Artículo **Revista de Aplicaciones de la Ingeniería**
Septiembre, 2019 Vol.6 No.20 18-25

Secuenciador dinámico para el control de movimiento de robot hexápodo en una arquitectura FPGA

Dynamic sequencer for hexapod robot motion control in an FPGA architecture

IBARRA-BONILLA, Mariana Natalia†*[1], SÁNCHEZ-TEXIS, Fernando[1], EUSEBIO-GRANDE, Raúl[1] y QUIÑONES-NOVELO, Fernando Julián[2]

[1] *Instituto Tecnológico Superior de Atlixco, División de Ingeniería Mecatrónica*
[2] *Intesc Electrónica & Embebidos, Dirección de Investigación*

ID 1er Autor: *Mariana Natalia, Ibarra-Bonilla* / **ORC ID:** 0000-0001-7123-9105, **CVU CONACYT ID:** 237756

ID 1er Coautor: *Fernando, Sánchez-Texis* / **ORC ID:** 0000-0002-1792-8855, **CVU CONACYT ID:** 95289

ID 2do Coautor: *Raúl, Eusebio-Grande* / **ORC ID:** 0000-0001-7062-3244, **CVU CONACYT ID:** 1000358

ID 3er Coautor: *Fernando Julián, Quiñones-Novelo* / **ORC ID:** 0000-0001-8964-1039, **CVU CONACYT ID:** 233888

DOI: 10.35429/JEA.2019.20.6.18.25 Recibido: 19 de Junio, 2019; Aceptado Septiembre, 30, 2019

Resumen

Se presenta la arquitectura en hardware de un sistema de control para un robot-hexápodo. El sistema de control está integrado en un FPGA XC6SLX16-Spartan-6. El diseño es realizado en lenguaje de descripción de hardware VHDL. Una arquitectura robusta, dividida en tres bloques lógicos de máquina de estados, se implementa en el FPGA. El bloque uno consiste en un sistema de comunicación serial para el intercambio de información entre el usuario y el FPGA. El segundo bloque corresponde al secuenciador y gestor de datos que controla las funciones de toda la arquitectura. El tercer bloque es un generador de pulsos PWM de 18 canales con un control de cambio de ciclo de trabajo progresivo para realizar los movimientos del robot. El sistema está diseñado para permitirle al usuario descargar una secuencia de movimientos complejos mediante la combinación de diferentes posiciones y tiempos de espera. La arquitectura permite guardar hasta 10 diferentes secuencias de movimientos en su bloque de memoria interna. Adicionalmente, permite modificar cualquier secuencia en tiempo de ejecución, sin que se afecte el funcionamiento del robot. La arquitectura controla 18 servomotores permitiendo 18 grados de libertad al robot. Se presentan las pruebas de funcionamiento y movimiento del robot.

Control, Hexápodo, FPGA

Abstract

A hexapod robot control system hardware architecture is presented. The control system is integrated in an FPGA XC6SLX16-Spartan6. The design is developed using hardware description language, VHDL. A robust architecture, divided into three logical state machine blocks, is implemented in the FPGA. Block one consists of a serial communication system for the information interchange between the user and the FPGA. The second block corresponds to the sequential circuit and data manager in charge of controlling the functions of the entire architecture. The third block is an 18-channel PWM pulse generator with a progressive duty cycle change control to perform robot movements. The system is designed to allow the user to download a complex movements sequence by combining different positions and waiting times. The architecture allows to save up to 10 movements different sequences in the internal memory block. Additionally, it allows modifying any sequence in real time, without affecting the robot operation. The architecture controls 18 servomotors allowing 18 degrees of freedom to the robot. The performing and movement test of the robot are presented.

Pico-Satellite, ARM, FPGA

Citación: IBARRA-BONILLA, Mariana Natalia, SÁNCHEZ-TEXIS, Fernando, EUSEBIO-GRANDE, Raúl y QUIÑONES-NOVELO, Fernando Julián. Secuenciador dinámico para el control de movimiento de robot hexápodo en una arquitectura FPGA. Revista de Aplicaciones de la Ingeniería. 2019. 6-20: 18-25

* Correspondencia del Autor (Correo electrónico: mariana.ibarra@itsatlixco.edu.mx)
† Investigador contribuyendo como primer autor.

