Spy Remote Control Car Using ESPNOW Protocol

A

REPORT

ON

PROJECT OF
EMBEDDED AND IOT SYSTEM DESIGN
(3EL04)

Bachelor of Technology
(Electronics Engineering)

Submitted by:

ROHIT VIJAY GUPTA
(21EL044)

Under the guidance of:

Prof . Anita Bhatt

Birla Vishvakarma Mahavidyalaya
Engineering Collage (An Autonomous Institution)
Vallabh Vidyanagar - 388 120
Affiliated to Gujarat Technological University

CERTIFICATE

This is to certify that the Mini Project (2EL31) entitled **"Spy Remote Control Car Using ESPNOW Protocol"** has been carried out by ***Rohit Gupta*** in the 5th semester under my guidance in Electronics Engineering, Birla Vishvakarma Mahavidyalaya, Vallabh Vidyanagar, during the academic year 2022- 23.

Guided by: **Prof . Anita Bhatt**

ACKNOWLEDGMENT

It is indeed pleasure and pride moment for us to undertake this Mini Project entitled **“Spy Remote Control Car Using ESPNOW Protocol”.**We would like to thank **Prof. Anita Bhatt**, and **Dr. T.D.Pawar**, Head of Electronics department, Birla Vishvakarma Mahavidylaya, Vallabh Vidyanagar, Anand to inspire us to undertake this project and also to support during this project.

Table of Contents

1. Abstract

This project involves building a remote-controlled spy RC car with a live camera feed. The ESP32 acts as a transmitter, while the ESP8266 functions as the receiver. Communication is established using the ESP-NOW protocol. The L298N motor driver is used to control four motors, including two servo motors for steering and camera control. The ESP32-CAM provides a live camera feed for real-time surveillance.

2. Introduction

The spy RC car is designed for surveillance and remote monitoring. The ESP32 remote controller sends commands to the ESP8266 receiver via the ESP-NOW protocol. The receiver controls the movement of the RC car, including two servo motors for steering and camera orientation, using the L298N motor driver. The ESP32-CAM captures and streams live video Using Wifi Protocal.

- **List of component** **Price(rs)**

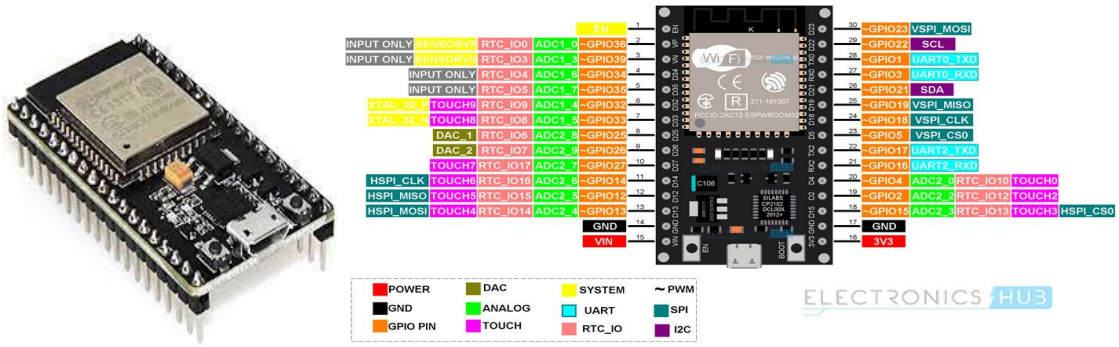
1.	ESP32	450
2.	ESP8266	200
3.	ESP32 CAM	475
4.	2 x Joystick	80
5.	Motor driver	140
6.	2 x servo	200
7.	Servo mount	80
8.	2 x 18650 battery	200
9.	Battery slot	50
10.	2 x 7805	24
11.	Screw Terminal	20
12.	Pcb board	50
13.	Dc motor with gear	400
14.	Connector wire	50
15.	Miscellaneous cost	81

Total cost = 2500

3. Hardware Components

3.1. ESP32 (Transmitter)

- ESP32 development board



Prize = Rs450

- The ESP32 is a popular microcontroller and system-on-chip (SoC) that is widely used for a variety of IoT (Internet of Things) and embedded projects. It has numerous pins for various functions. However, there might be some confusion regarding your reference to "38 pin," as the ESP32 typically has 38 or 40 GPIO (General Purpose Input/Output) pins.

