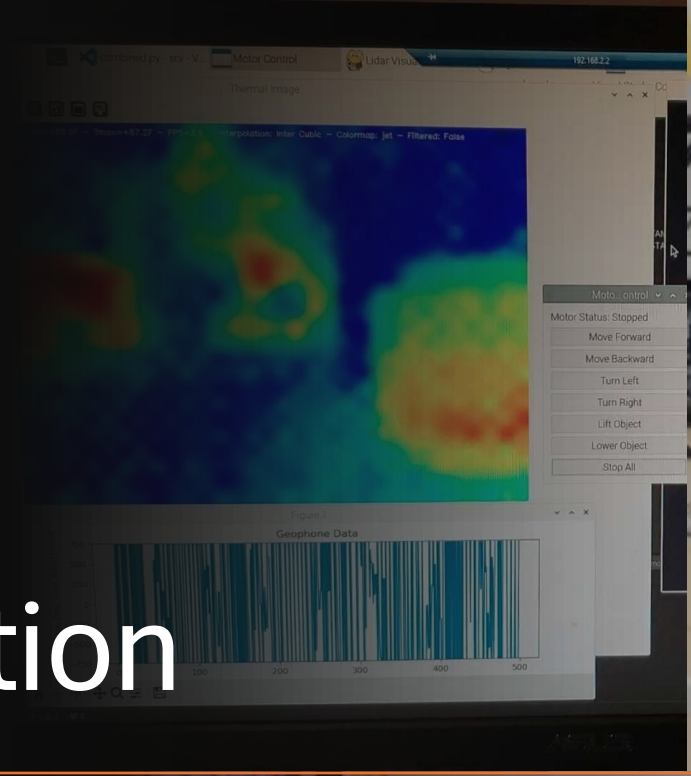
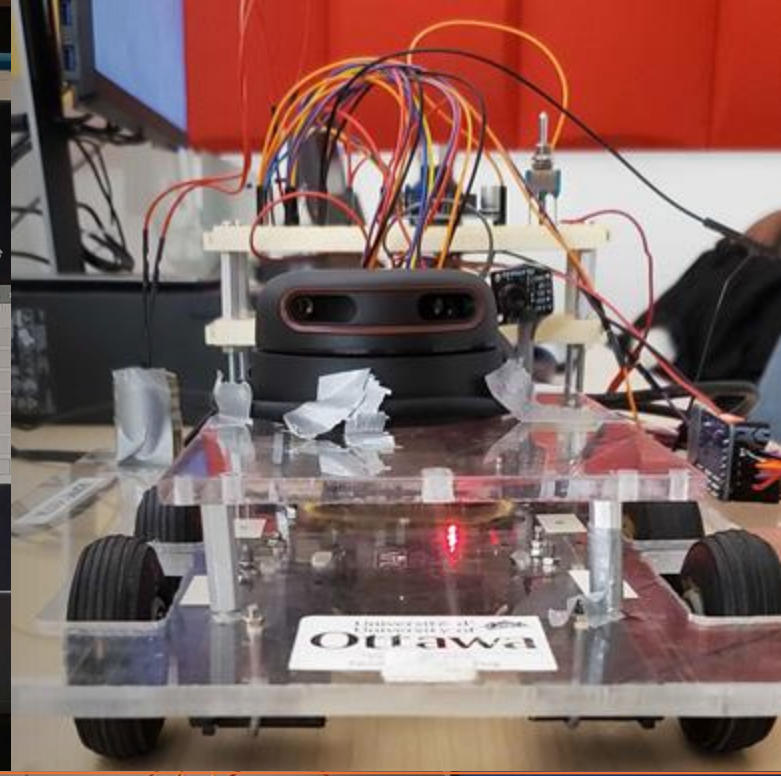
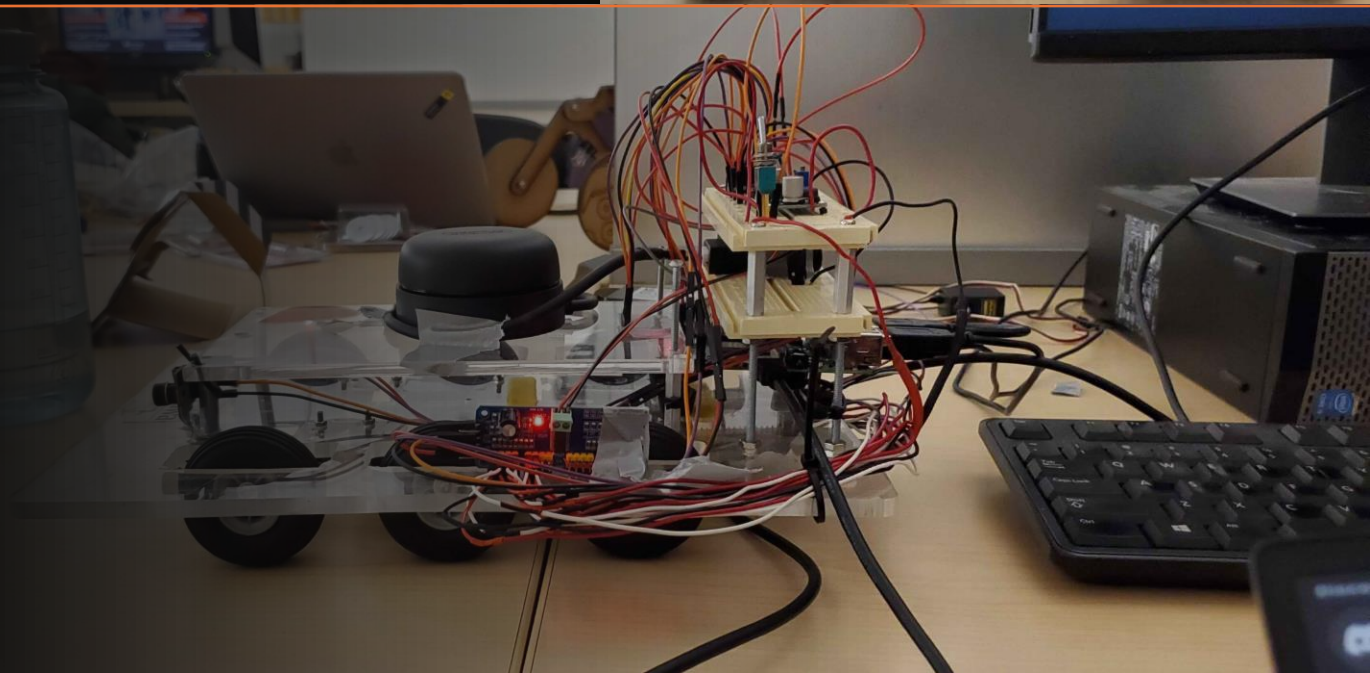
ELG4913 Final Project Presentation







Search and Rescue Remote-Controlled Car with Life Detection

Group 5

- 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 primary challenges for search and rescue teams after an earthquake is that many modern sensors cannot detect victims buried under several meters of heavy concrete.
- **Our Goal:**
- Design an RC car capable of navigating through collapsed buildings to enhance sensor effectiveness.
- Incorporate various sensors to reliably detect 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 na vigate through complex debris.		

Moktar Abdillahi

Customer and how/where is the project used?

The RC vehicle is designed for disaster zones, such as earthquake sites, offering rapid mobility and enhanced search-and-rescue capabilities for efficient 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.

Moktar Abdillahi

Business Case

1 . Problem:

- Search and rescue operations expose first responders to hazardous conditions and are time-consuming.

2. Solution:

- Remote-controlled car equipped with advanced sensors (LiDAR, thermal, and seismic) to navigate disaster zones, map environments, and detect life while keeping responders safe.

3. Market Demand:

- Growing number of natural disasters highlights the need for efficient and safe rescue tools.

Example: During the 2023 Turkey-Syria earthquake, traditional methods faced delays due to hazardous conditions. An RC car could have expedited rescue efforts while minimizing risks.

