DESARROLLO DE UN LABORATORIO VIRTUAL DE FÍSICA I, CON ACCESO REMOTO DESDE LA RED LAN DE LA CUAO

ANDRÉS ALBARRACÍN BRAVO
LEONARDO CALDERÓN JARAMILLO

CORPORACIÓN UNIVERSITARIA AUTÓNOMA DE OCCIDENTE
DIVISIÓN DE INGENIERÍAS
PROGRAMA INGENIERIA ELECTRÓNICA
PROGRAMA INGENIERIA MECATRONICA
SANTIAGO DE CALI
2.003

DESARROLLO DE UN LABORATORIO VIRTUAL DE FÍSICA I, CON ACCESO REMOTO DESDE LA RED LAN DE LA CUAO

ANDRÉS ALBARRACÍN BRAVO
LEONARDO CALDERÓN JARAMILLO

PROYECTO DE GRADO PARA OPTAR A LOS TÍTULOS DE INGENIERO ELECTRÓNICO Y MECATRONICO

DIRECTOR
ING. DIEGO MARTÍNEZ.
DOCENTE PROGRAMA DE INGENIERÍA ELECTRÓNICA.

CORPORACIÓN UNIVERSITARIA AUTÓNOMA DE OCCIDENTE
DIVISIÓN DE INGENIERÍAS
PROGRAMA INGENIERIA ELECTRÓNICA
PROGRAMA INGENIERIA MECATRONICA
SANTIAGO DE CALI
2.003

Nota de aceptación

Trabajo aprobado por el comité de grado en cumplimiento de los requisitos exigidos por la Corporación Universitaria Autónoma De Occidente para optar al título de ingeniero electrónico y mecatrónico

JESÚS ROBERTO SOTO G

jurado

ZEIDA MARIA SOLARTE ASTAIZA

jurado

DIEGO FERNANDO ALMARIO ALVAREZ

jurado

Santiago de Cali, 4 de Junio del 2.003

Dedico este trabajo a mis padres Maria Eugenia Bravo y Oscar Albarracín Posada por todos sus esfuerzos al brindarme la oportunidad de una formación profesional. Igualmente a todas aquellas personas que me apoyaron en el transcurso de la carrera, en particular a la Dra. Angélica Maria Bejarano por su sincera amistad.

Andrés Albarracín Bravo.

Dedico este esfuerzo a las personas que estuvieron conmigo a lo largo de estos años de academia, a mi novia Milena Correa Briceño y a mi padre Efraín Calderón Ramírez. A la Dra. Cecilia, la Dra. Ana Belén, la Dra. Ma. Cristina Ávila y el Dr. Roberto Soto, infinitos agradecimientos por todo el apoyo brindado en los momentos más difíciles.

Leonardo Calderón Jaramillo.

AGRADECIMIENTOS

Agradecimiento muy sincero a la División de Ciencias Básicas y la Vicerrectoría de Investigaciones por su apoyo en la planeación y realización del proyecto, en especial al profesor Robert Sánchez por su continuo acompañamiento y colaboración durante todo el proceso.

A nuestro director de tesis, el Ing. Diego Martínez por su confianza depositada. A la Dra. Angélica María Bejarano por sus sabios aportes y acompañamiento del trabajo. A Jairo Camayo y Jimmy Suárez por su colaboración en el área de diseño y comunicación.

CONTENIDO

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

LISTA DE ANEXOS

RESUMEN

En la primera etapa del proyecto, llamada de *Monitoreo*, se desarrollo un sistema Hardware-Software para la adquisición de variables (análogas y digitales) y realizar la práctica *Movimiento Rectilíneo Uniforme* mediante la adecuación de una planta del laboratorio de física 1 (carril neumático), el diseño de una tarjeta de adquisición de datos microcontrolada y un programa de monitoreo utilizando el puerto serial hacia el computador.

En la segunda etapa se desarrolló un sistema de *laboratorio virtual* (remoto) sobre la misma planta, basado en el sistema de monitoreo, para lo cual fue necesario implementar nuevas características hardware que garanticen un control adecuado sobre la planta para los requerimientos de un mando a distancia mediante una red.

INTRODUCCIÓN

El proyecto se basa en al necesidad de integrar las nuevas tecnologías de la información y las comunicaciones (NTIC's) con la práctica y la enseñanza de la electrónica y áreas afines. Su desarrollo se planteó en dos etapas independientes y complementarias a la vez, cada una con un mayor nivel de cobertura para los usuarios.

El software se desarrolló bajo plataforma Java, teniendo como principales funciones la inicialización del sistema, la comunicación con la tarjeta de adquisición, la recopilación, graficación y visualización de la información de manera constante.

El sistema finalmente permite a uno o varios estudiantes mediante un PC de las salas de computo de la universidad, acceder sobre la red LAN interna a un software de interfaz remota, permitiéndole la interacción con la planta para realizar las tomas de datos correspondientes a la práctica y utilizarlos para su tabulación, graficado, animación e interpretación de los resultados.

1. OBJETIVOS

1.1 OBJETIVO GENERAL

Desarrollar un sistema que posibilite la prestación de los servicios de laboratorio de física I de manera remota con acceso a la planta *Carril de Aire* para realizar la practica *Movimiento rectilíneo uniforme*.

1.2 OBJETIVOS ESPECÍFICOS

- Implementar un sistema Hardware a partir del acondicionamiento previo realizado en el proyecto *Sistematización de laboratorios de física I,* que permita la adecuación de la planta a los requerimientos necesarios para el desarrollo del laboratorio remoto.
- Desarrollar e implementar el software necesario bajo una arquitectura Cliente-Servidor que permita al usuario acceder a la planta y realizar la practica de laboratorio de manera remota.
- Validar el funcionamiento del sistema a desarrollar, mediante la ejecución de pruebas de acceso, interacción, robustez y fiabilidad de la planta

2. ANTECEDENTES

Los procesos de formación académica y profesional se han venido acompañando de avances tecnológicos que permiten mejorar la calidad y confiabilidad de la transmisión y adquisición de conocimientos prácticos y teóricos. Estos avances brindan la posibilidad de implementar aplicaciones que soporten el afianzamiento de conceptos en las diversas prácticas de laboratorio propuestas para cursos básicos de ingeniería.

Así por ejemplo, las prácticas de laboratorio han evolucionado con la aparición y desarrollo de la electrónica en las áreas de la instrumentación, informática y las telecomunicaciones. De esta manera se ha buscado vincular el uso del computador y las comunicaciones para transmitir la información desde el lugar mismo de la práctica hacia cualquier lugar del mundo.

A pesar de esto, se viene observando que la mayoría de los laboratorios asequibles a través de Internet están constituidos por simulaciones de modelos teóricos sin interacción directa con la planta. Por ello se han

iniciado procesos de investigación y desarrollo en el mundo entero para la realización de los llamados *Laboratorios Remotos*.

Estos procesos se llevan a cabo actualmente en instituciones como el ITESM(Instituto Tecnológico de Estudios Superiores de Monterrey), donde se desarrolla un proyecto de laboratorio virtual de robótica cuyo principal objetivo es implementar herramientas que permitan desarrollar experimentos sobre equipos reales, particularmente en robótica móvil de manera remota. El proyecto tiene desarrollada una primera etapa que consiste en la conexión de un robot móvil a la red Internet para su manipulación a distancia mediante una serie de comandos simples.

La segunda y tercera fase se encuentran en desarrollo y consisten en la realización de modelos tridimensionales y la implementación de métodos en lenguaje Java que permitan integrar los módulos anteriores dando mayores capacidades al usuario como la realización de programas para robots en forma remota. Este Proyecto es dirigido actualmente por el Dr. Luis Enrique Sucar Succar, Profesor Investigador de la división de Ingenierías y ciencias.

En Colombia, también se han iniciado procesos de implementación de laboratorios remotos en universidades como la de Antioquia, donde se desarrolla un proyecto dirigido por el Ingeniero José Edinson Aedo que busca desarrollar una tecnología Hardware-Software para realizar experimentos de electrónica en forma remota, usando Internet y el lenguaje de programación Java y la utilización de Servlets.

De igual forma, en la Corporación Universitaria Autónoma de Occidente se ha iniciado el proceso de realización de laboratorios a distancia de física mediante el proyecto "*Sistematización de los Laboratorios de Física I* " en el que se desarrollo el acondicionamiento de plantas, la realización de la interfaz *Autofisica1* y la implementación del Software de adquisición de datos y simulaciones para las diferentes practicas del curso Física I.

En esta oportunidad, el proyecto culminó con la sistematización de las prácticas de laboratorio mas no con la implementación del laboratorio remoto.

Con la realización del servicio "*Laboratorio remoto de física I* " para la CUAO, se pretende desarrollar sobre la plataforma tecnológica que

posee actualmente, el primer servicio de realización de un laboratorio remoto para los estudiante de la asignatura Física I, de manera que puedan realizar las prácticas correspondientes a la planta C*arril de Aire* sin necesidad de un desplazamiento al laboratorio. Brindando con ello una mejor prestación de los laboratorios, una adquisición de los datos mas confiables y la aplicación de las nuevas tecnologías de la información a la formación académica.

3. JUSTIFICACION

En la formación de todo profesional en la áreas de ingeniería, las asignaturas básicas constituyen un componente esencial para brindar herramientas necesarias para la asimilación de conceptos mas especializados aplicados a la solución de problemas.

La comprensión de las leyes físicas es un núcleo fundamental en las ciencias básicas como soporte para el desarrollo de nuevos procesos, productos o materiales, de manera que se hace necesaria la realización de prácticas de laboratorio adecuadas, enfocadas tanto al afianzamiento de los conceptos como a la contrastación de modelos, las cuales se han desarrollado tradicionalmente de forma presencial. A medida que la demanda de formación universitaria crece, los paradigmas de formación académica deben ser replanteados teniendo en cuenta los avances tecnológicos en el área del manejo de la información.

Con el surgimiento de las nuevas tecnologías de la información y las comunicaciones (NTIC`s), se brindan nuevos espacios pedagógicos y comunicativos en los cuales el estudiante cuenta con mejores

herramientas para interactuar y abstraer información, aumentando de esta manera el componente pedagógico para dedicarse al estudio del tema en particular sin preocuparse de los por menores adicionales de la práctica, enfocando su atención en los fenómenos de interés.

De esta manera, la implementación de laboratorio remotos brinda varias ventajas considerables. En primer lugar facilita al estudiante la realización de las prácticas, una sistematización para mejorar la eficiencia y una generación de datos mas confiables para su posterior análisis. En segundo lugar, beneficia a la institución, por permitir un cubrimiento más amplio de la población estudiantil, avanzando otro escalón importante hacia la realización de cursos totalmente a distancia utilizando los recursos tecnológicos con que cuenta la Universidad mejorando la calidad de la formación académica.

Así el estudiante podrá contar con herramientas para la adquisición y procesamiento de datos reales de manera remota, el uso de simulaciones virtuales interactivas que permitan realizar comparaciones entre la teoría y la práctica y el uso de gráficas para visualizar el comportamiento de las variables, lo cual permitirá procesos de afianzamiento, comprensión y aplicación de conceptos.

4. LABORATORIOS REMOTOS

El propósito de diseñar y desarrollar laboratorios de acceso remoto para el área de física, es un asunto que demanda un minucioso análisis, puesto que no se trata tan sólo de hacer viable la incorporación de las tecnologías de la información y la comunicación (NTIC´s), a un diseño experimental específico, buscando ampliar las posibilidades de acceso de los estudiantes a prácticas de laboratorio concretas, lo cual de por sí ya, en cierta forma, implica una ganancia en términos de prestación y ampliación de un servicio, sino que además de eso, la complejidad del tema se hace evidente una vez se reconoce que, un laboratorio de acceso remoto debe, como mínimo, garantizar el logro de los objetivos de aprendizaje propuestos para cada una de las prácticas a realizar, las cuales también se ejecutan en el laboratorio de manera presencial, utilizando la organización instrumental predispuesta para tal efecto.

Básicamente, un laboratorio remoto se puede definir como la interacción a distancia que se establece entre un agente observador y una organización instrumental adecuada para la contrastación de procesos y modelos teóricos, a partir de la recolección e interpretación de datos

obtenidos mediante el mismo instrumental sobre un medio o red de comunicación que así lo permita.

De esta manera, la utilización de las plataformas tecnológicas que se han desarrollado para la implementación de este tipo de aplicaciones, permiten al estudiante llevar a cabo procesos de identificación, comparación, análisis y verificación del fenómeno implicado en el diseño experimental, mediante el uso de herramientas interactivas como simulaciones, análisis gráficos, generación de datos, a través del acceso remoto.

Finalmente tales posibilidades en términos funcionales, contribuirán con el logro de los propósitos de formación y en el desarrollo de las competencias de investigación previstas en el área de física para los estudiantes de Ingeniería de la CUAO.

5. DESCRIPCIÓN FUNCIONAL DEL SISTEMA

El sistema completo se concibe bajo una arquitectura predominantemente modular en todos los niveles, teniendo en cuenta los siguientes puntos:

?? Facilidad de diseño, construcción, mantenimiento y la plantación de una línea de sistemas similares para la implementación de otras prácticas de laboratorio diferente a la propuesta.

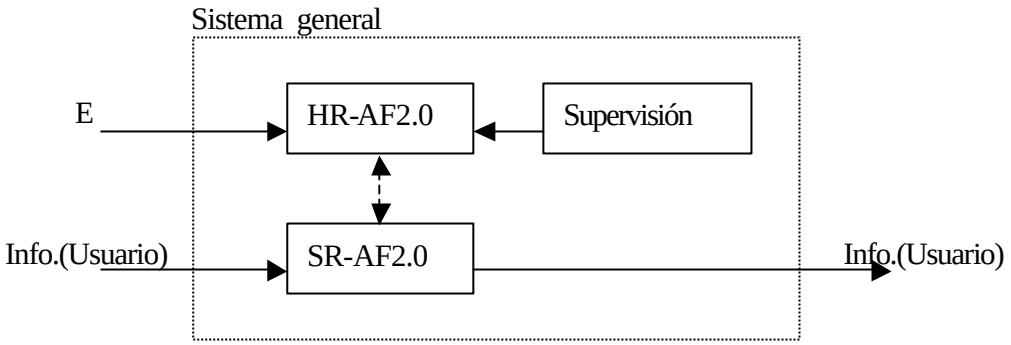
?? Los factores económicos no son un punto crítico a tener en cuenta y unas prestaciones medianamente buenas son satisfactorias, por lo cual no se desarrollo una mayor integración.

Desde este enfoque el sistema se divide en dos módulos principales altamente desacoplados los cuales se llamarán en adelante:

?? Hardware Remoto Autofísica 2.0 (HR-AF2.0)

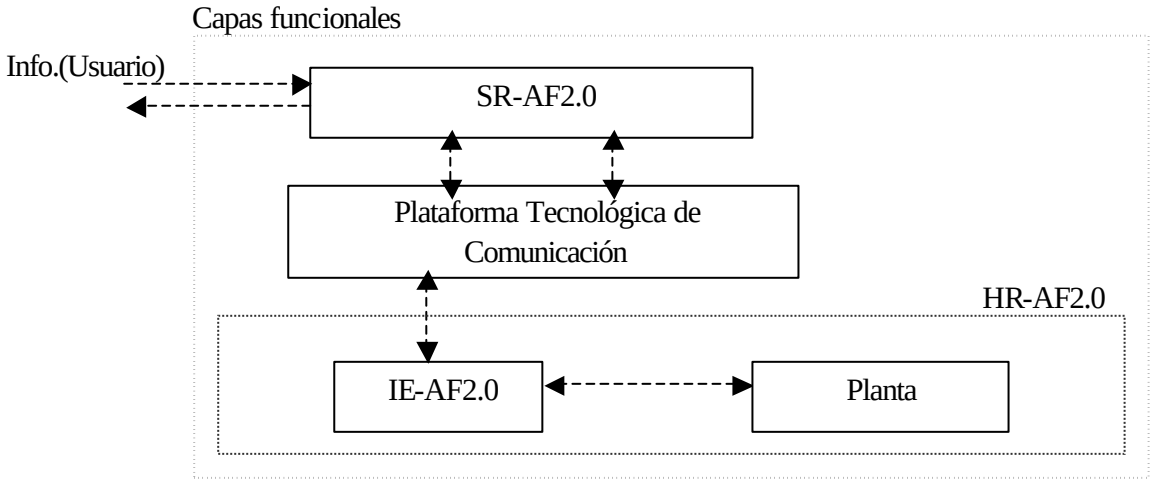
?? Software Remoto Autofísica 2.0 (SR-AF2.0)

Figura 1. Diagrama funcional del sistema general



El usuario inicialmente suministra unos datos al sistema para realizar la conexión, verificación e inicialización del sistema, procesando la información a través del *SR-AF2.0* el cual inicia la comunicación con HR-AF2.0, le transmite las peticiones del usuario y devuelve los resultados (Datos).

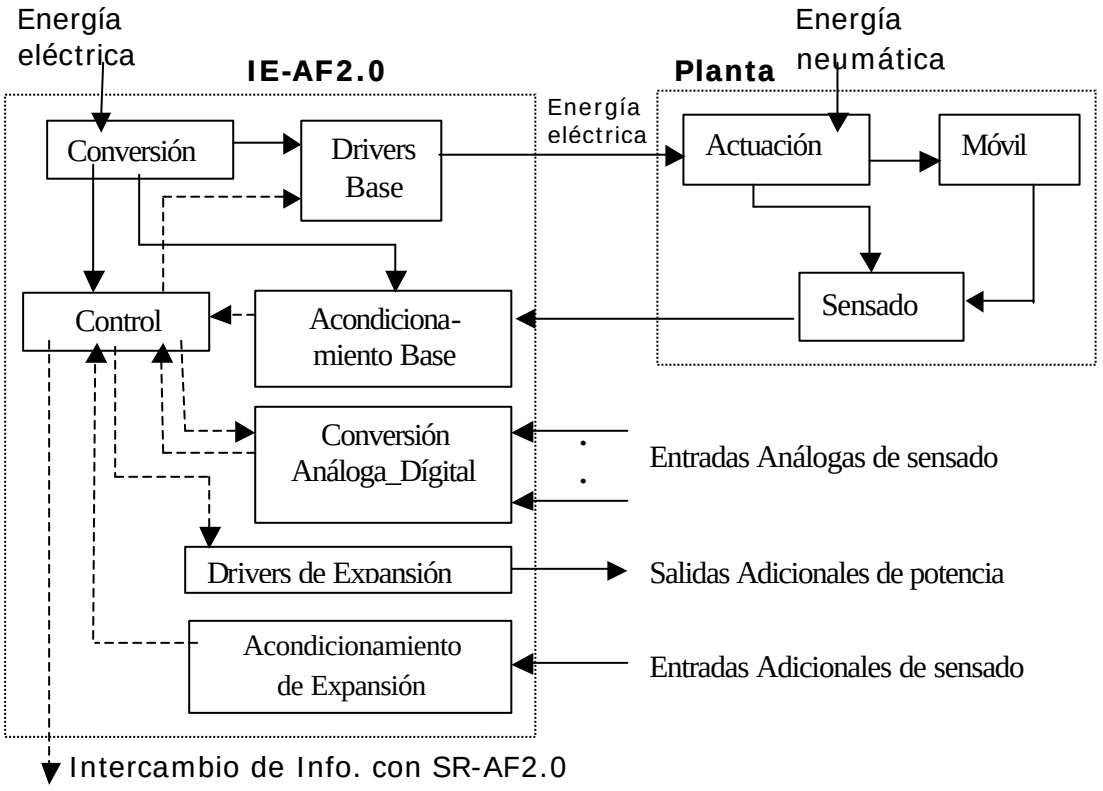
Figura 2. Diagrama funcional del sistema general por capas.



5.1 DESCRIPCIÓN DEL MÓDULO HARDWARE REMOTO (HR_AF2.0)

El HR-AF2.0 consta de dos módulos separados, la Interfaz Electrónica Autofísica 2.0 (IE-AF2.0) y la Planta propiamente dicha. Esta interfaz recibe las ordenes a través de un sistema microcontrolador encargado de ejecutarlas mediante el accionamiento de unos drivers que dirigen el flujo de energía previamente convertida y el procesamiento de la información de la planta para luego retornar los datos correspondientes hacia el SR-AF2.0.

Figura 3. Diagrama funcional del HR-AF2.0.



5.1.1 Interfaz electrónica IE_AF2.0: La IE-AF2.0 se compone de los siguientes submódulos funcionales.

Conversión**:** Convierte los niveles y formas energía entrante en niveles y formas de energía útil para el funcionamiento y operación de los módulos a los cuales provee (Control, Drivers y Acondicionamiento base, Conversión analoga_digital, drivers y acondicionamiento de expansión).

Convirtiendo mediante la implementación de una fuente regulada 120Vac a los diferentes voltajes requeridos (5Vdc, 12Vdc, 24Vdc, con requerimientos de corriente de 1 Amp máximo).

La alternativa seleccionada es de bajo costo y cumple con los requisitos electrónicos de los sistemas asociados dado que no se manejan altas prestaciones de potencia y eficiencia.

Drivers Base: Maneja y proporciona los niveles de señales necesarios para el funcionamiento del módulo de *Actuación* de la *Planta*, bajo el mando de señales eléctricas del módulo de *Control* de IE-AF2.0.
En este caso se utilizan dispositivos semiconductores (Transistores BJT de referencia TIP122) para el disparo de las señales de actuación

(señales en niveles de 0-9V para el motor PAP y 0-24V para el actuador de disparo del móvil).

Acondicionamiento Base: Este submódulo se encarga del acondicionamiento de señal, proporcionando niveles adecuados de señal para realizar el correspondiente procesamiento por parte del submódulo de *Control*. A su vez mejora la calidad de la señal proveniente del sistema de sensado de la planta, mediante un arreglo electrónico digital. Este submódulo se ha implementado por medio de configuraciones *AntiRebote* con dispositivos digitales de tecnología TTL (circuito integrado 74LS14) con el fin de evitar posibles fluctuaciones de la señal proveniente de los sensores y contactores, a demás de mejorar la calidad de la señal a procesar.

Control: este módulo se encarga de procesar los datos provenientes del módulo de *Sensado* de la *Planta* que han sido previamente acondicionados por el módulo *Acondicionamiento Base*, emitir ordenes para los módulos *Drivers Base* y *Drivers de expansión* y responder ante peticiones y requerimientos del sistema software. Así mismo, establece funciones de comunicación y secuencias de control de acciones.

Adicional a esto, este módulo puede realizar la gestión y manejo del módulo de C*onversión Analgo_Digital* para obtener valores digitales de entradas análogas y realizar un procesamiento llegado el caso de utilizar dicho módulo de conversión. Para nuestros requerimientos este módulo no es utilizado pero se ha implementado para efectos de expansibilidad.

El módulo *Control* se implementa mediante un microcontrolador de la familia 8952 como núcleo central de una tarjeta expansible para la adquisición y procesamiento de datos, SISDEI (Ver glosario).

Conversión Analoga_Digital: Este submódulo se concibe para efectos de expansibilidad y de escalabilidad de la interfaz. En nuestro caso no es utilizado, pero su función puede ser muy importante en futuras implementaciones. Este componente del modelo permite realizar la conversión de señales análogas a señales digitales para realizar el procesamiento de dichas señales mediante el módulo *Control*.

Así mismo el módulo *Control* ejerce la gestión sobre este dispositivo mediante señales bien conocidas de la tarjeta SISDEI. La implementación de este módulo se realiza por medio de un dispositivo

electrónico de conversión cuya referencia es ADC0808 que presenta 8 canales de entrada análoga y una resolución de 8 bits.

Drivers de Expansión: Este submódulo brinda soporte para utilizar tres drivers de baja potencia para efectos de expansibilidad de la interfaz electrónica. Fundamentalmente se compone de dispositivos semiconductores idénticos a los utilizados en el módulo *Drivers Base*. Se han adicionado estos drivers para futuras implementaciones que requieran la utilización de otros actuadores de baja potencia.

Acondicionamiento de Expansión: Este submódulo brinda soporte para la utilización de entradas adicionales de sensado. La gestión de este módulo al igual que el procesamiento de los datos que capture, es realizado también por el módulo *Control*. Este componente del modelo se implementa de la misma forma que el módulo *Acondicionamiento Base* por medio de dispositivos digitales 74LS14 y arreglos *AntiRebote.*

5.1.2 Planta: Este módulo se compone de los siguientes submódulos funcionales.

Actuación: Se encarga por un lado de convertir la energía entrante en energía útil para interactuar con el módulo Móvil (conversión de energía neumática en energía mecánica). También se encarga de interactuar con el módulo de sensado, desplazando mecánicamente la configuración de sensores. Específicamente, este módulo refiere al sistema de actuación neumático (pistón neumático) que realizará el disparo del móvil y la sujeción del mismo y el sistema de actuación eléctrico (Motor de Pasos) que posicionará los sensores de acuerdo a peticiones de usuario. Este módulo se ha implementado mediante un sistema Electro_Neumatico para el disparo del móvil. Así mismo, se utiliza la configuración mecánica establecida para el movimiento del sensor de tiempos mediante al acción mecánica que realiza el motor de pasos PAP.

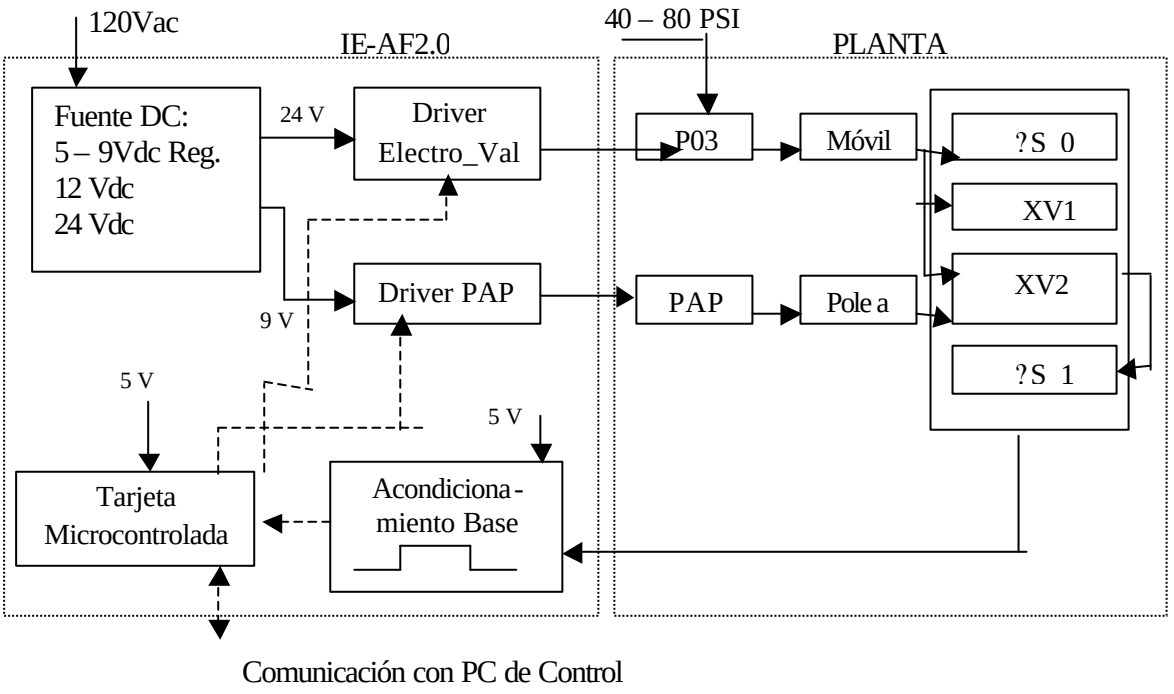
Móvil: Identifica el comportamiento funcional del móvil físico en la planta para estudiar el fenómeno en cuestión.

Sensado: Identifica toda la configuración instrumental de sensores utilizados para medir las variables del proceso, interactuando con el módulo Móvil y enviando datos al módulo de control.

Este módulo refiere en si a los sensores ópticos utilizados para obtener información temporal para el estudio del fenómeno y a los contactores finales de carrera adecuados para la señalización de eventos como posición inicial del móvil y posición inicial de sensores ópticos.

Un modelo funcional más especifico del módulo del hardware remoto es el siguiente:

Figura 4. Diagrama funcional específico del HR-AF2.0.



En este caso la simbología del sistema de sensores refiere a:

?S_0: Micro_switch que detecta la posición inicial del móvil (listo para el lanzamiento).

XV1: Primer sensor óptico que indica el punto de inicial del fenómeno (inicio de la toma del tiempo T1).

?S_1: Micro_switch que indica el punto de distancia inicial o de calibración de los sensores ópticos.

XV2: Segundo sensor óptico que indica el punto final del fenómeno (Toma de T2 y dt para determinar ?T). Este sensor corresponde al sensor móvil del sistema.

Fuente dc: Como se menciono se ha realizado un diseño sencillo de fuente DC regulada para obtener así los niveles de voltaje DC requeridos por la interfaz y la instrumentación de la planta.

Driver Electro_Valvula y Driver PAP: Estos dos drivers hacen parte del ya mencionado y especificado Módulo *Drivers Base* de la interfaz electrónica.

P03: Refiere a la especificación del instrumental Electro_Neumatico utilizado para el disparo del móvil. Esta especificación se mostrara mas adelante con sus diagramas instrumentales.

PAP: Refiere a la especificación del motor de pasos utilizado para el realizar la acción mecánica que ubicara el sensor XV2 en la posición deseada. Esta especificación se mostrará mas adelante.

Móvil: Refiere al dispositivo móvil utilizado para estudiar el fenómeno físico en cuestión. Hace parte del instrumental físico del carril de aire.

Polea: Se menciona como el mecanismo mediante el cual el motor de pasos ubica el sensor de tiempo XV2 en la posición deseada.

6 . IMPLEMENTACIÓN DE ALGORITMOS PARA EL MÓDULO DE CONTROL DE LA INTERFAZ IE-AF2.0

Como se mencionaba anteriormente, este módulo es el encargado de realizar la gestión de dispositivos, comunicación con la planta, procesamiento de datos y además de realizar el control secuencial de las operaciones a bajo nivel relacionadas con las peticiones de usuario. Esta interacción la realiza mediante un algoritmo secuencial de control que a su vez se basa en eventos y estados propios de la planta.

El algoritmo de control implementado para responder a las peticiones de los usuarios a bajo nivel es el mostrado en la figura 5.

6.1 RUTINA PLANTA_CONECTADA

Esta rutina indica al software de control que la planta se encuentra lista para funcionar adecuadamente una vez conectada la planta e inicializada la conexión remota (Ver figura 6).

Figura 5. Diagrama de flujo del ciclo principal del assembler.

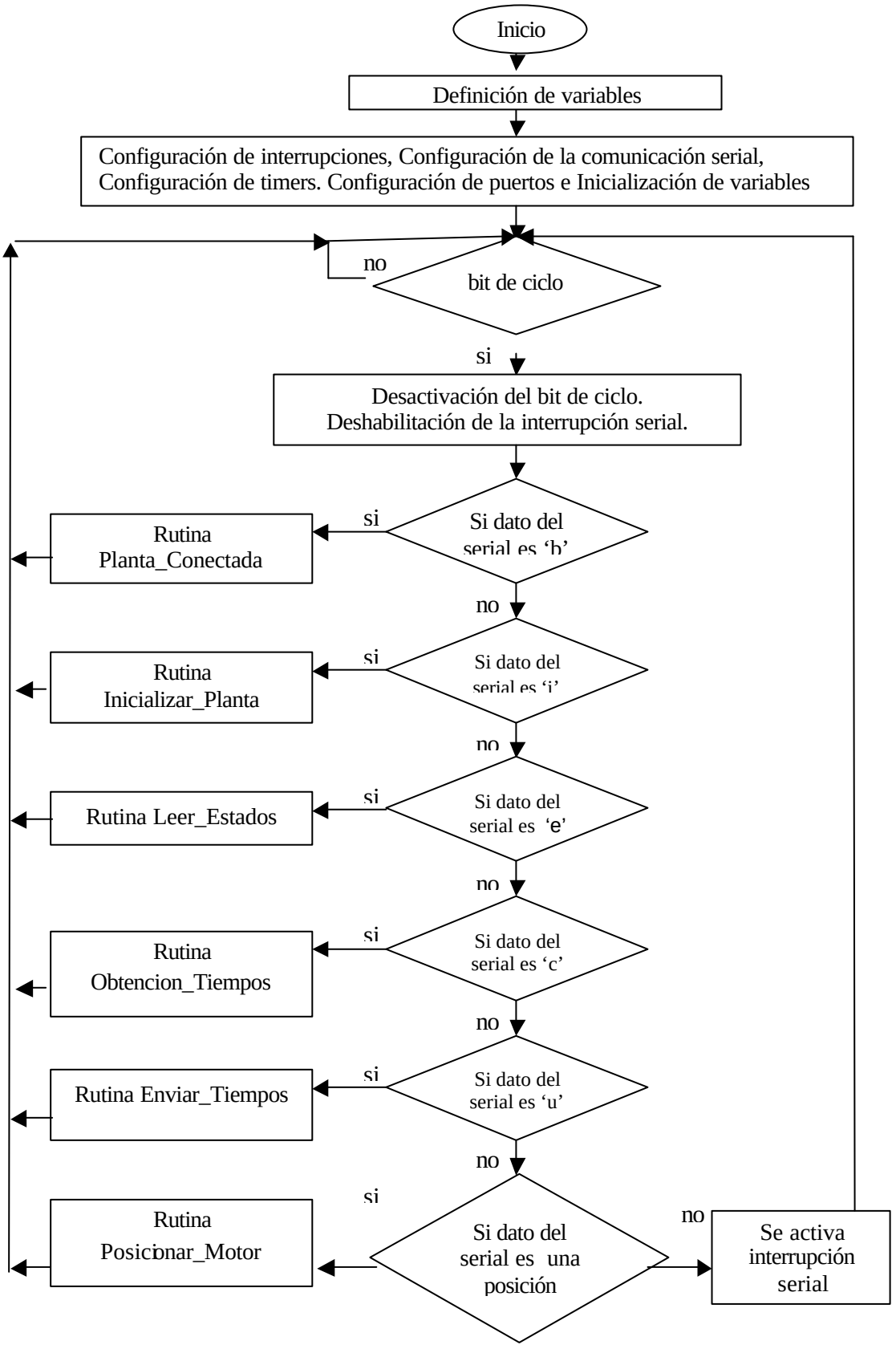
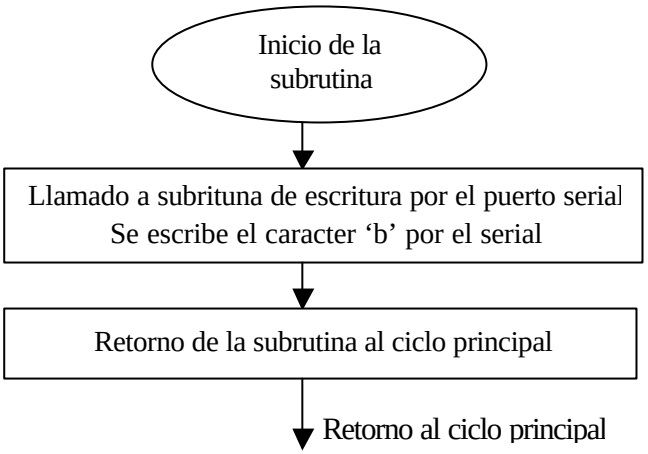


Figura 6. Diagrama de flujo de la subrutina Planta_Conectada



6.2 RUTINA INICIALIZAR_PLANTA.

Esta rutina, coloca en su lugar los sensores y elementos activos de la planta. Por lo cual no entrega al software de control una señal si la planta no esta bien inicializada. En este caso la planta se encuentra bien inicializada si el móvil y el sensor de tiempo XV1 se encuentra en la posición cero. Ver figura 7.

6.3 RUTINA LEER_ESTADOS.

Esta rutina es útil para leer los estado de la planta, es decir, observar el comportamiento de los sensores mediante la lectura del puerto al cual están asociadas sus señales. Ver figura 8.

Figura 7. Diagrama de flujo de la subrutina Inicializar_Planta

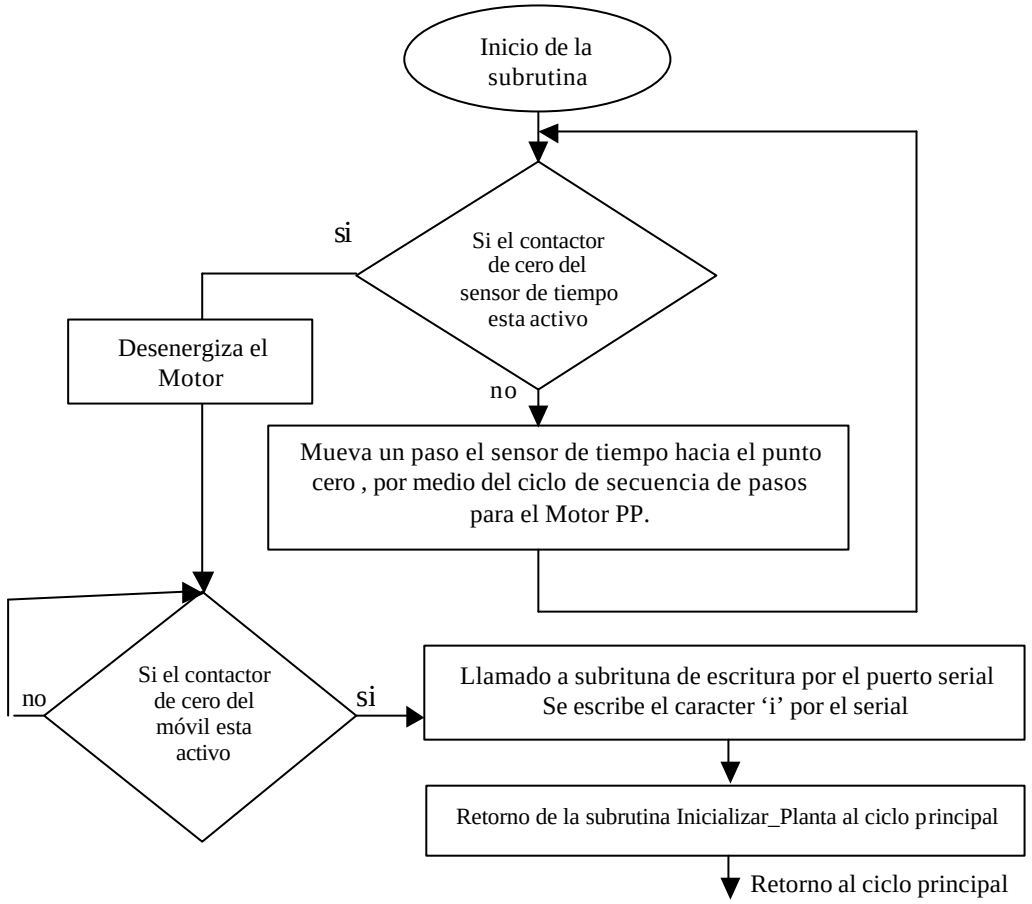
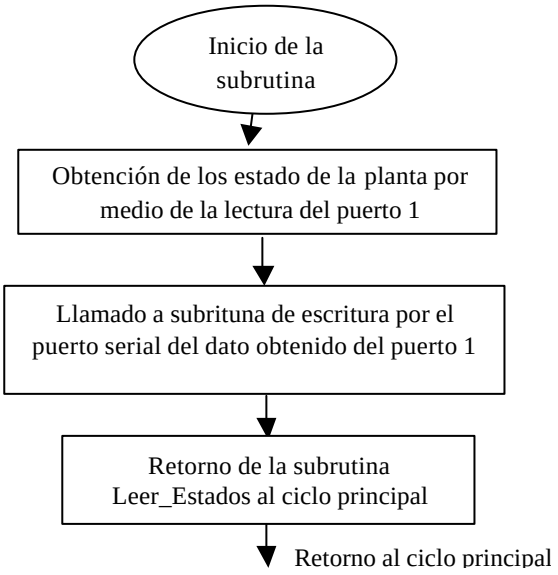


Figura 8. Diagrama de flujo de la subrutina Leer_Estados



6.4 RUTINA ENVIAR_TIEMPOS

Esta rutina permite enviar los datos almacenados en las variables internas de tiempos, que corresponden a los desbordamientos y a los valores de los registros del timer para ?T y ?t. En total los datos a enviar son: Numero de desbordamientos para ?T, valores de registros TH0 y TL0 para ?T, Numero de desbordamientos para ?t, valores de registros TH0 y TL0 para ?t. Ver figura 9.

