Universidad de Guadalajara

Piensa y trabaja

REPORTE DE PRÁCTICA

Proyecto Final: Carrito seguidor 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

El presente proyecto final consiste en el desarrollo de un vehículo autónomo seguidor de luz, cuyo propósito es moverse en la dirección en la que haya mayor intensidad lumínica. Se trata de una aplicación de sistemas embebidos, sensores analógicos y control de motores, basada en el microcontrolador ESP32.

El vehículo está equipado con seis sensores de luz (fotoresistencias o LDRs) ubicados estratégicamente: dos al frente, dos atrás y uno en cada lateral. Estos sensores permiten determinar en qué dirección hay más luz, y con base en esa información, el ESP32 decide cómo mover el carrito mediante el control de cuatro motores DC (uno por rueda).

Este tipo de sistemas tiene aplicaciones prácticas en robots exploradores, paneles solares móviles, juguetes inteligentes o prototipos educativos. Durante su desarrollo se aplicaron conocimientos de electrónica digital, programación en C++, lógica condicional y sistemas de control.

2. Desarrollo

2.1 Materiales y herramientas

- 1 microcontrolador ESP32
- 4 motores DC (una por cada rueda)
- 6 fotoresistencias (LDRs)
- Resistencias de 10kΩ para divisor de voltaje
- Puente H o transistores para control de motores (L298N o similar)
- Batería o fuente de alimentación externa para los motores
- Protoboard y cables de conexión
- Chasis para carrito (impreso o armado manualmente)
- Computadora con Arduino IDE configurado para ESP32

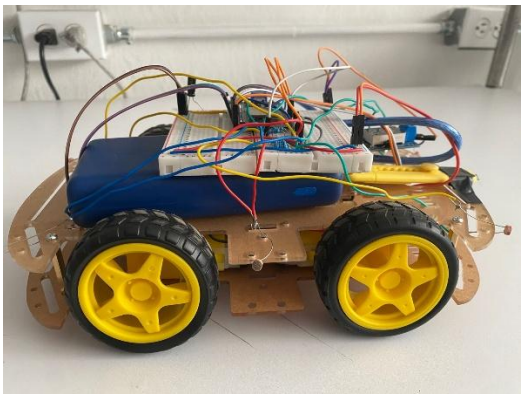
2.2 Configuración del sistema

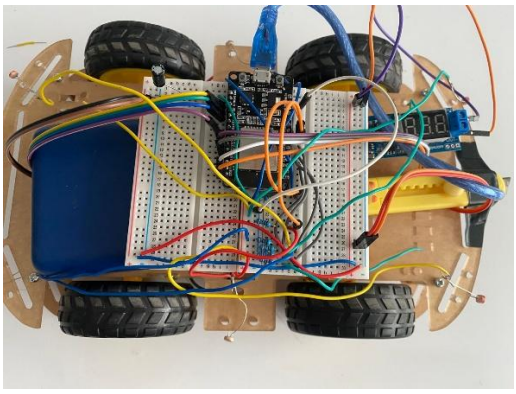
- Motores: cada una de las cuatro ruedas tiene un motor controlado por dos pines digitales, para dirección adelante o detenerse.

• Sensores de luz: se colocaron en las siguientes posiciones:

- LDR_FD / LDR_FI: Frente derecha / izquierda
- LDR_AD / LDR_AI: Atrás derecha / izquierda
- LDR_LD / LDR_LI: Lateral derecha / izquierda

• Umbrales: se establecieron manualmente valores de referencia para cada sensor, determinados mediante prueba y error en condiciones reales.





2.3 Lógica de funcionamiento

La lógica de este proyecto se basa en comparar los valores de los seis sensores para determinar cuál lado del carrito recibe más luz. El programa ejecuta los siguientes pasos en cada ciclo:

