Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Eléctrica

Control de motor brushless con dispositivos embebidos

Autor:

De la Hera Liébana, Daniel

Tutor:

Serrano Sanz, José Andrés
Departamento de Ingeniería Eléctrica

Valladolid, Noviembre de 2019.

Resumen:

En el presente Trabajo Fin de Grado se ha desarrollado el software para el control del motor brushless 44x25 SI de la casa alemana Dunkermotoren mediante dos dispositivos embebidos en tiempo real, SubRio y MyRio, de la casa estadounidense National Instruments. También se ha diseñado y construido el hardware preciso para la comunicación entre el motor y los dispositivos embebidos. Tanto el motor, los dispositivos de control embebido y los demás medios precisos para este trabajo han sido puestos a mi disposición por el Departamento de Ingeniería Eléctrica de la Escuela, con la finalidad de utilizar éstos en las prácticas de laboratorio de algunas asignaturas.

Se muestra en el trabajo las limitaciones que presenta el sistema “SubRio” y la mejor adecuación para el control del motor del dispositivo “MyRio” para su uso en prácticas. La comunicación entre los sistemas embebidos y el ordenador puede realizarse, si se desea, via Wifi.

Palabras clave: Control, Motor, Brushless, LabView, Embebidos

Abstract:

In this Final Degree Project, the software for the control of the brushless motor 44x25 SI of the German house Dunkermotoren has been developed by means of two devices embedded in real time, SubRio and MyRio, from the American house National Instruments. The precise hardware for communication between the engine and embedded devices has also been designed and built. The motor, the embedded control devices and the other precise means for this work have been available to me thanks to the Department of Electrical Engineering of the School, with the purpose of using these in the laboratory practices of some subjects.

The work shows the limitations presented by the “SubRio” system and the best adaptation for the control of the “MyRio” device motor for use in practice. Communication between embedded systems and the computer can be carried out, if desired, via Wifi.

Key-words: Control, Motor, Brushless, LabView, Embedded

El final de una etapa yla perfecta oportunidad para dar un agradecimiento a todas las personas que han estado presentes a lo largo no solo de la ejecución del proyecto sinode toda mi carrera universitaria.

Gracias a Janet, mi mayor descubr

Gracias a mi familia, en especial a mis padres los cuales me convirtieron en lo que soy y cuya ética y valores me guiaron para comenzar esta etapa.

Gracias a Jose Adrés, a Jorge y a todos los profesores y compañeros del Departamento de Ingeniería Eléctrica de la Escuela de Ingenierías Industriales en Valladolid que me inspiraron a elegir esta temática para el TFG y que tanto me han apoyado a lo largo de su desarrollo.

A todos, los haya mencionado o no, gracias.

Índice de contenido

Índice de fotos

Índice de tablas

Capítulo 1

Introducción

El TFG tiene como objetivo el control del motor brushless **44x25 SI de Dunkermotoren** a través de sistemas embebido. Ambos elementos se encuentran en el Departamento de Ingeniería Eléctrica de la Escuela de Ingenierías Industriales en Valladolid. De esta manera ponemos en servicio ambos elementos para formar a los alumnos de distintas titulaciones en el manejo de LabView como sistema de control y adquisición de datos y en el conocimiento y uso de los motores brushless.

Con este proyecto trataremos de controlar el motor y aprovechar todas las funciones que ofrece.

Demostraremos la viabilidad de utilizar un equipo de National Instruments (NI), más un programa (software) hecho con LabView para ejecutar el control.

El objetivo principal del proyecto es realizar la programación LabView requerida para un control efectivo sobre el motor brushless a través de un PC. Para ello seguiremos los pasos descritos en el diagrama de flujos desglosados a continuación:

Los objetivos que buscamos alcanzar con el proyecto son:

- Simplicidad en el manejo.
- Control directo y en tiempo real sobre el motor por parte del usuario.
- Utilización de un PC por parte del usuario para interactuar con el sistema embebido.
- Pilotado completo del motor realizado exclusivamente a través del dispositivos de NI.
- Protección del equipo y las personas.
- Uso de WIFI como red inalámbrica más extendida evitando la infraestructura de cableados y facilitando la conexión además de facilitar el envío de datos en tiempo real.

A continuación se muestra un esquema que describe el método de control del motor:

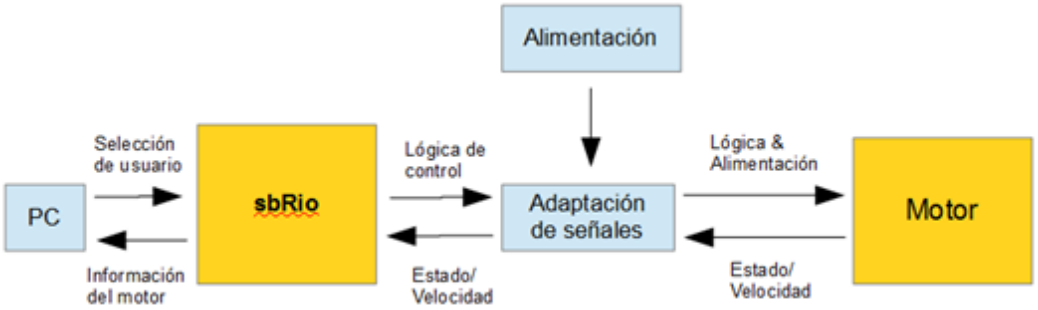


Figura 1.1. Esquema básico sobre lógica de control

Para hacer un control lo más rápido e intuitivo posible por parte del usuario no solo se utilizará el script visual de Windows a través de un PC sino que también se habilitarán una botonera y un display LCD integrados en la placa del sistema embebido.

El control del motor brushless se hará de manera íntegra mediante sistemas embebidos de manera que podamos aprovechar al máximo las propiedades de este sistema, que ofrece flexibilidad, fiabilidad y máximo rendimiento aparte de simplicidad gracias al método de programación LabView.

El motor que pondremos en servicio será el modelo 44x25 SI Dunkermotoren, el cual tiene una lógica interna que nos proporcionará una serie de entradas y salidas digitales y analógicas de manera que no solo nos facilitará el manejo del motor sino que también seremos capaces de recibir información del mismo como por ejemplo la velocidad.

Para controlar el motor, la comunicación entre el sistema embebido y el PC se hará efectiva a través de comunicación inalámbrica Wifi, de manera que el control del motor se realizará de forma remota sin necesidad de ubicar el PC en un lugar concreto. El PC se comunicará con el sistema embebido y éste a su vez con el motor mediante el sistema de programación LabView.

La conexión entre el sistema embebido y el motor se realizará a través de una placa electrónica que adaptará la tensión a los máximos y mínimos tolerables por ambos elementos.

Finalmente la potencia eléctrica requerida por los equipos proviene de una fuente de alimentación conectada a la red doméstica de 230V de

corriente alterna que será reducida y estabilizada a una corriente continua de 24V.

Las herramientas y componentes utilizados para este trabajo se justificarán a continuación:

- Uso de NI Labview (versión 2015): compatible con sbRio 9636 y myRio1900 como herramienta a través de la cual generar un software de control.

- Uso de un adaptador de señales: debido a la limitada tensión de las salidas digitales del sistema embebido, menor que la requerida por el motor. Se utilizaran transistores y fototransistores como interruptores que pilotarán la tensión de 24V que será proporcionada por la fuente de alimentación.

- Puesta en servicio del motor 44x25 SI de Dunkermotoren: adquirido por el Departamento de Ingeniería Eléctrica de la Escuela de Ingenierías Industriales para formar debidamente a los alumnos sobre la teoría y práctica de los motores brushless.

Capítulo 2

Motores de CC sin escobillas (Motores Brushless)

Los motores brushless son motores de corriente continua con imanes permanentes que suponen una ventaja frente a sus antecesores, los cuales para su control tienen un colector de delgas y unas escobillas. Estos elementos suponen un mayor volumen para la misma potencia. También rozan y producen calor, lo que supone una pérdida de rendimiento a parte de ruidos, vibraciones y la necesidad de un mayor y más periódico mantenimiento.

En los motores brushless se realiza el control de manera electrónica y son ampliamente utilizados por ejemplo en pequeños ventiladores o para impulsar maquetas radiocontrol. Aunque cada vez se fabrican motores con más potencia y con mayores aplicaciones gracias al avance de la electrónica, un buen ejemplo de ellos son los patinetes eléctricos.

2.1. Motores de corriente continua

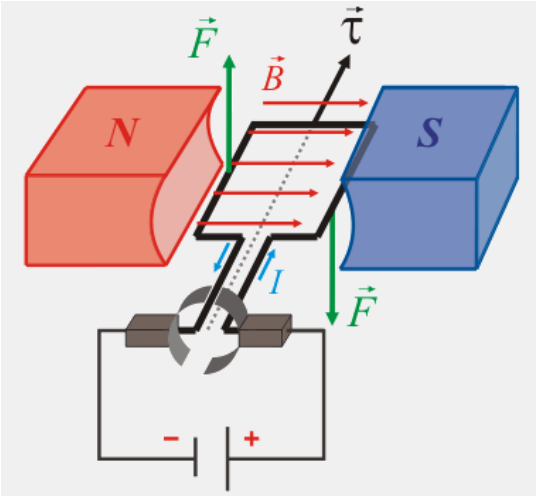


Figura 2.1. Simplificación visual de la fuerza ejercida sobre una única espira

Si hay una corriente recorriendo la bobina se crea una fuerza que tenderá a hacer rotar la espira. Esta fuerza aparece representada por la siguiente ecuación física:

$$\vec{F} = K(l\vec{i} \times \vec{B})$$

Donde: F=fuerza, K=constante, B=Campo magnético, l=longitud del conductor, i=intensidad que recorre el tramo de conductor expuesto al campo magnético, K=número de conductores (en caso de haber más espiras)

En los motores el conductor gira alrededor de un eje central, como se puede ver en la figura 2.1. de manera que el par se define con siguiente expresión:

$$\vec{T} = \vec{F} \wedge \vec{R}$$

Donde: T=Par de rotación, R=Radio hasta el entrehierro.

En la figura 2.1. se puede apreciar que la bobina se encuentra en una posición de par máximo y la posición de par mínimo se encuentra a 90 grados eléctricos de la situación que figura en la imagen. Durante esta rotación desde el par máximo al mínimo este cae de manera sinusoidal hasta llegar a cero.

Un motor eléctrico a su vez se compone de una gran cantidad de espiras que componen el estator/rotor, de esta manera el par motor siempre es constante, ya que en el momento que una de las espiras entra en un momento de mínimo par otra espira entrará en un estado de máximo par.

En un motor de escobillas cada bobina se encuentra angularmente desplazada, de manera que cuando una ha perdido el par motor, la intensidad se conmuta a otra bobina que esté bien posicionada para seguir aportando el par motor máximo. Esta conmutación es la realizada a través de las escobillas, las cuales rozan sobre el colector de delgas como se indica en la siguiente figura.

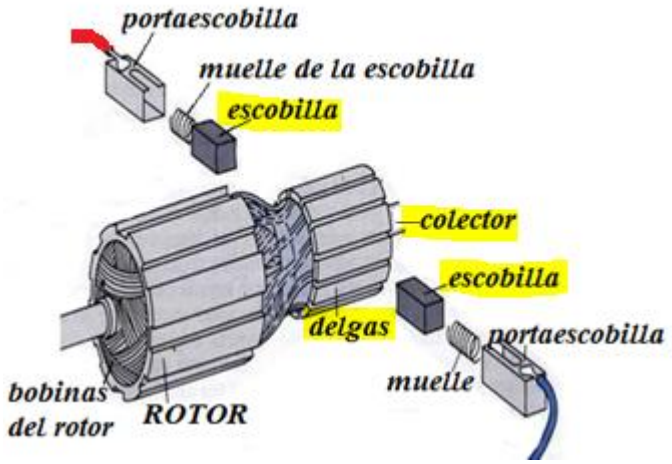


Figura 2.2. Rotor de motor de CC con colector y escobillas

2.2. MotoresBrushless (sin escobillas)

Un motor brushless se diferencia de los tradicionales descritos anteriormente en el método de conmutación.

La posición de las bobinas (fases) se detecta electrónicamente, por ejemplo mediante sensores Hall y a continuación la corriente es conmutada a la bobina correspondiente también electrónicamente, a través de Mosfets trabajando en conmutación o transistores en modo saturación-corte.

Un motor con escobillas se puede transformar en un motor brushless retirando el colector de delgas y sustituyéndolo por transistores, pero debido a toda la cantidad de transistores que serían necesarios se utilizan bobinados polifásicos al igual que los motores de corriente alterna.

Las configuraciones más habituales están entre 2, 3 o 4 bobinados, a criterio fabricante. La gran mayoría de fabricantes suelen usar tres bobinados aunque esto requiera de más transistores y otros componentes que los de dos bobinados, y el control sobre estos tres bobinados se suelen hacer de dos maneras: unipolar y bipolar.

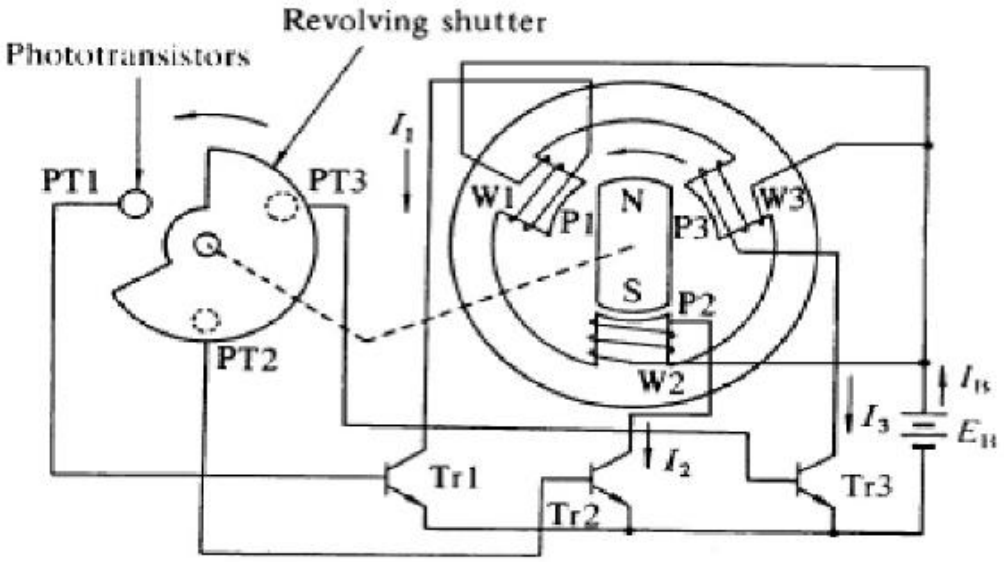


Figura 2.3. Configuración unipolar

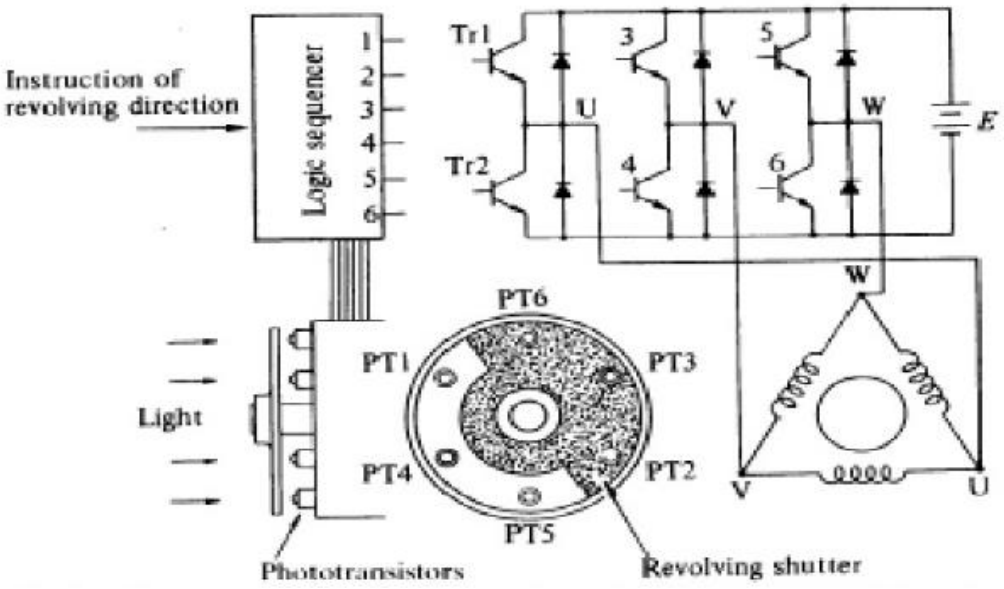


Figura 2.4. Configuración bipolar

A continuación, en la figura 2.5. se describirá la conmutación para la configuración más común de motor brushless, la cualutiliza tres hilos que generan 3 bobinas independientes. El rotor será siempre el que tenga el campo magnético permanente mientras que el estator será el que disponga de los bobinados por los que circulará la corriente.

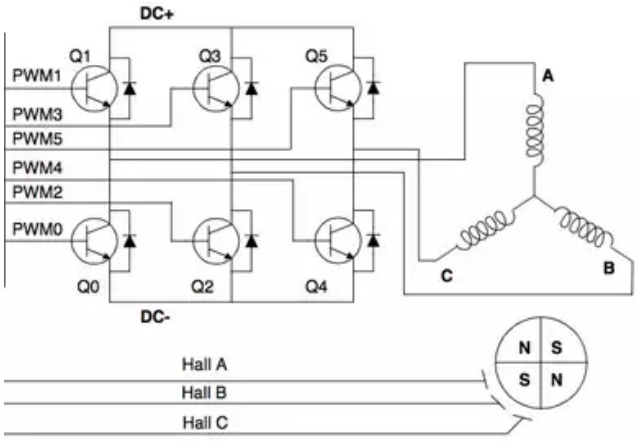


Figura 2.5. Pasos secuenciales de la conmutación

Cerrando los interruptores Q1 y Q2 se cerrará la corriente a través de los bobinados A-B. El rotor girará hasta la posición de par cero. Si el transistor Q1 se abre y el Q5 se cierra entonces la corriente recorrerá los bobinados C-B, recorriendo el rotor 120º eléctricos. De igual manera si el Q2 se abre y el Q0 se cierra circulará corriente por C-A. De esta última manera la corriente que pasa por la bobina A lo hará en dirección contraria a la que lo hacía en la primera operación y habríamos completado una vuelta de rotor.

A continuación abriendo Q5 y cerrando Q3 giraríamos otros 120º haciendo circular la corriente en la dirección B-A. Después cerrando Q0 y abriendo Q4 haremos que la corriente pase por B-C y finalmente abriendo Q3 y cerrando Q1 recorro los últimos 120º. Si tomásemos este punto como el inicial, paso previo a el paso de corriente cerrando Q1 y Q2 nos damos cuenta que habríamos completado un ciclo completo de conmutación, el cual abarca 720º eléctricos, es decir, dos vueltas del rotor en sentido horario.

Llegados a este punto vemos necesario llevar un orden correcto a la hora de ejercer esta conmutación por lo que es necesaria una lógica de control que pilote los transistores. Lo más utilizado para dar información sobre la posición del rotor son los sensores Hall, de esta manera la lógica de control toma una referencia para pilotar la apertura de dichos

transistores. La gráfica 2.6. nos ayudará a comprender este funcionamiento.

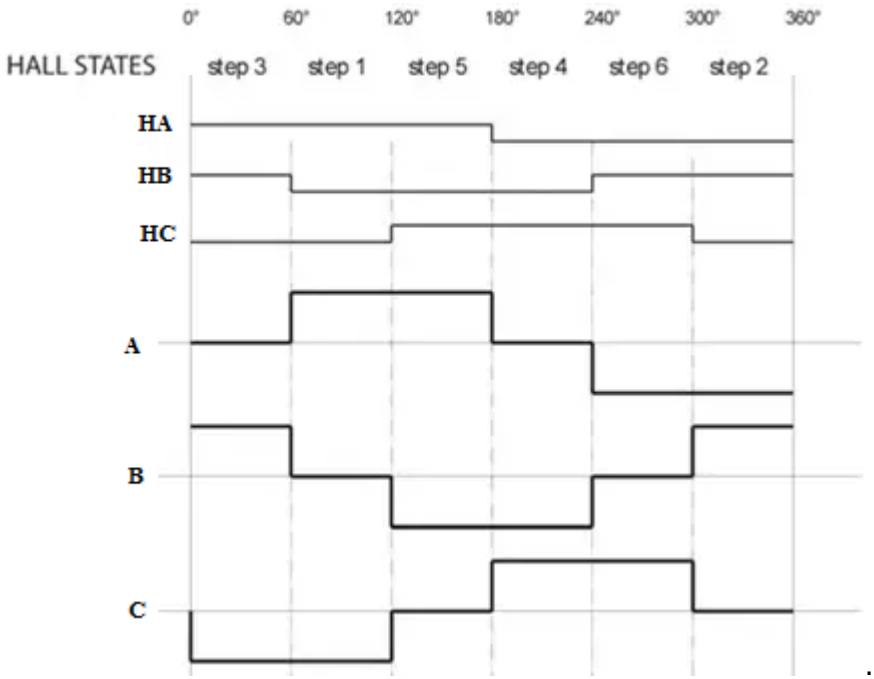


Figura 2.6. Método de control mediante sensores HALL

Vemos que en función de la polarización vista por los sensores Hall manejamos la conmutación de las bobinas. Por ejemplo, en el caso de un giro en sentido horario, el sensor HB al detectar un polo S abrirá la bobina A (Q1) y cerrará la B (Q3). A continuación (60º eléctricos después) HC detectará un polo N y cortará el paso de corriente a través de la bobina C abriendo el interruptor Q4 y lo habilitará a través de B cerrando Q2, etc… De esta manera cada 60º ejerceremos una conmutación aprovechando el par máximo ejercido por el motor.

