

الجمهورية الجزائرية الديمقراطية الشعبية

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم العالي والبحث العلمي

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA  
RECHERCHE SCIENTIFIQUE

جامعة سعيدة - د. مولاي الطاهر

UNIVERSITE DE SAÏDA DR MOULAY TAHAR



Faculté de technologie

Département d'électronique

MEMOIRE DE FIN D'ETUDES EN VUE DE L'OBTENTION

DU DIPLOME DE MASTER EN ELECTRONIQUE

OPTION : INSTRUMENTATION

THEME :

---

COMMANDE GESTUELLE A DISTANCE D'UNE VOITURE ROBOT BASEE  
SUR L'ESP32

---

**Présenté par :**

MEBREK Ahmed Houssam Eddine

LAMARI Youcef

**Soutenu le 24 Juin 2024, Devant le jury composé de :**

CHAMI Nadir	Maître de conférences à l'Université de Saïda	Président
BOUARFA Abdelkader	Maître de conférences à l'Université de Saïda	Examineur
MAACHOU Fatima	Maître de conférences à l'Université de Saïda	Encadrante

**Année Universitaire : 2023 - 2024**

الجمهورية الجزائرية الديمقراطية الشعبية

DEMOCRATIC AND POPULAR ALGERIAN REPUBLIC

وزارة التعليم العالي والبحث العلمي

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

جامعة سعيدة - د. مولاي الطاهر

UNIVERSITY OF SAIDA DR MOULAY TAHAR



Faculty of Technology

Department of electronics

**FINAL THESIS FOR THE ATTAINMENT OF A MASTER'S  
DEGREE IN ELECTRONICS**

**OPTION : INSTRUMENTATION**

**THEME :**

---

---

**HAND GESTURE REMOTE CONTROL OF A ROBOT CAR BASED ON  
ESP32**

---

**Presented by :**

MEBREK Ahmed Houssam Eddine

LAMARI Youcef

**Defended on June 24, 2024, in front of the jury composed of:**

CHAMI Nadir	Lecturer at the University of Saida	President
BOUARFA Abdelkader	Lecturer at the University of Saida	Reviewer
MAACHOU Fatima	Lecturer at the University of Saida	Supervisor

**Academic Year : 2023 - 2024**

# Abstract

This project focuses on developing a wireless hand gesture-controlled robot car using the ESP32 microcontroller and ESPNOW communication protocol. The system uses the ESP32's Wi-Fi capabilities to enable real-time, reliable communication between a gesture recognition module and the robot car. Hand movements are detected by an accelerometer and gyroscope, which interpret gestures and convert them into commands sent via ESPNOW. The robot car receives these commands and navigates accordingly, providing an intuitive and responsive control experience without the need for traditional controllers. This implementation emphasizes low power consumption, cost-effectiveness, and ease of integration, making it suitable for various applications in robotics and human-machine interaction. The project demonstrates the potential of combining gesture recognition with wireless communication to enhance user interfaces and control systems.

**Keywords :** ESP32, ESPNOW Protocol, MPU-6050, Accelerometer, Gyroscope, L298N, Mecanum Wheels.

# Résumé

Ce projet se concentre sur le développement d'une voiture robot sans fil contrôlée par gestes manuels utilisant le microcontrôleur ESP32 et le protocole de communication ESPNOW. Le système utilise les capacités Wi-Fi de l'ESP32 pour permettre une communication fiable et en temps réel entre un module de reconnaissance gestuelle et la voiture robot. Les mouvements des mains sont détectés par un accéléromètre et un gyroscope, qui interprètent les gestes et les convertissent en commandes envoyées via ESPNOW. La voiture robot reçoit ces commandes et navigue en conséquence, offrant une expérience de contrôle intuitive et réactive sans avoir besoin de contrôleurs traditionnels. Cette mise en œuvre met l'accent sur une faible consommation d'énergie, la rentabilité et la facilité d'intégration, ce qui la rend adaptée à diverses applications en robotique et en interaction homme-machine. Le projet démontre le potentiel de combiner la reconnaissance gestuelle avec la communication sans fil pour améliorer les interfaces utilisateur et les systèmes de contrôle.

**Mots clés:** ESP32, Protocole ESPNOW, MPU-6050, Accéléromètre, Gyroscope, L298N, roues Mecanum.

## المخلص :

يركز هذا المشروع على تطوير سيارة روبوتية لاسلكية يتم التحكم فيها عن طريق إيماءات اليد باستخدام وحدة التحكم الدقيقة ESP32 وبروتوكول الاتصال ESPNOW. يستخدم النظام إمكانات Wi-Fi الخاصة بـ ESP32 لتمكين الاتصال الموثوق به في الوقت الفعلي بين وحدة التعرف على الإيماءات والسيارة الآلية. يتم اكتشاف حركات اليد بواسطة مقياس التسارع والجيروسكوب، الذي يفسر الإيماءات ويحولها إلى أوامر يتم إرسالها عبر ESPNOW. تتلقى السيارة الآلية هذه الأوامر وتتنقل وفقاً لذلك، مما يوفر تجربة تحكم بديهية وسريعة الاستجابة دون الحاجة إلى وحدات تحكم تقليدية. يؤكد هذا التنفيذ على انخفاض استهلاك الطاقة، وفعالية التكلفة، وسهولة التكامل، مما يجعله مناسباً لمختلف التطبيقات في مجال الروبوتات والتفاعل بين الإنسان والآلة. يوضح المشروع إمكانية الجمع بين التعرف على الإيماءات والاتصال اللاسلكي لتحسين واجهات المستخدم وأنظمة التحكم.

**كلمات البحث:** ESP32، بروتوكول ESPNOW، MPU-6050، مقياس التسارع، الجيروسكوب، L298N، عجلات ميكانوم.

# Acknowledgements

*We would like to express our deepest thanks to our supervisor Dr.MAACHOU Fatima for her support and her availability throughout this dissertation through the work sessions organized.*

*We thank the members of the jury for their interest in our work. Our thanks also go to all of our teachers who have contributed to our training.*

*We extend my heartfelt thanks to my parents. Your unwavering love, guidance, and encouragement have been the foundation of my success. Your belief in my abilities has always inspired me to strive for excellence. Thank you for your endless sacrifices and for always being there for me.*

# Dedication

*This work is dedicated to my parents, whose unwavering love, support, and guidance have been my greatest source of strength. Your belief in me has been a constant source of inspiration.*

*To my brothers, for always standing by me, encouraging me, and being my greatest allies. Your support and camaraderie have been invaluable.*

*And to my friends, for their steadfast support, understanding, and encouragement. Your presence in my life has been a blessing.*

*Thank you all for being my pillars of strength.*

*Houssam*

# Dedication

*This work is dedicated to my parents, whose unwavering love, support, and guidance have been my greatest source of strength. Your belief in me has been a constant source of inspiration.*

*To my brothers, for always standing by me, encouraging me, and being my greatest allies. Your support and camaraderie have been invaluable.*

*And to my friends, for their steadfast support, understanding, and encouragement. Your presence in my life has been a blessing.*

*Thank you all for being my pillars of strength.*

*Youcef*

# Contents

<b>Introduction</b>	<b>1</b>
<b>1 Study of ESP32 card development</b>	<b>2</b>
1.1 Introduction . . . . .	2
1.2 ESP32 Module . . . . .	2
1.2.1 Features and specifications of ESP32 . . . . .	3
1.2.2 Description of ESP32 . . . . .	4
1.3 ESPNOW Protocol . . . . .	5
1.3.1 Features of ESP-NOW: . . . . .	6
1.3.2 How ESP-NOW Works: . . . . .	6
1.3.3 Applications of ESP-NOW: . . . . .	7
1.3.4 One-Way Communication: . . . . .	7
1.3.5 Two-Way Communication: . . . . .	8
1.3.6 ESP-NOW One Board to Multiple Board Communication . . . . .	8
1.3.7 ESP-NOW Multiple Board to One Board Communication . . . . .	9
1.3.8 The MAC address . . . . .	10
1.3.9 Wireless networking modes . . . . .	10
1.4 Conclusion . . . . .	12
<b>2 MPU-6050 Accelerometer and Gyroscope Module</b>	<b>13</b>
2.1 Introduction . . . . .	13
2.2 MPU-6050 Accelerometer and Gyroscope Module . . . . .	13
2.2.1 3-Axis Gyroscope . . . . .	14
2.2.2 3-Axis Accelerometer . . . . .	14
2.2.3 DMP (Digital Motion Processor) . . . . .	15
2.2.4 On-chip Temperature Sensor . . . . .	16
2.2.5 MPU6050 Module Pinout . . . . .	16
2.3 Interfacing ESP32 with MPU-6050 . . . . .	17
2.4 Reading Accelerometer, Gyroscope, and Temperature Data with MPU6050 and ESP32	18
2.4.1 Installing Libraries . . . . .	18
2.4.2 MPU-6050 readings sensors flowchart . . . . .	19
2.4.3 MPU6050 Output on Serial Monitor . . . . .	21
2.5 Conclusion . . . . .	21
<b>3 Wireless Hand Gesture Controlled Robot Car Using ESP32 with ESPNOW</b>	<b>22</b>
3.1 Introduction . . . . .	22
3.2 Interfacing the L298N Module with ESP32 . . . . .	22
3.3 Introducing the L298N "Motor Driver" . . . . .	22

# Contents

---

3.3.1	Specifications . . . . .	23
3.3.2	Pinout . . . . .	23
3.3.3	L298N working principle . . . . .	24
3.4	DC motors . . . . .	27
3.4.1	Constitution . . . . .	27
3.4.2	Principle of operation . . . . .	27
3.5	The Wheels . . . . .	28
3.5.1	Mecanum Wheels . . . . .	28
3.6	Controlling DC motors through L298N Driver Module . . . . .	32
3.6.1	Control Pins . . . . .	32
3.7	DC Motor control by ESP32 via L298N . . . . .	34
3.8	ESPNOW Protocol . . . . .	38
3.8.1	Getting Media Access Control (MAC) Address from ESP32 receiver . . . . .	38
3.8.2	ESP32 sender flowchart . . . . .	39
3.8.3	ESP32 receiver flowchart . . . . .	42
3.9	Wireless hand gesture controlled robot car via ESPNOW . . . . .	44
3.9.1	Sender part . . . . .	44
3.9.2	Receiver part . . . . .	47
3.10	Conclusion . . . . .	53
	<b>Conclusion</b>	<b>54</b>
	<b>References</b>	<b>55</b>
	<b>A Datasheet of L298N</b>	<b>57</b>
	<b>B Datasheet of MPU-6050 module</b>	<b>62</b>



# List of Figures

1.1	The ESP32 development board. . . . .	2
1.2	The ESP32 function Block diagram(SoC). . . . .	3
1.3	ESP32 Board Highlights. . . . .	5
1.4	ESP32 Pinout Diagram. . . . .	5
1.5	ESP-NOW Communication Architecture. . . . .	6
1.6	One-Way Communication. . . . .	7
1.7	Two-Way Communication. . . . .	8
1.8	ESP-NOW One-to-Multiple Board Communication. . . . .	9
1.9	ESP-NOW Multiple-to-One Board Communication. . . . .	10
1.10	STA Mode in ESP32 Wireless Networking. . . . .	11
1.11	AP Mode in ESP32 Wireless Networking. . . . .	11
2.1	MPU6050 Module. . . . .	14
2.2	MPU6050 orientation and polarity of rotation. . . . .	14
2.3	Angles of Spherical Coordinate System. . . . .	15
2.4	MPU6050 Module Pinout. . . . .	16
2.5	Schematic diagram ESP32 with MPU-6050. . . . .	17
2.6	Circuit of ESP32 interfacing with MPU-6050. . . . .	18
2.7	Flowchart for getting MPU-6050 readings sensors. . . . .	20
2.8	Serial Monitor Output of MPU 6050 Sensor Data. . . . .	21
3.1	L298N motor driver Module. . . . .	23
3.2	L298N Module Pinout. . . . .	24
3.3	Internal diagram of the L298N. . . . .	25
3.4	H-bridge switch working. . . . .	25
3.5	Pulse Width Modulation (PWM) Switch. . . . .	26
3.6	Visual image of a DC motor. . . . .	27
3.7	Mecanum Wheel Design. . . . .	28
3.8	Installing mecanum wheels position. . . . .	29
3.9	Straight direction. . . . .	29
3.10	Sideway direction. . . . .	30
3.11	Diagonal direction. . . . .	30
3.12	Pivot direction. . . . .	31
3.13	Rotate direction. . . . .	31
3.14	Pivot Sideways direction. . . . .	32
3.15	Speed Control Pins. . . . .	33
3.16	Direction Control Pins. . . . .	34

## List of Figures

---

3.17 Schematic diagram of DC motor control by ESP32 via L298N. . . . .	35
3.18 Circuit of DC motor control by ESP32 via L298N. . . . .	36
3.19 Implementation of Control DC Motors via L298N using ESP32. . . . .	37
3.20 Circuit of sender receiver ESPNOW communication. . . . .	38
3.21 Flowchart for getting MAC Address of ESP32 receiver. . . . .	39
3.22 The MAC address of Receiver . . . . .	39
3.23 Sender Flowchart. . . . .	41
3.24 Serial Monitor Output of Sender. . . . .	42
3.25 Receiver Flowchart. . . . .	43
3.26 Serial Monitor Output of receiver. . . . .	44
3.27 Synoptic diagram of Wireless hand gesture controlled robot car via ESPNOW. . .	44
3.28 Flowchart of sender. . . . .	46
3.29 Schematic Diagram of Car Receiver. . . . .	47
3.30 DC Motors Installation. . . . .	48
3.31 DC Attaching Wheels. . . . .	48
3.32 Mounting a battery. . . . .	49
3.33 Mounting L298N Module. . . . .	49
3.34 Installing ESP32 board into Robot Car's chassis. . . . .	50
3.35 Receiver car Flowchart. . . . .	52
3.36 Serial Monitor of the mapped angle data. . . . .	53

# List of Tables

2.1	Pin mapping between ESP32 and MPU6050. . . . .	17
3.1	L298N Module Specifications. . . . .	23
3.2	Truth Table of The "Motor" Speed . . . . .	33
3.3	Truth Table of the "Motor" Direction. . . . .	34
3.4	The pin mapping between ESP32 and L298N. . . . .	35
3.5	The pin mappings for the ESP32 with Front and Back L298N motor drivers. . . .	47

# Abbreviations

<b>SoC</b>	System-on-Chip
<b>IoT</b>	Internet of Things
<b>Wi-Fi</b>	Wireless Fidelity
<b>DMIPS</b>	Dhrystone Million Instructions Per Second
<b>GPIO</b>	General Purpose Input/Output
<b>LED</b>	Light Emitting Diode
<b>ADC</b>	Analog-to-Digital Converter
<b>DAC</b>	Digital-to-Analog Converter
<b>I2C</b>	Inter-Integrated Circuit
<b>UART</b>	Universal Asynchronous Receiver/Transmitter
<b>CAN</b>	Controller Area Network
<b>SPI</b>	Serial Peripheral Interface
<b>I2S</b>	Integrated Inter-IC Sound
<b>RMII</b>	Reduced Media-Independent Interface
<b>PHY</b>	Physical Layer
<b>PWM</b>	Pulse Width Modulation
<b>DMA</b>	Direct Memory Access
<b>ROM</b>	Read Only Memory
<b>SRAM</b>	Static Random Access Memory
<b>IC</b>	Integrated Circuit
<b>SDA</b>	Serial Data
<b>SCL</b>	Serial Clock
<b>MAC</b>	Media Access Control
<b>EPROM</b>	Erasable Programmable Read-Only Memory
<b>ESP-NOW</b>	Espressif Simple Peripheral Network Over Wi-Fi
<b>MEMS</b>	Micro Electro Mechanical Systems
<b>DMP</b>	Digital Motion Processor

## Abbreviations

---

<b>STA</b>	Station
<b>AP</b>	Access Point
<b>VR</b>	Virtual Reality
<b>AR</b>	Augmented Reality
<b>ESP</b>	Motion Processing Unit
<b>MPU</b>	Espressif System's Product

# Introduction

Today, a discernible shift is occurring from traditional electronics towards programmed electronics, often termed as embedded systems or embedded computing. The objective is to simplify electronic diagrams, decrease the use of electronic components, and subsequently reduce manufacturing costs. This shift results in the development of more intricate and efficient systems within smaller spaces. Since its inception, the growth of electronics has been remarkable and continues unabated. Electronics has become accessible to anyone with a desire to learn, blending elements of electronics and programming.

In this project, our aim is to integrate our expertise in embedded electronics with programming to design a mobile robot controlled remotely through gestures. The project consists of two primary components: a mobile wheeled robot and its remote gesture control unit, both of which communicate via WiFi communication modules.

The code to be implemented on the ESP32 card of the control unit must detect hand gestures by receiving information from the inertial measurement unit MPU 6050. Subsequently, this data is translated into forward, backward, and rotation commands which will be transmitted to the robot via WiFi communication. On the other side, the robot receives the commands, converts them into signals, and sends them to the motor drivers to carry out the required commands .

In this manuscript, we present the essential components of this work, which will be structured as follows:

- The first chapter delves into the development of the ESP32 card, highlighting its features, specifications, and the ESP-NOW protocol, which enables efficient wireless communication between devices.
- The second chapter focuses on integrating the ESP32 with the MPU-6050 accelerometer and gyroscope module, demonstrating how to read sensor data and display it via a serial monitor.
- The final chapter showcases the development of a wireless hand gesture-controlled car robot using the ESP32, L298N motor driver, and DC motors, facilitated by the ESPNOW protocol for seamless communication.

# Chapter 1

## Study of ESP32 card development

### 1.1 Introduction

The ESP32 is a powerful, low-cost microcontroller with built-in Wi-Fi and Bluetooth capabilities, making it an ideal choice for a wide range of applications in the Internet of Things (IoT) space.

This chapter provides an in-depth examination of the ESP32 development board, detailing its features, specifications, and various communication protocols. We will explore the ESP-NOW protocol, a unique wireless communication method that enhances the functionality of the ESP32 in networked environments.

### 1.2 ESP32 Module

The ESP32 is a highly integrated system-on-chip (SoC) designed by Espressif Systems, a semiconductor company. It is specifically tailored for Internet of Things (IoT) and embedded applications. The ESP32 features a dual-core processor, built-in WiFi and Bluetooth connectivity, various input/output interfaces, and support for a wide range of peripherals. It offers a low-cost and power-efficient solution for developing a lot devices and other connected applications. The ensuing image illustrates the visual representation of the card. 1.1.

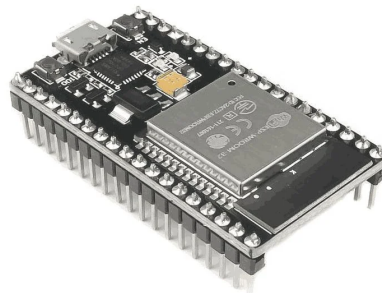


Figure 1.1: The ESP32 development board.

The functional diagram of the System-on-Chip (SoC) is shown in Figure 1.2.

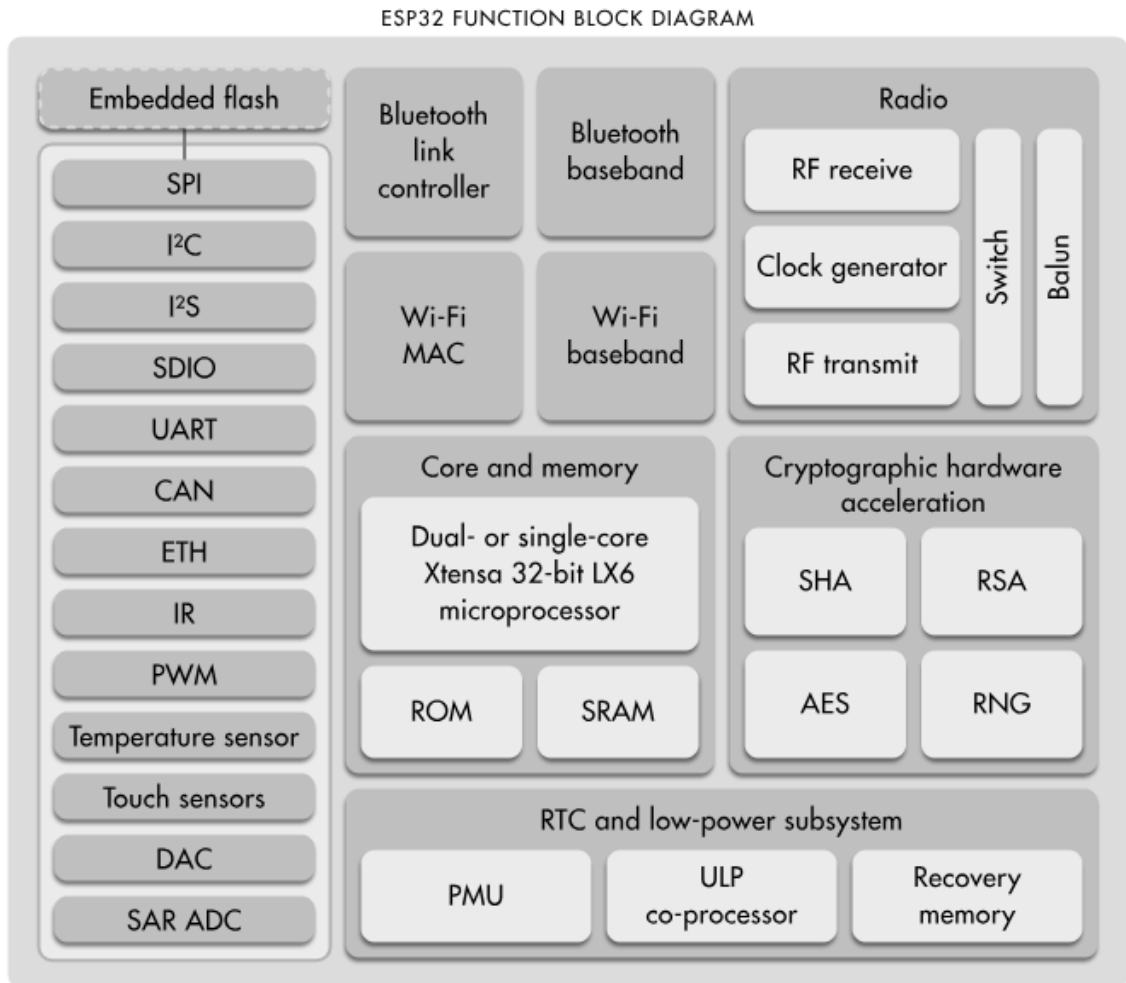


Figure 1.2: The ESP32 function Block diagram(SoC).