6.5 RUTINA OBTENER TIEMPOS

Esta rutina permite la obtención de los tiempos ?t y ?T del fenómeno cinemático en cuestión. Ver figura 10.

Figura 9. Diagrama de flujo de la subrutina Enviar_Tiempos

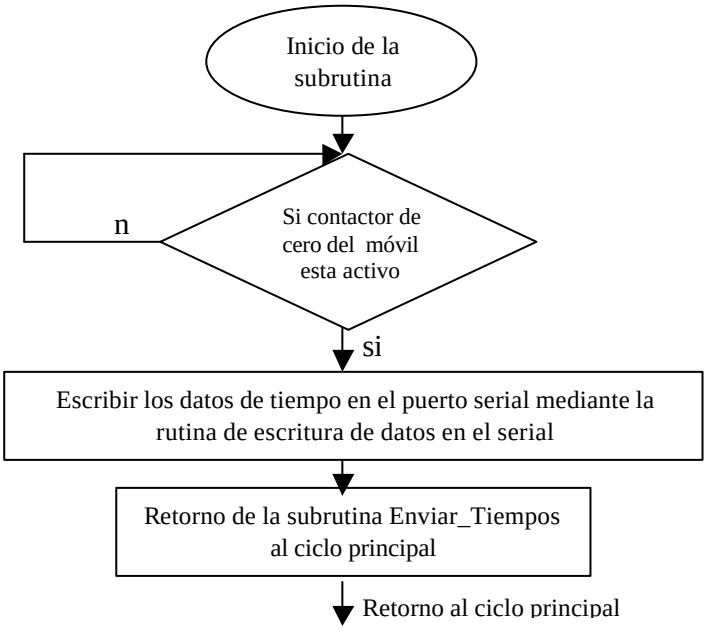
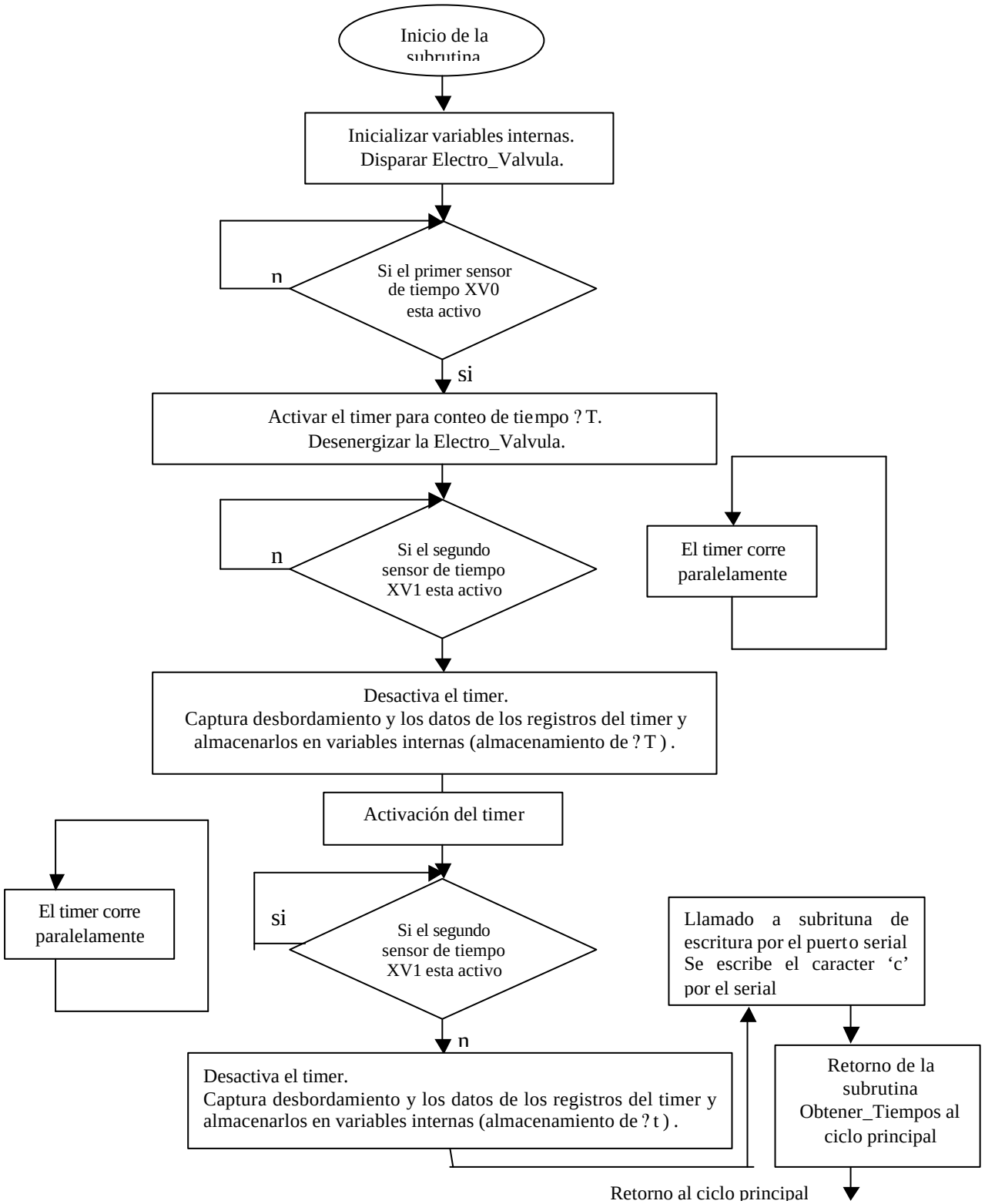


Figura 10. Diagrama de flujo de la subrutina Obtener_Tiempos



6.6 RUTINA POSICIONAR_MOTOR

En el software ensamblador se han definido dos funciones especificas que mueven el sensor de tiempo 10 cm bien sea hacia delante o hacia atrás. Esto se hace por medio de las rutinas ava_10 y ret_10 las cuales avanzan 10 cm el sensor y retroceden 10cm el sensor respectivamente (mas adelante explicaremos estas rutinas). Para posicionar el sensor de tiempo XV2 cada 10cm en un rango de 100cm se ha definido un rango propio de posicionamiento que va desde 40h(64d) hasta 4Ah (74d) para retroceder el sensor (40h retroceder 0cm y 4Ah retroceder 100cm) y otro rango que va desde 4bh(75d) hasta 55h(85d) para avanzar (4bh avanzar 0cm y 55h avanzar 100cm). Esto quiere decir que si el dato proveniente del puerto serial se encuentra en el rango de retroceso, el motor se accionara moviendo el sensor hasta la posición deseada en centímetros equivalente al numero en el rango . De igual manera sucede en caso de que el dato se encuentre en el rango de avanzar.

6.7 RUTINA RETROCEDER_SENSOR

Esta rutina permite accionar el Motor PP y mover así el sensor de tiempo XV1 hasta la posición deseada que se especifica por el dato que ha

llegado por el puerto serial y que se encuentra en el rango especificado para retroceder. Bien sea no avanzar o avanzar 100cm pero siempre de a 10 en 10cm. Ver figura 12.

Figura 11. Diagrama de flujo de la subrutina Posicionar_Motor

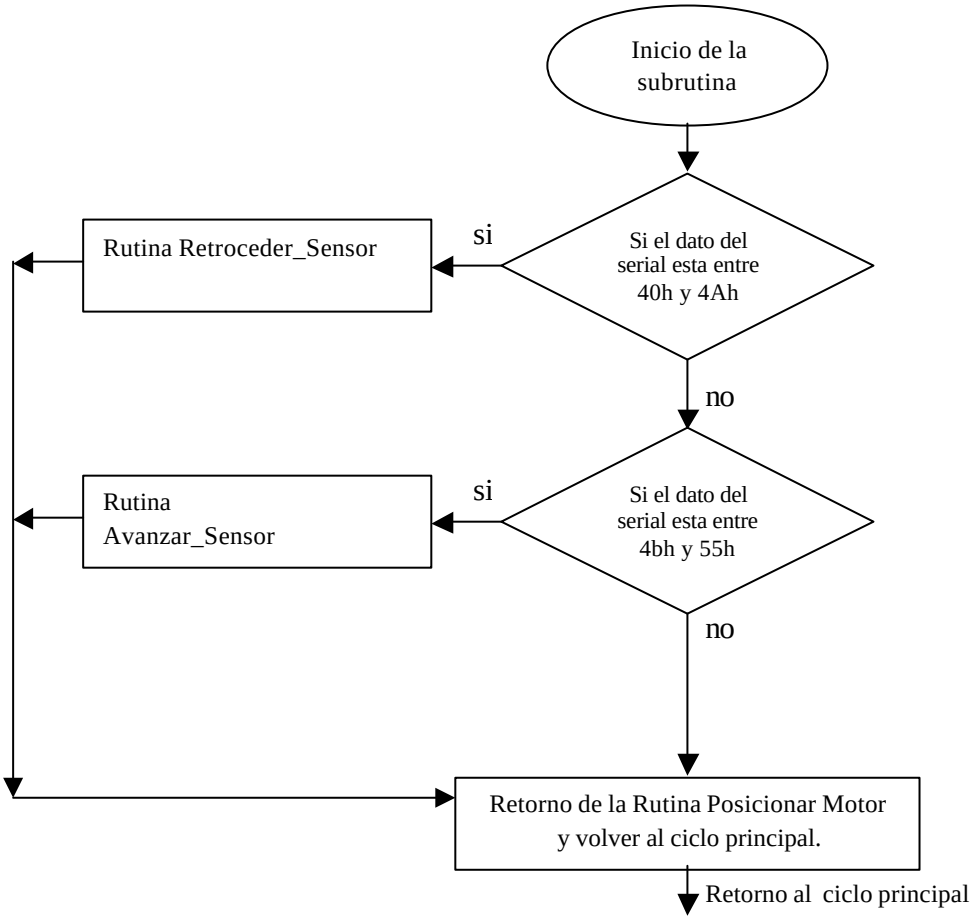
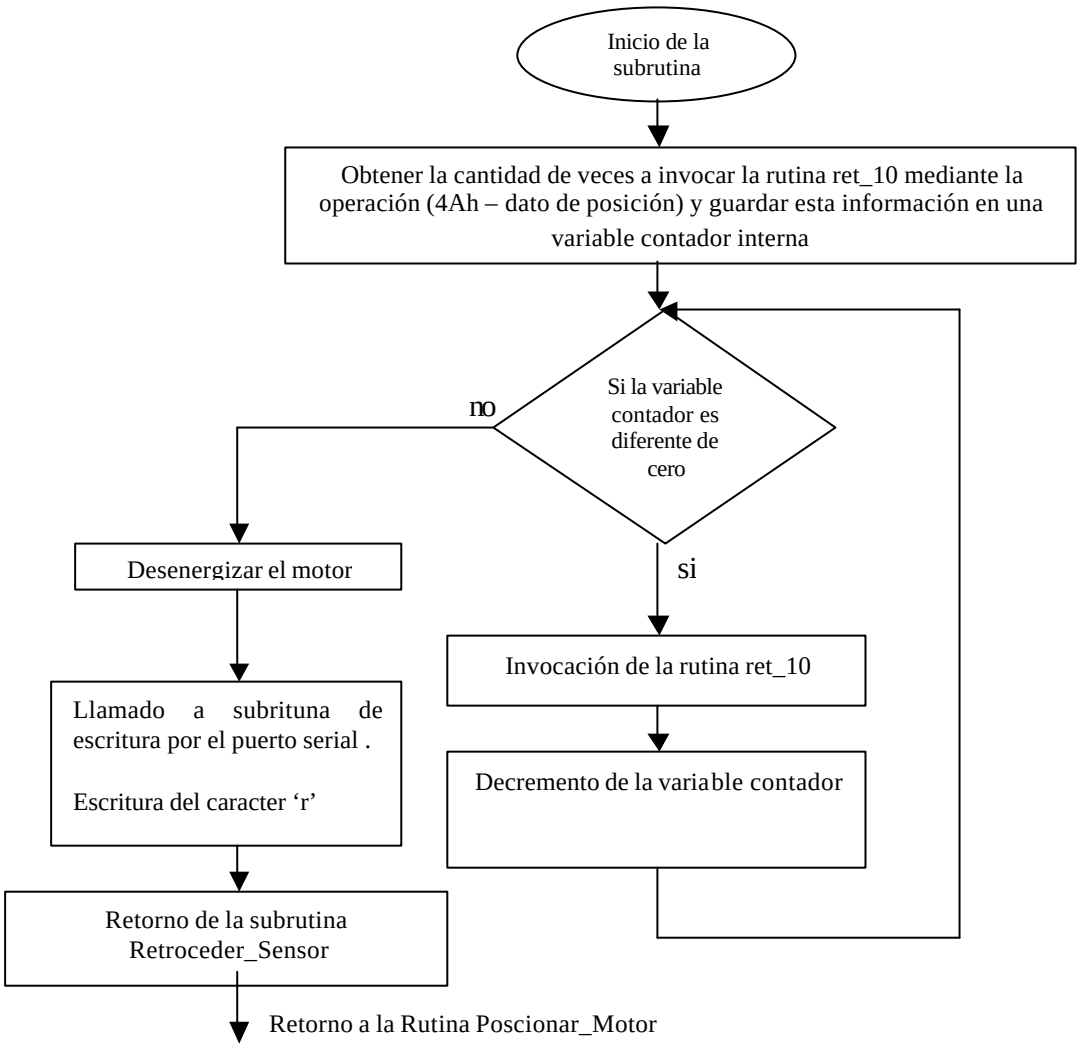


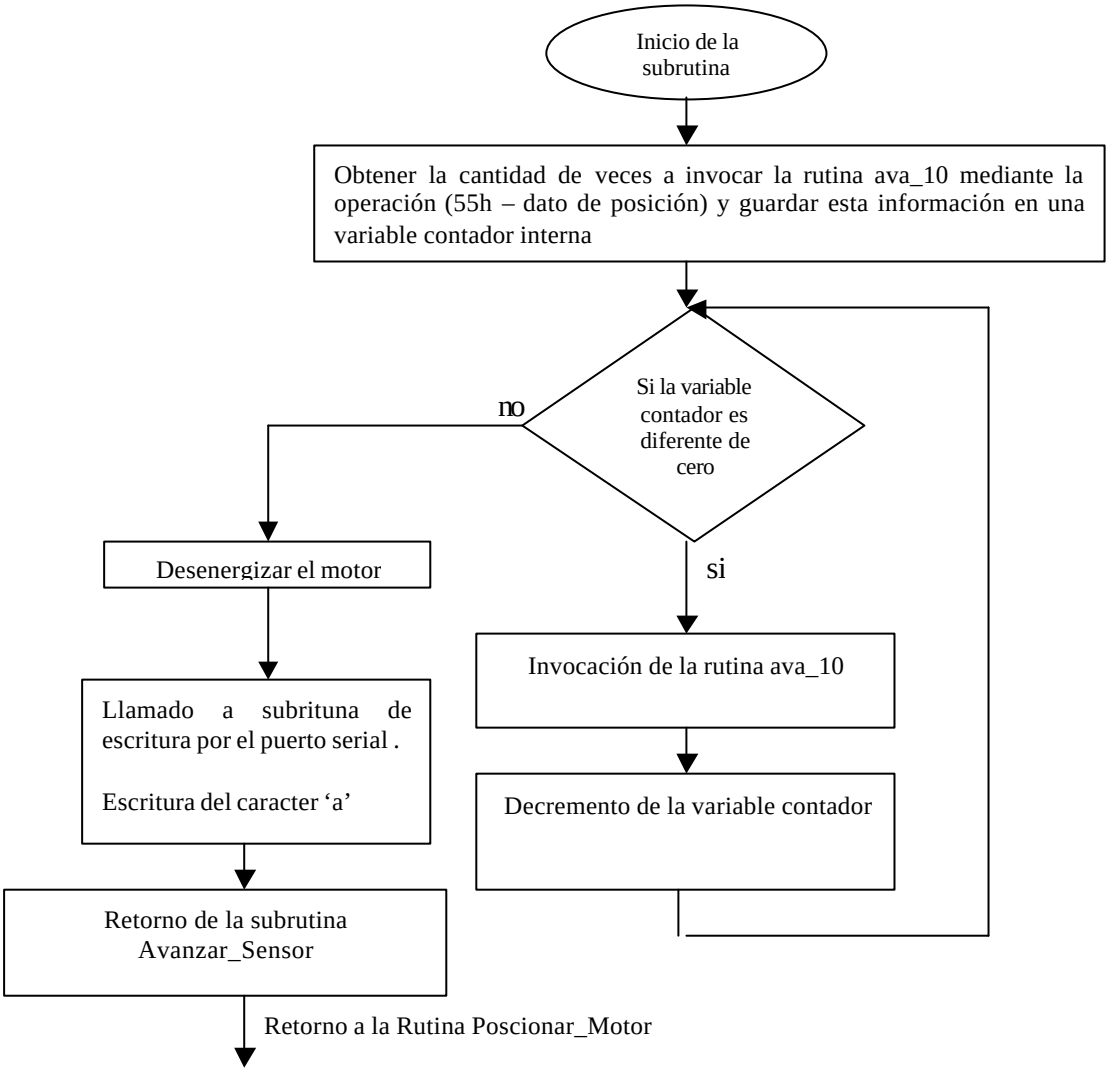
Figura 12. Diagrama de flujo de la subrutina Retroceder_Sensor



6.8 RUTINA AVANZAR_SENSOR

Esta rutina permite accionar el Motor PP y mover así el sensor de tiempo XV1 hasta la posición deseada que se especifica por el dato que ha llegado por el puerto serial y que se encuentra en el rango especificado para avanzar. Ver figura 13.

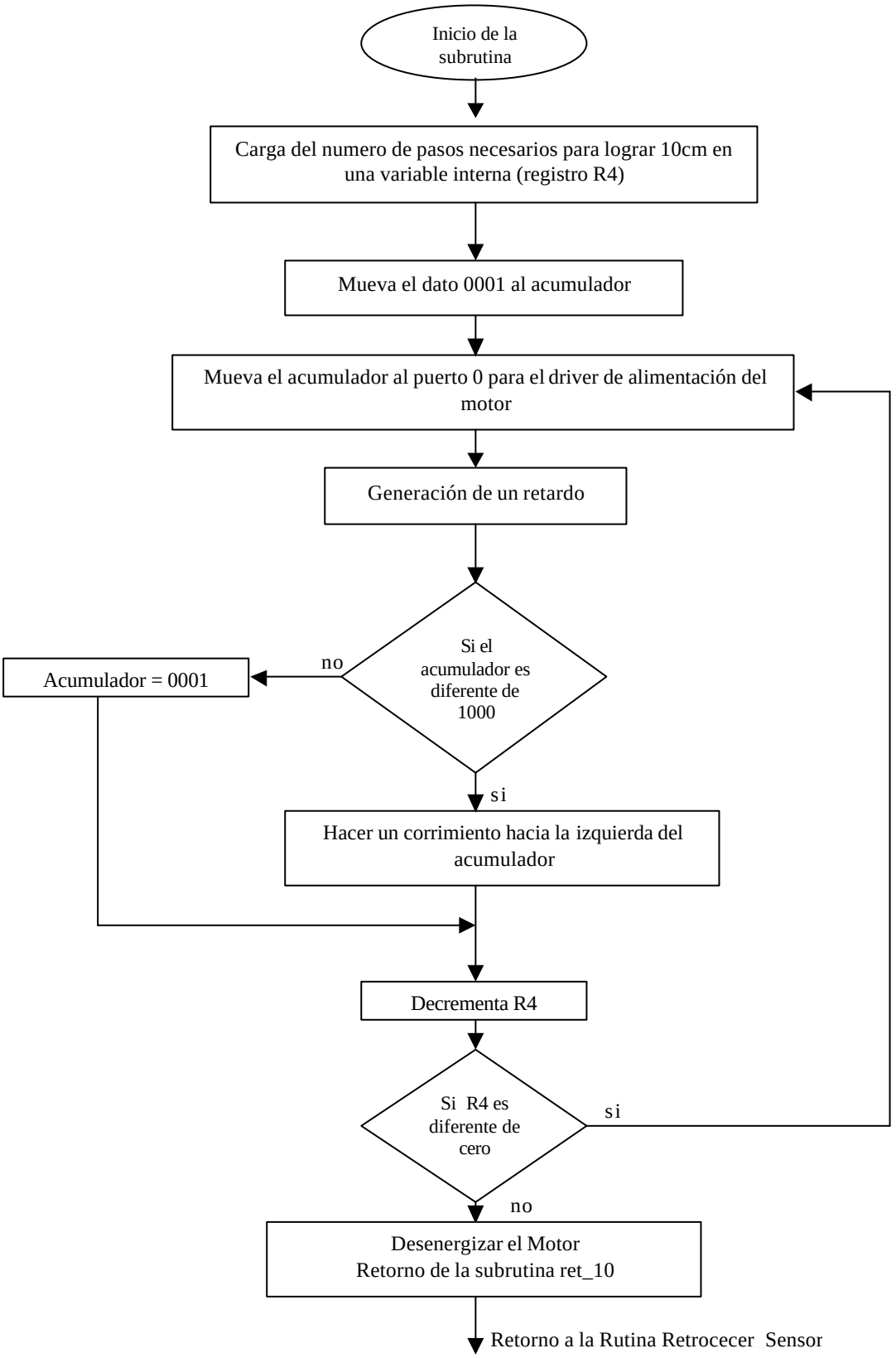
Figura 13. Diagrama de flujo de la subrutina Avanzar_Sensor



6.9 SUBRUTINA RET_10

La función de esta rutina es mover 10 cm el sensor hacia atrás y tener esta cantidad como base para siempre mover el sensor de a 10 en 10 cm. La secuencia de pulsos entregada al motor PP es una secuencia por olas. Ver figura 14.

Figura 14. Diagrama de flujo de la subrutina Ret_10



6.10 SUBRUTINA AVA_10

La función de esta rutina es mover 10 cm el sensor hacia adelante y tener esta cantidad como base para siempre mover el sensor de a 10 en 10 cm. La secuencia de pulsos entregada al motor PP es una secuencia por olas. Ver figura 15.

6.11 SUBRUTINA LECTURA DEL PUERTO SERIAL

Esta subrutina es invocada cada vez que se ejecuta una llamada a la subrutina de interrupción serial, por lo cual solo almacena el dato que llega por el serial en un registro interno(R2) para su posterior evaluación en el ciclo principal. Ver figura 16.

Figura 15. Diagrama de flujo de la subrutina Ava_10

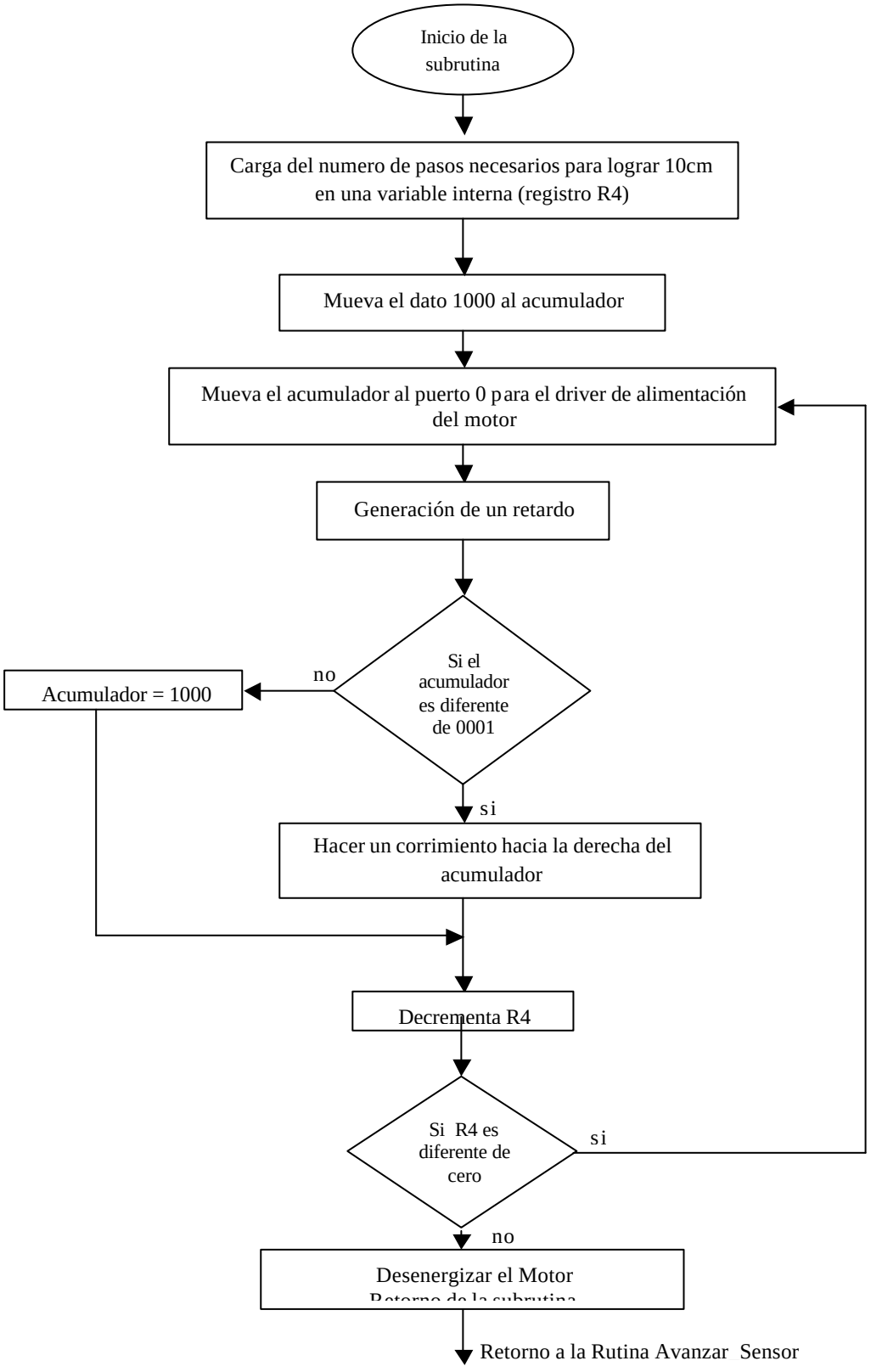
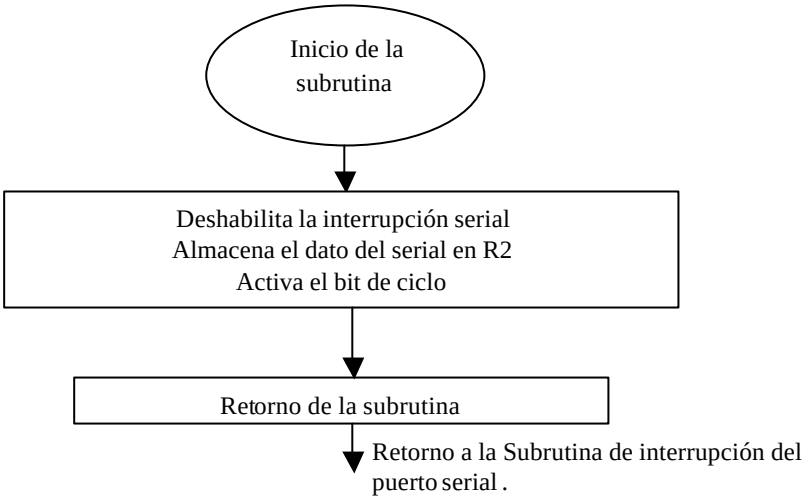


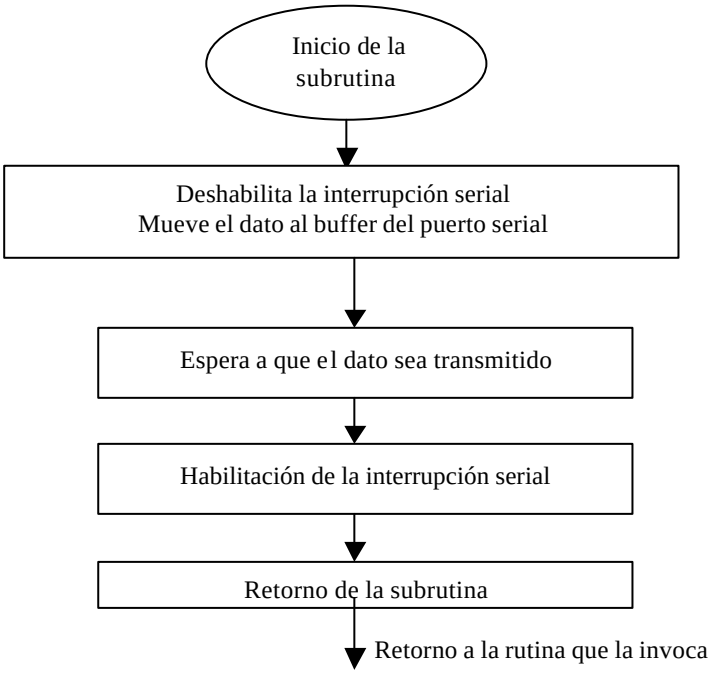
Figura 16. Diagrama de flujo de la subrutina Leer_Serial



6.12 SUBRUTINA ESCRITURA DEL PUERTO SERIAL

Esta subrutina es invocada cada vez que se pretende escribir un dato en el puerto serial.

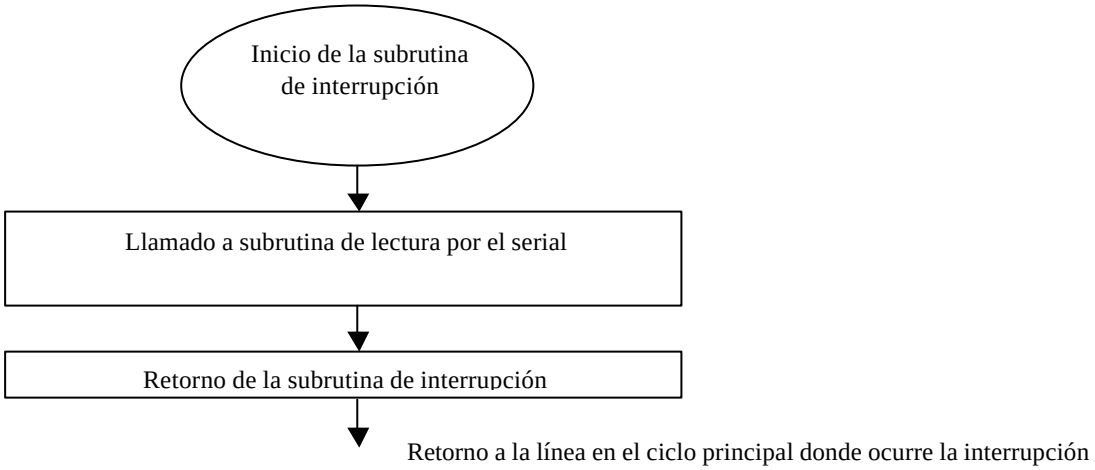
Figura 17. Diagrama de flujo de la subrutina Escribir_Serial



6.13 SUBRUTINA INTERRUPCIÓN SERIAL

En esta subrutina se invoca la subrutina de lectura del puerto serial.

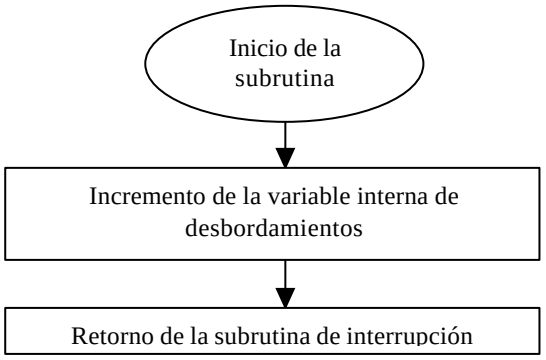
Figura 18. Diagrama de flujo de la subrutina Interrupción Serial



6.14 SUBRUTINA INTERRUPCIÓN TIMER

En esta subrutina se incrementa una variable interna que indica el numero de desbordamiento del timer. Es útil únicamente para obtener los tiempos ?t y ?T del fenómeno físico en cuestión.

Figura 19. Diagrama de flujo de la subrutina Interrupción Timer



7. DESCRIPCIÓN DEL MÓDULO S

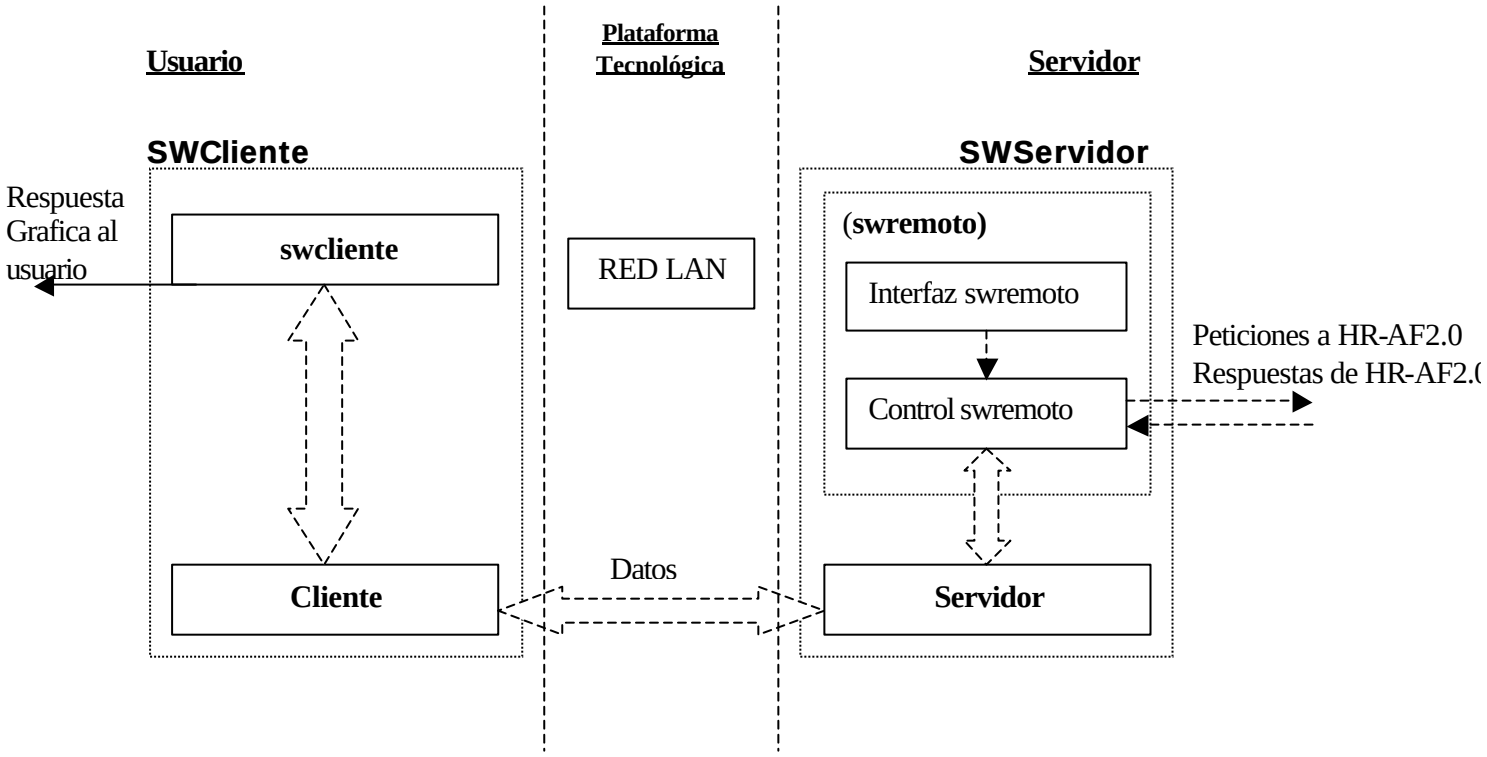
OFTWARE REMOTO: SR-AF2.0

En términos generales, el software remoto del sistema *AutoFisica 2.0* comprende la ultima capa funcional del sistema la cual se implementa bajo los criterios definidos en el modelo OSI con sus especificaciones para las capas de Red, Transporte y Aplicación. Todo esto con el soporte que brinda el lenguaje de programación JAVA para el manejo y desarrollo de aplicaciones en RED como se vera mas adelante.

Básicamente SR-AF2.0 se divide en dos módulos principales conocidos como *SWCliente* y *SWServidor*. Estos dos módulos son los encargados de realizar todas las labores de acceso, comunicación, transmisión de datos, análisis y gestión de dispositivos entre otras que se explicarán mas adelante.

