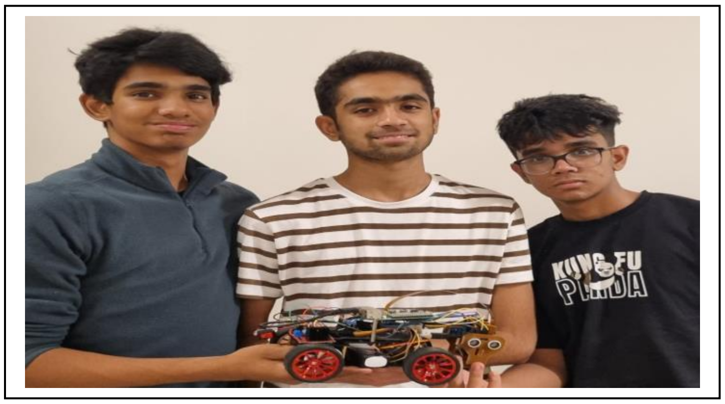
WORLD ROBOT OLYMPIAD™ WRO INDIA

CATEGORY: FUTURE ENGINEERS



TEAM NAME: ROBOTRONS

TEAM MEMBERS NAME:

ADVAITH HARISH, GRADE 10

MUKUND V, GRADE 10

SIDDANTH KARTHA, GRADE 10

MENTOR NAME: PURUSHOTHAM NK

SCHOOL : GOPALAN NATIONAL SCHOOL

GOPALAN NATIONAL SCHOOL

ICSE Code : KA 217

COACHING SUPPORT : Pair India

PAiR Ai | ROBOTICS | IoT

WORLD ROBOT OLYMPIAD™ WRO® INDIA

TABLE OF CONTENTS

1. TEAM INTRODUCTION AND ROLES

Team ROBOTRONS have been a regular participant at major Robotics competitions, the recent one being the Vigyantram National Robotics Competition, held at the Indian Institute of Mumbai, India, where they won the 1st prize amongst 50+ finalists from different schools in India.

Mukund V – Concept and Software Development

Mukund currently studies in Grade 10 at Gopalan National School and has been an avid developer of code for school and hobby projects. His specialization is in Java and Python and intends to pursue Computer Science as a future career with interests in Space and Robotics.

Mukund has worked on a high-level framework for the code and on integrating the different hardware components needed for the self-drive car.

Advaith Harish – Concept & Design

Advaith is a student in Grade 10 at Gopalan National School and have shown avid interest in Robotics from an early age. He learnt how to code in languages like HTML, Python and C at a very young age. He intends to actively take part in other upcoming robotic competitions with great zeal.

Advaith has worked on the overall Design of the project and its feasibility in real life scenarios.

Siddanth Kartha – Concept and Hardware Development

Siddanth currently studies in Grade 10 at Gopalan National School and has interests in hardware and component assembly. His specialization has been motion controllers and integration of different components and he intends to pursue Robotics as a future career with interests in device controllers.

Siddanth has been involved in identifying the different hardware components required and available to make the self-drive car work.

2. PROJECT SUMMARY

Team Robotrons have developed this self driving car project as part of the Future Engineers category for the World Robotics Olympiad nationals. This was done using Raspberry Pi, Arduino, gyroscope-magnetometer, ultrasonic sensors, and a servo motor. Please find below a brief overview of each of the systems.

2.1 Main Control System:

- **Raspberry Pi**: This serves as the central processing unit for high-level operations like image processing, decision-making, and coordination of sensor data. The Pi handles communication between sensors, motors, and the camera, providing an overall control layer.
- **Arduino Mega**: This microcontroller manages real-time operations such as controlling motors, reading sensor data, and providing immediate responses to obstacles. It communicates with the Raspberry Pi for instructions and handles the execution of motor movements.

