AUTONOMOUS VEHICLE

Guided By:

Dr. V. Sathiesh Kumar,

Assistant Professor,

Department of Electronics Engineering,

Madras Institute of Technology,

Anna University,

Chennai - 600 044.

Presented by:

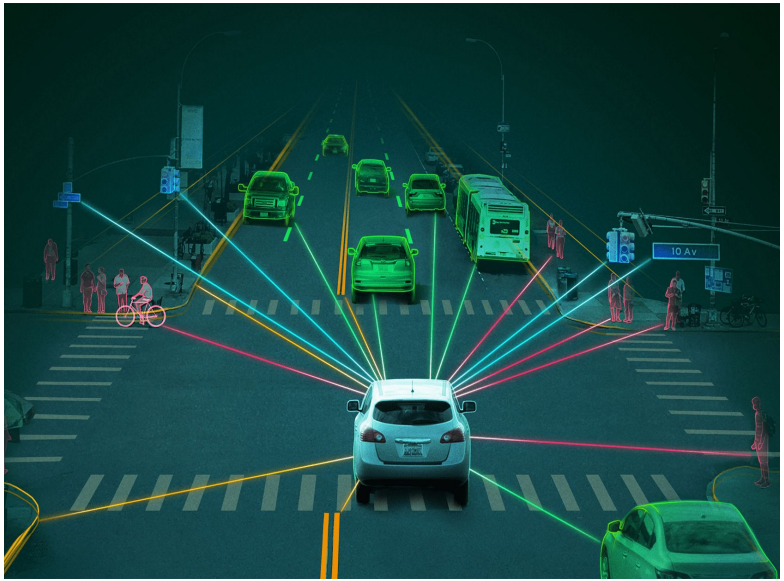
Vaikunth Guruswamy - 2019504602

Tarun U - 2019504597

Enesh Naren - 2019504517

INTRODUCTION

- This is the Phase 2 of our project where we implement level 5 automation into our vehicle.
- The purpose of this project is to develop autonomous vehicles that can cooperatively improve traffic and raise safety without the usage of a centralized server by sharing sensor data.
- The goal is to provide a secure set of protocols for automobile navigations, as well as guidelines for how they should behave.



LITERATURE REVIEW

AUTONOMOUS CARS: RESEARCH RESULTS, ISSUES AND FUTURE CHALLENGES

- NavLab experimental analysis on Autonomous Land Vehicle (ALV)
- Detailed description about different modes of communication used in autonomous car
- LIDAR generated data and other RGB images as input and predict a three-dimensional (3D) representation of that data.
- Software complexities and maneuver motor controls

[1] Rasheed Hussain, and Sherali Zeadally, “Autonomous Cars: Research Results, Issues and Future Challenges” IEEE Communications, September 2018

- Inter Vehicle Communication (IVC) has been examined each of the vehicle movement and to better identify the driving behavior
- Using YOLOV3 to detect obstacles
- Implemented feature pyramid network and Q learning Reinforcement learning for movement
- [2] Mehdi Masmoudi, Hakim Ghazzai, Mounir Frikha, and Yehia Massoud “Autonomous Car-Following Approach Based on Real-time Video Frames Processing” IEEE International Conference on Vehicular Electronics and Safety (ICVES), 2019

AUTONOMOUS CAR-FOLLOWING APPROACH BASED ON REAL-TIME VIDEO FRAMES PROCESSING

LITERATURE REVIEW

DESIGN & IMPLEMENTATION OF REAL TIME AUTONOMOUS CAR BY USING IMAGE PROCESSING & IOT

- Neural system in the field of a self-driving vehicle or Autonomous Vehicle or Driving Assistant System (DAS).
- Using a deep convolutional neural network and various image processing Techniques detecting real-time objects
- Detecting Lane using a neural network. Classifying the detected object using deep neural network.
- Implementation on raspberry pi along with camera module

[3] Irfan, Karunakar Pothuganti “Design & implementation of real time autonomous car by using image processing & IoT ” International Conference on Smart Systems and Inventive Technology (ICSSIT 2020)

- A CNN model with Monocular vision algorithm, Haar cascade classifier CNN used to enable lane detection haar cascade classifier used to detect signals ultrasonic sensor used for front collision avoidance
- The COCO dataset is used with a model of MobileNet v2 COCO Quantized model
- Lane line detection using OpenCV
- [4] Hiral Thadeshwar, Vinit Shah, Mahek Jain, Prof. Rujata Chaudhari, Prof. Vishal Badgujar “Artificial Intelligence based Self-Driving Car” 4th International Conference on Computer, Communication and Signal Processing (ICCCSP) 2020

ARTIFICIAL INTELLIGENCE BASED SELF-DRIVING CAR

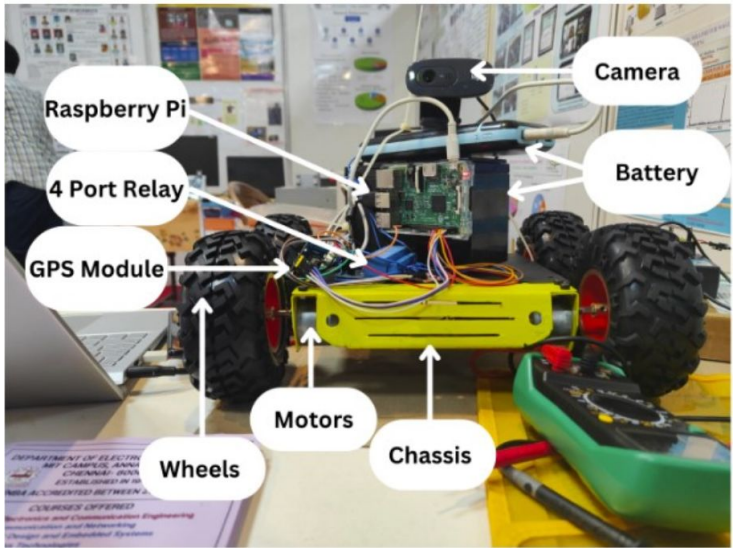
LITERATURE REVIEW

DRIVERLESS CAR: AUTONOMOUS DRIVING USING DEEP REINFORCEMENT LEARNING IN URBAN ENVIRONMENT

- Used sensor fusion on two types of input sensor data: vision based camera sensor and a laser sensor.
- Hokuyo Lidar Sensor and camera sensors embedded with Nvidia Jetson TX2 embedded GPU
- A deep neural network based on fully connected CNN to approximate Deep reinforcement learning based Q-function.
- The network consists of 3 convolutional layers and 4 dense layers.
- Two types of data input: front camera image (RGB image of size 80 x 80) and LIDAR data
- (grid-map of size 80 x 80) are fed into the neural network, processed using the convolutional layers for each data type presented a reinforcement-learning based approach with Deep Q Network implemented in autonomous driving.
- Concatenation of LIDAR data and camera image is used to provide high accuracy
- Deployed and trained by custom environment created by UNITY.
- [5] Abdur R. Fayjie, Sabir Hossain, Doukhi Oualid, and Deok-Jin Lee, “Driverless Car: Autonomous Driving Using Deep Reinforcement Learning In Urban Environment” 15th International Conference on Ubiquitous Robots (UR) Hawaii Convention Center, Hawai'i, USA June 2018

PHASE -1

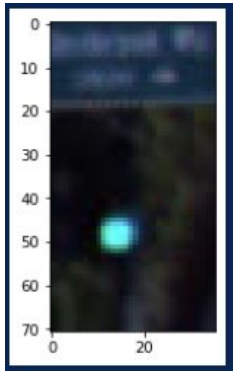
The following functionalities have been implemented during the phase one of our project :



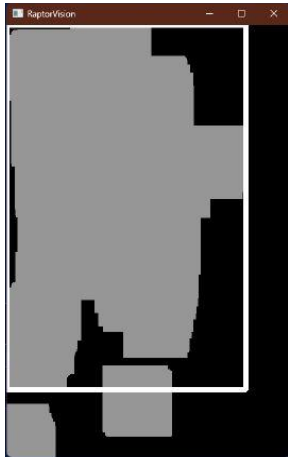
- ➔ Software perspective
  - ◆ Road Sign Detection and Classification
  - ◆ Traffic Light Detection
  - ◆ Road Detection
  - ◆ Obstacle Detection
  - ◆ Web app for additional vehicle control
  - ◆ 2 layered dynamic map
  - ◆ Live Video stream of vehicle's FPV

- ➔ Hardware perspective
  - ◆ Motor instructions based on ORS and the output given by the Deep Neural Network
  - ◆ Interfaced modules such as GPS, Camera, Relay with Raspberry Pi
  - ◆ Interfacing the Raspberry Pi with the web.
  - ◆ Further battery management (12V 7A) with 1000 RPM motors