Un primer diagrama general de SR-AF2.0 es el que se muestra a continuación:

Figura 20. Diagrama general de SR-AF2.0



Como se observa en el diagrama, se definen tres lugares de acción como son el lado del usuario, la red o plataforma tecnológica y el lado del servidor. El módulo *SWCliente* pertenece al lado del usuario y el módulo *SWServidor* pertenece al lado del Servidor. No se tocará en detalle el lado de la red pues la plataforma tecnología utilizada es la red LAN de la universidad, la cual esta perfectamente definida y fuera de nuestro alcance para realizar en ella alguna modificación o adaptación

según nuestros requerimientos. Simplemente hace parte del modelo general mas no juega un papel trascendental en el diseño del modelo Software del proyecto, debido a la transparencia que representa la plataforma tecnológica para el desarrollador.

Como se muestra en el diagrama general del modelo software, el módulo SWCliente se compone de dos submódulos que son swcliente y Cliente. En términos generales, el módulo SWCliente transmite peticiones al módulo SWRemoto y recibe respuestas del mismo a través de la plataforma tecnológica expuesta y de esta forma presentar gráficamente los resultados del proceso al usuario.

El módulo SWServidor se compone a su vez de dos submódulos principales que son **swremoto** y **Servidor**. En términos generales, este módulo recibe peticiones del módulo SWCliente y envía respuestas a dicho módulo. También se encarga de la gestión de la comunicación con el hardware remoto (HR-AF2.0) enviando peticiones al hardware y recibiendo respuesta del mismo.

Esto hace que el modelo expuesto, corresponda a un modelo Cliente_Servidor basado en un esquema de Peticion_Respuesta en todos

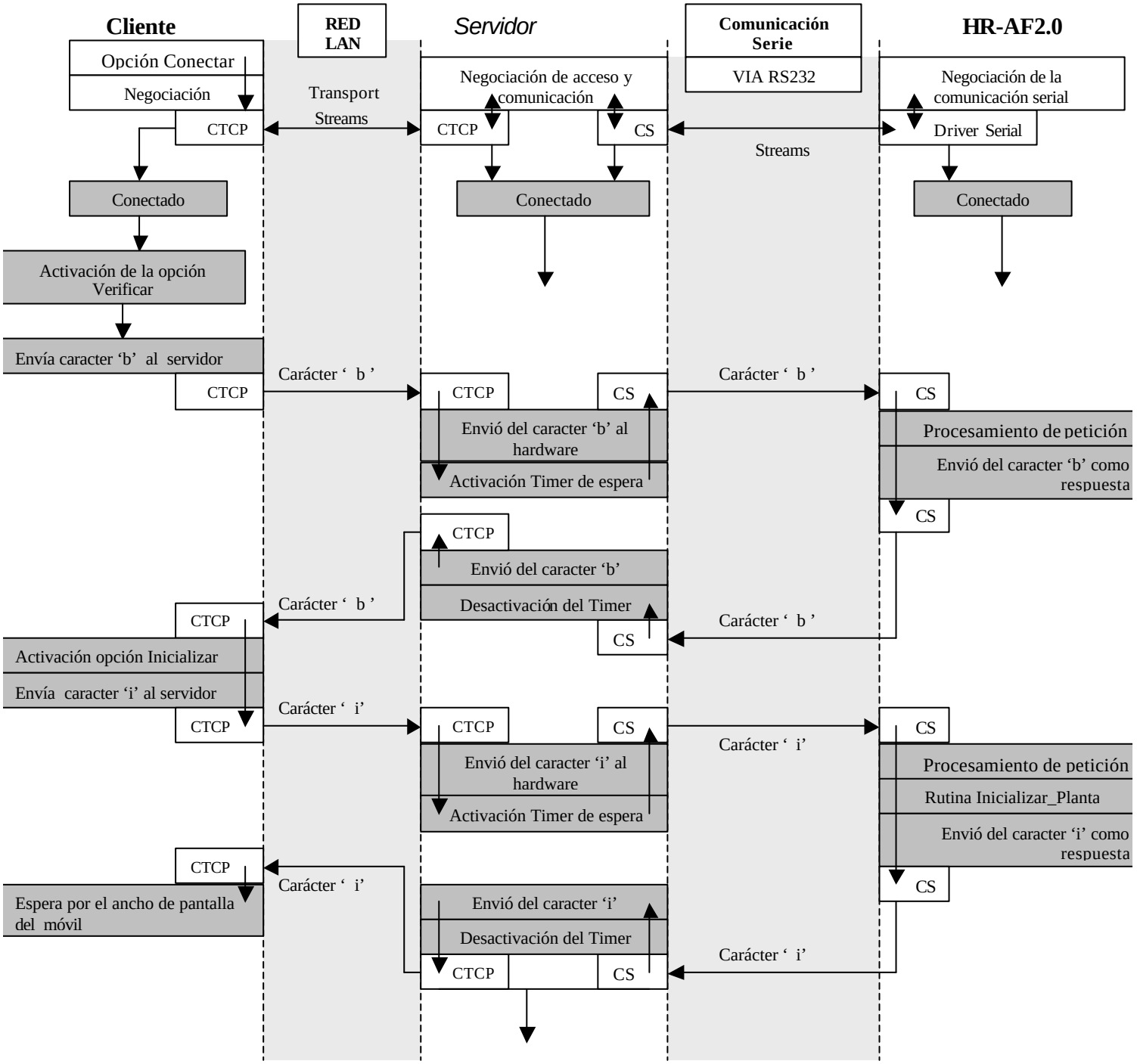
sus niveles, tanto a alto nivel como a bajo nivel. Con esto, se garantiza la secuencialidad de eventos y por tanto se puede decir que el modelo general establece de manera implícita un control total sobre las acciones, disminuyendo así la probabilidad de error en la comunicación, en el esquema funcional y en el mecanismo de comunicación utilizado para nuestros fines.

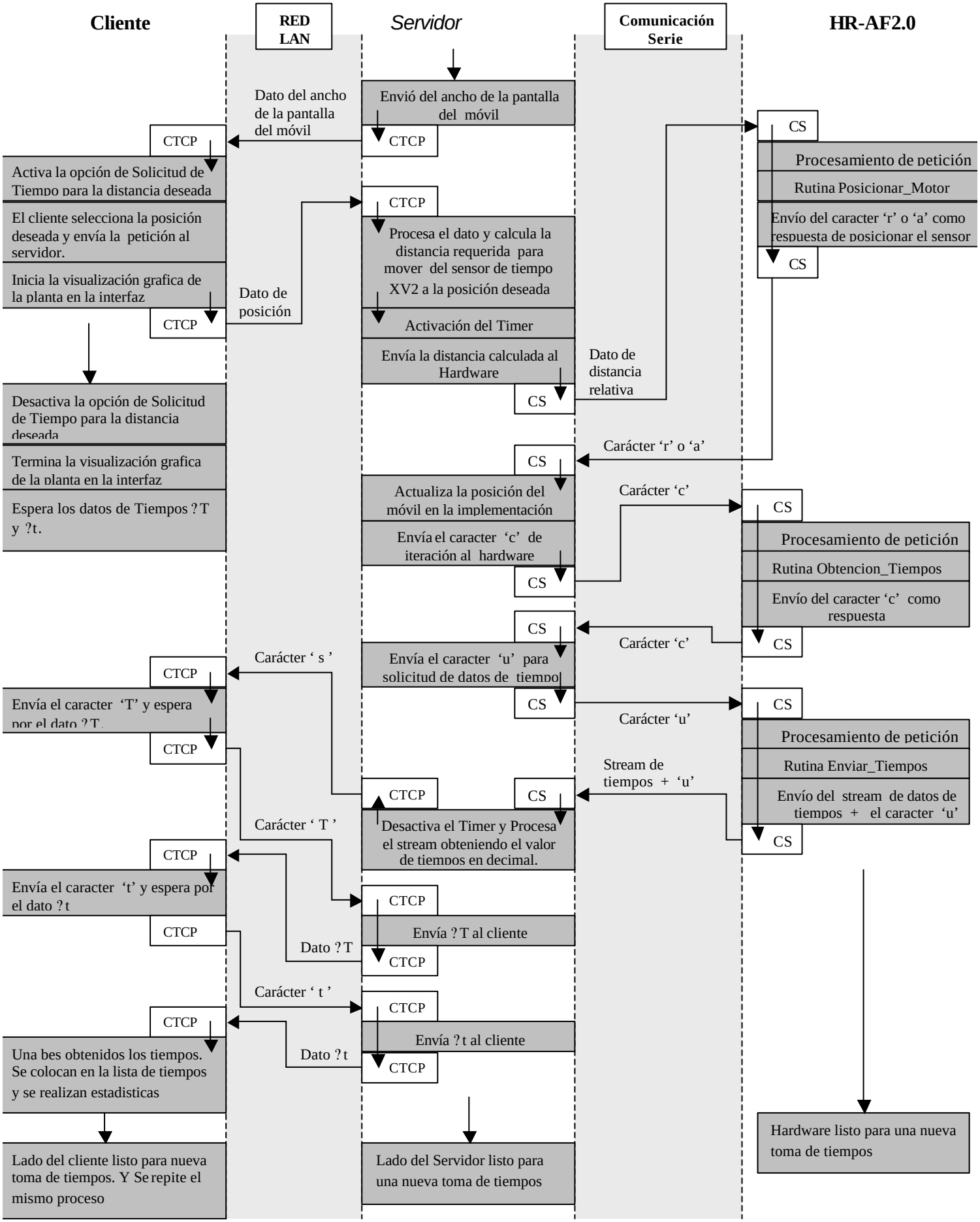
Para entender mas claramente esta situación, explicaremos en detalle el modelo funcional de comunicación entre el cliente, el servidor y el hardware antes de seguir con la especificación detallada de los módulos y componentes del modelo del Software Remoto y su correspondiente implementación.

8. DESCRIPCIÓN DEL MODELO FUNCIONAL DE COMUNICACIÓN DEL SISTEMA AUTOFÍSICA2.0

El Modelo Funcional de Comunicación se fundamenta en un mecanismo de comunicación establecido para cumplir con nuestros requerimientos del problema, proporcionando un control implícito sobre las acciones a realizar y las interacciones entre el usuario y la planta. Todo esto, basado en un modelo *Cliente_Servidor* bajo un esquema de *Peticion_Respuesta Orientado a Objetos* soportado en eventos del sistema.

Figura 21. Diagrama general de comunicación. Continua en la siguiente




Cliente
RED LAN
Servidor
Comunicación Serie
HR-AF2.0
Envió del ancho de la pantalla del móvil
Dato del ancho de la pantalla del móvil
CTCP
CS
Procesamiento de petición
Rutina Posicionar_Motor
Envío del caracter 'r' o 'a' como respuesta de posicionar el sensor
Activa la opción de Solicitud de Tiempo para la distancia deseada
El cliente selecciona la posición deseada y envía la petición al servidor.
Inicia la visualización grafica de la planta en la interfaz
Dato de posición
Procesa el dato y calcula la distancia requerida para mover del sensor de tiempo XV2 a la posición deseada
Activación del Timer
Envía la distancia calculada al Hardware
Dato de distancia relativa
Desactiva la opción de Solicitud de Tiempo para la distancia deseada
Termina la visualización grafica de la planta en la interfaz
Carácter 'r' o 'a'
Actualiza la posición del móvil en la implementación
Carácter 'c'
Envía el caracter 'c' de iteración al hardware
Rutina Obtencion_Tiempos
Envío del caracter 'c' como respuesta
Espera los datos de Tiempos ?T y ?t.
Carácter ' s '
Envía el caracter 'u' para solicitud de datos de tiempo
Envía el caracter 'T' y espera por el dato ?T
Carácter 'u'
Rutina Enviar_Tiempos
Stream de tiempos + 'u'
Envío del stream de datos de tiempos + el caracter 'u'
Carácter ' T '
Desactiva el Timer y Procesa el stream obteniendo el valor de tiempos en decimal.
Envía el caracter 't' y espera por el dato ?t
Envía ?T al cliente
Dato ?T
Carácter ' t '
Envía ?t al cliente
Dato ?t
Una bes obtenidos los tiempos. Se colocan en la lista de tiempos y se realizan estadisticas
Hardware listo para una nueva toma de tiempos
Lado del cliente listo para nueva toma de tiempos. Y Se repite el mismo proceso
Lado del Servidor listo para una nueva toma de tiempos

Este modelo se establece como una representación abstracta del mecanismo de comunicación que se ha implementado para el sistema AutoFísica2.0. No se realiza una especificación de módulos dentro de este modelo, solo pretende describir con claridad la forma en que se comunican el Cliente, el Servidor y el Hardware Remoto. La especificación mas detallada de los módulos y submódulos del Software remoto se continuará mas adelante en una sección posterior.

Como se observa en el diagrama anterior, el Modelo de Comunicación se inicia con la negociación del enlace y conexión por parte del Cliente y el Servidor. Es necesario también que exista una conexión y una comunicación entre el Servidor y HR-AF2.0 para poder acceder al hardware e interactuar con la planta de manera remota. De lo contrario si se intenta establecer algún nivel de interacción con la planta sin que exista comunicación previa entre el Servidor y HR-AF2.0, se producirá un caso de Excepción o Error.

La comunicación entre el Cliente y el Servidor se establece por un canal TCP que hemos llamado ***CTCP*** en el diagrama. Se ha realizado toda la implementación sobre este canal, debido a que este proporciona el soporte adecuado para una comunicación Punto a Punto y de la forma

más segura sobre la capa de Transporte del modelo OSI. De esta manera reducimos la probabilidad de error en el envío y recepción de la información al utilizar un protocolo *Orientado a Conexión*.

Así mismo la comunicación entre el Servidor y el Hardware Remoto se establece por medio de un canal serial bajo la especificación RS232 que hemos llamado *CS* en el diagrama. Tanto el Canal TCP (*CTCP*) como el Canal Serial (*CS*) se encuentran implementados a manera de Objetos con tecnología JAVA.

Los objetos *CTCP* se encargan básicamente de establecer el canal de comunicación y de realizar la negociación de la comunicación, así como el envió y recepción de datos a través de la red por medio de una conexión TCP y un puerto especifico. Estos elementos corresponden al submódulo Servidor en el módulo SWServidor y al submódulo Cliente del módulo SWCliente, ambos especificados anteriormente de manera general. Al nivel de implementación estos dos submódulos se encuentran representados por Objetos (Clases) cuyos nombres son Cliente para el lado del usuario y Servidor para el lado del servidor.

Se ha definido la utilización de caracteres de señalización para establecer una comunicación entre el cliente, el servidor y el hardware remoto. Se ha realizado de esta manera para efectos de simplicidad de la comunicación, dado que esta forma sencilla de señalización permite establecer un modelo de comunicación estable para ejercer un control de acciones basado en eventos, además que cumple con nuestros requerimientos sin la necesidad de utilizar complejos S*treams* de datos para que el cliente interactúe con el servidor y a su vez el servidor con el hardware.

Una vez hecha la negociación y establecido el canal de comunicación entre el Cliente y el Servidor y entre el Servidor y HR-AF2.0, el sistema se encuentra listo para el envío de peticiones y recepción de respuestas.

De esta manera el Cliente ya tiene la posibilidad de ejecutar la acción de verificación de *Plata_Lista* mediante la activación de la opción Verificar de la interfaz de usuario.

Una vez el cliente ejecute la acción de verificación, se envía el caracter *'b'* mediante el canal CTCP del cliente representado a su vez por el objeto *Cliente* en la implementación. Esta información se transmite

hacia el servidor mediante la Red LAN y este recibe esta información mediante el canal CTCP del servidor representado por su correspondiente objeto *Servidor* en la implementación. Con esta información el servidor únicamente transfiere este caracter al hardware remoto mediante la comunicación serial establecida previamente y activa un elemento Timer para evitar cualquier caso de error si llegase a presentarse una falla física en la planta. De esta manera se avisa al usuario y al auxiliar de laboratorio que ha ocurrido una falla cuando el timer cumple su tiempo de activación.

Una vez el hardware obtenga esta información, simplemente devuelve el caracter *'b'* al servidor y a su vez este desactiva el timer si no ha cumplido con su tiempo de activación y reenvía el caracter *'b'* como respuesta al cliente. Cuando el cliente obtiene la respuesta realiza una acción especifica, que en este caso será activar la siguiente opción en la interfaz de usuario que habilita al mismo para ejecutar la acción de inicialización de la planta.

De esta manera se ha realizado una primera acción bajo un esquema de Petición_Respuesta. El resto de las interacciones resultan funcionalmente muy similares.

Una vez el usuario ejecuta la acción de inicialización de la planta mediante la opción Inicializar de la interfaz de usuario, el cliente envía el carácter *'i'* al servidor, este ultimo lo envía al hardware e inicializa de nuevo el Timer. Una vez el hardware ha recibido esta información, procede a ejecutar una de las rutinas definidas para el microcontrolador que gobierna todo el sistema electrónico, esta rutina es conocida como *Rutina Inicializar_Planta*,* la cual se encarga de ubicar todo el instrumental de la planta en los puntos iniciales. Una vez ejecutada con éxito la rutina invocada, el sistema hardware retorna al servidor el carácter *' i '* y este a su vez al recibir esta información proveniente del hardware desactiva el Timer y envía el carácter *'i'* al cliente y procede a enviar el dato correspondiente al valor decimal del tamaño de la pantalla del móvil. Cuando el cliente obtiene el carácter *'i'* en la entrada de su canal CTCP procede a esperar por el dato adicional que representa el ancho de la pantalla del móvil. Dato utilizado para realizar estadísticas en tiempo de ejecución como son velocidad instantánea del móvil entre otras. Cuando el cliente ya ha obtenido esta información procede a activar la opción de Tiempos en la interfaz de usuario, la cual habilita al mismo para hacer la petición de toma de tiempos y obtención de los valores decimales de dichos tiempos.

Una vez el usuario ejecute la acción de obtener los tiempos del fenómeno mediante la opción Tiempos en la interfaz de usuario, el cliente procede a enviar al servidor el dato de posición deseada para el sensor de tiempos XV2 obtenido de la lista de posiciones de la interfaz. A su vez en la interfaz de usuario se procede a generar una salida grafica de la ubicación del sensor y del movimiento del móvil en la planta. En este momento el cliente ya solo le resta esperar a que el servidor le devuelva los datos de tiempos. Mientras tanto la opción de Tiempos en la interfaz permanece deshabilitada.

El servidor recibe esta información y procesa dicho dato, calculando así la distancia requerida por el hardware para ubicar el sensor en la posición deseada, activa el Timer del servidor y envía un dato al hardware que representa la distancia requerida para posicionar el sensor. El hardware toma esta información y ejecuta una rutina llamada *Rutina Posicionar_Motor**, una vez ejecutada esta rutina devuelve al servidor el carácter *'r'* o *'a'* según sea el caso de avanzar retroceder el sensor. De cualquier forma, el servidor toma esta información que le indica que el sensor ya ha sido ubicado en la posición deseada y envía el carácter *'c'* al hardware remoto.

* Para observar la descripción de cada rutina referirse a la explicación expuesta en la Implementación del algoritmo para el módulo de Control de la interfaz IE-AF2.0

Este carácter le indica al algoritmo de control del hardware que debe ejecutar la rutina de obtención de tiempos llamada *Rutina Obtencion_Tiempos.* Una vez ejecutada esta rutina con éxito el hardware procede a enviar el carácter *'c'* al servidor y a su vez este toma esta información y envía el caracter *'u'* al hardware. Este carácter le indica al hardware que debe transmitirle los datos de tiempo del fenómeno al servidor mediante su rutina *Rutina Enviar_Tiempos*, la cual envía un Stream de datos al servidor finalizando con el carácter *'u'* dicho Stream. Nótese que todavía sigue activo el Timer del servidor.

Una vez el servidor recibe todo el Stream, desactiva el Timer si no ha cumplido su tiempo de activación y calcula los valores decimales de ?T y ?t en base al Stream enviado por el hardware. Con los valores de tiempos ya definido, el servidor envía el carácter *'s'* al cliente que le indica que esta listo para enviarle los tiempos ?T y ?t. Una vez el cliente recibe el carácter *'s'* le solicita al servidor que le envíe en su orden los datos ?T y ?t mediante las peticiones *'T'* y *'t*. Cuando el servidor recibe en su orden estos caracteres le envía al cliente los

tiempos y el cliente a su vez coloca dichos tiempos en la Lista de Tiempos de la interfaz de usuario.

De esta manera tanto el cliente como el servidor y el hardware remoto quedan de nuevo listos y a la espera para una siguiente iteración y obtención de nuevos tiempos para otra posición del sensor dentro del rango de estudio del fenómeno.

Esta es la forma en que se ha establecido el modelo de comunicación entre estos agentes. Posteriormente describiremos en mas detalle la especificación modular presentada con anterioridad y la implementación de este modelo.

9. DESCRIPCIÓN FUNCIONAL DE LOS MÓDULOS DEL SOFTWARE REMOTO (SR-AF2.0).

Como se mencionaba anteriormente, el modelo funcional del software esta compuesto de dos módulos principales que hemos llamado *SWCliente* y *SWServidor*. A continuación explicaremos en detalle la función de cada módulo y submódulo del sistema para posteriormente observar la especificación de la implementación del sistema a manera de objetos.

9.1 MÓDULO SWCLIENTE

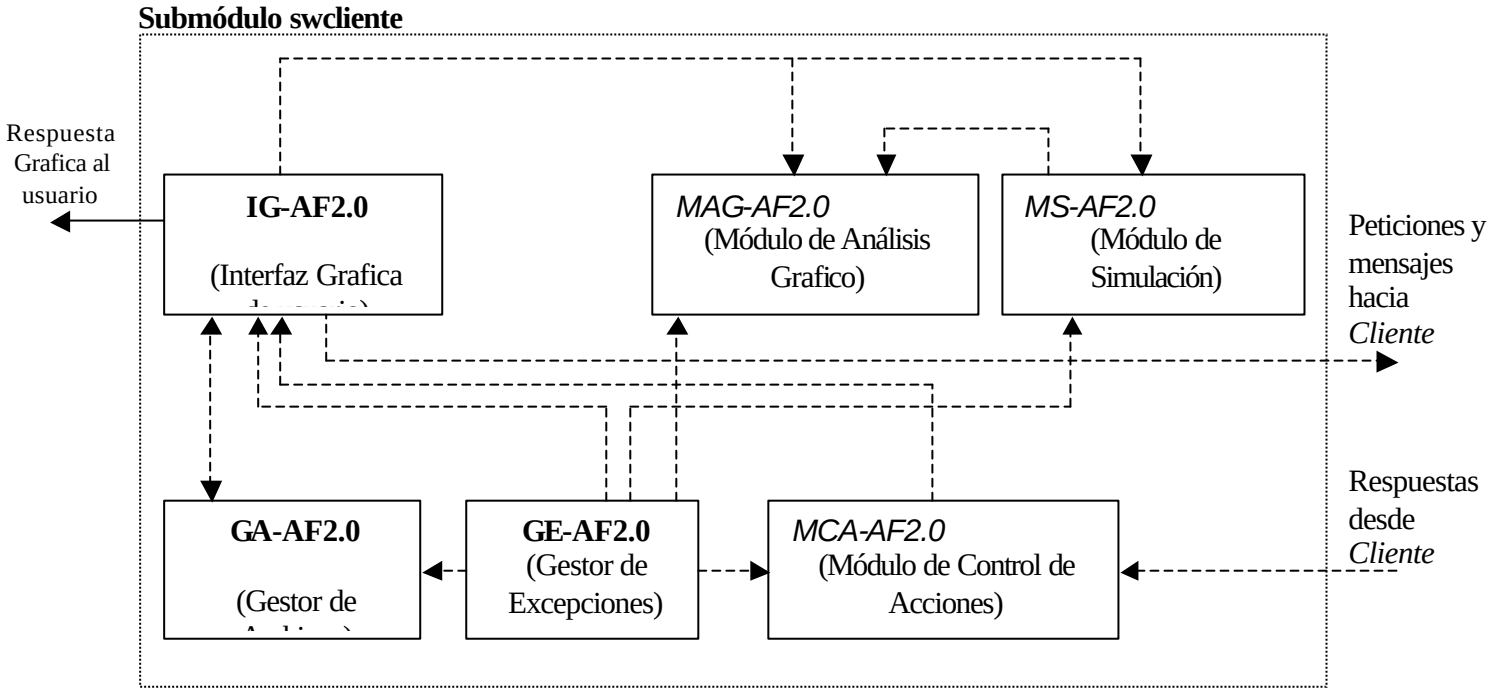
Este módulo, se encarga básicamente de establecer una comunicación con el módulo SWServidor para el lado del usuario mediante un canal de transporte de datos seguro a través de la red establecida, además de realizar las peticiones, obtener las respuestas, procesar los datos provenientes de SWServidor, realizar análisis y estadísticas del fenómeno y finalmente presentar al usuario los resultados del proceso de manera gráfica mediante el uso de simulaciones y gráficas interactivas.

El módulo SWCliente consta principalmente de dos Submódulos que hemos llamado *swcliente* y *Cliente*. Ambos submódulos se comunican entre si, transfiriéndose información de peticiones y respuestas. Las peticiones son enviadas desde *swcliente* a *Cliente* y las respuestas desde *Cliente* a *swcliente*. Una vez el submódulo *Cliente* recibe una petición este se encarga de enviarla por el correspondiente canal de transporte de datos hacia el módulo SWServidor.

9.1.1 Submódulo swcliente: Este submódulo se encarga básicamente de labores como control de acciones de interfaz de usuario, procesamiento y análisis de datos provenientes de SWServidor, envío de peticiones, gestión de archivos, generación de respuestas graficas, Simulaciones y finalmente Análisis gráficos de datos.

Observemos el siguiente diagrama funcional de este submódulo:

Figura 22. Diagrama del submódulo swcliente.



Este esquema representa de manera grafica las Características Funcionales del submódulo *swcliente*. Es necesario aclarar que dicho esquema no representa la implementación a manera de objetos, sino el comportamiento Funcional de *swcliente* en términos de módulos funcionales. Como se vera mas adelante la implementación de varios módulos funcionales puede encontrarse en un mismo objeto.

Submódulo IG-AF2.0 (Módulo de Interfaz Grafica): Este módulo representa todas las acciones y el comportamiento funcional de la interfaz grafica del usuario, por medio de la cual se presentan gráficamente las diferentes opciones que habilitan al mismo para acceder remotamente a la planta y visualizar gráficamente el procesamiento y análisis de los datos correspondientes al fenómeno físico de estudio. Es mediante esta interfaz, representada en este modelo por el módulo funcional IG-AF2.0, que el usuario puede acceder a opciones de conexión remota, realizar peticiones al Servidor, hacer Gestión de archivos, Gestión de Simulaciones, Gestión de Análisis Grafico, Visualización de Errores, Listado de datos, envío de mensajes de texto al servidor y Visualización de Estadísticas propias del fenómeno como son Velocidades y Tiempos.

Por otro lado, el módulo MCA-AF2.0 indica a IG-AF2.0 en que momento actualizar la información que muestra gráficamente. Esta acción se realiza cada vez que un dato del fenómeno es entregado a *swcliente*. Como consecuencia de esto, el usuario vera gráficamente un efecto actualización de estadísticas en la interfaz grafica por cada interacción con la planta.

Submódulo MAG-AF2.0 (Módulo de Análisis Grafico): Este módulo representa el comportamiento gráfico de la información y de las estadísticas realizadas. Dicho módulo funcional es el encargado de realizar todas las operaciones matemáticas para mostrar gráficamente los resultados del análisis de los datos del fenómeno físico. Se podría decir entonces que este módulo contiene un gestor de estadísticas para el análisis y procesamiento de los datos. Así, el módulo realiza la gestión de interpolaciones lineales y cuadráticas a partir de los datos de tiempos y distancias para obtener los parámetros y coeficientes de las ecuaciones del fenómeno de estudio. Una vez realizado este procesamiento, el módulo se encarga de mostrar de manera grafica e interactiva los resultados del análisis mediante un plano de coordenadas cartesianas.

Submódulo MS-AF2.0 (Módulo de Simulación): Este módulo funcional representa el mecanismo de simulación realizado para generar datos teóricos de tiempos y distancias a partir de una simulación del fenómeno físico sin interacción directa con la planta.

De esta manera el usuario tiene la posibilidad de realizar ejercicios de contrastación entre la información teórica generada por al simulación y la información real del fenómeno generada por la planta.

Funciona de manera muy similar al módulo IG-AF2.0, solo que en lugar de opciones de interacción remota con la planta, soporta acciones simuladas como ingreso de datos iniciales de velocidad, aceleración, incertidumbres, así como listado de tiempos y graficas de los datos obtenidos durante la simulación. En este ultimo caso, el módulo MS-AF2.0 interactúa con el módulo MAG-AF2.0 transfiriéndole información teórica para que este realice los correspondiente procesos estadísticos y gráficos de la información.

Submódulo GA-AF2.0 (Módulo Gestor de Archivos): Este módulo representa los procedimientos y mecanismos adecuados para la gestión de archivos y sus correspondientes formatos. De esta manera, el usuario tiene la posibilidad de almacenar la información de tiempos y

distancias obtenida tanto de la planta real como de la simulación de fenómeno, en ficheros de extensión propia y en ficheros exportados a extensiones estándar. Así, el usuario cuenta con opciones que le permiten ejecutar acciones como Abrir Archivos, Guardar Archivos y Exportar Archivos. Las opciones de interfaz que acceden a estas funcionalidades se soportan mediante el módulo IG-AF2.0.

Submódulo GE-AF2.0 (Módulo de Gestión de Excepciones): Este módulo representa el mecanismo mediante el cual se manejan todos los posibles casos de error y de excepción que se puedan presentar en tiempo de ejecución del Software para el lado del usuario. De esta manera se capturan y se procesan las excepciones que se puedan producir a causa de acciones no permitidas en la carga y almacenamiento de archivos, acciones de flujo de Streams incorrectas, manejo inadecuado de estructuras de datos, operaciones matemáticas no permitidas, procesamientos estadísticos incorrectos y posibles excepciones en la ejecución incorrecta de hilos.

Es por tanto, altamente significativa la existencia de dicho módulo funcional para fines de captura y procesamiento de errores en tiempo de

ejecución, evitando de esta manera un bloqueo de la aplicación y un posible colapso del sistema del lado del cliente.

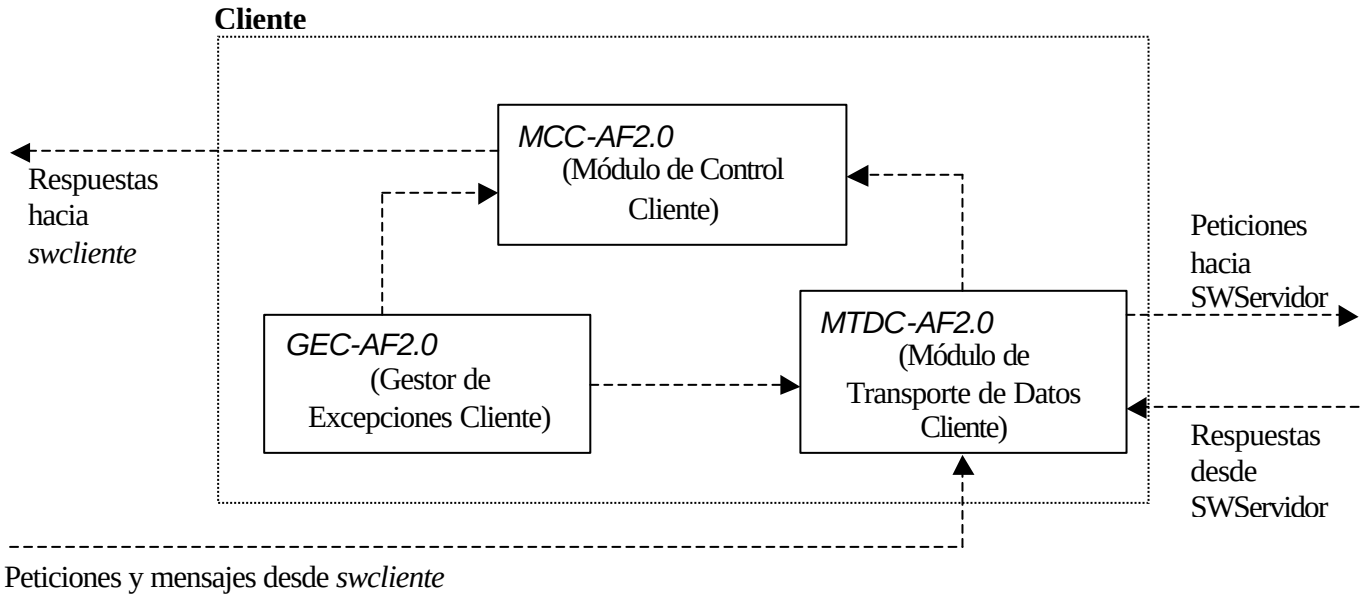
Submódulo MCA_AF2.0 (Módulo de Control de Acciones): Se podría decir que este módulo representa el núcleo funcional del software del lado del usuario. Básicamente este módulo hace referencia al mecanismo de control a alto nivel utilizado para la correcta ejecución de las acciones de interfaz. Este módulo representa un ciclo principal continuo e infinito de encuesta de los datos que provienen del submódulo *Cliente* que a su vez recibe datos del módulo SWServidor a través del canal de transporte de datos.

Una vez el módulo MCA-AF2.0 obtiene la respuesta a una petición remota evalúa dicha respuesta y en función de esta habilita o deshabilita una opción de interfaz para realizar una acción especifica de interacción con la planta. Es así como se ejerce un control de acciones para el usuario mediante la llegada de respuestas desde servidor, que en términos de implementación corresponderán a respuestas de eventos de llegada de datos por el canal de transporte. Este módulo presenta un comportamiento concurrente respecto a los demás submódulos de *swcliente* debido a que su función de encuesta es implementada

mediante un hilo de control. Esto se verá mas adelante en la especificación de la implementación de los módulos del software del sistema.

9.1.2 Submódulo Cliente: Este submódulo del módulo general SWCliente es el encargado fundamentalmente de realizar exitosamente todas las labores de negociación de la conexión a red y establecer de esta manera un canal de transporte de datos seguro para transmitir peticiones del lado del usuario y recibir respuestas del lado del servidor (es decir del módulo SWServidor). Dentro de este módulo se representa funcionalmente el CTCP mencionado en el modelo de comunicación para el lado del usuario. El diagrama de este submódulo se muestra a continuación:

Figura 23. Diagrama del submódulo cliente.



Como se observa en el diagrama, este submódulo consta de tres módulos funcionales como son el Módulo de Control Cliente, el Módulo de Transporte de Datos Cliente y el Módulo de Excepciones Cliente.

MCC-AF2.0 (Módulo de Control Cliente): Este módulo representa el mecanismo de control mediante el cual se transmite la información hacia *swcliente* en respuesta a un evento de llegada de información por el canal de transporte de datos establecido mediante el módulo MTDC-AF2.0. Este mecanismo corresponde a un ciclo continuo e infinito en el cual se espera la llegada de un dato por el canal de transporte. Una vez llega un dato por el canal se responde a este evento enviando dicha información al Módulo de Control de Acciones de *swcliente* (MCA_AF2.0) para que este módulo responda adecuadamente y realice sus correspondiente funciones y control de acciones.

Este módulo presenta un comportamiento concurrente respecto a los demás módulo de *Cliente* debido a que su función de espera y respuesta a eventos de llegada de datos es implementada mediante un hilo de control. Esto se verá mas adelante en la especificación de la implementación de los módulos del software remoto.

GEC-AF2.0 (Módulo de Gestión de Excepciones Cliente): Este módulo representa el mecanismo mediante el cual se realiza la gestión y el correcto manejo de todos los posibles casos de error y de excepción que se puedan presentar en el tiempo de ejecución de la implementación del módulo *Cliente.*

De esta manera se capturan y se procesan las excepciones que se puedan producir a causa de negociaciones incorrectas de la comunicación con el lado servidor (módulo SWServidor), no establecimiento del canal de transporte de datos, rompimiento de la conexión, problemas de Streams de datos y flujos de información entre *Cliente y* SWServidor, problemas en la ejecución del hilo de control etc.

Es por tanto, altamente significativa la existencia de dicho módulo funcional para fines de captura y procesamiento de errores en tiempo de ejecución, evitando de esta manera un bloqueo de la aplicación y un posible colapso del sistema del lado del cliente.

MTDC-AF2.0 (Módulo de Transporte de Datos Cliente): Este módulo es de vital importancia para el esquema funcional del sistema, debido a

que representa el mecanismo para establecer un canal de transporte de datos seguro para el lado del cliente (CTCP) y de esta manera poder enviar información hacia el lado del servidor y obtener las respuestas del mismo.

Básicamente el módulo es una representación funcional de cómo se establecer un canal final de comunicación bajo el concepto de Sockets y Puertos sobre la capa de transporte de datos del modelo OSI. Se ha realizado de esta manera por cuestiones de seguridad en el envió y recepción de la información, en vez de utilizar los muy conocidos datagramas bajo protocolo UDP. Este módulo tiene estrecha relación con el módulo MCC-AF2.0 debido a que es mediante este mecanismo de conexión que se responde a los eventos de llegada de información en el canal final de comunicación (socket) y una vez este módulo responda a este evento le transfiere la información al Módulo de Control Cliente para que este en su ciclo realice la acción correspondiente al valor de llegada por el canal. Es importante mencionar, que es a través de este mecanismo que el submódulo *swcliente* envía las peticiones a SWServidor y es a través de este mecanismo que todo el módulo SWCliente recibe los datos desde SWServidor.

9.2 MÓDULO SWSERVIDOR

Este módulo, se encarga básicamente de establecer una comunicación con el módulo SWCliente para el lado del servidor mediante un canal de transporte de datos seguro a través de la red establecida. Además se encarga de establecer un canal serial de comunicación con el hardware remoto del sistema, con el fin de acceder directamente a toda la configuración instrumental diseñada para interactuar con la planta. También se encarga de realizar un control secuencial de acciones con el hardware evitando así posibles casos de error en el proceso de interactividad remota con la planta. Proporciona un soporte de interfaz para el auxiliar de laboratorio de tal manera que este pueda realizar la configuración previa de la comunicación y actuar bajo posibles casos de falla física en la planta.

Así, en términos generales, el módulo SWServidor se encarga de:

- Establecer un canal de transporte de información para recibir peticiones de usuario y enviar respuestas al mismo a través de la red LAN .

