UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DISEÑO E IMPLEMENTACIÓN DEL CONTROL ELECTRÓNICO DIGITAL DEL MOTOR DE UN COCHE ELÉCTRICO

AUTOR: BACETE PÉREZ, MARIO

TUTOR: ESTEVE BOSH, RAÚL

COTUTORA: ROMERO PÉREZ, LUCÍA

Curso Académico: 2013-14

Índice General

Memoria

1. Objeto del proyecto

Durante el proyecto se pretende diseñar el control digital síncrono del motor DC presente en un coche eléctrico utilizando una FPGA CYCLONE IV E implementada en una tarjeta DE0-NANO.

El objetivo del proyecto será un sistema que permita controlar la puesta en marcha, apagado y velocidad del motor de un coche mediante potenciómetros (simulando acelerador y freno) con un modo de control de crucero automático que también permita regular la velocidad de referencia.

La comunicación con el usuario se establecerá mediante una botonera y un display. El usuario seleccionará la puesta en marcha, el apagado, la velocidad del motor del coche y el programa de la velocidad de crucero mediante la botonera. A su vez, en el display se mostrará la velocidad del motor, el sentido de la marcha y si está activado el programa crucero.

Para ello, se utilizará junto con la tarjeta DE0-NANO, un display LCD HITACHI HD44780U y una etapa de potencia en puente H. También se hará uso de un programa que controla los disparos de la etapa de potencia.

En primer lugar se diseñará la lógica del circuito para la medición de velocidad, determinación del sentido de giro, control de crucero y control del display. A continuación el diseño se programará utilizando el software de diseño Quartus II y se simulará con QSIM. Posteriormente se implementará el diseño en la tarjeta.

Finalmente se realizarán las pruebas en la DE0-NANO, que está conectada a la botonera, al display y a la etapa de potencia en H que controla el motor DC, puliendo el diseño para su uso real.

2. Antecedentes, motivación y justificación.

2.1 Motores Eléctricos en la Automoción.

Los motores eléctricos están presentes en ya muchos vehículos de calle y previsiblemente serán el futuro de los sistemas de transporte.

El primer vehículo eléctrico apareció en torno a 1830 desarrollado por el escocés Robert Anderson.

La irrupción posterior del motor de 4 tiempos y la dificultad de almacenar energía eléctrica hicieron que su desarrollo en automoción se ralentizase durante el siglo XX, pero no así en otros campos donde sí se empleó.

Llegado al siglo XXI la conciencia ecológica que se extendió sobre todo por Europa y América del norte reavivó el interés por el motor eléctrico en los vehículos de calle.

Finalmente fue una empresa japonesa, Toyota, quien gracias a las ayudas del gobierno americano desarrolló el primer vehículo híbrido comercializable y funcional. Y tras superar los problemas de los primeros modelos se convirtió en un éxito.

Hoy en día, más fabricantes se han sumado a los vehículos híbridos como Lexus, Porche o Renault. Así como a los vehículos totalmente eléctricos como la nueva empresa Tesla que desarrolla vehículos 100% eléctricos de altas prestaciones.

También en el mundo de la alta competición están siendo introducidos este tipo de motores como en la F1 con los sistemas de recuperación de energía o la Formula E con monoplazas totalmente eléctricos.

Ya sea en modo híbrido (junto a un motor de combustión de apoyo) o eléctricos puros, estos requieren de sistemas digitales que los hagan tan cómodos de conducir como los actuales vehículos de combustión, permitiendo a los usuarios cambiar la tecnología del vehículo sin que ello suponga un cambio en su forma de conducir.

2.2 Control de crucero

El control de crucero nace, inicialmente, de la necesidad de mantener constante el régimen de giro de la máquina de vapor y para ello se desarrolló el regulador centrífugo.

Basado en dos masas conectadas a las válvulas de entrada del vapor, la fuerza centrifuga separa o junta las masas cerrando o abriendo respectivamente la válvula de entrada de vapor y con ello el régimen de giro.

La implementación de este sistema en vehículos con motores térmicos se hizo controlando la válvula de admisión, y permitiendo mantener la velocidad constante incluso en pendientes.

Finalmente, el control de velocidad como lo conocemos hoy en día se desarrolló 1945 por el ingeniero mecánico Ralph Teetor quien introdujo medios electromagnéticos (una bobina) para la regulación del acelerador.

Los sistemas modernos se basan en la tecnología PID.

El PID es un controlador que se basa en tres algoritmos, el primero proporcional se encarga de, ante un cambio en el régimen de carga, y el error asociado a este, dar una señal de control proporcional al error.

La parte integral calcula el error acumulado que hay una vez estamos fuera de la referencia y genera una señal de control proporcional a este que es sumado junto con la señal de control proporcional.

Y la parte derivativa trata de predecir la acción de control que devuelva al sistema a la referencia mediante la tendencia que sigue el error en los instantes anteriores. Con esta tendencia, genera una señal de control que también se sumará a las dos anteriores

Actualmente este sistema interactúa con otros como medidores de distancia (láser o sonar) y cámaras para adaptar automáticamente la velocidad a las condiciones de tráfico y velocidades máximas y mínimas de cada vía.

2.3 Motores de corriente continua

El principio de funcionamiento de todo motor eléctrico es la ley de Lorentz.

Cuando un conductor por el que pasa una corriente eléctrica se sumerge en un campo magnético, el conductor sufre una fuerza perpendicular al plano formado por el campo magnético y la corriente.

$$F = B * l * I$$

Siendo:

- F: Fuerza en Newton.
- B: Densidad del campo magnético en Teslas.
- l: Longitud del conductor en metros.

- I: Intensidad de la corriente por el conductor en amperios.

En su construcción física, para aplicar la ley de Lorenz se emplean imanes y electroimanes distribuidos en las dos partes fundamentales de un motor eléctrico, estator y rotor.

El rotor (parte móvil) está compuesto por una serie de bobinas (l) de conductores recorridas por una intensidad variable (I) encargada de generar los campos eléctricos variables, estos deben cambiar su polaridad consecutivamente para que la fuerza generada siempre sea favorable al giro del motor y para ello, una de las tecnologías más empleadas son los anillos rozantes.

El estator está compuesto por dos imanes fijos encargados de generar un campo magnético (B) dentro del cual gire el rotor.

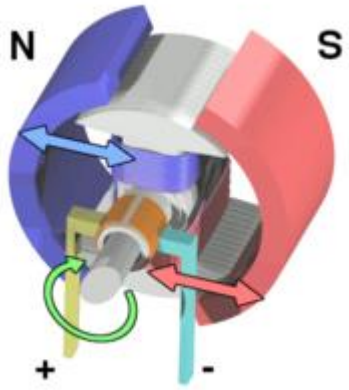

Ilustración 1. Esquema de un motor DC con la representación de los polos magnéticos.

2.4 Control de motores de corriente continua

La velocidad en un motor de continua se regula mediante el nivel de tensión que se le dispone en bornes. Pero en la práctica, es costoso el variar la tensión en corriente continua.

Por lo tanto, para controlar el motor se emplea una onda cuadrada, de nivel alto a la tensión de continua y de nivel bajo a tensión cero, de manera que controlando el tiempo que esta señal pasa a nivel alto y nivel bajo se controla el valor eficaz de la tensión y con ello, el régimen de giro. Este sistema de control, se implementa con los siguientes elementos:

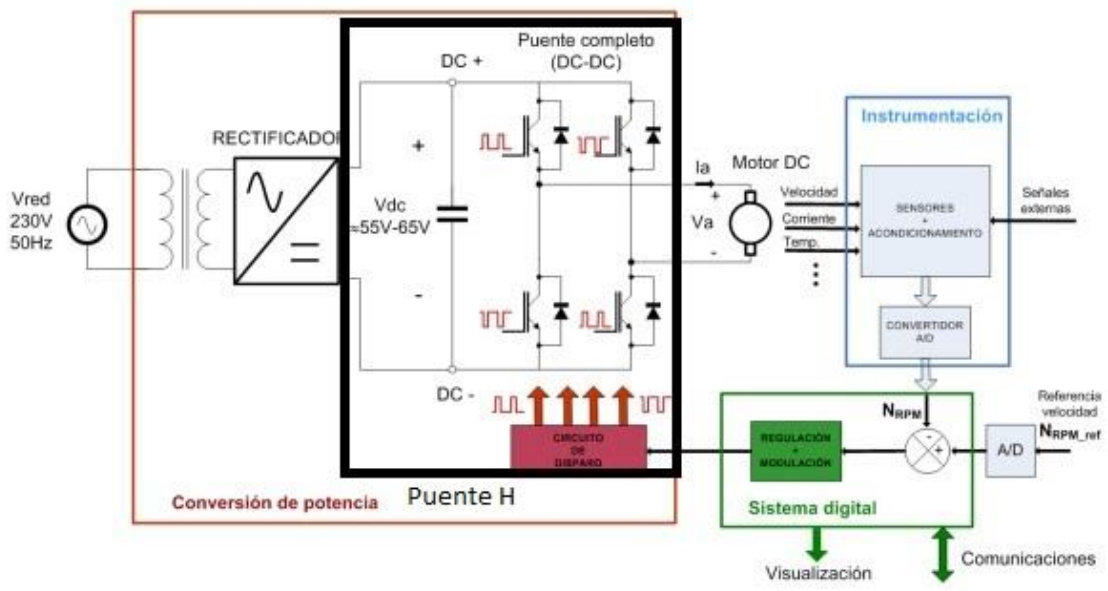

Ilustración 2. Esquema de la etapa de potencia

2.4.1 Puente en H

Como se observa en Ilustración 2, el puente en H es un conjunto de transistores, que en función de una señal de control permiten el paso de la corriente continua en un sentido, el contrario o cortan la corriente.

Cuando están activos ambos transistores de una diagonal, se permite el paso de la corriente en un sentido. La activación de cada una de las diagonales representa un sentido de giro.

La activación de los cuatro transistores simultáneamente debe evitarse, pues se genera un cortocircuito de la entrada y salida que destruiría los mismos.

Cualquier otra combinación de transistores no generaría diferencia de potencial alguna en bornes del motor.

2.4.2 Modulación por ancho de pulso (PWM)

Los transistores del puente en H son controlados por pulsos de tensión de duración variable, es decir, por una señal variable.

Esta señal es generada mediante modulación por ancho de pulso (PWM).

Para generarla, se comparán dos señales (Ilustraciones 3 y 4). Una triangular con una frecuencia y amplitud constante y una señal de referencia continua, proporcional a la velocidad de referencia que deseamos para el motor.

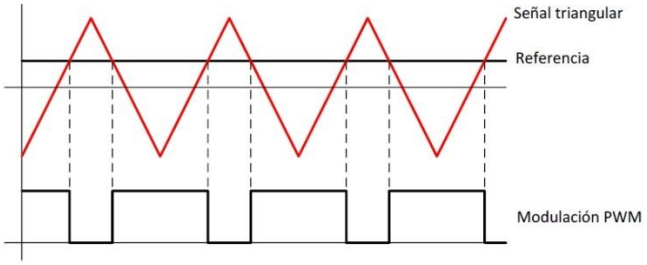

Ilustración 3. Generación de los pulsos

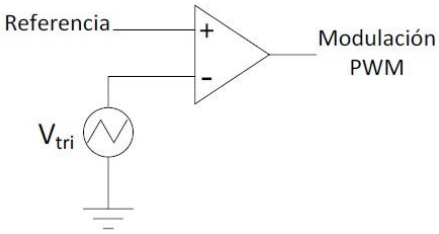

Ilustración 4. Esquema PWM

Como se observa en la ilustración 3, cuando la señal triangular es superior a la de referencia la onda modulada pasa a cero y viceversa.

2.5 Diseño Digital Síncrono

El diseño digital síncrono se basa en la existencia de un reloj (Onda cuadrada de frecuencia constante) que marca con alguno de sus flancos los cambios en las entradas y salidas lógicas.

Se recurre al reloj para coordinar las acciones debido a que los componentes encargados de realizar la lógica tienen un tiempo de retardo en su operación. Esto supone un tiempo transitorio durante el cual la salida de la puerta lógica es errónea.

Por lo tanto, la frecuencia del reloj debe ser lo suficientemente baja como para que todos los tiempos de transición de todas las lógicas se cumplan, de forma que cuando vuelva a aparecer el flanco de activación todas las puertas estén trabajando en régimen permanente.

Un mal diseño síncrono generará señales erróneas que si al acumularse alcanzan el flanco de activación del reloj, se propagarán a lo largo del circuito provocando su mal funcionamiento. Estas señales son comúnmente conocidas como *glitches*.

Para evitarlo existen una serie de normas para el diseño digital síncrono que de seguirse, eliminan en muy alto porcentaje la aparición de *glitches*.

1. Todos los circuitos secuenciales deben realizarse con biestables sensibles al mismo flanco del reloj.
2. Las señales asíncronas del biestable solo pueden estar manejadas por una señal de inicialización (Reset) global y nunca durante la operación normal del sistema.
3. El circuito debe disponer de una señal de reloj única y común para todos los biestables del circuito.
4. Las conexiones entre bloques síncronos deben asegurar a su salida o entrada la sincronización de la señal.

5. Las entradas no síncronas como pulsadores deben ser sincronizadas al entrar al sistema mediante biestables.
6. Está prohibido utilizar latches síncronos o asíncronos.

2.6 FPGA (Field Programmable Gate Array)

Se trata de un dispositivo semiconductor que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada tantas veces como sea necesario, mediante un lenguaje de descripción especializado.

La interconexión de estos bloques lógicos es lo que ha de definir el diseñador, empleando para ello diversos lenguajes como Verilog, VHDL o ABEL. También pueden ser empleados entornos gráficos como programación por bloques, pero finalmente se compilan para regenerarse en los lenguajes de programación escrita anteriores.

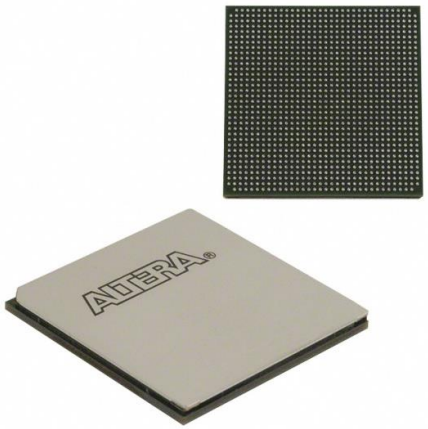

Ilustración 5. FPGA

Dispone, además, de pines digitales tanto de entrada como de salida para gestionar los datos que intercambia con el exterior.

Es, por lo tanto, una plataforma flexible para el diseño y prueba de la lógica digital encargada de controlar un sistema.

La FPGA empleada en nuestro proyecto es una CYCLONE IV E.

Consta de 153 pines digitales de salida y 594Kbits de memoria.

2.7 Tarjeta DE0-NANO

La tarjeta DE0-NANO es una plataforma para la fácil utilización y programación de la FPGA.

Consta de la FPGA y conecta sus salidas a otros puntos de más fácil acceso, así como a convertidores A/D y otros periféricos para mostrar u obtener información. También incluye módulos de memoria extra

Ilustración 6. DE0-NANO

2.7.1 Sistemas de Entrada/Salida

Una FPGA cuenta siempre con pines digitales de entrada y salida para comunicarse con su entorno, la tarjeta se encarga de juntarlos en bloques y buses cómodos de usar.

- Pines digitales: 2 bloques de 40 pines cada uno y un puerto mini-USB para la configuración desde el ordenador.
- Entradas Analógicas: un convertidor A/D de 12 bits de fácil acceso y otro convertidor A/D de 13 bits asociado a un acelerómetro.
- Otros Periféricos: Dos pulsadores a nivel alto en reposo, 8 leds, un acelerómetro.

2.7.2 Memoria

La tarjeta cuenta, además, con los módulos de memoria encargados de almacenar e inicializar los programas en la FPGA. Para ello emplea varios tipos de memoria.

- RAM: Memoria volátil de rápido acceso. Cuenta con 32MB de SD RAM.
- ROM: Memoria más lenta pero no volátil, suele emplearse para inicializar determinados elementos de la FPGA. Posee 2kB de esta memoria implementada mediante tecnología EEPROM.
- Memoria para la configuración interna del equipo de 16MB.

2.8 Encoder

Sensor electromecánico que se encarga, mediante diferentes tecnologías (óptica, mecánica...), de convertir el movimiento rotatorio de un eje en una señal proporcional a su velocidad de giro.

En el encoder utilizado para el proyecto, la velocidad viene representada por dos señales (Ilustración 7). A partir de la frecuencia de cualquiera de ellas obtendremos la relación de velocidades, y gracias al desfase entre las dos obtendremos el sentido de giro.

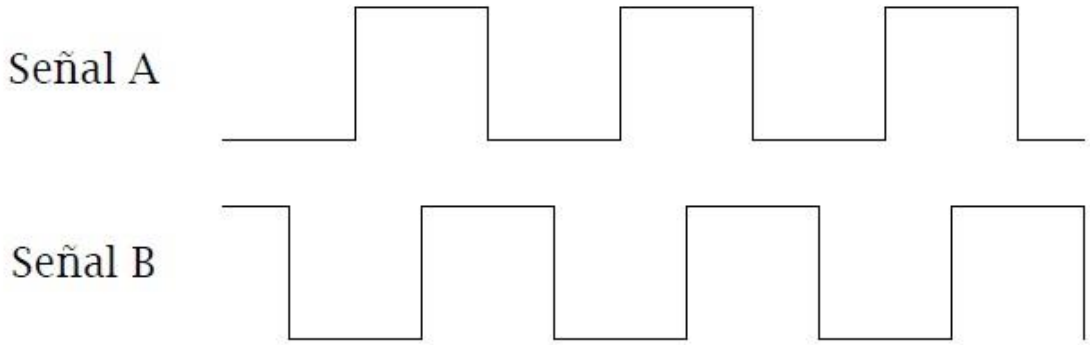

Ilustración 7. Señales del encoder

El sistema de control necesita la velocidad real del eje para ajustarla a la referencia que deseemos.

2.9 Placa de Potencia

Son los componentes encargados de manejar la potencia transmitida al motor de acuerdo con las órdenes del sistema de control, y alimentar algunos periféricos como el encoder.

Consta de:

- Conexión de alimentación a red.
- Rectificador de corriente.
- Puente en H
- Conexión a motor DC
- Alimentación y conexión del encoder

Los primeros cuatro puntos se encargan sucesivamente de recibir la corriente, rectificarla a continua para que el puente en H la controle mediante PWM y enviar la señal controlada al motor.

En el último punto, actúa como entrada de información del sensor.