Introducción

Las aplicaciones de la industria actual demandan la automatización de diversos mecanismos. En específico, los servomecanismos son usados de manera extensa en diversos procesos de manufactura, en el sector automotriz, aéreo-espacial, entre otros. Para cubrir la demanda, el estudio de la ingeniería exige tener plataformas de prueba que permitan desarrollar nuevas aplicaciones de automatización. Por esta razón, las plataformas robóticas móviles con servomecanismos son comúnmente usadas en laboratorios de investigación y universidades.

Actualmente existen numerosas aplicaciones y herramientas para la implementación de servocontroladores digitales con el uso de microcontroladores (Karakurt, Durdu y Yilmaz, 2015; Verma et al., 2017; Sendari et al., 2018; Ariawan, Santyadiputra y Sutaya, 2019), pues existe una gran variedad y disponibilidad del hardware, además de proporcionar múltiples plataformas de diseño y una fácil programación.

Sin embargo, en un microcontrolador comercial las características de hardware y periféricos disponibles son fijadas por los fabricantes. Por lo que si cambian los requerimientos del sistema, y el microcontrolador no tiene la disponibilidad del hardware y/o periféricos, se tendrá que reemplazar por un nuevo chip.

En los últimos años han surgido diseños de arquitecturas hardware de Sistemas-en-Chip (SoC, del inglés *System-on-Chip*) para la automatización de servomecanismos en tiempo real, así como el desarrollo de algoritmos para la ejecución en plataformas de hardware, tales como los arreglos de compuertas programables, conocidos como FPGA (Pullteap, 2016; Guerra-Hernandez et al., 2017**;** Zhou et al., 2018; Martínez-Prado et al., 2018).

La implementación del control de servomotores para plataformas móviles en FPGA, como un robot hexápodo, ofrece diferentes ventajas con respecto a los microcontroladores, por ejemplo la flexibilidad del diseño, pues los FPGA no están limitados por una arquitectura definida y el sistema es fácilmente reconfigurable.

Los FPGA poseen la capacidad de ejecutar procesos concurrentes en paralelo lo que permite aumentar la velocidad de ejecución y el rendimiento del sistema, mientras que la naturaleza secuencial de la ejecución de un programa en un microcontrolador afectaría significativamente la sincronización del movimiento de múltiples servomotores. Por otra parte, las herramientas de diseño disponibles en los FPGA, como simulación y síntesis, permiten desarrollar sistemas embebidos de bajo costo (Banjanovic-Mehmedovic et al., 2019).

El objetivo de este trabajo es presentar el diseño e implementación de un servocontrol integrado en un FPGA XC6SLX16 Spartan-6 usando lenguaje de descripción de hardware VHDL. Este control se aplica al movimiento autónomo de un robot hexápodo con 18 servomotores. La metodología consiste en la ejecución de tres bloques lógicos de máquina de estado finita (FSM, del inglés *finite state machine*), por el FPGA.

El primer bloque establece la comunicación serial entre el usuario y el FPGA, el segundo es el secuenciador y gestor de datos, y el tercero es el generador de los pulsos PWM de 18 canales con control de cambio de ciclo de trabajo progresivo para realizar los movimientos del robot. La arquitectura en hardware permite guardar hasta 10 secuencias diferentes de movimientos en su bloque de memoria interna, así como modificar cualquiera de ellas en tiempo de ejecución, sin afectar el funcionamiento del robot.

El sistema propuesto se ha diseñado como herramienta didáctica para que los estudiantes de Ingeniería Mecatrónica del Instituto Tecnológico Superior de Atlixco desarrollen diferentes rutinas de movimiento para el robot, y así obtengan una visión inmediata de los conceptos prácticos de robótica y así contribuir al desarrollo de sistemas digitales embebidos involucrados en estos campos de la ingeniería.