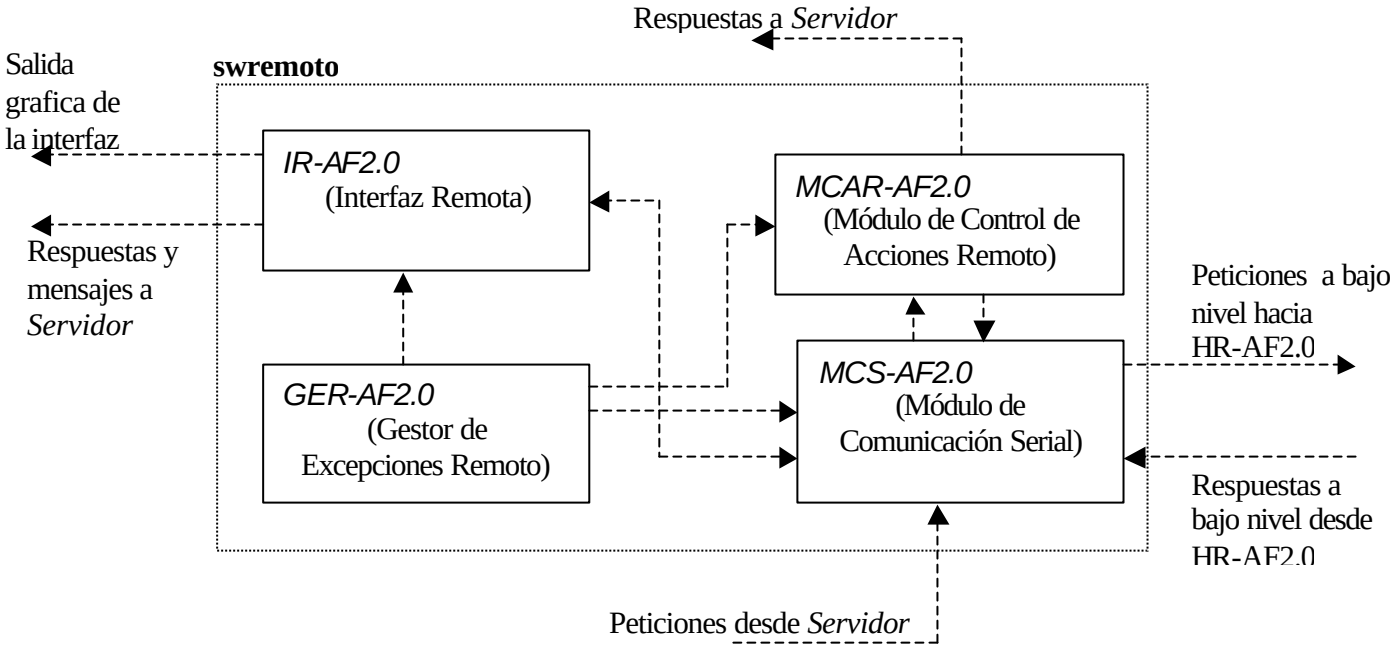
- Establecer un canal de comunicación serial basado en la especificación RS232 con el hardware remoto con el fin de transmitir las peticiones del usuario al hardware, y recibir las respuestas de dicho hardware, el cual realizará el procesamiento de dicha petición a bajo nivel.
- Presentar un mecanismo de control secuencial concordante con el Modelo de Comunicación establecido.
- Finalmente, establecer una interfaz para el auxiliar de laboratorio que permita indicar posibles fallas físicas en la planta.

Este módulo software que se encuentra en el lado del servidor y consta principalmente de dos submódulos que hemos llamado swremoto y Servidor.

9.2.1 Submódulo swremoto: Este submódulo es el encargado de realizar las tres ultimas funciones de la lista general de funciones indicada anteriormente para SWServidor.

A continuación presentamos un esquema de este submódulo.

Figura 24. Diagrama del submódulo swremoto.



Como se observa en el diagrama, existen cuatro módulos principales que se distribuyen las funciones entre si. Estos módulos son: Un módulo de Interfaz Remota(IR-AF2.0), un módulo de Control de Acciones Remoto(MCAR-AF2.0), un módulo de Gestión de Excepciones Remoto (GER-AF2.0) y un módulo de Comunicaciones Serial(MCS-AF2.0).

IR-AF2.0 (Módulo de Interfaz remota): Este módulo representa todas las acciones y el comportamiento funcional de la interfaz grafica establecida para el lado del servidor, por medio de la cual se presentan

gráficamente las diferentes opciones que habilitan al auxiliar para realizar acciones propias de configuración.

Permite la visualización de la información que el hardware le transfiere a SWServidor mediante cuadros de texto en pantalla. También permite el envío de mensajes y respuestas no automáticas al submódulo *Servidor* que a su vez envía información hacia lado del cliente por medio de opciones manuales que se muestran en la interfaz. Así mismo presenta la opción de configuración de la comunicación serie, envío de datos manualmente hacia el hardware remoto y recepción de datos desde el hardware remoto. Es por ello que este módulo se interrelaciona con el módulo MCS-AF2.0 de manera bidireccional.

Por medio de este módulo se generan avisos gráficos y auditivos al auxiliar de laboratorio cuando se presenta una posible falla física en la planta.

GER-AF2.0 (Módulo de Gestión de Excepciones Remoto): Este módulo representa el mecanismo mediante el cual se manejan todos los posibles casos de error y de excepción que se puedan presentar en tiempo de ejecución del Software para el lado del servidor. De esta manera se

capturan y se procesan las excepciones que se puedan producir a causa de negociación de la comunicación serial fallida, negociación de la conexión a la red fallida, posible ruptura del canal de transporte de datos para el lado del servidor, posible ruptura del canal de comunicación serial con el hardware, ejecución indebida de hilos de control etc. Este módulo se comunica con todo los demás módulos del esquema para indicar que alguna Excepción ha ocurrido durante la ejecución de sus procesos.

Es por tanto, altamente significativa la existencia de dicho módulo funcional para fines de captura y procesamiento de errores en tiempo de ejecución, evitando de esta manera un bloqueo de la aplicación y un posible colapso del sistema del lado del servidor.

MCS-AF2.0 (Módulo de Comunicación Serial): Este módulo es el encargado de establecer el canal de comunicación serial con el hardware remoto mediante acciones de configuración del canal. Para ello realiza toda la negociación de la conexión serial. También establece un grado de transparencia significativamente alto en la utilización de Streams de datos seriales para el procesamiento de la información proveniente del

hardware. De esta manera el manejo y la gestión de los datos seriales se realiza a alto nivel y sin mayores problemas.

Realiza funciones de envío de información al hardware mediante sus mecanismos específicos de escritura en el canal serial y recibe información del hardware a manera de eventos en el canal serial establecido. Esto hace muy sencilla la utilización de este módulo en términos de implementación. Se interrelaciona con el módulo de Interfaz Remoto enviando y recibiendo datos del mismo y con el Módulo de Control de Acciones Remoto de la misma forma.

MCAR-AF2.0 (Módulo de Control de Acciones Remoto): Este módulo representa el núcleo funcional de *swremoto*. Básicamente es el encargado de ejecutar el ciclo principal de control para la correcta interacción remota con la planta. Este ciclo es continuo e infinito por lo cual se hace necesario la utilización de un hilo de control para este módulo. De esta manera siempre se encuentra a la espera de la llegada de un dato por el canal serial de comunicación con el hardware a través del Módulo de Comunicación Serial(MCS-AF2.0). Una vez llega un dato por medio del canal serial desde la planta, el módulo MCAR-AF2.0, toma esta información y la procesa ejecutando una acción especifica como

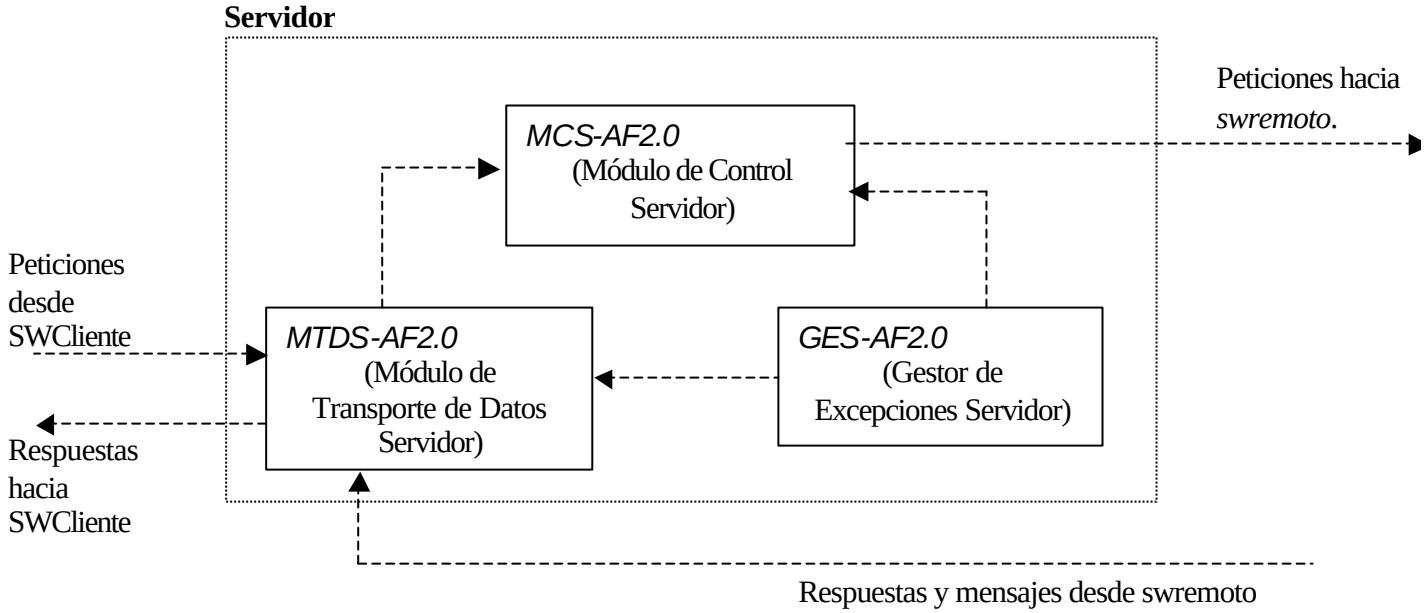
puede ser, el envío de una respuesta al lado del cliente o una orden de control para la planta.

Por tanto se puede decir que este módulo no solo ejerce un control secuencial de la planta y envía respuestas al lado del cliente por medio del submódulo *Servidor* del módulo general SWServidor, sino que además realiza funciones internas de escritura de datos en el canal serial para efectos de cumplimiento de ordenes de control de planta, captura de los Streams de datos seriales provenientes del módulo MCS-AF2.0, calculo de distancias relativas para el posicionamiento del sensor de tiempos, obtención de los datos de tiempos del fenómeno y calculo interno del correspondiente valor decimal de dichos tiempos.

9.2.2 Submódulo servidor: Este submódulo del módulo SWServidor es el encargado fundamentalmente de realizar exitosamente todas las labores de negociación de la conexión a red y establecer de esta manera un canal de transporte de datos seguro para recibir peticiones del lado del usuario y transmitir respuestas del lado del servidor (es decir del módulo SWCliente). Dentro de este módulo se representa funcionalmente el CTCP mencionado en el modelo de comunicación para el lado del servidor.

El diagrama de este submódulo se muestra a continuación:

Figura 25. Diagrama del submódulo servidor.



Como se observa en el diagrama, este submódulo consta de tres módulos funcionales como son el Módulo de Control Servidor, el Módulo de Transporte de Datos Servidor y el Módulo de Excepciones Servidor.

MCS-AF2.0 (Módulo de Control Servidor): Este módulo representa el mecanismo de control mediante el cual se transmite la información hacia *swremoto* en respuesta a un evento de llegada de información por el canal de transporte de datos establecido mediante el módulo MTDS-AF2.0. Este mecanismo corresponde a un ciclo continuo e infinito en el

cual se espera la llegada de un dato por el canal de transporte. Una vez llega un dato por el canal se responde a este evento enviando dicha información al Módulo de Control de Acciones Remoto de *swremoto* (MCAR-AF2.0) para que este módulo responda adecuadamente y realice sus correspondiente funciones y envío de ordenes de control a la planta.

Este módulo presenta un comportamiento concurrente respecto a los demás módulo de *Servidor* debido a que su función de espera y respuesta a eventos de llegada de datos es implementada mediante un hilo de control. Esto se verá mas adelante en la especificación de la implementación de los módulos del software remoto.

GES-AF2.0 (Módulo de Gestión de Excepciones Servidor): Este módulo representa el mecanismo mediante el cual se realiza la gestión y el correcto manejo de todos los posibles casos de error y de excepción que se puedan presentar en el tiempo de ejecución de la implementación del módulo *Servidor.*

De esta manera se capturan y se procesan las excepciones que se puedan producir a causa de negociaciones incorrectas de la comunicación con el lado del cliente (módulo SWCliente) , no

establecimiento del canal de transporte de datos, rompimiento de la conexión, problemas de Streams de datos y flujos de información entre *Servidor y* SWCliente , problemas en la ejecución del hilo de control etc.

Es por tanto, altamente significativa la existencia de dicho módulo funcional para fines de captura y procesamiento de errores en tiempo de ejecución, evitando de esta manera un bloqueo de la aplicación y un posible colapso del sistema del lado del servidor.

MTDS-AF2.0 (Módulo de Transporte de Datos Servidor): Este módulo es de vital importancia para el esquema funcional del sistema, debido a que representa el mecanismo para establecer un canal de transporte de datos seguro para el lado del servidor (CTCP) y de esta manera poder enviar información hacia el lado del cliente y recibir peticiones del mismo.

Básicamente el módulo es una representación funcional de cómo se establece un canal final de comunicación bajo el concepto de ServerSockets, Sockets y Puertos sobre la capa de transporte de datos del modelo OSI para el lado de un servidor. Se ha realizado de esta manera por cuestiones de seguridad en el envío y recepción de la

información, en vez de utilizar los muy conocidos datagramas bajo protocolo UDP. Una vez establecida la conexión con el servidor (ServerSocket), este módulo establece una estrecha relación con el módulo MCS-AF2.0 debido a que es mediante este mecanismo de conexión que se responde al evento de llegada de información por el canal final de comunicación (Socket) y una vez se responda a este evento se transfiere la información al Módulo de Control Servidor para que este en su ciclo realice la acción correspondiente al valor de llegada por el canal.

Es importante mencionar, que es a través de este mecanismo o módulo que el submódulo *swremoto* recibe las peticiones del lado del cliente y es a través de este mecanismo que todo el módulo SWServidor recibe los datos desde SWCliente.

De esta manera se ha especificado el comportamiento funcional del software remoto(SR-AF2.0) a manera de módulos funcionales cuya implementación resulta ser un poco menos compleja que este esquema dado que varios módulos del sistemas se pueden implementar en un mismo objeto.

10. RESULTADOS OBTENIDOS

El sistema *Auto Física 2.0* consta esencialmente de dos componentes que se pueden identificar como resultados del proceso de diseño e implementación.

10.1 COMPONENTE HARDWARE

10.1.1 Descripción: Fundamentalmente este componente se divide en dos módulos principales que son el módulo Electrónico de control de planta y la Planta como tal. El primer módulo consta de toda la electrónica necesaria para procesar las peticiones del usuario a bajo nivel y de esta manera ejercer un control sobre la planta mediante algoritmos de control secuencial. El segundo módulo consta de la planta Carril de Aire y de todo el instrumental de sensado y actuación necesario para obtener la información del fenómeno físico y enviarla al módulo Electrónico de control de planta para que este a su vez envié esta información al software final para realizar el correspondiente procesamiento a alto nivel y enviar los datos finales al usuario.

10.1.2 Alcances: Se ha realizado un reacodicionamiento de la planta , mejorando el sistema de sensado y adicionando el sistema de actuación para el disparo del móvil. De esta manera, se ha dotado a la planta con instrumental necesario para un accionamiento remoto. Se ha adicionado un sensor para la calibración e inicialización del punto de referencia de tiempos (posición cero de los sensores de tiempo) y otro para detectar la posición inicial del carro. Así mismo, se mejoro el sistema de sensado óptico para la toma de tiempos, cambiando así, la referencia de los sensores ya existente por una referencia que proporciona la misma precisión pero que a su vez brinda mas versatilidad y comodidad al instrumental en términos de espacio.

Como se menciono, es necesario la colocación de un actuador para el disparo remoto y los correspondientes drivers y acondicionamiento.

10.2 COMPONENTE SOFTWARE

10.2.1 Descripción: La implementación del software permite la manipulación del componente hardware para el sensado y actuación, transmitir información de manera remota a través de una red y llevarla hasta el usuario quien tiene básicamente la opción de conexión,

interacción con la planta real del laboratorio, interacción con una simulación de un modelo de la planta, comparación teoría-práctica y la opción de gráficas y tablas de información.

Estos dos componentes permiten brindar un servicio, que facilite al usuario (estudiante de física 1), la realización de sus prácticas de laboratorio de manera remota, mucho más eficiente y cómoda, contando además con otras herramientas informáticas que no están presentes en las prácticas presenciales como simulaciones interactivas, comparación teórico-práctica, tabulación , gráficas de los datos y datos estadisticas.

10.2.2 Alcances: Se ha desarrollado una aplicación cliente-servidor que permite al usuario acceder a la planta de forma remota y manipularla para la toma de datos. Este permite mediante una interfaz amigable, intuitiva y con el menor número de interacciones la conexión al software de control de la planta, la adquisición, tabulación y graficación de los datos.

El sistema se ha implementado específicamente sobre la planta sistematizada *carril de aire* del laboratorio de física 1, para las prácticas *Movimiento Rectilíneo Uniforme y Movimiento Uniformente Acelerado*

que se desarrollan sobre esta y ser trabajado sobre la plataforma tecnológica de la CUAO, específicamente para trabajar sobre la red de área local (LAN).

10.3 SISTEMA GENERAL

El sistema de manera general permite a un estudiante de asignatura de Física 1, utilizar un PC de las salas de computo de la universidad para acceder sobre la red LAN, a una aplicación software servidor residente en un PC del laboratorio de fisica, la cual a su vez permite la interacción con el software de control de planta encargado de la comunicación con la Interfaz Autofísica 2.0 mediante el puerto serial y esta última realizará las acciones de control y sensado de las variables físicas de interés suministrando como respuesta la información recopilada llevándola hasta el estudiante en donde tendrá la posibilidad de tabular, graficar y animar los datos obtenidos.

11. CONCLUSIONES

Todo el sistema esta basado en lenguaje Java por lo cual el sistema es altamente escalable, no se tienen problemas de compatibilidad y de comunicaciones y se facilitó la programación y el mantenimiento del sistema brindando además posibilidades futuras de acceso seguro, asignación de ingresos programados (passwords), manejo de tareas distribuidas(RMI), procesos en paralelo (Threads), expansibilidad para el de manejo de bases de datos (ODBC) y acceso a Internet (Servlets y Java web Services).

Como resultado se obtuvo un producto bastante confiable, dadas las características intrínsecas del lenguaje (manejo de objetos, excepciones, hilos), de bajo consumo de recursos físicos y de computo, modular, por lo cual es posible implementarse a múltiples laboratorios o sistemas industriales con solo cambiar la interfaz de visualización y expandirla. La principal limitante de este sistema es la necesidad de tener el computador de monitoreo cercano a la planta y la utilización monousuario.

12. RECOMENDACIONES

El sistema es actualmente funcional aunque para ser práctico en la implementación a escala institucional es necesario realizar algunas mejoras como se listan a continuación:

- Servicio multiusuario (atención simultanea de varios estudiantes)
- Ingresos programados (passwords).
- Implementación de bases de datos para estadísticas y control.
- Acceso del servicio de laboratorio remoto a través de Internet.
- Implementación de una red tipo industrial con un microcontrolador administrándola y con acceso directo a la red LAN, para un sistema de varias plantas, evitando el uso de un computador dedicado.

Dadas las características del sistema desarrollado (escalable y modular), es posible utilizarlo para la realización de otras prácticas de laboratorio similares sin mayores modificaciones al nivel de hardware y software. Esto contribuye a cimentar un escalón firme para alcanzar un laboratorio de física completamente sistematizado y accesible por medio de Internet para la realización de las prácticas correspondientes

BIBLIOGRAFÍA

MORGAN, Mike. Descubre Java 1.2. Madrid : Prentice Hall, 1.998. 220p.

NAUGHTON, PATRICK y SHILDT, HEBERT. Java manual de referencia. Madrid : Osborne, 1.997. 370p.

HAROLD, ELLIOTTE Rusty. Java secrets. Foster City : IDG Books Worldwide, 1.997. 320p.

DEITEL, H.M. Como programar en java. México : Prentice Hall, 1.998. 350p.

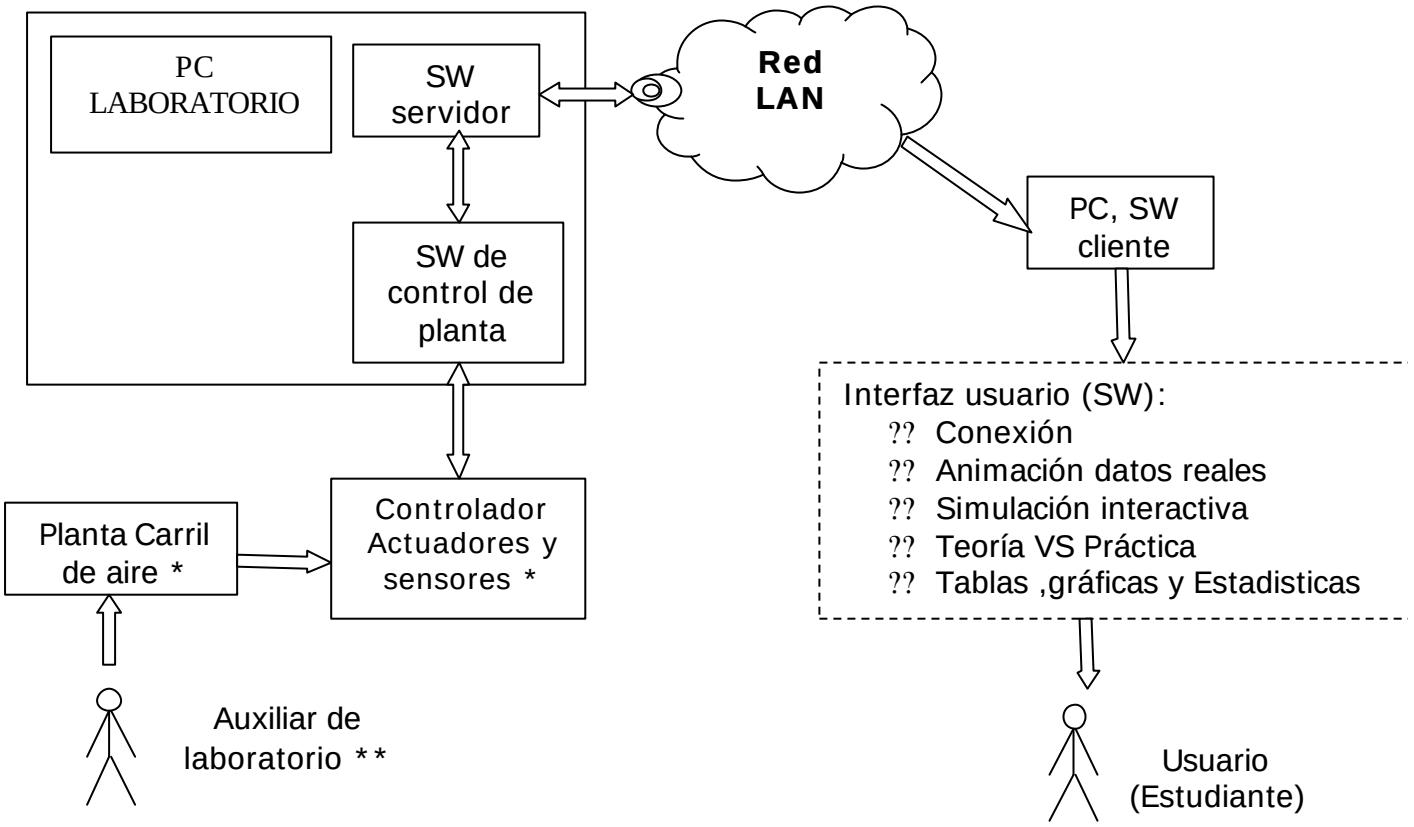
WUTKA, Mark. Hacking Java: the professional's resource kit. Indianápolis : Que Corporation, 1.997. 420p.

LARMAN, CRAIG. UML y Patrones: introducción al análisis y diseño orientado a objetos. México : Prentice Hall, 1.999. 260p.

JACOBSON, IVAR; BOOCH, GRADY y RUMBAUGH, JAMES. El proceso unificado de desarrollo de software. Madrid : Pearson Educación S.A., 2.000. 240p.

ANEXO A - DIAGRAMA DE BLOQUES GENERAL

Figura 26. Diagrama de bloques general.



***** Modificados totalmente para adecuarlos a los requerimientos de un laboratorio remoto.

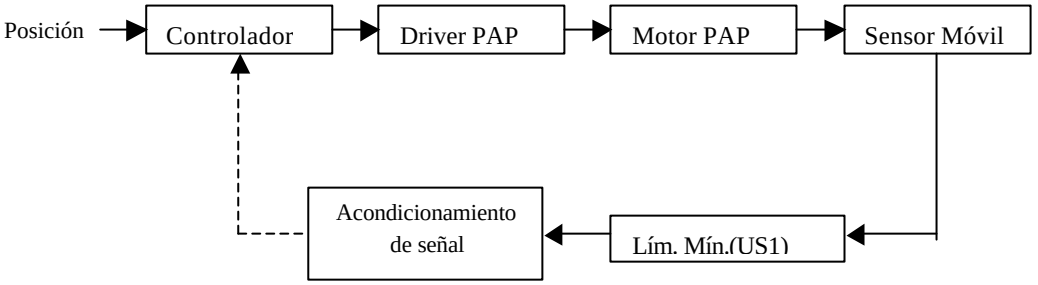
** Se encargará únicamente de iniciar la práctica alistando los equipos y atender las posibles fallas mecánicas que llegaran a producirse.

Nota: El proyecto se ha centrado en el reacondicionamiento y mejoramiento de la planta, diseño y desarrollo del hardware de control y del software cliente-servidor y las interfaces de usuario, así mismo como el modelo de comunicación entre el usuario y la planta. La plataforma tecnológica utilizada consta básicamente de un computador como servidor y una interfaz electrónica de control de planta, la red LAN de la universidad y las terminales de usuario (PC´s) en las salas de computo.

ANEXO B - DIAGRAMAS DE LAZO DEL HARDWARE REMOTO

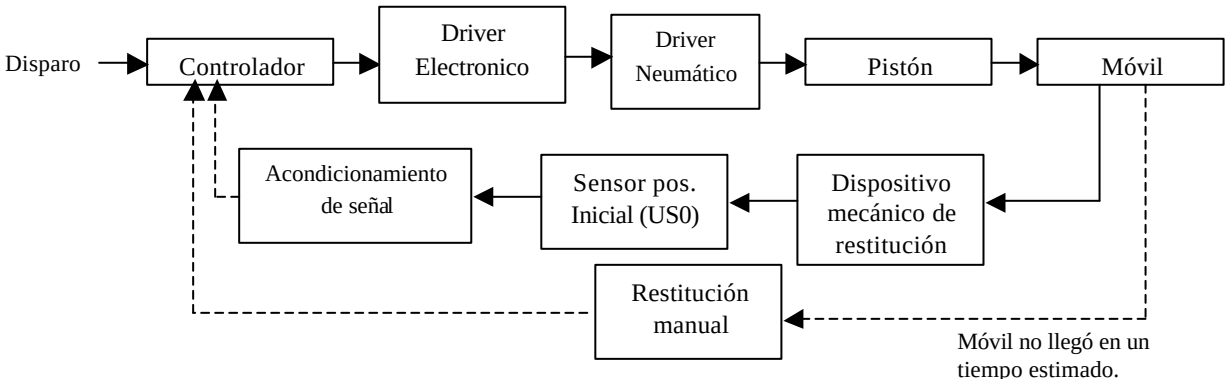
Posicionar sensor

Figura 27. Diagrama de lazo de control del sensor.



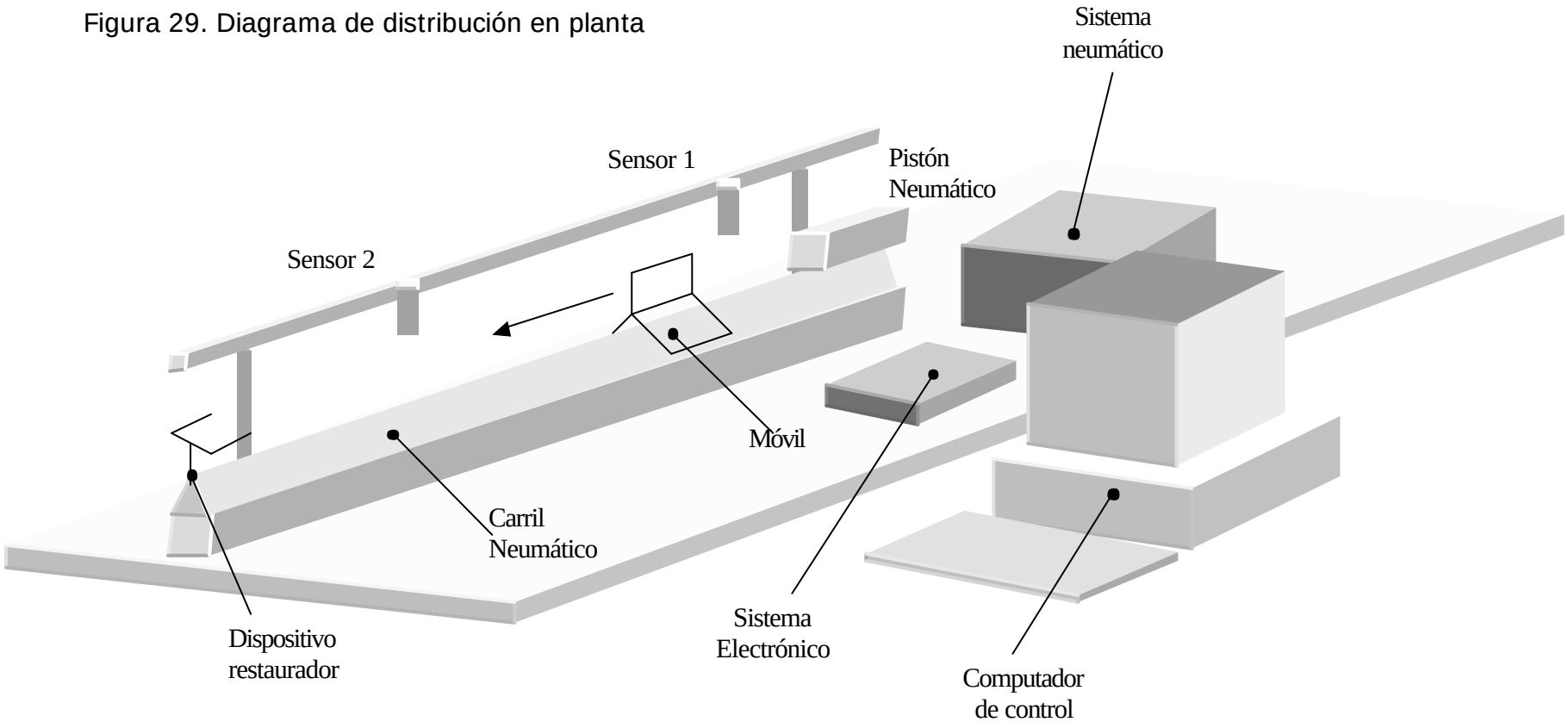
Desplazamiento del móvil

Figura 28. Diagrama de lazo de control del móvil.



ANEXO C DIAGRAMA ESQUEMATICO DE DISTRIBUCIÓN EN PLANTA

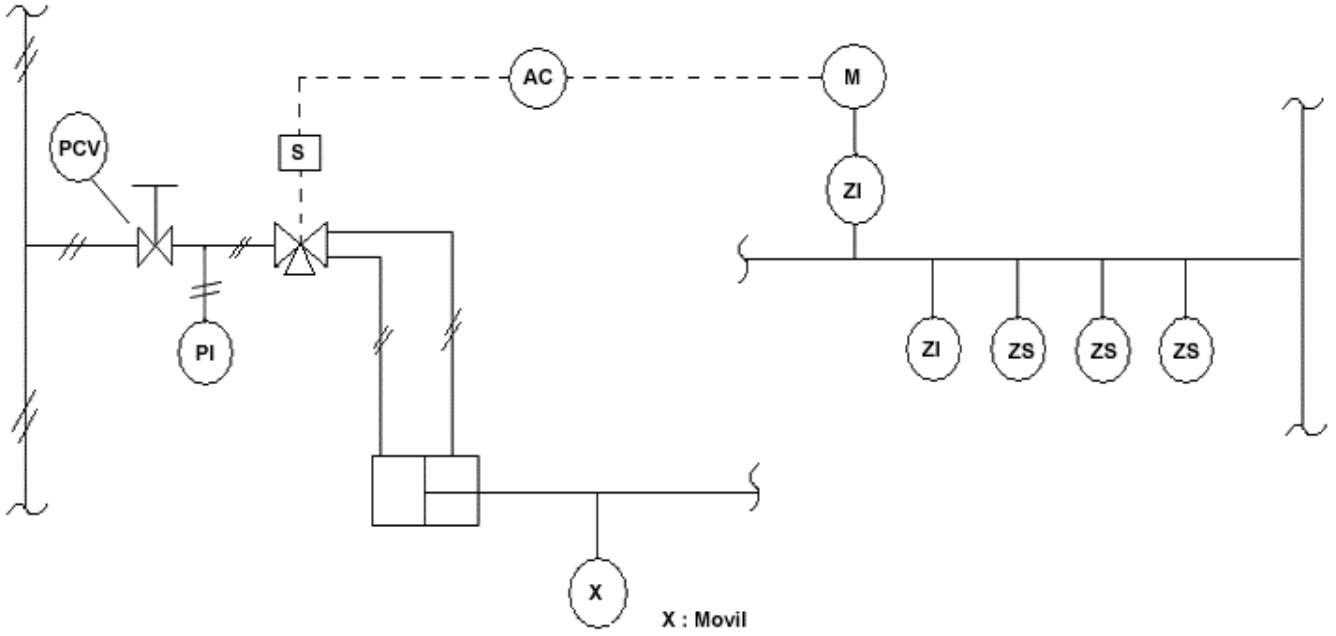
Figura 29. Diagrama de distribución en planta



.

ANEXO D DIAGRAMA PID

Figura 30. Diagrama PID.



ANEXO E PRUEBA DE PRECISIÓN DEL DISPOSITIVO

Es importante aclarar, que en la definición de los conceptos solución para el dispositivo disparador se realizaron pruebas y según la toma de datos se escogió el pistón neumático por un criterio de precisión.

Tabla 1. Resultados de pruebas de precisión de actuadores.

	Disparo con actuador manual base de referencia					Disparo con actuador eléctrico (solenoide)				
X(m)	T(S)	t(S)	dVm	Vm	E%	T(S)	t(S)	dVm	Vm	E%
0.100	0.123	0.029	0.015	0.813	1.810	0.325	0.072	0.004	0.307	1.300
0.200	0.245	0.029	0.007	0.816	0.910	0.605	0.073	0.002	0.331	0.660
0.300	0.361	0.029	0.005	0.831	0.610	0.925	0.074	0.001	0.324	0.430
0.400	0.480	0.029	0.004	0.833	0.460	1.250	0.075	0.001	0.320	0.340
0.500	0.600	0.028	0.003	0.833	0.370	1.575	0.077	0.001	0.317	0.250
0.600	0.722	0.028	0.003	0.831	0.300	1.900	0.077	0.001	0.316	0.220
0.700	0.835	0.028	0.002	0.838	0.260	2.190	0.076	0.001	0.320	0.190
0.800	0.940	0.028	0.002	0.851	0.240	2.550	0.078	0.001	0.314	0.160
0.900	1.060	0.028	0.002	0.849	0.200	2.825	0.078	0.001	0.319	0.160
1.000	1.175	0.028	0.002	0.851	0.190	3.180	0.079	0.000	0.314	0.130
		0.028		**0.840**			**0.077**		**0.318**	

	Pistón neumático débilmente soportado (1)					Pistón neumático fuertemente soportado (2)				
X(m)	T(S)	t(S)	dVm	Vm	E%	T(S)	t(S)	dVm	Vm	E%
0.100	0.089	0.021	0.024	1.124	2.130	0.054	0.017	0.053	1.852	2.580
0.200	0.186	0.021	0.011	1.075	1.040	0.124	0.017	0.021	1.613	1.310
0.300	0.278	0.022	0.008	1.079	0.700	0.198	0.017	0.013	1.515	0.840
0.400	0.361	0.021	0.006	1.108	0.520	0.265	0.017	0.010	1.505	0.630
0.500	0.450	0.021	0.005	1.111	0.420	0.337	0.017	0.007	1.484	0.500
0.600	0.530	0.021	0.004	1.132	0.350	0.405	0.017	0.006	1.482	0.410
0.700	0.626	0.021	0.003	1.118	0.300	0.485	0.017	0.005	1.443	0.350
0.800	0.721	0.021	0.003	1.110	0.260	0.558	0.017	0.004	1.434	0.310
0.900	0.830	0.022	0.003	1.084	0.230	0.620	0.017	0.004	1.452	0.280
1.000	0.923	0.022	0.002	1.083	0.210	0.692	0.017	0.004	1.445	0.240
		0.021		**1.103**			**0.017**		**1.470**	

Todos los resultados de T y t son promedios de las prácticas. Los datos inferiores de las columnas t y Vm son promedios de las columnas correspondientes excluyendo los 2 primeros valores.

E%=dVm.100/Vm , donde se toma dVm como

dVm=|(dX/T)|+|(X.dt/T^2)|

e%=(v-Vm).100/v

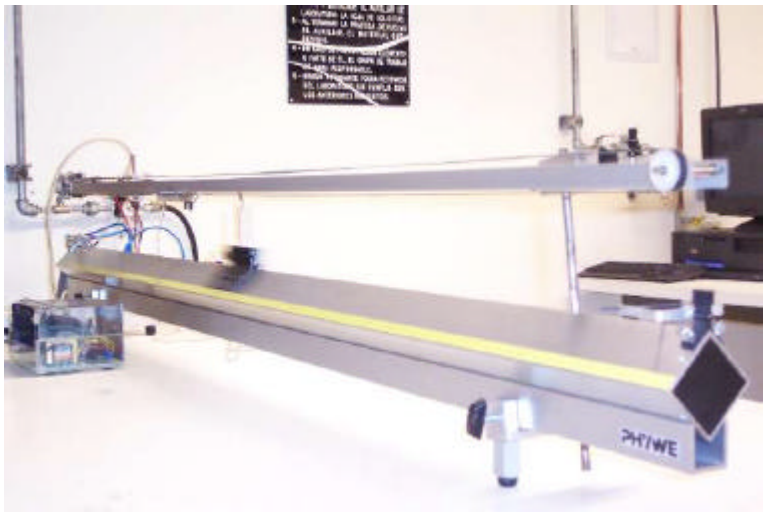
La siguiente tabla muestra en resumen, los resultados de las prácticas de laboratorio de movimiento rectilíneo uniforme realizadas para estudiar y comparar los diferentes tipos de actuadores para el disparo automático del móvil con relación a un disparo manual. El valor de v (velocidad "instantánea") se calculó con el tiempo t y la distancia de la placa oscura (0.025m).

