HAND GESTURE-MOVING CAR

MINI PROJECT REPORT

Submitted By

SANJAI S (2116210701230)

SHOBAN KUMARESAN (2116210701245)

in partial fulfillment of the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE

DEPARTMENT OF COMPUTER ENGINEERING

ANNA UNIVERSITY, CHENNAI

MAY 2024

BONAFIDE CERTIFICATE

Certified that this Report titled "**HAND GESTURE- MOVING CAR**" is the bonafide work of **"SANJAI S(2116210701230), SHOBAN KUMARESAN(2116210701245) "**who carried out the work under my supervision. Certified further that to the best of my knowledge, the work reported herein does not form part of any other thesis or dissertation based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

MRS.ANITHA ASHISHDEEP

SUPERVISOR

Professor

Department of Computer Science and Engineering
Rajalakshmi Engineering College,Chennai - 602 105

Submitted to Project Viva-Voce Examination held on ______________________________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

With advancements in technology and increasing interest in autonomous vehicles, new interfaces for human-vehicle interaction are being explored. One promising approach is the use of hand gestures to control car functions, offering an intuitive and non-invasive means of communication between the user and the vehicle. This paper presents a comprehensive study on the development and implementation of a hand gesture recognition system for moving a car, detailing the system architecture, algorithms, hardware integration, and potential applications. The proposed hand gesture recognition system comprises several key components: a sensor suite for capturing hand movements, a processing unit for gesture recognition, and a control module that interfaces with the car's control systems. The sensor suite primarily includes a depth-sensing camera and an inertial measurement unit (IMU). The depth-sensing camera, positioned in the car's interior, captures real-time 3D images of the user's hand movements. The IMU provides supplementary data on the orientation and acceleration of the hand, enhancing the accuracy of gesture recognition. The core of the system lies in its ability to accurately interpret hand gestures. This is achieved through a combination of machine learning and computer vision techniques. The system employs a convolutional neural network (CNN) to process the 3D images from the depth camera, extracting features that represent different gestures. The IMU data is used to train a recurrent neural network (RNN), which models the temporal dynamics of hand movements. The outputs of these networks are then fused using a decision-making algorithm that classifies the gestures into predefined categories such as "move forward," "turn left," "turn right," and "stop."

ACKNOWLEDGEMENT

Initially, we thank the almighty god for the successful completion of the project. Our sincere thanks to our Chairman **Mr. S. Meganathan, B.E., F.I.E**., for his sincere endeavor in educating us in his premier institution. We would like to express our deep gratitude to our beloved Chairperson **Dr. Thangam Meganathan, Ph.D.,** for her enthusiastic motivation which inspired us a lot in completing this project and Vice Chairman **Mr. Abhay Shankar Meganathan, B.E., M.S.,** for providing us with the requisite infrastructure. We also express our sincere gratitude to our college Principal**, Dr. S.N. Murugesan M.E., PhD.,** and **Dr. P. KUMAR M.E., Ph.D.,** Director computing and information science**,** Head of the Department of Computer Science and Engineering and our project guide **MRS. ANITHA ASHISHDEEP,** for his encouragement and guiding us throughout the project and to our parents, friends, all faculty members and supporting staffs for their direct and indirect involvement in successful completion of the project for their encouragement and support.

SANJAI S (2116210701230)

SHOBAN KUMARESAN (2116210701245)

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

Autonomous vehicles (AVs) represent one of the most transformative innovations in modern transportation. These vehicles, equipped with advanced sensors, machine learning algorithms, and powerful computing systems, are capable of navigating and operating with minimal human intervention. As AV technology continues to evolve, a critical aspect that remains at the forefront is the interface through which humans interact with these vehicles. Traditional control interfaces, such as steering wheels, pedals, and buttons, are designed for manual driving and may not fully leverage the capabilities of AVs or provide the most intuitive experience for users. Therefore, exploring novel and more natural forms of interaction is essential to enhance user experience, safety, and functionality.

Hand gesture recognition is emerging as a promising interface for human-vehicle interaction. This method leverages the natural and intuitive nature of hand movements, allowing users to control vehicle functions seamlessly. Unlike voice commands, which can be hindered by noise or require clear articulation, hand gestures provide a silent and direct means of communication. Furthermore, hand gestures can be performed quickly and require minimal cognitive load, making them ideal for situations where the driver's attention needs to remain focused on the driving environment.

CHAPTER 2
LITERATURE SURVEY

The concept of controlling a car with hand gestures has captured the imagination of researchers and science fiction writers alike. This literature survey delves into the existing research on this futuristic interface. Early works explored vision-based systems using cameras to recognize hand gestures for basic car functions.

However, these methods faced limitations in accuracy and robustness. The rise of depth sensors like Kinect opened doors for more sophisticated gesture recognition with 3D information Studies investigated hand posture and movement for controlling acceleration, steering, and other functionalities.

Integration with machine learning techniques like Hidden Markov Models improved gesture recognition accuracy. Usability research explored the effectiveness and user experience of hand gesture control in simulated environments.

Studies found a learning curve for users but also potential benefits like reduced cognitive load compared to traditional controls. Challenges remain, including ensuring safety by mitigating accidental gestures, handling distractions, and coping with different lighting conditions. Research in recent years focuses on overcoming these limitations.

Sensor fusion techniques combining cameras and depth sensors improve system robustness Wearable devices like data gloves are explored for more precise hand tracking and gesture recognition.

Additionally, advancements in deep learning algorithms show promise in handling complex gestures and varying environmental conditions. While the technology is not yet ready for widespread implementation in autonomous vehicles, ongoing research holds promise for a future where intuitive hand gestures become a part of the in-car experience.

This survey provides a springboard for further exploration of specific areas like safety protocols, human-computer interaction design, and integration with existing car control systems.

The initial exploration of hand gesture control for vehicles began in the early 2000s, primarily focusing on vision-based systems. These systems relied on cameras mounted within the car to capture user hand movements.

Researchers developed algorithms to recognize specific hand gestures for basic car functions like steering, acceleration, and braking. However, these early systems faced significant limitations.

