Universidad Nacional del Centro de la Provincia de Buenos Aires

Trabajo Final de la Cátedra Taller de Microcontroladores

Diseño e implementación de un sistema embebido para control de servomotor vía joysticks

---

Alumno: Sr. Marcelo Rodríguez[1].

Profesores: Mg. Marcelo Tosini.

Ing. José Marone.

Tandil, Marzo de 2013

1 mrodriguez@alumnos.exa.unicen.edu.ar . **LU: 24 52 21**

Resumen

Se presenta el desarrollo de un sistema embebido de tiempo real duro mediante el cual “samplear” valores analógicos que pueden provenir de diferentes fuentes, particularmente palancas de mando. A partir de los valores muestreados se controlan motores servo en direcciones desde cero a 180°. En este contexto el proyecto puede ser de utilidad para movimiento de cámaras, robots, y todo objeto que pueda moverse con fuerza y precisión. Este documento expone someramente la funcionalidad y arquitectura del sistema en cuestión.

1. Sistemas embebidos

Un sistema embebido es una combinación de hardware, software y componentes electromecánicos diseñados para aplicaciones específicas. El software generalmente se ejecuta en un DSP, microcontrolador o FPGA, y es responsable de coordinar todos los componentes para cumplir requerimientos funcionales y no funcionales predefinidos (costo, disponibilidad, consumo etc.). El proyecto utiliza un microcontrolador, este es un IC que consta de CPU, memorias y periféricos de E/S. El software empotrado en estos sistemas puede clasificarse en las siguientes categorías: tiempo real duro y tiempo real suave. Una tarea de software será un proceso o thread. Los sistemas de tiempo real duro garantizan que la ejecución de sus tareas se realice en intervalos de tiempo definido, de lo contrario el resultado de su ejecución no tendrá efecto o será erróneo. Los sistemas de tiempo real suave son menos estrictos respecto al tiempo de ejecución: dentro de todas las posibles tareas simplemente priorizan la ejecución de algunas en particular. Dentro de las etapas de desarrollo de un sistema será en el análisis de los requerimientos iniciales la que determine cuál categoría y configuración de sistema conducirá a la solución final.

2. Arquitectura del sistema controlado vía joystick.

Desde ahora comenzamos una exposición Top-Down. La estructura general del sistema desarrollado: desde los componentes y sus conexiones físicas hasta la organización del software. La siguiente figura es una representación pictórica de los dispositivos involucrados en la implementación del sistema: motores servo, resistencias variables (joysticks), entradas desde un teclado, y el microcontrolador como componente principal.

