WRO 2023 - FUTURE ENGINEERS

Autonomous Buggies Team- Documentary

Ammar Daher
Salman Daher
Modar Ibrahim
Jaafar Mayya(Coach)

October 31, 2023

0.1 INTRODUCTION

This Engineering journal presents the development and design process of an autonomous vehicle created for participation in the WRO 2023 Future Engineers Category. The vehicle represents a fusion of various theoretical, mathematical, and mechanical engineering principles, combined with complex programming tasks and computer vision techniques. The primary objective of this project is to create a vehicle capable of autonomously navigating its environment while embodying the overarching theme of connecting the world.

Throughout the report, we will delve into the intricate aspects of this project, exploring the theoretical foundations that underpin its design. We will discuss the mathematical principles employed to enable precise control and navigation of the vehicle. Additionally, we will delve into the mechanical engineering aspects, highlighting the physical components and systems that contribute to its functionality.

Moreover, this project heavily relies on programming tasks and computer vision techniques. We will explore the algorithms and software implementations utilized to process visual information and make informed decisions in real-time. The integration of computer vision into the vehicle's functionality allows it to perceive and interpret its surroundings, facilitating safe and efficient navigation.

By combining these diverse disciplines, we aim to create an autonomous vehicle that not only showcases engineering prowess but also embodies the spirit of connecting the world. Through advanced technology and innovative design, we aspire to contribute to the evolution of autonomous vehicles and their potential to revolutionize transportation and connectivity on a global scale.

CHAPTER 1

Mobility Management

1.1 VEHICLE MAIN BODY

The vehicle main body is built using LEGO, building instructions are available via this link.The main Micrprocessor in the vehicle is Raspberry Pi 4 Model b provided with Raspberry pi Camera V2.1.Main Microcontroller is Atmega mounted on Arudino Mega 2560 board. To ensure the accurate and efficient control and direction of the vehicle's movement, we employed crucial components:

- Servo Motor Model
- J Sumo DC Motor
- BTS Motor Driver

The servo motor model was selected for controlling the vehicle's steering, while the BTS7960 motor driver is utilized to control the movement of J Sumo DC motor in order to provide the necessary power to drive the vehicle's motion.

1.2 MOTION MECHANISM

We used Direct-Acting steering and to achieve that a servo motor was installed in way that can move the steering mechanism freely, while the DC motor (J Sumo 1000 rpm) serves as the driving motor. The kinetic energy generated by the DC motor is transferred to the back wheels through a differential gearbox. The wheels receive this energy via a drive shafts of LEGO components, which transmits torque to the differential gearbox through a suitable coupler model.

1.3 THE STEERING MECHANISM

Our design incorporates front Direct-Acting Steering, a mechanism that eliminates the need for a steering gear by connecting the steering wheel directly to the steering linkage. Our front direct-acting steering method offers several advantages over conventional steering systems:

- **Enhanced Responsiveness:** The direct connection between the steering wheel and the steering linkage results in immediate and precise response to driver input, providing enhanced control and maneuverability.

- **Reduced Complexity:** Our steering method simplifies the overall design, reducing weight and complexity, and potentially lowering manufacturing and maintenance costs.
- **Lower Maintenance:** Requirements With fewer components involved, the direct-acting steering method can reduce the need for maintenance and potential points of failure, resulting in improved reliability and durability.

The working mechanism the direct-acting steering method can be summarized with:

- Steering Input Translation: When the servo motor turns the steering axle, the input is directly transmitted to the steering linkage, causing the front wheels to turn accordingly.
- Cornering: Cornering: During cornering, the steering mechanism maintains a smooth trajectory based on the servo motor input.

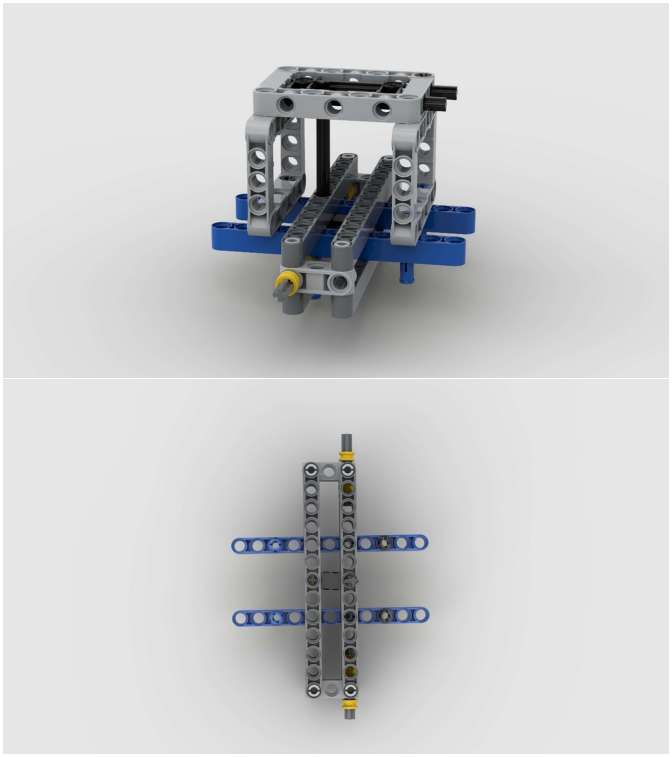


Figure 1.1: Steering Mechanism

When designing a steering mechanism, some parameters should be considered to measure the characteristics and specifications of the steering system:

1.3.1 Steering Ratio:

It is a key parameter in a steering system that describes the relationship between the rotation of the steering wheel (which is the servo axle in our case) and the resulting rotation of the wheels. Moreover, it quantifies the mechanical advantage or amplification provided by the steering mechanism.
Mathematically speaking, it is defined as the ratio of the angle turned by the steering wheel (servo axle) θ_{servo} to the angle turned by the wheels θ_{wheels}, in the direct acting steering θ_{wheels} is equal to

θ_{servo}. It can be expressed as [1]:

$$\theta_{servo} = \theta_{wheels} \Rightarrow \text{Steering Ratio} = \frac{\theta_{servo}}{\theta_{wheels}} = 1 \tag{1.1}$$

The steering ratio determines how much the wheels will turn in response to a given rotation of the servo. A higher steering ratio means that a smaller rotation of the steering wheel will result in a larger rotation of the wheels, providing a greater turning effect. Conversely, a lower steering ratio means that a larger rotation of the steering wheel is required to achieve the same wheel rotation. So, it can be described as a measure of the sensitivity and responsiveness of the steering system.

1.3.2 Steering Wheel Torque

Steering wheel torque T_{wheels} is the influence of servo motor force or moment applied on the steering. In other words, it refers to the force or moment applied to the wheels as a result of the torque output from the servo motor. T_{wheels} can be calculated by multiplying the applied force by servo motor F_{servo} by the effective lever arm or steering arm length L_{eff}, which is the distance between the point where the force is applied and the axis of rotation of the steering wheel [1]:

$$T_{wheel} = F_{servo} \times L_{eff} \tag{1.2}$$

When interpreting the steering wheel torque, it should be noted that a higher steering wheel torque indicates that more force is needed from the servo motor to turn the steering wheel, providing a greater resistance to steering. Conversely, a lower steering wheel torque implies that less force is required, resulting in lighter and easier steering.

1.3.3 Steering Wheel angle

The steering wheel angle refers to the angle of rotation of the servo motor's output shaft, which is directly connected to the steering mechanism of the robot. Furthermore, it represents the angular displacement of the servo motor's output shaft from its neutral or reference position. This angle determines the orientation of the robot's steering mechanism and thus influences the robot's movement and direction [1].

$$\theta_{wheel} = \theta_{motor} \times \text{Steering Ratio} \tag{1.3}$$

1.4 DIFFERENTIAL GEAR

Differential gear is a crucial component in our robot that allows the rear-wheels to rotate at different speeds at the same time while maintaining power distribution. The primary function of the differential gear is to enable the wheels on the same axle to rotate at different speeds when the robot is turning or when there is a difference in traction between the two wheels. It distributes torque from the engine to the wheels while allowing them to rotate at varying speeds [2]. Our differential gear consists of several components:

- Ring Gear: A large gear attached to the axle shaft that receives power from the engine or the driveshaft.
- Pinion Gear: A smaller gear connected to the drive shaft, which meshes with the ring gear.

- Side Gears: Two gears that connect the differential to the axle shafts.
- Spider Gears: These gears are positioned between the side gears and allow the wheels to rotate at different speeds.

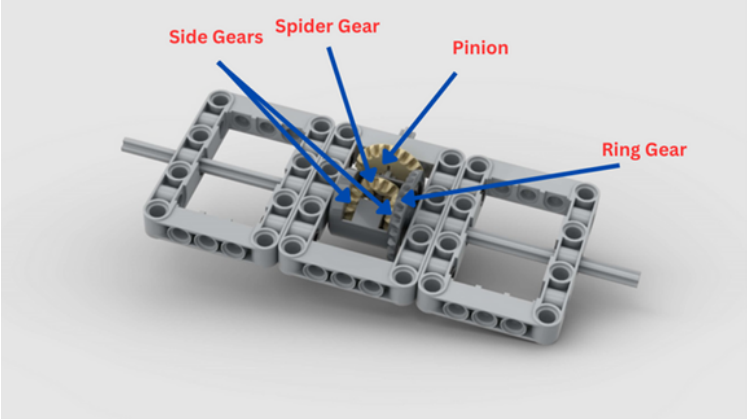


Figure 1.2: LEGO Differential Gear

When the vehicle is moving in a straight line, the differential distributes torque equally to both wheels. However, when the vehicle turns, the inside wheel (closer to the center of rotation) needs to rotate at a slower speed than the outside wheel to prevent slippage. The differential allows this speed difference by allowing the side gears to rotate at different speeds while maintaining power distribution.
It is important to determine gear ratio before designing a differential gear, which involves understanding the specifications and characteristics of the motors, as well as the mechanical setup of the differential. To calculate the differential gear ratio, the ratio between the ring gear and the side gears should be determined. This can be done by counting the number of teeth on each gear:

- The number of teeth on the ring gear (crown wheel). Let's call it R.
- The number of teeth on each side gear (sun gear). Let's call them $S1$ and $S2$.

The differential gear ratio can be calculated as the ratio between the teeth on the ring gear and the teeth on one side gear. Since there are two side gears in the differential, the gear ratio is multiplied by 2 [3]:

$$GearRatio = \frac{2 \times R}{S_1} = \frac{2 \times R}{S_2} \tag{1.4}$$

The gear ratio represents the rotational speed and torque distribution between the two side gears of the differential. Once the gear ratio calculation is done, motor speed can be related to the wheel speed. Since the side gears are connected to the wheels, their speed is directly proportional to the wheel speed. The no-load speed of the motor $N_{\text{MotorNoLoad}}$ from the motor's specifications, which will enable us to calculate the wheel speed using the formula[3]:

$$N_{\text{wheel}} = \frac{N_{\text{MotorNoLoad}}}{GearRatio} \tag{1.5}$$

The wheel speed represents the rotational speed of the LEGO differential's side gears, which directly affects the vehicle's speed. However, the calculated values may need to be calibrated when experimenting them (Real-world values may differ from the calculated ones due to multiple

constraints that can't be considered when doing the calculations). However, the results and objectives of our design showed no need for calibration.
The following diagram represents the steps of calculating N_{wheel} with respect to $N_{MotorNoLoad}$ and the *GearRatio* [4]:

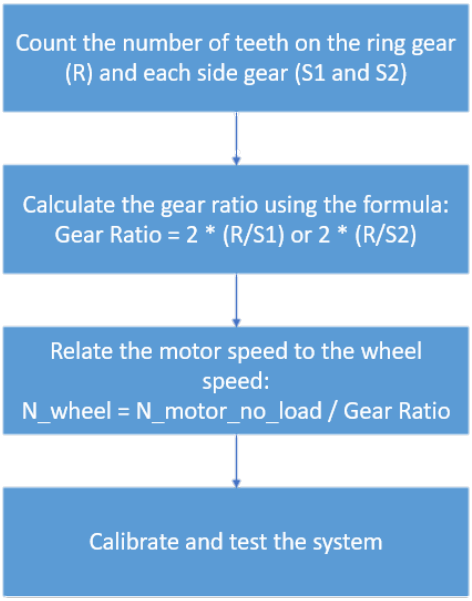


Figure 1.3: Calculation of N_{wheel} with Gear Ratio and $N_{motorNoLoad}$

1.5 MOTOR DRIVER

A motor driver is crucial for controlling motors as it enables the safe handling and delivery of the required power to drive the motor, while also providing protection features to prevent damage in abnormal operating conditions. Furthermore, it offers precise control, speed, direction, and acceleration, so a driver is essential for applications that require accurate positioning or motion control as in our case. It offers bidirectional control, compatibility with different motor types, and optimizes power usage for improved system efficiency. We tested two types of them in our robot (BTS7960 and L298N), a comprehensive explanation can be found in chapter 2.

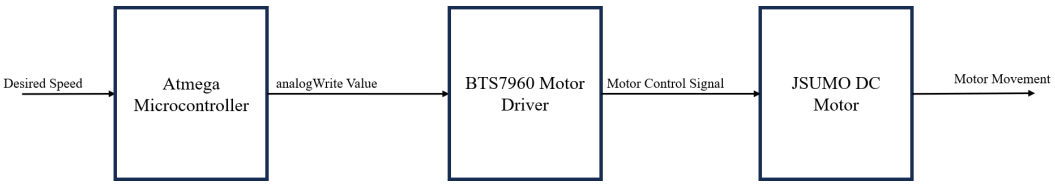


Figure 1.4: Basic-open loop control of motor speed

1.6 WHEELS

Lego wheels were used in the robot's design, their advantage comes from their efficiency and light weight. The table below clarifies the specifications of the wheels.