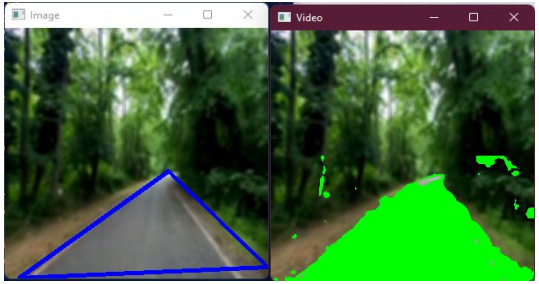
Traffic sign detection



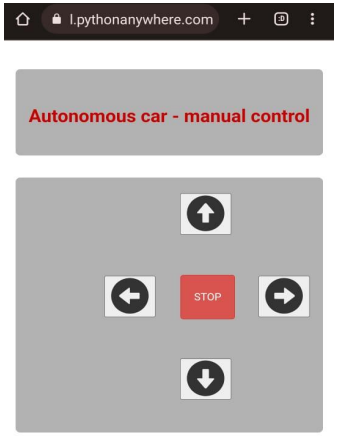
Traffic Light Detection



Obstacle detection



Road detection



Web app

2 Layered map

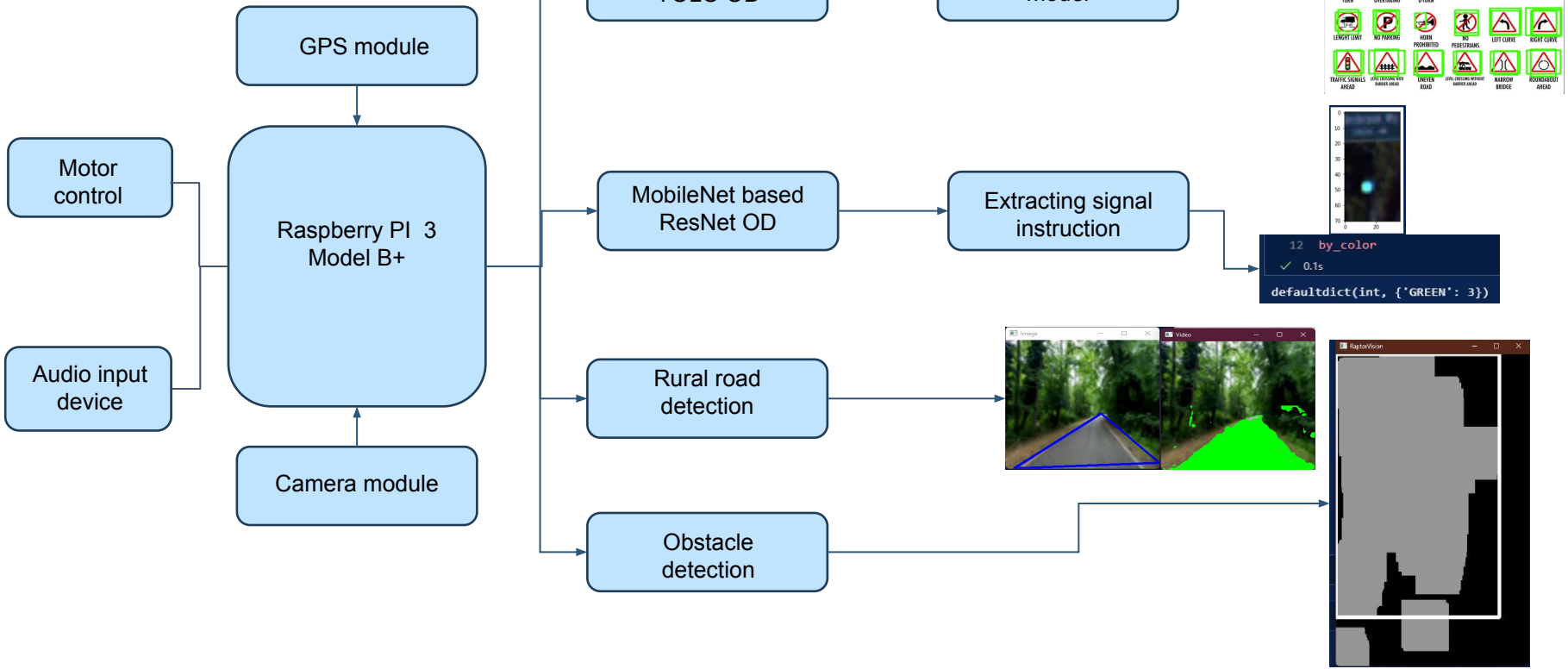
PHASE - II

ZEROTH REVIEW

OBJECTIVE

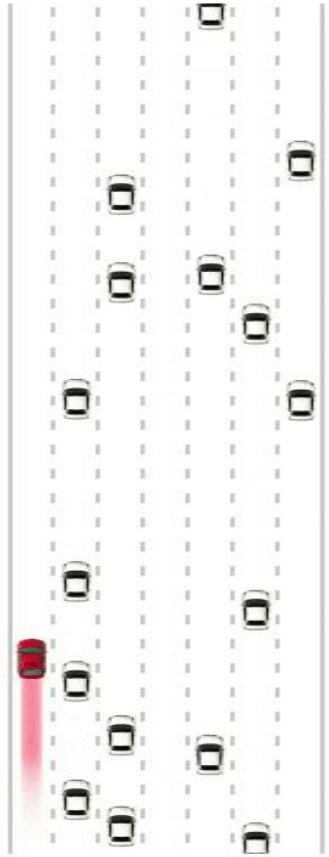
- Substantial weight reduction of the vehicle
- Implementing the rover using a single DC motor
- Speed control using Potentiometer
- Building a steering mechanism using a servo
- Implementing Reinforcement Learning for road and object avoidance
- Thread based approach for parallel processing
- Full-End user interface application

SOFTWARE


GPS module
Motor control
Raspberry PI 3 Model B+
Audio input device
Camera module
Darknet based YOLO OD
Custom built ML model
MobileNet based ResNet OD
Extracting signal instruction
Rural road detection
Obstacle detection
12 by_color
0.1s
defaultdict(int, {'GREEN': 3})

WORKFLOW

- **First Review**
  - Implement the rover using a BLDC motor
  - Perform speed control using ESC
  - Building a steering mechanism using a servo motor
  - Integrate the features in Raspberry Pi
  - Implement thread based approach for features
  - Manipulate motor instructions based on software results
  - Implement reinforcement learning algorithm for vehicle navigation - utilizing CARLA, Voyage and Deep Traffic environment



WORKFLOW

- **Second Review**
  - Perform vital testing
  - Check robustness of the rover
  - Increase in the user interactable features
  - Provide software aided vehicle control to the user
  - Optimization both software by reducing time complexity and hardware power utilization
  - Fine tune road detection algorithm
- **Final Review**
  - Fully functioning model of the vehicle will be completely tested and deployed.

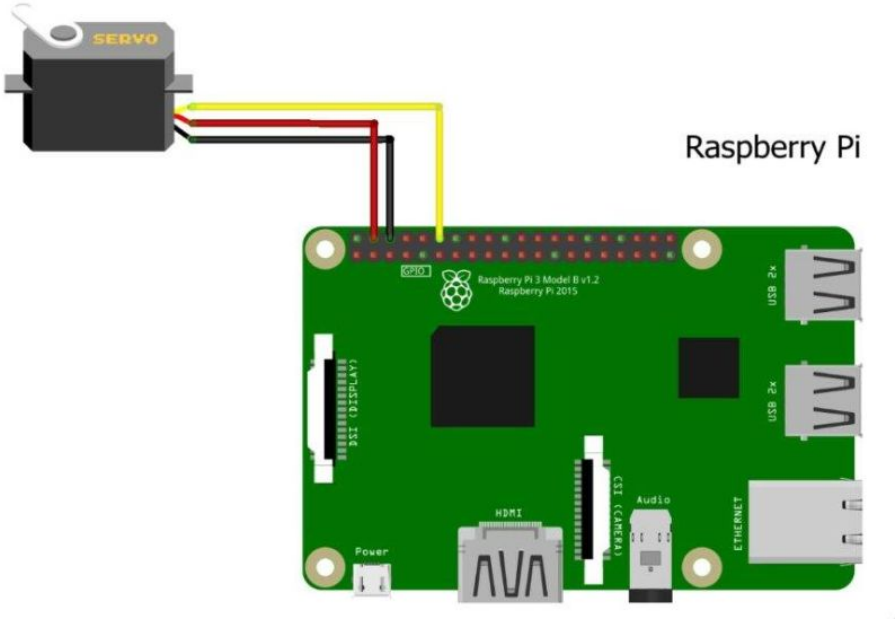
FIRST REVIEW

OBJECTIVES

- Substantial weight reduction
- Optimize the performance of the vehicle
- Interface servo motor with Raspberry Pi
- Implement Ackermann steering mechanism
- Create a custom built chassis to accommodate a steering mechanism
- Perform speed control of motor
- Integrate and run Object detection in Raspberry Pi

INTERFACING SERVO MOTOR WITH RASPBERRY PI

- Raspberry Pi sends PWM signals to servo motor
- Signal interval : 20mS
- Width of pulse determines position of servo
- Map/ scale is used to convert digital signal of Raspberry Pi to angle of rotation

