UNIVERSIDAD DE ANTIOQUIA

Módulo de comunicación para la máquina de juegos Silver Route

Jhon Fredy Mercado Agudelo

Informe de trabajo de grado como requisito para optar al título de:
Ingeniería de Telecomunicaciones

Asesor
Natalia Gaviria Gómez
Profesor Facultad de Ingeniería

Universidad de Antioquia
Facultad de Ingeniería
Pregrado
Medellín, Antioquia, Colombia
2022

Tabla de contenido

Índice de figuras

Índice de tablas

Resumen

La confiabilidad, rapidez y seguridad de las máquinas y juegos de azar es un tema importante en la industria, pero estas también deben cumplir con toda la reglamentación de los entes reguladores; en el presente trabajo se realizó una búsqueda y exploración de la máquina de juegos Silver Route para obtener información de relevancia sobre el funcionamiento de esta, explorando tanto la infraestructura física y lógica que la componen.

Una vez conocida en detalle la máquina de juegos a tratar, se realizó un estudio del protocolo SAS (Slot Accounting System), que es un protocolo de facto para automatizar los informes de medidores y el registro de eventos para este tipo de máquinas de juegos de azar; pasando así a diseñar y realizar un módulo de comunicación entre la máquina de juegos y el host, por medio del protocolo SAS, permitiendo así el control y obtención de los contadores necesarios para la generación de información de control requerida por los entes reguladores.

Con esto se concluyó que aunque el protocolo SAS es una gran herramienta que permite de buena manera la automatización de informes de medidores y el registro de eventos, es un protocolo que fue diseñado hace muchos años, por lo que han aparecido otros protocolos con mejores características que permiten la comunicación directa entre la máquina de juegos y el host sin necesidad de una tarjeta SMIB.

1. Introducción

Las apuestas son uno de los pasatiempos más antiguos del mundo [2] y actualmente la industria de los juegos de azar ha presentado un crecimiento significativo en Colombia, además de la regularización y reglamentación realizada por Coljuegos, que es el ente regulador del sector.

Teniendo en cuenta que los recursos recaudados van para los fondos del estado, para financiamiento de la salud [3], se ha reforzado la lucha contra la ilegalidad, permitiendo así el desarrollo de nuevas técnicas que promuevan una industria potencialmente competitiva y segura, tanto en máquinas físicas o a través de internet por plataformas virtuales.

Si miramos hacia atrás, se puede observar claramente la evolución que ha tenido la tecnología con el pasar de los años, y este avance de tecnología permea también la industria de los juegos y las apuestas. Ante esta tendencia, surgió la necesidad de aplicar nuevas herramientas que agilicen los procesos, permitiendo suplir las demandas de los usuarios y de los clientes.

La empresa IES (Intelligent Electronic Solutions) es una compañía colombiana de tecnología, especializada en el desarrollo de aplicaciones web y móviles, con énfasis en soluciones de alto impacto para la gestión, generación de la información y el conocimiento. Uno de los grandes proyectos con los que cuentan es el proyecto SLOT, que se basa en el desarrollo y la fabricación tanto de hardware como de software de máquinas de juegos para casinos, entre ellos el modelo Silver Route, que es un proyecto que está en marcha y se trata de una máquina de juegos de azar que cuenta con 6 diferentes juegos: póker, blackjack, sic bo, bailarina, rieles y ruleta. Esta máquina está pensada para ser usada en cualquier tipo de negocio o tienda, no solo en los casinos, por lo que se busca que pueda ser de bajo costo, pero además debe cumplir con toda la normativa que la rige.

Para la reglamentación de la máquina, el ente regulador es Coljuegos y uno de los requerimientos es el uso de dispositivos electrónicos o componentes de software que tengan la funcionalidad necesaria para la lectura de los contadores y generación de información de control requerida [13], por lo que esta máquina hace uso de la SMIB (Slot Machine Interface Board), ver **Fig. 1**, que es un dispositivo electrónico que hace de interfaz entre el servidor host y la máquina Silver Route.

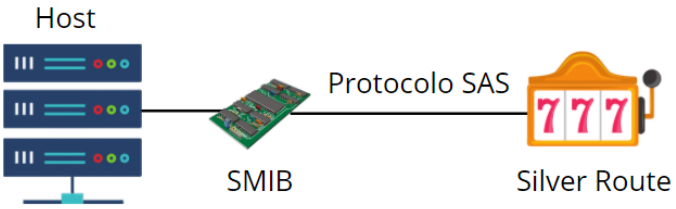

Figura 1. Arquitectura del sistema del protocolo SAS, protocolo usado para la comunicación entre la máquina de juegos Silver Route y SMIB

Para lograr esta comunicación es necesario que la máquina Silver Route se comunique por medio del protocolo SAS (Slot Accounting System) con la SMIB, por lo cual se ve en la necesidad de crear este módulo de comunicación bajo este estándar.

2. Objetivos

2.1. Objetivo general

Desarrollar e integrar el módulo de comunicación que permita la comunicación entre la máquina de juegos Silver Route y el servidor por medio de la interfaz SMIB con base en el estándar de comunicación SAS.

2.2. Objetivos específicos

- Definir los requerimientos funcionales del módulo de comunicación para que este cumpla con los requerimientos de aceptación establecidos para la máquina de juegos Silver Route.
- Diseñar los diferentes diagramas para la realización del módulo que cumpla con los requerimientos funcionales.
- Implementar e integrar el módulo de comunicación a la máquina de juegos Silver Route.
- Verificar el correcto funcionamiento del módulo referente a la lectura de los contadores y la generación de información de control requerida en un escenario real.

3. Marco teórico

3.1. Máquina de juegos Silver Route

La máquina de juegos Silver Route es una máquina de apuestas que cuenta con múltiples juegos, en la cual el apostador podrá seleccionar el juego que más le interese y cambiar entre los diferentes juegos, si así lo desea. Silver Route cuenta actualmente con 6 diferentes juegos: Sicbo, ruleta, rieles, póker, black jack y bailarina. Algunos de estos juegos tienen pequeñas variaciones, como es el caso de rieles y póker.

3.1.1. Infraestructura física

La arquitectura física de Silver Route se ilustra en la **Fig 3.** Lo primero a destacar es la CPU, al cual se conectan todos los componentes, sensores, y controlar toda la lógica de los juegos y el comportamiento de uso. Cuenta con un procesador Intel Core i5-7200U, una tarjeta gráfica integrada y una memoria RAM de 8G y cuenta con los conectores ilustrados en la **Fig 2**.

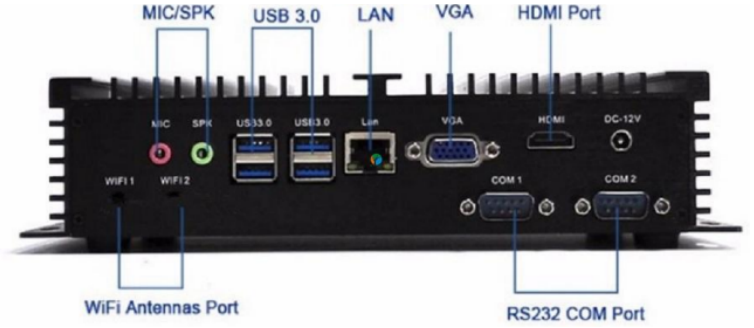

Figura 2. Puertos del miniPC.

Cuenta con un pulsador como sensor de puerta abierta, 2 pulsadores de llave: uno permite acceder a la configuración de la máquina, el otro para realizar el handpay o refresca la máquina en caso de fallos y 10 botones que permiten la opción de controlar los juegos, todos estos sensores están conectados a un joystick.

La máquina de juegos Silver Route cuenta también con un billetero donde los usuarios depositan el dinero a apostar en los diferentes juegos. Este billetero se conecta directamente por puerto serial al miniPC. Este billetero cuenta con un stacker en el cuál se irá guardando todo el dinero ingresado, que el operador puede extraer al abrir la puerta de la máquina con su respectiva llave.

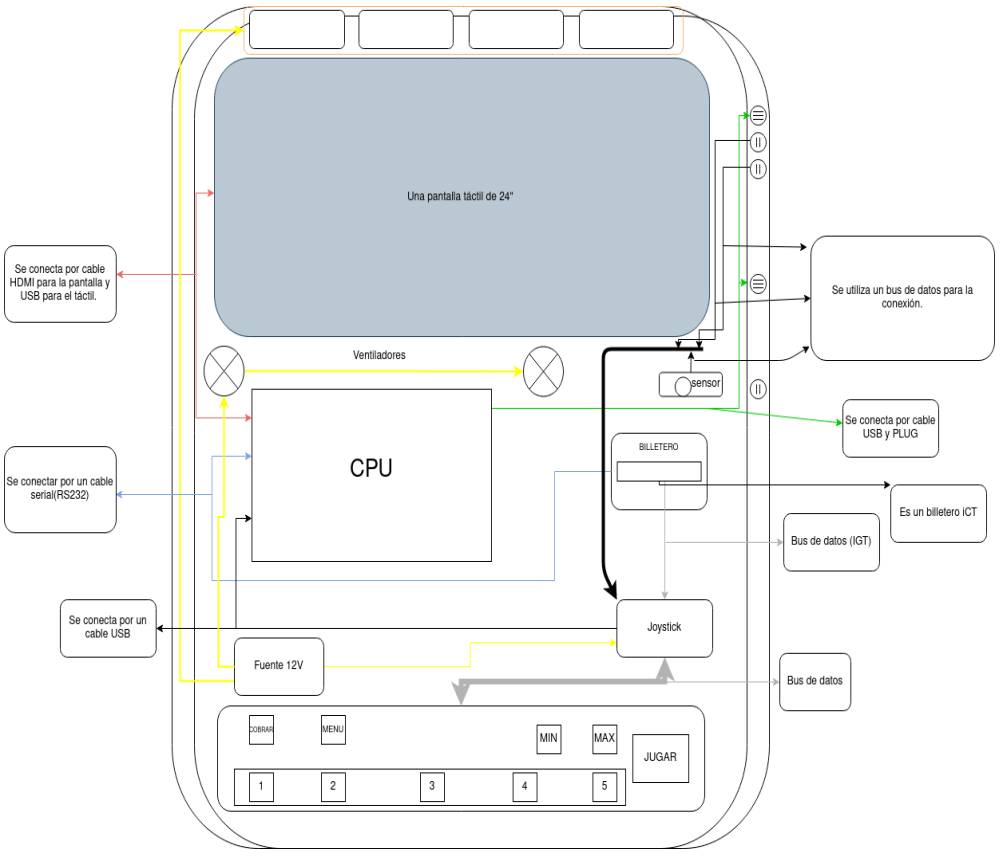

Figura 3. Arquitectura física de la máquina de juegos Silver Route

3.1.2. Infraestructura lógica modular

La infraestructura lógica de la máquina de juegos Silver Route se muestra en la **Fig 4.** Está conformada por 7 módulos principales, donde cada uno de estos cumple una función específica y se comunica con los diferentes módulos para lograr el funcionamiento general de los juegos, las apuestas realizadas, los bloqueos la verificación de activación de los diferentes botones o pulsadores, entre otros.