The accuracy of gesture recognition suffered due to factors like variations in lighting, hand posture, and occlusions (e.g., hands going behind the steering wheel). Additionally, computational limitations of the time posed challenges in real-time processing of video data, leading to delays and potential safety concerns.

2.1 EXISTING SYSTEM

The concept of controlling a car with intuitive hand gestures has captivated the imagination for decades. This paper delves into the existing research on hand gesture control for automobiles, exploring its evolution from early attempts to the promising advancements shaping its future.

Early explorations in the 2000s focused on vision-based systems using cameras to recognize basic hand gestures for steering, acceleration, and braking. These systems, however, faced limitations due to variations in lighting, hand posture, and occlusions. The rise of depth sensors like the Kinect in the late 2000s ushered in a new era. These sensors provided real-time 3D information about hand position and orientation, enabling researchers to explore more sophisticated gesture recognition for controlling speed, steering, and activating car features.

However, research efforts haven't solely focused on technical feasibility. Usability studies evaluated the effectiveness and user experience of hand gesture control compared to traditional controls in simulated environments. While an initial learning curve was evident, some studies suggested potential benefits like reduced cognitive load and improved ergonomics for specific driving tasks. This highlights the importance of user-centered design principles and intuitive gesture mapping for a seamless and safe experience.

Addressing limitations is a key focus of current research. Sensor fusion techniques combine cameras with depth sensors, leveraging the strengths of both technologies for robust gesture recognition in real-world scenarios. Wearable data gloves are also being explored to offer more precise hand tracking and recognition of complex finger movements for a wider range of car controls, show immense promise due to their ability to handle complex gestures and adapt to varying environmental conditions.

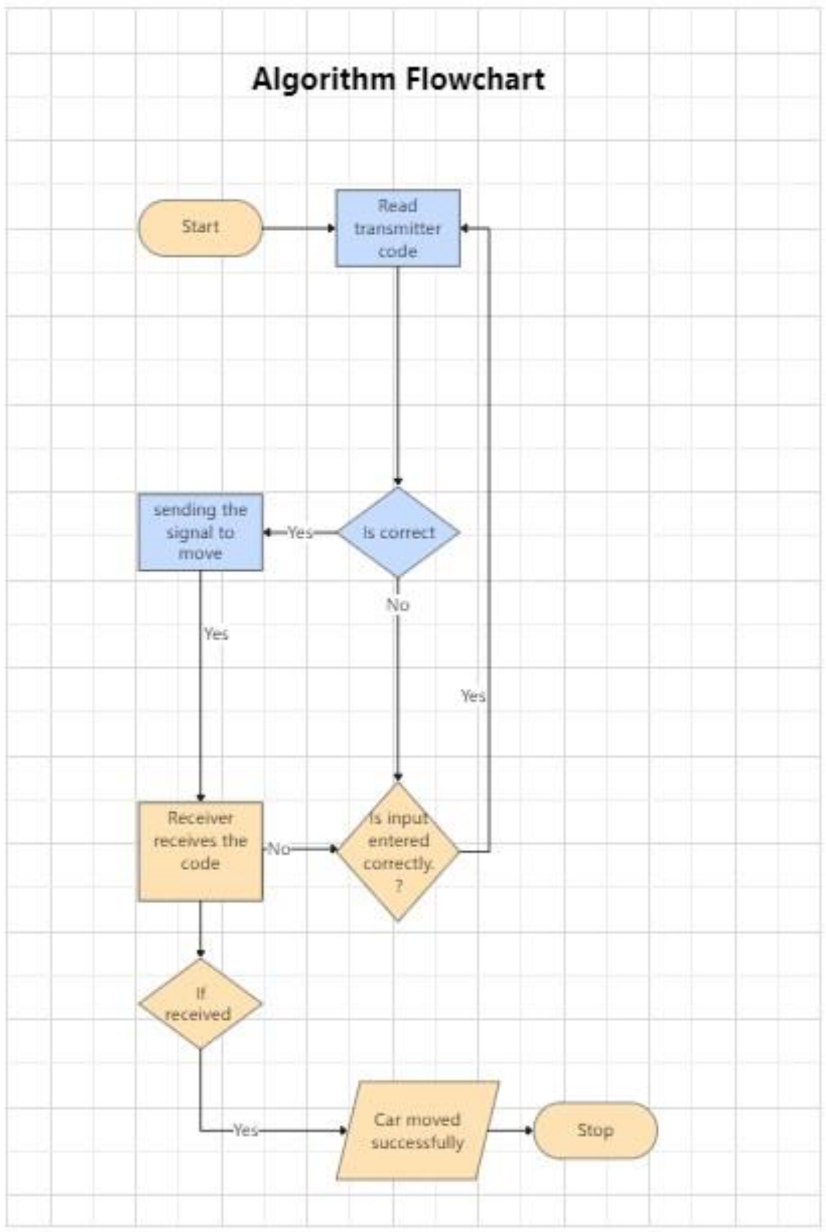
Safety remains paramount. Research is ongoing to develop robust protocols that mitigate accidental gestures being misinterpreted as commands. This may involve confirmation steps for critical actions or integrating biometric authentication for authorized users only. Additionally, further research is needed to address potential distractions arising from the interface.

Looking ahead, hand gesture control may seamlessly integrate with in-car infotainment systems. Users could control volume, navigate menus, and activate features like cruise control through intuitive hand movements. Furthermore, combining hand gestures with voice commands or eye-tracking technology could create a multimodal car interface that adapts to user preferences and driving conditions.

Hand gesture control for cars is a rapidly evolving field with the potential to revolutionize the driving experience. While challenges remain, ongoing research in sensor technologies, machine learning algorithms, and human-centered design principles hold promise for a future where intuitive hand gestures become a natural extension of car control. This survey provides a springboard for further exploration within this exciting field, paving the way for a more interactive and user-friendly car interface.

