Sistema embebido de bajo costo como instrumento auxiliar de laboratorio para operación remota

Ing. Alfredo Taddei, Mg. Ing. Héctor Riso, Dr. Pedro E. Colla
Especialidad en Sistemas Embebidos – Instituto Universitario Aeronáutico
Av. Fuerza Aerea Km 8 ½, Córdoba, Argentina.
{ataddei,hriso,pcolla}@iua.edu.ar

Abstract- **Se hace un análisis sobre la utilización de sistemas embebidos como instrumentos de laboratorio tanto para realizar tareas que ya eran ejecutadas como para nuevos usos auxiliares. Se hace hincapié en la capacidad de trabajo remoto provista por la conexión de red. Se describirá el hardware objeto de análisis y se demostrará a través de dos sencillas aplicaciones su potencialidad como instrumento auxiliar de laboratorio.**

Keywords: instrumentos, laboratorio, sistemas embebidos.

I. PROBLEMA

En el ámbito de un laboratorio de comunicaciones ópticas donde se testea un circuito ASIC (Application Specific Integrated Circuit) transceptor de señales, una de las pruebas que se realiza consiste en hacer funcionar el integrado por un tiempo ya sea para caracterizar su performance en sistemas con tasas de error muy precisas o su integridad física [1]. Es deseable durante estas pruebas sensar variables del entorno y registrarlas para poder detectar problemas en caso de fallos e identificar el estado del entorno en dicho momento sin un personal técnico presente.

Para auxiliar a esta operación se necesita de un instrumento encargado de adquirir estas señales, registrarlas, almacenarlas y tenerlas disponibles remotamente para consulta. Las variables de entorno a recolectar pueden ser analógicas o digitales tales como tensión proporcional a la temperatura del integrado, tensiones de alimentación al integrado, la posición de las llaves de configuración en la placa o registros con información provenientes del integrado o de la placa de evaluación. Un instrumento de tamaño reducido, ciertamente facilita su traslado a distintas ubicaciones donde se requieren sus funcionalidades para el tipo de prueba requerida.

En otro orden, cuando el integrado se encuentra en fase de validación y verificación es necesario trabajar con la última versión de firmware; esto se asegura típicamente mediante la utilización de una PC con conexión un versionador (SVN x ej). Resulta de interés entonces reemplazar esta PC para la función de programación de firmware por un instrumento con conexión Ethernet e interfaz de comunicación con el integrado mediante protocolos como el I²C (Inter Integrated Circuit) o SPI (Serial Peripheral Interface). De esta manera se podría lograr un programador transportable capaz de actualizarse remotamente.

En resumen, se identifica la clara necesidad de tener un instrumento de laboratorio suficientemente flexible y de bajo costo para reemplazar el uso de PC. Este trabajo describe un sistema de esta índole basado en sistemas embebidos.

II. PROPUESTA

Los sistemas embebidos han ido desarrollándose a tal punto que en muchos casos la totalidad de las funciones de un instrumento (adquisición, control, almacenamiento, análisis y presentación de mediciones) pueden ser cubiertas por un sistema de estas características. Como solución a lo planteado en la sección anterior se diseñará una prueba de concepto de un instrumento para esta situación real de laboratorio y se explicará como mejora aspectos de trabajo y agrega posibilidades que no eran contempladas anteriormente, como obtener.

Con esta propuesta, que va mas allá de un sensor autónomo, muchas de las funciones antes residentes en un instrumento o en una plataforma PC pueden ser realizadas en un sistema embebido de bajo costo; aún teniendo en consideración las limitaciones de performance. Su valor agregado, consiste en el acceso remoto a mediciones auxiliares en circuitos donde no se había contemplado la posibilidad de medición de estas variables en el diseño del hardware de integración del ASIC.