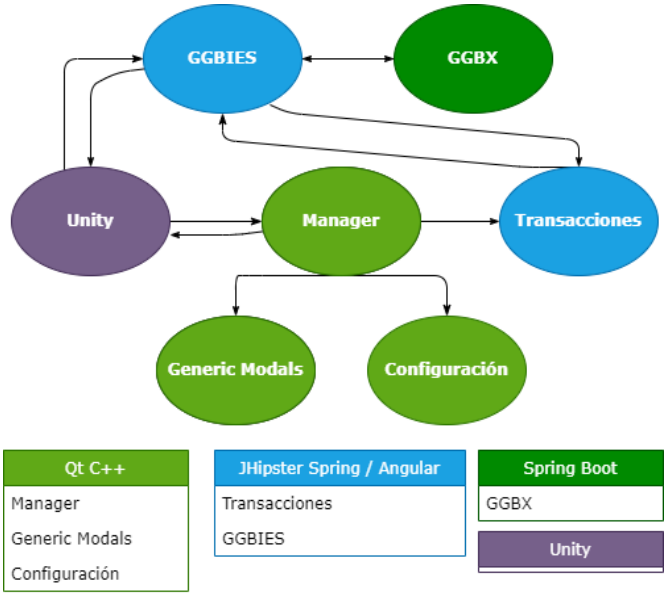

Figura 4. Infraestructura lógica modular de la máquina de juegos Silver Route con sus lenguajes de programación y tecnologías de cada módulo

GGBX es el módulo de juegos. Hay un módulo GGB"X" por cada juego, la X hace referencia al nombre de cada juego (Ejemplo GGBPóker), allí se implementan la lógica y las reglas que definen el juego, estos están desarrollados en Java y utilizan el framework Spring.

El módulo GGBIES (Game Generic Backend en IES). Es el gestor de sesiones de los juegos, cada vez que se apuesta se genera una sesión. Cada juego maneja un flujo de peticiones desde que comienza la sesión hasta que termina. Además se quiere una comunicación con el módulo de Transacciones, el cual es una API donde se realizan todas las transacciones referentes a la contabilidad de los juegos y de la máquina en general, tanto GGBIES como Transacciones están realizados por el generador de código JHipster.

Para el diseño, creación y el funcionamiento de los juegos y servicios se usa el motor de juego Unity que cuenta con grandes ventajas en los entornos de calidad, aunque existen diferentes opciones como Godot, Unreal, Qt e incluso HTML. El motor de juegos seleccionado fue Unity, ya que cuenta con una gran documentación, es soportada por más de 25 plataformas (desde dispositivos móviles, plataformas de realidad virtual y plataformas de escritorio), además cuenta con una baja curva de aprendizaje y permite diseños en 3D que llaman la atención a los usuarios [14].

Los diferentes módulos que se conectan directamente con los periféricos y manipulación y control de toda la parte electrónica se desarrollan en C++, ya que es un lenguaje veloz porque puede hacer un muy buen manejo de la memoria sin máquinas virtuales ni recolectores de basura (a diferencia de otros lenguajes de programación como Java o C#). Además, C++ es un lenguaje complejo, tiene una gran cantidad de librerías e implementaciones y es constantemente actualizado [7].

3.2. Qt

Es un framework multiplataforma y un marco de interfaz gráfica de usuario (GUI) [15], un kit de herramientas que se utiliza para desarrollar software que se puede ejecutar en diferentes plataformas de hardware y sistemas operativos. Qt facilita el desarrollo de software con GUI de aspecto nativo (para el sistema operativo en el que se ejecuta) utilizando C++, por lo que también se clasifica como un kit de herramientas de widgets, utilizando el procesador MOC (Meta-Object Compiler) para ampliar el lenguaje C++ con las funciones de las *signals* y *slots*.

Escribir en C++ da la posibilidad de trabajar con gran variedad de librerías que se van actualizando constantemente por la comunidad y su código se compila en binarios nativos que se ejecutarán en las diferentes plataformas a toda velocidad sin la necesidad de una máquina virtual, permitiendo portar una aplicación a múltiples plataformas a través de una simple compilación.

Qt es de código abierto y está desarrollado por Qt Group (anteriormente Trolltech) en Nokia, por lo que tiene una empresa muy grande que lo mantiene con el apoyo de la comunidad y asegurando su evolución y crecimiento.

3.2.1. *Signals* y *slots*

Son utilizadas para la comunicación entre objetos [16]. El mecanismo de señales y ranuras es una característica central de Qt y estas son posibles gracias al sistema de metaobjetos de Qt.

Una *Signal* es emitida por un objeto cuando ocurre un evento en particular o su estado interno ha cambiado de alguna manera. Las *signals* son funciones de acceso público, por lo que pueden ser emitidas desde cualquier lugar, y un *slot* es una función que se llama en respuesta a una señal particular. permitiendo pasar parámetros o información de un objeto a otro o simplemente activar un *slot* para cumplir una función específica.

Un *slot* es llamado cuando se emite una señal conectada a ella. Los *slots* son funciones normales de C++ y se pueden llamar normalmente; su única característica especial es que se les pueden conectar señales.

Dado que los *slots* son funciones miembro normales, siguen las reglas normales de C++ cuando se les llama directamente. Sin embargo, como *slots*, pueden ser invocadas por cualquier componente, independientemente de su nivel de acceso, a través de una conexión *signal-slot*. Esto significa que una *signal* emitida desde una instancia de una clase arbitraria puede hacer que se invoque un *slot* privado en una instancia de una clase no relacionada.

Cuando se emite una *signal*, los *slots* conectados a ella generalmente se ejecutan de inmediato, como una llamada de función normal. Cuando esto sucede, el mecanismo de *signals* y *slots* es totalmente independiente de cualquier bucle de eventos. La ejecución del código que sigue a la declaración de emisión ocurrirá una vez que todas los *slots* hayan regresado.

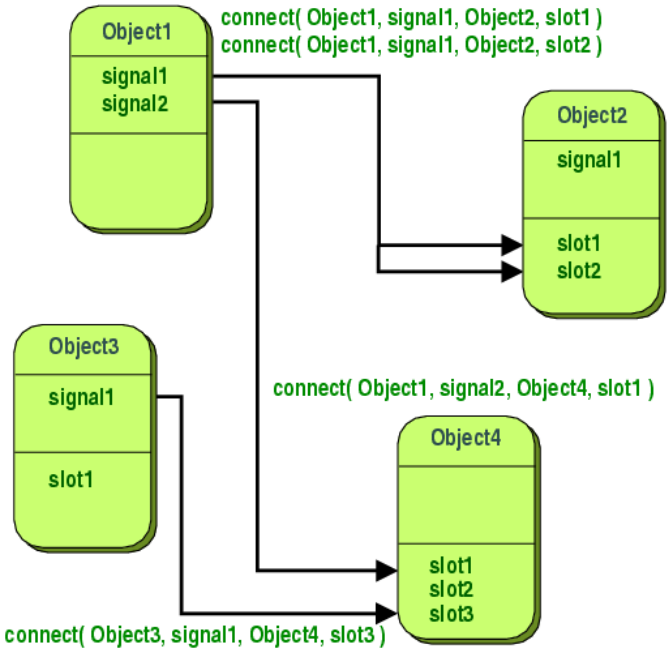

Figura 5. Conexión de *signals* y *slots* entre diferentes objetos. [16]

3.2.2. QByteArray

En una clase proporcionada por Qt la cual se puede utilizar para almacenar y procesar bytes, permitiendo facilitar la comunicación en serie, pudiendo manipular de manera fácil los datos y realizar diferentes conversiones, como la de hexadecimal [18].
Por ejemplo, si se desea convertir un código hexadecimal a char:

```
QByteArray text = QByteArray::fromHex("517420697320677265617421");
text.data();     // retorna "Qt is great!"
```

3.2.3. QTcpSocket

TCP (Transmission Control Protocol) es un protocolo de transporte confiable, orientado a la transmisión y orientado a la conexión [19]. Es especialmente adecuado para la transmisión continua de datos.

QTcpSocket es una subclase de QAbstractSocket que permite establecer una conexión TCP y transferir flujos de datos

3.2.4. QSerialPort

Es una clase que permite acceder, configurar y manipular los puertos seriales disponibles del sistema, permitiendo abrirlos en modo de sólo lectura, sólo escritura, o lectura y escritura, permitiendo el total control de la comunicación por el puerto que se desee [17].

3.3. Slot Accounting System (SAS) [10]

Desde que la empresa IGT presentó **SAS** a principios de la década de 1990, ha sido el estándar de facto diseñado para automatizar los informes de medidores de máquinas tragamonedas y el registro de eventos, el seguimiento de jugadores, las bonificaciones, la emisión de boletos y los juegos sin efectivo. Además, este ha ido evolucionando a lo largo del

tiempo bajo las demandas de la industria, sacando diferentes versiones, donde cada versión nueva es compatible con las anteriores.

Para implementar la comunicación **SAS** entre una máquina de juego y un **host** es necesario conectar cada máquina de juego a una placa de interfaz inteligente (SMIB). El **SMIB** realiza *polls* a la máquina de juego a la que está conectada y pasa la información de esta máquina de juego al host.

Existen dos formas principales de *polls,* que el host puede usar para interrogar a la máquina de juego, son los de tipo *long* y los de tipo *general*. Los *general polls* se utilizan para hacer *request* de excepciones de eventos de la máquina de juegos mientras que los *long polls* se utilizan para hacer *request* a una información específica de la máquina de juego y también para configurarla.

Hay varios tipos *long polls* disponibles para la comunicación entre el host y las máquinas de juego. Las encuestas largas de tipo R se utilizan para obtener información básica de la máquina de juego. Los sondeos largos de tipo S se utilizan para enviar información a la máquina de juego y para configurar la máquina de juego. Las encuestas largas de tipo M se utilizan para configurar un juego específico u obtener información de un juego específico de una máquina de juegos de varios juegos.

3.3.1. Cyclic Redundancy Check (CRC)

Para la verificación de errores se utiliza la verificación de redundancia cíclica (CRC) que es un código de detección de errores que se usa comúnmente en redes digitales y dispositivos de almacenamiento para detectar cambios accidentales en los datos sin procesar [20]. A los bloques de datos que ingresan a estos sistemas se les asigna un breve valor de verificación, basado en el resto de una división polinomial de sus contenidos. En la recuperación, el cálculo se repite y, en caso de que los valores de verificación no coincidan, se pueden tomar medidas correctivas contra la corrupción de datos, pero estos no se pueden utilizar para la corrección de errores.

Los CRC se llaman así porque el valor de verificación (verificación de datos) es una redundancia (amplía el mensaje sin agregar información) y el algoritmo se basa en códigos cíclicos. Los CRC son populares porque son fáciles de implementar en hardware binario, fáciles de analizar matemáticamente y particularmente buenos para detectar errores comunes causados por el ruido en los canales de transmisión. Debido a que el valor de verificación tiene una longitud fija.

Para el cálculo del CRC el protocolo SAS usa la convención CCITT básica comenzando con el byte más significativo hasta el bit menos significativo y aplicando el polinomio $x^{16} + x^{12} + x^{5} + 1$ a los *polls* y excepciones para la verificación de los datos, agregando 2 bytes al final de los datos.

Por ejemplo, si de la máquina de juegos se envían los datos ABCD en hexadecimal a la SMIB, el CRC calculado sería BE56, enviando así el dato más el CRC, enviando así ABCDBE56, la SMIB recibe este mensaje, separando los datos del CRC, ésta calcula el CRC de los datos y si coincide con el recibido procederá a realizar el procesamiento, si no, simplemente descarta los datos recibidos.

3.3.2. Binary-coded decimal (BCD)

Es un estándar para representar números decimales en el sistema binario, en donde cada dígito decimal es codificado con una secuencia de 4 bits. Con esta codificación especial de los dígitos decimales en el sistema binario, se pueden realizar operaciones aritméticas como suma, resta, multiplicación y división.