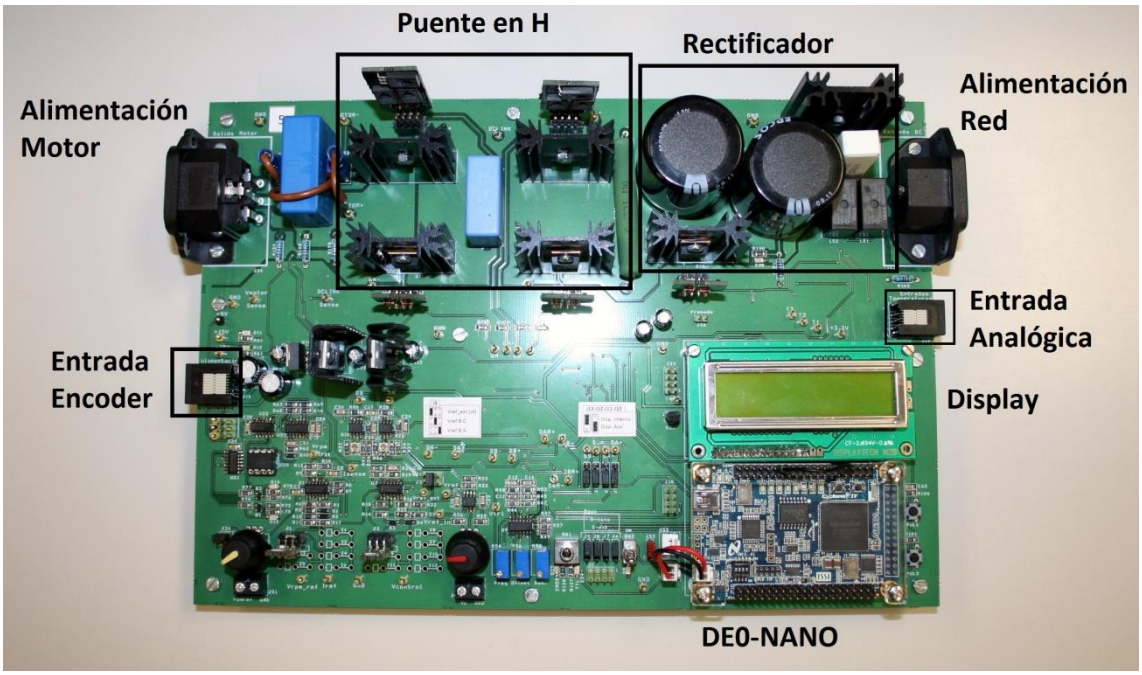

Ilustración 8. Placa de potencia

2.10 Botonera

Consta de 13 pulsadores normalmente a nivel alto repartidos en un cuadrado de 9 pulsadores y otro de 4, y dos potenciómetros con retorno por muelle a modo de acelerador y freno.

Los pulsadores cuentan además con condensadores antirebote para evitar enviar pulsos dobles y se pude observar también un puerto de salida analógica para los potenciómetros.

La placa se conecta a la tarjeta DE0-NANO mediante un bus de pines digitales para los pulsadores y mediante cable a la entrada analógica.

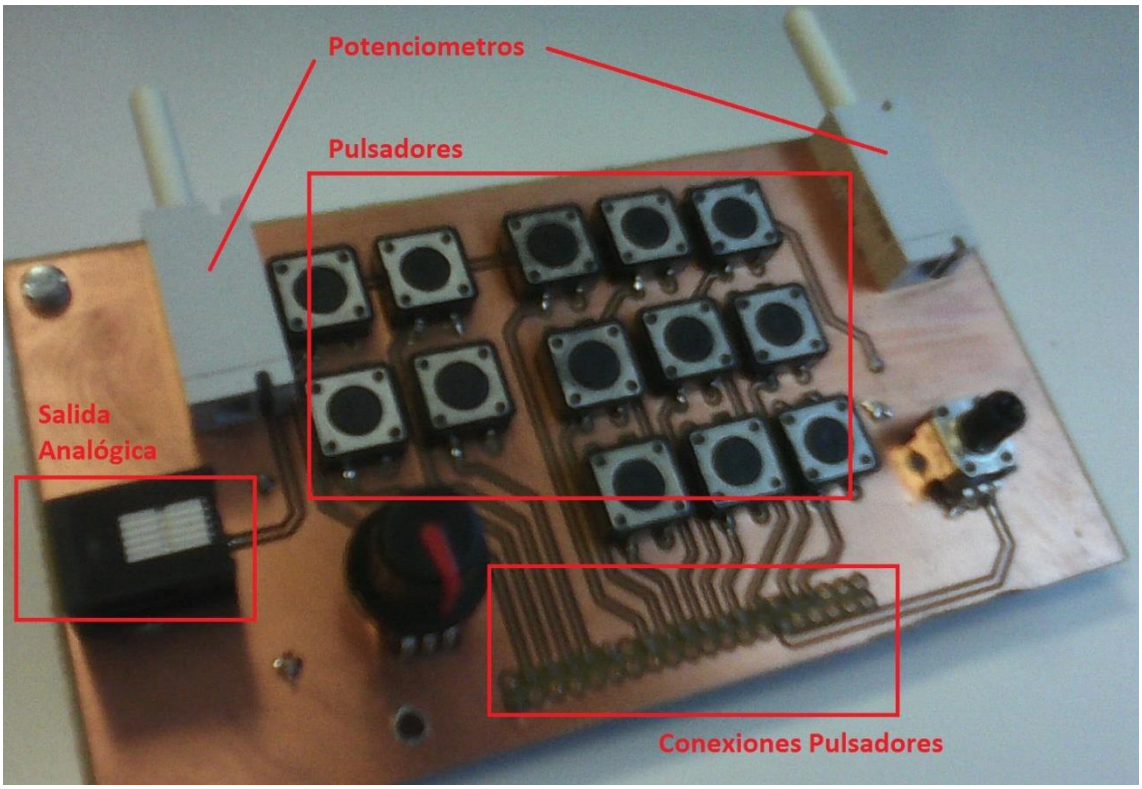

Ilustración 9. Botonera

2.11 LCD

Es la pantalla junto con algunas memorias que nos permiten mostrar datos escritos desde la FPGA hacia el usuario.

En nuestro caso, contamos con una pantalla HD44780 de HITACHI. Esta, puede presentar 2 líneas de 16 caracteres cada una (Ilustración 10).

Ilustración 10. Muestra LCD

En cuanto a memorias, este dispositivo presenta una primera memoria ROM donde se almacenan todos los posibles caracteres que puede mostrar la pantalla (tanto letras como símbolos y números).

Dispone también de dos RAM. Una primera CGRAM (Character Generator RAM) capaz de generar caracteres distintos a los almacenados en la ROM con códigos de 8 bits.

Y una DDRAM (Data Display RAM) encargada de almacenar los datos que se están mostrando en ese momento a través de la pantalla.

3 Diseño del software

Todo el software desarrollado para este proyecto ha sido realizado en Quartus II mediante diagramas de bloques y maquinas de estados quedando el nivel TOP del diseño de la siguiente manera.

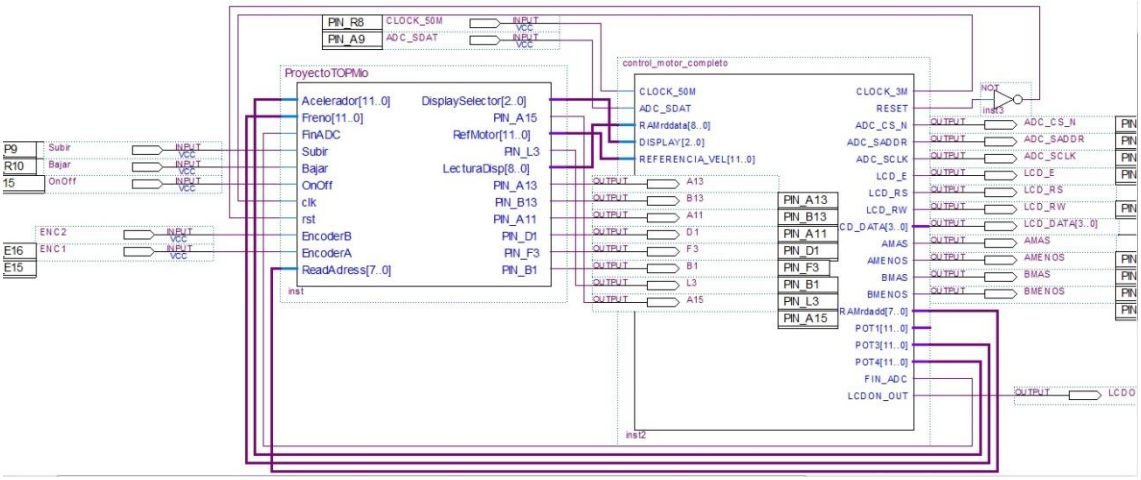

Ilustración 11. Vista general del nivel más alto del programa

En la Ilustración 11 se observan dos bloques. El bloque de la derecha de la Ilustración 11 se muestra con más detalle en la Ilustración 12. Este bloque se encarga de gestionar varias cosas:

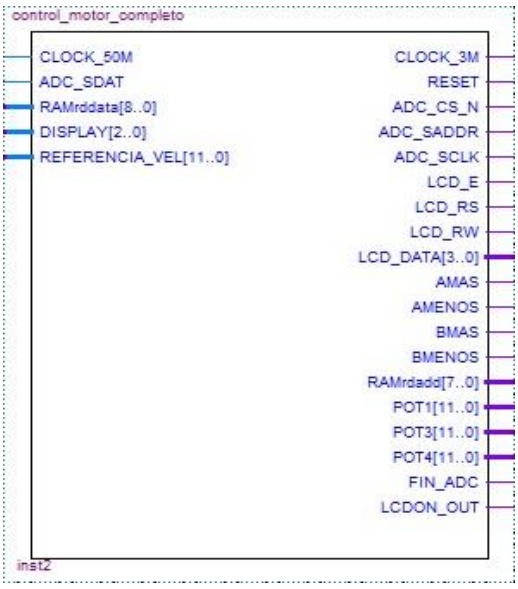

Ilustración 12. Control motor completo

- Ajusta el reloj de 50MHz (CLOCK_50M) a 3MHz (CLOCK_3M) que es nuestra frecuencia de trabajo.

- Controla la lectura por parte del display de los datos (RAMrddata, DISPLAY, LCD_E, LCD_RS, LCD_DATA, RAMrdadd).
- Conecta y controla los convertidores A/D (ADC_SDAT, ADC_CS_N, ADC_SADDR, ADC_SCLK, FIN_ADC, POT1, POT2, POT3).
- Conecta las referencias con el sistema de control (PWM + Puente en H) (REFERENCIA_VEL, AMAS, AMENOS, BMAS, BMENOS)

Estos sistemas exceden el alcance del proyecto que se limita a realizar el software de control de la velocidad del motor, y por lo tanto este bloque había sido diseñado previamente.

El bloque de la izquierda de la Ilustración 11 se muestra con más detalle en la Ilustración 13. Este bloque engloba todo el diseño realizado y contiene toda la lógica para el funcionamiento del sistema.

Además, como se observa en las anotaciones que aparecen junto a las entradas y salidas (Ilustración 11), se ha realizado la asignación de pines para estas, de forma que corresponden cada una a un pin de la FPGA que la tarjeta corresponderá a un reloj, un botón, un led o lo que corresponda en cada caso.

3.1 Bloque Nivel Alto

Este bloque de nivel superior (Ilustración 13) está dividido en 4 bloques principales que se encargan cada uno de una de las funciones que debe realizar mi sistema.

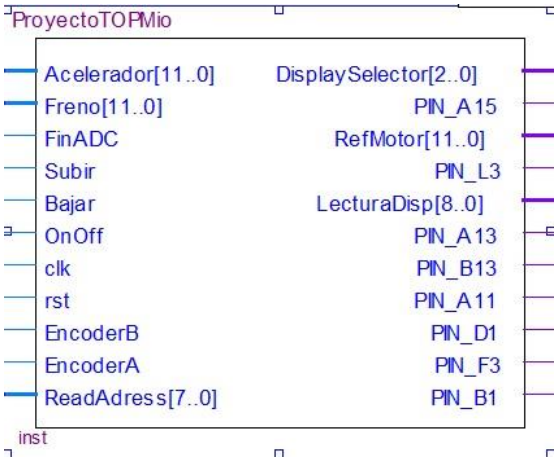

Ilustración 13. Bloque TOP

Como entradas para todo el control empleo:

- Acelerador y Freno: señales convertidas por el ADC encargadas de marcar la referencia de velocidad al sistema de control.
- Fin ADC: Se activa cuando el convertidor ha realizado satisfactoriamente la conversión e indica cuando son validos sus datos.

- Subir, Bajar, OnOff: señales de los botones.
- Encoder A y B: señales del encoder para la velocidad.
- ReadAdress: Dirección de lectura gestionada por el bloque de la derecha.
- Rst y clk: Señales de reset a nivel alto y reloj a la frecuencia de trabajo 3MHz.

Como salidas tenemos:

- DisplaySelector: Señal que da al display la información de que plantilla de pantalla ha de usar.
- RefMotor: Señal proporcional a la velocidad deseada y que enviamos como referencia al PWM.
- LecturaDisp: Datos que debe mostrar el display y que son la parte variable de las plantillas o pantallas predefinidas creadas.
- PIN_XX: Salidas a los leds para uso como señalizadores.

Una vista interna de este bloque puede verse en la Ilustración 14.

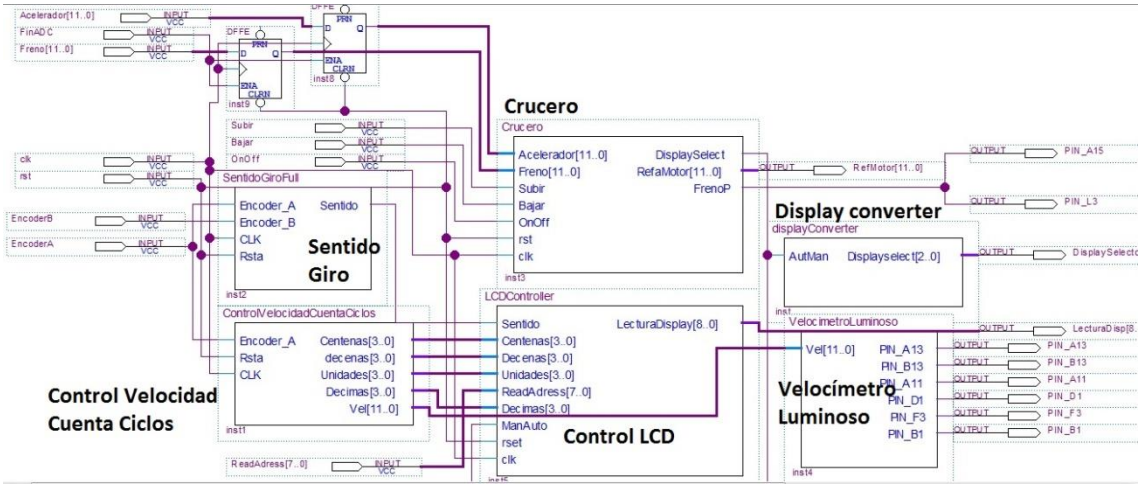

Ilustración 14. Bloque nivel alto

3.2 Control Velocidad Cuenta Ciclos

El control de velocidad cuenta ciclos es el bloque encargado de gestionar una de las señales del encoder (Ilustración 7) y proporciona un valor de la velocidad de giro real del motor, que posteriormente se empleará en el velocímetro del display.

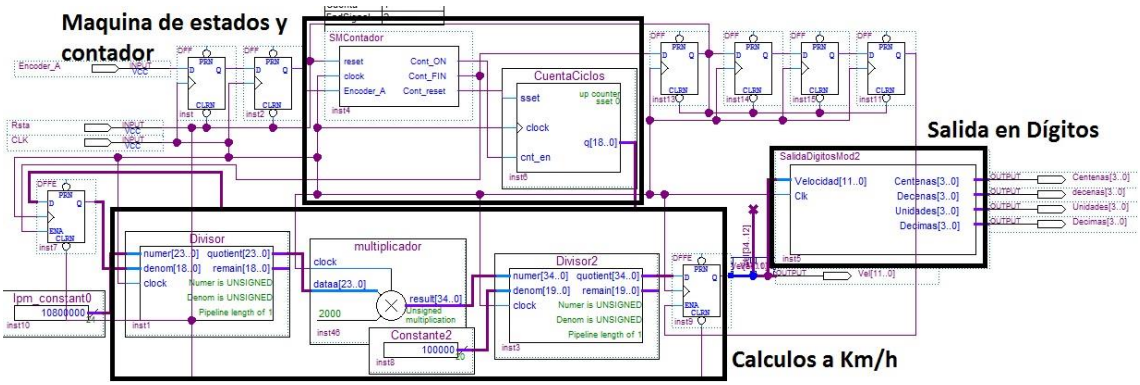

Ilustración 15. Control velocidad cuenta ciclos

Como se observa en la ilustración 15, el sistema ha sido dividido en dos grandes bloques. La máquina de estados y contador de ciclos (Ilustración 16) y el bloque de cálculos (Ilustración 17).

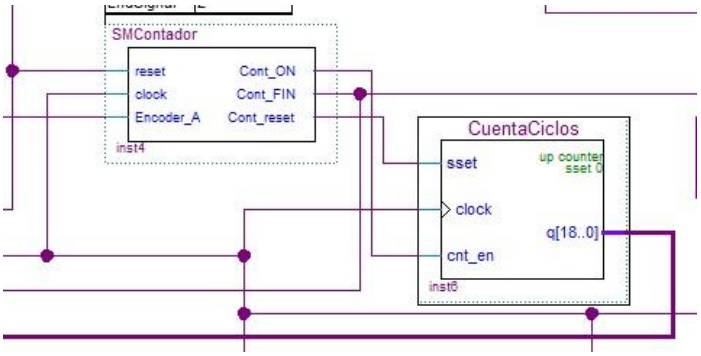

Ilustración 16. Maquina de estados y contador

La máquina de estados (SMContador) activa el contador (CuentaCiclos) cuando detecta que la señal del encoder ha pasado a uno.

Tras esto, espera a que la señal vuelva a cero desactivando y reseteando el contador a la vez que guarda el valor final de este para su procesamiento.

Ese valor es el número de ciclos de reloj que ha pasado el encoder a uno, es decir, el número de ciclos en medio periodo de la señal del encoder.

