A PROJECT REPORT ON

GESTURE CONTROL ROBOT CAR USING ACCELEROMETER AND MICROCONTROLLER

Submitted By

KETHYREDDY NAVEEN KUMAR 21HQ1A4221

In partial fulfilment for the award of the degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

Under the Guidance

of

Mrs. Y. LAVANYA

(Assistant Professor)

AVANTHI'S RESEARCH & TECHNOLOGICAL ACADEMY

(AFFILIATED TO JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY GV,A.P)

BASAVAPALEM, BHOGAPURAM(MANDAL), VIZIANAGARAM (DISTRICT)

AVANTHI'S RESEARCH AND TECHNOLOGICAL ACADEMY

(BMASAVAPALEM, BIOGAPURAM MANDAL, VIZIANAGARAM DISTRICT)

BONAFIDE CERTIFICATE

Certified that this project report "**GESTURE CONTROL ROBOT CAR USING ACCELEROMETER AND MICROCONTROLLER**" is the Bonafide work **"K. Naveen Kumar (21HQ1A4221)"** who carried out project work under our supervision. The result embodied in this project have been verified and found satisfactory.

INTERNAL GUIDE	**HEAD OF DEPARTMENT**
Mrs. Y. LAVANYA	**Mrs. B. SAILAJA**
Asst. Professor	**Asst. professor**
Department of CSE	**Department of CSE**

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

I own a great many thanks to a people who helped and supported me during the writing of this book.

Our deepest thanks to Lecturer**, Mrs. Y.LAVANYA** the Guide of the project for guiding and correcting various documents of mine with attention and care. She has taken pain to go through the project and make necessary correction as and when needed.

We express our thanks **to Dr. P.GOVINDA RAO**, principal of **AVANTHI'S RESEARCH & TECHNOLOGICAL ACADEMY, BHOGAPURAM** for extending his support.

Our deep sense of gratitude **Mrs. B. SAILAJA (HOD),** support and guidance. Thanks and appreciation to the helpful people at, for their support.

We would also thank our Institution and our faculty members without whom this project would have been a distant reality.

Thanking you all

PROJECT ASSOCIATES

KETHYREDDY NAVEEN KUMAR 21HQ1A4221

DECLARATION

I am **“K. Naveen Kumar (21HQ1A4221)”** hereby declare that the project report entitled **"GESTURE CONTROL ROBOT CAR USING ACCELEROMETER AND MICROCONTROLLER"** is an original and authentic work done in the Department of **COMPUTER SCIENCE AND ENGINEERING (AI&ML),** submitted in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology The Gesture Robot in this project work has not been submitted earlier for award of any degree or diploma to the best of my knowledge.

PROJECT ASSOCIATES

KETHYREDDY NAVEEN KUMAR **21HQ1A4221**

ABSTRACT

The Gesture Control Robot Car Using Accelerometer and Microcontroller project explores the integration of gesture recognition technology with robotics, providing an innovative way to control a robot car. Traditionally, robotic vehicles are controlled using remote controls, joysticks, or predefined programs, but this project uses an Accelerometer (infrared sensor) to detect hand gestures and translate them into movement commands for the robot. The system is based on a microcontroller that processes input signals from the Accelerometer and controls the robot's motors accordingly. Through gesture recognition, the system provides a user-friendly interface that enhances the overall experience of operating the robot. The infrared sensor works by emitting light and detecting the reflection from a hand or object, allowing it to interpret gestures such as moving a hand up, down, left, or right. These gestures are then processed by a microcontroller, which translates them into specific movement commands for the robot, such as moving forward, backward, or turning. The robot car, equipped with motors, follows these commands to move in realtime. This gesture-controlled system offers an intuitive, hands-free way to interact with robots, making it more accessible and user-friendly. This innovative approach eliminates the need for physical controllers, offering a more natural and intuitive interface for controlling the robot.

LIST OF FIGURES

Figure no	Title	Page no
Figure 3.1	Architecture	11
Figure 3.2	Use Case Diagram	25
Figure 3.3	Class Diagram	27
Figure 3.4	Sequence Diagram	29
Figure 3.5	Activity Diagram	30

TABLE OF CONTENT

Page no

CHAPTER -1: INTRODUCTION

1.1 INTRODUCTION………………………………………….. 2-3
1.2 LITERATURE REVIEW…………………………………………… 3-4

CHAPTER -2: SYSTEM ANALYSIS

2.1 EXISTING SYSTEM………………………………………………… 6

2.2 PROPOSED SYSTEM……………………………………………….. 7-8

2.3 SYSTEM CONFIGURATION……………………………………….. 8-9

2.3.1 HARDWARE REQUIREMENTS………………………………. 8

2.3.2 SOFTWARE REQUIREMENTS……………………………….. 9

CHAPTER -3: SYSTEM DESIGN

3.1 ARCHITECTURE…………………………………………………. 11-20

3.1.1 CONECTIVITY OF THE CAR…………………………………20-21

3.1.2. CONECTIVITY OF THE HAND(GLOVE)……………………22-23

3.2 UML DIGRAMS……………………………………………………25-30

3.2.1 USE CASE DIGRAM…………………………………………….25-26

3.2.2 CLASS DIGRAM…………………………………………………26-27

3.2.3 SEQUENCE DIGRAM……………………………………………28-29

3.2.4 ACTIVITY DIGRAM…………………………………………..29-30

CHAPTER -4: IMPLEMENTATION

4.1 CODING FOR TRANSMITTER……………………………………..32-36
4.2 CODING FOR RECIVER……………………………………………..36-41

Page no

CHAPTER -5: OUTPUT

5.1 SCREEN SHOTS……………………………………………………….43-44

CHAPTER -6: TESTING

6.1 TEST CASE…………………………………………………………...46-48

6.2 FUNCTIONAL TESTCASE………………………………………..48-49

CHAPTER -7: CONCLUSION…………………………………….49-50

CHAPTER -8: FUTURE SCOPE…………………………………51-52

CHAPTER -9: REFERENCES……………………………………53-54

CHAPTER – 1
INTRODUCTION

1.1 INTRODUCTION

In recent years, the use of gesture control in technology has gained significant attention. Gesture control allows users to interact with machines simply by moving their hands or fingers, providing a more natural and intuitive way to control devices. This project focuses on creating a robot car that can be controlled through hand gestures using an Accelerometer and a microcontroller.

A gesture-controlled robot car enables the user to control the movement of the car without needing a physical remote control. Instead, the user can perform specific hand movements, such as waving or pointing, to make the car move forward, backward, or turn in different directions. This makes the interaction feel more organic and user-friendly, offering an innovative solution for controlling robots.

The core components of this project are the Accelerometer and the microcontroller. The Accelerometer is a device that can detect changes in light intensity, which makes it perfect for tracking hand gestures. When the user moves their hand in front of the Accelerometer, the light intensity changes, which the Accelerometer detects. The microcontroller then processes this information and translates it into commands that control the robot's movements, such as moving forward, backward, or turning.

The microcontroller, often an Arduino or similar board, acts as the brain of the system. It receives data from the Accelerometer and processes it based on pre-programmed instructions. For example, if the user waves their hand to the left, the microcontroller can command the robot to turn left. If the user moves their hand forward, the robot will move forward. This allows for precise control over the robot car using simple hand gestures.

This system eliminates the need for physical buttons or joysticks, which are typically used in traditional remote-controlled devices. Instead, the user can perform gestures in the air, making the control process faster and more interactive. Gesture control also opens up new possibilities for accessibility, making it easier for people with physical disabilities to interact with robots and other devices.

In addition to its intuitive nature, gesture control offers a fun and engaging experience for users. It can be used in various applications, such as entertainment, smart home systems, or educational robots. The robot car controlled by gestures can navigate different environments, perform tasks, and even follow the user's commands in real-time.

The integration of the Accelerometer and microcontroller is a simple yet effective way to create a system that can detect and respond to gestures. The Accelerometer's ability to detect light changes allows it to accurately track hand movements, while the microcontroller ensures that the robot responds appropriately.

In conclusion, a gesture-controlled robot car using an Accelerometer and microcontroller combines simplicity and innovation to create an easy-to-use, interactive system. It demonstrates how technology can be used to improve the way we interact with machines, offering a more natural and intuitive control experience. As gesture recognition technology continues to evolve, systems like this could become more common in everyday life, making robots and other devices easier and more enjoyable to control.

1.2 Literature Survey

The evolution of robotic control systems has seen significant advancements over the years, with traditional methods relying primarily on remote-controlled interfaces such as joysticks and wired or wireless controllers. These conventional control methods require constant manual input, which limits the user's ability to interact naturally with the robot.

Several studies and research efforts have focused on improving robotic mobility and control mechanisms. Some robots have been programmed to follow predefined paths or execute automated tasks, increasing efficiency but sacrificing adaptability. These systems lack the ability to respond dynamically to changes in the environment, reducing their usability in realtime applications.

Gesture-based control systems have emerged as a promising alternative, offering a more intuitive and immersive way to interact with robots. The integration of accelerometer sensors enables the detection of hand movements, which can be translated into commands for robot navigation. Previous research has demonstrated the feasibility of accelerometer-based gesture recognition in various applications, including virtual reality, human-computer interaction, and assistive technologies. Microcontrollers such as Arduino and Raspberry Pi play a crucial role in processing sensor inputs and executing motion commands effectively.

Wireless communication technologies such as RF (Radio Frequency), Bluetooth, and Zigbee have been extensively used in gesture-controlled robotic systems. RF modules (like 433 MHz transmitter-receiver pairs) are particularly favored for their simplicity and long range. In researchers implemented an RF communication system between the gesture detection module and the robotic car to enable seamless data transmission over moderate distances.

Studies have also highlighted the advantages of gesture-based control in overcoming limitations faced by individuals with restricted mobility. By eliminating the need for physical remote controls, gesture-controlled robot cars offer a more user-friendly experience. However, challenges remain, such as optimizing response time, minimizing signal interference, and ensuring the accuracy of gesture detection in diverse environmental conditions.

While gesture control provides an intuitive interface, it also presents challenges such as signal noise, misinterpretation of gestures, and limited gesture vocabulary. Researchers are continually working on improving gesture recognition algorithms and incorporating sensor fusion (e.g., combining gyroscopes and accelerometers) to enhance reliability and accuracy.

The system builds upon existing research by integrating an accelerometer-based gesture recognition system with a microcontroller-driven motor control mechanism. This approach enhances real-time adaptability, improves the user experience, and allows for greater freedom of movement in controlling robotic cars.

1.2 Problem Statement

Traditional robot car control methods, such as remote controllers and predefined programming, present significant limitations in terms of user interaction, mobility, and realtime adaptability. These conventional approaches often rely on manual operation or rigid, precoded instructions, which do not allow the system to respond dynamically to its environment or changing user needs. As a result, user engagement can be limited, and the robot’s functionality is often restricted to specific, predictable tasks.

Manual control through physical devices requires the user’s constant attention and input, making the operation less immersive and more cumbersome. For instance, joystick or button based controllers can hinder the natural flow of interaction, especially during extended use or in complex navigation scenarios. Furthermore, such systems may not be practical or inclusive for individuals with mobility challenges, reducing accessibility and usability.

Additionally, both wired and wireless traditional control mechanisms can become restrictive in real-world scenarios. Wired systems limit the range of motion and can lead to physical clutter or entanglement, while wireless systems, although more flexible, still depend on external devices that must be carried and managed by the user. These constraints can hinder the overall mobility and fluidity of the robot's operation in dynamic environments.

To overcome these challenges, this project proposes a Gesture Control Robot Car Using Accelerometer and Microcontroller. The design leverages accelerometer sensors to capture and interpret hand gestures, which are then processed by a microcontroller to command the robot car’s movements in real-time. This gesture-based control system aims to provide a more intuitive and seamless human-machine interaction experience, enhance accessibility for users of all abilities, and enable the robot car to respond instantly to user input without the need for conventional physical controllers.

CHAPTER – 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

Traditional remote-controlled robot cars rely on physical remote controls or joysticks, necessitating constant manual input from users. This approach restricts the range of motion and interaction, as users must concentrate on operating the controller rather than enjoying a more immersive and intuitive control experience. In contrast, some robots are pre-programmed to follow specific paths or perform designated tasks automatically. While this method is efficient, it lacks real-time adaptability and requires a fixed environment, which diminishes the robots' versatility. Additionally, many robot cars utilize wired or wireless controllers, which can limit the user's freedom of movement and complicate the control experience, particularly for individuals with limited mobility or those seeking a more natural interaction with the robot.

Traditional robot cars are controlled using physical remote controls or joysticks, requiring constant manual input. These devices limit the range of motion and interaction, and users must focus on operating the controller rather than enjoying a more immersive or intuitive control experience.

