ELG4913 Progress Review Presentation

Group 5 - Search and Rescue Remote-Controlled Car with Life Detection

Fatmah Bayrli 300159193

Papa Kane 300090159

Walid Rashad 300205109

Julien Kapro 300173066

Moktar Abdillahi-Abdi 300156852

Geoffrey Hooton 300187367

Introduction

Project Goal

- One of the main challenges faced by search and rescue teams after an earthquake is the inability of many modern sensors to detect victims buried beneath several meters of heavy concrete.
- **Our Objective:**
- Develop an RC car that can maneuver through a collapsed building, improving the chances for sensors to work effectively.
- Integrate multiple sensors to accurately identify signs of life.

User-Friendly Navigation Interface:	Detecting Heat Beneath the Surface:	Detecting Underground Vibrations:
Create an intuitive interface for controlling the car.	A thermal camera will capture sub-surface heat data, relayed to the operator.	A seismic sensor will detect sub-surface vibrations, with data sent to the operator.
LiDAR data will provide a 3D map of the surroundings to the operator.	The system alerts the operator if a heat signature is detected.	The system alerts the operator if unusual vibrations are identified.
This allows the operator to easily navigate through complex debris.		

1. Problem
2. Solution
3. Market Demand
4. Financial Viability
5. Competitive Advantage
6. Social Impact

Business Case

Customer and how/where is the project used?

- The RC vehicle is built for disaster zones like earthquakes, providing fast mobility and advanced search and rescue capabilities for effective emergency response.

Customer	Application
Government Agencies	Deployed by military or civil defense units for search and rescue operations following natural disasters.
Private Sector Companies	• Companies specializing in earthquake response technologies could integrate this vehicle into their solutions. • Example: Companies like **Tempest Technology Corp.**
Non-Governmental Organizations (NGOs)	Humanitarian groups and NGOs involved in disaster relief efforts can use the vehicle for rescue missions, helping to locate and aid victims in disaster zones.

Requirements Specification

Foreseeable Features/Challenges

- User interface for remote control on a central computer (laptop).
- Intuitive remote-control UI with easy steering and acceleration controls.
- Detailed, real-time 3D map display of the vehicle's surroundings for navigation.
- Reliable communication of sensor data to the operator.
- Detection of human movements/vibrations underground.
- Detection of heat signatures underground.
- Rugged and durable design.
- Navigation in constrained environments and over uneven terrain.
- **Rocker-Bogie suspension/chassis**

Non-Included Features

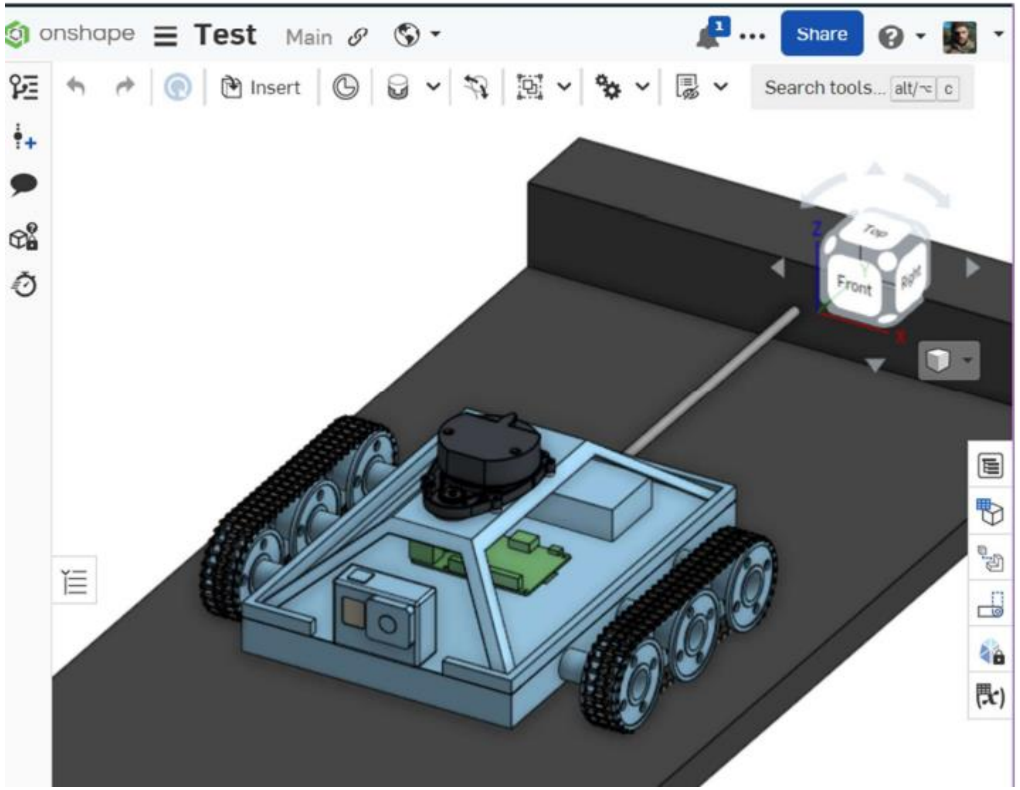
- Detailed mapping of the entire collapsed structure (focus is put on immediate surroundings).
- Interaction with objects other than detecting signs of life.

