Omni-Directional Bluetooth Controlled Car

End-Term Project Report
For

Internet of Things Projects using Python (CSE 4110)

Submitted by

Ashwini Kumar Behera	1941012390
Ratikanta Sahoo	1941012547
Amitosh Mahapatra	1941012578
Sushovan Kar	1941012580

B. Tech. 7th Semester, CSE

Project Supervisor

Dr. Hara Prasada Tripathy	(Assistant Professor)
Dr. Siddharth Sahany	(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Institute of Technical Education and Research
SIKSHA 'O' ANUSANDHAN DEEMED TO BE UNIVERSITY
Bhubaneswar, Odisha, India.

(January, 2023)

DECLARATION

We certify that

a. The work contained in this report is original and has been done by us under the guidance of our supervisor(s).

b. The work has not been submitted to any other Institute for any degree or diploma.

c. We have followed the guidelines provided by the Department in preparing the report.

d. Whenever we have used materials (data, theoretical analysis, figures, and text) from other sources, we have given due credit to them by citing them in the text of the report and giving their details in the reference.

Name of the Student	Registration Number	Signature
Ashwini Kumar Behera	1941012390	
Ratikanta Sahoo	1941012547	
Amitosh Mahapatra	1941012578	
Sushovan Kar	1941012580	

REPORT APPROVAL

The report entitled

Omni-Directional Bluetooth Controlled Car

by

Ashwini Kumar Behera	**1941012390**
Ratikanta Sahoo	**1941012547**
Amitosh Mahapatra	**1941012578**
Sushovan Kar	**1941012580**

is approved for

Internet of Things Projects using Python (CSE 4110)

in

Computer Science and Engineering

Dr. Hara Prasada Tripathy	**Dr. Siddharth Sahany**
(Assistant Professor)	**(Assistant Professor)**

Date:

Place:

CONTENTS

ABSTRACT

Mecanum wheels offer a means to implement omni-directional translational motion in vehicle platform. As the wheel rotates, rollers around its circumference create a force directed at an angle relative to the plane of rotation. Various combinations of wheel rotational velocities can provide a net force in any direction. This provides enhanced maneuverability in close quarters, but is often limited to smoother surfaces as the wheels require many moving rollers that can become easily obstructed by debris. Applications of this model run form small hobby kits up to heavy duty industrial vehicles.

In this Project we are using Bluetooth wireless technology to control our robot car which is a very simple communications system. The remote in this project is an android device which has Bluetooth feature built in. The user has to install an application on his/her mobile and turn on the Bluetooth in mobile phone. User can perform various actions like moving Forward, Backward, move Left and move Right using commands that are sent from the android mobile. The Bluetooth is a serial communication medium through which we can connect two devices wirelessly. Here we have used a Bluetooth module in our robot car which gets connected to the phone's Bluetooth, that allows us to communicate and allows to take command over it. The task of controlling the car is done by the Raspberry Pi Pico. Raspberry Pi Pico played a major role in the robotic section and has made it easier to convert digital and analog signal to physical movements. The project is Bluetooth based because it gives us wider range control and more efficiency. It also gives us the advantage of changing the remote anytime, meaning that we can use any android devices including phones, tablets, computers. Physical barriers like walls, doors, etc. do not effect in controlling the car

Chapter 1: Introduction

1.1. Motivation

The motivation for an omni-directional Bluetooth controlled car is likely to provide a convenient and efficient way to control the movement of the car remotely, but with the added benefits of the unique design of mecanum wheels. Mecanum wheels have a series of rollers mounted at an angle on the circumference of the wheel, which allows the wheel to move in any direction. This design allows the car to move laterally or diagonally, as well as forward and backward, providing even greater flexibility and maneuverability than an omni-directional car. Additionally, using Bluetooth technology to control the car allows for easy and convenient remote control. Bluetooth technology allows for wireless communication between the car and a device such as a smartphone or tablet, allowing for easy control of the car's movement in any direction. Additionally, an omni-directional car can move in any direction, providing greater flexibility and maneuverability. This is particularly useful in tight spaces or for tasks that require precise movement. The mecanum wheel based Bluetooth controlled car is particularly useful in applications where precise movement and maneuverability are required, such as in industrial and research settings.

1.2. Design Goals

1.2.1 Purpose

An omni-directional Bluetooth controlled car is to provide a convenient and efficient way to remotely control the movement of the car. The omni-directional capabilities of the car allow for greater flexibility and maneuverability, which is useful in tasks that require precise movement and navigation in tight spaces.

Additionally, the Bluetooth control allows for wireless communication between the car and a device such as a smartphone or tablet, making it easy to operate and control the car's movement. The remote control feature of the car can be used for a variety of applications, including but not limited to:

- Remote inspection of hard-to-reach areas

- Surveying and mapping
- Remote monitoring and surveillance
- Robotics research and development
- Automated warehouse and logistics
- Remotely controlled transportation in hazardous environments.

The purpose of the car is to provide a practical, efficient and versatile solution for tasks that require remote control and precise movement.

1.2.2 Scope

An omni-directional Bluetooth controlled car can be quite broad, as the car's capabilities and potential applications are diverse. Some of the main areas in which an omni-directional Bluetooth controlled car can be used include:

- Industrial and manufacturing: An omni-directional Bluetooth controlled car can be used for tasks such as remote inspection of hard-to-reach areas, automated warehouse and logistics, and remotely controlled transportation in hazardous environments.
- Robotics research and development: The car's capabilities can be used to test and develop new control algorithms, sensors, and other technologies for autonomous robots.
- Surveying and mapping: The car's ability to move in any direction can be used for tasks such as creating 3D maps of indoor spaces, or for mapping out difficult terrains.
- Remote monitoring and surveillance: An omni-directional Bluetooth controlled car can be used for tasks such as monitoring wildlife, inspecting power plants, or surveying disaster areas.
- Remote controlled transportation in hazardous environments: The car's ability to move in any direction allows it to navigate challenging terrains, such as in mines, oil rigs, and other hazardous environments.

- Entertainment: The car can be used in remote-controlled car racing, and other activities that require precise movement and remote control.

Overall, the scope of an omni-directional Bluetooth controlled car is quite broad, as it can be used in a variety of applications that require remote control and precise movement.

1.2.3 Applicability

Omni-directional Bluetooth controlled car is quite broad and varied, as it can be used in a wide range of applications that require remote control and precise movement. Some of the main areas of applicability include:

- Robotics research and development: The car's capabilities can be used to test and develop new control algorithms, sensors, and other technologies for autonomous robots.
- Remote monitoring and surveillance: An omni-directional Bluetooth controlled car can be used for tasks such as monitoring wildlife, inspecting power plants, or surveying disaster areas.
- Search and Rescue: The car can be used to search inaccessible and dangerous places, such as mines, caves, or collapsed buildings.
- Entertainment: The car can be used in remote-controlled car racing, and other activities that require precise movement and remote control.
- Agriculture: The car can be used for remote monitoring, inspecting, and mapping of crops and fields.

Overall, the applicability of an omni-directional Bluetooth controlled car is quite broad, as it can be used in a variety of applications that require remote control and precise movement.

1.3. Problem Statement

The current methods for remote control of vehicles in industrial and research applications are limited in their ability to navigate in tight spaces and maneuver in multiple directions. This can lead to inefficiencies and difficulties in tasks such as remote inspection,

surveying and mapping, and transportation in hazardous environments. The development of an omni-directional Bluetooth controlled car that can move in any direction and be controlled remotely via Bluetooth technology, would greatly improve the efficiency and versatility of these tasks.

1.4. Organization of the Report

An omni-directional Bluetooth controlled car can vary depending on the specific information that needs to be included, but a general outline could include the following sections:

Chapter 1: Introduction

This section should provide an overview of the problem statement, the purpose of the report, and the research question(s) that the report aims to answer.

Chapter 2: Literature Survey

This section should provide an overview of the current state of the field, including relevant research and literature on omni-directional vehicles, Bluetooth control, and related topics.

Chapter 3: Design Scheme

Chapter 4: Testing, Analysis, and Evaluation

This section should describe the methods and techniques used to design, build, and test the omni-directional Bluetooth controlled car. This should include details on the materials and components used, any software or algorithms developed, and the testing procedures used to evaluate the car's performance.

This section should present the results of the testing and analysis of the car's performance, including any data collected, and the conclusions that can be drawn from the results.

Chapter 5: Socio-Economic Issues associated with the Project

Chapter 6: Engineering Tools and Standards used in the Project

Chapter 7: Problems, faults, bugs, challenges

Chapter 2: Literature Survey

2.1. Background work done so far

The Internet of Things (IoT) is an important topic in technology industry, policy, and engineering circles and has become headline news in both the specialty press and the popular media. This technology is embodied in a wide spectrum of networked products, systems, and sensors, which take advantage of advancements in computing power, electronics miniaturization, and network interconnections to offer new capabilities not previously possible. An abundance of conferences, reports, and news articles discuss and debate the prospective impact of the ―IoT revolution‖—from new market opportunities and business models to concerns about security, privacy, and technical interoperability. It is basically connecting the electronic items to the cloud and in other words internet in order to make them more useful and worthful.

The large-scale implementation of IoT devices promises to transform many aspects of the way we live. For consumers, new IoT products like Internet-enabled appliances, home automation components, there are many uses to household people and energy management devices are moving us toward a vision of the ―smart home‖, offering more security and energy efficiency. Other personal IoT devices like wearable fitness and health monitoring devices and network enabled medical devices are transforming the way healthcare services are delivered. This technology promises to be beneficial for people with disabilities and the elderly, enabling improved levels of independence and quality of life at a reasonable cost. IoT systems like networked vehicles, intelligent traffic systems, and sensors embedded in roads and bridges move us closer to the idea of ―smart cities‖, which help minimize congestion and energy consumption. IoT technology offers the possibility to transform agriculture, industry, and energy production and distribution by increasing the availability of information along the value chain of production using networked sensors. There are lots of sensors in this electronic world using iot we can make them more worthful than ever. However, IoT raises many issues and challenges that need to be considered and addressed in order for potential benefits to be realized.

