Aplicación del Protocolo OGC® PUCK para Instrumentación Plug and Play

Luis Vieira Silva
Departamento de Ingeniería Electrónica. EPSEVG

Resumen

En el mercado reducido de los instrumentos de medida no existe una estandarización sobre los protocolos a utilizar para el control y configuración de los instrumentos. Cada fabricante define la sintaxis utilizada y el conjunto de comandos necesarios para cada uno de sus instrumentos. Debido a este particular problema, fue necesario crear un protocolo de comunicación de modo que sea factible el uso de una sintaxis común, y un conjunto de comandos comunes además de comandos específicos según la naturaleza del instrumento. Este protocolo se llama OGC® PUCK y puede ser implementado en cualquier instrumento de medida u otro tipo de aparato. Este proyecto viene a satisfacer una de las exigencias y necesidades del mercado industrial y de investigación en facilitar las comunicaciones entre el ordenador y el instrumento de medida, utilizando una plataforma visual *plug&play* (conectar y funcionar).

1. Introducción

Actualmente existe un problema de interoperabilidad a nivel de comunicación con instrumentos de medida debido a la falta de estándares que faciliten el diseño de sistemas de instrumentación Plug and Play. No existe un protocolo de comunicación común entre los instrumentos de medidas ya existentes, sobre los buses de comunicación serie RS232, RS485, más tradicionales y conocidos en el mundo de las comunicaciones serie.

Una vez instalado y configurado el instrumento a través del "PUCK Mode" podrá proceder a la lectura y escrita de los valores obtenidos de los instrumentos solo cambiando en el programa para "Instrument Mode". A partir de este momento el instrumento está listo para realizar sus funciones de lectura y enviar los datos obtenidos a través de la misma puerta COM o TCPIP.

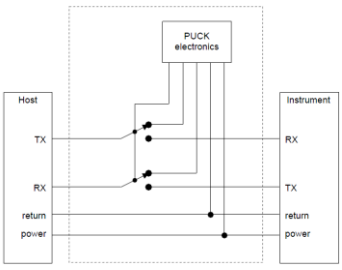

Figura 1. Cambio de PuckMode a Instrument Mode

Pero la configuración del instrumento en el ordenador no es solo una virtud de este proyecto pero también en acceder a sus datos tanto para su procesado, archivo o visualización compartida tanto en el instrumento como el ordenador siendo el principal objetivo la implementación y posterior evaluación del protocolo sobre su usabilidad en instrumentos de medida.

El protocolo OGC® PUCK Standard describe una interfaz de comunicación serie RS232 y Ethernet (TCP-IP) por la cual es posible identificar de manera única un dispositivo de medida y el almacenaje de información respecto al dispositivo dentro del mismo. La implementación de dichas interfaces mediante el lenguaje de programación LabVIEW y el desarrollo de una API que facilite la generación de aplicaciones posteriores. Los objetivos a conseguir son:

- La detección automática del puerto COM y velocidad Baudrate para el caso RS232 y buscar en una red TCPIP el instrumento.
- Detección y alerta de conexión y desconexión de un instrumento PUCK.
- Notificación de la conexión o desconexión del instrumento.

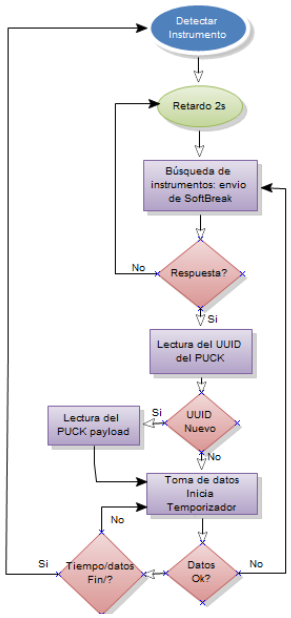

Figura 2. Algoritmo de conexión y desconexión de instrumentos PUCK.