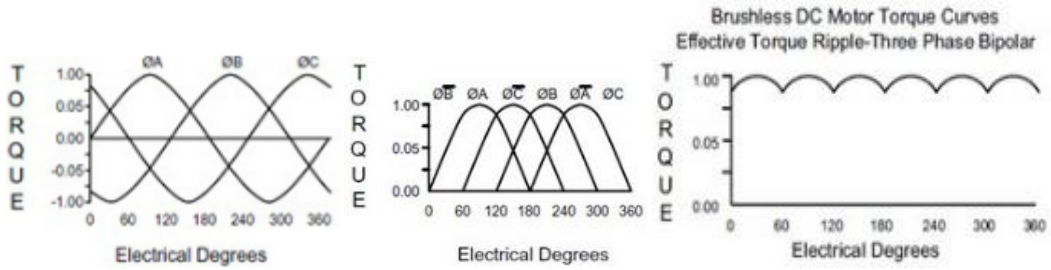


Figura 2.7. Izq. A derecha: Curvas de par de un motor brushless, curvas con las fases conmutadas para par positivo y curvas de par efectivo con dos fases bipolares.

Gracias a la electrónica de control somos capaces de regular la tensión aportada al motor y la intensidad, de esta manera ejercemos un control total sobre el motor.

La regulación del voltaje nos permite variar la velocidad de rotación, e incluso invertir el sentido de giro (tensión negativa) y el control de la intensidad nos permite variar el par aportado por el motor siendo capaces incluso de frenar invirtiendo la intensidad aportada (intensidad negativa).

Además, gracias a el control de posición angular aportado por los sensores Hall seríamos capaces de pararnos en una posición deseada y mantenerla venciendo fuerzas externas o la propia inercia de la carga aportando el par deseado.

En un motor brushless el rotor será siempre el que disponga de los imanes permanentes y el estator el elemento que está bobinado. En base a esto, hay dos tipos de configuración mecánica: una con rotor externo y otra con rotor interno. Cada una con sus ventajas y sus virtudes.

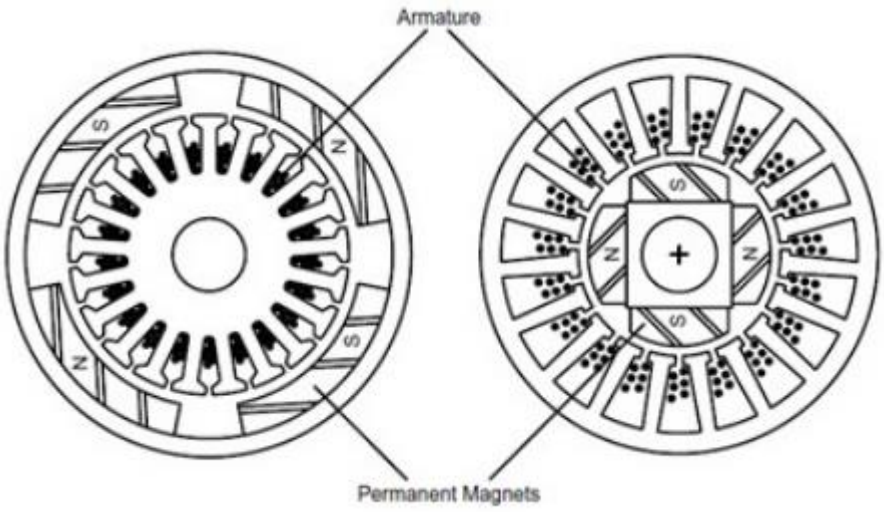


Figura 2.8. De izq. A derecha: rotor externo y rotor interno.

	Rotor externo	**Rotor interno**
Inercia	Mayor	Menor

Par/potencia	Mayor	Menor
Componentes HALL	Más	Menos
Posicionamiento	Aproximado	Preciso

Tabla 2.1. Comparación rotor interno-externo

2.3. Motor brushless BG 44x25 SI de Dunkermotoren

El motor que nos concierne es el Dunkermotoren modelo BG 44x25 SI al cual le hemos añadido un mecanismo reductor de relación 4 a 1. Se trata de un motor de corriente continua, de 3 fases y 4 polos de imanes permanentes. Contiene una electrónica de control de velocidad interna para la conmutación de los bobinados.

La velocidad deseada de establece a través de una señal analógica de entrada que abarca de 0V a +10V. El motor posee 4 modos de operación:

- Rotación en sentido horario.
- Rotación en sentido antihorario.
- Parada (motor apagado).
- Frenada, manteniendo un par de bloqueo.

Estos distintos modos de operación se establecen a través de una combinación de dos señales digitales.

Existen otras dos señales digitales de entrada que proporcionan un control sobre la velocidad del motor, estos modos serán los siguientes:

- Velocidad controlada independientemente del par requerido.
- Velocidad no controlada y dependiente del par requerido.
- Velocidades memorizadas (speed1 y speed2).

El motor también tiene integrado un modo de enseñanza para memorizar otras velocidades y rampas de velocidad entre dichas velocidades.

También existen dos salidas digitales del motor, una será la señal de posición indicada por los sensores de efecto Hall que aportarán 6 pulsos por vuelta y la otra indicará el fallo del motor en caso de producirse.

Rango de velocidad máxima del motor	0-5000 rpm
Rango de velocidad ajustable	150-4096 rpm
Mínimo voltaje del motor	20 V DC
Máximo voltaje del motor	30 V DC
Máximo rizado en el voltaje de alimentación	5%
Corte (fallo) por subvoltaje	< 19V
Límite de destrucción por sobrevoltaje	> 35 V
Fusible externo requerido	8 AT
Protección por sobretemperatura	> 100ºC
Máximo pico de corriente (motor)	9 A
Rango de temperatura motor	-20ºC a 100ºC
Temperatura ambiental recomendada	0ºC a 50ºC
Humedad relativa	Max: 90%
Clase de protección	IP50
Conector 12 vías	Conector circular (DIN 45326), Binder, 723.

Tabla 2.2. Datos eléctricos y mecánicos del motor BG 44x25 SI

Dimensiones del motor:

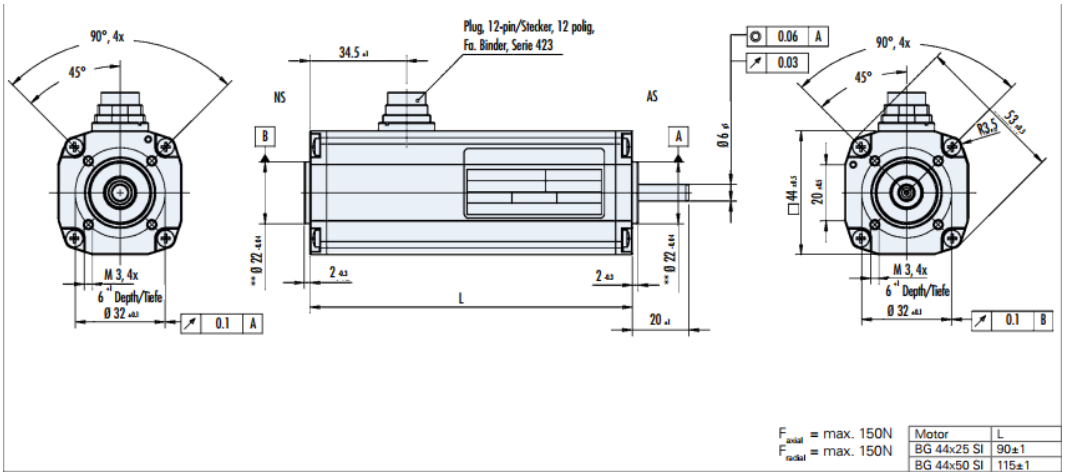


Figura 2.9. Dimensiones del motor BG 44x25 SI

Características del motor:

Potencia nominal	20W
Par Nominal	6(7.2) Ncm
Velocidad nominal	3200rpm
Tensión de alimentación nominal	24V
Intensidad nominal	1.54A
Longitud	90mm
Peso	470g

Tabla 2.3. Características del motor BG 44x25 SI.

Diagrama característico del motor:

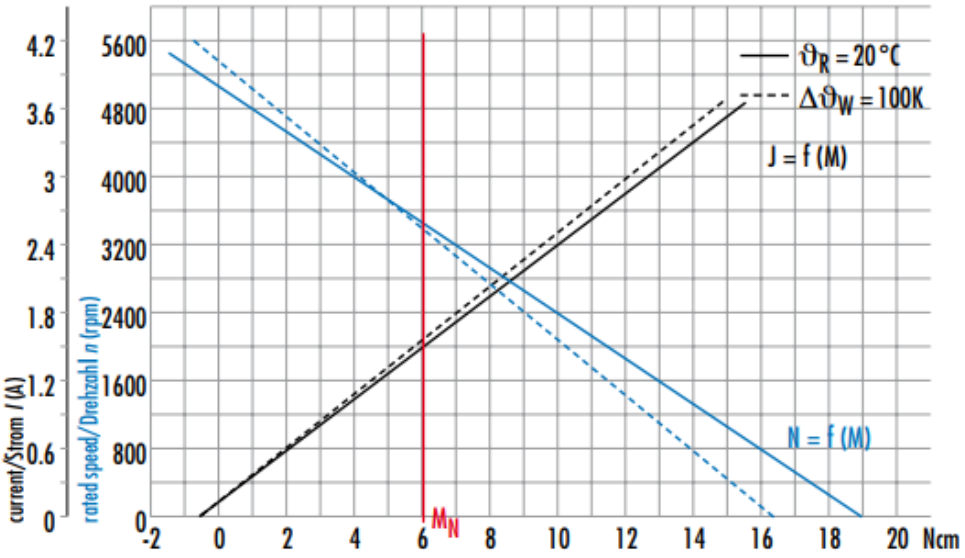


Figura 2.10. Diagrama característico del motor

En rojo se puede apreciar el par nominal, el eje X representa el par. En azul se observa la velocidad respecto al par y en negro el consumo de amperios respecto al par.

A continuación representamos el conector circular de 12 vías según la norma DIN 45326, éste tendrá la siguiente distribución:

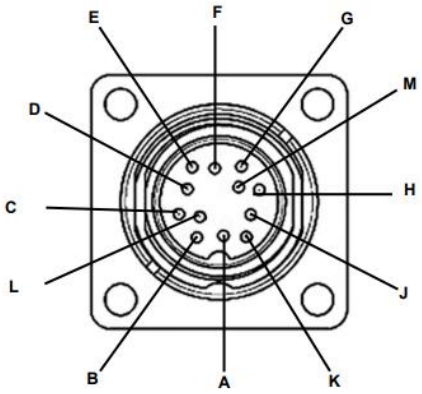


Figura 2.11. Conector circular Blinder, serie 723.

Correspondencia de los pines con respecto a las señales:

Pin	Conexión	Función	Color
E+F	U_e	+24 V alimentación del motor	Rojo
D	IN4	Control de velocidad	Verde
M+G	Gnd	0V alimentación motor	Negro
B	IN1	Control estado	Amarillo
C	IN2	Control estado	Azul
J	AI+	Entrada analógica 0…+10V	Rosa
H	AL-	Referencia de AL+ (0V)	Morado
A	OUT1	Pulsos de los sensores HALL (6 por vuelta)	Naranja
K	OUT2	Error: (‘1’= NO falta, ‘0’=falta)	Blanco
L	IN3	Control de estado (velocidad)	Marrón

Tabla 2.4. Correspondencia de pines con las señales transferidas y color de cable

El nivel de tensión de la lógica para que el motor reconozca un ‘0’ digital es de 0-6V positivos, y el nivel de tensión para que reconozca un ‘1’ es de 7-24V positivos.

Para controlar el motor utilizaremos las secuencias lógicas de las entradas digitales descritas anteriormenteutilizando la siguiente tabla:

IN1	IN2	IN3	IN4	Función
0	0	X	X	Control no activo, ni par motor ni par de frenada
1	0	X	X	Rotación en sentido horario
0	1	X	X	Rotación en sentido anti horario
1	1	X	X	Frenada manteniendo un par de frenada (STOP)
X	X	0	0	Velocidad regulada (150…4096rpm) con +Al y -Al
X	X	0	1	Velocidad no regulada (0…5000rom) con +Al y -Al
X	X	1	0	Velocidad memorizada 1 (speed1) 200rpm
X	X	1	1	Velocidad memorizada 2 (speed2) 2500rpm

Tabla 2.5. Lógica de control del motor, señales digitales.

La enseñanza de las velocidades y las rampas de aceleración no podremos llevarlas a cabo debido a la limitación de nuestro sistema embebido, el cual solo nos permite tres salidas digitales.

De esta manera nos queda imposibilitado el uso del ‘teach mode’ y de la entrada digital al motor IN3, la cual quedará referenciada a masa, imposibilitando también el uso de las velocidades memorizadas.

Por último la señal de fallo que da la salida digital OUT3 (‘1’ no fallo, ‘0’ fallo) puede deberse a los siguientes factores:

Tipo de fallo	Tipo de detección	Rango
Sobretemperatura	Software	95ºC off; 70ºC on
Bajo Voltaje	Software	16.5V off; 18.5V on
Sobrecorriente	Hardware	9A (10 seg)
Bloqueo en RPM controlada	Software	

Tabla 2.6. Factores que activan el fallo del motor.

Capítulo 3

Sistemas Embebidos (SE)

Un sistema embebido (SE), es un conjunto de elementos electrónicos, informáticos e incluso mecánicos (relés) que en conjunto forman un sistema de computación diseñado para realizar una o varias tareas específicas frecuentemente en tiempo real, hecho que los diferencia de los ordenadores de propósito general. Destaca su escaso tamaño, su gran fiabilidad, mantenibilidad y disponibilidad, su bajo consumo y su bajo coste.

Un sistema en **tiempo real**depende de 2 cosas a la vez. De realizar su tarea correctamente y hacerlo en un tiempo finito previamente especificado. Los sistemas de tiempo real se pueden caracterizar como blandos o duros. Si un sistema de tiempo real blando no cumple con sus restricciones de tiempo, simplemente se degrada el rendimiento del sistema, pero si el sistema es de tiempo real duro y no cumple con sus restricciones de tiempo, el sistema fallará. Este fallo puede tener posiblemente consecuencias catastróficas.

La mayoría de componentes se suelen situar en su placa base (tarjetas de video, audio, modem, etc…). Algunos ejemplos básicos son un taxímetro, la electrónica que controla una fotocopiadora o una máquina expendedora…

Los sistemas normalmente se encuentran conectados a ambientes físicos mediante sensores y actuadores (poseen partes analógicas y digitales) y suelen ser sistemas reactivos, que son “aquellos que están en interacción continua con su entorno y su ejecución es a un ritmo determinado por ese entorno” (Bergé, 1995).

Los lenguajes más comúnmente usados son ensamblador (programados directamente en el microprocesador) o compiladores específicos en C, C^{++} o JAVA. Este último en casos en los que no se trate de aplicaciones en tiempo real, es decir, cuando la respuesta del sistema no sea un factor crítico.

Un sistema embebido podrá ser considerado como tal cuando esté compuesto por los siguientes componentes:

- Generador de señales de reloj, por ejemplo, un resonador de cuarzo.
- Memoria RAM para almacenamiento de datos.
- ROM o sus variantes (PROM, EPROM o FLASH) para el almacenamiento del programa.
- Interfaces inputs/outputs, puertos en serie/paralelos.

- Otros periféricos como contadores, temporizadores, ADC, etc…
- Una CPU (memoria central de procesamiento) que interpreta una a una todas las instrucciones del programa

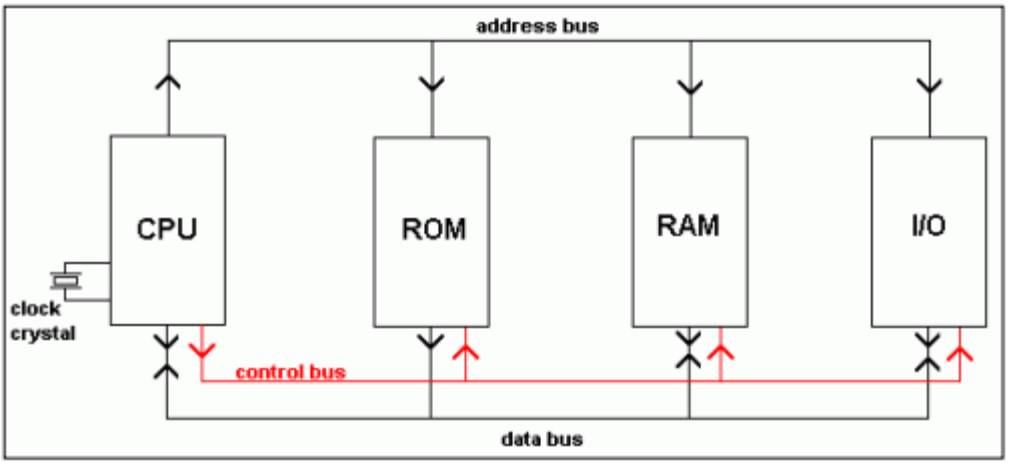


Figura 3.1. Configuración mínima de un sistema embebido.

Llegados a este punto hemos descrito la configuración básica de un sistema embebido como la de un microcontrolador (foto 1.4.B). Un sistema embebido como ya hemos mencionado es un elemento cuyo hardware y software están diseñados para una tarea específica y su cerebro será siempre un microprocesador elegido de manera específica y al que le acompañarán circuitos exteriores y periféricos también específicos.

3.1. Aplicaciones de los sistemas embebidos.

Para entender la importancia de estos sistemas, tenemos que hacernos una idea general no solo de su definición, sino también de su potencial y su alcance.

Últimamente debido a la amplia aplicabilidad de los sistemas embebidos en distintos ámbitos sectoriales y al valor añadido que estos suponen para el producto de aplicación se está convirtiendo en un elemento estratégico para la mayoría de empresas tecnológicas que buscan un aumento de su competitividad. Algunos ejemplos que ilustran las posibilidades de estos sistemas son los siguientes:

- Control en procesos de montaje y producción a través de un procesador y distintos interfaces hombre-máquina para gestionar una o varias máquinas (sean motores, hornos, etc…) que se encarguen de realizar una o varias tareas al mismo tiempo que informa del estado del proceso.
- Puntos de venta como cajas de supermercado donde se requieren numerosos puertos de entrada y salida porque se integran teclados numéricos, lectores de barras y tarjetas bancarias, pantallas con displays, etc… y debe de ser robusto para soportar operaciones continuadas.
- Puntos de información al ciudadano o de servicio como por ejemplo oficinas de turismo, grandes almacenes o bibliotecas que en muchos casos requieren de pantallas táctiles, respuestas personalizadas y entornos gráficos intuitivos.
- Decodificadores y set-top boxes para la recepción de televisión que se encargan de realizar un proceso de decodificación de manera que solo puedan tener acceso a ella los usuarios abonados.
- Sistemas de radar en aviones. El procesado de la señal reflejada por el radar embarcado en un avión requiere de una alta potencia de cálculo, ocupar y pesar poco y aguantar con mucha fiabilidad y robustez condiciones extremas de funcionamiento.
- Equipos de medicina en hospitales y uvi móvil.
- Cajeros automáticos
- Electrónica interna del vehículo, plagada de sistemas embebidos que controlan sistemas complejos del vehículo como es la inyección, el ABS o los airbags que a su vez están comunicados vía CAN(protocolo de comunicación para multiplexado)entre ellos formando un sistema cerrado y sólido. Más abajo describiremos algunos ejemplos.

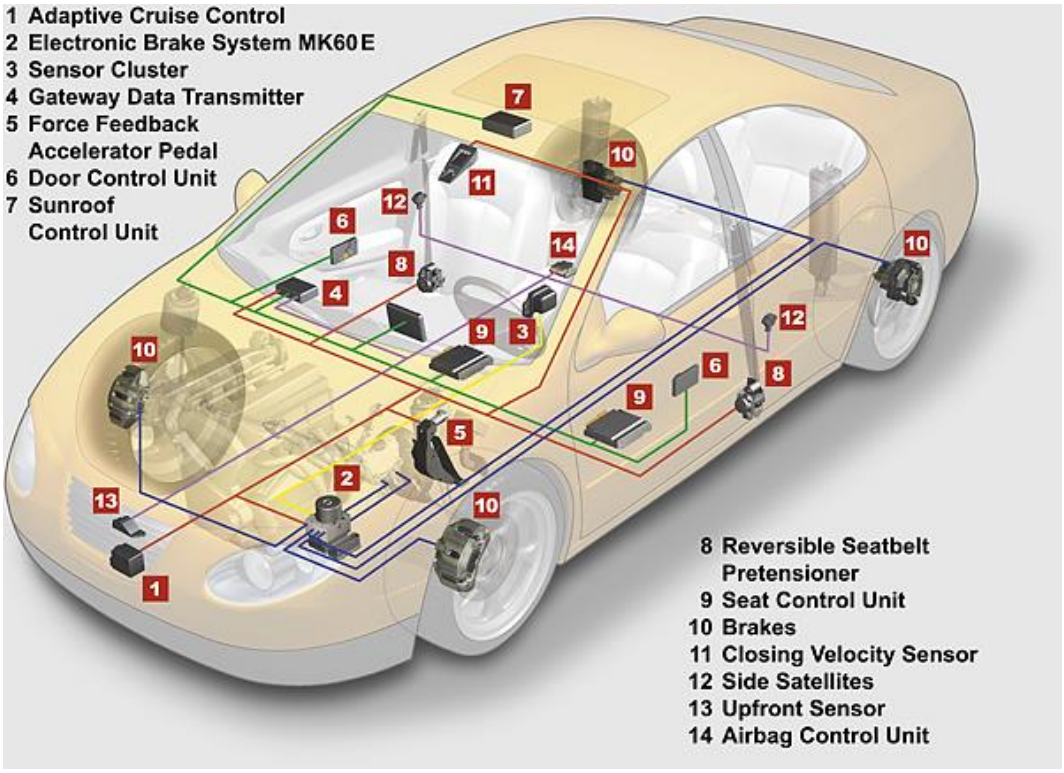


Figura 3.2. Algunos ejemplos de sistemas embebidos en automoción.