Tabla 2. Resultados generales de pruebas de precisión de actuadores.

	Vm	v	e%
Manual	0.840	0.893	5.940
Solenoide	0.318	0.329	3.340
Pistón (1)	1,103	1,190	7.310
Pistón (2)	1,470	1,471	0.070

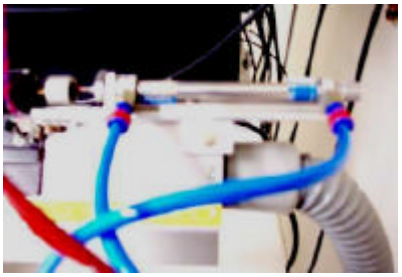
Cabe anotar en este punto que las incertidumbres del sistema una vez montado y probado son del orden de +/-0.001s según se pudo apreciar en comparación con los cronómetros electrónicos del laboratorio.

ANEXO F RESUMEN FOTOGRÁFICO

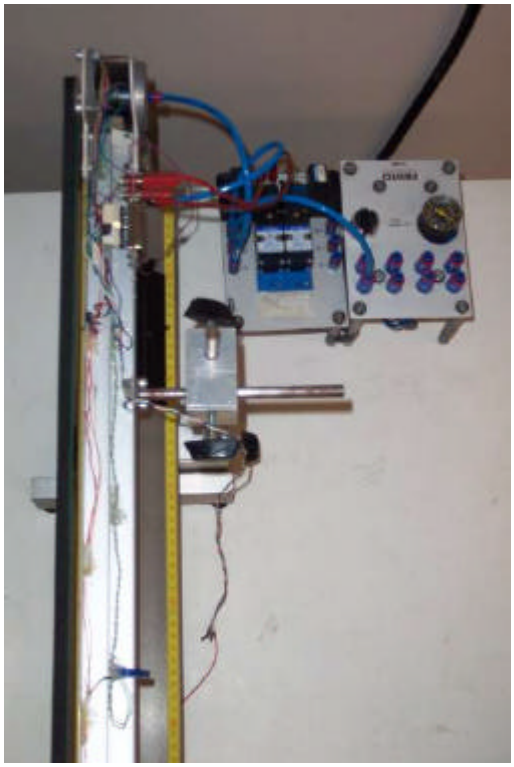


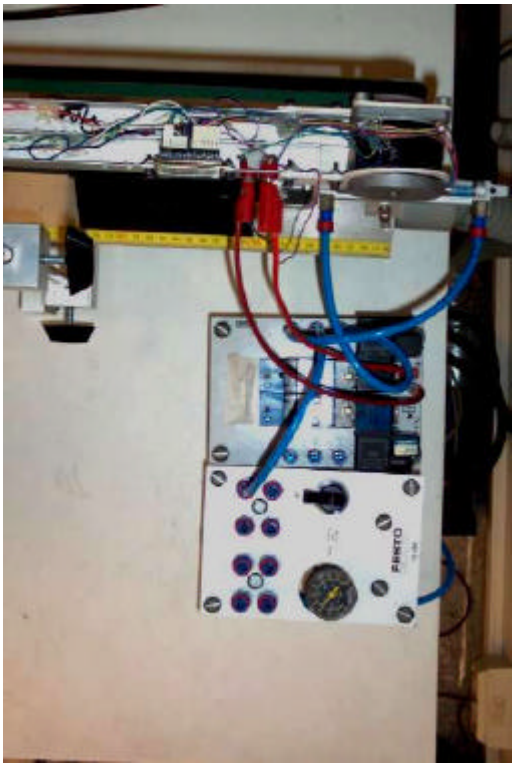
Distribución general de la planta carril de aire.

Interfaz Electrónica Autofísica Versión 2.0. (IE-AF2.0).

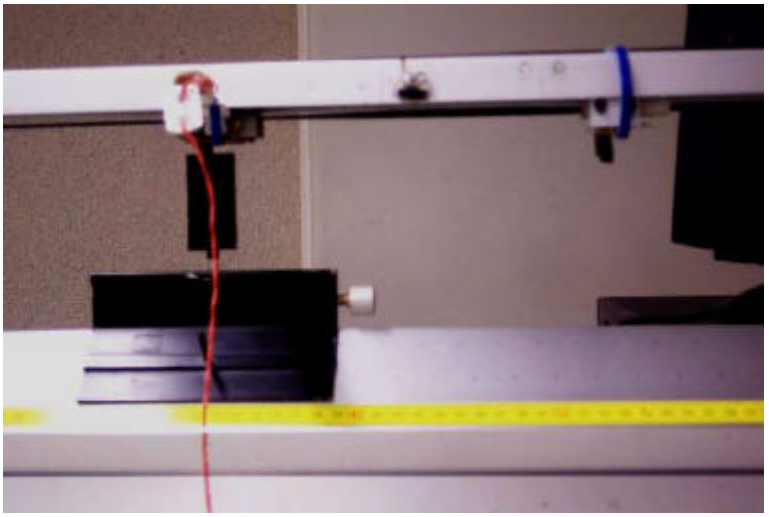


Detalles del sistema neumático. Toma de aire comprimido y pistón neumático (Actuador de disparo remoto).

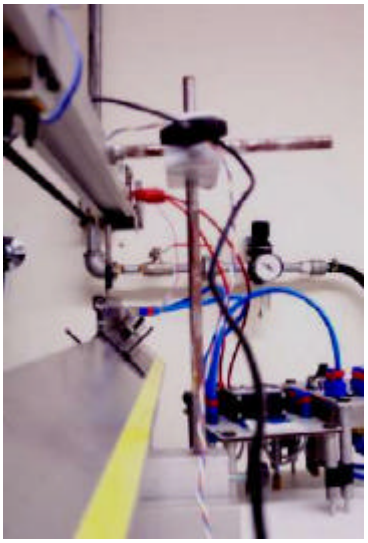


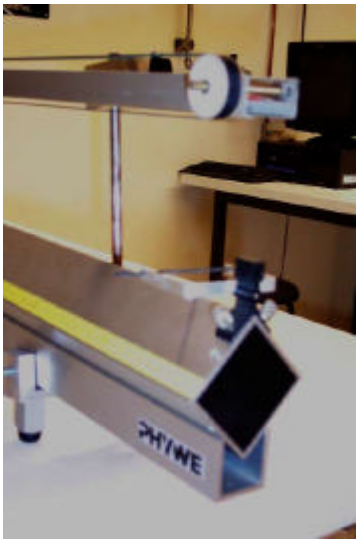


Distribución del sistema neumático (válvulas, electroválvulas y reguladores de presión). Vistas superiores. Sistema neumático y conexiones en color azul.

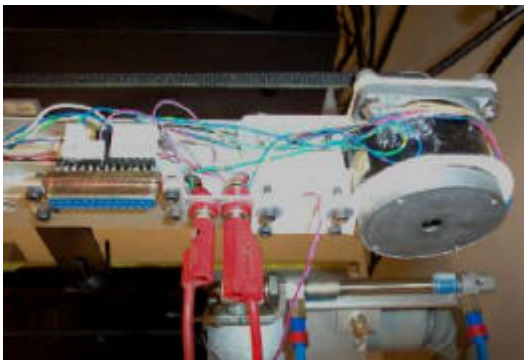


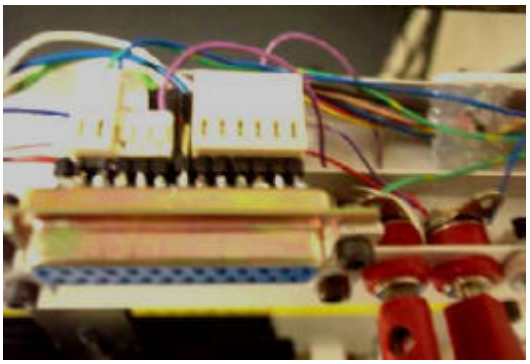
Sistema de sensado fotoeléctrico. Vista lateral. El móvil (en negro) posee una aleta vertical, la cual pasa por dos sensores ópticos. El primero es fijo y el segundo (con cable rojo) es móvil. Se aprecia en la parte central superior el sensor para la posición cero del sensor móvil.



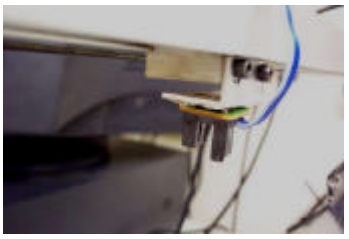


Vistas del soporte mecánico, soportes del carril de los sensores y el resorte restaurador del móvil al final.





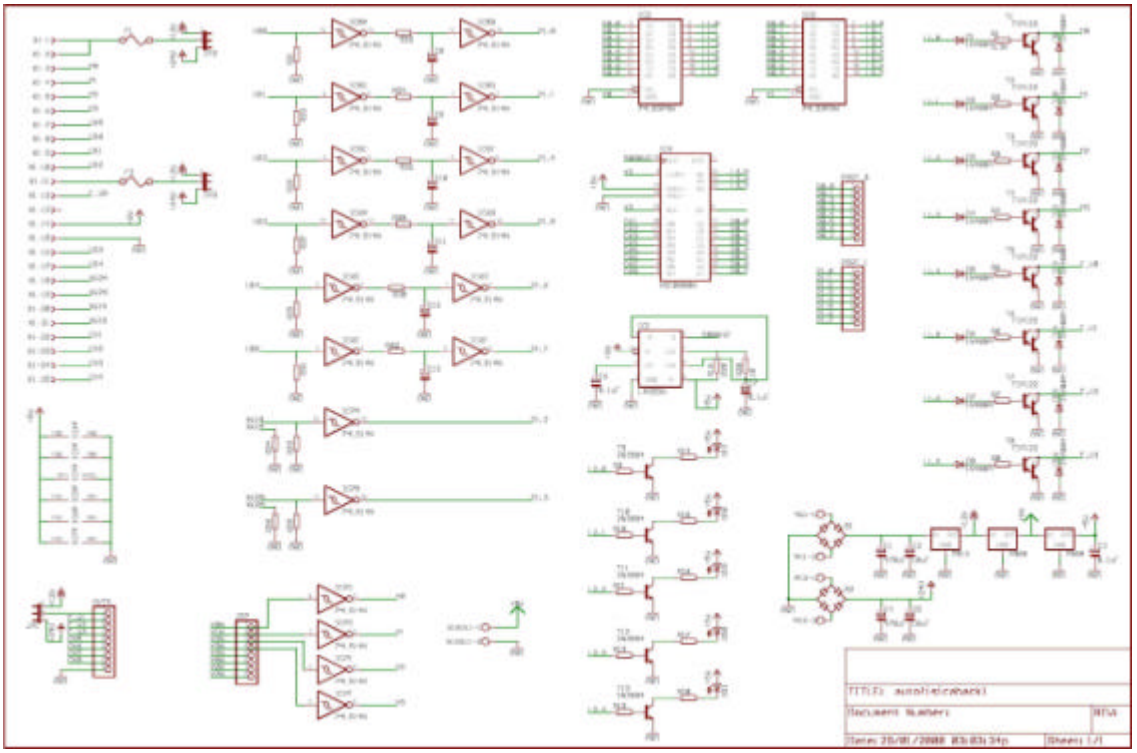
Detalles del sistema de cableado eléctrico. Se puede apreciar el conector DB-25 (en azul) al cual se conecta el cable respectivo desde la IE-AF2.0, el cableado interno hacia los sensores y el motor PAP y las dos borneras (rojas), a las cuales van conectadas los dos cables para el disparo de la electroválvula.



Detalle del sensor fotoeléctrico (fototransistor).

ANEXO G PLANO ELÉCTRICO DEL SISTEMA IE-AF2.0.

Figura 31. Plano eléctrico del sistema IE-AF2.0



ANEXO H. IMPLEMENTACIÓN DEL SOFTWARE REMOTO (SR-AF2.0).

Como se ha mencionado en contadas ocasiones, a continuación explicaremos la implementación de los módulos funcionales a manera de objetos.

Principalmente existen cinco objetos fundamentales que realizan todas las labores del software desarrollado expuestas anteriormente, es decir, existen cinco objetos principales(clases) que implementan el modelo comunicativo y el modelo funcional del software. Estas cinco clases son *swcliente, Cliente, swremoto, Servidor y SerialConnection.* Como es de suponer las clases *swcliente* y *Cliente* corresponden a la implementación de todo el lado del cliente en el modelo y las clases *swremoto, Servidor y SerialConnection* a la implementación de todo el lado del servidor.

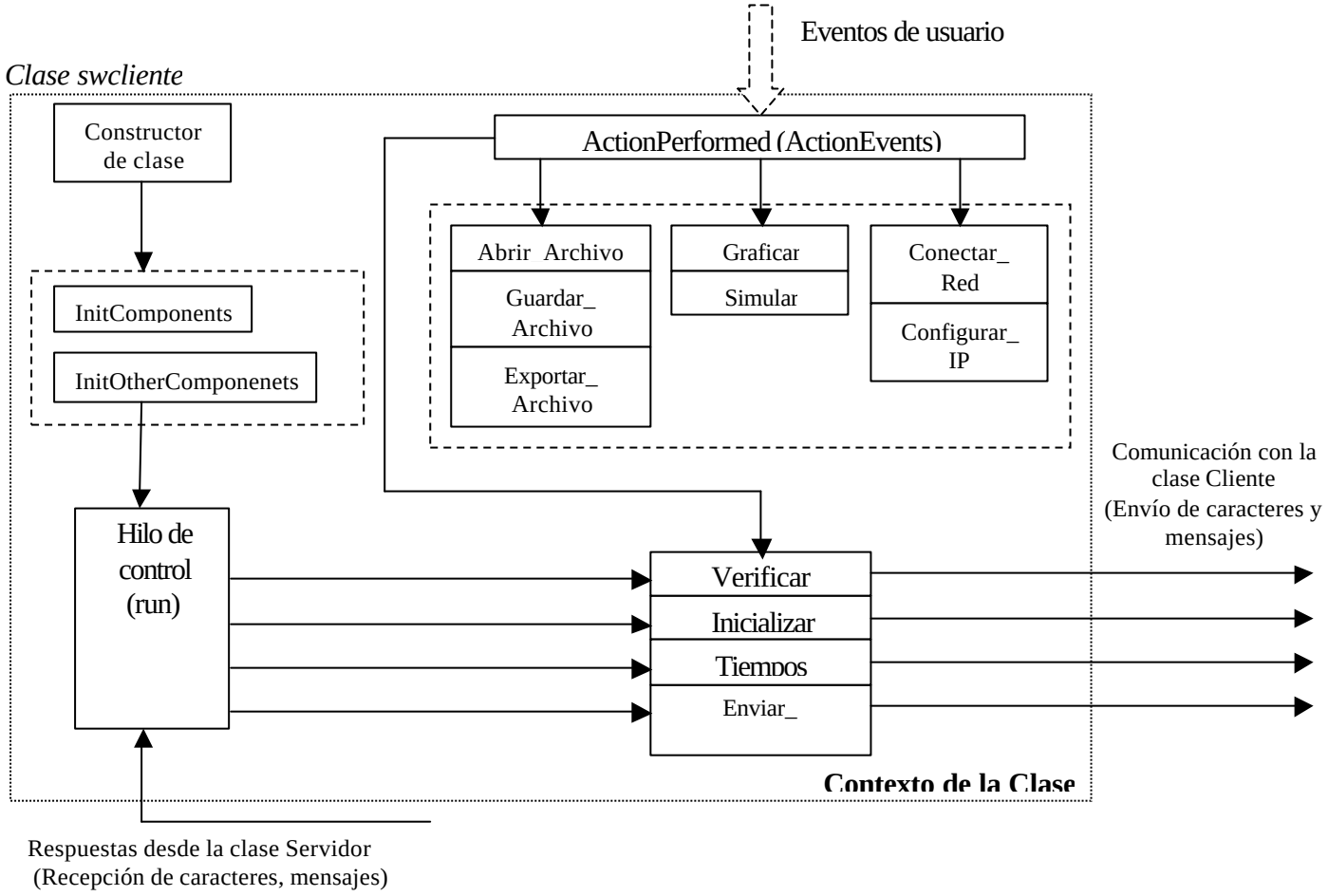
Clase swcliente

La clase swcliente corresponde a la implementación de todo el submódulo funcional swcliente del módulo SWCliente del lado del usuario. Esta clase contiene la implementación de los módulos

funcionales IG-AF2.0, GA-AF2.0, GE-AF2.0, MCA-AF2.0 que componen dicho submódulo funcional.

Por tanto se puede decir que la clase swcliente contiene la implementación de la gestión de archivos, la gestión de excepciones para el para las operaciones y acciones de interfaz de usuario, los componentes gráficos de interfaz de usuario y el hilo de control que gestiona todo el control de acciones de interfaz basado en eventos.

El siguiente diagrama presenta los principales métodos de la clase mediante los cuales se implementan estos módulos funcionales. Internamente la clase implementa otros métodos secundarios que son utilizados por los métodos principales que se muestran en el diagrama, dichos métodos secundarios se explicarán a medida que se describa la función de cada método y su relación con los demás métodos de la clase.



Como se observa en el diagrama, la clase cuenta con un constructor y a partir de este, se inicializan funciones , procedimientos y relaciones entre los diferentes componentes de la clase.

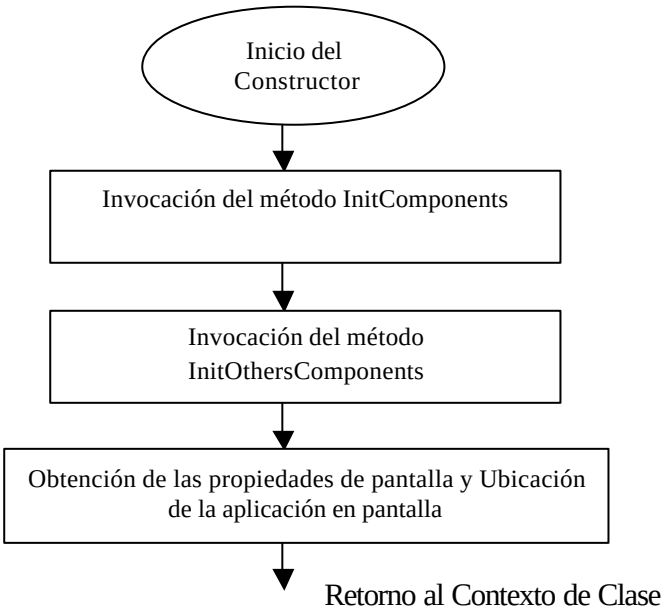
Hemos definido el contexto de la clase como el ambiente en el cual se procesan los distintos cambios y estados que la clase puede sufrir según la ejecución de eventos internos en un sistema de ejecución. Esto es

debido a que al terminar de ejecutarse las acciones realizadas por cada método, la clase misma en un entorno de ejecución se queda a la espera de eventos para realizar nuevas acciones a través de sus métodos.

Constructor de la clase

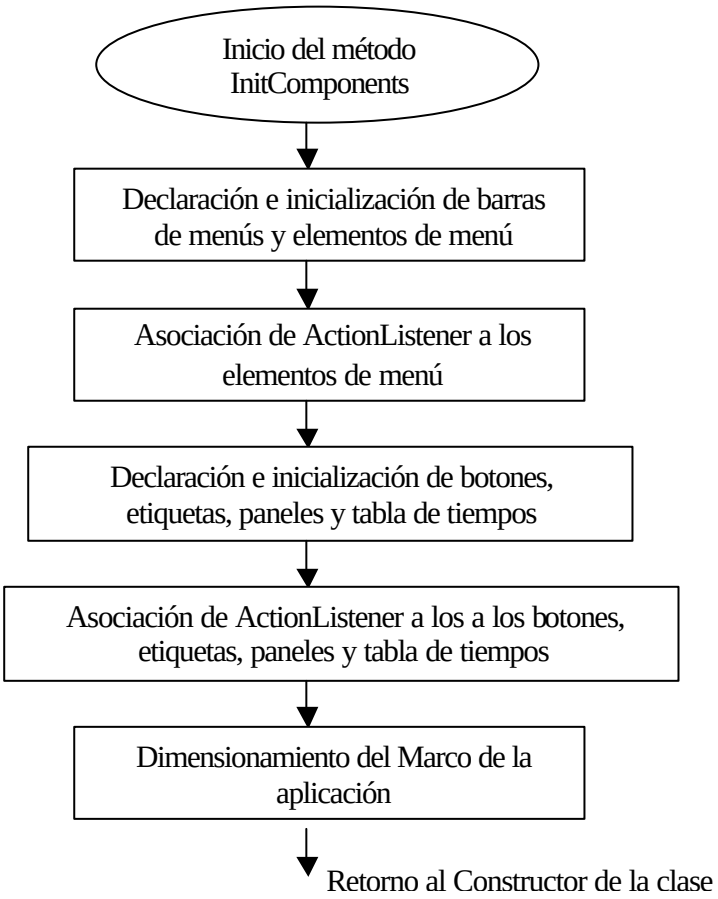
Este método en particular invoca los métodos de inicialización de componentes gráficos de la interfaz de usuario y declaración e inicialización de otros componentes funcionales dentro de la implementación.

Un simple diagrama de flujo nos ilustra su funcionamiento.



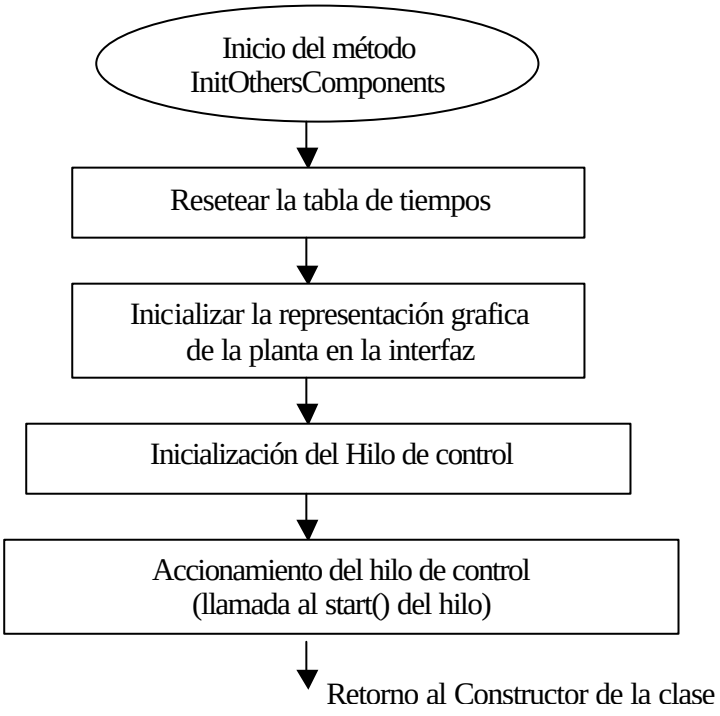
Método InitComponents

Este método permite la declaración e inicialización de todos los componentes graficos de la interfaz de usuario como son Barras de Menus, Elementos de menus, Botones, Etiquetas, Tabla de listado de tiempos etc. En este método no solamente se inicilaizan sino que además se configuran todas las propiedades de dichos componentes graficos para su utilización y visualización en pantalla.



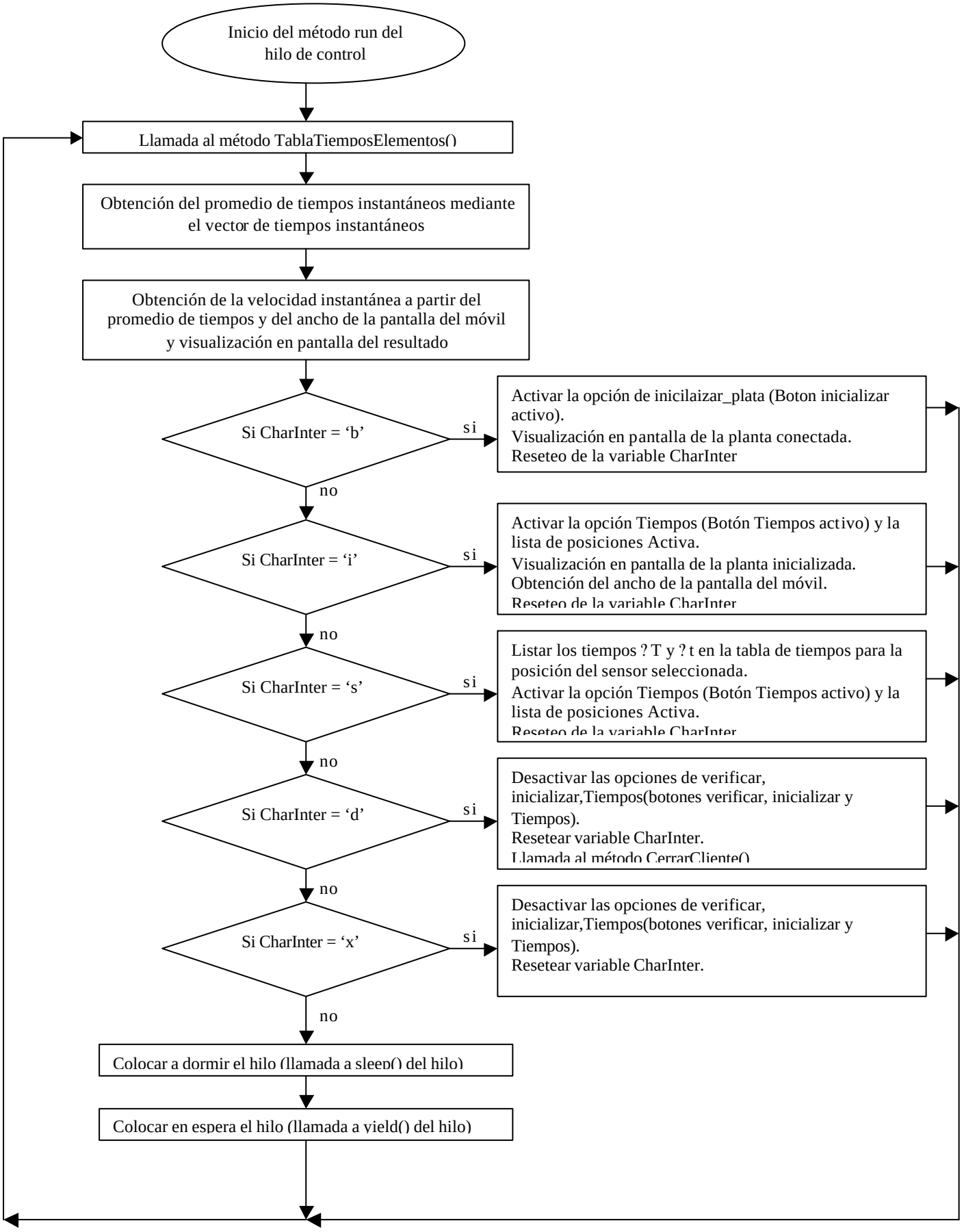
Método InitOthersComponents

Este método permite la inicialización de otros componentes funcionales del programa como pueden ser hilos de control ademas de otros objetos que no constituyan componentes GUI. En este método se inicializan tambien la representación grafica de la planta en la interfaz.



Método run del hilo de control

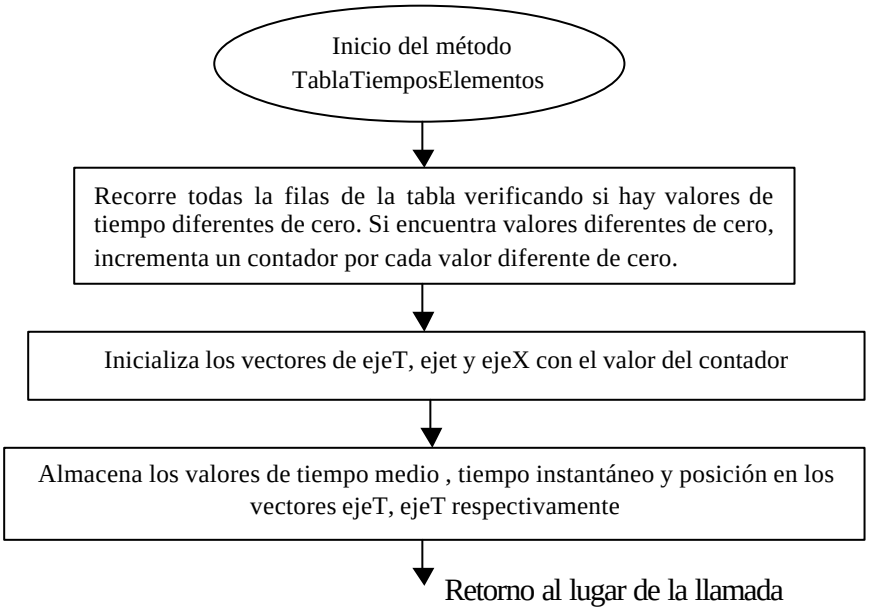
Este método se ejecuta una vez se haga la llamada al metodo *start()* del hilo y es el encargado de ejecutar el ciclo pricipal del programa, el cual correponde a un ciclo continuo e infinito. Por esta razón y para evitar bloqueos del proceso principal, se ha definido este ciclo en un hilo de control con un tiempo de espera de ejecucion definido por el monitor de hilos del entorno de ejecución de la clase(en este caso el entorno de ejecución de JAVA JVM). En dicho ciclo se evalua constantemente una variable llamada *CharInter* en la cual se almacena el valor de llegada por el canal TCP que implementa la clase Cliente como se vera mas adelante. Por tanto, en este ciclo se encuesta dicha variable y según su valor se realiza la activación de los elementos de interfaz grafica que habilitan al usuario para realizar peticiones al servidor y por ende interactuar remotamente con la planta.


Inicio del método run del hilo de control
Llamada al método TablaTiemposElementos()
Obtención del promedio de tiempos instantáneos mediante el vector de tiempos instantáneos
Obtención de la velocidad instantánea a partir del promedio de tiempos y del ancho de la pantalla del móvil y visualización en pantalla del resultado
Si CharInter = 'b'
si
Activar la opción de inicilaizar_plata (Boton inicializar activo).
Visualización en pantalla de la planta conectada.
Reseteo de la variable CharInter
no
Si CharInter = 'i'
si
Activar la opción Tiempos (Botón Tiempos activo) y la lista de posiciones Activa.
Visualización en pantalla de la planta inicializada.
Obtención del ancho de la pantalla del móvil.
Reseteo de la variable CharInter
no
Si CharInter = 's'
si
Listar los tiempos ? T y ? t en la tabla de tiempos para la posición del sensor seleccionada.
Activar la opción Tiempos (Botón Tiempos activo) y la lista de posiciones Activa.
Reseteo de la variable CharInter
no
Si CharInter = 'd'
si
Desactivar las opciones de verificar, inicializar,Tiempos(botones verificar, inicializar y Tiempos).
Resetear variable CharInter.
Llamada al método CerrarCliente()
no
Si CharInter = 'x'
si
Desactivar las opciones de verificar, inicializar,Tiempos(botones verificar, inicializar y Tiempos).
Resetear variable CharInter.
no
Colocar a dormir el hilo (llamada a sleep() del hilo)
Colocar en espera el hilo (llamada a yield() del hilo)

Como se observa en el diagrama anterior, se realizan llamdas a dos metodos que explicaremos a continuación. Estos metodos son TablaTiempos() y CerrarCliente().

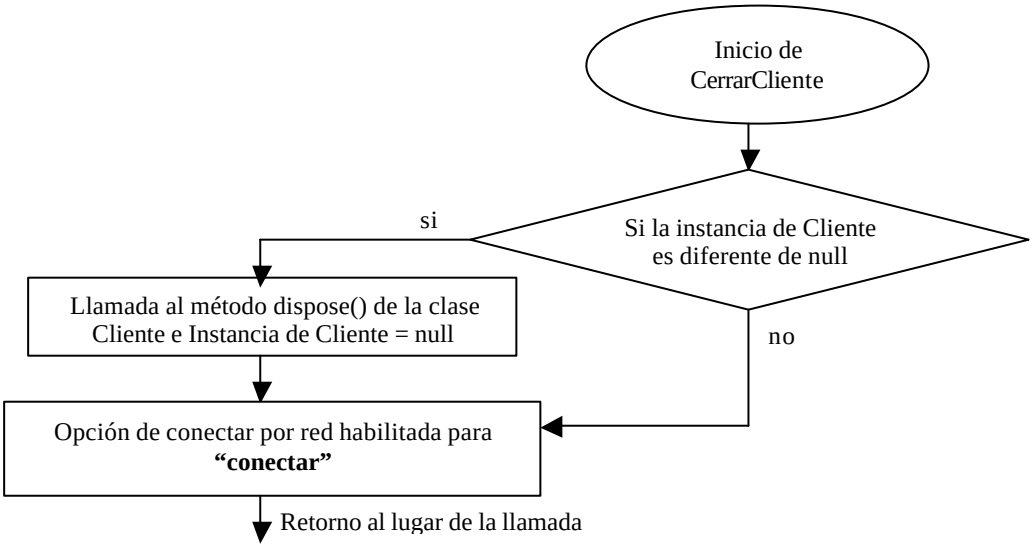
TablaTiemposElementos()

Basicamente, este método obtinene los verifica si hay valores de tiempo en la tabla diferentes de cero e inicializa los vectores de tiempo donde se almacenaran dichos datos. Una vez inicializados los vectores se almacenan los valores de tiempo en los mismos.



CerrarCliente

Este método cierra la conexión TCP desde la clase swcliente. Esto lo realiza, mediante la invocación del metodo dispose() de la clase Cliente, el cual libera todos los recursos de red solicitados para establecer el canal TCP que comunicara con el servidor.



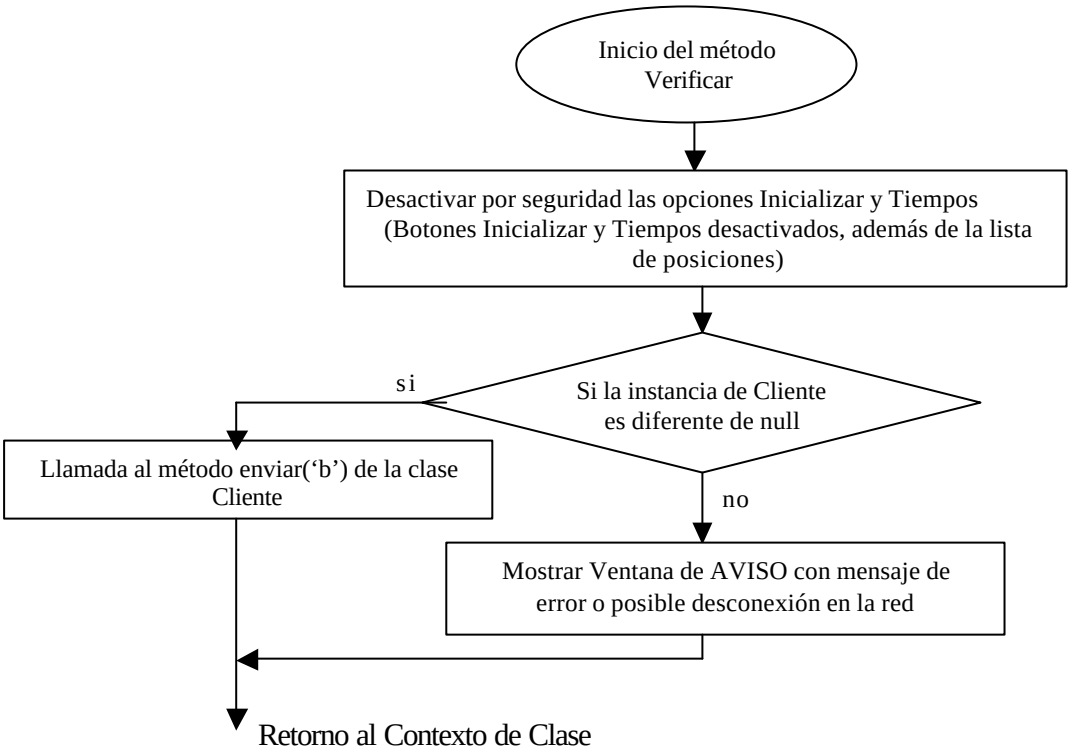
Método ActiónPerformed

Este método es invocado y ejecutado cuando el oyente de acciones y eventos de los componentes de interfaz recibe eventos procedentes de acciones del mouse o del teclado a través del manejador de ventana y

el adaptador de teclado del sistema. En este caso el mouse y el teclado pueden actuar sobre los elementos de interfaz grafica, seleccionándolos y activándolos por lo cual el oyente asociado a los elementos ejecuta el ActiónPerformed y a su vez dicho método, de acuerdo al elemento de interfaz sobre el cual se realiza la acción o el evento , puede invocar a los métodos propios de la clase como son Abrir_Archivo, Guardar_Archivo, Exportar_Archivo, Graficar, Simular, Conectar_Red, Configurar_IP, Verificar, Inicializar, Tiempos y Enviar_Mensaje.

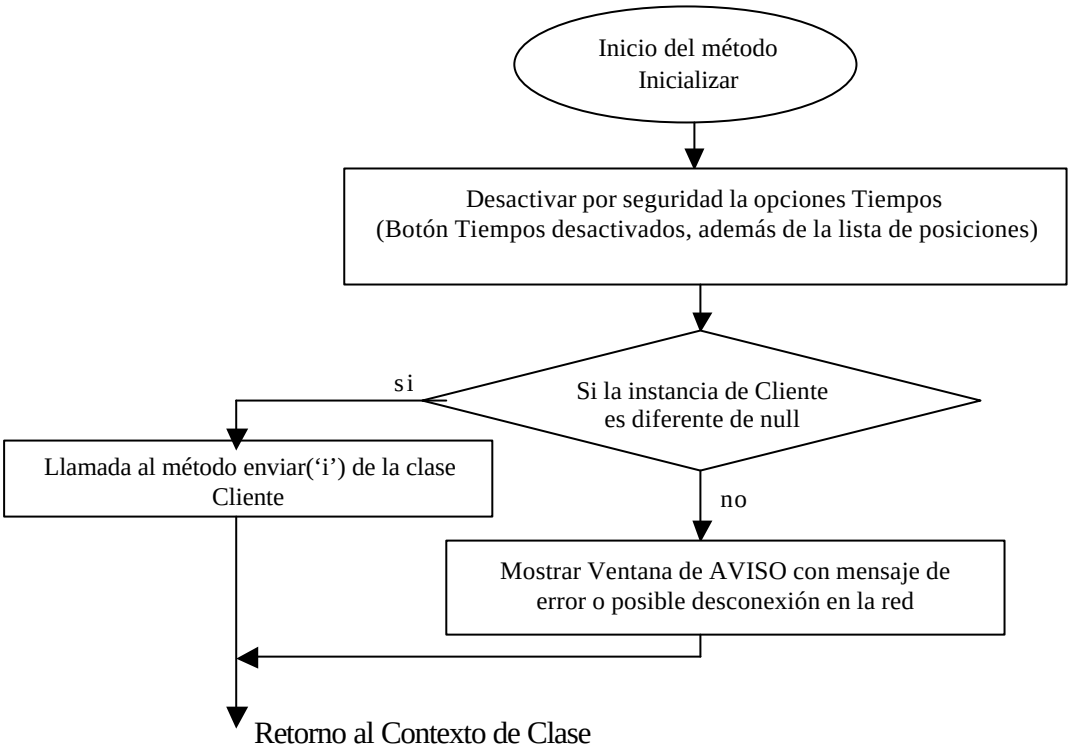
Método Verificar

En términos generales, este método envía el caracter ‘b’ como petición al servidor para verificar que la planta esta conectada. Esto lo realiza mediante la invocación del método enviar(int) de la clase Cliente el cual envía por el canal TCP el dato que obtiene como parámetro en su invocación (en este caso será el caracter ‘b’). Es necesario aclarar, aunque ya se había mencionado, que el método Verificar se invoca cuando se recibe el evento del accionamiento del botón **Verificar** de la interfaz grafica.