Chapter 3: Design Scheme

3.1. System Design

The system design of an omni-directional Bluetooth controlled car would typically involve several key components:

- Mechanical design: This would include the design and construction of the car's chassis, as well as the integration of the omni-directional wheels, such as mecanum wheels, which allow the car to move in any direction.

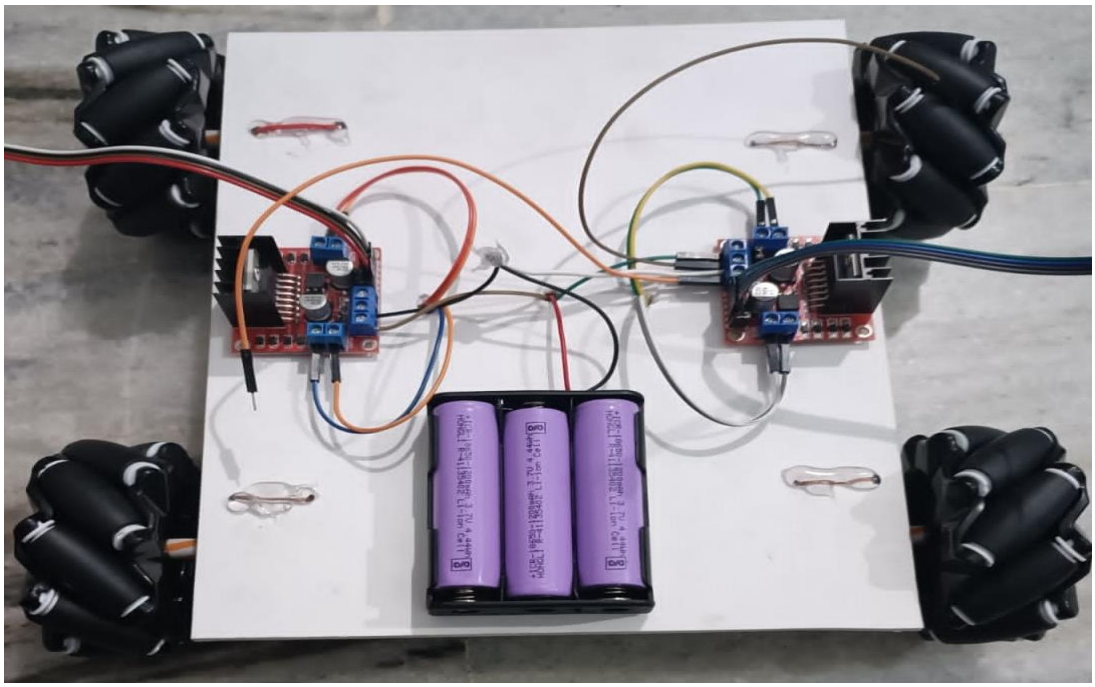

Figure 3.1.1: l298n motor module connection with mecanum wheels

- Control system: This would include the design and implementation of the control algorithms and software that govern the car's movement, such as the control of the wheel motors and the car's overall movement. It would also include the design of the communication system, such as the Bluetooth module, to enable remote control of the car.

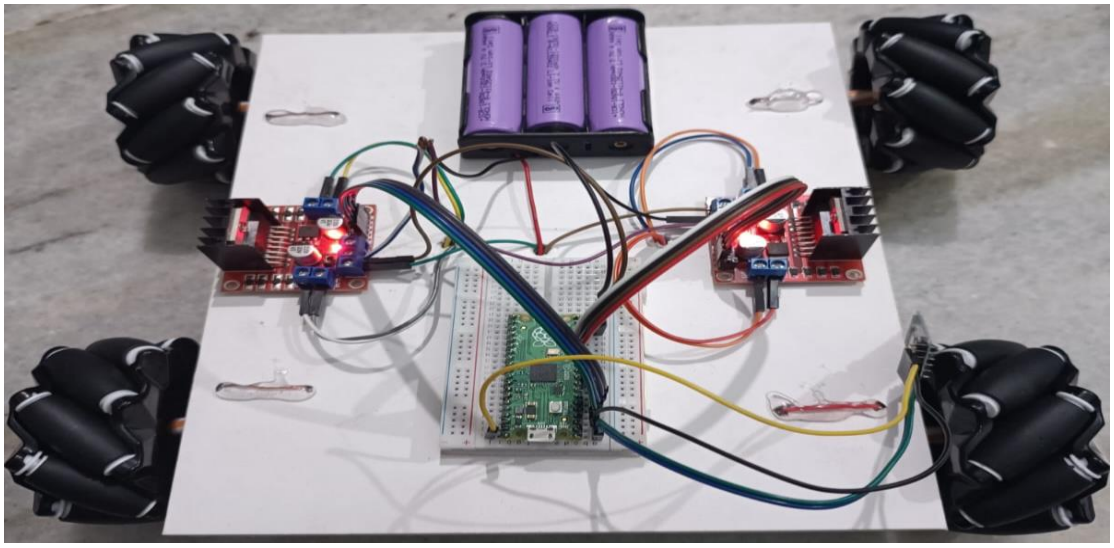

Figure 3.1.2: Raspberry Pi Pico connected to the l298n motor module

- Power: The design of power system, including the battery and charging circuit, to ensure the car has sufficient power to operate.
- User interface: This would include the design and development of a user interface, such as a smartphone app, to enable the user to control the car and receive feedback from the car's sensors.

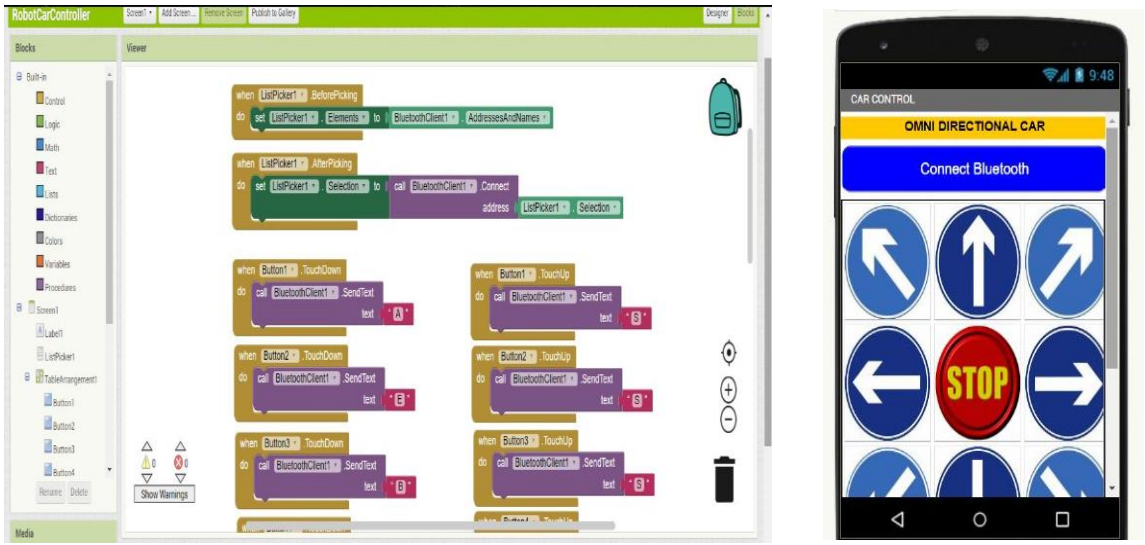

Figure 3.1.3: Omni Directional Car application

- Safety Features: This would include the design of safety features like obstacle detection and avoidance system, emergency stop, and other features to ensure the safety of the car and the people around it.

The exact details of the system design will vary depending on the specific requirements of the application, but these are the general components that would be involved in the design of an omni-directional Bluetooth controlled car.

3.2. Architecture

Here basically we will show the block diagram if our project which basically show how we can connect each component with each other with the power supply and the pico and how can they be connected show that the can generate a secure connection to transform instruction with each other and work smoothly. Mainly we can denote the block diagram as a blueprint of the project. So the diagram is shown below:

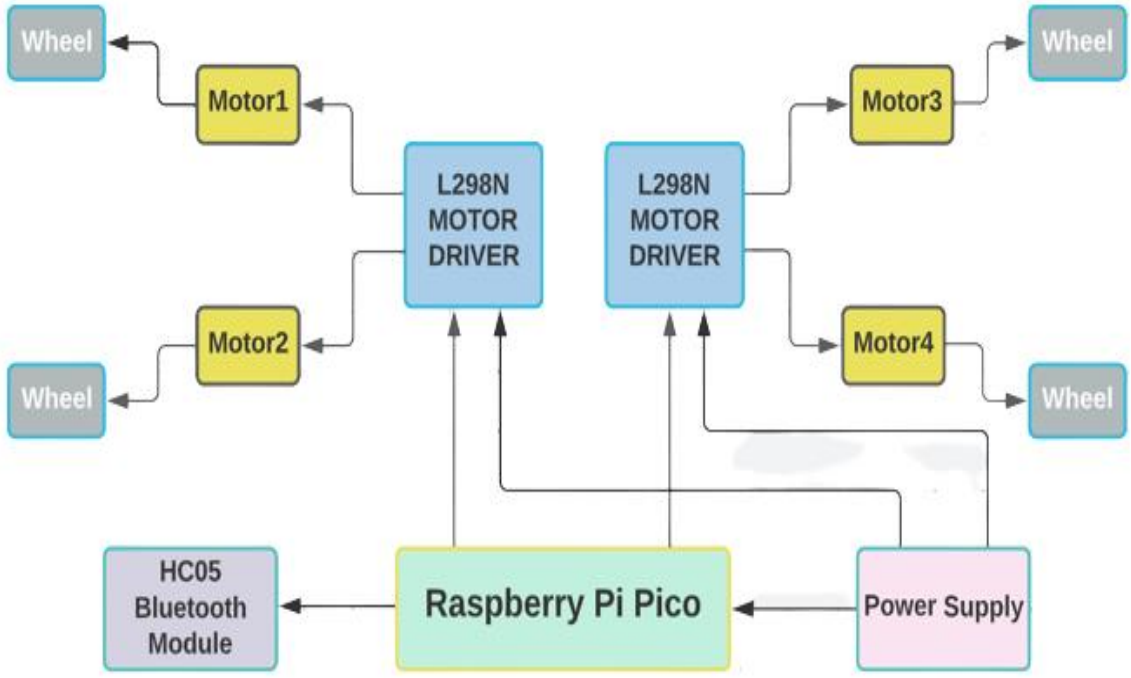

Figure 3.2.1: Block Diagram

3.3. Component Design

We did not do any component design specially for our project work because all the sensors, drivers and other components are easily available in the market.