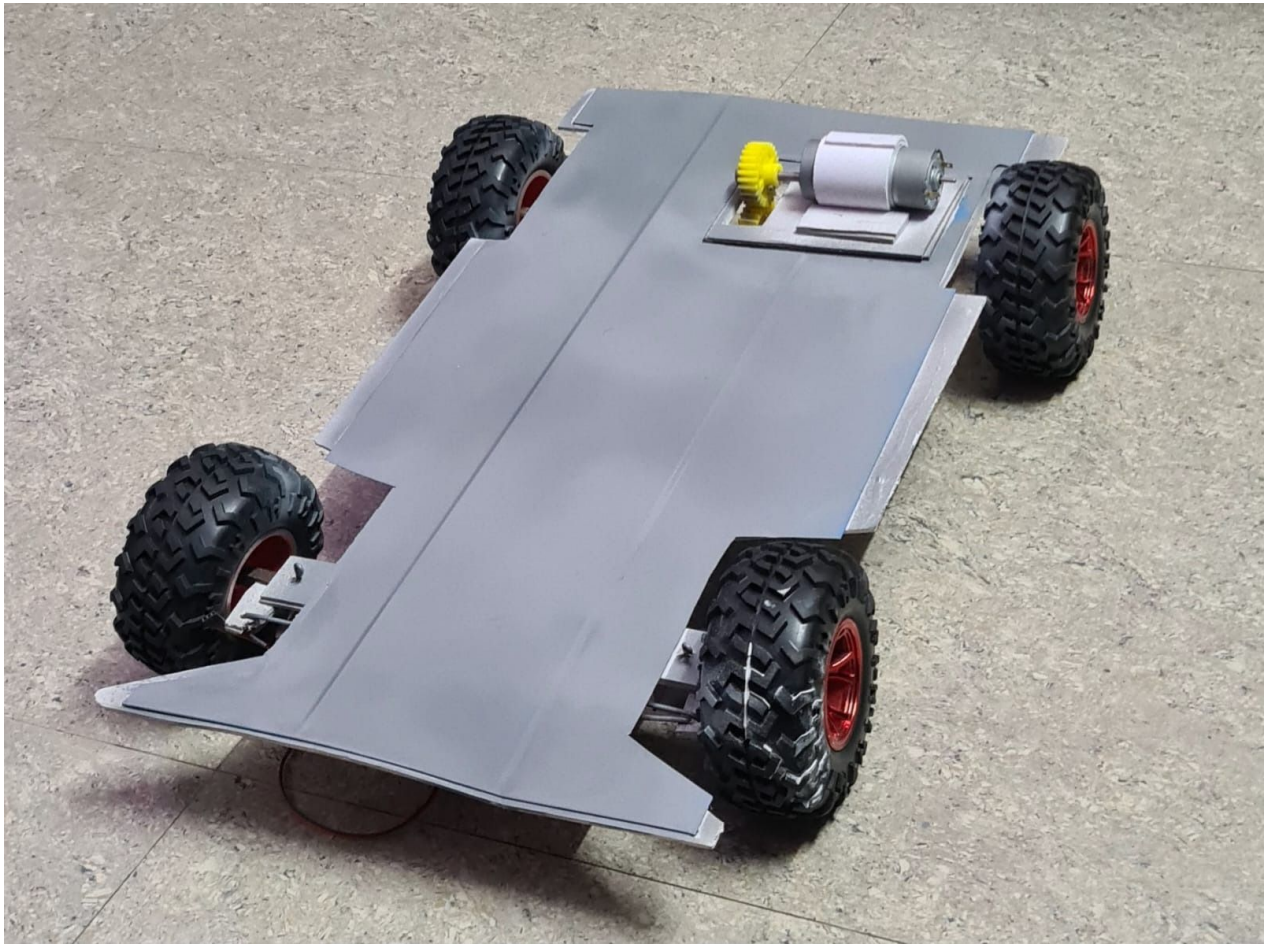


Servo	Raspberry Pi
+ve	Vcc
-ve	gnd
PWM signal	gpio 2,17

COMPARISON BETWEEN PHASE 1 AND PHASE 2

Parameters	1st GEN	2nd GEN
Dimension	26x21	58x34
Weight	3KG	1.6KG
Battery Capacity		
No of Motors	4	1
Run Time	10 mins	30mins
Material used	Steel	Foam Sheet
Steering Mechanism	Differential Steering	Ackermann Steering

HARDWARE IMPLEMENTATION



HARDWARE IMPLEMENTATION - CHASSIS

MOTOR AND BATTERY SPECIFICATIONS

- MOTOR specifications:
  - Requires 12V DC
  - 1000 RPM
  - Torque: 34.3 N-cm
  - Full load current: 6.836 A

- BATTERY specifications:
  - Charge Capacity(C) : 2200mAh.
  - Rated Voltage: 11.1V
  - Exact weight: 150 - 160 Grams
  - LxHxW : 100*20*30mm Approx

- Mini Servo motor specifications:
  - Model: SG90
  - Weight: 9 gm
  - Operating voltage: 3.0V~ 7.2V
  - Servo Plug: JR
  - Stall torque @4.8V : 1.2kg-cm
  - Stall torque @6.6V : 1.6kg-cm

- High Torque Servo Motor specifications:
  - Operating Voltage: 4.8-6.0 Volts
  - Torque:4.8V: 44.0 oz-in (3.17 kg-cm),6.0V: 57.0 oz-in (4.10 kg-cm)
  - Rotational Range:60°
  - Pulse Cycle:30 ms
  - Pulse Width:500-3000 µs

HARDWARE IMPLEMENTATION - POTENTIOMETER

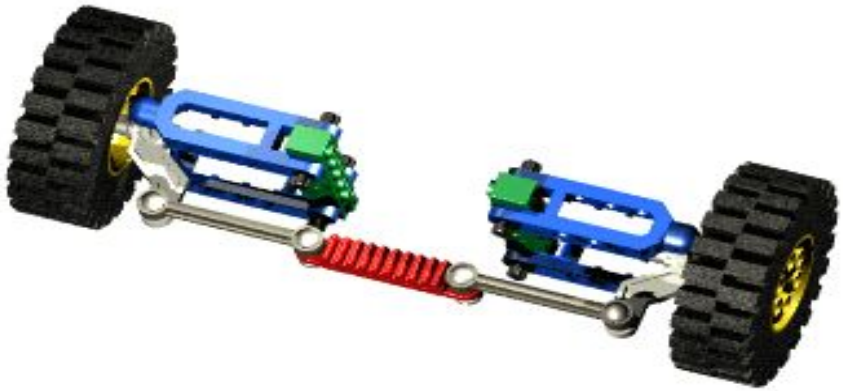
- This is a device that allows us to control the speed of a DC motor by adjusting the voltage and current supplied to it.
- The main advantage of this regulator is that it can work with a wide range of DC motors, from 1.8V to 12V, and provide a maximum current of 2A. This means that it can be used in various applications, such as robotics, automation, and DIY projects.
- The key feature of this regulator is the Pulse Width Modulation (PWM) control, which allows us to adjust the duty cycle of the pulse signal to change the average voltage and current supplied to the motor. This means we can control the speed of the motor very precisely, which is essential in many applications.

HARDWARE IMPLEMENTATION - SERVO MOTOR

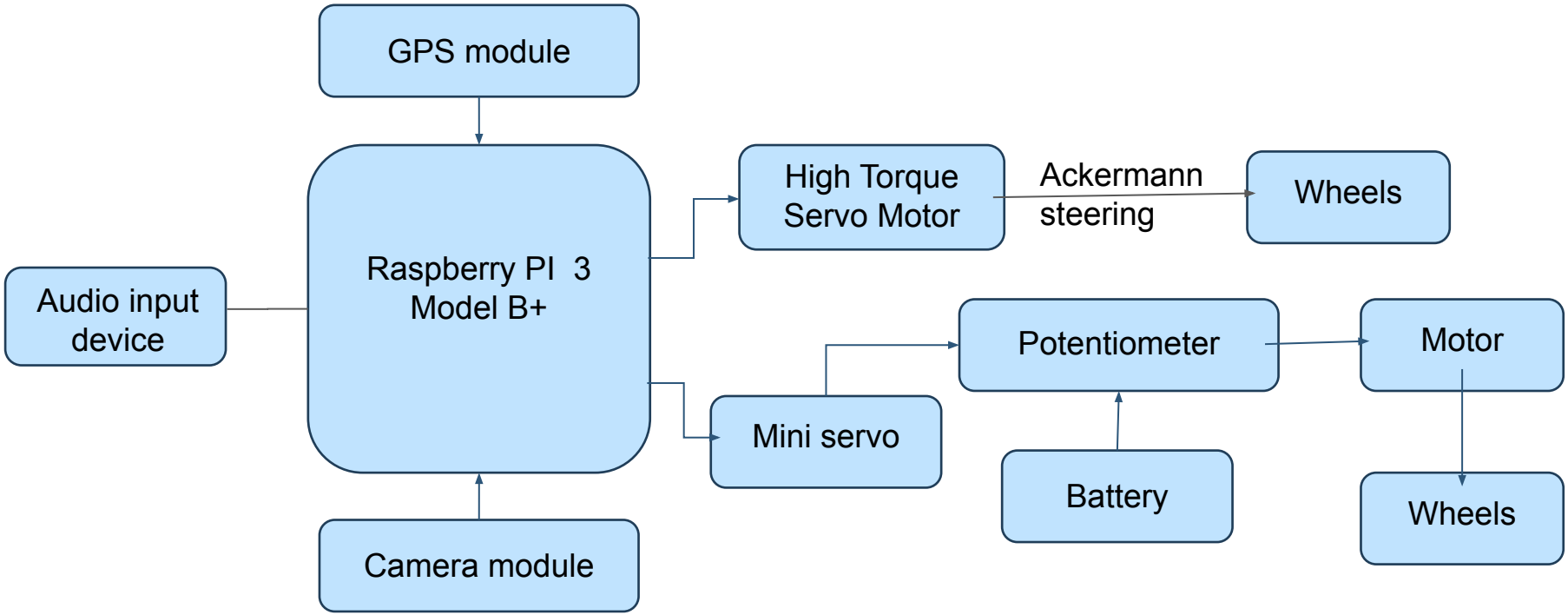
- Servo motors work by using feedback control systems to maintain their position. When a signal is sent to the motor, it rotates to the required position, and the feedback system measures the rotation and sends a signal back to the motor to adjust its position if needed.
- In this project we will be using two servo motors, one high torque servo motor for Ackermann Steering and one mini servo motor for controlling the potentiometer.

Servo Steering Mechanism

This is a type of steering mechanism that mimics the steering of a real car. It involves using two wheels that turn on pivots at different angles to make the car turn smoothly. This can be achieved by using a linkage system that connects the two wheels together and allows them to turn in sync.



Hardware Block Diagram



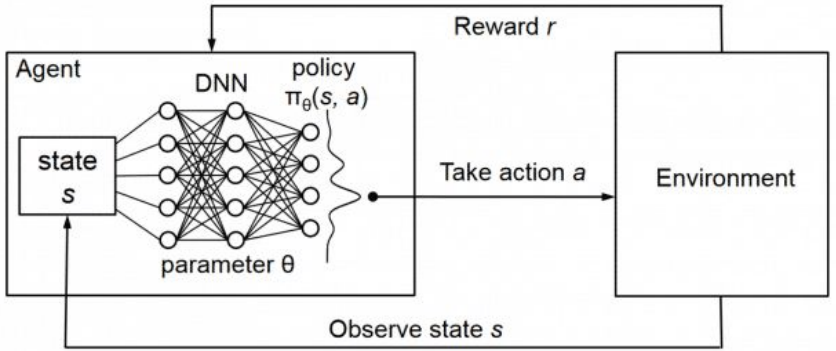
SOFTWARE IMPLEMENTATION - RL ALGORITHM

- The environment used is a experimental based GYM OpenAI
- The Reinforcement procedure used are :
  - Policy based learning algorithm (π)
  - Deep Q-Network(DQN) learning based model
- The Action Space consists of 5 actions (left, right, gas, brake, null state)
- A top-down 96x996 RGB image as Observation Space
- The reward is -0.1 every frame and +1000/N for every track tile visited, where N is the total number of tiles visited in the track

- Executed for 300000 learning cycles
- Achieved a maximum score of 947

SOFTWARE IMPLEMENTATION - RL ALGORITHM

- It sets a **'target divergence'** δ; we would like our updates to be somewhere within the neighborhood of this divergence. The target divergence should be large enough to substantially alter the policy, but small enough for updates to be stable.



