Gestión de obstáculos

Lógica de programación

Nuestro robot autónomo está programado para para evitar chocar con obstáculos, nuestra lógica de programación es que en un principio el robot permanezca apagado hasta la activación mediante un push button, seguido a eso el carro empieza a avanzar y el servomotor se posiciona en un ángulo de 90° hasta que el sensor frontal detecte algún obstáculo a una cierta distancia, una vez el sensor haya detectado un obstáculo el motor se detiene y retrocede durante 0.75 segundos y seguido a eso se vuelve a parar, cuando el motor se para los sensores colocados al costado del carro empiezan a censar y el que detecte un obstáculo a menor distancia mandará una señal al servomotor para que gire en dirección contraria y así evitar chocar.

Pseudocódigo:

1.-Inicio

2.-Condicion: ¿Botón presionado? /¿Hay lectura "LOW" del pushbutton asociado al pin 3?
--- No: Esperar hasta que la condición se cumpla
--- Si: Avanzar con el programa

3.-Enviar señal PWM al pin L del puente H y hacer girar el motor en sentido horario para avanzar

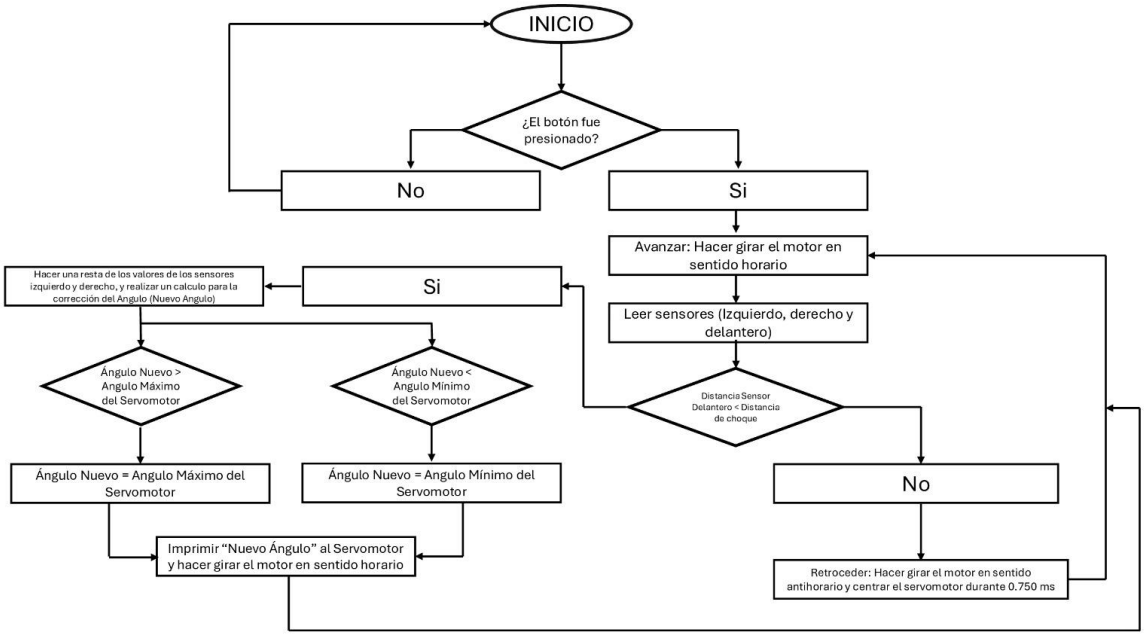
4.-Leer valores de los sensores laterales y frontal

5.-Condicion: ¿La distancia (El valor) del sensor delantero es MENOR que distancia de choque?
--- Si: Imprimir un valor giro de 80 grados al servomotor y enviar señal PWM al pin R del puente H y hacer girar el motor en sentido antihorario para retroceder, luego esperar 0.75 segundos y regresar al paso 3
--- No: Comparar distancia de los 2 sensores laterales y aplicar una resta para saber de qué lado hay un objeto más cerca, para de esta manera hacer una corrección al multiplicar el resultado de la resta por la constante "K" (Cuyo valor es 1.2) para hacer una corrección del ángulo del servomotor, luego se vuelve a comparar:
------- Si el valor del nuevo ángulo es mayor que el valor máximo del servo (Que vale 115 grados), entonces el valor del ángulo corregido pasa a ser el valor máximo del servo
------- Si el valor del nuevo ángulo es menor que el valor mínimo del servo (Que vale 55 grados), entonces el valor del ángulo corregido pasa a ser el valor mínimo del servo

--- Luego se imprime el valor del nuevo ángulo para el servomotor y se siguen enviando señales PWM al pin L al puente H para que haga girar el motor en sentido horario y avanzar

6.-Se retorna a la condición 4

Diagrama de flujo:



Implementación del pseudocódigo (Algoritmo) en código:

```
#include <Servo.h>

// --- 1. DEFINICIÓN DE PINES ---

// Servomotor MG996R (Dirección)
Servo direccionServo;
const int SERVO_PIN = 3;

// Driver BTS7960 (Propulsión)
const int MOTOR_PWM_L = 6;
const int MOTOR_PWM_R = 5;

// Sensores Ultrasónicos HC-SR04
const int TRIG_L = 9;
const int ECHO_L = 10;

const int TRIG_R = 11;
const int ECHO_R = 12;
```

```
const int TRIG_D = 7;
const int ECHO_D = 8;

// Botón con pull-down en el pin 2
const int BUTTON_PIN = 2;

// --- VARIABLE PARA ACTIVAR EL ROBOT ---
bool robotActivo = false;

// --- 2. CONSTANTES DE CALIBRACIÓN ---

const float Kp = 1.2;

const int SERVO_CENTRO = 80;
const int SERVO_LIMITE_MAX = 115;
const int SERVO_LIMITE_MIN = 55;

const int velocidad = 30;
const int retroceso = 60;

const float DISTANCIA_CHOQUE = 25.0;
const int TIEMPO_REVERSA = 250;

// --- 3. FUNCIONES ---

float leerDistancia(int trigPin, int echoPin) {
   digitalWrite(trigPin, LOW);
   delayMicroseconds(2);
   digitalWrite(trigPin, HIGH);
   delayMicroseconds(10);
   digitalWrite(trigPin, LOW);

   long duracion = pulseIn(echoPin, HIGH);
   float distancia = duracion * 0.034 / 2;

   if (distancia > 200 || distancia < 0) return 200;
   return distancia;
}

void avanzar(int velocidad) {
   analogWrite(MOTOR_PWM_L, velocidad);
   analogWrite(MOTOR_PWM_R, 0);
}
```

```
void detener() {
  analogWrite(MOTOR_PWM_L, 0);
  analogWrite(MOTOR_PWM_R, 0);
}

void retrocederMotor(int velocidad) {
  analogWrite(MOTOR_PWM_L, 0);
  analogWrite(MOTOR_PWM_R, retroceso);
}

// --- 4. SETUP ---

void setup() {
  delay(2000);

  pinMode(TRIG_L, OUTPUT); pinMode(ECHO_L, INPUT);
  pinMode(TRIG_R, OUTPUT); pinMode(ECHO_R, INPUT);
  pinMode(TRIG_D, OUTPUT); pinMode(ECHO_D, INPUT);

  pinMode(MOTOR_PWM_L, OUTPUT);
  pinMode(MOTOR_PWM_R, OUTPUT);

  direccionServo.attach(SERVO_PIN);
  direccionServo.write(SERVO_CENTRO);
  detener();

  Serial.begin(9600);

  // --- CONFIGURACIÓN DEL BOTÓN (PULL-DOWN REAL) ---
  pinMode(BUTTON_PIN, INPUT);

  Serial.println("Robot Listo. Presiona el botón para iniciar...");

  // Espera hasta que el botón sea presionado
  while (digitalRead(BUTTON_PIN) == LOW) {
    delay(20);
  }

  Serial.println("Boton presionado. Robot ACTIVADO.");
  robotActivo = true;
}

// --- 5. LOOP PRINCIPAL ---
```

```
void loop() {

  // Si aún no se ha activado el robot, no hace nada
  if (!robotActivo) {
    return;
  }

  // ---- LECTURA DE SENSORES ----
  float dist_delantera = leerDistancia(TRIG_D, ECHO_D);
  float dist_izquierda = leerDistancia(TRIG_L, ECHO_L);
  float dist_derecha = leerDistancia(TRIG_R, ECHO_R);

  // --- EVITACIÓN FRONTAL ---
  if (dist_delantera < DISTANCIA_CHOQUE) {

    detener();
    delay(TIEMPO_REVERSA);

    direccionServo.write(SERVO_CENTRO);
    retrocederMotor(velocidad);
    delay(750);

    detener();
  }

  // --- CONTROL PROPORCIONAL LATERAL ---
  else {

    float error = dist_derecha - dist_izquierda;
    int correccion = (int)(error * Kp);

    int nuevo_angulo = SERVO_CENTRO + correccion;

    if (nuevo_angulo > SERVO_LIMITE_MAX) nuevo_angulo =
SERVO_LIMITE_MAX;
    if (nuevo_angulo < SERVO_LIMITE_MIN) nuevo_angulo =
SERVO_LIMITE_MIN;

    direccionServo.write(nuevo_angulo);
    avanzar(velocidad);
  }

  // ---- DEBUG SERIAL (Opcional) ----
  Serial.print("DI: "); Serial.print(dist_izquierda);
```

```
    Serial.print(" | DD: "); Serial.print(dist_derecha);
    Serial.print(" | DF: "); Serial.print(dist_delantera);
    Serial.print(" | Angulo: "); Serial.println(direccionServo.read());

    delay(50);
}
```