3.4. Implementation

The Project name “Omni-Directional Bluetooth Controlled Car” has been made by following some major steps to implement the items with the raspberry pi pico which helps us from several damage which can be occurs at the time of implementation if we will not follow some specific steps.

And the All figures in sequence step by step:

Figure 3.4.1: Mecanum Wheel with motor

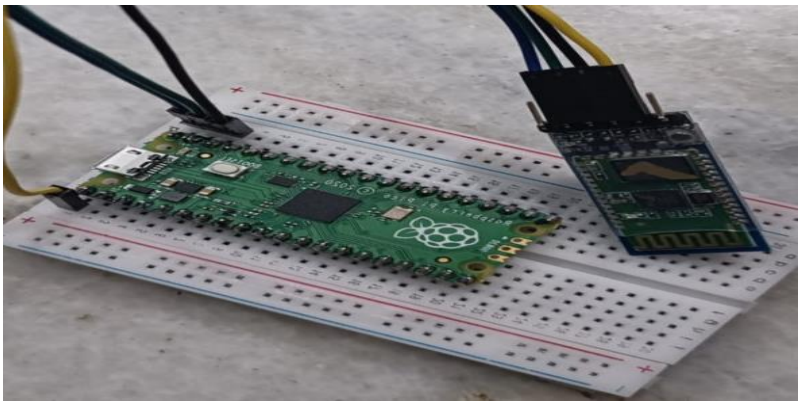

Figure 3.4.2: HC-05 Pinned in Raspberry Pi Pico

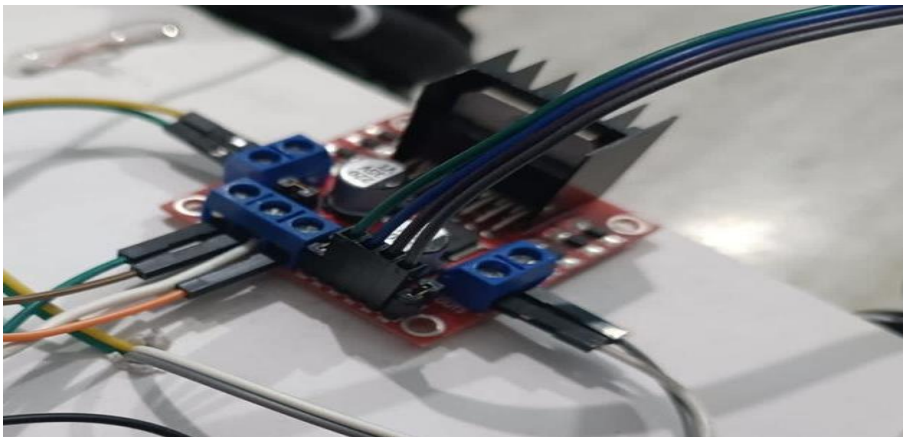

Figure 3.4.3: L298N Motor driver connected with Pico

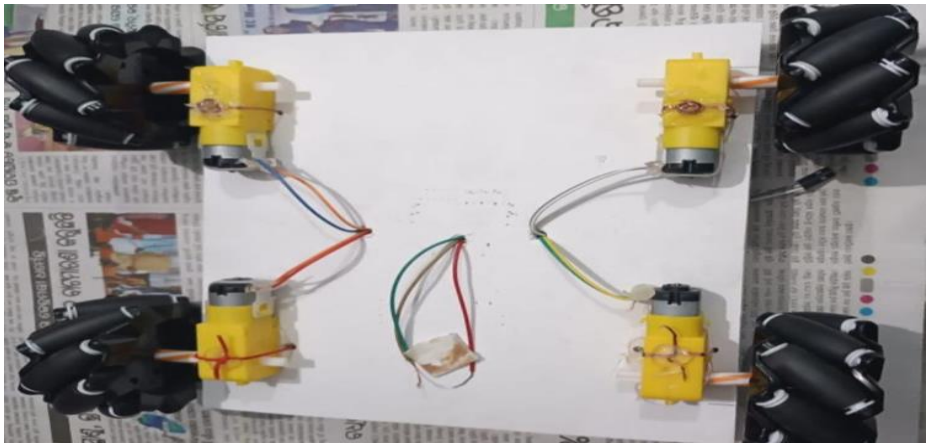

Figure 3.4.4: Position of all four wheels

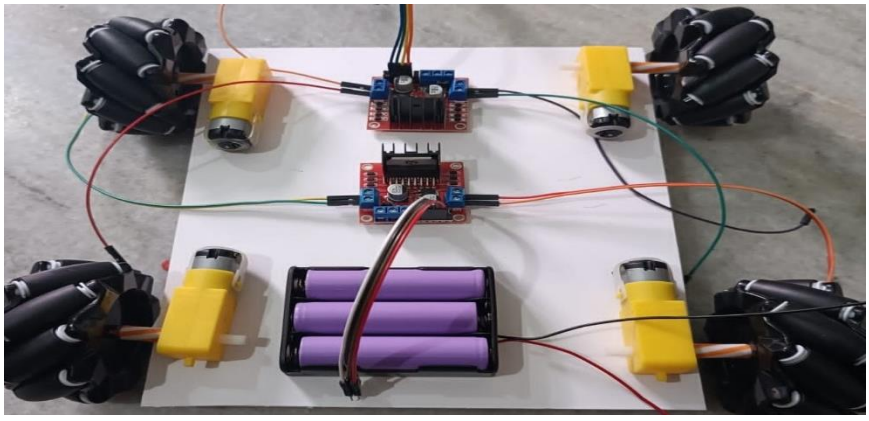

Figure 3.4.5: Three 3.7 V Battery giving the power to the Motor

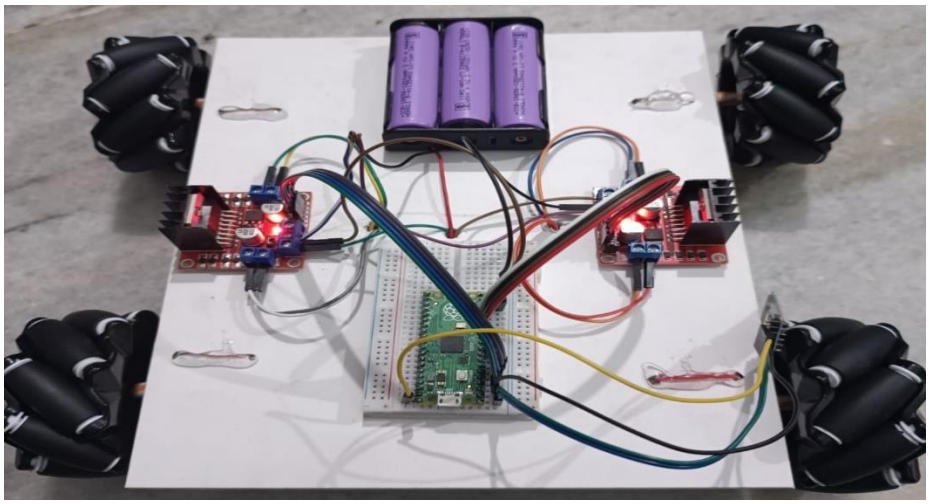

Figure 3.4.6: Final Prototype of the Project

3.5. Design Evolution

At first the project “Omni-Directional Bluetooth Controlled Car” was done by the use of IR Sensor but there was no implementation of any Internet Of Things that’s why we, all the member of our team decided to implement some IoT in this project so we decided to implement a Bluetooth module and create a mobile application to run the car wirelessly and here basically we give the instruction using the mobile application and using that instruction we will move the car according to us. And here we can also implement wifi module at the place of Bluetooth and can control the car with the help of IP address.

Chapter 4: Testing, Analysis, and Evaluation

4.1. Testing Criterions

4.1.1. Relevance

An omni-directional Bluetooth controlled car can be relevant in a number of ways, depending on the specific application or use case. Some potential benefits of such a car include:

- Increased maneuverability: The use of omni-directional wheels or other similar technologies can enable the car to move in any direction without the need for turning. This can be useful in situations where space is limited or where the car needs to navigate through tight spaces or around obstacles.
- Improved control: Bluetooth-controlled cars can be operated remotely, which can be useful in situations where the car needs to be operated in hazardous environments or in areas where human operators cannot easily access.
- Low cost: Bluetooth control systems can be relatively simple and inexpensive to implement, which can make them well-suited for use in small-scale autonomous cars. Additionally, Bluetooth technology is widely available, which makes it easy to control a car from a wide range of devices such as smartphones, tablets, and laptops.
- Easy to use: Bluetooth technology is user-friendly and easy to use, which makes it accessible to a wide range of users.
- Educational and research purposes: The car can be used for educational and research purposes such as teaching students about control systems, robotics, and other related topics.
- Autonomous transportation: Bluetooth controlled cars can be used for autonomous transportation in small scale, for example, in indoor environments, such as in malls, airports, and hospitals.