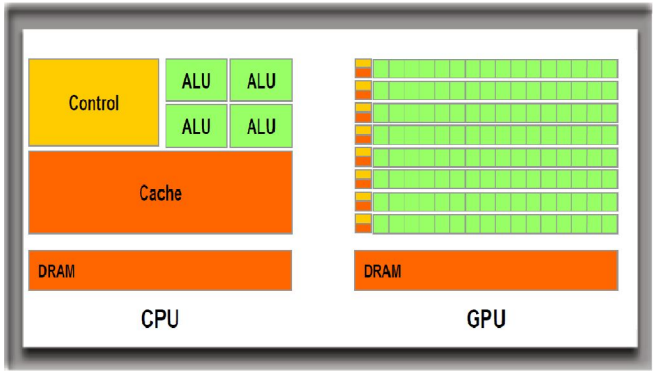
$$Q(s_t, a) \leftarrow Q(s_t, a) + \alpha \left[r_{t+1} + \gamma \max_p Q(s_{t+1}, p) - Q(s_t, a) \right]$$

$$\Delta\theta^* = \arg\max_{\Delta\theta} \mathcal{L}_{\theta+\Delta\theta}(\theta + \Delta\theta) - \beta(\mathcal{D}_{\mathrm{KL}}(\pi_\theta \parallel \pi_{\theta+\Delta\theta}))$$

k

SOFTWARE IMPLEMENTATION - CUDA PROCESSING

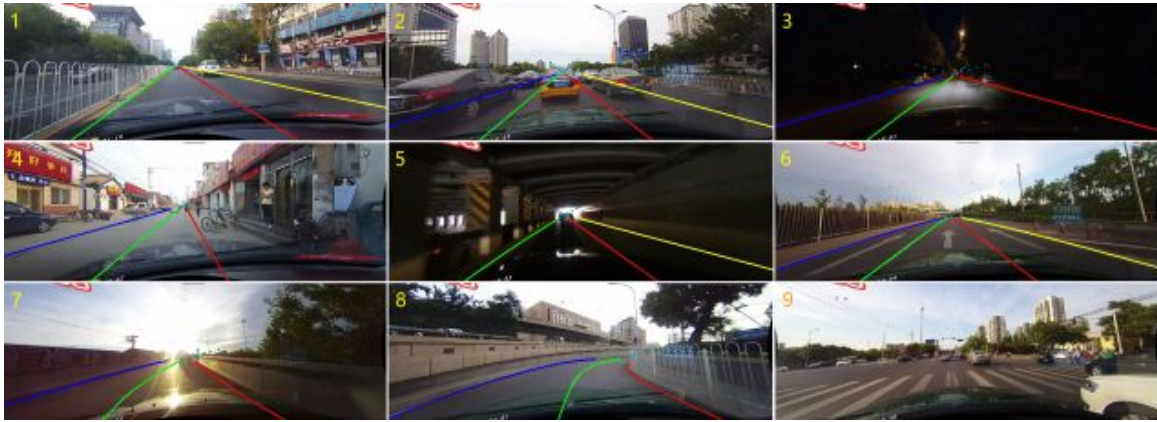
- GPU based Acceleration has been used for both model training purpose and execution of various features of the vehicle
- CUDA technology enables parallel processing by breaking down a task into thousands of smaller "threads" executed independently
- Features such as Traffic sign, Reinforcement env, lane detection extracts a lot of ALU process within the CPU resulting in less FPS - GPU resolves this difficult

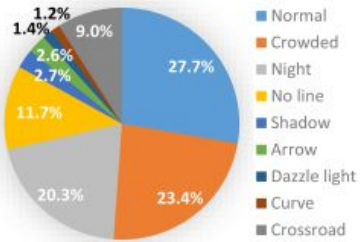


- Hardware accelerator used - NVIDIA GTX 1650
- Reduction in FPS observed from an average of 6.8 seconds per frame to 0.5 seconds per frame

SOFTWARE IMPLEMENTATION - LANE DETECTION

- This proj utilizes CULane dataset (88880 images)
- Conventional methods uses hardcode image processing technique - Hough lines
- Dataset consisting of original image, corresponding mask and coordinates for points.

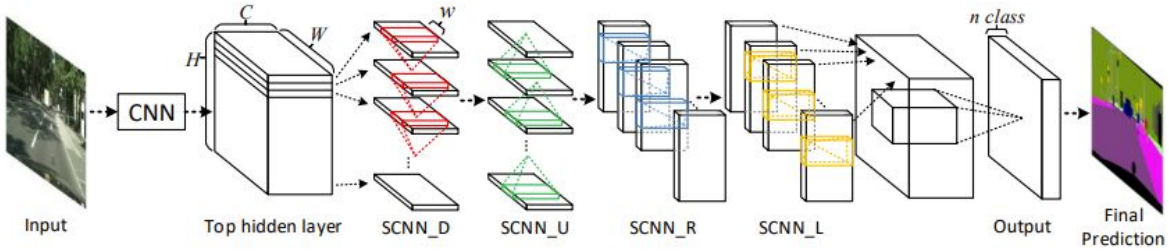




- With a maximum of 4 lanes of classification is possible
- Irrespective of complete visibility of road the lane detection is done
-

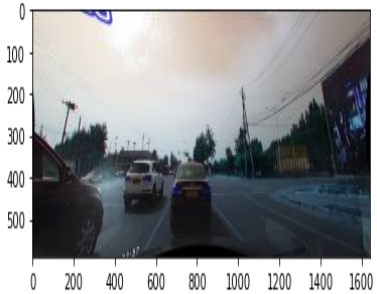
SOFTWARE IMPLEMENTATION - LANE DETECTION

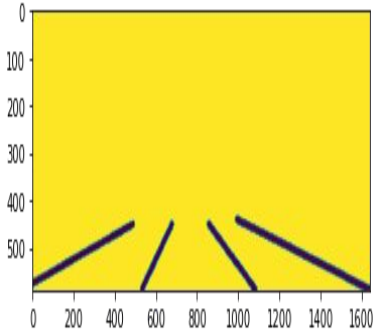
- SCNN applied on a 3-D tensor of size C × H × W where C, H, and W denote the number of channel, rows, and columns respectively.
- Model trained for entire dataset and over 16 Lakhs trainable parameters were trained
- With an MSE of 11.4K over 10 epochs

