ITESO, Universidad Jesuita de Guadalajara

PRÁCTICA 4: SISTEMA AUTÓNOMO REACTIVO CON SENSOR LiDAR C1

Implementación de detección, decisión y evasión con control cinemático en tiempo real

Luis Eduardo Diaz Macias, Pablo Perez Sanchez, Miguel de Jesus Flores Gonzalez & Jesus Alejandro Ocegueda Melin

Título de la práctica	Práctica 4
Autor o Responsable	Luis Eduardo Diaz Macias, Pablo Perez Sanchez, Miguel de Jesus Flores Gonzalez & Jesus Alejandro Ocegueda Melin
Asignatura	PAP
Fecha de entrega	16 de octubre de 2025
Objetivo	Desarrollar un sistema autónomo reactivo para la evitación de obstáculos mediante la integración del sensor LiDAR C1 con el control cinemático del robot. Validar la detección, decisión y actuación del sistema bajo baja latencia.
Materiales y Recursos	• Raspberry Pi 5 • RPLIDAR C1 • Motores Mecanum controlados por I2C • Fuente 5V 3A • Laptop con Python 3.11
Duración Estimada	8 horas
Lugar o Modalidad	Laboratorio de Mecatrónica

Resumen— Esta práctica aborda la implementación de un sistema autónomo reactivo para la evitación de obstáculos en un carrito omnidireccional utilizando un sensor LiDAR. Se implementa la cadena completa de procesamiento (adquisición, filtrado, decisión y actuación), integrando la lectura del sensor con el control cinemático de los motores vía I2C. El sistema utiliza una lógica de máquina de estados (*Avanzando*, *Girando*) para asegurar una respuesta rápida y robusta ante la detección de objetos en una zona frontal crítica. La librería `Pygame` se usa como herramienta de verificación visual en tiempo real del barrido y del estado interno del robot.

---

1. Introducción

El objetivo es lograr que el robot opere de forma autónoma, detectando y reaccionando a un obstáculo para evitar colisiones. A diferencia de la práctica con visión computacional (ArUco), este sistema se basa en la detección local y en tiempo real del LiDAR C1. Se implementa un control de lazo cerrado solo en la etapa de evasión y un lazo abierto para la cinemática, priorizando una respuesta de baja latencia para frenado y giro.

2. Configuración de Parámetros Críticos

Tabla 2: Parámetros de detección y decisión

Parámetro	Valor	Unidad	Función
MIN_DISTANCE	50	mm	Ignora ruido cercano.
MAX_DISTANCE	3000	mm	Limita la zona útil de trabajo.
OBSTACLE_THRESHOLD	800	mm	Umbral para activar evasión.
FRONT_ANGLE_RANGE	60	grados	Rango crítico frontal.
Velocidad_Avance	100	mm/s	Movimiento lineal en estado *avanzando*.
Velocidad_Giro	50	grados/s	Velocidad angular constante en *girando*.

Los motores se controlan mediante el bus I2C en la dirección `0x34`, y la comunicación del LiDAR se realiza a través de `/dev/ttyUSB0` con `baudrate 460800`. La cinemática utiliza una matriz de transformación W para calcular el PWM de cada motor.

3. Preparación del Entorno y Librerías

Creación de Carpeta de Trabajo y Ambiente Virtual

```
mkdir ~/LiDAR_C1_Project
cd ~/LiDAR_C1_Project

# Crear ambiente virtual
python3.11 -m venv lidar_env
source lidar_env/bin/activate  # Linux/Mac
# lidar_env\Scripts\Activate.ps1  # Windows
```

Instalación de Librerías

```
pip install numpy
pip install pygame
pip install pyserial
pip install smbus2
```

Clonado de Librería Oficial de RPLIDAR

```
git clone https://github.com/Slamtec/rplidar_ros.git
cd rplidar_ros
```

- Ubicar el archivo `pyrplidar_protocole.py` dentro de la carpeta de Python 3.11 (`lib64/python3.11/site-packages/`).
- Modificar la línea:

```
RPLIDAR_CMD_FORCE_SCAN = b'\x21'
# por
RPLIDAR_CMD_FORCE_SCAN = b'\x20'
```

para que funcione con LiDAR C1.