Papa Kane

Business Case

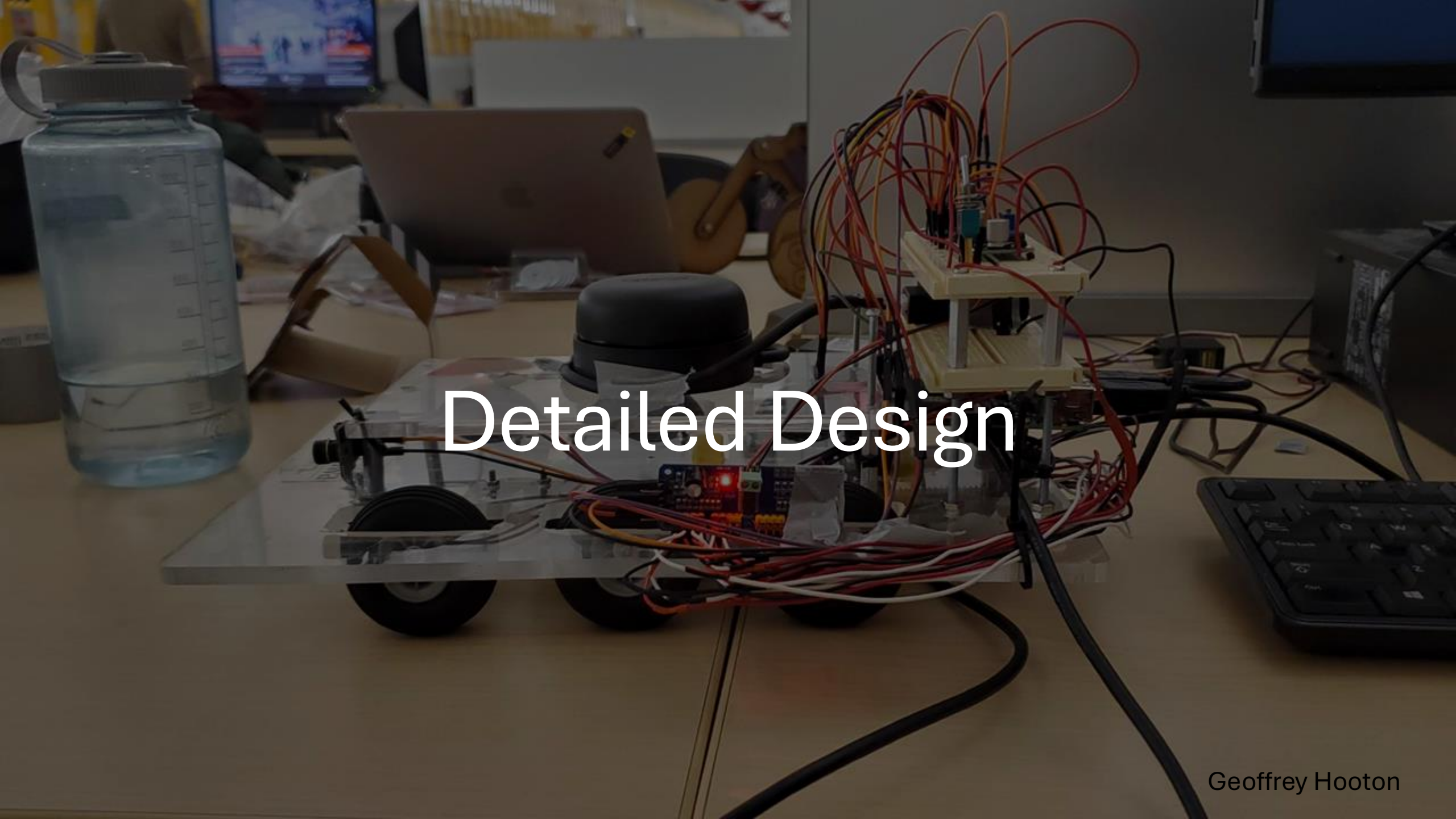
Benefits of the RC Car

- **Financial Viability**:
  - Reduces personnel and mission time by automating search tasks, cutting labor costs by 30-40%.

- **Competitive Advantage**:
  - Combines multiple technologies (LiDAR, thermal, seismic) into one rugged, portable platform for life detection and mapping.

- **Social Impact**:
  - Reduces risks for first responders.
  - Increases the likelihood of saving lives, especially in scenarios like collapsed buildings post-earthquake.
  - Aligns with government and NGO goals for efficient disaster response.

Papa Kane



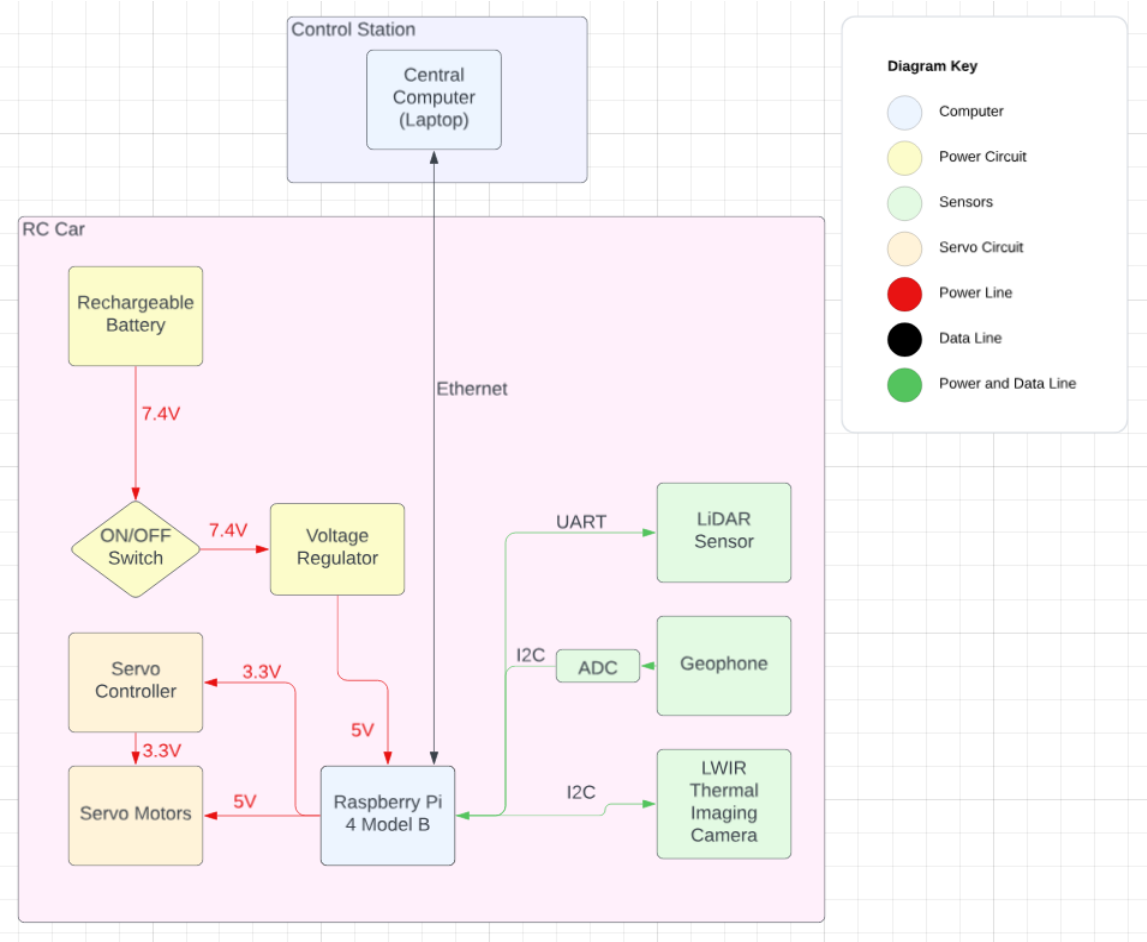
Detailed Design

Geoffrey Hooton

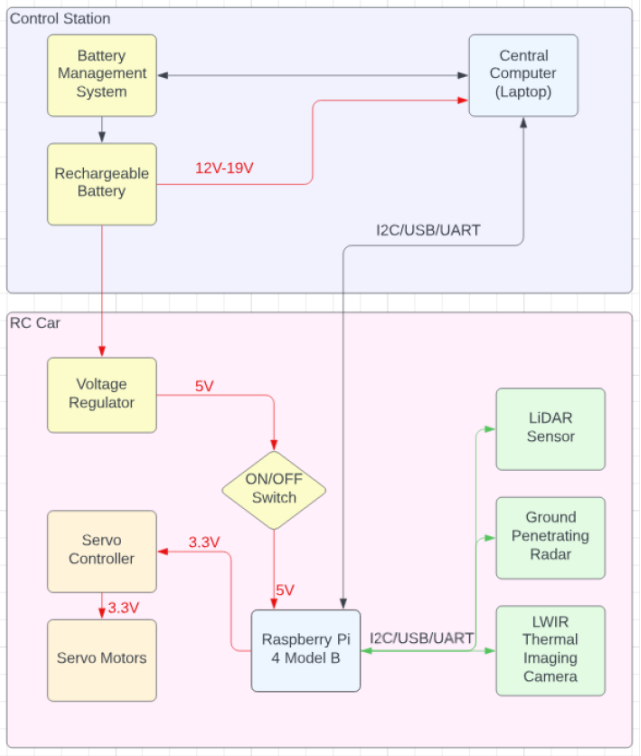
Design Choices

Component Name	Usage
Raspberry Pi	The primary computer for the project.
Slamtec RPLIDAR Sensor	3D mapping of the environment.
MLX90640 Thermal Camera	Detect heat signatures.
Geophone SM-24	Detect underground vibrations.
Servo motors	Control of angular position, velocity, and acceleration.
Servo Motor Driver	Controlling the power supplied to the car's motors from the battery, control speed and direction.
Ethernet Cable	Used to establish a wired connection for reliable long-distance communication.
ADS1115 ADC	Converts analog geophone signal to digital signal for the Raspberry Pi computer
Voltage Regulator	Maintain a consistent 5V supply to the sensors and control systems.
Battery 7.4V	Provide portable power to the car.
Switch on/off	Shut down or activate equipment, helping to prevent accidents, damage to the system, or further escalation of the emergency.

