Sincronización Microcontroladores Interconectados: Evaluación de Factibilidad y Detalles de Implementación

Fernando G. Tinetti[1] , Ricardo A. López, Sebastián P. Wahler

Departamento de Informática Sede Trelew, Facultad de Ingeniería - UNPSJB
III-LIDI, Facultad de Informática - UNLP

fernando@info.unlp.edu.ar, lopez.ricardoa@gmail.com, sebastian.wahler@gmail.com

RESUMEN

Este proyecto se enfoca directamente en las posibilidades de sincronización de relojes o de algún mecanismo que represente el tiempo en microcontroladores interconectados. Se debe tener en cuenta que el método de sincronización es dependiente del contexto de utilización de la red de microcontroladores para adquisición de datos y control. También se debe tener en cuenta que los microcontroladores no tienen mucha capacidad de cómputo y almacenamiento, con lo que el método de sincronización tiene fuertes restricciones en cuanto al procesamiento disponible en los microcontroladores utilizado para este fin. También es importante remarcar que el problema de sincronización de relojes e incluso de sincronización en general se sigue siendo un área activa de investigación en el contexto de los sistemas distribuidos de computadoras de escritorio o servidores en la actualidad. Este mismo problema en el contexto de microcontroladores interconectados puede aprovechar lo que ya se ha estudiado, pero también tiene restricciones y aplicaciones específicas que son importantes a la hora de evaluar factibilidad en general, implementar y evaluar la o las implementaciones. Parte de los temas de investigación pueden ser utilizados casi directamente para docencia en el contexto de sistemas distribuidos y los temas más avanzados son el objeto de estudio específico para investigación. Dentro de los temas más básicos se puede mencionar la propuesta y análisis de factibilidad básico de algoritmos y/o métodos de sincronización, así como también el análisis del impacto de las redes de microprocesadores (usualmente de bajo rendimiento) sobre estos algoritmos y/o métodos. El resto de los problemas, como la evaluación de la sincronización, tanto analítica como experimental, es de vital importancia en los temas de investigación que se relacionan con los sistemas SCADA (*Supervisory Control And Data Adquisition*).

***Palabras Clave*:** sincronización de relojes, redes de microcontroladores, sistemas distribuidos, sistemas de tiempo real, sistemas SCADA (*Supervisory Control And Data Adquisition*).

1. INTRODUCCION

Las redes de microcontroladores proporcionan una de las plataformas de más bajo costo para la construcción de sistemas distribuidos y la evaluación, al menos experimental, de varios detalles importantes de redes, sistemas operativos, sistemas de tiempo real y subsistemas de sistemas distribuidos en general. De hecho, este proyecto involucra varios aspectos interesantes que de una manera u otra tienen relación con estos temas mencionados. En realidad, este proyecto debe ser tenido en consideración en el contexto del proyecto presentado en [8], que se puede describir esquemáticamente con la Fig. 1. Una de las tareas definidas para el sistema de la Fig. 1, consiste en la adquisición de series de datos, donde cada dato debería tener asociada una *marca de tiempo*

[1] Investigador Asistente, Comisión de Investigaciones Científicas de la Provincia de Buenos Aires

(*timestamp*), para que pueda ser posible crear una secuencia de eventos (o cambio de estado de determinadas señales) asociados al sistema completo más que a un microcontrolador en particular.