Para obtener la velocidad, se parte de que el encoder esta calibrado para dar a 4000rpm una señal de frecuencia 33.33KHz, conociendo también la frecuencia del reloj que es de 3MHz se puede obtener el numero de ciclos que contará a 4000rpm. (Los periodos T son la inversa de la frecuencia y el periodo de la señal del encoder A es el doble del que medimos)

$$\frac{Ta}{Tclk} = \frac{3 * 10^3}{33.3 * 2} = 45\ cuentas$$

Y por lo tanto, la ecuación quedará:

$$\frac{4000rpm}{Num.Cuentas} * 45cuentas * \frac{mm}{vuelta\ de\ rueda} * \frac{60min/h}{100000mm/(Km * 10)} = \frac{Km}{h} * 10$$

Asumiendo una rueda con un radio de 30cm obtenemos un perímetro de 2 metros:

$$\frac{10800000}{Num.Cuentas} * \frac{2000mm}{vuelta} * \frac{1}{100000} = \frac{Km}{h} * 10$$

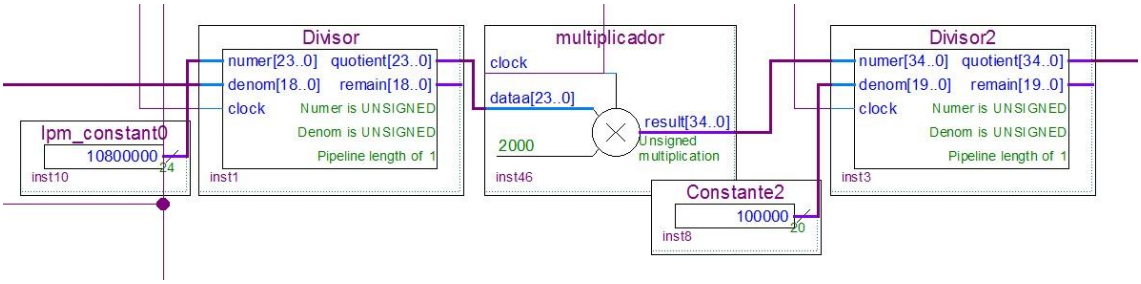

Ilustración 17.Implementación de los cálculos necesarios para obtener la velocidad del coche en Km/h

El resultado se queda en Km/h*10 debido a que así se dispondrá del primer decimal de la velocidad para mostrar por el display.

El número de cuentas variará entre las 45 cuentas que se recogerán a 4000rpm y las 360000 cuentas a 0.5 rpm. Para contar hasta un valor tan alto, necesitamos como mínimo 19 bits.

Los dos primeros biestables que se observan en la parte superior izquierda de la Ilustración 15 junto a la máquina de estados, se encargan de sincronizar la señal del encoder con el reloj interno y facilitar así su lectura.

El biestable a la izquierda de la parte de cálculos (Ilustración 15), evita que el contador envié datos a las operaciones antes de que la cuenta este completa y por último los cuatro biestables a la derecha del contador retrasan la señal encargada de activar el biestable de salida para que solo almacene los datos correctos.

El último bloque (Salida en dígitos, en la Ilustración 15), se encarga de separar cada uno de los dígitos de la velocidad para en última instancia enviárselos al display.

El proceso de diseño ha tenido como puntos críticos la correcta implementación de la máquina de estados y la separación de la velocidad en dígitos, así como el evitar con los biestables dotados de entrada ENABLE el paso de datos falsos. Este último aspecto ya

ha sido explicado anteriormente por lo que solo falta por explicar con detalle los dos primeros puntos críticos.

3.2.1 SMContador (Maquina de estados)

La máquina de estados (SMContador, Ilustración 18), activa el modo cuenta mientras el encoder está a uno y al pasar a cero de nuevo envía la orden de guardar dato (END Signal 1 y 2, Ilustración 18) y reseteando la cuenta en el estado de reposo ParadaReset.

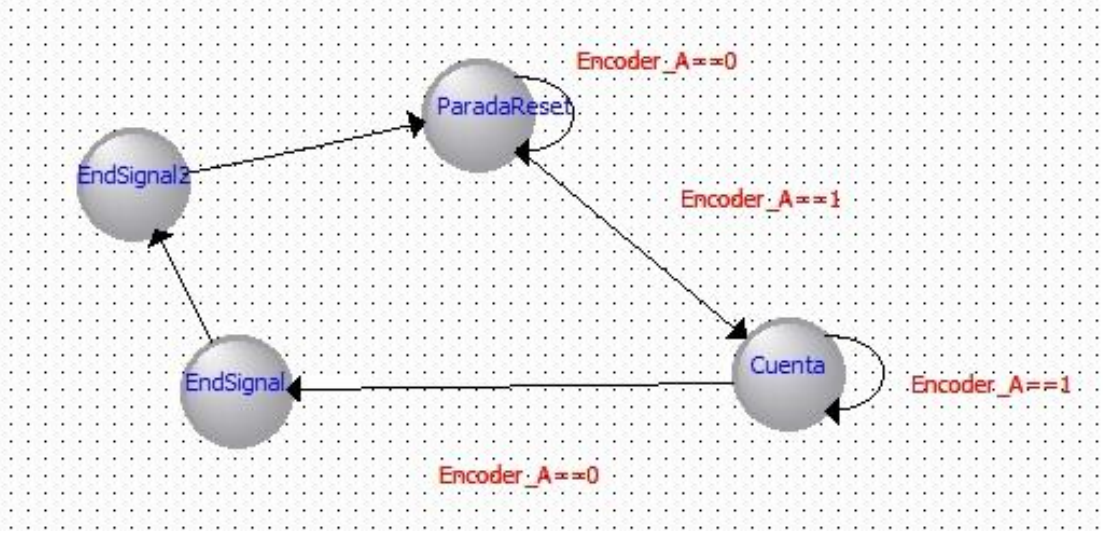

Ilustración 18. SM contador

Los dos estados ENDSignal y ENDSignal2 dan orden al biestable de guardar la cuenta. Se han implementado dos estados (ENDSignal y ENDSignal2) para evitar que cualquier fallo de sincronización (un cálculo que no haya finalizado a tiempo) de cómo resultado una pérdida del valor de la cuenta o un dato falso del mismo.

3.2.2 Conversión al sistema decimal (Salida Dígitos)

La conversión de datos al sistema decimal se realiza a través del bloque Salida Dígitos (Ilustración 19).

A partir del valor de velocidad en 12 bits en Km/h * 10 debemos obtener los dígitos que componen la velocidad en Km/h.

Esto se realiza mediante la división sucesiva entre 10. Al tratarse de un número de 4 dígitos emplearemos 3 divisiones.

De las dos primeras obtendremos los residuos como Decimas y Unidades y de la última división el resto corresponderá a las Decenas y el resultado a las Centenas.

A continuación se, se realiza el proceso para el número 3527

$$\frac{3527}{10} = 352\ y\ Residuo = 7(Decimas)$$

$$\frac{352}{10} = 35\ y\ Residuo = 2(Unidades)$$

$$\frac{35}{10} = 3(Centenas)\ y\ Residuo = 5(Decenas)$$

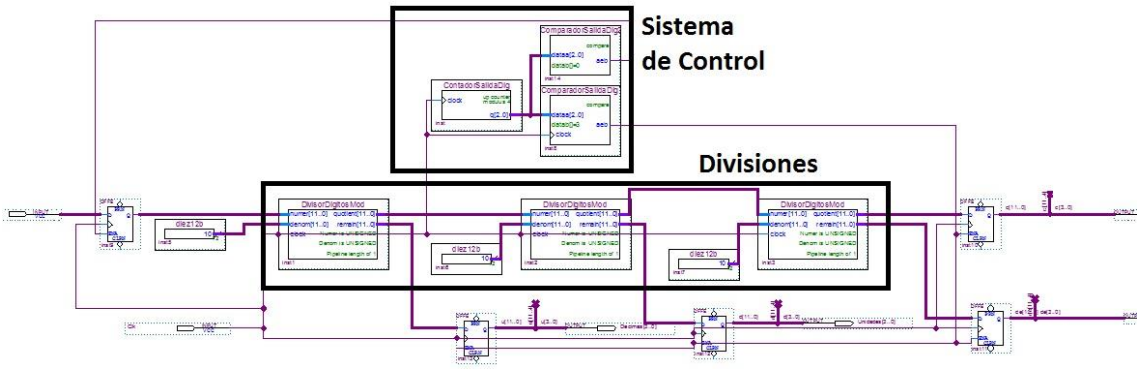

Ilustración 19. Conversor al sistema decimal (Salida dígitos)

Como se puede observar en las ilustraciones 19 y 20, existe un sistema de control que solo permite a los biestables almacenar el dato cuando todos los dígitos estén listos.

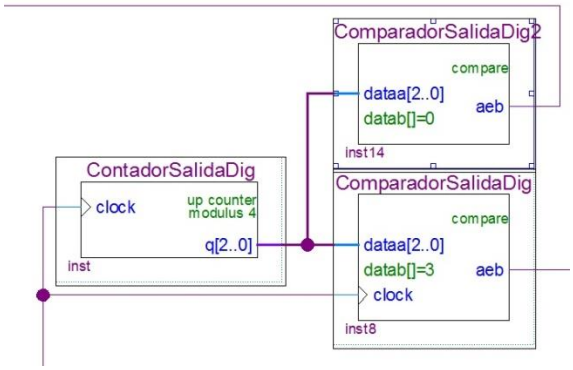

Ilustración 20. Sistema de control

Esta función se implementa mediante un contador unido a dos comparadores (Ilustración 20).

Al principio de la conversión el contador está a cero, y uno de los comparadores envía un pulso al biestable de entrada (derecha Ilustración 19) para recibir un valor. Tras 3 pulsos, este valor ya ha sido tratado por los divisores y están listos los dígitos, de forma que el comparador activa los biestables de cada una de las salidas (Ilustración 21) para almacenar los dígitos.

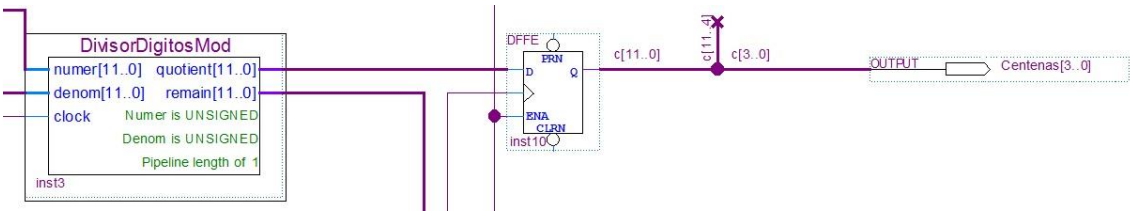

Ilustración 21. Detalle de la salida del dígito

Por último, como los dígitos son valores entre 0 y 9 y pueden codificarse con 4 bits, se han desechado los 8 bits de mayor peso que no transmiten información (A la izquierda del biestable, Ilustración 21).

3.3 Sentido de Giro

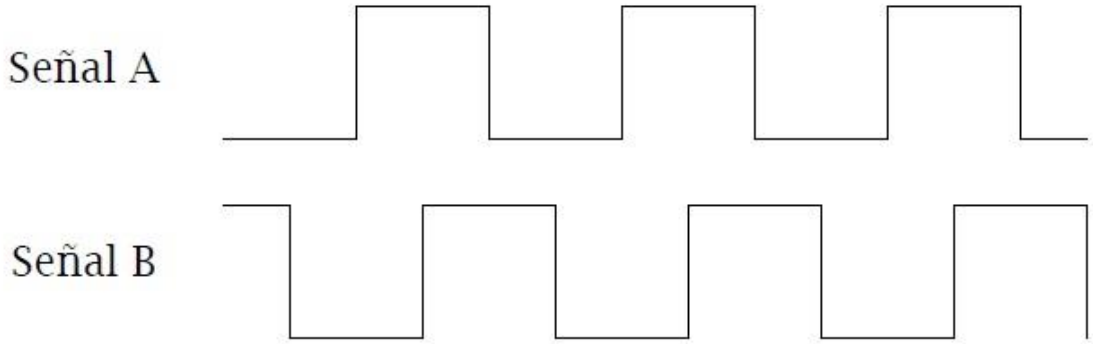

Ilustración 22. Señales del encoder

Este bloque permite saber el sentido de giro del motor gracias a los desfases entre las señales del encoder A y B. (Ilustración 22).

Así pues, una de las dos señales pasa a 1 antes que la otra y en función de cuál sea nos determinará el sentido de giro. Cuando es la señal A la primera en pasar a 1 tenemos giro a derechas, y cuando es la B la primera en pasar a 1 tenemos giro a izquierda.

Para su implementación, se ha empleado una máquina de estados (Ilustración 23) y para su correcto funcionamiento, se han sincronizado las dos señales del encoder con dos biestables cada una.

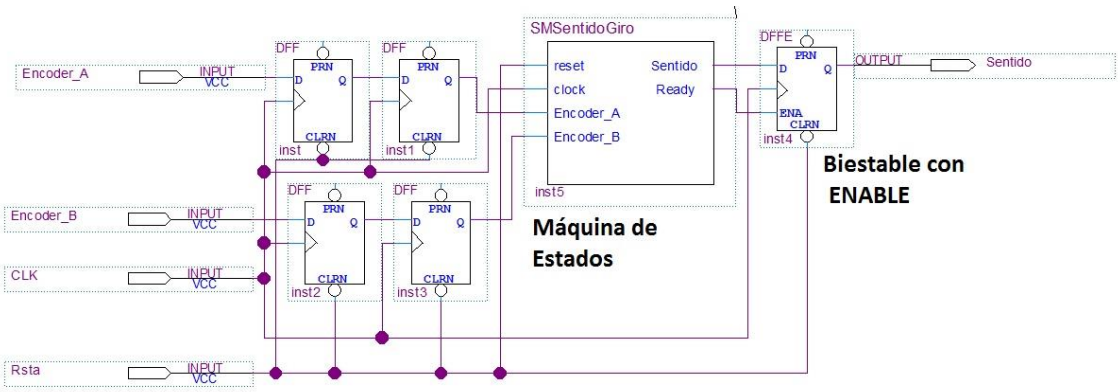

Ilustración 23. Bloque sentido de giro

La máquina de estados (Ilustración 24), presenta dos caminos en función de que señal del encoder (A o B) se active primero, pero no dará la dirección definitiva hasta que la segunda señal pase a 1(Estados Izquierda2 o Derecha 2, Ilustración 24) ya que podría darse el caso de detectar una señal a uno cuando la otra señal acaba de pasar a cero, dando una dirección contraria a la que realmente lleva.

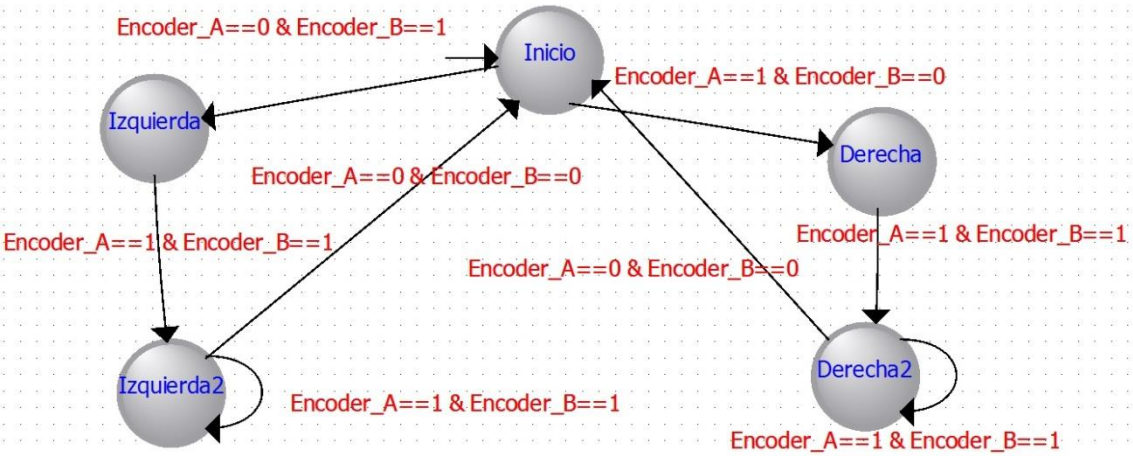

Ilustración 24. SMSentidoGiro (Máquina de estados)

El bloque SMSentidoGiro generará como salida la señal sentido con un 1 cuando se detecte giro a derecha y un 0 cuando el giro sea a izquierda, pero este no será enviado al sistema hasta que se haya producido la confirmación que habilitará el último biestable para almacenar el sentido.

3.4 Control de Crucero

El bloque llamado Control de Crucero (Ilustración 25), es el encargado de gestionar las demandas del usuario en cuanto a velocidad y modo de funcionamiento, por su complejidad será explicado por partes: Bouncers, Simulador de inercia y Maquina de estados.

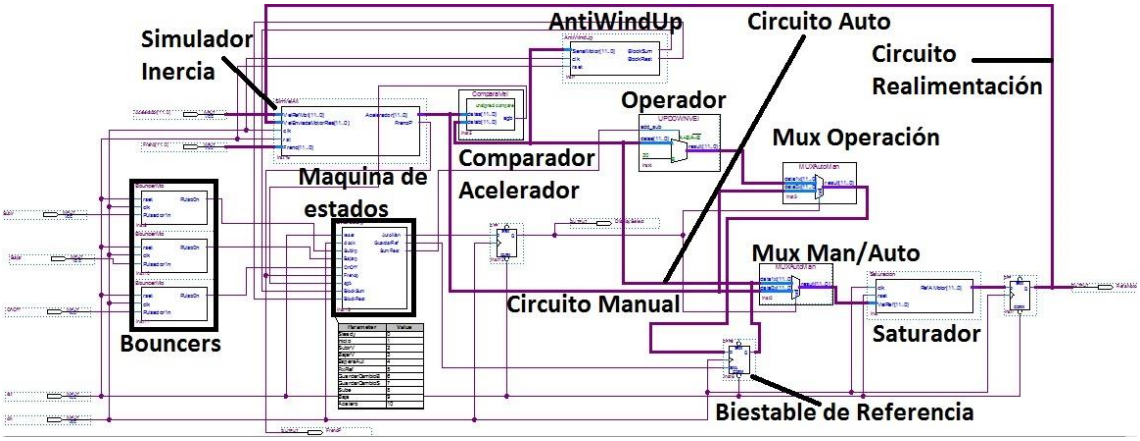

Ilustración 25. Bloque control de crucero

Este bloque recibe las señales de acelerador y freno provenientes de los convertidores AD y las señales de los 3 pulsadores utilizados directamente de la botonera y envía la señal definitiva al motor para fijar su velocidad.

3.4.1 Bouncer

El funcionamiento de un pulsador no genera la señal perfecta y nítida de un pulso, sino que se obtiene un tren de pulsos. Además, puede generar corrientes inducidas en otros pulsadores que producen falsos pulsos.