1- Control adaptativo (radar de proximidad).

2- Sistema electrónico de frenado (ABS).

4- Transmisor de datos (TCU).

6- Centralita de control de puertas.

7- Centralita para el control del techo solar.

9- Centralita de control de asientos.

14- Calculador Airbag.

En base a sus distintas aplicaciones de los SE podemos deducir una serie de características básicas y requerimientos de estos sistemas:

- Deben ser confiables. La confiabilidad (reliability), R(t) que es la probabilidad de que el sistema trabaje correctamente dado que está funcionando en t=0.
- También deben de ser mantenibles. La mantenibilidad (Maintainability) M(d) es la probabilidad de que el sistema vuelva a trabajar correctamente d unidades de tiempo después de un fallo.

- Y disponibles. La disponibilidad (Availability), A(t) es la probabilidad de que el sistema esté funcionando en el tiempo t.
- La seguridad informática, dependiendo de la aplicación, suele ser una prioridad. Consiste en disponer de una comunicación confidencial y autentificada.
- Deben ser eficientes en cuanto a la energía, al tamaño de código, al peso y al costo.
- Están dedicados a ciertas aplicaciones específicas
- Interfaces de usuario dedicadas (no necesariamente con ratón, teclado o pantalla).

3.2. Componentes comunes de un sistema embebido

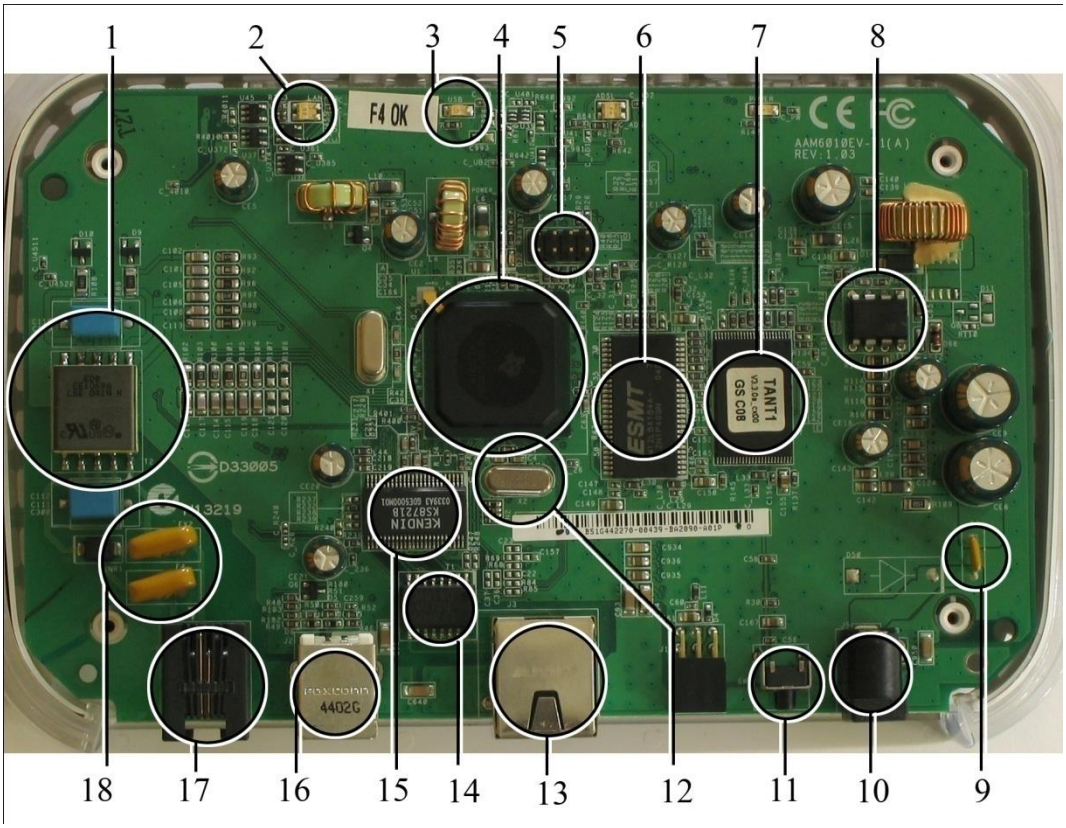


Figura 3.3. Componentes de un sistema embebido

Enel caso de la imagen anterior se trata de un router. Los elementos básicos serán descritos a continuación.

3.2.2. Microprocesador (Nº4)

Este elemento es el encargado de realizar las operaciones de cálculo principales del sistema. Ejecuta el código para realizar una determinada tarea y dirige el funcionamiento de los demás elementos que le rodean. Un microprocesador es una implementación en forma de circuito integrado (IC) de la Unidad Central de Proceso CPU de una computadora. Frecuentemente nos referimos a un microprocesador como simplemente “CPU”, y la parte de un sistema que contiene al microprocesador se denomina subsistema de CPU. Los subsistemas de entrada/salida y memoria pueden ser combinados con un subsistema de CPU para formar una computadora o sistema embebido completo. Estos subsistemas se interconectan mediante los buses de sistema (formados a su vez por el bus de control, el bus de direcciones y el bus de datos).

La CPU, así mismo, se puede dividir en dos zonas diferenciadas. La zona de operaciones está compuesta fundamentalmente por los registros y las ALU y en ella se ejecutan las operaciones que es capaz de realizar la CPU. La zona de control sin embargo está coordina todo el funcionamiento de la CPU, enviando micro-órdenes a los elementos que deben intervenir en cada momento.

3.2.3. La memoria

En la memoria se encuentra almacenado el código de los programas que el sistema puede ejecutar así como los datos. Su característica principal es que debe tener un acceso de lectura y escritura lo más rápido posible para que el microprocesador no pierda tiempo en tareas que no son meramente de cálculo. Al ser volátil el sistema requiere de un soporte donde se almacenen los datos aun sin disponer de alimentación o energía.

La**caché** es una memoria más rápida que la principal en la que se almacenan los datos y el código accedido últimamente. Dado que el sistema realiza microtareas, muchas veces repetitivas, la caché hace ahorrar tiempo ya que no hará falta ir a memoria principal si el dato o la instrucción ya se encuentra en la caché. Dado su alto precio tiene un tamaño muy inferior (8-512 KB) con respecto a la principal (8-256 MB). En el interior del chip del microprocesador se encuentra una pequeña caché (L1), pero normalmente se tiene una mayor en otro chip de la placa madre (L2).

El **disco duro (Nº7)**, la memoria no volátil, suele estar implementada en los sistemas embebidos con unidades de estado sólido. Mucho menos voluminosas y menos sensibles a condiciones mecánicas exigentes. Como por ejemplo entornos con grandes vibraciones.

BIOS-ROM(Basic Input & Output System, sistema básico de entrada y salida) es código que es necesario para inicializar la computadora y para poner en comunicación los distintos elementos de la placa madre. La ROM (Read Only Memory, memoria de sólo lectura no volátil) es un chip donde se encuentra el código BIOS.

Dentro de las memorias ROM hay dos tipos principales. Las PROM o simplemente ROM son memorias que solo se pueden programar una sola vez por el programador o vienen ya programadas de fábrica (ROM de máscara). Las EPROM son reprogramables y puede borrarse su programación exponiendo el silicio a una potente luz ultravioleta más adelante serían sustituidas por las EEPROM borrables eléctricamente y más adelante por las FLASH, que con las mismas características incrementan enormemente la cantidad de memoria.

CMOS-RAM (Nº6) Es un chip de memoria de lectura y escritura alimentado con una pila donde se almacena el tipo y ubicación de los dispositivos conectados a la placa madre (disco duro, puertos de entrada

y salida, etc.). Además contiene un reloj en permanente funcionamiento que ofrece al sistema la fecha y la hora.

3.2.4. Contadores y temporizadores

Los temporizadores son unos de los periféricos más habituales en los microcontroladores y se utilizan para muchas tareas, como por ejemplo, la medición de frecuencia e implementación de relojes para el trabajo de conjunto con otros periféricos que requieren una base estable de tiempo entre otras funcionalidades.

3.2.5. El Chip Set (Nº12).

Se encarga de controlar las interrupciones dirigidas al microprocesador, el acceso directo a memoria (DMA) y al bus ISA, además de ofrecer temporizadores, etc. Es usual encontrar la CMOS-RAM y el reloj de tiempo real en el interior del Chip Set.

3.2.6. Lospuertos de entrada/salida (I/O)

Son puntos (nodos) en los que los dispositivos periféricos se pueden conectar y pueden intercambiar información con el sistema embebido. Los puertos contienen en sí mismos un número definido de registros, los cuales se utilizan para el almacenamiento temporal de varios tipos de datos. Las direcciones de los registros y sus funciones están definidas con precisión (standard). El subsistema de entrada acepta datos del exterior para ser procesados mientras que el subsistema de salida transfiere los resultados hacia el exterior. Lo más habitual es que haya varios subsistemas de entrada y varios de salida. A estos subsistemas se les reconoce habitualmente como periféricos de E/S. Hay tres tipos de puertos:

1. Puertos serie: en los que se transfieren datos bit a bit de manera secuencial. (COM1, COM2).
2. Puertos paralelos: en los que los datos se transfieren en paralelo (un byte, una palabra, etc.) (LPT1).
3. Puertos universales (USB).

3.3. Desarrollo del FPGA

Un componente aplicable a los sistemas embebidos es el FPGA.

Las FPGAs son unos dispositivos nos permiten describir un circuito digital usando un lenguaje específico (los dos más comunes son VHDL y Verilog) y que tras cargarlo en el integrado, es creado 'físicamente' en el chip. Su nombre es un acrónimo inglés que significa matriz de puertas reprogramable o '*Field Programmable Gate Array'.*

Internamente se componen principalmente de cables, puertas lógicas, biestables, y puertos de entrada y salida. Todo ello sin conectar, como una plantilla en blanco, hasta que se les carga un computo digital en software o en su defecto un bitstream, que consiste en un archivo de configuración que contiene información de cómo deben de conectarse los componentes.

La gran baza de los FPGA es que son elementos reconfigurables, lo que hace que puedan ser cambiados en el momento en el que se requiera. Antes esto no era una tarea sencilla pero los lenguajes de alto diseño son capaces de simplificar su reprogramación.

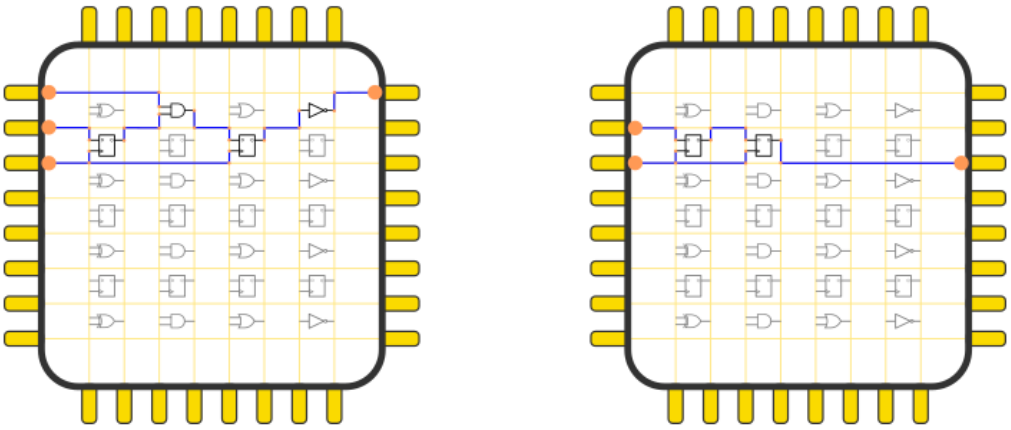


Figura 3.4. Estructura interna de un FPGA.

3.3.2. Ventajas de los FPGA

- Rendimiento: Aprovechando del paralelismo del hardware, los FPGAs exceden la potencia de cómputo de los procesadores digitales de señales (DSPs) rompiendo el paradigma de ejecución secuencial y logrando más en cada ciclo de reloj. BDTI. Esto es así porque el controlar entradas y salidas (E/S) a nivel de hardware ofrece tiempos de respuesta más veloces y funcionalidad especializada que coincide con los requerimientos de una aplicación.

- Tiempo en llegar al mercado: la tecnología FPGA ofrece flexibilidad y capacidades de rápido desarrollo de prototipos para enfrentar los retos de que un producto se libere tarde al mercado. Todo ello facilitado por loslenguajes de alto nivel que permiten una programación rápida e intuitiva por parte del usuario.
- Precio: La fuerte inversión inicial de otros sistemas es fácilmente justificable para los fabricantes de equipos originales que embarcan miles de chips por año, pero muchos usuarios finales necesitan la funcionalidad de un hardware personalizado para decenas o cientos de sistemas en desarrollo. La misma naturaleza programable del silicio implica que no hay precio de fabricación o largo plazos de ejecución de ensamblado.
- Fiabilidad: el núcleo de un procesador sólo puede ejecutar una instrucción a la vez, y los sistemas basados en procesadores están siempre en riesgo de que sus tareas se obstruyan entre sí. Los FPGAs, que no necesitan sistemas operativos, minimizan los retos de fiabilidad con ejecución paralela y hardware preciso dedicado a cada tarea.
- Mantenimiento a largo plazo: los chips FPGA, al ser reconfigurables, son capaces de mantenerse al tanto con modificaciones a futuro que pudieran ser necesarias. Mientras el producto o sistema se va desarrollando, se puede implementar mejoras funcionales sin la necesidad de invertir tiempo rediseñando el hardware o modificando el diseño de la tarjeta.

3.3.3. Combinación con un procesador

Una solución fiable y en auge es la combinación del elemento FPGA, que realizará múltiples tareas (procesando y calculando en paralelo las señales recibidas o enviadas) de manera fiable y robusta en un solo ciclo, con un procesador de tiempo real realizará la gestión de los datos.

El procesador comunicará los input y los outputs calculados en el FPGA al usuario a través de displays o de manera indirecta a través de por ejemplo un ordenador.

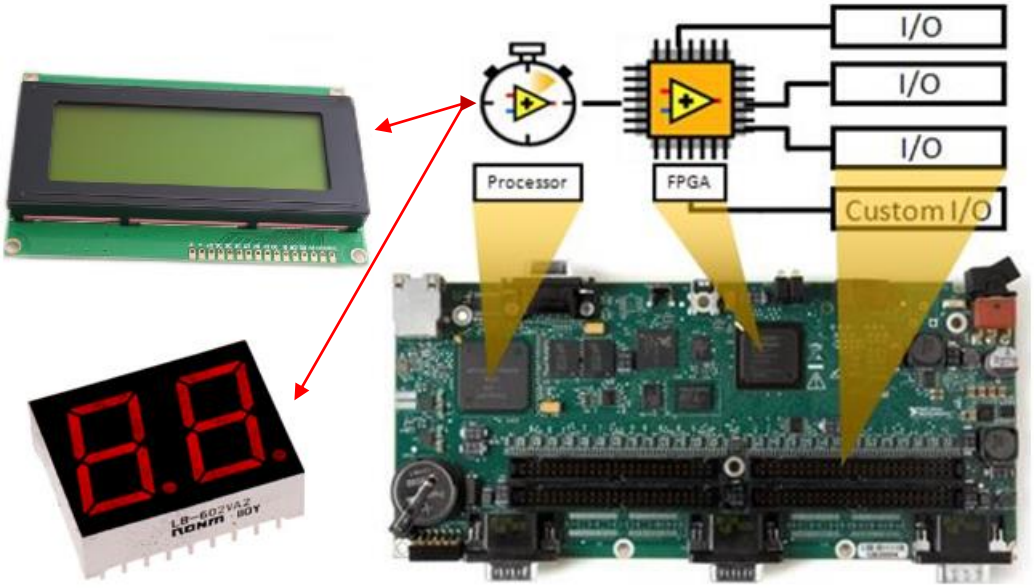


Figura 3.5. Configuración general de un sistema embebido con FPGA

3.4. NI sbRIO-9636

En nuestro caso estudiaremos el sistema embebido de National Instruments (NI) sbRIO-9636.

La arquitectura de este sistema incluye un procesador de punto flotante (floating point processor), que hace funcionar un sistema operativo en tiempo real (RTOS), un sistema FPGA y un conjunto de entradas y salidas que pueden ser reprogramadas utilizando LabView.

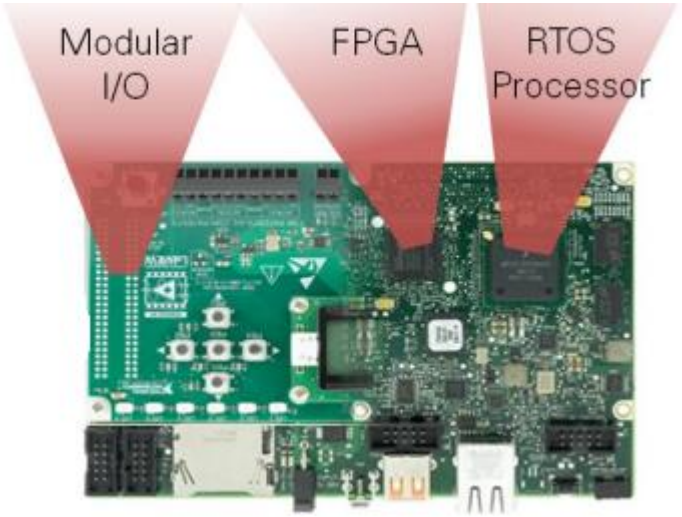


Figura 3.6. Arquitectura interna del SbRio-9636

El **punto flotante** es una forma de notación científica usada en los microprocesadores con la cual se pueden representar números racionales extremadamente grandes y pequeños de una manera muy eficiente y compacta, y con la que se pueden realizar operaciones aritméticas.Esta es más moderna y resulta suficientemente exacta y rápida para la mayoría de las aplicaciones. Es muy frecuentemente utilizada para lograr una buena aproximación del número que se desea representar, pero a menudo requiere de un “redondeo”, debido a su limitada precisión. Su representación involucra un número entero (la mantissa ‘m’) multiplicado por una base (‘b’) elevado a un exponente (‘e’), de tal forma que cualquier número de punto flotante a puede ser representado como:

$$a = m.b^{e}$$

Es posible especificar cuántos dígitos de precisión se requieren, asignando un valor al parámetro ‘P’.

La gran ventaja de esta arquitectura reside en que permite la representación de un rango de magnitudes mucho más amplio que el de la arquitectura de punto fijo. De acuerdo a la cantidad de bits utilizados para almacenar un determinado número de punto flotante, decimos que éste es de precisión simple (32 bits) o precisión doble (64 bits). En el caso de precisión simple, típicamente se le asignan a la mantisa los 23 bits menos significativos (bit 0 a bit 22), luego el exponente ocupa los siguientes 8 bits (bit 23 a bit 30) y el bit 31 está destinado a indicar el signo (0 = positivo, 1= negativo).

Los componentes anteriormente descritos otorgan a nuestro sistema embebido gran precisión en los datos procesados,una respuesta del sistema en tiempo real y gran capacidad de procesamiento de datos en paralelo gracias al FPGA. Lo que convierte a este elemento en una herramienta de adquisición de datos muy fiable.

