

Sistema para el control inteligente de coches de Scalextric

Guizán Carballeira, Rodrigo
DNI 33340356 Q
ESEI de Ourense

INTRODUCCIÓN	4
1. ANÁLISIS DE REQUISITOS.....	5
1.1 ANÁLISIS DE REQUISITOS HARDWARE.....	5
1.2 ANÁLISIS DE REQUISITOS SOFTWARE.....	6
2. FASE DE ANÁLISIS.....	8
2.1 ANÁLISIS DE HARDWARE	8
2.1.1 Sensores.....	8
2.1.2 Botones.....	13
2.1.2 LCD	13
2.1.2 Alimentación.....	14
2.1.5 Microcontrolador.....	14
2.1.6 Cableado	15
2.1.7 Reutilización de materiales.....	15
2.1.8 Proceso de extracción de los sensores.....	18
2.2 ANÁLISIS DE SOFTWARE	21
2.2.1 Diagrama general.....	22
2.2.2 Diagrama de calibración.....	24
2.2.3 Diagrama de calibración de los sensores de la pista 2	25
2.2.4 Diagrama de Juego.....	27
3. FASE DE DISEÑO	30
3.1 DISEÑO DE HARDWARE	30
3.1.1 Sensores Hall.....	30
3.1.2 Cableado del Scalextric.....	33
3.1.3 Power MOSFET.....	33
3.1.4 Microcontrolador.....	36
3.1.5 Alimentación.....	36
3.1.6 Diseño Lógico del circuito	37
3.1.7 PCB.....	43
3.2 DISEÑO DE SOFTWARE.....	47
3.2.1 Configuración de los registros.....	47
3.2.2 Diagrama de flujo para la selección de nivel.....	52
3.2.3 Diagrama de flujo para la selección del número de vueltas	54
3.2.4 Diagrama de flujo para el procedimiento decisión	56
3.2.5 Diagrama de flujo para el procedimiento acelerar.....	58
3.2.6 Diagrama de flujo para el procedimiento decelerar	60
3.2.7 Diagrama de flujo para el procedimiento recalibrar	62
3.2.8 Diagrama de flujo para el procedimiento interrupt	65
3.2.8 Diagrama de flujo para la detección de interrupciones	69
3.2.9 Diagrama de flujo para la cuenta atrás.....	71
3.2.11 Diagrama de flujo para el programa principal.....	72
4. FASE DE IMPLEMENTACIÓN DE SOFTWARE.....	76
4.1 IMPLEMENTACIÓN PARA LA SELECCIÓN DE NIVEL.....	76
4.2 IMPLEMENTACIÓN PARA LA SELECCIÓN DEL NÚMERO DE VUeltas	77
4.3 IMPLEMENTACIÓN DEL JUEGO	78
4.3.1 Implementación del procedimiento decisión	78
4.3.2 Implementación del procedimiento acelerar.....	78
4.3.3 Implementación del procedimiento decelerar	79
4.3.4 Implementación del procedimiento recalibrar	79
4.3.5 Implementación del procedimiento Interrupt.....	80
4.4 IMPLEMENTACIÓN DEL PROCEDIMIENTO CONFIGURACIÓN.....	82
4.5 IMPLEMENTACIÓN DEL PROGRAMA PRINCIPAL.....	83
4.6 VARIABLES GLOBALES	86
5. FASE DE PRUEBAS.....	87

5.1. CONSTRUCCIÓN DE UN PROTOTIPO	87
5.2 PRUEBAS DE SIMULACIÓN POR SOFTWARE.....	89
<i>5.2.1 Prueba para la selección de Nivel.....</i>	<i>91</i>
<i>5.2.3 Prueba para la selección del número de vueltas.....</i>	<i>92</i>
<i>5.2.3 Prueba para el Juego</i>	<i>93</i>
6. CONSTRUCCIÓN PLACA DE CIRCUITOS FINAL	101
7. CONSTRUCCIÓN DE LA MAQUETA.....	102
8. PRUEBA FINAL	103
9. DESAJUSTES Y ERRORES COMETIDOS	105
10. CONCLUSIÓN.....	107
11. FUTURAS AMPLIACIONES.....	108
11.1 SISTEMA IMBATIBLE.....	108
11.2 JUGADOR FANTASMA	108
11.3 TESTEO DEL ESTADO DE LOS SENORES.....	108
11.4 RECUPERACIÓN DE LA ESCALABILIDAD	108
12. MANUAL DE USUARIO	110
12.1 DESCRIPCIÓN DE LA INTERFAZ.....	110
12.2 MONTAJE	111
12.3 Uso	115
12.4 ADVERTENCIAS	117
13. GLOSARIO	118
13. BIBLIOGRAFÍA.....	121
14. CONTENIDO DEL CD.....	122

Introducción

Scalextric es un juego escalable basado en las competiciones de automoción. En él, se emplean reproducciones de vehículos a pequeña escala y de tracción eléctrica que circulan por una vía fija.

El juego surgió en Inglaterra en 1952 de la mano de Fred Francis. Este inventó fue dado a conocer al público en la feria anual de juguetes de Harrogate en 1957, dónde se pudo apreciar la primera muestra de coches de Scalextric. Los primeros modelos fueron fabricados en metal, siendo sustituidos poco después por coches de plástico.

En 1962, la compañía inglesa Lines Bros Ltd., que por entonces poseía la patente de Scalextric, llegó a un acuerdo con Exin para comercializar sus productos en la Península Ibérica. Cabe destacar que los modelos fabricados en España han sido considerados los mejores a escala mundial hasta 1992, año en el que la empresa quebró. Tras la desaparición de Exin, los derechos de fabricación pasaron primero a Tyco, en 1993, y posteriormente a TecnyToys (1998), fabricante actual.

Hoy en día, los videojuegos han supuesto un gran escollo para la mayoría de los juguetes clásicos, entre los que se encuentra el Scalextric. Estos juegos han ido perdiendo posiciones en el mercado actual. Esta forma de juego, aunque menos plausible, resulta más cómoda al poder jugar en solitario sin que disminuya la diversión.

Por este motivo, se pensó que sería posible relanzar este producto clásico de forma que una sola persona pueda interaccionar con él de una manera atractiva.

Por ello, se desarrollará el “Sistema para el control inteligente de coches de Scalextric”. La propuesta es construir un “Scalextric” en el que una persona pueda competir contra el sistema seleccionando previamente el nivel de dificultad deseado, de forma que el jugador controle un coche mientras el sistema controla automáticamente el coche secundario. De esta forma será posible jugar en solitario tal como ocurre con los videojuegos.

1. Análisis de requisitos

Para realizar el análisis de este proyecto y obtener tanto los requerimientos de software como de hardware se ha decidido que la mejor opción posible sería estudiar el comportamiento del coche sobre la pista del Scalextric. Para poder obtener el comportamiento de forma medible se ha utilizado un polímetro, pudiendo observar en todo momento la variación del voltaje y, por tanto, del comportamiento del coche.

Una vez conectado se ha procedido a jugar unas partidas pudiendo captar así la velocidad adecuada en cada punto. Todo ello a sido grabado en vídeo para facilitar más tarde la toma de datos.

Una vez obtenido el comportamiento del coche de forma empírica, se ha podido empezar tanto con el análisis de software como de hardware.

1.1 Análisis de requisitos hardware

El primer reto a vencer del diseño, es que se debe desarrollar uno que nos permita conocer cuál es la posición del coche en las curvas, puesto que en las rectas el coche no varía su comportamiento, simplemente circula a una velocidad constante hasta que se debe aminorar para tomar la curva.

Para ello, es evidente que se ha de utilizar algún tipo de sensor. Tanto el número como el tipo de los mismos serán estudiados más a fondo durante la fase de análisis de hardware.

Teniendo en cuenta que el sistema no será usado por ningún personal técnico, se debe diseñar una interfaz de usuario amigable por lo que el sistema deberá contener una pantalla LCD, para facilitar la comunicación con el usuario, y unos botones con los que el usuario pueda comunicarse con la máquina.

El microcontrolador, será el centro neurálgico del sistema. Se ha determinado que será de la familia 18LF. Por una parte, para asegurar que el tamaño de la memoria será más que suficiente y por otra, de aspectos más físicos, para que el sistema funcione con tensión baja. No obstante, aún no es posible concretar qué microcontrolador de esta familia será utilizado pues, primero se deberá conocer si los sensores que se utilizarán serán analógicos o digitales y, dependiendo de esto, se determinará el número de entradas analógico/digital que debe contener dicho chip.

Por último, y no menos importante es la alimentación del hardware. Éste se puede obtener tanto del Scalextric como de una fuente externa, una batería o un transformador. En función del tamaño de la placa deseada se estudiará cuál es más factible y práctica.

1.2 Análisis de requisitos software

La característica más destacable del software es que garantice que el programa a desarrollar responda en un tiempo finito. Es decir, que cuando el coche esté entrando en una curva sea capaz de disminuir la velocidad en ese mismo instante ya que de lo contrario el vehículo se acabaría saliendo del trazado. Para ello el software ha de ser lo más simple posible y se han de utilizar interrupciones que permitan al microcontrolador responder inmediatamente.

En cuanto a los requisitos funcionales podemos destacar tres claros objetivos:

Niveles de dificultad

El primer objetivo que debe cumplir este proyecto es que el sistema permitirá jugar con tres niveles de dificultad: fácil, medio y difícil.

Para ello, el sistema dispondrá de un botón de selección y otro de inicio, así como una pequeña pantalla LCD que permitirá la comunicación visual con el usuario.

En primer lugar, aparecerá un primer mensaje en la pantalla LCD solicitando al usuario que seleccione el nivel de dificultad. Para ello, deberá ir pulsando el botón de selección hasta obtener en la pantalla el nivel deseado. Para confirmar, será necesario pulsar el botón de inicio.

Número de vueltas por juego

El segundo objetivo a cumplir es que el sistema permita seleccionar el número de vueltas que el jugador desea efectuar. Para ello, el jugador deberá pulsar el botón de selección hasta obtener en la pantalla el número de vueltas deseado. Una vez obtenido el número de vueltas, el jugador deberá pulsar el botón de inicio.

Juego

El tercer objetivo de este sistema es dotarlo de la posibilidad de juego.

Una vez que el usuario haya confirmado el nivel y el número de vueltas, comenzará el juego. Una pequeña cuenta atrás de 3 segundos proporcionará al usuario el tiempo necesario para prepararse para la carrera. Una sucesión descendente de números desde el 3 al 0 irán apareciendo en la pantalla LCD para alertar al usuario.

Dentro del desarrollo de la carrera el sistema se encargará de:

- Controlar el coche secundario (el que no maneja el usuario).
- Contabilizar el número de vueltas de ambos coches (que se irán reflejando en la pantalla LCD).

Cuando uno de los coches complete las vueltas, terminará la carrera y en la pantalla LCD indicará si se ha ganado o perdido.

 **Reset**

El cuarto y último objetivo será dotar al sistema de la posibilidad de reinicio.

2. Fase de Análisis

Durante esta etapa se estudiarán las diversas posibilidades que ofrecen los distintos componentes existentes en el mercado para el diseño de hardware.

Desde el punto de vista del software se realizarán los primeros diseños que más tarde serán desglosados y completados, en caso de ser necesario, durante la etapa de diseño.

2.1 Análisis de hardware

En esta parte se estudiarán los diferentes componentes necesarios para la construcción de la parte hardware relativo al sistema. Este apartado se dividirá en sensores, microcontrolador, botones, LCD, cableado y alimentación, que a grandes rasgos serán la parte más importante del mismo.

2.1.1 Sensores

Tras el visionado y estudio de los vídeos obtenidos durante la fase de requerimientos, se puede observar como el coche solamente varía su velocidad tanto en la entrada como en la salida de las curvas, mientras que en las rectas la velocidad es constante.



Entrada en la curva



Mitad de la curva tras la entrada



Mitad de la curva antes de la salida

Debido a este comportamiento, se ha decidido utilizar cuatro sensores en cada curva, de forma que uno sirva para cuando el coche entra en la curva frenando, el segundo para que siga frenando justo cuando la curva es más cerrada, el tercero para que empiece a acelerar y el cuarto para que acelere al máximo justo antes de entrar en la recta, donde mantendrá una velocidad constante.

Como el circuito tiene dos curvas y dos rectas, solamente harían falta ocho sensores.

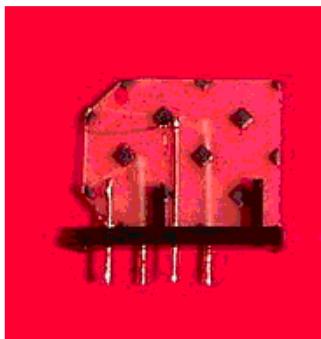
No obstante, y degradando la calidad del sistema de forma aceptable, este sistema podría construirse con un mínimo de tres sensores por curva de forma que uno a la entrada de la misma indique que el coche debe decelerar, otro en la mitad para que empiece a acelerar y el tercero, al final de la misma, indicando que el coche puede acelerar al máximo pues a continuación está la recta. Con esto, se reduciría en 2 sensores el coste del proyecto. Esta posibilidad no debe descartarse por el momento pues todo depende del valor económico de los mismos.

También se ha de tener en cuenta que es necesario el uso de dos sensores más que permitirán contar las vueltas de cada coche. Estos deberán ir colocados en la recta inferior, ya que sería ilógico comenzar el juego en una curva.

Por lo tanto, para la construcción del sistema se ha determinado que el número ideal de sensores debe ser 10. Para una mayor comodidad a la hora de diseñar e implementar el hardware es recomendable que los sensores sean digitales, pues resultan menos críticos que los analógicos.

Como en el mercado existen varios tipos de sensores, a continuación se realizará un breve estudio atendiendo a sus capacidades para la medición de distintas magnitudes.

● Sensores piezoeléctricos



Sensor sin cápsula

La piezoelectricidad es un fenómeno presentado por determinados cristales que al ser sometidos a tensiones mecánicas adquieren una polarización eléctrica en su masa, apareciendo así, una diferencia de potencial en su superficie.

Estos sensores pueden estar formados por cristales naturales, como el cuarzo o la turmalina, o sintéticos, también conocidos con el sobrenombre de ferro-eléctricos.

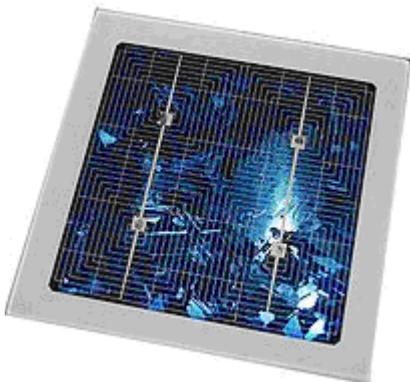
Dentro de las aplicaciones cotidianas destacan su uso en los mecheros electrónicos. Estos llevan en su interior un cristal piezoeléctrico que es golpeado de forma brusca por el mecanismo de encendido, provocando así, una elevada concentración de carga eléctrica capaz de producir la chispa.

Otro uso muy conocido es el de sensor de vibración, el cuál es aprovechado para la construcción de guitarras eléctricas.

De cara al proyecto que se quiere realizar, este sensor ha sido descartado por dos motivos:

- El primero es que este sensor es analógico, con lo que el PIC debería contar con entradas que conviertan la señal analógica en digital para poder trabajar con ella.
- El segundo y más importante es que su colocación en la pista sería una tarea ardua, pues habría que perforar la pista con una profundidad determinada, la suficiente como para que la rueda del coche pase por encima sin que se produzca un bache.

● Células fotoeléctricas



Célula fotoeléctrica

Una célula fotoeléctrica, también denominada fotocélula o celda fotovoltaica, es un dispositivo electrónico que permite transformar la energía luminosa en energía eléctrica.

Estas células están compuestas por un material con propiedades fotoeléctricas, de forma que absorben fotones de luz y emiten electrones. Estos pueden ser capturados para crear una corriente eléctrica que puede ser utilizada como señal.

Las células se pueden agrupar en lo que se denomina panel fotovoltaico. Este tipo de sensores se han descartado por dos motivos:

- El primero, es el mismo que el anterior, la señal que generan es analógica.
- El segundo es que dependen de un factor ambiental, en este caso concretamente de que haya luz, con lo que en caso de proyectarse alguna sombra del jugador sobre la pista el sensor dejaría de funcionar correctamente.

● Fotodiodos



Fotodiodo

Un fotodiodo es un semiconductor sensible a la incidencia de la luz visible o infrarroja.

Cuando la luz infrarroja llega al diodo se excita un electrón dándole movimiento y creando un hueco con carga positiva.

Los materiales empleados en la composición de un fotodiodo varían en función de la longitud de onda con la que tienen que trabajar. Los más comunes son los de silicio empleados con luz visible (longitud de onda de hasta 11 μ m) y de germanio para luz infrarroja (longitud de onda hasta 1,8 μ m).

Dentro del mundo electrónico destacan su uso en los lectores de CD, recuperando la información grabada en el surco del CD transformando la luz del haz de láser reflejada en el mismo en impulsos eléctricos que serán procesados por el sistema.

Este sensor sería viable para la realización de este proyecto, no obstante ha sido descartado por la siguiente razón:

- Sería necesario equipar a ambos coches con un emisor de infrarrojos y con una fuente de alimentación independiente. Es decir, se tendría que perforar ambos vehículos para situar un emisor cuyo haz se cuele entre los raíles de la pista, donde estarían situados los receptores, y habría que colocar una batería en el interior de cada coche para alimentar sendos emisores. Con esto se aumentaría el coste de producción y se aumentaría la dificultad de realización del proyecto.

● Sensores de efecto Hall



Sensor hall

El sensor de efecto Hall, o simplemente sensor Hall o sonda Hall, se sirve del efecto Hall para la medición de campos magnéticos.

Si fluye corriente por un sensor Hall y se aproxima a un campo magnético cuya dirección es vertical al sensor, entonces se crea un voltaje saliente proporcional al producto de la fuerza del campo magnético y de la corriente.

Uno de sus usos más importantes es el de funcionar como emisor de señales sin contacto. Así, por ejemplo, en la industria del automóvil este sensor es utilizado en los cierres de cinturones de seguridad, en sistemas de cierres de puertas, para el reconocimiento de la posición del pedal... La gran ventaja de estos sensores es la invariabilidad frente a la suciedad (no magnética) y el agua.

Además puede encontrarse en circuitos integrados de impresoras láser, disqueteras de ordenador o en ventiladores de PC.

Teniendo en cuenta que los coches de Scalextric tienen imanes, tanto en el motor del coche como en la parte inferior del mismo, este sensor es el idóneo dentro de los anteriormente descritos, y se decantará por su uso.

2.1.2 Botones



Botones

Un botón o pulsador es un dispositivo utilizado para activar alguna función. Estos son de diversa forma y tamaño y se encuentran en todo tipo de dispositivos, aunque principalmente en aparatos electrónicos.

Estos dispositivos funcionan, por norma general, como un interruptor eléctrico, es decir, en su interior tienen dos contactos: NA (normalmente abierto) o NC (normalmente cerrado). De este modo al pulsarlo se activará la función inversa de la que en ese momento esté realizando el botón.

Para este proyecto hay que tener en cuenta el denominado Efecto Rebote. Este se produce debido a que los botones son dispositivos mecánicos y en su interior tienen un muelle para retornar a la posición inicial, lo que conlleva que tras pulsar y soltar el mismo es posible que se generen varias señales.

Existen una variante de botones y pulsadores antirrobote. No obstante este problema puede ser subsanado mediante la programación, por lo tanto la elección de un tipo u otro vendrá determinado principalmente por el coste y el tamaño de los mismos.

2.1.2 LCD



Display

Una pantalla de cristal líquido o LCD, es una pantalla delgada y plana formada por un número de píxeles, en color o monocromos, colocados delante de una fuente de luz o reflectora. La ventaja de estas pantallas es que utilizan cantidades muy pequeñas de energía eléctrica.

En este sistema la pantalla es un elemento importante, ya que será la encargada de permitir la comunicación con el usuario, aunque tampoco tiene que ser excesivamente completa. No es necesaria una pantalla en color, por lo que se ha decidido utilizar una pantalla monocroma. Dentro de esta gama existen principalmente dos variantes. Unas que tienen el fondo de color azul y

otras cuyo fondo es verde. Se ha acordado el uso de esta última pues con un coste similar posee una visibilidad superior.

Destacar la importancia de que tenga dos filas de caracteres ya que, debido a los requerimientos del sistema, es imperante que muestre las vueltas tanto del jugador como del vehículo controlado, resultando más legible con el empleo dos filas.

2.1.2 Alimentación

Como ya se ha mencionado anteriormente, la alimentación del sistema puede obtenerse de dos formas:

- El sistema se alimentará conjuntamente con el Scalextric. Para ello es necesario emplear un dispositivo capaz de transformar los 12 voltios del transformador del Scalextric en 5. Es decir, un regulador de voltaje, más conocido como 7805, como los que muestra la siguiente imagen.



- EL sistema se alimentará independientemente del Scalextric. Para ello es necesario utilizar un transformador y enchufarlo a la red eléctrica de forma independiente.

Mencionar que cabe la posibilidad de obtener algún transformador con coste cero para el proyecto reutilizando alguno de otro aparato obsoleto, como un móvil viejo, algún MODEM... En cualquier caso, antes de realizar el diseño del hardware y en función de las piezas obtenidas, se optará por el uso de uno u otro dispositivo.

2.1.5 Microcontrolador

El microcontrolador es una de las partes esenciales de este proyecto. Finalmente se ha decidido utilizar el microcontrolador 18LF452.

Una de las razones que han llevado a la elección de este microcontrolador y no otro, es que es un microcontrolador de gama alta muy utilizado en la industria y, por lo tanto, existe mucha documentación al respecto.

Además, el microcontrolador dispone de 32KB de memoria, con lo que queda garantizado que el software a desarrollar no tendrá problemas con el tamaño de la memoria. No obstante, una vez determinado el tamaño del programa, a la hora de comercializar este sistema el fabricante podría optar por un microcontrolador de gama media logrando así reducir costes.

Otra razón de peso para la elección del 18LF452 es que dispone de 8 entradas analógicas, por lo que en caso de que los sensores fueran

analógicos se implementaría el sistema siguiendo el diseño de los tres sensores en cada curva, siendo todavía posible la construcción de éste sin sufrir demasiadas alteraciones en el calendario previsto.

Este microcontrolador dispone de dos módulos de PWM o control de ancho de pulso. El PWM será necesario para poder aumentar y disminuir la tensión de la pista variando con ello la velocidad del coche.

Destacar que este microcontrolador dispone de tres entradas de interrupciones externas. Es decir, que el microcontrolador es capaz de generar una interrupción cuando le llegue una señal a una de estas patillas, lo que es estrictamente necesario para garantizar que el sistema pueda responder inmediatamente y variar la velocidad del coche.

Por último, comentar que dispone de cuatro tipos de Timer, que permiten generar interrupciones con una periodicidad previamente definida, permitiendo por tanto controlar el tiempo.

En cuanto a la forma del chip será de los denominados DIP, ya que permiten más facilidad de montaje en una placa de forma manual que sus otros hermanos.

2.1.6 Cableado

Puesto que los sensores deben alimentarse con corriente y el microcontrolador necesita recibir las señales emitidas de estos, es indudable que el Scalextric deberá ir cableado por debajo.

Tras estudiar diversas opciones, la mejor y más económica es la utilización de cable de bus ya que es flexible y de fácil instalación.

2.1.7 Reutilización de materiales

Para reducir los costes del proyecto, se ha intentado reciclar una parte del hardware.

Como se ha mencionado anteriormente, los sensores hall se encuentran en equipos cotidianos de los que pueden ser extraídos.

Se han estudiado tres posibilidades: las dos primeras referentes a los sensores y la tercera referente a la posible alimentación de la placa.

Disqueteras

Hoy en día las disqueteras están prácticamente obsoletas, por lo que encontrar equipos en desuso que tengan este dispositivo es relativamente fácil.

En éstos periféricos, el sensor hall es utilizado para controlar en todo momento la velocidad del motor que hace girar el disquete. Esto es debido a que los disquetes con el tiempo se van endureciendo y por lo tanto la fuerza que debe ejercer el motor para alcanzar la misma velocidad varía.

- Disqueteras 3 ½

Tras desmontar y obtener el sensor hall de la disquetera se ha comprobado que la manipulación del mismo resulta prácticamente imposible porque es excesivamente pequeño. Por lo tanto la obtención de los sensores de este dispositivo queda totalmente descartada.

- Disqueteras 5 ¼

Tras el fracaso obtenido en la búsqueda anterior se ha intentado buscar disqueteras más viejas ya que al estar implementadas con una tecnología inferior, tendrán sensores hall de mayor tamaño.

Los sensores obtenidos serían válidos para el desarrollo del proyecto siempre y cuando se permita una pequeña degradación del sistema, pues son de carácter analógico.

No obstante conseguir este tipo de dispositivos hoy en día resulta casi imposible por lo que esta opción debe quedar descartada.

Ventiladores

En los equipos informáticos se encuentran numerosos ventiladores que ayudan en la refrigeración del mismo. En estos dispositivos, el sensor hall es utilizado para controlar las revoluciones a las que deben girar dichos ventiladores.

- Ventiladores de procesadores

Los sensores de los ventiladores que traen algunos procesadores, como por ejemplo los de los portátiles tienen la misma desventaja que los de las disqueteras de 3 ½, es decir, son demasiado pequeños para su manipulación. Por lo que indudablemente deben quedar descartados.

- Ventiladores de las fuentes de alimentación

Los sensores obtenidos de los ventiladores de las fuentes de alimentación son perfectamente viables para el desarrollo del proyecto. Tienen la ventaja de que son digitales, lo que permitiría aplicar el desarrollo más eficiente (cuatro sensores en cada curva).

Son fáciles de conseguir pues existen numerosas fuentes de alimentación en equipos obsoletos.

Hay que recalcar que no todos los ventiladores sirven, pues algunos sensores funcionan como biestables, lo que provocaría que el sensor quedase en estado activo hasta que el vehículo volviera a desactivarlo tras dar otra vuelta.

Durante la fase de diseño de hardware se explicará más detalladamente como son estos sensores y como serán utilizados.

Alimentación

Inicialmente, se barajaron dos posibilidades para llevar a cabo la alimentación de la placa.

Se ha conseguido un transformador que proporcionan la tensión y el amperaje necesarios enchufándolo a una corriente alterna de 220 voltios. Por lo tanto, el sistema se diseñará de forma que la alimentación sea independiente, descartando así el uso del regulador de voltaje.

2.1.8 Proceso de extracción de los sensores

En este apartado se tratará de ilustrar la forma utilizada para la extracción de los sensores.

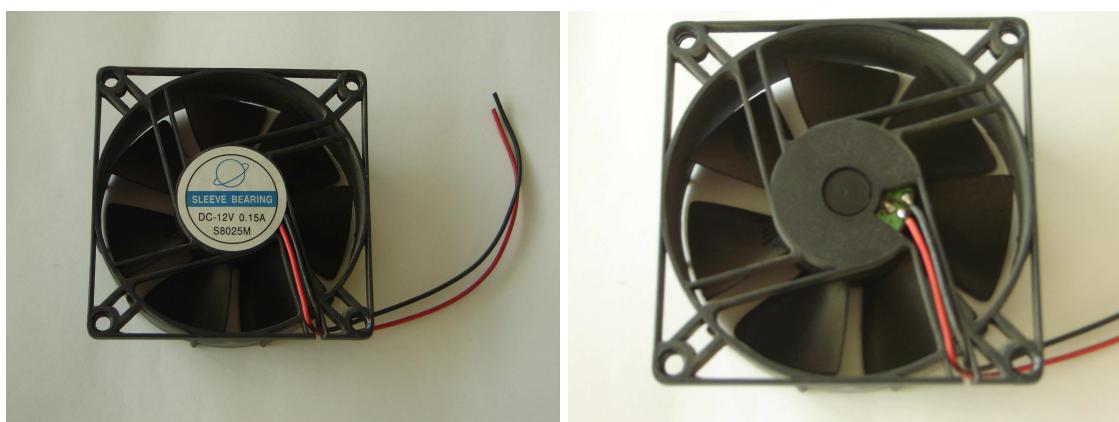


Los materiales empleados para la extracción son:

- Ventilador
- Destornillador
- Alicantes de punta plana
- Estañador de 15 W
- Base para componentes torneada (tira de pines)
- Malla de soldar (no es estrictamente necesaria).

