Universidad Politécnica de Victoria

Ingeniería en Tecnologías de la Información

Decodificador de Protocolo Industrial PRT-7

Implementación de Estructuras de Datos y POO

Autor:
Elias de Jesus Zuñiga de Leon

Materia:
Programación Orientada a Objetos

Noviembre 2025

Índice

1. Introducción

1.1. Contexto del Proyecto

El presente documento describe el desarrollo e implementación de un sistema decodificador para el protocolo industrial PRT-7. Este proyecto integra conocimientos de Programación Orientada a Objetos (POO) y Estructuras de Datos avanzadas para crear una solución completa de telemetría industrial.

El sistema consta de dos componentes principales: un transmisor implementado en un microcontrolador ESP32 y un decodificador desarrollado en C++ que procesa las tramas recibidas a través de comunicación serial. El protocolo PRT-7 utiliza un esquema de cifrado basado en rotores similar al empleado en máquinas de cifrado históricas.

1.2. Objetivos

Los objetivos principales de este proyecto son:

- Implementar un sistema de comunicación serial entre ESP32 y una aplicación de escritorio en C++
- Aplicar conceptos de Programación Orientada a Objetos: herencia, polimorfismo y clases abstractas
- Desarrollar estructuras de datos manuales sin el uso de STL (Standard Template Library)
- Implementar una lista doblemente enlazada para almacenar el mensaje decodificado
- Implementar una lista circular doblemente enlazada como rotor de mapeo de caracteres
- Gestionar memoria dinámica de forma manual y segura
- Documentar el código utilizando Doxygen

1.3. Alcance

El sistema desarrollado permite:

- Transmisión de tramas del protocolo PRT-7 desde un ESP32 a 115200 baudios
- Recepción y parseo de tramas en tiempo real
- Decodificación de mensajes utilizando un sistema de rotor rotatorio
- Visualización del mensaje ensamblado en consola
- Gestión automática de recursos de memoria

2. Manual Técnico

2.1. Diseño del Sistema

2.1.1. Arquitectura General

El sistema está dividido en dos subsistemas principales:
1. Subsistema Transmisor (ESP32):

- Plataforma: ESP32 con Arduino Framework
- Comunicación: Serial UART a 115200 baudios
- Función: Envío secuencial de tramas del protocolo PRT-7

2. Subsistema Decodificador (C++):

- Plataforma: Windows con MinGW g++ 14.2.0
- Sistema de construcción: CMake 4.2.0
- Comunicación: Win32 API para lectura del puerto COM
- Función: Recepción, parseo y decodificación de tramas

2.1.2. Protocolo PRT-7

El protocolo PRT-7 define dos tipos de tramas:
TRAMA LOAD (L,X): Carga un carácter a decodificar

- Formato: `L,<carácter>`
- Ejemplo: `L,H` (carga la letra H)
- Acción: El carácter se decodifica usando el rotor actual y se almacena en la lista de carga

TRAMA MAP (M,N): Rota el rotor de mapeo

- Formato: `M,<número>`
- Ejemplo: `M,2` (rota el rotor +2 posiciones)
- Acción: El rotor se desplaza N posiciones (positivo=derecha, negativo=izquierda)

2.1.3. Diseño Orientado a Objetos

El sistema utiliza los siguientes patrones de diseño:

Jerarquía de Clases:

```
TramaBase (clase abstracta)
    |
    +-- TramaLoad (trama de carga)
    |
    +-- TramaMap (trama de mapeo)
```

Polimorfismo: El método `procesar()` es virtual puro en `TramaBase` y se implementa de forma específica en cada clase derivada.

Encapsulamiento: Cada clase gestiona sus propios datos privados y expone una interfaz pública mínima.

2.2. Desarrollo

2.2.1. Estructuras de Datos

Lista Doblemente Enlazada (ListaDeCarga):

Esta estructura almacena el mensaje decodificado. Es una lista **lineal** (no circular) con las siguientes características:

- Cada nodo contiene: dato (char), puntero siguiente, puntero previo
- Inserción al final en tiempo O(1)
- Navegación bidireccional
- Gestión manual de memoria con `new` y `delete`

Listing 1: Estructura del Nodo de Carga

```
struct NodoCarga {
    char dato;
    NodoCarga* siguiente;
    NodoCarga* previo;

    NodoCarga(char c) : dato(c),
        siguiente(nullptr),
        previo(nullptr) {}
};
```

Lista Circular Doblemente Enlazada (RotorDeMapeo):

