MAKERNOVA 3.0

PS4 AND WEB-GUI DRIVEN OMNIDIRECTIONAL MOBILITY

Problem Statement

Develop an omnidirectional drive system using omniwheels, enabling movement in any direction without changing orientation. The system will be controlled via a PS4 controller and a Web GUI It will use STM32 and ESP32 microcontrollers - STM32 for motor control and CAN communication, ESP32 for PS4 and Web GUI via Wi-Fi. Communication between them will use UART or CAN Web GUI will use WebSocket for real-time control. The robot will support 3-wheel configurations for precise, smooth movement.

Members

- Shubhangi Singh
- Mehak Oberoi
- Aryan Jain
- Bhavya Desai
- Rudra Prajapati
- Shubham Kumar
- Hitesh Bihani
- Harsh Patel
- Hari Kiran

Mentors

- Kevin Dudakiya
- Shashwat Thakur
- Omkar Sinh
- Divyavardhan Singh

Acknowledgment

We would like to express our heartfelt gratitude to all those who supported and guided us throughout the process of this project. First and foremost, we extend our sincere thanks to Kevin Dudakiya, Shashwat Thakur, Divyavardhan Singh and Omkar Sinh. From their invaluable insights, encouragement, and expertise, they were instrumental in shaping this work. Their guidance helped us stay focused and motivated throughout the journey. We are also deeply thankful to DRISHTI for organising Makernova , providing the resources and environment necessary to carry out this project successfully. A special note of appreciation goes to every Mentor and friend for their unwavering support, patience, and belief in us. Their encouragement kept our Team going even during the most challenging moments. Lastly, we would like to acknowledge everyone who contributed directly or indirectly to this endeavor. Your help has been deeply appreciated.

Abstract

This document outlines the development of an omnidirectional robotic platform powered by omniwheels and controlled via two distinct interfaces: a PlayStation 4 (PS4) controller and a web-based graphical user interface (GUI). The use of omniwheels enables the bot to move fluidly in any direction without changing its orientation, offering superior maneuverability in tight or dynamic environments. The PS4 controller provides real-time manual control, while the web GUI allows remote operation and monitoring, making the system versatile for both local and networked applications. The integration of microcontrollers, motor drivers, and wireless communication modules forms the backbone of the system, ensuring responsive and reliable performance. This documentation details the design architecture, control algorithms, hardware-software integration, and potential use cases in fields such as automation, surveillance, and interactive robotics. This documentation serves as a valuable resource for robotics enthusiasts and developers looking to build web development skills, robots using ESP32, and STM32 for connectivity of PS4 and Web interface to control the robot.

Contents

List of Figures

List of Tables

1 Introduction

In the evolving landscape of robotics,the mobility and control precision are critical factors that define the effectiveness of autonomous and semi-autonomous systems. This project introduces an omnidirectional robot equipped with omniwheels, which enables seamless movement in any direction without the need for rotational alignment.This degree of mobility is particularly advantageous in constrained or quickly evolving environments where agility and quick response are essential.

To enhance user interaction and control flexibility, the bot is designed to operate through two distinct interfaces: a PlayStation 4 (PS4) controller for direct, tactile control, and a web-based graphical user interface (GUI) for remote access and monitoring. This dual-control architecture not only broadens the usability of the bot in various scenarios, but also shows the integration of hardware and software systems in modern robotics. Let's get the adventure started!

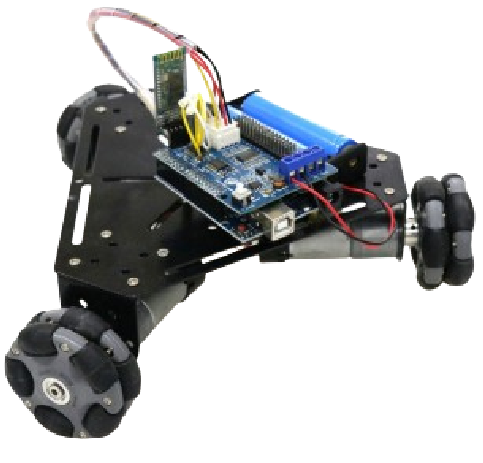


Figure 1.1: The three wheels omnidirectional mobile robot

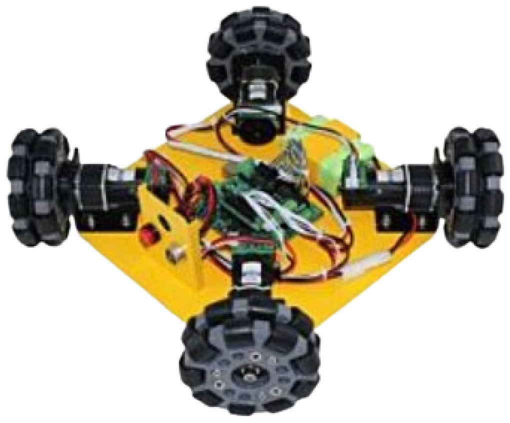


Figure 1.2: The four wheels omnidirectional mobile robot

2 Wheels

Omnidirectional robots are made possible through specialized omni wheels, with the most common designs being

1. Omni Wheels

2. Mecanum Wheels

2.1 Omni Wheels

Omniwheels are unique wheels that have rollers positioned at 90° angles all around the wheel. The wheel may slide laterally while still moving forward or backward thanks to these rollers. They enable the robot to travel in any direction without rotating its body when placed in a particular configuration.