Pasos a seguir

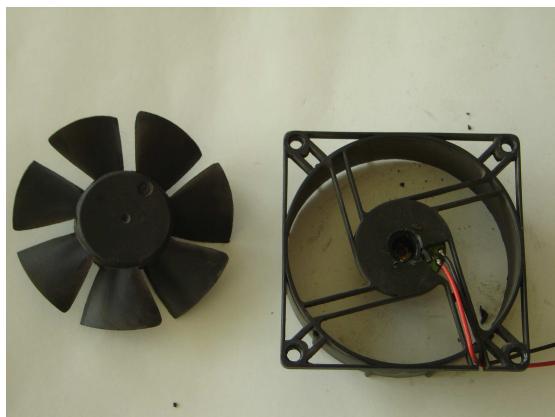
I. Quitar el precinto de seguridad del ventilador para acceder a la tapa.



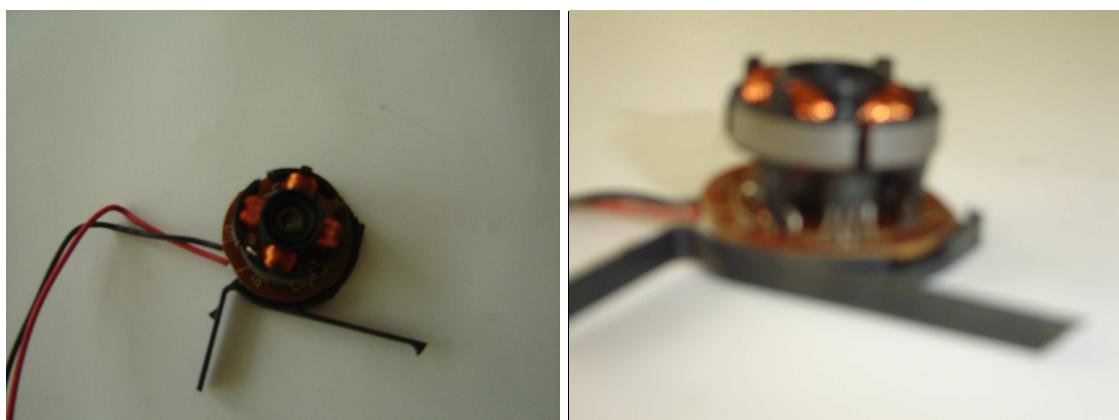
II. Quitar la tapa para obtener acceso a la arandela de seguridad. Después, utilizando el destornillador se debe romper y extraer esa arandela para poder separar las aspas del eje del motor.



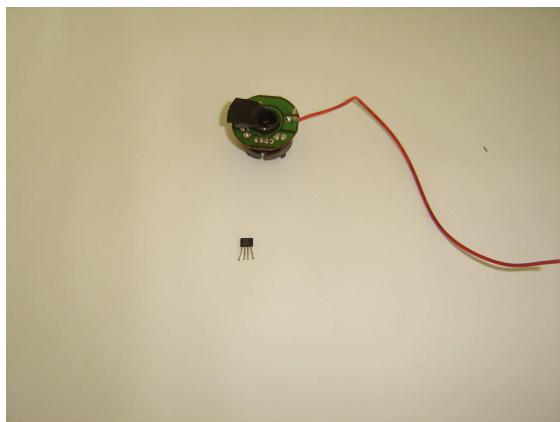
III. Separar las aspas del eje del motor



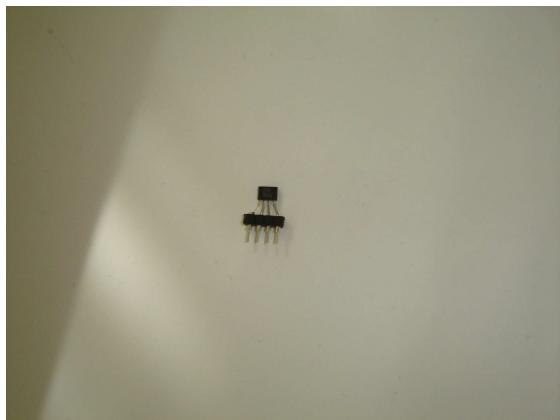
IV. Extraer el motor junto con la circuitería. Para ello basta con romper la carcasa con cuidado para no dañar el silicio y el sensor.



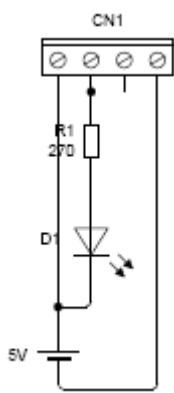
V. Aplicar calor para derretir el estaño (puede utilizarse la malla) y extraer el sensor.



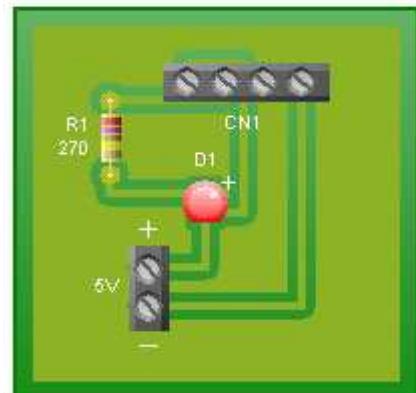
VI. Montar el sensor en la base para componentes torneada y soldarlo. Se recomienda la utilización de los alicates para la sujeción del sensor debido al calor producido por el estañador.



VII. Comprobar el estado del sensor. Para ello, conectar el sensor en una protoboard siguiendo este esquema y observar que el LED funciona al interactuar el sensor con un imán.



Esquema lógico



Esquema físico

2.2 Análisis de software

Debido a que este proyecto tiene un alto porcentaje de desarrollo de hardware, el análisis de software se ha tenido que llevar a cabo paralelamente con el diseño de hardware complementándose así el uno al otro.

Esto implica que no es posible concretar qué patillas del microcontrolador serán utilizadas. Para evitar que esta parte sufra retrasos se empleará pseudocódigo para denominar estas señales.

Así, por ejemplo, se utilizará *pulsar start* y no se concretará ni que puerto ni que patilla será la encargada de recibir esta señal.

Una vez terminado el diseño de hardware se podrá transcribir sin ningún problema este pseudocódigo al código concreto durante la fase de diseño de software.

Para el análisis de software se ha decidido la utilización de diagramas de flujo empleando la herramienta Microsoft Visio ya que el sistema se desarrollará empleando el lenguaje C. Estos diagramas se irán descomponiendo según sea necesario profundizar en los problemas que se puedan presentar.

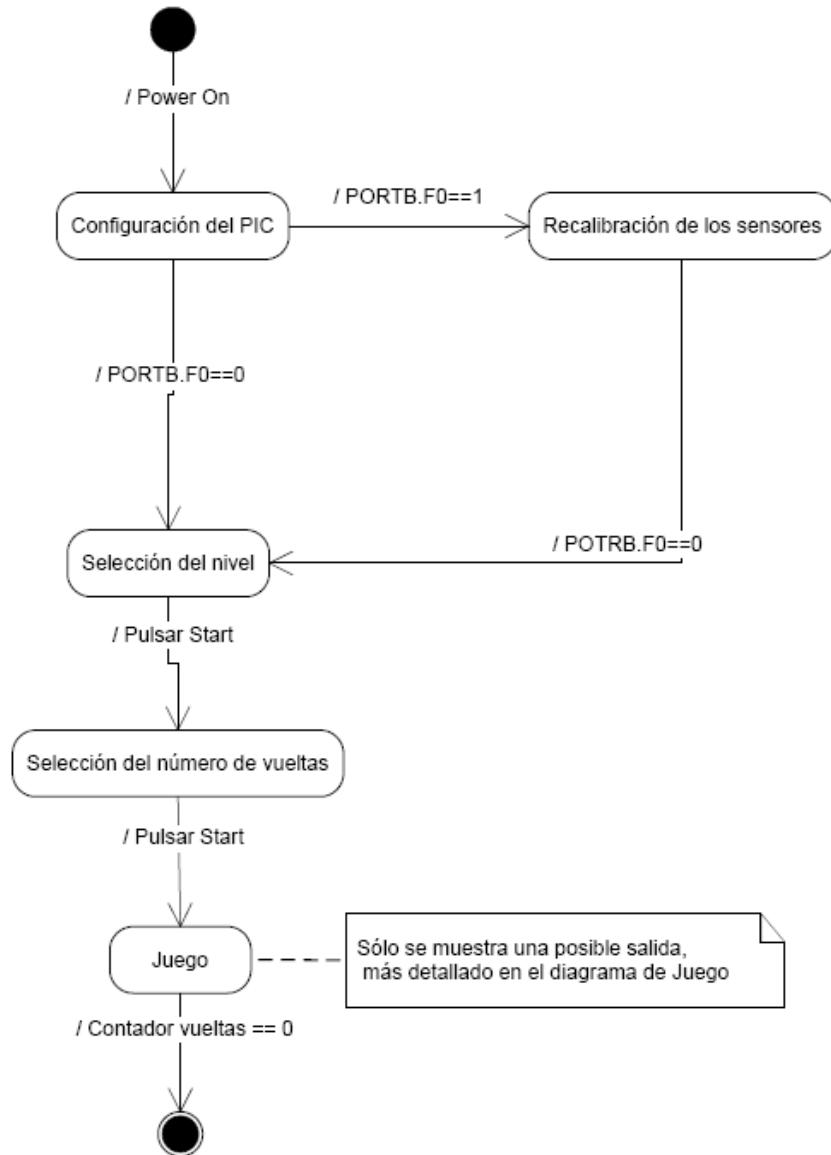
Existen cuatro requerimientos claros que el sistema debe cumplir:

- I. Posibilidad de seleccionar nivel
- II. Posibilidad de selección de nº de vueltas
- III. Posibilidad de Juego
- IV. Posibilidad de reinicio o reseteo

Una vez estudiados los requerimientos iniciales, se ha llegado a la conclusión de que el requerimiento de reseteo será realizado mediante hardware, ya que el microcontrolador dispone de una patilla capaz de realizar esa función.

Los otros tres requerimientos se desarrollarán de forma secuencial de manera que el sistema permita: primero, seleccionar el nivel deseado; después, seleccionar el número de vueltas de la carrera y finalmente, el juego en sí.

2.2.1 Diagrama general



El sistema se activará tras accionar el interruptor a ON y así alimentarlo.

Lo primero que deberá hacer el sistema es configurar el PIC, es decir, configurar las entradas, las interrupciones, el modo de funcionamiento... Este apartado se podrá concretar una vez terminado por completo el diseño de hardware durante la fase de diseño de software.

PORTB.F0 representa la señal generada por cualquier sensor, por lo tanto, será la señal que deberá activar las interrupciones del microcontrolador para que este pueda responder acelerando o decelerando el vehículo según sea conveniente. En este primer diagrama esta señal servirá para determinar el estado de los sensores.

Los sensores hall, al ser de tipo magnéticos, quedan orientados en uno u otro sentido. Esto quiere decir que inicialmente se desconoce si estos sensores están abriendo o cerrando el circuito eléctrico. Por lo tanto, en caso de estar recibiendo una señal por esta entrada, querrá decir que hay que

calibrarlos. La forma más sencilla de hacerlo es que el coche de una vuelta y así oriente los sensores de forma correcta.

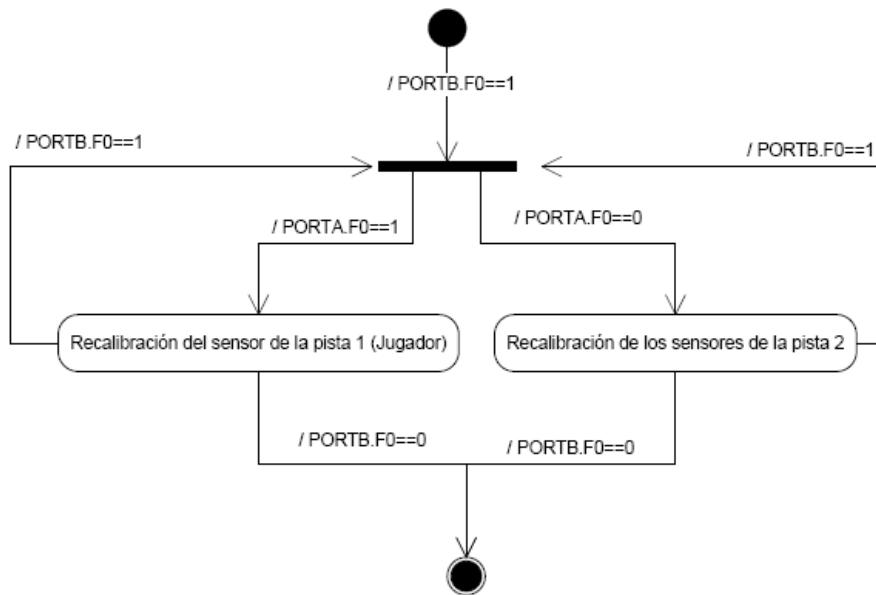
Una vez el sistema esté configurado y calibrado, el jugador podrá seleccionar el nivel que desea. Como aún se desconoce cual será la señal que utilizará el botón de start se empleará el pseudocódigo *pulsar start*. Cuando el jugador presione el botón de start el sistema capturará el nivel seleccionado y dará paso al siguiente estado.

La selección de número de vueltas es similar a la de selección de nivel. Cuando el jugador presione el botón de start el sistema habrá completado la toma de datos necesaria para el juego y por lo tanto podrá empezar el estado denominado como juego.

Finalmente cuando el número de vueltas del jugador o del coche controlado por el sistema llegue a cero se terminará el juego.

Tanto la selección de nivel como la selección del número de vueltas son sencillas, por lo que una vez conocido todo el diseño hardware se podrán diseñar directamente. El calibrado de los sensores y el juego en sí son mucho más complejos y por lo tanto se debe de profundizar en ellos.

2.2.2 Diagrama de calibración



Para poder comprender este diagrama se debe aclarar que la señal `PORTA.F0` será la que produzca el sensor para contar las vueltas del coche del jugador.

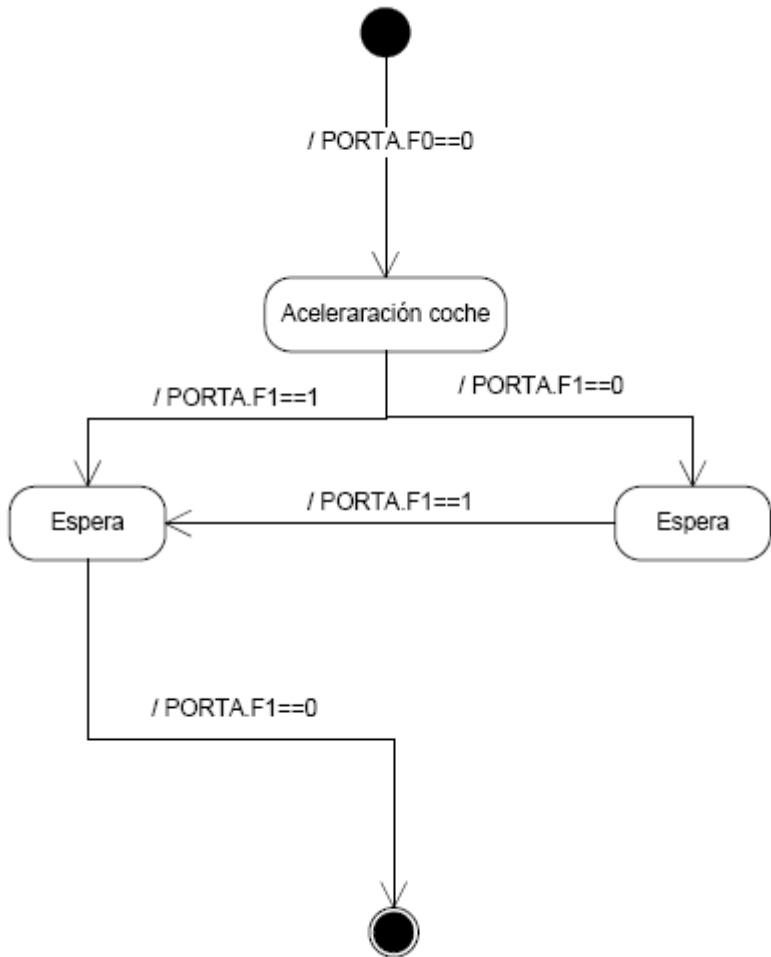
Si tras configurar el sistema se estuviera produciendo alguna señal en `PORTB.F0` significaría que algún sensor no está calibrado y por lo tanto se debe calibrar.

Como el Scalextric tiene dos pistas, una para el vehículo del jugador y otra para el coche controlado por el sistema, se debe discernir en cuál de estas dos pistas existe algún sensor que necesite ser calibrado.

Primero se comprueba la pista del jugador. En caso de ser necesario se le pedirá dar una vuelta con el coche ya que el sistema no controla ese coche y por tanto no puede realizar esa tarea.

En caso de que se trate de algún sensor de la pista controlada por el sistema, éste será quien deba calibrarlo dando una vuelta completa con el coche. No obstante, la calibración de los sensores de esta pista resulta un poco más compleja, ya que no es posible asegurar que el sensor que controla las vueltas esté funcionando correctamente y no sea él precisamente el que necesite calibrarse. Por este motivo la calibración de los sensores de la pista controlada por el sistema debe de ser más estudiada.

2.2.3 Diagrama de calibración de los sensores de la pista 2



Para poder comprender este diagrama debe aclararse que la señal producida en *PORTA.F1* corresponde al sensor encargado de controlar las vueltas del coche dirigido por el sistema.

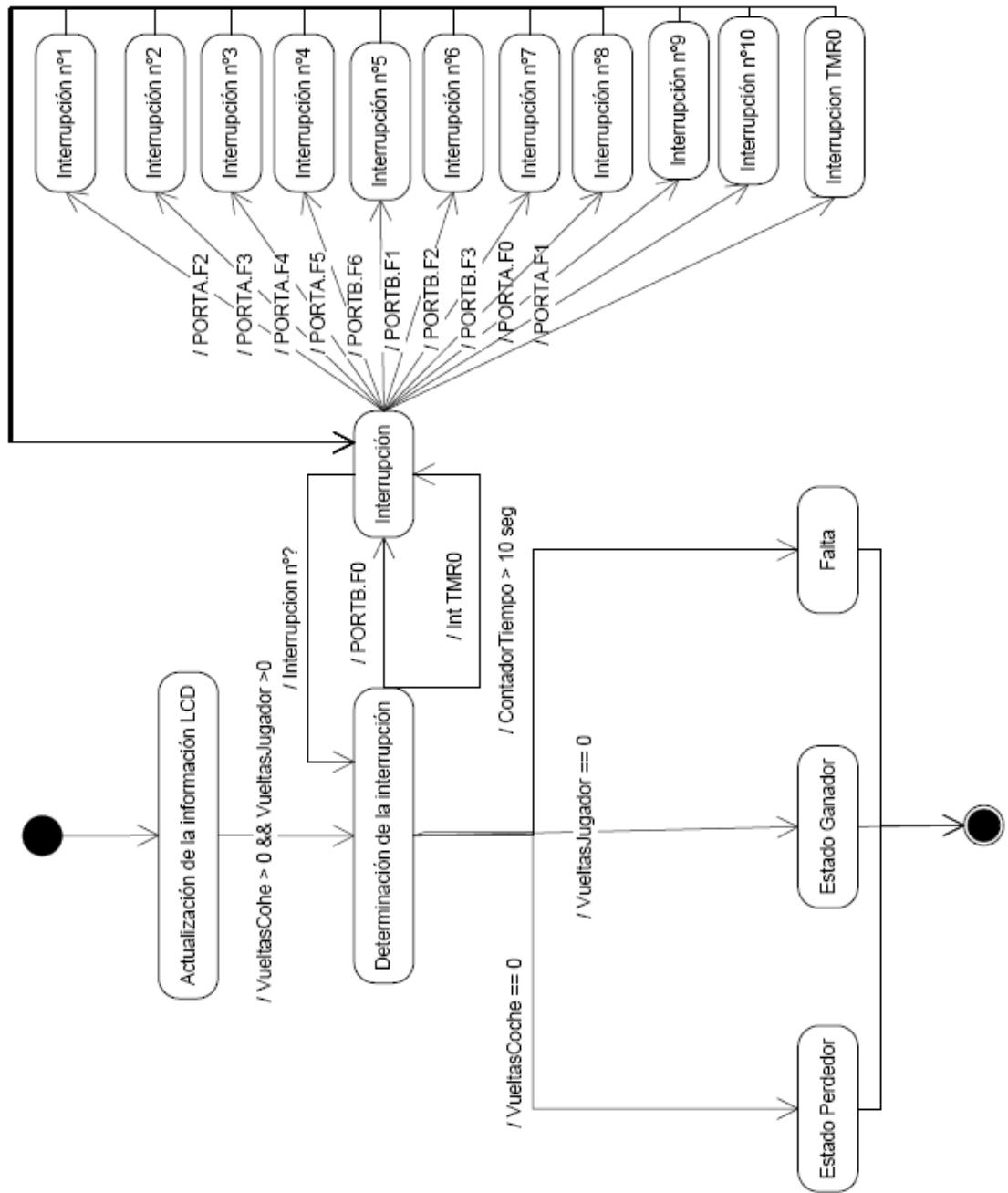
Si tras calibrar el sensor de la pista que utiliza el jugador se sigue produciendo una señal en *PORTB.F0*, significará que existe un sensor que necesita ser calibrado en la pista que dirige el sistema. Por esta razón se empezará a acelerar el coche para que pueda dar una vuelta completa y calibre los sensores. No obstante, se desconoce el estado del sensor encargado de controlar las vueltas para este coche.

Existen dos posibilidades:

- I. Sensor activo (*PORTA.F1 ==1*): si el sensor está activo el coche habrá dado una vuelta completa cuando éste se apague y, por tanto, ya habrá calibrado todos los sensores situados en esa pista.
- II. Sensor inactivo (*PORTA.F1==0*): si el sensor está inactivo, es decir, está configurado correctamente, simplemente habrá que esperar a que se active (lo cual querrá decir que habrá dado una vuelta completa) y a

que se desactive (es decir acabe de calibrar el sensor que cuenta las vueltas, ya que sino este quedaría activo y generaría interrupciones continuamente agotando la pila del microcontrolador e interfiriendo en el juego).

2.2.4 Diagrama de Juego



Para poder comprender este diagrama se ha tener en cuenta que las señales *PORTA.F0*, *PORTA.F2*, *PORTA.F3*, *PORTA.F4*, *PORTA.F5*, *PORTB.F6*, *PORTB.F2*, *PORTB.F1* y *PORTB.F3* son las señales producidas por los sensores colocados en la pista controlada por el sistema.

La señal *Int TMR0* es una señal producida por el Timer0. Éste genera una interrupción con una periodicidad determinada que será utilizada para poder ajustar el comportamiento del coche. Debe recordarse que el motor del coche se va calentando con el tiempo con lo que el sistema debe ir ajustando su comportamiento ligeramente. La única forma de hacerlo es verificar como el calor del motor afecta al vehículo.

Si el coche tarda más de lo previsto en dar una vuelta, el sistema deberá aumentar el voltaje en proporción para compensar esa pérdida producida por el calor. Para ello se utilizará la interrupción de tiempo TMR0 a modo de contador, pudiendo controlar por tanto el tiempo que tarda el coche en dar una vuelta completa.

Cuando el juego dé comienzo, lo primero que debe hacer el sistema es mostrar en la pantalla LCD el número de vueltas restantes tanto del jugador como del coche controlado por el sistema.

Mientras el número de vueltas restantes del jugador y del coche controlado por el sistema sea mayor que cero el sistema estará en el estado juego.

En este estado el flujo del programa es controlado por las interrupciones producidas por los sensores, de forma que, tras producirse una interrupción, el sistema podrá detectar en qué sensor se ha producido.

Una vez detectado ese sensor, el sistema podrá realizar cuatro acciones:

- I. Aumentar el voltaje y acelerar el coche
- II. Disminuir el voltaje y decelerar el coche
- III. Disminuir en uno las vueltas restantes del jugador
- IV. Disminuir en uno las vueltas restantes del coche controlado por el sistema

Como salidas posibles del juego se han contemplado las siguientes posibilidades:

- **Estado perdedor:** se produce cuando *vueltasCoche*, que representa las vueltas restantes del coche controlado por el sistema, llegue a cero antes de que el jugador pueda completar todas sus vueltas.
- **Estado ganador:** se produce cuando *vueltasJugador*, que representa las vueltas restantes del coche controlado por el jugador, llegue a cero antes de que el sistema pueda completar todas sus vueltas.
- **Estado falta:** se produce si el contador de tiempo alcanza un valor para 10 segundos. Es decir, que el sistema no recibe señal alguna del coche controlado por el mismo durante un período de 10 segundos. Lo que implicaría que se ha salido de la pista.

Debido al diseño del estado falta, el sistema deberá garantizar que es prácticamente imposible que el coche se salga de la pista a excepción de que sea expulsado por el coche controlado por el jugador.

Una vez concluido el diseño de hardware en su totalidad, podrá dar comienzo la fase de diseño de software, pudiendo profundizar en los diagramas aquí expuestos.

3. Fase de diseño

Una vez completada la fase de análisis puede dar comienzo la fase de diseño. Al igual que el resto del proyecto, esta fase se dividirá en dos apartados atendiendo por una parte al Hardware y por otra al Software.

3.1 Diseño de Hardware

Al igual que la fase de análisis de software, la fase de diseño de Hardware se puede subdividir en función de los componentes que serán utilizados.

3.1.1 Sensores Hall

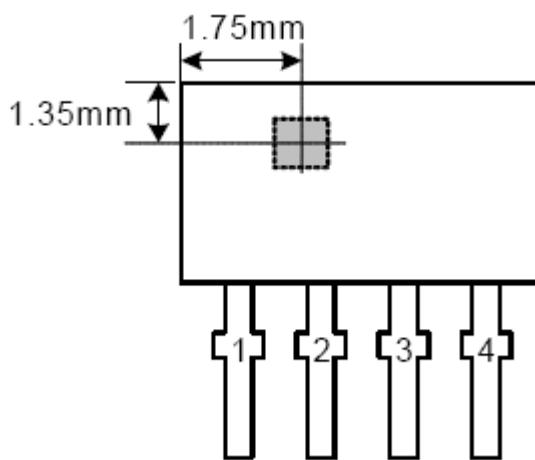
Para realizar el diseño de hardware se ha de comprender cuál es el funcionamiento de los sensores hall.

Tras la extracción de los mismos se han obtenido 14 sensores Hall válidos y 4 que han sido descartados.

Los sensores Hall que han sido descartados se deben a que su patillaje es distinto del de los otros 10, con lo que a la hora de sustituir un sensor por otro en caso de ser necesario sería imposible.

A continuación se explicará el funcionamiento y las características de estos sensores.

