

Decodificador PRT-7

Sistema de Comunicación Serial Arduino-PC

Manual Técnico

November 6, 2025

Contents

1	Introducción	2
1.1	Objetivo del Sistema	2
1.2	Características Principales	2
1.3	Contexto de Uso	2
2	Manual Técnico	3
2.1	Diseño del Sistema	3
2.1.1	Arquitectura General	3
2.1.2	Protocolo PRT-7	3
2.1.3	Diseño de Clases	4
2.2	Desarrollo	4
2.2.1	Tecnologías Utilizadas	4
2.2.2	Restricciones de Diseño	4
2.2.3	Flujo de Ejecución	5
2.3	Componentes del Sistema	7
2.3.1	Componente 1: SerialPortWin32	7
2.3.2	Componente 2: RotorDeMapeo	7
2.3.3	Componente 3: ListaDeCarga	8
2.3.4	Componente 4: Sistema de Tramas	9
2.3.5	Componente 5: Emisor Arduino	10
2.3.6	Diagrama de Secuencia	10
2.4	Compilación y Ejecución	11
2.4.1	Requisitos del Sistema	11
2.4.2	Proceso de Compilación	11
2.4.3	Ejecución del Sistema	12
2.5	Ejemplo de Salida	13
3	Conclusiones	13

1 Introducción

El proyecto **PRT-7 Decoder** es un sistema de comunicación y decodificación de mensajes que implementa el protocolo PRT-7. Este sistema consta de dos componentes principales: un emisor basado en Arduino que transmite datos codificados a través del puerto serial, y un receptor desarrollado en C++ para Windows que decodifica estos mensajes utilizando un algoritmo de cifrado tipo rotor César.

1.1 Objetivo del Sistema

El objetivo principal del sistema es demostrar la implementación de un protocolo de comunicación serial personalizado que utiliza técnicas de cifrado por sustitución. El sistema es capaz de:

- Establecer comunicación serial bidireccional entre Arduino y PC
- Transmitir tramas de datos en un formato específico (protocolo PRT-7)
- Decodificar mensajes utilizando un rotor de mapeo dinámico
- Gestionar estructuras de datos sin utilizar la librería estándar de C++ (STL)

1.2 Características Principales

1. **Comunicación Serial Win32:** Implementación directa usando la API de Windows para máximo control y mínimas dependencias.
2. **Estructuras de Datos Manuales:** Lista doblemente enlazada para almacenar los caracteres decodificados y lista circular para el rotor de mapeo.
3. **Protocolo Personalizado:** Formato de tramas simple pero efectivo con dos tipos de comandos (L y M).
4. **Cifrado Dinámico:** Rotor de mapeo que cambia su estado durante la decodificación para mayor complejidad.
5. **Documentación Doxygen:** Código comentado profesionalmente para facilitar mantenimiento y comprensión.

1.3 Contexto de Uso

Este sistema es ideal para aplicaciones educativas en comunicaciones digitales, sistemas embebidos y algoritmos de cifrado. También puede servir como base para proyectos más complejos de IoT que requieran comunicación segura entre dispositivos.

2 Manual Técnico

2.1 Diseño del Sistema

2.1.1 Arquitectura General

El sistema sigue una arquitectura cliente-servidor donde el Arduino actúa como emisor (servidor) y el programa C++ como receptor (cliente). La comunicación se realiza a través de puerto serial COM utilizando el protocolo UART.