Resumen de las especificaciones:

General	
Form Factor	Single-Board RIO
Compatible con RoHS	Sí
Certificaciones de Productos	RoHS UL - Product Safety
Entrada Analógica	
Número de Canales	16
Resolución de Entrada Analógica	16 bits
Muestreo Simultáneo	No
Rango de Voltaje Máximo	-10 V - 10 V
Salida Analógica	
Número de Canales	4
Resolución	16 bits
Rango de Voltaje Máximo	-10 V - 10 V
E / S Digital	
Canales Bidireccionales	28
Niveles Lógicos	3.3 V
FPGA Reconfigurable	
FPGA	Spartan-6 LX45
Controlador Reconfigurable	
Memoria No Volátil	512 MB
Memoria del Sistema	256 MB
Procesador	PowerPC de 400 MHz
Eléctrico	
Rango de Voltaje de Salida (Fuente Externa)	9 V - 30 V
Especificaciones Físicas	
Longitud	15.4 cm
Ancho	10.3 cm
Conector de E / S	IDC de 50 pines
Temperatura de Operación	-40 °C - 85 °C

Figura 3.7. Especificaciones del SbRio-9636

Nuestro sistema embebido constará de las siguientes partes, cada parte con los componentes que figuran en la tabla a continuación:

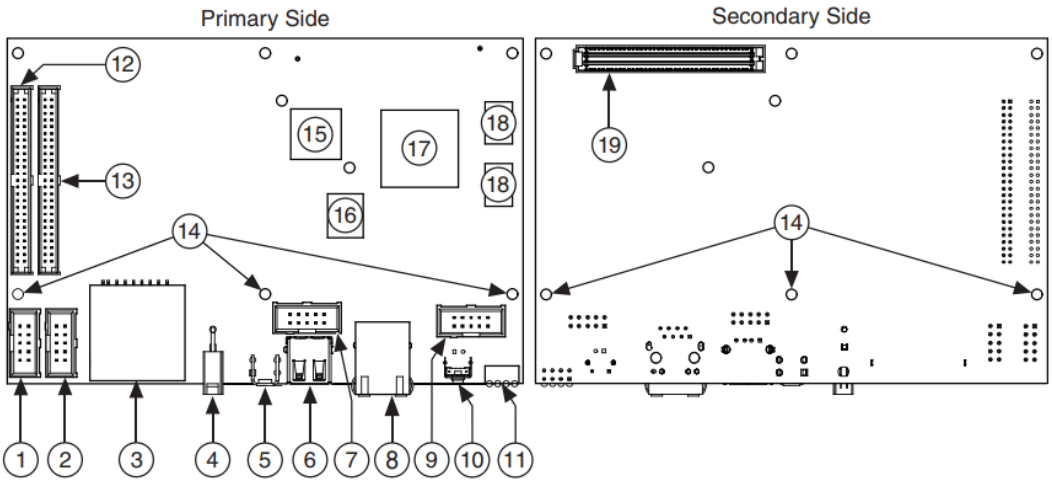


Figura 3.8. Componentes del sistema embebido SbRio-9636

1. W502, RS-485 (Puerto serie COM3)	11. LEDs
2. W503, RS-232 (Puerto serie COM2)	12. J502 DIO (entradas/salidas)
3. J504, SDHC (Lector de tarjetas SD)	13. J503 MIO (entradas/salidas)
4. J506, Power supply	14. Ranuras de montaje para atornillar
5. Masa del chasis	15. FPGA
6. J507, Puerto USB	16. NAND flash
7. W500, CAN (puerto CAN0)	17. Procesador
8. J505, RJ-45 puerto Ethernet	18. Memorias DDR RAM
9. W501, RS-232 (Puerto seria COM1)	19. J1, Conector de la tarjeta Mezzanine
10. Botón de reset	

Tabla 3.1. Componentes del SbRio-9636

Esta configuración dividida en dos partes fundamentales constituye en su conjunto un sistema embebido fundamental.

Las entradas y salidas (puntos 12 y 13) estarán conectadas a su vez a una placa acoplada en la parte superior del sistema embebido descrito previamente. Esta placa hará más accesibles las entradas y salidas analógicas y digitales además de añadir más funcionalidades como una pantalla LCD, botones, leds y otros elementos que aparecerán descritos a continuación.

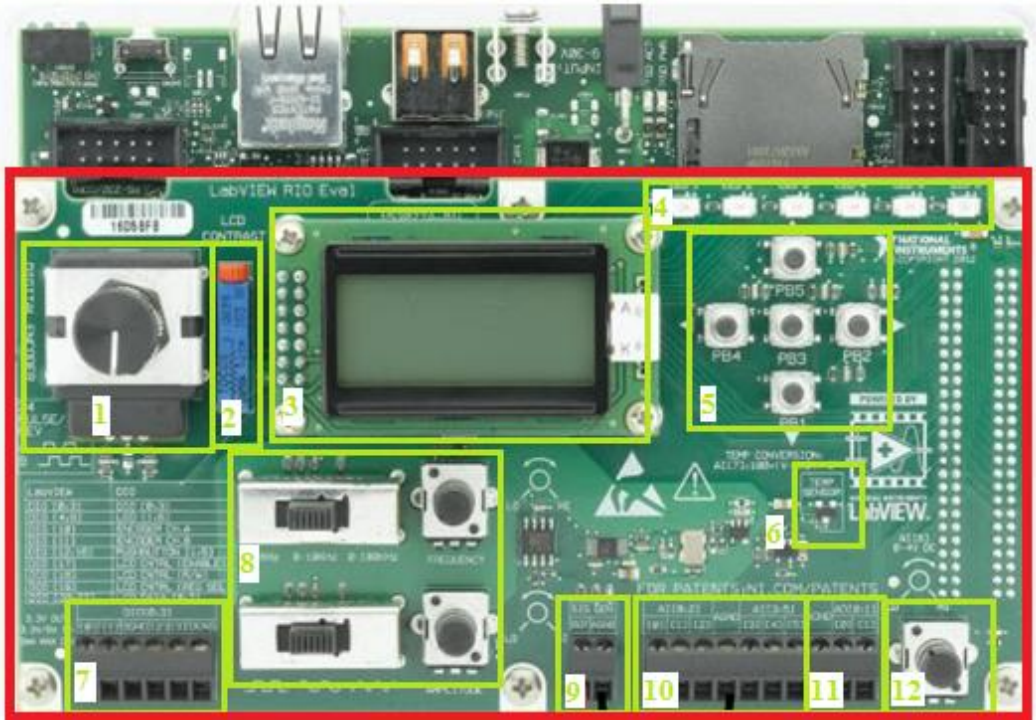


Figura 3.9. Placa superior del SbRio-9636

1. Potenciómetro	7. Entradas/salidas digitales (4)
2. Variador de contraste del LCD	8. Generador de señales periódicas
3. Pantalla LCD	9. Salida de la señal generada (1)
4. LEDs	10. Entradas analógicas (6)
5. Botones	11. Salidas analógicas (2)
6. Termómetro	12. Potenciómetro

Tabla 3.2. Partes de la placa superior del SbRio-9636

Será a través de esta placa como se realizarán todas las conexiones necesarias para manejar el motor de corriente continua a través de las entradas y las salidas digitales y analógicas.

Las **entradas/salidas digitales**serán de tensión de entre 5V y 3.3Vcon una intensidad de 3mA. También, están ruteadas a una impedancia característica de unos 55 Ohmios.

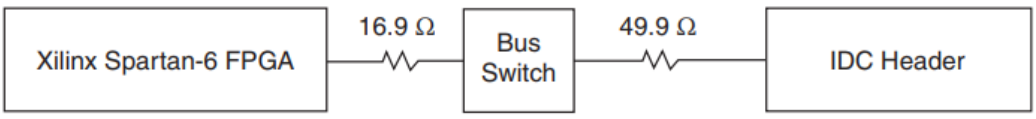


Figura 3.10. Esquema de una entrada/salida digital.

Las **entradas analógicas** tendrán un rango de entre -10V/10V. La resolución es de 16bits, 2^{16} (65.536) posibles valores dentro de dicho rango. De esta manera la precisión de la adquisición es de 305uV.

Las entradas analógicas son de punto flotante

$$\frac{10\ \mathrm{V} - (-10\ \mathrm{V})}{2^{16}} = 305\mathrm{uV}$$

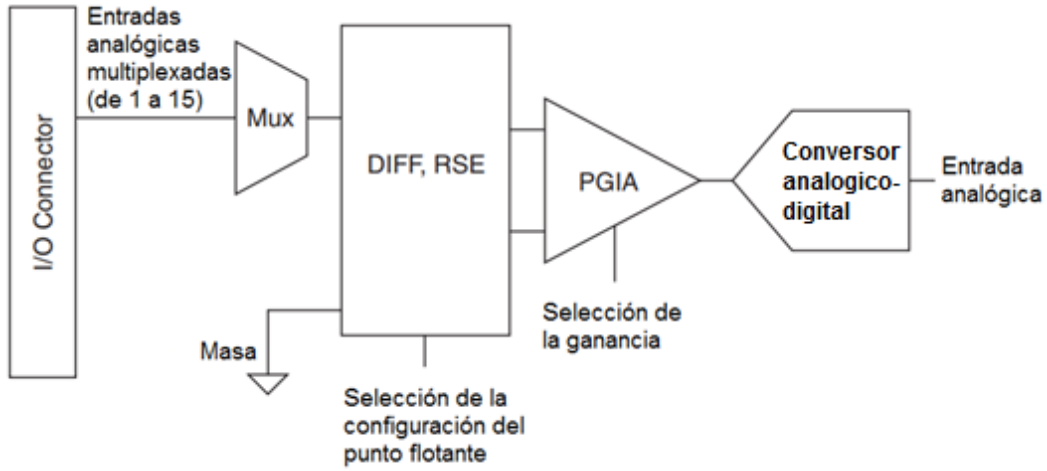


Figura 3.11. Esquema de una entrada analógica

Las **salidas analógicas** también serán entre un rango de -10V/10V y con la misma resolución que las entradas (2^{16}), de manera que la precisión es la misma que la aportada por las entradas analógicas, 305 uV.

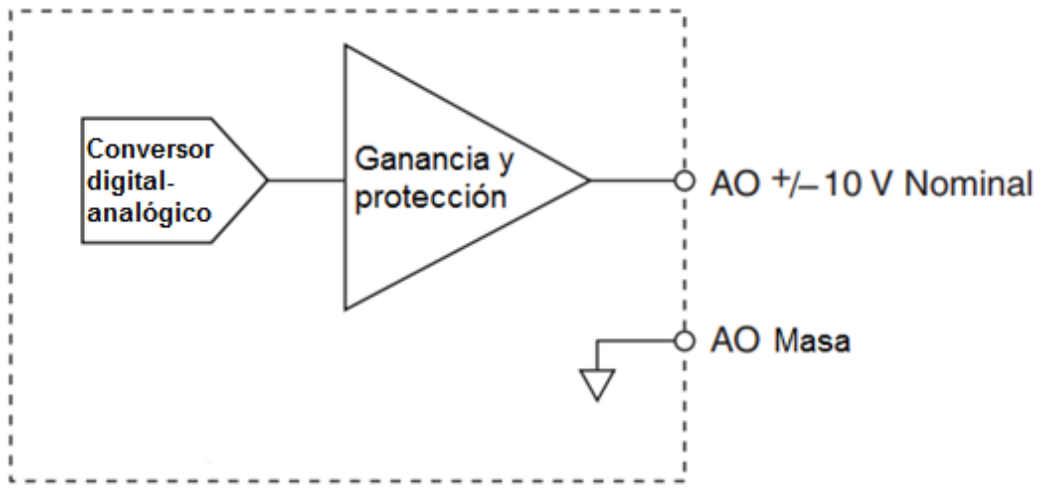


Figura 3.12. Esquema de una salida analógica

3.5. NI myRIO-1900

El controlador de automatización programable NI myRIO es un dispositivo embebido que cuenta con diferentes módulos de E/S tanto analógicas como digitales, LEDs, un push-button, un acelerómetro interno, un FPGA Xilinx y un procesador dual-core ARM Cortex-A9. Además, este modelo incluye conexión Wi-Fi. [NI]

Como podremos ver a continuación, el Software y gran parte del Hardware de este dispositivo es el mismo que el del sbRIO-9636 siendo la gran diferencia y ventaja frente a este la cantidad de entradas ó salidas digitales de las que disponemos.

Figura 3.13. NI myRIO 1900

En el siguiente diagrama podemos ver la distribución y funciones de los componentes del NI myRIO 1900, entre las que se distinguen el procesador y el FPGA.

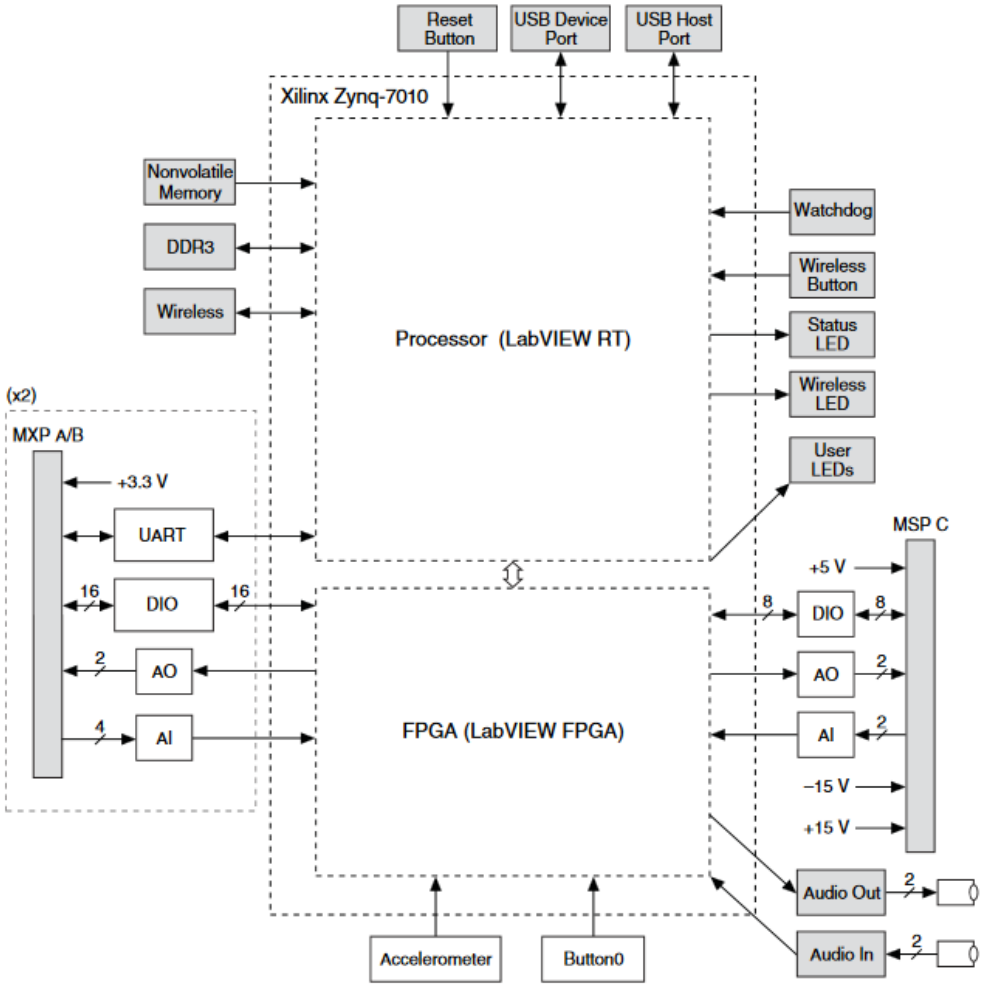


Figura 3.14. Componentes NI myRIO 1900

Conectores

El equipo dispone de tres conectores claramente diferenciados, nombrados A, B y C. Los conectores A y B son del tipo MXP, y son idénticos. Las señales pueden diferenciarse en el software por el nombre del conector, por ejemplo: ConnectorA/DIO1 o ConnectorB/DIO. En la siguiente figura podemos observar el pinout de estos conectores, alguno de los pins ofrece funciones secundarias:

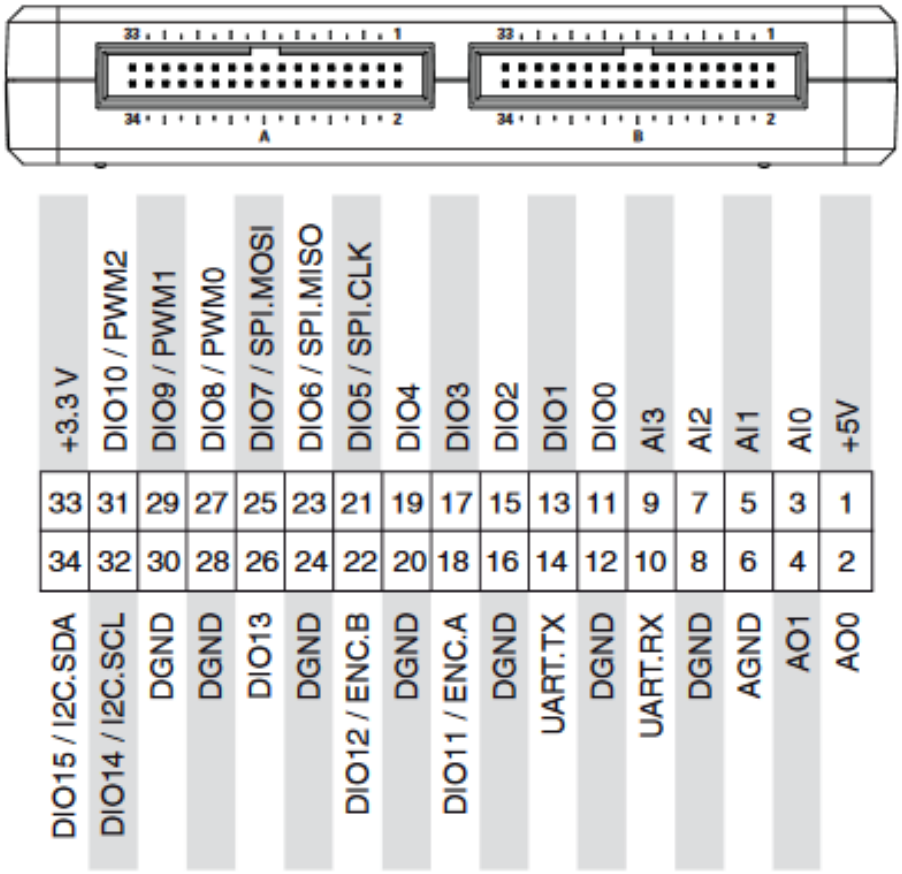


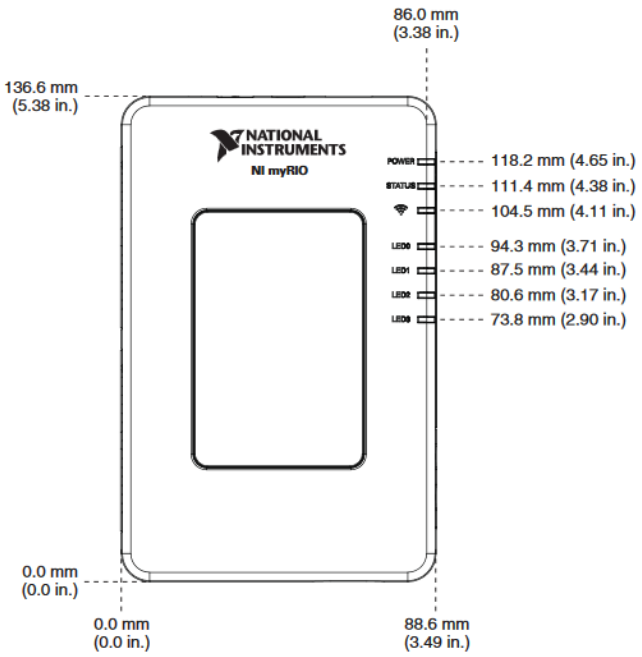
Figura 3.15. Pinout conector MXP NI myRIO 1900

Además, en la siguiente tabla se recoge la descripción de las señales que ofrece cada conector, así como el rango de voltaje al que trabajan las señales digitales y analógicas.

Signal Name	Reference	Direction	Description
+15V/-15V	AGND	Output	+15 V/-15 V power output.
AI0+/AI0-; AI1+/AI1-	AGND	Input	±10 V, differential analog input channels. Refer to the *Analog Input Channels* section for more information.
AO <0..1>	AGND	Output	±10 V referenced, single-ended analog output channels. Refer to the *Analog Output Channels* section for more information.
AGND	N/A	N/A	Reference for analog input and output and +15 V/-15 V power output.
+5V	DGND	Output	+5 V power output.
DIO <0..7>	DGND	Input or Output	General-purpose digital lines with 3.3 V output, 3.3 V/5 V-compatible input. Refer to the *DIO Lines* section for more information.
DGND	N/A	N/A	Reference for digital lines and +5 V power output.

Tabla 3.3. Conector MXP NI myRIO-1900

En cuanto a las dimensiones del equipo, aparecen recogidas en las siguientes imágenes.



