UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

"ESTUDIO DEL PROTOCOLO IEEE 488 MEDIANTE EL DESARROLLO DE UNA HERRAMIENTA DE SIMULACIÓN"

TESIS

PARA OBTENER EL TÍTULO DE
INGENIERO EN COMPUTACIÓN

PRESENTA

ANGEL FERNANDO GONZÁLEZ HERNÁNDEZ

DIRECTOR DE TESIS

M. C. HERIBERTO ILDEFONSO HERNÁNDEZ MARTÍNEZ

HUAJUAPAN DE LEÓN, OAX., ABRIL DE 2003

Tesis presentada el 10 de abril de 2003
ante los siguientes sinodales:

M.C. Enrique Guzmán Ramírez
M.C. Mario Alberto Moreno Rocha
M.C. David Martínez Torres

Director de Tesis:
M.C. Heriberto I. Hernández Martínez

Dedicatoria

A mi madre, por vivir el sueño en el que me he convertido.

A mi padre, por estar presente en todo momento y confiar en la persona que soy.

A mis abuelos, por regalarme toda su existencia.

A mi hermano, por enseñarme que a pesar de todo siempre hay otra oportunidad.

A mi hermana, por aparecer en el mejor momento y reemplazar mi ausencia.

Angel Fernando.

Agradezco especialmente a:

A Heriberto I. Hernández Martínez

Por enseñarme que el camino mas fácil para llegar a ser alguien está en uno mismo.

Angel Fernando.

Agradecimientos

A todas las personas que me apoyaron cuando más lo necesitaba.

A Jeanett, por su incondicional apoyo y por emprender junto a mi grandes proyectos, además de brindarme un lugar en su familia, que ahora son parte de mí, gracias.

A la Sra. Maria Antonia, quien es una de las personas más comprensivas que me han acompañado en todo momento.

A Erendira, por ser un gran apoyo y brindarme una amistad verdadera.

A mis sinodales, M.C. Enrique Guzmán Ramírez, M.C. Mario Alberto Moreno Rocha y M.C. Martínez Torres David, por el tiempo dedicado a la revisión de esta tesis y por su valiosa colaboración.

A las siguientes personas, quienes me alentaron a seguir adelante, para lograr este objetivo:

De la Vega Marín Loth Eduardo, Mateos Blanhir Cristian, Aquino Vázquez Leonel, Méndez Aquino Ernesto, Herrera Armenta Samuel, Martínez García Carlos Alberto, así como a todos aquellos que me acompañaron en mi vida universitaria.

Angel Fernando.

Índice

Índice xi
Lista de figuras xv
Lista de tablas xix
Resumen xxi
Abstract xxiii
1. Introducción 1
1.1. Planteamiento y objetivos de la tesis 2
1.2. Estructura de la tesis 3
2. Estado del arte de la tecnología orientada a objetos 5
2.1. Antecedentes del software 5
2.1.1. Crisis del software 5
2.1.2. Complejidad del software 6
2.1.2.1. Descomposición algorítmica 6
2.1.2.2. Descomposición orientada a objetos 6
2.1.2.3. Comparación entre la descomposición algorítmica y la orientada a objetos 6
2.1.3. Paradigma de la orientación a objetos 7
2.1.3.1. Análisis y diseño orientado a objetos 8
2.2. Historia de las metodologías orientadas a objetos 8
2.3. Esfuerzo unificado 9
2.3.1. Estandarización 10
2.3.2. Lenguaje unificado de modelado 11
2.4. Proceso Unificado de Rational 14
2.4.1. Evolución del RUP 14
2.4.2. Objetivos del RUP 15
2.5. Metodología RUP base 17
2.6. Modelado del software SepiGPIB 18
3. Instrumentación programable 19
3.1. Antecedentes de la instrumentación 19
3.1.1. Estándares IEEE 19
3.2. Instrumentación electrónica 20

3.2.1. Evolución de la instrumentación electrónica 21
3.3. Bus de interfaz de propósito general 22
3.3.1. Historia del GPIB 22
4. Bus de interfaz de propósito general 27
4.1. Estándar IEEE 488 27
4.2. Estándar IEEE 488.1 29
4.2.1. Especificaciones mecánicas 29
4.2.2. Especificaciones eléctricas 30
4.2.3. Especificaciones funcionales 30
4.2.3.1. Tipos de instrumentos y señales del bus 30
4.2.3.1.1. Señales de datos 31
4.2.3.1.2. Señales de control de transferencia 32
4.2.3.1.3. Señales de gestión del bus o líneas de manejo de interfaz 32
4.2.3.1.4. Señales de tierra 33
4.2.3.2. Direccionamiento 33
4.2.4. Especificaciones de procedimiento 35
4.2.4.1. Función de transferencia 35
4.2.4.2. Técnicas de direccionamiento e identificación 35
4.2.4.2.1. Sondeo serie 36
4.2.4.3. Transferencia de información 37
4.2.4.4. Funciones IEEE 488 37
4.2.4.5. Órdenes universales 38
4.2.4.6. Órdenes direccionadas 38
4.2.4.7. Terminación de mensajes 39
4.3. Estándar IEEE 488.2 40
4.3.1. Capacidades de la interfaz GPIB 40
4.3.2. Sintaxis y formatos de datos 40
4.3.3. Protocolos de mensajes 41
4.3.4. Conjunto de órdenes 42
4.3.5. Informe de estado 44
4.3.6. Sondeo paralelo 46
5. Modelado del SepiGPIB 49
5.1. Metodología RUP implementada 49
5.2. Planeación del SepiGPIB 49
5.2.1. Herramientas de modelado y desarrollo 51
5.2.1.1. C++Builder 51
5.2.1.2. Rational Rose 2000 52
5.3. Requerimientos del SepiGPIB 52
5.3.1. Fase de requerimientos 52
5.3.2. Modelado de casos de uso 53

5.3.2.1. Seleccionar instrumento 54
5.3.2.2. Agregar instrumento 54
5.3.2.3. Eliminar instrumento 55
5.3.2.4. Seleccionar emulación 55
5.3.2.5. Configurar simulación 56
5.3.2.6. Monitor de bus 56
5.3.2.7. Configurar instrumento 56
5.3.2.8. Ayuda 56
5.4. Análisis y diseño del SepiGPIB 58
5.4.1. Análisis 58
5.4.1.1. Abstracciones clave 58
5.4.1.2. Definición de capas 59
5.4.1.3. Descripción de los atributos utilizando notación UML 59
5.4.1.4. Asociaciones utilizando notación UML 59
5.4.1.5. Agregación utilizando notación UML 59
5.4.1.6. Multiplicidad 60
5.4.2. Diseño 61
5.4.2.1. Modelo conceptual 61
5.4.2.2. Diagrama de clases 63
5.4.2.3. Diagrama de secuencia 65
5.4.2.4. Diagrama de colaboración 65
5.4.2.5. Especificación de la realización de casos de uso 65
5.4.2.5.1. Especificación de la realización del caso de uso Agregar instrumentos 65
5.4.2.5.2. Especificación de la realización del caso de uso Eliminar instrumento 67
5.4.2.5.3. Especificación de la realización del caso de uso Configurar simulación 70
5.4.2.5.4. Especificación de la realización del caso de uso Monitor de bus 72
5.4.2.6. Diseño de clases 74
5.4.2.6.1. Diseño de clase TComandos 74
5.4.2.6.2. Diseño de clase TConfig_sim 75
5.4.2.6.3. Diseño de clase TControlador 75
5.4.2.6.4. Diseño de la clase TInstrumentos 76
5.5. Implementación del sistema SepiGPIB 78
5.5.1. Configuración del bus 78
5.5.2. Tipo de instrumentos 79
5.5.3. Datos y órdenes 79
5.5.4. Direccionamiento 79
5.5.5. Mecanismos de sondeo 80
5.5.6. Estructura del bus 80
5.6. Pruebas al SepiGPIB 81
5.7. Evaluación del SepiGPIB 82

6. Descripción de la herramienta SepiGPIB 83
6.1. Instrumentos modulares simulados 83
6.1.1. Generador de funciones 84
6.1.2. Multímetro 84
6.1.3. Fuente de voltaje 85
6.2. Simulación del SepiGPIB 85
6.2.1. Especificaciones mecánicas 85
6.2.2. Especificaciones funcionales 86
6.2.3. Función de transferencia 87
6.2.4. Simulación de los instrumentos 88
7. Conclusiones y líneas futuras de investigación 91
Bibliografía 93
A. Manual del SepiGPIB A-1
A.1. Instalación del sistema A-1
A.2. Inicialización del programa A-2
A.3. Descripción del sistema principal A-2
A.3.1. Ventana principal SepiGPIB A-2
A.3.2. Ventana de Instrumentos activos A-4
A.3.3. Ventana de Recursos GPIB A-5
A.4. Descripción de los módulos del sistema A-5
A.4.1. Configurar simulación A-5
A.4.2. Agregar Instrumentos A-6
A.4.3. Configurar instrumento A-7
A.4.4. Eliminar instrumento A-7
A.4.5. Ventanas activas A-7
A.4.6. Controlador A-8
A.4.7. Sondeo A-9
A.4.8. Monitor de bus A-10
A.4.9. Nueva simulación A-10
A.4.10. Abrir archivo capturado A-10
A.4.11. Guardar simulación A-11
A.4.12. Ayuda A-11
B. Hojas de especificación de los instrumentos GPIB B-1
B.1. Generador de funciones modelo 33120A de la firma Agilent B-2
B.2. Multímetro modelo 34401A de la firma Agilent B-6
B.3. Fuente de voltaje modelo 3610A de la firma Agilent B-10
Índice Alfabético C-1

Lista de figuras

Figura 1.1. Estructura del presente trabajo de tesis. 4
Figura 2.1. Diagrama a bloques de la construcción de software OO. 8
Figura 2.2. Autores de UML. 10
Figura 2.3. Evolución del UML [URL3]. 11
Figura 2.4. Elementos del proceso de desarrollo [URL3]. 14
Figura 2.5. Evolución del RUP [24]. 15
Figura 3.1. Componentes básicos de un sistema ATE. 20
Figura 3.2. Instrumento tradicional. 21
Figura 3.3. a) Tarjeta de adquisición de datos (DAQ), b) Equipo de adquisición de datos, c) GUI en una PC controlando instrumentos por medio de comunicación GPIB, d) GUI sin instrumentos físicos. 22
Figura 3.4. Relación entre los estándares IEEE 488.1, IEEE 488.2 y SCPI. 25
Figura 4.1. Funcionamiento del estándar IEEE 488. 28
Figura 4.2. Conector IEEE 488, mostrando la identificación de los pines de conexión. 28
Figura 4.3. Tipos de configuración aceptados por el estándar IEEE 488. 29
Figura 4.4. Cable IEEE 488. 29
Figura 4.5. Estructura del bus GPIB. 32
Figura 4.6. Datos GPIB transmitidos en octeto serial y en formato de bit en paralelo. 33
Figura 4.7. Transmisión de datos usando la función de transferencia (*handshake*). 36
Figura 4.8. Octeto de estado del sondeo serie definido por el estándar IEEE 488. 36
Figura 4.9. Estructura funcional del IEEE 488. 40
Figura 4.10. Estructura del registro de estado del estándar IEEE 488.2. 44
Figura 4.11. a) Estado de eventos de registros (SESR), b) Servicio de petición. 45
Figura 4.12. Sondeo paralelo del estándar IEEE 488.2. 46
Figura 5.1. Etapas del desarrollo interactivo del RUP [26]. 51
Figura 5.2. Diagrama de casos de uso para el SepiGPIB. 53
Figura 5.3. Clases identificadas por abstracción. 58
Figura 5.4. Clasificación de las clases del SepiGPIB. 59
Figura 5.5. Asociaciones identificadas en el SepiGPIB. 60
Figura 5.6. Agregaciones identificadas en el SepiGPIB. 61
Figura 5.7. Multiplicidad identificada en el SepiGPIB. 62

Figura 5.8. Diagrama del modelo conceptual para el SepiGPIB. 63
Figura 5. 9. Diagrama de clases del SepiGPIB. 64
Figura 5.10. Diagrama de clases para el caso de uso Agregar instrumento. 66
Figura 5.11. Diagrama de secuencia para el caso de uso Agregar instrumentos. 66
Figura 5.12. Diagrama de secuencia alterno para el caso de uso Agregar instrumento. 67
Figura 5.13. Diagrama de colaboración para el caso de uso Agregar instrumento. 68
Figura 5.14. Diagrama de clases para el caso de uso Eliminar instrumento. 68
Figura 5.15. Diagrama de secuencia para el caso de uso Eliminar instrumento. 69
Figura 5.16. Diagrama de colaboración para el caso de uso Eliminar instrumento. 69
Figura 5.17. Diagrama de clases para el caso de uso Configurar simulación. 70
Figura 5.18. Diagrama de secuencia para el caso de uso Configurar simulación. 70
Figura 5.19. Diagrama de secuencia alterno para el caso de uso Configurar simulación. 71
Figura 5.20. Diagrama de colaboración para el caso de uso Configurar instrumento. 71
Figura 5.21. Diagrama de clases para el caso de uso Monitor de bus. 72
Figura 5.22. Diagrama de secuencia para el caso de uso Monitor de bus. 73
Figura 5.23. Diagrama de colaboración para el caso de uso Monitor de bus. 74
Figura 5.24. Diagrama de diseño de la clase TComandos. 75
Figura 5.25. Diagrama de diseño de la clase TConfig_sim. 76
Figura 5.26. Diagrama de diseño de la clase TControlador. 77
Figura 5.27. Diagrama de diseño de la clase TInstrumentos. 78
Figura 5.28. Diagrama de clases de la agregación de instrumentos al bus. 79
Figura 5.29. Diagrama de clases de la transmisión de datos al monitor de bus. 80
Figura 5.30. Representación de la estructura del GPIB. 81
Figura 6.1. Generador de funciones modelo 33120A. 84
Figura 6.2. Multímetro modelo 34401A. 84
Figura 6.3. Fuente de voltaje modelo 3610A. 85
Figura 6.4. Configuración del bus lineal del SepiGPIB. 86
Figura 6.5. Tipos de instrumentos implementados en el SepiGPIB. 86
Figura 6.6. Señales del bus manejadas dentro del sistema. 87
Figura 6.7. Ventana Configuración de instrumentos. 87
Figura 6.8. Tipos de captura de la simulación del SepiGPIB. 87
Figura 6.9. Ventana Configuración de simulación del SepiGPIB. 87
Figura 6.10. Activación de señales de la función de transferencia. 88
Figura 6.11. Ventana de Instrumentos activos en el bus. 88
Figura 6.12. Ventana del controlador del GPIB para el generador de funciones. 89
Figura A.1. Cuatro de diálogo de bienvenida. A-1
Figura A.2. Cuadro de diálogo que indica la instalación completa del SepiGPIB. A-2
Figura A.3. Ruta de acceso a la aplicación SepiGPIB en el sistema operativo Windows XP. A-2
Figura A.4. Sistema SepiGPIB integrado. A-3
Figura A.5. Menú principal SepiGPIB. A-3

Figura A.6. Activación de los menús del sistema Archivo, Bus, Monitor y Ayuda. A-3
Figura A.7. Ventana de Instrumentos activos en el bus. A-4
Figura A.8. Representación de la ventana de Recursos GPIB de la simulación. A-5
Figura A.9. Ventana Configuración de simulación. A-6
Figura A.10. Barra de herramientas de instrumentos. A-6
Figura A.11. Ventana Configurar instrumento. A-7
Figura A.12. Ventana Eliminar instrumento. A-7
Figura A.13. Administrador de ventanas activas. A-8
Figura A.14. Ventana Controlador del GPIB. A-8
Figura A.15. Ventana de comunicación IEEE 488.2. ..... A-9
Figura A.16. Ventana de Sondeo de instrumentos activada por el controlador. A-9
Figura A.17. Salida del Monitor de bus. A-10
Figura A.18. Abrir archivo de simulación. A-10
Figura A.19. Visualización de un archivo capturado. A-11
Figura A.20. Guardar simulación. A-11
Figura A.21. Ventana del módulo Acerca de. A-12

Lista de tablas

Tabla 2.1. Comparación entre la descomposición algorítmica y la descomposición orientada a objetos. 6
Tabla 2.2. Eventos importantes de los lenguajes de programación OO. 7
Tabla 2.3. Metodologías orientadas a objetos. 9
Tabla 3.1. Evolución de los buses de instrumentación. 23
Tabla 4.1. Señales del estándar IEEE 488 para un conector de 24 pines (americano). 31
Tabla 4.2. Direcciones GPIB válidas para emisores y receptores. 34
Tabla 4.3. Código ASCII asociado a los valores del octeto de datos. 37
Tabla 4.4. Funciones de interfaz del GPIB. 38
Tabla 4.5. Órdenes universales del estándar IEEE 488. 38
Tabla 4.6. Órdenes direccionadas. 39
Tabla 4.7. Órdenes de terminación en un dispositivo GPIB. 39
Tabla 4.8. Capacidades mínimas del estándar IEEE 488.2. 41
Tabla 4.9. Formato de datos especificado por el estándar IEEE 488.2. 41
Tabla 4. 10. Conjunto de estados operacionales de los dispositivos. 42
Tabla 4.11. Órdenes organizadas por grupos. 43
Tabla 4.12. Especificaciones básicas del estándar IEEE 488. 47
Tabla 5.1. Flujos de eventos del caso de uso Seleccionar instrumento. 54
Tabla 5.2. Flujos de eventos del caso de uso Agregar instrumento. 54
Tabla 5.3. Flujos de eventos del caso de uso Eliminar instrumento. 55
Tabla 5.4. Flujos de eventos del caso de uso Seleccionar emulación. 55
Tabla 5.5. Flujos de eventos del caso de uso Configurar simulación. 56
Tabla 5.6. Flujos de eventos del caso de uso Monitor de bus. 57
Tabla 5.7. Flujos de eventos del caso de uso Configurar instrumento. 57
Tabla 5. 8. Flujos de eventos del caso de uso Ayuda. 57
Tabla 6.1. Instrumentos GPIB simulados por el SepiGPIB. 83
Tabla A.1. Listado de funciones de la barra de iconos. A-4

Resumen

El desarrollo de sistemas software se ha visto afectado por la falta de estandarización en sus procesos de metodología de desarrollo y modelado. Los desarrolladores de software han creado sistemas que carecen de una planeación y como consecuencia son difíciles de modificar para actualizarlos a las necesidades emergentes. En la actualidad, la ingeniería de software y la tecnología orientada a objetos ofrecen herramientas de metodologías de desarrollo, de modelado y lenguajes de programación de alto nivel para lograr sistemas robustos, de alta calidad y fuertemente respaldados por documentación mediante modelos que facilitan su comprensión y actualización en modificaciones futuras.

Por otro lado, la instrumentación electrónica ha dejado de ser un campo de estudio basado en el conocimiento y manipulación de instrumentos modulares sencillos, en la actualidad los términos instrumentación electrónica programable e instrumentación virtual han dado origen a sistemas automatizados de medida complejos que interconectan instrumentos programables y/o instrumentos virtuales para análisis, procesamiento y presentación de resultados, todo ello con la finalidad de controlar procesos, verificar productos, explorar servicios, analizar la calidad del producto, etc.

Este trabajo de tesis presenta el desarrollo de una herramienta de software para el estudio y análisis del protocolo IEEE 488 mediante la utilización de herramientas que proporciona la tecnología orientada a objetos y de los conceptos de la instrumentación electrónica programable. El desarrollo de la herramienta de simulación SepiGPIB se describe en base a la metodología RUP, el modelado en UML y la realización en el lenguaje de programación de alto nivel C++ Builder. Asimismo, se describe el funcionamiento del software final y se presentan los estudios realizados en los campos de la tecnología orientada a objetos y de la instrumentación electrónica programable.

La herramienta de simulación SepiGPIB intenta ser una introducción a las nuevas herramientas software de metodología y modelado para el desarrollo de sistemas de simulación de protocolos de comunicaciones industriales, de instrumentación y de redes de computadoras.

Abstract

The development of software has been affected by the lack of standardization of development and modeling methodology processes. Software developers have created systems that lack planning and consequently are difficult to modify in order to update them to the emerging needs. Nowadays, software engineering and object-orientated technology offer tools for methodology development for modeling and for high-level programming languages to achieve robust systems of high quality, and that are strongly endorsed by documentation by means of models wich facilitate understanding and updating in future modifications.

On the other hand, the electronic instrumentation has stopped being a field of study based on the knowledge and manipulation of simple modular instruments, the terms programmable instrumentation and virtual instrumentation have given rise to automated systems of average complexes that interconnect programmable instruments and/or virtual instruments for analysis, processing and presentation of results. All this with the purpose of controlling processes, of verifying products, of exploring services, of analyzing the quality of the product, etc.

This thesis presents the development of a software tool for the study and analysis of the IEEE 488 protocol by means of the utilization of tools that provide the object-orientated technology and the concepts of the programmable instrumentation. The development of the simulation tool (SepiGPIB) is described on the basis of a methodology (RUP), modeled on UML and carried-out in a high level programming language (C ++ Builder). The final software is described and the studies carried out on object-orientated technology are presented alongside those on programmable instrumentation.

The simulation tool SepiGPIB attempts to be an introduction to the new methodology and modeling software tools for the system development of industrial communication, instrumentation and computer networks protocol simulation.

1. Introducción

En la actualidad, la instrumentación electrónica afronta constantes cambios y se ha convertido en una herramienta indispensable para ingenieros, científicos y técnicos que requieren de sistemas electrónicos de medida de gran exactitud y precisión [29]. Por un lado, el continuo avance de la microelectrónica, las prestaciones de los paquetes informáticos y el desarrollo de nuevas tecnologías en el diseño de sistemas de medida para el control de procesos, verificación de productos, explotación de servicios, análisis de calidad, etc., han permitido el desarrollo de potentes sistemas automatizados de medida (ATE, *Automated Test Equipment*) [30, 31, 33], mientras que por el otro lado, es cada vez más común la utilización de las computadoras personales (PC, *Personal Computer*) como el principal recurso en diversas áreas de aplicación como son laboratorios, entornos industriales, sistemas de instrumentación, etc.

Los sistemas ATE guardan una gran dependencia respecto a los sistemas de adquisición de datos (DAQ*, Data Acquisition*) y para lograr la interconexión de los diversos sistemas electrónicos de medida, existe un amplio número de protocolos de comunicaciones dedicados a dicha tarea, como son IEEE 488 o GPIB (*General Purpose Instrumentation Bus*), VXI (VME *Bus eXtension for Instrumentation*), PXI/CompactPCI (PCI *eXtension For Instrumentation*), MXI (*Multisystem Instrument Interface*), etc.

El estudio del GPIB ha alcanzado una enorme expansión permitiendo el diseño de complejos sistemas ATE implementados en diversas plataformas de computadoras bajo diferentes sistemas operativos, lo cual ha dado lugar al concepto de instrumentación virtual que, de forma paulatina, ha venido a reemplazar al concepto de instrumentación clásico [32].

Las aplicaciones en un entorno de instrumentación programable se realizan comúnmente mediante herramientas software propietarias, por ejemplo la herramienta VEE (*Visual Engineering Enviroment*) de la firma Agilent Technologies [18] o las herramientas LabView y LabWindows de National Instruments [URL1], con un gran soporte matemático y con funciones destinadas al control y programación de los instrumentos. La libertad con la que cuenta el programador de sistemas se ha obtenido de la propia estandarización de los protocolos, resultado de ello son las librerías, controladores (*drivers*), etc., que los fabricantes proporcionan al usuario para utilizarse junto a lenguajes de programación de alto nivel, como C++, Visual C++, etc., para el desarrollo de aplicaciones de usuario.

El desarrollo de sistemas de software es una industria relativamente joven que aún no ha alcanzado un nivel de madurez, consecuentemente, los productos desarrollados a menudo carecen de la estabilidad requerida para ser explotados como productos comerciales. Por lo tanto, uno de los aspectos más importantes de la ingeniería del software orientada a objetos (OOSE, *Object Oriented Software Engineering*) es proveer de alternativas para mejorar el proceso de desarrollo de software.

Dentro de las alternativas propuestas existen dos aspectos vitales, por un lado, plantear una metodología de desarrollo del software que permita realizar un modelo del sistema a construir y por otro lado, brindar una documentación adecuada para presentar los aspectos más relevantes que se involucran en el desarrollo del software.

La construcción de un sistema debe iniciar por conocer las demandas de los usuarios finales y los requerimientos que el sistema debe cubrir, para ello es necesario visualizar un bosquejo preliminar de cómo será el sistema mediante dibujos, textos, diagramas, etc.

La importancia de los modelos ha sido evidente en todas las disciplinas de ingeniería a lo largo de la historia [8, 14]. Para poder construir algo, se realizan dibujos que describen cuál será su apariencia final. Los dibujos, simples o complejos, dan pauta a la especificación de lo que será el producto final y en ellos se planean los costos, tiempos y estimaciones de distribución y de los recursos para el desarrollo total del producto o sistema bajo desarrollo.

El modelar un proyecto requiere de varios niveles de modelado con la finalidad de poder entender el sistema y poder comunicar las ideas a otras personas. En el desarrollo de software un modelado asegura un software final de calidad.

Por otra parte, el contar con una buena documentación permite tener la habilidad de reutilizar la tecnología, construyendo bloques de un sistema que son plenamente identificados y explotados en base a su documentación para ser aplicados a nuevos proyectos. El documentar el desarrollo del software tiene la finalidad de describir la forma en la que se desarrolla un sistema presentando las alternativas que satisfacen las necesidades de los usuarios finales, además de su utilidad para documentar los diferentes aspectos de la estructura, visualización y control de un sistema se obtiene modelándolo. Por lo anterior, el modelado de un sistema debe ser parte esencial de todas las actividades de desarrollo de software.

Cabe destacar que desde la aparición de la tecnología orientada a objetos han surgido serios problemas para modelar sistemas en base a esta nueva tendencia, debidos principalmente, a que cada modelo presenta sus propias herramientas, las cuales contienen símbolos y terminologías propias que resultan en frustraciones para quienes intentan modelar en base a la tecnología orientada a objetos [2, URL4]. Sin embargo, los procesos de estandarización han dado origen a herramientas de modelado como es el Lenguaje Unificado de Modelado (UML, *Unified Modeling Language*) y de metodología como es el Proceso Unificado de Rational (RUP, *Rational Unified Process*).

1.1. Planteamiento y objetivos de la tesis

El objetivo principal de esta tesis es el desarrollo de una herramienta software para el estudio y análisis del protocolo de comunicaciones GPIB con fines académicos, denominada Sistema educativo para el estudio del protocolo de instrumentación GPIB (SepiGPIB). El trabajo realizado en esta tesis forma parte de una línea de investigación, abierta por el Cuerpo Académico de Redes de Instrumentación del Instituto de Electrónica y Computación de la Universidad Tecnológica de la Mixteca, consistente en el estudio de técnicas de descripción formal (FDT, *Formal Description Techniques*) en el desarrollo de sistemas de instrumentación programable y de los protocolos de comunicaciones industriales y de instrumentación. Dichas técnicas formales permiten el diseño de aplicaciones de usuario que cumplan con los estándares para el desarrollo de software de calidad y aporten herramientas académicas para el conocimiento de los protocolos de comunicación en estudio.

Una vez planteado el objetivo principal y el ámbito donde se centra el trabajo de investigación, se proponen los siguientes objetivos secundarios:

- Analizar los protocolos de comunicaciones empleados en instrumentación programable existentes en el mercado y elaborar un estado del arte de los mismos.
- Realizar un estudio detallado del protocolo de comunicaciones IEEE 488, haciendo especial énfasis en su funcionamiento y en el estándar IEEE 488.1.
- Estudiar las distintas herramientas para desarrollo de software con tecnología orientada a objetos y elaborar un estado del arte de los mismos.
- Modelar el protocolo de comunicaciones IEEE 488 mediante UML y desarrollar una herramienta software con fines académicos, atendiendo a la metodología RUP e implementar el modelado con el lenguaje de alto nivel C++ Builder.
- Validar y verificar el funcionamiento de la herramienta resultante para su utilización en la enseñanza de los protocolos de comunicaciones, particularmente en instrumentación virtual.
- Documentar el desarrollo del software mediante la utilización de las herramientas de modelado y realizar un documento que acompañe a la aplicación, para que el usuario obtenga los conocimientos teóricos del funcionamiento del protocolo IEEE 488.

La hipótesis que se plantea es que con la ayuda de una herramienta software de calidad, los usuarios interesados en el conocimiento del protocolo IEEE 488 puedan conocer mejor su funcionamiento en un entorno amigable y con las capacidades que la ingeniería del software y la tecnología orientada a objetos proporcionan. Para ello, se considera de vital importancia el estudio teórico y práctico de la tecnología orientada a objetos y de la instrumentación programable con la finalidad de conocer detalladamente el protocolo en estudio y las posibles soluciones a los objetivos planteados.

1.2. Estructura de la tesis

La presentación del trabajo realizado se estructura como muestra la figura 1.1 y de la forma que se detalla a continuación.

En el capítulo 2 se hace un estudio de la tecnología orientada a objetos, se exponen los conceptos básicos y se abordan las distintas soluciones existentes en el mercado para el desarrollo de software orientado a objetos.

En el capítulo 3 se hace un estudio de la evolución de la instrumentación electrónica y del desarrollo de interfaces para control de mediciones que han dado origen a los conceptos de instrumentación programable e instrumentación virtual.

En el capítulo 4 se hace una descripción detallada del protocolo IEEE 488 comentando los aspectos más importantes de su funcionalidad.

El capítulo 5 presenta el modelado del sistema SepiGPIB realizado mediante una metodología y un lenguaje de modelado, los cuales comprenden su análisis, diseño e implementación.

En el capítulo 6 se presenta el software SepiGPIB de acuerdo a las especificaciones del protocolo IEEE 488.

La exposición del trabajo de esta tesis finaliza con una serie de conclusiones y posibles líneas de investigación, las cuales se exponen en capítulo 7.

Por último se presentan las referencias bibliográficas utilizadas en la realización de esta tesis y finalmente, se presentan los anexos donde se presenta el manual de usuario del SepiGPIB y las hojas de características de los dispositivos electrónicos de medida simulados.

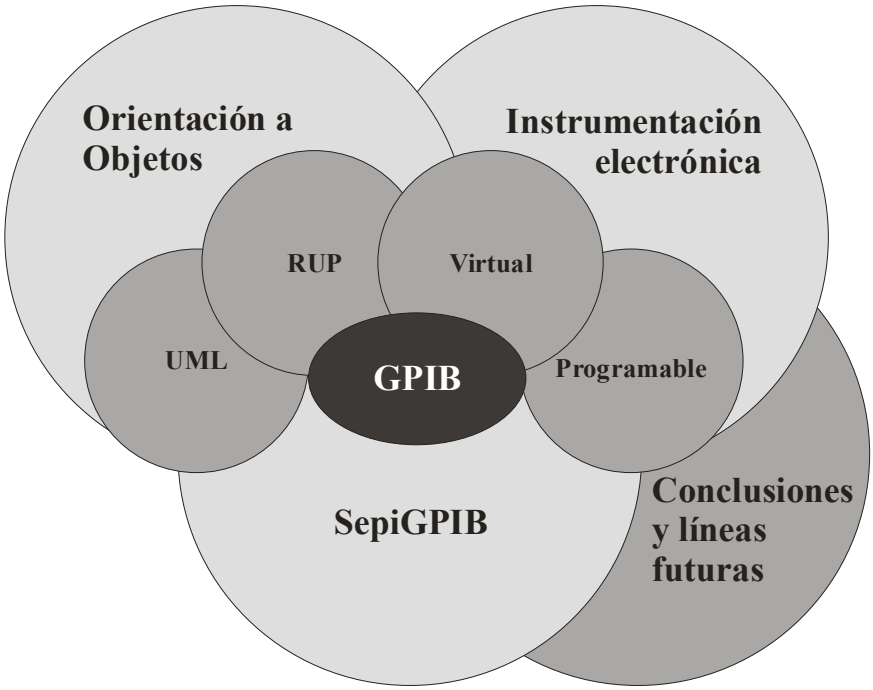


Figura 1.1. Estructura del presente trabajo de tesis.

2. Estado del arte de la tecnología orientada a objetos

2.1. Antecedentes del software

Desde la aparición de las computadoras en la década de los años cincuenta, éstas han ido evolucionando gracias a los avances en la microelectrónica, sin embargo, el desarrollo del software no ha seguido el mismo nivel de crecimiento, debido a que la mayoría de aplicaciones se desarrollaban sin normas, métodos o procesos a seguir. En los últimos años el avance en las metodologías, lenguajes de programación y procesos de desarrollo de software ha favorecido la creación de software profesional.

2.1.1. Crisis del software

La crisis del software tiene que ver con la velocidad a la que se construye [37], las dificultades de mantenimiento, problemas de costos y confiabilidad del software. El propósito del desarrollador es intentar eliminar esta crisis al solucionar problemas como son:

- La calendarización y estimación de costos.
- Problemas de calidad de software.
- Satisfacer las expectativas del cliente.

Hoy en día, la industria ofrece "fábricas de software que envejece" [37]. Existen aplicaciones basadas en software que necesitan ser renovadas urgentemente debido a que presentan algunas de las siguientes problemáticas:

- Son sistemas de información realizados hace más de 25 años y que resultan difíciles de modificar debido a que no cuentan con documentación y/o se desconoce el lenguaje de desarrollo.
- Son aplicaciones avanzadas de ingeniería construidas sin técnicas de desarrollo de software.
- Son sistemas firmware[1] que presentan un comportamiento que tiende a provocar fallos.

Para resolver tales problemáticas es necesario aplicar una reingeniería a los sistemas y hacerlos más competitivos, desafortunadamente la tendencia de los desarrolladores de software es actualizar los sistemas, con la justificación de que no cuentan con suficiente dinero ni tiempo para sustituir los programas.

[1] Software en forma de hardware [17].

2.1.2. Complejidad del software

El software tiene una complejidad inherente ya que por sí solo es de naturaleza abstracta, mantiene un comportamiento discreto y presenta dificultades al administrar su desarrollo y lograr un total dominio del problema [25].

La solución de problemas mediante herramientas de software presenta una complejidad en ocasiones contradictoria, debido a que tanto el usuario final como el desarrollador tienen diferentes visiones de cómo resolver el problema. El hecho de contar con más de un desarrollador dificulta la comunicación y coordinación del desarrollo del software, pero éste se optimiza al descomponer el problema en módulos independientes.

A continuación se describen dos métodos de descomposición [25] utilizados para reducir la complejidad del software, ambos utilizan la técnica conocida desde tiempos antiguos como "divide y vencerás", la cual consiste en afrontar un problema mediante la descomposición del mismo en partes más pequeñas. La comparación de las características de estos métodos se presenta en el apartado 2.1.2.3.

2.1.2.1. Descomposición algorítmica

La descomposición algorítmica se aplica para descomponer un problema en otros más pequeños. El programa se descompone en partes pequeñas, llamadas subprogramas, obteniendo una estructura, en forma de árbol, donde cada unidad fundamental realiza su trabajo al llamar ocasionalmente a otro subprograma o programa.

2.1.2.2. Descomposición orientada a objetos

En el método de descomposición orientada a objetos cada objeto modela algún evento del mundo real y es visto como un conjunto de entidades autómatas que al interrelacionarse muestran cierta conducta. En este tipo de descomposición, los objetos, sus datos y funciones se combinan en forma modular.

2.1.2.3. Comparación entre la descomposición algorítmica y la orientada a objetos

La mayoría de los programadores trabajan en un lenguaje de desarrollo y utilizan sólo un estilo de programación, frecuentemente desconocen métodos alternativos de resolución y por consiguiente se enfrentan a dificultades al elegir el estilo apropiado para resolver el problema. La Tabla 2.1 presenta una comparación entre la descomposición algorítmica y la descomposición orientada a objetos.

Tabla 2.1. Comparación entre la descomposición algorítmica y la descomposición orientada a objetos.

Operaciones	Descomposición algorítmica	Descomposición orientada a objetos
Descomposición	Se basa en algoritmos orientados a procedimientos.	Se basa en clases y objetos.
Modelado	Se basa en la descomposición del problema.	Se basa en eventos, clases y objetos.
Comandos u órdenes	En base a eventos.	Muestran el mundo tal como se percibe.
Funcionalidad	Cada subprograma realiza una función.	Los objetos y las funciones realizan un comportamiento.
Característica	Presenta dificultades en el tamaño del software.	Se favorece la reutilización de código.

Algunos autores sostienen que lo más recomendable es aplicar inicialmente una descomposición orientada a objetos y, posteriormente, continuar con la descomposición algorítmica [25].

2.1.3. Paradigma de la orientación a objetos

La tecnología orientada a objetos (OOT, *Object-Oriented Technology*) ofrece métodos y técnicas para desarrollar software de calidad. El desarrollo de programas orientados a objetos es un enfoque diferente del mundo informático, ya que implica la creación de modelos del mundo real y en base a ellos se construyen programas informáticos. El proceso de programación inicia con el análisis del problema, a continuación se construye un modelo, el cual será implementado en un lenguaje de programación y como resultado final se obtiene un programa que resuelve el problema inicial. El principio básico de la programación orientada a objetos es ver un sistema de software como una secuencia de transformaciones en un conjunto de objetos [37].

Las tecnologías orientadas a objetos se han convertido en uno de los motores claves de la industria del software [16]. El nacimiento de la orientación a objetos no es nuevo, sino que es una vieja y madura tecnología que se remonta a los años sesentas. La Tabla 2.2 muestra algunos eventos importantes de la evolución de los lenguajes de programación OO.

Tabla 2.2. Eventos importantes de los lenguajes de programación OO.

Año	Eventos
1967	Surge el lenguaje Simula, el cual introdujo por primera vez los conceptos de clases, corrutinas y subclases.
1976	Primera versión comercial del lenguaje Smalltalk, considerado el primer lenguaje OO ya que utiliza clases y objetos.
1980	Borland desarrolla C++ como una extensión del lenguaje C, en esta misma década se desarrollaron Objective C (extensión de C), object Pascal (extensión de Pascal), Ada y otros.
1984	Mejoras en herramientas y lanzamientos comerciales de C++ por distintos fabricantes, que justificaron la atención hacia la programación OO en la comunidad de desarrollo de software.
1985	La primera versión comercial C++ se documento en el libro de Bjarne Stroustrup “The C++ Programming Language” editado por Addison-Wesley.
1990	Se consolida la OO dada la necesidad de agilizar la creación de prototipos, dando lugar a nuevos conceptos como el de Aplicación de Desarrollo Rápido (RAD, *Rapid Aplication Developments*).
1995	Surge Eiffell, creado por Beltrand Meyer como un lenguaje OO enfocado a ambientes universitarios y de investigación.
1997	Se desarrollan herramientas tipo CASE (*Computer Aided Software Engineering*) para crear nuevos prototipos de lenguajes.
1998	Desarrollo de arquitecturas de objetos distribuidos como son el Métodos de Invocación Remota (RMI, *Remote Method Invocation*), los Agentes de Arquitectura de Petición Común (CORBA, *Common Object Request Broker Architecture*) y el Modelo de componentes (COM, *Component Object Model*).

Actualmente, la orientación a objetos parece ser la mejor opción para desarrollo de software y una alternativa para eliminar su crisis. Entre los creadores de metodologías orientadas a objetos se encuentran Grady Booch [7, 8], James Rumbaugh [39, 40], Ivar Jacobson [23, 24], entre otros. A continuación se describen las fases del desarrollo de software.

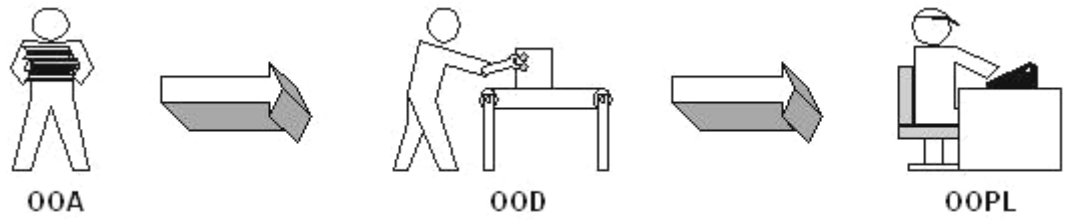


Figura 2.1. Diagrama a bloques de la construcción de software OO.

2.1.3.1. Análisis y diseño orientado a objetos

Los autores de metodologías OO proponen abordar aspectos de implementación en la construcción del software al intentar abarcar la totalidad de su ciclo [25]. Esta utilización de la tecnología de objetos, para plantear problemas más que soluciones, se conoce con el nombre de Análisis Orientado a Objetos (OOA, *Object-Oriented Analysis*).

El OOA se basa en conceptos como objeto y atributo, el todo y las partes, clases y miembros. En un problema, las especificaciones[2] son parte del OOA que responden a la pregunta *qué hace*. Durante la fase del OOA se modela el sistema mediante los objetos que lo forman y las relaciones estáticas o de uso que existen entre ellos. Este enfoque pretende conseguir modelos que se ajusten mejor al problema real a partir del conocimiento denominado *dominio del problema*. Por ejemplo algunas de las herramientas de ayuda para el OOA son el análisis de requerimientos, el análisis de flujos de trabajo y el análisis de negocios, entre otras.

La siguiente fase del proceso de desarrollo de software es el Diseño Orientado a Objetos (OOD, *Object-Oriented Desing*), que responde a la pregunta *cómo se hace*. Durante esta fase se crea un modelo basado en el análisis de la tarea específica, se buscan atributos útiles y comportamientos aún no definidos con claridad por cada objeto. Algunas herramientas de ayuda para el OOD son UML, RUP y herramientas tipo CASE (*Computer Aided Software Engineering*).

Los límites entre el OOA y el OOD no son claros, es difícil definir la transición entre ambas fases, debido a que no representan un proceso estricto de dos etapas y a veces se unen en una sola.

La fase de diseño conduce a la fase de implementación, que consiste en traducir el modelo obtenido en código de un Lenguaje de Programación Orientado a Objetos (OOPL, *Object-Oriented Programming Language*). La fase de codificación del proceso de desarrollo OO (*Object-Oriented*) se llama Programación Orientada a Objetos (OOP, *Object-Oriented Programming*). Algunos de los OOPL son Delphi, C++ Builder, Visual C, etc.

En síntesis, el proceso de desarrollo orientado a objetos supone la construcción de un modelo del mundo real que se pueda traducir posteriormente en un código escrito en un OOPL. El análisis, diseño y programación no constituyen un proceso único y el no poder contar con estándares para construir un software orientado a objetos ha originado la creación de diversos métodos que proporcionan reglas para la identificación de clases y sus relaciones. La figura 2.1 propone un diagrama a bloques de las fases de construcción de software OO.

2.2. Historia de las metodologías orientadas a objetos

Hasta ahora, se ha mencionado que el paradigma orientado a objetos ayuda a resolver problemas en la construcción de software con la utilización del OOA y del OOD. A continuación se mencionan algunos de los métodos más utilizados por las metodologías OO.

---

[2] Las especificaciones se refieren a los detalles que constituyen el problema real.

Los métodos de desarrollo para los lenguajes de programación, originalmente desarrollados por Constantine, DeMarco, Stephen J. Mellor, Ward, Yourd y otros, alcanzaron a cubrir las necesidades existentes en el mercado del software [25]. Muchos sistemas de ingeniería de software tipo CASE fueron utilizados para la construcción de grandes sistemas.

La OOP se formuló en los inicios de la década de los ochentas, pero es hasta finales de la misma que surgen infinidad de metodologías de Análisis y Diseño Orientado a Objetos (OOAD, *Object-Oriented Analysis and Design*), es decir, el análisis que existía anteriormente, análisis estructurado, no se adecuaba del todo a la OOP. La Tabla 2.3 presenta las diferentes metodologías orientadas a objetos.

Tabla 2.3. Metodologías orientadas a objetos.

Año	Siglas	Metodología	Autores
1990	DOOS	Diseño de Software Orientado a Objetos (*Design Object-Oriented Software*)	Wirfs-Brock (et. al.)
1991	OOA/OOD	Análisis Orientado a Objetos (*Object-Oriented Analysis*) Diseño Orientado a Objetos (*Object-Oriented Design*)	Coad, P. y Yourdon, E.
1991	OMT	Técnica de Modelado de Objetos (*Object Modeling Technique*).	Rumbaugh, J. (et. al.)
1992	OL	Ciclos de vida de los Objetos (*Object Lifecycles*)	Sally Shlaer y Steve Mello
1992	OOSE	Ingeniería de Software Orientada a Objetos (*Object-Oriented Software Engineering*)	Jacobson, I. (et. al.)
1993	OOAD	Análisis y Diseño Orientado a Objetos (*Object-Oriented Analysis and Design*)	Martin James
1993	OOSD	Desarrollo de Sistemas Orientados a Objetos (*Object-Oriented System Development*)	Champeaux (et. al.)
1994	OOADA	Análisis y Diseño con Aplicaciones Orientadas a Objetos (*Object-Oriented Analysis and Design with Applications*)	Grady Booch
1994	MOSES	Método MOSES	Henderson-Sellers y Edwards
1994	FUSION	Método Fusion	Coleman (et. al.)
1994	ROOM	Modelado de Tiempo Real Orientado a Objetos (*Real-Time Object-Oriented Modeling*)	Selic (Object Time Ltd.)
1996	UML	Lenguaje Unificado para el Modelado (*Unified Modeling Language*)	Booch, G., Rumbaugh, J. y Jacobson, I.
1997	RUP	Proceso Unificado de Rational (*Process Unified Rational*)	Booch, G., Rumbaugh, J. y Jacobson, I. (Rational Software Corp.)

2.3. Esfuerzo unificado

El avance en las metodologías orientadas a objetos dio origen a grandes esfuerzos por unificar los conceptos que planteaban los métodos existentes mediante algunas fusiones entre sus autores. El primer éxito en combinar y reemplazar los métodos existentes, llegó en 1994 cuando James Rumbaugh se unió a Grady Booch en Rational Software Corporation, resultado de ello fue la publicación en 1995 de una propuesta que combinaba los conceptos de las metodologías OMT (*Object Modeling Technique*) [39] y Booch [7]. Más tarde, la integración de Ivar Jacobson [23] al consorcio Rational unificó el trabajo de estos autores en un conjunto llamado Lenguaje Unificado de Modelado (UML, *Unified Modeling Language*), lenguaje que favorecería el campo de las metodologías OO.

Grady Booch

James Rumbaugh

Ivar Jacobson

Figura 2.2. Autores de UML.

En 1996, el grupo de administración de objetos (OMG, *Object Management Group*) publicó una petición de propuesta para un enfoque estándar sobre el modelado orientado a objetos. Los autores Grady Booch, James Rumbaugh e Ivar Jacobson, conocidos como "los tres amigos", empezaron a trabajar en lo que sería un lenguaje de modelado ampliamente aceptado por los fabricantes de herramientas, metodólogos y desarrolladores, quienes serían los usuarios eventuales [8, 24, 40]. La figura 2.2 muestra a los autores del esfuerzo unificado.

2.3.1. Estandarización

El resultado del esfuerzo unificado fue el UML, adoptado por los miembros del OMG como estándar en noviembre de 1997, el OMG asume la responsabilidad de futuros desarrollos con el estándar UML.

La estandarización favorece tanto la expansión del uso de modelado orientado a objetos entre los desarrolladores, como la aparición de herramientas de formación y utilización. Cabe aclarar que UML es un lenguaje de modelado y no un método. Un método consiste en un lenguaje y un proceso que sugiere los pasos a seguir para modelar un sistema y un lenguaje de modelado es la notación que usan los métodos para expresar los diseños.

Grady Booch, James Rumbaugh e Ivar Jacobson presentaron la primera recomendación del estándar bajo la versión Método Unificado 0.8. En junio de 1996 se liberó la primera documentación del estándar UML en sus versiones 0.9 y 0.91 y durante ese mismo año muchas organizaciones vieron al UML como una estrategia para sus negocios.

En enero de 1997, compañías como IBM, Objectime, Platinum Tecnnology, Ptecn, Reinch Technologies trabajaron para aportar ideas, lo cual originó la publicación de las versiones UML 1.0 y 1.1. El OMG adoptó el UML como estándar el 17 de noviembre de 1997, manejando su nueva versión UML 1.3. La figura 2.3 muestra la evolución de UML y se puede ver la unificación de los métodos OMT, Booch, y OOSE.

UML debe, en gran parte, su gran aceptación a que no incluye un proceso como parte de su propuesta, es fácil de comprender y modificar, combina lo mejor de los modelados de datos, negocios, objetos y componentes, etc.

