Battlefield Effects Simulator Robot

Group 6 Final Presentation

30 November 2022

Jared Rymkos
Computer engineering

Julia Kemper
Computer engineering

Michael Rodriguez
Electrical engineering

Nicholas Nachowicz
Computer engineering

UCF UNIVERSITY OF CENTRAL FLORIDA

PEO STRI

Sponsors/Mentors:
James Todd
Jeremy Lanman
David Howard
Tom Waligora

Project Overview

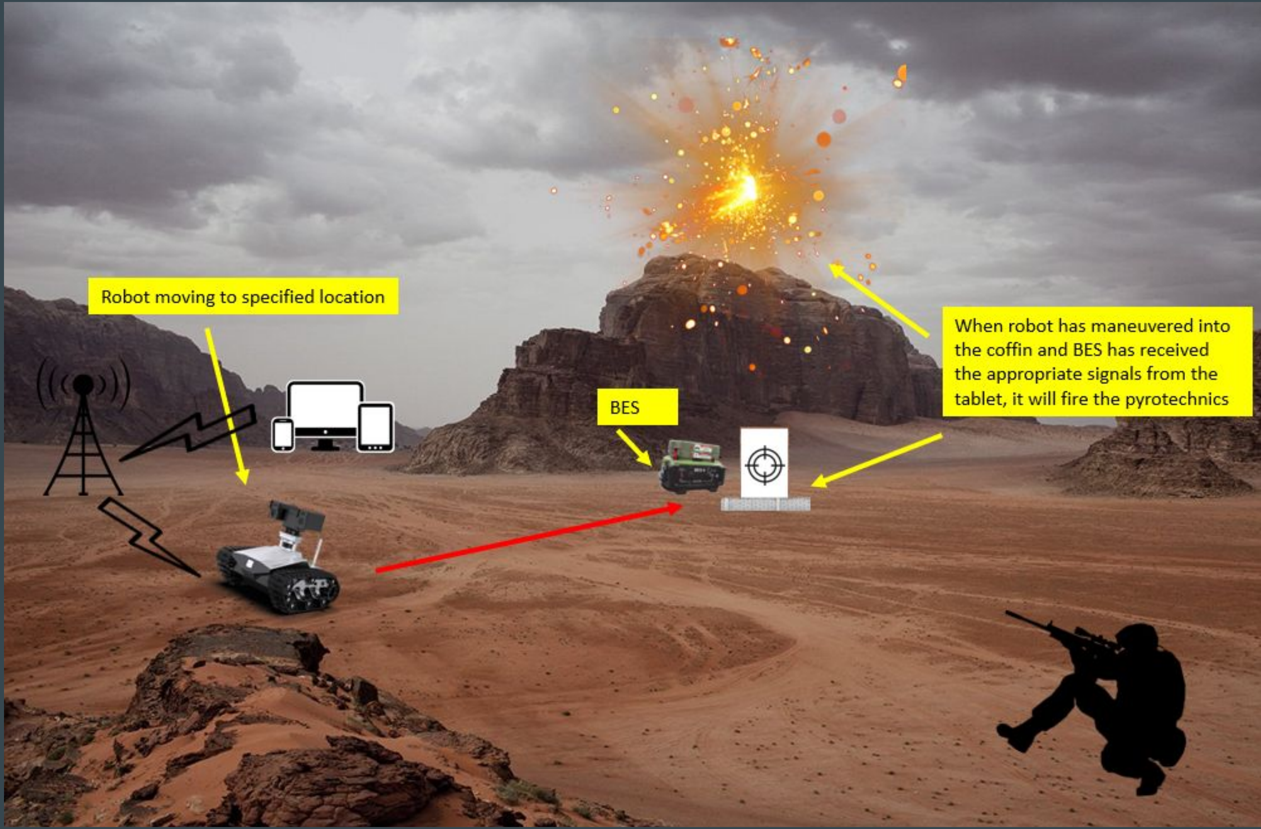
The Battlefield Effects Simulator robot is a small scale version of a robotic system that will be utilized on live fire ranges to support training realism and feedback, threat representation, and pyrotechnical handling.

This version of the robot will maneuver through rough terrain to and from desired locations to accurately fire pyrotechnics when proper safety signals are sent. The BES for this prototype will not be on top of the robot, but will still be in communication with it. The robot will interact with the BES to send the safety conditions for when it is clear to fire the pyrotechnics.

The safety conditions include:

1. The robot has arrived at the correct location on the map

2. The robot is on a flat surface plus or minus ten degrees


Robot moving to specified location
BES
When robot has maneuvered into the coffin and BES has received the appropriate signals from the tablet, it will fire the pyrotechnics

I/ITSEC Presentation

I/ITSEC is the world's largest modeling, simulation and training event

Briefed PEO STRI leadership, PEO STRI industry partners, UCF Board of Trustees, PEO STRI PEO, PEO STRI DPEO, and The Pentagon’s National Director of NSIN at I/ITSEC.

Motivation

The motivation for this project comes from the governments derived need for a Modular Open System robotic platform that aligns with the Robot Operating System Military (ROS-M) framework and ecosystem.

The robot is intended to increase training realism and feedback and improve safety measures during training by removing the human element when it comes to pyrotechnics handling.

Takeaway: This product will provide a rapid prototype and technical data package that can be leveraged by the Army to: **increase training throughput, improve safety,** and **reduce cost** of live fire training

Goals and Objectives

Main goal: For this robotic system to be utilized on live fire ranges to support training realism and feedback, threat representation, and pyrotechnical handling.

Our objectives consist of the following:

- The robot will use lidar data to create a map of its environment
- The robot will correctly navigate to its specified destination
- The robot will detect obstacles in its path and navigate around them
- The robot will ensure it is level +- 10 degrees when it has arrived at its specified destination
- The robot will ensure it has arrived at the correct destination
- The robot will send signal to GUI via ESP32 that it is safe to fire
- GUI will send signal to BES to fire pyrotechnics

Constraints

- **Economic Constraints**
  - $1,000 Budget
- **Environmental Constraints**
  - Robot must be able to accurately navigate over rough terrain, hills, and holes
- **Social Constraints**
  - N/A
- **Political Constraints**
  - N/A
- **Ethical Constraints**
  - N/A
- **Health and Safety Constraints**
  - Senior Design team will not test with pyrotechnics due to the safety of the team
- **Manufacturability Constraints**
  - Arrival of parts on time
  - Arrival of parts in good condition
- **Sustainability Constraints**
  - N/A
- **Time Constraints**
  - Senior Design timeline
  - Sponsor timeline

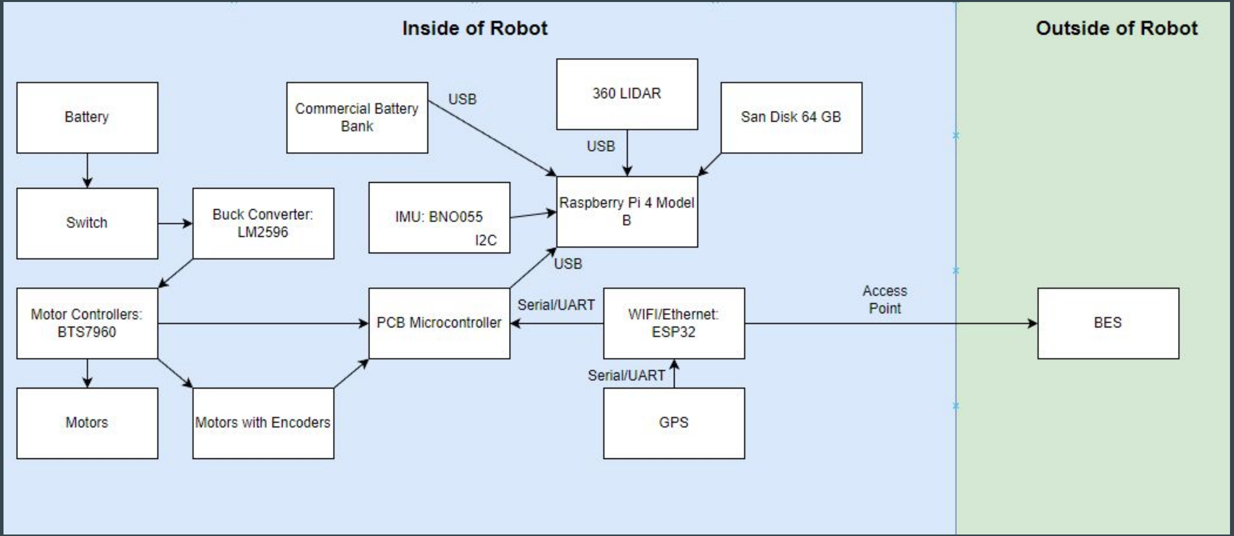
Standards

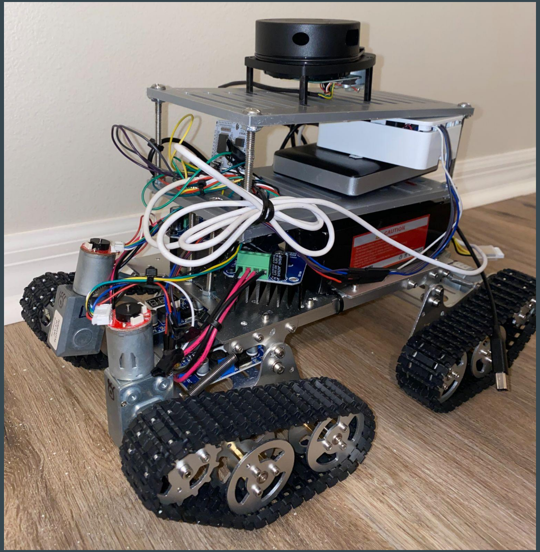
- IEEE/ISO/IEC 29148-2018 - Software Requirements Specification
  - Followed to create unambiguous, verifiable, and complete requirements
  - Avoid terms like “if possible” in requirements in order to make them clear
- ISO/IEC/IEEE 29919 - Software Testing
  - Followed to have a clear procedure when troubleshooting the code
  - Different test methods including Specification, Structure, and Experience based techniques
- Coding Conventions - Java/Python/C++
  - Followed in order to have clear and cohesive code that mentors and team can follow
  - Allow smooth transition to future teams that build off our project
- IEEE 1725 - Rechargeable Battery Standard
  - Followed in order to maintain a safety requirement involving the battery
  - Lowers risks of accidents occuring with the battery and allows for all specs of the batteries to be available for the user.

