CONTROL DE INSTRUMENTOS MEDIANTE EL BUS GPIB PROGRAMADO CON MATLAB

Memoria del proyecto Final de Carrera de Ingeniería Técnica de Telecomunicaciones (especialidad en Sistemas Electrónicos)

Realizado por:
Eduardo Amores Rubio

Dirigido por:
Rosana Rodríguez
Javier Martín

Bellaterra, 8 de julio de 2010

Los asignados, Rosana Rodríguez y Javier Martín,
Profesores de la Escuela Técnica Superior de Ingeniería de la UAB,

CERTIFICAN:

Que el trabajo que corresponde a esta memoria ha estado realizado bajo su dirección por:

Eduardo Amores Rubio

Y para que conste firman la presente:

Firmado:

Bellaterra, 8 de julio de 2010

El proyecto que se expone a continuación está dedicado al control de instrumentos mediante el bus de instrumentación GPIB programado con el software Matlab.

Está dividido en dos partes. La primera, será llevada a cabo en el laboratorio de docencia y el objetivo será controlar el osciloscopio y el generador de funciones. Como ejemplo del control realizado se desarrollará una aplicación que permitirá obtener el diagrama de Bode de módulo de cualquier sistema electrónico. La segunda parte será llevada a cabo en el laboratorio de investigación y el objetivo será controlar el analizador de semiconductores. En este caso, la aplicación desarrollada permitirá la realización de medidas para la caracterización de transistores.

Las aplicaciones de ambas partes estarán realizadas mediante una interfaz gráfica de usuario diseñada con la herramienta GUIDE de Matlab.

El projecte que s'exposa a continuació està dedicat al control d'instruments mitjançant el bus d'instrumentació GPIB programat amb el software Matlab.

Està dividit en tres parts. La primera, es durà a terme en el laboratori de docència i l'objectiu serà controlar l'oscil·loscopi i el generador de funcions. Com exemple del control realitzat es realitzarà una aplicació que permetrà obtenir el diagrama de Bode de mòdul de qualsevol sistema electrònic. La segona part es durà a terme en el laboratori d'investigació i l'objectiu serà controlar l'analitzador de semiconductors. En aquest cas, l'aplicació realitzada permetrà la realització de mesures per a la caracterització de transistors.

Les aplicacions de les dues parts seran realitzades mitjançant una interfaç gràfica d'usuari dissenyada amb l'eina GUIDE de Matlab.

The project that follows is dedicated to the control of instruments with the GPIB instrumentation bus programmed with Matlab software.

It's divided into two parts. The first will be made in the teaching laboratory and the objective is to control the oscilloscope and function generator. As an example of the monitoring executed, we will develop an application to obtain the Bode plot of any electronic system module. The second part will be made in the research laboratory and the objective is to control the semiconductor analyzer. In this case, the developed application will allow taking measurements for the characterization of transistors.

The applications of both parts will be made by a graphical user interface designed with the Matlab's GUIDE tool.

ÍNDICE:

CAPÍTULO I: INTRODUCCIÓN

1. PRESENTACIÓN DEL PROBLEMA

Este proyecto tiene como finalidad realizar el control de instrumentos electrónicos mediante el bus de instrumentación GPIB programado con el software Matlab. El proyecto estará dividido en dos partes. La primera, está dedicada al control de instrumentos del laboratorio de docencia mientras que la segunda, se refiere al control de instrumentos del laboratorio de investigación en el que se realiza la caracterización eléctrica de dispositivos.

2. OBJETIVOS

- **Laboratorio docente**

El objetivo de esta primera parte del proyecto es el control del osciloscopio y el generador de funciones del laboratorio docente mediante la programación del bus GPIB con el software Matlab. Como ejemplo de aplicación del control de estos instrumentos se realizará la representación del diagrama de Bode de un sistema. La configuración por parte del usuario de todos los parámetros se realizará mediante una interfaz gráfica.

Conseguir este objetivo servirá como base para la consecución del objetivo de la segunda parte.

- **Laboratorio de investigación**

El objetivo de esta segunda parte del proyecto es controlar el analizador de semiconductores de un laboratorio de investigación, para poder configurar medidas para la caracterización de dispositivos mediante una interfaz gráfica de usuario.

3. CONTROL DE INSTRUMENTOS

En este apartado se describirán las ideas básicas sobre el control de instrumentos así como los buses de instrumentación más utilizados [1].

3.1 Ideas teóricas básicas sobre el control de instrumentos

En cualquier actividad relacionada con la ciencia y la tecnología surge la necesidad de medir variables físicas. Un sistema de adquisición de datos (SAD) es el instrumento del que nos servimos para obtener información de un determinado proceso.

Los sistemas de adquisición de datos pueden estar basados en instrumentos de adquisición independientes (Figura I.1), o en sistemas modulares (Figura I.2). El control de instrumentos independientes se puede llevar a cabo de forma autónoma desde el panel frontal o bien realizarse mediante un computador que permite la conexión de varios instrumentos a través de un bus de comunicación. Para el caso de instrumentos

modulares, la interconexión entre los módulos del sistema se realiza a través de un bus de conexión local de altas prestaciones como el VXI o PXI. Este bus adopta la forma de un panel posterior al que se conectan los distintos módulos que componen el sistema de adquisición montado en un rack. Estos sistemas pueden conectarse a su vez a otros racks de módulos o a computadores, permitiendo así soluciones muy versátiles.

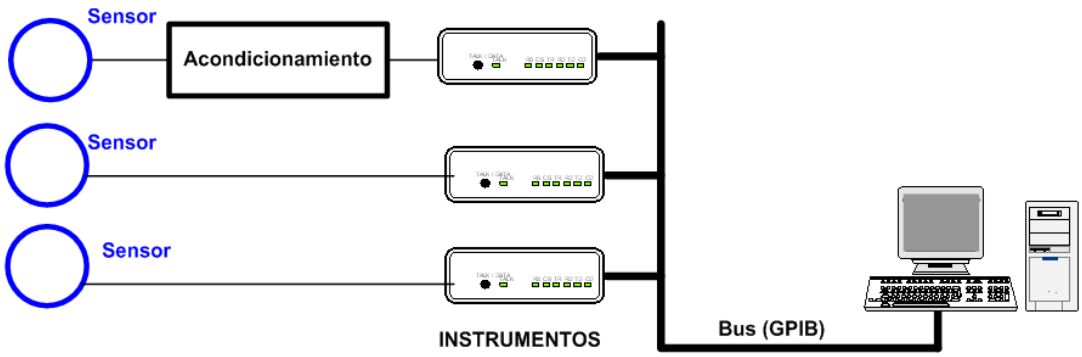


Figura I.1 - Sistema de adquisición de datos basado en instrumentos de adquisición de datos independientes

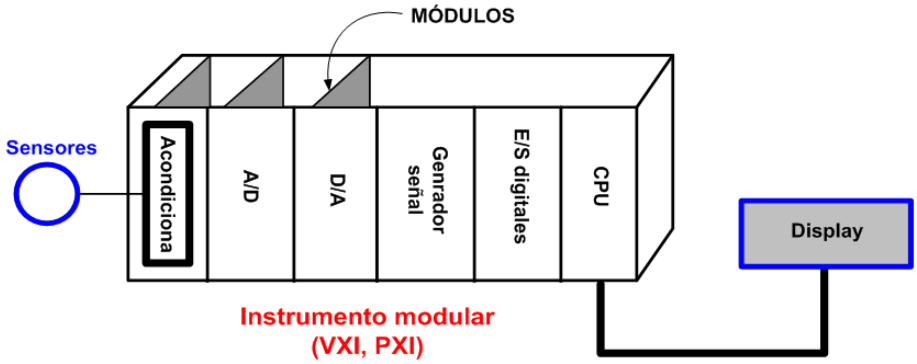


Figura I.2 - Sistema modular de adquisición de datos

Para nuestro proyecto, utilizaremos un sistema de adquisición de datos basado en instrumentos de adquisición, es decir, en instrumentos autónomos e independientes del bus de un computador pero con capacidad de conectarse a éste a través de buses de instrumentación.

3.2 Buses de instrumentación

Los sistemas de adquisición de datos basados en instrumentos independientes pueden funcionar de forma autónoma o bien conectarse a un computador y a otros instrumentos a través de un bus de comunicación [2] [4].

En los siguientes apartados se describirán las características más importantes de los buses de instrumentación más comúnmente utilizados, explicando más detalladamente el bus GPIB, por ser el utilizado en este proyecto.

3.2.1 VXI

Físicamente, VXI (VMEbus eXtensión for Instrumentation) consiste en un chasis plano posterior sobre el que se conectan unos módulos en forma de tarjetas enchufables; mediante este sistema puede configurarse una solución modular de instrumentación (Figura I.2).

VXI se utiliza fundamentalmente cuando se necesita un sistema de adquisición de datos fiable, de altas prestaciones, con gran número de variables a capturar y con posibilidades de ampliación. En general, para la adquisición de pocos canales (hasta 20) una tarjeta de adquisición de datos puede ser suficiente. Sin embargo, para un número de canales superior (hasta 100) puede utilizarse un instrumento externo de adquisición independiente. Cuando las necesidades aumentan, VXI puede ser la mejor solución. En general, el coste de los módulos es menor que el de un instrumento independiente y su potencia es superior a la de una tarjeta de adquisición. Las ventajas del sistema pueden resumirse en:

- **Sistema abierto (IEEE-1014), flexible y modular.** La arquitectura está soportada por numerosos fabricantes de instrumentos siendo posible elegir entre cientos de productos para satisfacer las más variadas necesidades (convertidores A/D, acondicionadores de señal, salidas analógicas y digitales, generadores de formas de onda, etc.). Es posible añadir más módulos a un rack para dotar al sistema de mayores prestaciones, conforme lo demande nuestro sistema de adquisición de datos. En general, el VXI es una solución muy adecuada para un sistema con un número de canales comprendidos entre 10 y 100.

- **Alto rendimiento** al estar basado en un bus de 32 bits de altas prestaciones. Puede transferir datos a una velocidad teórica de 10MB/s (superior a GPIB, que tiene 1MB/s). En caso de que se desee comunicación entre tarjetas, VXI dispone de un bus local que puede llegar a 100 MB/s.

- **Fiable y robusto:** Se aseguran medidas precisas y fiables gracias a las protecciones y al apantallamiento que establece límites estrictos sobre las interferencias por conducción y radiación

- **Tamaño reducido** que favorece la movilidad, portabilidad y su proximidad al proceso a controlar.

- **Altas velocidades de adquisición** y mecanismos de temporización y disparo entre todas las tarjetas del rack.

- **Configuración y programación sencilla** mediante varios lenguajes de programación.

3.2.2 PXI, PCI y PCI-Express:

Otra alternativa de bus muy difundida para instrumentación modular es el PXI. PXI utiliza una variante del popular bus PCI, muy utilizado en los computadores personales, para la interconexión de los módulos. A diferencia del bus PCI, esta arquitectura dispone de unas características de sincronización avanzadas, pudiendo trasmitirse por el bus señales digitales (TTL) a alta velocidad e incluso señales analógicas entre los distintos módulos.

En la actualidad, el más utilizado es el PCI-Express, la evolución del bus PCI, que es una reformulación radicalmente distinta de éste aunque mantiene totalmente la compatibilidad software. PCI-Express es un bus de red de comunicaciones de alta velocidad en distancias cortas y con baja latencia para interconectar dispositivos y tarjetas entre sí dentro de un chasis. Sus características más importantes son:

- Es una red punto a punto de conmutación de paquetes. La red se construye mediante enlaces punto a punto full-dúplex + conmutadores. Cada enlace está compuesto por uno o varios lanes, que son dos pares de hilos, un par de transmisión y otro de recepción. Cada par es un canal símplex en el que la señal de tensión transmitida es diferencial (LVDS - Low Voltage Diferential Signaling), lo que mejora la inmunidad al ruido. Igual que un router de Internet, el conmutador encamina las tramas entre las distintas tarjetas conectadas al bus, cada una de las cuales tiene su propia dirección de red.
- La arquitectura de la red está divida en capas. El transceptor PCI-Express implementa en hardware los 3 niveles inferiores del modelo OSI. Estos niveles son: Nivel de transacción (Transaction Layer), Nivel de enlace (Link layer), Nivel físico (Physical Layer).
- La transmisión es serie.
- La velocidad del bus es escalable añadiendo más lanes al enlace. Cada Lane (hasta un máximo de 32) proporciona una velocidad bruta de 250MB/s/sentido.
- Calidad de servicio (QoS). Es capaz de diferenciar distintos tipos de tráfico y de ofrecer cierto nivel de calidad de servicio.
- Compatibilidad software con PCI. La interfaz software del bus permanece igual, no hay que cambiar los drivers del sistema operativo ni el software de sistema al cambiar de bus, todo queda exactamente igual.

Cabe tener en cuenta que las prestaciones del bus se degradan rápidamente al aumentar el número de usuarios/dispositivos.

3.2.3 RS-232

Se trata de un antiguo estándar de comunicación serie que se diseñó para la comunicación entre un equipo terminal de datos (DTE), como un computador y un equipo de comunicaciones de datos (DCE), como un módem. Es un enlace de tipo full dúplex y punto a punto. La distancia máxima que abarca sin ningún circuito de ampliación es de aproximadamente 15m (en la práctica puede llegar a funcionar hasta con distancias de 100m) y su velocidad típica ronda los 19000 baudios aunque puede

ser superior. Este tipo de enlace dispone de unas líneas de datos y unas líneas de control sobre las que se implementa el protocolo de comunicación (ciertas señales presentes en el conector que permiten el bloqueo de la transmisión o la recepción). La norma básica se ocupa de las especificaciones físicas del conector, los niveles de tensión de las señales y las señales de protocolo. Se suele utilizar un conector de 25 pines. Cuando no son necesarias todas las señales se puede adoptar un conector de 9 pines. Los niveles lógicos "0" y "1" se representan por los niveles físicos de tensión que muestra la Figura I.3:

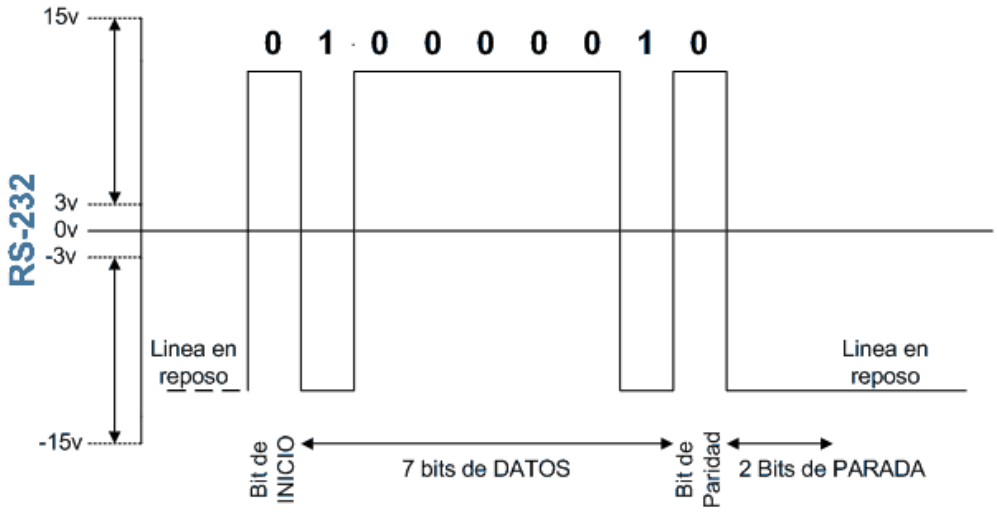


Figura I.3 - Niveles de tensión RS-232C

Cualquiera que sea el tipo de transmisión es necesario que el receptor se sincronice para saber en todo momento dónde comienza la transmisión de un bit, un carácter o un bloque. Cada carácter va precedido de un bit de inicio (bit de Start) y finaliza con 1 ó 2 bits de parada (bits de Stop), que garantizan la sincronización del receptor y permiten el reconocimiento del comienzo y el final del carácter (Figura I.4).

3.2.4 RS-422

Se trata de otro estándar de comunicación serie full-dúplex que utiliza señales diferenciales. Es una interfaz muy apropiada para aplicaciones industriales que conectan un maestro con varios terminales. También permite la comunicación punto a punto entre dos nodos utilizando, generalmente, un par de cables trenzados para cada línea de señal (4 hilos). La transmisión en modo diferencial presenta como ventaja principal su inmunidad al ruido electromagnético (de modo común). Los unos y ceros lógicos se establecen en función de la diferencia de tensión entre ambos conductores. Permite establecer una mayor distancia de conexión, respecto a la interfaz serie RS-232, sobre todo, en entornos ruidosos como el industrial.

Las distancias y frecuencias que se utilizan en este bus son de 1200 a 1500m a 100 Kbits/s o 50m a 10 Mbits/s. Las líneas deben cargarse en los extremos con resistencia de terminación de línea (120 Ω generalmente). El propósito de la resistencia de terminación es prevenir la reflexión de los datos en el fin de línea.

3.2.5 RS-485

Este enlace es uno de los más utilizados en la industria. Se trata de un enlace serie, con transmisión de señales en modo diferencial, multipunto. Está basado en la interfaz RS-422 pero utiliza sólo un par trenzado para la comunicación. Permite usar una topología de bus, conectando los dispositivos en paralelo a los dos conductores, aunque desde un punto de vista lógico puede después organizarse como un anillo, estrella u otro tipo. Presenta una alta inmunidad al ruido y se pueden crear redes multipunto maestro-esclavo de forma muy sencilla. La figura I.4 muestra los niveles de tensión de la interfaz RS-485:

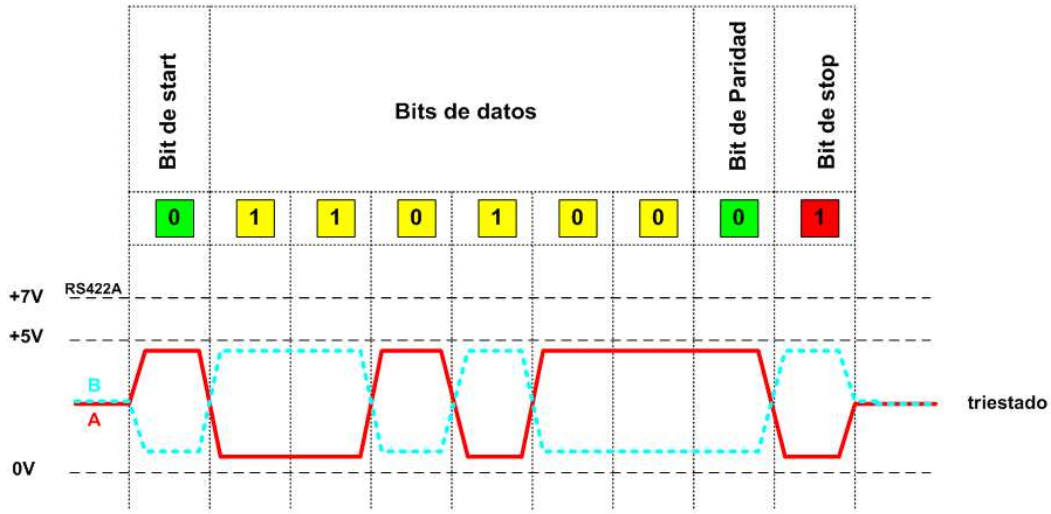


Figura I.4 - Niveles de tensión de las inverfaces RS-422 y RS-485

Normalmente, el número de nodos está limitado a 32 por razones de carga, pero puede ampliarse a más si se utilizan repetidores. Para aumentar la inmunidad al ruido se suele forzar el estado de la línea a un estado inactivo, si no existe un driver activo en la red, con un circuito formado por dos resistencias. Sin este circuito, es posible que la línea se vea afectada por ruidos que pueden activar los receptores y causar graves problemas en las comunicaciones. Las características del enlace en cuanto a niveles lógicos, distancia y velocidad de transmisión son idénticas al enlace RS-422.

La tabla I.1 resume las características de los tres tipos de enlaces:

	RS-232	RS-422	RS-485
Tipo de línea	Desbalanceada	Balanceada	Balanceada
Nº máximo de dispositivos	1	1	32
Nº máximo de receptores	1	32	32
Longitud máxima (m)	15	1200	1200
Velocidad máxima	20 Kb/s	10 Mb/s	10 Mb/s

Tabla I.1 – Características de los tres tipos de enlaces: RS-232, RS-422, RS-485

3.2.6 USB: Universal Serial Bus

El diseño del USB tenía como objetivo eliminar la necesidad de adquirir tarjetas separadas para poner en los puertos bus ISA o PCI, y mejorar las capacidades plug-and-play permitiendo a esos dispositivos ser conectados o desconectados al sistema sin necesidad de reiniciar. Sin embargo, en aplicaciones donde se necesita ancho de banda para grandes transferencias de datos, los buses PCI o PCI-Express salen ganando. Igualmente sucede si la aplicación requiere de robustez industrial. A favor del bus USB, cabe decir que cuando se conecta un nuevo dispositivo, el servidor lo enumera y agrega el software necesario para que pueda funcionar (esto dependerá ciertamente del sistema operativo que esté usando el computador).

Los dispositivos USB se clasifican en cuatro tipos según su velocidad de transferencia de datos:

- **Baja velocidad (1.0):** tasa de transferencia de hasta 1,5Mbps. Utilizado en su mayor parte por dispositivos de interfaz humana como los teclados y los ratones.
- **Velocidad completa (1.1):** tasa de transferencia de hasta 1,5MB/s. ésta fue la más rápida antes de la especificación USB 2.0, y muchos dispositivos fabricados en la actualidad trabajan a esta velocidad.
- **Alta velocidad (2.0):** tasa de transferencia de hasta 60MB/s pero por lo general de 16MB/s. el cable USB 2.0 dispone de cuatro líneas, un par para datos, una de corriente y una toma tierra.
- **Súper alta velocidad (3.0):** actualmente se encuentra en fase experimental y tiene una tasa de transferencia de hasta 600MB/s. Esta especificación será diez veces más veloz que la anterior 2.0, se presentará a finales de 2010 por Intel y está pensada para ser totalmente compatible con todos los estándares anteriores.

Las señales del USB se transmiten en un cable de par trenzado con impedancia característica de 90 Ω ± 15%, cuyos hilos se denominan D+ y D-. Éstos, colectivamente, utilizan señalización diferencial en full dúplex para combatir los efectos del ruido electromagnético en enlaces largos. Los niveles de transmisión de la señal varían de 0 a 0.3 V para bajos (ceros) y de 2.8 a 3.6 V para altos (unos) en las versiones 1.0 y 1.1, y en ± 400mV en alta velocidad (2.0). Las señales eléctricas del bus se muestran a continuación (Tabla I.2):

Pin	Nombre	Color	Descripción
1	VCC	Rojo	+5 V
2	D-	Blanco	Data -
3	D+	Verde	Data +
4	ID	Ninguno	Permite la distinción de Micro-A y Micro-B Tipo A: conectado a tierra Tipo B: no conectado
5	GND	Negro	Señal tierra

Tabla I.2 – Señales eléctricas del bus USB

3.2.7 CAN (Controller Area Network):

Bus que fue diseñado originalmente para su aplicación en el sector de la automoción aunque debido a sus buenas características, es muy robusto y fácil de implementar, cada vez se usa más en la industria. El protocolo de comunicaciones CAN proporciona los siguientes beneficios:

- Es un protocolo de comunicaciones normalizado, con lo que se simplifica y economiza la tarea de comunicar subsistemas de diferentes fabricantes sobre una red común o bus.
- El procesador anfitrión (*host*) delega la carga de comunicaciones a un periférico inteligente, por lo tanto el procesador anfitrión dispone de mayor tiempo para ejecutar sus propias tareas.
- Al ser una red multiplexada, reduce considerablemente el cableado y elimina las conexiones punto a punto, excepto en los enganches.

CAN es un protocolo orientado a mensajes, es decir la información que se va a intercambiar se descompone en mensajes, a los cuales se les asigna un identificador y se encapsulan en tramas para su transmisión. Cada mensaje tiene un identificador único dentro de la red, con el cual los nodos deciden aceptar o no dicho mensaje. Dentro de sus principales características se encuentran:

- Prioridad de mensajes
- Flexibilidad en la configuración
- Recepción por multidifusión (*multicast*) con sincronización de tiempos
- Sistema robusto en cuanto a consistencia de datos
- Sistema multimaestro
- Detección y señalización de errores
- Retransmisión automática de tramas erróneas
- Distinción entre errores temporales y fallas permanentes de los nodos de la red, y desconexión autónoma de nodos defectuosos

Protocolo de comunicaciones CAN

CAN es un protocolo de comunicaciones serie que soporta control distribuido en tiempo real con un alto nivel de seguridad y multiplexación. Soporta velocidades de transferencia de datos de hasta 1 Mbps.

De acuerdo al modelo de referencia OSI, la arquitectura de protocolos CAN incluye tres capas: física, de enlace de datos y aplicación, además de una capa especial para gestión y control del nodo llamada capa de supervisor.

- **Capa de supervisor:** La sustitución del cableado convencional por un sistema de bus serie presenta el problema de que un nodo defectuoso puede bloquear el funcionamiento del sistema completo. Cada nodo activo transmite una bandera de error cuando detecta algún tipo de error y puede ocasionar que un nodo defectuoso pueda acaparar el medio físico. Para eliminar este riesgo el protocolo CAN define un mecanismo autónomo para detectar y desconectar un nodo defectuoso del bus. Dicho mecanismo se conoce como aislamiento de fallos.

Cuando un nodo necesita enviar información a través de una red CAN, puede ocurrir que varios nodos intenten transmitir simultáneamente. CAN resuelve lo anterior al asignar prioridades mediante el identificador de cada mensaje. El identificador con el menor número es el que tiene mayor prioridad. El método de acceso al medio utilizado es el de Acceso Múltiple por Detección de Portadora, con Detección de Colisiones y Arbitraje por Prioridad de Mensaje (CSMA/CD).

En cuanto a la detección y manejo de errores, un controlador CAN cuenta con la capacidad de detectar y manejar los errores que surjan en una red. Todo error detectado por un nodo, se notifica inmediatamente al resto de los nodos.

3.2.8 GPIB

En esta sección se descubrirá el bus GPIB, su evolución histórica, elementos que lo constituyen así como las características más importantes que presenta.

3.2.8.1 Evolución histórica y Situación Actual

El bus GPIB tuvo sus orígenes en el Hewlett-Packard Instrument Bus (HP-IB), que es un estándar bus de datos digital de corto rango desarrollado por Hewlett- Packard en los años 1970 para conectar dispositivos de prueba y medida (por ejemplo multímetros, osciloscopios, etc.) con dispositivos que los controlen como un ordenador (Figura I.5). En 1978 el bus fue estandarizado por el Institute of Electrical and Electronics Engineers (IEEE) como el IEEE-488 (488.1). El IEEE-488 permite que 15 dispositivos inteligentes compartan un simple bus, con el dispositivo más lento determinando la velocidad de transferencia. La máxima velocidad de transmisión es aproximadamente de 1 Mbps. Las 16 líneas que componen el bus están agrupadas en tres grupos de acuerdo con sus funciones: bus de datos, bus de control de transferencia de datos y bus general. Algunas de ellas tienen retornos de corrientes comunes y otras tienen un retorno propio, lo que provoca un aumento del número de líneas totales. Además del IEEE otros comités han adoptado el HP-IB. El American National Standards Institute (ANSI) lo llama ANSI Standard MC 1.1, y para la International Electrotechnical Commission (IEC) es el IEC Publication 625-1. En junio de 1987 el IEEE aprobó una revisión del estándar para instrumentos programables llamado IEEE-488-1987 (488.2). En él se definieron códigos, formatos, protocolos y comandos comunes para todos los instrumentos, como por ejemplo la adopción del formato de comandos SCPI (Standard Commands for Programmable Instruments) que estructura las órdenes a los dispositivos de forma coherente, permitiendo, la sustitución de instrumentos de distintos fabricantes con mínimos cambios.

