GESTURE CONTROLLED ROBOT USING ESPNOW

Project By :

Nagiri Sai Akshaya
BTech(IoT)
ACE Engineering College

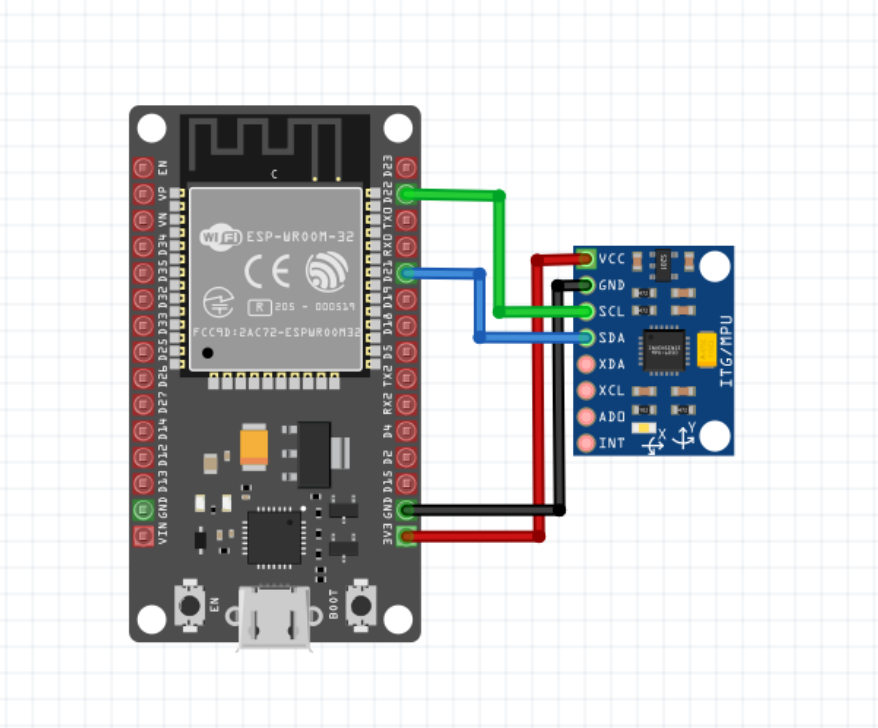
Goal: The goal is to build a gesture-controlled robot using ESP-NOW communication, where the car moves in different directions based on control signals received from transmitter, allowing wireless and responsive control.

Required Materials: ESP32-2, L298N Motor Driver, 4-wheel robotic chassis, 18650 Li-ion battery - 2, MPU6050 , Half Breadboard, 5V battery supply, F-F, M-F,M-M Jumper wires.

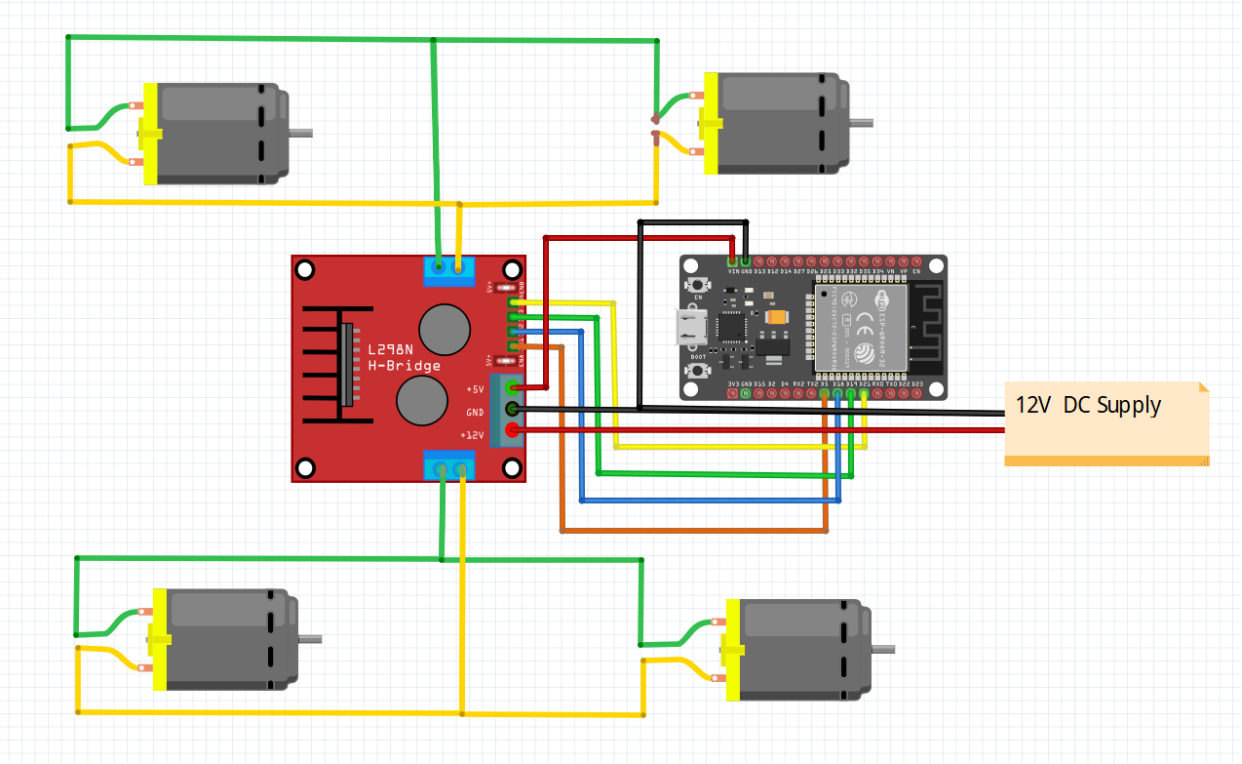
Software Requirements: Arduino IDE.