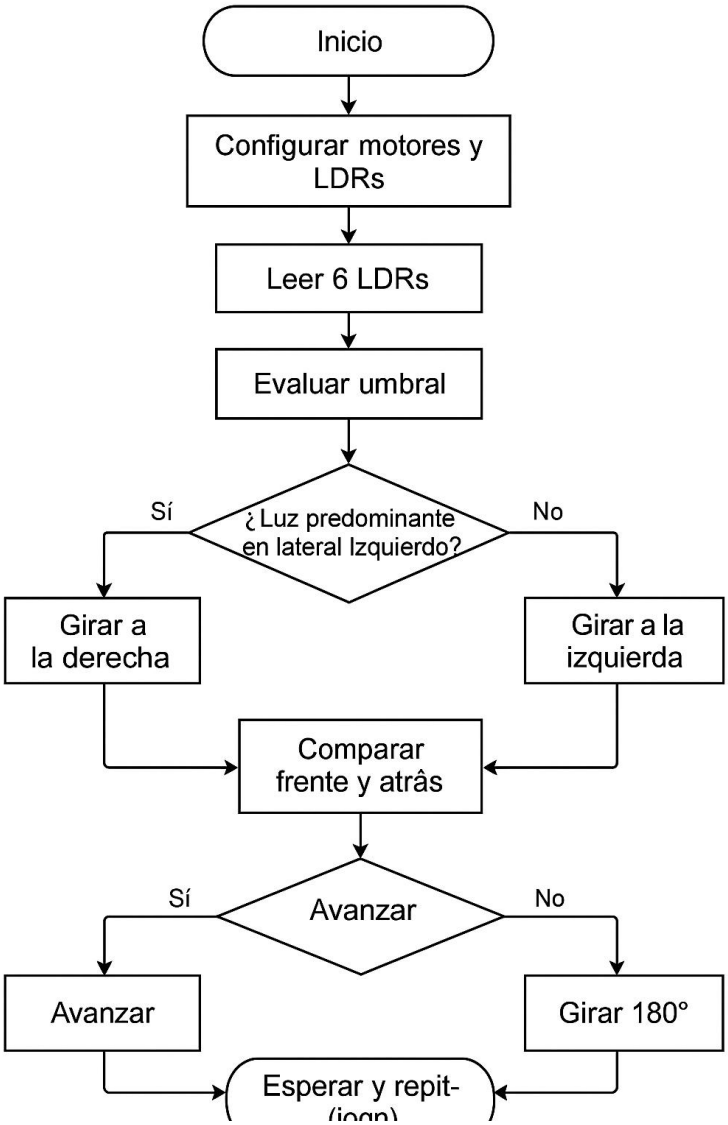
1. **Lectura de sensores:** Se obtienen los valores analógicos de cada LDR.
2. **Evaluación de umbral:** Si ningún sensor supera su umbral, el carrito se detiene.

3. **Comparación lateral:** Si el sensor derecho lateral detecta más luz, el carrito gira hacia la derecha. Si el izquierdo detecta más, gira a la izquierda.
4. **Comparación frontal y trasera:** Si no hay luz predominante en los lados, se comparan los valores promedio del frente y atrás.

    - Si el frente tiene más luz, el carrito avanza.
    - Si la luz es mayor atrás, el carrito gira 180° para reorientarse.

El programa repite este ciclo constantemente, permitiendo una adaptación dinámica a las condiciones del entorno.

2.4 Diagrama De Flujo



3. Resultados y observaciones

Durante las pruebas, se verificó que el comportamiento del carrito respondía adecuadamente a la luz dirigida desde un celular u otra fuente. Se lograron los siguientes resultados:

Logros:

- Movimiento autónomo hacia fuentes de luz reales.
- Giros suaves y consistentes basados en sensores laterales.
- Detección eficiente de condiciones sin luz (modo de espera/detención).
- Control total sobre cada una de las ruedas gracias al mapeo individual.

Observaciones:

- Los umbrales de los sensores deben ajustarse manualmente dependiendo del entorno.
- La luz ambiental puede afectar el comportamiento si los sensores no están protegidos.
- Se sugiere aplicar una calibración inicial o valores promedio dinámicos.

4. Conclusión

Este proyecto final permitió integrar múltiples habilidades fundamentales de la ingeniería en sistemas embebidos: lectura de sensores, control de actuadores, lógica condicional, procesamiento de datos y programación modular.

El **carrito seguidor de luz** constituye una base ideal para seguir explorando proyectos de robótica móvil, seguimiento de estímulos y control autónomo. La experiencia también sirvió para comprender mejor los retos reales de calibración, sensibilidad de sensores y estructura mecánica de robots.