Los sensores Hall traen incorporado una zona sensible al efecto hall, que está situada en la parte superior izquierda tal y como se muestra en el siguiente dibujo:



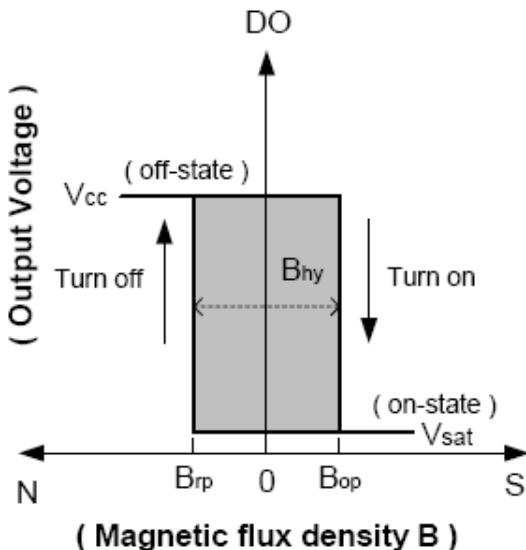
Como se puede observar, estos sensores disponen de cuatro patillas.

- Patilla 1: VCC. Entrada de alimentación.
- Patilla 2: DO. Pin de salida.
- Patilla 3: DOB. Pin de salida.

- Patilla 4: *GROUND*. Masa o tierra.

A continuación se explicará la función de las patillas 2 y 3:

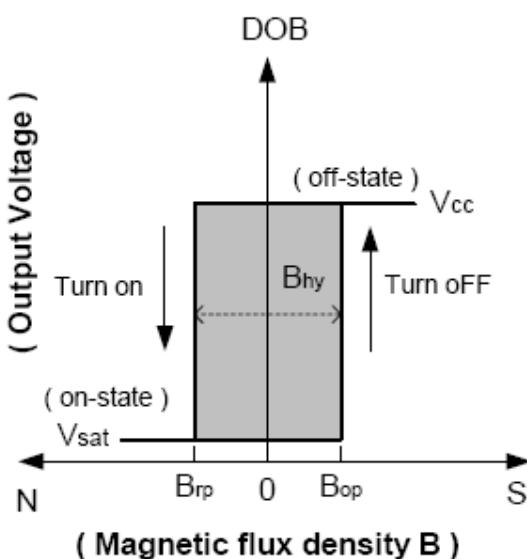
● Patilla 2 o DO



Como se puede observar en la gráfica superior, el DO genera una señal cuando la superficie del sensor, sensible al efecto hall, es alterada por un campo magnético perpendicular orientado al sur. Cuando el campo magnético esté orientado al norte se desactivará.

A efectos prácticos, se puede afirmar que si se pasa un imán por encima del sensor de forma perpendicular, de manera que primero pase la zona del imán orientada al sur y luego al norte, por la patilla de DO se generará primero una señal de nivel lógico alto y luego bajo, es decir primero un uno y luego un cero.

● Patilla 3 o DOB



Tal y como muestra la gráfica anterior, el DOB realiza la función inversa del DO. Cuando la superficie del sensor sensible al efecto hall es alterada por un campo magnético orientado al norte, éste genera una señal digital de nivel lógico alto, mientras que si está orientado al sur el sensor se desactiva.

De cara al proyecto se ha escogido la patilla 2 para poder seguir un estándar durante el cableado. No obstante, sería irrelevante puesto que si observamos las gráficas y tenemos en cuenta que los imanes de los coches tienen un polo norte y un polo sur, lo único que habría que hacer para orientar el sensor es darle la vuelta, de forma que el campo entrara perpendicularmente pero con la dirección contraria.

Características físicas

Characteristics		Symbol	Values	Unit
Supply voltage		V _{CC}	20	V
Reverse V _{CC} Polarity Voltage		V _{RCC}	-20	V
Magnetic flux density		B	Unlimited	
Output "on" current	Continuous	I _C	0.4	A
	Hold		0.5	
	Peak (Start Up)		0.7	
Operating temperature range		T _A	-20~+85	°C
Storage temperature range		T _S	-65~+150	°C
Package Power Dissipation		P _D	550	mW
Maximum Junction Temp		T _J	150	°C

Características eléctricas

Characteristic	Symbol	Conditions	Min	Typ	Max	Units
Low Supply Voltage	V _{CE}	V _{CC} =3.5V, I _L =100mA	-	0.4	-	V
Supply Voltage	V _{CC}	K version	3.5	-	20	V
		Q version	2.5*	-	20	
Output Zener Breakdown	V _Z	K version	-	46	-	V
		Q version	-	35	-	
Output Saturation Voltage	V _{CE(sat)}	V _{CC} =14V, I _L =300mA	-	0.7	0.8	V
Output Leakage Current	I _{CEx}	V _{CE} =14V, V _{CC} =14V	-	<0.1	10	µA
Supply Current	I _{CC}	V _{CC} =20V, Output Open	-	16	25	mA
Output Rise Time	t _r	V _{CC} =14V, R _L =820Ω, C _L =20pF	-	3.0	10	µs
Output Falling Time	t _f	V _{CC} =14V, R _L =820Ω, C _L =20pF	-	0.3	1.5	µs
Switch Time Differential	Δt	V _{CC} =14V, R _L =820Ω, C _L =20pF	-	3.0	10	µs

*The output of DO/DOB will be switched on/off after supply voltage reaching the 2.5V.

Como datos a destacar, se debe tener en cuenta que soportan una tensión máxima de 20 voltios y funcionan con una mínima de 3.5 V. Además son capaces de funcionar a una temperatura entre -20 °C y 85 °C.

Una vez comprendido el funcionamiento de estos sensores se puede diseñar el cableado del Scalextric.

3.1.2 Cableado del Scalextric

El cableado será utilizado tanto para alimentar a los sensores como para transmitir la señal digital hasta el microcontrolador.

Teniendo en cuenta esto, queda claro que el cable debe estar formado por tres hilos. Uno para el *Vcc*, otro para el *Gnd* y un tercero que irá conectado al *DO*.

Como ya se ha comentado anteriormente, el cable empleado será el cable de bus ya que es flexible y muy maleable, lo que facilitará su instalación.

Teniendo en cuenta que los sensores se pueden estropear, se ha pensado en un sistema que permita su sustitución. Para ello se emplearán tiras de pines, de forma que el sensor pueda ser acoplado en el extremo del conector a diseñar.



Se utilizaran dos tiras de pines, de cuatro y tres pines, que se conectarán de la siguiente forma.

Pin 1 → Pin 1
Pin 2 → Pin 4
Pin 3 → Pin 2

El pin 3 del sensor quedará al aire ya que no será usado.

A su vez, el sensor será ensamblado en otra tira de 4 pines de forma que pueda ser sustituido en caso de avería.

3.1.3 Power MOSFET

Para variar la velocidad del coche controlado por el sistema se necesita un componente que permita variar el voltaje transmitido a la pista. Para ello se empleará un dispositivo denominado transistor de efecto de campo. Este transistor se basa en el campo eléctrico para controlar la conductividad de un canal en un material semiconductor. Desde un nivel más práctico, los transistores de efecto de campo pueden plantearse como resistencias controladas por voltaje.

Este dispositivo es necesario ya que el microcontrolador en la patilla de PWM (control de ancho de pulso) sólo sería capaz de sacar como máximo 3,5 voltios mientras que el Scalextric necesita una tensión máxima de 14 voltios.

Por tanto se puede utilizar el voltaje del PWM para controlar el transistor de efecto de campo y que éste a su vez controle el voltaje que, finalmente, alimenta motor del coche.

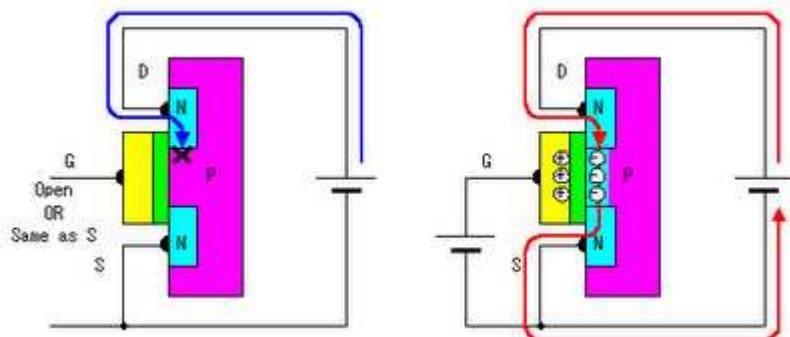
Dentro de los transistores de efecto de campo existen numerosas opciones, aunque la más extendida para circuitos electrónicos es la MOSFET (en inglés Metal Oxide Semiconductor Field Effect Transistor),



Power MOSFET

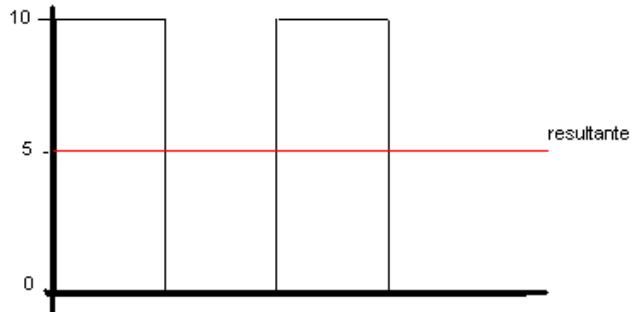
El transistor MOS básico es un componente de tres terminales: dos terminales denominados **D** (de la palabra inglesa *drain* que significa drenaje) y **S** (de la palabra inglesa *source* que significa fuente) en las que se aplica una fuente de voltaje, y una terminal de "control" denominada **G** (de la palabra inglesa *gate* que puede tomarse como gatillo o compuerta).

Para entender mejor su funcionamiento a continuación se expondrán dos diagramas. La región de color amarillo es la capa "metálica", la región de color verde es la región de "óxido" y la región de color magenta es el sustrato del semiconductor:



Como se puede ver en el diagrama de la izquierda, cuando la patilla terminal **G** está abierta, sin que se le esté aplicando voltaje alguno, entonces la corriente eléctrica que está tratando de enviar el polo positivo de la batería (resaltada en la trayectoria incompleta de color azul) llega hasta la región (de color cian) denominada **N** en donde la posible conducción de corriente eléctrica queda detenida al no haber en la región **P** (de color magenta) camino alguno mediante el cual se pueda completar la trayectoria. La situación cambia por completo cuando en el diagrama de la derecha le aplicamos un voltaje a la terminal **G**. Al aplicar dicho voltaje, se polarizan las regiones de color amarillo, verde y magenta, de modo tal que mientras en la región de color amarillo se crean cargas eléctricas positivas, en la región **P** de color magenta (que llamaremos el sustrato) se crea una contraparte de igual número de cargas eléctricas negativas, formando dentro de la región **P** un "canal" **N** de color azul que conecta las dos regiones **N**, con lo que se establece un camino para la conducción de energía eléctrica (resaltada en la trayectoria cerrada de color rojo).

Simplificándolo mucho se podría decir que este dispositivo funciona como un interruptor eléctrico de gran velocidad, con lo que al abrir y cerrar el circuito la tensión resultante podría ser modificada. Es decir, si con una frecuencia de 1 Hz tenemos una señal periódica la cuál está 0,5 segundos a cero y 0,5 segundos con un valor de 10, la integral resultante nos daría 5.



Características físicas y eléctricas

Rating	Symbol	Value	Unit
Drain-to-Source Voltage	V_{DSS}	28	Vdc
Gate-to-Source Voltage – Continuous	V_{GS}	± 20	Vdc
Drain Current – Continuous @ $T_A = 25^\circ\text{C}$ – Single Pulse ($t_p = 10 \mu\text{s}$)	I_D I_{DM}	60* 120	Adc
Total Power Dissipation @ $T_A = 25^\circ\text{C}$	P_D	75	Watts
Operating and Storage Temperature Range	T_J, T_{stg}	-55 to 150	$^\circ\text{C}$
Single Pulse Drain-to-Source Avalanche Energy – Starting $T_J = 25^\circ\text{C}$ ($V_{DD} = 28 \text{ Vdc}$, $V_{GS} = 10 \text{ Vdc}$, $I_L = 17 \text{ Apk}$, $L = 5.0 \text{ mH}$, $R_G = 25 \Omega$)	E_{AS}	733	mJ
Thermal Resistance – Junction-to-Case – Junction-to-Ambient (Note 1) – Junction-to-Ambient (Note 2)	R_{eJC} R_{eJA} R_{eJA}	1.65 67 120	$^\circ\text{C/W}$
Maximum Lead Temperature for Soldering Purposes, 1/8" from case for 10 seconds	T_L	260	$^\circ\text{C}$

Este dispositivo se debe conectar de la siguiente manera:

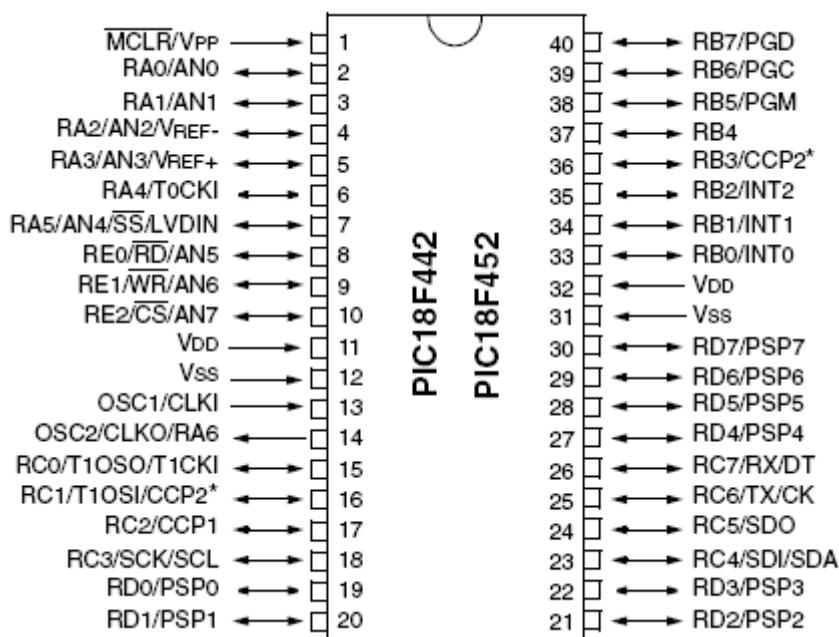
- Patilla 1 o Gate: se conectará a la patilla de PWM del microcontrolador a través de una resistencia de 22Ω .
- Patilla 2 o Drain: se conectará a la pista del Scalextric que proporciona la masa.
- Patilla 3 o Source: se conectará a la pista del Scalextric que proporciona el positivo, y a su vez, se conectará a la masa de la placa para que exista una diferencia de potencial de 14 voltios.

3.1.4 Microcontrolador

Como ya se ha mencionado anteriormente, el microcontrolador que se empleará será el 18LF452. Antes de diseñar la placa se debe conocer cómo es el microcontrolador físicamente.

Existen varias implementaciones físicas para este tipo de microcontrolador. No obstante el elegido es el formato DIP ya que, a la hora de trabajar a mano con él, resulta mucho más cómodo pues, no es necesaria la adquisición de ninguna placa especial para colocarlo.

El esquema físico del mismo es el siguiente:

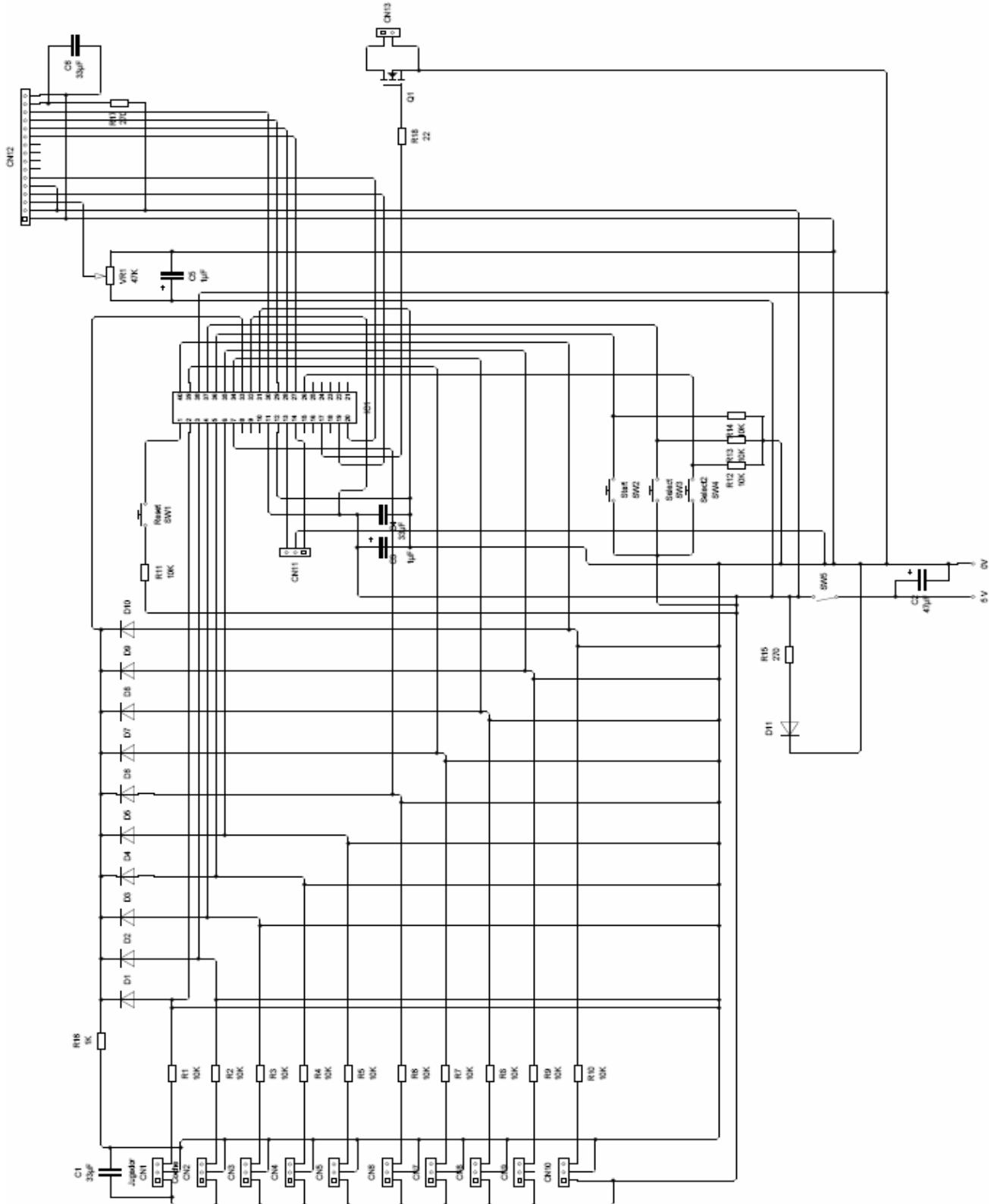


Teniendo en cuenta que este PIC es de gama alta una vez realizado el programa cabe la posibilidad de poder emplear un microcontrolador con un precio más asequible. En este caso este PIC guarda compatibilidad con los de la familia 16C7x.

3.1.5 Alimentación

Por último y antes de diseñar el prototipo de la placa se ha de tener en cuenta la alimentación. Finalmente se ha obtenido un transformador capaz de producir una tensión de 5 voltios. Esto le permitirá a la placa disponer de una fuente de alimentación al margen del Scalextric e implica que en el diseño de la placa se ha de colocar una toma junto con un condensador electrolítico de 47 microfaradios.

3.1.6 Diseño Lógico del circuito



Para el diseño del prototipo se ha empleado el programa Livewire. Este es un programa de diseño de circuitos utilizado en los institutos de Educación Secundaria Obligatoria. Ha sido escogido por su sencillez y porque permite la transformación del diseño lógico al PCB. Es decir, una vez obtenido el diseño empleando la simbología de circuitos, se pueden obtener los planos para la construcción de una placa de circuitos integrados.

A continuación se explicará brevemente el diseño generado.

El conector *CN1* representa la conexión con el cable usado para contar las vueltas del coche que utilizará el jugador.

El conector *CN2* será el cable usado para contar las vueltas del coche controlado por el sistema.

El resto de los conectores *CN* (desde el 3 hasta el 10) serán usados para poder determinar la posición del coche controlado por el sistema pudiendo decidir entonces si el coche necesita más o menos velocidad.

Como se puede observar, todos estos conectores poseen el mismo entramado de pistas.

Empezando de izquierda a derecha, el Pin 1 está unido a la línea de 5 voltios ya que el sensor en la pata 1 necesita esa alimentación. El Pin 2 está unido a 0 voltios o tierra y el Pin 3 está unido a una resistencia, un diodo y al *IC1*.

Este último elemento (*IC1*) representa el zócalo sobre el que será colocado el microcontrolador 18LF452.

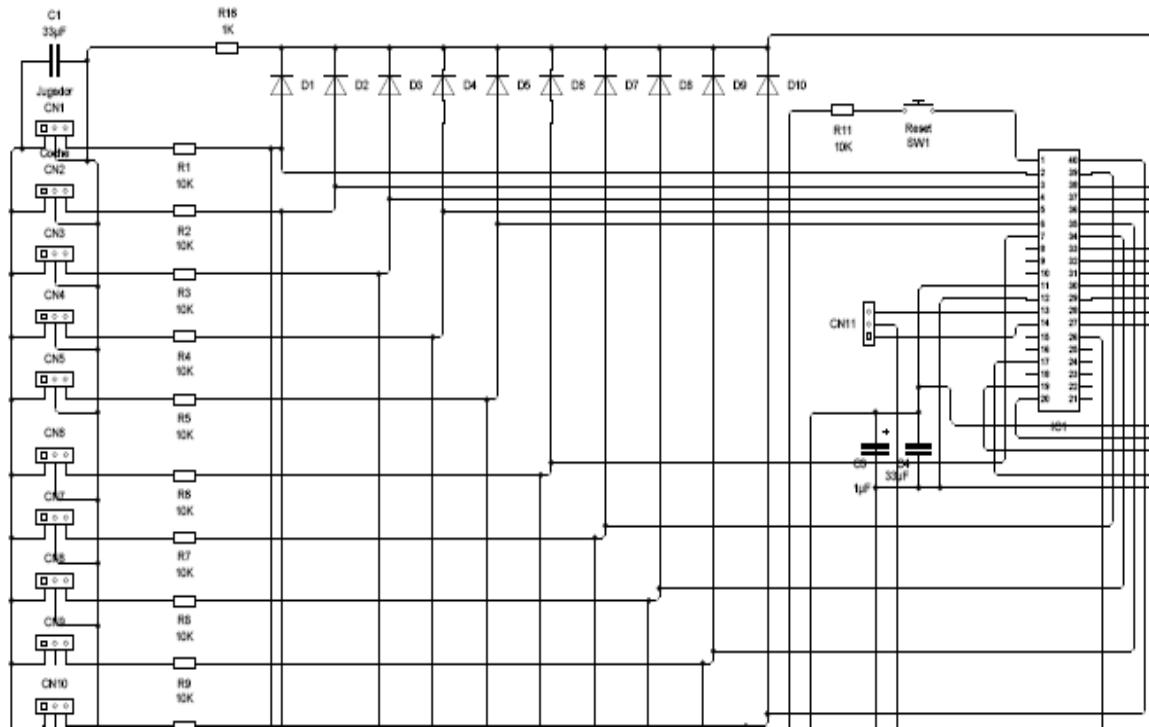
Hay que destacar que en la línea de 5 voltios y masa se debe colocar un condensador cerámico de $33\ \mu F$ (microfaradios) para desacoplar los sensores.

La resistencia debe ser de $10\ k\Omega$. Ésta es empleada para fijar el nivel de tensión cuando el sensor no transmite señal. Tal y como indica el esquema eléctrico de los sensores esta debe ser conectada a la línea de 5 voltios. Es decir, que el sensor trabaja con nivel lógico alto.

Los diodos, en este caso, sirven para implementar una puerta lógica OR de forma que cuando algún sensor envíe una señal, por el hilo saliente a los diodos circulará una tensión de 5 voltios, lo que en programación equivale a un 1, que será utilizado para generar interrupciones.

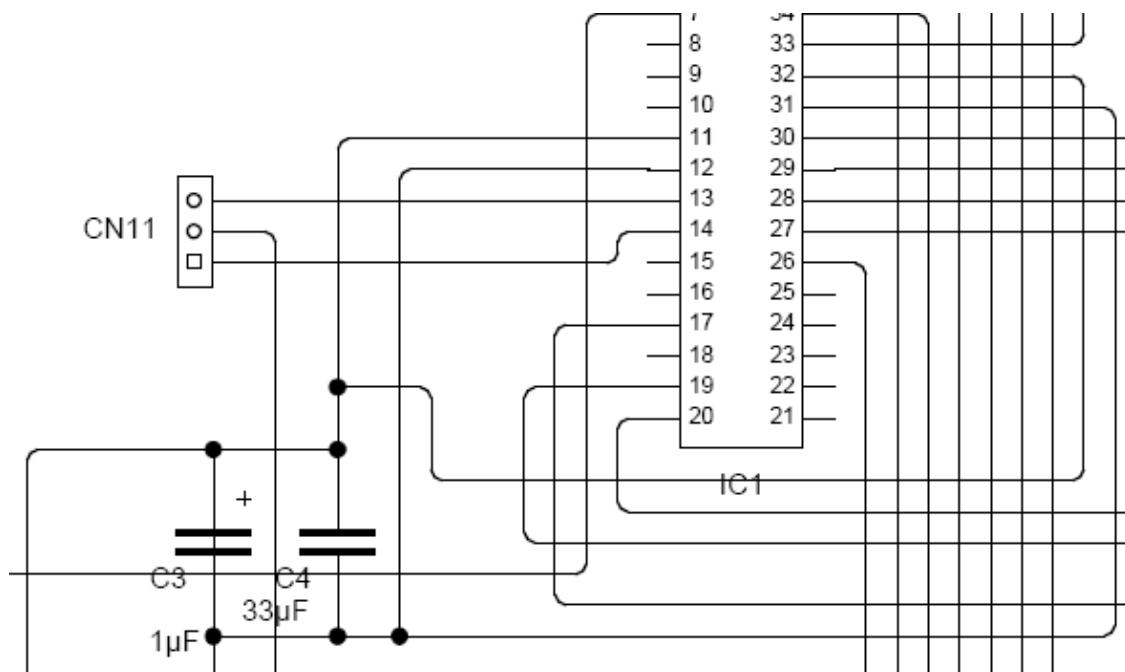
Este último hilo será conectado en el pin 33 del *IC1*, que representa la patilla de interrupción INT0 del microcontrolador.

Por último de cara a la detección de los sensores, cabe destacar que de cada sensor llega un hilo al *IC1*, concretamente a las patillas 2, 3, 4, 5, 6, 7, 39, 34, 35 y 40 respectivamente. Estas conexiones serán utilizadas para discernir qué sensor concreto produjo la interrupción.



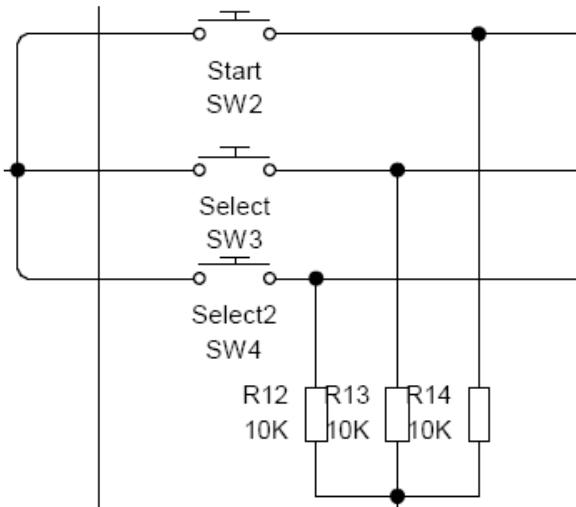
El elemento *CN11* representa el cristal de cuarzo, necesario para que el microcontrolador ejecute las interrupciones de tiempo con mayor exactitud. Se podría emplear el reloj interno del PIC, pero este componente no supone un coste muy elevado y merece la pena disponer de él.