Lego Wheel Specifications	Values
Total diameter	56 mm (2.2")
Tire width	28 mm (1.1")
Rim diameter	43 mm (1.69")
Rim width	26 mm (1.02")
Weight	23g
Rim material	ABS (Acrylonitrile Butadiene Styrene)
Tire material	SEBS (Styrene-Ethylene Ethylene-Butylene-Styrene)

Table 1.1: Wheels specifications [5] [6]

Figure 1.5: Lego Wheel

1.7 LEGO AXLES

Using LEGO axles (4.8mm Diameter) in both differential gears mechanism and wheels connecting significantly improves the robot's efficiency. LEGO axles provide precise rotational motion, and minimize friction and energy losses. It is made of PA (Polyamide) which is a type of material that can withstand high loads and impacts. This means it's suitable for elements that need to be tough and able to interact with each other, ensuring smooth power transmission and prolonged operation, for example, gear wheels and different connectors [6].

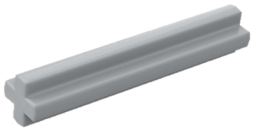


Figure 1.6: Lego Axle

1.8 COUPLER DESIGN

Power transmission plays a critical role in ensuring efficient and reliable movement. We utilized a coupler to transfer torque from a Jsumo DC motor to a LEGO axle and also another coupler to transmit torque from a servo motor to the steering LEGO axle. This solution combines the flexibility and customization of 3D manufacturing with the versatility and precision of LEGO components, resulting in a robust and seamless power transmission mechanism. The coupler acts as the mediator between motors and the LEGO axles, seamlessly connecting both components while maintaining torque transfer efficiency. The design of the coupler is carefully engineered to fit snugly onto the motor shaft and securely attach to the LEGO axle, ensuring optimal power transmission without slippage or loss. This design can be produced using 3D Printers or CNC Cutting Machine

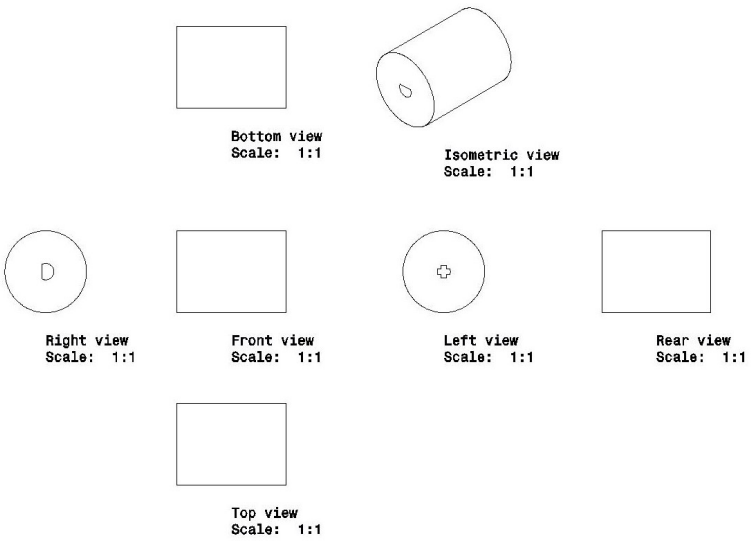


Figure 1.7: Coupler Design Drawings

CHAPTER 2

Power and Sense Management

2.1 ELECTRONIC COMPONENTS

Raspberry Pi 4 Model B: The Raspberry Pi 4 Model B is a popular single-board computer (SBC) developed by the Raspberry Pi Foundation. It is the fourth generation of the Raspberry Pi series and offers significant improvements over its predecessors. The Raspberry Pi 4 Model B is designed to be a versatile and affordable platform for various applications, including education, prototyping, home automation, media centers, and more. The key features and specifications of the Raspberry Pi 4 Model B [7]:

1. **Processor:** The board is powered by a Broadcom BCM2711 quad-core Cortex-A72 (ARMv8) 64-bit system-on-a-chip (SoC) running at 1.5 GHz. This processor provides a significant performance boost compared to previous Raspberry Pi models, enabling smoother multitasking and faster execution of applications.
2. **Memory:** The Raspberry Pi 4 Model B is available in different memory configurations, including 2GB, 4GB, and 8GB LPDDR4 RAM options. The increased memory capacity allows for better performance, especially when running resource-intensive applications or multitasking.
3. **GPU:** It features a VideoCore VI GPU, which supports OpenGL ES 3.x and 4Kp60 hardware decode of HEVC video. The GPU enables multimedia applications, gaming, and hardware-accelerated video playback.
4. **Connectivity:** The board offers improved connectivity options, including:
    - Gigabit Ethernet: Provides fast wired network connectivity.
    - Dual-band Wi-Fi: Supports 2.4 GHz and 5 GHz wireless networks, offering improved wireless performance.
    - Bluetooth 5.0: Enables wireless connectivity with Bluetooth-enabled devices.
    - USB Ports: The Raspberry Pi 4 Model B has two USB 2.0 ports and two USB 3.0.
5. **Video and Display:** The Raspberry Pi 4 Model B supports dual-monitor output with resolutions up to 4K. It features two micro HDMI ports, allowing you to connect two displays simultaneously. The board also has a MIPI DSI display port and a MIPI CSI camera port for connecting touchscreens and cameras.

6. **Storage:** It includes a microSD card slot for the operating system and data storage. Additionally, it has two USB 3.0 ports and two USB 2.0 ports, which can be used to connect external storage devices such as hard drives or USB flash drives.
7. **GPIO Pins:** The board maintains compatibility with previous Raspberry Pi models and provides 40 GPIO (General Purpose Input/Output) pins, which allow for the connection of various sensors, actuators, and other electronic components.
8. **Operating System:** The Raspberry Pi 4 Model B is compatible with a wide range of operating systems, including Linux distributions such as Raspbian (now known as Raspberry Pi OS), Ubuntu, and others. This flexibility enables developers to choose the OS that best suits their needs and leverage the extensive software ecosystem available for the Raspberry Pi.

Figure 2.1: RaspberryPi 4

Raspberry pi camera:
The Raspberry pi camera was used to discover the playfield of the robot and to detect the color of the pillars existed in front of the vehicle.

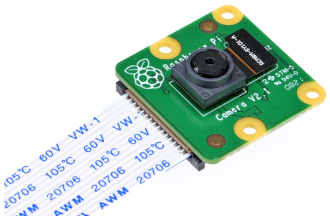


Figure 2.2: Enter Caption

In our robot, Raspberry Pi is connected with Raspberry Pi camera and Arduino. It is used to process the frames of real time video recording. A python code using Open Computer Vision (OpenCV) library is developed in order to detect the pillars with its colors (red and green) and send the image processing results to the Arduino board via Serial Connection. A detailed explanation of the algorithm in chapter 3.

Arduino Mega 2560: The Arduino Mega 2560 is an open-source microcontroller board based on the ATmega2560 microcontroller. It is one of the most popular boards in the Arduino family due to its versatility and extensive I/O capabilities. The Mega 2560 is designed to offer a large number of digital and analog inputs and outputs, making it suitable for complex projects that require multiple sensors, actuators, and communication interfaces. The key features and specifications of the Arduino Mega 2560 [8]:

1. Microcontroller: The Mega 2560 is powered by the ATmega2560 microcontroller, which has 256KB of flash memory for storing code, 8KB of SRAM for data storage, and 4KB of EEPROM for non-volatile storage.
2. Digital I/O Pins: The board offers 54 digital I/O pins, of which 15 can be used as PWM outputs. These pins can be used for connecting various components such as sensors, switches, LEDs, and other digital devices.
3. Analog Inputs: The Mega 2560 has 16 analog inputs, allowing you to read analog signals from sensors and other devices. These inputs have a 10-bit resolution, providing 1024 different voltage levels.
4. Communication Interfaces: The board supports multiple communication interfaces, including UART (Universal Asynchronous Receiver/Transmitter), SPI (Serial Peripheral Interface), and I2C (Inter-Integrated Circuit). These interfaces enable communication with other devices such as sensors, displays, and modules.
5. USB Interface: The Mega 2560 features a USB interface that allows it to be connected to a computer for programming and serial communication. It can appear as a virtual serial port for easy integration with software development environments.
6. Operating Voltage: The board operates at 5V, making it compatible with a wide range of sensors, modules, and other components.
7. Compatibility: The Mega 2560 is fully compatible with the Arduino software and programming language, making it easy to get started with coding and prototyping.
8. Shield Compatibility: The board is designed to be compatible with Arduino shields, which are additional modules that can be stacked on top of the board to extend its functionality. This allows you to easily add features such as Wi-Fi, Bluetooth, motor control, and more.

In our project, We used serial communication to send data from Raspberry Pi to Arduino and vise versa. Arduino is used to manage the mobility of the robot by finding the appropriate path depending on the sensor readings (Ultra Sonic and Gyroscope) alongside with the image processing results received from Raspberry Pi.

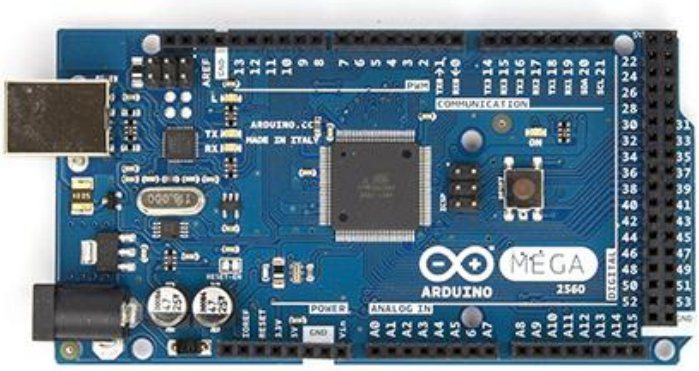


Figure 2.3: Arduino Mega2560

BTS7960 Motor Driver:
The BTS7960 Motor Driver is a popular dual H-bridge motor driver module that allows to control the speed and direction of DC motors. It is commonly used in robotics, automation, and other projects that require precise motor control. The key features and specifications of the BTS7960 Motor Driver [9]:

- H-Bridge Configuration: The BTS7960 Motor Driver utilizes an H-bridge configuration, which consists of four power MOSFETs (Metal-Oxide-Semiconductor Field-Effect Transistors) arranged in pairs. This configuration enables bidirectional control of the motor, allowing to control its rotation in forward and reverse directions.
- Motor Compatibility: The motor driver module is designed to work with DC motors and can handle a wide range of motor voltages, typically between 5V and 27V. The maximum continuous current per channel is typically around 43A, making it suitable for driving high-power motors.
- Current Sensing: The BTS7960 Motor Driver features built-in current sensing capabilities. It has two current sense pins that allow you to monitor the current flowing through each motor channel. This feature can be useful for various purposes, such as implementing motor current protection or feedback control.
- PWM Control: The motor driver supports Pulse Width Modulation (PWM) control for speed regulation. By varying the duty cycle of the PWM signal, you can adjust the motor's speed precisely. The module accepts a PWM input signal from a microcontroller or any other PWM source.

- Overcurrent and Overtemperature Protection: The BTS7960 Motor Driver incorporates protection mechanisms to safeguard both the driver and the motor. It includes built-in overcurrent and overtemperature protection circuits that can help prevent damage to the motor driver module and the connected motor.
- Input and Output Connections: The motor driver module typically has separate input pins for controlling the motor direction and speed. These control pins can be connected to the digital output pins of a microcontroller or other control devices. The motor connections are made to the driver's output terminals, allowing you to connect the DC motor.
- Heat Dissipation: Due to its high current handling capacity, the BTS7960 motor driver may generate heat during operation. To dissipate heat effectively, the module often includes a heat sink or a mounting hole for attaching an external heat sink.
- Compatibility: The BTS7960 Motor Driver can be used with various microcontrollers, such as Arduino, Raspberry Pi, or any other microcontroller capable of generating the necessary control signals for the driver module.

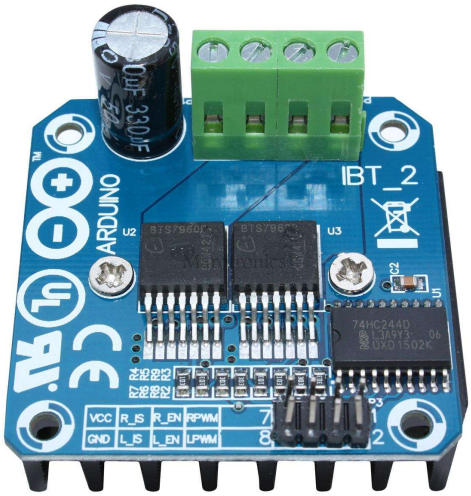


Figure 2.4: BTS7960 Motor Driver

L298N Motor Driver:

The L298N Motor Driver is another widely used motor driver module that provides control over the speed and direction of DC motors. It also utilizes an H-bridge configuration, consisting of four power transistors arranged in pairs, enabling bidirectional control of the motor's rotation [10].

- H-Bridge Configuration: The L298N utilizes a dual H-bridge motor driver that allows speed and direction control of two DC motors at the same time, this configuration contains four switching elements, transistors or MOSFETs, with the motor at the center. It also has two switches that can be used to control the motor rotation in either direction. Moreover, the input pins of the L298N are used for controlling the rotation direction of the motor A and B, and by controlling these pins, the switches of the H-Bridge inside the L298N IC are controlled.
- Motor Compatibility: The L298N Motor Driver is designed to work with DC motors and can handle a wide range of motor voltages, it is suitable for driving medium-power motors.