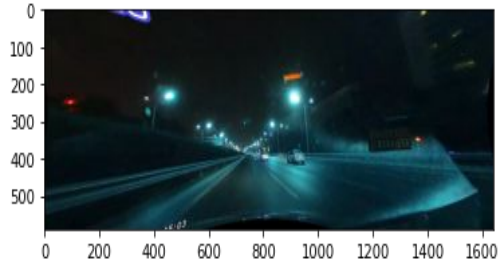


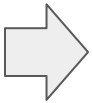

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 300, 300, 1)]     0
conv2d_9 (Conv2D)            (None, 300, 300, 32)      320
conv2d_10 (Conv2D)           (None, 300, 300, 64)      18496
conv2d_11 (Conv2D)           (None, 300, 300, 64)      36928
conv2d_12 (Conv2D)           (None, 300, 300, 128)     73856
conv2d_13 (Conv2D)           (None, 300, 300, 128)     147584
conv2d_14 (Conv2D)           (None, 300, 300, 64)      73792
max_pooling2d (MaxPooling2D  (None, 150, 150, 64)      0
)
conv2d_15 (Conv2D)           (None, 150, 150, 256)     147712
conv2d_16 (Conv2D)           (None, 150, 150, 512)     1180160
conv2d_17 (Conv2D)           (None, 150, 150, 1)       4609
=================================================================
Total params: 1,683,457
Trainable params: 1,683,457
Non-trainable params: 0
```

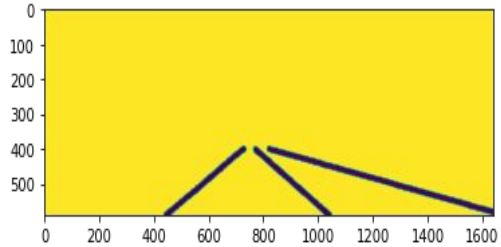
SOFTWARE IMPLEMENTATION - LANE DETECTION









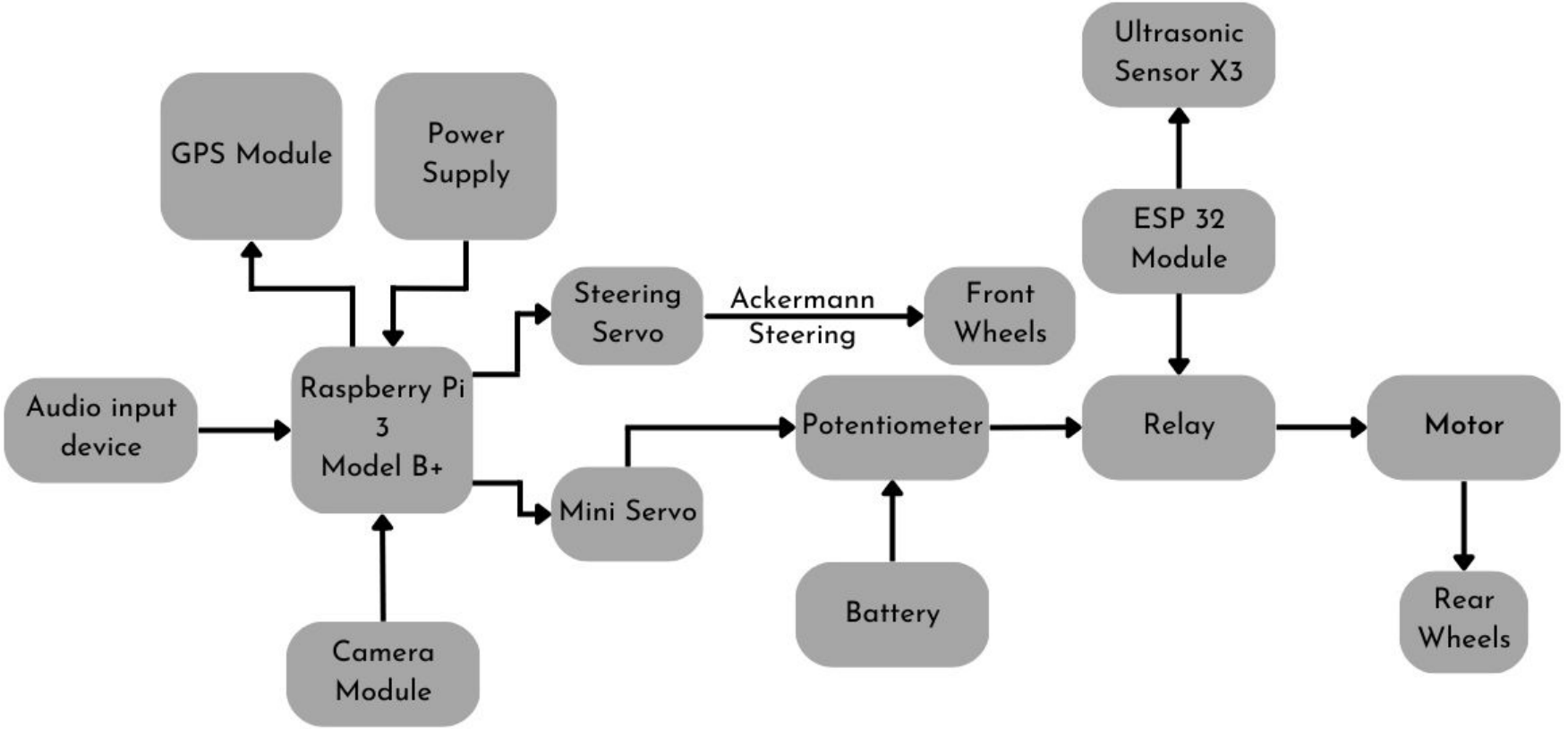


SECOND REVIEW

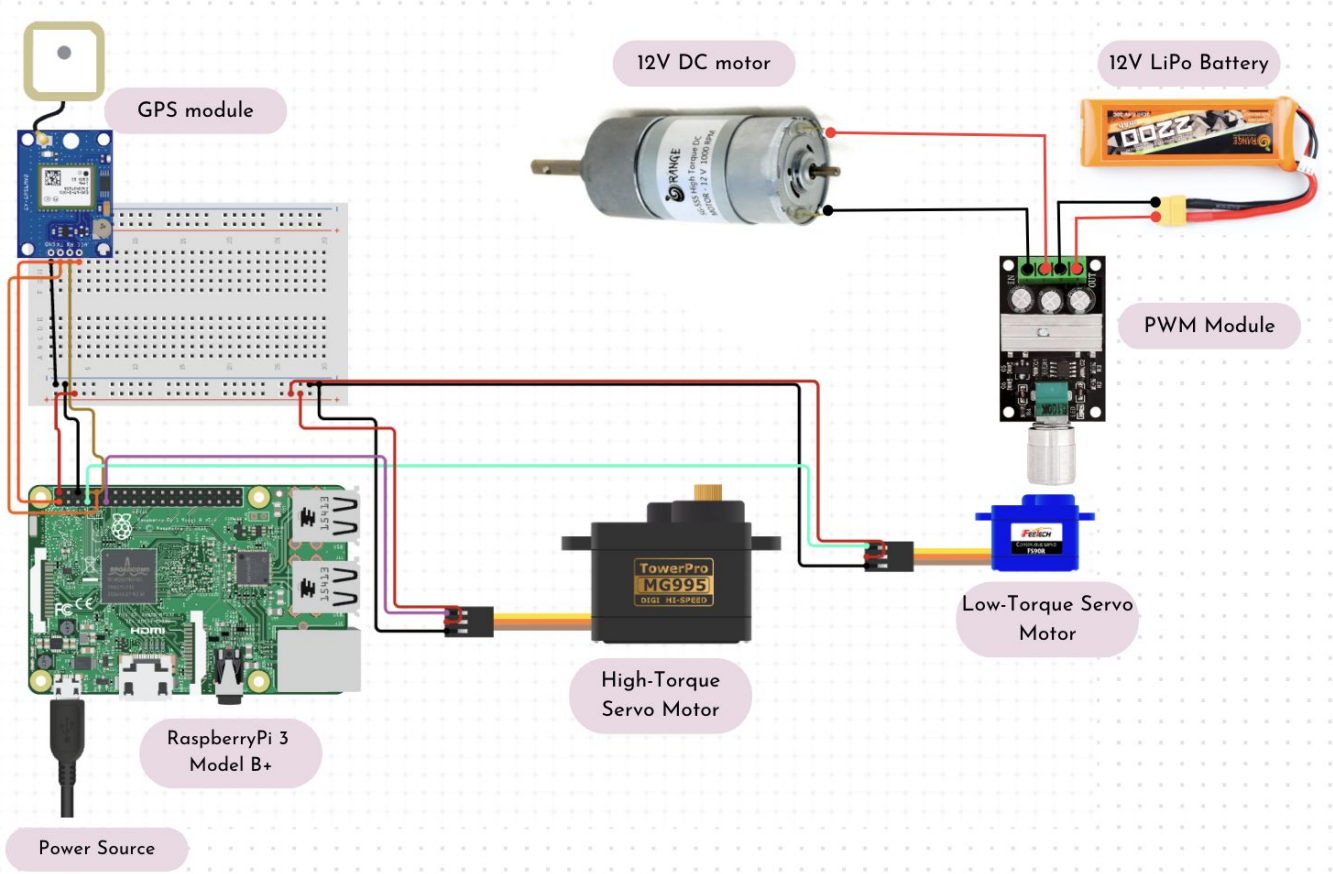
OUTCOMES

- The chassis was made more rigid and tensile with a metal build
- The precision of the vehicular speed control was enhanced
- The angular displacement of the steering mechanism of the vehicle was amplified, resulting in a greater degree of turning of the wheels
- Implementing Reinforcement Learning for road and object avoidance
- Thread based approach for parallel processing
- Full-End user interface application

BLOCK DIAGRAM



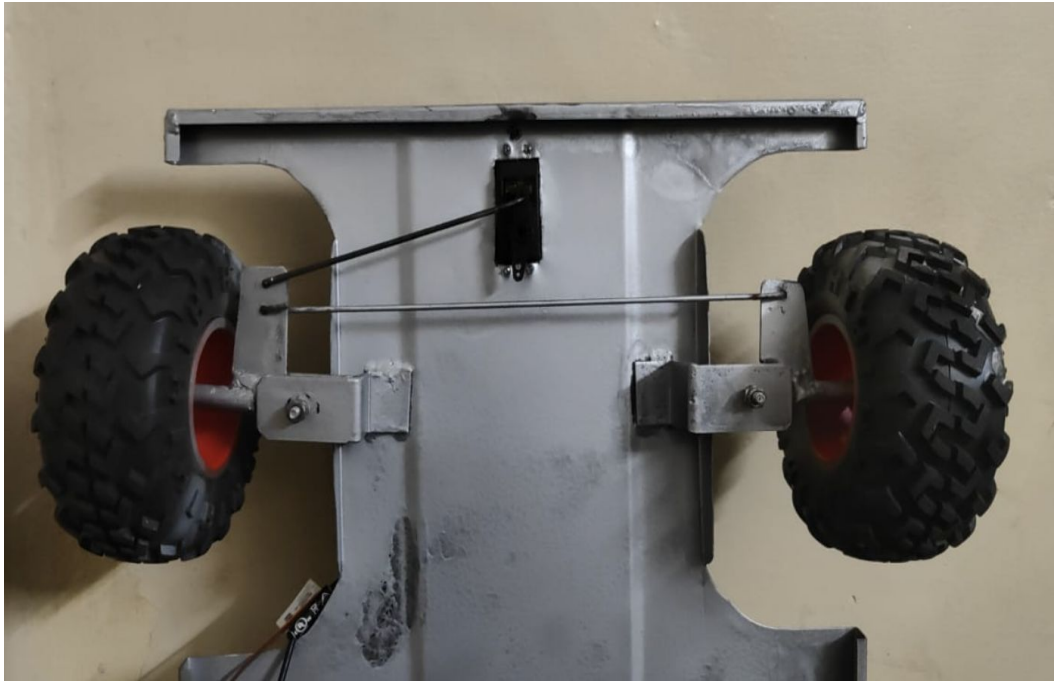
SCHEMATIC DIAGRAM



OBJECTIVES	IMPLEMENTATION
Substantial weight reduction of the vehicle	✓
Implementing the rover using a single DC motor	✓
Speed control using Potentiometer	✓
Building a steering mechanism using a servo	✓
Implementing Reinforcement Learning for road and object avoidance	✓
Thread based approach for parallel processing	✓