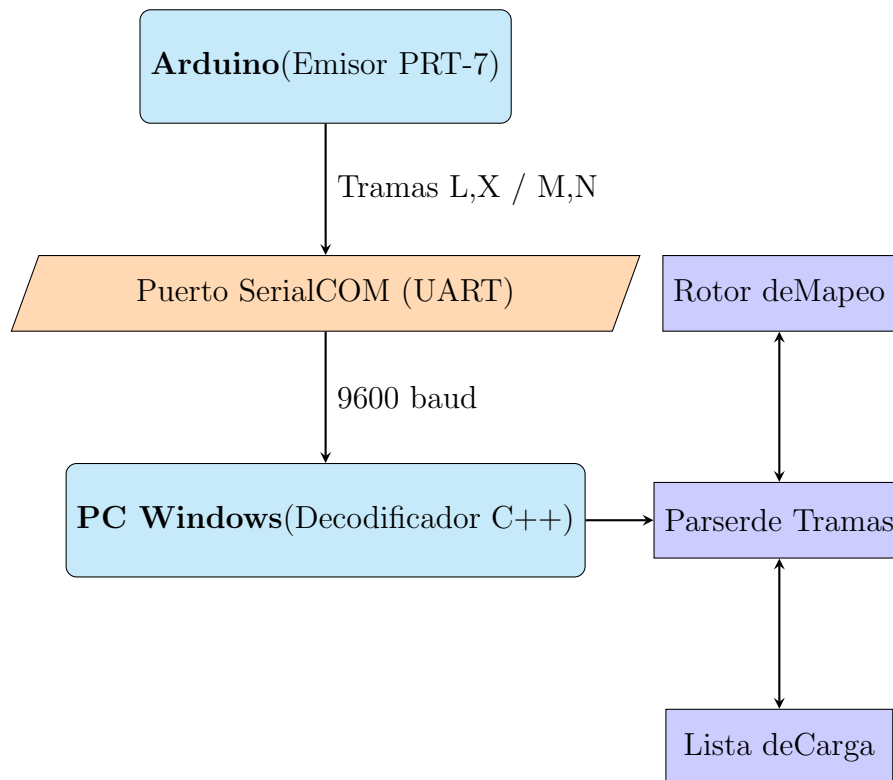


Figure 1: Arquitectura general del sistema PRT-7

2.1.2 Protocolo PRT-7

El protocolo define dos tipos de tramas transmitidas como líneas de texto terminadas en salto de línea:

- **Trama de Carga (L,X):** Indica que se debe cargar el carácter X, decodificarlo usando el rotor actual y agregarlo al mensaje.
 - Ejemplo: L,H carga la letra 'H'
 - Especial: L,Space carga un espacio
- **Trama de Mapeo (M,N):** Indica que se debe rotar el rotor N posiciones (positivo o negativo).
 - Ejemplo: M,2 rota el rotor 2 posiciones adelante
 - Ejemplo: M,-2 rota el rotor 2 posiciones atrás

2.1.3 Diseño de Clases

El sistema utiliza programación orientada a objetos con las siguientes clases principales:

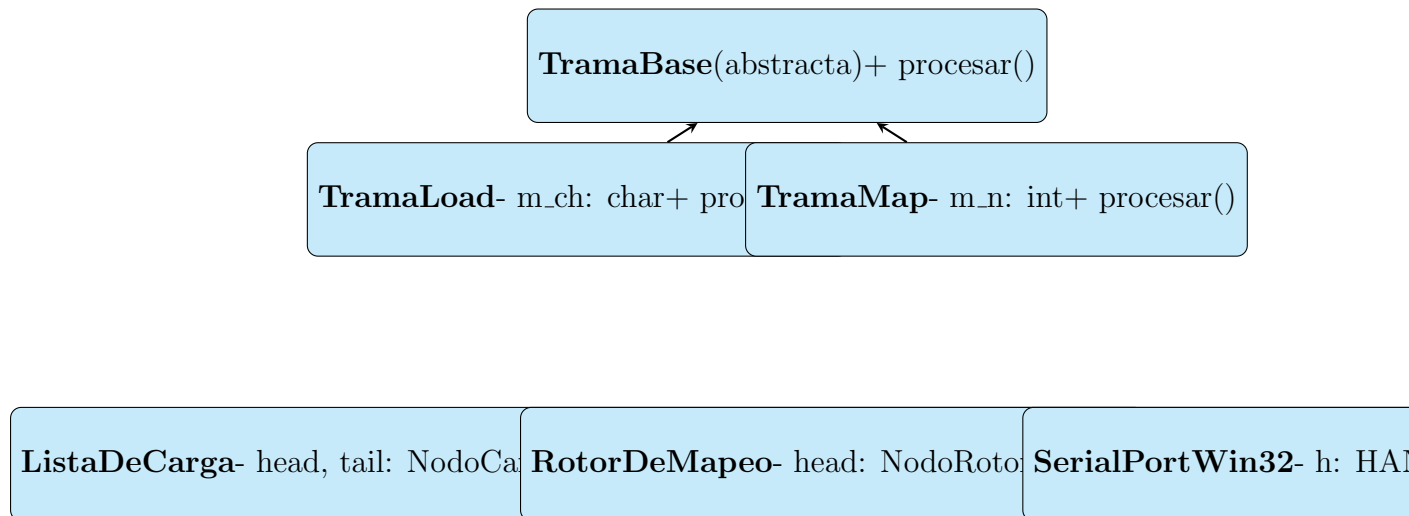


Figure 2: Diagrama de clases del sistema

2.2 Desarrollo

2.2.1 Tecnologías Utilizadas

- **Lenguajes:** C++ (receptor), C/Arduino (emisor)
- **Compilador:** MinGW (g++) para Windows
- **Sistema de Build:** CMake
- **API:** Win32 API para comunicación serial
- **Hardware:** Arduino (cualquier modelo con puerto serial)
- **Documentación:** Doxygen

2.2.2 Restricciones de Diseño

El desarrollo siguió las siguientes restricciones importantes:

1. **Sin STL:** No se utilizan contenedores de la librería estándar (`std::vector`, `std::string`, etc.)
2. **Gestión Manual de Memoria:** Todas las estructuras dinámicas se manejan con `new/delete`
3. **API Win32 Directa:** Comunicación serial sin librerías de terceros
4. **Estructuras Personalizadas:** Implementación propia de listas enlazadas

2.2.3 Flujo de Ejecución

El siguiente diagrama muestra el flujo principal del programa decodificador:

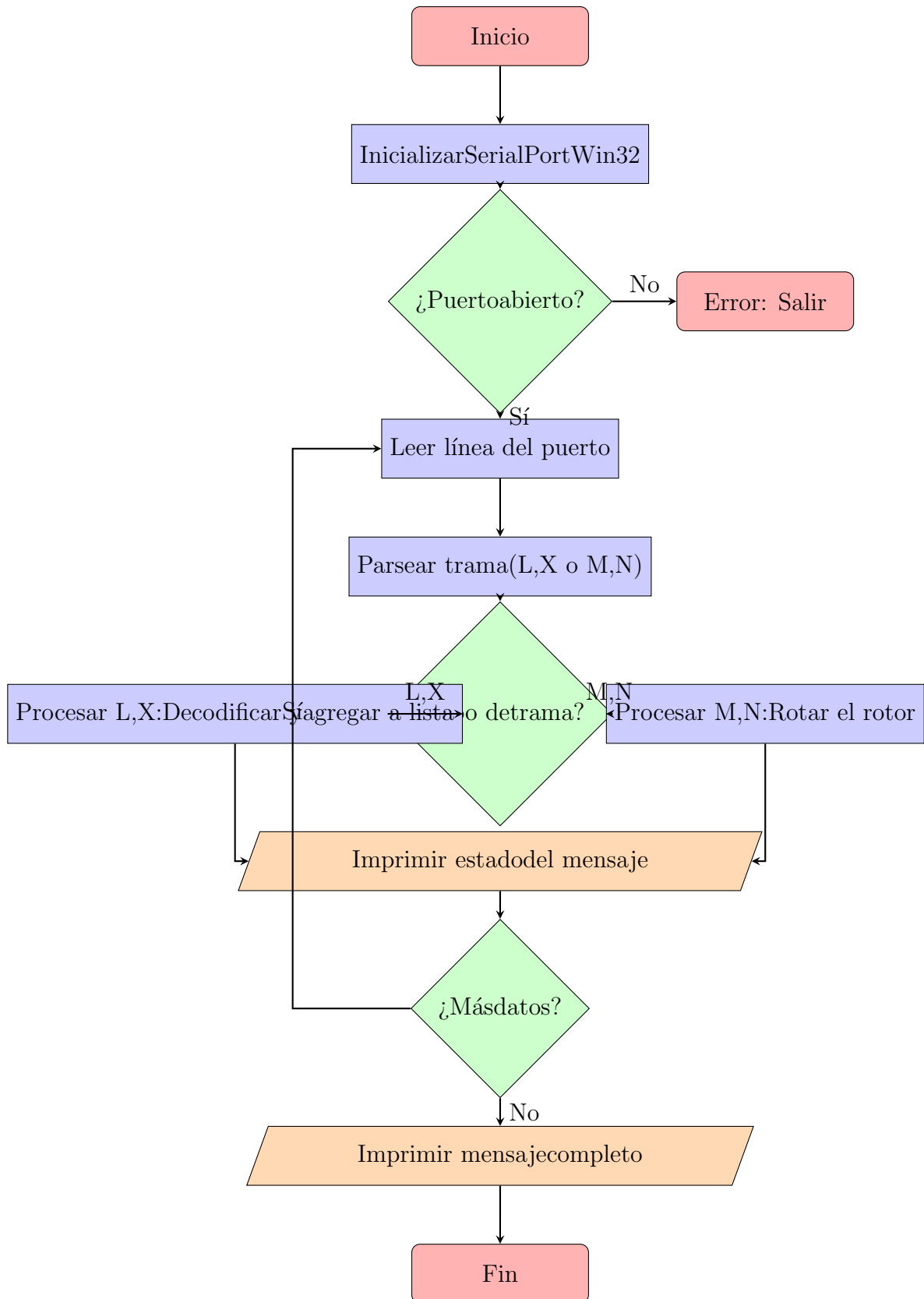


Figure 3: Diagrama de flujo del decodificador

2.3 Componentes del Sistema

2.3.1 Componente 1: SerialPortWin32

Propósito: Gestionar la comunicación serial con el Arduino usando la API de Windows.

Funcionalidades principales:

- Apertura y cierre del puerto COM
- Configuración de velocidad de transmisión (baud rate)
- Lectura de líneas completas con detección de `\n`
- Manejo de errores de comunicación

Implementación clave:

```
1 bool SerialPortWin32::open(const char* portName, unsigned long baud) {
2     h = CreateFileA(portName, GENERIC_READ | GENERIC_WRITE,
3                     0, NULL, OPEN_EXISTING, 0, NULL);
4     if (h == INVALID_HANDLE_VALUE) return false;
5
6     DCB dcb = {0};
7     dcb.DCBlength = sizeof(dcb);
8     GetCommState(h, &dcb);
9     dcb.BaudRate = baud;
10    dcb.ByteSize = 8;
11    dcb.Parity = NOPARITY;
12    dcb.StopBits = ONESTOPBIT;
13    SetCommState(h, &dcb);
14    return true;
15 }
```

Listing 1: Apertura del puerto serial

2.3.2 Componente 2: RotorDeMapeo

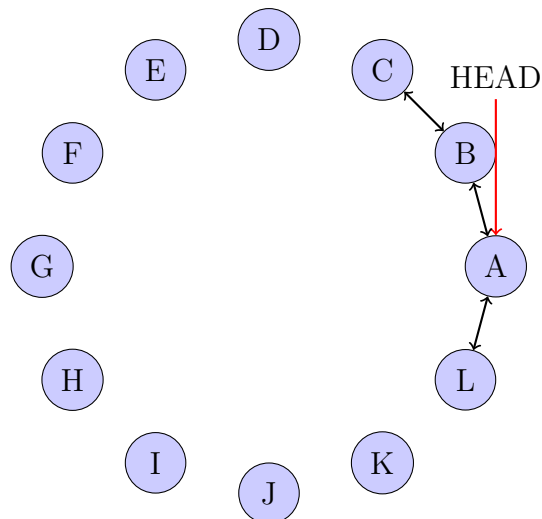
Propósito: Implementar un rotor de cifrado tipo César con lista circular.

Estructura de datos:

- Lista circular doblemente enlazada con 26 nodos (A-Z)
- Puntero `head` que marca la posición actual (offset cero)
- Rotación modifica el offset desplazando el `head`

Algoritmo de mapeo:

1. Convertir carácter de entrada a mayúscula
2. Si no es A-Z, devolver sin cambios
3. Calcular índice: `idx = (char - 'A')`
4. Avanzar `idx` posiciones desde `head`
5. Devolver el carácter almacenado en ese nodo



Ejemplo simplificado: Lista circular con 12 elementos

Figure 4: Estructura del rotor de mapeo (simplificado)

2.3.3 Componente 3: ListaDeCarga

Propósito: Almacenar los caracteres decodificados en orden secuencial.