- Roller Design: Smoothes transitions by lowering friction during lateral movement.

2.2 Mecanum Wheels

The mecanum wheels are specially designed to allow omnidirectional movement. Each wheel has a series of rollers mounted at a 45-degree angle, which allows the car to move in any direction, including forward, backward, sideways, diagonally, and even rotate in place.

- Roller Design: The angled rollers on the wheels create lateral forces that, when combined, produce omnidirectional movement.

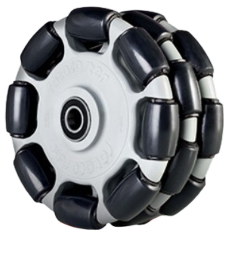


Figure 2.1: Omni Wheel

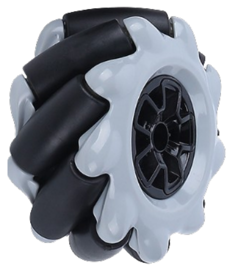


Figure 2.2: Mecanum Wheel

3 Omni and Mecanum Wheels

	Omni-Wheel	**Mecanum-Wheel**
Movement Type	True omnidirectional (smooth lateral, diagonal, rotational motion)	Holonomic (omnidirectional with more drift and vibration)
Responsiveness	Faster directional changes due to direct vector mapping	Slight lag due to roller angle and friction
Control Complexity	Moderate — simpler kinematics and vector math	High — requires complex inverse kinematics and calibration
Energy Efficiency	More efficient in lateral and diagonal movement	Less efficient due to roller friction and slippage
Precision	High — ideal for fine-tuned movement and tight navigation	Moderate — more drift during diagonal motion
Ideal Use Cases	Agile indoor bots, educational platforms, GUI/joystick-controlled systems	Industrial bots, heavy payload transport, rough terrain navigation

Table 3.1: Comparison between Omni and Mecanum Wheel

4 Omnidirectional Drive

An omnidirectional robot can move in any direction. A mobile robot occupies space in three dimensions: ω (orientation), x, and y (position). The robot can travel in any direction, regardless of its position or orientation, using the omnidirectional mobility approach, as illustrated in Fig. 1. This enables it to produce movement simultaneously on the x and y axes (linear velocities Vx and Vy). The robot has three degrees of freedom (DOF) and can move freely by adjusting these velocities. A robot that can move in any direction is called an omnidirectional robot. The orientation of such a mobile robot (ω) and x and y define its three-dimensional space. As shown in Figure 4.1.

The robot can travel freely in any direction thanks to the omnidirectional motion mechanism, regardless of its orientation or current position. This feature produces linear velocities (Vx and Vy) by enabling simultaneous movement in the x and y directions. The robot can move freely and has three degrees of freedom (DOF) by adjusting these velocities.

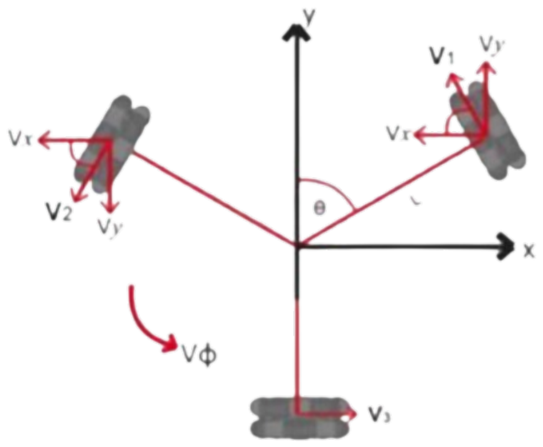


Figure 4.1: Velocity components

5 Three wheel or Four wheel

The 3-wheel omnidirectional robot is often preferred in small-scale robotics projects due to its simplicity and efficiency. Some key advantages are the following.

- Reduced Cost and Simplicity : A 3-wheel design requires only three omni-wheels and three motors. This reduces the number of components, making the system more economical, easier to assemble, and easier to maintain compared to a four-wheeler robot.

- Lower Weight: With one wheel and motor less, the overall weight of the robot is reduced. This contributes to improved efficiency and less power consumption, which is especially beneficial for battery-operated robots.

- Kinematic Simplicity : The triangular wheel arrangement of a 3-wheel system simplifies both inverse kinematic equations. This makes programming and control implementation easier.

- Better Mobility: A 3-wheel omnidirectional robot can rotate smoothly around its center point, providing high agility in tight spaces. This is particularly useful in indoor environments or in applications that require precise movements.

- Lower Power Consumption : Since the robot operates with three motors instead of four, the overall energy requirement is reduced, increasing the operational time on a given power source.

Although the 3-wheel robot is advantageous in terms of simplicity and efficiency, a 4-wheel omnidirectional robot offers greater stability, higher load carrying capacity, and better in the event of wheel or motor failure.

In summary, a 3-wheel omnidirectional robot is ideal for lightweight, educational and prototyping applications where simplicity is prioritized, while a 4-wheel design is better suited for heavy-duty and industrial applications.

5.1 Vector Diagram

Vector diagram of a three wheel omnidirectional robot.

