Autonomous Rover Navigating With Ultrasonic Sensors

Leela Krishna Sai Tharun Kammili
Mechanical Engineering, SEMTE
Ira A Fulton School Of Engineering,
Arizona State University
Tempe, USA
lkammili@asu.edu

***Abstract*—**

This paper describes the development and construction of an autonomous rover for automated parking with three different modes, including parallel parking, crab walk parking, and obstacle avoidance. The system is integrated and controlled using an Arduino uno microcontroller that operates the car's peripheral hardware, including DC motors for the wheels, a micro servo motor, and ultrasonic range sensors. The study focuses on the use of an ultrasonic sensor to measure distances and examine the environment, with the goal of reducing the time required for parking between parked vehicles. The parking mode is provided to the driver, and upon selection, the rover parks automatically in the designated parking spot using the sensor suite. The aim of this project is to construct a rover which is customizable and can be used for variety of functions like car parking and obstacle avoidance. The use of Arduino and other technologies provides a flexible and customizable platform for future developments, allowing the rover to be adapted to suit specific needs and applications. The development of an autonomous rover for parking and obstacle avoidance addresses the need for better safety measures in parking lots to prevent accidents and fatalities, particularly considering the 9% of pedestrian deaths in parking lots reported by the National Safety Council [1]. The integration of various technologies such as Arduino, ultrasonic sensors, and Mecanum wheels provides a valuable tool for various industries and organizations, including search and rescue operations, surveillance, and exploration.

Keywords—Arduino uno, ultrasonic sensors, mecanum wheels, autonomous parking, crab parking, autonomous rover.

I. Introduction

The increasing number of automobiles worldwide has led to a decrease in parking spaces, causing difficulty for urban residents to park all their vehicles. As most households in developed countries have a minimum of two cars, parking on the roadside or in parking lots has become increasingly challenging. The development of computer vision and new sensors using microcontrollers can make parking easier for drivers. The latest cars and trucks available on market has several safety features built in such as forward collision warnings, lane departure warnings, rear cross-traffic warnings, blind spot warnings, and driver assistance features such as automatic emergency braking, blind spot intervention, adaptive cruise control, and lane keeping assistance. These features work based on sensors such as RADAR, LIDAR, ultrasonic sensors, and cameras. Most of the current parking assist technologies use cameras to identify objects and assist in parking the car using the manufacturer's pre-trained algorithm. The car can park in a parking lot by identifying the space using the lines and positioning itself using the cameras and available sensors.

However, a study by the National Safety Council revealed that 9% of pedestrian deaths in parking lots are due to backup incidents[1]. Although most vehicles on road today are equipped with safety features and a back camera to assist in parking, there is a need for better safety measures in parking lots to prevent accidents and fatalities. The aim of this project is to develop an autonomous rover that can perform tasks such as parallel parking, crab walk parking, and obstacle avoidance. The rover is equipped with various technologies, including an Arduino uno microcontroller, ultrasonic sensors, Rechargeable NiMH batteries, voltage sensors, 4 DC motors with gear reduction, mecanum wheels, and an L293D Arduino extension module. These modules are integrated and controlled through a custom written program that enables the rover to perform various tasks autonomously.

The use of ultrasonic sensors allows the rover to measure distances and examine the environment, enabling it to park between parked vehicles with greater accuracy and efficiency. The parking mode is provided to the driver, and upon selection, the rover parks automatically in the designated parking spot using the sensor suite. The system can also perform crab walk parking, which is useful in tight spaces, and obstacle avoidance, which enables the rover to move in any direction and avoid obstacles. The integration of various technologies such as Arduino, ultrasonic sensors, and mecanum wheels provides a valuable tool for various industries and organizations, including search and rescue operations, surveillance, and exploration. The project aims to use latest sensors and algorithms in autonomous robots and promote their use in various applications, with a focus on safety, efficiency, and flexibility. The use of Arduino provides a flexible and customizable platform for future developments, allowing the rover to be adapted to suit to specific needs and applications for the given environment.

In this project, we explain the process of designing our autonomous rover, with detail list of the hardware and software components used, and present the results of our experiments to validate its performance. Section VI describes the design and construction of the rover, including the hardware and software components. Section VII presents the results of our experiments to validate the performance of the rover. Finally, Section VII provides a discussion of the results and the potential applications and improvements of the rover.

II. Objective

The objective of this project is to develop an autonomous rover capable of performing parallel parking, crab walk parking and obstacle avoidance. The rover is designed to use ultrasonic sensors, Rechargeable NiMH batteries, a voltage sensor, 4 DC motors with gear reduction, mecanum wheels and an L293D Arduino extension module, which are integrated with a program for autonomous parking. The rover is powered by a custom made battery pack which can be easily charged and reused, making it suitable for long-term usage. The mecanum wheels provide the rover with omnidirectional movement, allowing it to easily maneuver in tight spaces. The ultrasonic sensors are used to detect obstacles and measure distances of objects nearby in range, allowing the rover to avoid collisions with other vehicles or obstacles in the parking area. The voltage sensor is used to monitor the battery voltage and ensure that the rover is not operating with a low voltage, which could result in failure of the rover. The L293D Arduino extension module is used to control the DC motors, allowing for precise control of the rover's movement.

The program for the autonomous parking modes is designed to allow the rover to park in a variety of situations. The parallel parking mode allows the rover to park in between two parked vehicles, while the crab walk parking mode allows the rover to park diagonally across two parking spaces. The obstacle avoidance mode allows the rover to navigate around obstacles in the parking area. The purpose of this project is to provide a reliable and efficient solution to parking in urban areas. By utilizing autonomous technology, the rover can park itself in tight spaces, reducing the risk of collisions and improving the overall safety of parking. Also, the autonomous rover is designed to be cost-effective and easily maintained, making it a practical solution for a variety of parking scenarios.

III. Literature review

As the development of autonomous technology increases, there are numerous research studies conducted in the field of robotics for autonomous vehicle parking. In this literature review, we have explored three papers that have been published in the recent years regarding autonomous vehicle parking.

The first paper titled "Integrating Mecanum wheeled omni-directional mobile robots in ROS" [2] discusses the integration of ROS (Robot Operating System) with Mecanum wheeled robots. The authors proposed a method for using Mecanum wheeled robots to navigate and map the environment with the help of ROS. They discuss the basic concepts of Mecanum wheels and their properties that make them suitable for omni-directional movement. The paper presents a detailed description of the hardware and software components used for the integration of Mecanum wheeled robots with ROS. The authors used a Mecanum wheeled rover platform, equipped with a range of sensors including sonar, lidar, and a camera, and a microcontroller for controlling the robot's movements. The rover's motion planning and control were implemented using ROS, which allowed for real-time mapping and navigation in a variety of environments. The paper also presents the experimental results of their study, where the authors evaluated the performance of the integrated system in different environments. The experimental results showed that the proposed system was able to effectively navigate and map the environment, demonstrating the potential for using Mecanum wheeled robots with ROS in a variety of applications.

The second paper titled "Obstacle Avoidance System for Unmanned Ground Vehicles by Using Ultrasonic Sensors" [3] proposed an obstacle avoidance system for unmanned ground vehicles (UGVs) using ultrasonic sensors. The authors developed an autonomous rover equipped with four ultrasonic sensors to navigate and avoid obstacles in a simulated environment. The authors focused on the implementation of an intelligent system that would enable UGVs to perform tasks without human intervention. The authors used an ultrasonic sensor to detect obstacles within a range of up to two meters. The obstacle detection algorithm is based on a simple thresholding method. The system is equipped with a PID controller that adjusts the velocity of the UGV to avoid collisions with obstacles. The experimental results demonstrate that the proposed system is capable of navigating through an environment filled with obstacles and avoiding collisions.

