“PLATAFORMA DE SOFTWARE PARA LA INTEGRACIÓN DE PROTOCOLOS CAN Y LXI”

Karol Jimena Bernal Cortés

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE ELECTRÓNICA
BOGOTÁ, D.C.
2013

“PLATAFORMA DE SOFTWARE PARA LA INTEGRACIÓN DE PROTOCOLOS CAN Y LXI”

KAROL JIMENA BERNAL CORTÉS

Trabajo de grado para optar al título de Ingeniería Electrónica

Director
Carlos Eduardo Cotrino Badillo
Ingeniero Electrónico, M. Sc.

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE ELECTRÓNICA
BOGOTÁ, D.C.
2013

PONTIFICIA UNIVERSIDAD JAVERIANA

FACULTAD DE INGENIERÍA

CARRERA DE INGENIERÍA ELECTRÓNICA

RECTOR MAGNÍFICO: R. P. JOAQUÍN SANCHÉZ GARCÍA S. J.

DECANO ACADÉMICO:	**Ing. FRANCISCO REBOLLEDO MUÑOZ.**
DECANO DEL MEDIO UNIVERSITARIO:	**P. SERGIO BERNAL RESTREPO S. J.**
DIRECTOR DE CARRERA:	**Ing. JAIRO ALBERTO HURTADO, Ph. D.**
DIRECTOR DEL PROYECTO:	**Ing. CARLOS COTRINO BADILLO, M. Sc.**

NOTA DE ADVERTENCIA

"La universidad no se hace responsable de los conceptos emitidos por sus alumnos en sus proyectos de grado Sólo velará porque no se publique nada contrario al dogma y la moral católica y porque los trabajos no contengan ataques o polémicas puramente personales. Antes bien, que se vea en ellos el anhelo de buscar la verdad y la justicia".

Artículo 23 de la Resolución No. 13, del 6 de julio de 1946, por la cual se reglamenta lo concerniente a Tesis y Exámenes de Grado en la Pontificia Universidad Javeriana.

TABLA DE CONTENIDO

INTRODUCCIÓN 10

1. MARCO TEÓRICO 11

1.1. Estándar CAN 11

1.1.1. Formato de datos en CAN 15

1.1.2. Mensajes en el protocolo CAN 16

1.1.3. Arbitraje 16

1.1.4. Estructura de una trama de datos 17

1.1.5. Tipos de mensajes 18

1.1.6. Chequeo de errores y confinamiento de fallas 19

1.1.7. Módulo CAN del microcontrolador Texas TMS320F2808 22

1.2. Comunicación mediante el protocolo LXI 24

1.2.1. Protocolo LXI 24

1.2.2. Protocolo LXI en las fuentes Magna Power 26

2. ESPECIFICACIONES 28

3. DESARROLLOS 36

3.1. Configuración módulo CAN 38

3.1.1. Configuración del reloj 38

3.1.2. Parámetros de bit de configuración 39

3.2. Estructura del código de programación en archivo .C 39

3.3. USB-CAN Adapter 42

3.3.1. Propiedades y características 43

3.3.2. Conexión al Bus CAN 43

3.3.3. Cableado 44

3.4. Configuración LXI 44

3.5. Generación del perfil fotovoltaico en las fuentes Magna Power 45

3.6. Interoperabilidad entre los dos Protocolos de comunicación 45

4. PROCEDIMIENTO 47

4.1 Implementación y configuración módulo CAN 47

4.2 Implementación y configuración CAN-USB Adapter 47

4.2.1 Instalación y Configuración Software PCAN-View 47

4.2.2 Recepción y Transmisión 48
4.2.3 Exportación de datos en archivo XMT 50
4.3 Implemetación y configuración de las fuentes Magna Power en una red LAN privada 53
4.3.1 Configuración fuentes Magna Power (instrumento LXI) 53
4.3.2 Configuración Router Allied Telesis AR410 55
4.3.3 Configuración PC 56
4.4 Implementación y configuración perfil fotovoltaico PPPE 58
5. ANALISIS DE RESULTADOS 62
5.1 Comunicación del módulo CAN 62
5.2 Comunicación de las Fuentes Magna Power 64
5.3 Interoperabilidad de los dispositivos con el PC maestro 64
6. CONCLUSIONES 66
7. BIBLIOGRAFÍA 69
8. ANEXOS 72

ÍNDICE DE FIGURAS

Figura 1. Arquitectura estándar de ISO 11898 [4]....12
Figura 2. Capa Física CAN [3]....12
Figura 3. Diagrama de bloques módulo CAN y circuito transceiver [7]....13
Figura 4. 3.3 V eCAN transceivers [7]....14
Figura 5. Conectores DB9 [3]....14
Figura 6. Lógica de inversión en el Bus CAN [4]....16
Figura 7. Arbitraje en el bus CAN [4]....17
Figura 8. Formato de Trama CAN2.0B [7]....18
Figura 9. Trama de Error [9]....20
Figura 10. Evolución entre estados de error [11]....22
Figura 11. Estructura interna módulo CAN [3]....24
Figura 12. Clases de instrumentos LXI [18]....25
Figura 13. Sistema LXI que incluye dispositivos LXI e instrumentos no LXI [13]....26
Figura 14. Magna Power Electronics, XR series [22]....26
Figura 15. Características del cable [25]....27
Figura 16. JS5, Ethernet (visto desde terminación hembra) [20]....27
Figura 17. Diagrama de bloques....30
Figura 18. Plataforma de software para la integración de protocolos CAN y LXI....30
Figura 19. Imagen real tarjeta de desarrollo TMS320F2808....36
Figura 20. Detalles de un Bus CAN [4]....37
Figura 21. Bus CAN [8]....38
Figura 22. CAN Bit timing [7]....39
Figura 23. Registros PIE [3]....41
Figura 24. PCAN-USB adapter [29]....43
Figura 25. Asignación de Pines CAN (Visto desde el conector del PCAN-USB Adapter) [29]....43
Figura 26. Conexión Bus CAN con el PCAN-USB adapter....44
Figura 27. Conexión instrumentos LXI en una red LAN privada [32]....45
Figura 28. Software PCAN-Explorer....46
Figura 29. Selección de parámetros CAN específicos (hardware) [29]....47
Figura 30. Ventana de transmisión de un nuevo mensaje [29]....48
Figura 31. Imagen Recepción y transmisión de mensajes Bus CAN....49
Figura 32. Imagen pestaña Trace Software PCAN-View....50
Figura 33. Imagen Selección de hardware....51
Figura 34. Imagen insertar mensaje....51
Figura 35.Imagen guardar archivo XMT....52
Figura 36. Imagen selección carpeta destino....52
Figura 37. Imagen software Eureka....53
Figura 38. Imagen panel trasero Fuente Magna Power....54
Figura 39. Panel de Información del instrumento LXI....54
Figura 40. Imagen panel de Configuración del instrumento....55

Figura 41. Imagen Conexión del router 56
Figura 42. Imagen propiedades de protocolo internet TCP/IP 57
Figura 43. Imagen software Eureka 57
Figura 44.Imagen Menú Photovoltaic power profile emulation 58
Figura 45.Imagen Listado de Fuentes en la red LAN 59
Figura 46. Imagen del software Photovoltaic power profile emulation 60
Figura 47. Imagen options software Photovoltaic power profile emulation 61
Figura 48. Selección de parámetros CAN específicos (hardware) [33] 63

ÍNDICE DE TABLAS

Tabla 1. Formato de datos en CAN [3] 15
Tabla 2. Comunicación Full – CAN [3] 16
Tabla 3. Trama de datos de CAN [3]..... 18
Tabla 4.Características de tramas de error..... 20
Tabla 5Condiciones para evolución entre estados TEC [11] 21
Tabla 6.Condiciones para evolución entre estados REC..... 21
Tabla 7. Estructura del interna módulo CAN en la tarjeta TMS320F2808..... 23
Tabla 8. Tabla de especificaciones de LXI..... 29
Tabla 9. Estructura del código de programación en archivo .C 42

INTRODUCCIÓN

El trabajo de grado titulado *"plataforma de software para la integración de protocolos CAN y LXI"* hace parte del proyecto de investigación en curso denominado "Conversión de múltiples puertos, interconexión, control y asignación dinámica de energía de generadores PV distribuidos" del grupo CEPIT, financiado por la Pontificia Universidad Javeriana.

El presente trabajo contiene conceptos tales como el control y la comunicación industrial aplicados a una plataforma de software mediante la cual se integran 3 fuentes Magna Power [1] que estarán configuradas bajo el estándar LXI (LAN Extensions for Instrumentations), de manera que el usuario pueda acceder remotamente a través de un servidor web a los sistemas de prueba que ofrecen las fuentes y 5 controladores Texas (TMS320F28335) [2] que permiten el intercambio de datos que suministra el mismo usuario o las fuentes Magna.

Así mismo, es preciso mencionar que el presente trabajo de grado no sólo colaborará con el proyecto de investigación aludido, sino que, adicionalmente, apoyará las actividades académicas propias de los estudiantes de maestría como soporte de futuros trabajos de grado, de la misma manera, constituye un recurso de práctica en diferentes asignaturas (comunicaciones y controles) y aporte para otros proyectos de investigación.

El objetivo general, entonces, es diseñar e implementar una plataforma de software que integre un protocolo de comunicaciones (CAN) y una plataforma de instrumentación (LXI). Para esto, se formularon y desarrollaron los siguientes objetivos específicos: i) identificar y fijar los parámetros necesarios (capa física, capa de datos, topología, entre otros) para el funcionamiento del sistema de control en el protocolo de comunicaciones CAN; ii) configurar los cinco controladores TEXAS (TMS320F28335 eZdsp) de tal forma que permitan la gestión automática del sistema de distribución de energía fotovoltaica (Nota: En la actualidad sólo hay un controlador TMS320F2808 en buen estado); iii) establecer las pautas y requisitos necesarios para la implementación del protocolo de instrumentación LXI; iv) configurar las fuentes de alimentación DC, para que generen el perfil de la energía fotovoltaica y, v) implementar y probar la interoperabilidad de la plataforma de software basada en los protocolos de comunicaciones e instrumentación.

El presente documento se estructura de la siguiente forma: *i)* Un capítulo teórico en el que se describen los fundamentos necesarios para la comprensión del trabajo realizado, *ii)* luego, se exponen las especificaciones alcanzadas en el presente trabajo de grado, *iii)* a continuación, se describen cada una de las partes que conforman el trabajo, explicando la tecnología utilizada y los diseños realizados, *iv)* finalmente, se exponen los resultados obtenidos y las conclusiones como aportes al conocimiento dentro de las áreas de comunicaciones y de controles.

1. MARCO TEÓRICO

1.1. Estándar CAN

CAN (*Controller Area Network*) fue introducido y patentado por Robert Bosch GmbH, Alemania, para uso en la industria automotriz aunque hoy en día también es utilizado como bus industrial.

CAN es una red de comunicación serial que transmite a través de dos líneas diferenciales, no utiliza direcciones para dirigirse a estaciones, es decir, que cada mensaje es enviado con un identificador el cual es reconocido por diferentes nodos. El identificador determina si al transmitirse un mensaje puede ser recibido por el módulo CAN, además establece la prioridad del mensaje cuando dos o más nodos quieren transmitir al mismo tiempo [3]. Las principales características principales del estándar CAN son:

- Utiliza transmisión serial y sincrónica de datos.
- Acceso aleatorio con prevención de colisión (*Random Access with collision avoidance*)
- La longitud de los mensajes es corta, máximo 8 bytes por mensaje.
- Velocidad de transmisión de datos de 100 KBPS hasta 1MBPS.
- La longitud del bus es corta, dependiendo de la velocidad de datos.

El protocolo de comunicaciones CAN está basado en la norma europea ISO-11898 que describe como la información es transferida entre dispositivos de una red conformada por el modelo de sistema abierto de interconexión conocido como OSI, el cual está definido en capas. Actualmente, la comunicación entre dispositivos está definida a través de la capa física del modelo. La arquitectura de ISO 11898 define las dos primeras capas de las 7 capas del modelo OSI como la capa de enlace de datos y la capa física como lo muestra la figura 1. [4]

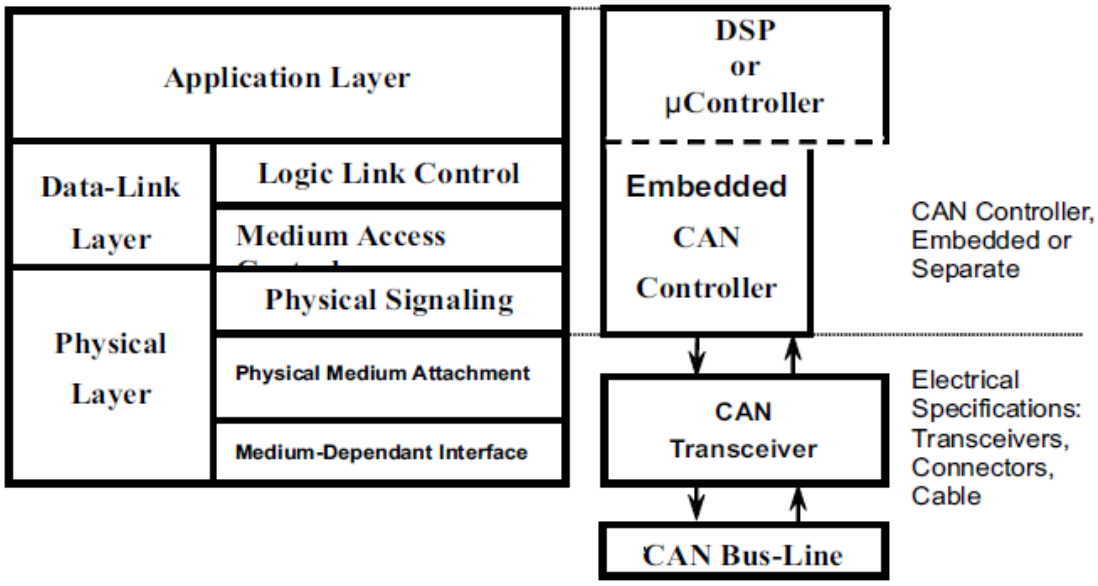


Figura 1. Arquitectura estándar de ISO 11898 **[4]**

La capa física permite especificar el medio por el cual se realizará la transmisión de datos entre los nodos de una red, y sus principales características son los niveles de señal, representación, sincronización y tiempos en que los bits se tranfieren por el bus [5]. Dicha capa se subdivide en tres subcapas [6]:

- Señalización Medio físico: Codificación del bit, temporización y sincronización.
- Unión del medio físico: Características de recepción
- Interfaz dependiente del medio: Conector del bus

El medio Físico consiste en un par de cables trenzados (CAN_H y CAN_L) los cuales fluyen en ambos sentidos; los extremos de la red deben estar terminados mediante una resistencia de 120Ω (como lo muestra la figura 2), de lo contrario cada trama transferida causaría una reflexión que puede originar fallos de comunicación. El uso de los voltajes diferenciales permite que las redes CAN funcionen cuando una de las líneas de señales es desconectada. En la especificación básica de CAN la velocidad de transmisión es de 1 Mbps [5].

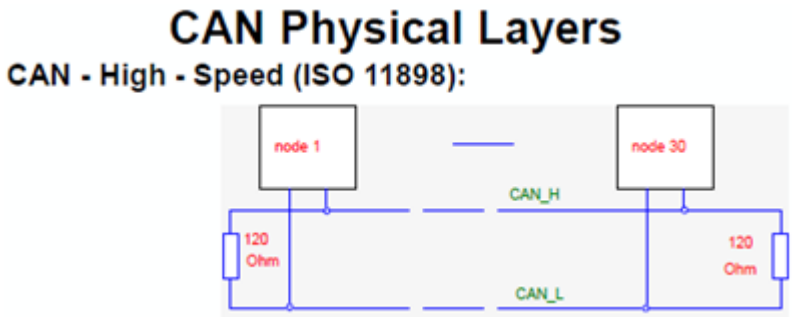


Figura 2. Capa Física CAN **[3]**

Debido al principio de transmisión de CAN como "*broadcast*", todos los módulos CAN están forzados a "escuchar" al bus todo el tiempo. Es muy probable que uno de estos pueda perderse en el proceso de arbitraje, por esta razón se hace necesario que este módulo en particular cambie a modo de recepción de datos.

Para esto se requiere un circuito *transceiver* que provea permanente el voltaje del bus al módulo CAN. En este caso el controlador F2808 cuenta internamente con este *transceiver* entre las señales digitales del controlador TMS320F2808 y las líneas del bus para ajustar los voltajes físicos como lo muestra la imagen.

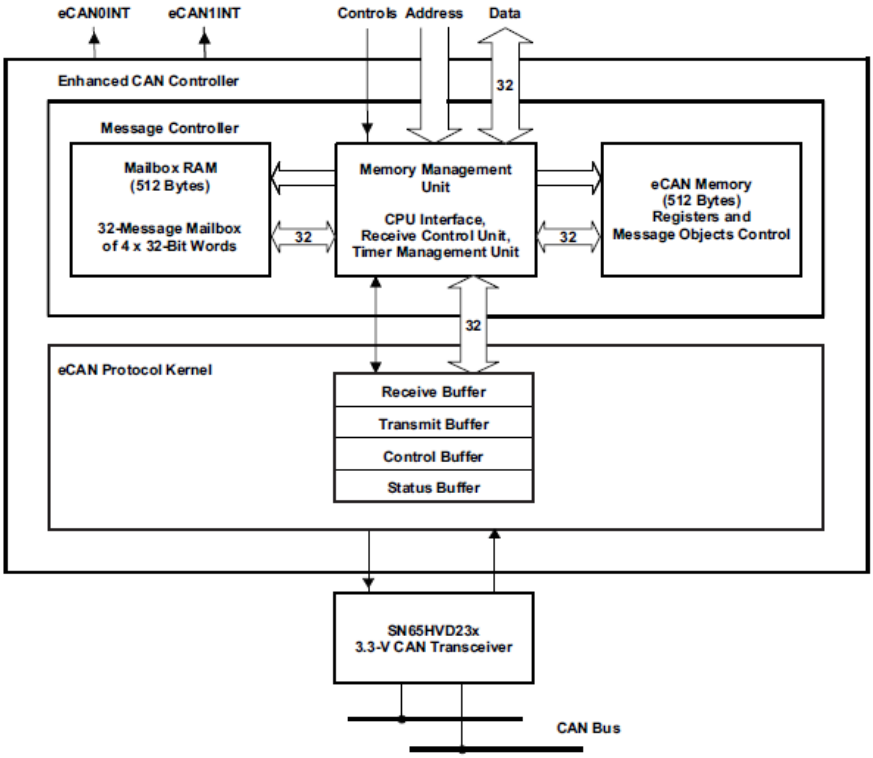


Figura 3. Diagrama de bloques módulo CAN y circuito transceiver **[7]**

De acuerdo con el diagrama de bloques anterior, la capa física del protocolo CAN requiere un circuito transceiver entre las señales digitales del F2808 y las líneas del bus (GPIO30 (CAN- RX) y GPIO31 (CAN - TX)) para ajustar los voltajes físicos. La Figura 4, muestra los circuitos transceivers disponibles para la tarjeta TMS320F2808.

PART NUMBER	SUPPLY VOLTAGE	LOW-POWER MODE	SLOPE CONTROL	VREF	OTHER	T_A
SN65HVD230	3.3 V	Standby	Adjustable	Yes	–	–40°C to 85°C
SN65HVD230Q	3.3 V	Standby	Adjustable	Yes	–	–40°C to 125°C
SN65HVD231	3.3 V	Sleep	Adjustable	Yes	–	–40°C to 85°C
SN65HVD231Q	3.3 V	Sleep	Adjustable	Yes	–	–40°C to 125°C
SN65HVD232	3.3 V	None	None	None	–	–40°C to 85°C
SN65HVD232Q	3.3 V	None	None	None	–	–40°C to 125°C
SN65HVD233	3.3 V	Standby	Adjustable	None	Diagnostic Loopback	–40°C to 125°C
SN65HVD234	3.3 V	Standby and Sleep	Adjustable	None	–	–40°C to 125°C
SN65HVD235	3.3 V	Standby	Adjustable	None	Autobaud Loopback	–40°C to 125°C

Figura 4. 3.3 V eCAN transceivers **[7]**

El transceiver SN65HVD230 está diseñado para el uso en aplicaciones que emplean en su capa física, comunicación CAN serial de acuerdo con el estándar ISO 11898. Dicho transceiver está diseñado para que sea capaz de transmitir diferencialmente al bus y recibir diferencialmente desde el controlador CAN a velocidades mayores a 1 Mbps. Provee tres modos diferentes de operación: *i)* alta velocidad (≥ 1Mbps): se selecciona colocando el pin 8 del integrado a tierra, permitiendo que la salida del transmisor cambie de encendido a apagado lo más rápido posible sin limitación en las pendientes de subida y bajada. *ii)* control de pendiente: puede ser ajustado conectando una resistencia (valores de 10KΩ a 100KΩ) a tierra en el pin 8 *iii*) modo de potencia baja: cuando el transceiver cambia a modo de baja corriente, el driver se apaga y únicamente el pin 8 permanece activo a un nivel lógico alto [8].
El conector CAN se observa en la figura 5 y su distribución de pines es la siguiente: , CANL (pin 2), CANH (pin 7) y CAN_GND (pin3)

Pin Nr.	Signal	Description
1	-	Reserved
2	CAN_L	CAN Bus Signal (dominant low)
3	CAN_GND	CAN ground
4	-	Reserved
5	CAN_SHLD	Optional shield
6	GND	Optional CAN ground
7	CAN_H	CAN Bus Signal (dominant high)
8	-	Reserved
9	CAN_V+	Optional external voltage supply Vcc

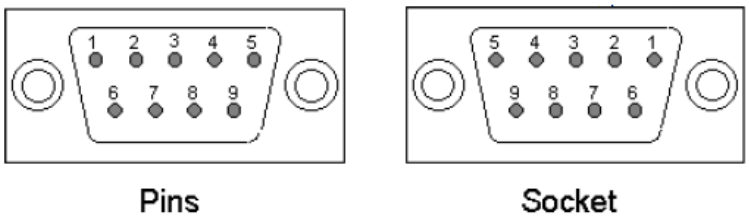


Figura 5. Conectores DB9 **[3]**

El protocolo de comunicación CAN es de acceso múltiple con detección de colisión y arbitraje en la prioridad del mensaje (CSMA/CD+AMP). CSMA significa que cada nodo en el bus debe esperar por un periodo predeterminado de inactividad antes de intentar enviar un mensaje.

CD+AMP indica que las colisiones se resuelven a través de una operación de arbitraje, basado en una prioridad pre-programada para cada mensaje en el campo identificador del mensaje [4]. El mensaje con la prioridad más alta continúa su transmisión sin tener en cuenta la colisión con otros mensajes, esta se denomina evasión de colisiones (*collision avoidance*) [3].