En paralelo a la evolución del bus GPIB la International Electronic Comisión (IEC), responsable de la estandarización fuera de los Estados Unidos, aprobó el estándar IEC625.1. Este protocolo empleaba un conector DSUB de 25 pines, a diferencia del IEEE-488.1 que empleaba uno similar al Centronics pero de 24 terminales. Ésta última versión es la que se emplea hoy día, pero siguen existiendo adaptadores de 25 a 24 terminales, con el fin de evitar la obsolescencia de determinados instrumentos.

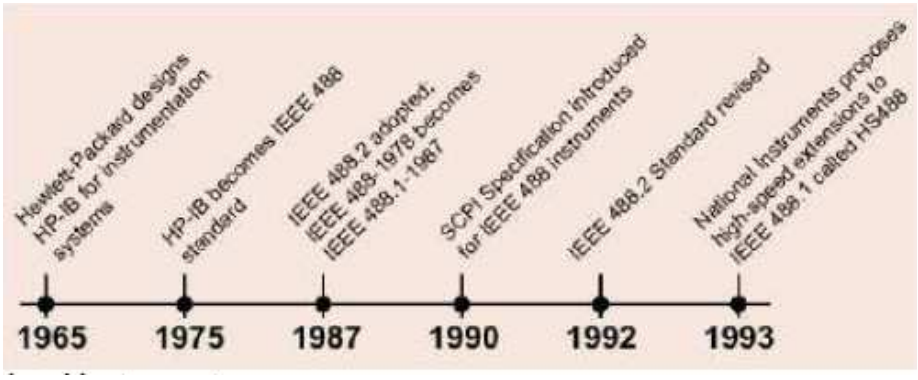


Figura I.5 - Evolución del estándar IEEE 488

3.2.8.2 Características eléctricas del bus GPIB

El bus de transmisión de datos de GPIB es de 8 bits en paralelo, y lógica negativa con niveles TTL estándar (cierto si el voltaje es < 0.8 V y falso si el voltaje es > 2.0 V). El cable utilizado y los conectores se muestran en las Figuras I.6 y I.7 respectivamente [3].

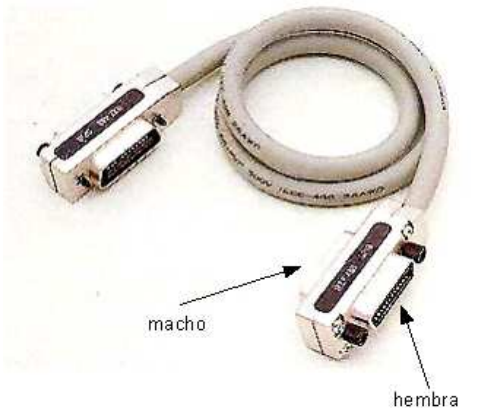
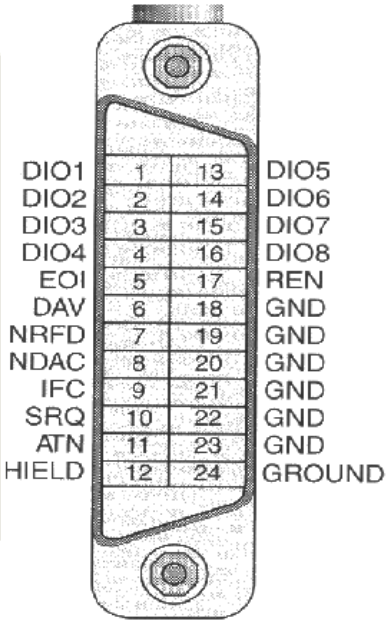


Figura I.6 – Imagen del cable utilizado en el bus GPIB

El bus consta de 24 pines, repartidos de la siguiente forma:

- 8 líneas de transmisión de datos (DIO1-DIO8)
- 3 líneas para el control asíncrono de la comunicación (NRFD, NDAC y NRDAV). Mediante estas líneas se verifica la correcta transmisión de los datos, que es una de las fortalezas del GPIB
- 5 líneas que gestionan la transmisión de comandos (ATN, IFC, REN, SRQ y EOI)
- El resto componen las tierras de las diferentes líneas

DATA LINES	Pin No.
DIO1	1
DIO2	2
DIO3	3
DIO4	4
DIO5	13
DIO6	14
DIO7	15
DIO8	16



MANAGEMENT LINES	Pin No.
IFC	9
REN	17
ATN	11
SRQ	10
EOI	5

HANDSHAKE LINES	Pin No.
DAV	6
NRFD	7
NDAC	8

Figura I.7 - Conector GPIB y asignación de señales

- Para que el bus GPIB alcance la velocidad de transmisión para la que fue diseñado (hasta 8 Mbytes/s), deben cumplirse los siguientes requisitos:

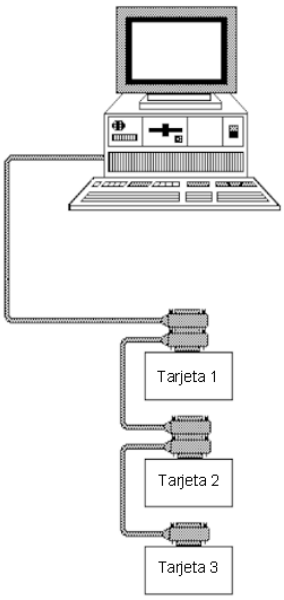
  - Puede haber un máximo de 15 dispositivos conectados al bus, y al menos dos tercios de ellos deben estar encendidos
  - La separación máxima entre dos dispositivos es 4m, y la separación promedio en toda la red debe ser menor de 2m
  - La longitud total de la red no debe exceder los 20m

La tabla I.3 muestra los valores eléctricos del bus:

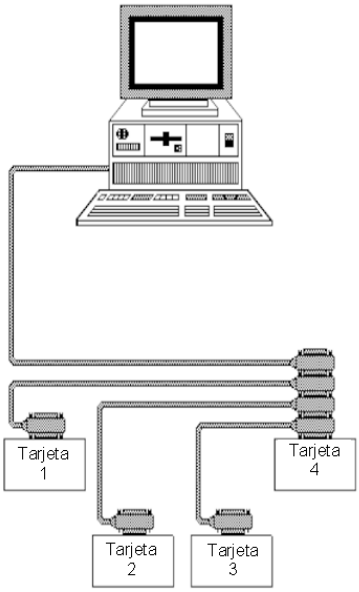
Tipo de Señal	Valor Digital
Entrada de Alto Voltaje	V_{IH} = 3.4 volts nominal, 2.4 volts mínimo
Entrada de Bajo Voltaje	V_{IL} = 0.22 volts nominal, 0.4 volts máximo
Entrada de Corriente Alta	I_{IH} = 2.5mA máximo
Entrada de Corriente Baja	V_{IL} = -3.2mA máximo
Salida de Alto Voltaje	V_{OH} = 3.4 volts nominal, 2.5 volts mínimo
Salida de Bajo Voltaje	V_{OL} = 0.22 volts nominal, 0.5 volts máximo
Salida de Corriente Alta	I_{OH} = -5.2mA máximo
Salida de Corriente Baja	I_{OL} = 48mA máximo

Tabla I.3 - Características Eléctricas del bus GPIB

En la Figura I.8, se muestran las diferentes configuraciones que se pueden realizar utilizando el bus GPIB: topología en bus, en estrella y híbrida.



a) Topología en bus



b) Topología en estrella

c) Configuración híbrida

Figura I.8 – Configuraciones que puede adoptar el bus GPIB: a) Topología en bus b) Topología en estrella c) Configuración Híbrida (Estrella y Bus)

3.2.8.3 Características funcionales del bus GPIB

– Elementos que lo componen

Son tres los elementos que componen un sistema GPIB: receptores, transmisores y controladores. Cada uno cumple funciones distintas, lo que no significa que un elemento no pueda, en un momento dado, cumplir las funciones de los otros dos elementos. A continuación se explica cada componente:

Un **transmisor** (talker) envía mensajes de datos a uno o más **receptores** (listener)**.** El **controlador** (controller) administra el flujo de información en el bus GPIB enviando comandos a todos los dispositivos. Un analizador de espectro, por ejemplo, es un receptor y transmisor también. GPIB es como un bus ordinario de un computador, excepto que un computador tiene sus tarjetas interconectadas en un mismo plano,

mientras que GPIB tiene los dispositivos separados interconectados por cables standards.

El papel del controlador GPIB es comparado con el papel de la CPU en un computador, pero una mejor analogía es la comparación del controlador con la central de conmutación de un sistema telefónico de una ciudad. La central conmutadora (controlador) monitorea la red de comunicaciones (GPIB). Cuando la central (controlador) nota que un abonado (dispositivo) quiere realizar una llamada (enviar un mensaje de datos), conecta el que llama (transmisor) con el que recibe la llamada (receptor).

Hay que tener en cuenta que se debe asignar a cada dispositivo GPIB una dirección exclusiva dentro del bus. Las direcciones GPIB (Tabla I.4) constan de dos partes: una dirección primaria (DP) y una dirección secundaria (DS). La DP es un número entero, comprendido entre 0 y 30, que indica el número de dirección GPIB asignada al dispositivo. La DS, en cambio, configura si el dispositivo conectado se encuentra en modo “talker” (TA) o “listener” (LA).

Posición del bit	7	6	5	4	3	2	1	0
Significado	0	TA	LA	Dirección primaria GPIB (de 0 a 30)				

Tabla I.4- Bits de las direcciones del bus GPIB

El controlador habilita un transmisor y un receptor antes que el transmisor pueda enviar sus mensajes al receptor. Después de que el mensaje es transmitido, el controlador puede direccionar otros transmisores y receptores. Algunas configuraciones GPIB no requieren un controlador. Por ejemplo, cuando un dispositivo que es siempre un transmisor es conectado a uno o más dispositivos que únicamente son receptores. Por tanto, un controlador es necesario cuando el transmisor o receptor direccionado debe ser cambiado. Esta función es usualmente realizada por un computador.

Aunque puede haber múltiples controladores en un sistema GPIB, en cualquier momento solo un controlador es el controller-in-charge (CIC), es decir, controlador encargado o principal. A continuación, en la figura I.9, se muestran las líneas que unen los elementos funcionales de un sistema GPIB:

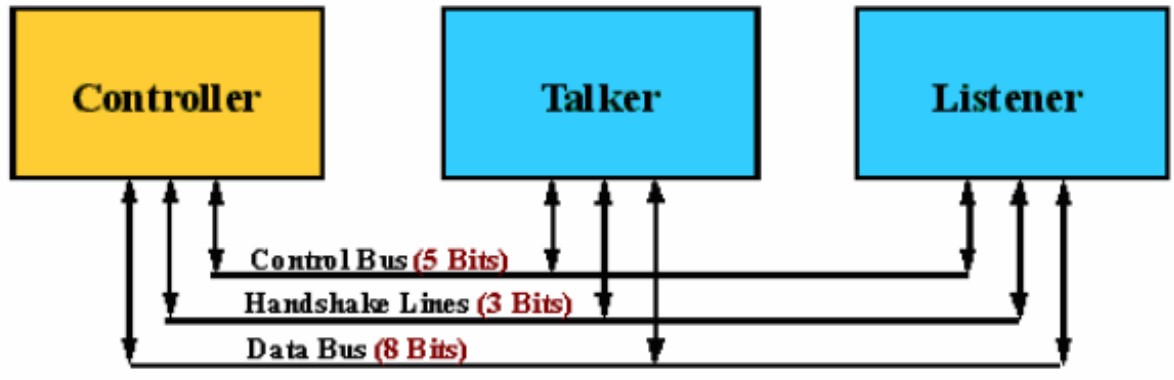


Figura I.9 - Elementos funcionales de un sistema GPIB

A continuación, se muestra una tabla resumen de las características del bus GPIB (Tabla I.5):

Especificación	Concepto	Descripción
Mecánicas	topología	Estrella o bus
	longitud	2m entre instrumentos, máximo 20m
	conector	Instrumento (Hembra) Cable (apilable: macho-hembra)
Eléctricas	"1" lógico	< 0,8V
	"0" lógico	> 2V
Funcionales	Tipos de instrumentos	*Controller* (controlador) *Talker* (emisor) *Listener* (receptor)
	Nº max de Instrumentos	15
	Nº instrumentos activos	> 2/3
	Velocidad	1MBps
	Dirección en instrumento	Conmutadores o memoria pasiva

Tabla I.5 - Resumen características bus GPIB

– Tipos de Mensajes GPIB

Los dispositivos GPIB se comunican con otros dispositivos GPIB enviando dos tipos de mensajes: mensajes dependientes de dispositivo y mensajes de gestión a través de la interfaz.

- **Mensajes de dispositivo:** A menudo llamados datos o mensajes de datos, contienen información de un dispositivo específico, como instrucciones de programación, resultados de mediciones, estatus del instrumento y archivos de datos.

- **Mensajes de Gestión:** Controlan los mensajes a través del bus. Usualmente llamados comandos o mensajes de comandos. Los mensajes de gestión funcionan como las funciones que inicializan el bus, habilitan y deshabilitan dispositivos, y ajustan los dispositivos en modo de programación remota o local. Los comandos más comunes se enuncian en la siguiente tabla:

Orden	Nombre de la orden	Función
*CLS	Clear Status Command	*-Despeja el registro de estado y los registros de incidencia.*
*ESE	Event Status Enable Command	*-Habilita bits del registro de habilitación de incidencias*
*ESE?	Event Status Enable Query	*-Interroga el registró de habilitación de Incidencias estándar.*
*ESR?	Event Status Register Query	*-Interroga el registro de Incidencias estándar.*
*IDN	Identification Query	*-Identifica tipo de instrumento y versión software.*
*LRN?	Learn Device Setup Query	*-Requiere el estado actual del equipo.*
*OPC	Operation Complete Command	*-Fija el bit de "Operación completa" del registro estándar.*
*OPC?	Operation Complete Query	*-Responde con "1" si se han ejecutado ordenes previas.*
*OPT?	Option Identification Query	*-Requiere la opción instalada en el equipo.*
*RCL	Recall Command	*-Restaura el estado del equipo del registro save/recall.*
*RST	Reset Command	*-Sitúa al equipo en el estado básico de referencia.*
*SAV	Save Command	*-Almacena el estado actual en un registro save/recall.*
*SRE	Service Request Enable Command	*- Habilita los bits del registro de habilitación de Byte de estado.*
*SRE?	Service Request Enable Query	*- Requiere el contenido del registro SER de habilitación del byte de estado*
*STB?	Read Status Byte Query	*- Requiere el estado del registro resumido del Byte de estado.*
*TRG	Trigger Command	*- Arranca o dispara la operación del equipo de forma remota.*
*TST?	Selft-Test Query	*-Requiere el resultado del autotest del equipo.*
*WAI	Wait-to-Continue Command	*- Espera a que se realicen todas las operaciones pendientes.*

Tabla I.6 - Comandos GPIB más comunes

– Protocolo de Transferencia del bus GPIB

La interfaz GPIB consta de 16 líneas. Las 8 restantes del bus corresponden a líneas de retorno a tierra. De las 16 líneas, 8 son de datos (información que queremos transmitir) y 8 para comandos (mensajes de control y estados de los dispositivos). De estas últimas 8 líneas, 3 son para el control de transferencia de datos (handshake) y 5 para el control general de la interfaz. La figura I.10 muestra la estructura de las líneas del bus:

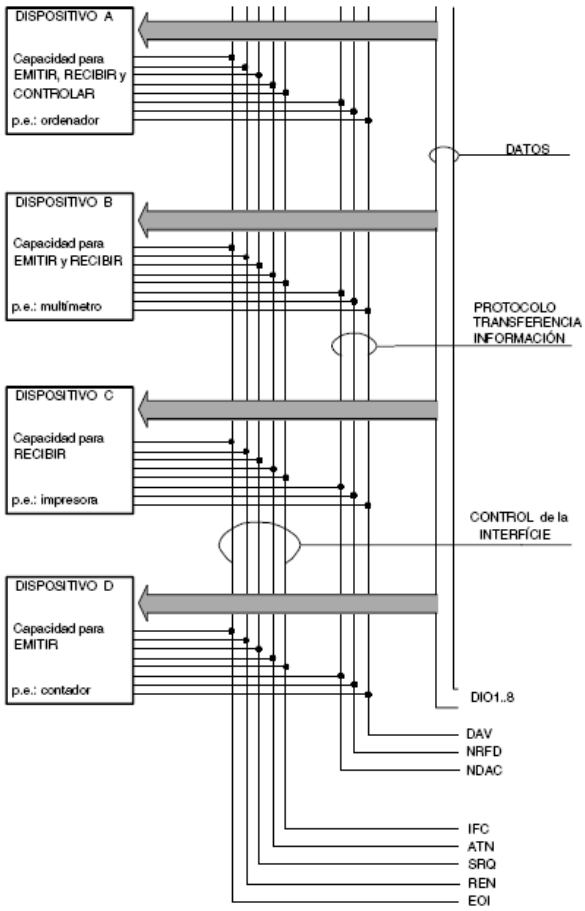


Figura I.10 - Líneas y señales del bus GPIB

- **Líneas de Datos**

Las 8 líneas de datos DIO1-DIO8 pueden transportar tanto datos como comandos. El estado de la línea ATN determina si la información son datos o comandos. Si está a nivel bajo son órdenes o direcciones, y si está a nivel alto son datos. Todos los comandos y la mayoría de datos emplean 7 bits codificados en ASCII o ISO, en este último caso el octavo bit se emplea para paridad o no se emplea.

- **Líneas de Control de Transferencia HandShake**

El estándar IEEE 488.2 establece un "handshake" de 3 líneas de control de la transferencia: DAV (Data Valid), NRFD (Not Ready For Data) y NDAC (Not Data Accepted). De esta forma, no se transmiten datos hasta que no esté listo el receptor (listener) más lento, y queda asegurado que la transmisión sea lo suficientemente lenta como para que al receptor más lento le dé tiempo a aceptar el dato.

La línea NRFD es controlada por cada receptor e indica si cada uno de ellos no está listo (nivel bajo) o lo está (nivel alto) para recibir datos. La línea DAV es controlada por el transmisor e indica si los datos en las líneas de datos (DIO) son correctos y, en consecuencia, pueden ser aceptados por los receptores. Finalmente, la línea NDAC es controlada por cada receptor para indicar que no ha recibido los datos (nivel bajo) o que los ha recibido (nivel alto).

La Figura I.11 muestra el diagrama de tiempos de operación. En principio, el transmisor comprueba que las líneas NRFD y NDAC están a nivel bajo. La primera indica que no todos los receptores están listos para recibir datos y la segunda indica que no han aceptado ningún nuevo byte. La línea NRFD no pasa a nivel alto hasta que todos los receptores están listos. Una vez que el transmisor ha detectado que la línea NRFD está a nivel alto y transcurre cierto retardo, necesario para dar tiempo a estabilizar los niveles de los datos que envía a los receptores, pone la línea DAV a nivel bajo indicando que los datos que envía son válidos (instante 3). Se transfiere así un byte de datos. El receptor más rápido pone la línea NRFD a nivel bajo con el fin de indicar que no está listo para recibir otro byte (instante 4). Los demás harán lo mismo cada uno a su ritmo. Es decir, el receptor más rápido indica al equipo que no mande más información porque él ha tomado ya la que había y tiene que aceptarla o procesarla (es posible que se requiera de él una respuesta).

Finalmente, los receptores van aceptando el byte poniendo a nivel alto sus líneas NDAC. Cuando todos han aceptado los datos (instante 6), la línea pasa a nivel alto, el transmisor lo detecta y pone la línea DAV a nivel alto para indicar que ya no valen los datos (instante 7). El primer receptor que detecta que la línea DAV ha pasado a nivel alto pone la línea NDAC a nivel bajo (instante 8). El transmisor pondrá otros datos nuevos en las líneas DIO y comienza otro nuevo ciclo.

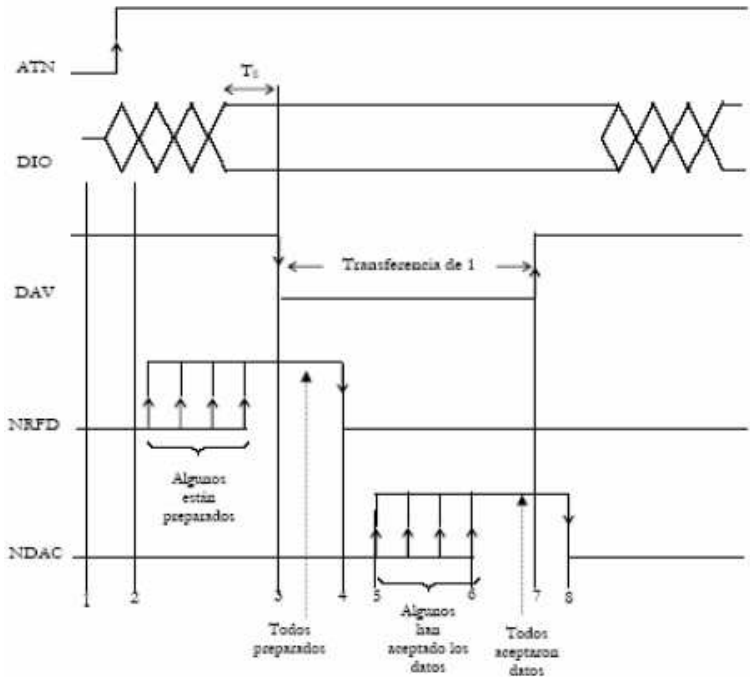


Figura I.11 - Diagrama de Tiempos de Operación de las líneas de control de la transferencia (Handshake) del bus GPIB

- **Líneas de Control General de la Interfaz**

Estas cinco líneas controlan el flujo de información a través de la interfaz:

- **IFC** (Interface Clear): El controlador activa esta línea para inicializar el bus interrumpiendo el proceso que se estaba realizando. Se deshabilita al transmisor y a los receptores activos, quedando todos inactivos. El controlador asume el mando del bus. Todos los dispositivos deben responder a esta línea en cualquier instante.

- **SRQ** (Service Request): cualquier dispositivo puede conducir una línea SRQ asíncronamente para solicitar un servicio del controlador.

- **ATN** (Attention): el controlador envía una línea ATN en alto cuando usa las líneas de datos para enviar comandos, y envía una línea ATN en bajo para que un transmisor pueda enviar mensajes de datos.

- **REN** (Remote Enable): el controlador envía la línea REN, para colocar los dispositivos en modo de programación local o remota.

- **EOI** (End or Identify): la línea EOI tiene dos propósitos. El transmisor usa la línea EOI para marcar el final del mensaje y con ATN en alto, identifica las respuestas de los dispositivos en un testeo paralelo.

– **Hardware Disponible para Sistemas GPIB**

Un sistema típico constará de un ordenador con una tarjeta controladora GPIB, más los instrumentos (compatibles con IEEE 488, obviamente). Existen tarjetas GPIB para prácticamente todos los ordenadores presentes en el mercado (PC, Macintosh,estaciones Sun, Silicon Graphics, DEC Alpha, HP RS/6000, etc). En el caso concreto del PC, las controladoras GPIB pueden conectarse al bus ISA, PCI, PCMCIA (portátiles), USB, Ethernet, Firewire, y los puertos serie y paralelo. Existen asimismo adaptadores para los estándares de comunicación RS-232 y RS-485. La figura I.12 muestra una tarjeta GPIB PCII/IIA de tecnología ISA de National Instruments empleada para la realización de este proyecto.

Figura I.12 - Tarjeta GPIB-PCI de National Instruments para ordenadores PC

3.2.8.4 Programación del bus GPIB

La estructura genérica de programación del bus GPIB para el control de instrumentos se divide jerárquicamente en los niveles que muestra la figura I.13. Dicho control se puede

llevar a cabo a través de un acceso directo al bus (comandos SCPI) o a través del driver del propio instrumento (IVI). Ambos métodos se explicarán a continuación. Para nuestro proyecto, como ya hemos indicado anteriormente, el entorno de programación elegido es el software Matlab. Accederemos al bus GPIB de manera directa a través de los comandos SCPI para poder llevar a cabo el control de instrumentos, tanto del laboratorio de docencia como del de investigación.

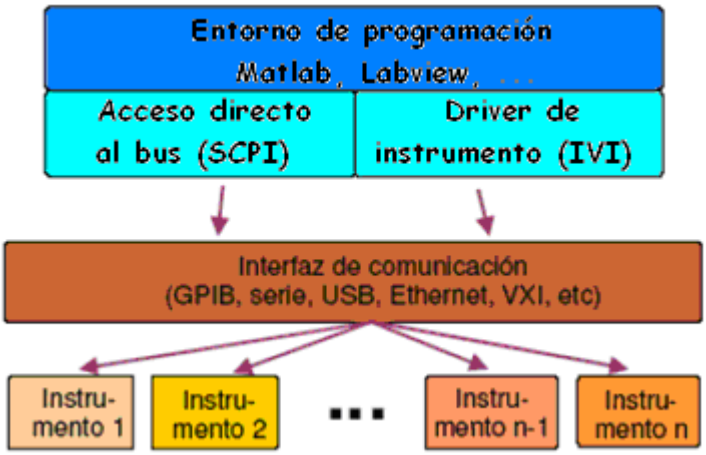


Figura I.13 - Estructura de Programación de un sistema GPIB

Comandos SCPI

La norma ANSI/IEEE 488.2 de 1987 (revisada en 1992) introdujo una estandarización de muchos aspectos que no estaban incluidos en la norma previa de 1975, como son: el formato de datos, el informe de estados, el tratamiento de errores, la funcionalidad del controlador y algunos comandos básicos al que todos los instrumentos deben responder. Así, esta norma establece algunas especificaciones del *software* que no estaban incluidas en la norma original. Sin embargo, deja abierto el formato y tipo de comandos que hay que enviar a los instrumentos. La norma SCPI (Standard Commands for Programmable Instruments) aparece en 1991 para conseguir una estandarización de los comandos de control y el formato de los datos de los instrumentos. El objetivo es que, independientemente del fabricante, equipos que tienen la misma funcionalidad respondan de igual forma a un conjunto estándar de comandos.

La norma SCPI es el escalón más alto dentro de la jerarquía normativa para el control de sistemas de instrumentación. Tal como se puede ver en la figura I.14, la norma SCPI se asienta sobre la IEEE-488.2 y esta, a su vez, se basa en la IEEE-488.1.

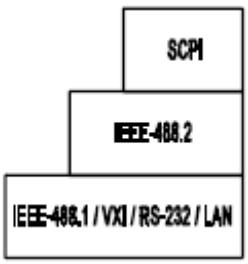


Figura I.14 - Estructura por niveles de los comandos SCPI

A pesar de esta jerarquía los comandos y la estructura de datos basados en la norma SCPI pueden usarse, y se usan, en sistemas de instrumentación que no estén basados en IEEE-488, por ejemplo en sistemas basados en VXI, RS-232 o LAN.

