

# Universidad de Alcalá Escuela Politécnica Superior

Complementos de Diseño Electrónico

Práctica final

Control de un motor DC

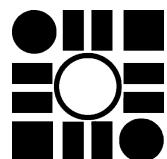
ESCUELA POLITECNICA  
SUPERIOR

**Autor:** David Carrascal Acebron

2021



Máster Universitario en Ingeniería de Telecomunicación



Universidad  
de Alcalá

Madrid, 3 de enero de 2021

# Índice general

<b>1</b>	<b>Introducción</b>	<b>1</b>
1.1	Objetivos . . . . .	1
1.2	Fundamentos teóricos . . . . .	2
1.2.1	Motor de DC . . . . .	2
1.2.2	Puente en H . . . . .	3
1.2.3	Accionamiento de un motor de DC . . . . .	4
1.3	Sistemas Hardware a emplear . . . . .	4
<b>2</b>	<b>Análisis y diseño</b>	<b>7</b>
2.1	Sistema de Control de un motor DC . . . . .	7
2.2	Módulo I - Generación de PWM . . . . .	9
2.3	Módulo II - Selección de PWM . . . . .	11
2.4	Módulo III - Visualización . . . . .	12
2.5	Módulo IV - Velocidad . . . . .	14
2.6	Mejora del sistema . . . . .	16
<b>3</b>	<b>Evaluación</b>	<b>17</b>
3.1	Recursos hardware consumidos . . . . .	17
3.2	Evaluación del sistema . . . . .	18
<b>4</b>	<b>Conclusiones</b>	<b>19</b>
	<b>Bibliografía</b>	<b>21</b>
	<b>Lista de Acrónimos y Abreviaturas</b>	<b>23</b>



# Índice de figuras

1.1	Distribución del sistema a desarrollar . . . . .	2
1.2	Funcionamiento básico de un motor de continua . . . . .	3
1.3	Funcionamiento básico de puente en H . . . . .	4
1.4	Tarjeta Nexys 4 DDR proporcionada en el laboratorio . . . . .	5
1.5	Puente en H, motor y encoder proporcionados en el laboratorio . . . . .	5
1.6	Fuente de alimentación del laboratorio . . . . .	6
2.1	Sistema completo dividido en módulos . . . . .	8
2.2	Interfaz del módulo I . . . . .	9
2.3	Diagrama temporal del módulo I . . . . .	10
2.4	Diseño de la máquina de estados del módulo I . . . . .	10
2.5	Interfaz de usuario en la propia tarjeta Nexys 4 . . . . .	11
2.6	Interfaz del módulo II . . . . .	11
2.7	Diseño propuesto para el módulo II . . . . .	12
2.8	Interfaz del módulo III . . . . .	13
2.9	Diseño del módulo III . . . . .	13
2.10	Interfaz del módulo IV . . . . .	14
2.11	Diseño del módulo IV . . . . .	15
2.12	Diseño de la mejora a implementar . . . . .	16
3.1	Controlador del motor apagado . . . . .	18
3.2	Controlador del motor encendido . . . . .	18
4.1	Vía de trabajo futuro . . . . .	19



# 1 Introducción

En este capítulo de introducción se quiere exponer de manera breve los objetivos del proyecto e indicar los conceptos teóricos más importantes para proyecto, como por ejemplo, cómo funciona un motor de continua comúnmente conocido como un motor Direct Current (DC), cómo se suele accionarlo y qué es un puente en H. Por último, se detallará los distintos sistemas hardware a emplear.

Dado que este capítulo tiene un carácter introductorio, de forma adicional, se quiere indicar la estructura básica que va a seguir la memoria del proyecto haciendo un breve resumen de los aspectos más importantes y significativos de cada capítulo.

**Capítulo 1: Introducción.** Se hará una breve explicación de los aspectos generales del proyecto, y de los objetivos que se quieren alcanzar.

**Capítulo 2: Análisis y diseño.** Se analizará que funcionalidades básicas deben tener los distintos módulos del sistema a desarrollar previo al desarrollo.

**Capítulo 3: Evaluación.** Se indicará el número de *slices* consumidos, multiplicadores, etc. así como una evaluación funcional del sistema desarrollado.

**Capítulo 4: Conclusiones.** Se terminará la memoria con las conclusiones del trabajo realizado, y se presentarán las vías de trabajo futuro que tiene este proyecto.

**Bibliografía y referencias.** Se añadirán todos los artículos, libros, materiales consultados y empleados en la elaboración de esta memoria.

## 1.1 Objetivos

El objetivo de esta práctica final es el diseño y desarrollo de un sistema de control de un motor DC. El desarrollo del sistema se llevará a cabo mediante Field-Programmable Gate Array (FPGA). La funcionalidad de la FPGA, se describirá en software mediante el lenguaje de programación VHSIC-HDL, Very High Speed Integrated Circuit Hardware Description Language (VHDL) visto en la asignatura.

El sistema debe proporcionar la generación de una señal de control de sentido de giro y velocidad del motor DC. Dicho motor estará conectado a un puente en H [1], al cual se le conectarán una tarjeta Nexys 4 DDR [2] para recibir la señal de control, y una fuente de alimentación para proporcionarle el valor de tensión de continua necesario (en este caso 6V). Además, para poder apreciar la velocidad de giro del motor, mediante los displays de la tarjeta se visualizará dicha velocidad ó en su defecto, el ciclo de trabajo de la señal Pulse-Width Modulation (PWM) de control. En la figura 1.1, se contemplarán como se vería dicho sistema al completo.



**Figura 1.1:** Distribución del sistema a desarrollar

## 1.2 Fundamentos teóricos

A continuación, se recogen los fundamentos teóricos necesarios para la realización de la práctica. Con motivo de no aportar información redundante, se han omitido aquellos conceptos teóricos vistos en la asignatura.

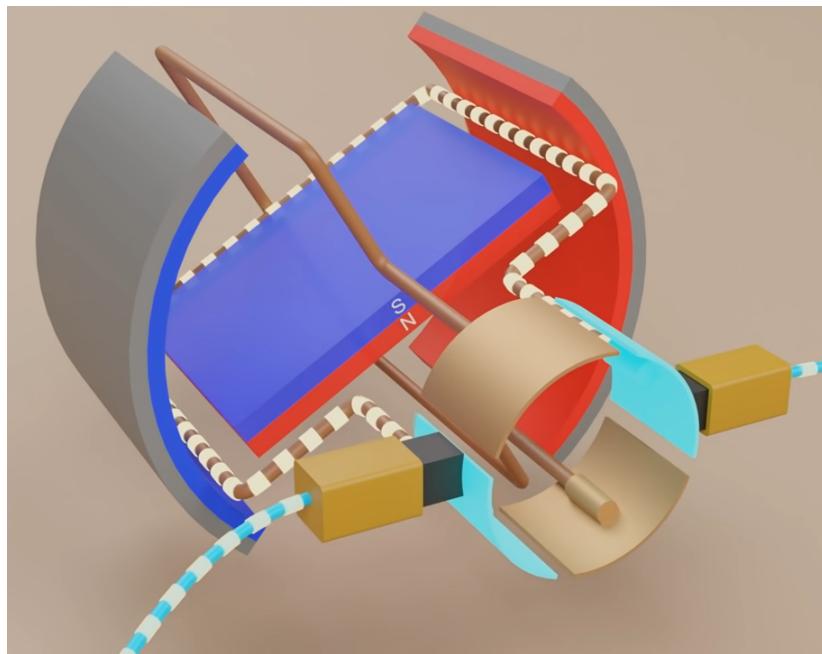
### 1.2.1 Motor de DC

Un motor generalmente es una pieza fundamental de muchos sistemas, que son capaces de transformar algún tipo de energía, por ejemplo de combustibles fósiles ó eléctrica, en energía mecánica. En este caso, se trabajará con un motor de corriente continua, por lo que transformaremos energía eléctrica en energía mecánica en forma de un movimiento rotatorio. Este motor consta de dos partes fundamentales [3] claramente diferenciadas entre sí:

- **Estátor**, la parte fija del motor, suministra un campo magnético constante que atraviesa el rotor.
- **Rotor**, la parte móvil del motor, se puede ver como una serie de bobinas entrelazadas a un anillo de conmutación. Desde dicho anillo se le suministrará una fuente de continua, creando una diferencia de potencial entre sus bornes que logrará que exista una corriente que fluya por la bobina. Teniendo en cuenta ahora el campo magnético constante del estátor, y la corriente que fluye por la bobina, de acuerdo a la ley de Lorentz generará una fuerza magnética.

En el caso de que solo exista una bobina, en el momento de que ésta se sitúe de forma perpendicular con el estátor, habrá un cambio de polaridad en el flujo de la corriente, haciendo que por unos instantes el par<sup>1</sup> tienda a cero , como resultado el movimiento del motor será irregular. Por ello, se suelen añadir varias bobinas al motor para que la rotación lo más suave posible.

<sup>1</sup>Fuerza de empuje que va a tener el eje de salida del motor



**Figura 1.2:** Funcionamiento básico de un motor de continua<sup>2</sup>

### 1.2.2 Puente en H

Es puente en H, conocido en inglés como *H-Bridge*, un sistema para controlar el sentido de giro de un motor DC. Para controlar la dirección en la que gira un motor de corriente continua, se necesita invertir la polaridad de la conexión eléctrica del motor en cuestión [4]. Eso implicaría intercambiar positivo y masa de nuestra fuente de continua, algo inviable cuando se pretende controlar el sistema de forma remota.

Una posible solución, sería la de tener en cada borna del motor de DC tanto positivo como masa. Una de las formas de implementar esta solución sería considerando interruptores que controlen las conexiones a masa y positivo. Por tanto, si se acciona el interruptor de positivo de una borna y el asociado a la masa de la otra, el motor giraría en un sentido. En caso de accionar el otro par de interruptores el motor giraría en sentido contrario. Dicho funcionamiento se puede apreciar en la figura 1.3.

---

<sup>2</sup><https://www.youtube.com/watch?v=CWu1Q1ZSE3c>

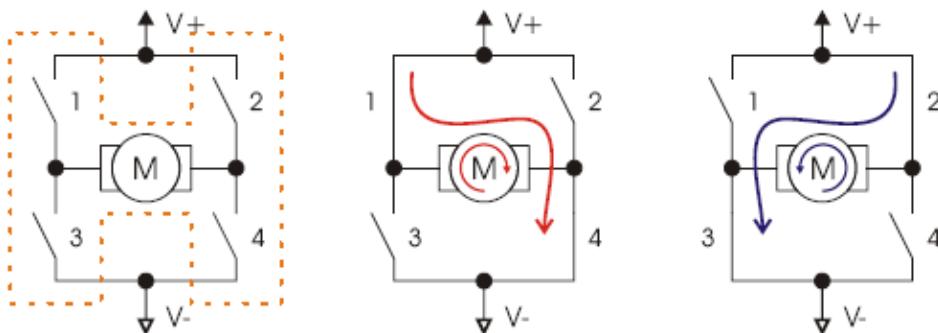


Figura 1.3: Funcionamiento básico de puente en H<sup>3</sup>

La implementación de este tipo de circuitos se puede llevar a cabo con componentes muy diversos, como por ejemplo relés con pulsadores, transistores o con circuitos integrados [5]. Pero al final la idea es la que se ha indicado, controlar la polarización de la alimentación que llega al motor.

### 1.2.3 Accionamiento de un motor de DC

Como ya se ha explicado en la subsección 1.2.2, es posible modificar el sentido de giro del motor de DC con un puente en H. Pero en este caso, para el sistema a desarrollar será necesario también controlar la velocidad de giro del motor.

La velocidad del motor se codificará en el ciclo de trabajo de una señal PWM. El valor medio de dicha señal será la que excitará el motor y definirá la velocidad del mismo. Dado que un puente en H tiene dos ramas, hay que asegurar que en ningún instante pueda producirse un cortocircuito. Para evitar esta situación, se debe asegurar que los interruptores anteriores estén abiertos antes de cerrar los siguientes. Por ello, se introducirá un tiempo muerto, denominado  $T_{blanking}$ .

## 1.3 Sistemas Hardware a emplear

En esta sección se van a describir todos los componentes hardware necesarios para implementación del sistema a desarrollar.

- **Tarjeta Nexys 4 DDR**, será modulo principal de control desde el cual se podrá controlar la velocidad y sentido de giro de motor, a través de sus switches. Además, se utilizarán sus displays para visualizar la velocidad de giro del motor ó en su defecto el porcentaje de ciclo de trabajo de la señal PWM generada.

La FPGA de la tarjeta se programará desde el entorno de trabajo proporcionado por Xilinx, Vivado 2017.4, a través de la interfaz serie del ordenador.

<sup>3</sup><https://www.pololu.com/docs/0J21/5.c>



Figura 1.4: Tarjeta Nexys 4 DDR proporcionada en el laboratorio

- **PMOD HB5**, el puente en H, que se conectará a la tarjeta Nexys 4 por el conector PMOD JA. Además, dicho módulo incorpora la toma de continua por lo que los conectores de la fuente irán directos a él.
- **Motor DC** de 6V con las siguientes características:
  - Rango de tensión entrada  $6\text{ V} \pm 10\%$  DC
  - Corriente máxima sin carga 220 mA
  - Velocidad máxima sin carga 150 rpm  $\pm 15\%$
- **Encoder**, incorporado en la parte trasera del motor de DC. Será el encargado de indicarnos las revoluciones reales que está dando el motor. Está compuesto de dos canales Hall (SA, SB) y dan tres vueltas por pulso cada canal.