Constraints

- The car's design will incorporate a remote emergency shut-off mechanism.
- The vehicle will have a physical on/off switch and a start command on the central computer.
- The system will be portable and easily deployed.
- The RC car will be properly grounded to eliminate the possibility of sparking.
- The car’s chassis will include emergency lighting LEDs for enhanced visibility.
- The vehicle’s design will adhere to Canadian safety standards for electrical and material safety.

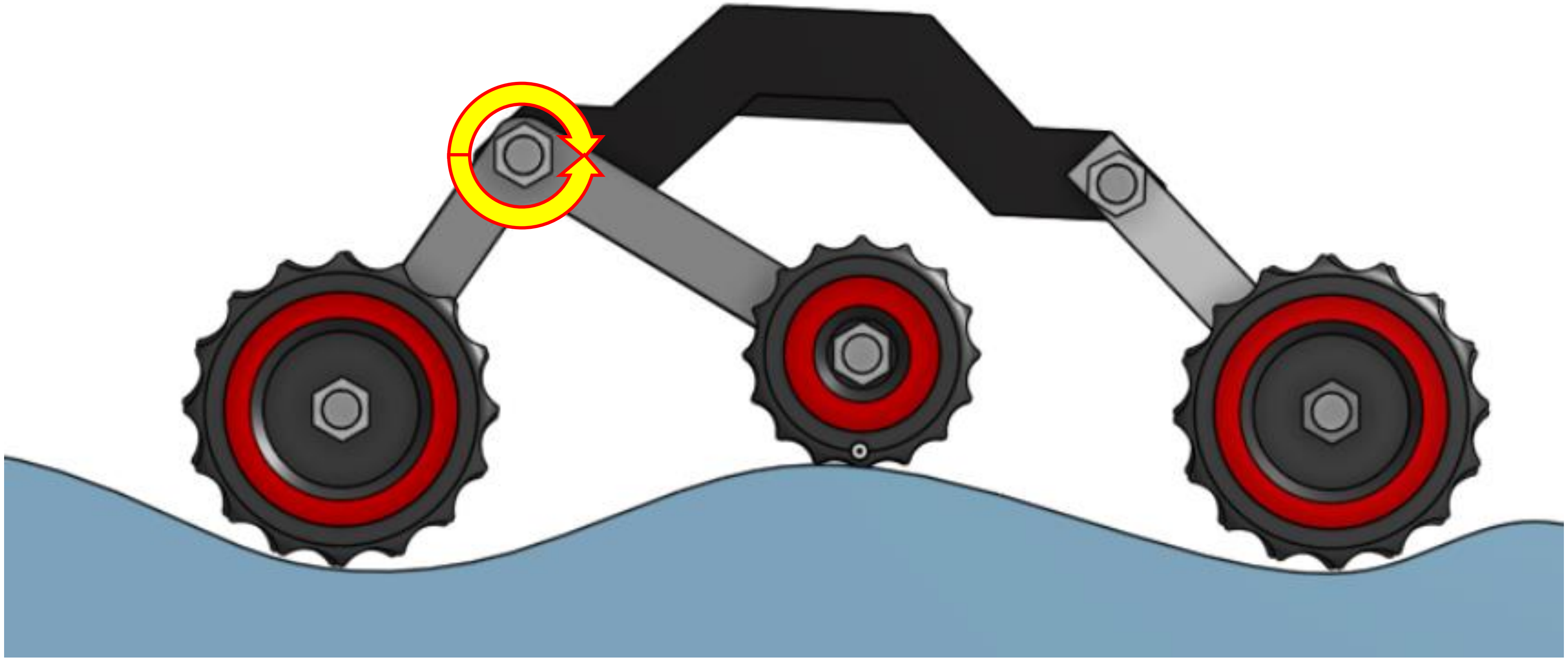
Notable Modifications: suspension and thermal imagery