### 1.2.1 Features and specifications of ESP32

The ESP32 DevKit V1 is a popular development board for the ESP32 microcontroller. Here are the specifications for the ESP32 DevKit V1:

- **Processors:** Tensilica Xtensa 32-bit LX6 microprocessor
- **Cores:** Dual-core or single-core (depending on variant)
- **Architecture:** 32-bit
- **Performance:** capabilities extend up to 600 DMIPS
- **Clock Frequency:** 240 MHz
- **Wireless Connectivity:** Wireless Fidelity (Wi-Fi) 802.11 b/g/n (2.4 GHz) Bluetooth v4.2 BR/EDR and BLE (Bluetooth Low Energy)
- **Peripheral Interfaces:** The ESP32 DevKit V1 offers a variety of peripheral interfaces, providing extensive functionality for various applications. These interfaces include:
  - **General Purpose Input/Output (GPIO):** These pins allow the microcontroller to communicate with and control external devices such as sensors, actuators, LEDs, displays, and more.



- **Analog-to-Digital Converter (ADC):** Convert analog signals into digital data for processing, enabling measurement and monitoring of analog sensors or signals.
- **Digital-to-Analog Converter (DAC):** Convert digital data into analog signals, useful for generating analog output signals.
- **Inter-Integrated Circuit (I2C):** A communication protocol for connecting multiple peripheral devices with a two-wire interface, facilitating data exchange between them.
- **Universal Asynchronous Receiver/Transmitter (UART):** Serial communication interface for asynchronous data transfer between the ESP32 and other devices.
- **Controller Area Network (CAN):** A robust serial communication protocol commonly used in automotive and industrial applications for high-reliability data transmission.
- **Serial Peripheral Interface (SPI):** Synchronous serial communication interface for connecting peripheral devices with a master-slave architecture, facilitating high-speed data transfer.
- **Integrated Inter-IC Sound (I2S):** Interface for digital audio communication between devices, commonly used for connecting audio codecs, DACs, and ADCs.
- **Ethernet Media-Independent Interface (RMII):** Ethernet interface standard used for connecting the ESP32 to Ethernet Physical Layer (PHY) devices.
- **PWM:** Technique for controlling analog circuits digitally by varying the duty cycle of a pulse waveform, commonly used for controlling motors, LEDs, and other devices requiring variable output voltages or currents.

These peripheral interfaces, along with Direct Memory Access (DMA) support, provide versatile connectivity options for interfacing with sensors, actuators, displays, communication modules, and other external devices, enabling a wide range of IoT and embedded applications.

- **Memory:**

- **Internal ROM:** 448 KB for booting and core functions
- **Internal SRAM:** 520 KB for data and instructions
- **Embedded flash:** 4 MB
- **External Flash:** Up to 16 MB

### 1.2.2 Description of ESP32

The figure 1.3 illustrates the components of the ESP32 module.

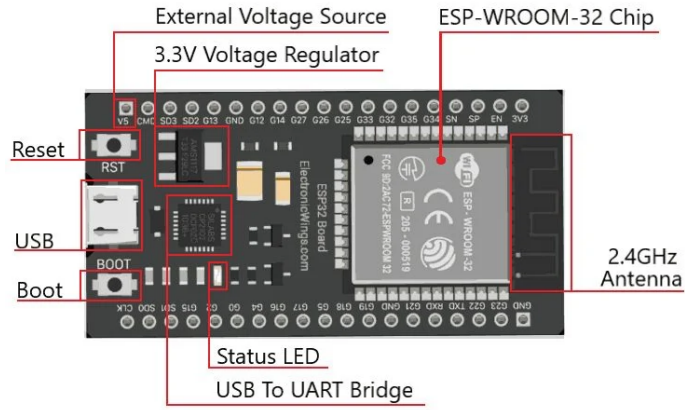


Figure 1.3: ESP32 Board Highlights.

The ESP32 development board has a total of 38 pinouts, the pin mapping that are shown in figure 1.4. It has support 8-bit 32 channels PWM. The pins with the symbol ' ~ ' represent that it has PWM support. It can be used for dimming LEDs or controlling motors.

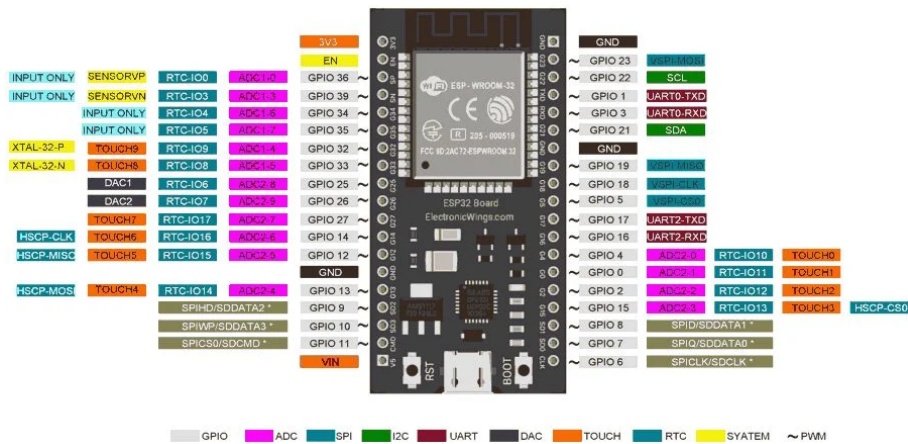


Figure 1.4: ESP32 Pinout Diagram.

## 1.3 ESPNOW Protocol

Espressif Simple Peripheral Network Over Wi-Fi (ESP-NOW) is a protocol developed by Espressif Systems specifically for low-power, peer-to-peer communication between and ESP32 devices. It enables these devices to communicate with each other directly, without the need for an access point or router.

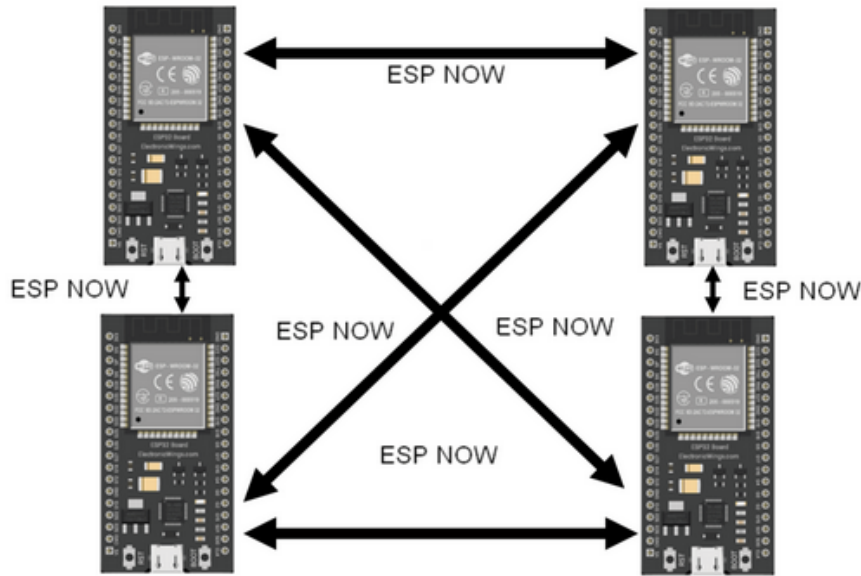


Figure 1.5: ESP-NOW Communication Architecture.

### 1.3.1 Features of ESP-NOW:

- **Low Power Consumption:** ESP-NOW is designed to operate efficiently on battery-powered devices, making it suitable for IoT applications where power consumption is a concern.
- **Peer-to-Peer Communication:** Devices communicate directly with each other without the need for an intermediary access point or router. This reduces latency and simplifies network topology.
- **High Throughput:** ESP-NOW provides high throughput communication, enabling devices to exchange data at relatively high speeds.
- **Simple API:** The ESP-NOW API is straightforward and easy to use, with functions for initializing communication, sending and receiving data, and error handling.
- **Security:** ESP-NOW supports data encryption to ensure the confidentiality and integrity of transmitted data. This helps protect against eavesdropping and tampering.

### 1.3.2 How ESP-NOW Works:

- **Initialization:** Devices initialize ESP-NOW by configuring communication parameters such as channel, encryption key, and callback functions for sending and receiving data.
- **Peer Discovery:** Devices need to discover each other's MAC addresses to establish communication. This can be done statically by pre-configuring MAC addresses or dynamically by scanning for nearby devices.
- **Data Transmission:** Once devices have discovered each other, they can start exchanging data using ESP-NOW. Data transmission can occur in both directions, allowing for bidirectional communication.

- **Error Handling:** ESP-NOW includes mechanisms for error detection and recovery to ensure the reliability of data transmission. This includes checksum verification and retransmission of lost packets.

### 1.3.3 Applications of ESP-NOW:

- **Sensor Networks:** ESP-NOW can be used to create low-power sensor networks where multiple sensor nodes communicate directly with a central controller or gateway.
- **Home Automation:** ESP-NOW enables communication between different IoT devices within a home automation system, allowing for seamless integration and control.
- **Mesh Networks:** ESP-NOW can be used to create mesh networks where devices relay data to each other, extending the network's coverage and improving reliability.
- **Industrial Monitoring:** ESP-NOW is suitable for industrial monitoring applications where low-power wireless communication is required to collect data from remote sensors and devices.

### 1.3.4 One-Way Communication:

In one-way communication mode, ESP-NOW allows a sender device to transmit data to one or more receiver devices without expecting a response. This mode is suitable for scenarios where only one-way data transmission is required, such as sensor data collection, broadcasting messages, or remote control applications. The figure 1.6 is Visual Representation of One-Way Communication.

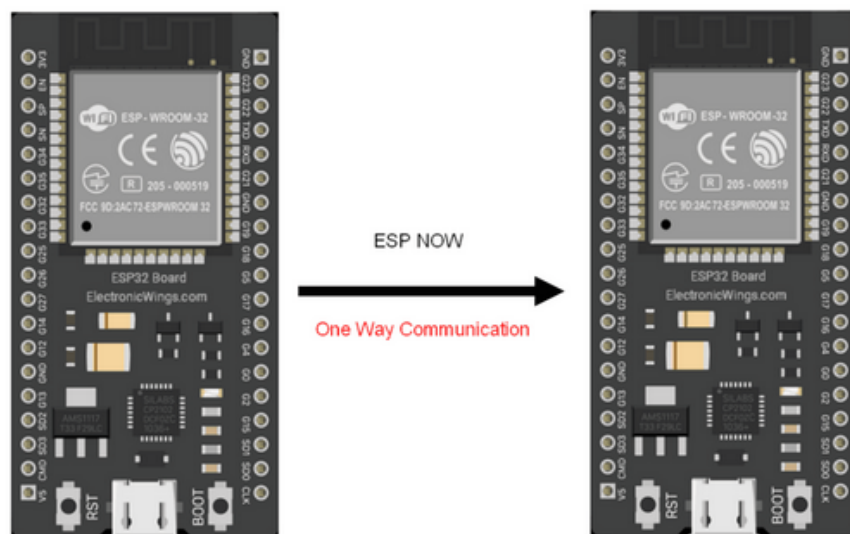


Figure 1.6: One-Way Communication.

Key Characteristics:

- **Unidirectional:** Data is transmitted from the sender to the receiver(s) without any response from the receiver(s).
- **Broadcasting:** The sender can broadcast data to multiple receivers simultaneously.
- **Low Latency:** One-way communication typically has lower latency compared to two-way communication since there's no need for acknowledgment or response from the receiver(s).

### 1.3.5 Two-Way Communication:

In two-way communication mode, ESP-NOW enables bidirectional communication between devices, allowing both sender and receiver devices to exchange data with each other. This mode is suitable for interactive applications where devices need to send and receive data in both directions, such as chat applications, remote control with feedback, or real-time monitoring systems.

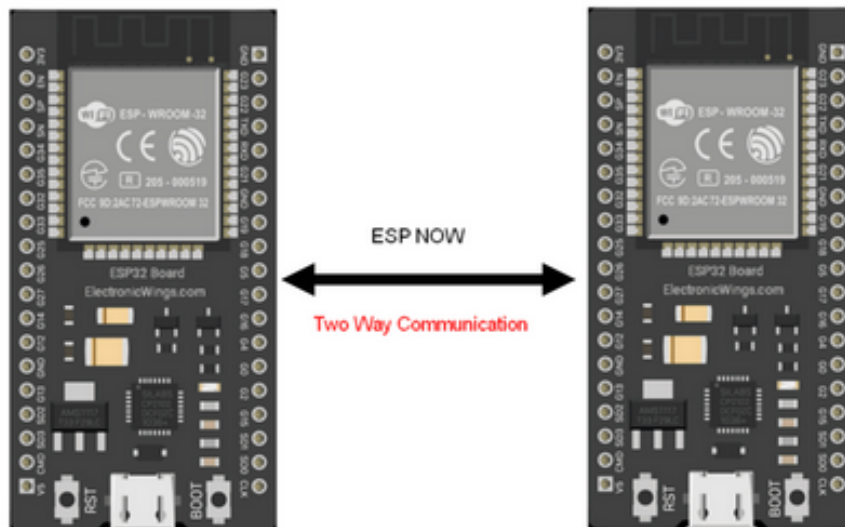


Figure 1.7: Two-Way Communication.

Key Characteristics:

- **Bidirectional** : Devices can send and receive data from each other, enabling interactive communication.
- **Acknowledgment**: After receiving data, the receiver device can send an acknowledgment (ACK) back to the sender to confirm successful reception.
- **Higher Complexity**: Two-way communication involves additional overhead for managing acknowledgments and handling potential communication errors, compared to one-way communication.

### 1.3.6 ESP-NOW One Board to Multiple Board Communication

In a one-to-multiple configuration, a single ESP32 device (the sender) communicates simultaneously with multiple other ESP32 devices (the receivers).

This setup enables the sender to broadcast data to several receivers efficiently, making it ideal for applications such as sensor networks, remote control systems, and real-time data sharing.

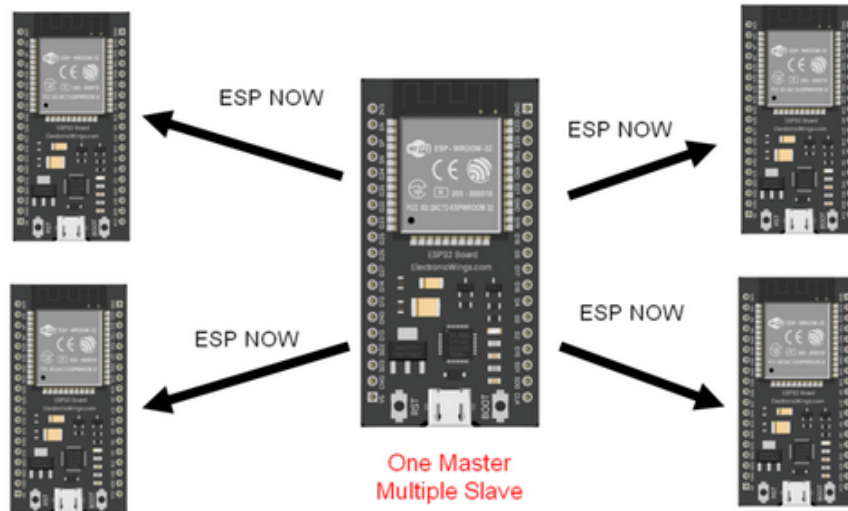


Figure 1.8: ESP-NOW One-to-Multiple Board Communication.

Key features include:

- **Low Latency:** Quick data transmission with minimal delay.
- **Low Power Consumption:** Efficient use of power, suitable for battery-operated devices.
- **No Need for Wi-Fi Network:** Devices communicate directly without the need for a Wi-Fi router or access point.
- **Scalability:** Easily add more receiver devices to the network.

### 1.3.7 ESP-NOW Multiple Board to One Board Communication

ESP-NOW Multiple Board to One Board Communication refers to a wireless communication protocol that allows multiple ESP32 devices (senders) to transmit data to a single ESP32 device (receiver).

This configuration is useful in scenarios where data from various sources needs to be collected and processed by a central device, such as in sensor networks, IoT data aggregation, and remote monitoring systems.



Figure 1.9: ESP-NOW Multiple-to-One Board Communication.

Key features include:

- **Low Latency:** Ensures quick data transmission with minimal delay, making it suitable for real-time applications.
- **Low Power Consumption:** Optimized for low power use, making it ideal for battery-powered devices.
- **No Wi-Fi Network Required:** Devices communicate directly with each other, without the need for a Wi-Fi router or access point.
- **Scalability:** Multiple devices can send data to a single receiver, allowing for scalable network architectures.

### 1.3.8 The MAC address

The MAC address, short for Media Access Control Address, serves as a distinct hardware identifier for every device within a network. It comprises six sets of two hexadecimal digits, segregated by colons. For instance, a MAC address could appear as 30:AE:A4:07:0D:64.

### 1.3.9 Wireless networking modes

- **STA Mode (Station Mode):**

In Station (STA) mode, the ESP32 connects to an existing wireless network as a client device, just like your laptop or smartphone connects to a Wi-Fi router.

The ESP32 acts as a station that joins an existing network, allowing it to communicate with other devices on that network and access resources such as the internet.

It requires the SSID (network name) and password of the existing Wi-Fi network to establish a connection.



Figure 1.10: STA Mode in ESP32 Wireless Networking.

- **AP Mode (Access Point Mode):**

In Access Point (AP) mode, the ESP32 acts as a Wi-Fi access point, essentially creating its own wireless network that other devices can connect to.

Other devices, such as smartphones or laptops, can connect to the ESP32's network just like they would connect to any other Wi-Fi hotspot.

This mode is useful when you want the ESP32 to serve as a hub or controller for other devices, allowing them to communicate with each other through the ESP32.

In AP mode, you need to specify the SSID and password for the network that the ESP32 creates.



Figure 1.11: AP Mode in ESP32 Wireless Networking.

In summary, STA mode enables the ESP32 to connect to an existing Wi-Fi network as a client, while AP mode turns the ESP32 into an access point, allowing other devices to connect to it. These modes provide flexibility for different networking scenarios in ESP32-based projects.



### 1.4 Conclusion

By the end of this chapter, readers will have gained a basic understanding of the ESP32 module, a versatile and powerful platform suitable for many IoT applications. A rich feature set, including the efficient ESP-NOW protocol, allows seamless wireless communication between devices.

Understanding the capabilities and applications of the ESP32 module is crucial to leveraging its full potential in developing innovative and effective IoT solutions.

# Chapter 2

## MPU-6050 Accelerometer and Gyroscope Module

### 2.1 Introduction

Integrating the ESP32 with the MPU-6050 accelerometer and gyroscope module allows for the creation of sophisticated motion-sensing applications. This chapter discusses the functionalities of the MPU-6050, including its 3-axis gyroscope, 3-axis accelerometer, and Digital Motion Processor Digital Motion Processor (DMP). We will cover how to interface the MPU-6050 with the ESP32 to display sensor data in real-time.

### 2.2 MPU-6050 Accelerometer and Gyroscope Module

The MPU6050 sensor module is a comprehensive 6-axis Motion Tracking Device that integrates a 3-axis Gyroscope, a 3-axis Accelerometer, and a Digital Motion Processor into a compact package. Additionally, it features an on-chip Temperature sensor for added functionality. The module communicates with microcontrollers via the I2C bus interface.

Moreover, the MPU6050 includes an Auxiliary I2C bus interface, enabling communication with other sensor devices such as a 3-axis Magnetometer or Pressure sensor. When a 3-axis Magnetometer is connected to the auxiliary I2C bus, the MPU6050 can deliver complete 9-axis Motion Fusion output. The Figure 2.1 showcases the MPU-6050 Module.

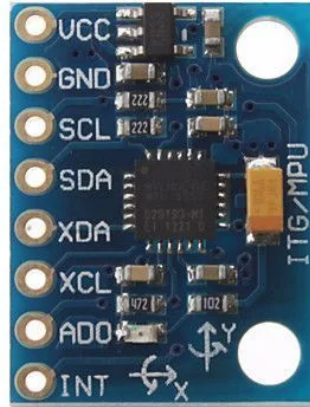


Figure 2.1: MPU6050 Module.

## 2.2.1 3-Axis Gyroscope

The MPU6050 features a 3-axis Gyroscope equipped with Micro Electro Mechanical Systems (MEMS) technology. This gyroscope is designed to detect rotational velocity along the X, Y, and Z axes, as depicted in the figure 2.2.

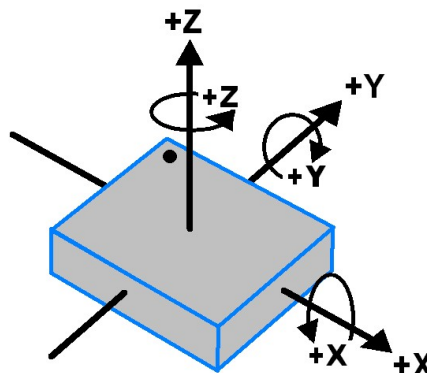


Figure 2.2: MPU6050 orientation and polarity of rotation.

- When the gyros rotate around any of the sense axes, the Coriolis Effect induces a vibration detected by a MEMS inside the MPU6050.
- This signal undergoes amplification, demodulation, and filtering to yield a voltage proportional to the angular rate.
- The voltage is then digitized using a 16-bit ADC to sample each axis.
- The output has a full-scale range of  $\pm 250$ ,  $\pm 500$ ,  $\pm 1000$ , or  $\pm 2000$ .
- The MPU6050 measures the angular velocity along each axis in degrees per second.

## 2.2.2 3-Axis Accelerometer

The MPU6050 features a 3-axis Accelerometer utilizing MEMS technology. It is employed to detect the angle of tilt or inclination along the X, Y, and Z axes, as illustrated in the figure 2.3.

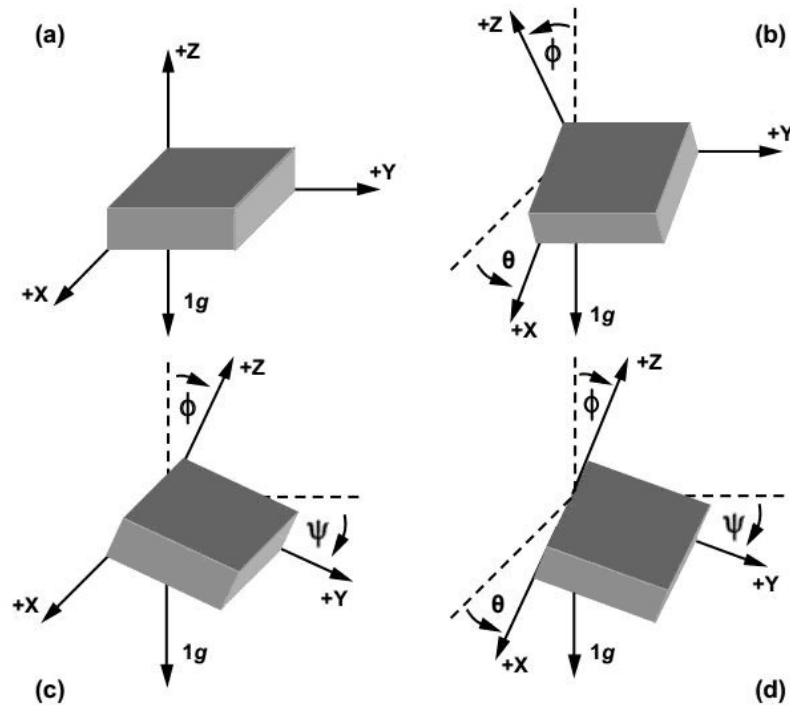


Figure 2.3: Angles of Spherical Coordinate System.