Clonado del Código de la Práctica

```
git clone https://github.com/LuisDiazMac/Practicas-PAP.git
cd Practicas-PAP/PRACTICA-04/
```

4. Selección de Dirección Libre

El robot utiliza la información del LiDAR para determinar la dirección más libre cuando detecta un obstáculo frontal:

```
frontal = distancias[centro-FRONT_ANGLE_RANGE : centro+FRONT_ANGLE_RANGE]
if min(frontal) < OBSTACLE_THRESHOLD:
    izquierda = min(distancias[0:centro-1])
    derecha = min(distancias[centro+1:-1])
    if izquierda > 1000 and izquierda >= derecha:
        direccion = 'izquierda'
    elif derecha > 1000 and derecha > izquierda:
        direccion = 'derecha'
    else:
        direccion = 'detener'
else:
    direccion = 'avanzar'
```

5. Uso del Código de la Práctica

Funcionalidad Principal del Script `Autonomo_LiDAR.py`

- Inicializa LiDAR C1 y motores Mecanum mediante I2C.
- Configura parámetros críticos (`MIN_DISTANCE`, `MAX_DISTANCE`, `OBSTACLE_THRESHOLD`, `FRONT_ANGLE_RANGE`).
- Detecta obstáculos frontales y decide dirección más libre (izquierda/derecha) o se detiene si no hay espacio suficiente.
- Calcula y envía PWM a los motores con la cinemática omnidireccional.
- Visualiza en `pygame` los puntos, zona frontal y estado del robot.
- Permite detener el sistema con Ctrl+C, apagando motores y liberando recursos.

Ejecución del Script

```
source ~/LiDAR_C1_Project/lidar_env/bin/activate  # Linux/Mac
# lidar_env\Scripts\Activate.ps1  # Windows PowerShell

python Autonomo_LiDAR.py
```

6. Implementación del Sistema de Control y Detección

Detección de Obstáculos

1. Recolección de puntos dentro de `MIN_DISTANCE` y `MAX_DISTANCE`.
2. Filtrado por zona frontal (± FRONT_ANGLE_RANGE).
3. Confirmación del obstáculo cuando más de 5 puntos están por debajo de `OBSTACLE_THRESHOLD`.
4. Determinación de la dirección libre comparando distancias mínimas laterales.

Lógica de Máquina de Estados

Tabla 3: Transiciones de la máquina de estados

Estado	Condición de Transición	Acción del Motor
Avanzando	Detección de obstáculo	Detener motores → Girando.
Girando	Tiempo de giro completado	Avanzando.

7. Protocolo de Pruebas y Visualización

Protocolo de Pruebas

Tabla 4: Ensayos de validación del sistema

Ensayo	Escenario	Comportamiento Esperado	Criterio de Aceptación
Respuesta a Obstáculo Frontal	El robot avanza hacia una pared o caja.	Se detiene y gira antes del umbral.	Cambio inmediato de estado a *Girando*.
Evasión Direccional	El robot avanza hacia una esquina.	Gira hacia el lado más libre.	Giro hacia el lado con mayor distancia mínima.
Robustez de Ciclo	Entorno cerrado con múltiples objetos.	Mantiene el ciclo avance–giro–avance.	Sin fallos de comunicación LiDAR/I2C.

Visualización y Monitoreo

La interfaz gráfica en `Pygame` muestra:

- Puntos del LiDAR en coordenadas cartesianas.
- Zona crítica frontal (arco blanco de radio `OBSTACLE_THRESHOLD`).
- Puntos en rojo dentro del umbral de peligro.
- Estado actual del robot (*AVANZANDO* / *GIRANDO*) y distancia mínima detectada.

8. Conclusiones y Desafíos Futuros

Conclusiones del Aprendizaje

Se logró la integración hardware–software de un sistema de evasión reactivo. El análisis por ciclo de barrido garantiza decisiones coherentes y la máquina de estados permite movimientos predecibles. La identificación de la dirección libre (izquierda/derecha) es esencial para una navegación funcional.

Desafíos y Próximos Pasos

1. **Medición de Latencia (T_{reac}):** instrumentar *logging* para medir el tiempo entre detección y orden de giro.
2. **Integración con navegación global:** conectar esta lógica de evasión con la navegación por objetivos desarrollada previamente.
3. **Filtrado y robustez:** aplicar promedios móviles o filtros estadísticos para evitar falsas detecciones por ruido.