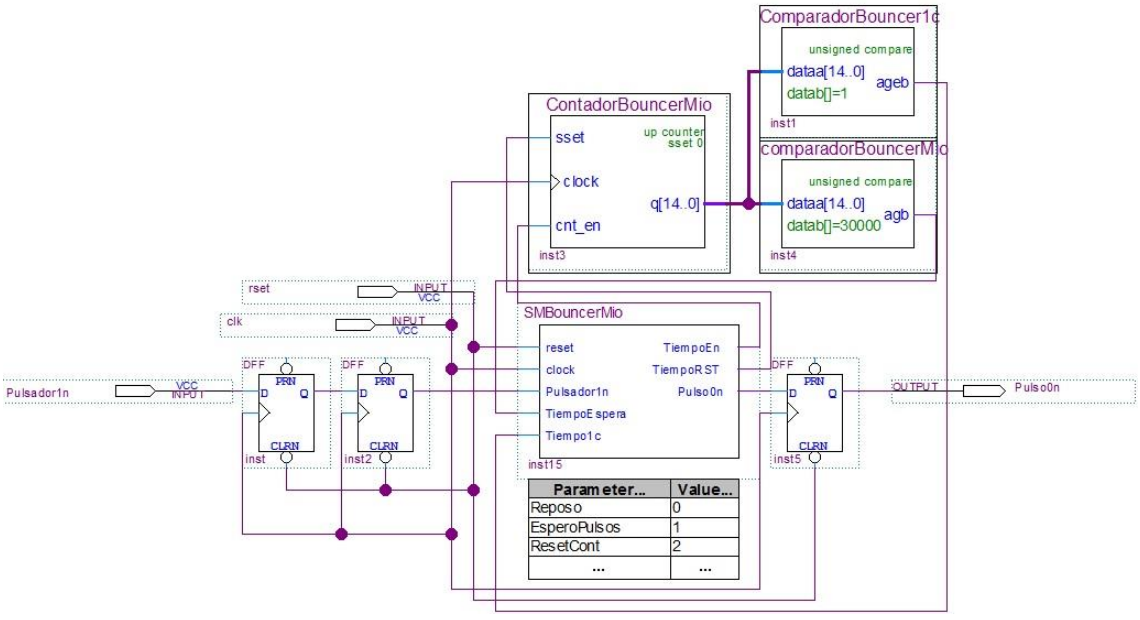

Ilustración 26.Maquina de estados, contador y comparadores

Para evitar todo esto se ha diseñado el bouncer (Ilustración 25). Su funcionamiento es sencillo, cuando detecta que el pulsador ha sido activado (Pulsador1n en SMBouncerMio, Ilustración 26) espera un muy pequeño tiempo (10ms) para asegurarse que no es un pulso inducido por otro pulsador y a su vez también deja pasar el tren de pulsos generado. Si tras ese tiempo no sigue activado el pulsador, la máquina de estados asume que es un único pulso inducido y vuelve al estado inicial sin enviar el pulso al sistema. Si ese pulso se mantiene, se envía un pulso de 1 ciclo de duración de manera que sea posible trabajar con él (Pulso0n en SMBouncerMio, Ilustración 26). El resto del tiempo hasta volver a estar preparado para recibir un pulso lo pasa reseteando el contador de ciclo (TiempoRST en SMBouncerMio, Ilustración 26).

Todo el control lo realiza la máquina de estados (SMBouncerMio, Ilustración 26) que se encarga de ir pasando las etapas del proceso, las cuales avanzan tras las señales de contadores (ContadorBouncerMio, Ilustración 26), comparadores (ComparadorBouncerMio y ComparadorBouncer1c, Ilustración 26) y pulsadores.

Destacar que debido a la lógica de los pulsadores (a nivel alto en reposo) y a que la lógica del diseño se ha basado en pulsos de nivel alto (lógica inversa) la señal del bouncer ya corrige esto entendiendo que un pulso del pulsador exterior es un nivel bajo y al emitir el pulso a mi sistema lo hace a nivel alto.

3.4.2 Simulador Inercia Coche

En este bloque se simula el comportamiento del coche ante una aceleración y una frenada simulando la inercia del coche a mantener su velocidad.

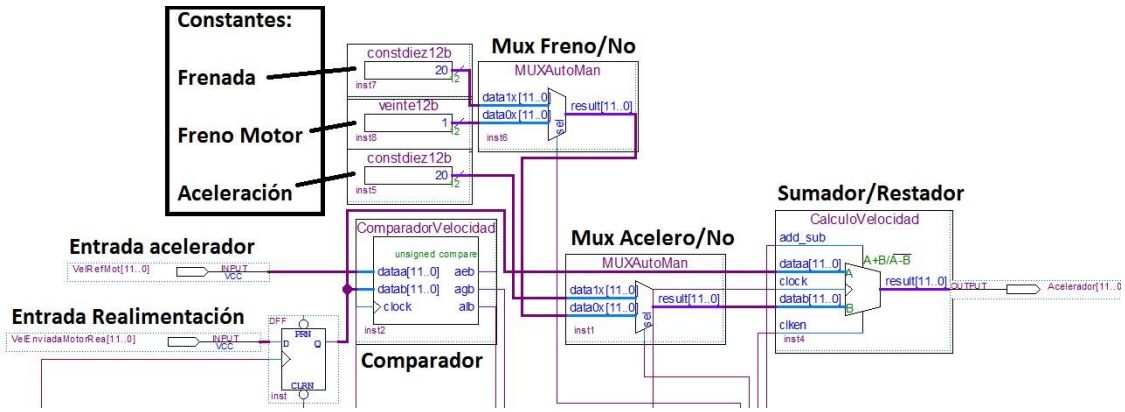

Ilustración 27. Manejo señal principal

Se conectan a este bloque el acelerador y el freno, así como la realimentación con la velocidad que todo el bloque Control de Crucero genera para enviar al motor

(Ilustración 27). Obteniéndose a la salida una señal de freno pulsado[1] y una señal de aceleración que varía con aceleración constante.

Para esto último, el acelerador no controla directamente la velocidad sino que compara esta con la que actualmente se está enviando al motor (realimentación). De esta manera cuando detecta un incremento o decremento de la velocidad de referencia (del acelerador) va subiendo o bajando progresivamente desde la velocidad enviada al motor en el ciclo anterior gracias al sumador/restador (Ilustración 27) que activado cada 10ms genera un gradiente de velocidades.

El funcionamiento del sistema pasa por las siguientes fases. En primer lugar el comparador situado en la entrada (Ilustración 27) compara las señales realimentada y de referencia (Acelerador), y genera una señal en caso de que la realimentada sea mayor que la de referencia, u otra señal desde otra de las salidas salida del comparador en caso de que sea menor. Mientras, el *ClockEnable* del sumador-restador impide cualquier operación hasta que el comparador haya emitido alguna señal de diferencia entre velocidades, momento en el cual, realizar la operaciones pertinentes (Sumar si la referencia es mayor que la realimentación o restar en caso contrario) pero limitándolas a 1 cada 10ms.

En función de que el sistema este acelerando, neutro (ni acelerador ni freno) o frenando este gradiente de velocidades es distinto gracias a un sistema de multiplexores que conectan al sumador la constante que va modificar la velocidad (Ilustración 27).

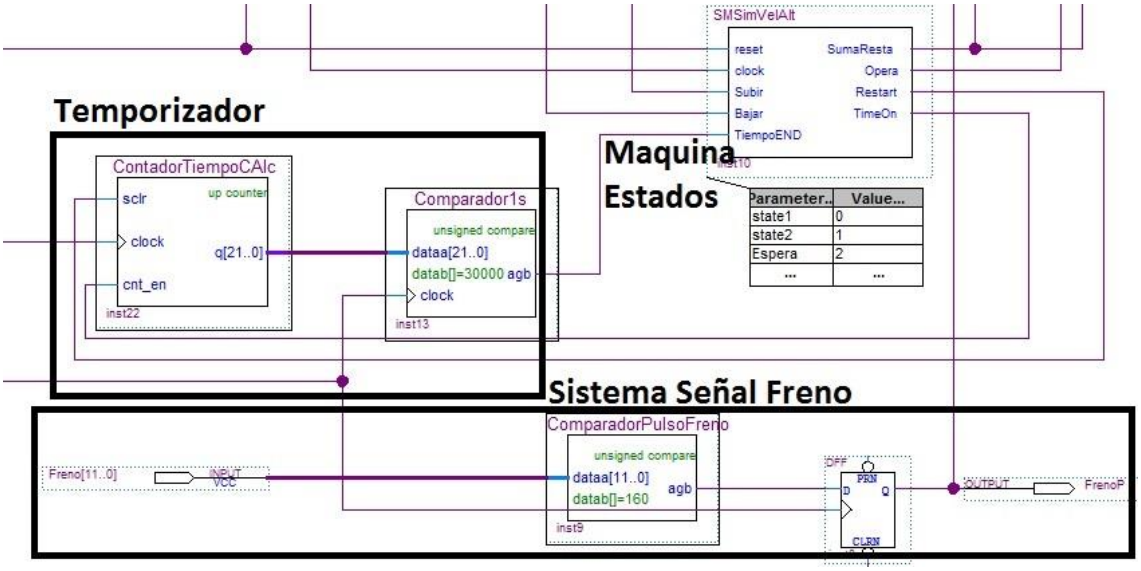

Ilustración 28. Control de multiplexores y señal de freno

Todo el sistema está controlado por una máquina de estados (Ilustración 28) que recibe las señales de diferencia de velocidades (Subir o Bajar) y tiempos cumplidos (TiempoEND), para generar, tras cumplirse las condiciones, las señales de operación (habilitar la operación y suma o resta).

[1] Señal generada a partir de un comparador en el freno que detecta cuando este está pulsado. (Se ha diseñado con un umbral de 160).

El hecho de que una de las señales comparadas sea la enviada al motor no es casual, pues permite mantener activo el sistema de inercia mientras esta en modo automático y por lo tanto, una salida brusca del modo automático no dará como resultado un frenazo brusco, sino una bajada progresiva de velocidad, al igual que sucedería si se levantase el pie del acelerador.

3.4.3 Maquina de estados del control de crucero y elementos controlados

La máquina de estados del control de crucero es la encargada de definir el modo de funcionamiento entre manual y automático.

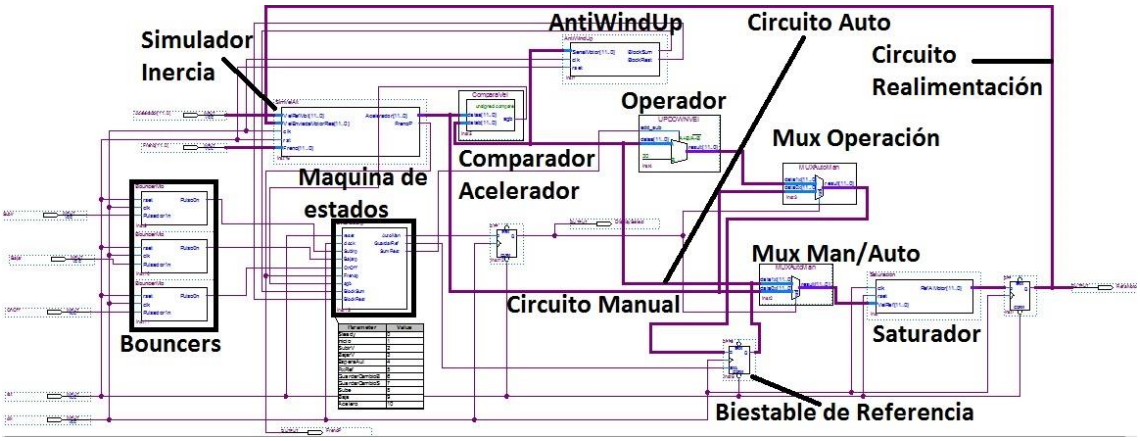

Ilustración 29. Bloque control crucero

Se ha decidido implementar una única máquina de estados y no varias para los diferentes elementos para evitar los problemas de coordinación que se daban en las primeras pruebas realizadas con varias máquinas.

El modo manual se ha diseñado conectando vía multiplexor (Ilustración 29, Mux Man/Auto) el acelerador (corresponde a la salida del bloque de simulación) a la salida que manda la referencia al motor. Este modo es el predefinido en la máquina de estados.

El modo automático se decidió que como en algunos modelos de coche se iniciase con la pulsación consecutiva del botón ON y del botón Bajar Velocidad, de forma que la máquina de estados pase a los estados de modo automático (Ilustración 30). Para poner en funcionamiento el modo automático, se envía la señal al multiplexor (Ilustración 29, Mux Man/Auto) que desconecta la unión directa del acelerador con la referencia a motor y conecta en su lugar el biestable de referencia (Ilustración 29). Este biestable mantiene almacenada la última velocidad pasando a ser esta la referencia.

Se puede apreciar que la parte automática se gestiona en un circuito paralelo al manual (Ilustración 29, Circuito Manual/Auto) El Circuito Manual y el Circuito Auto solo se conectan a través del biestable de referencia en el momento de adquirir este la velocidad de referencia en el paso de manual a automático.

A partir de aquí se presenta un abanico de posibilidades a las que la máquina de estados (Ilustración 30) ha de gestionar.

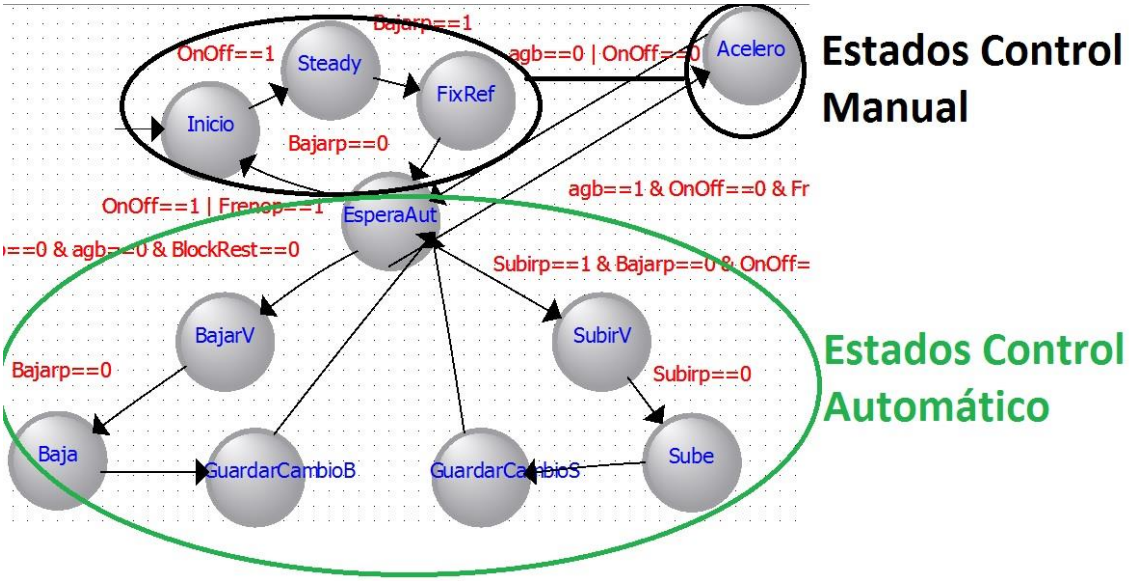

Ilustración 30.SMCruiserp maquina de estados

Partiremos del estado Inicio (Ilustración 30) que corresponde con un modo de conducción manual a todos los efectos.

Para activar el modo automático deberemos pulsar consecutivamente los botones OnOff y Bajar velocidad de forma que la máquina de estados pase por el estado Steady (Idéntico al estado Inicio) y llegue al estado FixRef (Ilustración 30).

El estado FixRef genera la señal Enable al biestable de referencia (Ilustración 29) guardando la velocidad que en ese momento lleva el vehículo. Este estado está activo solo hasta que detecta que la señal del pulsador Bajar vuelve a cero. (Recordar que las señales de los pulsadores provienen de los Bouncer (Punto 3.4.1) y que aunque se mantenga el botón pulsado al sistema solo le llega un único pulso de un ciclo de duración).

Pasamos entonces a los estados de modo automático. Estos se caracterizan por enviar, en todos los estados automáticos, una señal al multiplexor Mux Man/Auto (Ilustración 29) para desconectar la entrada directa del acelerador (Proveniente del bloque Simulador de Inercia, Ilustración 29) y conectar la entrada del biestable que guarda la referencia de velocidad guardada en el estado (FixRef, Ilustración 30).

A continuación describiremos con detalle todos los estados automáticos.

El estado automático por defecto es el Espera Auto, como se puede observar en la Ilustración 30, este bloque presenta numerosas transiciones ya que es a partir de este estado desde donde se pueden llevar a cabo todas las acciones del modo automático.

El estado Espera Auto permite desconectar el sistema automático y volver al estado manual de Inicio (Ilustración 30) mediante la pulsación del freno o del pulsador OnOff.

El sistema también pasará a modo manual en caso de que el comparador del acelerador (Ilustración 29) mande la señal de que estamos acelerando por encima de la referencia, pasando al estado Acelerando de la Ilustración 30. Cuando cese dicha señal volveremos al modo Espera Auto.

Desde el estado Espera Auto también se puede acceder a los otros dos estados del modo automático, SubirV y BajarV (Ilustración 30). Ambos estados funcionan de forma muy parecida y por eso solo se explicará uno de ellos, con las explicaciones oportunas de las pocas diferencias que hay.

El estado BajarV (*SubirV*) se activará cuando estando en modo automático se pulse el botón Bajar (*Subir*). Este estado mandará la señal de resta (*suma*) al Operador (Ilustración 29) y pasará al siguiente estado Baja (*Sube*). En este estado además de mantenerse la señal de resta (*suma*) al operador, se enviará al Mux Operación (Ilustración 29) la señal de selección que conectará la salida del multiplexor con la salida del operador. Tras este estado se llega al estado GuardarCambioB (*GuardarCambioS*) (Ilustración 30) que habilita el biestable de referencia (Ilustración 29) para guardar la nueva referencia que será 20 unidades menores (mayores) que la inicial.

Tras este proceso se vuelve al estado Espera Auto.

La señal enviada al motor ya sea la enviada por el acelerador del bloque Simulador Inercia (Ilustración 29) o la registrada en el biestable de referencia ha de pasar por un último bloque.

Este último bloque llamado Saturador (Ilustración 29) limita por precaución la señal de salida para no forzar el motor, manteniendo fija la salida en caso de que esta sea superior o inferior a unas referencias previas. La referencia inferior y la superior han sido fijadas a 30 y 2800 respectivamente, pero pueden ser ajustadas para cada motor en particular.

3.5 Control LCD (Control de los datos enviados al display)

El Control LCD es el bloque que reúne todos los elementos necesarios para el uso del display, desde el bloque de memoria RAM donde se almacenan los datos a mostrar hasta los diferentes convertidores que transforman los números y señales del sistema en códigos que permiten su representación en el display.

Este módulo consta de varios módulos menores por lo que se explicarán individualmente partiendo del nivel más alto.

3.5.1 Control LCD. Nivel superior

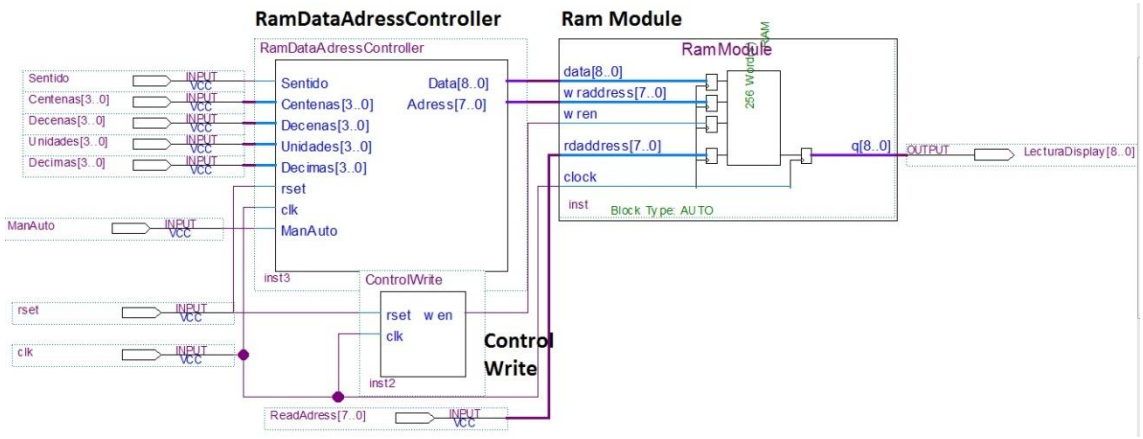

Ilustración 31. Control LCD

Este nivel contiene 3 módulos, como se puede observar en la Ilustración 31. El primero de ellos, es el *Ram Data Adress Controller* encargado de gestionar los datos y explicado con más detalle en el capítulo siguiente.

