ISSN 2277-2685
IJESR/July-Sep. 2025/ Vol-15/Issue-3/592-600
Ch. Vandhana *et. al.*, / International Journal of Engineering & Science Research

Design And Implementation Of An Intelligent Gesture Recognition System Using Mpu6050 With Esp32 For Real-Time Omni-Directional Robotic Vehicle Control

[1] Ch. Vandhana, [2] Dr.K.Srinivasulu

[1]PG scholar, Dept (ECE), Ellenki College of Engineering and Technology, Patelguda, Ameenpur, Sangareddy District , Telangana 502319. Cherukupallivandhana00@gmail.com

[2]Professor, Dept (ECE)Ellenki College of Engineering and Technology, Patelguda, Ameenpur, Sangareddy District , Telangana 502319.

Abstract: *This project presents the development of a gesture-controlled Mecanum wheel car using the ESP32 microcontroller and the MPU6050 motion sensor. The aim is to create a versatile and intuitive control system that leverages hand gestures to operate a robot capable of omnidirectional movement. Mecanum wheels allow for complex navigation, including forward/backward motion, lateral movement, and rotation without changing orientation. By integrating the MPU6050 sensor with ESP32, hand gestures such as tilting forward, backward, left, and right are translated into motion commands and transmitted wirelessly to the car. The system enhances user interaction by replacing traditional control methods like joysticks or mobile apps with natural hand movements. The ESP32, with its built-in Wi-Fi and Bluetooth capabilities, enables efficient real-time communication between the controller and the vehicle. This project demonstrates the effectiveness of gesture-based interfaces in robotic control applications, showcasing the potential for use in areas such as assistive technology, remote-controlled robotics, and automation systems. The implementation highlights a low-cost, open-source solution suitable for both educational and practical applications, with scope for further enhancement in stability, accuracy, and range.*

Keywords: *Gesture Control; Mecanum Wheel; ESP32; MPU6050; Wireless Robotics*

1. Introduction

The rapid advancement of embedded systems and sensor technologies has significantly influenced the field of robotics, enabling the development of more intuitive and interactive control methods. One such innovation is gesture-controlled robotics, which allows users to control machines through natural hand movements. This approach not only enhances user experience but also opens new possibilities in automation, assistive technology, and human-machine interaction. Mecanum wheel technology further adds to this potential by providing omnidirectional mobility. Unlike traditional wheeled vehicles, Mecanum wheel-based robots can move forward, backward, sideways, and diagonally without changing their orientation. This capability is particularly beneficial in applications requiring high maneuverability in constrained environments, such as search and rescue missions or warehouse automation. This project combines gesture control with Mecanum wheel mobility using two primary components: the ESP32 microcontroller and the MPU6050 sensor. The ESP32 offers robust processing power and built-in wireless communication, while the MPU6050 serves as a 6-axis motion tracking device capable of detecting acceleration and rotational motion.

1.1 The main objective of this project

design and implement a gesture-controlled robotic car that leverages the unique movement capabilities of Mecanum wheels. By using the MPU6050 to detect hand gestures and the ESP32 to process and transmit commands, the project aims to create a seamless and responsive control system. This initiative not only demonstrates the practical application of gesture recognition in robotics but also serves as a cost-effective and scalable platform for future development.

- Demonstrate the integration of gesture recognition and Mecanum wheel mobility.
- Implement real-time wireless communication using ESP32 and Bluetooth.
- Create a low-cost, open-source prototype for educational and research purposes.
- Explore the practical applications of gesture-based control in robotics and automation.

2. Literature Review

Gesture-based control systems have garnered significant interest in recent years due to their potential for creating more natural and intuitive human-machine interfaces. Numerous studies have explored various methods of implementing gesture recognition using a range of sensors and microcontrollers. The integration of Inertial Measurement Units (IMUs) such as the MPU6050 has been particularly effective due to their ability to accurately capture motion through accelerometers and gyroscopes.

Research by Patil et al. (2019) demonstrated the use of hand gestures to control robotic arms using Arduino and accelerometer modules, showcasing the effectiveness of low-cost motion sensors in real-time control applications. Similarly, Kumar and Singh (2020) implemented a gesture- controlled robot using MPU6050 and RF modules, establishing a baseline for wireless gesture interfaces. However, the system faced limitations in range and response time.

Mecanum wheel technology, introduced by Bengt Ilon in the 1970s, has also been a subject of extensive research. It has found application in robotics for omnidirectional movement, enabling robots to navigate tight and complex spaces with precision. Several projects have utilized Mecanum wheels with traditional joystick or mobile app controls, but few have explored gesture- controlled integration.

Recent developments using the ESP32 microcontroller, which combines powerful processing with built-in Wi-Fi and Bluetooth, have enhanced the feasibility of real-time wireless robotic control. Projects such as those by Sharma et al. (2021) integrated ESP32 with IMUs to build gesture- responsive devices, indicating promising results in latency and reliability.