2.2 Sensors:

- **HC-SR04 Ultrasonic Sensors**: These sensors are crucial for detecting obstacles and measuring the distance to objects around the car. The data collected is used for obstacle avoidance and navigation in real-time. Multiple sensors can provide a wide field of view for object detection.
- **Gyro GY-271**: This sensor provides orientation and heading data, acting as a compass to keep the car on course. It helps the car make smooth turns and maintain a straight path by providing feedback on its current direction.

2.3 Camera System:

- **Raspberry Pi Camera v2**: The camera is used for image processing tasks like lane detection, traffic sign recognition, or object identification. With the help of libraries like OpenCV, the Raspberry Pi can process visual data to enhance the car's navigation and decision-making capabilities.

2.4 Motor and Steering Control:

- **L298N Motor Driver**: This controls the motors that drive the wheels of the car. The Arduino Mega sends signals to the motor driver, which then controls the speed and direction of the motors (for forward, backward, and turning).
- **Servo Motors**: These control the steering mechanism of the car. By adjusting the angle of the servo motor, the car can steer left or right, allowing for smooth navigation.

2.5 Control Flow:

- The **Raspberry Pi** processes visual data from the camera and sensor data from the ultrasonic sensors and gyro. It then sends high-level control commands to the **Arduino Mega**.
- The **Arduino Mega** handles the low-level tasks of controlling the motors through the **L298N driver** and adjusting the **servo motor** for steering based on data from the sensors and the Raspberry Pi.
- **Obstacle detection** is performed by the ultrasonic sensors, and **direction control** is maintained using the gyro GY-271, ensuring the car stays on course or makes necessary turns.

2.6 Camera Processing (Optional Enhancements):

- The **Pi Camera v2** is used for object recognition, traffic signal detection. This allows the car to, avoid obstacles and follow specific visual markers.

2.7 Navigation and Driving:

- **Ultrasonic sensors** continuously monitor the surroundings, and computes safe paths, while the **Arduino Mega** controls motor speed and steering through the **L298N** and **servo motors**.
- **Gyro** helps maintain orientation and ensures the car follows its intended path, while visual data from the **Pi Camera** helps enhance navigation decisions.

2.8 Workflow Overview:

1. **Obstacle detection**: Ultrasonic sensors identify objects, and the data is processed by the Arduino for immediate obstacle avoidance.
2. **Direction control**: The gyro provides heading and orientation data to ensure the car stays on its intended course.
3. **Steering and driving**: The L298N driver controls the motors while the servos adjust the steering.
4. **Image processing**: The Pi Camera v2 captures the environment for visual cues like lanes or obstacles, enhancing the car’s autonomy.

3. MOBILITY MANAGEMENT SYSTEM

3.1 Introduction

The mobility management system for our self-driving car is through efficient navigation and control by integrating various components to handle mobility-related tasks. The Raspberry Pi processes data from the Pi Camera for real-time visual recognition, such as obstacle recognition, while the ultrasonic sensors feed distance measurements to the Arduino Mega for collision avoidance. The GY-271 gyroscope provides orientation data to ensure the car maintains its correct heading, enabling smooth directional changes. The Arduino Mega manages the L298N motor driver to control wheel movement and speed, while the servo motors adjust the steering based on sensor input. Together, this system allows the car to autonomously navigate, avoid obstacles, and stay on its intended path with precision.

Step 1 Sensor Integration and Data Acquisition

- **Integrate the GY-271**: We connected the GY-271 to the Arduino Mega via I2C communication (using the Arduino SDA and SCL pins). We used the magnetometer to determine the car's current heading, which is essential for making navigation decisions.

Step 2 Motor and Servo Control

- **L298N Motor Driver**: We interfaced the L298N motor driver with the Arduino Mega to control the speed and direction of the car's motors. The input pins of the L298N were connected to the digital pins on the Arduino and the motor to the output pins of the L298N.
- **Servo Motor for Steering**: We connected the servo motor to one of the PWM (Pulse Width Modulation) pins on the Arduino Mega. The servo will control the steering mechanism for the car.