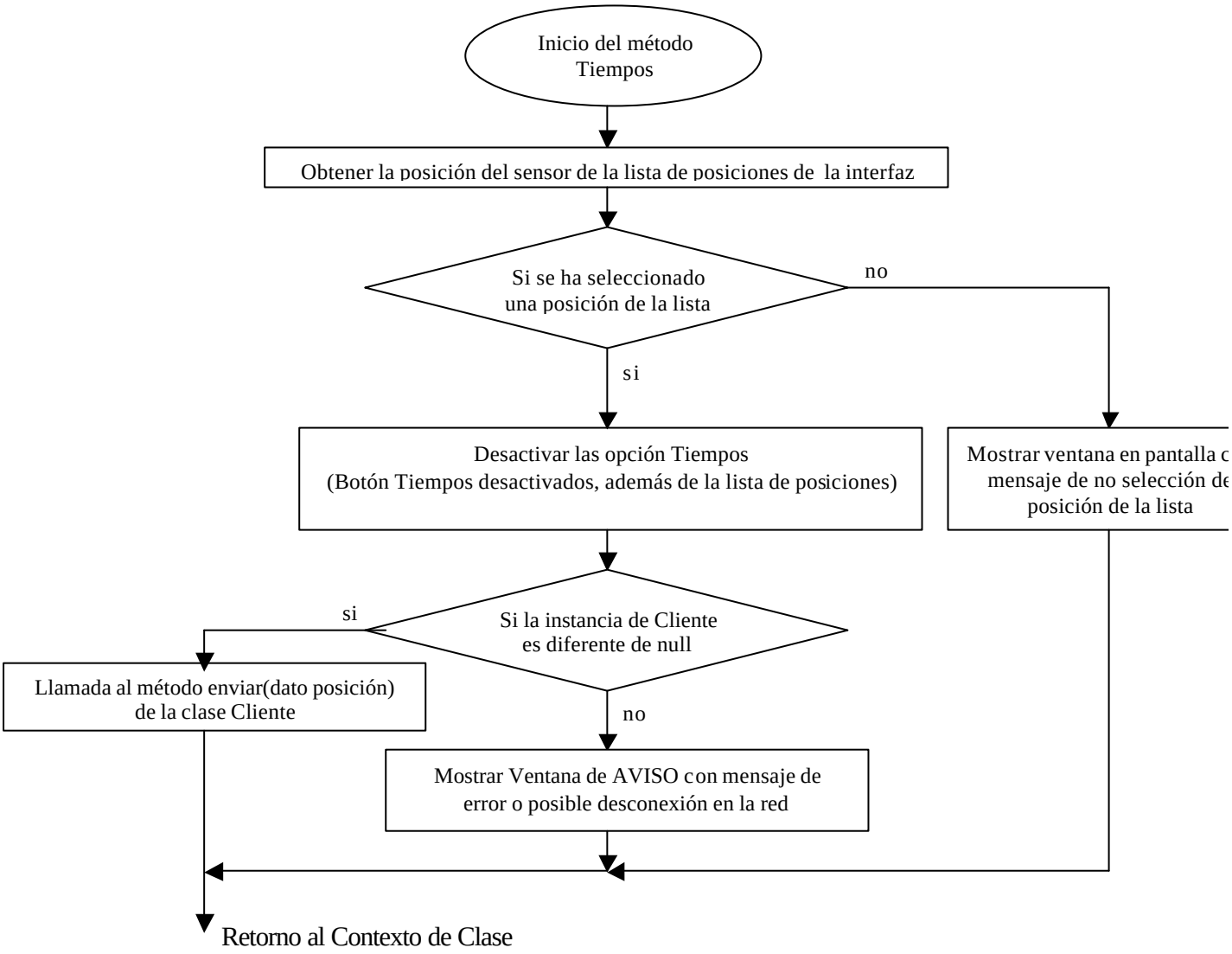
Método Inicializar

Este método envía el caracter 'i' como petición al servidor para la inicialización de la planta (Posicionamiento de sensores y elementos en sus puntos iniciales). Esto lo realiza mediante la invocación del método enviar(int) de la clase Cliente. El método Inicializar se invoca cuando se recibe el evento del accionamiento del botón **Inicializar** de la interfaz grafica.



Método Tiempos

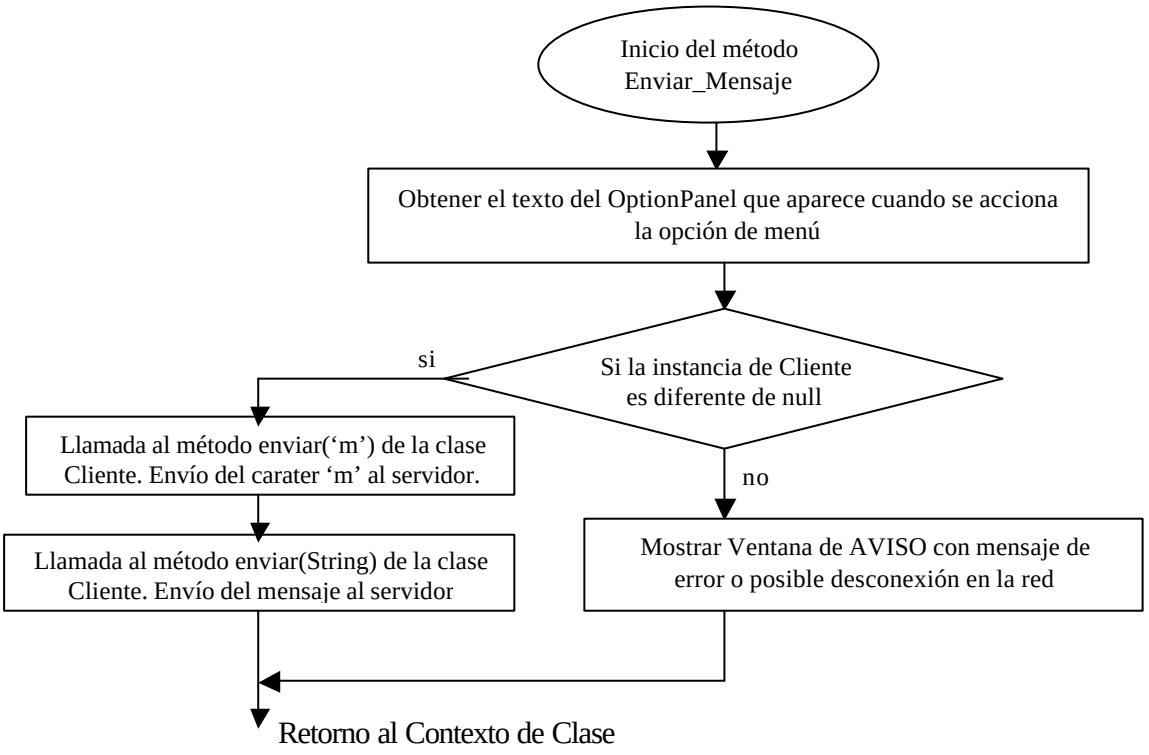
Este método captura la posición deseada para el sensor de tiempos de la lista de posiciones de la interfaz grafica. Con la posición deseada le envía el valor de la posición al servidor mediante la invocación del método enviar(int) de la clase Cliente. El método Tiempos se invoca cuando se recibe el evento del accionamiento del botón **Tiempos** de la interfaz grafica.



Método Enviar_Mesaje

Este método envía un mensaje de texto al servidor, en caso de que se requiera una comunicación con el auxiliar de laboratorio. Esto lo realiza

mediante la invocación del método enviar(String) de la clase Cliente. El método Enviar_Mensaje se invoca cuando se recibe el evento del accionamiento del elemento de menú **Enviar Mensaje** del menú de **Conexión** en la barra de menús.



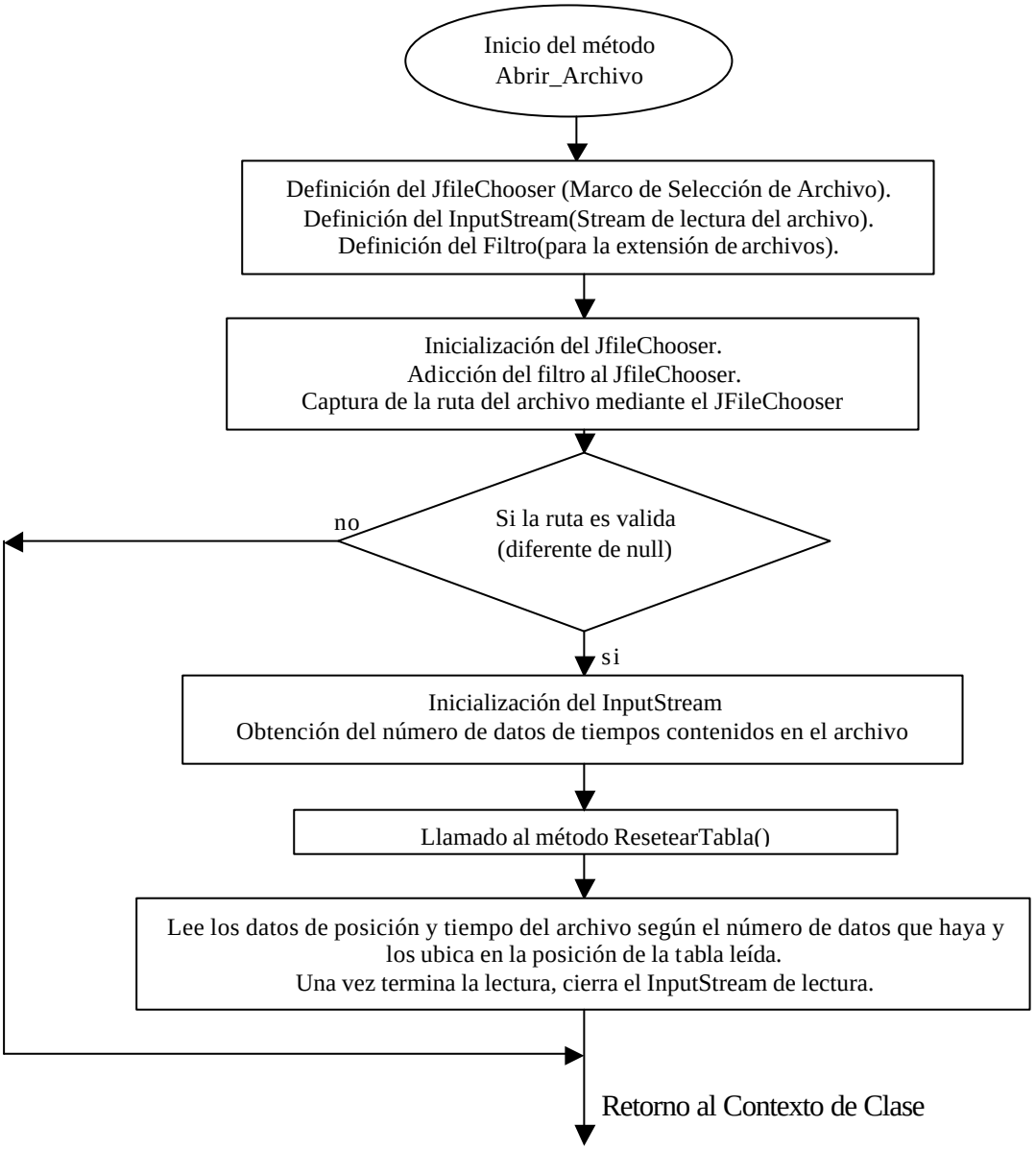
Hasta el momento podemos decir que mediante el constructor de la clase y los método de Inicialización de componentes se ha implementado el módulo de Interfaz Grafica (IG-AF2.0) y mediante el método run del hilo de control se ha implementado el módulo de Control

de Acciones(MCA_AF2.0) del submódulo *swcliente* en el lado de usuario del sistema AF-2.0.

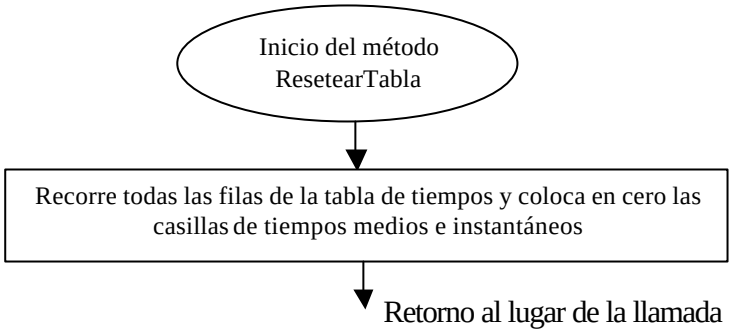
Método Abrir_Archivo

Este método se encarga de abrir un archivo que contenga los datos de tiempos y posiciones para listarlos en la tabla de tiempos de la interfaz grafica. Los archivos que se lean deben tener una extensión especifica que maneja internamente el programa.

El método Abrir_Archivo se invoca cuando se recibe el evento del accionamiento del elemento de menú **Abrir** del menú de **Archivo** en la barra de menús.



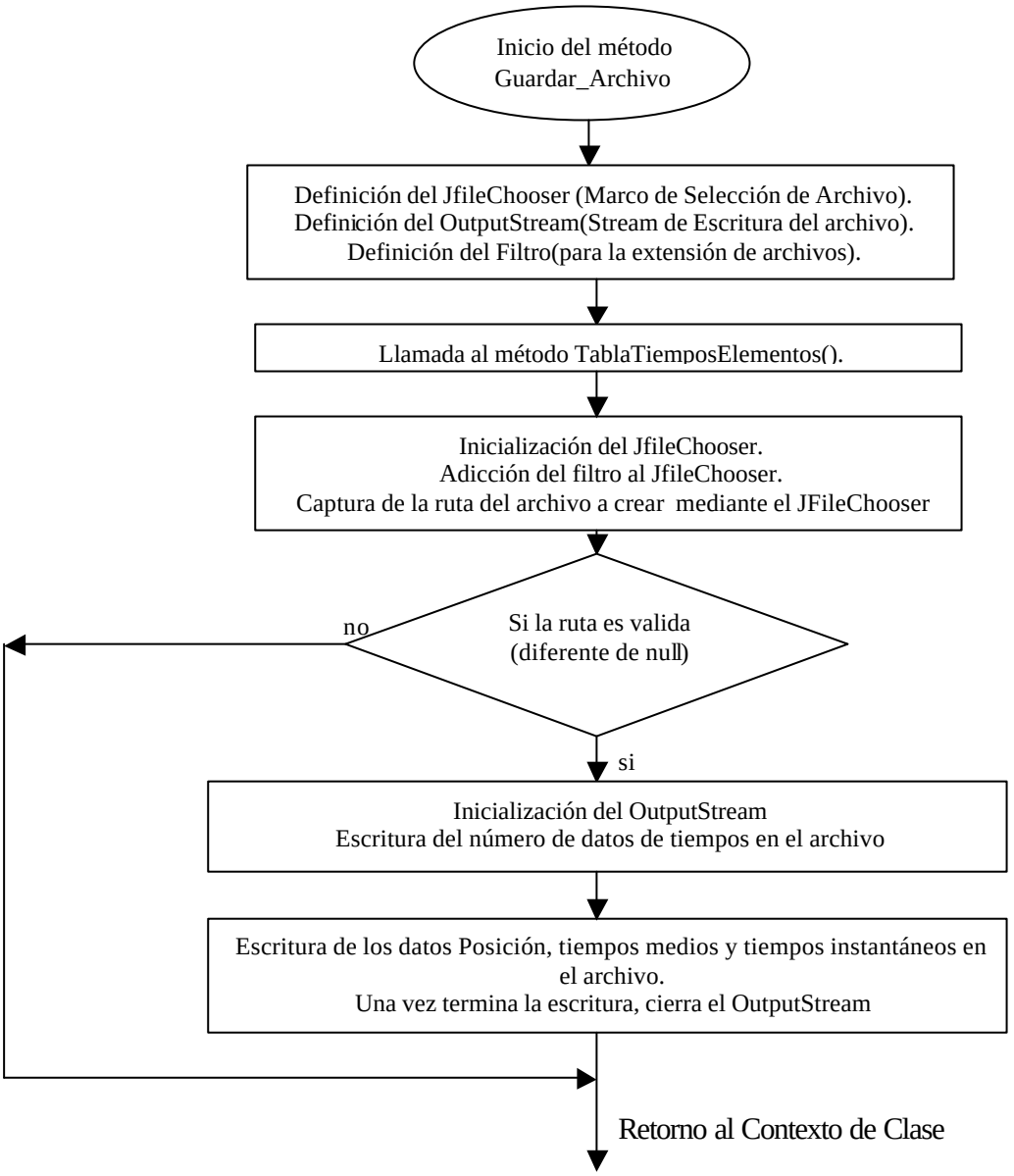
Aquí se invoca un método llamado ResetearTabla() cuya función es colocar en cero los valores de tiempos de la tabla, con el fin de que no existan valores previos a la lectura del archivo en la tabla y de esta manera solo se listen los valores leídos del archivo.



Método Guardar_Archivo

Este método se encarga de crear un archivo que contenga los datos de tiempos y posiciones bajo una extensión especifica que maneja internamente el programa (extensión *.ddt*).

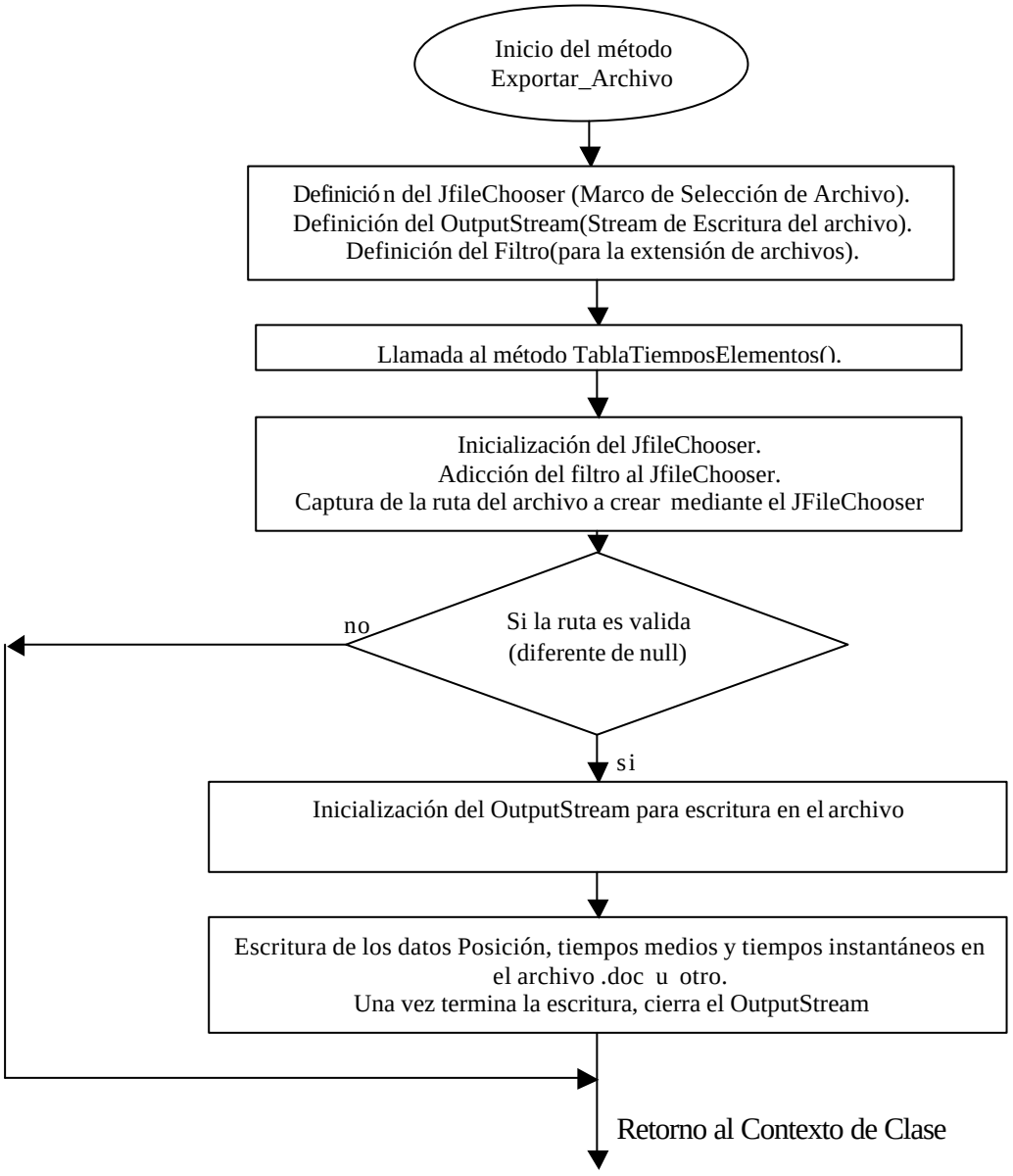
El método Guardar_Archivo se invoca cuando se recibe el evento del accionamiento del elemento de menú **Guardar** del menú de **Archivo** en la barra de menús.



Método Exportar_Archivo

Este método se encarga de crear un archivo que contenga los datos de tiempos y posiciones bajo extensión *.doc* o alguna otra extensión

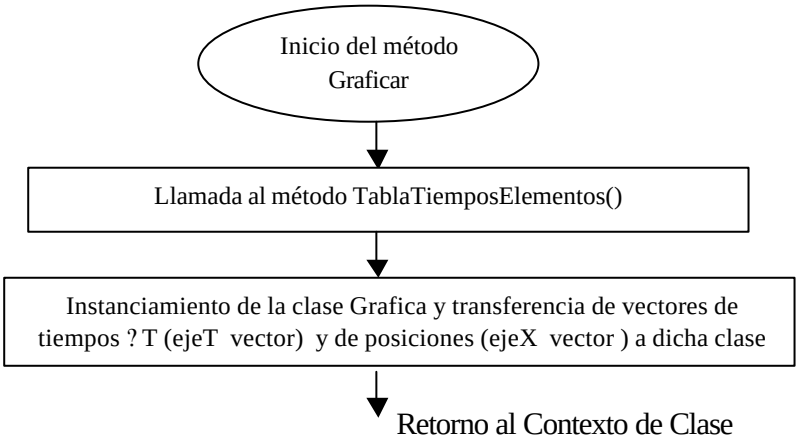
definida por el usuario. El método Exportar_Archivo se invoca cuando se recibe el evento del accionamiento del elemento de menú **Exportar** del menú de **Archivo** en la barra de menús.



Como se observa, a través de los métodos anteriores se implementa el módulo de Gestión de Archivos (GA-AF2.0) del Submódulo *swcliente* del lado de usuario.

Método Graficar

A través de este método se realiza una instancia de la clase Grafica que implementa el módulo de Análisis Grafico(MAG-AF2.0) del submódulo *swcliente* del lado del usuario. El método Graficar se invoca cuando se recibe el evento del accionamiento del elemento de menú **Graficar** del menú de **Herramientas** en la barra de menús.



La clase Grafica, implementa todos los métodos requeridos para realizar un análisis grafico de los datos de tiempo y posición. Es así como

implementa entones el módulo de Análisis Grafico (MAG-AF2.0). Sus principales método son:

Initcomponents(): Inicializa todos los componentes gráficos de la clase incluido el plano cartesiano.

lineal(): Dibuja en el plano cartesiano la recta obtenida según la ecuación y los coeficientes de la ecuación obtenidos mediante la regresión lineal de los datos de tiempo y posición.

cuadrática(): Dibuja en el plano cartesiano la curva obtenida según la ecuación y los coeficientes de la ecuación obtenidos mediante la regresión cuadrática de los datos de tiempo y posición.

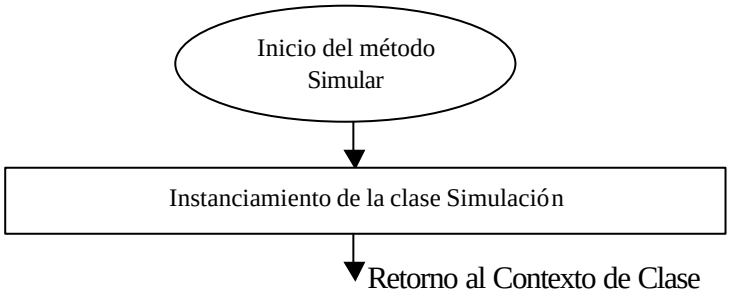
regresionLineal(): Realiza la regresión lineal de los datos de tiempo y posición para así obtener los coeficiente de la ecuación lineal. Dentro de este mismo método también se realiza la regresión cuadrática obteniendo los coeficiente de la ecuación de la curva según los datos del fenómeno.

correlacionLineal(): Obtiene la correlación lineal de los datos del fenómeno.

En esta clase también se obtiene los errores estándar para la aproximación lineal y cuadrática.

Método Simular

A través de este método se realiza una instancia de la clase Simulación que implementa el módulo de Simulación(MS-AF2.0) del submódulo *swcliente* del lado del usuario. El método Simular se invoca cuando se recibe el evento del accionamiento del elemento de menú **Simular** del menú de **Herramientas** en la barra de menús.

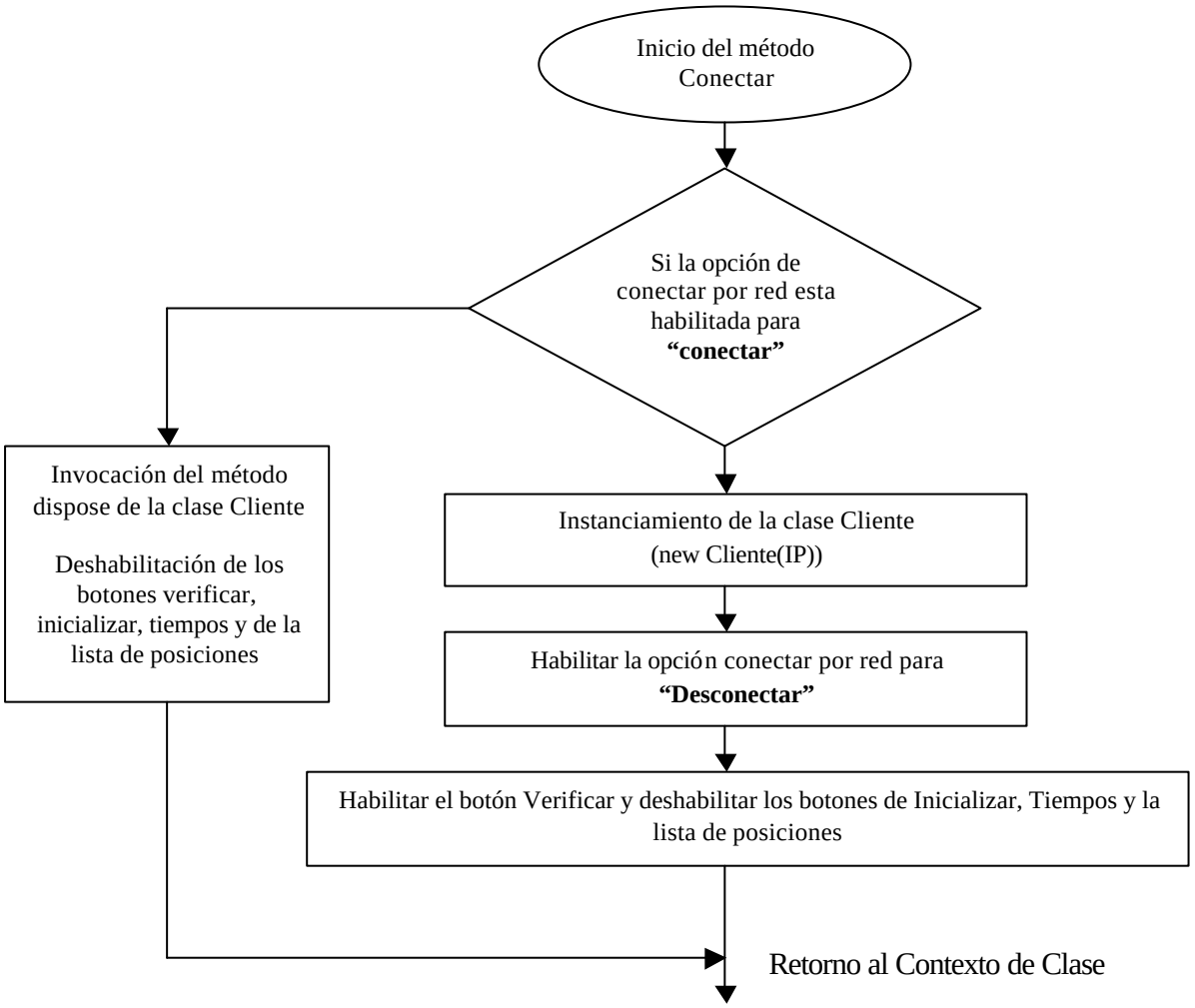


Como se menciona, la clase Simulación implementa el módulo de Simulación(MS-AF2.0) del lado del usuario. Esta clase posee métodos de inicialización de componentes gráficos, botones, tablas de tiempos etc. También ejecuta un hilo de control para llevar a cabo la simulación sin ningún problema en tiempo de ejecución respecto a otros procesos. Básicamente soporta funciones muy similares a las del módulo *swcliente* del lado del usuario pero todo referido a los datos propios de la simulación.

Método Conectar

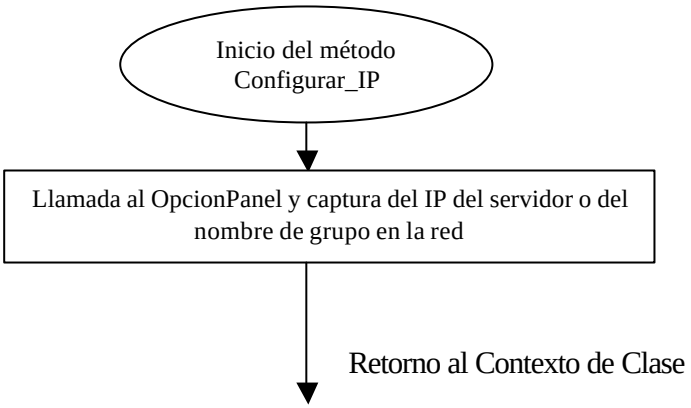
Este método se encarga de realizar la conexión a red mediante el instanciamiento de la clase Cliente. Una vez instanciada esta clase, se realiza la conexión mediante una dirección IP y un puerto especifico por el cual se establecerá el canal TCP de conexión con el servidor.

El método Conectar se invoca cuando se recibe el evento del accionamiento del elemento de menú **Conectar** del menú de **Conexión** en la barra de menús. Su diagrama de flujo es el siguiente:



Método Configurar_IP

Este método llama una ventana de opción para introducir el IP del servidor o el nombre de grupo en la red.



Como se observa en todos los diagramas, casi todos los módulos funcionales del submódulo *swcliente* se implementan en la clase swcliente a excepción de los módulos MS-AF2.0 y MAG-AF2.0 que se implementan en clases diferentes.

En este caso, el Módulo Gestor de Excepciones (GE-AF2.0) del submódulo *swcliente*, se implementa mediante todos los procedimientos y sentencias en la clase que hacen referencia a la captura de excepciones y errores en tiempo de ejecución.

Estas sentencias son los llamados try_catch y throw propios del lenguaje de programación utilizado y del entorno de ejecución de la clase. Se realizan sentencias try_catch para Procesos multihilos, captura de excepciones en el tiempo de ejecución de hilos, Conexión con el

servidor, inicialización de conexión, manejo de Streams tanto de lectura como de escritura para archivos, procesamiento grafico de los datos, operaciones matemáticas no validas etc.

Clase Cliente

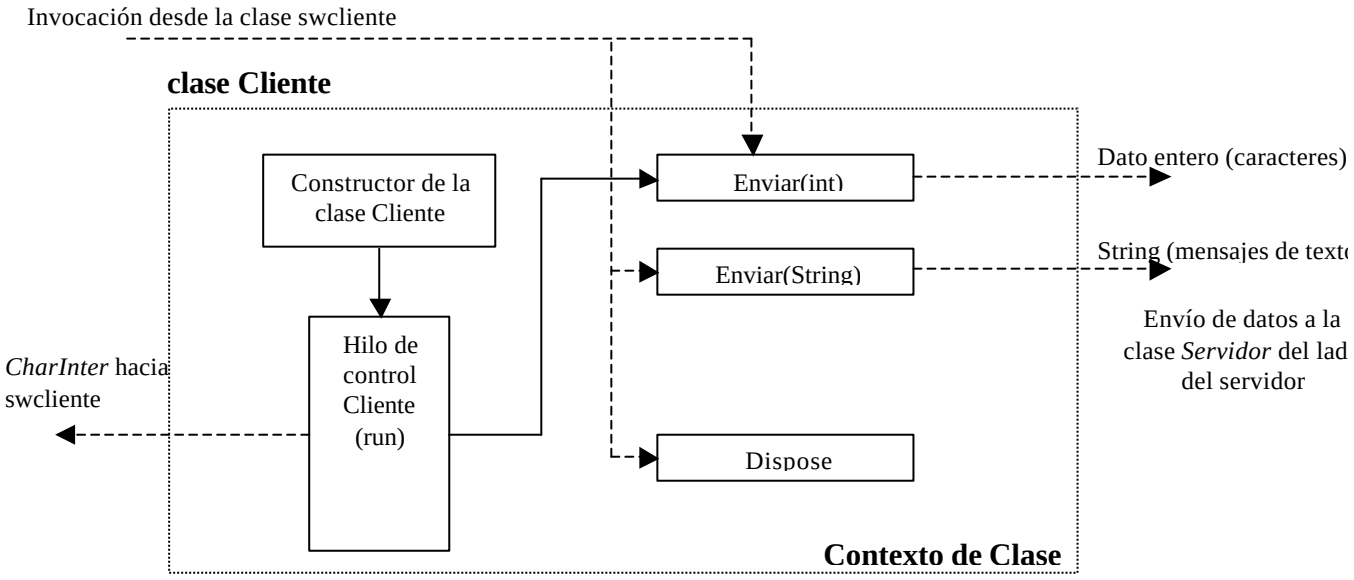
La clase Cliente corresponde a la implementación de todo el submódulo funcional Cliente del módulo SWCliente del lado del usuario. Esta clase contiene la implementación de los módulos funcionales MCC-AF2.0, GEC-AF2.0 y MTDC-AF2.0 que componen dicho submódulo funcional.

Por tanto se puede decir que la clase Cliente contiene el Ciclo de Control Cliente, la gestión de excepciones Cliente y el mecanismo de transporte de datos para el cliente.

En esta clase se realiza implementación directa del canal TCP mediante el concepto de sockets y puertos como punto final de un enlace TCP. Así, una vez establecido el canal de comunicación con el servidor, se

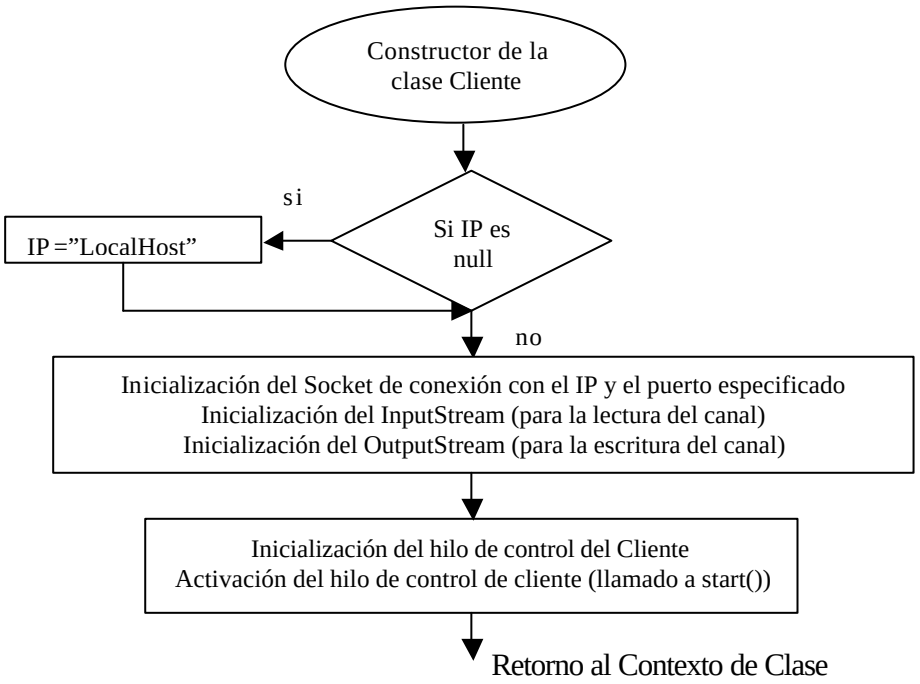
puede transferir información y recibir información mediante el uso de Transports Streams de lectura y escritura del canal.

El siguiente diagrama presenta los principales métodos de la clase mediante los cuales se implementan estos módulos funcionales.