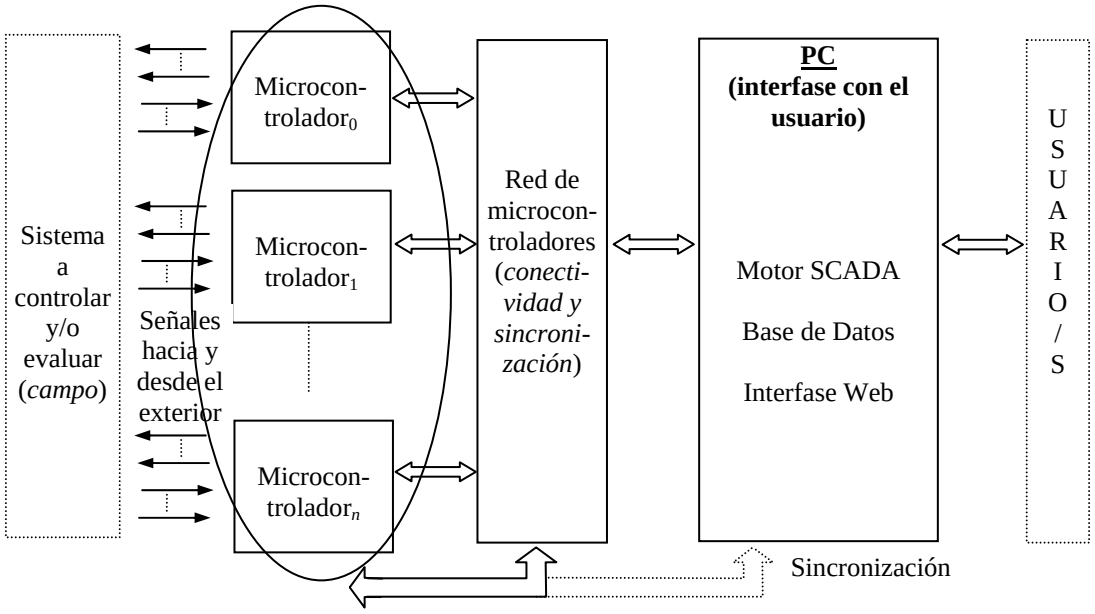

Figura 1: Contexto de Sincronización de Microcontroladores Interconectados.

Por lo tanto, no solamente se debe definir alguna representación de tiempo, sino que además los dispositivos Microcontrolador$_0$, …, Microcontrolador$_n$ de la Fig. 1 (dentro de la elipse) deberían estar sincronizados para que todos los eventos tengan una *referencia de tiempo común.* Esta sincronización hará posible que en el software de interfase con el usuario sea posible presentar la secuencia de eventos independientemente del microcontrolador que haya registrado el evento. De allí que la sincronización tiene impacto en la presentación de los resultados o en la evaluación del estado del sistema que se le presenta al usuario, tal como también lo muestra la Fig. 1 de manera esquemática.

2. LINEAS DE INVESTIGACION Y DESARROLLO

Sin lugar a dudas, una de las primeras líneas de estudio se enfoca al análisis caso por caso de las estrategias de sincronización existentes en el contexto de sistemas distribuidos [6]. Quizás es posible que ninguna de estas estrategias sea factible en el contexto de microcontroladores interconectados, pero al menos provee un punto de partida interesante y fuertemente relacionado con este proyecto específico. En cierta forma, es esperable que haya que llegar a una propuesta de sincronización que por un lado tenga en cuenta las limitaciones de capacidad de los microcontroladores y también aproveche las ventajas de un sistema que podría considerarse de tiempo real estricto, con cotas de tiempo de respuesta a los eventos conocidas de antemano. Esta característica podría considerarse *propia* de los sistemas basados en microcontroladores [1] [2], donde en muchos casos la cantidad de eventos por unidad de tiempo puede acotarse y el software que se ejecuta en los microcontroladores es muy sencillo de analizar y también acotar en cuanto a tiempo de ejecución.

Una vez que se decide utilizar una estrategia de sincronización (sea definida en la bibliografía o propuesto específicamente en el contexto de este proyecto de investigación), es importante el análisis exhaustivo y lo más formal posible desde dos puntos de vista:

- Requerimientos impuestos a los microcontroladores y a la red de interconexión de los mismos. Esto debe tener en cuenta que los microcontroladores no se dedican exclusivamente a la sincronización.
- Cotas de error de sincronización, ya que toda sincronización tiene en sí misma una definición y especificación de error.

Por otro lado, es importante la implementación y experimentación al menos en un entorno de implementación real, aunque sea preliminar de una red de microcontroladores. Como mínimo, deberían haber resultados experimentales para comparar y analizar y en función de los cuales avanzar hacia una implementación probada con la experiencia de uso de los investigadores que se involucren en el proyecto y también de los alumnos que se incorporen en proceso de formación de recursos humanos en el contexto universitario.