Specifications

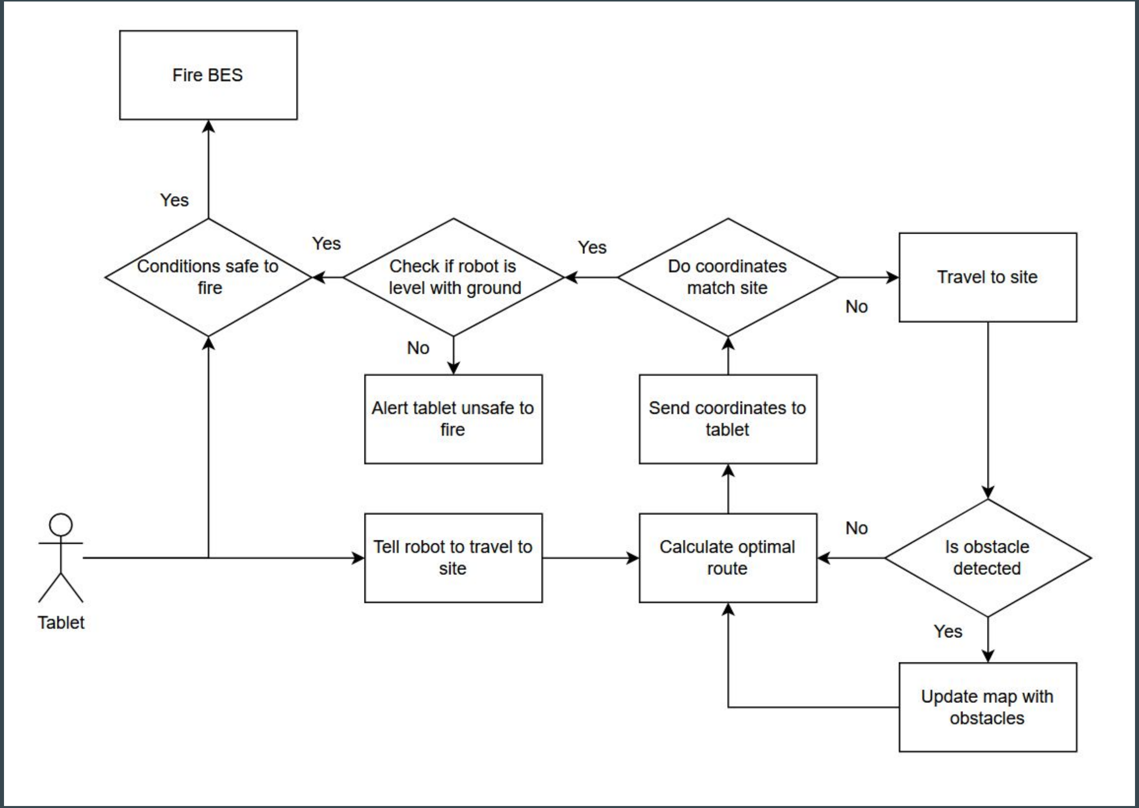
Robot Dimensions	14.76 * 9.45 * 11 in (L*W*H)
Robot Weight	8 lbs.
Robot Run Time	1h 44 min minimum
The system will be mobile	-
The system will detect objects	-
The system with detect if it is level	-

Overall Design





Overall Movement Flow Chart


Fire BES
Yes
Conditions safe to fire
Yes
Check if robot is level with ground
Yes
Do coordinates match site
No
Travel to site
No
Alert tablet unsafe to fire
Send coordinates to tablet
Tablet
Tell robot to travel to site
Calculate optimal route
No
Is obstacle detected
Yes
Update map with obstacles

Project Accomplishments

General Software/Hardware:

- Creation of custom printed circuit board
- Ubuntu integration with Raspberry Pi
- ROS integration with Ubuntu/Raspberry Pi
- Communication between ROS, Raspberry Pi and Arduino
- Creation of GUI that tracks robots location and

Sensor Data:

- Lidar integration with Raspberry Pi
- IMU integration with Raspberry Pi
- Output of lidar data in terminal
- Visualization of lidar data through ROS RVIZ application
- SLAM (Simultaneous Localization and Mapping) integration
  - Creating a map of robots environment using lidar
  - Creating costmap configurations to detect obstacles in robot's environment
  - Publishing IMU data and visualizing it to RVIZ application

Odometry Data:

- Publishing robot velocity commands through Arduino code to motor controllers
- Remotely controlling robot velocity through ROS steering GUI
- SLAM (Simultaneous Localization and Mapping) integration
  - Creating initial and goal pose publishiner
  - Visualizing robot moving in a mapped environment
  - Autonomous navigation of robot
- Motor encoder, motor controller, and arduino integration

Communication:

- Windows tablet runs GUI and acts as client while ESP32 acts as server
- Tablet and ESP32 are on the same network and communicate through sockets
- BES is tethered to ESP32 which acts as a DHCP server and assigns the IP
- Integration with existing product line software

Work Distribution

	ROS Navigation	Software Integration	PCB Design	GUI	Communication	Hardware Design
Jared		S		P	P	
Julia	P	P			S	
Michael						S
Nicholas	S		P		S	P

P = Primary
S = Secondary

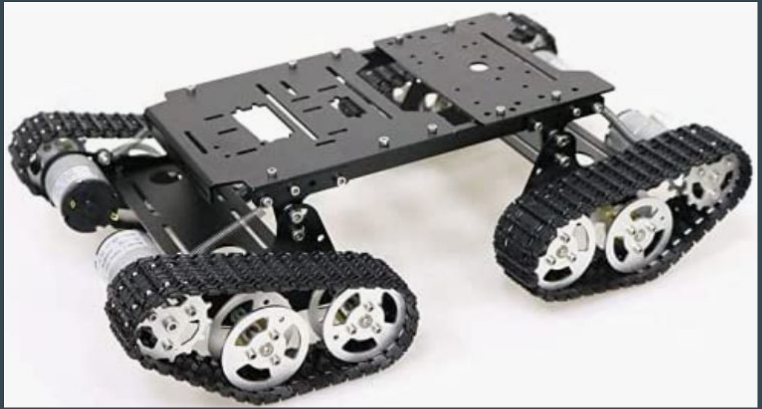
Hardware - Physical Design

Robot Base

- The base is a pre-designed model that was purchased through Amazon.
- The base utilizes 4 treads and 4 motors to operate
- Utilizes several slots to attach components or extra platforms through the usage of screws.

Base specs

- Dimensions: 14.76 * 9.45 * 4.13 in. (L*W*H)
- Weight: 1.66 kg = 3.66 lbs
- Material: Aluminum Alloy

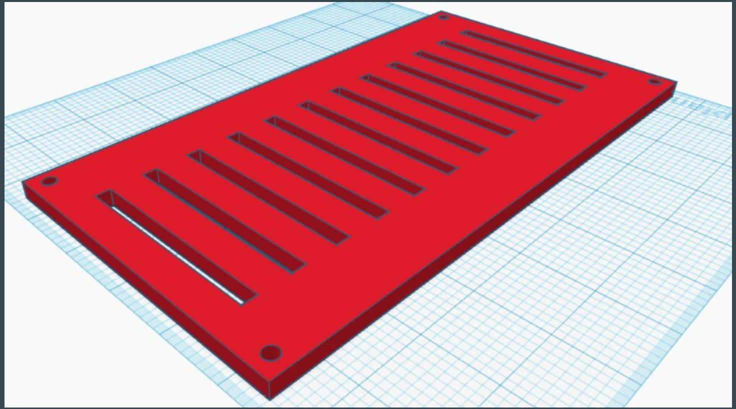


3D Printed Platform

- A base was designed to be 3D printed in order to have more space for components on the robot.
- Utilizes slots for multipurpose use like utilizing either wires to go through for organization or screws to hold a component in place.
- Utilizes 4 circular holes in the corners for stacking platforms on top of each other.

Platform information

- Dimension: 8.55 * 4.5* 0.225 in. (L*W*H)
- Slot Dimension: 0.225 * 2.7 in. (L*W)
- Hole Diameter: 0.2052 in.