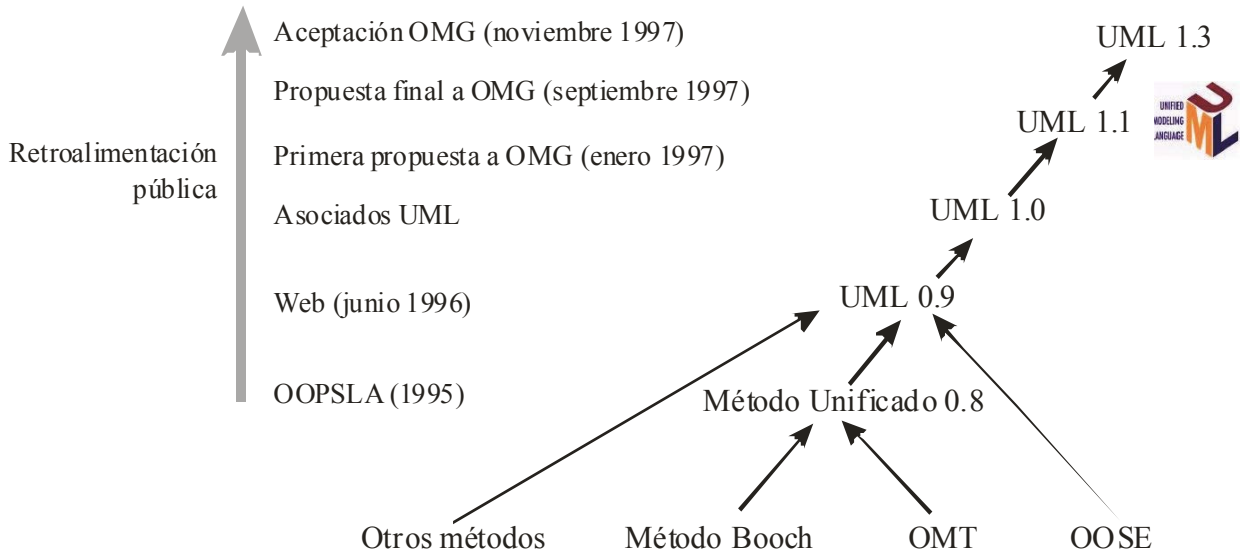


Figura 2.3. Evolución del UML [URL3].

2.3.2. Lenguaje unificado de modelado

Al modelar un proyecto se tienen varios niveles de modelado con la finalidad de entender mejor el sistema o lograr comunicar las ideas a otras personas, el modelar es un buen inicio para proveer software de calidad y para realizar un esquema principal de cómo visualizar el producto final.

Durante el proceso de modelado es importante saber que [14]:

- Modelar es una simplificación de la realidad, es decir, el modelo de un sistema puede desarrollarse desde diferentes puntos de vista y cada uno de los modelos resultantes es una abstracción del mismo sistema.
- Se construyen modelos para entender mejor cómo se desarrolla el sistema. Un modelo ayuda a visualizar el sistema que se quiere obtener, permitiendo especificar su estructura y proporcionando una guía en la construcción del mismo.
- El construir modelos complejos se debe a que no se comprende la forma en la cual está conformado el sistema y a las limitaciones de las capacidades del entendimiento de los seres humanos, lo cual implica la dificultad de lograr una comprensión de los requerimientos del software a modelar.

El modelar es un proceso informal debido a que cada persona tiene una forma única de percibir el dominio del sistema que se modela, sin embargo, es cada vez más común el uso de herramientas formales para ayudar a estandarizar el proceso de modelado. Los principios del modelado son los siguientes [8]:

- La elección de los modelos influye directamente en la forma de afrontar y solucionar los problemas que se presentan durante el desarrollo del software.
- Los modelos pueden ser expresados con diferentes niveles de precisión dependiendo del número de niveles empleado y del grado de detalle que se exija.
- Los mejores modelos son aquellos que se apegan a la realidad, es decir, deben representar las condiciones del mundo real.
- No es suficiente un solo modelo, debido a que existen algunas condiciones del sistema que deben ser modelados independientemente.

Durante la creación de modelos, o modelado, se investigan los requerimientos del producto, los cuales deben incluir las áreas funcionales, apariencias, rentabilidad, etc., con la finalidad de que el modelo resultante pueda describir todos los aspectos del producto [13].

La presentación del modelado se realiza mediante vistas. Una vista es un subconjunto de UML que modela construcciones que representan un aspecto del sistema, es decir, cada vista describe aspectos específicos del sistema en construcción, la clasificación en las diversas vistas es arbitraria. Para mayor información se recomienda referirse a [8, 15, 24, 40, 41].

Las características que debe cumplir un modelo son [14]:

- *Exactos*: el modelo debe describir correctamente el sistema en construcción.
- *Consistentes*: las diferentes vistas del modelo no deben expresar "cosas" diferentes.
- *Fácil de comunicar*: el modelo debe facilitar la comprensión de la solución propuesta.
- *Fácil de modificar*: el modelo debe tener la capacidad de aceptar modificaciones o adecuaciones a lo largo del desarrollo del software.

Para lograr lo anterior, la industria del software ha invertido grandes esfuerzos en la creación de herramientas de modelado, dando origen a técnicas de programación para la construcción de programas visuales que mediante la interconexión y manipulación de símbolos faciliten la tarea de modelado de sistemas, ejemplo de ello es la herramienta Rational Rose [URL2].

Antes de explicar el lenguaje de modelado se debe diferenciar lo que es un lenguaje de modelado de un método. Un método es explícitamente un camino estructurado que se toma al realizar una acción, un método nos dice qué hacer, cómo hacerlo, cuándo hacerlo y porqué se hace, los métodos contienen modelos y estos modelos se utilizan para describir aspectos importantes y para comunicar los resultados al emplear un método.

Un lenguaje de modelado consiste en una notación, la cual contiene un grupo de reglas que dictaminan su utilización y proporciona reglas sintácticas, semánticas y pragmáticas. Las reglas sintácticas se encargan de especificar el uso de los símbolos y sus combinaciones, las reglas semánticas especifican la interpretación de los símbolos y las reglas pragmáticas están destinadas a revelar la intención de los símbolos dentro del modelado.

El UML se utiliza para especificar, visualizar, construir y documentar artefactos de un sistema de software, para ello incluye conceptos semánticos, notación y principios generales. El UML pretende unificar las técnicas de modelado y es idóneo para utilizarlo en generadores de código[3] y generadores de informes.[4]

El modelado de un sistema en UML inicia con la captura de información sobre la estructura estática, mediante la definición de objetos y sus relaciones, para posteriormente continuar con el comportamiento dinámico, en donde se definen tanto la historia temporal de los objetos como la comunicación entre objetos.

El UML utiliza un modelado visual con notaciones gráficas para analizar y diseñar las aplicaciones, las cuales distinguen entre los dominios del negocio y los dominios de la computadora. Es un lenguaje que puede usarse en cualquier proceso a lo largo del desarrollo del ciclo de vida y es independiente de la tecnología de implementación [15, 41]. UML tiene como objetivo el describir cualquier tipo de sistema en términos de diagramas orientados a objetos, para ello UML especifica elementos, diagramas y símbolos basados en el paradigma OO y se emplea en las diferentes fases del desarrollo de un sistema, desde la fase de requerimientos hasta la fase final de pruebas [6, 14].

[3] Software que produce código a partir de las especificaciones de un problema.

[4] Software de producción de informes de salida a partir de una especificación de un fichero de entrada.

El UML está constituido de las siguientes partes [8]:

- *Vistas*: cada vista muestra diferentes aspectos del sistema que se está modelando. Una vista no es una gráfica, sino representa una abstracción en forma de diagramas que constituye un aspecto particular del sistema.
- *Diagramas*: los diagramas son gráficas que describen el contenido de una vista. El UML cuenta con nueve tipos de diagramas cuya combinación proporciona una vista general del sistema.
- *Elementos del modelo*: los conceptos usados en los diagramas son elementos del modelo que representan conceptos OO comunes como son las clases, los objetos y los mensajes, así como sus relaciones, las cuales incluyen asociaciones, dependencia y generalización.
- *Mecanismos generales*: un mecanismo provee de información adicional sobre la semántica del elemento modelado.

UML es un lenguaje para la visualización, especificación, construcción y documentación de un sistema de software. La visualización es de suma importancia para proyectar el desarrollo del sistema y para ello es necesario realizar un bosquejo general del sistema con ayuda de las ideas obtenidas de las abstracciones del dominio del problema. UML es un lenguaje de especificación debido a que ayuda al modelado de sistemas precisos, no ambiguos y completos, mediante la descripción detallada de las partes esenciales del proyecto. El UML no es un lenguaje de programación visual, pero sus modelos resultantes pueden implementarse directamente en la mayoría de lenguajes de programación como son Java, C++, Visual C, etc. Se dice que UML es un lenguaje de documentación ya que permite organizar el software producido en forma de artefactos, dentro de los artefactos generados por UML se tienen los requerimientos, la arquitectura, el diseño, el código fuente, los planes del proyecto, las pruebas y los prototipos, dependiendo del desarrollo se agregan algunos otros tipos de artefactos al modelado.

A continuación se listan los aspectos más importantes del UML [URL3]:

- Representa los sistemas con conceptos de objetos y sus relaciones.
- Divide grandes sistemas en modelos de paquetes, para entender y controlar sus dependencias.
- Es una herramienta para desarrollar el modelado completo de un sistema.
- Es un estándar abierto.
- Soporta la totalidad del ciclo de vida de desarrollo del software.
- Está basado en la experiencia y necesidades de la comunidad de usuarios.
- Actualmente es soportado por diversas herramientas.

En conclusión, UML unifica esencialmente los métodos de Grady Booch, Ivar Jacobson y James Rumbaugh para obtener un lenguaje de modelado de propósito general. UML pretende abordar los problemas actuales del desarrollo del software como gran tamaño, distribución, concurrencia, patrones y desarrollo de equipo.

Grady Booch, Ivar Jacobson y James Rumbaugh trabajaron también en la creación de un proceso unificado llamado Objectory, que en la actualidad se conoce como Proceso Unificado de Rational (RUP, *Process Unified Rational*) y que se describe a continuación.

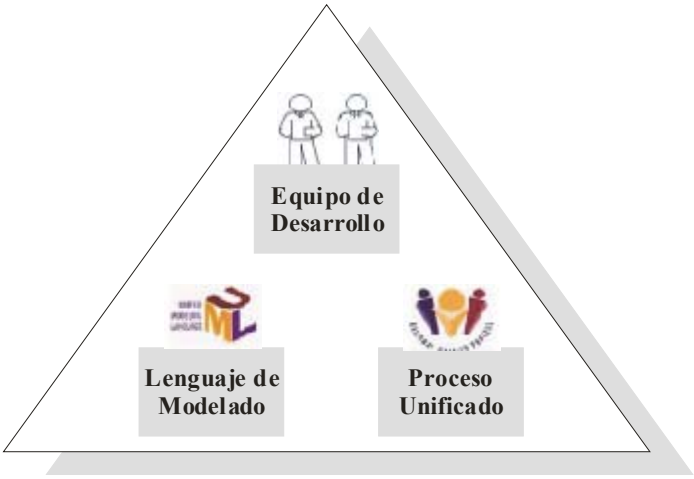


Figura 2.4. Elementos del proceso de desarrollo [URL3].

2.4. Proceso Unificado de Rational

El RUP fue propuesto por los creadores del UML (subcapítulo 2.3) y describe una serie de pasos para llegar al resultado final (*quién* está haciendo *qué*, *cuándo* lo hace y *cómo* alcanzar un objetivo).

Tanto RUP como UML son herramientas de ayuda en el desarrollo de un sistema, sin embargo, para realizar un sistema completo se requiere de un tercer elemento conocido como equipo de desarrollo. En la figura 2.4 se presenta un esquema que integra los elementos del proceso de desarrollo y donde cada elemento realiza una función particular:

- UML proporciona un lenguaje de modelado visual.
- RUP proporciona la metodología de desarrollo del sistema.
- El equipo de desarrollo es la parte vital del sistema debido a que se encarga del análisis y diseño del sistema y de la elección de las herramientas a utilizar.

El RUP cumple con las siguientes funciones:

- Proporcionar una metodología para la ejecución de las actividades de los equipos.
- Especificar qué artefactos deben ser desarrollados y cuándo deben realizarse.
- Dirigir las tareas de los desarrolladores, individual o en equipo, como una sola.
- Ofrecer criterios para monitorear, medir productos y actividades del proyecto.

2.4.1. Evolución del RUP

El proceso unificado es el resultado de tres décadas de investigación, desarrollo y práctica. En 1967, la firma Ericsson desarrolló software en base a requerimientos críticos, la interconexión de bloques, responsabilidades y actividades.

En 1987, Ivar Jacobson retomó la metodología de Ericsson y estableció Objectory, abreviación de *Object Factory*, conocido como desarrollo basado en componentes. Los trabajos sucesivos del proceso Objectory (*Objectory Process*) liberaron la versión 1.0 y en 1997 se dispuso de la versión 3.8 en línea.

Rational Software Corporation adquirió Objectory en 1995 y emprendió la tarea de unificar los principios básicos del proceso de desarrollo de software en la metodología Objectory, resultado de lo anterior fue la versión Rational Objectory Process 4.1.

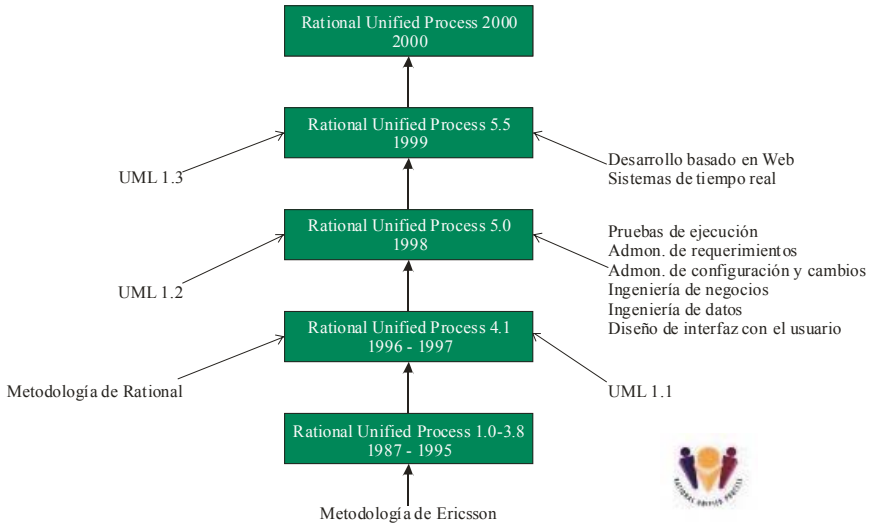


Figura 2.5. Evolución del RUP [24].

Al mismo tiempo se desarrollaba UML como un lenguaje de modelado del proceso de Rational Objectory (ROP, *Rational Objectory Process*), el cual incorporaba lo mejor del modelado de negocios usando los requerimientos de los procesos[5], además de extender el diseño de interfaces de usuario mediante casos de uso. La figura 2.5 presenta un diagrama de la evolución del RUP.

Para 1998, el RUP ya soportaba el ciclo de vida completo del desarrollo de software al unificar una gran variedad de contribuciones. En junio del mismo año, Rational liberó una nueva versión, Rational Unified Process 5.0, que posteriormente agregó un desarrollo basado en tiempo real, versiones 5.5 y 2000, las cuales soportan la versión de UML 1.3.

2.4.2. Objetivos del RUP

El RUP es un proceso de ingeniería de software que aprovecha la asignación, el manejo de tareas y las responsabilidades dentro de una organización para producir software de alta calidad. El RUP captura las mejores prácticas en el desarrollo moderno de software y las presenta en un amplio rango de proyectos dentro de las organizaciones.

En la actualidad, el software es el combustible que hace crecer los negocios, aplicar reglas y unir las sociedades. Globalmente el software es una pieza de progreso que ayuda a manejar la economía mundial. Una sociedad demanda diferentes clases de software en base a su tamaño, complexión e importancia.

A continuación se mencionan las fallas más comunes que se presentan en el desarrollo de software [26]:

- Entendimiento inadecuado por los usuarios finales.
- Incapacidad para modificar los requerimientos.
- Módulos no ajustados.
- Software difícil de implementar o adaptar.
- Calidad deficiente del software.
- Funcionamiento incorrecto del software.
- Poco entendimiento entre los desarrolladores.

[5] Condición o capacidad con el que un sistema debe conformarse [17].

Una alternativa para mejorar la calidad de un software en particular consiste en implementar algunas de las técnicas que se definen a continuación [26]:

- *Desarrollo de software interactivo*: uno de los métodos clásicos de desarrollo de software es el ciclo de vida de cascada, el cual se lleva a cabo mediante un desarrollo lineal de las etapas que lo conforman (análisis, diseño, codificación y sistemas de pruebas). sin embargo su implementación representa riesgos para el proyecto.

  El ciclo de vida de cascada es un proceso interactivo e incremental, en el que es posible identificar el riesgo del proyecto y manejarlo eficientemente mediante el continuo descubrimiento, invención e implementación de cada uno de los artefactos del proyecto.

- *Manejo de los requerimientos*: los cambios que se presentan durante el ciclo de vida del proyecto, con respecto al manejo de requerimientos del software, fortalecen el sistema final, lo anterior se debe a que es posible identificar los requerimientos y asignarles un peso dependiendo de su importancia en el sistema. Las actividades del manejo de requerimientos son: obtener los requerimientos, organizarlos y documentarlos dentro del sistema.

- *Uso de componentes basados en la arquitectura*: visualizar, especificar, construir y documentar un software demanda, contar con gran número de perspectivas de diseño, en donde cada método brinde una solución diferente dentro de la vida del proyecto. Por lo anterior, el documento más importante, para controlar el proceso interactivo e incremental del desarrollo del software dentro de su ciclo de vida, sea la arquitectura. La arquitectura del sistema es de gran ayuda para la toma de decisiones en cuanto a:

  - La organización del sistema.
  - La sección de la estructura de los elementos e interfaces que componen el sistema.
  - El comportamiento del sistema, al especificar como colaboran los elementos.
  - La composición estructural del sistema y el comportamiento de sus elementos dentro de los avances progresivos de los subsistemas.
  - El estilo de organización que guía la arquitectura, sus elementos, interfaces, colaboraciones y su composición.

  La arquitectura del software no sólo se usa para representar la estructura y comportamiento, sino también para especificar la funcionalidad, resistencia, rehúso, comprensibilidad, economía y contraste tecnológico.

- *Modelado de software visual*: como se mencionó con anterioridad, un modelo es una simplificación de la realidad que describe completamente el sistema desde una perspectiva en particular. Un modelo es importante porque ayuda al equipo de desarrolladores a visualizar, especificar, construir y documentar la estructura y el comportamiento de la arquitectura de un sistema. Las ventajas son mayores si al modelar un sistema se emplea una herramienta estándar como UML, ya que los desarrolladores pueden comunicar sus decisiones sin ambigüedad.

  Las herramientas de modelado facilitan el manejo de estos modelos debido a que logran exponer detalladamente, aspectos necesarios para comprender el modelado. Un modelado visual ayuda a lograr cierta consistencia entre los artefactos del sistema como son los requerimientos, los diseños y la implementación.

- *Continua verificación de la calidad del software*: el costo de corregir los errores de un software es directamente proporcional al tiempo empleado en corregirlos, por esta razón es importante tener una revisión continua de la calidad del sistema con respecto a la funcionalidad, rentabilidad, comportamiento de la aplicación y sistema de funcionamiento.

Para verificar la calidad del software es necesario diseñar pruebas (*test*) para cada escenario, en donde cada escenario representa algún aspecto deseado del sistema.

- *Control de cambios al software*: el manejo de cambios en el software, con respecto a coordinar las actividades, desarrollar artefactos y establecer flujos de trabajo, permite asignar, de forma interactiva, los recursos en base a prioridades y riesgos. El completar desarrollos de software interactivo proporciona un monitoreo continuo de los cambios que pueden ayudar a descubrir y reaccionar ante los problemas, el control de los cambios en software ofrece soluciones a los problemas inherentes a su desarrollo.

El definir la metodología a seguir ayuda al desarrollador a tener un control más exacto del desarrollo del proyecto, sin embargo existen algunos aspectos en los cuales el desarrollador tiene que improvisar para lograr un buen desarrollo.

El RUP organiza los proyectos en términos de flujos de trabajo y de fases, las cuales consisten en una o más iteraciones a lo largo del ciclo de desarrollo de software y en cada interacción se hace énfasis en un flujo de trabajo particular. Esta metodología contempla a cada iteración como un miniproyecto, ya que en cada una se realiza el análisis, diseño, codificación y pruebas del sistema en desarrollo. La finalidad de esta metodología es reducir el riesgo del producto o sistema.

En conclusión el RUP [26]:

- Es una guía del proceso a desarrollar.
- Es interactivo e incremental.
- Maneja casos de uso.
- Es diseñado para ser flexible y adaptable.
- Permite una variedad de estrategias de ciclos de vida del software.
- Elige qué artefactos producir.
- Define actividades y trabajadores.
- No es un proceso universal.

El RUP proporciona los lineamientos a seguir en el desarrollo de software basándose en el UML y en un equipo de desarrollo, logrando con ello abarcar los puntos necesarios para la construcción de software. El éxito del RUP se debe a que logra una buena organización mediante la eliminación de los errores más comunes en el proceso de desarrollo del software y a la implementación de prácticas que ayudan a mejorar dicho desarrollo.

2.5. Metodología RUP base

Un software de calidad requiere de personas con orientación y principios para lograr el éxito. El paradigma orientado a objetos ofrece nuevas metodologías para el desarrollo de software. En la actualidad la metodología más utilizada para el desarrollo de software es RUP y cuenta con las siguientes etapas base del desarrollo OO [5]:

- *Análisis*: en esta etapa se define el problema y, básicamente, responde a lo qué se pretende hacer. El resultado obtenido son los modelos que describen a los actores, procesos e interacciones involucrados en el problema.
- *Diseño*: el objetivo de esta etapa es obtener los modelos del diseño mediante los modelos de análisis, tópicos y restricciones del sistema para especificar cómo se procederá al desarrollo.
- *Implementación*: en esta etapa el programador se encarga de codificar el modelo en un lenguaje de programación, que posteriormente tendrá como salida un programa ejecutable.

- *Pruebas*: esta etapa es de suma importancia debido a que representan la evaluación exhaustiva del software final y asegura el cumplimiento de los objetivos iniciales.

Es importante resaltar que estas etapas se desarrollan repetidamente durante el ciclo de desarrollo del sistema para lograr la creación de un software de calidad.

A continuación se describen algunos de los artefactos generados como parte de la metodología RUP:

- *Requerimientos*: se encargan de describir las condiciones o capacidades que conforman al sistema y consideran:
  - *La visión*: documento narrativo que presenta un panorama general de los objetivos que se pretenden alcanzar al desarrollar el sistema.
  - *Las especificaciones*: describen los requerimientos del sistema y de los usuarios.
  - *El glosario de términos*: describe el significado de los términos técnicos o poco comunes que se utilizarán en desarrollo del sistema.
- *Casos de uso*: un caso de uso es una secuencia de acciones que dan un resultado observable para un actor particular y se consideran como las ilustraciones gráficas de los requerimientos del sistema [4, 11]. UML especifica formalmente la notación usada por los casos de uso.
- *Modelo conceptual*: su misión es ilustrar los conceptos referentes al dominio de un problema y a los artefactos más importantes, creados durante el análisis orientado a objetos, dicho modelo representa el funcionamiento de sistema mediante las relaciones de clases y las asociaciones[6] identificadas.

2.6. Modelado del software SepiGPIB

En general, la creación de programas informáticos abarca un gran número de metodologías y el desarrollo del presente trabajo de tesis incluye el modelado del SepiGPIB mediante la metodología del RUP para desarrollo de software OO.

El SepiGPIB utiliza el RUP para elegir qué artefactos y cuándo deben ser realizados. Además, cuenta con un método de desarrollo interactivo que integra elementos en forma progresiva. El SepiGPIB utiliza los formatos del RUP para documentar, organizar y explicar los procesos desarrollados. La herramienta de modelado empleada es UML, ya que brinda los lineamientos para construir y documentar el sistema en desarrollo.

El objetivo de utilizar RUP y UML, en el desarrollo de este sistema, es brindar un modelado confiable, fácil de modificar o retomar para algún desarrollo futuro y garantizar un software robusto, en este caso el modelado es una herramienta utilizada para comprender el desarrollado del software SepiGPIB, objetivo planteado en el presente trabajo de tesis.

Los dispositivos de medida que implementa el SepiGPIB son un generador de funciones modelo 33120A [21, 22], un multímetro digital modelo 34401A [19, 20] y una fuente de alimentación modelo E3610A (Anexo B.3), todos de la firma Agilent Technologies. Su selección se basó en que son los instrumentos de medida con que cuenta el Laboratorio de Comunicaciones Digitales de la Universidad Tecnológica de la Mixteca.

[6] Una asociación representa las relaciones entre instancias de clases e indican una conexión significativa entre las instancias, una asociación es bidireccional, es decir, que existe una conexión entre los objetos de una clase y los de la asociada.

3. Instrumentación programable

3.1. Antecedentes de la instrumentación

El proceso de medición requiere el uso de instrumentos como medios físicos para determinar la magnitud de una variable. Los instrumentos constituyen una extensión de las facultades humanas y en muchos casos permiten a las personas determinar el valor de una cantidad desconocida, la cual no podría medirse utilizando solamente las facultades sensoriales, por lo tanto, un instrumento se puede definir como un dispositivo para determinar el valor o la magnitud de una cantidad o variable. El instrumento electrónico, como lo indica su nombre, se basa en principios eléctricos o electrónicos para efectuar alguna medición, puede ser un aparato relativamente sencillo y de construcción simple o un sistema complejo. El desarrollo de la tecnología demanda la existencia de nuevos diseños e instrumentos con mayor exactitud[7] y precisión[8].

Ninguna medición se puede realizar con una exactitud perfecta, pero es importante descubrir el tipo de error, el cual puede ser de origen humano, sistemático o por causas que no se pueden establecer debido a las variaciones en los parámetros [12].

3.1.1. Estándares IEEE

El Instituto de Ingenieros Eléctricos y Electrónicos (IEEE, *Institute of Electrical and Electronics Engineers*) publica y conserva un conjunto de estándares de diversos tipos, por ejemplo especificaciones de controles, funciones, etc., y recomendaciones para el uso correcto de equipos de prueba, existen estándares relativos a la seguridad del alambrado de fábricas de energía, barcos, edificios industriales, etc.

Uno de los estándares más importantes dentro de la instrumentación electrónica es el estándar IEEE 488 [3]. La estandarización de está interfaz permite la conexión entre sistemas de medida de diferentes fabricantes, dando origen a los sistemas ATE.

---

[7] Grado de aproximación o conformidad al valor real de la cantidad medida.

[8] Grado de concordancia dentro de un grupo de mediciones o instrumentos.

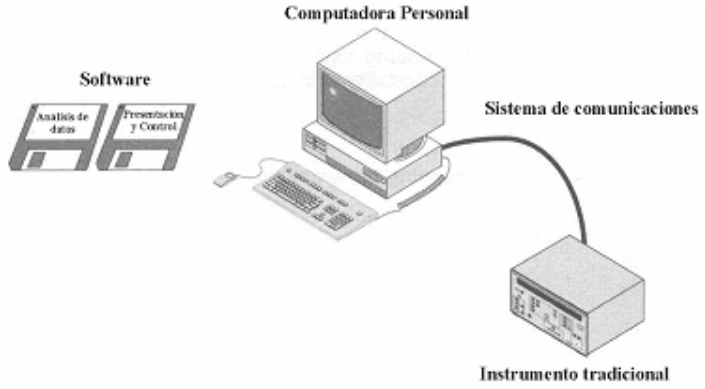


Figura 3.1. Componentes básicos de un sistema ATE [36].

3.2. Instrumentación electrónica

Los procesos en los que se apoyan la ciencia y tecnología generan variables físicas, las cuales se pueden medir mediante instrumentos que tienen como misión determinar la magnitud de una variable, visualizarla, generarla o convertirla en otra diferente.

La electrónica aplicada es el área de la tecnología que estudia las características de los dispositivos electrónicos y la forma de interconectarlos. El avance en la microelectrónica ha permitido convertir las señales de variables físicas (distancia, velocidad, temperatura, densidad, etc.) en señales eléctricas. De acuerdo con esto, los instrumentos electrónicos se pueden clasificar en tres grandes grupos [33]:

- *Instrumentos de medida y visualización*: son los sistemas electrónicos que realizan la evaluación de uno o varios parámetros de una señal eléctrica y los presentan en forma gráfica, numérica o alfanumérica, por ejemplo el osciloscopio, voltímetro, etc.
- *Instrumentos generadores de señales*: tienen como misión generar señales eléctricas de características determinadas, un ejemplo de este tipo de instrumento es el generador de funciones.
- *Instrumentos convertidores de señales*: son dispositivo o circuitos electrónicos que convierten una señal eléctrica o no eléctrica, en otra señal eléctrica de características y rango determinado, ejemplo de ello son los sensores y actuadores.

Los sistema de adquisición de datos (DAQ) se utilizan para medir y registrar señales obtenidas a partir de mediciones de cantidades eléctricas y señales originadas a partir de instrumentos convertidores (transductores).

Los sistemas de instrumentación se pueden clasificar en dos clases principales analógicos y digitales. El tipo de sistema de adquisición, así como también el instrumento a utilizar, depende del tipo de datos que se desea registrar. Los sistemas de adquisición de datos se utilizan en un gran número de aplicaciones, en diversas áreas industriales y científicas.

En el campo de la electrónica, uno de los avances más valiosos es el desarrollo de complejos sistemas de prueba y evaluación controlados por computadora, conocidos como sistemas ATE, los cuales básicamente se conforman de los siguientes elementos (figura 3.1):

- *El equipo de medida tradicional*: se encarga de efectuar las mediciones necesarias.
- *Una computadora con software especifico*: para realizar el análisis y procesamiento deseados, y presentar los datos en forma correcta.
- *Un sistema de comunicaciones*: permite la comunicación entre la computadora y el equipo de prueba.

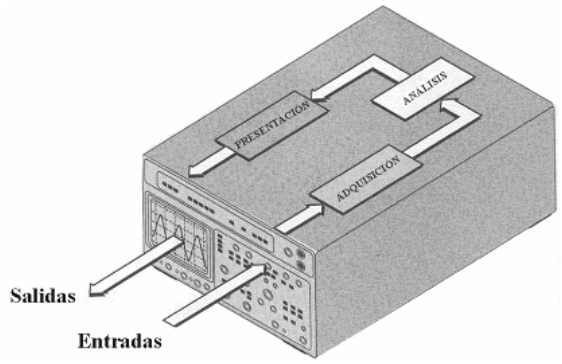


Figura 3.2. Instrumento tradicional [36].

3.2.1. Evolución de la instrumentación electrónica

Desde la aparición de los primeros instrumentos de medida, los cuales eran controlados mediante sus paneles frontales, éstos han adquirido gran flexibilidad y un alto grado de integración para interconectarse con otros instrumentos en sistemas más complejos, dependiendo de su funcionalidad, los instrumentos se pueden clasificar para:

- *Adquisición de datos*: se utilizan para acondicionar, medir y registrar señales de las líneas de entrada del instrumento.
- *Análisis de datos*: se realiza el análisis y procesamiento a la señal obtenida.
- *Presentación*: despliegan la señal de salida.

A los instrumentos que pertenecen a la primera generación se les conoce como instrumentos tradicionales (figura 3.2), estos instrumentos limitaban la manipulación y procesamiento de medidas debido a que funcionaban manualmente a través de un panel frontal.

Generalmente, cada instrumento se diseña para realizar una medición en específico, por lo que los usuarios pueden incrementar el número de instrumentos para crear un sistema completo de medición, sin embargo el espacio de trabajo requerido aumenta considerablemente. Con el origen del bus de interfaz de propósito general (GPIB) los usuarios son capaces de controlar sistemas de instrumentación de forma remota mediante una secuencia de órdenes, dando origen al término de instrumentación electrónica programable.

Los instrumentos electrónicos programables son una combinación de computadoras de propósito general e instrumentos tradicionales y de nueva generación, estos modernos instrumentos pueden alcanzar nuevos niveles de ejecución y flexibilidad al combinar rutinas de software con nuevas clases de hardware de instrumentación, además, con la implementación de nuevas técnicas de control de instrumentos, como son la programación basada en registros y memoria compartida, se puede optimizar el uso de la computadora para un procesamiento avanzado al adquirir, analizar y presentar resultados de las mediciones requeridas en forma virtual, dando origen al término instrumentación virtual (VI, *Virtual Instrumentation*).

Con los instrumentos virtuales, se puede tener una gran variedad de dispositivos de instrumentación, el cual se puede respaldar con rutinas de software que permiten la creación de interfaces gráficas de usuario (GUI, *Graphic User Interface*) de alta resolución y flexibilidad.

En la mayoría de casos, el término de instrumentación programable es sinónimo de VI y en casos específicos se puede definir como un sistema electrónico de medida, o generación de variables, basado en un procesador digital en cuya memoria se sitúa un programa que automatiza la realización de las medidas [33].

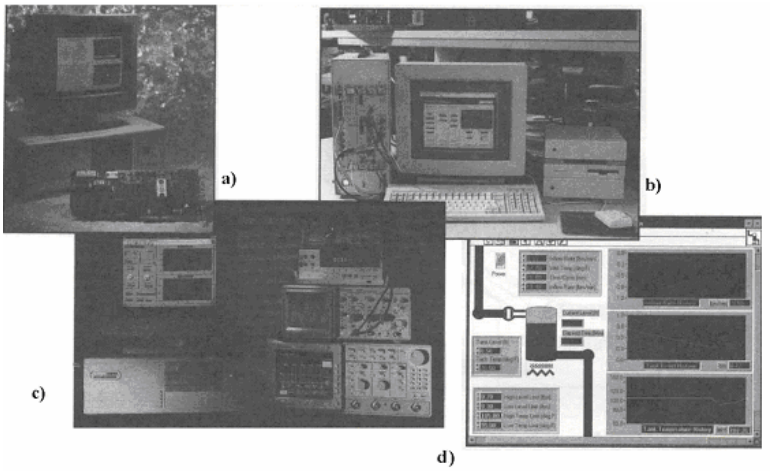


Figura 3.3. a) Tarjeta de adquisición de datos (DAQ), b) Equipo de adquisición de datos, c) GUI en una PC controlando instrumentos mediante comunicación GPIB, d) GUI sin instrumentos físicos [36].

Las siguientes son definiciones de la instrumentación electrónica programable y virtual respectivamente (figura 3.3):

- *Instrumentación electrónica programable*: se define como una GUI en una PC controlando a un instrumento tradicional mediante un medio de comunicación GPIB o RS-232, o una GUI en una PC controlando a una tarjeta de adquisición de datos (DAQ) de tipo plug-in o a un modulo VXI mediante la comunicación GPIB.
- *Instrumentación virtual*: se define como una GUI sin instrumentos físicos conectados a una computadora personal.

En algunas aplicaciones es necesario utilizar las medidas de los instrumentos para la toma de decisiones en cuanto al control de procesos de fabricación, de gestión, etc., lo cual ha dado origen a los sistemas ATE.

Para comprender la instrumentación electrónica, es importante conocer como han evolucionado los instrumentos a lo largo de los años. La Tabla 3.1 presenta algunos hitos y eventos importantes en historia de la instrumentación electrónica.

3.3. Bus de interfaz de propósito general

La interfaz IEEE 488, conocida ampliamente como GPIB, fue diseñada para integrar uno o más instrumentos a una computadora o controlador, desde su diseño se contempla al bus como medio de simplificación en la realización de pruebas de ingeniería implementadas en sistemas ATE.

En el bus GPIB se pueden conectar hasta 15 instrumentos o dispositivos, los cuales se comunican unos con otros bajo una configuración maestro/esclavo mediante cables y conectores requeridos por el bus. El control del sistema lo realiza un dispositivo maestro, llamado controlador, el cual generalmente es una computadora personal o un controlador del bus dedicado basado en un microcontrolador. El software requerido para el sistema puede ser implementado en cualquier lenguaje de programación, por ejemplo: Borland C++, C++ Builder, Borland Delphi, Visual C, etc.

Hoy en día, el bus GPIB contempla un protocolo que, mediante pocas órdenes, permite a los usuarios diseñar sistemas simples para realizar pruebas de medición complejas.

3.3.1. Historia del GPIB

El avance de la microelectrónica ha incrementado la complejidad y la capacidad de medición de los dispositivos, además de contar con una gran variedad de funciones para realizar pruebas.

Tabla 3.1. Evolución de los buses de instrumentación.

Año	Evento histórico
1960	Bucle de corriente de 4-20mA para sensores y actuadores, basado en la red télex de 1920 (teletipo o TTY). Norma ANSI MC12.1 e ISA S50.1. Posteriormente fue incorporado al protocolo HART para sensores inteligentes.
1965	Hewlett-Packard desarrolla un bus paralelo para conectar instrumentos de medida a una computadora, denominado HP-IB (*Hewlett-Packard Instrument Bus*). La computadora actúa como controlador y tiene la capacidad de transmitir órdenes a los instrumentos y leer de los mismos las medidas realizadas.
1969	Conexión serie IEA RS232-C (también CCITT V.24 y V.28), que se convirtió en el "puerto serie" de propósito general para comunicarse a baja velocidad (9600 Kbps) en distancias cortas (10m). Evolución a la norma IEA RS-485 que en los años 80 se implementa como la capa física de los buses de campo (*FieldBus*) de dos hilos, dando un gran impulso a las comunicaciones industriales.
1975	Primer sistema de tarjetas de bus común de propósito general S-100 de la firma MITS para el µC 8080 de Intel (Norma IEEE 696). El IEC y el IEEE adoptan el bus de instrumentación programable HP-IB bajo las normas IEC 625 e IEEE 488 respectivamente. Este bus pasa a denominarse GPIB.
1978	La ISO presenta un modelo de interconexión de sistemas abiertos (OSI) para el desarrollo de protocolos normalizados en comunicaciones (LAN, MAN y WAN).
1981	Bus común (*mother board*) para PC´s denominado ISA propuesto por IBM (norma IEEE 996). Desarrollo generalizado de tarjetas conectables al PC para aumentar sus funciones y convertirlo así en una plataforma multipropósito.
1983	Bus FASTBUS de los comités NIM y ESONE para experimentación en física de altas energías (Normas IEEE 960 e IEC 935).
1984	General Motors impulsa la norma MAP (*Manufacturing Automation Protocol*) para el desarrollo de un protocolo de mensajes de fabricación en tiempo real en el nivel de aplicación OSI. Aparición de numerosos buses de campo.
1985	Ante la falta de compatibilidad entre controladores e instrumentos GPIB de diferentes fabricantes, Tektronix propone un conjunto de formatos estándares para órdenes y respuestas de instrumentos programables GPIB.
1987	La propuesta de Tektronix es adoptada por el IEEE bajo la norma IEEE 488.2 de protocolos de intercambio de mensajes, formatos de datos, sintaxis, informes de estado y órdenes comunes a distintos tipos de instrumentos programables. La norma original IEEE 488 pasa a denominarse IEEE 488.1. Bus común VXI basado en VME para instrumentación programable modular con instrumentos en forma de tarjetas. Creación del consorcio VXI (*VXI consortium*).
1989	National Instruments desarrolla el bus de cable MXI (*Multisystem Instrument Interface*) proporcionando así un mecanismo de altas presentaciones para interconectar sistemas de instrumentación programable VXI y controlarlos desde una PC.
1990	Propuesta de la norma SCPI (*Standard Commands for Programmable Instruments*) que especifica un extenso conjunto de órdenes en formato ASCII (*American Standard Code for Information Interchange*) para manejar la gran variedad de funciones que realizan los instrumentos programables. Junto con IEEE 488.2 permite la compatibilidad entre instrumentos modulares en tarjetas VXI. Creación del consorcio SPCI (*SCPI Consortium*) y desarrollo acelerado de herramientas de programación compatibles en instrumentación programable.
1992	Bus de interconexión SCI para enlazar sistemas de bus común en aplicaciones distribuidas de alta velocidad (IEEE 4596). Proyecto ISP (*Interoperable System Project*) para especificar una norma de bus de campo global que dio lugar al estándar ISA/IEC SP50.
1993	El IEEE adopta el bus VXI como norma IEEE 1151. Fundación de la alianza *VXI&plug&play* para buscar un mayor nivel de estandarización con respecto a todos los componentes de un sistema de instrumentación programable VXI.
1997	Bus común PXI (*PCI eXtensions for Instrumentation*) basado en PCI propuesto por National Instruments para instrumentación programable en tarjetas. Norma IEEE 802.11 para redes inalámbricas de corto alcance (WLAN). National Instruments propone la especificación HS 488 para aumentar la velocidad del bus de instrumentación programable GPIB (IEEE 488.1 y 488.2) a 8 MBps.
2000	La división de instrumentación de Hewlett-Packard para a ser Agilent Technologies.

Al tratar de eliminar el elemento humano de las pruebas de instrumentación, que cada vez son más complejas, se necesita que los instrumentos tengan la capacidad de transmitir datos (hablar) y recibir datos (escuchar). Esto originó que las empresas consideraran la necesidad de desarrollar una interfaz que permitiera la comunicación entre instrumentos de distintos fabricantes.

En el año 1965, la firma Hewlett-Packard, en la actualidad Agilent Techologies, dio a conocer un estándar que denominó HP-IB (*Hewlett-Packard Instrumentation Bus*), el cual especifica la forma de interconectar sus sistemas de instrumentación mediante el uso de una computadora, ésta actúa de controlador y tiene la capacidad de transmitir órdenes a los instrumentos y leer de los mismos las medidas realizadas. Sin embargo la falta de un estándar que compatibilizara la interconexión de las computadoras con los sistemas de instrumentación de distintos fabricantes hizo

que a partir de 1972 el IEEE iniciara una normalización, el resultado fue el estándar IEEE 488-1975 conocido como bus de interfaz de propósito general, o simplemente GPIB. Esta norma constituye un conjunto de especificaciones eléctricas, mecánicas y funcionales.

La norma GPIB fue adoptada en Europa bajo la denominación IEC 625.1 por la Comisión Electrotécnica Internacional (IEC, *International Electrotechnical Commission*). En enero de 1976, el Instituto Nacional de Estándares Americanos (ANSI, *American National Standard Institute*) publicó un estándar idéntico denominado MC1.1 [33].

En el año 1987 se logró un avance significativo al conseguir sistemas más fiables y fáciles de programar, por lo que el IEEE propuso modificaciones, estándar IEEE 488.2, para crear sistemas compatibles y desarrollar programas flexibles mediante la estandarización de formatos y código de datos, por lo tanto, IEEE 488.2 define la forma en la que los controladores envían las órdenes y cómo los instrumentos responden a ellas mediante la especificación de un conjunto de órdenes comunes para todos los instrumentos y los requerimientos del protocolo. Al mismo tiempo, en el área militar, esta especificación se incorporó al programa de pruebas de equipo automatizadas modularmente (MATE, *Modular Automated Test Equipment*) [33].

Paralelamente a los avances en el desarrollo de sistemas ATE compatibles, se realizaron esfuerzos para aumentar la capacidad de procesamiento de los instrumentos GPIB y la velocidad de transferencias en el bus, hasta 1MBps.

Años después, y como respaldo del estándar obtenido, se formó un consorcio de empresas fabricantes de instrumentos programables cuya principal aportación ha sido la especificación de órdenes estándares para instrumentos programables (SCPI, *Standard Commands for Programmable Instruments*), la cual define un conjunto de órdenes de programación para instrumentos de diferentes fabricantes. Las SCPI especifican el procedimiento para el intercambio de mensajes, el formato de datos y el modelo de reporte de estados, la figura 3.4 muestra la relación entre los estándares IEEE 488.

La evolución del GPIB fue la base para que otras arquitecturas lograran su estandarización, por ejemplo VXI, basado en el bus VME, es una arquitectura de instrumentos modulares implementados en tarjetas que se insertan en un chasis, este sistema de instrumentación alcanza velocidades de hasta 40 Mbps. La norma VXI logró su estandarización en el año 1987 con el apoyo del consorcio VXI. El estudio de GPIB y VXI ha alcanzado una enorme expansión permitiendo el diseño de complejos sistemas ATE implementados en diversas plataformas de computadoras bajo diferentes sistemas operativos.

En el año 1990 la propuesta SCPI e IEEE 488.2 fueron adoptadas como estándares de compatibilidad entre instrumentos GPIB e instrumentos modulares en tarjetas VXI.

Debido a la amplia utilización y a los avances tecnológicos respecto al GPIB, en 1993 la compañía National Instruments presentó la especificación HS 488 destinada a conseguir una velocidad máxima de 8 MBps compatible con el conjunto de especificaciones del protocolo IEEE 488.1 y 488.2, en el mismo año, IEEE adopto al bus VXI bajo el estándar IEEE 1551.

Dentro de las soluciones software destinadas al diseño de sistemas ATE bajo el GPIB, cabe destacar NI-488.2, diseñado por National Instruments, que soporta 20 plataformas de computadoras y más de 25 sistemas operativos diferentes, proporcionando gran flexibilidad para la adaptación de aplicaciones y sistemas operativos, también incrementa la productividad y asegura la compatibilidad con los lenguajes de programación modernos al combinar un conjunto de herramientas para aplicaciones en tiempo real y de alto desempeño mediante la utilización de controladores (*drivers*) para núcleos (*kernel*) de 32-bit destinados a sistemas operativos de Win32 (Windows 2000/NT/XP/Me/9x) y otras versiones como pueden ser Windows 3.1, MS-DOS, Solaris, Unix, OS/2, etc. [1, 36].

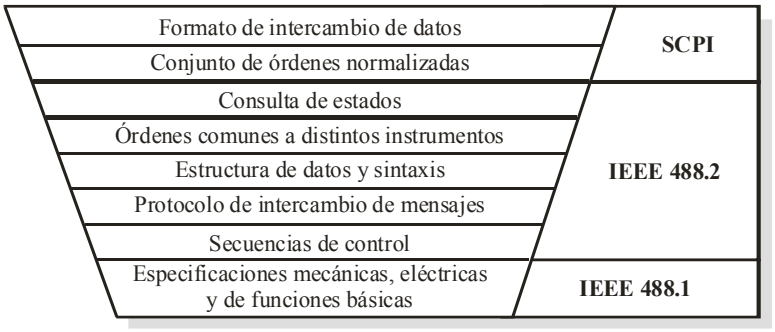


Figura 3.4. Relación entre los estándares IEEE 488.1, IEEE 488.2 y SCPI.

Además, con la aparición del software NI-488.2 [34, 35], se pueden obtener interfaces de programación amigables, flexibles, portables y aplicaciones GPIB distribuidas. Es oportuno hacer mención que a pesar de que el GPIB alcanzo la estandarización hace más de dos décadas, su amplia utilización ha mantenido actualizadas sus aplicaciones, ejemplo de ello es la continua mejora que sufre NI-488.2 para adaptarse a los avances tecnológicos de la instrumentación electrónica, además de las constantes herramientas hardware de conversión a otros protocolos de comunicación como son:

- *PCI (Peripheral Communications Interconnect Local Bus)*: una interfaz PCI provee de una conexión interna de alta velocidad entre los dispositivos interconectados a una computadora, es la interfaz más utilizada para la conexión de sistemas de adquisición de datos basados en plataformas de PC.
- *PXI/CompactPCI (PCI eXtensions for Instrumentation)*: se refiere a las especificaciones basadas en plataformas de computadoras destinadas a la medida y automatización de sistemas, combinan su diseño hardware con un conjunto de herramientas de desarrollo para aplicaciones GPIB permitiendo mayores velocidades de conexión interna.
- *PCMCIA (Personal Computer Memory Card International Association)*: es una interfaz para computadoras portátiles (*laptop* o *notebook*) que incorpora un controlador PCMCIA-GPIB y un analizador GPIB para dar una solución eficiente sin comprometer la flexibilidad y la potencia de procesamiento.
- *ISA (Industry Standard Architecture)*: representa una alternativa de comunicación GPIB y permite la conexión interna de dispositivos GPIB con altas velocidades de transferencia, es compatible con una gran cantidad de sistemas, hardware y software, que implementan el GPIB.
- *USB (Universal Serial Bus)*: es un protocolo ideal para aplicaciones con ranuras de entrada/salida limitadas, por ejemplo computadoras portátiles.

Las interfaces descritas cuentan con distintas combinaciones de hardware y software que les permiten ampliar sus capacidades referentes a la funcionalidad, transmisión de datos, velocidades de transferencia, manejo y operación, para ofrecer una mejor conectividad entre sistemas de instrumentación.

Generalmente, el desarrollo de aplicaciones en un entorno de instrumentación programable, basado en algún protocolo de comunicaciones como GPIB o VXI, se lleva a cabo mediante herramientas software propietarias, ejemplo de ello son la herramienta VEE (*Visual Engineering Enviroment*) de la firma Agilent Technologies y las herramientas LabView y LabWindows de National Instruments, herramientas con un gran respaldo matemático y funciones prediseñadas que facilitan el diseño de GUIs destinadas al control y programación de los instrumentos.

En la actualidad, los fabricantes de instrumentos programables e investigadores se encuentran desarrollando herramientas que mejoren las prestaciones del GPIB en nuevas arquitecturas de instrumentación programable.

4. Bus de interfaz de propósito general

4.1. Estándar IEEE 488

La interfaz IEEE 488 está destinada fundamentalmente a la comunicación entre instrumentos electrónicos programables de medida y estimulación en configuraciones ATE, en algunos casos se emplea en la conexión de un computador a sus periféricos.