En las figuras 1 tenemos la configuración de hardware y en la figura 2 exponemos el algoritmo que implementa en la detección de conexión y desconexión de un instrumento PUCK. El programa envía periódicamente un PUCK *soft-break* a diferentes velocidades a cada uno de los puertos serie buscando un instrumento con el protocolo PUCK. Si se reconoce, el programa descarga el PUCK "instrument datasheet" y verifica si el instrumento conectado es un nuevo instrumento o no mediante su UUID (modelo del instrumento ID).

Una vez inicializado, el controlador recibirá los datos del instrumento periódicamente. Para identificar la desconexión del instrumento y el algoritmo tiene en cuenta el periodo de lectura de datos del instrumento. Si el periodo de lectura es muy grande, por ejemplo superior a 2 segundos, o más, el programa verificará con una frecuencia mayor que el instrumento sigue conectado mediante un "Softbreak".

2. Protocolo PUCK

PUCK (Programmable Underwater Connector, with Knowledge) es un protocolo de comandos simples que ayuda a automatizar el proceso de configuración y guardar información sobre el instrumento en el propio instrumento. La información almacenada *payload* puede ser relativa al instrumento (metadatos), código ejecutable o driver, o cualquier otra información que sea considerada necesaria para el sistema de observación. Cuando un instrumento PUCK-enable está conectado a un controlador, éste puede recuperar la información desde el instrumento a través del protocolo PUCK.

El controlador podría instalar y ejecutar código relacionado con el instrumento, o incluir un datasheet electrónico (hoja de especificaciones) sobre el instrumento en un formato conocido (por ejemplo TEDS, Transducer Electronics DataSheet, de IEEE Std. 1451 o SensorML de OGC), o incluso información de cómo comunicar y configurar el instrumento (SID, Sensor Interface Descriptor). El proceso de configuración automática se refiere en conectar el instrumento y listo para trabajar mas conocido como plug&play o plug-and-work.

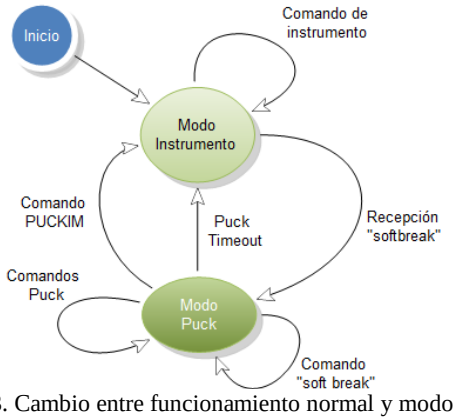

Figura 3. Cambio entre funcionamiento normal y modo Puck

El protocolo PUCK no define cómo debe operar el instrumento, ni cómo debe controlarse sino cómo obtener información sobre el instrumento de una manera estándar. Cuando conectamos un instrumento PUCK-enable a un equipo controlador, éste puede enviar comandos PUCK estándar que permiten almacenar o recuperar información del *payload* del instrumento.

La mayoría de los instrumentos están diseñados para interactuar con un controlador a través de una interfaz serie. El controlador puede proporcionar una interfaz de usuario para el instrumento, registro de datos, o distribuir los datos a una red más amplia.

Con el propósito de realizar estas funciones, el controlador requiere información sobre el instrumento, tal como la configuración del puerto serie, conocimiento de los comandos reconocidos por el instrumento, los metadatos que describen el instrumento y los datos científicos que genera. Los instrumentos que actualmente se comercializan no suministran esta información de forma automática, sino que requieren que el controlador sea configurado de antemano, y una parte del proceso de configuración generalmente consiste en la instalación del software o driver en el controlador quien gestiona la interacción con el instrumento. Esta configuración por lo general implica varios pasos, y puede ser un proceso largo, tedioso, y potencialmente propenso a errores. PUCK aborda esta cuestión de modo que permite la configuración automática del sistema cuando el instrumento está conectado al receptor.