Las patillas 11, 12, 32 y 31 son las patillas de alimentación del microcontrolador. A su vez se les debe colocar un condensador electrolítico de 1 μF y otro cerámico de 33 μF .

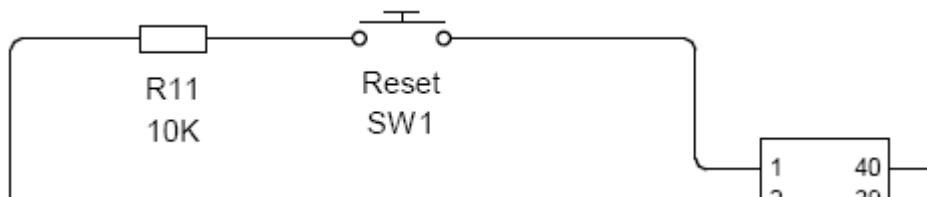


El sistema también deberá disponer de 4 botones. Los 3 situados en parte derecha del plano general funcionan con lógica normal, es decir, al

pulsar generarán 5 voltios (un 1 lógico) y al soltarlos retornarán a cero. Al igual que en la línea de los sensores, para los pulsadores se han empleado unas resistencias de $10\text{ k}\Omega$ para fijar el nivel del hilo cuando no haya ningún pulsador cerrando el circuito. Como estos pulsadores funcionan con lógica normal las resistencias de carga deben de ir conectadas a masa. Los pines del conector *IC1* utilizados por estos tres pulsadores son 36, 37 y 26.



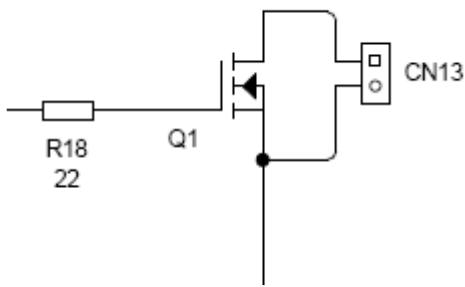
El botón de reset en cambio debe funcionar al revés que el resto de pulsadores. Si observamos el esquema del microcontrolador, en la patilla 1 el master-clear (*MCLR*) indica que el PIC se resetea cuando recibe un cero lógico, es decir cero voltios. Por lo tanto la resistencia de este pulsador debe ser conectada a la línea de 5 voltios, fijando al igual que los sensores la línea con un nivel lógico alto.



La patilla 17 será conectada al Power MOSFET. Éste se encargará de transformar la tensión de salida del PWM del microcontrolador en la necesaria para poder mover el coche.

De este dispositivo se derivarán dos cables que serán conectados en la toma del mando a sustituir. En este caso esas derivaciones están representadas por la figura *CN13*.

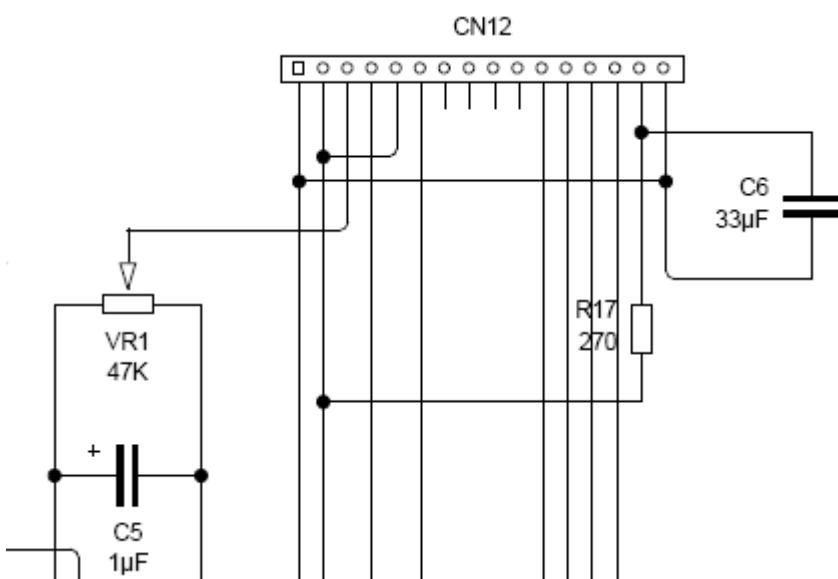
El proceso de conexión de estos cables se explicará más adelante.



La figura CN12 representa el zócalo en el que se anclará el LCD. Este es una tira de pines de 16 elementos. Los pines 1 y 2 estarán conectados a positivo y masa respectivamente. Los pines 3 y 4 se encargan de regular el contraste. El 4 irá conectado a masa mientras que 3 se conecta a un potenciómetro o resistencia variable que permitirá regular el contraste de la pantalla con el nivel deseado. A su vez, a este potenciómetro se le acoplará un condensador electrolítico de $1\ \mu F$.

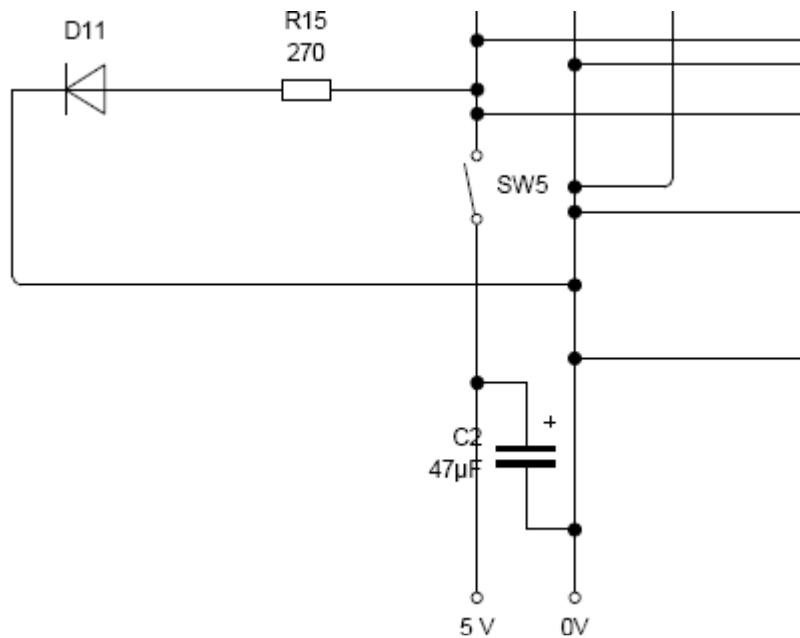
Los pines 15 y 16 son los encargados de iluminar la pantalla LCD por lo que llevan incorporados un condensador cerámico de $33\ \mu F$ para desacoplar dicho display y evitar así interferencias en el mismo que produzcan caracteres no deseados.

Los pines 6, 11, 12, 13 y 14 son conectados al microcontrolador a través de los pines 20, 27, 28, 29 y 30 respectivamente. Es decir, son los pines empleados por el microcontrolador para escribir en la pantalla.



Por último, para la alimentación será imprescindible usar un conector adecuado para el transformador y un interruptor, así como un diodo LED que actúe de testigo e indique cuando la placa está encendida o apagada. Teniendo en cuenta que los LEDs funcionan con una tensión de 1 voltio, es necesario colocarlo en serie con una resistencia de $270\ \Omega$.

Además entre la línea de positivo (5 voltios) y masa se debe acoplar un condensador electrolítico de $47 \mu F$ para reducir las interferencias eléctricas que puede producir el transformador.



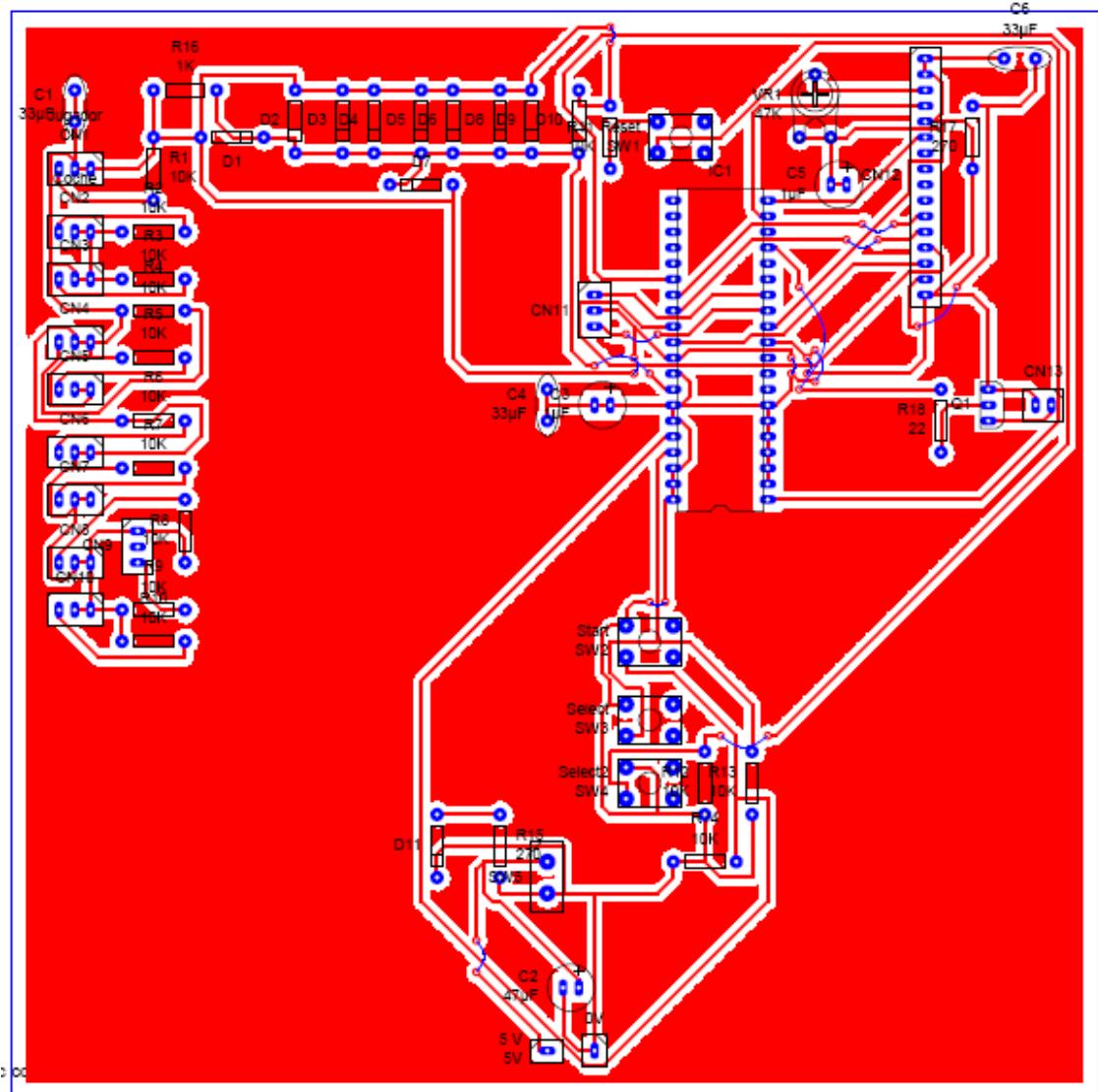
Tal y como se ha mencionado anteriormente, una vez desarrollado el esquema lógico, con el programa Livewire es posible la obtención del PCB.

Antes de mostrar los diseños se debe aclarar que las líneas azules son puentes. Es posible construir una placa que no utilice puentes si se construyen las pistas por ambos lados de la misma. No obstante y debido a su dificultad, para este proyecto es preferible un diseño más sencillo.

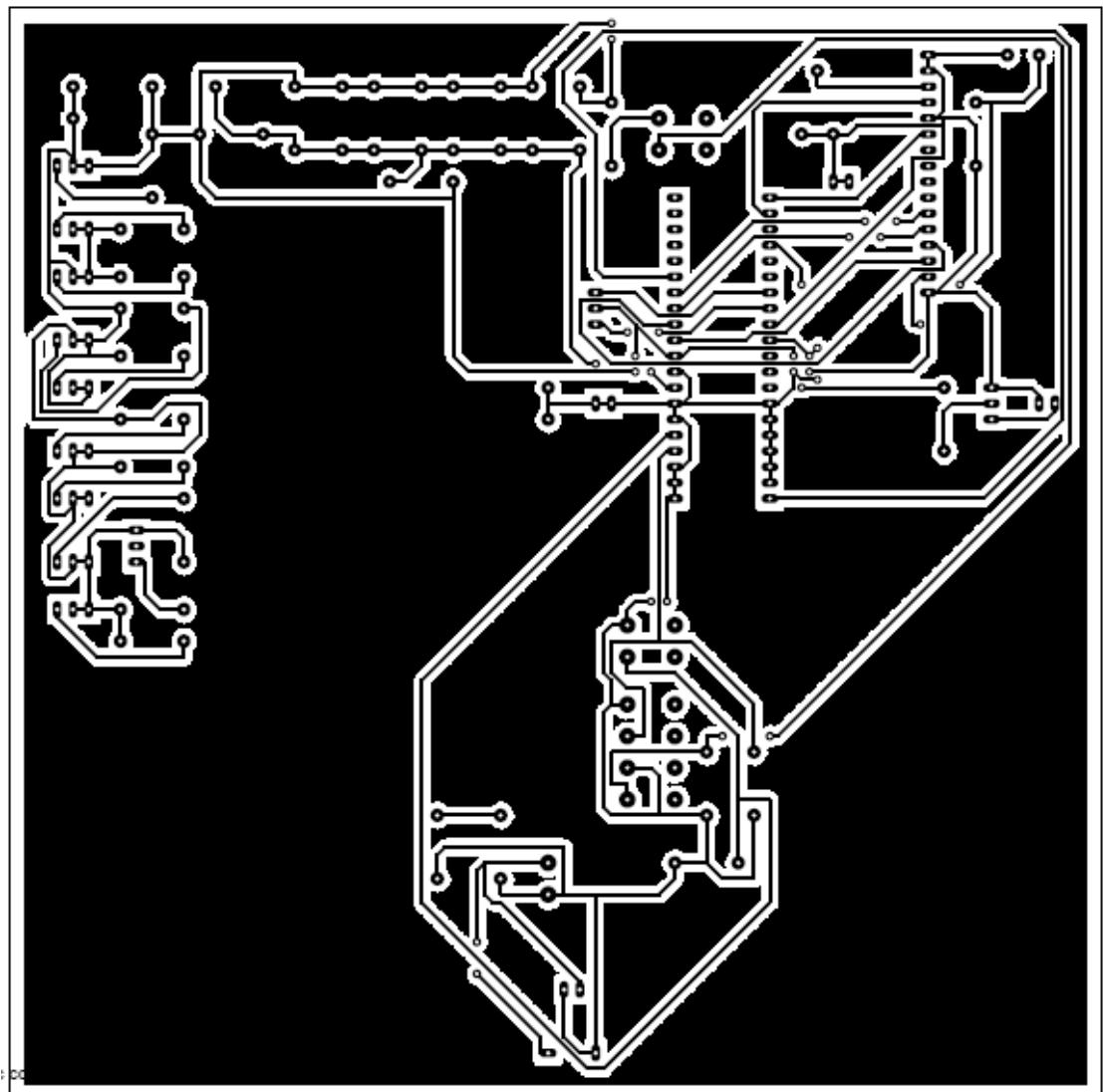
A continuación se muestran los diseños para la realización de la placa de circuito impreso:

3.1.7 PCB

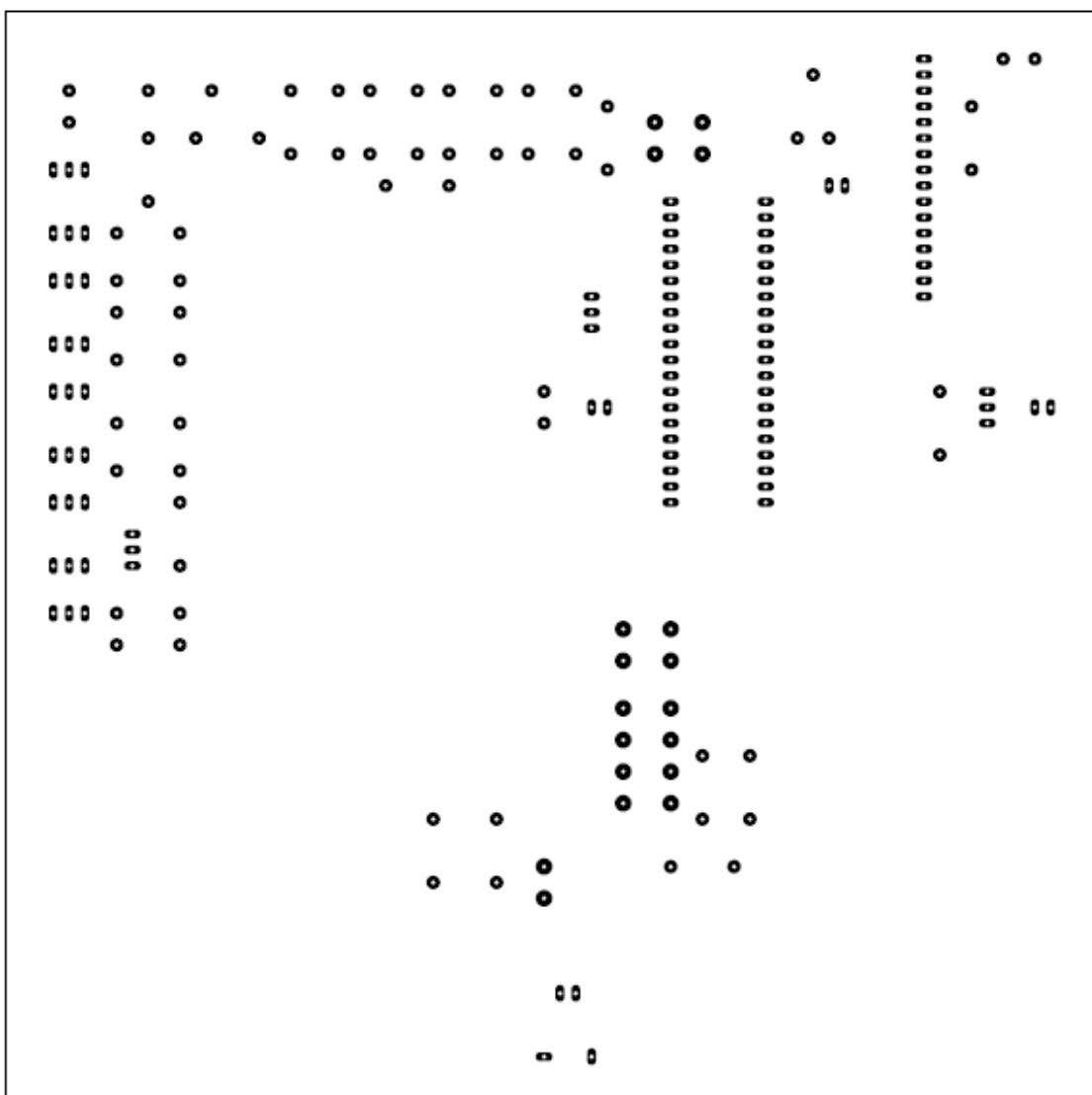
● Placa Normal:



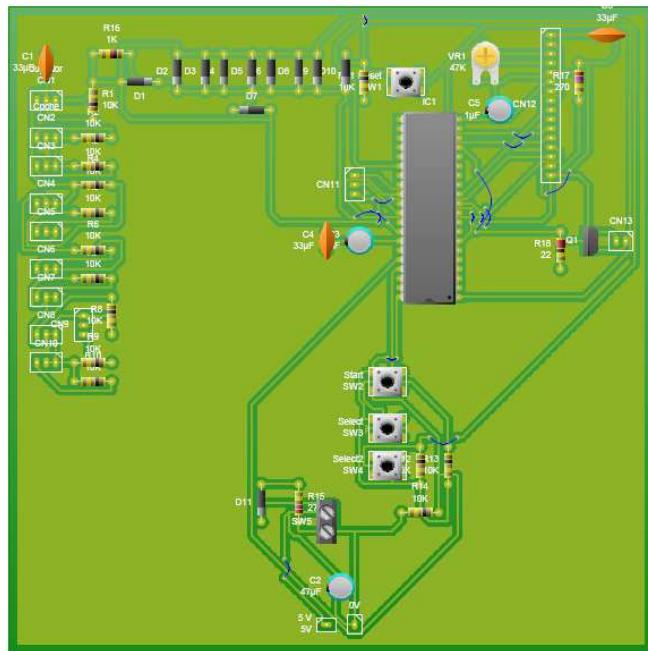
- Placa para el ruteado de las pistas:



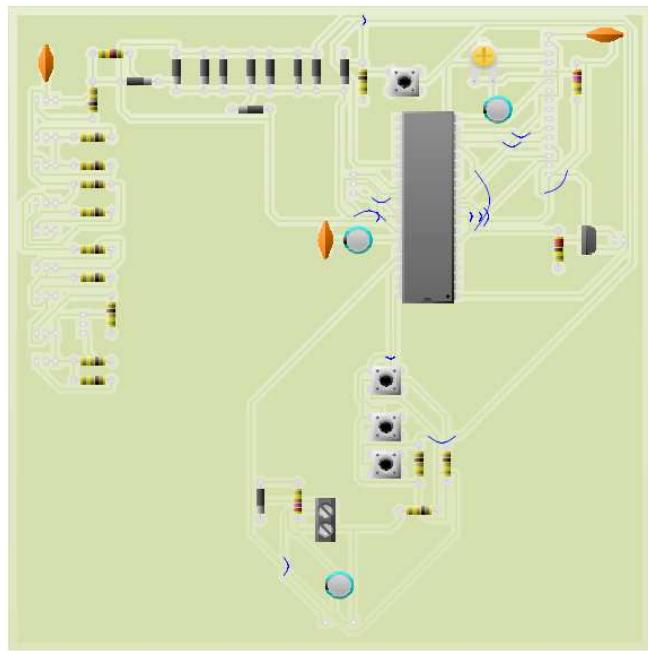
● Placa para el taladrado:



● Placa real de silicio:



● Placa real de prototipo:



Debe tenerse en cuenta que el tamaño real de las placas ha sido modificado. No obstante, con la placa para el ruteado de pistas y la de taladrado, el sistema podrá ser reproducido sin ningún problema.

3.2 Diseño de software

Para poder realizar el diseño del software se deben conocer cuáles son los voltajes que se le deben imprimir al coche según su posición. Para ello se han estudiado los vídeos con el programa Kino. Este programa permite la edición de vídeo, lo que lo hace perfecto para poder examinar fotograma a fotograma estos vídeos facilitando así la toma de datos. Para ello se han analizados los vídeos filmados por cada tramo de la pista y se ha obtenido el mayor voltaje en cada punto. Se ha procurado obtener la mayor velocidad posible en cada punto del Scalextric, de forma que en el nivel 1 el sistema imprima este voltaje en el coche.

La siguiente tabla muestra los datos obtenidos.

Posición	Sensor	Voltaje	PWM
Curva izquierda entrada	Sensor 3	9.63 voltios	175
Curva izquierda medio entrada	Sensor 4	9.72 voltios	178
Curva izquierda medio salida	Sensor 5	9.44 voltios	172
Curva izquierda salida	Sensor 6	10,91 voltios	200
Curva derecha entrada	Sensor 7	8,22 voltios	150
Curva derecha medio entrada	Sensor 8	9.0 voltios	165
Curva derecha medio salida	Sensor 9	9.81 voltios	180
Curva derecha salida	Sensor 10	9.88 voltios	188

Teniendo en cuenta la información obtenida de los vídeos resulta imposible conocer el tiempo que tarda el coche en completar una vuelta. Debe recordarse que el comportamiento del coche irá variando en función de la temperatura a la que se encuentre su motor. Por lo tanto, durante esta fase se diseñará la forma de reajustar el PWM para que el coche se comporte igual en todo momento, aunque no se podrá precisar por falta de datos. Una vez obtenida una placa de circuitos totalmente funcional se podrá solucionar este problema y cronometrar el tiempo de una vuelta en los distintos niveles de Juego.

3.2.1 Configuración de los registros

Como ya se ha mencionado en la parte de diseño de hardware, el microcontrolador empleado es el 18LF452. Debido a que es necesario el uso de la patilla de PWM se utilizará una frecuencia de 4 Mhz pues varios expertos consultados a través del foro que proporciona la página de mikropic.com coinciden en que es la mejor frecuencia para el empleo de esta tecnología.

● Configuración para la habilitación de interrupciones

Una vez conocida la frecuencia a la que trabajará el microcontrolador, es posible realizar los cálculos para la obtención de las interrupciones del *TMR0* con un período de tiempo determinado.

La fórmula que se debe emplear es:

$$T = 4 \times T_{osc} \times (256 - TMR_0) \times \text{Preescaler}$$

siendo *T* el tiempo total en segundos y *T_{osc}* el tiempo de oscilación, es decir, la inversa de la frecuencia.

El preescaler es un valor escogido por el diseñador que puede tomar los valores 2, 4, 8, 16, 32, 64, 128 y 256. Por lo tanto con una frecuencia de 4 Mhz el valor más alto que se puede obtener sería.

$$T = 4 \cdot \frac{1}{10^{(-6)}} \cdot (256) \cdot 256 = 0.065\text{seg}$$

Para facilitar los cálculos se ha decidido producir interrupciones cada 50 milisegundos. Por lo tanto el cálculo que se debe realizar es el siguiente:

$$0.05 = 4 \cdot \frac{1}{10^{(-6)}} \cdot (256 - TMR_0) \cdot 256$$

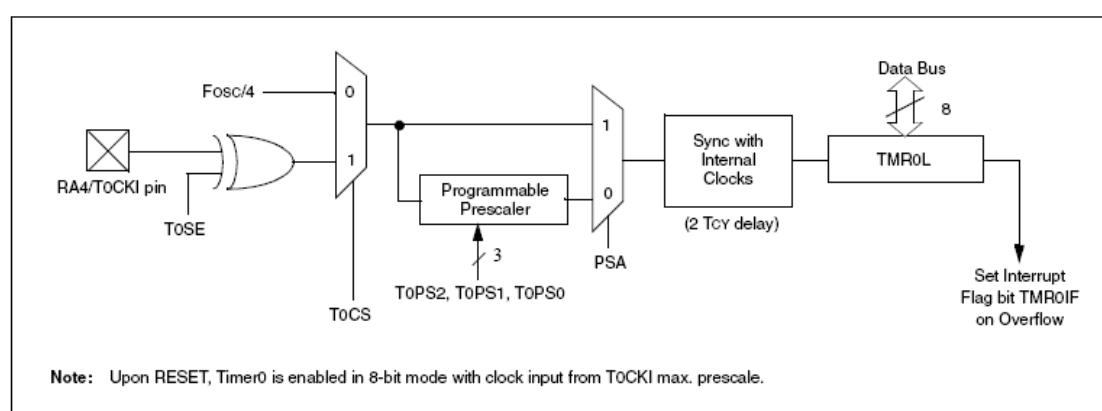
$$0.05 = -0.000256TMR_0 + 0.065536$$

$$TMR_0 = 60.68$$

De este modo el registro *TMR0* debe tomar el valor 61.

Este microcontrolador dispone de dos registros denominados *TMR0H* y *TMR0L* para la configuración del registro *TMR0*. No obstante será suficiente emplear el *TMR0L* y trabajar en modo 8 bits ya que 61 es menor que 256 y por lo tanto el registro puede ser configurado de este forma.

A modo de curiosidad en el diagrama lógico se incluye cómo funciona este timer en modo 8 bits.