1.1.1. Formato de datos en CAN

El estándar ISO- 11898 describe dos tipos de formatos de mensajes estándar y extendido, la siguiente tabla enuncia las características principales de cada formato.

Formato de datos en CAN [3]	
Estándar-CAN	• CAN –Versión 2.0A • Mensajes con 11 bits de identificador
Extendido – CAN	• CAN –Versión 2.0B • Mensajes con 29 bits de identificador

Tabla 1. Formato de datos en CAN **[3]**
Fuente: Elaboración propia

En el presente proyecto, se advierte que la versión utilizada del protocolo CAN en el controlador es la 2.0B, en el cual se describen tanto los formatos de mensajes estándar como extendidos, además, tiene la ventaja de configurarse como transmisor o receptor y soporta datos o tramas remotas [9]. En el formato CAN 2.0B se destacan como características generales, las siguientes: *i)* Está compuesto de 0 a 8 bits de datos, *ii)* usa un tiempo de 32-bits para recibir y transmitir el mensaje, *iii)* sostiene la prioridad dinámicamente programable de la transmisión del mensaje, *iv)* emplea una alarma programable cuando existe una desconexión por tiempo en la transmisión o recepción, *v)* responde automáticamente a un mensaje de petición remoto, *vi)* retransmite automáticamente una trama de datos en caso de pérdida de las señales de control de arbitraje que determinan que dispositivo puede comenzar una transacción y, finalmente, *vii)* alcanza velocidades de datos mayores a 1 Mbps [9].

La implementación en hardware del protocolo CAN viene predeterminado por el módulo CAN en el controlador F28xx, se utiliza entonces FULL-CAN.

COMUNICACIÓN FULL – CAN [3]	
FULL – CAN	• Provee mensajes al servidor • Utiliza *mailboxes* configurables • Reconocimiento avanzado de errores

Tabla 2. Comunicación Full – CAN [3]
Fuente: Elaboración propia

1.1.2. Mensajes en el protocolo CAN

Una característica fundamental de CAN (figura 6) es el estado lógico opuesto entre el bus, la entrada driver y la salida del receptor. Normalmente, un estado lógico alto es asociado con uno (1) y un estado lógico bajo está asociado con cero (0), pero esto no ocurre en el bus CAN, es por esto que, los TI CAN *transceivers* poseen los pines de entrada driver y la salida del receptor, entonces, en ausencia de alguna entrada el dispositivo automáticamente (por defecto) coloca un estado recesivo del bus en todos los *pins* de entrada y salida. [4]

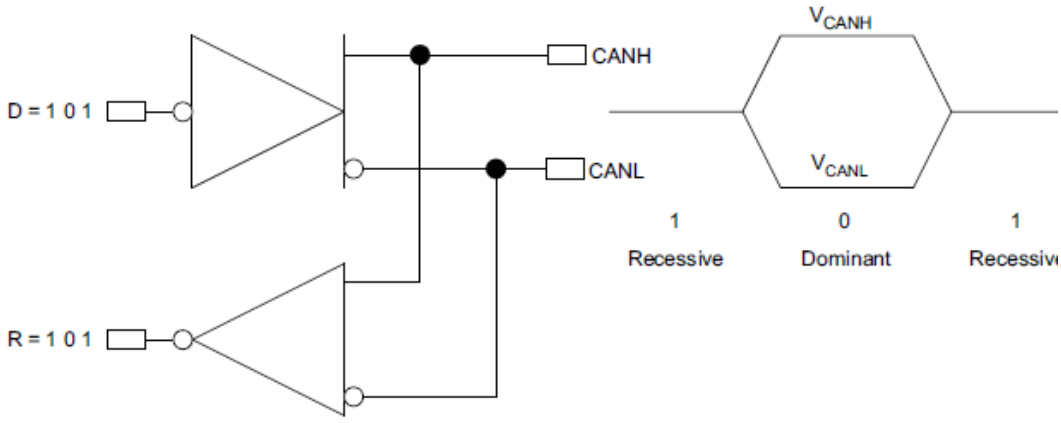


Figura 6. Lógica de inversión en el Bus CAN **[4]**

1.1.3. Arbitraje

El acceso al bus es *event-driven* (la transmisión de un mensaje es causada por la aparición de un evento específico definido en el perfil del dispositivo) y tiene lugar aleatoriamente. Si dos nodos tratan de ocupar el bus simultáneamente, el acceso se implementa con una operación de arbitraje no destructivo. No destructivo indica que el nodo de arbitraje con prioridad continúa con el mensaje sin que el mensaje sea destruido por otro nodo.

La asignación de prioridad de los mensajes en el identificador es la característica de CAN que lo hace particularmente atractivo e interesante en un ambiente de control en tiempo real. Un

identificador consiste en varios ceros y es la prioridad más alta del mensaje en una red debido a que este soporta el bus dominante. Finalmente, cuando dos nodos quieren transmitir simultáneamente uno de estos nodos envía el último bit identificador como cero (dominante) mientras que el otro nodo envía un uno (recesivo) manteniendo el control en el bus CAN procediendo a completar el mensaje [4].

La figura 7 muestra el proceso de arbitraje en CAN manejado automáticamente por un controlador. Cada nodo monitorea continuamente su propia transmisión, como el nodo B es un bit recesivo este es sobrescrito por el nodo C que posee el bit dominante de mayor prioridad, B detecta que el estado del bus no corresponde al bit que es transmitido, por lo tanto, el nodo B detiene la transmisión mientras el nodo C continúa con su mensaje. El nodo B intenta nuevamente transmitir el mensaje una vez que el bus es liberado por el nodo C.

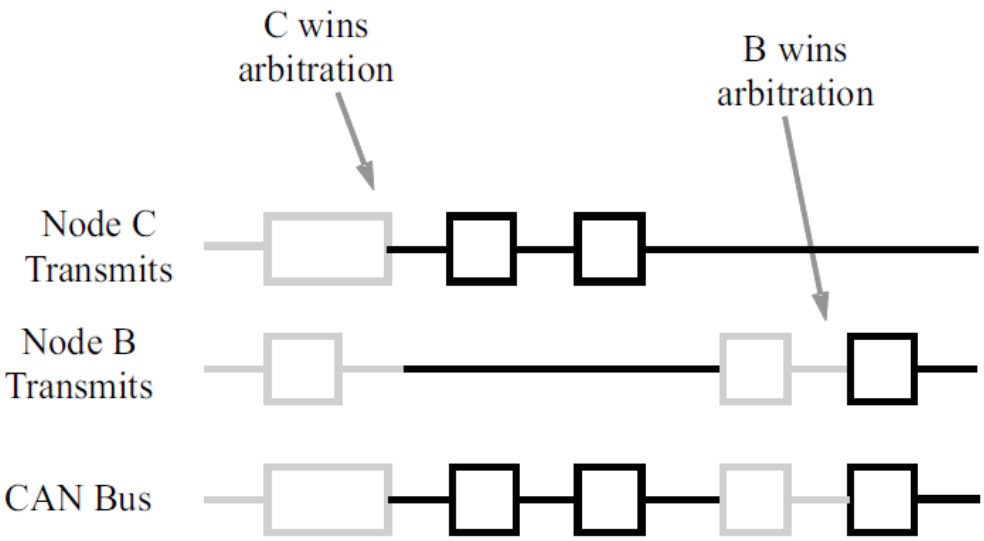


Figura 7. Arbitraje en el bus CAN **[4]**

1.1.4. Estructura de una trama de datos

El protocolo de comunicaciones CAN como se enunció anteriormente, se encuentra basado en mensajes, no en direcciones. El nodo emisor transmite el mensaje a todos los nodos de la red sin especificar un destino, todos escuchan el mensaje para luego filtrarlo según les interese. A continuación se especifican los campos que conforman la trama de datos:

TRAMA DE DATOS DE CAN [3]	
1. Campo de arbitraje	• Denota la prioridad del mensaje • Dirección lógica del mensaje (identificador) • Denota el tipo de mensaje, tramas extendidas o estándar
2. Campo de datos	• Más de 8 bits por mensaje • También es permitido 0 bits por mensaje
3. Campo CRC (comprobación de redundancia cíclica)	• Contiene *checksum*: comprobación que se hace para ver que una transferencia no haya provocado errores.
4. Campo de finalización	• Contiene la señal de reconocimiento (*acknowledgement*) y la finalización del mensaje.

Tabla 3. Trama de datos de CAN [3]

Fuente: Elaboración Propia

Varios tipos de tramas están predefinidos por el protocolo CAN para la gestión de transferencia de mensajes a continuación se observa la estructura de trama de protocolo CAN:

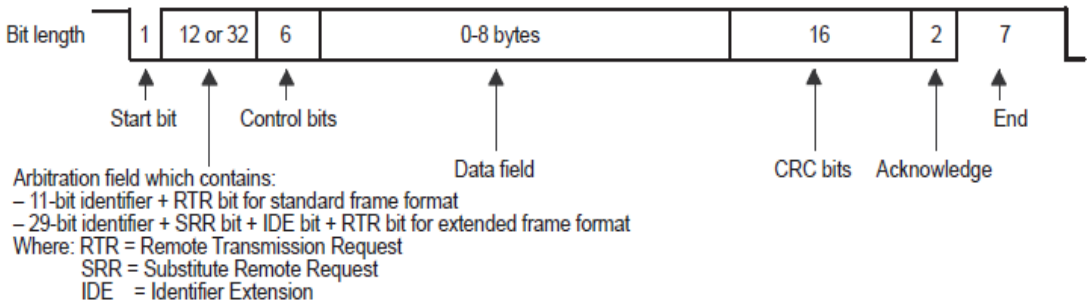


Figura 8. Formato de Trama CAN2.0B **[7]**

1.1.5. Tipos de mensajes

El protocolo CAN soporta 4 diferentes tipos de tramas para comunicación tales como:

- **Trama de datos:** es el tipo de mensajes más común, compuesto por el campo de arbitraje, campo de datos, el campo de comprobación (CRC) y el campo de reconocimiento. El campo de arbitraje contiene 11 o 29 bits según el formato escogido, el campo de datos contiene de cero a 8 bits de datos, el campo de CRC contiene 16 bits de comprobación usado para la detección de errores y Finalmente, el último es el campo de reconocimiento. Es aquella que transporta datos desde un nodo transmisor hasta un nodo receptor [7].

- **Trama remota:** el objetivo de dicha trama es solicitar la transmisión de datos de uno a otro nodo. La trama remota es similar a la trama de datos, sin embargo, las dos diferencias principales son: *i)* el bit RTR es recesivo en el campo de arbitraje, *ii)* no hay datos.

- **Trama de error:** la trama de error es un mensaje especial que viola las reglas de formateo de los mensajes de CAN. Este es transmitido cuando un nodo detecta un mensaje de error y causa que los otros nodos en la red también envíen una trama de error, el transmisor original automáticamente retransmite el mensaje. El controlador CAN es un sistema elaborado de contador de errores el cual asegura que el nodo no pueda estar vinculado a un bus transmitiendo repetidamente tramas de error [4].

- **La trama overload:** es similar a la trama de error en formato y es transmitida por un nodo que se encuentra ocupado. Principalmente es usado para proveer un retraso extra entre anterior y la siguiente la trama de datos o trama remota [10].

1.1.6. Chequeo de errores y confinamiento de fallas

El protocolo CAN emplea métodos para detectar errores en las tramas. Esta detección de error es utilizada para retransmitir la trama hasta que sea recibida con éxito. Si un mensaje falla y uno de estos métodos de error no es aceptado, un trama de error se genera desde el nodo receptor, obligando al nodo transmisor a enviar nuevamente el mensaje hasta que este es recibido correctamente, sin embargo, si un nodo defectuoso se engancha continuamente al bus repitiendo un error, este transmisor tiene la capacidad de ser removido por el controlador después de que un error límite es alcanzado [4].

La capa 2 del protocolo CAN incluye una estrategia de mejora para detectar transmisión de errores. Se destacan como características generales, las siguientes: *i)* Una trama de error se transmite tan pronto como el error ha sido detectado, *ii)* está basado en dos campos diferentes: *Error Flag Field, Error delimitied field* [3] como lo muestra la figura 9.

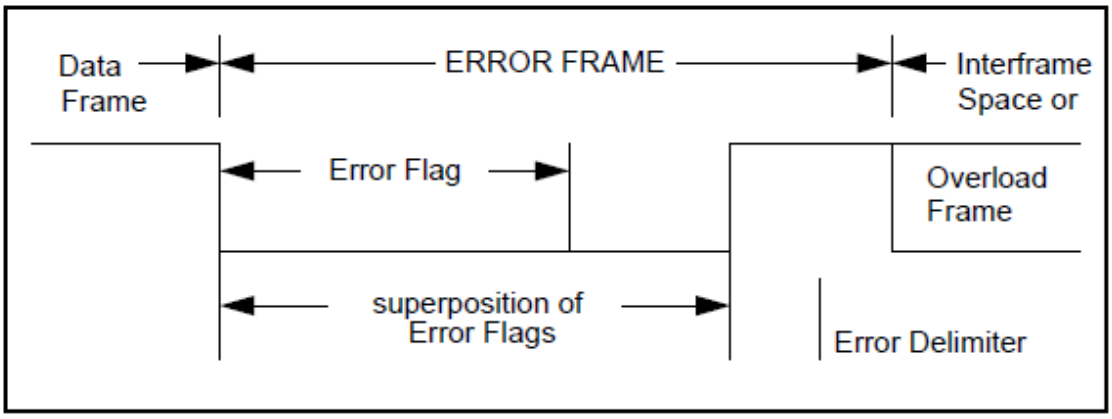


Figura 9. Trama de Error **[9]**

A continuación se especifican las características de los campos que posee una trama de error:

***Error Flag Field* [3]**	***Error delimitied field***
Depende del estado de error en el nodo: • **Error Activo: 6 bits dominantes consecutivos de error; lo otros nodos responderán a esta violación con sus propias tramas de error.** • **Error Pasivo: 6 bits recesivos consecutivos más 8 bits de error = 14 bits recesivos.** **Recesivo: no corrompe el mensaje** **Transmisor: otros nodos pueden responder con tramas de error activo.**	• 8 bits recesivos. • Permite a los nodos del bus restaurar la comunicación después de un error.

Tabla 4.Características de tramas de error.
Fuente: Elaboración propia

Para evitar que un nodo en problemas condicione el funcionamiento del resto del bus, el protocolo de comunicaciones CAN incorpora medidas de aislamiento de nodos defectuosos. Un nodo puede encontrarse en uno de los 3 estados siguientes:

- Estado de error activo: estado normal de un nodo, participa en la comunicación y en caso de detección de error envía una trama de error activa.

- Estado de error pasivo: Participa también en la comunicación, sin embargo, debe esperar una secuencia adicional de bits recesivos antes de transmitir y sólo puede señalar errores con una trama de error pasiva.

- Estado de Bus apagado: No participa en la comunicación en este estado.

La evolución entre estos estados se basa en dos contadores incluidos en el controlador de comunicaciones: contador de errores de transmisión (TEC) y contador de errores de recepción (REC), cada uno de estos se actualizan de acuerdo a las siguientes reglas:

CONDICIÓN	TEC
Si un nodo transmisor detecta un error de bit al enviar un mensaje activo de error	Incrementa 8
Si es cualquier otro tipo de error (relleno, trama, CRC, ACK)	Incrementa 8
Ante una transmisión de trama válida	Decrementa en 1 si > 0

Tabla 5Condiciones para evolución entre estados TEC **[11]**

CONDICION	REC
Si un nodo receptor detecta un error de bit al enviar un mensaje activo de error	Incrementa 8
Si un nodo receptor detecta un bit dominantecomo primer bit tras el envío de un indicador de error	Incrementa 8
Si es cualquier otro tipo de error (relleno, trama, CRC, ACK)	Incrementa 1
Ante una recepción de trama válida	Decrementa en 1 si>0 y <=127

Tabla 6.Condiciones para evolución entre estados REC

Un nodo pasa de estado activo a pasivo cuando su TEC supera 127 o cuando el REC supera 127. Cuando un nodopasa de activo a pasivo envía una trama de error activo.

Un nodo pasa de pasivo a anulado cuando TEC supera 255.

Un nodo pasa de pasivo a activo cuando el TEC como el REC bajan de 128

Un nodo puede pasar de anulado a activo (reiniciando TEC y REC) tras 128 secuencias de 11 bits recesivos sucesivos.

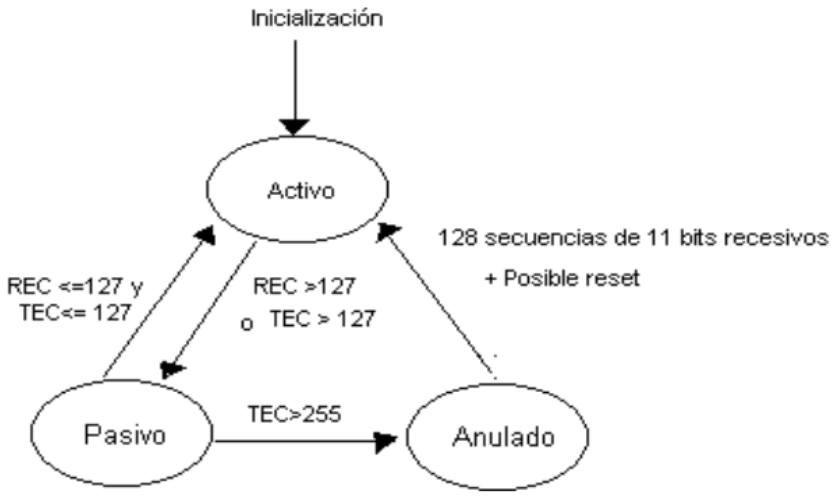


Figura 10. Evolución entre estados de error **[11]**

Estos tres niveles tienen la habilidad de identificar fallas potenciales en el nodo así como también aislarlos de tal manera que se mantenga una parte del bus funcionando.

1.1.7. Módulo CAN del microcontrolador Texas TMS320F2808

Los controladores Texas (TMS320F2808 eZdsp) integran un módulo CAN 2.0B, este se usa como protocolo para comunicar serialmente con otros controladores en ambientes con ruido eléctrico. Las características del módulo CAN son las siguientes [7]: *i)* Obedece a la versión 2.0B, *ii)* soporta velocidades de datos mayores a 1Mbps, *iii)* Posee 32 mailbox, cada uno con las siguientes características: Configurable como receptor o transmisor, soporta datos y estructura remotas, soporta desde 0 a 8 bits de datos, utiliza 32- bits de tiempo en recibir y transmitir mensajes, *iv)* Programable como actividad bus, *v)* Responde automáticamente un mensaje remoto, *vi)* Opera en modo de auto-prueba, es decir, funciona en un modo lazo cerrado que recibe su propio mensaje.

Estructura del interna módulo CAN en la tarjeta TMS320F2808 [3]		
Constitución	**Componentes**	**Registros**
Constituido por 32 mailboxes para objetos con una longitud de datos de 0-8 byte, configurables como: - **Mailboxes transmisor/receptor** - **Identificador estándar o extendido**	- MID: contiene el identificador del mailbox. - MCF (Campo de control del mensaje): contiene la longitud del mensaje (receptor o transmisor) y el bit RTR (Remote Transmission Request- usado para enviar tramas remotas) - MDL y MDH: contiene los datos.	Están divididos en 5 grupos y están localizados en los datos de memoria desde 0x006000 hasta 0x0061FF y son: - Registros de control y estado - Máscara de aceptación Local - Message Object Time Stamps - Message Object Timeout - Mailboxes

Tabla 7. Estructura del interna módulo CAN en la tarjeta TMS320F2808
Fuente: Elaboración Propia

En la siguiente figura se observa la estructura interna del módulo CAN en el controlador TEXAS (TMS320F2808),

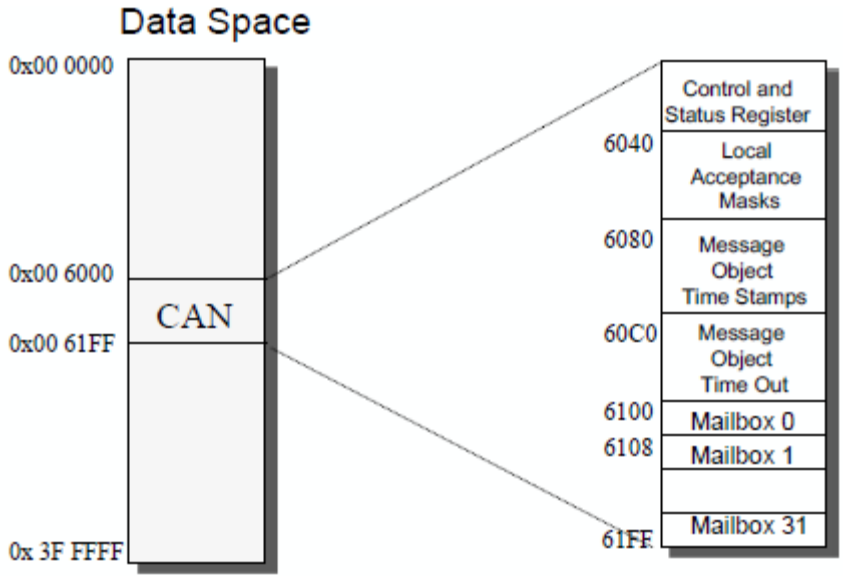


Figura 11. Estructura interna módulo CAN **[3]**

1.2. Comunicación mediante el protocolo LXI

La comunicación y configuración de las fuentes Magna Power debe realizarse a través del estándar LXI (*Lan eXtensions for Instruments*), que en la actualidad ha sido estudiado y utilizado en diferentes tipos de dispositivos de comunicación industrial.

1.2.1. Protocolo LXI

El estándar LXI define dispositivos que usan un estándar abierto LAN (Red de área local) (Ethernet) para sistemas de comunicación entre dispositivos [12].Dicho estándar posee atributos claves como son: *i)* Una interfaz LAN estandarizada que proporciona una estructura para la web. La interfaz soporta operaciones punto a punto tan bien como la operación maestro esclavo. LAN tolera intercambio de señales a través de disparo, *ii)* Basado en IEEE 1588, permite módulos que tengan un sentido del tiempo y eventos de disparo sobre la interfaz LAN [13]. Físicamente el alambrado de un sistema de disparo está basado en M-LVDS (*Multi Low Voltaje Diferential Signal*) interfaz eléctrica que permite que los módulos estén conectados entre ellos por una interfaz alambrada usando una línea de transmisión de par trenzado, *iii)* Los eventos de disparo sobre LAN pueden ser enviados por UDP (User Datagram Protocol) para minimizar la latencia que se puede experimentar con los protocolos TCP/IP [14].