La lista de comandos que utiliza el protocolo Puck V1.4 es solo de 14 comandos y son los siguientes:

Comando	Descripción
PUCKRM	Lectura de la memoria PUCK
PUCKWM	Escribir en la memoria
PUCKFM	Finalizar sesión de escritura en la memoria
PUCKEM	Borrar toda la memoria
PUCKGA	Obtener dirección del puntero interno de la memoria
PUCKSA	Fijar la dirección del puntero interno de la memoria
PUCKSZ	Obtener el tamaño de la memoria
PUCKTY	Leer el tipo de PUCK
PUCKVR	Leer la versión del protocolo PUCK
PUCK	Comando nulo
PUCKIM	Configurar en modo instrumento (Solo para RS232)
PUCKVB	Verificar soporte a una determinada velocidad (Solo para RS232)
PUCKSB	Fijar la velocidad de comunicación (Solo para RS232)
PUCKIP	Obtener la IP del instrumento. (Solo para IP Puck)

Tabla 1. Comandos del protocolo PUCK

El mapa de memoria PUCK *payload* se define de la siguiente manera: los primeros 96 bytes (Instrument Datasheet) son reservados para la identificación del instrumento puede configurarse del tipo solo-lectura para asegurar los datos guardados no se altere accidentalmente durante toda la vida del instrumento y los restantes bytes hasta acabar la capacidad de la memoria y es de uso libre. Es aconsejable dividir esa información por bloques "component #" en que la identificación de cada bloque en el mapa de memoria es dado por el "Payload tag". En la figura 4 podemos ver el mapa de toda la memoria PUCK y como está fraccionada.

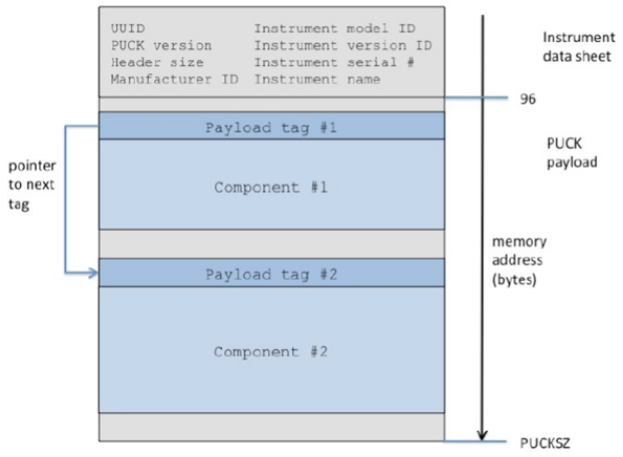

Figura 4. Mapa de memoria PUCK

El objetivo del *payload* es almacenar información sobre el instrumento, es decir, los metadatos del instrumento o también un driver ejecutable. En la tabla 2 podemos observar la descripción, el tamaño y el formato de cada campo con un total de 96 bytes (identificación del instrumento).

Descripción	Tamaño(Bytes)	Formato
UUID instrumento datasheet	16	UUID
Versión del datasheet del instrumento	2	U16
Tamaño del datasheet	2	U16
ID del fabricante	4	U32
Modelo del fabricante	2	U16
Versión del fabricante	2	U16
Numero de serie	4	U32
Nombre del instrumento	64	Char
Total	**96**	

Tabla 2. Descripción del Instrument Datasheet del instrumento

Puck atributos (Puck Attribute) describe el tipo de hardware que estamos trabajando, es decir, a través de los códigos que aparece en pantalla se puede determinar si podemos leer o escribir o leer los datos del instrumento o si el hardware es externo al instrumento como se indica en la tabla 3.

Atributos Puck	Descripción
0000	Puck instalado internamente, lectura y escrita en la memoria
0001	Puck instalado internamente, solo lectura en la memoria
0002	Puck hardware externo al instrumento
0003-8000	Reservado

Tabla 3. Atributos PUCK

En la tabla 4 se describe la definición de cada nombre de atributo de cada "tag payload", es decir cómo está definida cada etiqueta dentro de la memoria.