La norma SCPI reduce los costes de desarrollo y mantenimiento de programas de control de sistemas de instrumentación para pruebas automáticas. Esto se consigue ya que:

- facilita el aprendizaje y uso de los comandos y los datos
- facilita el desarrollo y mantenimiento de los programas
- posibilita la sustitución de equipos con los mínimos cambios de software. Para conseguir estos objetivos la norma establece la sintaxis y los formatos de los mensajes para que instrumentos con la misma funcionalidad o instrumentos del mismo tipo utilicen los mismos comandos. Por ejemplo, los comandos para medir una frecuencia utilizando frecuencímetros de distintos fabricantes serán los mismos. Además, la medida de la frecuencia con otro instrumento que lo permita, por ejemplo un osciloscopio digital o un multímetro, también utilizarán los mismos comandos

Los comandos SCPI se escriben como texto ASCII, y tienen una estructura jerárquica por niveles, separados por dos puntos, como se aprecia en la Figura I.15:

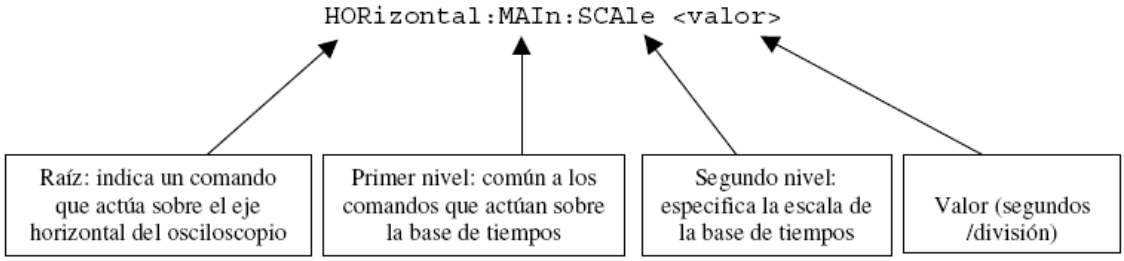


Figura I.15 – Estructura jerárquica de la norma SCPI

La norma establece tres tipos de compatibilidad entre instrumentos:

1. **Vertical:** equipos de un mismo tipo tienen los mismos controles. Ejemplo: en dos multímetros distintos la selección de escala se realizará de la misma forma.

2. **Horizontal**: equipos que realizan la misma medida, aunque sea de formas distintas, utilizarán los mismos comandos. Ejemplo: un osciloscopio que pueda medir períodos y un frecuencímetro-contador utilizarán los mismos comandos para medir el período de una señal.

3. **Funcional**: dos equipos distintos que puedan realizar la misma función lo harán con los mismos comandos. Ejemplo: un analizador de espectros que pueda realizar barridos de frecuencia y un generador de señal serán funcionalmente compatibles si el barrido de frecuencia se programa con los mismos comandos.

Para conseguir un conjunto de comandos que puedan ser utilizados en cualquier instrumento, optimizando la compatibilidad, la norma define un modelo de instrumento universal. Este modelo es el que pasamos a ver de forma más detallada a continuación. Para facilitar el aprendizaje de los nombres de los comandos, que provienen de

abreviaciones de palabras inglesas, usaremos las denominaciones inglesas para las distintas partes de los instrumentos.

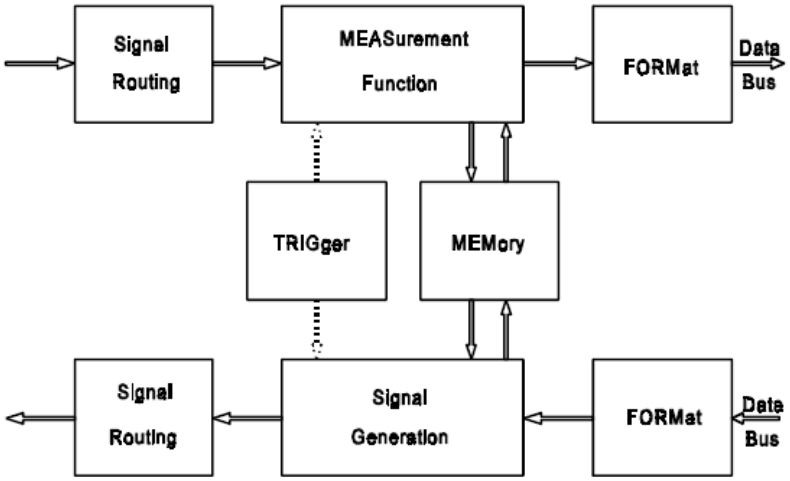


Figura I.16 - Modelo de instrumento según la norma SCPI

El objetivo de establecer un modelo general de instrumento es estandarizar un conjunto de bloques funcionales a los que se les asignará un conjunto de comandos para su programación. De esta forma cada instrumento concreto podrá ser descrito mediante un subconjunto de estos bloques funcionales y su programación se realizará utilizando los comandos correspondientes a estos bloques. El modelo de instrumento es el representado en la figura I.16. Este modelo describe el flujo de datos en un instrumento genérico; las líneas de trazo continuo representan flujo de datos y las líneas a trazos representan controles.

Cada instrumento específico se representa sólo por los bloques que lo constituyen. Por ejemplo, un multímetro digital puede tener sólo los bloques de función de medida, TRIGger y FORMat (Figura I.17). En cambio, para un generador de funciones sólo tendríamos: FORMat y Signal generation (Figura I.17).

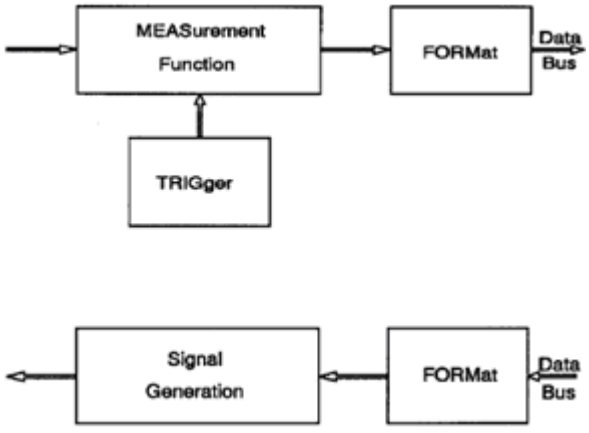


Figura I.17 – Ejemplos de bloques multímetro y generador de funciones

A continuación se describe cada uno de los bloques del modelo de instrumento definido:

- **Signal ROUTing:** Representa la posibilidad que tienen ciertos instrumentos para direccionar las señales de sus conectores de entrada a distintos circuitos

internos. Está representado por el bloque de direccionamiento de señal visto en la figura I.16. Los comandos que controlan esta sección se denominan ROUTe.

- **MEASurement Function:** Este bloque es el que realiza la medida, entendida en su sentido más amplio como una transformación de una señal en un código que luego podrá ser procesado. Este bloque se subdivide a su vez en los siguientes: (INPut, SENSe y CALCulate. La subdivisión del bloque de medida en estas tres partes no se ha incluido en el modelo de instrumento porque hay instrumentos que no serían 'horizontalmente compatibles a este nivel pero sí lo son a nivel del bloque funcional de medida. Por ejemplo, la medida del valor eficaz de una senoide con un voltímetro y con un osciloscopio digital podrían ser horizontalmente compatibles utilizando instrucciones al nivel del bloque MEASurement (utilizarían los mismos comandos), pero los comandos para programar la medida a un nivel más bajo serían distintos. En el voltímetro la medida se realiza con un comando del bloque SENSe, mientras que en el osciloscopio habría que calcular (utilizando comandos del bloque CALCulate) el valor eficaz a partir de las muestras adquiridas.

- **INPut:** Representa la etapa de adaptación de la señal de entrada, por ejemplo el acoplo AC, DC, GND de osciloscopios.

- **SENSe:** Es propiamente el bloque de medida, que pasa de una señal eléctrica a un código que luego puede ser procesado digitalmente.

- **CALCulate:** Su función es pasar los datos adquiridos por el bloque de medida a valores que sean más apropiados para una aplicación concreta. Por ejemplo, calcular períodos, frecuencias, tiempos de subida, etc. en un osciloscopio digital.

- **Signal Generation:** Este bloque también esta subdividido al igual que el de medida. Su función es la de pasar datos a una señal eléctrica. Para ello se pueden distinguir los siguientes sub-bloques.

- **SOURce:** La función de este bloque es determinar las características de la señal generada a partir del flujo de datos suministrado.

- **Output:** Es la parte que determina cómo se aplica la señal generada. Incluye las funciones de atenuación, filtrado, suma de tensiones continuas, etc.

- **TRIGger:** Su función es proveer al instrumento de medios para sincronizarse con eventos de las señales, tanto internas como externas.

- **MEMory:** Es el almacén de datos interno al instrumento. Según sea el caso pueden realizarse lecturas de datos, escrituras o guardarse parámetros de calibración internos.

- **FORMat:** Es el encargado de transformar el formato de los datos digitales internos al instrumento a otros que sean transferibles a través del bus de control.

Sintaxis y estilo

Los comandos en la norma SCPI se agrupan jerárquicamente en forma de árbol. En la raíz del árbol se encuentran los comandos que hacen referencia a los bloques que aparecen en el modelo de instrumento visto en la sección anterior. Cada uno de estos comandos se divide en un conjunto de ramas identificadas por palabras clave que a su vez identifican a funciones subordinadas al bloque raíz y así sucesivamente.
La notación utilizada es la siguiente:

- **:** indican el paso a un nivel jerárquico inferior. Para clarificar la notación también se ha utilizado identificación de los niveles inferiores

- **[ ]** Palabras clave opcionales

- < > Encierran el tipo de parámetro

- | separa parámetros opcionales (solo se puede poner uno de ellos)

- **;** separa comandos que están en la misma línea (no cambia el nivel del último comando)

- **,** se usa para separar distintos parámetros dentro de un mismo nivel

- **?** Indica que es un comando de consulta y que se espera una respuesta del equipo al que se envía. Es muy importante leer el dato solicitado; de no ser así al enviar otro comando se crea una situación de error en el instrumento

Existen dos grupos de comandos definidos en la norma: **comunes y específicos de instrumento.** Los comandos comunes son los mismos comandos que se definen como obligatorios en la norma IEEE 488.2. Éstos sirven para funciones tales como: reinicialización, autocomprobación y operaciones de estado. Los comandos específicos de instrumento son los propios que define la norma SCPI. A continuación, se muestra la lista de comandos comunes:

- *CLS Clear Status Command
- *ESE Standard Event Status Enable Command
- *ESE? Standard Event Status Enable Query
- *ESR? Standard Event Status Register Query
- *IDN? Identification Query
- *OPC Operation Complete Command
- *OPC? Operation Complete Query
- *RST Rest Command
- *SRE Service Request Enable Command
- *SRE? Service Request Enable Query
- *STB? Read Status Byte Query
- *TST? Self Test Query
- *WAI Wait-to-Continue Command

Los comandos específicos definidos en la norma SCPI se dividen a su vez en dos grupos, los obligatorios y los opcionales. Los únicos bloques que son obligatorios son el de SYSTem y el de STATus.

Los comandos específicos definidos en la norma SCPI son los siguientes:

- MEASure	- INPut	- ROUTe	- TRACe|DATA
- CALCulate	- INSTrument	- SENSe	- TRIGger
- CALibration	- MEMory	- SOURce	- UNIT
- DIAGnostic	- MMEMory	- STATus	- VXI
- DISPlay	- OUPut	- SYSTem	
- FORMat	- PROGram	– TEST	

Una ventaja del estándar SCPI es la definición homogénea de dichos comandos específicos para todos los aparatos de una misma clase; por ejemplo, para un osciloscopio, las principales categorías de raíz se muestran en la tabla I.7:

ACQuire	Adquisición de señales (muestra, promedio, etc)
CALibrate	Calibración y diagnóstico del aparato
CURSor	Control del cursor
DISplay	Control de la pantalla (formato YT o XT, contraste, etc)
HORizontal	Control de la base de tiempos (posición, escala, etc)
MATH	Operaciones matemáticas sobre las ondas (suma, resta, FFT)
MEASUrement	Medidas sobre las formas de onda (amplitud, frecuencia, etc)
TRIGger	Control del disparo (fuente, nivel, pendiente, etc)
CH	Control vertical de las señales (posición, amplitud, etc)
CURV, DAT y WFMP	Captura de ondas del osciloscopio

Tabla I.7 - Principales apartados de la jerarquía de comandos de un osciloscopio en SCPI

VISA e IVI

Para realizar una comunicación con un instrumento podemos utilizar diferentes buses de comunicación siempre y cuando el instrumento disponga de ellos. Para acceder al bus desde un PC, es necesaria una tarjeta controlador del bus, ya sea GPIB, VXI o cualquiera de ellos. Para acceder a cada una de estas tarjetas podemos utilizar las funciones propias del bus, como por ejemplo los comandos SCPI para el bus GPIB. Pero la utilización de éstas fuerzan que las aplicaciones que desarrollamos para un determinado instrumento sirven únicamente para ese bus y ese instrumento. Por ejemplo, si utilizamos comandos SCPI para controlar un multímetro vía bus GPIB, no podremos utilizar dicho programa para controlar el mismo instrumento utilizando un bus diferente. Para solucionar este y otros problemas semejantes, en 1993 National Instruments junto con GenRad, Racal Instruments, Tektronix y Wavetek formaron un consorcio llamado VXI plug&play Systems Aliance. Uno de los estándares más desarrollados por este grupo fue VISA (Virtual Instrument Software Architecture), que es un conjunto de funciones de alto nivel que se encarga de hacer transparente los recursos software que estemos utilizando.

Utilizando VISA podemos controlar buses tales como GPIB, VXI, PXI, serie y otros buses basados en computador (Figura I.18). VISA se encargará de utilizar las funciones de bajo nivel para cada uno de los buses de manera transparente para el programador.

De esta manera, el mismo código de programa para el control de cualquier instrumento utilizando VISA podrá ser usado vía GPIB, serie, etc., como muestra la figura siguiente:

Figura I.18 – Arquitectura de VISA

Las características principales de una comunicación que utiliza VISA son:

- Independencia de la plataforma utilizada
- Independencia de la interface utilizada

Posteriormente, en 1998, surgió el consorcio **IVI** (Interchangeable Virtual Instruments) entre una treintena de compañías, incluyendo las comentadas anteriormente, con el objetivo de alcanzar una estandarización de los drivers de los instrumentos. En concreto, IVI aportó las siguientes novedades:

- Adopción del conjunto de funciones VISA
- Posibilidad de intercambio de instrumentos, incluso de distintos fabricantes
- Posibilidad de trabajar con instrumentos simulados durante el desarrollo de aplicaciones, cuando la disponibilidad de los equipos está restringida

Los drivers IVI están clasificados en ocho clases, correspondientes a los siguientes instrumentos de medida:

- Fuentes DC
- Multímetros digitales
- Generadores de funciones
- Osciloscopios
- Medidores de potencia
- Generadores de RF
- Analizadores de espectros
- Conmutadores de señales (switches)

Por ejemplo, para el bus GPIB controlado a través de funciones de librerías IVI, el programador puede emplear rutinas de alto nivel sin necesidad de conocer el conjunto de comandos SCPI que el instrumento entiende. Como consecuencia, el desarrollo de aplicaciones de esta forma se ha agilizado considerablemente respecto al uso de comandos SCPI.

– **Software existente para GPIB**

El éxito de la "instrumentación virtual" reside en la existencia de potentes herramientas que permitan un diseño sencillo y eficiente del software. A continuación, se describen los principales programas usados para el control de los buses de instrumentación.

La primera alternativa (y, en ocasiones, la más práctica), es el uso de software propietario desarrollado por los mismos fabricantes del instrumento. Por ejemplo, el programa gratuito Intuilink de Agilent Technologies, o el programa de pago Wavestar para los osciloscopios de Tektronix Inc. La ventaja evidente de estos programas es que pueden ser empleados nada más conectar los instrumentos, y proporcionan ya hechas las funciones más comunes que uno puede desear realizar, sin necesidad de programar. Las desventajas son también claras: por tratarse de software cerrado, sólo puede ser usado para la tarea para la que fue diseñado, y además son imposibles de integrar con otros programas.

Otras alternativas (más configurables y adaptables a cualquier instrumento) son la utilización de lenguajes de programación más extendidos. En el siguiente apartado se describirá brevemente el software Labview por ser ampliamente utilizado para acabar describiendo el software Matlab, que es el utilizado en este proyecto.

- **LABVIEW:**

Labview es un software especialmente concebido para la adquisición, el análisis y la representación de datos y está basado en un lenguaje de programación gráfico muy intuitivo [5]. Los programas se componen de objetos conectados entre sí y organizados de forma jerárquica. Básicamente, el programador comienza diseñando la interfaz de usuario o apariencia del instrumento/programa que pretende realizar: botones, interruptores, indicadores numéricos, gráficos, etc. Para ello, utiliza una ventana denominada panel frontal (Figura I.19), donde sitúa los objetos con los que interactuará el usuario. El programa en sí se confecciona en otra ventana panel de control (Figura I.20), donde enlaza y opera los objetos que situó en el panel frontal.

Los objetos básicos que se utilizan en el panel de control de un programa en Labview son los iconos (que representan subprogramas), nodos (que realizan funciones e implementan estructuras de control) y terminales (que representan variables). Todos estos elementos se enlazan por medio de cables, que se asimilan a tuberías por donde circulan los datos. La programación se simplifica mucho gracias a que se dispone de una extensa librería de funciones para controlar de forma sencilla tarjetas de adquisición de datos y también multitud de instrumentos de test y medida. Es la herramienta perfecta para personas que no tengan muchas nociones de programación, puesto que ésta se realiza mediante iconos gráficos. El gran inconveniente es que si el programa es complejo, el esquema de iconos y líneas resultante también puede llegar a serlo hasta el punto de hacerse ininteligible.

Las Figuras I.19 y I.20 representan un osciloscopio concebido como un instrumento virtual. El sistema está formado por una tarjeta de adquisición, que reside en el interior del PC y el programa Labview de control adecuado. El instrumento funciona como si fuese un osciloscopio real pero su apariencia la especifica el usuario mediante el

software Labview definiendo la interfaz más ajustada a sus necesidades y dotándole de la funcionalidad que precise.

En la figura I.21 se muestra la arquitectura de Labview, en la que se puede observar que uno de los buses soportados a través de los drivers correspondientes es el bus GPIB. La instalación de estos drivers nos permitirá utilizar la tarjeta controladora del bus GPIB. La mayoría de tarjetas GPIB son actualmente plug&play, lo que significa que después de instalar los drivers, el sistema operativo reconoce automáticamente la tarjeta y configura los parámetros necesarios.

Para el control de instrumentos a través del bus GPIB, las operaciones básicas son las de iniciar el instrumento, leer y escribir datos desde o hacia él vía un PC. Dentro de la librería GPIB de Labview, existen iconos que se encargan de realizar estas y otras muchas funciones (Figura I.22):

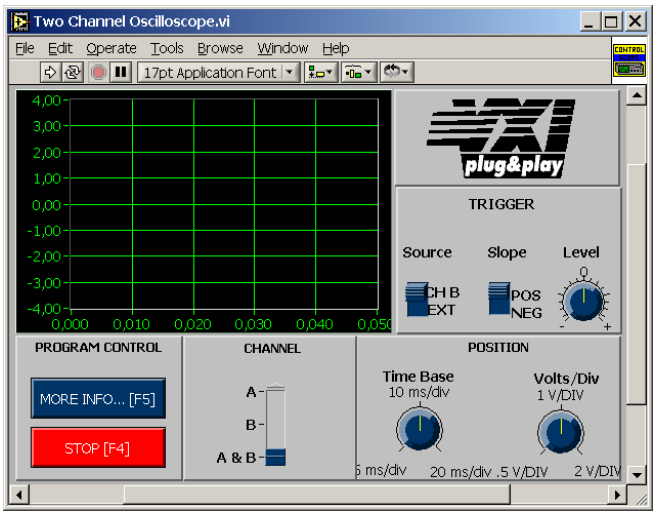


Figura I.19 - Panel frontal de un osciloscopio virtual programado mediante Labview

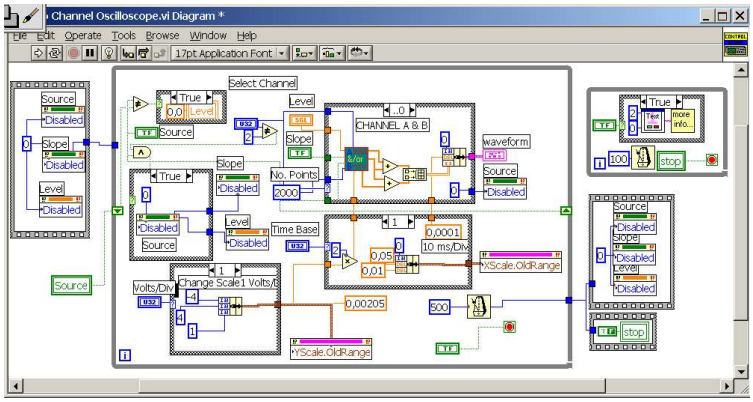


Figura I.20 - panel de control de un osciloscopiovirtual programado mediante Labview

Los aspectos más destacables del software Labview son:

- Posee completas librerías para la comunicación entre dispositivos (puerto serie, paralelo, GPIB, TCP/IP, etc.)

- Presenta especial facilidad para el desarrollo de interfaces gráficas adaptadas a los instrumentos de medida (dispone de elementos para mostrar formas de onda, conmutadores, potenciómetros, etc.)

- Soporte para los drivers IVI de numerosos instrumentos

- Tiene la posibilidad de ofrecer el programa final desarrollado mediante una aplicación de instalación. NI mantiene la compatibilidad con otros compiladores de propósito general, como las distintas suites de Microsoft Visual Studio (Visual Basic, Visual C++, .NET, etc)

- Extensa librería de funciones de procesado de señal y de toolbox (librerías que contienen programas especializados)

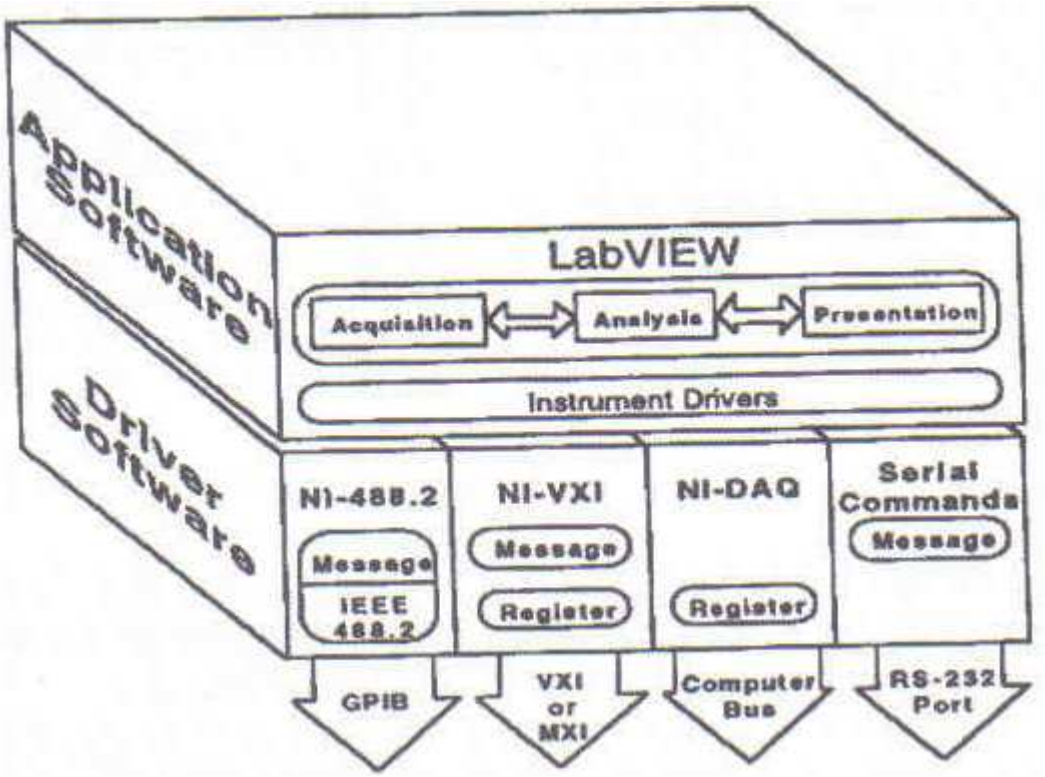
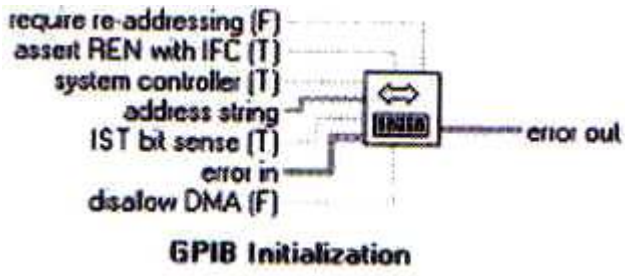
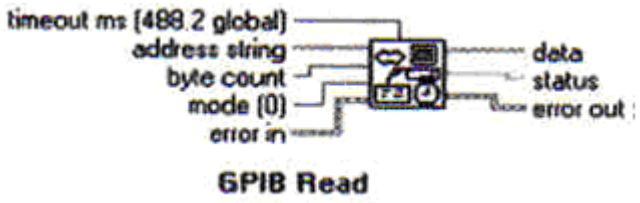


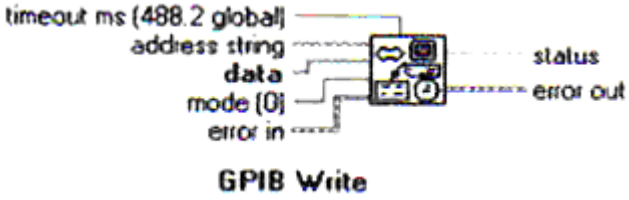
Figura I.21 – Arquitectura del software Labview



a)



b)



c)

Figura I.22 – Ejemplos de iconos gráficos Labview para la programación del bus GPIB: a) inicializar instrumento, b) leer del instrumento, c) escribir en el instrumento

- **MATLAB:**

Matlab, abreviatura de *MATrix LABoratory* (laboratorio de matrices), es un software matemático que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows y Apple Mac OS X. Es un software muy usado en universidades y centros de investigación y desarrollo.

Fue creado por *The MathWorks* en 1984, surgiendo la primera versión con la idea de emplear paquetes de subrutinas escritas en Fortran en los cursos de álgebra lineal y análisis numérico, sin necesidad de escribir programas en dicho lenguaje. El lenguaje de programación M fue creado en 1970 para proporcionar un sencillo acceso al software de matrices sin tener que usar Fortran. Es un software de cómputo, de alta ejecución numérica y de visualización. Matlab integra el análisis numérico, cálculo de matrices, procesamiento de señales, y graficación, en un ambiente sencillo de utilizar.

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUIDE) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete Matlab dispone de dos herramientas adicionales que expanden sus prestaciones, Simulink (plataforma de simulación multidominio). Esta última la utilizaremos en nuestro proyecto para realizar las interfaces gráficas de ambas partes.

El elemento básico de dato es una matriz, la cual no requiere de dimensionamiento, lo que permite resolver problemas en una fracción de tiempo, del que nos tomaría al escribir un programa en cualquier lenguaje.

Actualmente, Matlab, también cuenta con varias familias de soluciones para aplicaciones específicas llamadas toolboxes, que son colecciones de funciones utilizadas para resolver alguna clase particular de problema. Las áreas en donde los toolboxes están disponibles incluyen la adquisición y análisis de datos, herramientas para medición y verificación, conexión con dispositivos externos, visualización de resultados, etc.

