

Sistema de Gestión Polimórfica de Sensores para IoT

César Euresti
Estructura de Datos

30 de octubre de 2025

1. Introducción

El presente documento describe la solución implementada para el caso de estudio de la unidad: un sistema de monitoreo polimórfico que administra sensores heterogéneos (temperatura y presión) empleando listas enlazadas simples sin depender de la STL. El objetivo principal es registrar lecturas, procesarlas con lógica específica por sensor y liberar memoria de forma segura, todo mientras se conserva un único contenedor polimórfico y se habilita la integración con un dispositivo Arduino/ESP32 mediante puerto serial.

2. Manual Técnico

2.1. Diseño General

- **Jerarquía de Sensores:** La clase abstracta `SensorBase` define la interfaz común (`imprimirInfo()`, `registrarLecturaInteractiva()`, `registrarLecturaDesdeCadena()` y `procesarLectura()`). Las clases derivadas `SensorTemperatura` y `SensorPresion` implementan la lógica específica para lecturas `float` e `int`, respectivamente, incluyendo la eliminación del mínimo (temperatura) y el cálculo de promedio (presión).
- **Contenedor Polimórfico:** `ListaGeneral` gestiona punteros a `SensorBase` mediante una lista enlazada manual. Se asegura que cada sensor se inserte solo una vez y al liberar la lista se invocan los destructores correctos gracias al destructor virtual.
- **Listas Genéricas:** Las lecturas internas utilizan `ListaSensor<T>` y `Nodo<T>`, plantillas que implementan inserción al final, búsqueda, extracción, cálculo de promedio y regla de los tres (constructor de copia, asignación y destructor). Esto cumple con el requisito de manejar memoria dinámica sin STL.
- **Interacción CLI:** `AuxiliarCli` centraliza la captura validada de datos y el formateo de logs coloreados (`STATUS`, `WARNING`, `SUCCESS`). Los mensajes `ERROR` se reservan para cierres críticos, como se estipuló.

2.2. Flujo de Ejecución

El módulo principal (`src/main.cpp`) ofrece un menú que permite:

1. Crear sensores de temperatura o presión, solicitando un identificador textual.
2. Registrar lecturas de tres formas: (a) captura manual con validación, (b) pegando una línea `ID,valor` en consola o (c) escuchando directamente un puerto serial conectado a Arduino/ESP32.
3. Procesar polimórficamente todas las lecturas registradas.

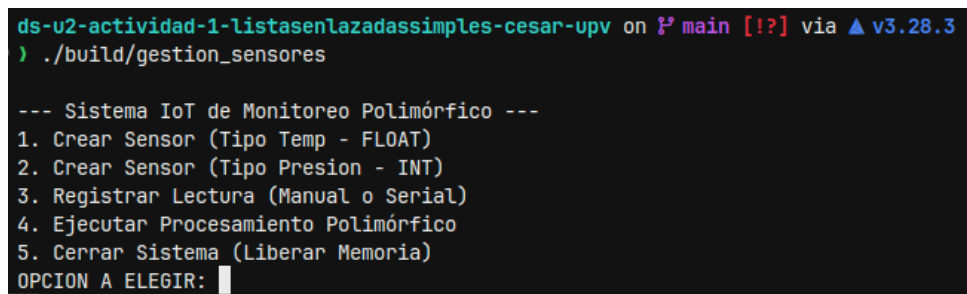
4. Liberar la memoria de manera controlada antes de finalizar.

La compatibilidad con Arduino/ESP32 se valida con el sketch `arduino/SerialEmitter.ino`, que emite periódicamente lecturas en el formato requerido. El programa principal parsea estas cadenas y delega el registro a cada sensor; en modo de escucha directa se utilizan las funciones POSIX (`open`, `termios`, `select`) para leer en tiempo real y permitir al usuario detener el proceso pulsando `ENTER` o desconectando el dispositivo. Antes de iniciar la escucha se solicita la ruta del puerto (`/dev/ttyUSB0`, `/dev/ttyACM0`, etc.) y se recuerda cerrar otros monitores seriales para evitar conflictos.