<puck_payload type="*type*" name="*name*" size="*size*" md5="*checksum*" next_addr="*address*" *version="version"* />

Nombre del atributo	Requiere	Descripción
Tipo	Si	Indica el tipo de payload. En la tabla 2 muestra los tipos estándar
Nombre	Si	Nombre del payload. Puede ser grabado en un archivo con este nombre
Tamaño	Si	Tamaño del payload in bytes, no incluye su tag(etiqueta)
md5	Si	MD5 Checksum del payload, no incluye el tag
Próximo enderezo	Si	Indica la próxima posición de payload tag. -1 indica que no hay mas payloads
Versión	No	Opcional indica la versión como desee el utilizador

Tabla 4. Descripción de cada nombre de atributo

3. Descripción de PUCK v1.4 TCP-IP

La novedad en la versión 1.4 en relación a la versión anterior es de poder comunicarse con el instrumento a través de una red TCP-IP. La implementación IP PUCK implica comunicaciones a través de la red Ethernet mediante este protocolo. Para el establecimiento de una comunicación TCP-IP entre los dos dispositivos o aplicaciones, la aplicación cliente deberá conocer la dirección IP de la aplicación servidor y además el puerto de comunicaciones donde dicha aplicación se mantiene a la espera de recibir peticiones de conexión. En el caso de IP PUCK, el puerto donde el instrumento que implemente IP PUCK espera la petición de una conexión por parte de una aplicación cliente que lo denominamos PUCK *port*.

El descubrimiento por parte de la aplicación cliente del valor del PUCK *port* se realiza a través del protocolo ZeroConf. El instrumento que implemente IP PUCK implementa también el protocolo ZeroConf para advertir su presencia en la red. Si el instrumento tiene comunicación Ethernet integrada en el propio instrumento como parte de su firmware, o a través de un interfaz programable serie-Ethernet puede realizar todas estas operaciones. Este programa tiene la ventaja de buscar automáticamente en la red dispositivos que implemente el protocolo PUCK sin conocer su IP. Se conecta al primero que responde y muestra una lista de todos los instrumentos que hay conectados en la red.

La parte más complicada de este proyecto fue sin duda la implementación del ZeroConf. Esto porque fue necesario implementar los 2 protocolos: UDP y TCP. Después estudiar la documentación ZeroConf llegamos a la conclusión que no era necesario aprender todo el protocolo pero solo la parte que nos interesa que es el contenido del mensaje con los datos que nos interesa porque todo el resto ya lo implementa el proprio sistema Windows que son: el protocolo internet, puertos, frames, protocolo de comunicación, entre otros.

Como el objetivo era solo descubrir donde están ubicados los mensajes dentro las tramas de comunicación entre el ordenador y el instrumento pasamos a la ingeniería inversa. Realizar pruebas con un programa que implemente el ZeroConf como por ejemplo el "JMDns", ejecutamos el programa y con el "Wireshark" estudiamos donde está ubicada el mensaje de datos que nos interesa. Pero no es tan simple como parece, fue necesario estudiar cómo se componen cada una de las tramas.

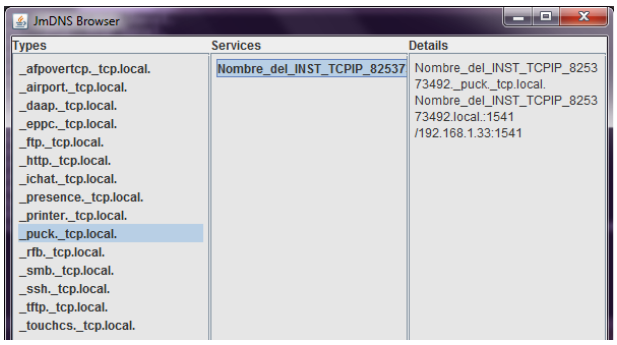

Figura 5. Programa JmDNS, servicios ZeroConf.

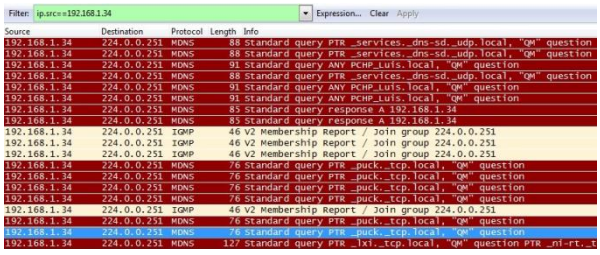

Figura 6. Resultados del pedido de servicios ZeroConf del JmDNS

Una vez estudiado como funciona las tramas y como están constituida cada trama lo aplicamos en el programa cambiando los nombres del ordenador y del instrumento que queremos comunicar.