- Old chassis



- New suspension/chassis

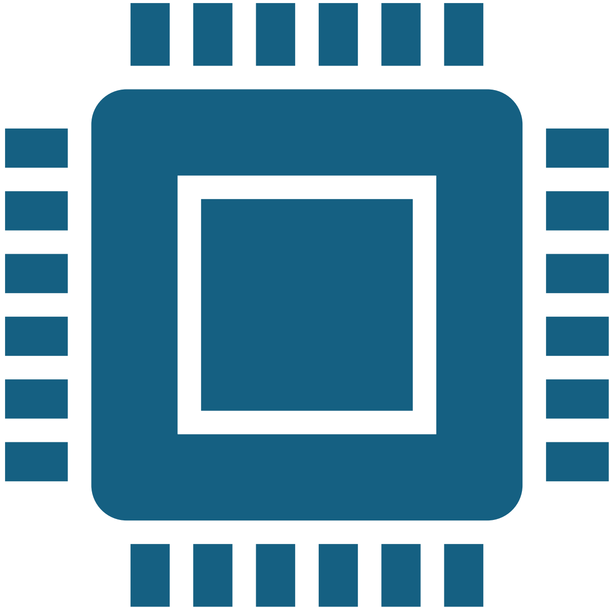
Rocker-Bogie Suspension



Progress Overview

Tasks Completed

- Ordered/Received components
- Set up Raspberry Pi
- Built battery circuit
- Initial thermal camera test
- Initial LiDAR test
- Initial Geophone test
- First iteration of data acquisition system
- Design of vehicle suspension



Tasks to be Completed

- Integrate all sensors with Raspberry Pi simultaneously
  - Code
  - Circuit
    - Connect I2C components on the same line
- Controlling vehicle motors with Raspberry Pi
- Continue building chassis
- Design method for controlling the vehicle (UI/remote controller) and integration with motors
- Integrate sensors with ROS2
  - Allows us to transmit data from Raspberry Pi to main computer using ethernet
- Continue developing data acquisition system for all sensors

Current Distribution of Tasks

Group Member	Tasks Completed	In Progress/Next Task
Moktar	• Power circuit	• Controlling motors • Simultaneous sensor integration
Papa	• Suspension Optimization Research	• Chassis Design • Vehicle Controller/UI
Fatmah	• LiDAR Initial Test	• ROS2 Development • Vehicle Controller/UI
Walid	• Suspension Design	• Chassis Design • Vehicle Controller/UI
Julien	• Temperature data analysis and storage in CSV files	• Further developing data analysis tools
Geoffrey	• LiDAR Initial Test • Thermal Camera Initial Test • Geophone Initial Test	• Simultaneous sensor integration • ROS2 Integration

Risk Analysis Plan

Risk	Probability	Initial Safety Risk Level	Impact	Mitigations	Residual Risk Level
Battery runs out	High	High	High	Ensure adequate battery charge before deployment	Medium
Connection Loss	High	High	High	Verify stable communication protocol efficiency. Test wireless communication stability over required range under different conditions.	Medium
Sensor's failure	Medium	High	High	Utilize ROS to integrate multiple redundant sensors (LIDAR, thermal) and fuse their data for more reliable detection and navigation. Develop ROS-based routines to check sensor outputs against expected patterns for failure.	Low
Navigation Error	Medium	High	High	Calibrate and test the LIDAR and navigation systems using ROS in real-world mapping scenarios. Implement advanced path planning algorithms within ROS that dynamically adjust the vehicle's route based on real-time sensor inputs to ensure accurate and reliable navigation.	Medium
Group member sick	High	Medium	Medium	Shift work to others.	Low
Data integrity issues	Medium	Medium	Medium	Encrypt all data transmitted via I2C using ROS security packages. Set up local backup systems that periodically sync with the main data repository to prevent data loss.	Low
Critical Components failure(e.g servo motor,	Medium	High	High	Select high-reliability components tested for compatibility with ROS. Set up ROS-based telemetry to monitor real-time performance and predict failures before they occur.	Low

Legal Team Approval Status

- Compliance Check :

  - Environmental Regulations : Adhere to environmental laws regarding battery usage during testing.

  - Safety Standards : Ensure the RC car meet all the mechanical and electrical safety standards to prevent any accidents. Conduct regular safety audits to maintain compliance, crucial for students' safety.

- Liability Assessments :

  - Risk of Accidents : Assess potential liabilities from malfunctions. Implement rigorous testing and emergency stop features to mitigates risks.

- Approval and Documentations : Maintain comprehensive records of all modifications and approvals.

Test Plan

- **Hardware Testing :**

Setup: Connect each sensor and motor controller to a Raspberry Pi.

Functionality Test: Use Python scripts to validate communication and response accuracy with the Raspberry Pi.

Validation: Devices passing functionality tests are approved for assembly; failing devices are retested.

- **Software Testing**

Connectivity Test: Establish and verify connections between the Raspberry Pi and hardware components.