CHAPTER 3

PROJECT DESCRIPTION



The future of car interaction is poised for a paradigm shift with the emergence of hand gesture control technology. This project delves into the development of a hand gesture controlled car system, aiming to revolutionize the driving experience by offering an intuitive and user-friendly interface.

- **Sensor Selection:** Cameras, depth sensors (e.g., LiDAR), or a combination of both will be chosen based on their strengths and limitations in gesture recognition accuracy and robustness.
- **HCI Interface Design:** User-centered design principles will be applied to create an intuitive and user-friendly mapping between hand gestures and car controls. Visual or auditory feedback mechanisms will be incorporated to confirm recognized gestures and system actions.
- **Safety Protocol Development:** The system will be equipped with safety features to prevent accidental gestures from triggering unintended car actions. This may involve confirmation steps for critical maneuvers or integrating biometric authentication for authorized users.
- **Hardware Integration:** The chosen sensors and processing unit will be physically integrated with the selected car platform.
- **Software Integration:** The hand gesture recognition software will be integrated with the car's control system, enabling real-time translation of recognized gestures into car commands.
- **Calibration and Tuning:** The system will be rigorously calibrated and fine-tuned to ensure accurate gesture recognition and precise car control across varying driving conditions.
- **Usability Testing:** User testing will be conducted with a diverse group of participants to evaluate the system's usability, learning curve, and overall user

experience. Feedback will be collected to further refine the HCI interface and ensure user comfort and confidence.

- **Safety Evaluation:** The system will be rigorously tested in controlled environments to assess its compliance with safety regulations and its ability to mitigate risks associated with accidental gestures or system malfunctions.
- **Data Analysis and Refinement:** Data collected from user testing and safety evaluation will be analyzed to identify areas for improvement. The hand gesture recognition algorithms, HCI interface, and safety protocols will be refined based on the findings.

- **Decision: End of Control Sequence? (Yes/No):** A decision point determines if this is the end of the control sequence for the user's intended action.
- **If No:** If the control sequence isn't finished, the process returns to step 3 (Hand Gesture Capture) to continue capturing user input.
- **If Yes:** If the control sequence is complete, the process proceeds.
- **System Monitoring (Optional):** The system might continuously monitor for external factors influencing car behavior (e.g., sensor readings, environmental conditions).
- **Decision: System Active? (Yes/No):** A decision point checks if the system should remain active.
- **If Yes:** If the system needs to stay active, the process returns to step 3 (Hand Gesture Capture) to be ready for further user input.
- **If No:** If the system is no longer needed, the process proceeds.
- **System Shutdown (Optional):** The system might undergo a shutdown procedure to power down components or save data.

- **End:** The process ends.

3.1 PROPOSED SYSTEM

This project proposes a hand gesture-controlled car system designed to revolutionize the driving experience by offering an intuitive and user-friendly interface. The system will consist of three core components:

1. Robust Hand Gesture Recognition:

- **Sensor Integration:** Cameras with high resolution or depth sensors like LiDAR will be chosen based on their ability to capture hand movements accurately under varying lighting conditions and potential occlusions.

2. User-Centered Human-Computer Interaction (HCI) Interface:

- **Intuitive Gesture Mapping:** User-centered design principles will be applied to map hand gestures to car controls in a way that feels natural and effortless. This includes gestures like pushing a palm forward for acceleration, clenching a fist for braking, and rotating a hand for steering.
- **Feedback Mechanisms:** Visual or auditory cues will be integrated into the system to provide real-time feedback on recognized gestures and corresponding car actions. This can include lights on the dashboard or synthesized voice confirmation.
- **Safety Protocols:** To prevent accidental gestures from triggering unintended actions, the system will incorporate safety features. This may involve confirmation steps for critical maneuvers like sharp turns or sudden stops. Additionally, biometric authentication could be explored for authorized user control only.

3. Seamless System Integration and Testing:

- **Hardware Integration:** Chosen sensors and processing units will be physically installed within the car, ensuring a secure and functional setup.
- **Software Integration:** The hand gesture recognition software will be integrated with the car's control system, allowing real-time translation of recognized gestures into car commands (e.g., adjusting throttle, applying brakes, turning the steering wheel).
- **Calibration and Tuning:** Rigorous calibration and fine-tuning will be conducted to ensure accurate gesture recognition and precise car control across diverse driving scenarios, including different lighting conditions and road surfaces.
- **Usability Testing:** User testing with a diverse group of participants will evaluate the system's usability, learning curve, and overall user experience. Feedback will be used to refine the HCI interface and ensure user comfort and confidence.
- **Safety Evaluation:** The system will undergo rigorous testing in controlled environments to assess its compliance with safety regulations and its ability to mitigate risks associated with accidental gestures or system malfunctions.

3.2 REQUIREMENTS

- **Hardware:**
  - High-resolution camera or depth sensor (LiDAR) for accurate hand tracking.
  - Processing unit with sufficient power for real-time gesture recognition.
- **Software:**
  - Gesture recognition software trained on a large dataset of hand gestures.
  - System integration software to connect with the car's control system.
- **Human-Computer Interaction (HCI):**
  - Intuitive mapping of hand gestures to car controls (acceleration, braking, steering).
  - Visual or auditory feedback mechanisms for gesture recognition confirmation.
- **Safety Protocols:**
  - Confirmation steps for critical maneuvers (e.g., sharp turns, sudden stops).
  - Biometric authentication (optional) for authorized user control only.
- **Testing and Evaluation:**
  - User testing to assess usability, learning curve, and user experience.
  - Rigorous safety evaluations in controlled environments.