Some robots are pre-programmed to follow specific paths or perform certain tasks automatically. While efficient, these robots lack real-time adaptability and require a fixed environment, which reduces their versatility.

Many robot cars use wired or wireless controllers for operation. These controllers limit the user’s freedom of movement and may complicate the control experience for users with limited mobility or those who want to interact more naturally with the robot.

2.1.1 Disadvantages

i. **Gesture Recognition Accuracy**: The system relies on detecting hand movements accurately. Factors like shaky hands, incorrect gestures, or sensor sensitivity issues may lead to incorrect robot movements.

ii. **Limited Range** : The accelerometer-based control may have a restricted range, meaning the robot might not respond well if the user is too far away. Wireless communication technology used in the system may also have signal limitations.

iii. **Learning Curve for Users** : Users need to learn and get comfortable with the specific gestures required to control the robot car. Unlike simple remote controls, gesture-based control may take some practice.

iv. **Complexity in Setup and Maintenance** : Unlike a simple remote-controlled car, this system involves integrating various components like an accelerometer, microcontroller, and motor driver. Any hardware or software issues may require troubleshooting, making it slightly more complex to maintain.

2.2 PROPOSED SYSTEM

The Gesture Control Robot Car is designed to improve the way we control robot cars by replacing traditional remote controls with gesture-based commands. This system uses an accelerometer sensor to detect different hand movements, such as waving, pointing, or swiping in different directions. The accelerometer works like an infrared sensor, identifying specific hand gestures like moving left, right, up, or down. These gestures are then sent to a microcontroller, which acts as the "brain" of the robot car.

The microcontroller processes the signals from the accelerometer in real-time and converts them into movement commands. For example, if the user swipes forward, the robot car moves forward. A swipe to the left or right makes the car turn in that direction, and raising a hand stops the car. The microcontroller then sends these commands to the motor driver, which controls the wheels and movement of the robot.

To build this system, several key components are required. The microcontroller (such as Arduino Uno or Raspberry Pi) is essential for processing gestures and controlling the motors. The accelerometer sensor detects hand movements with high precision and must respond quickly (within 50 milliseconds) to ensure real-time control.

The robot car moves using DC motors (for driving) and servo motors (for precise steering). DC motors operate between 6V and 12V and provide enough speed and torque for smooth movement. Servo motors help with steering by adjusting angles accurately. Other important components include wires and connectors for secure connections, a breadboard for assembling circuits, and LEDs for status indications.

By carefully selecting and integrating these components, we can build an efficient and responsive gesture-controlled robot car. This system allows users to control the car easily, wirelessly, and in real time, making it more fun, accessible, and futuristic compared to traditional remote-controlled robots.

2.2.1 Advantages

i. **Educational and Practical Applications** : The project is an excellent learning tool for students and hobbyists interested in robotics, microcontrollers, and gesture-based technology. It demonstrates how sensors and programming can work together in real-world applications.

ii. **Hands-Free and Intuitive Control** : Instead of using a remote control or joystick, users can operate the robot car with simple hand gestures. This makes the experience more natural and user-friendly.

iii. **Improved Accessibility** : The system benefits individuals with limited mobility who may find traditional controllers difficult to use. Gesture control provides an easier and more inclusive way to interact with the robot.

iv. **Enhanced User Experience** : Gesture-based control makes operating the robot more interactive and engaging. Users can enjoy a more immersive and futuristic way of controlling the car.

v. **Real-Time Response** : The accelerometer sensor detects gestures quickly, with a response time of less than 50 milliseconds. This ensures smooth and immediate control of the robot's movement

2.3 SYSTEM CONFIGURATION

For the **gesture-controlled robot car** using an Accelerometer and microcontroller, there are specific hardware and software components required to ensure the system works efficiently and effectively. Below is a breakdown of the **hardware** and **software** requirements for this project.

2.3.1 Hardware Requirements

1. **Robot Car Chassis :** A simple robot car chassis, which typically includes wheels, motors, and a frame to hold the components together. Included Components are DC motors (2 or 4, depending on the chassis), Wheels, Motor driver (such as L298N or L293D), Battery (Li-ion, Li-Po, or AA batteries).

2. **Microcontroller :** The microcontroller is the brain of the system, responsible for processing the inputs from the Accelerometer and controlling the motors. Recommended Microcontroller are Arduino (e.g., Arduino Uno, Nano, or Mega) and it's function is Receives the signals from the Accelerometer, processes them, and controls the motor driver to move the robot car.

3. **Accelerometer (Light Sensor) :** The Accelerometer detects changes in light intensity caused by hand gestures . Recommended Component are An optical sensor like an LDR (Light Dependent Resistor) or an IR sensor to detect hand movements based on changes in light levels. And it's function is Detects the gestures made by the user and sends corresponding signals to the microcontroller.

4. **Motor Driver** : A motor driver is used to control the movement of the DC motors by receiving signals from the microcontroller. Recommended Components are L298N or L293D motor driver IC. And it's function is Enables forward, backward, and turning movements by controlling the direction and speed of the motors.

5. **Power Supply :** A reliable power source is needed to power the robot, the microcontroller, and the sensor. Recommended Power Sources are Rechargeable battery pack (e.g., 7.4V Li-ion battery),Power regulator or step-down converter (if needed) to supply proper voltage levels to the components.

6. **Jumper Wires and Breadboard** : To connect all the components to the microcontroller and motor driver. And it's function is Facilitates prototyping and easy connections between the components.

7. **Chassis Assembly and Mounting Materials :** Materials to mount the components onto the robot car, such as screws, nuts, and standoffs. And function is Holds all the parts in place and ensures the robot car is stable.

2.3.2 Software Requirements

1. **Arduino IDE** : The integrated development environment (IDE) used to write and upload code to the Arduino microcontroller. And it's version is Latest stable version of Arduino IDE (available for Windows, macOS, and Linux). Function is Provides the necessary tools for writing the program (sketch), compiling it, and uploading it to the microcontroller.

2. **Arduino Libraries** : Needed libraries are Motor Driver Library (if using a library to control the motor driver): For example, L298N or L293D libraries. Sensor Libraries (if using advanced sensors) is for handling sensor inputs may be required (e.g., Adafruit Sensor for some specific sensors). Servo Motor Library (if controlling a servo for steering) If your robot has servo motors to control the direction of movement, you may need to use the Servo.h library. And it's function is Libraries provide easy-to-use functions to control hardware components like motors, sensors, and LEDs, making coding more efficient.

2.4 System Configuration Overview

1. **Microcontroller (Arduino) :**

   Receives data from the **Accelerometer** (light sensor) to detect hand gestures and sends control signals to the **motor driver** to move the robot car.

2. **Accelerometer (Light Sensor) :**

   Detects light changes based on hand movements (gestures) and sends signals to the Arduino for gesture recognition.

3. **Motor Driver (L298N/L293D) :**

   Receives signals from the microcontroller to control the robot's motors and Controls the movement (forward, backward, left, right).

4. **Robot Car Motors :**

   Moves the robot car based on the signals sent by the motor driver.

The system configuration involves a combination of hardware components such as a robot car chassis, microcontroller, Accelerometer, motor driver, and power supply, along with software components like the Arduino and code libraries. Together, they work to enable gesture-based control of the robot car, offering a user-friendly and innovative approach to controlling robotic systems.

CHPATER – 3

SYSTEM DESIGN

3.1 ARCHITECTURE

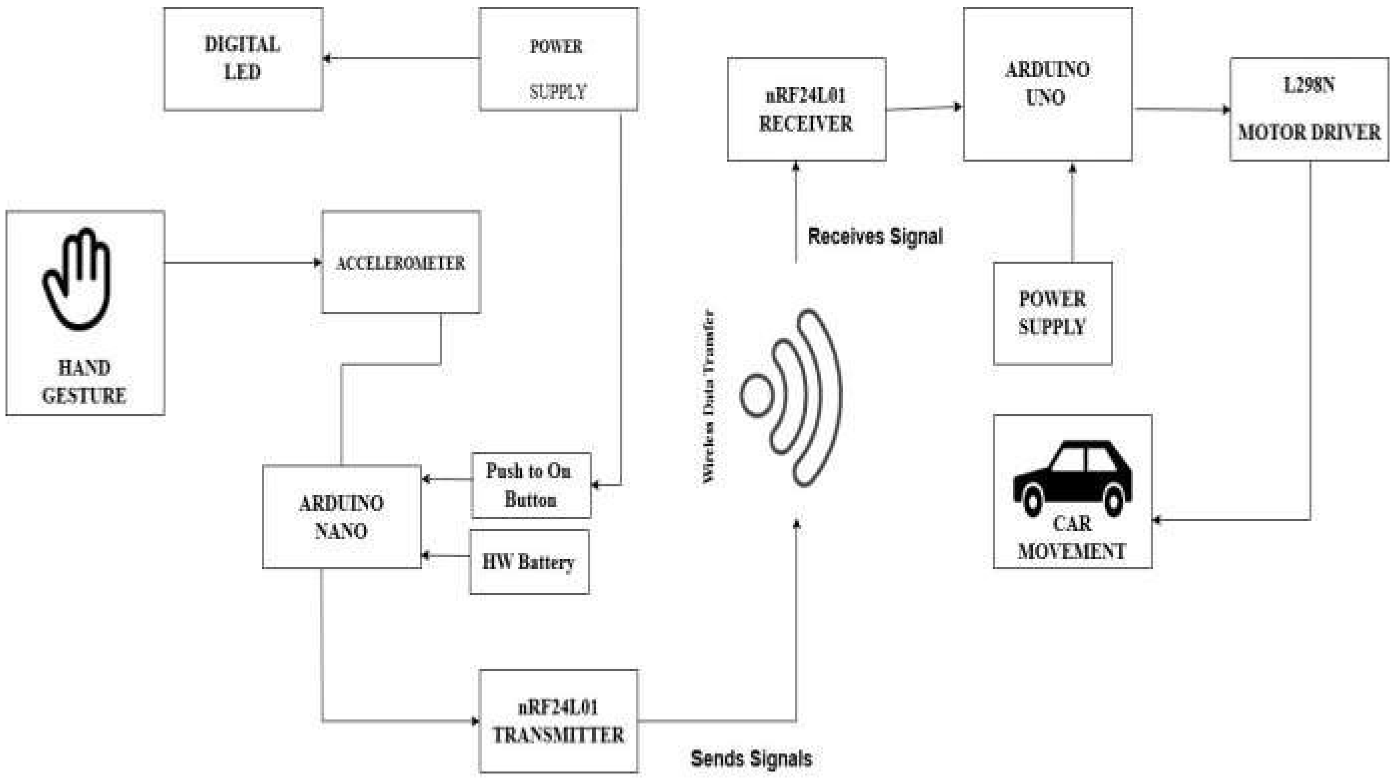


Fig (3.1) : Architecture

The Gesture Control Robot Car is designed to operate using hand gestures, making it more interactive and user-friendly. The system consists of two main parts: the controller (gesture recognition) and the robot (movement control). The user controls the robot by moving their hand, and an accelerometer detects these movements. The accelerometer sends the data to an Arduino Nano, which processes the signals and identifies the gesture based on trained data. Once recognized, the Arduino Nano transmits the command wirelessly using an nRF24L01 Transmitter. A digital LED is also included to indicate power status, and the entire system is powered by a Lithium-ion battery, which is activated using a push button.

On the robot side, the nRF24L01 Receiver picks up the transmitted signals and sends them to an Arduino Uno R3, which acts as the brain of the robot. The Arduino Uno interprets the received data and sends appropriate signals to an L298N motor driver, which controls the DC motors attached to the wheels of the robot. Based on the user's hand movement, the motors move forward, backward, left, or right. The entire system is powered by a separate power supply to ensure smooth operation.

when the user moves their hand, the accelerometer detects the motion and sends data to the Arduino Nano. The Arduino Nano processes the data and wirelessly transmits it to the robot using the nRF24L01 module. The robot's Arduino Uno receives this data, processes it, and controls the motors accordingly, allowing the robot car to move as per the user's gestures. This system provides a real-time, wireless, and hands-free way to control the robot, making it an innovative alternative to traditional remote-controlled cars.

HARDWARE DESCRIPTION

Arduino uno : The Arduino Uno is a microcontroller board that is principally based on the ATmega328 microcontroller series and has an integrated development environment (IDE) to write, compile, and upload the programming

Fig : Arduino Microcontroller

codes to the microcontroller. Various sensors will forward the environmental data as an input to the microcontroller, which will correspondingly send the attached peripherals, e.g., actuators. It has a total number of 28 pins; 14 digital input/output pins (six are pulse width modulation (PWM) pins) and six pins are the analogs used for interaction with the electronic components like sensors, motors; 3 GND pins (for grounding) and remaining pins for 3.3 V, 5 V, VIN, RESET and AREF (Analogue Reference). Arduino contains a microcontroller with a 32 KB storage memory, 2 KB of SRAM (Static random-access memory), and 1 KB of EEPROM (Electrically Erasable Programmable Read-Only Memory).

