH, PN, OA	complete draft
0.9	2.24.2020	UA, JO, AH, PN, OA	official release 1
1.0	5.15.2020	UA, JO, AH, PN, OA	official release 2

CONTENTS

1 Introduction **7**

2 System Overview **7**

3 Command Layer Subsystems **8**
- 3.1 Command Layer Hardware . . . 8
- 3.2 Command Layer Operating System . . . 8
- 3.3 Command Layer Software Dependencies . . . 8
- 3.4 Server . . . 8
- 3.5 ROS . . . 11

4 Control Layer Subsystems **12**
- 4.1 Layer Hardware . . . 12
- 4.2 Subsystem Operating System . . . 12
- 4.3 Subsystem Software Dependencies . . . 12
- 4.4 Subsystem Programming Languages . . . 12
- 4.5 Subsystem Data Structures . . . 12
- 4.6 Autonomous Subsystem . . . 13
- 4.7 Manual Subsystem . . . 14

5 Vision Layer Subsystems **15**
- 5.1 Layer Hardware . . . 15
- 5.2 Layer Operating System . . . 15
- 5.3 360 Camera Subsystem . . . 15
- 5.4 LIDAR Subsystem . . . 16

6 Client Layer Subsystems **17**
- 6.1 Client Layer Hardware . . . 17
- 6.2 Client Layer Operating System . . . 17
- 6.3 Client Layer Software Dependencies . . . 17
- 6.4 Client Layer programming languages . . . 17
- 6.5 User Input . . . 17
- 6.6 GUI . . . 18

7 Mower Layer Subsystems **20**
- 7.1 Layer Hardware . . . 20
- 7.2 Layer Operating System . . . 20
- 7.3 Layer Software Dependencies . . . 20
- 7.4 Power . . . 20
- 7.5 Blades Subsystem . . . 21
- 7.6 Motor Controller . . . 22
- 7.7 Arduino . . . 22

8 Safety Layer Subsystems **24**
8.1 Layer Hardware 24
8.2 Layer Operating System 24
8.3 Layer Software Dependencies 24
8.4 GPS 24
8.5 Geofence system 25
8.6 Obstacle avoidance 26
8.7 Interface 27

List of Figures

1 System architecture . . . 7
2 Command layer diagram . . . 9
3 Server . . . 10
4 ROS . . . 11
5 Autonomous subsystem diagram . . . 13
6 Manual subsystem diagram . . . 14
7 360 Camera Subsystem . . . 15
8 LIDAR Subsystem . . . 16
9 Client Layer . . . 17
10 User Input subsystem . . . 18
11 GUI subsystem . . . 18
12 Mower layer description diagram . . . 20
13 Mower layer description diagram . . . 21
14 Mower layer description diagram . . . 22
15 Mower layer description diagram . . . 23
16 safety subsystem description diagram . . . 24
17 safety subsystem description diagram . . . 25
18 safety subsystem description diagram . . . 26
19 safety subsystem description diagram . . . 27

1 INTRODUCTION

This Detailed Design Specification document will describe in-depth what is needed for each subsystem to preform their task. Each system and subsystem is represented in a diagram and their responsibilities are listed in the Architectural Design Specification document. This document will include the hardware, operating system, and software dependencies of each system and the hardware, operating system, software dependencies, programming languages, data structures and data processing used in each subsystem. Each system and subsystem make it possible for all the requirements of the vehicle to be met. The requirements for this vehicle can be found in the System Requirements Specification document.

2 SYSTEM OVERVIEW

Below is the Architectural Design diagram for the system. Input is given from the client layer to the control layer. The control layer sends that input to the safety layer to check if the input is safe to execute. The safety layer uses the vision layer to determine if the input is safe or not. The safety layer will then send the input to the command layer. If the input is safe to execute, the command layer will then send the input to an Arduino on the mower for it to control components of the mower to carry out the desired command. If the input is not safe to execute, the command layer will send an error message back to the client layer.