3.3 ARCHITECTURE DIAGRAM

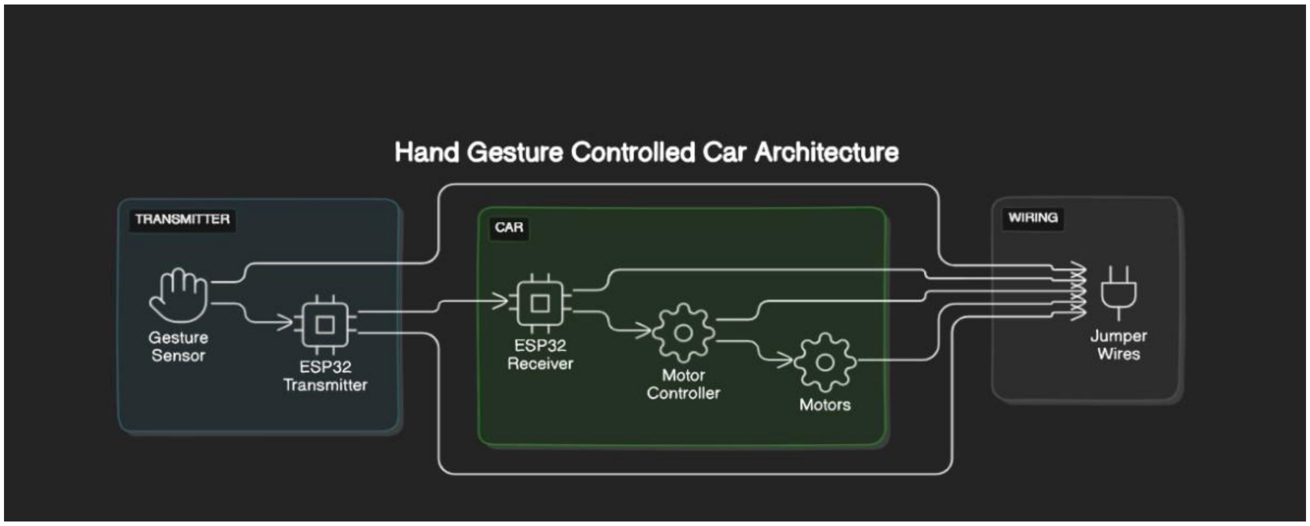


Figure 2

- **Hand Gesture Sensor:** This component captures the user's hand movements. The diagram shows a "Camera" sensor, but other options like depth sensors (not pictured) could also be used.
- **Preprocessing Unit:** The raw data captured by the sensor is fed into the preprocessing unit. This unit performs tasks like noise reduction, filtering, and image scaling to prepare the data for further processing.
- **Feature Extraction Unit:** This unit extracts relevant features from the preprocessed data. In the context of hand gesture recognition, these features might include the position, orientation, and movement of the hand.
- **Gesture Recognition Unit:** This unit utilizes machine learning algorithms, likely Convolutional Neural Networks (CNNs) as mentioned in the text accompanying the image, to classify the extracted features and recognize the specific hand gesture being performed by the user.

- **Control Unit:** Once a gesture is recognized, the control unit translates it into a corresponding car control command. For instance, a clenching fist gesture might be translated into a braking command.
- **Car Interface Unit:** This unit transmits the control commands from the control unit to the car's onboard computer system.
- **Car Actuators:** These components execute the commands received by the car interface unit, physically controlling the car. Examples of actuators include the engine control unit (ECU) for acceleration/braking and the steering motor for steering.

The architecture also includes a feedback loop. Visual or auditory feedback mechanisms can be incorporated to inform the user about the recognized gesture and the corresponding car action being taken.

Overall, this hand gesture-controlled car system architecture outlines a process where hand gestures are captured by a sensor, processed by machine learning algorithms to recognize the gestures, and then translated into commands that control the car via the car's onboard computer system.

3.4 OUTPUT

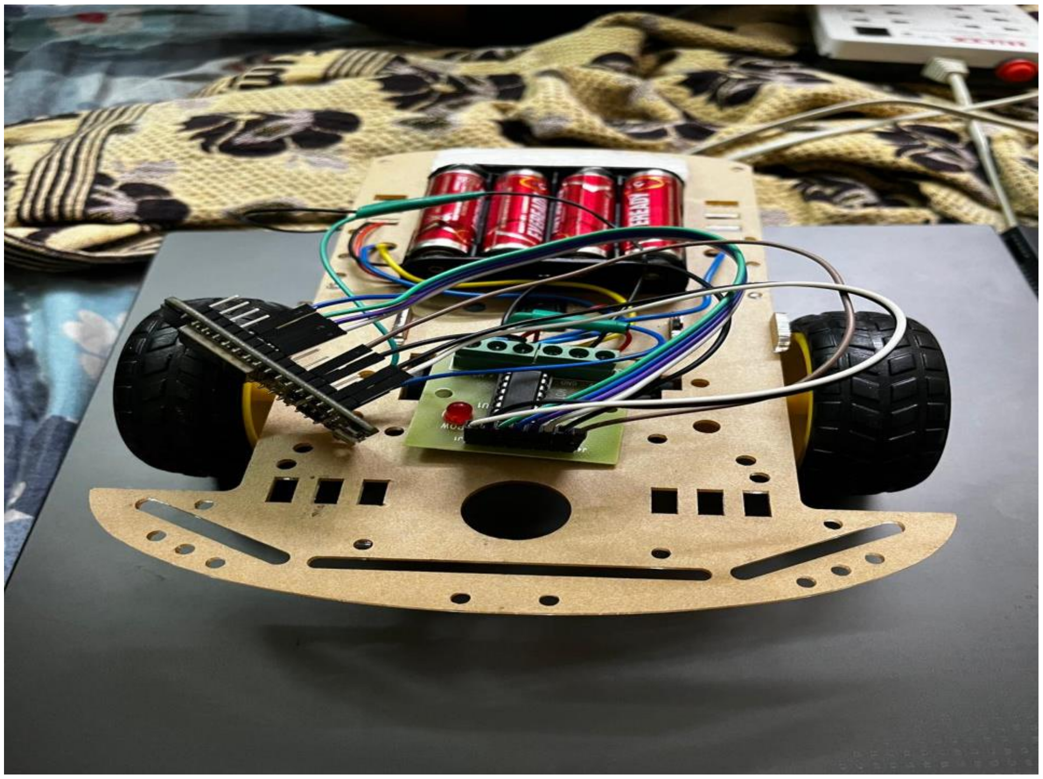


Figure 3

- **Body:** The car body is constructed from wood, likely assembled with glue or fasteners.
- **Wheels:** The car has two wheels, each with an axle for attachment to the body.
- **Battery compartment:** A designated battery compartment is visible on the underside of the car. This compartment likely houses batteries that would supply power to the motor.
- **Wires:** There are visible wires connecting the battery compartment to other parts of the car, likely the motor.
- **ESP32 Board:** A rectangular component with a USB port sits on top of the car. This component is most likely an ESP32 microcontroller board, not an Arduino Uno. The ESP32 is commonly used for projects requiring Wi-Fi or Bluetooth connectivity, which could be used for controlling the car remotely.