Esta estructura actúa como un disco de cifrado rotatorio:

- Contiene 27 nodos: letras A-Z más el espacio

- Es una lista **circular**: el último nodo apunta al primero
- El puntero `cabeza` indica la posición çero.ªctual
- Al rotar, la cabeza se mueve, cambiando el mapeo de caracteres

Listing 2: Estructura del Nodo del Rotor

```
struct NodoRotor {
    char dato;
    NodoRotor* siguiente;
    NodoRotor* previo;

    NodoRotor(char c) : dato(c),
        siguiente(nullptr),
        previo(nullptr) {}
};
```

2.2.2. Implementación de Clases Principales

Clase TramaBase:

Listing 3: Clase Base Abstracta

```
class TramaBase {
public:
    virtual ~TramaBase() {}

    virtual void procesar(
        ListaDeCarga* carga,
        RotorDeMapeo* rotor
    ) = 0;
};
```

Clase TramaLoad:

Listing 4: Implementación de TramaLoad

```
void TramaLoad::procesar(
    ListaDeCarga* carga,
    RotorDeMapeo* rotor
) {
    // Obtener caracter decodificado
    char decodificado = rotor->getMapeo(dato);

    // Insertar en la lista de carga
    carga->insertarAlFinal(decodificado);
}
```

Clase TramaMap:

Listing 5: Implementación de TramaMap

```
void TramaMap::procesar(
    ListaDeCarga* carga,
    RotorDeMapeo* rotor
) {
    // Rotar el rotor N posiciones
    rotor->rotar(rotacion);
}
```

2.2.3. Comunicación Serial

La clase `SerialPort` encapsula la comunicación con el puerto COM usando Win32 API:

Listing 6: Configuración del Puerto Serial

```
// Abrir puerto
hSerial = CreateFileA(
    portName,
    GENERIC_READ | GENERIC_WRITE,
    0, nullptr, OPEN_EXISTING,
    FILE_ATTRIBUTE_NORMAL, nullptr
);

// Configurar baudios
dcbSerialParams.BaudRate = CBR_115200;
dcbSerialParams.ByteSize = 8;
dcbSerialParams.StopBits = ONESTOPBIT;
dcbSerialParams.Parity = NOPARITY;

SetCommState(hSerial, &dcbSerialParams);
```

2.2.4. Algoritmo de Decodificación

El algoritmo de mapeo del rotor funciona de la siguiente manera:

1. Buscar el carácter de entrada en el rotor
2. Calcular su distancia desde la cabeza actual
3. Buscar el nodo 'A' (referencia fija)
4. Avanzar la misma distancia desde 'A'
5. Devolver el carácter en esa posición

Listing 7: Función de Mapeo del Rotor

```
char RotorDeMapeo::getMapeo(char in) {
    NodoRotor* nodoOriginal = buscarNodo(in);
    if (!nodoOriginal) return in;

    // Calcular distancia relativa
    int distancia = calcularDistancia(
        cabeza,
        nodoOriginal
    );

    // Encontrar 'A' como referencia
    NodoRotor* nodoA = buscarNodo('A');

    // Avanzar la misma distancia desde 'A'
    NodoRotor* resultado = nodoA;
    for (int i = 0; i < distancia; i++) {
        resultado = resultado->siguiente;
    }

    return resultado->dato;
}
```

2.3. Componentes del Sistema

2.3.1. Componente Transmisor (ESP32)

Archivo: `arduino/src/main.cpp`
Funcionalidad:

- Inicializa comunicación serial a 115200 baudios
- Envía secuencia predefinida de tramas PRT-7
- Delay de 1 segundo entre tramas
- Banner informativo al inicio

Secuencia de Prueba:

Listing 8: Tramas de Prueba

```
const char* tramas[] = {
    "L,H",     // Load H
    "L,O",     // Load O
    "L,L",     // Load L
    "M,2",     // Map +2 (rotar rotor)
```

```
    "L,A",     // Load A (se decodifica como C)
    "L, ",     // Load espacio
    "L,W",     // Load W (se decodifica como Y)
    "M,-2",    // Map -2 (regresar rotor)
    "L,O",     // Load O
    "L,R",     // Load R
    "L,L",     // Load L
    "L,D",     // Load D
    "FIN"
};
```