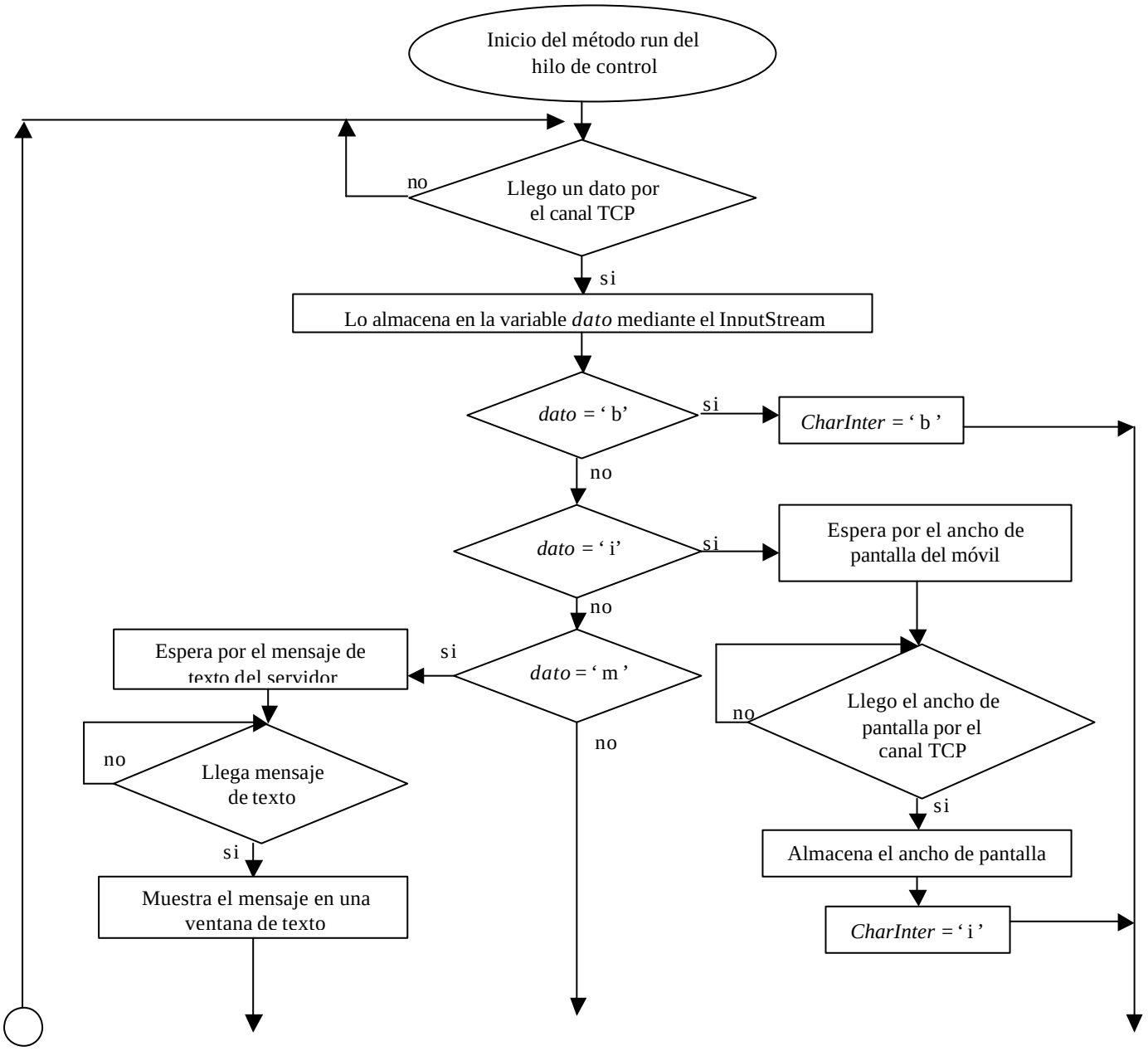
Constructor de la clase Cliente

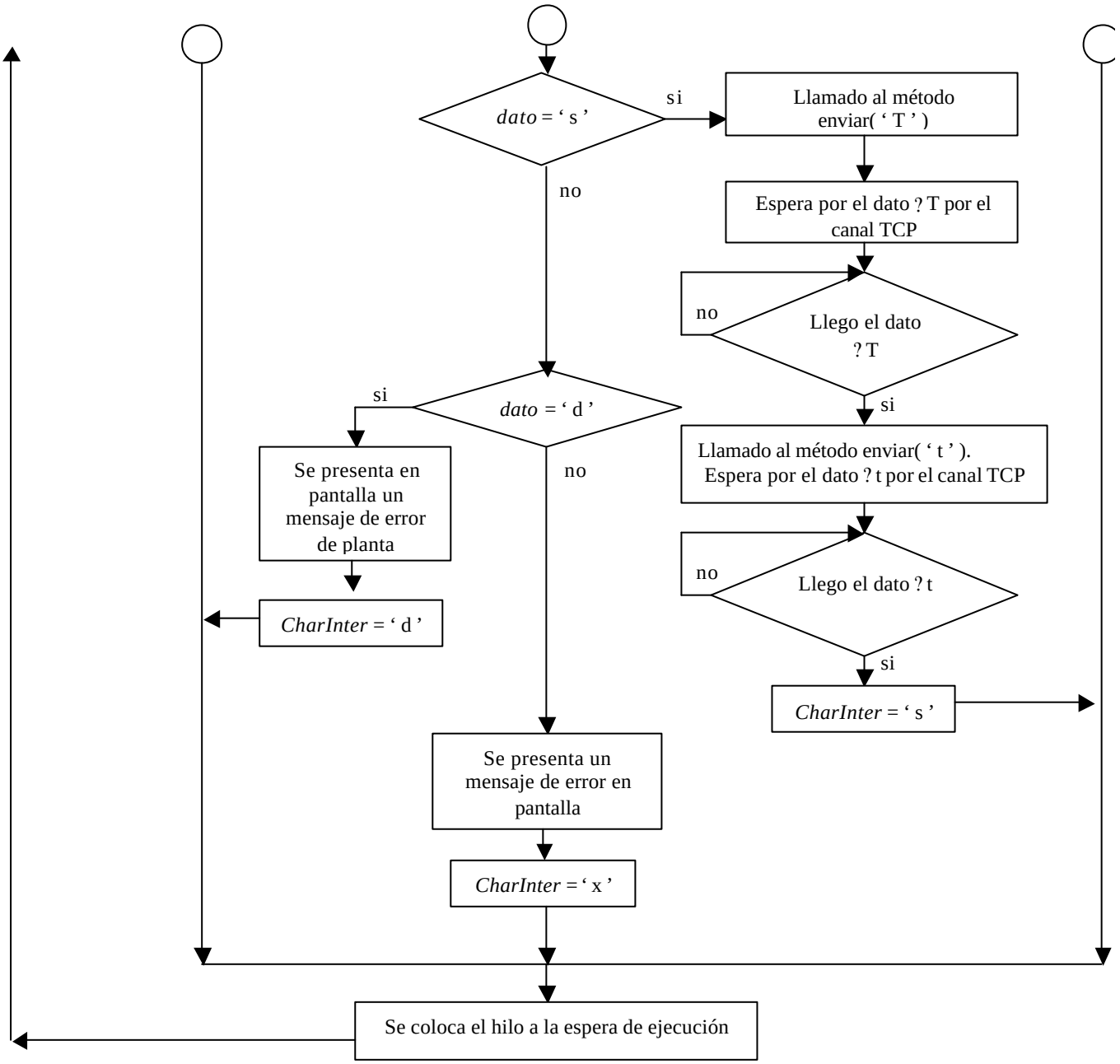
Al constructor de esta clase se le pasa como parámetro la dirección IP de servidor en el momento de su instanciamiento en la clase swcliente. A partir de allí se crea y se inicializa el punto final del canal de conexión TCP representado por un Socket y un puerto especifico. Posteriormente se inicializan los correspondientes InputStream y OutputStreams para la lectura y escritura de datos por el canal de comunicación. Finalmente se procede a activar el hilo de control para esta clase.



Método run del Hilo de Control Cliente

Este método se ejecuta una vez se haga la llamada al metodo *start()* del hilo y es el encargado de ejecutar el ciclo pricipal de la clase Cliente, el cual correponde a un ciclo continuo e infinito. Por esta razón y para evitar bloqueos del proceso principal, se ha definido este ciclo en un hilo de control. En dicho ciclo se espera la llegada de datos por el canal TCP que correponderan a las respuestas del servidor. Una vez llegue un dato, es almacenado en una variable interna y se procede realizar la comparación de dicha variable para asi entregar el valor a *CharInter* correpondiente a la respuesta que el servidor envia según la petición del cliente por el canal TCP. De esta forma, la clase swcliente realiza la encuesta y toma de desiciones con el valor de C*harInter* en su ciclo principla de control.

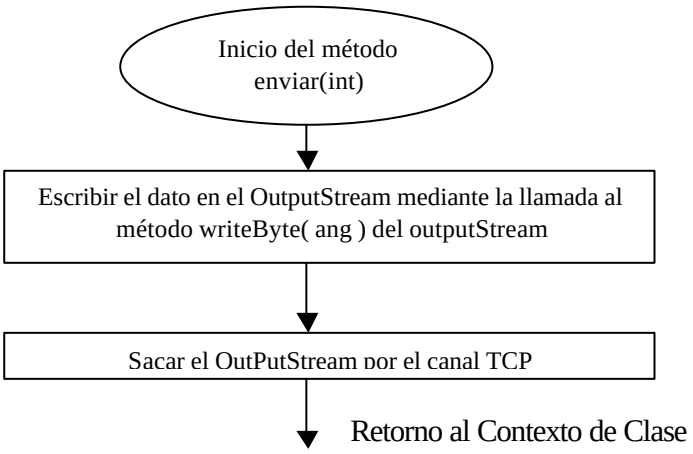

Inicio del método run del hilo de control
Llego un dato por el canal TCP
no
si
Lo almacena en la variable *dato* mediante el InputStream
dato = ‘ b’
si
CharInter = ‘ b ’
no
dato = ‘ i’
si
Espera por el ancho de pantalla del móvil
no
dato = ‘ m ’
si
Espera por el mensaje de texto del servidor
no
Llego el ancho de pantalla por el canal TCP
no
si
Almacena el ancho de pantalla
CharInter = ‘ i ’
Llega mensaje de texto
no
si
Muestra el mensaje en una ventana de texto


dato = ‘ s ’
si
Llamado al método enviar(‘ T ’)
Espera por el dato ? T por el canal TCP
no
Llego el dato ? T
no
si
dato = ‘ d ’
si
Llamado al método enviar(‘ t ’). Espera por el dato ? t por el canal TCP
Se presenta en pantalla un mensaje de error de planta
no
no
Llego el dato ? t
CharInter = ‘ d ’
si
CharInter = ‘ s ’
Se presenta un mensaje de error en pantalla
CharInter = ‘ x ’
Se coloca el hilo a la espera de ejecución

Como se observa en el diagrama, dependiendo del valor de llegada por el canal TCP establecido para la comunicación con el servidor, se realiza un proceso especifico, bien sea de esperar por un nuevo dato o presentar información en pantalla, para finalmente entregar un valor a la variable *CharInter* correspondiente a la respuesta por parte del servidor según la petición realizada por el usuario.

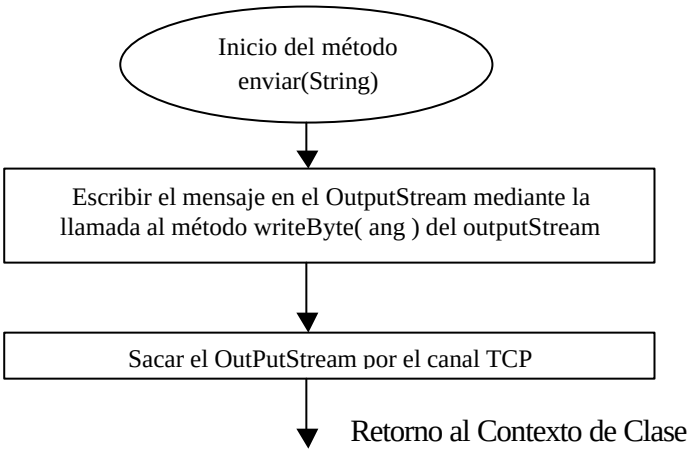
Método enviar(int)

Este método permite enviar datos enteros y caracteres al servidor a manera de peticiones por el canal TCP mediante el OutputStream definido para escribir por dicho canal.



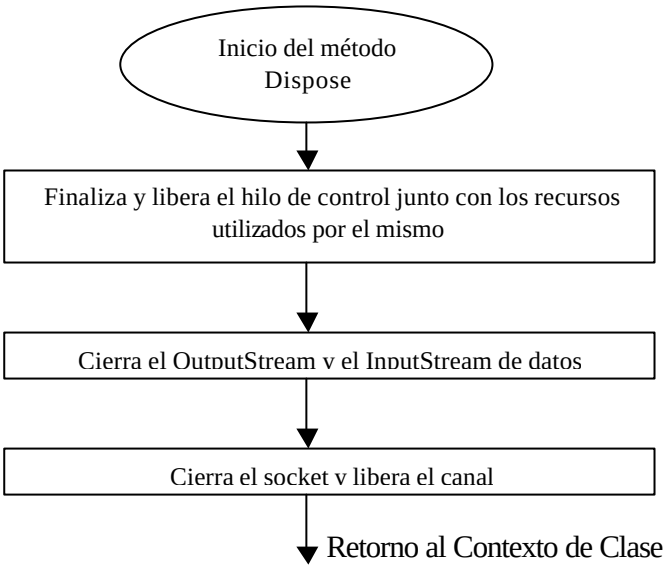
Método enviar(String)

Este método permite enviar mensajes de texto al servidor por el canal TCP mediante el OutputStream definido para escribir por dicho canal.



Método Dispose

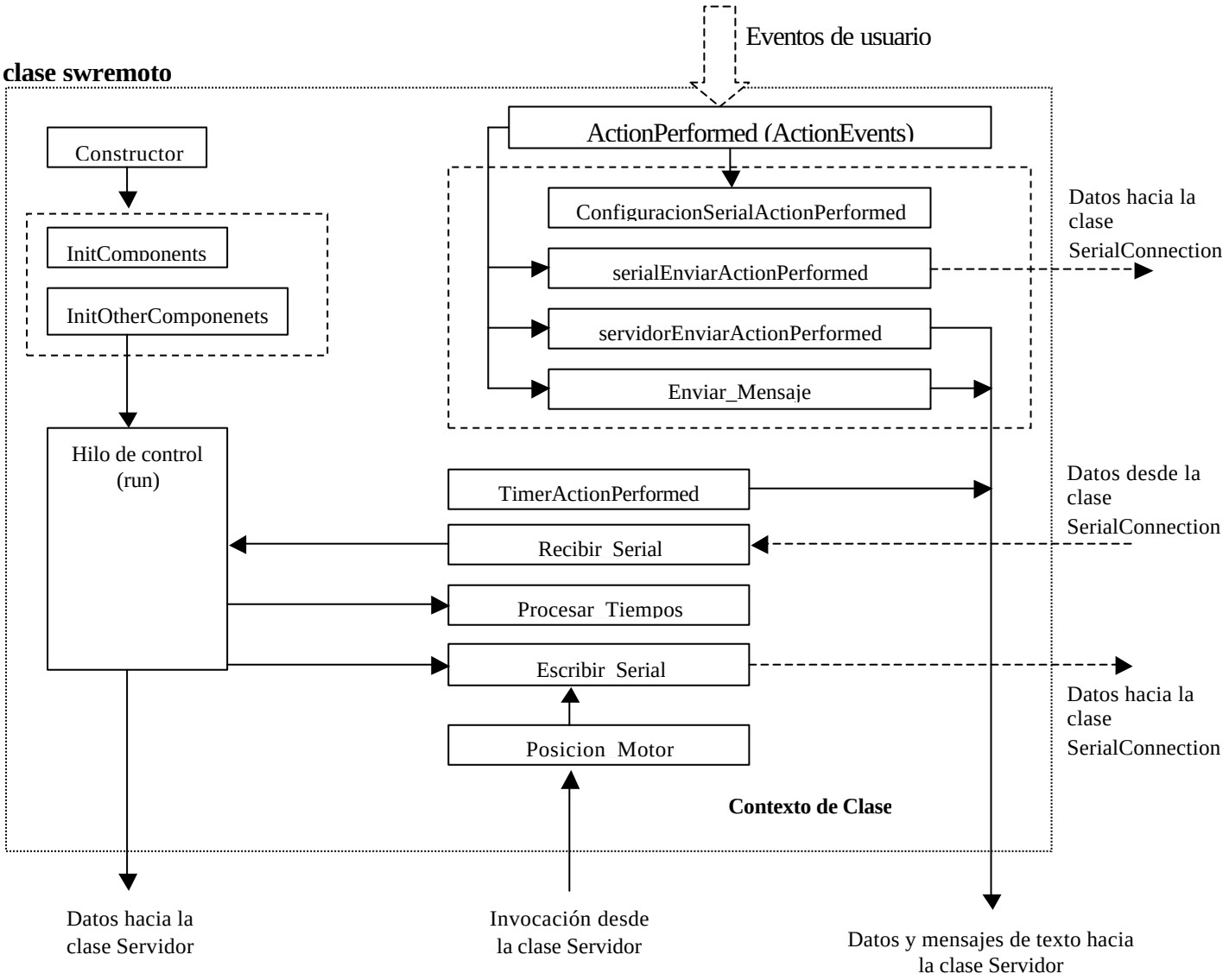
Este método se encarga de liberar el hilo de control, cerrar los Transport Stream de lectura y escritura y finalmente cerrar el Socket de comunicación con el servidor.



Clase swremoto

La clase swremoto corresponde a la implementación de todo el submódulo funcional swremoto del módulo SWServidor del lado del servidor. Esta clase contiene la implementación de los módulos funcionales IR-AF2.0, MCAR-AF2.0 y GER-AF2.0 que componen dicho submódulo funcional. El módulo MCS-AF2.0 se implementa en otra clase diferente. Por tanto se puede decir que la clase swremoto contiene la interfaz grafica para el lado del servidor, la gestión de excepciones para las operaciones y acciones propias del lado del servidor, el hilo de control que gestiona todo el control de ordenes de

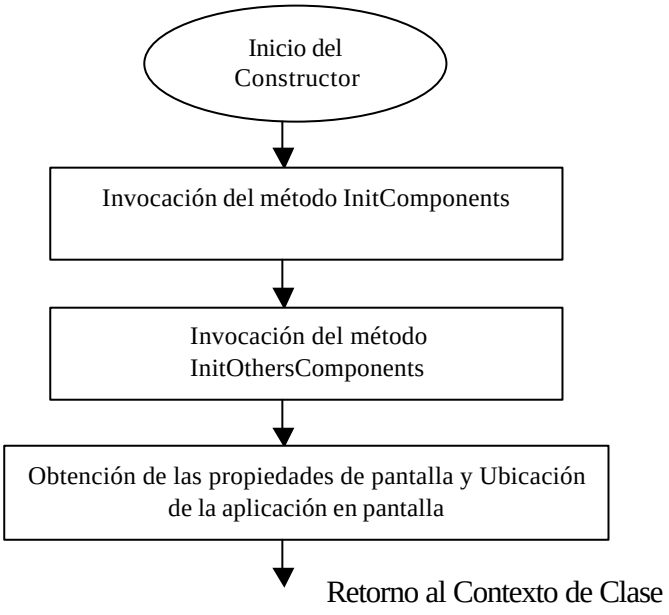
planta y el acceso al canal serial de comunicación con el hardware remoto. El siguiente diagrama presenta los principales métodos de la clase mediante los cuales se implementan estos módulos funcionales.



Constructor de la clase swremoto

Este método en particular invoca los métodos de inicialización de componentes gráficos de la interfaz de usuario y declaración e inicialización de otros componentes funcionales dentro de la implementación.

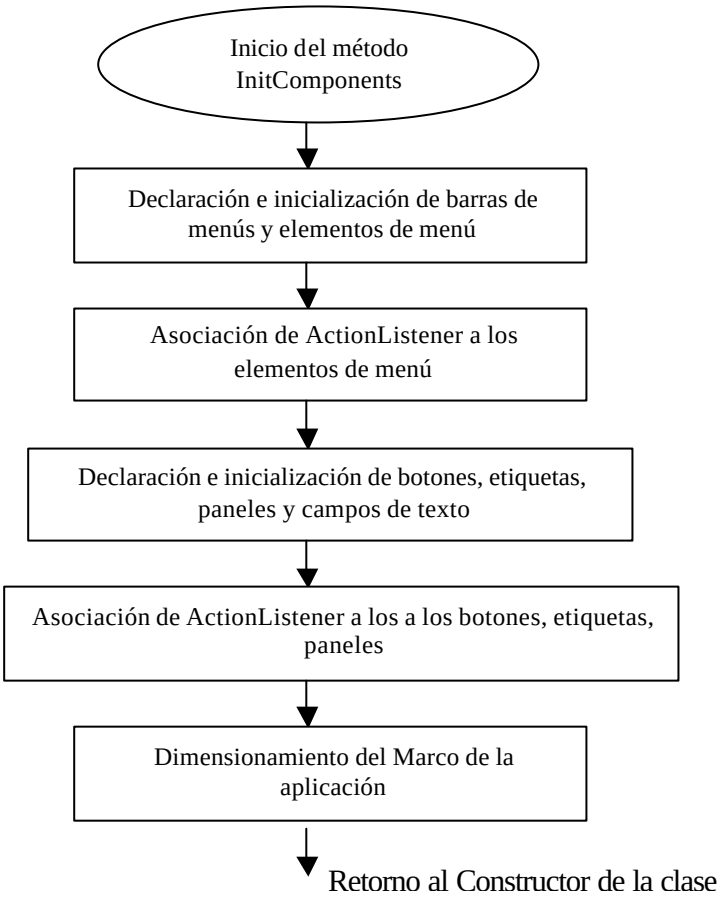
Un simple diagrama de flujo nos ilustra su funcionamiento.



Método InitComponents

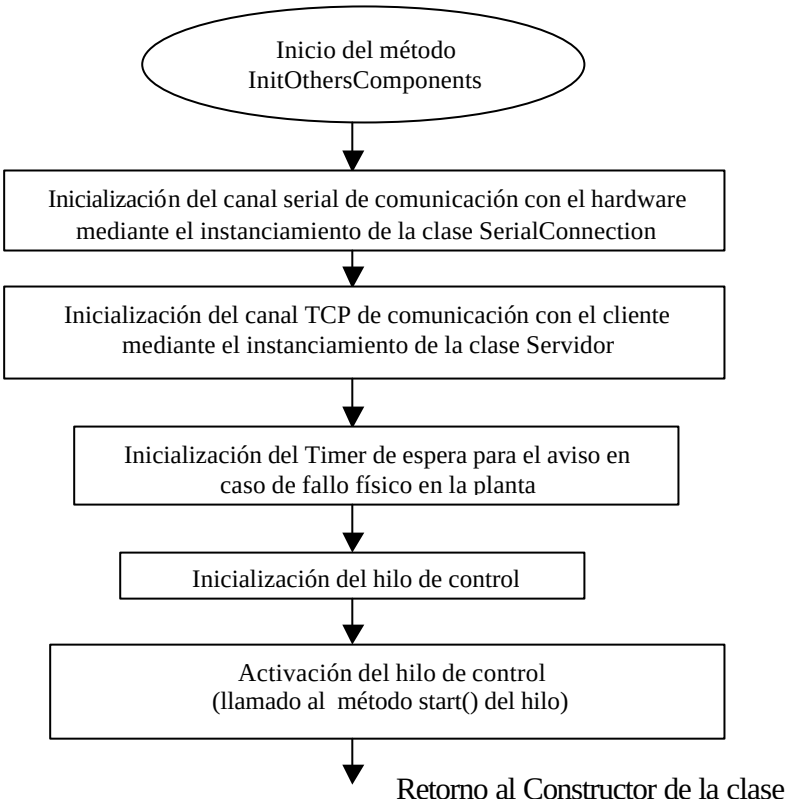
Este método permite la declaración e inicialización de todos los componentes graficos de la interfaz grafica para el lado dels servidor como son Barras de Menus, Elementos de menus, Botones, Etiquetas, Campos de texto etc.

En este método no solamente se inicializan sino que además se configuran todas las propiedades de dichos componentes graficos para su utilización y visualización en pantalla. Es através de esta interfaz que el auxiliar de laboratorio puede poner en marcha el sistema configurando la comunicación serial con el hardware remoto y comunicarse con el usuario através de mensajes de texto en caso de algun error. Ademas se permite mediante dicha interfaz, comunicar al auxiliar de laboratorio fallos en la planta mediante avisos gráficos y auditivos.


Inicio del método InitComponents
Declaración e inicialización de barras de menús y elementos de menú
Asociación de ActionListener a los elementos de menú
Declaración e inicialización de botones, etiquetas, paneles y campos de texto
Asociación de ActionListener a los a los botones, etiquetas, paneles
Dimensionamiento del Marco de la aplicación
Retorno al Constructor de la clase

Método InitOthersComponents

Este método permite la inicialización de otros componentes funcionales del programa como son el canal serial de comunicación con el hardware, el canal TCP para el lado del servidor, inicialización del Timer de espera para evitar casos de inanición y espera de respuestas cuando ocurre un error físico en la planta, inicialización del vector interno para almacenar los datos de tiempos contenidos en el Stream que el hardware envía cuando ha terminado de obtener los tiempos del fenómeno y finalmente la inicialización y activación del hilo de control que se encargará de realizar acciones según la información que el hardware devuelva al servidor despues de ejecutar con éxito alguna orden de control.

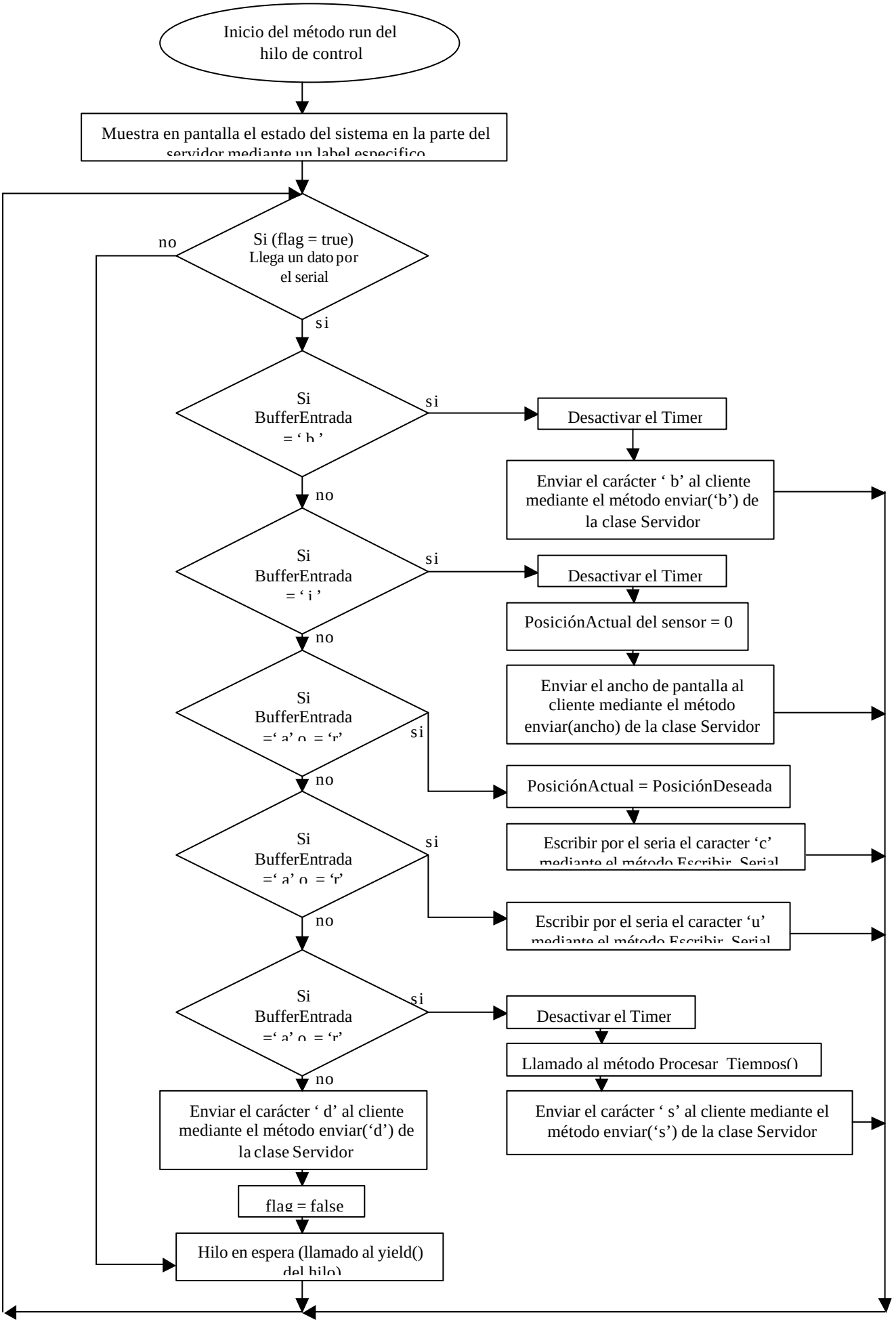


Método run del hilo de control

Este método se ejecuta una vez se haga la llamada al metodo *start()* del hilo y es el encargado de ejecutar el ciclo pricipal del programa de control de planta , el cual correponde a un ciclo continuo e infinito. Por esta razón y para evitar bloqueos del proceso principal, se ha definido este ciclo en un hilo de control con un tiempo de espera de ejecucion

definido por el monitor de hilos del entorno de ejecución de la clase(en este caso el entorno de ejecución de JAVA JVM). En dicho ciclo se evalua constantemente un *flag* que indica si ha llegado un dato por el canal serial a manera de evento. Si esto sucede, entonces se evalua que dato ha llegado por el canal serial de comunicación con la planta y de acuerdo al dato que llegue se ejecuta una acción especifica, bien sea enviar una respuesta al cliente de orden ejecutada con éxito o bien sea reenviar una nueva orden de control a la planta.

En terminos generales evalua que dato envio la planta como respuesta a una orden de control si se recibe el evento de llegada de un dato por el canal serial de comunicación establecido entre el servidor y la planta.

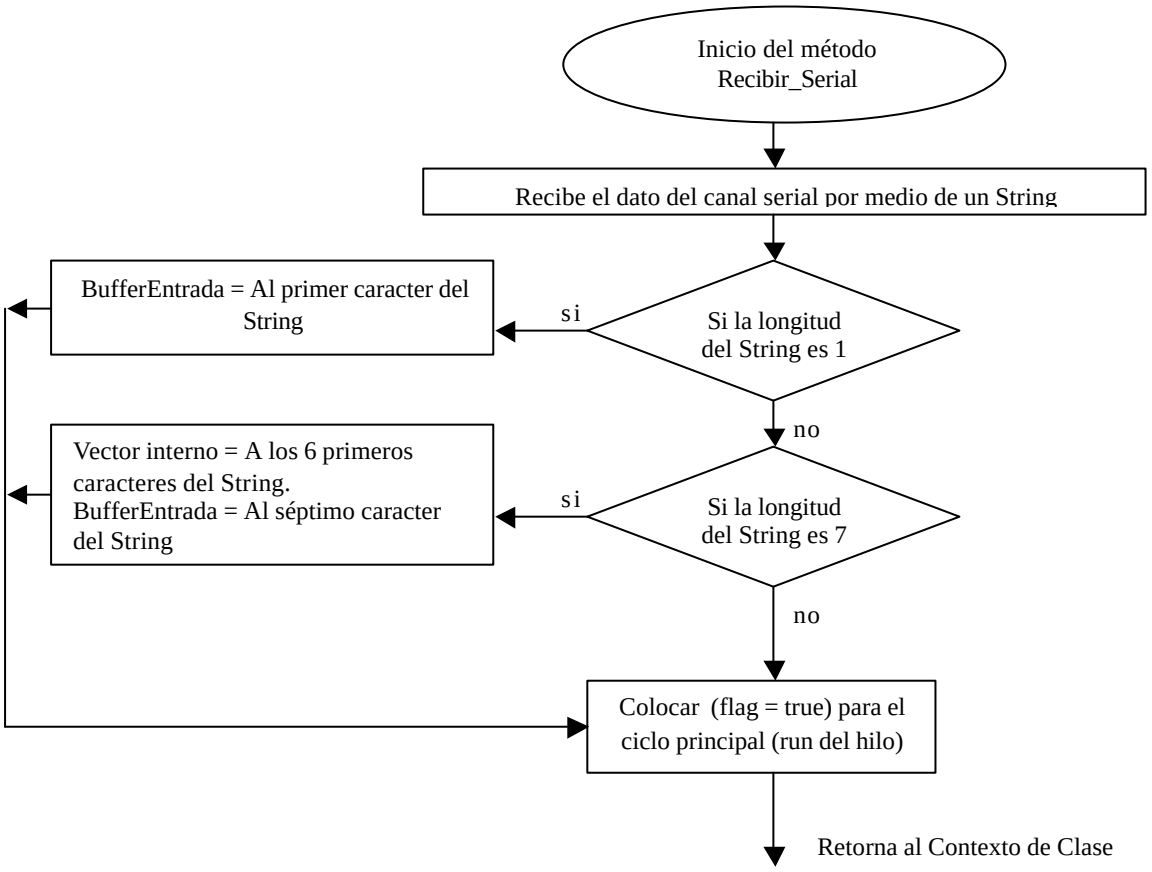

Inicio del método run del hilo de control
Muestra en pantalla el estado del sistema en la parte del servidor mediante un label especifico
Si (flag = true) Llega un dato por el serial
no
si
Si BufferEntrada = ‘ b ’
si
Desactivar el Timer
Enviar el carácter ‘ b’ al cliente mediante el método enviar(‘b’) de la clase Servidor
no
Si BufferEntrada = ‘ i ’
si
Desactivar el Timer
PosiciónActual del sensor = 0
Enviar el ancho de pantalla al cliente mediante el método enviar(ancho) de la clase Servidor
no
Si BufferEntrada =‘ a’ o = ‘r’
si
PosiciónActual = PosiciónDeseada
Escribir por el seria el caracter ‘c’ mediante el método Escribir_Serial
no
Si BufferEntrada =‘ a’ o = ‘r’
si
Escribir por el seria el caracter ‘u’ mediante el método Escribir_Serial
no
Si BufferEntrada =‘ a’ o = ‘r’
si
Desactivar el Timer
Llamado al método Procesar_Tiempos()
Enviar el carácter ‘ s’ al cliente mediante el método enviar(‘s’) de la clase Servidor
no
Enviar el carácter ‘ d’ al cliente mediante el método enviar(‘d’) de la clase Servidor
flag = false
Hilo en espera (llamado al yield() del hilo)

Como se observa en el diagrama, existe una variable interna de clase llamada *BufferEntrada*, en la cual se almacena el valor del dato que llega por el canal serial proveniente del hardware remoto. El almacenamiento del dato en la variable *BufferEntrda* se realiza en el método Recibir_Serial() el cual es invocado cada vez que ocurre un evento de llegada de datos por el canal serial. Un dato llega por el canal serial cuando la planta responde que ha ejecutado una orden con éxito. De esta manera, dicho dato es analizado en el ciclo anterior.

Método Recibir_Serial

Este método se encarga de recibir los datos que llegan por el canal serial implementado en la clase *SerialConnection*. Una vez llega un dato por el canal serial, se invoca este método desde la clase *SerialConnection* y se le transfiere el dato para ser almacenado en la variable B*ufferEntrada* como se menciono anteriormente. En caso de que el hardware envíe por el canal serial un Stream de Datos que contenga los valores de tiempos calculados por el sistema

microcontrolador, dichos datos se almacenarán en un vector interno, para luego ser procesado en el método Procesar_Tiempos().



Método Procesar_Tiempos

Este método se encarga de procesar el vector interno en el cual se encuentran almacenados los datos de tiempo del fenómeno calculados por el hardware y enviados por el canal serial. Básicamente los valores de tiempos que llegan por el canal serial son 6 datos hexadecimales contenidos en un Stream. Los tres primeros datos constituyen los valores del número de desbordamientos del timer del microcontrolador para ?T y los valores de registros TH0,TL0 para ?T respectivamente. Los tres últimos valores corresponden al número de desbordamientos para ?t y los valores de registros TH0, TL0 para ?t respectivamente.

De esta forma se realiza la conversión de dichos valores a un equivalente decimal para ?T y ?t del fenómeno.

La operación a realizar es la siguiente:

CM = 65535*Vect[0] +256*Vect[1] +Vect[2]

?**T** = $(1.08x10^{-6}$ seg) * CM

Donde **CM** es número de ciclos de maquina del microcontrolador.

? **T** es el valor decimal del tiempo medio del fenómeno.

El valor $1.08x10^{-6}$ se obtiene del calculo del tiempo equivalente a un ciclo de maquina del microcontrolador. Vect[0], Vect[1], Vect[2] son el número de desbordamientos del timer del microcontrolador para ?T y los valores de registros TH0,TL0 para ?T respectivamente almacenados en las tres primeras posiciones del vector interno.

La misma operación se realiza para obtener ?t del fenómeno (tiempo instantáneo).

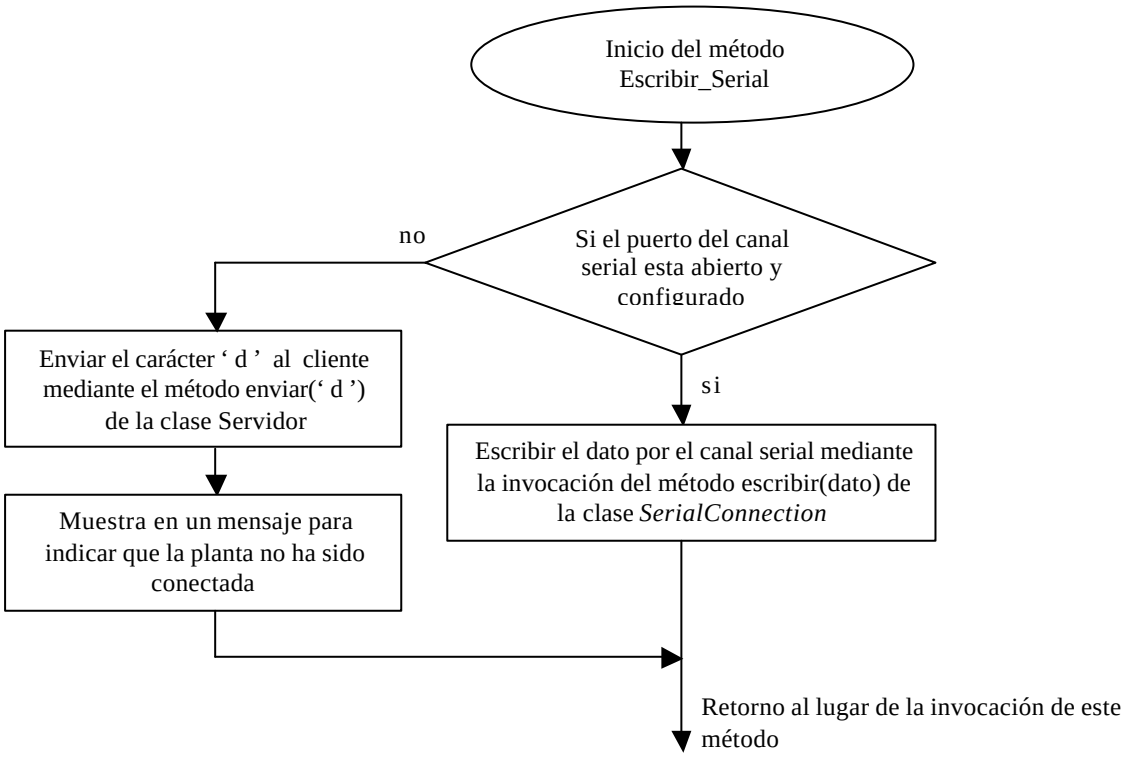
CM = 65535*Vect[3] +256*Vect[4] +Vect[5]

? **t** = $(1.08x10^{-6}$ seg) * CM

Donde Vect[3], Vect[4], Vect[5] son el número de desbordamientos del timer del microcontrolador para ?t y los valores de registros TH0,TL0 para ?t respectivamente almacenados en las posiciones 4, 5 y 6 del vector interno.