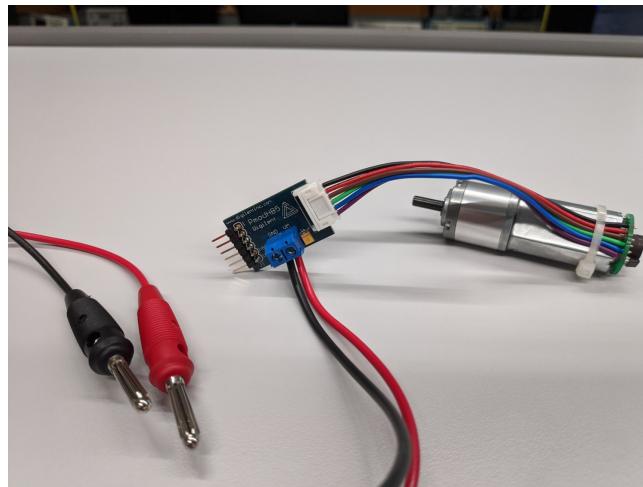
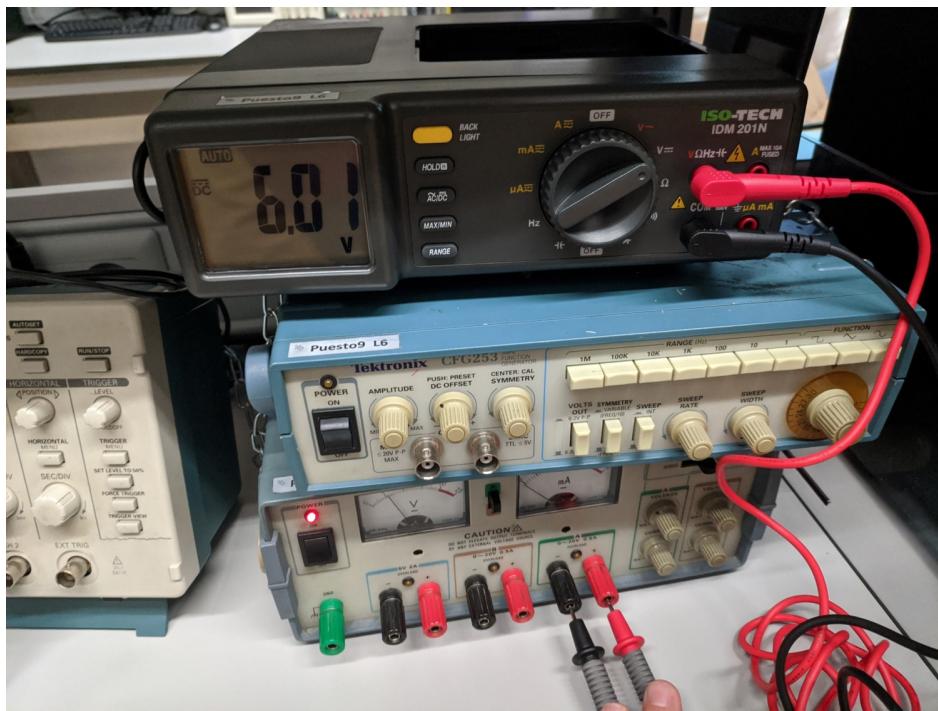


Figura 1.5: Puente en H, motor y encoder proporcionados en el laboratorio

---

- **Fuente de alimentación**, se utilizará la fuente de alimentación del laboratorio para suministrar los 6V necesarios al PMOD HB5.



**Figura 1.6:** Fuente de alimentación del laboratorio

## 2 Análisis y diseño

En este capítulo, teniendo en cuenta la introducción teórica realizada en el capítulo anterior, se analizará qué funcionalidades básicas deben ser implementadas por el sistema completo, se dividirá el sistema en diferentes bloques con el fin de trabajar de forma modular y autónoma cada uno de los integrantes del equipo. Cada bloque, se verá como un modulo, en este capítulo veremos que requerimientos debe cumplir cada uno de ellos y que diseño se ha planteado para lograr tales requerimientos.

### 2.1 Sistema de Control de un motor DC

El sistema que se desea realizar debe controlar la excitación de un motor DC conectado al puente en H a través de la FPGA. El sistema de control generará 2 señales para el puente en H, la señal PWM y a señal de sentido de giro. La señal PWM deberá tener un ciclo de trabajo (D) configurable, que posteriormente el usuario podrá seleccionar qué porcentaje de ciclo de trabajo desea a través de los switches de la tarjeta.

Como se puede apreciar, es un sistema bastante complejo, por lo que al ser pocos en clase el profesor orientó la práctica a un diseño modular, donde cada módulo del sistema debía ser desarrollado por cada alumno. Por lo tanto, todos los diseños son libres y lo único que se debe respetar es la entidad jerárquica superior indicada en el bloque 2.1.

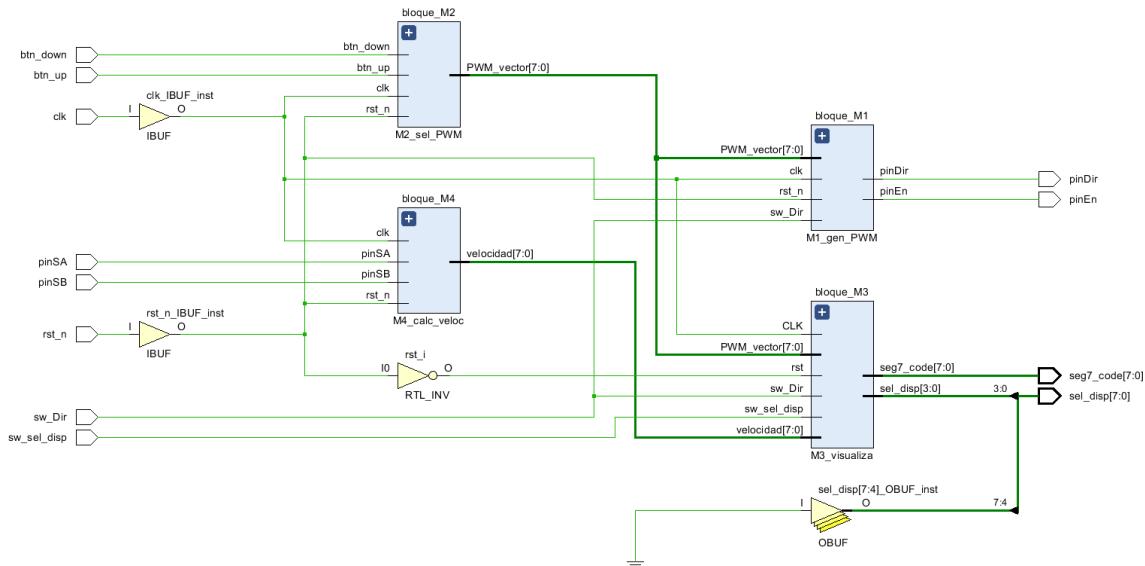
Código 2.1: Entidad jerárquica superior del sistema a desarrollar

```
1 entity contr_motor is
2   port (
3     clk : in std_logic; -- reloj de 100 MHZ
4     rst_n : in std_logic; -- reset del sistema (síncrono a nivel bajo)
5     btn_up : in std_logic; -- pulsador de incremento de D(%)
6     btn_down : in std_logic; -- pulsador de decremento de D(%)
7
8     **** + botones
9
10    sw_Dir : in std_logic; -- switch (1) para sentido de giro
11    sw_sel_disp : in std_logic; -- switch (1) para selección de info-display
12
13    **** + switches
14
15    pinSA : in std_logic; -- entrada Sensor A Encoder (PMOD)
16    pinSB : in std_logic; -- entrada Sensor B Encoder (PMOD)
17    pinEn : out std_logic; -- salida PWM para el puente en H (PMOD)
18    pinDir : out std_logic; -- sentido giro del motor (PMOD)
19    seg7_code : out std_logic_vector (7 downto 0); -- bus de 7 segmentos
20    sel_disp : out std_logic_vector (7 downto 0)); -- bus de anodos de los displays
21 end entity contr_motor;
```

Debido a lo cual, los cuatro bloques principales a desarrollar y sus encargados de desarrollarlos son los siguientes:

- Módulo de generación de PWM, **M1\_gen\_PWM**.
  - Encargado de desarrollarlo: David Carrascal Acebron.
  - Sección: 2.2
- Módulo de seleccionar PWM, **M2\_sel\_PWM**.
  - Encargado de desarrollarlo: Victoria Noci Luna.
  - Sección: 2.3
- Módulo de Visualización, **M3\_visualiza**.
  - Encargado de desarrollarlo: Mohamed Malki.
  - Sección: 2.4
- Módulo de cálculo de velocidad, **M4\_calc\_veloc**.
  - Encargado de desarrollarlo: Alfredo Gardel Vicente.
  - Sección: 2.5

A continuación, en la figura 2.1, se puede apreciar la topología estructural de los módulos con su conexionado en el sistema final. De forma adicional, se ha propuesto realizar alguna mejora sobre el proyecto, por lo que es posible que dicha topología se vea modificada a la hora implementar la mejora.