Arduino primarily supports a C programming language compiler, macro-assemblers, and evaluation kits. It also has 16 MHz ceramic resonators, a USB connection jack for connecting with a computer, a jack for external power supply, an ICSP (in-circuit serial programmer) header, and a reset button to reset to the factory settings. It operates with an input voltage of 7 to 12 V (limit up to 20 V).

Arduino Nano pin's : The Arduino Uno is one of the most popular boards in the Arduino family. It's based on the ATmega328P microcontroller and provides a range of pins for generalpurpose input/output (I/O), communication, power, and more.

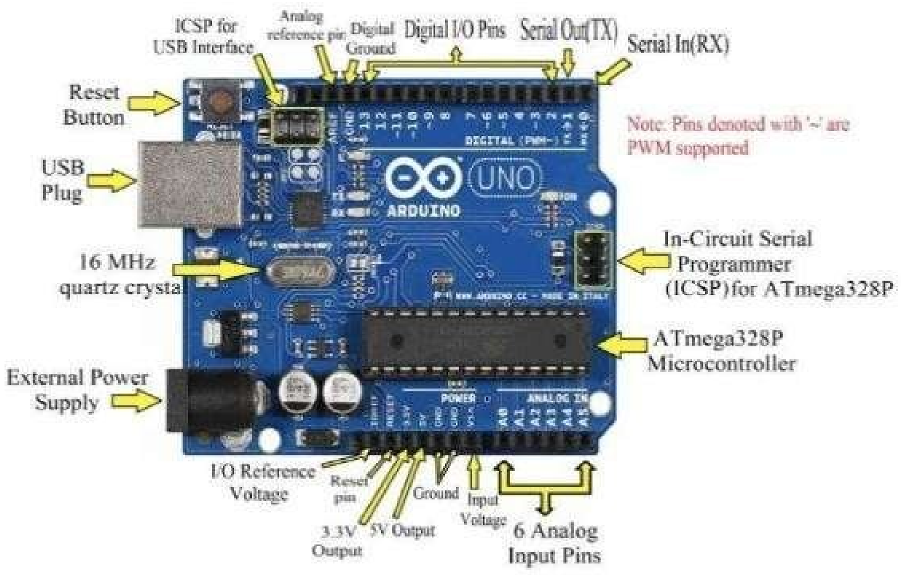


Fig : Nano pin's

1. Digital Pins (0-13) :

These are used for general digital input and output. Each of these pins can be set HIGH (5V) or LOW (0V), and they can also be configured for special functions such as PWM, interrupts, or serial communication.

Pin 0 (RX): Serial Reception (RX) for UART communication. Can also be used as a digital I/O pin.

Pin 1 (TX): Serial Transmission (TX) for UART communication. Can also be used as a digital I/O pin.

Pin 2-13: These are general-purpose digital I/O pins, and can be used for input or output. Some of these pins have special functions:

Pin 3-11: Can output PWM (Pulse Width Modulation) signals.

Pin 12: SPI MISO (Master In Slave Out).

Pin 13: Typically connected to the onboard LED and can also be used for general-purpose digital I/O.

2. Analog Pins (A0-A5)

These pins can read analog signals (0-5V) and convert them to a digital value (0-1023) using the analogRead() function. They can also be used as digital pins if needed.

A0-A5: These are standard analog input pins and can also be used as digital I/O pins (with digitalRead()/digitalWrite()).

3. Power Pins

These pins are used to supply power to the board or external components in your circuit. 5V: Provides 5V power to the board and can be used to power external components. 3.3V: Provides

3.3V power to external components. (Note: limited current, typically 50mA). GND: Ground pins, used to complete the circuit. Vin: This pin is used to supply external voltage (6-12V, typically 7-12V) to the board. It will be regulated down to 5V by the onboard voltage regulator.

4. Reset Pin

RESET: This pin is used to manually reset the Arduino Uno by pulling the pin LOW. You can use a button to connect this pin to ground to reset the board.

5. Other Pins and Functions

AREF: This pin is used to provide an external reference voltage to the analog inputs for more accurate readings. It is usually connected to 5V but can be supplied with a different voltage to adjust the reference.

IOREF: This pin provides the reference voltage (5V or 3.3V) for the board, depending on the operating voltage. It's useful when designing shields to ensure compatibilitywith different Arduino boards.

6. Communication Pins

These pins are used for serial, I2C, and SPI communication.

Pin 0 (RX) & Pin 1 (TX): Used for Serial Communication (UART). Pin 0 is RX (receive) and Pin 1 is TX (transmit). These are connected to the USB-to-serial converter on the board for uploading code and serial communication.

Pin 2, 3: These pins can be used for External Interrupts (INT0, INT1) and can be triggered by changes in voltage, useful for handling external events.

Pin 4: Can be used for I2C communication (SDA).

Pin 5: Can be used as SPI (MOSI) for communication with other devices.

Pin 6-13: These pins are used for SPI and PWM (Pulse Width Modulation) outputs.

7. Special Features:

1. PWM (Pulse Width Modulation): Certain digital pins can generate PWM signals to simulate analog voltage levels using the analogWrite() function. These pins are 3, 5, 6, 9, 10, 11
2. SPI (Serial Peripheral Interface): SPI communication is useful for high-speed data transfer between the Arduino and other peripherals like sensors, displays, or memory cards. The following pins are used for SPI communication, MOSI (Master Out Slave In) = Pin 11, MISO (Master In Slave Out) = Pin 12, SCK (Serial Clock) = Pin 13, SS (Slave Select) = Pin 10.
3. I2C (Inter-Integrated Circuit): The Arduino Uno uses the following pins for I2C communication, SDA (Data) = Pin A4, SCL (Clock) = Pin A5.
4. External Interrupts: These pins can be used to trigger interrupts, which allow the Arduino to respond to changes in external signals quickly, Pin 2 (INT0) and Pin 3 (INT1) can handle external interrupts.

Arduino Nano

Arduino is comprised of a microcontroller of ATmega328P with 32 KB memory, of which 2 KB is used for the bootloader, 2 KB of Static random-access memory (SRAM), 16 MHZ Clock Speed, and 1 KB of electrically erasable programmable read only memory (EEPROM).

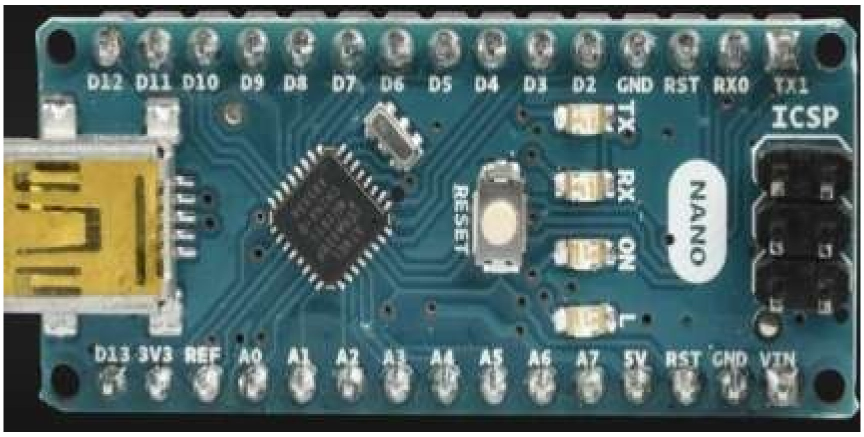


Fig : Arduino Nano

ATmega328P supports C compiler, evaluation kits, and macro-assemblers. It has a total 30 pins of which 14 are digital input and output pins (six are pulse width modulation (PWM)), 8 Analog IN pins, 2 GND pins, 2 RESET and remaining pins for VIN, 5V, 3V and REF respectively. It has a USB connection jack, an external power supply jack, an ICSP header and a reset button. It operates with an input voltage of 7 to 12 V (limit up to 20 V).

Arduino Nano pin's

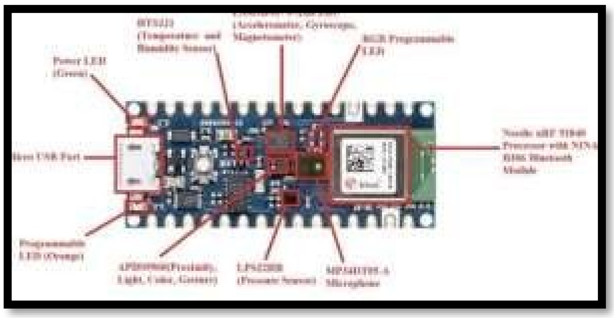


Fig : Nano pin's

The Arduino Nano is a compact, breadboard-friendly board based on the ATmega328P microcontroller. It has 22 pins in total, which can be used for various digital, analog, power, and communication functions.

A detailed explanation of each pin:

1. Digital Pins (0-13)

These pins are used for general digital input/output. Each pin can be set to HIGH or LOW, meaning on or off (typically 5V or 0V). Some of these pins have additional special functions (like PWM or serial communication).

Pin 0 (RX): Used for Serial Reception (RX) in UART communication. Can also be used as a digital I/O pin.

Pin 1 (TX): Used for Serial Transmission (TX) in UART communication. Can also be used as a digital I/O pin.

Pin 2-13: These are standard digital pins and can be used for input or output. Some also have additional functions:

Pin (3-11) : Can output PWM (Pulse Width Modulation) signals (via analogWrite()).

Pin 12: Can be used for SPI communication (MISO). Pin

13: Often connected to an onboard LED.

2. Analog Pins (A0-A7)

These pins are used to read analog signals, with a 10-bit resolution (0-1023 range). They can also be used as general digital I/O pins (if needed).

Pin A0-A5: These pins are standard analog inputs (0-5V) and are used with functions like analogRead().

Pin A6-A7: These are also analog pins, but not available for digital input/output on the Nano. Pin A0-A5 can also be used as digital I/O pins if necessary.

3. Power Pins

These pins provide power to the Arduino Nano or can be used to supply power to other components in your circuit.

VCC (5V): This pin provides 5V power from the board.

GND: These are ground pins, used to complete the electrical circuit.

RAW: This is the input voltage pin. It can accept 6-12V (typically 7-12V) to power the Arduino board through its voltage regulator.

4. Reset Pin

RESET: This pin can be used to manually reset the Arduino Nano. It can be connected to a button to restart the program, or used to reset the board when needed.

5. Other Pins and Functions

AREF: This pin is used to provide a reference voltage for the analog inputs. It'stypically used for precise analog readings.

IOREF: This pin provides the reference voltage (5V) for the board to operate at, which is useful for ensuring compatibility with shields.

6. Communication Pins

These pins are used for serial, I2C, and SPI communication.

Pin 0 (RX) & Pin 1 (TX): Used for Serial Communication (UART). These pins are connected to the USB-to-serial converter, so using them can interfere with uploading code to the board.

Pin 2, 3: These can be used for External Interrupts (INT0, INT1) to detect changes in state on external pins (useful for timing-sensitive tasks).

Pin 4: This can be used for I2C communication (SDA), though the standard I2C pins are usually A4 and A5 on the Nano.

Pin 5: Can also be used as SPI (MOSI) for communication.

Pin 6, 7, 8, 9: These are also typically used in communication protocols like SPI (MISO, SCK, etc.).

Special Features:

1. PWM (Pulse Width Modulation): The following digital pins can output PWM signals to simulate analog voltage levels (via analogWrite()) these Pins are, 3, 5, 6, 9, 10, 11
2. SPI (Serial Peripheral Interface): This communication protocol is typically used to connect devices like sensors, displays, etc. The default SPI pins on the Nano are, MOSI (Master Out Slave In): Pin 11, MISO (Master In Slave Out): Pin 12, SCK (Serial Clock): Pin 13, SS (Slave Select): Pin 10.
3. I2C: The default I2C pins for communication are, SDA (Data Line): Pin A4, SCL (Clock Line): Pin A5.
4. External Interrupts: Some pins are capable of handling external interrupts. The pins capable of interrupts are, Pin 2 (INT0), Pin 3 (INT1)

Accelerometer

An Accelerometer is an instrument used by optometrists or ophthalmologists in clinics to assess a patient’s vision. They determine both the range and power of a person’s eye through an Accelerometer. An Accelerometer checks the eyeball for gathered light. The emmetropic eye state is taken as a standard to measure the divergence degree between the emmetropic and examined eye. Thus, measuring the refractive power of an eye is one of the primary uses of an Accelerometer.