- Acceleration along the axes causes a deflection in the movable mass.
- This displacement of the moving plate (mass) disrupts the differential capacitor, resulting in sensor output. The amplitude of the output is proportional to the acceleration.
- A 16-bit ADC is employed to obtain digitized output.
- The full-scale range of acceleration is  $\pm 2g$ ,  $\pm 4g$ ,  $\pm 8g$ , or  $\pm 16g$ .
- Acceleration is measured in  $g$  (gravity force) units.
- When the device is placed on a flat surface, it registers  $0g$  on the  $X$  and  $Y$  axes and  $+1g$  on the  $Z$  axis.

### 2.2.3 DMP (Digital Motion Processor)

DMP is a specialized microprocessor designed to handle complex motion-related data processing tasks. It is often integrated into motion sensors like accelerometers and gyroscopes, allowing for efficient and accurate processing of motion data. Here are some key points about DMPs:

#### 1. Purpose and Functionality:

- **Sensor Data Fusion:** DMPs can integrate data from multiple sensors, such as accelerometers, gyroscopes, and magnetometers, to provide accurate motion tracking and orientation data.
- **Filtering and Calibration:** They apply advanced algorithms to filter noise, compensate for sensor biases, and calibrate the sensor data in real-time.
- **Complex Calculations:** DMPs can perform complex calculations required for applications like gesture recognition, activity tracking, and 3D motion capture.

## 2. Applications:

- **Consumer Electronics:** Used in smartphones, tablets, and gaming controllers to provide features like screen orientation, motion-based gaming, and augmented reality.
- **Wearable Devices:** In fitness trackers and smartwatches to monitor physical activities and health metrics.
- **Virtual Reality (VR) and Augmented Reality (AR):** Enhance user experiences by accurately tracking head and body movements.
- **Robotics:** Enable precise control and navigation of robots by providing accurate motion data.

## 3. Advantages:

- **Offloading Processing:** By handling motion data processing, DMPs offload these tasks from the main CPU, improving system performance and reducing power consumption.
- **Accuracy and Precision:** DMPs enhance the accuracy and precision of motion sensing applications through advanced algorithms and sensor fusion techniques.
- **Low Power Consumption:** Designed to operate efficiently, making them ideal for battery-powered devices.

## 2.2.4 On-chip Temperature Sensor

The output of the on-chip temperature sensor is digitized using an ADC. This process converts the analog temperature measurement into a digital format that can be processed by the microcontroller. The temperature reading from the sensor can then be accessed and read from the sensor data register, allowing the microcontroller to retrieve the temperature information and utilize it as needed within the application.

## 2.2.5 MPU6050 Module Pinout

The MPU6050 module pinout is shown in figure 2.4

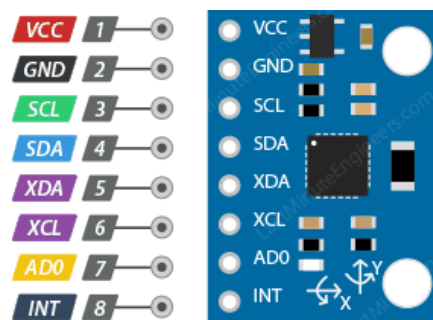


Figure 2.4: MPU6050 Module Pinout.

The MPU-6050 module has 8 pins:

- **VCC:** supplies power to the module.
- **GND:** is the ground pin.

- **SCL:** is a serial clock pin for the I2C interface.
- **SDA:** is a serial data pin for the I2C interface.
- **XDA:** is the external I2C data line. The external I2C bus is for connecting external sensors, such as a magnetometer.
- **XCL:** is the external I2C clock line.
- **AD0:** provides the capability to adjust the I2C address of the MPU6050 module. This feature can be utilized to prevent conflicts with other I2C devices or to link two MPU6050s to the same I2C bus. When the AD0 pin is left unconnected, the default I2C address is 0x68 (hexadecimal); however, connecting it to 3.3V alters the I2C address to 0x69 (hexadecimal).
- **INT:** is the Interrupt Output pin. The MPU6050 can be programmed to generate an interrupt upon detection of gestures, panning, zooming, scrolling, tap detection, and shake detection.

### 2.3 Interfacing ESP32 with MPU-6050

MPU-6050 is interfacing to the ESP32 card via I2C protocol as shown in schematic diagram of the Figure 2.5. The pin connection between ESP32 and mpu-6050 is given in Table 2.1. By default, GPIO 21 (SDA) and GPIO 22 (SCL) are used for I2C on the ESP32.

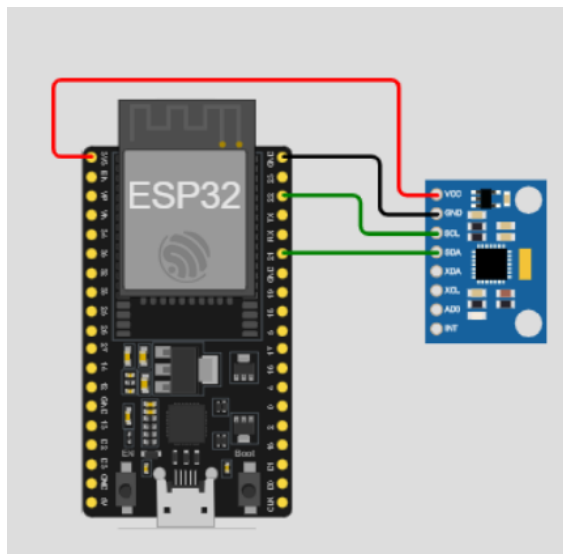


Figure 2.5: Schematic diagram ESP32 with MPU-6050.

MPU6050	ESP32
SCL	GPIO 22
SDA	GPIO 21
GND	GND
V <sub>CC</sub>	V <sub>CC</sub>

Table 2.1: Pin mapping between ESP32 and MPU6050.

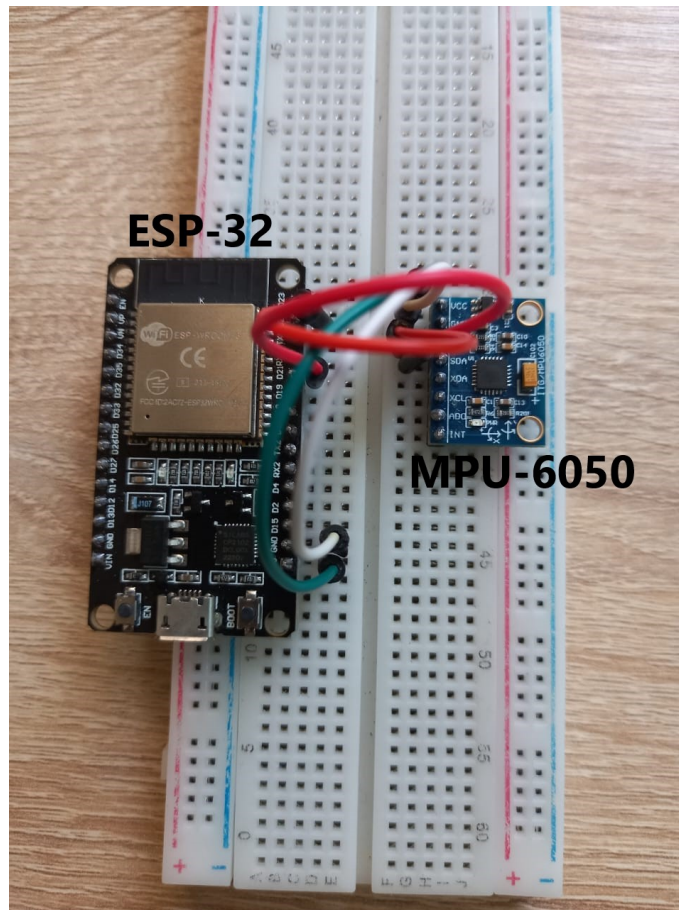


Figure 2.6: Circuit of ESP32 interfacing with MPU-6050.

## 2.4 Reading Accelerometer, Gyroscope, and Temperature Data with MPU6050 and ESP32

We'll employ the Adafruit MPU6050 libraries to fetch accelerometer, gyroscope, and temperature data from the MPU6050 module and exhibit it on the Arduino Serial Monitor.

To accomplish this, we'll install the Adafruit MPU6050 library via the Arduino Library Manager.

### 2.4.1 Installing Libraries

The following libraries are required for the mpu-6050 sensor :

- **Adafruit-tMPU6050.h:** This library provides functions to communicate with the MPU6050 sensor.
- **Adafruit-Sensor.h:** This library provides a unified sensor interface.
- **Wire.h:** This library used for I2C communication, enabling Arduino boards to interact with I2C-compatible devices such as sensors, displays, and Erasable Programmable Read-Only Memory (EPROM)s.

### 2.4.2 MPU-6050 readings sensors flowchart

In this section, we will explore the program we have developed for getting sensor readings from the MPU6050 sensor.

The flowchart in Figure 2.7 illustrates the process of reading data from the MPU6050 sensor.

The code includes essential libraries for interacting with the MPU6050 sensor and establishing communication via the I2C protocol.

Upon successful initialization, the program configures various parameters of the MPU6050 sensor, including the accelerometer range, gyroscope range, and filter bandwidth.

The program continuously retrieves acceleration (x,y,z), rotational velocity (x,y,z), and temperature data from the MPU6050 sensor and prints the sensor readings to the Serial Monitor, as depicted in Figure 2.8.



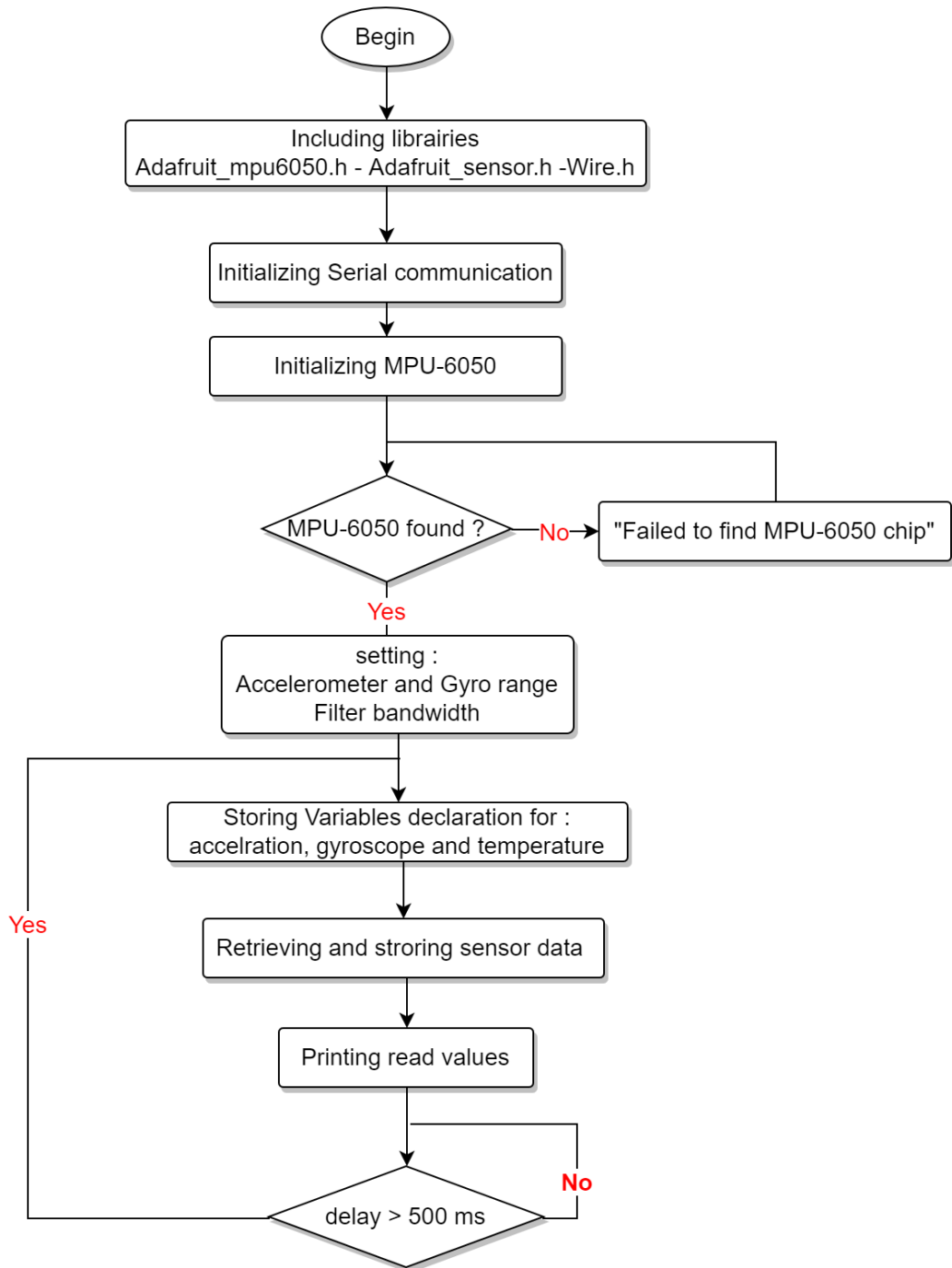
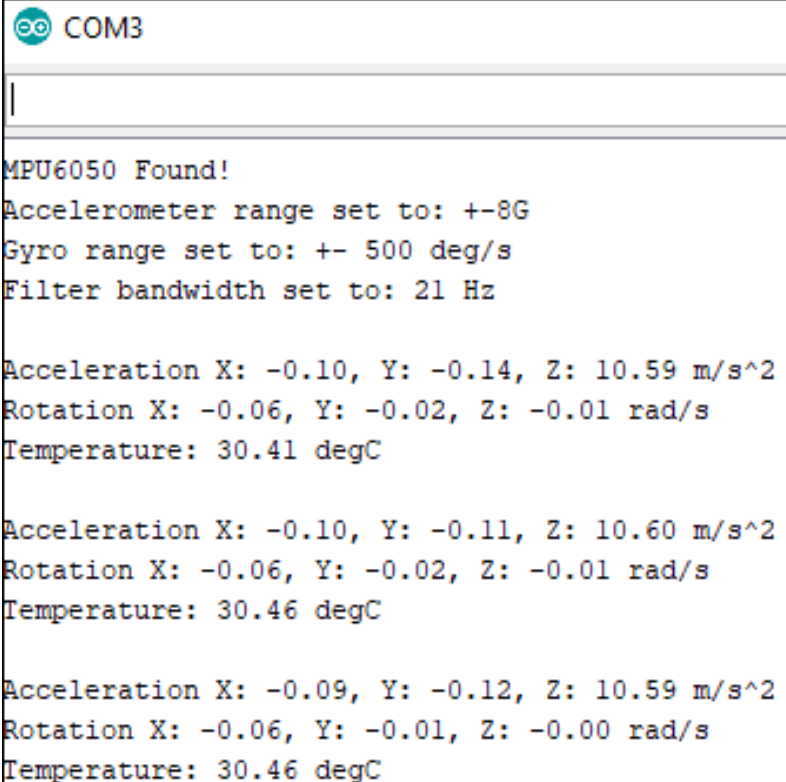


Figure 2.7: Flowchart for getting MPU-6050 readings sensors.

### 2.4.3 MPU6050 Output on Serial Monitor

A screenshot of a serial monitor window titled 'COM3'. The window displays the following text: 'MPU6050 Found!', 'Accelerometer range set to: +-8G', 'Gyro range set to: +- 500 deg/s', 'Filter bandwidth set to: 21 Hz', followed by three sets of sensor data: 'Acceleration X: -0.10, Y: -0.14, Z: 10.59 m/s^2', 'Rotation X: -0.06, Y: -0.02, Z: -0.01 rad/s', 'Temperature: 30.41 degC'; 'Acceleration X: -0.10, Y: -0.11, Z: 10.60 m/s^2', 'Rotation X: -0.06, Y: -0.02, Z: -0.01 rad/s', 'Temperature: 30.46 degC'; and 'Acceleration X: -0.09, Y: -0.12, Z: 10.59 m/s^2', 'Rotation X: -0.06, Y: -0.01, Z: -0.00 rad/s', 'Temperature: 30.46 degC'.

```
COM3
|
MPU6050 Found!
Accelerometer range set to: +-8G
Gyro range set to: +- 500 deg/s
Filter bandwidth set to: 21 Hz

Acceleration X: -0.10, Y: -0.14, Z: 10.59 m/s^2
Rotation X: -0.06, Y: -0.02, Z: -0.01 rad/s
Temperature: 30.41 degC

Acceleration X: -0.10, Y: -0.11, Z: 10.60 m/s^2
Rotation X: -0.06, Y: -0.02, Z: -0.01 rad/s
Temperature: 30.46 degC

Acceleration X: -0.09, Y: -0.12, Z: 10.59 m/s^2
Rotation X: -0.06, Y: -0.01, Z: -0.00 rad/s
Temperature: 30.46 degC
```

Figure 2.8: Serial Monitor Output of MPU 6050 Sensor Data.

## 2.5 Conclusion

In conclusion, interfacing the MPU6050 sensor with the ESP32 microcontroller allows for the seamless acquisition of accelerometer, gyroscope, and temperature data.

This integration provides a versatile platform for various applications, including motion tracking, orientation sensing, and environmental monitoring.

By leveraging the capabilities of both the MPU6050 sensor and the ESP32 microcontroller, developers can create robust and efficient solutions for a wide range of IoT and embedded projects.

# Chapter 3

## Wireless Hand Gesture Controlled Robot Car Using ESP32 with ESPNOW

### 3.1 Introduction

This chapter explores the creation of a wireless hand gesture-controlled robot car using the ESP32 microcontroller and ESPNOW protocol. It covers interfacing the L298N motor driver with the ESP32, detailing its specifications, pinout, and working principles.

The chapter discusses DC motors, their operation, and the use of Mecanum wheels for enhanced maneuverability. It explains controlling DC motors via the L298N driver module, and the direct control by ESP32. The ESPNOW protocol is examined, including obtaining MAC addresses and flowcharts for communication.

Finally, the implementation of the robot car and its code are presented.

### 3.2 Interfacing the L298N Module with ESP32

The combination of the ESP32 microcontroller and the L298N motor driver module enables the development of a wireless car robot. This section introduces the L298N motor driver, its specifications, and its working principles. We will also explore the basics of DC motors and Mecanum wheels, essential components for building a responsive robot car.

### 3.3 Introducing the L298N "Motor Driver"

The L298N serves as a widely favored dual H-bridge motor driver Integrated Circuit (IC), facilitating the control of both speed and direction for DC motors and stepper motors. With a maximum current handling capacity of 2A per channel, it accommodates a broad spectrum of motor types and power supplies, operating within a voltage range spanning from 5V to 35V. Equipped with a sizable heatsink to manage the heat generated by motors, it necessitates an external capacitor to ensure power supply stability. Additionally, it boasts built-in protection mechanisms including

# Wireless Hand Gesture Controlled Robot Car Using ESP32 with ESPNOW

---

thermal shutdown and over-current protection, safeguarding both the IC and connected motors from potential damage. Visual image of the module in the Figure 3.1.

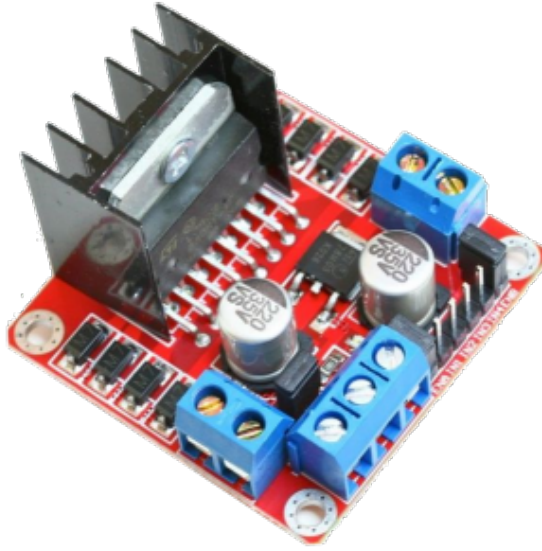


Figure 3.1: L298N motor driver Module.

## 3.3.1 Specifications

The Table 3.1 shows some specifications of the L298N motor driver module:

Driver Model	L298N
Driver Chip	Double H-bridge L298N
Maximum Power	25W
Maximum Motor Supply Voltage	46V
Maximum Motor Supply Current	2A
Driver Voltage	5-35V
Driver Current	2A
Size	43x43x26mm

Table 3.1: L298N Module Specifications.

## 3.3.2 Pinout

look at the pinout of the module as shown in figure 3.2:

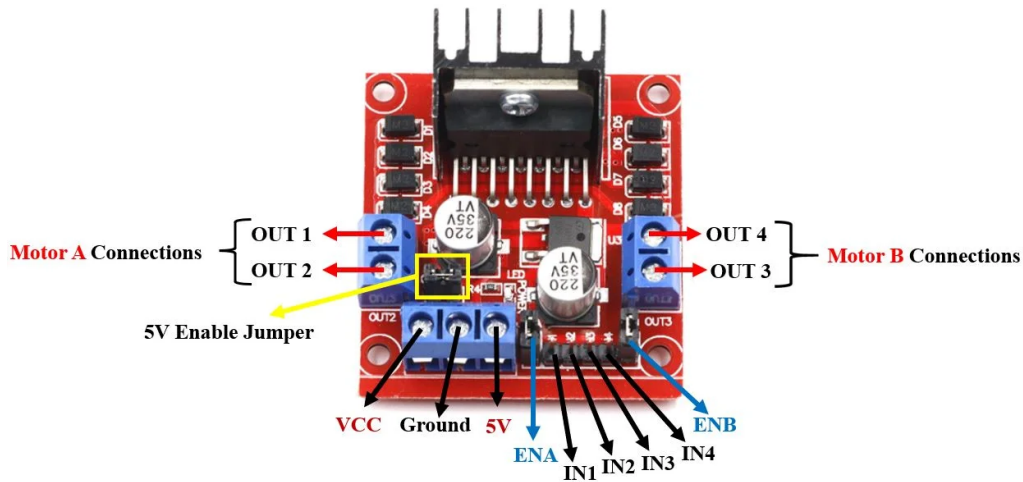


Figure 3.2: L298N Module Pinout.