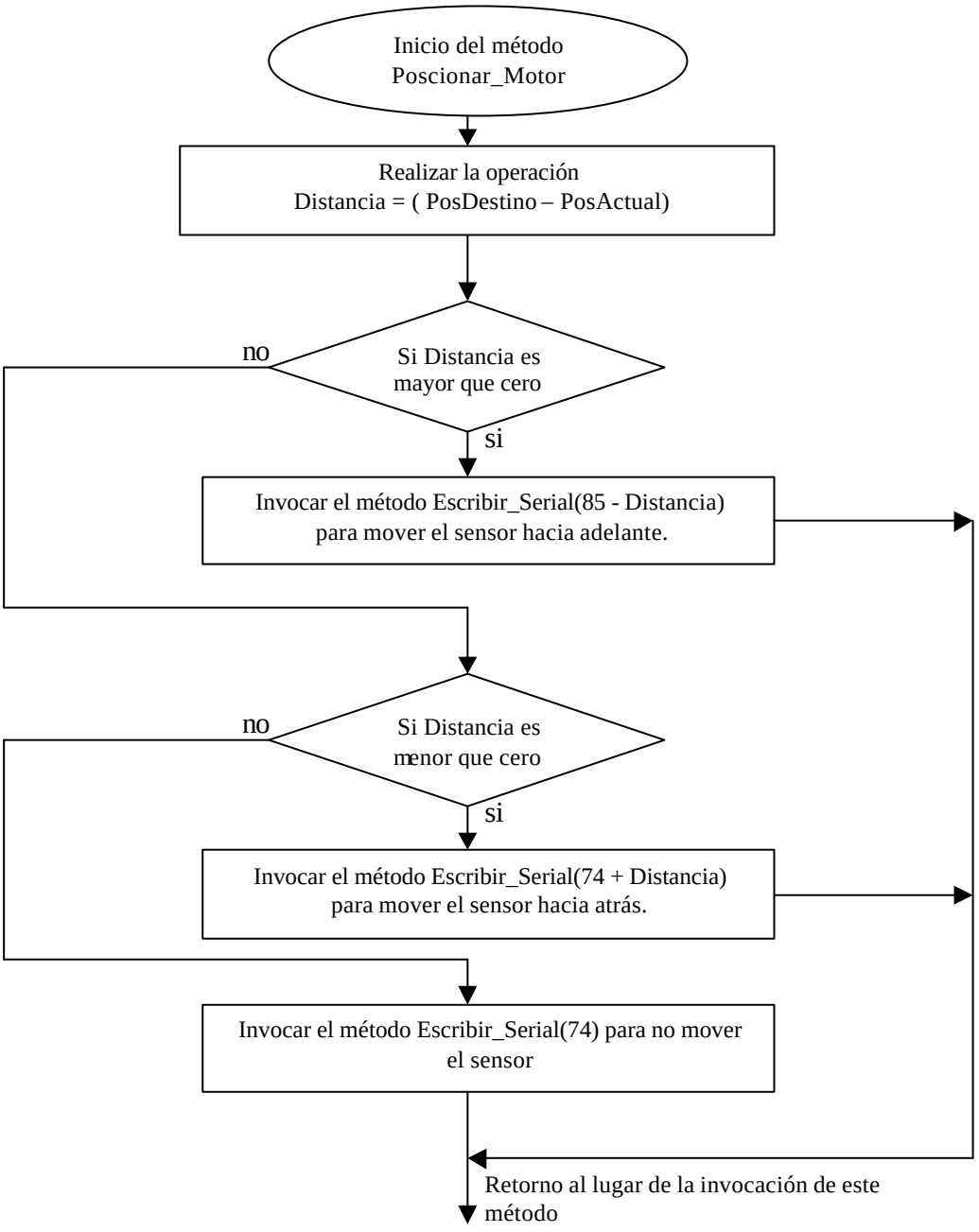
Método Escribir_Serial

Mediante este método la clase swremoto le envía ordenes a la planta. Básicamente este método se encarga de invocar el método escribir de la clase *SerialConnection* la cual a su vez envía el dato directamente por el canal serial mediante un OutputStream.



Método Posicionar_Motor

A través de este método se calcula la distancia relativa que se debe mover el sensor de tiempos para ubicarlo en una posición deseada por el usuario. Como se menciono en la descripción del funcionamiento del hardware y en la descripción del algoritmo de control para el hardware, se establecieron dos rangos de movimientos para sensor, para adelantar y para retroceder el sensor en la línea del carril de aire. Para lograr ubicar el sensor en una posición deseada, se debe mantener almacenada la posición actual del sensor y restar esta posición a la posición deseada y así logramos obtener una un valor de distancia relativa para mover el sensor hacia adelante o hacia atrás dependiendo de si este valor de distancia es positivo o negativo.

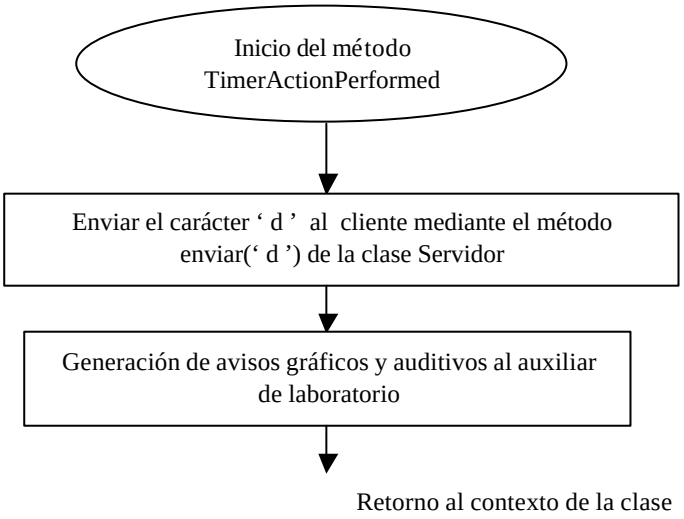

Inicio del método
Poscionar_Motor
Realizar la operación
Distancia = (PosDestino – PosActual)
Si Distancia es
mayor que cero
no
si
Invocar el método Escribir_Serial(85 - Distancia)
para mover el sensor hacia adelante.
Si Distancia es
menor que cero
no
si
Invocar el método Escribir_Serial(74 + Distancia)
para mover el sensor hacia atrás.
Invocar el método Escribir_Serial(74) para no mover
el sensor
Retorno al lugar de la invocación de este
método

Como se observa en este diagrama, el dato que se envía por el serial para la ubicación del sensor en la posición deseada no es solamente la distancia relativa obtenida. Si no que se envía el resultado de la operación (85d – Distancia) para avanzar el sensor y (74d - Distancia) para retroceder el sensor. Esto para efectos de compatibilidad con el algoritmo de control del hardware. Para mayor información referirse a la descripción del algoritmo de control para el hardware.

Método TimerActionPerformed

Este método es invocado por el oyente de eventos cuando se termina el tiempo de espera para el Timer utilizado en el software para evitar los casos de inanición en caso de alguna falla física en la planta. Como se observará en el diagrama del ciclo de control de la clase Servidor, el Timer se activa cada vez que llega una orden del cliente para interactuar con la planta. Una vez la planta ejecute con éxito una orden de control, el Timer se desactiva en el ciclo de control de la clase *swremoto* como se observa en su correspondiente diagrama de flujo. En caso de que la planta no pueda ejecutar con éxito una orden por causa de alguna falla

física, el evento del Timer se disparará después de cumplido su tiempo de espera, es aquí cuando el oyente de este evento dentro del entorno de ejecución de la clase *swremoto* invoca el método TimerActionPerformed el cual enviará al cliente una respuesta de falla en la planta y lanzará avisos gráficos y auditivos al auxiliar de laboratorio para que tome las correspondientes acciones.



Método ActiónPerformed

Este método es invocado y ejecutado cuando el oyente de acciones y eventos de los componentes de interfaz recibe eventos procedentes de acciones del mouse o del teclado a través del manejador de ventana y el adaptador de teclado del sistema. Invoca a los métodos ConfiguracionSerialActionPerformed, serialEnviarActionPerformed, servidorEnviarActionPerformed y Enviar_Mensaje.

Método ConfiguracionSerialActionPerformed

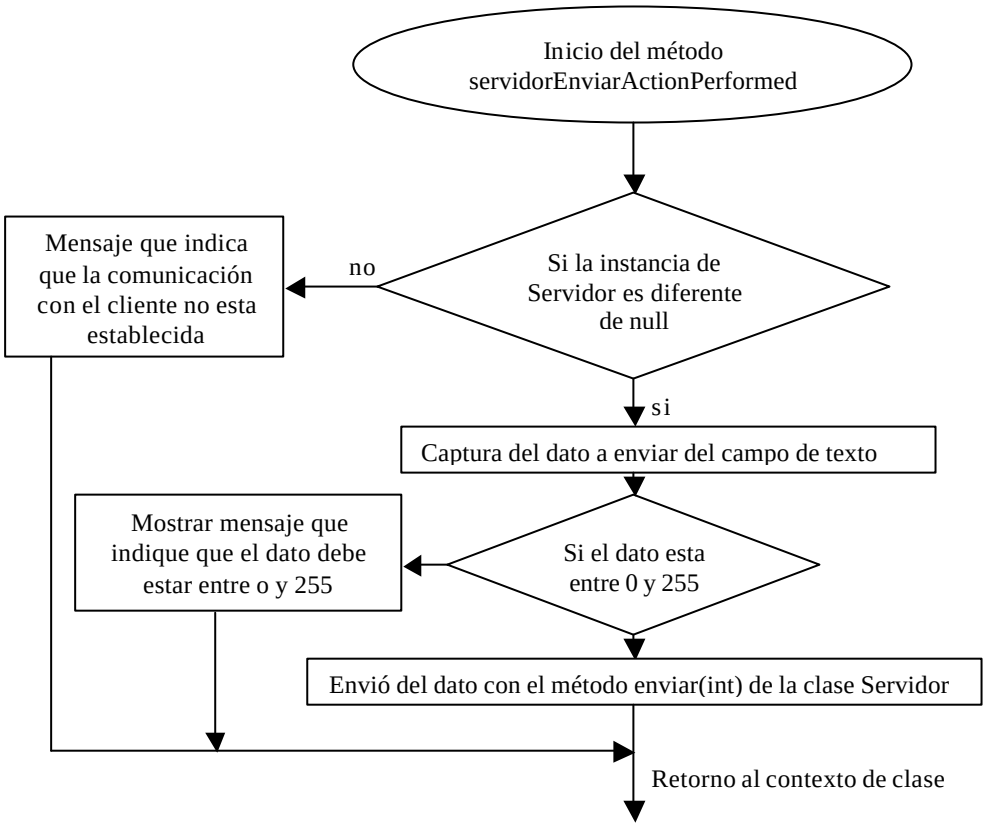
Este método se llama una vez se accione la opción **Planta** del menú de **Conexión** de la barra de menús de la interfaz grafica. Básicamente se encarga de mostrar en pantalla un marco de configuración para el canal serial. Esto lo hace mediante los métodos setVisible() de la clase SerialDemo. La clase SerialDemo no se explicará pues solo basta saber que es la encargada de instanciar la clase *SerialConnection,* enviarle los parámetros de configuración del canal e invocar sus método directamente para enviar y recibir datos por el canal. Es por esta razón

que se explicará la clase *SerialConnection* únicamente para nuestros propósitos descriptivos.

Método servidorEnviarActionPerformed

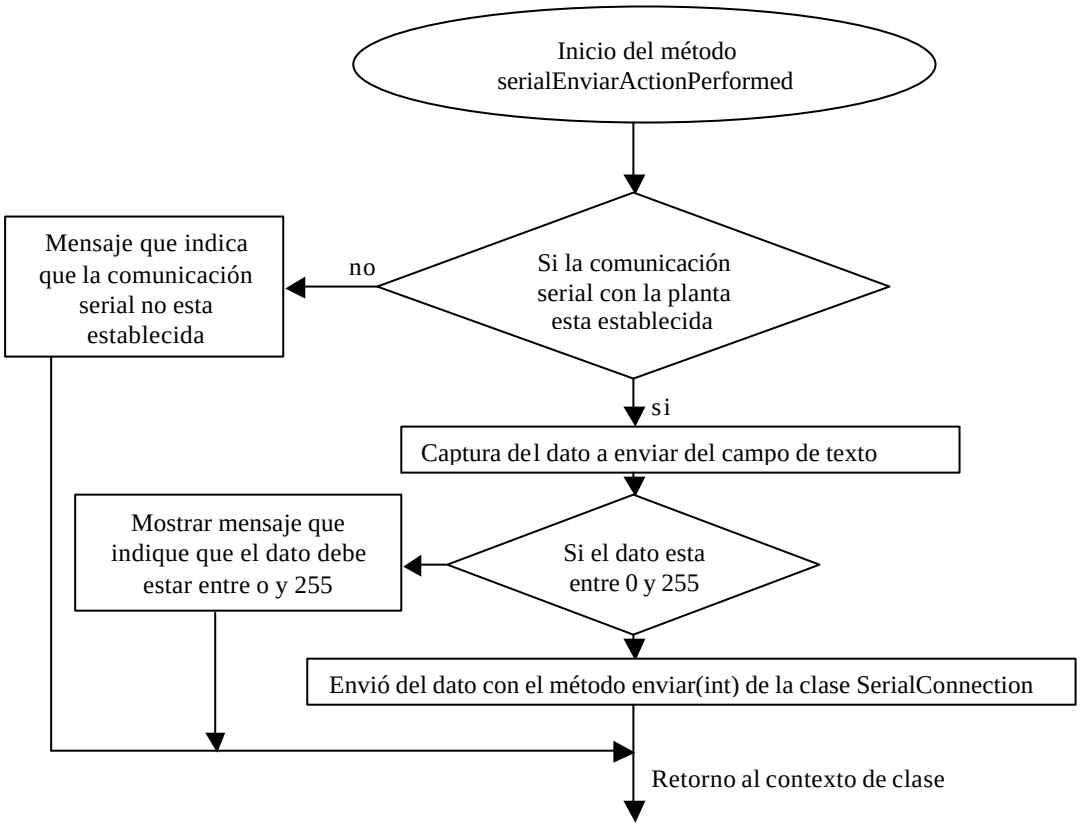
Este método se encarga de enviar un dato al cliente mediante la invocación del método enviar(int) de la clase Servidor. Lo particular es que el dato a enviar se obtiene de un campo de texto de la interfaz de usuario. Esto se ha realizado básicamente para situaciones de testeo en la comunicación con el cliente y en casos en que el auxiliar tenga que enviar un dato al cliente de forma manual por causa de algún error.

El método servidorEnviarActionPerformed se invoca cuando se ejecuta la acción del botón **Enviar** en la sección de enviar datos al cliente de la interfaz.



Método serialEnviarActionPerformed

Mediante este método se envían datos al hardware por el canal serial. El método serialEnviarActionPerformed se invoca cuando se ejecuta la acción del botón **Enviar** en la sección de enviar datos por el serial de la interfaz grafica.



Método Enviar_Mensaje

Mediante este método se envía un mensaje de texto al cliente. Al igual que el método Enviar_mensaje del cliente, este método recibe el mensaje de texto desde un JoptionPanel que aparece en la interfaz

grafica, envía primero por el canal TCP el carácter ‘ m ’ y luego el mensaje de texto. Es invocado cuando se ejecuta la acción de la opción **Enviar Mensaje** del menú de **Conexión** de la barra de menús.

De esta manera damos por especificada la clase *swremoto* que implementa gran parte del Submódulo **swremoto** del módulo **SWServidor** en el modelo funcional.

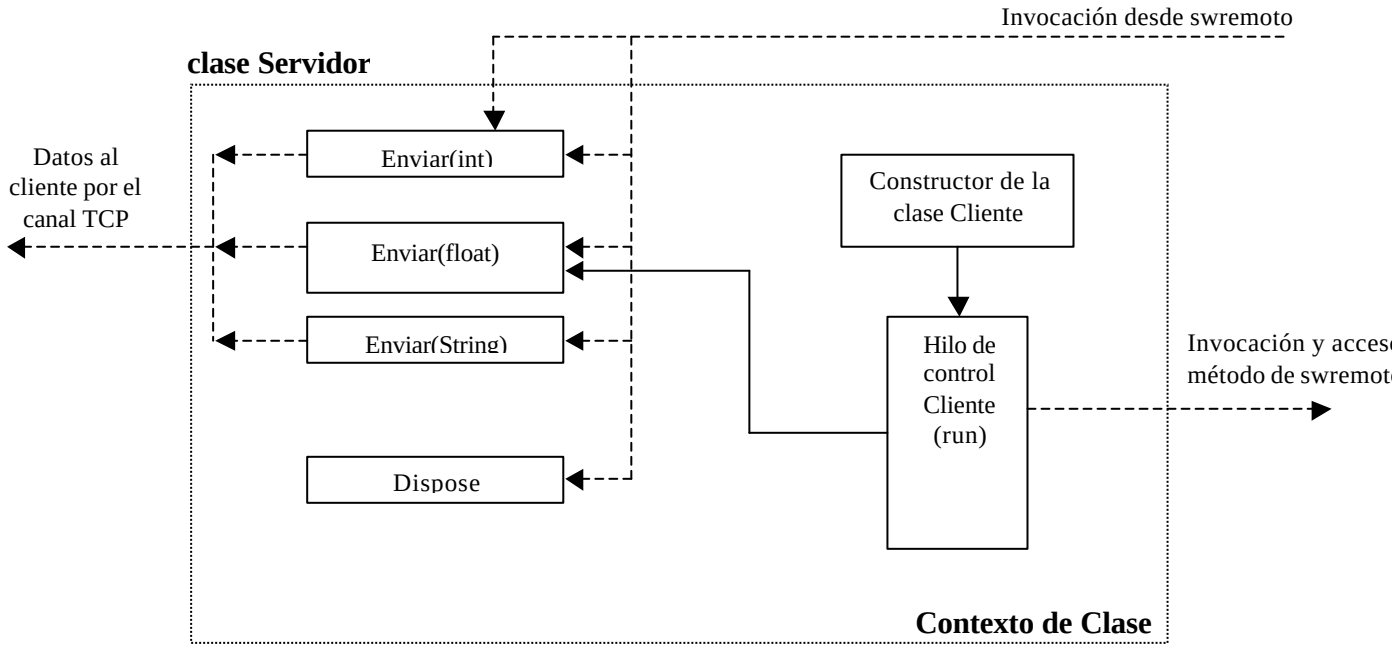
Clase Servidor

La clase Servidor corresponde a la implementación de todo el submódulo funcional Servidor del módulo SWServidor en el modelo funcional. Esta clase contiene la implementación de los módulos funcionales MCS-AF2.0, MTDS-AF2.0 y GES-AF2.0 que componen dicho submódulo.

Por tanto se puede decir que la clase Servidor contiene el Ciclo de Control Servidor, la Gestión de Excepciones Servidor y el mecanismo de transporte de datos para el Servidor.

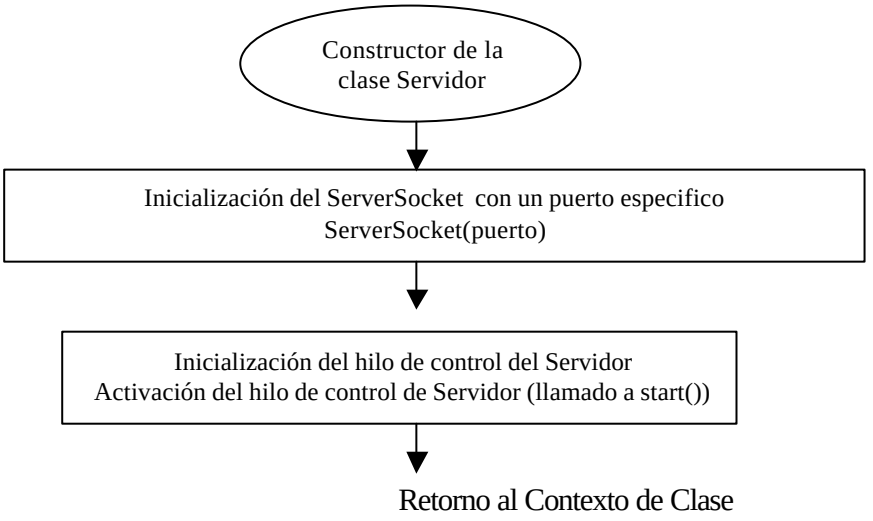
En esta clase se realiza la implementación directa del canal TCP mediante el concepto de ServerSocket, Sockets y puertos como punto final de un enlace TCP. Así, una vez establecido el canal de comunicación con el cliente, se puede transferir información y recibir información mediante el uso de Transports Streams de lectura y escritura del canal.

El siguiente diagrama presenta los principales métodos de la clase mediante los cuales se implementan estos módulos funcionales.



Constructor de la clase Servidor

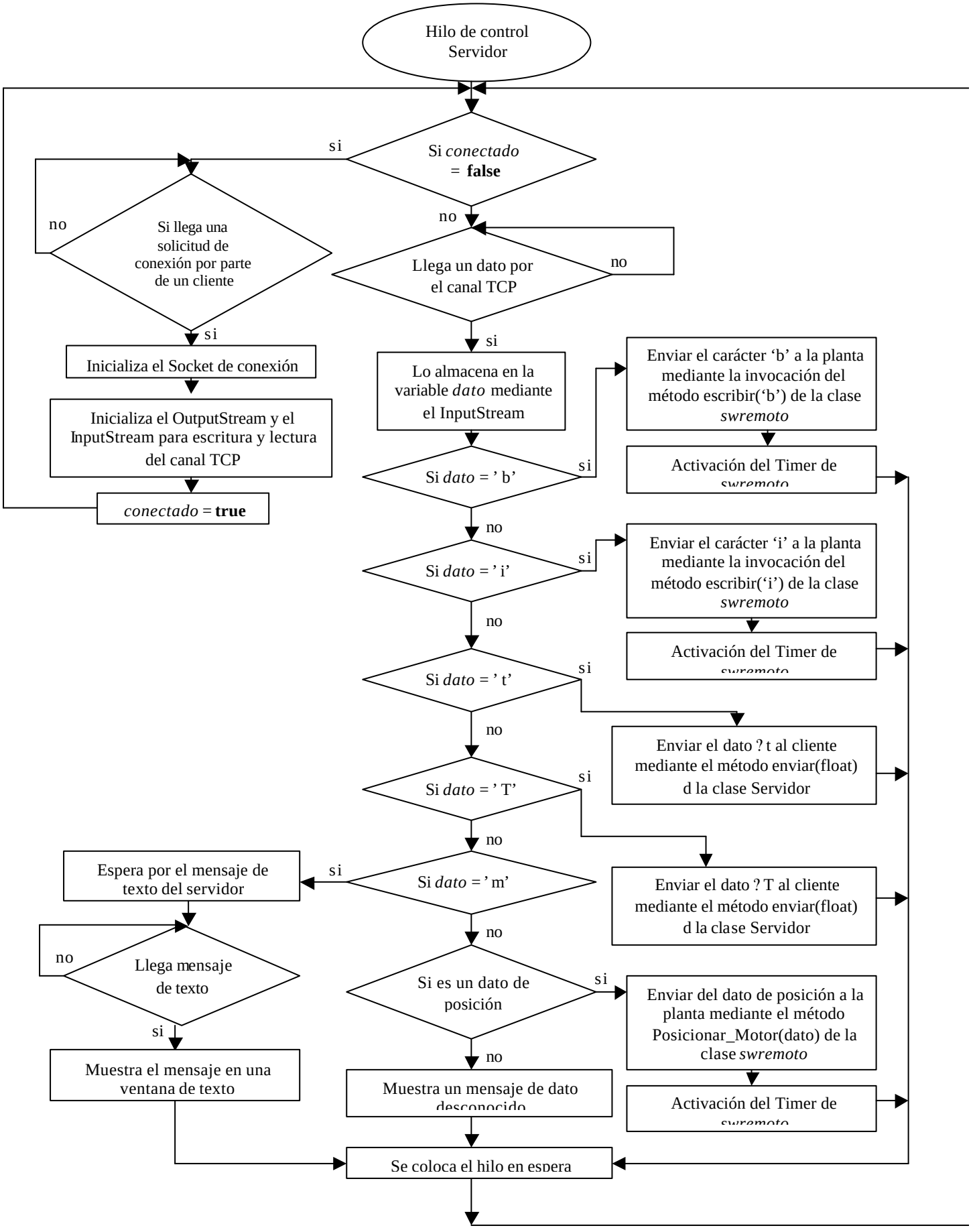
El constructor de esta clase inicializa un ServerSocket con un puerto asociado por el cual escuchará las peticiones de conexión de clientes. A su vez, este constructor se encarga de inicializar el hilo de control Servidor.



Método run del Hilo de Control Servidor

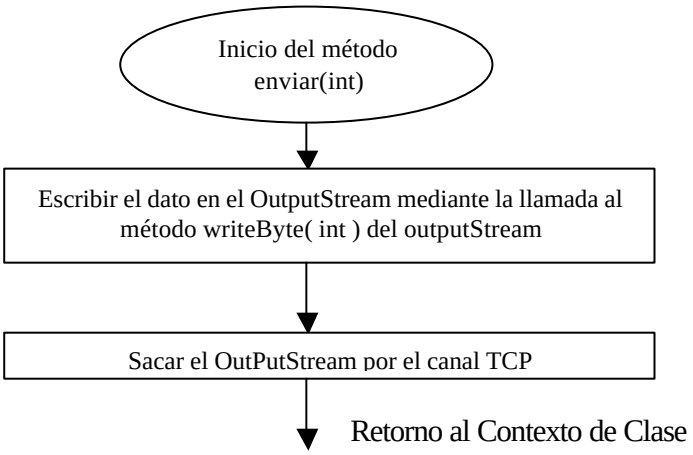
Este método se ejecuta una vez se haga la llamada al metodo *start()* del hilo y es el encargado de ejecutar el ciclo pricipal de la clase Servidor, el cual correponde a un ciclo continuo e infinito. Por esta razón y para evitar bloqueos del proceso principal, se ha definido este ciclo en un hilo de control. En dicho ciclo se espera a la llegada de datos por el canal TCP que correponderan a las peticiones del cliente. Una vez llegue un dato es almacenado en una variable interna y se procede realizar la comparación de dicha variable para poder enviar una orden a la planta mediante el método de escritura por el canal serial de la clase *swremoto.* En terminos generales, esta clase espera por las peticiones del cliente y de acuerdo a la petición del cliente se procede a enviar la correspondiente orden de control a la planta.

Dentro del ciclo se utiliza una variable booleana llamada *conectado* para reconocer si existe continuamente una conexión con el cliente, de esta forma si se presenta alguna exepción se reestablecen automaticamente el Socket de comunicación y los Transports Streams de lectura y escritura por el canal.


Hilo de control Servidor
Si *conectado* = **false**
si
no
Si llega una solicitud de conexión por parte de un cliente
no
si
Inicializa el Socket de conexión
Inicializa el OutputStream y el InputStream para escritura y lectura del canal TCP
conectado = **true**
Llega un dato por el canal TCP
no
si
Lo almacena en la variable *dato* mediante el InputStream
Si *dato* = ' b'
si
Enviar el carácter 'b' a la planta mediante la invocación del método escribir('b') de la clase *swremoto*
Activación del Timer de swremoto
no
Si *dato* = ' i'
si
Enviar el carácter 'i' a la planta mediante la invocación del método escribir('i') de la clase *swremoto*
Activación del Timer de swremoto
no
Si *dato* = ' t'
si
Enviar el dato ? t al cliente mediante el método enviar(float) d la clase Servidor
no
Si *dato* = ' T'
si
Enviar el dato ? T al cliente mediante el método enviar(float) d la clase Servidor
no
Si *dato* = ' m'
si
Espera por el mensaje de texto del servidor
no
Llega mensaje de texto
si
Muestra el mensaje en una ventana de texto
no
Si es un dato de posición
si
Enviar del dato de posición a la planta mediante el método Posicionar_Motor(dato) de la clase *swremoto*
Activación del Timer de swremoto
no
Muestra un mensaje de dato desconocido
Se coloca el hilo en espera

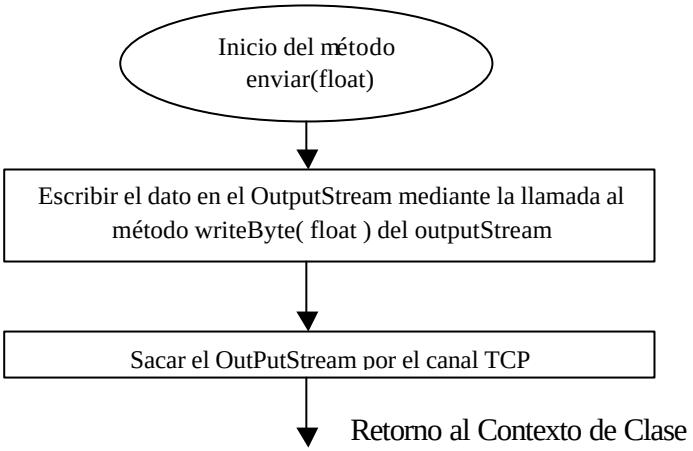
Método enviar(int)

Este método permite enviar datos enteros y caracteres al cliente a manera de respuestas por el canal TCP mediante el OutputStream definido para escribir por dicho canal.



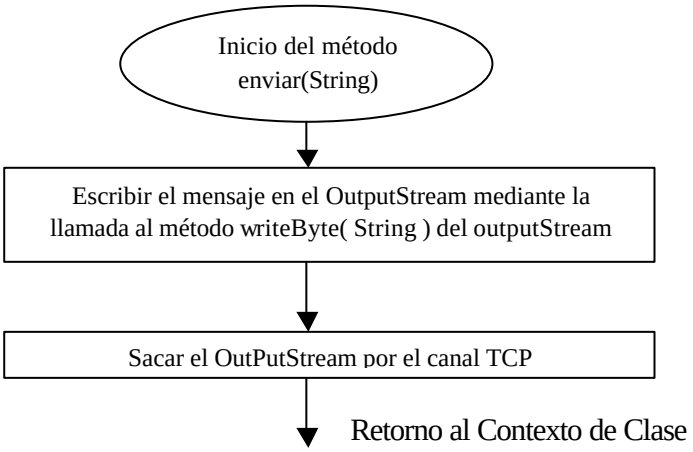
Método enviar(float)

Este método permite enviar datos decimales (punto flotante) al cliente a manera de respuestas por el canal TCP mediante el OutputStream definido para escribir por dicho canal. Es utilizado para enviar los datos de tiempos y el ancho de la pantalla del movil.



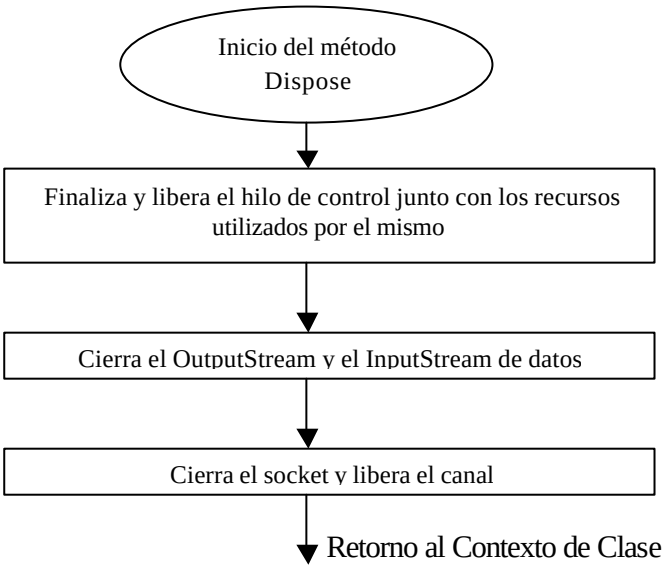
Método enviar(String)

Este método permite enviar mensajes de texto al cliente por el canal TCP mediante el OutputStream definido para escribir por dicho canal.



Método Dispose

Este método se encarga de liberar el hilo de control, cerrar los Transport Stream de lectura y escritura y finalmente cerrar el Socket de comunicación con el servidor.



Clase SerialConnectión

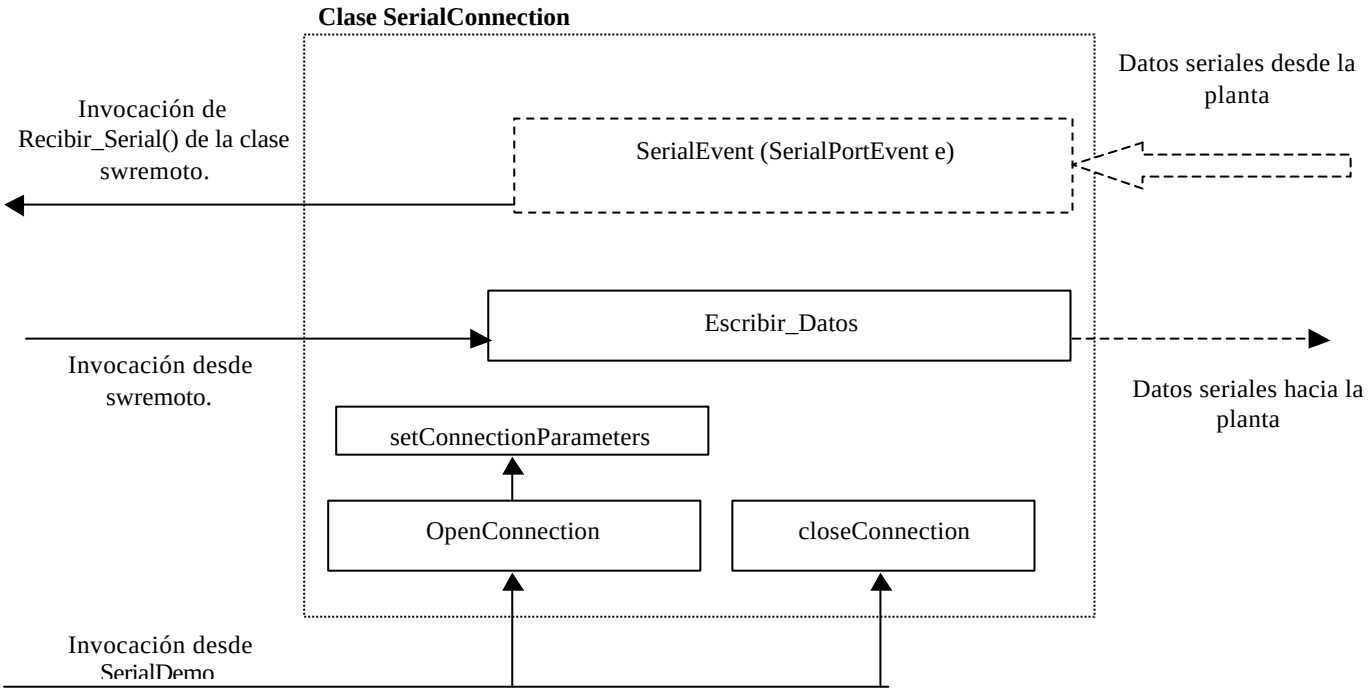
En esta clase se implementa directamente la configuración e inicialización del canal serial de comunicación entre el servidor y el hardware que gobierna la planta (Tarjeta microcontrolada).

Básicamente esta clase es invocada desde una clase interna llamada *SerialDemo.*

La clase *SerialDemo* simplemente muestra una interfaz gráfica para poder configurar el canal serial de comunicación asociado a un puerto serial especifico. A partir de allí dicha clase solo instancia e invoca métodos de configuración de la clase *SerialConnection* por lo cual lo que interesa describir en particular es el funcionamiento de *SerialConnection*.

Como se menciono anteriormente, SerialConnection es la clase que implementa directamente el canal serial de comunicación, por lo cual establece todos los parámetros de configuración para la transmisión de datos por el puerto serial asociado al canal, abre el puerto serial y lo habilita para la transmisión, Inicializa Stream de datos para la transmisión de información por el puerto serial y finalmente asocia un Oyente de eventos de llegada de datos por el puerto serial por lo cual la recepción de información se hace transparente para el desarrollador. Todo esto se realiza mediante el paquete de comunicación serial *javax.comm* que proporciona JAVA para este tipo de comunicación.

Los principales métodos se muestran en el siguiente diagrama:



Método OpenConnectión

Este método es invocado desde la clase SerialDemo mediante la cual se transfieren los parámetros de configuración del puerto serial obtenidos a través de la interfaz gráfica. Básicamente, este método invoca a setConnectionParameters(), establece los InputStream y OutputStream de datos para transmitir información por el canal serial y finalmente adiciona el oyente de eventos de llegada de datos por el puerto serial.

Método setConnectionParameters

Mediante este método se establecen todos los parámetros de configuración para la comunicación por le canal serial como son Tasa de Baudios, Numero de bit de datos, Bit de inicio y Bit de parada, Paridad, Modo de control de flujo etc. Estos parámetros se transfieren directamente a un objeto de tipo *SerialPort* proporcionado en el paquete de comunicación serial de JAVA.

Método CloseConnectión

Este método es invocado desde la clase SerialDemo y es utilizado para cerrar la conexión serial.

Método Escribir_Datos

Este método es el encargado de transferir directamente los datos que envía swremoto a la planta. Esto se hace mediante el OutputStream establecido para escribir datos por el puerto serial.

Método SerialEvent

Este método es invocado cada vez que un evento de llegada de datos por el puerto serial es detectado por el oyente de eventos seriales de la clase. Básicamente, la información que llega por el puerto serial es leída mediante el InputStream establecido y almacenada en un buffer de entrada (StrigBuffer). Una vez obtenida la información se invoca el método Recibir_Serial() de la clase *swremoto*.