Es de vital importancia que los resultados y la experiencia a la que se llegue se puedan reutilizar (o, en cierta forma, *extrapolar*) independientemente de la implementación por varias razones. Por un lado, en el área de docencia no es razonable enseñar el uso de *un* microcontrolador, sino los conceptos asociados a los sistemas embebidos, de tiempo real, etc. y se ejemplifica con una implementación en particular. Por otro lado, es muy difícil que se pueda experimentar y generar resultados de investigación con todas las posibilidades de implementación con microcontroladores. Es por esto que el objetivo es llegar a un diseño paramétrico de un mecanismo de sincronización y una metodología de evaluación, más que a un conjunto de microcontroladores sincronizados. Nuevamente, un conjunto particular de microcontroladores sincronizados dentro de un error acotado será el ejemplo que de alguna manera *corrobore* la factibilidad de la sincronización (suele ser también denominado *proof of concept*).

También es importante estudiar la posibilidad de clasificar las aplicaciones por sus requerimientos de sincronización (básicamente en términos de *precisión*) y de seguridad tanto en comunicaciones (que afectan indirectamente a los mecanismos de sincronización) como en *reactividad* o *respuesta* a determinados eventos (algo usual en el contexto de los sistemas de tiempo real. Actualmente solamente se tiene una idea aproximada de determinados sistemas de control como el caso del transporte eléctrico, control de edificios o semaforización, pero está pendiente un estudio exhaustivo o al menos más formal de tales casos y otros similares.

3. RESULTADOS OBTENIDOS/ESPERADOS

Dado que se espera que los microcontroladores a sincronizar sean de mediana o baja capacidad, se decide una característica importante de la red de interconexión casi directamente *orientada hacia* la posibilidad de sincronización: debe existir la posibilidad de *broadcast* físico de datos. Por un lado, la gran mayoría de las redes físicas más populares cuentan con esta posibilidad, por ejemplo EIA-485 (RS485) [7] y Ethernet [4]. Por otro lado, esta posibilidad permite que haya un controlador que establezca la hora en toda la red con una única transferencia de datos por la red. De manera *relativamente simultánea* todos los microcontroladores recibirían la información relevante o necesaria para la sincronización. Inicialmente, se decide no recurrir a la sincronización externa mediante la utilización de dispositivos como receptores de GPS (*Geographical Positioning System*, que proveen información de sincronización) sino directamente a utilizar un microcontrolador como *maestro* en cuanto a determinar toda la información relacionada con el tiempo de la red. Se prefiere el uso del término *maestro* en vez de *servidor* porque el servidor normalmente *responde a requerimientos* de los clientes, y justamente esto es lo que se tiende a evitar en el contexto de utilización de quizás varias decenas de microcontroladores sencillos a sincronizar. Esto por un lado simplifica el inicio de la elaboración de un prototipo y por el otro deja abierta la posibilidad para

que, sincronizando el *maestro* con una fuente de tiempo externa confiable se sincronice toda la red de microcontroladores casi *inmediatamente*.

Se implementó una red muy sencilla (EIA-485 con control maestro/esclavo) con la capacidad de *broadcast* físico y una representación de tiempo también sencilla. En esta red se implementó una estrategia muy *directa* de sincronización: desde el maestro se indica periódicamente vía *broadcast* la hora (*el tiempo*) de todos microcontroladores de la red. La actualización del *reloj local* (reloj de cada microcontrolador) se realiza de manera absolutamente distribuida durante el intervalo de tiempo entre dos *broadcasts de sincronización*. Si bien no necesariamente es la mejor alternativa, sí permite el análisis y la experimentación de conceptos claves directamente relacionados con la sincronización:

- Evaluación de la *deriva* de los *relojes locales* de los microcontroladores. Dado que la actualización de los relojes locales se lleva a cabo de manera distribuida entre dos *broadcasts de sincronización*, es de esperar que cada reloj difiera de los demás dependiendo de varios factores, y estas diferencias deberían ser conocidas y evaluadas al menos experimentalmente.
- Evaluación de los tiempos involucrados en la asignación del tiempo recibido en los *broadcasts de sincronización*, que pueden resultar en errores de sincronización que deben ser conocidos. Más específicamente, se puede conocer *a priori* el tiempo que implica transferir un *broadcast* a por de la red de interconexión (depende casi exclusivamente de la velocidad de transferencia de la red), pero no necesariamente cuándo este *broadcast* es procesado por el software que se ejecuta en cada microprocesador.
- Definición de un algoritmo de actualización para determinar la frecuencia de los *broadcasts de sincronización* de acuerdo a una determinada cota de error de sincronización.