Treads

We decided on utilizing treads over wheels due to the following factors

- Treads have better off-road capabilities
- Treads has a larger surface area which leads to less pressure into the ground which decreases chance of getting stuck
- Has a better grip than standard wheels which leads to less slipping

Problems with treads:

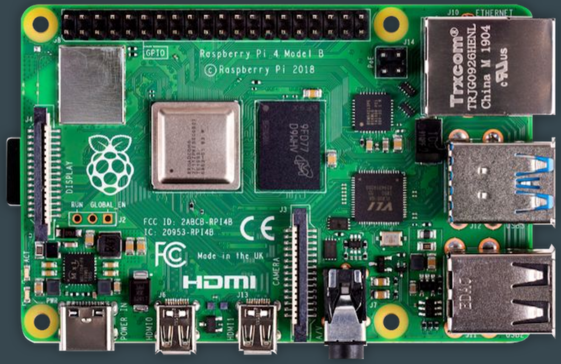
- Lower speed than wheels
- More power required to move
- Can’t move if one of the treads break

Hardware - Electrical Components

Raspberry Pi 4B Model

- Best performance specs compared to Jetson Nano, Odroid XU4, and Jetson Xaiver
- Lots of documentation regarding Raspberry Pi 4B and ROS integration
- Affordable and available for purchase by reliable vendor

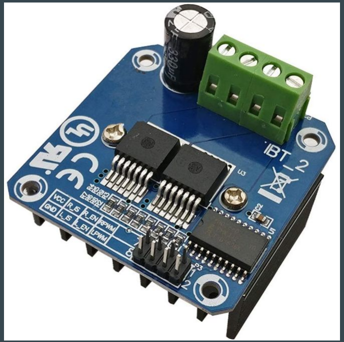
- Ran Ubuntu Desktop
- Used to power LiDAR and process LiDAR data
- Used to power IMU and process IMU data
- Used to power PCB
- Acts as a virtual machine to run all ROS programs

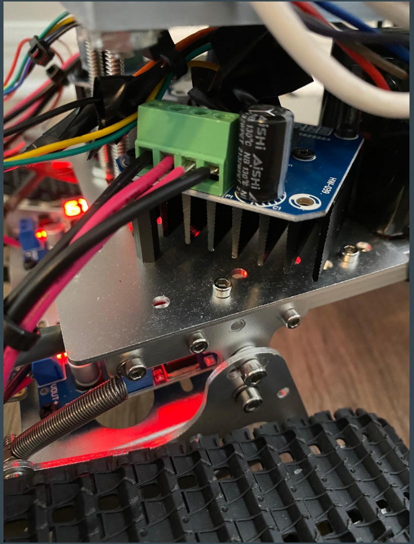


BTS7960 Motor Driver

- Used to power and control the motors
- Reliable for powering both low voltage and high voltage motors
- Low cost and available from a reputable vendor

- Capable of powering motors with 6-27 V
- Has a max output of 43 A for motors
- Only needs 5 V to operate logic side





LiDAR - RPLIDAR A1

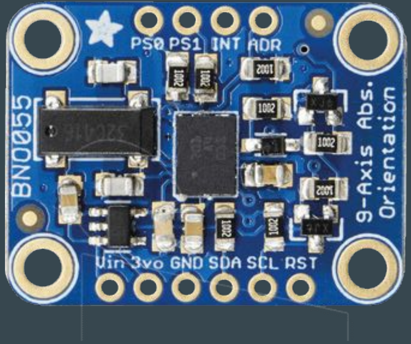
- Used to create map environment
- Used to detect obstacles in environment
- Used to accurately navigate through mapped environment
- Used RPLiDAR Package created by ROS

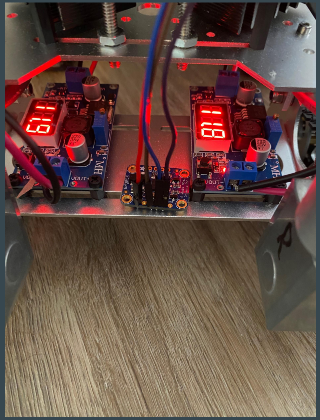
- 360 degree 2D laser scan
- 12 meter laser scan distance
- 5.5 Hz scan rate
- 8000 per time samples

IMU - BNO055

- Used to determine if robot is level once it has reached its specified destination
- Used IMU BNO055 package created by ROS to integrate with OS

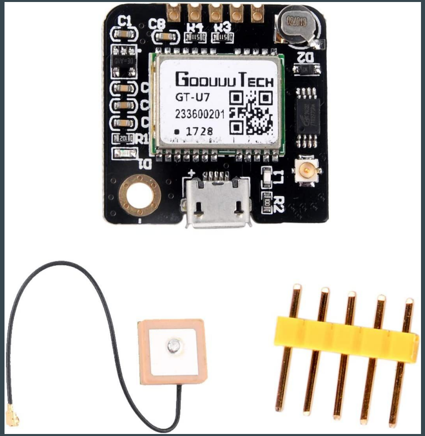
- Three axis orientation data based on a 360° sphere
- Four point quaternion output for more accurate data manipulation





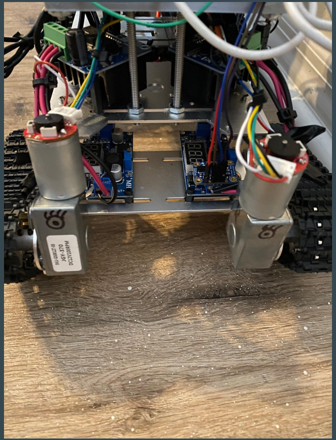
GPS - NEO 6

- Used to receive longitude and latitude from satellite
- Needs time to pick up a satellite before its accurate
- Coordinates are transferred to GUI to be displayed
- Connected to ESP32 through UART using GPIO pins

