Diseño e implementación de gobernador de motores basado en PWM y FPGA para aplicarlo en el desarrollo de controlador de vuelo de UAV

Noé Monterrosa
Instituto de Investigación e Innovación electrónica
Universidad Don Bosco
El Salvador, C.A.
noe.m92@hotmail.com

Carlos Bran
Instituto de Investigación e Innovación electrónica
Universidad Don Bosco
El Salvador, C.A.
cbran@udb.edu.sv

Resumen—En el presente documento se expondrá como se realizó el diseño Top Down del controlador de navegación a implementar en el UAV "Drone Bosco", además se presentará la programación e implementación del driver para modulación por ancho de pulso que será usado en el Proyecto Drone Bosco para controlar sus motores. En el documento se explicarán las razones para la utilización de FPGA con lenguaje descriptivo de hardware, y como sus características ayudaron a lograr el objetivo. Por último se darán a conocer los resultados obtenidos en el laboratorio de pruebas con servomotores.

Palabras Clave—FPGA, VHDL, PWM, servomotor, drone, UAV, driver.

I. INTRODUCCION

Los UAV (Vehículos Aéreos No Tripulados, por sus siglas en inglés) o drones como son comúnmente conocidos, son aeronaves pilotadas desde tierra o que pueden volar de manera autónoma. Estos aparatos se han vuelto muy populares en los últimos años y son actualmente utilizados para muchas aplicaciones como juguetes, dispositivos de vigilancia, asistentes de camarógrafos, entre otras. En Estados Unidos se están empezando a utilizar drones para repartir paquetes y grandes compañías como FedEx y UPS se encuentran trabajando con el gobierno para regular el uso de los éstos en zonas urbanas [1]. La capacidad de autonomía de estos vehículos es proporcional al poder de cálculo de su controlador, ya que este permite agregar inteligencia al aparato, entre mayor es la inteligencia del mismo los controladores se vuelven menos dependientes de la intervención humana para su operación. En este trabajo se presenta la solución para el gobernador de los motores de un UAV, tomando en cuenta las especificaciones de diseño de todo el controlador de vuelo con la idea de integrar toda la solución en un sistema embebido de paralelismo completo como un FPGA, que nos brinde el suficiente poder de cálculo para ampliar la inteligencia del controlador y obtener un diseño con mayor autonomía. El trabajo consta de 7 partes, en la segunda se presentara una visión general del controlador a diseñar, el concepto completo del UAV y el detalle funcional del controlador, en la tercera y cuarta parte se hará una revisión técnica de los moduladores por ancho de pulso y su aplicabilidad en el manejo de motores DC, específicamente servomotores y motores sin escobillas, en la quinta parte se introducirá a los lenguajes de descripción de hardware y la facilidad de implementación de los diseños sobre FPGA, en la sexta y séptima parte se presentara la solución para el gobernador de los motores PWM y los resultados obtenidos, y en la última parte las conclusiones y consideraciones finales.

II. DISEÑO CONCEPTUAL

Para el diseño conceptual de la aeronave se usa la metodología de diseño Top-Down, con lo que se puede describir de forma completa el funcionamiento de la unidad para luego dividirla en partes que pueden ser trabajadas y probadas de forma separada por distintos grupos, lo que reduce el tiempo de depuración, prototipado e integración. Todo el UAV se divide en 4 grandes bloques que son:

- Propulsión y fuselaje.
- Generación de energía
- Telecomunicaciones y video
- Control de vuelo

El control de vuelo es el bloque en el que nos enfocaremos para tener una visión general de la solución que se plantea para el gobernador de los motores, después de una investigación fue posible diferenciar las partes principales que debe tener el control de vuelo. Se muestran en la figura 1.

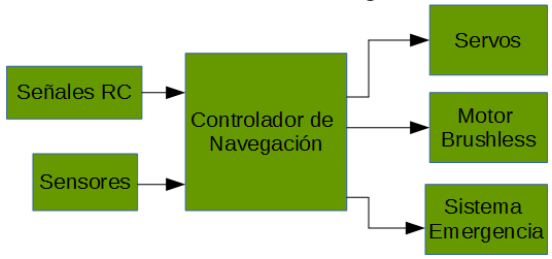

Fig. 1. Diseño conceptual general de la sección de control de vuelo.