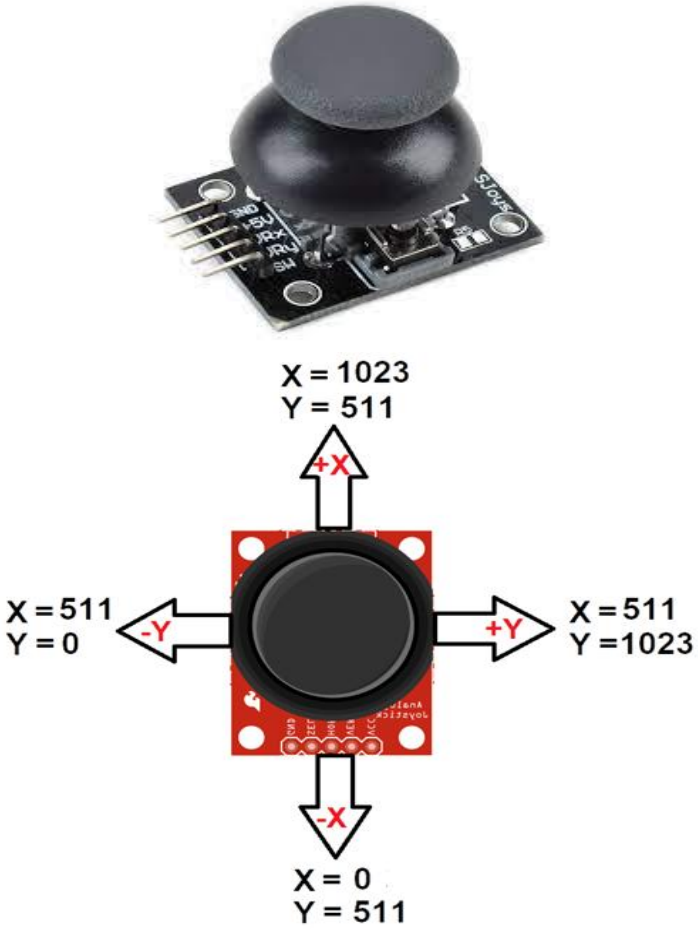
1. Pin Number: The ESP32 has GPIO pins numbered from 0 to 39. So, when you mention "38 pin," you are referring to a specific GPIO pin.

2. General Purpose Input/Output (GPIO): These pins can be configured as either inputs or outputs, allowing you to connect and control various external devices, such as sensors, LEDs, displays.

3. Pin Functions: The specific functions and capabilities of a GPIO pin may vary depending on the pin and its capabilities. Some pins have special functions, such as PWM (Pulse Width Modulation), analog input, UART (serial communication), I2C, SPI, etc.

4. Pin Configuration: To use pin 38 for a specific purpose, you would typically configure it in your code to serve that function. For example, you might configure it as a digital output to control an LED or as an input to read a sensor value.

- **Joystick module for control**



- A joystick is an input device commonly used in electronic devices and gaming controllers to provide analog control over movement or selection. Here are two lines to describe a joystick:

1. Analog Input Control: Joysticks allow users to provide precise analog input control by moving a lever or stick in different directions. This analog input can be used to control the movement of objects, navigate menus, or adjust settings.

2. Versatile Application: Joysticks are versatile input devices used in various applications, including gaming, robotics, aviation, and industrial control systems, providing a natural and intuitive way to interact with electronic devices and machinery.

- **Concept of remote control**

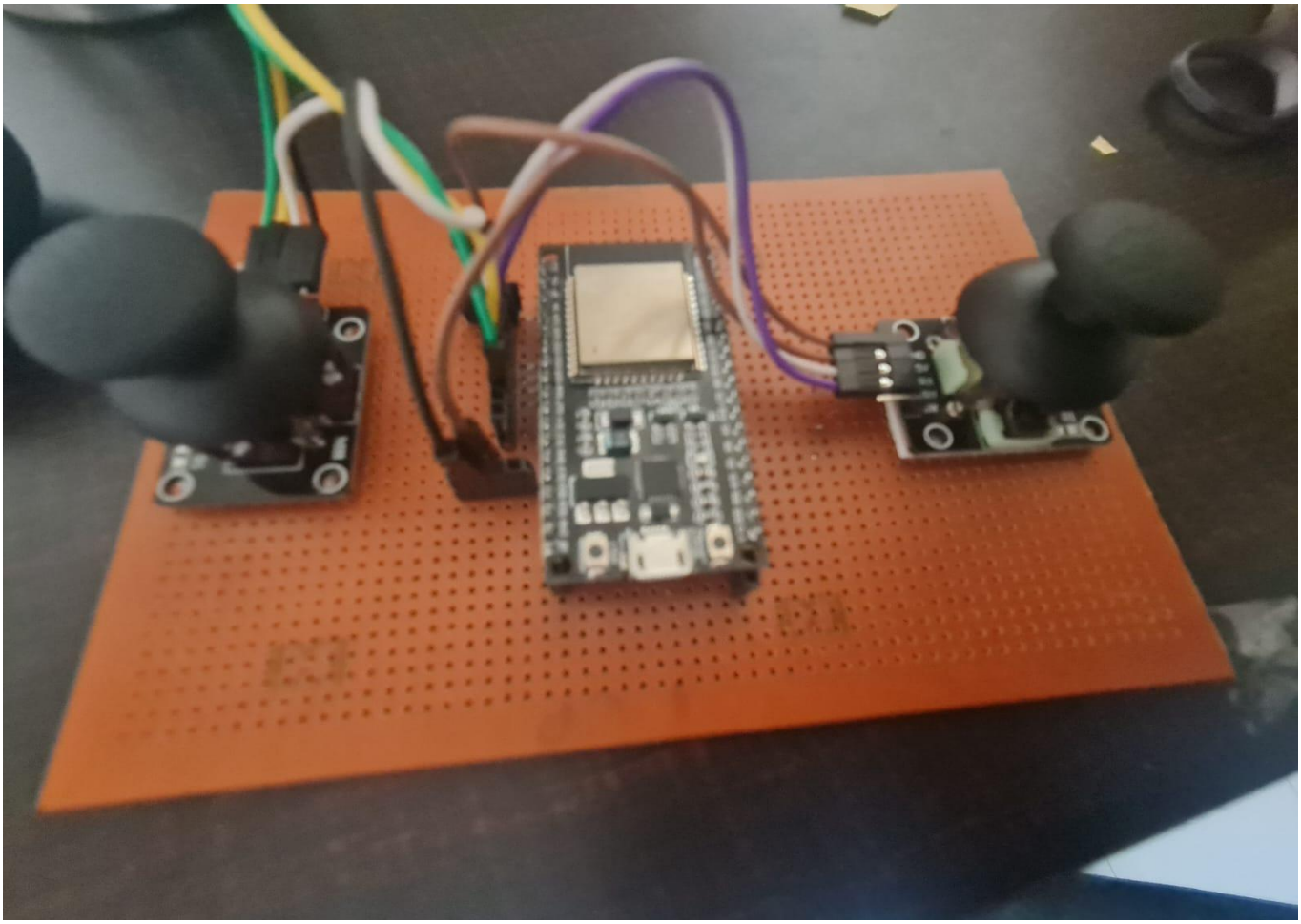
Here in esp 32,joystick give analog input between from 0 to 4096 whenever joystick move from center.