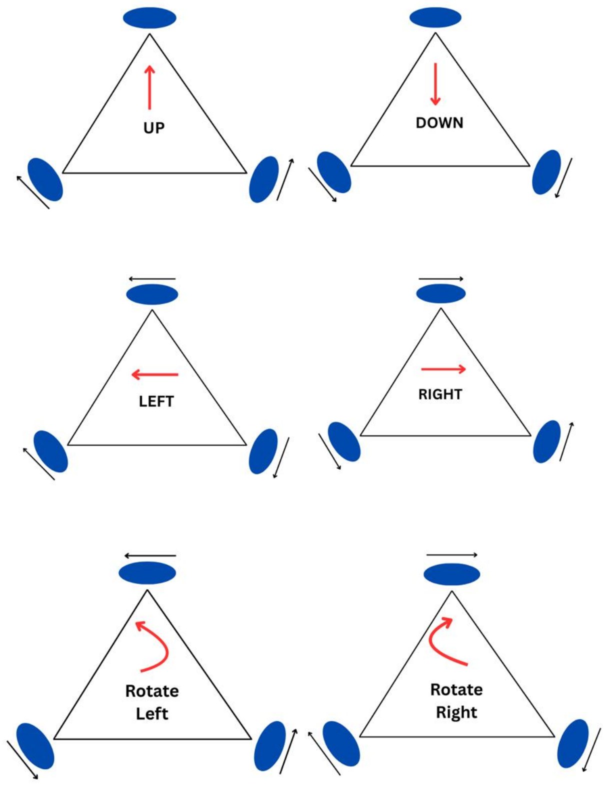


Figure 5.1: Three wheel vector diagram

6 Holonomic and Non-Holonomic Motion

6.1 Holonomic Motion

A system is said to be holonomic if its motion constraints can be expressed purely in terms of position coordinates and time. This means the system's final position depends only on its starting and ending points, not the path taken.In holonomic systems, if we know the initial and final coordinates, we can describe the motion without worrying about the intermediate path.
Examples: A pendulum constrained to swing around a fixed pivot (constant length).

6.2 Non-Holonomic Motion

A system is non-holonomic if its constraints cannot be expressed only with position coordinates and time. Instead, the constraints often involve velocities and cannot be integrated into simple position equations. In such systems, the final position also depends on the path taken.The path matters. To reach a certain position, the robot must follow specific movements that respect the velocity-based constraints.
Examples: A car or bicycle: It can move forwards/backwards but cannot move directly sideways.

- **Application :**
  The robot uses three omni-wheels arranged at 120 ° angles, which allow it to move in any direction without need to rotate first. This makes the robot's motion holonomic, since it can directly achieve any position and orientation in the plane by combining wheel velocities, without being limited to specific movement paths. In contrast, a car-like robot with normal wheels would be non-holonomic, as it cannot move directly sideways and must

follow a curved path to reach certain points.Thus, our project demonstrates a holonomic motion system, offering higher mobility and control compared to non-holonomic systems.

7 Kinematics

Kinematics is the study of motion in physics. It describes how objects move using concepts such as displacement, velocity, and acceleration, without considering the forces that cause the motion.

7.1 Forward Kinematics

Forward kinematics refers to the use of the kinematic equations of a robot to compute the position of the end effector from specified values for the joint parameters. Given a kinematic chain composed of links and joints with multiple degrees of freedom , finding the position and orientation of the end effector in the operational workspace when all the joint parameters are known. This is called Forward Kinematics.
Examples: Calculating the final position of a robot gripper given its joint angles

7.2 Inverse Kinematics

Determination of joint variables in terms of the end-effector position and orientation is called inverse kinematics.Inverse kinematics makes use of the kinematics equations to determine the joint parameters that provide a desired configuration (position and rotation) for each of the robot's end-effectors. This is important because robot tasks are performed with the end effectors, while control effort applies to the joints. Determining the movement of a robot so that its end-effectors move from an initial configuration to a desired configuration is known as motion planning. Inverse kinematics transforms the motion plan into joint actuator trajectories for the robot. Examples: Figuring out the angles for your own arm to reach a specific point, like a cup of coffee on a table.

7.3 Kinematics of Three Wheel Omnidirectional Bot

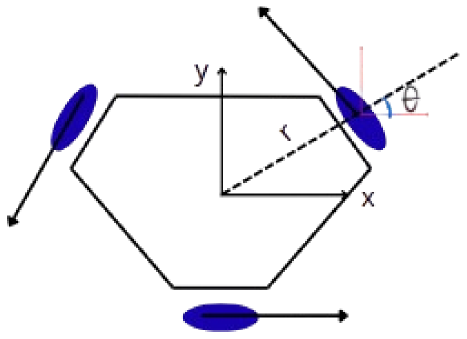


Figure 7.1: Velocity components

$V_1 = -V_1 sin\theta\hat{i} + V_1 cos\theta\hat{j}$
$V_2 = -V_2 sin(\theta + 120°)\hat{i} + V_2 cos(\theta + 120°)\hat{j}$
$V_3 = -V_3 sin(\theta + 240°)\hat{i} + V_3 cos(\theta + 240°)\hat{j}$

Now we have to find all 3 D.O.F of our system i.e X,Y and ω

$V_x = -V_1 sin\theta - V_2 sin(\theta + 120°) - V_3 sin(\theta + 240°)$
$V_y = V_1 cos\theta + V_2 cos(\theta + 120°) + V_3 cos(\theta + 240°)$
$\omega = \omega_1 + \omega_2 + \omega_3$