La interfaz IEEE 488 se conoce como bus de instrumentación de propósito general o simplemente GPIB[9]. Las características más destacadas de la interfaz IEEE 488 son [33]:

- Permitir la operación simultánea de instrumentos de medida y periféricos de computador sin degradar sus prestaciones específicas.
- Soportar la interconexión de instrumentos de distintos fabricantes para operar conjuntamente.
- Proporcionar un sistema de comunicación asíncrona con un amplio margen de velocidades de transferencia de información.
- Obtener sistemas ATE válidos a distancias cortas y de diferente complejidad, con o sin la existencia de un controlador central.
- Diseñar sistemas de bajo costo que no requieran circuitos de interconexión complejos para los instrumentos más económicos y que al mismo tiempo proporcionen altas capacidades de funcionamiento que necesitan los instrumentos más potentes.
- Ejecutar la transferencia fiable de información entre instrumentos.
- Establecer un conjunto de órdenes y métodos de direccionamiento válidos para todos los instrumentos.
- Habilitar mecanismos de sondeo, serie y paralelo, que proporcionen informes sobre el estado de los instrumentos.

Como ya se presenta en el apartado 3.3.1, paralelamente a los avances en el desarrollo de sistemas ATE se realizaron esfuerzos para aumentar la capacidad del GPIB, principalmente incorporando nuevas modificaciones al estándar IEEE 488, resultado de tales esfuerzos es la capacidad de comunicación que en la actualidad tienen los instrumentos electrónicos de medida.

[9] En el presente documento se utiliza el término GPIB para hacer referencia al protocolo de comunicaciones en general, mientras que el término IEEE 488 hace referencia directamente al estándar del GPIB.

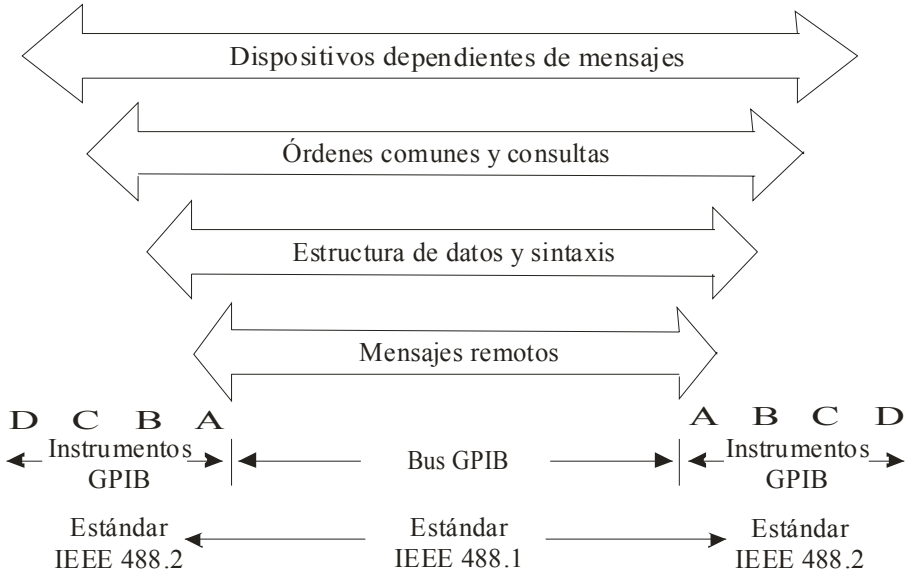


Figura 4.1. Funcionamiento del estándar IEEE 488.

En la actualidad, la mayoría de los fabricantes incorporan a sus equipos electrónicos de medida la capacidad de comunicación GPIB, lo cual los hace más competitivos, un instrumento que no incorpora tal capacidad, está limitado para competir en el vasto mundo de la electrónica y particularmente en el campo de la instrumentación. El uso de un sistema con interfaz GPIB incrementa la eficiencia, reduce costos y elimina errores en la línea de producción.

La figura 4.1 complementa la descripción del estándar IEEE 488 (figura 3.4) al ampliar su funcionamiento bajo el estándar IEEE 488.2.

El estándar IEEE 488.1 contiene las especificaciones mecánicas, eléctricas y funcionales.

El estándar IEEE 488.2 proporciona una comunicación segura, ya que permite la transferencia de mensajes entre dispositivos mediante la ayuda de órdenes estandarizadas, otra función de este estándar es el direccionamiento automático bajo los requerimientos del estándar IEEE 488.1. El estándar IEEE 488.2 fue diseñado para utilizarse como una interfaz mediante la cual los dispositivos conectados al bus GPIB pudieran responder a las órdenes y/o a los servicios de petición.

El conjunto de órdenes estandarizadas para instrumentos programables SCPI define las órdenes de alto nivel a implementarse en todos los instrumentos desarrollados bajo su especificación.

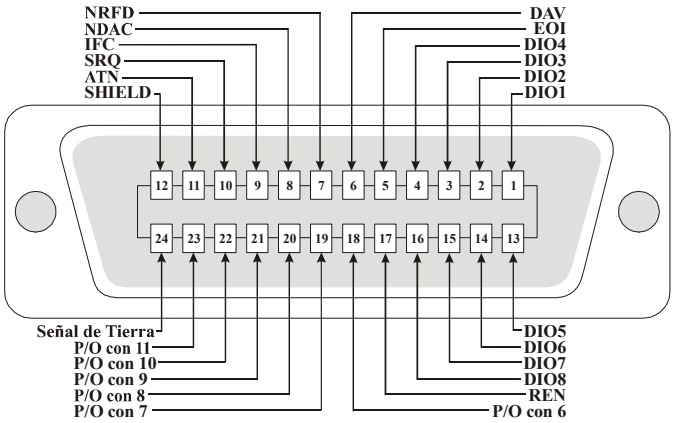


Figura 4.2. Conector IEEE 488, mostrando la identificación de los pines de conexión.

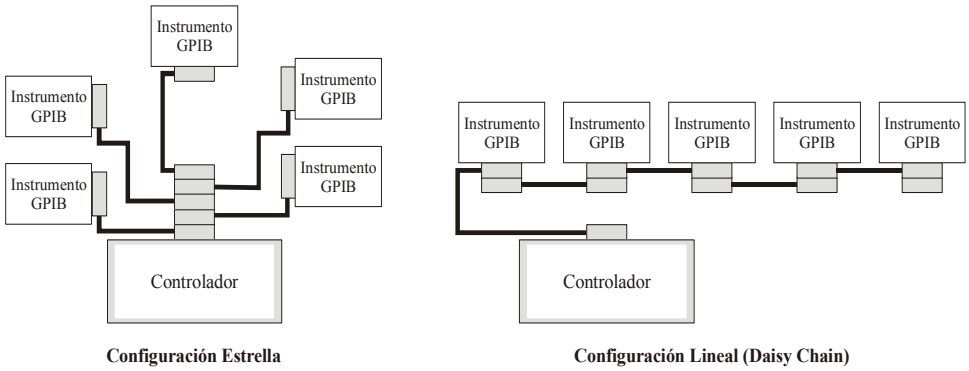


Figura 4.3. Tipos de configuración aceptados por el estándar IEEE 488.

4.2. Estándar IEEE 488.1

El estándar IEEE 488.1 describe un conjunto de especificaciones mecánicas, eléctricas, funcionales y de procedimiento para realizar la interconexión de instrumentos programables [9]. A continuación se describe cada una de ellas.

4.2.1. Especificaciones mecánicas

Las especificaciones mecánicas definen el tipo de conector, la configuración de los instrumentos, la longitud máxima del bus y el tipo de cables que conforman un sistema GPIB.

Los conectores son de tipo americano de 24 pines, figura 4.2, con terminación hembra en los instrumentos y macho en los cables.

Este tipo de conectores permite la construcción de sistemas GPIB bajo la configuración lineal y estrella como se ilustra en la figura 4.3.

El estándar IEEE 488 permite conectar hasta 15 instrumentos activos en el bus, incluyendo el controlador como parte del sistema, la longitud máxima de la red del bus es de 20 m y se recomienda conectar los dispositivos a distancias máximas de 2 m, es posible exceder el límite de 20 m mediante un extensor IEEE 488.

Comercialmente, los cables ofrecen las siguientes características:

- Blindaje ligero, sencillo o doble.
- Cable hembra o macho.
- Longitudes de 1, 2, 4 y 8 m.
- Tipo de conectores.

La especificación determina el tipo de conectores para interconectar a los instrumentos en paralelo mediante el cable IEEE 488, figura 4.4, que contiene 16 líneas activas y que finaliza en un conector especial hermafrodita (*hermaphrodite*).

Figura 4.4. Cable IEEE 488.

4.2.2. Especificaciones eléctricas

Se refieren a los niveles de voltaje de las señales en el bus y son los siguientes:

- "1" lógico en la salida 0 ÷ 0,5 V.
- "1" lógico en la entrada -0,6 ÷ 0,8 V.
- "0" lógico en la salida 2,4 ÷ 5 V.
- "0" lógico en la entrada 2 ÷ 5,5 V.

El sistema GPIB utiliza lógica negativa, es decir, codifica una señal verdadera con un nivel "0" lógico y un señal no verdadera con un nivel "1" lógico. Una razón importante para utilizar esta técnica es permitir a los dispositivos GPIB ser diseñados con transistores de colector abierto, los cuales cumplen con las condiciones anteriores, además, la lógica negativa reduce el ruido en estado verdadero y mantiene en estado falso en cualquier línea desconectada.

4.2.3. Especificaciones funcionales

Definen los tipos de instrumentos conectados al bus (controlador, emisor y receptor), las órdenes, datos, direcciones y funciones básicas, también considera los protocolos o mecanismos de transferencia de información y el mecanismo de reconocimiento.

4.2.3.1. Tipos de instrumentos y señales del bus

El GPIB permite la conexión en paralelo de hasta 15 dispositivos, de los cuales más de la mitad deben estar activos, la velocidad de transmisión máxima es 1 Mbps y disminuye a distancias mayores a las recomendadas (200 a 300 Kbps). Este tipo de conexión requiere la existencia de un elemento físico para el control de las señales, donde cada dispositivo conectado al bus puede actuar como:

- *Receptor (Listener)*: un dispositivo direccionado en este modo de operación sólo acepta datos y órdenes del bus.
- *Emisor (Talker)*: un dispositivo direccionado en este modo de operación envía datos a través del bus a los dispositivos receptores activos, sólo puede haber un dispositivo emisor en cada instante.
- *Controlador (Controller)*: se encarga de la gestión del bus, del envío de órdenes, de solicitar el estado de los dispositivos y del control del flujo de datos. El controlador es el cerebro del sistema GPIB y puede ser una computadora, equipada y configurada para estos propósitos, o simplemente un sistema dedicado basado en microcontroladores que implementan el protocolo IEEE 488.

A continuación se listan algunas de las especificaciones funcionales del bus GPIB:

- Sólo se permite un emisor en el bus a un mismo tiempo.
- Sólo puede estar activo un controlador.
- Los emisores también pueden ser receptores.
- Los receptores usualmente no son emisores.
- El controlador inicialmente tiene un modo de operación de receptor o emisor.
- Un sistema GPIB mínimo consiste en un emisor y un receptor únicamente.
- Las 24 señales del bus se agrupan en ocho líneas de datos, tres de control de transferencia (*handshake*), cinco líneas de gestión del bus (*bus management*) y ocho líneas de tierra, como muestra la Tabla 4.1.

Tabla 4.1. Señales del estándar IEEE 488 para un conector de 24 pines (americano).

Norma IEEE 488 (24 Pines)					Sentido controlador o emisor → receptor ←	Tipo
No. Pin		Descripción de la señal				
Señal	Retorno (Tierra)	Nemónico	Denominación	Función		
1		DIO1	(*Data input - output 1*)	Bit 1	→	Octeto de datos
2		DIO2	(*Data input - output 2*)	Bit 2	→	
3		DIO3	(*Data input - output 3*)	Bit 3	→	
4		DIO4	(*Data input - output 4*)	Bit 4	→	
13		DIO5	(*Data input - output 5*)	Bit 5	→	
14		DIO6	(*Data input - output 6)*	Bit 6	→	
15		DIO7	(*Data input - output 7*)	Bit 7	→	
16		DIO8	(*Data input - output 8*)	Bit 8	→	
6	18	DAV	(*Data valid*)	Dato válido	C,T→	Control de transfe-rencia (*hand-shake*)
7	19	NRFD	(*Not ready for data*)	Instrumento no preparado	C,T←	
8	20	NDAC	(*Not data accepted*)	Dato no aceptado	C,T←	
5		EOI	(*End or identify*)	Fin de transferencia (T) o de consulta (C)	C,T→	Gestión del bus
9	21	IFC	(*Interface clear*)	Inicialización de instrumentos	C →	
10	22	SRQ	(*Service request*)	Petición de servicio	C ←	
11	23	ATN	(*Attention*)	Atención	C →	
17		REN	(*Remote enable*)	Control Local o remoto	C →	
12		SHIELD	(*Shield*)	Blindaje		Tierra
24		LOGIG GND	(*Logic ground*)	Tierra		

La comunicación de los instrumentos conectados al GPIB se realiza a través de un cable que conecta todos los dispositivos a la interfaz en paralelo, este cable contiene 16 líneas activas, de la cuales ocho se usan para la transmisión de datos y las restantes para el manejo y control de las comunicaciones en el bus. La figura 4.5 muestra la estructura del GPIB.

4.2.3.1.1. Señales de datos

Corresponden a las señales DIO1 a DIO8 que se utilizan para transferir secuencias de octetos (*bytes*) de direcciones, datos y órdenes desde un instrumento a otro con la ayuda de las señales de control de transferencia y de gestión del bus. El bit DIO1 es el de menor peso (LSB, *Least Significant Bit*) y el bit DIO8 es el de mayor peso (MSB, *Most Significant Bit*).

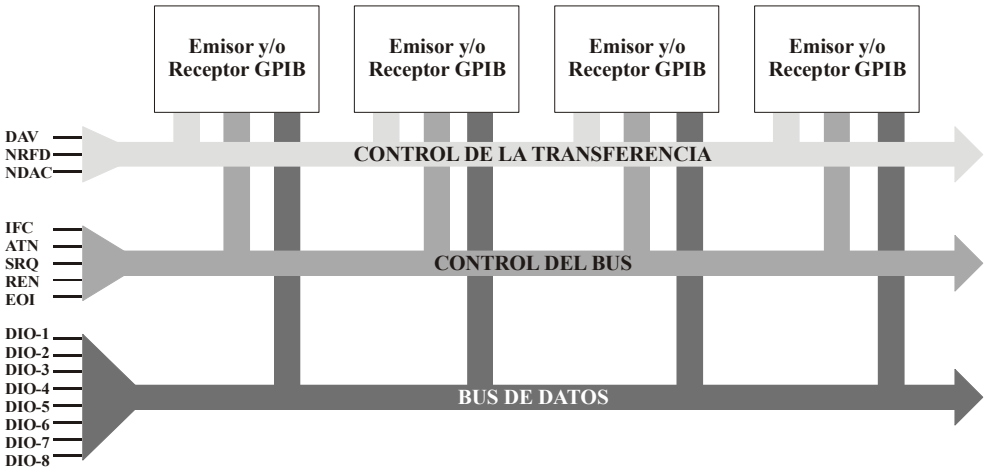


Figura 4.5. Estructura del bus GPIB.

4.2.3.1.2. Señales de control de transferencia

El propósito de las señales de control es asegurar que todos los mensajes transmitidos sean recibidos correctamente por el receptor direccionado. El GPIB utiliza tres de sus líneas para realizar la función de transferencia:

- *NRFD (Not Ready For Data)*: los dispositivos en el bus utilizan esta línea para indicar las condiciones de lectura y aceptación de los datos transmitidos, si un instrumento no está listo para recibir datos por alguna razón, pone a un nivel lógico "0" la línea NRFD para indicar una condición verdadera e inhabilitar al controlador para la transmisión de más datos y únicamente cuando todos los dispositivos en el bus liberan esta línea, poniendo un nivel "1" lógico, se permite la transmisión del siguiente octeto de datos.
- *DAV (Data Valid)*: un emisor usa esta línea para indicar que los ocho bits de datos presentes en el bus son válidos.
- *NDAC (Not Data Acepted)*: esta línea es activada por los instrumentos receptores para indicar que aún no han leído el octeto situado en las líneas de datos.

4.2.3.1.3. Señales de gestión del bus o líneas de manejo de interfaz

El GPIB maneja cinco líneas de gestión de control y de estado para el control del flujo de órdenes y de datos a través del bus. El GPIB asocia un nemónico a cada línea:

- *ATN (Attention)*: el propósito de esta línea es indicar la existencia de datos u órdenes en el bus, cuando el bus está en modo orden (*command*), todos los dispositivos receptores habilitan su capacidad de recibir una transmisión y aceptar la información presente en las líneas de datos, cuando el dato se envía al receptor direccionado, la línea ATN se pone a un nivel "1" lógico (no verdadera), esta línea debe ser monitoreada por todos los dispositivos en el bus y tiene un tiempo de respuesta de 200 nseg.
- *IFC (Interface Clear)*: el controlador usa esta línea para inicializar la interfaz del bus en un modo de inactividad (*stand by*), todos los dispositivos deben monitorear esta línea teniendo un tiempo de respuesta de 100 mseg.
- *REN (Remote Enable)*: el controlador utiliza esta línea para indicar a los dispositivos que la operación a realizar será en modo remoto, cuando REN es puesta a un nivel "1" lógico, todos los dispositivos regresan a su modo de operación local. Todos los dispositivos deben monitorear esta línea con un tiempo de respuesta de 100 μseg.

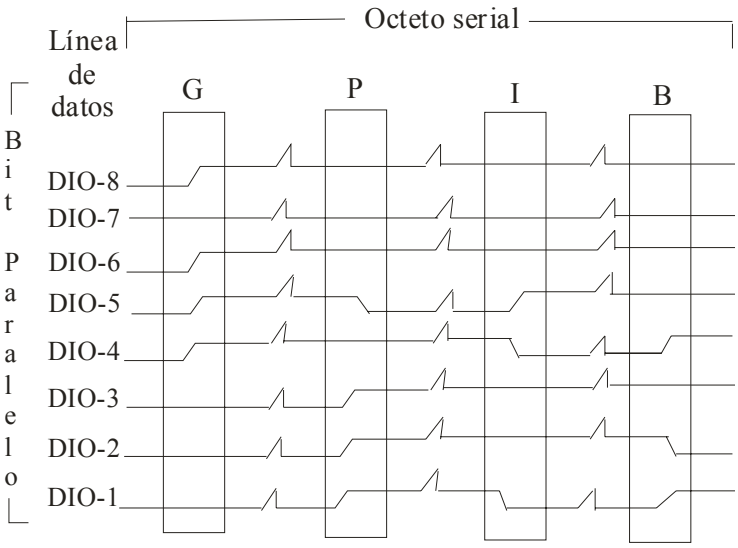


Figura 4.6. Datos GPIB transmitidos en octeto serial y en formato de bit en paralelo.

- *SRQ (Service Request)*: los dispositivos utilizan esta línea para solicitar la atención del controlador debido a un error de sintaxis, sobre flujo, etc., cualquier dispositivo puede activar e interrumpir los eventos en el bus hasta obtener la atención del controlador.
- *EOI (End Or Identify)*: un dispositivo emisor utiliza está línea de dos formas distintas, para indicar que ha enviando el último octeto de datos de un mensaje y, en combinación con ATN, para consultar el estado de los dispositivos activos en el bus.

4.2.3.1.4. Señales de tierra

El estándar IEEE 488 especifica la utilización de tres tipos de señales de tierra:

- *Las señales de retorno*: son seis líneas de tierra (pines 18, 19, 20, 21, 22 y 23 del conector americano) respecto a las cuales se miden las señales DAV, NRFD, NDAC, IFC, SRQ y ATN respectivamente, véase Tabla 4.1.
- *La tierra de chasis (Shield)*: como su nombre lo indica, es la que se conecta al chasis del instrumento.
- *La tierra de de referencia (Logic ground)*: es el pin de tierra respecto a la cual se miden las señales EOI y REN.

Los mensajes y datos transferidos en el bus utilizan la función de transferencia para asegurar la integridad de los datos en un sistema de múltiples transmisiones, el GPIB transfiere los datos de forma asíncrona en un octeto serial y en formato de bit en paralelo, lo cual se ilustra en la figura 4.6.

4.2.3.2. Direccionamiento

Desde el momento en que el bus se inicia como medio de comunicación, el controlador asigna a los instrumentos una dirección única, que se utiliza para transmitir un mensaje o datos a través del bus. Se permite el uso de 31 direcciones, conocidas como direcciones primarias, las cuales se configuran en cada instrumento o dispositivo mediante interruptores (*switches*) o conectores (*jumpers*) localizados en la parte trasera de cada dispositivo, cada posición en el interruptor tiene un valor en decimal de 1, 2, 4, 8 y 16 respectivamente, la sumatoria de los valores en decimal es la dirección GPIB del dispositivo. La Tabla 4.2 presenta las direcciones válidas para emisores y receptores.

Tabla 4.2. Direcciones GPIB válidas para emisores y receptores.

Direcciones GPIB	Interruptor No 5 4 3 2 1					Direcciones emisor	Direcciones receptor
	Peso 16	8	4	2	1		
00	0	0	0	0	0	@	SP
01	0	0	0	0	1	A	¡
02	0	0	0	1	0	B	“
03	0	0	0	1	1	C	#
04	0	0	1	0	0	D	$
05	0	0	1	0	1	E	%
06	0	0	1	1	0	F	&
07	0	0	1	1	1	G	‘
08	0	1	0	0	0	H	(
09	0	1	0	0	1	I	)
10	0	1	0	1	0	J	*
11	0	1	0	1	1	K	+
12	0	1	1	0	0	L	,
13	0	1	1	0	1	M	-
14	0	1	1	1	0	N	.
15	0	1	1	1	1	O	/
16	1	0	0	0	0	P	0
17	1	0	0	0	1	Q	1
18	1	0	0	1	0	R	2
19	1	0	0	1	1	S	3
20	1	0	1	0	0	T	4
21	1	0	1	0	1	U	5
22	1	0	1	1	0	V	6
23	1	0	1	1	1	W	7
24	1	1	0	0	0	X	8
25	1	1	0	0	1	Y	9
26	1	1	0	1	0	Z	:
27	1	1	0	1	1	[	;
28	1	1	1	0	0	\	<
29	1	1	1	0	1	]	=
30	1	1	1	1	0	^	>
31	1	1	1	1	1	-	?

El estándar IEEE 488 define los números del 0 al 30 como direcciones válidas, la dirección 31 se reserva para propósitos de control y la dirección 21 se asigna al controlador (emisor/receptor), por lo tanto ambas direcciones no deben asignarse a los instrumentos en el bus, una característica del direccionamiento GPIB es que cada dirección primaria es única.

Algunos instrumentos o dispositivos requieren de una segunda dirección, la cual es una forma de identificación de dispositivos que constan de más de dos submódulos independientes pero que sólo cuentan con una configuración de dirección primaria, ejemplo de ello son los instrumentos VXI, este tipo de dirección es configurada de fábrica y puede repetirse en dos o más instrumentos que forman parte del GPIB.

El direccionamiento en el bus se realiza a través de la dirección primaria de cada dispositivo y se especifica en dos códigos de direcciones, esta dirección se envía a la línea de datos, donde el sexto y séptimo bit del octeto de datos distingue si corresponde a la dirección de un dispositivo emisor o a la de un dispositivo receptor. En los instrumentos modernos no es necesario especificar el direccionamiento debido a que el controlador configura automáticamente el sexto y séptimo bit del octeto de datos.

4.2.4. Especificaciones de procedimiento

Las especificaciones de procedimiento definen, principalmente, los mecanismos de transferencia de información y de sondeo, además de considerar las siguientes características:

- La longitud de los datos y órdenes: 8 bits (octeto) en código ASCII o binario.
- Tipo de órdenes: 34 principales y 32 secundarias.
- Dirección en instrumento: interruptores (6) o memoria ROM.
- Direcciones del estándar: emisores: 31 principales y 32 secundarios, receptores: 31 principales y 32 secundarios.
- Funciones básicas de los instrumentos: 10 (no presentes en todos los tipos de instrumentos).
- Mecanismos de sondeo: serie (*serial polling*) y paralelo (*parallel polling*).
- Estructura de datos y sintaxis: código ASCII, hexadecimal y octal.

4.2.4.1. Función de transferencia

Esta función utiliza las señales descritas en el párrafo 4.2.3.1.2 y se lleva a cabo de la siguiente manera:

1. El dispositivo emisor lee las líneas del bus y verifica la presencia de datos.
2. Todos los receptores están listos para recibir datos, en cada línea NRFD se pone un nivel “0” lógico.
3. El emisor pone la línea DAV a un nivel “1” lógico para indicar que el dato es válido.
4. Los receptores aceptan el primer dato y ponen la línea NRFD a un nivel “1” lógico para indicar que no están listo para recibir el siguiente octeto.
5. Cuando el receptor más lento recibe y acepta un dato, la línea NDAC es puesta en un nivel “0” lógico indicado que todos los receptores en el bus han aceptado el dato.
6. La línea DAV se coloca en un nivel “0” lógico indicado que el dato no es válido.
7. Se coloca la línea NDAC en un nivel “1” lógico, preparándose para recibir el próximo octeto.
8. NRFD toma un nivel lógico “0” y se repite el ciclo.

La figura 4.7 ilustra la transición de las líneas de control utilizados por la función de transferencia.

4.2.4.2. Técnicas de direccionamiento e identificación

El controlador se encarga de realizar el direccionamiento de los instrumentos, para ordenar la acción de un dispositivo, el controlador debe primero colocar en el bus la dirección del dispositivo, cada instrumento compara la dirección con la suya y sólo en uno de ellos se produce la identificación.

Por otra parte, un instrumento puede solicitar los servicios del controlador mediante la activación de la señal SRQ, en respuesta, el controlador debe realizar un proceso de identificación o sondeo, para lo cual, el controlador cuenta con dos métodos de sondeo que le ayudan a determinar el estado de los instrumentos o dispositivos en el bus, estos son el sondeo serie y el sondeo paralelo.

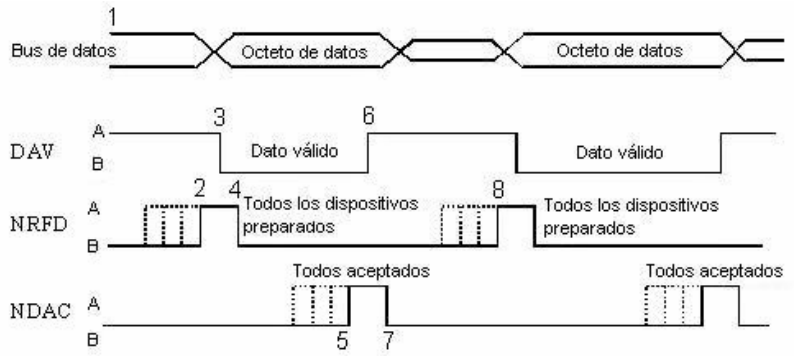


Figura 4.7. Transmisión de datos usando la función de transferencia (*handshake*).

4.2.4.2.1. Sondeo serie

El sondeo serie (*Serial Poll*) está formado por una secuencia de acciones en las que cada dispositivo que es direccionado regresa un octeto de estado indicando su condición actual, el controlador debe direccionar a cada dispositivo en el bus y asegurarse que la petición SRQ (*Service Request*) fue realizada. Cuando el controlador finaliza el sondeo, desactiva el sondeo serial y utiliza las órdenes para regresar al dispositivo a su estado normal.

La ventaja de realizar un sondeo serial es que cada dispositivo direccionado regresa su identificación y su octeto de estado, con lo cual el controlador tiene un completo conocimiento de las actividades en el bus, la desventaja de este tipo de sondeo radica en el consumo de tiempo que representa este procedimiento.

Originalmente el estándar IEEE 488.1 establece que el controlador puede leer el estado de los instrumentos presentes en el bus a través del bit RSQ (*Recall Status Query*) del octeto de estado, el estándar IEEE 488.2 usa dos bits más del octeto de estado para obtener un mayor control (ver figura 4.8):

- El bit 4 determina si el mensaje es válido (MAV, *Message Available*) e indica si la cola (*queue*) del instrumento está vacía con un nivel "0" lógico.
- El bit 5 determina el estado de los eventos (ESB, *Event Status Bit*) e indica si ha ocurrido algún evento estándar, como pueden ser la falta de energía, las requisiciones, los errores de órdenes o los errores de ejecución.

El sondeo paralelo, analizado en el apartado 4.3.6, es más rápido que el sondeo serie pero tiene la desventaja de que sólo ocho dispositivos pueden ser sondeados al mismo tiempo, si hay más de ocho dispositivos en el bus, el sondeo se debe realizar en dos o más pasos.

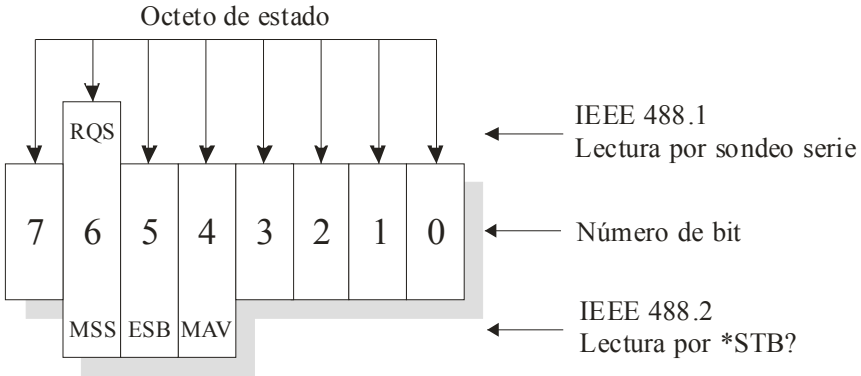


Figura 4.8. Octeto de estado del sondeo serie definido por el estándar IEEE 488.

4.2.4.3. Transferencia de información

El GPIB cuenta con un flujo bidireccional de datos y órdenes entre el controlador y los dispositivos interconectados, la comunicación entre los dispositivos se establece a través de las 16 líneas del bus. El estándar IEEE 488 define dos categorías básicas para la transmisión de mensajes:

- *Mensajes de interfaz*: llamados también órdenes, los cuales son utilizados por los dispositivos para adoptar un modo deseado o realizar alguna función requerida.
- *Mensajes de dispositivos dependientes*: contienen la información que se está transmitiendo en el bus y no son órdenes. Los mensajes de dispositivos dependientes son los datos transmitidos en las ocho líneas de entrada/salida en el bus cuando el estado de la línea ATN tiene un "1" lógico y el dispositivo emisor envía datos a todos los receptores activos.

La Tabla 4.3 presenta los valores, en código ASCII, asociados a cada octeto transmitido y recomendados en el estándar IEEE 488, sin embargo, los fabricantes de dispositivos GPIB son libres de implementar cualquier técnica de codificación para representar la información en las ocho líneas de datos.

Tabla 4.3. Código ASCII asociado a los valores del octeto de datos.

ASCII / ISO & Código de caracteres IEEE															
B7				**0**		**0**		**0**		**0**		**1**		**1**		**1**		**1**	
B6				**0**		**0**		**1**		**1**		**0**		**0**		**1**		**1**	
Bits	**B5**			**0**		**1**		**0**		**1**		**0**		**1**		**0**		**1**	
B4	**B3**	**B2**	**B1**	**Control**				**Símbolos numéricos**				**Opción Alta**				**Opción baja**			
0	0	0	0	NUL		DEL		SP		0		@		P		‘		p	
0	0	0	1	SOH		DC1		¡		1		A		Q		A		q	
0	0	1	0	STX		DC2		“		2		B		R		b		r	
0	0	1	1	ETX		DC3		#		3		C		S		C		s	
0	1	0	0	EOT		DC4		$		4		D		T		d		t	
0	1	0	1	ENQ		NAK		%		5		E		U		E		u	
0	1	1	0	ACK		SYN		&		6		F		V		f		v	
0	1	1	1	BEL		ETB		‘		7		G		W		g		w	
1	0	0	0	BS		CAN		(		8		H		X		h		x	
1	0	0	1	HT		EM		)		9		I		Y		i		y	
1	0	1	0	LF		SUB		*		:		J		Z		j		z	
1	0	1	1	VT		ESC		+		;		K		[		k		{	
1	1	0	0	FF		FS		,		<		L		\		l		:	
1	1	0	1	CR		GS		-		=		M		]		m		}	
1	1	1	0	SO		RS		.		>		N		^		n		~	
1	1	1	1	SI		US		/		?		O		-		o		Rubout	
				Orden Direcc.		**Orden Univer.**		**Direcciones receptor**				**Direcciones emisor**				**Direcciones secundarias u órdenes**			

4.2.4.4. Funciones IEEE 488

El estándar IEEE 488 específica un total de 11 funciones de interfaz, las cuales se implementan en todo dispositivo GPIB, a cada función se asocia un nemónico que describe una capacidad particular. La Tabla 4.4 presenta las funciones de interfaz manejadas en el bus GPIB.

Tabla 4.4. Funciones de interfaz del GPIB.

Función de interfaz	Nemónico	Descripción
Emisor, Emisor extendido (*Talker*, *Extended Talker*)	T(TE)	Dispositivo capaz de transmitir.
Receptor, Receptor extendido (*Listener*, *Extended Listener*)	L(LE)	Dispositivo capaz de recibir datos y órdenes.
Fuente de transferencia (*Source Handshake*)	SH	Dispositivo capaz de transferir un mensaje multilínea.
Controlador de transferencia (*Acceptor Handshake*)	AH	Dispositivo capaz de recibir mensajes multilínea.
Remoto o local (*Remote*/*Local*)	RL	Dispositivo capaz de operar desde el panel frontal y obtener información en forma remota desde el controlador.
Petición de servicio (*Service Request*)	SR	El dispositivo puede realizar una petición de servicio desde el controlador.
Sondeo Paralelo (*Parallel Poll*)	PP	La petición del dispositivo controlador debe identificarse así misma sólo sí está activada la petición de servicio.
Inicialización de dispositivo (*Device Clear*)	DC	El dispositivo puede ser inicializado en un estado predeterminado.
Controlador (*Controller*)	C	El dispositivo puede enviar direcciones, órdenes universales, órdenes direccionadas y sondeos.
Manejadores (*Drivers*)	E	Este código describe el tipo de controladores de un dispositivo.

4.2.4.5. Órdenes universales

El GPIB cuenta con órdenes universales, Tabla 4.5, que se transmiten en las líneas de datos para que los dispositivos interpreten los datos u órdenes cuando la línea ATN tiene un nivel "0" lógico.

Tabla 4.5. Órdenes universales del estándar IEEE 488.

Órdenes universales	Descripción
Inicialización de dispositivo (DCL, *Device Clear*)	Asigna a un dispositivo un estado de inicialización.
Desactivación local (LLO, *Local Lockout*)	Inhibe el funcionamiento del control manual (local) de todos los dispositivos, además de predeterminar su modo de operación.
Sondeo serie activo (SPE*, Serial Poll Enable*)	Indica el comienzo de una operación en un modo de sondeo serie.
Sondeo serie no activo (SPD, *Serial Poll Disable*)	Permite que cada dispositivo en modo SPE regrese a su estado normal al finalizar la operación del sondeo serie.
Desconfigurar sondeo paralelo (PPU, *Parallel Poll Unconfigure*)	Indica el final de una operación de sondeo paralelo.
Desactivar emisores (UNT, *Untalk*)	Se encarga de desactivar a todos los dispositivos emisores.
Desactivar receptores (UNL, *Unlisten*)	Se encarga de desactivar a todos los dispositivos receptores.

Algunas órdenes en el GPIB son necesarias en operaciones de control remoto, ya que cuentan con las capacidades de recibir y enviar datos e información, de predeterminar estados y de activar modos de operación.

4.2.4.6. Órdenes direccionadas

Cuando los dispositivos en el bus se encuentran en modo remoto, el controlador utiliza el grupo de órdenes direccionadas para asignarles un modo de operación deseado. La Tabla 4.6 describe los principales órdenes que se utilizan para configurar los dispositivos direccionados.

Tabla 4.6. Órdenes direccionadas.

Órdenes direccionadas	Descripción
Grupo de activación de disparo (GET, *Group Execute Trigger*)	Permite a los dispositivos diseccionados, con capacidad de recibir y responder a las órdenes, inicializar una acción preprogramada.
Inicialización del dispositivo seleccionado (SDC, *Selected Device Clear*)	Orden de reinicialización de dispositivos receptores a un estado predeterminado especificado por el fabricante.
Ir a modo local (GTL, *Go to Local*)	Orden de dispositivos receptores que permite el retorno de un estado remoto a un control manual (local) a través del panel frontal del dispositivo.
Configuración del sondeo paralelo (PPC, *Parallel Poll Configure*)	Orden usada con la orden secundaria sondeo paralelo activo (PPE, *Parallel Poll Enable*), para permitir a los receptores ser configurados. Otra orden secundaria es la de sondeo paralelo desactivo (PPD, *Parallel Poll Disable*), la cual previene y responde desde el dispositivo direccionado.

4.2.4.7. Terminación de mensajes

En el estándar IEEE 488 no está estipulado el tamaño de los mensajes que se transmiten por el bus, por lo tanto se definen los siguientes métodos para informar a los dispositivos conectados al bus que el mensaje ha finalizado:

- El dispositivo que transmite datos en el bus agrega el carácter ASCII “10” al final de cada mensaje.
- Mediante el uso de la línea EOI (*End or Identify*), la cual tiene las funciones de finalización de mensaje o de identificación del dispositivo dependiendo del valor que tome la línea ATN, cuando ATN es falsa, nivel “1” lógico, la línea EOI indica que se transmite el último octeto del mensaje, mientras que cuando es verdadera, nivel “0” lógico, identifica al dispositivo correspondiente.
- Se utiliza la línea EOI concurrentemente a la línea de alimentación de caracteres para indicar el final de un mensaje.

Los receptores en el bus pueden responder a cualquiera de los tipos de mensajes de terminación.

Algunos fabricantes de instrumentos GPIB han agregado la capacidad de determinar el tipo de mensaje de terminación para que se logre un mejor servicio del sistema, por ejemplo los mostrados en la Tabla 4.7. Los terminadores CR y LF se usan para seleccionar el método de indicación de terminación de un mensaje, en forma similar a EOI, un controlador puede finalizar la transmisión cuando recibe un terminador CR y como respuesta envía LF en el emisor.

Tabla 4.7. Órdenes de terminación en un dispositivo GPIB.

Órdenes	Terminadores
W0 (predefinido)	Habilita CR, LF, EOI
W1	Habilita CR & LF
W2	Habilita CR & EOI
W3	Habilita CR
W4	Habilita LF & EOI
W5	Habilita LF
W6	Habilita EOI
W7	Desactiva todas la terminaciones de salida

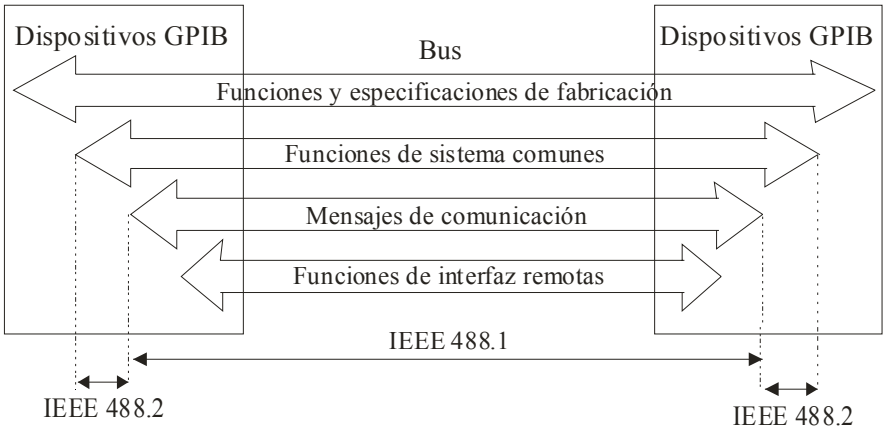


Figura 4.9. Estructura funcional del IEEE 488.

4.3. Estándar IEEE 488.2

El estándar IEEE 488.2 surgió con la intención de eliminar los problemas que presentaba el estándar original:

- La interconexión de dispositivos de distintos fabricantes.
- Intercambio de formatos de datos totalmente diferentes.
- Mensajes de protocolo no estandarizados.
- Gran diversidad de reportes de estado en los instrumentos.
- Duplicidad de funciones con nombres similares.

Dichos problemas fueron resueltos con la adopción del estándar IEEE 488.2 "Códigos, formatos, protocolos y órdenes comunes con el uso del estándar ANSI/IEEE-488.1-1987" [3], el cual establece un sólo formato para los mensajes, un conjunto de órdenes comunes, una estructura de reporte de estado y un protocolo para el controlador con la finalidad de unificar el control de instrumentos de diversos fabricantes, con estos cambios se mejora el desempeño del estándar original, renombrado como IEEE-488.1. La figura 4.9 ilustra la estructura del nuevo estándar para el GPIB.

Con la adopción del nuevo estándar se cumple con las características mencionadas en el subcapítulo 4.1.

4.3.1. Capacidades de la interfaz GPIB

El estándar IEEE 488.2 define un mínimo de capacidades que cada instrumento o dispositivo debe implementar, la Tabla 4.8 muestra las capacidades requeridas.

4.3.2. Sintaxis y formatos de datos

El estándar IEEE 488.2 especifica los formatos de datos para cualquier tipo de mensajes que pueden ser intercambiados en el bus, ya sean números o cadenas de caracteres, los tipos de números pueden ser binarios, octales y hexadecimales. La Tabla 4.9 lista los formatos para la transmisión de mensajes en el bus GPIB.

Los datos transmitidos por cualquier instrumento GPIB deben contar con una sintaxis y con el formato establecido en el estándar IEEE 728 para asegurar que los nuevos dispositivos puedan comunicarse con los dispositivos diseñados con anterioridad.

Tabla 4.8. Capacidades mínimas del estándar IEEE 488.2.

Capacidad	Código	Comentario
Fuente de transferencia (*Source Handshake*)	SH1	Capacidad total.
Controlador de transferencia (*Acceptor Handshake*)	AH1	Capacidad total.
Emisor (*Talker*)	T (TE)5 o T(TE)6	Emisor básico, sondeo serie, desdireccionamiento de emisores básicos MLA.
Receptores (*Listener*)	L (LE)3 o L(LE)4	Receptor básico, desdireccionamiento MTA.
Petición de servicios (*Service Request*)	SR1	Capacidad total.
Inicialización de dispositivo (*Device Clear*)	DC1	Capacidad total.
Remoto o local (*Remote/ Local*)	RL0 o RL1	Ninguna o capacidad total.
Sondeo paralelo (*Parallel Poll*)	PP0 o RL1	Ninguna o capacidad total.
Disparo de dispositivo (*Device Trigger*)	DT0 o DT1	Ninguna o capacidad total.
Controlador (*Controller*)	C0 o C4 con C5, C7, C8, C11	Ninguna o respuesta a SRQ, envía si el mensaje es válido y recibe el control.
Interfaz eléctrica (Electrical Interface)	E1o E2	Colector abierto o transistor.

Tabla 4.9. Formato de datos especificado por el estándar IEEE 488.2.

Formato de datos	
Formatos de receptores	**Estado**
<Decimal Numérico Dato Programado>	Requerido
<Carácter Dato Programado>	Opcional
<Sufijo Dato Programado>	Opcional
<No-Decimal Numérico Dato Programado>	Opcional
<Cadena Dato Programado>	Opcional
<Arbitrario Bloque Dato Programado>	Opcional
<Expresión Dato Programado>	Opcional
Formatos de emisores	**Estado**
<NR1 Numérico Respuesta de datos>	Requerido
<Arbitrario ASCII Respuesta de datos>	Requerido
<Carácter Respuesta de datos>	Opcional
<NR2 Numérico Respuesta de datos>	Opcional
<NR3 Numérico Repuesta de datos>	Opcional
<Hexadecimal Numérico Respuesta de datos>	Opcional
<Octal Numérico Respuesta de datos>	Opcional
<Binario Numérico Respuesta de datos>	Opcional
<Cadena Respuesta de datos>	Opcional
<Longitud Definida Bloque Arbitrario Respuesta de datos>	Opcional
<Longitud Indefinida Bloque Arbitrario Respuesta de datos>	Opcional

4.3.3. Protocolos de mensajes

En el estándar original, cualquier dispositivo en el bus que recibe un mensaje debe interpretarlo y reaccionar de acuerdo a la solución de cada fabricante, lo cual dificultó a la ingeniería de software la estandarización de un protocolo de mensajes.

El documento del estándar IEEE 488.2 describe, cuidadosamente, el protocolo de intercambio de mensajes considerando las especificaciones del estándar IEEE 488.1, el estándar IEEE 488.2 define un conjunto de estados operacionales, los cuales se listan en la Tabla 4.10.

Tabla 4. 10. Conjunto de estados operacionales de los dispositivos.

Estado	Propósito
IDLE	Pausa para mensajes.
READ	Lectura y ejecución de mensajes.
QUERY	Almacena respuestas para ser enviadas.
SEND	Enviar respuestas.
RESPONSE	Respuesta de envío completada.
DONE	Respuesta de envío finalizada.
READLOCK	El dispositivo no puede almacenar más datos en el buffer.
UNTERMINATED	El dispositivo intenta leer un mensaje indeterminado.
INTERRUPTED	El dispositivo es interrumpido por un nuevo mensaje mientras envía una respuesta.

4.3.4. Conjunto de órdenes

Los instrumentos y dispositivos que conforman un sistema ATE deben utilizar órdenes similares con funciones idénticas. La documentación del estándar IEEE 488.2 especifica un conjunto de órdenes comunes para ser utilizadas por todos los dispositivos cuando son parte de un sistema más complejo, lo cual elimina los problemas que presentaba el estándar IEEE 488.1 en el que los dispositivos de diferentes fabricantes implementaban un conjunto de órdenes diferentes para sus funciones y reportes de estado. A continuación se describe cada una de las órdenes descritas en el estándar IEEE 488.2:

- *Sistema de datos*: se usan para almacenar y recuperar la información correspondiente a la identificación del dispositivo, la descripción, las opciones, los datos y los recursos de transferencia.
- *Operaciones internas*: incluyen operaciones para reinicializar, calibrar y autodiagnosticar los dispositivos GPIB.
- *Estado y eventos*: a través de estas órdenes se controla el estado y los eventos registrados en los dispositivos GPIB.
- *Sincronización*: para sincronizar las operaciones de todos los dispositivos dentro del sistema.
- *Sondeo paralelo*: dedicadas a controlar la respuesta del sondeo paralelo.
- *Disparo (Device Trigger)*: activan, desactivan y especifican la forma de respuesta al mensaje de disparo.
- *Controlador*: el control del bus puede ser transferido entre dispositivos usando la orden de paso control al dispositivo anterior.
- *Auto-configuración*: el estándar IEEE 488.2 define un algoritmo que permite asignar automáticamente la dirección de un dispositivo emisor o receptor.
- *Macros*: son órdenes opcionales que ayudan a definir nuevas órdenes para el control de los instrumentos.
- *Configuración de almacenamiento*: son usadas para almacenar el estado de un dispositivo y poder recuperarlo posteriormente.

El uso de órdenes comunes simplifica la programación de los instrumentos, proporcionando al programador un conjunto mínimo de órdenes para programar a los instrumentos GPIB en forma individual o en sistemas ATE. La Tabla 4.11 muestra las principales funciones de las órdenes comunes especificadas en el estándar IEEE 488.2.

Tabla 4.11. Órdenes organizadas por grupos.