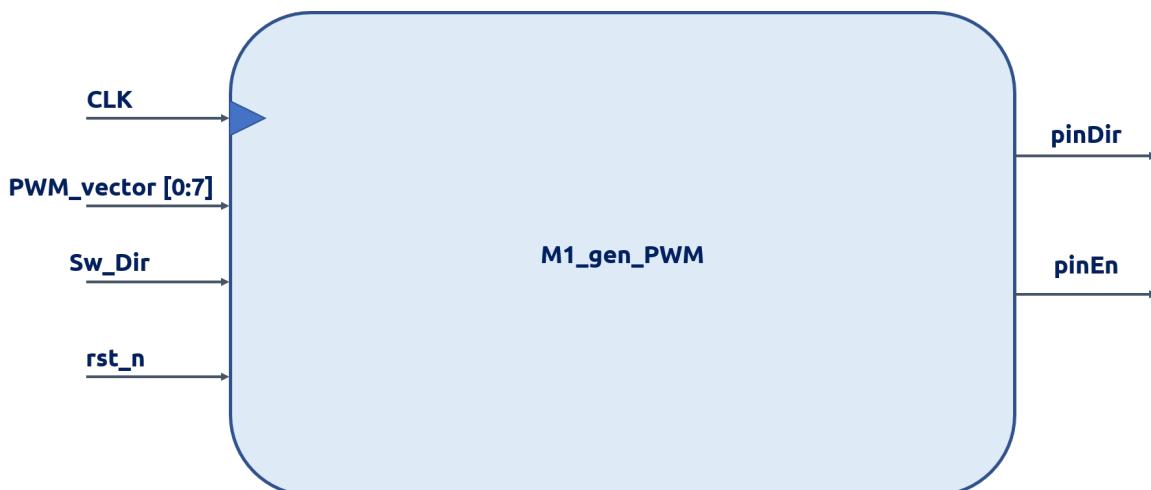
**Figura 2.1:** Sistema completo dividido en módulos

## 2.2 Módulo I - Generación de PWM

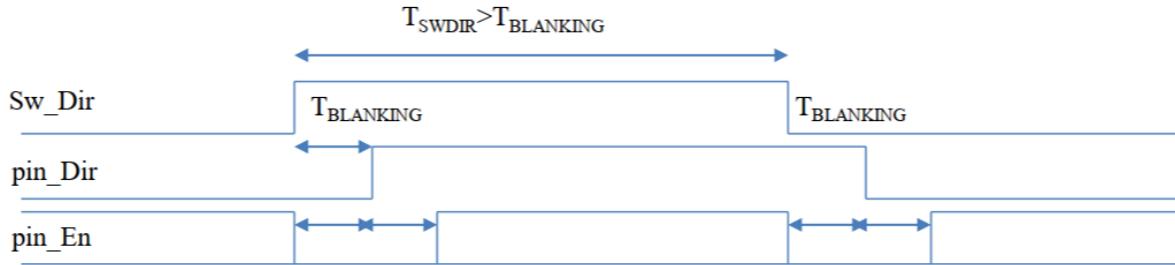
El módulo de generación de PWM se encargará de generar la señal PWM que ataque al puente en H. Los requerimientos de dicho modulo son los siguientes:

- El **PWM\_vector** viene codificado con 8 bits, donde el valor 0 significará un valor del ciclo de trabajo  $D = 0\%$  y un valor de 100 equivaldrá a  $D = 100\%$ . Este valor de ciclo de trabajo **PWM\_vector** se generará posteriormente mediante los pulsadores del módulo **M2\_sel\_PWM**.
- La frecuencia del PWM es también será importante. Ésta será constante en el tiempo y se fijará a 2000 Hz. Por lo que ya sabemos que el periodo de la PWM será de  $T_{PWM} = 0.5 \text{ ms}$ .
- Como ya se explico en la sección 1.2, podríamos romper el sistema si hacemos cambios bruscos en la direccionalidad del giro del motor. Por lo tanto, se dejará un tiempo muerto (lo denominaremos, *Blanking Time*) cuando se desee cambiar el giro del motor. Se va a considerar un tiempo de  $T_{blanking} = 10 \text{ ms}$ .
- El cambio de dirección del motor se le indicará al módulo por la entrada **sw\_Dir**, y cuando se haya cumplido el tiempo muerto de guarda, se le indicará al puente en H por el pin de salida **pinDir**.
- Por otra parte, la señal PWM que se aplica al motor, **pinEn**, cuando haya un cambio en el sentido de giro, valdrá 0 durante  $2 \times T_{blanking}$ .

Por tanto, si atendemos a todos los requerimientos expuestos en el guión del proyecto, veremos que debemos cumplir cronograma expuesto en la figura 2.3 y cumplir con la interfaz que se puede apreciar en la figura 2.2.



**Figura 2.2:** Interfaz del módulo I



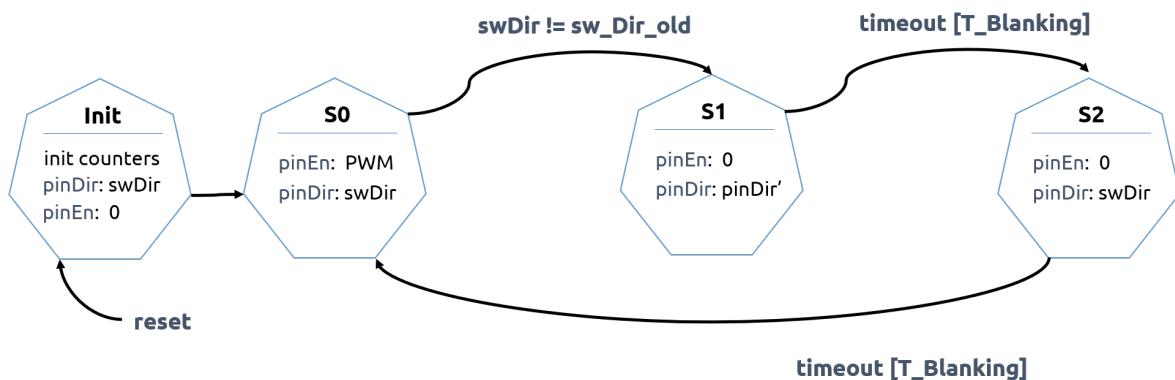
**Figura 2.3:** Diagrama temporal del módulo I

El diseño que se propone para este módulo es hacer una máquina de estados, ver figura 2.4, donde cada estado es un intervalo diferente del cronograma. Se iniciará de un estado inicial (*init*), un estado seguro donde iremos siempre y cuando se accione la señal de reset. En este estado únicamente permaneceremos un único ciclo de reloj, si únicamente se ha accionado el botón de reset una vez. Si el botón asociado al reset si mantiene pulsado permaneceremos en dicho estado hasta que se deje de pulsar.