La organización de este artículo es la siguiente: Sección 2 presenta las especificaciones del hardware del robot. La sección 3 describe el desarrollo del sistema de control. La sección 4 presenta la descripción de los resultados. Las conclusiones se presentan en la sección 5.

ISSN 2410-3454

Especificaciones del hardware

El hexápodo es una plataforma robótica con seis patas, cada pata está constituida por 3 servomotores, completando un total de 18 servomotores en el robot. La Figura 1 presenta un diseño CAD de la estructura de la araña y de una pata. Cada servomotor en la pata constituye una articulación que permite el movimiento del hexápodo. La estructura del robot está hecha de acrílico, pues al tratarse de un material liviano reduce la carga de los servomotores y disminuye el peso total del robot.

Las piezas que constituyen las patas y el cuerpo utilizan acrílico de 5 mm de espesor, pues en ellas se concentrará mayor esfuerzo de carga, fricción y torsión, y las piezas de sujeción y acople de los servomotores utilizan acrílico de 3 mm de espesor. Los servomotores son de radiocontrol a bajo costo y con engranaje de metal, cada uno proporciona un ángulo de rotación de 180° con un torque de 15 kg/cm.

La posición de la flecha de los servomotores es controlada por una señal PWM, de acuerdo al ancho del pulso de la señal corresponde una posición determinada. El ancho del pulso de la señal está comprendido en el rango de 0.5 ms a 2.5 ms con 20 ms de duración por cada período, siendo la posición central un ancho de pulso de 1.5 ms.

Para la plataforma FPGA, el diseño del hardware es desarrollado usando el chip FPGA XC6SLX16 Spartan-6 de Xilinix contenido en una tarjeta de desarrollo Avanxe. La tarjeta Avanxe se presenta en la Figura 2 y sus características principales son: PSoC 3 de Cypress, un circuito de reloj de 50 MHz, convertidor USB/UART de hasta 12 MBaudios, memoria síncrona de acceso-aleatorio (SDRAM) de 4 Mb por 16 bits, 51 puertos de entrada/salida para propósito general, 8 interruptores, 8 leds, 4 displays de siete segmentos y una pantalla LCD con interfaz SPI (Intesc, 2019).

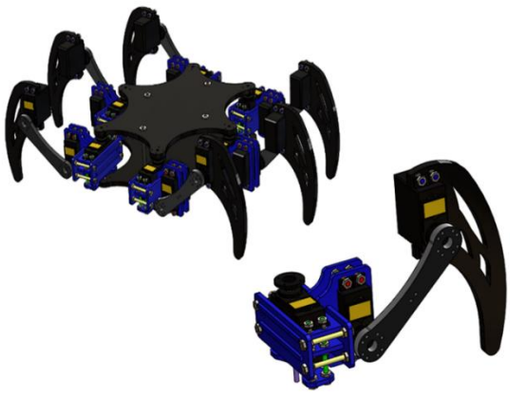

Figura 1 Modelo CAD del robot hexápodo
Fuente: Elaboración Propia

Diseño y desarrollo del control en FPGA

El diseño debe adaptarse a diferentes secuencias de movimiento que sean establecidas por los distintos usuarios, para este caso, los estudiantes. La Figura 3 presenta un diagrama a bloques de las estructuras que se implementaron en el FPGA para ejecutar el control.

Figura 2 Tarjeta de desarrollo Avanxe
Fuente (Intesc, 2019)

El servocontrol es implementado en el FPGA de acuerdo a las instrucciones provenientes, por RS-232, desde una interfaz maestra. La interfaz de software fue desarrollada en lenguaje C# para enviar los comandos y las secuencias de hacia el robot. Se utilizó una fuente externa de 5 V a 20 A para la alimentación de los servomotores y la tarjeta electrónica de control. El funcionamiento general del sistema es el siguiente: el usuario a través de la interfaz de software puede crear una rutina de movimiento, esta rutina se puede guardar y/o ejecutar en cualquier momento en el robot.

La tarjeta del FPGA es la encargada de generar las señales de control para los servomotores, a partir de las secuencias almacenadas en ella.