- **VCC:** This pin serves as the power source for the motor. While it is labeled with +12V on the board, it can effectively power the motor within a range of 6 to 12 volts.
- **Ground:** This pin serves as the common ground connection.
- **5V:** This pin provides power (5V) for the internal circuitry of the L298N IC. It is utilized only if the 5V enable jumper is not connected. However, if the jumper is intact, this pin functions as an output pin.
- **ENA:** This pin controls the speed of the motor A by enabling the PWM signal.
- **IN1 , IN2** These are the input pins for motor A. They control the spinning direction for that particular motor.
- **IN3 , IN4** These are the input pins for motor B. They control the spinning direction for that particular motor.
- **ENB:** This pin controls the speed of the motor B by enabling the PWM signal.
- **OUT1 , OUT2:** OUT1 Positive terminal , OUT2 Negative terminal. These are the output pins for motor A. Motor A having voltage between 5-35V, will be connected through these two terminals.
- **OUT3 , OUT4:** OUT3 Positive terminal , OUT4 Negative terminal. These are the output pins for motor B.

### 3.3.3 L298N working principle

- **H-Bridges:** The H bridge is an electronic circuit that allows the direction and speed of a DC motor to be controlled. It consists of four switches (transistors or MOSFETs) arranged in the shape of the letter "H". By controlling the switching states of these switches, the H-bridge can apply voltage to the motor terminals in both forward and reverse directions, as well as control the motor speed. As shown in Figure 3.3

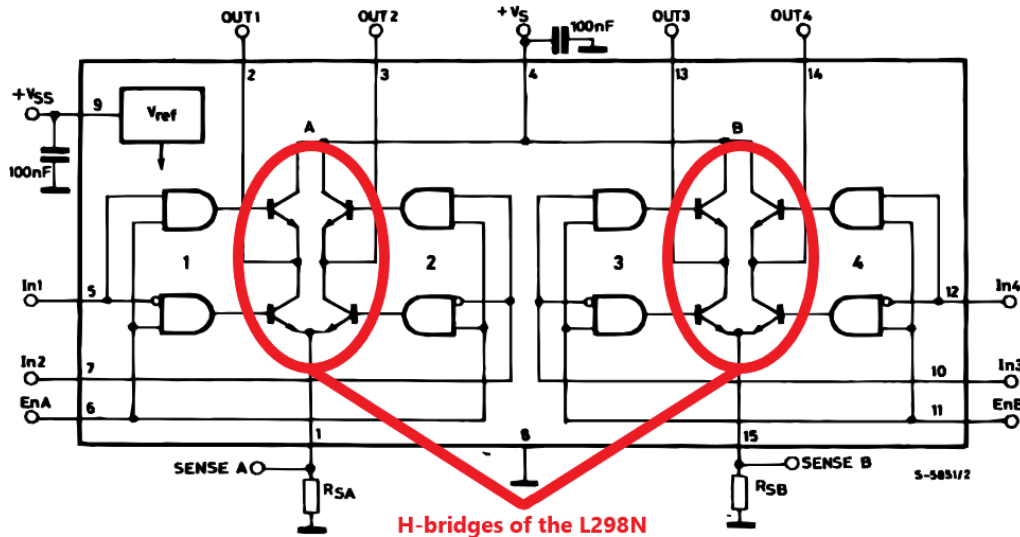


Figure 3.3: Internal diagram of the L298N.

Here's how it works:

- When the switches on one side of the H-bridge are closed and the switches on the other side are open, current from the power supply flows through the motor in one direction, causing it to rotate in one direction (for example, forward).
- When the switches on the opposite side are closed and the original switches are open, current flows through the motor in the opposite direction, causing it to rotate in the opposite direction (Backward).

As shown in Figure 3.4

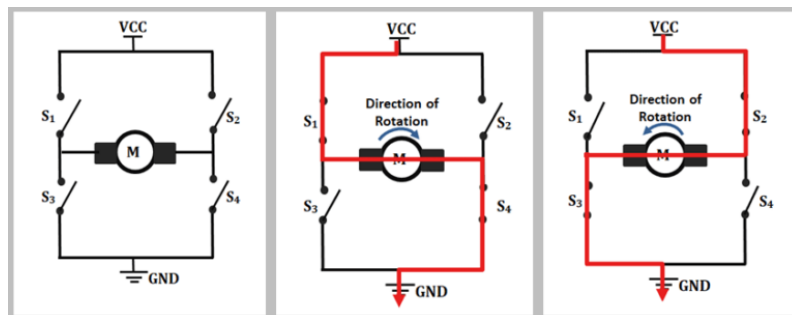


Figure 3.4: H-bridge switch working.

• **PWM Switch:**

PWM stands for Pulse Width Modulation. It's a technique used in electronics to control the power delivered to a load by rapidly switching a digital signal on and off. The ratio of the time the signal is on (high) to the time it is off (low), known as the duty cycle, determines the average power delivered to the load.

PWM is widely used in various applications, including:

- **Motor Speed Control:** PWM is commonly used to control the speed of DC motors. By adjusting the duty cycle of the PWM signal, you can regulate the average voltage applied to the motor, thus controlling its speed.

## Wireless Hand Gesture Controlled Robot Car Using ESP32 with ESPNOW

---

- LED Dimming: PWM is used to control the brightness of LEDs (Light Emitting Diodes). By varying the duty cycle of the PWM signal, you can adjust the average current flowing through the LED, thereby controlling its brightness.
- Power Regulation: PWM is employed in voltage regulation circuits to efficiently control the output voltage. By adjusting the duty cycle of the PWM signal, you can regulate the output voltage to match the desired level.
- Analog Signal Generation: PWM signals can be used to generate analog-like signals with varying amplitudes. By modulating the duty cycle of the PWM signal, you can generate signals with different average voltages.

Overall, PWM provides a flexible and efficient way to control power delivery in various electronic systems, making it a fundamental technique in modern electronics design.

The L298N's operation relies on a signal with a variable duty cycle to adjust the motors' speed. Essentially, a DC motor's speed correlates with its average voltage, which can be manipulated by altering the power supply's average voltage. Pulse Width Modulation (PWM) achieves this voltage variation. In PWM, a signal with a fixed frequency and variable duty cycle is generated. Mathematically, the average voltage equals the maximum voltage multiplied by the duty cycle. Figure 3.5 visually illustrates this PWM control principle.

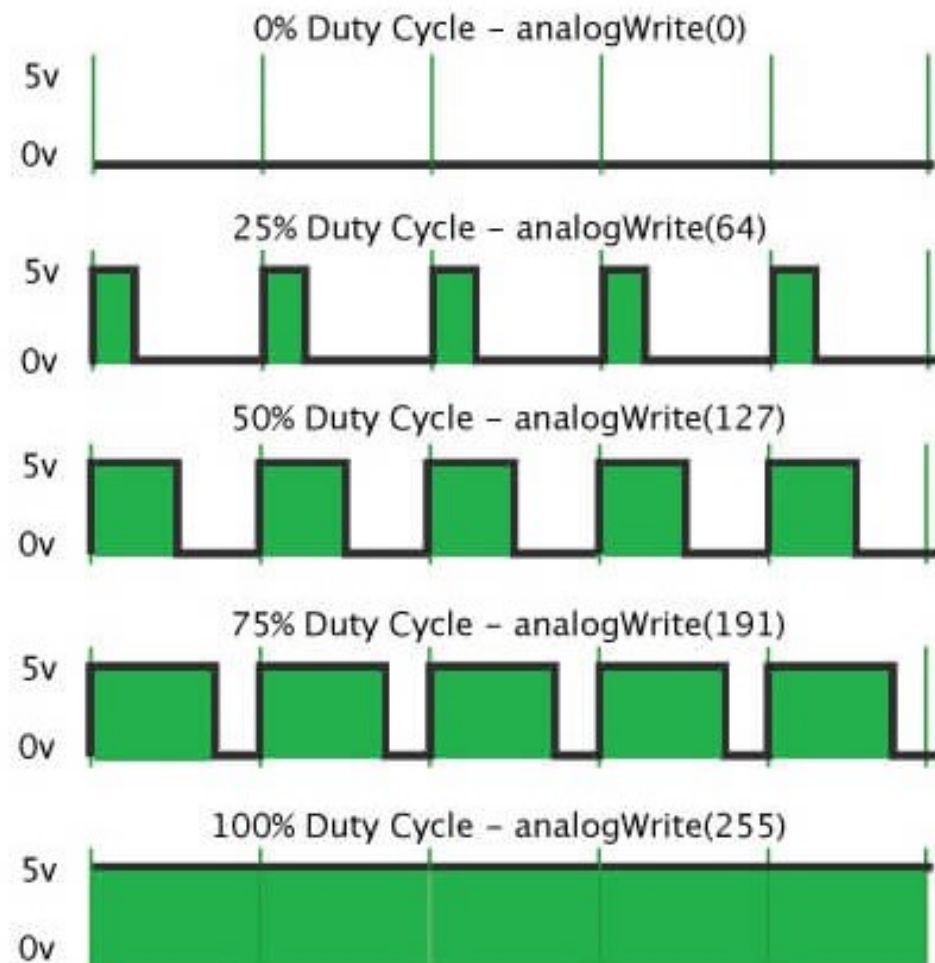


Figure 3.5: PWM Switch.

## 3.4 DC motors

Separately excited DC motors are still sometimes used for variable speed drive of machines. Very easy to miniaturize, they are essential in very low powers and low voltages. They also lend themselves very well, up to significant powers (several megawatts), to speed variation with simple and inexpensive electronic technologies for high performance. Their characteristics also allow precise regulation of torque, in motor or generator. Their nominal rotation speed, independent of the network frequency, is easily adaptable by construction to all applications.

### 3.4.1 Constitution

A direct current motor is made up of the following elements:

- **The inductor or stator:** It is an element of the stationary magnetic circuit on which a winding is wound in order to produce a magnetic field. The electromagnet thus produced has a cylindrical cavity between its poles.
- **The armature or rotor:** It is a cylinder made of magnetic sheets insulated from each other and perpendicular to the axis of the cylinder. The armature is movable in rotation around its axis and is separated from the inductor by an air gap. On its periphery, drivers are regularly distributed.
- **The collector and the brushes:** The collector is integral with the armature. The brushes are fixed, they rub on the collector and thus supply the armature conductors.



Figure 3.6: Visual image of a DC motor.

### 3.4.2 Principle of operation

- An inductor (stator) creates a magnetic field of fixed direction. This field can be obtained by a permanent magnet or by an electromagnet.
- The armature (rotor) carries conductors carrying a direct current. The turns are the seat of forces which create a torque causing the rotation of the rotor. This rotation results in a variation in the flow passing through the turn, a variation which generates an electromotive force (e.m.f.) at the terminals of the turn.
- A device allows the conductors to be supplied by reversing their direction twice per turn. This device is called a collector. More recent solutions allow an electronic solution to dispense with the collector. The motors are then called DC motors.



## 3.5 The Wheels

When designing a robot car, selecting a steering method is crucial. There are several popular options:

- **Skid Steering:** This method, widely favored for robot cars, employs two wheels. By adjusting the speed and direction of each wheel independently, the robot car can maneuver in any direction.
- **Standard Steering:** Similar to the steering used in automobiles, often referred to as rack and pinion steering in cars and trucks. In this setup, the front (or all four) wheels are coordinated to change their direction of travel. In smaller models and robot cars, the front wheels can be manipulated using a servo or stepper motor.

Now, we'll explore a third alternative "Mecanum Wheels" as we construct a robot car.

### 3.5.1 Mecanum Wheels

Mecanum wheels were pioneered in 1972 by Swedish engineer Bengt Erland Ilon while working at the design firm Mecanum AB.

These wheels represent a breakthrough in omnidirectional motion, capable of propelling a vehicle in any direction. Unlike conventional tires, they feature a set of rubberized external rollers strategically positioned at 45-degree angles around the wheel's circumference.

For optimal omnidirectional movement, Mecanum wheels are typically utilized in groups of four or more. However, it's possible to achieve some degree of control using just three wheels. Figure 3.7 illustrates the design and configuration of mecanum wheels.



Figure 3.7: Mecanum Wheel Design.

The Mecanum wheels possess pulleys that can be oriented in two distinct directions, necessitating the presence of two of each type for comprehensive control.

One category of wheels features rollers positioned at a 45-degree angle to the axle, while the other category employs rollers mounted at a 45-degree angle to the wheel itself.

These four wheels, comprising two of each type, are affixed to the robotic car's chassis as depicted in the figure 3.8.

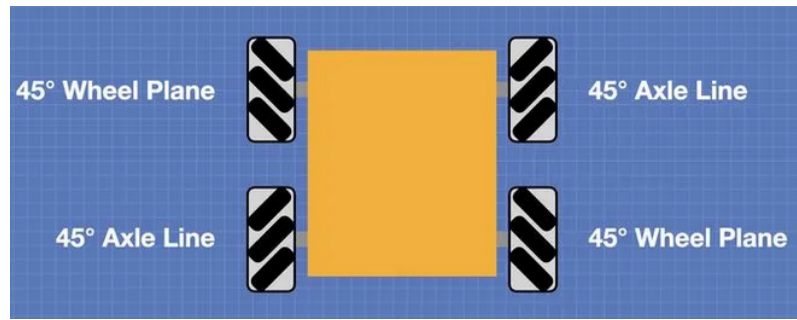


Figure 3.8: Installing mecanum wheels position.

By installing the wheels in this configuration, we enable the vehicle to move in six distinct directions.

- **Move Straight:** In this configuration, our Mecanum Wheel vehicle functions similarly to a standard robot car. We can move forwards or backward by adjusting the direction of rotation of the wheels accordingly. As shown in the figure 3.9.

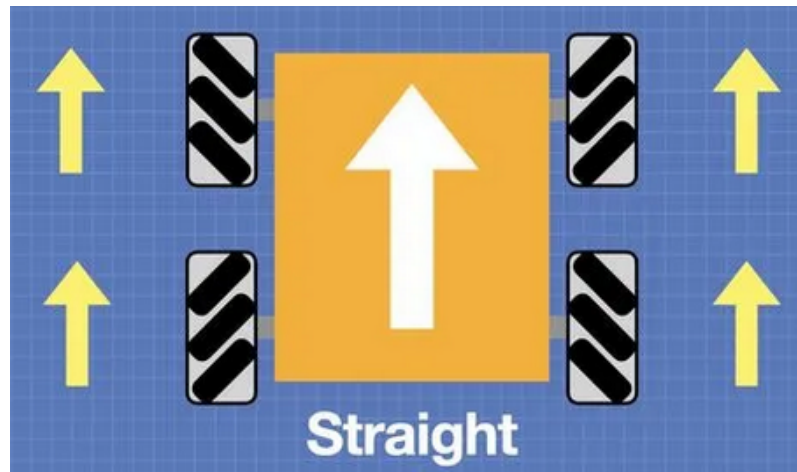


Figure 3.9: Straight direction.

In this mode, all four wheels are propelled in the same direction and at equal speeds.

- **Move Sideways:** By rotating the wheels on each side in opposite directions and reversing the sequence between sides, we can induce sideways movement in our vehicle. As shown in the figure 3.10.



Figure 3.10: Sideway direction.

Reversing the directions of all four motors will result in the vehicle moving sideways in the opposite direction.

- **Move Diagonally:** To achieve diagonal movement, we utilize only two motors located on opposite corners, ensuring they possess the same orientation of wheel rollers, as shown in the figure 3.11



Figure 3.11: Diagonal direction.

Changing the direction of rotation will indeed cause the vehicle to move at the same angle but in the opposite direction. Similarly, utilizing the other two Mecanum wheels will result in diagonal movement, albeit 90 degrees opposite to the original configuration.

- **Pivot:** Activating two wheels on the same side of the vehicle to rotate in the same direction will induce a pivoting motion. As shown in the figure 3.12.

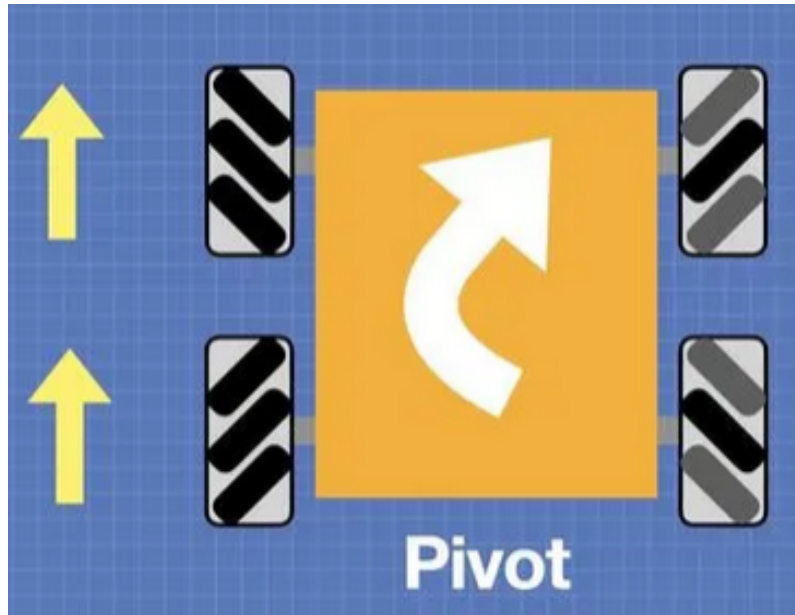


Figure 3.12: Pivot direction.

- **Rotate:** By rotating the wheels on one side of the chassis in one direction and those on the other side in the opposite direction, we can induce the vehicle to rotate. As shown in the figure 3.13.

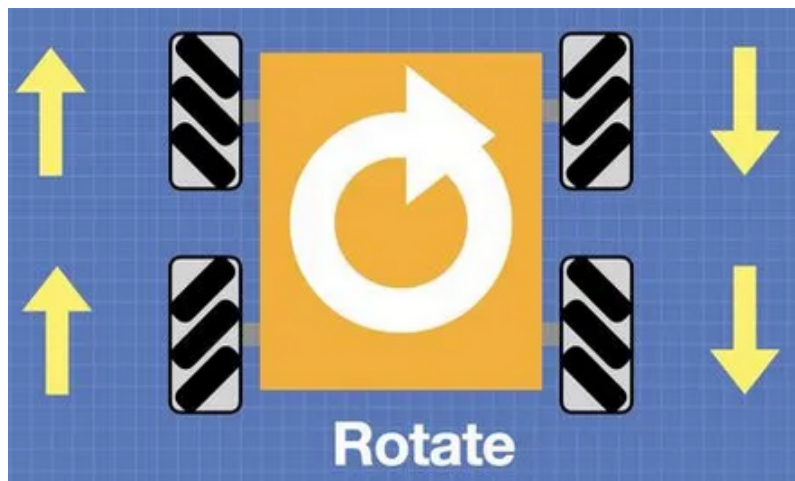


Figure 3.13: Rotate direction.

This operation mirrors that of a standard 4-wheel robot car. Reversing the direction of wheel rotation will result in the car spinning in the opposite direction.

- **Pivot Sideways:** The final Mecanum wheel movement is a variation of the Pivot, known as the Pivot Sideways. It operates similarly to the Pivot, but instead of using two wheels on the same side, it utilizes two wheels on the same axle.



Figure 3.14: Pivot Sideways direction.

Once more, reversing the direction of wheel rotation will reverse the direction of the pivot.

## 3.6 Controlling DC motors through L298N Driver Module

### 3.6.1 Control Pins

At the bottom right side of the module, you'll find two types of control pins. One type is responsible for regulating the speed of the motor, while the other type governs its direction.

- **Speed Control "ENABLE" Pins** : The speed control pins, labeled ENA and ENB on the module, are responsible for regulating the speed of the DC motor and toggling its power state, turning it ON and OFF. Take a look at the figure 3.15

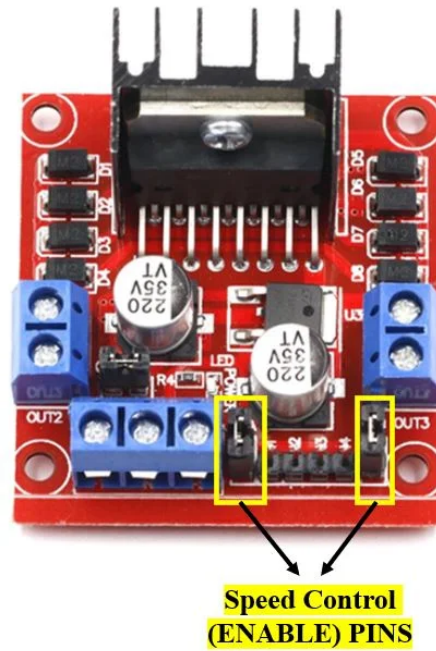


Figure 3.15: Speed Control Pins.

ENA controls the speed of motor A, while ENB controls the speed of motor B. When both pins are set to a logic HIGH (5V) state, both motors are ON and spinning at maximum speed. Conversely, when both pins are set to a logic LOW (ground) state, both motors are OFF. Through the PWM functionality, we can also adjust the speed of the motor.

By default, there is a jumper connected to these pins, maintaining them in a HIGH state. To regulate the speed, we must remove the jumper and connect these terminals to the PWM pins of the ESP32, then program them accordingly in the code. The table 3.2 illustrating the logic signals required for controlling Motor A.

ENA / ENB	Motor Action
1	ON
0	OFF
PWM	ON, speed proportional to the duty cycle

Table 3.2: Truth Table of The "Motor " Speed

- **Direction Control "INPUT" Pins:** The direction control pins consist of the four input pins labeled IN1, IN2, IN3, and IN4 on the module. These pins determine the direction of rotation for the connected motors. look at the figure 3.16

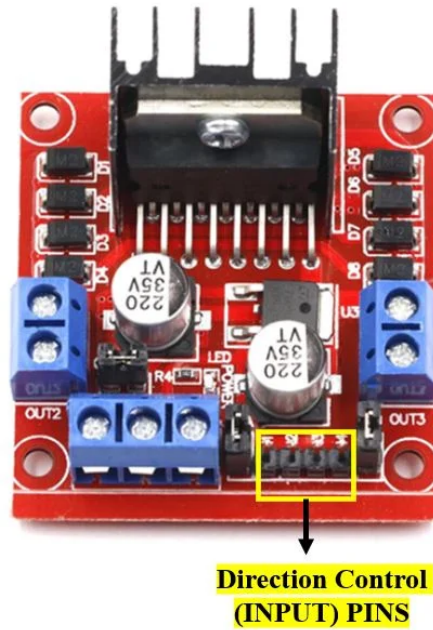


Figure 3.16: Direction Control Pins.