So to we mapped the 0 to 4096 into 0 to 1024,

Even joystick has In-built switch, so we use switch as input and transmit the value to recevier.

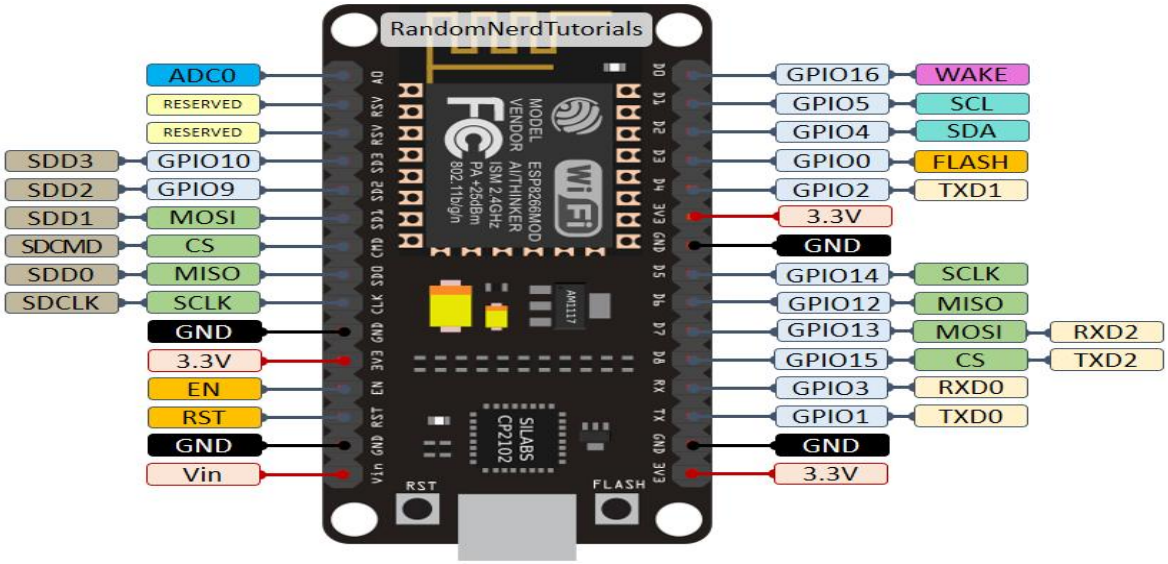
Left Joystick is used for car control and Right joystick is used for servo control

Image of transmitter :-



3.2. ESP8266 (Receiver)

- ESP8266 development board



The ESP8266 is a popular and widely used microcontroller and Wi-Fi module. Here's some information about it:

1. Microcontroller: The ESP8266 is a low-cost, low-power, and highly integrated microcontroller with built-in Wi-Fi connectivity. It is designed for embedded applications and IoT (Internet of Things) projects.

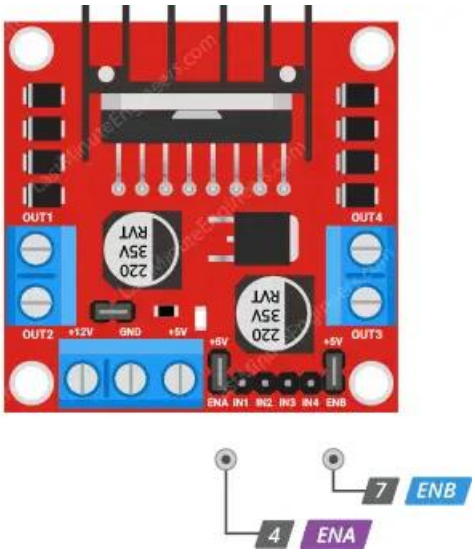
2. Wi-Fi Connectivity: One of the key features of the ESP8266 is its ability to connect to Wi-Fi networks, making it suitable for projects that require wireless communication and internet connectivity. It can be used to send and receive data over Wi-Fi.