Optical Principle of Accelerometer

An Accelerometer works on the basis of automation of refraction. The primary principles of Accelerometer

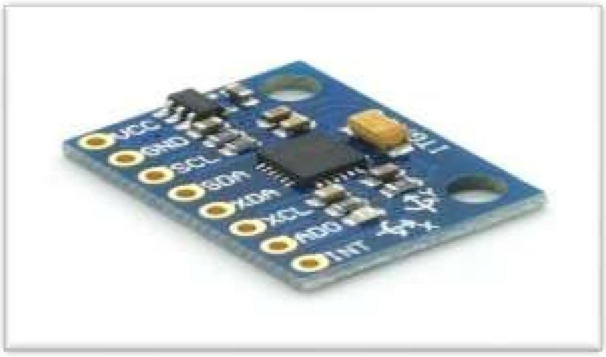


Fig : Accelerometer

RF Transmitter and Receiver (nRF24L01)

RF modules are 433 MHz RF transmitter and receiver modules that are used to transmit and receive the infrared waves. RF transmitter consists of three pins; ATAD (Signal pin used to send data for the receiver), VCC (for providing voltage) and Ground pin (for grounding the module). Its working voltage is 3–12 V and consumes 12 V power. The RF transmitter can transmit up to 90 m in an open area. The RF Receiver module consists of four pins; VCC (for providing Voltage), two DATA pins for receiving data from transmitter and Ground (for grounding the module). Its working voltage is 5 VDC.

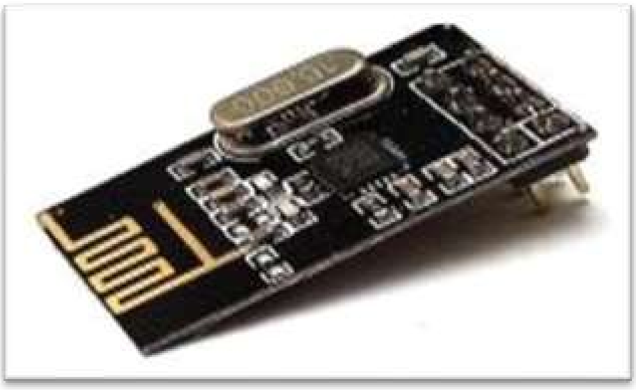


Fig : RF Transmitter and Receiver

L298N motor driver

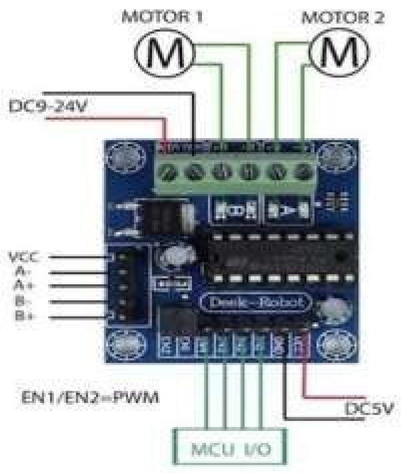


Fig : L298N for 2 Motor

The L298N is quadruple high-current half-H drivers that are used to control the direction and speed of up to four direct current (DC) motors with individual 8-bit speed selection simultaneously up to 0.6 A each. It is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 to 36 V. It has eight output pins for connecting four DC motors or two Stepper motors, one reset button, six pins for connecting two servo motors, one +M pin for providing external motor power, five Analog input pins, thirteen digital input pins for connecting with Arduino and ground pins for grounding.

DC Motor

DC Motors The basic principle of DC motor is it converts electrical signals into mechanical energy. A DC motor (Direct Current motor) is a type of electrical machine that converts electrical energy (from DC sources like batteries) into mechanical energy (motion). It operates on the principle that a current-carrying conductor, placed in a magnetic field, experiences a force (Lorentz force) that causes the conductor to move. In simple terms, a DC motor uses direct current electricity to generate rotational motion, making it a versatile choice for applications like robotics, electric vehicles, and many household appliances.

Structure of a DC Motor:

A typical DC motor consists of several key components, each playing a crucial role in its operation:

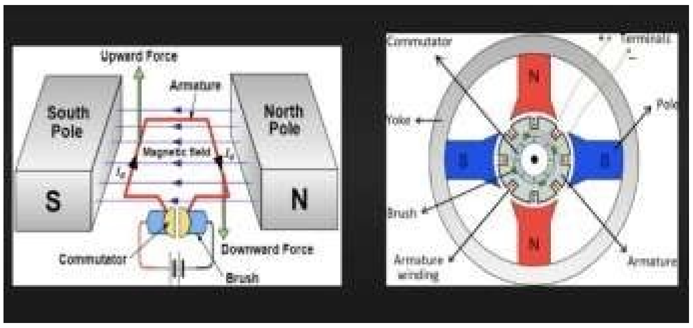


Fig : DC Motor

Control Flow for DC Motor Movement in the Robot Car

In the gesture-controlled robot, the NodeMCU processes accelerometer data to control motor movement:

1. **Forward Gesture**: Tilt hand forward. o IN1 = HIGH, IN2 = LOW; IN3 = HIGH, IN4 = LOW; PWM for speed.
2. **Backward Gesture**: Tilt hand backward.
   - o IN1 = LOW, IN2 = HIGH; IN3 = LOW, IN4 = HIGH; PWM for speed.
3. **Left Gesture**: Tilt hand left.
   - o IN1 = LOW, IN2 = HIGH; IN3 = HIGH, IN4 = LOW; PWM for speed. 4.

   Right Gesture: Tilt hand right.
   - o IN1 = HIGH, IN2 = LOW; IN3 = LOW, IN4 = HIGH; PWM for speed.

5. **Stop Gesture**: .
   - o IN1 = LOW, IN2 = LOW; IN3 = LOW, IN4 = LOW; PWM = 0% (motor off).

APPLICATIONS

- o Wireless controlled robots are very useful in many applications like remote surveillance, military etc.
- o Hand gesture controlled robot can be used by physically challenged in wheelchairs.
- o Hand gesture controlled industrial grade robotic arms can be developed.

2.1.1 CONECTIVITY OF THE CAR :

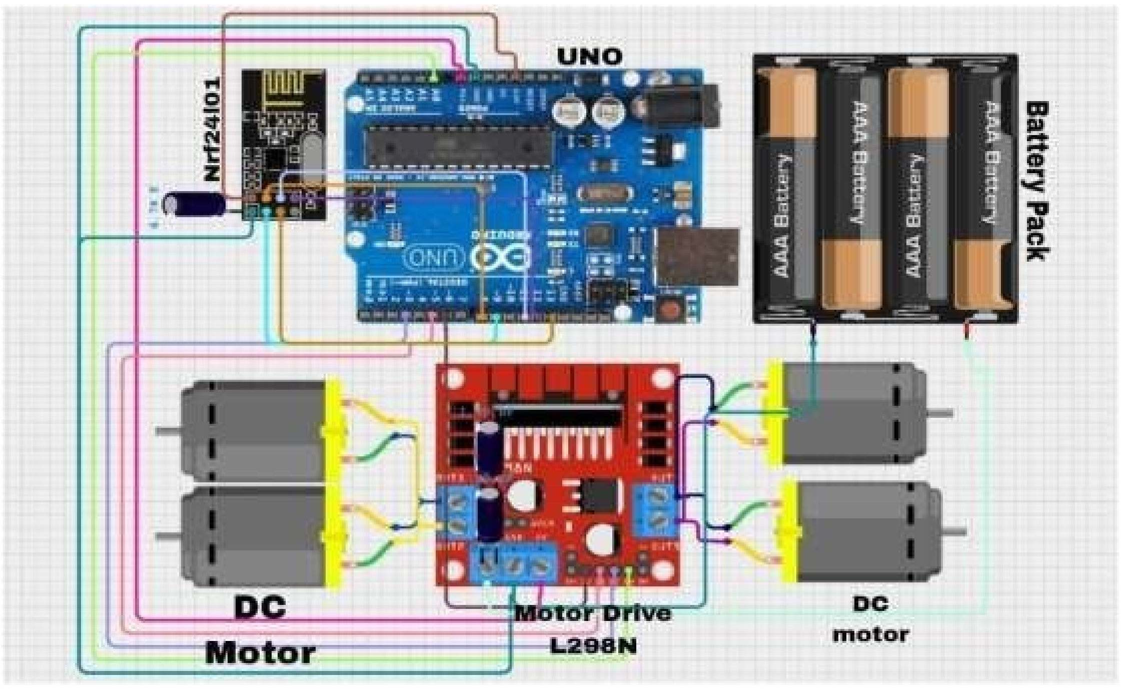


Fig : Prototype of Receiver

Components

1. Arduino UNO (Center-Top)
    - Main microcontroller that controls the entire setup.

    - Connected to the motor driver and ESP8266 for logic and control.
2. L298N Motor Driver (Center-Bottom)
    - Used to control the DC motors.

    - Receives control signals from the Arduino and drives the motors using an external power source.
3. ESP8266 Wi-Fi Module (Left of Arduino)

- Wireless communication module.
- Allows the Arduino to communicate over Wi-Fi (e.g., for IoT projects or mobile app control).

4. Capacitor (Near ESP8266)

• Likely a 10μF or 100μF electrolytic capacitor, used for power stabilization for ESP8266 (which can be power-sensitive).

5. 4x AAA Battery Pack (Top-Right)

• Provides external power supply for the motors through the L298N motor driver.

6. 4x DC Gear Motors (Bottom Corners)

- These are the actuators that receive power from the motor driver to create movement.

Connections:

Arduino to L298N Motor Driver

- Pins 9, 10, 11, 12 (digital): Connected to IN1, IN2, IN3, IN4 on the L298N — these control the direction of each motor.
- GND of Arduino connected to GND of L298N — common ground reference.

L298N to Motors

- OUT1 & OUT2: Connected to one motor (right side).
- OUT3 & OUT4: Connected to the other motor (left side).
- 12V and GND pins: Connected to the battery pack to power motors.

Arduino to ESP8266

- TX of ESP8266 to RX (Pin 0) of Arduino.
- RX of ESP8266 to TX (Pin 1) of Arduino (often with a voltage divider or level shifter, though not shown here — ESP8266 is 3.3V tolerant).
- VCC & GND of ESP8266 connected to 3.3V and GND of Arduino (power supply).
- The capacitor across VCC and GND helps stabilize ESP power.

Power Connections

- AAA Battery Pack powers the L298N motor driver directly.
- Arduino is powered via USB (likely for programming and serial monitoring).

2.1.2 CONECTIVITY OF THE HAND(GLOVE):

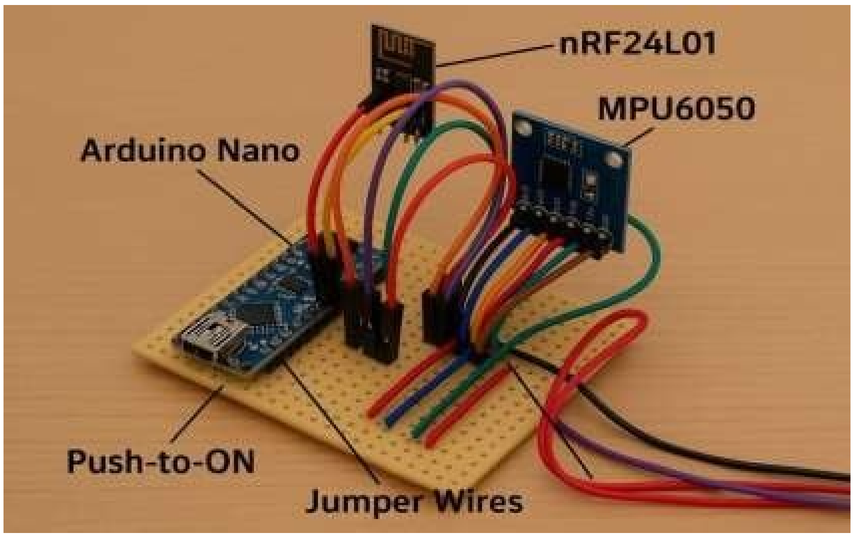


Fig : Prototype of Transmitter

1. Arduino Nano

- Acts as the microcontroller — processes data and controls connected components.
- Uploads code through USB Mini-B.
- Replace Uno in the setup and connect respective pins based on the Nano pinout.

2. hRF24L01 (2.4GHz Wireless Module)

- Function: Wireless communication; used for data transmission between devices (e.g., remote sensors).
- Pins:

  VCC → 3.3V on Nano

  GND → GND

  CSN → D10

  SCK → D13 MOSI → D11

  MISO → D12

3. MPU6050 (Accelerometer + Gyroscope)

- Function: Senses motion, acceleration, and rotation.
- Pins:

VCC → 5V on Nano

GND → GND

SDA → A4

SCL → A5

Communicates via I2C protocol.

4. Push-to-ON Button

- Function: User input for triggering events (e.g., reset, start action).
- One side connected to GND, the other to a digital input pin (e.g., D2), with a pull-up resistor (can use INPUT_PULLUP in code).