Como se puede observar el controlador de navegación recibe las señales que envía el usuario por medio de un receptor de radiofrecuencia, desde el cual se brindará la información del vector de vuelo (way point), dicha información se integrara con la telemetría que se obtendrá de sensores como altímetro, giróscopo, acelerómetro, entre otros para ajustar la posición y corregir el vector de vuelo de forma más exacta, usando un algoritmo de cálculo basado en una máquina de estados, la cual brindara la información necesaria para los actuadores que serán controlados por el gobernador PWM. Los actuadores serán 4 servomotores, dos para gobernar los alerones, uno para el timón de cola y uno para el timón de profundidad, además de un motor DC sin escobillas (brushless motor) que brindará el impulso completo a la nave; por último el gobernador también contará con los mecanismos para el manejo de eventualidades o emergencias, para garantizar la integridad de la nave y evitar que esta se dañe en caso de accidentes, ejemplo de estos sistemas son el control del paracaídas, la asistencia en el aterrizaje y el despegue, y la gestión de la energía. Luego de tener este diseño general se realizó un diseño específico de las partes del controlador de navegación, el cual se muestra en la figura 2:

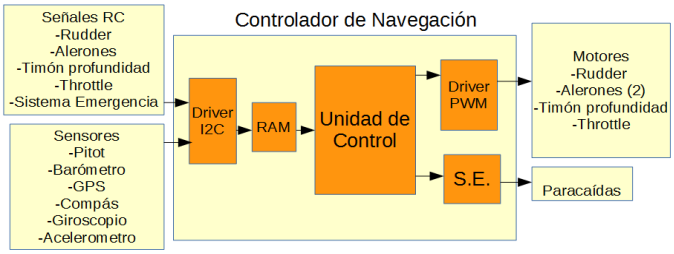

Fig. 2. Diseño conceptual específico de la sección de control de vuelo.

En este esquema se da una idea más específica de las partes que involucran el control de vuelo, se detallan los sensores a utilizar, las señales de control remoto, los buses de comunicación interna y los gobernadores de los actuadores de la nave, donde se presenta el relacionado con nuestro trabajo para la generación de las señal de control con el PWM; este recibirá la información del vector de vuelo ajustado con la telemetría con la cual el driver generará las 4 señales que alimentaran los servomotores y el motor sin escobillas para completar la ejecución de la maniobra recibida del navegador; la velocidad de respuesta y la precisión del control de la señal PWM son especificaciones críticas para el diseño.

III. Modulación por ancho de pulso

Según la web de National Instruments tenemos la siguiente definición: *"La modulación de ancho de pulso (PWM, por sus siglas en inglés) de una señal es una técnica que logra producir el efecto de una señal analógica sobre una carga, a partir de la variación de la frecuencia y ciclo de trabajo de una señal digital. El ciclo de trabajo describe la cantidad de tiempo que la señal está en un estado lógico alto, como un porcentaje del tiempo total que esta toma para completar un ciclo completo. La frecuencia determina que tan rápido se completa un ciclo (por ejemplo: 1000 Hz corresponde a 1000 ciclos en un segundo), y por consiguiente que tan rápido se cambia entre los estados lógicos alto y bajo."* [2].

La capacidad de regulación del ancho del pulso que facilitan los moduladores PWM los hacen una solución muy precisa para gobernar válvulas, bombas, sistemas hidráulicos, algunos dispositivos mecánicos y motores de corriente directa entre otros. Con el driver del proyecto se gobernaran los servomotores que controlan las superficies de vuelo de la aeronave y también el motor brushless que controla la potencia y provee el impulso necesario para volar, una especificación importante será el nivel de precisión del control del ancho de pulso, ya que esta determinara la posición angular de los servomotores y la velocidad del motor brushless.