3. Programming: The ESP8266 can be programmed using various development platforms and programming languages. The most common way to program it is using the Arduino IDE with the ESP8266 core, which provides an easy-to-use environment for writing and uploading code.

4. GPIO Pins: The ESP8266 typically comes with a number of GPIO (General Purpose Input/Output) pins that can be used to interface with sensors, displays, and other external devices. The number of GPIO pins can vary depending on the specific variant of the ESP8266.

5. Memory and Processing Power: The ESP8266 comes with its own onboard memory and processing power, making it capable of running applications and handling data.

- L298N motor driver



➢ The L298N is a popular dual H-bridge motor driver integrated circuit (IC) used in a wide range of robotics and motor control applications. Here are some key points about the L298N:

1. H-Bridge Motor Driver: The L298N is designed to control the direction and speed of two DC motors or a single stepper motor. It uses an H-bridge configuration, which allows you to control the polarity of the voltage applied to the motor terminals, enabling forward, reverse, and braking actions.

2. Dual Motor Control: It has two separate H-bridge circuits, which means it can control two DC motors independently or a single stepper motor with two coils.

3. Input Voltage: The L298N typically works with a wide range of input voltages, making it suitable for various power supply setups. It can handle voltages ranging from 7V to 35V.

- Two servo motors for steering and camera control

- A servo, short for "servomechanism" or "servomotor," is a type of rotary or linear actuator that provides precise control of angular or linear position, velocity, and acceleration. Servos are widely used in various applications, including robotics, remote control systems, automation, and model hobbyist projects. Here are some key points about servos:

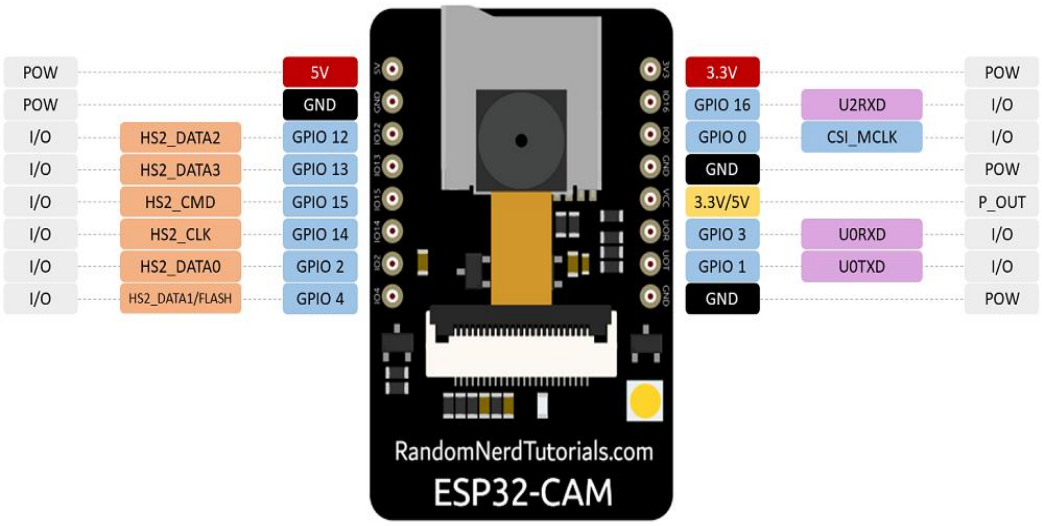
1. Control: Servos are controlled by sending electrical signals, typically in the form of pulses, to the servo's control input. The width and frequency of these pulses determine the position of the servo's output shaft.

2. Feedback System: Most servos incorporate a feedback mechanism, such as a potentiometer or an encoder, which allows the servo to continually compare the actual position of the output shaft with the desired position. This feedback system helps the servo to make necessary adjustments to achieve and maintain the desired position accurately.

3. Position Control: Servos excel at maintaining a specific position. When the control signal is adjusted, the servo will move its output shaft to the new position and hold it there. This makes them ideal for tasks that require accurate positioning.

3.5. ESP32-CAM (Camera Module)

- ESP32-CAM development board



The ESP32-CAM is a popular development board that combines the ESP32 microcontroller with a camera module. This versatile board is used for various IoT and image processing projects. Here are some key features and information about the ESP32-CAM:

1. ESP32 Microcontroller: The ESP32-CAM is built around the ESP32 microcontroller, which is a powerful and versatile microcontroller with Wi-Fi and Bluetooth capabilities. This allows the board to connect to the internet and communicate wirelessly with other devices.