ISSN 2410-3454

Se pueden guardar hasta 6 secuencias diferentes de 10 movimientos o posiciones por cada servomotor. La interfaz de software envía las secuencias de movimientos a la tarjeta de control mediante un puerto serial asíncrono. Las secuencias recibidas por el FPGA son almacenadas en un bloque interno de memoria RAM. Las secuencias almacenadas son gestionadas por una FSM principal dentro del FPGA. Esta máquina de estados se encarga de generar las señales de control y los tiempos de espera, para cada servomotor.

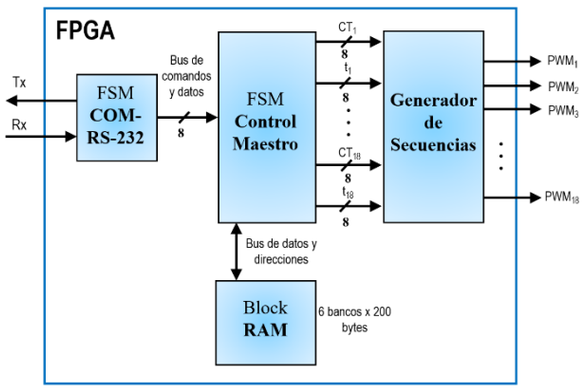

Figura 3 Esquema de las estructuras internas del FPGA que conforman el diseño
Fuente: Elaboración Propia

El usuario puede controlar el inicio y paro de una secuencia, así como también modificar o sobrescribir una rutina de movimiento. Por lo que el sistema tiene la capacidad para cambiar de una secuencia a otra en el momento que el usuario lo indique sin que la ejecución del sistema se afecte. La arquitectura interna implementada en el FPGA consta de tres máquinas de estado interconectadas entre sí. La primera FSM se denomina "COM-RS-232", y es la encargada de recibir los datos provenientes de la interfaz de software y de enviar la respuesta. La segunda FSM recibe el nombre de "Control-Maestro" y su función es gestionar el comportamiento del robot. La tercera FSM "Generador-Secuencias" se encarga de generar las señales de control PWM para los servomotores. En las secciones siguientes se describe el funcionamiento de cada bloque, así como las técnicas usada en su desarrollo.

Bloque lógico COM-RS-232

Esta FSM internamente se compone de un buffer de datos de 20 bytes de longitud y de una FSM de control interno. La FSM de control interno está compuesta por 5 estados lógicos.

Los estados 1 y 5 se encargan de la recepción y transmisión de datos respectivamente, hacia la interface. Cuando los datos de entrada son detectados en el estado 1, la FSM avanza al estado 2 y permanece en este estado hasta que un byte completo es recibido, entonces avanza al estado 3. En el estado 3 se realiza la decodificación del comando recibido. Dependiendo del comando ingresado, este bloque informará al Bloque Control-Maestro, para que este lo ejecute. Los comandos establecidos para la gestión del comportamiento del robot se describen en la Tabla 1.

Si el comando corresponde al de "Guardar", la FSM saltará al estado 4 para recibir una secuencia de datos. La secuencia completa, formada por 10 posiciones, es recibida en tramas de 20 bytes hasta completar un total de 10 tramas (secuencia de movimiento completa). Cada trama de 20 datos es enviada al bloque de Control-Maestro para su almacenamiento en la memoria RAM.

Cabe mencionar que los comandos recibidos son acompañados por bytes adicionales que proporcionan información adicional al bloque de Control. Estos bytes adicionales son presentados en la Tabla 2. El estado 5 de la FSM es destinado para el envío de tramas de respuesta hacia la interface de software.