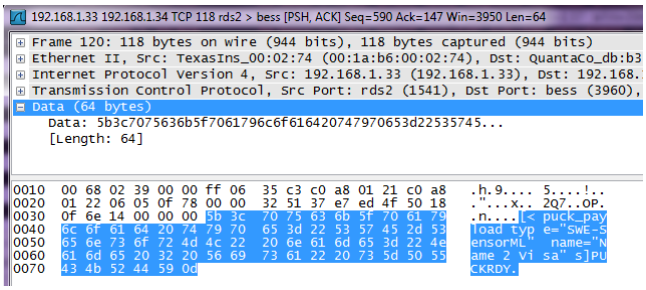

Figura 7. Resultado de una respuesta del instrumento

En la figura 8 tenemos el panel del programa para la detención automática de servicios en la red que para nuestro caso solo nos interesa el servicio "_puck_tcp_local." Este pequeño programa se ejecuta en paralelo con el programa principal de modo que esté a la escucha si algún instrumento se conecta a la red.

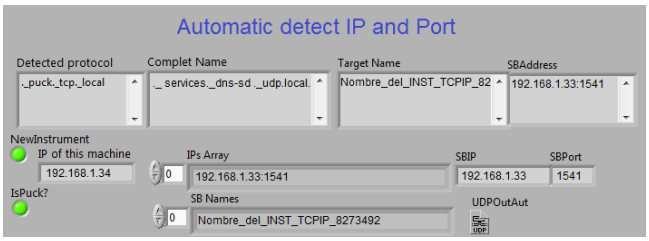

Figura 8. Programa en LabVIEW para detectar servicios en la red

4. Desarrollo del programa

Este programa fue realizado en LabVIEW de la National Instruments, con el objetivo crear una plataforma visual, fácil de configurar el instrumento con el ordenador que tengan implementado el protocolo PUCK: leer y escribir la identificación del instrumento, leer y escribir información libre en la memoria del instrumento a través de los PUCK *payloads*. También tiene la posibilidad de grabar esa información en un archivo en el ordenador u otro medio de memoria conectado al ordenador. Puede controlar el instrumento en *Puckmode o Instrument Mode* y modificar la identificación del instrumento. No obstante este programa no lee valores realizados con los instrumentos, solo realiza la configuración entre el ordenador y el instrumento.

El programa puede trabajar con instrumentos conectados en los puertos COM utilizando el protocolo RS232, Visa (National Instruments) u otro tipo de programa que pueda realizar un puente a un puerto, sea RS232, USB o otro y una IP conectados a una red.