2. Camera Module: The ESP32-CAM board features a camera module that can capture both still images and video. It typically comes with a 2MP OV2640 camera, which is capable of capturing decent-quality images and video.

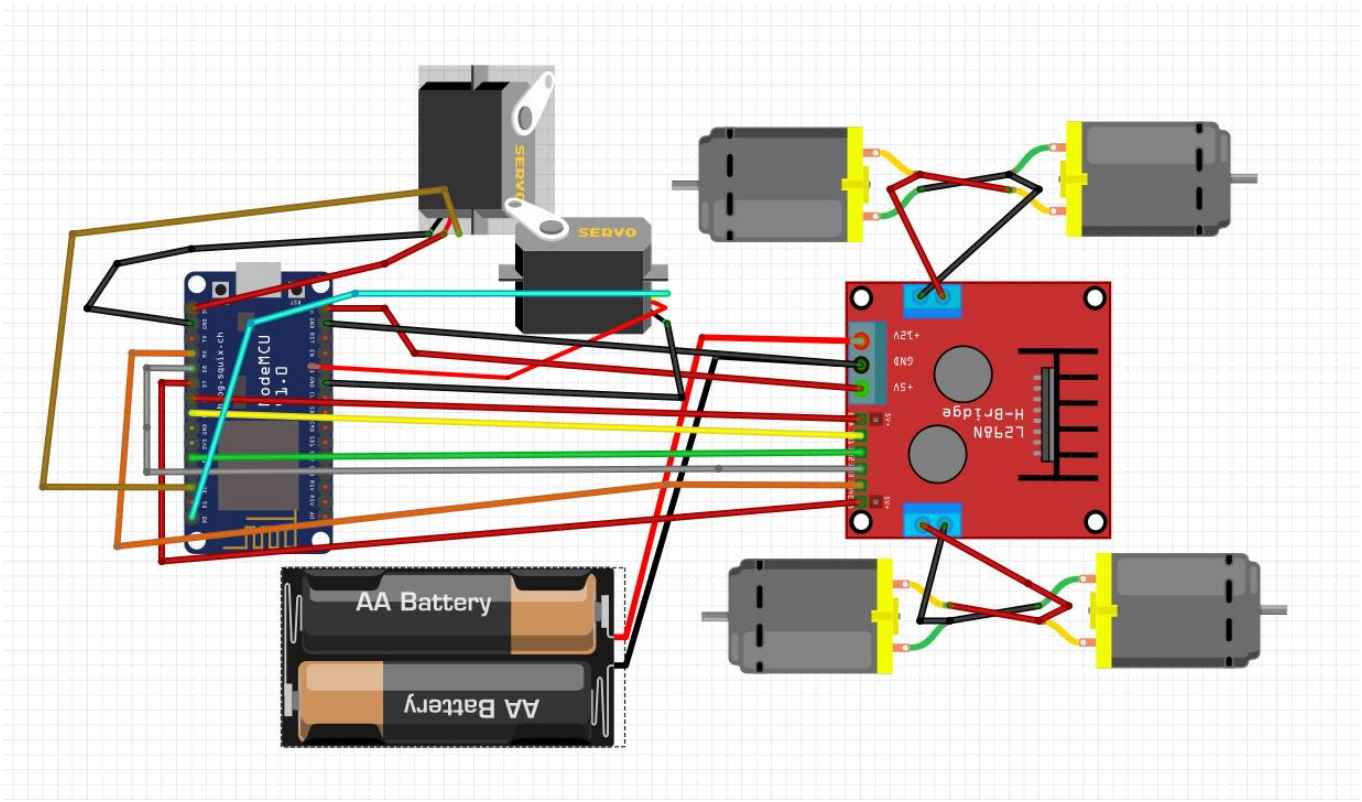
3.6. Four DC motors with gear and four wheel