Wiring Diagram:

Transmitter:



Receiver:



Procedure:

- Firstly upload the mac address code in ESP32 board of Receiver and retrive the mac address.
- Then place receiver mac address in transmitter code and upload into the transmitter ESP32 and also upload receiver code in receiver ESP32.
- Make sure to connect Transmitter and Receiver ESP32 to the WIFI STA.
- After connecting the wires according to the diagram. Powerup the setup
- Then according to the hand movements we give through the transmitter the car moves its directions.

Troubleshooting points:

• Make sure the ESP-NOW communication between the transmitter and receiver is properly set up. Both devices should be initialized with correct MAC addresses.
• Verify that the motor control pins (IN1, IN2, and En) are correctly connected to the corresponding GPIO pins on the ESP32.
• Ensure the PWM channels are properly configured and attached to the correct motor enable pins for speed control.
• Check the motor driver connections to ensure no loose wires, as improper connections can lead to inconsistent motor performance.

• Ensure the motor speeds and directions are correctly assigned in the code, particularly during diagonal movements and turns.
• Monitor signal timeout: If no signal is received for a long period, the car should stop. Ensure this timeout is functioning as expected.
• Make sure WiFi mode is set to STA (Station Mode) for ESP-NOW to work correctly, as using another mode can disrupt communication.
• Check for power supply issues like voltage drops, as they can affect motor performance and ESP32 behavior. Fully charge batteries before use.
• Verify that the received data values (x, y, z) from the transmitter are within expected ranges for the car to move in the right direction.

Output :

- The car is controlled by hand gestures and moves in response to signals received from the transmitter.
- Based on the tilt direction, the car moves forward, backward, left, or right.
- The transmitter and receiver need to stay stable for 1 minute to establish a proper connection.
- Tilting the transmitter forward makes the car move forward, while tilting it backward makes the car move in reverse.
- Tilting the transmitter sideways causes the car to turn left or right accordingly.

Future Enhancements:

Ultrasonic Sensor and Buzzer for Obstacle Detection:

- Integrate an ultrasonic sensor to detect obstacles in the car’s path.
- Trigger a buzzer alarm when the car encounters an obstacle while being controlled through gestures via the transmitter.

LCD Display for Speed and Direction Monitoring:

- Add an LCD display to show real-time information on the car's speed and direction.
- Implement a buzzer alarm to trigger if the car’s speed exceeds a predefined limit, ensuring safety.

Code:

Receiver Mac Address :

```
#include "WiFi.h"

void setup() {
  Serial.begin(115200);
  // Initialize WiFi
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  delay(100);

  // Get and print the MAC address
  Serial.print("MAC address: ");
  Serial.println(WiFi.macAddress());
}

void loop() {
}
```

Receiver Code:

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

#define IN1 5
#define IN2 18
#define IN3 19
#define IN4 21
#define CHANNEL 1

bool front = false;
bool back = false;
bool left = false;
bool right = false;
bool carIsStopped = true;

// Offset values for calibration
const int xOffset = 127;
const int yOffset = 127;

// Timeout settings
unsigned long lastReceivedTime = 0;
const unsigned long timeout = 200;  // Timeout period in milliseconds

typedef struct {
  byte xAxisValue;
```

```
  byte yAxisValue;
} PacketData;

PacketData data;

void InitESPNow() {
  WiFi.disconnect();
  if (esp_now_init() == ESP_OK) {
    Serial.println("ESPNow Init Success");
  } else {
    Serial.println("ESPNow Init Failed");
    ESP.restart();
  }
}

void configDeviceAP() {
  String ssid = "slave_" + String((uint32_t)ESP.getEfuseMac(), HEX);
  WiFi.softAP(ssid.c_str(), "Slave_Password", CHANNEL, 0);
  Serial.print("AP Config Success. Broadcasting with SSID: ");
  Serial.println(ssid);
}
```

Transmitter code:

```
#include <esp_now.h>
#include <WiFi.h>
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"

MPU6050 mpu;
bool dmpReady = false;
uint8_t devStatus;
uint8_t fifoBuffer[64];
Quaternion q;
VectorFloat gravity;
float ypr[3];
bool espConnected = false;

struct PacketData {
  byte xAxisValue;
  byte yAxisValue;
};
PacketData data;

esp_now_peer_info_t slave;
```

```
#define CHANNEL 1

void InitESPNow() {
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  if (esp_now_init() == ESP_OK) {
    Serial.println("ESP-NOW Init Success");
  } else {
    Serial.println("ESP-NOW Init Failed");
    ESP.restart();
  }
}
```