Final Design:



ELG4912 Design:

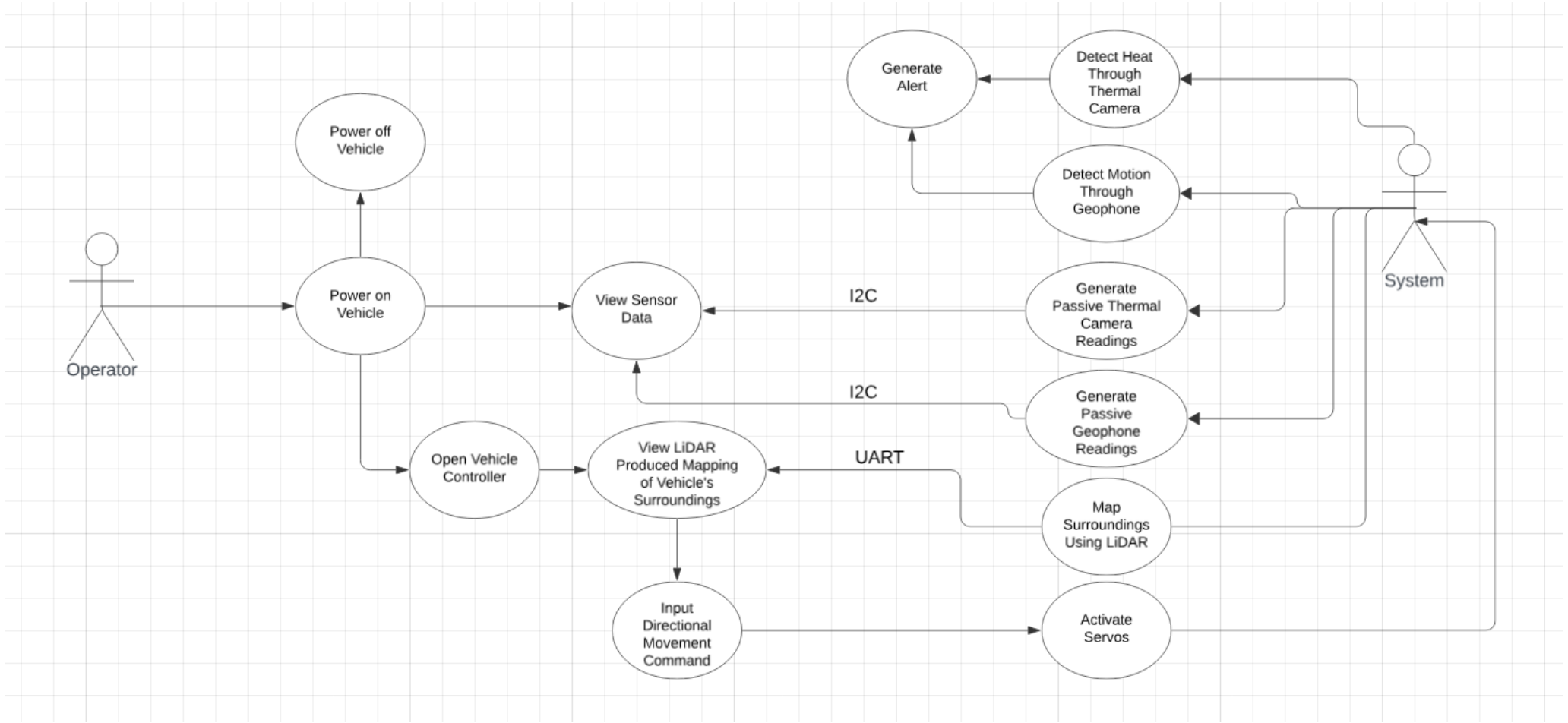


List of Changes:

- Removed battery management system
- Moved rechargeable battery from control station to the vehicle
- Changed ground penetrating radar to geophone (requires ADC).
- Updated serial communication protocols
- More accurately represented power lines
- LiDAR mapping requirement changed from 3D to 2D due to hardware limitations