These input pins allow us to specify the direction of movement for the DC motors. IN1 and IN2 dictate the spinning direction of motor A, while IN3 and IN4 control motor B spinning direction. The Table 3.3 illustrating the required logic signals for the desired spinning action of motor A.

IN1	IN2	Motor A Action
IN3	IN4	Motor B Action
1	1	OFF
1	0	Forward
0	1	Backward
0	0	OFF

Table 3.3: Truth Table of the "Motor" Direction.

### 3.7 DC Motor control by ESP32 via L298N

In this section, we focus on control DC motors by ESP32 board via Motor driver L298N. The L298N module is interfaced to the ESP32 card as shown in schematic diagram of the Figure 3.17. The pin mapping between ESP32 and L298N Motor Driver is given in Table 3.4. Each motor is linked to the motor outputs. The Motor driver LN298N is powered by two AA 3.7V batteries. All components were assembled as depicted in the Figure 3.18 according to the schematic diagram in Figure 3.17.

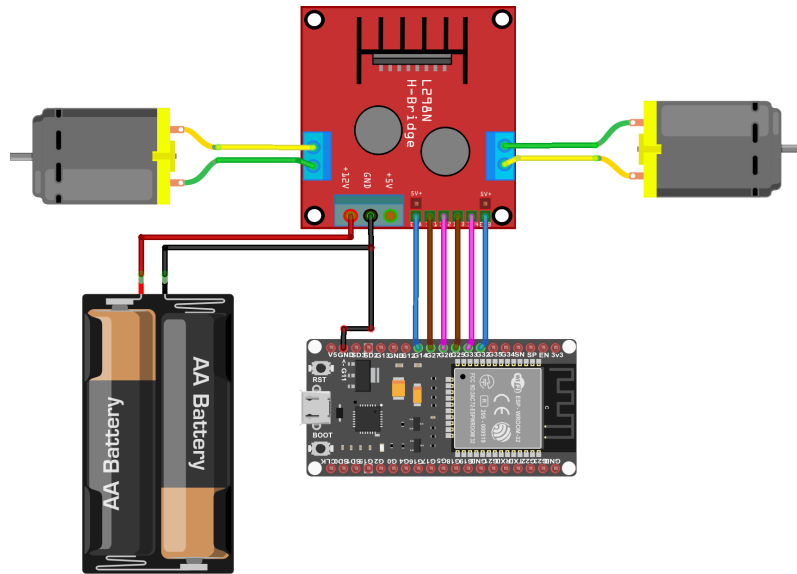


Figure 3.17: Schematic diagram of DC motor control by ESP32 via L298N.

L298N	ESP32
IN1	GPIO 27
IN2	GPIO 26
ENA	GPIO 14
IN3	GPIO 25
IN4	GPIO 33
ENB	GPIO 32
GND	GND

Table 3.4: The pin mapping between ESP32 and L298N.



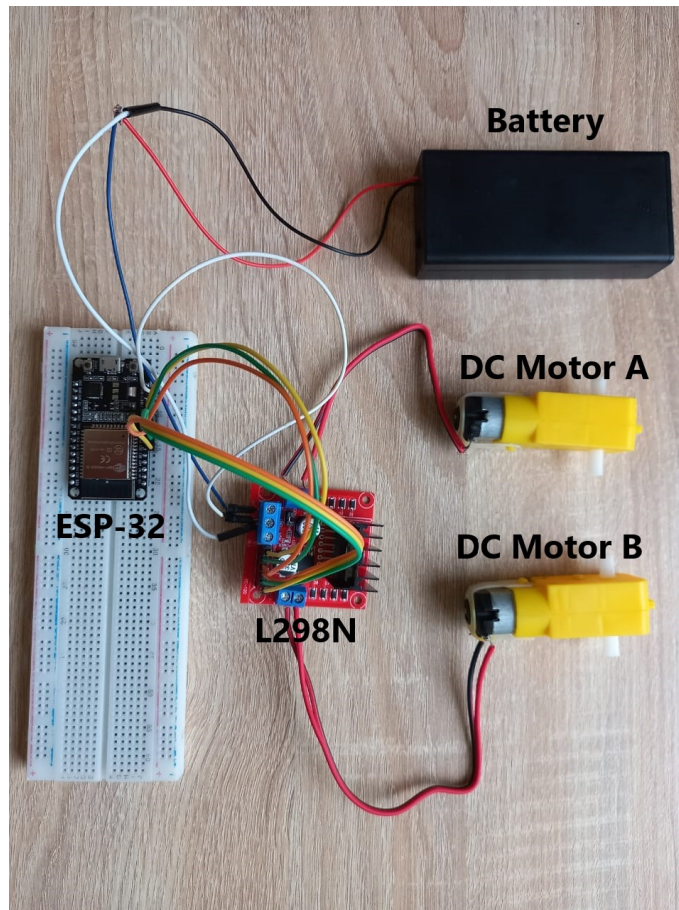


Figure 3.18: Circuit of DC motor control by ESP32 via L298N.

Now, let's move on to the code illustrated by flowchart of Figure 3.19. This code allows for basic testing and validation of DC motor control by ESP32 board.

Firstly, we create a structure containing the pins associated to IN1, IN2 and EN, as well as PWM Speed Channel used to control a motor.

Next, we specify the PWM settings, including frequency, resolution, and PWM value, which are essential for initializing the PWM channel to regulate the motor speed.

Furthermore, we implement a function to manage motor movement, allowing for forward, stop, and backward directions.

After configuring the motors pins as outputs and initializing the PWM channels, the channels are attached to the motor driver pins.

Initially, all motors are setting to stop by passing the "STOP" argument to the function.

Next, the function is called to control the motors in different directions based on specific parameters : "FOWARD", "STOP" or "BACKWARD". This ensures that each motor rotates in desired direction under a specific command.

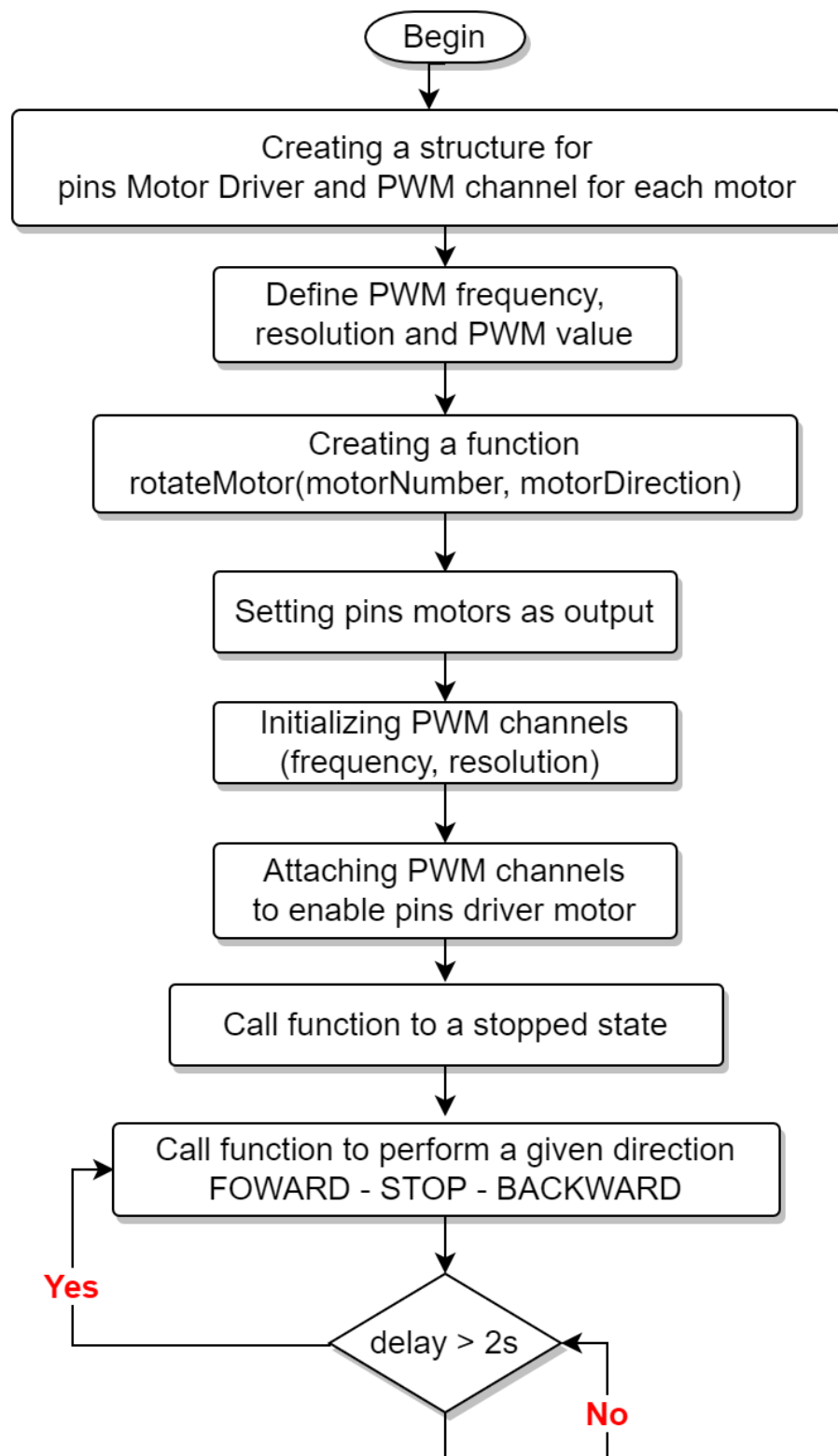


Figure 3.19: Implementation of Control DC Motors via L298N using ESP32.

## 3.8 ESPNOW Protocol

This section introduces the ESPNOW protocol, focusing on its application in facilitating communication between an ESP32 sender and receiver.

We begin by explaining the process of retrieving the MAC address from the ESP32 receiver, which is a critical step for establishing a seamless connection. A detailed flowchart illustrates this procedure.

Next, we explain in detail the flowcharts that depict the ESP32 data transmission and reception process. Every step, from library import to status monitoring, is comprehensively explained, and supplemented with relevant visuals and sequential screen outputs, to provide a comprehensive overview of the ESPNOW communications implementation.

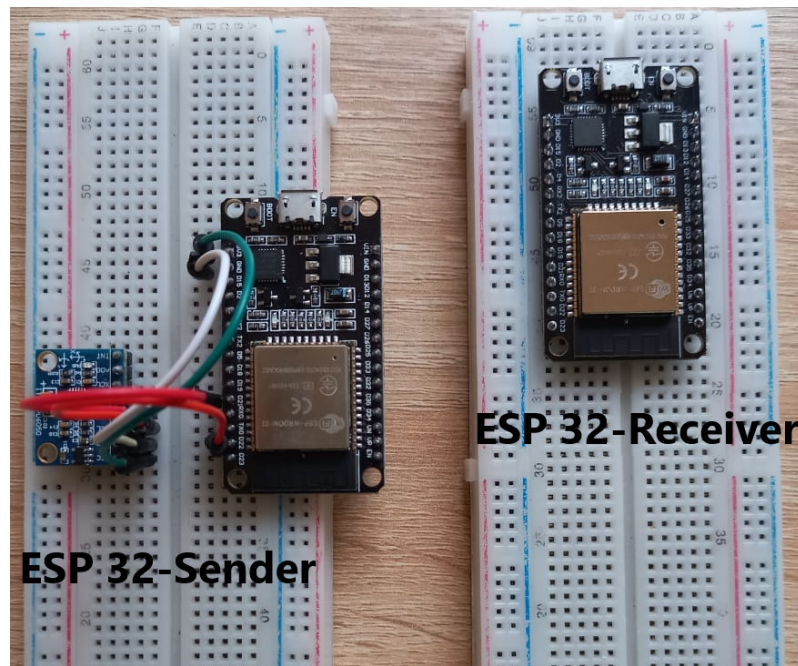


Figure 3.20: Circuit of sender receiver ESPNOW communication.

### 3.8.1 Getting MAC Address from ESP32 receiver

When we use the ESPNOW protocol for communication between a sender and a receiver, it's crucial to ascertain the receiver's MAC address. This ensures seamless communication between the devices.

Let's now move on to explaining the flowchart for Getting a MAC Address in Figure 3.21:

The code starts by importing the essential libraries required to obtain a MAC Address, Wi-Fi. After the successful initialization of Wi-Fi, we set the ESP32 to Station Mode.

Then, Retrieve the unique MAC address of the ESP32 once it is connected to the WiFi network. finally, Print the retrieved MAC address to the serial monitor as shown in the Figure 3.22.

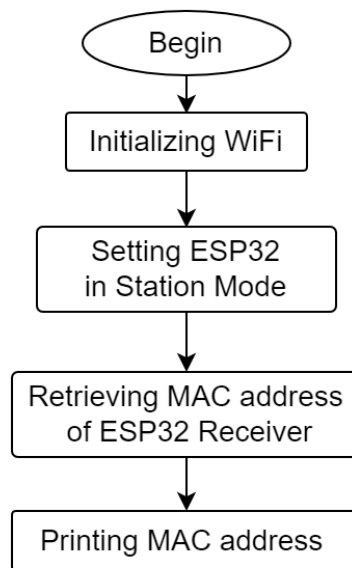


Figure 3.21: Flowchart for getting MAC Address of ESP32 receiver.

```
COM3  
ets Jun  8 2016 00:22:57  
  
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)  
configsip: 0, SPIWP:0xee  
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00  
mode:DIO, clock div:1  
load:0x3fff0030,len:1184  
load:0x40078000,len:13260  
load:0x40080400,len:3028  
entry 0x400805e4  
0C:B8:15:C3:09:20
```

Figure 3.22: The MAC address of Receiver

### 3.8.2 ESP32 sender flowchart

Let's now move on to explaining the flowchart for data transmission in Figure 3.23 :

The code begins by including necessary libraries for the MPU6050 sensor, I2C and ESP-NOW communication. The Adafruit's mpu6050.h is developed and maintained by Adafruit. It provides basic functions to access raw accelerometer and gyroscope data from the MPU6050. It is designed to be simple to use and integrated with the Adafruit library ecosystem.

Next, it sets up the environment to use an MPU6050 sensor with an ESP32 board by defining the data structure for sensor's readings, MAC Addresses, and an object to interact with MPU-6050.

Following that, it defines a callback function **OnDataSent** which is called after an ESPNOW

## Wireless Hand Gesture Controlled Robot Car Using ESP32 with ESPNOW

---

data packet is sent. This function is a useful tool for monitoring the success or failure of ESPNOW data transmissions.

It then sets the device to WiFi station mode, initializes ESPNOW and MPU-6050. If the initialization of ESPNOW fails, the code prints an error message and stops execution.

After the successful initialization of MPU-6050, the code proceeds to configure the accelerometer and gyroscope settings of the MPU6050 sensor such as range and bandwidth.

Subsequently, the code registers the **OnDataSent** callback function to monitor the status of sent packets.

Finally, the code continuously reads sensor data, displays it on the serial monitor as shown in Figure 3.24, stores it in a structure, sends it to the peer via ESPNOW, and checks the transmission status.

# Wireless Hand Gesture Controlled Robot Car Using ESP32 with ESPNOW

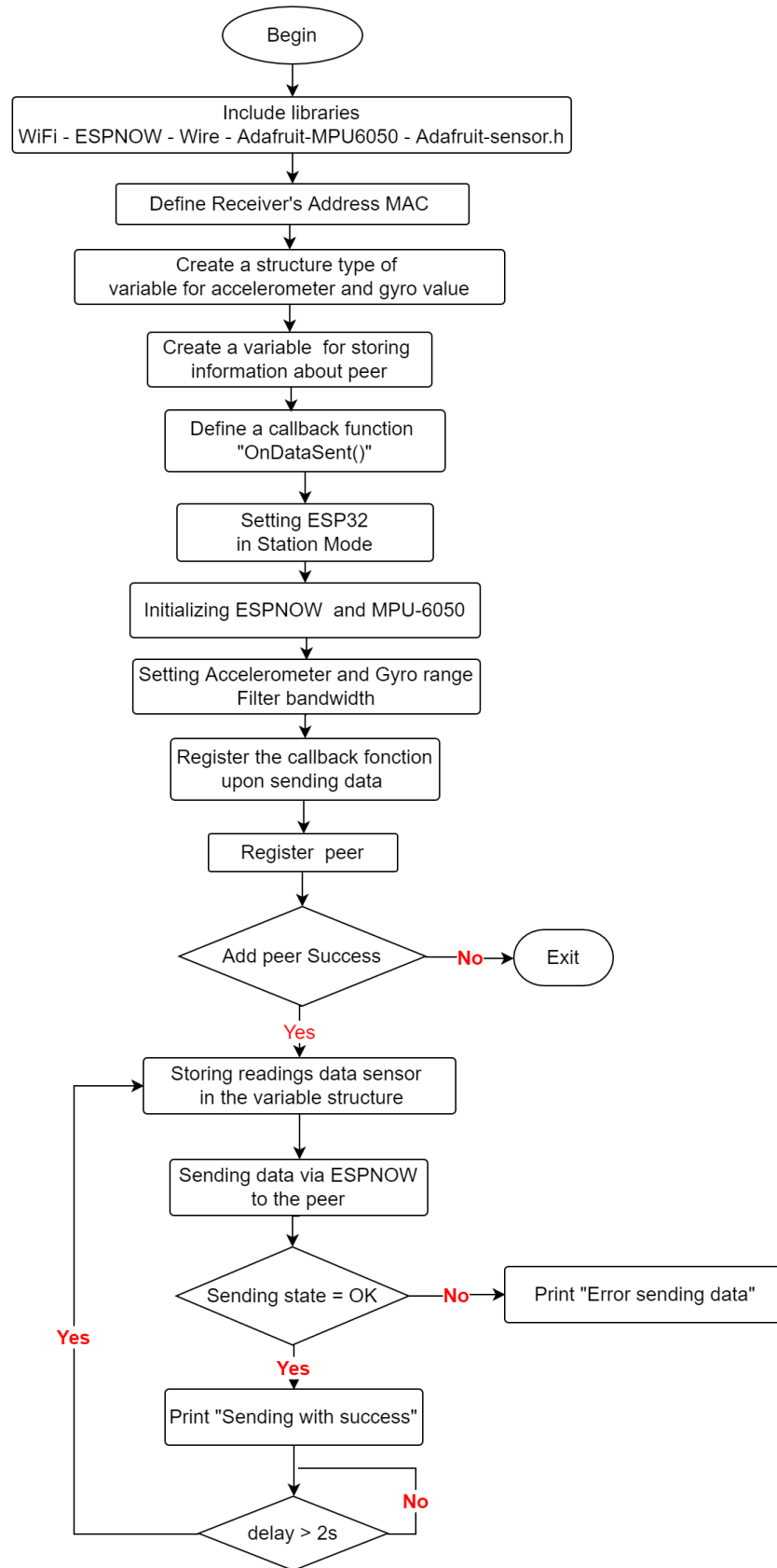
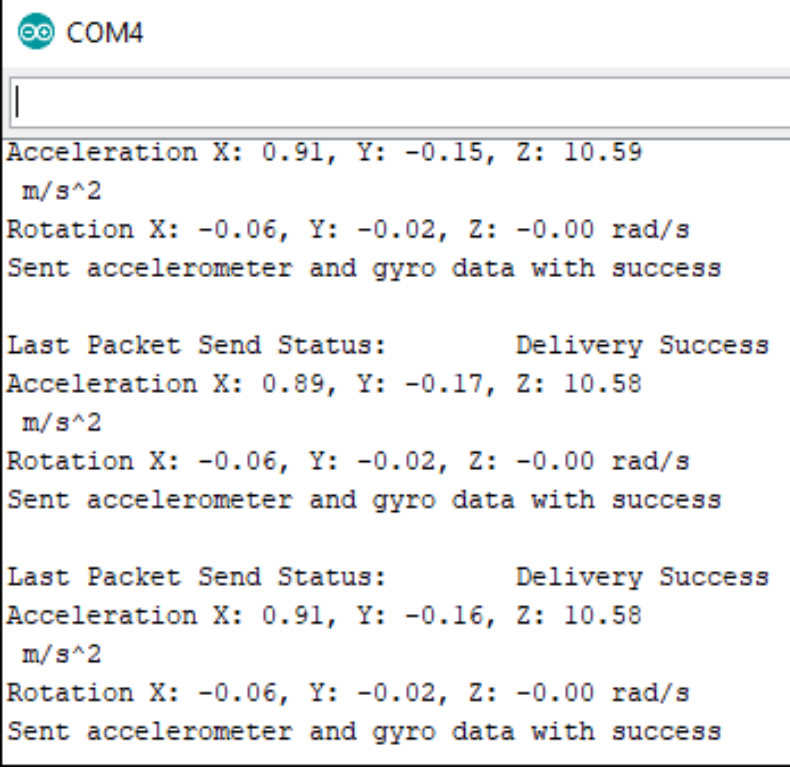


Figure 3.23: Sender Flowchart.



```
COM4
Acceleration X: 0.91, Y: -0.15, Z: 10.59
m/s^2
Rotation X: -0.06, Y: -0.02, Z: -0.00 rad/s
Sent accelerometer and gyro data with success

Last Packet Send Status:      Delivery Success
Acceleration X: 0.89, Y: -0.17, Z: 10.58
m/s^2
Rotation X: -0.06, Y: -0.02, Z: -0.00 rad/s
Sent accelerometer and gyro data with success

Last Packet Send Status:      Delivery Success
Acceleration X: 0.91, Y: -0.16, Z: 10.58
m/s^2
Rotation X: -0.06, Y: -0.02, Z: -0.00 rad/s
Sent accelerometer and gyro data with success
```

Figure 3.24: Serial Monitor Output of Sender.

### 3.8.3 ESP32 receiver flowchart

Let's now move on to explaining the flowchart for data reception in Figure 3.25:

The code begins by including necessary libraries for Wi-Fi and ESP-NOW communication.

Next, it defines the data structure for sensor's readings.

Then, it defines a callback function **OnDataRecv** which is called after an ESP NOW data packet is received.

After that, it sets the device to WiFi station mode, initializes ESPNOW. If the initialization of ESPNOW fails, the code prints an error message and stops execution.

Finally, the code continuously receives sensor data, displays it on the serial monitor as shown in Figure 3.26, receive it from the sender via ESPNOW, and checks the reception status.