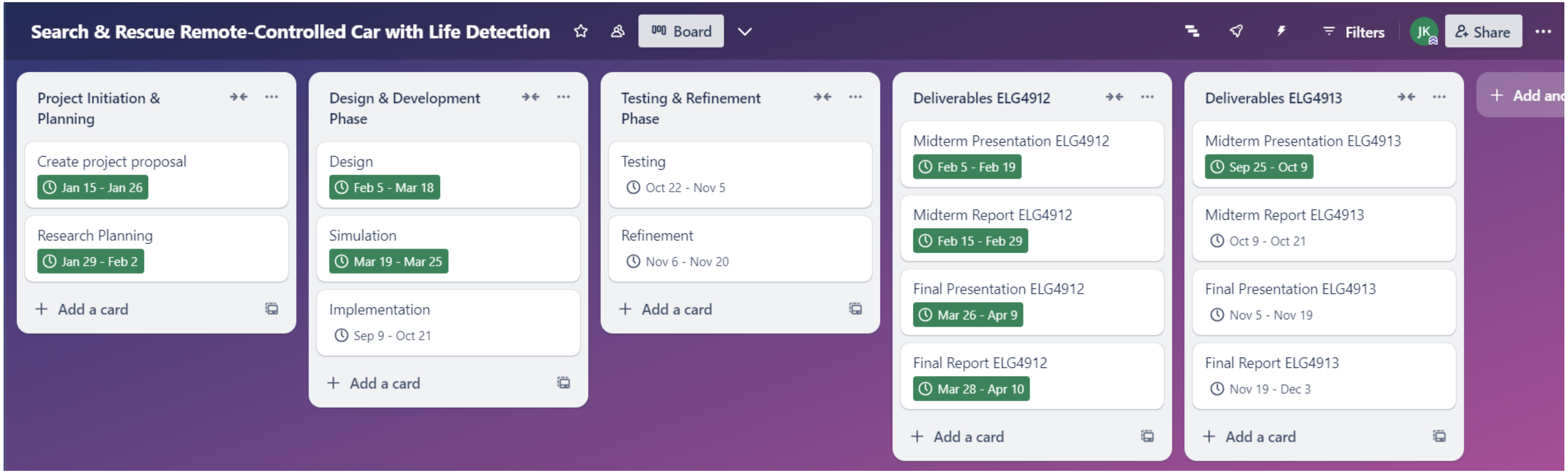
Integration Test: Load all embedded software, ensure individual and combined hardware operations via controls.

- **Communication Testing**

Protocol Verification: Confirm that communication protocols between sensors, Raspberry Pi, and motor controllers are consistent and reliable.

Data Integrity: Ensure data transmitted between components is accurate and timely, with no losses or errors.