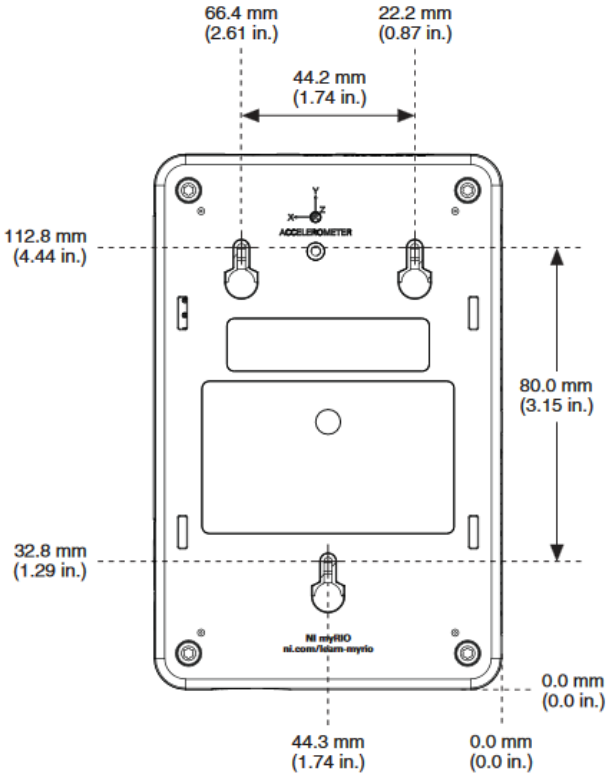


Figura 3.16. NI myRIO-1900 frontal (izquierda) y trasera (derecha)

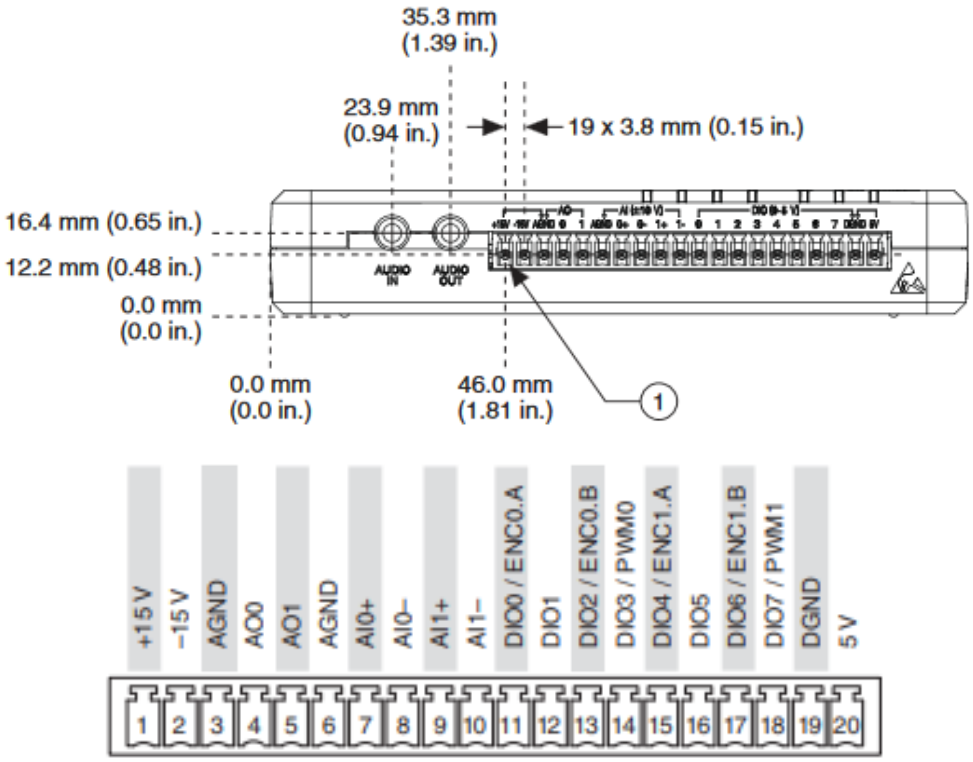


Figura 3.17. NI myRIO-1900 lateral 1

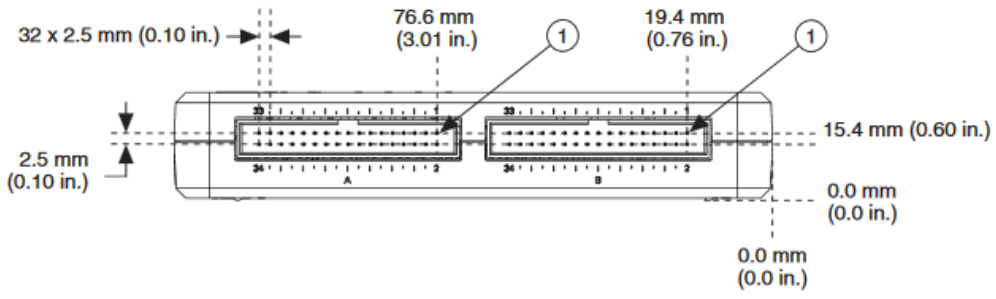


Figura 3.18. NI myRIO-1900 lateral 2

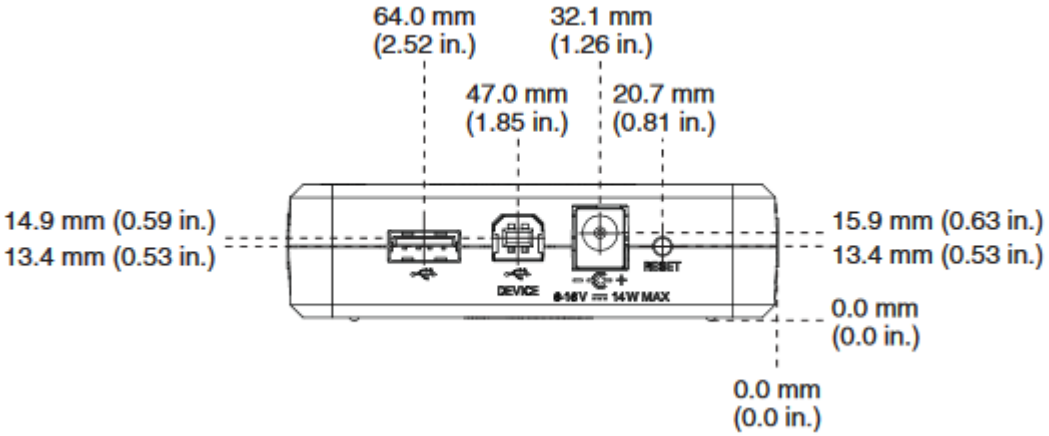


Figura 3.19. NI myRIO-1900 lateral 3

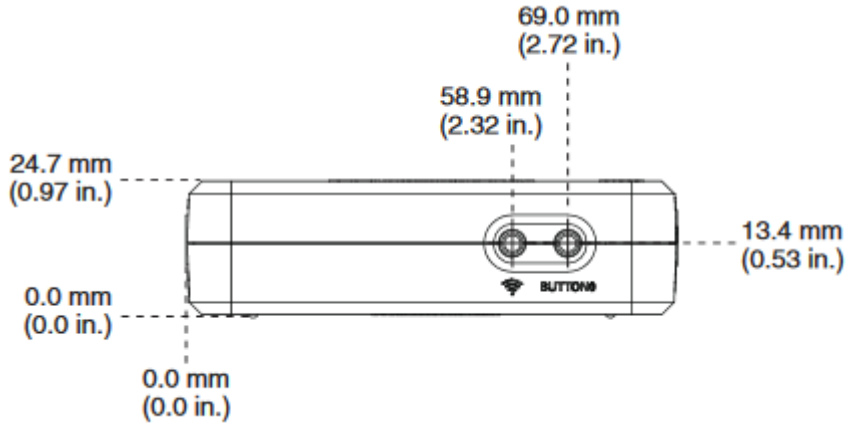


Figura 3.20. NI myRIO-1900 lateral 4

Capítulo 4

Problema de Hardware con el sbRIO 9636 y solución con myRIO 1900

Como hemos podido ver en el apartado anterior, para el manejo de motor necesitamos en el sistema embebido de 6 entradas ó salidas digitales y 2 salidas analógicas.

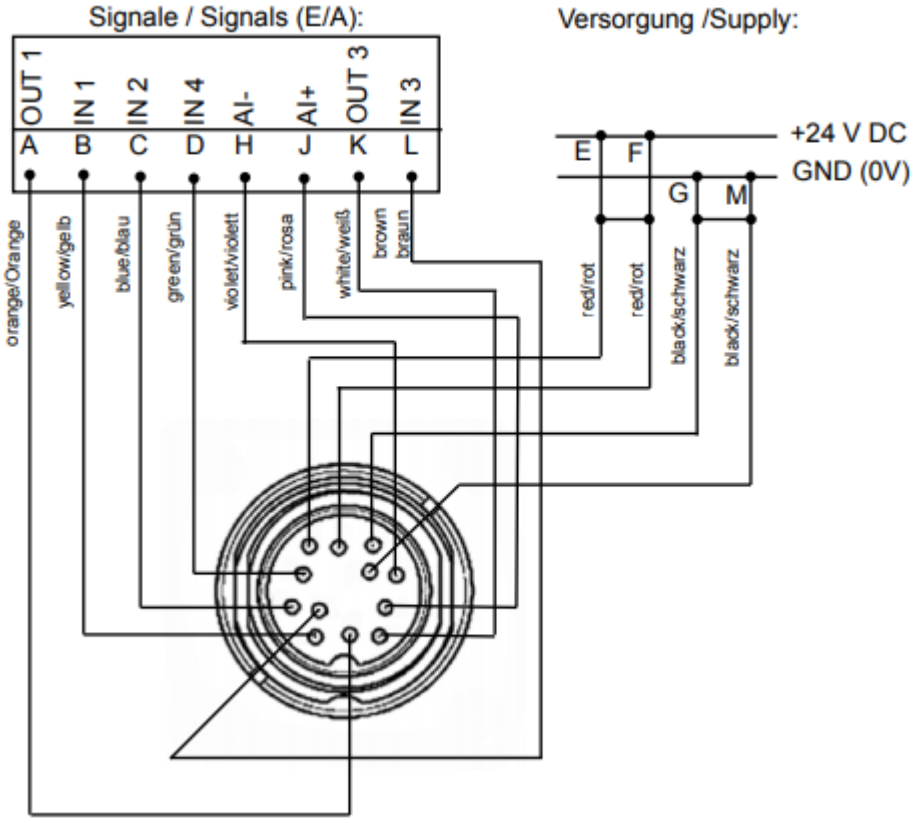


Figura 4.1. Conexionado sistema embebido-motor.

Sin embargo en el sistema embebido sbRIO 9636 únicamente disponemos de 4 entradas ó salidas digitales, 6 entradas analógicas y 2 salidas analógicas.

Figura 4.2. En rojo entradas ó salidas digitales, en azul entradas analógicas y en verde las salidas analógicas del sbRIO 9636.

Esto hace que tengamos limitado el número de funciones que podemos utilizar. Estudiando las opciones que podríamos eliminar para hacer que el sistema funcione llegamos a la conclusión de que la mejor opción sería referenciar a 0 V o 24 V de tensión las entradas digitales IN3 e IN4 del motor, de manera que mantendríamos un control sobre el motor debido a que controlamos sus funcionalidades más básicas.

Tras fijar las entradas digitales IN3 e IN4 pasaríamos de una situación con 2^4 opciones de control (IN1, IN2, IN3 e IN4) descritas en la tabla 2.6. a 2^2 opciones de control.

IN1	IN2	IN3	IN4	Función
0	0	X	X	Control no activo, ni par motor ni par de frenada
1	0	X	X	Rotación en sentido horario
0	1	X	X	Rotación en sentido anti horario
1	1	X	X	Frenada manteniendo un par de frenada (STOP)
X	X	0	0	Velocidad regulada (150…4096rpm) con +Al y -Al
X	X	0	1	Velocidad no regulada (0…5000rpm) con +Al y -Al
X	X	1	0	Velocidad memorizada 1 (speed1) 200rpm
X	X	1	1	Velocidad memorizada 2 (speed2) 2500rpm

Tabla 4.1. Funcionalidades del motor en negro con una de las opciones en rojo configurada de manera fija.

La mejor opción en caso de usar sbRio sería fijar la IN3 a '0' (0 V) y la IN4 a '1' (24 V) de manera que tendríamos un rango de velocidad del eje en RPM ligeramente más amplio.

IN1	IN2	IN3	IN4	Función a velocidad NO regulada
0	0	0	1	Control no activo, ni par motor ni par de frenada
1	0	0	1	Rotación en sentido horario
0	1	0	1	Rotación en sentido anti horario
1	1	0	1	Frenada manteniendo un par de frenada (STOP)

Tabla 4.2. Funcionalidades del motor bajo la condición fija velocidad no regulada (0-5000rpm).

Finalmente y debido a esta limitación optamos por no usar el sbRIO 9636 como sistema embebido y pasaremos a usar el myRIO 1900.

Este sistema embedido myRIO 1900 dispone de 16 entradas ó salidas digitales, 4 entradas analógicas y 2 salidas analógicas, con las cuales podremos controlar sin ningún problema y utilizando el 100% de las utilidades del motor.

Capítulo 5

Adaptación de señales

Para que se puedan comunicar el sistema embebido y motor debemos de adaptar las señales digitales de entrada y salida de ambos para facilitar el conexionado ya que los rangos de tensión a los que trabajan son diferentes.

	Cero digital ‘0’	Uno digital ‘1’
Rango de tensión de entrada del ‘SE’	0V – 0.8V	2V – 5.25V
Rango de tensión de salida del ‘SE’	0V – 0.4V	2.4V – 3.465V
Rango de tensión del motor	0V – 7V	7V – 24V

Tabla 5.1. Rango de tensión de las entradas y salidas digitales.

Como hemos podido apreciar anteriormente las señales digitales proporcionadas por el motor trabajan a una tensión en un rango de 0V a 24V. De esta manera que obtendremos un ‘0’ con una tensión de 0V-7V y un ‘1’ con una tensión entre 7V-24V.

Para reducir la tensión de salida de las señales digitales enviadas por el motor a los inputs del sistema embebido utilizaremos un divisor de tensión. De esta forma las reduciremos desde un rango de 24V-0V a 5V-0V.

Del mismo modo, para elevar la tensión de entrada al motor que envía el sistema embebido necesitaremos una fuente externa de tensión que nos proporcione los 24V necesarios para conseguir un valor de ‘1’ en el input de la lógica digital del motor.

5.2. Optoacoplador

Un optoacoplador, es un dispositivo de emisión y recepción que funciona como un interruptor activado mediante la luz emitida por un diodo LED que satura un componente optoelectrónico, normalmente en forma de fototransistor o fototriac. De este modo se combinan en un solo dispositivo semiconductor, un fotoemisor y un fotorreceptor cuya conexión entre ambos es óptica. Estos elementos se encuentran dentro de un encapsulado que por lo general es del tipo DIP. Se suelen utilizar para aislar eléctricamente a dispositivos muy sensibles.

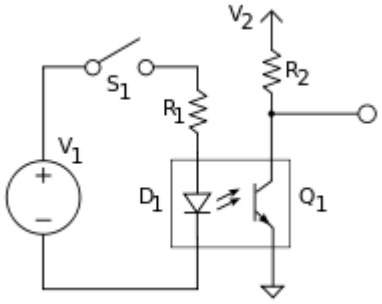


Figura 5.2. Optoacoplador 4N35 formado por un LED y un fototransistor.

La tensión de la fuente V1 de la izquierda y laresistencia en serie R1 establecen una corriente en el LED emisor D1 cuando se cierra el interruptor S1. Si dicha corriente proporciona un nivel de luz adecuado que, al incidir sobre el fototransistor Q1, lo saturará generando una corriente en R2. De este modo la tensión de salida será igual a cero con S1 cerrado y a V2 con S1 abierto.

La ventaja fundamental de un optoacoplador es el aislamiento eléctrico entre los circuitos de entrada y salida. Mediante el optoacoplador, el único contacto entre ambos circuitos es un haz de luz.

Este elemento es ideal para adaptar las señales de comunicación entre nuestro sistema embebidoy el motor sin escobillas debido a que todas las señales de entrada y de salida de ambos trabajan a distinta tensión.

El sistema que utilizaremos nosotros será el optoacoplador TPL281:

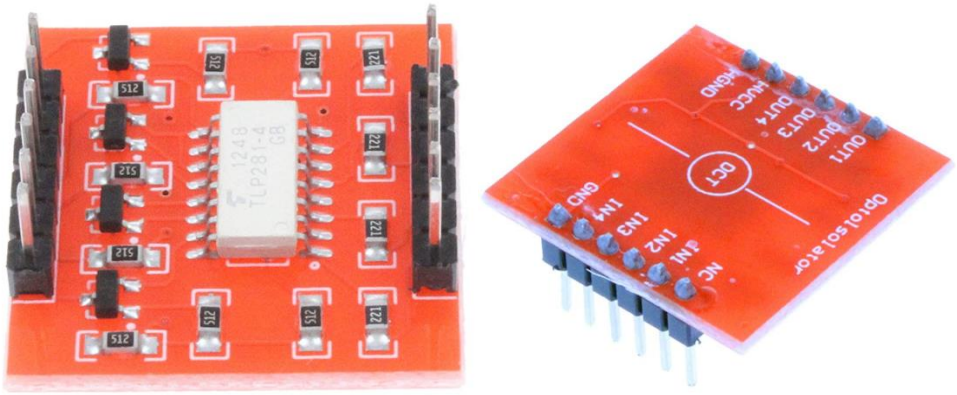


Figura 5.3. Optoacoplador TPL281

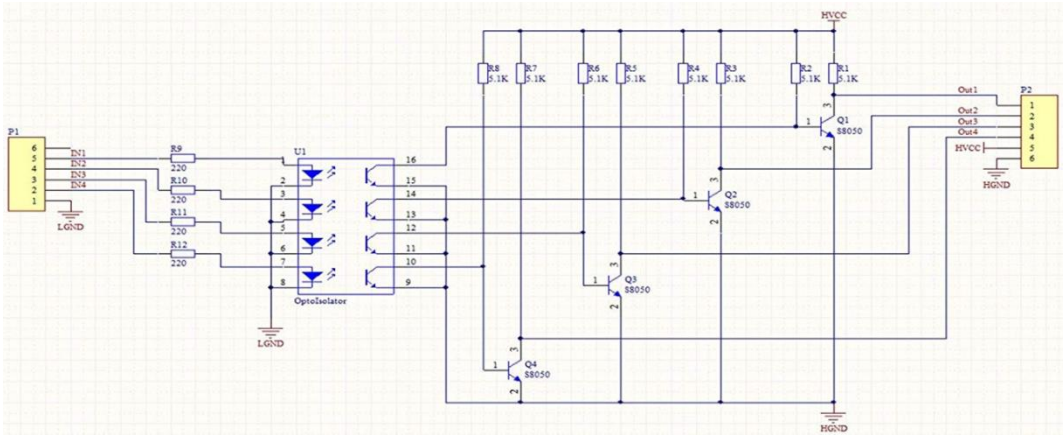


Figura 5.4. Esquema eléctrico del Optoacoplador TPL281

El sistema en caso de estar siendo aplicada una tensión comprendida entre 0 Voltios y 3,3 Voltios en las entradas aportará como salidas una tensión mayor que será la aportada por la fuente de alimentación de 24 V.

Esta configuración será perfecta para la elevación de tensión de las señales digitales IN1, IN2, IN3 e IN4, las cuales deberán ser elevadas de un ‘1’ de 5 V a 24 V. Sin embargo todavía tenemos la problemática de las señales digitales de entrada al sistema embebido, que tendrán valores de 0-24 V. Estas señales también tendrán que ser reducidas de 0-24 V a 0-5 V para ser incluidas en el motor. Para ello utilizaremos un segundo Optoacoplador, el ILD213T el cual alimentaremos a 12 V, reduciendo a la mitad la tensión a través de un divisor de tensión.

Finalmente el conexionado de la señal analógica de 10V que pilota la velocidad del motor será directo entre motor y sistema embebido.

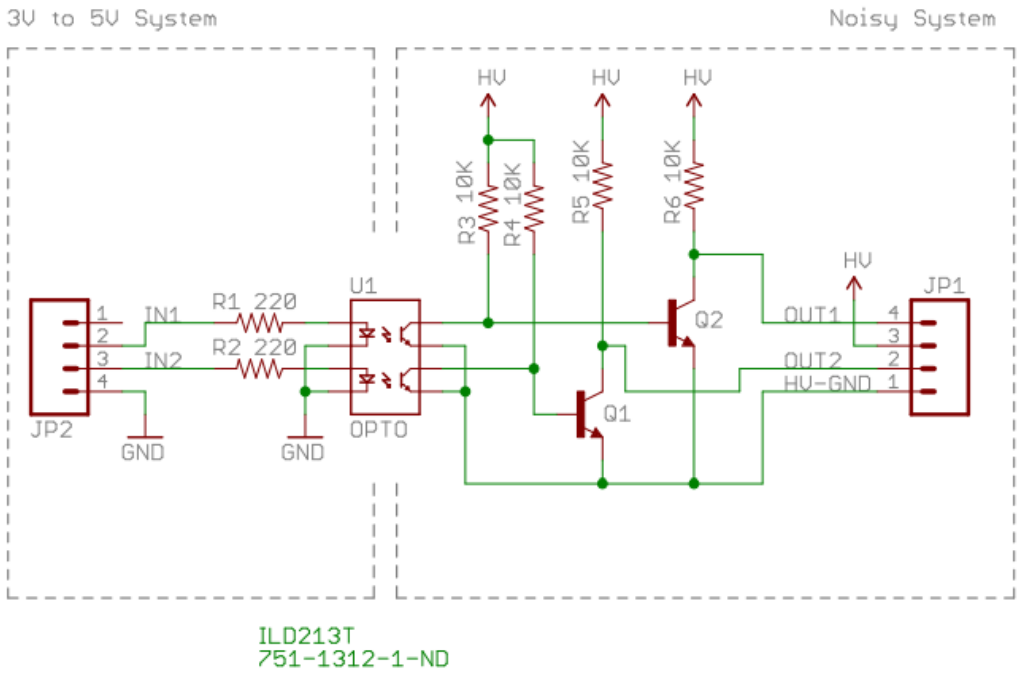


Figura 5.5. Esquema eléctrico del Optoacoplador ILD213T

Finalmente, para robustecer el conexionado e integrar los Optoacopladores hemos fabricado una caja plástica con una impresora 3D a través de OpenScad, cuyo diseño es prefabricado y adaptado dimensionalmente a los requerimientos espaciales de nuestro proyecto.

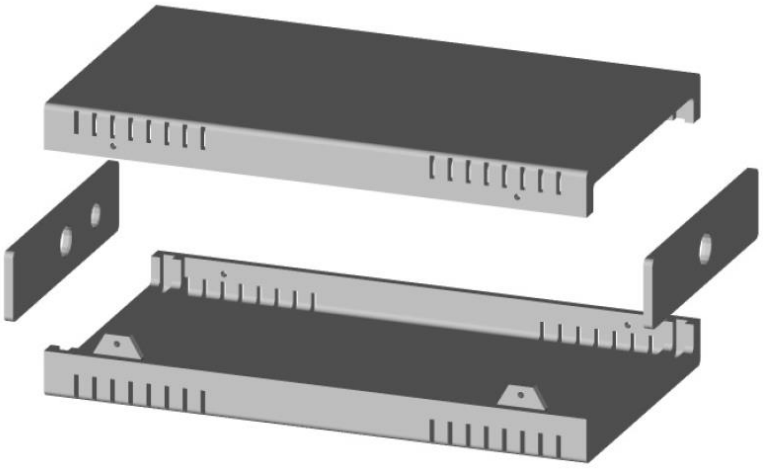


Figura 5.6. Diseño de la caja

A continuación se puede observar el resultado final del conexionado, y más abajo una tabla con equivalencias en cuanto a color de cable, función y rango de tensión de cada conexión.