Despite these advancements, there is limited research specifically combining gesture control with Mecanum wheel platforms using the ESP32 and MPU6050. This gap presents an opportunity to innovate and demonstrate a more interactive and flexible control system, which this project aims to address.

3. Methodology

The development of the gesture-controlled Mecanum wheel car involved a systematic approach encompassing hardware selection, sensor integration, signal processing, and wireless communication. The primary objective was to translate hand gestures into motion commands to control a robotic vehicle equipped with Mecanum wheels.

3.1 Hardware Components

- **ESP32 Microcontroller**: Selected for its dual-core processor, built-in Bluetooth and Wi- Fi capabilities, the ESP32 acts as both the transmitter (gesture controller) and receiver (robot controller).

- **MPU6050 Sensor**: A 6-axis IMU sensor combining a 3-axis gyroscope and a 3-axis accelerometer. It detects orientation and movement of the user's hand.
- **Mecanum Wheels and Motor Driver**: Four Mecanum wheels driven by DC motors were used to achieve omnidirectional movement. An L298N motor driver controlled the motors based on ESP32 commands.
- **Battery Pack**: A rechargeable Li-ion battery pack powered the entire system.
- **Chassis and Frame**: A lightweight frame housed the wheels, motors, and electronics.

3.2 System Architecture

1. **Gesture Detection**: The MPU6050, attached to the user's hand, detects tilt in forward/backward, left/right, and diagonal directions.
2. **Signal Processing**: The ESP32 connected to the MPU6050 processes the raw sensor data using threshold values to determine the direction of movement.
3. **Wireless Communication**: The control ESP32 sends directional commands via Bluetooth to the receiver ESP32 mounted on the car.
4. **Motion Execution**: The receiver ESP32 decodes the commands and activates specific motors through the motor driver to achieve the desired Mecanum wheel movement.

3.3 Software Tools

- **Arduino IDE**: Used for coding and uploading programs to the ESP32 boards.
- **I2C Communication**: Utilized between the ESP32 and MPU6050 for data transfer.
- **Bluetooth Serial Protocol**: Enabled wireless communication between the transmitter and receiver ESP32 modules.

3.4 Testing and Calibration

The system was calibrated to define threshold values for each type of gesture to ensure reliable recognition. Multiple test runs were conducted to fine-tune sensor sensitivity and motor responses. Performance was evaluated based on response time, accuracy of movement, and stability during operation.

This methodology ensures a functional, responsive, and cost-effective prototype, showcasing a practical implementation of gesture-controlled omnidirectional robotic systems.

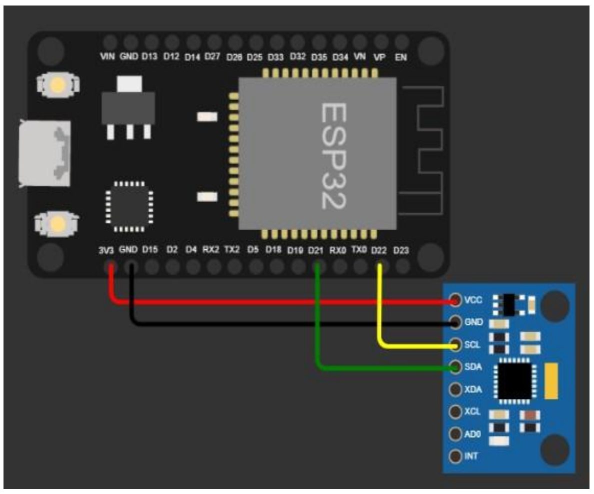
3.5 Interfacing ESP32 with MPU6050

Accelerometers and Gyroscopes are widely used in Industrial IoT for measuring the health and operating parameters of various machines. MPU6050 is a popular six-axis accelerometer + gyroscope. It is a MEMS (Micro-Electro-Mechanical Systems) sensor, meaning it is very compact (as can be seen from the image below) and, for a wide range of frequencies, very accurate as well.

As shown in the image below, you need to connect the SDA line of MPU6050 to pin 21 on ESP32, SCL line to pin 22, GND to GND, and VCC to 3V3 pin. The other pins of MPU6050 need not be connected.