A continuación, describimos las herramientas adicionales de Matlab que nos han sido útiles en el desarrollo de nuestro proyecto:

Integración de adquisición y análisis

A menudo el proceso de test y medición no finaliza una vez recogidos los datos o realizado el test. Necesitamos transformar esos datos en resultados concretos mediante un cuidadoso análisis y obtención de un modelo. En el pasado, esto requería que el proceso de test y medición se realizara en múltiples entornos. Los datos recogidos se guardaban en un fichero y después se transferían manualmente a otro paquete de software para el análisis.

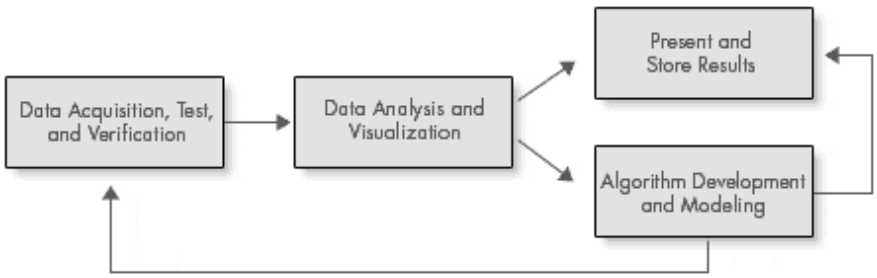


Figura I.23 – Proceso de obtención y análisis de datos

Actualmente, Matlab proporciona un medio sencillo de ir directamente desde la adquisición de datos a la obtención de resultados informativos y contiene herramientas para comunicar con el equipo de test y controlarlo (Figura I.23).

Test y medición

Mediante Matlab podemos realizar todo el proceso de adquisición y análisis de datos, incluida la conexión con los instrumentos y dispositivos de adquisición de datos, la visualización y la producción de salidas en calidad de presentación. Las principales ventajas que presenta son:

- Comunicación con el hardware de estándar industrial y posibilidad de almacenar de manera directa los datos medidos
- Lectura y escritura de/hacia los instrumentos con los comandos específicos.
- Posibilidad de realizar cálculos y análisis de datos en tiempo real
- Visualizar los datos durante la adquisición para la verificación de los mismos
- Posibilidad de crear gráficos e informes de sus resultados

Herramientas para test y medición

Existen dos métodos para la obtención de medidas con Matlab. El primero es mediante tarjetas de adquisición de datos (toolbox Data Acquisition); el segundo consiste en la obtención de dichas medidas a través del control de un bus de instrumentación que controle los equipos (toolbox Instrument Control).

Data Acquisition Toolbox

El Data Acquisition Toolbox nos permite llevar directamente a Matlab los datos medidos y recogidos a través de la tarjeta de adquisición de datos para su análisis y visualización. Proporciona una serie de herramientas para controlar y comunicar con el hardware (tarjetas) de adquisición de datos.

Instrument Control Toolbox

El Instrument Control Toolbox nos permite controlar y comunicarnos con dispositivos externos a través de los protocolos de comunicación GPIB y VXI directamente desde Matlab. Proporciona también interfaces intuitivas para conectar y comunicar con su instrumento. También podemos usar estas interfaces con dispositivos de puerto en serie.

En nuestro caso, hemos utilizado este toolbox para poder llevar a cabo el control de instrumentos a través de la programación directa del bus GPIB.

Interfaz con dispositivos externos

Las herramientas de verificación y medición Matlab combinan adquisición de datos y capacidad de conexión de los instrumentos con las potentes funciones de análisis y visualización de Matlab, todo en un único entorno.

Con Matlab y los Data Acquisition e Instrument Control Toolboxes, podemos:

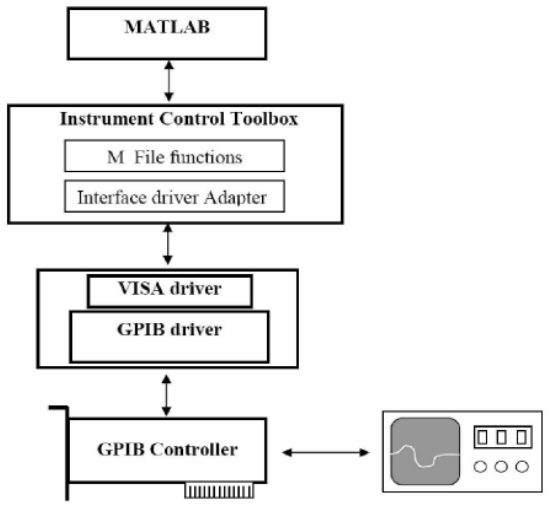
- Crear una interfaz con tarjetas de adquisición de datos para conducir directamente a Matlab datos medidos en directo
- Combinar análisis y recogida de datos sin transferir datos entre los entornos separados de análisis y de adquisición de datos
- Usar una única interfaz consistente hacia diversos dispositivos, independientemente del fabricante del hardware
- Crear su propia interfaz gráfica personalizada usando la aplicación GUIDE

Estas herramientas permiten moverse de manera continua entre la recogida de datos y el análisis, con lo que se puede generar resultados reales con mayor velocidad. Son ideales para una amplia variedad de aplicaciones que impliquen test de componentes a nivel de sistema.

Análisis y visualización de datos

Para nuestro proyecto, controlaremos los instrumentos a través de un bus GPIB. Éste será controlado a su vez por la tarjeta controladora GPIB instalada en el PC.

Accederemos al bus de manera directa programándolo con Matlab (Figura I.24). Matlab soporta todo el proceso de exploración y análisis de datos, permitiendo:



- Acceder a características matemáticas y de visualización avanzadas
- Hacer actualizaciones iterativas del test basándonos en los resultados del análisis
- Diseñar front-ends de GUI personalizados para automatizar el proceso de recogida y análisis de datos
- Usar algoritmos punteros verificados en otros toolboxes de MATLAB

Figura I.24 – Esquema de control de instrumentos mediante el software Matlab

Presentación e informe de resultados

Una vez realizado el test y finalizados los resultados, mediante Matlab, podemos compartir y visualizar los resultados de manera instantánea. Con Matlab y los toolboxes complementarios podemos:

- Intercambiar datos con bases de datos
- Generar informes personalizados en múltiples formatos de salida
- Exportar sus datos a hojas de cálculo de Microsoft Excel y documentos de Word

4. CONCLUSIÓN

En este capítulo se han definido los objetivos del proyecto y se han descrito las opciones más relevantes sobre los buses de instrumentación existentes. En particular se ha descrito con más nivel de detalle el bus GPIB así como el software Matlab para su control por ser los utilizados en este proyecto.

El bus GPIB sigue siendo uno de los buses de instrumentación más utilizados y que sigue manteniéndose en instrumentos de medida de gama alta. En cuanto al software de programación, no existe actualmente en el mercado uno que destaque de manera clara como pasa con los buses en lo que a la programación del control de instrumentos se refiere puesto que depende de las características de la aplicación a desarrollar. En el siguiente capítulo se explicará cómo se ha realizado el control de instrumentos del laboratorio docente.

CAPÍTULO II

INTRODUCCIÓN

En este capítulo se explicará la parte del proyecto llevada a cabo en el laboratorio docente. Se fijarán los objetivos a alcanzar y se dará una explicación completa de los equipos utilizados y el funcionamiento de la aplicación; analizando el procedimiento de control, el procesado de la información, la estructura del programa y la interfaz gráfica de usuario. Por último, se mostrará un ejemplo completo de la aplicación.

1. OBJETIVOS

En esta primera parte del proyecto, el objetivo final es desarrollar una aplicación que permita controlar los instrumentos básicos del laboratorio de docencia (osciloscopio y generador de funciones). Dicha aplicación debe de ser capaz de recoger los datos de las mediciones con estos instrumentos y procesarlos adecuadamente para dar información útil sobre las características de un determinado circuito eléctrico. Todo ello será realizado desde una interfaz gráfica de usuario diseñada con el aplicativo GUIDE de Matlab [7].

Como ejemplo, la aplicación se utilizará para determinar y representar el diagrama de Bode en magnitud de un circuito RC (filtro paso bajo). El Diagrama de Bode es una representación gráfica que sirve para caracterizar la respuesta en frecuencia de un sistema. Consta de dos gráficas separadas, una que corresponde representación del módulo y otra que corresponde con la fase de la función de transferencia de un sistema.

El diagrama de Bode de módulo es una herramienta muy utilizada en el análisis de circuitos en electrónica, siendo fundamental para el diseño y análisis de filtros y amplificadores. Consiste en dibujar el módulo de la función de transferencia (ganancia) en decibelios en función de la frecuencia (o la frecuencia angular) en escala logarítmica (Figura II.1a). Se suele emplear en procesado de señal para mostrar la respuesta en frecuencia de un sistema lineal e invariante en el tiempo. Es de amplia aplicación en ingeniería de control, pues permiten representar la magnitud de la función de transferencia de un sistema, sea éste eléctrico, mecánico,... Su uso se justifica en la simplicidad con que permiten, atendiendo a la forma del diagrama, sintonizar diferentes controladores, y porque se puede, en un reducido espacio, representar un amplio espectro de frecuencias.

El diagrama de Bode de fase (Figura II.1b) permite evaluar el desplazamiento en fase de una señal a la salida del sistema respecto a la entrada para una frecuencia determinada. Por ejemplo, tenemos una señal $A\sin(\omega t)$ a la entrada del sistema y asumimos que el sistema atenúa por un factor x y desplaza en fase $-\Phi$. En este caso, la salida del sistema será $(A/x)\sin(\omega t - \Phi)$. Generalmente, este desfase es función de la frecuencia ($\Phi = \Phi(f)$); esta dependencia es lo que nos muestra el Bode.

La respuesta en amplitud y en fase de los diagramas de Bode no pueden por lo general cambiarse de forma independiente: cambiar la ganancia implica cambiar también desfase y viceversa.

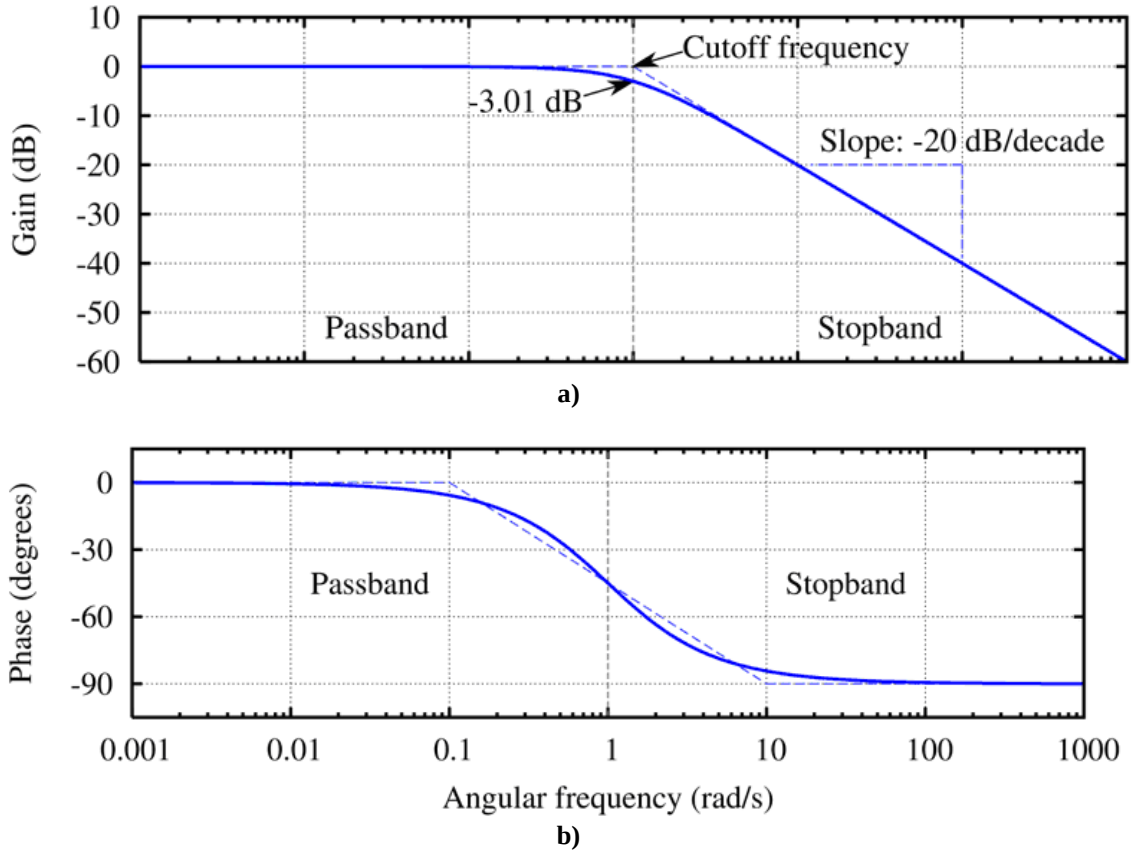


Figura II.1 – Representación del diagrama de Bode de módulo (a) y fase (b) de un filtro paso bajo

Por otro lado, se define la frecuencia de corte de un sistema como aquella frecuencia en al que el diagrama de bode de módulo del sistema cae 3dB respecto del máximo. Es un valor vital en muchos aspectos de la ingeniería y la electrónica, sobretodo en el diseño de filtros y amplificadores ya que por ejemplo en un filtro paso-bajo, se considera que en la práctica, a partir de la frecuencia de corte, el sistema rechazará las demás frecuencias superiores.

En nuestro proyecto, con los instrumentos que tenemos en el laboratorio de docencia somos incapaces de medir la fase del sistema a analizar, por lo que sólo representaremos [8] el diagrama de módulo. No obstante, podremos simular el desfase de las señales mediante la introducción de los parámetros iniciales de la onda de entrada, como explicaremos más adelante.

2. DESCRIPCIÓN DE EQUIPOS UTILIZADOS

A continuación, se describe el software y los equipos del laboratorio docente utilizados en esta primera parte del proyecto:

- **PC con el software Matlab y una tarjeta controladora GPIB instalada**:

Toda la aplicación está desarrollada en Matlab [6]. Para el control de la instrumentación se ha utilizado el paquete "Instrument Control toolbox" mientras que para el desarrollo de la interfaz gráfica de usuario hemos utilizado el "Open GUI Layout Editor" (GUIDE).

- **Osciloscopio Agilent TDS220:**

Los osciloscopios TDS220 (Figura II.2) constan de dos canales y están montados en un armazón de mesa ligero y tiene las características técnicas que se mencionan a continuación:

- Ancho de banda de 100MHz con un ancho de banda seleccionable de 20 MHz
- Velocidad de muestreo de 1 GS/s y longitud de registro de 2,500 puntos por canal
- Cursores con lecturas y cinco mediciones automáticas
- Pantalla LCD de alta resolución y alto contraste, con compensación de temperatura y luz de fondo reemplazable
- Almacenamiento de parámetros y formas de onda
- Ajuste automático para una configuración rápida
- Cálculo de promedios y detección de picos de la forma de onda
- Osciloscopio digital de tiempo real (sobremuestreo de al menos diez veces)
- Base de tiempo doble
- Capacidad de disparo de video
- Se puede agregar con facilidad puertos de comunicación RS-232, GPIB y centronix con módulos de extensión optativos
- Interfaz del usuario que puede seleccionarse en 10 idiomas diferentes

Figura II.2 - Osciloscopio Agilent TDS220

- **Generador de señales Agilent 33120A:**

El Agilent 33120 (Figura II.3) es un generador de señales sinusoidales que utiliza técnicas directas de síntesis digital para crear ondas estables. Precisa la señal de salida para ondas limpias. Proporciona formas de onda cuadradas y rampas lineales de hasta 100μHz. Las características técnicas que presenta son:

- 15MHz de ancho de banda
- Formas de onda disponible: sinusoidal, rampa, triángulo, ruido y DC

- 12 bits de resolución
- 40MSa/s
- 16000 puntos de formas de onda arbitraria
- Modulaciones AM, FM, FSK
- Barridos en frecuencia tanto lineales como logarítmicos
- Conexión bus GPIB y RS-232
- Compatibilidad de programación con los comandos SCPI

Figura II.3 - Generador de señales Agilent 33120A

- **Sonda pasiva de medición Agilent:**

Son sondas de uso general diseñadas para un funcionamiento óptimo con osciloscopios. Están equipadas con un cable duradero y un cuerpo de sonda sólido de acero inoxidable encapsulado con plástico duro y resistente a fracturas (Figura II.4).

También presenta una punta de gancho retráctil de uso general: se engancha en cables y puntos de prueba para utilizar las sondas sin manos Varilla de tierra: proporciona un cable de conexión a tierra flexible y corto para medidas de alta frecuencia. Cable de tierra con pinza de cocodrilo de uso general: permite una conexión a tierra versátil. Características técnicas:

- Ancho de banda: 20MHz
- Impedancia de entrada: 1Mohm
- Capacidad de entrada: 70pF
- Voltaje máximo de entrada: 400Vp
- Tiempo de subida: <17.5ns
- Retardo de propagación: 7ns (cable 1,5m)
- Rango de temperatura: -10-50º C

Figura II.4 - Sonda pasiva de medición Agilent

3. FUNCIONAMIENTO DE LA APLICACIÓN

3.1 COMUNICACIÓN CON LOS INSTRUMENTOS

La comunicación de los instrumentos sobre los cuales queremos realizar el control se realiza de manera directa mediante la programación del bus GPIB a través los comandos SCPI. A continuación, se muestran los comandos básicos que se han utilizado para la comunicación [9] [10]:

- **Inicio de los instrumentos:** para poder llevar a cabo el control de los instrumentos es necesario inicializarlos, es decir, asignar una dirección GPIB que identifique el instrumento dentro del bus. Ejemplo:

```
%con estas líneas abriríamos el canal GPIB del osciloscopio
osc=gpib('ni',0,8);
fopen(osc);
```

Mediante estas líneas, se identifica un instrumento de la marca National Instruments (ni) que se encuentra en la dirección GPIB número 8 y que estará controlado por la tarjeta controladora GPIB situada en el PC que tiene la dirección 0 (dirección por defecto). Todo ello quedará almacenado en la variable "osc". Mediante la instrucción "fopen", se iniciala comunicación con el instrumento que queda listo para ser controlado a través del bus.

```
%con estas líneas abriríamos el canal GPIB del generador de
funciones
gen=gpib('ni',0,4);
fopen(gen);
```

- **Enviar información:** para poder comunicarnos con el instrumento a controlar, en primer lugar, creamos una variable de tipo "string", en la que se almacena la orden a enviar al instrumento. Ejemplo:

```
%con estas líneas se envía al generador de funciones una onda
senoidal con los parámetros indicados:
c=sprintf('APPL:SIN %f HZ, %f VPP, %f V',freq,amp,offset);
fprintf(osc,c)
```

Almacenamos en la variable “c” la instrucción que ordena al generador de señales emitir una señal sinusoidal de frecuencia (freq), amplitud (amp) y una tensión de offset (offset); dónde “freq”, “amp” y “offset” son variables leídas a través de Matlab. Por último, la instrucción almacenada en “c” se envía al instrumento mediante la instrucción “fprintf”; donde “gen” es la variable que enlaza Matlab con el generador de señales.

- **Lectura de datos:** el comando utilizado para la lectura de información de los equipos es el “fscanf”. Primero, hay que enviarle una pregunta sobre qué variable queremos medir para posteriormente leerla y analizarla. Ejemplo:

```
%con estas líneas leemos el valor de la tensión que hay en el
canal “x” del osciloscopio y lo guardamos en la variable
“Vppchar”:
c=sprintf('measurement:meas%d:VALUE?',x);
fprintf(osc,c);
Vppchar=fscanf(osc,c);
```

- **Cerrar los instrumentos:** una vez realizadas las medidas y almacenadas los resultados de interés, es necesario cerrar los instrumentos utilizados, es decir, dejamos de controlarlos a través del bus. Ejemplo:

```
%con estas líneas cerramos el osciloscopio y el generador de
funciones:
fclose(osc);
fclose(gen);
```

3.2 PROCESADO DE DATOS

Una vez que hemos conseguido, a través del control del bus, realizar las medidas, hemos de procesar los datos de los instrumentos, pues éstos no se leen en un formato numérico convencional. Por tanto, es necesario un procesado previo para poder llevar a cabo el análisis matemático.

Las lecturas directas de los instrumentos son una cadena de caracteres en formato alfanumérico ASCII que contienen el valor medido que nos interesa. El procesado previo consiste en eliminar los caracteres tanto por delante como por detrás y quedarnos sólo con el valor numérico de la medida; teniendo en cuenta el signo para los números negativos. Ejemplo: a continuación mostramos un ejemplo de la lectura directa que se obtiene del osciloscopio:

XXXXXXXXXXXXXXXXX-5.0000e-007XXXXX; donde las “X” corresponden a los caracteres ASCII que deseamos eliminar pues no contienen información útil para nuestra aplicación. El procesado consistiría en obtener de toda la cadena resultante de la medida, el valor 0.0000005 en este caso.

3.3 ESTRUCTURA DEL PROGRAMA

La programación de la aplicación está realizada con el lenguaje propio de Matlab. El programa consta de cinco partes principales:

1. leer las variables introducidas por el usuario
2. iniciar el control de los instrumentos a través del bus GPIB
3. cerrar los instrumentos
4. procesado de datos
5. presentación de resultados

3.4 INTERFAZ GRÁFICA

Usando la aplicación GUIDE de Matlab, hemos realizado una interfaz gráfica que actúa como puente entre el usuario y el programa. Mediante esta pantalla, se configuran de manera sencilla los parámetros de entrada al circuito y se representan los resultados obtenidos del análisis del mismo (Figura II.5):

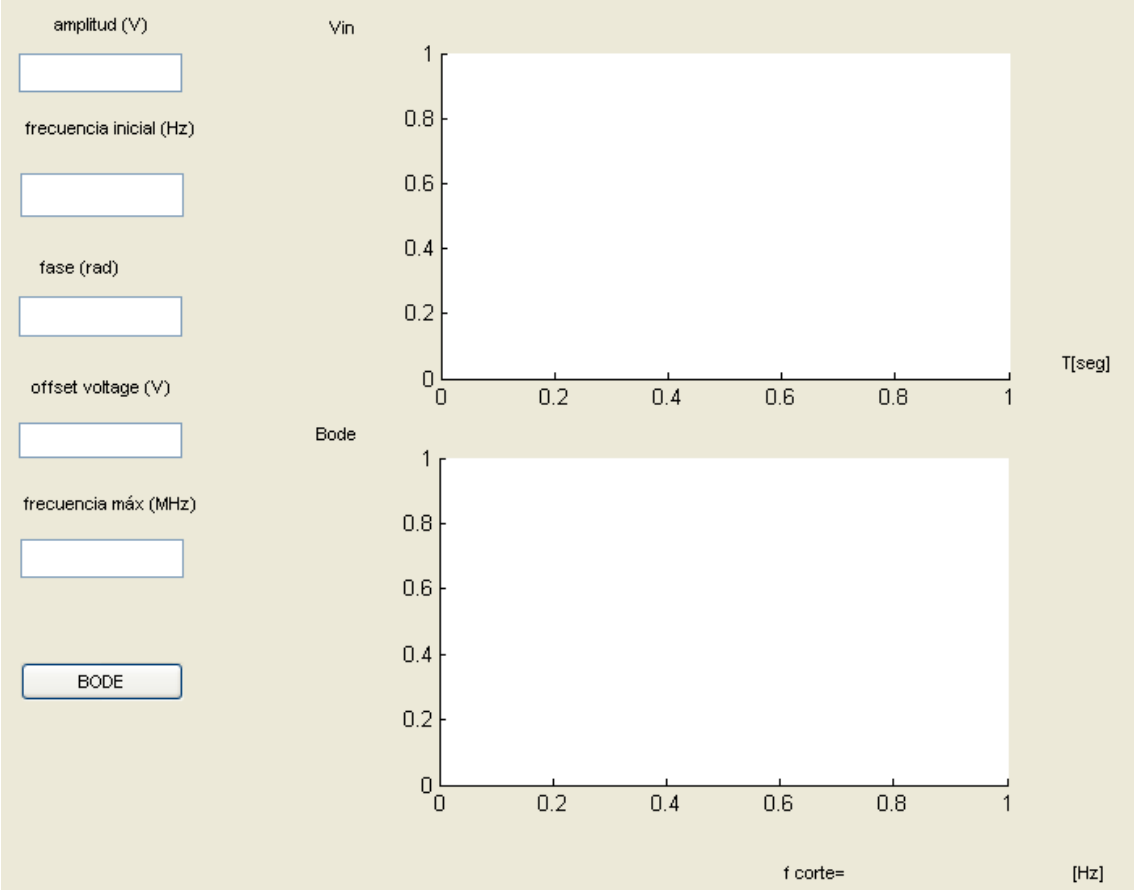


Figura II.5 – Interfaz gráfica de usuario

4. RESULTADOS

En este apartado mostraremos un ejemplo real de la aplicación. Obtendremos el diagrama de Bode de un sistema. En este caso, nuestro sistema será un circuito RC (filtro paso bajo), por lo que montaremos y conectaremos el circuito siguiente (Figura II.6):

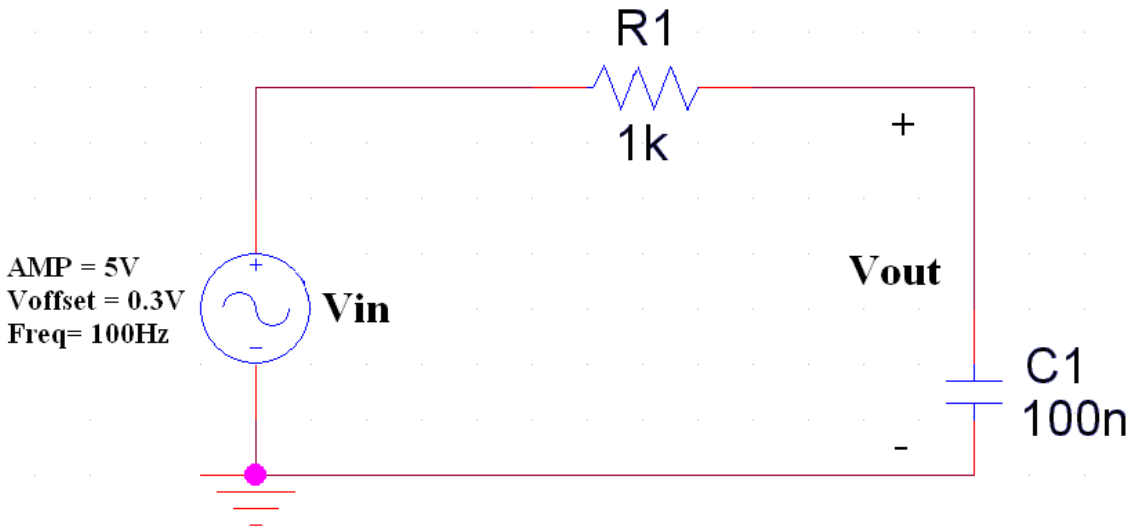


Figura II.6 – Esquema circuital del filtro paso-bajo del que se realiza el diagrama de Bode

El control de la aplicación por parte del usuario es realmente sencillo gracias a la interfaz gráfica de programa. Cualquiera, con conocimientos de Matlab o no, puede realizar el diagrama de bode de cualquier sistema. El procedimiento de control de la aplicación se muestra a continuación en un desglose de pasos:

1. Antes de empezar, el usuario debe conectar el generador de señales a la entrada del sistema. Además, para poder medir la tensión, conectará las sondas de osciloscopio a la entrada y la salida del sistema para leerlas por el canal 1 y 2.

2. Al abrir la interfaz gráfica, el usuario introduce los parámetros de la onda de entrada que quiere enviar al sistema. Estos parámetros iniciales son: amplitud, frecuencia inicial, fase, voltaje offset y frecuencia máxima.

