Sistema de Gestión Polimórfica de Sensores para IoT con Listas Enlazadas

Estructura de datos - Unidad 2

Diego Eduardo Zapata Aguilar
Ingeniería en Tecnologías de la Información
Universidad Politécnica de Victoria
Semestre: Sep-Dic 2025

***Resumen*—Este proyecto presenta la implementación de un sistema de gestión polimórfica de sensores IoT utilizando estructuras de datos dinámicas (listas enlazadas simples) en C++. El sistema integra sensores de temperatura y presión con comunicación serial a través de Arduino, demostrando conceptos fundamentales de programación orientada a objetos como herencia, polimorfismo y gestión dinámica de memoria. Se implementaron contenedores genéricos mediante plantillas (templates) para el almacenamiento eficiente de lecturas, y se desarrolló una interfaz de usuario interactiva por consola. El proyecto incluye documentación técnica completa generada con Doxygen y organización modular del código fuente.**

I. Introducción

El Internet de las Cosas (IoT) ha transformado la manera en que los dispositivos interactúan y recopilan datos del entorno. En este contexto, el desarrollo de sistemas capaces de gestionar múltiples tipos de sensores de forma eficiente y escalable es fundamental para aplicaciones de monitoreo en tiempo real.

Este proyecto aborda el diseño e implementación de un sistema de gestión de sensores que aprovecha las características de la programación orientada a objetos para crear una arquitectura flexible y extensible. El uso de listas enlazadas simples permite la gestión dinámica de sensores sin limitaciones de tamaño predefinido, mientras que el polimorfismo facilita el procesamiento uniforme de diferentes tipos de sensores.

I-A. Objetivos

- Implementar estructuras de datos dinámicas (listas enlazadas simples) para gestión de sensores
- Aplicar conceptos de herencia y polimorfismo en un contexto práctico
- Desarrollar comunicación serial con Arduino para adquisición de datos
- Demostrar el uso de plantillas (templates) para código genérico y reutilizable
- Generar documentación técnica profesional mediante Doxygen

II. Manual Técnico

II-A. Diseño del Sistema

El sistema está diseñado siguiendo principios de programación orientada a objetos, con una arquitectura de tres capas:

II-A1. Capa de Abstracción de Sensores: La clase base abstracta `SensorBase` define la interfaz común para todos los tipos de sensores:

```
class SensorBase {
public:
    SensorBase(const char* nom);
    virtual ~SensorBase();
    virtual void procesarLectura() = 0;
    virtual void imprimirInfo() const = 0;
protected:
    char nombre[50];
};
```

Esta clase proporciona:

- Métodos virtuales puros para garantizar implementación en clases derivadas
- Destructor virtual para correcta liberación de memoria polimórfica
- Encapsulamiento de datos comunes (nombre del sensor)

II-A2. Capa de Especialización: Las clases concretas `SensorTemperatura` y `SensorPresion` heredan de `SensorBase` y especializan el comportamiento:

- **SensorTemperatura**: Maneja lecturas tipo `float`, calcula promedios y elimina valores mínimos
- **SensorPresion**: Maneja lecturas tipo `int`, calcula estadísticas básicas

Cada sensor mantiene un historial de lecturas mediante la plantilla `ListaSensor<T>`.

II-A3. Capa de Gestión: La clase `ListaGeneral` actúa como contenedor principal:

```
class ListaGeneral {
private:
    Nodo<SensorBase*>* cabeza;
public:
    void insertar(SensorBase* valor);
    void procesarTodos();
    SensorBase* buscarPorNombre(const char* nombre);
};
```

Implementa operaciones de:

- Inserción dinámica de sensores
- Recorrido polimórfico para procesamiento
- Búsqueda por identificador
- Liberación automática de memoria en cascada

II-B. Componentes del Sistema

II-B1. Estructuras de Datos: **Nodo Genérico (`Nodo.h`)**
Plantilla de nodo para listas enlazadas simples:

```
template <typename T>
class Nodo {
public:
    Nodo<T>* siguiente;
    T valor;
    Nodo(T valor);
};
```

Lista de Sensores (`ListaSensor.h`)
Implementación genérica con operaciones de:

- Inserción al final: $O(n)$
- Búsqueda de mínimo: $O(n)$
- Eliminación de valor: $O(n)$
- Cálculo de promedio: $O(n)$

II-B2. Comunicación Serial: **Clase ArduinoReader**
Gestiona la comunicación POSIX con Arduino mediante `termios.h`:

- Configuración a 9600 baudios
- Lectura línea por línea no bloqueante
- Sincronización post-reset del Arduino
- Gestión automática de recursos (RAII)

Protocolo de comunicación:

```
FORMATO: SENSOR_ID,VALOR\n
EJEMPLO: T-001,25.3
         P-105,1015
```

II-B3. Sketch de Arduino: El firmware simula dos sensores con las siguientes características:

Tabla I: Especificaciones de sensores simulados

ID	Tipo	Rango	Tipo Dato
T-001	Temperatura	20.0-29.9°C	float
P-105	Presión	1000-1020 hPa	int

II-C. Desarrollo

II-C1. Herramientas Utilizadas:

- **Compilador**: GCC 13.3.0 (C++11)
- **Sistema de Build**: CMake 3.28
- **IDE**: CLion / Visual Studio Code
- **Documentación**: Doxygen 1.9.8
- **Control de Versiones**: Git
- **Hardware**: Arduino Uno (ATmega328P)

II-C2. Estructura del Proyecto:

```
proyecto/
|-- include/          # Archivos de encabezado (.h)
|-- src/              # Implementaciones (.cpp)
|-- build/            # Ejecutables compilados
|-- arduino/          # Firmware Arduino
|-- docs/             # Documentacion Doxygen
+-- reporte_latex/    # Documentacion LaTeX
```