Full-End user interface application	✓

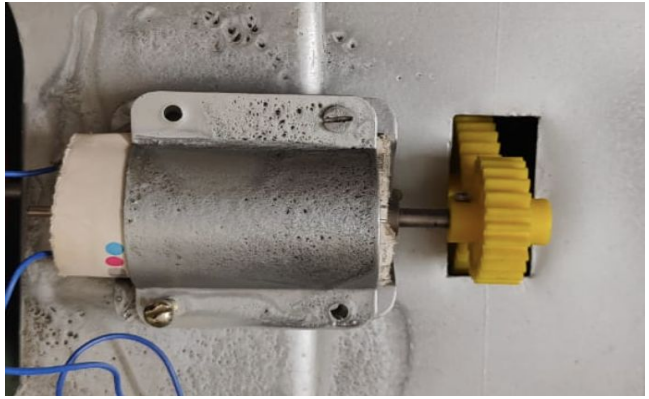
ACKERMAN STEERING MECHANISM

We have implemented an enhanced version of the Ackermann steering mechanism in our vehicle. The steering of the wheels is achieved through a servo motor, which is connected to a metal wire that links the two wheels.

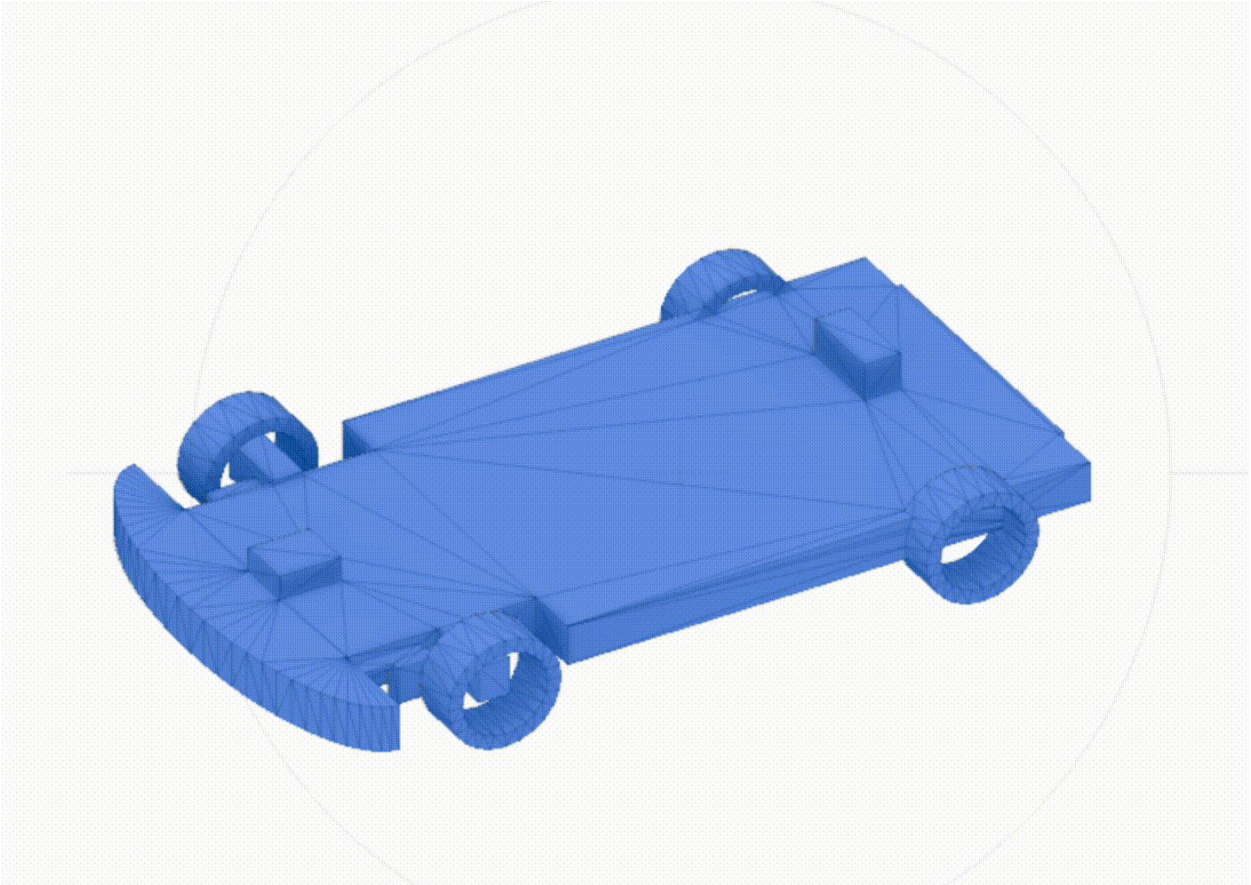


MORE ON MECHANICAL PARTS

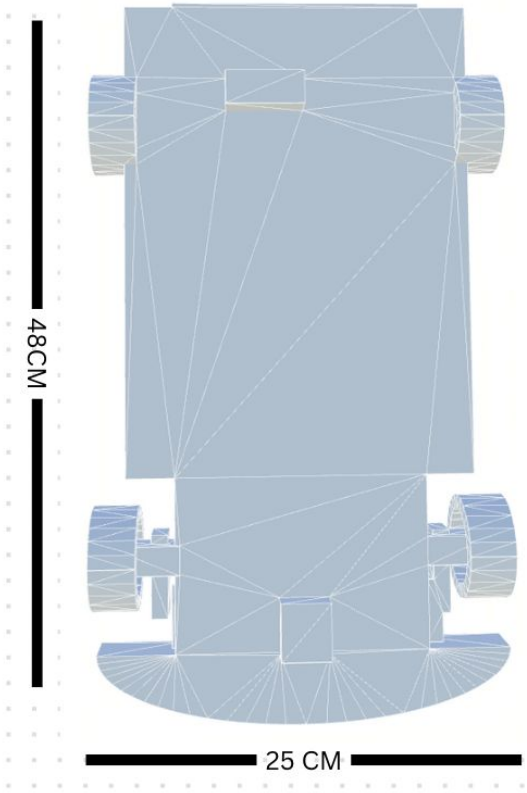
The motor is secured firmly to the chassis through the use of a clamp to ensure precise alignment between the gear teeth of the motor and those of the wheels, which are connected via the shaft, thus preventing any mismatch between the two.

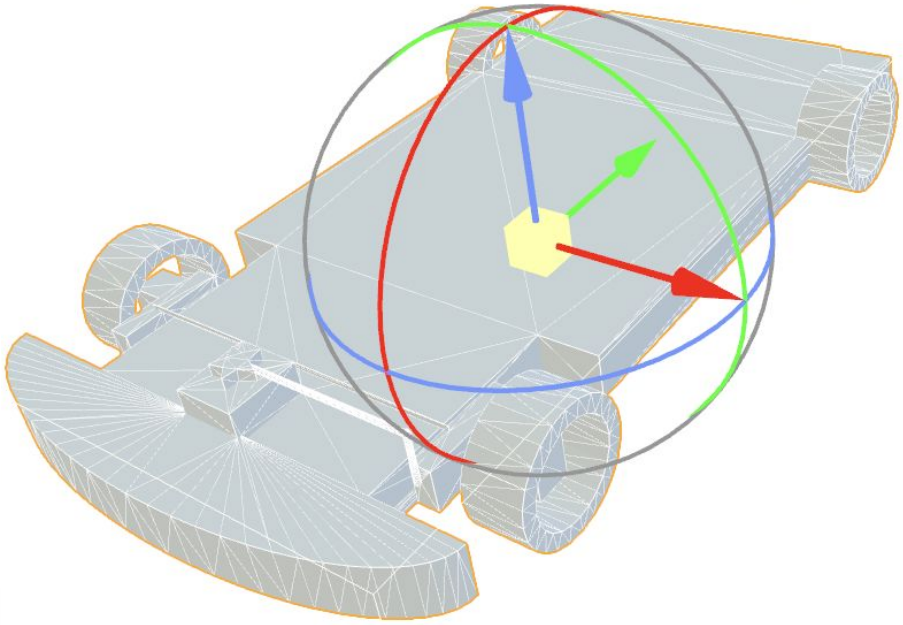


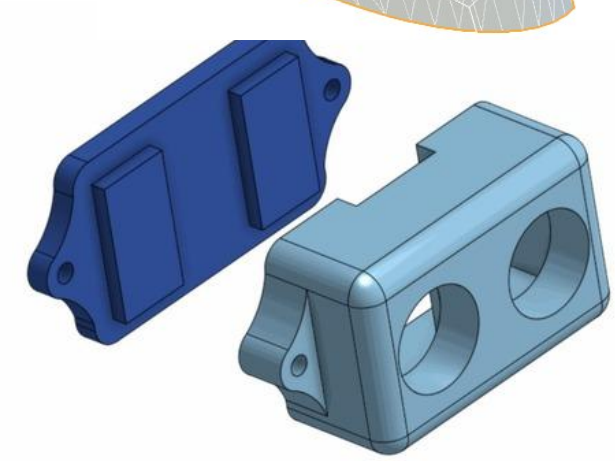
CAD Design of the Vehicle



CAD Design of the Vehicle



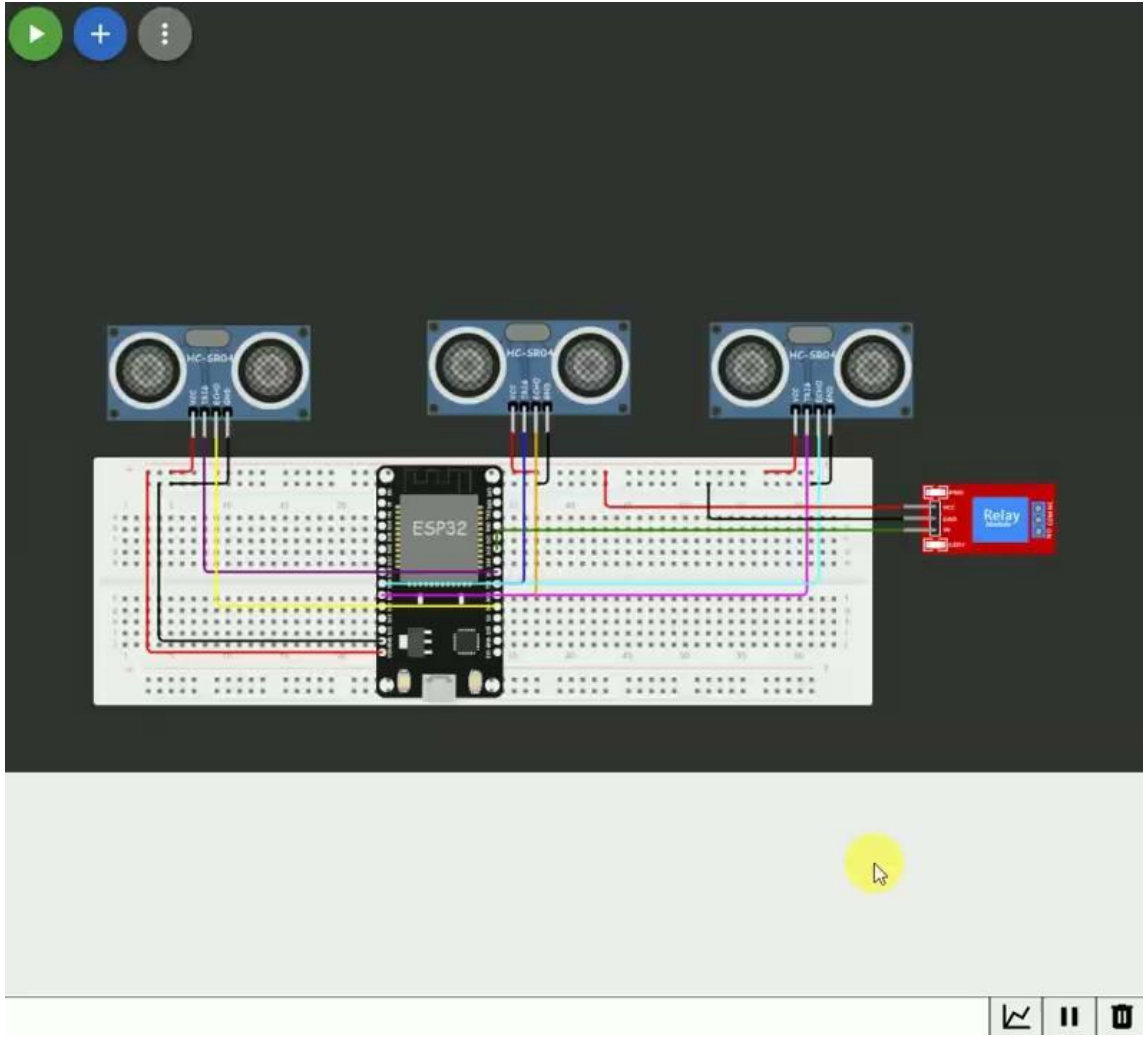




FAIL-SAFE MODE USING ESP8266

- The highlighting feature of an autonomous vehicle is its fast response in order to prevent a collision making it suitable to give a faster reflex to the system
- To achieve the quick response dedicated processing unit is used to establish the connection
- The circuit consists of 3 ultrasonic sensors placed in the front of vehicle
- Total coverage is 150 deg
- Threshold is currently fixed at 20cm and can be customisable
- Upon crossing threshold power supply to the motor is stopped as a result motor de-accelerates
- this process is separate and independent wrt to the main raspberry processes
- response signal from esp will have highest priority in motor control

Circuit Diagram and Simulation

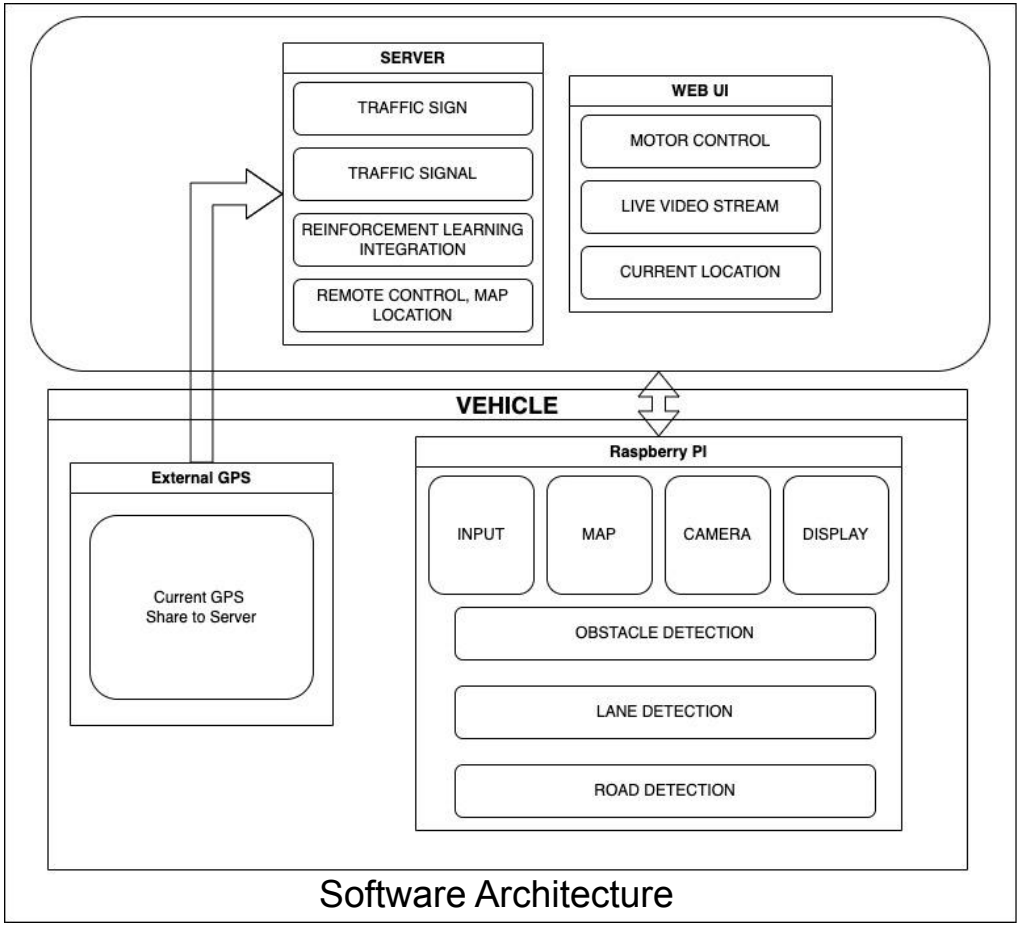


INPUT PROCESSING

- Input is taken through the peripherals such as
  - Camera for road visuals
  - Mic for audio instructions
- Received input is processed within the raspberry pi and instruction set is generated:
  - Checkpoints - latitude and longitude
  - total and intermediate distance
  - angle of deviation