Overall, an omni-directional Bluetooth controlled car can be relevant in situations where maneuverability, cost-effectiveness, and ease of use are important considerations.

4.1.2. Effectiveness

An omni-directional Bluetooth controlled car will depend on a number of factors, including the specific design and implementation of the car, as well as the environment in which it is being used.

- One key factor that can affect the effectiveness of such a car is the control system. The car's ability to accurately and reliably respond to Bluetooth commands can be critical to its effectiveness in navigating through different environments.
- Another important factor that can affect the effectiveness of an omni-directional Bluetooth controlled car is the design of the omni-directional wheels or other similar technologies. The car's ability to move in any direction can be dependent on the quality and efficiency of these wheels, and the car's overall effectiveness may be limited by the capabilities of these wheels.
- The environment in which the car is being used can also have a significant impact on its effectiveness. For example, a Bluetooth controlled car may be less effective in outdoor environments, where it may be more difficult to maintain a reliable Bluetooth connection, or where the car may be more likely to encounter obstacles that it cannot navigate around. Indoor environments, on the other hand, can offer a more controlled environment with better access to power and wifi, making the car more effective.
- Overall, the effectiveness of an omni-directional Bluetooth controlled car will depend on a number of factors, including the quality of the control system, the design of the omni-directional wheels, and the environment in which the car is being used. While Bluetooth controlled cars have its own limitations, it can be an effective solution for specific use cases, such as indoor autonomous transportation.

4.1.3. Efficiency

The efficiency of an omni-directional Bluetooth controlled car, such as one using mecanum wheels, would depend on a few factors including:

- The quality and performance of the motors used to power the wheels. High-performance motors with high torque and low energy consumption would increase the efficiency of the car.
- The weight and design of the car. A lightweight car with a streamlined design would require less energy to move, thus increasing efficiency.
- The control system and Bluetooth communication. A well-designed control system with minimal latency in communication would increase the responsiveness and efficiency of the car.
- The surface or terrain the car is moving on. A smooth surface would require less energy to move on than a rough or uneven surface.

Overall, an omni-directional Bluetooth controlled car using mecanum wheels could be more efficient than traditional wheeled vehicles in certain situations, such as maneuvering in tight spaces or on smooth surfaces.

However, in general, remote-controlled cars are not as efficient as self-driving cars, since they need human interaction to be controlled.

Analysis:

An omni-directional Bluetooth controlled car is a type of remote-controlled car that can move in any direction, as opposed to traditional remote-controlled cars which can only move forward, backward, left, and right. This is achieved by incorporating multiple wheels or treads that can rotate in different directions, allowing for greater maneuverability and control. The car is controlled through a Bluetooth connection, typically using a smartphone or other mobile device as the remote control. The car can be programmed or controlled using an app or software that allows the user to control the car's speed and direction.

Evaluation:

The evaluation of an omni-directional Bluetooth controlled car would depend on various factors, including the car's design, performance, and functionality.

- Design: The car should have a sturdy and durable build with good maneuverability and control. The car should also have a sleek and attractive design, as well as a good balance of weight and size.
- Performance: The car should have a smooth and responsive control, with a good range and a strong Bluetooth connection. The car should also have a decent speed and be able to handle different terrains and surfaces.
- Functionality: The car should have a user-friendly interface and be easy to set up and use. The car should also have a variety of features such as obstacle avoidance, programmed movements, and the ability to interact with other devices.

Overall, an omni-directional Bluetooth controlled car can offer a lot of fun and excitement for users of all ages, but it's important to evaluate the design, performance, and functionality to ensure you get a good quality product.

Chapter 5: Socio-Economic Issues associated with the Project

5.1. Detailed Cost Analysis

5.1.1. Cost Analysis

The cost of an omni-directional Bluetooth controlled car can vary widely depending on the features and specifications of the vehicle. However, here are some general factors that can affect the cost:

- Size: Larger cars with more complex mechanics and features will generally be more expensive than smaller, simpler cars.
- Quality: Higher-quality materials and better craftsmanship will also drive up the cost of the car.
- Features: Additional features such as lights, cameras, and sensors will also increase the cost of the car.
- Brand: Wheels and motor from well-known and established brands will generally be more expensive than those from lesser-known brands.
- Quantity: If buying in bulk, the cost will be lower than buying just one unit.

A basic omni-directional Bluetooth controlled car can cost around 1500, while more advanced models with additional features can cost upwards of 2000 or more. It's important to compare prices and features before making a purchase to ensure that you are getting a good value for your money.

5.1.2. Bill of Materials

Sl. No.	Name of the Components	**Price** (in Rupees)
1	Raspberry Pi Pico × 1	375
2	Bluetooth module × 1	250
3	L298N motor drive × 2	280
4	Gear motor × 4	240
5	Mecanum wheel × 4	900
6	Li-ion battery × 3	185
7	Battery Holder	60
7	Switch × 1	5
8	Sun board × 3	120
9	Breadboard × 1	70
10	Jumper wires	……..

5.2. Safety issues

There are several safety issues to consider when using an omni-directional Bluetooth controlled car. Here are a few examples:

- Collision: The car can collide with people, pets, or other objects, potentially causing injury or damage.
- Battery safety: If the car uses a rechargeable battery, it is important to ensure that the battery is charged and handled properly to avoid fire or other hazards.
- Bluetooth interference: Bluetooth signals can be disrupted by other electronic devices, which can cause the car to lose control or behave unexpectedly.
- Range: If the car is operated too far away from the controller, it can be difficult to control and may wander into danger.

It is important to follow the manufacturer's instructions when using the car and to supervise children when they are using it. It's also important to use the car in an open space and avoid using it in crowded areas.

It's always important to be aware of your surroundings when operating an omni-directional Bluetooth controlled car, and to be prepared for unexpected events, such as loss of control or battery failure.

5.3. Global Impact

It is difficult to predict the global impact of an omni-directional Bluetooth controlled car as it would depend on the specific usage and application of the technology. However, here are a few potential impacts that could be considered:

- Convenience: An omni-directional Bluetooth controlled car can provide a level of convenience and flexibility, allowing people to remotely control the car to complete various tasks.
- Accessibility: These cars can be used in a variety of settings, including homes, offices, and industrial environments, providing greater accessibility to people with mobility impairments.
- Cost-effectiveness: The cost of these cars might be cheaper than traditional self-driving cars and they can be used in a variety of settings, increasing cost-effectiveness.
- Job displacement: Remote controlled cars could potentially automate some tasks and displace human workers.
- Cyber-security: As these cars are controlled by Bluetooth, the security of the Bluetooth connection and the device controlling the car should be considered.

Overall, the global impact of an omni-directional Bluetooth controlled car would depend on how the technology is implemented and used in various industries and settings.

5.4. Lifelong Learning

Lifelong learning from an omni-directional Bluetooth controlled car refers to the ability of the car to continuously gather and process data, and use it to improve its performance over time. An omni-directional Bluetooth controlled car, equipped with machine learning capabilities, can gather data on its environment, its usage and performance, and use it to improve its navigation, speed, and safety.

For example, the car could gather data on the most common routes it takes, and use that data to optimize its path and avoid obstacles. Additionally, the car could also learn from its past mistakes and use that knowledge to improve its decision making in the future.

Moreover, the car could also gather data on its usage and performance, and use it to predict maintenance needs or identify potential failures. Lifelong learning capabilities in an omni-directional Bluetooth controlled car can also be used in industries such as logistics, transportation, and delivery to optimize routes and improve the overall performance and efficiency.

Chapter 6: Engineering Tools and Standards used in the Project

RASPBERRY PI PICO

A Raspberry Pi Pico is a low-cost microcontroller device. Microcontrollers are tiny computers, but they tend to lack large volume storage and peripheral devices that you can plug in (for example, keyboards or monitors).

A Raspberry Pi Pico has GPIO pins, much like a Raspberry Pi computer, which means it can be used to control and receive input from a variety of electronic devices.

Figure 6.1: Raspberry Pi Pico

Features:

- RP2040 microcontroller chip designed by Raspberry Pi in the United Kingdom
- Dual-core Arm Cortex M0+ processor, flexible clock running up to 133 MHz
- 264kB of SRAM, and 2MB of on-board flash memory
- USB 1.1 with device and host support
- Low-power sleep and dormant modes
- Drag-and-drop programming using mass storage over USB
- 26 × multi-function GPIO pins
- × SPI, 2 × I2C, 2 × UART, 3 × 12-bit ADC, 16 × controllable PWM channels
- Accurate clock and timer on-chip
- Temperature sensor

- Accelerated floating-point libraries on-chip
- 8 × Programmable I/O (PIO) state machines for custom peripheral support
- 1 × Contains 1 × USB 1.1 controller and PHY, with host and device support
- Accurate clock and timer on-chip
- Low-Power sleep and dormant modes
- Accelerated integer and floating-point libraries on-chip
- Works with Windows, Mac, Linux machines and Raspberry Pi Computers
- Provides drag-and-drop programming using mass storage over USB