# Wireless Hand Gesture Controlled Robot Car Using ESP32 with ESPNOW

---

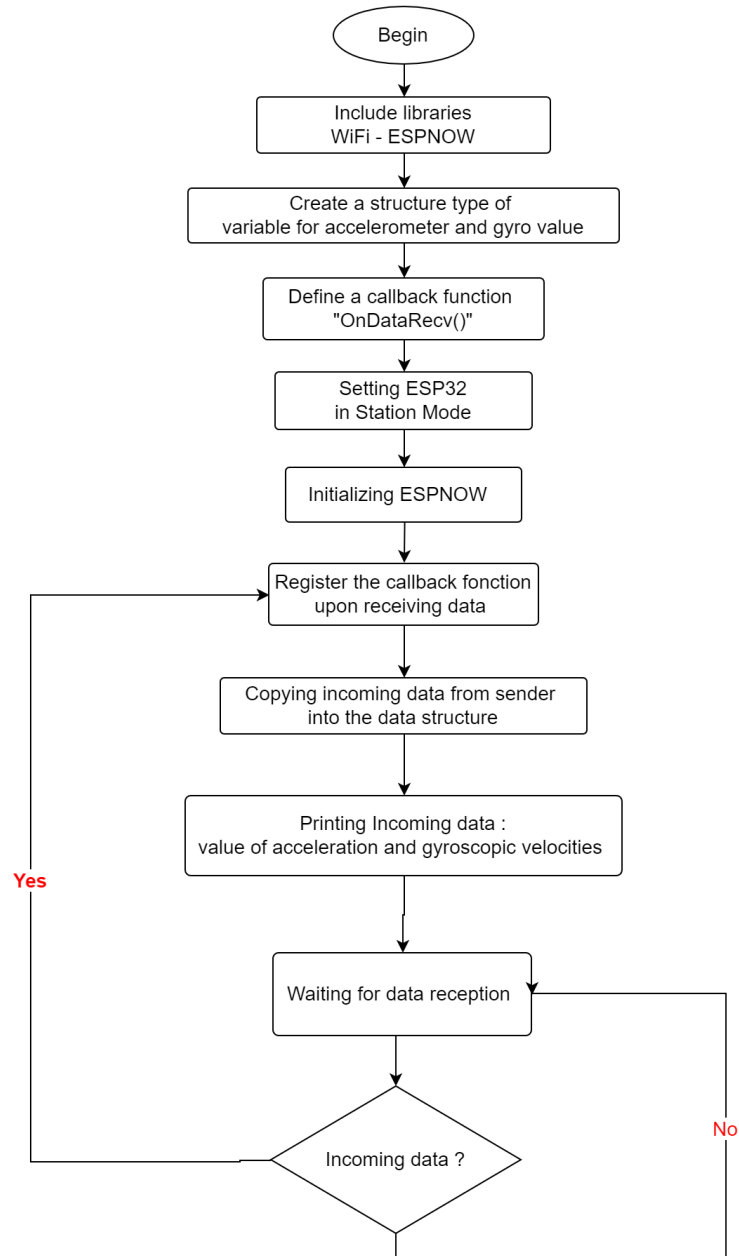


Figure 3.25: Receiver Flowchart.



```
COM3
Received data: 24
Acceleration X: 0.91, Y: -0.15, Z: 10.59
Gyro X: -0.06, Y: -0.02, Z: -0.00
Received data: 24
Acceleration X: 0.89, Y: -0.17, Z: 10.58
Gyro X: -0.06, Y: -0.02, Z: -0.00
Received data: 24
Acceleration X: 0.91, Y: -0.16, Z: 10.58
Gyro X: -0.06, Y: -0.02, Z: -0.00
Received data: 24
Acceleration X: 0.91, Y: -0.16, Z: 10.58
Gyro X: -0.06, Y: -0.02, Z: -0.00
Received data: 24
Acceleration X: 0.91, Y: -0.17, Z: 10.58
Gyro X: -0.06, Y: -0.02, Z: -0.01
```

Figure 3.26: Serial Monitor Output of receiver.

### 3.9 Wireless hand gesture controlled robot car via ESPNOW

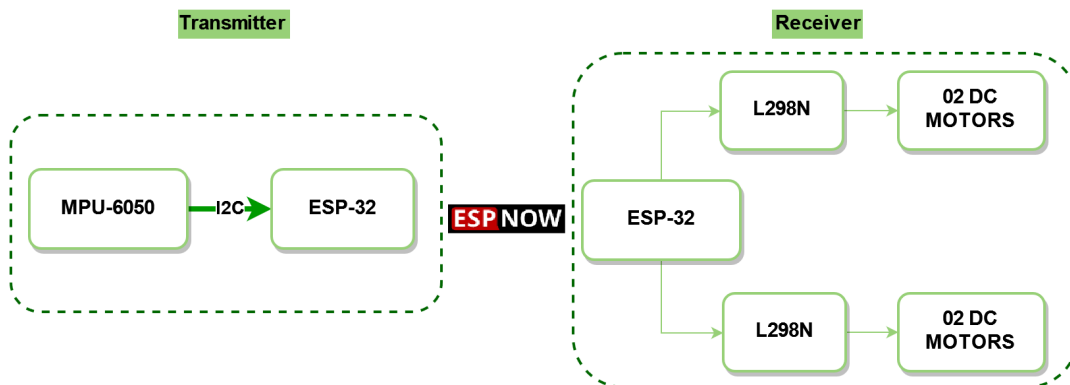


Figure 3.27: Synoptic diagram of Wireless hand gesture controlled robot car via ESPNOW.

#### 3.9.1 Sender part

In the previous code (Figure 3.23), we established ESP-NOW communication to transmit raw sensor data using the Adafruit library. Now, we will further develop the code for a Hand Gesture Controlled Robot Car using ESP32 with ESPNOW, as depicted by the flowchart in Figure 3.28.

## Wireless Hand Gesture Controlled Robot Car Using ESP32 with ESPNOW

---

Hence, it is necessary to utilize the **MPU6050 6Axis MotionApps20.h** library, which enables advanced functionality for sensor fusion and motion processing. This library, developed by Jeff Rowberg, includes advanced functions to utilize the DMP of the MPU6050 which can calculate quaternions and provides filtered data directly from the sensor.

The codes begins by setting up ESP-NOW communication. It then proceed to initialize the I2C bus, the MPU6050, and the DMP. If the DMP initialization was successful, it performs calibration for the MPU6050's accelerometer and gyroscope to generate necessary offsets, enables the DMP, and checks if it is ready.

The DMP of the MPU6050 processes raw data from the accelerometer and gyroscope to generate filtered and fused information, which is then stored in the FIFO buffer.

The DMP attempts to read the latest packet from the FIFO buffer, retrieving the quaternion data, which represent the sensor's orientation. It then computes the gravity vector using the extracted quaternion data which is crucial for accurate orientation estimation.

Yaw, pitch, and roll angles are subsequently calculated using both the quaternion and gravity vector. These angles are then converted from radians to degrees, constrained within the range of -90 to 90 degrees, and mapped to the range of 0 to 254.

Finally, the mapped angle data is sent to the receiver, the robot car, using ESP-NOW communication.

# Wireless Hand Gesture Controlled Robot Car Using ESP32 with ESPNOW

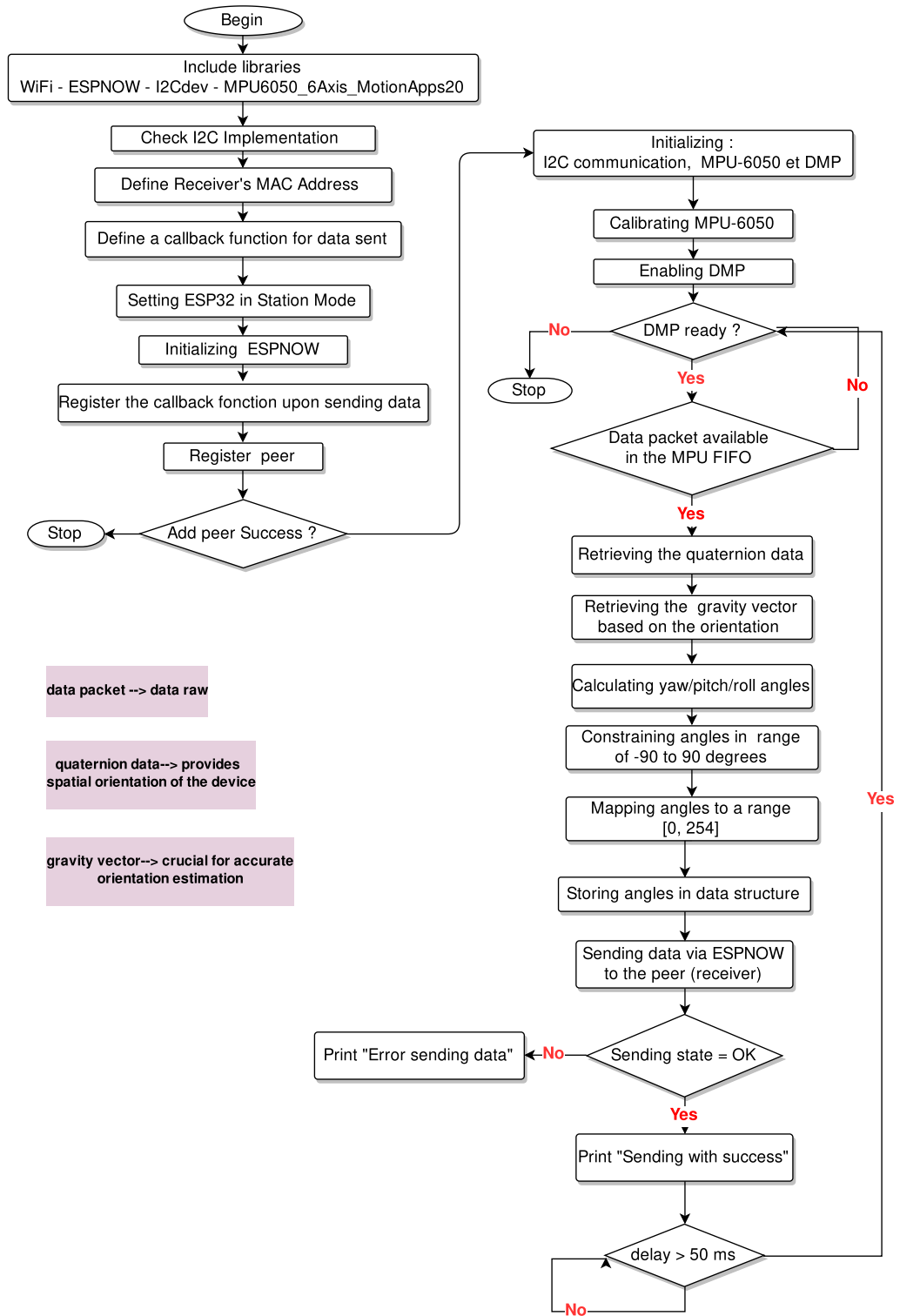


Figure 3.28: Flowchart of sender.

### 3.9.2 Receiver part

#### 1. Schematic diagram of receiver

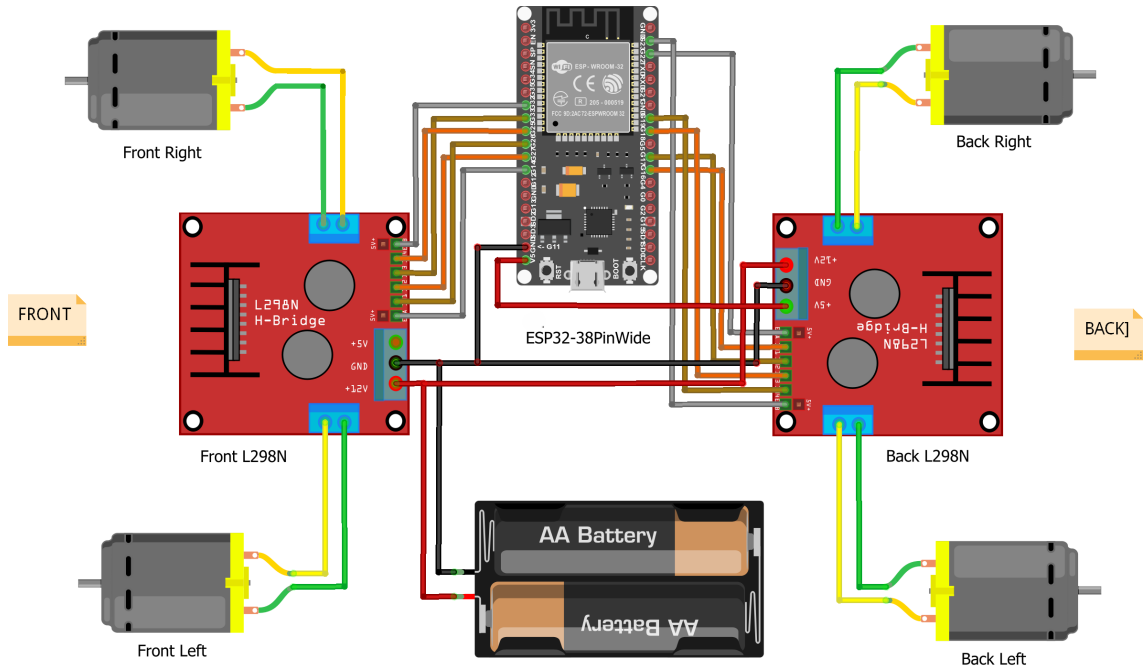


Figure 3.29: Schematic Diagram of Car Receiver.

Front L298N	ESP32	Back L298N	ESP32
IN1	GPIO 26	IN1	GPIO 16
IN2	GPIO 27	IN2	GPIO 17
ENA	GPIO 14	ENA	GPIO 22
IN3	GPIO 33	IN3	GPIO 18
IN4	GPIO 25	IN4	GPIO 19
ENB	GPIO 32	ENB	GPIO 23
GND	GND	GND	GND

Table 3.5: The pin mappings for the ESP32 with Front and Back L298N motor drivers.

#### 2. Building the robot car

Now, we will describe the different stages of robotic car assembly accordingly the schematic diagram in Figure 3.29

- Solder the jumper wires to the terminals of the 04 DC TT motors.
- Connect the TT DC motors to the robot car undercarriage according to the configuration shown in Figure 3.30.

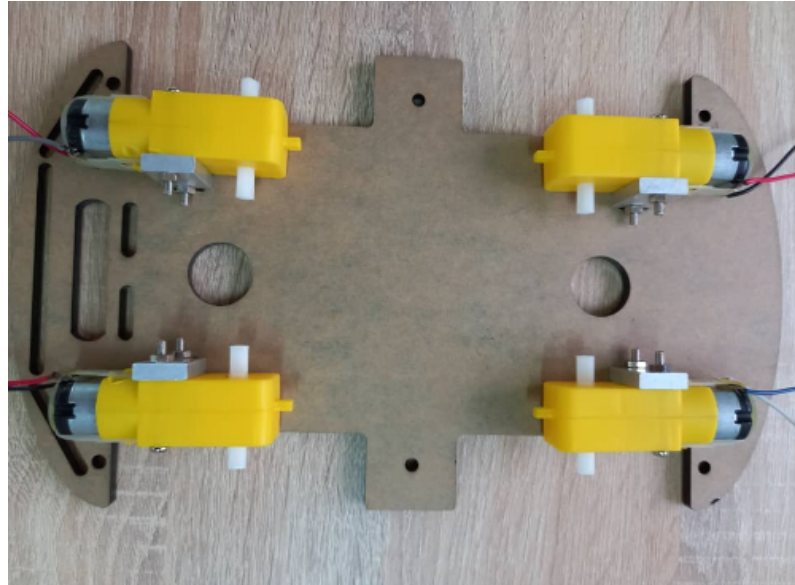


Figure 3.30: DC Motors Installation.

- Assemble the wheels into the motor hub as shown in the figure 3.31.

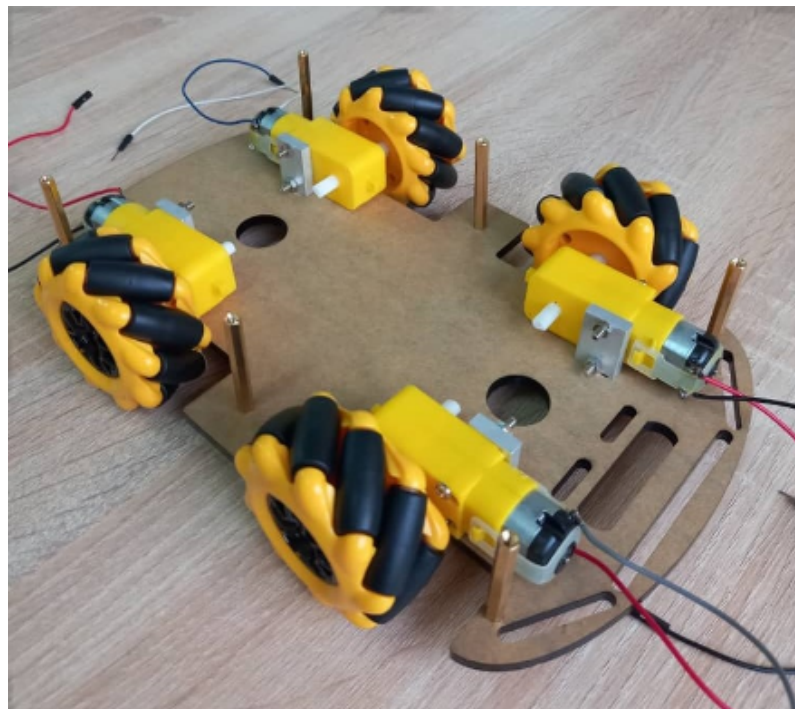


Figure 3.31: DC Attaching Wheels.

- Then install the upper chassis with the connecting wires for the right motors and those for the left motors directed outward through the holes in the upper chassis with the battery installed, as shown in Figure 3.32.

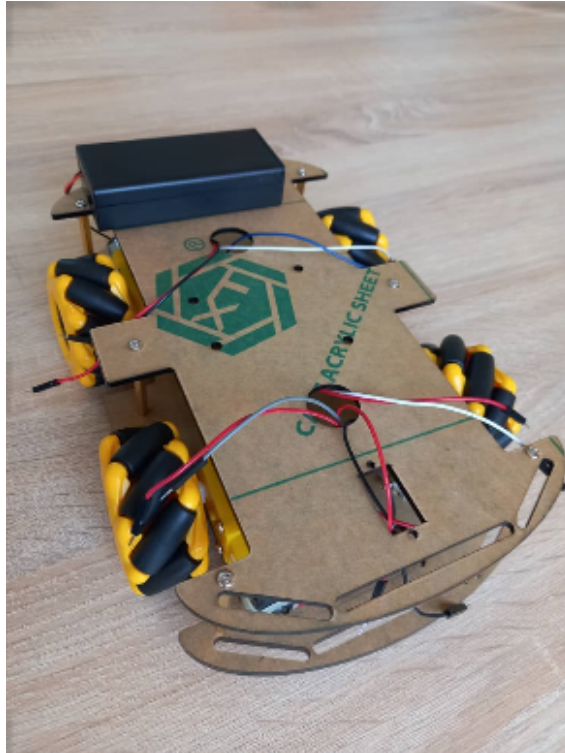


Figure 3.32: Mounting a battery.

- Install the L298N power unit on the upper chassis, then connect the front and back motors to the front and back L298N terminal blocks, as shown in Figure 3.33.

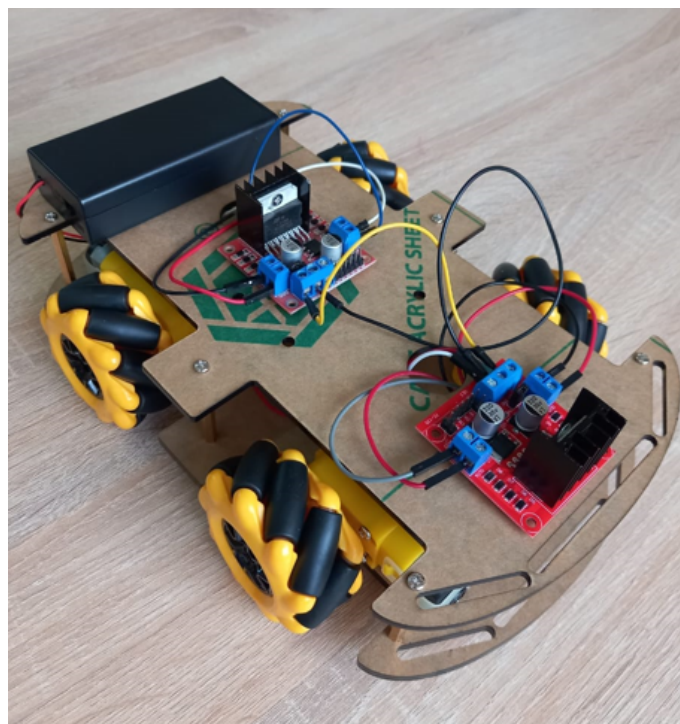


Figure 3.33: Mounting L298N Module.

- Mounting the ESP32 on the upper chassis of the robotic car and connect it to the L298N

# Wireless Hand Gesture Controlled Robot Car Using ESP32 with ESPNOW

---

motors as shown in Figure 3.34.

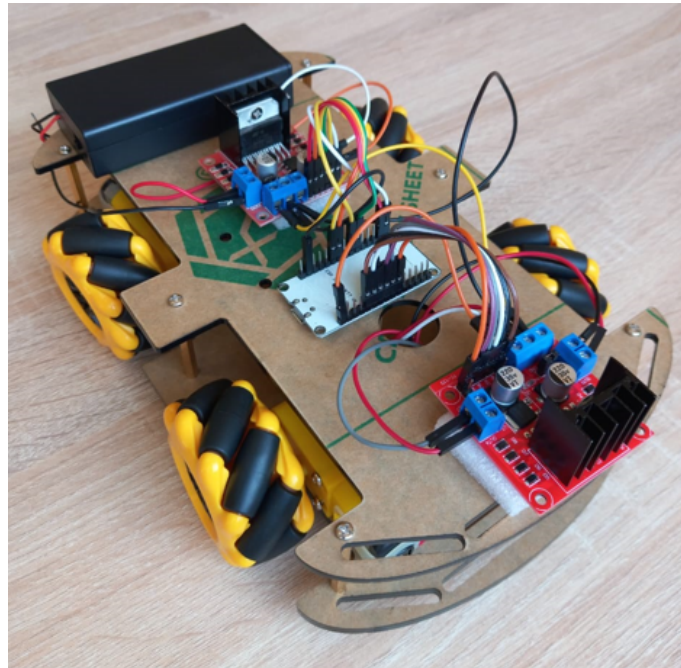


Figure 3.34: Installing ESP32 board into Robot Car's chassis.

### 3. Code for the receiver

The provided code, depicted in Figure 3.35, is designed to control our robot car via ESP-NOW communication. The car is the receiver part. It begins by initializing the motor control pins and configuring PWM settings. Subsequently, it sets up the ESP-NOW protocol, sets the Wi-Fi mode to station mode and registers a callback function.