Step 3 Basic Movement Functions

- **Forward, Reverse, Stop**: We implemented functions to control the car's basic movements by manipulating the L298N's inputs from the Arduino. For example, to move forward, we ensure both motors spin in the same direction.

- **Steering**: We used the servo motor to steer. A function was taken that takes angle as input and adjusts the servo's position accordingly, allowing the car to turn.

Step 4 Navigating and Decision making

- **HC-SR04 Ultrasonic Sensor**: We used 3 ultrasonic sensors mounted on the front of the car to detect obstacles. These were connected to the trigger and echo pins to the Arduino Mega.
- **Implementing Compass Navigation**: We used the heading data from the GY-271 to navigate. For instance, if the car needs to head north and the current heading is east, the system will calculate the required turn to head in the right direction.

- **Basic Obstacle Avoidance**: If an obstacle is detected within a predefined safe distance, stop the motors, or adjust the direction to avoid the obstacle.
- **Advanced Navigation**: Combine distance measurements with sensor data from a GY-271 compass module follow a specific route and to avoid obstacle.
- **Simple Decision Making**: Our decision-making logic was simple, such as turning in a specific direction upon reaching a certain heading or stopping/moving around if an obstacle is detected.

Step 5 Programming the Arduino

- **Code Structure**: We began by initializing all the components and setting up the necessary libraries for the GY-271, HC-SR04 and servo motor. Then, in the main loop, we continuously read the compass heading, decide on the movement based on the heading, and control the motors and servo accordingly.

- **Libraries and Functions**: We utilized the existing Arduino libraries for the GY-271 (such as Wire.h for I2C communication) and servo motor (Servo.h) and wrote specific functions for motor control that interact with the L298N.

3.2 Testing an Iteration

- **Initial Testing**: We started with simple straight-line movements and turns to ensure the car can move forward, reverse, and stop as expected.

- **Navigation Testing**: We tested the compass-based and ultrasonic sensor-based navigation by setting a target direction and having the car adjust its heading to match.

- **Iterate**: Based on the testing results, we refined the control algorithms, especially how the car decides to turn and adjust its speed.

3.3 Wiring the Components

a. **HC-SR04 Ultrasonic Sensor**:
   - VCC to 5V on the Arduino.
   - GND to GND.

   - TRIG to a digital pin (pin 2).
   - ECHO to another digital pin (pin 3).

b. **GY-271 Compass Module**:
   - VCC to 3.3V on the Arduino.
   - GND to GND.
   - SCL to SCL (pin 21 on Mega).
   - SDA to SDA (pin 20 on Mega).

c. **L298N Motor Driver**:
   - Motor outputs to your motors.
   - Input pins (IN1, IN2, IN3, IN4) to digital pins on the Arduino (pins 4, 5, 6, 7).
   - ENA and ENB to PWM-capable pins for speed control (pins 9 and 10).
   - VCC to battery pack +, GND to battery pack - and Arduino GND.

d. **Servo Motor**:
   - Control wire to a digital pin (pin 11).
   - VCC to 5V on the Arduino.
   - GND to GND.

3.4 Programming the Arduino

a. Initialize Components:

   - Include the necessary libraries (Servo.h for the servo, Wire.h for I2C communication, and any library needed for the GY-271).
   - Define pin numbers and initialize variables.

b. Reading from the HC-SR04:

   - Use the pulseIn() function to measure the duration of the echo from the HC-SR04, and calculate the distance to the nearest obstacle.

c. Reading from the GY-271:

   - Initialize the GY-271 and read the heading information. You might need to calibrate the compass for accurate readings.

d. Controlling the Motors with the L298N:

   - Write functions to control the motors' speed and direction based on the inputs from the sensors.

e. Controlling the Servo Motor:

   - Use the Servo.write() function to set the steering angle.

f. Implementing Logic for Obstacle Avoidance and Navigation:

   - Combine the sensor readings to make decisions about the car's movement. For example, if an obstacle is detected within a certain distance, stop or steer away from the obstacle.

