# Project Plan

## Introduction

The investigation seeks to revolutionise robotic navigation by integrating Large Language Models (LLMs) into autonomous vehicle systems. Existing leaders in the space apply neural networks and reinforcement learning algorithms; however, the untapped potential of LLMs proposes a novel trajectory. The project aims to transcend these conventional modalities, fostering advanced navigation capabilities, especially in handling edge cases.

## Problem Formulation

Conventional autonomous systems often stumble upon unpredictability; their performance in edge-case scenarios remains inadequate. Command systems exhibit a deficiency in the nuanced understanding that could be provided by natural language processing. The integration of instructive language into vehicle commands proposes a more intuitive machine-user interaction and an enriched comprehension in machine directive processes.

## Methodology

The methodological framework synthesizes principles from backend engineering, prioritizing David Shapiro's ACE Framework principles in the microservice architecture. The approach will:

1. Transform sensor data into textual prompts through vectorised object representation.
2. Implement asynchronous communication via an event stream to maintain a non-blocking, reactive system.
3. Use Prometheus for real-time metric tracking, crucial for addressing LLM latency.
4. Choose Rust for system development to exploit its speed and reliability, leveraging Python's flexibility where necessary.
5. Investigate the pretraining and fine-tuning of LLMs to enhance responsiveness and accuracy or use Auto Chain of Thought with Zero Shot learning
6. Initiate with LLaMA 7B models, considering its optimised balance of size and performance.
7. Apply prompt engineering to generate concise, context-rich directives for the LLMs.

The project is designed to be iterative, with refinements based on continuous prompt optimization and feedback from the system.

## Results Expected

The project's ambition is outlined in successive complexity stages:

* Level 1: Basic command recognition and execution, guiding through simple terrains via voice instruction.
* Level 2: Adherence to traffic regulations through sensor-detected signs and urban environmental integration.
* Level 3: Proficient obstacle detection and avoidance, replicating dense urban scenarios with active pedestrian and vehicle movements.

Each stage serves as a foundational base for subsequent complexity.

## Objectives and Anticipated Outcomes

Key milestones include:

1. Benchmarking against extant navigation systems to establish a baseline.
2. Developing a seamless interface for sensor-to-LLM communication.
3. Enhancing decision-making in autonomous vehicles using LLMs.
4. Achieving minimal response time from sensory data acquisition to vehicular action.
5. Testing and validating the model in complex simulated environments.

## Anticipated Challenges and Solutions

The project anticipates two main challenges:

1. Latency: The inherent processing time for LLMs is significant. The adoption of Rust and an efficiency-driven system architecture is a strategic priority.
2. Data Interpretation: Accurate translation of sensor data to textual information is crucial. Through meticulous prompt engineering and potential model pretraining, the project aims to refine this translation process.

## Validation and Future Work

The proposed approach will undergo validation against the outlined objectives and through comparative analysis with established systems. Success could spearhead further exploration into LLM applications within broader contexts such as the Internet of Things and smart city infrastructures.

## Conclusion

The research is poised to lead in the evolution of autonomous vehicle technologies. By merging the cognitive capacities of LLMs with the precision of cutting-edge sensory apparatus, it aspires to establish a new standard in robotic navigation.

# Project Plan Draft

Plan (2 page):

* Introduction
* Problem formulation
  + What am I going to solve?
  + Mitigating Plan
* Methodology
* Results expected - objective
* By Friday

Introduction:

* The project is on Robot navigation performance enhanced using LLMs
* Focusing on autonomous vehicle navigation
* There are many different autonomous vehicle companies but they are built on neural networks and reinforcement learning algorithms.
* Examples are Tesla and Luminar are brilliant examples

Problem Formulation:

* These cars are not sophisticated enough to be able to tackle edge cases
* They also do not allow deep communication between human and car, it is not in natural language
* Imagine being able to give your car voice commands and it can detect what to do and how to do it
* The car should also be able to avoid dangerous obstacles

Methodology:

* Bringing together concepts from backend engineering architectures such as microservices and concepts such as the ACE Framework from researcher David Shapiro
* Translate input from sensors (LiDAR and Camera) to vectorised object representation (concept from Driving with LLMs)
* This is passed in as natural language to a microservice that can then publish necessary data to decision making LLMs
* Communication will be asynchronous on an event stream to allow for constant processing
* Metrics will be collected using Prometheus as one of the issues that will be faced will be latency, LLMs take time to output answers
* The microservices will mainly be built with Rust but it is possible that some will be built with python. Rust will be used preferably for robustness, memory safety, static typing system and speed. Reducing latency in the system will be very important as LLMs are known to take time before giving an output.
* Prompt Engineering will be used to provide context to the LLMs and assist in getting the desired output
* Will look into the possibility of pretraining and finetuning the models
* The models used will be the LLaMA 7B
* It will be an iterative process as most of the changes will come from the prompts once the rest of the system is working properly.

Results Expected:

* There will be different levels of achievement, each stage will provide a more difficult objective while also incorporating the objectives from before. This increases the scope of the project if it ends up being easier than it seems but also provides a catching net if the project is way more intense that originally predicted.
* Level 1:
  + The car can take language instructions in and adjust its plotted path to fulfil the request:
    - Example: Go to McDonalds, but also stop at the filling station to get some gas on the way there.
  + Terrain will be mostly empty except for that single car
* Level 2:
  + The car can use sensors to detect road signs and adjust accordingly:
    - Example: Sign saying 40 is the max speed causes the car to slow down if over that speed
  + The LLMs will be finetuned with traffic rules and the context of the prompt will specifically ask it to follow the rules.
  + The Terrain will be populated with signs and traffic signals that are typically found in an urban environment
* Level 3:
  + The car will include LiDAR and Camera sensors that will pick up objects/people to avoid and allow the car to make necessary path/speed adjustments to make sure it doesn’t crash or have an accident
  + The terrain will be populated with cars and people

Steps to take (for Gantt Chart):

* Project Scope and Planning Document: Due Date 06/12/23
  + Project Scope
    - Write the draft – this is complete
  + Literature Review
    - Finish Reading 20 papers
    - Each paper should have a detail of the abstract, the key points and things I may apply to my own project
  + Project work plan and ethical considerations
    - Write a general draft for the plan – completed
    - Complete a Gantt Chart with tasks – In Progress
    - Create milestones and written description of tasks
  + Continuous compilation of references
* Rust LLM Microservice Template: Due Date 06/12/23
  + Learn Rust
  + Write a microservice in Rust with Axum
  + Implement a Kafka Event Bus with Microservices in Rust
  + Combine Rust with an LLM in the microservice
* Simulated environment: Due Date 21/12/23
  + Try out one or two simulation frameworks and decide one
  + Build a car and add sensors (camera and LiDAR)
  + Figure out a way to transfer data from sensors to microservices
  + Figure out a way to control the car using microservices and LLM
  + Add necessary environments (landscapes) associated with each objective stage
* LLM Finetuning, Prompting and Architecture: This should be where I spend most of my time in 2024
  + Continuously list ideas that are relevant
  + Formulate a combination/implementation plan for prompting and finetuning
  + Create an architecture plan for each level with suggestions for variations to offset the idea
  + Test out prompts and contexts, record their responses when passed to an LLM and pick out the best for the project
  + Continue to add tasks here as knowledge in this area broadens