II-C3. Proceso de Compilación:

```
# Configurar con CMake
cmake .

# Compilar
make -j2

# Ejecutar
./build/ListasEnlazadasSimples
```

II-C4. Generación de Documentación:

```
# Generar docs con Doxygen
doxygen DoxyFile

# Abrir documentacion
xdg-open docs/html/index.html
```

III. Implementación y Resultados

III-A. Interfaz de Usuario

El sistema presenta un menú interactivo por consola con las siguientes opciones:

1. Crear Sensor de Temperatura (float)
2. Crear Sensor de Presión (int)
3. Registrar Lectura desde Arduino
4. Ejecutar Procesamiento Polimórfico
5. Cerrar Sistema (liberación de memoria)
6. Registro Manual de Lectura

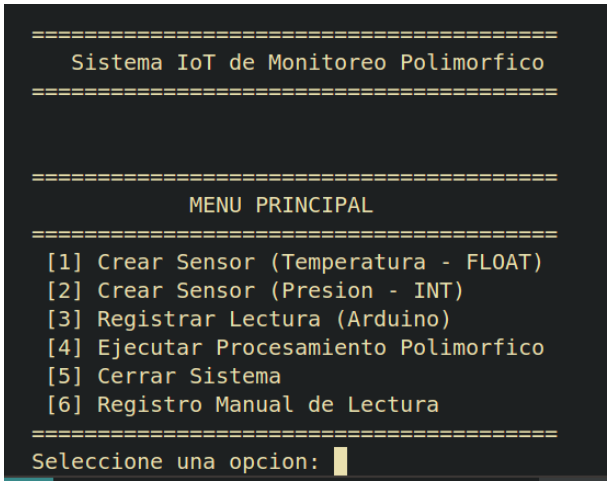

Figura 1: Menú principal del sistema de gestión de sensores

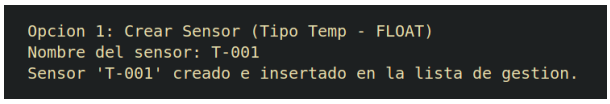

Figura 2: Proceso de creación de sensores de temperatura y presión

III-B. Validación de Funcionalidad

El sistema fue validado mediante:

- **Pruebas de memoria**: Verificación con Valgrind (sin memory leaks)

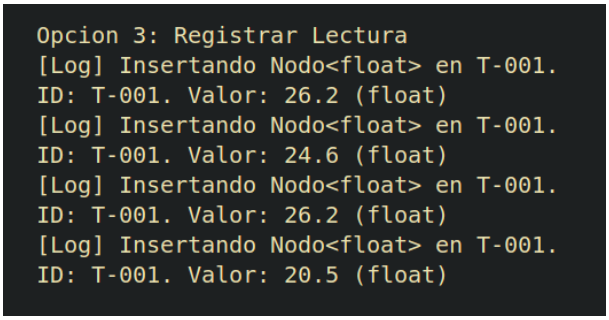

Figura 3: Adquisición de datos desde Arduino vía puerto serial

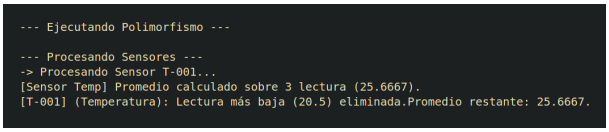

Figura 4: Ejecución del procesamiento polimórfico de sensores

- **Pruebas de comunicación**: Correcta recepción de datos desde Arduino
- **Pruebas de polimorfismo**: Procesamiento diferenciado por tipo de sensor
- **Pruebas de robustez**: Manejo de errores de conexión y datos inválidos

III-C. Documentación Generada

La documentación Doxygen incluye:

- Jerarquía de clases con diagramas de herencia
- Documentación de cada método con parámetros y valores de retorno
- Ejemplos de uso y notas de implementación
- Índice alfabético de símbolos
- Código fuente navegable con resaltado de sintaxis

IV. Conclusiones

Se logró implementar exitosamente un sistema de gestión polimórfica de sensores IoT que demuestra:

1. **Estructuras dinámicas**: Las listas enlazadas permiten gestión flexible sin límites predefinidos
2. **Polimorfismo**: El procesamiento uniforme de diferentes tipos de sensores mediante interfaces abstractas
3. **Plantillas**: El código genérico reutilizable para diferentes tipos de datos
4. **Comunicación serial**: Integración efectiva con hardware externo (Arduino)
5. **Gestión de memoria**: RAII y destrucción en cascada garantizan ausencia de memory leaks

El proyecto cumple con todos los objetivos planteados y constituye una base sólida para sistemas de monitoreo IoT más complejos. La arquitectura modular facilita la extensión con nuevos tipos de sensores y funcionalidades adicionales.

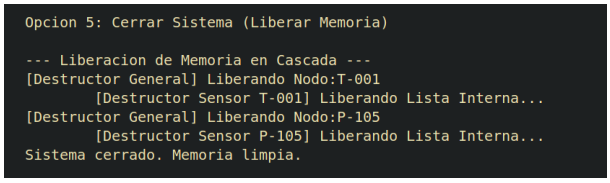

Figura 5: Liberación de memoria en cascada al cerrar el sistema

IV-A. Trabajo Futuro

Posibles mejoras incluyen:

- Implementación de persistencia de datos (archivos/base de datos)
- Interfaz gráfica con Qt o GTK+
- Soporte para múltiples Arduinos simultáneos
- Análisis estadístico avanzado de lecturas
- Alertas y umbrales configurables