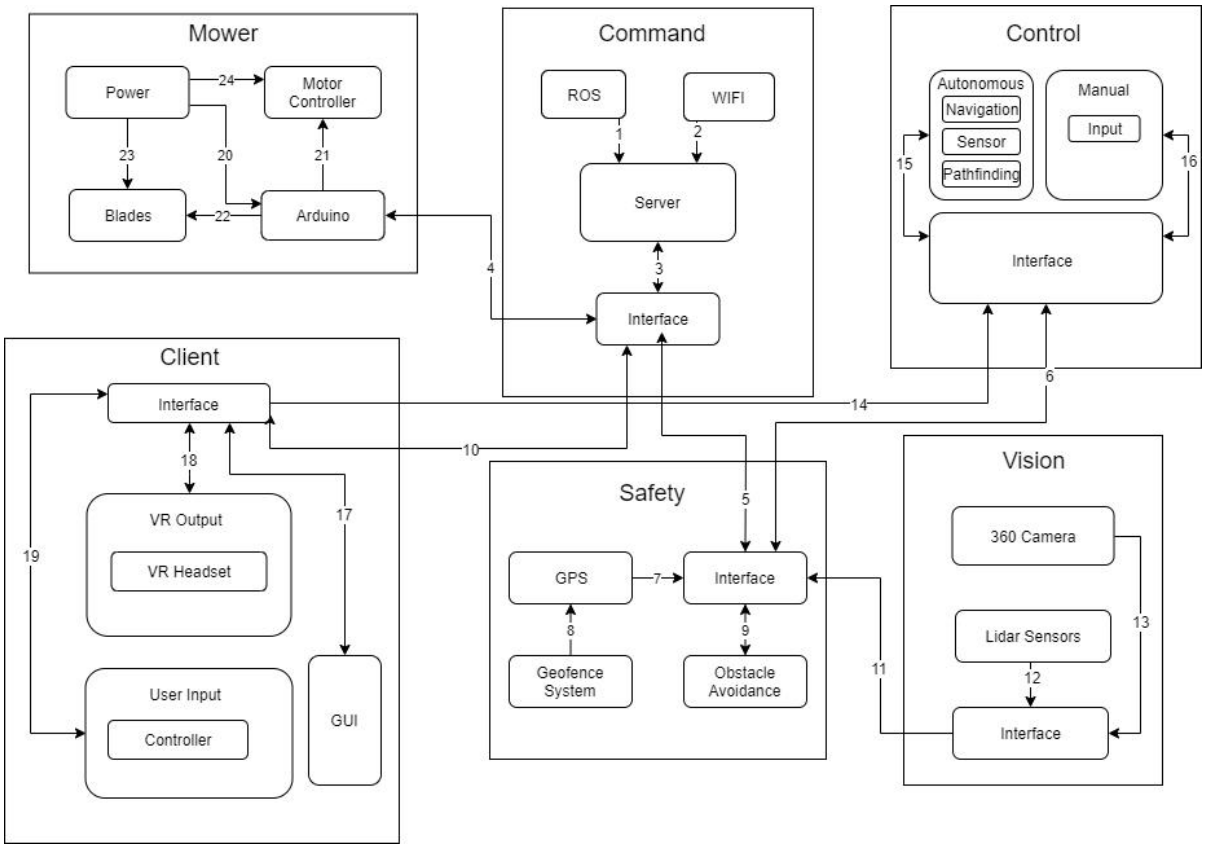

Figure 1: System architecture

3 Command Layer Subsystems

Command layer subsystems include a Server, ROS, WiFi, and an interface for interaction with other layers. The server used will be the Stealth LPC-862. The Server will be responsible for processing inputs and generating outputs. ROS packages will be loaded on to the server to handle the different functionalities of the system. The server will also be connected to WiFi so that it can communicate with the client machine. The server will interact with an interface to receive inputs and send outputs to other layers.

3.1 Command Layer Hardware

- Stealth LPC-862

3.2 Command Layer Operating System

- Ubuntu 18.04

3.3 Command Layer Software Dependencies

- ROS Melodic Morenia
- teleop_twist_joy
- rosserial_arduino
- ublox
- rplidar_ros

3.4 Server

The on-board server is the brains of the vehicle. It will be processing all the data and sending commands to other subsystems.

3.4.1 Server Hardware

- Stealth LPC-862

3.4.2 Server Operating System

- Ubuntu 18.04

3.4.3 Server Programming Languages

- C++
- Python 3

3.4.4 Server Data Structures

- **geometry_msgs/Twist**. Twist is an object that generates messages for an Arduino to receive to control the vehicle based on input from the user's controller. The messages Twist generate range from [-1,1] based on placement of the analog sticks of the controller. Twist will generate a 1 when the analog stick is pushed all the way forward and it will generate a -1 when the analog stick is pushed all the way backwards. The left analog stick will accelerate the left wheel of the vehicle in both directions and the right analog stick will accelerate the right wheel. When the Arduino receives the message, it will tell the motor controller to send voltage to the wheels based on the input from the controller.

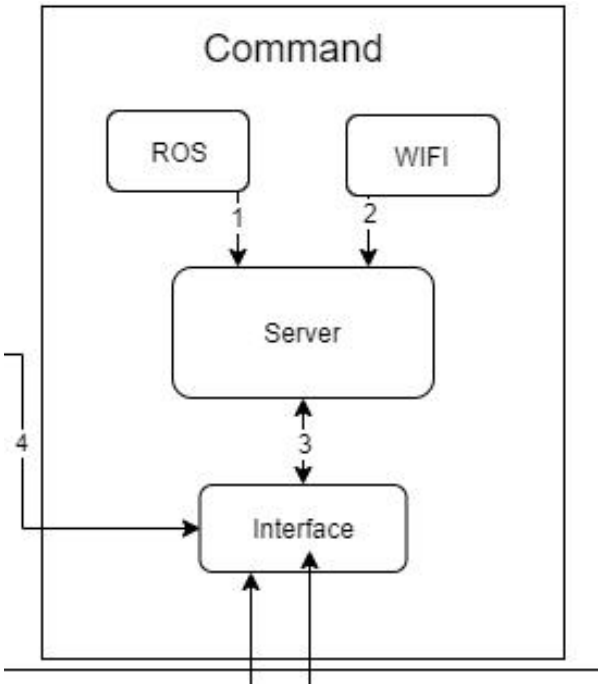

Figure 2: Command layer diagram

- **sensor_msgs/LaserScan**. LaserScan is an object that contains data from LIDAR readings. The most important member variable belonging to LaserScan is the ranges array. The ranges array contains the distances the vehicle is from objects in a 180 degree sweep. The ranges array starts at value 0 scanning directly to the right of the vehicle, and ends at value 381 scanning directly to the left of the vehicle. Using the readings from the LIDAR, we can set minimum distances each laser should be from an object and if those minimum distances are violated we can make the vehicle stop before a collision occurs.