- Current Sensing: The L298N Motor Driver does not have built-in current sensing capabilities, but external current sensing circuitry can be implemented separately if current monitoring is required.
- PWM Control: The L298N Motor Driver supports PWM control for regulating the motor's speed. By adjusting the duty cycle of the PWM signal, the motor's speed can be precisely controlled.
- Overcurrent Protection: The L298N Motor Driver incorporates overcurrent protection mechanisms to prevent damage to the driver and the connected motor. However, it does not typically include built-in overtemperature protection.
- Input and Output Connections: The motor driver module has separate input pins for controlling the motor direction and speed. These control pins can be connected to the digital output pins of a microcontroller or other control devices, while the motor connections are made to the driver's output terminals.
- Heat Dissipation: The L298N Motor Driver may generate heat during operation due to its current handling capacity. It often includes a heat sink or a mounting hole for attaching an external heat sink to dissipate heat effectively.
- Compatibility: The L298N Motor Driver is compatible with various microcontrollers, including Arduino, Raspberry Pi, and other microcontrollers capable of generating the necessary control signals for the driver module.

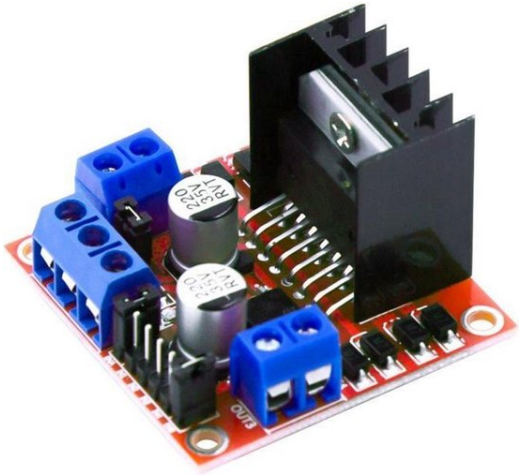


Figure 2.5: L298N Motor Driver

Comparison between L298N and BTS7960:

We tested both circuits and evaluated their performance on the robot. However, BTS7960 Motor Driver showed noticably better results, on the other hand, L298N didn't meet the requirements of our robot which resulted in overheated circuit and not enough current input current to the JSumo DC Motor. Here are the key differences between both motor drivers:

- Current Handling: The BTS7960 Motor Driver has a higher current handling capacity compared to the L298N Motor Driver, making it more suitable for driving high-power motors (J-Sumo Motor in our case).
- Current Sensing: The BTS7960 Motor Driver includes built-in current sensing pins, allowing for easy monitoring of motor current. On the other hand, the L298N Motor Driver does not have this feature, requiring external circuitry for current sensing.
- Overtemperature Protection: The BTS7960 Motor Driver incorporates overtemperature protection, which helps safeguard the driver and the motor. The L298N Motor Driver does not typically include this protection mechanism.
- Voltage Range: The L298N Motor Driver has a wider voltage range compared to the BTS7960.

RioRand LM2596 step-down DC-DC Converter:
The RioRand LM2596 is a popular step-down DC-DC converter module based on the LM2596 switching regulator integrated circuit. It is commonly used to convert higher DC voltages to lower, more stable voltages for various electronic applications. The LM2596 module offers efficient and adjustable voltage regulation, making it widely used in projects requiring reliable power supply solutions. In our robot, we used RioRand to provide the required input voltage and ampere to the Raspberry Pi. Separately, we used a RioRand to provide the power supply for the sensors and micro servo motor taking into account the common ground principle [11].

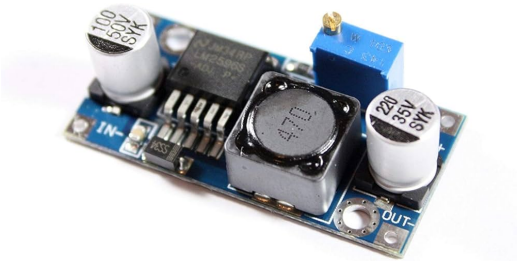


Figure 2.6: RioRand LM2596 step-down DC-DC Converter

Servo Motor: [12]

1. Torque: 1.6kgcm
2. Stall current 650±80mA
3. Voltage 4.8 V - 7.2 V
4. Speed 0.12 sec / 60 degrees (4.8V)
5. Gears Nylon
6. Length 22 mm
7. Width 11,5 mm
8. Height 22.5 mm Weight 16 g

This servo motor is mounted on the steering mechanism in order to provide the required torque discussed in chapter (1) to move the steering in a precise angle.

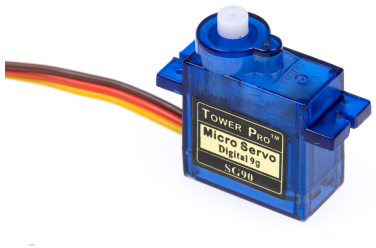


Figure 2.7: SG-90 Servo Motor

2 Li-Po RC Batteries 12V: [13] Lithium Polymer (Li-Po) batteries are a type of rechargeable battery that are widely used due to their high energy density, lightweight, and ability to deliver high currents. It provides a good balance between capacity and weight.

Battery Parameter	Value
Model	ZOP Power 11.1V 2200mAh 30C
Capacity	2200mAh
Size	28x34x116mm
Plug Style	XT60 Plug
Weight	140g

Table 2.1: Li-Poly Battery key features

Figure 2.8: Lipo Battery

J Sumo DC Motor:[14]

1. The diameter of the motor is 37mm. Length is 75mm.
2. 6mm diameter shaft section, shaft length 17 mm.
3. 6 M3 screw holes on the front side are available.
4. No load current is 240ma, and stall current is 5.9 Ampere.
5. All Gearhead gears are metal.
6. Motor's stall torque is 7.5 kg-cm.

Figure 2.9: Jsumo DC Motor

MPU 6050:[15] The MPU-6050 is a popular and widely used integrated circuit (IC) that combines a 3-axis gyroscope and a 3-axis accelerometer in a single package. It is commonly used in various applications such as motion sensing, orientation tracking, gesture recognition, and stabilization control systems.
The key features and specifications of the MPU-6050:

1. 3-Axis Gyroscope: The MPU-6050 includes a 3-axis gyroscope, which measures angular velocity around three axes (X, Y, and Z). It provides accurate motion sensing capabilities and can be used to detect rotational movements.
2. 3-Axis Accelerometer: The MPU-6050 also incorporates a 3-axis accelerometer, which measures linear acceleration along the three axes (X, Y, and Z). It enables the detection of changes in velocity and the determination of tilt or inclination.
3. Digital Motion Processing: The MPU-6050 features an onboard digital motion processor (DMP) that offloads motion processing tasks from the main microcontroller. The DMP performs sensor fusion, combining data from the gyroscope and accelerometer to provide accurate and reliable motion tracking information.

4. I2C Interface: The MPU-6050 communicates with the microcontroller or other devices using the I2C (Inter-Integrated Circuit) protocol. It has an I2C interface that allows for easy integration and control with various microcontrollers and development boards.

5. Motion Detection Interrupts: The MPU-6050 includes built-in motion detection capabilities and supports programmable interrupts. This allows the IC to generate interrupts to the microcontroller when predefined motion or orientation thresholds are met, enabling efficient and responsive operation.

6. Low Power Consumption: The MPU-6050 is designed to operate with low power consumption, making it suitable for battery-powered applications and devices with power constraints.

7. Applications: The MPU-6050 is widely used in applications such as robotics, drones, gaming controllers, virtual reality (VR) and augmented reality (AR) devices, inertial navigation systems, motion-controlled user interfaces, and many other projects where motion sensing and orientation tracking are required. [9]

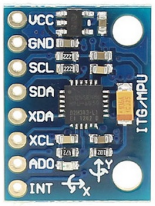


Figure 2.10: MPU-6050 Sensor

Cooling Fan: The cooling fan is used to cool down the temperature of the Raspberry Pi microprocessor.

Figure 2.11: Cooling Fan

UltraSonic Sensors: Ultrasonic sensors are electronic devices that use ultrasonic waves to measure distance or detect objects. They work based on the principle of echolocation, similar to how bats navigate in the dark. Ultrasonic sensors emit high-frequency sound waves (typically above the range of human hearing) and then receive the reflected waves to determine the distance or presence of objects [16].
The key characteristics and applications of ultrasonic sensors:

1. Distance Measurement: Ultrasonic sensors are commonly used for non-contact distance measurement. They typically emit ultrasonic pulses and measure the time it takes for the sound waves to bounce back after hitting an object. By knowing the speed of sound in the medium (usually air), the sensor can calculate the distance to the object.
2. Object Detection: Ultrasonic sensors can detect the presence or absence of objects within their range. When an object is detected, the sensor receives the reflected sound waves, and based on the intensity or time delay of the received signal, it can determine if an object is present or not.
3. Range and Accuracy: Ultrasonic sensors can have varying range capabilities, typically ranging from a few centimeters to several meters. The accuracy of distance measurement depends on factors such as the sensor's resolution, beam angle, and environmental conditions.
4. Ultrasonic Transducer: The core component of an ultrasonic sensor is the transducer, which converts electrical energy into ultrasonic sound waves and vice versa. It consists of a piezoelectric crystal that vibrates when an electrical current is applied, generating the ultrasonic waves.
5. Beam Pattern: Ultrasonic sensors emit sound waves in a specific beam pattern, usually conical or cylindrical. The beam angle determines the width of the detection area. Narrower beam angles provide more focused sensing, while wider angles cover a larger area but with reduced accuracy.
6. Environmental Considerations: Ultrasonic sensors are affected by environmental factors such as temperature, humidity, and air turbulence. These factors can influence the speed of sound and affect the accuracy of distance measurements.
7. Applications: Ultrasonic sensors are widely used in various fields, including industrial automation, robotics, automotive, security systems, level monitoring, parking assistance, liquid flow measurement, and presence detection in consumer electronics.

The 10 Sensor were mounted as the following:

- **L1 Sensor** Left Front (Closer to the frontend of the robot).
- **L2 Sensor** Left Back (Closer to the backend of the robot).
- **R1 Sensor** Right Front (Closer to the Frontend of the robot).
- **R2 Sensor** Right Back (Closer to the backend of the robot).
- **F Sensor** Frontal Sensor (In the middle of the frontend of the -obot).
- **B Sensor** Rear Sensor (In the middle of the backend of the robot).
- **BL Sensor** Back Left Sensor - mounted on the edge of the vehicle and tilted by 45 counter-clockwise degrees about the x-axis of the robot.
- **BR Sensor** Back Right Sensor - mounted on the edge of the vehicle and tilted by 45 clockwise degrees about the x-axis of the robot.

- **FL Sensor** Front Left Sensor - mounted on the edge of the vehicle and tilted by -45 counterclockwise degrees about the x-axis of the robot.
- **FR Sensor** Front Right Sensor - mounted on the edge of the vehicle and tilted by -45 clockwise degrees about the x-axis of the robot.

Figure 2.12: Ultrasonic Sensor

2.2 OVERALL CIRCUIT CURRENT USAGE

Accurate knowledge of the electrical component requirements is a crucial factor to be considered. Both theoritical and experimental information should be gathered to investigate the properties of the electrical circuit, this can be reduced to the following reasons:

- **Optimal Power Supply Design:** Understanding the current needs of the robot's electrical components allows for the design of an appropriate power supply system, this helps in selecting a power source that can provide the required currents at the correct voltage levels so that the system's stability and performance can be ensured.
- **Battery Life Optimization:** Since Lipo Batteries were used in the robot, knowing the current needs of the components is crucial for optimizing battery life.In results, Estimating accurately the current draw during different operational states, power-saving strategies can be implemented, such as allocating convenient batteries to the needed components. Additionally, this extends the robot's operating time and reduces the frequency of battery replacements or recharging.
- **Heat Dissipation and Cooling:** Some electrical components may suffer from overheating due to its continous high current draw, so accurate knowledge of the overall circuit consumption aids in determining the heat generated by the components, which in sequence helps in implementing effective cooling mechanisms to prevent overheating, which can lead to performance degradation or even component failure. For example, we noticed that Raspberry Pi heat's increases noticably after long time of working, so we added a cooling fan in addition to heat sink wich helped avoiding overheating problem.
- **Safety Considerations:** Understanding the overall circuit consumption is surely essential from a safety standpoint. By ensuring that the power supply system can handle the maximum expected current draw, the risk of electrical failures, such as voltage drops or circuit damage, can be mitigated, which also prevents potential hazards to both the robot and its environment.
- **Cost Optimization:** When the electrical needs are known for each component, costs can be reduced without compromising performance or reliability by selecting components that meet the necessary specifications.

The properties of the robot's circuit can be understood in the table and figure below:

- **UltraSonic Sensor:** The input current for an Ultrasonic Sensor can vary depending on the specific model and operating conditions. Typically, it ranges from a few milliamperes (mA) to around 15 mA. For more information check here.
- **MPU6050 Gyroscope Sensor:** The input current for an MPU6050 Gyroscope Sensor is about 36 mA. For more information check here.
- **Raspberry Pi Model B 4:** The input current for a Raspberry Pi Model B 4 can vary depending on the connected peripherals and the workload. Typically, it requires around 2.5 A, but it is recommended in our case to use a power supply capable of providing 3A for stable operation. For more information check here.

- **Arduino Mega:** The input current for an Arduino Mega board is typically around 50mA when idle. However, the current consumption can increase when using additional peripherals or when running more complex programs as the case in our robot.
  For more information check here and here.
- **Servo Motor SG Tower Pro:** The input current for a Servo Motor can vary depending on the load and the specific model. Typically, it ranges from 0.5A – 2A, depending on the torque and speed requirements.
  For more information check here.
- **Raspberry Pi Camera:** The input current for a Raspberry Pi Camera Module can vary depending on the model and the camera settings. Typically, it requires around 250-300mA.
- **BTS7960 Motor Driver:** The input current for a BTS7960 Motor Driver depends on the motor being driven and the load conditions. It can range from a few milliamperes (mA) to several amperes (A), depending on JSumo motor's state.