IV. Actuadores del UAV

Dos son los tipos de actuadores que tiene el UAV, el primero y más importante es el servomotor, estos serán acoplados a los alerones, timón de profundidad y timón de cola para controlar de manera precisa los movimientos de la aeronave, por lo que un control fino de estas unidades es necesario. Los servomotores se utilizan debido a su tamaño, peso reducido, gran precisión y eficiencia, estos cuentan en su interior con un pequeño motor DC, un juego de engranajes que ayuda a aumentar su torque, un potenciómetro y un circuito de control para garantizar su posición con gran precisión. El eje del motor se encuentra acoplado al potenciómetro el cual varía su resistencia conforme el movimiento del eje del motor y es por medio de este cambio de resistencia que el circuito de control puede determinar de manera precisa cuanto movimiento hay y en qué dirección. La posición del servomotor se controla enviando una señal eléctrica PWM por el cable de control, en la figura 3 se representa el tiempo del estado en alto y los grados mecánicos del servomotor, el periodo es de 20 ms.

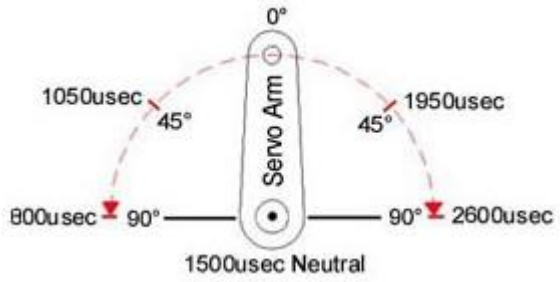

Fig. 3. Relación del ancho de pulso con la posición del servomotor.

Según las especificaciones de cada unidad hay valores máximos y mínimos de ancho de pulso, el servomotor a utilizar para el manejo de las superficies de control del drone es el 1501MG de la compañía PowerHD, las especificaciones técnicas se pueden encontrar en [6]. Esta señal debe de mantenerse constante para que el servomotor mantenga su posición, y a su vez, lograr que el servomotor se oponga a torques ejercidos por fuerzas externas y se mantenga en la posición que ha sido enviada por su señal de control, esto prueba ser muy útil en los UAV ya que las posiciones donde se encuentran las superficies de vuelo se mantendrán a pesar de cualquier fuerza externa ejercida por ráfagas de viento. Lo que

se pretende con nuestro gobernador de PWM es traducir las señales de la Unidad de Control para cada uno de los servomotores a ciclos de trabajo correspondientes en la señal PWM, obteniendo una posición equivalente en grados mecánicos.

El otro tipo de motor que el driver va a gobernar es el motor brushless que controlara el impulso y velocidad que necesita la aeronave para permanecer en vuelo. Este tipo de motor eléctrico no emplea escobillas para realizar el cambio de polaridad en el rotor. Estos motores en la mayoría de los casos poseen 3 fases y trabajan con corriente alterna por lo cual se necesita un circuito electrónico inversor para transformar la corriente directa que suministraran las baterías, este circuito electrónico controla la alimentación de cada una de las fases y de esta manera regula las RPM del motor. Comúnmente se conoce este circuito como ESC (Circuito Electrónico de Velocidad, por sus siglas en inglés). Para controlar las RPM del motor, este circuito recibe una señal PWM de parte del controlador de navegación, luego interpreta el ciclo de trabajo de la señal de control y lo transforma en señales analógicas equivalentes para regular la alimentación de cada una de las fases del motor y controlar así sus RPM. Es muy importante aclarar que para cada uno de los tipos de actuadores se requieren características distintas de frecuencia y rangos de variación del ciclo de trabajo, por lo que el diseño del gobernador debe ser los más genérico posible para lograr adaptarse a las características de cada actuador.

V. LENGUAJE DESCRIPTIVO DE HARDWARE Y FPGA

A continuación se resume lo que es VHDL: *"VHDL es un lenguaje de descripción de hardware. Describe el comportamiento de un circuito electrónico o un sistema, donde el circuito o sistema puede ser implementado. VHDL es la abreviación de VHSIC Hardware Description Language (Lenguaje Descriptivo de Hardware VHSIC). VHDL fue el primer lenguaje descriptivo de hardware y fue estandarizado por la IEEE con el estándar IEEE 1076. El propósito de VHDL es para la síntesis de circuitos así como su simulación. Una motivación fundamental para utilizar VHDL es que VHDL es un lenguaje estándar e independiente de cualquier proveedor por lo tanto es portable y reusable. Las dos aplicaciones principales de VHDL se encuentran en el campo de los Dispositivos Lógicos Programables (incluyendo los CPLDs-Complex Programable Logic Devices y los FPGAs- Field Programable Gate Arrays) y en el campo de los ASICs (Application Specific Integrated Circuits). Una vez que el código de VHDL ha sido escrito, se puede usar tanto para implementar en dispositivos programables (de las compañías Altera, Xilinx, Atmel, etc.) o puede ser utilizado para la fabricación de chips ASIC. En la actualidad, muchos chips comerciales complejos como microcontroladores son diseñados usando esta tecnología. Una nota final sobre VHDL es que, al contrario de los software regulares que son secuenciales, sus declaraciones son por defecto concurrentes (o paralelas)."* [7].