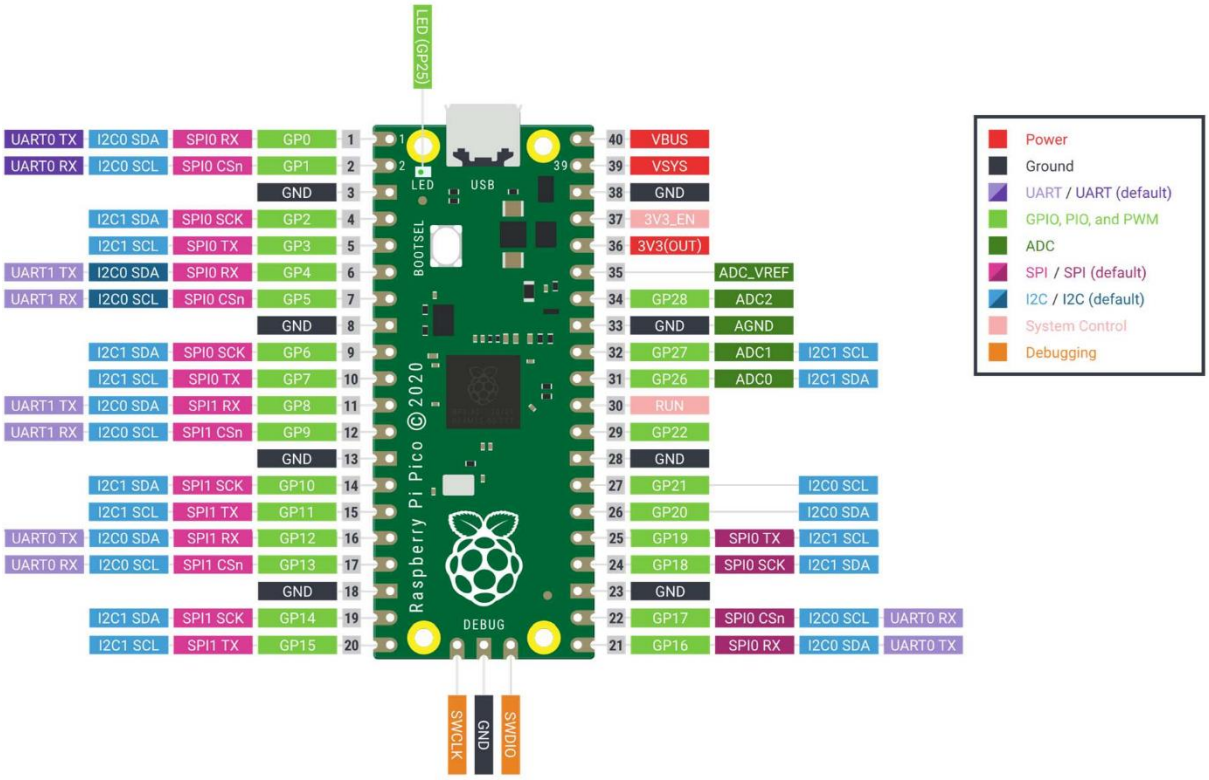

Figure 6.2: Raspberry Pi Pico pin diagram

Key Specifications:

- Microcontroller: RP2040 designed by Raspberry Pi in the UK
- Processor: Dual-Core Arm Cortex-M0+ processor, flexible clock running up to 133 MHz
- Input power: 1.8 - 5.5 V DC
- Operating temperature: -20°C to +85°C

- Dimensions: 51.0 x 21.0 mm
- Onboard Sensors: Temperature Sensor
- Memory: 264KB of on-chip internal SRAM and can support up to 16MB of off-chipFlash 2MB on-board QSPI Flash (Adafruit's Feather RP2040, features 16MB of storage)
- GPIO: It has 40 GPIO through-hole pins also with edge castellation
- 26 × multi-function 3.3V GPIO pins, which includes 3 analogue inputs (The Analog inputs is something other Raspberry Pi's lack. They use variable voltages to connect to devices like a potentiometer, joystick or a LDR)
- × SPI, 2 × I2C, 2 × UART, 3 × 12-bit ADC, 16 × controllable PWM channels
- 8 × Programmable I/O (PIO) state machines for custom peripheral support that can offload many kinds of timing-critical processes from the CPU

Theory of Operation:

Pico provides minimal (yet flexible) external circuitry to support the RP2040 chip: flash (Winbond W25Q16JV), crystal, power supplies and decoupling, and USB connector. The majority of the RP2040 microcontroller pins are brought to the user IO pins on the left and right edge of the board. Four RP2040 IO are used for internal functions - driving an LED, onboard Switched Mode Power Supply (SMPS) power control and sensing the system voltages. Pico has been designed to use either soldered 0.1" pin-headers (it is one 0.1" pitch wider than a standard 40-pin DIP package) or can be used as a surface mountable 'module', as the user IO pins are also castellated.

BLUETOOTH MODULE:

To communicate smartphone with HC-05 Bluetooth module, smartphone requires Bluetooth terminal application for transmitting and receiving data. You can find Bluetooth terminal applications for android and windows in respective app.

Bluetooth serial modules allow all serial enabled devices to communicate with each other using Bluetooth.

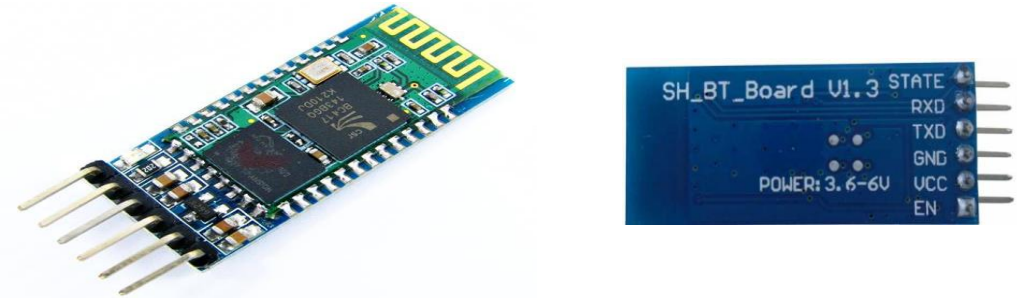

Figure 6.3: HC-05 Bluetooth module

It has 6 pins,

1. **Key/EN:** It is used to bring Bluetooth module in AT commands mode. If Key/EN pin is set to high, then this module will work in command mode. Otherwise by default it is in data mode. The default baud rate of HC-05 in command mode is 38400bps and 9600 in data mode.
2. **VCC:** Connect 5 V or 3.3 V to this Pin.
3. **GND:** Ground Pin of module.
4. **TXD:** Transmit Serial data (wirelessly received data by Bluetooth module transmitted out serially on TXD pin)
5. **RXD:** Receive data serially (received data will be transmitted wirelessly by Bluetooth module).
6. HC-05 module has two modes,
   - Data mode: Exchange of data between devices.
   - Command mode: It uses AT commands which are used to change setting of HC-05. To send these commands to module serial (USART) port is used.
7. It tells whether module is connected or not.

HC-05 module Information

- HC-05 has red LED which indicates connection status, whether the Bluetooth is connected or not. Before connecting to HC-05 module this red LED blinks continuously in a periodic manner. When it gets connected to any other Bluetooth device, its blinking slows down to two seconds.
- This module works on 3.3V. We can connect 5V supply voltage as well since the module has on board 5 to 3.3 V regulator.

- As HC-05 Bluetooth module has 3.3V level for RX/TX and microcontroller can detect 3.3 V level, so, no need to shift transmit level of HC-05 module. But we need to shift the transmit voltage level from microcontroller to RX of HC-05 module.
- The data transfer rate of HC-05 module can vary up to 1Mbps is in the range of 10 meters.

Modes of Operation:

The HC-05 Bluetooth Module can be configured in two modes of operation: Command Mode and Data Mode. In Command Mode, you can communicate with the Bluetooth module through AT Commands for configuring various settings and parameters of the Module like get the firmware information, change UART Baud Rate, change module name, set it as either Master or slave etc. An important point about HC-05 Module is that it can be configured as Master or Slave in a communication pair. In order to select either of the modes, you need to activate the Command Mode and sent appropriate AT Commands. Coming to the Data Mode, in this mode, the module is used for communicating with other Bluetooth device.

L298N MOTOR DRIVER:

This L298N Motor Driver Module is a high power motor driver module for driving DC and Stepper Motors. This module consists of an L298 motor driver IC and a 78M05 5V regulator. L298N Module can control up to 4 DC motors, or 2 DC motors with directional and speed control.

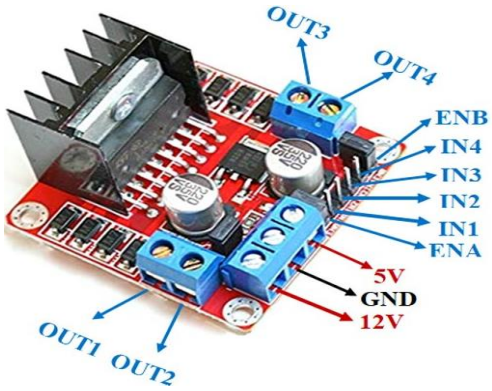

Figure 6.4: L298N motor module

L298N Module Pinout Configuration:

Pin Name	Description
IN1 & IN2	Motor A input pins. Used to control the spinning direction of Motor A
IN3 & IN4	Motor B input pins. Used to control the spinning direction of Motor B
ENA	Enables PWM signal for Motor A
ENB	Enables PWM signal for Motor B
OUT1 & OUT2	Output pins of Motor A
OUT3 & OUT4	Output pins of Motor B
12V	12V input from DC power Source
5V	Supplies power for the switching logic circuitry inside L298N IC
GND	Ground pin

Features & Specifications:

- Driver Model: L298N 2A
- Driver Chip: Double H Bridge L298N
- Motor Supply Voltage (Maximum): 46V
- Motor Supply Current (Maximum): 2A
- Logic Voltage: 5V

- Driver Voltage: 5-35V
- Driver Current:2A
- Logical Current:0-36mA
- Maximum Power (W): 25W
- Heatsink for better performance
- Power-On LED indicator

Brief about L298N Module

The L298N Motor Driver module consists of an L298 Motor Driver IC, 78M05 Voltage Regulator, resistors, capacitor, Power LED, 5V jumper in an integrated circuit. 78M05 Voltage regulator will be enabled only when the jumper is placed. When the power supply is less than or equal to 12V, then the internal circuitry will be powered by the voltage regulator and the 5V pin can be used as an output pin to power the microcontroller. The jumper should not be placed when the power supply is greater than 12V and separate 5V should be given through 5V terminal to power the internal circuitry.

ENA & ENB pins are speed control pins for Motor A and Motor B while IN1& IN2 and IN3 & IN4 are direction control pins for Motor A and Motor B.