5. Jumper Wires

- Connect components to Arduino.
- Color-coded for clarity (e.g., red for VCC, black for GND, etc.).

WORKING

Gesture controlled robot moves according to the user's hand movement recognized by the device in our hand. When we tilt hand in front side, the robot starts to moving forward and continues moving forward until the next command is given. When we tilt hand in the backside, the robot changes its state and start moving in the backwards direction until another command is given. When we tilt it towards the left side, it will turn left till next command. When we tilt our hand in right side robot is turned to the right

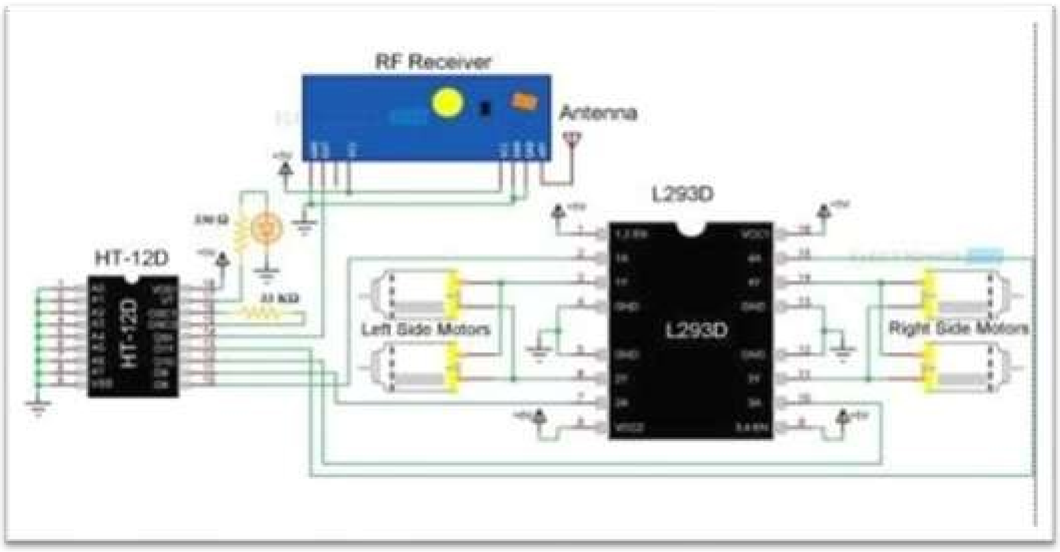


Fig : Transmitter

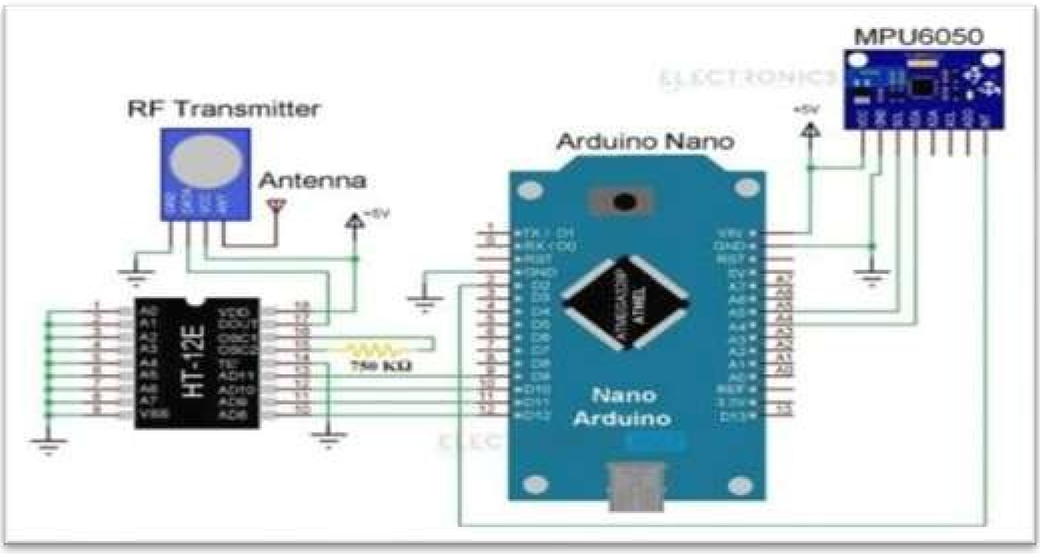
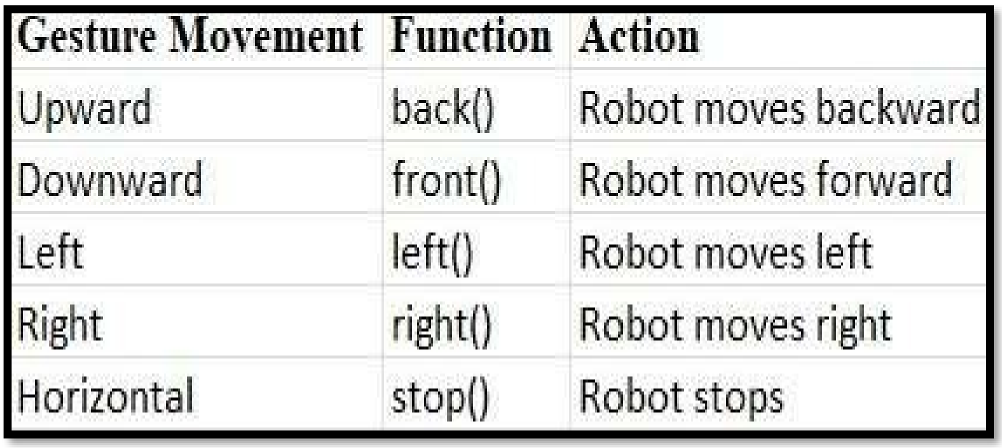


Fig : Receiver

Circuit for this hand gesture-controlled robot is quite simple. An RF pair is used for communication and connection to the Arduino. The motor driver is connected with the Arduino to operate the robot. Motor driver's input pins 2, 7, 10 and 15 are connected to Arduino digital PIN 6, 5, 4 and 3. Here we are using used two DC motors to drive the robot in which one motor is connected at the output of motor driver 3 and 6, and another motor is connected at 11 and 14. A 9-volt battery is also used to power the motor driver for driving motors.



Gesture Movement	Function	Action
Upward	back()	Robot moves backward
Downward	front()	Robot moves forward
Left	left()	Robot moves left
Right	right()	Robot moves right
Horizontal	stop()	Robot stops

Fig : Conditions for movement

3.2 UML DIGRAMS

UML (Unified Modeling Language) diagrams are visual tools used in software engineering to represent and design systems. They provide a standard way to model the structure, behavior, and interactions within a system. UML diagrams can be divided into two main categories: **structural diagrams** (e.g., class diagrams, component diagrams) and **behavioral diagrams** (e.g., use case diagrams, sequence diagrams).

3.2.1 USE CASE DIGRAM

The use case diagram you provided represents a gesture-controlled robot car system. It visually describes how a user interacts with a system to control a robot car using hand gestures. The main components and their interactions are shown in a step-by-step flow, which helps in understanding the complete working of the system.

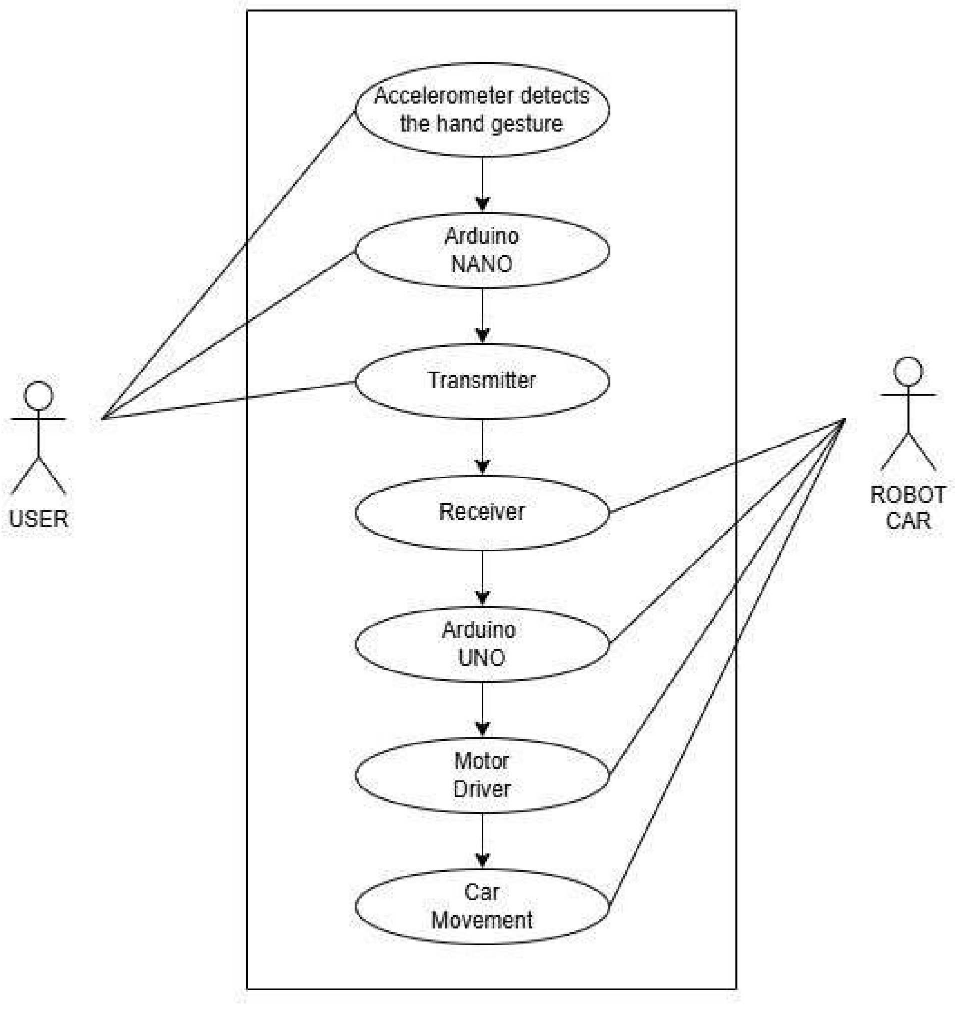


Fig 3.2 : USE CASE DIGRAM

This use case diagram illustrates how a gesture-controlled robot car works using a combination of sensors, microcontrollers, and communication modules. The system involves two main actors: the User and the Robot Car. The user controls the car using hand gestures, while the robot car reacts based on the transmitted signals. The entire setup operates in a sequential flow from detecting gestures to moving the car.

The process begins when the accelerometer detects the hand gestures made by the user. The accelerometer senses the tilt or direction of the user’s hand and converts this motion into corresponding electrical signals. These signals represent specific commands like forward, backward, left, or right. This is the first and crucial step, as the accuracy of gesture detection determines the reliability of the car’s movements.

User: The person interacting with the system, initiating commands through hand gestures.

Accelerometer: This sensor detects the user's hand gestures and translates them into signals.

Arduino NANO: A small microcontroller that processes the signals from the accelerometer.

Transmitter: Sends the processed signals wirelessly to the robot car.

Receiver: Located on the robot car, it receives the signals transmitted by the user.

Arduino UNO: Another microcontroller on the robot car that interprets the received signals.

Motor Driver: Controls the motors of the robot car based on the commands from the Arduino UNO.

Car Movement: The robot car executes the actions dictated by the user's hand gestures.

3.2.2 CLASS DIAGRAM

A class diagram is a type of structural UML diagram that illustrates the static structure of a system by showing its classes, their attributes, methods, and the relationships between them. It defines the blueprint for objects in a system, depicting how they are organized and interact.

The class diagram represents the structure and functionality of the Gesture Control Robot Car System, showing how different components interact to control the robot car using hand gestures.

The Gesture Control Robot Car System class is the main system that manages the entire process. It takes input from gestures and communicates with different components to interpret gestures, send commands to the motor, and control the car’s movement. It interacts with the Accelerometer Sensor, which detects hand movements and generates corresponding signals. These signals are then sent to the Microcontroller, which processes the received data, interprets the gestures, and sends appropriate commands to the MotorDriver.

The MotorDriver class is responsible for executing commands received from the microcontroller and controlling the RobotCar. The RobotCar class defines the movement functions such as moveForward(), turnLeft(), turnRight(), and stop(), ensuring that the car responds correctly to the user's gestures.

when a user moves their hand, the accelerometer detects the movement and sends a signal to the microcontroller. The microcontroller processes the gesture and sends a command to the motor driver, which in turn controls the robot car's movement accordingly. This structured interaction ensures smooth and accurate real-time gesture-based control of the robot car.