Cada dígito decimal tiene una representación binaria codificada con 4 bits:

Decimal:	0	1	2	3	4	5	6	7	8	9
BCD:	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Los números decimales, se codifican en BCD con los bits que representan sus dígitos. Por ejemplo, la codificación en BCD del número decimal 1529 es:

Decimal:	1	5	2	9
BCD:	0001	0101	0010	1001

3.3.3. Direccionamiento de la máquina de juego

La máquina de juego debe admitir una dirección configurable por el operador con un rango de 0 a 127. Esta dirección es de gran importancia ya que al comienzo de cada *requiest* o *response* va la dirección de la máquina de juego y esta dirección sirve para enviar NACK y ACK

3.3.4. *Long polls* no admitidos

Si una máquina de juego recibe un *long poll* que no admite, debe ignorar la encuesta larga y no rechazarla (NACK), por lo tanto, es responsabilidad del host determinar qué *long polls* son compatibles con la máquina de juegos.

3.3.5. SAS Test Program

La empresa IGT cuenta con un programa ejecutable en windows el cual hace el papel de una SMIB, sirve como test para probar los *polls* y *response* a la máquina a la cual se conecta. La interfaz de este programa se ilustra a continuación en la **Fig 6.**

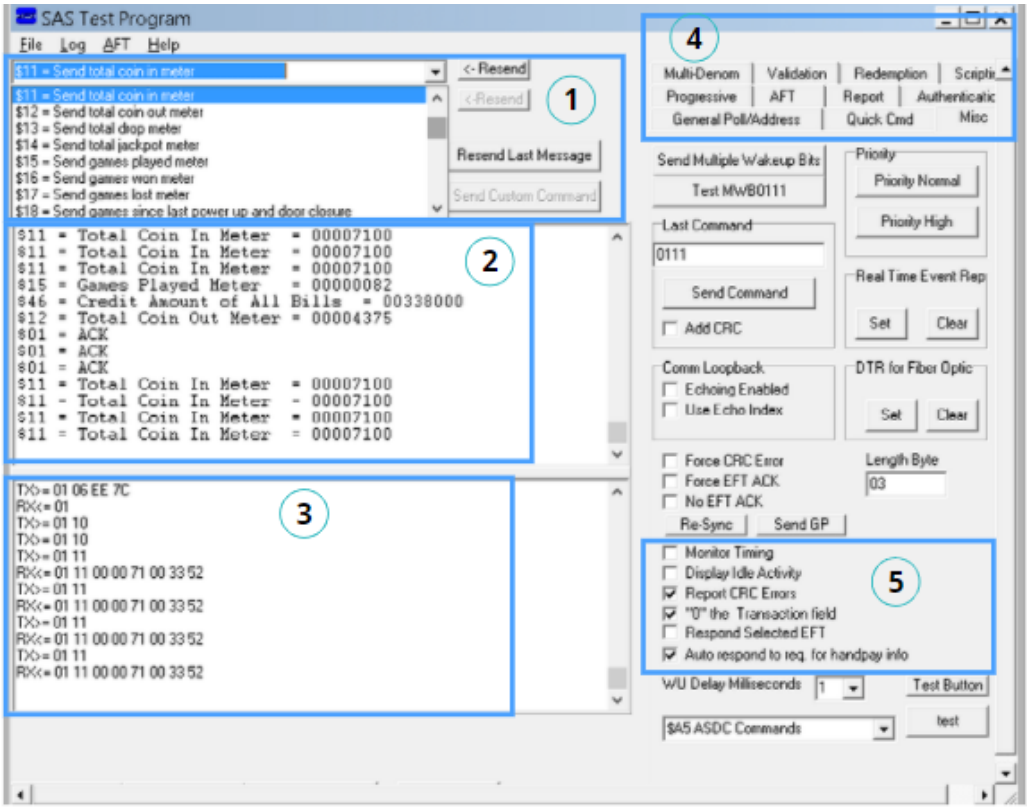

Figura 6. Interfaz gráfica de SAS Test Program.

Del SAS Test Program se destaca el desplegable de *polls* que se pueden enviar a la máquina de juegos mostrados en la parte **1**. En la parte **3** se ven todos los comandos enviados a la máquina de juegos (TX) y los mensajes recibidos de la máquina de juegos (RX). En el apartado **2** se ve la "traducción" de los mensajes que recibe el programa de una manera más amigable para el usuario. En la parte **4** se puede acceder a diferentes características especiales a cambiar en los *polls* recibidos y enviados, como por ejemplo los que se pueden ver en la parte **5**, donde podemos activar el display de errores en el CRC, ver el estado *idle* de la máquina, entre otros.

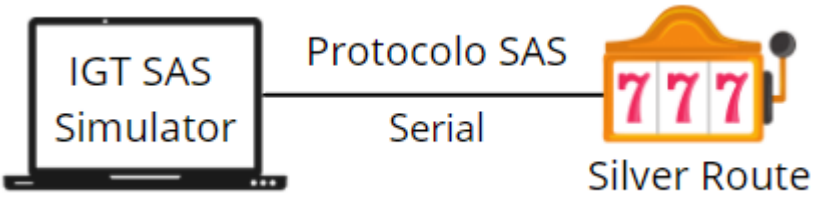

Figura 7. Ilustración del montaje con el simulador del protocolo SAS

Para hacer uso del simulador, éste se ejecuta en un computador y se conecta por el puerto serial con un cable cruzado directamente a la máquina de juegos, como se ve en la **Fig. 7**. En el simulador se podrá probar cada uno de los *polls* con los que cuenta la máquina de juegos.

3.3.6. Slot Machine Interface Board (SMIB)

La **Fig. 1** ilustra la topología para la contabilidad de las máquinas de juego Silver Route, donde se usa la SMIB como intermediario entre el host y la máquina de juegos. La comunicación entre la SMIB y la máquina de juegos se sigue bajo los estándares del protocolo SAS a través de un enlace de datos en serie a 19 KBaud, mientras que la

comunicación entre el host y la SMIB se da mediante una comunicación TCP/IP mediante los certificados SSL (Secure Sockets Layer), que permite cifrar el tráfico de datos.

La SMIB es un dispositivo electrónico que es configurado por el host u operador, definiendo así los *polls* que hará y el tiempo en el que la SMIB realizará dichos *polls* a la máquina de juego a la que está conectada y pasa la información de esa máquina de juego al host, guardando y organizando la información requerida por el host.

4. Desarrollo de la propuesta

En el desarrollo de de este apartado, fue necesaria la intervención, el acompañamiento y el apoyo por las diferentes áreas de la empresa, en las cuales se destacan las siguientes:

- Inicialmente, el equipo de desarrollo, que brinda apoyo con las buenas prácticas de desarrollo de software adoptadas por la empresa, buscando tener mayor calidad en el código, facilitar el proceso de desarrollo, mejorar la legibilidad y la mantenibilidad del código.
- Posteriormente el equipo SLOT, que es el equipo encargado del desarrollo de las máquinas de juegos con los que cuenta la empresa, conociendo así el funcionamiento de estas.
- El equipo I+D, que es el encargado de toda la parte electrónica, conociendo así el funcionamiento de la SMIB.
- Por último el equipo SMOL, que son los encargados de monitorear las máquinas de juegos y enviar los reportes a la entidad reguladora Coljuegos.

Con la información recolectada por el personal encargado y con el equipo necesario, se logró obtener datos del funcionamiento de la máquina de juegos Silver Route, como se indicó en el capítulo anterior.

Desarrollando lo propuesto en el primer objetivo, junto con el con equipo de trabajo, descrito anteriormente, poco a poco se fueron definiendo los requerimientos funcionales para la comunicación entre la SMIB y la máquina de juegos Silver Route, que es crear un nuevo módulo para integrarlo a la infraestructura lógica con la que se cuenta actualmente, como se ve en la **Fig. 4**, este debe ser desarrollado en C++, este debe ser integrado al módulo Mánager que es el encargado de conectar el resto de módulos y el funcionamiento de toda la máquina.

Para garantizar la comunicación entre la SMIB y la máquina de juegos Silver Route, esta máquina debe responder a los *long polls* y enviar los *general polls* de excepciones que fueron discutidos para los requerimientos, todos estos comandos del protocolo SAS se detallan posteriormente en las **tablas 1 y 2**.

El desarrollo del módulo debe hacerse bajo el paradigma de la programación orientada a objetos, con clases bien definidas enfocándose en las características específicas de cada clase y que estas estén encapsuladas para que cada clase pueda proteger la integridad de los datos, aplicando buenas prácticas en el desarrollo de software, aportando así calidad en el código, mejorando la legibilidad, la mantenibilidad y futuros desarrollos que se quieran hacer al módulo.

Con este nuevo módulo lo que se busca también es poder hacer uso de este no sólo en las máquinas de juego Silver Route, sino también en todas las máquinas con las que cuenta la empresa IES y las máquinas a futuro, e incluso con cualquier máquina que requiera comunicarse por medio del protocolo SAS.

4.1. ManagerRoute

El nuevo módulo que logrará la comunicación entre la SMIB y la máquina de juegos Silver Route es ManagerRoute, que está generado con el framework de Qt bajo el lenguaje de C++, buscando que este pueda ser rápido y no interfiera bajo ningún motivo la experiencia de juego de los usuarios.

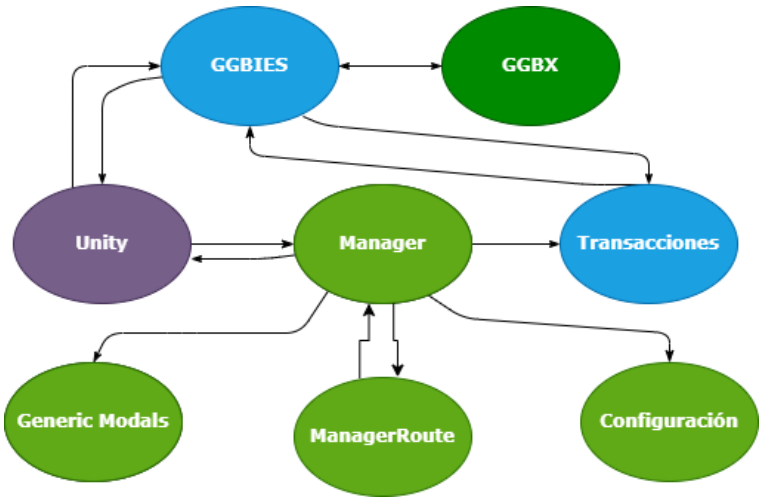

Figura 8. Infraestructura lógica modular con la integración de ManagerRoute

La integración de este nuevo módulo con la infraestructura lógica ilustrada en la **Fig 8**, donde se comunica directamente con la librería Manager por medio de una conexión TCP que se detalla posteriormente.