Desde un punto de vista práctico, el microcontrolador pondrá el flag denominado *TMROIF* a uno trascurrido cierto tiempo. En este caso 50 milisegundos, de forma que el tiempo pueda ser controlado por el programador.

Una vez obtenido el valor del registro *TMROL* hay que configurar el registro *TOCON*, también denominado registro de control del *TMR0*.

TOCON: TIMER0 CONTROL REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

El funcionamiento del mismo es el siguiente:

- TMR0ON: habilita o deshabilita las interrupciones del TMR0.
- T08BIT: configura el TMR0 en modo 8 bits o 16 bits.
- TOCS: habilita el uso del reloj interno del microcontrolador o el uso externo de un cristal de cuarzo.
- TOSE: configuración para utilizar el flanco de subida o de bajada producida por la señal del reloj.
- PSA: habilita o deshabilita el uso del preescaler.
- TOPSx: selección del preescaler :
 - 000 = 1:2.
 - 001 = 1:4.
 - 010 = 1:8.
 - 011 = 1:16.
 - 100 = 1:32.
 - 101 = 1:64.
 - 110 = 1:128.
 - 111 = 1:256

Teniendo en cuenta toda la información mostrada, el registro *TOCON* deberá tomar el valor 11000111 o, lo que es lo mismo, C7 en hexadecimal.

El último registro que se debe tener en cuenta para trabajar con interrupciones es el registro *INTCON* cuyo formato es el siguiente:

INTCON REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMROIE	INT0IE	RBIE	TMROIF	INT0IF	RBIF
bit 7							bit 0

- GIE/GIEH: habilita o deshabilita todas las interrupciones.
- PEIE/GIEL: habilita o deshabilita las interrupciones periféricas.
- TMROIE: habilita o deshabilita las interrupciones del TMR0.
- INT0IE: habilita o deshabilita las interrupciones producidas por la patilla INT0.

- RBIE: habilita o deshabilita las interrupciones producidas por el puerto B.
- TMR0IF: flag para el TMR0.
- INT0IF: flag para el INT0.
- RBIF: flag para el puerto B.

Debido a que en el sistema se utilizarán las interrupciones producidas por el *TMR0* y por la entrada *INT0*, cuando estas interrupciones estén habilitadas, el registro quedará configurado con el valor 10110000 o, lo que es lo mismo, B0 en notación hexadecimal.

Debe tenerse en cuenta que una vez que se produzca una interrupción los flags *TMR0IF* e *INT0IF* deben ser reseteados volviéndolos a colocar con valor cero.

Configuración de la entrada digital para el puerto A

Como el sistema utiliza el puerto A como entrada de señales digitales y el microcontrolador dispone en ese puerto de convertidores analógicos/digitales, se debe configurar este puerto de forma correcta. Para ello se utiliza el registro ADCON1.

ADCON1 REGISTER

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

- ADFM: Justificación de la conversión A/D a la derecha o a la izquierda.
- ADCS2: habilitación de la conversión A/D.
- PCFGx: configuración de las patillas del puerto A según la siguiente tabla.

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C / R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8 / 0
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7 / 1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5 / 0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4 / 1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3 / 0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2 / 1
011x	D	D	D	D	D	D	D	D	—	—	C / 0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6 / 2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6 / 0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5 / 1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4 / 2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3 / 2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2 / 2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1 / 0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1 / 2

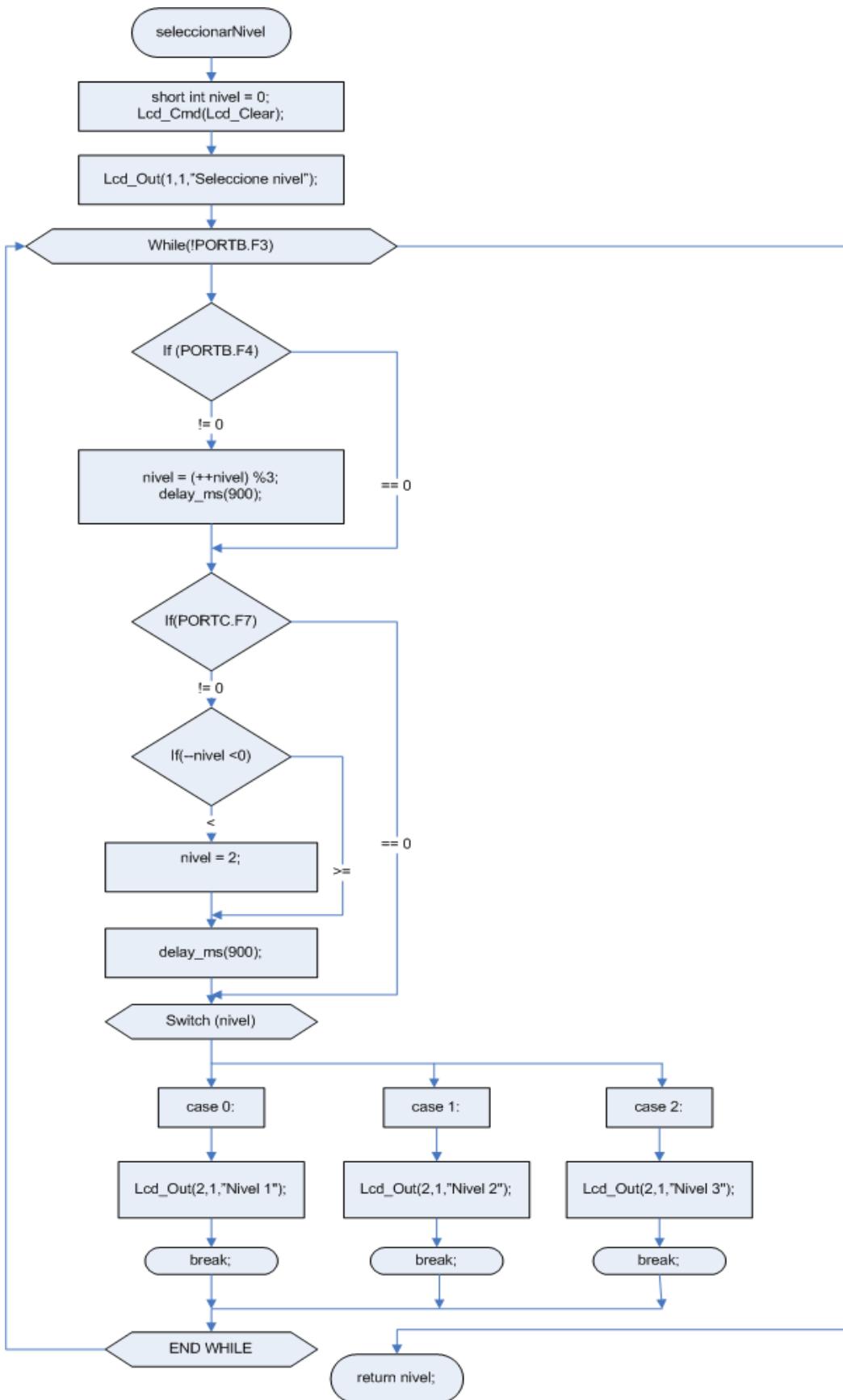
A = Analog input D = Digital I/O

C/R = # of analog input channels / # of A/D voltage references

Por lo tanto el valor que se le debe asignar al registro A en este caso es 10000111 o, lo que es lo mismo, 87 en notación hexadecimal.

Una vez realizado el diseño para la configuración del microcontrolador se puede comenzar a diseñar el software, pues ya se dispone de un sistema hardware funcional, al menos teóricamente.

3.2.2 Diagrama de flujo para la selección de nivel



Para la construcción de la función encargada de la selección de nivel es necesario el uso de una variable local denominada *nivel*. Esta variable será utilizada como contador y se irá incrementando y disminuyendo según dicte el usuario.

Primero se borrará toda la pantalla con el comando *Lcd_Cmd(Lcd_Clear)* y a continuación se mostrará en la primera fila del display la frase “Seleccione el nivel”. Como la variable *nivel* está inicializada con el valor cero se mostrará en la segunda fila la frase “Nivel 1”.

Si el usuario pulsa el botón *select1* se recibirá una señal en *PORTB.F4* y el sistema incrementará en una unidad la variable *nivel*. Los niveles fijados durante los requerimientos son tres, por lo que los valores que se le permiten tomar a la variable *nivel* son:

- 0, representa el nivel 1.
- 1, representa el nivel 2.
- 2, representa el nivel 3.

Por este motivo después de incrementar en uno el nivel se le aplica la operación módulo de 3 de forma que, si la variable toma valor 2 y el usuario pulsa el botón *select1* la variable tomará el valor 0. A efectos prácticos, lo que se consigue es que el usuario pueda pasar del nivel 3 al 1 solamente pulsando el botón *select1*.

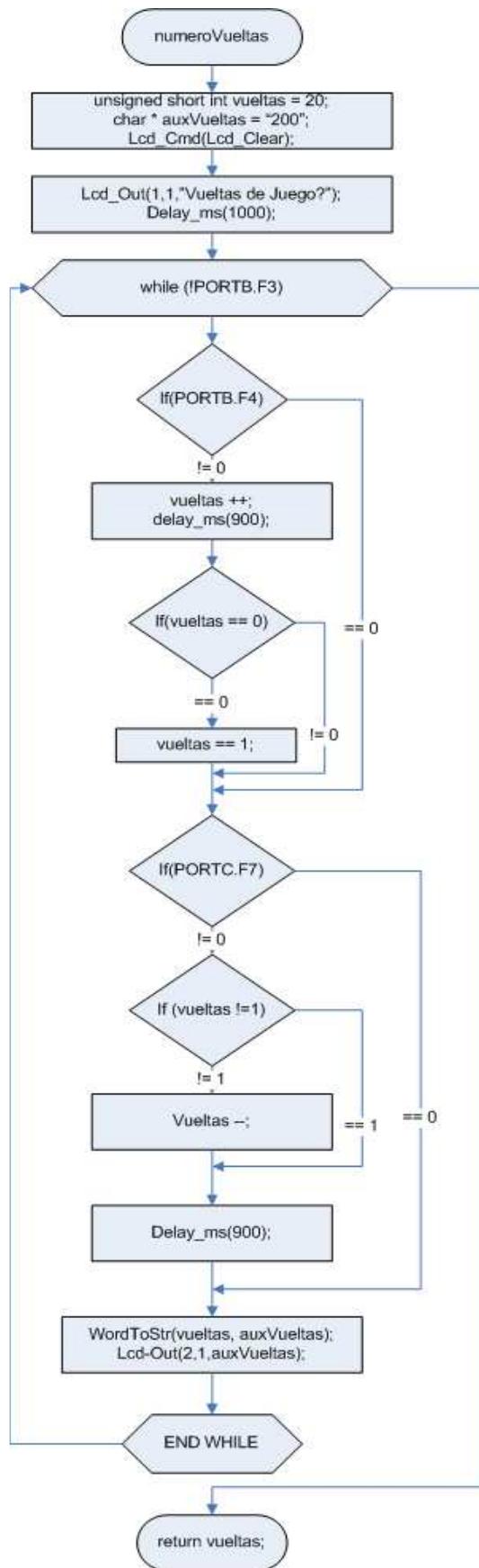
Si el usuario pulsa el botón *select2* se recibirá una señal en *PORTC.F7* y el sistema disminuirá en una unidad la variable *nivel*. Para seguir un estándar, igual que cuando se incrementa el valor, el sistema comprobará si el resultado es menor que cero y en caso de serlo se le asignará automáticamente el valor 2 a esta variable. A efectos prácticos, lo que se consigue es que el usuario pueda pasar del nivel 1 al 3 solamente pulsando el botón *select2*.

La función *delay_ms()* obliga al sistema a realizar una pausa. Esto garantiza que no haya rebotes a la hora de pulsar los botones, ya que sino el usuario podría incrementar o disminuir en varias unidades la variable con tan solo una pulsación.

Esta función dispone de un switch, que en función del valor de la variable *nivel* irá actualizando la información del LCD de forma que el usuario pueda distinguir qué nivel está seleccionando. Esta información se irá mostrando en la segunda fila del mismo.

Por último, cuando el usuario pulse *start* se recibirá una señal en *PORTB.F3* y la función retornará el nivel deseado por el usuario.

3.2.3 Diagrama de flujo para la selección del número de vueltas



La función *numeroVueltas* permite al usuario seleccionar el número de vueltas que se tendrán que dar durante el juego para ganar. Este rango estará comprendido entre 1 y 255.

Para realizar esta función se utilizarán dos variables locales denominadas *vueltas* y *auxVueltas*. La primera, de tipo entero, será utilizada como contador para llevar la cuenta de las vueltas que desea el jugador. Esta variable será inicializada con un valor de 20, pues se ha considerado que una carrera de 20 vueltas es más que suficiente.

La variable *auxVueltas* es de tipo *char ** y será utilizada para transformar el valor de variable *vueltas*, de tipo entero, en una cadena de caracteres que pueda ser mostrada a través del LCD.

Cuando el usuario pulse *select1* se recibirá una señal en *PORTB.F4*, con lo que la variable *vueltas* se incrementará en una unidad. Como la variable está definida como *unsigned short*, el valor máximo que podrá alcanzar es 255. Si la variable ya tiene este valor y el usuario pulsa *select1*, automáticamente pasará a tener el valor 0. Como no puede existir una carrera con 0 vueltas, el sistema se encarga de asignar el valor 1 a la variable *vueltas*, respetando así el rango definido.

Si el usuario pulsa *select2* se recibirá una señal en *PORTC.F7*. La variable *vueltas* automáticamente disminuirá en una unidad siempre y cuando ésta sea mayor que uno, lo que impide que tome el valor cero.

Para esta función no se han utilizado los comparadores > y < puesto que el uso de los comparadores de igualdad permiten un ahorramiento de memoria.

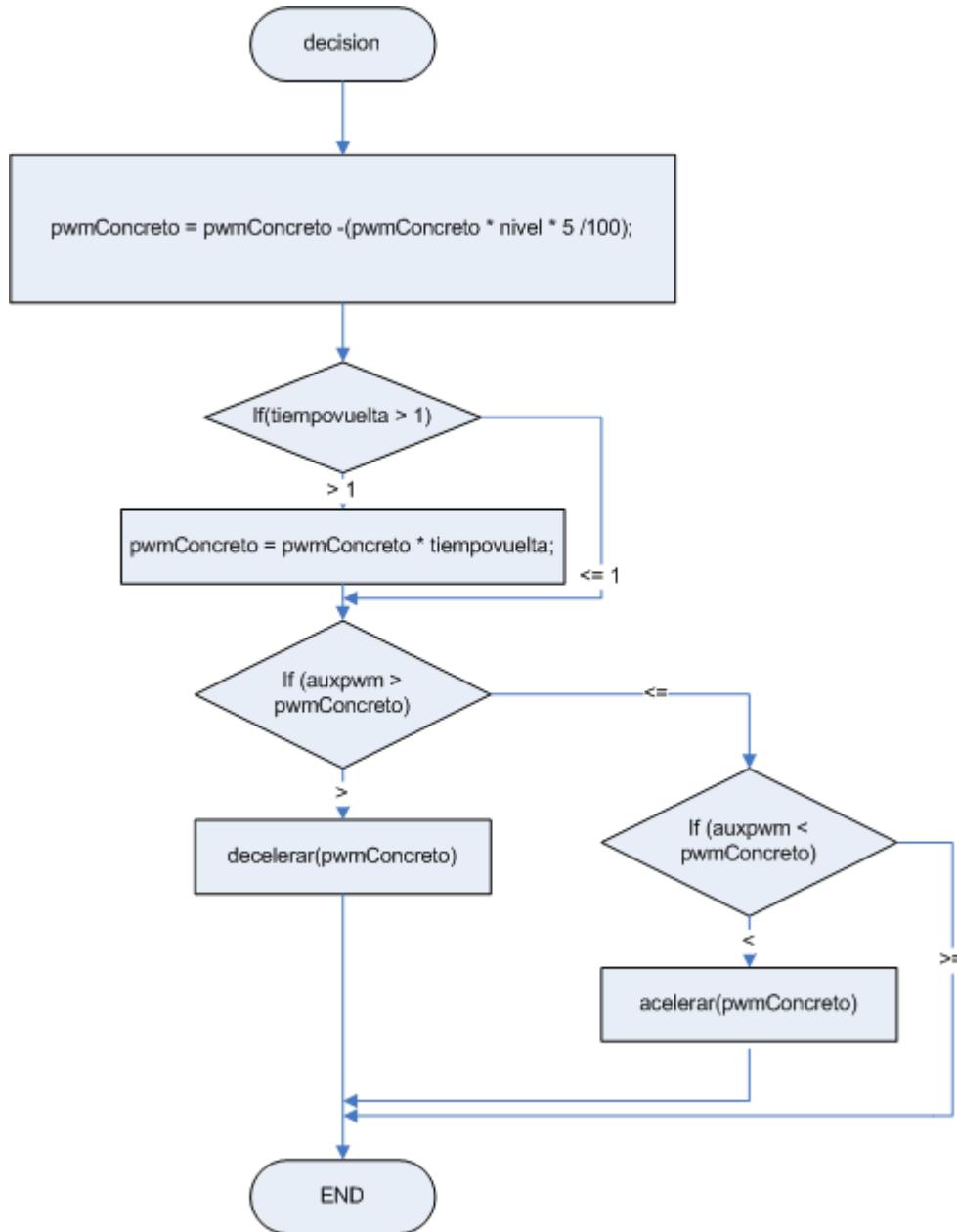
La función *delay_ms()*, al igual que en la función anterior, permite evitar los rebotes producidos por los botones.

La función *WordToStr()* permite la transformación del valor de la variable *vueltas*, de tipo entero, en una cadena de caracteres que, en este caso, será almacenado en *auxVueltas*. Esto permitirá al usuario poder ver en todo momento cuantas vueltas de juego puede seleccionar.

Una vez el usuario tenga en la pantalla el número de vueltas deseadas pulsará *start*, lo que generará una señal en *PORTB.F3* tras lo cual la función retornará el valor de *vueltas*.

3.2.4 Diagrama de flujo para el procedimiento decisión

Para la realización de este sistema es necesario contar con dos funciones (explicadas más adelante) que permitan tanto acelerar como decelerar el coche. Para una mayor comodidad a la hora de diseñar se ha decidido crear un procedimiento denominado decisión. Este será el encargado de decidir si a la hora de variar la velocidad del coche el sistema debe de acelerarlo o ralentizarlo.



Este procedimiento recibirá como primer parámetro *pwmConcreto*, que representa el valor que se quiere asignar al duty del PWM. A efectos prácticos hace que el coche acelere o decelere de una forma proporcional.

El segundo parámetro que recibirá será el nivel, ya que la velocidad del coche debe de variar según el nivel para aumentar o disminuir la velocidad.

Como se puede observar al comienzo del procedimiento se reajusta *pwmConcreto*, es decir, la velocidad a la que tendrá que ir el coche según el nivel. Teniendo en cuenta que *nivel* tomará los valores comprendidos entre 0 y 2, se ha desarrollado una fórmula que permite ajustar la velocidad del coche partiendo de la máxima. En función del nivel se tomarán los siguientes valores:

- Nivel 1: la variable *nivel* valdrá 0, por lo que *pwmConcreto* tomará el 100% del valor.
- Nivel 2: la variable *nivel* valdrá 1, por lo que *pwmConcreto* tomará el 95% del valor.
- Nivel 3: la variable *nivel* valdrá 2, por lo que *pwmConcreto* tomará el 90% del valor.

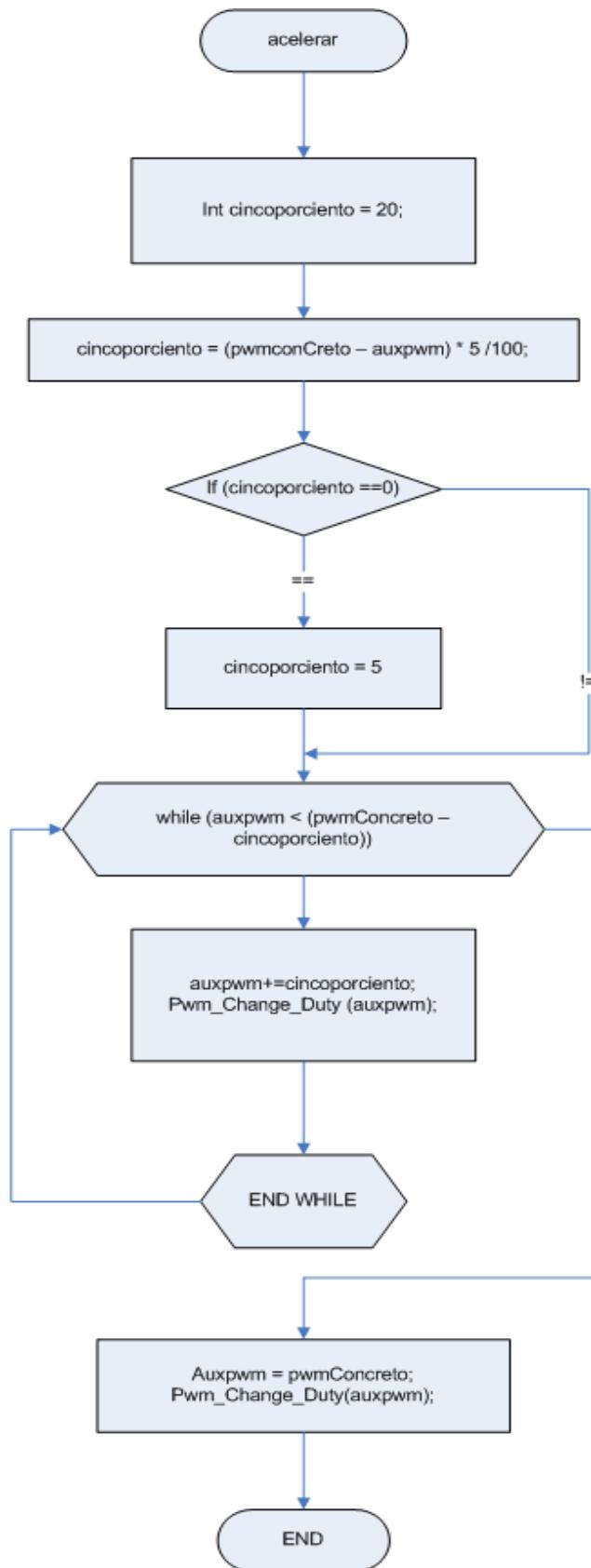
Si la última vuelta que ha dado el coche supera el tiempo estimado, la variable global *tiempovuelta* tomará un valor superior a uno. Cuando esto suceda, se debe ajustar la velocidad del coche. Para ello, el *dutty* será ajustado multiplicando el valor del *pwmConcreto* por el valor de la variable *tiempovuelta*.

Una vez obtenido el valor del PWM que se quiere asignarle, el sistema deberá comprobar si la velocidad actual del coche es mayor o menor que la que se desea imprimir. Para ello, se utilizará la variable *auxpwm*. Ésta tiene que ser de tipo global, ya que el compilador no permite pasar variables por referencia, por lo que se debe tratar con especial cuidado.

Si *auxpwm* es mayor que la velocidad que se desea imprimir, es decir mayor que *pwmConcreto*, se invocará el procedimiento *acelerar*. Por el contrario, si es menor, se invocará el procedimiento *decelerar*.

En caso de que sean iguales, el sistema no debe hacer nada ya que el coche estará circulando con la velocidad deseada.

3.2.5 Diagrama de flujo para el procedimiento acelerar



Para aumentar la velocidad del coche ha de tenerse en cuenta que ésta no puede incrementarse de golpe. Si se toma como ejemplo cualquier vehículo que circula por la carretera, es imposible que pase de 0 Km/h a 100 Km/h de forma instantánea, ya que entonces, se descontrolaría y acabaría saliéndose del trazado. Como las leyes de la física también afectan al coche eléctrico, ha de tenerse en cuenta que se debe construir una rampa de aceleración, es decir, que la velocidad del coche aumente gradualmente.

Se ha decidido ir aumentando la velocidad del coche en rangos del 5 % del total de la diferencia de velocidad que se desea imprimirlle y la velocidad actual. Este valor será almacenado en la variable local *cincoporcientos*.

Existen casos en los que la división entera podría dar como resultado cero, lo que provocaría que el sistema no pudiera salir en el segundo bucle.

Por ejemplo: si *pwmConcreto* valiese 125 y *auxpwm* 100, la fórmula quedaría como $(105 - 100) * 5 / 100$ dando como resultado $25 / 100$ o 0,25. Al tratarse de una variable entera el compilador automáticamente despreciaría los decimales y se quedaría como cero.

Esto sólo ocurre cuando la diferencia de velocidad es muy pequeña, por lo que para este caso se ha previsto que *cincoporcientos* tome el valor 5.

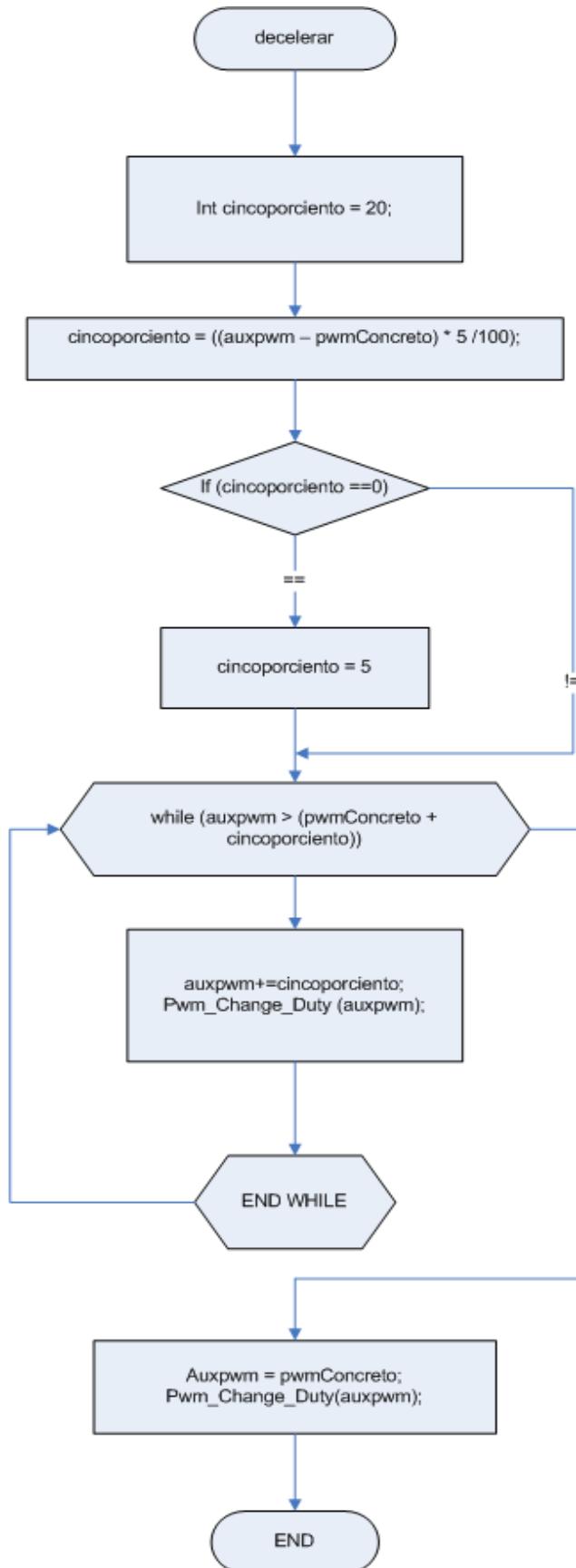
Este procedimiento recibe como parámetro *pwmConcreto*, que al igual que en el procedimiento anterior representa el dutty que se desea asignarle al PWM. También se utiliza la variable *auxpwm* de tipo global, ya que el compilador no permite el paso de variables por referencia.

