ISSN 1870-4069

Efectos en la resolución de servomotores con interfaz PWM por la generación de señales en microcontroladores

Miguel Ángel Castillo-Martínez[1], Blanca Esther Carvajal-Gámez[2], Francisco Javier Gallegos Funes[1]

[1] Instituto Politécnico Nacional,
Escuela Superior de Ingeniería Mecánica y Eléctrica, Unidad Culhuacán,
México

[2] Instituto Politécnico Nacional,
Escuela Superior de Cómputo,
México

mcastillom1503@alumno.ipn.mx

Resumen. Se realiza el análisis para la generación de señales mediante el microcontrolador ATmega328P con el objetivo de programar una interfaz con servomotores controlados por modulación de ancho de pulso. El análisis está basado en el uso de los temporizadores y algunos de sus modos de operación, además de realizar observaciones en los casos de su configuración y sus posibles efectos en la posición del eje del servomotor. Se aborda la teoría de operación del servomotor y su relación matemática para obtener un modelo que ayude en la debida gestión de parámetros y su correcta programación en el microcontrolador, logrando una operación del servomotor basada en los recursos de hardware contenidos en el dispositivo programable utilizado para la generación de la señal. Una simulación muestra como el código implementado genera las señales y cómo se comportan para la operación de los dispositivos aquí tratados.

Palabras clave: modulación de ancho de pulso, servomotor, microcontrolador.

Microcontroller Signal Generation Effects in PWM Controlled Servomotor Resolution

Abstract. In this work, an ATmega328P microcontroller-based signal generation analysis is presented to develop a Phase Width Modulation Servomotor interface. The analysis was made of timers and some of its operation modes, configuration cases and effects over servomotor axis were observed as well. An operation theory and its mathematical relation can be seen to obtain a model that helps in the correct parameter management, in addition to right microcontroller programming to operate servomotors with hardware resources contained in the programmable device used to generate the signals. A simulation shows how the implemented code produces signals and which operating behavior has in the devices.

Keywords: Pulse width modulation, servomotor, microcontroller.

1. Introducción

Uno de los lenguajes más utilizados para el diseño de sistemas embebidos es C, cuyo objetivo es generar el mayor código necesario en la capa de aplicación para simplificar la portabilidad a otras plataformas [1,2,3]. C provee un control aceptable y acceso a funciones de bajo nivel que, generalmente, contiene controladores necesarios para acceder a características específicas de recursos de hardware. Un sistema embebido, regularmente, contiene un microcontrolador para procesar entradas y salidas, generando un enlace o interfaz entre entradas y salidas de sistemas mediante un algoritmo codificado y almacenado en la memoria del sistema (firmware) [4].

Los ATmega AVR son una serie de microcontroladores que fueron diseñados para aplicaciones que requieren una gran cantidad de código, teniendo una memoria Flash de 4KB hasta los 512KB de acuerdo con la demanda del firmware. Estos dispositivos vienen empaquetados desde 28 terminales (ATmega 328P) hasta 100 terminales (ATmega 2560), su arquitectura cuenta con una serie de sistemas embebidos como convertidores analógicos a digitales, comunicaciones, temporizadores, etc. haciéndolos ideales para la comunicación entre numerosos dispositivos periféricos [2,3,5].

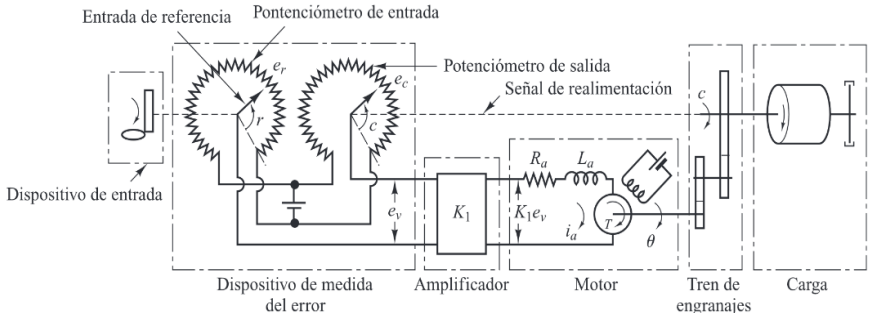

Fig. 1. Diagrama general de un servomotor (Fuente: [6]).