Now rotate the axis by θ in anti clockwise direction such that $\theta = 0°$, our equation in matrix form becomes:

$$\begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix} = \begin{bmatrix} 0 & -\frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \\ 1 & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{R} & \frac{1}{R} & \frac{1}{R} \end{bmatrix} \cdot \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \cdot R$$

where, R is the radius of the wheel
ω_i (where ,i =1,2,3)is the angular velocity of each wheel

This matrix is of the form :

$$V = (MW)R$$

taking M^{-1} on both sides,

$$W = \frac{M^{-1}V}{R}$$

and we know that,

$$M^{-1} = \frac{Adj(M)}{|M|}$$

$$\left[Adj(M)\right] = \begin{bmatrix} 0 & \frac{\sqrt{3}}{R} & \frac{\sqrt{3}}{2} \\ -\frac{3}{2R} & -\frac{\sqrt{3}}{2R} & \frac{\sqrt{3}}{2} \\ \frac{3}{2R} & -\frac{\sqrt{3}}{2R} & \frac{\sqrt{3}}{2} \end{bmatrix}$$

$$|M| = \frac{3\sqrt{3}}{2R}$$

∴ the matrix equation now becomes:

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \frac{2R}{3\sqrt{3}} \cdot \frac{1}{R} \cdot \begin{bmatrix} 0 & \frac{\sqrt{3}}{R} & \frac{\sqrt{3}}{2} \\ -\frac{3}{2R} & -\frac{\sqrt{3}}{2R} & \frac{\sqrt{3}}{2} \\ \frac{3}{2R} & -\frac{\sqrt{3}}{2R} & \frac{\sqrt{3}}{2} \end{bmatrix} \cdot \begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix}$$

solving this, we get:

$w_1 = \frac{2V_y}{3R} + \frac{\omega}{3}$

$w_2 = \frac{-V_x}{\sqrt{3}} - \frac{V_y}{3R} + \frac{\omega}{3}$

$w_3 = \frac{V_x}{\sqrt{3}R} - \frac{V_y}{3R} + \frac{\omega}{3}$

8 Hardware Components

Components	**Description**
Motor Driver (RMCS2305)	This is a dual-channel, optically isolated DC motor driver for high power brushed DC motors. It operates from 6-30V, supports up to 20A continuous and 60A peak per motor.
Motor Driver (MD10C)	Designed to drive high-current brushed DC motors up to 13A continuously. This motor driver offers bi-directional control over a single brushed DC motor.
Buck Converter (LM2596)	It is a step-down (DC-DC) voltage regulator that can step down from 16.8V to 5V/3.3V.
Planetary Motor	This type of motor uses a gear system, where small gears (planets) rotate around a central gear (sun) inside a ring gear.
Li-Po Battery	This lithium polymer battery acts as the main power source. It has a capacity of 10000mAh and provides a nominal voltage of 14.8V (16.8V when fully charged).
Connecting Wires	• XT60: A type of high-current electrical connector. • Grove Connector: A set of hardware components that make connections easy without the need for breadboards and jumper wires.
PS4 Controller	The controller serves as the primary wireless controller for the bot. It has dual analog joysticks that allow precise speed and direction control. It connects to ESP32 via Bluetooth.

Table 8.1: Components used

9 Micro-controllers

A microcontroller is a compact, integrated circuit that acts as a small computer on a single chip, combining a processor, memory, and input/output peripherals. It is designed to perform specific control tasks within embedded systems, such as those found in appliances, vehicles, and consumer electronics

9.1 ESP 32

The ESP32 is a powerful, low-cost microcontroller developed by Espressif Systems, designed for embedded and IoT applications. It is widely used in robotics, smart devices, and DIY electronics because of its extensive features and capabilities.

9.1.1 Features:

- Processor : Dual-core (or single-core in some variants) 32-bit LX6 microprocessor, running up to 240MHz and delivering up to 600 DMIPS processing power
- Memory : 520KB SRAM, 448KB ROM, and usually 4MB Flash memory for program/data storage
- Wireless : Integrated Wi-Fi (802.11 b/g/n, up to 150Mbps) and Bluetooth v4.2 (Classic + BLE)
- GPIO : 34 programmable general-purpose input/output pins, supporting various sensors and peripherals
- Analog/Digital :18 channels of 12-bit ADC, 2 channels of 8-bit DAC, 10 capacitive touch sensors, and multiple PWM channels for motor/LED control

9.1.2 Applications:

- IoT devices (smart home, automation)
- Robotics and motor control systems
- Wireless sensor networks
- Wearable technology

9.1.3 Why ESP 32?

- Wireless Control : ESP32 has built-in Wi-Fi and Bluetooth, so you can control your robot remotely using a smartphone, computer, or even over the internet.
- Powerful Processor: It can handle complex calculations and real-time tasks needed for omni-directional movement and sensor integration.
- Multiple Connections : It supports many input/output pins, making it easy to connect motors, sensors, encoders, and other hardware essential for robot control.
- Energy Efficient : ESP32 supports low-power modes, allowing your robot to run longer on batteries if needed.
- Cost-Effective : Offers advanced features at a low price, keeping your project affordable without sacrificing performance.