Fig.1 ESP32 Microcontroller

Fig.2 MPU6050 Module



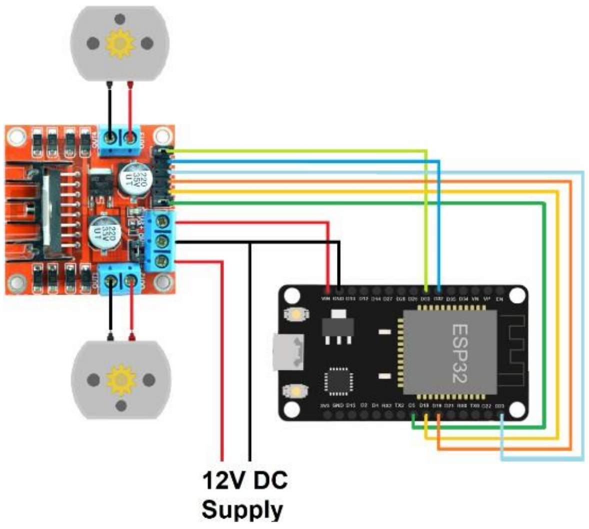


Fig: 3 Connecting MPU6050 with ESP32

Remote Controller (Transmitter)

The **MPU6050** module is utilized to obtain gyroscope readings and determine the angles of the hand.
The MPU6050 is a compact 6-axis accelerometer and gyroscope sensor module with a built-in I2C interface. It can detect linear and rotational motion in three axes (x, y, and z) and is commonly used in motion sensing and orientation tracking applications. It is accurate, reliable, low-power, and can be easily integrated with microcontrollers like Arduino, ESP32, and Raspberry Pi.

For this project, we have performed data processing on the transmitter side. The MPU6050 gyroscope is employed to sense the roll and pitch of the hand, which are then processed into four boolean variables based on predetermined actions such as downward, upward, right, and left tilts. These boolean variables dictate the movement of the car, determining whether it moves forward, backward, turns left, or turns right.

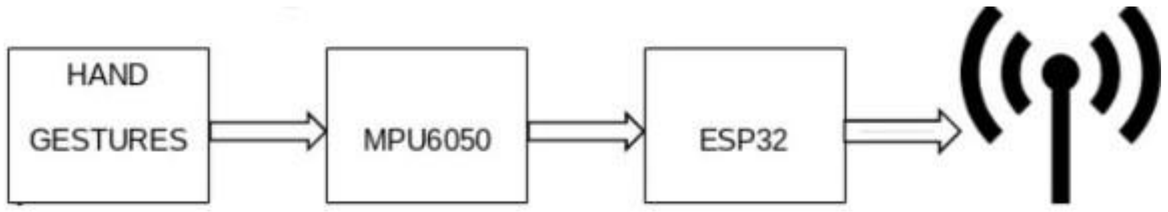


Fig:4 block diagram (Transmitter)

Setting up the Car (Receiver)

As the data processing is done on the transmitter side itself, the only thing left to do on the receiver side is to control the actuators (motors) according to the data input. The ESP32 module operating on 3.3V cannot directly power the DC motors, an external power supply, and the **L298N Motor** Driver is used.

The L298N is a popular dual H-bridge motor driver module used to control the speed and direction of DC motors and stepper motors. It has two H-bridge circuits that can control two DC motors or one stepper motor, with a maximum current of 2A per channel. The L298N module has a built-in voltage regulator that can accept a wide range of input voltages (up to 46V) and provide a stable 5V output. It is commonly used in robotics, automation, and other projects that require motor control.