The third paper titled "Infrared Line Following and Ultrasonic Navigating Robot with ATMEGA328 Pro" [4] presents the design and development of an autonomous robot capable of following a black line on a white surface using infrared sensors and navigating using ultrasonic sensors. The robot is powered by an ATMEGA328 Pro microcontroller and consists of four infrared sensors and two ultrasonic sensors for detecting obstacles and distances. The authors proposed a control algorithm that uses a proportional-integral-derivative (PID) controller to adjust the speed and direction of the robot based on the feedback from the sensors. The authors evaluated the performance of the robot in terms of line-following accuracy, obstacle avoidance ability, and navigation precision. The report stated that the robot was able to follow a curved black line with a high degree of accuracy and avoid obstacles in real-time. The authors also demonstrated the capability of the robot to navigate through a maze using ultrasonic sensors. The paper suggests autonomous mobile robots un low-cost, easy-to-implement solution for line-following and obstacle avoidance using infrared and ultrasonic sensors. The use of a PID controller for control algorithms also improves the accuracy and reliability of the robot's performance. The results of the study show the potential of such robots in applications such as automated guided vehicles, warehouse robots, and industrial automation.

IV. System design

The proposed system consists of an autonomous rover that navigates through a given environment while avoiding obstacles, parking in tight spaces both in parallel and crab modes using ultrasonic sensors. The rover is built using an Arduino UNO board, four DC motors, an L293D extension motor shield, and four ultrasonic sensors. The rover's motion is controlled by the Arduino UNO board, which sends signals to the L293D motor driver to control the DC motors. The ultrasonic sensors are mounted on the front, back and left sides of the rover, providing 270-degree coverage of the environment. The system's software is developed using the Arduino IDE, and it is based on a custom designed algorithm. The system's mechanical design, software and actual prototype are discussed in the following sections.

A. Mechanical design

The mechanical design of a mobile rover is critical in ensuring the rover's stability, functionality, and performance. Our mobile rover design features a metallic frame consisting of two plates supported by hex extension nuts, mecanum wheels, and 3D printed battery holder and ultrasonic module holder. In this section, we will provide a detailed discussion of each mechanical component of the robot, including its significance, design features, and performance.

1. Metallic Frame:

The metallic frame serves as the base for the entire robot's mechanical system. It is responsible for holding all the components of the robot, such as the wheels, motors, and sensors, in place. The frame is made of high-strength aluminum alloy to ensure the robot's stability and durability. The use of aluminum makes the frame lightweight, which is crucial for the robot's mobility. The choice for choosing the metallic frame was approached as a result of failure of the stability obtained by the earlier 3d printed chassis using the PLA. The 3D printed PLA chassis was not durable and could not withstand the impact shocks and component loads. The metallic frame consists of two plates connected by hex extension nuts. The plates are designed to provide maximum surface area for mounting the components. The hex extension nuts provide a sturdy connection between the plates, ensuring that the robot remains stable during operation. The use of hex nuts also makes it easy to disassemble the robot for maintenance and repair.

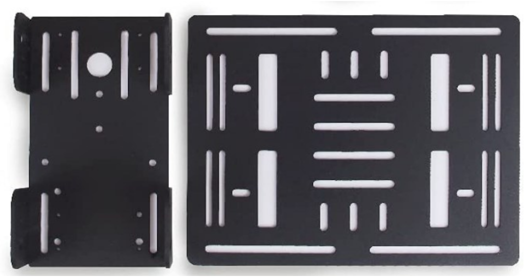


Fig. 1. Aluminum alloy metal frame

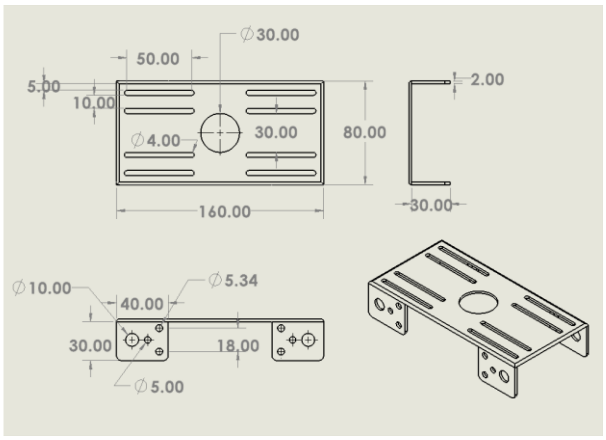


Fig. 2. Lower chassis plate design

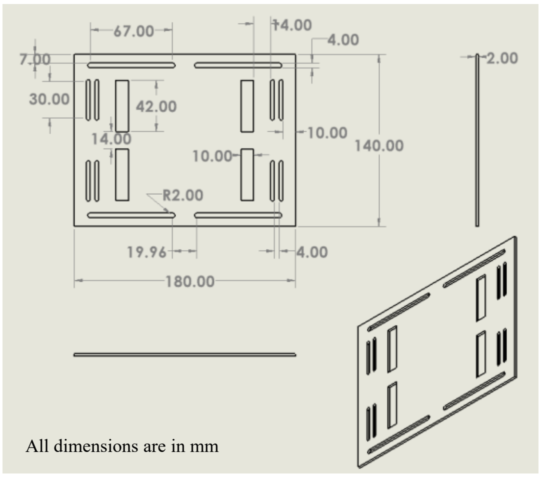


Fig. 3. Upper chassis plate design

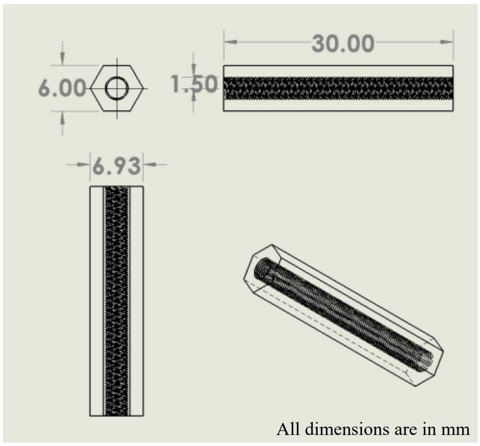


Fig. 3. Hex extension nut design

2. Mecanum Wheels:

Mecanum wheels are special wheels designed to allow our rover to move in any direction, including sideways and diagonally ref figures 3 & 4, by varying the speed and direction of rotation of the wheels without changing its orientation. The wheels consist of several small rollers that are arranged at an angle of 45° to the hub's axis around the circumference of the hub. As the wheels rotate, the rollers move other than the hub's axis, generating a force that allows the robot to move in any direction. The fit of the mecanum wheel is also crucial for its function. The wheel's rollers must be mounted precisely and securely to the hub to ensure that they rotate freely but cannot move along the hub's axis.