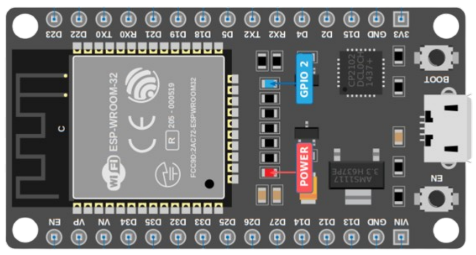

D23 D22 TX0 RX0 D21 D19 D18 D5 TX2 RX2 D4 D2 D15 GND 3V3
ESP-WROOM-32
GPIO 2
POWER
BOOT
EN
EN VP VN D34 D35 D32 D33 D25 D26 D27 D14 D12 D13 GND VIN

Figure 9.1: ESP 32

9.2 STM 32

STM32 is a family of powerful 32-bit microcontrollers developed by STMicroelectronics, built around ARM Cortex-M cores. Known for high performance and efficiency, STM32 MCUs are widely used in embedded systems and IoT projects. These microcontrollers combine robust processing, extensive peripherals, and diverse memory options in a single chip.

9.2.1 Features:

- Core Processing : The family uses various ARM Cortex-M cores (e.g., Cortex-M0+, M4, M7), offering a wide spectrum of performance from ultra-low-power to very high-speed applications.
- Frequency range : Wide operating frequency range, up to 480 MHz in some variants
- Communication Interfaces : Features a rich set of communication protocols, such as USART, SPI, I2C, USB (including USB Type-C with PD), CAN, and SDIO.
- Timers and Counters : A wide range of high-resolution timers, PWM outputs, and watchdog timers are available for precise control and real-time operations.
- Memory and Storage : STM32 MCUs include both high-density Flash memory for program storage and embedded SRAM for data, with varying sizes and capabilities depending on the series.
- Peripherals : Includes multiple Analog-to-Digital Converters (ADCs) and Digital-to-Analog Converters (DACs) with selectable resolution and high precision, along with comparators.
- Power Management : The family includes many ultra-low-power modes (Sleep, Stop, Standby) and features like low-power DMA to reduce energy consumption, making them suitable for battery-powered devices.

9.2.2 Application:

- Consumer electronics
- Industrial automation and robotics
- Automotive electronics

9.2.3 Why STM32?

- Versatility : A broad range of products catering to different performance and feature requirements.
- Performane : Offers robust processing power and advanced features for demanding applications.
- Low Power Consumption : Many variants are designed for energy-efficient, battery-powered devices.
- Extensive Ecosystem : A rich feature set and a strong support ecosystem, including tools like
- STM32CubeIDE : make development easier.
- Scalability: Designed to scale from low-cost, entry-level applications to powerful, high-end systems.

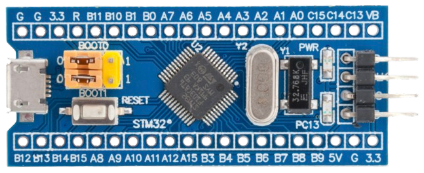

G G 3.3 R B11 B10 B1 B0 A7 A6 A5 A4 A3 A2 A1 A0 C15 C14 C13 VB
BOOT0
BOOT1
RESET
STM32
PC13
PWR
B12 B13 B14 B15 A8 A9 A10 A11 A12 A15 B3 B4 B5 B6 B7 B8 B9 5V G 3.3

Figure 9.2: STM32 Bluepill board

10 Communication Protocols

In embedded systems and microcontroller-based projects, communication protocols play a vital role in data transfer between devices. They define how data is formatted, transmitted, and received, ensuring reliable communication between sensors, controllers, and peripheral devices. Among the most widely used protocols are UART (Universal Asynchronous Receiver/Transmitter), I^2C (Inter-Integrated Circuit), and SPI (Serial Peripheral Interface).

10.1 UART

UART or Universal Asynchronous Receiver/Transmitter is a serial communication protocol that uses only two wires: TX (transmit) and RX (receive).It is asynchronous, meaning no clock signal is required; instead, data is framed with start and stop bits for synchronization.

Features :

- Simple and low-cost communication method.
- Point-to-point communication (usually between two devices).
- Data format typically includes 1 start bit, 8 data bits, optional parity bit, and 1 stop bit.

Applications :

- Debugging (Serial Monitor in Arduino, STM32, ESP32).
- Communication with modules like GPS, GSM, and Bluetooth.

Advantages :

- Easy to implement.
- No clock required.

Limitations

- Supports only two devices at a time.
- Lower speed compared to SPI and I^2C.

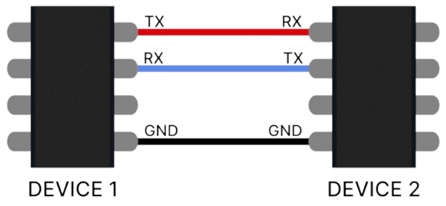


Figure 10.1: UART

10.2 I^2C

I^2C or Inter Integrated Circuit is a synchronous, multi-master, multi-slave protocol.Uses only two wires: SDA (Serial Data) and SCL (Serial Clock).Devices are addressed using unique 7-bit or 10-bit addresses.

Features :

- Can connect multiple devices on the same bus.
- Clock is controlled by the master device.
- Supports standard (100 kbps), fast (400 kbps), and high-speed (3.4 Mbps) modes.