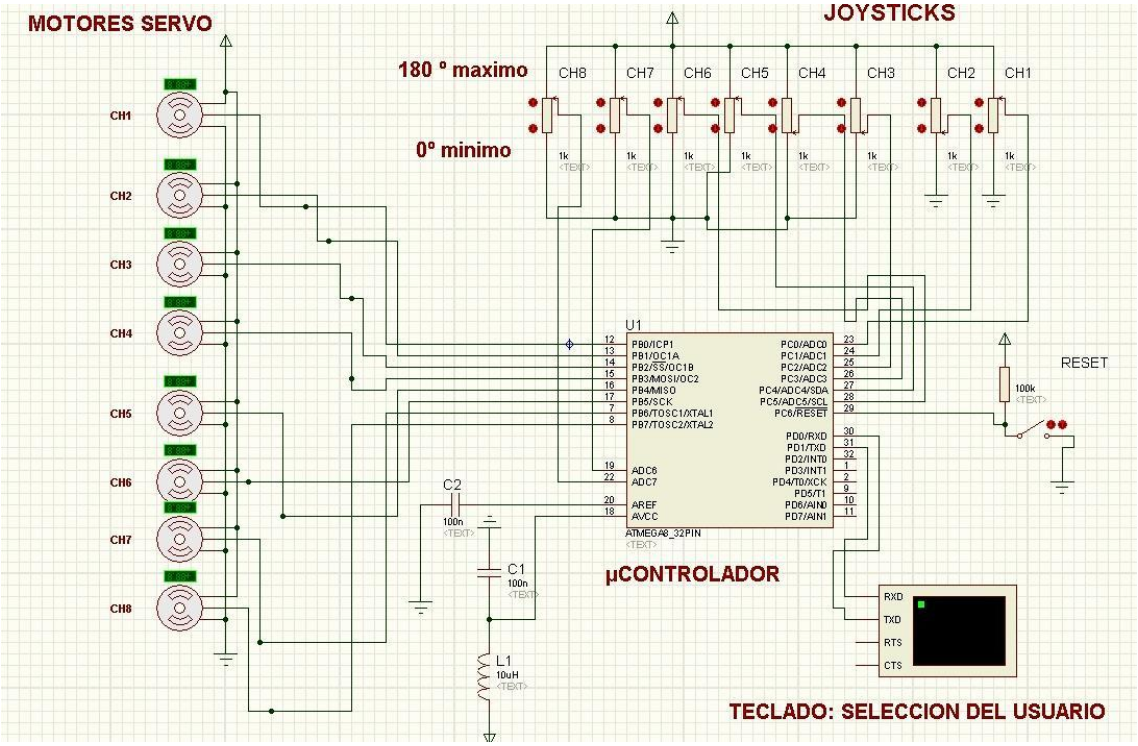


Fig1. Diagrama esquemático del sistema controlador obtenida del simulador Proteus.

2.1. Aspectos tecnológicos

Para clarificar los detalles del software es conveniente presentar algunas características y detalles tecnológicos de los componentes que participan en el sistema. Tales características son necesarias pues se reflejan en la codificación del sistema.

El microcontrolador utilizado es un ATMEL AVR ATMEGA8. Este dispone de 3 timers: 2 de ocho bits y 1 de 16 bits; dos de los cuales tienen características para generar modulación por ancho de pulso, en adelante pwm. El dispositivo también tiene un puerto serie y un conversor analógico digital.

Como dispositivos externos, en el esquemático vemos 2: joysticks y servomotores. Las palancas de mando o joysticks considerados para esta implementación en particular generan valores en el rango 0-5V por cada coordenada o eje. Cada canal del ADC se corresponde con un eje del joystick.

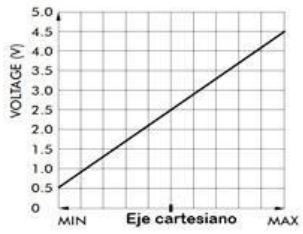


Fig2. Voltaje de salida de una palanca de mando en función de la posición.

Los servomotores tienen la particularidad de poder situarse y mantenerse estables en cualquier posición dentro de su rango de operación (0-180º). Estos motores hacen uso de pwm para controlar su posición o dirección. Esta señal tiene una frecuencia de 50Hz, es decir 20ms, y un ciclo útil de entre 0.9 y 2.1 ms proporcionales a los grados de libertad de dirección del motor (en otras ref. 0.5 y 2.5 ms).

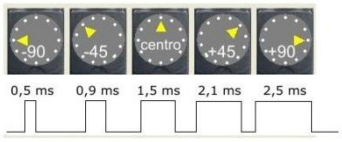


Fig3. Señal pwm aplicada a un motor servo.

El Conversor Analógico Digital realiza una traducción a binario puro. Puede realizar conversiones en forma continua o emitir interrupción por fin de conversión. El voltaje de referencia puede ser interna o externamente configurable. En esta aplicación la frecuencia de muestreo está entre 62.5-125KHz con 10 bit por muestra siendo 200KHz y 10bits la máximas frecuencia y resolución soportada por el dispositivo.

2.2. Vista estructural de los módulos software del sistema.

El siguiente diagrama expone los módulos que forman parte de la solución software con sus métodos y asociaciones más relevantes.

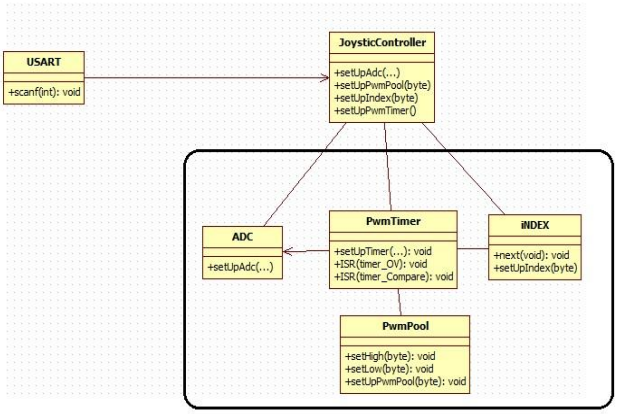


Fig4. El módulo JoystickController actúa como una fachada.