La Fig. 2 muestra el *pseudocódigo* del software que se ejecuta en cada microcontrolador de la red a sincronizar, que no deja de ser similar a la mayoría del software de control embebido: una iteración con un conjunto de evaluaciones para registrar cambios de estado y procesar *comandos de control*. Lo relacionado con las *tramas* en la Fig. 2 se corresponde al procesamiento de nivel de capa 2 (capa de enlace) del modelo OSI [5] y lo relacionado con los *comandos* en la Fig. 2 se corresponde con el sistema SCADA en general y con el mecanismo de sincronización en particular, dado que uno de los tipos de comandos a procesar son los de sincronización, donde cada uno de los microcontroladores debe asignar la *hora local* con la recibida en el comando.

```
Configurar/Inicializar (puertos, timer/s, UART/s, etc.)
Loop foerever Do
  If (hay un trama a procesar) Then
    Procesar trama
  End if
  If (hay un comando recibido a procesar) Then
    Procesar recepción de un comando
    If (hay una respuesta de comando a procesar) Then
      Preparar la respuesta del comando (Armar el mensaje de respuesta)
      Iniciar la respuesta del comando (Enviar el mensaje de respuesta)
    End If
  End If
  Registrar estado
End Loop
```

Figura 2: Pseudocódigo del Código de cada Microcontrolador a Sincronizar.

En el contexto de establecer una secuencia de eventos, este esquema parece completo una vez que se haga el análisis de error o determinación de cota de error. En cierta forma, la presentación de una secuencia de eventos es *posterior* al registro de los eventos mismos (incluyendo su marca de

tiempo), y esto permite resolver algunos inconvenientes como el de asignar un tiempo local en un microcontrolador previo al tiempo local de cuando se recibió el *broadcast de sincronización*. Este problema en particular se puede solucionar corrigiendo las marcas de tiempo con la interpolación de los valores, quedando de esta manera una sucesión creciente de tiempos para las marcas de tiempo de eventos sucesivos. Sin embargo, otros problemas no son tan sencillos de resolver: si se debe producir un *evento sincronizado a futuro* como por ejemplo que todos los microcontroladores pongan en 1 un determinado *pin a la vez*. Este tipo de problemas y otros relacionados requieren mayor tiempo de análisis y estudio de factibilidad para determinar si es posible su resolución.

Siempre en el contexto de los proyectos de investigación académicos queda abierta la posibilidad de generar proyectos de código abierto (*Open Source*) para sincronización de microcontroladores. En este sentido, el aporte sería importante dado que la gran mayoría de los sistemas SCADA y/o embebidos son propietarios y cualquier propuesta bajo licencia GPL [3] o similar sería interesante y también enriquecedora por la posibilidad de aportes/mejoras externas al propio grupo de investigación.

4. BIBLIOGRAFIA

[1] S. F. Barrett, D. J. Pack, Microcontrollers Fundamentals for Engineers and Scientists, Morgan & Claypool Publishers, 2006, ISBN: 1598290584.

[2] F. M. Cady, Microcontrollers and Microcomputers: Principles of Software and Hardware Engineering, Oxford University Press, 1997, ISBN: 0195110080.

[3] Free Software Foundation, Inc., GNU General Public License, www.gnu.org/copyleft/ gpl.html

[4] Institute of Electrical and Electronics Engineers, Local Area Network - CSMA/CD Access Method and Physical Layer Specifications ANSI/IEEE 802.3 - IEEE Computer Society, 1985.

[5] A. S. Tanenbaum, Computer Networks, 4th Edition, Prentice Hall Ptr, ISBN 0130661023, 2002.

[6] A. S. Tanenbaum, M. van Steen, Distributed Systems: Principles and Paradigms, Prentice Hall, 2nd Edition, ISBN 0132392275, 2006.

[7] Telecommunications Industry Association, “Application Guidelines for TIA/EIA-485-A”, TIA/EIA Telecommunications Systems Bulletin, 1998.

[8] F. G. Tinetti, R. A. López , "Ambiente de Desarrollo y Puesta en Marcha de Sistemas Basados en Microcontroladores", IX Workshop de Investigadores en Ciencias de la Computación, Universidad Nacional de La Patagonia San Juan Bosco (UNPSJB), Trelew, Chubut, Argentina, Mayo 3-4 de 2007.