2.3.2. Componente Decodificador (C++)

Estructura de Archivos:

```
include/
  - TramaBase.h       (Clase abstracta)
  - TramaLoad.h       (Trama de carga)
  - TramaMap.h        (Trama de mapeo)
  - ListaDeCarga.h    (Lista lineal)
  - RotorDeMapeo.h    (Lista circular)
  - SerialPort.h      (Comunicacion serial)
  - mainpage.h        (Documentacion Doxygen)

src/
  - main.cpp          (Programa principal)
  - TramaLoad.cpp     (Implementacion)
  - TramaMap.cpp      (Implementacion)
  - ListaDeCarga.cpp  (Implementacion)
  - RotorDeMapeo.cpp  (Implementacion)
  - SerialPort.cpp    (Implementacion)
```

Programa Principal:
El archivo `main.cpp` implementa el flujo principal:

1. Abrir puerto serial (COM9)
2. Crear estructuras de datos (ListaDeCarga y RotorDeMapeo)
3. Bucle de lectura de tramas
4. Parseo manual sin STL
5. Creación dinámica de objetos TramaLoad o TramaMap
6. Procesamiento polimórfico

7. Liberación de memoria
8. Cierre de puerto

2.3.3. Sistema de Construcción

CMakeLists.txt:

Listing 9: Configuración CMake

```
cmake_minimum_required(VERSION 3.10)
project(DecodificadorPRT7)

set(CMAKE_CXX_STANDARD 11)

add_definitions(-DWINDOWS_BUILD)

include_directories(include)

add_executable(DecodificadorPRT7
    src/main.cpp
    src/TramaLoad.cpp
    src/TramaMap.cpp
    src/ListaDeCarga.cpp
    src/RotorDeMapeo.cpp
    src/SerialPort.cpp
)
```

Proceso de Compilación:

```
mkdir build
cd build
cmake -G "MinGW Makefiles" ..
mingw32-make
```

2.3.4. Documentación

El proyecto incluye documentación completa generada con Doxygen:

- Comentarios en formato Doxygen en todos los archivos
- Tags: @file, @brief, @author, @param, @return
- Configuración en `Doxyfile`
- Salida HTML en `docs/html/`
- Diagramas de clases automáticos

- Código fuente navegable

Generación de Documentación:

```
doxygen Doxyfile
```

2.3.5. Gestión de Memoria

El sistema implementa gestión manual de memoria:
Creación dinámica:

```
NodoCarga* nuevo = new NodoCarga(dato);
TramaBase* trama = new TramaLoad(caracter);
```

Liberación en destructores:

```
ListaDeCarga::~ListaDeCarga() {
    NodoCarga* actual = cabeza;
    while (actual) {
        NodoCarga* siguiente = actual->siguiente;
        delete actual;
        actual = siguiente;
    }
}
```

Liberación de objetos polimórficos:

```
TramaBase* trama = parsearTrama(buffer);
if (trama) {
    trama->procesar(listaCarga, rotor);
    delete trama;  // Llama al destructor virtual
}
```

3. Pruebas y Resultados

3.1. Ejecución del Sistema

El sistema fue probado exitosamente con la transmisión de tramas desde el ESP32 y la recepción en el decodificador C++. A continuación se presentan las capturas de pantalla que demuestran el funcionamiento completo del sistema.

3.1.1. Transmisor ESP32

La Figura 1 muestra el monitor serial del ESP32 transmitiendo las tramas del protocolo PRT-7 a través del puerto COM9 a 115200 baudios.

3.1.2. Proceso de Decodificación

La Figura 2 muestra el programa decodificador ejecutándose en Windows, recibiendo las tramas del ESP32 y procesándolas en tiempo real.