CONNECTIONS:

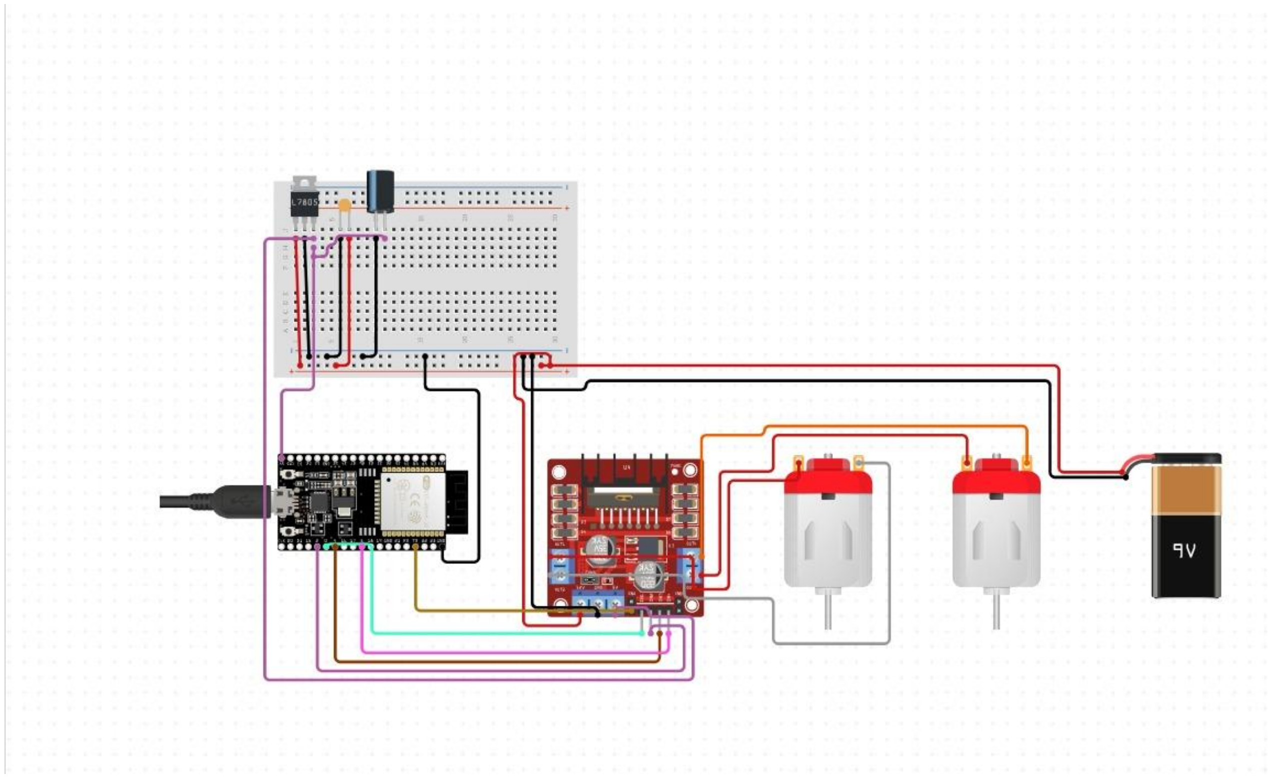


Figure 4 Transmitter connection.

- **Power Supply:** The circuit is powered by a battery (represented by the long and short lines) connected to the ESP32 board's designated power and ground pins (specific pins may vary depending on the ESP32 model).
- **ESP32 Board:** The ESP32 board takes the place of the Arduino Uno. It's a microcontroller board programmed to control the motor driver IC based on user input or sensor readings (not shown in the diagram). The ESP32 might be chosen for its Wi-Fi or Bluetooth capabilities for wireless control.
- **L298N Motor Driver IC:** This chip functions the same way as before, acting as an intermediary between the ESP32 control signals and the motor.

- **Control Inputs (IN1, IN2, IN3, IN4):** These ESP32 pins (specific pins would depend on the wiring scheme) send control signals to the L298N motor driver IC, determining the direction of the motor's rotation.
- **Motor (M1):** The motor is connected to the L298N motor driver IC in the same way as before.
- **Motor Power Supply (Vcc and GND):** The motor receives power from the battery through separate connections to the motor driver IC's "Vcc" (motor supply voltage) and "GND" pins.
- **Enable Pins (EnA, EnB):** These ESP32 pins (not labeled in the image but likely connected) control whether the motor driver outputs power to the motor. By setting these pins high or low, the ESP32 can enable or disable the motor entirely.
- **Bypass Diodes (D1, D2):** These diodes function as explained previously, protecting the circuit from voltage spikes.
- **Ground Connections:** Various components share a common ground connection for proper circuit operation.