Applications :

- Connecting sensors (e.g., MPU6050).
- Used when multiple peripherals need to communicate with a single controller.

Advantages :

- Requires only 2 wires for multiple devices.
- Address-based communication.

Limitations:

- Slower than SPI.

- Limited bus length and speed due to capacitance.

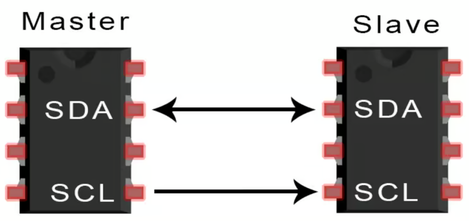


Figure 10.2: I^{2}C

10.3 SPI

Serial Peripheral Interface is a synchronous, full-duplex protocol commonly used for high-speed communication.

It uses 4 main wires: MISO (Master In, Slave Out), MOSI (Master Out, Slave In), SCLK (Serial Clock),CS/SS (Chip Select/Slave Select)

Features:

- Single master, multiple slaves (selected using CS pins).
- Faster data transfer rates compared to UART and I^{2}C.
- Data transmission is full-duplex (simultaneous send and receive).

Applications:

- SD cards, Flash memory, TFT displays, high-speed ADC/DACs.
- Used where speed is critical.

Advantages:

- Very high speed (tens of Mbps).
- Full-duplex communication.

Limitations:

- Requires more pins compared to UART and I^{2}C.
- No device addressing (each slave needs a separate CS pin).

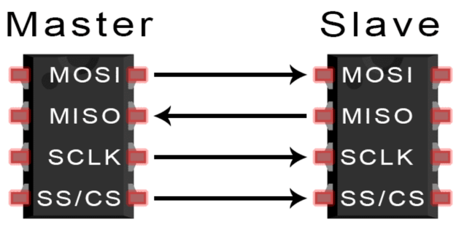


Figure 10.3: SPI

10.4 Conclusion

UART is best for simple, point-to-point communication.
I^{2}C is ideal when multiple devices need to share same bus with fewer pins.
SPI is preferred for high-speed applications where performance is critical.

Choosing the right communication protocol depends on project requirements such as speed, number of devices, distance, and hardware constraints.

11 Web GUI

This omnidirectional bot will also be controlled by a Web GUI interface using WebSocket and JavaScript in the backend. On the frontend , Html and CSS have been used.

11.1 Web Interface

A web interface is the visual and interactive part of a website or web application that users see and use to navigate, interact with, and consume digital content. It acts as the bridge between the user and the system, allowing them to perform actions like clicking buttons, filling out forms, and viewing information through a web browser or mobile app. The main objective is to provide a user-friendly and engaging experience and to operate the bot remotely without having the need for using PS4.

11.2 Languages Used

- **HTML :**
  HTML stands for HyperText Markup Language and is the standard language for creating web pages. It defines the structure and content of web pages by using tags to mark up text, images, links, and multimedia, instructing web browsers on how to display them. HTML serves as the fundamental building block of the web, providing the core structure that is then enhanced for appearance and interactivity by CSS and JavaScript respectively.

- **CSS :**
  Cascading Style Sheets (CSS) is used to format the layout of a webpage.With CSS, you can control the color, font, the size of text, the spacing between elements, how elements are positioned and laid out, what background images or background colors are to be used, different displays for different devices and screen sizes, and much more!

- **JavaScript :**
  JavaScript is a scripting or programming language that enables you to create dynamically updating content, control multimedia, animate images, and implement complex features on web pages .It is the third layer of the layer cake of standard web technologies, two of which are HTML and CSS.

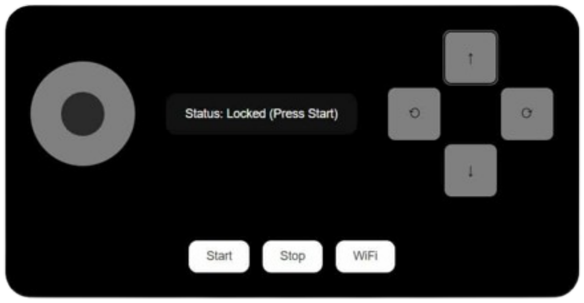


Figure 11.1: Web GUI Interface

- Start Button for connecting the interface to robot.
- Joystick used for giving direction and speed
- Buttons with rotate symbol is used for clockwise and anti-clockwise movement
- Up Button for increasing the speed
- Down Button for decreasing the speed

12 Web Socket

WebSocket is a communication protocol that enables full-duplex, real-time, bidirectional communication between a client (like a web browser) and a server over a single, long-lived connection. It starts with an HTTP handshake and then upgrades to a persistent WebSocket connection, allowing both the client(the phone) and server(ESP32) to send data at any time without needing to open a new connection for each message. WebSockets are ideal for applications requiring continuous data streams and server-pushed updates, such as live chat, gaming, financial feeds, and real-time collaboration tools.