3. Dichos valores son leídos con Matlab y enviados por el bus GPIB mediante los comandos SCPI al generador de señales, que realizará un barrido entre las frecuencias indicadas, midiendo las tensiones de entrada (Vin) y salida (Vout).

4. Los datos, una vez leídos y almacenados, se procesan para poder ser analizados matemáticamente. Posteriormente, se calculan las ganancias en decibelios y se calcula la frecuencia de corte del sistema.

5. Por último, cuando el análisis ha finalizado, se mostrará al usuario en la interfaz gráfica la onda inicial que se envía, el diagrama de bode del sistema y la frecuencia de corte del mismo.

A continuación, se presentan las funciones utilizadas. Las funciones son estructuras de programación que se utilizan para la simplificación y optimización del mismo. Necesitan unos argumentos de entrada que son los parámetros que servirán para hacer los cálculos de cada una de ellas. Hemos utilizado cuatro funciones:

1. función **inicia_inst:** esta función conecta el PC con el equipo deseado y establece la conexión a través del bus GPIB para el tráfico de información. Argumentos que hay que proporcionarle:

- la dirección GPIB del instrumento que queremos controlar.

2. función **genera_onda2:** envía las características de la onda que se quiere obtener del generador de funciones. Los argumentos que hay que proporcionar a la función son:

    - frecuencia de la onda
    - amplitud de la onda
    - voltaje Offset de la onda

3. función**VppCH:** obtiene el valor de la tensión pico a pico del canal del osciloscopio que le indiquemos. Los argumentos que le han de ser proporcionados son:

    - canal en el que queremos realizar la medida. En nuestro proyecto, hemos configurado previamente el osciloscopio de manera que la tensión de entrada está en el canal 3 y la de salida en el canal 4

4. función **fcorte**: calcula la frecuencia de corte del sistema. Los argumentos que se le deben proporcionar son:

    - el vector de las ganancias calculadas en dB
    - el vector de las frecuencias utilizadas en el barrido

A continuación, se muestra el esquema de bloques del funcionamiento de la aplicación (Figura II.7):

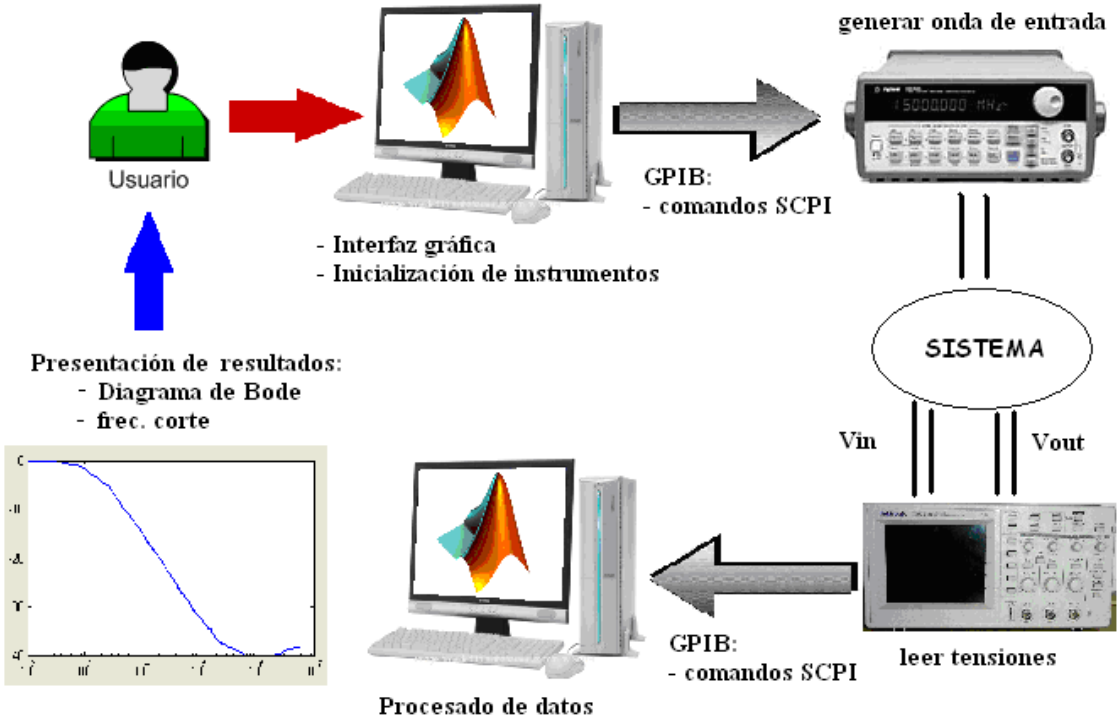


Figura II.7 - Esquema de bloques de funcionamiento

A continuación, se muestra el diagrama de flujo del programa (Figura II.8):

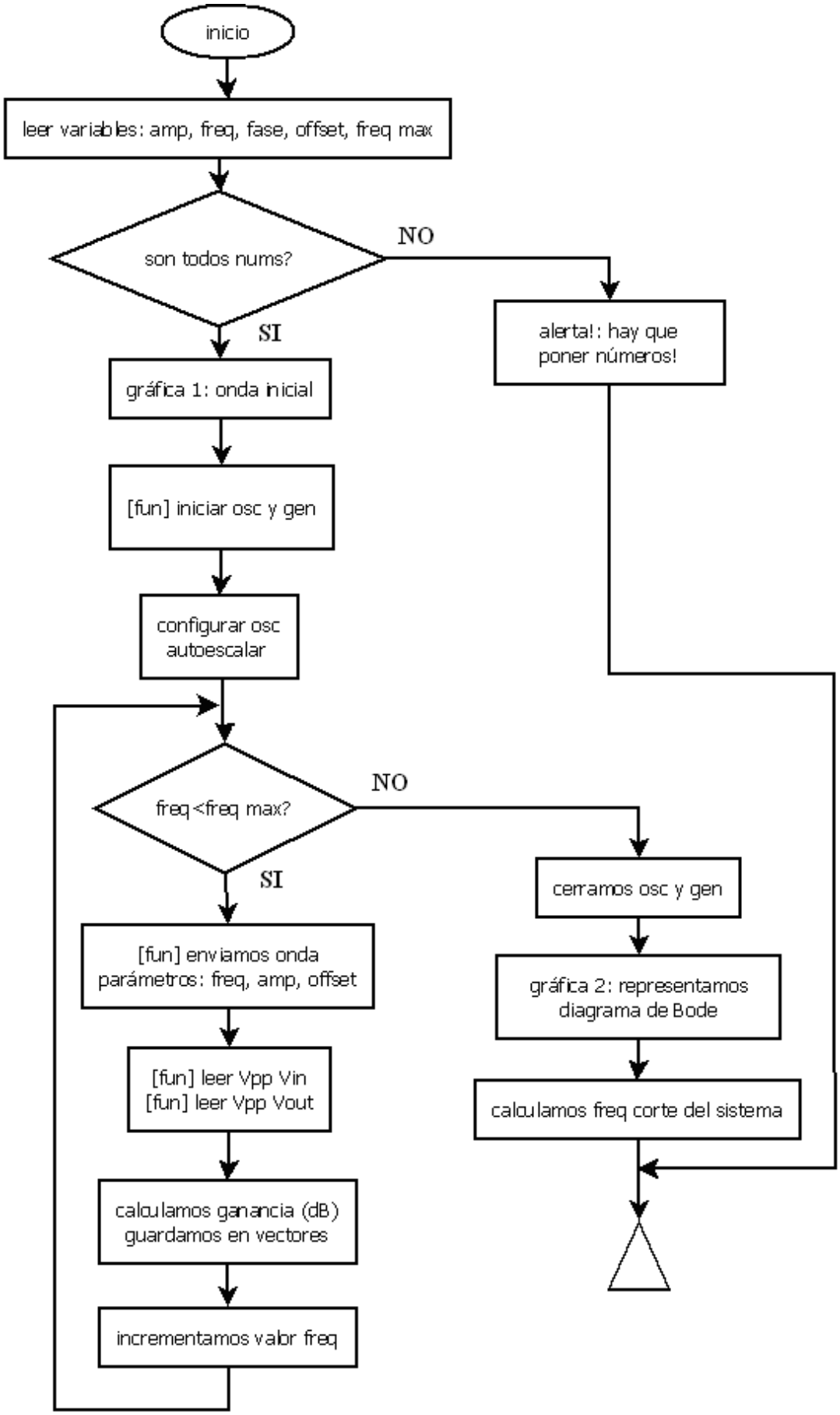


Figura II.8 – Diagrama de flujo del programa

A continuación, se muestra un ejemplo de la obtención del diagrama de Bode en módulo del circuito RC de la Figura II.6. Una vez hemos realizado las medidas y almacenado los valores en el formato adecuado, hacemos los cálculos necesarios para determinar el diagrama de Bode del circuito. Éstos se muestran a continuación en un desglose de pasos:

1. cálculo de la ganancia del sistema en lineal: consiste en dividir la tensión pico-pico de la salida del sistema entre la tensión pico-pico de la entrada
2. cálculo de la ganancia del sistema en lineal: pasamos el valor de la ganancia a dB
3. Almacenamiento de datos
4. cálculo de la frecuencia de corte del sistema: miramos a qué frecuencia corresponde el punto en el que el diagrama de Bode cae 3dB respecto del máximo
5. Representación gráfica: una vez se haya hecho el barrido entre las frecuencias indicadas, se representará el diagrama de Bode junto con la frecuencia de corte del sistema

Para nuestro ejemplo, introducimos en la interfaz gráfica de usuario los siguientes parámetros:

- Amplitud: 5 V
- Frecuencia inicial: 100 Hz
- Fase: 0.4 radianes.
- Voltaje Offset: 0.3 V
- Frecuencia máxima: 15 MHz.

La frecuencia máxima es el parámetro que utilizamos para acotar el barrido de frecuencias que queremos representar en el diagrama de Bode. Este valor también puede venir determinado por las características propias de los equipos que estemos utilizando.

Al pulsar en botón de “BODE”:

- Se representa en la gráfica superior la onda inicial enviada al sistema configurada con los parámetros introducidos.
- El programa realiza el barrido en frecuencias desde los 100Hz hasta los 15MHz y representa el diagrama de Bode de módulo en la gráfica inferior de la pantalla.
- Se calcula la frecuencia de corte del sistema y se muestra en la esquina inferior derecha.

A continuación, se muestran los resultados (Figura II.9):

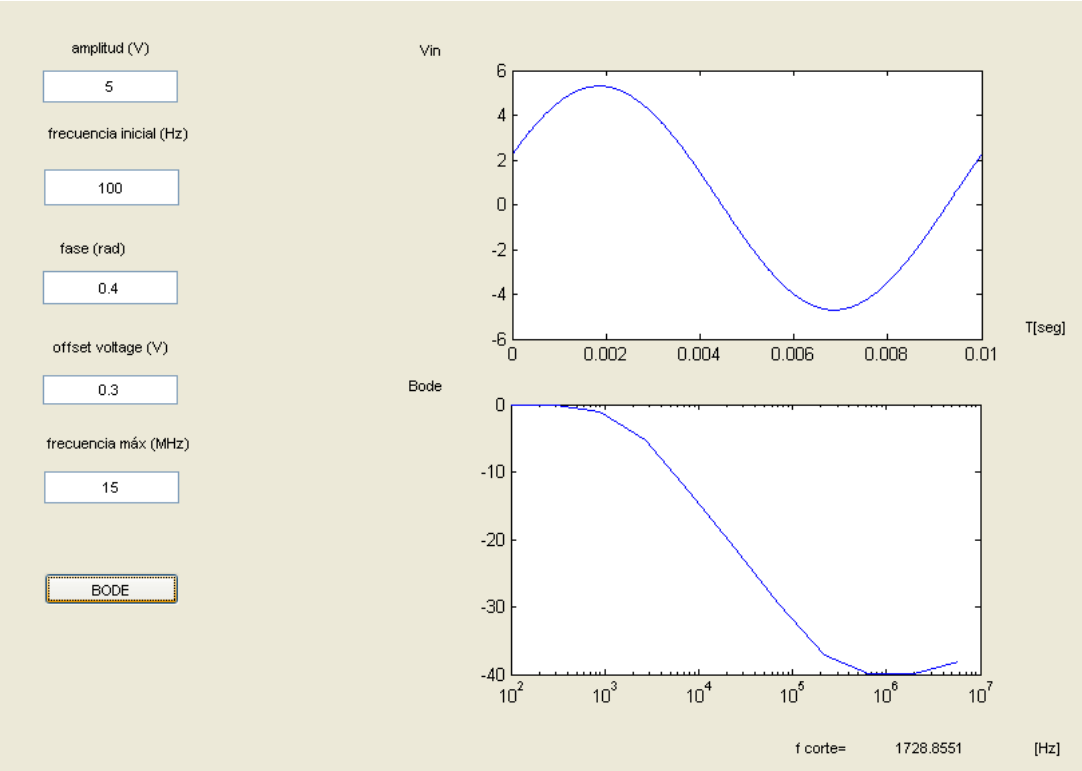


Figura II.9 – ejemplo de la aplicación

Como se puede observar, con las características introducidas por el usuario, la frecuencia de corte del sistema es de 1728,85 Hz, por lo que en la práctica, el sistema atenuará todas las frecuencias superiores a ésta.

5. CONCLUSIÓN

En este capítulo se ha descrito la primera parte del proyecto que ha consistido en el control de instrumentos del laboratorio docente (osciloscopio y generador de funciones). Como ejemplo de la aplicación se ha obtenido el diagrama de Bode de un filtro paso bajo controlando estos equipos a través del bus GPIB. La programación de dicho bus se ha hecho mediante los comandos SCPI. Se ha presentado la estructura del programa Matlab realizado, describiendo las funciones utilizadas en el procesado de datos. También se ha desarrollado una interfaz gráfica con la aplicación GUIDE de Matlab para facilitar el uso del programa. En este capítulo se muestran los pasos para realizar el ejemplo del cálculo del diagrama de Bode de un circuito a través de ella.

El osciloscopio y el generador de funciones son instrumentos de laboratorio relativamente sencillos por lo que el control de éstos me servirá como base para la segunda parte del proyecto, en la que llevaré a cabo el control sobre un analizador de semiconductores, un instrumento sin duda, muchísimo más complejo.

CAPÍTULO III

INTRODUCCIÓN

En este capítulo se explicará la parte del proyecto llevada a cabo en el laboratorio de investigación dedicado a la caracterización de dispositivos a nivel de oblea. Se fijarán los objetivos a alcanzar, se describirán los equipos utilizados y se dará una explicación completa del funcionamiento de la aplicación; analizando la comunicación con los instrumentos, la interfaz gráfica de usuario y la estructura de programa. Por último, se mostrará un ejemplo en el que la aplicación desarrollada se utilice para caracterizar transistores MOSFET.

1. OBJETIVOS

En esta segunda parte del proyecto, el objetivo final es desarrollar una aplicación que permita controlar el analizador de semiconductores del laboratorio de investigación. El analizador de semiconductores es un instrumento que sirve para caracterizar componentes o circuitos electrónicos. Permite:

- Aplicar tensión (en forma de rampas de tensión o tensiones constantes) y medir corrientes
- Aplicar corrientes (rampas o corrientes constantes) y medir tensiones

En nuestro caso, mediante la aplicación a realizar debemos de ser capaces de enviar a un circuito rampas de tensión o tensiones constantes configurables y medir las corrientes que se producen. De nuevo, y como en el capítulo 2, tendremos que realizar un procesado previo de los datos para poder analizarlos matemáticamente y obtener información útil del circuito o componente electrónico que queramos caracterizar. Todo ello será realizado desde una interfaz gráfica de usuario diseñada con el aplicativo GUIDE de Matlab.

Cabe decir que el analizador de semiconductores posee una pantalla propia mediante la cual se configuran los parámetros de las señales que queremos enviar al circuito. La finalidad de esta aplicación nace de la dificultad de configurar estos parámetros de esta manera, ya que es un proceso muy lento cuando se pretende realizar varias medidas sobre el mismo dispositivo. Mediante la interfaz, realizaremos esta tarea de una forma mucho más sencilla e intuitiva.

2. DESCRIPCIÓN DE EQUIPOS UTILIZADOS

A continuación, se describen los equipos del laboratorio de investigación utilizados en esta segunda parte del proyecto:

- **Analizador de semiconductores Agilent 4156C:**

El analizador de semiconductores es un instrumento que permite registrar las características eléctricas de un circuito o componente electrónico. Permite aplicar:

- **Estrés de tensión constante**: aplica al circuito una tensión constante durante un tiempo determinado y permite registrar las corrientes durante el mismo.
- **Estrés de corriente constante**: aplica una corriente constante durante un tiempo determinado y permite registrar las tensiones.
- **Rampas de tensión**: aplica una rampa de tensión y registra la corriente.
- **Rampas de corriente**: aplica una rampa de corriente y registra la tensión.
- **Secuencias de medidas**: secuencias combinando las medidas anteriores.

Tiene 8 salidas repartidas de la siguiente forma:

- **4 cables SMU**: aplican tensión y miden corriente o viceversa. Son cables triaxiales.
- **2 VMU (Voltage Monitor Unit):** mide tensión sin aplicar corriente. Es un cable coaxial.
- **2 VSU (Voltage Source Unit):** aplica una tensión constante sin medir corriente. Es un cable coaxial.

La gran ventaja de este equipo es que permite aplicar tensiones y medir corrientes muy bajas con gran exactitud (del orden de fA). Está orientado a la caracterización de circuitos y componentes micro y nano electrónicos. A continuación, se muestra el analizador de semiconductores utilizado (Figura III.1):

Figura III.1 – Analizador de seminconductores Agilent 4156C

Capacidades de medición:

- Rango de medidas SMU's:

Voltaje: 2 μV/200 V
Corriente: 10 fA/0.1 A

- Resolución de los SMU's:

Voltaje: 2 μV
Corriente: 10 fA

- Precisión de los SMU's:

Voltaje: 700 μV

Corriente: 3 pA

➢ VMU:
Resolución: 2 µV
Precisión: 200 µV

- **Mesa de puntas:**

Es el soporte de la oblea formada por los dispositivos que se quieren caracterizar. Está formada por:

- **Microscopio**: instrumento que permite localizar la muestra.
- **Chuck**: es el soporte sobre el cual se coloca la oblea. Si se desea realizar medidas en temperatura se utiliza un chuck térmico.
- **Microposicionadores**: son los elementos encargados de poner las puntas en los lugares de medición del circuito adecuados. Permiten realizar movimientos de gran precisión. Se necesita uno por cada punto del circuito donde queramos aplicar señal o medir.

A continuación se muestra una imagen de la mesa de puntas del laboratorio (Figura III.2):

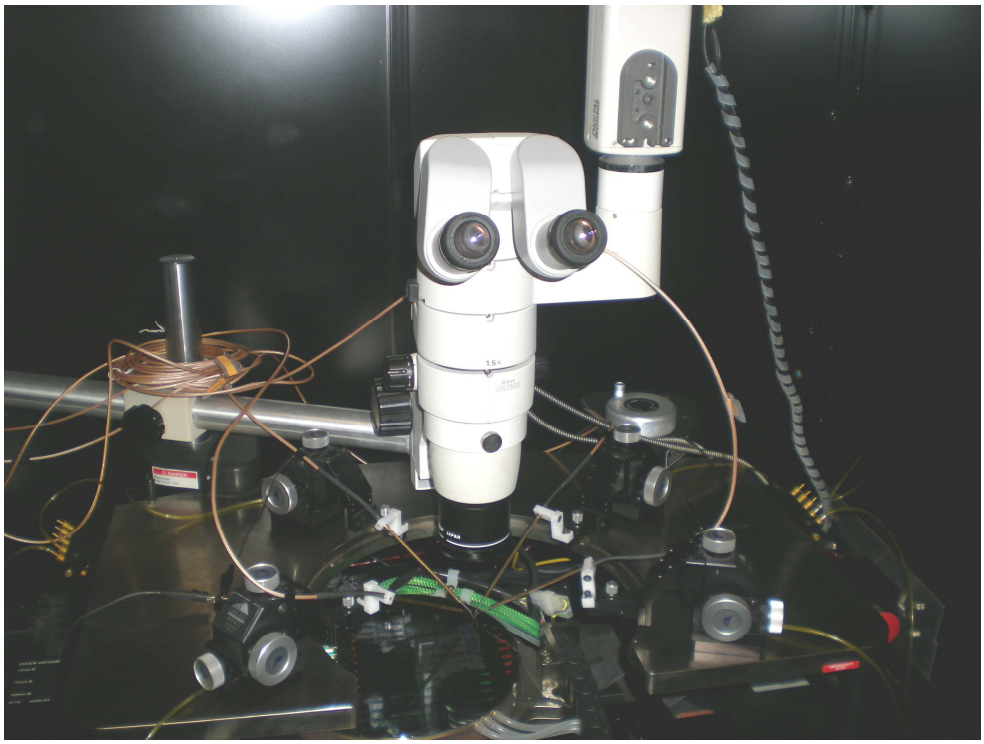


Figura III.2 – Mesa de puntas del laboratorio de investigación

- **Caja de Faraday:**

La mesa de puntas se coloca dentro de una caja de Faraday para proteger la medida de descargas electrostáticas. Para conectar la mesa de puntas con el exterior, en la caja de Faraday hay unos orificios para conectar el cableado.

- **Cable triaxial:**

Es un tipo de cable similar al cable coaxial, pero con el añadido de una capa adicional de aislamiento y una funda de segundo conductor. Posee un mayor ancho de banda y mayor rechazo a las interferencias, pero es más caro. Está formado por:

  - Un conductor central que lleva la señal
  - El conductor de en medio está al mismo potencial que el conductor central para eliminar corrientes de fugas
  - El conductor exterior, que es el de tierra

El analizador se conecta al circuito que queramos caracterizar a través de cables SMU (Source Monitor Unit), Los SMU's son cables triaxiales que permiten realizar las dos funciones comentadas anteriormente en un solo cable. El esquema del terminal SMU se muestra en la Figura III.3:

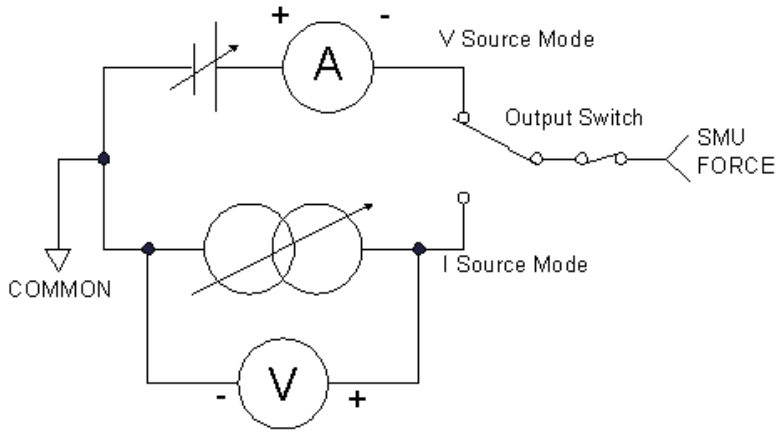


Figura III.3 – Esquema del terminal SMU

En nuestro caso, utilizaremos el cable triaxial para enviar señales de tensión a través de los SMU's y medir las corrientes. A continuación se muestra una imagen comparativa entre el cable triaxial y el coaxial, en la que se aprecia la mayor precisión del primero (Figura III.4):

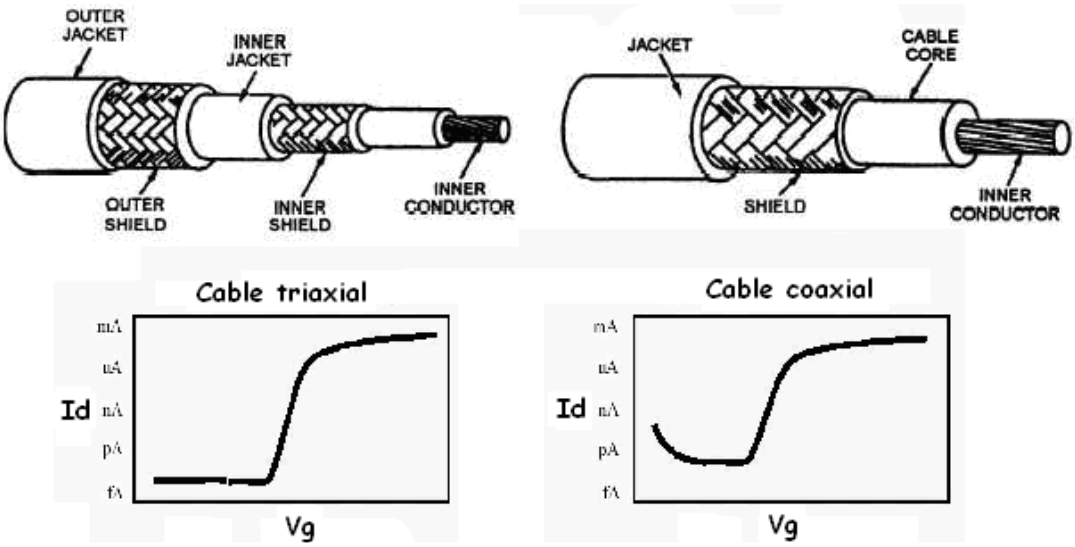


Figura III.4 – Comparativa entre cable coaxial y triaxial

3. FUNCIONAMIENTO DE LA APLICACIÓN

3.1 COMUNICACIÓN CON LOS INSTRUMENTOS

La comunicación del instrumento se realiza de manera similar a como se ha trabajado en el laboratorio docente (apartado II 3.1). La principal diferencia es que la comunicación con el instrumento a través del bus GPIB se realiza mediante los comandos SCPI y también GPIB Command Reference (comandos GPIB propios del analizador) [11], que tienen más versatilidad y pueden realizar todas las funciones propias del analizador. A continuación, se muestra la estructura que se ha utilizado para la comunicación:

- **Inicio de los instrumentos:** para poder llevar a cabo el control de los instrumentos es necesario inicializarlos, es decir, asignar una dirección GPIB que identifique el instrumento dentro del bus. Ejemplo:

```
%con estas líneas abriríamos el canal GPIB del analizador:
analizador=gpib('ni',0,27);
fopen(analizador);
```

  Mediante estas líneas, se identifica un instrumento de la marca National Instruments (ni) que se encuentra en la dirección GPIB número 27 y que estará controlado por la tarjeta controladora GPIB situada en el PC que tiene la dirección 0 (dirección por defecto). Todo ello quedará almacenado en la variable "analizador". Mediante la instrucción "fopen", se inicializa el instrumento y queda listo para ser controlado a través del bus.

- **Enviar información:** se realiza mediante el comando Matlab "fprintf", a través del cual se envían los comandos tanto SCPI como Command Reference. Ejemplo:

```
%Ejemplo norma SCPI:
%con estas líneas se envía la orden al analizador de
semiconductores que configuramos el SMU determinado en la
variable "NSMU" en modo voltaje o intensidad según indique la
variable "type":
cadena=sprintf(':PAGE:CHAN:SMU%d:FUNC %s',NSMU,type);
fprintf(obj_gpib,cadena)

%Ejemplo norma GPIB Command Reference:
%con esta línea configuramos los cuatro SMU's para que puedan
realizar medidas de intensidad instantáneas:
fprintf(c,'MM 1,1,2,3,4');
```

- **Lectura de datos:** el comando Matlab utilizado para la lectura de información de los equipos es el "fscanf". Primero, hay que enviarle una pregunta sobre qué variable queremos medir para posteriormente leerla y almacenarla. Ejemplo:

```
%con estas líneas preguntamos al analizador qué valor de
intensidad está pasando por el SMU1 utilizando el modo auto
rango. Posteriormente, almacenamos esta lectura en la variable
"I1":
fprintf(analizador,'TI? 1,0')
I1=fscanf(c);
```

- **Cerrar los instrumentos:** una vez realizadas y almacenadas las medidas, es necesario cerrar los instrumentos utilizados, es decir, dejamos de controlarlos a través del bus. Ejemplo:

```
%con estas líneas cerramos el analizador de semiconductores:
fclose(analizador);
```

De nuevo, y como en el capítulo II, una vez que hemos conseguido realizar las medidas a través del control del bus, debemos procesar los datos para conseguir obtener valores numéricos convencionales y poder así, almacenarlos de forma correcta.