Después de calcular el 5% se inicia un bucle que permite ir acelerando el coche. Para ello modificamos la variable *auxpwm* aumentándola con el 5% calculado anteriormente.

Con la instrucción *Pwm_Change_Dutty()* se consigue modificar el dutty del control del ancho de pulso y en este caso aumentar la velocidad del coche.

Finalmente, tras salir del bucle, el sistema imprime la velocidad deseada exacta. Esto se debe a que como *cincoporcientos* no siempre es el 5% exacto al final del bucle *while*, podría darse el caso de tener que incrementar *auxpwm* sólo 1,2,3 o 4 unidades. De esta forma, el sistema garantiza que la velocidad deseada es la que será aplicada.

3.2.6 Diagrama de flujo para el procedimiento decelerar



El procedimiento decelerar realiza la operación inversa del procedimiento acelerar. Para ello, el sistema calcula el 5% de la diferencia de la velocidad actual menos la que se desea imprimirle. Al igual que en el procedimiento anterior, en caso de que la división sea cero, el sistema tomará el valor 5 por defecto.

A continuación se irá disminuyendo la velocidad hasta alcanzar la deseada.

Para llevar a cabo esta acción, este procedimiento recibe como parámetro un entero denominado *pwmConcreto*, que representa la velocidad que se desea imprimir al coche controlado por el sistema.

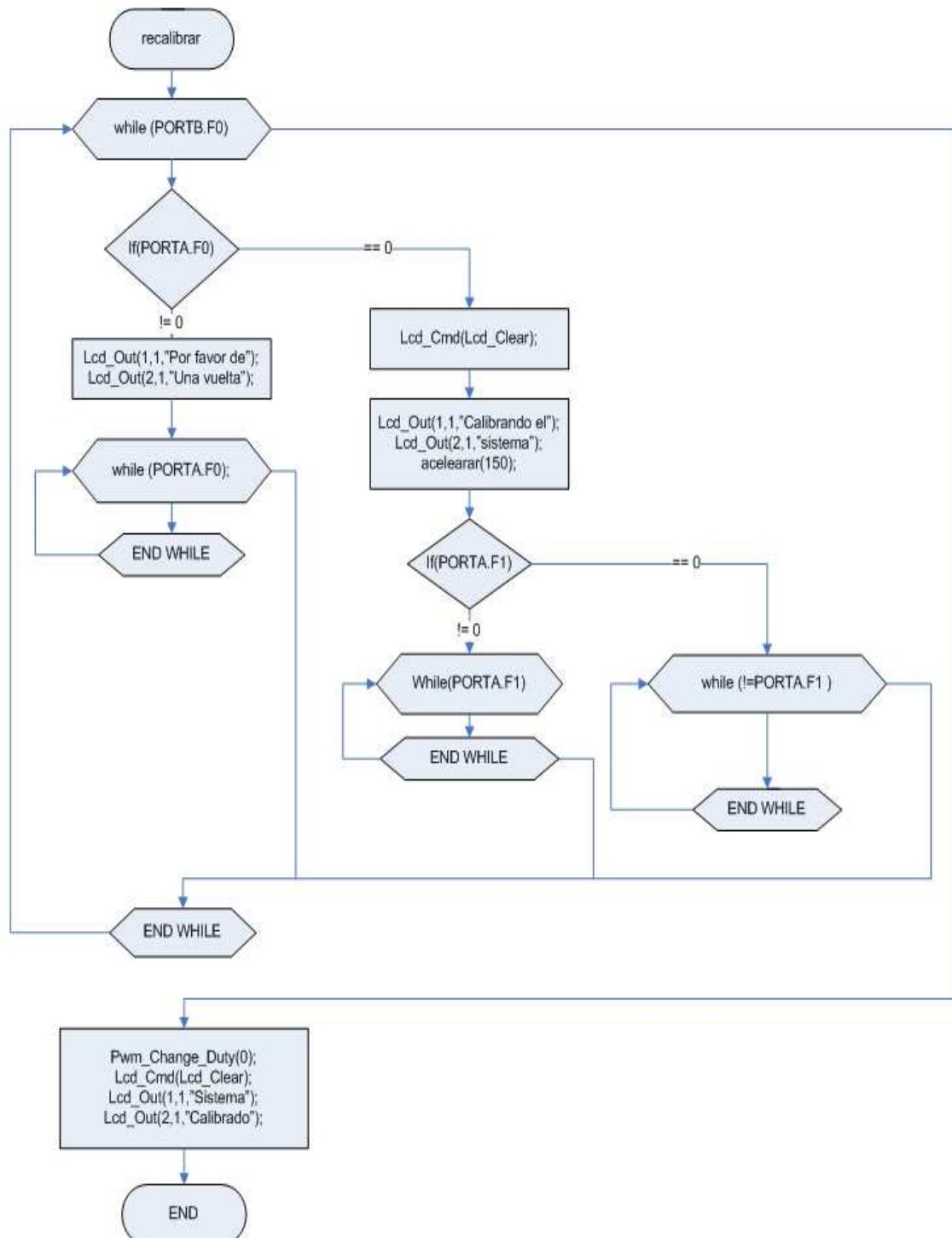
También se utiliza la variable *auxpwm* de tipo global que, representa la velocidad actual del coche en todo momento.

Debe recordarse que la instrucción *Pwm_Change_Dutty()* sirve para hacer efectivo este cambio.

Con este algoritmo quedará garantizado que el coche pueda ir decelerando paulatinamente, evitando así los cambios bruscos de velocidad que lo desestabilizarían.

3.2.7 Diagrama de flujo para el procedimiento recalibrar

Estudiando los primeros diagramas realizados durante la fase de análisis, se puede concluir que es posible la creación de un único procedimiento que permita calibrar los sensores para un correcto funcionamiento del sistema.



El funcionamiento del procedimiento es el siguiente:

El programa estará en un bucle *while* mientras exista una señal en *PORTB.F0*. Se debe tener en cuenta que todos los sensores están conectados a través de la puerta OR (diodos) que está conectada a su vez en *PORTB.F0*. Por lo tanto, en caso de que algún sensor estuviera activo se recibiría una señal en dicha entrada que implicaría que algún sensor tiene que ser calibrado.

Para facilitar el calibrado, se han tenido en cuenta dos posibilidades dependiendo de si el sensor o los sensores que hay que recalibrar están en una pista u otra.

■ **Pista del jugador:** En caso de que el sensor activo estuviera en la pista del jugador, el sistema le pedirá a través del display que el jugador de una vuelta. Teóricamente bastaría con colocar el coche detrás del sensor y pasarlo por encima una sola vez. Para comprobar que el sensor activo es éste, se efectuará la operación *if (PORTA.F0)*, ya que en *PORTA.F0* se encuentra conectado el sensor para la pista del jugador. Una vez se ha detectado que está activo, el sistema deberá esperar a que el jugador lo desactive, tras lo que el sistema comprobará si se tiene que calibrar algún otro sensor.

■ **Pista controlada por el sistema:** En caso de que algún sensor estuviera activo en esta pista, el sistema tendría que dar una vuelta con el coche. Esta acción producirá que se calibren todos los sensores situados en la pista controlada por el sistema, por lo que no haría falta comprobar cuál es el sensor exacto que se debe ajustar.

Si existe algún sensor en estado activo se producirá una señal en *PORTB.F0* pero no en *PORTA.F0*. Una vez detectada esta señal, el sistema mostrará que se está calibrando a través del display.

Para confirmar que el coche ha dado una vuelta completa, el sistema utilizará el sensor que cuenta las vueltas, conectado en *PORTA.F1*. Al no ser posible conocer cuál es el estado de éste sensor (activo o pasivo) se deberán tener en cuenta dos opciones:

- **Activo:** la vuelta se completará cuando pase a estado pasivo. Es decir, si el sistema está recibiendo una señal en *PORTA.F1*, deberá esperar hasta que deje de recibirla. Esto se producirá una vez que el coche haya pasado por encima del sensor y lo desactive.
- **Pasivo:** estaría calibrado, por lo tanto se deberá esperar a que se ponga en estado activo. Es decir, el sistema deberá esperar hasta que reciba una señal en *PORTA.F1*. Una vez se produzca esta señal, el coche habrá realizado una vuelta completa y todos los sensores habrán sido calibrados. Como el sensor que cuenta las vueltas quedaría activo, el programa lo desactivará entrando por el caso de Activo sin tener que dar una vuelta.

Finalmente, los sensores quedarán calibrados y así el sistema podrá frenar el coche por completo. No será necesario utilizar la función decelerar, pues el coche no circulará a gran velocidad.

3.2.8 Diagrama de flujo para el procedimiento interrupt

El compilador MikroC facilita el uso de una función que permite programar interrupciones de una forma sencilla.

El procedimiento a utilizar es el siguiente:

```
void interrupt ()  
{  
    //código añadido  
    //por el programador  
}
```

Esto implica que la función no puede recibir parámetros, lógicamente, pues esta función se ejecutará tanto si la interrupción es producida por una entrada como por un temporizador.

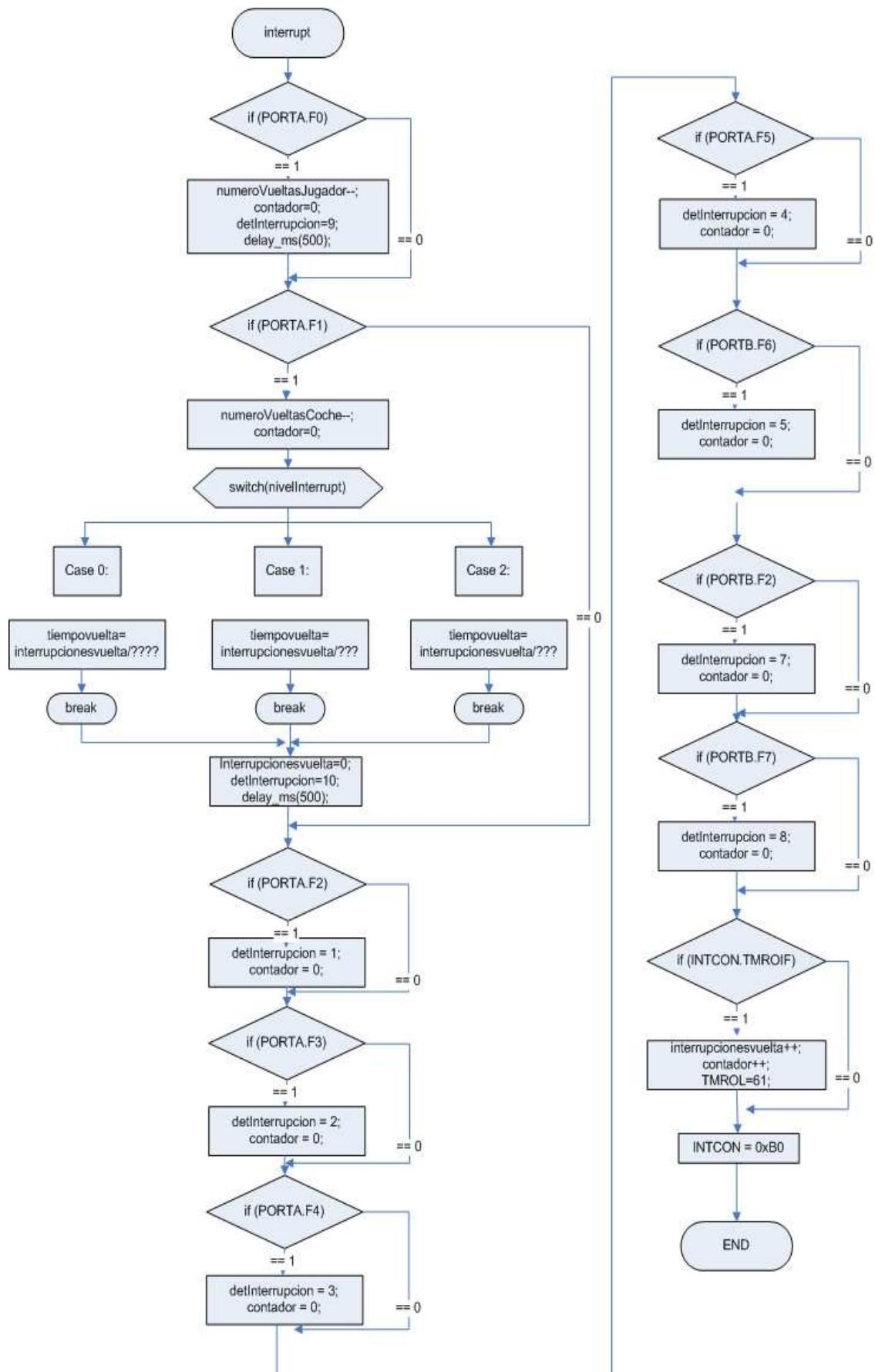
El inconveniente de tener que utilizar esta función es que todas las variables que sean usadas en el cuerpo de la misma y en otras funciones o procedimientos tienen que ser obligatoriamente declaradas de forma global.

En un principio se pensó en diseñar el sistema de forma que la velocidad fuese alterada en la propia interrupción de forma que, tras detectar que sensor habría producido la interrupción, se llamaría directamente al procedimiento decisión.

El problema de este planteamiento es que el procedimiento decisión utiliza los procedimientos acelerar y decelerar y estos a su vez invocan la instrucción *Pwm_Change_Duty()*. El compilador solamente permite el uso de esta instrucción o bien, en el procedimiento principal (*main*) y funciones llamadas, o bien en las interrupciones y funciones llamadas desde el mismo, ya que sino se acabaría consumiendo la pila. Por lo tanto, y como forzosamente acelerar se tiene que invocar desde el procedimiento principal (ya que sino sería imposible empezar a mover el coche) este enfoque quedó descartado.

La segunda forma que se planteó para la resolución de este problema fue la siguiente. En el procedimiento *interrupt* se utilizará una variable a modo de flag. Se empleará una variable de tipo entero de forma que tras identificar cuál ha sido el sensor que produce dicha interrupción le sea asignado un código.

Por lo tanto el diagrama de flujo de esta función quedará de la siguiente manera:



Antes de analizar el código debe tenerse en cuenta el uso de variables globales.

La variable *numeroVueltasJugador* contendrá el valor del número de vueltas restantes que tiene que realizar el jugador.

La variable *numeroVueltasCoche* contendrá el valor del número de vueltas restantes que tiene que realizar el coche controlado por el sistema.

La variable *contador* será una variable utilizada para controlar el tiempo de forma que debe resetearse cada vez que se reciba una señal proveniente de algún sensor situado en la pista del coche controlado por el sistema. En caso de que esta variable superase el valor 200 se produciría un fallo.

La variable *detInterrupción* será la empleada como flag, de forma que tras identificar la interrupción le sea asignado un código.

La variable *nivelInterrupt* contendrá el nivel seleccionado por el jugador.

La variable *interrupcionesvuelta* servirá para contar el número de interrupciones producidas por el *TMR0* en una vuelta, de forma que el sistema pueda ajustarse teniendo en cuenta el comportamiento variable del coche. Como todavía no se conoce el tiempo que tiene que tardar el coche cuando esté frío se han utilizado interrogantes. Cuando se programe inicialmente el microcontrolador se sustituirán los interrogantes por uno, de manera que se pueda contar cuanto tarda el coche en dar una vuelta teniendo en cuenta los valores iniciales asignados para el uso del PWM.

Para la realización de la detección de las interrupciones se ha aplicado una técnica proveniente de los primeros sistemas operativos conocida como cadena de saltos. Dicha técnica consiste en detectar las interrupciones mediante software componiendo una cadena secuencial de sentencias if.

La primera sentencia *if (if PORTA.F0)* se corresponde con la interrupción producida por el sensor situado en la pista del jugador. Esta señal se producirá cuando el coche del jugador complete una vuelta. Entonces, el sistema disminuirá el número de vueltas restantes que tiene que dar este jugador y le asignará el código 9 a *detInterrupción* para que el sistema pueda ejecutar las operaciones pertinentes para tratar esa interrupción. Debe recordarse una vez más que el *delay_ms()* sirve para evitar los rebotes producidos por el sensor, ya que sino podría ocurrir que un coche produjese dos interrupciones seguidas y el sistema descontara dos vueltas en vez de una sola.

La segunda sentencia *if (if PORTA.F1)* se corresponde con la interrupción producida por la señal del sensor situado en la pista controlada por el sistema, encargado de verificar cuando el coche dirigido por el sistema completa una vuelta. Tras producirse la señal que active la interrupción, el sistema disminuirá el valor de *numeroVueltasCoche* en una unidad, como esta variable contiene el número de vueltas restantes que le quedan por dar y se le asignará el código 10 a *detInterrupción*. Se debe destacar que el sistema debe resetear la variable *contador* para que no se alcance el estado falta.

El código utilizado por estas dos interrupciones será empleado para detectar cuando se debe actualizar la información relativa a las vueltas restantes en el LCD.

La interrupción producida por el *TMR0* se utilizará para aumentar en uno tanto la variable *contador*, utilizada para comprobar cuando se produce una falta, como para incrementar en uno la variable *interrupcionesvuelta*, controlando de esta manera el tiempo que tarda el coche en completar una vuelta del trazado.

El resto de las sentencias if, con excepción de la producida por el *TMR0*, se encargarán simplemente de asignarle un código a *detInterrupción*, de forma que el sistema sea capaz de discernir la posición y por tanto distinguir cuando debe acelerar o decelerar el coche, reseteando el valor de la variable contador para que no se produzca una falta.

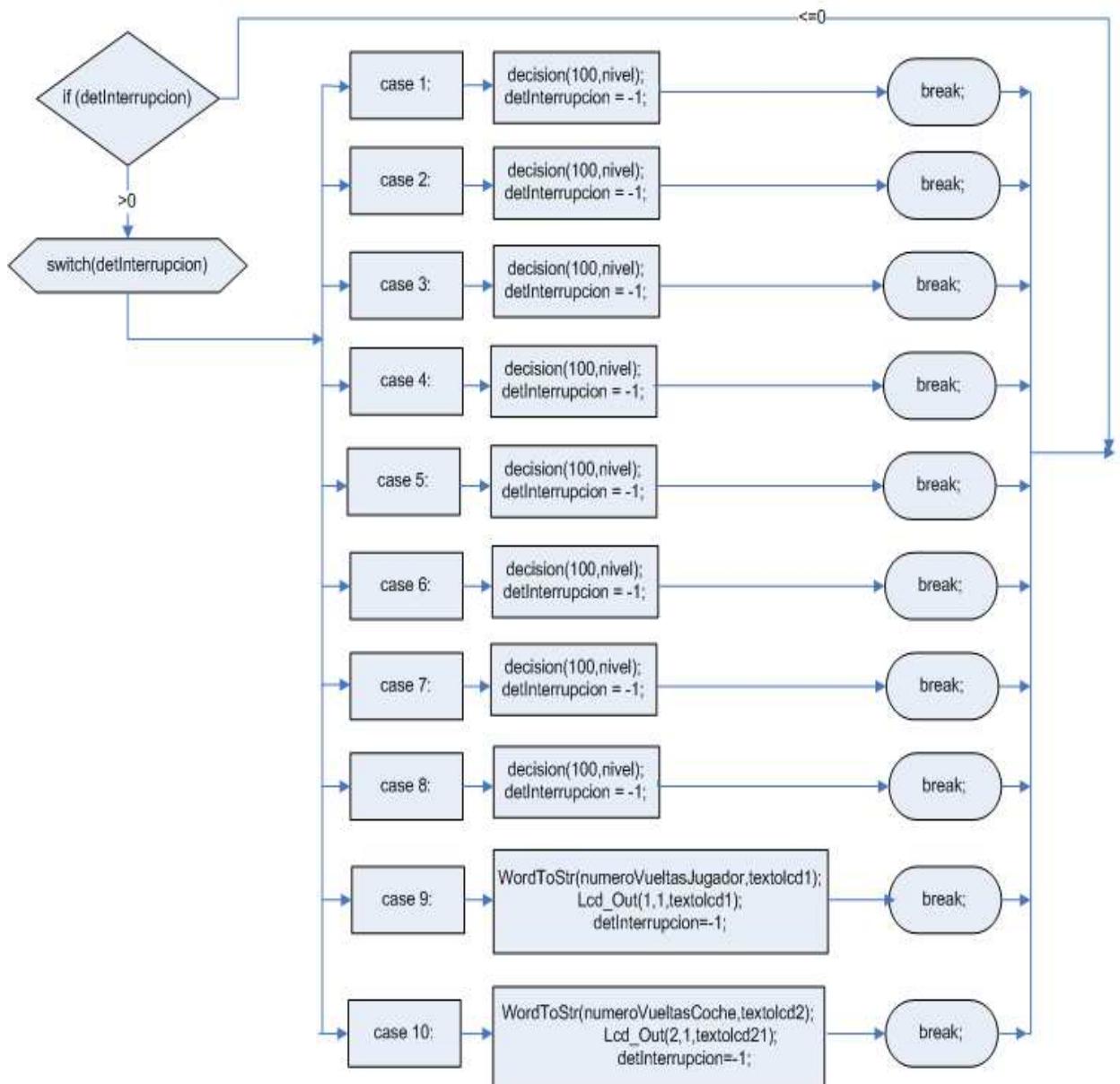
Finalmente el sistema debe volver a resetear los valores del registro *INTCON*, ya que se habrán activado los flags del *INT0* o del *TMR0*.

3.2.8 Diagrama de flujo para la detección de interrupciones

La detección de interrupciones podría ser diseñada mediante un procedimiento que recibiera como parámetro *detInterrupcion*. No obstante, esto produciría que el sistema tuviera que llamar a esta función, guardar registros en la pila... ralentizando el efecto que se quiere conseguir.

Como el planteamiento inicial del modo de uso de interrupciones ya ha sido descartado y se emplea una variable a modo de flag, se prefiere sacrificar la legibilidad del código e incrustar directamente esta función en el programa principal.

Para ello el diagrama de flujo que se desarrolló es:



En caso de producirse una interrupción, la variable *detInterrupcion* tomará un valor mayor que 0.

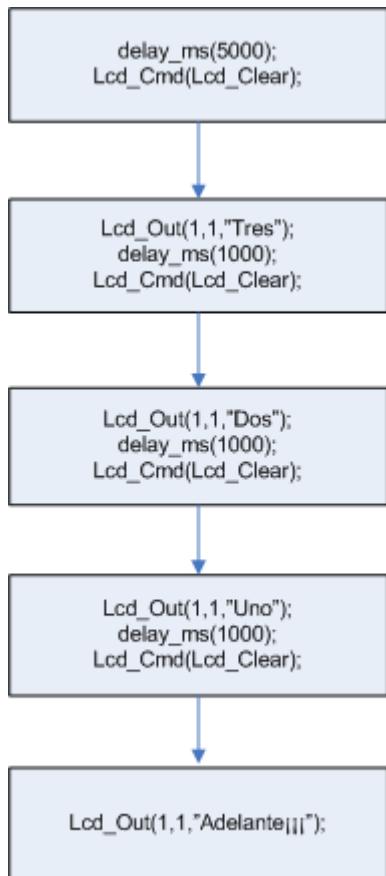
Al ser una variable cuyos valores son asignados por el propio sistema, permite la construcción de una sentencia switch en vez de sentencias if, tal y como ocurre en el procedimiento *interrupt*.

Tras determinarse la interrupción producida en el procedimiento *interrupt*, asignándole un código específico a *detInterrupción*, en este switch es posible la diferenciación de dos tipos claros de interrupciones desde un punto de vista conceptual.

- **Actualización de la información:** será producida cuando la variable *detInterrupción* tome los valores 9 y 10. Se utilizará la función *WordToStr(char *, number)* que permite transformar el valor de las variables *numeroVueltasCoche* y *numeroVueltasJugador* en una cadena de caracteres que será utilizada para imprimirla en el LCD.
- **Modificación del comportamiento del coche:** será producido cuando la variable *detInterrupción* tome valores comprendidos entre 1 y 8, ambos incluidos. Entonces, el sistema invocará al procedimiento *decision* con el dutty que se desea y con el nivel para que el dutty del control de ancho de pulso sea modificado en función de este último parámetro. De esta forma, se conseguirá variar la velocidad del coche controlado por el sistema tanto para acelerarlo como para decelerarlo.

Por último, e independientemente del tipo de interrupción, se reseteará el valor de *detInterrupción* retornándolo a -1 y mientras, el sistema seguirá esperando a que se produzca una nueva interrupción.

3.2.9 Diagrama de flujo para la cuenta atrás

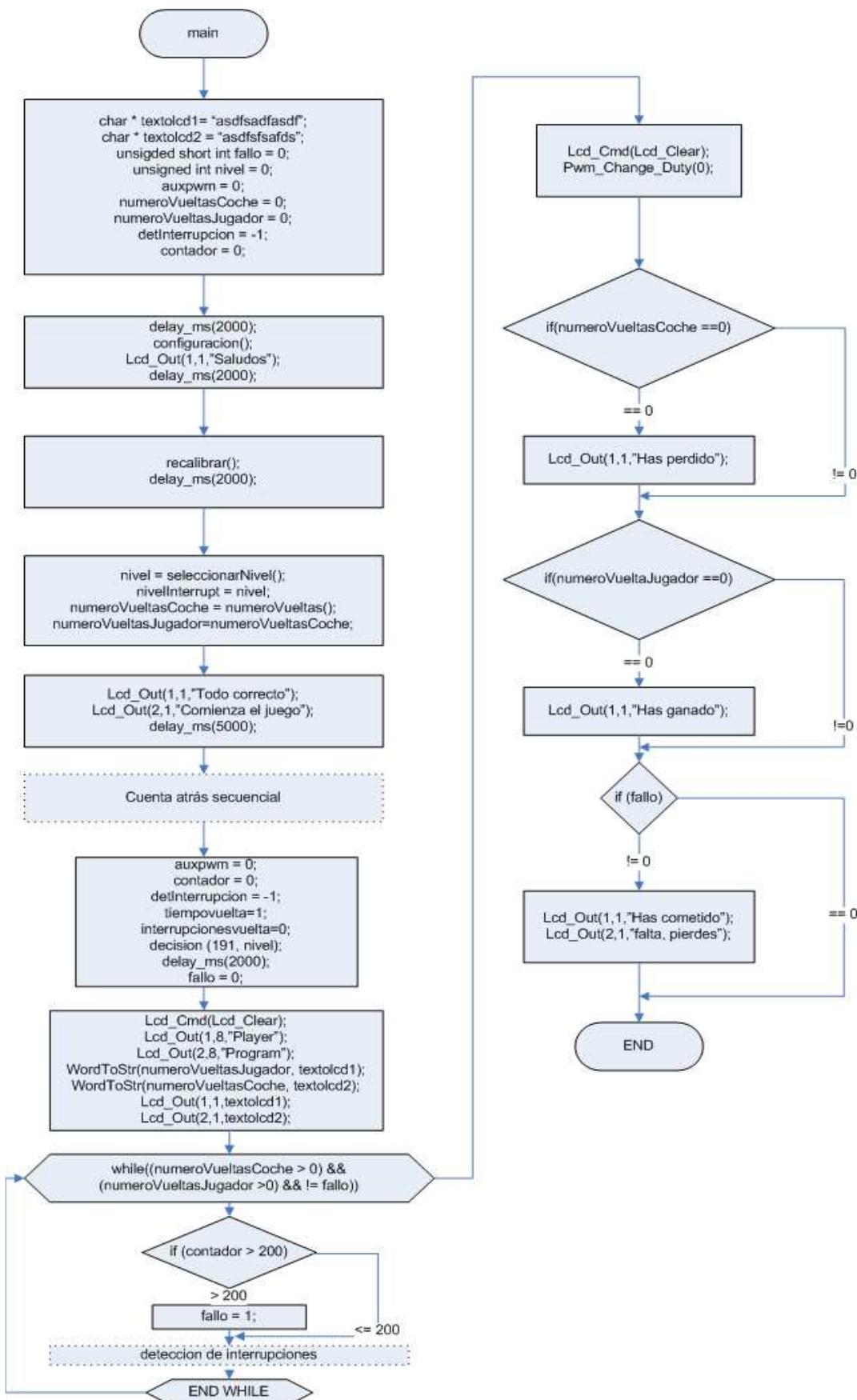


Una vez el sistema confirme que los sensores han sido calibrados y que está preparado para comenzar la carrera se creará una cuenta atrás secuencial de la siguiente forma:

- El sistema mostrará en el LCD la palabra “Tres” y esperará 1 segundo.
- El sistema mostrará en el LCD la palabra “Dos” y esperará 1 segundo.
- El sistema mostrará en el LCD la palabra “Uno” y esperará 1 segundo.
- El sistema mostrará en el LCD la palabra “Adelante” y podrá dar comienzo la carrera.