LXI es un protocolo de comunicación basado en LAN sucesor de GPIB (estándar de conexión que permite la comunicación de un ordenador con insrumentos electrónicos de medida [15]), provee habilidades adicionales que permite la creación de sistemas de pruebas más eficientes,

fáciles y rápidas. Las principales ventajas de LXI son: *i)* Velocidad, simplicidad, ubiquidad, bajo costo, mejora en desarrollo y compatibilidad con LAN, *ii)* Configuración fácil a través de una interfaz web construida en un instrumento, *iii)* LXI provee la estructura para una interfaz web, *iv)* Fácil de programar a través de IVI drivers, *v)* Para aplicaciones complicadas, es muy fácil crear un software de manejo a través de IVI drivers y VISA (*Virtual Instrument Software Architecture*) que permite una comunicación fácil entre el controlador Ethernet en un PC y el instrumento LXI [16].

Conrad Proft publicó el paper titulado *"The power of LXI Instrumentation – Use Cases and Performance* [17]*"* en el cual proporciona una descripción detallada de los instrumentos LXI. Proft explica que los instrumentos LXI poseen un servidor web que hace posible acceder al instrumento usando un estándar de un navegador web desde cualquier plataforma, dando así un acceso remoto a los instrumentos. El consorcio de LXI definió tres clases de instrumentos:

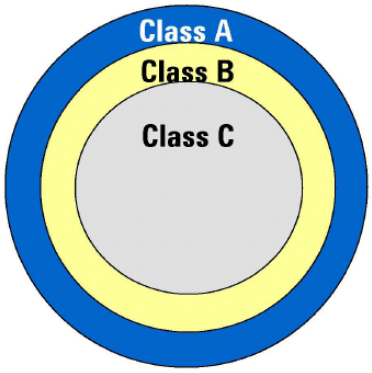


Figura 12. Clases de instrumentos LXI **[18]**

Clase C: los instrumentos de esta clase tienen la habilidad de controlarse y programarse a través de la red LAN y pueden trabajar junto con otros equipos de diferentes proveedores.

Clase B: Integra todas las habilidades de los instrumentos clase C, además soporta el protocolo IEEE 1588 el cual se enfoca en un protocolo de tiempo, precisión y sincronización.

Clase A: Integra las habilidades de los instrumentos clase B, además se comunica a otros instrumentos utilizando un hardware denominado Bus trigger.

Las fuentes Magna Power a implementar en el presente proyecto pertenecen a los dispositivos clase C. Este tipo de instrumentos son la base de las clases la cual incluye todos los requerimientos para la interfaz LAN y una interfaz de un navegador web. Los dispositivos clase C están adecuados para aplicaciones donde existen instrumentos que no son LXI pero que pueden ser adaptados al estándar, así mismo no es necesario ofrecer funcionalidad de disparo o de tiempo. Esta clase puede incluir físicamente productos pequeños, tales como, sensores, entre otros. En la mayoría de los casos los instrumentos clase C son equivalentes a los instrumentos GPIB [18] (Instrumentos electrónicos programables- norma IEEE- 488.2) [19] en funcionalidad, excepto por la interfaz LAN adicional, construida en un servidor web y con comportamiento LXI.

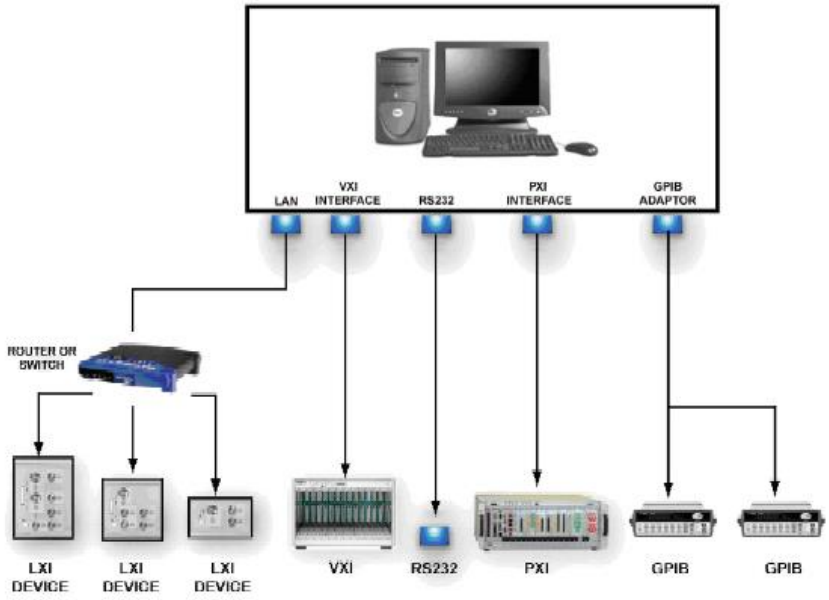


Figura 13. Sistema LXI que incluye dispositivos LXI e instrumentos no LXI **[13]**

1.2.2. Protocolo LXI en las fuentes Magna Power

Las fuentes programables de alimentación DC (XR125-16/208 XR series *programmable DC Power Supply* y XR600-9.9/208 XR Magna Power), combinan DC power (corriente directa, tipo de energía eléctrica que es producida por celdas de combustible, baterías y paneles solares) con un multiprocesador con control integrado [20]. Se encuentran programadas para tener funciones de control accesibles a través del panel frontal, conector RS232, IEEE-488 o mediante comunicaciones vía Ethernet [21]. Las fuentes XR ofrecen entradas análogas para modular la posición del voltaje o la corriente mediante una aproximación lineal a trozos, esta modulación permite que pueda ser utilizada para aplicaciones avanzadas de procesos de control, así como emulación de fuentes de energía.

Figura 14. Magna Power Electronics, XR series **[22]**

Las fuentes Magna Power serie XR, incluye un software con interfaz remota que suministra un control desde el servidor, dicho software provee: *i)* un panel de control virtual que emula el suministro de energía del panel frontal, *ii)* un panel de comando que envía comandos SCPI (Standard Comands for Programmable Instruments) *iii)* Panel de registro que supervisa dichos registros, *iv)* un programa oficial del fabricante para mejorar el control del microprocesador y Finalmente, un panel de modulación para programar fácilmente los parámetros de modulación [20].

En la comunicación vía Ethernet se incluye un servidor web que permite que el usuario observe y ajuste la posición de los módulos de la red y además provee un control básico. Magna Power como opción Ethernet utiliza el estándar LXI clase C [23].

El módulo Ethernet se debe configurar mediante el protocolo de Configuración de Host Dinámico (DHCP): configuración dinámica de los parámetros de red necesarios para que un sistema pueda comunicarse efectivamente [24], en el evento que el servidor DHCP no puede localizarlo, el dispositivo automáticamente selecciona un dirección IP de la subred 169.254 así como esta descrito en el estándar RFC 3927 (Sistema de redes privada: gama de dirección acoplamiento-local codificado dentro de dicho estándar, proporciona una dirección IP e implícitamente conectividad de la red). Esta rutina permite que el usuario pueda usar la interfaz Ethernet en ausencia de un servidor DHCP.

El cable a estándar de conecxión es redondo consiste de 8 pares trenzados y cruzados, cada par trenzado debe estar cubierto por aluminio y además debe estar construido bajo las siguientes características eléctricas según el estándar [25]:

Nom. Diff. Characteristic Impedance	100 ohms (+10 ohms, -15 ohms)
Nom. Inductance @ 10 kHz	0.47 uH/meter
Nom. Capacitance – Conductor To Conductor @ 10 kHz	41.0 pF/meter
Nom. Capacitance – Conductor To Other Conductor & Shield @ 10 kHz	65.6 pF/meter
Nom. Velocity of Propagation	74%
Nom. Conductor DC Resistance @ 20C	14.57 ohms / 100 meters
Nom. Outer Shield DC Resistance @ 20C	1.7 ohms / 100 meters
Max. Operating Voltage – UL	30 V RMS

Figura 15. Características del cable **[25]**

La interfaz Ethernet reconoce diferentes unidades que siguen ciertos estándares; en el caso de instrumentos LXI clase C (fuentes magna power) se utiliza la siguiente conexión:

Figura 16. JS5, Ethernet (visto desde terminación hembra) **[20]**

2. ESPECIFICACIONES

La plataforma de software diseñada para el presente proyecto integrará 3 fuentes Magna Power que se configurarán a través del estándar de comunicación LXI (*LAN Extensions for Instrumentations*). Dicho estándar utiliza como centro de comunicación entre dispositivos: LAN (Ethernet). Este protocolo permite la comunicación entre varios dispositivos, asi como es posible acceder remotamente al instrumento a través de un navegador web permitiendo al usuario manejar los parámetros y funciones del dispositivo LXI.

Las fuentes Magna Power se encuentran programadas para tener funciones de control a través de Ethernet, lo que indica que la comunicación vía Ethernet se hará a través de un servidor Web que permita que el usuario observe y ajuste la posición de los módulos de la red y además tenga un control básico sobre el instrumento.

El control del instrumento LXI se hará a través del servidor web mediante la dirección IP de cada dispositivo, dicho servidor a su vez proporciona acceso a los drivers, programas y documentación del instrumento. Una vez configurada la dirección IP del instrumento LXI (Fuentes Magna Power) la cual debe encontrarse en la misma red (mismo identificador) que las demás así como el router y el PC, es posible conectarlas al router y obtener una red LAN privada que maneje los instrumentos LXI desde un solo controlador (PC).

Dicha plataforma también integrará 1 controlador TEXAS (TMS320F2808) que permite desarrollar y ejecutar el funcionamiento de aplicaciones en tiempo real, a su vez dicho controlador cuenta con un módulo eCAN integrado compatible con la versión CAN 2.0B, este protocolo de comunicación permite establecer comunicación serial con otros controladores en ambientes con ruido eléctrico lo que indica que dicho módulo provee una interfaz de comunicación serial y robusta [26].

CAN (*Controller Area Network*) usa un protocolo de comunicación serial multimaestro que soporta aplicaciones de control en tiempo real y velocidad de comunicación de hasta 1Mbps. Su principal característica es enviar mensajes en sistemas multimaestro de hasta 8 bits transmitiéndolos por orden de prioridad en función de un protocolo de arbitraje de envío datos y detección de errores de los mismos.

Protocolo	Entradas	Procesos	Salidas
LXI TCP/IP Ethernet Communications Interface.	• Fuentes (XRii125-16 XR series programmable DC Power Supply Magna Power) • Servidor (PC maestro) • Conectores estándar RJ-45	• Configuración de los dispositivos LXI (fuentes XRii125-16 XR series programmable DC PowerSupply Magna Power) a través del estandar de instrumentación LXI. • El control de los dispositivos LXI se hará a través de la creación de una red LAN, se debe programar la dirección IP del dispositivo de tal manera que todas las fuentes, router y PC pertenezcan a la misma red. Así mismo se instalará el software (photovoltaic power profile emulation) del dispositivo LXI que permita programar la curva I-V según lo que desea el usuario(modulación de un panel solar).	Habilidad para controlar los dispositivos LXI a través de un navegador remoto que permita que el usuario pueda acceder a sistemas de prueba que ofrecen las fuentes Magna Power tales como modulación de paneles solares, fue cells o baterías.
Tecnología DSP de Texas Instruments implementada en la tarjeta de evaluación eZdsp F2808.	• 1 Kit eZdsp F2808: contiene el controlador TMS320F2808, una tarjeta de evaluación con los conectores necesarios y un CD el software y drivers necesarios. • Servidor (PC maestro)	• Instalación del software específico suministrado por el fabricante (Texas instruments) Code composer studio V3.1 • Inicialización módulo CAN y configuración de los parámetros de funcionamiento	Interoperabilidad del PC maestro con el controlador Texas (TMS320F2808) de manera que sea posible el intercambio de datos en tiempo real a gran velocidad.

Tabla 8. Tabla de especificaciones de LXI
Fuente: Elaboración Propia

El siguiente diagrama de bloques agrupa las actividades a realizar en orden jerárquico y consecutivo de cómo implementar la plataforma de software que integre CAN y LXI.

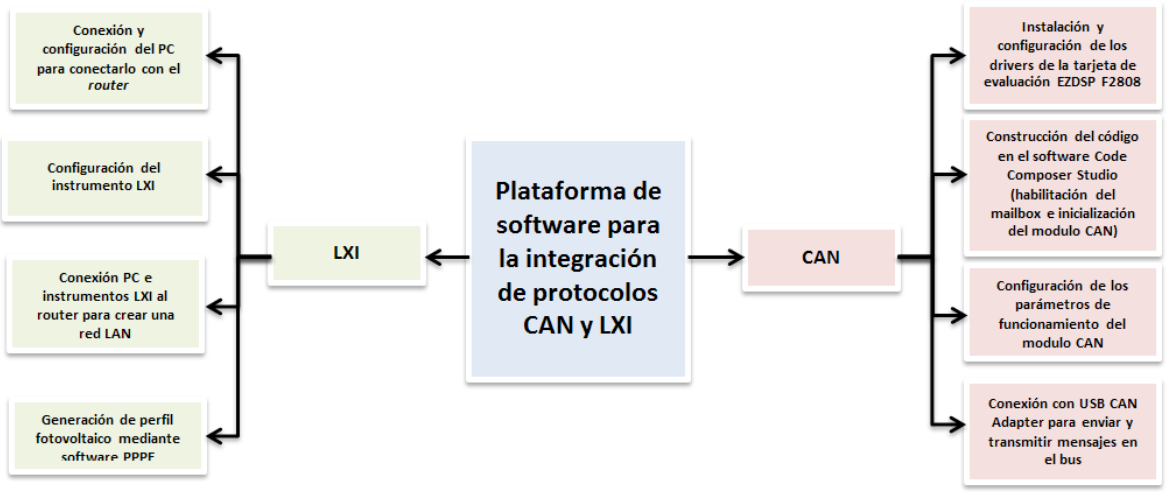
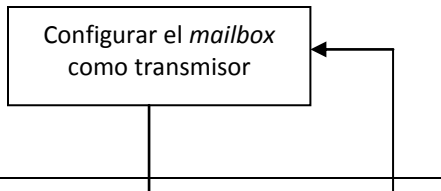
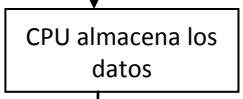
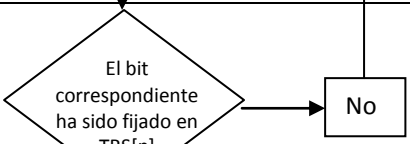
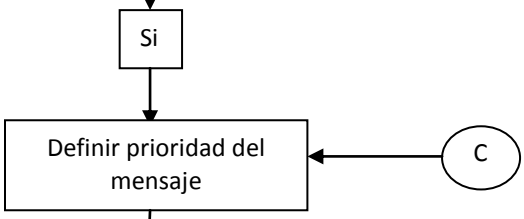
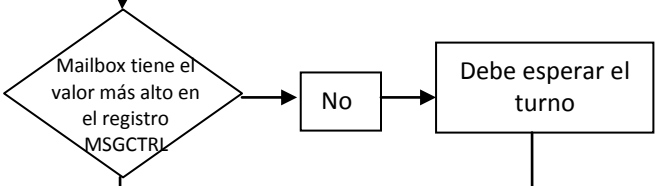
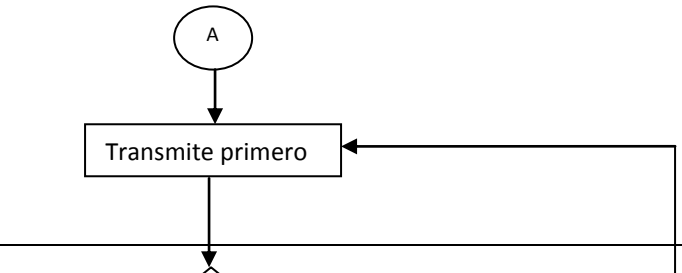
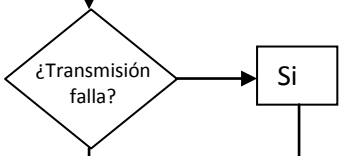
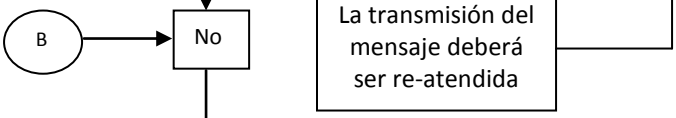
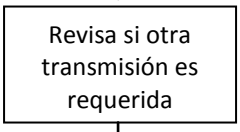
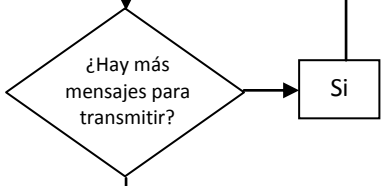
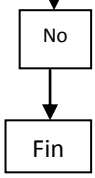
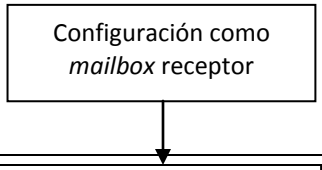
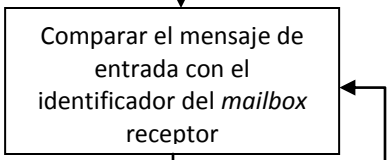
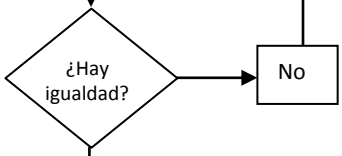
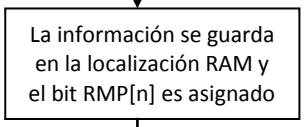
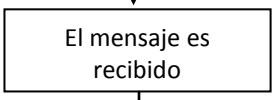
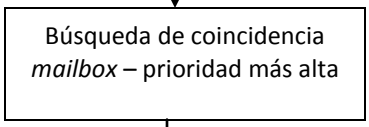
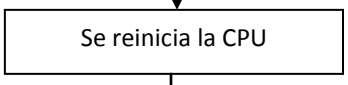


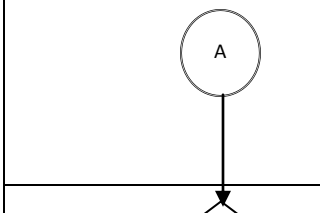
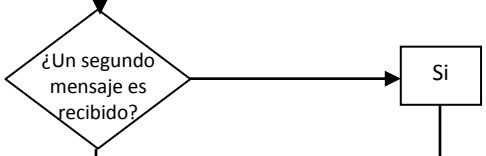
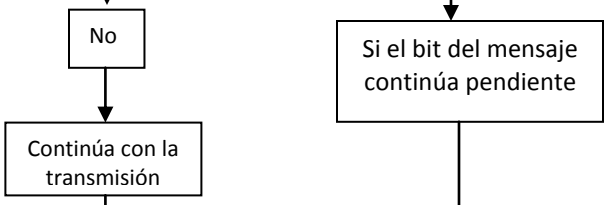
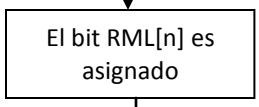
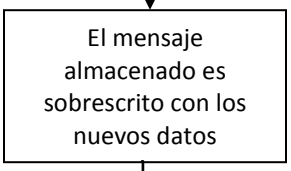
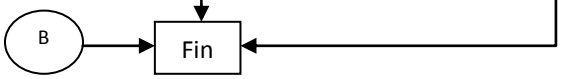
Figura 17. Diagrama de bloques
Fuente: Elaboración Propia

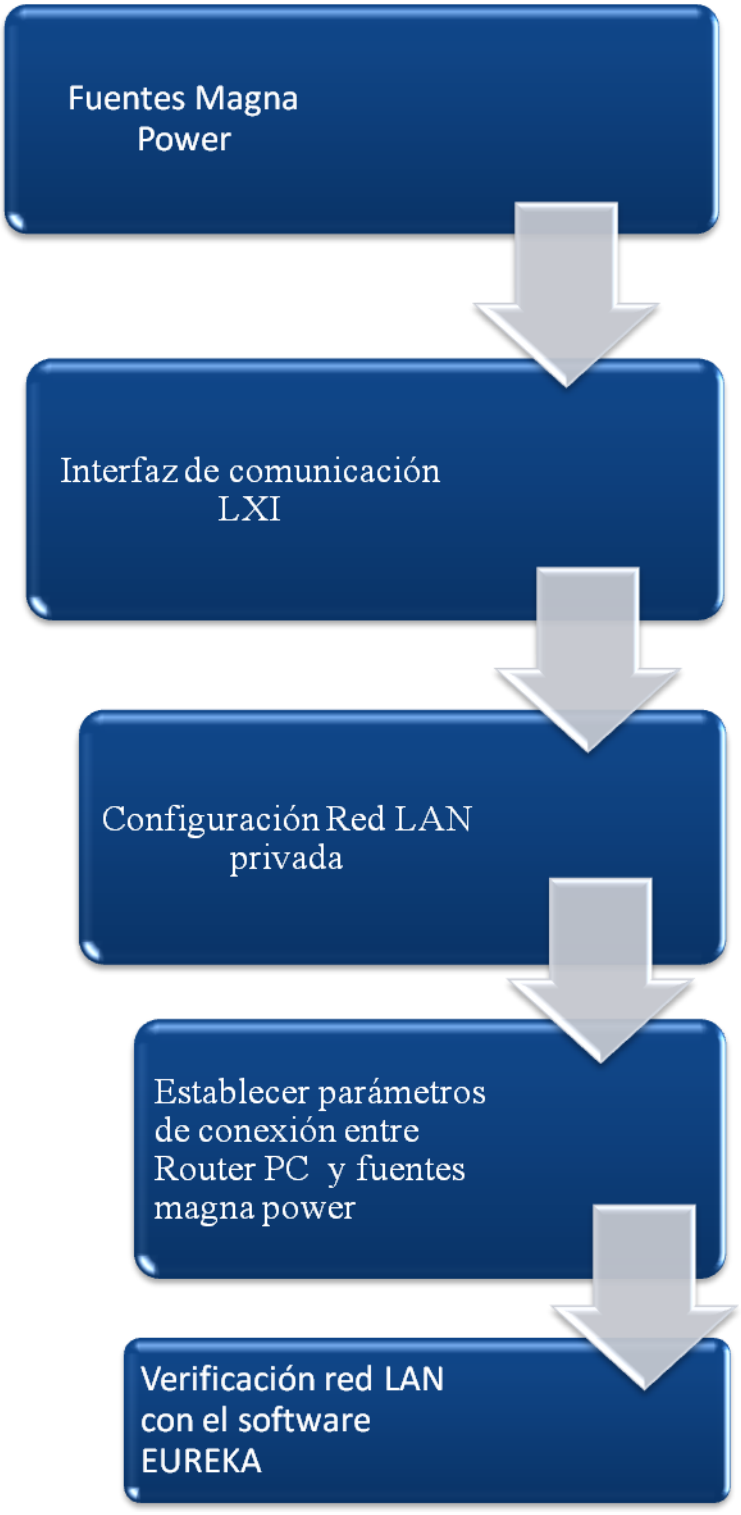
Figura 18. Plataforma de software para la integración de protocolos CAN y LXI
Fuente. Elaboración Propia

Trasmisión de un mailbox	
Flujograma	**Descripción**
	En el registro CANME.N habilitar el bit correspondiente.
	Escribe los datos y la identificación en la RAM
	Cuando el *mailbox* n está listo para transmitir, se debe colocar el bit TRS[n] a 1 para comenzar la transmisión.
	Si existe más de un mailbox configurado como transmisor, los mensajes serán enviados en orden dependiendo de la prioridad.
	El registro MSGCTRL posee una trama con distintos campos. El campo TPL Define la prioridad del mensaje.
	