3.2 INTERFAZ GRÁFICA

La aplicación parte, como en la primera parte del proyecto, de una interfaz gráfica de usuario realizada mediante la aplicación GUI de Matlab. En ella, se configuran las señales que queremos enviar al circuito así como todos los parámetros de las mismas. A continuación, se muestra la interfaz gráfica de la aplicación (Figura III.5):

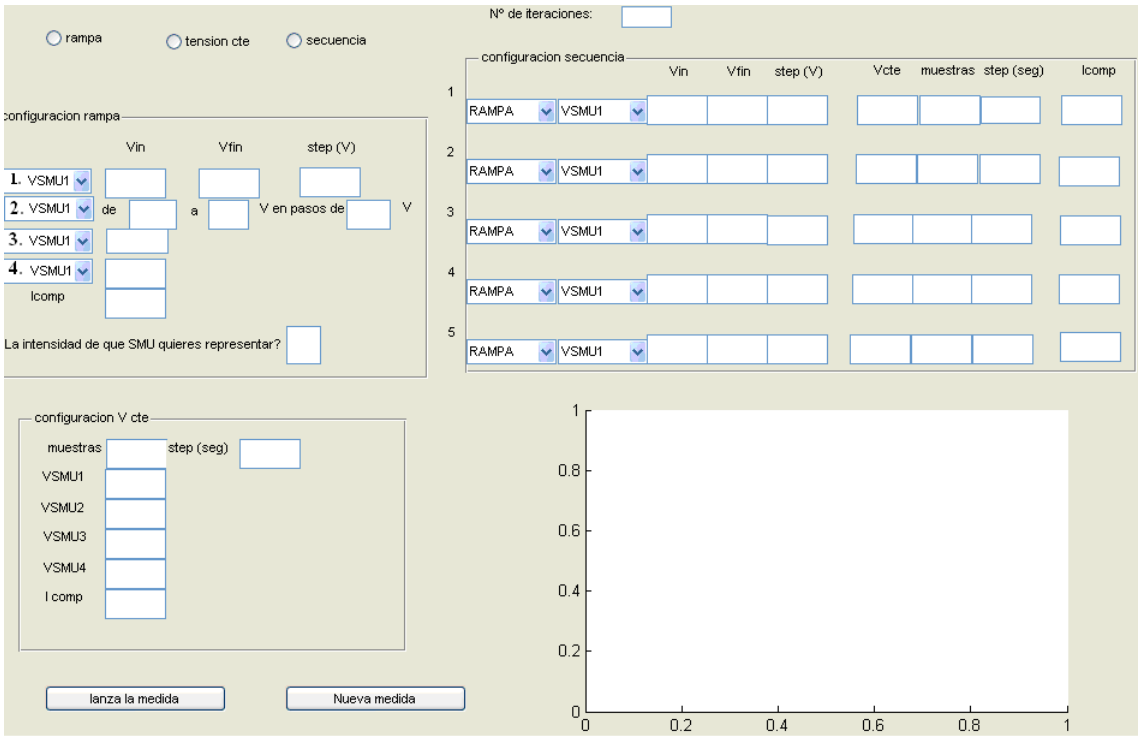


Figura III.5 - Interfaz gráfica de la aplicación

Al abrir el ejecutable, se darán tres opciones a elegir por el usuario: rampa, tensión constante o secuencias, según el tipo de señal que desee transmitir al sistema. A continuación se describen junto con sus características:

- **Rampa:**

  Consiste en la aplicación de una rampa de tensión por uno de los SMU's y de tensiones constantes en los otros tres, pudiendo uno de ellos ser variable. Mediante un menú desplegable, el programa permite seleccionar el SMU sobre el que se debe aplicar la rampa. Una vez finalizada la medida, se representa en la gráfica de la

interfaz la rampa de tensión en función de la intensidad del SMU que elijamos. Para los cuatro SMU's, la variable "Icomp" determina la intensidad máxima que puede pasar por cada uno de ellos, como medida de protección para el circuito.

Los cuatro SMU's disponibles en el analizador son modificables como se describe a continuación.

1. Los datos de entrada que se deben introuducir para aplicar la rampa son:

   - Vin1: tensión punto de partida de la rampa
   - Vfin1: tensión final de la rampa
   - Step1: saltos (en V) de la rampa

   Ejemplo: Vin = 0; Vfin = 5; step = 0.5: se transmitirá al circuito una rampa de tensión de 0-5V en saltos de 0.5V, tomando 10 muestras de intensidad.

2. Adicionalmente, el programa permite cambiar el valor de tensión constante de uno de los SMU's aplicando una nueva rampa (segundo barrido):

   - Vin2: tensión inicial a introducir en el SMU
   - Vfin2: tensión final a introducir en el SMU
   - Pasos: intervalos (en V) de la tensión introducida en el SMU

   Ejemplo: continuando con los mismos parámetros del ejemplo anterior y Vin2 = 1V; Vfin2 = 3 V; step2 = 1V: se transmitiría al circuito una tensión de 1V por el SMU elegido y simultáneamente se realizaría la rampa de 0-5V de tensión por el otro SMU configurado en la rampa, midiendo la intensidad del SMU que deseemos. Una vez finalizada la rampa, se colocarían 2V en el segundo SMU configurado y de nuevo, simultáneamente se realizaría la rampa de tensión por el otro SMU. En este caso particular, por último, se colocarían 3V en el segundo SMU configurado y se volvería a realizar la rampa de tensión, midiendo de nuevo la intensidad elegida.

3.y 4. Tensión constante: durante el tiempo que dura la rampa, se pone una tensión constante en el SMU deseado. Esta tensión se aplicará constante durante todo el proceso. Parámetros a introducir:

   - Tensión constante: valor (en V) de la tensión a configurar

- **Tensión constante:**

Esta opción configura una tensión constante en cada uno de los cuatro SMU's durante un tiempo determinado y recoge la corriente a través de ella. La duración del CVS viene determinada por los siguientes parámetros:

   - Muestras: es el número de muestras de tensión constante que se transmitirá al circuito

- Step (segundos): es el intervalo de tiempo entre muestra y muestra

Los cuatro SMU's disponibles en el analizador son configurables como se describe a continuación. Para todos ellos, la variable "Icomp" configura la intensidad máxima que puede pasar por cada uno de ellos, como medida de protección para el circuito.

- VSMU1: valor de la tensión que será transmitida al circuito por el smu1
- VSMU2: valor de la tensión que será transmitida al circuito por el smu2
- VSMU3: valor de la tensión que será transmitida al circuito por el smu3
- VSMU4: valor de la tensión que será transmitida al circuito por el smu4

Ejemplo: muestras = 100; step = 0.1; VSMU1 = 5V; VSMU2 = 3V; VSMU3 = 1V; VSMU4 = 2V: se aplicaría una tensión constante en los SMU's indicados con los valores introducidos de 100 muestras espaciadas entres sí 0.1 segundos. El tiempo total en el que estaríamos aplicando tensión constante en este caso sería de 10 segundos. Una vez finalizado este tiempo, se representarían las intensidades que pasan por los 4 SMU's, siempre y cuando las tensiones introducidas fueran diferentes de 0V.

- **Secuencias:**

Esta opción realiza iteraciones de rampas de tensión y/o tensiones constantes configurables por el usuario. Se pueden realizar en cada secuencia un máximo de 5 medidas distintas seguidas. Para cada una de ellas, se puede elegir entre si queremos aplicar una rampa, una tensión constante o nada. En caso de elegir una de las dos primeras, hay que introducir los valores de configuración comentados anteriormente.

La variable "nº iteraciones" selecciona el número de veces que queremos que se repitan las secuencias. Por último y de nuevo, la variable "Icomp" configura la intensidad máxima que puede pasar por cada uno de de los SMU's, como medida de protección para el circuito.

Ejemplo:

- nº iteraciones = 17
- configuración secuencia:
  1. rampa por el smu4
  2. rampa por el smu3
  3. tensión constante por el smu1
  4. none
  5. none

Con estos parámetros de entrada, se transmitirá primero una rampa por el SMU4 con la configuración de los parámetros introducida. Posteriormente, se transmitirá una rampa por el smu3 con los parámetros configurados y por último, se transmitirá una tensión constante por el SMU1 con la configuración elegida.

En este caso, este proceso se repetirá diecisiete veces. Las intensidades medidas por los SMU's correspondientes para el caso de las secuencias, se representarán en una figura a parte y en gráficas independientes para poder así comparar los resultados obtenidos.

3.3 ESTRUCTURA DE PROGRAMA

En este apartado, analizaremos la estructura del programa de cada una de las tres opciones que el usuario puede escoger al iniciar la interfaz gráfica: rampa, tensión constante o secuencias. Para cada una de ellas, explicaremos detalladamente los siguientes apartados:

- **Estructura**: Analizaremos las subpartes en las que se divide el programa principal y mostraremos la estructura general mediante el diagrama de flujo del programa.

- **Funciones utilizadas**: se enumerarán y explicarán todas las funciones que forman parte de cada apartado, describiendo su función dentro del programa y enumerando los argumentos que las constituyen.

- **Ejemplo**: por último, mostraremos un ejemplo completo de cada opción, en el que podremos ver las gráficas resultantes de cada parte. En este caso, nuestro circuito a analizar será un transistor. Características:

    - Transistor P-MOS
    - 0,3 x 0,3 µm de canal

  Hemos conectado los SMU's de la siguiente forma (Figura III.6):

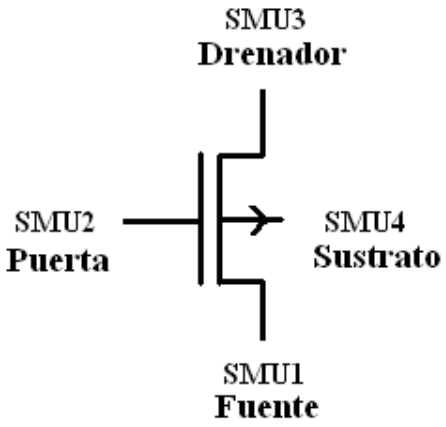


Figura III.6 – Conexionado de SMU's al transistor

El programa principal parte de la base de la elección del usuario en la interfaz gráfica de una de las tres opciones que podemos realizar mediante esta aplicación, tal y como se puede ver en el diagrama de flujo de esta parte que se muestra a continuación (Figura III.7):

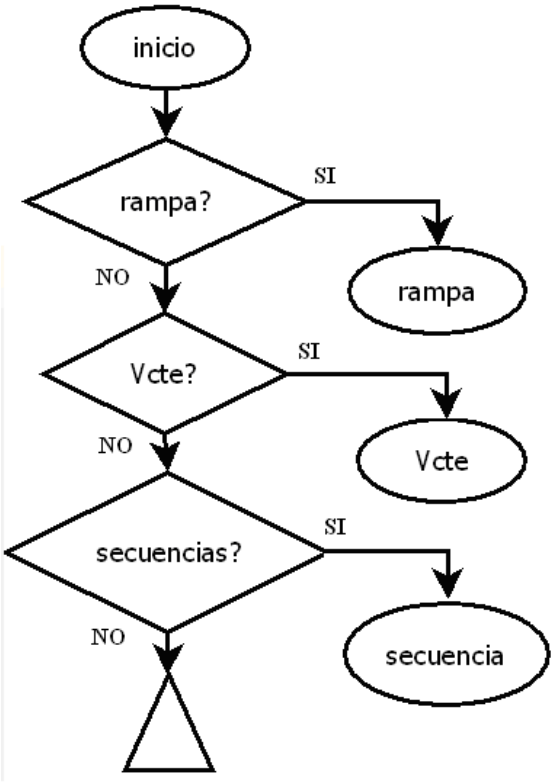


Figura III.7 – Diagrama de flujo inicial

A continuación, se describe la estructura del programa de cada una de estas tres posibles opciones:

1. **Rampa de tensión:**

   - Estructura:

     1) Leemos el valor de todas las variables introducidas así como los SMU's de salida configurados por el usuario.
     2) Inicializamos el analizador, es decir, lo conectamos al PC mediante el bus GPIB asignándole una dirección.
     3) Configuramos el analizador en modo "sweep" para que podamos enviar rampas de tensión al circuito por el SMU's seleccionado.
     4) Abrimos los 4 SMU's para poder enviar tensiones y medir intensidades.
     5) Ponemos los valores de tensión introducidos por el usuario en los SMU's correspondientes y lanzamos la medida.
     6) Medimos y almacenamos la intensidad que queremos analizar y la representamos en una gráfica en la que se muestra en el eje de las "x" la rampa de tensión y en el eje de las "y" la

intensidad elegida. Cabe tener en cuenta que todas las gráficas se auto escalan por defecto.

7) Cerramos el analizador y guardamos datos.

A continuación, se muestra el diagrama de flujo de esta parte (Figura III.8):

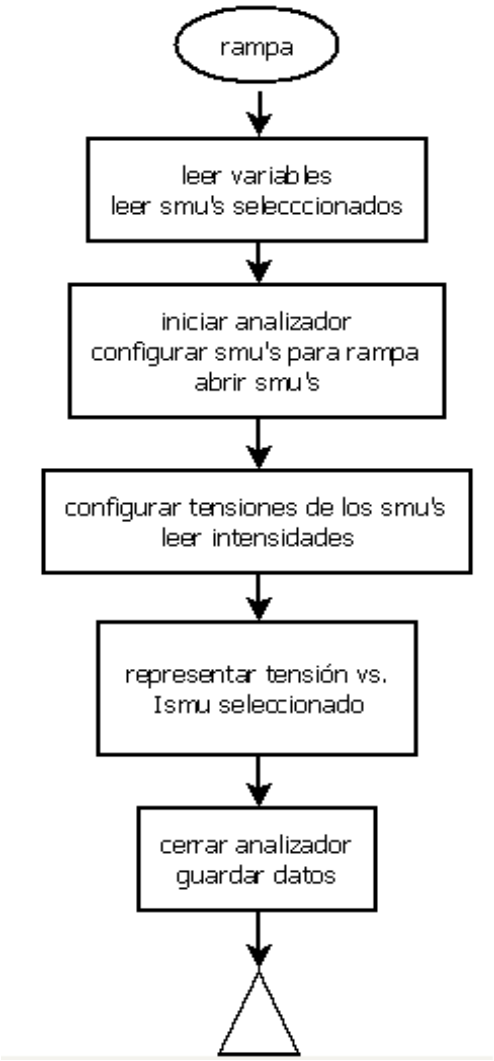


Figura III.8 – Diagrama de flujo de la rampa de tensión

- Funciones utilizadas:

  - Iniciayconfig_rampa2:

    - Inicia el analizador de semiconductores y lo conecta al ordenador mediante el bus GPIB.
    - Configura el analizador para que pueda transmitir rampas de tensión por el SMU seleccionado. Los demás SMU's se configuran como tensiones constantes.
    - Abre los cuatro SMU's y los configuramos para que puedan enviar tensión y recoger corrientes del circuito. Argumentos que necesita:

- Número de los SMU's donde queremos aplicar la rampa y las tensiones constantes.

❖ Inicia_ins_2:

➢ Conecta el analizador de semiconductores con el PC a través del bus GPIB. Argumentos que necesita:

- Número de la dirección GPIB del analizador

❖ Conf_SMU:

➢ Establece los SMU's para que envíen tensiones y configura cada uno de ellos para que las tensiones que envían sean tensiones constantes o rampas de tensión. Argumentos que necesita:

- SMU seleccionado. Este argumento es un número entero del 1 al 4.
- Indicar si vamos a transmitir tensiones o intensidades. En nuestro caso, indicaremos con una "V" que deseamos enviar tensiones
- Indicar si las tensiones a enviar son constantes ("CONS") o variables ("VAR1"). Cabe tener en cuenta que el analizador sólo puede enviar dos rampas de tensión por dos SMU's diferentes de manera simultánea.
- Identificador del analizador

o RVS2:

➢ Una vez configurados los SMU's, esta función realiza todo el proceso aplicar las tensiones elegidas en los SMU's seleccionados, lanzar la medida al circuito y recoger y almacenar la intensidad configurada. Por último, se representa la gráfica en la interfaz. Todos los argumentos necesarios por esta función excepto el identificador del analizador son valores introducidos por el usuario que se recogen directamente de la interfaz gráfica:

- Tensión inicial de la rampa
- Tensión final de la rampa
- Pasos (en V) que queremos que salte la rampa

En caso de querer realizar más de una rampa con diferentes tensiones constantes en uno de los SMU's, serán necesarios los siguientes tres parámetros, en caso contrario, pondremos el mismo valor en los tres.

- Tensión contante inicial del SMU
- Tensión constante final del SMU
- Pasos de tensión (en V) en el que queremos que cambie.
- Tensión constante del otro SMU
- Tensión constante del otro SMU
- Intensidad máxima que puede pasar por cada uno de los SMU's
- Los SMU's de las tensiones de salida
- Identificador del analizador
- El número del SMU por el cual quiero leer, almacenar y representar la intensidad

- Ejemplo: Obtención de la gráfica Id-Vg, es decir la representación de la tensión aplicada en la puerta en función de la intensidad que pasa por el drenador. Los parámetros introducidos así como la gráfica resultante se muestran en las figuras III.9 y III.10 respectivamente:

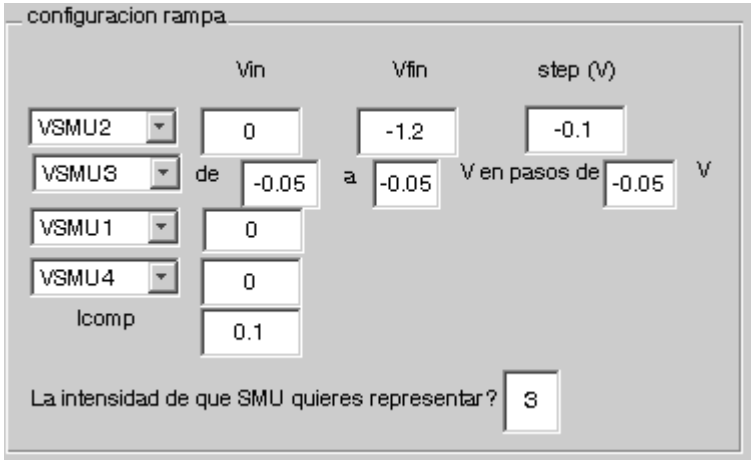


Figura III.9 – Parámetros introducidos

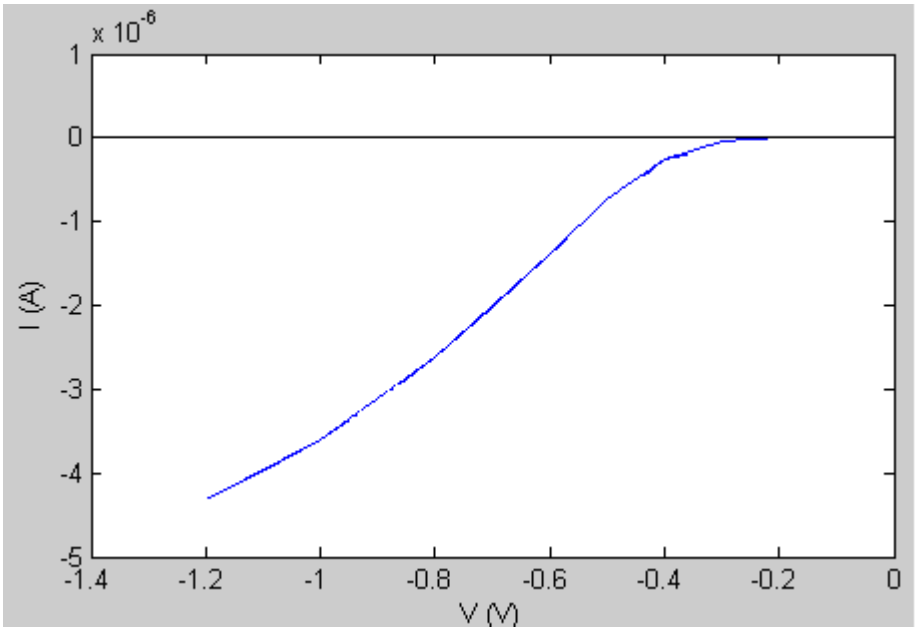


Figura III.10 – Gráfica resultante

2. Tensiones constantes:

- Estructura:

1) Leemos todas las variables introducidas por el usuario.
2) Inicializamos el analizador.
3) Configuramos el analizador en modo "sampling" para que podamos enviar tensiones constantes al circuito. También establecemos que los 4 SMU's van a enviar tensiones constantes.
4) Abrimos los 4 SMU's para poder enviar tensión y leer intensidad.
5) Ponemos los valores de tensión introducidos por el usuario en los SMU's correspondientes y lanzamos la medida.
6) Medimos y almacenamos las intensidades que pasan por los cuatro SMU's. Una vez finalizada la medida, representamos en la interfaz la gráfica resultante, mostrando el tiempo de duración en el eje de las "x" y las intensidades de los cuatro SMU's en el eje de las "y".
7) Por último, cerramos el analizador y guardamos los datos.

A continuación, se muestra el diagrama de flujo de esta parte (Figura III.11):

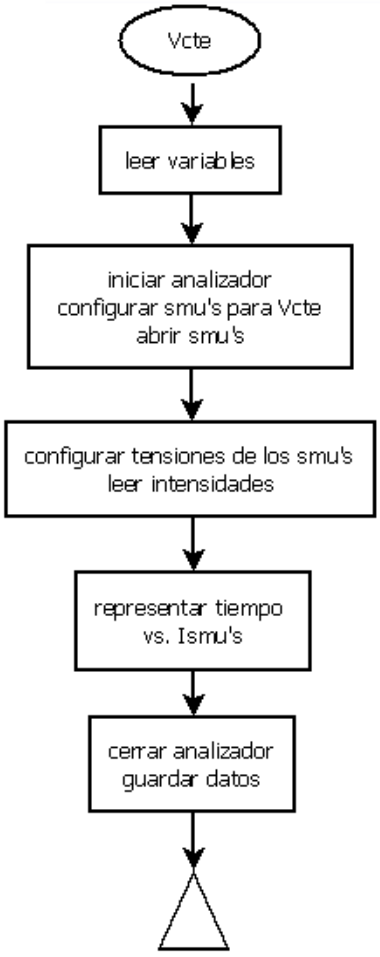


Figura III.11 – Diagrama de flujo de la tensión constante

- Funciones utilizadas:

  - Iniciayconfig_Vcte2:

    - Inicia el analizador y lo conecta al PC mediante el bus GPIB.
    - Configura el analizador para poder enviar tensiones constantes por los 4 SMU's.
    - Abre los SMU's y los configura para que puedan enviar tensión y medir corriente del circuito.

    Esta función no necesita argumentos.

  - Inicia_ins_2:

    - Realiza lo mismo que para el caso de la rampa.

  - Conf_SMU:

    - Es la misma función que para el caso anterior, con la única diferencia que ahora indicaremos con el argumento "CONS" que deseamos que todos los SMU's envíen tensiones constantes.

  - CVS2:

    - Esta función realiza todo el proceso de aplicar las tensiones elegidas para cada SMU, lanzar la medida al circuito y medir y almacenar las cuatro intensidades. Como en el caso de la rampa, los argumentos necesarios excepto el identificador del analizador son valores recogidos directamente de la interfaz de usuario.Argumentos necesarios:
      - El número de muestras que deseamos que tenga nuestra medida.
      - El intervalo de tiempo en que queremos espaciar las muestras.
      - Las tensiones constantes de los cuatro SMU's.
      - La intensidad máxima que puede pasar por cada uno de ellos.

- Ejemplo: Aplicamos tensiones constantes en los SMU's. Este procedimiento sirve para degradar el transistor y comparar sus propiedades. Los valores introducidos en la interfaz así como la gráfica resultante se muestran a continuación en las figuras III.12 y III.13 respectivamente:

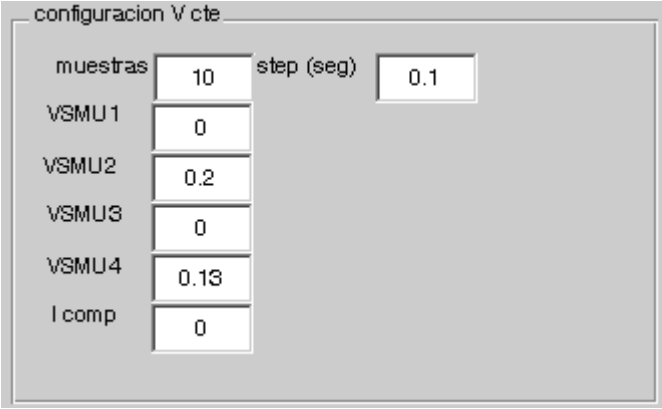


Figura III.12 – Valores introducidos en la interfaz

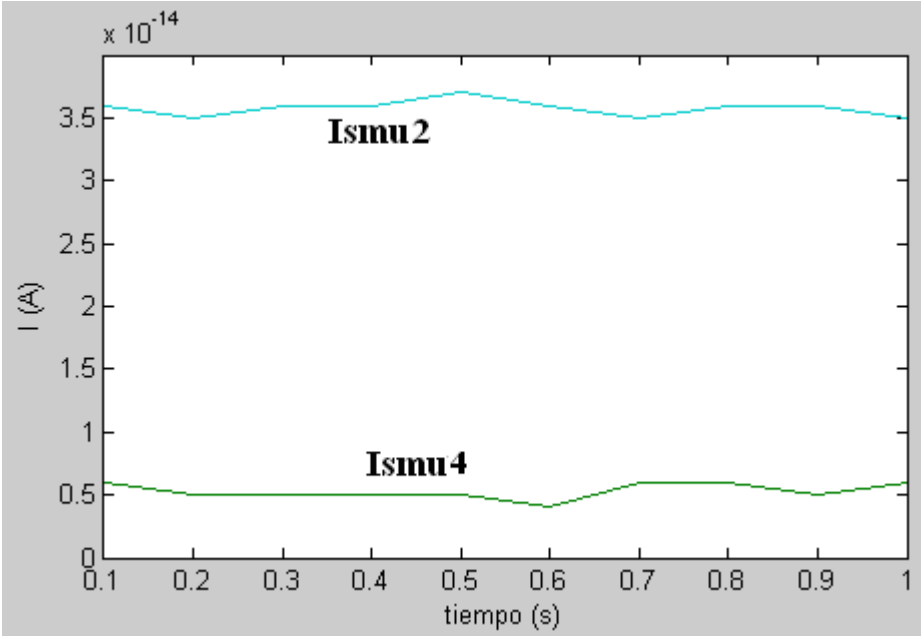


Figura III.13 – Gráfica resultante

3. Secuencias:

Como ya se ha comentado anteriormente, eligiendo esta opción, se pueden configurar un máximo de cinco secuencias seguidas distintas. Para cada una de ellas, el usuario podrá elegir entre enviar una rampa, una tensión constante al circuito o no realizar nada. Escogiendo una de las dos primeras, el programa tendrá para cada caso muchas similitudes con el primer y segundo punto de este apartado. La diferencia recabará en que las funciones aquí utilizadas son ligeramente más sencillas, puesto que únicamente podremos configurar uno de los SMU's por secuencia. A continuación, se explica la estructura del programa en caso de elegir una rampa o una tensión constante:

1.1 Rampa:

- Estructura:

1) Leemos las acciones (rampa, tensión constante o nada) que el usuario quiere realizar, así como los SMU's por los cuales deben realizarse.
2) Inicializamos el analizador
3) Leemos el valor de todas las variables introducidas
4) Seleccionamos el analizador en modo "sweep" y configuramos el SMU elegido para que pueda enviar una rampa de tensión.
5) Abrimos los 4 SMU's para poder enviar tensión y medir intensidad
6) Lanzamos la medida con la rampa de tensión en el SMU seleccionado y medimos la intensidad que pasa por él.
7) Hacemos un reset al analizador y guardamos los datos.