Comando	Descripción	Codificación en Hexadecimal.
Guardar	Guarda una secuencia completa en uno de los 6 bancos disponibles de la memoria RAM.	0x01
Ejecutar	Ejecuta una secuencia completa, almacenada en alguno de los 6 bancos disponibles.	0x02
Detener	Detiene completamente el robot manteniendo la última posición que se ejecutó.	0x03
Reposo	Posiciona al robot directamente a su postura inicial de extremidades retraídas, sin importar si se está ejecutando una rutina o está detenido. Esta posición se encuentra declarada en el código del firmware y no ocupa espacio en los bancos de memoria de las secuencias.	0x04
Levantar	Extiende las extremidades del robot y lo levanta de la posición de reposo o de una rutina que esté ejecutando. Esta posición se encuentra declarada en el código del firmware y no ocupa espacio en los bancos de memoria de las secuencias.	0x05

Tabla 1 Comando utilizados en el control del robot.
Fuente: Elaboración Propia

ISSN 2410-3454

Comando (Byte 0)	Byte 1	Byte 2	Byte 3
Guardar	Dirección de memoria del banco de secuencias (byte más significativo)	Dirección de memoria del banco de secuencias (byte menos significativo)	No usado
Ejecutar	Dirección de memoria del banco de secuencias (byte más significativo)	Dirección de memoria del banco de secuencias (byte menos significativo)	Número de veces a ejecutar la secuencia
Detener	No usado	No usado	No usado
Reposo	No usado	No usado	No usado
Levantar	No usado	No usado	No usado

Tabla 2. Descripción de datos adicionales utilizados por los comandos
Fuente: Elaboración Propia

Bloque lógico Control-Maestro

Consiste de una sola FSM de 35 estados, si el comando GUARDAR es recibido, su ejecución iniciará desde el estado 3 hasta el 27; si el comando EJECUTAR es recibido, se iniciará en el estado 28 y pasará a los estados 34 y 35. El comando DETENER se ejecutará a partir del estado 29. REPOSO inicia en el estado 30 hasta el 31. Por último el comando LEVANTAR inicia en el estado 32 al 33. Los bancos de memoria 0 y 1 guardan los valores de las posiciones de Reposo y Levantar. Al iniciar el sistema estos valores se guardarán automáticamente en bancos de memoria separados, de los 6 reservados para el usuario.

El estado 2 espera a que un comando llegue del bloque COM-RS232 y hará que la FSM salte a los estados indicados. Este bloque tiene integrada una memoria RAM de 1200 bytes, distribuidos en 6 bancos de 200 bytes. Cada banco de memoria almacena las 10 posiciones que constituyen una rutina de movimiento. Mantenga en cuenta que una posición de la secuencia está formada por 18 posiciones individuales para cada servomotor. De modo que una posición individual ajusta un servomotor a una posición angular.

Por lo tanto, cada posición de una secuencia está constituido por 20 bytes, los primeros 18 bytes almacenan las posiciones, que cada servomotor adoptará, en un rango de 0 a 180º. El byte 19 corresponde al tiempo de desplazamiento, es decir, es el tiempo que tardarán en llegar los servomotores de una posición a otra.

Este dato es expresado como un número entero en mili segundos. El byte 20 corresponde al tiempo de espera, es decir, el tiempo de retardo para mantener una posición.

Bloque lógico Generador-Secuencias

Este bloque genera las señales de control para cada servomotor. Estas señales corresponden a pulsos cuadrados utilizando la técnica de modulación por ancho de pulso (PWM). Cada señal es generada a 50 Hz dentro de un rango de 0.5 a 2.5 ms de ancho de pulso, en el ciclo de trabajo (CT), las cuales son las condiciones estándar para controlar un servomotor desde una posición de 0º hasta 180º.

Además, este bloque tiene la función de identificar la posición actual en la que se encuentra cada servomotor y así generar la señal adecuada para llevarlo hacia la posición siguiente. Por ejemplo si un servomotor se encuentra anclado en una posición de 90º grados (CT = 1.5ms) y la siguiente posición indica un ángulo de 45º (CT=.75ms) el bloque generará una secuencia descendente del CT para que el motor se mueva de forma suave hasta la posición final.

Por lo que el bloque controla el tiempo de desplazamiento del servo, entre cada movimiento, dependiendo del valor almacenado en el byte 19. Como se mencionó anteriormente este dato guarda un número entero de 0 a 255 que representa un tiempo de retardo en ms. El byte 20 establece el tiempo de retardo o de espera entre cada posición de la secuencia.