Nemonico	Descripción	Tipo
	Órdenes de Autoconfiguración	
*AAD	Asignación de direcciones.	Opcional
*DLF	Desactivar función receptores.	Opcional
	Sistema de órdenes de datos	
*IDN?	Consulta de identificación.	Requerida
*OPT?	Opción de consulta de identificación.	Opcional
*PUD	Dato de usuario protegido.	Opcional
*PUD?	Consulta de dato protegido de usuario.	Opcional
*RDT	Descripción de recursos de transferencia.	Opcional
*RDT?	Consulta de descripción de recursos de transferencia.	Opcional
	Órdenes de operación interna	
*CAL?	Consulta de calibración.	Opcional
*LRN	Consulta la configuración del dispositivo.	Opcional
*RST	Reinicio.	Requerida
*TST	Autodiagnóstico.	Requerida
	Órdenes de sincronización	
*OPC	Operación completa.	Requerida
*OPC?	Consulta de operación completa.	Requerida
*WAI	Pausa para completar operación.	Requerida
	Órdenes de macros	
*DMC	Define macro.	Opcional
*EMC	Activa macro.	Opcional
*EMC?	Activa consulta de macro.	Opcional
*GMC?	Obtiene el contenido de la consulta de macro.	Opcional
*LMC?	Aprende la consulta de macro.	Opcional
*PMC	Depura macros.	Opcional
	Órdenes de llamado paralelo	
*IST?	Consulta de estado individual.	Reqd. si PP1
*PRE	Registro de llamado paralelo activado.	Reqd. si PP1
*PRE?	Consulta de registro de llamado paralelo activado.	Reqd. si PP1
	Órdenes de estado y eventos	
*CLS	Borra el estado actual del dispositivo.	Requerida
*ESE	Activa los eventos de estado.	Requerida
*ESE?	Activa la consulta de eventos de estado.	Requerida
*ESR?	Consulta del registro de eventos de estado.	Requerida
*PSC	Inicio de limpieza de estado.	Opcional
*PSC?	Consulta de inicio de limpieza de estado.	Opcional
*SRE	Activa la petición de servicios.	Requerida
*SER?	Consulta de petición de servicio.	Requerida
*STB?	Consulta del octeto de lectura de estado.	Requerida
	Órdenes de disparo de dispositivos	
*DDT	Define disparo de dispositivo.	Opc. si DT1
*DDT?	Define la consulta de disparo del dispositivo.	Opc. si DT1
*TRG	Disparo.	Reqd. si DT1
	Órdenes de controlador	
*PCB	Paso de bloque de control.	Reqd. si controlador
	Órdenes de configuración de almacenamiento	
*RCL	Llamada de estado de instrumento.	Opcional
*SAV	Almacenar estado de instrumento.	Opcional

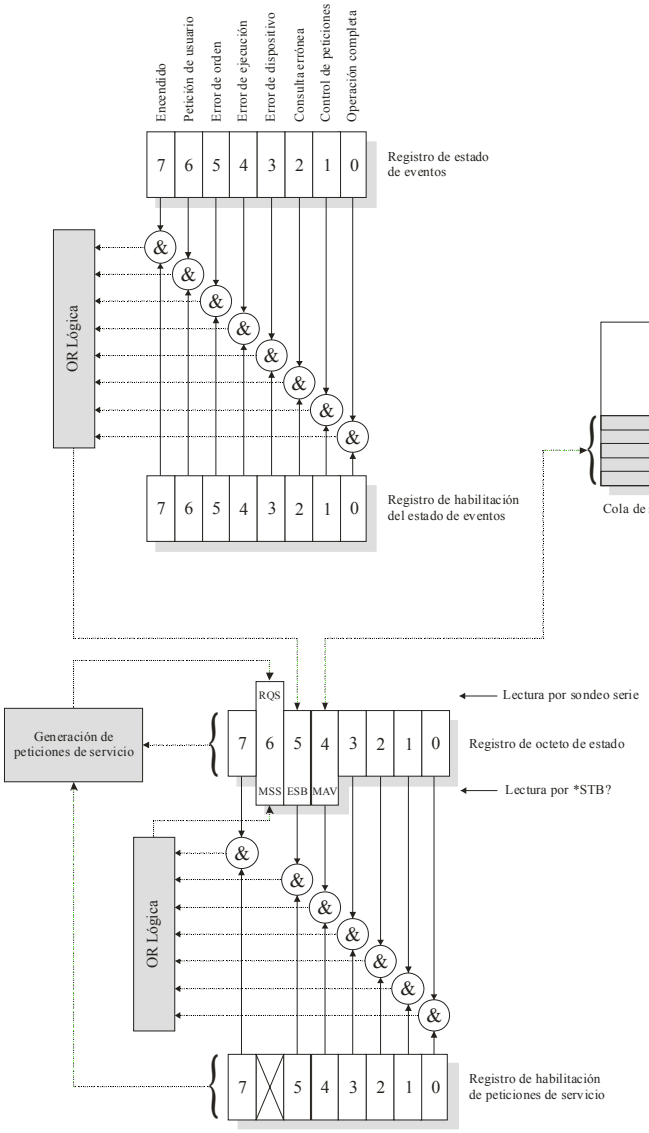


Figura 4.10. Estructura del registro de estado del estándar IEEE 488.2.

Probablemente la orden más utilizada es la de consulta *IDN?, la cual comprueba la comunicación con el instrumento direccionado y su respuesta incluye los datos de identificación del mismo. La mayoría de las órdenes forman parte de la estructura del reporte de estado.

Todas las órdenes son enviadas en modo de datos en el bus (cuando ATN tiene un nivel "1" lógico). El estándar IEEE 488.2 define las órdenes comunes a implementar en las operaciones de dispositivos GPIB, mientras que otras se implementan a criterio del programador.

4.3.5. Informe de estado

El estándar IEEE 488.2 cuenta con una estructura de reporte de estado que amplía las especificaciones del octeto de estado definido por el estándar IEEE-488.1, el nuevo estándar define siete bits, verificados por la estructura de datos (*Status Data Structure*), para determinar si existen mensajes. El estándar IEEE 488.1 define un llamado para consultas de estado (RSQ, *Recall Status Query*) y el estándar IEEE 488.2 define los eventos de estado en el bit ESB y los mensajes válidos en el bit MAV, el usuario puede habilitar un servicio de petición a un dispositivo mediante la suma de los bits en el octeto de estado.

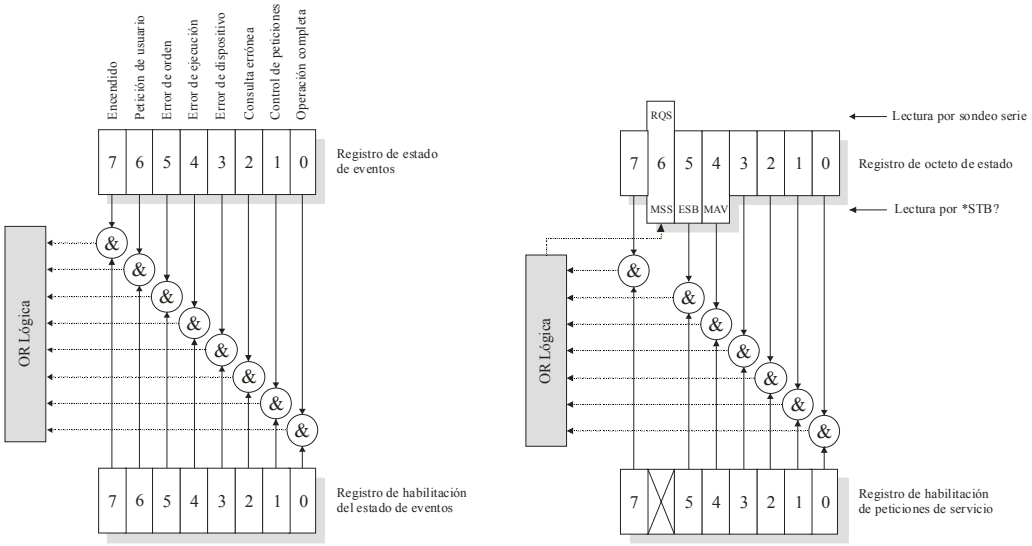


Figura 4.11. a) Estado de eventos de registros (SESR), b) Servicio de petición.

El octeto de estado se transfiere al controlador mediante el sondeo serie del estándar IEEE 488.1 o realizando las consultas comunes del estándar IEEE 488.2, además, se cuenta con órdenes y consultas adicionales para obtener información desde los dispositivos a control remoto. Una visión global de la estructura del reporte global de estado del estándar IEEE 488.2 se muestra en la figura 4.10.

Los bits definidos por el estándar IEEE 488.2 son los bits 4, 5 y 6, el bit 4 indica cuando un mensaje es válido (MAV), un "0" lógico indica que la salida de la cola contiene un dato válido, el bit 5 (ESB) indica si ha ocurrido un evento.

La suma de los bits de estado (MSS, *Master Summary Status*) indica si en el dispositivo ha ocurrido un servicio de petición, el bit MSS no es considerado como parte del octeto de estado ya que sólo envía su respuesta al sondeo serie. Debido a la doble función del bit 6, éste responde a los servicios de petición y al sondeo serie implementado en el estándar IEEE-488.1.

La figura 4.11b ilustra la habilitación del servicio de peticiones (SRER, *Service Request Enabling Operation*), donde el usuario puede activar los bits del SRER correspondientes a los bits del octeto de estado.

El estándar IEEE 488.2 define una orden de lectura para los eventos estándares en el registro de estado (SESE, *Standard Event Status Register*), si un dispositivo tiene más de un evento registrado debe haber otro dispositivo dependiente de órdenes para acceder al dato almacenado.

Un dispositivo GPIB cuenta con una activación de eventos registrados (EER, *Event Enable Register*) similares a la orden SRER, la configuración de los bits de la orden EER permite sumar los bits de este registro dentro del octeto de estado.

La figura 4.11a muestra los ocho bits de la orden SESR que pueden ser monitoreados y reportados a los usuarios, estos eventos son:

- *Operación completa* (OPC, *Operation Complete*): este bit genera una respuesta a la orden OPC, se activa cuando un dispositivo completa sus operaciones y está listo para aceptar una nueva orden.
- *Petición de control* (RQC, *Request Control*): este bit es usado por un dispositivo para indicar al controlador que desea inicializarse como nuevo controlador.
- *Consulta errónea* (QYE, *Query Error*): una consulta errónea ocurre cuando se captura el buffer y éste no presenta el dato debido a pérdidas por sobreflujo.

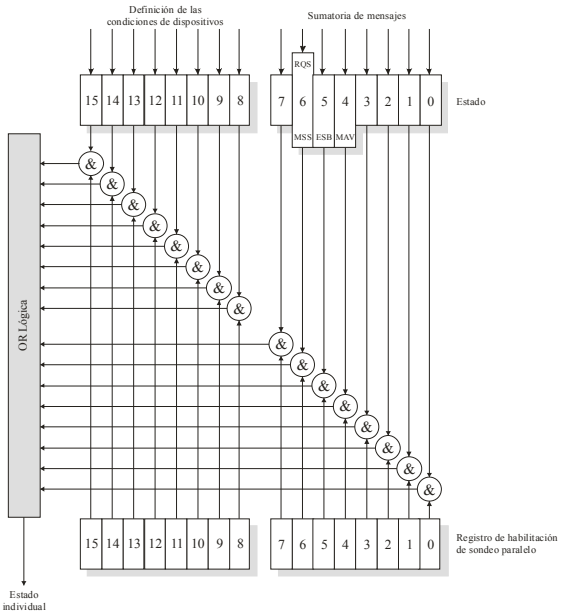


Figura 4.12. Sondeo paralelo del estándar IEEE 488.2.

- *Error de dispositivo* (DDE, *Device-Dependent Error*): este bit se activa cuando se presenta un error en una función debido a la ejecución repetida de una orden.
- *Error de ejecución* (EXE, *Execution Error*): ocurre cuando la orden recibida por el dispositivo no cuenta con las capacidades o es inconsistente con el diseño de la operación.
- *Error de orden* (CME, *Command Error*): este bit indica qué dispositivo recibió una orden con un error de sintaxis, un error sistemático o no implementado en el dispositivo.
- *Petición de usuario* (URQ, *User Request*): el propósito de este bit es el de captar la atención del controlador resguardando el estado del dispositivo, ya sea remoto o local.
- *Encendido* (PON, *Power On*): este bit indica que el dispositivo ha sido apagado.

Todos los dispositivo cuentan con el registro SESR y otros eventos, este registro es escrito con la orden *ESE (*Enable Status*) y captado con la orden *ESE? (*Enable Status Query*).

Las colas de registros son usadas para permitir a un dispositivo reportar el estado del controlador u otra información, cada cola de registros tiene la sumatoria de los bits de los mensajes que contienen información. La cola de salida del dispositivo usa al bit MAV del octeto de estado para indicar que contiene un dato válido.

4.3.6. Sondeo paralelo

El sondeo paralelo (*Paralell poll*), definido por el estándar IEEE 488.1, es un procedimiento que provee de una rápida identificación del estado de cada uno de los dispositivos en el bus al contar con la capacidad de generar y controlar las respuestas del dispositivo con un mecanismo paralelo.

Cuando el controlador inicia el sondeo paralelo, cada uno de los ocho dispositivos regresan su bit de estado en una de las líneas de datos de entrada/salida. Cada dispositivo puede ser direccionado y responder en cada una de las líneas de datos usando la orden secundaria PPE (*Parallel Poll Enable*), la cual es una orden de configuración del sondeo paralelo.

Para que varios dispositivos usen una línea de datos es necesario el contar con dispositivos que tengan circuitos de salida de colector abierto, que permitan una conexión en paralelo, usando ANDs y ORs en los bits de estado. Además de ello el estándar IEEE 488 garantiza la terminación de los mensajes mediante el carácter EOI (*End or Identify line*) en el bus.

La figura 4.12 ilustra la estructura del sondeo paralelo IEEE 488.2, parecida a un registro de evento (*Event register*), esta sumatoria de bits se envía en respuesta a un sondeo paralelo y al octeto de estado, también es referenciada a una consulta individual de estado (IST, *Individual Status Query*) o a un estado local individual.

La orden *IST? permite al usuario determinar el estado del IST al captar la respuesta del dispositivo sin necesidad de un sondeo paralelo.

La orden *PPE determina las condiciones que se consideran en la sumatoria del IST, estas órdenes deben proporcionar números, convertidos a binario, representados en los bits PPE.

En la Tabla 4.12 se presenta un resumen de las especificaciones básicas del estándar IEEE 488.

Tabla 4.12. Especificaciones básicas del estándar IEEE 488.

Versión del estándar	Tipo de especificaciones	Concepto	Descripción
IEEE 488.1	Mecánicas	Configuración del bus	Estrella (*Star*) o Serie (*Líneal*).
		Longitud del bus	2 m entre instrumentos (*Máxima longitud total 20 m).*
		Conector	Tipo americano de 24 pines.
			Instrumento (*hembra*).
			Cable (*apilable: macho y hembra*).
	Eléctricas	"1" lógico en la salida	0 ÷ 0,5 V.
		"1" lógico en la entrada	-0,6 ÷ 0,8 V.
		"0" lógico en la salida	2,4 ÷ 5 V.
		"0" lógico en la entrada	2 ÷ 5,5 V.
	Funcionales y de procedimiento	Tipo de instrumentos	Controlador (*Controller*), Emisor (*Talker*), Receptor (*Listener*).
		N. de instrumentos conectables	Máximo 15.
		N. de instrumentos activos	Más de la mitad de los instrumentos conectados deben estar activos.
		Velocidad	Máxima 1 Mbps (*distancias largas 200-300 Kbps*).
		Longitud de datos y órdenes	8 bits (*Octeto*) en código ASCII o binario.
		Tipos de órdenes	34 principales y 32 secundarias.
		Dirección de instrumento	Conmutadores (6) o memoria pasiva.
		Direcciones del estándar	Emisores (*Talkers*): 31 principales y 32 secundarios.
			Receptores (*Listeners*): 31 principales y 32 secundarios.
		Funciones básicas de los instrumentos	10 (no presentes en todos los tipos de instrumentos).
		Mecanismos de sondeo	Serie (*Serial polling*) y Paralelo (*Parallel polling*).
IEEE 488.2	De procedimiento	Estructura de datos y sintaxis	Código ASCII, Hexadecimal y Octal.

5. Modelado del SepiGPIB

El modelado del SepiGPIB aplica el paradigma orientado a objetos (capítulo 1) al diseño e implementación de un software para simular el funcionamiento del protocolo de instrumentación GPIB.

La metodología utilizada durante el desarrollo de este sistema fue RUP (subcapítulo 1.4) y algunos aspectos de la orientación a objetos, las herramientas que contribuyeron al desarrollo de este sistema fueron Rational Rose y UML como lenguajes de modelado y Borland C++ Builder como lenguaje de implementación del modelo obtenido. El sistema SepiGPIB utiliza los formatos del RUP para documentar, organizar y explicar los procesos desarrollados.

5.1. Metodología RUP implementada

El SepiGPIB consideró las etapas más relevantes de la metodología RUP y se desarrolló en forma interactiva mediante un desarrollo en cascada, es decir, se realizaron las etapas de requerimientos, análisis y diseño de manera secuencial hasta llegar a una versión ejecutable del sistema. La figura 5.1 ilustra las etapas del desarrollo interactivo.

Cabe mencionar que el desarrollo interactivo tiene que ajustarse al software bajo desarrollo, en donde pueden presentarse algunas variaciones, sin olvidar las etapas base del desarrollo orientado a objetos.

Cada fase de desarrollo genera una documentación[10] cuya finalidad es explicar cómo es que dicha fase se lleva a cabo en el modelado del sistema. A continuación se describe cada una de las fases mencionadas.

5.2. Planeación del SepiGPIB

El SepiGPIB tiene como finalidad el simular el comportamiento del GPIB, para ello es necesario configurar la simulación y los instrumentos conectados al bus para posteriormente mostrar su funcionamiento con ayuda de la herramienta de monitoreo.

El sistema permitirá a los usuarios realizar experimentos en el campo de los sistemas de instrumentación electrónica sin tener que estar en un laboratorio real, además les permitirá comprender el funcionamiento del GPIB mediante el monitoreo de sus señales.

La visión que se tiene del software es la de desarrollar el SepiGPIB para servir de ayuda a futuras implementaciones y como referencia a quien pretenda desarrollar estudios basados en instrumentación electrónica. El SepiGPIB es un sistema enfocado, principalmente, a los estudiantes

[10] La documentación generada por cada fase del desarrollo se incluye en los documentos auxiliares de esta tesis.

con conocimientos en electrónica que se inician en el complejo mundo de la instrumentación y a todos aquellos que deseen conocer el funcionamiento del protocolo de comunicaciones GPIB.

El SepiGPIB es un producto innovador en el mercado debido a que los sistemas actuales no se enfocan a capacitar o introducir al estudiante en el campo de la instrumentación electrónica y por lo tanto no existe ningún competidor en el mercado, además, los estudiantes de ingeniería en electrónica e ingeniería en computación no cuentan con sistemas de software desarrollados para la enseñanza.

A continuación se describen algunos de los aspectos identificados al planear el sistema:

- *El problema*: no se cuenta con sistemas especializados destinados al estudio de los buses de comunicación utilizados en instrumentación electrónica, los existentes presentan un alto grado de complejidad y altos costos.
- *Afecta a*: los estudiantes de ingeniería electrónica y aquellas personas que desean realizar estudios en instrumentación electrónica.
- *Una solución exitosa ofrece*: a los estudiantes, un método práctico para comprender el funcionamiento del GPIB mediante la simulación y monitoreo de las señales en el bus, además de reducir costos debidos a posibles daños en los instrumentos físicos por un uso incorrecto.
- *Para*: estudiantes de ingeniería en electrónica e ingeniería en computación y personas con conocimientos básicos de instrumentación electrónica.
- *Quien*: requiera contar con sistemas documentados de instrumentación programable con un monitor de las acciones realizadas en el bus y le permita experimentar con instrumentos conectados a un bus estandarizado.
- *Ayuda*: en la comprensión del funcionamiento de sistemas ATE.
- *Diferente a*: los productos comerciales debido a que el SepiGPIB es un producto con fines académicos, orientado a la enseñanza de los protocolos de comunicaciones en áreas de ingeniería.

Para realizar una simulación de un sistema GPIB se requiere de un usuario y del software SepiGPIB ejecutándose bajo un sistema operativo Windows 9x, XP o 2000, en donde el cliente y el usuario son la misma persona.

El SepiGPIB esta diseñado para ser:

- *Entendible*: el SepiGPIB es una herramienta de simulación intuitiva al usuario.
- *Sencillo*: la información se presenta de forma sencilla mediante el diseño de una GUI.
- *Rápido*: las operaciones se llevan a cabo mediante pocas ventanas.
- *Interfaz gráfica*: con ayuda de una GUI se presentan las opciones para hacer más amigable el entorno del sistema.

Al momento de adquirir SepiGPIB, el usuario es propietario de una licencia y puede instalarlo únicamente para uso académico.

Algunos de los aspectos más importantes del sistema son:

- Presenta una GUI amigable.
- El sistema contiene, en su primera versión, 3 instrumentos con funciones básicas (subcapítulo 2.6).
- El software cuenta con su propio sistema de ayuda.

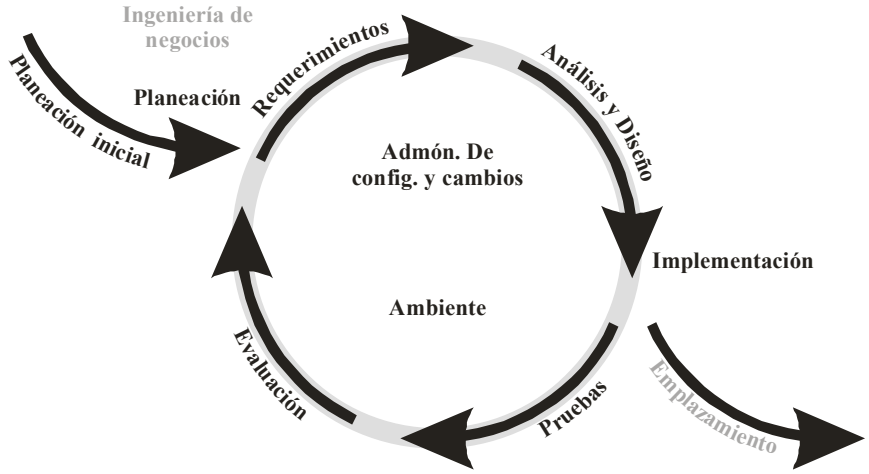


Figura 5.1. Etapas del desarrollo interactivo del RUP [26].

- Sólo se puede simular un sistema mínimo, que consta de un instrumento con interfaz GPIB.
- Se adaptan las características más importantes del GPIB para lograr una simulación con la mayor calidad posible.
- La simulación se realiza de acuerdo a las especificaciones del estándar IEEE 488.
- Se cuenta con los términos, funciones y ejemplos de ayuda para comprender el funcionamiento de un sistema básico GPIB.

Algunas especificaciones adicionales que el sistema SepiGPIB debe cumplir son:

- *Funcional*: el SepiGPIB es un programa funcional y sirve como herramienta de aprendizaje o de apoyo didáctico en materias afines a la electrónica, permitiendo con ello realizar prácticas que ilustran el funcionamiento del GPIB.
- *Usabilidad*: el programa ofrece un entorno amigable para que los usuarios tengan una fácil comprensión del sistema.
- *Confiabilidad*: los resultados del sistema se basan en prácticas realizadas en el Laboratorio de Comunicaciones Digitales de la Universidad Tecnológica de la Mixteca.
- *Rendimiento*: la eficiencia de los resultados se apega al comportamiento del GPIB y se basan en su estudio previo (capítulo 4).

5.2.1. Herramientas de modelado y desarrollo

Las herramientas utilizadas para implementar el sistema fueron un compilador visual y herramientas de modelado, las cuales se describen a continuación.

5.2.1.1. C++Builder

C++Builder es una herramienta visual desarrollada por la firma Borland y en este proyecto se utiliza como compilador, aprovechando su interfaz gráfica que permite aplicar un análisis y diseño orientado a objetos [10, 38], la versión utilizada fue la 6.0.

El compilador visual facilita el desarrollo de una GUI con las principales características de los programas comerciales disponibles en el mercado y proporciona un ambiente amigable a los usuarios del sistema final.

La elección de un compilador basado en C++ se debe, principalmente, a que en el área de la instrumentación programable la mayoría de los sistemas son programados en el lenguaje de programación C.

C++Builder fue instalado en una computadora personal bajo un sistema operativo Windows 9x, 2000 y XP.

5.2.1.2. Rational Rose 2000

Rational Rose es una herramienta tipo CASE utilizada por los desarrolladores de diagramas en UML y desarrollada por la firma Rational Software Inc. [URL2], en este sistema se utilizo de apoyo al modelado de un sistema orientado a objetos. También se contó con herramientas de modelado para UML.

Se utilizo Rational Rose para proveer de una herramienta de soporte a todos los modelos requeridos por el sistema, incluyendo el modelo conceptual, los diagramas de secuencia, los diagramas de colaboración y el modelo de clases.

5.3. Requerimientos del SepiGPIB

Esta fase se basa en las especificaciones iniciales o funcionales del sistema de forma sistemática y no ambigua, cada uno de los requerimientos proporcionó una mejor comprensión del sistema bajo estudio y consistió en dos actividades: fase de requerimientos y modelado de casos de uso.

5.3.1. Fase de requerimientos

Los requerimientos del sistema se encargan de analizar las expectativas, las especificaciones, la visión y demás características que debe cubrir el sistema. Esta fase consistió en describir las necesidades, requerimientos, metas y clientes del sistema.

En el desarrollo del SepiGPIB el manejo de los requerimientos es dinámico, es decir, éstos cambian durante la vida del proyecto. Inicialmente se parte de la especificación del problema a resolver para, posteriormente continuar con las etapas restantes de su desarrollo. A continuación se describen los aspectos más relevantes que dan lugar al desarrollo del SepiGPIB:

- El sector industrial exige la formación de profesionales en electrónica y computación con un amplio conocimiento en el desarrollo de sistemas para controlar procesos, verificar productos, explorar servicios, analizar la calidad del producto, etc. La continua reducción de costos, el aumento de potencia en procesamiento y la miniaturización que ofrece la microelectrónica junto al aumento de las prestaciones de paquetes informáticos, han permitido la aparición de sistemas de medida automatizados ATE basados en instrumentación electrónica programable (capítulo 3).
- La ciencia y tecnología requieren de equipos tecnológicos de altas prestaciones para medir, generar o convertir variables físicas. La instrumentación electrónica es el área de la tecnología que estudia los equipos realizados mediante circuitos y sistemas electrónicos destinados a realizar dichas tareas, dicho estudio incluye la normalización de los buses de comunicaciones, cuya importancia radica en lograr que los equipos o sistemas de diferentes fabricantes puedan comunicarse entre sí.
- Uno de los buses de comunicación empleados en el área de la instrumentación electrónica es el GPIB, estandarizado bajo las normas IEEE 488 e IEC 625, en la actualidad existe una diversidad de protocolos de comunicaciones destinados a la misma tarea, por ejemplo VXI, PXI, etc. El bus GPIB es de vital importancia para comprender un entorno de instrumentación programable debido a su amplio y variado uso para la comunicación en sistemas ATE.

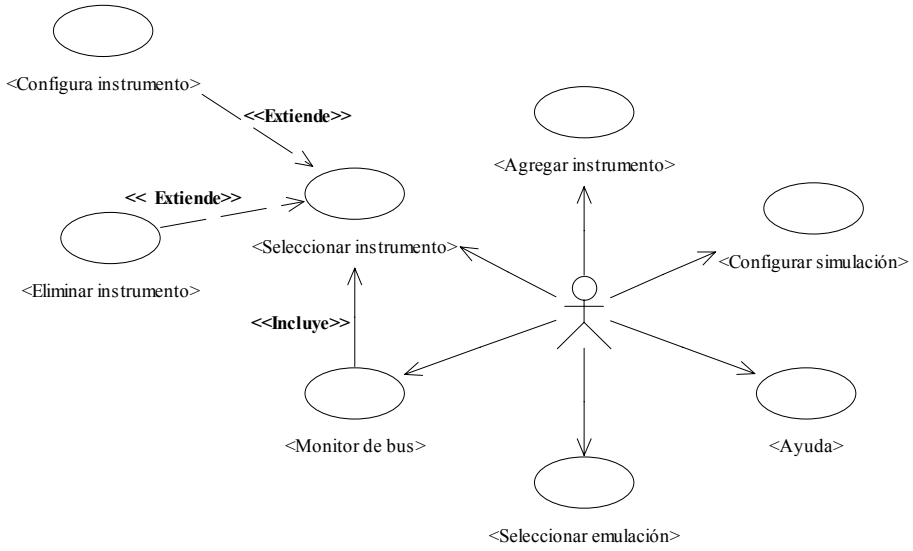


Figura 5.2. Diagrama de casos de uso para el SepiGPIB.

- El análisis del protocolo de comunicaciones GPIB tiene como objetivos: documentarlo en forma no ambigua, implementarlo en una herramienta informática de uso académico y ayudar a la comprensión de su completo funcionamiento.
- Por otro lado, las tecnologías informáticas de modelado orientado a objetos proporcionan potentes herramientas de diseño de software como son el lenguaje de modelado unificado (UML), la programación orientada a objetos (POO) y lenguajes de alto nivel para implementación. El desarrollo del software, denominado SepiGPIB, se basa en tales herramientas.
- El SepiGPIB tiene como objetivo el simular el comportamiento del GPIB mediante la configuración de los instrumentos en el bus y el análisis de su funcionamiento. El SepiGPIB realiza la función de controlador, es decir, se encarga de controlar las actividades en el bus.
- Las metas a lograr con el SepiGPIB son las siguientes:
  - Especificar el diseño del sistema bajo las normas del UML.
  - Simular el funcionamiento del GPIB mediante un lenguaje de alto nivel.
  - Proporcionar una GUI que permita una fácil interacción con el usuario.
  - Visualizar el contenido del protocolo GPIB a bajo nivel durante la simulación.

5.3.2. Modelado de casos de uso

Una vez obtenidos los requerimientos del sistema, se desarrolló el modelado de casos de uso, dicho modelado se utilizó para mejorar la comprensión de los requerimientos del sistema. La identificación de los casos de uso se obtuvo a partir del punto de vista del usuario con respecto al comportamiento esperado del sistema. Cada caso de uso se modela con respecto a una secuencia de operaciones entre el usuario y el sistema.

Durante el modelado de los casos de uso, se analizó cada uno de ellos y se fueron eliminando aquellos que eran redundantes, como resultado final se obtuvieron únicamente casos de uso expandidos, los cuales son parte de la documentación generada por el sistema. La figura 5.2 muestra la relación existente entre los casos de uso identificados en el SepiGPIB, los cuales se describen a continuación.

5.3.2.1. Seleccionar instrumento

El caso de uso Seleccionar instrumento comienza cuando el usuario selecciona un instrumento para realizar una operación dentro del programa, se considera de tipo secundario porque la selección de un instrumento no afecta la simulación.

La ventana de **Instrumentos activos** contiene, como mínimo, un instrumento o dispositivo activo en el bus y como resultado el sistema muestra el instrumento seleccionado, representado previamente por un icono. La tabla 5.1 muestra los flujos de eventos básicos y alternativos de este caso de uso.

Tabla 5.1. Flujos de eventos del caso de uso Seleccionar instrumento.

Flujo de eventos básico	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona un instrumento para realizar alguna operación.	2. El sistema presenta la opción **Configurar instrumento** del menú **Bus**, sólo se puede seleccionar un dispositivo conectando a la barra de instrumentos al mismo tiempo.
3. El usuario selecciona un instrumento proporcionando la dirección primaria correspondiente.	4. Muestra el instrumento seleccionado.
5. El usuario obtiene el instrumento seleccionado.	
Flujos alternativos	
1. El usuario selecciona la aplicación de control del GPIB.	2. Muestra la aplicación seleccionada.

5.3.2.2. Agregar instrumento

El caso de uso Agregar instrumento da inicio cuando el usuario selecciona un instrumento para agregarlo al bus y realizar operaciones dentro de la simulación, se considera de tipo primario y esencial debido a que sin instrumentos no se puede realizar la simulación.

El sistema muestra la **Barra de instrumentos**, en donde se presenta cada uno de los instrumentos activos en el bus. La tabla 5.2 muestra los flujos de eventos básicos y alternativos de este caso de uso.

Tabla 5.2. Flujos de eventos del caso de uso Agregar instrumento.

Flujo de eventos básico	
Acción del actor	**Respuesta del sistema**
1. El usuario desea agregar un instrumento al bus.	
2. El usuario selecciona un instrumento de la **Barra de instrumentos**.	3. Se agrega el instrumento a la ventana de **Instrumentos activos** y cada uno de los módulos del sistema reconoce el nuevo instrumento.
4. El usuario obtiene un instrumento activo en el bus.	
Flujos alternativos	
2. El usuario selecciona un instrumento, de diferente tipo, en la **Barra de instrumentos**.	3. Muestra el instrumento seleccionado como dispositivo activo en el bus.
2. El usuario no desea seleccionar ningún instrumento de los presentados.	4. El sistema se mantiene en espera de datos para realizar la simulación.

5.3.2.3. Eliminar instrumento

El caso de uso Eliminar instrumento comienza cuando el usuario selecciona un instrumento que se desea dar de baja en la simulación, se considera de tipo primario debido a que la eliminación de un instrumento afecta la simulación y una simulación no se puede llevar a cabo sin instrumentos.

La ventana de **Instrumentos activos** contiene al menos un instrumento en el bus y únicamente se pueden eliminar los instrumentos activos en el bus. La tabla 5.3 muestra los flujos de eventos básicos y alternativos de este caso de uso.

Tabla 5.3. Flujos de eventos del caso de uso Eliminar instrumento.

Flujo de eventos básico	
Acción del actor	**Respuesta del sistema**
1. El usuario desea eliminar un instrumento activo en el bus.	
2. El usuario abre la opción **Eliminar instrumento** en el menú **Bus**.	3. Muestra una ventana que solicita la selección del instrumento a eliminar.
4. Se proporciona la dirección primaria del instrumento a eliminar.	5. Muestra la descripción del instrumento a eliminar.
6. Se solicita la confirmación para eliminar el instrumento.	7. Elimina el instrumento y sus correspondientes registros.
Flujos alternativos	
2. Se realiza otra operación del menú **Bus**.	3. Espera nueva información.

5.3.2.4. Seleccionar emulación

El caso de uso Seleccionar emulación da inicio al seleccionar la opción **Emulación**, ubicada dentro del menú **Archivo** del sistema. Para realizar una emulación se debe contar con una interfaz GPIB (hardware) y una actualización del software SepiGPIB. Este caso de uso se considera de tipo primario debido a que la función del software cambia de simulación a emulación y afecta a todas las operaciones que realiza el sistema.

El sistema debe contar con un hardware especial de conexión GPIB y con el acoplo de las funciones elaboradas para el sistema. El **Monitor de señales** se muestra únicamente como resultado de la operación de emulación. La tabla 5.4 muestra los flujos de eventos básicos y alternativos de este caso de uso.

Tabla 5.4. Flujos de eventos del caso de uso Seleccionar emulación.

Flujo de eventos básico	
Acción del actor	**Respuesta del sistema**
1. El usuario desea realizar una emulación, previa actualización del hardware y software del sistema.	
2. El usuario selecciona del menú **Archivo** la opción **Emulación**.	3. Muestra la confirmación de la emulación.
4. Se realiza la confirmación.	5. Aparece la ventana de **Configuración de emulación**.
6. Se proporciona la configuración deseada.	7. Acepta la configuración.
8. Elige la opción **Ejecutar**.	9. Inicia el monitoreo de las señales del bus.
Flujos alternativos	
4. Rechaza la confirmación.	5. Cancela la operación de emulación.

5.3.2.5. Configurar simulación

El caso de uso Configurar simulación comienza cuando el usuario desea determinar los parámetros de la simulación a realizar y elige las opciones de configuración a su libre conveniencia, este caso de uso se considera de tipo secundario debido a que aún sin configurar el sistema, el modulo de confirmación de la simulación presenta valores predeterminados.

El sistema tiene una configuración predeterminada para realizar la simulación y el resultado de ello se muestra en la ventana **Recursos de la simulación**. La tabla 5.5 muestra los flujos de eventos básicos y alternativos de este caso de uso.

Tabla 5.5. Flujos de eventos del caso de uso Configurar simulación.

Flujo de eventos básico	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona la opción **Configurar simulación** desde el menú **Bus** o a través de su acceso directo.	2. Muestra la ventana **Configurar simulación**.
3. Proporciona los valores de la configuración.	4. Espera la opción aceptar.
5. Realiza la confirmación.	6. Muestra los valores determinados para la simulación a través de la ventana de **Recursos GPIB**.
7. El usuario obtiene la operación deseada.	
Flujos alternativos	
3. Elige las opciones predeterminadas.	4. Reestablece las opciones predeterminadas para la configuración de la simulación activa.
5. Rechaza la confirmación.	6. Cancela la operación de simulación.

5.3.2.6. Monitor de bus

El caso de uso Monitor de bus comienza cuando el usuario configura la simulación y realiza algunas operaciones con los dispositivos con la finalidad de analizar los resultados de la simulación, se considera de tipo primario y esencial debido a que la función principal del software es realizar una simulación del comportamiento del bus mediante el monitoreo de sus señales. La simulación debe configurarse de acuerdo a las funciones que maneje la aplicación y a las especificadas en el estándar GPIB. La tabla 5.6 muestra los flujos de eventos básicos y alternativos de este caso de uso.

5.3.2.7. Configurar instrumento

El caso de uso Configurar instrumento comienza cuando el usuario configura la aplicación al proporcionar valores y funciones a los instrumentos para realizar la simulación, este caso de uso se considera de tipo primario y esencial debido a que la función del sistema es proporcionar las operaciones a los instrumentos para realiza la simulación del GPIB.

Se debe contar con instrumentos activos en el bus y una vez realizada la configuración de los instrumentos, el sistema está listo para realizar la simulación. La tabla 5.7 muestra los flujos de eventos básicos y alternativos de este caso de uso.

5.3.2.8. Ayuda

El caso de uso Ayuda comienza cuando el usuario desea obtener ayuda sobre algún procedimiento a ejecutar durante el desarrollo de la simulación, se considera de tipo secundario porque la solicitud de ayuda no afecta el funcionamiento del sistema.

El sistema cuenta con un manual, que se puede ejecutar desde la ayuda del SepiGPIB en cualquier momento. La tabla 5.8 muestra los flujos de eventos básicos y alternativos de este caso de uso.

Tabla 5.6. Flujos de eventos del caso de uso Monitor de bus.

Flujo de eventos básico	
Acción del actor	**Respuesta del sistema**
1. El usuario desea observar los datos que se encuentran en el bus después de realizar alguna petición al sistema.	
2. El usuario proporciona la configuración deseada de la simulación.	3. Acepta la configuración.
4. Se agrega un instrumento al bus.	5. Muestra el instrumento.
6. Se accede a la opción **Comunicación directa** con el instrumento.	7. Muestra la configuración del controlador.
8. Se realizan operaciones de consulta y configuración al instrumento.	9. Registra los datos de las operaciones realizadas.
10. Se abre la opción **MonGPIB** ubicada dentro del menú **Monitor** o a través de su acceso directo en la barra de iconos.	11. Muestra el comportamiento de la simulación mediante una trama de datos que ilustra cómo se realizaron las peticiones al controlador.
Flujos alternativos	
6. Se cierra la ventana **Comunicación directa** mediante un doble clic al instrumento activo.	7. Desactiva el panel frontal del instrumento activo.
8. Se elige la opción **Comunicación directa IEEE 488.2**.	9. Muestra una lista de órdenes.
10. Se elige una orden de comunicación directa con el instrumento	11. Muestra el resultado a la orden seleccionada.
12. Se abre la opción **MonGPIB** ubicada dentro del menú **Monitor** o a través de su acceso directo en la barra de iconos.	13. Muestra el comportamiento de la simulación mediante un trama de datos que ilustra cómo se realizó el envió de órdenes al controlador.

Tabla 5.7. Flujos de eventos del caso de uso Configurar instrumento.

Flujo de eventos básico	
Acción del actor	**Respuesta del sistema**
1. El usuario ejecuta la opción **Configurar instrumento** del menú **Bus**.	2. Muestra la ventana **Configurar instrumentos**.
3. Selección de la dirección primaria del instrumento a configurar.	4. Muestra el instrumento seleccionado.
5. Proporciona los valores deseados para el dispositivo.	6. Acepta los datos de configuración.
6. Selecciona el objeto a configurar para finalizar la configuración del dispositivo.	7. Cierra la ventana.
Flujos alternativos	
1. El usuario elige el instrumento a configurar y accede directamente al instrumento.	2. Muestra el instrumento seleccionado.
3. Se introducen valores a las funciones del instrumento.	4. Aceptan los datos de configuración.

Tabla 5. 8. Flujos de eventos del caso de uso Ayuda.

Flujo de eventos básico	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona la opción **Ayuda** del sistema.	2. Presenta un listado de tres opciones de ayuda del sistema.
3. Elige la opción **Ayuda del SepiGPIB**.	4. Muestra la ayuda del sistema referente al funcionamiento del programa.
5. Se elige el tema a consultar.	6. Muestra la ayuda correspondiente al modulo consultado.
Flujos alternativos	
3. Elige otra opción o recurre a los manuales en formato PDF incorporados al sistema.	4. Muestra los archivos en formato PDF.

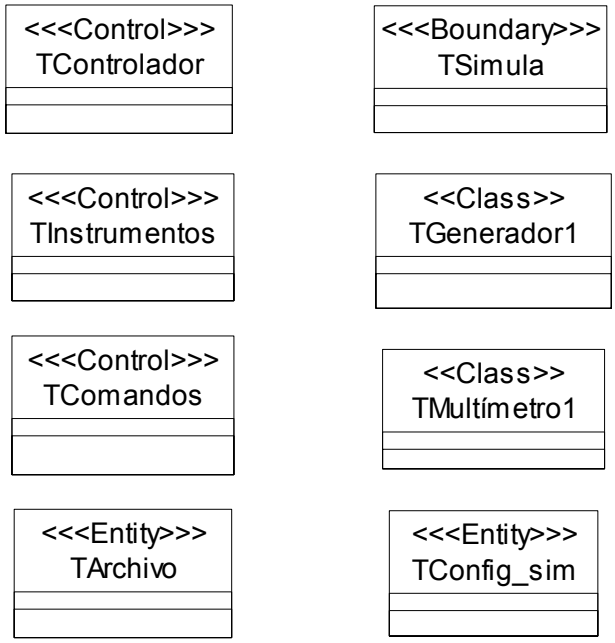


Figura 5.3. Clases identificadas por abstracción.

5.4. Análisis y diseño del SepiGPIB

El análisis se realizo mediante el modelado de las clases, miembros y especificaciones que constituyen el problema real y las relaciones estáticas, o de uso, que existen entre ellas, dicho análisis tiene la finalidad de obtener modelos que se ajusten mejor al problema real.

Con respecto al diseño, se baso en los atributos útiles y los comportamientos no definidos para cada objeto considerando el modelado obtenido de la fase de análisis.

5.4.1. Análisis

La etapa de análisis incluyó la identificación de los conceptos referentes al dominio del problema y de las operaciones y funciones requeridas por el sistema final.

Se propuso un bosquejo inicial de la arquitectura del sistema como referencia para el análisis y establecimiento de un conjunto de mecanismos de ayuda para dividir y organizar el sistema.

La fase de análisis comenzó con la especificación del problema y de las funciones de cada bloque, en este contexto, las funciones son vistas como servicios que ofrece el sistema y cada función se clasifica en evidente u oculta, de acuerdo a sí su visibilidad contribuye al sistema o debe ser oculta a los usuarios, los atributos del sistema se usan principalmente cuando se relacionan funciones específicas.

Las abstracciones identificadas a partir del problema son clases, capas, atributos, asociaciones, agregación, multiplicidad y unificación de las clases, las cuales se presentan en los incisos siguientes.

5.4.1.1. Abstracciones clave

La abstracción consistió en realizar un análisis de las frases nominales de cada caso de uso y en identificar las clases a partir del comportamiento descrito por los casos de uso. La figura 5.3 presenta las clases identificadas en el modelado de casos de uso.

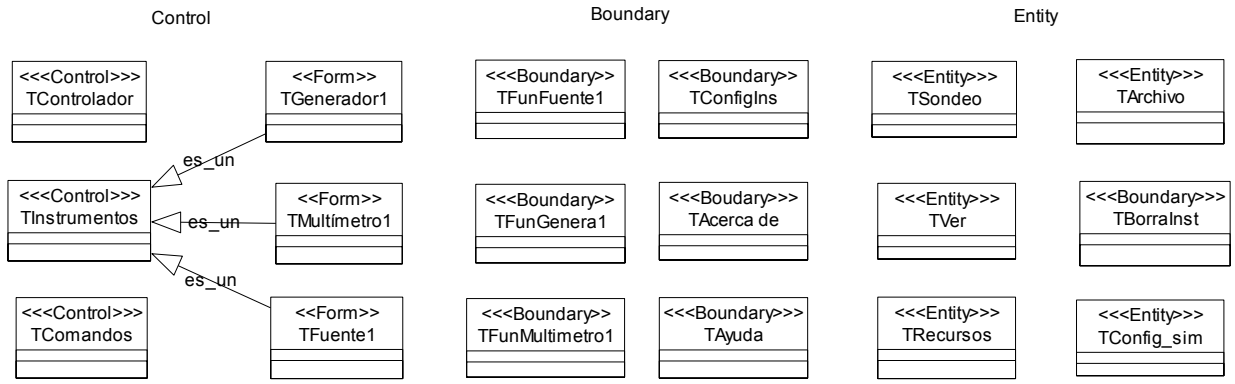


Figura 5.4. Clasificación de las clases del SepiGPIB.

5.4.1.2. Definición de capas

El SepiGPIB presenta distintas capas o niveles de ejecución al usuario final del sistema y las clasifica de acuerdo al tipo de transparencia de datos e información que se requiere. Cada capa del sistema realiza una función específica, las capas de ejecución diseñadas para el SepiGPIB son capa de presentación, capa de reglas de negocios y capa de almacenamiento. La capa que puede ser accesada por el usuario es la capa de presentación. A continuación se describen las capas manejadas por el sistema:

- *Capa de presentación*: se encarga de presentar los instrumentos, la aplicación del controlador, la configuración de la simulación, los mensajes y los datos que interactúan directamente con el usuario. El sistema muestra una lista de sondeo donde aparecen los elementos seleccionados.
- *Capa de reglas de negocios*: cuando se realiza una simulación, el sistema manipula una gran cantidad de datos referentes a los instrumentos. La capa de negocios cuenta con procesos para su manipulación, mientras que la capa de presentación y el controlador se encargan de capturarlos y de administrarlos respectivamente.
- *Capa de almacenamiento*: el sistema interactúa con un archivo de registro encargado de almacenar el comportamiento del GPIB en una simulación.

5.4.1.3. Descripción de los atributos utilizando notación UML

Cada una de las clases se clasifica en base a su funcionamiento dentro del sistema y pueden ser de control (Control), de intefaz (Boudary) y de almacenamiento (Entity) (figura 5.4).

5.4.1.4. Asociaciones utilizando notación UML

Existen diferentes tipos de asociaciones en el SepiGPIB, algunas de ellas son identificadas a partir del comportamiento de las clases y otras en base a su función dentro del sistema. La figura 5.5 presenta algunas de las asociaciones identificadas en el SepiGPIB.

5.4.1.5. Agregación utilizando notación UML

Una agregación se obtiene a partir del comportamiento de una clase y se origina al tener una clase que es parte de otra. El SepiGPIB presento agregaciones lógicas a partir del diseño del software, las relaciones de agregación son manejadas independientemente por cada modulo diseñado en el sistema (figura 5.6).

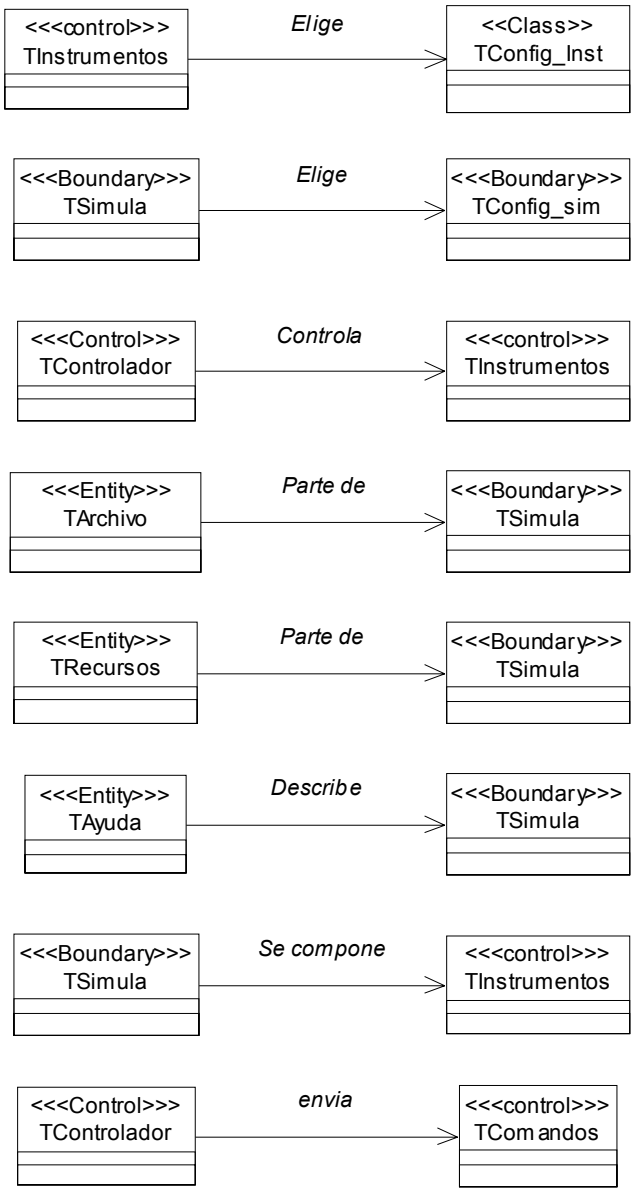


Figura 5.5. Asociaciones identificadas en el SepiGPIB.

