Universidad de Guadalajara

Piensa y trabaja

REPORTE DE PRÁCTICA

Práctica 7: Torreta seguidora de luz con ESP32

Participantes:

Lorena Villalobos Carrillo

Ximena Lucía Rodríguez Oliva

Brayam De Jesus De La Cerda Valdivia

Seminario de problemas de programación embebidos

(D01 –I9893)

Universidad de Guadalajara

Piensa y trabaja

1. Introducción

La robótica autónoma busca imitar ciertos comportamientos de organismos vivos como la percepción, la adaptación al entorno y la toma de decisiones en tiempo real. En este contexto, una de las aplicaciones más comunes y formativas es la creación de mecanismos que responden a estímulos externos como la luz.

En esta práctica se desarrolló una torreta seguidora de luz utilizando el microcontrolador ESP32, dos servomotores y cuatro fotorresistencias (LDRs). El sistema fue diseñado para detectar la dirección de la fuente luminosa más intensa en un entorno y orientar la torreta hacia ella. Como fuente luminosa se utilizó, por ejemplo, la linterna de un celular.

Este tipo de mecanismo tiene aplicaciones prácticas en sistemas solares (seguimiento solar), vigilancia automática, robótica orientada al entorno, y proyectos de mecatrónica educativa.

2. Desarrollo

2.1 Materiales y herramientas

- Microcontrolador ESP32
- 2 servomotores SG90 o similares
- 4 fotorresistencias (LDRs)
- 4 resistencias de 10kΩ
- Fuente de alimentación de 5V para los servos (opcional)
- Protoboard
- Cables Dupont
- Estructura de soporte para la torreta (impresa en 3D o de cartón/plástico)
- Computadora con Arduino IDE y librería ESP32Servo instalada

2.2 Configuración del sistema

• Sensores de luz (LDRs): Se colocaron en un arreglo en forma de cruz (arriba a la derecha, arriba a la izquierda, abajo a la derecha, abajo a la izquierda) conectados a los pines analógicos 35, 32, 34 y 33 del ESP32.

• Servomotores:

1. ServoY (eje vertical): conectado al pin digital 13.
2. ServoZ (eje horizontal): conectado al pin digital 14.

• Microcontrolador ESP32: configurado para controlar los servos y leer los sensores analógicos.

• Librería utilizada: ESP32Servo.h, necesaria para controlar servomotores desde el ESP32 debido a diferencias de temporización respecto a Arduino Uno.

2.3 Lógica de funcionamiento

El sistema se basa en una lectura simultánea de las cuatro fotorresistencias dispuestas en cuadrante. Se calculan dos diferencias principales:

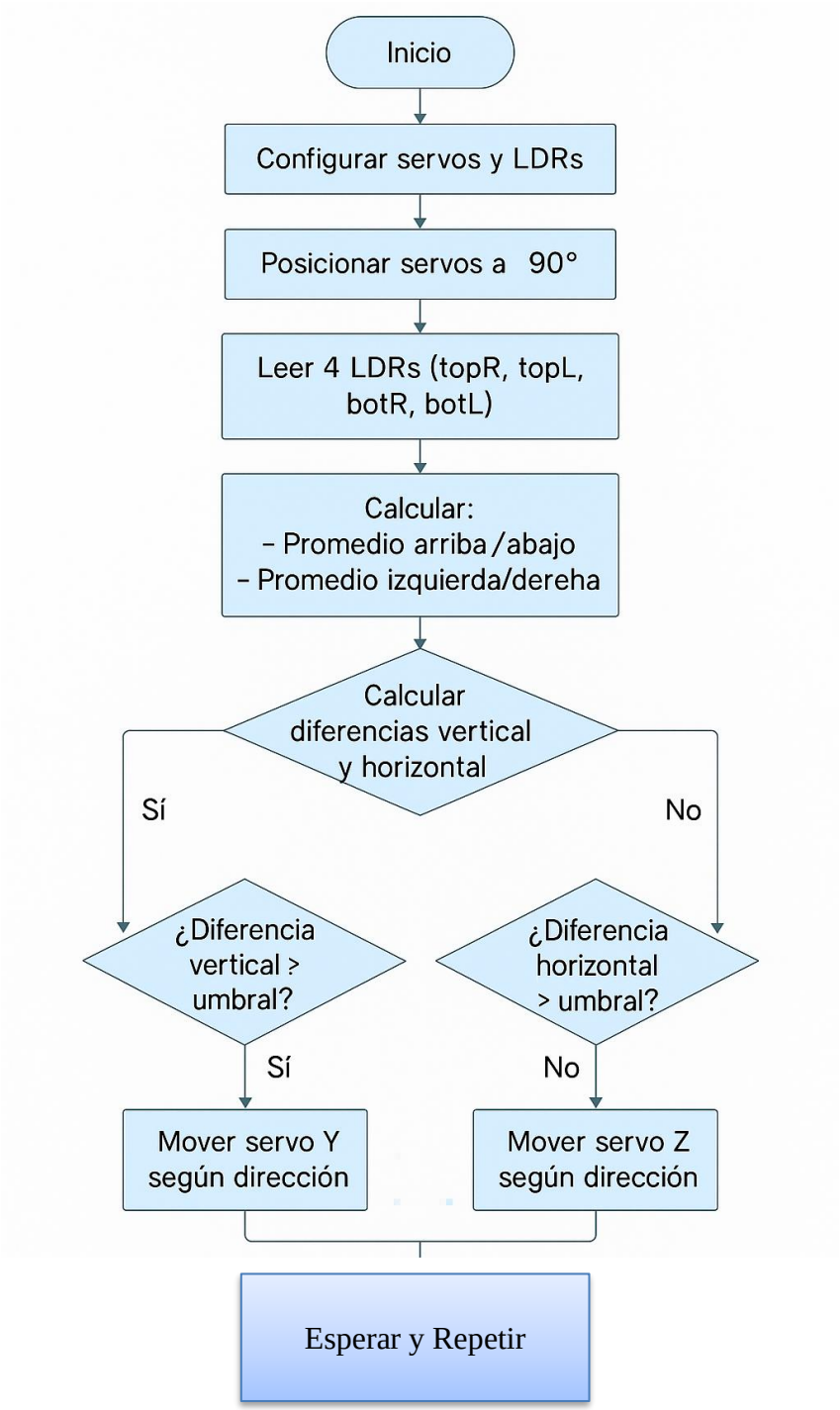
- Vertical: diferencia entre el promedio de la luz en la parte inferior y la parte superior. Un valor positivo indica más luz abajo (la torreta debe mirar hacia abajo), y viceversa.
- Horizontal: diferencia entre la izquierda y la derecha, para saber si la luz está desplazada lateralmente.