	

Trasmisión de un mailbox	
Flujograma	**Descripción**
	El *mailbox* con el máximo valor de prioridad en el TPL transmite primero. Cuando dos *mailbox* poseen el mismo valor en el TPL el *mailbox* con número mayor transmite
	Si llega existir pérdida de arbitraje o un error
	La transmisión deberá ser re-atendida
	El módulo CAN chequea si se requiere transmitir
	Transmite el *mailbox* que tenga la mayor prioridad
	Valida si existen más mensajes para transmitir
	

Recepción de un *mailbox*	
Flujograma	**Descripción**
	Inicio del proceso.
	El identificador de cada mensaje de entrada es comparado con el identificador que lleva en el *mailbox* receptor utilizando la máscara apropiada
	El mensaje no es almacenado
	
	El identificador receptor, bits de control, bits de datos se escriben en la memoria en la posición de la RAM.
	
	Cuando el mensaje es recibido, el controlador del mensaje comienza a buscar si coincide el identificador y el mensaje con mayor prioridad
	El bit RMP[n] reinicia la CPU después de leer los datos
	

Recepción de un *mailbox*	
Flujograma	**Descripción**
	
	El mensaje se sobre-escribe con el nuevo dato
	Si continúa pendiente el bit de recepción del mensaje
	el bit correspondiente de pérdida del mensaje es asignado (RML[n])
	
	Si un *mailbox* es configurado como *mailbox* receptor y el bit RTR es asignado, el *mailbox* puede enviar una trama remota

mplementación LXI	
Diagrama	**Descripción**
	Los instrumentos LXI utilizan el protocolo TCP/IP, cada instrumento cuenta con una dirección MAC única y la interfaz web de comunicación de dicho instrumento permite la sincronización y configuración de los parámetros de dicho dispositivo El estándar LXI utiliza Ethernet como mecanismo de control que permite manejar varios dispositivos en un sistema de prueba a través de una red LAN privada, en este caso para la implementación del protocolo de instrumentación LXI se hace necesario configurar múltiples instrumentos (3 fuentes Magna Power) en una red LAN privada utilizando un PC y un router. La dirección IP del router proporcionada (192.168.1.2), nos indican las pautas para configurar las direcciones IP de los demás instrumentos LXI así como del PC. Los identificadortes de red (NetID) deben ser idénticos y los identificadores de host (hostID) deben cambiar. Una vez se han configurado los parámetros de conexión se procede a realizar conexión física de todos los instrumentos implicados en la red Finalmente, se comprueba con el software Eureka que todos los instrumentos LXI estén en la misma red. Eureka es un software diseñado específicamente para reconocer la dirección IP de los instrumentos LXI en una red

3. DESARROLLOS

Para el intercambio de datos que suministra el usuario o las fuentes Magna Power, se ha utilizado un kit de evaluación eZdspF2808, el cual contiene el controlador se señales digitales (DSC) , los conectores necesarios para su conexión al PC y un CD con el software y drivers necesarios. La aplicación en tiempo real desarrollada en este proyecto como ejemplo de programación, tiene la finalidad de realizar la gestión, supervisión, control e intercambio de datos en tiempo real.

La tarjeta de evaluación eZdspF2808 (figura 20) permite desarrollar y ejecutar aplicaciones en tiempo real, dicha tarjeta se suministra con el controlador de señales digitales DSC TMS320F2808.

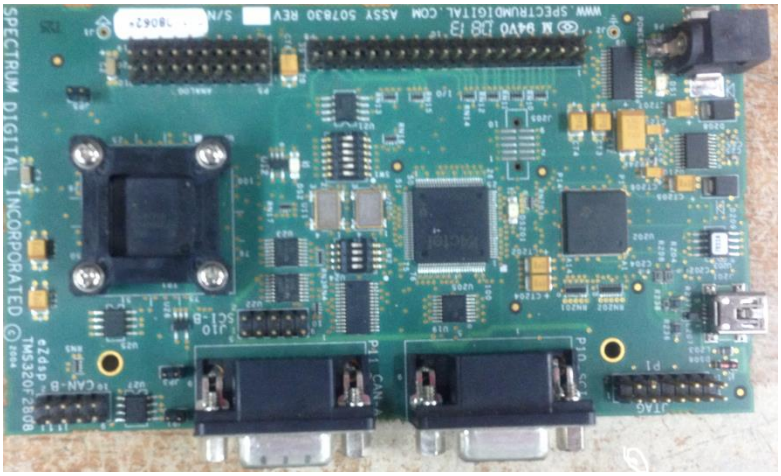


Figura 19. Imagen real tarjeta de desarrollo TMS320F2808

Los controladores de Texas Instruments se clasifican por familias dependiendo de su numeración, el DSC TMS320F2808 pertenece a la familia de procesadores C2000. Los controladores de señales digitales son diseñados para realizar soluciones de alto rendimiento en aplicaciones de control.

El controlador TEXAS (TMS320F2808) y la tarjeta de evaluación eZdspF2808 permiten desarrollar y ejecutar el funcionamiento de aplicaciones en tiempo real, dicha tarjeta se encuentra acompañada con un controlador de señales DSC (Digital Signal Controllers) TMS320F2808 que posee las siguientes características hardware [12]:

- DSC TMS320F2808.
- Chip de memoria Flash de 64K x 16 Flash.
- Convertidor ADC de 12 bits con 16 canales de entrada.
- Interfaz CAN 2.0 con línea de drivers y conector.
- Controlador USB JTAG incrustado.
- Soporta velocidades de datos mayores a 1Mbps.

En un controlador CAN pueden incluirse varios registros de recepción, conocidos como mailbox, Cada uno de estos se pueden programar para la recepción de un mensaje en concreto, en este caso el controlador se convierte en una memoria compartida entre el nodo y la red, y direccionable por medio del identificador CAN [11]. Se identifican 32 *mailboxes* en el módulo CAN de las tarjetas TMS320F2808 que poseen las siguientes características [3]:

- Configurables como transmisor o receptor.
- Configurable como identificador estándar o extendido.
- Soporta datos y tramas remotas.
- Compuesto de 0 a 8 bits de datos.
- Usa 32-bits time stamp (secuencia de caracteres que denotan fecha y hora) en mensajes
- Esquema de interrupción programable (2 niveles).
- Alarma Programable para tiempos de desconexión
- Alerta programable de la actividad del bus
- Modo de prueba

El modelo de implementación física de la plataforma de software con el protocolo de comunicación CAN se observa en la siguiente figura:

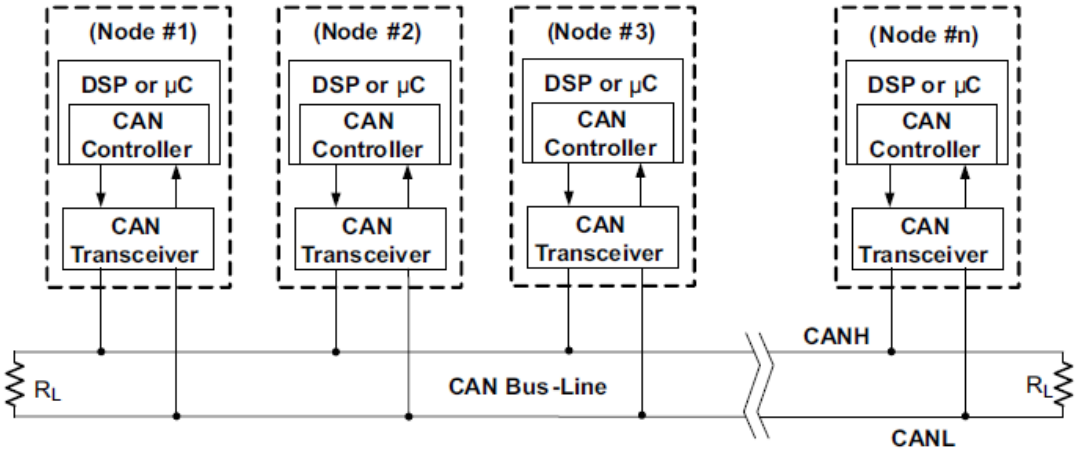


Figura 20. Detalles de un Bus CAN **[4]**

El bus de comunicación CAN se implementó de la siguiente manera: Se conectó el controlador Texas de manera serial mediante la interfaz CAN2.0 al PC maestro, así como también se instalaron los drivers de la tarjeta de evaluación eZdsp F2808. Una vez inicializado el módulo es necesario configurar los siguientes parámetros [8]: *i)* Activar el reloj del módulo, *ii)* Configurar un buzón para transmitir, *iii)* Configuración de la transmisión del mensaje, *iv)* Transmitir un mensaje, *v)*Configurar un buzón para recibir, *vi)* Recibir un mensaje, *vii)* Solicitar datos de otro nodo, *viii)* Responder a una solicitud remota, *ix)* Actualizar los campos de los datos.

Finalmente, el bus de comunicación CAN se implemento de la siguiente manera: la salida del controlador Texas TMS320F2808 es conectada a la entrada del driver serial pin D y la salida serial pin R del transceiver. Este es conectado a las líneas diferenciales del bus a través de los pines CANH y CANL.

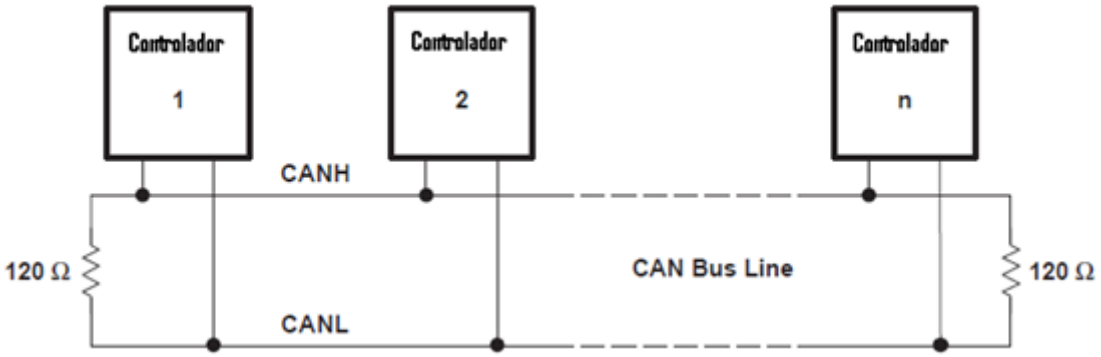


Figura 21. Bus CAN **[8]**

3.1. Configuración módulo CAN

3.1.1. Configuración del reloj

La especificación del protocolo CAN divide el bit del tiempo nominal en cuatro segmentos diferentes:

- **SYNC_SEC**:
  - Se utiliza para sincronizar los nodos en el bus
  - Longitud: siempre 1 TIME QUANTUN (TQ).
- **PROP_SEG:**
  - Se utiliza para compensar los retrasos físicos en la red.
  - Este es dos veces la suma de las señales de tiempo de propagación en las líneas del bus: la entrada de retraso en el comparador y la salida de retraso del driver Programable de 1 a 8 TIME QUANTA (TQ)
- **PHASE_SEG1:**
  - Programable de 1 a 8 TIME QUANTA (TQ)
- **PHASE_SEG2:**
  - Programable de desde 2 a 8 TIME QUANTA (TQ)

En el módulo CAN, la longitud de un bit en el bus CAN es determinada por los parámetros TSEG1, TSEG2 y BRP.

- TSEG1: Combina los dos segmentos PROP_SEG y PHASE_SEG1
- TSEG2: Define la longitud del segmento de tiempo PHASE_SEG2

- BRP: Baud rate Prescaler

De acuerdo al estándar CAN, las siguientes reglas se deben tener en cuenta para determinar cada uno de los valores de los segmentos:

- TSEG1(min) ≥ TSEG2
- 3/BRP ≤ tseg1 ≤ 16 TQ
- 3/BRP ≤ tseg2 ≤ 8 TQ

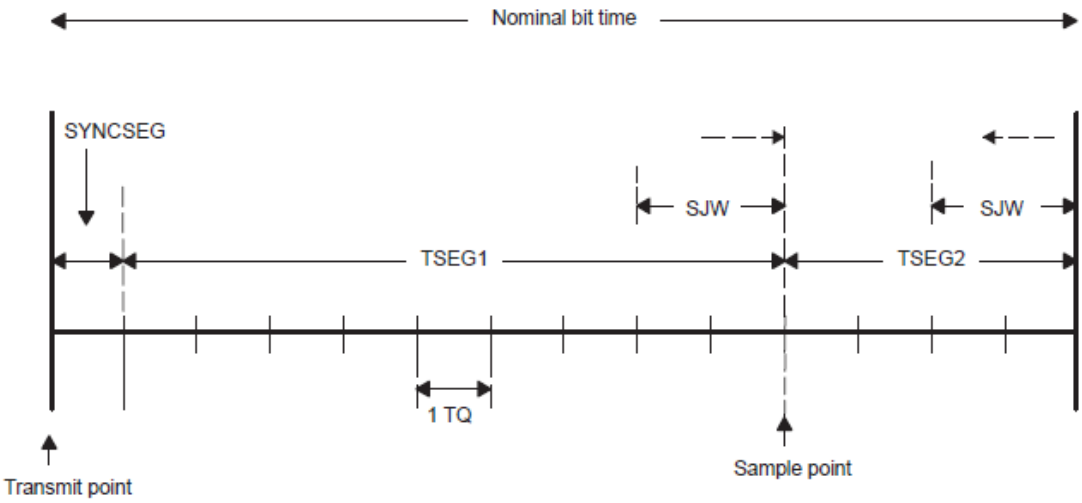


Figura 22. CAN Bit timing **[7]**

3.1.2. Parámetros de bit de configuración

La Velocidad de Baudios de CAN es 1Mbit/s, los parámetros de configuración a tener en cuenta para un reloj de 100 MHz de acuerdo a la guía de referencia de CAN para TMS320F2808 [7] son los siguientes:

BRPREG=9 BRP(Baud Rate Prescaler)

TSEG2REG=1

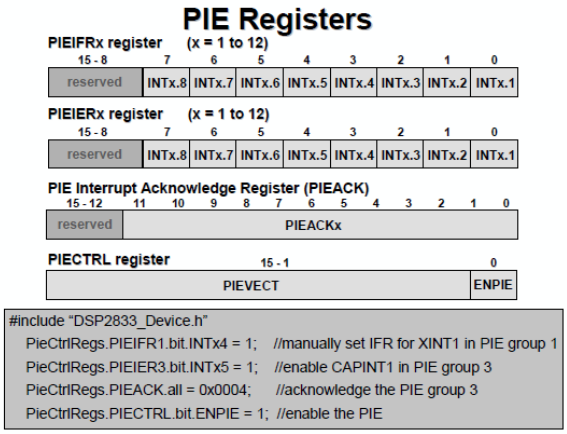
TSEG1REG=6

La inicialización o cambio de dichos parámetros se realiza en el archivo DSP2833x_ECan.c

3.2. Estructura del código de programación en archivo .C

La siguiente tabla muestra cada una de las estructuras y registros utilizados en la elaboración del código para configurar el módulo CAN y su funcionalidad.

Estructura	Observaciones
ECANbshadow	Esta estructura es utilizada como una copia local de los registros originales de CAN, es necesaria debido a que algunos registros son únicamente accesibles por 32-bits y copiando la estructura completa se asegura que únicamente se generan 32-bits de acceso [27].
EALLOW /EDIS	EALLOW: Algunos registros/bits están protegidos para modificaciones inadvertidas, estos registros pueden ser modificados si la protección EALLOW está desactivada. EDIS: Después de modificar los registros, deben ser protegidos con la instrucción EDIS [7].
WATCHDOG	Es un temporizador de control que envía un disparo de reset si este no es reiniciado periódicamente ppor una instrucción de secuencia específica. Es usado para reconocer eventos donde el programa abandona su secuencia de ejecución, por ejemplo, si el programa colisiona. Para reiniciar el registro del temporizador de control antes de que este se desborde es necesario escribir una secuencia "clave válida" en el registro WDKEY. 0X55 = Contador disponible para reset 0XAA = Colocar contador a cero si el reset es disponible [3].
Configuración módulo GPIO [28]	1. Identificar el esquema de pines, 35 GIPO disponibles con su funcionalidad. GPIO 30 =CANRXA GPIO 31 = CANTXA 2. Configurar los registros GPAMUX1, GPAMUX2, GPBMUX1 de acuerdo a una de las tres funciones periféricas. GPAMUX2: GPIO30 = CANRXA Y GPIO31 = CANTXA. 3. Seleccionar la dirección del pin: Si el pin es configurado como GPIO es necesario especificar la dirección del pin como entrada o salida en el registro GPBDIR. GPIO9 = 1 GPIO11 = 1 GPIO34 = 1

Estructura	Observaciones
Interrupts	Son eventos asíncronos, generados por unidades de hardware internas o externas. Estos eventos causan en la DSP una interrupción en la ejecución del programa que está en funcionamiento para comenzar una rutina de servicio. Una vez se haya ejecutado la rutina de servicio el programa que fue interrumpido será reanudado. **PIE** (unidades periféricas de expansión): IER (Interrupt Enable Register): Activado colocando IER=1  Figura 23. Registros PIE **[3]** InitPieCtrl() = Es una función usada para limpiar todos las interrupciones PIE pendientes, además desactiva todas las líneas de interrupción de tal manera que el usuario pueda inicializar PIE-unit según necesite.
Mailbox Enable Register (CANME)	Este registro es utilizado para activar/desactivar un mailbox individual [3].
Mailbox Direction Register (CANMD)	Este registro es utilizado para configurar un mailbox para transmitir o recibir

Estructura	Observaciones
Transmission Request Set Register (CANTRS)	Cuando un mailbox n está listo para transmitir, la CPU coloca el bit TRS[n] en 1 para iniciar la transmisión.
Transmission Acknowlede Register (CANTA)	Si el mensaje de un mailbox n fue enviado exitosamente, el bit TA[n] es fijado.
Message-Control Register (MSGCTRL)	Especifica el número de bytes para ser transmitidos así como la prioridad de transmisión. MSGCTRL.bit.DLC: El número en este bit determina cuantos datos son enviados o recibidos dentro de un rango valido de 0 a 8.
InitCpuTimers	Coloca el temporizador en un estado conocido

Tabla 9. Estructura del código de programación en archivo .C
Fuente. Elaboración Propia

Con el fin de observar los datos (mensajes) recibidos a través del Bus CAN se emplea el USB-CAN adapter que se enuncia a continuación:

3.3. USB-CAN Adapter

El dispositivo USB CAN ADAPTER (Figura 24) permite la visualización de las tramas CAN recibidas y transmitidas a través del bus. Inicialmente, por medio de la universidad se adquirió el dispositivo CAN USB Light de "Grid Connect" el cual se comunica via puerto RS232 con el computador y funciona mediante el software Tera Term(hyperterminal). Una vez fue configurado se evaluó directamente con el proveedor el por qué no era posible visualizar las tramas de datos en la interfaz. El resultado de la validación arrojó la conclusión que el dispositivo CAN USB Light no era compatible con la tarjeta EzdspF2808. Por esta razón, con recursos propios, fue adquirido el dispositivo USB CAN ADAPTER el cual se comunica mediante puerto USB y emplea como interfaz de visualización PCAN-VIEW. A continuación se explicarán los aspectos más relevantes del dispositivo:

Figura 24. PCAN-USB adapter **[29]**

3.3.1. Propiedades y características

- Adaptador USB para la conexión
- Velocidad de datos 1Mbit/s
- Compatible con la especificación de CAN 2.0A (11-Bit ID) y 2.0 (29-Bit ID)
- Conexión CAN-BUS vía D-Sub, 9 pines
- Software PCAN-View para Windows para monitorear los datos del Bus CAN

3.3.2. Conexión al Bus CAN

El bus CAN de alta velocidad (Salida de la tarjeta ezdspF2808) se conecta a la terminal de 9 pines del USB-CAN adapter, de acuerdo a la siguiente imagen:

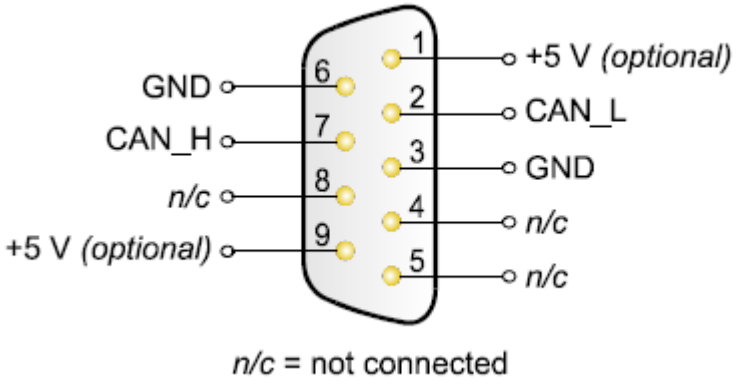


Figura 25. Asignación de Pines CAN (Visto desde el conector del PCAN-USB Adapter) **[29]**

3.3.3. Cableado

El bus CAN de alta velocidad debe poseer resistencias de terminación (120Ω) en ambas terminaciones del cable ya que impide reflexiones e interferencias de señal. Como se muestra en la figura:

Figura 26. Conexión Bus CAN con el PCAN-USB adapter

3.4. Configuración LXI

LXI (LAN eXtensions for Instrumentation), como estándar desarrollado para el control de sistemas de prueba, emplea tres tipos de configuraciones básicas de conexión, a saber:

i) Conexión del PC o controlador directamente al instrumento LXI utilizando un cable LAN (en términos generales, la forma más sencilla).
ii) Creación de su propia red local utilizando un *switch* o *router* el cual permite el control de múltiples instrumentos desde el mismo PC. Este y todos los instrumentos deseados están conectados vía LAN al switch o router [30].
iii) Conectar todos los instrumentos a una red existente (intranet).

El manual de usuario de las fuentes de alimentación Magna Power establece la procedencia de los tres métodos de comunicación anteriores. El mecanismo más viable, y por tanto empleado en la implementación de nuestra plataforma LXI es la creación de una red local utilizando un router el cual permite el control de múltiples instrumentos desde el un sólo PC.