El siguiente estado, llamado *s0*, será el estado fundamental del módulo donde se generará la PWM indicada por el vector de **PWM\_vector** y se le suministrará la dirección de giro tal cual le llega al puente en H.

En caso de que haya un cambio de dirección, es decir la entrada al módulo llamada **swDir** es diferente entre dos ciclos de reloj contiguos, pasaremos al estado llamado *s1*. En dicho estado debemos dejar de generar la PWM y mantener la señal de salida **pinDir** un  $T_{blanking}$ .

Una vez se hay cumplido dicho *timeout* de valor un  $T_{blanking}$ , pasaremos al siguiente estado *s2*, el cual actualizará el valor de la salida de **pinDir** con el valor actual de **swDir** y mantendrá un  $T_{blanking}$  más la señal de la PWM a 0. Cuando se haya alcanzado dicho *timeout*, volveremos al estado fundamental *s0*.

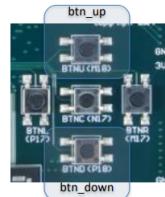


**Figura 2.4:** Diseño de la máquina de estados del módulo I

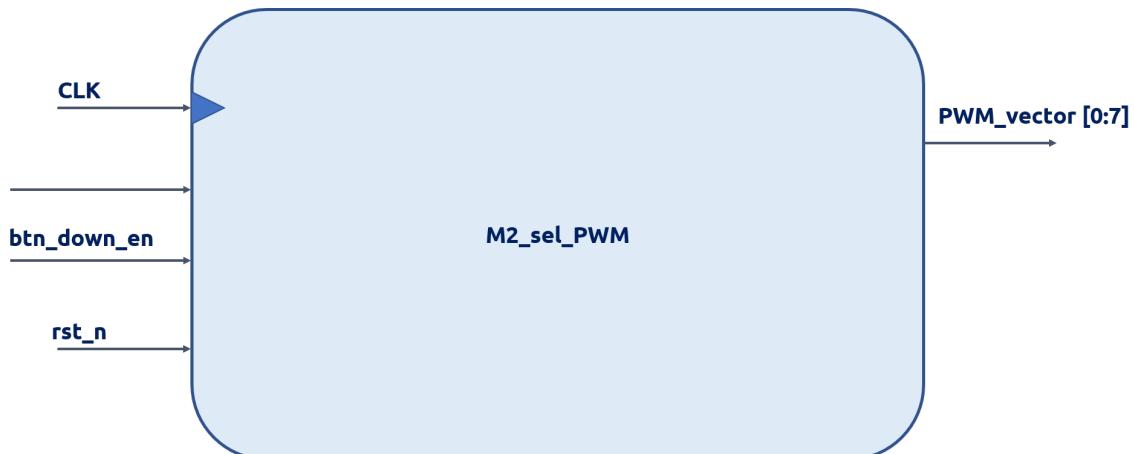
## 2.3 Módulo II - Selección de PWM

El módulo de la selección de la PWM se encargará de leer los pulsadores de la tarjeta Nexys 4 (`btn_up`, `btn_down`) los cuales incrementarán y decrementarán un vector que le indique el ciclo de trabajo para la generación de la PWM. Los requerimientos de este módulo son los siguientes:

- Inicialmente el valor del vector que gobierne el ciclo de trabajo, llamado `PWM_vector`, será igual a 0.
- Cada pulsación sobre `btn_up` o `btn_down` hará aumentar o disminuir el valor de 1 en 1, respetando siempre los límites 0 y 100 para así no superar nunca el valor máximo/mínimo del ciclo de trabajo posible.
- Si el pulsador se mantiene pulsado, el valor de `PWM_vector` se incrementará/decrementará de forma continua cada 250 ms.



**Figura 2.5:** Interfaz de usuario en la propia tarjeta Nexys 4

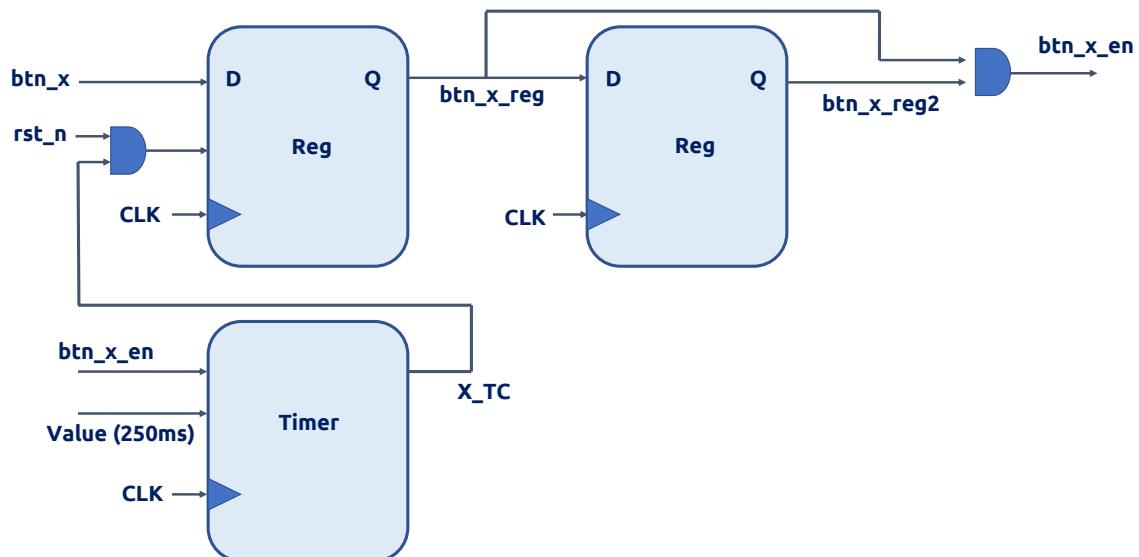


**Figura 2.6:** Interfaz del módulo II

Como se puede ver la figura 2.5, la interfaz de usuario utilizada en la Nexys 4 serán los botones superior e inferior correspondientes a `btn_up` y `btn_down`. Por lo tanto, teniendo los requerimientos anteriormente expuestos, se deberá respetar la interfaz que se indica en la figura 2.6.

El diseño propuesto para satisfacer todos los requerimientos del módulo se puede ver en la figura 2.7. Donde se tiene un primer registro para sincronizar los cambios en los pulsadores con el `clk`, y un segundo registro a modo de detector de flancos.

Además, para satisfacer el requerimiento de un incremento constante si se mantiene presionado el pulsador durante 250 ms, se ha añadido un contador descendente que hacer *match* cuando se cumpla el *timeout* de los 250ms. En este *match*, se habilitará la suma/resta del valor decimal del vector `PWM_vector`. El diseño es completamente análogo para ambos botones.