El segundo módulo es el *RAM Module* que consta de la memoria RAM que almacena las plantillas de pantallas y es el lugar donde han de refrescarse periódicamente las variables del sistema.

Esta RAM tiene las siguientes características:

- Tiene dos puertos de datos, uno de lectura y otro de escritura que evitan problemas en caso de que estas dos acciones se lleven a cabo simultáneamente. (lee el dato que será sobrescrito)
- Tiene capacidad para 256 palabras de 9 bits cada una.
- Tiene un puerto *ENABLE* para la escritura

La RAM esta inicializada por el archivo INIT_RAM_LCD_PANTALLAS.mif donde aparecen guardadas las plantillas que utiliza la pantalla. En este caso estas plantillas son 2, y están controladas por la señal Manual/Automático del control de crucero anteriormente descrito. Esta señal se recodifica a 3 bits para adaptarla al controlador de lectura del bloque Control Motor Completo del nivel superior de todo el programa.

Las plantillas de pantallas almacenadas en la RAM han sido diseñadas para mostrar el modo en el que está trabajando el sistema (Manual o Automático), el sentido de giro del motor y la velocidad (Ilustraciones 32 y 33).

Ilustración 32. LCD modo manual

Ilustración 33. LCD modo auto

Por último el tercer módulo o *Control Write* se basa en dos contadores con sus comparadores que mandan la señal a una máquina de estados (Explicada en el punto 3.6.2) de forma que cada 10ms se habilite la escritura durante 5 ciclos, que son los necesarios para refrescar los 5 datos (Sentido, Centenas, Decenas, Unidades y Decimas) que se muestran por pantalla y que se deben actualizar.

3.5.2 Ram Data Adress Controller

Este bloque (Ilustración 34) tiene la función de recibir los datos en binario natural, traducirlos al lenguaje del LCD y enviarlos, junto a las direcciones del LCD donde se deben de visualizar dichos datos a la RAM para la correcta visualización. El envío de los datos y de las direcciones a la RAM no sigue el mismo procedimiento, por ello su implementación se ha separado en dos bloques diferentes (Ilustración 34, Datos y Direcciones)

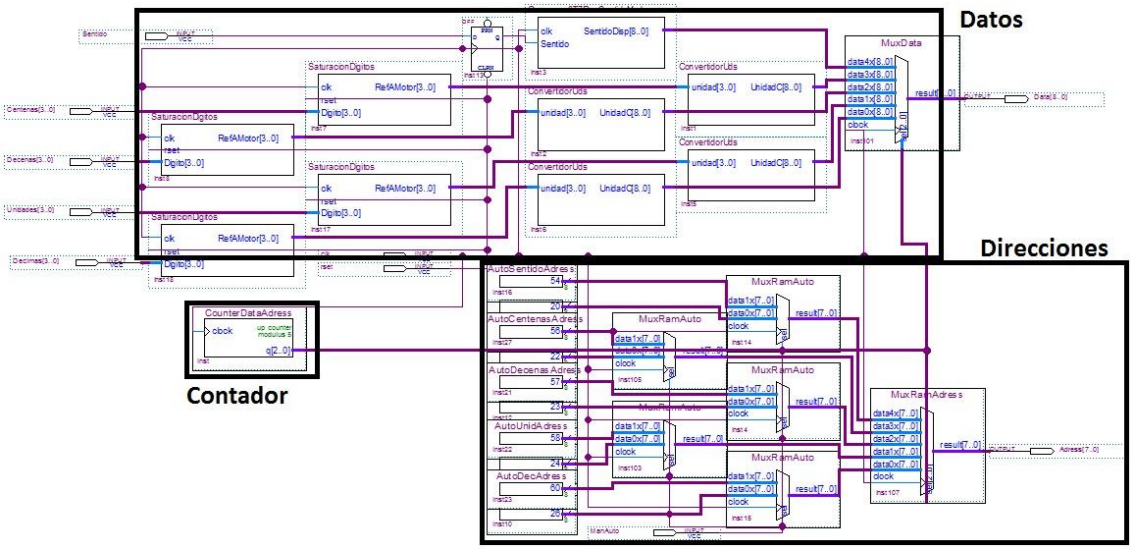

Ilustración 34. RAMDataAdressController

El sistema está gestionado por un contador de 3 bits (Ilustración 34, Contador). Encargado de contar en módulos de cinco, de manera que en la ventana de 5 ciclos en los que tenemos habilitada la escritura por el *Write Enable* (Punto 3.6.2), el multiplexor pase por todas sus posiciones.

El envío de los datos y de las direcciones a la RAM sigue distintos procedimientos.

3.5.2.1 Datos

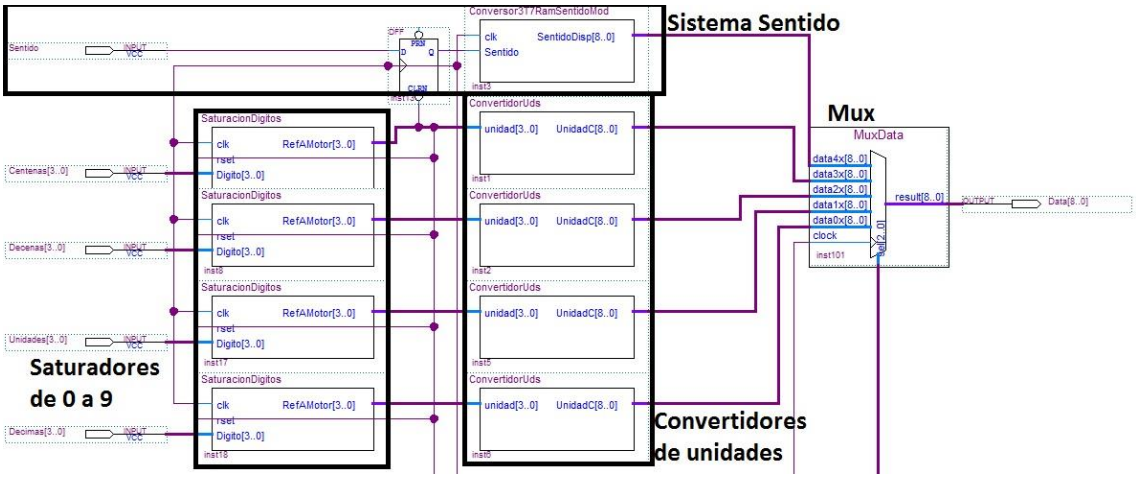

Ilustración 35. Datos (Expandido para mejor visualización)

Los datos a mostrar son el sentido y los 4 dígitos que componen la velocidad. La visualización del sentido de la marcha (Ilustración 35) tiene un tratamiento distinto a los datos numéricos ya que en este caso se utilizan letras en vez de números.

El sentido es una señal de 1 bit que vale 1 cuando la marcha es hacia delante y 0 cuando es marcha atrás. Esta señal de un bit debe ser transformada en las letras D y R respectivamente, de 9 bits cada una y para ello utilizaremos el bloque convertidor (Ilustración 36).

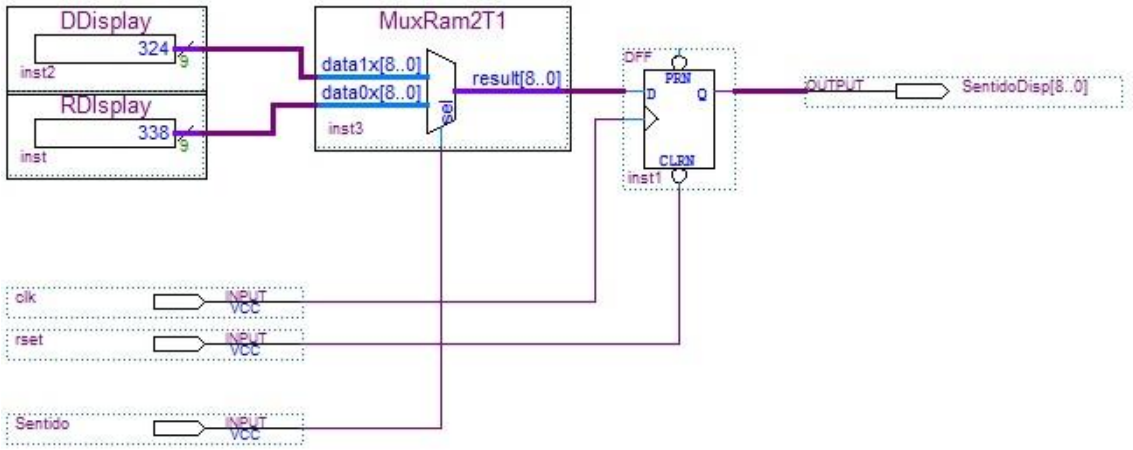

Ilustración 36. Bloque convertidor sentido

El bloque convertidor (Ilustración 30) está compuesto de un multiplexor, dos constantes y un biestable.

El multiplexor (MuxRam2T1) realiza la función de convertir la señal de Sentido (0 o 1) en una de las dos señales asignadas a constantes, que son R y D. Las entradas son las constantes que corresponden con los códigos de cada letra y la señal de Sentido selecciona cuál de las entradas pasará a ser salida del multiplexor.

Esta nomenclatura, se ha adecuado a la aplicación concreta de este trabajo, que es un coche eléctrico. La nomenclatura D y R para Marcha Adelante y Marcha Atrás se ha obtenido de un cambio de marchas automático. Así pues, en nuestro trabajo se ha asignado el 1 (Giro a derecha) como Marcha Adelante (D) y el 0 como Marcha Atrás (R).

El convertidor de unidades (Ilustración 35) tiene otro principio de funcionamiento. Previo paso por el convertidor de unidades, los datos pasan por un saturador inicial (Ilustración 35) que limita las señales entre 0 y 9. Esto se realiza para evitar problemas asociados a que el bloque que obtiene los dígitos (Conversor al sistema decimal, Ilustración 19) opera cada digito sucesivamente lo que generaba señales transitorias erróneas que no correspondían a los 10 números representables en ausencia de los saturadores iniciales.

Posteriormente, el convertidor de unidades (Ilustración 35) se encarga de pasar esos dígitos de 4 bits a señales propias de los caracteres del LCD (9 bits). Como los 4 bits menos significativos de este código corresponden con el número a representar, y los otros 5 son una constante, la implementación del bloque que compone la señal es sencilla (Ilustración 37).

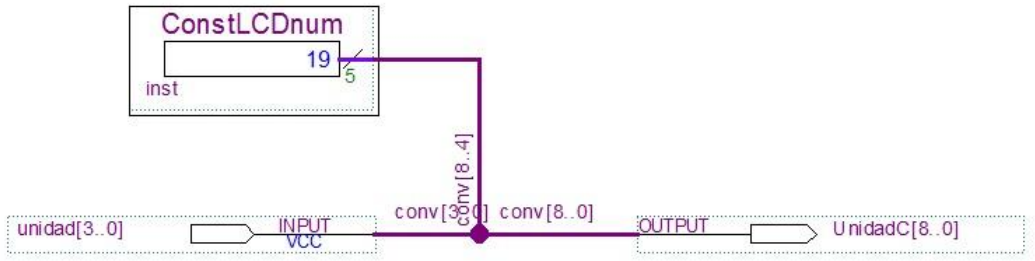

Ilustración 37. Convertidor de unidades

El motivo por el cual no se puede emplear este sistema para generar las letras es que estas presentan un código sin parte constante y la señal de sentido no puede ser aprovechada para generar el código del carácter.

Finalmente, el multiplexor (Ilustración 35) conecta las señales con la RAM secuencialmente para mostrar toda la información por el LCD.

3.5.2.2 Direcciones

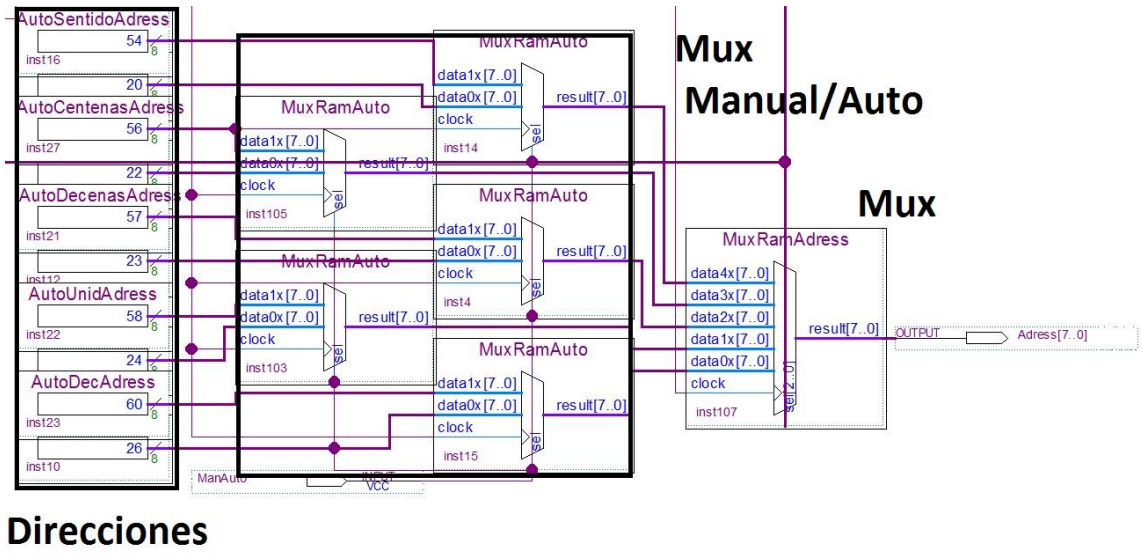

Ilustración 38. Direcciones

La implementación de la codificación de las direcciones sigue un esquema similar a la parte de datos pero como en este caso las direcciones son constantes, aparecen como bloques (Direcciones de memoria, Ilustración 38). La única variación en las direcciones se produce cuando se escribe en una plantilla u otra y como estas están coordinadas con el modo manual o automático es esta misma señal la que coordina una batería de multiplexores que presentan una u otra dirección. (Mux Manual/Auto, Ilustración 38)

El último multiplexor (Ilustración 38) conecta todas las direcciones habilitadas (Las de control manual o las del automático) secuencialmente con la entrada de direcciones de la RAM.

3.5.2.3 Contador

El contador (Ilustración 34) se encarga de coordinar las entradas seleccionadas del multiplexor de datos y de direcciones, de forma que se asegure siempre que el dato se envía a la dirección correcta. Este contador cuenta en módulos de 5 que corresponde con el número de datos a enviar. Cuenta de 0 a 4 y vuelve a empezar, y así aseguramos que se envíen todos los datos cada vez que se refresca la pantalla.

Puede parecer problemática la relación entre este bloque y el *Control Write* (Ilustración 31) que ya hemos mencionado y que posteriormente se explicará al detalle. Pero no hay problemas aunque permanezcan independientes pues el *Control Write* habilitará la lectura durante 5 ciclos cualesquiera, de forma que aunque nuestro contador no esté a cero en ese momento si recorrerá todos los valores de las entradas desde el multiplexor hasta el valor de entrada que tenia al inicio, refrescando todos los dígitos.

3.6 Otros Bloques

3.6.1 Display Converter (Pasa la señal Auto/Man de 1 a 3 bits)

La señal que indica si estamos en modo automático o manual sale de la máquina de estados del control de crucero (Ilustración 30) en forma de un bit, pero debido a que el bloque que controla la lectura de la RAM (Control Motor Completo, Ilustración 12) venia previamente diseñado, se debe adaptar a una señal de 3 bits.

Para ello se emplea en el *Display Converter* (Ilustración 14) una técnica idéntica al convertidor de sentido para el LCD (Bloque convertidor de Sentido, Ilustración 36) que convierte de binario a código para el LCD

Así pues, un multiplexor con dos entradas y controlado por la señal Manual/Automático se encarga de conectar la salida con las dos constantes de 3 bits que representan el 1 (Automático) y el 0 (Manual) (Ilustración 39).

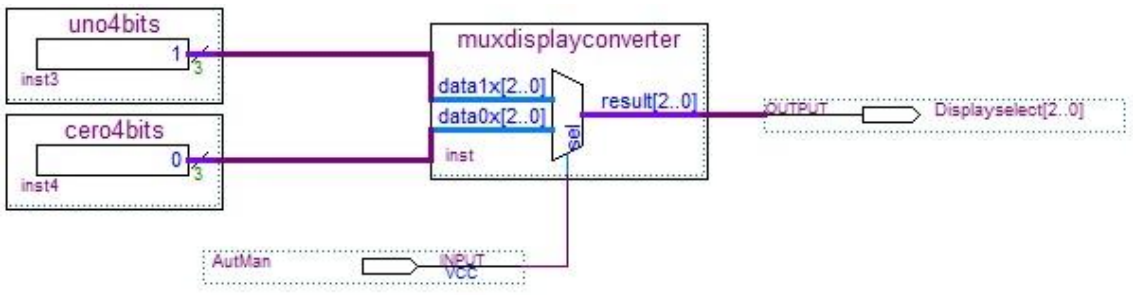

Ilustración 39. Display converter

3.6.2 Control Write (Temporizador)

Se trata del bloque encargado de comunicarle a la RAM cuando debe escribir (Ilustración 31). En la práctica, funciona como un doble contador por lo que sirve de ejemplo para todos los temporizadores del sistema.