Component	Input Current (mA)
Ultrasonic Sensor	15
MPU6050 Gyroscope	3.6
Raspberry Pi Model B 4	3000
Arduino Mega	500
Servo Motor SG Tower Pro	500-2000
Raspberry Pi Camera	250-300

Table 2.2: Caption

2.3 CIRCUIT GROUND

A common ground in a circuit is essential for maintaining accurate voltage measurements, ensuring reliable signal transmission, and mitigating signal noise and interference. It provides a consistent voltage reference for all components, simplifies circuit design and troubleshooting, and aids in maintaining signal integrity throughout the circuit. By establishing a shared reference point, the common ground enables effective communication between components and promotes optimal circuit operation [17].

2.4 POWER SUPPLY

We used 2 Li-Poly RC Batteries (12V – 2200mAh) as a source of power. The first one supplies the Arduino Board, Sensors, Servo motor, and DC motor; the second one supplies the Raspberry pi microprocessor. The robot is power supplied by two batteries as we mentioned before. The distribution of power is done by Three RioRand step-down DC-DC Converter that take their input from the batteries directly (12V input) and convert this input to (5V output). The first one supplies the raspberry pi microprocessor and its cooling fan. The second gives supply to a "bridge" which in turn supply the sensors and the gyroscope. The third supplies the servo motor. The DC motor is supplied via BTS motor driver that's supplied directly from the batteries (12V).

2.5 OVERALL SCHEME

Due to the large number of electronic components used in our circuit we used a special power supply distribution using basic electronic components. The following figure illustrates the electrical design of the circuit:

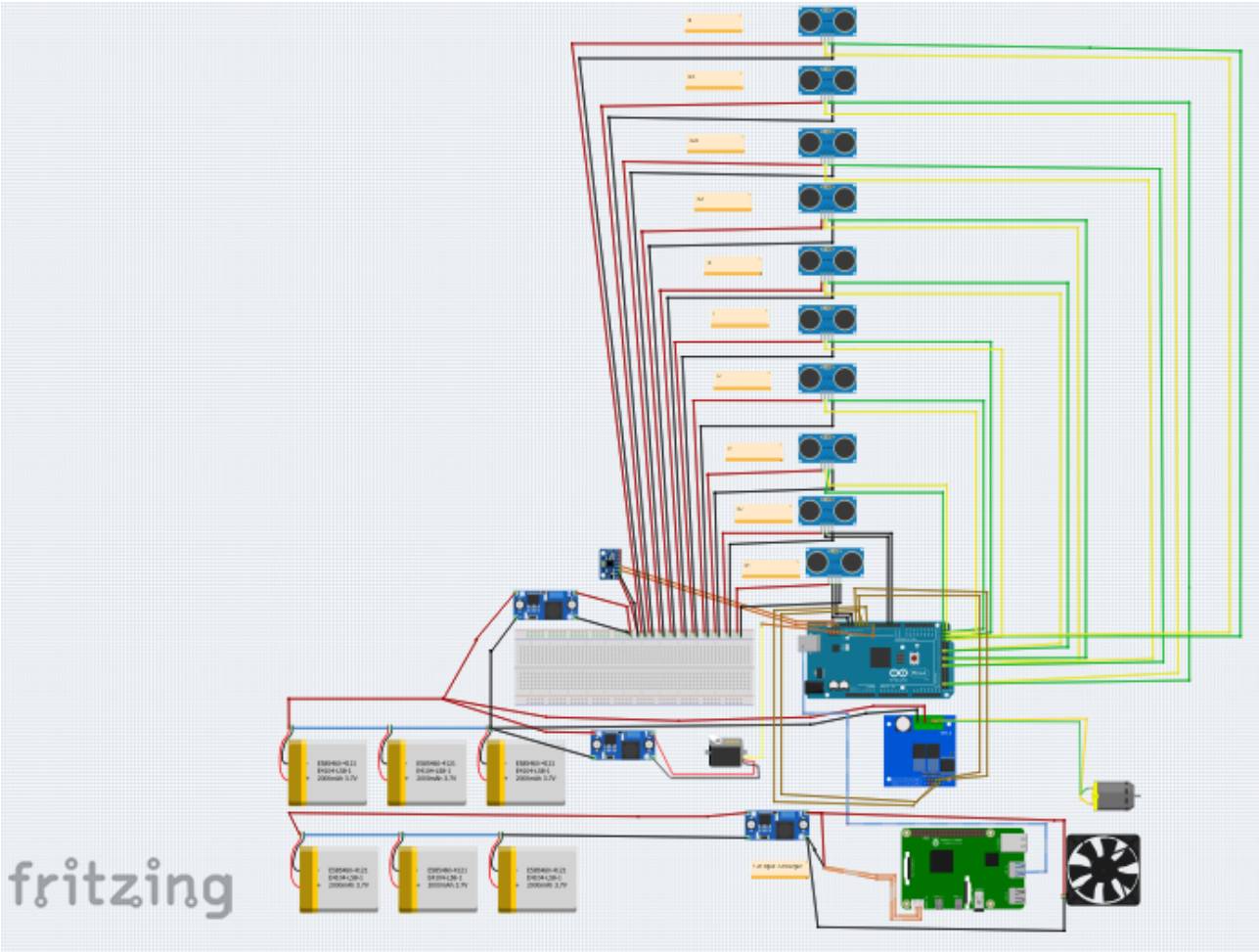


Figure 2.13: Vehicle electronic - electrical scheme

Note: Due to the limited types of components in Fritzing software, we were unable to find the battery we used so we designed an alternative circuit for it as shown in the previous figure.

2.6 WIRING DIAGRAM

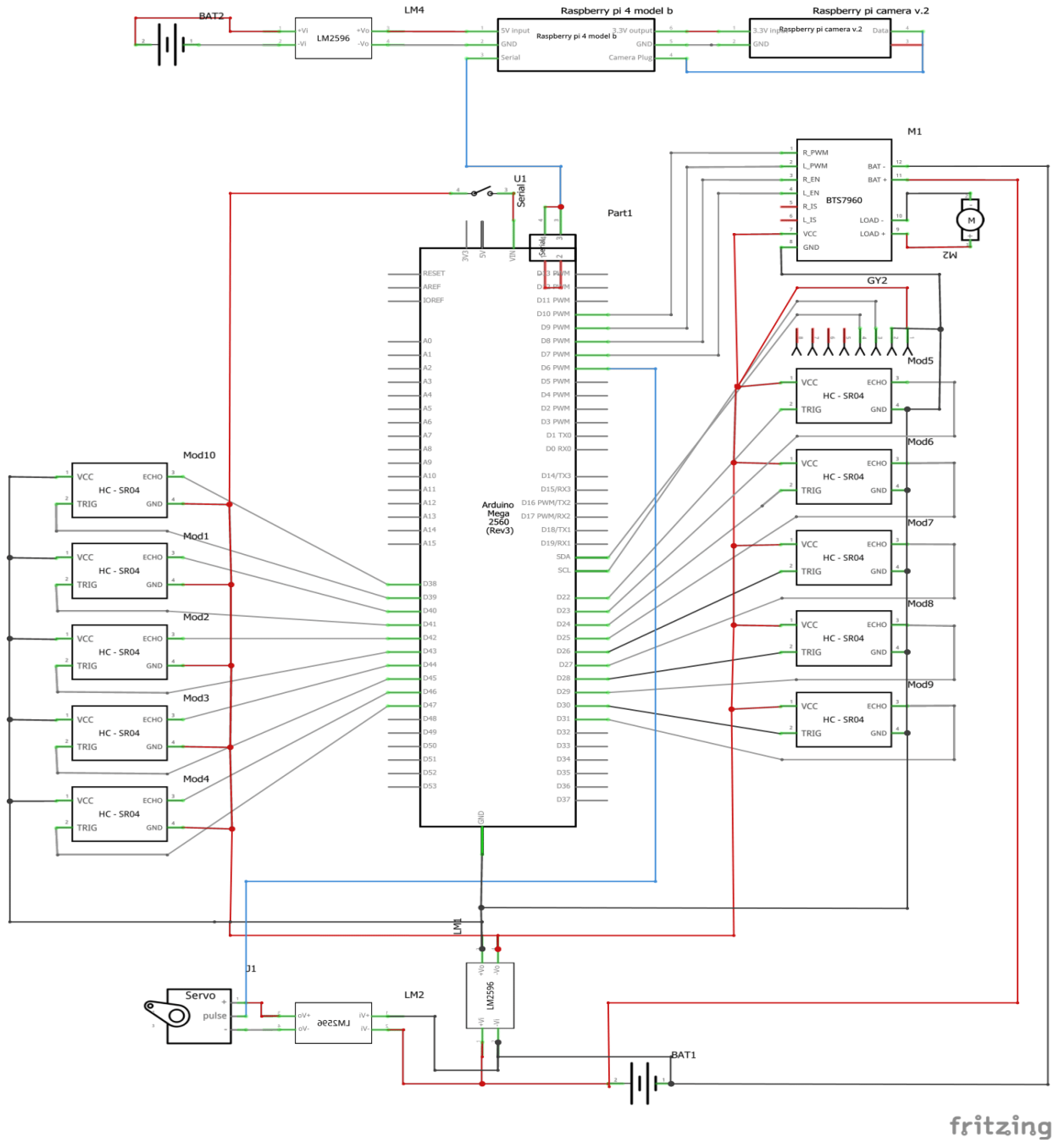


Figure 2.14: Vehicle electronics: Wiring Diagram

2.7 BOM (BILL OF MATERIALS)

Label	Part Type	Properties
BAT1	12 battery	package battery ; Li-Po Battery
BAT2	12 battery	package battery ; Li-Po Battery
GY2	MPU-6050 Board GY-521	pins 8
J1	Basic Servo Motor	SG-Tower pro
LM1	RioRand LM2596	DC-DC Converter - 2A
LM2	RioRand LM2596	DC-DC Converter - 2A
LM4	RioRand LM2596	DC-DC Converter - 5A
M1	BTS7960	pins 12
M2	DC Motor	Jsumo Motor
Mod1	HC-SR04 Ultrasonic Distance Sensor	chip LM324
Mod2	HC-SR04 Ultrasonic Distance Sensor	chip LM324
Mod3	HC-SR04 Ultrasonic Distance Sensor	chip LM324
Mod4	HC-SR04 Ultrasonic Distance Sensor	chip LM324
Mod5	HC-SR04 Ultrasonic Distance Sensor	chip LM324
Mod6	HC-SR04 Ultrasonic Distance Sensor	chip LM324
Mod7	HC-SR04 Ultrasonic Distance Sensor	chip LM324
Mod8	HC-SR04 Ultrasonic Distance Sensor	chip LM324
Mod9	HC-SR04 Ultrasonic Distance Sensor	chip LM324
Mod10	HC-SR04 Ultrasonic Distance Sensor	chip LM324
Part1	Arduino Mega 2560 (Rev3)	type Arduino MEGA 2560 (Rev3)
Raspberry pi 4 model b	Raspberry pi 4 model B	
Raspberry pi camera v.2	Raspberry pi camera v.2	
U1	SWITCH-SPST-2	package $r_s w_t h$; *variantra*

Table 2.3: List of used items (Assembly List)

Amount	Part Type	Properties
2	12 battery	package battery ; Li-Po Battery
1	MPU-6050 Board GY-521	pins 8
1	Basic Servo Motor	SG-Tower pro
2	RioRand LM2596	DC-DC Converter - 2A
1	RioRand LM2596	DC-DC Converter - 5A
1	BTS7960	pins 12
1	DC Motor	Jsumo Motor
10	HC-SR04 Ultrasonic Distance Sensor	chip LM324
1	Arduino Mega 2560 (Rev3)	type Arduino MEGA 2560 (Rev3)
1	Raspberry pi 4 model B	
1	Raspberry pi camera v.2	
1	SWITCH-SPST-2	package $r_s w_t h$; $variantra$

Table 2.4: List of shopping items

2.8 SENSE MANAGEMENT

In our vehicle, we incorporated sensors to enhance its functionality and improve safety. Sensors are crucial components that allow us to gather data about the vehicle's surroundings and its own internal state. By utilizing sensors, we can automate certain tasks, make informed decisions, and ensure smooth operation of the vehicle.

One of the sensors we employed is the ultrasonic sensor, which is used to calculate distances. The ultrasonic sensor emits high-frequency sound waves and measures the time it takes for these waves to bounce back after hitting an object. By knowing the speed of sound, we can mathematically calculate the distance to the object using the formula:

$$Distance = (SpeedofSoundTime)/2 \tag{2.1}$$

Here, the speed of sound is a known constant, and the time represents the round-trip time of the ultrasonic wave. By accurately measuring the distance to objects in the vehicle's vicinity, we can detect obstacles, avoid collisions, and enable safe maneuvering.

Additionally, we integrated a gyroscope sensor to determine the vehicle's yaw or rotation rate. The gyroscope measures the angular velocity of the vehicle around its vertical axis. Mathematically, the gyroscope provides us with the rate of change of the yaw angle over time. By integrating these angular velocity measurements, we can obtain the yaw angle itself. This information is valuable for maintaining stability, controlling vehicle orientation, and aiding in navigation.

To ensure accurate and reliable measurements from these sensors, we implemented a mean filter. The mean filter is a signal processing technique that helps reduce noise in the sensor measurements. It works by taking multiple samples of the sensor data over a period of time and calculating their average. This averaging process smooths out random fluctuations or outliers in the sensor readings, resulting in more reliable and consistent measurements.

CHAPTER 3

Obstacle Management

Obstacle management is a crucial challenge that should be driven by robust algorithms, which offers precision without the loss of the speed. We developed our algorithms based on the previous requirements, and thus achieved noticeably reliable performance.