- Use the compass module to navigate in a desired direction.

4. POWER AND SENSE MANAGEMENT

4.1 Introduction

We have taken extreme care to ensure that we efficiently manage power consumption while ensuring reliable sensor data collection for navigation and obstacle avoidance. The components that we used, like the Arduino Mega, Raspberry Pi with Camera V2, GY-271 (a magnetometer), HC-SR04 (ultrasonic sensor), L298N (motor driver), and a servo motor, require careful planning of power distribution and sensor data collection.

For the self-driving car, we used a rechargeable battery pack (11.1V lithium-polymer (LiPo) battery to provide a good balance between capacity, rechargeability, and voltage compatibility with the components. The Arduino Mega and Raspberry Pi are powered via voltage regulators that step down the 11.1V to the required 5V. The L298N motor driver can handle the 11.1 V directly for driving motors.

4.2 Power Management

1. **Voltage Regulation**: We used voltage regulators to step down the 11.1V to 5V for the Raspberry Pi, Arduino Mega, and sensors that require 5V.
2. **Power Distribution**: We used a breadboard to manage connections between the battery, regulators, and components. This ensured a clean setup and reduced the risk of short circuits.
3. **Video Processing:** The Raspberry Pi Camera V2 is powered directly from the Raspberry Pi via the camera serial interface (CSI) connector, which does not significantly increase the overall power consumption of the system. However, the processing of video data is computationally intensive and will increase the power demand on the Raspberry Pi. To ensure that the power supply is capable of handling this additional load, we used a 11.1 V 2200mAH LiPO battery with a step-down converter to 5V.

4.3 Sensor Management

The sensors were carefully selected to ensure the right balance of efficient data acquisition and stability. The following sensors were used for the development of the self-driving car :

1. **GY-271 (Magnetometer)**: Provides orientation relative to the Earth's magnetic field. Useful for navigation and heading correction. It's low power and communicates via I2C, which is supported by both Arduino and Raspberry Pi.
2. **HC-SR04 (Ultrasonic Sensor)**: Detects and avoids obstacles. It measures the distance by emitting ultrasonic waves and measuring the time it takes for the echo to return. Multiple sensors may be used around the vehicle for 360-degree coverage. They are inexpensive and have a low power draw.

3. **Servo Motor**: Controls steering. A servo motor is chosen for its ability to move to specific angles, essential for precise steering control. It's powered directly from the battery through the L298N, with signal control from the Arduino.
4. **L298N (Motor Driver)**: Drives the car's motors, receiving control signals from the Arduino. It can handle the high current required by the motors and allows for speed and direction control.

4.4 Bill of Materials

The Bill of Materials include:

- Raspberry Pi (1x)
- Raspberry Pi Camera V2 (1x)
- Arduino Mega (1x)
- GY-271 magnetometer (1x)
- HC-SR04 ultrasonic sensors (multiple, depending on coverage needs)
- L298N motor driver (1x)
- Servo motor (1x)
- 12V LiPo or Li-ion battery (1x)
- Voltage regulators (5V) (multiple, for each component requiring 5V)
- Power distribution board or breadboard (1x)
- Various cables and connectors for I2C, GPIO, and power connections
- Voltage and current sensor module for power monitoring (1x)

5. OBSTACLE MANAGEMENT

The prime component used for Obstacle avoidance is the Raspberry Pi Camera V2 alongside an Arduino Mega, GY-271 magnetometer, HC-SR04 ultrasonic sensors, L298N motor driver, and a servo motor to significantly enhance the vehicle's ability to manage and navigate through obstacles. This setup combines ultrasonic distance measurement with advanced visual processing, enabling the vehicle to understand its environment more comprehensively.