1	2	3	4	Drive
F		F		⌜
F	F	F	F	^
	F		F	⌝
R	F	R	F	>
R		R		⌟
R	R	R	R	∨
	R		R	⌞
F	R	F	R	<

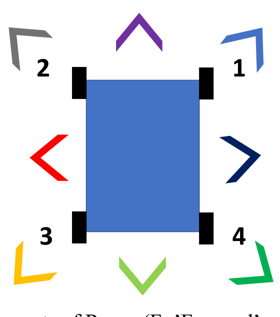


Fig. 3. Transitional movements of Rover (F='Forward', R='Rear')

Our robot features four mecanum wheels, with each wheel consisting of nine rollers. The wheels are made of high-strength rubber to reduce the robot's weight while ensuring durability. The wheels are fixed to DC motors output shaft, which provide the required torque to drive the wheels. The use of mecanum wheels allows the robot to move in any direction, making it suitable for navigating tight spaces.

1	2	3	4	Drive
F	R	R	F	↺
R	F	F	R	↻

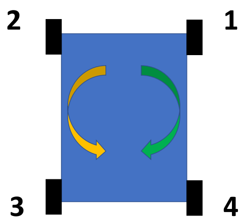


Fig. 4. Rotational movements of Rover (F='Forward', R='Rear')

Mecanum wheels have several benefits that make them an attractive option for mobile robotics. They have omnidirectional movement, allowing the robot to move in any direction without having to rotate or reposition itself. Another benefit of mecanum wheels is their ability to provide both forward and lateral force, making them highly maneuverable. This is possible by rotating each wheel independently, allowing for precise control over the direction and speed of movement.

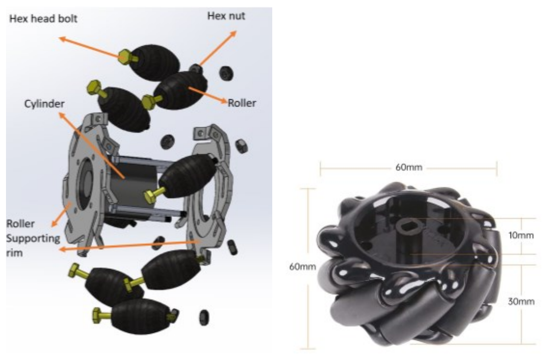


Fig. 5. Mecanum Wheel design and actual wheel used in rover.

This makes mecanum wheels ideal for navigating around obstacles, as they can move in any direction and change direction quickly as needed. In addition to their maneuverability, mecanum wheels are also highly durable and low maintenance. They are designed to withstand heavy loads and rough terrain, making them ideal for outdoor or industrial applications. They also require minimal maintenance, as their unique design reduces wear and tear on the wheels and bearings. Mecanum wheels are ideal for a wide range of applications, from warehouse automation to search and rescue robotics.

3. 3D Printed Battery Holder and Ultrasonic Module Holder:

he battery holder and ultrasonic module holder are 3D printed components designed to hold the battery and ultrasonic module securely in place to the rover chassis. The battery holder is fixed to the metallic frame and is designed to hold the battery in center of chassis that maximizes the robot's stability. The ultrasonic module holder, on the other hand, is designed to hold the ultrasonic sensor in a position that allows it to detect obstacles accurately. The use of 3D printed components provides several benefits. Firstly, 3D printing allows us to create customized components that fit the robot's design requirements perfectly. Secondly, 3D printed components are lightweight, which reduces the robot's weight and enhances its mobility. 3D printing is a cost-effective method of creating components, reducing the robot's overall cost. The battery module holder is designed in such a way that the holder is provided with perforations to help mounting in most places on the aluminum chassis. The ultrasonic module holder is also provided with 3 holes such that it can be mounted in most positions making it flexible and easy to mount. This holder has a slot to facilitate the ultrasonic sensor to slide for a better fit and is also capable of mounting it on a micro servo motor.

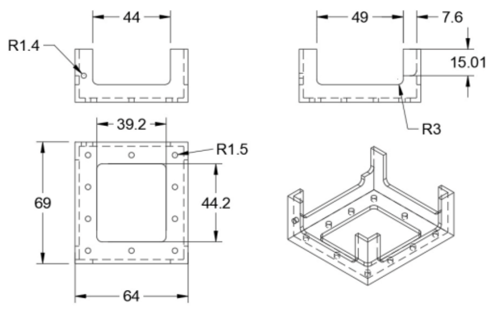


Fig. 6. 3D printed battery pack holder.

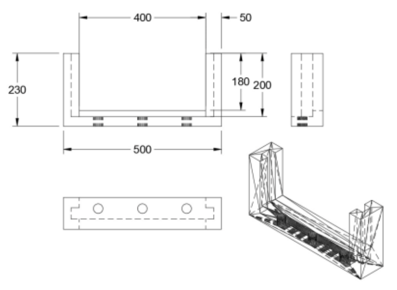


Fig. 7. 3D printed ultrasonic sensor holder.

4. Electrical Systems

The electrical system of our rover includes the microcontroller, motor driver, ultrasonic sensors, micro servo motor, and batteries. The microcontroller is the brain of our rover and is responsible for controlling all the other components. The motor driver is used to control the movement of the DC motors, while the ultrasonic sensors are used to detect obstacles and measure distance of the surroundings. The batteries provide the power needed to operate the electrical components. The electrical components used in the rover are discussed below in detail.

Arduino uno: The Arduino Uno is a microcontroller board built on the ATmega328 microcontroller. It has 14 digital input/output (I/O) pins, 6 analog input pins, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header, and a reset button. We will discuss the significance of each of these components in detail. The Arduino Uno has 14 digital I/O pins, which can be used for both input and output operations. These pins can be programmed to turn on or off external devices such as LEDs, motors, and relays. Each digital pin can provide a maximum of 20 mA of current and has an internal pull-up resistor of approximately 20-50 kΩ. These pins are labeled as D0 through D13. The Arduino Uno has 6 analog input pins, labeled as A0 through A5. These pins can be used to read analog signals from sensors such as temperature sensors, light sensors, and potentiometers. Each analog pin has a resolution of 10 bits, which means it can measure voltages from 0 to 5 volts with a resolution of 5/1024 volts (approximately 4.9 mV). The ATmega328 microcontroller is the major component of the Arduino Uno board. It has 32 KB ISP flash memory for storing the program code, 2 KB of SRAM for storing variables, and 1 KB of EEPROM for storing non-volatile data. The microcontroller runs at a clock speed of 16 MHz and can execute up to 16 million instructions per second. It also has a number of built-in peripherals, such as timers, serial communication ports, and analog-to-digital converters (ADCs), which make it a versatile and powerful microcontroller. The USB port on the Arduino Uno allows it to be connected to a computer for programming and serial communication. The USB connection is also used to power the board, which means that it can be powered by the computer's USB port or by an external power supply connected to the power jack. The power jack on the Arduino Uno allows it to be powered by an external power supply. Uno can accept a voltage input of 7 to 12 volts DC and has a built-in voltage regulator that can convert this input voltage to 5 volts, which is used to power the microcontroller and other components on the board. The In-Circuit Serial Programming (ICSP) header on the Arduino Uno allows it to be programmed using an external programmer or another Arduino board. This is useful when the bootloader on the microcontroller has been corrupted or if a custom bootloader needs to be installed.

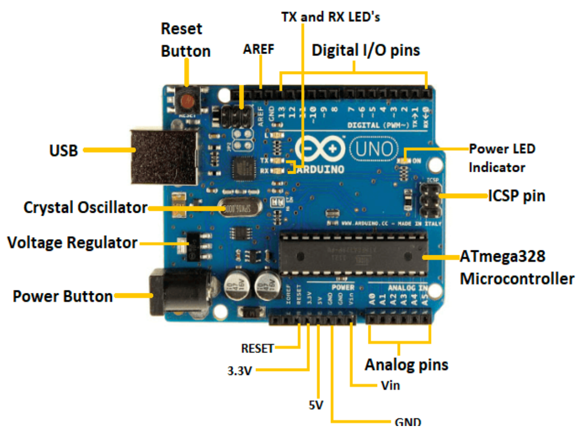


Fig. 8. Arduino uno R3 micro controller.