Final Hardware Design Overview

Geoffrey Hooton



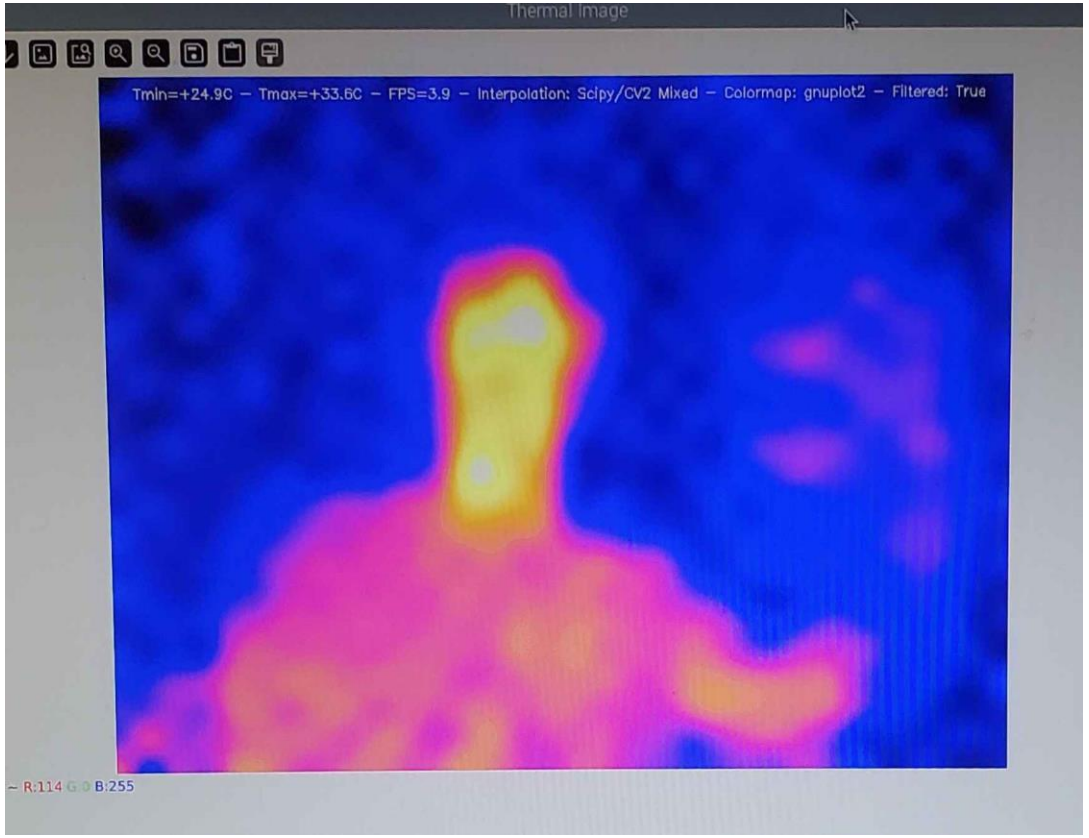
Final Software Design Overview

Use case diagram

Geoffrey Hooton

MLX90640 Thermal Camera

Screenshot

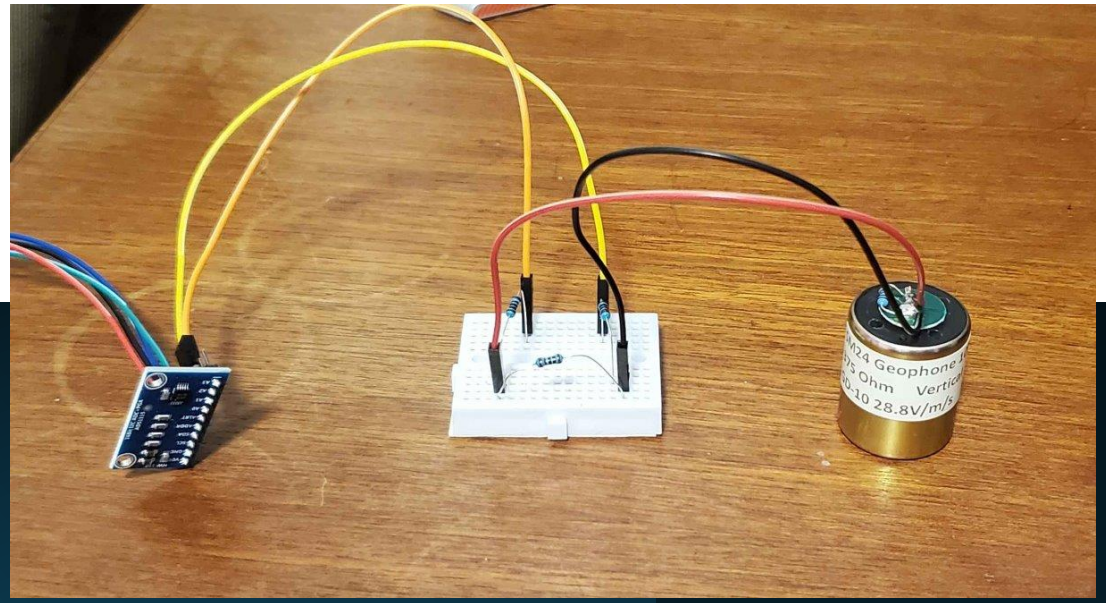


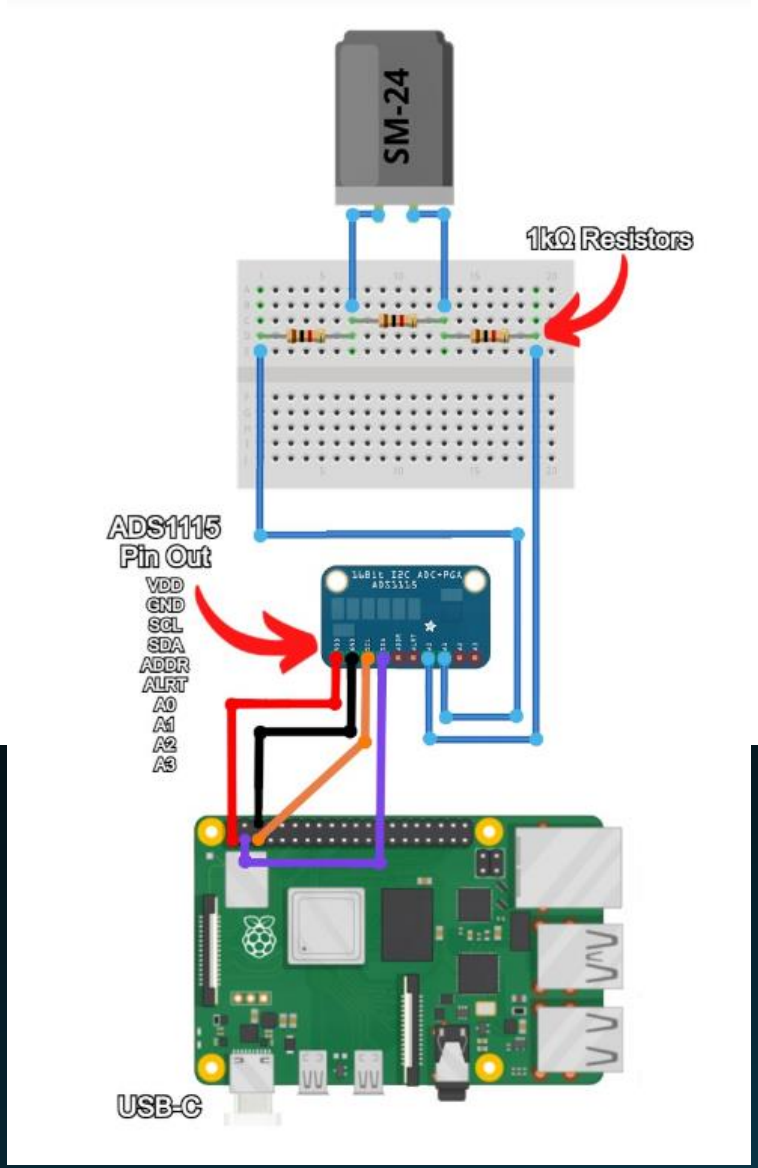
I2C Protocol Camera

Geoffrey Hooton

SM-24 Geophone

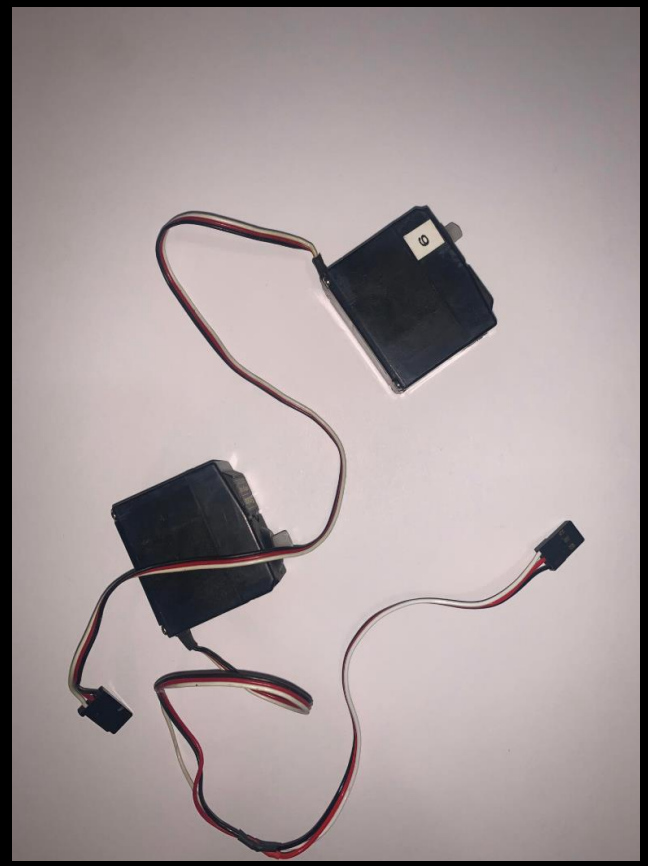
...and ADS1115 ADC





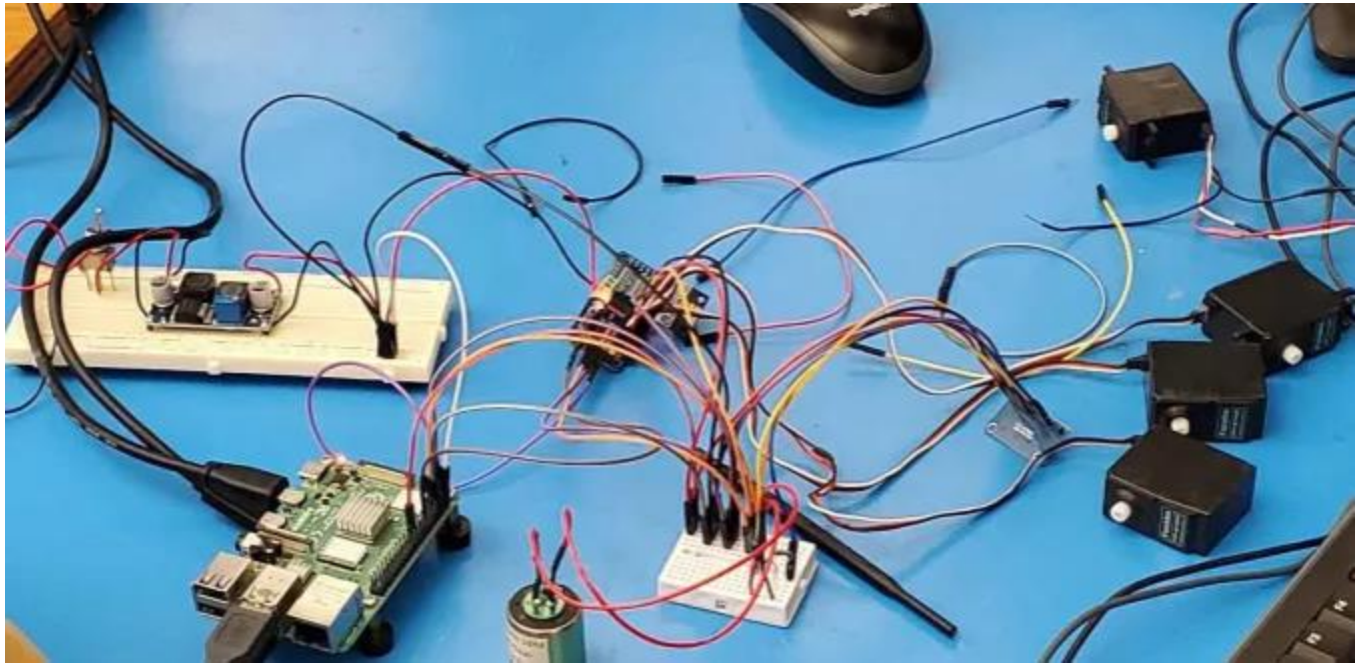
Credit for diagram and circuit to Core Electronics: https://core-electronics.com.au/guides/geophone-raspberry-pi/