3.1 COMPUTER VISION

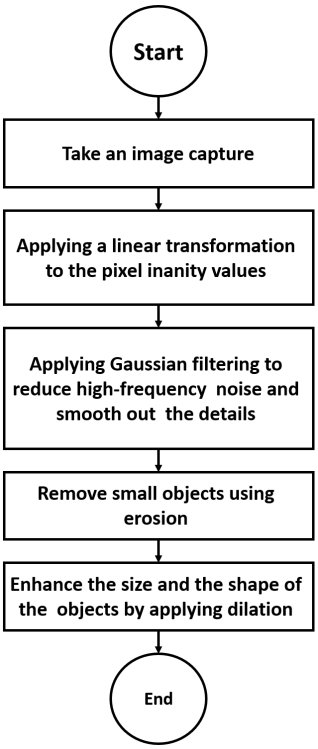
The integration of computer vision in latest techniques is considered an essential technology in robotics and automation. This can be seen especially in self-driving cars, where enabling the vehicle to see is essential, which means that it can perceive and understand the environment. By employing CV (computer vision) techniques, the robot in our task can gather valuable information from its surrounding, make informed decisions, and perform the required tasks. In this context, the isolation of red and green pillars is critical to prevent collisions and ensure the robot's path remains obstacle-free. However, since the computer vision function is only identifying red and green pillars of same size, the complexity can be reduced to color detection problem without the need for object detection. After determining the required CV algorithm, among the various color detection approaches available, such as HSL (Hue, Saturation, Lightness), HSB (Hue, Saturation, Brightness), RGB (Red, Green, Blue), and HSV (Hue, Saturation, Value), we have chosen to utilize the HSV color space, because in comparison to these color spaces [18]:

- The HSV color space offers distinct advantages over other color spaces for our specific requirements. HSV separates color information from both brightness and saturation. This separation allows for better adaptation to different lighting conditions, making the HSV color space more robust and reliable for color detection tasks.
- While RGB is widely used and suitable for many applications, it presents challenges in color-based segmentation and sensitivity to lighting variations. The complex color manipulation in RGB makes it less suitable for our specific needs. On the other hand, the HSV color space simplifies the extraction and separation of specific colors from images, making it ideal for efficient color-based segmentation.

3.2 IMAGE PROCESSING

3.2.1 Pre-Processing:

Before isolating the pillars to extract information, it was necessary to apply pre-processing on the captured images to improve the overall effectiveness of subsequent isolation. Pre-processing helps to eliminate unwanted artifacts and enhance relevant image features leading to more accurate and reliable segmentation results. It allows to prepare the image data in a way that maximizes the CV algorithm's performance, ensuring robust color detection and accurate boundary delineation. Our pre-processing operations contained applying a linear transformation to the pixel intensity values of the images. We used alpha and beta parameters, the alpha parameter controls the contrast adjustment, determining how much the pixel values are scaled. A larger alpha value increases the contrast, making the dark regions darker and the bright regions brighter. Additionally, the beta parameter represents the brightness adjustment, determining the amount of brightness added to the image. By adjusting these parameters, they can be used to modify the image's appearance, enhancing or reducing the contrast and brightness levels. Moreover, we applied Gaussian filtering. By averaging the pixel values in the neighborhood, with more emphasis on the nearby pixels according to their proximity to the center. it effectively reduced high-frequency noise and smoothed out details. We also applied erosion to shrink the boundaries of objects in an image in order to remove small details and smoothen the image. Finally, Dilatation was applied to expand the boundaries of objects to enhance the size and shape of objects.



3.2.2 Masking:

1. Convert the input image from the RGB color space to the HSV color space.
2. Define the lower and upper thresholds for the desired color range in terms of hue, saturation, and value values.
3. Iterate through each pixel in the HSV image and check if it falls within the specified color range.
4. For pixels within the range, set the corresponding pixel in the mask image to white (255); otherwise, set it to black (0).
5. The resulting mask image will have white pixels representing the desired color range and black pixels representing all other colors.

HSV values in our work were as follows: (([0, 100, 50] to [0, 255, 255] and [163, 100, 54] to [179, 255, 255] for red due to the cylindrical shape of the hue values) - ([43, 50, 45] to [43, 50, 45] for green).

3.2.3 Image Thresholding:

Thresholding is a technique used to convert a grayscale or single-channel image into a binary image, where pixel values are classified into two categories based on a specified threshold value. The general syntax for using OpenCV library thresholding:

```
ret, threshold = cv2.threshold(src, thresh, maxval, type)
```

- **src**: The source image, which should be a single-channel (grayscale) image.
- **thresh**: The threshold value used for classifying pixel values.
- **maxval**: The maximum value assigned to pixels that exceed the threshold.
- **type**: The thresholding type, which determines the specific algorithm used for thresholding.

Return values:

1. **ret**: The threshold value that was used (sometimes automatically calculated).
2. **threshold**: The resulting binary image after applying the thresholding operation.

The `cv2.threshold` function compares each pixel value in the source image with the threshold value. Pixels with values greater than the threshold are assigned the `maxval` value, while pixels with values below the threshold are set to 0 or a lower value (depending on the thresholding type).

3.2.4 Contouring:

Contouring in OpenCV and computer vision refers to the process of detecting and representing the boundaries of objects or shapes in an image. Contours are widely used in computer vision for various tasks, such as object detection, shape analysis, motion tracking, and image segmentation.

What pillar is the closest pillar to the vehicle? The decision of what pillar is closest to the vehicle is made by many selection processes as following and for each contour in the image:

1. Finding the moments (specifically the centroid) of the contour.
2. Finding the bounding rectangle about the contour.
3. Calculating the area of the contour using cv2.contourArea function.
4. Calculating the mean area of the two previous contours(Since the first contour is a rectangle, it may not give an accurate area with some orientations of the camera. Additionally, the second contour may contain noise and not thus don't give the correct area)
5. Taking the largest area contour as the selected one. These steps are applied on the red and green masks. It should be noted that the orientation of the used camera (Raspberry pi camera) maintain a range of view that allow to detect more than one pillar, more specifically, when the robot is in the corner section, the pillars in the end of the next section can be seen on the camera, but without objects from outside (We cropped the images to ensure that no outer objects are evident in the image).

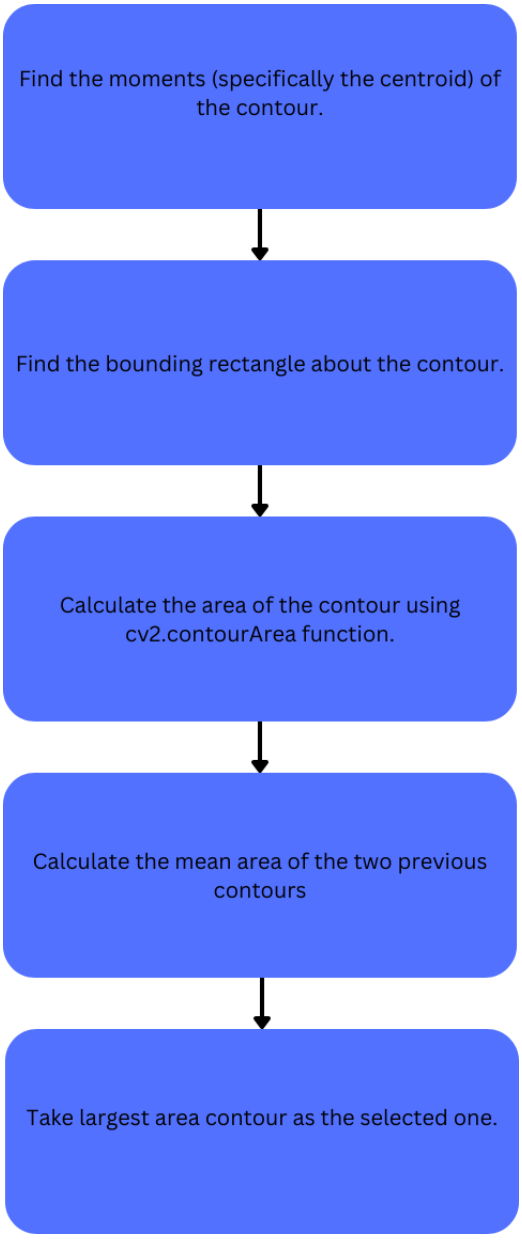


Figure 3.1: Contouring Process (Diagram)

What is the distance between the pillar and the vehicle? After finding the largest contour we can find the distance between the vehicle and the pillar using a mathematical expression found by a curve fitting model that was built on experimental data of the contour area and the actual distance. The data can be found in Appendix.

$$Distance = \alpha \times Area^{\beta} \tag{3.1}$$

3.3 HOW IS CV USED IN THE ROBOT?

Two algorithms were developed and tested for CV processing, One algorithm ”Persistent Vision” operates continuously, processing incoming images in real-time and promptly relaying the results. The other algorithm ”Event Driven Vision”, on the other hand, remains dormant until it receives a start signal, and sends the results once only. **Persistent Vision:** The Persistent Vision algorithm continuously processes camera input images and performs a series of pre-processing operations detailed in Pre-Processing. It utilizes the HSV color space to isolate colors and identifies the region of interest, which corresponds to the red and green pillars. Subsequently, it calculates the area of all contours present in the image and selects the closest one. Based on the measured distance, the algorithm determines the state of the pillar. Once the closest pillar to the vehicle is identified, the Raspberry Pi sends a signal composed of a single character via a serial connection to the microcontroller. This signal consists of one of the following letters that define the state of the pillar: ”G,” ”R,” ”g,” ”r,” or ”N”. These letters indicate the presence of a close green pillar (distance $<$50cm), close red pillar (distance $<$50cm), far green pillar (distance $>$50cm), far red pillar (distance $>$50cm), or no green or red pillar, respectively.

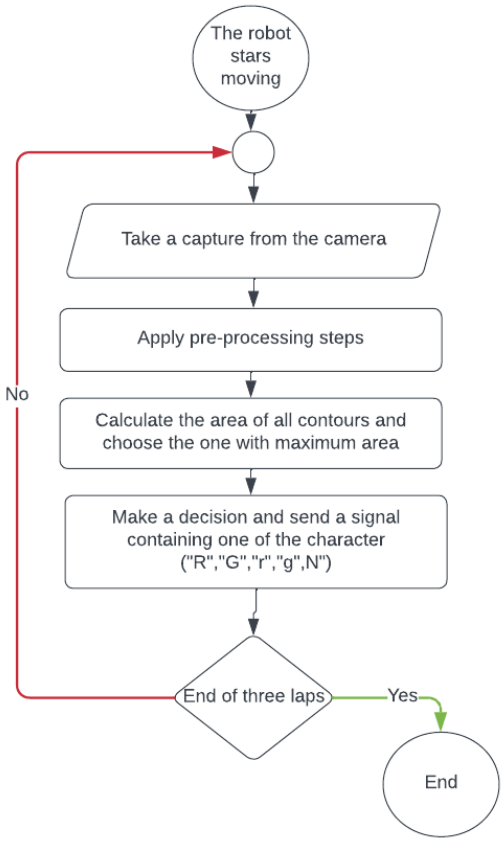


Figure 3.2: Persistent Vision Algorithm (Diagram)

Event Driven Vision: In the Event Driven Vision algorithm, processing occurs once a signal is received from the Arduino. The Arduino sends this signal at the beginning of each section, after the robot has stopped and assumed the correct position. The algorithm follows the same steps as before but with some modifications. It takes all pillars contours that are present in the scene (camera input images) and applies the following procedures. We will refer to the section the robot is entering as the ”current section” and the following section as the ”next section”:

1. Count the number of pillars within the maximum range present in the image. Pillars located beyond the end of the current section are disregarded. This comes from the fact that they do not impact the robot’s movement decision in the current section.
2. Once the number of pillars is determined, their information is stored in an array that includes the area, X and Y coordinates of the contour centroid, color, and distance for each contour.
3. There are three cases to consider:
   - If there are three stored pillars, the array is sorted based on the area. This situation arises when there are two pillars in the current section and one pillar is present at the start of the next section. In a such case, the HSV color isolation may fail

to ignore the farthest pillar, leading to inaccurate distance calculation and the inclusion of unwanted pillars. The contour centroids are used to identify the pillars within the borders of the current section. This is based on the understanding that when the moving direction (CW or CCW) is known, the x-coordinates of contours centroids can be used to determine which pillar is unwanted.

- If there are only two stored pillars, the array is sorted directly based on the centroids. The presence of a pillar at the beginning of the current section can be easily determined through distance, there are two subcases to consider. In the first subcase (Existence of a pillar at the start of the current section), the other pillar is taken into account. In the contrary subcase, where it is impossible for two pillars to exist in the same section when neither is at the beginning of the current section, only the closest pillar is considered.
- The simplest case involves the detection of only one pillar. In this scenario, the decision is made directly based on the color alone.

After the completion of processing, only one signal is sent to arduino and image capturing with processing will stop, it will wait until a signal from the arduino arrives again. The raspberry pi signal contains one of the strings (“R”,”G”,”GG”,”RR”,”RG”,”GR”).