- **sensor_msgs/NavSatFix**. NavSatFix is an object that contains data gathered from the ZED-F9P GPS module. From the NavSatFix object, we can extract latitude and longitude values from the GPS. Using those values, we can compare our current latitude and longitude to our goal latitude and longitude and create a path to get to our goal location.

3.4.5 Server Data Processing

- **LIDAR data processing**. The on-board server first calculates the minimum acceptable distance for each laser in the 180 degree sweep. During operation, our custom ROS safety node constantly compares the distance readings from the LIDAR to the minimum acceptable distances that were calculated. If at any point the minimum distance of 5 contiguous lasers are violated, the vehicle will come to a stop and will not be able to proceed forward. The only movement allowed when there is an object impeding the vehicle's path is straight backwards.

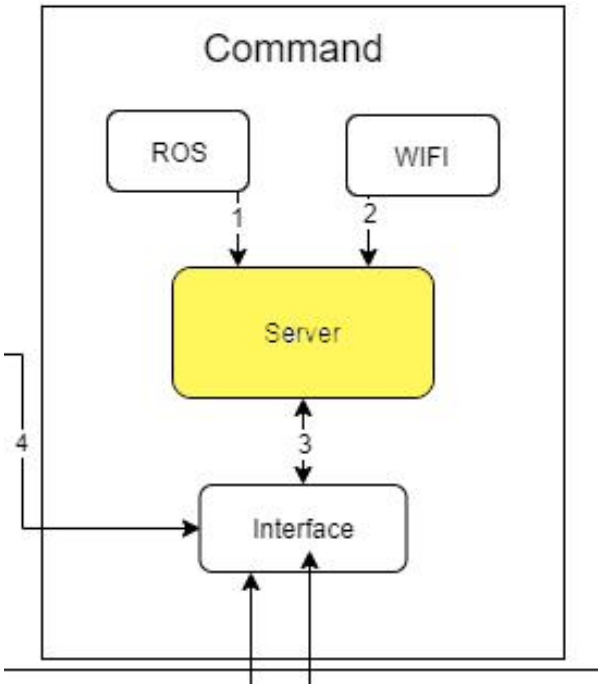

Figure 3: Server

- **Waypoint navigation**. The on-board server first reads the goal latitude and goal longitude from a text file. It then gets the current latitude and longitude of the vehicle from the on-board GPS module. Once the current and goal GPS coordinates are known, a vector is calculated by subtracting the current latitude and longitude from the goal latitude and longitude. The quadrant of the goal destination is known based on sign of the difference of the latitudes and longitudes. The distance between the current location and goal destination is then calculated by using the difference formula. Lastly, the angle from the current location to the goal location is calculated using the ArcTan function. The vehicle will get its current angle and turn in place until its current angle is equal to the angle needed to reach the destination. The vehicle will then travel in a straight line to its destination and will correct itself if it veers off course.

3.5 ROS

Robot Operating System (ROS) is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms [**?**]. ROS reduces the time to complete functionalities of the vehicle significantly and provides a way for all the different systems to communicate with each other.

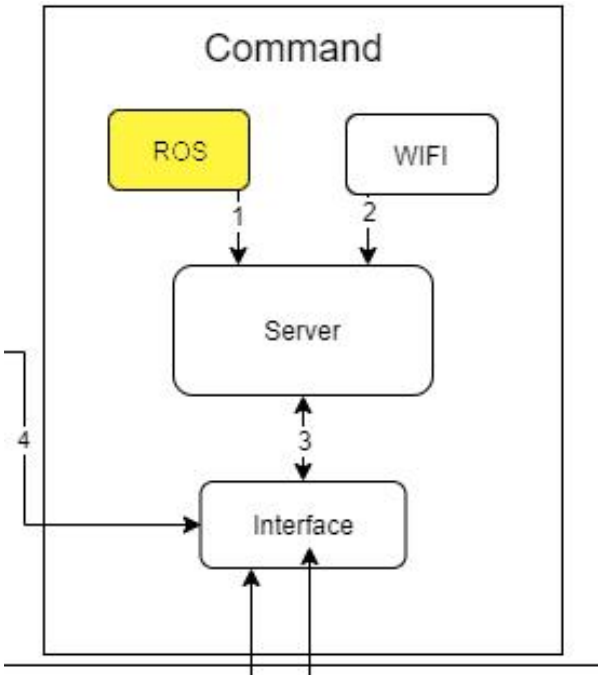

Figure 4: ROS

3.5.1 ROS Programming Languages

- C++
- Python 3

4 Control Layer Subsystems

The control layer subsystems include Autonomous, Manual, and an interface for interaction with other layers. The user will be able to select between the autonomous and manual mode by pressing a button on the Xbox controller.