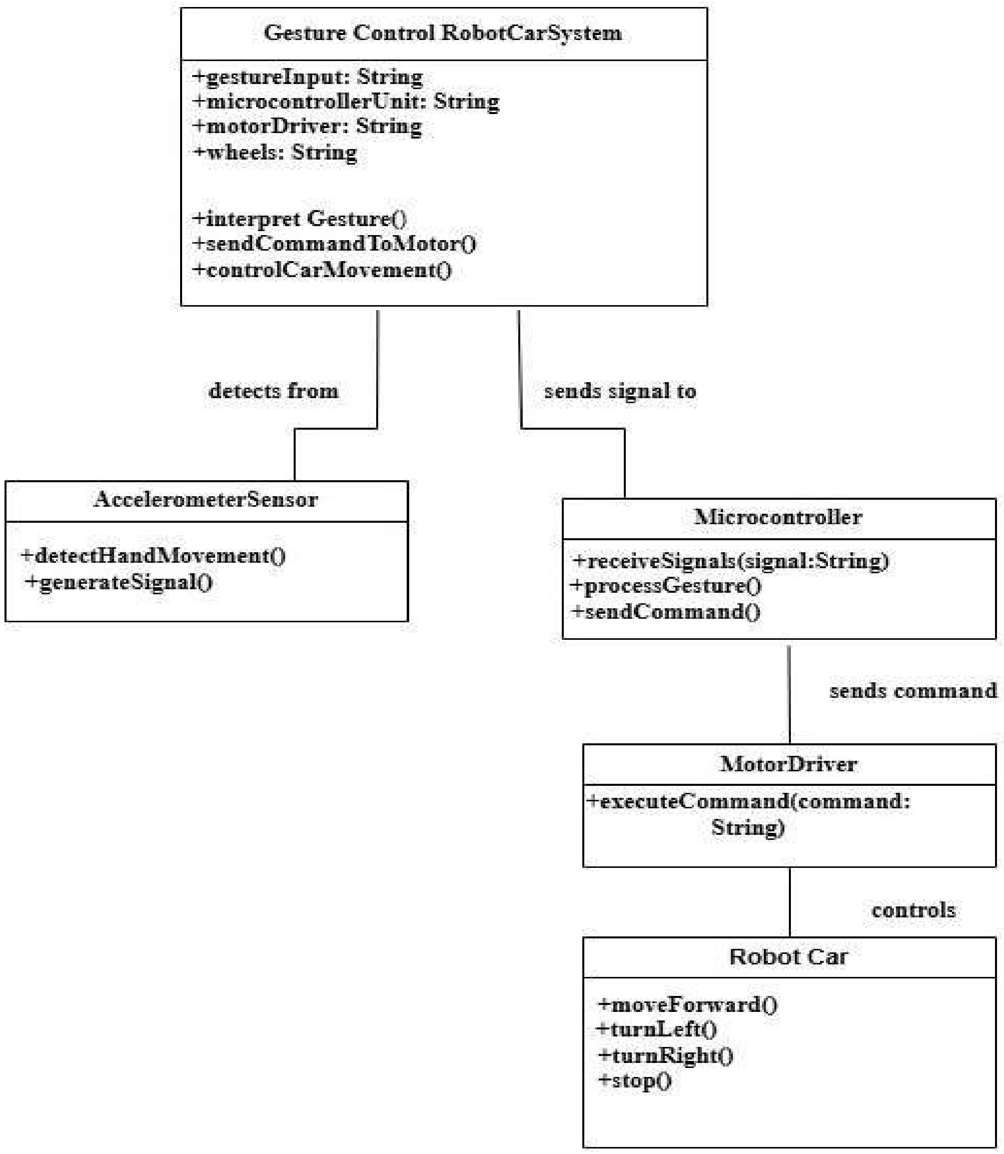


Fig 3.2 : CLASS DIAGRAM

3.2.3 SEQUENCE DIAGRAM

A sequence diagram is a type of behavioral UML diagram that models the interactions between objects in a system over time. It focuses on the sequence of messages exchanged between objects to carry out a particular function or process. The diagram shows objects as vertical lifelines and interactions as horizontal arrows, ordered from top to bottom.
The sequence diagram illustrates the step-by-step interaction between different components of the Gesture Control Robot Car System, showing how a user's hand gestures are translated into robot car movements.

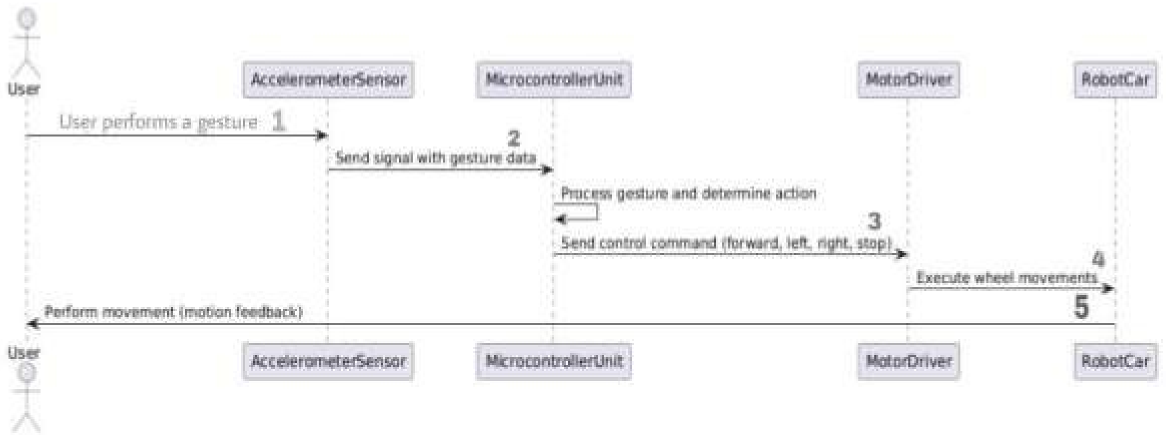

User
AccelerometerSensor
MicrocontrollerUnit
MotorDriver
RobotCar
User performs a gesture 1
2
Send signal with gesture data
Process gesture and determine action
3
Send control command (forward, left, right, stop)
4
Execute wheel movements
Perform movement (motion feedback)
5
User
AccelerometerSensor
MicrocontrollerUnit
MotorDriver
RobotCar

Fig 3.3 : SEQUENCE DIAGRAM

The gesture control process begins with the user performing a specific hand movement, such as a wave, swipe, or a simple raise of the hand. These gestures serve as input commands for the system, representing different directional instructions. By using natural hand motions, the system eliminates the need for traditional physical controllers, creating a more intuitive user experience.

The Accelerometer Sensor plays a crucial role in detecting these gestures. As the user moves their hand, the accelerometer captures changes in acceleration along different axes. It then converts this motion into an electrical signal that represents the specific gesture performed. This sensor-based input method ensures quick and accurate recognition of user intentions.

Once the gesture is detected and converted into a signal, it is transmitted to the Microcontroller Unit (MCU). The microcontroller processes the input data to interpret which motion the user intends to execute. Based on this analysis, it determines the appropriate action—such as moving forward, turning left or right, or bringing the robot to a stop.

After deciding on the movement, the microcontroller sends a corresponding control signal to the Motor Driver module. This motor driver acts as an interface between the microcontroller and the robot car's motors. It receives the command and activates the motors accordingly. For instance, if a right-swipe gesture is detected, the motor driver adjusts the wheel direction to steer the robot car to the right.

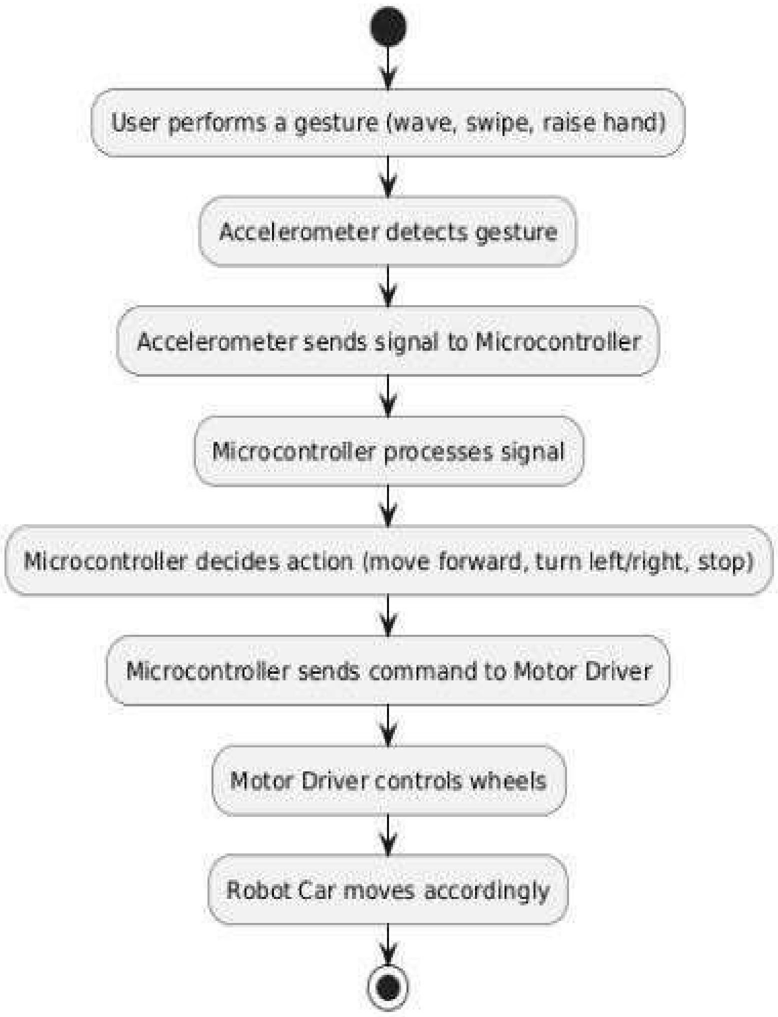
Finally, as the robot car begins to move in response to the gesture, the user observes real-time feedback through its motion. This completes the interaction cycle and reinforces the connection between gesture and action. The system's instant responsiveness and ease of control offer a seamless and wireless interaction, making it both user-friendly and highly accessible.

3.2.4 ACTIVITY DIAGRAM

An activity diagram is a type of behavioral UML diagram that represents the flow of control or data in a system, focusing on the dynamic aspects of a process. It models the sequence of activities, actions, or steps involved in a particular operation or workflow. Activity diagrams use nodes to represent actions or activities and arrows to indicate the flow of control between them.

The given activity diagram represents the working process of the Gesture Control Robot Car in a structured flow. It begins with the user performing a hand gesture, such as waving, swiping, or raising a hand. The accelerometer sensor detects this motion and converts it into an electrical signal. Once the movement is detected, the accelerometer sends this signal to the microcontroller, which acts as the processing unit. The microcontroller processes the received signal and interprets the user's gesture. Based on the recognized gesture, the microcontroller decides the appropriate action that the robot car should take, such as moving forward, turning left, turning right, or stopping. After determining the action, the microcontroller sends a command to the motor driver, which is responsible for controlling the movement of the robot's wheels. The motor driver receives the command and executes it by adjusting the speed and direction of the motors accordingly. As a result, the robot car moves in the intended direction based on the user's gesture. The process continues as the user makes further gestures to control the car's movements. This gesture-based control system provides a hands-free and interactive way to operate the robot car, making it user-friendly and efficient.

The diagram clearly shows how different components interact in a sequential manner, ensuring smooth and accurate operation. It highlights the importance of sensor-based gesture detection, real-time processing, and motor control in enabling a fully functional gesture-controlled robotic system. The communication between the accelerometer, microcontroller, and motor driver ensures a seamless response to user commands. This system can be extended to various applications such as assistive technology, gaming, and automation. The diagram effectively represents the logical flow of operations, making it easy to understand the working of a gesturecontrolled robot car in real time.


User performs a gesture (wave, swipe, raise hand)
Accelerometer detects gesture
Accelerometer sends signal to Microcontroller
Microcontroller processes signal
Microcontroller decides action (move forward, turn left/right, stop)
Microcontroller sends command to Motor Driver
Motor Driver controls wheels
Robot Car moves accordingly

The diagram clearly shows how different components interact in a sequential manner, ensuring smooth and accurate operation. It highlights the importance of sensor-based gesture detection, real-time processing, and precise motor control in enabling a fully functional gesture-controlled robotic system. The communication between the accelerometer, microcontroller, and motor driver ensures a seamless response to user commands, minimizing latency and maximizing responsiveness.

Furthermore, the modular nature of this system allows it to be easily scaled or adapted for various applications beyond just a robot car. For instance, it could be integrated into wearable devices for remote machinery control, used in rehabilitation tools to track and encourage movement in therapy, or applied in smart environments where gestures control household devices. Additionally, the implementation of gesture control paves the way for advancements in human computer interaction, allowing machines to interpret and act on natural body language.