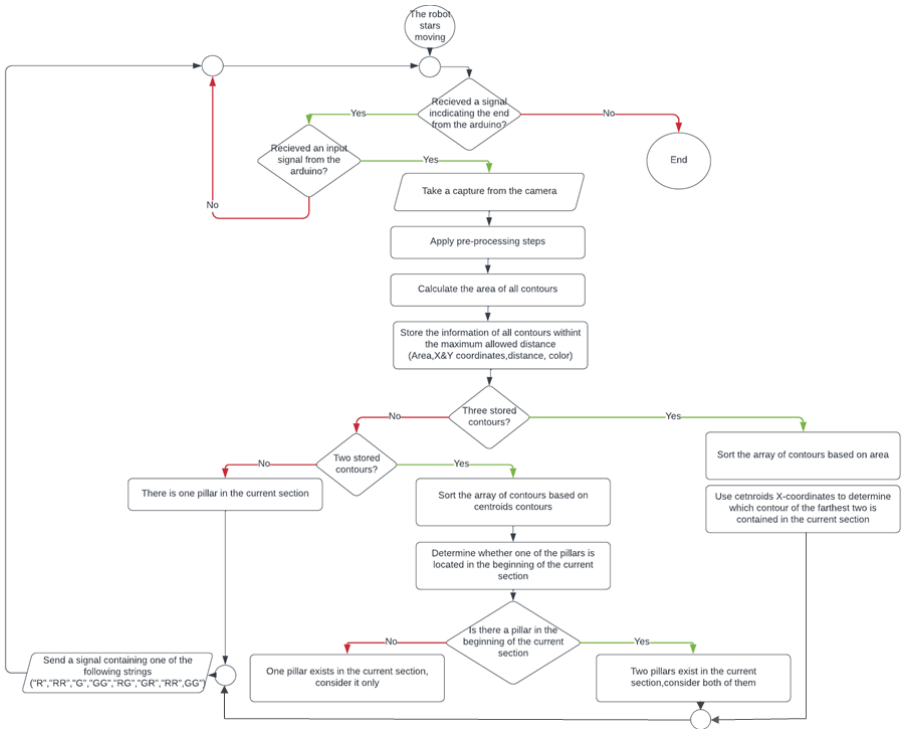


Figure 3.3: Event Driven Vision (Diagram)

Some cases are visually explained below:

- In this case, two green pillars are located in the current section, so they will both be considered:

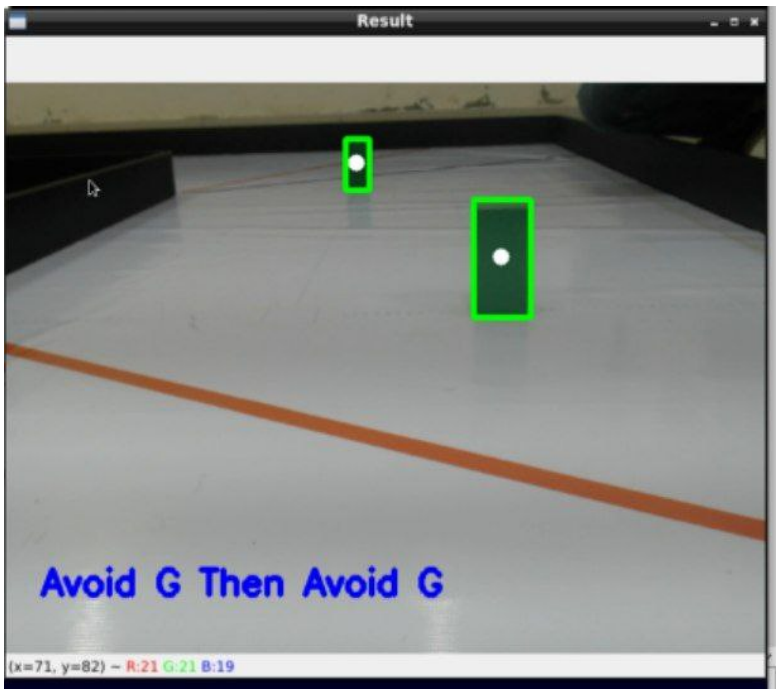


Figure 3.4: Event Driven Vision

– In this case, two red pillars are located in the current section, so they will both be considered:

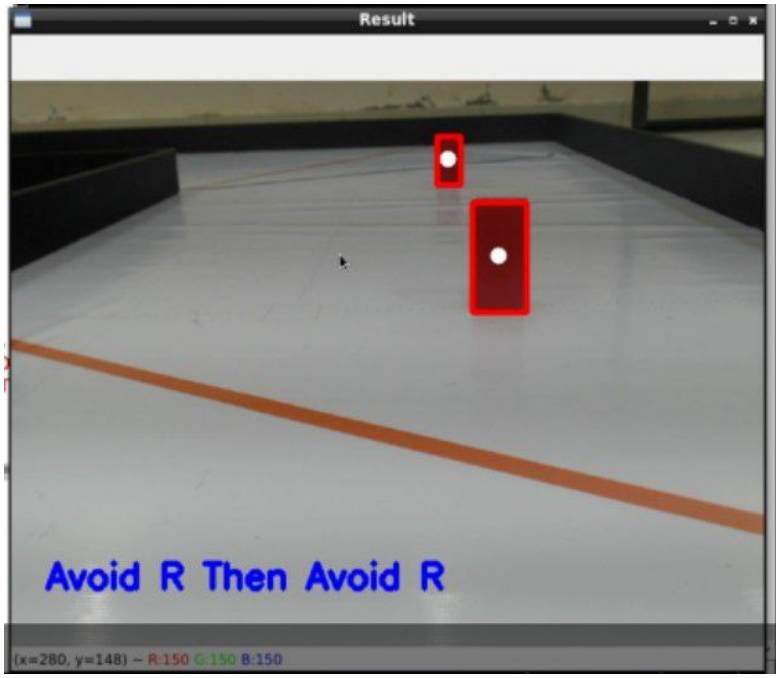


Figure 3.5: Event Driven Vision

– In this case, only one green pillar is located in the current section:

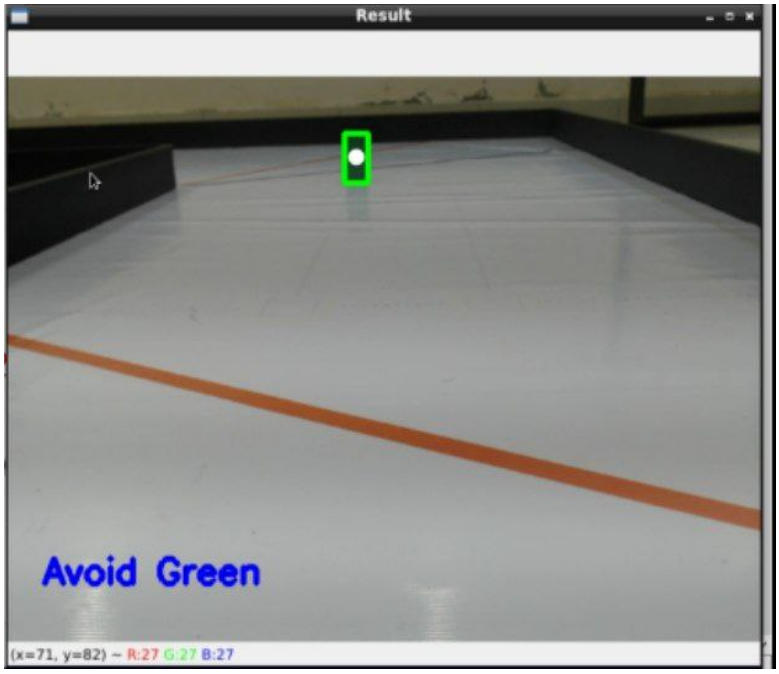


Figure 3.6: Event Driven Vision

– In this case, one green pillar is located in the current section, and the other is located in the next section, so only the first one will be considered:

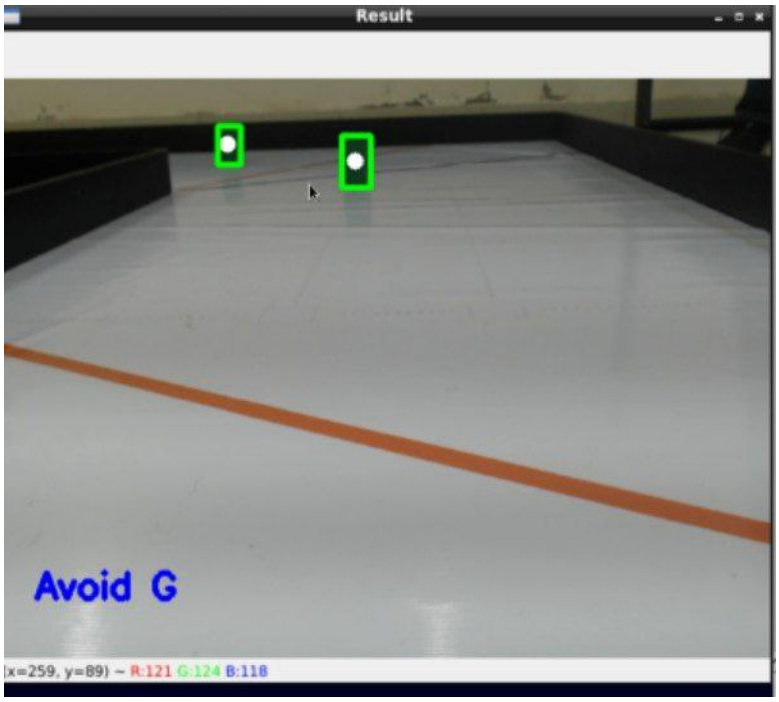


Figure 3.7: Enter Caption

– In this case, a red pillar and a green pillar are present in the current section, and one green pillar is in the next section, which won't be considered:

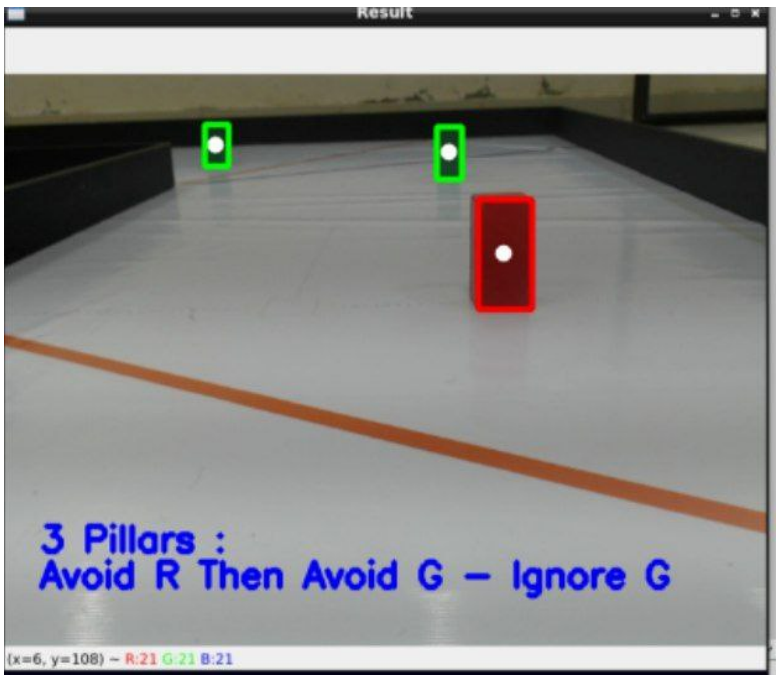


Figure 3.8: Enter Caption

– In this case, a red pillar exist in the current section, and one green pillar is in the next section, which won't be considered:

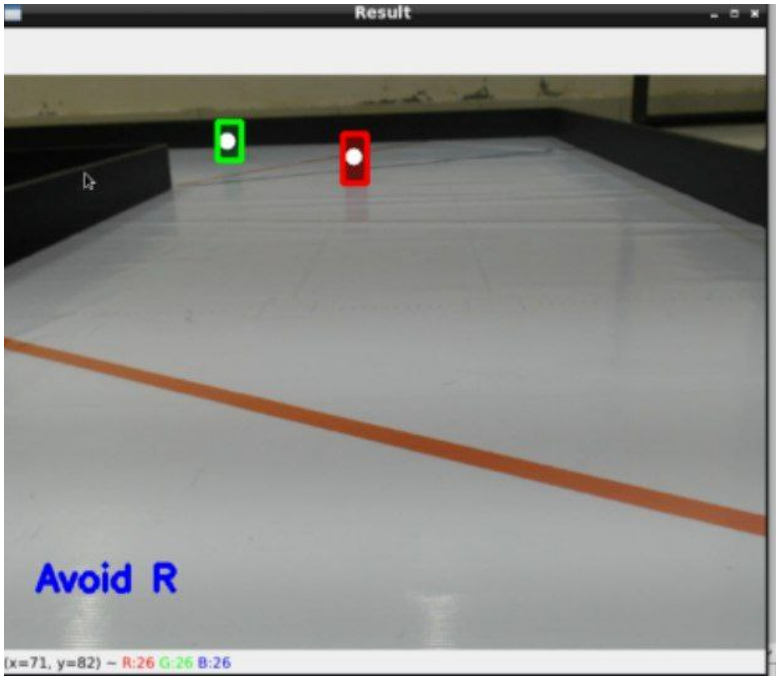


Figure 3.9: Enter Caption

3.4 OPEN CHALLENGE ALGORITHM

In this challenge, the required task is completing three laps in the least possible time, this requires constructing an algorithm that guarantees precise navigation in addition to fast and smooth movement between the sections. Our main idea is built upon partitioning the section for two areas, this approach enables us to address and solve each problem independently. The first area, referred to ”Protected Area”, contains the majority of the robot’s path inside of it (Green area) as illustrated in the figure below. The second area, which is ”Unprotected Area”, is the unsafe area in which the robot becomes very close to the wall as illustrated in the figure below (Red area).