The system leverages the Raspberry Pi for image processing and decision-making, while the Arduino Mega handles real-time sensor data collection and actuator control. The GY-271 provides orientation data, crucial for navigation; the HC-SR04 sensors detect nearby obstacles through ultrasonic waves; the L298N driver controls the motors; and the servo motor adjusts the steering angle.

5.1 Obstacle Detection Strategy

1. **Ultrasonic Detection**: The HC-SR04 sensors continuously scan the front and sides of the vehicle for obstacles, providing immediate distance measurements.
2. **Visual Detection**: The Raspberry Pi Camera Module V2 captures real-time video. Image processing algorithms identify obstacles by shape, size, and colour.

5.2 Flow Diagram for Obstacle Detection and Management

1. Collect data from HC-SR04 sensors and Camera Module.
2. If an obstacle is detected by either system:
    - Assess the obstacle's position, size, and colour.
    - Determine the next course of action depending on the colour:
        1. If green, then turn through left
        2. If red, then turn through right
3. Execute one of the above chosen actions based on the colour.
4. Continue navigation.

Pseudo Code for Combined Obstacle Detection :

```
1. Initialize Ultrasonic Sensors (left_sensor, right_sensor)
2. Initialize Gyroscope (gyro_sensor)
3. Set target_heading = current reading from gyro_sensor
4. Initialize Motor Controls (left_motor, right_motor)
5. Function: Check_Obstacles()
   a. Read distance from left_sensor
   b. Read distance from right_sensor
   c. If left_sensor < threshold OR right_sensor < threshold:
      Return True (obstacle detected)
   d. Else:
      Return False (no obstacle)
6. Function: Adjust_Course()
   a. current_heading = read gyro_sensor
   b. If current_heading deviates from target_heading:
      Calculate correction based on difference
      Adjust left_motor and right_motor to steer towards target_heading

7. Main Loop:
   While True:
      a. If Check_Obstacles() = True:
         Stop motors
         If left_sensor < threshold:
            Rotate right to avoid obstacle
         Else If right_sensor < threshold:
            Rotate left to avoid obstacle
         Update target_heading after avoiding
      b. Else:
         Move forward
```

```
        Adjust_Course()
```

This pseudo code outlines the logic for integrating ultrasonic and visual obstacle detection. This will prioritize the ultrasonic data but also consider the visual information to understand the surrounding environment and take relevant actions.

5.3 Visual Obstacle Detection with Raspberry Pi Camera

Using vision techniques, the Raspberry Pi analyzes video input to detect obstacles. This can involve color detection, shape recognition, or machine learning models to identify specific objects.

Example Code Snippet (Python with OpenCV)

```
1. Import Libraries:
   a. Import OpenCV for image processing
   b. Import necessary Raspberry Pi GPIO libraries (for motor control, etc.)
   c. Import PiCamera module for camera interface

2. Initialize Camera:
   a. Start PiCamera
   b. Set camera parameters (resolution, frame rate, etc.)

3. Function: Process_Frame(image)
   a. Convert image to grayscale
   b. Apply Gaussian Blur to reduce noise
   c. Apply Canny Edge Detection to detect edges
   d. Find contours in the edge-detected image
   e. If significant contours (obstacles) are found:
      Return True (obstacle detected)
   f. Else:
      Return False (no obstacle)

4. Function: Adjust_Course()
   a. Use the center of detected contours (if any) to determine obstacle position
   b. If obstacle is towards the left side of the frame:
      Adjust motors to steer right
   c. If obstacle is towards the right side of the frame:
      Adjust motors to steer left
   d. If no obstacle is detected:
      Move forward

5. Main Loop:
   While True:
      a. Capture frame from PiCamera
```

```
    b. If Process_Frame(frame) = True:
       Stop motors
       Call Adjust_Course() to avoid obstacle
    c. Else:
       Move forward
```

This snippet demonstrates initializing the camera and processing frames to detect obstacles visually based on pre-trained models to identify patterns, contours, and colour.