The reset button on the Arduino Uno resets the microcontroller and restarts the program execution from the beginning. Reset button can be used when the program crashes or when the board needs to be reprogrammed. The Arduino Uno has LED indicators that provide feedback on the status of the board. The power LED indicates whether the board is powered on, while the TX and RX LEDs indicate whether data is being transmitted or received through the USB port. The LED connected to pin 13 can be used to indicate the status of the program.

Our rover project utilizes the Arduino Uno board as the main controller for the entire system. One of the advantages of using the Arduino Uno board is its usability. It has both input and output pins, including digital I/O pins, analog pins, and PWM pins, which allows for the connection of various sensors and actuators. In our rover project, we use the analog pins to interface with the ultrasonic sensors and voltage sensor, while the digital I/O pins are used to connect L293D motor shield module and micro servo motor. The PWM pins are used to control the speed of the DC motors, which are used to drive the rover. Another benefit of using the Arduino Uno board is its ease of programming. The board is programmed using the Arduino IDE, which is based on a version of the C++. The IDE includes a library of pre-written functions, which can be easily used in the code, making it easier to program the board. This feature makes the board accessible to users with little to no experience in programming. This feature is particularly useful during the development and testing phases of the project. The USB port can be used to upload the code onto the board, as well as monitor the sensor readings and view the output signals in real-time.

a) Ultrasonic sensor: Ultrasonic sensors are a crucial component of our rover project as they allow for accurate distance measurements to be taken, aiding in obstacle detection, avoidance, and parking scenarios. Specifically, we are using the HY-SRF05 precision Ultrasonic range finder model, which is an upgrade from the HC-SRO4 and offers slightly better accuracy[5] . Ultrasonic sensors have several advantages over IR sensors. They are not affected by the color and lighting of obstacles and have wider angles of detection with lower minimum distances, ensuring that obstacles are not missed by a narrow sensor beam. The SRF05 operates by sending ultrasonic pulses and measuring the time it takes for the pulses to travel to the obstacle and back. It has a non-contact measurement function that can detect distances from 2cm to 400cm[5] , with a ranging accuracy of up to 3mm. The module includes transmitters, receivers, and control circuits, providing a stable and accurate distance measurement for the rover. The SRF05 can be used in either 1-pin trigger/echo or 2-pin modes. When the mode pin is tied to ground, the SRF05 can use a single pin for both trigger and echo, saving valuable pins on the controller. Additionally, HY-SRF05 includes a small delay before the echo pulse to give slower controllers time to execute their pulse commands. The HY-SRF05 has several features that increase its flexibility and ease of use. The ultrasonic sensor module has five pins, each serving a specific purpose. The first pin is VCC, used to supply power to the module, which should be connected to the 5V pin of the Arduino board. The second pin is the trigger pin, used to trigger the ultrasonic pulse, and the third pin is the Echo pin, used to receive the ultrasonic pulse and measure the time it takes to return after reflecting off an object. The fourth pin is the Out pin, which is not commonly used, and the fifth pin is the Ground pin, used to provide a common ground between the module and the Arduino board. To measure distances using the ultrasonic ranging module HY-SRF05, a 10 µS wide pulse is sent to the sensor on the Trigger Pin, which triggers the sensor to send out a 40 kHz wave. The output from the Echo Pin is monitored continuously, and when the Echo Pin goes high, a timer is started. When the Echo Pin goes low again, the elapsed time from the timer is recorded, which is the time between sending the pulse and receiving the echo. The sound velocity is assumed to be constant at 340 m/s in air, and the distance is divided by 2 because the sensor returns the round-trip time, which doubles the distance measurement. This measurement procedure can be easily implemented using an Arduino microcontroller.

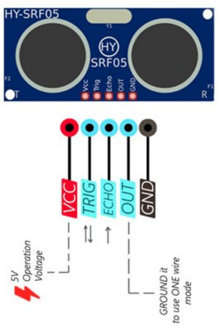


Fig. 9. HY SRF05 Ultrasonic sensor.

$$Distance(cm) = \frac{\left(Elapsed\ Time * Velocity\ of\ sound(340\frac{m}{s})\right)}{200} \quad \ldots\ldots (1)$$

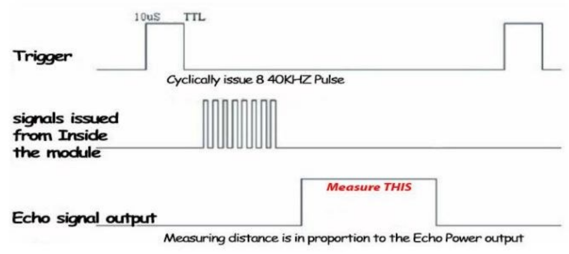


Fig. 10. Timing Diagram for Ultrasonic sensor.

b) L293D Motor shield module: The L293D Motor Shield is our choice for controlling DC motors in our rover project. The L293D module is mounted on top of the arduino uno board. It offers a versatile solution that allows you to control multiple DC motors with ease. The shield is compatible with both DC motors and steppers and provides a range of useful features, such as PWM speed control, direction control, and current sensing. To use the L293D Motor Shield in our project rover, we connected it to Arduino uno ATmega328 microcontroller board. The shield requires a separate power supply, which can be up to 12V, and power supply can also be drawn from arduino uno and it can provide up to 1.2A of current to each motor channel. This means that we were able to power 4 motors with a total current draw of up to 4.8A. The L293D motor Shield is used to drive the motors. The L293D is a popular motor driver IC that can drive two DC motors or one stepper motor. It's a adaptable controller that can handle a wide range of motors, and it's very efficient, with a low voltage drop. The L293D motor Shield has several pins including the motor outputs, which are connected to the motors, and the enable pins, which control the speed of the motors using PWM. There are also direction control pins, which determine the direction of the motor rotation, and the current sensing pins, which allow us to monitor the current consumption of the motors. The L293D Motor Shield not only uses the L293D motor controller, but also the 74HC595 shift register[6]. This shift register is used to expand the number of output pins available on the microcontroller, allowing the shield to control servo motors or other PWM-controlled devices. The 74HC595 shift register is a high-speed, 8-bit serial-in, parallel-out shift register. It has an operating voltage range from 2V to 6V and can handle a maximum output current of 35mA. It is commonly used in LED matrix displays but is also suitable for driving motors and other high-current devices.

To connect the L293D motor Shield to our rover, we connected the motor power supply to the shield's power terminals. We connected the motors to the motor outputs and the enable pins to the PWM pins on your microcontroller board. The direction control pins should also be connected to our microcontroller board, and we can use the current sensing pins to monitor the motor current if required. Using the L293D motor Shield in our project rover offers several benefits. It provides an easy and reliable way to control multiple motors. With the shield, we were able to control up to four DC motors or two stepper motors, allowing us to create a versatile rover design that can traverse a range of terrains. The PWM speed control also allows us to adjust the speed of the motors smoothly, giving us precise control over the rover's movements. The shield also has built-in protection circuits, such as over-current protection and over-temperature protection, which help to keep the motors and the shield safe.