Se agrega a esta propuesta, la capacidad de administrar el firmware que configura los registros del ASIC. Mediante una conexión a un servidor se descarga la última versión de firmware y luego se la envía al dispositivo bajo testeo. Esta aplicación de programación remota, potencia los futuros usos del instrumento ya que permite tomar control del dispositivo y cambiar sus condiciones internas.

En la Figura 1, se describe esquemáticamente la arquitectura de la interacción entre el instrumento y el hardware del integrado a testear. En la misma se puede observar el ASIC con su entorno de testeo y el instrumento auxiliar con sus componentes de comunicación (SPI, USB,I2C, etc) asi como el registro de variables (ADC, GPIO), el almacenamiento (SD Card) y los medios de comunicación remota (Ethernet, Web Server).

En resumen, se propone un instrumento novedoso que toma mediciones auxiliares del entorno de prueba, las pone disponibles para su consulta remota y accede a los registros y programa el firmware del ASIC a testear. Tantas funciones solo podrían integrarse en una PC con hardware adicional o un instrumento dedicad. Ambas alternativas requerirían diferentes costos que esta propuesta supera, como se describe en la sección III, combinando lo mejor de ambas.

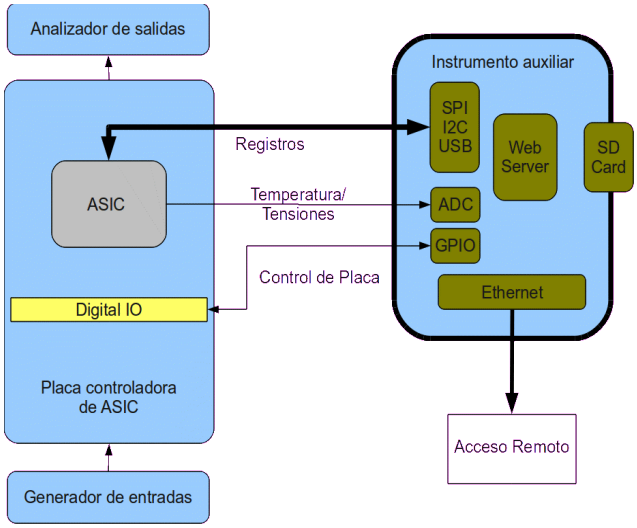

Figura 1. Arquitectura de Instrumento de Medición

III. VENTAJAS

Actualmente conviven dos tendencias en el área de instrumentación. La primera consiste en usar a la PC como "instrumento virtual" añadiéndole el hardware necesario para la adquisición. La segunda consiste en realizar el instrumento íntegramente en un sistema embebido con un microcomputador, microprocesador o microcontrolador, conformando lo que comúnmente se conoce como sistema autónomo de adquisición y registro [2].

Los "instrumentos virtuales" son periféricos, o placas de adquisición, conectadas a una computadora la cual configura, controla y accede a la placa y sus mediciones a través del sistema operativo. Esto dota a una computadora la habilidad de ser un dispositivo modular al cual se añade un hardware con el cual se comunica para realizar las funciones de instrumento propuestas (en este caso el sensado y comunicación de la medición). Bajo este paradigma, la medición se realiza en el periférico externo y el análisis, almacenamiento, comunicación, exposición de los datos y control en el sistema operativo o software dedicado. Ejemplos de este tipo de instrumental encontramos en laboratorios virtuales de universidades, donde todo la configuración de prueba se hace operable remotamente [3]. También existen aplicaciones de monitoreo en varios ámbitos cuya comparativa con un instrumento especifico para su propósito resulta mucho más económica y flexible [4].