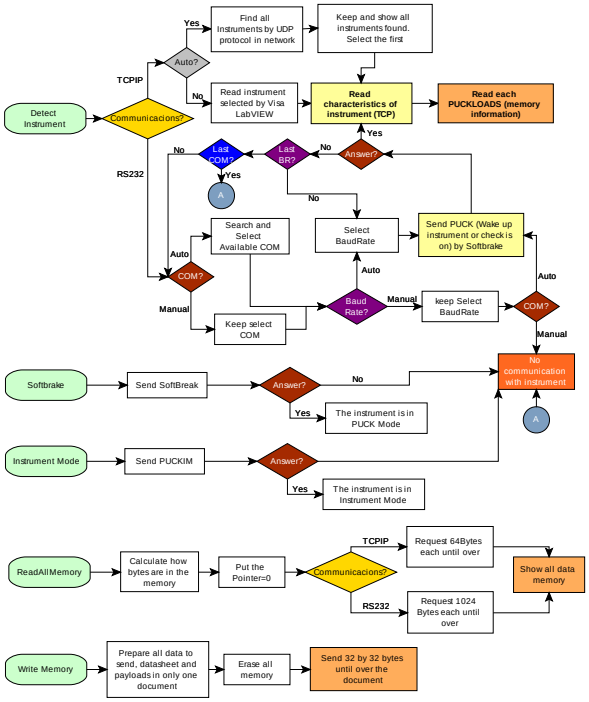

Figura 9. Diagrama con los principales funciones del programa

El programa final es como un puzle de varios pequeños programas (vi's) que están conectados entre sí. Mucho de estos pequeños programas fueron pruebas asiladas de envío y recepción de datos con las placas que contenían los programas PUCK tanto en RS232 y TCPIP. En la figura 9 podemos ver el diagrama con las principales funciones del programa: detención de un instrumento, sotfbreak (modo PUCK, modo instrumento, leer toda la memoria y escribir en la memoria.

5. Pruebas con RS232 y TCP-IP

En la figura 10 podemos ver la ventana o panel principal del programa donde tenemos tanto las opciones que queremos ejecutar como las características relacionadas con el instrumento que queremos ejecutar.

La primera orden del programa será siempre en ejecutar el detectar el instrumento, escogiendo unas de las opciones automáticas o manual del puerto RS232 o TCPIP.

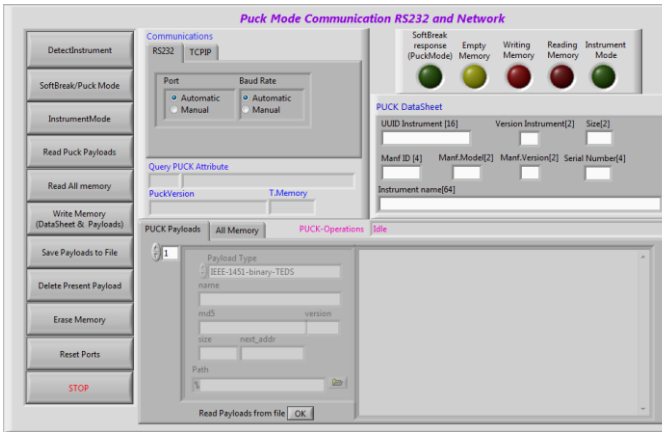

Figura 10. Pantalla principal.

En RS232 el usuario tiene la opción de escoger la detención automáticamente del puerto y de la velocidad de trabajo Baudrate o escoger el automático de uno de ellos (puerto o velocidad). También tiene la opción de seleccionar los dos en manual.

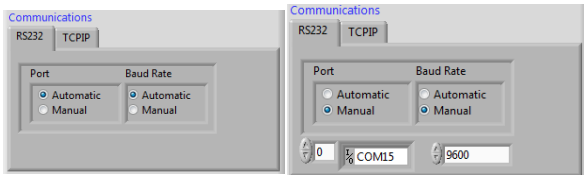

Figura 11. Comunicación RS232 automático o manual

En la parte de TCPIP tenemos 2 opciones: Manual y automática. En manual es dedicada al Visa lo que deberá programar y activar la IP y puerto que está conectado el instrumento antes de ejecutar el programa. En automático el programa buscara en la rede instrumentos conectados con el protocolo PUCK

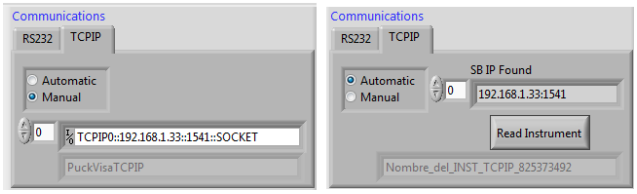

Figura 12. TCPIP Visa y TCPIP Ethernet

Una vez ejecutado el programa el utilizador tiene la opción de escoger como encontrar el instrumento con RS232 o TCPIP. Luego deberá ejecutar el botón "Detect Instrument" y esperar que el programa detecte el instrumento. Si lo encuentra, automáticamente lee los datos que se encuentra en la memoria: identificación del instrumento y todo que tenga en la memoria *payload*.