Estructura:

- Lista doblemente enlazada
- Punteros `head` y `tail` para inserción eficiente
- Cada nodo contiene un carácter decodificado

Operaciones:

- `insertarAlFinal(char c)`: Añade carácter al final
- `imprimirMensajeEnBrackets()`: Muestra `[c1] [c2] ... [cn]`
- `imprimirMensajePlano()`: Muestra `c1c2...cn`

```

1 void ListaDeCarga::insertarAlFinal(char c) {
2     NodoCarga* nuevo = (NodoCarga*)malloc(sizeof(NodoCarga));
3     nuevo->dato = c;
4     nuevo->next = NULL;
5     nuevo->prev = tail;
6
7     if (!head) {
8         head = tail = nuevo;
9     } else {
10        tail->next = nuevo;
11        tail = nuevo;
12    }
13    size_++;
14 }

```

Listing 2: Inserción en lista de carga

2.3.4 Componente 4: Sistema de Tramas

Jerarquía de clases:

- TramaBase: Clase abstracta con método `procesar()`
- TramaLoad: Procesa tramas de carga (L,X)
- TramaMap: Procesa tramas de mapeo (M,N)

Patrón de diseño: Strategy pattern permite procesar diferentes tipos de tramas polimórficamente.

```
1 TramaBase* trama = parseLine(line);  
2 if (trama) {  
3     trama->procesar(&lista, &rotor);  
4     delete trama;  
5 }
```

Listing 3: Procesamiento polimórfico de tramas

2.3.5 Componente 5: Emisor Arduino

Propósito: Transmitir secuencia predefinida de tramas PRT-7.

Características:

- Inicialización del puerto serial a 9600 baud
- Transmisión secuencial con delay de 1 segundo
- Array predefinido de tramas
- Detención automática al finalizar

```
1 void loop() {  
2     const char* frames[] = {  
3         "L,H", "L,0", "L,L", "M,2",  
4         "L,A", "L,Space", "L,W", "M,-2",  
5         "L,0", "L,R", "L,L", "L,D"  
6     };  
7     for (unsigned i = 0; i < sizeof(frames)/sizeof(frames[0]); ++i) {  
8         Serial.println(frames[i]);  
9         delay(1000);  
10    }  
11    while (true) { delay(1000); }  
12 }
```