4.1 Layer Hardware

The Arduino Uno WiFi V2 is responsible for taking the messages that are published by the cmd_vel topic and converts them into PWM signals that are sent to
the motor controller.

4.2 Subsystem Operating System

- Ubuntu 18.04 LTS
- ROS Melodic Morenia

4.3 Subsystem Software Dependencies

- ros-melodic-rosserial-arduino
- ros-melodic-rosserial
- libusb-dev
- libspnav-dev
- libbluetooth-dev
- libcwiid-dev

4.4 Subsystem Programming Languages

- Python 3
- C/C++

4.5 Subsystem Data Structures

The data coming from cmd_vel is encoded in geometry_msgs/Twist.

4.6 Autonomous Subsystem

The autonomous subsystem is responsible for storing the position of the obstacles in the field of operation. After sensing them, it will calculate the path of the lawnmower based on the obstacles positions and navigate through the field. This subsystem is a custom ROS node named waypoint_navigation.

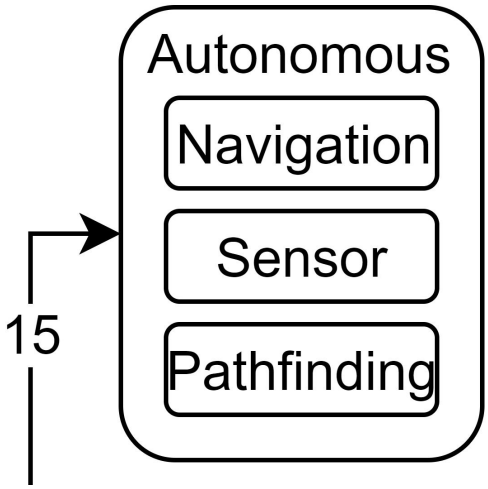

Figure 5: Autonomous subsystem diagram

4.6.1 Subsystem Hardware

- Triple-axis Accelerometer+Magnetometer (Compass) Board - LSM303

4.7 MANUAL SUBSYSTEM

The manual subsystem is responsible for the manual control ability of the lawnmower. This is an option that the user can choose.

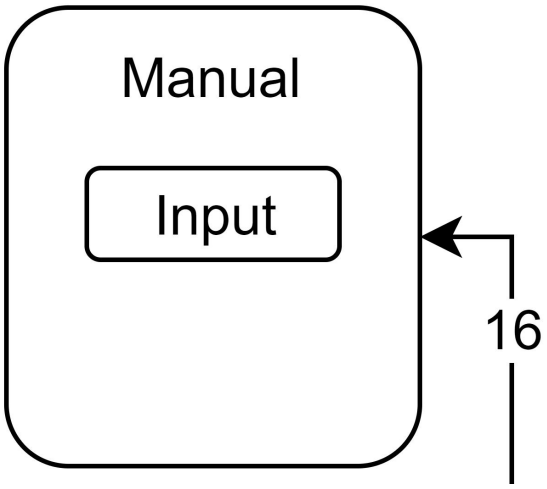

Figure 6: Manual subsystem diagram

5 Vision Layer Subsystems

Layer responsible for all vision related sensors.

5.1 Layer Hardware

- Linux Server: On-board Linux server used to host ROS
- Local WiFi Router: Dependant on the LAN of the location

5.2 Layer Operating System

- Ubuntu 18.04 LTS: Current Linux distro that can run ROS Melodic
- ROS Melodic: Modular operating system for robot developement

5.3 360 Camera Subsystem

Subsystem responsible for 360 Vision of the robot. It consists of a Theta V 360 Camera and software running on the server to deliver a video.

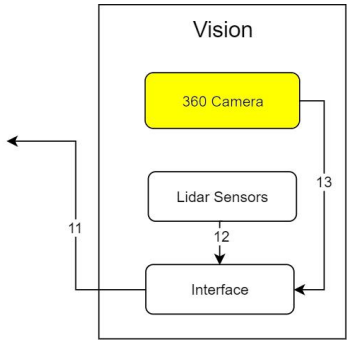

Figure 7: 360 Camera Subsystem

5.3.1 Subsystem Hardware

- Theta V 360 Camera

5.3.2 Subsystem Software Dependencies

- OpenCV Python 4.2.0.32: Computer Vision software with video streaming capabilities
- A-frame 1.03: VR Framework for mobile device browsers
- Flask: Lightweight Python webserver framework
- imutils: Series of convenient functions for OpenCV and Python

5.3.3 Subsystem Programming Languages

- Python 3.8.1

5.3.4 Subsystem Data Structures

Video feed is formatted as a MJPEG stream.