Motors



Moktar Abdillahi

Motors control



```python
# Function to move forward
def move_forwardspeed2():
    print("The car is moving forward")
    for servo in [front_left_servo, front_right_servo, middle_left_servo, middle_right_servo, rear_left_servo, rear_right_servo]:
        set_servo_speed(servo, 100)

def move_forwardspeed1():
    print("The car is moving forward")
    for servo in [front_left_servo, front_right_servo, middle_left_servo, middle_right_servo, rear_left_servo, rear_right_servo]:
        set_servo_speed(servo, 50)
# Function to move backward
def move_backwardspeed2():
    print("The car is moving backward")
    for servo in [front_left_servo, front_right_servo, middle_left_servo, middle_right_servo, rear_left_servo, rear_right_servo]:
        set_servo_speed(servo, -100)

def move_backwardspeed1():
    print("The car is moving backward")
    for servo in [front_left_servo, front_right_servo, middle_left_servo, middle_right_servo, rear_left_servo, rear_right_servo]:
        set_servo_speed(servo, -50)

# Function to turn left (pivot)
def turn_left():
    print("The car is turning left (pivot)")
    for servo in [front_left_servo, middle_left_servo, rear_left_servo]:
        set_servo_speed(servo, -50)
    for servo in [front_right_servo, middle_right_servo, rear_right_servo]:
        set_servo_speed(servo, 50)

# Function to turn right (pivot)
def turn_right():
    print("The car is turning right (pivot)")
    for servo in [front_left_servo, middle_left_servo, rear_left_servo]:
        set_servo_speed(servo, 50)
    for servo in [front_right_servo, middle_right_servo, rear_right_servo]:
        set_servo_speed(servo, -50)

# Function for a gentle curve to the left
def gentle_curve_left():
    print("The car is gently curving left")
    set_servo_speed(front_left_servo, 50)
    set_servo_speed(middle_left_servo, 50)
    set_servo_speed(rear_left_servo, 50)
    set_servo_speed(front_right_servo, 100)
    set_servo_speed(middle_right_servo, 100)
    set_servo_speed(rear_right_servo, 100)

# Function for a gentle curve to the right
def gentle_curve_right():
    print("The car is gently curving right")
    set_servo_speed(front_left_servo, 100)
    set_servo_speed(middle_left_servo, 100)
    set_servo_speed(rear_left_servo, 100)
    set_servo_speed(front_right_servo, 50)
    set_servo_speed(middle_right_servo, 50)
    set_servo_speed(rear_right_servo, 50)
```

Moktar Abdillahi

Motors control

Motor Control Code Development:

- Developed Python code to control six servos for the car's movement and one additional servo for lifting.
- Created functions to enable various maneuvers, such as:
  - **Move forward/backward (two speed option)**: Synchronize all servos for linear motion.
  - **Turn left/right**: Adjust servo speed and direction for pivoting.
  - **Gentle curves**: Implement differential speed for smooth directional changes.
  - **Stopping**: Ensure precise and immediate halts.

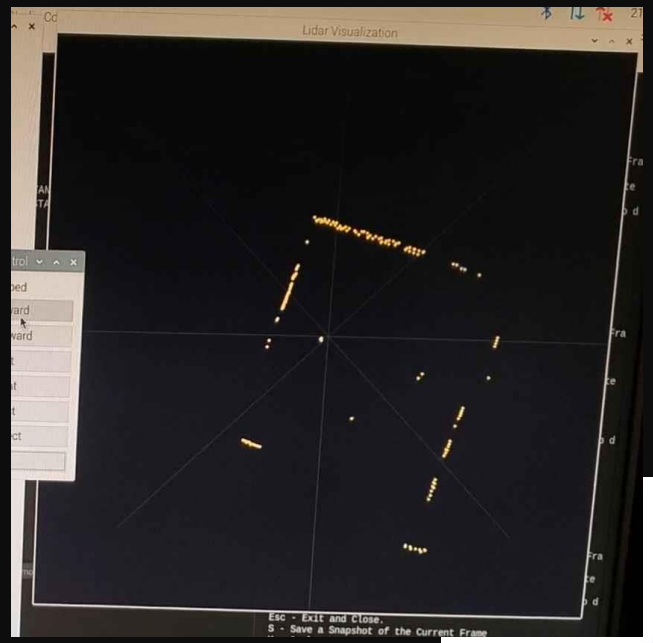
Optimization and Testing:

- Initially tested motor functionality using angular control before transitioning to PWM values for smoother and more efficient continuous rotations.
- Calibrated and refined servo control to achieve accurate trajectories and speed adjustments.

```python
# Function to stop all servos
def stop_all():
    print("The car is stopping")
    for servo in [front_left_servo, front_right_servo, middle_left_servo, middle_right_servo, rear_left_servo, rear_right_servo]:
        set_servo_speed(servo, 0)

# Function to lift the object with the 7th servo
def lift_object():
    print("Lifting the object")
    set_servo_speed(lifter_servo, 100)  # Raise the object

# Function to lower the object with the 7th servo
def lower_object():
    print("Lowering the object")
    set_servo_speed(lifter_servo, -100)  # Lower the object

# Function to stop the lifter servo
def stop_lifter():
    print("Stopping the lifter")
    set_servo_speed(lifter_servo, 0)  # Stop the lifter
```

Slamtec RPLidar

```python
import pygame
import math
from math import floor
from adafruit_rplidar import RPLidar

# Setup the RPLidar
PORT_NAME = "/dev/ttyUSB0"
lidar = RPLidar(None, PORT_NAME, timeout=3)

# Screen settings
WIDTH, HEIGHT = 800, 800
CENTER_X, CENTER_Y = WIDTH // 2, HEIGHT // 2
MAX_DISTANCE = 4000  # Set according to your lidar's ma

# Initialize pygame
pygame.init()
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Lidar Visualization")
clock = pygame.time.Clock()

# Colors
BACKGROUND_COLOR = (20, 20, 30)
POINT_COLOR = (255, 215, 0)  # Gold color for latest fr
GRID_COLOR = (50, 50, 70)
CIRCLE_COLOR = (100, 100, 150)

# Initialize data storage
scan_history = []  # Store recent scan frames

# Function to map lidar data to screen coordinates
def polar_to_cartesian(angle, distance):
    if distance > MAX_DISTANCE:
        distance = MAX_DISTANCE
    angle_rad = math.radians(angle)
    x = CENTER_X + int(distance * math.cos(angle_rad) *
    y = CENTER_Y + int(distance * math.sin(angle_rad) *
    return x, y
```



Papa Kane

Prototype

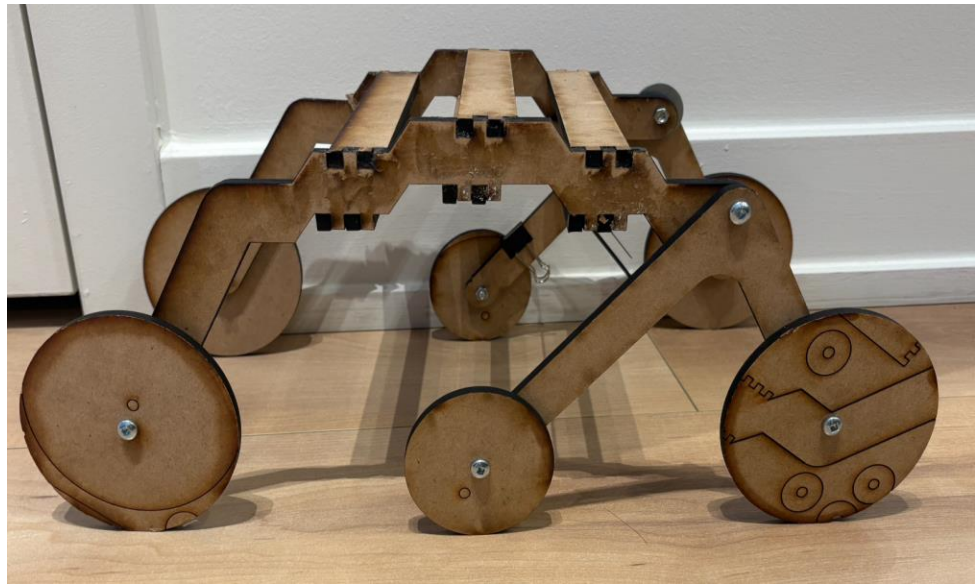
•**Sturdy Chassis**: Designed for durability and to withstand harsh environments like disaster sites.

•**Traction Wheels**: Provides stable navigation on uneven terrain and rubble.

•**Modular Base**: Allows for easy integration of sensors like LiDAR, thermal cameras, and geophones for life detection.

•**Compact Design**: Optimized to fit through narrow spaces and navigate tight environments.

•**Lightweight Material**: Ensures portability for easy deployment in emergency scenarios.



Papa Kane

User Interface Development

Geophone Data Visualization UI Code: Utilizes Matplotlib and PyQt5 to animate real-time data from a geophone sensor, enhancing the visualization of seismic data.

Lidar UI Code: Provides a graphical user interface using PyQt5 for controlling LIDAR devices, enabling operations such as turning the LIDAR on/off, measuring distances, and adjusting scanning speeds.