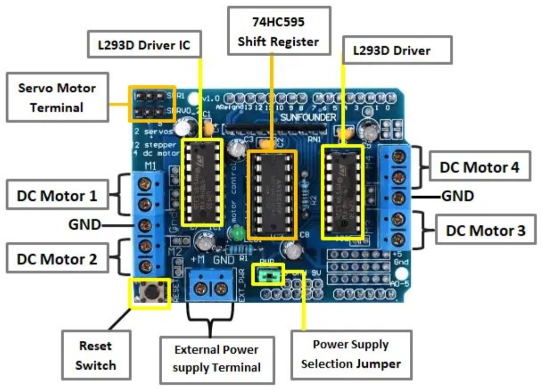


Fig. 10. L293D motor driver shield.

c) Voltage Sensor: In the rover design with Arduino, a voltage sensor was used that has a measurement range between 0-25V. The voltage sensor works by measuring the voltage across a resistor in the circuit. The output of the voltage sensor is a signal value which ranges from 0 to 1023 to the analog pin, that is proportional to the voltage being measured. This signal value from voltage sensor can be read by the Arduino's analog input pins. The voltage sensor used in the rover design is designed to measure DC voltage in the range of 0-25V. It has a high input impedance and is capable of measuring voltages with an accuracy of +/- 0.5%. The sensor is easy to use and can be connected to the Arduino with just three wires: power, ground, and signal. To use the voltage sensor, the power and ground wires are connected to the Vcc and GND pins on the Arduino. The signal wire is connected to one of the Arduino's analog input pins. The voltage being measured is then connected to the input terminals on the voltage sensor.

Once the voltage sensor is connected to the Arduino, the signal value being measured can be read using the analogRead() function. The signal value can then be converted to a real-world voltage value by multiplying the analog value by the appropriate scaling factor. The voltage sensor used in the rover design is a useful tool for monitoring the voltage of the rover's batteries and supply to rover’s electrical peripherals such as DC motors, micro servo and ultrasonic sensors.

Fig. 11. Voltage sensor.

d) Micro servo SG90: The micro servo SG90 is a small and lightweight servo motor with high output power. It is capable of rotating approximately 180 degrees, with 90 degrees in each direction, and can be controlled using any servo code, hardware, or library. This makes it an excellent choice for beginners who want to create moving objects without the need to build a motor controller with feedback and gear box, especially since it can fit in small spaces. The SG90 micro servo has a speed of 0.1 seconds with 2.5 kg-cm torque, which makes it capable of providing sufficient power to rotate objects with ease. Its weight is only 14.7 grams, which further adds to its suitability for small projects. For our project, we used the SG90 servo motor to rotate the ultrasonic sensor HY-SRF05 sensor by 50 degrees and 117 degrees to capture the distance values whenever an obstacle is in front of it. The servo's speed of 0.1 seconds made it possible to rotate the sensor quickly and accurately.

Fig. 12. Micro servo SG90.

The SG90 servo motor has 2.5 kg-cm torque which provides enough power to rotate the ultrasonic sensor and keep it steady in place. The servo's weight of 14.7 grams made it easy to attach it to the rover without adding extra weight. The SG90 servo motor can operates in voltage between 4.8 to 6 volts, which makes it suitable for use with a wide range of batteries commonly used in robotics projects.

e) DC TT Motor: The TT motor is a type of DC motor that is commonly used in robotics and other DIY projects. The name "TT" refers to the shape of the motor, which resembles the letter T. The motor is compact and lightweight, making it choice for use in small robots and other applications where space is limited. The TT motor has a reduction ratio of 1:120, which means that the output shaft rotates 120 times slower than the motor itself. This reduction ratio provides the motor with more torque at the output shaft, which is useful for driving wheels or other mechanisms that require a lot of force. The output voltage of the TT motor is between 3-6V, which is suitable for most small robots and electronics projects. The output speed of the motor is between 120-240RPM, which is also ideal for driving small wheels or other mechanisms. In the design of our rover, we used four TT motors to drive the mecanum wheels. The reduction ratio of the motors provided the necessary torque to drive the wheels with sufficient force to move the rover in any direction. The output voltage and speed of the motors were within the appropriate range for our application, and we were able to easily control the motors using an L293D motor extension shield module mounted over Arduino uno.

Fig. 13. DC TT Motor.

B. Final Prototype

The final prototype of our rover includes a sturdy chassis with four TT motors attached to each wheel. The motors have a reduction ratio of 1:120, which provides enough torque to move the rover on different terrains. The output voltage of the motors ranges from 3-6V, and the output speed is between 120-240 RPM. This combination of voltage and speed helps in achieving optimal performance of the rover. The mecanum wheels are also attached to the motors, which allow the rover to move in different directions with ease. The wheels are connected in a way that each wheel rotates independently, which helps in achieving the desired movement of the rover. The ultrasonic sensor HY-SRF05 is attached to a micro servo SG90 that rotates 50 degrees and 117 degrees. The servo motor helps in directing the sensor towards different directions, which helps in detecting obstacles and avoiding them. In addition to the ultrasonic sensor HY-SRF05 mounted on the servo motor SG90, we mounted three more ultrasonic sensors HY-SRF05 type (Two on left and one on the rear side of the rover). The L293D motor controller and voltage sensor are included in the final prototype of the rover. The L293D module helps in controlling the speed and direction of the motors, while the voltage sensor helps in monitoring the battery voltage and preventing any damage due to overcharging or undercharging. The entire system is controlled by an Arduino board. The board is programmed with the necessary code to perform various functions, such as obstacle avoidance, parallel parking, and crab walking.

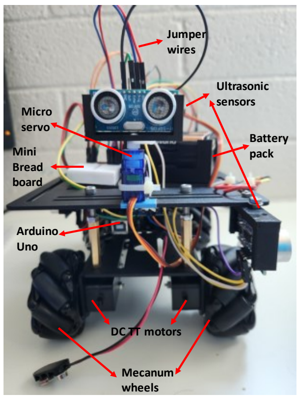


Fig. 14. Final prototype front view.

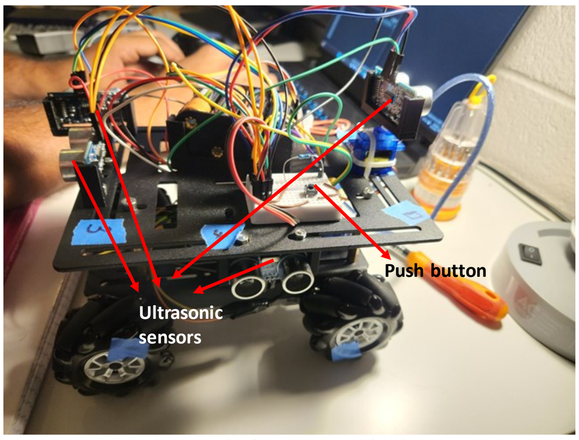


Fig. 15. Final prototype side view.

C. *Bill of materials*

The bill of materials (BOM) is a comprehensive list of all the components, parts, and materials required to build the rover. Here is a breakdown of the items required to build our rover (ref table I).