**Figura 2.7:** Diseño propuesto para el módulo II

## 2.4 Módulo III - Visualización

El módulo de visualización se encargará de visualizar en los displays la velocidad de giro del motor o el ciclo de trabajo. Los requerimientos de este módulo son los siguientes:

- Mediante el pin `sw_sel_disp` se seleccionará qué variable ver.
- También mostrará el sentido de giro con un signo “-” (menos) en el display de mayor peso.
- Por tanto se emplearán solo 4 de los 8 displays disponibles en la tarjeta.

Atendiendo a los requerimientos expuestos tendremos que seguir con la siguiente interfaz del módulo, la cual se puede ver en la figura 2.8.

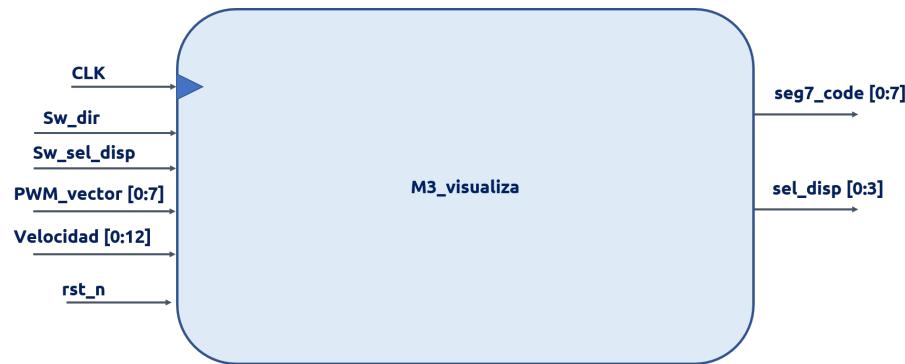


Figura 2.8: Interfaz del módulo III

El diseño propuesto para cumplir con todos los requerimientos del módulo y visualizar correctamente tanto velocidad como porcentaje de ciclo de trabajo de la PWM se puede apreciar en la figura 2.9.

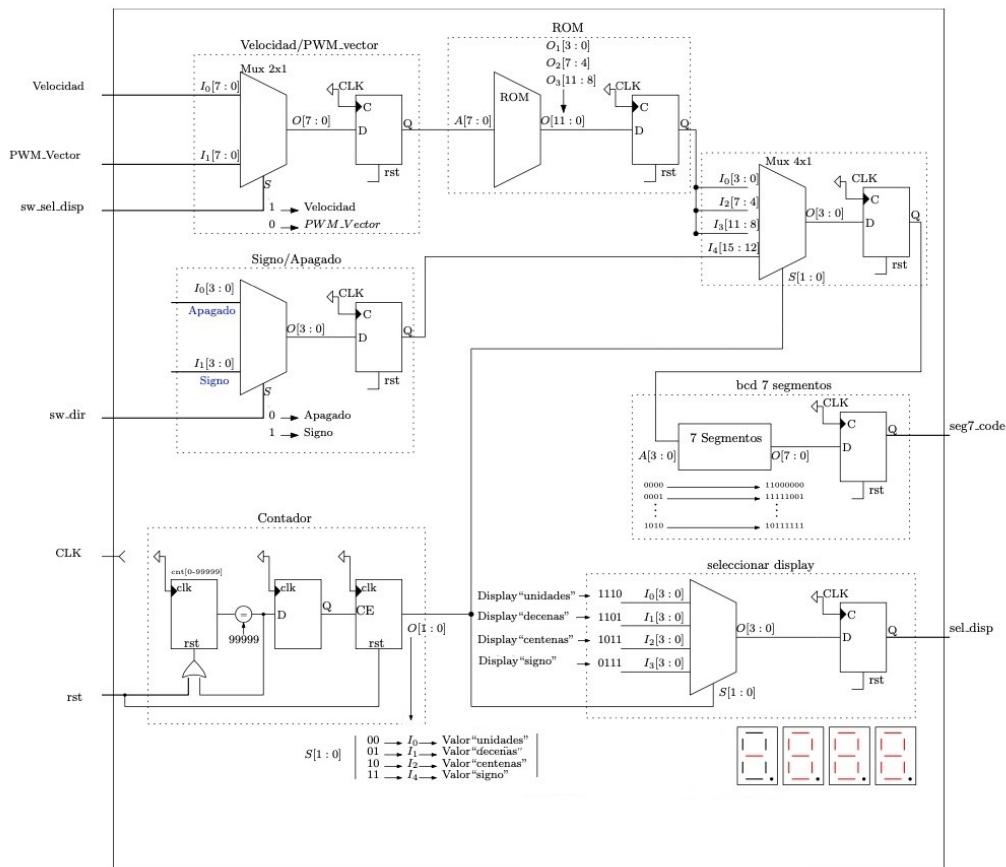


Figura 2.9: Diseño del módulo III

Como se puede ver en la figura 2.9, tenemos un modulo para multiplexar que queremos representar en los displays, si la velocidad o el vector PWM\_vector en base a la señal sw\_sel\_disp. Acto seguido, buscamos en la ROM el valor de representación en los displays asociado a dicha velocidad/vector. Por último, empleamos un conversor a bdc de 7 segmentos para finalmente representar en los displays.

De forma paralela, se va actualizando el valor del signo en función de la señal de dirección de giro del motor llamada sw\_Dir. El valor del signo se multiplexa junto al valor de representación, y será seleccionado cuando se seleccione el display de signo.

Por último, hay un módulo que va generando una señal de forma periódica de un 1 KHz. Esta señal será la encargada de ir seleccionando los displays de unidades, decenas, centenas y signo, e ir actualizando dichos valores cuando toque. Por tanto, podemos afirmar que la tasa de refresco es de 1ms, pero si nos fijamos en un único display, su tasa de refresco sería de 4ms dado que tiene que actualizar los otros tres displays, y el suyo propio.

## 2.5 Módulo IV - Velocidad

El último módulo se encargará de calcular la velocidad real del motor en base a la información procedente de los sensores Hall de la tarjeta del puente en H (pinSA, pinSB). Los requerimientos de este módulo son los siguientes:

- A partir de la relación del número de vueltas del motor se debe determinar la velocidad del motor.

Teniendo en cuenta el único requerimiento, y atendiendo a los pines que nos vienen desde el encoder, la interfaz que se deberá seguir la indicada en la figura 2.10.