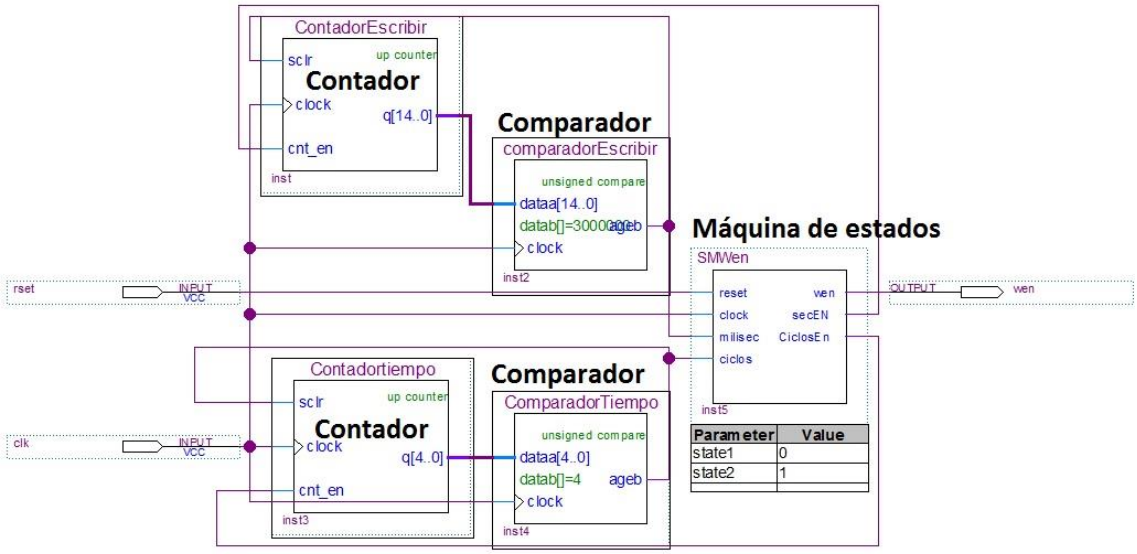

Ilustración 40. Control write

La máquina de estados (SMWen, Ilustración 40) inicia la cuenta del contador (Ilustración 40) mediante la entrada *Count Enable* (puede iniciarse siempre o esperar a alguna señal que le pida hacerlo) y este cuenta y envía los valores a un comparador (Ilustración 40) que detecta cuando la cuenta sobrepasa cierto valor, momento en el cual este envía una señal (normalmente un a>b) a la máquina de estados (Ilustración 40), que realiza la acción para la que este diseñado ese temporizador.

En este caso particular, la maquina pasa entre dos estados durante diferente tiempo, 10ms en los que no se realiza acción y 5 ciclos durante los cuales se habilita la escritura.

3.6.3 AntiWindUp

El antiwindup (Ilustración 29) es un sistema empleado fundamentalmente en PID para evitar que el sumador integral del error crezca cuando la señal de control está saturada.

Se ha empleado para impedir el que el control de crucero siga sumando o restando a partir de cierto nivel debido a que esto comporta problemas de *overflow* o *underflow* en los operadores provocando el mal funcionamiento del sistema. (Ilustración 29)

Este sencillo bloque se encarga de comparar la señal que va a ser sumada o restada con dos limites, mediante comparadores (Ilustración 41) tarados en esos límites. Cuando estos son sobrepasados, se envía la señal correspondiente de bloqueo suma o bloqueo resta.

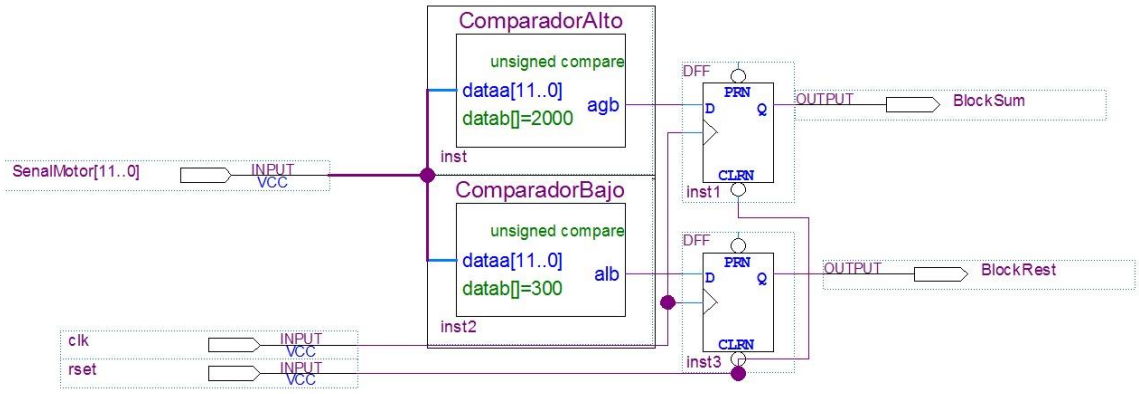

Ilustración 41. ***Antiwindup***

3.6.4 Saturador

El saturador (Ilustración 29) evita que una señal sobrepase cierto valor, de forma que se pueden proteger partes del circuito de fallos por enviarles señales que no son capaces de manejar.

El sistema compara, al igual que el antiwindup, la señal con una referencia por medio de comparadores tarados a esos límites.

La señal emitida por estos al sobrepasar los límites es empleada en cambiar la entrada de un multiplexor, que deja de conectar la entrada directa de la magnitud a controlar, para conectar una entrada de valor constante el límite, de forma que este no se puede sobrepasar en la salida (Ilustración 42).

El sistema es válido tanto para valores altos como para valores pequeños, en cuyo caso, el comparador debe activarse cuando la señal sea menor que el límite impuesto.

El motor empleado para las pruebas por ejemplo, no responde hasta una señal interna de 300 y por encima de 3000 el PWM no es capaz de generar correctamente la onda. Sin embargo, el límite inferior ha sido fijado teniendo en cuenta solo el evitar *underflow* en los operadores que no tienen *antiwindup*. El límite superior se ha fijado con un margen generoso para evitar que cualquier suma sobrepase los 3000 de referencia máxima. Recordar el límite inferior asignado es de 30 y el superior de 2800.

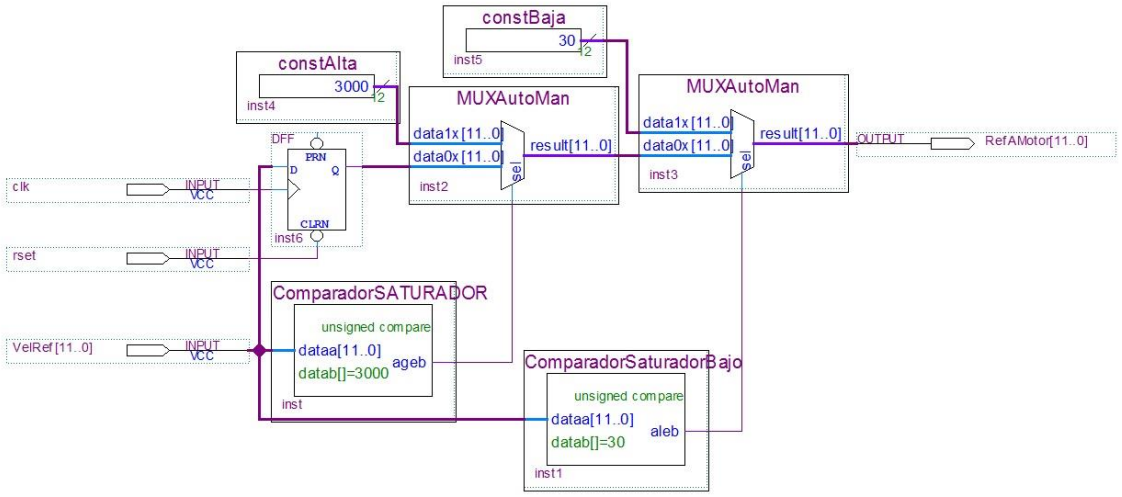

Ilustración 42. Saturador

3.6.5 Referencias Luminosas

También se han introducido dos sistemas luminosos de información para comodidad y seguridad del usuario.

El primero son las luces de freno, activadas junto con la señal de freno pulsado. (Simulador Inercia Coche, Ilustración 28)

Estas luces se corresponden con los dos leds extremos (Ilustración 44) de la DE0-NANO señalizan la intención del conductor de perder velocidad.

El segundo sistema es un velocímetro de apoyo al del LCD. Este consta de los 6 leds centrales (Ilustración 44) que se encenderán de izquierda a derecha según se aumente la velocidad.

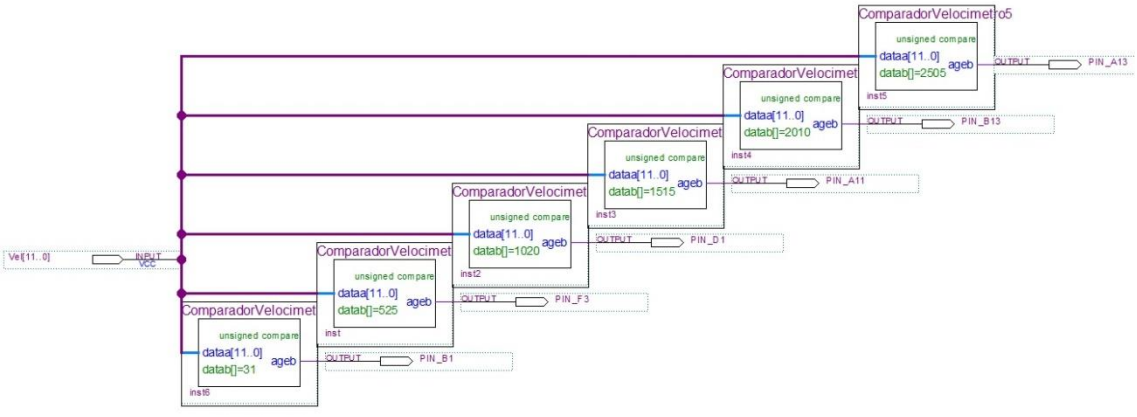

Ilustración 43. Velocímetro luminoso

Este Velocímetro Luminoso (Bloque Nivel Alto, Ilustración 14) ha sido preparado para encender el primer led en cuanto se empiece a mover el vehículo y el resto a intervalos regulares hasta los 250 Km/h momento en el que todos los leds estarán encendidos.

Como se observa en la Ilustración 43, esto se ha conseguido mediante una batería de comparadores que mandan la señal de encendido a los leds según la velocidad vaya aumentando.

En la Ilustración 44 se pueden localizar los leds que actúan en cada caso

Ilustración 44. Esquema luces

4 Trabajo en laboratorio

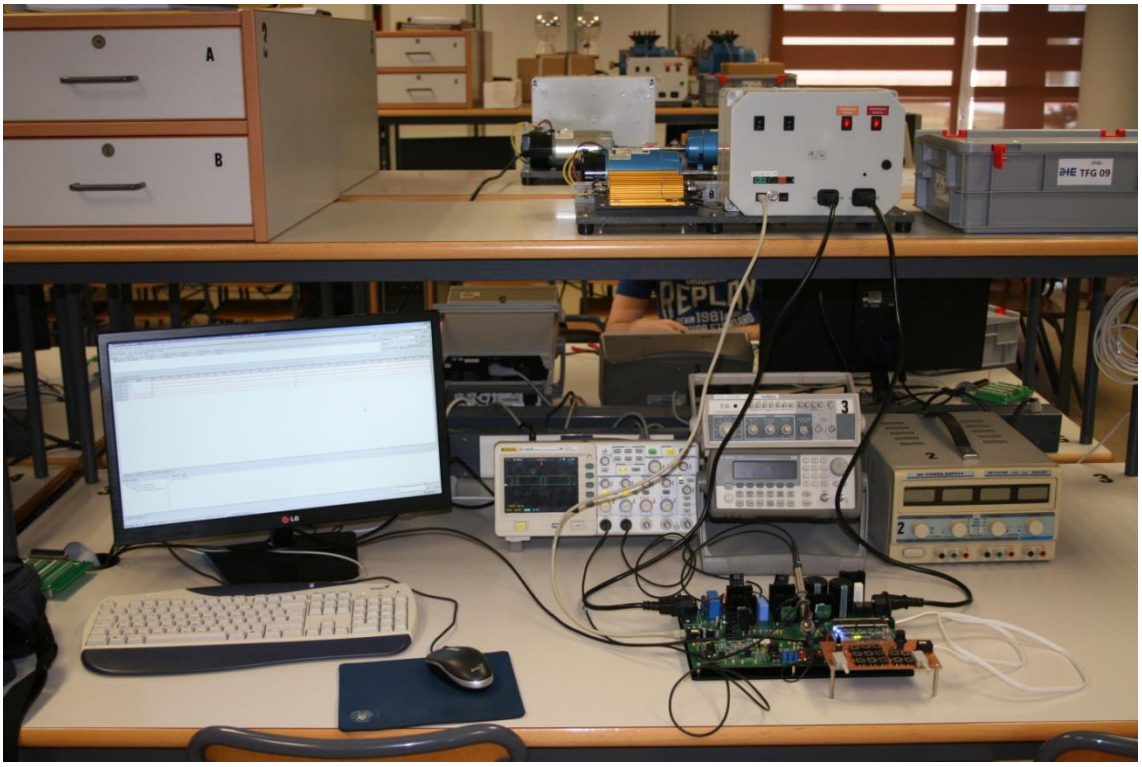

Ilustración 45. Puesto de pruebas

4.1 Puesto de Pruebas

Para implementar el control del motor se disponía en el laboratorio de una placa de potencia con el puente en H (Punto 2.9), una FPGA CYCLONE IV E (2.6) instalada en una tarjeta DE0-NANO(2.7), un LCD (2.11), una botonera con dos potenciómetros con retorno por muelle (2.10), un motor DC (2.3) y una fuente de alimentación. Además del equipamiento normal de cualquier laboratorio de electrónica, como un ordenador y un osciloscopio (Ilustraciones 45, 46 y 47).

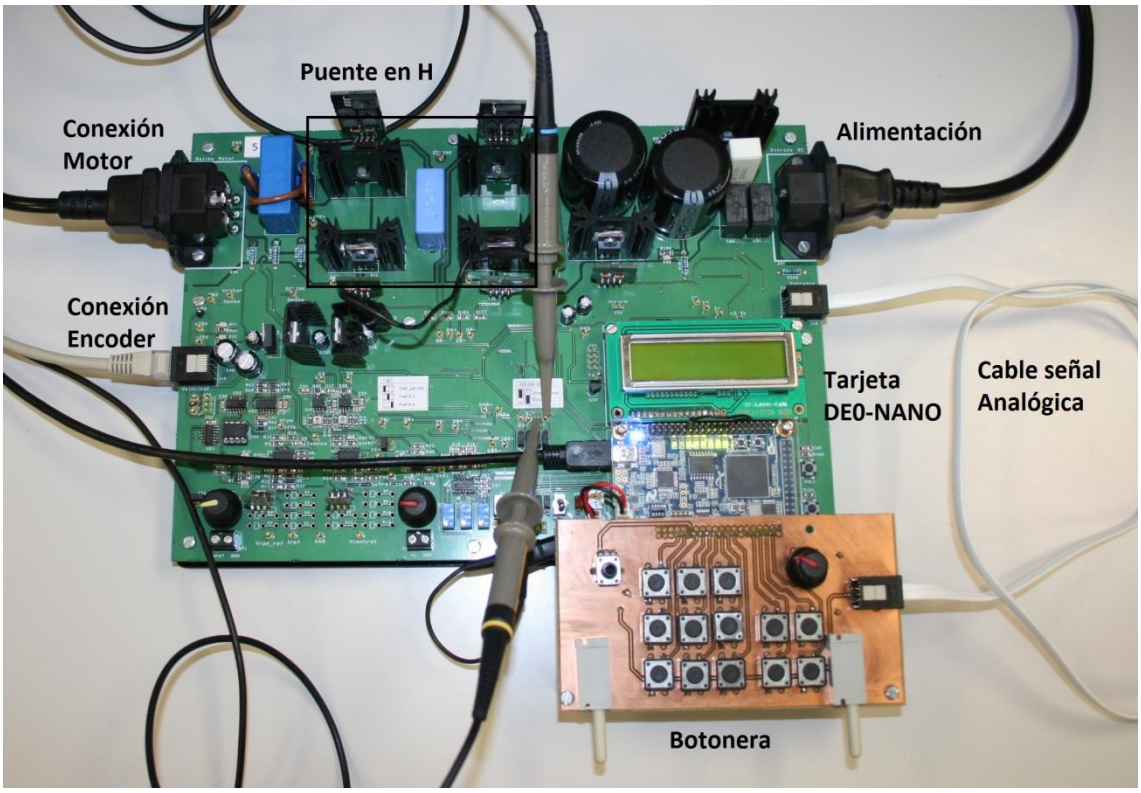

Ilustración 46. Puesto de pruebas

En la izquierda de la Ilustración 46 se observa un cable blanco que corresponde a la entrada del encoder y en la derecha se observa otro cable blanco encargado de llevar la señal analógica de los potenciómetros de la botonera al convertidor AD de la tarjeta.

Los dos cables negros conectados en la parte superior de la placa (Ilustración 46) son la alimentación (Derecha) y la señal enviada al motor (Izquierda) y se conectan con la fuente de alimentación (Ilustración 47)

El montaje del puesto de pruebas queda finalmente como se observa en la Ilustración 47 con las sondas mostrándonos a través del osciloscopio la señal de control del puente en H (Ilustración 48).

.