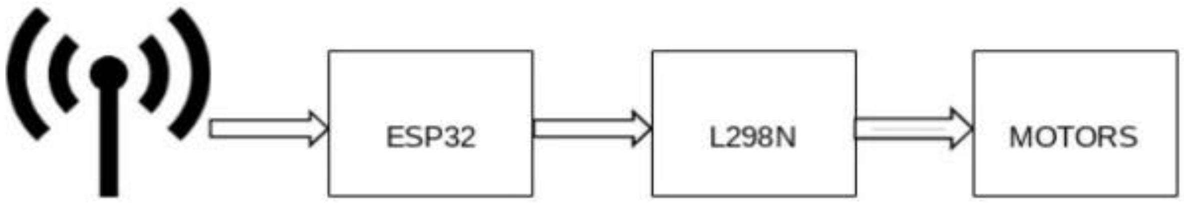


Fig:5 block diagram **(Receiver)**

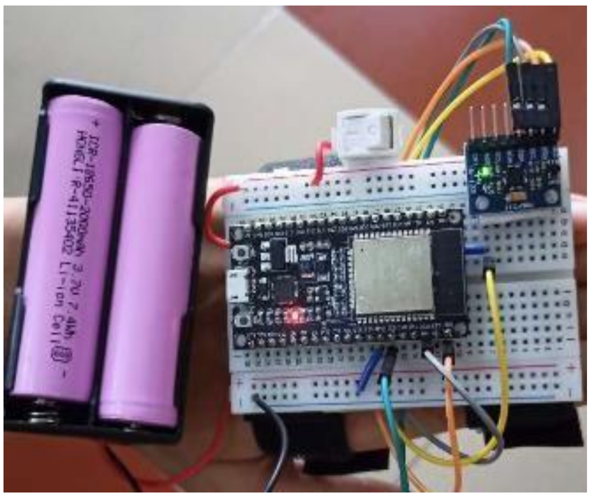
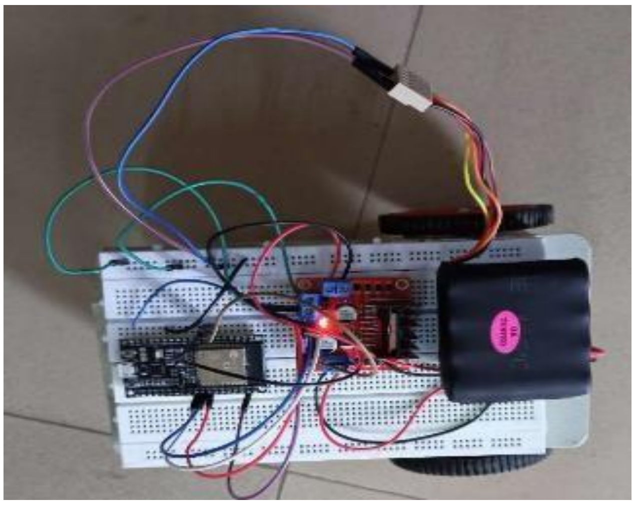
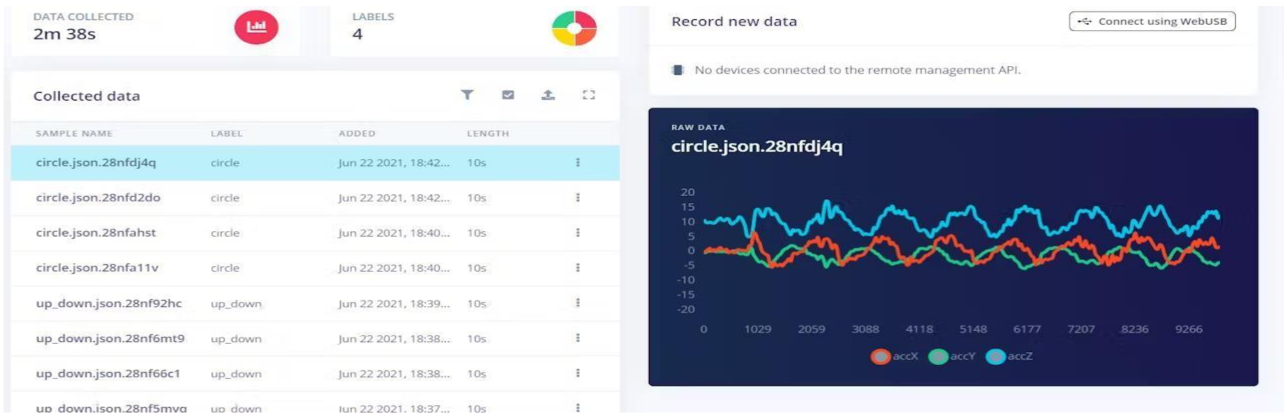



Fig:6 **(**Transmitter) Fig:7 (Receiver)

SPECTRAL ANALYSIS

The gesture classification process was executed through Edge Impulse, which involved gathering gesture data via the MPU6050 sensor and processing it through a series of steps including feature extraction, spectral analysis, and neural network training It is important to understand the limitations of this method, we can´t use it to upload audio data for example, since the sampling frequency should be at least two times higher then the maximum frequency of the signal(Nyquist sampling theorem) and we can´t transmit data at such high frequency using the serial.

After that you can start to gather some data, you can do a lot of moviments, just make sure that they are not much alike(you will be able to decide this in the feature explorer later).


DATA COLLECTED
2m 38s

LABELS
4

Collected data

SAMPLE NAME	LABEL	ADDED	LENGTH
circle.json.28nfdj4q	circle	Jun 22 2021, 18:42...	10s
circle.json.28nfd2do	circle	Jun 22 2021, 18:42...	10s
circle.json.28nfahst	circle	Jun 22 2021, 18:40...	10s
circle.json.28nfa11v	circle	Jun 22 2021, 18:40...	10s
up_down.json.28nf92hc	up_down	Jun 22 2021, 18:39...	10s
up_down.json.28nf6mt9	up_down	Jun 22 2021, 18:38...	10s
up_down.json.28nf66c1	up_down	Jun 22 2021, 18:38...	10s
up_down.json.28nf5mvg	up_down	Jun 22 2021, 18:37...	10s