Las características principales que podemos resumir del lenguaje VHDL son las siguientes:

- Es un Lenguaje Descriptivo de Hardware. Este tipo de lenguaje se utiliza para definir la estructura, diseño y operación de circuitos electrónicos. Hacen posible una descripción formal de un circuito electrónico y posibilitan su análisis automático y simulación. El lenguaje de descripción de hardware permite a los ingenieros diseñar desde el concepto e ir bajando hasta llegar al nivel de circuito, probando cada fase de forma estricta hasta su implementación final (Top-Down design), lo que reduce los costos de prototipado significativamente.
- Sus dos aplicaciones principales se encuentran en el campo de los Dispositivos Lógicos Programables y los ASICs (Circuitos Integrados de Aplicación Especifica). Prácticamente todos los nuevos chips son diseñados utilizando los lenguajes HDL (VHDL o su hermano Verilog). Cuando ya se tiene un código que cumpla los requerimientos del diseñador primero es probado en un dispositivo lógico programable y luego de que se comprueba su correcto funcionamiento, se implementa en un ASIC, que son chips construidos para cumplir una función específica y son reproducidos en masa [8].
- Sus declaraciones son concurrentes a diferencia de los software de programación regular, los cuales trabajan de manera secuencial. Esto quiere decir que puede realizar diferentes tareas al mismo tiempo para diferentes procesos, al contrario de la programación secuencial en la que se tiene que ir del paso A, al paso B y al paso C, realizando para cada tiempo una acción en específico.

Es debido a estas 3 características principales que se decidió utilizar codificación VHDL con una tarjeta FPGA, primero sabemos que al ser un lenguaje descriptivo de hardware nos permite describir las funciones que requerimos del sistema sin preocuparnos en sus detalles eléctricos ya que la herramienta de síntesis EDA se encarga de hacer el diseño del circuito en base a los requerimientos. Para nuestro diseño lo que necesitamos es que el driver tome un valor numérico de la Unidad de Control, que representa el ángulo al que queremos colocar el servomotor, y lo transforme a una señal PWM que el servomotor pueda interpretar; esta es la descripción del hardware en la que se basará el diseño. Por otro lado tenemos que este tipo de programación está siendo utilizada desde hace varios años en los países de primer mundo para el diseño de chips y otras aplicaciones de diseño [9], [10] y modelado, sin embargo, en El Salvador y los países de la región Centroamericana el uso de esta tecnología es poca (prueba de esto es que en el último CONCAPAN sólo hubo 2 papers usando esta tecnología de un total de 87) [11], lo cual representa un retraso en relación a los países donde sí se está utilizando. Por último tenemos que sus declaraciones son concurrentes, esto quiere decir que a diferencia de otros dispositivos, como los microcontroladores, pueden realizar diferentes tareas al mismo tiempo. Un sistema secuencial tarda más tiempo en reaccionar ante los cambios que se presentan, en contraste, al ser nuestro sistema concurrente, se pueden llevar diferentes procesos simultáneamente.

El gran poder de procesamiento que brindan los FPGA en espacios reducidos y con consumos muy bajos de energía lo hacen ideal para el desarrollo de los controladores de vuelo, ya que es posible implementar muchas funciones de control con un costo muy reducido en tiempo, lo que brinda los mejores

tiempos de respuesta de cualquier sistema embebido, por otro lado la facilidad de implementación y el nivel de precisión que se puede obtener permite que los tiempos de prototipado se reduzcan significativamente y se pueda trabajar de forma paralela con otros módulos que serán fácilmente integrados.