Internal circuit diagram of L298N Motor Driver module is given below:

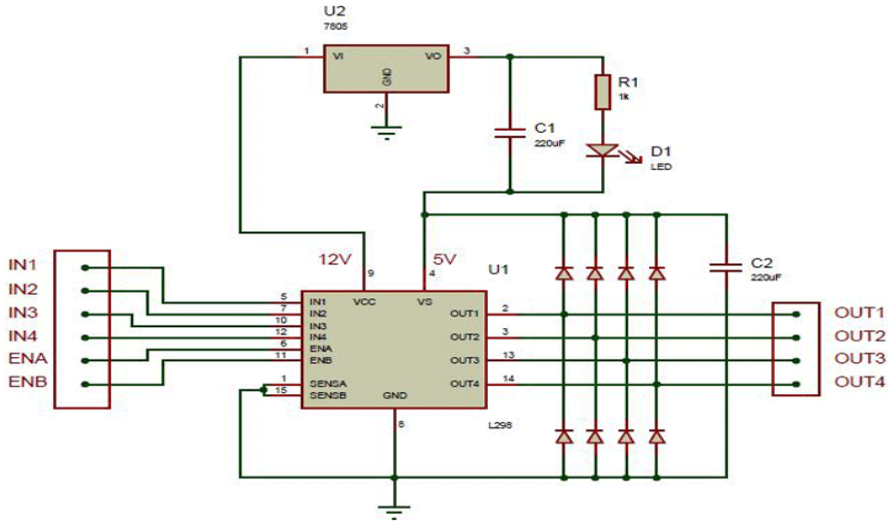

Figure 6.5: L298n motor driver connection

MECANUM WHEEL:

Mecanum wheel Also known as ilon wheels It is a conventional wheel with a series of rollers attached to its circumference, these rollers having an axis of rotation at 45° to the plane of the wheel in a plane parallel to the axis of rotation of the wheel. Depending on each individual wheel direction and speed, the resulting combination of all these forces produce a total force vector in any desired direction thus allowing the platform to move freely in the direction of the resulting force vector, without changing of the wheels themselves. It is perfect for tight space.

Figure 6.6: Mechanum Wheels

Benefits:	**Application:**
Compact size.	Robotics
High load capacity	Wheelchair
Add omni direction capabilities to your vehicle	Transfer vehicle

Characteristic :

The four mecanum wheels are each connected to a motor for independent control. The robot can move forward, reverse and spin just like four regular wheels. The configuration of rollers at 45° also allows the robot to translate sideways and through a combination of these, in any direction (even while spinning!).We splits the force into two vectors, one forward/backward and one right/left. When the wheels on one side are spun in opposite directions, the forward and backward vectors cancel out while both sideways vectors add up. Doing the reverse with the other two wheels results in four added sideways vectors.

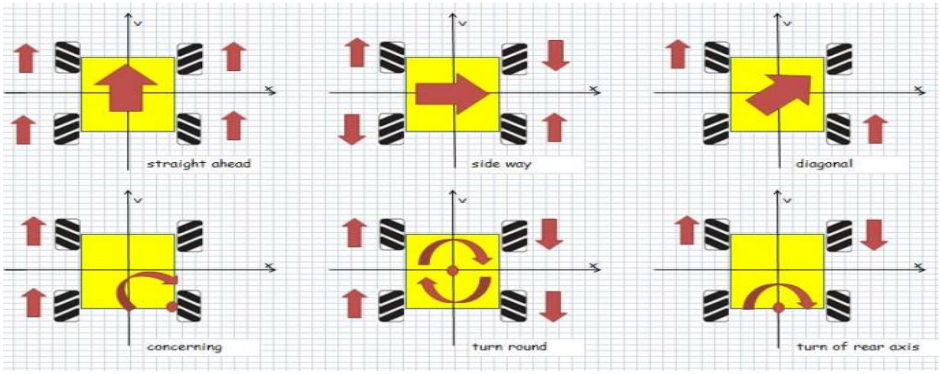

Figure 6.7: Mechanum Wheels Directions

P/N	Description	Wheel Dia	Axia width	No. of Rollers	Body Material	Roller Material	Net Weight	Load Capacity
14092	100mm Mecanum Wheel Right	100mm	50mm	9	Aluminum Alloy	Rubber	400g	15kg
14087	100mm Mecanum Wheel Left	100mm	50mm	9	Aluminum Alloy	Rubber	400g	15kg
14095	152mm Mecanum Wheel Right	152.4mm	55.5mm	15	Aluminum Alloy	Rubber	600g	15kg
14099	152mm Mecanum Wheel Left	152.4mm	55.5mm	15	Aluminum Alloy	Rubber	600g	15kg
14126	203mm Mecanum Wheel Left	203mm	67mm	12	Aluminum Alloy	Nylon	1.6kg	50kg
14127	203mm Mecanum Wheel Right	203mm	67mm	12	Aluminum Alloy	Nylon	1.6kg	50kg

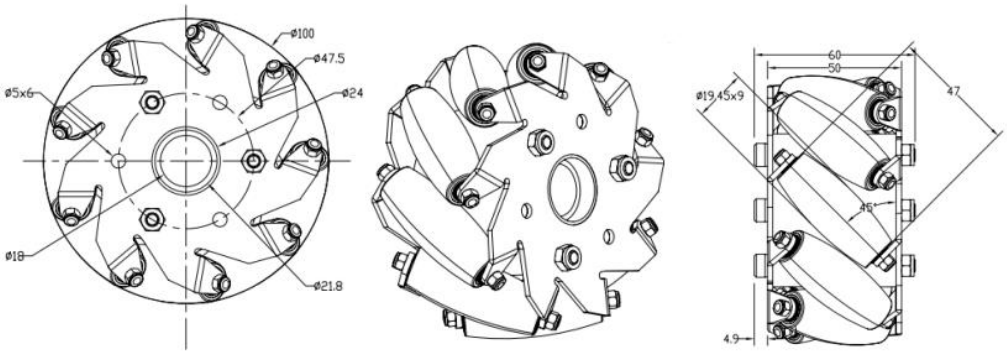

Figure 6.8: Mechanum Wheels Specification and Dimesions

150 RPM DUAL SHAFT BO MOTOR:

DC (BO) motor converts electrical energy into mechanical energy. Why DC gear motor used in robot Motor control circuit. DC MOTOR concept is where gears reduce the speed of the vehicle but increase its torque is known as gear reduction. DC motor is assembled with multiple gear setups. The speed of the motor is counted in terms of rotations of the soft per minute is called RPM. RPM means Revolution Per Minute. The setup assemble helps to increase the torque and reduce the motor speed. All micro-controller-based Robots this type of DC motor can be used. It is an alternative to our metal gear DC motors. It comes with an operating voltage of 3-12V and is perfect for building small and medium robots. The motor is ideal for DIY enthusiasts. This motor set is inexpensive, small, easy to install, and ideally suited for use in a mobile robot car. They are commonly used in our 2WD platforms

Figure 6.9: BO 150 RPM Motor

Specification:

Speed	**150 RPM**
Voltage	**5V-12V**
IP Rating	**IP54**
Torque	**2N.m**
No Load Current	**1-5A**

Features:

- Cost-effectiveness of the injection-molding process.
- Elimination of machining operations.
- Low density: lightweight, low inertia.
- Uniformity of parts.
- Capability to absorb shock and vibration as a result of elastic compliance.
- Ability to operate with minimum or no lubrication, due to inherent lubricity.
- The relatively low coefficient of friction.
- Corrosion-resistance; elimination of plating, or protective coatings.
- The quietness of operation.
- Tolerances often less critical than for metal gears, due in part to their greater resilience.
- Consistency with the trend to greater use of plastic housings and other components.

SWITCH:

An electric switch is a device that interrupts the electron flow in a circuit. Switches are primarily binary devices: either fully on or off and light switches have a simple design. When the switch is turned off, the circuit breaks and the power flow is interrupted. Circuits consist of a source of power and load. A load is a power-powered device. The function of an electric switch is to regulate the current between the load and source of power. The power source is the electrons that push through the circuits. The voltage is the quantity of force or pressure applied by the power source. Power sources must have a negative and positive endpoint. The negative terminal connects to the charge, and the electrons drive through the circuit. The load receives the current and returns it via the positive terminal to the power source. The electrical switch is inserted in this loop.

Figure 6.10: Switch

Function:

Electrical switches work according to a basic design. The most common switches are the on/off toggle ones. Three-way and dimmer circuits are similar in design. Three-way circuits consist of two separate switches that control the same device, while a dimmer circuit controls only the amount of electricity that passes. Electric circuits work when electricity can move in a continuous loop. The electricity cuts off once the circle is broken. This is where the switch comes in. A toggle on/off circuit breaks the current when it is off. The current or loop completes itself when it's in "on" position.

- Hotwire connects to one of the terminals on the switch and the load-powering outlet.
- The neutral wire links the other terminal to the load.
- The ground wire connects to the outlet.
- Electrical contacts inside the switch join the two terminals together. The connections link when you turn on the switch. As the switch is turned off, these contacts break. The current is controlled by the position in which the switch is located.

There are different types of switches such as toggle switches, joystick switches, pushbutton switch are a few of them. Toggle switches and pushbutton switches are common light switches used in a household. Finolex cables-make Finoswitch has a range of switches made of superior quality. They are durable and tested to last over 1,00,0000 clicks and have a fluorescent strip that glows in the dark and acts like a guide. Purchasing good quality products is always beneficial in the long run.

LI-ION BATTERY:

A lithium-ion or Li-ion battery is a type of rechargeable battery which uses the reversible reduction of lithium ions to store energy. It is the predominant battery type used in portable consumer electronics and electric vehicles. Compared to other rechargeable battery technologies, Li-ion batteries have high energy densities, low self-discharge, and no memory effect.