5.4.1.6. Multiplicidad

Una relación de multiplicidad en el sistema se realiza al contar con una comunicación entre clases. La creación de objetos o relaciones de asociación permite identificar relaciones de multiplicidad entre las clases.

Existen diferentes tipos de multiplicidad en el sistema, cada uno de los cuales es modelado en base al comportamiento de las clases principales en el modelado de casos de uso. Al definirse una relación de multiplicidad se tiene la asociación directa entre los objetos de cada clase. La figura 5.7 ilustra las relaciones de multiplicidad identificadas en el SepiGPIB.

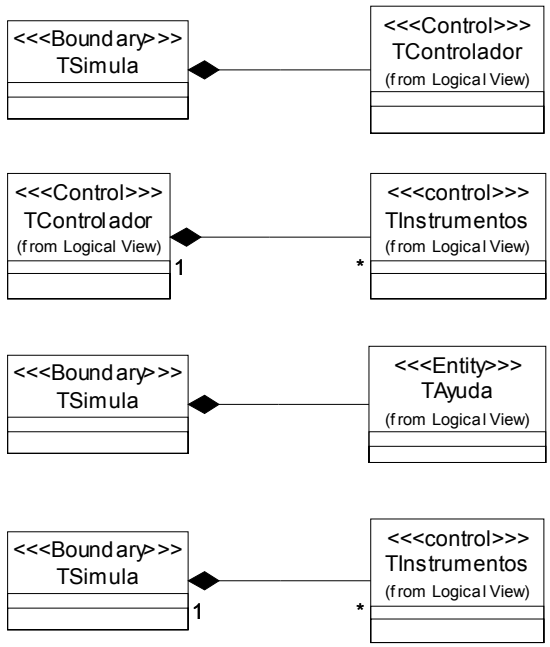


Figura 5.6. Agregaciones identificadas en el SepiGPIB.

5.4.2. Diseño

El desarrollo de la fase de diseño se basó en describir los componentes del software a implementar en el sistema, finalmente estos modelos se utilizan para crear un modelo de clases del proyecto.

Se diseñaron las fases de ayuda para definir el comportamiento de las clases y de los objetos manejados por el sistema. El modelado conceptual, los diagramas de clases, de secuencia y colaboración describen el comportamiento dinámico del sistema.

5.4.2.1. Modelo conceptual

El proceso, utilizado por el análisis y diseño orientado a objetos, hace uso del modelado conceptual para especificar el manejo de los objetos empleados en el sistema para poder determinar cuáles son los conceptos y términos a utilizar.

Como se describe en el subcapítulo 2.5, un modelado conceptual ilustra los conceptos del dominio de un problema y los artefactos más importantes creados durante el AOO, después de identificar las clases y asociaciones, se procedió a indicar las relaciones entre los conceptos y a crear los atributos necesarios para satisfacer la información de los requerimientos.

El modelo conceptual ilustra la relación existente entre la clase TSimula y cada una de las clases asociadas, indicando sus responsabilidades. La clase TControlador se encarga de manipular los módulos de los instrumentos, archivos y configuraciones de las clases asociadas a la simulación. Este modelo también ilustra el tipo de clase que se está manipulando, es decir, si pertenece a una clase de control, interfaz o de almacenamiento. La figura 5.8 muestra el modelo conceptual del proyecto.

Una función importante de las asociaciones es permitir la relación directa entre las clases y los objetos generados a partir de la asociación, favoreciendo con ello la reutilización de funciones en el sistema, además ayuda a identificar los enlaces entre las distintas clases, a esta relación se le conoce como multiplicidad.

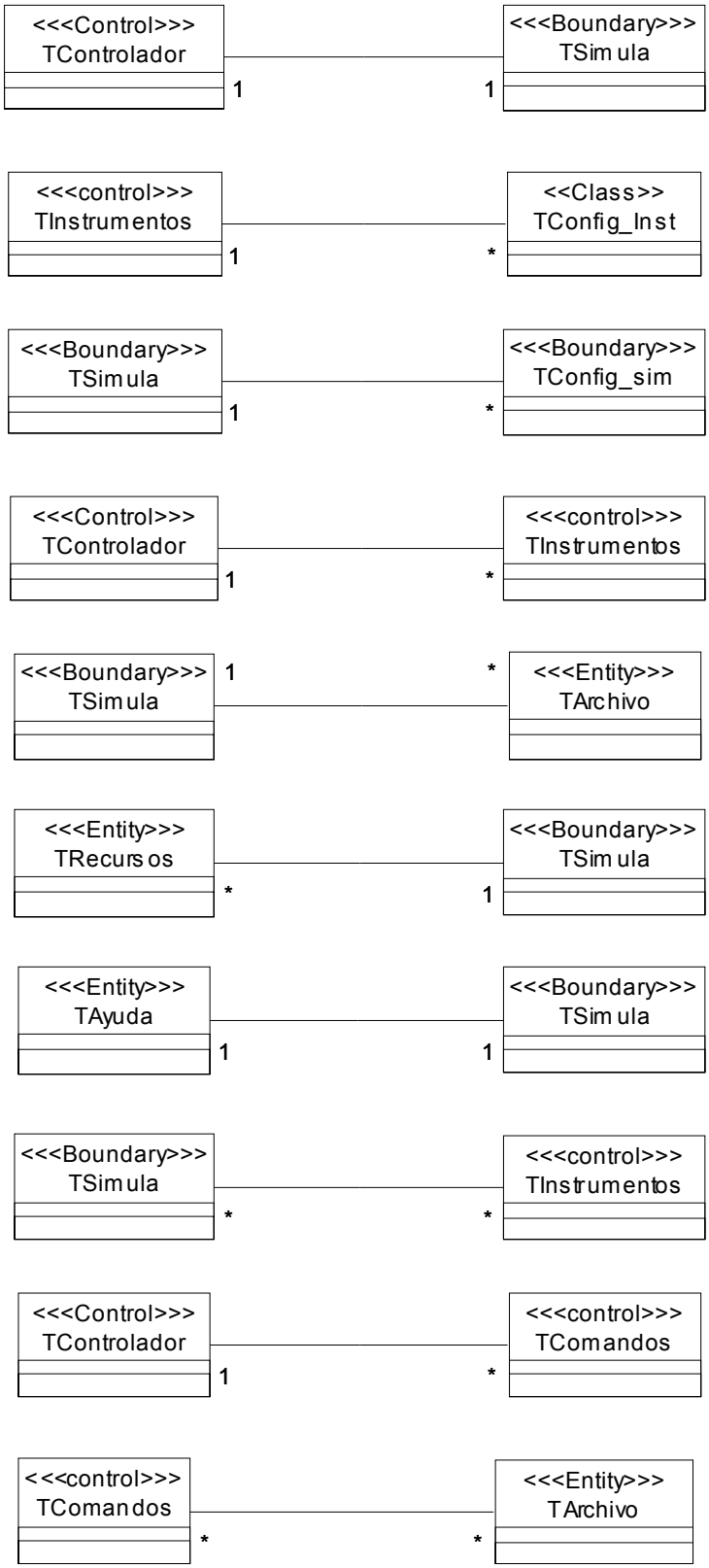


Figura 5.7. Multiplicidad identificada en el SepiGPIB.

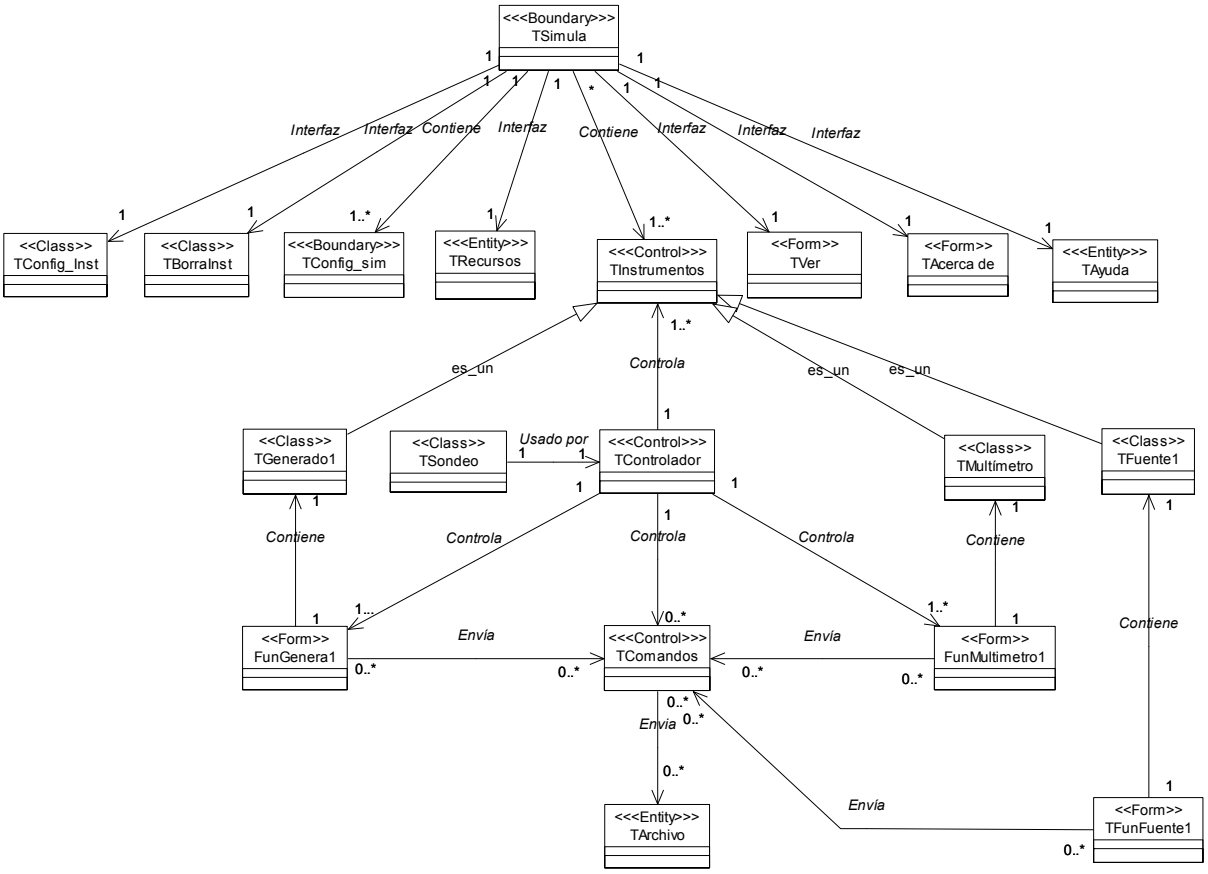


Figura 5.8. Diagrama del modelo conceptual para el SepiGPIB.

5.4.2.2. Diagrama de clases

El diagrama de clases ilustra los métodos de cada clase, sus atributos, el tipo de información y la visibilidad entre clases, se utiliza, junto al diagrama de colaboración, para agregar asociaciones y atributos a las clases.

En esta fase se realizaron actividades para la identificación de clases del sistema, la forma en la cual se llevo la identificación de clases fue mediante el análisis de los casos de uso y la adopción de las estrategias identificación de clases por categoría y por frases nominales. La figura 5.9 muestra el diagrama de clases identificado a partir de las necesidades del sistema.

Las clases identificadas se agruparon en tres perspectivas diferentes de control, de interfaz y de almacenamiento, donde cada una de las clases identificadas forma parte de una estructura inicial y describen las relaciones estáticas existentes. Los diagramas encontrados a partir de este análisis muestran las asociaciones, la generalización, los atributos, las operaciones y agregación entre las clases.

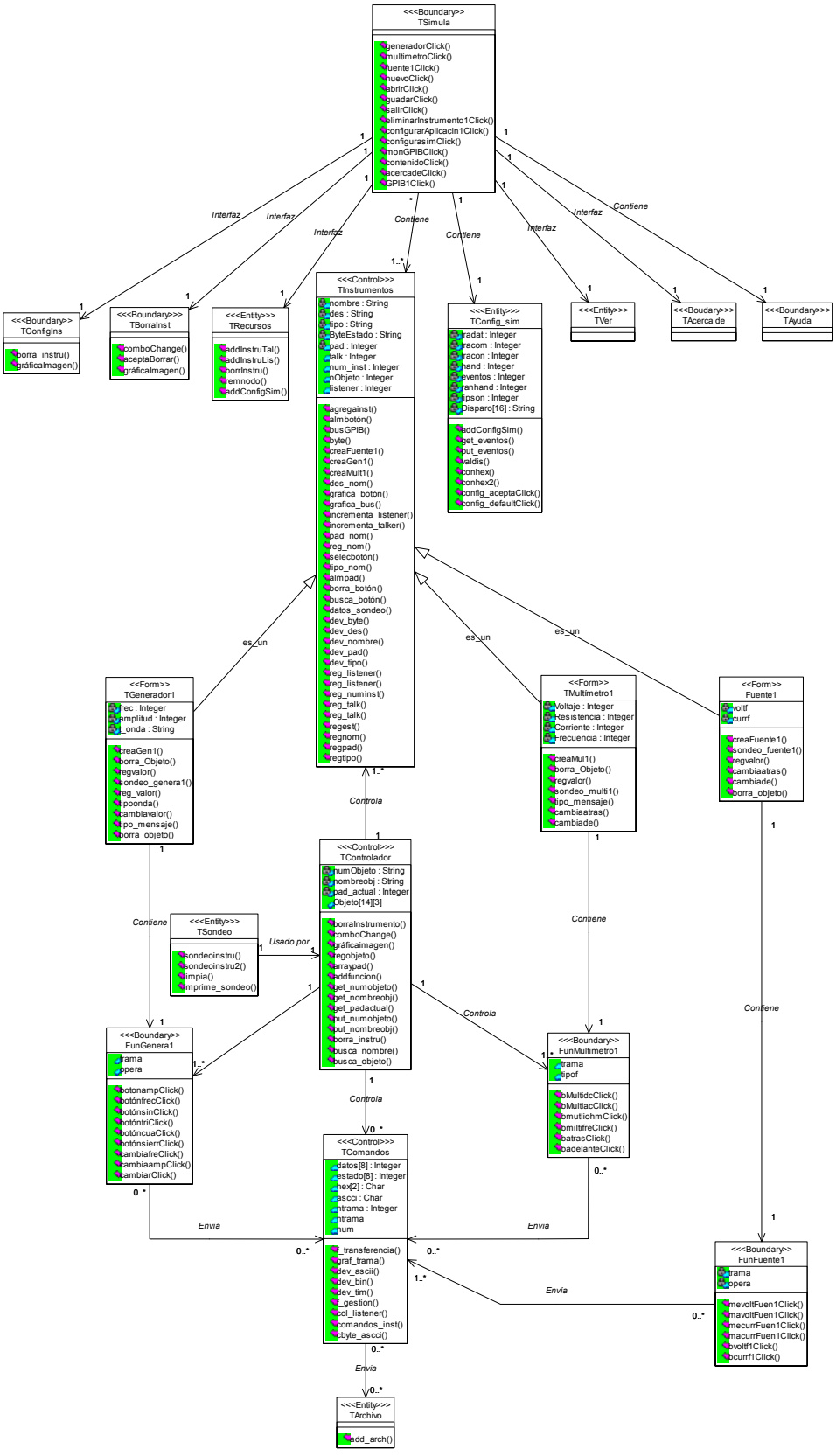


Figura 5. 9. Diagrama de clases del SepiGPIB.

5.4.2.3. Diagrama de secuencia

El diagrama de secuencia resalta el orden de los eventos entre las clases y los objetos, dichos diagramas ilustran los eventos que ocurren desde el punto de vista del actor que utiliza el sistema. Un diagrama de secuencia es un escenario en particular de un caso de uso que toma los eventos externos generados por el actor y presenta el orden en el cual se generan.

5.4.2.4. Diagrama de colaboración

Los diagramas de colaboración ilustran la interacción de mensajes entre los objetos definidos en el modelo conceptual, dentro de los diagramas de colaboración se asignan responsabilidades a los objetos utilizados.

Los diagramas de colaboración muestran las interacciones entre los objetos en un formato de red, donde se muestra a los objetos como iconos y los mensajes son numerados por orden y precedencia.

5.4.2.5. Especificación de la realización de casos de uso

La especificación de un caso de uso es una descripción de los aspectos de su implementación y sirve como complemento a la descripción del modelado ya que tiene la función de documentar y registrar las operaciones en el sistema.

La realización de casos de uso muestra la interacción entre las clases por cada caso de uso, los mensajes que se envían y su secuencia correspondiente, además, ilustra gráficamente el diseño formal del sistema a construir y la lógica de su funcionamiento.

A continuación se describen diferentes casos de uso y sus especificaciones.

5.4.2.5.1. Especificación de la realización del caso de uso Agregar instrumentos

El caso de uso Agregar instrumento es parte esencial del sistema ya que es necesario agregar un instrumento para realizar alguna operación en el sistema, además, cada instrumento contiene una configuración especial de acuerdo a su funcionamiento en particular.

Es importante resaltar que cada instrumento agregado al bus cuenta con un número restringido de características y operaciones, debido a que el objetivo es simular funciones básicas mediante operaciones en el bus.

El diagrama de clase presentado en la figura 5.10 muestra el comportamiento estático que se realiza al agregar un instrumento, así como las clases involucradas en la creación de objetos pertenecientes a la clase TInstrumentos dentro de la simulación.

El diagrama de secuencia (figura 5.11) ilustra la forma en la cual se agrega un instrumento mediante un comportamiento secuencial, en tal caso, el instrumento agregado es un generador de funciones creado a partir de la clase TGenerador1 y registrado en las clases TRecursos y TControlador.

El diagrama de secuencia alterno ilustra el comportamiento del flujo de datos al no agregar un instrumento emisor, en la figura 5.12 se ilustra la agregación de un instrumento de la clase Tmultímetro1.

La figura 5.13 ilustra el diagrama de colaboración que presenta la forma en que fueron enviados los mensajes para la creación de un instrumento generador, así como la participación y colaboración de las clases del sistema involucradas en este proceso.

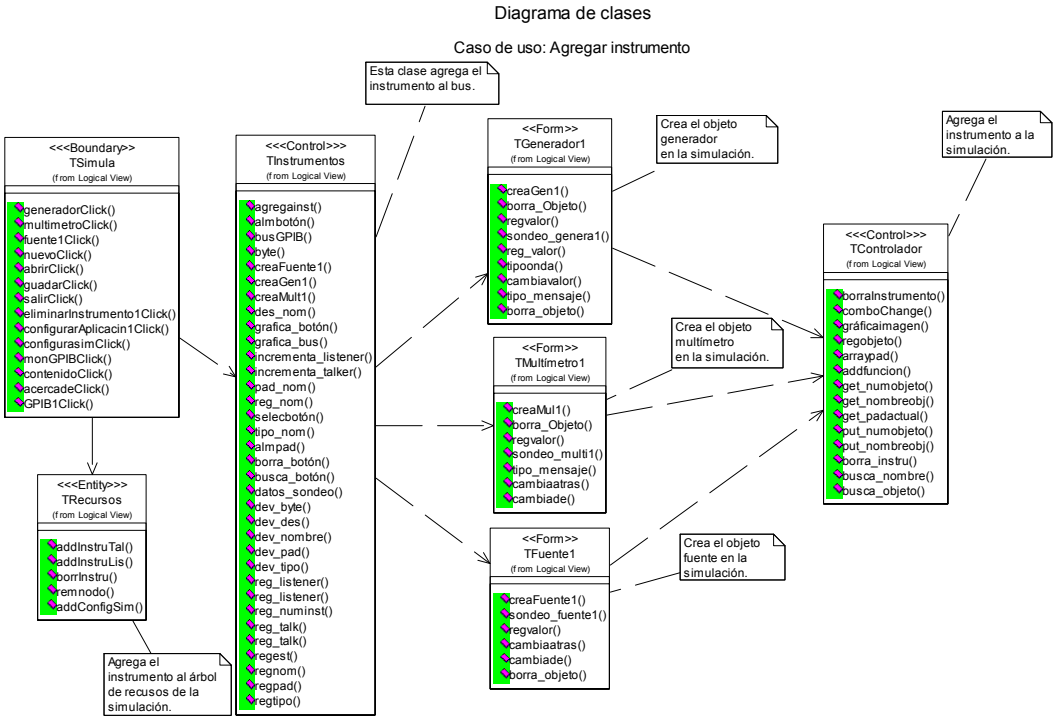


Figura 5.10. Diagrama de clases para el caso de uso Agregar instrumento.

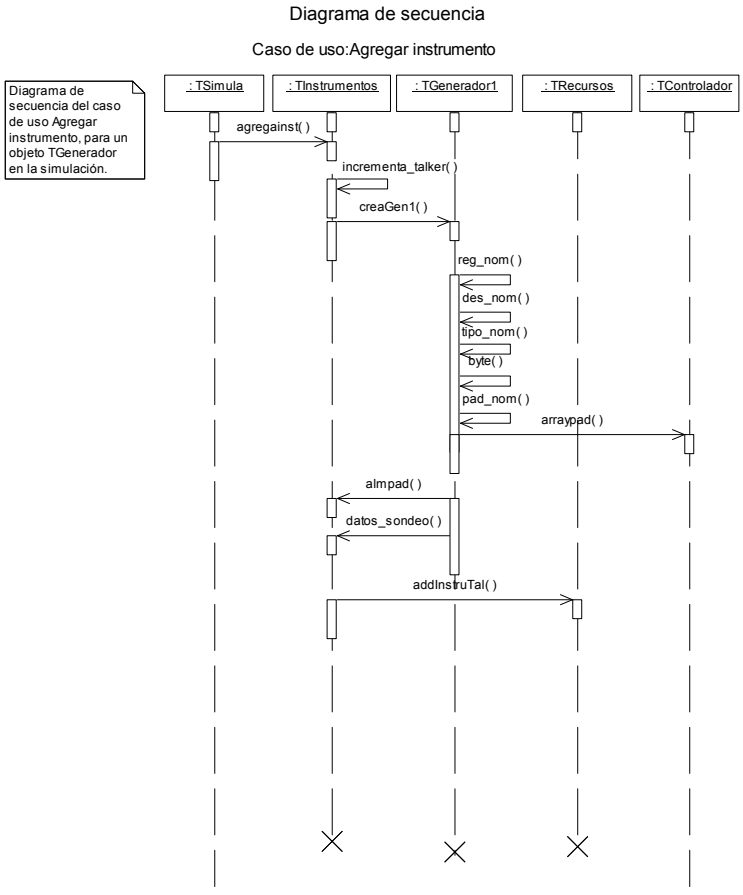


Figura 5.11. Diagrama de secuencia para el caso de uso Agregar instrumentos.

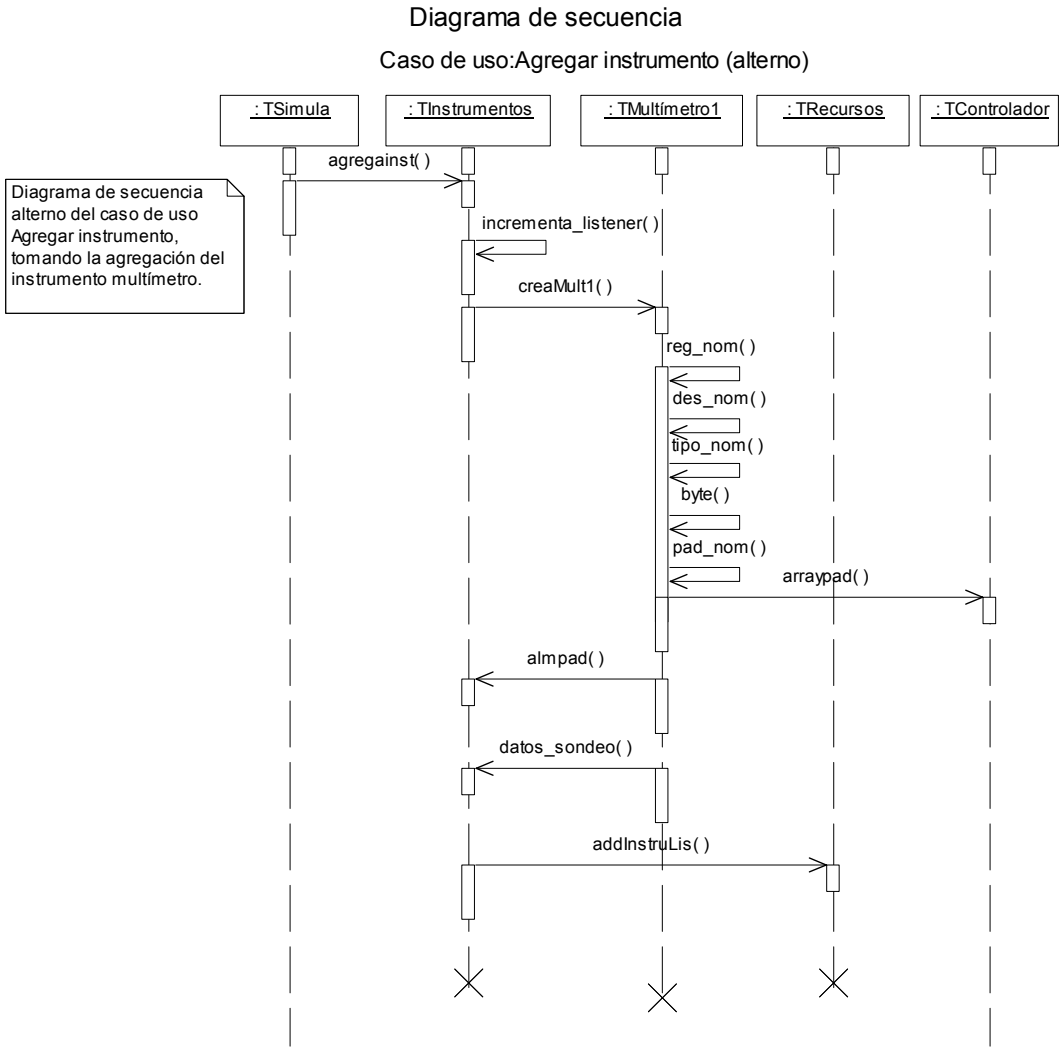


Figura 5.12. Diagrama de secuencia alterno para el caso de uso Agregar instrumento.

5.4.2.5.2. Especificación de la realización del caso de uso Eliminar instrumento

El caso de uso Eliminar instrumento tiene la función de brindar al usuario del sistema la posibilidad de dar de baja un dispositivo en la simulación y remover cada uno de sus registros creados por el sistema. Otra razón importante para contar con este caso de uso es proporcionar un sistema confiable basado en objetos.

El propósito del diagrama de clases (figura 5.14) es mostrar, de forma clara y sencilla, cómo el sistema elimina un instrumento activo en el bus mediante la interacción de sus clases.

El diagrama de secuencia (figura 5.15) muestra las interacciones entre las clases, los mensajes que se envían y su secuencia correspondiente, además, ilustra el diseño formal del sistema a construir y la lógica de su funcionamiento.

La función del diagrama de colaboración es presentar, en forma grafica, cómo se envían y reciben los mensajes en una simulación y presentar la secuencia numerada de un proceso de finalización, cada una de las clases involucradas (figura 5.16) colaboran para la eliminación de un objeto de clase TInstrumentos.

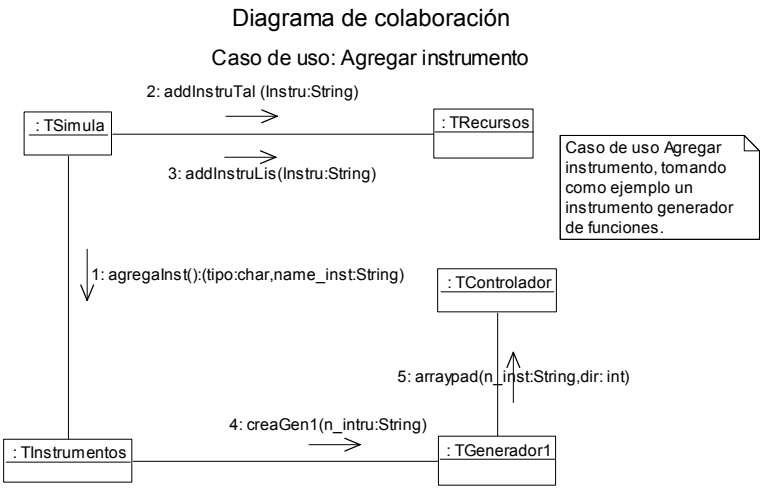


Figura 5.13. Diagrama de colaboración para el caso de uso Agregar instrumento.

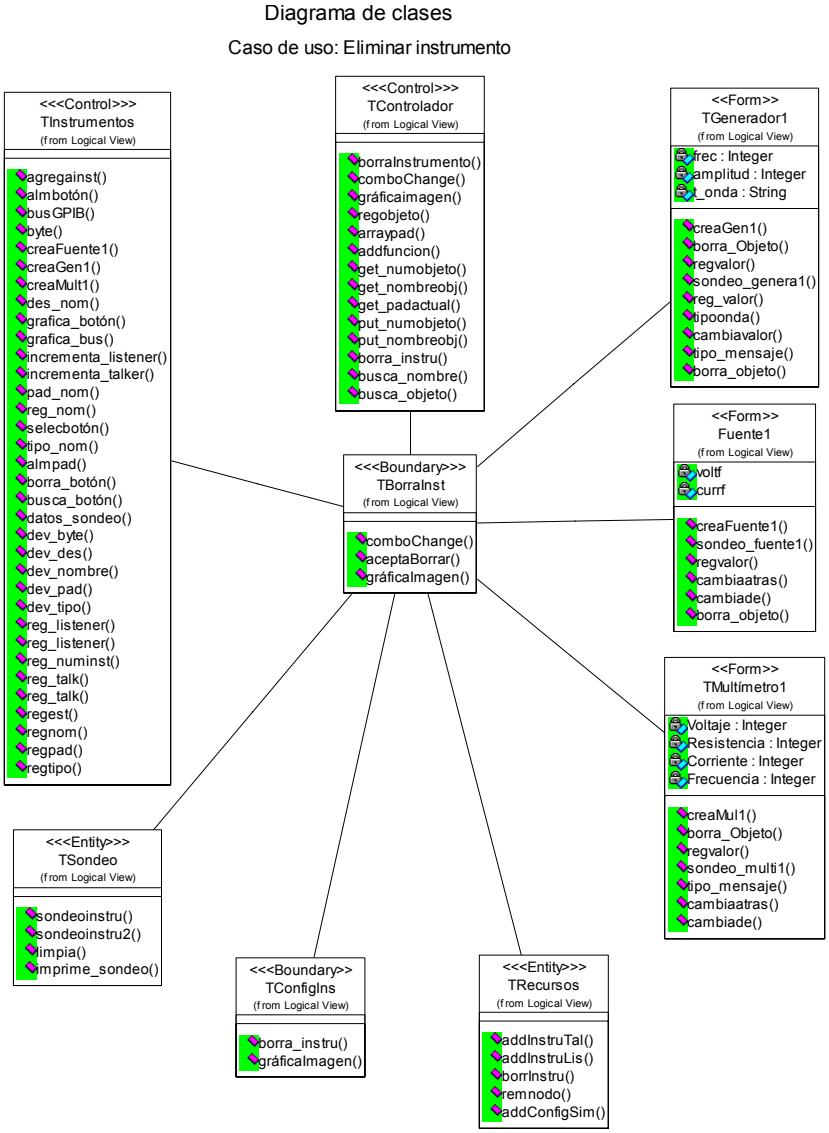


Figura 5.14. Diagrama de clases para el caso de uso Eliminar instrumento.

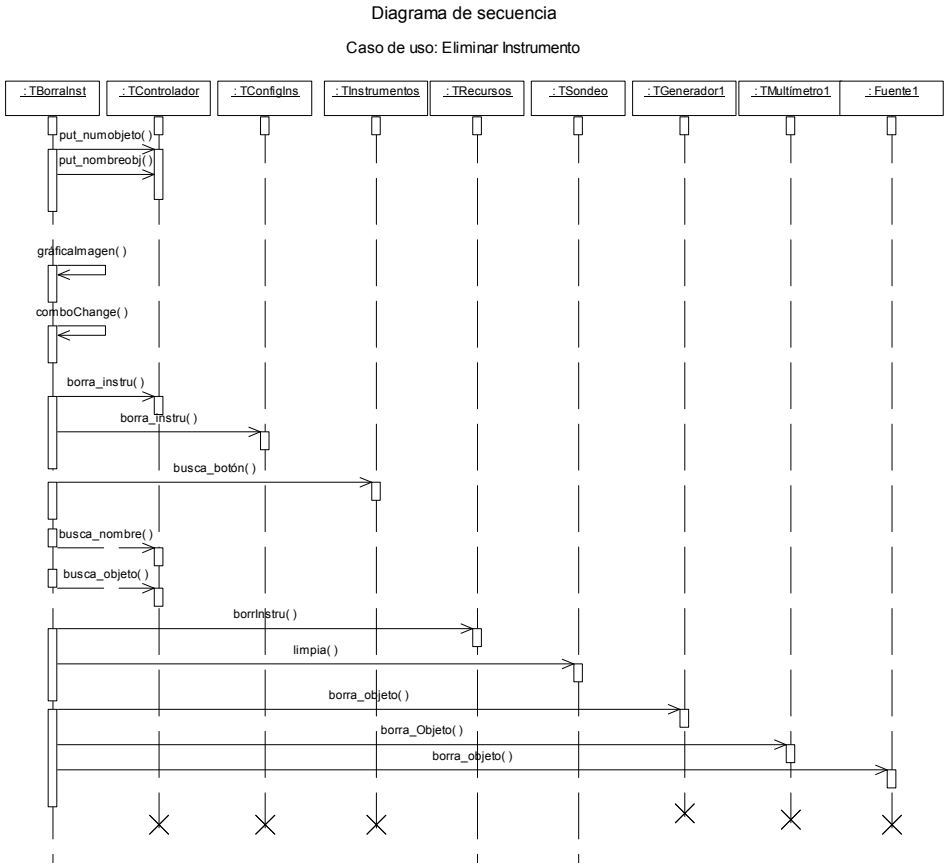


Figura 5.15. Diagrama de secuencia para el caso de uso Eliminar instrumento.

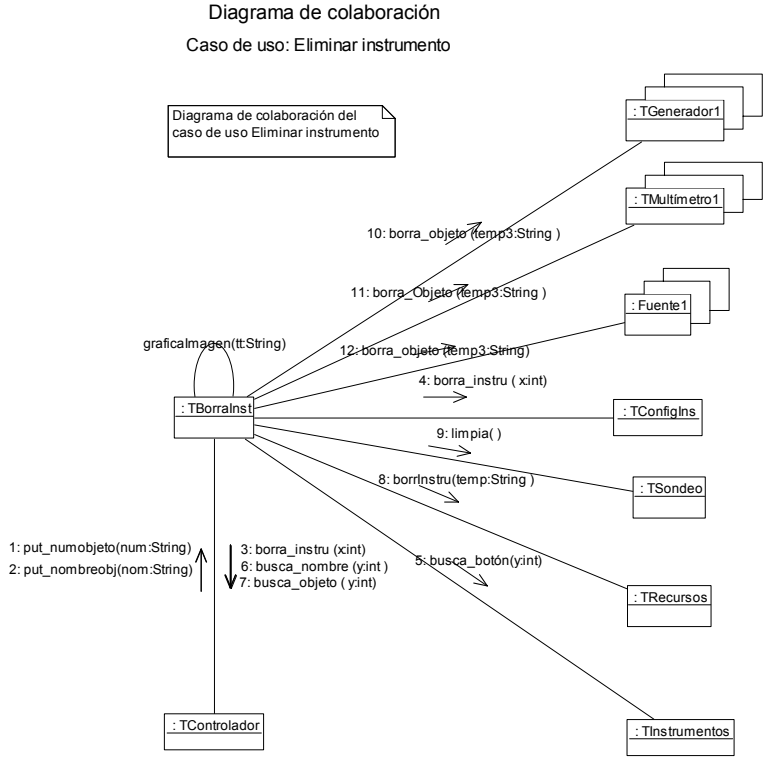


Figura 5.16. Diagrama de colaboración para el caso de uso Eliminar instrumento.

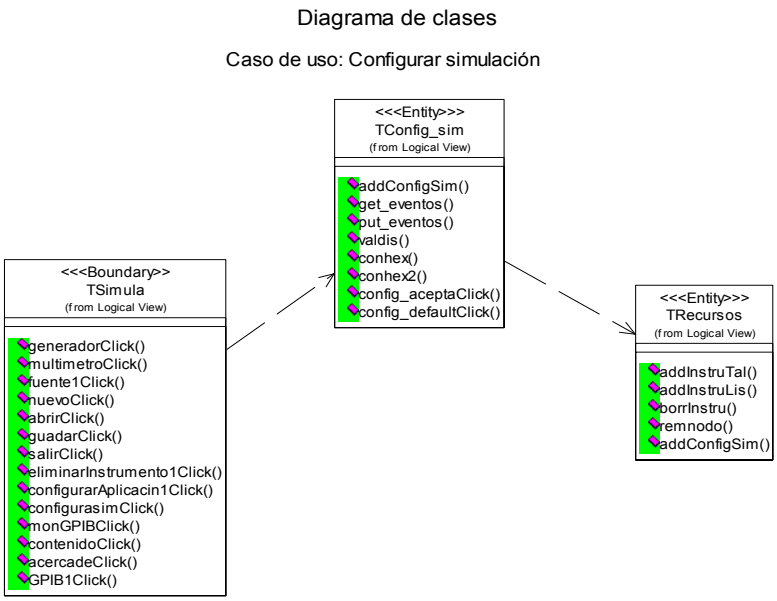


Figura 5.17. Diagrama de clases para el caso de uso Configurar simulación.

5.4.2.5.3. Especificación de la realización del caso de uso Configurar simulación

El caso de uso Configurar simulación es de gran importancia para el sistema debido a que en este modulo se determinan los parámetros de la simulación y se configuran las salidas de las tramas de datos del monitor de bus (**MonGPIB**).

Este modulo realiza un filtrado de información correspondiente a las operaciones de la simulación, los datos son filtrados al monitor de bus.

El SepiGPIB permite modificar las opciones de salida de trama, configuración de disparo, registro de eventos y el tipo de sondeo, el modulo de configuración brinda la posibilidad de contar con una configuración predeterminada por el sistema.

La figura 5.17 presenta las clases involucradas al configurar la simulación, la clase principal TSimula cuenta con un acceso a la clase TConfig_sim que permite personalizar la simulación.

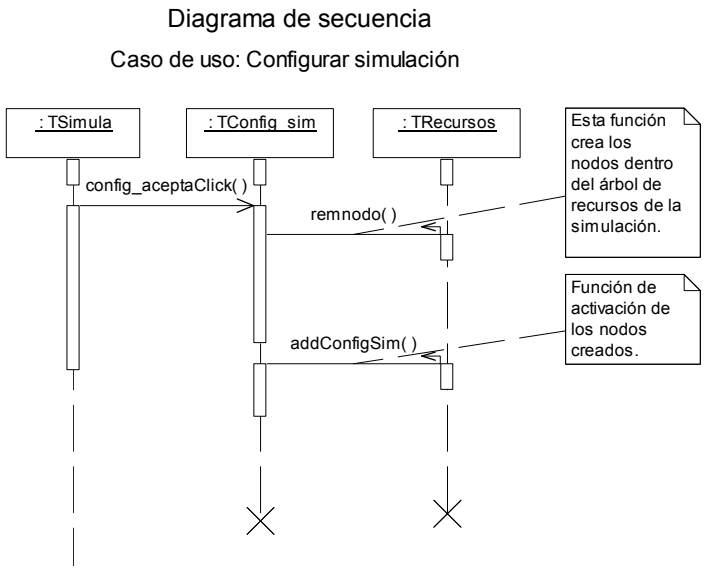


Figura 5.18. Diagrama de secuencia para el caso de uso Configurar simulación.

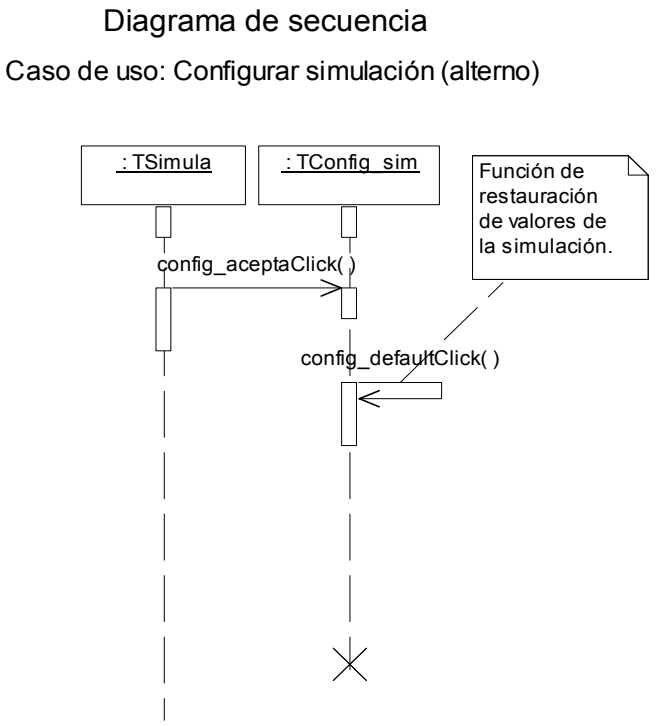


Figura 5.19. Diagrama de secuencia alterno para el caso de uso Configurar simulación.

El propósito del diagrama de secuencia (figura 5.18) es mostrar las clases y eventos que se siguen al configurar una simulación, ilustrando la secuencia real de los eventos.

La figura 5.19 ilustra una secuencia de eventos alternos, los cuales son resultado de elegir una configuración predeterminada por el sistema.

Mediante la utilización de los mensajes que se envían entre las clases y su secuencia, se puede realizar un bosquejo de los módulos de colaboración del sistema y de la lógica de su funcionamiento. La figura 5.20 ilustra la colaboración entre las clases al configurar una simulación en el sistema.

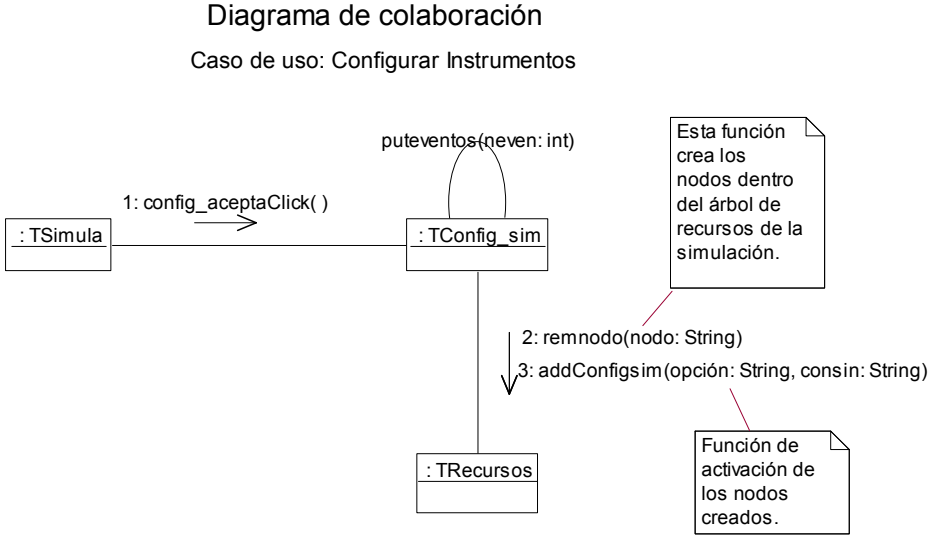


Figura 5.20. Diagrama de colaboración para el caso de uso Configurar instrumento.

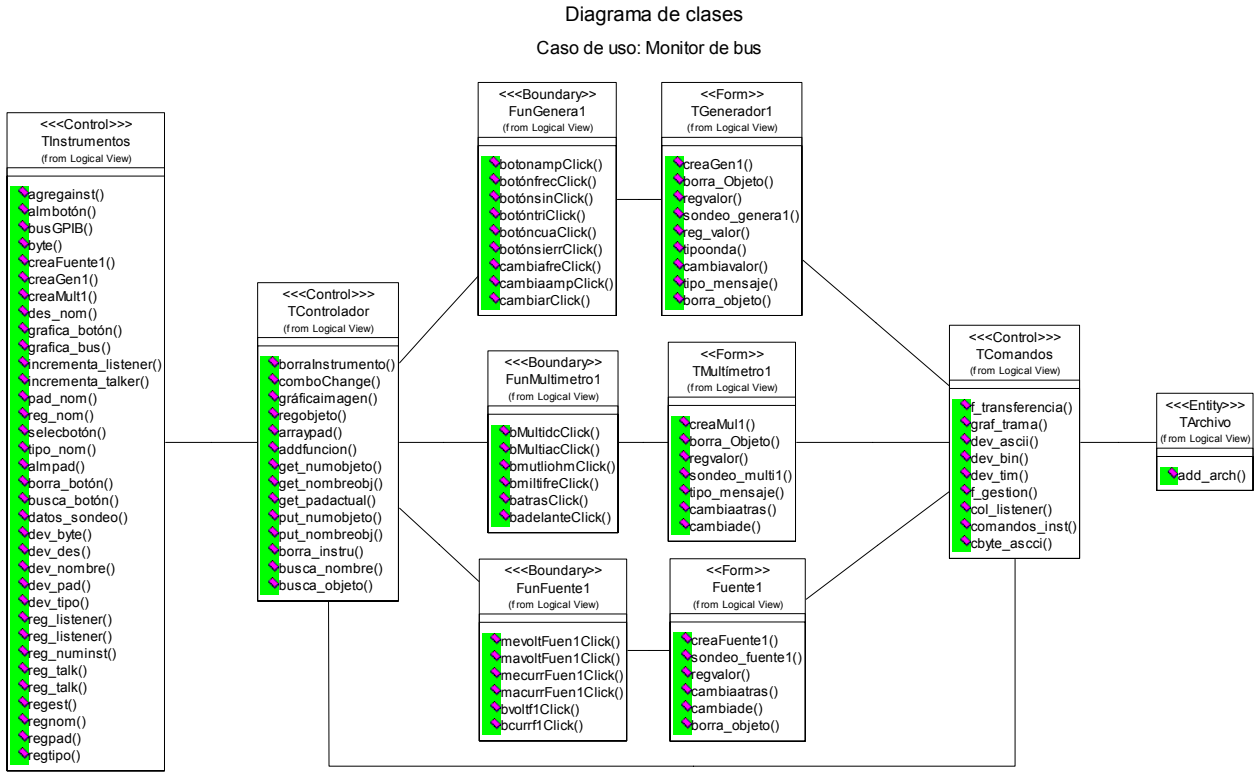


Figura 5.21. Diagrama de clases para el caso de uso Monitor de bus.

5.4.2.5.4. Especificación de la realización del caso de uso Monitor de bus

Este caso de uso es considerado como primario y esencial debido a que en el se implementan las funciones principales del sistema SepiGPIB.

El proporcionar un instrumento, configurarlo y realizar una consulta en forma directa o a través de las funciones del instrumento es una de las tareas del SepiGPIB. El controlador del bus manipula el comportamiento de los instrumentos y su comunicación en forma directa o mediante funciones del bus.

El monitor de bus muestra los eventos mediante las señales presentes en el bus de comunicaciones y sus respectivos nemónicos asociados.

La figura 5.21 ilustra la interacción de las clases al realizar una operación de monitoreo, en dicha figura se tiene el diagrama de clases como resultado de una operación de monitoreo de actividades registradas a partir de un instrumento de la clase TGenerador1.

La figura 5.22 ilustra la interacción de las clases al realizar una operación de monitoreo, presenta las funciones utilizadas para realizar el monitoreo de las señales en el bus y el tiempo de vida de los objetos creados.

El diagrama de colaboración (figura 5.23) presenta el flujo de mensajes enviados a cada una de las clases involucradas en la operación de monitoreo del bus.