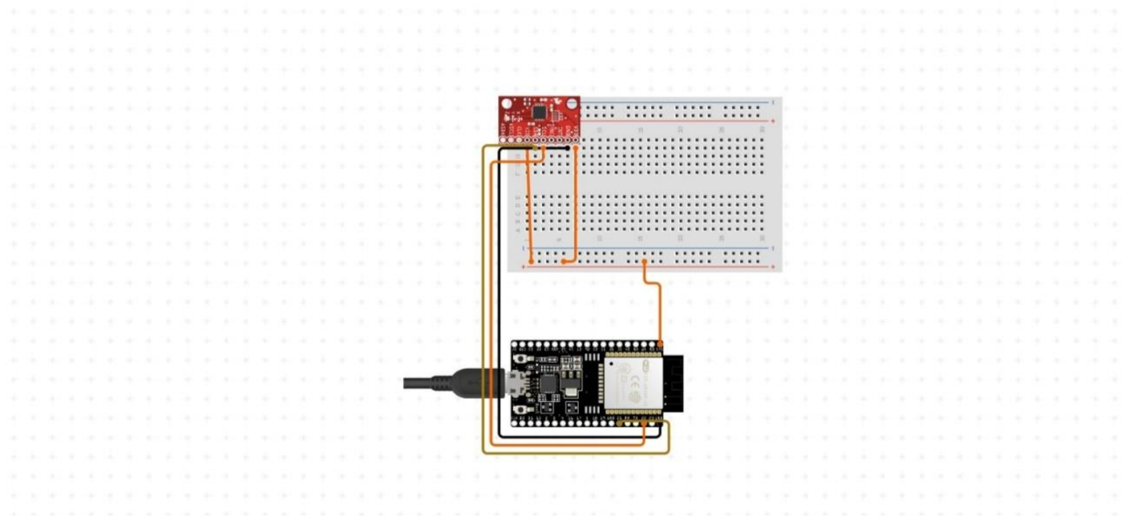


Figure 5

- **ESP32 Board:** The ESP32 board takes the place of the Arduino Uno. It's a microcontroller board programmed to control the motor driver IC based on user input or sensor readings (not shown in the diagram). The ESP32 might be chosen for its Wi-Fi or Bluetooth capabilities for wireless control.

- **Control Inputs (IN1, IN2, IN3, IN4):** These ESP32 pins (specific pins would depend on the wiring scheme) send control signals to the L298N motor driver IC, determining the direction of the motor's rotation.
- **Bypass Diodes (D1, D2):** These diodes function as explained previously, protecting the circuit from voltage spikes.
- **Ground Connections:** Various components share a common ground connection for proper circuit operatio

CHAPTER 4
CONCLUSION AND FUTURE WORK

This project has explored the design and development of a hand gesture controlled car system. By integrating hand gesture recognition with car control mechanisms, the project aimed to revolutionize the driving experience, offering an intuitive and user-friendly interface. The proposed system utilizes cameras or depth sensors to capture hand movements, followed by machine learning algorithms (like CNNs) for real-time gesture recognition. A user-centered HCI design ensures intuitive gesture mapping to essential car controls like acceleration, braking, steering, and activation of features. Safety protocols, including confirmation steps or biometric authentication, are crucial to mitigate accidental gestures.

The project's success hinges on the seamless integration of the hand gesture recognition system with a chosen car platform, enabling real-time car control based on recognized gestures. Comprehensive user testing is vital for evaluating usability, learning curve, and overall user experience. Rigorous safety evaluations will ensure compliance with regulations and prioritize driver and passenger safety.

Looking ahead, several avenues for future work hold immense promise. Refining the hand gesture recognition algorithms through ongoing data acquisition and training will enhance accuracy and robustness in diverse lighting and environmental conditions. Integrating the system with advanced driver-assistance systems (ADAS) could further improve safety and driving experience. Furthermore, exploring the potential for hand gesture control in autonomous vehicles could open doors for a new era of mobility, particularly for individuals with physical limitations.