Los microcontroladores hoy son la alternativa mas económica ya que integran el procesamiento y los periféricos. Su utilización en varios mercados y en particular para el consumo masivo ha conseguido que sus precios se reduzcan. Con un sistema embebido, la totalidad de las funciones de un instrumento pueden ser realizadas en el mismo hardware. La adquisición a través de sus periféricos (dentro del micro controlador o en la misma placa de desarrollo), el almacenamiento en memorias accedidas por el micro, la comunicación y el control a través de sus interfaces, y la exposición mediante los servicios que brinda el sistema operativo, ya sea un servidor Web o una gráfica en pantalla. Los sistemas autónomos de recolección de datos, como el propuesto para este trabajo, tienen en su núcleo un microprocesador o un microcontrolador, siendo este ultimo la opción preferencial ya que provee las funcionalidades necesarias para los periféricos en el chip mismo, reduciendo espacio y costos [5]. Asimismo, al considerar la propuesta como un "computador reducido" podemos migrar a futuro todas las funcionalidades que actualmente se resuelven en una PC de escritorio dotando a esta propuesta la capacidad de ser integradora de mas actividades y capaz de ser versátil para ajustarse a estos nuevos usos. Algo que instrumentos sensores autonomos con GPIB no pueden llegar a ser. Por último el sistema embebido es mas eficiente en consumo de energía siendo esta otra ventaja económica ya que el costo operativo de un "instrumento virtual" tiene asociado el consumo de una PC de escritorio.

En resumen, las ventajas a obtener de esta propuesta son:

- Costo de adquisición y operación.
- Acceso remoto a los datos.
- Facilidad de transporte en uso de campo.
- Versatilidad.
- Capacidad de integración a futuro.
- Funcionalmente satisfacer la problemática expuesta.

IV. DISEÑO

Con el fin de satisfacer las necesidades anteriormente planteadas se implementaron en este instrumento dos funcionalidades.

La primera, será funcionalidad de sensor. Consiste en adquisición de variables del entorno del laboratorio descriptas anteriormente, así como su registro y almacenamiento, todo esto con la posibilidad de comando y consulta de manera remota.

Como segunda, será funcionalidad de programación remota. El instrumento deberá conectarse a un servidor desde el cual descargará un firmware y una vez descargado procederá a enviarlo al dispositivo bajo testeo. De esta manera remotamente se puede ajustar la programación del integrado y funcionar como un programador que obtenga la ultima revisión de configuración. A futuro esta función facilitaría la posibilidad de hacer pruebas reactivas, es decir que luego de procesar variables recogidas se detecte una condición que dispare un cambio en la configuración del integrado mediante la funcionalidad descripta.

A. Hardware

Por cuestiones de conveniencia logística para el trabajo se utilizó la placa de desarrollo MINI2440 (Figura 2). Consiste en un integrado Samsung S3C2440A que implementa un microprocesador ARM920T [6]. El propósito de este trabajo es brindar la solución optando por este tipo de sistema como alternativa a los Instrumentos Virtuales por las ventajas descriptas en la sección III.

Las prestaciones de la placa de desarrollo del microcontrolador ARM[7] significativas y suficientes para el propósito de este trabajo son:

1. Interfaces SPI e I²C.
2. 8 entradas analógicas multiplexadas a 500Ksps de 10 bit de resolución.
3. 130 entradas de propósitos generales (digitales) GPIO.
4. Controlador de tarjeta de almacenamiento SD.
5. Puerto Ethernet RJ-45 10/100M (Integrado DM9000)
6. Sistema operativo Linux con servicios de FTP y WebServer (BOA).

B. Análisis de funcionalidad de sensor

El flujo de actividades involucradas en el proceso se describen en la Figura 3. En ella quedan reflejadas las distintas tareas a codificar y orquestar en el programa. En 1) El inicio del programa configura el entorno, declara los procesos y sus variables y registra las variables externas tomadas del argumento con el que se llama al programa. En 2) se da inicio a los procesos concurrentes que sensarán la variable analógica en 2.1) y las banderas digitales en 2.2). Siguiendo la secuencia, 3.1) realiza una lectura periódica, cuyo periodo es programable externamente, del periférico conversor analógico digital y hace accesible al programa principal el valor. Al mismo tiempo, 3.2) sensa el estado de las entradas digitales externas y registras los cambios de estado para que también sean accesibles desde el orquestador. Una vez que estos datos están disponibles 4) los imprime en un archivo ubicado en el espacio de la tarjeta SD. El flujo recircula nuevamente desde las actividades del nivel 3) volviendo a tomar las muestras periódicas y los cambios de estado. La condición de salida del programa puede ser una señal externa que interrumpa el flujo o el fin del tiempo estimado de la prueba del laboratorio.