5.4 Obstacle Avoidance and Navigation

Upon detecting an obstacle, the system evaluates the best course of action, such as stopping, slowing down, or manoeuvring around the obstacle, based on its size, distance, and position relative to the vehicle as well as executing relevant actions based on the colour.

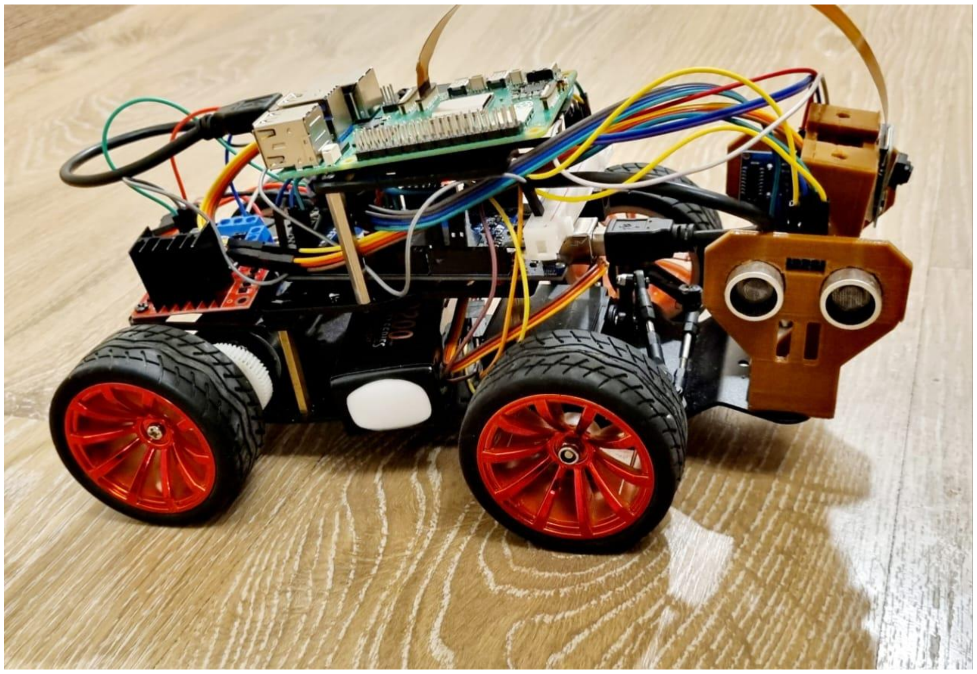
Pseudo Code for Obstacle Avoidance

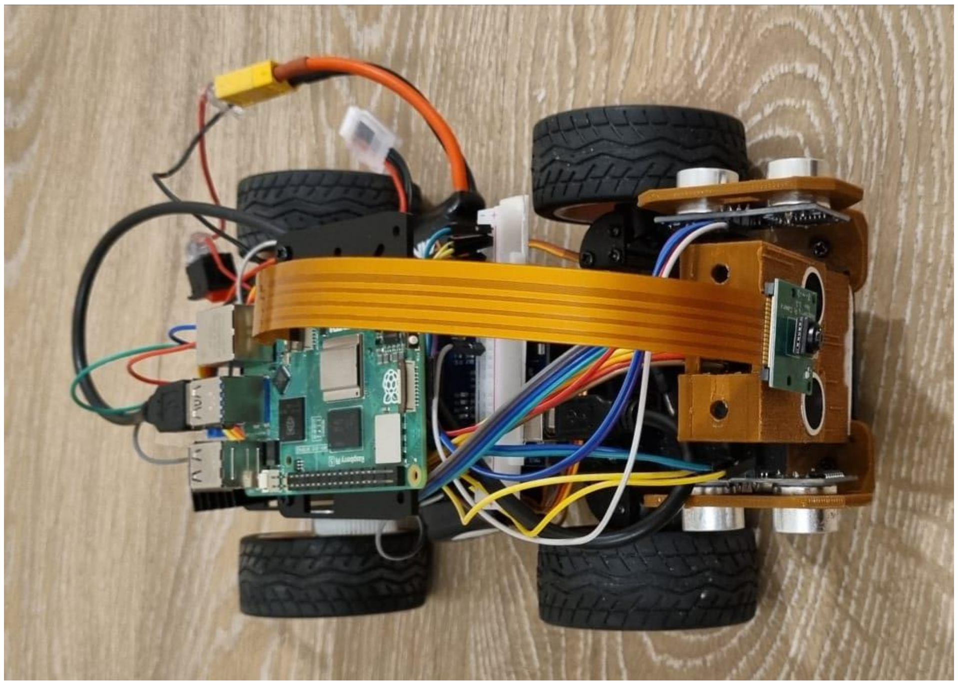
```
1. Import Libraries:
   a. Import OpenCV for image processing
   b. Import PiCamera from picamera2 module for camera interface
   c. Import necessary GPIO libraries for motor control (e.g., RPi.GPIO)
   d. Import NumPy for image manipulation

2. Initialize Camera:
   a. Start PiCamera V2
   b. Set camera parameters (e.g., resolution to 640x480 for real-time performance)
   c. Initialize motor pins (left_motor, right_motor)

3. Function: Process_Frame(image)
   a. Convert image to grayscale
   b. Apply GaussianBlur to reduce noise
   c. Perform Edge Detection using Canny algorithm
   d. Identify Contours from edge-detected image
   e. If significant contours found (i.e., contours representing objects in front):
      Return True (obstacle detected) and contour position (left, center, right)
   f. Else:
      Return False (no obstacle)

4. Function: Adjust_Course(contour_position)
   a. If contour_position is 'left':
      Adjust motors to steer right (slow down right motor, speed up left motor)
   b. If contour_position is 'right':
      Adjust motors to steer left (slow down left motor, speed up right motor)
   c. If contour_position is 'center':
      Stop the robot or rotate to avoid obstacle
   d. If no obstacle:
      Move forward with balanced motor speeds

5. Function: Motor_Control(action)
   a. If action = 'stop':
      Stop both motors
   b. If action = 'move forward':
      Set both motors to forward motion
   c. If action = 'steer left':
      Adjust motor speeds to turn left
   d. If action = 'steer right':
      Adjust motor speeds to turn right

6. Main Loop:
   While True:
      a. Capture frame from PiCamera V2
      b. Resize frame for faster processing (e.g., 320x240 resolution)
      c. Process the frame using Process_Frame()
      d. If Process_Frame() returns True:
         Adjust_Course(contour_position) based on where the obstacle is detected
      e. Else:
         Call Motor_Control('move forward')
```