En términos de configuraciones de red, los instrumentos LXI utilizan el protocolo TCP/IP, cada instrumento cuenta con una dirección MAC única y la interfaz web de comunicación de dicho instrumento permite la sincronización y configuración con la página web [31].

El estándar LXI utiliza Ethernet como mecanismo de control que permite manejar cada sistema de prueba a través de navegador de una página web, sin embargo, en este caso para la implementación del protocolo de instrumentación LXI se hace necesario configurar múltiples instrumentos (3 fuentes Magna Power) en una red LAN privada utilizando un PC y un *router*.

El *router* debe utilizar DHCP (Protocolo de configuración de host dinámico: configuración dinámica de los parámetros de red necesarios para que un sistema pueda comunicarse efectivamente) [24], en el evento que el servidor DHCP no puede localizarlo, el dispositivo automáticamente selecciona un dirección IP. En la siguiente figura se observa la conexión de los instrumentos LXI en una red LAN privada.

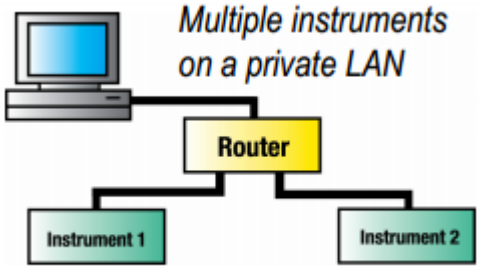


Figura 27. Conexión instrumentos LXI en una red LAN privada **[32]**

3.5. Generación del perfil fotovoltaico en las fuentes Magna Power

La generación del perfil fotovoltaico en las fuentes Magna Power se realiza a través de un software denominado **photovoltaic power profile emulation**, el cual permite una entrada análoga para simular el voltaje o corriente utilizando diagrama lineal de aproximación. Esta característica permite que el voltaje y la corriente pueda ser ajustada por un sensor de entrada, como un termistor o monitoreando su propio voltaje y corriente.

Este software utiliza parámetros dentro una matriz, creando 50 puntos en una tabla los cuales utilizan características de modulación en la fuente. Los datos pueden ser exportados en conjunto en un archivo .csv o pueden ser almacenados en tiempo real habilitando la opción data logging y colocando una carga de tal manera que a medida que esta cambia los datos se actualicen con un intervalo de 10s.

3.6. Interoperabilidad entre los dos Protocolos de comunicación

Los datos de voltaje y corriente generados por el perfil fotovoltáico (PPPE) es posible almacenarlos en el computador con una latencia de 10s en un archivo con formato .csv.

Para que puedan ser leidos automáticamente por el software PCAN deben estar en formato .xls con una estructura previamente definida: el protocolo de comunicaciones CAN en su trama de datos cuenta con ciertos campos definidos tales como: Identificador (ID), DLC, Data (Hex) entre otros. Estos campos hacen parte de la estructura del archivo .xls el cual se puede enviar a través del bus CAN, por esta razón, se decide buscar un programa que permita leer los archivos y modificarlos con la nueva estructura.

Con este fin, se creó un código en el software Matlab el cual utiliza funciones básicas, tales como, leer archivo .csv, transformar datos decimales a hexa, concatenar valores y, finalmente, exportar datos a un archivo .xls. El archivo .xls resultante puede ser enviado a través del software PCAN-Explorer.

El dispositivo USB CAN Adapter posee una plataforma que le permite exportar datos de otros archivos con formatos diferentes, PCAN Explorer (figura 28), es una herramienta que permite monitorear el tráfico de datos en el bus CAN, se encuentra integrada con VBscript el cual permite la creación de macros, lectura de archivos .xls, transmisión, análisis y almacenamiento de los datos del bus CAN.

Finalmente, el archivo .xls que se exporta desde matlab es cargado en el software Pcan Explorer, para ello se deben programar ciertas funciones tales como leer columnas, habilitar el bus, definir el cycle time (tiempo de envío de cada mensaje). Los datos son enviados a través del bus CAN y visualizados en los mailbox del software Code Composer Studio.

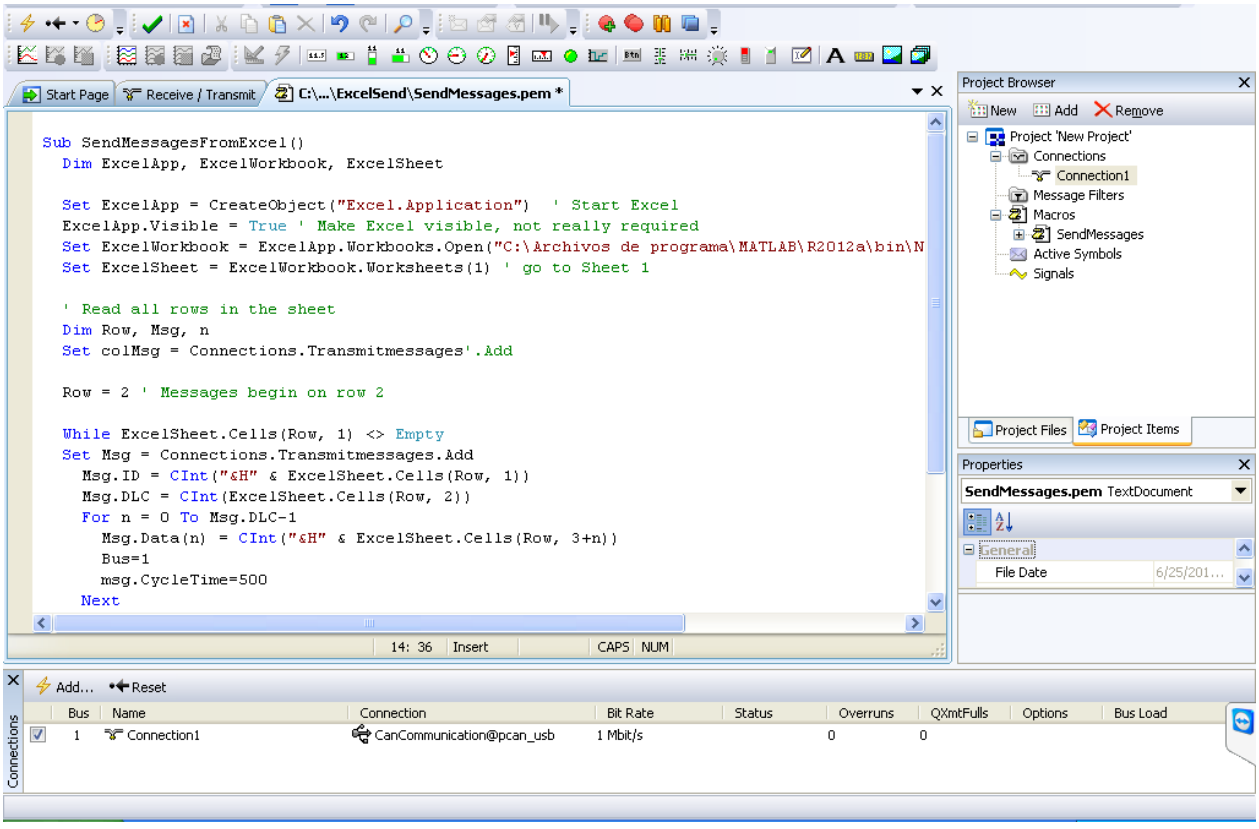


Figura 28. Software PCAN-Explorer

4. PROCEDIMIENTO

4.1 Implementación y configuración módulo CAN

Aunque inicialmente se planteó la conexión y configuración de 5 controladores Texas TMS320F28335, durante la ejecución del proyecto se pudo verificar que los controladores asignados no funcionaban correctamente, razón por la cual fue necesario modificar o adaptar el proyecto de tal forma que sólo se empleo un solo controlador Texas TMS320F2808, de la misma familia, que respondía a una configuración igual a la inicialmente propuesta.

4.2 Implementación y configuración CAN-USB Adapter

4.2.1 Instalación y Configuración Software PCAN-View

En la siguiente figura se observa que el software PCAN-View tiene la posibilidad de escoger la velocidad de recepción de datos, así como el formato de los mensajes que se desea (Estándar - Extendido)

Figura 29. Selección de parámetros CAN específicos (hardware) **[29]**

4.2.2 Recepción y Transmisión

Una vez se conecta el dispositivo a la tarjeta de desarrollo ezdspF2808 automáticamente se obtienen los mensajes que se están enviando a través del Bus CAN; el envío de dichos mensajes como se mencionó anteriormente se programó a través del Software *Code Composer Studio*. A su vez la transmisión de los mensajes se realiza directamente desde el software PCAN-View realizando lo siguiente:

1. En el menú seleccionar el comando transmitir > Nuevo mensaje y aparecerá la siguiente pantalla

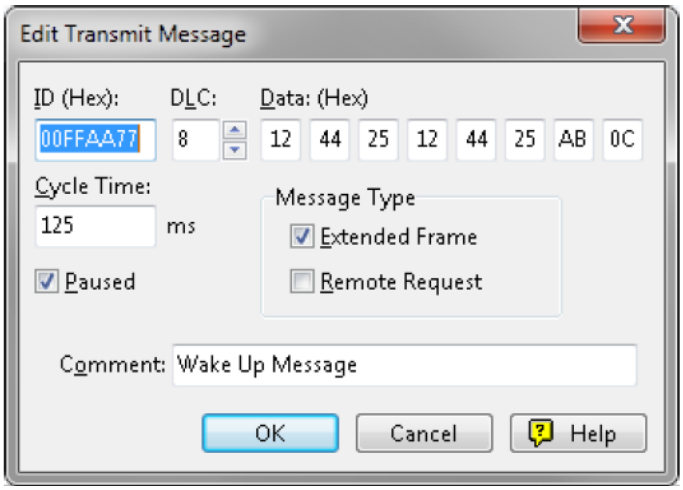


Figura 30. Ventana de transmisión de un nuevo mensaje **[29]**

2. Diligenciar el ID, DLC y los datos que se desean enviar a través del Bus CAN
3. El campo Cycle Time indica si el mensaje puede ser transmitido periódicamente o manualmente. Si se desea transmitir el mensaje periódicamente, se debe ingresar un valor mayor a 0. Modo manual un valor igual a 0.
4. Se observará en la pestaña receive/transmit los datos recibidos y enviados como se observa en la imagen:

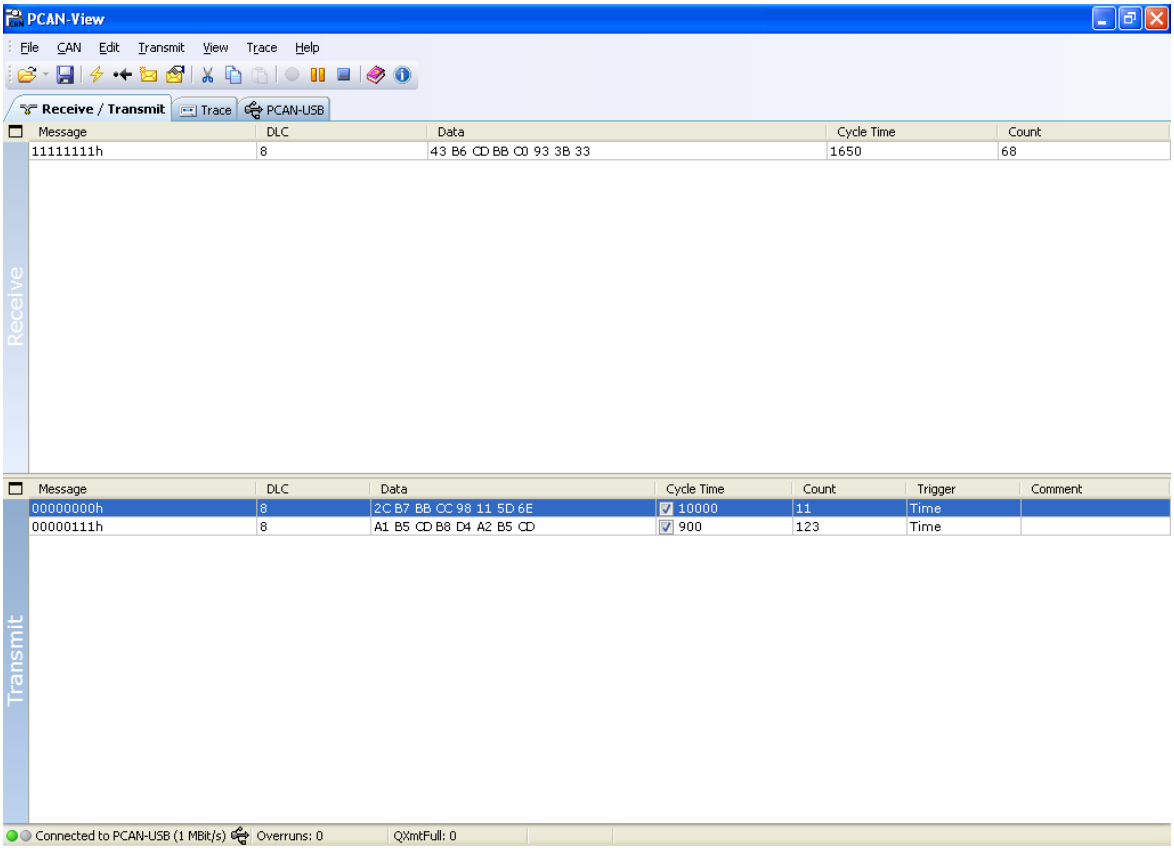


Figura 31. Imagen Recepción y transmisión de mensajes Bus CAN

En la parte superior de la imagen se encuentran los datos recibidos a través del módulo CAN. En la imagen anterior se identifican las siguientes columnas:

1. Message: corresponde al número 11111111h que es el identificador del mensaje.
2. DLC: corresponde al número de Bytes que contiene el campo de datos que en este caso es de 8.
3. Data: contiene los datos transmitidos en formato hexadecimal, en la imagen se evidencia 43 B6 CD BB C0 93 3B 33.
4. Cycle Time: Indica si el mensaje debe ser transmitido manual o periódicamente si se quiere transmitir el mensaje periódicamente debe ser un valor mayor a cero. En la imagen el valor es 1650 ms.
5. Count: Cuenta el número de mensajes recibidos, en la imagen el valor es de 68.

Finalmente, en la pestaña *trace* las columnas type, ID, DLC y Data (explicadas en la imagen anterior) indican la trama de datos que se está recibiendo y transmitiendo. Mientras que, en la columna time se visualiza el tiempo que tarda el proceso.

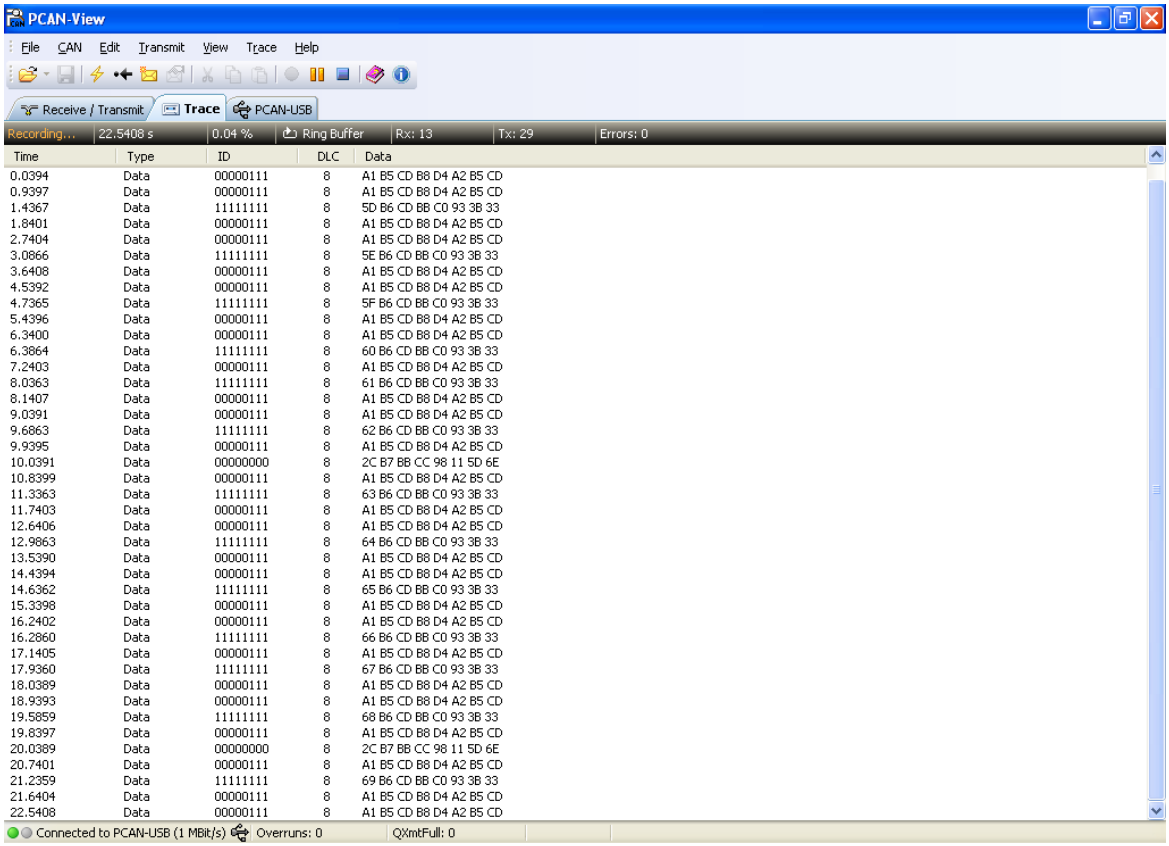


Figura 32. Imagen pestaña Trace Software PCAN-View

4.2.3 Exportación de datos en archivo XMT

El software PCAN-VIEW permite al usuario exportar datos en un archivo con extensión XMT, para ello deberán seguirse los siguientes pasos:

1. Abrir el software PCANView y seleccionar el Hardware deseado

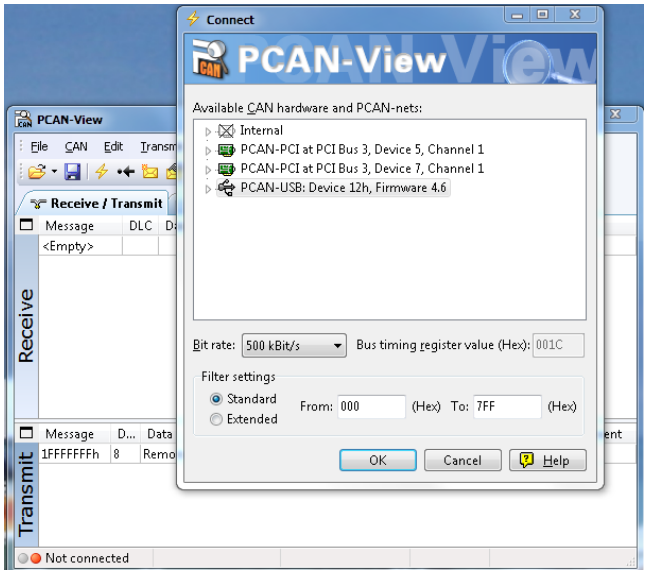


Figura 33. Imagen Selección de hardware

2. Presionar el botón insertar (Ins) y definir el mensaje que se desea enviar.

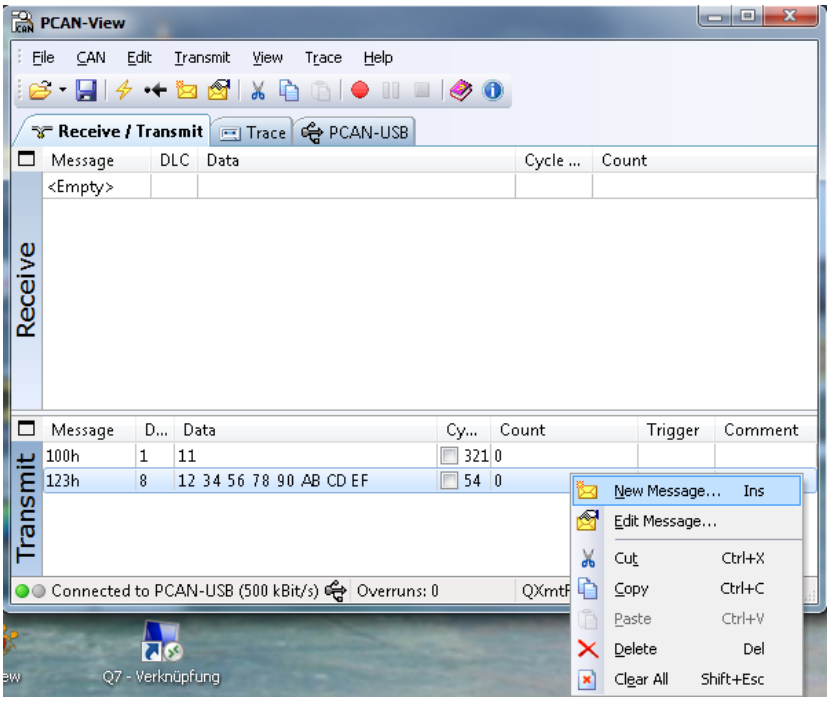


Figura 34. Imagen insertar mensaje

3. Presionar "Ctrl+S" o dirigirse a la barra del menu y seleccione guardar.

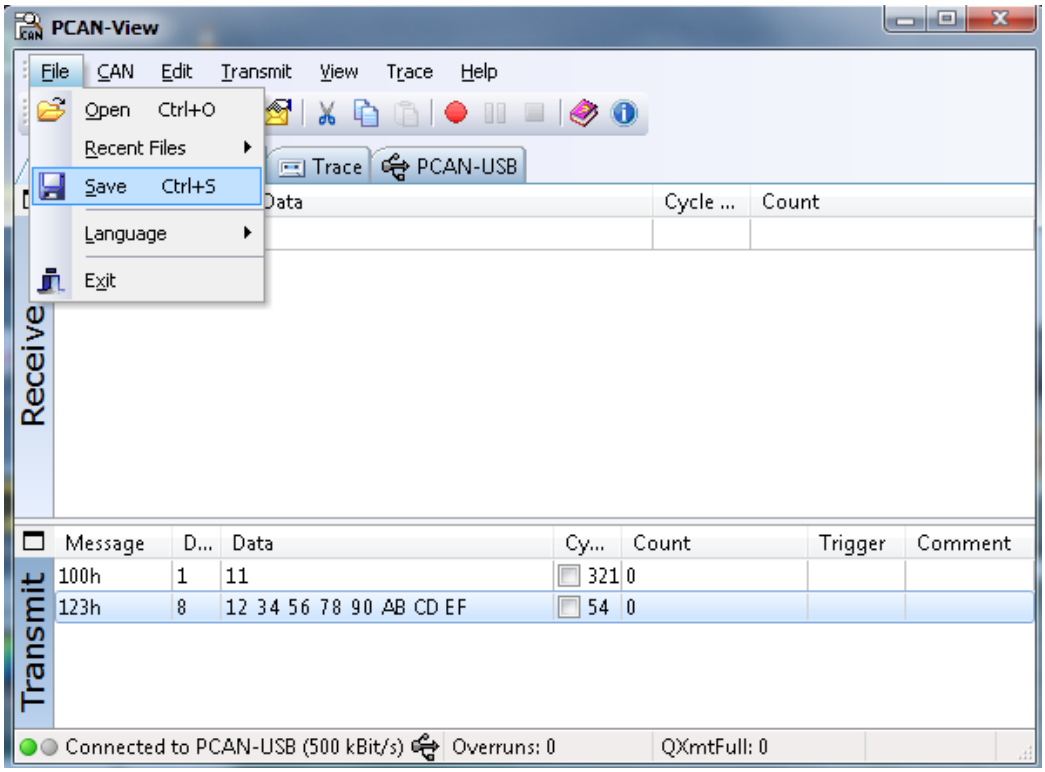


Figura 35.Imagen guardar archivo XMT

4. Seleccionar una carpeta en el escritorio en el cual se desea guardar el archivo

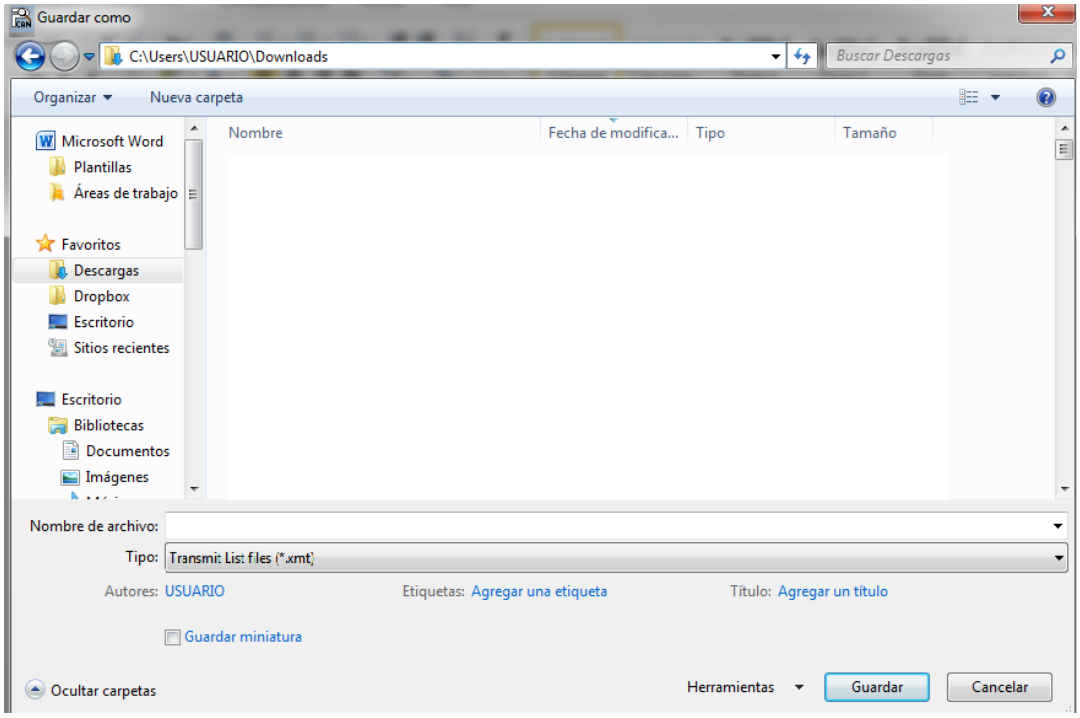


Figura 36. Imagen selección carpeta destino

5. Finalmente, se genera el archivo XMT en el cual se exportan los datos que se están enviando o transmitiendo.

4.3 Implemetación y configuración de las fuentes Magna Power en una red LAN privada

A continuación se explicará la configuración de cada una de las partes que componen la red LAN privada con el instrumento LXI.

4.3.1 Configuración fuentes Magna Power (instrumento LXI)

En primer lugar, se ha instalado un software llamado *Eureka Browser* que permite reconocer la dirección IP del instrumento LXI de manera fácil y rápida. Una vez se conozca su IP es necesario configurar la dirección IP del PC de tal manera que esté pueda comunicarse con todos los instrumentos LXI existentes en la red.

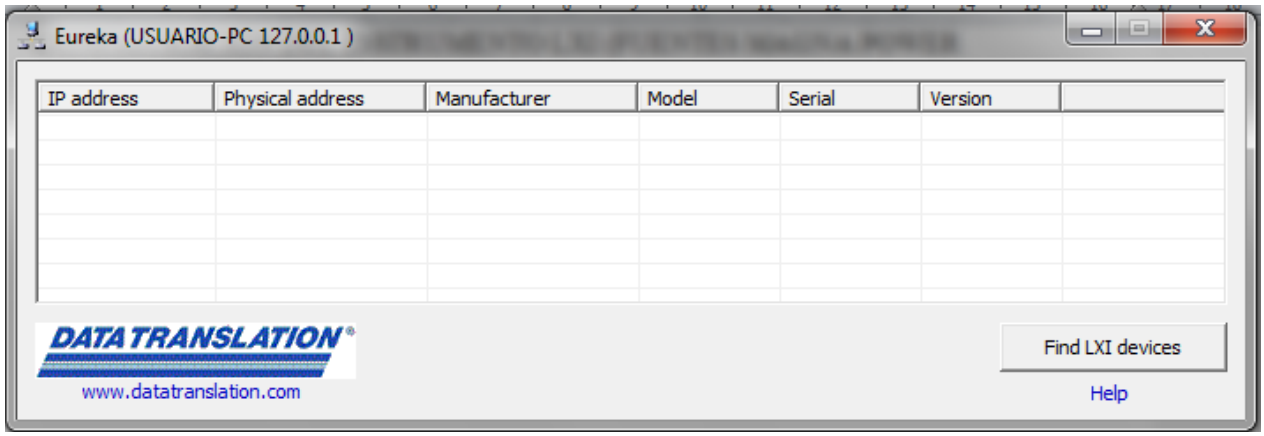


Figura 37. Imagen software Eureka

Se desea que las tres fuentes y el PC pertenezcan a la misma red, por lo tanto se realizó la configuración de la siguiente manera:

1. Inicialmente se debe conectar el PC a la fuente Magna power través del puerto RJ45 mediante un cable Ethernet. El puerto JS5 (ethernet) se encuentra en el panel trasero del instrumento así:

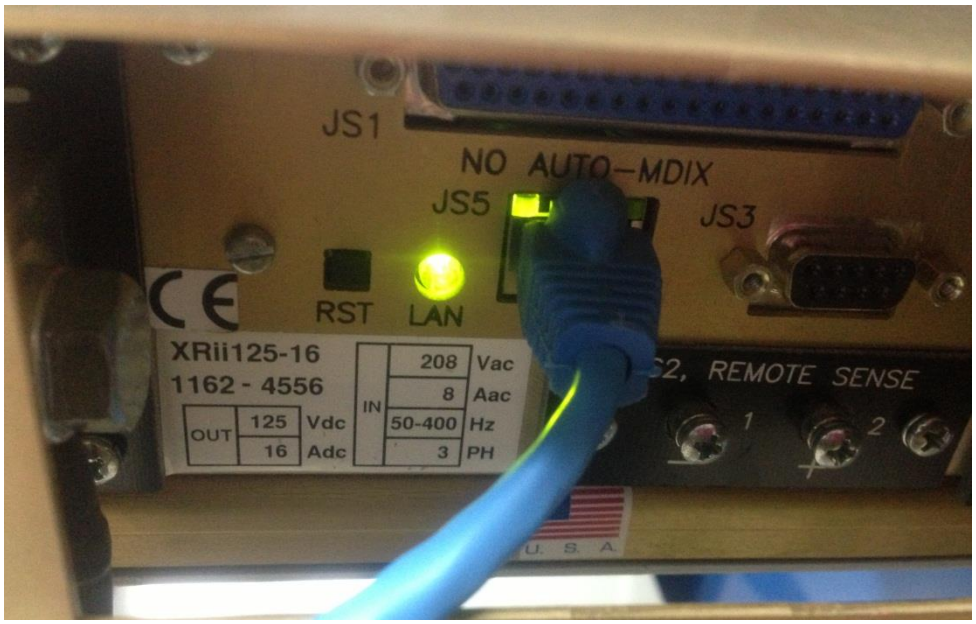


Figura 38. Imagen panel trasero Fuente Magna Power

2. Abrir el software Eureka para identificar la dirección IP del instrumento.
3. Abrir un browser web como internet explorer y digitar [http://[ipaddress]/](http://[ipaddress]/)
4. La información del instrumento se mostrara en la pantalla así:

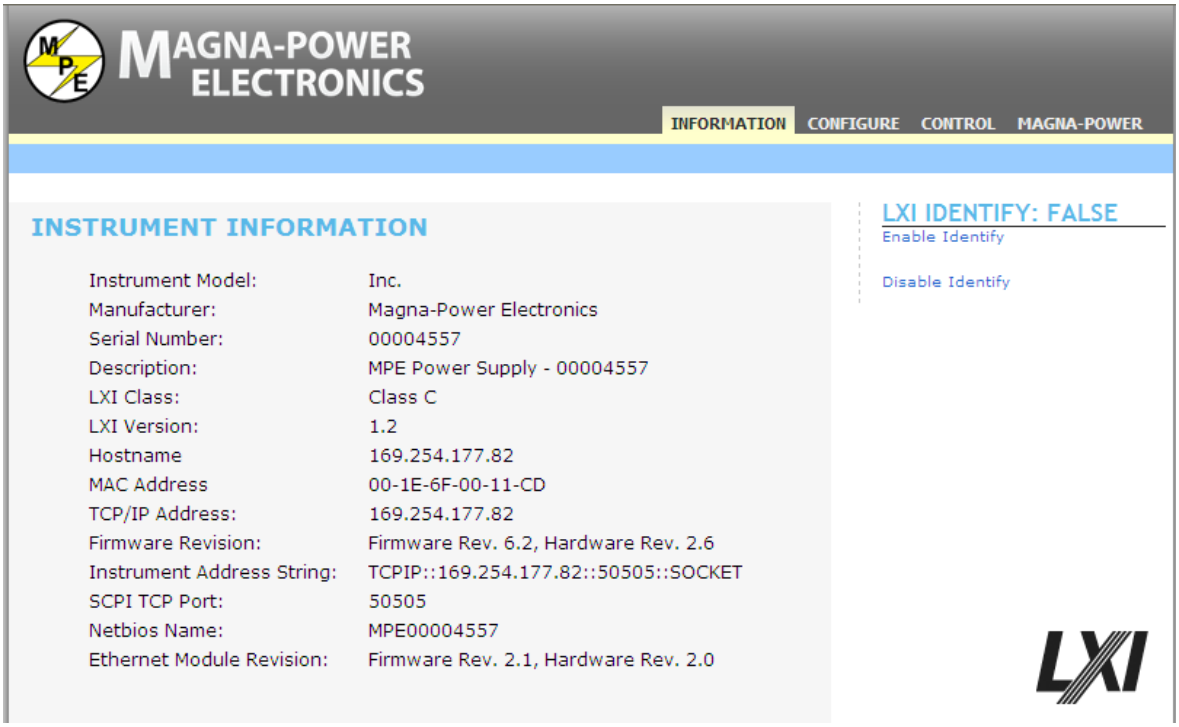


Figura 39. Panel de Información del instrumento LXI

5. En la pestaña “configure” es posible cambiar la dirección IP del instrumento y su máscara de red por un identificador de red que comparta con las otros instrumentos y con el PC, así:

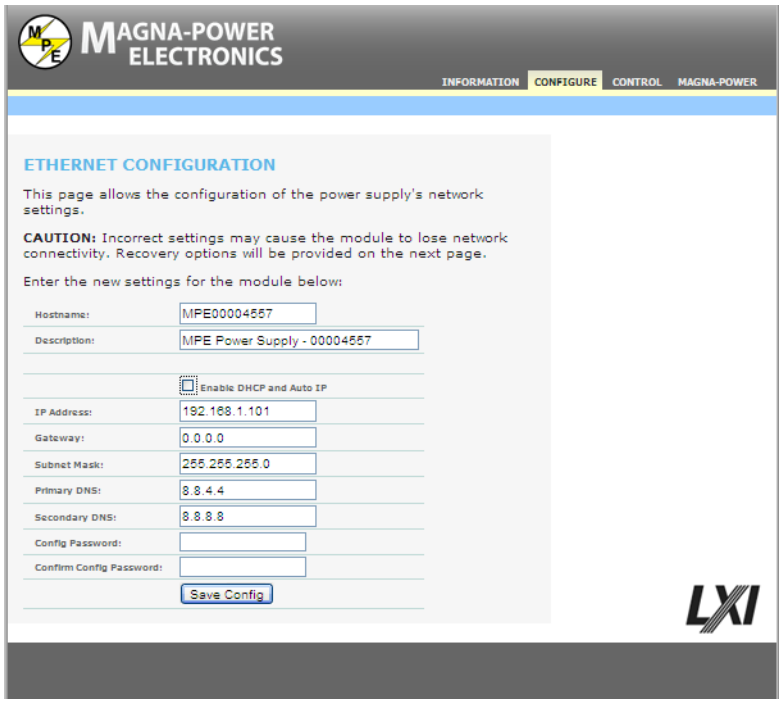


Figura 40. Imagen panel de Configuración del instrumento

6. Configurar las otras dos fuentes (magna power) de la misma manera.

4.3.2 Configuración Router Allied Telesis AR410

El router utilizado para la configuración es Allied Telesis AR410, se conecta al pc a través del puerto COM (RS232) y el programa de emulación es el Hyperterminal de Windows.

1. Ajustar los parámetros de configuración en el hyperterminal de la siguiente manera:

   - Baud rate: 9600
   - Data bits: 8
   - Parity: None
   - Stop Bits: 1

2. Presionar Enter y automáticamente en la pantalla aparecerá el usuario por default del router (Manager) y la contraseña (friend)
3. Digitar Show ip interface e inmediatamente aparecerá la dirección IP por default que tiene configurado el router (**192.168.1.2**).

4. Para la conexión via LAN, utilizar un cable Ethernet de tal manera que se pueda conectar el router a los puertos switch de los instrumentos de la red LAN. Así como se observa en la imagen

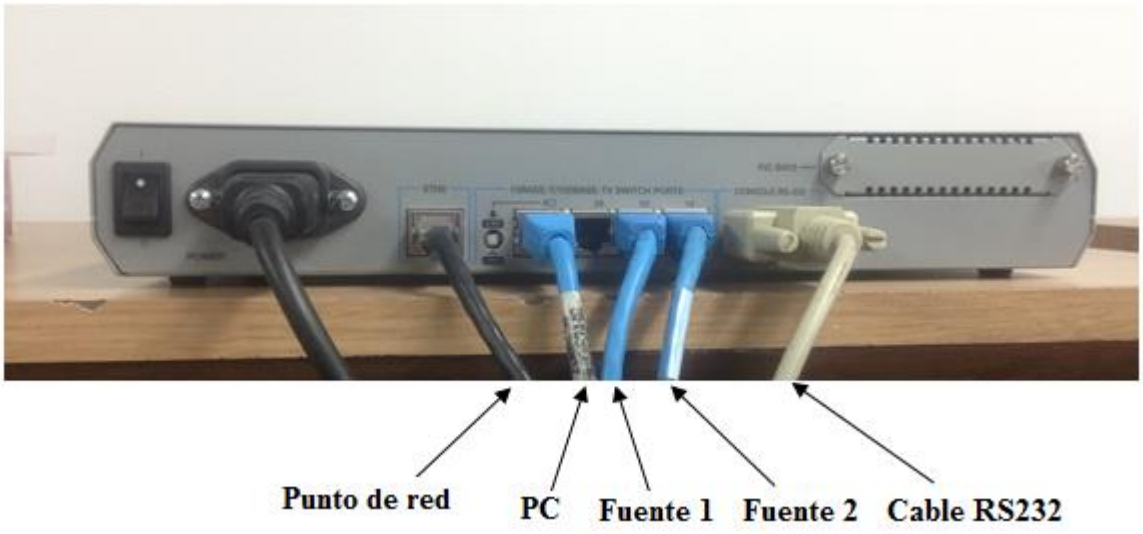


Figura 41. Imagen Conexión del router

4.3.3 Configuración PC

5. Una vez se ha configurado los instrumentos LXI se procede a configurar la dirección IP del PC.
6. En Conexiones de red (Conexión de área local 2), Click derecho en propiedades.
7. Seleccionar internet protocol (TCP/IP), click en propriedades
8. Seleccione usar la siguiente dirección IP y diligenciar dicho campo con lo siguiente:

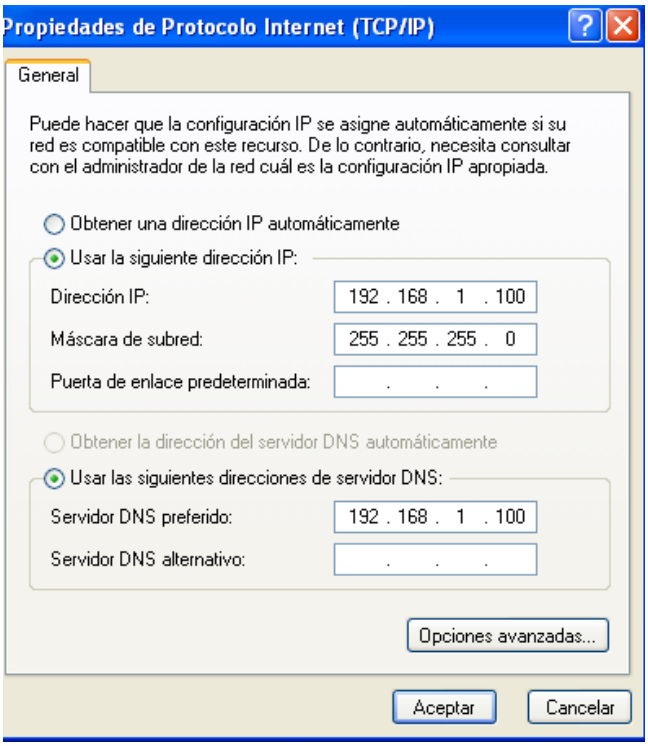


Figura 42. Imagen propiedades de protocolo internet TCP/IP

Finalmente, para comprobar que las fuentes, el PC y el *router* están interconectadas entre si, nuevamente se utiliza el software Eureka y se observa los dispositivos existentes en la red:

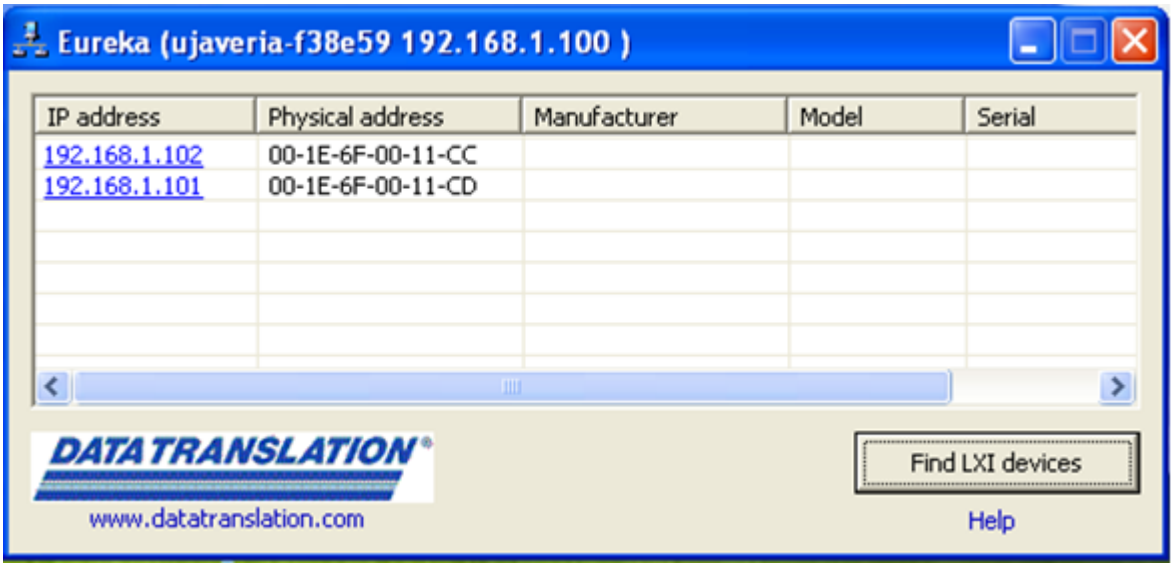


Figura 43. Imagen software Eureka

4.4 Implementación y configuración perfil fotovoltaico PPPE

Para comenzar a utilizar el software se realizaron los siguientes pasos:

1. **Photovoltaic power profile emulation** cuenta con diferentes tipos de comunicación entre las fuentes como son:
   - RS232 (Serial)
   - TCP/IP (Ethernet)
   - GPIB

   Para efectos del presente proyecto, se empleará la comunicación vía TCP/IP debido a que el estándar LXI define dispositivos que usan un estándar abierto LAN (Red de área local) (Ethernet) para sistemas de comunicación entre dispositivos.