Specification:

Specific energy	100–265 Wh/kg (0.360–0.954 MJ/kg)
Self-discharge rate	0.35% to 2.5% per month depending on state of charge
Cycle durability	400–1,200 cycles
Nominal cell voltage	3.6 / 3.7 / 3.8 / 3.85 V, $LiFePO_4$ 3.2 V, $Li_4Ti_5O_{12}$ 2.3 V

Advantages of Lithium-Ion Batteries:

- High Energy Density: One of the biggest advantages of a lithium-ion battery is its high energy density. To put it straight, lithium-ion batteries can last way longer between charges all the while maintaining a high current output. That makes it the perfect battery for most modern needs.
- Low Self Discharge: Not only whilst being used, but lithium-ion batteries have a clear advantage when not being used as well. When kept idle, the rate of self-discharge, a common phenomenon in batteries, is extremely low.
- Low to Minimum Maintenance: Lithium-ion batteries are popular for their low maintenance batteries too. Most other cells like Nickel Cadmium batteries have a huge cost of ownership and maintenance.
- Options: One of the biggest advantages of lithium-ion batteries is the fact that they come in all shapes and sizes- presenting users with a large number of options to choose from according to their needs.

THONNY IDE:

Thonny is a free Python Integrated Development Environment (IDE) that was especially designed with the beginner Pythonista in mind. Specifically, it has a built-in debugger that canhelp when you run into nasty bugs, and it offers the ability to do step through expression evaluation, among other really awesome features.

The User Interface

Let's make sure you understand what Thonny has to offer. Think of Thonny as the workroom in which you will create amazing Python projects. Your workroom contains a toolbox containing many tools that will enable you to be a rock star Pythonista. In this section, you'll learn about each of the features of the UI that'll help you use each of the tools in your Thonny toolbox.

The Code Editor and Shell

Now that you have Thonny installed, open the application. You should see a window with several icons across the top, and two white areas:

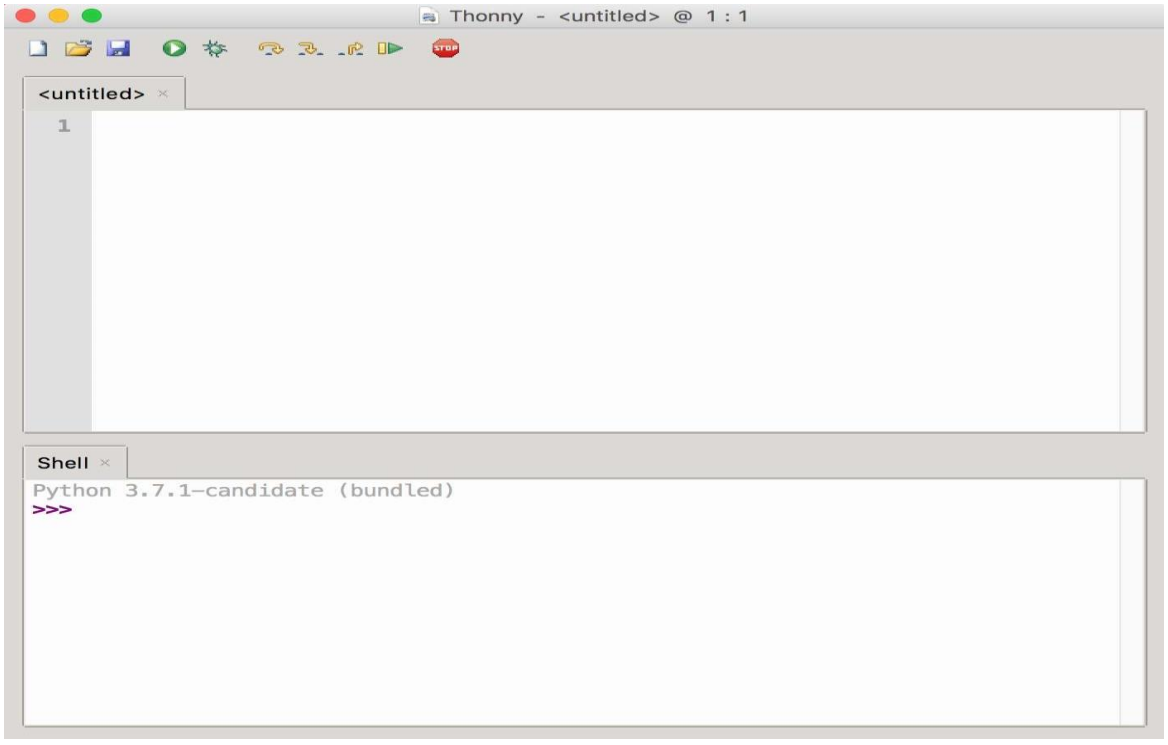

Figure 6.11: Software Interface

Notice the two main sections of the window. The top section is your code editor, where you will write all of your code. The bottom half is your Shell, where you will see outputs from your code.

The Icons

Across the top you'll see several icons. Let's explore what each of them does. You'll see an image of the icons below, with a letter above each one. We will use these letters to talk about each of the icons:

Figure 6.12: Software's Functions

Working our way from left to right, below is a description of each of the icons in the image.

A: The paper icon allows you to create a new file. Typically in Python you want to separate your programs into separate files. You'll use this button later in the tutorial to create your first program in Thonny!
B: The open folder icon allows you to open a file that already exists on your computer. This might be useful if you come back to a program that you worked on previously.
C: The floppy disk icon allows you to save your code. Press this early and often. You'll use this later to save your first Thonny Python program.
D: The play icon allows you to run your code. Remember that the code you write is meant to be executed. Running your code means you're telling Python, "Do what I told you to do!" (In other words, "Read through my code and execute what I wrote.")
E: The bug icon allows you to debug your code. It's inevitable that you will encounter bugs when you're writing code. A bug is another word for a problem. Bugs can come in many forms, sometimes appearing when you use inappropriate syntax and sometimes when your logic is incorrect.

Thonny's bug button is typically used to spot and investigate bugs. You'll work with this later in the tutorial. By the way, if you're wondering why they're called bugs, there's also a fun story of how it came about!

F-H: The arrow icons allow you to run your programs step by step. This can be very useful when you're debugging or, in other words, trying to find those nasty bugs in your code. These icons are used after you press the bug icon. You'll notice as you hit each arrow, a yellow highlighted bar will indicate which line or section Python is currently evaluating:

- The F arrow tells Python to take a big step, meaning jumping to the next line or block of code.
- The G arrow tells Python to take a small step, meaning diving deep into each component of an expression.
- The H arrow tells Python to exit out of the debugger.

I: The resume icon allows you to return to play mode from debug mode. This is useful in the instance when you no longer want to go step by step through the code, and instead want your program to finish running.

J: The stop icon allows you to stop running your code. This can be particularly useful if, let's say, your code runs a program that opens a new window, and you want to stop that program. You'll use the stop icon later in the tutorial.

Click the play button to run your program.

See the output in the Shell window.

Before continuing, make sure that you have your device attached to a USB port. If required, install the necessary drivers for serial communication.

In Thonny, open the back-end manager (Run → Select interpreter) and choose the type of your device from the drop-down list. If your device doesn't have a specific back-end, then try "MicroPython (generic)". NB! Since version 3.3 you can also use the back-end switcher in the lower-right corner of Thonny's main window.

Now you can specify the port your device is connected to. If you have only one board connected, then you could leave this setting for automatic detection.

After you press "OK", Thonny's shell should present either the prompt of your device's REPL or the output of the script which is currently running on your device.

(If you don't have suitable MicroPython (or CircuitPython) firmware installed on your device yet, then see below for the section about firmware installation.)

Editing scripts

When you have a MicroPython back-end selected, "File => Open" allows you to open a file either from local disk, device's USB drive, or "directly" form your device's file system. Same goes to save commands ("Save", "Save as", "Save copy as").

Note that Thonny always synchronizes file writes — this reduces the risk of corrupt files when you accidentally reset your device without proper eject right after saving a file onto its USB drive.

You can also open files by double-clicking them in Thonny's file browser (View => Files).

During editing your scripts, Thonny offers autocomplete (performed by the jedi library). This is based on the API-s scraped from specific devices with specific MicroPython / CircuitPython versions (different for each backend type). This means that if you are using a different version of firmware then the autocompleter may offer some names that are not present on your device and miss some that are.

Although Thonny allows you to work without saving anything locally, for serious work we recommend keeping master copies of your scripts in your main computer (preferably under version control) and copy them to the device as required (File => Save copy as). It is too easy to lose your work if you rely only on the device's flash memory.

Running scripts

The easiest way to test your script is to execute it like you would with the CPython back-end — "Run → Run current script (F5)". It doesn't matter whether your script is stored on

the local disk or on the device — in both ways Thonny simply submits the program text to device's REPL. When you have saved your script onto the device (eg. as "main.py"), you can check how your device would perform without Thonny. This can be done easily by selecting "Run => Send EOF / Soft reboot (Ctrl+D)".

Managing other files

You can upload / download data and library files and directories via respective commands in the file browser (View => Files) via files' context menu (opened by right-clicking on a file).

Thonny REPL vs. plain MicroPython serial REPL

After connecting to your device, Thonny switches it to raw REPL mode and uses this mode under the hood to run the scripts, query global variables, etc. For the user it is still presented as regular REPL. (Note that this mode change is not permanent — if you reset the device and open it with another program, REPL is back in regular mode. If you don't want to reset it then you can get back to regular mode with Ctrl+B.)

Because of using raw REPL under the hood, Thonny doesn't support Ctrl-A and Ctrl-B for switching between raw and regular REPL. It also doesn't support Ctrl-E for entering paste mode -- this is not necessary as you can paste any amount of code into the regular Thonny REPL.