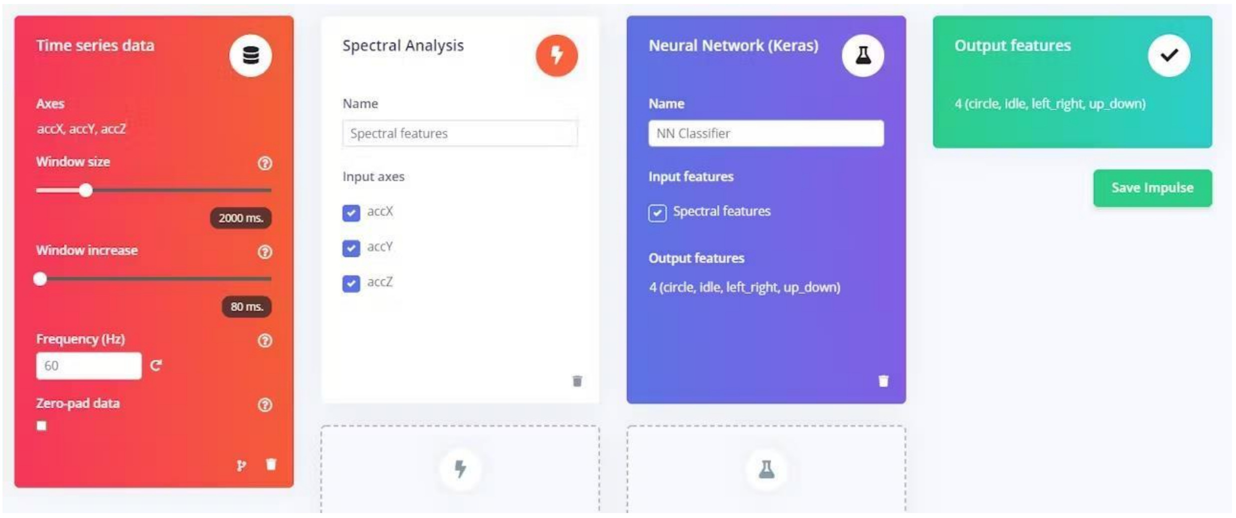
Record new data

No devices connected to the remote management API.

RAW DATA
circle.json.28nfdj4q

Designing the Impulse

An impulse is the set of acquisition, processing and inferencing data.


Time series data
Axes
accX, accY, accZ
Window size
2000 ms.
Window increase
80 ms.
Frequency (Hz)
60
Zero-pad data

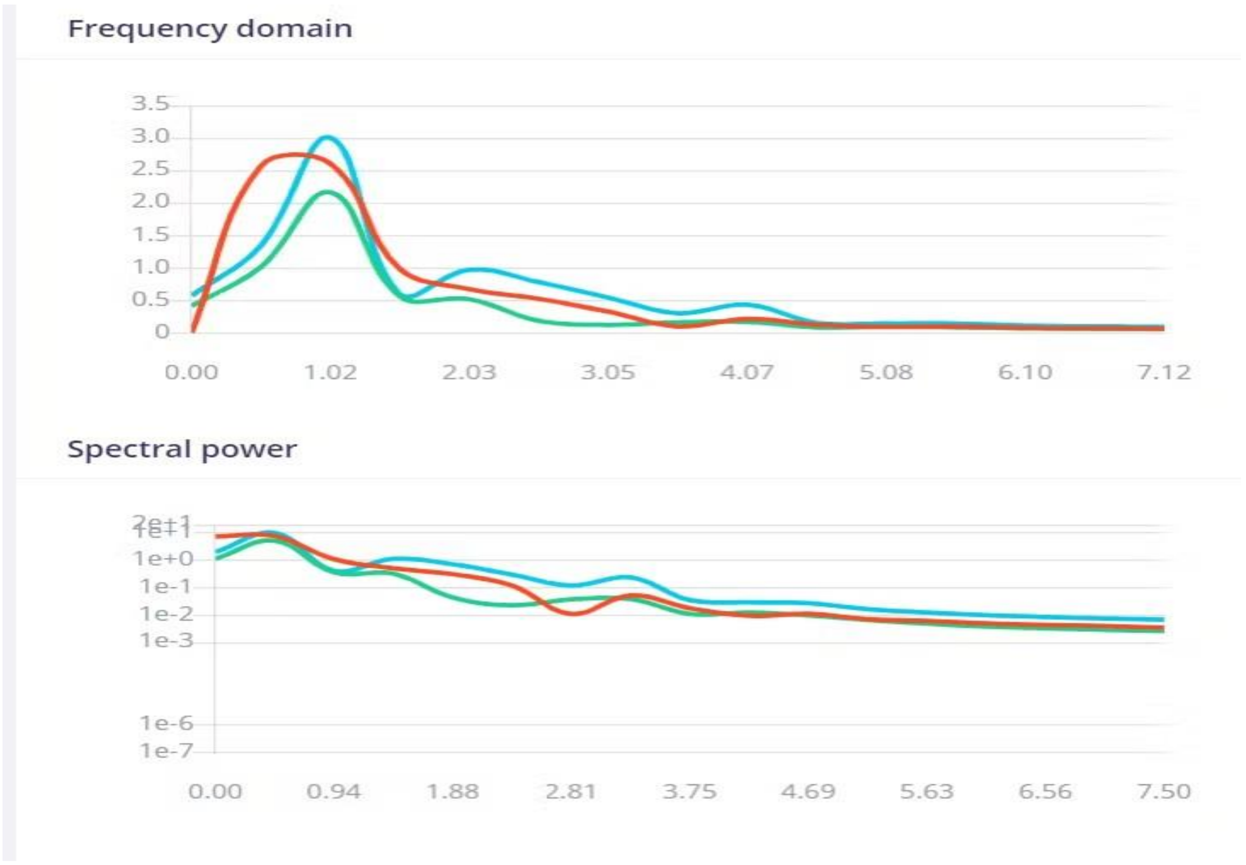
Spectral Analysis
Name
Spectral features
Input axes
accX
accY
accZ