El módulo ManagerRoute está compuesto principalmente por 5 clases principales, como se ilustra en la **Fig. 9**, que se detallarán a continuación:

- **SerialComm:** Esta clase es la encargada de la comunicación serial directamente con la SMIB, esta clase hace uso de la clase QSerialPort proporcionada por Qt, los métodos y atributos de la clase SerialComm son los siguientes
    - El atributo *_pSerialPort que es un puntero hacia una instancia de la clase QSerialPort
    - El constructor SerialComm() que inicializa el puerto por el que se conectará la máquina y se selecciona la velocidad de transmisión.
    - write(QByteArray) que es el encargado de enviar las tramas hacia la SMIB.
    - El slot readData() que es el que recibirá la trama enviada por la SMIB
    - Por último la señal messageToProcess que permite conectarse con el resto de las clases

- **ServerComm:** El funcionamiento de esta clase es similar a la anterior, la diferencia radica en que la comunicación es por medio de TCP Socket por medio de la librería QTcpServer proporcionada por Qt. ServerComm se comunica directamente con la librería Mánager (o con cualquier máquina que se conecte al puerto TCP), los métodos y atributos de la clase **ServerComm** son los siguientes.
    - El atributo *_TcpServerComm que es un puntero hacia una instancia de la clase QTcpServer
    - Un atributo QList<QTcpSocket*> m_clients para manejar los diferentes clientes que que se puedan conectar a este puerto (El propósito de este es más relacionado a pruebas, pudiéndose conectar por este puerto y simular las respuesta de la máquina de juegos)

- El constructor ServerComm() que inicializa el puerto con el que se conectará a la librería Mánager
- El slot acceptConnection() que permite administrar las conexiones que hayan por este puerto
- sendFrame(QByteArray) que es el slot encargado de enviar las tramas hacia la librería Mánager.
- El slot getDataFrame() que es el que recibirá la trama enviada por la librería Mánager
- Por último la señal analyzeFrame que permite conectarse con el resto de las clases

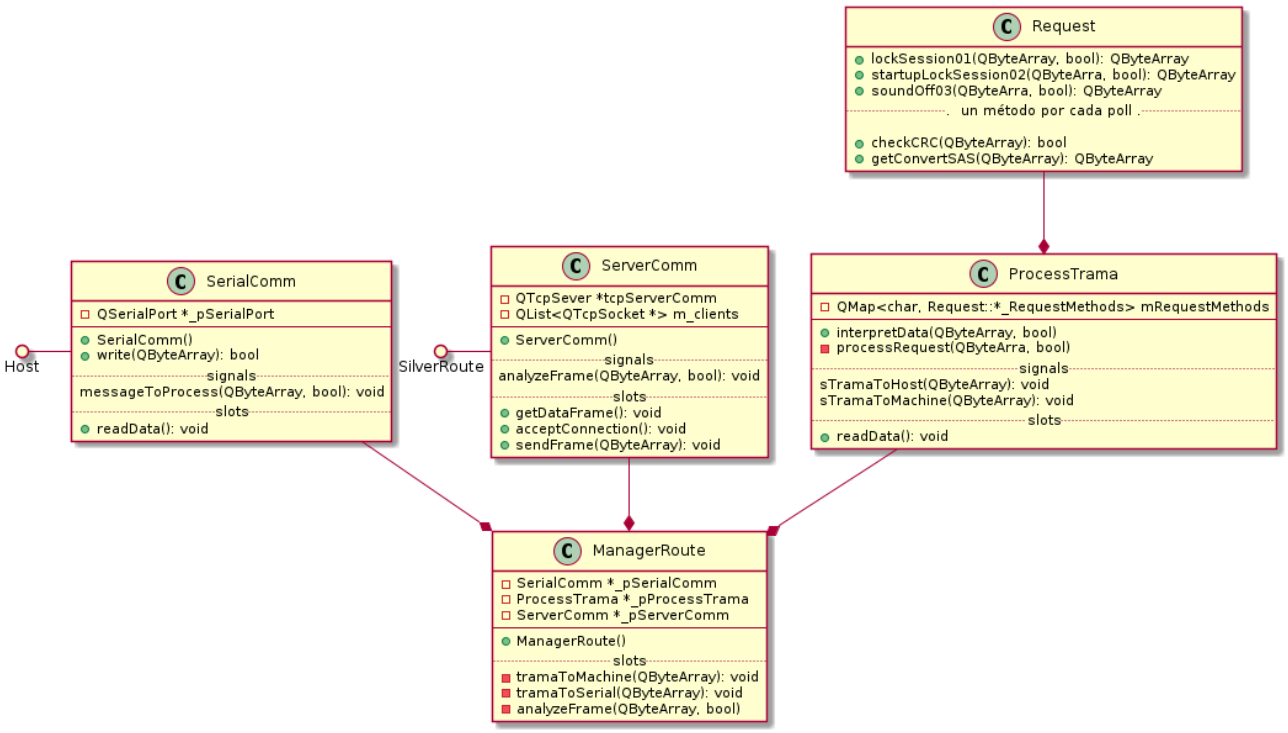

Figura 9. Diagrama de clases del módulo ManagerRoute

- **Request:** Esta clase es la encargada de revisar de que el CRC generado por los *polls* de la SMIB sea el correcto, además de generar el CRC de los *response* de la máquina de juegos, hay un método por cada uno del listado de *polls* del protocolo SAS.
  - El métdodo checkCRC, el cual recibe un QByteArray y retorno true o false si el CRC es correcto
  - getConvertSASSerial que recibe un QByteArray y retorna este mismo QByteArray con el CRC calculado.

- **ProcessTrama:** Esta clase es la encargada de procesar las tramas que llegan, ver de dónde vienen las tramas y hacia dónde irán, generando y verificando los CRC.
  - Cuenta con el atributo mRequestMethods, el cual es un objeto QMap que asocia el código SAS con su respectivo método, por ejemplo, el código '11' que es *poll* del totcal de coin in con el método totalCoinIn11 de la clase Request
  - InterpretData que interpreta las tramas, analizando el tamaño de estas y hacia dónde irían, validando el CRC o agregándolo cuando sea necesario, todo esto con ayuda del método privado processRequest

- Finalmente cuenta con las *signals* sTramaToHost(QByteArray) y sTramaToMachine(QByteArray), para enviar las tramas una vez procesadas según a quien corresponda.

- Finalmente la clase ManagerRoute, el cuál instancia las clases SerialComm, ServerComm y ProcessTrama. Esta clase funciona como un router, realizando las diferentes conexiones de las *signals* y *slots*.
  - Cuenta con los punteros *_pSerialComm, *_pServerComm y *_processTrama para hacer eso de estas clases
  - Cuenta también con los *slots* analyzeFrame, tramaToMachine y tramaToSerial, para ser conectadas con las *signals* de las clases que hace uso, para procesar estas tramas y enviarlas al host (SMIB) o hacia la máquina (Mánager).

Para comprender mejor el módulo ManagerRoute, se muestran las conexiones *signals* y *slots* entre las diferentes clases en la **Fig 10**, donde vemos que ManagerRoute es la que recibe las tramas enviadas por la SMIB y la máquina de juegos en el *slot* analyzeFrame, que para identificar de dónde viene la trama recibe como parámetro un booleano, siendo *true* que va hacia el host y *false* hacia la máquina de juegos

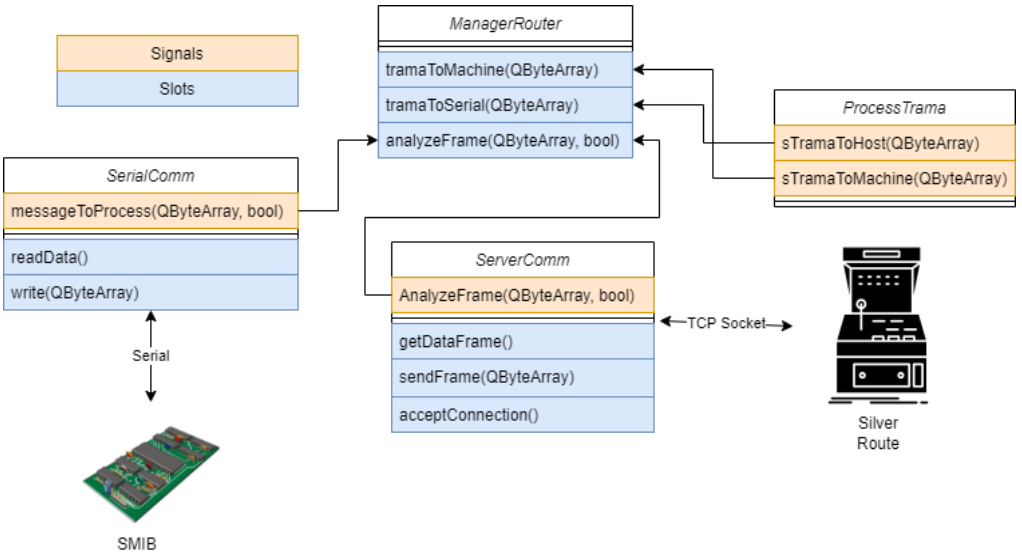

Figura 10. Conecciones entre *signals* y *slots* del módulo ManagerRoute

4.2. Mánager

Una vez implementado el módulo ManagerRoute que permite la comunicación entre la SMIB y la máquina de juegos Silver Route, es necesario que esta última responda a los *polls*, función que ejecuta el módulo Mánager. Este está compuesto por las carpetas mostradas en la **Fig 11**, cada una de estas carpetas contiene diferentes clases que permiten tener control o realizar una acción en específico, por lo que es importante explorarlas.

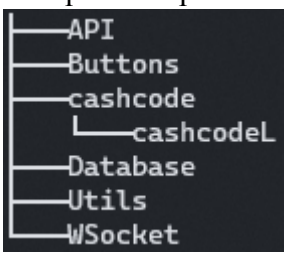

Figura 11. Carpetas que conforman el módulo Mánager

- **CommApi:** Que es la encargada de comunicarse con el módulo de transacciones teniendo así control de los dineros ingresados, validando los contadores y emitiendo las diferentes señales si las cuentas no cuadran, además de reportar los créditos actuales a Unity y el usuario los pueda visualizar.
- **WSocket:** Es la encargada de la comunicación por medio de web sockets, pudiendo comunicar por medio de esta clase los botones presionados a Unity o desplegar los diferentes mensajes de bloqueo de la máquina.
- **Buttons:** Encargada de toda la botonería física de la máquina y los 2 pulsadores de llaves, las *signals* que estos generan son conectados a los *slots* de WSocket.
- **CashCode**: Encargada del billetero, desde acá se configura los billetes y/o tickets que pueden ser ingresados por el billetero, lee el valor recibido y notifica el estado del billetero.
- **DataBase**: Encargado de conectarse directamente con la base de datos que maneja la máquina de juegos Silver Route, pudiendo acceder a las sesiones de juegos hechas, los dineros ingresados, eventos generados, entre otros.
- **Utils:** Contiene diversas clases auxiliares de uso para las otras y genera también las diferentes conexiones de *signals* y *slots* para la interconexión de todas las clases que contienen las carpetas descritas anteriormente.

Estas clases están instanciadas en el main.cpp del módulo Mánager, en este simplemente se instancian todas las clases y se hacen las respectivas conexiones. Con el fin de generar la comunicación con la SMIB y el módulo ManagerRoute, se crea una nueva carpeta dentro de Mánager, esta es ComunicationLibrarySAS.

4.3.*General polls* y *long polls*

Para la máquina de juegos Silver Route no se implementarán todos los *polls*, se harán sólo los requeridos por la empresa IES, cumpliendo con la normativa de Coljuegos para la generación de los reportes diarios.

Para los *general polls* de excepciones se agregaron los mostrados en la **Tabla 1,** donde estos son reportes que la máquina de juegos envía directamente a la SMIB cuando un evento de estos sucede, por ejemplo, si no hay ninguna novedad constantemente el código “00”, apenas se prende la máquina este le informa a la SMIB con el código “17”, si se ingresa un billete de $5000 este envía el código “4F”, seguido de la denominación que sería el código “48” en este caso.