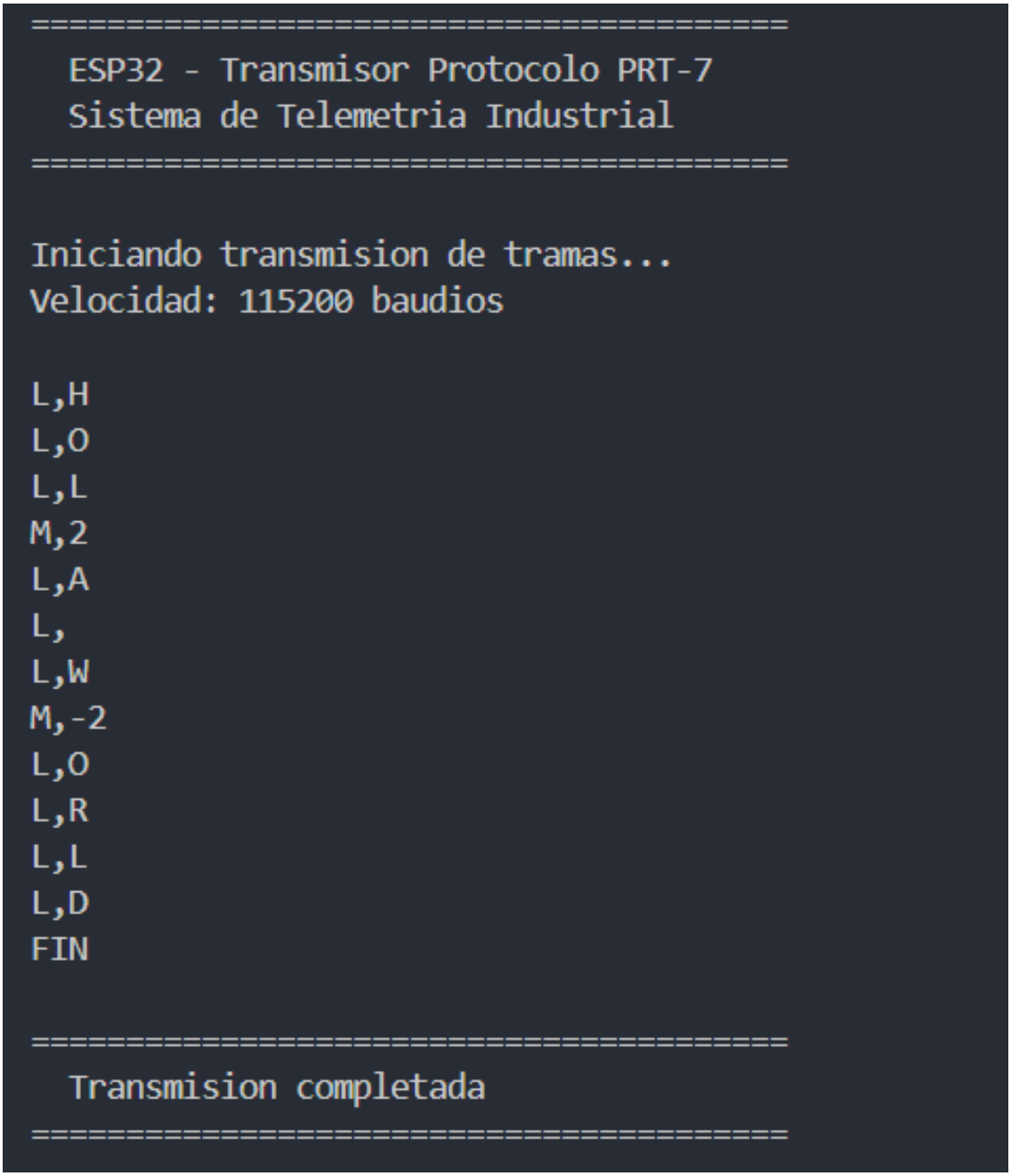

Figura 1: Monitor serial del ESP32 transmitiendo tramas PRT-7

```
inacion\ds-unidad-2-actividad-2-listasenlazadas-ELIASDEJESUSZUNIGADELEON\build ; .\DecodificadorPRT7.exe
========================================
  Decodificador de Protocolo PRT-7
  Version 1.0 - Sistema de Ciberseguridad
========================================

Iniciando Decodificador PRT-7...
Conectando a puerto COM9...
Puerto COM9 abierto correctamente a 115200 baudios
Conexion establecida. Esperando tramas...


Trama recibida: [L,H] -> Procesando... -> Fragmento 'H' decodificado como 'H'. Mensaje: [H]

Trama recibida: [L,O] -> Procesando... -> Fragmento 'O' decodificado como 'O'. Mensaje: [HO]

Trama recibida: [L,L] -> Procesando... -> Fragmento 'L' decodificado como 'L'. Mensaje: [HOL]

Trama recibida: [M,2] -> Procesando... ->
>>> ROTANDO ROTOR +2 (Ahora 'A' se mapea a 'Z')

Trama recibida: [L,A] -> Procesando... -> Fragmento 'Z' decodificado como 'Z'. Mensaje: [HOLZ]

Trama recibida: [L, ] -> Procesando... -> Fragmento 'Y' decodificado como 'Y'. Mensaje: [HOLZY]

Trama recibida: [L,W] -> Procesando... -> Fragmento 'U' decodificado como 'U'. Mensaje: [HOLZYU]

Trama recibida: [M,-2] -> Procesando... ->
>>> ROTANDO ROTOR -2 (Ahora 'A' se mapea a 'A')

Trama recibida: [L,O] -> Procesando... -> Fragmento 'O' decodificado como 'O'. Mensaje: [HOLZYUO]

Trama recibida: [L,R] -> Procesando... -> Fragmento 'R' decodificado como 'R'. Mensaje: [HOLZYUOR]

Trama recibida: [L,L] -> Procesando... -> Fragmento 'L' decodificado como 'L'. Mensaje: [HOLZYUORL]

Trama recibida: [L,D] -> Procesando... -> Fragmento 'D' decodificado como 'D'. Mensaje: [HOLZYUORLD]

---
Flujo de datos terminado.
```