5.4 LIDAR Subsystem

Subsystem responsible for getting LIDAR distance readings. It consists of a RPLIDAR S1 and a ROS Node

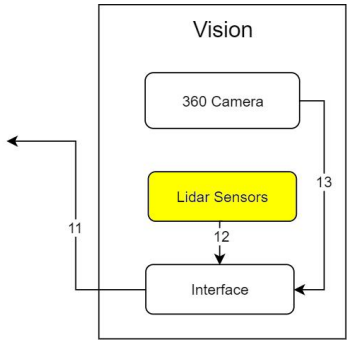

Figure 8: LIDAR Subsystem

5.4.1 Subsystem Hardware

- RPLIDAR S1

5.4.2 Subsystem Software Dependencies

- RPLidar SDK: 1st party ROS Node SDK for RPLIDAR

5.4.3 Subsystem Programming Languages

- Python 3.8.1

5.4.4 Subsystem Data Structures

LIDAR lasers are read as an array of lasers, with a given minimum angle, maximum angle, and angle increment between lasers.

6 Client Layer Subsystems

The current client layer comprises of the User Input, GUI and Interface subsystems. Its is responsible for taking inputs from the other layers and creating outputs based on the inputs given. The command layer communicates with the client layer, safety layer, and the mower layer. The client and command layers send connectivity messages to each other. The safety layer first checks if a command from the control layer is safe to execute. If the command is safe to execute, the safety layer will send the command to the command layer.

6.1 Client Layer Hardware

- Xbox controller
- Arduino

6.2 Client Layer Operating System

- Ubuntu 18.04 LTS

6.3 Client Layer Software Dependencies

- ROS Melodic Morenia
- teleop_twist_joy
- rosserial_arduino
- ublox
- rplidar_ros

6.4 Client Layer programming languages

- ROS Melodic Morenia
- C++
- Python 3

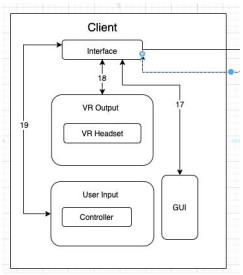

Figure 9: Client Layer

6.5 User Input

This is the hardware used to pilot the entire system when it is manual mode

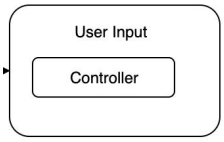

Figure 10: User Input subsystem

6.5.1 Subsystem Hardware

- Xbox controller to pilot the entire system
- Arduino

6.5.2 Subsystem Operating System

- ROS Melodic Morenia

6.5.3 Subsystem Software Dependencies

- **teleop_twist_joy**. to send and receive data about the current positions of the joystick on the axis of the xbox controller

6.5.4 Subsystem Programming Languages

- ROS Melodic Morenia

6.5.5 Subsystem Data Structures

Data is sent from Xbox to the arduino which drives the motor controllers thereby drives the system.

6.5.6 Subsystem Data Processing

messages in the teleop node of ROS are published periodically based on the current position of the joysticks on its axis .

6.6 GUI

This is the front-end software that allows the user to pick the gps coordinates for the lawnmower to follow as a waypoint when its in autonomous mode.

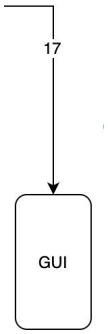

Figure 11: GUI subsystem

6.6.1 Subsystem Software Dependencies

- ROS Melodic Morenia
- teleop_twist_joy
- rosserial_arduino
- ublox
- rplidar_ros

6.6.2 Subsystem Programming Languages

- C++
- Python 3

6.6.3 Subsystem Data Structures

We can generate a waypoint file to be for the waypoint following alogrithm.

6.6.4 Subsystem Data Processing

Uses gps coordinates to guide lawnmower on areas to avoid.

7 Mower Layer Subsystems

The mower layer is very simple it uses its subsystems to handle commands and sends status reports of the mower to let the system now what is on and what is off.

7.1 Layer Hardware

- Arduino Uno WiFi Rev2
- Motor controller will depend on the mower, but in our case it is a RoboteQ HDC2450

7.2 Layer Operating System

Micro controller, so no operating system

7.3 Layer Software Dependencies

- Arduino API
- Rosserial Arduino
- Arduino Timer library

7.4 Power

Power is what we will use to provide power to our system along with battery level data.

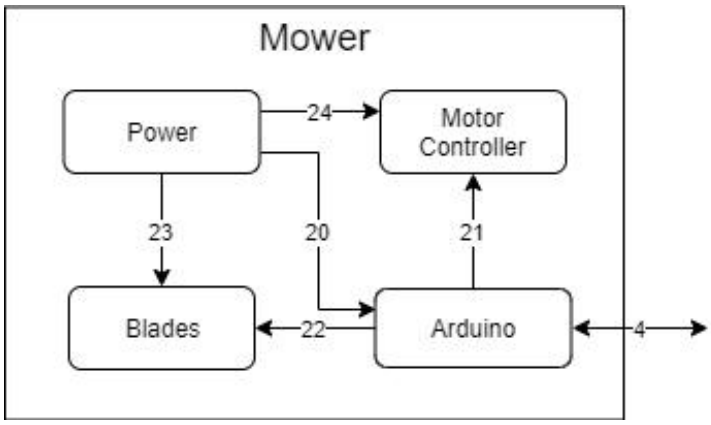

Figure 12: Mower layer description diagram

7.4.1 Subsystem Hardware

For the mower we will have several batteries in an arrangement in series and parallel depending on mower. These batteries will be wired using a plug that we can connect to charge or connect to complete the circuit to power our system. Our mower will also have appropriate fuses to protect valuable hardware that will be physically powered by these batteries. I order to measure battery levels we will have a voltmeter board wired directly to the fuse box and to the Arduino to send data.