Upon received data from the sender, the callback function processes the incoming data packet, determining the car's movement based on x,y and z values. Notably, these values are initially within range of -90 to 90 degrees and are mapped to a range of 0 and 254. This means:

- -90° maps to 0
- 0° maps to 127
- 90° maps to 254

In code, values of 75 and 175 are used which correspond to specific thresholds that determine the car's movement direction. So, we define a central Zone between 75 and 175 which Corresponds to the neutral zone where no extreme movement is detected. This helps avoid overly sensitive direction changes.

- A value of 75 roughly corresponds to an angle of -40 degrees.
- A value of 175 roughly corresponds to an angle of 40 degrees.

Based on these thresholds, the code determines the appropriate movement for the car.

- If x values is less than 75, the car moves leftward rotation.
- If y values is less than 75, the car moves left.

## Wireless Hand Gesture Controlled Robot Car Using ESP32 with ESPNOW

---

- If x value is greater than 175, the car moves rightward rotation.
- If y value is greater than 175, the car moves backward.
- If y value is less than 75, the car moves forward.
- If z value is greater than 175, the car moves left.
- If z value is less than 75, the car moves right.
- If both x and y values are less than 75, the car moves forward left.
- If x value is greater than 175 and y value is less than 75, the car moves forward right.
- If x value is less than 75 and y value is greater than 75, the car moves backward left.
- If both x and y values are greater than 175, the car moves backward right.
- Otherwise, the car stops.

The Value of x, y and z is shown in serial Monitor as shown in figure 3.36.



# Wireless Hand Gesture Controlled Robot Car Using ESP32 with ESPNOW

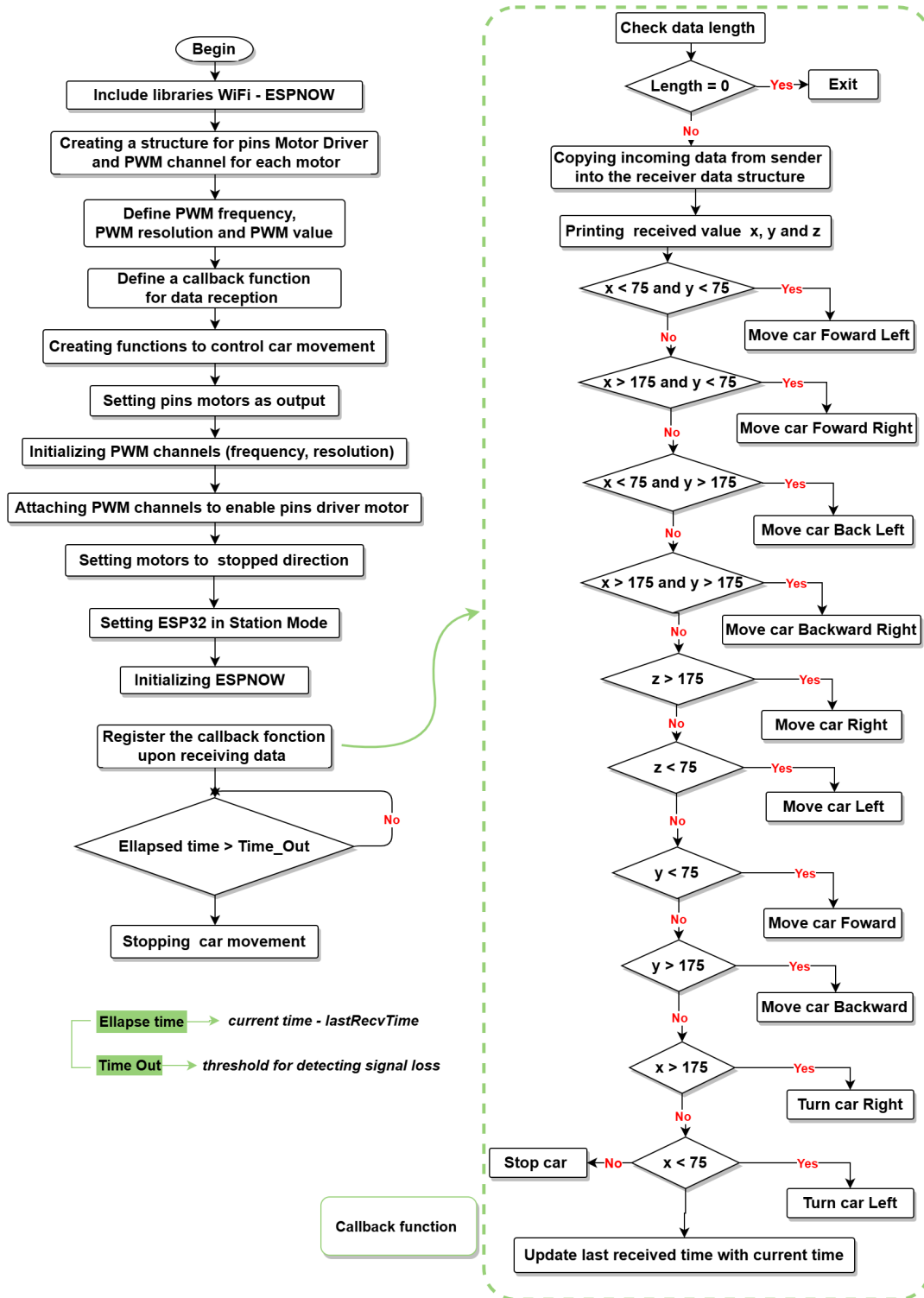
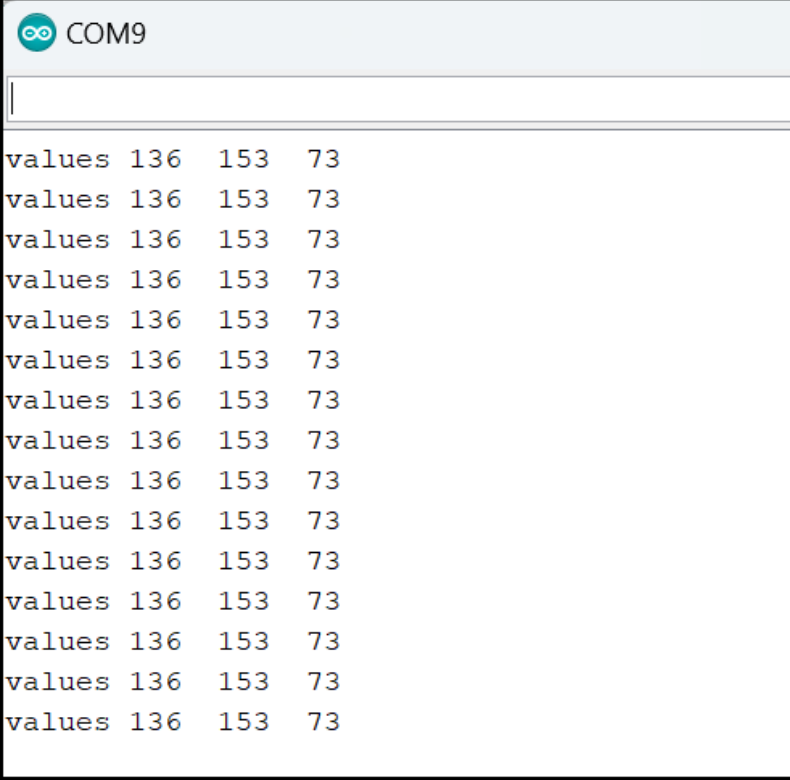


Figure 3.35: Receiver car Flowchart.



```
COM9  
values 136 153 73  
values 136 153 73  
values 136 153 73  
values 136 153 73  
values 136 153 73  
values 136 153 73  
values 136 153 73  
values 136 153 73  
values 136 153 73  
values 136 153 73  
values 136 153 73  
values 136 153 73  
values 136 153 73  
values 136 153 73  
values 136 153 73  
values 136 153 73  
values 136 153 73
```

Figure 3.36: Serial Monitor of the mapped angle data.

## 3.10 Conclusion

The integration of the ESP32, L298N motor driver, DC motors, and Mecanum wheels, combined with the ESPNOW protocol, provides a robust framework for creating a wireless hand gesture-controlled car robot.

This chapter outlined the essential components and steps involved in building and programming the robot, demonstrating the practical application of these technologies in robotics. By mastering these concepts, developers can create innovative and interactive robotic systems with wireless control capabilities.

# Conclusion

The successful completion of this Final Year Project marks a significant achievement in our academic journey, demonstrating our ability to apply theoretical knowledge to practical applications. This project focused on the development and application of the ESP32 microcontroller, exploring its capabilities through various implementations, such as creating a wireless hand gesture-controlled car robot.

In the first part of the project, we conducted a comprehensive study of the ESP32 development board. We detailed its features, specifications, and functionalities, providing a solid foundation for understanding its potential in various applications. This included an in-depth analysis of the ESP-NOW protocol, which is essential for establishing efficient and reliable wireless communication.

The second part involved integrating the ESP32 with the MPU-6050 accelerometer and gyroscope module. We successfully interfaced these components and demonstrated how to read and process data from the sensors. This segment of the project highlighted the importance of precise sensor data acquisition and its application in real-time monitoring systems.

The third part of the project focused on the development of a wireless hand gesture-controlled car robot using the ESP32 and the ESPNOW protocol. This involved interfacing the ESP32 with the L298N motor driver and DC motors, designing control algorithms, and implementing gesture recognition for robot navigation. This application showcased the practical utility of the ESP32 in robotics and wireless communication.

Throughout this project, we encountered several challenges, such as ensuring stable wireless communication and accurate sensor data processing. Overcoming these obstacles required innovative problem-solving and a thorough understanding of the hardware and software components involved.

In conclusion, this project has been an invaluable learning experience, providing us with hands-on expertise in embedded systems, wireless communication, and robotics. The knowledge and skills acquired will undoubtedly serve as a strong foundation for our future professional endeavors. Moving forward, further enhancements could include improving the accuracy of gesture recognition, expanding the range of wireless communication, and exploring additional applications of the ESP32 in various fields..

# References

1. *ESP32 MPU-6050 Accelerometer and Gyroscope (Arduino) — Random Nerd Tutorials* en-US. Jan. 2021. <https://randomnerdtutorials.com/esp32-mpu-6050-accelerometer-gyroscope-arduino/> (2024).
2. *The Internet of Things with ESP32* <http://esp32.net/> (2024).
3. C1F3R. *MPU6050 Transmission Data with ESPNOW* en-US. Mar. 2022. <https://cifertech.net/mpu6050-transmission-data-with-espnw/> (2024).
4. *Algorithme tunisie—Introduction ESP32* <http://algo.tn/esp32/introduction/> (2024).
5. Barybin, O. & Zaitseva, E. *Testing the Security ESP32 Internet of Things Devices in 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T)* (Oct. 2019), 143–146. <https://ieeexplore.ieee.org/abstract/document/9061269> (2024).
6. *ESP32 Web Server MPU-6050 Accelerometer Gyroscope (3D object) — Random Nerd Tutorials* <https://randomnerdtutorials.com/esp32-mpu-6050-web-server/> (2024).
7. *Gesture\_Controlled\_Mecanum\_Wheel\_Car/Car\_Receiver\_Simple\_Movement/Car\_Receiver\_Simple\_Movement* · un0038998/*Gesture\_Controlled\_Mecanum\_Wheel\_Car* en. [https://github.com/un0038998/Gesture\\_Controlled\\_Mecanum\\_Wheel\\_Car/blob/main/Car\\_Receiver\\_Simple\\_Movement/Car\\_Receiver\\_Simple\\_Movement.ino](https://github.com/un0038998/Gesture_Controlled_Mecanum_Wheel_Car/blob/main/Car_Receiver_Simple_Movement/Car_Receiver_Simple_Movement.ino) (2024).
8. Nehesh. *Hand Gesture Controlled Car* en. <https://www.instructables.com/Hand-Gesture-Controlled-Car/> (2024).
9. *Gesture Control Robot — project* en. <https://www.electronicwings.com/users/NilutpolKashyap/projects/1963/gesture-control-robot> (2024).
10. *Gyroscope accéléromètre ESP32 Web Server MPU-6050 (objet 3D)* fr-FR. Section: ESP Tutoriels. Mar. 2023. <https://www.raspberryme.com/gyroscope-accelerometre-esp32-web-server-mpu-6050-objet-3d/> (2024).

## References

---

11. *Tutoriel L298N : fonctionnement, branchement, code arduino* fr-FR. Section: Tutoriels. May 2021. <https://passionelectronique.fr/tutoriel-l298n/> (2024).
12. *ESP-NOW Introduction — ESP32* en. <https://www.electronicwings.com/esp32/esp-now-introduction> (2024).
13. *Getting Started with ESP-NOW (ESP32 with Arduino IDE) — Random Nerd Tutorials* en-US. Jan. 2020. <https://randomnerdtutorials.com/esp-now-esp32-arduino-ide/> (2024).
14. Workshop, D. *Mecanum Wheel Robot with ESP-NOW Remote Control* en-US. Dec. 2022. <https://dronebotworkshop.com/mecanum/> (2024).
15. *Introduction to ESP32 — ESP32* en. <https://www.electronicwings.com/esp32/introduction-to-esp32> (2024).
16. *MPU6050 Gyroscope Interfacing with ESP32 — ESP32* en. <https://www.electronicwings.com/esp32/mpu6050-gyroscope-interfacing-with-esp32> (2024).

# Appendix A

## Datasheet of L298N



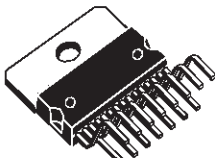
# L298

## DUAL FULL-BRIDGE DRIVER

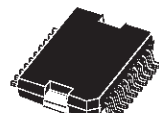
- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)

### DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the con-



**Multiwatt15**

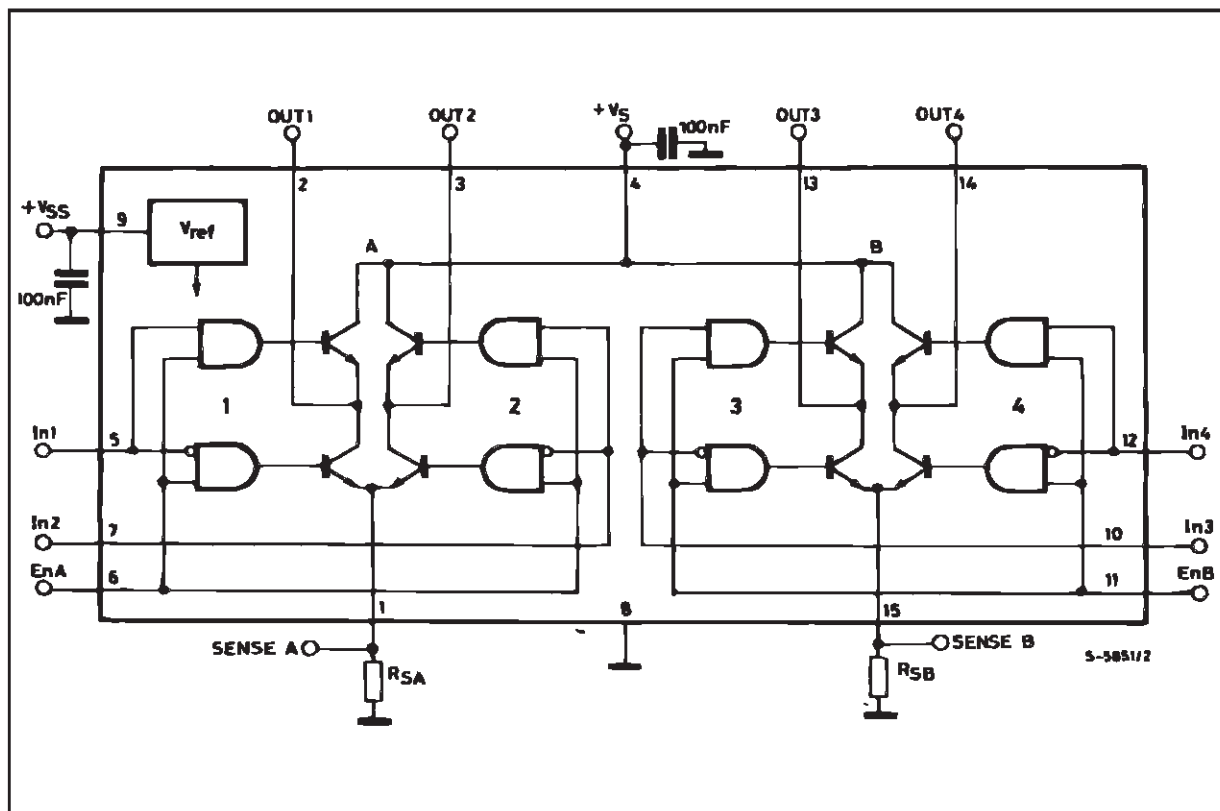


**PowerSO20**

**ORDERING NUMBERS :** L298N (Multiwatt Vert.)  
 L298HN (Multiwatt Horiz.)  
 L298P (PowerSO20)

nection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

### BLOCK DIAGRAM

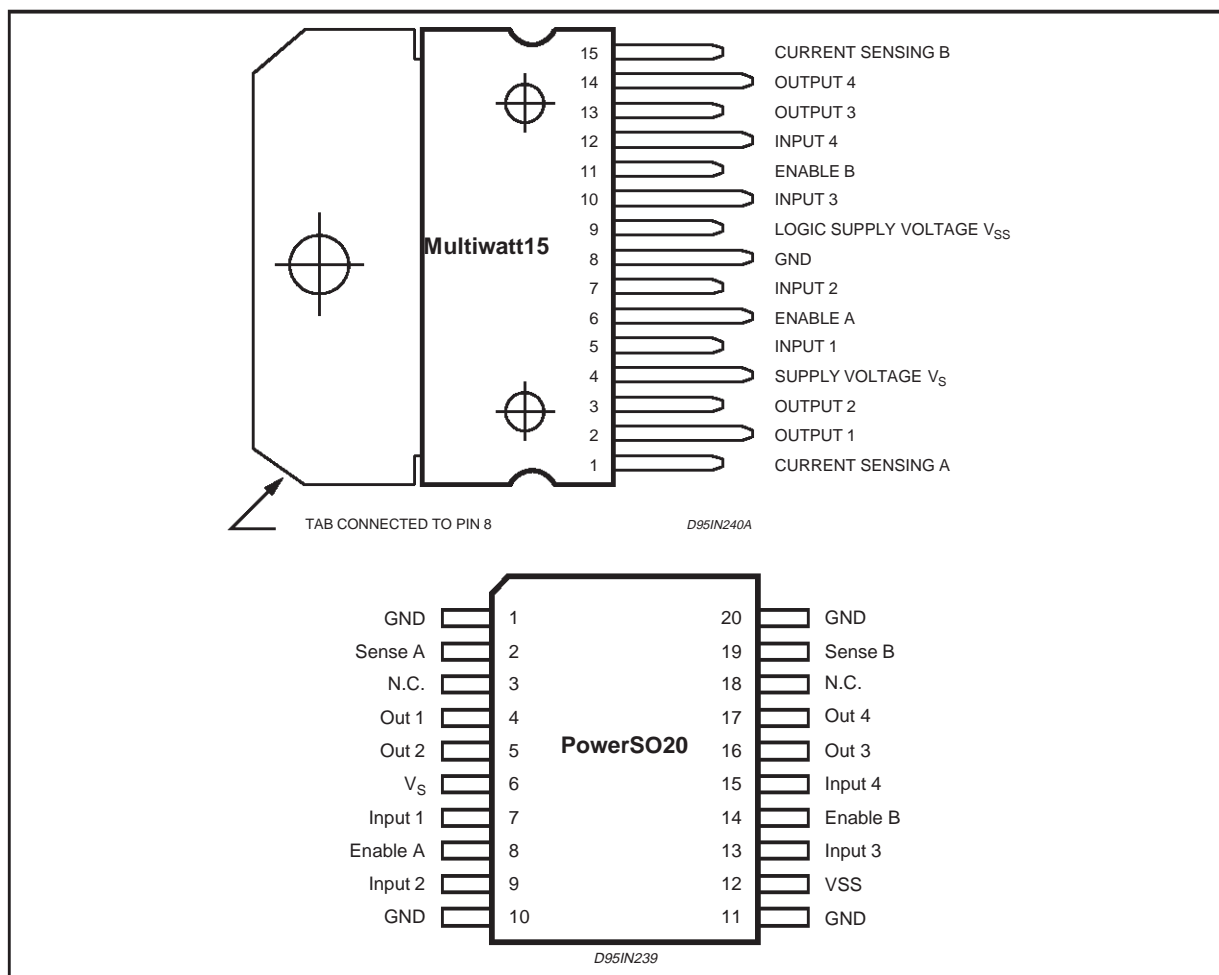


**L298**

**ABSOLUTE MAXIMUM RATINGS**

Symbol	Parameter	Value	Unit
$V_S$	Power Supply	50	V
$V_{SS}$	Logic Supply Voltage	7	V
$V_I, V_{En}$	Input and Enable Voltage	-0.3 to 7	V
$I_O$	Peak Output Current (each Channel)		
	- Non Repetitive ( $t = 100\mu s$ )	3	A
	- Repetitive (80% on -20% off; $t_{on} = 10ms$ )	2.5	A
	-DC Operation	2	A
$V_{sens}$	Sensing Voltage	-1 to 2.3	V
$P_{tot}$	Total Power Dissipation ( $T_{case} = 75^\circ C$ )	25	W
$T_{op}$	Junction Operating Temperature	-25 to 130	$^\circ C$
$T_{stg}, T_j$	Storage and Junction Temperature	-40 to 150	$^\circ C$

**PIN CONNECTIONS (top view)**



**THERMAL DATA**

Symbol	Parameter		PowerSO20	Multiwatt15	Unit
$R_{th\ j-case}$	Thermal Resistance Junction-case	Max.	-	3	$^\circ C/W$
$R_{th\ j-amb}$	Thermal Resistance Junction-ambient	Max.	13 (*)	35	$^\circ C/W$

(\*) Mounted on aluminum substrate





**PIN FUNCTIONS** (refer to the block diagram)

MW.15	PowerSO	Name	Function
1;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V <sub>S</sub>	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
6;11	8;14	Enable A; Enable B	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	V <sub>SS</sub>	Supply Voltage for the Logic Blocks. A100nF capacitor must be connected between this pin and ground.
10; 12	13;15	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
13; 14	16;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
–	3;18	N.C.	Not Connected