El software que se ejecuta en el microcontrolador está organizado como un conjunto de módulos que representan las abstracciones de los bloques funcionales relevantes al proceso. Estos módulos están adaptados para ser flexibles en sus capacidades. Sus responsabilidades son los siguientes:

- **Usart:** en el diagrama aparece como cliente, pues constituye el evento inicial de comunicación y su responsabilidad es captar la el número de canales que desea manejar el usuario. El valor adquirido sirve como argumento para los métodos de configuración en el resto de los módulos.
- **JoystickController:** provee una interface unificada que facilita el uso otros módulos: inicializar pwmTimer, pwmPool, Adc, Usart, e Index.
- **Adc:** realiza muestreo sobre una señal analógica de entrada (voltaje) y la traduce a un binario proporcional. Este módulo permite ajustar dinámicamente la cantidad de canales de entrada, resolución de las muestras, y velocidad (vía prescaler) en función de la velocidad de CPU del microcontrolador. Por defecto se toman muestras de 10 bits.
- **Index:** establece la entrada de ADC a muestrear. El beneficio de tener este módulo reside en la implementación de políticas de movimiento sobre los canales. Ej.: muestrear entradas secuencialmente, intercaladas, solo canales impares, etc. Por defecto muestrea los canales secuencialmente.
- **PwmPool:** esta abstracción representa un conjunto de señales pwm y sus estados. El estado "alto" de los bits será el ciclo útil de una señal pwm. PwmPool configura tantos bits de salida de un puerto de E/S del microcontrolador como canales (joysticks) se especifica al iniciar el sistema.
- **PwmTimer:** realiza la multiplexión por división de tiempo de una señal pwm. Para lograrlo divide el periodo de la señal en umbrales. Al completar una ventana de tiempo este módulo básicamente activará una salida mediante pwmPool durante tiempo proporcional al valor obtenido desde ADC. Este módulo se adapta dinámicamente a diferentes frecuencias de funcionamiento del CPU, de esta forma garantiza que si tanto el micro funciona a 1, 8 y 16 MHz sus operaciones tendrán una duración constante (tics de 1µs).

2.3. Vista dinámica e implementación del sistema.

En esta sección se especifica la lógica implementada en los módulos **ADC** y **PwmTimer** para controlar los motores servo mediante multiplexión por división de tiempo generada desde el microcontrolador. Finalmente hay un diagrama dinámico del sistema con intención de exhibir las relaciones temporales entre los módulos, actividades y eventos que surgen durante la ejecución del sistema e imágenes de simulación del sistema.

2.3.1. Multipliexación por división de tiempo y control de motor servo.

2.3.1.1. Implementación de Multiplexación por división de tiempo.

De la sección 2.1 se conoce la funcionalidad relevante en este contexto de los motores servo, y las características de la señal pwm. De las capacidades del IC del microcontrolador en el esquemático se ve que pueden conectarse hasta ocho ejes y los correspondientes motores servo. La idea de es esta multiplexión consiste en dividir en ranuras de 2.5ms de tiempo el periodo de 20ms de la señal pwm. Para cada canal podemos considerar que cada ranura de tiempo es un ciclo útil y en consecuencia el movimiento de un motor, el resto de las ranuras representan el ciclo en off de la señal recibida.

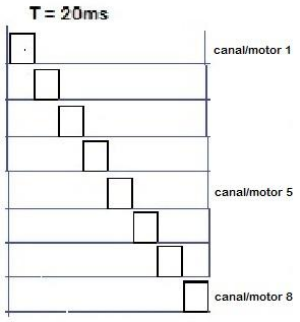


Fig. 5. Multiplexión por división de tiempo en 8 canales. El estado alto de la señal es el ciclo útil y dura un máximo de casi 2.5 ms.