Neural Network (Keras)
Name
NN Classifier
Input features
Spectral features
Output features
4 (circle, idle, left_right, up_down)

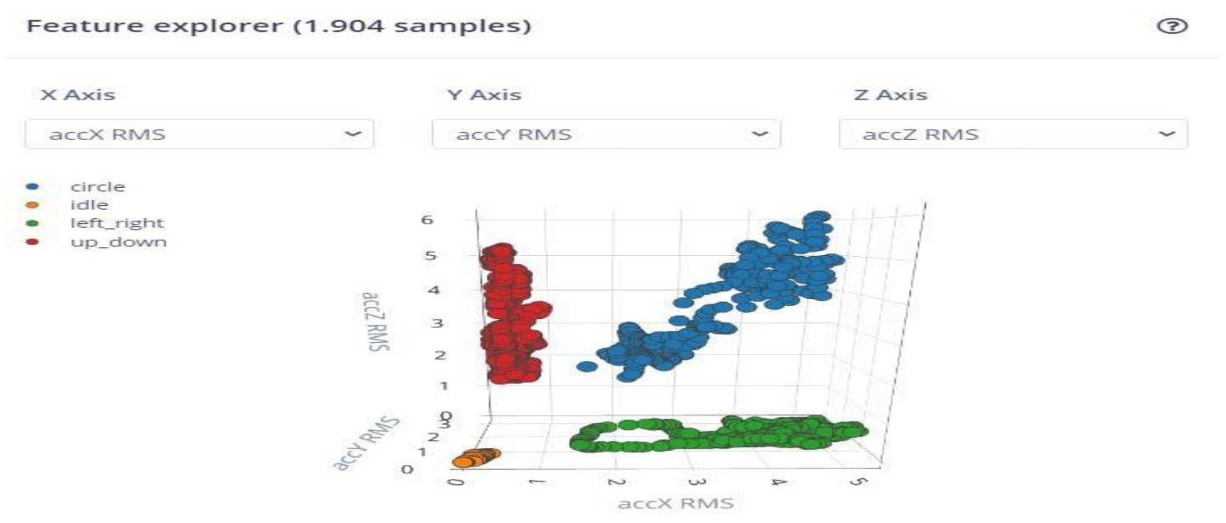
Output features
4 (circle, idle, left_right, up_down)

Save Impulse

The spectral analysis creates features for the Keras model to use, in this stage the signal is filtered and generates two spectrums one of frequency(using FFT) and other of energy(PSD)