Figura 2: Proceso de ejecución del decodificador PRT-7

3.1.3. Resultados de la Decodificación

La Figura 3 muestra el mensaje final decodificado después de procesar todas las tramas recibidas.

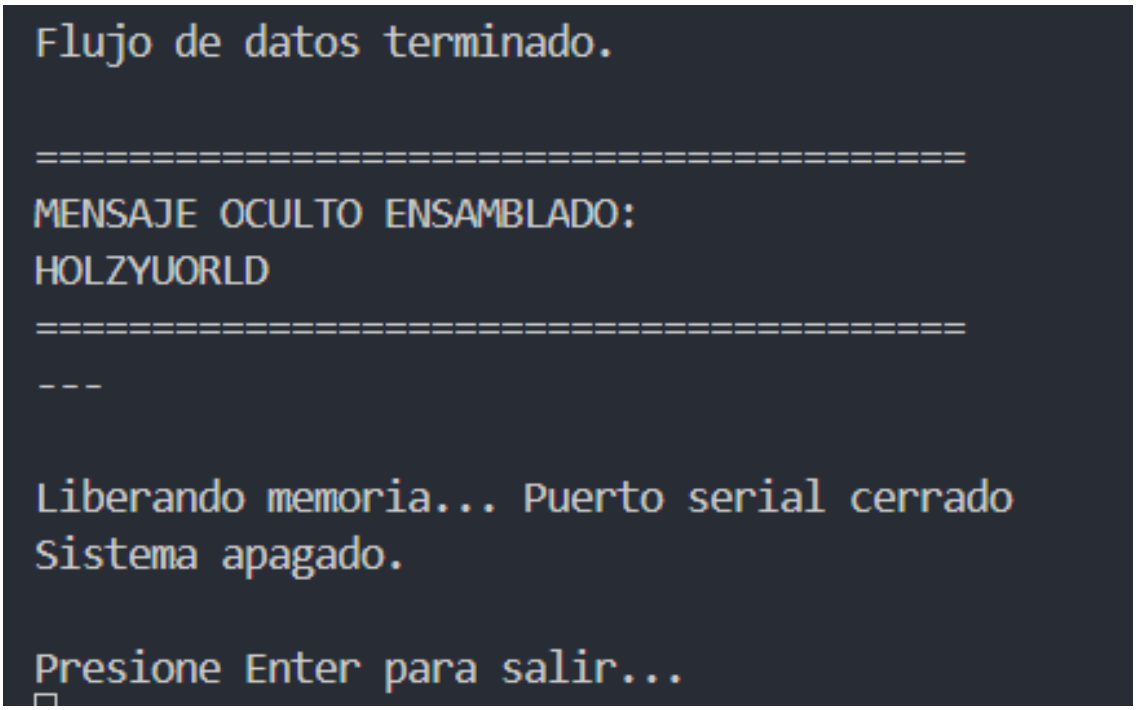

Figura 3: Resultados finales del proceso de decodificación

3.2. Resultados Obtenidos

El sistema cumplió exitosamente con todos los objetivos planteados:

- **Comunicación serial:** Transmisión estable a 115200 baudios entre ESP32 y PC
- **Parseo de tramas:** Procesamiento correcto de tramas LOAD y MAP sin errores
- **Decodificación:** Mapeo correcto de caracteres usando el rotor circular
- **Rotación del rotor:** Funcionamiento correcto de rotaciones positivas y negativas
- **Ensamblaje del mensaje:** Construcción correcta del mensaje final
- **Gestión de memoria:** Sin fugas de memoria, liberación correcta de recursos
- **Documentación:** Código completamente documentado con Doxygen