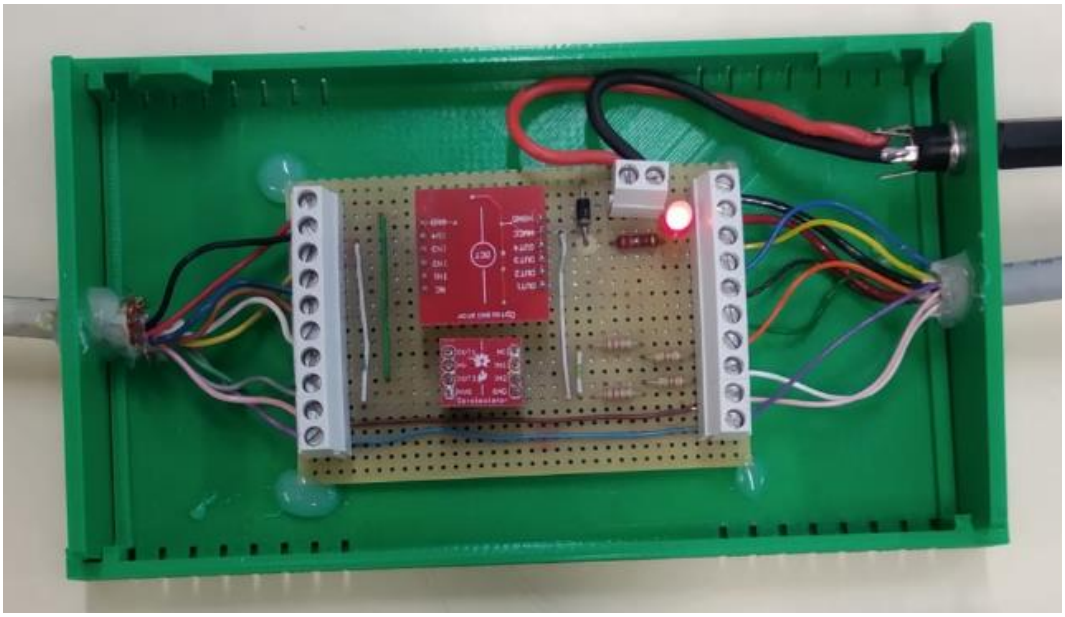


Figura 5.7. Caja con placa de optoacopladores y conexionado

Capítulo 6

Programación en LabView

6.1. ¿Qué es LabView?

LabView es un entorno de programación gráfico que se utiliza con el fin de crear aplicaciones rápida y eficazmente con interfaces de usuario profesionales. Es escalable a través de distintos objetivos y sistemas operativos, para ello, LabView ofrece integración en distintos tipos de dispositivos y facilita una gran cantidad de librerías integradas de acuerdo a las necesidades del usuario. Herramientas de análisis, visualización de datos, etc…

Ya que LabView imita la apariencia y operación de instrumentos físicos, como osciloscopios y multímetros, los programas de LabView son llamados instrumentos virtuales o VIs. Los VIs tienen un panel frontal y un diagrama de bloques. El panel frontal es el interface de usuario. El diagrama de bloques es el programa que hay detrás de la interface de usuario. Una vez creada la ventana del panel frontal, se puede añadir código usando representaciones gráficas de funciones para controlar los objetos del panel frontal. El código en el diagrama de bloques es código gráfico, también conocido como código G o código de diagrama de bloques.

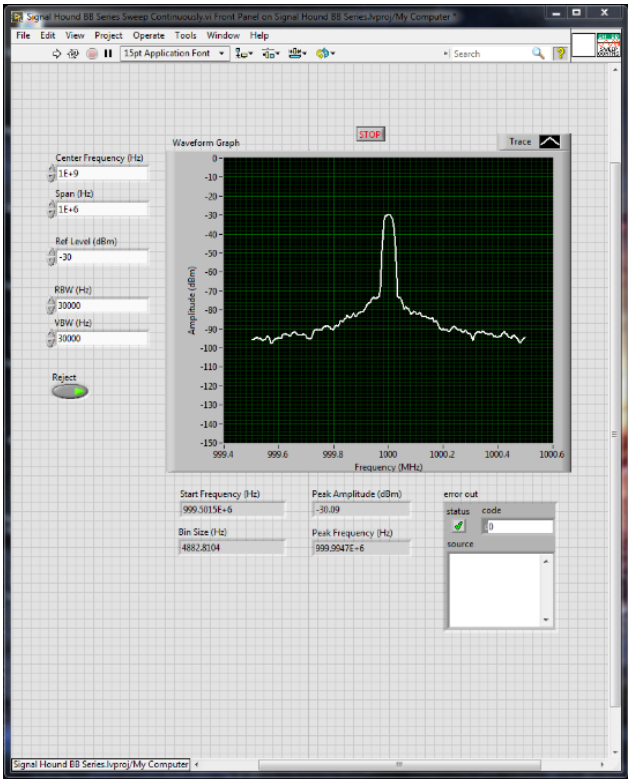


Figura 6.1. Ventana del panel frontal

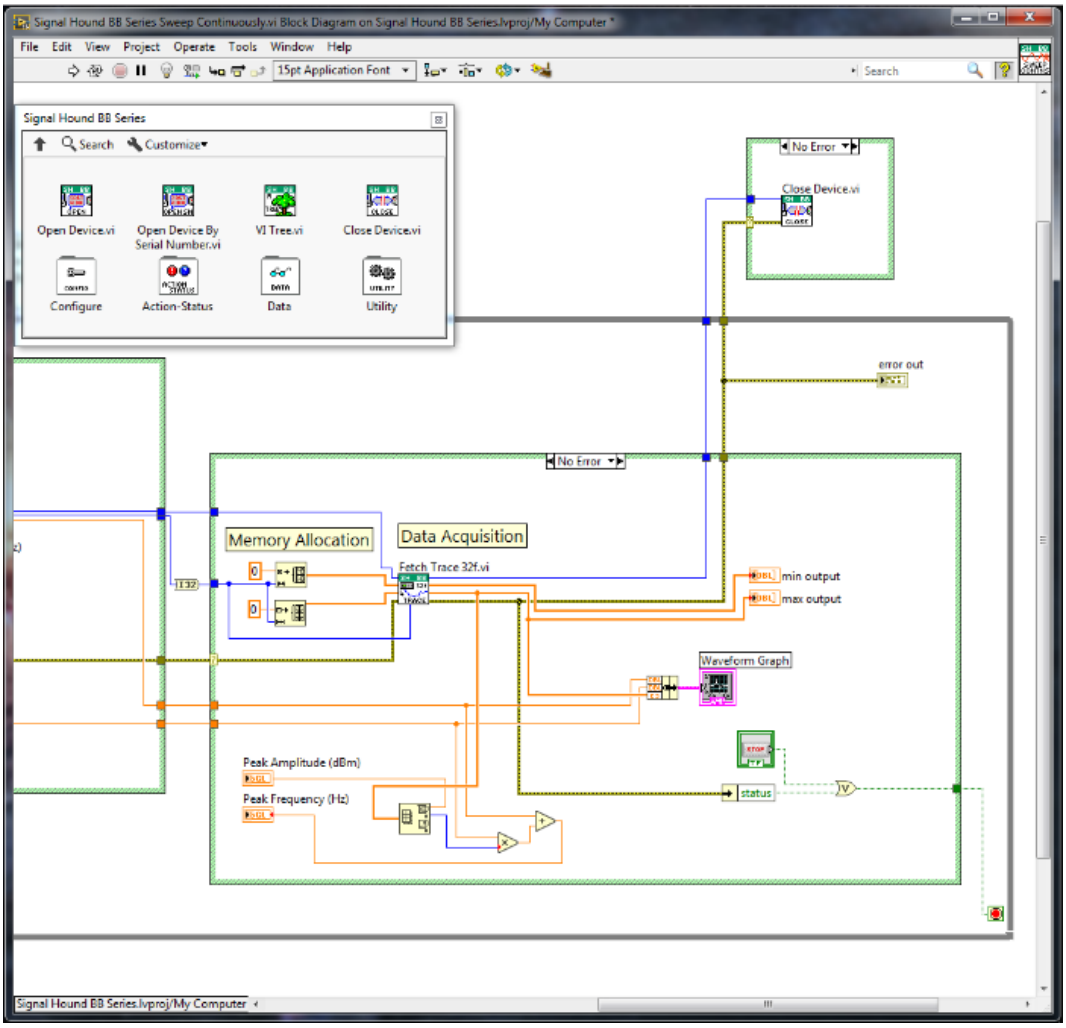


Figura 6.2. Ventana del diagrama de bloques.

A diferencia de lenguajes de programación basados en texto como C++ y Visual Basic, LabView usa iconos en vez de líneas de texto para crear las aplicaciones. De esta manera las instrucciones no determinan el orden de ejecución sino un flujo de datos gráficos. En este modelo de programación los datos fluyen a través de los nodos en el diagrama de bloques, lo cual determina el orden de ejecución. Son estos dos aspectos, la programación gráfica y la ejecución mediante flujo de datos son los que diferencian a LabView de otros lenguajes de propósito general.

Las características principales de Labview son:

- Naturaleza gráfica y compilada.
- Flujo de datos y/o programación basada en eventos.

- Capacidades multi-objetivo y plataforma.
- Flexibilidad orientada a objetos.
- Posibilidades multithreading (multitarea).

Aunque representado gráficamente con iconos y cables en vez de texto, el código G en el diagrama de bloques contiene los mismos conceptos de programación encontrados en la mayoría de los lenguajes tradicionales. LabView compila el código G directamente a código máquina, de manera que el procesador pueda ejecutarlo. No necesita compilar el código G en un paso adicional.

Los programas de LabView se ejecutan de acuerdo a las reglas de flujo de datos en lugar de la manera tradicional encontrada en la mayoría de lenguajes de programación basados en texto (C y C++). Esta depende de los datos, sin embargo, el flujo de datos entre los nodos en el código G determina el orden de ejecución.

La programación orientada a eventos extiende el concepto del flujo de datos para permitir al usuario interacción directa con el programa y permite también otras actividades asíncronas para influenciar la ejecución del código G en el diagrama de bloques.

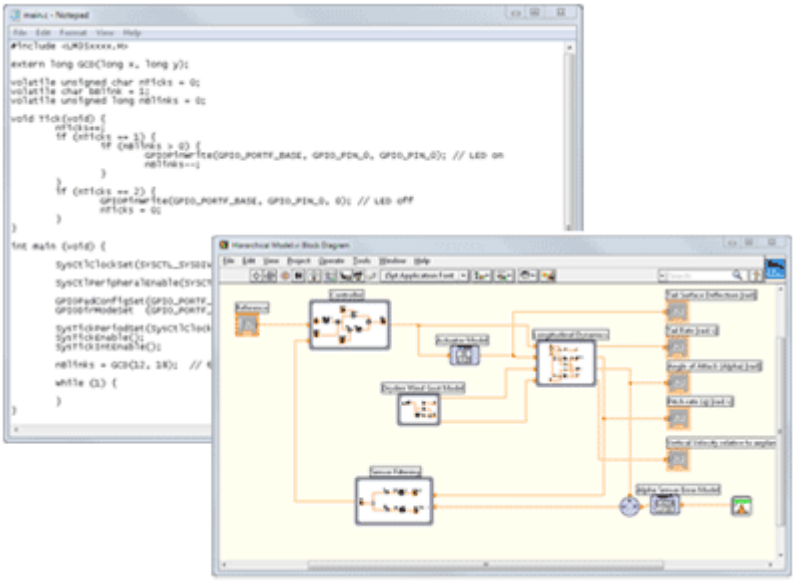


Figura 6.3. En la ventana posterior la programación en C correspondiente a la ventana anterior, programada en G.

Las aplicaciones ligadas a LabView tienen como objetivo procesadores multinúcleo y otros hardware paralelos como los Fiel-Programable GateArrais (FPGAs) de manera que es posible escalar automáticamente aplicaciones de LabView a CPUs con 2, 4 o más núcleos sin usar programación adicional.

6.2. Principales ventajas de usar LabView

Las ventajas que proporciona el empleo de LabView se resumen en las siguientes:

- Se reduce el tiempo de desarrollo de las aplicaciones al menos de 4 a 10 veces, ya que es muy intuitivo y fácil de aprender.
- Dota de gran flexibilidad al sistema, permitiendo cambios y actualizaciones tanto del hardware como del software.
- Da la posibilidad a los usuarios de crear soluciones completas y complejas.
- Con un único sistema de desarrollo se integran las funciones de adquisición, análisis y presentación de datos.
- El sistema está dotado de un compilador gráfico para lograr la máxima velocidad de ejecución posible.
- Tiene la posibilidad de incorporar aplicaciones escritas en otros lenguajes.

Todas estas ventajas se basan en las principales diferencias con respecto a los métodos de programación tradicionales.

- Los citados lenguajes de programación se basan en líneas de texto para crear el código fuente del programa, mientras que LabVIEW emplea la programación gráfica o lenguaje G para crear programas basados en diagramas de bloques.
- Para el empleo de LabVIEW no se requiere gran experiencia en programación, ya que se emplean iconos, términos e ideas familiares a científicos e ingenieros, y se apoya sobre símbolos gráficos en lugar de lenguaje escrito para construir las aplicaciones. Por ello resulta mucho más intuitivo que el resto de lenguajes de programación convencionales.
- LabVIEW posee extensas librerías de funciones y subrutinas. Además de las funciones básicas de todo lenguaje de programación, LabVIEW incluye librerías específicas para la adquisición de datos, control de instrumentación VXI, GPIB y comunicación serie, análisis presentación y guardado de datos.
- LabVIEW también proporciona potentes herramientas que facilitan la depuración de los programas. Facilitando de esta manera la compilación y la ejecución.

6.3. ¿Cómo trabajar con LabView?

Para describir el método de programación en LabView partimos de los instrumentos virtuales o VIs, que como hemos dicho anteriormente se componen del panel frontal y el diagrama de bloques, los cuales describiremos a continuación.

El panel frontal recoge las entradas procedentes del usuario y representa las salidas proporcionadas por el programa. Está formado por una serie de botones, pulsadores, potenciómetros, gráficos, etc…

Todos estos elementos se diferencian únicamente en 2 cosas, unos tendrán un rol de control y otros de indicación. Los primeros sirven para introducir parámetros al VI, mientras que los indicadores se emplean para mostrar los resultados producidos, ya sean datos adquiridos o resultados de alguna operación.

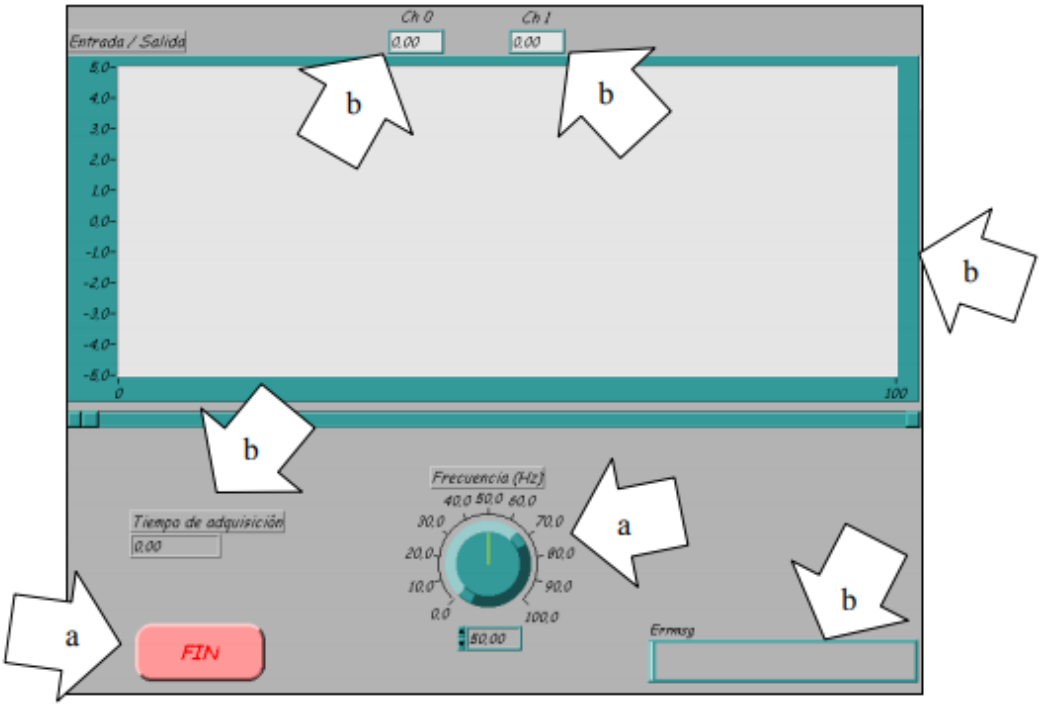


Figura 6.4. Visualización de un panel de control en el cual identificamos elementos de control (a) e indicación (b).

El diagrama de bloques constituye el código fuente del VI. En el diagrama de bloques es donde se realiza la implementación del programa del VI para controlar o realizar cualquier procesado de las entradas y salidas que se crearon en el panel frontal.

El diagrama de bloques incluye funciones y estructuras integradas en las librerías que incorpora LabView. En el lenguaje G las funciones y las estructuras son nodos elementales. Son análogas a los operadores o librerías de funciones de los lenguajes convencionales. Los controles e indicadores que se colocaron previamente en el Panel Frontal, se materializan en el diagrama de bloques mediante los **terminales**.

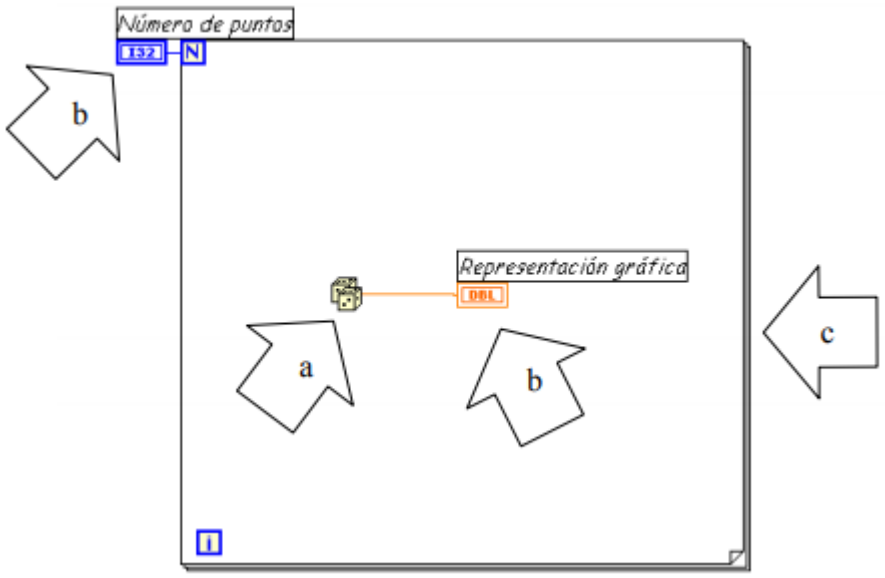


Figura 6.5. Visualización del diagrama de bloques donde se observan las funciones (a), los Terminales (b) y las estructuras (c).

El diagrama de bloques se construye conectando los distintos objetos entre sí, como si de un circuito se tratara. Los cables unen terminales de entrada y salida con los objetos correspondientes, y por ellos fluyen los datos.

LabVIEW posee una extensa biblioteca de **funciones**, entre ellas, aritméticas, comparaciones, conversiones, funciones de entrada/salida, de análisis, etc.

Las **estructuras**, similares a las declaraciones causales y a los bucles en lenguajes convencionales, ejecutan el código que contienen de forma condicional o repetitiva (bucle for, while, case,...).

Los cables son las trayectorias que siguen los datos desde su origen hasta su destino, ya sea una función, una estructura, un terminal, etc. Cada cable tiene un color o un estilo diferente, lo que diferencia unos tipos de datos de otros.

Las paletas son los herramientas a través de las que LabView permite modificar tanto el panel central como el diagrama de bloques. Las paletas disponibles son las siguientes:

- Paleta de herramientas (tools palette): Se emplea tanto en el panel frontal como en el diagrama de bloques. Contiene las herramientas necesarias para editar y depurar los objetos tanto del panel frontal como del diagrama de bloques.

Figura 6.6. Paleta de herramientas (tolos palette)

	Operating tool – Cambia el valor de los controles.
	Positioning tool – Desplaza, cambia de tamaño y selecciona los objetos.
	Labeling tool – Edita texto y crea etiquetas.
	Wiring tool – Une los objetos en el diagrama de bloques.
	Object Pop-up Menu tool – Abre el menú desplegable de un objeto.
	Scroll tool – Desplaza la pantalla sin necesidad de emplear las barras de desplazamiento.
	Breakpoint tool – Fija puntos de interrupción de la ejecución del programa en VIs, funciones y estructuras.
	Probe tool – Crea puntos de prueba en los cables, en los que se puede visualizar el valor del dato que fluya por dicho cable en cada instante.
	Color Copy tool – Copia el color para después establecerlo mediante la siguiente herramienta.
	Color tool – Establece el color de fondo y el de los objetos

Tabla 6.1. Herramientas de la paleta

- La paleta de controles (Controls palette): Se utiliza únicamente en el panel frontal. Contiene todos los controles e indicadores que se emplearán para crear la interfaz del VI con el usuario.

