}CODIGO PARA EL MOTOR Y SERVOMOTOR:

```
#include <Servo.h>
// Definición de pines para el motor DC
const int pinIN1 = 12; // Control de dirección 1
const int pinIN2 = 11; // Control de dirección 2
const int pinENA = 10; // Control PWM
// Crear un objeto Servo
Servo miServo;
// Definir el pin donde está conectado el servomotor
const int pinServo = 13;
void setup() {
// Configurar los pines del motor como salida
pinMode(pinIN1, OUTPUT);
pinMode(pinIN2, OUTPUT);
pinMode(pinENA, OUTPUT);
// Asociar el objeto Servo al pin
miServo.attach(pinServo);
}
// Funciones para controlar el servomotor
void moverIzquierda() {
miServo.write(60);
delay(1000); // Esperar 1 segundo
}
void moverCentro() {
miServo.write(90);
delay(1000); // Esperar 1 segundo
```

```
}
void moverDerecha() {
miServo.write(120);
delay(1000); // Esperar 1 segundo
}
// Funciones para controlar el motor DC
void retroceder() {
digitalWrite(pinIN1, HIGH);
digitalWrite(pinIN2, LOW);
analogWrite(pinENA, 255); // Velocidad máxima
delay(2000); // Avanzar por 2 segundos
detenerMotor(); // Detener el motor
}
void avanzar() {
digitalWrite(pinIN1, LOW);
digitalWrite(pinIN2, HIGH);
analogWrite(pinENA, 255); // Velocidad máxima
delay(2000); // Retroceder por 2 segundos
detenerMotor(); // Detener el motor
}
void detenerMotor() {
analogWrite(pinENA, 0); // Detener el motor
}
void loop() {
// Control del motor DC
avanzar();
```

```
delay(1000); // Esperar 1 segundo
retroceder();
delay(1000); // Esperar 1 segundo
// Control del servomotor
moverIzquierda();
moverCentro();
moverDerecha();
}
```

CÓDIGO DE UN SOLO SENSOR:

```
#include <NewPing.h>
// Definición de pines para el sensor ultrasónico HC-SR04
#define TRIGGER_PIN 1
#define ECHO_PIN 2
#define MAX_DISTANCE 200 // Distancia máxima en centímetros para medir (en este
caso, 200 cm o 2 metros)
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // Crea un objeto
NewPing
void setup() {
 Serial.begin(9600); // Inicializa la comunicación serial
}
void loop() {
 delay(1000); // Espera medio segundo entre cada medición
 // Realiza una medición de distancia
 unsigned int distancia = sonar.ping_cm();
 // Muestra la distancia medida en el monitor serial
 Serial.print("Distancia: ");
```

```
 if (distancia == 0) {
 Serial.println("Fuera de rango");
 } else {
 Serial.print(distancia);
 Serial.println(" cm");
 }
}

// Definir pines para el sensor ultrasónico
#define TRIG_PIN 1
#define ECHO_PIN 2
// Definir pines para el puente H
#define ENA 3
#define IN1 4
#define IN2 5
// Definir distancia para detener el motor
#define DISTANCIA_DETECCION 13
#define DISTANCIA_FRENADO 12
// Definir pines para el sensor de color TCS3200
#define S0 6
#define S1 7
#define S2 8
#define S3 9
#define OUT 10
void setup() {
pinMode(ENA, OUTPUT);
```

```
pinMode(IN1, OUTPUT);
pinMode(IN2, OUTPUT);
pinMode(TRIG_PIN, OUTPUT);
pinMode(ECHO_PIN, INPUT);
pinMode(S0, OUTPUT);
pinMode(S1, OUTPUT);
pinMode(S2, OUTPUT);
pinMode(S3, OUTPUT);
pinMode(OUT, INPUT);
Serial.begin(9600);
// Configurar el sensor TCS3200 para detectar colores
digitalWrite(S0, HIGH);
digitalWrite(S1, LOW);
}
void loop() {
unsigned int distancia = obtenerDistancia();
Serial.print("Distancia: ");
Serial.println(distancia);
if (distancia <= DISTANCIA_DETECCION) {
detenerMotor();
String color = detectarColor();
Serial.print("Color detectado: ");
Serial.println(color);
if (color == "Verde") {
retrocederMotor(1500); // Retrocede 1.5 segundos si el objeto es verde
} else if (color == "Rojo") {
```

```
retrocederMotor(5000); // Retrocede 3 segundos si el objeto es rojo
delay(3000); // Pausa antes de avanzar de nuevo
avanzarMotor(); // Avanza de nuevo
}
} else if (distancia <= DISTANCIA_FRENADO) {
frenarMotor();
} else {
avanzarMotor();
}
delay(100); // Pequeña pausa para no saturar el sensor
}
// Función para avanzar el motor
void avanzarMotor() {
digitalWrite(IN1, HIGH);
digitalWrite(IN2, LOW);
analogWrite(ENA, 170);
}
// Función para retroceder el motor
void retrocederMotor(int duracion) {
digitalWrite(IN1, LOW);
digitalWrite(IN2, HIGH);
analogWrite(ENA, 130);
delay(duracion);
detenerMotor();
}
// Función para frenar el motor
```

```
void frenarMotor() {
digitalWrite(IN1, HIGH);
digitalWrite(IN2, HIGH);
analogWrite(ENA, 200);
}
// Función para detener el motor
void detenerMotor() {
digitalWrite(IN1, LOW);
digitalWrite(IN2, LOW);
analogWrite(ENA, 0);
}
// Función para obtener la distancia usando el sensor ultrasónico
unsigned int obtenerDistancia() {
digitalWrite(TRIG_PIN, LOW);
delayMicroseconds(10);
digitalWrite(TRIG_PIN, HIGH);
delayMicroseconds(10);
digitalWrite(TRIG_PIN, LOW);
unsigned long duracion = pulseIn(ECHO_PIN, HIGH);
unsigned int distancia = duracion / 58;
return distancia;
}
// Función para detectar el color usando el sensor TCS3200
String detectarColor() {
int frecuenciaRojo = leerColor(LOW, LOW);
int frecuenciaVerde = leerColor(HIGH, HIGH);
```

```
if (frecuenciaVerde < frecuenciaRojo) {
return "Verde";
} else if (frecuenciaRojo < frecuenciaVerde) {
return "Rojo";
} else {
return "Desconocido";
}
}
// Función para leer la frecuencia de color del sensor TCS3200
int leerColor(bool estadoS2, bool estadoS3) {
digitalWrite(S2, estadoS2);
digitalWrite(S3, estadoS3);
delay(100);
int frecuencia = pulseIn(OUT, LOW);
return frecuencia;
}
```