A continuación, se muestra el diagrama de flujo de esta parte, tanto para la rampa como para la tensión constante (Figura III.14):

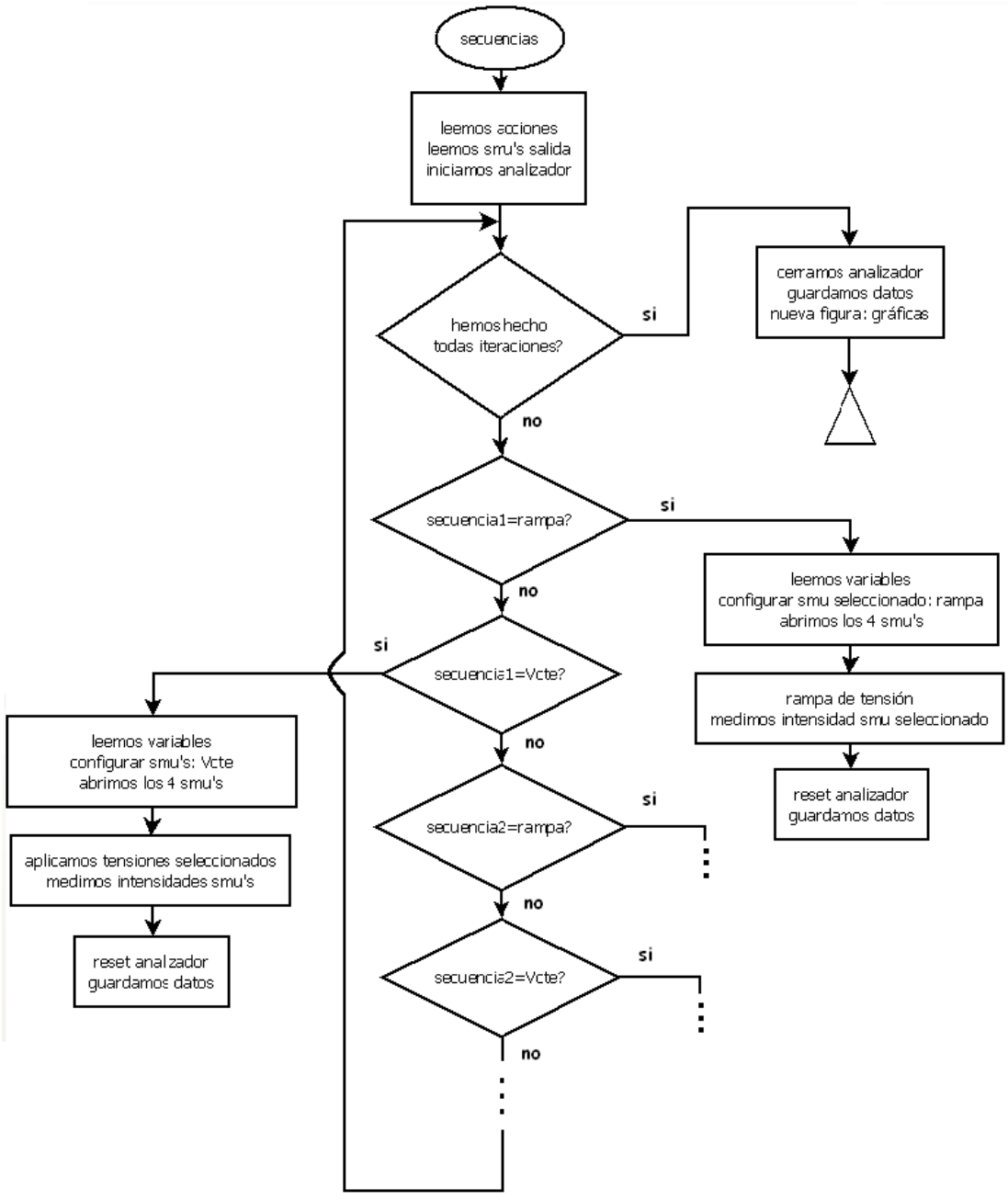


Figura III.14 – Diagrama de flujo de las secuencias

- Funciones utilizadas:

  - Inicia_ins_2:

    - Es la misma función descrita anteriormente.

  - Iniciayconfig_rampa3:

    - Selecciona el analizador en modo “sweep”
    - Configura el SMU de salida seleccionado para que pueda enviar una rampa
    - Abre los cuatro SMU’s

    Argumentos que necesita:

    - El SMU de salida seleccionado por en la interfaz
    - El identificador del analizador

    ❖ Conf_SMU:

    - Es la misma función que para la Rampa del caso 1, pero pudiendo configurar sólo el SMU seleccionado con la variable “VAR1”.

  - RVS3:

    - Esta función realiza el proceso de aplicar la rampa de tensión en el SMU seleccionado, medir y almacenar la intensidad que circula a través de él.
      Argumentos que necesita:

    - Tensión inicial de la rampa
    - Tensión final de la rampa
    - Pasos (en V) que queremos que salte la rampa
    - El SMU de salida
    - La intensidad máxima que puede pasar por él
    - El identificador del analizador

1.2 Tensión constante:

- Estructura:

  1) Leemos las acciones (rampa, tensión constante o nada) que el usuario quiere realizar, así como los SMU’s por los cuales deben realizarse.
  2) Inicializamos el analizador
  3) Leemos el valor de todas las variables introducidas
  4) Seleccionamos el analizador en modo “sampling” y configuramos el SMU elegido para que pueda enviar una tensión constante.
  5) Abrimos los 4 SMU’s para poder enviar tensión y medir intensidad

6) Lanzamos la medida con la tensión en el SMU seleccionado y medimos la intensidad que pasa por él.
7) Hacemos un reset al analizador y guardamos los datos.

- Funciones utilizadas:

  - Inicia_ins_2:

    - Es la misma función descrita anteriormente.

  - Iniciayconfig_Vcte3:

    - Selecciona el analizador en modo "sampling"
    - Configura el SMU de salida seleccionado para que pueda enviar una tensión constante
    - Abre los cuatro SMU's

    Argumentos que necesita:

    - El SMU de salida seleccionado por en la interfaz
    - El identificador del analizador

    ❖ Conf_SMU:

    - Es la misma función que en el caso de haber seleccionado la opción tensión constante del caso 2, pero pudiendo configurar sólo el SMU seleccionado mediante la variable "CONS"

.

  - CVS3:

    - Esta función realiza el proceso de aplicar la tensión en el SMU seleccionado, medir y almacenar la intensidad que circula a través de él.

      Argumentos que necesita:

      - Número de muestras que queremos que tenga nuestra medida
      - El intervalo de tiempo de espaciado entre muestras
      - La tensión constante de salida
      - El SMU de salida
      - La intensidad máxima que puede pasar por él
      - El identificador del analizador

- Ejemplo: realizaremos una rampa y una tensión constante. El número de iteraciones que se repetirá este proceso es de 2. Los valores introducidos así como las gráficas resultantes se muestran a continuación en las figuras III.15 y III.16 respectivamente:

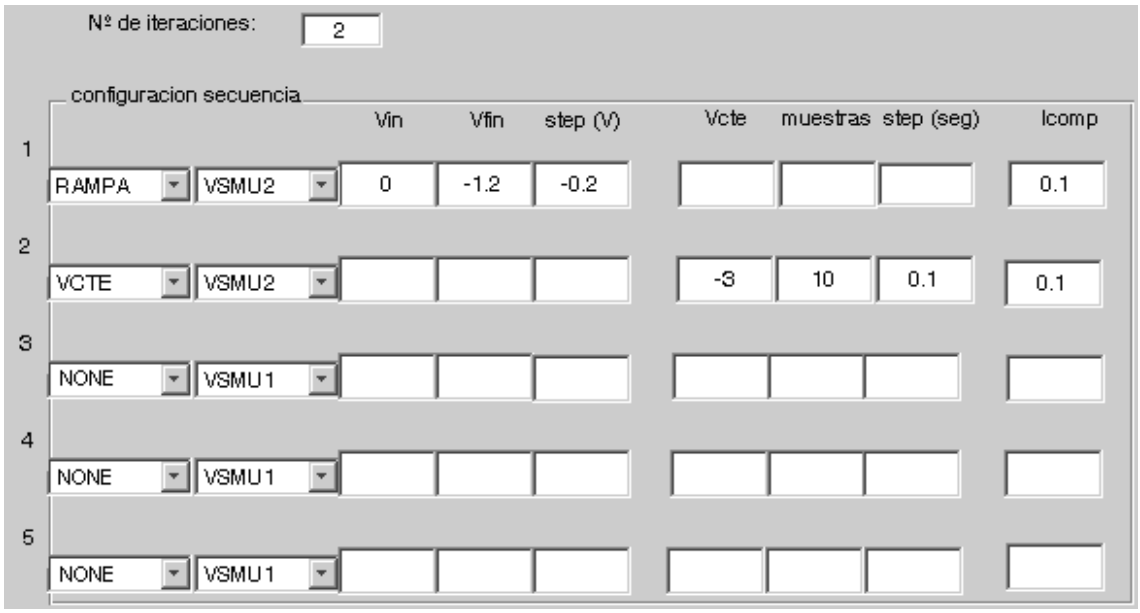


Figura III.15 – Valores introducidos en la interfaz

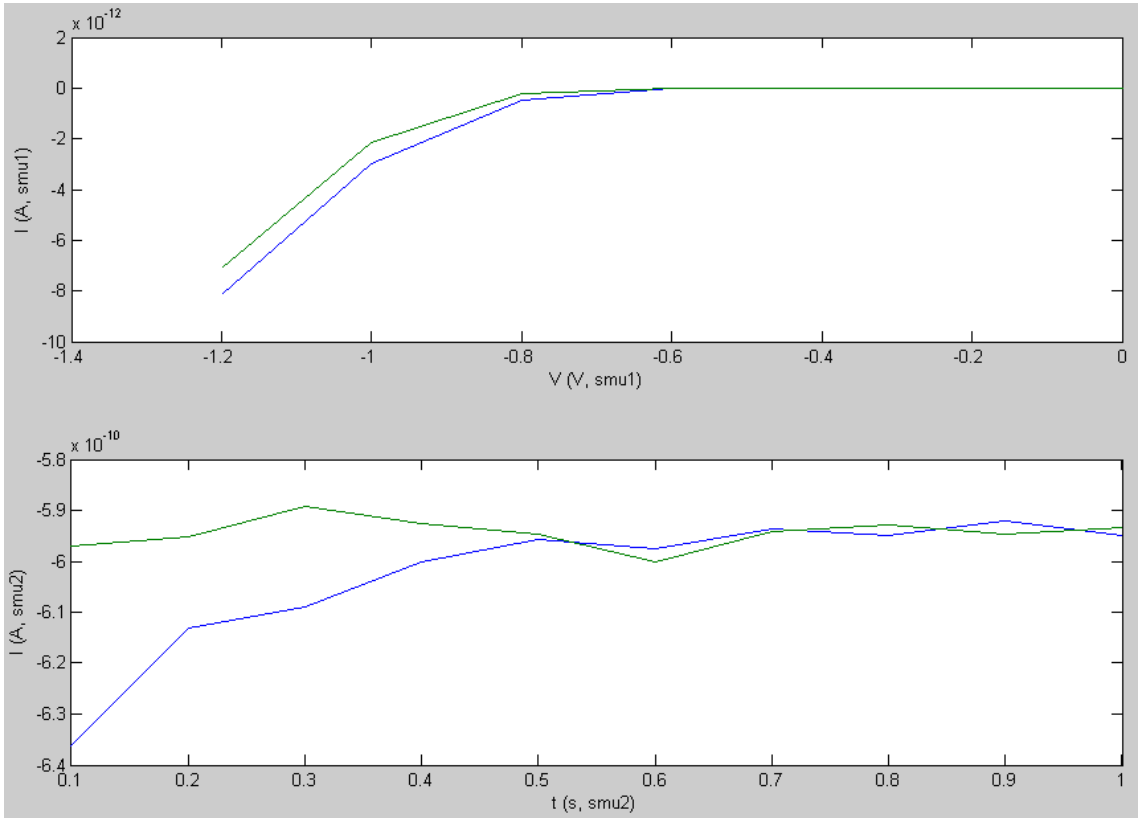


Figura III.16 – Gráfica resultante

3.4 MODO TEXTO

Mediante la opción "secuencias" sólo podemos realizar cinco secuencias independientes. Puesto que para algunas aplicaciones puede ser interesante y/o necesario realizar más, hemos creado con Matlab una función llamada "caract_pmos" en la cual se pueden hacer infinitas secuencias distintas en forma de rampa. Esta opción

no es tan intuitiva como la interfaz gráfica, pero una vez conocido el procedimiento es potente recurso. Para utilizarla, realizaremos los siguientes pasos:

1. Abrir el archivo Matlab donde está la función
2. Introducir los parámetros que configuran la rampa de la función RVS4. Son los siguientes:

    - Vin: tensión inicial de la rampa
    - Step: pasos (en V) en que separamos la rampa
    - Vfin: tensión final de la rampa
    - smu_rampa: nº del SMU por el que enviamos la rampa
    - Vnext1: tensión que aplicaremos en el primer SMU libre
    - Vnext2: tensión que aplicaremos en el segundo SMU libre
    - Vnext3: tensión que aplicaremos en el tercer SMU libre
    - Icomp: intensidad máxima que puede pasar por los SMU's
    - c: es el objeto GPIB identificador del analizador

Ejemplo: queremos obtener la rampa Id_Vd del transistor, por lo que aplicaremos una rampa en el drenador (SMU3) y una tensión constante en la puerta (SMU2). Representaremos la tensión del drenador en función de la intensidad que pasa por él. En consecuencia, configuramos la función RVS4:

- Vin: 0V
- Step: -0.1V
- Vfin: -1.2V
- smu_rampa: 3
- Vnext1: 0V
- Vnext2: -0.05V
- Vnext3: 0V
- Icomp: 0.05A

Con esta configuración aplicaremos una rampa de tensión en el SMU3 de 0 a -1.2V espaciados entre sí -0.1V, una tensión constante de -0.05V en el SMU2 y 0V en los SMU's 1 y 4. A continuación, mostramos el texto íntegro de la función "caract_pmos":

```
%limpiamos todas las variables
clear all

%iniciamos el analizador
c=inicia_ins_2(27)

%hacemos la rampa:
RVS4(Vin,step,Vfin,smu_rampa,Vnext1,Vnext2,Vnext3,Icomp,c)
[I1,V1]=RVS4(0,-0.1,-1.2,3,0,-0.05,0,0.05,c)

%cerramos el analizador
fclose(c)

%gráfica
plot(V1,d(:,3))

%guardamos los datos en el fichero output2
```

```
save 'output2.mat'
```

Con esta configuración, obtenemos la siguiente gráfica Id-Vg (Figura III.17):

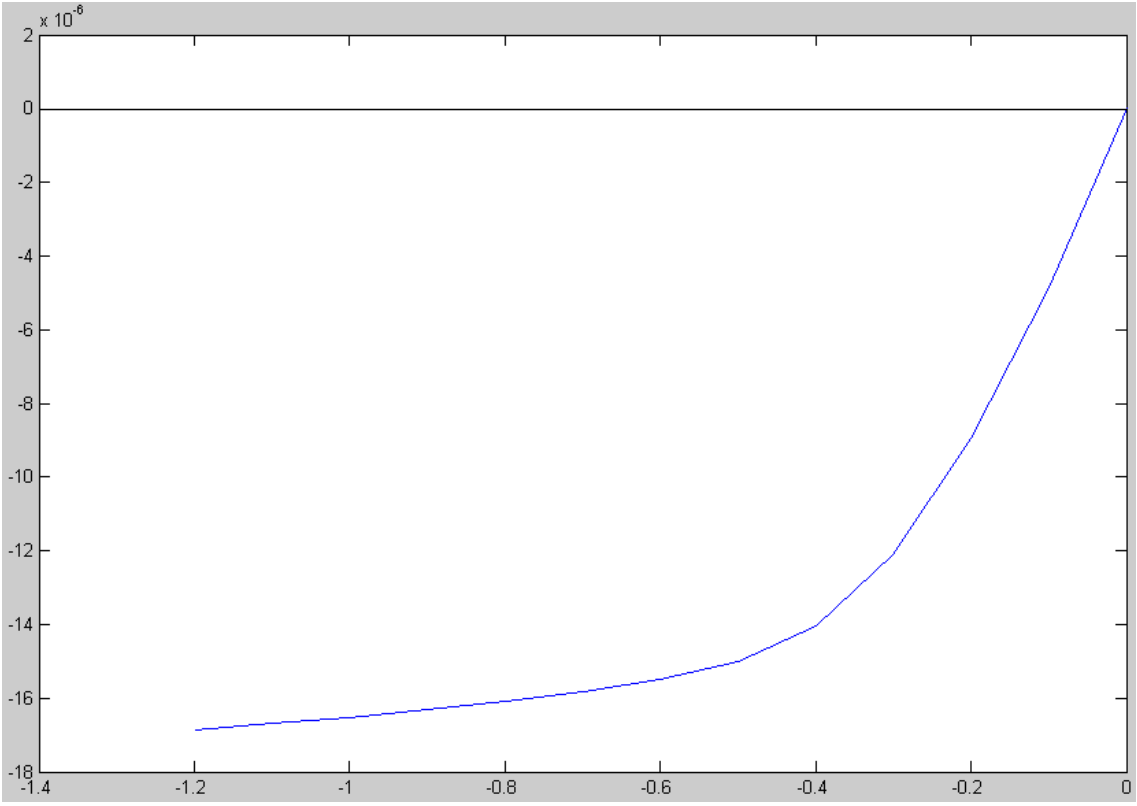


Figura III.17 – Gráfica Id-Vd obtenida mediante la función "caract_pmos"

Hay que tener en cuenta que podemos hacer la secuencia de rampas tan larga como deseemos. Para realizar más, tan sólo debemos copiar la línea de la función RVS4 tantas veces como queramos.

Finalmente, y volviendo a la interfaz gráfica, en caso de querer realizar más de una medidas independientes, el botón "Nueva medida" resetea la aplicación. A continuación, se muestra el diagrama de flujo (Figura III.18):

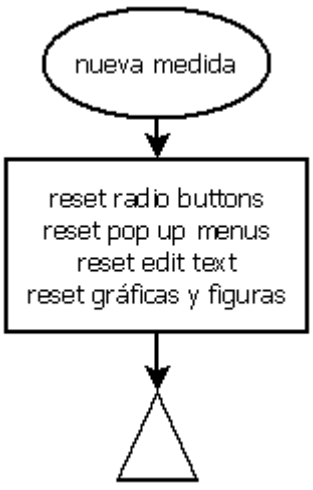


Figura III.18 – Diagrama de flujo de Nueva medida

4. RESULTADOS

El control de la aplicación por parte del usuario es realmente sencillo e intuitivo gracias a la interfaz gráfica. Cualquiera que nunca haya visto un analizador de semiconductores, puede enviar señales para analizar un circuito y obtener las gráficas que desee. El procedimiento genérico de control de la aplicación se muestra a continuación en un desglose de pasos:

1. para empezar, el usuario debe situar las puntas de la mesa de puntas en los puntos donde desee aplicar las tensiones y/o realizar medidas. También debe conectar las salidas de los 4 SMU's del analizador a las conexiones adecuadas de la mesa de puntas.
2. se enciende el analizador y se abre la interfaz gráfica. En ella, el usuario escoge entre rampa, tensión constante o secuencias y configura los parámetros correspondientes.
3. al iniciar la aplicación, esta configuración es transmitida al analizador de semiconductores mediante el bus GPIB utilizando los comandos SCPI y GPIB Command Reference. Éste, envía las tensiones al circuito y lee las intensidades en el punto deseado. Éstas son procesadas y almacenadas en el PC mediante Matlab.
4. una vez el análisis ha finalizado, se representan las gráficas elegidas.

A continuación, se muestra el esquema de bloques del funcionamiento de la aplicación (Figura III.19):

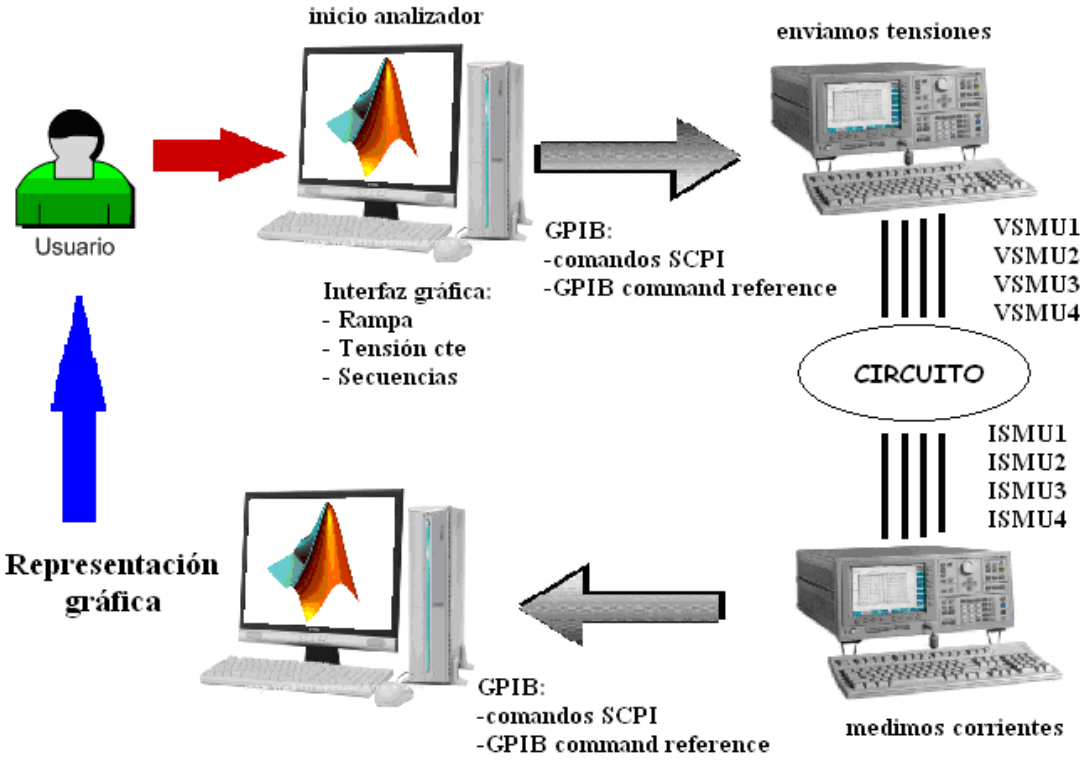


Figura III.19 – Esquema de bloques del funcionamiento de la aplicación

Para nuestro ejemplo, de nuevo, el sistema a analizar será un transistor. Realizaremos su caracterización completa mediante la opción rampa con el fin de obtener las gráficas Id-Vg y Id-Vd. Características del transistor:

- transistor P-MOS
- 0,3 x 0,3 µm de canal

A continuación, se muestra la oblea donde se encuentran el transistor que vamos a analizar (Figura III.20):

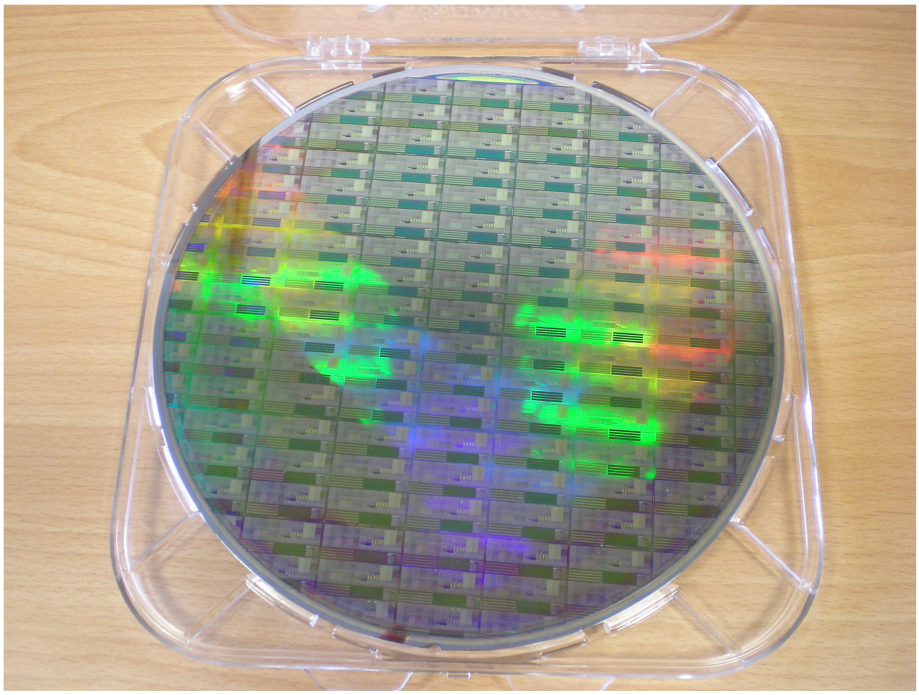


Figura III.20 – Oblea de transistores

Cabe tener en cuenta que al ser transistores de tamaño del orden de los µm, sólo se puede acceder a ellos a través de un microscopio. Puesto que los transistores son para investigación, los pads son mucho más grandes que el propio transistor con el objetivo de poder pinchar las puntas de manera relativamente sencilla. A continuación, se muestran dos imágenes de la oblea con distintos zoom donde se pueden apreciar los pads del transistor que vamos a caracterizar (Figuras III.21 y III.22):