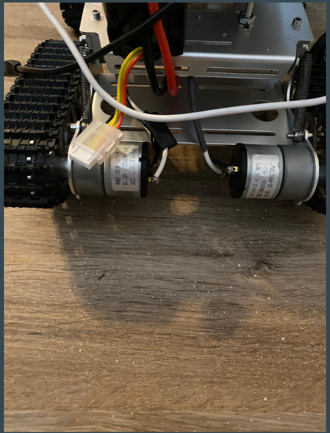


Motors with and without Encoders

- Front Motors
- Used to help with Nav Stack Tranforms.
- 12V
- 100RPMS

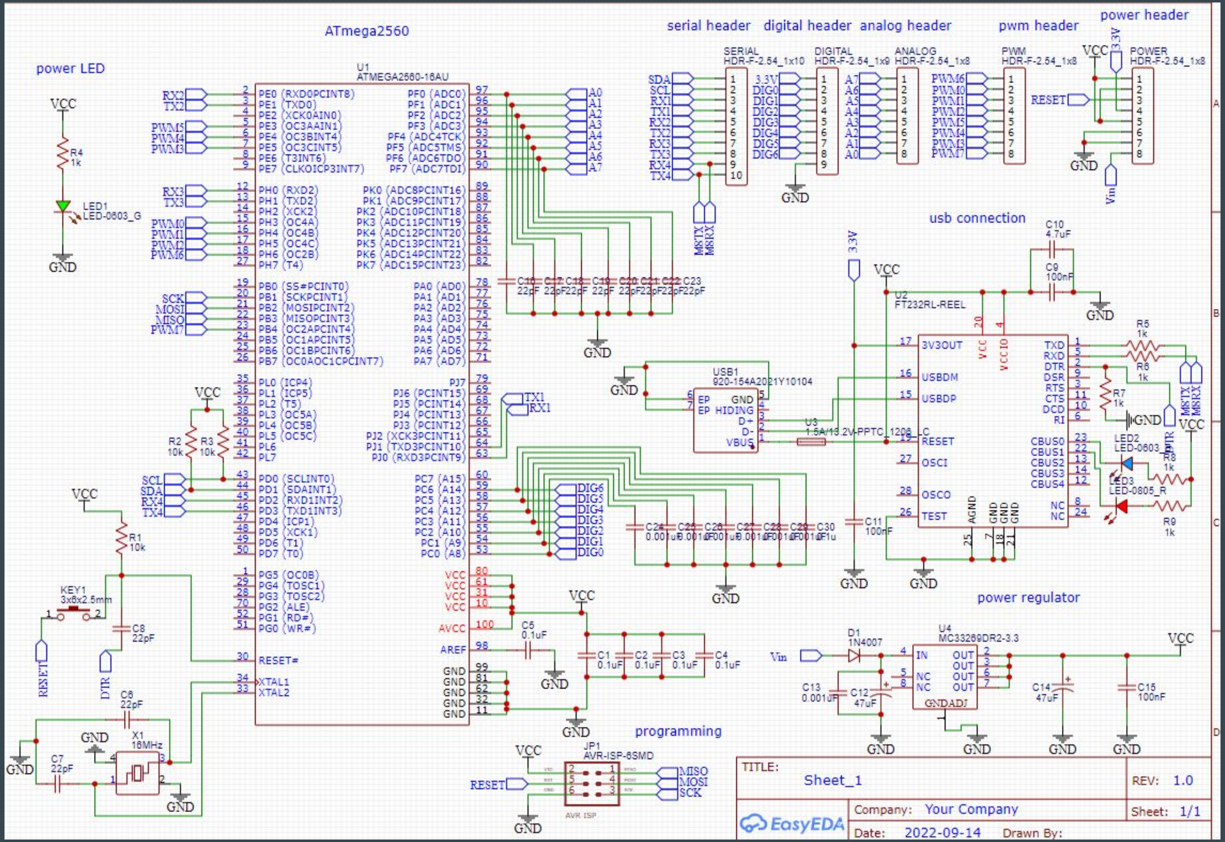


- Back Motors
- 12V
- 350RPM

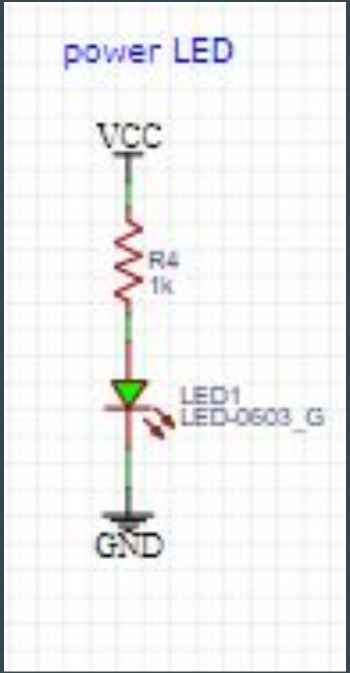


Printed Circuit Board

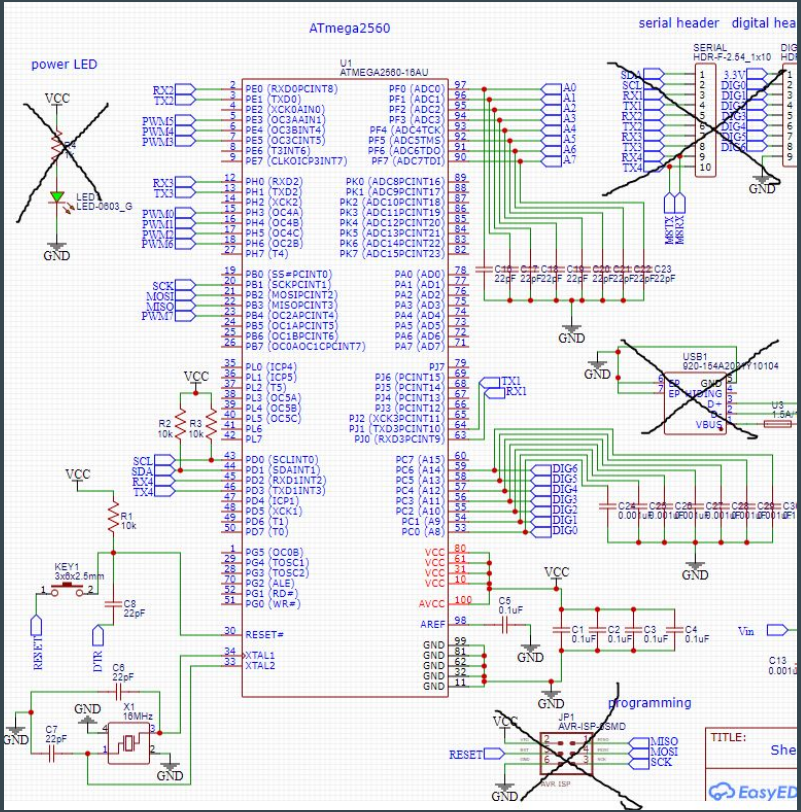
Overall Schematic



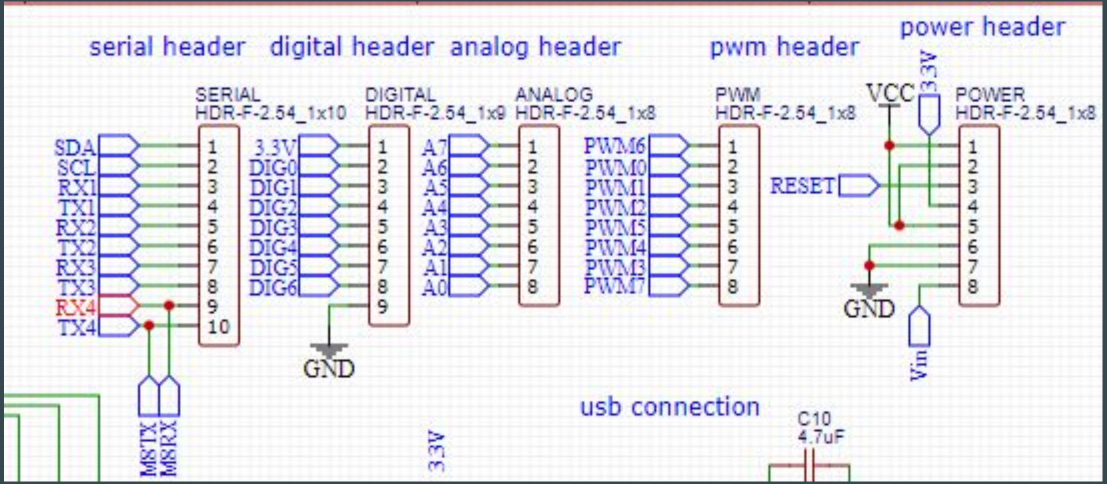
Schematic Power LED



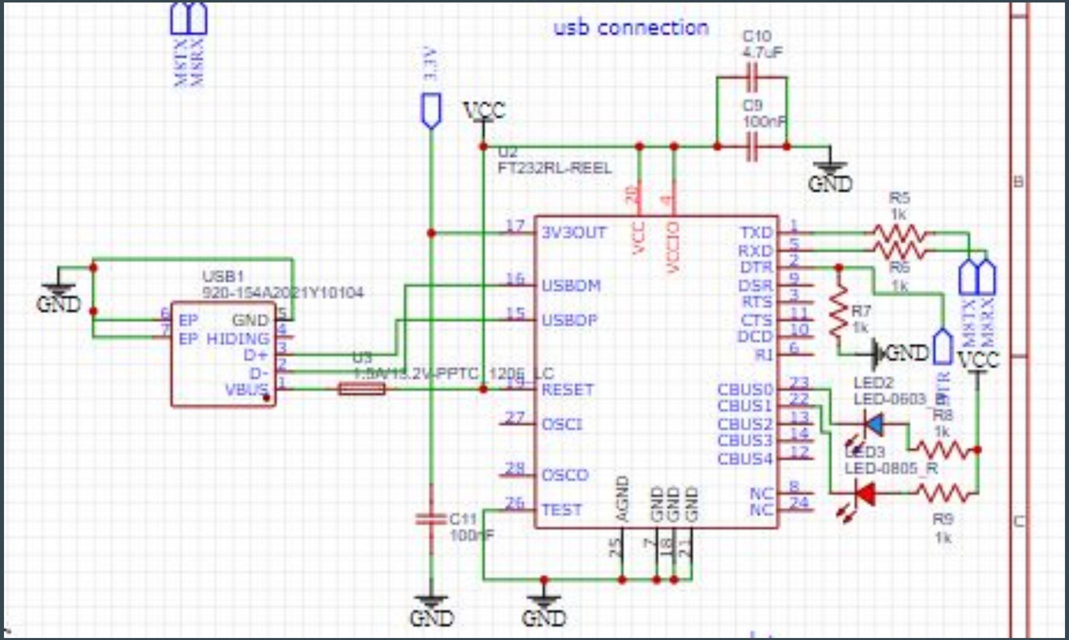
Schematic Microcontroller



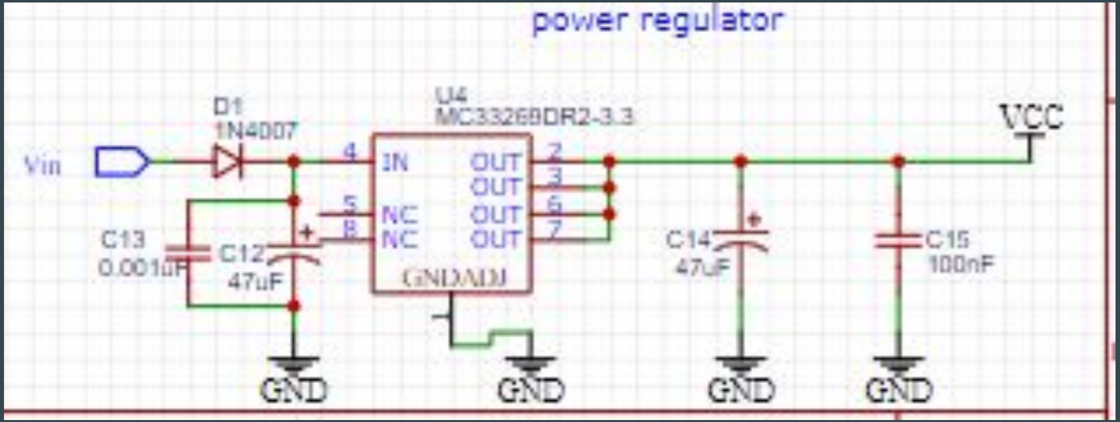
Schematic Headers



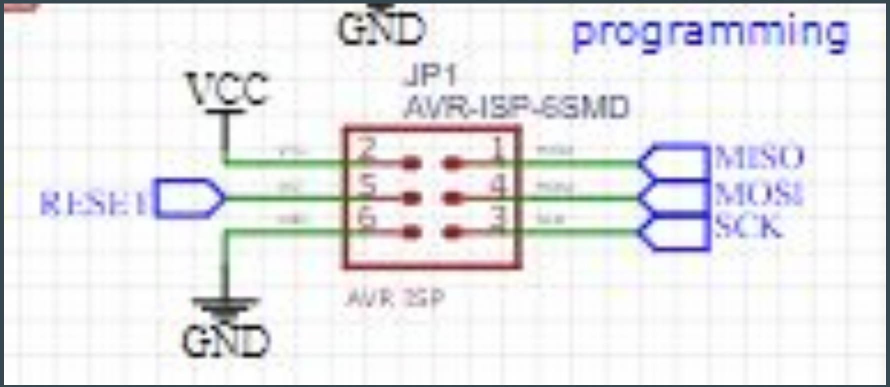
Schematic USB Connection



Schematic Power Regulator

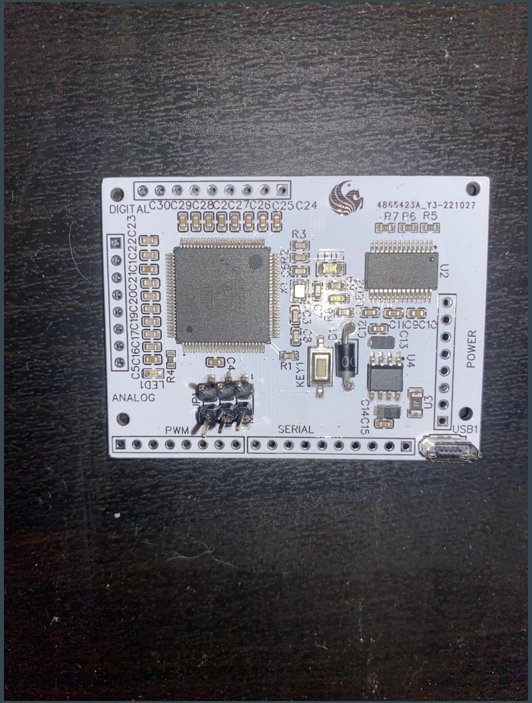


Schematic Programming

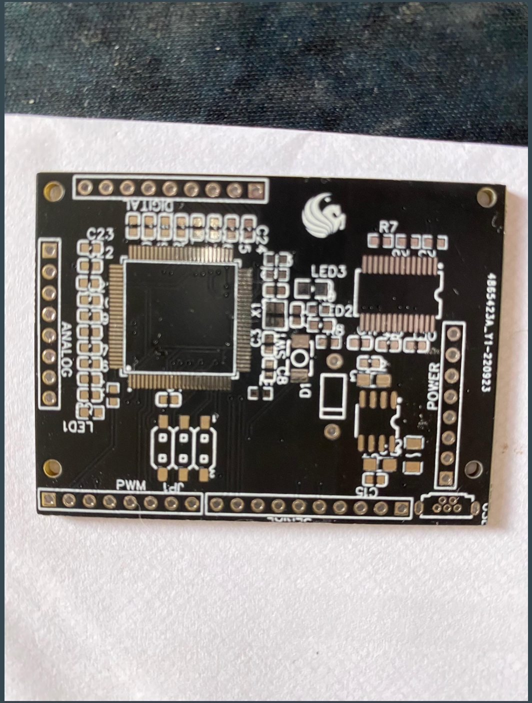


PCB Cont.