  - direction of rotation
- Normalizing the initial speed of the vehicle : 3m/s
- Based in the instruction generated and camera feed motor is being navigated

SOFTWARE

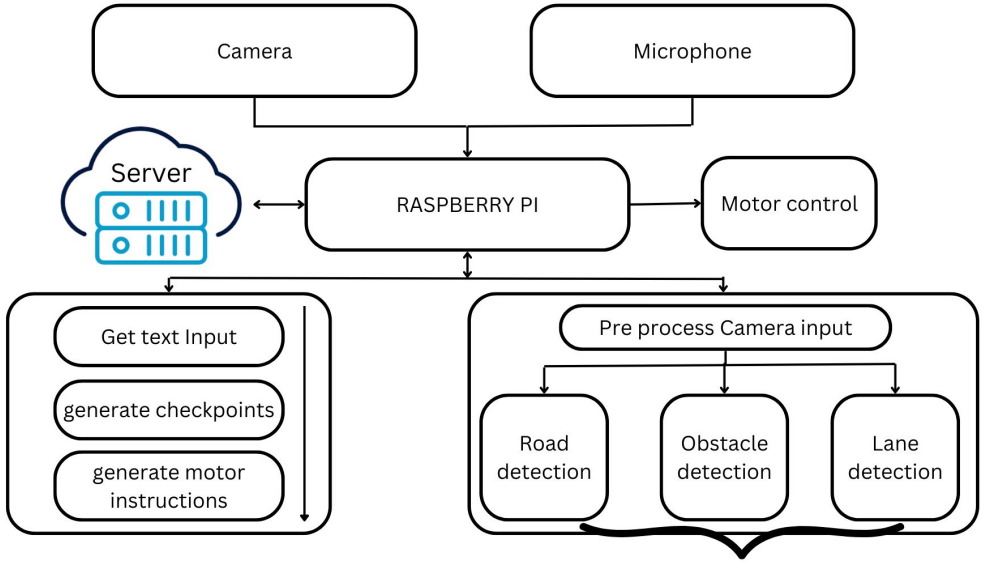
- Working within raspberry pi
- Server functionality
- External gps utilization



Software Architecture

WORKING WITHIN RASPBERRY PI

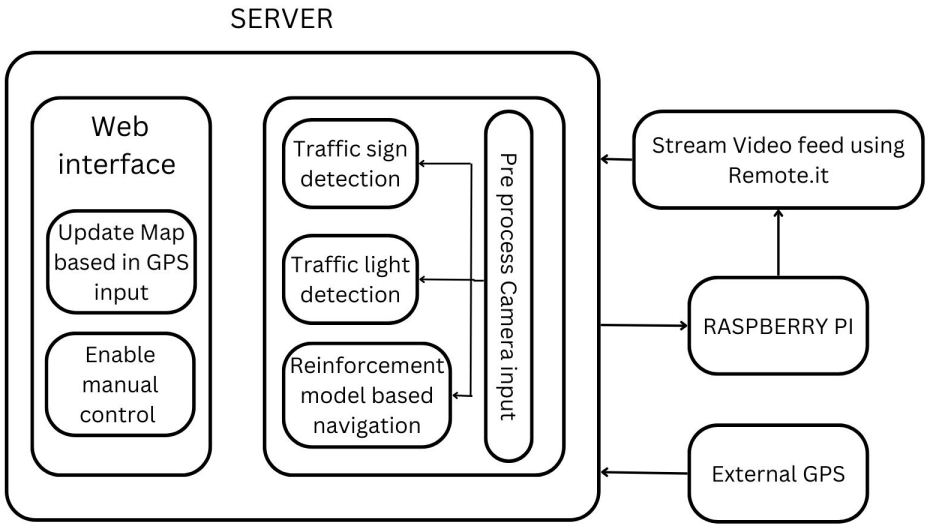
- Along with the iteration of input instructions precautionary measures to avoid collision and to establish safe travel features such - road,lane, and obstacle detections are executed in parallel
- Based upon the alert message triggered immediate termination of vehicle movement is established



3 Modules are being executed in parallel using inbuilt system thread

SERVER FUNCTIONALITY

- Time compensational features such as traffic sign, traffic light, and RL model outputs are executed in parallel within the server
- Based upon the alert signal and response of the model the speed is controlled for the vehicle
- Post this feature the website navigation is enable



Rerouting Algorithm

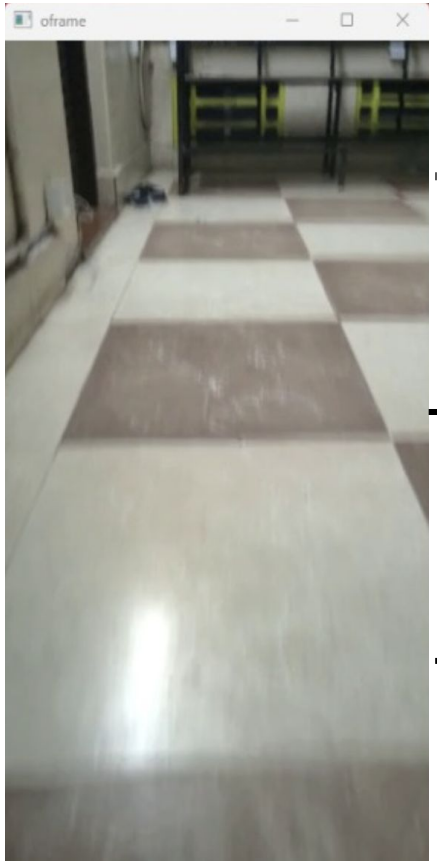
- Depending upon the current position of the vehicle if the navigation is stagnant Rerouting takes place
- Current threshold for this Rerouting to begin after 10 seconds of stay
- Further if the path of the vehicle deviates from the stipulated path new session is created to establish the user need
- With a threshold for distance as 90m deviation from the required path

RESULTS

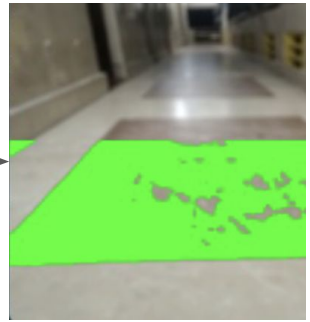
Computational Analysis

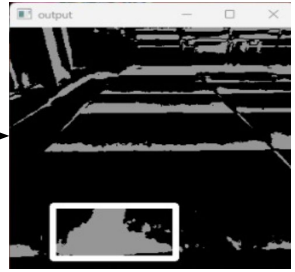
Feature\time	Computational time	location
Traffic Sign Detection	0.9050568 seconds	Server
Traffic Light Detection	0.1142535 seconds	Server
Obstacle Detection	0.0019950 seconds	Native
Road Detection	0.0135312 seconds	Native
Lane Detection	0.2893030 seconds	Native
Importing dependency	43.4 seconds	Both Native and Server