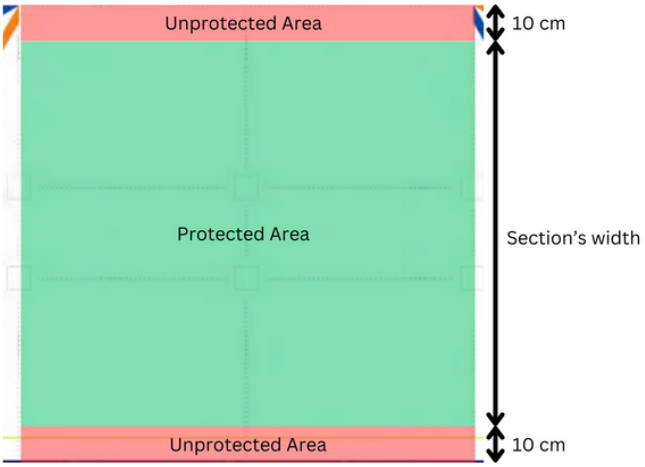


Figure 3.10: Areas Division

3.4.1 Moving in Protected Area

In the Protected area, the robot can move safely without the risk of hitting the wall, building upon this fact, it is enough for the robot to keep moving in a straight line and parallel to the wall (no high risk of hitting the wall). PD controller is used to keep the robot's angle constant, the algorithm takes the reading of the gyroscope as input, and setpoint is taken after the completion of the turn. Equation below represents the formula for calculating the error and the needed angle for the servo to turn:

$$PIDOutput = K_{p(protected)} \times Error_t + \frac{Error_t - Error_{t-1}}{IntervalLength} \times k_{d(Protected)} \tag{3.2}$$

$$Error_t = GyroscopeReading - SetPoint \tag{3.3}$$

However, relying only on the gyroscope sensor arises inaccurate setpoint calculation sometimes. This can be seen after the completion of the robot's turn, the robot may not turn exact 90 degrees, it can be less or more. Therefore, a correction formula is utilized to correct gyroscope's reading and calculate the correct setpoint:

$$Angle = tan^{-1}(\frac{D_{S1} - D_{S2}}{D_{S1S2}}) \tag{3.4}$$

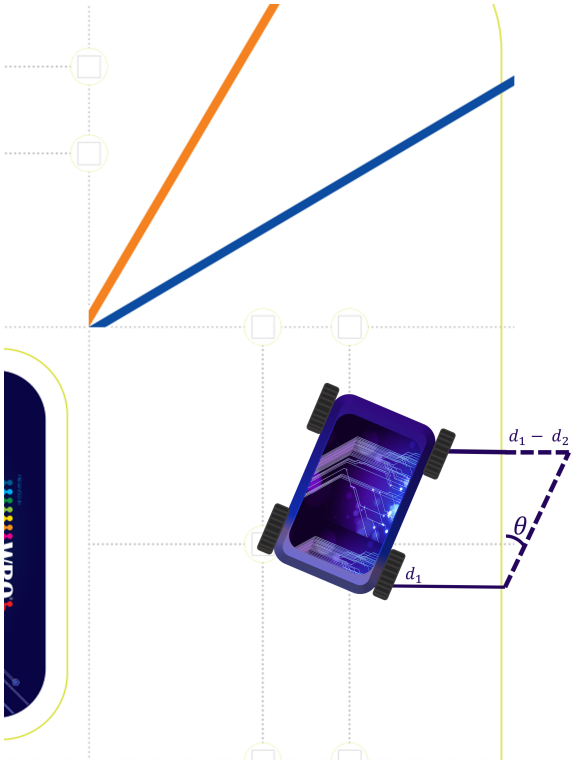


Figure 3.11: Angle Calculation

Final output can be calculated using the formula below:

$$ServoAngle = Angle + PIDOutput \tag{3.5}$$

Where:

- $PIDOutput$ is the calculated servo angle using PD only.
- $SetPoint$ is the reference angle (in which the robot is parallel to the wall).
- $Error_t$ is the difference between the current reading and $SetPoint$ in time t.
- K_p is the proportional gain.
- K_d is the derivative gain.
- $IntervalLength$ is the time period to take one measurement.
- D_{S1} is the front right/left sensor reading.
- D_{S2} is the rear right/left sensor reading.
- D_{S1S2} is the distance between the front right/left sensor and the rear right/left sensor.
- $Angle$ is the angle between the robot and the wall.
- $ServoAngle$ is the final servo angle.

3.4.2 Moving in Unprotected Area

When entering the unprotected area, the movement is unsafe inside this area. The robot will have to immediately move away from the unprotected area until it reaches the protected area. We used P controller (just to move away from the wall) with a relatively high proportional gain:

$$ServoAngle = K_{p(unprotected)} \times (D_{vertical} - D_{unprotected}) \tag{3.6}$$

$$D_{vertical} = cos(Angle) \times \frac{D_{S1} + D_{S2}}{S2} \tag{3.7}$$

where:

- $ServoAngle$ is the turning angle of servo motor.
- $K_{p(unprotected)}$ is the proportional gain in unprotected area.
- $D_{vertical}$ is the vertical distance between the robot and the wall.
- $D_{unprotected}$ is the unprotected area width.

The vertical distance was chosen to inlcude in the formula because it measures the accurate position of the robot relative to the wall, this yield more accureate results compared to the measured distance alone by one sensor or the average of the two. The distance can be calculated using simple geometry.

3.4.3 Detecting turns and direction

Once the robot detects a significant distance $> 100cm$, The robot identifies a turn. However, the robot can be not parallel to the wall in a way that results in wrong ultrasonic readings, as a consequence of the small this rein this case, the robot detects the front wall using front ultrasonic sensor, in case it is impossible to turn safely, the robot move backward until it reaches a point where the turn is safe. The robot will determine the direction of movement (CW/CCW) in the first turn, this can be done by determining on which side of the robot the turn was detected.

3.5 OBSTACLE CHALLENGE ALGORITHM

We mentioned previously that we have developed two algorithms, one that continuously process the scene and send information ”Persistent Vision”, and one that work only when it receives a signal from the arduino to start processing and send the information once ”Event Driven Vision”. Based on the chosen algorithm, the robot follows a unique way to manage its movement and avoid red and green obstacles, we’ll first discuss the general approach and mathematical formulas that drives the robot movements when avoiding the obstacles, then we’ll discuss in depth each algorithm separately.

3.5.1 General Approach and Mathematical Formulas

This section presents the general approach and mathematical formulas utilized in the algorithm design for the robot’s movement and obstacle avoidance. The primary objective of our approach is to ensure the robot moves in a manner that guarantees a safe path without collisions. So During the algorithm’s design stage, our main motivations were focused on achieving robustness, adaptability, and fault tolerance.

Robustness and Adaptability

- The algorithm is designed to function effectively regardless of the robot’s relative position to obstacles.
- It takes into account various scenarios where the robot may be situated in proximity to obstacles from different angles or distances.
- By considering relative position independence, the algorithm ensures consistent obstacle avoidance irrespective of the robot’s location.

Fault Tolerance

- It is essential to address possible faults in the robot’s movements that can affect the obstacle avoidance and ensure uninterrupted obstacle avoidance.

Those considerations led us to a mathematical formula, that calculates the needed angle to turn, we considered three cases:

- Normal Case

The robot receives a signal from Raspberry Pi indicating that the robot should turn to avoid the pillar, the robot will calculate the desired angle using the formula:

$$DesiredAngle = abs(tan^{-1}(2 \times \frac{D_{wall}}{D_{obstacle}})) \tag{3.8}$$

Where:

- $D_{obstacle}$ is the distance between the robot and the obstacle.
- D_{wall} is the distance between the robot and the wall.
- *DesiredAngle* is the angle that the robot should turn to avoid colliding with the wall.

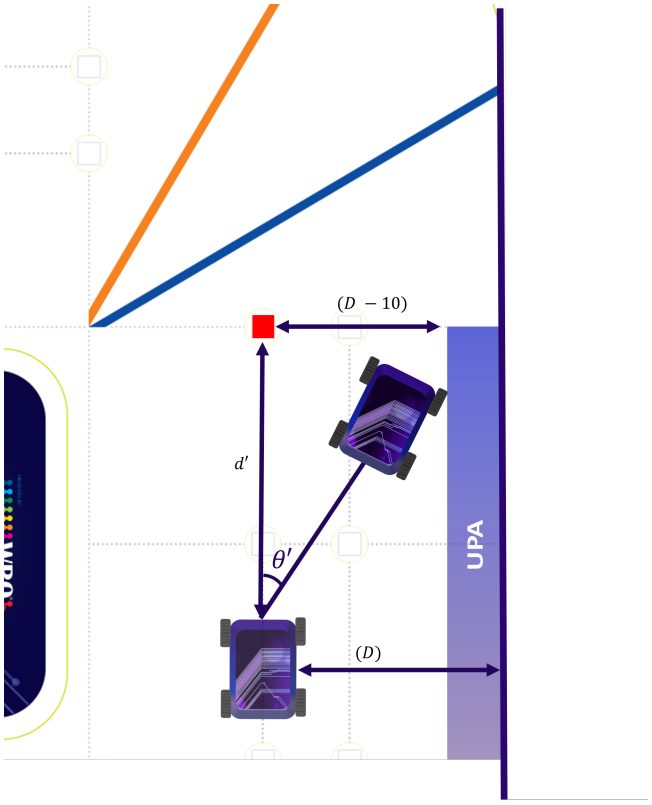


Figure 3.12: Desired Angle Calculation

- Safe Case

The robot receives a signal from Raspberry Pi indicating that the robot should turn to avoid the pillar, the robot will calculate the desired angle using previous formula, if it is less than 11° degrees, this indicates that the robot is within the limits of the safe range (where it can continue its movement in a straight line without the need to turn), it sets the angle to zero, which means that the robot won't turn.

- Extreme Angle Case

The robot receives a signal from Raspberry Pi indicating that the robot should turn to avoid the pillar, the robot will also calculate the desired angle using previous formula, if it is higher than a certain value 64° degrees, this indicates that the robot is dangerously far from the wall in a way that makes it have to turn a big angle. We set an upper limit to fix this angle, this ensures that the robot will turn safely.

After achieving the desired angle, the robot will move straight along the angle that was turned, using a PD controller to maintain a steady path until it reaches a specified distance from the wall. This distance is not the vertical distance from the wall:

- 1- If the robot heading towards the outer wall, it will keep moving until the front radical sensor, which is on the wall side, read lower than a specified distance.
- If the robot heading towards the inner wall, it will keep moving until the back radical sensor, which is on the outer wall side, read higher than a specified distance.

The Arduino will send signal to the servo motor to turn the same angle, but this time in the opposite direction using Gyroscope sensor. In both Normal case and Extreme Angle case, the used formula considers that the robot's angle is 90 degrees. However, some errors can happen that may lead to more or less than 90 degrees, we use the formula below to calculate this angle simple geometry and correct the desired angle:

$$AngleCorrection = tan^{-1}((D_{S1} - D_{S2})/D_{S1S2}) \tag{3.9}$$

It should be noted that both equations yield results in Radian, so a conversion formula is used to convert the angles to degrees.

3.5.2 Navigating Methods

The process of robot's navigation at Straight-Froward sections controls the movements of robot after passing the pillar until it reaches a corner section. We divide these movements into three main methods:

- Follow-Right (for avoiding red pillar)
- Follow-Left (for avoiding green pillar)

– Follow-Middle only when robot detects no traffic sign (it is applicable for first section).

for every method we focused on the development of robust navigation methods to ensure safe and stable following of a robot. A fundamental requirement is the implementation of a reliable following algorithm capable of maintaining these essential criteria. While the Proportional-Derivative (PD) following controller effectively ensures a safe path, it keeps the robot at a calibrated setpoint (constant path) parallel to the wall, exploiting its dynamic capabilities.

The PD controller enables the vehicle to maintain a zero degrees angle from the wall, thereby keeping the robot moving in a straight trajectory. The angle calculation is derived from a specified formula based on ultrasonic readings.

Nonetheless, scenarios may arise where the robot approaches the wall in a hazardous manner, necessitating a prompt response. To handle such situations effectively, we devide the area within the straight forward section into two distinct regions: a protected area and an unprotected area. In the protected area, the robot utilizes the PD controller to follow the specified angle. Conversely, the unprotected area mandates the robot to execute abrupt turns, facilitating a swift exit from the dangerous zone and subsequent return to the protected area for continued PD following.

3.6 AVOID PILLARS (GENERAL)

These steps outline the process for navigating a vehicle using a gyroscope and radical sensors. The navigation will be divided into three distinct phases:

– **Phase 1:** Turn a Fixed Angle using the Gyroscope
  i. The navigation process starts by receiving an ”R” or ”G” signal from the Raspberry Pi.
  ii. Use the gyroscope to turn a fixed angle in either a clockwise (CW) or counterclockwise (CCW) direction, depending on the scenario.
– **Phase 2:** Move Straight using the PD Controller
  i. Move straight along the angle that was turned, using a PD controlling to maintain a steady path.
  ii. Continue moving straight until the radical sensor reads a close distance. Depending on the situation, either the front or back radical sensor will be utilized.
– **Phase 3:** Turn in the Opposite Direction and Continue
  i. Turn the same angle as in phase 1, but in the opposite direction.
  ii. Continue following either the right or left path, based on the particular scenario.

3.7 MOVEMENT IN CORNER SECTION

These steps outline the process for navigating a vehicle in corner sections through three distinct scenarios. The following steps are considered for a counterclockwise (CCW) direction. For (CW) steps for scenarios 2 and 3 will be reversed.

Scenario 1: Entering the Corner from Follow-Middle

- **Phase 1:** Move in Follow-Middle
  - i. Continue moving in Follow-Middle within the corner section until the front sensor reads lower than 30cm.
- **Phase 2:** Turn Backward and Calibrate
  - i. Turn backward for 90 degrees CCW to face the following straight forward section, using the gyroscope.
  - ii. The turn-back move is calibrated to be finished in the middle of the corner (close to the outer border and not too far from it).
- **Phase 3:** Move Backward and Turn
  - i. Move backward (angle-Right) until the back sensor reads lower than 10cm.
  - ii. Move to the following straight forward section in the middle.

Scenario 2: Entering the Corner from Follow-Right

- **Phase 1:** Move in Follow-Right
  - i. Continue moving in Follow-Right within the corner section until the front sensor reads lower than 50cm.
- **Phase 2:** Turn Forward
  - i. Turn for 90 degrees CCW to face the following straight forward section, using the gyroscope.
- **Phase 3:** Move to the Following Straight Forward Section
  - i. Move to the following straight forward section in the middle.