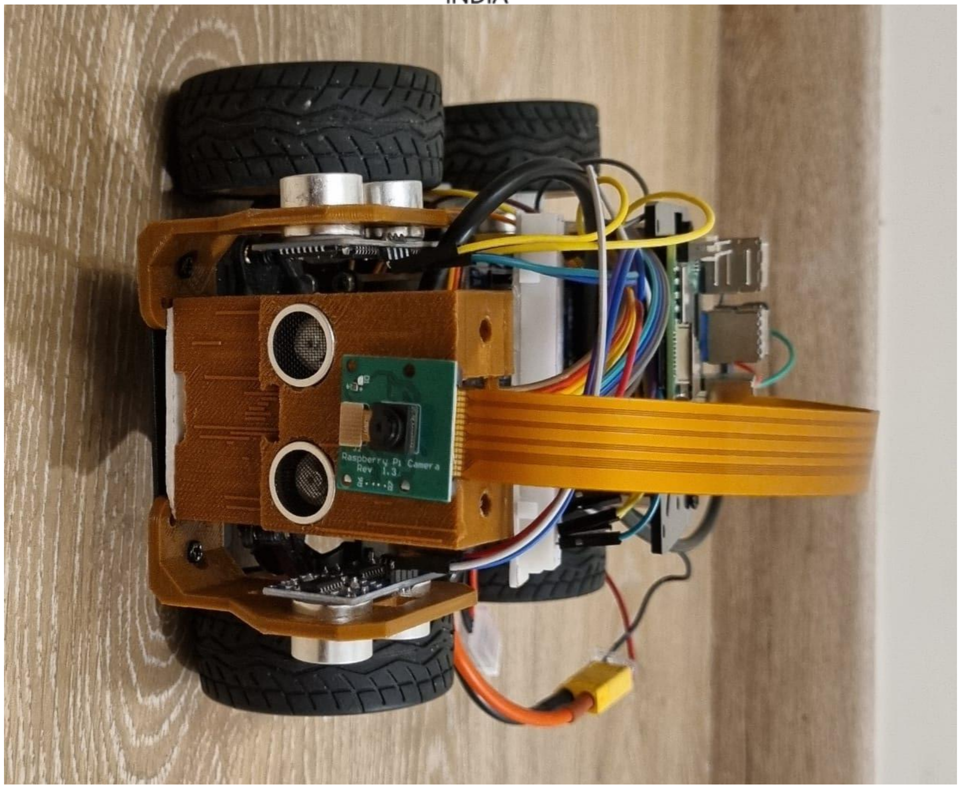
This pseudo code outlines a basic strategy for deciding how to avoid detected obstacles, incorporating safety checks before executing manoeuvres.

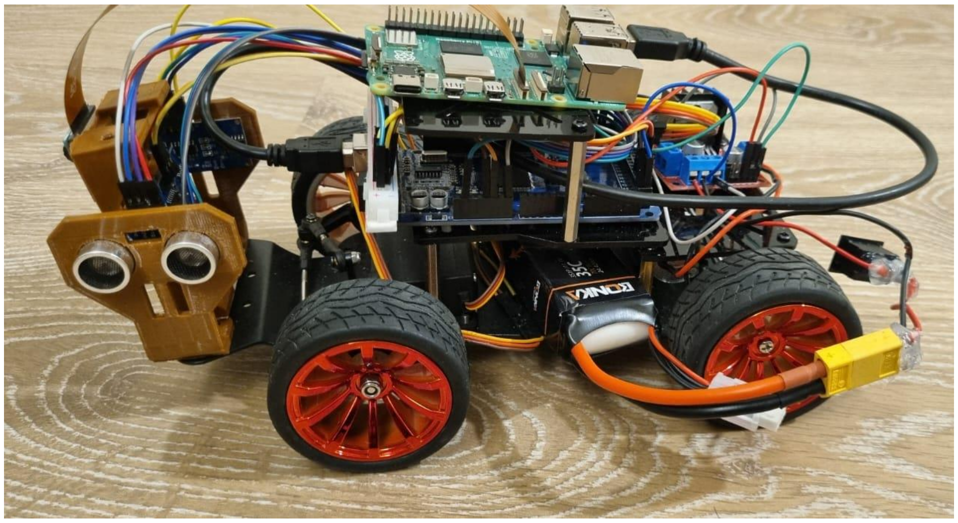
The combination of ultrasonic sensors and visual processing through the Raspberry Pi Camera Module V2 provides a robust obstacle detection and management system for a self-driving car. By leveraging both distance measurements and visual cues, the vehicle can make informed decisions to safely navigate through its environment. This strategy emphasizes the importance of sensor fusion and adaptive decision-making in the case of the self-driving car.

6. PICTURES OF VEHICLE









7. ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to our coach Mr. Purushotham N K from Pair AI, Mr. Prateek Baishkhiyar from Pair AI, our guide Dr. Kuheli Mondal, Gopalan National School, India STEM Foundation and WRO for giving us this wonderful learning opportunity in building this self-drive car.

We also like to thank Pair India for their constant technical support, guidance and encouragement throughout this project work. We would specifically like to thank Shri Rakesh Gupta of Pair AI for his constant moral support throughout this endeavor.