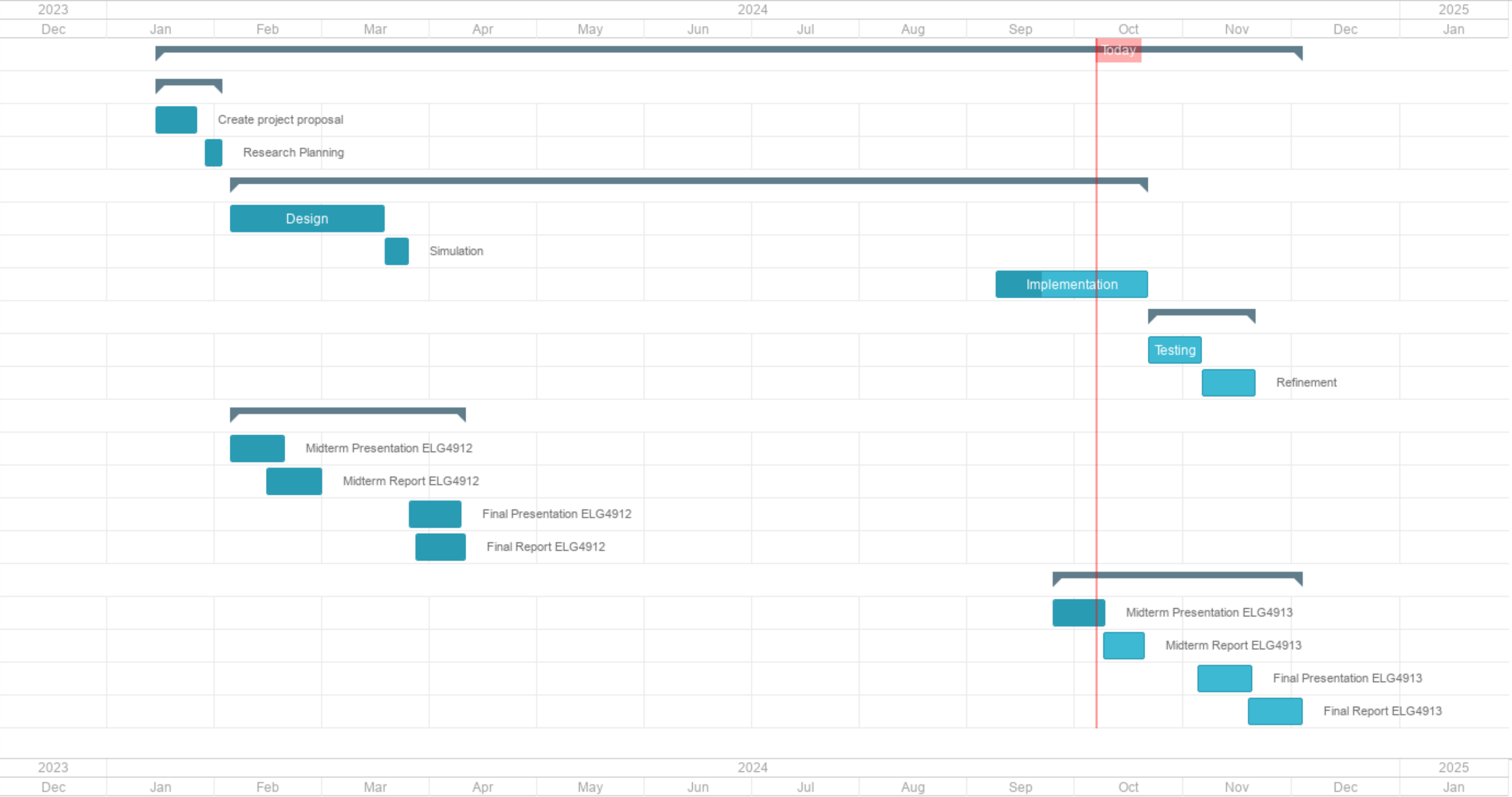
Schedule & Budget Outlook


Search & Rescue Remote-Controlled Car with Life Detection
Board
Filters
JK
Share
Project Initiation & Planning
Create project proposal
Jan 15 - Jan 26
Research Planning
Jan 29 - Feb 2
Add a card
Design & Development Phase
Design
Feb 5 - Mar 18
Simulation
Mar 19 - Mar 25
Implementation
Sep 9 - Oct 21
Add a card
Testing & Refinement Phase
Testing
Oct 22 - Nov 5
Refinement
Nov 6 - Nov 20
Add a card
Deliverables ELG4912
Midterm Presentation ELG4912
Feb 5 - Feb 19
Midterm Report ELG4912
Feb 15 - Feb 29
Final Presentation ELG4912
Mar 26 - Apr 9
Final Report ELG4912
Mar 28 - Apr 10
Add a card
Deliverables ELG4913
Midterm Presentation ELG4913
Sep 25 - Oct 9
Midterm Report ELG4913
Oct 9 - Oct 21
Final Presentation ELG4913
Nov 5 - Nov 19
Final Report ELG4913
Nov 19 - Dec 3
Add a card

Gantt Chart

	Status	%	Dur
⊟ **Search & Rescue Remote**			46w 2d
⊟ **Project Initiation & Planning**			3w
Create project proposal	Completed	100%	2w
Research Planning	Completed	100%	1w
⊟ **Design & Development Phase**			37w 1d
Design	Completed	100%	6w 1d
Simulation	Completed	100%	1w
Implementation	Open	30%	6w 1d
⊟ **Testing & Refinement Phase**			4w 2d
Testing	Open	0%	2w 1d
Refinement	Open	0%	2w 1d
⊟ **Deliverables ELG4912**			9w 3d
Midterm Presentation ELG49	Completed	100%	2w 1d
Midterm Report ELG4912	Completed	100%	2w 1d
Final Presentation ELG4912	Completed	100%	2w 1d
Final Report ELG4912	Completed	100%	2w
⊟ **Deliverables ELG4913**			10w
Midterm Presentation ELG49	Completed	100%	2w 5h 30m
Midterm Report ELG4913	Open	0%	1w 3d
Final Presentation ELG4913	Open	0%	2w 1d
Final Report ELG4913	Open	0%	2w 1d

Name



Exported from Placker.com on Oct 6th 19:01

Budget

Component	Cost
Slamtec RPLIDAR Sensor	$ -
Thermal Camera	$ 100.21
Geophone SM-24	$ 87.68
ON/OFF Switch	$ -
Servo Controllers	$ -
Raspberry Pi	$ -
Software (SBC, simulation environments...)	$ -
Servo motors	$ -
Acrylic glass base with 3 pairs of wheels	$ -
Battery 7.4V	$ -
Voltage Regulator	$ -
Ethernet Cable (100ft)	$ 25.88
ADS1115 ADC	$ 20.32
Ultrasonic Sensor	$ -
Total	**$ 234.09**