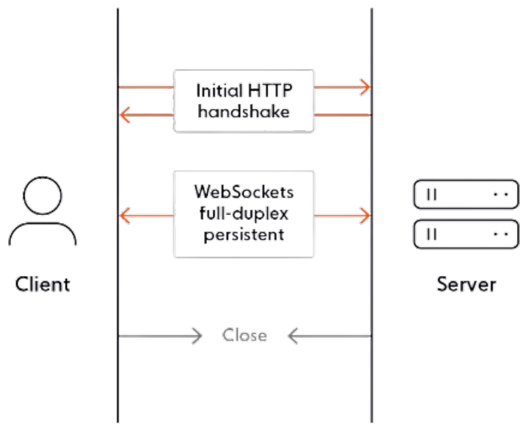


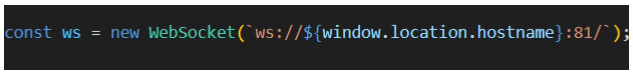
Figure 12.1: WebSocket Connection

12.1 HTTP

HTTP (Hypertext Transfer Protocol) is the core communication protocol for the World Wide Web, defining a client-server model where clients (like web browsers) send requests to servers for resources (like web pages or images), and servers send back responses. It's an application-layer protocol that facilitates the transfer of hypermedia documents and operates in a stateless manner, meaning each request is independent.

12.2 Connecting WebSocket Server

In order to communicate using the Web Socket protocol, we need to create a Web Socket object using the Web Socket constructor ; this will automatically attempt to open the connection to the server.

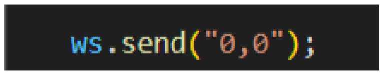


```
const ws = new WebSocket(`ws://${window.location.hostname}:81/`);
```

This automatically connects the browser to the ESP32 on port 81, using the current page's hostname and allows real-time, bidirectional communication between the browser and the ESP32.

12.2.1 Sending data to the server

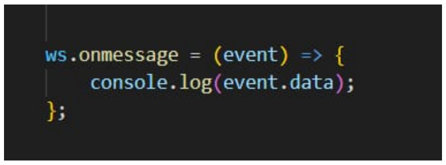
Once we 've opened our connection, we can begin transmitting data to the server. To do this, call the WebSocket object's send() method for each message we want to send:



```
ws.send("0,0");
```

12.2.2 Receiving messages from the server

WebSockets is an event-driven API; when messages are received, a message event is sent to the WebSocket object. To handle it,we have to add an event listener for the message event, or use the onmessage event handler. To begin listening for incoming data, we can do something like this:



```
ws.onmessage = (event) => {
    console.log(event.data);
};
```

13 IMU Sensor

An Inertial Measurement Unit (IMU) is defined as a 9 axis sensor that measures orientation, velocity, and gravitational forces by combining Accelerometer, Gyroscope, and Magnetometer into one. IMUs typically come in large packages, but with recent developments like MEMS technology, they are now more commonly seen as miniaturized sensors designed for easy integration with ESP32 or other microcontrollers.

- IMUs work by detecting the rotational movement of the 3 axis, commonly known as Pitch, Roll, and Yaw. To achieve such, it relies on the functionality of Accelerometers, Gyroscopes, and Magnetometers.

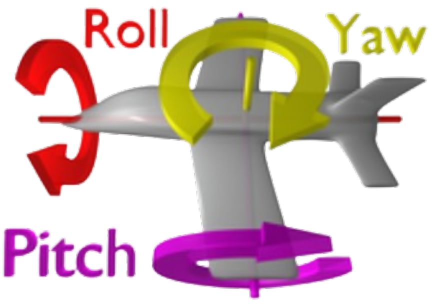


Figure 13.1: Three Degree of Freedom

- **Accelerometers**
  Accelerometers serve as the tool for velocity measurement on an IMU since we know its functionality is to detect the rate of change in velocity of an objec. Can only measure pitch and roll, no information about yaw

- **Gyroscopes**
  Gyroscopes serve as the tool for rotation/rotational rate measurement on an IMU since we know its functionality is to detect rotational changes or maintain orientation

- **Magnetometer**
  Magnetometer serves as the tool for gravitational force measurement on an IMU. Compared to an Accelerometer that can't measure yaw since it works on a constant gravitational force, a magnetometer is a great complement to accelerometer sensors

Combining these three sensors gives you a 9 D.O.F IMU that measures orientation, velocity, and gravitational force.

13.1 MPU6050

The MPU-6050 is a module with a 3-axis accelerometer and a 3-axis gyroscope.The gyroscope measures rotational velocity (rad/s). This is the change of the angular position over time along the X, Y, and Z-axis (roll, pitch, and yaw). This allows us to determine the orientation of an object.The accelerometer measures acceleration (rate of change of the object's velocity). It senses static forces like gravity (9.8m/s2) or dynamic forces like vibrations or movement. The MPU-6050 measures acceleration over the X, Y, and Z-axis. Ideally, in a static object, the acceleration over the Z-axis is equal to the gravitational force, and it should be zero on the X and Y-axis.

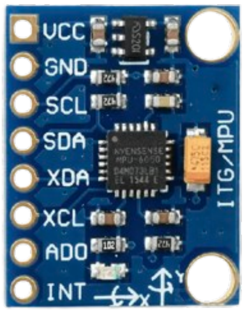


Figure 13.2: MPU6050

Using the accelerometer's values, it is possible to calculate the roll and pitch angles using trigonometry. However, it is not possible to calculate the yaw.