Código	Descripción
00	Estado idle, sin novedad
11	Puerta fue abierta
12	Puerta fue cerrada
17	Máquina de juegos prendida
18	Máquina de juegos apagada

27	Stacker de billetes lleno
28	Billete atascado
29	Fallo en el hardware del billetero
47	Billete de $1000 aceptado por el billetero
48	Billete de $5000 aceptado por el billetero
49	Billete de $10000 aceptado por el billetero
51	Solicitud de retirar dinero
52	Notificar hand pay realizado
2C	Billete falso detectado
3C	Opciones modificadas en la máquina. (Esto se envía cada vez que el operador cambia las opciones de configuración. Esto incluye, entre otros, denominación, cualquier opción que afecte la respuesta a encuestas largas o cambio en los archivos .ini de la máquina) .
4A	Billete de $20000 aceptado por el billetero
4B	Billete de $50000 aceptado por el billetero
4C	Billete de $100000 aceptado por el billetero
4D	Billete de $2000 aceptado por el billetero
4F	Billete aceptado por el billetero
7E	Inicio de una jugada
7F	Jugada terminada

Tabla 1. Lista de *general polls* implementados de la máquina de juegos Silver Route

Para los *long polls* se agregaron los mostrados en la **Tabla 2,** especificando el código del *poll*, el tipo, una descripción y la respuesta que debería ir por parte de la máquina de juegos. Cada *poll* en esviado con el código de la máquina de juegos ‘01’, para cualquier máquina de juegos Silver Route puede tener este mismo código, ya que cada máquina debe contar con su respectiva SMIB; y la respuesta de la máquina debe ser con este mismo código, seguido por el valor solicitado (exceptuando los ACK y NACK) y finalmente el CRC.

Para la máquina de juegos Silver Route solo hay 2 tipos de *long polls* implementados:

1. **Tipo S:** Consta de la dirección de la máquina de juego, código del *long poll*, datos opcionales para la máquina de juego y un mensaje CRC de dos bytes. Cuando la máquina de juegos recibe un *long poll* de tipo S, valida el mensaje CRC y cualquier

dato del mensaje. Si el mensaje es válido, la máquina de juego reconoce este comando y realiza la acción que se está solicitando y responde ACK. Si la máquina de juegos no recibe correctamente este *long poll* o no se puede realizar la acción que se solicita, la máquina de juego responde negativamente (NACK) o ignora el mensaje.

2. **Tipo R:** Consta de la dirección de la máquina de juego, seguido por el código del *long poll*. La respuesta de la máquina consiste en su dirección, código del *long poll*, datos solicitados y un mensaje CRC de dos bytes.

Nota: Si la máquina de juegos envía ACK este envía un sólo byte de la dirección de la máquina, si envía NACK este envía sólo un byte de la dirección de la máquina + 80.

Poll	Tipo	Descripción	Respuesta
01	S	Deshabilitar máquina	ACK o NACK
02	S	Habilitar máquina	ACK o NACK
03	S	Apagar sonidos	ACK o NACK
04	S	Prender sonidos	ACK o NACK
06	S	Habilitar billetero	ACK o NACK
07	S	Deshabilitar billetero	ACK o NACK
08	S	Configurar denominaciones billetero	ACK o NACK
11	R	Total coin in	4-byte BCD
12	R	Total coin out	4-byte BCD
13	R	Total drop	4-byte BCD
14	R	Total jackpot	4-byte BCD
15	R	Juegos ganados	4-byte BCD
1A	R	Créditos actuales	4-byte BCD
1E	R	Total de billetes ingresados	4-byte BCD por cada denominación
20	R	Total valor en pesos de los billetes ingresados	4-byte BCD total de dinero en pesos
31	R	Cantidad de billetes de $1000	4-byte BCD número de billetes
32	R	Cantidad de billetes de $2000	4-byte BCD número de billetes
33	R	Cantidad de billetes de $5000	4-byte BCD número

			de billetes
34	R	Cantidad de billetes de $10000	4-byte BCD número de billetes
35	R	Cantidad de billetes de $20000	4-byte BCD número de billetes
36	R	Cantidad de billetes de $50000	4-byte BCD número de billetes
37	R	Cantidad de billetes de $100000	4-byte BCD número de billetes
7E	R	Fecha y hora actual	4-byte BCD fecha 3-byte BCD hora
1B	R	Información hand pay	
54	R	Versión del SAS e información de la máquina	
6F	R	Contadores extendidos	
72	S	Transferencia de fondos avanzada (AFT)	
73	S	Registro AFT de la máquina	
94	S	Reinicio hand pay	1-byte código de reinicio
AF	R	Contadores extendidos (6F)	

Tabla 2. Lista de *long polls* implementados de la máquina de juegos Silver Route

Para entender mejor el flujo de la comunicación de la SMIB y la máquina de juegos se puede ver la **Fig. 12**, donde se comienza con el *long poll* de total coin in (código 11), por lo tanto envía 0111, que es la dirección más el comando, la máquina le responde con la dirección, el comando, la información, que en este caso es de 4 bytes en BCD (00015200), representando que la máquina en total le han ingresado $15200 en apuestas, seguido por los 2 bytes del CRC. Luego de esto se supone que algún usuario realiza alguna apuesta en un juego, por lo cual la máquina envía la excepción 7E que indica esto, una vez finalizada la partida la máquina envía la excepción 7F. Ahora la máquina envía un *long poll* tipo S para deshabilitar la máquina, por lo cúal esta envía el código de la máquina, más el *poll* y por ser tipo S 2 bytes del CRC, enviando finalmente 01010851, la máquina de juegos recibe este y si todo está en orden deshabilita la máquina y responde con un ACK (01).

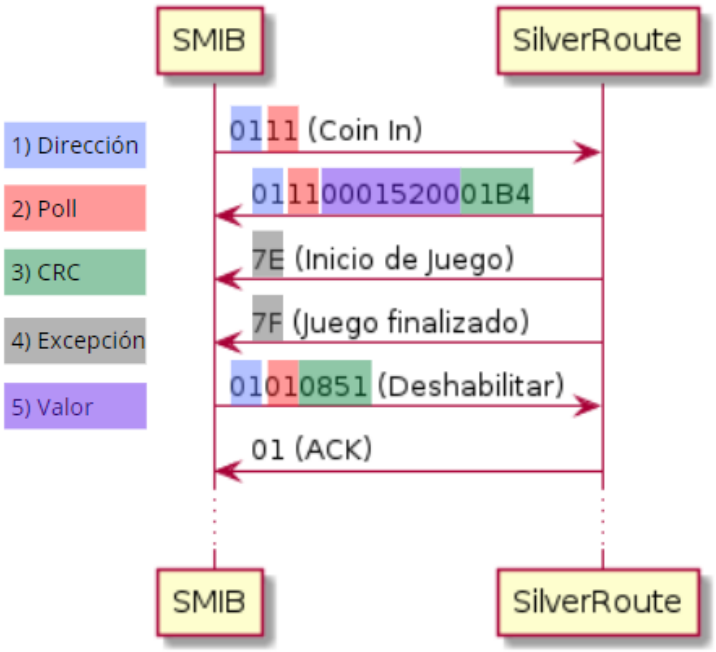

Figura 12. Flujo representativo de comunicación entre la máquina de juegos Silver Route y la SMIB

4.4. ComunicationLibrarySAS

Para responder a los *polls* y mandar las excepciones hacia la SMIB se creó la carpeta ComunicationLibrarySAS dentro del módulo Mánager, Esta carpeta está compuesta por 3 clases mostradas en la **Fig. 13,** que se detallarán a continuación.

- **ComunicationTcpSocket:** Esta clase se encarga de conectarse con la clase **ServerComm** descrita anteriormente en el módulo **ManagerRoute**.
    - Cuenta con el *slot* sendFrame(QByteArray) para enviar los datos
    - El *slot* succesConnection() para establecer la conexión
    - El *slot* getDataFrame() para recibir los datos
    - Por último la *signal* analyzeFrame(QByteArray) para enviar la información recibida a otra clase
- **Response:** Encargada de armar las tramas a responder según el *poll* recibido, por lo cuál cuenta con un método por cada uno de los *long polls* implementados. Debido que hay *long polls* que realizan cambios en la máquina de juegos, se implementan las *signals* para ser emitidas y se realicen dichos cambios solicitados.

- **ComunicationLibrarySAS:** Clase principal que instancia las 2 clases mencionadas anteriormente, además es la encargada de enviar los *long polls* de excepciones, por lo que estos *slots* están conectados a señales que emiten las otras clases de **Mánager** para enviar los eventos que suceden en la máquina a la SMIB

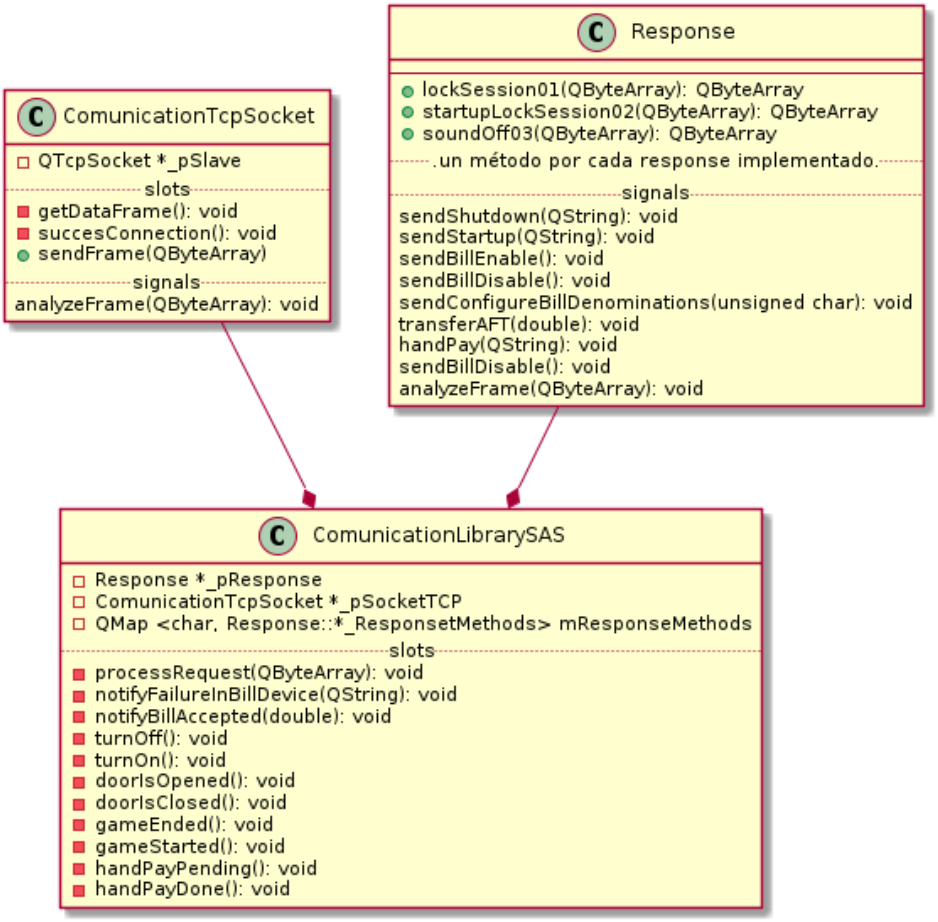

Figura 13. Diagrama de clases del para los response y excepciones de la máquina de juegos Silver Route

Para comprender mejor el funcionamiento de estas clases se muestran las conecciones *signals* y *slots* entre estas en la **Fig. 14**, donde vemos que para el funcionamiento de las excepciones y las respuestas de los *polls* de Silver Route es necesario hacer uso del main del módulo Mánager, para obtener los valores solicitados, monitorear y reportar constantemente los eventos de la máquina y emitir las *signals* correspondientes si hay que realizar algún cambio en el funcionamiento de la máquina de juegos.

