DEMO SEGUIDOR SOLAR

```
#include <ESP32Servo.h>

// ---------- Definición de Pines y Parámetros ----------
#define TRIG_PIN 23  // Pin de disparo TRIG del sensor (AZUL)
#define ECHO_PIN 22  // Pin de ECHO del sensor (NARANJA)
#define LED_PIN 2    // LED para indicar detección de movimiento
#define BOTON 12     // Pin del botón para iniciar

// Pines de los servos
const int SERVO_PIN_HORIZONTAL = 18;
const int SERVO_PIN_VERTICAL   = 19;

// Valores iniciales para los servos
const int SERVO_INICIAL_H = 90;
const int SERVO_INICIAL_V = 120;

// Límites de movimiento para los servos
const int SERVOH_LIMIT_HIGH = 175;
const int SERVOH_LIMIT_LOW  = 5;
const int SERVOV_LIMIT_HIGH = 170;
const int SERVOV_LIMIT_LOW  = 70;

// Pines de los LDR en configuración de cruz
const int LDR_TOP_PIN    = 32; // Sensor superior
const int LDR_BOTTOM_PIN = 33; // Sensor inferior
const int LDR_LEFT_PIN   = 34; // Sensor izquierdo
const int LDR_RIGHT_PIN  = 35; // Sensor derecho

// Parámetros de control
const int LIGHT_THRESHOLD = 200; // Umbral global de luz para retornar a
posición inicial
const int TOLERANCIA      = 90;   // Tolerancia para el ajuste
const int DTIME           = 10;   // Retardo entre lecturas (ms)
const int obstacleDistance = 15;  // Distancia mínima para detección de
obstáculos (cm)

// ---------- Variables Globales ----------
Servo servoHorizontal; // Servo para el movimiento horizontal
Servo servoVertical;   // Servo para el movimiento vertical

int anguloHorizontal = SERVO_INICIAL_H;
int anguloVertical   = SERVO_INICIAL_V;

// ---------- Función para Esperar el Botón ----------
void waitForButtonPress() {
  Serial.println("Presiona el botón para iniciar el seguimiento solar");
  while (digitalRead(BOTON) == HIGH) {
```

```
    delay(50);
  }
  delay(200);
  while (digitalRead(BOTON) == LOW) {
    delay(50);
  }
  delay(200);
  Serial.println("¡Botón pulsado!");
}

// -------------------- Función para inicializar los servomotores -----------
---------
void initializeServos() {
  waitForButtonPress();  // Nos aseguramos que se ha pulsado el boton de ON

  Serial.println("Posicionando servos en posiciones iniciales...");
  servoHorizontal.write(SERVO_INICIAL_H);
  servoVertical.write(SERVO_INICIAL_V);
  anguloHorizontal = SERVO_INICIAL_H;
  anguloVertical = SERVO_INICIAL_V;
}

void setup() {
  Serial.begin(115200);
  Serial.println("Iniciando sistema...");

  servoHorizontal.attach(SERVO_PIN_HORIZONTAL);
  servoVertical.attach(SERVO_PIN_VERTICAL);
  servoHorizontal.write(anguloHorizontal);
  servoVertical.write(anguloVertical);

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(LED_PIN, OUTPUT);
  pinMode(BOTON, INPUT_PULLUP);

  waitForButtonPress();
  // Posiciona los servos en las posiciones iniciales
  initializeServos();

  Serial.println("Sistema listo. Comenzando bucle principal...");
  delay(2500);
}

void loop() {
  // Lectura de LDRs
  int ldrTop    = analogRead(LDR_TOP_PIN);
  int ldrBottom = analogRead(LDR_BOTTOM_PIN);
  int ldrLeft   = analogRead(LDR_LEFT_PIN);
```

```
  int ldrRight  = analogRead(LDR_RIGHT_PIN);

  // Imprime lecturas de LDRs
  Serial.print("LDR Superior: ");
  Serial.print(ldrTop);
  Serial.print("\tLDR Inferior: ");
  Serial.print(ldrBottom);
  Serial.print("\tLDR Izquierdo: ");
  Serial.print(ldrLeft);
  Serial.print("\tLDR Derecho: ");
  Serial.println(ldrRight);

  // Verifica obstáculo