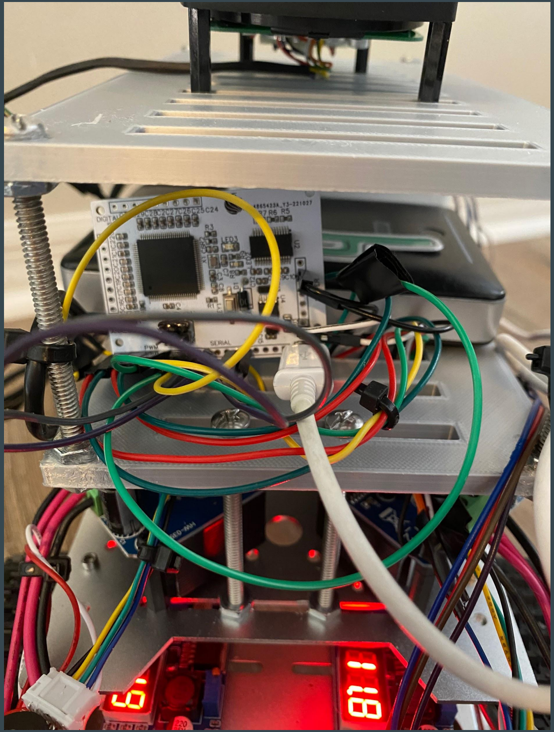
NEW



OLD



PCB



Hardware Challenges

Challenge	Solution
No Camera via Sponsor Recommendation	Use lidar
PCB Footprint Error	Adjusted Footprint to match the tactical switch purchased in the second board layout
PCB Serial Lines	Connected pins RX4 to TX2 and TX4 to RX2
PCB PWM Pins	Added 2 new PWM pins to the PWM header in the second board layout
PCB Ft232 Chip	Learned most of these are counterfeit in the market. Also lots of compatibility issues with windows
ESP32 Board DOA	Purchased a new board from a more reputable distributor

Power Design

Power

Item	Voltage Requirement	Maximum Current
Motors	12 V	1.3 A
BTS7960 Motor Controller	5 V	3 mA
Raspberry Pi 4B	5 V	1.5 - 3 A
RPLIDAR A1	5 V	100 mA
PCB	5V	1 A
GPS	3.3 or 5 V	0.045 A
ESP32	3.3 V	0.15 mA

14.8 V Lipo Battery

The battery is being used to provide power to only the motors since the motors use the most amount of voltage and current to operate. The following is the specs of the battery:

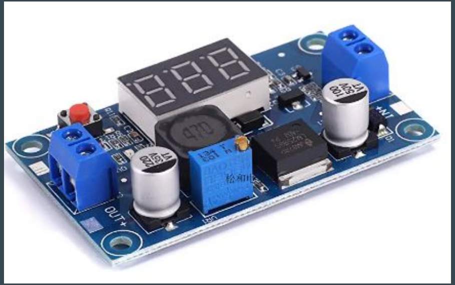
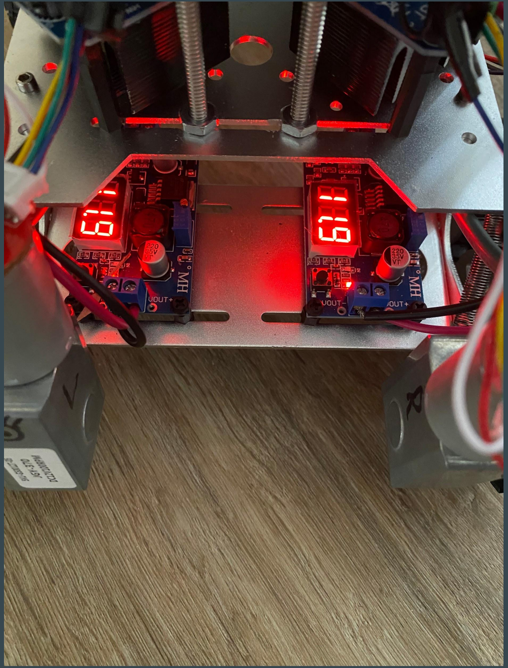
- Voltage: 14.8 V
- Capacity: 9 Ah
- Max Discharge Rate: 100C

DC to DC Buck Converter

The Project required 2 DC to DC Buck Converters both with a 12 volt output. These are attached and powering motors and motor drivers of their respective side.

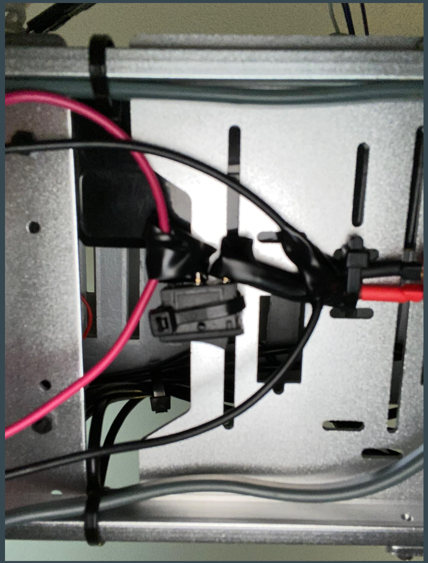
Songhe LM2596S

- Wire only connection
- V-40V to 1.25V-37V
- 2A

Rocker Switch

- There is a rocker switch on the robot that toggles the power
- ESP32 begins running as a server once powered

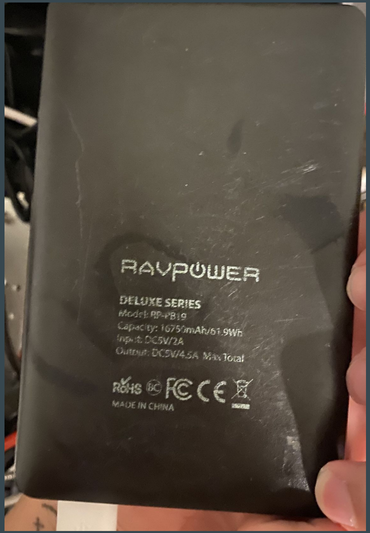


Ravpower Battery Pack

- The battery pack is utilized to power the rest of the electrical systems on the robot
- Does not power the motors
- Easy to use and is rechargeable

Battery Specs

- Voltage: 5 V
- Max current output: 4.5 A
- Battery Capacity: 16.75 Ah



Software Design

Software Design: Overview, ROS Navigation Stack

Environment Mapping

- Used ROS RPLIDAR and ROS Hector SLAM (Simultaneous localization and mapping) packages to map and record results in ROS RVIZ application
- Written with ROS Noetic and Python

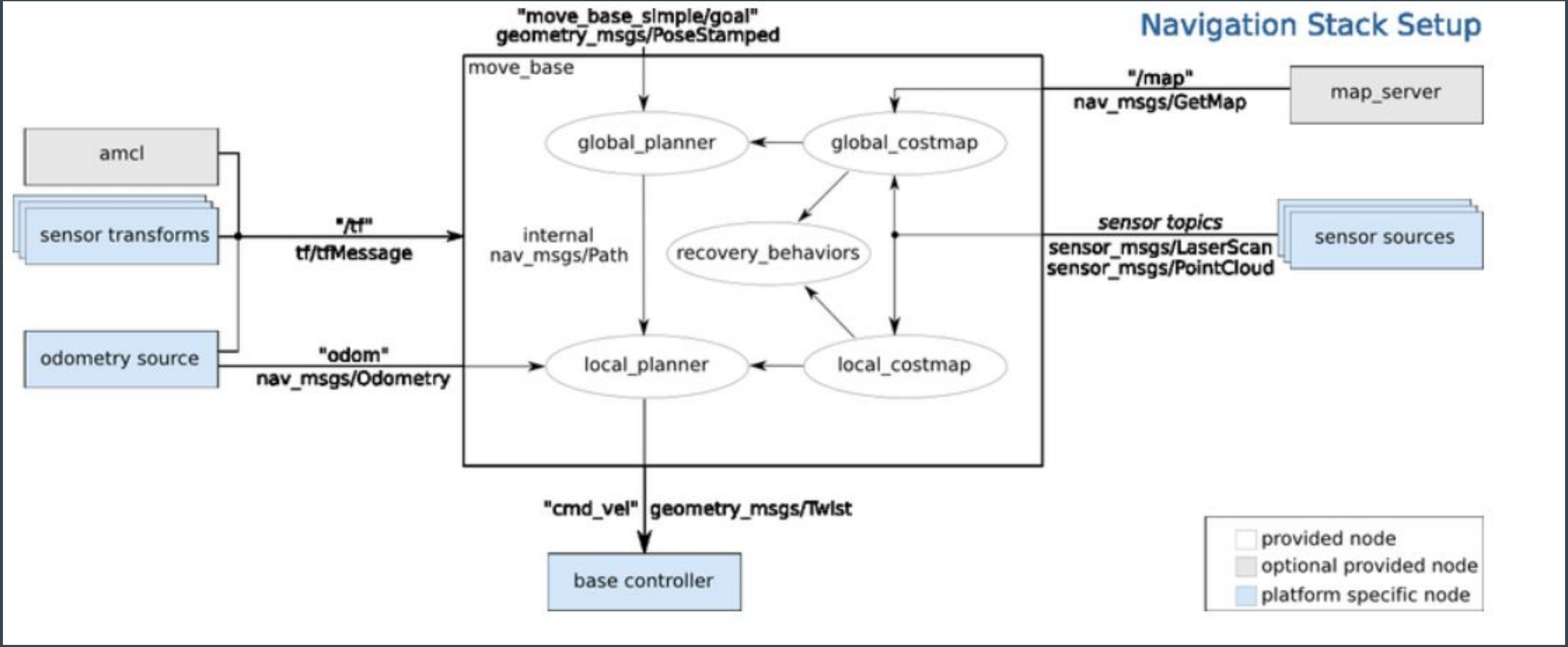
Robot Navigation

- Used ROS RPLIDAR and ROS Hector SLAM packages to navigate throughout environment
- Arduino code written in C++ is uploaded to PCB and sends velocity commands to Raspberry Pi and motor controllers