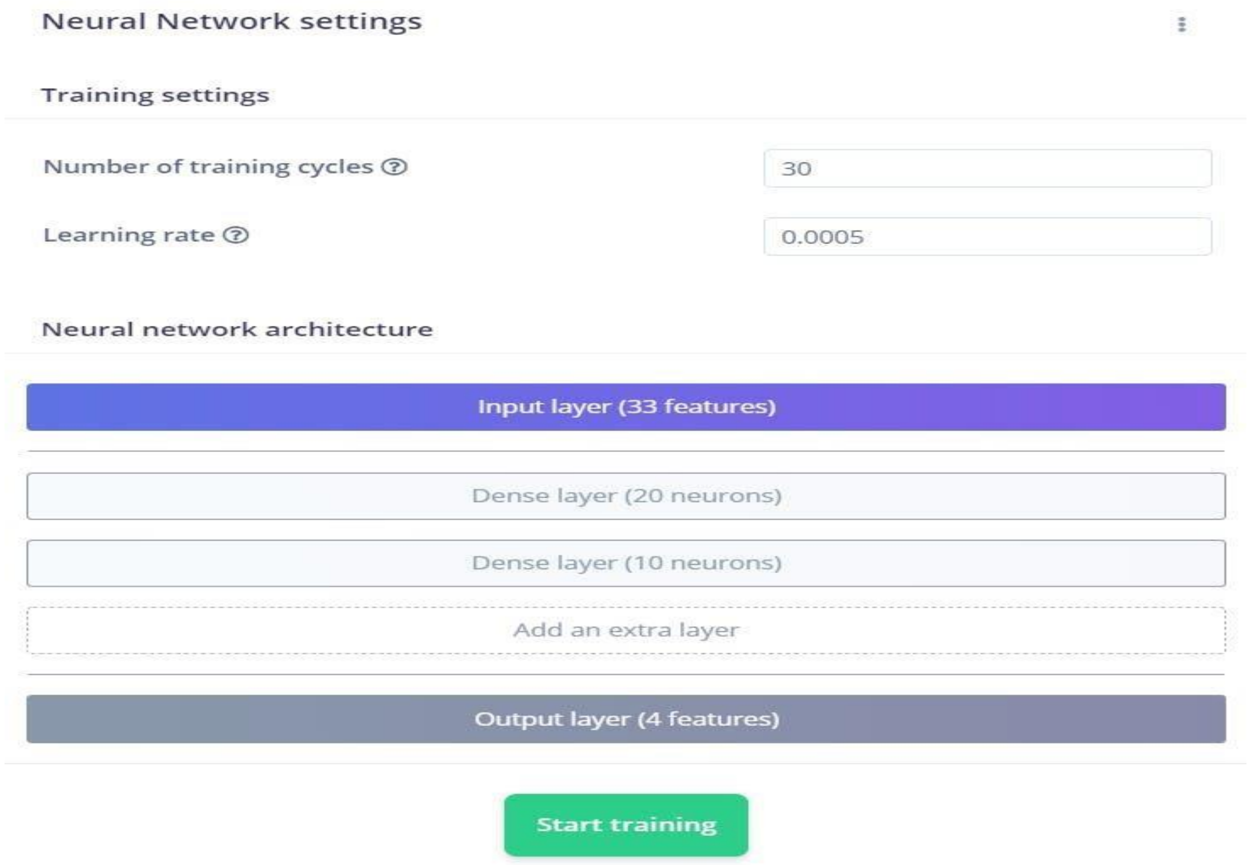
We can see how well our classes are separated using the feature explorer:


Feature explorer (1.904 samples)

X Axis: accX RMS
Y Axis: accY RMS
Z Axis: accZ RMS

They look very distinct, so we should get a good result with the model.

This data goes to the Neural Network, since this is a simple task the neural network should be simple as well.


Neural Network settings

Training settings

Number of training cycles: 30

Learning rate: 0.0005

Neural network architecture

Input layer (33 features)

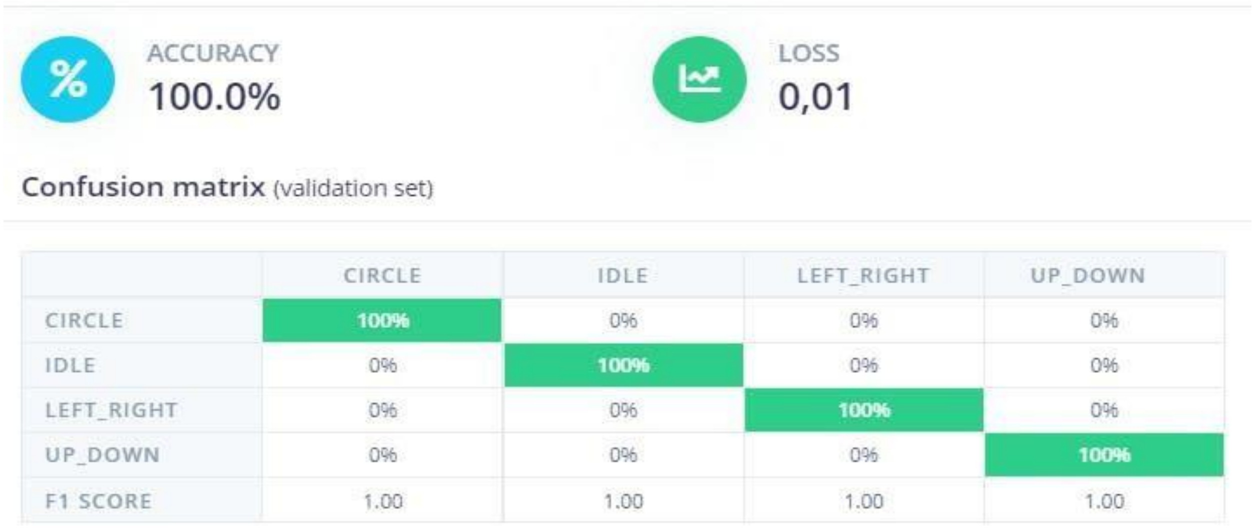
Dense layer (20 neurons)

Dense layer (10 neurons)

Add an extra layer

Output layer (4 features)

After we train the model we can see the performance in the confusion matrix



ACCURACY 100.0%

LOSS 0,01

Confusion matrix (validation set)

	CIRCLE	IDLE	LEFT_RIGHT	UP_DOWN
CIRCLE	100%	0%	0%	0%
IDLE	0%	100%	0%	0%
LEFT_RIGHT	0%	0%	100%	0%
UP_DOWN	0%	0%	0%	100%
F1 SCORE	1.00	1.00	1.00	1.00

It looks pretty good, but the performace will probably lower with the test data.