Data Acquisition System (DAQ)

```python
#Thermal Cam DAS

import pithermalcam as ptc
import csv
import time
import board
import busio

#Need to initialize the camera connected to the i2c bus
i2c_bus = busio.I2C(board.SCL,board.SDA,frequency=800000)
mlx = ptc.pi_therm_cam.adafruit_mlx90640.MLX90640(i2c_bus)

def write_csv(data, filename = 'avg_temp.csv'):
    with open(filename, mode = 'a', newline = '') as file:
        myWriter = csv.writer(file)
        myWriter.writerow(data)

write_csv(["Date & Time", "Avg temp C", "Message"])
#The camera frame is a 32x24 matrix, so we initialize to 768=32x24
frame = [0]*768
while True:

    #Calculating the avg temp in a frame
    mlx.getFrame(frame)
    avg_temp_C = sum(frame)/len(frame)
    message = 'N/A'
    if avg_temp_C > 30 and avg_temp_C<50:
        print("There may be a human there.")
        message = "There may be a human there."
    elif avg_temp_C >200:
        print("Don't go further. There may be a fire up ahead.")
        message = "Don't go further. There may be a fire up ahead."
    date_time = time.strftime('%Y-%m-%d %H:%M:%S')

    app_row = [date_time, avg_temp_C, message]
    write_csv(app_row)
    #Get data every 5 seconds
    time.sleep(5)
```

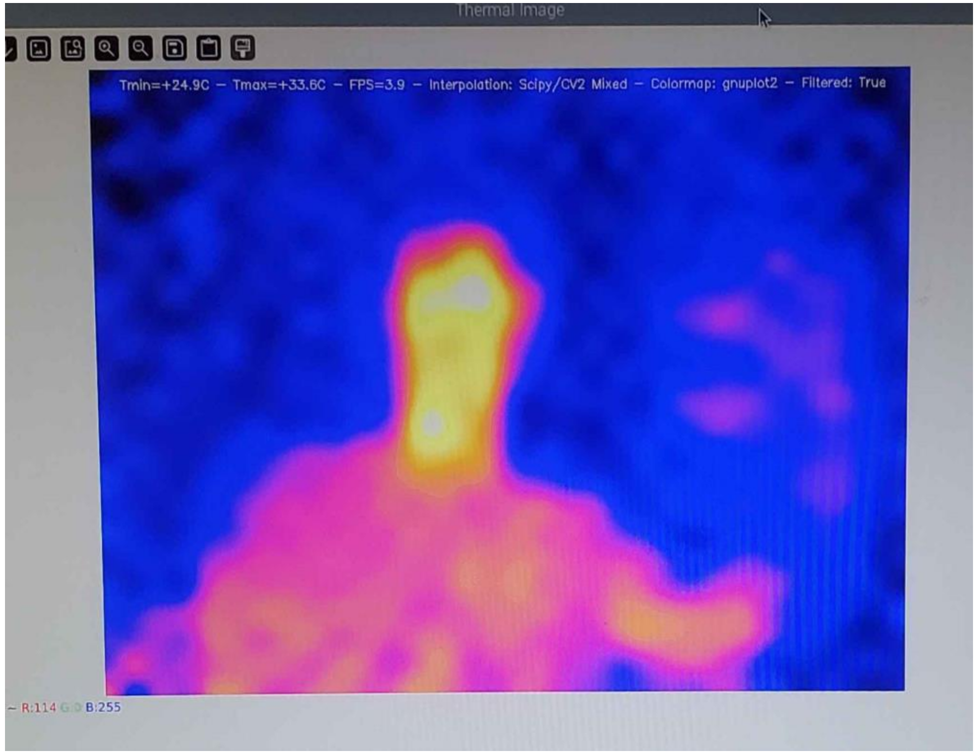
Date & time	Avg temp C	Message
2024-10-06 18:26	27.92964153	N/A
2024-10-06 18:26	27.95298826	N/A
2024-10-06 18:26	27.94995239	N/A
2024-10-06 18:26	28.48186942	N/A
2024-10-06 18:27	28.72624848	N/A
2024-10-06 18:27	27.53844653	N/A
2024-10-06 18:27	28.80652479	N/A
2024-10-06 18:27	28.85844877	N/A
2024-10-06 18:27	34.8917658	There may be a human there.
2024-10-06 18:27	35.52517178	There may be a human there.
2024-10-06 18:27	27.55785042	N/A
2024-10-06 18:27	27.20409141	N/A
2024-10-06 18:27	29.13072519	N/A
2024-10-06 18:27	31.88137963	There may be a human there.
2024-10-06 18:27	29.44009885	N/A