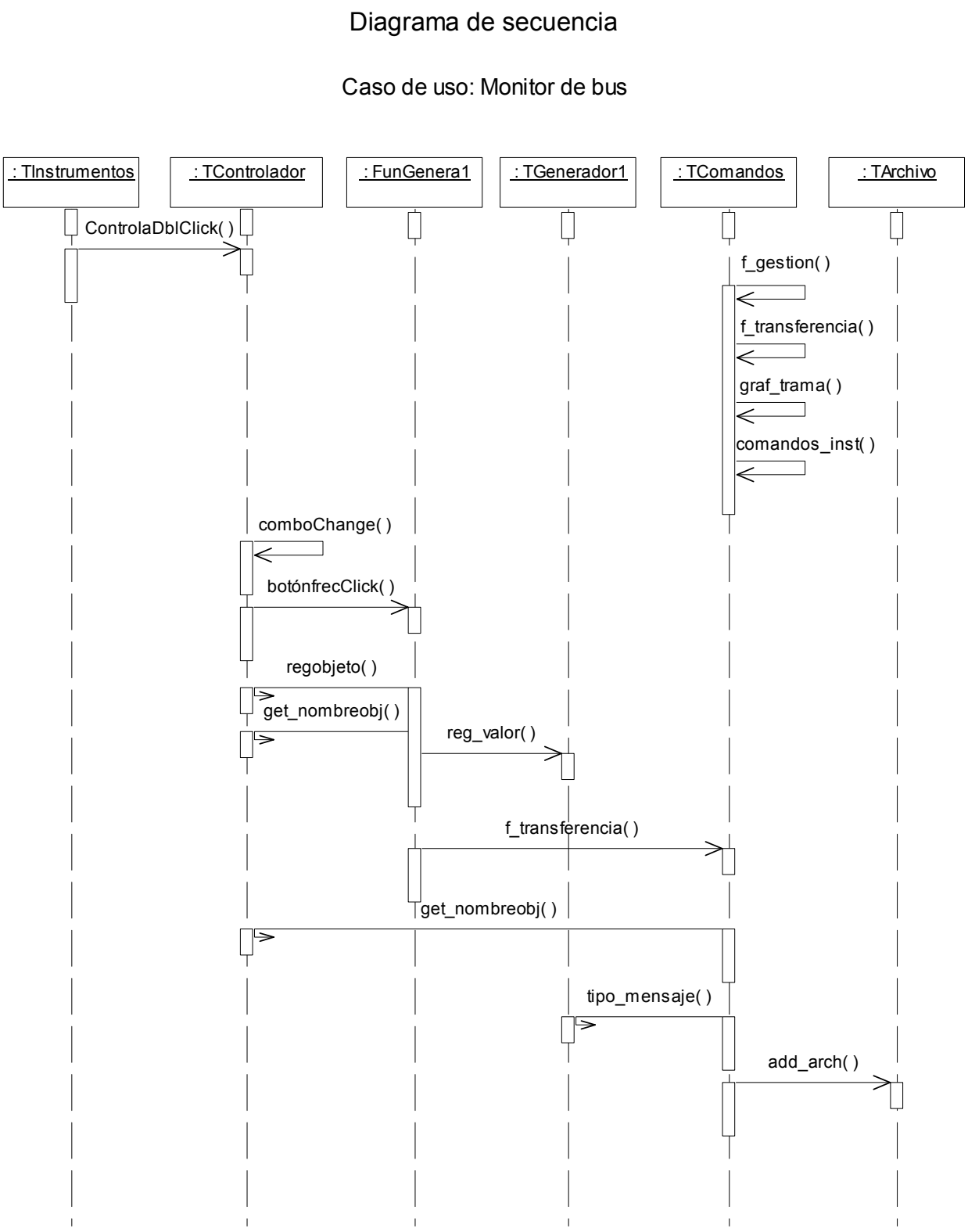


Figura 5.22. Diagrama de secuencia para el caso de uso Monitor de bus.

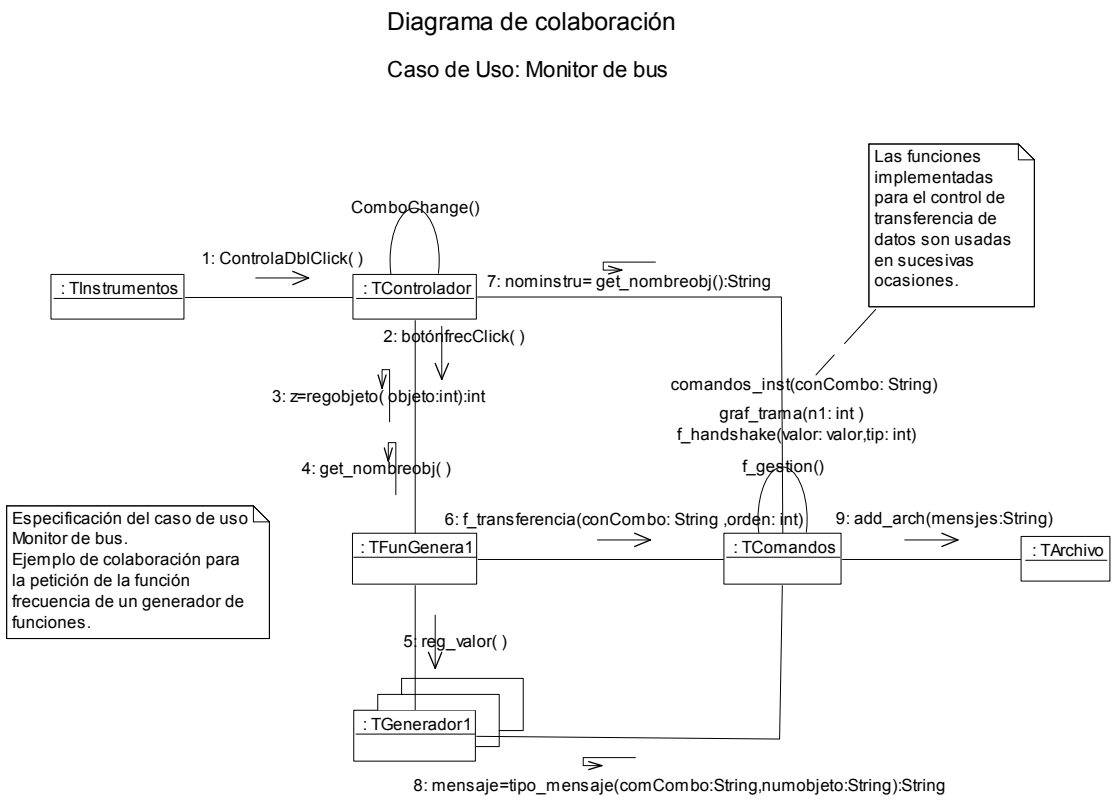


Figura 5.23. Diagrama de colaboración para el caso de uso Monitor de bus.

5.4.2.6. Diseño de clases

El diseño de clases parte de la identificación de las clases de mayor importancia en el sistema, analizando cada uno de sus atributos, responsabilidad, asociación y multiplicidad, para realizar un bosquejo inicial del funcionamiento individual de cada uno de los módulos que integraran el sistema. La finalidad de diseñar una clase es verificar cada una de sus relaciones existentes.

5.4.2.6.1. Diseño de clase TComandos

La clase TComandos permite enviar comandos u órdenes a los instrumentos activos en el bus y almacenar la respuesta de los instrumentos en un archivo como resultado de la simulación, además controla la comunicación a través de órdenes, permitiendo a los usuarios conocer del estado de los instrumentos activos. Otra responsabilidad de esta clase es la de enviar datos para ser registrados en el archivo de simulación, la clase TControlador es de tipo control.

La multiplicidad que participa en la relación de las clases TControlador y TComandos es de uno a muchos debido a que un controlador puede utilizar muchas órdenes y una orden sólo puede ser utilizada por un controlador.

La relación de multiplicidad existente entre las clases TComandos y TArchivo es de muchos a muchos, ya que una orden puede aparecer en muchos archivos y en un archivo pueden aparecer muchas órdenes.

Existe una asociación entre las clases TControlador y TComandos debido a que el controlador envía órdenes, los atributos encontrados para esta clase fueron datos[8], estado[8], hex[2], ascii y ntrama, cada atributo encontrado en esta clase ayuda a conformar la trama de datos presente en el bus para ser almacenada en un archivo.

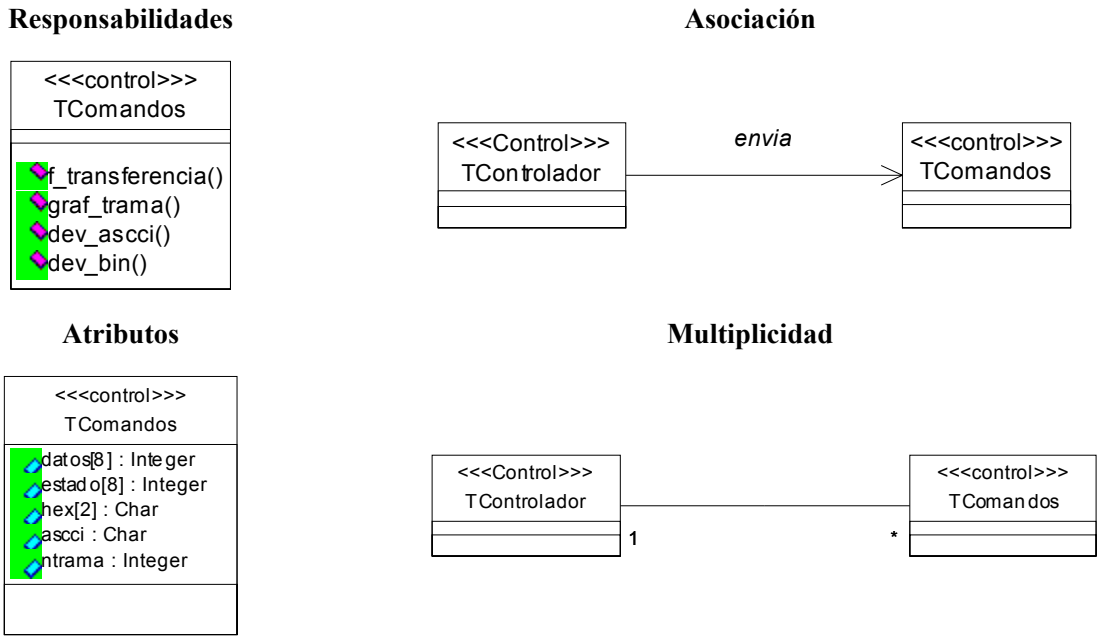


Figura 5.24. Diagrama de diseño de la clase TComandos.

La figura 5.24 ilustra los atributos, responsabilidad en funciones, asociación y multiplicidad de la clase TComandos.

5.4.2.6.2. Diseño de clase TConfig_sim

La clase TConfig_sim permite establecer los valores y parámetros necesarios para realizar una simulación, además registra el tipo de salida de la simulación en base a parámetros asignados, esta clase es de tipo interfaz debido a que en ella se almacenan los registros de la simulación.

La multiplicidad que participa en la relación entre las clases TSimula y TConfig_sim es de uno a muchos debido a que una simulación puede contar con muchas configuraciones y por cada simulación sólo existe una configuración.

Existe una asociación entre las instancia de clases TSimula y TConfig_sim debido a que cada simulación tiene que elegir una configuración del bus.

Los atributos encontrados para esta clase fueron tradat, tracom, tracon, hand, eventos, ranhand, tipson, Disparo[16], cada uno de los atributos es utilizado por el controlador para determinar los parámetros de la simulación.

La figura 5.25 ilustra los atributos, responsabilidad en funciones, asociación y multiplicidad de la clase TConfig_sim.

5.4.2.6.3. Diseño de clase TControlador

La clase TControlador permite realizar la comunicación con los instrumentos activos en el GPIB y llevar a cabo las peticiones de servicios a los instrumentos, en ella se registran y se lleva un control de los instrumentos mediante consultas a los dispositivos, lo que permite asignar correctamente los datos a un dispositivo determinado.

Esta clase permite comunicarse con los dispositivos activos en el bus en base a órdenes y consultas del estado de los instrumentos en el bus.

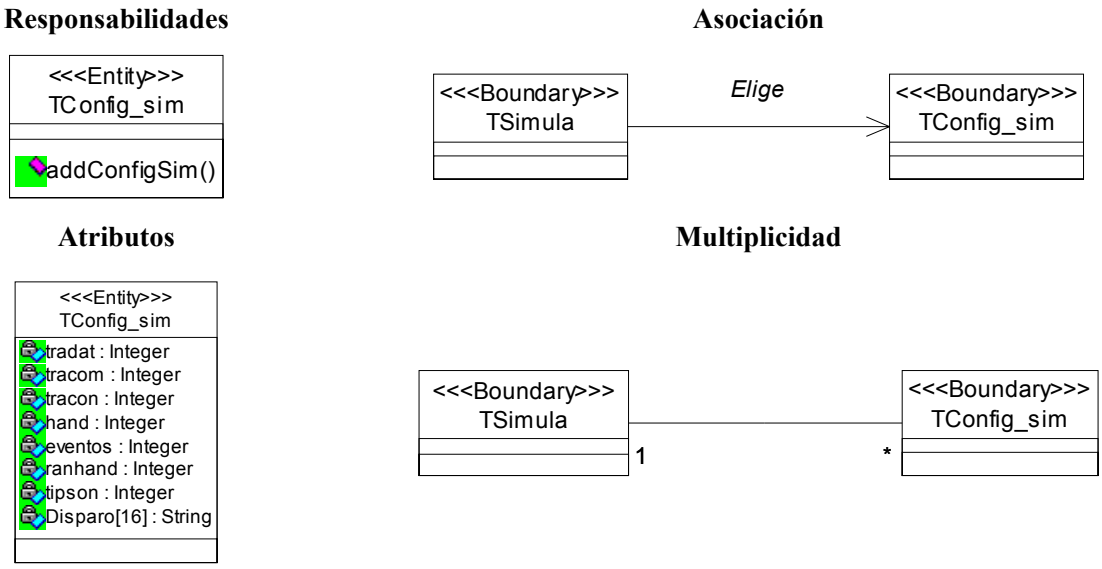


Figura 5.25. Diagrama de diseño de la clase TConfig_sim.

Otra responsabilidad de esta clase es registrar los datos en el archivo de la simulación. Esta clase es de tipo control debido a que manipula la comunicación entre el controlador y los instrumentos.

La multiplicidad que participa en la relación de la clase TControlador y TSimula es de uno a uno debido que un controlador sólo esta presente en una simulación y por cada simulación existe sólo un controlador, la multiplicidad entre las clase TInstrumentos y TControlador es de muchos a uno, donde un instrumento tiene un único controlador y un controlador puede tener muchos instrumentos.

Existe una asociación entre las clase TControlador y TSimula debido a que un controlador se encarga de realizar una simulación, otra asociación se presenta entre las instancias de clase TControlador e TInstrumentos debido a que un conjunto de instancias de instrumentos es manejada exclusivamente por el controlador.

Los atributos encontrados para esta clase fueron numObjeto, nombreobj, pad_actual, Objetos [14][3]. Cada registro de los objetos se lleva a cabo mediante un arreglo que controla los instrumentos en el bus. La figura 5.26 ilustra los atributos, responsabilidad en funciones, asociación y multiplicidad de la clase TControlador.

5.4.2.6.4. Diseño de la clase TInstrumentos

La clase TInstrumentos muestra las características principales de los instrumentos como son nombre, descripción, tipo, octeto de estado, ruta primaria y las variables asociadas al manejo de la clase y de los instrumentos.

El usuario elige un instrumento que contiene valores predeterminados, pero que puede modificar. La clase TInstrumentos es de tipo control debido a que en ella se controlan todos los accesos respecto a la configuración de los instrumentos.

La multiplicidad que participa en la relación entre las clases Tsimula y TInstrumentos es de muchos a muchos debido a que una simulación puede tener muchos instrumentos y un instrumento puede aparecer en diferentes simulaciones según lo desee el usuario. Otra relación de mulltiplicidad existe entre las clases TInstrumentos y TConfigIns y es de uno a muchos debido a que un instrumento puede tener muchas configuraciones y una configuración puede presentarse en muchos instrumentos.

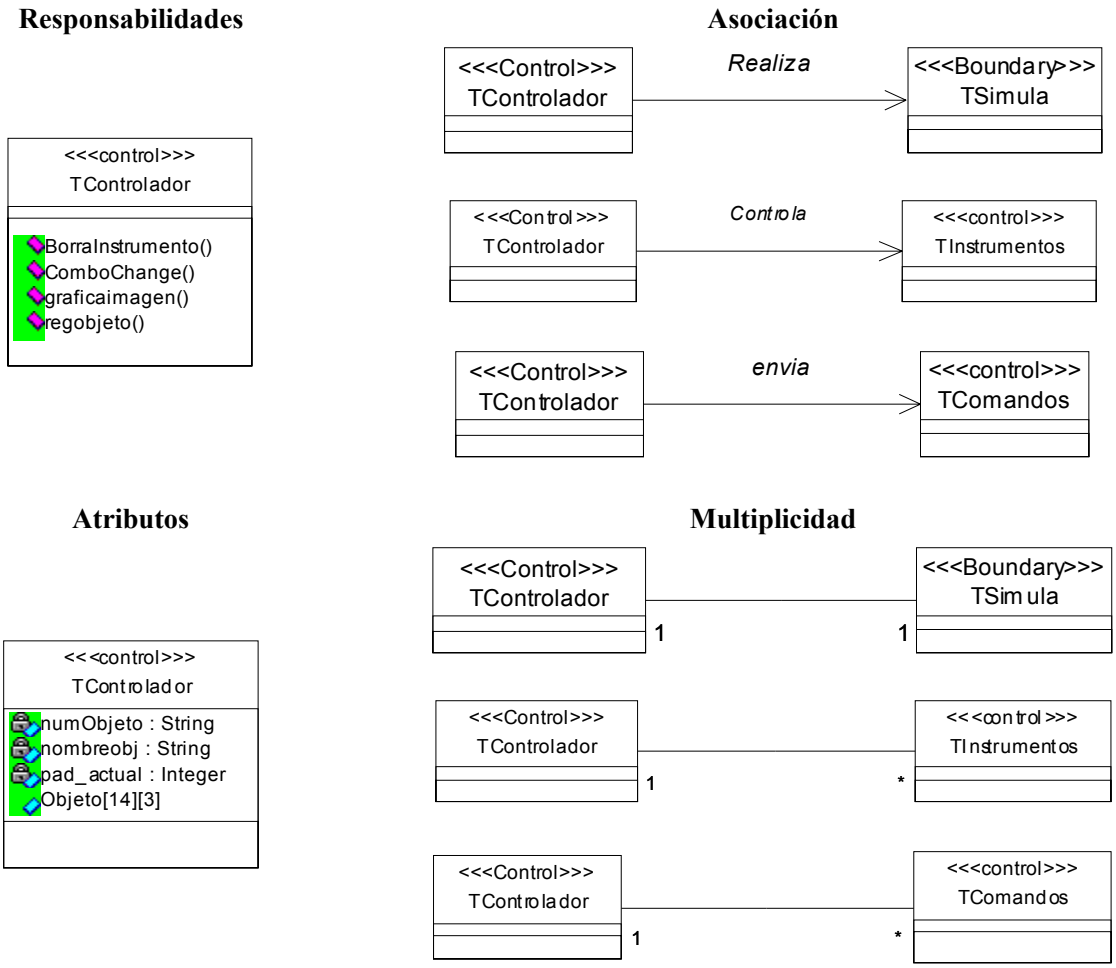


Figura 5.26. Diagrama de diseño de la clase TControlador.

Existe multiplicidad entre las clases TInstrumentos y TControlador, ésta es de muchos a uno porque un instrumento tiene un único controlador y un controlador puede tener muchos instrumentos.

Una asociación se presenta entre las instancias de las clases TInstrumentos y TConfigIns debido a que cada instrumento elige una configuración durante la simulación. Otra asociación existe entre las instancias de clases TControlador e TInstrumentos debido a que un conjunto de instancias de instrumentos es manejado por el controlador exclusivamente y por último se presenta una asociación entre las instancias de las clases TSimula y TInstrumentos, donde cada simulación se compone de instrumentos.

Los atributos encontrados para esta clase son nombre, des, tipo, ByteEstado, pad, talk, num_inst, nObjeto, listener, los cuales representan la información del instrumento seleccionado. Cada elección de instrumentos requiere de atributos especiales para el manejo de los instrumentos, implementar el instrumento y las restricciones manejadas en el bus.

La figura 5.27 ilustra los atributos, responsabilidad en funciones, asociación y multiplicidad de la clase TInstrumentos.

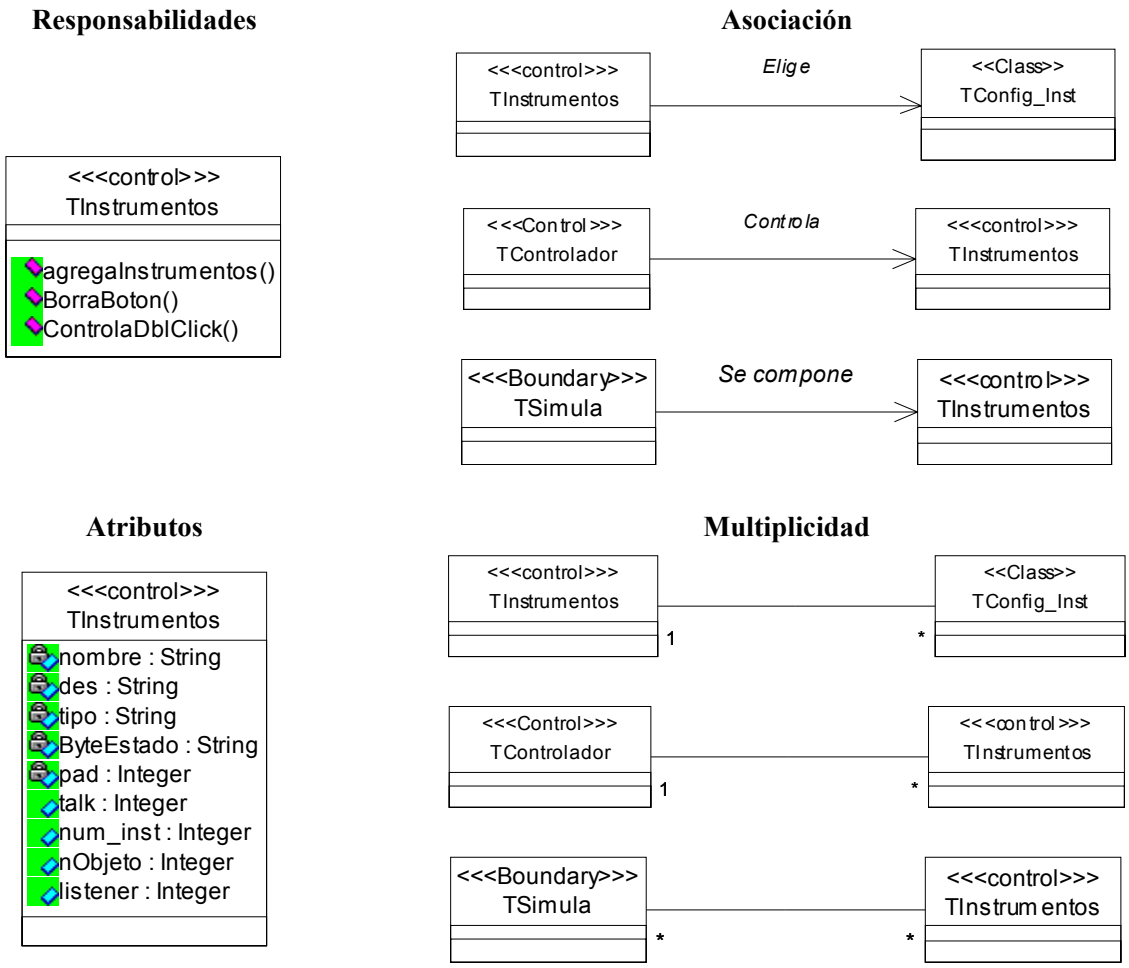


Figura 5.27. Diagrama de diseño de la clase TInstrumentos.

5.5. Implementación del sistema SepiGPIB

El objetivo principal durante la etapa de diseño del sistema fue desarrollar un software que incorpore y simule las características más relevantes del protocolo de instrumentación GPIB. Dentro de las especificaciones básicas del estándar IEEE 488.1 (subcapítulo 4.2) se tienen las especificaciones mecánicas, eléctricas, funcionales y de procedimiento, el estándar IEEE 488.2 incorpora una nueva especificación de procedimiento (subcapítulo 4.3).

Basando la descripción del sistema en las especificaciones mencionadas, se consideraron como los aspectos más importantes del protocolo, la configuración del bus, los tipos de instrumentos, los datos y las órdenes, el direccionamiento, los mecanismos de sondeo y la estructura del bus. A continuación se describe la forma en que el SepiGPIB adopta las especificaciones del protocolo de instrumentación GPIB.

5.5.1. Configuración del bus

Dentro de las especificaciones mecánicas, la configuración implementada es la lineal y se representa a través del enlace entre la clase principal TSimula y la clase TInstrumentos, la interacción de dichas clases permite agregar al bus un máximo de 14 instrumentos de forma lineal (figura 5.28).

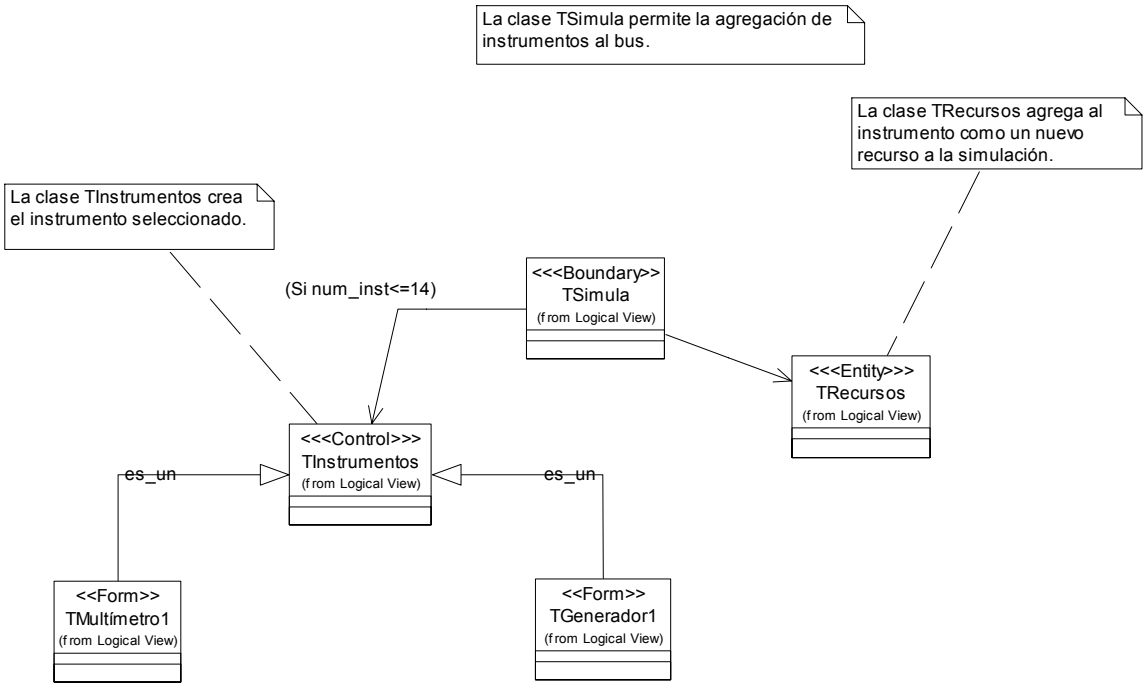


Figura 5.28. Diagrama de clases de la agregación de instrumentos al bus.

5.5.2. Tipo de instrumentos

La clase TSimula permite al sistema manejar dos tipos de instrumentos GPIB, emisor y receptor, dicha clase realiza la llamada a la función de agregación de instrumentos al bus. Al agregar el primer instrumento al bus, inmediatamente se agrega un segundo elemento como controlador, permitiendo con ello manejar los tres tipos de instrumentos incorporados por el estándar IEEE 488.

5.5.3. Datos y órdenes

El GPIB cuenta con un flujo bidireccional de datos y órdenes entre el controlador y los dispositivos conectados al bus, cada dato enviado a través del bus es codificado en formato ASCII asociado a un octeto, dicho octeto de datos guarda una equivalencia respecto a los bits 1 al 7 correspondientes.

Dentro de la clase TComandos se realiza la conversión de los datos a transmitir y su correspondiente decodificación a formato ASCII para ser presentado en el bus, esta función de conversión se emplea al enviar los datos al monitor de bus de la clase TArchivo. La figura 5.29 ilustra la interacción de las clases que interviene en la transmisión, decodificación y presentación de datos en el monitor de bus.

5.5.4. Direccionamiento

Un instrumento agregado a la simulación cuenta con una dirección primaria única definida en el estándar IEEE 488, se permite el uso de 31 direcciones primarias, del 0 al 30 como direcciones válidas, la dirección 31 se reserva para propósitos de control y la dirección 21 se asigna al controlador.

La clase TInstrumentos realiza la asignación de direcciones de forma aleatoria respetando las condiciones establecidas en el estándar IEEE 488.

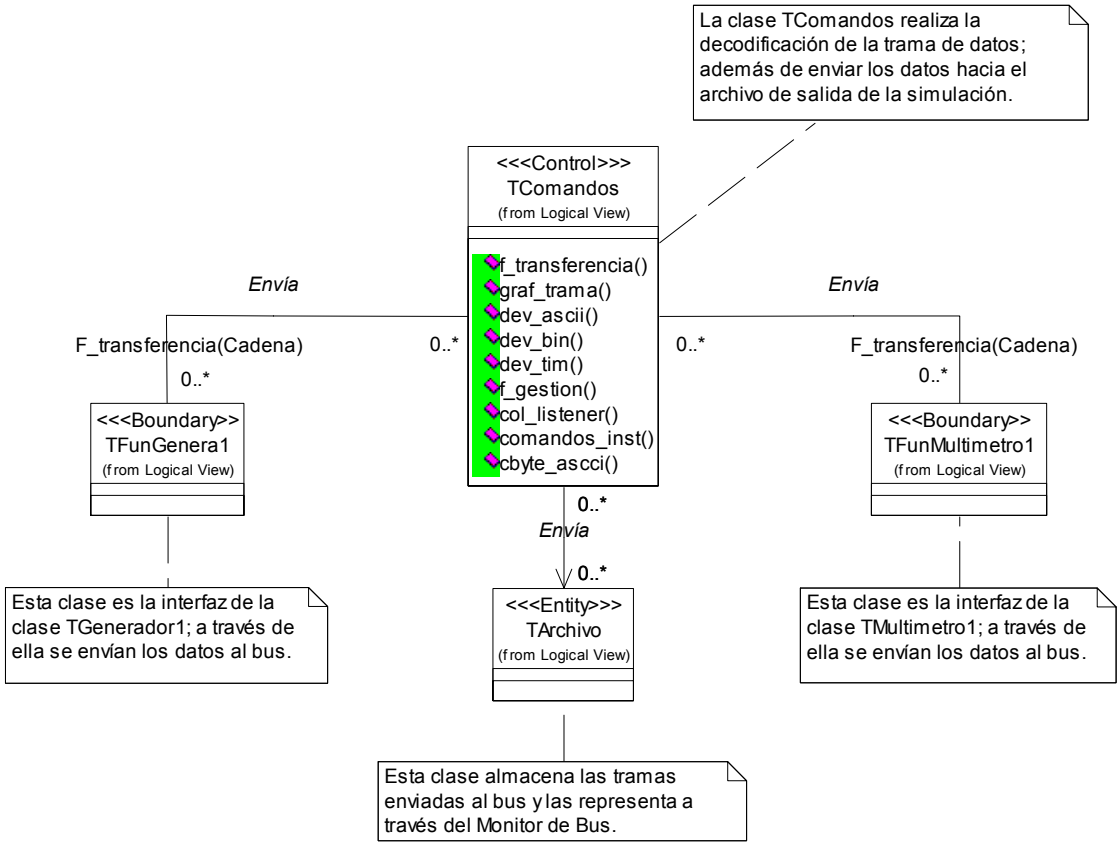


Figura 5.29. Diagrama de clases de la transmisión de datos al monitor de bus.

5.5.5. Mecanismos de sondeo

La clase TSondeo implementa dos tipos de sondeo, sondero serie y sodeo paralelo. El sondeo serie realiza una búsqueda lineal a través de arreglos, es decir, lleva a cabo una búsqueda secuencial de cada uno de los instrumentos activos en el bus, por otra parte el sondeo paralelo realiza una búsqueda mediante llamadas a la función y en cada llamada puede direccionar hasta 8 instrumentos en el bus.

5.5.6. Estructura del bus

La estructura del GPIB está integrada por 16 líneas activas agrupadas en ocho de datos, tres de control de transferencia y cinco de gestión del bus, el bus se implementó dentro de la clase TComandos mediante el uso de dos arreglos de ocho elementos cada uno:

- datos[8]: representa las ocho líneas de datos.
- estado[8]: representa las líneas correspondientes al control de transferencia y de gestión del bus.

Tales arreglos representan el bus y se definieron como miembros públicos debido a que cualquier objeto de la clase TInstrumentos y de la clase TControlador puede accederlos, logrando con ello una simulación del bus en forma de conexión paralela en sus líneas. La figura 5.30 muestra la estructura del bus mediante clases dentro del sistema desarrollado.

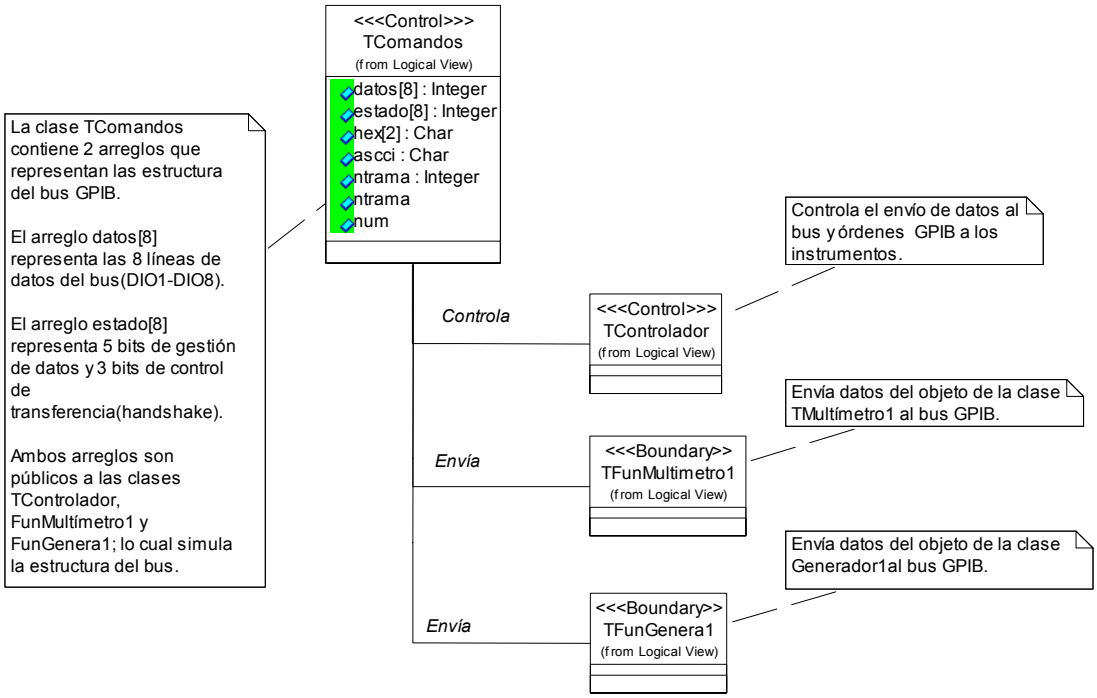


Figura 5.30. Representación de la estructura del GPIB.

5.6. Pruebas al SepiGPIB

Las fases de prueba consistieron en realizar un análisis de los módulos que constituyen la arquitectura del SepiGPIB. Cada uno de los módulos principales que componen el sistema fue sometido a sesiones de pruebas para localizar posibles errores de ejecución o de funcionamiento lógico, para realizar estas pruebas de funcionamiento se utilizaron dos tipos de técnicas:

- *Pruebas de caja blanca*: consistieron en realizar las pruebas de funcionamiento interno de los módulos del sistema y se basan en la depuración línea a línea, por parte del controlador, de la parte funcional de los bloques, tanto de funciones propias como de funciones definidas, con el objetivo de encontrar posibles errores de funcionamiento y comportamiento de los parámetros utilizados al implementar la mayor parte del sistema.
- *Pruebas de caja negra*: consisten en realizar un análisis del funcionamiento de los módulos del sistema enviando parámetros de entrada y observando los resultados, es decir, se analizan únicamente las entradas a los módulos y la respuesta que se tienen a la salida del módulo.

Además de realizar las pruebas anteriores y debido a que es un software desarrollado en forma incremental y a través de un ciclo, estas pruebas de funcionamiento se realizaron repetidas veces a lo largo del desarrollo del software para asegurar su calidad.

Cada una de los errores, encontrados a partir de la realización de las pruebas, fue corregido de forma inmediata gracias a que cada una de las fases se desarrollaron de forma separada.

5.7. Evaluación del SepiGPIB

El software desarrollado fue elaborado en base a ciertos requerimientos definidos por el problema, al ser un software de simulación, el diseño se basó en la existencia de software de instrumentación electrónica. A continuación se listan los diversos tipos de programas empleados en el análisis del sistema:

- *NI Spy*: es una utilidad empleada para el monitoreo del GPIB en tiempo real y permite visualizar el envió y recepción de datos en el bus a partir del monitoreo de las órdenes utilizadas por cada dispositivo en el bus.
- *GPIB Analyzer AT-GPIB/TNT+***:** es un monitor de tramas en tiempo real que cuenta con un visualizador de tramas para analizar cada uno de los datos que se trasmiten en el bus.
- *NI-488*: software cuya función principal es proporcionar los controladores necesarios para el funcionamiento óptimo de los instrumentos GPIB.
- *Borland C++ Buider*: compilador de la firma Borland utilizado para la ejecución de nuestro sistema y base principal del SepiGPIB, además fue de gran ayuda para la prueba de órdenes SCPI.
- *VEE*: es un lenguaje de programación visual de la firma Agilent que basa su funcionamiento en la utilización de una GUI para la programación y manejo de instrumentos en forma directa, entre el instrumento y el computador, o mediante del manejo de instrumentos virtuales.
- *LabView*: es un programa de desarrollo gráfico de la firma National Instruments que ofrece soporte en funciones matemáticas, de control, de manejo de instrumentos en tiempo real y para la programación de instrumentos, soporta el desarrollo de instrumentos virtuales.

Para la realización de SepiGPIB se consideraron características y atributos principales de las herramientas software mencionadas. La evaluación del software consistió en una comparación de los programas utilizados y el SepiGPIB.

Durante el desarrollo del SepiGPIB se realizaron evaluaciones referentes a la funcionalidad, la interfaz, los procedimientos y el comportamiento del software.

Se realizaron pruebas de evaluación con diez ingenieros en electrónica y computación respecto al funcionamiento del SepiGPIB, mismos que aportaron valiosos comentarios para la realización de un software robusto.

6. Descripción de la herramienta SepiGPIB

El SepiGPIB es una herramienta desarrollada a partir de las especificaciones mecánicas, eléctricas, funcionales y de procedimiento del estándar IEEE 488 (subcapítulo 4.2), en base al análisis de dichas especificaciones, éstas se adecuaron al sistema de simulación.

A continuación, se presenta la realización del SepiGPIB como resultado del estudio (capítulo 4) y modelado (capítulo 5) del estándar IEEE 488.

6.1. Instrumentos modulares simulados

El SepiGPIB simula el funcionamiento completo del GPIB mediante la utilización de tres instrumentos de medida, los cuales se listan en la tabla 6.1.

Tabla 6.1. Instrumentos GPIB simulados por el SepiGPIB.

Modelo	Fabricante	Función
33120A	Agilent Technologies	Generador de funciones.
34401A	Agilent Technologies	Multímetro.
E3610A	Agilent Technologies	Fuente de voltaje.

Debido a que los instrumentos de medida utilizados en la simulación presentan características propias de cada modelo y fabricante, se realizó un análisis funcional de los atributos reales de los dispositivos a incorporar. Cada atributo incorporado al sistema presenta un comportamiento en función al tipo de dispositivo manejado.

Al ser un software de simulación, el hecho de agregar un instrumento al bus requiere obtener los datos a través de su panel frontal y configurar las funciones propias de cada dispositivo. El panel frontal es la interfaz con el usuario para realizar una petición de medida o de modificación a los datos almacenados en un instrumento.

El SepiGPIB especifica la forma de agregación de los instrumentos al bus y clasifica a los instrumentos en dos tipos, emisor o receptor, de acuerdo a las características propias de cada instrumento.

A continuación se describe el análisis e implementación de los instrumentos incorporados al SepiGPIB.

Figura 6.1. Generador de funciones modelo 33120A.

6.1.1. Generador de funciones

El generador de funciones modelo 33120A de la firma Agilent (figura 6.1) es un dispositivo que genera señales estables, de gran exactitud y con baja distorsión de salida, soporta la generación de señales senoidales, cuadradas, triangulares, rampa y de usuario.

El panel frontal de este instrumento permite acceder a las funciones del dispositivo para configurar la forma de onda en frecuencia, amplitud (Vp-p, Vrms, dBm) y desplazamiento (*offset*). Soporta modulaciones en AM, FM, con barridos lineales de registros en rangos desde 1 ms a 500 s y permite la comunicación GPIB y RS-232 mediante órdenes SCPI. Las especificaciones funcionales del generador de funciones se presentan en el anexo B.1.

Los atributos del generador de funciones que son simulados por el SepiGPIB son:

- *Tipo de onda*: consiste en la forma de onda de la señal que se genera y puede ser: senoidal, cuadrada, triangular y de rampa.
- *Frecuencia*: es el valor de frecuencia de la señal.
- *Amplitud*: es el valor de amplitud de la señal generada.

6.1.2. Multímetro

El multímetro modelo 34401A de la firma Agilent (figura 6.2) provee una resolución de 6.5 dígitos y una velocidad de hasta 1000 lecturas/seg. para la medición de voltaje de CD y de CA, corriente CD y CA, resistencia, frecuencia y periodo. Soporta la comunicación directa GPIB mediante órdenes en formato ASCII.

Este dispositivo fue diseñado para realizar mediciones de distintos tipos y permite almacenar hasta 512 lecturas en su memoria interna, todos sus atributos y funciones se acceden mediante su panel frontal. Las especificaciones funcionales de este dispositivo se presentan en el anexo B.2.

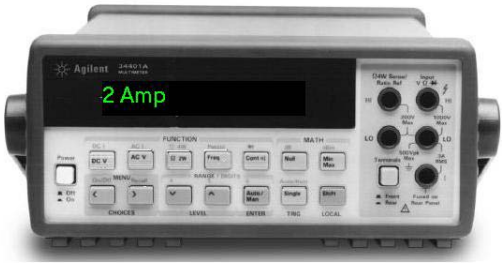


Figura 6.2. Multímetro modelo 34401A.

Figura 6.3. Fuente de voltaje modelo 3610A.

En el SepiGPIB, el multímetro, tiene la función de realizar la medición de variables y se configura a través de su panel frontal. Los de atributos que simula el SepiGPIB son:

- *Voltaje CD*: consiste en la medición del voltaje CD.
- *Voltaje CA*: es el valor de medición de voltaje CA.
- *Resistencia*: para medir la resistencia en ohms.
- *Frecuencia*: parámetro referente a la frecuencia.

6.1.3. Fuente de voltaje

La fuente de voltaje E3610A de la firma Agilent (figura 6.3) suministra voltaje y corriente, simultáneamente, con gran precisión. Cuenta con un rango dual de voltaje constante (CV) y corriente constante (CC), dos tipos de salida con rangos de 8 V - 3 A hasta 15 V - 2 A. Las especificaciones funcionales del dispositivo se presentan en el anexo B.3.

Los atributos de este dispositivo que incorpora el SepiGPIB son:

- *Voltaje*: el valor del voltaje CD.
- *Corriente*: el valor de corriente CD.

6.2. Simulación del SepiGPIB

De acuerdo a las especificaciones del estándar IEEE 488, el SepiGPIB implementa las especificaciones funcionales de los dispositivos a simular. Los resultados obtenidos se describen a continuación.

6.2.1. Especificaciones mecánicas

Las especificaciones mecánicas del GPIB permiten dos tipos de configuraciones (apartado 4.2.1), la herramienta SepiGPIB implementa la configuración lineal que permite realizar operaciones simultáneas en los instrumentos de medida. La figura 6.4 ilustra el tipo de configuración realizada en el sistema.

El SepiGPIB realiza una representación gráfica del GPIB mediante la cual se realiza la interconexión de los instrumentos simulados. El modelado del bus se realiza en el apartado 5.5.1.

El SepiGPIB soporta la simulación de 14 instrumentos activos (inciso 4.2.3.1) y del dispositivo controlador. Las clases realizadas para el manejo de los instrumentos y sus relaciones de asociación, multiplicidad y unificación se describen en el apartado 5.4.1.

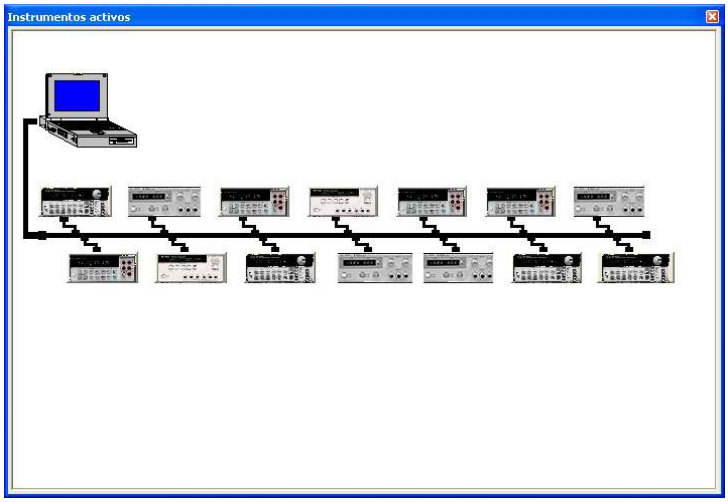


Figura 6.4. Configuración del bus lineal del SepiGPIB.

6.2.2. Especificaciones funcionales

Las especificaciones funcionales definen los tipos de instrumentos en el bus, sus órdenes, datos, direcciones y funciones básicas (apartado 4.2.3). La figura 6.5 muestra cómo el SepiGPIB implementa tres tipos de instrumentos en el bus, emisor, receptor y controlador, esté último representado por una computadora portátil (figura 6.4).

El SepiGPIB implementa las dieciséis líneas del GPIB, ocho líneas de datos, tres de transferencia de datos y cinco de gestión del bus (figura 6.6). Los datos enviados al bus se convierten a un formato hexadecimal y ASCII.

El SepiGPIB asigna, aleatoriamente, una dirección primaria a cada instrumento agregado al bus considerando las restricciones impuestas por el estándar IEEE 488 (inciso 4.2.3.2). La ventana de configuración de instrumentos permite visualizar la dirección primaria de un dispositivo activo en el bus (figura 6.7).

Cada simulación, en el SepiGPIB, debe configurarse correctamente para que el sistema realice la función de transferencia y pueda controlar, enviar y recibir información en el bus. El sistema cuenta con filtros de tramas, que pueden ser configurados por el usuario mediante las opciones presentadas en la sección Tipo de Captura ubicada dentro de la ventana Configuración de simulación, este filtro de tramas permite determinar el tipo salida que se desea como resultado de la simulación (figura 6.8). Por defecto, el SepiGPIB realiza la configuración con valores predeterminados por el sistema.

Cada tipo de captura presenta un resultado distinto en la realización de una simulación, el tipo de captura, el número de eventos a capturar, el tipo de sondeo y la especificación del disparo se pueden observar en la figura 6.9.

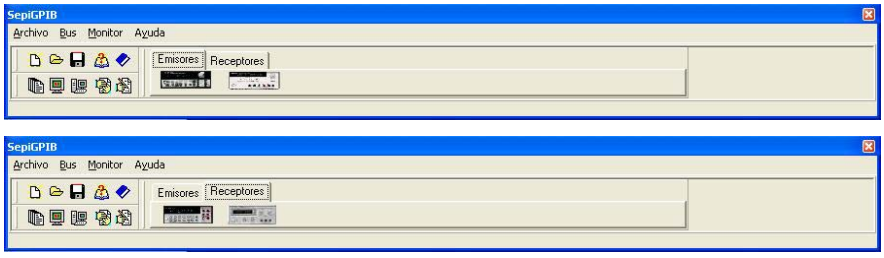


Figura 6.5. Tipos de instrumentos implementados en el SepiGPIB.

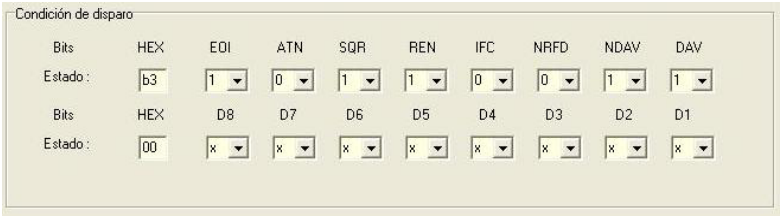


Figura 6.6. Señales del bus manejadas dentro del sistema.

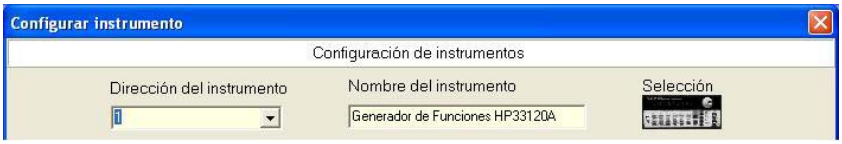


Figura 6.7. Ventana Configuración de instrumentos.