Una vez comience la carrera el sistema deberá empezar a acelerar el coche controlado por el mismo.

3.2.11 Diagrama de flujo para el programa principal



Para el programa principal se utilizarán las siguientes variables:

- **textolcd1**: variable local empleada para mostrar información en la primera fila del LCD.
- **textolcd2**: variable local empleada para mostrar información en la segunda fila del LCD.
- **fallo**: variable local empleada para detectar cuando se produce una falta por parte del jugador.
- **nivel**: variable local empleada para almacenar el nivel que selecciona el usuario.
- **auxpwm**: variable global empleada para almacenar el dutty (velocidad) que lleva en todo momento el coche controlado por el sistema.
- **numeroVueltascoche**: variable global empleada para almacenar el número de vueltas restantes que le quedan por dar al coche controlado por el sistema.
- **numeroVueltasJugador**: variable global empleada para almacenar el número de vueltas restantes que le quedan por dar al coche controlado por el jugador.
- **detInterrupción**: variable global empleada para determinar el tipo de interrupción producida por un sensor.
- **contador**: variable global empleada para controlar el tiempo y ver por lo tanto si se produce una falta.
- **tiempovuelta**: variable global utilizada como factor de ajuste del coche en función del tiempo que tarde en dar una vuelta.
- **nivellInterrupt**: variable global utilizada para que el nivel pueda ser consultado en el procedimiento interrupt.
- **Interrupcionesvuelta**: variable global utilizada para contar el número de interrupciones producidas durante una vuelta.

Tras encender la placa se comenzará a ejecutar el main. Lo primero que se debe ejecutar es un *delay_ms(int)* de dos segundos de duración, que proporcionará el tiempo suficiente para que los elementos electrónicos del circuito se alimenten correctamente.

Una vez hecho esto, se procederá a ejecutar el procedimiento configuración. Esto permitirá configurar tanto los registros como las entradas y salidas del microcontrolador correctamente.

A continuación el sistema saludará al jugador imprimiendo en el display la palabra “Saludos”.

El procedimiento *recalibrar* garantizará que todos los sensores funcionan correctamente tras su invocación.

Después de llevar a cabo la ejecución de este procedimiento, se activarán las interrupciones. No se deben activar en la configuración inicial del programa porque en caso de que un sensor estuviera activo, la pila del microcontrolador acabaría agotándose y se resetearía el programa continuamente.

Una vez hecho esto, el usuario deberá seleccionar el nivel con el que desea jugar. Para ello se invocará la función *seleccionarNivel*, que devolverá el nivel seleccionado por el usuario que será almacenado en las variables *nivel* y *nivelInterrut*.

Después, el sistema invocará a la función *numeroVueltas* de forma que el usuario podrá seleccionar el número de vueltas que considere oportuno para la realización de la carrera. Esta función retornará un valor que será almacenado tanto en *numeroVueltasCoche* como en *numeroVueltasJugador*, variables que serán empleadas para controlar el número de vueltas restantes tanto por parte del jugador como por parte del sistema.

Una vez completada la toma de datos necesaria para el desarrollo del juego, se informará al usuario de que todo está correcto y que comienza el juego, tras lo que seguirá la cuenta atrás secuencial.

Una vez termine esta cuenta, se debe asegurar que las variables globales contienen el valor adecuado, por lo que se volverán a resetear con sus valores iniciales. Entonces el coche comenzará a acelerar con una velocidad determinada en función del nivel seleccionado, invocando por tanto la función *decisión*.

Durante el juego, en el LCD se irá mostrando en todo momento las vueltas restantes tanto para el jugador como para el sistema de la siguiente forma:

- Primera fila del display: se mostrará el mensaje “Player” para referirse al jugador acompañado de su número de vueltas restantes.
- Segunda fila del display: se mostrará el mensaje “Program” para referirse al sistema acompañado de su número de vueltas restantes.

El juego deberá estar ejecutándose hasta que el jugador gane, el sistema gane o se produzca una falta, por lo que la estructura más adecuada es un bucle while. Dentro de este bucle se irá determinando el tipo de interrupción que se producirá y por lo tanto la acción que se deberá llevar a cabo.

Por último, tras salir del bucle, el coche controlado por el sistema será frenado por completo, y se comprobará porque se ha roto este bucle, aplicando tres sentencias de tipo if. La rotura puede producirse por los siguientes casos:

- **numeroVueltasCoche toma valor cero:** implica que el coche controlado por el sistema ha completado todas las vueltas antes que el jugador, por lo que el sistema imprimirá en el LCD el mensaje “Has perdido”:

- **numeroVueltasJugador toma valor cero:** implica que el coche controlado por el jugador ha completado todas las vueltas antes que el coche controlado por el sistema, por lo que se imprimirá en la pantalla el mensaje “Has ganado”.
- **fallo toma un valor superior a cero:** implica que el coche controlado por el sistema se ha salido de la pista y se ha producido una falta, por lo tanto el jugador pierde. A continuación se imprimirá el mensaje “Has cometido falta, pierdes”.

4. Fase de Implementación de Software

Para la implementación de software se ha utilizado el entorno de desarrollo MikroC, que permite la generación del fichero .hex que será cargado en el microcontrolador.

4.1 Implementación para la selección de nivel

```
unsigned short int seleccionarNivel(){
    short int nivel = 0;
    Lcd_Cmd(Lcd_Clear);
    Lcd_Out(1,1,"Seleccione nivel");

    while (!PORTB.F3){ //pulsar boton start

        if (PORTB.F4)
        {
            nivel = (++nivel ) % 3; //aumentar pero solo hasta 2
            delay_ms(900);
        }
        if (PORTC.F7)
        {
            if (--nivel < 0 ) nivel = 2; //volvemos al nivel 2 despues del 1
            delay_ms(900);
        }
        switch (nivel)
        {

            case 0 : Lcd_Out(2,1,"Nivel 1");
            break;
            case 1: Lcd_Out(2,1,"Nivel 2");
            break;
            case 2: Lcd_Out(2,1,"Nivel 3");
            break;
        }

    }
    return nivel;
}
```

El programa entra en un bucle en el que se permite ir aumentando o disminuyendo el valor de la variable *nivel* hasta que el botón Start es pulsado y genera una señal de entrada en *PORTB.F3*.

Una vez se sale del bucle se retorna el nivel seleccionado.

4.2 Implementación para la selección del número de vueltas

```
unsigned short int numeroVueltas()
{
    unsigned short int vueltas = 20;
    char * auxVueltas="200";

    Lcd_Cmd(Lcd_Clear);
    Lcd_Out(1,1,"Vueltas de juego?");
    delay_ms(1000); //esperamos 1 segundo para evitar los rebotes del pulsador

    while (!PORTB.F3) //boton star
    {

        if (PORTB.F4)
        {
            vueltas++; //aumentar
            if (vueltas == 0) vueltas =1;
            delay_ms(900);
        }
        //disminuir si es mayor que 1
        if (PORTC.F7)
        {
            if (vueltas != 1 ) vueltas--;
            delay_ms(900);
        }

        WordToStr(vueltas,auxVueltas);
        Lcd_Out(2,1,auxVueltas);

    }
    return vueltas;
} //fin numeroVueltas
```

Al igual que la función anterior, esta función entra en un bucle en el que el usuario puede ir aumentando y disminuyendo el valor de vueltas, siempre y cuando este valor oscile entre 1 y 255, hasta que el botón Start sea pulsado y genere de nuevo la señal de entrada en *PORTB.F3*.

Una vez se salga del bucle se retornará el valor de la variable vueltas.

4.3 Implementación del Juego

Como se ha mencionado durante la fase de diseño de software, se han creado distintas funciones y distintos procedimientos que permiten la implementación del mismo.

Para seguir un orden, las implementaciones de las funciones y los procedimientos se irán exponiendo siguiendo la misma cronología que en la fase de diseño.

4.3.1 Implementación del procedimiento decisión

Este procedimiento es el encargado de conocer la velocidad a la que se encuentra el coche y que velocidad debe imprimírsela, de forma que automáticamente se invoque al procedimiento encargado de acelerarlo y decelerarlo.

```
void decisión (int pwmConcreto, int nivel)
{
    pwmConcreto = pwmConcreto - (pwmConcreto * 2 * nivel /100);
    // utiliza el factor de corrección que sera calculado en la interrupción
    if (tiempoVuelta > 1)
        pwmConcreto = pwmConcreto * tiempoVuelta;
    if (auxpwm > pwmConcreto)
        decelerar(pwmConcreto);
    else if (auxpwm < pwmConcreto)
        acelerar(pwmConcreto);
}
```

Debe recordarse que *tiempoVuelta* es una variable global, que se modifica mediante el uso de interrupciones y que contiene el factor de ajuste para que la diferencia de potencial aplicada en coche sea aumentada al ir disminuyendo su velocidad por el calor producido en el motor eléctrico.

4.3.2 Implementación del procedimiento acelerar

```
void acelerar(int pwmConcreto)
{
    int cincoporciento = (pwmConcreto - auxpwm) *5/100;
    //a veces la division puede dar cero
    if (cincoporciento ==0) cincoporciento = 5;
    while (auxpwm < (pwmConcreto- cincoporciento))
    { auxpwm+=cincoporciento;
        Pwm_Change_Duty(auxpwm);
    }
    auxpwm = pwmConcreto;
    Pwm_Change_Duty(auxpwm);
}
```

Debe tenerse en cuenta que la variable *aux pwm*, de tipo global, contiene el valor que hace referencia al *dutty* que en todo momento se le está imprimiendo al coche.

4.3.3 Implementación del procedimiento decelerar

```
void decelerar(int pwmConcreto)
{
    int cincoporiento = ((aux pwm - pwmConcreto) *2/100);

    //a veces la division de arriba puede dar cero
    if (cincoporiento == 0) cincoporiento = 5;

    while(aux pwm > (pwmConcreto+cincoporiento))
    {
        aux pwm -= cincoporiento;
        Pwm_Change_Duty(aux pwm);
    }
    aux pwm = pwmConcreto;
    Pwm_Change_Duty(aux pwm);

}
```

Al igual que el procedimiento anterior, en éste también es utilizado y modificado el valor de *aux pwm*.

4.3.4 Implementación del procedimiento recalibrar

```
void recalibrar()
{
    while (PORTB.F0) //existe una interrupción y hay que calibrar
    {
        if(PORTA.F0) //calibrar el sensor del jugador
        {
            Lcd_Out(1,1,"Por favor de");
            Lcd_Out(2,1,"una vuelta");
            while(PORTA.F0){}//esperando
        }
        else //ahora solo hay que calibrar la pista del jugador
        { Lcd_Cmd(Lcd_Clear);
            Lcd_Out(1,1,"Calibrando el");
            Lcd_Out(2,1,"sistema");
            acelerar(165);
            if(PORTA.F1) // si esta encendido
            { while(PORTA.F1){} //esperamos a que se apague
            }
            else //esta apagado y esperamos a que se encienda y luego se apague
            {while(!PORTA.F1) {}
            }
        }
    }
}
```

```

        }

Pwm_Change_Duty(0); //frenamos el coche
Lcd_Cmd(Lcd_Clear);
Lcd_Out(1,1,"Sistema");
Lcd_Out(2,1,"calibrado");

}

```

El procedimiento *recalibrar* se encarga de ajustar los sensores magnéticos, ya que puede darse el caso de que alguno se encuentre en estado activo.

4.3.5 Implementación del procedimiento Interrupt

```

void interrupt()
{
    INTCON.GIE=0; //desactivamos el resto de las interrupciones

    if (PORTA.F0) //sensor vueltas jugador
    {
        numeroVueltasJugador--;
        detInterrupcion=9;
        delay_ms(200);

    }
    if (PORTA.F1) //sensor vueltas coche
    {
        numeroVueltasCoche--;
        contador=0; //reseteamos contador porque paso por un sensor
        switch (nivellInterrupt)
        {
            case 0: //tarda ???
                tiempovuelta = 1;
                break;
            case 1://tarda ???
                tiempovuelta = 1;
                break;

            case 2: //tarda ???
                tiempovuelta = 1;
                break;
        }
        interrupcionesvuelta=0;
        detInterrupcion=10;
        delay_ms(200);

    }
    if (PORTA.F2)

```

```

{
    detInterrupcion=1;
    contador =0;
}
if (PORTA.F3)
{
    detInterrupcion=2;
    contador=0;
}
if (PORTA.F4)
{
    detInterrupcion=3;
    contador=0;
}
if (PORTA.F5)
{
    detInterrupcion=4;
    contador=0;
}
if (PORTB.F6)
{
    detInterrupcion=5;
    contador=0;
}
if (PORTB.F1)
{
    detInterrupcion=6;
    contador=0;
}
if (PORTB.F2)
{
    detInterrupcion=7;
    contador=0;
}
if (PORTB.F7)
{
    detInterrupcion=8;
    contador=0;
}
if (INTCON.TMR0IF)
{
    interrupcionesvuelta++;
    contador++; // incremento valor de contador en cada IRQ
    TMR0L = 61;
}

INTCON = 0xB0;

}

```

Debe tenerse en cuenta que todavía no es posible conocer el tiempo de una vuelta, ni por tanto el número de interrupciones que se deben producir, por lo que este procedimiento deberá ser modificado ajustando *tiempovuelta* con el valor que corresponda. Para llevar a cabo la primera prueba se dejará con valor 1, de forma que el microcontrolador pueda ser grabado y se pueda cronometrar cuando tarda el coche en dar una vuelta.

Una vez verificados los tiempos para cada nivel, se podrán calcular el número de interrupciones que se producirán.

4.4 Implementación del procedimiento configuración

Este procedimiento se encarga agrupar las instrucciones pertinentes para configurar el PIC.

```
void configuracion()
{
    TRISA=0x2F;    //puerto a entrada
    ADCON1=0x87; //puerto a digital
    TRISD = 0x00; // Puerto D salida
    TRISB= 0xDF; // Puerto B entrada
    TRISC =0x80; //Puerto C

    //configuración del LCD
    Lcd_Config(&PORTD, 0, 1, 2, 7, 6, 5, 4);
    Lcd_Cmd(Lcd_CLEAR);      // Clear display
    Lcd_Cmd(Lcd_CURSOR_OFF); // Turn cursor off

    //configuracion del TMR0
    T0CON = 0xC7;    // TMR0 modo 8bit, asigno preescalador a TMR0, div 256
    TMR0L = 61;      // inicializar Timer0
    INTCON =0x00;    //Desactivamos las interrupciones inicialmente

    //Configuracion del control de ancho de pulso
    Pwm_Init(15000);
    Pwm_Start();
    Pwm_Change_Duty(0);

}
```

Debe aclararse que el control de ancho de pulso se inicializa con una frecuencia de 15000 hertzios, ya que de ser menor sería una frecuencia audible y se escucharía un molesto pitido tras activarse el Power MOSFET.

4.5 Implementación del programa principal

```
void main() {  
  
    char * textolcd1 = "asfdasfdasfsadf"; //inicializamos valor a texto1lcd  
    char * textolcd2 = "asfasfasfasfsaff"; //inicializamos valor a texto2lcd  
    unsigned short int fallo = 0; //inicializamos valor de fallo  
    unsigned int nivel = 0; //inicializamos valor de nivel  
    aux pwm = 0; //inicializamos valor de aux pwm  
    INTCON=0x00; //desactivamos las interrupciones inicialmente.  
    numeroVueltasCoche=0; //inicializamos valor de numeroVueltasCoche  
    numeroVueltasJugador=0; //inicializamos valor de numeroVueltasCoche  
    detInterrupcion=-1; //inicializamos valor de detInterrupcion  
    contador=0; //inicializamos valor de contador  
    nivel=0;  
    delay_ms(2000); //esperamos dos segundos para que tome tensión el LCD  
                    //de la placa  
  
    configuracion(); //lanzamos la configuración inicial  
    Lcd_Out(1,1,"Saludos");  
  
    // Recalibrar los sensores hall de la pista  
    recalibrar();  
    delay_ms(2000);  
  
    //Obtención del nivel  
    nivel = seleccionarNivel();  
    nivellInterrupt = nivel;  
  
    //Obtención del numero de vueltas  
    numeroVueltasCoche=numeroVueltas();  
    numeroVueltasJugador=numeroVueltasCoche;  
    Lcd_Cmd(Lcd_Clear);  
  
    //Todo correcto  
    Lcd_Out(1,1,"Todo correcto");  
    Lcd_Out(2,1,"Comienza el juego");  
  
    //Cuenta atras secuencial  
    delay_ms(2000);  
    Lcd_Cmd(Lcd_Clear);  
    Lcd_Out(1,1,"Tres");  
    delay_ms(1000);  
    Lcd_Cmd(Lcd_Clear);  
    Lcd_Out(1,1,"Dos");  
    delay_ms(1000);  
    Lcd_Cmd(Lcd_Clear);  
    Lcd_Out(1,1,"Uno");  
    delay_ms(1000);
```

```

    Lcd_Out(1,1,"Adelante!!!");

    //Comienza el juego
    INTCON =0xB0;//activamos las interrupciones
    aux pwm=0; //reseteamos el valor para el control de ancho de pulso
    contador=0; //reseteamos el contador para el tiempo.
    detInterrupcion=-1;//reseteamos detInterrupción
    tiempovuelta=1; //inicializamos tiempovuelta a 1
    interrupcionesvuelta=0; //reseteamos interrupcionesvuelta
    delay_ms(1000);

    //Mostramos información para el jugador en el LCD
    Lcd_Cmd(Lcd_Clear);
    Lcd_Out(1,8,"Player");
    Lcd_Out(2,8,"Program");
    //Convertimos vueltas restantes a texto
    WordToStr(numeroVueltasJugador,textolcd1);
    WordToStr(numeroVueltasCoche,textolcd2);
    Lcd_Out(1,1,textolcd1);
    Lcd_Out(2,1,textolcd2);

    //Comienza a andar el coche
    decision(191,nivel);
    // Reseteamos contador por si se produjo alguna interrupción del TMR0
    contador = 0;
    fallo = 0;

    while ((numeroVueltasCoche > 0) && (numeroVueltasJugador > 0) &&
    (!fallo))
    {
        if (contador >= 200) fallo =1; //contador llega a 100 en cinco segundos

        if(detInterrupcion)
        { switch (detInterrupcion)
        { case 1 : decision(175,nivel);      //sensor 3
                    detInterrupcion=-1;
                    break;
        case 2 : decision(175,nivel);      //sensor 4
                    detInterrupcion=-1;
                    break;
        case 3 : decision(172,nivel);      //sensor 5
                    detInterrupcion=-1;
                    break;
        case 4 : decision(200,nivel);      //sensor 6
                    detInterrupcion=-1;
                    break;
        case 5 : decision(150,nivel);      //sensor 7
                    detInterrupcion=-1;
        }
    }
}

```

```

        break;
case 6 : decision(165,nivel); //sensor 8
    detInterrupcion=-1;
    break;
case 7 : decision(180,nivel); //sensor 9
    detInterrupcion=-1;
    break;
case 8 : decision(178,nivel);
    detInterrupcion=-1; //sensor 10
    break;
case 9 : IntToStr(numeroVueltasJugador,textolcd1);
    Lcd_Out(1,1,textolcd1);
    detInterrupcion=-1;
    break;
case 10 : IntToStr(numeroVueltasCoche,textolcd2);
    decision(191,nivel);
    Lcd_Out(2,1,textolcd2);
    detInterrupcion=-1;
    break;

} //fin switch

} //fin del if

} //fin del while
Lcd_Cmd(Lcd_Clear);
Pwm_Change_Duty(0); //paramos el coche
if(numeroVueltasCoche ==0)
    Lcd_Out(1,1,"Has perdido");
if(numeroVueltasJugador ==0)
    Lcd_Out(1,1,"Has ganado");
if (fallo)
{
    Lcd_Out (1,1,"Has cometido");
    Lcd_Out(2,1,"falta, pierdes");
}
}

```

Debe tenerse en cuenta que para la obtención de la implementación del main, además del diagrama referente al diseño del programa principal, se han tenido que utilizar también los diagramas de diseño de software de detección de interrupciones y de cuenta atrás secuencial.

4.6 Variables globales

Las variables globales que se han tenido que utilizar a lo largo del programa son las siguientes.

```
unsigned short int numeroVueltasJugador; // vueltas del jugador
unsigned short int numeroVueltasCoche; // vueltas del coche
unsigned long int contador; // tiempo por si el coche se saliera
int auxpwm; //duty del pwm (velocidad)
int detInterrupcion; //determinacion de la patilla en la interrupción
int nivelInterrupt; //procesamiento del tiempo en función del nivel
double tiempovuelta; //ajuste del tiempo de vuelta
unsigned int interrupcionesvuelta; //nº interrupciones por vuelta
```

Para obtener más información se recomienda ojear el archivo con la extensión .C donde se encuentra el código fuente completo con los respectivos comentarios.

Antes de pasar a la fase de pruebas debe tenerse en cuenta que el programa Proteus dispone de una herramienta denominada *Virtual Terminal*, que será empleada para poder comprobar si los algoritmos diseñados para acelerar y decelerar son correctos.

Para ello, se utilizará el puerto paralelo capaz de transmitir caracteres y la consola *Virtual Terminal*. Se ha desarrollado un procedimiento que permite que se muestren todos los caracteres a través de esta consola pasándole como parámetro una cadena de caracteres, pudiendo comprobar así cualquier valor que toman las variables.

```
void mostrarCadena(char * cadena)
{
    int i = 0;
    while (cadena[i] != '\0')
        Usart_Write(cadena[i++]);
    Usart_Write(' ');
    Usart_Write(13);
}
```

Este procedimiento utiliza la función *Usart_Write(char *)* que se encarga de enviar el carácter a través del puerto correspondiente. Para poder comprender este procedimiento debe recordarse que el carácter \0 implica el final de una cadena en C y que el carácter nº 13 de la tabla ASCII es el retorno de carro.

Debe tenerse en cuenta que al tener que usar este procedimiento, en la parte del código encargado de configurar el microcontrolador, debe añadirse la línea *Usart_Init(9600);*. Esta instrucción permite inicializar el puerto paralelo con una frecuencia de 9600 baudios, el *Virtual Terminal* debe configurarse con la misma frecuencia.

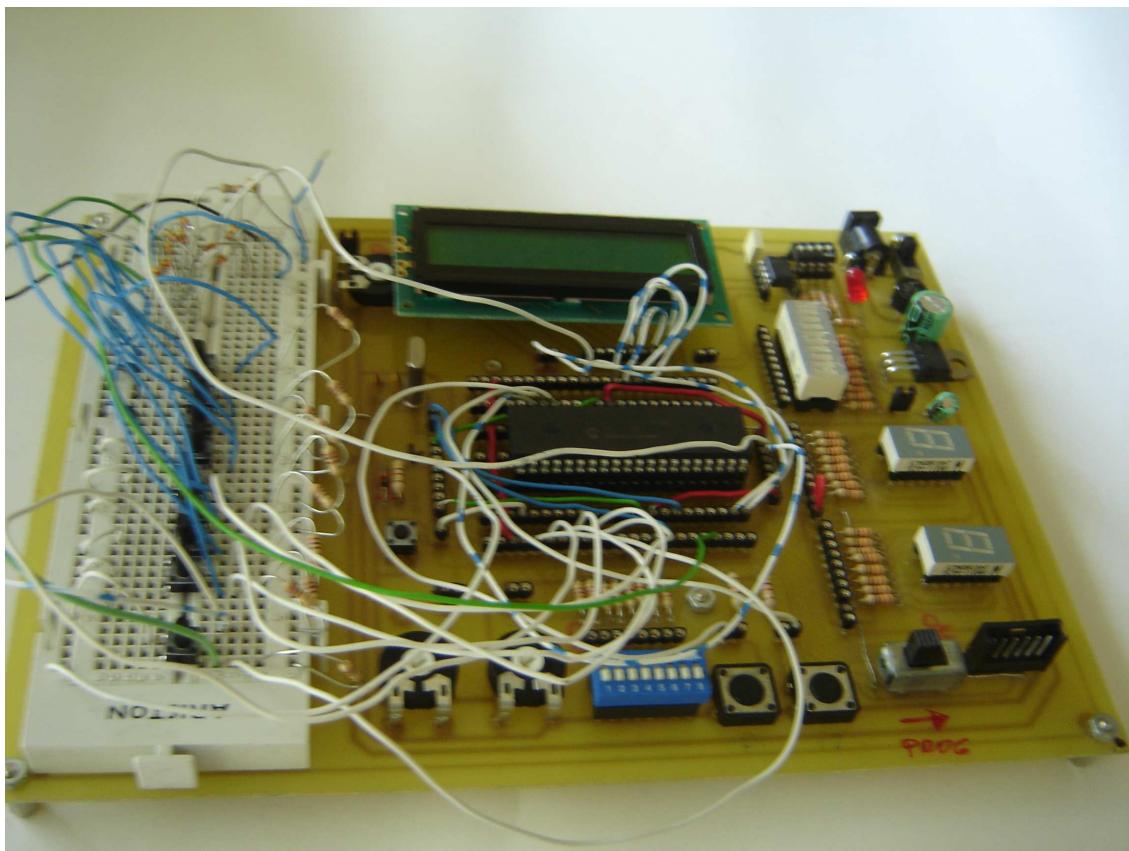
Para poder utilizar este procedimiento se ha decidido utilizar la función *IntToStr(char *, int)* y una cadena de caracteres auxiliar que será empleada para almacenar el valor del numero entero en caracteres ASCII.

5. Fase de pruebas

Durante esta fase se intentará demostrar que tanto el código implementado como el diseño del hardware son correctos.

5.1. Construcción de un prototipo

Una vez generado el diseño de la placa hardware se ha construido una placa de prototipo utilizando una placa de desarrollo.



A esta placa se le ha conectado un polímetro analógico entre la patilla de PWM del microcontrolador y masa, de forma que fuera posible la comprobación de la variación del voltaje que tendría que ocurrir al activarse o desactivarse un sensor.

Los sensores, en este caso, han sido sustituidos por botones, de manera que al pulsarse uno u otro la diferencia de potencial emitida por la patilla del PWM varíe y por tanto la aguja del polímetro oscile.

Aprovechando que la placa tiene un botón de reset (aclara que los botones de reset para el microcontrolador funcionan con una lógica inversa a la de los botones habituales) se ha testeado el reset y se ha comprobado que funciona correctamente.

La pantalla LCD disponible en esta placa no es retroluminosa, no obstante, permitirá comprobar que los mensajes que se utilizan para comunicarse con el usuario son legibles y claros. Una vez probado el software generado, se observó que el código encargado de imprimir los mensajes funcionaba correctamente.

Sin embargo tras pulsar los botones se pudo comprobar que efectivamente el voltaje varía aunque no en todos los botones pulsados. Se pudo concluir que, teniendo en cuenta que no fallaron todos los botones testeados, lo más probable es que fuera debido a que algún cable hacía mal contacto, por lo que se decidió seguir adelante con la construcción de la placa final.