**Figura 2.10:** Interfaz del módulo IV

El diseño propuesto para este modulo es el que se puede apreciar en la figura 2.11. Este consta únicamente en un temporizador, un sumador y un bloque que multiplique por 1,5. Esto es así, ya que si tomamos el numero de cuentas en un intervalo de 0,25 segundos, teniendo en cuenta que se emiten tres pulsos por vuelta, se puede ver:

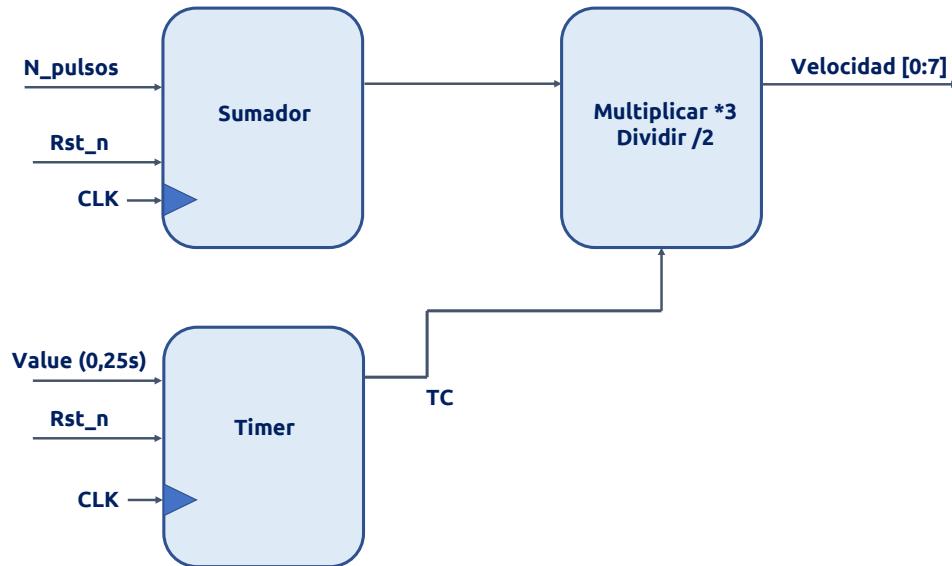
$$V_{max} = 150 \text{ r.p.m}; \text{ Reductora : } \frac{1}{53}; \quad (2.1)$$

$$\text{Pulsos} \times \text{Vuelta} = 3; \text{ base}_{sg} = 4 \quad (2.2)$$

$$\text{Velocidad} = \frac{n\_pulsos\_reg \times \text{base}_{sg} \times 60 \text{ sg}}{\text{reductora} \times \text{Pulsos} \times \text{Vuelta}} = \frac{n\_pulsos\_reg \times 4 \times 60 \text{ sg}}{53 \times 3} \quad (2.3)$$

$$\text{Velocidad} = n\_pulsos\_reg \times 1.5 = \frac{n\_pulsos\_reg \times 3}{2} \quad (2.4)$$

Que podemos obtener la velocidad simplemente multiplicando por una constante el número de cuentas almacenadas en un intervalo de 0,25 segundos.

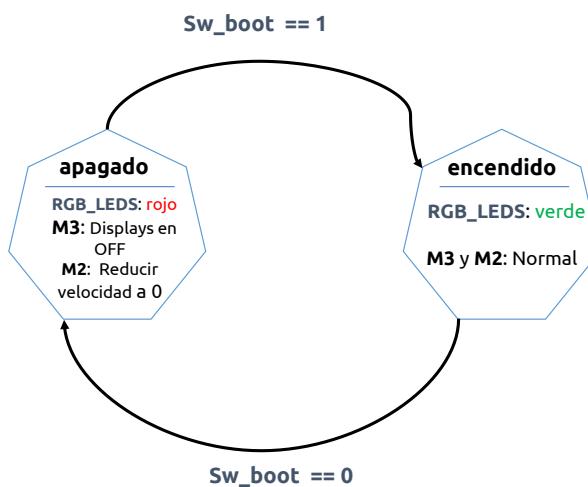


**Figura 2.11:** Diseño del módulo IV

## 2.6 Mejora del sistema

Se propone una mejora sobre el diseño actual del sistema. Esta mejora consiste en un controlador de arranque del motor, el cual a través de un switch de la tarjeta, a modo de llave, encenderá el sistema o lo apagará. Para que exista una mejor interfaz con el usuario, se hará uso de elementos visuales de la tarjeta para indicar el estado del sistema. En este caso se han decidido indicar en los displays donde se representa la velocidad, la palabra *OFF* en caso de que el sistema se encuentre apagado y la velocidad cuando se encuentre encendido.

Esta mejora impacta directamente sobre los módulos, M2\_sel\_PWM y M3\_visualiza, por lo que habrá que modificarlos. Dado que en el laboratorio se comprobó previamente que el diseño realizado del sistema funcionaba correctamente, se integró la mejora a modo de un estado nuevo. Donde el funcionamiento anterior del sistema será el estado *encendido* además de encender los leds rgb a verde, y el estado apagado consistirá en cambiar los leds RGB a rojo, mostrar en los displays *OFF* (Modificación en M3\_visualiza) y en caso de que el motor esté girando a una velocidad mayor que cero, ir reduciendo en una unidad el valor de la PWM cada 40ms hasta llegar a cero (Modificación en M2\_sel\_PWM). En la figura 2.12, puede apreciar dicho diseño.



**Figura 2.12:** Diseño de la mejora a implementar

## 3 Evaluación

Siguiendo el guión de la práctica se van a indicar el número de slices consumidos, multiplicadores y demás recursos hardware de nuestra tarjeta por cada de los módulos diseñados. Además, se proporcionará de una evaluación del sistema completa.

### 3.1 Recursos hardware consumidos

A continuación, se indican los recursos hardware ocupados por nuestro sistema. En la tabla 3.1 se indican las slices consumidas, en la tabla 3.2 puede consultar los módulos de memoria consumidos. Por último, en la tabla 3.3 puede comprobar los Digital Signal Processor (DSP) utilizados.

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	233	0	63400	0.37
LUT as Logic	233	0	63400	0.37
LUT as Memory	0	0	19000	0.00
Slice Registers	189	0	126800	0.15
Register as Flip Flop	189	0	126800	0.15
Register as Latch	0	0	126800	0.00
F7 Muxes	0	0	31700	0.00
F8 Muxes	0	0	15850	0.00

Tabla 3.1: Slice Logic

Site Type	Used	Fixed	Available	Util%
Block RAM Tile	0.5	0	135	0.37
RAMB36/FIFO*	0	0	135	0.00
RAMB18	1	0	270	0.37
RAMB18E1 only	1	0	270	0.37