Las señales de PWM son generadas con una resolución de 24 bits, lo que permite ajustar con precisión hasta en un grado la posición del motor. Este grado de resolución fue necesario para simplificar la lógica de control, debido a que el FPGA trabaja a una frecuencia de reloj de 50 MHz.

La generación de tiempos en la escala de ms requiere de contadores de retardo de entre 16 a 24 bits de resolución. Debido a la resolución utilizada internamente se cuenta con un bloque convertidor de resolución que transforma los datos de 8 bits a 24 bits.

ISSN 2410-3454

Resultados

Un robot hexápodo de seis patas puede tener una gran cantidad de modos de caminar (Aguilar, et.al, 2015; Haynes y Rizzi, 2006). Por ello se desarrolló la interfaz maestra en C#, la cual se muestra en la Figura 4. Usando la interfaz, el usuario puede introducir un máximo de 10 posiciones diferentes para cada servomotor y los tiempos que desea que dure el cambio de posición, esto con la finalidad de que el servo no se mueva a su velocidad máxima.

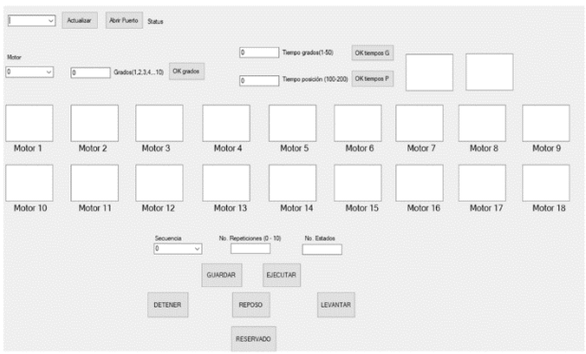

Figura 4 Interfaz maestra
Fuente: Elaboración Propia

La secuencia principal que se comprobó en el sistema de control, es la secuencia que hace al robot moverse hacia adelante. Para esto, se utilizó el principio del polígono de estabilidad de 3 puntos, el cual consiste en formar un triángulo entre las patas que se apoyan en la superficie para soportar la estructura.

La Figura 5 muestra la secuencia de los 10 estados que genera la rutina de movimiento hacia adelante. Las Figuras 6 y 7 muestran el robot hexápodo en la posición inicial de reposo (E1) y en la posición levantar (E2), respectivamente. De igual manera se comprobó la secuencia del polígono de 3 puntos, pero para que el robot se desplace lateralmente. Durante la ejecución de cada secuencia, una orden aleatoria de una secuencia diferente, es enviada a la mitad del tiempo de ejecución de la actual, esto fue con el objetivo de comprobar la estabilidad del robot y del sistema.

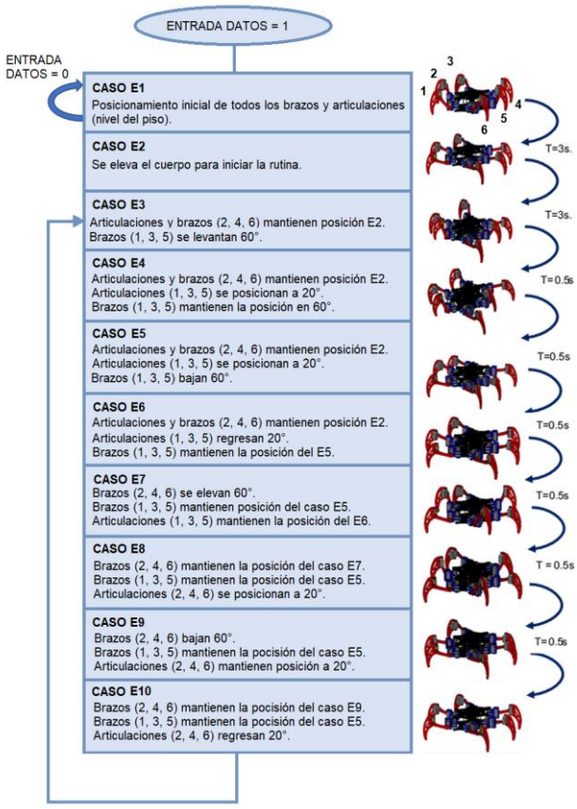

Figura 5 Secuencia de 10 estados para caminar hacia adelante
Fuente: Elaboración Propia