Los actuadores más comunes en robótica básica son los servomotores. Un servomotor consiste en un motor de corriente directa, controlado por un sistema embebido interno, el cual transfiere la energía mecánica mediante una serie de engranes hasta un eje externo. El sistema embebido interno se encarga de comparar una posición angular, determinada por la señal de control, y la posición actual del eje externo, un diagrama muy consultado se encuentra en [6] que, como se muestra en la figura 1, se ilustran de forma general los componentes de un servomotor.

Entre la gama de posibilidades en servomotores nos encontramos con aquellos que cuentan con una interfaz por Modulación de ancho de Pulso (PWM) debido a su costo reducido en comparación de aquellos con interfaces más avanzadas. La señal de PWM debe cumplir una serie de parámetros (ver figura 2). Para utilizar este tipo de servomotores, comúnmente, los parámetros frecuencia de funcionamiento f_{PWM}, ancho de pulso PW operativo, tiempo muerto T_D, tiempo central T_C y ángulo de operación θ

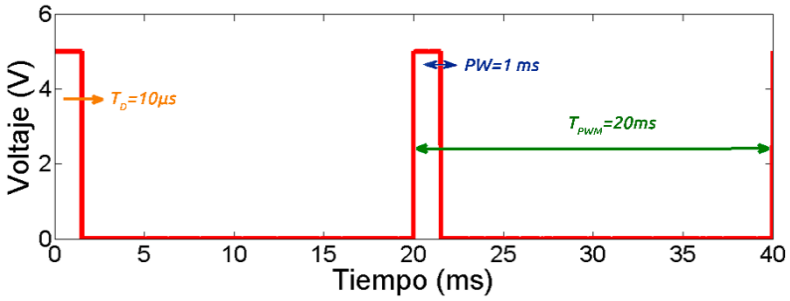

Fig. 2. Oscilograma común de señal para uso de servomotor controlado por PWM.

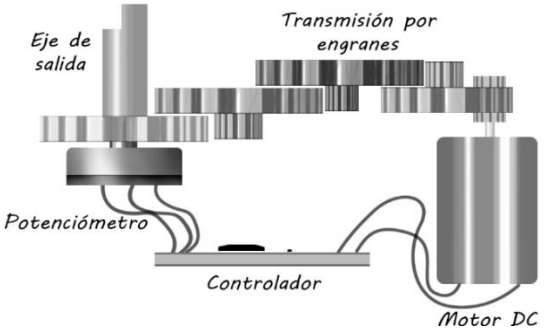

Fig. 3. Estructura interna de un servomotor.

tienen los siguientes valores [7,8]: $f_{PWM} = 50\ Hz$, $PW = 1\ ms$, $T_C = 1.5\ ms$, $T_D = 10\ \mu s$, $\theta = 180°$.

En el uso de estos servomotores como periféricos es común utilizar tarjetas de desarrollo basadas en microcontroladores o FPGA. Para hacer el análisis del funcionamiento estimado en un servomotor controlado por PWM se requieren tomar reglas de funcionamiento y sus analogías con respecto a la arquitectura del microcontrolador utilizado.

Comúnmente, para un servomotor con interfaz PWM, la posición angular se determina mediante un potenciómetro que gira de manera concéntrica con el eje externo del motor, como se muestra en la figura 3 [5,7].

2. Metodología

Para obtener una buena respuesta se requiere que el generador de señal, en nuestro caso los temporizadores del microcontrolador generen cambios mínimos de paso ΔT iguales o inferiores al T_D, si y solo si busca una buena resolución en la posición del servomotor, la cual se obtiene mediante ecuación (1):

$$R_P = \frac{\theta \times T_D}{PW}. \tag{1}$$

Considerando los datos anteriores y $\Delta T = T_D$ se obtiene una resolución de 1.8° por paso de 10 µs para el servomotor. Sin embargo, se busca generar una señal cuyo valor dependa de la posición angular deseada en el servomotor, esta relación se determina mediante la ecuación:

$$\mu s_{PWM} = \frac{PW}{\theta} \times (\theta_D - \theta_{min}) + \mu s_{min}\,, \tag{2}$$

donde μs_{PWM} indica el ancho de pulso que se necesita generar para una determinada f_{PWM}, los subíndices *min* y *D* indican, dentro de su rango, el mínimo y deseado respectivamente. Estos parámetros se determinan con:

$$\mu s_{min} = T_C - \frac{PW}{2}\,, \tag{3}$$

$$\theta_{min} = \theta_C - \frac{\theta}{2}. \tag{4}$$

Si el ángulo está centralizado, se considera que θ_{min}=-90° con un θ =180°, pero si se cuenta con un ángulo no centralizado y las características antes mencionadas a ecuación (3) se le asigna el ancho de pulso mínimo para operar el servomotor y ecuación (4) se reduce a 0, por lo que la ecuación (2) se reduce a lo siguiente:

$$\mu s_{PWM} = \frac{PW}{\theta} \times \theta_D + \mu s_{min}. \tag{5}$$

2.1. Temporizador de 16 bits

Considerando la arquitectura del microcontrolador Atmega328P, se cuenta con un temporizador de 16 bits, el cual se utilizará en el modo de operación Fast PWM. El Modo Fast PWM (modos 5, 6, 7, 14 y 15) provee una opción para la generación de señales PWM a alta frecuencia. Difiere de otros modos de PWM por su operación de pendiente única. El contador cuenta de mínimo (BOTTOM) hasta un límite (TOP) y reinicia en BOTTOM. En el modo de salida no invertida, el estado lógico en la terminal OCRnx se desactiva cuando TCNTn y OCRnx son iguales y se activa en el desbordamiento de conteo. En el modo invertido OCRnX es activada cuando TCNTn y OCRnx son iguales y se desactiva en el reinicio de conteo, estos fenómenos se observan en la figura 4.

Debido a la operación de pendiente única en el Modo Fast PWM, la frecuencia de operación puede ser dos veces mayor que en los modos PWM. La alta frecuencia hace al modo Fast PWM una buena opción para regulación de potencia, rectificación y aplicaciones de conversión digital a analógica [9].

La frecuencia de salida $f_{OCnxPWM}$ puede ser calculada mediante.

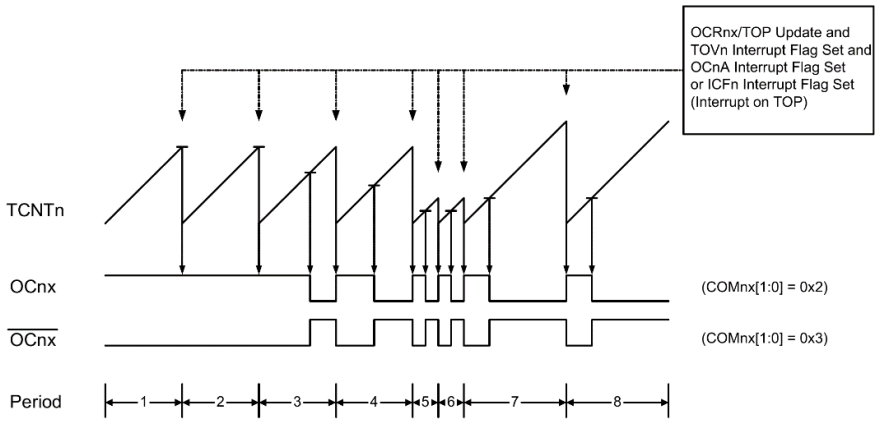

Fig. 4. Diagrama de tiempos del modo Fast PWM del temporizador de 16 bits.

$$f_{OCnxPWM} = \frac{f_{CLK}}{N \times (1 + TOP)}, \tag{6}$$

donde TOP es el límite superior de conteo, de manera nativa es un registro de 16 bits, sin embargo, este límite puede ser ajustado a 8, 9 o 10 bits, incluso puede ser definido por los registros ICR1 u OCR1A. El valor de TOP se obtiene a partir de ecuación (6) realizando se respectivo despeje obteniendo a ecuación (7).

$$TOP = \frac{f_{CPU}}{N \times f_{OCnxPWM}} - 1\,. \tag{7}$$

Considerando una frecuencia de operación f_{CPU} de 8 MHz, por su oscilador interno removiendo la programación del fusible CLKDIV del microcontrolador, y ajustando el divisor de frecuencia N = 8 de tal forma que no desborde el registro de 16 bits, el límite del contador TOP se iguala a 19,999 o 0x4E1F de forma hexadecimal.