6.2.3. Función de transferencia

El SepiGPIB implementa la función de transferencia (inciso 4.2.4.1) para la comunicación entre los dispositivo agregados al bus, esta función simula el comportamiento del GPIB en el envió y recepción de datos mediante el control de las tramas que son puestas en las líneas del bus.

La función de transferencia analiza cada uno de los datos en base a la activación de las señales en el bus, una vez activada la comunicación entre los dispositivos, la función de transferencia almacena cada uno de los datos enviados al bus en un archivo de tramas, que puede visualizarse durante la simulación. En la figura 6.10 se muestra la respuesta de la función de transferencia al envió de un octeto de datos al bus.

El SepiGPIB implementa dos tipos de sondeo, serial (párrafo 4.2.4.2.1) y paralelo (apartado 4.3.6), los cuales son activados en la configuración inicial de la simulación.

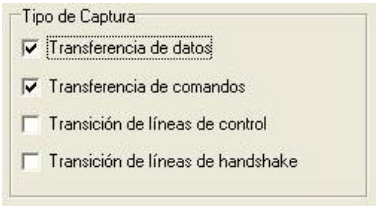


Figura 6.8. Tipos de captura de la simulación del SepiGPIB.

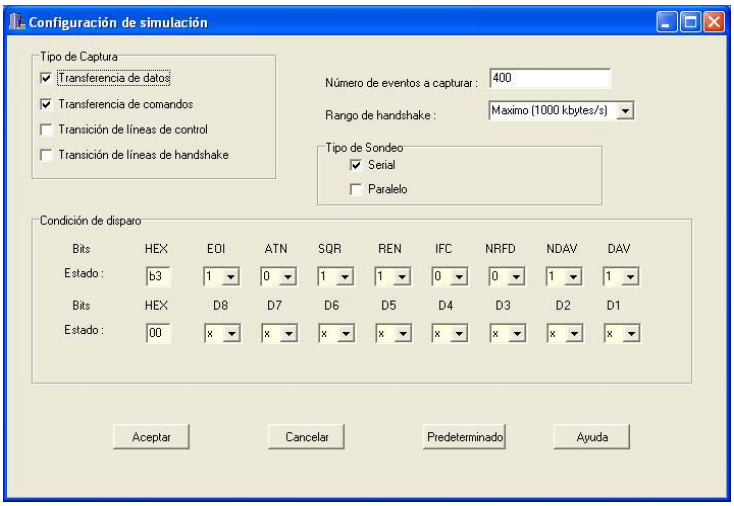


Figura 6.9. Ventana Configuración de simulación del SepiGPIB.

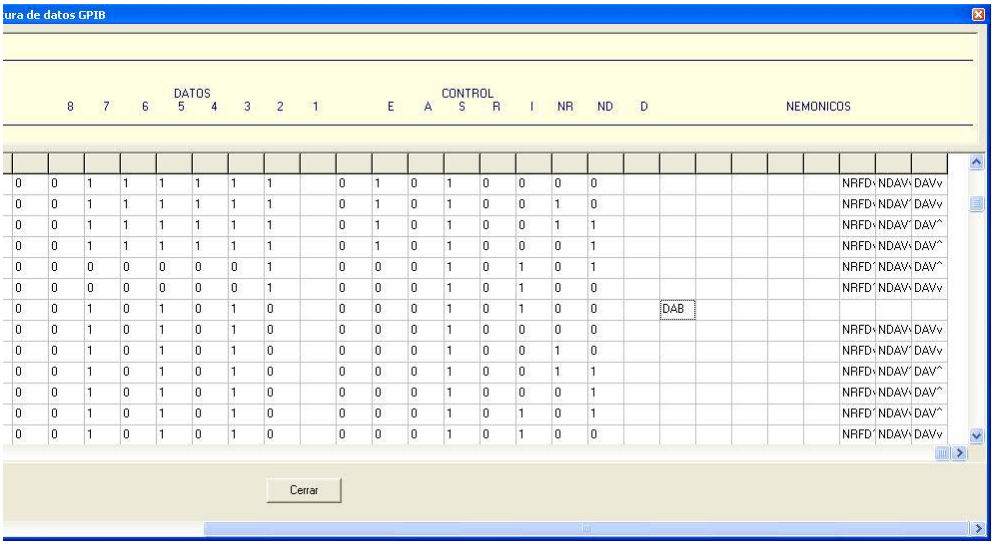


Figura 6.10. Activación de señales de la función de transferencia.

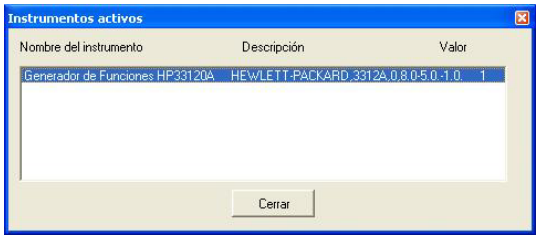


Figura 6.11. Ventana de Instrumentos activos en el bus.

Una vez configurado el tipo de sondeo, el modulo correspondiente almacena el nombre, descripción y dirección primaria del instrumento. La figura 6.11 ilustra la ventana de instrumentos activos una vez realizado un sondeo.

6.2.4. Simulación de los instrumentos

Como se menciona en el inciso 4.2.4.3, el GPIB cuenta con un flujo bidireccional de datos y órdenes entre el controlador y los dispositivos interconectados al bus, esta relación se presenta al activar la ventana del controlador del GPIB desde donde se realizan las peticiones a los instrumentos de forma directa o mediante el envío de órdenes.

El SepiGPIB cuenta con un controlador que realiza peticiones en forma directa a los instrumentos, el controlador incorpora un grupo de funciones y órdenes específicas para cada tipo de instrumento.

Un dispositivo activo puede ser manipulado por distintos módulos del sistema para realizar una agregación, configuración o eliminación, sólo en el caso de monitorear las peticiones de medición al instrumento se hace uso de la aplicación para la comunicación IEEE 488, la cual tiene el objetivo de enviar cada una de las peticiones u órdenes a un monitor de bus para la presentación, en forma ordenada, de la decodificación de las tramas.

La figura 6.12 presenta la ventana del controlador del GPIB al manejar un instrumento generador de funciones. El SepiGPIB proporciona dos alternativa de comunicación con un instrumento generador de funciones modelo 33120A, mediante la utilización del controlador del GPIB, que hace uso de la dirección primaria del instrumento para direccionar cada petición al dispositivo correspondiente, o mediante el panel frontal del instrumento, que permite al usuario realizar peticiones o configurar el instrumento.

Figura 6.12. Ventana del controlador del GPIB para el generador de funciones.

7. Conclusiones y líneas futuras de investigación

La presente tesis consistió en un trabajo de investigación en dos campos del conocimiento muy importantes en la actualidad, la tecnología de orientación a objetos y la instrumentación electrónica, con la finalidad de desarrollar una herramienta software para el estudio y análisis del protocolo IEEE 488 con fines académicos.

Este trabajo inicia la investigación sobre la aplicación de técnicas de modelado para el diseño de herramientas con fines académicos para el conocimiento de protocolos de comunicaciones industriales, de instrumentación y de redes de computadoras. Las herramientas utilizadas han sido RUP, de la firma Rational Software Corporation, como metodología de desarrollo y UML para el modelado del software, ambas herramientas fueron elegidas debido a su creciente utilización e importancia en el campo de la ingeniería del software y la tecnología orientada a objetos. La herramienta de implementación fue C++ Builder en su versión 6, de la firma Borland, dada su gran utilización en la implementación de aplicaciones de usuario y el soporte que obtiene del lenguaje de programación C++.

El protocolo de comunicaciones elegido para su estudio ha sido GPIB. Se ha elegido este protocolo debido a que cuenta con una estandarización, IEEE 488, y se puede realizar un estudio exhaustivo de su funcionalidad analizando el protocolo de comunicaciones que implementa. Otro motivo para la elección de este bus de instrumentación es que existe una gran cantidad de dispositivos y equipamiento que lo implementan, además de ser uno de los sistemas más usados en la industria actual. Cabe destacar también la existencia del Laboratorio de Comunicaciones Digitales de la UTM que basa su funcionamiento en el protocolo GPIB, dando con ello pauta a pruebas y ensayos de comunicaciones e instrumentación virtual que refuerzan los planteamientos del presente trabajo de tesis.

El desarrollo del trabajo de investigación requirió de trabajos complementarios, como resultados de la investigación pueden mencionarse las siguientes aportaciones:

- Se realizó un estudio de la instrumentación electrónica programable, imprescindible para obtener los conocimientos sobre instrumentación programable e instrumentación virtual, resultado de ello fue el estado del arte que se presenta en el capítulo 3.
- Se estudió el estándar IEEE 488 en sus dos versiones IEEE 488.1 e IEEE 488.2. El estudio del protocolo IEEE 488.1 se basó en sus especificaciones mecánicas, eléctricas, funcionales y de procedimiento, mientras que respecto al protocolo IEEE 488.2 el estudio consistió en el análisis de los mensajes, órdenes comunes, reportes de estado y del protocolo para el controlador. Resultado de tal estudio es el capítulo 4.
- Por otro lado, se realizó un estudio sobre la tecnología orientada a objetos y sobre la historia de las metodologías orientadas a objetos, como resultado se obtuvo el estado del

arte que se presenta en el capítulo 2.

- Resultado de un análisis de las metodologías, lenguajes, técnicas y herramientas para el desarrollo de software orientado a objetos, se realizo un estudio del UML (subcapítulo 2.3), del RUP (subcapítulo 2.4) y del lenguaje de programación C++Builder.
- En base a los conocimientos adquiridos del estándar IEEE 488, del paradigma orientado a objetos y de las herramientas de metodología y de modelado, se realizó el modelado en UML del SepiGPIB siguiendo la metodología propuesta por el RUP. El capítulo 5 describe el proceso de modelado del SepiGPIB.
- Una vez obtenido un entorno de simulación del estándar IEEE 488, basado en el programa ejecutable del SepiGPIB, se pudo realizar el análisis de las señales en el bus y de las primitivas de servicio que se intercambian entre instrumentos conectados al GPIB. Se realizaron pruebas de verificación al programa resultante para proporcionar una herramienta de apoyo al aprendizaje del protocolo GPIB. El capítulo 6 y el Anexo A se encargan de describir el funcionamiento del SepiGPIB y mostrar una herramienta de aprendizaje del mismo software respectivamente.

La valoración final del trabajo realizado en esta tesis es positiva, la experiencia personal adquirida en la utilización del RUP y del UML, como herramientas de modelado de sistemas de software, ha sido muy satisfactoria dado que cuentan con un gran soporte, son sencillas de utilizar y facilitan el modelado general del problema a resolver. Además de que la utilización de dichas herramientas facilita la comprensión del modelado y de la herramienta final.

El trabajo de investigación de esta tesis forma parte de la línea de investigación del Cuerpo Académico de Redes de Instrumentación del Instituto de Electrónica y Computación de la Universidad Tecnológica de la Mixteca, dedicado al estudio de la adecuación del uso de técnicas de descripción formal en el desarrollo de sistemas de instrumentación programable y estudio de los protocolos de comunicaciones industriales y de instrumentación. Los resultados obtenidos refuerzan algunos trabajos en desarrollo y plantean otros futuros en la misma línea de investigación:

- Se encuentra en proceso el desarrollo de una herramienta hardware, para convertir el protocolo IEEE 488 a USB, cuyo objetivo es adaptarse al SepiGPIB para realizar un monitoreo de las señales en el bus en tiempo real. Cabe destacar que el SepiGPIB ya ofrece esta opción pero por ahora se encuentra inhabilitada (inciso 5.3.2.4).
- Otro trabajo de investigación se basa en gestionar un laboratorio virtual basado en el protocolo GPIB, la gestión incluye la configuración de las herramientas hardware existentes en el Laboratorio de Comunicaciones Digitales de la Universidad Tecnológica de la Mixteca, la gestión de las herramientas software para programación y desarrollo de aplicaciones de propósito general, y el diseño de prácticas de instrumentación programable y virtual a distancia.
- Se encuentran en investigación proyectos para la creación de controladores y herramientas de monitoreo del protocolo GPIB en un ambiente Linux y para el desarrollo de interfaces inalámbricas para conectar dispositivos GPIB. También se propone la especificación del protocolo GPIB mediante un lenguaje de descripción de hardware (HDL, *Hardware Description Language*).
- En base al modelado del SepiGPIB se pretende ampliar el entorno mediante la incorporación de más dispositivos al bus.
- En el campo de la tecnología orientada a objetos, se encuentra en desarrollo un trabajo de investigación destinado a modelar el protocolo de comunicaciones DMX512-A orientado al área de entretenimiento.

Bibliografía

[1] Agilent technologies: Test & Measurement; Catalog 2001. Agilent technologies Corp., 2001.

[2] Aksit, M. & Bergmans, L.: "Obstacles in Object-Oriented Software Development". Technical report of the Faculty of Computer Science, University of Twente, The Netherlands.

[3] ANSI/IEEE Std 488-1978:: Standard Digital Interface for Programmable Instrumentation. Recommendes Practice for Code and Format Conventions for Use with ANSI/IEEE Std 488-1978. ANSI/IEEE Std 488-1978. IEEE Std 728-1982. The Institute of Electrical and Electronics Engineers, 1983.

[4] Armour, F. & Miller, G.: Avanced Use Case Modeling. Addison Wesley Longman, 2001.

[5] Bertrand, M.: Construcción de Software Orientado a Objetos. Prentice Hall, Segunda edición, 1999.

[6] Beringer, D.: "Modelling Global Behaviour in Object-Oriented Analysis: Scenarios, Use Cases and Interaction Diagrams". Technical report of the Laboratoire du Génie Logiciel, Lausanne, 1996.

[7] Booch, G.: Análisis y Diseño Orientado a Objetos con Aplicaciones. Addison Wesley Longman, Segunda edición, 1996.

[8] Booch G., Rumbaugh, J. & Jacobson I.: The Unified Modeling Language; User Guide. Addison Wesley Longman, 1998.

[9] Caristi, A.: IEEE-488 General Purpose Instrumentation Bus Manual. Academic Press, 1989.

[10] Charte, F.: Programación con C++ Builder 5. Anaya Multimedia, 2000.

[11] Cockborn, A: Writing Effective Use Case. Addison Wesley Longman, 2001.

[12] Helfrick, A. y Cooper, W.: Instrumentación Electrónica Moderna y Técnicas de Medición. Prentice Hall, 1991.

[13] Engels, G. & Groenewegen, L.: "Object-Oriented Modeling: A Roadmap". Technical report of the Dept. of Computer Science, Paderborn, Germany and Leiden University, The Netherlands.

[14] Eriksson, H. & Penker, M.: UML Toolkit. John Wiley & Sons, Inc., 1998.

[15] Fowler, M. y Kendall, S.: UML gota a gota. Addison Wesley Longman, 1999.

[16] Garlan, D. & Shaw, M.: "An Introduction to Software Architecture". Advances in Software Engineering and Knowledge Engineering, Volumen I, edited by V. Ambriola and G. Tortora, Publishing Company, New Jersey, 1993.

[17] Graf, R.: Diccionario de electrónica. Pirámide, 1984.

[18] Helsel, R.: Visual Programming with HP VEE. Hewlett Packard Professional Books, third edition, 1998.

[19] Hewlett Packard: HP 34401A Multimeter; User´s Guide. Hewlett Packard Company, 1996.

[20] Hewlett Packard: HP 34401A Multimeter; Service Guide. Hewlett Packard Company, 1996.

[21] Hewlett Packard: HP 33120A Function Generator/Arbitrary Waveform Generator; User´s Guide. Hewlett Packard Company, 1997.
[22] Hewlett Packard: HP 33120A Function Generator/Arbitrary Waveform Generator; Service Guide. Hewlett Packard Company, 1997.
[23] Jacobson, I.: Object Oriented Software Engineering; A use case driven approach. Addison Wesley Longman, 1992.
[24] Jacobson, J., Booch, G. & Rumbaugh, J.: The Unified Software Development Proccess. Addison Wesley Longman, 1999.
[25] Joyanes, L.: Programación Orientada a Objetos. McGraw-Hill, Segunda edición, 1998.
[26] Kruchten, P.: The Rational Unified Process; An Introduction. Addison Wesley Longman, 2000.
[27] Mandado, E. Mariño, P. y Lago, A.: Instrumentación Electrónica. Marcombo, 1995.
[28] Mariño, P.: Las comunicaciones en la Empresa; Normas, redes y servicios. RA-MA Editorial, 2ª edición, 2003.
[29] Mariño, P., Nogueira, J. y Hernández, H.: "Programmable Instrumentation Laboratory for Testing of Electronic Circuits and GPIB's Signal Analysis". Proceedings of the IASTED International Conference on Computers and Advanced Technology in Education, CATE'99, pp. 167-171, Philadelphia, Pennsylvania (USA), May 6-8, 1999.
[30] Mariño, P., Nogueira, J. y Hernández, H.: "Training on Programmable Instrumentation for a curriculum of Electronic Engineering". Proceedings of the 2nd International Conference in Recent Advances in Mechatronics. ICRAM'99. Estambul (Turquía). May 24-26, 1999.
[31] Mariño, P., Nogueira, J. & Hernández, H.: "Electronics Laboratory Practices based on Virtual Instrumentation". Proceedings of the FIE´99, ASEE/IEEE, vol. 12, pp. 6-10, San Juan (Puerto Rico), November 10-13, 1999.
[32] Mariño, P., Nogueira, J. &. Hernández, H.: "Laboratory of Virtual Instrumentation for Industrial Electronics". Proceedings of the IEEE International Conference on Industrial Technology, ICIT'2000, Goa (India), January 19-22, 2000.
[33] Mariño, P., Nogueira, J. y Hernández, H.: "Laboratorio de Instrumentación Programable para prueba de circuitos electrónicos y análisis de señales de bus GPIB". Revista TEMAS, Universidad Tecnológica de la Mixteca, Vol. 4, pp. 29-36, 2000, México.
[34] National Instruments: NI-488.2™ GPIB Analyzer User Manual. National Instruments Corp., 1999.
[35] National Instruments: NI-488.2™ User Manual for Windows. National Instruments Corp., 1999.
[36] National Instruments: The Measurement and Automation Catalog 2003. National Instruments Corp., 2003.
[37] Pressman, R.: Ingeniería del Software; Un enfoque práctico. McGraw-Hill, Cuarta edición, 1998.
[38] Reisdorph, K.: Aprendiendo Borland C++ Builder 3 en 21 Días. Prentice Hall, 1999.
[39] Rumbaugh, J. (et. al.): Modelado y diseño orientado a objetos. Prentice Hall, 1996.
[40] Rumbaugh, J., Jacobson, I. y Booch, G.: El Lenguaje Unificado de Modelado. Manual de Referencia. Addison Wesley Longman, 2000.
[41] Schuller, J.: Aprendiendo UML en 24 horas. Pearson Educación, 2000.

Sitios de Internet

[URL1] http://www.ni.com "Página de la firma National Instruments; proporciona una amplia información de los productos, servicios y soluciones para aplicaciones basadas en instrumentación programable".

[URL2] http://www.rational.com "Página destinada a difundir la información respecto a las nuevas tecnologías basadas en el RUP".

[URL3] http://www.utm.mx/~caff/prin/principal.html "Página del M.C. Carlos Alberto Fernández y Fernández, Profesor-Investigador del Instituto de Electrónica y Computación de la Universidad Tecnológica de la Mixteca".

[URL4] http://www.vico.org "Página dedicada a las metodologías e innovaciones tecnológicas de la OO."

A. Manual del SepiGPIB

El SepiGPIB es un software cuyo objetivo principal es servir como herramienta educativa y de apoyo a usuarios interesados en aspectos referentes al protocolo de instrumentación GPIB. A continuación se presenta un manual que describe los aspectos esenciales para el funcionamiento del SepiGPIB mediante la siguiente estructura:

- Instalación del SepiGPIB.
- Inicialización del programa.
- Descripción del sistema principal
- Descripción de los módulos del sistema.

A.1. Instalación del sistema

Al introducir el disco compacto, que contiene el programa, se ejecutará una aplicación para la instalación del SepiGPIB y automáticamente se presenta una serie de cuadros de diálogos que guiarán paso a paso la instalación del sistema (figura A.1).

Cada uno de los cuadros de diálogo tiene el propósito de capturar información para configurar los parámetros de la instalación. Los cuadros de diálogo que aparecen durante la instalación son:

- Cuadro de diálogo referente a la información del usuario.
- Representación del copiado de información del CD al disco duro.
- Cuadro de diálogo para indicar que la instalación se ha completado satisfactoriamente (figura A.2).

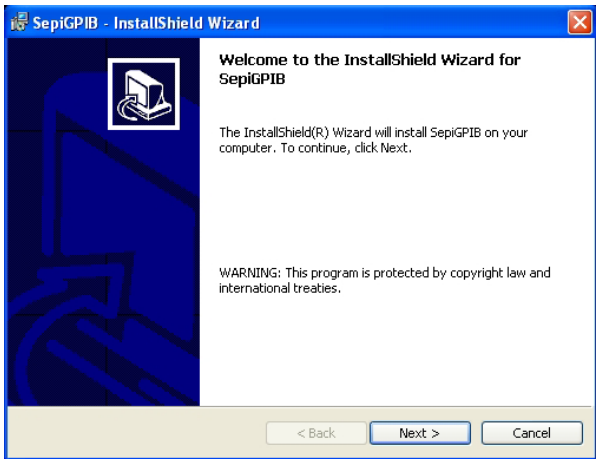


Figura A.1. Cuatro de diálogo de bienvenida.

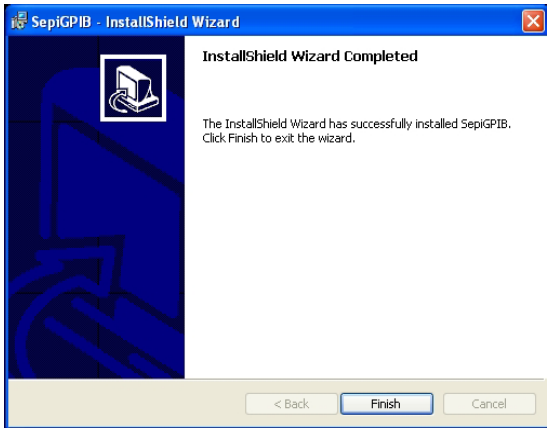


Figura A.2. Cuadro de diálogo que indica la instalación completa del SepiGPIB.

En caso de que el usuario desee abandonar la instalación debe elegir la opción **Cancel** que parece en cada uno de los cuadros de diálogo.

A.2. Inicialización del programa

Para ejecutar la aplicación SepiGPIB es necesario seguir la siguiente ruta (ver figura A.3).

- **Inicio → Todos los programas → SepiGPIB → Espia1 (Aceptar)**.

Dicha ruta puede presentar algunas variaciones dependiendo del sistema operativo de la computadora.

A.3. Descripción del sistema principal

Al inicializar la aplicación aparecen en la pantalla las tres ventanas principales que conforman el software SepiGPIB (figura A.4), **Ventana principal SepiGPIB**, ventana de **Instrumentos activos** y ventana de **Recursos GPIB**, las cuales se describen en los incisos siguientes.

A.3.1. Ventana principal SepiGPIB

Sirve como manejador principal de la aplicación y bajo su control se ejecutan, visualizan y almacenan todas las operaciones que se realizan dentro de la simulación. Se divide en tres secciones (figura A.5):

- *Barra de titulo*: contiene el nombre de la aplicación activa SepiGPIB y, en la parte superior derecha, el botón para cerrar la aplicación.
- *Barra de menús*: proporciona el acceso a menús desplegables que dan acceso a las aplicaciones más importantes del sistema. Mediante un clic en cada una de las opciones se despliegan los respectivos contenidos. Esta barra da acceso a todas las operaciones del sistema. Los menús desplegables de la barra contienen un único elemento denominado comando inmediato que se ejecuta al hacer clic sobre él.

Figura A.3. Ruta de acceso a la aplicación SepiGPIB en el sistema operativo Windows XP.

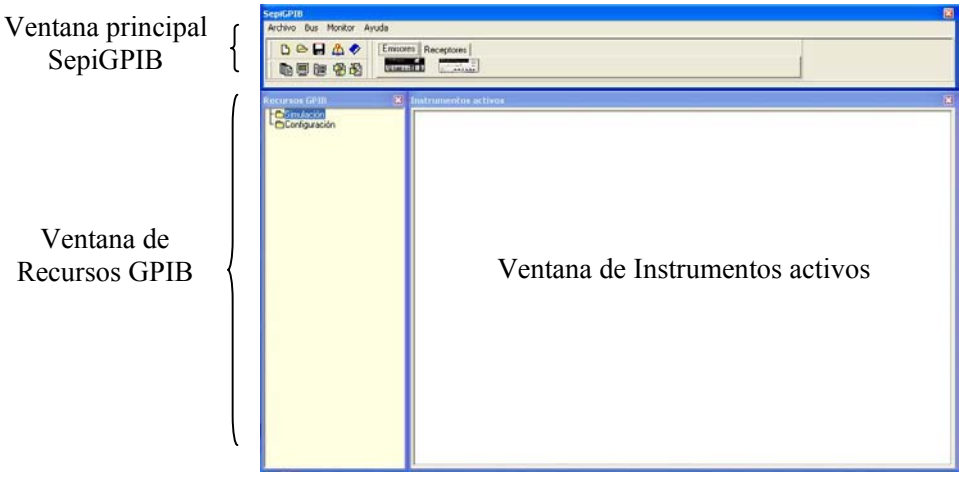


Figura A.4. Sistema SepiGPIB integrado.

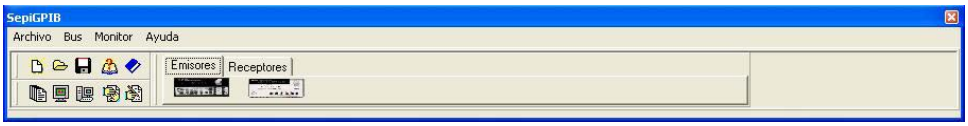


Figura A.5. Menú principal SepiGPIB.

- *Barras de herramientas*: en ellas se dispone de algunos iconos (botones) que permiten realizar las operaciones más frecuentes, sin tener que abrir los menús desplegables, y acceder a los instrumentos electrónicos correspondientes (emisor/receptor).

Se pueden distinguir cuatro divisiones, **Archivo**, **Bus**, **Monitor** y **Ayuda**, la activación de un menú se realiza mediante un clic sobre la opción correspondiente (figura A.6).

La tabla A.1 lista la función de cada botón de la barra de iconos.

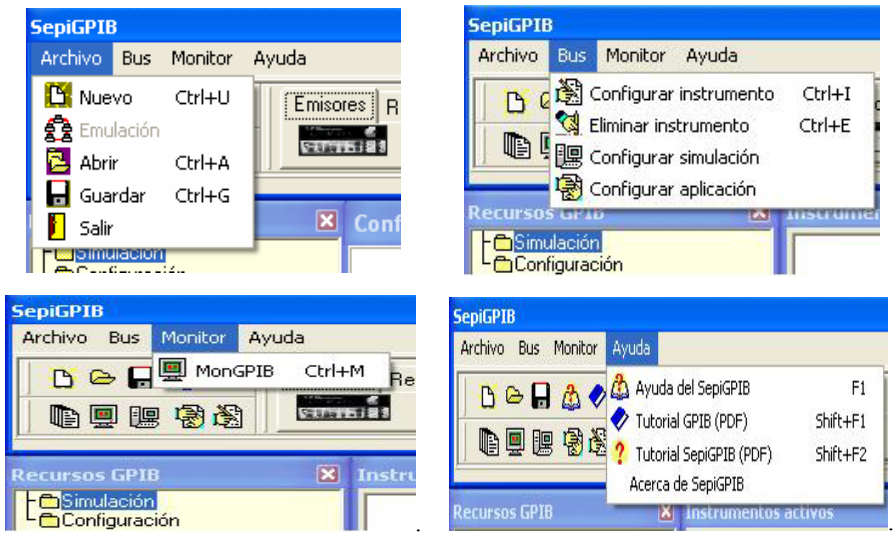


Figura A.6. Activación de los menús del sistema Archivo, Bus, Monitor y Ayuda.

Tabla A.1. Listado de funciones de la barra de iconos.

Icono	Función
	Nueva simulación.
	Abrir un archivo capturado anteriormente.
	Almacenar la captura de datos existente dentro de la simulación.
	Activar la opción de ayuda del sistema.
	Acceder a información del bus GPIB mediante un archivo PDF.
	Activar las ventanas principales del sistema.
	Acceder al módulo de configuración de la simulación.
	Acceder al monitor de bus.
	Acceder al módulo de configuración del controlador.
	Acceder al módulo de configuración del instrumento.

A.3.2. Ventana de Instrumentos activos

Tiene la función de representar gráficamente a los instrumentos que son agregados al bus (figura A.7), es la interfaz entre los instrumentos activos en el bus y el usuario, el cual mediante un clic sobre el instrumento puede acceder al panel frontal.

Al agregar el primer instrumento al bus, automáticamente se agrega el controlador cuya representación gráfica es la de una computadora.

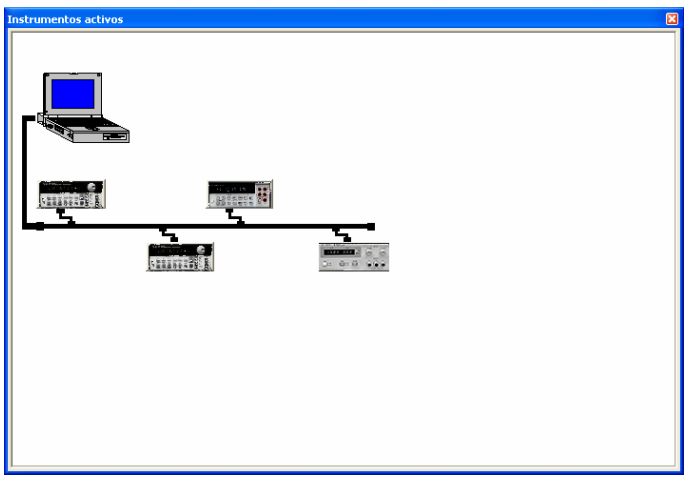


Figura A.7. Ventana de Instrumentos activos en el bus.

Figura A.8. Representación de la ventana de Recursos GPIB de la simulación.

A.3.3. Ventana de Recursos GPIB

Su función principal es proporcionar información al usuario acerca de los recursos disponibles dentro de la simulación. Esta ventana se encuentra dividida en dos secciones (figura A.8):

- *Instrumentos*: en esta sección aparece cada uno de los instrumentos activos en el bus, los cuales han sido agregados durante la simulación y son clasificados de acuerdo a sus características de emisor o receptor.
- *Configuración*: esta sección presenta la información referente a la activación y configuración de los parámetros de la simulación correspondientes a la transferencia de datos, transferencia de órdenes, líneas de comunicación (*handshake*), número de eventos, rango de transferencia, sondeo serie, sondeo paralelo y configuración del disparo de eventos.

Una simulación se inicia con una configuración predeterminada por el sistema, donde cada uno de los parámetros tiene un valor preasignado que se actualiza automáticamente al agregarse un instrumento al bus o reconfigurar la simulación.

A.4. Descripción de los módulos del sistema

A.4.1. Configurar simulación

Uno de los aspectos más importantes al realizar una simulación es la configuración de los parámetros ya que de ello dependerá el resultado final de la simulación.

Existen dos formas de acceder a la ventana **Configuración de simulación**, mediante la activación de la opción **Configurar simulación** ubicada en el menú **Bus** (figura A.6) y a través del acceso directo al icono  ubicado en la barra de iconos de la **Ventana principal SepiGPIB**.

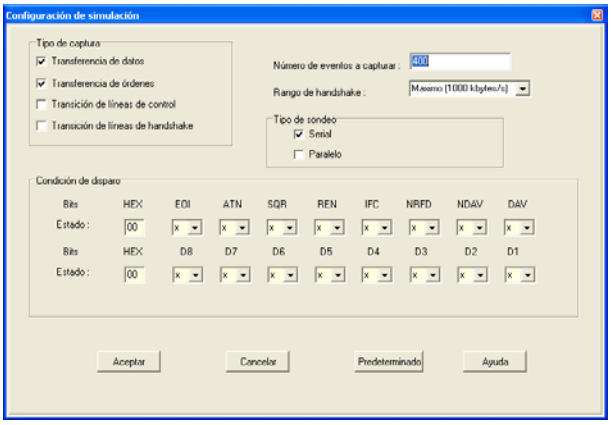


Figura A.9. Ventana Configuración de simulación.

La ventana **Configuración de simulación** consta de cuatro secciones (figura A.9):

- *Tipo de captura*: permite definir los distintos tipos de captura de datos en una simulación, cada activación o desactivación representa un filtro para el tipo de captura existente:
  - Transferencia de datos.
  - Transferencia de órdenes.
  - Transferencia de líneas de control.
  - Transferencia de líneas de función de transferencia.
- En la siguiente sección se configura el **Número de eventos a capturar** y el **Rango** de la función de transferencia entre instrumentos.
- *Tipo de sondeo*: se encarga de configurar el sondeo de los instrumentos dentro de la simulación y puede ser serie o paralelo.
- *Condición de disparo*: configura la activación de los instrumentos mediante dos octetos, el primero representa los bits del octeto de estado y el segundo los bits del octeto de datos. El monitor de bus considera estos valores, en el registro de condición de disparo, para filtrar los datos en la simulación

La opción **Predeterminado** restaura los valores iniciales del sistema, una simulación que no registre parámetros de configuración adoptará tales valores.

A.4.2. Agregar Instrumentos

Se agrega un instrumento al bus mediante la **Barra de herramientas** de la **Ventana principal SepiGPIB** y se selecciona el tipo de instrumento emisor o receptor. La figura A.10 representa los dos accesos a los menús de instrumentos **Emisores y Receptores**.

Cada uno de los instrumentos agregados a la simulación incorpora un grupo de funciones básicas para su manejo y es considerado por el sistema como un objeto independiente, es decir, que cada instrumento puede presentar una configuración específica para realizar la simulación.

Figura A.10. Barra de herramientas de instrumentos.

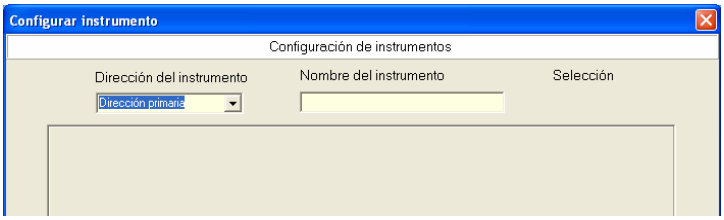


Figura A.11. Ventana Configurar instrumento.

A.4.3. Configurar instrumento

Un instrumento presenta un conjunto de funciones predeterminadas por el sistema que pueden modificarse de dos formas, mediante la opción **Configurar instrumento** ubicada en el menú **Bus** (figura A.6) o proporcionando la dirección primaria del instrumento que se desea configurar (figura A.11). Una vez agregado el instrumento al bus, se selecciona el icono del instrumento a configurar mediante un clic y se realiza la configuración.

Ambas opciones conducen a la visualización gráfica del instrumento seleccionado para ser configurado, una vez activado el instrumento se sobrepone a la ventana de instrumentos activos y la única forma de regresar a la simulación es dar doble clic sobre el instrumento.

A.4.4. Eliminar instrumento

Durante la simulación es posible eliminar un instrumento del bus mediante la activación de la opción **Eliminar instrumento** ubicada en el menú **Bus** (figura A.6).

Una vez que aparece la ventana de la figura A.12 se debe elegir la dirección primaria del instrumento que se desea eliminar. Al proporcionar la dirección del instrumento a eliminar, el sistema lo busca en sus registros y una vez localizado pide la confirmación al usuario.

Al eliminar un instrumento, automáticamente se actualiza la **Ventana de recursos GPIB** y los registros del sistema correspondientes.

A.4.5. Ventanas activas

Durante la simulación se activan distintas ventanas para realizar funciones básicas y debido a que el programa basa su funcionamiento en una interfaz windows, la ventana activa se sobrepone a las demás ventanas, permitiendo visualizar las ventanas que estén ocultas o que se encuentren cerradas. Para regresar las ventanas a su estado original se utiliza el **Administrador de ventanas activas**, el cual se puede activar mediante el icono  ubicado dentro de la barra de iconos de la **Ventana principal del SepiGPIB**. La figura A.13 muestra el **Administrador de ventanas activas**.

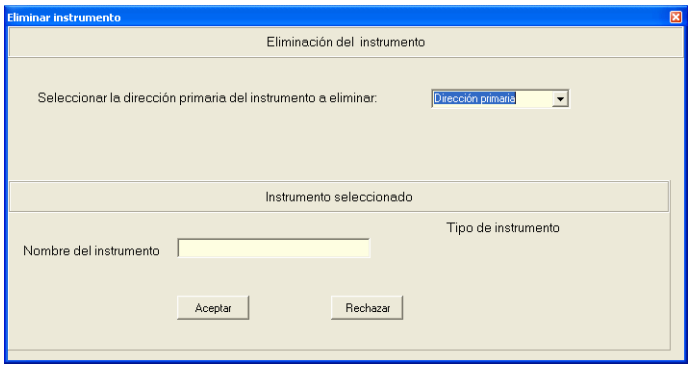


Figura A.12. Ventana Eliminar instrumento.

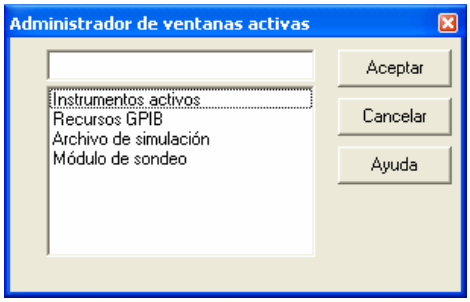


Figura A.13. Administrador de ventanas activas.

A.4.6. Controlador

Un sistema GPIB necesita contar con al menos un instrumento y un controlador para realizar la comunicación, el controlador tiene la función de asignar la dirección a los instrumentos, coordinar el envío de mensajes y órdenes IEEE 488.2 hacia los instrumentos y la recepción de los datos mediante la función de transferencia (*handshake*).

Existen dos forma de acceder al controlador del bus, de forma directa mediante un clic en el icono que se representa un computadora portátil (figura A.7) de la **Ventana de Instrumentos activos** ó mediante la activación de la opción **Configurar aplicación** del menú **Bus** (figura A.6).

La figura A.14 ilustra la activación del controlador al realizar una comunicación bajo el protocolo IEEE 488 mediante el envío de peticiones.

El controlador realiza las peticiones de medición de los valores de los instrumentos activos en el bus. Para realizar una petición se selecciona la dirección primaria del instrumento e inmediatamente se presenta el panel frontal del instrumento seleccionado. La forma en la cual se realiza la petición de valores es dando un clic en el botón correspondiente a la medida que se desea realizar. En la figura A.14 se muestra cómo el controlador realiza una petición del valor de frecuencia al generador de funciones mediante su panel frontal.

Otra alternativa de comunicación con los instrumentos se realiza mediante el envío de órdenes, para realizar esta comunicación se debe activar el controlador, proporcionar la dirección primaria del instrumento con que se desee comunicar, desactivar el panel frontal del instrumento activo y acceder al botón de **Comunicación IEEE 488.2**, ubicado dentro de la **Ventana del controlador del GPIB**, una vez activada la comunicación se presentan un ventana como lo muestra la figura A.15.

Figura A.14. Ventana Controlador del GPIB.

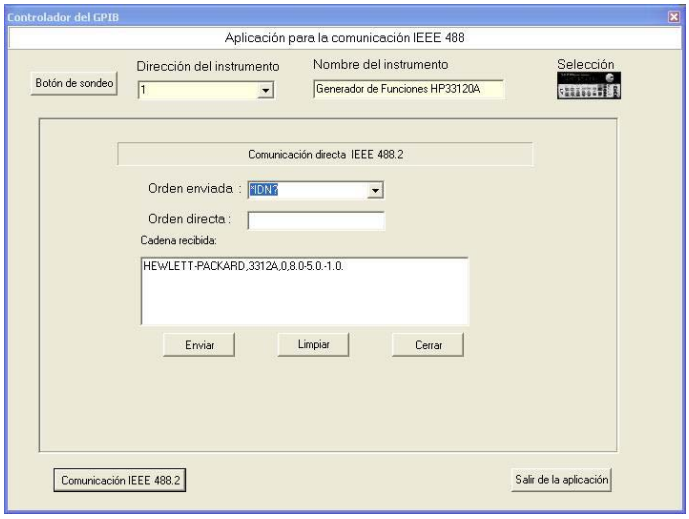


Figura A.15. Ventana de comunicación IEEE 488.2.

En la ventana de comunicación mediante de órdenes se debe elegir entre las dos alternativas de envío, a través de órdenes predeterminadas por una cortina, en la opción **Orden enviada**, o mediante la captura de la opción **Enviar**. El sistema identifica la orden enviada al instrumento y devuelve una cadena de datos como respuesta.

La figura A.15 presenta la activación de una comunicación IEEE 488.2 mediante el envío de un orden de identificación *IDN?, la cual devuelve la identificación del dispositivo.

A.4.7. Sondeo

Una simulación permite dos técnicas para la identificación de los dispositivos, sondeo serie y sondeo paralelo. La función del sondeo es determinar el nombre, tipo y dirección primaria de los instrumentos agregados al bus.

Esta función puede ser activada desde el módulo del controlador con el **Botón de sondeo** o desde el **Administrador de ventanas activas** descrito anteriormente, la figura A.16 ilustra el acceso al módulo sondeo activado por el controlador.

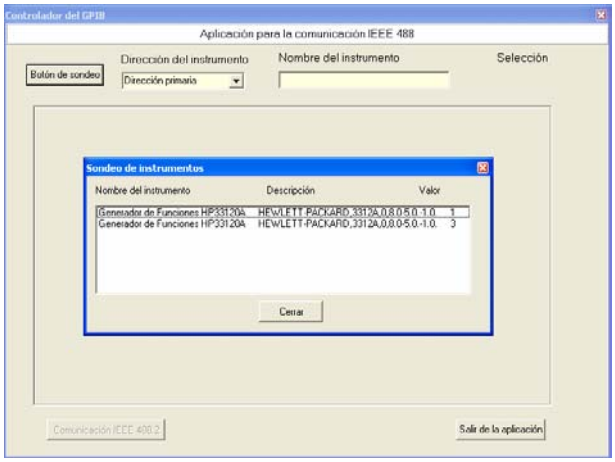


Figura A.16. Ventana de Sondeo de instrumentos activada por el controlador.

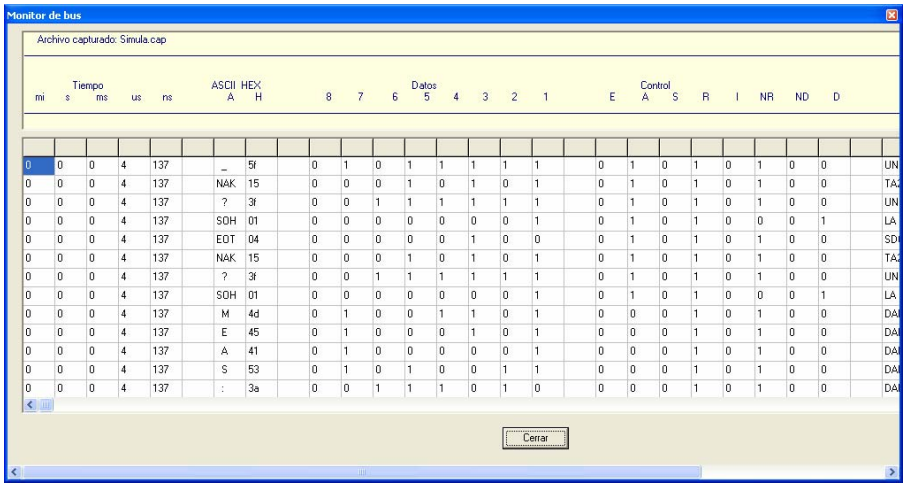


Figura A.17. Salida del Monitor de bus.

A.4.8. Monitor de bus

El objetivo de realizar una simulación GPIB es analizar el flujo de datos a través del bus y la decodificación de la trama de datos, por ello la función del **Monitor de bus** es visualizar cada uno de los datos enviados (órdenes o mensajes) del controlador a los instrumentos o viceversa.

Cada uno de los datos enviados al monitor es filtrado por la configuración de la simulación, la cual determina cuáles datos deberán ser presentados en el **Monitor de bus** debido a que cada filtro representa una salida distinta.

Existen dos formas de acceder al **Monitor de bus**, mediante la activación de la opción **MonGPIB** ubicado en el menú **Monitor** (figura A.6) y a través del acceso directo al icono de la barra de iconos de la **Ventana principal SepiGPIB**. La figura A.17 representa la salida de una simulación capturada por el **Monitor del bus**.

A.4.9. Nueva simulación

El sistema permite de reiniciar durante la simulación mediante la opción **Nuevo** del menú **Archivo** (figura A.6) o mediante el acceso directo ubicado en la barra de iconos de la **Ventana principal SepiGPIB.**

A.4.10. Abrir archivo capturado

El programa SepiGPIB cuenta con un módulo de visualización de captura que recupera un Archivo de tramas y lo muestra en una ventana para realizar el análisis de los datos.

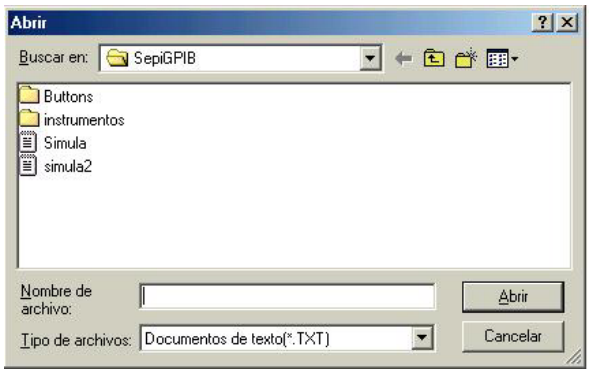


Figura A.18. Abrir archivo de simulación.

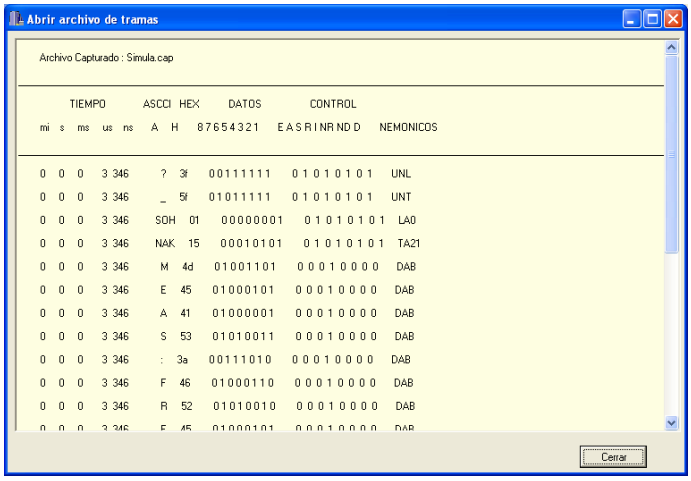


Figura A.19. Visualización de un archivo capturado.

La forma de acceder a este módulo es a través de la opción **Abrir** dentro del menú **Archivo** o con el icono que se encuentra en la barras de iconos de la **Ventana principal SepiGPIB**. Para abrir un archivo hay que elegir su ruta de ubicación, seleccionar el archivo y dar un clic en el botón **Abrir** (figura A.18).

La presentación de los datos del archivo es en formato texto, la figura A.19 ilustra la representación de los datos en la ventana correspondiente.

A.4.11. Guardar simulación

Toda captura del monitor de bus se almacena en un archivo llamado Simula.txt. Para almacenar la simulación con otro nombre es necesario acceder a la opción **Guardar** dentro del menú **Archivo** o a través del acceso directo al icono de la barra de iconos de la **Ventana principal SepiGPIB**, posteriormente se debe elegir la ubicación destino del archivo, agregar el nombre deseado y dar un clic en el botón **Guardar** (figura A.20).

A.4.12. Ayuda

Este menú cuenta con tres opciones para proporcionar ayuda a los usuarios del sistema:

- *Ayuda del SepiGPIB*: proporciona información acerca de los módulos, accesos y formas de activar el sistema.
- *Capítulo GPIB (PDF)*: contiene una descripción del protocolo de instrumentación GPIB.
- *Manual SepiGPIB (PDF)*: contiene un manual del sistema SepiGPIB.
- *Acerca de SepiGPIB*: brinda información acerca del sistema (figura A.21).

Figura A.20. Guardar simulación.

Figura A.21. Ventana del módulo Acerca de.