**ELECTRICAL CHARACTERISTICS** (V<sub>S</sub> = 42V; V<sub>SS</sub> = 5V, T<sub>j</sub> = 25°C; unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V <sub>S</sub>	Supply Voltage (pin 4)	Operative Condition	V <sub>IH</sub> +2.5		46	V
V <sub>SS</sub>	Logic Supply Voltage (pin 9)		4.5	5	7	V
I <sub>S</sub>	Quiescent Supply Current (pin 4)	V <sub>en</sub> = H; I <sub>L</sub> = 0 V <sub>i</sub> = L V <sub>i</sub> = H		13 50	22 70	mA mA
		V <sub>en</sub> = L V <sub>i</sub> = X			4	mA
I <sub>SS</sub>	Quiescent Current from V <sub>SS</sub> (pin 9)	V <sub>en</sub> = H; I <sub>L</sub> = 0 V <sub>i</sub> = L V <sub>i</sub> = H		24 7	36 12	mA mA
		V <sub>en</sub> = L V <sub>i</sub> = X			6	mA
V <sub>iL</sub>	Input Low Voltage (pins 5, 7, 10, 12)		–0.3		1.5	V
V <sub>iH</sub>	Input High Voltage (pins 5, 7, 10, 12)		2.3		V <sub>SS</sub>	V
I <sub>iL</sub>	Low Voltage Input Current (pins 5, 7, 10, 12)	V <sub>i</sub> = L			–10	μA
I <sub>iH</sub>	High Voltage Input Current (pins 5, 7, 10, 12)	V <sub>i</sub> = H ≤ V <sub>SS</sub> – 0.6V		30	100	μA
V <sub>en</sub> = L	Enable Low Voltage (pins 6, 11)		–0.3		1.5	V
V <sub>en</sub> = H	Enable High Voltage (pins 6, 11)		2.3		V <sub>SS</sub>	V
I <sub>en</sub> = L	Low Voltage Enable Current (pins 6, 11)	V <sub>en</sub> = L			–10	μA
I <sub>en</sub> = H	High Voltage Enable Current (pins 6, 11)	V <sub>en</sub> = H ≤ V <sub>SS</sub> – 0.6V		30	100	μA
V <sub>CEsat(H)</sub>	Source Saturation Voltage	I <sub>L</sub> = 1A I <sub>L</sub> = 2A	0.95	1.35 2	1.7 2.7	V V
V <sub>CEsat(L)</sub>	Sink Saturation Voltage	I <sub>L</sub> = 1A (5) I <sub>L</sub> = 2A (5)	0.85	1.2 1.7	1.6 2.3	V V
V <sub>CEsat</sub>	Total Drop	I <sub>L</sub> = 1A (5) I <sub>L</sub> = 2A (5)	1.80		3.2 4.9	V V
V <sub>sens</sub>	Sensing Voltage (pins 1, 15)		–1 (1)		2	V



L298

ELECTRICAL CHARACTERISTICS (continued)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
T <sub>1</sub> (V <sub>i</sub> )	Source Current Turn-off Delay	0.5 V <sub>i</sub> to 0.9 I <sub>L</sub> (2); (4)		1.5		μs
T <sub>2</sub> (V <sub>i</sub> )	Source Current Fall Time	0.9 I <sub>L</sub> to 0.1 I <sub>L</sub> (2); (4)		0.2		μs
T <sub>3</sub> (V <sub>i</sub> )	Source Current Turn-on Delay	0.5 V <sub>i</sub> to 0.1 I <sub>L</sub> (2); (4)		2		μs
T <sub>4</sub> (V <sub>i</sub> )	Source Current Rise Time	0.1 I <sub>L</sub> to 0.9 I <sub>L</sub> (2); (4)		0.7		μs
T <sub>5</sub> (V <sub>i</sub> )	Sink Current Turn-off Delay	0.5 V <sub>i</sub> to 0.9 I <sub>L</sub> (3); (4)		0.7		μs
T <sub>6</sub> (V <sub>i</sub> )	Sink Current Fall Time	0.9 I <sub>L</sub> to 0.1 I <sub>L</sub> (3); (4)		0.25		μs
T <sub>7</sub> (V <sub>i</sub> )	Sink Current Turn-on Delay	0.5 V <sub>i</sub> to 0.9 I <sub>L</sub> (3); (4)		1.6		μs
T <sub>8</sub> (V <sub>i</sub> )	Sink Current Rise Time	0.1 I <sub>L</sub> to 0.9 I <sub>L</sub> (3); (4)		0.2		μs
f <sub>c</sub> (V <sub>i</sub> )	Commutation Frequency	I <sub>L</sub> = 2A		25	40	KHz
T <sub>1</sub> (V <sub>en</sub> )	Source Current Turn-off Delay	0.5 V <sub>en</sub> to 0.9 I <sub>L</sub> (2); (4)		3		μs
T <sub>2</sub> (V <sub>en</sub> )	Source Current Fall Time	0.9 I <sub>L</sub> to 0.1 I <sub>L</sub> (2); (4)		1		μs
T <sub>3</sub> (V <sub>en</sub> )	Source Current Turn-on Delay	0.5 V <sub>en</sub> to 0.1 I <sub>L</sub> (2); (4)		0.3		μs
T <sub>4</sub> (V <sub>en</sub> )	Source Current Rise Time	0.1 I <sub>L</sub> to 0.9 I <sub>L</sub> (2); (4)		0.4		μs
T <sub>5</sub> (V <sub>en</sub> )	Sink Current Turn-off Delay	0.5 V <sub>en</sub> to 0.9 I <sub>L</sub> (3); (4)		2.2		μs
T <sub>6</sub> (V <sub>en</sub> )	Sink Current Fall Time	0.9 I <sub>L</sub> to 0.1 I <sub>L</sub> (3); (4)		0.35		μs
T <sub>7</sub> (V <sub>en</sub> )	Sink Current Turn-on Delay	0.5 V <sub>en</sub> to 0.9 I <sub>L</sub> (3); (4)		0.25		μs
T <sub>8</sub> (V <sub>en</sub> )	Sink Current Rise Time	0.1 I <sub>L</sub> to 0.9 I <sub>L</sub> (3); (4)		0.1		μs

- 1) Sensing voltage can be -1 V for t ≤ 50 μsec; in steady state V<sub>sens</sub> min ≥ -0.5 V.
- 2) See fig. 2.
- 3) See fig. 4.
- 4) The load must be a pure resistor.

Figure 1 : Typical Saturation Voltage vs. Output Current.

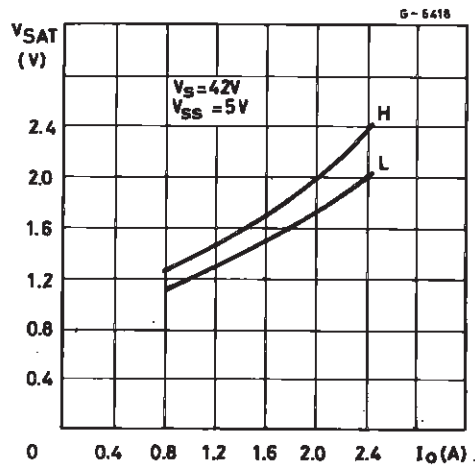
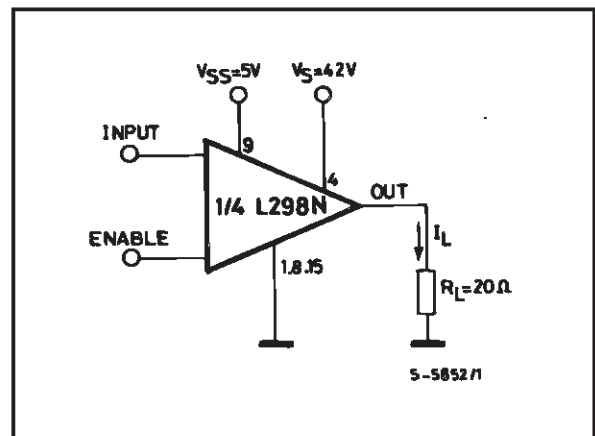


Figure 2 : Switching Times Test Circuits.



Note: For INPUT Switching, set EN = H  
For ENABLE Switching, set IN = H

# Appendix B

## Datasheet of MPU-6050 module

	<p><b>MPU-6000/MPU-6050 Product Specification</b></p>	<p>Document Number: PS-MPU-6000A-00  Revision: 3.4  Release Date: 08/19/2013</p>
---	---	--

### 3 Product Overview

#### 3.1 MPU-60X0 Overview

MotionInterface™ is becoming a “must-have” function being adopted by smartphone and tablet manufacturers due to the enormous value it adds to the end user experience. In smartphones, it finds use in applications such as gesture commands for applications and phone control, enhanced gaming, augmented reality, panoramic photo capture and viewing, and pedestrian and vehicle navigation. With its ability to precisely and accurately track user motions, MotionTracking technology can convert handsets and tablets into powerful 3D intelligent devices that can be used in applications ranging from health and fitness monitoring to location-based services. Key requirements for MotionInterface enabled devices are small package size, low power consumption, high accuracy and repeatability, high shock tolerance, and application specific performance programmability – all at a low consumer price point.

The MPU-60X0 is the world’s first integrated 6-axis MotionTracking device that combines a 3-axis gyroscope, 3-axis accelerometer, and a Digital Motion Processor™ (DMP) all in a small 4x4x0.9mm package. With its dedicated I<sup>2</sup>C sensor bus, it directly accepts inputs from an external 3-axis compass to provide a complete 9-axis MotionFusion™ output. The MPU-60X0 MotionTracking device, with its 6-axis integration, on-board MotionFusion™, and run-time calibration firmware, enables manufacturers to eliminate the costly and complex selection, qualification, and system level integration of discrete devices, guaranteeing optimal motion performance for consumers. The MPU-60X0 is also designed to interface with multiple non-inertial digital sensors, such as pressure sensors, on its auxiliary I<sup>2</sup>C port. The MPU-60X0 is footprint compatible with the MPU-30X0 family.

The MPU-60X0 features three 16-bit analog-to-digital converters (ADCs) for digitizing the gyroscope outputs and three 16-bit ADCs for digitizing the accelerometer outputs. For precision tracking of both fast and slow motions, the parts feature a user-programmable gyroscope full-scale range of ±250, ±500, ±1000, and ±2000°/sec (dps) and a user-programmable accelerometer full-scale range of ±2g, ±4g, ±8g, and ±16g.

An on-chip 1024 Byte FIFO buffer helps lower system power consumption by allowing the system processor to read the sensor data in bursts and then enter a low-power mode as the MPU collects more data. With all the necessary on-chip processing and sensor components required to support many motion-based use cases, the MPU-60X0 uniquely enables low-power MotionInterface applications in portable applications with reduced processing requirements for the system processor. By providing an integrated MotionFusion output, the DMP in the MPU-60X0 offloads the intensive MotionProcessing computation requirements from the system processor, minimizing the need for frequent polling of the motion sensor output.

Communication with all registers of the device is performed using either I<sup>2</sup>C at 400kHz or SPI at 1MHz (MPU-6000 only). For applications requiring faster communications, the sensor and interrupt registers may be read using SPI at 20MHz (MPU-6000 only). Additional features include an embedded temperature sensor and an on-chip oscillator with ±1% variation over the operating temperature range.

By leveraging its patented and volume-proven Nasiri-Fabrication platform, which integrates MEMS wafers with companion CMOS electronics through wafer-level bonding, InvenSense has driven the MPU-60X0 package size down to a revolutionary footprint of 4x4x0.9mm (QFN), while providing the highest performance, lowest noise, and the lowest cost semiconductor packaging required for handheld consumer electronic devices. The part features a robust 10,000g shock tolerance, and has programmable low-pass filters for the gyroscopes, accelerometers, and the on-chip temperature sensor.

For power supply flexibility, the MPU-60X0 operates from VDD power supply voltage range of 2.375V-3.46V. Additionally, the MPU-6050 provides a VLOGIC reference pin (in addition to its analog supply pin: VDD), which sets the logic levels of its I<sup>2</sup>C interface. The VLOGIC voltage may be 1.8V±5% or VDD.

The MPU-6000 and MPU-6050 are identical, except that the MPU-6050 supports the I<sup>2</sup>C serial interface only, and has a separate VLOGIC reference pin. The MPU-6000 supports both I<sup>2</sup>C and SPI interfaces and has a single supply pin, VDD, which is both the device’s logic reference supply and the analog supply for the part. The table below outlines these differences:

## Datasheet of MPU-6050 module

	<b>MPU-6000/MPU-6050 Product Specification</b>	Document Number: PS-MPU-6000A-00 Revision: 3.4 Release Date: 08/19/2013
---	--	---

### Primary Differences between MPU-6000 and MPU-6050

Part / Item	MPU-6000	MPU-6050
VDD	2.375V-3.46V	2.375V-3.46V
VLOGIC	n/a	1.71V to VDD
Serial Interfaces Supported	I <sup>2</sup> C, SPI	I <sup>2</sup> C
Pin 8	/CS	VLOGIC
Pin 9	AD0/SDO	AD0
Pin 23	SCL/SCLK	SCL
Pin 24	SDA/SDI	SDA

	<b>MPU-6000/MPU-6050 Product Specification</b>	Document Number: PS-MPU-6000A-00 Revision: 3.4 Release Date: 08/19/2013
---	--	---

## 4 Applications

- *BlurFree*<sup>™</sup> technology (for Video/Still Image Stabilization)
- *AirSign*<sup>™</sup> technology (for Security/Authentication)
- *TouchAnywhere*<sup>™</sup> technology (for “no touch” UI Application Control/Navigation)
- *MotionCommand*<sup>™</sup> technology (for Gesture Short-cuts)
- Motion-enabled game and application framework
- InstantGesture<sup>™</sup> iG<sup>™</sup> gesture recognition
- Location based services, points of interest, and dead reckoning
- Handset and portable gaming
- Motion-based game controllers
- 3D remote controls for Internet connected DTVs and set top boxes, 3D mice
- Wearable sensors for health, fitness and sports
- Toys

	<p><b>MPU-6000/MPU-6050 Product Specification</b></p>	<p>Document Number: PS-MPU-6000A-00 Revision: 3.4 Release Date: 08/19/2013</p>
---	---	--

## 5 Features

### 5.1 Gyroscope Features

The triple-axis MEMS gyroscope in the MPU-60X0 includes a wide range of features:

- Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of  $\pm 250$ ,  $\pm 500$ ,  $\pm 1000$ , and  $\pm 2000^\circ/\text{sec}$
- External sync signal connected to the FSYNC pin supports image, video and GPS synchronization
- Integrated 16-bit ADCs enable simultaneous sampling of gyros
- Enhanced bias and sensitivity temperature stability reduces the need for user calibration
- Improved low-frequency noise performance
- Digitally-programmable low-pass filter
- Gyroscope operating current: 3.6mA
- Standby current: 5 $\mu$ A
- Factory calibrated sensitivity scale factor
- User self-test

### 5.2 Accelerometer Features

The triple-axis MEMS accelerometer in MPU-60X0 includes a wide range of features:

- Digital-output triple-axis accelerometer with a programmable full scale range of  $\pm 2g$ ,  $\pm 4g$ ,  $\pm 8g$  and  $\pm 16g$
- Integrated 16-bit ADCs enable simultaneous sampling of accelerometers while requiring no external multiplexer
- Accelerometer normal operating current: 500 $\mu$ A
- Low power accelerometer mode current: 10 $\mu$ A at 1.25Hz, 20 $\mu$ A at 5Hz, 60 $\mu$ A at 20Hz, 110 $\mu$ A at 40Hz
- Orientation detection and signaling
- Tap detection
- User-programmable interrupts
- High-G interrupt
- User self-test

### 5.3 Additional Features

The MPU-60X0 includes the following additional features:

- 9-Axis MotionFusion by the on-chip Digital Motion Processor (DMP)
- Auxiliary master I<sup>2</sup>C bus for reading data from external sensors (e.g., magnetometer)
- 3.9mA operating current when all 6 motion sensing axes and the DMP are enabled
- VDD supply voltage range of 2.375V-3.46V
- Flexible VLOGIC reference voltage supports multiple I<sup>2</sup>C interface voltages (MPU-6050 only)
- Smallest and thinnest QFN package for portable devices: 4x4x0.9mm
- Minimal cross-axis sensitivity between the accelerometer and gyroscope axes
- 1024 byte FIFO buffer reduces power consumption by allowing host processor to read the data in bursts and then go into a low-power mode as the MPU collects more data
- Digital-output temperature sensor
- User-programmable digital filters for gyroscope, accelerometer, and temp sensor
- 10,000 g shock tolerant
- 400kHz Fast Mode I<sup>2</sup>C for communicating with all registers
- 1MHz SPI serial interface for communicating with all registers (MPU-6000 only)
- 20MHz SPI serial interface for reading sensor and interrupt registers (MPU-6000 only)

	<b>MPU-6000/MPU-6050 Product Specification</b>	Document Number: PS-MPU-6000A-00 Revision: 3.4 Release Date: 08/19/2013
---	--	---

- MEMS structure hermetically sealed and bonded at wafer level
- RoHS and Green compliant

#### 5.4 MotionProcessing

- Internal Digital Motion Processing™ (DMP™) engine supports 3D MotionProcessing and gesture recognition algorithms
- The MPU-60X0 collects gyroscope and accelerometer data while synchronizing data sampling at a user defined rate. The total dataset obtained by the MPU-60X0 includes 3-Axis gyroscope data, 3-Axis accelerometer data, and temperature data. The MPU's calculated output to the system processor can also include heading data from a digital 3-axis third party magnetometer.
- The FIFO buffers the complete data set, reducing timing requirements on the system processor by allowing the processor burst read the FIFO data. After burst reading the FIFO data, the system processor can save power by entering a low-power sleep mode while the MPU collects more data.
- Programmable interrupt supports features such as gesture recognition, panning, zooming, scrolling, tap detection, and shake detection
- Digitally-programmable low-pass filters
- Low-power pedometer functionality allows the host processor to sleep while the DMP maintains the step count.

#### 5.5 Clocking

- On-chip timing generator  $\pm 1\%$  frequency variation over full temperature range
- Optional external clock inputs of 32.768kHz or 19.2MHz



	<b>MPU-6000/MPU-6050 Product Specification</b>	Document Number: PS-MPU-6000A-00 Revision: 3.4 Release Date: 08/19/2013
---	--	---

## 6 Electrical Characteristics

### 6.1 Gyroscope Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T<sub>A</sub> = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
<b>GYROSCOPE SENSITIVITY</b>						
Full-Scale Range	FS_SEL=0		±250		°/s	
	FS_SEL=1		±500		°/s	
	FS_SEL=2		±1000		°/s	
	FS_SEL=3		±2000		°/s	
Gyroscope ADC Word Length			16		bits	
Sensitivity Scale Factor	FS_SEL=0		131		LSB/(°/s)	
	FS_SEL=1		65.5		LSB/(°/s)	
	FS_SEL=2		32.8		LSB/(°/s)	
	FS_SEL=3		16.4		LSB/(°/s)	
Sensitivity Scale Factor Tolerance	25°C	-3		+3	%	
Sensitivity Scale Factor Variation Over Temperature			±2		%	
Nonlinearity	Best fit straight line; 25°C		0.2		%	
Cross-Axis Sensitivity			±2		%	
<b>GYROSCOPE ZERO-RATE OUTPUT (ZRO)</b>						
Initial ZRO Tolerance	25°C		±20		°/s	
ZRO Variation Over Temperature	-40°C to +85°C		±20		°/s	
Power-Supply Sensitivity (1-10Hz)	Sine wave, 100mVpp; VDD=2.5V		0.2		°/s	
Power-Supply Sensitivity (10 - 250Hz)	Sine wave, 100mVpp; VDD=2.5V		0.2		°/s	
Power-Supply Sensitivity (250Hz - 100kHz)	Sine wave, 100mVpp; VDD=2.5V		4		°/s	
Linear Acceleration Sensitivity	Static		0.1		°/s/g	
<b>SELF-TEST RESPONSE</b>						
Relative	Change from factory trim	-14		14	%	1
<b>GYROSCOPE NOISE PERFORMANCE</b>	<b>FS_SEL=0</b>					
Total RMS Noise	DLPFCFG=2 (100Hz)		0.05		°/s-rms	
Low-frequency RMS noise	Bandwidth 1Hz to 10Hz		0.033		°/s-rms	
Rate Noise Spectral Density	At 10Hz		0.005		°/s/√Hz	
<b>GYROSCOPE MECHANICAL FREQUENCIES</b>						
X-Axis		30	33	36	kHz	
Y-Axis		27	30	33	kHz	
Z-Axis		24	27	30	kHz	
<b>LOW PASS FILTER RESPONSE</b>						
	Programmable Range	5		256	Hz	
<b>OUTPUT DATA RATE</b>						
	Programmable	4		8,000	Hz	
<b>GYROSCOPE START-UP TIME</b>	<b>DLPFCFG=0</b>					
ZRO Settling (from power-on)	to ±1°/s of Final		30		ms	

1. Please refer to the following document for further information on Self-Test: *MPU-6000/MPU-6050 Register Map and Descriptions*

	<b>MPU-6000/MPU-6050 Product Specification</b>	Document Number: PS-MPU-6000A-00 Revision: 3.4 Release Date: 08/19/2013
---	--	---

## 6.2 Accelerometer Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T<sub>A</sub> = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
<b>ACCELEROMETER SENSITIVITY</b>						
Full-Scale Range	AFS_SEL=0		±2		g	
	AFS_SEL=1		±4		g	
	AFS_SEL=2		±8		g	
	AFS_SEL=3		±16		g	
ADC Word Length	Output in two's complement format		16		bits	
Sensitivity Scale Factor	AFS_SEL=0		16,384		LSB/g	
	AFS_SEL=1		8,192		LSB/g	
	AFS_SEL=2		4,096		LSB/g	
	AFS_SEL=3		2,048		LSB/g	
Initial Calibration Tolerance			±3		%	
Sensitivity Change vs. Temperature	AFS_SEL=0, -40°C to +85°C		±0.02		%/°C	
Nonlinearity	Best Fit Straight Line		0.5		%	
Cross-Axis Sensitivity			±2		%	
<b>ZERO-G OUTPUT</b>						
Initial Calibration Tolerance	X and Y axes		±50		mg	1
	Z axis		±80		mg	
Zero-G Level Change vs. Temperature	X and Y axes, 0°C to +70°C		±35			
	Z axis, 0°C to +70°C		±60		mg	
<b>SELF TEST RESPONSE</b>						
Relative	Change from factory trim	-14		14	%	2
<b>NOISE PERFORMANCE</b>						
Power Spectral Density	@10Hz, AFS_SEL=0 & ODR=1kHz		400		μg/√Hz	
<b>LOW PASS FILTER RESPONSE</b>						
	Programmable Range	5		260	Hz	
<b>OUTPUT DATA RATE</b>						
	Programmable Range	4		1,000	Hz	
<b>INTELLIGENCE FUNCTION INCREMENT</b>			32		mg/LSB	

1. Typical zero-g initial calibration tolerance value after MSL3 preconditioning
2. Please refer to the following document for further information on Self-Test: *MPU-6000/MPU-6050 Register Map and Descriptions*