Figura 6.7. Paleta de controles (Controls palette)

	Numeric – Para la introducción y visualización de cantidades numéricas.
	Boolean – Para la entrada y visualización de valores booleanos.
	String & Table – Para la entrada y visualización de texto.
	List & Ring – Para visualizar y/o seleccionar una lista de opciones.
	Array & Cluster – Para agrupar elementos.
	Graph – Para representar gráficamente los datos.
	Path & RefNum – Para gestión de archivos.
	Decorations – Para introducir decoraciones en el panel frontal. No visualizan datos.
	User Controls – Para elegir un control creado por el propio usuario.
	ActiveX – Para transferir datos y programas de unas aplicaciones a otras dentro de Windows.
	Select a Control – Para seleccionar cualquier control.

Tabla 6.2. Herramientas de la paleta de controles

- Paleta de funciones (functions palette): Se emplea en el diseño del diagrama de bloques. La paleta de funciones contiene todos los objetos que se emplean en la implementación del programa del VI, ya sean funciones aritméticas, de entrada/salida de señales, entrada/salidad de datos a fichero, adquisición de señales, temporización de la ejecución del programa...

Figura 6.8. Paleta de funciones (functions palette)

	Structures – Muestra las estructuras de control del programa, junto con las variables locales y globales.
	Numeric – Muestra funciones aritméticas y constantes numéricas.
	Boolean – Muestra funciones y constantes lógicas.
	String – Muestra funciones para manipular cadenas de caracteres, así como constantes de caracteres.
	Array – Contiene funciones útiles para procesar datos en forma de vectores, así como constantes de vectores.
	Cluster – Contiene funciones útiles para procesar datos procedentes de gráficas y destinados a ser representados en ellas, así como las correspondientes constantes.
	Comparison – Muestra funciones que sirven para comparar números, valores booleanos o cadenas de caracteres.
	Time & Dialog – Contiene funciones para trabajar con cuadros de diálogo, introducir contadores y retardos, etc.

	File I/O – Muestra funciones para operar con ficheros.
	Communication – Muestra diversas funciones que sirven para comunicar varios ordenadores entre sí, o para permitir la comunicación entra distintos programas.
	Instrument I/O – Muestra un submenú de VIs, que facilita la comunicación con instrumentos periféricos que siguen la norma ANSI/IEEE 488.2-1987, y el control del puerto serie.
	Data Acquisition – Contiene a su vez un submenú donde puede elegirse entre distintas librerías referentes a la adquisición de datos.
	Analysis – Contiene un submenú en el que se puede elegir entre una amplia gama de funciones matemáticas de análisis.
	Tutorial – Incluye un menú de VIs que se utilizan en el manual LabVIEW Tutorial.
	Advanced – Contiene diversos submenús que permiten el control de la ayuda, de los VIs, manipulación de datos, procesado de eventos, control de la memoria, empleo de programas ejecutables o incluidos en librerías DLL, etc.
	Instrument drivers – En él se muestran los drivers disponibles de distintos instrumentos.
	User Libraries – Muestra as librerías definidas por el usuario. En este caso, la librería mostrada contiene los drivers de la tarjeta de adquisición de datos de Advantech.
	Aplication control – Contiene varias funciones que regulan el funcionamiento de la propia aplicación en ejecución.
	Select a VI – Permite seleccionar cualquier VI para emplearlo como subVI.

Tabla 6.3. Herramientas de la paleta de funciones.

6.4. Ejecución de un VI

Una vez finalizada la programación de un VI nos disponemos a ejecutarlo, lo cual se debe de hacer a través del panel central. Para comenzar con la ejecución del programa pulsaremos el botón 'run' para ejecutar una única vez el programa, o el botón 'continous run' para ejecutarlo de manera continua.

Figura 6.9. Botón RUN en el panel central

Para parar el programa de debe de pulsar los botones de pausa o stop, la diferencia entre ambos es que si pulsas stop el programa detiene de manera instantánea su ejecución, si pulsas el botón de stop paras el programa pudiendo relanzar de nuevo su ejecución más adelante.

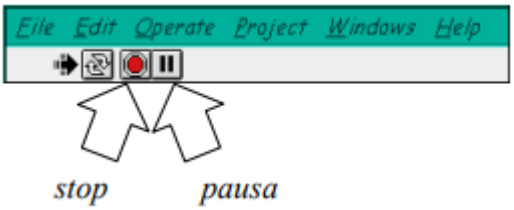


Figura 6.10. Botones de 'PAUSA' y 'STOP'

6.5. Estructuras

Las estructuras se ejecutan de manera automática todo lo que hay en su interior una vez disponibles todos los datos de entrada y tras finalizar las tereas programadas en su interior suministran los datos correspondientes a los cables unidos a sus salidas.

El subdiagrama será el conjunto de nodos, calves y terminales en el interior del rectángulo que constituye la estructura.

El For Loop y el While Loop únicamente tienen un subdiagrama. El Case Structure y el Sequence Structure, sin embargo, pueden tener múltiples subdiagramas, superpuestos como si se tratara de cartas en una baraja, por lo que en el diagrama de bloques únicamente será posible visualizar al tiempo uno de ellos. Los subdiagramas se construyes del mismo modo que el resto del programa.

Case structure

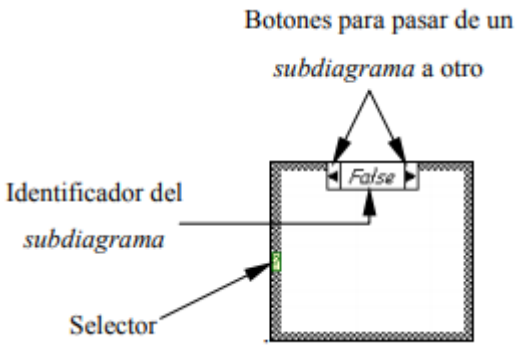


Figura 6.11. Case structure

Esta estructura se compone de subdiagramas como si de las cartas de una baraja se tratase y, a su vez, se puede cambiar de un subdiagrama a otro dependiendo del valor tomado por el selector. Manualmente se podrá cambiar pulsando en los botones y el identificador nos dirá qué subdiagrama estamos visualizando.

Sequence structure

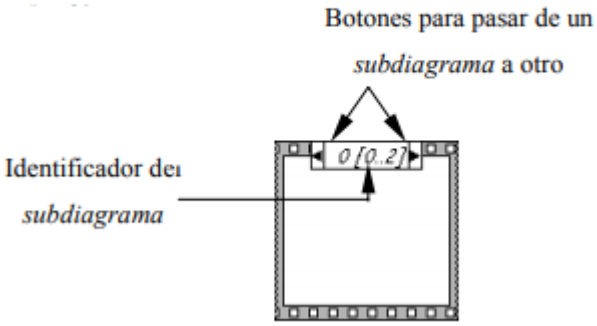


Figura 6.12. Sequence structure

De nuevo compuesto de subdiagramas superpuestos como si de una baraja de cartas se tratase de manera que solo se puede visualizar uno de los subdiagramas a la vez. Aun así se puede pivotar de un subdiagrama a otro con los botones que rodean al indicador, el cual te dice en qué subdiagrama concreto estás situado.

La ejecución de esta estructura es secuancial, es decir, en primer lugar se reproducirá el subdiagrama de la primera hoja '0' y a continuación el siguiente '1' y así sucesivamente tantas veces como hojas tenga la estructura.

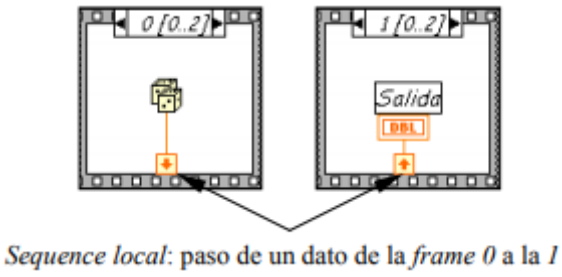


Figura 6.13. Método de ejecución de la Sequence structure

For loop

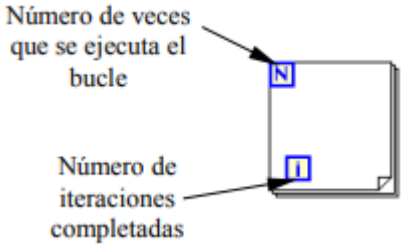


Figura 6.14. For Loop

Este es el equivalente al bucle 'for' de las herramientas de programación convencionales. Este bucle se rige por la siguiente lógica:

$$For\ 'i = 0'\ to\ 'N - 1' \rightarrow Ejecutar\ el\ subdiagrama\ en\ el\ bucle$$

En esta estructura se pueden transmitir valores entre iteración e iteración a través de 'shift registers'. Estos constan de dos terminales a ambos lados de la estructura. En el izquierdo se almacena el valor de la iteración anterior y el derecho el valor modificado tras la actual iteración, de manera que tras la última iteración el valor del lado derecho pasará al iquierdo preparándose para la siguiente iteración.

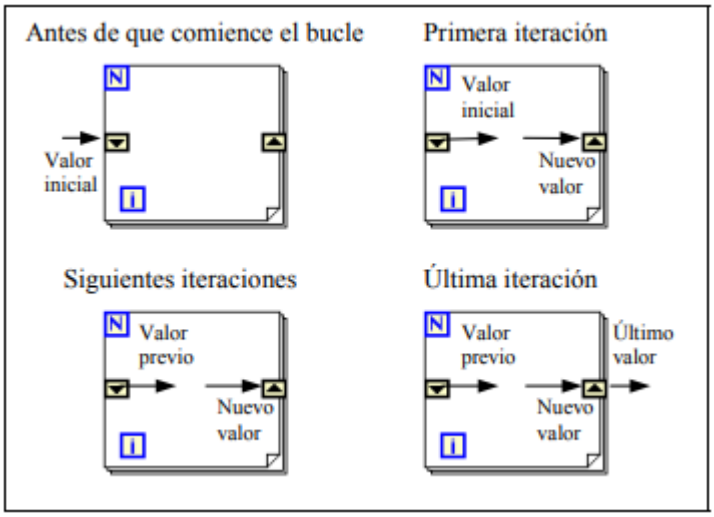


Figura 6.15. Secuencia de ejecución de los 'shift register'

Esta herramienta también puede almacenar valores de iteraciones previas para ello hay que pulsar con el derecho del ratón sobre el 'shift' izquierdo seleccionando a continuación 'Add Element'.

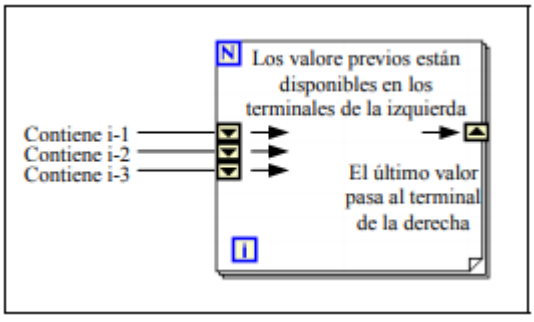


Figura 6.16. Mostrados 3 valores de iteraciones previas.

While loop

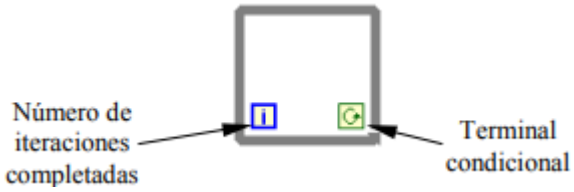


Figura 6.17. While loop

Éste bucle responde a la siguiente lógica:

$$\boldsymbol{Hacer}\ 'lógicainterna\ del\ bucle'\boldsymbol{mientras}\ condicional\ en\ el\ terminal\ sea\ True$$

Con esta lógica, el bucle se ejecutará al menos una vez y al igual que el bucle For se podrán utilizar 'shift registers'.

Formula node

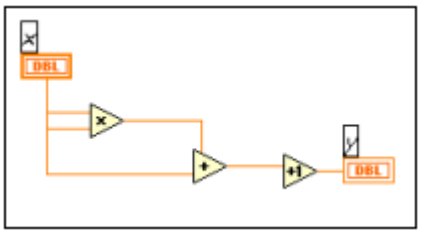


Figura 6.18. Fórmula ejecutada en bloques, programación en lenguaje 'G'

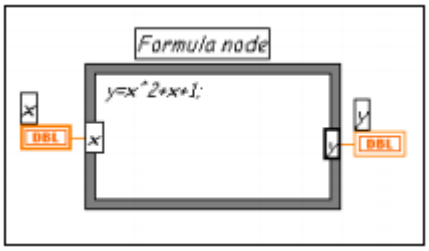


Figura 6.19. La misma fórmula anterior representada en fórmula dentro de la estructura formula node

Para definir una fórmula mediante este estructura, se actuará del siguiente modo:

- En primer lugar, se deben definir las variables de entrada y las de salida. Para ello, se pulsa con el botón derecho del ratón sobre el borde de la formula node. A continuación se seleccionará Add Input o Add Output, según se trate de una entrada o una salida, respectivamente. Todas las variables que se empleen deben estar declaradas como entradas o

salidas. Las que se empleen como variables intermedias se declararán como salidas, aunque no se usen.

- Una vez definidas las variables a emplear. Cada fórmula debe finalizar con un “;”.
- Los operadores y funciones que se pueden emplear se explican en la ayuda de LabVIEW.

Capítulo 7

Programa en LabView para el control del motor

7.1. *Panel de control*

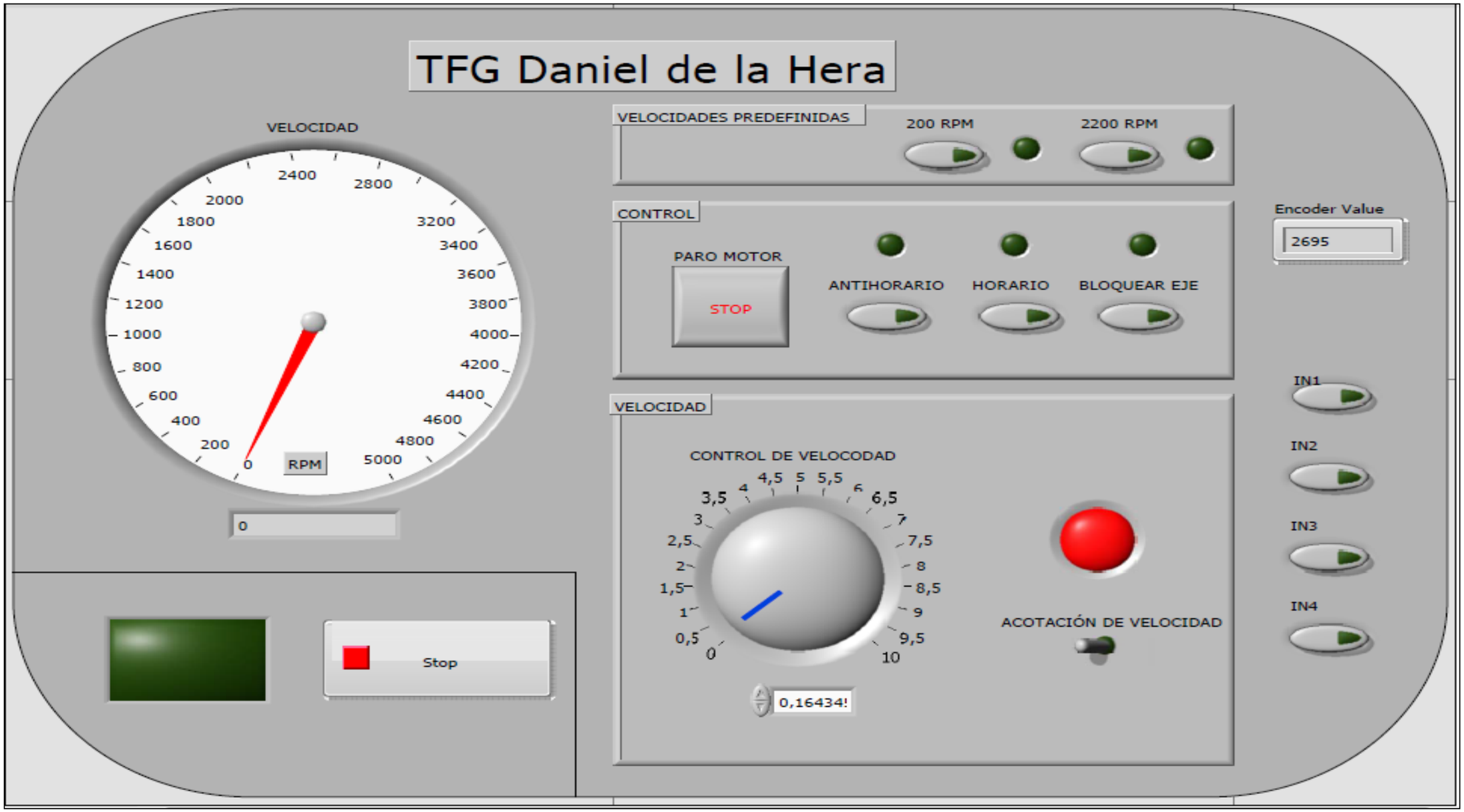


Figura 7.1. Panel de control

Para comenzar la simulación pulsaremos el botón RUN en la barra de herramientas del programa de LabView. El programa pasará a ejecutarse y, a continuación, pasaremos a gestionar el motor íntagramente con el Panel de control mostrado en la página anterior. Éste panel consta de los apartados:

- Control: En este apartado disponemos de un botón de parada motor, en el cual dejamos el eje sin carga, de manera se puede modificar su posición (el motor no aplica un par motor), el botón de selección de giro antihorario con un led que actua como testigo de esta función, el botón de selección de giro horario con su testigo de indicación y por último el botón de bloqueo del eje con su botón de indicación (el motor queda parado aplicando un par motor que se opone a cualquier fuerza que lo haga rotar)

Figura 7.2. Apartado de control

- Velocidades predefinidas: En este apartado tenemos dos botones con sus respectivos indicadores con los que activaremos las velocidades predefinidas por el fabricante, las cuales son 200 RPM y 2200 RPM.

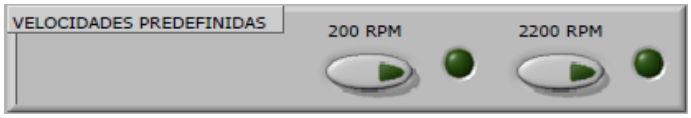


Figura 7.3. Apartado de velocidades predefinidas

- Testigo de funcionamiento de sistema, el cual se ilumina siempre que el programa esté funcionando y botón de parada 'STOP' que detiene la ejecución del programa.

Figura 7.4. Testigo verde de funcionamiento y botón de parada motor

- Velocidad: En este apartado disponemos de un selector circular (control de velocidad) a través del que pilotamos una salida analógica de 0 V a 10 V de tensión con el que controlaremos el motor siendo 0 V la tensión para 0 RPM y 10 V para llegar el máximo de RPM del motor. También habrá un interruptor 'acotación de velocidad' y un testigo del cual daremos la consigna de desactivación ó desactivación de velocidad controlada y se nos indicará en caso de que esté desactivada.

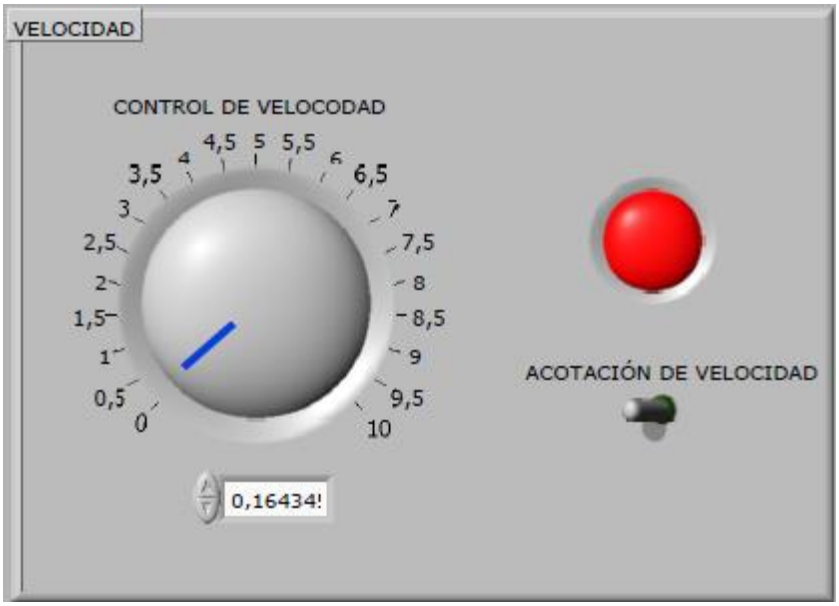


Figura 7.5. Apartado de Velocidad

- Indicador de velocidad, que indica las RPM reales del motor.

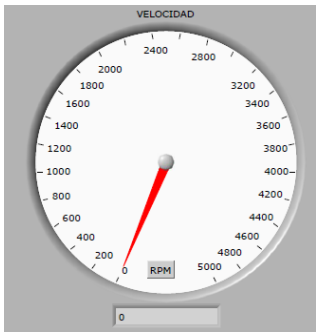


Figura 7.6. Indicador de RPM motor

Finalmente y como punto a destacar tenemos representados 4 leds con IN1, IN2, IN3 e IN4 de cuya combinación de valores depende los estados del motor y también tenemos una indicación del valor del encóder del cual

se obtiene la velocidad motor. Como no se trata de un producto destinado a cliente nos parece interesante visualizar estos parámetros a nivel didáctico.