Object Detection

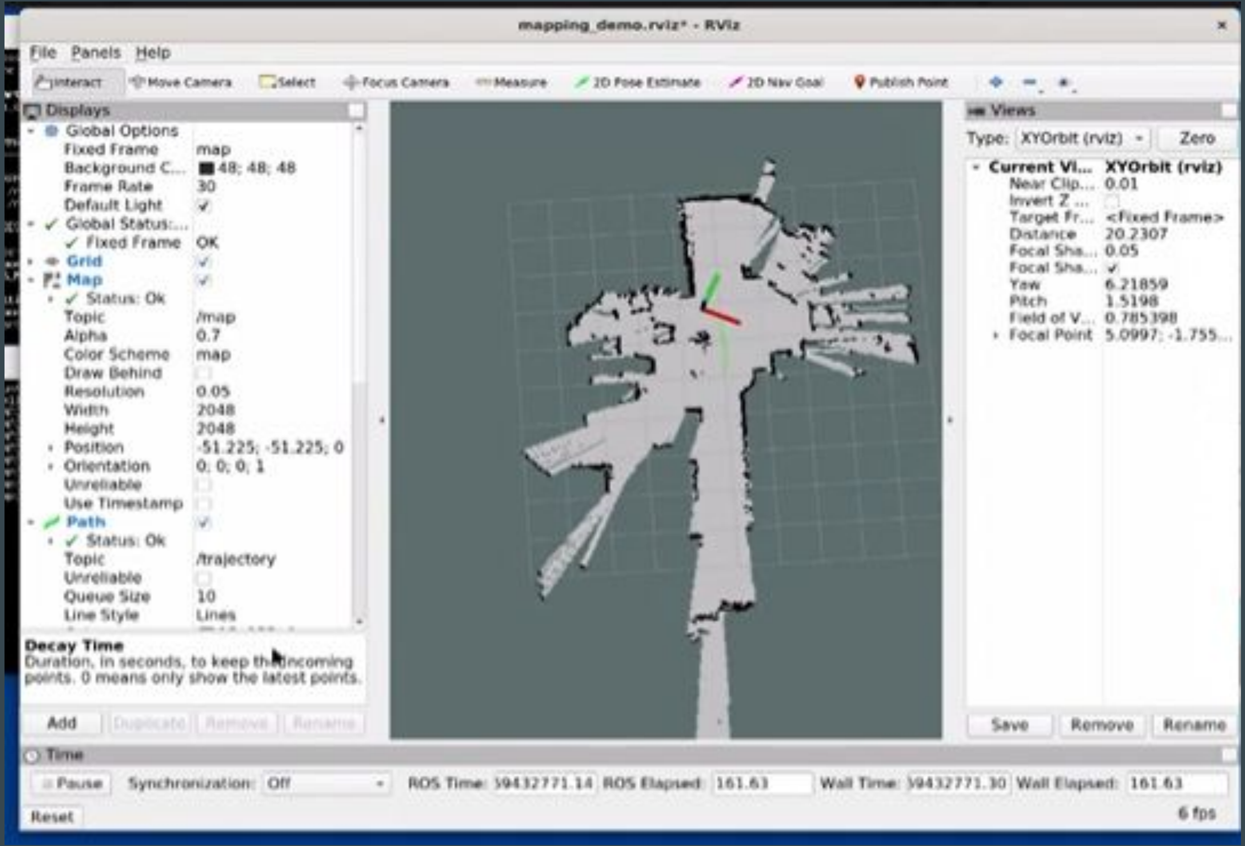
- Used ROS RPLIDAR package to detect objects in path while navigating around environment
- Written with ROS Noetic and Python

Software Design: Robot Operating System (ROS)



Software Design: Environment Mapping

- Map created in RVIZ by robot traveling around environment utilizing lidar data and ROS SLAM package



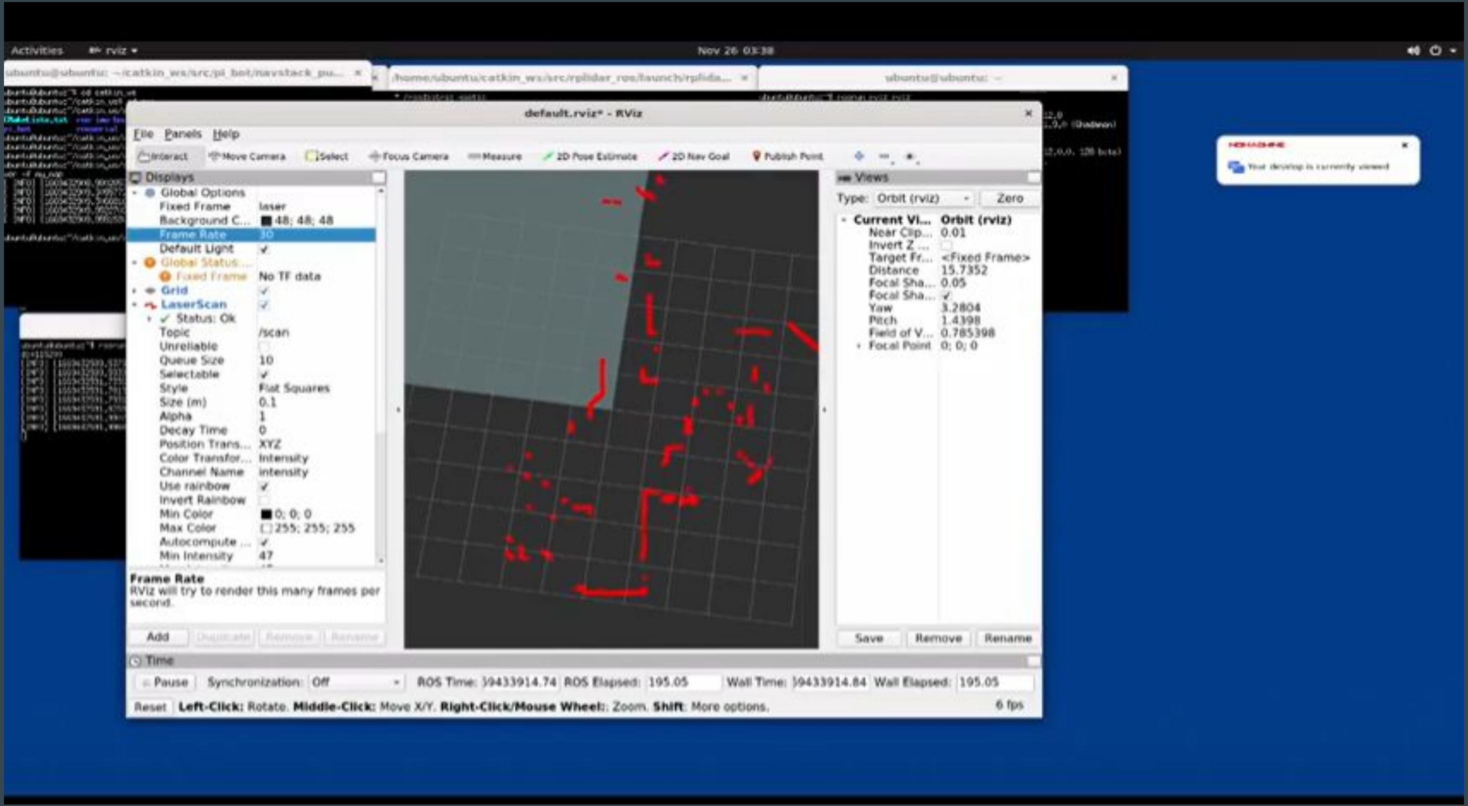
```
Global Options
  Fixed Frame          map
  Background C...      48; 48; 48
  Frame Rate           30
  Default Light        ✓
Global Status:...
  ✓ Fixed Frame        OK
Grid                   ✓
Map                    ✓
  ✓ Status: Ok
  Topic                /map
  Alpha                0.7
  Color Scheme         map
  Draw Behind
  Resolution           0.05
  Width                2048
  Height               2048
  Position             -51.225; -51.225; 0
  Orientation          0; 0; 0; 1
  Unreliable
  Use Timestamp
Path                   ✓
  ✓ Status: Ok
  Topic                /trajectory
  Unreliable
  Queue Size           10
  Line Style           Lines

Decay Time
Duration, in seconds, to keep the incoming points. 0 means only show the latest points.
```

```
Type: XYOrbit (rviz)
Current Vi...    XYOrbit (rviz)
  Near Clip...   0.01
  Invert Z ...
  Target Fr...   <Fixed Frame>
  Distance       20.2307
  Focal Sha...   0.05
  Focal Sha...   ✓
  Yaw            6.21859
  Pitch          1.5198
  Field of V...  0.785398
  Focal Point    5.0997; -1.755...
```

```
Synchronization: Off   ROS Time: 59432771.14   ROS Elapsed: 161.63   Wall Time: 59432771.30   Wall Elapsed: 161.63
6 fps
```

