Caso de Estudio: Decodificador de Protocolo Industrial (PRT-7) Reporte

Rodrigo Damian Alvarez Aguilar
Universidad Politécnica de Victoria
Ingeniería en Tecnologías de la Información e Innovación Digital

6 de noviembre de 2025

Introducción

Este proyecto implementa un decodificador simple del protocolo PRT-7 usando C++ y un Arduino real. El Arduino no envía el mensaje final, sino instrucciones para construirlo:

- Tramas **LOAD** (`L,X`): traen un carácter. Se decodifica con un "rotor" y se agrega al final del mensaje.
- Tramas **MAP** (`M,N`): rotan el rotor N posiciones (positivas o negativas), cambiando el mapeo de letras de forma dinámica.

El objetivo es leer las líneas por el puerto serie, aplicar la lógica de mapeo y armar el mensaje con el orden correcto, sin usar estructuras de la STL.

Manual técnico

Diseño

Arquitectura general

El programa tiene tres partes:

- **Lectura serie** (Win32): abre el puerto
  .
  `COMx`, configura el baud rate y lee líneas terminadas en `|`.
- **POO de tramas**: una clase base y dos derivadas para procesar cada línea.
- **Estructuras de datos manuales**: una lista doble para el mensaje y una lista circular doble para el rotor A–Z.

Clases principales

- **TramaBase**: clase abstracta con el método `procesar(...)` y destructor virtual.
- **TramaLoad**: guarda un carácter y, al procesar, pide al rotor su mapeo y lo inserta al final de la lista del mensaje.
- **TramaMap**: guarda un entero. Al procesar, rota el rotor esa cantidad de pasos.
- **ListaDeCarga** (lista doble): guarda los caracteres decodificados en orden. Métodos: `insertarAlFinal`, `imprimirMensaje`, `imprimirMensajeBrackets`.
- **RotorDeMapeo** (lista circular doble): contiene 26 nodos con letras `'A'..'Z'`. Mantiene un puntero *head* que indica la "posición cero". Métodos: `rotar(int)`, `getMapeo(char)`.
- **SerialPort**: envoltorio mínimo de Win32 para abrir, configurar y leer del puerto serie.

Lógica del rotor

- La lista se arma una vez con las letras A–Z y se cierra en círculo.
- Al inicio, `head` apunta a 'A' (`'A'` → `'A'`).
- Al rotar $+N$, `head` avanza N posiciones. Al rotar $-N$, `head` retrocede.
- Para mapear un carácter de entrada `in` (A–Z), se toma el desplazamiento `in - 'A'` y se devuelve la letra que está en `head + desplazamiento` (módulo 26). Otros caracteres (como espacio) se devuelven igual.

Desarrollo

Código fuente y organización

- Encabezados: `include/listas.h`, `include/tramas.h`, `include/serial.h`
- Fuentes: `src/listas.cpp`, `src/tramas.cpp`, `src/serial`
  $_w in32.cpp$, `src/main.cpp`
- `Construcción: CMakeLists.txt (genera con "MinGW Makefiles").`
- `Documentación: Doxyfile (genera HTML en docs/html).`
- `Arduino: arduino/prt7`
  $_s ender/prt7$
  $_s ender.ino(envalastramasdeejemplo)$.

Compilación en Windows (MinGW)

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

El ejecutable queda como `prt7.exe` en la carpeta `build`.

Ejecución

Conecta el Arduino (por ejemplo en COM3) y ejecuta:

```
./prt7 --com COM3 --baud 9600
```

Para terminar y ver el mensaje final, el Arduino envía la línea `END`.

Flujo de funcionamiento

1. Se inicializa la lista del mensaje vacía y el rotor con A–Z (`head` en 'A').
2. Se abre el puerto serial y se leen líneas como `L,H` o `M,2`.
3. Si llega `M,N`, se rota el rotor. Si llega `L,X`, se calcula el mapeo y se agrega el resultado al final del mensaje.
4. Al recibir `END`, se imprime el mensaje completo.

Componentes

Hardware

- Una placa Arduino (UNO/Nano/MEGA u otra compatible).
- Cable USB para la conexión con la PC.

Software

- Windows 10/11.
- CMake (generador "MinGW Makefiles").
- MinGW (`mingw32-make` y `gcc`).
- IDE de Arduino (para cargar el `.ino`).
- Doxygen (opcional, para generar documentación en HTML).

Notas de calidad

- No se usan contenedores de la STL. Se manejan punteros y memoria con `new/delete`.
- Los destructores liberan todos los nodos para evitar fugas de memoria.
- Se ignoran tramas mal formadas y se continúa con las siguientes.