2.3. Componentes Relevantes

- **CMake:** El archivo `CMakeLists.txt` compila el binario `gestion_sensores`, incluyendo el directorio `include/`.
- **Doxygen:** La configuración `Doxyfile` genera la documentación HTML dentro de `docs/html/`, con comentarios breves en cada archivo y método.
- **Simulación y Lectura Serial:** El proyecto añade la carpeta `arduino/` con el sketch que produce lecturas de ejemplo y, en el binario principal, una opción de escucha que abre el puerto (por ejemplo `/dev/ttyUSB0`), configura los parámetros (9600 baudios, 8N1) y procesa cada línea válida. Se alerta al usuario para cerrar monitores seriales externos y se reportan desconexiones y errores de formato mediante `AuxiliarCli`.
- **Manejo de Memoria:** Todos los destructores informan su actividad con `AuxiliarCli`, evidenciando la liberación de nodos internos y evitando fugas.

3. Evidencia de Funcionamiento



```
ds-u2-actividad-1-listasenlazadassimples-cesar-upv on P main [!?] via ▲ v3.28.3
) ./build/gestion_sensores

--- Sistema IoT de Monitoreo Polimórfico ---
1. Crear Sensor (Tipo Temp - FLOAT)
2. Crear Sensor (Tipo Presion - INT)
3. Registrar Lectura (Manual o Serial)
4. Ejecutar Procesamiento Polimórfico
5. Cerrar Sistema (Liberar Memoria)
OPCION A ELEGIR: █
```

Figura 1: Estado inicial del menú principal al ejecutar el binario.

```

--- Sistema IoT de Monitoreo Polimórfico ---
1. Crear Sensor (Tipo Temp - FLOAT)
2. Crear Sensor (Tipo Presion - INT)
3. Registrar Lectura (Manual o Serial)
4. Ejecutar Procesamiento Polimórfico
5. Cerrar Sistema (Liberar Memoria)
OPCION A ELEGIR: 1
ID del sensor de temperatura: TEMP02
[SUCCESS] Sensor 'TEMP02' insertado en la lista de gestión.

--- Sistema IoT de Monitoreo Polimórfico ---
1. Crear Sensor (Tipo Temp - FLOAT)
2. Crear Sensor (Tipo Presion - INT)
3. Registrar Lectura (Manual o Serial)
4. Ejecutar Procesamiento Polimórfico
5. Cerrar Sistema (Liberar Memoria)
OPCION A ELEGIR: 3

1. Registrar lectura manual
2. Registrar lectura desde cadena serial
Seleccione modo: 1
ID del sensor destino: TEMP02
Valor de temperatura (float): 53.35
[STATUS] Insertando nodo float en TEMP02.

```

Figura 2: Creación de un sensor y registro de lecturas manuales utilizando AuxiliarCli.

```

--- Sistema IoT de Monitoreo Polimórfico ---
1. Crear Sensor (Tipo Temp - FLOAT)
2. Crear Sensor (Tipo Presion - INT)
3. Registrar Lectura (Manual o Serial)
4. Ejecutar Procesamiento Polimórfico
5. Cerrar Sistema (Liberar Memoria)
OPCION A ELEGIR: 4
[STATUS] --- Ejecutando Polimorfismo ---
[STATUS] -> Procesando Sensor TEMP02...
[STATUS] [TEMP02] Lectura más baja (20.4) eliminada.
[STATUS] [Sensor Temp] Promedio calculado sobre 1 lectura (53.3).

```

Figura 3: Ejecución del procesamiento polimórfico con la lógica de cada derivada.

```
--- Sistema IoT de Monitoreo Polimórfico ---
1. Crear Sensor (Tipo Temp - FLOAT)
2. Crear Sensor (Tipo Presion - INT)
3. Registrar Lectura (Manual o Serial)
4. Ejecutar Procesamiento Polimórfico
5. Cerrar Sistema (Liberar Memoria)
OPCION A ELEGIR: 5
[STATUS] --- Liberación de Memoria en Cascada ---
[STATUS] [Destructor General] Liberando Nodo: TEMP02.
[STATUS] [Destructor Sensor TEMP02] Liberando Lista Interna...
[STATUS]     Nodo<float> 53.3 liberado.
[SUCCESS] Sistema cerrado. Memoria limpia.
```

Figura 4: Liberación ordenada de memoria al cerrar el sistema.

4. Conclusiones

La solución cumple con los requerimientos funcionales y no funcionales descritos en el `README.md`:

- Uso estricto de listas enlazadas manuales y plantillas para gestionar lecturas.
- Jerarquía polimórfica con métodos virtuales puros y destructores seguros.
- Integración con entradas seriales (manuales y en vivo) y logs uniformes mediante `AuxiliarCli`.
- Documentación automatizada, estructura modular (`include/`, `src/`) y soporte CMake.

Se recomienda, como trabajo futuro, añadir pruebas automatizadas y ampliar la jerarquía con nuevos tipos de sensores para demostrar extensibilidad.