13.2 BNO085

The BNO085 is a compact, integrated motion-sensing module that combines a 3-axis accelerometer, gyroscope, and magnetometer with an ARM Cortex-M0 processor running CEVA's SH-2 firmware. It provides advanced sensor fusion for precise orientation and motion tracking data, such as rotation vectors and linear acceleration, making it suitable for applications like robotics, drones, and AR/VR. As an upgrade to the BNO080, it offers improved reliability and backward compatibility while maintaining low power consumption and a developer-friendly interface

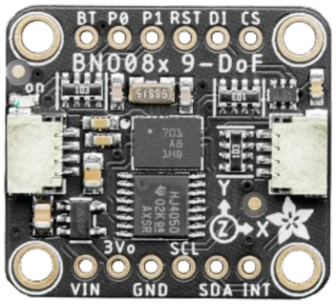


Figure 13.3: BNO085

The BNO085 integrates a 3-axis accelerometer, 3-axis gyroscope, and 3-axis magnetometer to provide 9 Degrees of Freedom (9-DOF). It features an ARM Cortex-M0 processor preloaded with CEVA's SH-2 firmware, which performs the complex sensor fusion, efficiently processing the raw sensor data.

13.3 MPU6050 or BNO055

- **Built-in Sensor Fusion**: BNO085 handles fusion internally, giving accurate orientation without extra code or processing
- **Absolute Heading**: Includes a magnetometer, so it knows its direction relative to the Earth's magnetic field—MPU6050 can't do this
- **Low Drift**: BNO085 maintains stable orientation over time, while MPU6050 drifts and needs frequent recalibration.
- **Simpler Integration**: BNO085 outputs clean orientation data (quaternions, Euler angles) directly—MPU6050 requires complex math
- **Better for Real-Time Control**: Fast, reliable data makes BNO085 ideal for dynamic movement and field-centric control.
- **Flexible Interfaces**: Supports I^{2}C and UART, making it easy to connect to most microcontrollers.

BNO085 is purpose-built for orientation tracking, while MPU6050 is a basic motion sensor that needs a lot of extra work to match BNO's capabilities.

14 Errors

Throughout the process of making this project, we encountered a variety of challenges and mistakes. This section details the key errors we faced, analyzing their causes and the steps we took to address or learn from them.

- Port Recognition : In the beginning, a lot of usbports were unable to detect the ESP32 connection. This was then solved by installing a driver CP210X which is a form USB to UART communication.
- PS4 connection : There were some errors while trying to connect PS4 with the microcontroller. This was resolved by using an example code (PS4Data) from the PS4 Library.
- Deadzone Correction : When PS4 was first connected to ESP32, we recieved a lot of garbage values. This was corrected by implementing normalisation.
- Delay in Web GUI : When the bot was first controlled through the web gui,it lagged a lot. This error was then corrected by using the function On websocketEvent.
- STM32 : We faced a lot of issues while trying to connect STM32 with the ESP32. Since its flash memory is very limited, it got full easily. We were unable to make it blink. When connected with ESP32, it was able to send its data, while it wasn't conclusive if STM32 could recieve anything or not.
- IMU Sensors : We tried implementing MPU6050 and BNO085 to ensure that the bot moved according to the directions provided to it via ps4 , irrespective of its configuration (i.e., w.r.t our orientation). Raw data was collected, but we faced errors while trying to implement it in our code. We faced a similar issue with both the sensors.
- Connecting BNO to PS4 : We faced problem during merging of BNO readings and our PS4 code due wrong pin defintions.
- Bot with BNO : We were able to implement it via breadboard but the Bot was facing lots of errors in terms of delay and processing data

15 Outcomes

This section summarizes the main results achieved throughout the project.Here, we outline our objectives and how closely they were met.

- Microcontrollers : Got to learn about ESP32, became familiar with its pin diagram, acquired hands-on experience with wireless communication, such as Wi-Fi and Bluetooth. Researched about STM32 and tried to gain an understanding of its architecture.
- Kinematics : Learned to establish a relationship between joint variables and end effectors using kinematic equations which introduced us to inverse kinematics.
- Web Socket and Web GUI : Built interactive and user responsive interface using HTML, CSS and JS and thus gained web development skills. Understood websocket and its advantages over the traditional HTTPS request.
- Implementation of Communication Protocols : We researched about UART, SPI and I^2C and tried implementing them in our project.
- IMU Sensor : We gave IMU sensor an integration a shot and got partial success.

16 Implementation Resources

1. Six-axis pair tool(to connect PS4)
2. Arudino IDE
3. Arduino libraries used :
   - ESP32 by espressif
   - Ps4Controller by Jaun Pablo
   - Web Sockets by Markus Sattler
   - Web Server
   - MPU6050 by Electronic Cats
   - BNO08x by Adafruit
4. STM32 Cube IDE
5. STM32 Cube Programmer
6. ST link server
7. Visual Code Studio
8. CodePen

References

[1] STMicroelectronics STM32F103C8 Datasheet

[2] Espressif ESP32-WROOM-32 datasheet

[3] Robokits India RMCS 2305 Datasheet

[4] Holonomic Implementation of Three Wheels

[5] Omnidirectional Mobile Robot using DC Motors

[6] Journal of Robotics and Control (JRC)

[7] Planetory Gear Motor Design and Application

[8] Texas Instruments LM2596 Datasheet

[9] Cytron Technologies MD10C Datasheet

[10] Adafruit 9-DOF Orientation IMU Fusion Breakout - BNO085 Datasheet