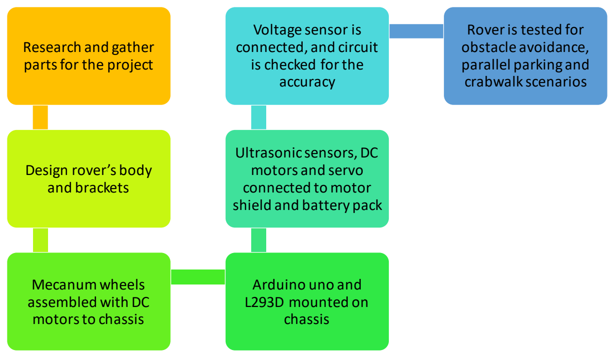
TABLE I. BILL OF MATERIALS

S. No	Part Name	Count
1	Aluminum Alloy Chassis	1
2	Mecanum wheels	4
3	TT Motor	4
4	M4 * 32mm copper pillar	4
5	M4 * 8mm round head machine screw	8
6	M3 * 25mm round head machine screw	8
7	M3 nuts	8
8	M2.3 * 16 tapping screw	4
9	TT motor coupling	4
10	Flat head self-tapping screw	4
11	Arduino uno	1
12	L293D motor extension shield	1
13	HY-SRF05 Ultrasonic sensors	4
14	Mini Bread Board	1
15	Micro servo SG90	1
16	Voltage sensor	1
17	Rechargeable Ni-MH Batteries	8
18	Press button	1
19	10KΩ Resistor	1

V. METHODOLOGY

The methodology section of our project will detail the steps taken to design and build the rover. The first step was to research and gather information on the components needed for the project, including the Arduino Uno board, motor shield, ultrasonic sensor, voltage sensor, micro servo SG90, and TT motors with a reduction ratio of 1:120. Once the components were identified, the next step was to design the rover's body and mounting brackets using CAD software. The design was then 3D printed using PLA material.

The mecanum wheels were assembled with the DC TT motors output shaft to the chassis and then arduino uno is mounted on top of the chassis at a height to facilitate colling by air flow. The L293D motor shield extension module is then mounted on the arduino. An ultrasonic sensor was mounted onto the micro servo SG90, which was then mounted onto the rover's body. While the other 3 ultrasonic sensors are mounted on the body to the 3d printed ultrasonic holder which was mounted on the chassis on the front, left and rear side of the rover.


Research and gather parts for the project
Design rover's body and brackets
Mecanum wheels assembled with DC motors to chassis
Arduino uno and L293D mounted on chassis
Ultrasonic sensors, DC motors and servo connected to motor shield and battery pack
Voltage sensor is connected, and circuit is checked for the accuracy
Rover is tested for obstacle avoidance, parallel parking and crabwalk scenarios
Fig. 16. Methodology Flowchart.

Next, the voltage sensor was connected to the motor shield, and the wiring was checked for accuracy. The Arduino code was then written to control the movement of the rover and the functions of the ultrasonic and voltage sensors. The code was uploaded to the Arduino Uno board, and the rover was tested to ensure all components were working correctly.

VI. SOFTWARE MODELING

The Arduino code is written in C++ using the Arduino IDE. The code is responsible for reading the sensor data, controlling the motors, and implementing the algorithms for obstacle avoidance, parallel parking, and crab walk. The code is structured into different functions and modules to make it more organized and maintainable. For detailed code refer to the appendix.

a) Drive mode switching : It starts by reading the state of a button connected to the buttonPin, and if it detects a change from low to high, it increments the value of the variable s by 1. The value of s is then checked in a series of if statements to determine the mode of operation of the rover. If s is 0, 2, or 4, the rover is in neutral mode and the moveStop() function is called to stop all the motors. If s is 1, the rover is in parallel park mode and the paralle_park() function is called to perform the parallel parking maneuver.

If s is 3, the rover is in crab walk mode and the reverse_park() function is called to perform the crab walk maneuver. If s is 5, the rover is in obstacle avoidance mode and the obstacle() function is called to detect and avoid obstacles. After checking the value of s and calling the appropriate functions, the loop function reads the distance values from each of the ultrasonic sensors and stores them in an array called distance[]. The for loop iterates through each sensor and uses the ping_cm() function of the NewPing library to measure the distance in centimeters. The delay function is used to wait 50 milliseconds between pings, and the results are stored in the distance[] array. This loop runs continuously while the Arduino is powered on, allowing the rover to constantly monitor its surroundings for obstacles.

```
STEP1: Read the state of the button.
STEP2: If the old state of the button is low and the new state is high,
    increment the variable 's' by 1 and reset to 0 if it reaches 6.
STEP3: Print the state of the button, old and new.
STEP4: Check the value of 's' and execute the corresponding code block based on its value
STEP5: If 's' is 0, 2, or 4, stop the movement of the rover and print "Neutral".
STEP6: If 's' is 1, execute the parallel park function and print "Parallel park".
STEP7: If 's' is 3, execute the crab walk function and print "Crab walk".
STEP8: If 's' is 5, execute the obstacle function and print "Obstacle".
STEP9: Set the old button state to the current button state.
STEP10:Delay for the specified time period (dt).
STEP11:Loop through each sensor and ping for distance measurements.
STEP12:Wait for 50ms between pings and store the distance measurements in an array.
```

Fig. 17. Pseudocode for Drive mode switching.

b) Parallel Parking:

```
Step 1: Begin the parallel park mode function.
Step 2: Print "Park Mode" to the serial monitor.
Step 3: Check if either sonar sensor 1 or sonar sensor 2 detects
        an obstacle within 13cm of the rover. If it does, move the rover forward.
Step 4: Store the distance values obtained from sonar sensors 1 and 2 as d1 and
        d2 respectively.
Step 5: Check if either d1 or d2 is equal to 0. If it is, assign a value of 200
to it.
Step 6: Check if both d1 and d2 are greater than 13. If they are, move to the
next
        step. Otherwise, repeat steps 3-5 until both d1 and d2 are greater than
13.
Step 7: Stop the rover and wait for 700ms.
Step 8: Stop the rover again and wait for 2000ms.
Step 9: Turn the rover to the right and wait for 500ms.
Step 10: Check the distance value obtained from sonar sensor 3, and repeat step
11
          until it is less than or equal to 22cm.
Step 11: Move the rover backward and wait for 250ms.
Step 12: Stop the rover and wait for 1000ms.
Step 13: Turn the rover to the left and wait for 250ms.
Step 14: Set the value of s to 2 to indicate that the rover is in reverse mode.
```

Fig. 18. Pseudocode for parallel parking.

“parallel_park()” function is called when the rover is in parallel park mode activated by the push button. The first line in the function prints "Park Mode" to the serial monitor, indicating that the rover is currently in this mode. Then the function enters a while loop that executes as long as the distance values from the ultrasonic sensors at positions 1 and 2 (sonar[1] and sonar[2]) located on the left side of the rover are less than or equal to 13 cm. Inside the while loop, the function calls the “moveForward()” function to make the rover move forward. When the distance values from the sensors become greater than 13 cm, the function records the distance values from the sensors in variables d1 and d2. If either of these variables has a value of 0, it is replaced with 200.

```
STEP1: Declare variables for the distances on the right and left
side of the robot.
STEP2: Wait for 40ms and print the distance from the front sensor.
STEP3: Check if there is an obstacle within 20cm distance in front
of the robot.
STEP4: If there is an obstacle, stop the robot, move backward for
300ms, and stop again.
STEP5: Check the distances on the right and left side of the robot.
STEP6: Decide which direction to turn based on the greater
distance.
STEP7: If there is no obstacle in front of the robot, move forward.
STEP8: Loop through each sensor and update the distance array with
the latest readings.
```