7.4.2 Subsystem Operating System

The power subsystem does not operate using an operating system

7.4.3 Subsystem Software Dependencies

The power subsystem does not operate using an new software.

7.4.4 Subsystem Data Structures

The power subsystem does not operate using Data Structures.

7.4.5 Subsystem Data Processing

The power subsystem does not operate using Data Processing.

7.5 Blades Subsystem

The blades subsystem will keep track of the status of the blades, and will accept commands for turning the blade on and off.

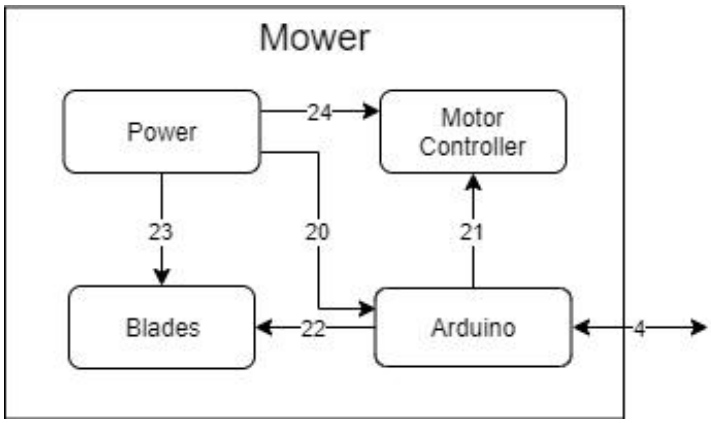

Figure 13: Mower layer description diagram

7.5.1 Subsystem Hardware

For blades the arduino will be strictly in charge of sending an on signal to the switch that activates the blades on the mower. This will be a switch that simply connects batter power to the blades when triggered on and disconnect when triggered off.

7.5.2 Subsystem Operating System

The blade subsystem does not operate using an operating system

7.5.3 Subsystem Software Dependencies

The blade subsystem does not operate using an new software.

7.5.4 Subsystem Data Structures

The blade subsystem does not operate using Data Structures.

7.5.5 Subsystem Data Processing

The blade subsystem does not operate using Data Processing.

7.6 Motor Controller

The motor controller subsystem will take in commands from the arduino and convert them into commands the motors can actually achieve our system is open loop so no feed back will be used.

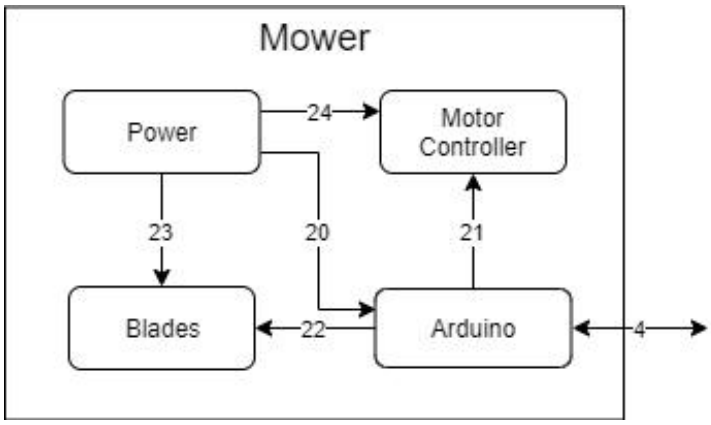

Figure 14: Mower layer description diagram

7.6.1 Subsystem Hardware

The motor controller varies from what platform our system is mounted on, but the motor controller will have an input for power from our batteries and is designed for a differential drive system. the controller will feed appropriate voltage to the two separate motors to spin them at a specific speed in forward or reverse depending on analog or PWM input from the Arduino.

7.6.2 Subsystem Operating System

The Motor Controller subsystem does not operate using an operating system

7.6.3 Subsystem Software Dependencies

The Motor Controller subsystem does not operate using an new software.

7.6.4 Subsystem Data Structures

The Motor Controller subsystem does not operate using Data Structures.

7.6.5 Subsystem Data Processing

The Motor Controller subsystem is configured to process PWM signals or analog voltages.

7.7 Arduino

The arduino subsystem acts as our interface into the mower layer. It will keep track of our mowers status, receive commands, and send commands withing the mower layer.

7.7.1 Subsystem Hardware

the Arduino is the interface for the mower layer. It is wired to outside layer via USB, and has jumper wires connected to the motor controller, and the voltmeter for the batteries.