7.2. Instrument panel

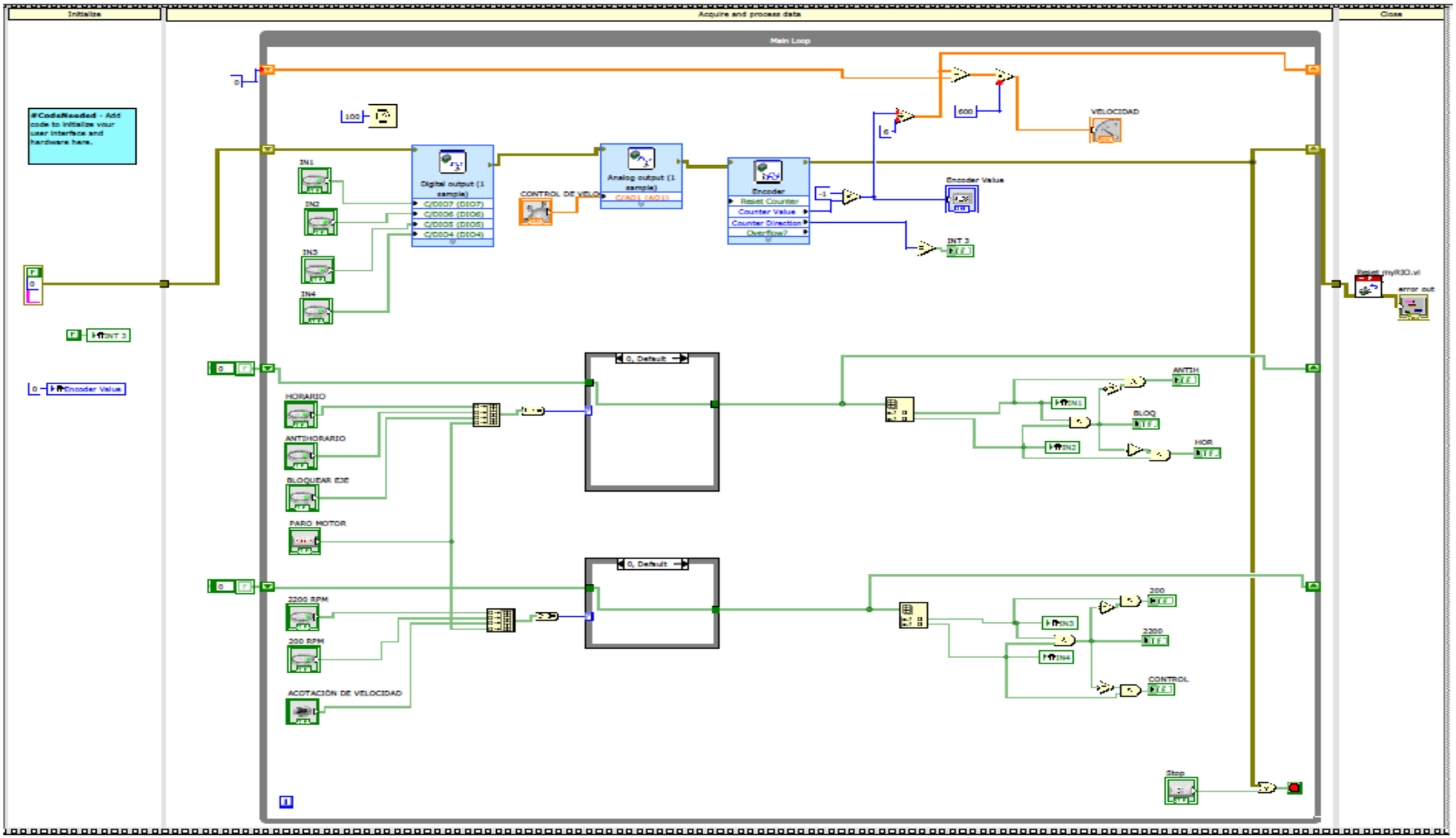


Figura 7.7. Instrument Panel

A continuación detallaremos el funcionamiento del diseño en labview. Antes de nada destacar que como indica el reloj en la parte superior izquierda de la pantalla, el ciclo Main Loop en el cual se incluye la ejecución del programa se repite cada 100ms y solo se parará en el momento que se pulse el botón de parada sobre el panel central o se transmita una señal de error por el sistema.

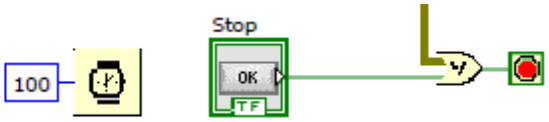


Figura 7.8. Tiempo de ejecución ciclo y parada motor del main loop

En primer lugar se pueden ver declaradas las distintas entradas y salidas entre el sistema embebido y el motor, las cuales son las 4 salidas digitales, la salida analógica y el encóder:

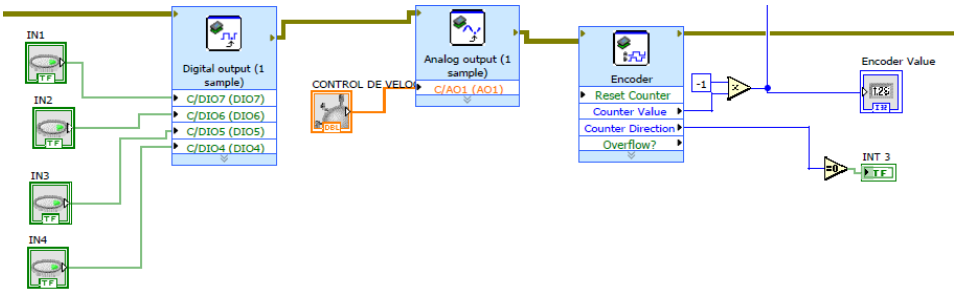


Figura 7.9. Entradas y salidas digitales y analógicas del sistema ambebido

El encoder funciona mediante sensores Hall, y por cada vuelta el motor enviará 6 pulsos digitales. Estos pulsos digitales los trataremos a través de la programación en LabView para calcular la velocidad motor a través de la siguiente fórmula:

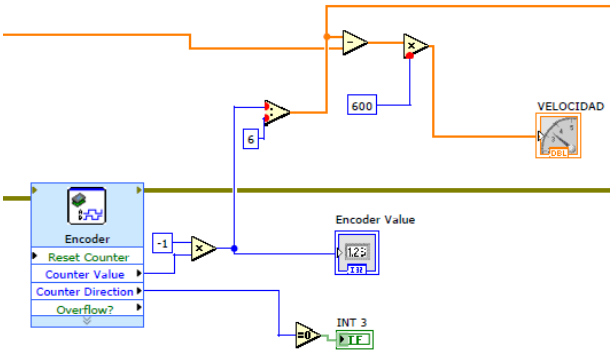


Figura 7.10. Gestión del Encoder

Para realizar el control del motor, usamos una seria de booleanas de manera que a través de botones sobre el panel central podemos cambiar el sentido de rotación de giro, la parada sin par o con par o las velocidades pre-programadas.

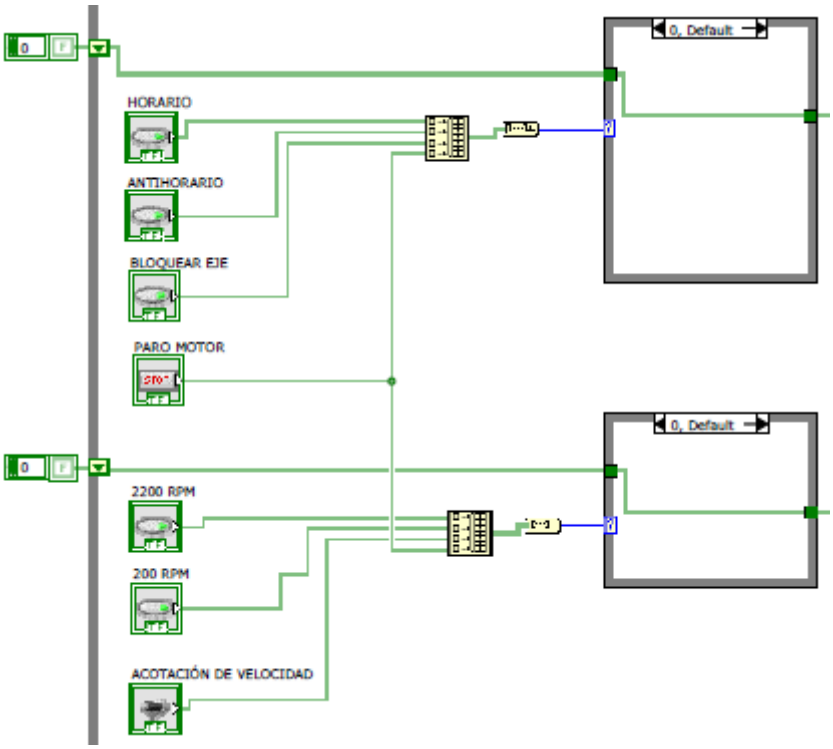


Figura 7.11. Gestión de los botones de control

Como se puede apreciar, dividimos las booleanas en dos grupos principales para facilitar su gestión, en primer lugar sentido horario, antihorario y bloqueao motor y por otro lado velocidades programadas 1, 2 y velocidad controlada. Estando presente en ambas combinaciones el botón de parada motor. El primer grupo gestiona las entradas IN1 e IN2 al motor y el segundo las IN3 e IN4.

Tanto en el primer grupo como en el segundo introduciremos la agrupación de las 4 booleanas en un array, que creará las 4 combinaciones numéricas que utilizaremos para pivotar entre cada una de las combinaciones de salidas digitales para gestionar el motor, es decir, dependiendo del botón pulsado, se cambiará el case estructure y este enviará la correspondiente combinación de entradas/ salidas.

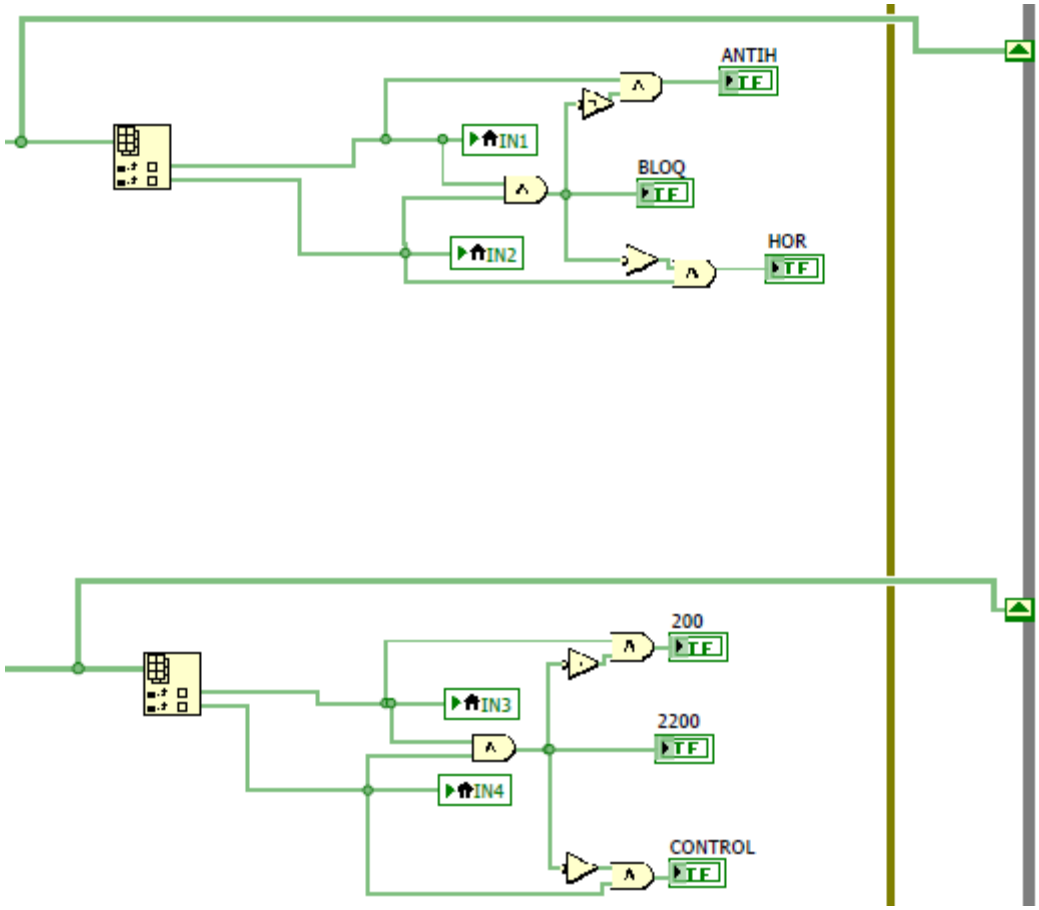


Figura 7.12. Gestión de las salidas digitales

Finalmente y una vez editadas las distintas salidas digitales, este valor actualizado pasará al right shift, de manera que el cambio ejercido al pulsar el botón se mantendrá hasta realizar otro cambio.

Capítulo 8

Presupuesto

En la siguiente tabla aparece expuesto el coste detallado de cada uno de los componentes y el coste total:

Component	Description	Part	References	Value	Footprint	Quantity Per PCB	Precio unidad	Total
1	Polarised capacitor	CP1	C1	100uF	CP_Radial_D10.0mm_P3.50mm	1	1,598	1,598
2	Polarised capacitor	CP1	C4	100uF	CP_Radial_D10.0mm_P5.00mm	1	1,598	1,598
3	Polarised capacitor	CP1	C2	10uF	CP_Radial_D10.0mm_P3.80mm	1	0,533	0,533
4	Polarised capacitor	CP1	C3	10uF	CP_Radial_D10.0mm_P5.00mm	1	0,533	0,533
5	Diode, alternativ symbol	D_ALT	D1 D2 D3	1N4004	D_DO-15_P10.16mm_Horizontal	3	0,071	0,213
6	940nm IR-LED, 5mm	LED	DL1	LED	LED_D5.0mm	1	1,21	1,21
7	Generic connector, single row, 01x02	Conn2	J1	Conn2	JST_NV_B02P-NV_2x5.00mm_Vertical	1	3,14	3,14
8	Generic connector, single row, 01x09	Conn_01x09	J2	Conn_01x09	JST_EH_B09B-EH-A_09x2.50mm_Straight	1	13,89	13,89
9	Generic connector, single row, 01x15	Conn_01x15	J3	Conn_01x15	JST_EH_B15B-EH-A_15x2.50mm_Straight	1	16,61	16,61
10	Resistor	R	R14	154	R_Axial_DIN0411_L9.9mm_D3.6mm_P15.24mm_Horizontal	1	0,53	0,53
11	Resistor	R	R1 R2 R3 R4	1K2	R_Axial_DIN0411_L9.9mm_D3.6mm_P12.70mm_Horizontal	4	0,53	2,12
12	Resistor	R	R13	1K2	R_Axial_DIN0411_L9.9mm_D3.6mm_P15.24mm_Horizontal	1	0,53	0,53
13	Resistor	R	R16	240	R_Axial_DIN0411_L9.9mm_D3.6mm_P15.24mm_Horizontal	1	0,53	0,53
14	Resistor	R	R11 R12	2K2	R_Axial_DIN0411_L9.9mm_D3.6mm_P15.24mm_Horizontal	2	0,53	1,06
15	Resistor	R	R5 R6 R7 R8 R9 R10	2K7	R_Axial_DIN0411_L9.9mm_D3.6mm_P12.70mm_Horizontal	6	0,53	3,18
16	Resistor	R	R15 R17	2K7	R_Axial_DIN0411_L9.9mm_D3.6mm_P15.24mm_Horizontal	2	0,53	1,06
17	Potentiometer	POT	RV1	5K	Potentiometer_Trimmer_Bourns_3296Z	1	3,38	3,38
18	Dual DC Optocoupler, Vce 35V, CTR 50%, DIP-	LTV-827	U3	EL827	DIP-8_W7.62mm	1	0,413	0,413
19	Dual DC Optocoupler, Vce 35V, CTR 50%, DIP-	K847PH	U1	K847PH	DIP-16_W7.62mm	1	1,12	1,12
20	500mA 35V Adjustable Linear Regulator, TO-39	LM117_TO39	U4	LM117	TO-39-3	1	13,56	13,56
21	DC Optocoupler, Vce 80V, CTR 50%,	LTV-358T	U2	VO610A-3X001	DIP-4_W7.62mm	1	0,59	0,59
22	Placa PcB					2	4,68	9,36
23	Caja impresora 3D					2	5	10
24	MyRIO					1	532,8	532,77
25	Motor BLDC					1	1356	1356
26	Desarrollo (jornada)					21	50	1050
						Total IVA		3025,5

Tabla 8.1. Presupuesto

Para verlo en detalle acceda al vínculo ‘presupuesto’ en el icono a continuación.

presupuesto.xls

Capítulo 9

Conclusiones

Hemos demostrado que si es posible hacer funcional un motor brushless programando y con un elemento sbRIO 9636 aun con la problemática del Hardware debido a la salida digital de la que no disponemos. La indisponibilidad de esta salida sin embargo es una limitación que no permite una funcionalidad completaya que nos impide una de las entradas IN1, IN2 IN3 o IN4.

Este problema es solventado utilizando el elemento myRIO 1900, el cual dispone de todas las entradas y salidas digitales necesarias para hacer el motor 100% funcional.

En cuanto a la continuidad a este proyecto, se puede tratar de ejecutar el aprendizaje de velocidades preprogramadas, el aprendizaje de rampas e incluso también se pueden implementar sistemas PID de control motor. En ambos casos sería necesario realizar una programación adicional en LabView que se puede ejecutar sobre el sistema embebido myRIO 1900.

Finalmente y debido a la fácil accesibilidad y programación de LabView el resultado de este proyecto se presenta como un buen ejemplo para enseñar a los alumnos de la EII el funcionamiento de un motor brushless y su control y posibilidades con un sistema embebido de control sencillo y eficaz programado con LabView.

Capítulo 10

Bibliografía

Cruz, J. M., & Lutenberg, A. (2012). Introducción general a los sistemas embebidos. *De SASE, Buenos Aires*,

Del Rio, J., Manuel, A., Sarria, D., & Shariat, S. (2011). *Labview: Programación para sistemas de instrumentación* Ibergarceta Publicaciones SL.

García Haro, J. M. (2011). *Desarrollo De Un Controlador Para Motores DC Brushless Basado En CompactRIO y LabVIEW De National Instruments Para El Estudio De Nuevos Algoritmos De Control*,

García Haro, J. M. (2011). *Desarrollo De Un Controlador Para Motores DC Brushless Basado En CompactRIO y LabVIEW De National Instruments Para El Estudio De Nuevos Algoritmos De Control*,

Hanselman, D. C. (2003). *Brushless permanent magnet motor design* The Writers' Collective.

Hendershot, J. R., & Miller, T. J. E. (2010). *Design of brushless permanent-magnet machines* Motor Design Books.

Ihra, R. (2000). Optoacopladores para aplicaciones de potencia. *Revista Espanola De Electrónica*, (551), 74-76.

Ihra, R. (2000). Optoacopladores para aplicaciones de potencia. *Revista Espanola De Electrónica*, (551), 74-76.

Kosow, I. L. (1993). *Máquinas eléctricas y transformadores* Pearson Educación.

Krause, K. W. (1996). *Brushless Motor*,

Krishnan, R. (2017). *Permanent magnet synchronous and brushless DC motor drives* CRC press.

Lawman, G. R. (2000). *Configuring an FPGA using Embedded Memory*,

Lienhardt, A., Gateau, G., & Meynard, T. A. (2007). Digital sliding-mode observer implementation using FPGA. *IEEE Transactions on Industrial Electronics, 54*(4), 1865-1875.

Lobosco, O. S., da Costa Dias, José Luis Pereira, & Oliver, D. (1990). *Selección y aplicación de motores eléctricos* Marcombo.

Lobosco, O. S., da Costa Dias, José Luis Pereira, & Oliver, D. (1990). *Selección y aplicación de motores eléctricos* Marcombo.

Martínez, J. M. M., & Buendía, M. J. (2010). *Programación gráfica para ingenieros* Marcombo.

Miller, T. J. E. (1989). Brushless permanent-magnet and reluctance motor drives.

National, I. (05/2016). User guide and specifications ni myrio-1900. Retrieved from http://www.ni.com/pdf/manuals/376047c.pdf

National, I. (08/2012). NI LabVIEW RIO evaluation kit [tutorial]. Retrieved from ftp://ftp.ni.com/pub/devzone/tut/ni_labview_rio_eval_tutorial.pdf

National, I. (12/08/2012). OEM OPERATING INSTRUCTIONS AND SPECIFICATIONS NI sbRIO-9605/9606 and NI sbRIO-9623/9626/9633/9636. Retrieved from http://www.ni.com/pdf/manuals/373378c.pdf

Pedre, S. (2012). Sistemas embebidos. *Laboratorio De Robotica y Sistemas Embebidos.Departamento De Computación FCEN,*

Pfendler K. (01/2008). Instruction manual / betriebsanleitung BLDC-motor with integrated speed controller EC-motor mit integriertem drehzahlregler BG 44 SI. Retrieved from http://www.microprivod.ru/files/IM/dunker/IM_BG44SI.pdf

Vega, J. I. H. (2010). El software embebido y los retos que implica su desarrollo. *Conciencia Tecnológica,* (40), 42-45.

Vitores, R. S. (2004). Aplicaciones de los sistemas embebidos. *Técnica Industrial,* (1), 24-27.

Vizcaíno, J. R. L., & Sebastiá, J. P. (2011). *LabVIEW: Entorno gráfico de programación* Marcombo.

Vizcaíno, J. R. L., & Sebastiá, J. P. (2011). *LabVIEW: Entorno gráfico de programación* Marcombo.

Wong, H., Cheng, L., Lin, Y., & He, L. (2005). FPGA device and architecture evaluation considering process variations. Paper presented at the *ICCAD-2005. IEEE/ACM International Conference on Computer-Aided Design, 2005.* 19-24.

Heartyl, G. (12/08/2012). The ultimate box maker designs. Retrieved from http://heartygfx.blogspot.com/2016/01/the-ultimate-box-maker.html