Demos

MLX90640 Thermal Camera

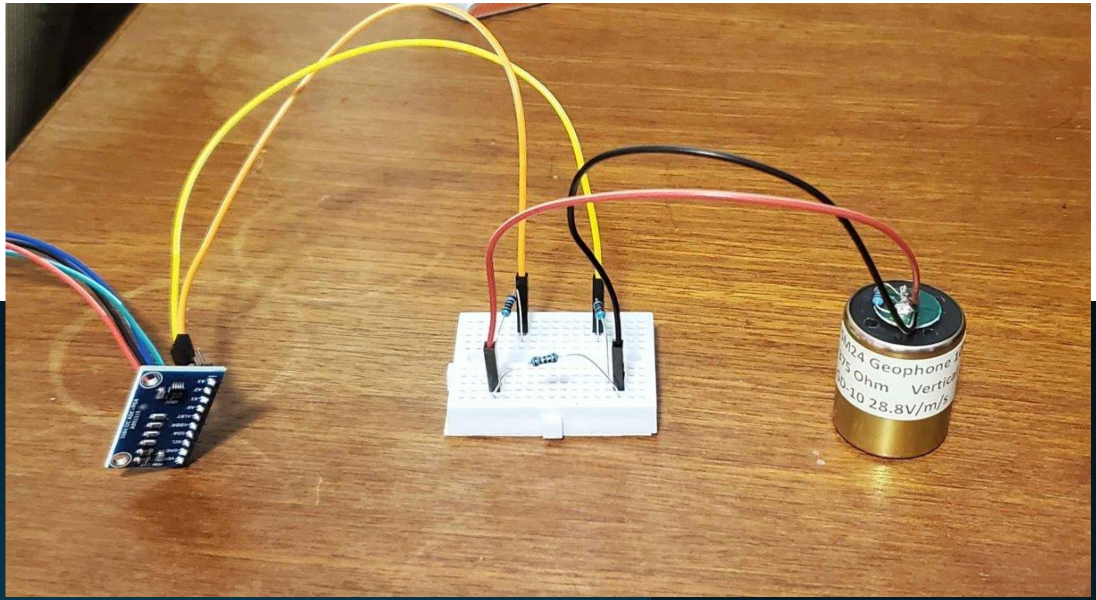
- I2C Protocol Camera

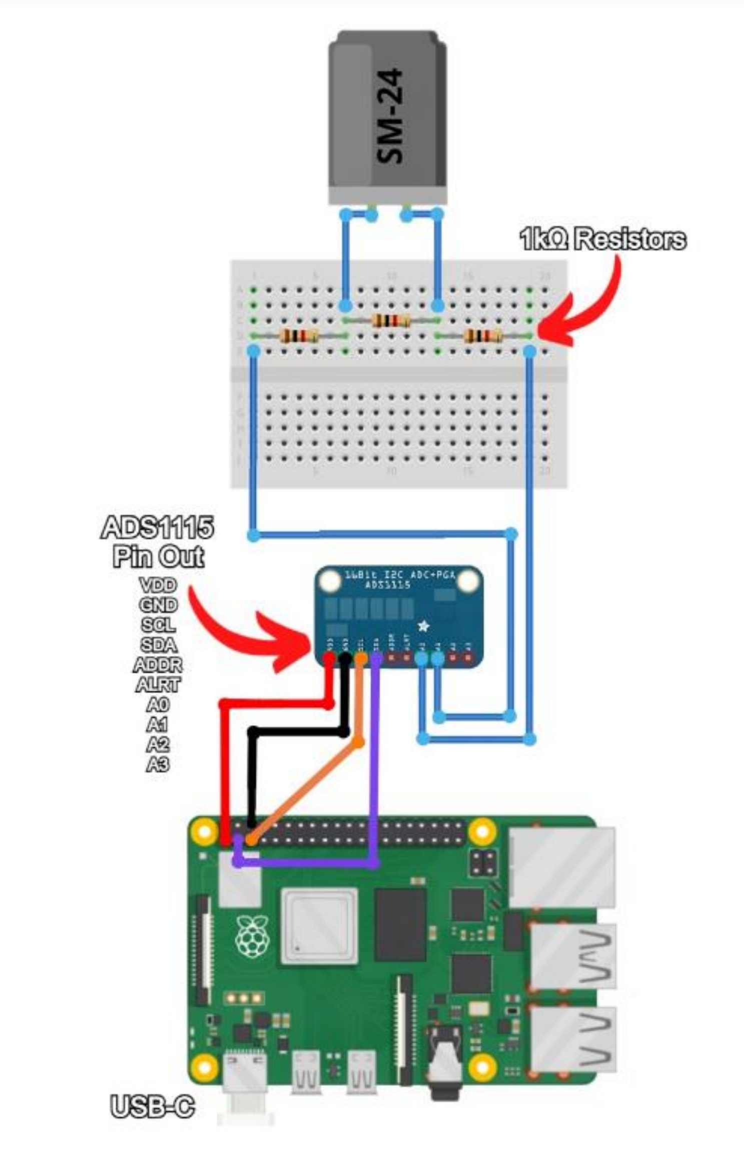
- Screenshot



- Raspberry Pi Setup

SM-24 Geophone





Credit for diagram and circuit to Core Electronics: https://core-electronics.com.au/guides/geophone-raspberry-pi/

LIDAR and ROS Integration for Search and Rescue RC Car

Credit for the code : [Joshnewans, “GitHub - joshnewans/articubot_one,” GitHub. https://github.com/joshnewans/articubot_one/](https://github.com/joshnewans/articubot_one/)

- **ROS 2 Setup:**

Install ROS 2 Humble and essential packages (`rplidar_ros`, `slam_toolbox`).