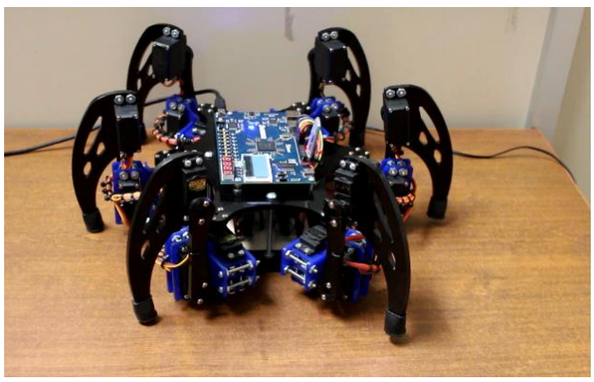

Figura 6 Robot hexápodo en la posición inicial de reposo
Fuente: Elaboración Propia

Para comprobar el generador de señales PWM se realizó una prueba con un analizador de estados lógicos utilizando un equipo Tektronix MSO-2024B de 8 canales digitales simultáneos. La prueba consistió en enviar una secuencia y comprobar la generación en paralelo de todas las señales respetando los ciclos de trabajo y tiempos de retardo establecidos desde la interfaz de usuario. La Figura 8 muestra una imagen de 8 señales PWM de prueba.

ISSN 2410-3454

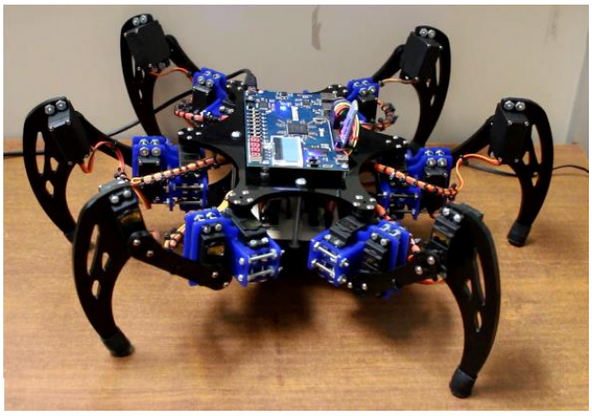

Figura 7 Robot hexápodo en la posición Levantar
Fuente: Elaboración Propia

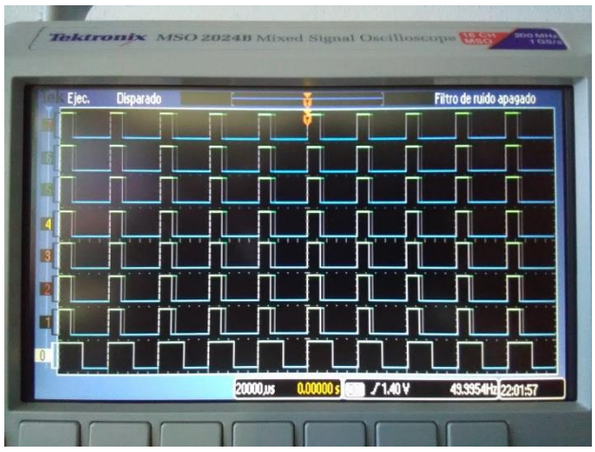

Figura 8 Señales PWM de prueba
Fuente: Elaboración Propia

Agradecimiento

Los autores agradecen a la empresa Intesc Electrónica & Embebidos por la colaboración y soporte en el desarrollo de este proyecto.

Conclusiones

Se presentó la implementación de un servocontrol para el movimiento de un motor hexápodo de 18 grados de libertad en un FPGA Spartan-6. El control por máquina de estados finitos (FSM) permite que los procesos sean independientes y dinámicos pues la FSM de control permite modificar cualquier secuencia de movimiento en tiempo de ejecución, sin reiniciar el funcionamiento del robot.