APPENDIX I

<u>TRANSMITTER.C</u>

```
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>
#include <esp_now.h>
#include <WiFi.h>
Adafruit_MPU6050 mpu;
int commands = 0;
// Global copy of slave
esp_now_peer_info_t slave;
#define CHANNEL 1
#define PRINTSCANRESULTS 0
#define DELETEBEFOREPAIR 0
int mac[6] = {0x40, 0x22, 0xD8, 0x4E, 0xF7, 0xDD}; // 40:22:D8:4E:F7:DD

// Init ESP Now with fallback
void InitESPNow() {
  WiFi.disconnect();
  if (esp_now_init() == ESP_OK) {
    Serial.println("ESPNow Init Success");
  }
  else {
    Serial.println("ESPNow Init Failed");
    // Retry InitESPNow, add a counte and then restart?
    // InitESPNow();
    // or Simply Restart
    ESP.restart();
  }
```

```
}

// Scan for slaves in AP mode
void ScanForSlave() {
 bool slaveFound = 0;
  memset(&slave, 0, sizeof(slave));


  for (int ii = 0; ii < 6; ++ii )
  {
    slave.peer_addr[ii] = (uint8_t) mac[ii];
  }

  slave.channel = CHANNEL; // pick a channel
  slave.encrypt = 0; // no encryption

  slaveFound = 1;



  if (slaveFound)
  {
    Serial.println("Slave Found, processing..");
  } else
  {
    Serial.println("Slave Not Found, trying again.");
  }
}

// Check if the slave is already paired with the master.
// If not, pair the slave with master
bool manageSlave() {
```

```
if (slave.channel == CHANNEL) {
  if (DELETEBEFOREPAIR) {
    deletePeer();
  }

  Serial.print("Slave Status: ");
  // check if the peer exists
  bool exists = esp_now_is_peer_exist(slave.peer_addr);
  if ( exists) {
    // Slave already paired.
    Serial.println("Already Paired");
    return true;
  } else {
    // Slave not paired, attempt pair
    esp_err_t addStatus = esp_now_add_peer(&slave);
    if (addStatus == ESP_OK) {
      // Pair success
      Serial.println("Pair success");
      return true;
    } else if (addStatus == ESP_ERR_ESPNOW_NOT_INIT) {
      // How did we get so far!!
      Serial.println("ESPNOW Not Init");
      return false;
    } else if (addStatus == ESP_ERR_ESPNOW_ARG) {
      Serial.println("Invalid Argument");
      return false;
    } else if (addStatus == ESP_ERR_ESPNOW_FULL) {
      Serial.println("Peer list full");
      return false;
    } else if (addStatus == ESP_ERR_ESPNOW_NO_MEM) {
      Serial.println("Out of memory");
      return false;
      } else if (addStatus == ESP_ERR_ESPNOW_EXIST) {
        Serial.println("Peer Exists");
        return true;
      } else {
        Serial.println("Not sure what happened");
        return false;
      }
    }
  } else {
    // No slave found to process
    Serial.println("No Slave found to process");
    return false;
  }
}

void deletePeer() {
  esp_err_t delStatus = esp_now_del_peer(slave.peer_addr);
  Serial.print("Slave Delete Status: ");
  if (delStatus == ESP_OK) {
    // Delete success
    Serial.println("Success");
  } else if (delStatus == ESP_ERR_ESPNOW_NOT_INIT) {
    // How did we get so far!!
    Serial.println("ESPNOW Not Init");
  } else if (delStatus == ESP_ERR_ESPNOW_ARG) {
    Serial.println("Invalid Argument");
  } else if (delStatus == ESP_ERR_ESPNOW_NOT_FOUND) {
    Serial.println("Peer not found.");
  } else {
    Serial.println("Not sure what happened");
  }
}
```

```
uint8_t data = 0;
// send data
void sendData() {
  const uint8_t *peer_addr = slave.peer_addr;
  Serial.print("Sending: "); Serial.println(data);
  esp_err_t result = esp_now_send(peer_addr, &data, sizeof(data));
  Serial.print("Send Status: ");
  if (result == ESP_OK) {
    Serial.println("Success");
  } else if (result == ESP_ERR_ESPNOW_NOT_INIT) {
    // How did we get so far!!
    Serial.println("ESPNOW not Init.");
  } else if (result == ESP_ERR_ESPNOW_ARG) {
    Serial.println("Invalid Argument");
  } else if (result == ESP_ERR_ESPNOW_INTERNAL) {
    Serial.println("Internal Error");
  } else if (result == ESP_ERR_ESPNOW_NO_MEM) {
    Serial.println("ESP_ERR_ESPNOW_NO_MEM");
  } else if (result == ESP_ERR_ESPNOW_NOT_FOUND) {
    Serial.println("Peer not found.");
  } else {
    Serial.println("Not sure what happened");
  }
}

// callback when data is sent from Master to Slave
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  char macStr[18];
  snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",
           mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3], mac_addr[4],
mac_addr[5]);
```

```
  Serial.print("Last Packet Sent to: "); Serial.println(macStr);
  Serial.print("Last Packet Send Status: "); Serial.println(status ==
ESP_NOW_SEND_SUCCESS ? "Delivery Success" : "Delivery Fail");
}

void setup(void) {

  Serial.begin(115200);
  while (!Serial)
    delay(10); // will pause Zero, Leonardo, etc until serial console opens

  Serial.println("Adafruit MPU6050 test!");

  // Try to initialize!
  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1) {
      delay(10);
    }
  }
  Serial.println("MPU6050 Found!");

  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
  Serial.print("Accelerometer range set to: ");
  switch (mpu.getAccelerometerRange()) {
  case MPU6050_RANGE_2_G:
    Serial.println("+-2G");
    break;
  case MPU6050_RANGE_4_G:
    Serial.println("+-4G");
    break;
  case MPU6050_RANGE_8_G:
```

```
  Serial.println("+-8G");
  break;
case MPU6050_RANGE_16_G:
  Serial.println("+-16G");
  break;
}
mpu.setGyroRange(MPU6050_RANGE_500_DEG);
Serial.print("Gyro range set to: ");
switch (mpu.getGyroRange()) {
case MPU6050_RANGE_250_DEG:
  Serial.println("+- 250 deg/s");
  break;
case MPU6050_RANGE_500_DEG:
  Serial.println("+- 500 deg/s");
  break;
case MPU6050_RANGE_1000_DEG:
  Serial.println("+- 1000 deg/s");
  break;
case MPU6050_RANGE_2000_DEG:
  Serial.println("+- 2000 deg/s");
  break;
}

mpu.setFilterBandwidth(MPU6050_BAND_5_HZ);
Serial.print("Filter bandwidth set to: ");
switch (mpu.getFilterBandwidth()) {
case MPU6050_BAND_260_HZ:
  Serial.println("260 Hz");
  break;
case MPU6050_BAND_184_HZ:
  Serial.println("184 Hz");
  break;
  case MPU6050_BAND_94_HZ:
    Serial.println("94 Hz");
    break;
  case MPU6050_BAND_44_HZ:
    Serial.println("44 Hz");
    break;
  case MPU6050_BAND_21_HZ:
    Serial.println("21 Hz");
    break;
  case MPU6050_BAND_10_HZ:
    Serial.println("10 Hz");
    break;
  case MPU6050_BAND_5_HZ:
    Serial.println("5 Hz");
    break;
  }

  Serial.println("");
  delay(100);
   WiFi.mode(WIFI_STA);
  Serial.println("ESPNow/Basic/Master Example");
  // This is the mac address of the Master in Station Mode
  Serial.print("STA MAC: "); Serial.println(WiFi.macAddress());
  // Init ESPNow with a fallback logic
  InitESPNow();
  // Once ESPNow is successfully Init, we will register for Send CB to
  // get the status of Trasnmitted packet
  esp_now_register_send_cb(OnDataSent);
  ScanForSlave();
}

void loop() {
  /* Get new sensor events with the readings */
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);
  commands = a.acceleration.x;

  data = commands;

  // If Slave is found, it would be populate in slave variable
  // We will check if slave is defined and then we proceed further
  if (slave.channel == CHANNEL) { // check if slave channel is defined
    // slave is defined
    // Add slave as peer if it has not been added already
    bool isPaired = manageSlave();
    if (isPaired) {
      // pair success or already paired
      // Send data to device
      sendData();
    } else {
      // slave pair failed
      Serial.println("Slave pair failed!");
    }
  }
  else {
    // No slave found to process
  }

  // wait for 3seconds to run the logic again
  delay(1);
  //Serial.println("");

}
```

