# Maze navigating car

Contents

[Maze navigating car 1](#_Toc116418761)

[Analysis and research 1](#_Toc116418762)

[Initial Problem identification 1](#_Toc116418763)

[Identification of stakeholders 1](#_Toc116418764)

[Research of similar systems 1](#_Toc116418765)

[Design 1](#_Toc116418766)

[Development 1](#_Toc116418767)

[Evaluation 1](#_Toc116418768)

## Analysis and research

### Initial Problem identification

Robots that can navigate small or restricted areas are invaluable in modern day engineering. They allow project managers and developers to fully map out an area, potentially for development or removal of rubble. Other applications of self-learning algorithms apply to many other sectors of the world, such as car development with calculating the path of least resistance for airflow or calculating the shortest distance between stations in a city. The nature of self-learning algorithms means they can adapt and evolve in a large array of ideas.

To continue this trend, I will hopefully be exploring a small portion of this in self driving cars. I will specifically be using a car to navigate a maze using several sensors to simulate the ability for cars to detect a path to follow. I will be exploring this by creating a small vehicle and attaching either a distance sensor or a colours sensor to follow a specific path, further expanding the path and even adding different routes.

### Identification of stakeholders

Myself: Since I am in charge of the development of system, any hold ups or push backs directly affect me.

Engineer:

Apprentice:

### Research of similar systems

## Design

### Main Car Menu

For this project one of the success criteria is to have a functional menu that can be used to interact with the basic abilities of the car. There are different ways to design and produce a menu, each of which have their drawbacks and benefits.

|  |  |  |
| --- | --- | --- |
| Design | Benefits | Drawbacks |
| All of the elements on one page | * Simple to program, will save a lot of time * Can be easily changed if necessary | * Cluttered and hard to read as a user * Difficult to use when testing iteratively as navigation won’t be easy |
| “Drop down” style menu | * Can break down and chunk out the different aspects of the car * Easier to navigate as each command has its respective location | * Potentially difficult to program * Difficult to change individual aspects of the menu |
| Speech driven Menu | * Has much more to offer in terms of versatility. Just add a voice line and the action. * There is already code to identify speech patterns | * Difficult to integrate whilst being time efficient. * Voice commands will need to be exact, otherwise they wont recognise, rendering difficult to push in a work environment |

#### Menu Style 1

This menu style will be developed from all elements of the robot being accessed on a single page. This means this menu will be the simplicist to code, but may be tedious to use for longer lists of commands, so will be perfect for early development of the cars movement and algorithms, but will not be sufficient once more commands are introduced.

## Development

## Evaluation