Es de subrayar, si hay más de un instrumento conectado en la red o en los puertos COM y la opción es de automático leerá los datos del primero instrumento que encuentre, puede que no sea el instrumento que se desea visualizar o trabajar. Se así es, bastará seleccionar el instrumento que desee trabajar en la lista de instrumentos encontrados y volver a ejecutar el "Detect Instrument" para el caso de RS232 y para el caso de TCPIP "ReadInstrument".

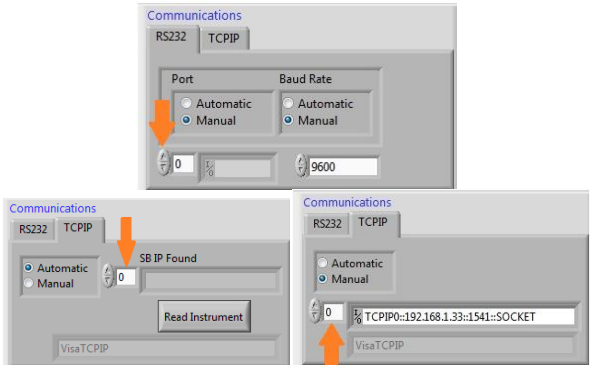

Figura 13. Lista instrumentos detectados

Este programa tiene la opción de cambiar de "Puck Mode" a "Instrument Mode" y viceversa, esto quiere decir que en el primero caso es para leer y programar el instrumento.

Es necesario tener en cuenta que al ejecutar esta opción el instrumento deja de trabajar como instrumento, es decir deja de realizar y enviar las medidas que estaba ejecutando. En la segunda opción es para que el instrumento vuelva a ejecutar las funciones normales de funcionamiento.

Los siguientes botones son para leer el contenido de la memoria: por separado "Read Puck Payloads". "ReadAll Memory", toda la memoria en una lista incluyendo la identificación del instrumento.

El botón "Write Memory" es para escribir la información en el instrumento y el botón "Save Payloads to File" es para grabar toda la información del instrumento en un archivo del ordenador pero no graba al mismo tiempo en el instrumento deberá ejecutar el "Write Memory" si desea grabar en la memoria PUCK del instrumento.

El botón "Delete Present Payload" es para borrar el contenido actual del payload que esta visualizado y al mismo tiempo reorganiza todos los payloads. El botón "Erase Memory" borra todo el contenido de la memoria incluyendo la identificación caso no esté activada.

El "Reset ports" es para buscar los puertos que están activos en el ordenador una vez ejecutado el programa y caso no quiera cerrar el programa, esta función es para RS232 y Visa.

En la figura 14 podemos ver el resultado de de una comunicación automática en RS232. En la figura 15 tenemos el resultado de la lectura de instrumento en una rede Ethernet.