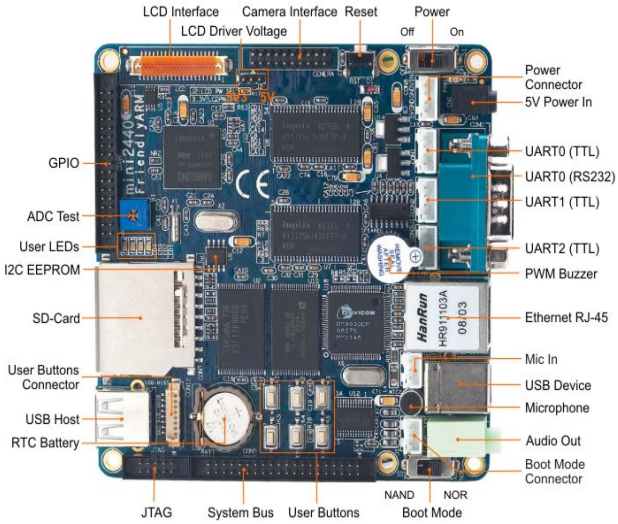

Figura 2 Fotografía de placa MINI2440

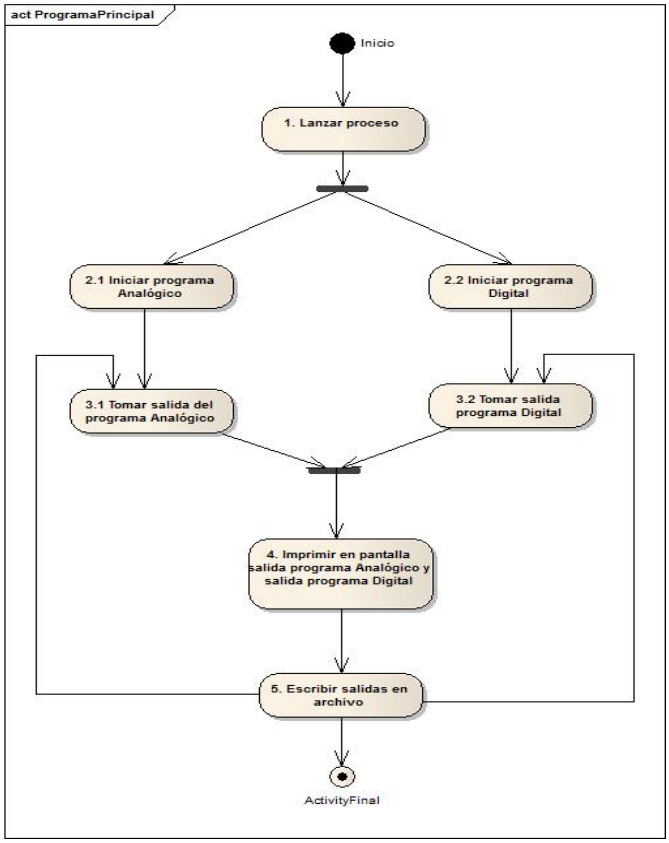

Figura 3 Diagrama de actividades de sensor

C. Análisis de funcionalidad de programación remota.

Para esta segunda funcionalidad, se desarrolló un código para un servidor TCP que ofrece el ultimo firmware a configurar en el ASIC a testear, y un código para el cliente que descarga del servidor este firmware y lo envía mediante el bus I²C. Se basa en un protocolo simple donde el servidor envía el Firmware ante un requerimiento del cliente mediante un comando. Una vez que el cliente recibe el dato del firmware, cierra la conexión y procede con su segunda tarea de enviarlo a la memoria donde se alojan los registros. La elección del protocolo TCP por sobre UDP para el envío de los datos se debió a la necesidad de contar con un control de transmisión que garantice que el firmware a testear es el enviado desde el servidor. En la Figura 4 se muestra el diagrama de secuencia de esta funcionalidad.