Ambas diferencias se comparan contra un umbral de sensibilidad (lightThreshold) para evitar movimientos por pequeñas fluctuaciones.

Cada vez que se detecta una diferencia significativa, se ajusta la posición del servo correspondiente en pasos de 1 grado (stepSize). Esto permite un seguimiento suave y progresivo.

El sistema es capaz de girar de 0° a 180° tanto en eje vertical como horizontal, y al inicio se posiciona en 90°.

2.4 Diagrama de flujo



3. Resultados y observaciones

El prototipo fue capaz de seguir eficazmente una fuente de luz como la linterna de un celular, realizando movimientos en ambas direcciones hasta alinear los sensores hacia el punto de mayor iluminación.

Logros principales:

- Los servomotores responden de forma fluida a los cambios de luz.
- El sistema no presenta oscilaciones cuando se estabiliza.
- La lógica de cuadrantes permite detectar la dirección general sin necesidad de sensores complejos.
- El código es modular y permite ajustes simples como sensibilidad o velocidad.

Observaciones:

- En ambientes muy iluminados, la diferencia de luz entre sensores disminuye, dificultando el seguimiento preciso.
- Si el sensor LDR no está adecuadamente sombreado entre sí, la luz difusa puede causar lecturas similares.
- Puede haber ligeras variaciones entre sensores, lo cual podría compensarse con calibración individual.

4. Conclusión

Esta práctica integró conceptos de percepción y acción en un sistema embebido. Se logró implementar una torreta autónoma que responde en tiempo real a cambios de iluminación utilizando sensores simples y actuadores mecánicos.

Se comprendió el funcionamiento de las fotorresistencias como entradas analógicas, la lectura simultánea de múltiples canales ADC, y el control por PWM de servomotores mediante el ESP32. Además, se reforzaron conceptos como el uso de librerías externas, estructuración del código, y diseño orientado a la eficiencia.

Este sistema constituye una base sólida para desarrollar mecanismos autónomos más complejos como paneles solares móviles, robots exploradores o dispositivos de orientación inteligente.

5. Código

```
#include <ESP32Servo.h>

// Pines de los servos
const int servoY_pin = 13;
const int servoZ_pin = 14;

// Pines de fotorresistencias
const int LDR_TopRight = 35;
const int LDR_TopLeft = 32;
const int LDR_BotRight = 34;
const int LDR_BotLeft = 33;

// Objetos Servo
Servo servoY;
Servo servoZ;

int posY = 90;  // Vertical
int posZ = 90;  // Horizontal

const int minAngle = 0;
```

```
const int maxAngle = 180;
const int stepSize = 1;
const int lightThreshold = 50; // Sensibilidad al cambio

void setup() {
  Serial.begin(115200);

  servoY.attach(servoY_pin, 500, 2400);
  servoZ.attach(servoZ_pin, 500, 2400);

  servoY.write(posY);
  servoZ.write(posZ);
}

void loop() {
  int topR = analogRead(LDR_TopRight);
  int topL = analogRead(LDR_TopLeft);
  int botR = analogRead(LDR_BotRight);
  int botL = analogRead(LDR_BotLeft);

  int topAvg = (topR + topL) / 2;
```

```
int botAvg = (botR + botL) / 2;
int leftAvg = (topL + botL) / 2;
int rightAvg = (topR + botR) / 2;

int verticalDiff = botAvg - topAvg;
int horizontalDiff = rightAvg - leftAvg;

if (abs(verticalDiff) > lightThreshold) {
  if (verticalDiff > 0 && posY < maxAngle) posY += stepSize;
  else if (verticalDiff < 0 && posY > minAngle) posY -= stepSize;
  servoY.write(posY);
}

if (abs(horizontalDiff) > lightThreshold) {
  if (horizontalDiff > 0 && posZ < maxAngle) posZ += stepSize;
  else if (horizontalDiff < 0 && posZ > minAngle) posZ -= stepSize;
  servoZ.write(posZ);
}

Serial.print("Y: "); Serial.print(posY);
Serial.print(" Z: "); Serial.print(posZ);
```

```
  Serial.print(" | Top: "); Serial.print(topAvg);
  Serial.print(" Bot: "); Serial.print(botAvg);
  Serial.print(" | Left: "); Serial.print(leftAvg);
  Serial.print(" Right: "); Serial.println(rightAvg);

  delay(100);
}
```