Implementar este funcionamiento requiere que un timer calcule la duracion de las ranuras de tiempo y emita una interrupcion por overflow cuando reinicia la cuenta de cada umbral de teimpo, eventualmente **PwmTimer** solicitará al módulo **Index** por la siguiente entrada desde **ADC**. El flanco ascendente y descendete de cada ciclo util se plasma en el software de **PwmPool**, este implementa la logica asociada a una interrupcion por la comparación exacta entre el sampleo desde el **ADC** con el conteo del timer.

Una vez el sistema en ejecucion, en este punto se destaca:

- El unico thread de control activo en el microcontrolador es el que se ejecuta en el modulo **PwmTimer**.
- **La solucion software implementada sigue a un sistema de tiempo real duro**, pues los motores servo requieren recibir su ciclo util de manera periodica y sin excepción; ademas lograr una division de tiempo presicisa para cada canal requiere respetar la longitudes de tiempo de cada umbral de forma precisa.

2.3.1.2. Implementación del Control del motor servo.

Aquí tenemos dos cuestiones.No se puede mover con precisión un motor llevando directamente el valor digital obtenido del ADC para comparar contra el conteo del timer. Por otro lado, se debe escoger el timer mas adecuado de entre los disponibles.

El ciclo util de la señal pwm debe ser de entre 0.9ms a 2.2ms. Las ranuras de tiempo que dividen el periodo de la señal son de 2.5ms. La configuracion de frecuencia de conteo del timer según **PwmTimer** se acomoda dinamicamente para tics de 1µs. De esta, 0.9ms y 2.2ms son el 36% y 88% de la duracion de cada ranura de tiempo. De esta forma, se tiene 52% de cada ranura de tiempo para grados de libertad de movimiento del servomotor lo cual equivale a 0.52*2500 = 1300 posiciones diferentes para mapearse en los 180° de movimiento completo del motor. En consecuencia, es conveniente escoger el timer de 16 bits.

Para lograr movimientos de motor correctos hay que considerar que el valor mínimo tomado desde ADC para indicar un movimiento a 0° será cero y se debe convertir a 1600 tics del timer; el maximo, 1024, para mover a 180° y se debe convertir a 300 ciclos del timer. Este factor de conversion está dado por una relacion lineal y será aplicado por **PwmTimer** al momento de iniciar una ranura de tiempo.

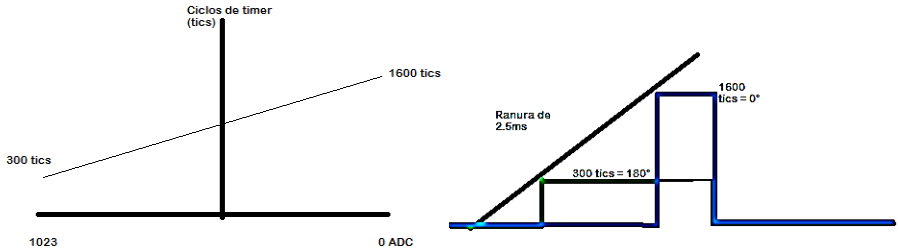


Fig 6. Relacion lineal entre el valor muestreado por ADC y el factor de corrección para generar la pwm: los pulsos mas cortos corresponden con valores mas altos de comparacion.

2.3.2. Diagrama de actividades y simulación del sistema.

En el diagrama siguiente, el orden cronológico de las tareas que se ejecutan está dado por la secuencia de transiciones. Las lineas verticales y horizontales denotan el dispositivo responsable de la actividad actual. Aunque trivial, el encuadre horizontal hace explicito que en cualquier dispositivo/módulo todo actividad siempre será ejecutado por la MCU del microcontrolador. Si no hay reset, las actividades se ejecutan en un ciclo infinito.