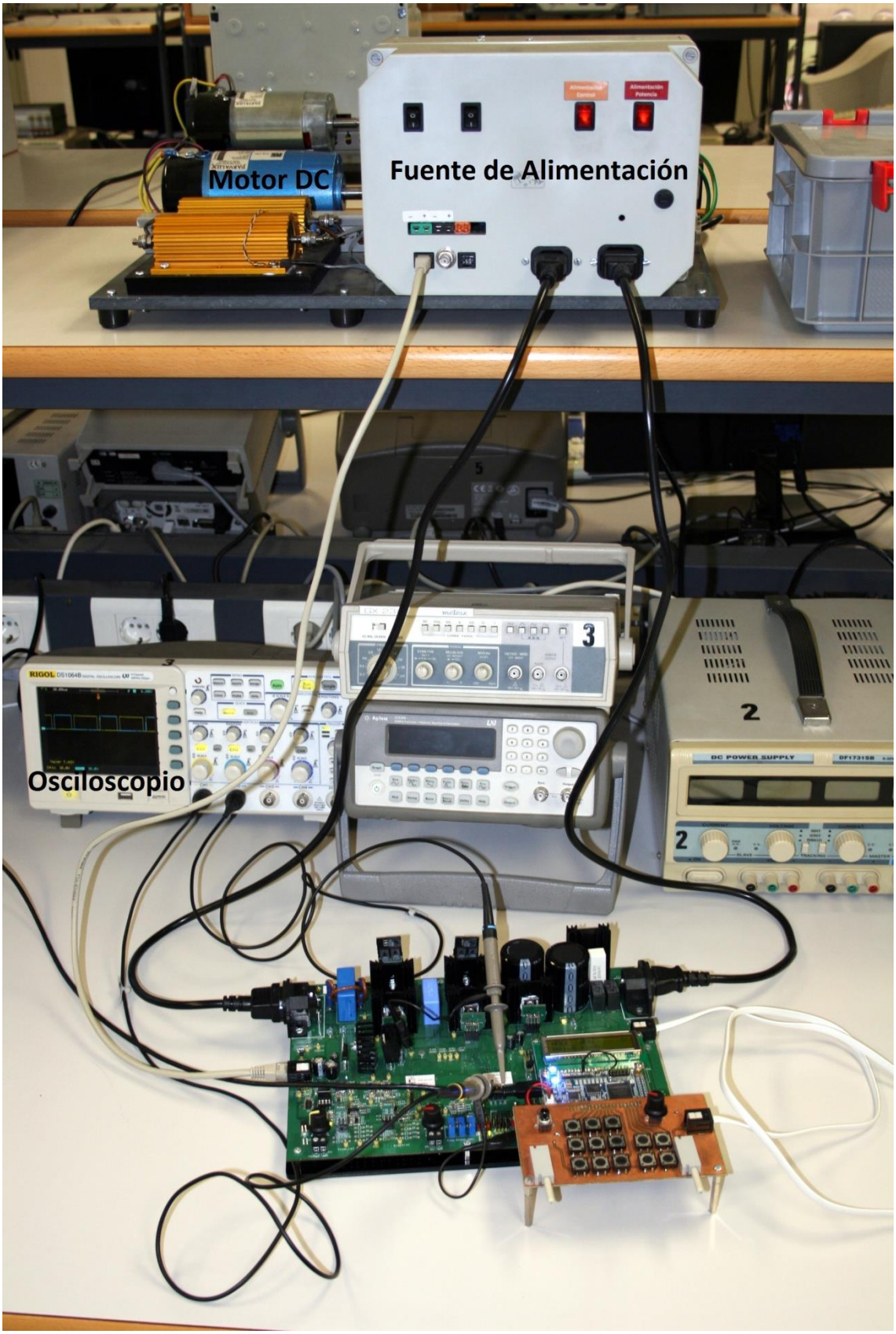

Ilustración 47. Montaje

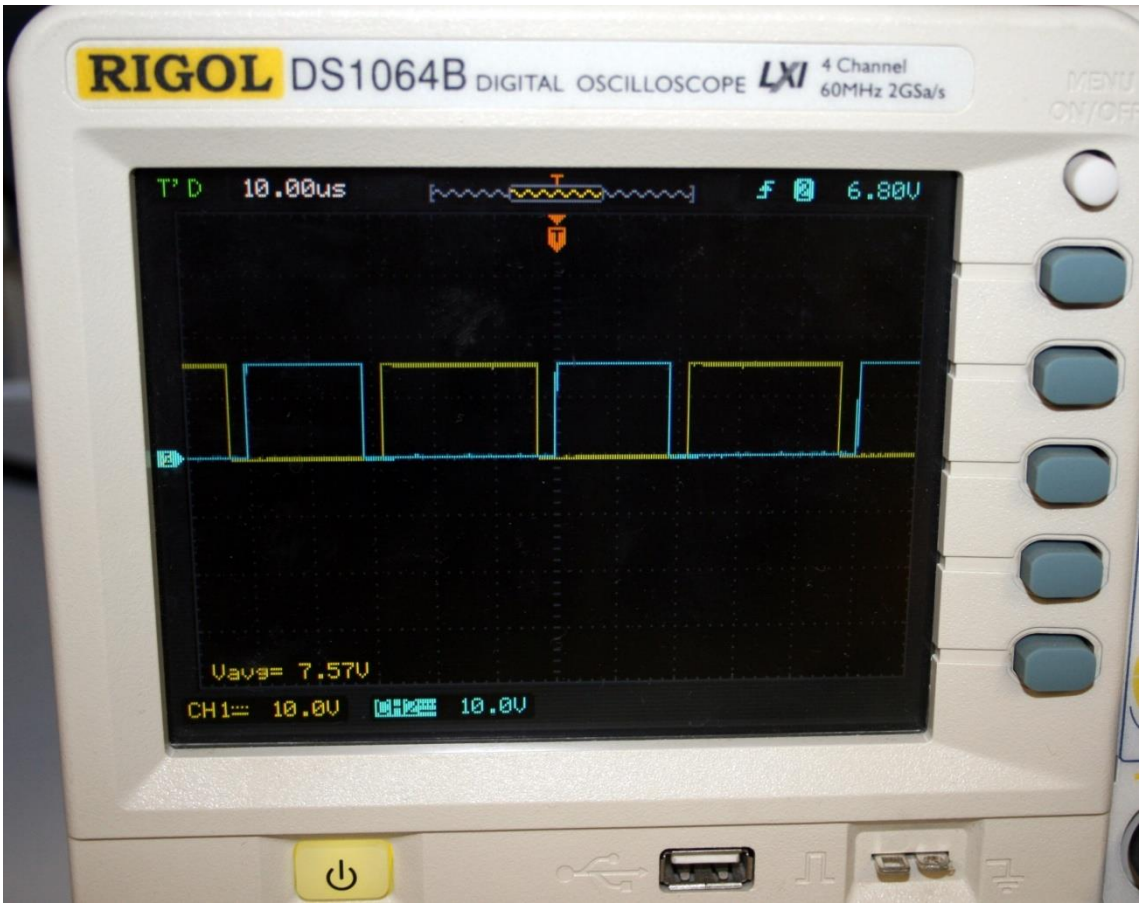

Ilustración 48. Señales enviadas al puente en H

A través de la fuente de alimentación también se envía la señal del encoder mediante el cable blanco antes descrito y que ahora podemos ver también en la Ilustración 47.

4.2 Pruebas de Software

Las primeras pruebas del control de crucero se realizaron con un diseño realizado durante los meses anteriores y solo probado en el simulador funcional de Quartus por lo que los fallos aparecieron enseguida.

Posteriormente se realizaron numerosas modificaciones del software hasta llegar al programa definitivo.

Para mantener cierta continuidad explicaré los fallos y correcciones de cada sistema independientemente aunque en muchos casos estos fallos se dieron simultáneamente y fueron resueltos también al mismo tiempo.

4.2.1 Rebote de Pulsadores y pulsos inducidos

Los primeros diseños de *Bouncers* (3.4.1) eran capaces de detectar el pulso de un botón pero tenían algunos problemas como pulsos inducidos en otros botones y pulsaciones múltiples en el botón. Se pulsaba una vez pero se enviaban varios pulsos al sistema.

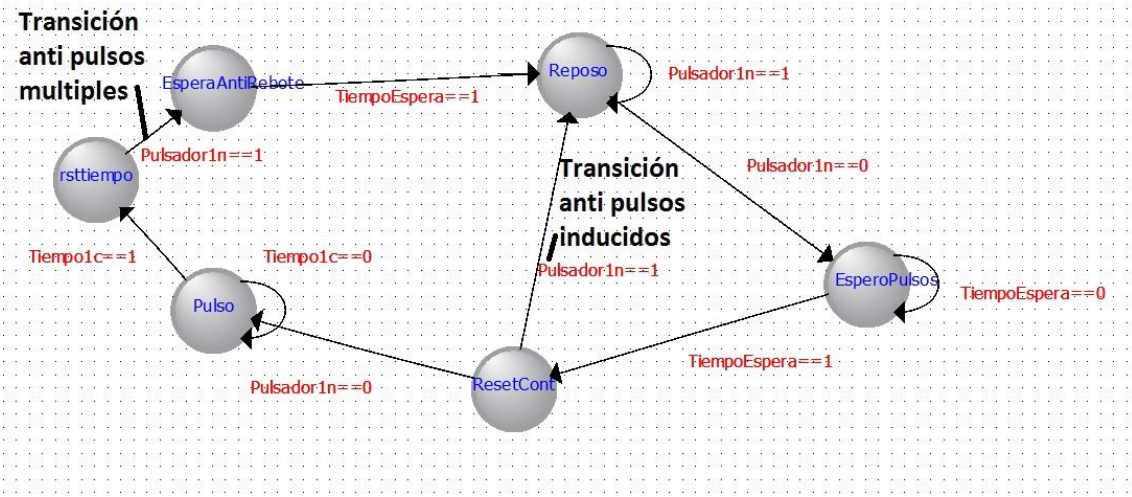

Ilustración 49. Máquina de estados del bouncer

Para resolver este problema se trabajó insistentemente en la máquina de estados (Ilustración 49), introduciendo la posibilidad de que si el pulso que se ha detectado cesa antes de un periodo de tiempo, este ha podido ser debido a un pulso inducido y es ignorado por el sistema (Transición anti pulsos inducidos, Ilustración 49).

Para evitar las pulsaciones múltiples se introdujo en la última transición la condición de que el pulso hubiese cesado para poder dar otro (Transición anti pulsos múltiples, Ilustración 49).

4.2.2 *Overflow* en los operadores y saturación de las salidas

En las primeras pruebas del control automático, podíamos observar gracias al osciloscopio un comportamiento extraño de la señal de control cuando se subía o bajaba velocidad cerca de los puntos de señal máxima y señal mínima.

Se llegó a la conclusión de que estos problemas ocurrían debido a que los operadores, tienen limitados los valores máximos y mínimos de sus salidas por el número de bits. Por lo tanto, cuando sus salidas no podían sumar más porque se sobrepasaba el rango de la cuenta, el valor de la cuenta pasaba a cero y se continuaba sumando otra vez a partir de ahí o el caso contrario, al no poder restar mas pasaban a su valor más alto y restaban desde ahí.

Esto era un grave problema ya que el bajar o subir la velocidad del vehículo incontroladamente haría que el sistema no fuese seguro para usar en la conducción.

Para ello se han instalado 2 medidas el *antiwindup* y el saturador.

El *antiwindup* (3.6.3) manda una señal a la máquina de estados del control de crucero cuando detecta un valor de salida al motor próximo al límite, de forma que bloquea la opción de subir o bajar velocidad según sea el caso. Así se impiden los temidos *overflow* y *underflow*.

El saturador (3.6.4) también detecta el momento en el que el valor a controlar esta cerca de los límites, pero en este caso, en vez de bloquear la operación, lo que hace es vía multiplexor fijar la referencia a una constante. Así se evitan también los problemas con los operadores.

4.2.3 Transiciones en el régimen de giro del motor

Aunque este fallo no fue de los primeros en detectarse, si que fue el más grave a la hora de limitar la posibilidad de realizar pruebas con el motor sin que la placa de potencia sufriese daños.

El control del motor que hay en el laboratorio es un control en bucle abierto, de forma que cambiaba el régimen de giro del motor bruscamente con la referencia (Acelerador).

A la hora de manejar estos cambios bruscos, el modulador PWM que en funcionamiento normal genera una señal similar a la de la Ilustración 48, pasaba a generar una señal sin los tiempos muertos (Ambas señales a cero) que se observan entre cada onda cuadrada.

Esto en la práctica significa que los 4 transistores del puente en H conducen simultáneamente, generando un cortocircuito con altas intensidades que queman los transistores de potencia.

Para evitar este problema se introdujo el bloque Simulador Inercia (3.4.2) que simula a un control en bucle cerrado.

Como ya se explico en el diseño, este bloque se realimenta de la señal final que se le está enviando al motor y mediante un operador controlado se aumenta o disminuye esta señal hasta alcanzar la referencia (Acelerador).

Se aprovecho este sistema para simular el freno motor y el freno físico del coche.

Una vez solucionado este problema, se pudo emplear el montaje completo con el motor encendido.

4.2.4 Parpadeo y desaparición de los caracteres del LCD

Este fue el problema más complicado de resolver para la correcta funcionalidad del sistema pues con el motor parado, el LCD funcionaba perfectamente y cambiaba entre las plantillas de los modos manual y automático sin problema alguno. Sin embargo, al acelerar el motor los caracteres parpadeaban y finalmente desaparecían, quedándose el LCD en blanco.

Tras muchas pruebas, se llego a la conclusión de que el fallo se debía al sistema mediante el cual se convertían los dígitos de binario natural al carácter del LCD.

En un principio, esto se llevaba a cabo con un multiplexor de forma muy parecida a como se hace para el sentido en el punto 3.5.2.1 (Ilustración 36).

El valor en binario natural entraba por la puerta de selección del multiplexor y cada entrada estaba asociada a una constante con el código del carácter completo. Si el valor en binario era un 2, la puerta de selección conectaba la entrada 2 que a su vez estaba conectada a una constante con la representación del número 2 en el LCD.

Este cambio no se producía tan rápidamente como variaban los valores de cada digito y por lo tanto se enviaban los datos con una serie de fallos que acababan por desconectar el display.

Finalmente, se descubrió que el código para los números era el mismo número en los 4 bits de menor peso y una constante en los 5 bits de mayor peso.

Mediante una constante y un bus que se definió como portador de los 5 bits de mayor peso se pudo hacer de una manera más rápida la conversión (Ilustración 50).

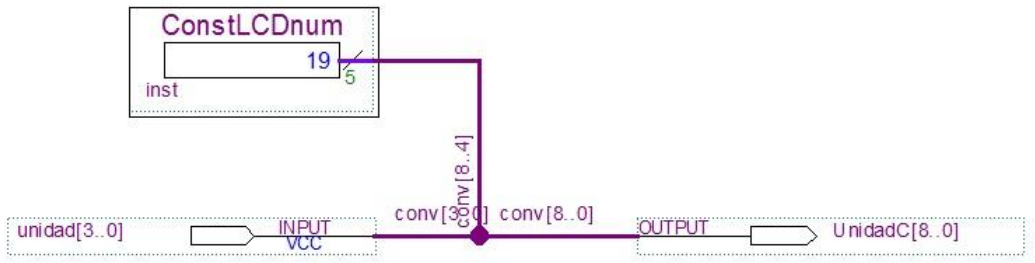

Ilustración 50. Convertidor

4.2.5 Datos Falsos del bloque Sentido de Giro

El bloque Sentido de Giro (3.3) enviaba su señal mediante un biestable que se mantenía siempre activo, lo que propiciaba que se enviaran datos aun cuando la señal de dato correcto (Ready) que generaba la máquina de estados no estuviese activa.

Esto generaba problemas en el display ya que, al variar tan deprisa el dato por culpa de datos falsos, no permitía leer con claridad el sentido.

Como solución se opto por cambiar el biestable a uno con sistema *ENABLE* de forma que la señal de dato correcto activa el biestable para enviar al sistema solo datos correctos.

4.2.6 Conversión al sistema decimal errónea

Una vez todo el sistema funcionaba de forma estable, se pudo apreciar que los valores del velocímetro eran erróneos pues saltaban de una velocidad a otra sin continuidad.

Se llego a la conclusión de que el sistema encargado de separar por dígitos el valor de la velocidad (3.2.2) debía estar fallando.

Fue confirmado posteriormente en una simple simulación en la que los se apreciaba que los cálculos que realizados no daban como salida los dígitos de la velocidad.

Para solucionar este problema, se empleó un algoritmo similar al que se encarga de pasar de decimal a binario, solo que al querer pasar a decimal dividiríamos entre diez.

(Ver el cálculo del punto 3.2.2).

También se ha añadido un sistema de control que almacena un dato de velocidad y lo mantiene durante los ciclos 4 ciclos que tarda el bloque en obtener los 4 dígitos, enviándolos solo cuando los dígitos sean correctos y permitiendo entonces aceptar otro valor de referencia de velocidad.

5 Modo de uso

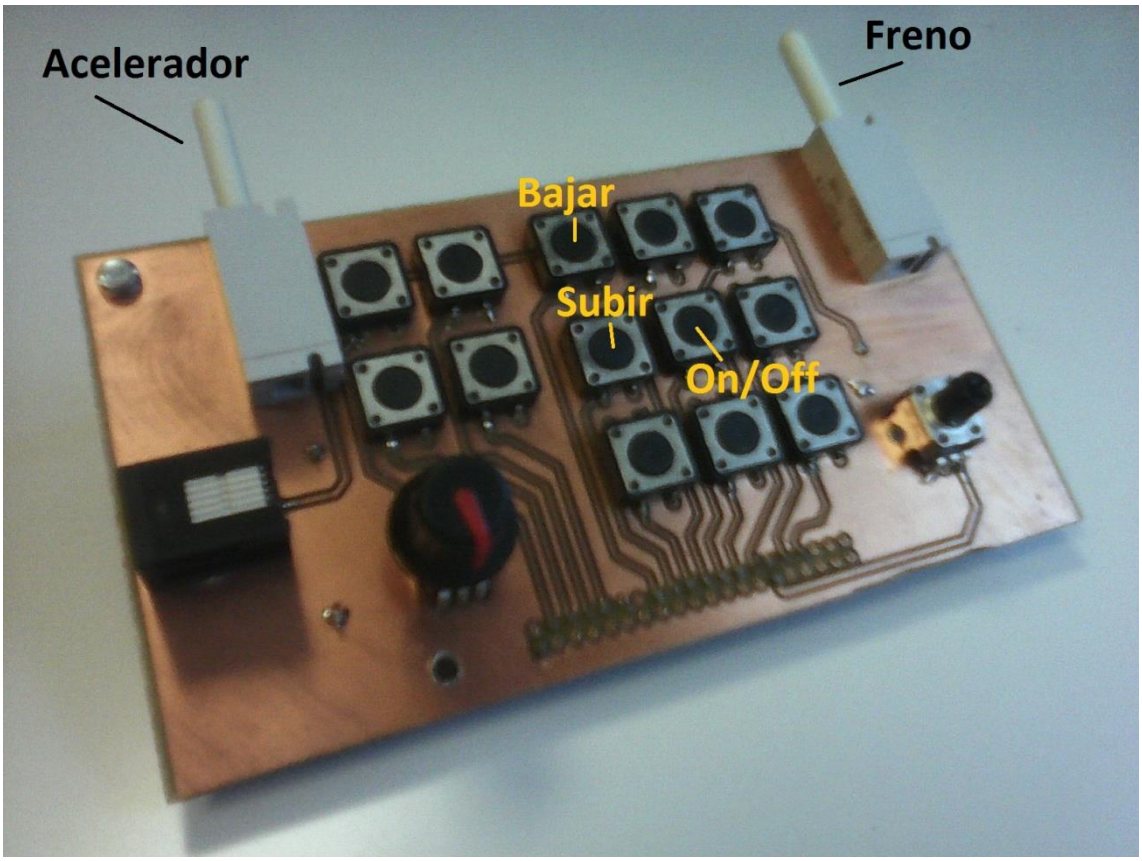

Ilustración 51. Botonera con la asignación de botones

El sistema por defecto comienza en modo manual con el vehículo parado y con la pantalla del LCD mostrando MODO MAN tal y como se observa en la ilustración 52.

Ilustración 52. Modo manual

Para iniciar la marcha solo hay que presionar el acelerador (Ilustración 51), de forma que comenzará a aumentar la velocidad a un ritmo constante (Por más rápido que se presione este ritmo no variará, solo aumentará la velocidad limite a la que dejará de subir).

Cuando se suelte el acelerador (Ilustración 51), entrará en funcionamiento el simulador de freno motor que ira deteniendo el vehículo también a ritmo constante.

Si fuese necesario, el freno (Ilustración 51) está en todo momento disponible para detener el vehículo más rápidamente que con el freno motor.

Durante la marcha, podrá activar el control de crucero. Para ello hay que pulsar primero el botón On/Off y al estar en la velocidad deseada presionar el botón Bajar. Automáticamente quedará la velocidad fijada y podrá observar en la pantalla MODO AUTO.