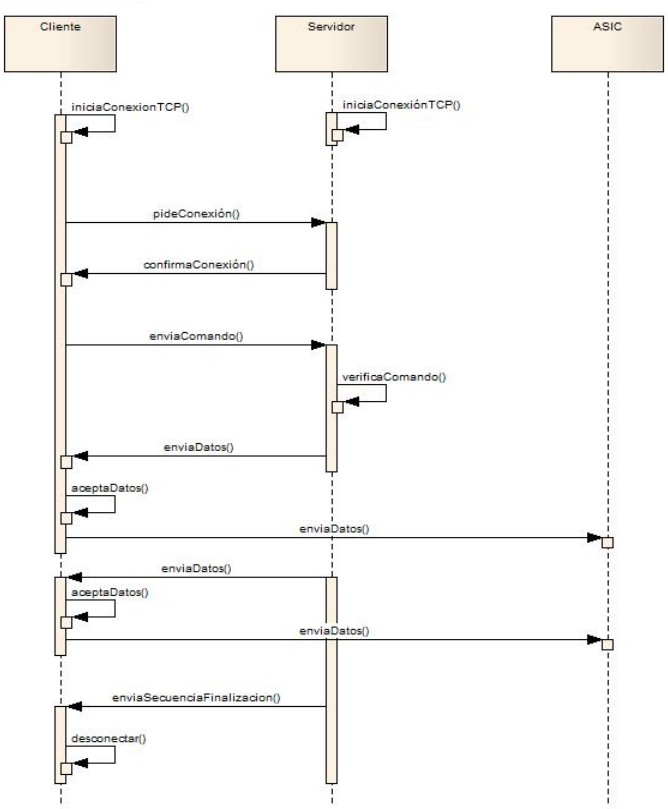

Figura 4 Funcionalidad de Programación Remota

V. IMPLEMENTACIÓN

Se implementaron ambas funcionalidades en la placa MINI2440 a un nivel de prueba de concepto que permita verificar la factibilidad funcional de lo propuesto. Las tareas a realizar del microprocesador fueron codificadas en lenguaje ANSI C y compiladas para un sistema operativo Linux embebido. Toda la comunicación remota con el instrumento se hizo mediante el servicio *Telnet* prescindiendo de una interfaz gráfica. El acceso remoto puede lograrse mediante un sitio web embebido con el WebServer BOA que provee el software original de la placa y *scripts* CGI (Common Gateway Interface) que lanzan los ejecutables.

Para la funcionalidad del sensor el programa principal crea dos procesos en Linux *pthreads* que se ejecutan concurrentemente.

```
//Creating buttons thread
pthread_create(&breader,NULL,
(void*)&buttons_reader_function,NULL);

//Creating ADC Read thread
pthread_create(&areader,NULL,
(void*)&adc_reader_function,NULL);
```

La funcion *buttons_reader_function*, que implementa el proceso de adquisición digital, lee el estado de los puertos de interrupción donde están conectados los botones de prueba de la placa y emulan el puerto GPIO a implementar. Inicia abriendo el archivo (*handler*) de donde leerá los valores

```
//Opening buttons handler
buttons_fd = open("/dev/buttons", 0);
```

Luego ejecuta el lazo del proceso testeando la integridad de la lectura y notificando los valores recibidos. El lazo y la función de lectura se representan en este código abreviado:

```
while(1)
{
if (read(buttons_fd, current_buttons, sizeof
      current_buttons) != sizeof current_buttons)
      {//ERROR EN LECTURA
      perror("read buttons:");exit(1);
      }
      //Codigo que detecta los cambios de
      //estado y registra los valores leidos
}//end while(1)
```

La función *read*() que lee *"/dev/buttons"* es bloqueante, lo cual determinó que el programa principal fuera hecho con hilos concurrentes (*pthreads*) para que los demás procesos pudieran ejecutarse concurrentemente.