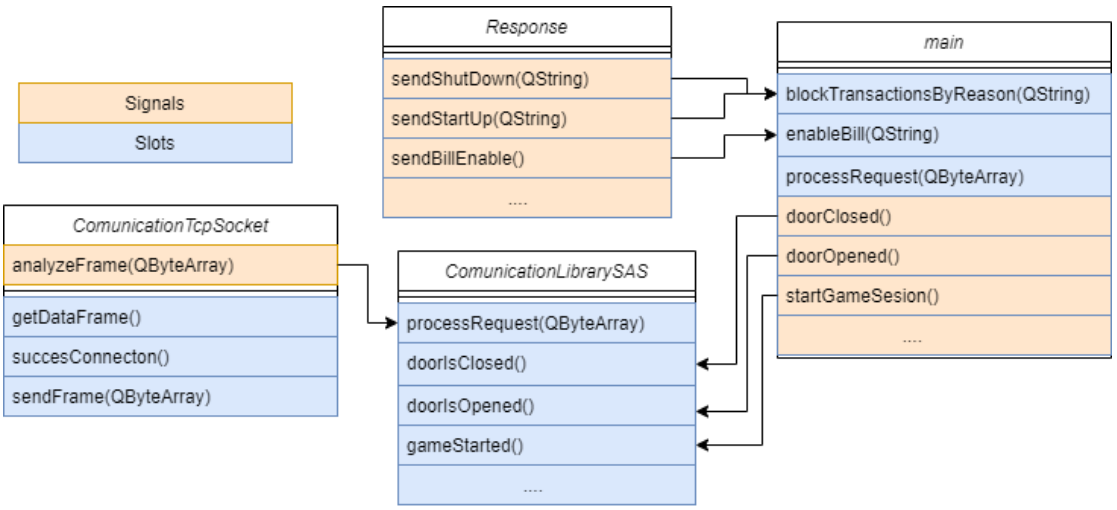

Figura 14. Conecciones entre *signals* y *slots* de las clases implementadas con el main de Mánager

4.5. Mánager Route y Mánager

Como se ha ido mencionando a lo largo del informe, Mánager Route es un módulo que comunica la tarjeta SMIB con cualquier otra máquina de juegos cuya comunicación está basada en el protocolo SAS; y para que la máquina de juegos Silver Route pueda responder a los *long polls* de la SMIB se implementó la carpeta ComunicationLibrarySAS dentro del módulo de manager, para poder enviar la información solicitada, aplicar la configuración enviada y enviar las excepciones que ocurren en la máquina, una vez implementadas estas se unen como se ilustra en la **Fig. 15**, donde la clase SerialComm se comunica directamente con la SMIB y para la comunicación de Manager Route con la máquina de juegos Silver Route se hace por medio de la librería ServerComm y ComunicationTcpSocket, por medio de TCP

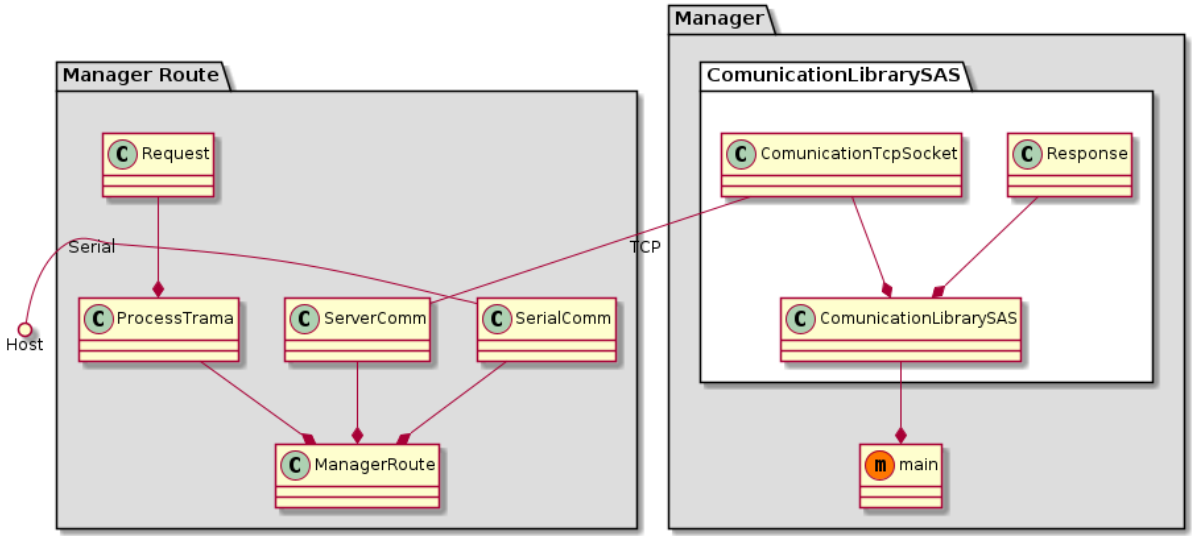

Figura 15. Integración del módulo Mánager Route con Comunication Library SAS

Finalmente, con todo integrado podemos ver el flujo de los datos por todas las clases implementadas, en la **Fig. 16** se ilustra el flujo partiendo desde el *poll* 11 (coin in) generado por la SMIB. SerialComm lee el dato y lo envía a ManagerRoute, con un valor booleano en false indicando que va hacia la máquina, este es enviado a ProcessTrama el cuál mira que todo esté en orden, si es así, lo envía de vuelta a ManagerRoute para que este lo envíe a la máquina, pasando primero por ServerComm, este se comunica con la clase ComunicationTcpSocket del módulo de Mánager, enviando este mensaje a la clase ComunicationLibrarySAS, éste interpreta el código recibido y responde según los datos pedidos. En el *response* este vuelve a pasar por cada una de las clases mencionadas anteriormente, pero cabe destacar el valor booleano agregado por serverComm indicando que este mensaje va hacia el host (SMIB) y la clase ProcessTrama que se encarga de calcular y agregar el CRC.

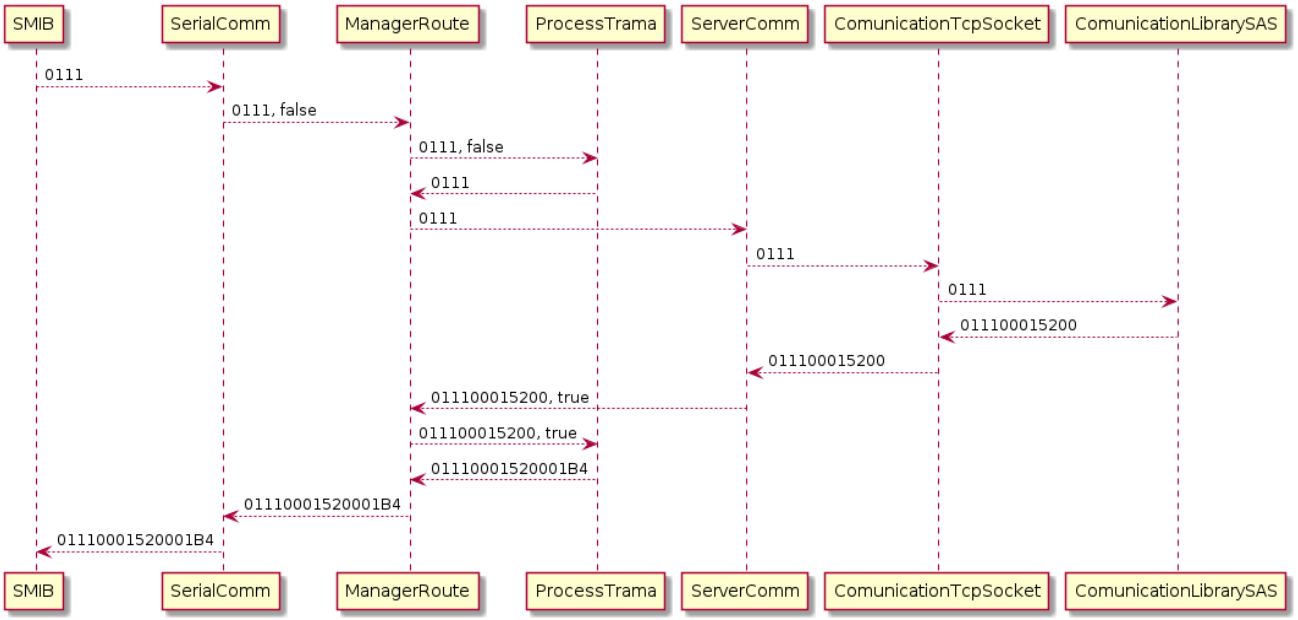

Figura 16. Flujo de datos por las clases implementadas con *long poll* tipo R

En la **Fig. 17** podemos observar el flujo de un *long poll* de excepciones, donde la máquina de juegos Silver Route reporta que ha iniciado un juego o partida, enviando el comando 7E la clase ComunicationLibrarySAS a ComunicationTcpSocket, este lo comunica con la clase ServerComm, que se lo envía a ManagerRoute con el booleano en true indicando que va hacia el host, este lo envía a processTrama para verificar que el comando esté en orden para envíarselo a ManagerRoute y este se lo envíe a la clase SerialComm para ser enviada finalmente a la SMIB.

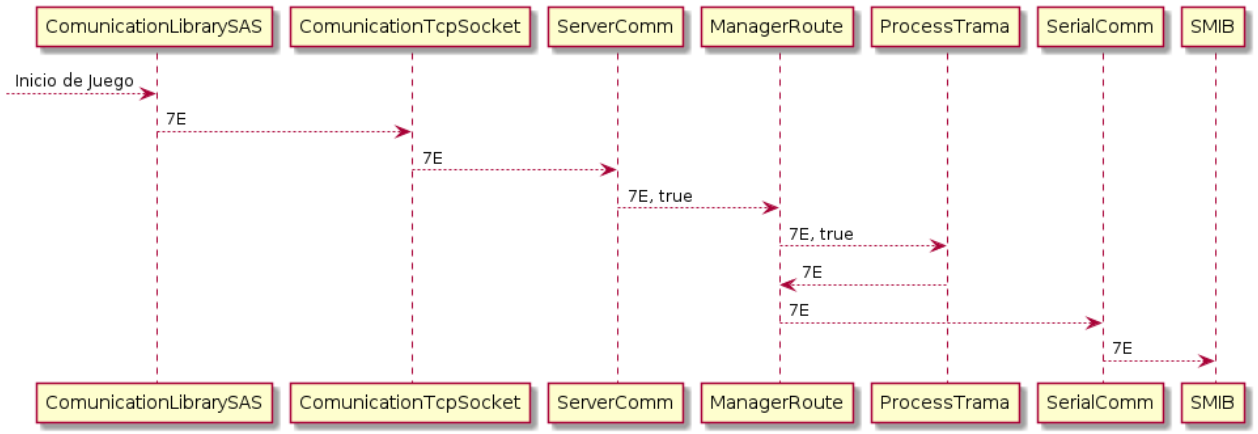

Figura 17. Flujo de datos por las clases implementadas con *long poll request* de excepción

Finalmente, otro de los posibles flujos que puede haber es el de un *long poll* de tipo S, ilustrado en la **Fig. 18**, donde desde el host quieren deshabilitar o bloquear la máquina, por lo que esta envía el código 01 (lockSession) con la dirección de la máquina y su respectivo CRC, el flujo es muy similar al primer flujo detallado, pero acá es importante destacar la clase ProcessTrama la cuál rectifica que el CRC sea correcto para seguir con el flujo normal y la respuesta de la máquina (comunicationLibrarySAS) es un ACK.