Ilustración 53. Modo auto

En este modo podrá bajar y subir la velocidad gradualmente con los botones Bajar y Subir (Ilustración 51). Resaltar que la subida de velocidad si es instantánea, pero no así

la bajada que empleará por defecto el freno motor, salvo que se pulse el freno en cuyo caso perderá velocidad más rápidamente. Durante la bajada de velocidad aparecerá en la pantalla MODO MANUAL (Ilustración 52) hasta que se alcance la nueva velocidad menor.

También podrá acelerar en este modo, y una vez se detecte que supera con el acelerador la velocidad fijada para el modo crucero este responderá aumentando la velocidad como si del modo manual se tratase (La pantalla también mostrará MODO MANUAL). También a efectos de frenada actuará como si estuviera en modo manual, hasta llegar a la velocidad de referencia donde volverá el modo automático.

Para desconectar el modo automático tiene dos opciones. Presionando el botón On/Off o presionando el freno, volverá al modo manual.

En el LCD tendrá además de la información del modo de marcha, un indicador del sentido en el que está circulando (Marcha adelante o Marcha atrás) así como el velocímetro (Ilustraciones 52 y 53).

También contará con las referencias visuales de Luz de freno y Velocímetro luminoso (Ilustración 54).

Ilustración 54. Sistema de luces

La luz de freno, está directamente accionada por la presión sobre el freno y el velocímetro luminoso ira iluminando un mayor número de leds según aumente la velocidad.

6 Conclusiones

El objetivo de este proyecto es el desarrollo de un sistema que permita controlar la puesta en marcha, apagado y velocidad del motor de un coche mediante potenciómetros (simulando acelerador y freno) con un modo de control de crucero automático que también permita regular la velocidad de referencia.

Se ha diseñado el control digital síncrono del motor DC presente en un coche eléctrico utilizando una FPGA CYCLONE IV E implementada en una tarjeta DE0-NANO.

Este sistema es capaz de recoger unas referencias de acelerador, freno y 3 botones para, mediante control digital síncrono, controlar la puesta en marcha, el apagado y la velocidad del motor de corriente continua de un modo suave.

La interacción con el motor se realizará mediante dos modos (Manual y Automático) diseñados para una cómoda conducción.

El modo manual es el activado por defecto y permite la conducción habitual, es decir, el conductor incrementará la velocidad del motor libremente presionando el acelerador. La presión que se ejerza en el acelerador marcará cuan alta es la velocidad que adquirirá el vehículo. El acelerador ha sido calibrado para aumentar la velocidad del vehículo de forma constante, previniendo así forzar el motor eléctrico.

En este modo también podremos soltar el acelerador, momento en el cual la velocidad irá descendiendo poco a poco simulando el freno motor de un vehículo de combustión interna. Y por último, también se ha implementado un freno que reduce la velocidad de forma más brusca que el freno motor y permite detener el vehículo.

El modo automático puede ser activado en cualquier momento de la marcha pulsando secuencialmente los botones OnOff y Bajar Velocidad (Ilustración 51) y una vez activado mantiene la velocidad del vehículo constante permitiendo al conductor relajar el pie del acelerador. Este modo nos permitirá también subir y bajar la velocidad a la que el modo automático mantiene al vehículo mediante los pulsadores Subir y Bajar.

En caso de necesitar incrementar momentáneamente la velocidad podremos también presionar el acelerador y el sistema responderá inmediatamente, devolviéndonos a la velocidad de referencia al soltarlo. Y finalmente, por seguridad, el modo automático se desconectará cuando presionemos el freno.

El sistema nos informará mediante el LCD del modo de conducción en el que vamos (Manual o Automático) y de la velocidad y sentido de la marcha que llevamos en todo momento (estos dos últimos datos se obtienen gracias al tratamiento de las señales del encoder).

Y mediante los leds podremos saber si está siendo pulsado el freno y tendremos una referencia visual de la velocidad.

El funcionamiento de este diseño se ha verificado tanto en modo de simulación como en un montaje real en el que se ha conectado la DE0-NANO a una botonera, un display y una etapa de potencia en H que controla el motor DC

6.1 Perspectivas de desarrollo

Por el diseño modular que se ha seguido en todo el proceso es sencillo añadir bloques que interactúen con nuestro sistema, permitiendo posteriores expansiones para aumentar las funcionalidades del sistema.

El control de la velocidad abre la puerta a múltiples aplicaciones que se podrían llevar a cabo gracias a otros sensores como:

- Sensor de proximidad (tecnología sonar o laser): Permitiría adaptar la velocidad del vehículo a la del vehículo que le precede y si es necesario incluso frenar el vehículo en caso de detectar un obstáculo.
- Sensor óptico de lectura de señales: Permitiría fijar las velocidades máximas y mínimas a las que el vehículo puede circular y evitar que el conductor salga de ese margen por despiste. También podría directamente fijar la velocidad a la de la vía y reducir esta en caso de encontrarse con señales de peligro.
- Acelerómetro: Tendríamos la posibilidad de detectar si el vehículo está realmente en movimiento o son las ruedas las que están patinando de modo que el sistema ya integrado de aceleración progresiva (Simulador de inercia) pudiese actuar a modo de control de tracción y adaptar el ritmo de aceleración del vehículo al instante. También en curva podría ser útil en caso de que detecte una fuerza G lateral mayor que la asumible por los neumáticos, controlando el acelerador si fuese necesario.

Otra área en la que podría ser desarrollado el sistema sin necesidad de nuevo hardware seria la obtención de estadísticas a partir de los datos de velocidad del vehículo. Como por ejemplo velocidades medias, consumos, distancias…

7. Bibliografía

- Diseño Digital: principios y prácticas. John F. Wakerly.
- Quartus II: http://dl.altera.com/?edition=web
- CYCLONE IV E: http://www.altera.com/devices/fpga/cyclone-iv/cyiv-index.jsp
- CYCLONE IV E, HANDBOOK: http://www.altera.com/literature/hb/cyclone-iv/cyclone4-handbook.pdf
- DE0-NANO, INFORMACIÓN: http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=593&PartNo=4
- LCD HD44780 de HITACHI: https://www.sparkfun.com/datasheets/LCD/HD44780.pdf

Pliego de Condiciones

1 Descripción del proyecto

El presente documento hace referencia al desarrollo de un sistema de control de crucero para un motor de corriente continua sobre una FPGA CYCLONE IV E.

Las características estarán sujetas a las especificaciones iniciales del promotor, siempre de acuerdo a lo establecido en este Pliego de Condiciones.

El proyecto queda definido en los siguientes documentos

1. Memoria.
2. Pliego de condiciones.
3. Presupuesto.

2 Condiciones Generales

2.1 Disposiciones generales

El fin de este apartado es regular la ejecución del proyecto, acotando las funciones que corresponden al promotor, proyectista y fabricante así como las relaciones entre ellos.

En el apartado, se recogen los aspectos legarles del proyecto y se fijan las condiciones que marcarán la ejecución y puesta en marcha del sistema objeto del proyecto. Se trata pues de un documento de carácter facultativo, económico y legal que regirá en el desarrollo de los distintos módulos que constituyen el presente proyecto.

2.2 Documentación del contrato

Será condición indispensable para el fabricante el conocimiento y aceptación del presente Pliego de Condiciones para la correcta materialización del proyecto. Por lo tanto, este pliego deberá ser incluido por el propietario como un documento mas a firmar por el fabricante al hacerse cargo de la ejecución. El contrato se encuentra formado por los siguientes documentos:

- Condiciones fijadas en el contrato
- Pliego de condiciones técnicas particulares
- El presente Pliego general de condiciones
- Otra documentación del proyecto: Memoria y Presupuesto.

2.3 Condiciones generales facultativas

2.3.1 El fabricante

Corresponde al fabricante:

1. Organizar los trabajos de montaje, así como elaborar los planos necesarios para la materialización del sistema y sus accesorios.
2. Velar por el cumplimiento de la normativa vigente en seguridad e higiene en el trabajo.
3. Coordinar los trabajos de los subcontratistas.
4. Asegurar la idoneidad de los materiales y otros elementos empleados para la implementación del sistema, rechazando aquellos que no cuenten con las garantías exigidas tanto por parte de la normativa como por el presente Pliego de Condiciones.
5. Firmar con el promotor las actas de recepción provisional y definitiva.
6. Hacerse cargo de los seguros por accidente de trabajo durante la fabricación.

Derechos y Obligaciones:

1. Verificar los documentos del proyecto así como poseer un conocimiento claro de la normativa al respecto. El fabricante deberá indicar que la documentación en el presente proyecto le resulta suficiente para la comprensión del mismo y en caso contrario deberá solicitar las aclaraciones pertinentes.
2. El fabricante ejecutará los trabajos necesarios para el correcto montaje y buen funcionamiento del sistema, aun cuando estos no se hallen expresamente indicados en los documentos pero si hayan sido indicados por el ingeniero proyectista. Siempre dentro de los límites del presupuesto.
3. Las modificaciones y aclaraciones de los documentos serán comunicadas por escrito al fabricante, el cual tendrá la obligación de extender un acuse de recibo.
4. Las reclamaciones por parte del fabricante deberán presentarse ante el promotor si son de índole económica o al ingeniero proyectista si se trata de reclamaciones técnicas. En ambos casos deberán presentarse por escrito. Debiendo, las partes expedir como mínimo un acuse de recibo del escrito al recibirlo.

2.3.2 El ingeniero proyectista

Es el responsable último de la ejecución del proyecto. Decide sobre el comienzo, ritmo y calidad de los trabajos.

Corresponde al ingeniero proyectista:

1. Redactar los complementos o modificaciones del proyecto que se precisen.
2. Asistir al montaje de elementos críticos, o cuando se le requiera para asistir al fabricante ante las posibles contingencias que se puedan dar durante el proceso de fabricación.
3. Asesorar al promotor.
4. Exigir modificaciones o añadidos al sistema si así lo cree conveniente, siempre y cuando no constituya una variación excesiva sobre el presupuesto inicial.
5. Coordinar la intervención de otros técnicos especializados en las tareas del punto

2.4 Condiciones generales de la ejecución

2.4.1 Orden de los trabajos

El fabricante será, salvo indicación expresa del ingeniero proyectista, el encargado de determinar el orden de los trabajos.

2.4.2 Prorroga y/o ampliación por causas de fuerza mayor

Si por causas de fuerza mayor el montaje del proyecto sufriese un retraso se le podría otorgar al fabricante una prorroga de duración igual al tiempo que ha estado el montaje parado, decisión que será tomada por parte del ingeniero proyectista.

Si fuese el ingeniero proyectista el que por causa mayor debiese ampliar el proyecto los trabajos de montaje seguirán, si es posible, mientras el ingeniero tramita el proyecto reformado. Si de esta tramitación se generase un retraso en el montaje el ingeniero deberá otorgar la prorroga antes mencionada al fabricante.

2.4.3 Condiciones generales de ejecución de los trabajos

Los trabajos se realizarán de acuerdo al proyecto, a modificaciones aprobadas del mismo y a las órdenes e instrucciones plasmadas por escrito, bajo la responsabilidad del ingeniero proyectista.

Cualquier variación sin el conocimiento y aprobación por parte del ingeniero proyectista, supondrá que el fabricante ejecutante del proyecto responderá de todas las consecuencias técnicas, económicas y legales que se ocasionen.

2.4.4 Trabajos sin prescripciones

En aquellos trabajos para los que no existan prescripciones en los documentos del proyecto, el fabricante se atendrá a las instrucciones dictadas por el ingeniero proyectista.

2.4.5 Trabajos defectuosos

Si por los efectos derivados de una mala ejecución de la fabricación, el ingeniero proyectista detectase defectos, anomalías o calidad deficiente en el producto seria responsabilidad del fabricante la modificación del producto hasta cumplir con lo contratado.

2.4.6 Averías y accidentes en la fabricación

El fabricante se encargará de proporcionar por su cuenta y riesgo, toda la maquinaria y mano de obra necesaria para la materialización del proyecto.

El fabricante es por tanto responsable de la mano de obra que emplee y deberá disponer todos los medios de seguridad necesarios, respondiendo ante el estado en caso de sanciones por este motivo.

2.4.7 Recepción del proyecto

Una vez concluido el montaje, se procederá a la inspección del mismo por parte del ingeniero proyectista. Si este considera necesaria una modificación, se otorgará una prórroga para realizar las modificaciones pertinentes. Cuando el resultado sea admitido, el ingeniero proyectista, el fabricante y el promotor firman el acta de recepción definitiva, documento con el que se finaliza la fabricación del sistema y se entrega este al cliente.

2.5 Condiciones generales económicas

2.5.1 Forma de pago y plazos

El fabricante percibirá el importe íntegro de todos los trabajos ejecutados siempre que estos se ajusten al proyecto o a las modificaciones del ingeniero proyectista. La forma de pago y los plazos serán acordados entre el promotor y el fabricante.

2.5.2 Fianzas

Las fianzas impuestas al fabricante serán devueltas una vez concluya el periodo de garantía del producto. En un tiempo máximo desde su conclusión de 5 días laborables.

2.6 Condiciones legales

2.6.1 Rescisión de contrato

Se considerarán causas suficientes de rescisión de contrato:

1. La modificación del proyecto en forma tal que represente alteraciones fundamentales del mismo a juicio del ingeniero proyectista
2. El incumplimiento por parte del fabricante de las condiciones del presente contrato.
3. La falta de cumplimiento del fabricante de las órdenes recibidas por el ingeniero proyectista.
4. Desviaciones del presupuesto más allá del 100% del mismo.

La interpretación de cuantas otras causas de rescisión que pudieran presentarse corresponderían al ingeniero proyectista.

3 Condiciones Particulares

3.1 Objeto de este capítulo

Exposición de las especificaciones técnicas de la FPGA y todos sus sistemas accesorios que deben emplearse para el correcto funcionamiento del proyecto, así como la naturaleza de algunos periféricos requeridos para el sistema.

3.2 FPGA

La FPGA utilizada para el diseño y optimización del sistema, y que por tanto deberá utilizarse para la fabricación del control de crucero es una CYCLONE IV E EP4CE22F17C6 de ALTERA.

Para el mejor funcionamiento del sistema deberá emplearse este equipo o en caso de imposibilidad de conseguirlo uno con características iguales o superiores.

Características de la CYCLONE IV E de ALTERA:

- 150K de elementos lógicos
- 594Kb de memoria embebida
- 153 pines Entrada/Salida disponibles para el usuario

3.3 Tarjeta DE0-NANO

La FPGA se conecta a los periféricos a través de una tarjeta DE0-NANO, también recomendada para la implementación del sistema debido a que posee componentes necesarios para su correcto funcionamiento.

Características de la DE0-NANO:

- Elementos de conexión:
  - USB Blaster para programación.
  - Dos bloques de 40 pines GPIO cada uno, de los cuales 72 son Entrada/Salida de 5V, dos pines de 3.3V y cuatro pines de conexión a masa.
- Memoria:
  - 32MB SDRAM
  - 2Kb I2C EEPROM
- Reloj: Reloj incorporado de 50 MHz
- Periféricos incorporados:
  - Acelerómetro ADI ADXL345 de 3 ejes y alta resolución (13bit)
  - A/D Convertidor NS ADC128S022 de 8 canales y 12 bits de resolución a una velocidad de entre 50 y 200 Ksps
  - 8 leds
  - 2 pulsadores
- Fuente de alimentación:
  - Puerto MiniUSB (5V)
  - DC Pin en cada bloque GPIO (2 pines de 5V DC)
  - 2 pines externos (3.6-5.7V)

3.4 Pulsadores y potenciómetros

Se requerirán pulsadores que en reposo estén a nivel alto. Este nivel alto deberá corresponder con 3.3 voltios para adaptarse a la entrada de la tarjeta. Sin otras restricciones.

Los potenciómetros instalados para el acelerador deberán generar un voltaje entre 0V en reposo y 3.3V totalmente presionados.

Presupuesto

1 Introducción

En el presente documento se desglosa el presupuesto del desarrollo del proyecto.

Por la naturaleza inmaterial del proyecto, solo se reflejarán los costes asociados a los recursos humanos empleados y la amortización de los equipos.

Quedan fuera del alcance del proyecto los costes de estudios de viabilidad y de mercado que no se realizarán.

2 Coste de la mano de obra

El proyecto se ha realizado en el departamento de ingeniería electrónica (DIE) de la Universidad Politécnica de Valencia. Se trata de un proyecto académico, y por lo tanto el autor no ha obtenido ingresos por la realización del mismo pero si deben computarse las horas de trabajo dedicadas.

Se estiman 300 horas de trabajo dedicadas al proyecto, y repartidas entre diseño del software, horas de laboratorio para implementación y optimización y horas de dedicación a la redacción de los documentos.

El coste de la mano de obra de un ingeniero industrial se sitúa de media en 50 € la hora.

3 Cuadro de precios unitarios

Cuadro de precios unitarios			
Nº	**Unidad**	**Descripción**	**Precio**
1	Hora	Trabajo de Ingeniero en Tecnologías Industriales	50,00 €

4 Cuadro de precios parciales

4.1 Presupuesto parcial de la mano de obra

Nº	Unidad	Descripción	Cantidad	Precio Unitario	Total
1	Hora	Trabajo de Ingeniero en Tecnologías Industriales	300,00 horas	50,00 €	15.000,00 €
Total Mano de Obra Directa					15.000,00 €
Total Mano de Obra Indirecta 10%					1.500,00 €
Total Presupuesto Parcial de Mano de Obra					**16.500,00 €**

4.2 Presupuesto parcial de amortización de equipos

Concepto	Periodo de Amortización	Periodo Computado	Precio	Coste
Tarjeta DE0-NANO	10 años	3 meses	120,00 €	3,00 €
Placa de potencia con puente en H	10 años	3 meses	312,00 €	7,80 €
Motor DC	15 años	3 meses	370,00 €	6,17 €
Osciloscopio	15 años	3 meses	1.310,00 €	21,83 €
Fuente de alimentación	15 años	3 meses	220,00 €	3,67 €
Ordenador ASUS ASPIRE 5738ZG	5 años	5 meses	400,00 €	33,33 €
Software Quartus II	1 año	3 meses	2.000,00 €	500,00 €
Software ModelSim	1 año	3 meses	700,00 €	175,00 €
Total Presupuesto Parcial de Amortización de Equipos				**750,80 €**

5 Presupuesto Total

5.1 Presupuesto de ejecución material

Concepto	Total
Mano de Obra	16.500,00 €
Amortización de Equipos	750,80 €
Presupuesto de ejecución material	**17.250,80 €**

5.2 Presupuesto de Desarrollo

Concepto	Total
Presupuesto de ejecución material	17.250,80 €
Gastos Generales 15%	2.587,62 €
Beneficio Industrial 10%	1.725,08 €
Subtotal Presupuesto Desarrollado	**21.563,50 €**
I.V.A. 21%	4.528,34 €
Total Presupuesto de desarrollo	**26.091,84 €**

El presupuesto total para el desarrollo de un control de crucero de un motor de corriente continua sobre una FPGA CYCLONE IV E asciende a **VEINTISEIS MIL NOVENTA Y UNO CON OCHENTA Y CUATRO EUROS.**