La función *adc_reader_function* que implementa la adquisición analógica abre el archivo de lectura (*handler*) donde el driver guarda el valor actual e inicia el *loop* donde lee un valor de tensión y lo registra.

```
int fd = open("/dev/adc", 0);
while(1)
{
char buffer[buffer_size];
int len = read(fd, buffer, sizeof buffer -1);    if (len
> 0) {
      buffer[len] = '\0';
      sscanf(buffer, "%d", &adcvalue);
            }
      else{
            perror("read ADC device:");
          }

usleep(delay);
}
```

Luego de la lectura en la variable *buffer* y su posterior asignación en la variable global *adcvalue*, el loop espera un retardo programable con la variable *delay* en la función *usleep*. Finalmente, se implementa una función que recoge las variables globales y las registra en un archivo dentro de la tarjeta SD.

Para la funcionalidad de programación remota se implementa una arquitectura cliente-servidor. El componente servidor abre una conexión (*stream*) TCP y configura su puerto con un argumento de entrada chequeando el éxito de la operación y aguardando peticiones.

```
sockfd = socket(AF_INET, SOCK_STREAM,0);
if (sockfd < 0)
      error("ERROR opening socket");
bzero((char *) &serv_addr, sizeof(serv_addr));
portno = atoi(argv[1]); //port number argument
serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = INADDR_ANY;
serv_addr.sin_port = htons(portno);
if (bind(sockfd,(struct sockaddr*)
&serv_addr,sizeof(serv_addr)) < 0)
      error("ERROR on binding");
listen(sockfd,queue_limit);
```

Posteriormente, luego de que detecta la cadena de texto BOOT enviada por el programa cliente, procede a enviar el firmware con la función *write()*.

```
n=read(newsockfd,buffer,buffer_size);
//codigo de chequeo de integridad de datos leidos
```

```
//Parsing cadena BOOT
if((strstr(buffer,"BOOT")!=NULL))
      n=write(newsockfd,minibuffer,buffer_size);
```

Por su parte el componente cliente se implementa de modo que éste abre una conexión (*stream*) TCP y configura la dirección IP (argv[1]) y el puerto del servidor (argv[2]) a través de argumentos de entrada chequeando el éxito de la conexión y abortando en caso de falla.

```
portno = atoi(argv[2]);
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd < 0)
      error("ERROR opening socket");
server = gethostbyname(argv[1]);
if (server == NULL) {
      fprintf(stderr,"ERROR, no such host\n");
exit(0);
}
bzero((char *)&serv_addr, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
bcopy((char *)server->h_addr, (char*)
&serv_addr.sin_addr.s_addr, server->h_length);
serv_addr.sin_port = htons(portno);
if(connect(sockfd,(struct sockaddr*)
&serv_addr,sizeof(serv_addr)) < 0)
      error("ERROR connecting");
```

Una vez iniciada la comunicación y enviado el comando "BOOT" por el cual el protocolo del servidor inicia el envío del firmware, el cliente a medida que recibe los datos los envía a una memoria conectada al bus I²C con los drivers provistos por el fabricante de la placa. La comunicación finaliza cuando el servidor envía la cadena "OUT".

```
n=read(sockfd,buffer,buffer_size);
if (n < 0)
      error("ERROR reading from socket");
while(strstr(buffer,"OUT")==NULL)
{
data=atoi(buffer);//parse to int
//Check valid data code
eeprom_write_byte(&e, addr,data);
//Write in EEPROM via I2C and verification
n = read(sockfd,buffer,buffer_size);//New read
if (n < 0)
      error("ERROR reading from socket");

}//end while
```

VI. VERIFICACIÓN Y VALIDACIÓN

Es necesario realizar la verificación de forma que se revise la integridad de lo implementado (¿hemos implementado correctamente?) así como la validación de forma de corroborar que la funcionalidad implementada satisface las necesidades planteadas (¿hemos implementado el dispositivo correcto?).