El límite obtenido, al no ser un número definido por defecto en el modo de operación, se requiere una calibración mediante algún registro que permita generar la f_{PWM} con el valor antes obtenido. Los modos 5 a 7 no nos permiten ajustar el límite para la generación de la señal deseada. El modo 15 nos permite realizar un ajuste para generar la señal deseada, pero al ser configurado en el registro OCR1A se perdería un canal para controlar algún dispositivo derivado que TCNT1 se compara continuamente con el registro asignado a TOP, por lo que, si el valor de OCR1A cambia, f_{PWM} cambia también. Por último, el modo 14 nos permite ajustar el límite sin perder algún canal de la generación de PWM, considerando que cada temporizador posee dos salidas cuyo ciclo de trabajo puede ser ajustado por OCR1x y su $f_{OCnxPWM}$ calibrada con ICR1. Con base a lo anterior se considera que al ajustar OCR1x será equivalente a:

$$\mu s_{PWM} = \frac{N}{f_{CLK}} \times (OCR1x + 1)\,, \tag{8}$$

para obtener el valor que debe ser escrito en el registro OCR1x se igualan las ecuaciones (5) y (8).

$$\frac{PW}{\theta} \times \theta_D + \mu s_{min} = \frac{N}{f_{CLK}} \times (OCR1x + 1)\,, \tag{9}$$

despejando OCR1x de la ecuación (9):

$$OCR1x = \left(\frac{PW}{\theta} \times \theta_D + \mu s_{min}\right) \times \frac{f_{CLK}}{N} - 1\,, \tag{10}$$

minimizando con los parámetros del servomotor definidos anteriormente la ecuación (10) queda de la siguiente manera:

$$OCR1x = \left(\frac{\theta_D}{180°} + 1\right) 1000 - 1\,. \tag{11}$$

Con la ecuación (11) se elige un ángulo centralizado obteniendo un valor de 1499 que le es asignado al registro OCR1A para generar 1.5 ms de ancho de pulso cada 20 ms. Las consideraciones anteriores solo funcionan para una salida no invertida. Comentado lo anterior, al registro ICR1 asigna el valor TOP calculado con (7), el registro TCCR0A debe ser configurado con una salida no invertida, junto a TCCR0B, ajustar el modo de operación 14 y un divisor de frecuencia de 8, sin olvidar asignar las terminales correspondientes a OCR1x como salida, quedando de la siguiente manera:

```
// Configuración de terminales OC1x como salidas
DDRB = 0x06;
// Configuración de salida no invertida
// y junto a TCCR2B un modo de operación Fast PWM (14)
TCCR1A = 0xA2;
// Selección de fuente de temporización con un divisor
// de 8
TCCR1B = 0x1A;
// Ajuste del límite de conteo
ICR1 = 0x4E1F;
// Posición en ángulo central en terminal 1
OCR1A = 1499;
// Posición en ángulo final en terminal 2
OCR1B = 1999;
```

El cambio mínimo en la generación de señal es considerado cuando OCR1X = 0 se obtiene con la siguiente ecuación:

$$\Delta T = \frac{N}{f_{CLK}}\,, \tag{12}$$

donde ΔT= 1µs. Por lo que $\Delta T \leq T_D$, cumpliéndose la condición para obtener la resolución del servomotor inalterada, es decir, al generar la señal en el microcontrolador no habrá una pérdida de resolución en la posición del servomotor, además de no generar un desfase en la posición al generar los 1000 µs requeridos para el ángulo inicial.

2.2. Temporizador de 8 bits

Además del temporizador de 16 bits se cuenta con un par de temporizadores de 8 bits cuyo diagrama de tiempos se muestra en la figura 5, para lo sucesivo se utiliza el temporizador 0. Comparando las unidades de temporización, se observa que el temporizador 0 tiene menos modos de operación que el temporizador 1, además de que el valor de límite solo puede ser calibrado de forma abierta con el registro OCR0A, significando la pérdida de un canal de interfaz. Las ecuaciones determinadas para el temporizador de 16 bits son funcionales para el temporizador de 8 bits con la diferencia que los valores determinados son inferiores al tener menor resolución.

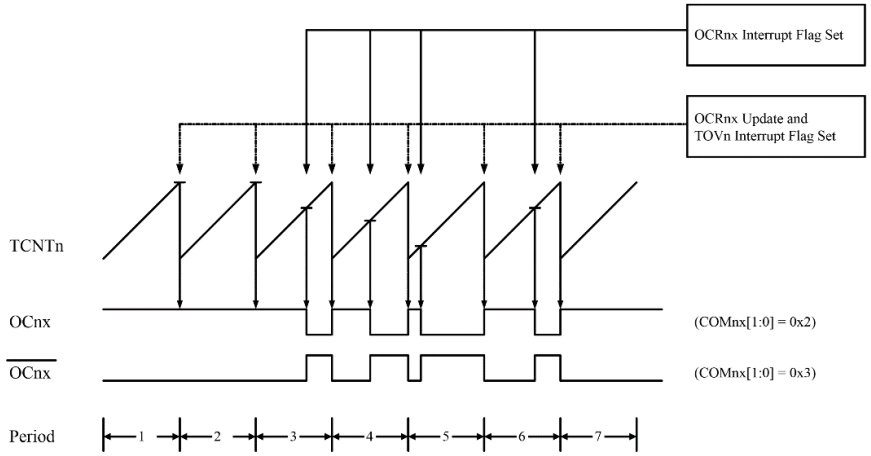

Fig. 5. Diagrama de tiempos del modo Fast PWM para el temporizador de 8 bits.