RECEIVER.C

```
#include <esp_now.h>
#include <WiFi.h>

#define CHANNEL 1
int m1 = 19;
int m2 = 21;
int m3 = 22;
int m4 = 23;
int en1 = 26;
int en2= 27;

// Init ESP Now with fallback
void InitESPNow() {
  WiFi.disconnect();
  if (esp_now_init() == ESP_OK) {
    Serial.println("ESPNow Init Success");
  }
  else {
    Serial.println("ESPNow Init Failed");
    // Retry InitESPNow, add a counte and then restart?
    // InitESPNow();
    // or Simply Restart
    ESP.restart();
  }
}

// config AP SSID
void configDeviceAP() {
  const char *SSID = "Slave_1";
  bool result = WiFi.softAP(SSID, "Slave_1_Password", CHANNEL, 0);
  if (!result) {
    Serial.println("AP Config failed.");
  } else {
    Serial.println("AP Config Success. Broadcasting with AP: " + String(SSID));
    Serial.print("AP CHANNEL "); Serial.println(WiFi.channel());
  }
}
```

```
void forward() {
  digitalWrite(m1, HIGH);
  digitalWrite(m2, LOW);
  digitalWrite(m4, HIGH);
  digitalWrite(m3, LOW);
}

void backward() {
  digitalWrite(m1, LOW);
  digitalWrite(m2, HIGH);
  digitalWrite(m4, LOW);
  digitalWrite(m3, HIGH);
}

void left() {
  digitalWrite(m1, HIGH);
  digitalWrite(m4, LOW);
  digitalWrite(m2, LOW);
  digitalWrite(m3, HIGH);
}

void right() {
  digitalWrite(m1, LOW);
  digitalWrite(m4, HIGH);
  digitalWrite(m2, HIGH);
  digitalWrite(m3, LOW);
}

void stopd() {
  digitalWrite(m1, LOW);
  digitalWrite(m4, LOW);
  digitalWrite(m2, LOW);
  digitalWrite(m3, LOW);
}

void setup() {

  Serial.begin(115200);
  Serial.println("ESPNow/Basic/Slave Example");
  //Set device in AP mode to begin with
  WiFi.mode(WIFI_AP);
  // configure device AP mode
  configDeviceAP();
  // This is the mac address of the Slave in AP Mode
  Serial.print("AP MAC: "); Serial.println(WiFi.softAPmacAddress());
  // Init ESPNow with a fallback logic

  InitESPNow();
  // Once ESPNow is successfully Init, we will register for recv CB to
  // get recv packer info.
  pinMode(m1,OUTPUT);
pinMode(m2,OUTPUT);
pinMode(m3,OUTPUT);
pinMode(m4,OUTPUT);
pinMode(en1,OUTPUT);
pinMode(en2,OUTPUT);
digitalWrite(en1,HIGH);
digitalWrite(en2,HIGH);
  esp_now_register_recv_cb(OnDataRecv);
}

// callback when data is recv from Master
void OnDataRecv(const uint8_t *mac_addr, const uint8_t *data, int data_len) {
  char macStr[18];
  snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",
         mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3], mac_addr[4],
mac_addr[5]);
  Serial.print("Last Packet Recv from: "); Serial.println(macStr);
  Serial.print("Last Packet Recv Data: "); Serial.println(*data);
  Serial.println("");
  if(*data == 5){
    Serial.println("R");
    right();
  }
  else if(*data == 2){
    Serial.println("F");
    forward();
  }
  else if(*data == 3){
    Serial.println("B");
    backward();
  }
  else if(*data == 4){
    Serial.println("L");
    left();
  }
  else{

    Serial.println("Stop");
    stopd();

  }
}

void loop() {
  // Chill
```

REFERENCES

1. Multimodal hand gesture classification for the human–car interaction A D'Eusanio, A Simoni, S Pini, G Borghi, R Vezzani… - Informatics, 2020 - mdpi.com
2. A hand-gesture-based control interface for a car-robot XH Wu, MC Su, PC Wang - 2010 IEEE/RSJ International …, 2010 - ieeexplore.ieee.org 2019.

3. Standardization of the in-car gesture interaction space A Riener, A Ferscha, F Bachmair, P Hagmüller… - Proceedings of the 5th …, 2013 - dl.acm.org

4. Multi-sensor system for driver's hand-gesture recognition P Molchanov, S Gupta, K Kim… - 2015 11th IEEE …, 2015 - ieeexplore.ieee.org

5. Motion-path based in car gesture control of the multimedia devices ASMM Rahman, J Saboune, A El Saddik - Proceedings of the first ACM …, 2011 - dl.acm.org
6. Hand Gesture Control Car R Shah, V Deshmukh, V Kulkarni, S Mulay… - Proceedings to ICSITS … - academia.edu
7. Hand gesture recognition system for in-car device control based on infrared array sensor S Tateno, Y Zhu, F Meng - 2019 58th Annual conference of the …, 2019 - ieeexplore.ieee.org

8. User evaluation of hand gestures for designing an intelligent in-vehicle interface H Jahani, HJ Alyamani, M Kavakli, A Dey… - Designing the Digital …, 2017 - Springer

9. Real-time hand gesture-based interaction with objects in 3D virtual environments JO Kim, M Kim, KH Yoo - International Journal of Multimedia and …, 2013 - gvpress.com

10. An automated robot-car control system with hand-gestures and mobile application using arduino S Ullah, Z Mumtaz, S Liu, M Abubaqr, A Mahboob… - 2019 - preprints.org

11. Givs: fine-grained gesture control for mobile devices in driving environments L Jiang, M Xia, X Liu, F Bai - IEEE Access, 2020 - ieeexplore.ieee.org.

12. Hand gestures for the human-car interaction: The briareo dataset
F Manganaro, S Pini, G Borghi, R Vezzani… - Image Analysis and …, 2019 - Springer.

13. B Verma, A Choudhary - IET Intelligent Transport Systems, 2018
14. Two-handed gesture-based car styling in a virtual environment C Hummels, AE Paalder, CJ Overbeeke… - … , Florence, Italy.
15. [CITATION] Hand gesture-controlled robot car B KUMAR, A REDDY, A GOWDA