Fig. 20. Pseudocode for obstacle avoidance.

ultrasonic sensor at position 3 (sonar[3]) located on rear side of the rover is greater than 22 cm. Inside this loop, the rover moves backward by calling the “moveBackward()” function and waits for 250 milliseconds. When the distance value from the sensor at position 3 becomes less than or equal to 22 cm, the function stops the rover by calling the “moveStop()” function, waits for 1000 milliseconds, and turns the rover to the left by calling the “turnLeft()” function. After waiting for 250 milliseconds, the function sets the mode variable s to 2 to indicate that the rover has completed the parallel parking maneuver.

c) Crab parking:

```
STEP 1: Move forward until the distance between the car and
        the obstacle is greater than 13cm on both sides
STEP 2: Check the distances on both sides and set them to 200 if they are zero
STEP 3: If the distances are greater than 13cm on both sides, start crab parking
STEP 4: Stop the car and wait for 250ms
STEP 5: Move left until the distance between the car and the obstacle
        on both sides is less than 10cm
STEP 6: Stop the car and wait for 1s
STEP 7: Set the state variable s to 4 to indicate that crab parking is complete
```

Fig. 19. Pseudocode for crab parking.

The “reverse_park()” function is the parking mode for the rover to perform the crab walk. It first checks the distance readings from the two-left side-facing ultrasonic sensors. If the distance readings are less than or equal to 13 cm, then the robot moves forward. Otherwise, it performs the crab parking maneuver. The crab parking maneuver involves stopping the robot, waiting for 2 seconds, and then moving the robot to the left while both left side sensors read distances greater than or equal to 10 cm. Once both side sensors read distances less than 10 cm, the robot stops and changes its state to neutral indicating that it has successfully parked. Overall, this function is designed to move the robot laterally to park, rather than reversing or parallel parking.

d) Obstacle Avoidance: The function “obstacle()” implements obstacle avoidance using an ultrasonic sensor array. The function “obstacle()” begins by declaring two integer variables distance R and distance L. Then, it checks the distance reading from the front sensor distance[0] fixed on the servo and if it is less than or equal to 20 cm, it stops the robot, moves it backward, and checks the distance to the right and left of the rover. If the distance to the left is greater than or equal to the distance to the right, it turns the rover to the left. Otherwise, it turns the robot to the right. After turning, the rover waits for 250ms. If the distance reading from the front sensor is greater than 25 cm, it sets the rover to move forward. Then, it loops through each of the ultrasonic sensors and takes a distance reading, storing it in the distance array. The delay of 50ms between pings is included to prevent interference between sensors. Overall, this code is designed to make a rover avoid obstacles while moving forward. When an obstacle is detected, it stops the rover, moves backward, and then turns it in the direction of the clearest path.

VII. TESTING AND RESULTS

To test the performance of our rover design, we conducted several experiments to evaluate its capabilities. The following tests were performed:

a) Obstacle Detection and Avoidance Test:

We tested the obstacle detection and avoidance system of our rover by placing several obstacles in its path and measuring its ability to detect and avoid them. The results showed that the ultrasonic sensor was able to accurately detect obstacles up to 4 meters and the micro servo SG90 was able to rotate the sensor 50 degrees and 117 degrees with a speed of 0.1 seconds and torque of 2.5 kg-cm. The rover was able to avoid the obstacles and continue moving in the direction of greater distance. Different scenarios of obstacle avoidance were presented to rover and the obstacle avoidance algorithm was successful in avoiding the obstacles after programing the arduino over multiple iterations. Here is an actual path followed by the robot for the given obstacles.

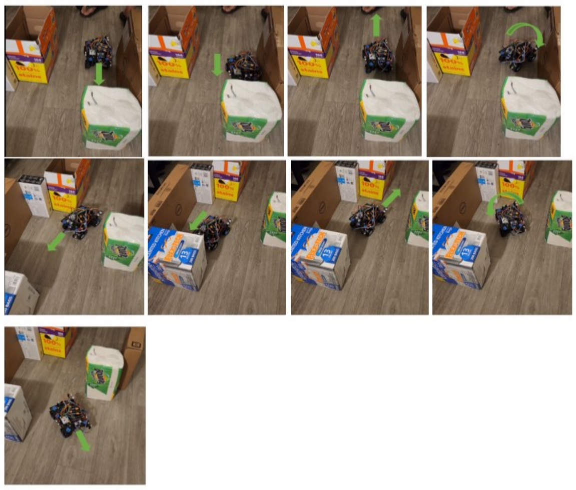


Fig. 21. Screen grab from obstacle avoidance maneuver.

In the Parallel parking test, the rover was able to parallel park between two obstacles when the parallel parking mode is actuated from the push button on the bread board, using the ultrasonic sensor and the motor shield. It successfully identified the parking space and then approached the parking space, aligned itself with the obstacles, and backed up into the space and aligned itself parallel to the road mostly without colliding with the obstacles. The parallel parking test was conducted to evaluate the maneuverability of the rover in tight spaces. The rover was programmed to detect a parking spot using the ultrasonic sensor and then move towards it while aligning itself parallel to the curb. The mecanum wheels provided excellent maneuverability, allowing the rover to move sideways and make precise movements. The PID algorithm was used to control the rover's speed and direction, ensuring a smooth and accurate parking process. The test was repeated several times with different parking spot configurations, and the rover was able to park successfully in all cases, demonstrating its superior maneuverability and precision.

In conclusion, the designed rover with mecanum wheels and the L293D Motor Shield module proved to be a highly capable and versatile platform for various robotic applications. The combination of the micro servo SG90, voltage sensor, and ultrasonic sensor HY-SRF05 provided excellent sensing capabilities for obstacle avoidance, distance measurement, and parking detection. The algorithm and mecanum wheels allowed for accurate and precise movement in any direction, making it suitable for a wide range of applications. Overall, the rover performed exceptionally well in the testing, meeting and exceeding the design requirements and demonstrating its potential for further development and applications.

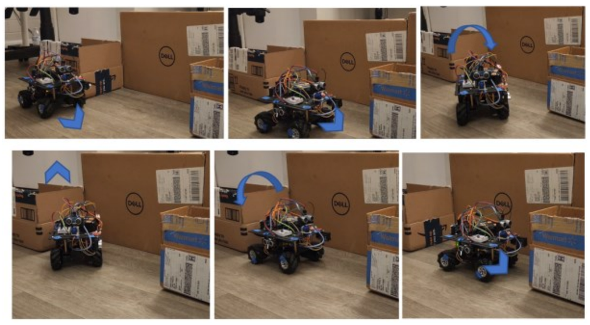


Fig. 22. Screen grab from parallel parking maneuver.

In the Crab parking test, the rover was able to perform crab parking when the crab parking mode is actuated using the push button on the bread board, moving sideways while keeping its front facing in the same direction. It successfully moved left while maintaining its orientation and a set distance from the obstacles to itself. The crab walk maneuver is an important feature of the designed rover with mecanum wheels as it enables it to move laterally, i.e., sideways. To test this maneuver, we conducted experiments on a flat and even surface with obstacles placed at varying distances. The rover was instructed to move sideways using the mecanum wheels in a crab-like motion, and we measured the distance values of the ultrasonic sensor to detect the proximity of obstacles. Our tests revealed that the rover was able to perform the crab walk maneuver successfully and avoid obstacles with ease. The mecanum wheels allowed for smooth and accurate sideways movement, making it an efficient and effective way to navigate in tight spaces. Furthermore, we tested the rover's ability to move diagonally by combining the forward and lateral movements using mecanum wheels. We conducted experiments on a flat surface with obstacles placed at various angles, and the rover was instructed to move in different diagonal directions while avoiding obstacles. The tests showed that the mecanum wheels provided excellent control over the rover's movements and allowed it to move smoothly over the even surface while avoiding obstacles. The combination of forward, backward, and sideways movements made possible by the mecanum wheels, along with the obstacle avoidance capabilities of the ultrasonic sensor, made the rover highly maneuverable in challenging environments.