Ajustando N=1024 de tal forma que no desborde la resolución de 8 bits, el parámetro TOP, calculado con la ecuación (7) queda igual a 155.25 el cual no es un valor entero, caracterizado por el registro OCR0A para su funcionamiento, por lo que es necesario truncar y asignar como 0x9C de forma hexadecimal.

El ΔT calculado con los nuevos parámetros de acuerdo con la ecuación (12), es de 128μs por lo que no se cumple la condición de ser menor al tiempo muerto del servomotor. Al no lograr que el cambio mínimo generado sea menor que el tiempo muerto, la resolución del servomotor con el temporizador 0 con base en la ecuación (1) es de 23.04°.

Para posicionar en el ángulo inicial θ_i se requiere saber el registro mínimo, el cual se obtiene mediante:

$$OCR0B = Truncar\left(\frac{\mu s_{min}}{\Delta T}\right). \tag{13}$$

El registro OCR0B obtenido con (13) es de 7 por lo que, de acuerdo con la ecuación (10), El ángulo inicial θ_i quedaría como sigue:

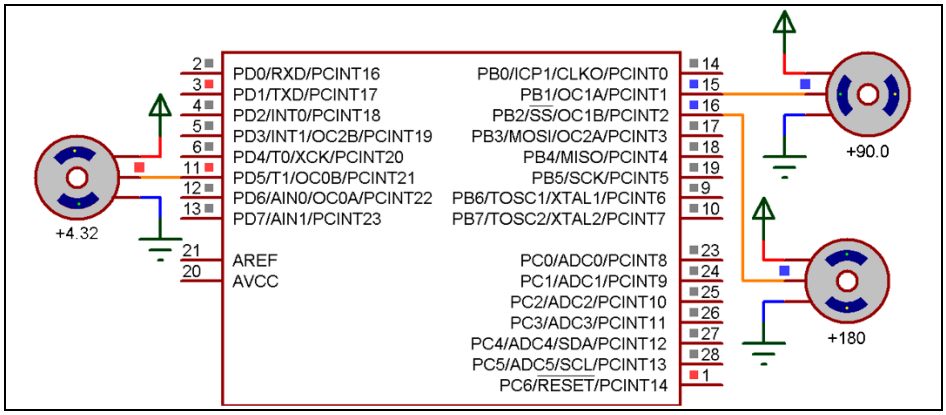

Fig. 6. Simulación activa en el software ISIS de Proteus.

$$\theta_i = \left(\frac{(OCR0B + 1) \times N}{f_{CLK}} - \mu s_{min}\right)\frac{\theta}{PW}. \tag{14}$$

Con la ecuación (14) se obtiene un θ_i=4.32°, un último paso a realizar para determinar el ángulo inicial real θ_{iq} es cuantificar el resultado obtenido con θ_i derivado que en los múltiplos de la resolución del servomotor no se encuentra este valor. Para determinar θ_{iq} de acuerdo con la resolución del servomotor se tiene lo siguiente:

$$\theta_{iq} = Redondear\left(\frac{\theta_i}{R_P}\right) \times R_P. \tag{15}$$

Obteniendo un ángulo inicial de 3.6°. Con base en el código generado para el temporizador 1 y los nuevos datos determinados, para colocar el servomotor en la posición inicial con el temporizador 0 se requiere el siguiente código.

```
// Configuración de OC0B como salida
DDRD = 0x20;
// Configuración de una salida no invertida para OCR0B
// y junto a TCCR0B un modo de operación Fast PWM (7)
TCCR0A = 0x23;
// Selección de fuente de temporización dividida por 1024
TCCR0B = 0x0D;
// Ajuste de límite de conteo
OCR0A = 0x9C;
// Posición ángulo inicial
OCR0B = 7;
```

3. Resultados

Las ecuaciones son programadas en el microcontrolador y su comportamiento básico puede ser observado en el simulador ISIS de Proteus en el cual se configuran algunos parámetros del servomotor como es el ángulo y anchos de pulso de operación (ver

figura 6), este simulador se popularizó por su amplia gama de componentes que cuentan con un perfil de simulación y su fácil uso. En la figura 6 se observa la representación del circuito y su simulación donde el comportamiento de la configuración de los registros OCRnx se observa en las respectivas terminales OCnx.