2. Seleccionar *Tools*, luego *setup communication* y observará lo siguiente:

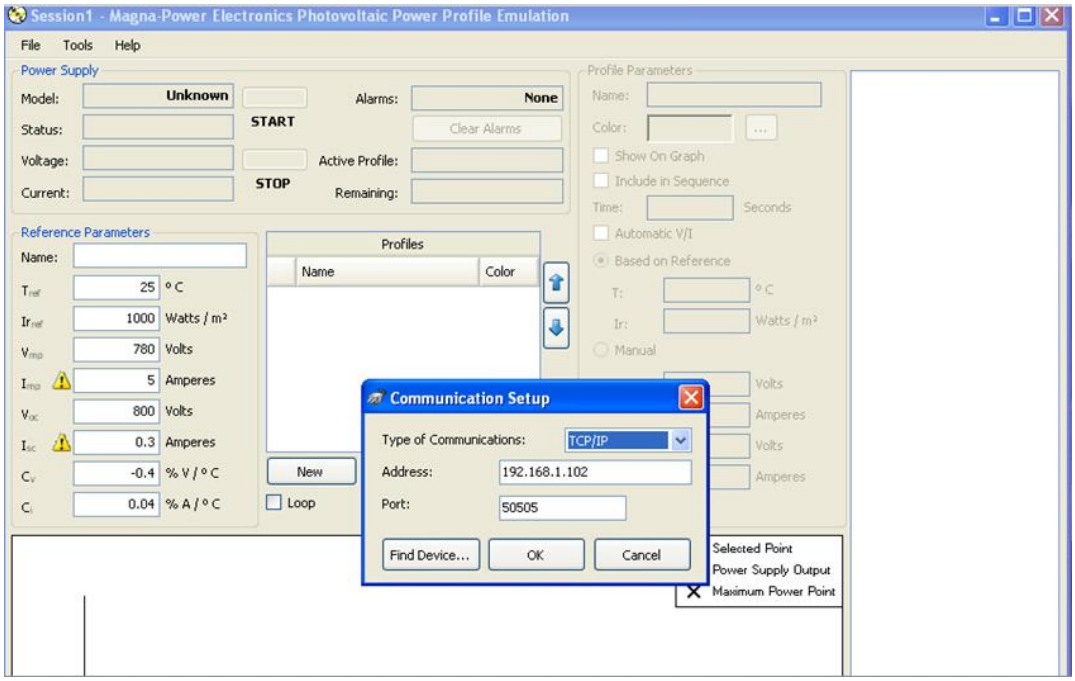


Figura 44.Imagen Menú Photovoltaic power profile emulation

3. Los instrumentos se pueden localizar en la red presionando el botón Find device, automáticamente aparecerá una lista con las fuentes disponibles dentro de la red LAN existente, como se observa en la imagen.

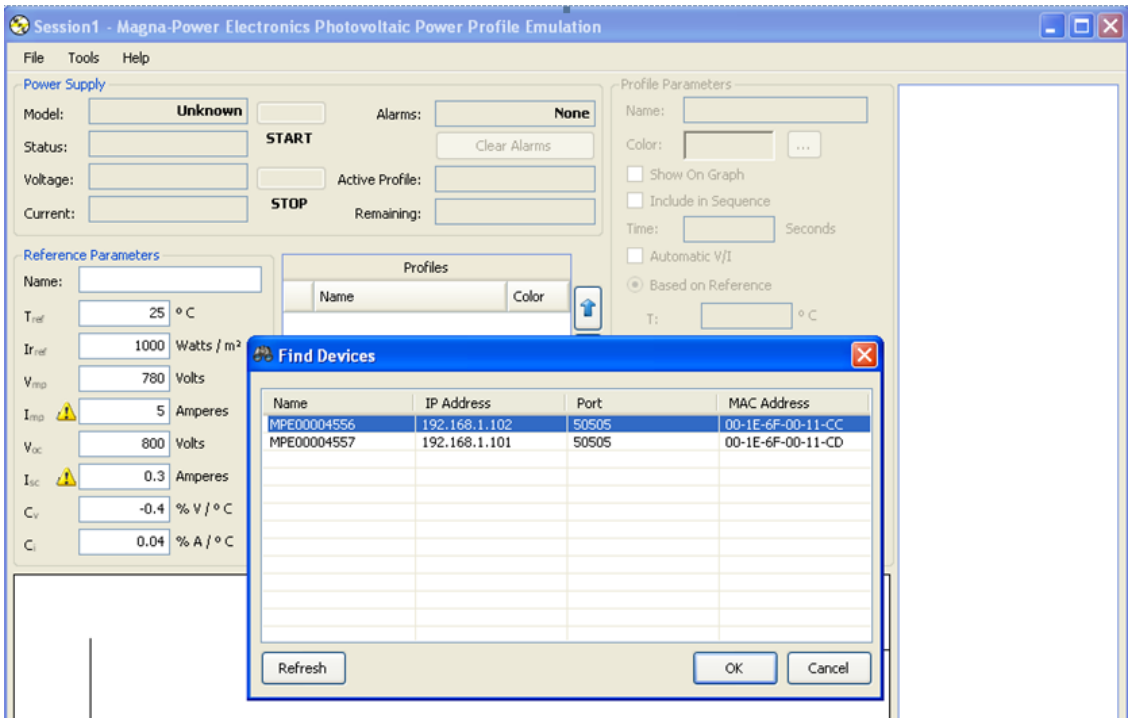


Figura 45.Imagen Listado de Fuentes en la red LAN

4. Generar un nuevo profile y diligenciar los parámetros que se solicitan tales como: T_{ref}, Ir_{ref}, V_{mp}, I_{mp}, V_{oc}, I_{sc}.
5. En seguida se activa el profile y Finalmente, se presiona el botón Start para que se generen los datos en la tabla de la derecha como se observa en la imagen.

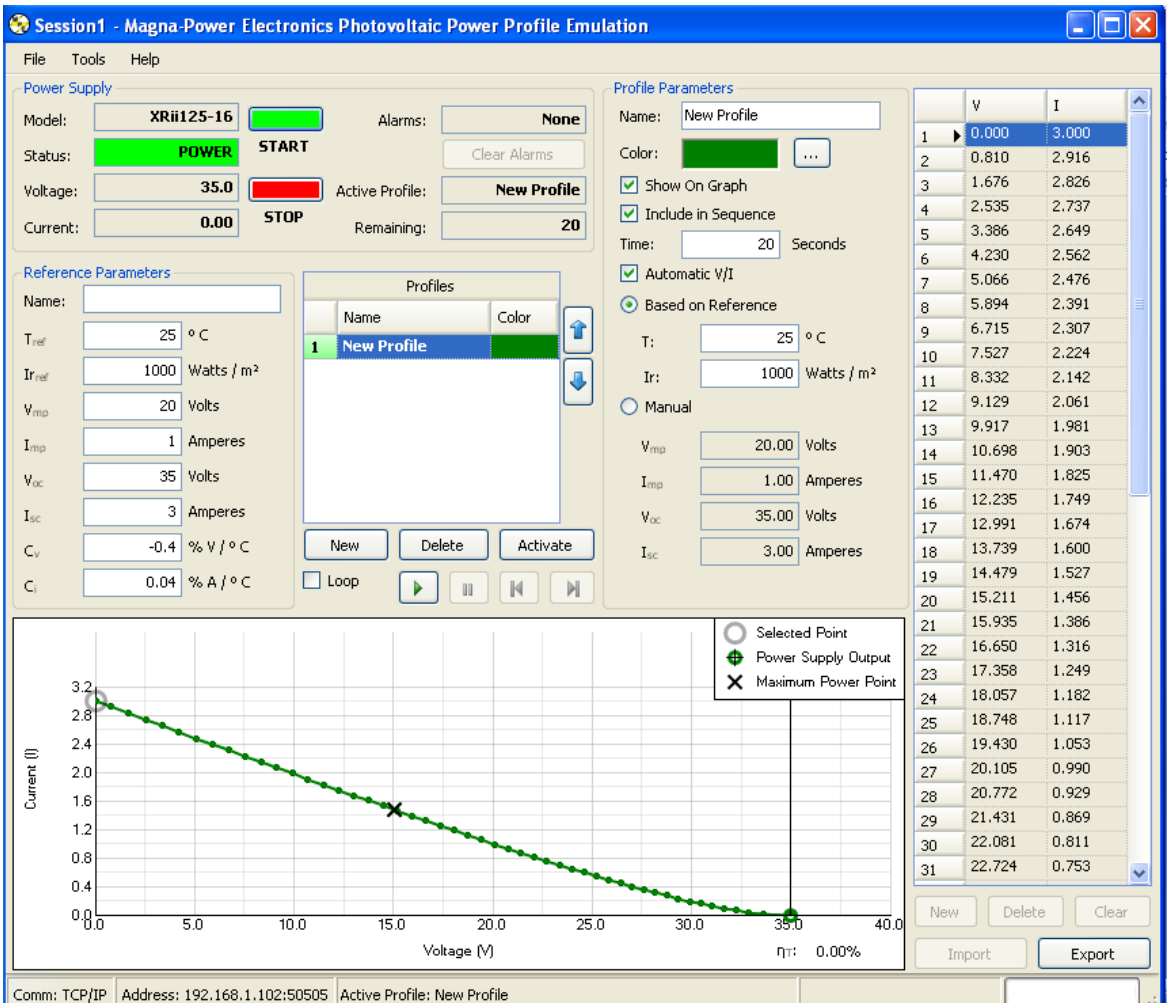


Figura 46. Imagen del software Photovoltaic power profile emulation

6. Al presionar el botón Export, es posible descargar los datos de la tabla voltaje vs corriente en un archivo con formato .csv, .ppe, o .txt.

Es posible transmitir en tiempo real los valores de la gráfica, colocando una carga variable, de tal manera que se pueda observar como cambian los datos de voltaje y corriente, de la siguiente manera:

7. En tools, options, la función data logging permite almacenar el perfil de datos en un archivo .csv en función del tiempo. Cada vez que la carga cambia se ubica en un nuevo punto de la gráfica cambiando los datos de voltaje y corriente.
8. En la opción filename se escribe el nombre del archivo, debajo se escoge el intervalo del tiempo con el cual se quiere obtener la información y finalmente las variables que se desean obtener. En la siguiente imagen se observan los campos mencionados:

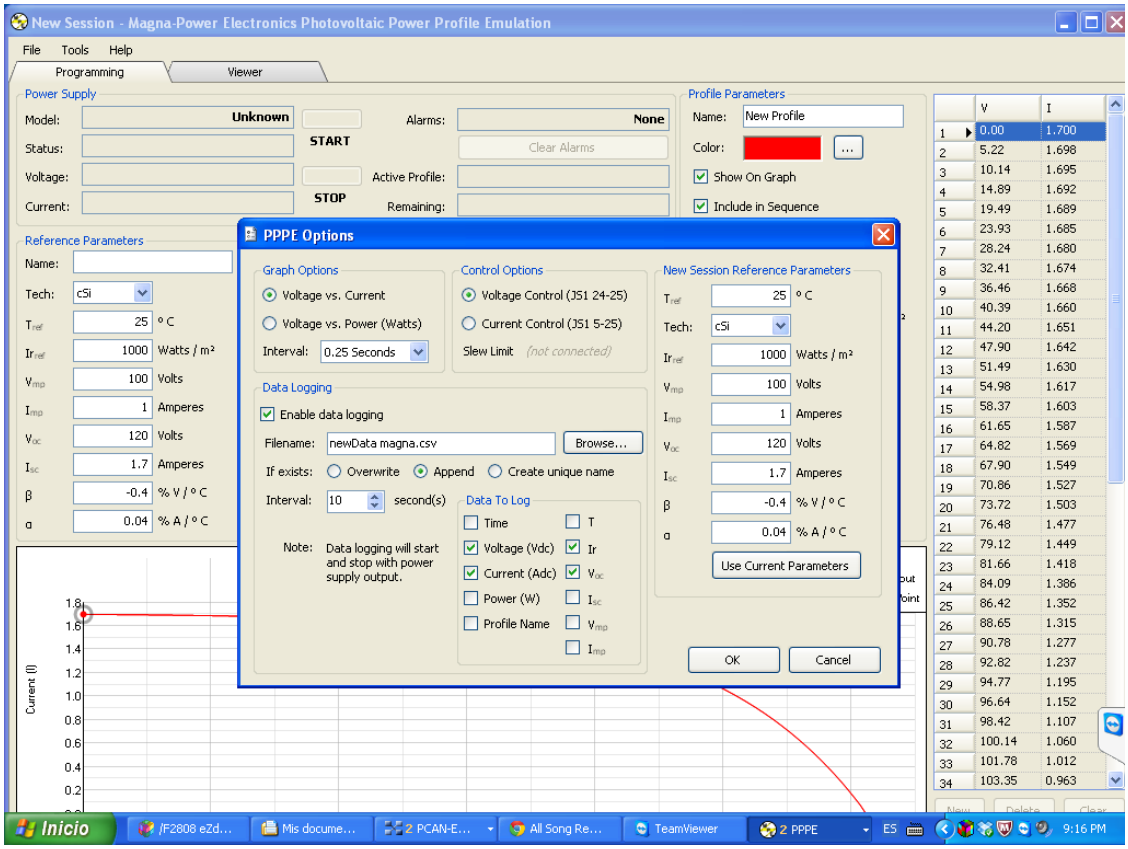


Figura 47. Imagen options software Photovoltaic power profile emulation

Es posible enviar los datos del software PPPE transformándolos en un archivo .XMT, el software PCAN- View permite cargar un archivo de dicho formato y enviar más de un dato a la vez. Si se desea transmitir únicamente un solo dato, es posible enviarlo siguiente las indicaciones en el numeral 4.2.2., esta alternativa no se contempla en tiempo real.

5. ANALISIS DE RESULTADOS

Teniendo en cuenta que el objetivo principal del proyecto es el diseño y la implementación de una plataforma de software que funciona a partir de protocolos de comunicación diferentes (CAN- LXI), es preciso desarrollar la siguiente estructura lógica que permitirá verificar y comprobar el funcionamiento y la interoperabilidad de dicha plataforma: *i)* en primer lugar, se presentarán los resultados de los desarrollos en el módulo CAN, *ii)* luego se procederá a informar sobre los resultados de comunicación de las fuentes Magna Power bajo el protocolo de comunicación LXI, para finalmente, *iii)* indicar los resultados obtenidos de la interoperabilidad de los dispositivos.

5.1 Comunicación del módulo CAN

A partir de la configuración explicada en el acápite de desarrollos la cual puede ser visualizada en las imágenes del capítulo aludido, tanto a la tarjeta ezdspF2808 como al Adaptador CAN USB, fue posible verificar la existencia de una comunicación efectiva (transmisión y recepción de datos) entre el controlador Texas TMS320F2808 y el computador, la cual pudo visualizarse utilizando el software PCAN-VIEW del dispositivo PCAN-USB ADAPTER que es la interfaz de comunicación

Por otro parte, se comprobó que la velocidad de transmisión de datos empleada por el software PCAN-VIEW es 1 Mbit/s (Ver figura 46), la cual corresponde al valor predeterminado en el archivo Examples.c en el Code Composer Studio. No obstante, es posible que el usuario ajuste o configure la velocidad de transmisión según las necesidades propias del proyecto a desarrollar.

Figura 48. Selección de parámetros CAN específicos (hardware) **[33]**

Adicionalmente, se comprueba que el dispositivo USB-CAN ADAPTER es la interfaz de comunicación compatible con el bus CAN que permite visualizar exitosamente la recepción de la trama de datos, así como, transmitir el mensaje con un ID único, campo de datos deseado y formato del mensaje (Estándar o Extendido). Las pruebas de envío y recepción de datos a través del módulo CAN se realizaron empleando la opción Cycle Time del software PCAN-VIEW, se verifica la transmisión programada y secuencial de datos dirigiéndose a la pestaña trace del software

En este punto, entonces, fue posible verificar la eficiencia de la capa física planeada y desarrollada para la conexión del módulo CAN de las tarjetas ezdspF2808 al PC Maestro empleando el dispositivo USB-CAN ADAPTER. Se tuvieron en cuenta los manuales y las normas de capa física del protocolo, manteniendo la integridad de la comunicación.

De igual forma, se evidenció que la configuración desarrollada en el proyecto generó los resultados esperados en términos de comunicación del dispositivo con el PC maestro, de tal forma que los datos o información que se introducía en el software del dispositivo USB-CAN ADAPTER fuera transmitida de forma integral, a la velocidad configurada por el usuario.

5.2 Comunicación de las Fuentes Magna Power

Se verificó la existencia y funcionalidad del servidor web integrado incluido en el estándar LXI de las fuentes Magna Power, que permite configurar características tales como la IP, la dirección MAC, la máscara de red, entre otros.

A su turno, la aludida configuración permitió comprobar que las fuentes Magna Power, tienen la posibilidad de comunicarse con otros instrumentos a través de una red local LAN. De acuerdo con lo anterior, en el presente proyecto fue instalada y configurada una red utilizando un router Allied Telesis AR410, que permitió conectar simultáneamente las fuentes.

Adicionalmente, se evidenció la necesidad de visualizar la dirección IP de cada una de las fuentes existentes en la red, a efectos de configurar y manipular la operabilidad de las mismas. Para ello se instaló y utilizó el software *Eureka Browser*, como herramienta de visualización y comprobación de que los instrumentos LXI se encontraban bajo una misma red, así como para la identificación de la dirección IP de cada uno de los instrumentos.

Por otra parte, la generacion del perfil fotovoltáico de energía se realizó a través del software Photovoltaic Power Profile Emulation. La comunicación empleada es vía TCP/IP debido a que el estándar LXI define dispositivos que usan un estándar abierto LAN (Red de área local) (Ethernet) para los sistemas de comunicación entre dispositivos del aludido estandar.

Finalmente, se comprobó que la comunicación del software PPPE con las fuentes Magna Power es exitosa a través del protocolo LXI, debido a que es posible generar un perfil de energía con datos tabulados de voltaje y corriente.

De acuerdo con lo anterior, entonces, fue posible verificar que el diseño (basado en los manuales de los instrumentos) y la implementación de la red local LAN para la interconexión simultánea de las fuentes Magna Power para el proyecto fue eficiente, con lo cual se válida el cumplimiento del objetivo específico relativo a la capa física del estándar LXI.

Igualmente, se demostró que la configuración del servidor web del estándar LXI relacionado con la IP, la dirección MAC y la máscara de red de los dispositivos permitió, de manera permanente, la comunicación (envío y recepción de datos) entre las fuentes y el PC maestro, evidenciando el cumplimiento de los resultados esperados en términos de comunicación.

5.3 Interoperabilidad de los dispositivos con el PC maestro

En primer lugar, se ha de destacar que los resultados obtenidos en este proyecto no sólo se limitaron a la capa física de conexión de los instrumentos (controladores Texas ezdspF2808 y las

Fuentes Magna Power) que emplean protocolos o estándares de comunicación diferentes (módulo CAN y LXI, respectivamente), sino que se evidenció como elemento esencial la prueba de la configuración realizada, con el fin de obtener una comunicación válida entre los dispositivos y el PC maestro.

La interoperabilidad en tiempo real de los dos protocolos de comunicación se logró mediante los siguientes mecanismos:

- Se implementó un programa en MATLAB que importa los datos .csv del perfil fotovoltaico y los transforma en un archivo .xls empleando la estructura definida, en este caso, relativa a los campos necesarios para el cargue de información a la tarjeta. Posteriormente, dicho archivo podrá ser enviado automáticamente al bus CAN mediante el software PCAN EXPLORER, el cual se encuentra integrado con una interfaz de programación con Visual Basic la cual permite recibir y transmitir datos en el bus CAN.
- Es importante resaltar que el software PCAN-Explorer permitió programar el envío de datos automático, contrariamente al software PCAN-View ya que este únicamente permite enviar archivos .XMT pero no de manera automática es decir el usuario debe cargar el archivo y enviarlo por el Bus CAN

A través de los mecanismos anteriores, se evidenció el cumplimiento del objetivo relacionado con la interoperabilidad de los dispositivos que emplean diferentes protocolos de comunicación, debido a que se logró la existencia de un intercambio de datos directo de voltaje y corriente en formato hexadecimal desde las fuentes Magna Power hacia el dispositivo del bus CAN.

6. CONCLUSIONES

En primer lugar, se destaca que le presente proyecto cumplió con el objetivo general planteado, de tal forma que se lograron verificar escenarios de comunicación e interoperabilidad de los instrumentos, para lo cual fue necesario la adecuada planeación e implementación de la capa física, así como, la eficaz configuración de los dispositivos.

Desde la perspectiva de las tarjetas Texas Ezdsp, es preciso reconocer varios aspectos. Primero, que inicialmente se planeó la utilización de cinco (5) controladores Texas TMS320F28335, suministrados por la Universidad. No obstante, durante el desarrollo del trabajo de grado, se evidenció el daño de la totalidad de las tarjetas mencionadas, por cuanto no se observó ninguna señal de recepción o de transmisión, a través de la consola de ejemplos (archivos.C) que ofrece Texas Instruments.

Por tal razón fue necesario buscar una tarjeta DSP, de la misma familia de la C2000 iniciales, que incluyera un módulo CAN, teniendo en cuenta que el funcionamiento de dicho protocolo es igual en todos los instrumentos de esta familia. La tarjeta seleccionada fue la TMS320F2808, la cual fue incorporada al diseño y desarrollo del bus CAN explicado con anterioridad.

Desde la configuración del módulo CAN, fue posible apreciar la importancia y el comportamiento de cada una de las variables a tener en cuenta en la construcción del código en el software Code Composer Studio. En este punto, se observó que es esencial la determinación precisa de elementos, tales como: i) los parámetros del reloj, ii) la velocidad de transmisión y, iii) la habilitación de los mailbox para la transmisión y recepción de datos en el módulo CAN.

En el proyecto se apreció que el USB CAN ADAPTER es la interfaz de comunicación que permite visualizar de manera efectiva la transmisión y recepción de datos del bus CAN, es por esto que la elección del dispositivo a adquirir es clave ya que no todos los dispositivos son compatibles con el bus. Para este caso inicialmente se adquirió el dispositivo CAN USB Light, que se comunica con el PC a través del puerto RS232, que funciona mediante el software Tera Term (hyperterminal) como interfaz de visualización de datos con el usuario. Una vez se instaló el software, se analizó y evaluó con ayuda del proveedor Grid Connect el correcto funcionamiento del dispositivo, se concluyó que el dispositivo NO es compatible ni consistente con el bus CAN que se deseaba implementar debido a que existía comunicación directa con la tarjeta Ezdsp, es decir, no fue posible en ningún momento observar los datos de transmisión y recepción en el bus CAN. Por esta razón se optó por buscar otro dispositivo USB CAN adapter que utilizará un canal de comunicación con el PC que no fuera por puerto serial, razón por la cual se adquirió el dispositivo PCAN-USB ADAPTER del proveedor PEAK SYSTEM el cual se comunica con el computador a través del puerto USB, con él se verificó una transmisión eficaz de datos.

Teniendo en cuenta que el presente proyecto planteó como requisito la transmisión en tiempo real de los datos entre los dispositivos y el PC, además de garantizar la delimitación de los tiempos de transmisión de los mensajes, fue necesaria la estructuración de un sistema en el cual las mismas entradas produzcan invariablemente las mismas salidas (determinístico), de tal forma que se cree un sistema de gestión que permitan disminuir la incertidumbre en el manejo de datos.

Con fundamento en lo anterior, el protocolo de comunicación CAN, de características determinísticas, controla la transmisión de mensajes basada en prioridades, esto indica que cuenta con un mecanismo de arbitraje para el envío de datos. Adicionalmente, el Bus CAN facilita la implementación de la capa física del proyecto en razón a que disminuye el cableado dentro del sistema. De igual forma, se destaca que este protocolo de comunicaciones permite la transmisión y recepción de datos a alta velocidad (1 Mbps). Por consiguiente, se deduce que el protocolo CAN se ajusta a la totalidad de requerimientos del proyecto.