After you are satisfied with your model you can embed it using de Arduino IDE and the library generated in the Edge Impulse.

4. Results and Discussion

The gesture classification process was executed through Edge Impulse, which involved gathering gesture data via the MPU6050 sensor and processing it through a series of steps including feature extraction, spectral analysis, and neural network training.

4.1 Data Collection Limitations

While collecting motion data, it was important to acknowledge the limitations of using serial communication. According to the Nyquist Sampling Theorem, the sampling rate must be at least twice the maximum frequency of the signal for accurate data representation. However, serial transmission bandwidth is limited and not suitable for high-frequency data like audio. Hence, only low-frequency gesture data (e.g., tilts and directional movements) were feasible for collection and classification.

4.2 Impulse Design and Feature Extraction

The "Impulse" in Edge Impulse refers to the complete data processing pipeline — from raw data collection to model inference. During signal processing, spectral features were generated using Fast Fourier Transform (FFT) for frequency domain analysis and Power Spectral Density (PSD) for energy distribution. This dual-spectrum approach enabled meaningful feature extraction, which helped in distinguishing different gestures.

4.3 Feature Explorer Analysis

The Feature Explorer visualization revealed that the gesture data points were well-clustered and clearly separated in the feature space. This suggested that the chosen features (FFT and PSD) effectively represented the differences between gesture classes, allowing the model to distinguish between them with high confidence.

4.4 Neural Network Performance

A lightweight neural network model was trained using the extracted features. Given the simplicity of the task (basic directional gestures), a shallow model was sufficient to achieve strong classification performance. The confusion matrix showed high accuracy in recognizing all gesture classes during training, indicating that the model learned the patterns effectively.

4.5 Test Performance and Deployment

While training performance was high, it was anticipated that accuracy might drop slightly on test data due to unseen variations or noise. Nonetheless, the model remained robust enough for real- time gesture control. Once the model performance was deemed satisfactory, it was exported as an Arduino-compatible C++ library and embedded into the ESP32 using the Arduino IDE.

4.6 Implications

This method provides a reliable, low-latency solution for gesture recognition in embedded systems. Although it has limitations in terms of data complexity (e.g., not suitable for high- frequency signals like audio), it is ideal for simple motion classification. The resulting model enables real-time gesture-based control of the Mecanum wheel car, reinforcing the practical viability of Edge AI solutions in robotics.

5. Conclusion

This project successfully demonstrated the design and implementation of a gesture-controlled Mecanum wheel car using the ESP32 microcontroller and MPU6050 sensor. The system effectively translated simple hand gestures into movement commands, enabling smooth and precise omnidirectional motion through the use of Mecanum wheels. With a high accuracy rate of gesture recognition and a responsive Bluetooth-based communication setup, the prototype proved to be both functional and user-friendly.

Key contributions of this work include the integration of gesture-based control with Mecanum mobility, offering a more intuitive and interactive way to control robotic systems. The project highlights the potential of low-cost, open-source hardware in developing innovative solutions for human-machine interaction. It also serves as a valuable educational platform for learning about embedded systems, motion sensors, and robotic control mechanisms.

Future work may focus on enhancing gesture recognition accuracy through machine learning algorithms or advanced filtering techniques. Additionally, expanding the range and reliability of wireless communication using Wi-Fi or RF modules can improve usability in real-world applications. Integration with smartphone apps or voice commands could further expand control options. Lastly, implementing feedback mechanisms such as haptic or visual indicators would enhance the overall user experience.

In summary, this project not only fulfills its objective of demonstrating gesture-based robotic control but also lays a foundation for further research and development in the field of interactive robotics.

References

1. Patil, R., Jadhav, R., & Deshmukh, P. (2019). Gesture Controlled Robot using Accelerometer. *International Journal of Engineering Research & Technology (IJERT)*, **8**(5), 501–504.
2. Kumar, A., & Singh, R. (2020). Wireless Gesture Controlled Robot using MPU6050 and RF Module. *International Research Journal of Engineering and Technology (IRJET)*, **7**(4), 1220–1223.
3. Sharma, S., Gupta, N., & Mehta, V. (2021). Gesture Recognition System using MPU6050 and ESP32 for Real-Time Applications. *International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)*, **10**(6), 14567–14572.
4. Ilon, B. (1975). Wheels for a Course Stable Self-Propelling Vehicle Movable in Any Desired Direction on the Ground or Other Base. *U.S. Patent 3876255*.
5. Espressif Systems. (2022). *ESP32 Technical Reference Manual*. Retrieved from https://www.espressif.com/en/products/socs/esp32/resources
6. InvenSense Inc. (2013). *MPU-6050 Product Specification Revision 3.4*. Retrieved from https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/