En el caso de la funcionalidad de sensor su verificación fue ejecutada en la misma placa de desarrollo con componentes conectados al microcontrolador destinados a probar periféricos (ver en Figura 2 los elementos ADC Test y User Buttons). Se comprobó el correcto funcionamiento de la adquisición digital y analógica utilizando períodos de registro distinto. Se accedió remotamente a los datos registrados y guardados en el archivo mientras se realizaba la prueba utilizando el servicio FTP que provee Linux (*ftpd*) y se comprobó que el contenido del archivo correspondía a los estímulos ejercidos manualmente en el entorno.

En el caso de la funcionalidad de programación remota, se probó emulando un servidor TCP en una computadora portátil que envía bajo petición el firmware al instrumento y el mismo, luego de recibido, es programado en una memoria mediante el bus I²C. Se verificó la integridad de los datos recibidos comparando la secuencia enviada con la lectura de los datos de la memoria EEPROM conectada al bus I²C.

Si bien abundan las alternativas dentro de los sistemas embebidos que consisten en una placa de desarrollo de un microcontrolador, el hardware que se propone se ajusta a las necesidades del instrumento a desarrollar en cuanto a funcionalidad y costo delimitadas para este trabajo. El aumento de prestaciones y rendimiento impactan en el costo de la solución. La placa de desarrollo MINI2440 del microcontrolador S3C2440 que se utiliza en este trabajo tiene un costo de US$89 o US$109 según se ordene con pantalla LCD touch de 3.5" o no [8]. A comparación con las placas de adquisición de datos que requieren de una PC o incluso las placas de computadores industriales (*Single Board Computers*) u otros sensores autónomos, cuyos valores van desde los US$200 a US$1000 observamos la ventaja de usar esta plataforma para desarrollar un instrumento nuevo [9]. Se aprecia en la Tabla 1 la diferencia de costos entre posibles soluciones comerciales [10][11].

TABLA 1 COMPARACIÓN ENTRE DISTINTAS ALTERNATIVAS

Nombre	Tipo	Costo	Consumo
MINI2440	Embebido económico con microcontrolador ARM	US$89	1.5W
iUSBDAQ-U120816	Tarjeta de Adquisición de datos USB para PC	US$99 + Computadora	>30W (PC)
DI-710-EHS	Sensor autónomo comercial, portable con Ethernet y tarjeta SD	US$799	2W

El costo de operación fue asociado a la potencia de consumo del instrumento completo. En la Tabla 1 se hace una comparación de un ejemplo de cada alternativa.

Otro aspecto valuable de esta propuesta consiste en la posibilidad de desmontar el instrumento del escritorio de trabajo fácilmente y transportarlo con comodidad por su tamaño menor (10cm x10cm) con respecto a una computadora. Al ser una prueba de carácter no frecuente, la posibilidad de desmontar el instrumento fácilmente y llevarlo a otro sitio donde se necesite es una mejora si se lo compara con la alternativa de implementar todas estas funcionalidades en todas las placas bajo testeo [12].

VII. CONCLUSIONES

En este trabajo se desarrolló una aplicación de adquisición, registro, monitoreo y acceso de variables de entorno en un laboratorio de testeo de ASIC de comunicaciones ópticas. Siendo la tendencia de esta investigación reemplazar la PC que coordina la prueba por un sistema embebido, se desarrolló una segunda aplicación que descarga de un servidor la ultima versión de firmware y su posterior aplicación al ASIC bajo testeo.