Ahora, desde el protocolo de comunicaciones LXI (LAN eXtensions for Instruments) se apreció que el elemento característico de este estándar es la posibilidad de comunicar varios dispositivos mediante una red LAN. Con base en este concepto, se decidió crear una red local utilizando un *router*. Ahora bien, las fuentes Magna Power son instrumentos clase C de LXI, esto es, que sus características específicas les permiten conectarse en red y trabajar paralelamente con otros dispositivos, aspecto que permitió el efectivo funcionamiento de la red implementada.

En este punto del proyecto se apreció la utilidad del software Eureka, desde dos perspectivas: i) permitió la verificación del estado de la red, es decir, ayudó a la validación de la correcta conexión de las fuentes a la red local y, ii) en la identificación de la dirección IP de cada uno de los instrumentos conectados a la red LAN.

De lo anterior, se aprecia el cumplimiento de los objetivos específicos relativos a conocer tanto los requisitos de la implementación física, como los requerimientos para la adecuada configuración de los dispositivos, teniendo en cuenta que funcionan con protocolos de comunicación diferentes.

Finalmente, la interoperabilidad se logró apreciar en dos (2) modalidades o escenarios a saber:

i) Empleando la posibilidad que ofrece el software PPPE (Photovoltaic Power Profile Emulation) de exportar datos en un formato .CSV, el cual es importado y modificado a formato .XLS, mediante el software matlab. Una vez se ha generado el archivo con la estructura deseada, se exporta al software PCAN Explorer el cual se encuentra integrado con VisualBasic. Esta modalidad permite la comunicación masiva de datos de un dispositivo a otro.

ii) También es posible que la comunicación se realice de manera manual, esto es, que el operador ingrese datos individuales en el software PCAN View tomados del software PPPE.

De lo anterior se deduce el logro o cumplimiento de la totalidad de los objetivos específicos planteados en el anteproyecto del presente trabajo de grado, esto es:

- Se identificaron los parámetros necesarios para las capas físicas de los protocolos de comunicación CAN y LXI.
- Se realizaron las configuraciones necesarias para el adecuado y eficiente funcionamiento De los mismos.
- Se implementaron escenarios de interoperabilidad entre los dispositivos base del proyecto (tarjeta TMS320F2808 y fuentes Magna Power).

Ahora bien, se destaca que el cronograma del proyecto se vió afectado o alterado principalmente por los siguientes aspectos: i) el amplio tiempo de espera para el arribo del dispositivo USB CAN ADAPTER de Grid Connect, ii) el daño de la totalidad de las tarjetas TMS320F28335 que obligaron, como se menciono con anterioridad, a buscar una tarjeta DSP de la misma familia C2000 para continuar con el proyecto, iii) la verificación de NO compatibilidad del USB CAN ADAPTER con la tarjeta Ezdsp y, iv) el tiempo de espera para el arribo del PCAN-USB ADAPTER.

7. BIBLIOGRAFÍA

[1] Instrel, 2011. [En línea]. Available: http://www.magna-power.com.

[2] Texas Instruments. [En línea]. Available: http://www.ti.com/tool/tmdsez28335.

[3] F. Bormann, F2833x - Tutorial, Communication III: Controller Area Network (CAN), Houston, Texas: Connexions, 2011.

[4] S. Corrigan, Texas Instruments;, July 2008. [En línea]. Available: http://www.ti.com/lit/an/sloa101a/sloa101a.pdf.

[5] J. C. Cuenca, «Diseño e implementación de protocolo CAN para el control de un modulo de red de sensores,» Quito, 2010.

[6] H. K. C. [En línea]. Available: http://cabierta.uchile.cl/revista/25/articulos/pdf/paper1.pdf.

[7] Texas Instruments, «TMS320F28335, TMS320F28334, TMS320F28332,TMS320F28235, TMS320F28234, TMS320F28232 Digital Signal Controllers (DSCs). Data Manual,» 2011.

[8] T. Instruments, «3.3-V CAN TRANSCEIVERS,» 2001. [En línea]. Available: http://www.ti.com/lit/ds/symlink/sn65hvd230.pdf.

[9] BOSCH, 1991. [En línea]. Available: http://www.gaw.ru/data/Interface/CAN_BUS.PDF.

[10] M. ROUAUX, DISEÑO Y PROTOTIPO DE MIDDLEWARE BASADO EN CORBA PARA EL BUS CAN, Buenos Aires: Universidad de Buenos Aires, Ingeniería informática, 2005.

[11] J. Q. Castellano, Bus CAN: estado de buses industriales y aplicaciones, 2000.

[12] A. Creque, R. Reddy y P. Franlklin, «Exploring LXI's advanced capabilities,» de *Autotestcon, IEEE*, 2005.

[13] LAN eXtensions for Instrumentation, October 2008. [En línea]. Available: http://www.lxistandard.org/Documents/about/LXI%201_3%202008-10-30.pdf.

[14] D. Pleasant, «LAN-based measurement triggering using LXI instrumentation,» de *Autotestcon, IEEE*, 2005.

[15] N. Instruments. [En línea]. Available: http://cnx.org/content/m12283/latest/.

[16] J. García-Zubia , U. Hernández, I. Angulo, P. Orduña y J. Irurzun, «Remote Laboratories Based on LXI,» *iJOE*, vol. 4, nº 3, pp. 25 - 27, August 2008 [Online] https://www.weblab.deusto.es/web/images/publications/ijoe2008_num_3.pdf.

[17] C. Proft, Escritor, *The power of LXI Instrumentation – Use Cases and Performance.* [Performance]. Conference: Autotestcon IEEE, 2007.

[18] G. Drenkow, «Triggering Differences Between GPIB and LXI,» de *Autotestcon, IEEE*, 2006.

[19] J. J. González de la Rosa, «Instrumentos Electrónicos Programables (GPIB; norma IEEE-488.2),» 2003. [En línea]. Available: http://www2.uca.es/grup-invest/instrument_electro/ppjjgdr/Electronics_Instrum/Electronics_Instrum_Files/temas/T7_Instrum_Prog.PDF.

[20] Magna Power Inc., December 2010. [En línea]. Available: http://www.google.com.co/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&ved=0CCQQFjAA&url=http%3A%2F%2Fwww.tmworks.co.kr%2Fmodule%2Fboard%2Fdownload.php%3Fboardid%3Dtech1%26b_idx%3D1%26idx%3D2&ei=xzWEUKKqOY6k8QTxk4HYDQ&usg=AFQjCNFi8YcOz0ht5VL61-2V7uusCC.

[21] L. Guohan, Y. Xiaofei y L. y. Huang Feng, «Development and design of a virtual instrument based on LXI technology,» de *Mechanic Automation and Control Engineering (MACE)*, 2010.

[22] Magna Power Inc., 2012. [En línea]. Available: http://www.magna-power.com/products/programmable-dc-power-supplies/xr-series.

[23] P. G. Schreier, July 2008. [En línea]. Available: http://www.lxistandard.org/Documents/Articles/0708_programming.pdf.

[24] FreeBSD, 2012. [En línea]. Available: http://www.freebsd.org/doc/es/books/handbook/network-dhcp.html.

[25] LXI Consortium Standards Board, August 2006. [En línea]. Available: http://www.lxistandard.org/Documents/About/LXI_Trigger_Bus_Cable_Terminator_Spec_1.1_Final.pdf.

[26] Texas Instruments, 2009. [En línea]. Available: http://www.ti.com/lit/ug/sprueu1/sprueu1.pdf.

[27] T. Instruments, «TMS320F2833x, 2823x Enhanced Controller Area Network (eCAN),» 2009. [En línea]. Available: http://www.ti.com/lit/ug/sprueu1/sprueu1.pdf.

[28] Texas Instruments, October 2012. [En línea]. Available: http://www.ti.com/lit/ug/spru712h/spru712h.pdf.

[29] P. System, USB to CAN Interface, User Manual V2.1.3, Alemania: PEAK-System Technik GmbH, 2012.

[30] N. Forcier, «Tips for Remote Testing Using LXI Devices and WiFi,» [En línea]. Available: http://www.lxistandard.org/Documents/Papers/LXI_2007_Getting_Started_Guide.pdf.

[31] L. G. Y. X. L. D. o. E. a. I. E. Huang Feng, Development and Design of a Virtual Instrument Based on LXI Technology, Xiang tan, China, 2010.

[32] LXIstandard, «Guide to Configuring and LXI Instrument,» [En línea]. Available: http://www.lxistandard.org/Documents/Papers/LXI_2007_Getting_Started_Guide.pdf.

[33] K. G. Fertitta y P. Mindworks. [En línea]. Available: http://www.ivifoundation.org/docs/Understanding%20the%20Benefits%20of%20IVI[3].pdf.

[34] Texas Instruments, 2011. [En línea]. Available: http://coecsl.ece.illinois.edu/ge423/datasheets/F28335Ref_Guides/F28335DataSheet.pdf.

[35] K. Shah, August 1999. [En línea]. Available: http://it360.tw/document/an/pdf/AN140.pdf.

[36] J. Pasquarette, 1998. [En línea]. Available: http://www.ni.com/ivi/interchangeability.pdf.

[37] T. Instruments, «TMS320F2808 Digital Signal Processors,» 2003. [En línea]. Available: http://www.ti.com/lit/ds/symlink/tms320f2808.pdf.

[38] H. N. Alepuz Soriano, «Bus CAN,» [En línea]. Available: http://server-die.alc.upv.es/asignaturas/PAEEES/2005-06/A03-A04%20-%20Bus%20CAN.pdf.

8. ANEXOS

```
//Titulo: CAN - Transmisión vía F2808controlCARD
//                and SH65HVD230 at Peripheral Explorer Board
////              Extended CAN-Frame transmit at 1Mbps
////              Objetivo:
//                Transmitir un mensaje de  8 bytes
//                repetir la transmission  1.0 segundos de intervalo
//                Mailbox 5  transmite
//                Identificador : 0x1000 0000
//                Data Length Code DLC =  8
////              physical eCAN is at GPIO31 (CANTXA) and GPIO30 (CANRXA)
////              Frequencia de oscillación @F2808controlCARD: 30MHz
//                PLLCR = 10   :  multiply by 5
//                SYSCLKOUT = 150MHz , 2808-CAN-CLKIN = 75MHz

#include "DSP280x_Device.h"     // DSP280x Headerfile Include File
#include "DSP280x_Examples.h"   // DSP280x Examples Include File

// Prototipos de funciones externas
extern void InitSysCtrl(void);
extern void InitPieCtrl(void);
extern void InitPieVectTable(void);
extern void InitCpuTimers(void);
extern void InitECan(void);
```

```
extern void ConfigCpuTimer(struct CPUTIMER_VARS *, float, float);
// Prototype statements for functions found within this file.
void Gpio_select(void);
interrupt void cpu_timer0_isr(void);

//###########################################################################
//                                        main code
//###########################################################################

void main(void)
{

   int counter=0;    // binary counter for digital output
   Uint16 temp;
   struct    ECAN_REGS ECanaShadow;       // local copy of CANA registers

   InitSysCtrl();    // Basic Core Initialization
                // SYSCLK=150MHz, HISPCLK=75MHz, LSPCLK=37.5MHz

   EALLOW;
     SysCtrlRegs.WDCR= 0x00AF;    // Re-enable the watchdog
     EDIS;          // 0x00AF  to NOT disable the Watchdog, Prescaler = 64

   DINT;              // Disable all interrupts
```

```
Gpio_select();    // GPIO9, GPIO11, GPIO34 and GPIO49 as output
                  // to 4 LEDs at Peripheral Explorer

/* Initialize the CAN module */
// NOTE: first modify TI-file: InitECan() to 100kbps by setting BTR = 49
InitECan();

/* Write to Mailbox 1 message ID field        */
ECanaMboxes.MBOX1.MSGID.all  = 0x069F6BC7;  // extended identifier
   ECanaMboxes.MBOX1.MSGID.bit.IDE = 1;

 // * Write to Mailbox 0 message ID field        */

    ECanaMboxes.MBOX0.MSGID.all  = 0x0001B207;  // extended identifier
   ECanaMboxes.MBOX0.MSGID.bit.IDE = 1;

/* Configure Mailbox 1 as Receiver mailbox */
ECanaShadow.CANMD.all = ECanaRegs.CANMD.all;
ECanaShadow.CANMD.bit.MD1 = 1;
ECanaRegs.CANMD.all = ECanaShadow.CANMD.all;

        /* Configure Mailbox 0 as Receiver mailbox */
ECanaShadow.CANMD.all = ECanaRegs.CANMD.all;
ECanaShadow.CANMD.bit.MD0 = 1;
ECanaRegs.CANMD.all = ECanaShadow.CANMD.all;
```

```
/* Enable Mailbox 1                    */
ECanaShadow.CANME.all = ECanaRegs.CANME.all;
ECanaShadow.CANME.bit.ME1 = 1;
ECanaRegs.CANME.all = ECanaShadow.CANME.all;

      /* Enable Mailbox 0                    */
ECanaShadow.CANME.all = ECanaRegs.CANME.all;
ECanaShadow.CANME.bit.ME0 = 1;
ECanaRegs.CANME.all = ECanaShadow.CANME.all;

/* Write to Mailbox 5 message ID field  */
ECanaMboxes.MBOX5.MSGID.all     = 0x11100000;    // message identifier
ECanaMboxes.MBOX5.MSGID.bit.IDE = 1;  // Extended Identifier

/* Configure Mailbox 5 as transmit mailbox */
ECanaShadow.CANMD.all = ECanaRegs.CANMD.all;
ECanaShadow.CANMD.bit.MD5 = 0;
ECanaRegs.CANMD.all = ECanaShadow.CANMD.all;

/* Enable Mailbox 5  */
ECanaShadow.CANME.all = ECanaRegs.CANME.all;
ECanaShadow.CANME.bit.ME5 = 1;
ECanaRegs.CANME.all = ECanaShadow.CANME.all;

/* Write to DLC field in Master Control reg */
ECanaMboxes.MBOX5.MSGCTRL.all = 0;
```

```
ECanaMboxes.MBOX5.MSGCTRL.bit.DLC = 4;

InitPieCtrl();        // basic setup of PIE table; from DSP2833x_PieCtrl.c

InitPieVectTable();    // default ISR's in PIE

EALLOW;
PieVectTable.TINT0 = &cpu_timer0_isr;
EDIS;

InitCpuTimers();    // basic setup CPU Timer0, 1 and 2

ConfigCpuTimer(&CpuTimer0,150,100000);

PieCtrlRegs.PIEIER1.bit.INTx7 = 1;

IER |=1;

EINT;
ERTM;

CpuTimer0Regs.TCR.bit.TSS = 0;    // start timer0

while(1)
{
        while(CpuTimer0.InterruptCount < 10)    // wait for 10*100 milliseconds
```

```
{
   EALLOW;
   SysCtrlRegs.WDKEY = 0xAA;    // service WD #2
   EDIS;
}
CpuTimer0.InterruptCount = 0;

ECanaMboxes.MBOX5.MDL.byte.BYTE0 = counter & 0x00FF ;
 ECanaShadow.CANTRS.all = 0;
 ECanaShadow.CANTRS.bit.TRS5 = 1;     // Set TRS for mailbox under test
 ECanaRegs.CANTRS.all = ECanaShadow.CANTRS.all;

while(ECanaRegs.CANTA.bit.TA5 == 0 ) // Wait for TA5 bit to be set.
 {
   EALLOW;
   SysCtrlRegs.WDKEY = 0xAA;               // Service watchdog #2
   EDIS;
 }

 ECanaShadow.CANTA.all = 0;
 ECanaShadow.CANTA.bit.TA5 = 1;         // Clear Transmit Acknowledge #5
 ECanaRegs.CANTA.all = ECanaShadow.CANTA.all;

counter++;
GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1;        // toggle red LED LD3 @ 28335CC
```

```
while(CpuTimer0.InterruptCount == 0);
CpuTimer0.InterruptCount = 0;

EALLOW;
SysCtrlRegs.WDKEY = 0x55;    // service WD #1
EDIS;

if(ECanaRegs.CANRMP.bit.RMP1 == 1 )     // valid new data in MBX1?
{
    temp = ECanaMboxes.MBOX1.MDL.byte.BYTE0;    // read message
   ECanaRegs.CANRMP.bit.RMP1 = 1;           // clear the status flag RMP1
                                // and prepare MBX1 for next receive
   if(temp & 1)                          // update LED LD1
      GpioDataRegs.GPASET.bit.GPIO9 = 1;
   else
      GpioDataRegs.GPACLEAR.bit.GPIO9 = 1;
   if(temp & 2)                         // update LED LD2
      GpioDataRegs.GPASET.bit.GPIO11 = 1;
   else
      GpioDataRegs.GPACLEAR.bit.GPIO11 = 1;
   if(temp & 4)                          // update LED LD3
      GpioDataRegs.GPBSET.bit.GPIO34 = 1;
   else
      GpioDataRegs.GPBCLEAR.bit.GPIO34 = 1;
 //  if(temp & 8)                          // update LED LD4
   //    GpioDataRegs.GPBSET.bit.GPIO49 = 1;
```

```
   //else
     //  GpioDataRegs.GPBCLEAR.bit.GPIO49 = 1;
}


                    if(ECanaRegs.CANRMP.bit.RMP0 == 1 )     // valid new data in MBX0?
{
    temp = ECanaMboxes.MBOX0.MDL.byte.BYTE0;    // read message
   ECanaRegs.CANRMP.bit.RMP0 = 1;           // clear the status flag RMP1
                                  // and prepare MBX1 for next receive
   if(temp & 1)                          // update LED LD1
      GpioDataRegs.GPASET.bit.GPIO9 = 1;
   else
      GpioDataRegs.GPACLEAR.bit.GPIO9 = 1;
   if(temp & 2)                         // update LED LD2
      GpioDataRegs.GPASET.bit.GPIO11 = 1;
   else
      GpioDataRegs.GPACLEAR.bit.GPIO11 = 1;
   if(temp & 4)                          // update LED LD3
      GpioDataRegs.GPBSET.bit.GPIO34 = 1;
   else
      GpioDataRegs.GPBCLEAR.bit.GPIO34 = 1;
  //  if(temp & 8)                          // update LED LD4
   //    GpioDataRegs.GPBSET.bit.GPIO49 = 1;
   //else
     //  GpioDataRegs.GPBCLEAR.bit.GPIO49 = 1;
}
```

```
    }
}

void Gpio_select(void)
{
    EALLOW;
    GpioCtrlRegs.GPAMUX1.all = 0;        // GPIO15 ... GPIO0 = General Puropse I/O
    GpioCtrlRegs.GPAMUX2.all = 0;        // GPIO31 ... GPIO16 = General Purpose I/O

    GpioCtrlRegs.GPAMUX2.bit.GPIO30 = 1;    // CANA_RX
    GpioCtrlRegs.GPAMUX2.bit.GPIO31 = 1;    // CANA_TX

    GpioCtrlRegs.GPBMUX1.all = 0;        // GPIO47 ... GPIO32 = General Purpose I/O

    EDIS;
}

interrupt void cpu_timer0_isr(void)
{
    CpuTimer0.InterruptCount++;
    EALLOW;
    SysCtrlRegs.WDKEY = 0x55;    // service WD #1
    EDIS;
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
}
```

CODE SOFTWARE EN MATLAB:

```
fid = fopen('archivomagna1-1.csv','r');  'Change name file'
   data = textscan(fid, '%u8 %n', 'Delimiter',',','HeaderLines',1); 'Extract file '
   fclose(fid);
   Volts=data{1};  'Store data in variable'
   K=length(Volts);
   Current=data{2};  'Store data in variable'
   VoltsHexa=dec2hex(Volts);  'Convert data store in hexa format'
   CurrentHexa=dec2hex(fix(Current)); 'Convert data store in hexa format'
   VoltsHexa=num2cell(VoltsHexa,2);  'Aprox data in decimal format'
   VoltsHexa = cellfun(@(x) {strcat(x,{char(10)})},VoltsHexa);
   VoltsHexa = cellfun(@(x) {deblank([x{:}])},VoltsHexa);
   CurrentHexa=num2cell(CurrentHexa,2); 'Aprox data in decimal format'
   CurrentHexa = cellfun(@(x) {strcat(x,{char(10)})},CurrentHexa);
   CurrentHexa = cellfun(@(x) {deblank([x{:}])},CurrentHexa);
   X=zeros([K,3]);
   DLC=4*ones(K,1);
   CANID=zeros(K,1);
   headers={'CAN-ID','DLC', 'Data0', 'Data1', 'Data2','Data3'}; 'Labels for new file .xls'
   L=101;
   for i=1:K
       CANID(i,1)=L;
      L=L+1;
   end
   xlswrite('Newprueba.xls',CANID,'hoja1','A2');  'Write data into new xls file'
```

```
Values={101,DLC ,X, 0 ,0, 0};
xlswrite('Newprueba.xls',[headers; Values]);  'Write data into new xls file'
xlswrite('Newprueba.xls',X,'hoja1','C2');  'Write data into new xls file'
xlswrite('Newprueba.xls',DLC,'hoja1','B2'); 'Write data into new xls file'
xlswrite('Newprueba.xls',VoltsHexa(:),'hoja1','F2:F3'); 'Write data into new xls file'
xlswrite('Newprueba.xls',CurrentHexa(:),'hoja1','F4:F5'); 'Write data into new xls file'
```

CODE- SOFTWARE PCAN-EXPLORER

```
Sub SendMessagesFromExcel()
  Dim ExcelApp, ExcelWorkbook, ExcelSheet

  Dim conn , str
  Set ExcelApp = CreateObject("Excel.Application")  ' Start Excel
  ExcelApp.Visible = True ' Make Excel visible, not really required
  Set ExcelWorkbook = ExcelApp.Workbooks.Open("C:\Archivos de
programa\MATLAB\R2012a\bin\Newprueba.xls")    ' Adjust path!
  Set ExcelSheet = ExcelWorkbook.Worksheets(1) ' go to Sheet 1

  ' Read all rows in the sheet
  Dim Row, Msg, n, bin
  Set colMsg = Connections.Transmitmessages'.Add
  Row = 2 ' Messages begin on row 2
  bin="11111"
  While ExcelSheet.Cells(Row, 1) <> Empty
   Set Msg = Connections.Transmitmessages.Add
```

```
 bin=bin+"1"
Msg.ID = bin
Msg.DLC = CInt(ExcelSheet.Cells(Row, 2))
For n = 0 To Msg.DLC-1
  Msg.Data(n) = CInt("&H" & ExcelSheet.Cells(Row, 3+n))
  Bus=1
  msg.CycleTime=500
Next
 Set Msg.Connection = Connections(1)
 Msg.Write 0
Row = Row + 1  ' Go to next row
  Wend
  ExcelApp.Quit
End Sub
```