VI. CODIFICACIÓN DE LA SOLUCIÓN

Debido a las diferencias entre los motores de control e impulso, y la necesidad de portabilidad del código en distintas arquitecturas, el núcleo del gobernador se desarrolló de forma genérica para luego ser instanciado según las características de los motores y el tipo de FPGA donde se implementará, para ello varios parámetros básicos son necesarios, la frecuencia nominal del FPGA, la frecuencia del motor y todos los parámetros relativos al cálculo del ciclo de trabajo. Lo primero que se hace es declarar las librerías necesarias según los tipos de datos y señales que usaremos, ya que se trabajará con valores signados y no signados (unsigned) se declara la librería Numeric_STD.all. Luego se declaran los parámetros genéricos, los cuales pueden modificarse según se necesite, estos son: la frecuencia del FPGA, la frecuencia de operación del servo, el máximo porcentaje de ciclo de trabajo, la máxima cantidad de grados que se moverá el servo y el valor de offset en términos de porcentaje que sirve para dar valor a 0 grados. En la declaración del puerto del driver tenemos las siguientes señales:

- Entrada: Es el puerto en donde la Unidad de Control enviará un número entre 0 y 180 indicando los grados a los que se quiere tener el servo.
- Reset: Inhabilita el driver, a un punto cero.
- Salida: Representa la señal de PWM, con la frecuencia y ciclo de trabajo gobernados.
- Address: Es el puerto de la dirección única de cada driver, este valor servirá a la hora de instanciar 2 o más drivers con el fin de diferenciarlos.
- Ena: Es una señal habilitadora. Cuando es igual que Address permite al driver tomar los datos en la entrada y transformarlos a la señal PWM.

En la arquitectura se declaran 4 señales que nos servirán para hacer conversiones y hacer las asignaciones que se detallan a continuación:

- fclk <= ffpga/fservo; Con esta fórmula se divide la frecuencia del reloj interno del FPGA entre la frecuencia del servo, por ejemplo para un servo que opera a 50 Hz y un reloj interno de 40 MHz tenemos como resultado 800,000, eso quiere decir que cada ciclo del servo tendrá como duración 800,000 ciclos del reloj del FPGA, así se resolverá el problema con divisores de frecuencia sin alterar la ruta óptima de reloj del FPGA.
- dutyhi <= ((fclk*maxduty)/100): Ya que el ciclo de trabajo del PWM no siempre trabaja al 100% del periodo se ha generado esta fórmula para definir una cota superior de ciclo de trabajo, así si deseamos un máximo de 50% tenemos (800,000* 50)/100 lo cual nos da 400,000 periodos de reloj.
- factor <= dutyhi/grados: Esta fórmula brinda el factor de conversión y determina la variabilidad del ciclo de trabajo multiplicándolo por la señal de "entrada". Por ejemplo si la máxima cantidad de grados es de 180 se divide 400,000/180 =2222, este número se multiplica por la entrada, si tengo en la entrada 180 * 2222= 400,000 que es la máxima cantidad de pulsos necesaria para un ciclo de trabajo de 50%. Variando la entrada se obtiene un rango de 0 a 400,000 ciclos.
- zerod <= ((fclk*zero)/100): Con esta fórmula lo que se logra son los ciclos de reloj interno para el offset que me indica en porcentaje la variable zero. Este offset es útil para cuando el valor del ángulo es 0.