Motor Control Buttons Code: Outlines the interface for motor operations, facilitating directional movements and adjustments with simple button controls.

```python
# Define motor control UI class
class MotorControlUI(QWidget):
    def __init__(self):
        super().__init__()

        self.setWindowTitle('Motor Control')

        layout = QVBoxLayout()

        self.status_label = QLabel("Motor Status: Stopped")
        layout.addWidget(self.status_label)

        self.forward_button = QPushButton("Move Forward")
        self.forward_button.clicked.connect(self.move_forward)
        layout.addWidget(self.forward_button)

        self.backward_button = QPushButton("Move Backward")
        self.backward_button.clicked.connect(self.move_backward)
        layout.addWidget(self.backward_button)

        self.left_button = QPushButton("Turn Left")
        self.left_button.clicked.connect(self.turn_left)
        layout.addWidget(self.left_button)

        self.right_button = QPushButton("Turn Right")
        self.right_button.clicked.connect(self.turn_right)
        layout.addWidget(self.right_button)

        self.lift_button = QPushButton("Lift Object")
        self.lift_button.clicked.connect(self.lift_object)
        layout.addWidget(self.lift_button)

        self.lower_button = QPushButton("Lower Object")
        self.lower_button.clicked.connect(self.lower_object)
        layout.addWidget(self.lower_button)
```

Motor Control UI Code

```python
import matplotlib
matplotlib.use('QtAgg')  # Ensure the QtAgg backend is set before importing pyplot

import sys
import time
import Adafruit_ADS1x15
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from multiprocessing import Process, Queue
from PyQt5.QtWidgets import QApplication, QMainWindow, QVBoxLayout, QWidget
from PyQt5.QtCore import QTimer

# Create an ADS1115 ADC (16-bit) instance
adc = Adafruit_ADS1x15.ADS1115(address=0x48, busnum=1)

# Choose a gain for reading voltages (16 = +/-0.256V)
GAIN = 16

# Graph settings
x_len = 500  # Number of points to display
y_range = [-750, 750]  # Y-axis range

# Sampling rate control (approximately 25 samples per second)
sampling_interval = 0.04  # In seconds (25Hz)

# Function to generate data
def generate_data(queue):
    """Simulate data collection from the ADC."""
    while True:
        value = adc.read_adc_difference(0, gain=GAIN)
        queue.put(value)
        time.sleep(sampling_interval)  # Control the sampling rate to 25Hz
```

Geophone UI Code

```python
# Main UI for Lidar control
class LidarUI(QWidget):
    def __init__(self):
        super().__init__()

        self.setWindowTitle('LIDAR Control')

        layout = QVBoxLayout()

        # LIDAR ON/OFF Button
        self.toggle_button = QPushButton("Turn LIDAR ON")
        self.toggle_button.setCheckable(True)
        self.toggle_button.clicked.connect(self.toggle_lidar)
        layout.addWidget(self.toggle_button)

        # Measure LIDAR distance button
        self.measure_button = QPushButton("Measure LIDAR Distance")
        self.measure_button.clicked.connect(self.measure_lidar)
        layout.addWidget(self.measure_button)

        # Scan Status Label
        self.status_label = QLabel("Idle")
        layout.addWidget(self.status_label)

        # Zoom Controls (Slider)
        self.zoom_slider = QSlider(Qt.Horizontal)
        self.zoom_slider.setRange(1, 5)
        self.zoom_slider.setValue(1)
        self.zoom_slider.valueChanged.connect(self.update_zoom)
        layout.addWidget(QLabel("Zoom Level"))
        layout.addWidget(self.zoom_slider)

        # Speed Control for Scanning (Slider)
        self.speed_slider = QSlider(Qt.Horizontal)
        self.speed_slider.setRange(1, 60)
        self.speed_slider.setValue(scan_speed)
        self.speed_slider.valueChanged.connect(self.update_speed)
        layout.addWidget(QLabel("Scan Speed (FPS)"))
```

Lidar UI Code

Fatmah Bayrli

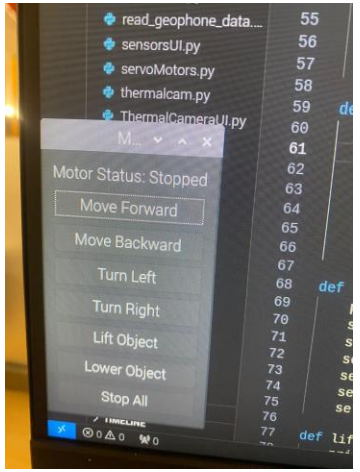
Integrated Control Interface for Sensors and Motor Operations

UI Visualization: Displays status and controls for sensors and motors, enabling real-time interaction and data visualization.

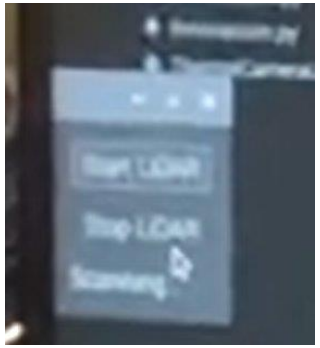
Motor Control: Provides a UI for precise motor operations, including direction and object manipulation.

Lidar Control: Features a dedicated button for starting and stopping LIDAR scans, essential for effective data management.