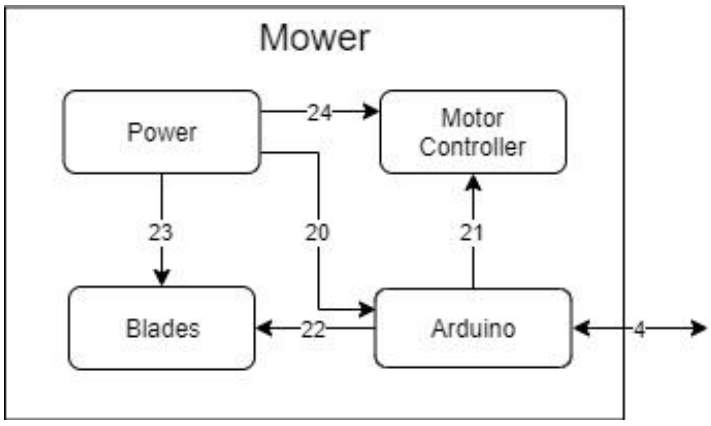

Figure 15: Mower layer description diagram

7.7.2 Subsystem Operating System

The Arduino subsystem does not operate using an operating system

7.7.3 Subsystem Software Dependencies

The Arduino subsystem uses the software dependencies stated in section 7.3.

7.7.4 Subsystem Programming Languages

- Arduino IDE
- Python 3

7.7.5 Subsystem Data Structures

The Arduino must follow the publishing and subscribing structures of the teleop ROS node. The arduino itself is a rosserial Python Node that pudblishes structured message, so user can know what values the arduino is receiving and sending to the motor controller.

7.7.6 Subsystem Data Processing

the Arduino must be able to process values published and sent via uart communication. these messages contain values the client compute wishes to send to the motor controller, or to the blades.

8 Safety Layer Subsystems

The following subsystems where picked for the safety layer to make it easier to make safety checks for our system.

8.1 Layer Hardware

- Sparkfun GPS-RTK2 Board-ZEDF9P
- Stealth LPC-862

8.2 Layer Operating System

Ubuntu 18.04

8.3 Layer Software Dependencies

- ROS Melodic
- ROS Ublox

8.4 GPS

The GPS subsystem is used to keep track of our position, orientation, and speed. It receives information from the geofence system to be able to determine if our location is allowed. The GPS subsystem will also tell us if our GPS connection is lost. If any of the data measurements from the GPS are out of our safety range flags will be set and sent to the interface subsystem as needed. The GPS data is also used by the control layer to navigate, so orientation, speed and position are sent to the interface as well.

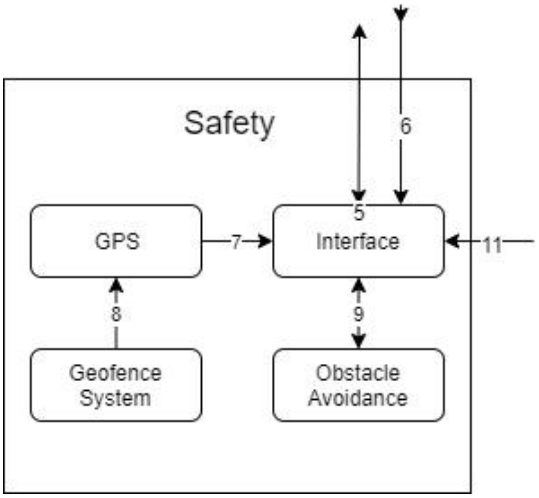

Figure 16: safety subsystem description diagram

8.4.1 Subsystem Hardware

As stated We use two sparkfun GPS-RTK boards one setup as a rover the other as a base. the base will be stationary and will transmit RTCM3 correction data over radio using the UART2 pins at a baud rate of 115200. The rover will get GPS data from an antenna along with RTCM3 correction data from base station through radio waves to UART 2 pins at a baud rate of 115200. GPS is connected via USB UART1 to the interface.

8.4.2 Subsystem Operating System

The GPS subsystem does not operate using an operating system

8.4.3 Subsystem Software Dependencies

The GPS subsystem uses Ublox API.

8.4.4 Subsystem Programming Languages

none.

8.4.5 Subsystem Data Structures

The GPS system has specific structure that data is sent and received which is determined by the developers at Ublox

8.4.6 Subsystem Data Processing

the GPS handles processing of signals from GPS satellites and RTCM3 correction data, which is developed by ublox.

8.5 Geofence system

The geofence system is setup here and constantly sends geofence position to GPS subsystem.

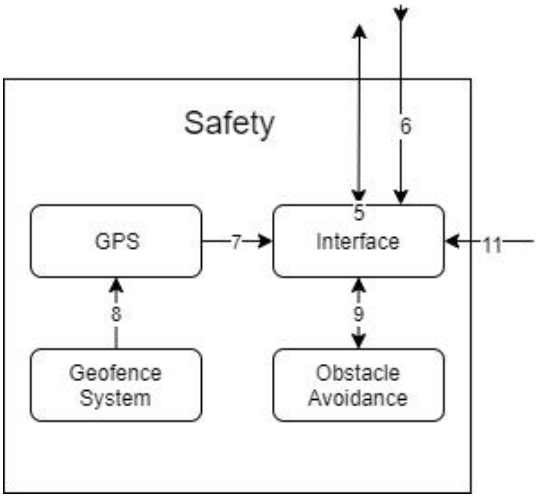

Figure 17: safety subsystem description diagram

8.5.1 Subsystem Hardware

The Geofence system is incorporated as part of the functionality of the Sparkfun GPS board.