Scenario 3: Entering the Corner from Follow-Right

- **Phase 1:** Move in Follow-Right
  - i. Turn forward for 30 degrees CW, using the gyroscope.
- **Phase 2:** Move Straight
  - i. Move straight following the same angle you have turned until the radical sensor reads lower than 50cm.
- **Phase 3:** Turn and Repeat
  - i. Turn the angle in phase 1 in the opposite direction.
  - ii. Repeat phases 1, 2, and 3 of scenario 1 to continue through the corner section.

3.8 OBSTACLE MANAGEMENT WITH PERSISTENT VISION

These Scenarios outline the process for navigating a vehicle through three distinct scenarios using a gyroscope and radical sensors, and detecting pillars using serial communication and image processing:

- **Phase 1:** 2 pillars in the section or one pillar in the beginning of the section
  - **i.** . If a big letter ”R” or ”G” is received, check if there is a close pillar using the back sensor (lower than 70cm).
  - **ii.** If the R or G signal is sent from the Raspberry Pi for a pillar closer than 50cm, that means the pillar is in the beginning of the section.
  - **iii.** Repeat phases 1 and 2 of the Avoiding Pillars navigation, but make sure to use the Radical Back sensors when needed.
  - **iv.** When phase 2 is completed, turn slightly inward to the straight forward section to check for another pillar.
  - **v.** If there is no pillar or a pillar of the same color, continue following either the right or left path.
  - **vi.** If there is a pillar of a different color, repeat phases 1, 2, and 3 of the Avoiding Pillars navigation.
- **Phase 2:** 1 pillar in the section in the middle or the end of the section
  The checking process in the first scenario gives information about where the only pillar is located and if the pillar is located in the middle or the end of the section the robot will continue following straight until it gets “R” or “G’ from the raspberry pi, and then it bypasses the pillar using general steps until it gets to the corner section.

3.9 OBSTACLE MANAGEMENT WITH EVENT DRIVEN VISION

This algorithm can be divided into three steps:

When the robot is in the proper place at start of cornern section, and facing the next straight forward section. The controller (Atmega) will send a signal containing either letters ”L” or ”R”, which indicates the direction of the vehicle (CW or CCW). Eventually, the processor recieves the signal and starts to take a capture for the faced straight forward section and identify the number of pillars in this section and their colors. Then it sends a signal to the controller taking the different possible cases. The vehicle deals with these cases as mentioned in 3.7 at Phase 1 based on previously established color information without looking for a second pillar in the section.

Algorithm	Event Driven Vision	Persistent Vision
Approach	- Waits for a signal indicating the robot is in the first section and stops. - Processes the scene using the camera only once and sends information as a signal to the Arduino controller. - Robot knows exactly what to do in the entire current section.	- Processes the scene continuously without stopping until the three laps are completed. - Constantly analyzes the image to detect and react to obstacles in real-time. - After avoiding a pillar, the robot turns a small angle to check for the existence of another pillar.
Advantages	- Provides precise information about the current section, enabling accurate decision-making. - Robot knows exactly what actions to take based on the processed image.	- Real-time processing allows the robot to react swiftly to obstacles. - Continuous processing minimizes the risk of missing or misclassifying pillars. - Less reliant on accurate long-distance recognition of red and green pillars.
Disadvantages	- Relies on a signal to indicate the start of each section, which would suffer from delays or synchronization issues. - May struggle with red or green pillars isolating and distance calculation from a distance, which may lead to incorrect actions.	- Lacks prior knowledge of the current section, requiring continuous processing and decision-making. - Requires the robot to turn and check for the presence of pillars, potentially affecting efficiency and speed.
Suitability	- Suitable when accurate knowledge of each section is crucial for decision-making. - May be suitable for scenarios where pillars are reliably detectable from a close distance.	- Suitable for real-time obstacle detection and avoidance. - Can handle scenarios where long-distance recognition of pillars is challenging or unreliable.

Table 3.1: Comparison of Algorithms: Event Driven Vision vs. Persistent Vision

3.10 EFFICIENT PLAY-FIELD PILLAR DATA STORAGE

When the play-field randomization has been completed, and the pillar colors and their positions within the play-field remain constant over the course of three laps, it becomes advantageous to retain the data regarding the play-field's pillar locations and colors. The optimal approach is for the robot to capture this information during the first lap and subsequently suspend image processing, relying solely on the stored data for subsequent laps.

As our algorithm is designed to account for the positions of these pillars, it is worthwhile to store this information in a List. Each element in the list should contain comprehensive information about a particular section, as explained in detail within the code.

3.11 LAST LAP DIRECTION

The determination of the final turn direction hinges on the color of the pillar encountered during the last lap, specifically in the second-to-last lap. Two scenarios arise: if the pillar's color is green, the robot should continue the final turn in the same direction. Conversely, when the pillar is red, the robot is required to execute a reversal of the previous turn. The process of reversing the direction is undertaken within the corner section and is detailed as follows:

To transition from a counterclockwise (CCW) to a clockwise (CW) direction, the following steps are followed:

- Continue moving until the front distance sensor registers a distance of less than 75cm.
- Execute a 90-degree turn in the CCW direction.
- Move in reverse until the rear distance sensor indicates a distance less than 45cm.
- Move forward in the CCW direction until the robot reaches an angle of 30 degrees.
- Turn 80 degrees and proceed in reverse in the CCW direction.
- Continue moving in reverse until the rear distance sensor detects a distance less than 10cm.

CHAPTER 4

Vehicle Photos

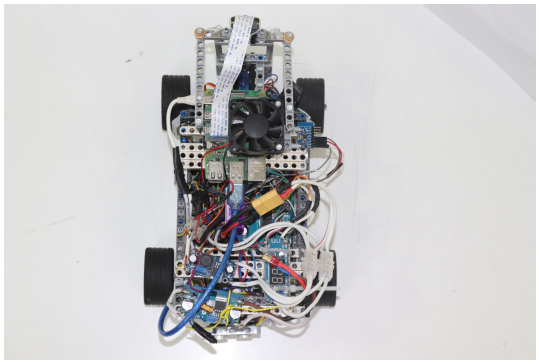


Figure 4.1: Top View

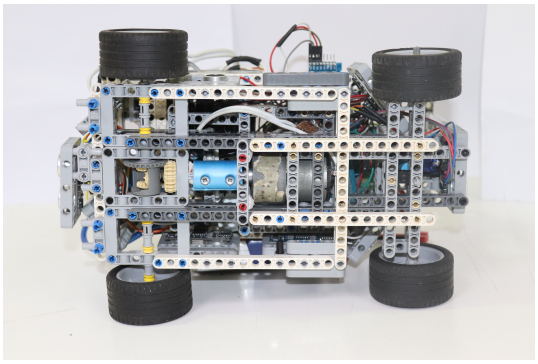


Figure 4.2: Bottom View

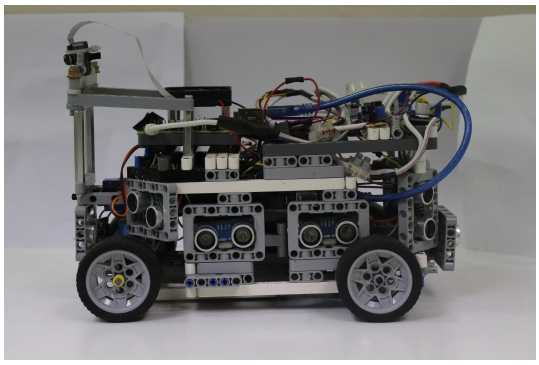


Figure 4.3: Side View (Left)

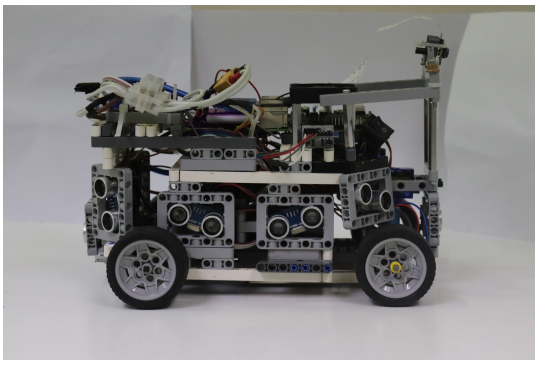


Figure 4.4: Side View(Right)

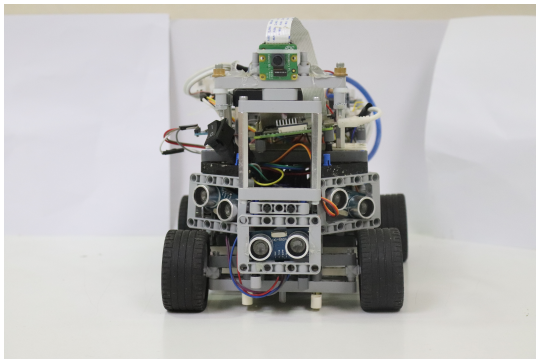


Figure 4.5: Front View

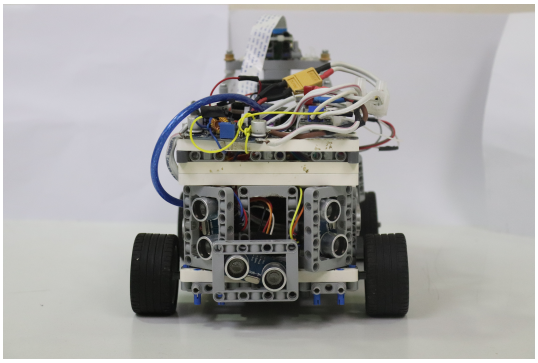


Figure 4.6: Back View

CHAPTER 5

Team Photos

Figure 5.1: Official Photo

Figure 5.2: Funny Photo

5.0 CONCLUSION

In conclusion, the autonomous vehicle we created for successfully solved the problems proposed in the WRO 2023 Future Engineers Category (Open Challenge - Obstacle Challenge) with the integration of mechanical and mathematical engineering principles, programming, and computer vision. The advanced programming, real-time decision-making and computer vision algorithms enabled the vehicle to plan paths and make informed choices leading to a vehicle can navigate autonomously, aligning with the goal of connecting the world. Finally, the algorithm can improved to solve the tasks in shorter time and this project paves the way for further advancements and underscores the potential for a more connected world through innovative engineering solutions.

Bibliography

[1] H. Kuntze, T. Schunck, and M. Gabele, "Steering ratio and steering wheel torque of automotive steering systems," *ATZ Automobiltechnische Zeitschrift*, vol. 114, no. 6, pp. 604–610, 2012.

[2] R. N. Jazar, *Advanced Vehicle Dynamics*. Springer, 2019, ISBN: 978-3-030-13062-6. DOI: https://doi.org/10.1007/978-3-030-13062-6. [Online]. Available: https://link.springer.com/book/10.1007/978-3-030-13062-6.

[3] S. International, "Motortoy car sample lesson plan," *SAE Learn Education*, 2016. [Online]. Available: https://www.sae.org/binaries/content/assets/cm/content/learn/education/motortoycar-samplelessonplan.pdf.

[4] W. H. Crouse and D. L. Anglin, *Automotive Mechanics*. McGraw-Hill, 2014. [Online]. Available: http://opac-library.adamasuniversity.ac.in/cgi-bin/koha/opac-detail.pl?biblionumber=14573.

[5] Sariel. (2023). "Wheels," [Online]. Available: http://wheels.sariel.pl/.

[6] L. Group. (2023). "Sustainability - product safety - materials," [Online]. Available: https://www.lego.com/ms-my/sustainability/product-safety/materials#:~:text=ABS,as%20LEGO%20DUPLO%C2%AE%20bricks.

[7] R. Pi, "Raspberry pi 4 model b," [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/.

[8] Arduino, "Arduino mega 2560," [Online]. Available: https://store.arduino.cc/products/arduino-mega-2560-rev3.

[9] H. Technology, "Bts7960 motor driver," [Online]. Available: https://uk.farnell.com/c/sensors-transducers/sensors/ultrasonic-sensors.

[10] STMicroelectronics, *L298n motor driver datasheet*, 2023. [Online]. Available: https://www.st.com/resource/en/datasheet/l298.pdf.

[11] Amazon, "Riorand lm2596," [Online]. Available: https://www.amazon.com/RioRand-LM2596-Converter-1-23V-30V-5Pcs-LM2596/dp/B008BHB4L8.

[12] O. Circuit, "Towerpro sg90 9g micro servo motor," *Servo Motors*, [Online]. Available: https://www.jsumo.com/jsumo-titan-dc-gearhead-motor-12v-1000-rpm-hp.

[13] robocraze, "11.1v 2200mah 25c lipo battery," [Online]. Available: https://robocraze.com/products/11-1v-2200mah-25c-li-po-battery.

[14] Jsumo, "Titan dc gearhead motor 12v 1000 rpm hp," [Online]. Available: https://www.jsumo.com/jsumo-titan-dc-gearhead-motor-12v-1000-rpm-hp.

[15] E. Wings, "Mpu 6050," [Online]. Available: https://www.electronicwings.com/sensors-modules/mpu6050-gyroscope-accelerometer-temperature-sensor-module.

[16] F. UK, "Ultrasonic sensor," [Online]. Available: https://uk.farnell.com/c/sensors-transducers/sensors/ultrasonic-sensors.

[17] S. A. Saleh, D. Jewett, and S. A. R. Panetta, "The analysis, modeling, and capabilities of grounding system designs," *IEEE Transactions on Industry Applications*, vol. 58, no. 5, pp. 5908–5920, 2022.

[18] N. Mohd Ali, "Performance comparison between rgb and hsv color segmentations for road signs detection," *Applied Mechanics and Materials*, vol. 393, Sep. 2013. DOI: 10.4028/www.scientific.net/AMM.393.550.