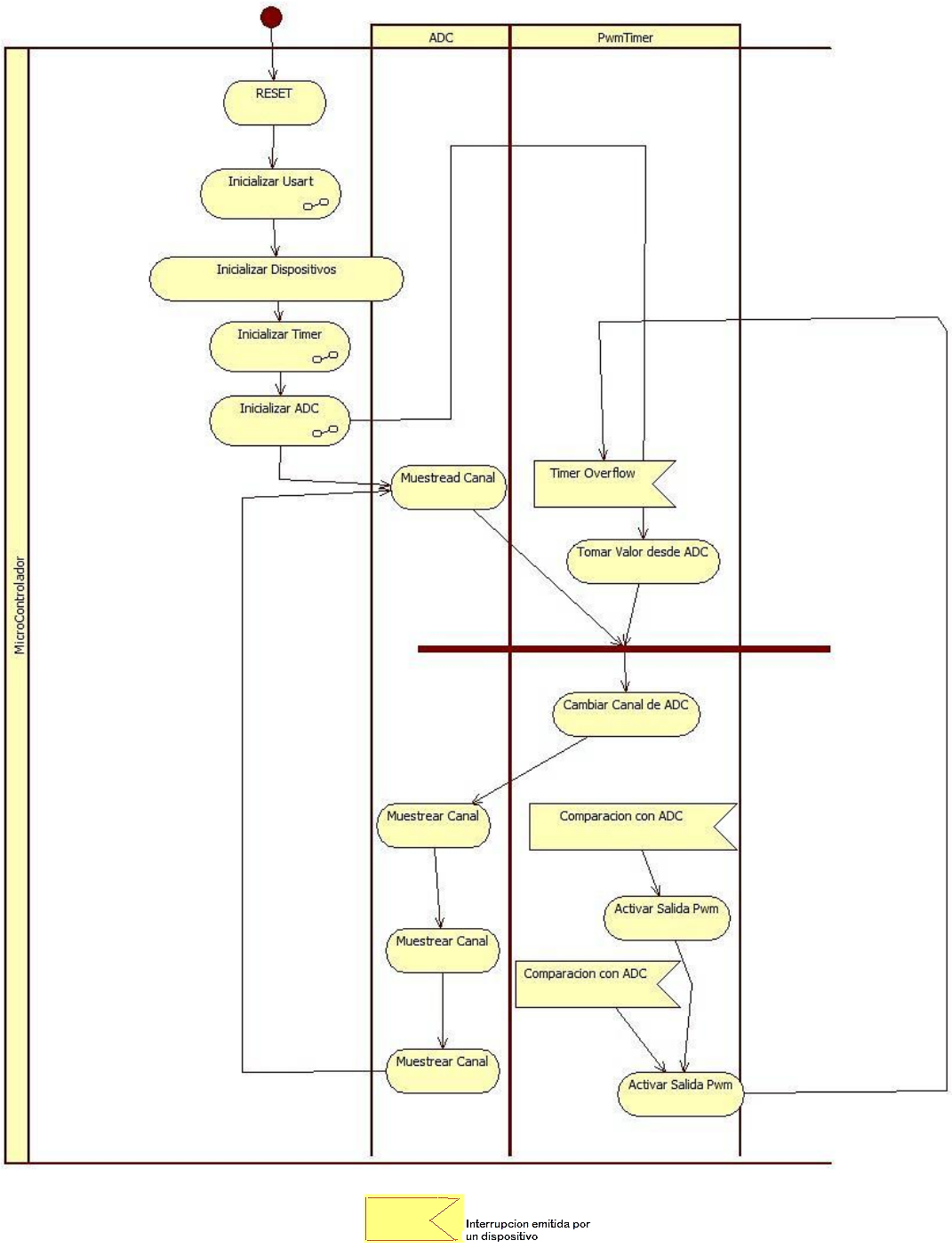
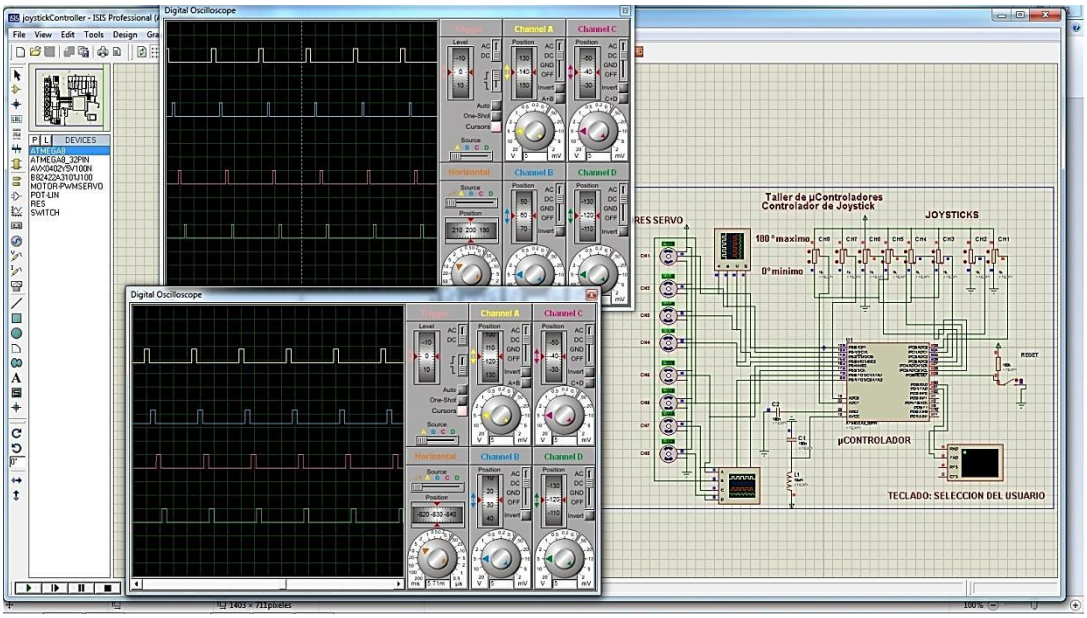