4. System Architecture

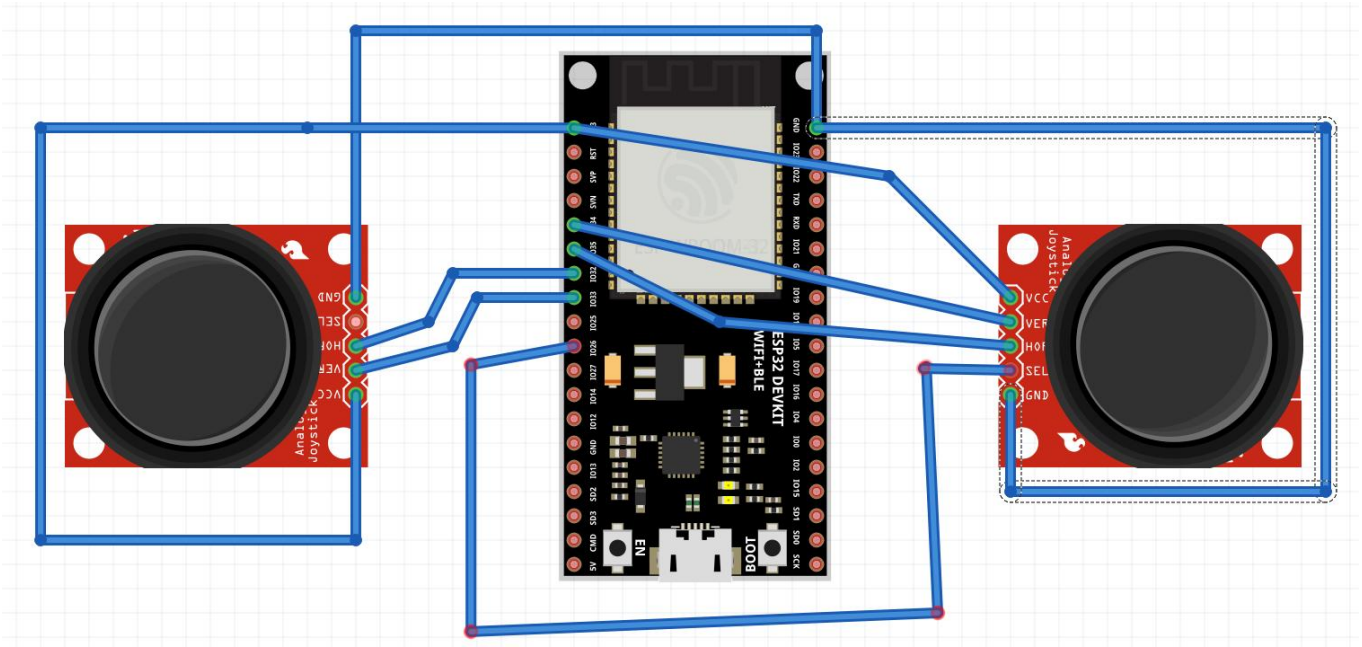
The ESP32 (transmitter) communicates with the ESP8266 (receiver) via the ESP-NOW protocol. The ESP8266 controls the motors using the L298N motor driver, and the servo motors for steering and camera control. The ESP32-CAM provides a live camera feed to the transmitter.

5. Circuit Diagram

- **Recevier diagram**



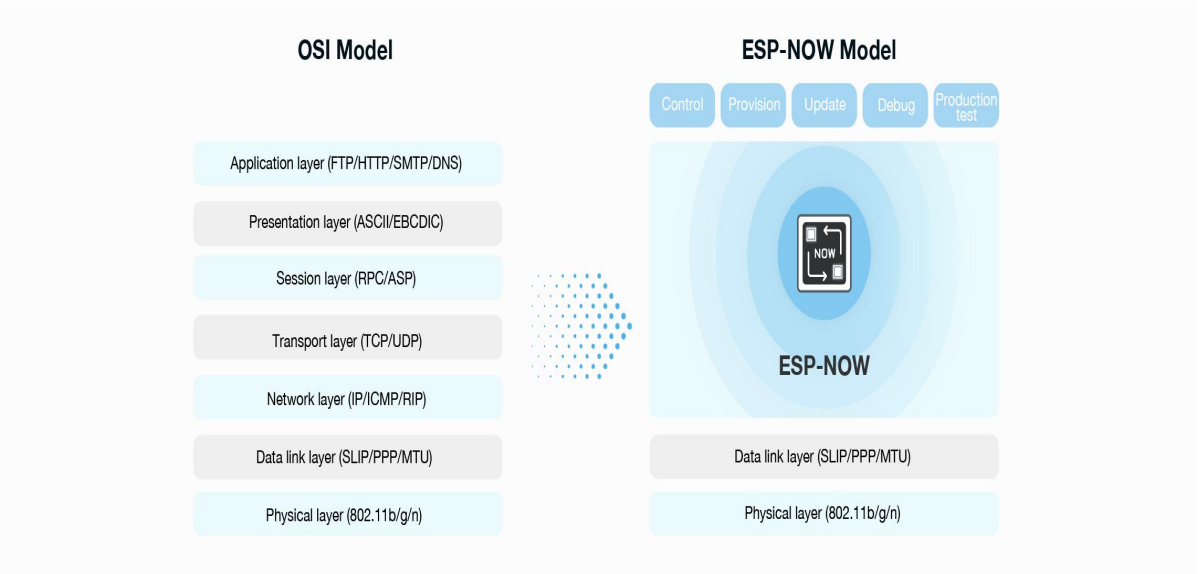
- **Transmitter diagram**



6. Software Components

- 6.1. ESP-NOW Protocol

The ESP-NOW protocol is used for communication between the transmitter and receiver. It allows for low-latency and robust communication.



➢ ESP-NOW supports the following features:

- Encrypted and unencrypted unicast communication;
- Mixed encrypted and unencrypted peer devices;
- **Up to 250-byte** payload can be carried;
- Sending callback function that can be set to inform the application layer of transmission success or failure.

➢ Each ESP32 has a unique MAC Address and that's how we identify each board to send data to it using ESP-NOW (learn how to Get and Change the ESP32 MAC Address).