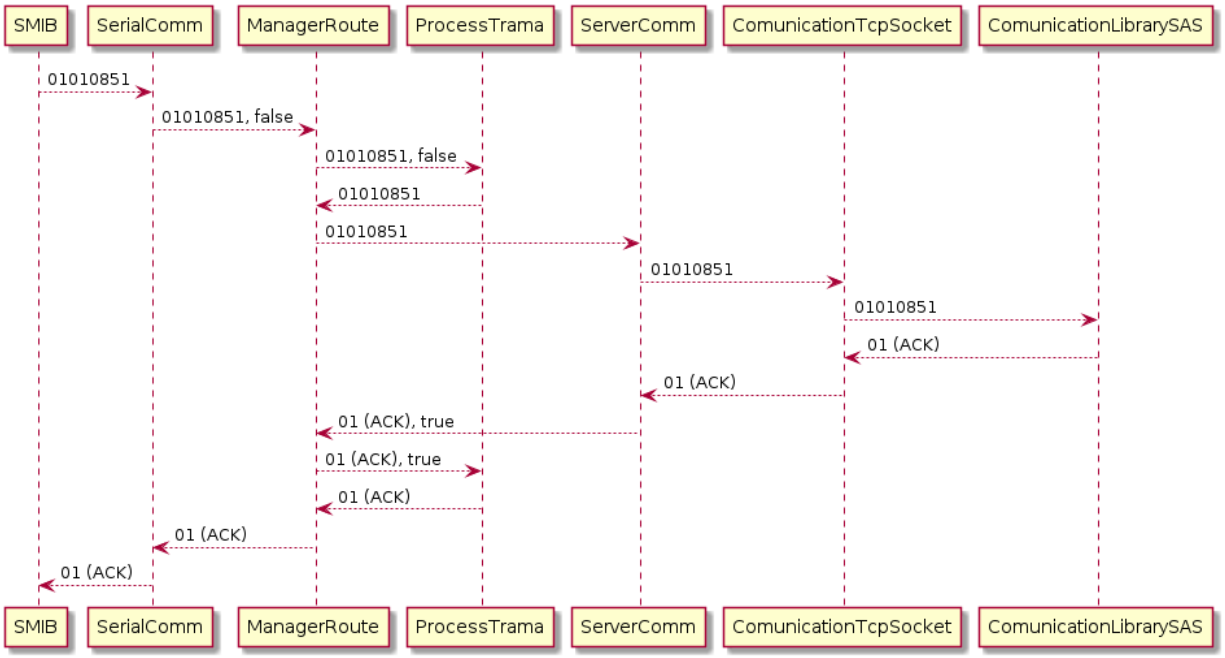

Figura 18. Flujo de datos por las clases implementadas con *long poll* tipo S

5. Resultados

Una vez integrados tanto el módulo ManagerRoute y la carpeta ComunicationLibrarySAS en la máquina de juegos Silver Route y realizando las conexiones de la tarjeta SMIB con SAS Test se como se ilustró en la **Fig. 19**, se empezó a usar la máquina de juegos como si fuese un escenario real, viendo las excepciones que la máquina podía generar y enviando algunos *long poll* desde SAS Test y corroborar que los contadores estén en orden, viendo que la máquina de juegos responda únicamente a los *polls* implementados.

```
TX>= 01 15                    Ô÷†
RX<= 01 15 00 00 00 82 25 31  $01 = ACK
TX>= 01 2D 00 00 FF E0        $7E = Game has started.
TX>= 01 46                    $7F = Game has ended.
RX<= 01 46 00 33 80 00 97 1B  Ô÷†
TX>= 01 2B                    $11 = Total Coin In Meter  = 00007100
TX>= 01 12                    $11 = Total Coin In Meter  = 00007100
RX<= 01 12 00 00 43 75 C7 EE  $11 = Total Coin In Meter  = 00007100
TX>= 01 07 67 6D              $15 = Games Played Meter   = 00000082
RX<= 01                       $46 = Credit Amount of All Bills  = 00338000
TX>= 01 06 EE 7C              $12 = Total Coin Out Meter = 00004375
RX<= 01                       $01 = ACK
                              $01 = ACK
```

Figura 19. Mensajes enviados y recibidos en el programa SAS Test

Una vez corroborados los *polls* con SAS Test se prueba lo desarrollado en un escenario más real, donde la SMIB estará conectada a la red, pudiendo ver la máquina en un servidor de la empresa IES, que es el servidor que usan para generar los informes de contadores de las máquinas y ver el estado de estas.

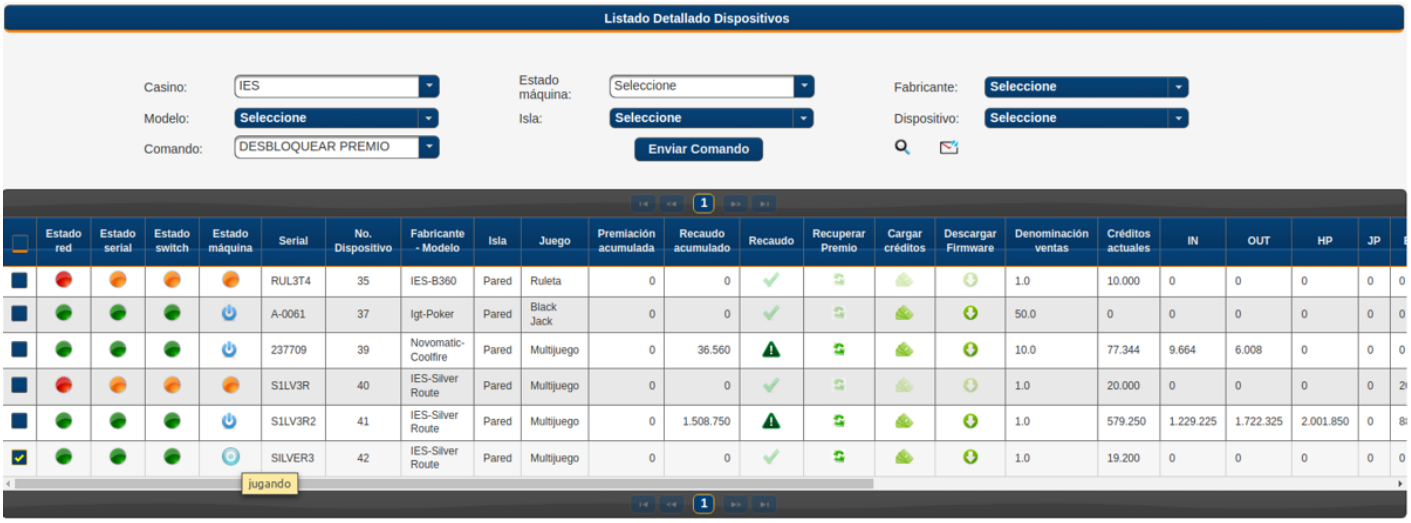

Figura 20. Máquinas de juego conectadas a una SMIB en la red local de IES.

En la **Fig. 20** podemos ver las máquinas de prueba conectadas a la red interna de IES, donde se cuenta con tres máquinas de juegos Silver Route, su estado y sus contadores, pudiendo ver estas máquinas en un escenario de pruebas con el servidor. En este servidor también se pueden enviar ciertos comandos que se han configurado para ser enviados a las máquinas seleccionadas, como se ilustra en la **Fig. 21.**

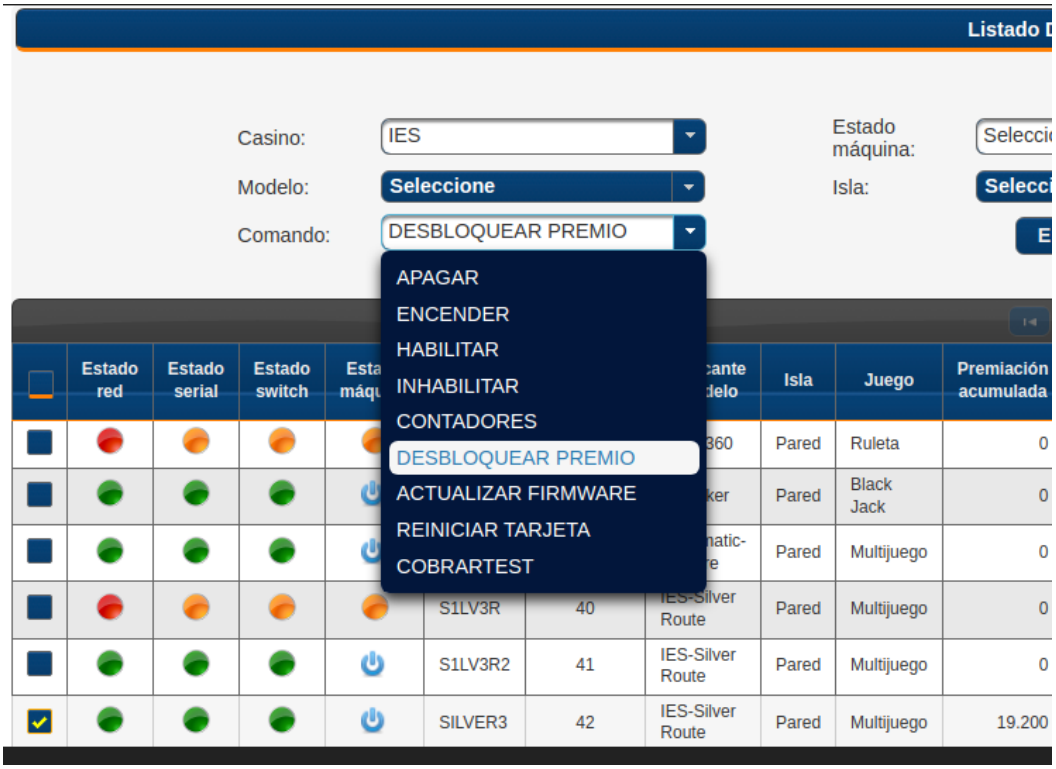

Figura 21. Comandos que están programados en el servidor para enviar a las máquinas por medio del protocolo SAS.

Figura 22. Juego de black jack en la máquina de juegos Silver Route.

En la **Fig. 22** podemos ver un escenario de apuestas en uno de los juegos con los que cuenta la máquina de juegos Silver Route, donde se está jugando black jack y se cuenta con un dinero actual de $19200, que vemos que es el que coincide con la **Fig. 20** mostrada del servidor de IES, donde se indica el estado de jugando y viendo que los créditos actuales coinciden con el mostrado en el juego.

Para ver todo el tráfico de datos se implementó un registro de todos *polls* enviados y recibidos en la máquinas de juegos Silver Route (exceptuando el estado idle), en la **Fig. 23**, podemos ver todo el tráfico generado en una prueba de la máquina de juegos Silver Route desde que es prendida, donde al comienzo todos los contadores están en cero y a medida que se ingresan los billetes y se realizan las apuestas en los diferentes juegos se puede ver cómo se van actualizando estos contadores por los *polls* automatizados que son enviados desde el servidor de IES por medio de la SMIB, donde se puede ver la fecha y hora del comando recibido y hacia dónde está dirigido.