La FSM está diseñada para aumentar más bloques de control PWM para expandir los motores, por lo que no está limitada únicamente a 18 servomotores.

La implementación en FPGA ofrece diferentes ventajas, en comparación con los sistemas implementados en microcontroladores, tales como la flexibilidad de la arquitectura del diseño y la ejecución de procesos en paralelo. Por lo tanto, el sistema propuesto contribuye al avance del desarrollo de sistemas digitales embebidos en FPGA.

Sugerencias y trabajo en progreso

Como sugerencias es posible incorporar en la plataforma del robot sensores de obstáculos, inerciales, ópticos para detectar las condiciones del ambiente donde se desplace y así desarrollar algoritmos de navegación autónomos.

El trabajo en progreso consiste en las mejoras de la interfaz gráfica, de tal manera que se pueda observar un modelo 3D con los movimientos del robot. Esto con la finalidad de aplicarlo como herramienta educativa y de investigación en sistemas de navegación y exploración.

Referencias

Aguilar, L. M., Torres, J. P., Jimenes, C. R., & Cabrera, D. R. (2015, October). Balance of a hexapod in real time using a FPGA. In 2015 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON) (pp. 825-828). IEEE.

Ariawan, K. U., Santyadiputra, G. S., y Sutaya, I. W. (2019, February). Design of Hexapod Robot Movement Based on Arduino Mega 2560. In Journal of Physics: Conference Series (Vol. 1165, No. 1, p. 012011). IOP Publishing.

Banjanovic-Mehmedovic, L., Mujkic, A., Babic, N., & Secic, J. (2019, June). Hexapod Robot Navigation Using FPGA Based Controller. In International Conference "New Technologies, Development and Applications" (pp. 42-51). Springer, Cham.

Guerra-Hernandez, E. I., Espinal, A., Batres-Mendoza, P., Garcia-Capulin, C. H., Romero-Troncoso, R. D. J., & Rostro-Gonzalez, H. (2017). A FPGA-based neuromorphic locomotion system for multi-legged robots. IEEE Access, 5, 8301-8312.

ISSN 2410-3454

Haynes, G. C., & Rizzi, A. A. (2006, May). Gaits and gait transitions for legged robots. In Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006. (pp. 1117-1122). IEEE.

Intesc Electrónic & Embebidos (01-04-2019). Página principal. México: Intesc. https://www.intesc.mx/.

Karakurt, T., Durdu, A., y Yilmaz, N. (2015). Design of six legged spider robot and evolving walking algorithms. International Journal of Machine Learning and Computing, 5(2), 96.

Martínez-Prado, M. A., Rodríguez-Reséndiz, J., Gómez-Loenzo, R. A., Herrera-Ruiz, G., & Franco-Gasca, L. A. (2018). An FPGA-based open architecture industrial robot controller. IEEE Access, 6, 13407-13417.

Pullteap, S. (2016, June). Development of a walking robot by using FPGA controller. In 2016 11th France-Japan & 9th Europe-Asia Congress on Mechatronics (MECATRONICS)/17th International Conference on Research and Education in Mechatronics (REM) (pp. 054-057). IEEE.

Sendari, S., Hadi, M. S., Handayani, A. N., Wahyudi, Y. R., & Lin, H. I. (2018, November). Implementation of PD (Proportional Derivative) Control System On Six-Legged Wall Follower Robot. In *2018 International Automatic Control Conference (CACS)* (pp. 1-6). IEEE.

Verma, G., Rai, P., Chauhan, B., Kumar, A., Pandey, P., y Karnail, V. (2017). Hardware implementation of autonomous hexapod spider robot. *International Journal of Information Technology*, *9*(4), 395-398.

Zhou, L., Liu, Q. X., Wang, B. J., Li, X. Q., & ZHANG, J. Q. (2018). Position Servo Controller for DC Micro Motor and Its Realization Based on FPGA. Small & Special Electrical Machines, (1), 12.

ISSN 2410-3454