Luego se declara el proceso, que es un segmento secuencial del código, y las variables que lograran hacer las conversiones de la entrada a un número entero (integer). El primer if que tenemos detecta si el puerto de reset está activo, si es así manda un 0 al puerto de salida, sino, avanza a la siguiente condición if, en esta el driver detecta los flancos ascendentes del reloj del FPGA y los cuenta con la variable cnt1. El siguiente if detecta si el puerto habilitador (ena) tiene el mismo valor que el de la dirección de nuestro driver (address), si es así prosigue al siguiente if, de lo contrario el valor de la salida se mantiene igual ya que no se ha aceptado ningún nuevo valor de entrada. Cuando si se cumple la condición se procede al siguiente if adonde se compara si el valor de la entrada es mayor o igual a 180, de ser así se guarda en la variable entrada4 el valor que corresponde a 180 grados. Si el valor de la entrada es menor a 180 se procede a transformarlo primero a entero y luego se multiplica por la señal factor, previamente calculada, esta multiplicación nos dará el número de ciclos de reloj que pasara la salida en alto acorde a la entrada, por último se le suma la variable zerod que sirve como offset y se guarda el resultado en entrada4. Se cierran los if y luego se procede a comparar el valor que se almaceno en entrada4 con el valor del contador, si entrada4 es menor o igual al contador la salida se pone en 1 lógico, si entrada4 es mayor que contador la salida se pone en 0 lógico y cuando el valor de contador ya ha alcanzado el valor del número de ciclos por periodo definido con la variable fclk el contador se resetea y se pone en 0. Este proceso se realiza por cada ciclo de salida de la señal PWM que va al servomotor. La figura 4 muestra los resultados de la simulación ejecutada en el software ISE Design Suite 14.5. Como se puede observar mientras reset se encuentra en estado alto no se tiene señal de salida, una vez reset cambia a estado bajo la salida actúa de acuerdo a la señal de entrada, ya que los valores de la dirección (address) y el habilitador (ena) son iguales. El valor de entrada en ese momento es 0 grados lo cual, según el datasheet del servomotor, corresponde a un tiempo en estado alto de 0.8 ms de un periodo de 20 ms para nuestro servomotor. Al cambiar la entrada a 30 grados el driver reacciona y la salida cambia a una señal con un tiempo en estado en alto equivalente, esta señal se mantiene a pesar de que los valores de la entrada cambian debido a que las señales address y ena ya no coinciden. Cuando vuelven a coincidir el driver transforma su salida al valor de 120 grados, luego a 180 grados y se mantiene en este valor a pesar

de que la entrada llega a ser de 254 grados ya que este no es un valor valido. El código completo puede consultarse en los anexos.

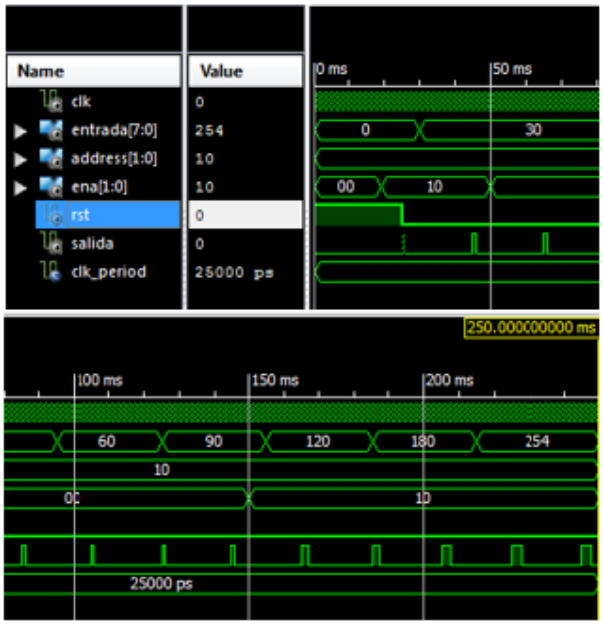

Fig. 4. Simulación de driver en software ISE Design Suite 14.5.

VII. Pruebas de laboratorio

Luego de obtener una simulación exitosa del driver se procedió a realizar una instanciación de 4 módulos de PWM para 4 servomotores y una máquina de estado que simula las señales que enviara la Unidad de Control. El código se implementó en una tarjeta FPGA Spartan 6 XC6SLX9-2CSG324 y a su salida se ocupó el circuito de la Fig. 5 para aislar eléctricamente el FPGA de la fuente de voltaje y los servomotores.

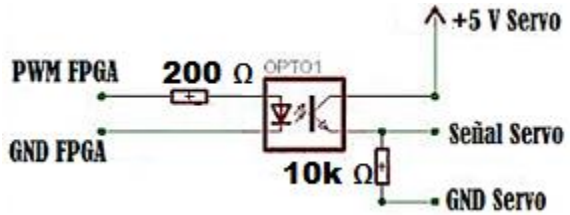

Fig. 5. Circuito implementado en la conexión del FPGA con los servomotores.

El optocoplador utilizado fue el 4n35. Este circuito se repitió para cada una de las 4 señales de PWM que generaba el FPGA.