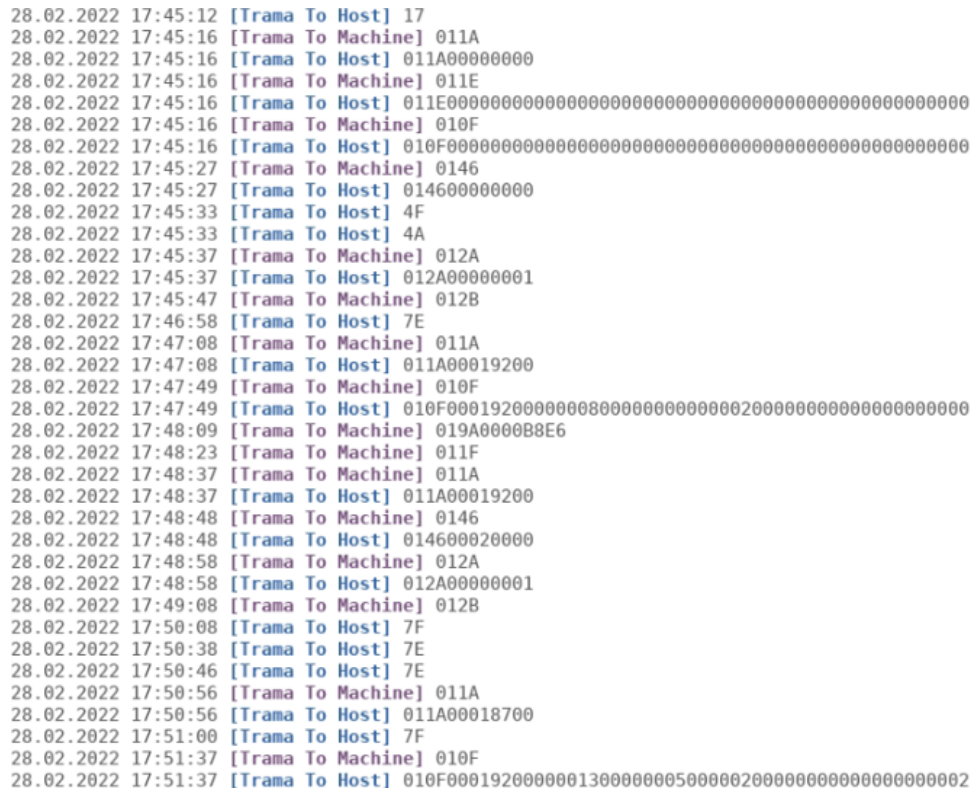

Figura 23. Registro de comandos enviados y recibidos por la máquina de juegos Silver Route en un escenario de pruebas con el servidor.

6. Conclusiones

Este trabajo de grado ha desarrollado un módulo de comunicación entre la máquina de juegos Silver Route y una SMIB, logrando su objetivo principal. Esto se logró gracias a la exploración e investigación del funcionamiento de la máquina de juegos Silver Route y el protocolo SAS. Además, el buen diseño y desarrollo de cada una de las diferentes clases y la comunicación entre estas.

El protocolo SAS aunque es un protocolo desarrollado hace mucho tiempo, sigue siendo un protocolo de facto entre la industria de las máquinas de juego tipo slot, permitiendo que por medio de este protocolo se pueda tener control de la máquina de juegos y obtener los contadores necesarios para la generación de informes para los entes reguladores; pero en todo este tiempo han aparecido nuevos protocolos como el G2S (Game To System) que permiten una comunicación directa entre la máquina de juegos y el servidor sin necesidad de una SMIB, por lo que se considera pertinente ir haciendo la transición a estos otros protocolos.

Si se logra garantizar la comunicación por medio del protocolo SAS con el programa SAS Test proporcionado por la misma empresa de IGT desarrolladora de dicho protocolo, se lograría garantizar esta comunicación con cualquier servidor que también haga uso de este protocolo.

El uso del framework Qt para la programación en C++ proporciona muchas herramientas como el de *signals* y *slots* para la comunicación entre clases y gran cantidad de librerías y documentación para el uso de estas, como lo es la librería QByteArray que permite trabajar

de mejor manera los bytes, y en este caso, con la comunicación serial y el manejo de las bytes, esta librería proporciona una gran ayuda en el desarrollo del módulo de comunicación.

El módulo Mánager Route permite la comunicación directa entre una SMIB y cualquier máquina de juegos que hable el protocolo SAS, por lo que este módulo podría ser usado en las diferentes máquinas que cuenta la empresa IES y en los proyectos futuros.

7. Proyecciones a futuro

Como la máquina de juegos Silver Route cuenta con múltiples juegos, la implementación de *long polls* tipo M sería una buena opción para tener información y control de cada uno de los juegos, esto podría servir para un futuro estudio para saber cuáles son los juegos más seleccionados por los usuarios y poder ver qué tanto dinero se logra recaudar en cada uno, esto podría ser útil para un estudio de los juegos implementados, ya que actualmente con el desarrollo los *polls* implementados y necesarios para la regulación sólo son los *polls* generales.

Viendo que cada máquina de juegos Silver Route debe contar con una SMIB y estos dispositivos cada vez son más escasos y costosos es pertinente pensar en alternativas para no depender de la SMIB, se podría pensar en un desarrollo en software que reemplace esta, incluyéndose en el módulo desarrollado Manager Route, cambiando la comunicación serial por una comunicación TCP/IP con el uso de certificados SSL para lograr una comunicación directa entre el host y la máquina de juegos. Otra posible opción para no depender de la tarjeta SMIB es la implementación de otros protocolos que han ido ganando fuerza en el mercado como lo es el protocolo G2S.

8. Glosario

Contadores: Son los totalizadores o acumuladores de la cantidad de jugadas y valores que genera el elemento de juego a partir de una sesión de juego.

Sesión de juego: Una sesión de juego inicia cuando se realiza la carga al contador de créditos una vez ingresado el dinero en efectivo o desde el tiquete recibido y validado por el elemento de juego y termina cuando el elemento de juego entrega premios en dinero en efectivo y/o devuelve créditos no apostados en dinero en efectivo, cuando emite un tiquete con el valor de los créditos no apostados y/o los ganados y/o cuando se pierden todos créditos.

Total Coin In: Créditos totales apostados en la máquina de juego de todas las fuentes. Se utiliza para medir la facturación total o los salarios brutos en una máquina de juego.

Total Coin Out: Valor total de todos los créditos pagados directamente por la máquina de juego, como resultado de ganar apuestas y premios de un sistema de bonificación externo, ya sea que el pago se realice en el medidor de crédito o desde un dispositivo de retiro (hopper, impresora, transferencia electrónica, etc.) . Se utiliza como componente de la ganancia estimada de una máquina de juego.

Total Jackpot: La suma acumulada de todos los créditos pagados por un asistente, como resultado de ganar apuestas y premios de un sistema de bonificación externo. Esto incluye pagos manuales resultantes de premios mayores progresivos, pagos de bonificación y/o ganancias en juegos, independientemente de si la ganancia es o no uno de los mejores premios mayores. Los créditos agregados a este medidor NO se agregan al medidor Total Coin Out.

Hand Pay: Pago manual al cliente realizado por el personal del casino

Juegos Jugados: Recuento total de juegos jugados en la máquina de juego. Se utiliza para calcular la apuesta promedio por juego y como medida bruta de la actividad del casino.

ACK: Reconocimiento enviado por el host y la máquina de juego durante las comunicaciones para indicar que la transmisión se recibió correctamente.

NACK: Reconocimiento negativo enviado por el host y la máquina de juego durante las comunicaciones para indicar que los datos transmitidos no son válidos o no se recibieron.

SMIB: Placa de interfaz inteligente.

BCD: Decimal codificado en binario empaquetado, formato en el que se enviarán los números.

CRC: Comprobación de redundancia cíclica. Método para verificar la validez de los datos transmitidos.

Idle: Estado de inactividad o reposo.

SAS: Slot Accounting System.

Long poll: Encuesta larga.

General poll: Encuesta general.

Signals & slots: Señales y ranuras.

Request: Petición.

Response: Respuesta.

9. Referencias bibliográficas

[1] P. Coljuegos, "Sala de prensa", *Coljuegos.gov.co*, 2021. [Online]. Available at: https://www.coljuegos.gov.co/documentos/200365/sala-de-prensa/.
[2] Revista Gestión.ec. 2021. *Un viaje por la historia de las apuestas | Gestión*. [online] Available at: <https://www.revistagestion.ec/index.php/cifras/un-viaje-por-la-historia-de-las-apuestas>.
[3] Gil Corredor, O. L. (2019). *El IVA en los juegos de suerte y azar en Colombia* (Doctoral dissertation, Bogotá: Universidad Externado de Colombia, 2019.).
[4] Team, J., 2021. JHipster - Full Stack Platform for the Modern Developer!. [online] Jhipster.tech. Available at: <https://www.jhipster.tech/>.
[5] Spring. 2021. Spring makes Java simple.. [online] Available at: <https://spring.io/>.
[6] 2021. [online] Available at: <https://platzi.com/tutoriales/1545-c-plus-plus/9141-que-es-la-programacion-orientada-a-objetos-poo/>
[7] 2021. [online] Available at: <https://platzi.com/clases/1545-c-plus-plus-2019/19061-que-es-programar-y-por-que-aprender-c/>
[8] Chávez Andrade, J. V. (2019). *Estandarización de los procesos de desarrollo de software utilizando buenas prácticas de programación y SCRUM como marco de trabajo ágil en departamentos de TI* (Master's thesis, Universidad Técnica de Ambato. Facultad de Ingeniería en Sistemas, Electrónica e Industrial. Maestría en Gerencia de Sistemas de Información).
[9] Desarrolloweb.com. 2021. Qué es la programacion orientada a objetos.
[10] IGT: Slot Accounting system version 6.01 (2003)
[11] Kim, S., & Ahn, H. (2014). Design and Implementation of Casino Slot Machine Accounting Protocol Engine. *Advanced Science and Technology Letters*, *54*, 65-68.
[12] Kim, S., & Ahn, H. (2014). Open source-based G2S (game to system) engine design and implementation. *Contemporary Engineering Sciences*, *7*(22), 1171-1179.
[13] Coljuegos (2019). Requerimientos técnicos de máquinas electrónicas tragamonedas - MET.
[14]"Diferencias entre Unity, Unreal Engine y Godot", OpenWebinars.net, 2022. [Online]. Available: https://openwebinars.net/blog/ventajas-diferencias-unity-unreal-engine-godot/.
[15]"About Qt - Qt Wiki", *Wiki.qt.io*, 2022. [Online]. Available: https://wiki.qt.io/About_Qt.
[16] “Signals & Slots: Qt core 5.15.8,” *Signals & Slots | Qt Core 5.15.8*. [Online]. Available: https://doc.qt.io/qt-5/signalsandslots.html.
[17] "QSerialPort Class | Qt Serial Port 5.15.8", Doc.qt.io, 2022. [Online]. Available: https://doc.qt.io/qt-5/qserialport.html.
[18]"QByteArray Class | Qt Core 5.15.8", Doc.qt.io, 2022. [Online]. Available: https://doc.qt.io/qt-5/qbytearray.html
[19]"QTcpServer Class | Qt Network 5.15.8", Doc.qt.io, 2022. [Online]. Available: https://doc.qt.io/qt-5/qtcpserver.html.
[20] W. Gao and Z. Tang, "Realization of CRC (Cyclic Redundancy Check) Based on LabView". Available: 10.4028/www.scientific.net/amm.530-531.686
.