UI Visualization screen of all sensors with the motors



Motor Control Button

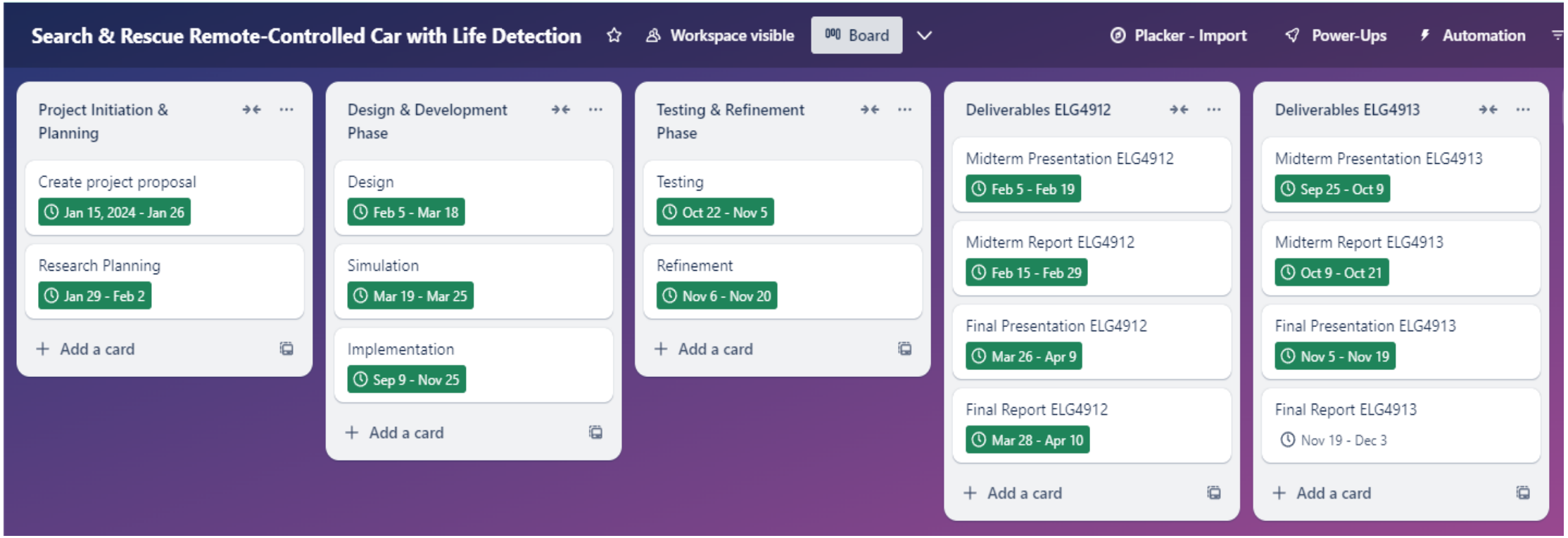


Start-Stop and Scanning for the Lidar

Fatmah Bayrli

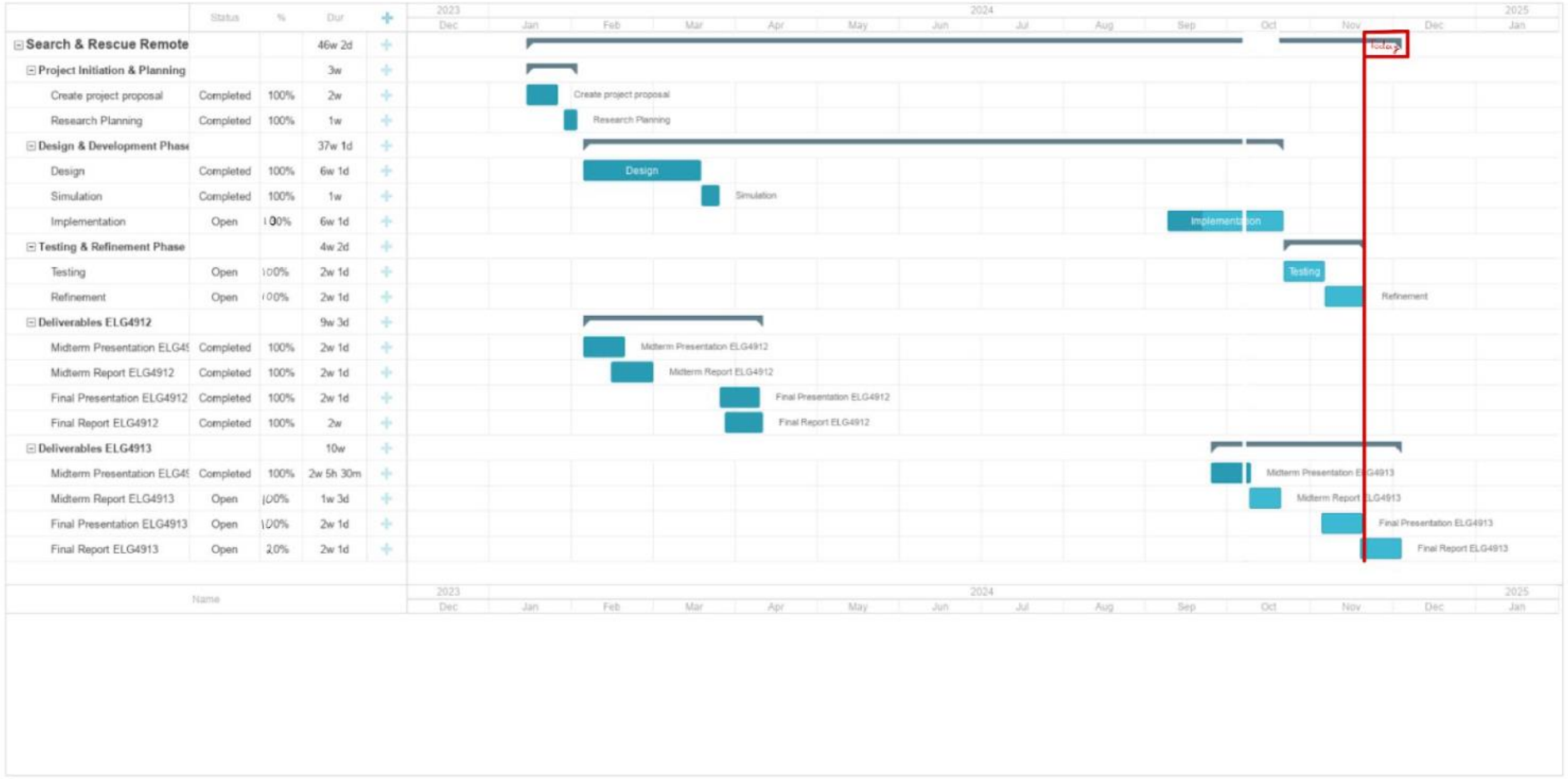
Schedule & Budget Outlook

Schedule



Julien Kapro

Gantt Chart



Exported from Placker.com on Nov 25

Julien Kapro

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)

Julien Kapro

MLX90640 Thermal Camera

```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

Julien Kapro

SM24 Geophone

```python
#Geophone DAS
import time
import Adafruit_ADS1x15
import numpy as np
from scipy.signal import find_peaks
from scipy.fft import fft, fftfreq
import csv

#Initializing the ADC (ADS1115)
adc = Adafruit_ADS1x15.ADS1115(address=0x48, busnum=1)
GAIN = 16

#-----------------------------------------------------
#-----------------------------------------------------

#Function to collect the data
def data_collect():

    data =[]
    start_time = time.time()

    #Sampling 20 seconds of data at a time
    #The time different between each data point is 0.01s
    #This way, we know that each point is at x0=0, x1=0.01, x2 = 0.02...
    while (time.time() - start_time) < 20:
        data.append(adc.read_adc_difference(0, gain=GAIN))
        time.sleep(0.02)

    return data

#-----------------------------------------------------
#-----------------------------------------------------

#Function is to detect a heartbeat in the dataset acquired from data_collect()
def heart_detect(data):