8.5.2 Subsystem Operating System

The Geofence subsystem does not operate using an operating system

8.5.3 Subsystem Software Dependencies

The Geofence subsystem uses the software dependencies from ublox, configured using p-center software.

8.5.4 Subsystem Programming Languages

none

8.5.5 Subsystem Data Structures

The Geofence data structures are once again determined by Ublox.

8.5.6 Subsystem Data Processing

the Geofenc subsystem is configured using P-center, and developed by ublox.

8.6 Obstacle avoidance

The obstacle avoidance subsystem is responsible for detecting obstacles from camera, and LIDAR data received from the safety layer interface. If an obstacle is detected flags will be set and a protocol to avoid the obstacle will be sent to the interface which will then be sent to the command layer.

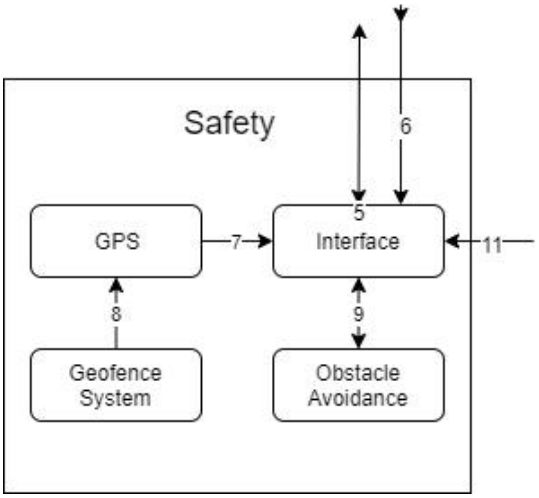

Figure 18: safety subsystem description diagram

8.6.1 Subsystem Hardware

Obstacle avoidance is the on board server receiving all safety data via USB, WiFi, or Ethernet.

8.6.2 Subsystem Operating System

Obstacle avoidance subsystem uses ubuntu 18.04

8.6.3 Subsystem Software Dependencies

- ROS Melodic
- ROS Ublox
- rplidar
- rosserial Arduino
- ROS safety node

8.6.4 Subsystem Programming Languages

- python 3
- c++

8.6.5 Subsystem Data Structures

obstacle avoidance has to handle ros node data structure published by the Lidar node, ROS ublox node, and rosserial arduino python node. It also must be able to publish safety Data structure in our safety node to let the system know when a violation has occurred.

8.6.6 Subsystem Data Processing

the obstacle avoidance subsystem must be able to process and compare the data from all the nodes listed in section 8.6.5 and given that data if an obstacle is detected it should be able to produce a series of commands that will avoid colliding with the object or maneuver around the object.

8.7 Interface

The Interface subsystem is used to route data from the safety layer, and data the safety layer need to function properly

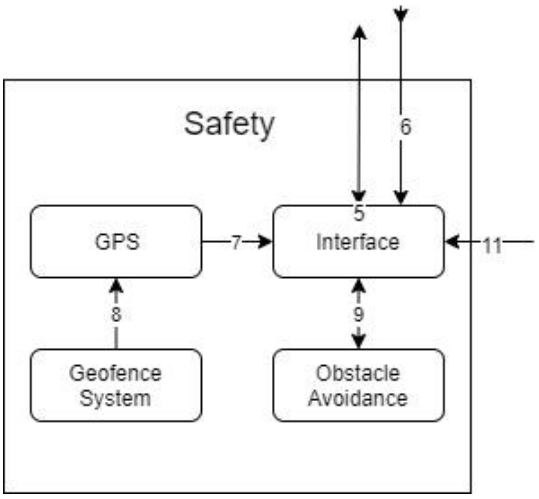

Figure 19: safety subsystem description diagram

8.7.1 Subsystem Hardware

The interface for the safety layer is mostly software based, but all connection to other layers or subsystems within our layer are through USB, WiFi or Ethernet.

8.7.2 Subsystem Operating System

Ubuntu 18.04

8.7.3 Subsystem Software Dependencies

- ROS Melodic
- ROS Ublox
- rplidar
- rosserial Arduino
- ROS safety node

8.7.4 Subsystem Programming Languages

- python 3
- c++
- Arduino IDE

8.7.5 Subsystem Data Structures

the interface subsystem has to handle ros node data structures published by the Lidar node, ROS ublox node, and rosserial arduino python node. It also must be able to publish safety Data structure in our safety node to let the system know when a violation has occurred.

8.7.6 Subsystem Data Processing

interface subsystem must be able to process and compare the data from all the nodes listed in section 8.7.5 and given that data if a safety violation is detected it should be able to produce a series of commands that will avoid colliding with the object or maneuver around the object. Setting off a series of flags sent to the proper layers, and essentially blocking unsafe behavior.

REFERENCES