Se construyó una maqueta en durapax para simular el avión y se colocaron 4 servomotores para representar los alerones, el timón de profundidad y el timón de cola, la Fig. 6 muestra el banco de pruebas completo, en ella se observa la maqueta del avión con los 4 servomotores, la tarjeta FPGA Spartan 6, el circuito de aislamiento eléctrico armado en breadboard y las señales de entrada de dos servomotores en el osciloscopio. La prueba fue exitosa y los servomotores se movieron de acuerdo a los comandos programados en la máquina de estado que simulaba la unidad de control, además no hubo ningún calentamiento en los servomotores y funcionaron con la normalidad y precisión necesaria para la solución.

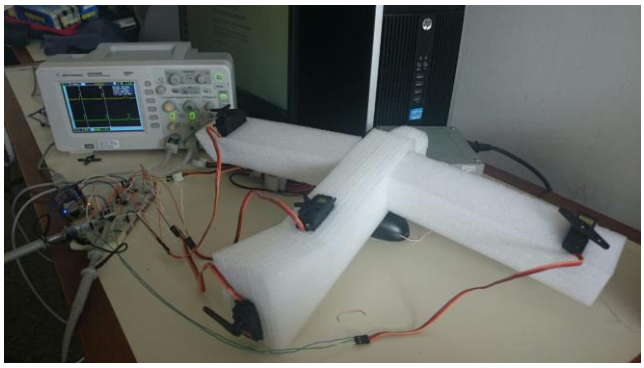

Fig. 6. Fotografía tomada el día de las pruebas de laboratorio.

VIII. Conclusiones

De acuerdo a los resultados obtenidos durante la simulación y en el banco de pruebas del laboratorio se puede concluir lo siguiente:

La implementación de la técnica de diseño Top Down, acelera el tiempo de prototipado de cualquier solución, ayuda a segmentar los bloques funcionales y garantiza los resultados exitosos del prototipo, además de la capacidad de los módulos para reutilizarse y fácilmente integrarse en otros diseños. La implementación del diseño en el FPGA presenta un nivel de utilización del chip menor al 1% de sus Slice Registers y del 3% de los Slice LUTs, porcentajes pequeños que permitirán integrar los restantes módulos de la solución general en el mismo FPGA, esto es debido a que las herramientas de síntesis optimizan el uso del área, con lo que se reduce el consumo de potencia, el tamaño de la solución, se incrementa la velocidad y el poder de procesamiento, lo que es muy importante para proveer mayor inteligencia al UAV. Ya que los FPGA trabajan de forma paralela el uso de los divisores de frecuencia para el gobernador de PWM no penalizaran el rendimiento de cualquiera de los otros módulos a implementar los cuales funcionaran de forma completamente independiente. El gobernador de PWM presentado puede ser fácilmente portado a cualquier arquitectura de FPGA y usado para aplicaciones distintas a las implementadas en este trabajo. En próximos trabajos se implementaran los módulos restantes, telemetría, comunicaciones, máquina de estados de control, compensación, etc. los que se están desarrollando de forma paralela y serán integrados en un solo chip.

Referencias

[1] A. Oppenheimer, ¡Crear o Morir!, Buenos Aires: Debate, 2014.

[2] National Instruments, "¿Que es una Señal Modulada por Ancho de Pulso (PWM) y Para Qué es Utilizada?," [Online]. Available: http://digital.ni.com/public.nsf/websearch/AA1BDEA4AA224E3E86257CE400707527?opendocument&Submitted&&node=133020_esa. [Accessed Junio 2015].

[3] P. Volnei A., Circuit Design with VHDL, Cambridge: MIT Press, 2004.

[4] J. Turley, The Essential Guide to Semiconductors, New Jersey: Pearson Education Inc., 2003.

[5] Xilinx, "Consumer Electronics," [Online]. Available: http://www.xilinx.com/applications/consumer-electronics.html. [Accessed Junio 2015].

[6] Xilinx, "Industrial," [Online]. Available: http://www.xilinx.com/applications/industrial.html. [Accessed Junio 2015].