Software Design: Robot Navigation and Object Detection



```
Global Options
  Fixed Frame          laser
  Background C...      48; 48; 48
  Frame Rate           30
  Default Light        ✓
Global Status: ...
  Fixed Frame          No TF data
Grid                   ✓
LaserScan              ✓
  ✓ Status: Ok
  Topic                /scan
  Unreliable
  Queue Size           10
  Selectable           ✓
  Style                Flat Squares
  Size (m)             0.1
  Alpha                1
  Decay Time           0
  Position Trans...    XYZ
  Color Transfor...    Intensity
  Channel Name         intensity
  Use rainbow          ✓
  Invert Rainbow
  Min Color            0; 0; 0
  Max Color            255; 255; 255
  Autocompute ...      ✓
  Min Intensity        47

Frame Rate
RViz will try to render this many frames per second.
```

```
Type: Orbit (rviz)    Zero
Current Vi...   Orbit (rviz)
  Near Clip...  0.01
  Invert Z ...
  Target Fr...  <Fixed Frame>
  Distance      15.7352
  Focal Sha...  0.05
  Focal Sha...  ✓
  Yaw           3.2804
  Pitch         1.4398
  Field of V... 0.785398
  Focal Point   0; 0; 0
```

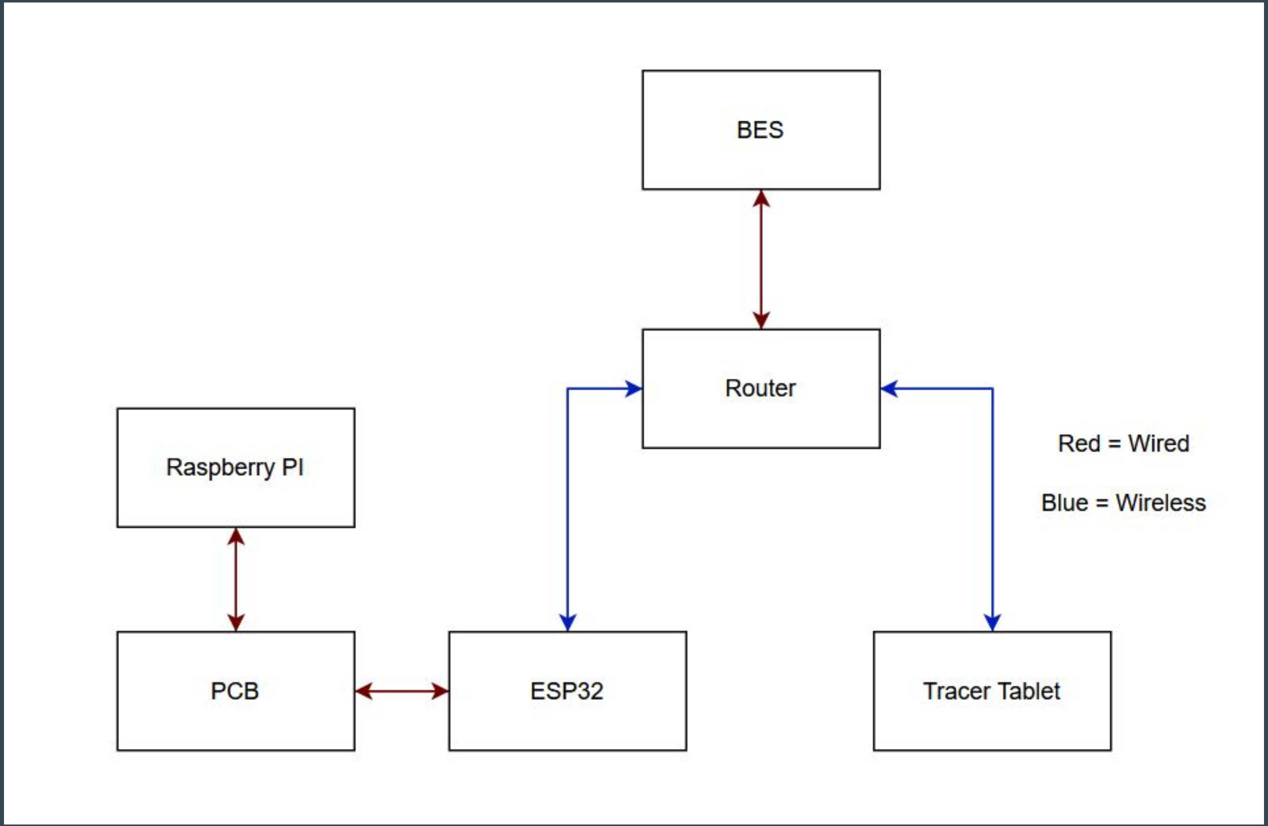
```
Synchronization: Off   ROS Time: 39433914.74   ROS Elapsed: 195.05   Wall Time: 39433914.84   Wall Elapsed: 195.05
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.   6 fps
```

Software Challenges

Challenge	**Solution**
ROS learning curve	Documentation/learning courses
Lidar data not showing in maps	Joined discord community, changed links to visualize data

Communication

Communication Diagram



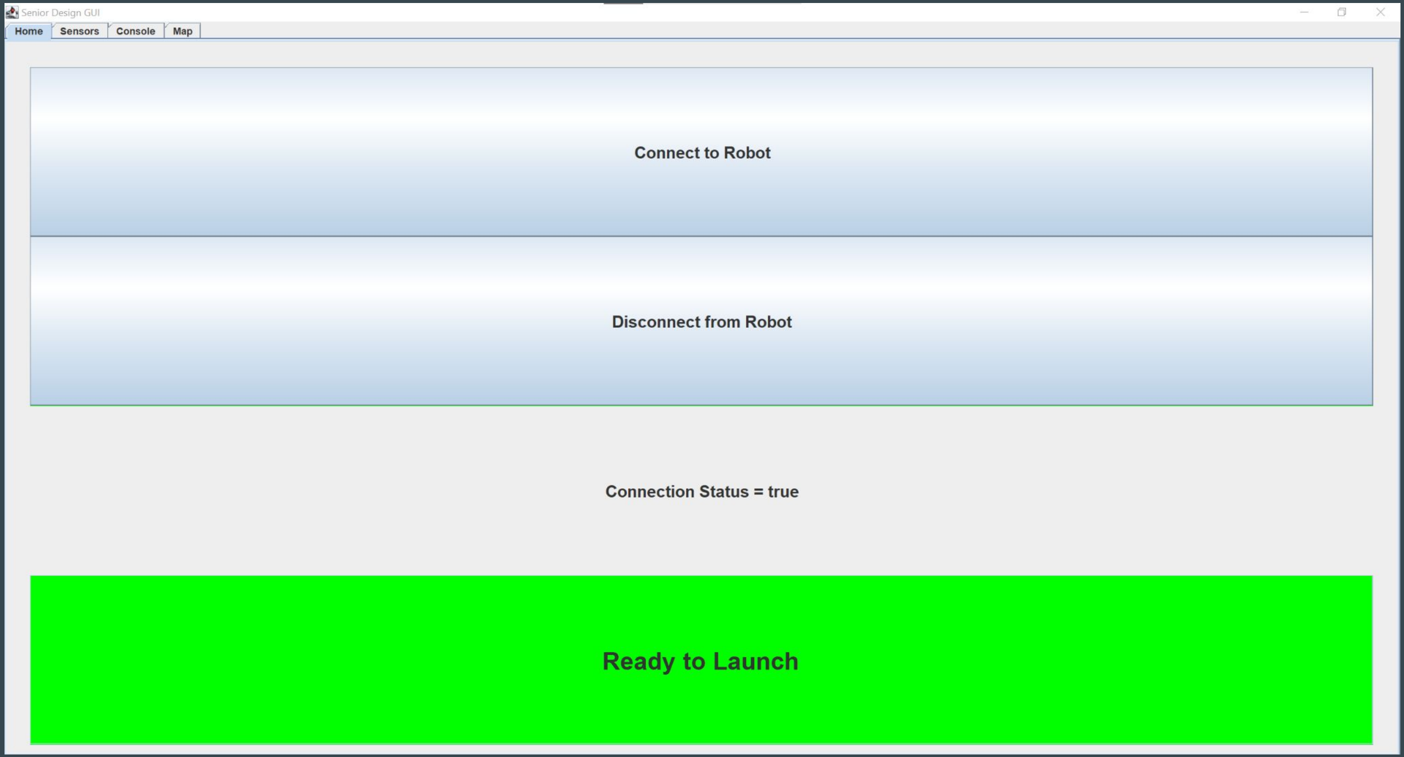
ESP32

- Acts as server for the client (tablet) to connect too when on same network
- GPS coordinates and launch status are sent every 2 seconds to tablet
- Receives a byte (‘A’, ‘B’, ‘C’, or ‘0’) signalling which site to travel too
- Powered by Arduino and code is written in C++
- Link to the code: https://github.com/jrymkos/ESP32_WIFI