B. Hojas de especificación de los instrumentos GPIB

B.1. Generador de funciones modelo 33120A de la firma Agilent

B.2. Multímetro modelo 34401A de la firma Agilent

B.3. Fuente de voltaje modelo 3610A de la firma Agilent

B.1. Generador de funciones modelo 33120A de la firma Agilent

Agilent 33120A

Function/Arbitrary Waveform Generator

Data Sheet

- **15 MHz sine and square wave outputs**
- **Sine, triangle, square, ramp, noise and more**
- **12-bit, 40MSa/s, 16,000-point deep arbitrary waveforms**
- **Direct digital synthesis for excellent stability**

Uncompromising performance for standard waveforms
The Agilent Technologies 33120A Function/Arbitrary Waveform Generator uses direct digital-synthesis techniques to create a stable, accurate output signal for clean, low-distortion sine waves. It also gives you fast rise- and fall-time square wave, and linear ramp waveforms down to 100 µHz.

Custom waveform generation
Use the 33120A to generate complex custom waveforms such as a heartbeat or the output of a mechanical transducer. With 12-bit resolution, and a sampling rate of 40 MSa/s, the 33120A gives you the flexibility to create any waveform you need. It also lets you store up to four 16,000-deep waveforms in nonvolatile memory.

Easy-to-use functionality
Front-panel operation of the 33120A is straightforward and intuitive. You can access any of ten major functions with a single key press or two, then use a simple knob to adjust frequency, amplitude and offset. To save time, you can enter voltage values directly in Vp-p, Vrms or dBm.

Internal AM, FM, FSK and burst modulation make it easy to modulate waveforms without the need for a separate modulation source. Linear and log sweeps are also built in, with sweep rates selectable from 1 ms to 500 s. GPIB and RS-232 interfaces are both standard, plus you get full programmability using SCPI commands.

Optional phase-lock capability
The Option 001 phase lock/TCXO timebase gives you the ability to generate synchronized phase-offset signals. An external clock input/output lets you synchronize with up to three other 33120As or with an external 10-MHz clock.

Option 001 also gives you a TCXO timebase for increased frequency stability. With accuracy of 4 ppm/yr, the TCXO timebase make a 33120A ideal for frequency calibrations and other demanding applications.

With Option 001, new commands let you perform phase changes on the fly, via the front panel or from a computer, allowing precise phase calibration and adjustment.

Link the Agilent 33120A to your PC
The included Agilent IntuiLink software allows you to easily create, edit, and download complex waveforms using the IntuiLink Arbitrary Waveform Editor. Or you can capture a waveform using IntuiLink Oscilloscope or DMM and send it to the 33120A for output. For programmers, ActiveX components can be used to control the instrument using SCPI commands. IntuiLink provides the tools to easily create, download, and manage waveforms for your 33120A. To find out more about IntuiLink, visit **www.agilent.com/find/intuilink**.

The 33120A can also be used in conjunction with the 34811A BenchLink Arb software. This Windows®-based program lets you create and edit waveforms on your PC and download them to the 33120A.

3-year warranty
With your 33120A, you get operating and service manuals, a quick reference guide, test date, and a full 3-year warranty, all for one low price.

Agilent Technologies

Waveforms

Standard	Sine, square, triangle, ramp, noise, sin(x)/x, exponential rise exponential fall, cardiac, dc volts.
Arbitrary	
Waveform length	8 to 16,000 points
Amplitude resolution	12 bits (including sign)
Sample rate	40 MSa/s
Non-volatile memory	Four (4) 16,000 waveforms

Frequency Characteristics

Sine	100 µHz - 15 MHz
Square	100 µHz - 15 MHz
Triangle	100 µHz - 100 kHz
Ramp	100 µHz - 100 kHz
White noise	10 MHz bandwidth
Resolution	10 µHz or 10 digits
Accuracy	10 ppm in 90 days, 20 ppm in 1 year, 18°C - 28°C
Temp. Coeff	< 2 ppm/°C
Aging	< 10 ppm/yr

Sinewave Spectral Purity

Harmonic distortion	
dc to 20 kHz	-70 dBc
20 kHz to 100 kHz	-60 dBc
100 kHz to 1 MHz	-45 dBc
1 MHz to 15 MHz	-35 dBc
Spurious (non-harmonic)	
DC to 1 MHz	< -65 dBc
1 MHz to 15 MHz	< -65 dBc + 6 dB/octave
Total harmonic distortion	
DC to 20 kHz	<0.04%
Phase noise	<-55 dBc in a 30 kHz band

Signal Characteristics

Squarewave	
Rise/Fall time	< 20 ns
Overshoot	4%
Asymmetry	1% + 5ns
Duty cycle	20% to 80% (to 5 MHz) 40% to 60% (to 15 MHz)
Triangle, Ramp, Arb	
Rise/Fall time	40 ns (typical)
Linearity	<0.1% of peak output
Setting Time	<250 ns to 0.5% of final value
Jitter	<25ns

Output Characteristics

Amplitude (into 50Ω)	50 mVpp - 10 Vpp [1]
Accuracy (at 1 kHz)	± 1% of specified output
Flatness *(sinewave relative to 1 kHz)*	
< 100 kHz	± 1% (0.1 dB)
100 kHz to 1 MHz	± 1.5% (0.15 dB)
1 Mz to 15 MHz	± 2% (0.2 dB) Ampl ≥ 3Vrms ± 3.5% (0.3 dB) Ampl < 3Vrms
Output Impedance	50Ω (fixed)
Offset (into 50Ω) [2]	+ 5 Vpk ac + dc
Accuracy	± 2% of setting + 2 mV
Resolution	3 digits, amplitude and offset
Units	Vpp, Vrms, dBm
Isolation	42 Vpk maximum to earth
Protection	Short circuit protected ± 15 Vpk overdrive < 1 minute
Modulation	
AM	
Carrier -3dB Freq.	10 MHz (typical)
Modulation	any internal waveform including Arb
Frequency	10 mHz - 20 kHz
Depth	0% - 120%
Source	Internal/External
FM	
Modulation	any internal waveform including Arb
Frequency	10 mHz - 10 kHz
Deviation	10 mHz - 15 MHz
Source	Internal only
FSK	
Internal rate	10 mHz - 50 kHz
Frequency Range	10 mHz - 15 MHz
Source	Internal/External (1 MHz max.)
Burst	
Carrier Freq.	5 MHz max.
Count	1 to 50,000 cycles or infinite
Start Phase	-360° to +360°
Internal Rate	10 mHz - 50 kHz ± 1%
Gate Source	Internal/External Gate
Trigger	Single, External or Internal Rate

Sweep	
Type	Linear or Logarithmic
Direction	Up or Down
Start F/Stop F	10 mHz - 15 MHz
Speed	1 ms to 500 s ± 0.1%
Trigger	Single, External, or Internal
Rear Panel Inputs	
Ext. AM Modulation	± 5 Vpk = 100% modulation 5kΩ input resistance
External Trigger/ FSK/Burst Gate	TTL low true

System Characteristics [3]

Configuration Times [4]	
Function Change: [5]	80 ms
Frequency Change: [5]	30 ms
Amplitude Change:	30 ms
Offset Change:	10 ms
Select User Arb:	100 ms
Modulation Parameter Change:	<350 ms

Arb Download Times over GPIB

Arb Length	Binary	ASCII Integer	ASCII Real [6]
16,000 points	8 sec	81 sec	100 sec
8,192 points	4 sec	42 sec	51 sec
4,096 points	2.5 sec	21 sec	26 sec
2,048 points	1.5 sec	11 sec	13 sec

Arb Download Times over RS-232 at 9600 Baud [7]

Arb Length	Binary	ASCII Integer	ASCII Real [8]
16,000 points	35 sec	101 sec	134 sec
8,192 points	18 sec	52 sec	69 sec
4,096 points	10 sec	27 sec	35 sec
2,048 points	6 sec	14 sec	18 sec

[1] 100 mVpp - 20 Vpp into open circuit

[2] Offset ≤ 2x pk - pk amplitude

[3] Times are typical. May vary based on controller performance

[4] Time to change parameter and output the new signal.

[5] Modulation or sweep off

[6] Times for 5-digit and 12-digit numbers

[7] For 4800 baud, multiply the download times by two; For 2400 baud, multiply the download times by four, etc.

[8] Time for 5-digit numbers; for 12-digit numbers, multiply the 5-digit numbers by two

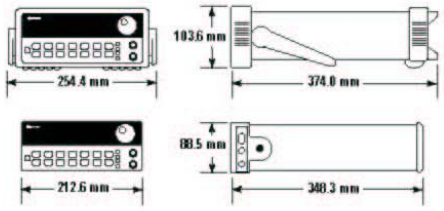
Option 001 Phaselock/TCXO Timebase

Timebase Accuracy	
Setability	< 0.01 ppm
Stability	± 1 ppm 0° - 50°
Aging	< 2ppm in first 30 days (continuous operation) 0.1 ppm/month (after first 30 days)
External Reference Input	
Lock Range	10 MHz ± 50 Hz
Level	-10 dBm to + 15 dBm +25 dBm or 10 Vpp max input
Impedance	50Ω ± 2%, 42 Vpk isolation to earth
Lock Time	< 2 seconds
Internal Reference Output	
Frequency	10 MHz
Level	> 1 Vpp into 50 Ω
Phase Offset	
Range	+ 360° to - 360°
Resolution	0.001°
Accuracy	25 ns
Trigger Output	
Level	5V zero-going pulse
Pulse Width	> 2µs typical
Fanout	Capable of driving up to three 33120As

Ordering Information
Agilent 33120A Function/Arb Generator
Opt. 001 Phase Lock/TCXO Timebase Option

General

Power Supply	110V/120V/220V/240V ± 10%
Power Line Frequency	45 Hz to 66 Hz and 360 Hz to 440 Hz
Power Consumption	50VA peak (28 W average)
Operating Environment	0°C to 55°C
Storage Environment	-40°C to 70°C
State Storage Memory	Power Off state automatically saved, 3 User Configurable Stored States
Interface	IEEE-488 and RS-232 standard
Language	SCPI - 1993, IEEE-488.2
Dimensions (W x H x D)	
Bench top	254.4mm x 103.6mm x 374mm
Rack mount	212.6mm x 88.5mm x 348.3mm
Weight	4 kg (8.8 lbs)
Safety Designed to	UL-1244, CSA 1010, EN61010
EMC Tested to	MIL-461C, EN55011, EN50082-1
Vibration and Shock	MIL-T-28800, Type III, Class 5
Acoustic Noise	30 dBa
Warm-up Time	1 hour
Warranty	3 years standard



Ordering Information

33120A Function/Arbitrary Waveform Generator

Accessories included

Operating manual, service manual, quick reference guide, IntuiLink connectivity software, test data, and power cord

Options

Opt. 001 Phase lock/TCXO timebase
Opt. 106 BenchLink Arb software (34811A)
Opt. 1CM Rack Mount Kit (34190A)*
Opt. W50 Additional 2-year warranty (5-year total)
Opt. 910 Extra manual set

Manual language options (please specify one)

ABA US English
ABD German
ABE Spanish
ABF French
ABJ Japanese
ABZ Italian
AB0 Taiwan Chinese
AB1 Korean

Accessories

Agilent 34161A Accessory pouch
Agilent 34811A BenchLink Arb software

*For racking two side-by-side, order both items below
Lock-link Kit (P/N 5061-9694)
Flange Kit (P/N 5063-9212)

Agilent Technologies' Test and Measurement Support, Services, and Assistance

Agilent Technologies aims to maximize the value you receive, while minimizing your risk and problems. We strive to ensure that you get the test and measurement capabilities you paid for and obtain the support you need. Our extensive support resources and services can help you choose the right Agilent products for your applications and apply them successfully. Every instrument and system we sell has a global warranty. Support is available for at least five years beyond the production life of the product. Two concepts underlie Agilent's overall support policy: "Our Promise" and "Your Advantage."

Our Promise

Our Promise means your Agilent test and measurement equipment will meet its advertised performance and functionality. When you are choosing new equipment, we will help you with product information, including realistic performance specifications and practical recommendations from experienced test engineers. When you use Agilent equipment, we can verify that it works properly, help with product operation, and provide basic measurement assistance for the use of specified capabilities, at no extra cost upon request. Many self-help tools are available.

Your Advantage

Your Advantage means that Agilent offers a wide range of additional expert test and measurement services, which you can purchase according to your unique technical and business needs. Solve problems efficiently and gain a competitive edge by contracting with us for calibration, extra-cost upgrades, out-of-warranty repairs, and on-site education and training, as well as design, system integration, project management, and other professional engineering services. Experienced Agilent engineers and technicians worldwide can help you maximize your productivity, optimize the return on investment of your Agilent instruments and systems, and obtain dependable measurement accuracy for the life of those products.

By internet, phone, or fax, get assistance with all your test & measurement needs

Online assistance:
www.agilent.com/find/assist

Phone or Fax
United States:
(tel) 1 800 452 4844

Canada:
(tel) 1 877 894 4414
(fax) (905) 282 6495

Europe:
(tel) (31 20) 547 2323
(fax) (31 20) 547 2390

Japan:
(tel) (81) 426 56 7832
(fax) (81) 426 56 7840

Latin America:
(tel) (305) 269 7500
(fax) (305) 269 7599

Australia:
(tel) 1 800 629 485
(fax) (61 3) 9210 5947

New Zealand:
(tel) 0 800 738 378
(fax) 64 4 495 8950

Asia Pacific:
(tel) (852) 3197 7777
(fax) (852) 2506 9284

Printed in USA May 2, 2001
5968-0129EN

Agilent Technologies

B.2. Multímetro modelo 34401A de la firma Agilent

Agilent 34401A Multimeter

Uncompromising Performance for Benchtop and System Testing

Product Overview

- Measure up to 1000 volts with 6 1/2 digits resolution
- 0.0015% basic dcV accuracy (24 hour)
- 0.06% basic acV accuracy (1 year)
- 3Hz to 300kHz ac bandwidth
- 1000 readings/sec. direct to GPIB

Superior performance

The Agilent Technologies 34401A multimeter gives you the performance you need for fast, accurate bench and system testing. The 34401A provides a combination of resolution, accuracy and speed that rivals DMMs costing many times more. 6 1/2-digits of resolution, 0.0015% basic 24-hr dcV accuracy and 1,000 readings/sec direct to GPIB assure you of results that are accurate, fast, and repeatable.

Use it on your benchtop

The 34401A was designed with your bench needs in mind. Functions commonly associated with bench operation, like continuity and diode test, are built in. A Null feature allows you to remove lead resistance and other fixed offsets in your measurements. Other capabilities like min/max/avg readouts and direct dB and dBm measurements make checkout with the 34401A faster and easier.

The 34401A gives you the ability to store up to 512 readings in internal memory. For trouble-shooting, a reading hold feature lets you concentrate on placing your test leads without having to constantly glance at the display.

Use it for systems testing

For systems use, the 34401A gives you faster bus throughput than any other DMM in its class. The 34401A can send up to 1,000 readings/sec directly across GPIB in user-friendly ASCII format.

You also get both GPIB and RS-232 interfaces as standard features. Voltmeter Complete and External Trigger signals are provided so you can synchronize to other instruments in your test system. In addition, a TTL output indicates Pass/Fail results when limit testing is used.

To ensure both forward and backward compatibility, the 34401A includes three command languages (SCPI, Agilent 3478A and Fluke 8840A /42A), so you don't have to rewrite your existing test software. An optional rack mount kit is available.

Easy to use

Commonly accessed attributes, such as functions, ranges, and resolution are selected with a single button press.

Advanced features are available using menu functions that let you optimize the 34401A for your applications.

The included Agilent IntuiLink software allows you to put your captured data to work easily, using PC applications such as Microsoft Excel® or Word® to analyze, interpret, display, print, and document the data you get from the 34401A. You can specify the meter setup and take a single reading or log data to the Excel spreadsheet in specified time intervals. Programmers can use ActiveX components to control the DMM using SCPI commands. To find out more about IntuiLink, visit **www.agilent.com/find/intuilink**

The 34401A can also be used in conjunction with the 34812A BenchLink Meter software. This Windows-based program lets you configure and initiate measurements from your computer, and transfer results from your test instrument to your PC.

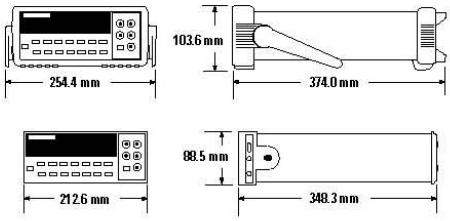
3-year warranty

With your 34401A, you get full documentation, a high-quality test lead set, calibration certificate with test data, and a 3-year warranty, all for one low price.

Agilent Technologies
Innovating the HP Way

Accuracy Specifications ± (% of reading + % of range)[1]

Function	Range[2]	Frequency, etc.	24 Hour [3] 23°C ± 1°C	90 Day 23°C ± 5°C	1 Year 23°C ± 5°C	Temperature Coefficient 0°C – 18°C 28°C – 55°C
dc Voltage	100.0000 mV		0.0030 + 0.0030	0.0040 + 0.0035	0.0050 + 0.0035	0.0005 + 0.0005
	1.000000 V		0.0020 + 0.0006	0.0030 + 0.0007	0.0040 + 0.0007	0.0005 + 0.0001
	10.00000 V		**0.0015 + 0.0004**	**0.0020 + 0.0005**	**0.0035 + 0.0005**	**0.0005 + 0.0001**
	100.0000 V		0.0020 + 0.0006	0.0035 + 0.0006	0.0045 + 0.0006	0.0005 + 0.0001
	1000.000 V		0.0020 + 0.0006	0.0035 + 0.0010	0.0045 + 0.0010	0.0005 + 0.0001
True rms ac Voltage[4]	100.0000 mV	3 Hz - 5 Hz	1.00 + 0.03	1.00 + 0.04	1.00 + 0.04	0.100 + 0.004
		5 Hz - 10 Hz	0.35 + 0.03	0.35 + 0.04	0.35 + 0.04	0.035 + 0.004
		10 Hz - 20 kHz	0.04 + 0.03	0.05 + 0.04	0.06 + 0.04	0.005 + 0.004
		20 kHz - 50 kHz	0.10 + 0.05	0.11 + 0.05	0.12 + 0.04	0.011 + 0.005
		50 kHz - 100 kHz	0.55 + 0.08	0.60 + 0.08	0.60 + 0.08	0.060 + 0.008
		100 kHz - 300 kHz[8]	4.00 + 0.50	4.00 + 0.50	4.00 + 0.50	0.20 + 0.02
	1.000000 V to 750.000 V	3 Hz - 5 Hz	1.00 + 0.02	1.00 + 0.03	1.00 + 0.03	0.100 + 0.003
		5 Hz - 10 Hz	0.35 + 0.02	0.35 + 0.03	0.35 + 0.03	0.035 + 0.003
		10 Hz - 20 kHz	**0.04 + 0.02**	**0.05 + 0.03**	**0.06 + 0.03**	**0.005 + 0.003**
		20 kHz - 50 kHz	0.10 + 0.04	0.11 + 0.05	0.12 + 0.04	0.011 + 0.005
		50 kHz - 100 kHz[5]	0.55 + 0.08	0.60 + 0.08	0.60 + 0.08	0.060 + 0.008
		100 kHz - 300 kHz[8]	4.00 + 0.50	4.00 + 0.50	4.00 + 0.50	0.20 + 0.02
Resistance[7]	100.0000 Ω	1 mA Current Source	0.0030 + 0.0030	0.008 + 0.004	0.010 + 0.004	0.0006 + 0.0005
	1.000000 kΩ	1 mA	0.0020 + 0.0005	0.008 + 0.001	0.010 + 0.001	0.0006 + 0.0001
	10.00000 kΩ	100 µA	**0.0020 + 0.0005**	**0.008 + 0.001**	**0.010 + 0.001**	**0.0006 + 0.0001**
	100.0000 kΩ	10 µA	0.0020 + 0.0005	0.008 + 0.001	0.010 + 0.001	0.0006 + 0.0001
	1.000000 MΩ	5.0 µA	0.002 + 0.001	0.008 + 0.001	0.010 + 0.001	0.0010 + 0.0002
	10.00000 MΩ	500 nA	0.015 + 0.001	0.020 + 0.001	0.040 + 0.001	0.0030 + 0.0004
	100.0000 MΩ	500 nA || 10MΩ	0.300 + 0.010	0.800 + 0.010	0.800 + 0.010	0.1500 + 0.0002
dc Current	10.00000 mA	<0.1 V Burden Voltage	0.005 + 0.010	0.030 + 0.020	0.050 + 0.020	0.002 + 0.0020
	100.0000 mA	<0.6 V	**0.010 + 0.004**	**0.030 + 0.005**	**0.050 + 0.005**	**0.002 + 0.0005**
	1.000000 A	<1 V	0.050 + 0.006	0.080 + 0.010	0.100 + 0.010	0.005 + 0.0010
	3.00000 A	<2 V	0.100 + 0.020	0.120 + 0.020	0.120 + 0.020	0.005 + 0.0020
True rms ac Current[8]	**1.000000 A**	3 Hz - 5 Hz	1.00 + 0.04	1.00 + 0.04	1.00 + 0.04	0.100 + 0.006
		5 Hz - 10 Hz	0.30 + 0.04	0.30 + 0.04	0.30 + 0.04	0.035 + 0.006
		10 Hz - 5 kHz	**0.10 + 0.04**	**0.10 + 0.04**	**0.10 + 0.04**	**0.015 + 0.006**
	3.00000 A	3 Hz - 5 Hz	1.10 + 0.06	1.10 + 0.06	1.10 + 0.06	0.100 + 0.006
		5 Hz - 10 Hz	0.35 + 0.06	0.35 + 0.06	0.35 + 0.06	0.035 + 0.006
		10 Hz - 5 kHz	0.15 + 0.06	0.15 + 0.06	0.15 + 0.06	0.015 + 0.006
Frequency or Period[6]	100 mV to 750 V	3 Hz - 5 Hz	0.10	0.10	0.10	0.005
		5 Hz - 10 Hz	0.05	0.05	0.05	0.005
		10 Hz - 40 Hz	0.03	0.03	0.03	0.001
		40 Hz - 300 kHz	**0.006**	**0.01**	**0.01**	**0.001**
Continuity	1000.0Ω	1mA Test Current	0.002 + 0.010	0.008 + 0.020	0.010 + 0.020	0.001 + 0.002
Diode Test	1.0000V	1mA Test Current	0.002 + 0.010	0.008 + 0.020	0.010 + 0.020	0.001 + 0.002



1 Specifications are for 1hr warm-up and 6½ digits, Slow ac filter.
2 Relative to calibration standards.
3 20% over range on all ranges except 1000Vdc and 750Vac ranges.
4 For sinewave input > 5% of range. For inputs from 1% to 5% of range and < 50kHz, add 0.1% of range additional error.
5 750V range limited to 100 kHz or 8 x107 Volt-Hz.
6 Typically 30% of reading error at 1MHz.
7 Specifications are for 4- wire ohms function or 2-wire ohms using Math Null. Without Math Null, add 0.2 Ω additional error in 2-wire ohms function.
8 Input >100 mV. For 10 mV inputs multiply % of reading error x10.

Measurement Characteristics

dc Voltage

Measurement Method	Continuously Integrating Multi-slope III A-D Converter
A-D Linearity	0.0002% of reading + 0.0001 % of range
Input Resistance	
0.1V, 1V,10 V ranges	Selectable 10 MΩ or >10,000 MΩ
100 V, 1000 V ranges	10 MΩ ± 1%
Input Bias Current	< 30pA at 25° C
Input Protection	1000 V all ranges
dcV:dcV Ratio Accuracy	V_{input} Accuracy + $V_{reference}$ Accuracy

True rms ac Voltage

Measurement Method	ac coupled True rms – measures the ac component of the input with up to 400 Vdc of bias on any range.
Crest Factor	Maximum of 5:1 at Full Scale
Additional Crest Factor Errors (non-sinewave)	Crest Factor 1–2 0.05 % of reading Crest Factor 2–3 0.15 % of reading Crest Factor 3–4 0.30 % of reading Crest Factor 4–5 0.40 % of reading
Input Impedance	1 MΩ ± 2% in parallel with 100 pF
Input Protection	750Vrms all ranges

Resistance

Measurement Method	Selectable 4-wire or 2-wire Ohms. Current source referenced to LO input.
Maximum Lead Resistance (4-wire)	10% of range per lead for 100Ω and 1kΩ ranges. 1kΩ per lead on all other ranges.
Input Protection	1000 V all ranges

dc Current

Shunt Resistance	5Ω for 10 mA,100 mA; 0.1 Ω for 1 A, 3 A
Input Protection	Externally accessible 3 A 250 V Fuse Internal 7 A 250 V Fuse

1 For 1kΩ unbalance in LO lead.
2 For power line frequency ± 0.1%.
3 For power line frequency ± 1% use 40dB or ± 3% use 30dB.
4 Reading speeds for 60Hz and (50Hz) operation.
5 Maximum useful limit with default settling delays defeated.
6 Speeds are for 4½ digits, Delay 0, Auto-zero and Display OFF.

True rms ac Current

Measurement Method	Direct coupled to the fuse and shunt. ac coupled True rms measurement (measures the ac component only).
Shunt Resistance	0.1 Ω for 1 A and 3 A ranges
Input Protection	Externally accessible 3 A 250 V Fuse Internal 7 A 250 V Fuse

Frequency and Period

Measurement Method	Reciprocal counting technique
Voltage Ranges	Same as ac Voltage Function
Gate Time	1 s, 100 ms, or 10 ms.

Continuity / Diode

Response Time	300 samples/s with audible tone
Continuity Threshold	Selectable from 1 Ω to 1000 Ω

Measurement Noise Rejection 60 (50) Hz[1]

dc CMRR	140 dB
ac CMRR	70 dB

Integration Time	Normal Mode Rejection[2]
100 plc / 1.67 s (2 s)	60 dB[3]
10 plc / 167 ms (200 ms)	60 dB[3]
1 plc / 16.7 ms (20 ms)	60 dB
<1 plc / 3 ms or 800 µs	0 dB

Operating Characteristics[4]

Function	Digits	Readings/s
dcV, dcI, and Resistance	6½	0.6 (0.5)
	6½	6 (5)
	5½	60 (50)
	5½	300
	4½	1000
acV, acI	6½	0.15 Slow (3Hz)
	6½	1 Medium (20Hz)
	6½	10 Fast (200Hz)
	6½	50[5]
Frequency or Period	6½	1
	5½	9.8
	4½	80

System Speeds[6]

Configuration Rates	26/s to 50/s
Autorange Rate (dc Volts)	>30/s
ASCII readings to RS-232	55/s
ASCII readings to GPIB	1000/s
Maximum Internal Trig. Rate	1000/s
Max. Ext. Trig. Rate to Memory	1000/s

Triggering and Memory

Reading HOLD Sensitivity	10%, 1%, 0.1%, or 0.01% of range
Samples/ trigger	1 to 50,000
Trigger Delay	0 to 3600 s: 10 µs step size
External Trigger Delay	< 1 ms
External Trigger Jitter	< 500 µs
Memory	512 readings

Math Functions

NULL, Min/Max/Average, dBm, dB, Limit Test (with TTL output)

Standard Programming Languages

SCPI (IEEE-488.2), Agilent 3478A, Fluke 8840A/42A

Accessories Included

Test Lead Kit with probe, alligator, and grabber attachments.
Operating Manual, Service Manual, test report, and power cord.

General Specifications

Power Supply	100 V/120 V/220 V/ 240 V ±10%
Power Line Frequency	45 Hz to 66 Hz and 360 Hz to 440 Hz Automatically sensed at power-on
Power Consumption	25 VA peak (10W average)
Operating Environment	Full accuracy for 0° C to 55° C Full accuracy to 80% R.H. at 40° C
Storage Environment	– 40° C to 70° C
Weight	3.6 kg (8.0 lbs)
Safety	Designed to CSA, UL-1244, IEC-348
RFI and ESD	MIL-461C, FTZ 1046, FCC
Vibration and Shock	MIL-T-28800E, Type III, Class 5 (Sine Only)
Warranty	3 years

Ordering Information

Agilent 34401A Multimeter

Accessories included

Test Lead Kit with probe, alligator, and grabber attachments, IntuiLink connectivity software, operating manual, service manual, calibration certificate, test report, and power cord.

Options

Opt. 908 Rack Mount Kit* (P/N 5062-3972)

Opt. 910 Extra manual set (English)

Opt. OBO DMM without manuals

Opt. W50 Additional 2-year warranty (5-year total)

Opt. 1BP MIL-STD-45662A calibration with data

Manual options (please specify one)

ABA US English
ABD German
ABE Spanish
ABF French
ABJ Japanese
ABZ Italian
ABO Taiwan Chinese
AB1 Korean
AB2 Chinese
AKT Russian

Agilent Accessories

11059A Kelvin Probe set

11060A Surface Mount Device (SMD) test probes

11062A Kelvin clip set

34131 Hard Transit Case

34161A Accessory pouch

34171A Input terminal connector (sold in pairs)

34172A Input calibration short (sold in pairs)

34330A 30 A current shunt

34812A BenchLink Meter software

E2308A 5K thermistor probe

*For racking two side-by-side, order both items below
Lock link kit (P/N 5061-9694)
Flange kit (P/N 5063-9212)

Agilent Technologies' Test and Measurement Support, Services, and Assistance

Agilent Technologies aims to maximize the value you receive, while minimizing your risk and problems. We strive to ensure that you get the test and measurement capabilities you paid for and obtain the support you need. Our extensive support resources and services can help you choose the right Agilent products for your applications and apply them successfully. Every instrument and system we sell has a global warranty. Support is available for at least five years beyond the production life of the product. Two concepts underlie Agilent's overall support policy: "Our Promise" and "Your Advantage."

Our Promise

Our Promise means your Agilent test and measurement equipment will meet its advertised performance and functionality. When you are choosing new equipment, we will help you with product information, including realistic performance specifications and practical recommendations from experienced test engineers. When you use Agilent equipment, we can verify that it works properly, help with product operation, and provide basic measurement assistance for the use of specified capabilities, at no extra cost upon request. Many self-help tools are available.

Your Advantage

Your Advantage means that Agilent offers a wide range of additional expert test and measurement services, which you can purchase according to your unique technical and business needs. Solve problems efficiently and gain a competitive edge by contracting with us for calibration, extra-cost upgrades, out-of-warranty repairs, and on-site education and training, as well as design, system integration, project management, and other professional engineering services. Experienced Agilent engineers and technicians worldwide can help you maximize your productivity, optimize the return on investment of your Agilent instruments and systems, and obtain dependable measurement accuracy for the life of those products.

By internet, phone, or fax, get assistance with all your test & measurement needs

Online assistance:
www.agilent.com/find/assist

Phone or Fax

United States:
(tel) 1 800 452 4844

Canada:
(tel) 1 877 894 4414
(fax) (905) 282-6495

Europe:
(tel) (31 20) 547 2323
(fax) (31 20) 547 2390

Japan:
(tel) (81) 426 56 7832
(fax) (81) 426 56 7840

Latin America:
(tel) (305) 269 7500
(fax) (305) 269 7599

Australia:
(tel) 1 800 629 485
(fax) (61 3) 9210 5947

New Zealand:
(tel) 0 800 738 378
(fax) 64 4 495 8950

Asia Pacific:
(tel) (852) 3197 7777
(fax) (852) 2506 9284

Printed in the USA. January 23, 2001
5968-0162EN

Agilent Technologies
Innovating the HP Way

B.3. Fuente de voltaje modelo 3610A de la firma Agilent

Agilent E36XX-Series Manual dc Power Supplies

Data Sheet

- **Linear power supply**
- **Single, dual or triple output**
- **10-turn voltage and current controls**
- **Digital voltage and current meters**
- **Low noise and excellent regulation**

Affordable, full-featured benchtop power supplies provide excellent performance and flexibility

A whole family of low-cost power supplies to meet your needs

The E3600-series of low-cost benchtop power supplies give you the performance of system power supplies without the high price. All E3600 family members give you clean power with dependable regulation and fast transient response. E3600-series single-output models are described on this page. See page 2 for information on dual- and triple-output models.

Single-output models

All E3600-series single-output power supplies feature separate digital-panel meters for monitoring voltage and current simultaneously, giving you precise reading and control capability. All models except the E3630A also feature 10-turn potentiometers for accurate adjustment of voltage and current output settings.

With 0.01 percent load and line regulation, these instruments keep the output steady when power line and load changes occur. The low normal-mode noise specification of less than 200µVrms ensures clean power for precision circuitry.

In all single-output models, either the positive or negative terminal can be connected to ground, providing a positive or negative voltage output. Outputs can also be floated up to 240V from ground.

These instruments also feature adjustable current limits, letting you set the safest current limit without having to short the output.

E3610A, E3611A, and E3612A single-output models

These popular 30-watt bench supplies are designed for general laboratory use. The constant-voltage, constant-current output allows operation as either a voltage or current source. The changeover occurs automatically, based on the load. Each of these models has two ranges, allowing more current at a lower voltage. For higher output voltages, supplies can be connected in series.

Agilent Technologies

E3614A, E3615A, E3616A and E3617A models feature overvoltage protection

These flexible 60-watt, single-range power supplies can be used as either voltage or current sources. When output terminal voltage increases to a preset shut-down level, an overvoltage protection circuit disables the output to protect the device under test (DUT) from damage. The overvoltage protection feature is easily monitored and adjusted from the front panel.

Using remote sensing capability, these instruments automatically compensate for voltage drop in the load leads, so you get accurate voltage at the DUT.

You can combine multiple units in auto-parallel, auto-series and auto-tracking configurations for greater output voltage or current capacity. Front and rear output terminals allow flexible configuration. Output voltage and current can be controlled with external 0- to10-volt analog voltage or variable resistance.

Multi-output models

With multiple supplies in a compact unit, the E3620A and E3630A give you excellent performance while saving space on your bench. Both instruments feature tight 0.01 percent line and load regulation and a low normal-mode noise specification of less than 0.35mV to ensure clean power for precision circuitry. With a common-mode current specification of less than 1uA, both multiple-output power supplies minimize power line current injection.

Like the single-output models in the E3600 series, the E3620A and E3630A feature separate digital panel meters so you can monitor voltage and current simultaneously. They also protect your DUT against overload and short-circuit damage. Smooth turn-on and turn-off transitions keep power spikes out of your circuits.

E3620A dual-output power supply

The 50-watt E3620A dual-output power supply provides two 0 V to 25 Vdc outputs with the maximum current of 1 A to satisfy most bench requirements. The outputs are completely independent and isolated.

E3630A triple-output power supply

The 35-watt E3630A triple- output power supply provides three dc outputs: 0 to 6 V with a maximum current of 1 to 2.5A and 0 to 20 V and 0 to -20 V with a maximum current of 0.5A. An autotracking feature lets you use one voltage control to adjust the +20 V and -20 V outputs simultaneously. The outputs track each other to within 1 percent, making it easy to adjust the power supply for circuits requiring balanced voltages.

3-year warranty

To ensure maximum reliablity and long life, all 3600-series power supplies undergo the same rigerous tests as Agilent top-of-the-line power supplies. Each instrument comes with a full 3-year warranty.

Specifications

	E3610A	E3611A	E3612A	E3614A	E3615A	E3616A	E3617A	E3620A	E3630A
Features	Dual range, 10 turn pots, Constant Voltage (CV), Constant Current (CC) modes.			Adjustable overvoltage protection, voltage & resistance programming, remote sense, rear outputs, ten turn pots, CV, CC modes. Multiple supplies can be connected for tracking or higher power.				Isolated dual outputs, 10 turn pots CV, CL	Tracking, CV, CL (±20 V) CV, CF (+6 V)
Number of outputs	1							2	3
Number of output Ranges	2	2	2	1	1	1	1	1	1
dc Output Rating	8 V, 3 A 15 V, 2 A	20 V, 1.5 A 35 V, 0.85 A	60 V, 0.5 A 120 V, 0.25 A	8 V, 6 A	20 V, 3 A	35 V, 1.7 A	60 V, 1 A	25 V, 1 A 25 V, 1 A	+6 V, 2.5 A +20 V, 0.5 A -20 V, 0.5 A
Load and Line Regulation	<0.01% + 2 mV								
Ripple and Noise (20 Hz to 20 MHz)									
Normal mode voltage	<200 µVrms, <2 mVpp			<200 µVrms, <1 mVpp				<350 µVrms, <1.5 mVpp	
Normal mode current	<200 µArms / 1 mApp			<0.02%+ 3 mA	<0.02%+ 1.5 mA	<0.02%+ 1 mA	<0.02%+ 0.5 mA	–	
Common mode current	not specified							<1 µArms	
Transient Response Time:	<50 µsec following change in output current from full load to half load for output to recover to within:								
	10 mV			15 mV					
Meter Accuracy	±0.5% + 2 counts at 25°C ±5°C								
Meter Resolution									
Voltage	10 mV	100 mV	100 mV	10 mV	10 mV (0-20 V), 100 mV (>20 V)				10 mV
Current	10 mA	10 mA	1 mA	10 mA	10 mA	1 mA	1 mA	1 mA	10 mA
Isolation	240 Vdc								

Supplemental Characteristics

Control Mode	CV/CC							CV/CL	CV/CL (±20 V) CV/CF (+6 V)
Temperature Coefficient per °C									
Voltage	<0.02% + 1 mV			<0.02% + 500 µV				<0.02% + 1 mV	
Current	<0.02% + 2 mA			<0.02% + 3 mA	<0.02% + 1.5 mA	<0.02% + 1 mA	<0.02% + 0.5 mA	–	
Output Drift									
Voltage	Less than 0.1% + 5 mV total drift for 8 hours after an initial warm-up of 30 minutes.								
Current	Less than 0.1% + 10 mA total drift for 8 hours after an initial warm-up of 30 minutes.								
Temperature Range									
	0 to 40°C for full rated output. Derate output current 1% per °C between 40°C and 55°C							Derate output current 3.3% per °C	
Cooling	Convection cooling								
Isolation	±240 Vdc								
AC Input	100 Vac ±10%, 47–63 Hz (opt. 0E9) 115 Vac ±10%, 47–63 Hz (std) 230 Vac ±10%, 47–63 Hz (opt. 0E3)								
Weight	3.8 kg (8.4 lb.) net, 5.1 kg (11.3 lbs) shipping			5.5 kg (12.1 lb.) net, 6.75 kg (14.9 lbs) shipping					Same as E3610A
Size	91 mm H x 213 mm W x 319 mm D 3.6" H x 8.4" W x 12.6" D			91 mm H x 213 mm W x 373 mm D 3.6" H x 8.4" W x 14.7" D					
Warranty	3 years								
Product Regulation	Certified to CSA 22.2 No. 231; conforms to IEC 1010-1; carries CE mark; complies with CISPR-11, Group 1, Class A								

www.agilent.com

Ordering Information

E3600-Series Power Supplies

E3610A 30-Watt Power Supply
E3611A 30-Watt Power Supply
E3612A 30-Watt Power Supply
E3614A 48-Watt Power Supply
E3615A 60-Watt Power Supply
E3616A 60-Watt Power Supply
E3617A 60-Watt Power Supply
E3620A Dual-output Power Supply
E3630A Triple-output Power Supply

Accessories included

Operating and service manuals and AC power cord

Power Options

Opt. 0E3 230 Vac ±10%
Opt. 0EM 115 Vac ±10%
Opt. 0E9 100 Vac ±10%

Other Options

Opt. 1CM Rack-mount kit* (E3614A, E3615A, E3616A, E3617A, E3620A)
Opt. 0L2 Extra Manual

Extra manual sets

E3610A/11A/12A Manual (P/N 5959-5304)
E3614A/15A/16A /17A Manual (P/N 5959-5310)
E3620A Manual (P/N E3620-90001)
E3630A Manual (P/N 5959-5329)

Rack Mount Kits*

E3610A/11A/12A/30A
(P/N 5063-9767)

E3614A/15A/16A/17A/20A

To rack mount instruments side by side
Lock-link Kit (P/N 5061-9694)
Flange Kit (P/N 5063-9212)

To rack mount one or two instruments in a sliding support shelf
Support Shelf (P/N 5063-9255)
Slide Kit (P/N 1494-0015), required for support shelf
For a single instrument, also order filler panel (P/N 5002-3999)

*Rackmounting with 1CM or lock-link/flange kit requires Agilent or customer supplied support rails
Agilent Support Rails - E3663AC

Agilent Technologies' Test and Measurement Support, Services, and Assistance
Agilent Technologies aims to maximize the value you receive, while minimizing your risk and problems. We strive to ensure that you get the test and measurement capabilities you paid for and obtain the support you need. Our extensive support resources and services can help you choose the right Agilent products for your applications and apply them successfully. Every instrument and system we sell has a global warranty. Support is available for at least five years beyond the production life of the product. Two concepts underlie Agilent's overall support policy: "Our Promise" and "Your Advantage."

Our Promise
Our Promise means your Agilent test and measurement equipment will meet its advertised performance and functionality. When you are choosing new equipment, we will help you with product information, including realistic performance specifications and practical recommendations from experienced test engineers. When you use Agilent equipment, we can verify that it works properly, help with product operation, and provide basic measurement assistance for the use of specified capabilities, at no extra cost upon request. Many self-help tools are available.

Your Advantage
Your Advantage means that Agilent offers a wide range of additional expert test and measurement services, which you can purchase according to your unique technical and business needs. Solve problems efficiently and gain a competitive edge by contracting with us for calibration, extra-cost upgrades, out-of-warranty repairs, and on-site education and training, as well as design, system integration, project management, and other professional engineering services. Experienced Agilent engineers and technicians worldwide can help you maximize your productivity, optimize the return on investment of your Agilent instruments and systems, and obtain dependable measurement accuracy for the life of those products.

www.agilent.com/find/emailupdates

Agilent Email Updates

Get the latest information on the products and applications you select.

Agilent T&M Software and Connectivity
Agilent Test Agilent's Test and Measurement software and connectivity products, solutions and developer network allows you to take time out of connecting your instruments to your computer with tools based on PC standards, so you can focus on your tasks, not on your connections. Visit www.agilent.com/find/connectivity for more information.

By internet, phone, or fax, get assistance with all your test & measurement needs

Online assistance:
www.agilent.com/find/assist

Phone or Fax

United States:
(tel) 1 800 452 4844

Canada:
(tel) 1 877 894 4414
(fax) (905) 282-6495

China:
(tel) 800 810 0189
(fax) 800 820 2816

Europe:
(tel) (31 20) 547 2323
(fax) (31 20) 547 2390

Japan:
(tel) (81) 426 56 7832
(fax) (81) 426 56 7840

Korea:
(tel) (82 2) 2004 5004
(fax) (82 2) 2004 5115

Latin America:
(tel) (305) 269 7500
(fax) (305) 269 7599

Taiwan:
(tel) 0800 047 866
(fax) 0800 286 331

Other Asia Pacific Countries:
(tel) (65) 6375 8100
(fax) (65) 6836 0252
Email: tm_asia@agilent.com

Product specifications and descriptions in this document subject to change without notice.

Printed in USA June 1, 2002
5968-9727EN

Agilent Technologies

Índice Alfabético

Análisis ..... xix, 1, 2, 4, 10, 12, 14, 20, 22, 23, 24, 28, 30, 65, 68, 70, 76, 77, 79, 82, 101, 103, 105, 113, 114, 118, 10

ATE ..... xiii, 1, 27, 28, 31, 33, 34, 37, 56, 66, 68, 69

C++ Builder .. xix, 4, 12, 32, 65, 113, 117, 118

CASE 10, 12, 14, 68

Ccasos de uso .... xiii, 21, 23, 24, 68, 70, 71, 77, 79, 82, 84

Clases ..... xiii, xiv, 8, 10, 12, 19, 21, 24, 28, 30, 68, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 88, 89, 90, 91, 92, 94, 95, 96, 97, 99, 100, 101, 107

CompactPCI 1, 34

Control de transferencia..... xii, 42, 43, 100

Controlador ..... xiv, xv, 41, 50, 54, 56, 61, 8

DAQ ..... xiii, 1, 28, 31

Desarrollo de software..... 2, 3, 4, 7, 10, 11, 12, 20, 21, 22, 23, 24, 114

Descomposición algorítmica ..... xi, xvii, 8, 10

Descomposición orientada a objetos xvii, 8, 10

Direccionamiento ..... xii, xiii, 45, 99

Diseño..... xiii, 12, 14, 24, 79, 94, 95, 96, 117

Emisor..... 41, 50, 54, 61

Especificaciones de procedimiento ..... xii, 47

Especificaciones eléctricas ..... xii, 41

Especificaciones funcionales .. xii, xiv, 41, 108

Especificaciones mecánicas38, 40, 98, 99, 105, 107, 113

Evaluación ..... xiv, 103

Fuente de alimentación 25

Generador de funciones .... xiv, 25, 28, 84, 106, 110, 111, 8

Gestión del bus..... xii, 41, 42, 43, 100, 108

GPIB ..... xii, xiv, xiii, xiv, xv, xvii, 1, 2, 30, 31, 32, 33, 34, 35, 37, 38, 40, 41, 42, 43, 44, 45, 46, 49, 50, 51, 53, 56, 58, 59, 65, 66, 67, 69, 70, 71, 73, 74, 77, 95, 98, 99, 100, 101, 103, 105, 106, 107, 108, 109, 110, 111, 113, 114, 115, 118, 1, 2, 4, 5, 7, 8, 10, 11, 1

GUI ..... xiii, 30, 31, 66, 67, 68, 70, 103

HP-IB 32, 33

HS 488 ..... 33, 34

IEC 625.1 33

IEEE 488 ..... i, xii, xiii, xv, xvii, xix, xxi, 1, 4, 5, 27, 31, 32, 33, 34, 37, 38, 39, 40, 41, 42, 44, 46, 48, 49, 50, 51, 53, 54, 55, 56, 58, 59, 60, 61, 62, 67, 69, 75, 98, 99, 100, 105, 107, 108, 110, 113, 114, 8, 9

IEEE 488.1 ..... 4, 32, 38, 55, 56, 58, 113

IEEE 488.2 ....xiii, 33, 34, 38, 48, 55, 56, 58, 59, 98, 113

IEEE 728 53

Implementación ..... xiii, 24, 98

Informe de estado..... xii, 58

Ingeniería de software..... xix, 14, 21, 54

Instrumentación electrónica ..... xii, xix, 1, 4, 27, 30, 31, 34, 65, 66, 68, 69, 103, 113

Instrumentación electrónica programable... xix

Instrumentación virtual ..... xix, 1, 4, 30, 113
Instrumentos convertidores de señales 28
Instrumentos de medida y visualización 28
Instrumentos generadores de señales 28
ISA 32, 35
LabView 1, 35, 103
LabWindows 1, 35
Lenguajes de programaciónxvii, xix, 1, 7, 10, 14, 19, 34
MATE 33
MC1.1 33
Método 8, 15, 17, 22, 25, 51, 66
Metodología de desarrollo ..... xix, 2, 20, 113
Mmodelado ..... xi, xii, xix, 2, 4, 15, 16, 17, 19, 20, 21, 22, 24, 25, 65, 67, 68, 70, 71, 76, 77, 79, 84, 105, 107, 113, 114, 115
Multímetro 25, 106, 107
MXI 1, 32
NI Spy 103
NI-488 ..... 34, 103, 118
Objetos ..... xi, xvii, xix, 2, 4, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 23, 24, 65, 68, 70, 78, 79, 80, 84, 86, 92, 96, 113, 114, 118
OMG 15
OMT 14, 15
OO.....xiii, xvii, 10, 12, 13, 15, 18, 19, 24, 119
OOA 12, 13, 14
OOD 12, 13, 14
OOP 12, 14
OOPL 12
OOSE 2, 14, 15
PCI 1, 33, 34
PCMCIA 35
Programación orientada a objetos 10, 70
Pruebas ..... xiii, 24, 101
PXI 1, 33, 34, 69
Receptor 41, 50, 54, 61
Requerimientos ..... xii, 2, 12, 16, 17, 18, 19, 20, 21, 22, 23, 24, 33, 38, 65, 68, 70, 79, 103
RUP ..... xi, xii, xiii, xix, xxi, 2, 4, 12, 14, 19, 20, 21, 23, 24, 25, 65, 67, 113, 114, 119
SCPI ..... xiii, 32, 33, 34, 38, 103, 106
SepiGPIB ..... xi, xii, xiii, xiv, xiii, xiv, xv, xvii, xix, xxi, 2, 4, 5, 24, 25, 65, 66, 67, 68, 70, 71, 73, 74, 75, 76, 77, 78, 79, 81, 82, 83, 89, 91, 98, 101, 103, 105, 106, 107, 108, 109, 110, 114, 115, 1, 2, 3, 5, 6, 7, 10, 11, 12
Simulación ..... xiv, 107, 110
Sistemas automatizados de medida xix
Sondeo ..... xiii, 37, 47, 48, 50, 51, 54, 56, 59, 60, 61, 62, 77, 89, 98, 100, 108, 109, 110, 5, 6, 9
Sondeo paralelo 47, 51, 61, 100, 5, 9
Sondeo serie 47, 48, 59, 100
Tecnología orientada a objeto ..... s .....xi, xix, 2, 4, 7, 10, 113, 114, 115
UML ..... xiii, xix, xxi, 2, 4, 12, 14, 15, 16, 17, 19, 20, 21, 22, 23, 24, 25, 65, 68, 70, 77, 78, 113, 114, 117, 118
USB 35, 114
VEE 1, 35, 103, 117
VI 30, 31
VXI 1, 31, 32, 33, 34, 35, 46, 69