[7] IEEE, "IEEE Explore Search Results," [Online]. Available: http://ieeexplore.ieee.org/search/searchresult.jsp?queryText=Central%20America%20And%20Panama%20Convention%20.LB.CONCAPAN%20XXXIV.RB.,%202014%20IEEE&pageNumber=1&newsearch=true&searchField=Search_All. [Accessed Junio 2015].

Noe José Monterrosa Vásquez, actualmente estudia quinto año de Ingeniería Mecatrónica en la Universidad Don Bosco, donde también es Presidente de la Asociación de Estudiantes de Ingeniería Mecatrónica (AEIMEC) y vicepresidente del Capítulo de Robótica y Automatización (RAS-IEEE). Es graduado en Técnico en Ingeniería Electrónica, con mención Cum Laude, de la Universidad Don Bosco. Se graduó con el título de Bachiller Técnico Vocacional en la rama de Electrónica del Instituto Técnico Ricaldone, consiguiendo el Diploma de Honor por las mejores notas de su promoción.

Carlos Bran: Nació en Santiago de María, Usulután, El Salvador, el 14 de febrero de 1972, se graduó de la Universidad Centroamericana José Simeón Cañas, como Ingeniero Electricista, cuenta además con estudios de maestría en gestión tecnológica de la Universidad Tecnológica de Pereira, Colombia y de Investigación en Tecnologías de la información de la Universidad de Santiago de Compostela, España. La experiencia laboral incluye empresas como Dymel, Dynamo, AT&T, diversas compañías en el área de las telecomunicaciones, Decano de la facultad de ingeniería de la Universidad Don Bosco Director del programa Cisco Networking Academy, entre otras. Es profesor de sistemas embebidos y desarrolla investigaciones con tecnologías de microcontroladores, FPGA, procesamiento paralelo, diseño electrónico avanzado, protocolos de comunicación, seguridad informática, lenguajes intuitivos y energy harvesting en los que está preparando estudios de doctorado en el Centro Singular de Investigación en Tecnologías de la Información CITIUS.

ANEXOS

Código en VHDL de driver PWM:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity pwm2 is
generic (ffpga: integer := 40000000;  --Frecuencia fpga
         fservo: integer := 50;       --Frecuencia Servo
         maxduty: integer:= 9;        --Duty cicle
         grados: integer := 180;      --Numero de grados
         zero:    integer:=4);        --Offset
    Port ( clk : in  STD_LOGIC;
         entrada: in STD_LOGIC_VECTOR (7 downto 0);
         address: in STD_LOGIC_VECTOR (1 downto 0);
         ena: in STD_LOGIC_VECTOR (1 downto 0);
         rst : in  STD_LOGIC;
         salida:out STD_LOGIC :='0');
end pwm2;
architecture Behavioral of pwm2 is
signal fclk: integer range 0 to 40000000 :=0;
signal dutyhi: integer range 0 to 40000000 :=0;
signal factor: integer range 0 to 40000000 :=0;
signal zerod: integer range 0 to 40000000 :=0;
begin
fclk <= ffpga/fservo;
dutyhi <= ((fclk*maxduty)/100);
factor <= dutyhi/grados;
zerod <=((fclk*zero)/100);
process (clk,rst, entrada)
       variable cnt1: integer:= 0;
       variable entrada1: unsigned (7 downto 0);
       variable entrada2: integer range 0 to 180 :=0;
       variable entrada3: integer range 0 to 40000000 :=0;
       variable entrada4: integer range 0 to 40000000 :=0;
    begin
       if rst = '1' then
    salida <= '0';
    elsif rst = '0' then
       if (clk'event and clk = '1') then
          cnt1 := cnt1 + 1;
          if ena=address then
          if entrada>"10110100" then
          entrada4:= dutyhi + zerod;
          elsif entrada <="10110100" then
          entrada1:= unsigned(entrada);
          entrada2:=to_integer( entrada1);
          entrada3:=entrada2*factor;
          entrada4:=entrada3 + zerod;
          else
          entrada4:= entrada4;
          end if;
          end if;
             if (cnt1 <= entrada4) then
                salida<= '1';
             elsif  (cnt1 > entrada4 and cnt1 /= fclk) then
                salida<='0';
             elsif cnt1=fclk then
                cnt1:= 0;
                end if;
       end if;
    end if;
end process;
end Behavioral ;
```