- Code for mac Address

```
#include "WiFi.h"

void setup(){
  Serial.begin(115200);
  WiFi.mode(WIFI_MODE_STA);
  Serial.println(WiFi.macAddress());
}

void loop(){

}
```

Output :

```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1216
ho 0 tail 12 room 4
load:0x40078000,len:9720
ho 0 tail 12 room 4
load:0x40080400,len:6352
entry 0x400806b8
30:AE:A4:07:0D:64
```

6.2. Programming ESP32 (Transmitter)

- Code for transitter:-

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

int x_motor_key = 32;
int y_motor_key = 33;
int x_servo_key = 34;
int y_servo_key = 35;

uint8_t broadcastAddress1[] = {0x24, 0xA1, 0x60, 0x2D, 0xCA, 0x6D};

String success;

typedef struct struct_message {

    int a;
    int b;
    int c;
    int d;

} struct_message;
struct_message datatosend;

esp_now_peer_info_t peerInfo;
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t
status) {
  char macStr[18];
  Serial.print("Delivery Status of: ");
  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(macStr);
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? " Send
Successfully" : " Fail to send");
  if (status ==0){
    success = "Delivery Success :)";
  }
  else{
    success = "Delivery Fail :(";
  }
}
```

```
void setup() {
  Serial.begin(9600);
 pinMode ( x_motor_key, INPUT) ;
 pinMode ( y_motor_key, INPUT) ;
 pinMode (x_servo_key, INPUT) ;
 pinMode (y_servo_key, INPUT) ;
  WiFi.mode(WIFI_STA);

  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
  esp_now_register_send_cb(OnDataSent);
   peerInfo.channel = 0;
  peerInfo.encrypt = false;

  memcpy(peerInfo.peer_addr, broadcastAddress1, 6);
  if (esp_now_add_peer(&peerInfo) != ESP_OK){
    Serial.println("Failed to add peer");
    return;
  }

}

void loop() {
  struct_message datatosend;
int x_motor_pos = analogRead ( x_motor_key) ;
int y_motor_pos = analogRead ( y_motor_key) ;

int x_servo_pos = analogRead ( x_servo_key) ;
int y_servo_pos = analogRead ( y_servo_key) ;

   datatosend.a =  map(x_motor_pos, 0, 4095, 0, 1993);
   datatosend.b =  map(y_motor_pos, 0, 4095, 0, 1993);
   datatosend.c =  map(x_servo_pos, 0, 4095, 0, 1993);
   datatosend.d =  map(y_servo_pos, 0, 4095, 0, 1993);


  Serial.print("X_motor: ");
  Serial.print(datatosend.a);
  Serial.print(" | Y_motor: ");
  Serial.print(datatosend.b);

  Serial.print("   X_servo: ");
  Serial.print(datatosend.c);
  Serial.print(" | Y_servo : ");
```

```
  Serial.println(datatosend.d);

   delay(10);

  esp_err_t result = esp_now_send(0, (uint8_t *) &datatosend,
sizeof(struct_message));

  if (result == ESP_OK) {
    Serial.println("Sent Successfullt");
  }
  else {
    Serial.println("Getiing Error while sending the data");
}

}
```

6.3. Programming ESP8266 (Receiver)

- Code for Rececier

```
#include <ESP8266WiFi.h>
#include <espnow.h>
#include <Servo.h>
  //  dc motor
int EN_A = 12;      //Enable pin for first motor
int IN1  = 14;       //control pin for first motor
int IN2  = 2;
int EN_B = 13;      //Enable pin for first motor
int IN3  = 15;       //control pin for first motor
int IN4  = 3;

int motor_speed1;
int motor_speed2;

 /// servo

  Servo servoA;
  Servo servoB;
int positionA = 90;
int positionB = 90;


String success;

typedef struct struct_message {
    int a;
    int b;
    int c;
    int d;
    int e;
} struct_message;

struct_message datatosend;

void OnDataRecv(uint8_t * mac, uint8_t *incomingData, uint8_t len)
{
  memcpy(&datatosend, incomingData, sizeof(datatosend));
int x_motor = datatosend.a;
int y_motor = datatosend.b;
int x_servo = datatosend.c;
int y_servo = datatosend.d;
```

```
int sw = datatosend.e;
/// dc forward and backward
if (x_motor < 700 ) {   // forward
  motor_speed1 = map(x_motor, 0, 699, 255, 0);  // Map the values
to reverse motor speed
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(EN_A, motor_speed1);
  analogWrite(EN_B, motor_speed1);
} else if (x_motor >= 700 && x_motor <= 1300) {  // Motors will
not move when the joystick is at the center
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
} else if (x_motor > 1301) {   // backward
  motor_speed1 = map(x_motor, 1301, 1993, 0, 255);  // Map the
values to forward motor speed
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  analogWrite(EN_A, motor_speed1);
  analogWrite(EN_B, motor_speed1);
}

if (y_motor < 700  ) {   // Rotating the left motor in clockwise
direction
  motor_speed2 = map(y_motor, 0, 699, 255, 0);  // Map the values
to reverse motor speed
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
   digitalWrite(IN4, HIGH);
  analogWrite(EN_A, motor_speed2);
  analogWrite(EN_B, motor_speed2);