Se comparó la alternativa de sistema embebido basado en un microcontrolador a usar placas de adquisición como "instrumentos virtuales", resultando más económica y versátil

al momento de hacer un instrumento transportable para la prueba objeto de este trabajo.

Se seleccionó un hardware que cubre las expectativas de prestaciones para este trabajo y posteriormente se describieron las implementaciones de las dos aplicaciones propuestas. La verificación de lo propuesto se hizo emulando las condiciones externas con hardware de la misma placa de desarrollo del sistema embebido elegido.

Como conclusión se puede afirmar que estas aplicaciones simples desarrolladas pueden ser cubiertas rápidamente con este hardware de sistema embebido económico y en pocas líneas de código obtener una aplicación útil que cumpla con lo requerido a menor costo y portable. A medida que los requerimientos de estas funcionalidades se hagan más estrictos, se deberán tener en cuenta muchos más aspectos de implementación de estos programas que podían ser soslayados anteriormente (por ej.: muestreo uniforme, concurrencia de tareas). A futuro debería hacerse una revisión minuciosa de la robustez del código programado.

Mejoras posibles incluyen desarrollar un servidor Web para comandar mas amigablemente el instrumento e ir incorporando actividades que hasta el día de hoy las realiza una computadora de escritorio. Una de las tareas centrales es la de coordinar el funcionamiento de los instrumentos que proveen las señales de entrada, analizan señales de salida y estresan las condiciones del entorno. El estándar utilizado es GPIB con tendencia a migrar las comunicaciones a LAN Ethernet, lo cual sería beneficioso para el tipo de hardware propuesto pues no habría ningún desarrollo de bus de comunicaciones que realizar. Se obtendría un instrumento capaz de comandar a cualquier instrumento con GPIB tal cual lo hace una PC y de esa manera realizar cualquier tipo de tests como test de regresion, test de humo o test reactivo. En [13], el fabricante de hardware de instrumentos Agilent, explica con una nota de aplicación esta posibilidad en Linux.

REFERENCIAS

[1] Birolini, Alessandro: Reliability Engineering, Springer (2007)

[2] Mackay S., Park, J: Practical Data Acquisition and Control Systems. Elsevier p. 204 (2003)

[3] Ko C.C., Chen B.M., Hu S., Ramakrishnan V., Cheng C.D, Zhuang Y., Chen J: A Web-Based Virtual Laboratory on a Frequency Modulation Experiment. IEEE Transactions on systems, man, cybernetics Part C: Applications and reviews, Vol. 31, No. 3, (August 2001)

[4] Batista J., Afonso J.L., Martins J.S., Low-Cost Power Quality Monitor Based on a PC, IEEE International Symposium on Industrial Electronics (2003)

[5] Mackay S., Park, J: Practical Data Acquisition and Control Systems. Elsevier p. 210 (2003)

[6] ARM Information Center, http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0151c/index.html

[7] Mini2440 Hardware Overview, http://www.friendlyarm.net/dl.php?file=mini2440_overview.pdf

[8] Friendly ARM MINI2440, http://www.andahammer.com/mini2440-sdk/

[9] Omega Press, Transactions in measurement and control, Vol II, Data Acquisition, Chapter 5. http://www.omega.com/literature/transactions/volume2/trantocvol2.html

[10] iUSBDAQ – U120816, Lowest cost USB data acquisition module with 8 channels 12bit analog inputs, 16 bi-directional DIOs, 2 PWM outputs, one 16bit counter. http://www.hytekautomation.ca/U120816.aspx?productId=1

[11] DI-710 Data Logger products operate Stand-alone or PC-connected. http://www.dataq.com/products/hardware/di710.htm#tblfn

[12] Mackay S., Park, J: Practical Data Acquisition and Control Systems. Elsevier p. 232 (2003)

[13] Agilent Application Note 1465-29, Using Linux to Control LXI Instruments Through TCP.