    #This removes the DC component of the data
    data = np.array(data) - np.mean(data)

```

Date	Time	Message	Frequency (Hz)
2024-11-18	15:29:51	Nothing Detected	0
2024-11-18	15:30:51	Potential Heartbeat Detected	2.51
2024-11-18	15:31:31	Potential Heartbeat Detected	2.46

Julien Kapro

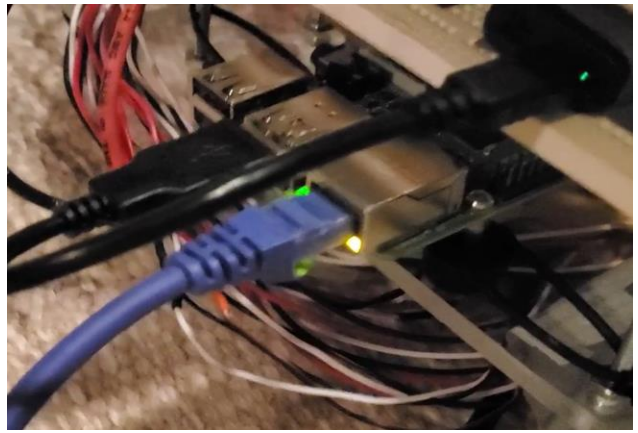
Risk Management and Safety

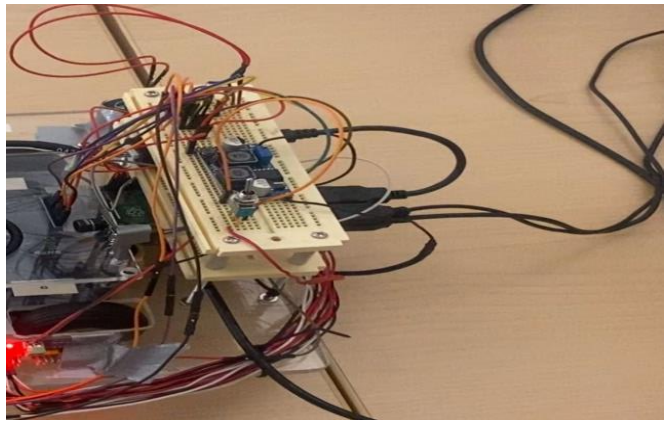
Category	Risk	Probability	Impact	Mitigations	Residual Risk Level
Electrical and Network Safety	Ethernet Usage IP configuration	Medium	Medium	Use reliable ethernet cable and protocol for testing.	Low
Battery and Electrical Safety	Battery depletion, Shock risk, Short circuit	Medium	High	Ensure proper Battery Management, insulation of connections, organize wiring to prevent short circuit	Medium
Mechanical Safety	Structural Integrity	Medium	High	Perform regular check for the chassis and the components	Medium
Sensors Safety	Lidar Sensor	Medium	High	Implement Stop Button for Lidar	Low
Motor Control Safety	Unexpected Motor Operations	High	High	Implement "Stop all" button to quickly Shut down in case of malfunction	Low
Testing and Maintenance	Regular Check	Medium	Medium	Conduct regular functionality and safety components check	Low

Fatmah Bayrli

Test Plan

Test Category	Test Description	Test Status
Hardware Testing	Individual and Integrated Components Tests	Passed
Software Testing	Integration of Software and Hardware	Passed
Communication Testing	Communication Testing	Passed
Ethernet Testing	Perform with and without Ethernet connectivity	Passed

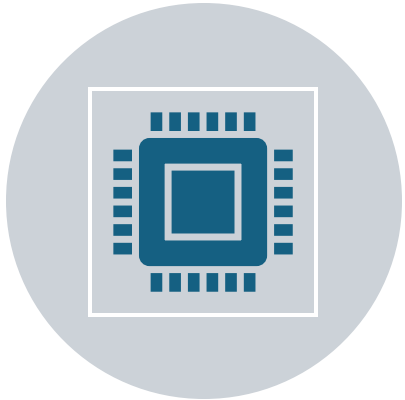




ROS Integration Research

RESEARCH : RESEARCHED AND IDENTIFIED THE NECESSARY LIBRARIES (LIDAR)

ACCOMPLISHMENT : SUCCESSFULLY INSTALLED AND CONFIGURED ROS ON OUR SYSTEM



CHALLENGES: ENCOUNTER DIFFICULTIES IN SETTING UP THE LIDAR REQUIRED LIBRARIES. FACED COMPATIBILITY PROBLEM WITH ROS PACKAGES WITH THE LIDAR.

Fatmah Bayrli

Future Enhancement

1.UI Enhancement :transition from Qt-based GUI to React

2. UI Integration with ROS :Integration of the UI with ROS

3. Chassis upgrade: use more rebuts chassis Design

4. System Scalability: use Temperature and Humidity Sensors, Gas Sensors, Pressure Sensors, Radio Frequency (RF) Sensors, and Vibration Sensors

5. Machine Learning and AI integrations Handle complex navigation scenario

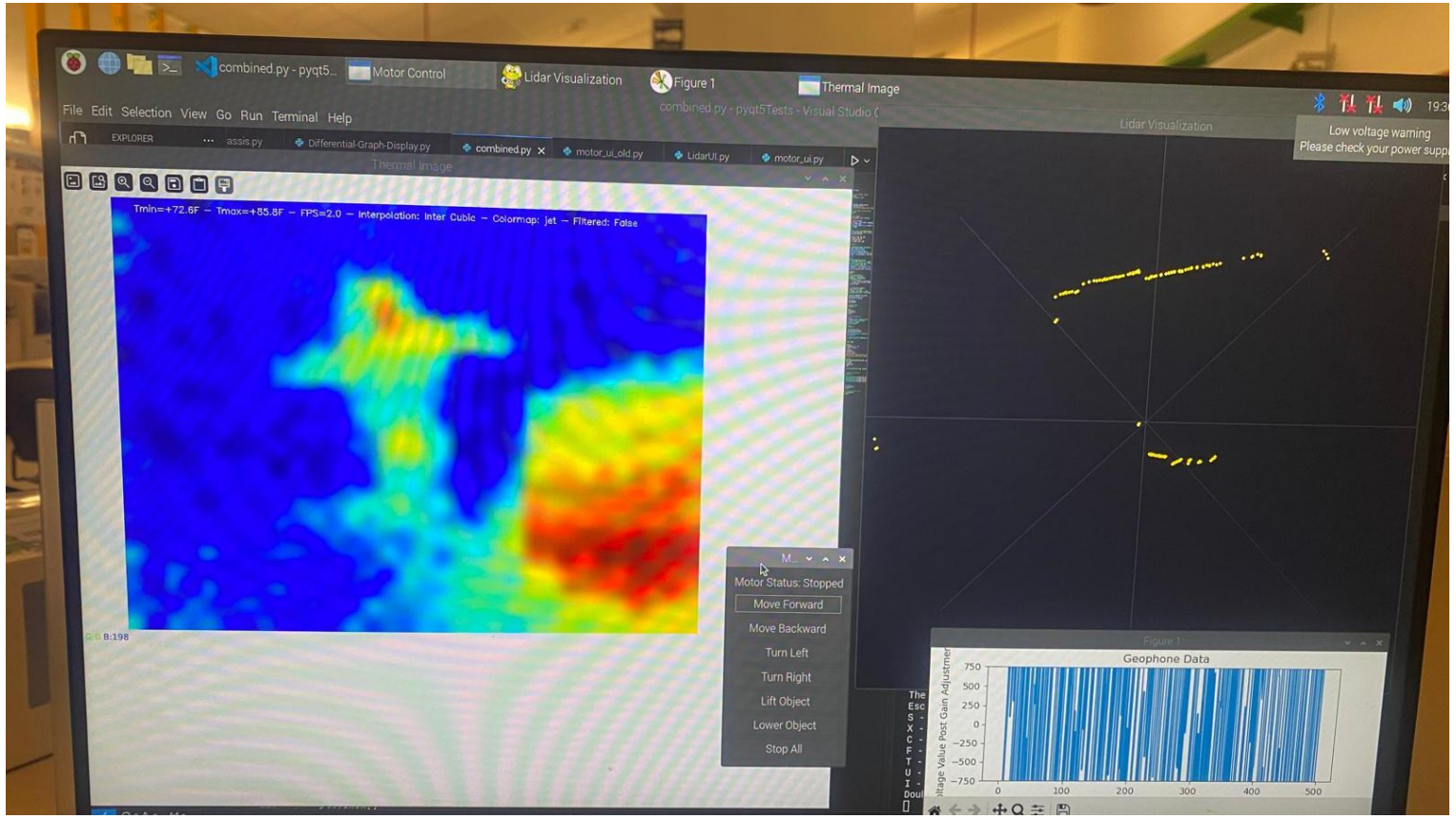
Fatmah Bayrli

Major Contributions

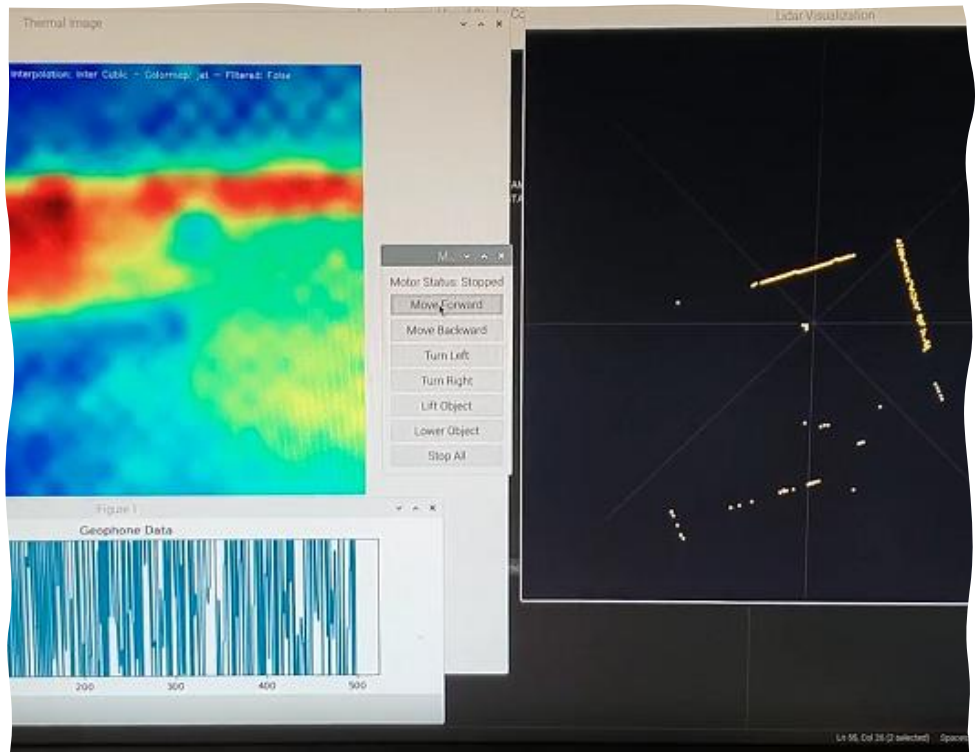
Team Member	Major Contributions
Fatmah(FB)	• UI implementation Development • Integrated Control Interface for Sensors and Motor Operations • Risk management and safety • Test plan • ROS2 integration research • Future Enhancement • Demos
Walid	• Chassis design ○ Suspension design • UI research • Requirements specification
Moktar	• Battery implementation • Motor implementation • Construction of power circuit • Project overview (objective, initial requirements) • Target customer and use cases
Papa	• Chassis Implementation (prototype manufacturing/laser cutting and assembling) • Business case
Julien	• Data acquisition and analysis system ○ Storing of data in CSV files • Budgeting and schedule
Geoffrey	• Geophone/LiDAR/Thermal camera implementation (circuits, soldering, initial code) • Remote desktop and IP configuration • Detailed design • Demos

Video Demos

Demo Without Using Ethernet Cable



Fatmah Bayrli


Thermal Image
Lidar Visualization
Motor Status: Stopped
Move Forward
Move Backward
Turn Left
Turn Right
Lift Object
Lower Object
Stop All
Figure 1
Geophone Data
200
300
400
500

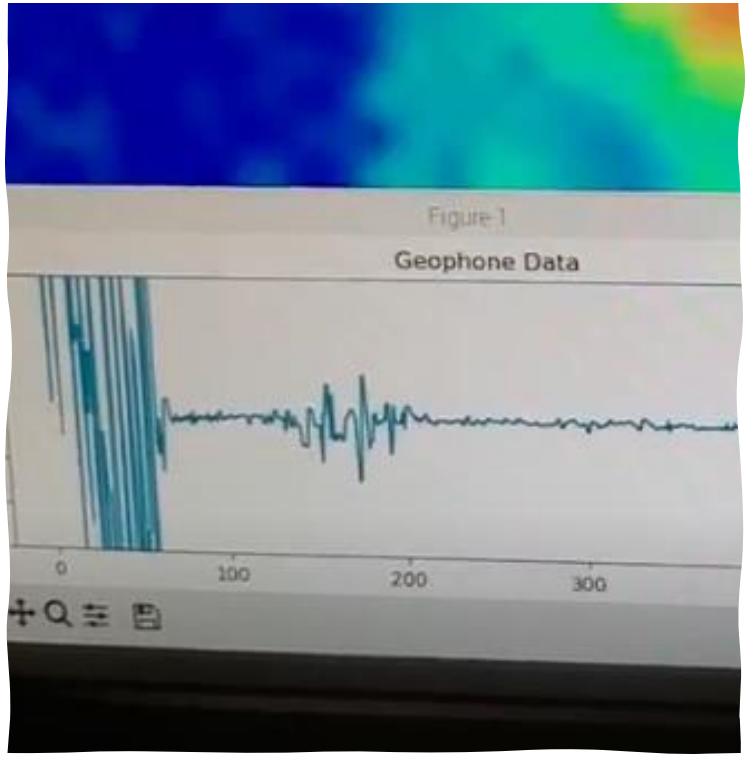


Figure 1
Geophone Data
0
100
200
300

Demos

Thanks for Listening

Questions?