CHAPTER – 4
IMPLEMENTATION

CODE :

4.1 TRANSMITTER CODE:

```
#include <SPI.h>

#include <RF24.h>

#include "Wire.h"

#include "MPU6050.h"


#define CSN_GPIO      8      // NRF24L01 CSN pin

#define CE_GPIO       9      // NRF24L01 CE pin

#define TX_ENABLE_KEY 2      // Toggle button on D2 (use INPUT_PULLUP)

#define TX_LED        3      // LED indicator on D3


RF24 radio(CE_GPIO, CSN_GPIO);  // NRF24L01 instance (CE=7, CSN=8)

MPU6050 mpu;


// Sensor variables int16_t

ax, ay, az; int16_t gx, gy,

gz;


// Communication address (must match RX) const byte

Address[6] = "00001";


// Two bytes to send:

//   command: 0 = Neutral, 1 = Forward, 2 = Backward, 3 = Left, 4 = Right
//   speed: value from 0 to 160 (max) unsigned
char command = 0; unsigned
char speed = 0;


bool txEnabled = false;  bool
lastButton = HIGH;


void                  setup()                  {
Serial.begin(115200);
  pinMode(TX_ENABLE_KEY, INPUT_PULLUP);   pinMode(TX_LED,
OUTPUT);   digitalWrite(TX_LED,
LOW);


  radio.begin();   radio.openWritingPipe(Address);
radio.setPALevel(RF24_PA_MAX);
  radio.stopListening();


  Wire.begin();   mpu.initialize();
  Serial.println(mpu.testConnection() ? "MPU6050 connected" : "MPU6050 connection
failed");
}


void loop() {
  // Toggle transmission with button (D2)   bool
btn = digitalRead(TX_ENABLE_KEY);   if
```

```
(btn == LOW && lastButton == HIGH) {
txEnabled = !txEnabled;
Serial.println(txEnabled ? "TX Enabled" : "TX
Disabled");     digitalWrite(TX_LED,
txEnabled ? HIGH : LOW);     delay(50);
// simple debounce
}       lastButton  =
btn;   // Read
MPU6050 sensor
data
mpu.getMotion6(
&ax, &ay, &az,
&gx, &gy, &gz);

  // Compute direction and speed based on tilt:
  // Using a threshold of 4000; maximum tilt is assumed to be ~16384.   const
int threshold = 4000;   command = 0;   speed = 0;

  // If Y-axis shows significant tilt (and X is near level), use forward/backward:   if
(abs(ay) > threshold && abs(ax) < threshold) {     if (ay < -threshold) { //
Forward       command = 1;
      // Map -ay from threshold to 16384 into speed 0 to 160       speed
= map(-ay, threshold, 16384, 0, 160);       if (speed >
160) speed = 160;     } else if (ay > threshold) { // Backward
command = 2;       speed = map(ay, threshold, 16384, 0,
160);       if (speed > 160) speed =
160;
    }
}
  // If X-axis shows significant tilt (and Y is near level), use left/right:
  else if (abs(ax) > threshold && abs(ay) < threshold) {
if (ax > threshold) { // Left       command = 3;       speed
= map(ax, threshold, 16384, 0, 160);       if
(speed > 160) speed = 160;
    } else if (ax < -threshold) { // Right       command
= 4;       speed = map(-ax, threshold,
16384, 0, 160);       if (speed > 160) speed =
160;
    }
  } else {
command = 0;     speed
= 0;
  }

  // Print computed values for debugging
Serial.print("Direction: ");   switch
(command) {     case 1:
Serial.print("Forward"); break;     case 2:
Serial.print("Backward"); break;     case 3:
Serial.print("Left"); break;     case 4:
Serial.print("Right"); break;     default:
Serial.print("Neutral");
```

```
  }
  Serial.print(" | Speed: ");
  Serial.println(speed);

  // Transmit data if enabled   if (txEnabled) {
radio.stopListening();  // Ensure TX mode
unsigned char data[2] = { command, speed };     bool
ok = radio.write(data, sizeof(data));     if
(ok) {
      Serial.println("Data sent successfully");
    } else {
      Serial.println("Send failed");
    }
  }

  delay(100);
}
```

tx code , gesture control robotcar

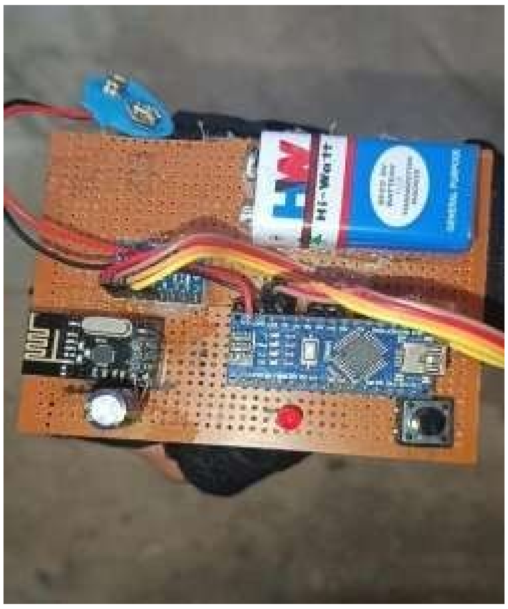
4.2 RECEIVER CODE:

```
#include <SPI.h>

#include <RF24.h>


// Motor Driver Pin Definitions for L298N with Enable pins

// Left Motor (Motor A)

#define LM_IN1 2    // Connects to IN1 of L298N (Motor A)

#define LM_IN2 3    // Connects to IN2 of L298N (Motor A)

#define ENA    5    // Connects to EN A (Motor A Enable)


// Right Motor (Motor B)

#define RM_IN1 4    // Connects to IN3 of L298N (Motor B)

#define RM_IN2 7    // Connects to IN4 of L298N (Motor B)

#define ENB    6    // Connects to EN B (Motor B Enable)


// NRF24L01 setup – using basic testing configuration (CE=9, CSN=10)
RF24 radio(9, 10);  // CE at pin 9, CSN at pin 10 const byte Address[6]
= "00001";  // Must match TX address


// Buffer for received data: [0]=command, [1]=speed (0-160) unsigned char
rxData[2];


void setup() {
  Serial.begin(115200);
  // Set up motor driver pins as outputs
  pinMode(LM_IN1,                         OUTPUT);
pinMode(LM_IN2, OUTPUT);   pinMode(ENA,
OUTPUT);       pinMode(RM_IN1,   OUTPUT);
pinMode(RM_IN2, OUTPUT);   pinMode(ENB,
OUTPUT);

  stopMotors();  // Ensure motors are off at startup

  //   NRF24L01   Initialization   –   using   basic   testing   settings
radio.begin();   radio.openReadingPipe(0, Address);  // Using reading
pipe 0   radio.setPALevel(RF24_PA_LOW);        // Set power level to
LOW   radio.startListening();


  Serial.println("Receiver ready...");
}

void loop() {   if (radio.available()) {
radio.read(&rxData, sizeof(rxData));
unsigned char cmd = rxData[0];     unsigned
char spd = rxData[1];
Serial.print("Received cmd: ");
    Serial.print(cmd);
    Serial.print("  Speed: ");
    Serial.println(spd);
    // Execute movement commands:
    // 1 = Forward, 2 = Backward, 3 = Left, 4 = Right, 0 = Stop (Neutral)
if (cmd == 1) {       moveForward(spd);     } else if (cmd == 2) {
moveBackward(spd);     } else if (cmd == 3) {       turnLeft(spd);     } else
if (cmd == 4) {       turnRight(spd);     } else {       stopMotors();
    }
  }
}

// Moves both motors forward at the given speed using PWM on ENA and ENB void
moveForward(unsigned char spd) {   // Set left motor forward: IN1 HIGH,
IN2 LOW   digitalWrite(LM_IN1, HIGH);   digitalWrite(LM_IN2, LOW);
  // Set right motor forward: IN1 HIGH, IN2 LOW
  digitalWrite(RM_IN1, HIGH); digitalWrite(RM_IN2,
  LOW);

  // Control speed using PWM on enable pins   analogWrite(ENA,
spd);   analogWrite(ENB,
spd);

  Serial.println("Moving Forward");
}
```

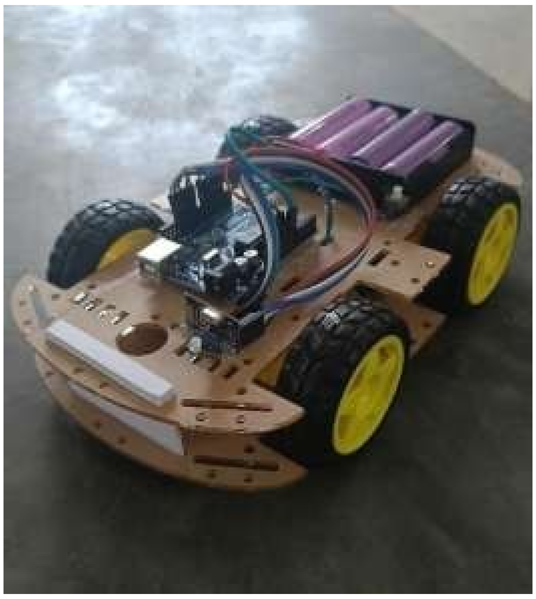
```
// Moves both motors backward at the given speed
void moveBackward(unsigned char spd) {   // Set left
motor backward: IN1 LOW, IN2 HIGH
digitalWrite(LM_IN1, LOW);
digitalWrite(LM_IN2, HIGH);
  // Set right motor backward: IN1 LOW, IN2 HIGH   digitalWrite(RM_IN1,
LOW);   digitalWrite(RM_IN2,
HIGH);

  // Control speed using PWM on enable pins   analogWrite(ENA,
spd);   analogWrite(ENB,
spd);

  Serial.println("Moving Backward");
}

// Turns left: left motor backward, right motor forward void turnLeft(unsigned
char spd) {
  //   Left   motor   backward:   IN1   LOW,   IN2   HIGH   digitalWrite(LM_IN1,   LOW);
   digitalWrite(LM_IN2, HIGH);
  // Right motor forward: IN1 HIGH, IN2 LOW   digitalWrite(RM_IN1,
HIGH);   digitalWrite(RM_IN2,
LOW);

  analogWrite(ENA,                              spd);
analogWrite(ENB, spd);
Serial.println("Turning Left");
}

// Turns right: left motor forward, right motor backward
void turnRight(unsigned char spd) {   // Left motor
forward: IN1 HIGH, IN2 LOW   digitalWrite(LM_IN1,
HIGH);   digitalWrite(LM_IN2, LOW);   // Right motor
backward: IN1 LOW, IN2 HIGH
digitalWrite(RM_IN1, LOW);   digitalWrite(RM_IN2,
HIGH);

  analogWrite(ENA,                    spd);
analogWrite(ENB, spd);

  Serial.println("Turning Right");
}
// Stops all motors void stopMotors()
{
  // Set the enable pins to 0 PWM value analogWrite(ENA, 0); analogWrite(ENB, 0);
  // Optionally, set motor direction pins to LOW
digitalWrite(LM_IN1, LOW);   digitalWrite(LM_IN2,
LOW);             digitalWrite(RM_IN1,      LOW);
digitalWrite(RM_IN2, LOW);
  Serial.println("Motors Stopped");
}
```