Resolve any initial build issues to ensure system readiness.

- **LIDAR Integration:**

Connect and configure RPLIDAR hardware.

Adjust serial port settings and launch ROS nodes for data transmission.

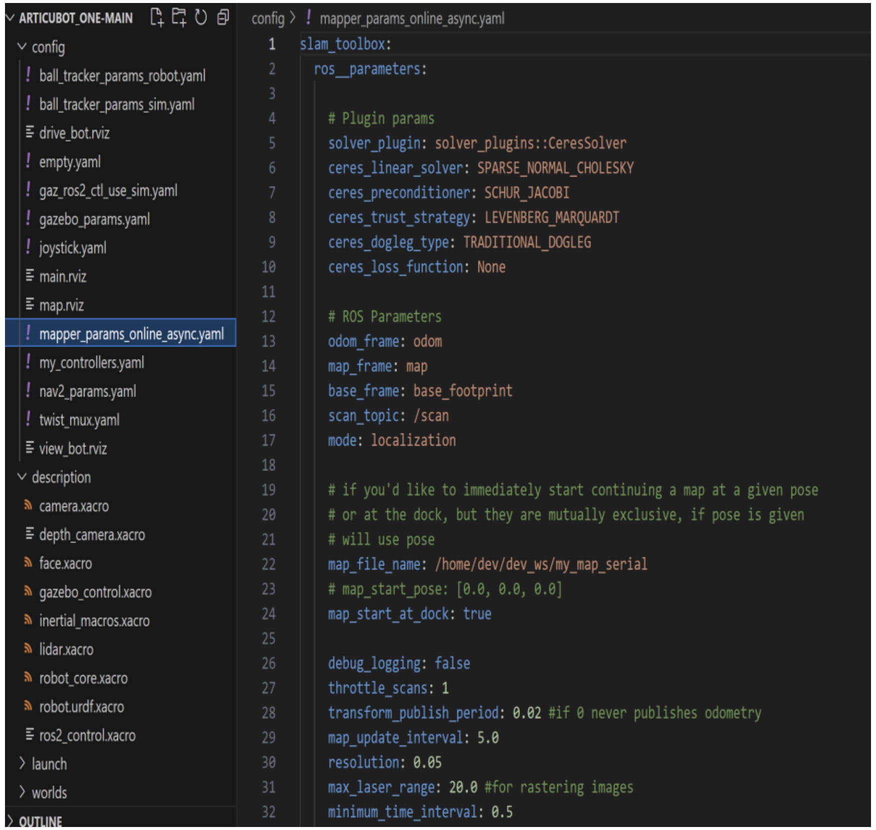
- **LIDAR Functionality:**

Use LIDAR for real-time mapping and dynamic obstacle avoidance with ROS tools.

- **UI Controller:**

Develop a remote UI to monitor LIDAR maps and manually control the vehicle.

Integrate features for obstacle tracking and operator intervention.




```yaml
slam_toolbox:
  ros__parameters:

    # Plugin params
    solver_plugin: solver_plugins::CeresSolver
    ceres_linear_solver: SPARSE_NORMAL_CHOLESKY
    ceres_preconditioner: SCHUR_JACOBI
    ceres_trust_strategy: LEVENBERG_MARQUARDT
    ceres_dogleg_type: TRADITIONAL_DOGLEG
    ceres_loss_function: None

    # ROS Parameters
    odom_frame: odom
    map_frame: map
    base_frame: base_footprint
    scan_topic: /scan
    mode: localization

    # if you'd like to immediately start continuing a map at a given pose
    # or at the dock, but they are mutually exclusive, if pose is given
    # will use pose
    map_file_name: /home/dev/dev_ws/my_map_serial
    # map_start_pose: [0.0, 0.0, 0.0]
    map_start_at_dock: true

    debug_logging: false
    throttle_scans: 1
    transform_publish_period: 0.02 #if 0 never publishes odometry
    map_update_interval: 5.0
    resolution: 0.05
    max_laser_range: 20.0 #for rastering images
    minimum_time_interval: 0.5
```

Slamtec RPLidar

Thanks for Listening!