Listing 4: Loop principal del Arduino

2.3.6 Diagrama de Secuencia

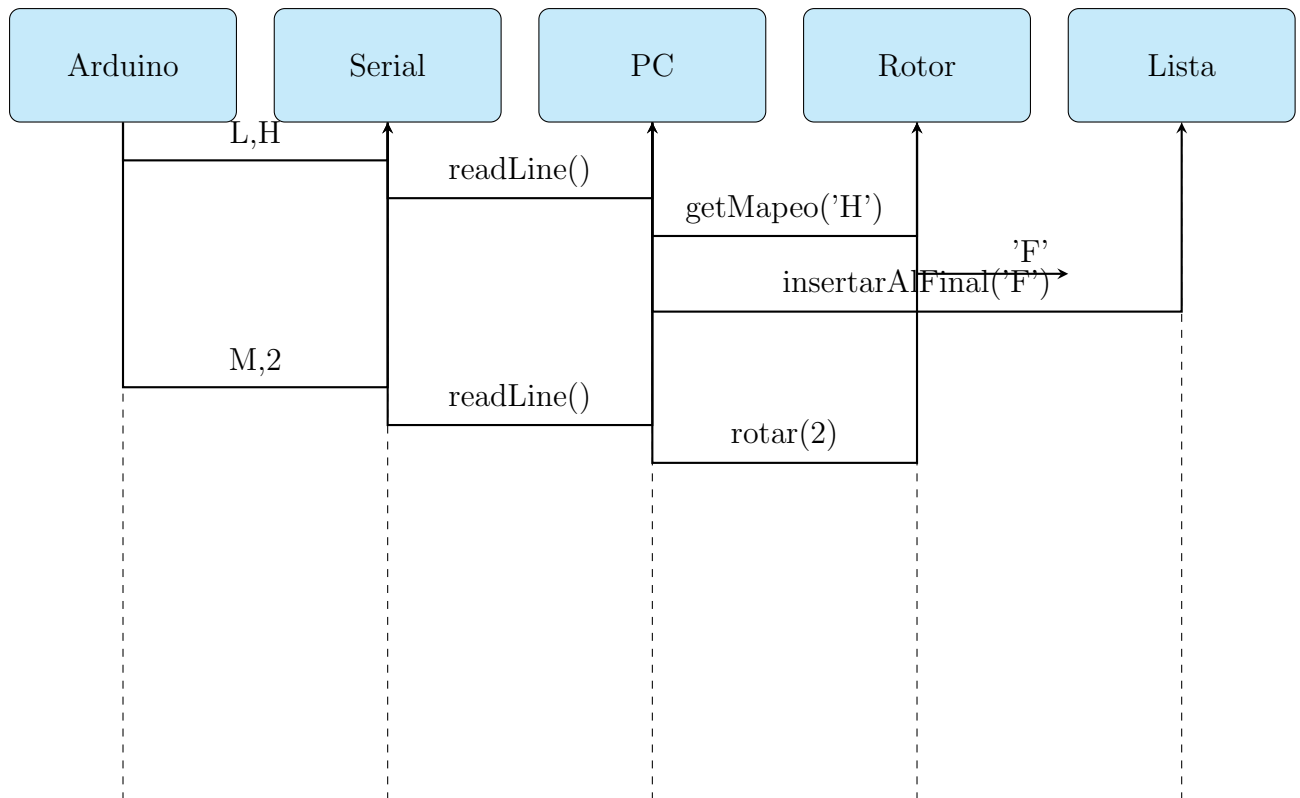


Figure 5: Diagrama de secuencia del procesamiento de tramas

2.4 Compilación y Ejecución

2.4.1 Requisitos del Sistema

- **Sistema Operativo:** Windows 10 o superior
- **Compilador:** MinGW (g++) con soporte C++11
- **Build System:** CMake 3.10 o superior
- **Hardware:** Arduino con puerto USB
- **Drivers:** Drivers del puerto serial del Arduino

2.4.2 Proceso de Compilación

1. Clonar o descomprimir el proyecto
2. Abrir terminal en el directorio raíz
3. Ejecutar los siguientes comandos:

```

1 mkdir build
2 cd build
3 cmake .. -G "MinGW_Makefiles"
4 mingw32-make

```

Listing 5: Comandos de compilación

El ejecutable `p7.exe` se generará en el directorio `build/`.

2.4.3 Ejecución del Sistema

Paso 1: Cargar código en Arduino

1. Abrir `arduino_prt7_sender.ino` en Arduino IDE
2. Conectar Arduino por USB
3. Compilar y cargar el programa
4. Verificar el puerto COM asignado (Herramientas → Puerto)

Paso 2: Ejecutar decodificador

```
1 ./prt7.exe COM3 9600
```

Donde `COM3` es el puerto del Arduino y `9600` es el baud rate.

Nota: Para puertos mayores a `COM9`, usar la notación: `\\.\COM10`

2.5 Ejemplo de Salida

```
1 Iniciando Decodificador PRT-7. Conectando a puerto COM...
2 Conexion establecida. Esperando tramas...
3
4 (Presiona Ctrl+C para detener)
5
6 Trama recibida: [L,H] -> Procesando... -> Fragmento 'H'
7 decodificado como 'F'. Mensaje: [F]
8
9 Trama recibida: [L,0] -> Procesando... -> Fragmento 'O'
10 decodificado como 'M'. Mensaje: [F][M]
11
12 Trama recibida: [L,L] -> Procesando... -> Fragmento 'L'
13 decodificado como 'J'. Mensaje: [F][M][J]
14
15 Trama recibida: [M,2] -> Procesando... -> ROTANDO ROTOR +2.
16
17 Trama recibida: [L,A] -> Procesando... -> Fragmento 'A'
18 decodificado como 'C'. Mensaje: [F][M][J][C]
19
20 Trama recibida: [L,Space] -> Procesando... -> Fragmento ' '
21 decodificado como ' '. Mensaje: [F][M][J][C][ ]
22
23 ---
24 Flujo de datos terminado.
25 MENSAJE OCULTO ENSAMBLADO:
26 FMJ WORLD
27 ---
28 Liberando memoria... Sistema apagado.
```

3 Conclusiones

El sistema PRT-7 Decoder demuestra exitosamente la implementación de:

1. **Comunicación serial eficiente** entre sistemas embebidos y PC sin librerías de alto nivel
2. **Estructuras de datos personalizadas** optimizadas para el problema específico
3. **Algoritmo de cifrado dinámico** con complejidad variable durante la decodificación
4. **Código modular y documentado** siguiendo principios de ingeniería de software

El sistema puede expandirse para incluir:

- Múltiples rotores (máquina Enigma)
- Protocolo bidireccional con confirmaciones
- Cifrado AES para mayor seguridad
- Interfaz gráfica para visualización en tiempo real