CHAPTER – 5
OUTPUTS

5.1 OUTPUT SCREENS



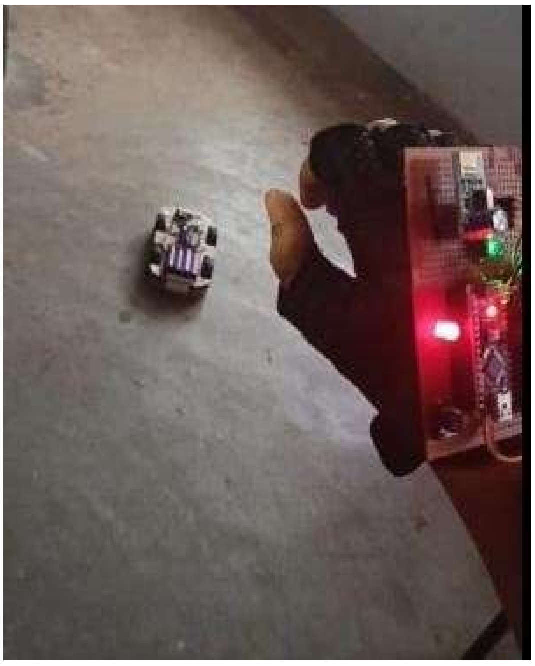
HAND GESTURE



GESTURE CONTROLLED CAR

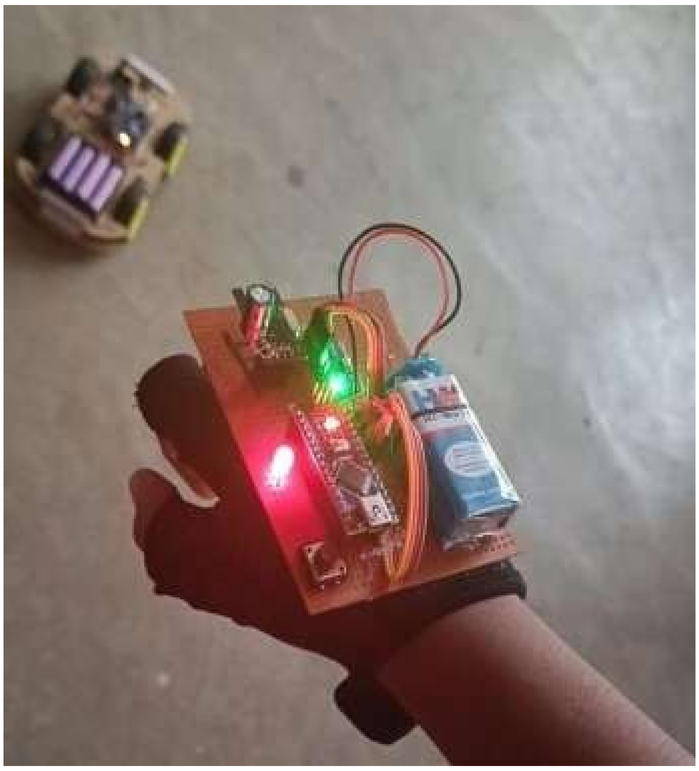
TURNING RIGHT

TURNING LEFT



MOVING BACKWARD

MOVING FORWARD



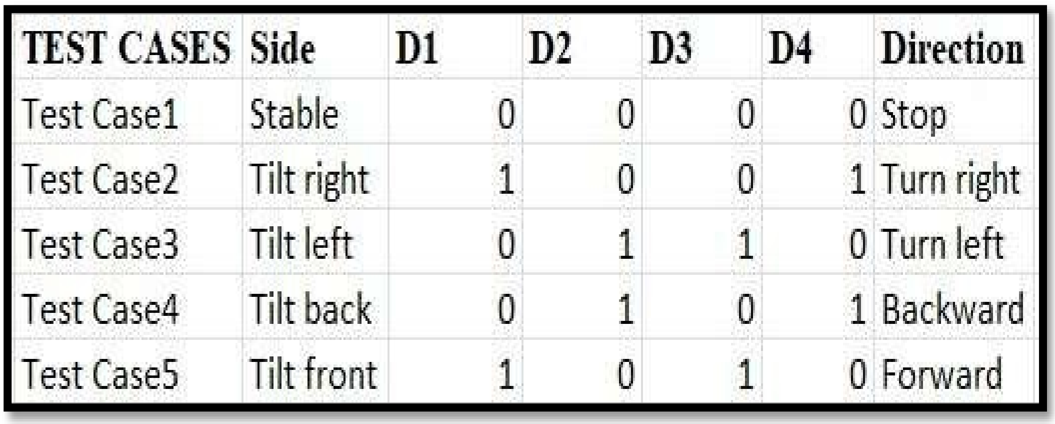
HOLD HAND STILL OR FLAT

CHAPTER – 6

TESTING

TESTING

6.1 TESTCASE 1:

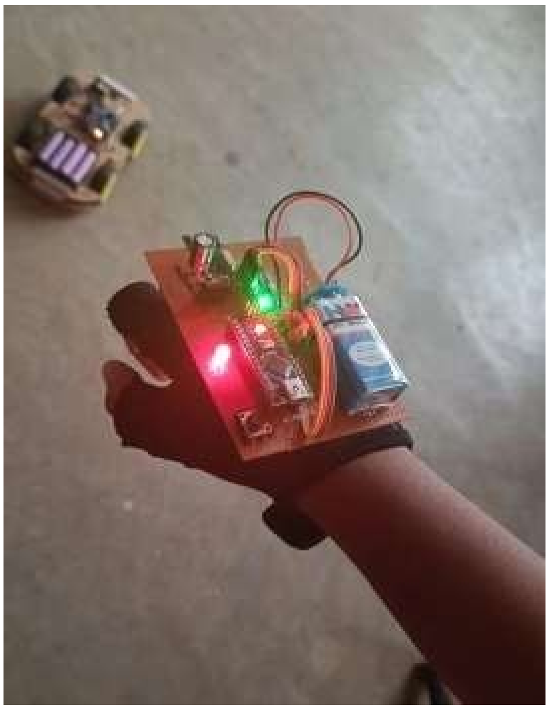


TEST CASES	Side	D1	D2	D3	D4	Direction
Test Case1	Stable	0	0	0	0	Stop
Test Case2	Tilt right	1	0	0	1	Turn right
Test Case3	Tilt left	0	1	1	0	Turn left
Test Case4	Tilt back	0	1	0	1	Backward
Test Case5	Tilt front	1	0	1	0	Forward

TEST CASE 1: HOLD HAND STILL OR FLAT

When we keep our hand flat, the car will not move and it will be in a stable position without any movement

- **Input:** Hold accelerometer flat / in neutral position.
- **Expected Result:** Car stops all movement.



TEST CASE 2 : TILT RIGHT

When we tilt our hand towards right side, the car starts moving to the right side

- **Input:** Tilt accelerometer to the right.
- **Expected Result:** Car turns right.

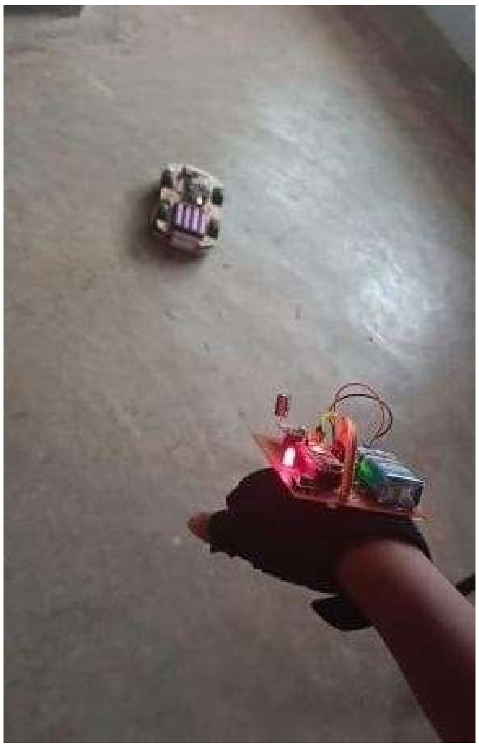
TEST CASE 3 : TILT LEFT

When we tilt our hand towards left side, the car starts moving to the left side.

- **Input:** Tilt accelerometer to the left.
- **Expected Result:** Car turns left.

TEST CASE 4 &5 :

TILT FORWARD & BACKWARD



When we move our hands to the forward

(hand move to the down side) the car moves forward direction

When we move our hands to the backwards (hand move to the upwards)

the car moves back direction

6.2 FUNCTIONAL TESTING

CASE 1 :

MPU6050 Accelerometer Testing

- **Calibration & Sensitivity**: The accelerometer accurately detected gestures (forward, backward, left, right), though small tilts sometimes caused misinterpretations, and neutral position detection could be improved.
- **Performance**: The accelerometer was responsive to quick gestures but occasionally misinterpreted small or slow movements.

CASE 2:

L298N Motor Driver Testing

- **Motor Direction Control**: The motor driver correctly controlled motor direction for forward, backward, left, and right movements.
- **Speed Control & Current Draw**: Speed control worked well, and the motor driver didn’t overheat during testing. Current draw remained within expected limits.

CASE 3:

DC Motor Testing

- **Operation & Load Testing**: The DC motors performed well, with smooth motion and no stalling even under higher loads.
- **Speed Testing**: The robot maintained smooth motion, though battery voltage slightly dropped at higher speeds.

CASE 4:

Power System Testing

- **Battery Life & Stability**: The power system remained stable, providing 1-2 hours of continuous operation with no power interruptions or overheating.

CHAPTER – 7

CONCLUSION

CONCLUSION

We introduced a Gesture Control Robot Car project successfully implements an innovative, intuitive, and user-friendly way of controlling a robot car using hand gestures. By leveraging an accelerometer sensor, a microcontroller, and a motor driver, the system effectively translates user gestures into real-time movements, eliminating the need for physical remote controls or predefined programming. The wireless communication between the controller and the robot ensures smooth and flexible operation, enhancing user experience.This project provides a more natural and accessible way of interaction, making it beneficial for individuals with limited mobility or those looking for an immersive robotic control experience. While the system improves convenience and adaptability, future enhancements such as machine learning-based gesture recognition, obstacle detection, and AI integration could further refine its capabilities. Overall, the Gesture Control Robot Car demonstrates a significant advancement in human machine interaction and paves the way for future developments in gesture-based robotic control.By exploring these applications and potential improvements, the gesturecontrolled robot car project can evolve into a versatile platform that not only showcases innovative technology but also addresses real-world needs across various sector.

CHAPTER-8

FUTURE SCOPE

FUTURE SCOPE

The Gesture Control Robot Car has great potential for future improvements and applications in various fields. One of the key areas for advancement is the integration of machine learning and AI-based gesture recognition, which can improve the accuracy and adaptability of the system. This would allow the robot to learn and recognize a wider range of gestures, making the control process more seamless and responsive. Additionally, incorporating obstacle detection and autonomous navigation using sensors like ultrasonic or infrared can enhance the robot’s ability to move safely in dynamic environments. Another important future development is expanding the system for real-world applications, such as assistive technologies for people with disabilities, industrial automation, and even smart home applications where gestures can control multiple devices. Improving the wireless communication range and reducing signal interference will also make the system more reliable for larger distances. Furthermore, integrating a camera module and computer vision could enable the robot to recognize facial expressions or voice commands, providing a multi-modal interaction experience. With these advancements, the Gesture Control Robot Car can evolve into a more intelligent and efficient robotic system with applications in daily life, healthcare, and automation industries.

CHAPTER – 9

REFERENCES

REFERENCES

1. Murthy, G R. S., & Jadon, R. S. (2009). A Review of Vision Based Hand Gestures Recognition; International Journal of Information Technology and Knowledge Management,405-410.

2. Garg, P., Aggarwal, N., & Sofat, S. (2009). Vision Based Hand Gesture Recognition: World Academy of Science, Engineering and Technology, 49, 972-977.

3. Fakhreddine, K., Milad, A., Jamil, A. S., & Mo Nours, A. (2008). Human-Computer Interaction: Overviewon State ofthe Art: International Journal on Smart Sensing and Intelligent Systems, .

4. Human computer interaction. (2019, December 2). Retrieved from https://en.wikipedia.org/wiki/Human% E2%80%93computer_interaction

5. Mokhtar, M., Hasan, Pramoud, K, Misra, (2011). Brightness Factor Matching For Gesture Recognition System Using Scaled Normalization: International Journal of Computer Science & Information Technology (IJCSIT).

6. Xingyan , Li. (2003). Gesture Recognition Based on Fuzzy C-Means Clustering Algorithm: Department of Computer Science. The University of Tennessee Knoxville.Smith, J. (2022). Introduction to Gesture Control Technologies. Robotics Journal, 112-119.Jones, A. & Wang, L. (2023). Microcontrollers in Modern Robotics.Tech Innovations.Brown, M. (2021). The Future of Gesture Recognition: Trends and Technologies. Journal of Interaction Science.This presentation outline provides a comprehensive overview of the nesture

7. Mitra, S., & Acharya, T., (2007). Gesture Recognition: A Survey: IEEE Transactions on systems. Man and Cybernetics. Part C: Applications are viewer, 311-324

8. Simci, G., Wysoski, Marcus, V., Lamar, Susuma, K. & Akira, Iwata, (2002). A Rotation Invariant Approach On Static Gesture Recognition Using Boundary Histograms And Neural Networks: IEEE0 Proceedings of the 9th leternational Conference on Neural Information Processing. Singapura,

9. Joseph J& LaViola, Jr. (1999), A Survey of Hand Posture and Gesture Recognition Techniques and Technology: Master Thesis, Science and Technology Center for Computer Graphics and Scientific Visualization. USA.

10. Rafiqul 7. Khan Noor A. & Ibraheem. (2012). Survey Gesture Recognition for Hand Image Postures: Prernational Journal of Compnder Anal Information Science 5(3)

11. Thomas B., Mocslund., & Erik G (2001). A Survey of Computer Vision-Based Human Motion Capture: Elsevier Conpider Asinu and Lange Understanding, 81,231-268.