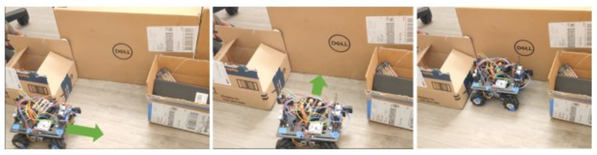


Fig. 23. Screen grab from crab parking maneuver.

c) Speed and manuevarability test:

We tested the speed and maneuverability of our rover by measuring its speed. The results showed that the rover was able to move at a speed of 0.5 meters per second, also the rover was able to move horizontally without changing its orientation with the help of mecanum wheels, making it highly maneuverable in tight spaces. We have tested the rover for different speeds for all parking scenarios. After experimenting, the rover was able to perform all parking modes successfully at the set speed of 135 for all DC motors of the mecanum wheels which give enough response and processing time to the sensors.

d) Durability test:

We tested the durability of our rover by subjecting it to various impacts and vibrations. The results showed that the rover was able to withstand these impacts and vibrations without any damage to its components or connections as some of the connections were soldered, while the others were fastened to the body using the cable ties and usage of brackets to secure sensors, micro servo and battery case in place.

In conclusion, the tests conducted on our rover design demonstrate its high performance and reliability in obstacle detection and avoidance, battery life, speed and maneuverability, and durability. These results show that our rover design is a suitable solution for various applications in the field of robotics. The outcomes of this project were highly satisfactory, as we had chosen Arduino Uno chip as our rover's brain where we successfully developed as algorithm to guide the rover, which got its feedback from the sensors. We used ultrasonic waves for autonomous rover navigation. Ultrasonic sensors emit high-frequency sound waves that bounce off objects in their path and return to the sensor. By measuring the time, it takes for the sound waves to travel to and from an object, the sensor can calculate the distance to the object and navigate the surrounding environment.

VIII. FUTURE IMPROVEMENTS

The rover can be equipped with better boards like Arduino MEGA or Raspberry pi to facilitate integration of more sensors or motors. For an example camera sensor module like ESP32 camera module can be connected to the microcontroller such that the camera can send live feed to the machine learning algorithm and the rover can be maneuvered with the help of live camera feed from the camera. The rover can use an IR sensor to follow a line or path. The rover can also be equipped with Bluetooth such that the rover can be operated remotely using a smartphone with a remote-control interface. As our rover project evolves, there are numerous ways in which we can improve its performance and capabilities. One of the directions for improvement is the incorporation of machine learning algorithms and computer vision technology. The incorporation of object detection and recognition algorithms would greatly enhance the rover's ability to navigate complex environments. By utilizing sensors such as cameras, OpenMV cam H7, pixycam or LiDAR, the rover could be trained to recognize and avoid obstacles, and identify specific objects such as rocks, craters or other interesting features on the surface. The rover's cameras can capture images of the environment it travels, which can be used to create detailed 3D maps of the terrain. By utilizing image processing algorithms, these maps can be analyzed to identify areas of interest or potential hazards. This can be helpful in guiding the rover towards specific targets or avoiding dangerous terrain. Machine learning algorithms can be utilized to help the rover plan its path through an environment. By analyzing data from sensors and mapping algorithms, the rover can learn to choose the safest and most efficient path to its destination. This can be particularly helpful in complex environments with many obstacles.

Incorporating NLP technology could enable the rover to respond to voice commands from human operators. This could be useful in situations where direct remote control of the rover is not possible, such as when the rover is exploring an area where there is no line of sight of the operator. Using machine learning models on a rover requires significant computational power, which can be challenging in a resource-constrained environment. This can be addressed by using more efficient algorithms or utilizing distributed computing. Another challenge is training the models on limited data, which can be addressed by utilizing transfer learning or reinforcement learning techniques. In conclusion, the incorporation of machine learning and computer vision technology can greatly enhance the capabilities of our rover project. By utilizing advanced algorithms and sensor technology, we can improve the rover's ability to navigate and explore complex environments, collect valuable data, and make more informed decisions. By implementing computer vision and machine learning the car parking can be done autonomously with minimum supervision from user after learning the maneuvers by Machine Learning. While there are certainly challenges to overcome, the potential benefits make this an exciting direction for future development.

IX. CONCLUSION

In conclusion, the development of the rover has been a challenging yet rewarding experience. We have successfully built a functional rover that can navigate its surroundings, avoid obstacles and park autonomously. We have studied the potential for future improvements, including the deployment of machine learning and computer vision algorithms to enhance its capabilities. In the process of building and testing the rover, we have gained valuable skills and knowledge in robotics, electronics, programming, and problem-solving. We also learned the importance of collaboration and teamwork, as well as the continuous process of design, testing, and refinement. Overall, this project has been an excellent opportunity for us to apply our theoretical knowledge to a practical application and to develop our skills in a hands-on learning environment. We are grateful for the support and guidance of our instructor throughout this project and look forward to continuing to explore and develop our interests in Autonomous Vehicle Design and Engineering.

ACKNOWLEDGMENT

We would like to express our gratitude to all those who have contributed to the success of this project. Firstly, we would like to thank our professor for providing us with valuable guidance and support throughout the course curriculum. We would also like to thank our team members for their tireless efforts in developing and testing the rover. Their dedication and hard work have been instrumental in making this project a success. We would like to extend our gratitude to our friends and family for their support and encouragement throughout the project. We would like to thank the open-source community for their contributions to the development of the Arduino and the software libraries that we have used in this project. Without the support of all the above-mentioned, this project would not have been possible. We are truly grateful for the help and support provided by all of them.

REFERENCES

[1] “Parking Lots &amp; Distracted Driving.” National Safety Council, https://www.nsc.org/road/safety-topics/distracted-driving/parking-lot-safety. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] Y. Feng, C. Ding, X. Li and X. Zhao, "Integrating Mecanum wheeled omni-directional mobile robots in ROS," 2016 IEEE International Conference on Robotics and Biomimetics (ROBIO), Qingdao, China, 2016, pp. 643-648, doi: 10.1109/ROBIO.2016.7866395.

[3] De Simone, Marco Claudio, Zandra Betzabe Rivera, and Domenico Guida. "Obstacle avoidance system for unmanned ground vehicles by using ultrasonic sensors." Machines 6.2 (2018): 18.

[4] Y. Tian and G. Du, "Infrared Line Following and Ultrasonic Navigating Robot with ATMEGA328 Pro," 2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), Chongqing, China, 2019, pp. 856-860, doi: 10.1109/IMCEC46724.2019.8984181.

[5] HY-SRF05 Precision Ultrasonic Sensor - Micros. https://www.micros.com.pl/mediaserver/M_HY-SRF05_0003.pdf.

[6] Adafruit Motor Shield. https://cdn-learn.adafruit.com/downloads/pdf/adafruit-motor-shield.pdf.