  if (isObstacleDetected()) {
    Serial.println("¡Obstáculo detectado! Deteniendo movimiento...");
    digitalWrite(LED_PIN, HIGH);
    returnToInitialPositions();
    delay(500);
    return;
  } else {
    digitalWrite(LED_PIN, LOW);
  }

  // Calcular nivel global de luz
  int avgLight = (ldrTop + ldrBottom + ldrLeft + ldrRight) / 4;
  Serial.print("Promedio LDR: ");
  Serial.println(avgLight);

  if (avgLight <= LIGHT_THRESHOLD) {
    returnToInitialPositions();
    delay(500);
    return;
  }

  // Actualiza movimientos de servos
  updateVerticalServo(ldrTop, ldrBottom);
  updateHorizontalServo(ldrLeft, ldrRight);

  delay(DTIME);
}

// ---------- Funciones para Actualizar los Servos ----------

void updateVerticalServo(int ldrTop, int ldrBottom) {
  if (abs(ldrTop - ldrBottom) > TOLERANCIA) {
    if (ldrTop > ldrBottom) {
      anguloVertical = min(anguloVertical + 1, SERVOV_LIMIT_HIGH);
      Serial.print("Aumenta ángulo vertical: ");
      Serial.println(anguloVertical);
```

```
    } else {
      anguloVertical = max(anguloVertical - 1, SERVOV_LIMIT_LOW);
      Serial.print("Disminuye ángulo vertical: ");
      Serial.println(anguloVertical);
    }
    servoVertical.write(anguloVertical);
  }
}

void updateHorizontalServo(int ldrLeft, int ldrRight) {
  if (abs(ldrLeft - ldrRight) > TOLERANCIA) {
    if (ldrLeft > ldrRight) {
      anguloHorizontal = max(anguloHorizontal - 1, SERVOH_LIMIT_LOW);
      Serial.print("Disminuye ángulo horizontal: ");
      Serial.println(anguloHorizontal);
    } else {
      anguloHorizontal = min(anguloHorizontal + 1, SERVOH_LIMIT_HIGH);
      Serial.print("Aumenta ángulo horizontal: ");
      Serial.println(anguloHorizontal);
    }
    servoHorizontal.write(anguloHorizontal);
  }
}

// -------------------- Función para retornar a posiciones iniciales --------
------------
void returnToInitialPositions() {
  Serial.println("Nivel de luz bajo, regresando suavemente a la posición
inicial...");

  while (anguloHorizontal != SERVO_INICIAL_H || anguloVertical !=
SERVO_INICIAL_V) {
    if (anguloHorizontal > SERVO_INICIAL_H) anguloHorizontal--;
    else if (anguloHorizontal < SERVO_INICIAL_H) anguloHorizontal++;

    if (anguloVertical > SERVO_INICIAL_V) anguloVertical--;
    else if (anguloVertical < SERVO_INICIAL_V) anguloVertical++;

    servoHorizontal.write(anguloHorizontal);
    servoVertical.write(anguloVertical);
    delay(10);  // Suaviza el retorno
  }

  Serial.println("Servos en posición inicial.");
}


// -------------------- Funciones sensor ultrasónico --------------------
```

```
// Función para medir la distancia con el sensor HC-SR04
long measureDistance() {
  long duration, distance;

  // Genera el pulso de disparo
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  // Lee el tiempo del pulso en el pin Echo
  duration = pulseIn(ECHO_PIN, HIGH);

  // Calcula la distancia (en cm)
  distance = (duration / 2) / 29.1;  // Conversión de tiempo a distancia

  return distance;  // Retorna el valor de la distancia medida
}

// Función para evaluar la distancia y verificar si hay un obstáculo
bool isObstacleDetected() {
  long distance = measureDistance();
  Serial.print("Distancia: ");
  Serial.println(distance);

  // Si la distancia es menor al umbral, se considera que hay un obstáculo
  if (distance < obstacleDistance) {
    return true;
  }
  return false;
}
```