Chapter 7: Problems, faults, bugs, challenges

7.1. Problems

There can be several problems that may arise when making an omni-directional Bluetooth controlled car, such as:

- Connectivity issues: If the car is not properly connected to the Bluetooth controller, it may not respond to commands or may not function properly.
- Power constraints: The car may not have enough power to run all of its components, such as the motors and Bluetooth module, simultaneously.
- Motor control: Controlling the car's motors to move in all directions can be challenging, as it requires precise control and calibration of the motors.
- Design challenges: The car's design may make it difficult to achieve omni-directional movement, such as a lack of space for additional motors or wheels.
- Interference: Other Bluetooth devices in the area may interfere with the car's communication with the controller.
- Software bugs: The software controlling the car's movement may have bugs that make it difficult to move the car in all directions or make it behave erratically.
- Cost: The cost of the components like Bluetooth module, Motors, PCB, Power source, etc. can be a major concern while building such a project.
- Debugging: Debugging the issues with the car's movement and connectivity can be time-consuming and challenging.

These are just some examples of problems that may arise, and the specific challenges you face will depend on the details of your design and implementation.

7.2. Faults

Faults that may occur when making an omni-directional Bluetooth controlled car include:

- Motor failure: The motors may fail due to overloading, overheating, or other causes, resulting in the car being unable to move.
- Battery failure: The car's battery may not have enough power to run all of its components, or it may become damaged and not hold a charge.
- Bluetooth module failure: The Bluetooth module may become damaged or malfunction, preventing the car from communicating with the controller.
- Circuit failure: The car's circuit board may become damaged or malfunction, causing issues with power distribution or control of the motors.
- Software bugs: The software controlling the car's movement may have bugs or errors that cause the car to behave erratically or not respond to commands.
- Mechanical failure: The car's mechanical components, such as wheels or gears, may become damaged or worn out, preventing the car from moving smoothly.
- Interference: Other Bluetooth devices in the area may interfere with the car's communication with the controller, causing the car to behave erratically or not respond to commands.
- Human error: The car may not function properly due to user error such as incorrect assembly, incorrect wiring, or incorrect programming.

It is important to test and debug the car thoroughly to identify and fix any faults that may arise. It is also good to have a backup plan when working with a complex project such as this.

7.3. Bugs

There are several bugs that can appear when building an omni-directional Bluetooth controlled car, including:

- Control algorithm bugs: The control algorithm can be a source of bugs, particularly if it is not well-designed or if it is not properly implemented. Common issues include poor responsiveness, instability, or unexpected behavior.
- Bluetooth connection bugs: Bluetooth connection bugs can occur if the car's Bluetooth module is not properly configured or if the signal is disrupted by other electronic devices. This can lead to lost or delayed commands, or an inability to connect to the car.
- Mechanical bugs: Mechanical bugs can occur if the car's mechanical design is not well-engineered or if the components are not properly assembled. This can lead to issues such as poor mobility or mechanical failure.
- Power management bugs: Power management bugs can occur if the car's power system is not properly configured or if the battery is not properly maintained. This can lead to issues such as poor battery life or unexpected shut downs.
- Sensor bugs: Sensor bugs can occur if the car's sensors are not properly calibrated or if they are not functioning properly. This can lead to issues such as poor navigation or unexpected behavior.
- Software bugs: Software bugs can occur if the car's firmware is not properly written or if it is not properly tested. This can lead to issues such as unexpected behavior, crashes, or other system failures.
- Integration bugs: Integration bugs can occur when different components of the car are not properly integrated, leading to issues such as poor performance, system failures, or unexpected behavior.

It's important to note that, a good testing and debugging process will help to minimize and fix those bugs, and also it's important to use reliable and well-tested components and software libraries.

7.4. Challenges

There are several challenges that may arise when building an omni-directional Bluetooth controlled car, including:

- Power consumption: Bluetooth consumes a significant amount of power, which can quickly drain the battery of a small car. This can be mitigated by using a low-power Bluetooth module or by implementing power-saving strategies.
- Control algorithm: Controlling an omni-directional car can be difficult, as the wheels need to be controlled independently to achieve the desired movement. This can be mitigated by using a robust control algorithm that can handle the complexities of an omni-directional car.
- Signal interference: Bluetooth signals can be easily blocked or disrupted by other electronic devices or physical obstacles, which can affect the reliability of the control system.
- Mechanical design: Building an omni-directional car requires a complex mechanical design, which can be difficult to implement and maintain.
- Safety concerns: As with any mobile robot, safety concerns must be addressed to prevent injury to people or damage to property.
- Cost: The cost of implementing such a system can be high due to the need for specialized components such as motors, wheels, control boards, and batteries.
- Calibration and maintenance: The car needs to be properly calibrated and maintained to ensure optimal performance and safety.

Chapter 8: Teamwork

8.1. Summary of team work

8.1.1 Attributes

1	Attends group meetings regularly and arrives on time.
2	Contributes meaningfully to group discussions.
3	Completes group assignments on time.
4	Prepares work in a quality manner.
5	Demonstrates a cooperative and supportive attitude.
6	Contributes significantly to the success of the project.

8.1.2 Score

1=strongly disagree; **2**=disagree; **3**=agree; **4**=strongly agree

Student 1: _____

Student 2: _____

Student 3: _____

Student 4: _____

		Evaluated by		
	Attributes	Student 2	Student 3	Student 4
Student 1	1			
	2			
	3			
	4			
	5			
	6			
	Grand Total			

Student 2		Evaluated by		
	Attributes	Student 1	Student 3	Student 4
	1			
	2			
	3			
	4			
	5			
	6			
	Grand Total			

Student 3		Evaluated by		
	Attributes	Student 1	Student 2	Student 4
	1			
	2			
	3			
	4			
	5			
	6			
	Grand Total			

Student 4		Evaluated by		
	Attributes	Student 1	Student 2	Student 3
	1			
	2			
	3			
	4			
	5			
	6			
	Grand Total			

Signature of Student 1

Signature of Student 2

Signature of Student 3

Signature of Student 4

Chapter 9: Conclusion

Creating an omni-directional Bluetooth controlled car requires a combination of mechanical, electrical, and software engineering skills. The car must be designed and built to move in any direction, and must be equipped with a Bluetooth module for remote control. Building a omni-directional Bluetooth controlled car involves several steps, including designing and 3D printing the Mecanum wheels, assembling the car chassis, integrating the wheels with motors and a control system, and programming the Bluetooth controller. The end result should be a car that is capable of moving in any direction and controlled wirelessly via Bluetooth. It may involve some troubleshooting and adjustments along the way, but with proper planning, materials, and knowledge, it should be possible to build a functioning omni-directional car. The car must also be able to navigate and avoid obstacles and other hazards. The process of creating an omni-directional Bluetooth controlled car involves several steps, including designing the mechanical components, designing and programming the electronics, and testing and refining the final product. Tools such as simulation software, and firmware and software development environments are used to design, program and test the car. It's important to note that creating an omni-directional Bluetooth controlled car can be a complex and challenging process, but with the right team, tools and skills, a successful and high-quality product can be developed.

References

The sites which were used while doing this project:

1. Research Paper(Journal, conference etc.)
2. https://www.elementzonline.com
3. https://www.youtube.com
4. https://www.flyrobo.in
5. http://www.microchip.com
6. https://srituhobby.com

APPENDIX

```
from machine import UART, Pin
from time import sleep

m11 = Pin(2,Pin.OUT)
m12 = Pin(3,Pin.OUT)
m21 = Pin(4,Pin.OUT)
m22 = Pin(5,Pin.OUT)
m31 = Pin(6,Pin.OUT)
m32 = Pin(7,Pin.OUT)
m41 = Pin(8,Pin.OUT)
m42 = Pin(9,Pin.OUT)

uart = UART(0, 9600)

def forward():   #move forward
    m11.off()
    m12.on()
    m21.off()
    m22.on()
    m31.off()
    m32.on()
    m41.on()
    m42.off()

def backward():   #move backward
    m11.on()
    m12.off()
    m21.on()
    m22.off()
    m31.on()
    m32.off()
    m41.off()
    m42.on()

def left():       #move left
    m11.off()
    m12.on()
    m21.on()
    m22.off()
    m31.on()
    m32.off()
    m41.on()
    m42.off()
```

```
def right():      #move right
    m11.on()
    m12.off()
    m21.off()
    m22.on()
    m31.off()
    m32.on()
    m41.off()
    m42.on()

def forwardleft():   #move forward left
    m11.off()
    m12.on()
    m21.off()
    m22.off()
    m31.off()
    m32.off()
    m41.on()
    m42.off()

def forwardright():   #move forward right
    m11.off()
    m12.off()
    m21.off()
    m22.on()
    m31.off()
    m32.on()
    m41.off()
    m42.off()

def backwardright():   #move backward right
    m11.on()
    m12.off()
    m21.off()
    m22.off()
    m31.off()
    m32.off()
    m41.off()
    m42.on()

def backwardleft():   #move backward left
    m11.off()
    m12.off()
    m21.on()
    m22.off()
```

```
    m31.on()
    m32.off()
    m41.off()
    m42.off()

def rotateleft():   #rotate left
    m11.on()
    m12.off()
    m21.on()
    m22.off()
    m31.off()
    m32.on()
    m41.on()
    m42.off()

def rotateright():   #rotate right
    m11.off()
    m12.on()
    m21.off()
    m22.on()
    m31.on()
    m32.off()
    m41.off()
    m42.on()

def stop():
    m11.off()
    m12.off()
    m21.off()
    m22.off()
    m31.off()
    m32.off()
    m41.off()
    m42.off()

while True:
    if uart.any():
        value = uart.readline()
        print(value)

        if value == b'A':
            forwardleft()
        elif value == b'B':
            forward()
        elif value == b'C':
            forwardright()
```

```
elif value == b'D':
    left()
elif value == b'E':
    right()
elif value == b'F':
    backwardleft()
elif value == b'G':
    backward()
elif value == b'H':
    backwardright()
elif value == b'I':
    rotateleft()
elif value == b'J':
    rotateright()
elif value == b'S':
    stop()
```