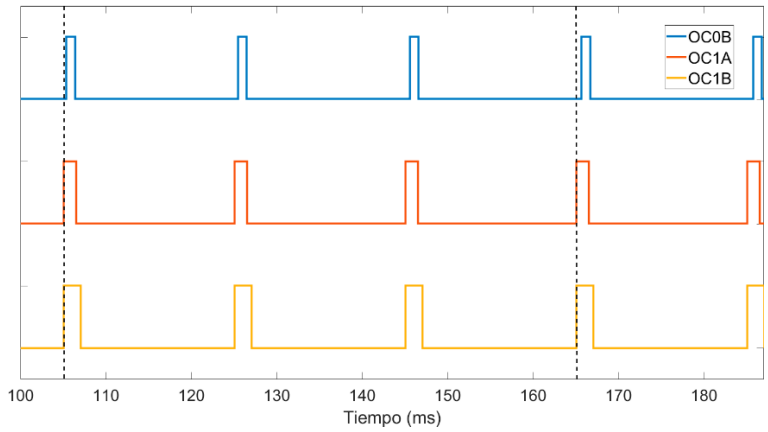

Fig. 7. Oscilograma de operación de los temporizadores.

La simulación muestra los ángulos iniciales seleccionados sin cuantificación aun el ángulo θ_i calcula con la ecuación (15). Por otra parte, los oscilogramas de la figura 7 muestran un desfase en la señal OC0B con respecto a los canales OC1A y OC1B (líneas discontinuas de referencia). El desfase se debe al error de truncamiento realizado con la ecuación (13) cuya $f_{OCnxPWM}$ calculada es de 50.08 Hz indicando un error de 0.16%, siendo viable utilizarlo para generar la frecuencia de operación. Sin embargo, no es suficiente para tener una resolución aceptable en el movimiento del servomotor.

En los oscilogramas se observan las señales generadas por el microcontrolador, la señal OCR0B responde a la configuración del temporizador 0 y las señales OCR1x responden al temporizador 1.

4. Conclusiones

El uso incorrecto de los recursos disponibles en sistemas integrados como los microcontroladores conduce al mal uso de periféricos o incluso su inutilidad para una aplicación especifica. Además de las consideraciones teóricas comentadas se deben tomar en cuenta los fenómenos físicos como son las variaciones de las fuentes de reloj y potencia requerida para un buen funcionamiento del microcontrolador, el servomotor y demás periféricos.

Las ecuaciones descritas pueden ser ajustadas de acuerdo con servomotor y dispositivo programable a utilizar, sin embargo, se deben considerar las capacidades del hardware para evitar una mala configuración, lo que conduciría un daño interno de los dispositivos.

El uso de este tipo de métodos reduce el tiempo de diseño, principalmente el error por un modelo sin definir completamente o incorrecto, lo que conduce desde una mala

codificación hasta el daño permanente de los dispositivos involucrados. Además, las ecuaciones aplican a las variaciones de servomotores con interfaz PWM donde los anchos de pulso para controlar el dispositivo varían.

Referencias

1. Valvano, J.W.: Embedded Microcomputer Systems: Real Time Interfacing. Cengage Learning (2012)
2. Grace, T.: Programming and Interfacing ATMEL ® AVR ® Microcontrollers. Cengage Learning (2016)
3. Russell, D.: Introduction to Embedded Systems: Using ANSI C and the Arduino Development Environment. Morgan & Claypool (2010)
4. Barrett, S.F.: Embedded Systems Design with the Atmel AVR Microcontroller: Part I. Morgan & Claypool (2010)
5. Langbridge, J.A.: Arduino™ Sketches. Tools and Techniques for Programming Wizardry, Wiley (2015)
6. Ogata, K.: Ingeniería de control moderna, 5ª edición. Pearson, Madrid (2010)
7. Pinckney, N.: Pulse-width modulation for microcontroller servo control, IEEE Potentials, 25(1), pp. 27–29 (2006)
8. Chu, P.P.: Pulse Width Modulation Core, FPGA Prototyping by VHDL Examples: Xilinx MicroBlaze MCS SoC, 2nd Edition, pp. 297–308 (2017)
9. Atmel: ATmega328 / P, AVR Microcontrollers, Atmel Corporation, pp. 442 (2016)