5. Código

```
// Pines de motores (4 ruedas)
#define IN1 15  // Trasero Izquierdo
#define IN2 2
#define IN3 4   // Trasero Derecho
#define IN4 5
#define IN5 23  // Delantero Izquierdo
#define IN6 22
#define IN7 21  // Delantero Derecho
#define IN8 19

// Pines de sensores de luz (LDRs)
const int LDR_FD = 34; // Frente Derecha
const int LDR_FI = 35; // Frente Izquierda
const int LDR_AD = 32; // Atrás Derecha
const int LDR_AI = 33; // Atrás Izquierda
const int LDR_LD = 25; // Lateral Derecha
const int LDR_LI = 26; // Lateral Izquierda

// Umbrales individuales (ajusta manualmente según pruebas)
int umbral_FD = 3300;
int umbral_FI = 3300;
int umbral_AD = 3300;
int umbral_AI = 3300;
int umbral_LD = 2900;
int umbral_LI = 2900;

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

  // Configurar pines de motores como salida
  int motorPins[] = {IN1, IN2, IN3, IN4, IN5, IN6, IN7, IN8};
  for (int i = 0; i < 8; i++) {
    pinMode(motorPins[i], OUTPUT);
  }

  // Configurar pines LDR como entrada
  int ldrPins[] = {LDR_FD, LDR_FI, LDR_AD, LDR_AI, LDR_LD, LDR_LI};
  for (int i = 0; i < 6; i++) {
    pinMode(ldrPins[i], INPUT);
  }

  Serial.println("Iniciado: Modo prueba LDR con 4 ruedas.");
}

void loop() {
  // Leer valores de sensores
  int luzFD = analogRead(LDR_FD);
  int luzFI = analogRead(LDR_FI);
```

```
  int luzAD = analogRead(LDR_AD);
  int luzAI = analogRead(LDR_AI);
  int luzLD = analogRead(LDR_LD);
  int luzLI = analogRead(LDR_LI);

  // Mostrar valores por el monitor serial
  Serial.print("FD: "); Serial.print(luzFD);
  Serial.print(" | FI: "); Serial.print(luzFI);
  Serial.print(" | AD: "); Serial.print(luzAD);
  Serial.print(" | AI: "); Serial.print(luzAI);
  Serial.print(" | LD: "); Serial.print(luzLD);
  Serial.print(" | LI: "); Serial.println(luzLI);

  // Verificar luz suficiente
  bool luzOK = (
    luzFD > umbral_FD ||
    luzFI > umbral_FI ||
    luzAD > umbral_AD ||
    luzAI > umbral_AI ||
    luzLD > umbral_LD ||
    luzLI > umbral_LI
  );

  if (!luzOK) {
    detenerTodo();
    Serial.println("Poca luz. Detenido.");
    delay(500);
    return;
  }

  // Lógica mejorada
  if (luzLD > luzLI && luzLD > 2500) {
    Serial.println("Girando fuerte a la derecha (por luz lateral
derecha).");
    girarSuaveDerecha();
  }
  else if (luzLI > luzLD && luzLI > 2500) {
    Serial.println("Girando fuerte a la izquierda (por luz lateral
izquierda).");
    girarSuaveIzquierda();
  }
  else {
    // Si no domina lado, considerar frente vs atrás
    int luzFrente = (luzFD + luzFI) / 2;
    int luzAtras = (luzAD + luzAI) / 2;

    if (luzFrente > luzAtras) {
      Serial.println("Avanzando hacia el frente.");
      avanzar();
    } else {
      Serial.println("Girando 180 grados (más luz atrás).");
```

```
      girar180();
    }
  }

  delay(500); // Tiempo entre decisiones
}

// FUNCIONES DE MOVIMIENTO

void avanzar() {
  digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW);  // Trasero Izq
  digitalWrite(IN3, HIGH); digitalWrite(IN4, LOW);  // Trasero Der
  digitalWrite(IN5, HIGH); digitalWrite(IN6, LOW);  // Delantero Izq
  digitalWrite(IN7, HIGH); digitalWrite(IN8, LOW);  // Delantero Der
}

void girarSuaveDerecha() {
  digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW);  // Trasero Izq adelante
  digitalWrite(IN5, LOW);  digitalWrite(IN6, HIGH); // Delantero Izq
adelante
  digitalWrite(IN3, LOW);  digitalWrite(IN4, LOW);  // Trasero Der detenido
  digitalWrite(IN7, LOW);  digitalWrite(IN8, LOW);  // Delantero Der
detenido
}

void girarSuaveIzquierda() {
  digitalWrite(IN3, HIGH); digitalWrite(IN4, LOW);  // Trasero Der adelante
  digitalWrite(IN7, LOW);  digitalWrite(IN8, HIGH); // Delantero Der
adelante
  digitalWrite(IN1, LOW);  digitalWrite(IN2, LOW);  // Trasero Izq detenido
  digitalWrite(IN5, LOW);  digitalWrite(IN6, LOW);  // Delantero Izq
detenido
}

void girar180() {
  // Igual que girar fuerte derecha pero más tiempo o directamente igual
  girarSuaveDerecha();
}

void detenerTodo() {
  digitalWrite(IN1, LOW); digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW); digitalWrite(IN4, LOW);
  digitalWrite(IN5, LOW); digitalWrite(IN6, LOW);
  digitalWrite(IN7, LOW); digitalWrite(IN8, LOW);
}
```