Declaring functions and Global Parameters	47.7 seconds	Both Native and Server
Initial input processing time and rerouting	10.1 seconds	Native

Software Implementation - Native



Road Detection



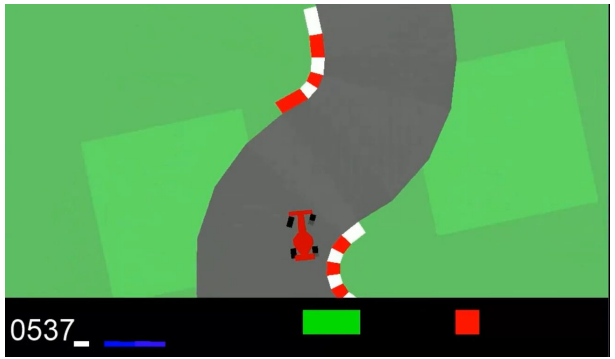


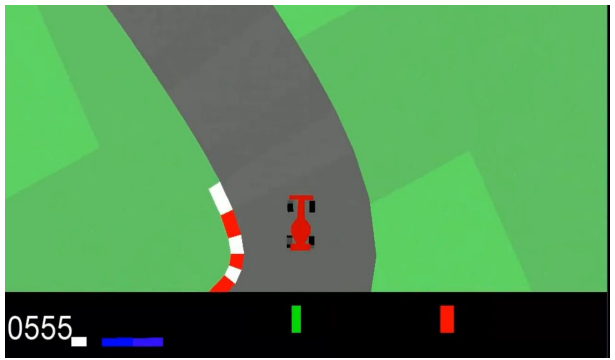
Obstacle detection

Lane Detection

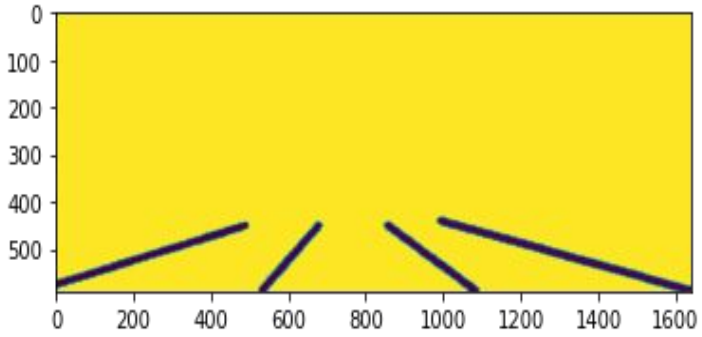
SOFTWARE IMPLEMENTATION - RL ALGORITHM

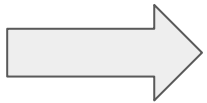
- The environment used is a experimental based GYM OpenAI
- The Reinforcement procedure used are :
  - Policy based learning algorithm (π)
  - Deep Q-Network(DQN) learning based model
- The Action Space consists of 5 actions (left, right, gas, brake, null state)
- A top-down 96x996 RGB image as Observation Space
- The reward is -0.1 every frame and +1000/N for every track tile visited, where N is the total number of tiles visited in the track
- Executed for 300000 learning cycles
- Achieved a maximum score of 947



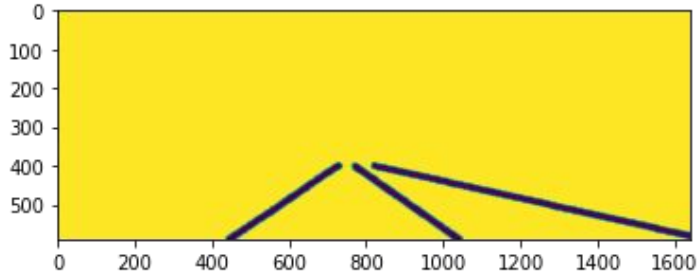


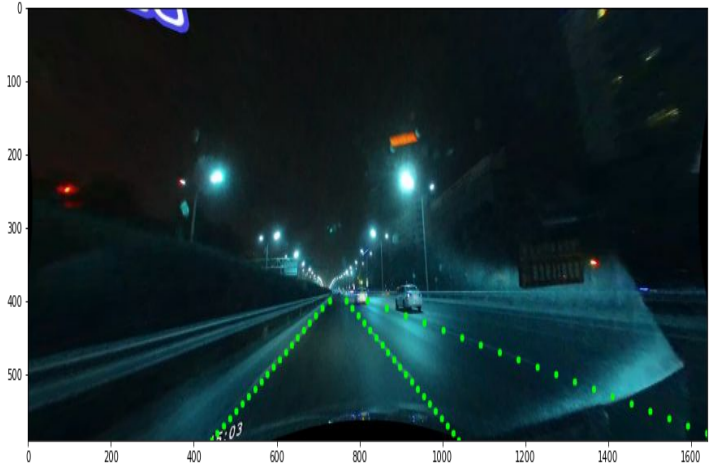
SOFTWARE IMPLEMENTATION - LANE DETECTION





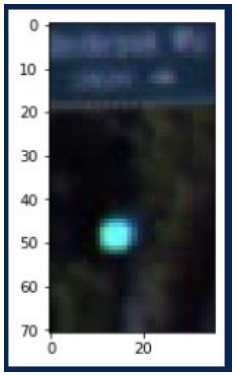
SOFTWARE IMPLEMENTATION - LANE DETECTION

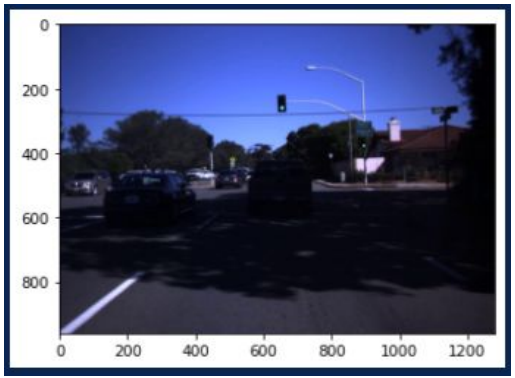




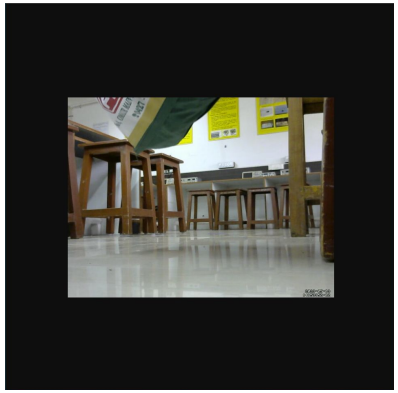
Software Implementation-Server side

Road sign detection





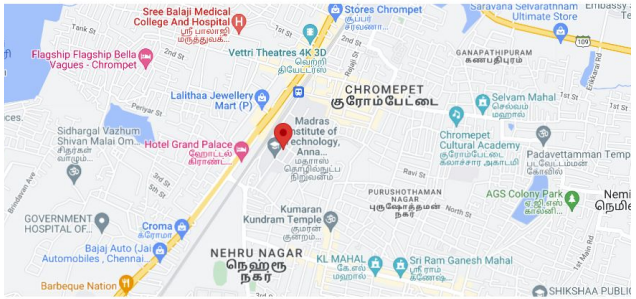
Software Implementation-Server side

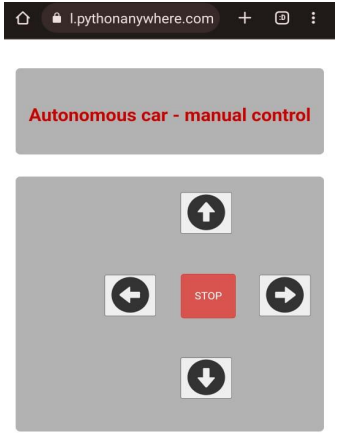


Video Stream

Latitude:12.94848

Longitude: 80.14025





Manual control

Map display

TEAM

DR. V. SATHIESH KUMAR

Assistant Professor,
Department of Electronics Engineering,

MIT Campus, Anna University

VAIKUNTH GURUSWAMY

2019504602

Electronics and Communication Engineering

MIT Campus, Anna University

ENESH NAREN

2019504517

Electronics and Communication Engineering

MIT Campus, Anna University

TARUN U

2019504597

Electronics and Communication Engineering

MIT Campus, Anna University