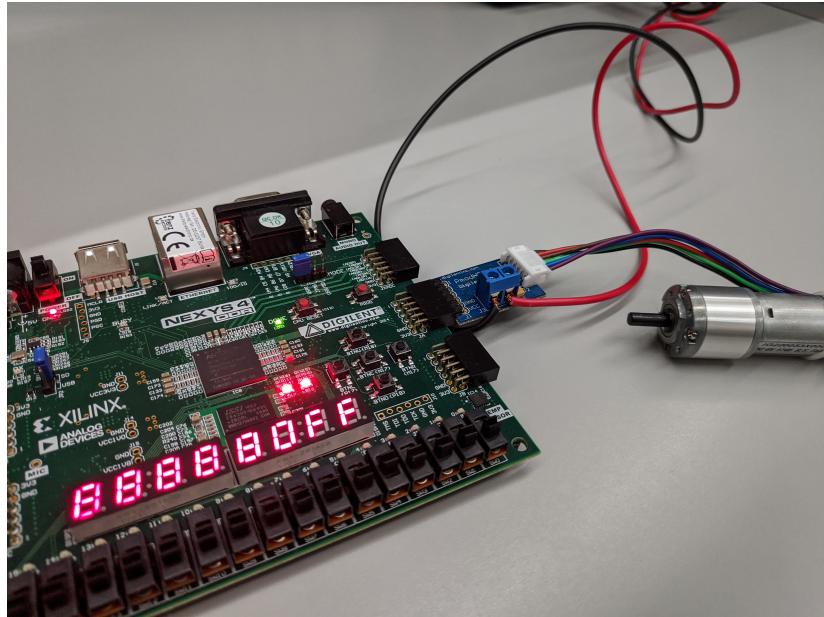
Tabla 3.2: Memory

Site Type	Used	Fixed	Available	Util%
DSPs	0	0	240	0.00

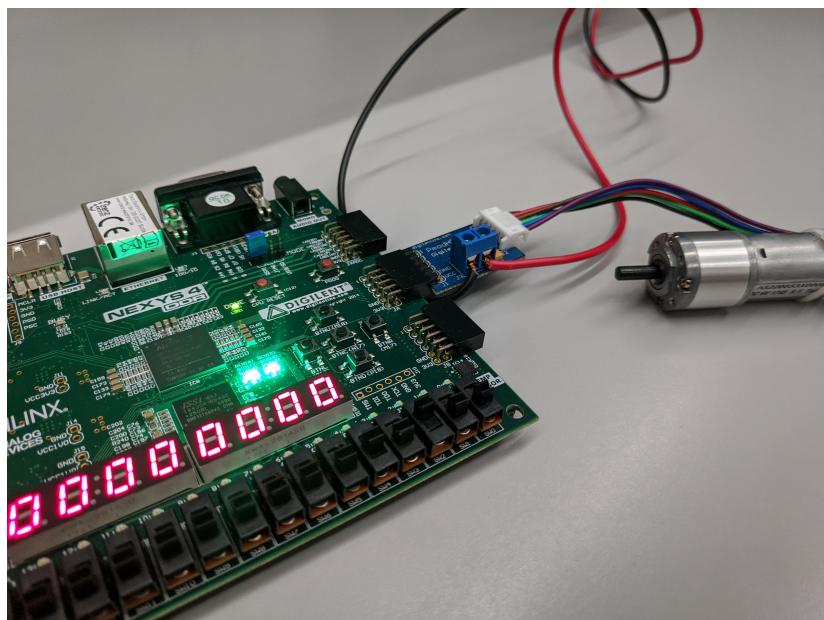
Tabla 3.3: DSPs

## 3.2 Evaluación del sistema

Dado que en una memoria es difícil plasmar una evaluación funcional del sistema, se ha grabado un vídeo donde se va probando paso a paso el sistema. Puede acceder al vídeo haciendo click [aquí](#).



**Figura 3.1:** Controlador del motor apagado



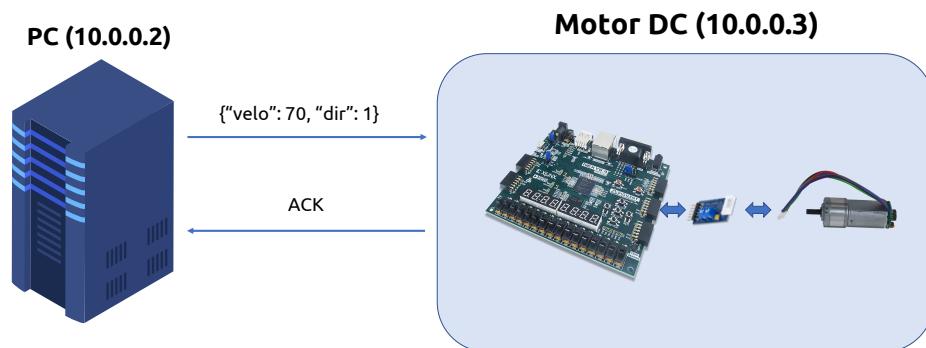
**Figura 3.2:** Controlador del motor encendido

## 4 Conclusiones

En esta práctica se ha podido ver en detalle las bondades y complicaciones que supone trabajar con dispositivos hardware programables. Se ha aprendido a como afrontar desarrollos de sistemas más complejos, particionando el sistema total en distintos módulos. Analizando así cada modulo y probando cada uno por separado, para finalmente integrarlos en uno solo.

Se quiere recalcar que además de haber podido poner en práctica los contenidos teóricos de la asignatura y aprender los fundamentos de motores de DC, destacar la posibilidad que nos ofreció el profesor de la asignatura al poder trabajar como equipo en el proyecto. Creando así un ambiente, en opinión del autor muy positivo, asemejándolo bastante al ambiente de trabajo en entornos profesionales del sector TIC.

Como vía de trabajo futuro se propone dar al proyecto un enfoque más cercano al ámbito de las telecomunicaciones, concretamente al ámbito de la telemática. Dado que la tarjeta Nexys 4 tiene un conector RJ-45 de Ethernet se propone, ver figura X, hacer uso de un algún subsistema ya programado<sup>1</sup> o algún stack TCP/IP que permita utilizar más fácilmente la interfaz de Ethernet para poder dotar de una interfaz en red del control del motor.



**Figura 4.1:** Vía de trabajo futuro

<sup>1</sup>[https://www.xilinx.com/support/documentation/application\\_notes/xapp1026.pdf](https://www.xilinx.com/support/documentation/application_notes/xapp1026.pdf)

Es decir, en vez de utilizar la interfaz que tiene la tarjeta Nexys 4 con los switches para subir o bajar la velocidad, y los displays para representar dicha velocidad del motor, se plantea que el sistema sea capaz de recibir la información de control en red. Que le llegue por ejemplo, un paquete con información de control en plano, que le indique el porcentaje de PWM a generar, y sentido de giro.

Una vez se desarrolle dicha interfaz en red, sería casi inmediato crear algún tipo de protocolo para separar peticiones con información control para modificar la velocidad y giro del motor, de peticiones de datos para pedir a la tarjeta que nos diga a qué velocidad real está girando el motor.

# Bibliografía

- [1] Digilent, “PMOD HB5 Reference Manual [Digilent Documentation].” [Online]. Available: <https://reference.digilentinc.com/reference/pmod/pmodhb5/reference-manual>
- [2] D. Forum, “Nexys 4 DDR [Digilent Documentation].” [Online]. Available: <https://reference.digilentinc.com/reference/programmable-logic/nexys-4-ddr/start>
- [3] P. Yedamale, “Brushless dc (bldc) motor fundamentals,” *Microchip Technology Inc*, vol. 20, pp. 3–15, 2003.
- [4] Bobby Schulz, “H-Bridge Basics.” [Online]. Available: [https://www.youtube.com/watch?v=l4Ty92JittY&ab\\_channel=bobbyschulz](https://www.youtube.com/watch?v=l4Ty92JittY&ab_channel=bobbyschulz)
- [5] N. Briceño, “Controlar motores de corriente continua con Puente H.” [Online]. Available: <http://www.arteymedios.org/tutoriales/item/76-controlar-motores-de-corriente-continua-con-puente-h>
- [6] L. Petru and G. Mazen, “Pwm control of a dc motor used to drive a conveyor belt,” *Procedia Engineering*, vol. 100, pp. 299–304, 2015.



## **Lista de Acrónimos y Abreviaturas**

**DC** Direct Current.

**DSP** Digital Signal Processor.

**FPGA** Field-Programmable Gate Array.

**PWM** Pulse-Width Modulation.

**VHDL** VHSIC-HDL, Very High Speed Integrated Circuit  
Hardware Description Language.

Universidad de Alcalá  
Escuela Politécnica Superior



ESCUELA POLITECNICA  
SUPERIOR