} else if (y_motor > 1301) {   // Rotating the left motor in
anticlockwise direction
  motor_speed2 = map(y_motor, 1301, 1993, 0, 255);  // Map the
values to forward motor speed
 digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, HIGH);
```

```
  digitalWrite(IN4, LOW);
  analogWrite(EN_A, motor_speed2);
  analogWrite(EN_B, motor_speed2);
}


// servo


if (x_servo == 0 && positionA < 180) {
    positionA = positionA + 2;
    servoA.write(positionA);
    digitalWrite(26,HIGH);
    delay(3);
  }


if (x_servo == 1993 && positionA > 0) {
    positionA = positionA - 2;
    servoA.write(positionA);
    delay(3);
  }


if (y_servo == 0 && positionB < 180) {
     positionB = positionB + 2;
    servoB.write(positionB);
    delay(3);
  }


if (y_servo == 1993 && positionB > 0) {
   positionB = positionB - 2;
    servoB.write(positionB);
    delay(3);
  }

if ( sw ==0) {
  positionA = 110;
  positionB = 80;
    servoA.write(positionA);
    servoB.write(positionB);
    delay(3);
  }

 Serial.print("X_motor: ");
  Serial.print(x_motor);
```

```
  Serial.print(" | Y_motor: ");
  Serial.print(y_motor);

  Serial.print("   X_servo: ");
  Serial.print(x_servo);
  Serial.print(" | Y_servo : ");
  Serial.print(y_servo);
 Serial.print(" | sw : ");
  Serial.println(sw);

}

void setup() {
  Serial.begin(9600);
  // dc motor
  pinMode(EN_A, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(EN_B, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  //servo
  servoA.attach(16,500,2400);
  servoB.attach(4,500,2400);
  servoA.write(positionA);
  servoB.write(positionB);
  WiFi.mode(WIFI_STA);
  if (esp_now_init() != 0) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
  esp_now_set_self_role(ESP_NOW_ROLE_SLAVE);
  esp_now_register_recv_cb(OnDataRecv);
}

void loop() {

}
```

7. Operating the Spy RC Car

- In recevier side,When we connect 2 x 18650 battery ,I.e 8v . This power get supply to L298n, esp32 cam ,esp8266, 2's servos and recevier device is start to connect and operate.

- In transmitter side , we supply power through micro usb for safer side as esp32 is very sensitive.

- As soon as, both recevier and transmitter get power .Transmitter start scanning near device ,and get connected if mac address is matched.

- Now peer to peer connection is established, we are ready to operate the spy rc car.

- Ouput of ESP 32 CAM is transmitter through WIFI protocal, when it get power , esp32 cam create access point .

- To view live streaming of esp32 cam, we have to connect to that acess point , search IP adress of esp32 cam on broswer

- For this project the IP address id :- http://192.168.4.1/

8. Results and Observations

- The performance of the system is quite good, sometime we have shakie motion. But work accurately according to the command recevied.

- The range if communication is said to be upto 220 meters but when I used it ,range was arround 110 to 115 meters. But its not bad. As its totally wireless communication ,that we don’t require wifi, bluetooth , internet .

- The quality of the camera feed is good, as we have 2MP camera.

9. Conclusion

- In conclusion, the development of the Spy RC Car with the ESP32 and ESP8266 using the ESP-NOW protocol has resulted in a successful remote surveillance platform. This project demonstrated the potential of combining microcontrollers, wireless communication, and motor control to create a versatile and remotely operated vehicle with a live camera feed.

The key takeaways from this project are as follows:

1. Effective Communication: The ESP-NOW protocol proved to be a reliable and low-latency method for transmitting control commands from the ESP32 transmitter to the ESP8266 receiver. This ensured responsive control of the RC car.

2. Motor Control: The integration of the L298N motor driver allowed for precise control of the four motors on the RC car. Two servo motors were used for steering and camera orientation, providing enhanced mobility and surveillance capabilities.

3. Live Camera Feed: The ESP32-CAM module was successfully integrated to provide a live video feed from the spy RC car to the remote controller, enabling real-time monitoring and surveillance.

4. Operational Efficiency: The project allowed for seamless remote operation, with the ESP32 transmitter providing intuitive joystick-based control. Users could drive the car, adjust the camera's view, and receive live video feedback with ease.

\

10. Future Enhancements

- **Future Potential:**

While the project achieved its primary objectives, there is still room for improvement and expansion.

Future enhancements could include obstacle avoidance, autonomous navigation, increased camera capabilities.

And proper and effective design.

11. References

1. Ardunio IDE

2. Google

3. https://youtu.be/6-0dbSYxQcM?si=1YOodHG0CC2DmMm9

4. https://www.instructables.com/Servo-Controlled-by-Pushbuttons/

5. https://robojax.com/learn/arduino/?vid=robojax-servo-push-button

6. https://www.edgemicrotech.com/connecting-two-esp8266-modules-using-esp-now-a-button-and-a-led/

Component from

Robo.in

THANK YOU