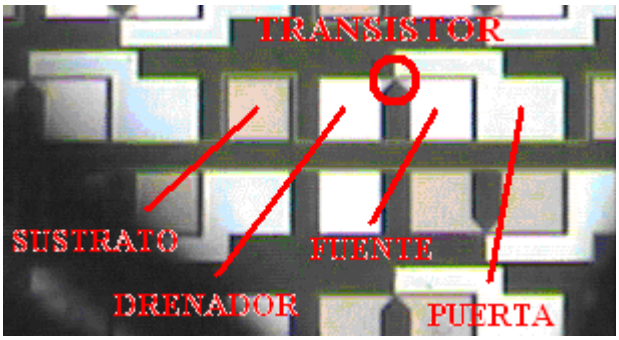


Figura III.21 – Detalle del transistor visto a través del microscopio

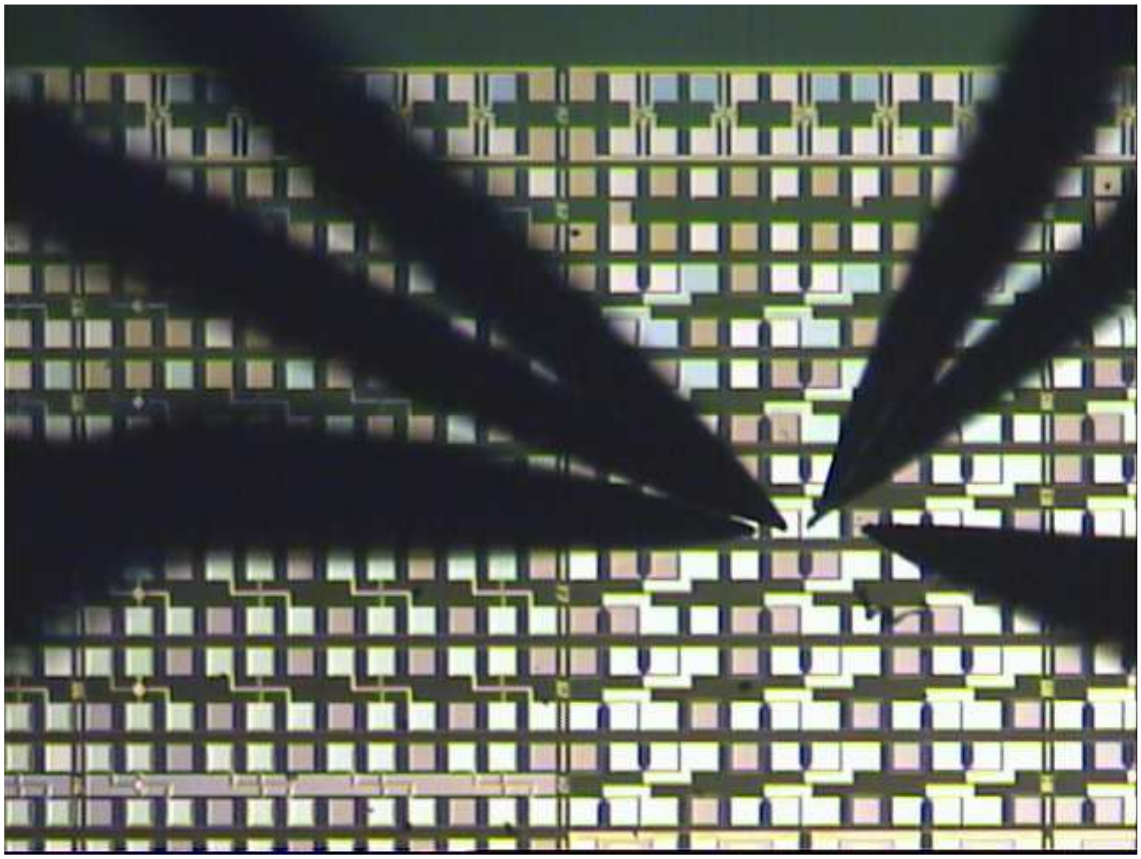


Figura III.22 – Oblea vista a través del microscopio

Para obtener la gráfica Id-Vd aplicamos una rampa de tensión en el drenador y medimos la intensidad que pasa por él para diferentes tensiones de puerta. En la gráfica, se puede apreciar a partir de qué tensiones de drenador el transistor comienza a saturase para cada tensión de puerta. Los parámetros introducidos en la interfaz así como la gráfica resultante se muestran a continuación en las figuras III.23 y III.24 respectivamente:

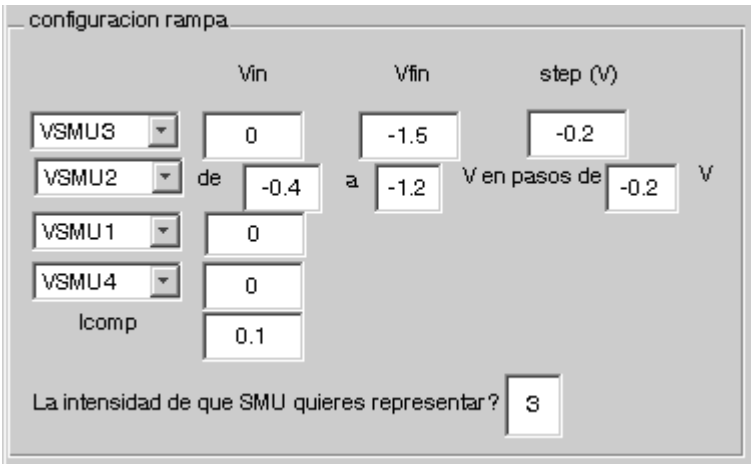


Figura III.23 – Parámetros introducidos en la interfaz

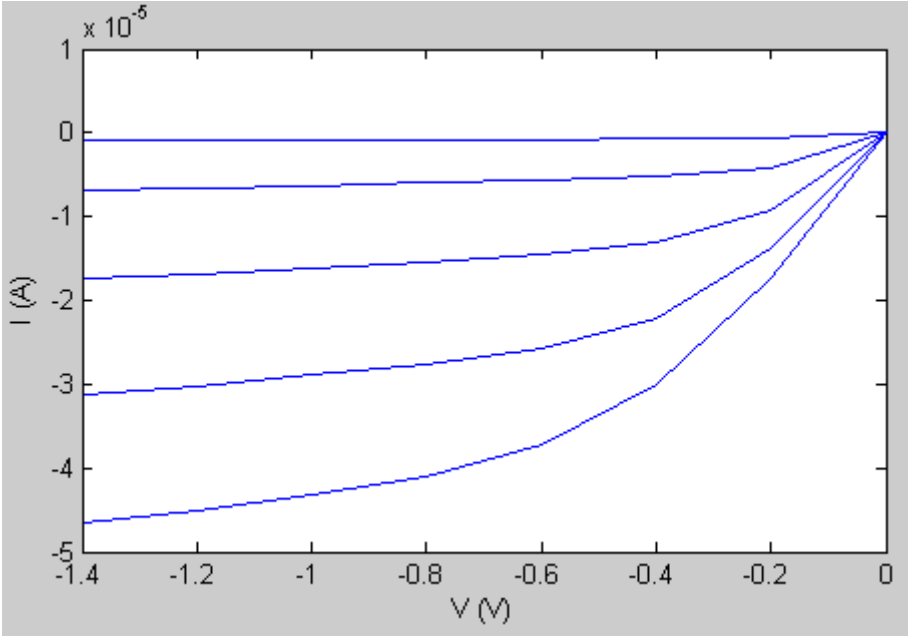


Figura III.24 – Gráfica Id-Vd resultante

Como se puede observar, al tratarse de un transistor PMOS, todas las tensiones introducidas y las intensidades leídas son negativas.

Para la obtención de la gráfica Id-Vg aplicamos una rampa de tensión en la puerta, una tensión constante en el drenador y medimos la corriente que pasa por él. La tensión de puerta que hace que en el drenador pase 1µA, es la tensión umbral del transistor. Los parámetros introducidos en la interfaz y la gráfica resultante se muestran a continuación en las figuras III.25 y III.26 respectivamente:

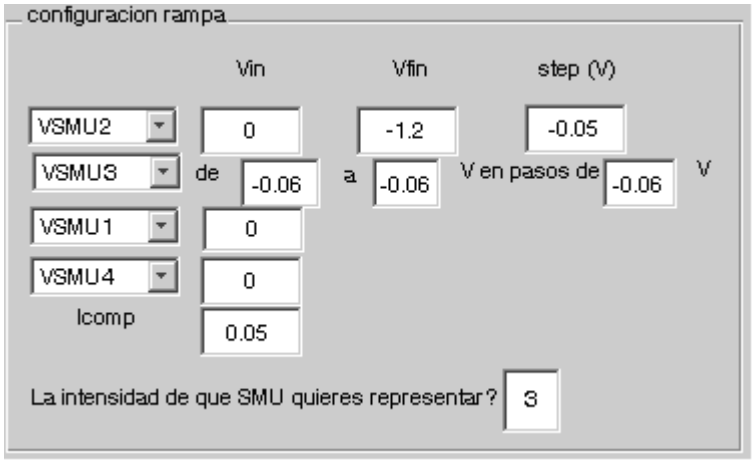


Figura III.25 – Parámetros introducidos en la interfaz

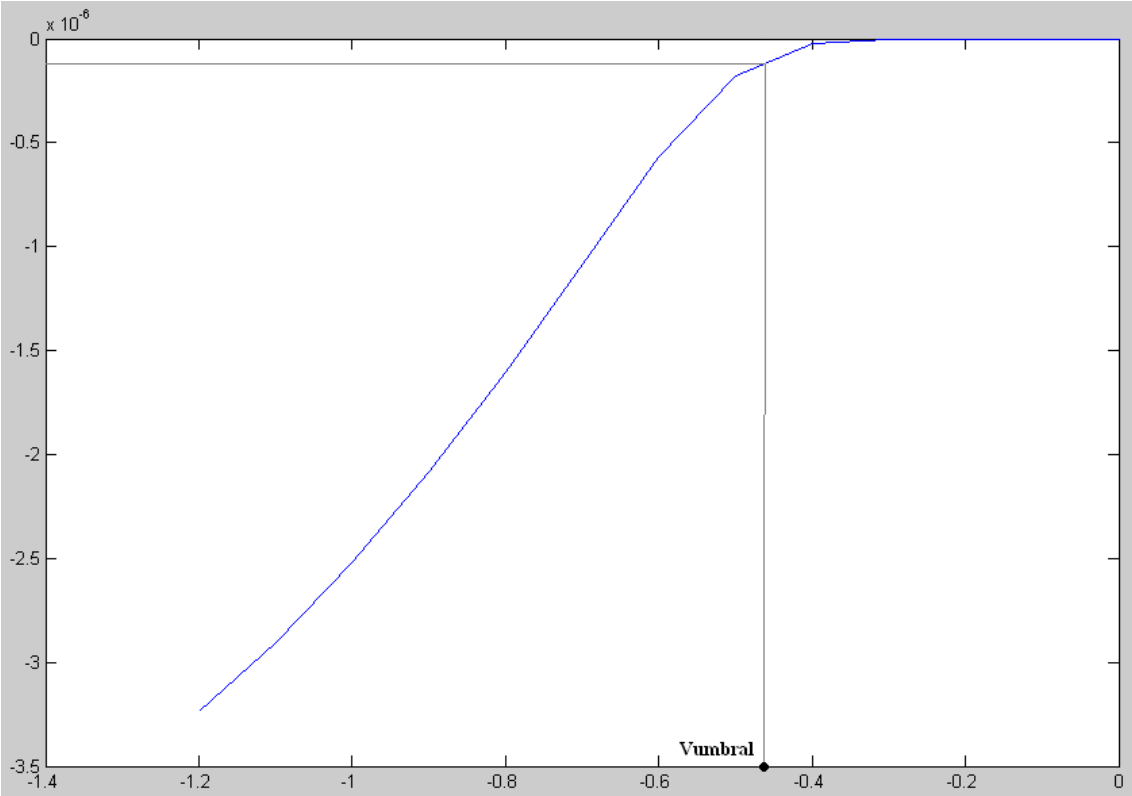


Figura III.26 – Gráfica Id-Vg resultante

5. CONCLUSIÓN

En este capítulo se ha descrito la segunda parte del proyecto que ha consistido en el control del analizador de semiconductores del laboratorio de investigación a través del bus GPIB. Como ejemplo de la aplicación, se ha desarrollado una interfaz gráfica que permite caracterizar transistores de dimensiones micrométricas.

La programación del bus GPIB se ha realizado mediante los comandos SCPI y los GPIB Command Reference. En este apartado, se ha presentado una explicación completa de la aplicación, analizando el procedimiento de control del instrumento, la interfaz gráfica y la estructura de programa.

CONCLUSIONES

Este proyecto ha estado dividido en dos partes bien diferenciadas pero con un objetivo final común: llevar a cabo el control de instrumentos mediante la programación del bus de instrumentación GPIB. Como ejemplo de este objetivo, hemos desarrollado para cada parte dos posibles aplicaciones.

En la primera, hemos desarrollado una aplicación que permite representar el diagrama de Bode de módulo y calcula la frecuencia de corte de un sistema. Todo ello realizado a través de una interfaz gráfica diseñada con el aplicativo GUIDE de Matlab. Los instrumentos sobre los cuales se ha llevado a cabo el control en esta parte han sido el osciloscopio y el generador de señales.

Para la segunda parte, la aplicación desarrollada permite realizar la caracterización de dispositivos o circuitos electrónicos a nivel de oblea mediante el control del analizador de semiconductores. Todo ello, de nuevo, será realizado a través de una interfaz gráfica.

La memoria del proyecto está dividida en tres capítulos. En el primero, se nombran los principales buses de instrumentación actuales, así como sus características más importantes; examinando con más detalle el bus GPIB, del que se analiza la evolución histórica y situación actual, las características eléctricas y funcionales y los diferentes métodos de programación. En el segundo capítulo se expone el desarrollo de la primera parte del proyecto, llevada a cabo en el laboratorio de docencia, mediante el análisis de los objetivos, la descripción de los equipos utilizados, el funcionamiento de la aplicación y un ejemplo completo de la misma. En el tercer capítulo, se expone el desarrollo de la segunda parte del proyecto, llevada a cabo en el laboratorio de investigación, analizando los objetivos, la descripción de los equipos utilizados, el funcionamiento de la aplicación y un ejemplo completo de la misma.

En cuanto a las dificultades encontradas, han sido muchas y muy variadas, ya que controlar equipos electrónicos a través de buses de instrumentación era un tema que me interesaba mucho antes de empezar el proyecto pero que desconocía por completo. Otro punto de dificultad importante ha sido la programación del bus GPIB y la creación de interfaces gráficas con Matlab, puesto que pese a que este software lo conocía a nivel usuario, no sabía que se podían llevar a cabo con él esas dos acciones. Es por tanto, que a nivel personal estoy muy satisfecho con el trabajo realizado durante estos seis meses al haber conseguido los objetivos marcados antes de iniciar el proyecto.

Como resultado final, hemos obtenido dos aplicaciones muy útiles aunque pienso que la gran ventaja de este proyecto es la inmensa versatilidad de las posibles aplicaciones a desarrollar pues opino que la gran dificultad reside en entender cómo se programa el bus GPIB pero que una vez asimilado, todas estas aplicaciones seguirán un patrón común. Cada una de ellas en sí, constituiría una posible ampliación de este proyecto.

BIBLIOGRAFÍA

Páginas Web consultadas:

[1] http://www.mathworks.com/

[2] http://www.home.agilent.com/agilent/home.jspx?cc=US&lc=eng

[3] http://www.tek.com/

Libros consultados:

[4] **"Instrumentación electrónica".**
Autor: Miguel A. Pérez García, Editorial: Thomson

[5] **"Sistemas de instrumentación".**
Autor: Pere J. Riu Costa, Editorial: Edicions UPC

[6] **"Conexión de instrumentos de medida con GPIB".**
Autor: Fernando Seco Granja

[7] **"Sistemas de medida y adquisición de datos".**
Autor: Ángel Garcimartín Montero

[8] **"Labview 7.1: Programación Gráfica para el Control de Instrumentación".**
Autor: Antoni Manuel Lázaro

[9] **"Aprenda MATLAB 7.0".**
Autor: Javier García de Jalón

[10] **"Interfaces Gráficas en Matlab usando GUIDE".**
Autor: Daniel de León Cárdenas

[11] **"Gráficas con MALTAB".**
Autor: Roberto Rodríguez del Río

Manuales consultados:

[12] **TDS200 Programmer Manual**

[13] **Agilent 33120A Programmer Manual**

[14] **Agilent 4155C Semiconductor Parameter Analyzer:**

SCPI Command Reference
GPIB Command Reference

ANEXO: Código de las funciones

Primera parte: Obtención del diagrama de Bode y la frecuencia de corte de un sistema

Inicia_inst:

```
function g = inicia_inst(x)
g=gpib('ni',0,x);
fopen(g);
fprintf(g, '*IDN?');
z=fscanf(g);
z
end
```

Genera_onda2:

```
function [c] = genera_onda2(inst,freq,amp,offset)
    c=sprintf('APPL:SIN %f HZ, %f VPP, %f V',freq,amp,offset);
    fprintf(inst,c)
end
```

VppCH:

```
function [Vpp] = VppCH(inst,x)
c=sprintf('measurement:meas%d:VALUE?',x);
fprintf(inst,c);
Vppchar=fscanf(inst,c);
Vppr=Vppchar(18:length(Vppchar));
Vpp=str2double(Vppr);
end
```

Fcorte:

```
function z =fcorte(vect_GdB,vect_freq)
bandera=0;
for i=1:length(vect_GdB)
    if(vect_GdB(i)<max(vect_GdB)-3)&&(bandera==0)
        x2=vect_freq(i);
        x1=vect_freq(i-1);
        y2=vect_GdB(i);
        y1=vect_GdB(i-1);
        v1=x2-x1;
        v2=y2-y1;

        fc=((((max(vect_GdB)-3)-y1)/v2)*v1)+x1;

        set(handles.fcorte, 'String',num2str(fc))
        bandera=1;
    end
end
z=fc;
```

Segunda parte: Caracterización de dispositivos electrónicos

Iniciayconfig_rampa2:

```
function c=iniciayconfig_rampa2(smus_rampa)

    %CONECTAMOS INSTRUMENTO
    c=inicia_ins_2(27)

    %MODO DE MESURA: RAMPA (SWEEP)
    fprintf(c,':PAGE:CHAN:MODE SWE')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    primero=smus_rampa(1);
    segundo=smus_rampa(2);
    tercero=smus_rampa(3);
    cuarto=smus_rampa(4);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %CONFIGURACIÓN SMU's
    conf_SMU(primero,'V','VAR1',c);
    conf_SMU(segundo,'V','CONS',c);
    conf_SMU(tercero,'V','CONS',c);
    conf_SMU(cuarto,'V','CONS',c);

    fprintf(c,':PAGE:CHAN:VSU1:DIS')
    fprintf(c,':PAGE:CHAN:VSU2:DIS')

    fprintf(c,':PAGE:CHAN:VMU1:DIS')
    fprintf(c,':PAGE:CHAN:VMU2:DIS')

    %CAMBIAMOS DE COMANDOS
    fprintf(c,'US')%seleccionamos gpib command reference
    fprintf(c,'CMD?')%preguntamos qué comandos estamos utilizando
    comandos=fscanf(c);%1=gpib command reference

    %ABRIMOS LOS 4 SMUS
    fprintf(c,'CN 1,2,3,4')%abrimos los SMU's que queramos

    %SELECCIONAMOS SPOT MEASUREMENT PARA LOS 4 SMUS
    fprintf(c,'MM 1,1,2,3,4');%spot measurement, smu1,2,3 y 4
```

Inicia_ins_2:

```
function c=inicia_ins_2(dir)

%INICIAMOS ANALIZADOR
c=gpib('ni',0,dir);
fopen(c)
fprintf(c,'*IDN?')
idn=fscanf(c);

%RESET ANALIZADOR
fprintf(c,'*RST')
```

Conf_SMU:

```
function conf_SMU(NSMU,mode,type,obj_gpib)

cadena=sprintf(':PAGE:CHAN:SMU%d:VNAME V%d',NSMU,NSMU);
fprintf(obj_gpib,cadena)

cadena=sprintf(':PAGE:CHAN:SMU%d:INAME I%d',NSMU,NSMU);
fprintf(obj_gpib,cadena)

cadena=sprintf(':PAGE:CHAN:SMU%d:MODE %s',NSMU,mode);
fprintf(obj_gpib,cadena)

cadena=sprintf(':PAGE:CHAN:SMU%d:FUNC %s',NSMU,type);
fprintf(obj_gpib,cadena)
```

RVS2:

```
function
[d,V1]=RVS2(Vin,step,Vfin,VSMU2_1,VSMU2_3,VSMU2_2,VSMU3,VSMU4,Icomp,sm
us_rampa,c,Ismu_representar)

primero=smus_rampa(1);
segundo=smus_rampa(2);
tercero=smus_rampa(3);
cuarto=smus_rampa(4);

for VSMU2=VSMU2_1:VSMU2_3:VSMU2_2
    cadena=sprintf('DV
%d,0,%f,%f,0',segundo,VSMU2,Icomp);%smu1,autorango,tensión,Icompliance
,autorango
    fprintf(c,cadena);
    cadena=sprintf('DV %d,0,%f,%f,0',tercero,VSMU3,Icomp);
    fprintf(c,cadena);
    cadena=sprintf('DV %d,0,%f,%f,0',cuarto,VSMU4,Icomp);
    fprintf(c,cadena);

    n=0;
    for i=Vin:step:Vfin
        n=n+1;

        cadena=sprintf('DV
%d,0,%f,%f,0',primero,i,Icomp);%smu1,autorango,tensión,Icompliance,aut
orango
        fprintf(c,cadena);
        fprintf(c,'XE');%lanza la medida

        fprintf(c,'TI? 1,0');%mide intensidad: smu1,autorango
        I1=fscanf(c);
        fprintf(c,'TI? 2,0');
        I2=fscanf(c);
        fprintf(c,'TI? 3,0');
        I3=fscanf(c);
        fprintf(c,'TI? 4,0');
        I4=fscanf(c);
        d(n,1)=str2double(I1(6:19));
```

```
        d(n,2)=str2double(I2(6:19));
        d(n,3)=str2double(I3(6:19));
        d(n,4)=str2double(I4(6:19));
        V1(n)=i;

    end
    VSMU2

    %PLOTEAMOS
    switch Ismu_representar
        case 1
            plot(V1,d(:,1))
        case 2
            plot(V1,d(:,2))
        case 3
            plot(V1,d(:,3))
        case 4
            plot(V1,d(:,4))
    end

%     plot(V1,d(:,2))
    hold on
    xlabel('V (V)')
    ylabel('I (A)')

end
```

Iniciayconfig_Vcte2:

```
function c=iniciayconfig_Vcte2

    %CONECTAMOS INSTRUMENTO
    c=inicia_ins_2(27)

    %MODO DE MESURA: SAMPLING
    fprintf(c,':PAGE:CHAN:MODE SAMP')

    %CONFIGURACIÓN SMU's
    conf_SMU(1,'V','CONS',c);
    conf_SMU(2,'V','CONS',c);
    conf_SMU(3,'V','CONS',c);
    conf_SMU(4,'V','CONS',c);

    fprintf(c,':PAGE:CHAN:VSU1:DIS')
    fprintf(c,':PAGE:CHAN:VSU2:DIS')

    fprintf(c,':PAGE:CHAN:VMU1:DIS')
    fprintf(c,':PAGE:CHAN:VMU2:DIS')

    %CAMBIAMOS DE COMANDOS
    fprintf(c,'US')
    fprintf(c,'CMD?')
    comandos=fscanf(c);

    %ABRIMOS LOS 4 SMUS
    fprintf(c,'CN 1,2,3,4')%abrimos los SMU's que queramos

    %SELECCIONAMOS SPOT MEASUREMENT PARA LOS 4 SMUS
```

```
    fprintf(c,'MM 1,1,2,3,4');%spot measurement, smu1,2,3 y 4
```

CVS2:

```
function [t,d]=CVS2(step,muestras,VSMU1,VSMU2,VSMU3,VSMU4,Icomp,c)

    cadena=sprintf('DV
1,0,%f,%f,0',VSMU1,Icomp);%smu1,autorango,tensión,Icompliance,autorang
o
    fprintf(c,cadena);

    cadena=sprintf('DV 2,0,%f,%f,0',VSMU2,Icomp);
    fprintf(c,cadena);

    cadena=sprintf('DV 3,0,%f,%f,0',VSMU3,Icomp);
    fprintf(c,cadena);

    cadena=sprintf('DV 4,0,%f,%f,0',VSMU4,Icomp);
    fprintf(c,cadena);

for i=1:muestras
    fprintf(c,'XE');%lanza la medida

    if(VSMU1~=0)
        fprintf(c,'TI? 1,0')%mide intensidad: smu1,autorango
        Ismu1=fscanf(c);
        d(i,1)=str2double(Ismu1(6:19))
    end

    if(VSMU2~=0)
        fprintf(c,'TI? 2,0')%mide intensidad: smu2,autorango
        Ismu2=fscanf(c);
        d(i,2)=str2double(Ismu2(6:19))
    end

    if(VSMU3~=0)
        fprintf(c,'TI? 3,0')%mide intensidad: smu3,autorango
        Ismu3=fscanf(c);
        d(i,3)=str2double(Ismu3(6:19))
    end

    if(VSMU4~=0)
        fprintf(c,'TI? 4,0')%mide intensidad: smu4,autorango
        Ismu4=fscanf(c);
        d(i,4)=str2double(Ismu4(6:19))
    end

    t(i)=i*step;
    pause(step);

end
```

Iniciayconfig_rampa3:

```
function e=iniciayconfig_rampa3(seq,c)
```

```
switch seq
    case 1
        smu_salida=1;
    case 2
        smu_salida=2;
    case 3
        smu_salida=3;
    case 4
        smu_salida=4;
end

    fprintf(c,':PAGE:CHAN:MODE SWE')

    conf_SMU(smu_salida,'V','VAR1',c);
    fprintf(c,':PAGE:CHAN:VSU1:DIS')
    fprintf(c,':PAGE:CHAN:VSU2:DIS')
    fprintf(c,':PAGE:CHAN:VMU1:DIS')
    fprintf(c,':PAGE:CHAN:VMU2:DIS')

    fprintf(c,'US')%seleccionamos gpib command reference
    fprintf(c,'CN 1,2,3,4')%abrimos los SMU's que queramos
    fprintf(c,'MM 1,1,2,3,4');%spot measurement, smu1,2,3 y 4
```

RVS3:

```
function [d1,V1]=RVS3(Vin,step,Vfin,num_smu,Icomp,c)

n=0;
    for i=Vin:step:Vfin
        n=n+1;

        cadena=sprintf('DV %d,0,%f,%f,0',num_smu,i,Icomp);

fprintf(c,cadena);%smu1,autorango,tensión,Icompliance,autorango

        fprintf(c,'XE');%lanza la medida

        fprintf(c,'TI? %d,0',num_smu)%mide intensidad: smu1,autorango
        I1=fscanf(c);
        d1(n)=str2double(I1(6:19));

        V1(n)=i;
    end

    %PLOTEAMOS
    hold on;
    %plot(V1,d1)
end
```

Iniciayconfig_Vcte3:

```
function e=iniciayconfig_Vcte3(seq,c)

switch seq
    case 1
        smu_salida=1;
    case 2
```

```
        smu_salida=2;
    case 3
        smu_salida=3;
    case 4
        smu_salida=4;
end

    fprintf(c,':PAGE:CHAN:MODE SAMP')

    conf_SMU(smu_salida,'V','CONS',c);
    fprintf(c,':PAGE:CHAN:VSU1:DIS')
    fprintf(c,':PAGE:CHAN:VSU2:DIS')
    fprintf(c,':PAGE:CHAN:VMU1:DIS')
    fprintf(c,':PAGE:CHAN:VMU2:DIS')

    fprintf(c,'US')
    fprintf(c,'CN 1,2,3,4')%abrimos los SMU's que queramos
    fprintf(c,'MM 1,1,2,3,4');%spot measurement, smu1,2,3 y 4
```

CVS3:

```
function [t,d]=CVS3(step,muestras,Vcte,num_smu,Icomp,c)

    cadena=sprintf('DV
%d,0,%f,%f,0',num_smu,Vcte,Icomp);%smu1,autorango,tensión,Icompliance,
autorango
    fprintf(c,cadena);

    for i=1:muestras
    fprintf(c,'XE');%lanza la medida

    fprintf(c,'TI? %d,0',num_smu)%mide intensidad: smu1,autorango
    Ismu=fscanf(c);
    d(i)=str2double(Ismu(6:19))

    t(i)=i*step;
    pause(step);
end
```