Fig 7. Orden cronológico y dispositivo/módulo responsable de la ejecución.



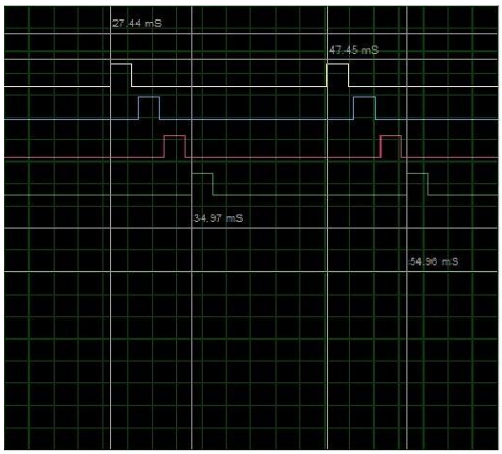


Fig 8. En esta figura se vé que la señal tiene un periodo de 20ms.

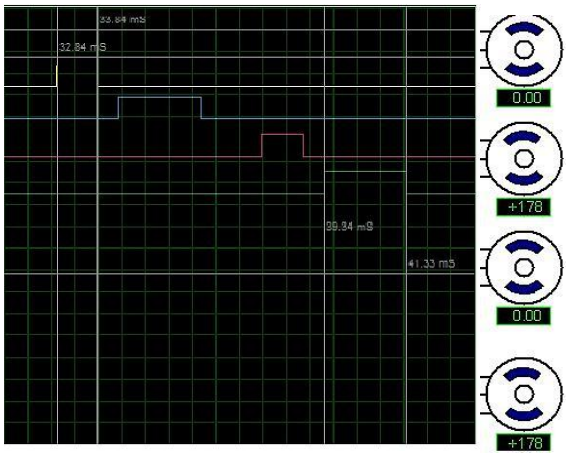


Fig. 9. Arriba un pulso mínimo de 1ms, abajo el máximo de 2ms. A la derecha los motores correspondientes.

3. Conclusión.

Se ha desarrollado y testeado un sistema embebido de tiempo real en un lenguaje de alto nivel. Esta experiencia puso en evidencia la dependencia entre las aplicaciones posibles y el criterio de selección de un microcontrolador en función de sus capacidades (ej. pwm hardware). También se ha hecho presente la dificultad de la tarea de testing de los sistemas embebidos: testear en hardware, en simulación y testear en software; diferente de aplicaciones específicas que se ejecutan en sistema de propósito general. Finalmente es importante destacar el trade-off entre el tiempo de ejecución de los microcontroladores pequeños y su efecto en la modificabilidad del software; sin embargo en un sistema de tiempo real duro las consecuencias de conocer y diseñar cada uno de los procesos que se ejecutan en el sistema da la libertad de mejorar tales restricciones.

Webgrafía

Atmega8 datasheet: http://www.atmel.com/Images/Atmel-2486-8-bit-AVR-microcontroller-ATmega8_L_datasheet.pdf