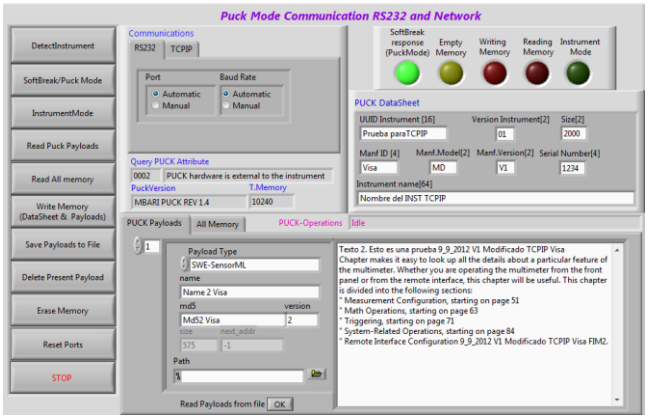

Figura 14. Resultado RS232 automático, puerto y Baudrate

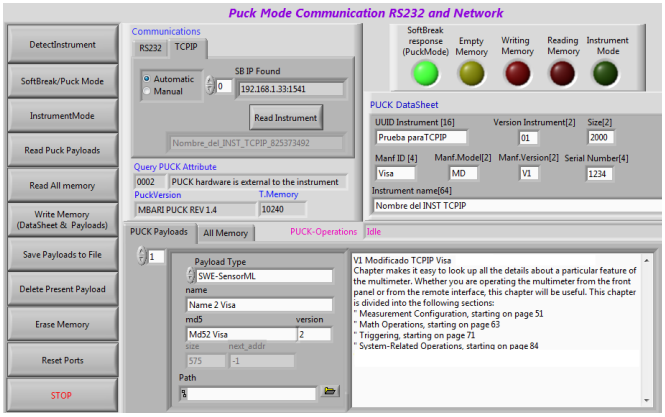

Figura 15. Ejemplo de una conexión TCPIP

6. Conclusiones

La parte de RS232 fue la más larga pero no la más difícil esto porque las pruebas de la implementación de la lectura y escrita en la memoria PUCK fueron realizadas paso a paso, es decir, hasta estar seguro que los datos eran correctos y que la programación se comportaría de una forma uniforme para todas las situaciones frente a algunos imprevistos.

Los imprevistos más importantes fueron con los tiempos de espera y confirmación de los datos, separar la información de modo a identificar y separar cada paquete tanto en la recepción como en envío de los datos y finalmente si eran correctas. Lo más importante es la identificación del instrumento, respectar los 96 bytes y dentro de esta trama separar todos las 8 diferentes partes de la identificación del instrumento. A partir de aquí verificar que los *payloads* estén bien identificados dentro de la memoria de modo a poder leer y escribir correctamente.

La parte más complicada de este proyecto fue sin duda la parte de TCPIP. Esto porque fue necesario implementar los protocolos ZeroConf: UDP y TCP. Después estudiar la documentación ZeroConf llegamos a la conclusión que no era necesario aprender todo el protocolo pero solo la parte que nos interesa que es el mensaje con los datos que nos interesa porque todo el resto ya lo implementa y se encarga el proprio sistema Windows.

Otro de los puntos fuertes de este trabajo es que el código con que está programada versión V1.3 en vez de la V1.4, esto para que este programa puede funcionar en cualquier versión anterior V1.2, V1.3 y V1.4. La diferencia más significativa entre las 2 versiones es solo el comando PUCKIP, obtener la IP del instrumento. Este comando no fue necesario implementarlo porque se obtiene de la IP del instrumento a través del protocolo UDP.

Este proyecto aporta una mejora a este nuevo protocolo en la configuración de los instrumentos de medida de modo a facilitar de una forma automática o manual a través de una plataforma *plug&play* interactiva, no solo en el control y configuración del instrumento pero también para la lectura y escrita de los datos en la memoria PUCK.

Agradecimientos

Agradezco el apoyo, la ayuda, dedicación y la disponibilidad de mi tutor y de todo el material cedido para la realización de este proyecto.

Referencias

[1] OGC Making location count http://www.opengis.net/spec/PUCK/v1.4/
[2] Geo Connexion http://www.geoconnexion.com/news/ogc-puck-protocol-standard-adopted
[3] WIRESHARK the world's foremost network protocol analyzer http://www.wireshark.org
[4] Developer for OS X https://developer.apple.com/technologies/mac/
[5] Monterey Bay aquarium Research Institute http://www.mbari.org/news/homepage/2012/puck/puck.html
[6] National instruments www.ns.com
[7] Tom O'Reilly. "Open Geospatial Consortium. OGC® PUCK, Protocol Standard Version 1.4." document: "09-127r2_OGC_PUCK_Protocol_Standard_Version_1.4". http://www.opengeospatial.org/legal/.
[8] DomainName System (DNS), RFC 1034 y RFC 1035 http://www.netfor2.com/dns.htm
[9] Tom O'Reilly "MBARI PUCK Specification Version 1.3." Monterey Bay Aquarium Research Institute "09-127r2_OGC_PUCK_Protocol_Standard_Version_1.3".
[10] Stuart Cheshire & Daniel H.Steingerg "Zero Configuration. Networking." The definite guide. O'Reilly
[11] P. Mockpetris ISI "Domain Names - Implementation and especification." Network working – RCF1035
[12] Apuntes y prácticas de las signaturas de ININ y de SIPE