GUI

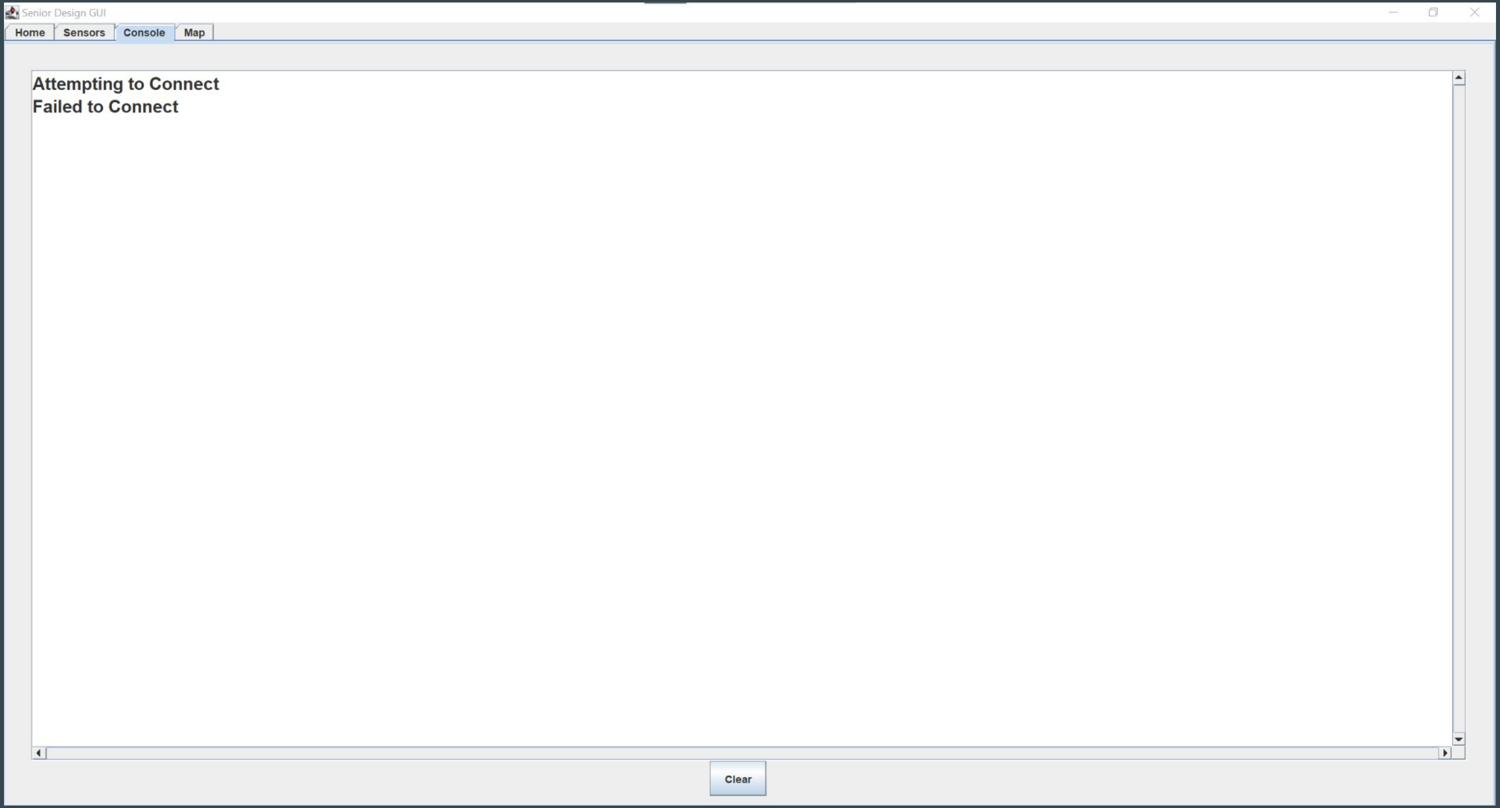
- Created in Java using Swing and Jxmapviewer2 libraries
- Ran as a client connecting to ESP32 (server) on the tablet
- Four main tabs including home, sensor, console, and map
- Able to handle either client or server disconnecting and recover
- Future work includes directly integrating with BES software and adding sensors
- Link to the code: https://github.com/jrymkos/SeniorGUI

GUI Images



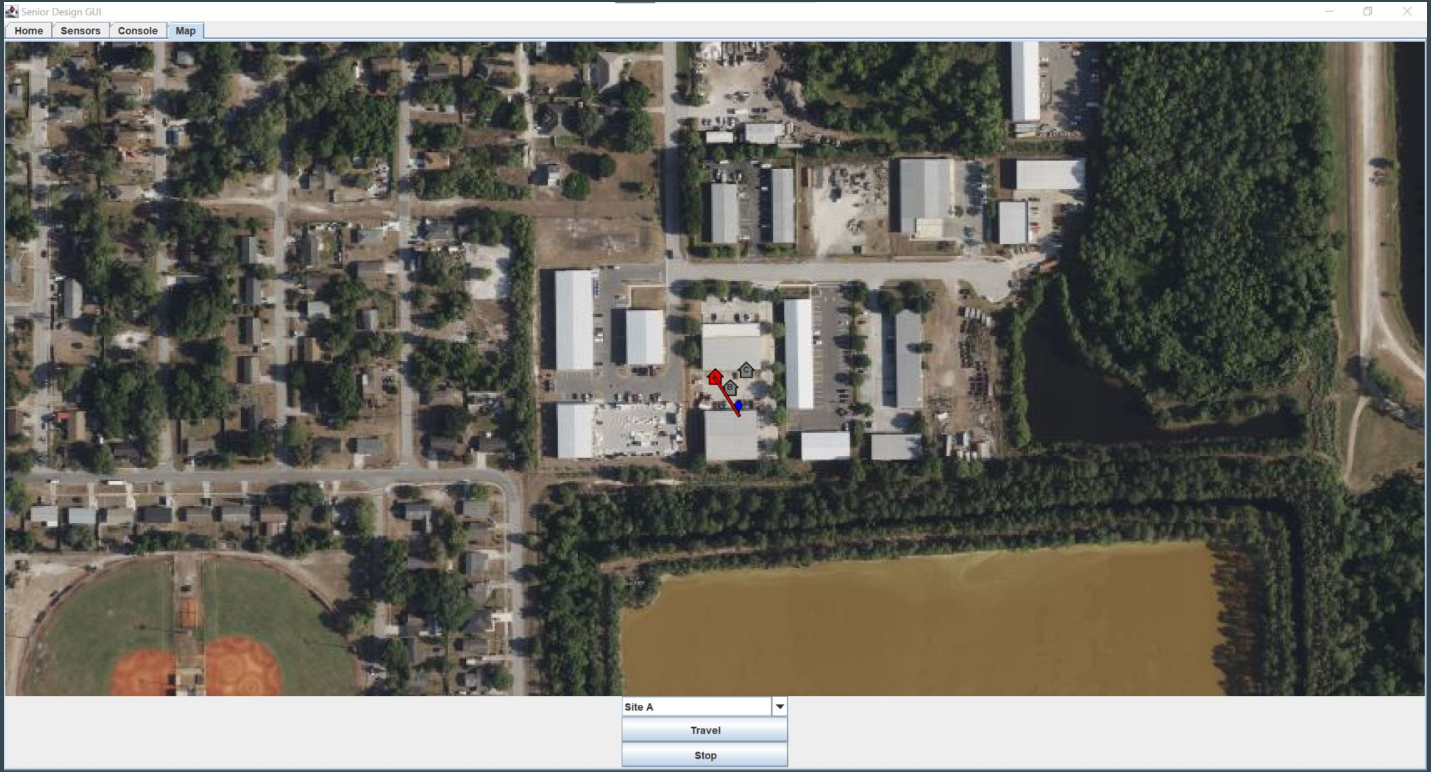
Connection Status = true

GUI Images



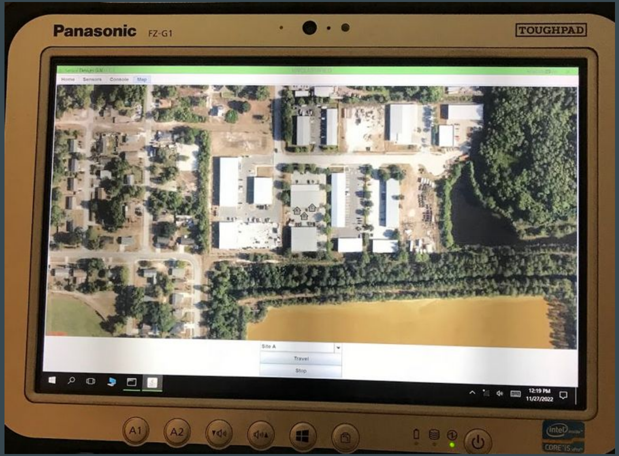
```
Attempting to Connect
Failed to Connect
```

GUI Images


Senior Design GUI
Home
Sensors
Console
Map
Site A
Travel
Stop

BES Tracer Tablet

- It is windows 10 which allows us to convert the GUI into a JAR file and run on it
- Also runs the proprietary software that fires the BES
- A test BES is used that does not fire any shots and instead makes a loud beep



Communication Challenges

Challenge	Solution
Bluetooth in Java	Using Sockets through WIFI
Disconnecting gracefully	Building a timer to check if bytes are received
Identifying which component in buffer	Using an identifier byte like ‘X’ or ‘Y’
Talking with BES without being directly wired	Wire BES to router and have tablet/ESP32 connect
Running the GUI JAR file on tablet	Configuring a specific java version to run for our GUI using bash

Prototyping and Testing

Prototyping

- The concept that is being tested is the robot driving, robot connection with the BES and tablet, and identifying if objects is in the robot's path

Testing

- Robot driving was tested by placing the robot in the testing site and have it move towards a specified destination.
- Identifying an object in the robot's path was tested by having the robot analyze the surrounding area.
- Used IMU to verify if robot is leveled.
- Connection with the BES and tablet was tested by making sure that the BES was seen in the tracer range simulation program as well as being fired through the command driver.

BUDGET	
NAME	**PRICE (TAX not included)**
CanaKit - Raspberry PI 4	$134.99
OAK-D Lite	$149.00
ESP32	$55.00
BNO055	$33.70
TFmini Plus	$49.90
360 LIDAR	$99.19
Robot Base	$109.99
Motor Controller	$15.99
Zeee 9000mAh	$83.69

BUDGET	
NAME	**PRICE (TAX not included)**
ESP32 with Ethernet	$15.99
ESP32 module	$10.99
Solder Paste	$7.95
Flux Paste	$12.45
Battery Connecters	$8.99
VK-162 GPS Mous	$15.49
Solder Wick	$11.50

BUDGET	
NAME	**PRICE (TAX not included)**
LM2596 Buck Converter	$5.15
Motors with Encoders	$32.68
Rocker Switch	$1.60
PCB Parts	$42.54
PCB	$47.06
TOTAL	
SPENT	**$943.84**
REMAINS	**$56.16**

Future Plans

PEO STRI has expressed interest in continuing to sponsor UCF senior design teams to continue and evolve this concept/project over multiple phases.

- Continue to improve accuracy of autonomous navigation
- Improve obstacle detection/clearing of obstacles that are no longer in robot path
- Camera integration to detect humans vs moving objects
- Build larger scale prototype

Thank you for watching our final presentation. We look forward to answering any questions during our final presentation Zoom meeting.