Mientras tanto, se decidió que el software debería ser testeado y depurado utilizando la herramienta de simulación Isis Proteus.

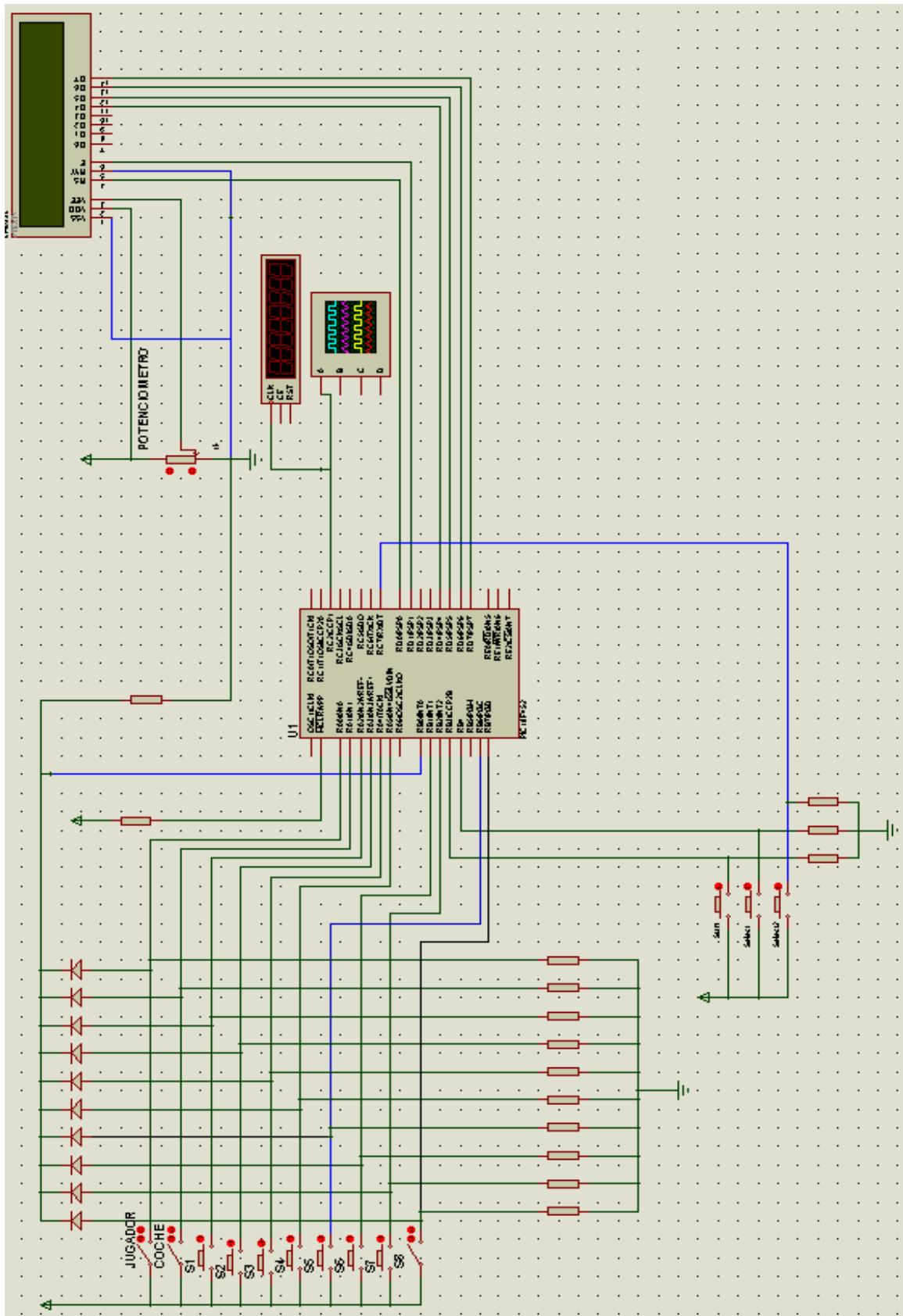
5.2 Pruebas de simulación por software

Para llevar a cabo las pruebas de simulación software se ha empleado el programa Isis Proteus, que ofrece la posibilidad de simular el comportamiento de un esquema electrónico en tiempo real.

Para ello, se ha tenido que construir un esquema que permita comprobar si el código generado es correcto. Se ha copiado el esquema lógico de la placa hardware y se han sustituido y añadido los siguientes elementos:

- **conectores CN1...CN10** : se han sustituido por interruptores y pulsadores, que en este caso representan los sensores al poder tomar dos estados exactamente igual que los hall:
 - **Abierto o no pulsado**: no circula corriente por la línea, lo que equivaldría a que el sensor estuviera inactivo.
 - **Cerrado o pulsado**: circula corriente por la línea, lo que equivaldría a que el sensor estuviera activo
- **cristal de cuarzo**: no hace falta colocarlo pues la herramienta ya lo hace por el usuario.
- **CN13**: este elemento y el Power Mossfet han sido sustituidos por un osciloscopio digital de forma que permita observar como varía la tensión gráficamente.
- **Virtual Terminal**: este elemento se ha añadido en la patilla 25 de forma que utilizando el procedimiento *mostrarCadena(char *)*, explicado durante la fase de diseño de software, se pueda comprobar si efectivamente el sistema envía la tensión adecuada al pulsar uno u otro botón, o al cerrar y abrir un interruptor.
- **Patillas 11, 12, 32 y 31 del microcontrolador**: no hace falta conectarlas a positivo y negativo respectivamente ya que la herramienta se encarga de alimentar el microcontrolador por defecto.

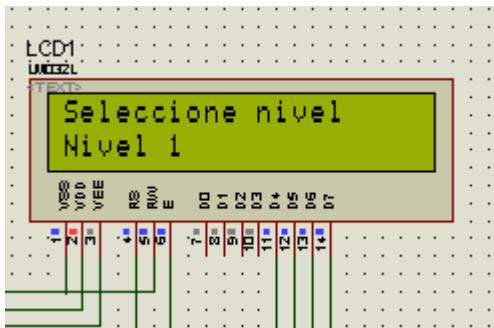
El esquema construido en este programa sería el siguiente:



5.2.1 Prueba para la selección de Nivel

Durante la prueba de selección de nivel se ha comprobado que el sistema es capaz de aumentar y disminuir el nivel empleando los botones *select1* y *select2*.

Por defecto el sistema comienza en el nivel 1.



Al pulsar el botón *select1* el nivel aumenta mientras que al pulsar el botón *select2* disminuye.



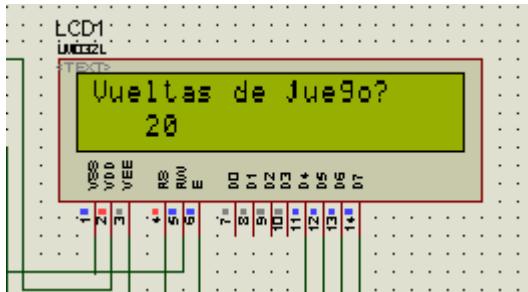
Se comprueba también que, efectivamente, en caso de que el nivel sea el tercero y se pulse el botón *select1*, el sistema retornará otra vez al nivel 1 y en caso de que el nivel sea el primero y se presione el botón *select2*, el sistema retornará al nivel 3.

Finalmente tras pulsar el botón *start* el sistema dará paso a la selección del número de vueltas.

5.2.3 Prueba para la selección del número de vueltas

Para la prueba de selección de número de vueltas se ha comprobado que el sistema es capaz de incrementar y disminuir el número de vueltas a gusto del jugador. Para ello se han utilizado los botones *select1*, capaz de aumentar este número, y *select2*, capaz de disminuirlo.

Por defecto el sistema comienza automáticamente con 20 vueltas.



En caso de pulsar *select1* las vueltas aumentarán, mientras que si se pulsa *select2* las vueltas de juego disminuirán, dentro del rango aceptable.



Se comprueba que efectivamente en caso de que las vueltas de juego tomen el valor 1 y se pulse el botón *select2* el sistema ignora la disminución pues es ilógico jugar carreras de cero vueltas.



Una vez se pulse *start* dará comienzo el juego.

5.2.3 Prueba para el Juego

Para llevar a cabo las pruebas del juego primero se ha de comprobar que el sistema es capaz de recalibrar los sensores.

Para ello se ha de tener en cuenta las diferentes situaciones que se pueden tener:

- **Sensor de la pista de jugador necesita ser calibrado:** para ello se prueba la simulación con el interruptor *Jugador* cerrado y el sistema pedirá que el jugador de una vuelta.

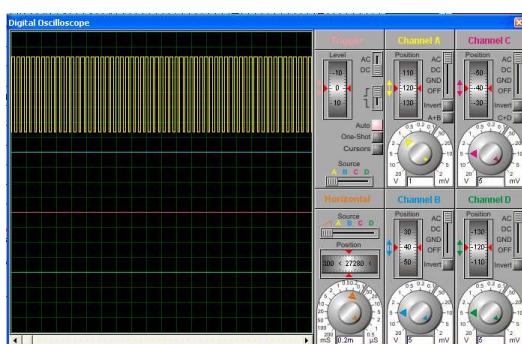


Finalmente, tras dar una vuelta, el sensor quedaría recalibrado, lo que en la simulación se interpretará como abrir el interruptor *Jugador*.

- **Algún sensor de la pista controlada por el sistema necesita ser calibrado:** en caso de que se dé esta situación el coche tendrá que dar una vuelta completa, por lo que el sistema informará de que se encuentra calibrando la pista.



Al mismo tiempo que el coche se mueve se puede observar como se eleva la línea en el osciloscopio, lo que implica que se ha aumentado la tensión.



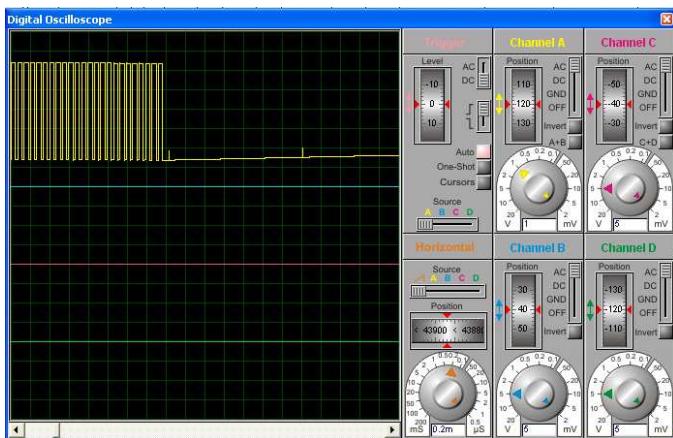
En caso de que el sistema tenga que calibrar los sensores se pueden dar dos casos:

- **sensor vuelta activo:** esto equivale en la simulación a cerrar el interruptor *vuelta* por lo que, una vez se abra, el sistema deberá indicar que se encuentra calibrado y significaría que el coche ya ha dado una vuelta y no se encuentra ningún sensor activo.
- **sensor vuelta inactivo:** esto equivale en la simulación a cerrar algún otro interruptor y dejar el interruptor *vuelta* abierto. Una vez abierto el interruptor causante de la calibración el coche todavía deberá completar una vuelta. Para indicarle esto en el programa ISIS se deberá cerrar y abrir el interruptor *vuelta*.

Efectivamente se comprueba que una vez completada la vuelta el sistema mostrará el mensaje “Sistema calibrado”.



Al mismo tiempo podemos observar como el coche se detiene observando el osciloscopio.

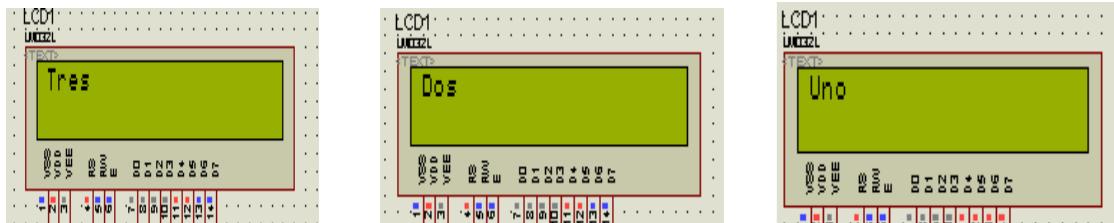


Una vez calibrados los sensores, el sistema pasará a pedir que el usuario introduzca el nivel con el que desea jugar.

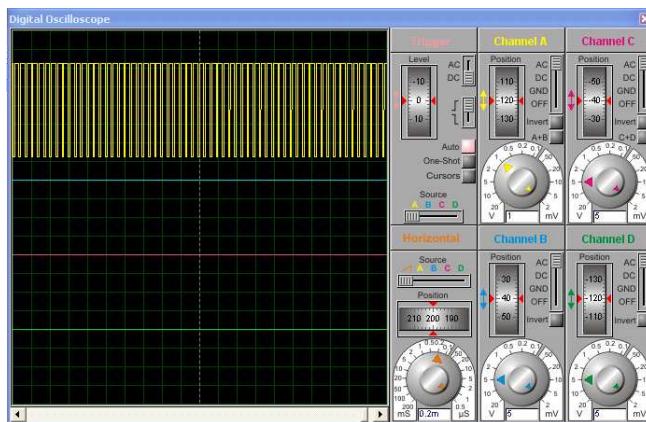
Cuando el usuario haya introducido tanto el nivel con el que desea jugar como el número de vueltas el sistema deberá informar que todo está correcto.



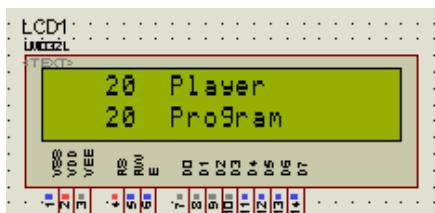
Se comprueba que una vez todo esté correcto, debe dar comienzo la cuenta atrás secuencial desde tres hasta uno.



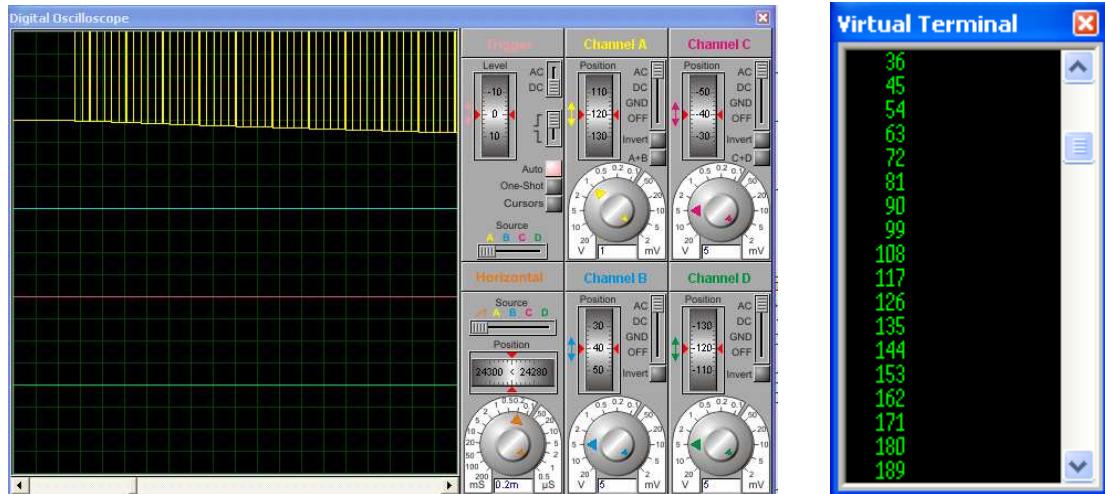
Finalmente, el sistema indicará con el mensaje *Adelante* que el jugador puede comenzar a jugar y el coche se pondrá en marcha.



Una vez se ponga el juego en marcha, en la pantalla LCD se debe ir mostrando el número de vueltas que le quedan por dar tanto al jugador como al sistema para ganar el juego. Inicialmente serán las mismas.



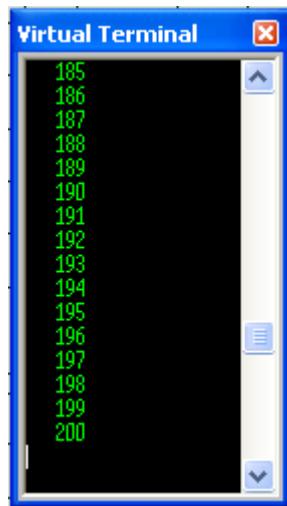
Para poder comprobar si el dutty (tensión) que se está emitiendo por la patilla del PWM es el correcto, se utilizará el Terminal virtual conjuntamente con el procedimiento *mostrarCadena*. Como se puede observar, este terminal muestra como va variando el valor del mismo conjuntamente con el osciloscopio.



Por lo tanto queda demostrado que el sistema es capaz de acelerar el vehículo.

Debe comprobarse que tras pulsar o cerrar y abrir los interruptores correspondientes los valores que muestra el *Virtual Terminal* varían en función de si se acelera o se decelera.

En caso de que tenga que acelerar el *Virtual Terminal* mostrará una sucesión ascendente de números enteros tal y como demuestra la siguiente imagen.



En el caso de que tenga que decelerar el *Virtual Terminal* mostrará una sucesión descendente de números enteros, tal y como demuestra la imagen inferior.

A screenshot of a 'Virtual Terminal' window. The title bar says 'Virtual Terminal'. The main area is a black terminal window with white text, displaying a list of numbers starting from 189 down to 178. There are scroll bars on the right and bottom of the window.

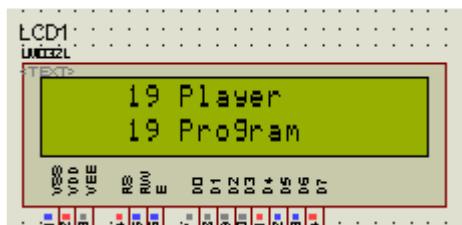
```

189
190
191
192
193
194
195
196
197
198
199
200
195
190
185
180
178

```

Queda demostrado por tanto que tras pulsar un botón o cerrar y abrir un interruptor, lo que en la pista equivaldría a que el coche activase un sensor, el sistema sería capaz de acelerar y decelerar el coche en función del tramo correspondiente.

Como el sistema debe ser capaz de llevar la cuenta de las vueltas restantes se cerrarán y abrirán los interruptores denominados *Jugador* y *Coche*, lo que indicará que se ha completado una vuelta y por lo tanto el sistema deberá reflejarlo.



Una vez llevadas a cabo estas acciones, se observa que efectivamente el sistema es capaz de disminuir las vueltas restantes tanto para el jugador como para el sistema.

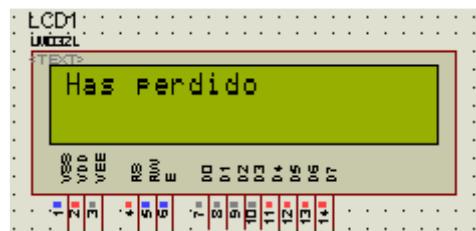
Una vez comprobado que el sistema es capaz de llevar a cabo el juego se comprobarán los tres posibles estados de salida.

- **Jugador gane el juego:** para ello se irá cerrando y abriendo el interruptor *Jugador*, hasta que al jugador ya no le queden más vueltas por dar. Debe tenerse en cuenta que se debe inutilizar la variable fallo dentro del bucle while comentando la línea de código `if (contador >= 200) fallo =1;` para no tener que estar pulsando algún otro botón (sino se interpretaría como una falta).



Como se puede observar, una vez el jugador complete su número de vueltas, el sistema le indica que ha ganado el juego.

- **Sistema gane el juego:** para ello se irá cerrando y abriendo el interruptor *vuelta*. El sistema es el mismo que el llevado a cabo para comprobar al Jugador, aunque en este caso no hará falta comentar la línea de código citada en el apartado anterior, pues contador se resetea al cerrar dicho interruptor.



Como se puede ver, una vez el sistema complete las vueltas correspondientes antes que el jugador éste mostrará el mensaje has perdido.

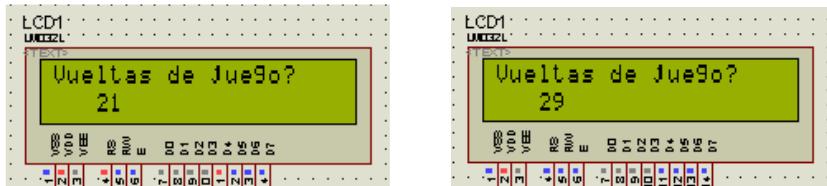
- **Jugador produzca falta:** teniendo en cuenta que el coche no puede salirse de la pista bajo ningún concepto, la única manera de que el sistema se quede sin respuesta es porque el coche del jugador habrá sacado de la pista al coche controlado por el sistema. Por lo tanto, si el sistema no recibe ninguna señal durante cinco segundos debe informar de que se ha producido una falta. Para comprobar esto basta con no pulsar ningún botón durante cinco segundos en el transcurso de la simulación una vez iniciada la carrera.



Una vez se produzca este hecho, el programa muestra el mensaje en la pantalla LCD.

Por último, se debe comprobar que el juego responde tanto a los distintos niveles de dificultad como al distinto número de vueltas.

Para garantizar que el juego es capaz de soportar distintos números de vueltas, bastará con asignar distintos valores seleccionado con el botón *select1* y *select2* y comprobar que efectivamente el número de veces que se cierran y abren los interruptores *Jugador* y *Vuelta* coincide con el número seleccionado.



Se ha determinado que el sistema es capaz de responder con el número de vueltas ajustado por el usuario.

Por último, hay que verificar que el dutty aplicado se modifica en función del nivel seleccionado. Para ello se deduce que pulsando el mismo botón en los tres niveles, el dutty varía en un 5% o un 10% debido a que la fórmula aplicada era:

$$X = X - \left(x \cdot \frac{N}{100} \right)$$

Donde X es el dutty y N el nivel que puede tomar los valores 0,1 y 2. Por lo tanto se puede deducir que X puede tomar los siguientes valores:

- N = 0. $X = X$
- N = 1. $X = 0.95X$
- N = 2. $X = 0.90X$

Para ello se utilizará el botón S4 que debe invocar la función con un dutty de 200.

En el nivel 1 el resultado obtenido a través del Virtual Terminal es el siguiente:

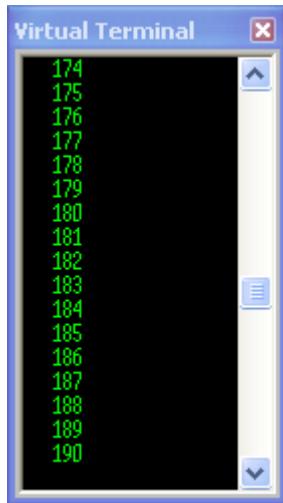
```

Virtual Terminal
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200

```

Como se puede observar el dutty es el esperado.

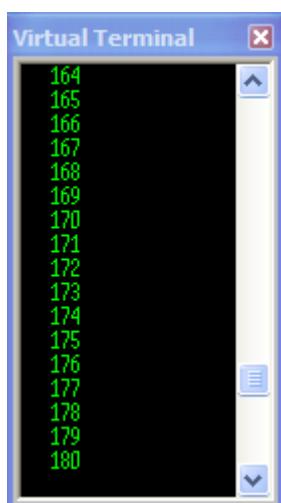
Con el nivel 2 y utilizando de nuevo el botón S4 se espera un dutty de 190 ya que $0.95 * 200 = 190$.



A screenshot of a Windows-style "Virtual Terminal" window. The title bar says "Virtual Terminal". The main area is a black text box containing green numerical data. The data starts at 174 and increments by 1 up to 190. There are scroll bars on the right and bottom of the window.

```
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190
```

Por último, empleando el nivel 3 el dutty esperado sería de 180 ($200 * 0,90 = 180$).



A screenshot of a Windows-style "Virtual Terminal" window. The title bar says "Virtual Terminal". The main area is a black text box containing green numerical data. The data starts at 164 and increments by 1 up to 180. There are scroll bars on the right and bottom of the window.

```
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180
```

Una vez testeado el sistema, se puede comenzar con la prueba final de software. Para ello es necesario grabar el programa en el microcontrolador y conectar la placa hardware al Scalextric.

Para poder escribir el programa en el microcontrolador se empleará el programa Pick Kit y un programador. Este programador es un periférico que se conecta al usb del ordenador y que automáticamente se encarga de generar los impulsos eléctricos necesarios para grabar el código en el microcontrolador.

El programa Pick Kit es una aplicación que permite tanto leer como escribir a través de este programador. Lo único que se debe hacer es seleccionar el fichero .hex generado por el compilador MikroC y cargarlo en el microcontrolador.

6. Construcción placa de circuitos final

Para la construcción de la placa final se han estudiado varias opciones.

La primera y más profesional consiste en utilizar una máquina llamada insoladora.

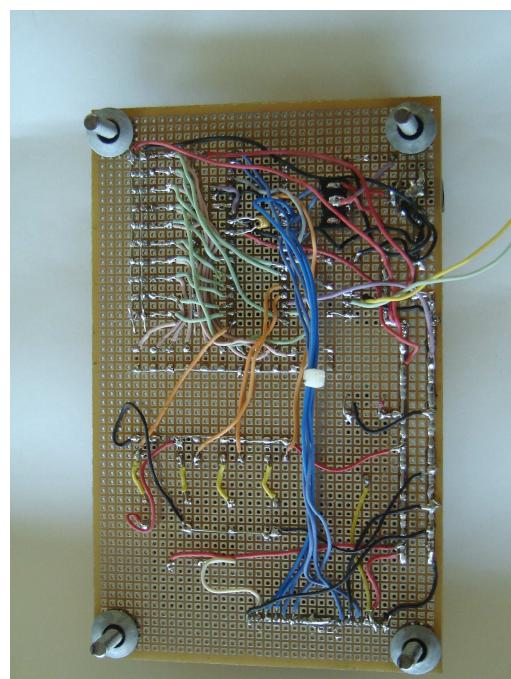
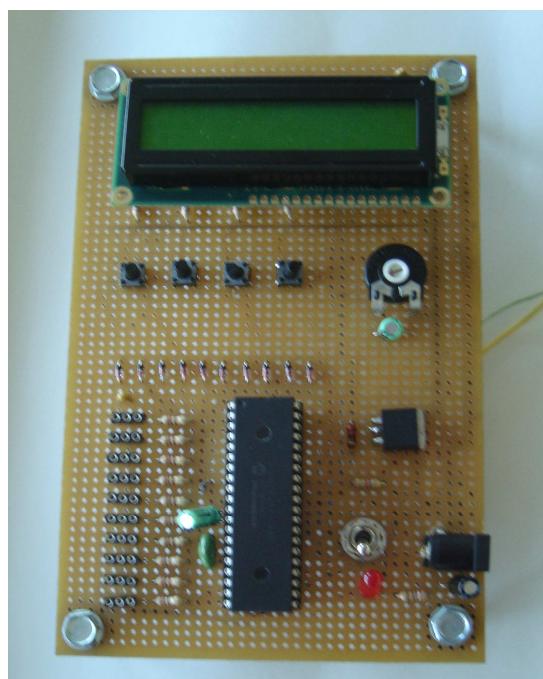
Esta máquina es capaz de plasmar el PCB en una placa de cobre. Una vez dibujado el circuito sobre la placa de cobre, se sumerge en una solución de ácido clorhídrico y agua oxigenada, de forma que el cobre es devorado a excepción de las partes que hayan sido protegidas. Una vez obtenida la placa con todas las pistas, debe ser taladrada para insertar y soldar todos los componentes necesarios.

Desgraciadamente no es posible realizarlo de esta forma, pues no se dispone de esta máquina.

Otra posibilidad que se ha barajado consiste en pintar el circuito con un rotulador especial. Sería el mismo procedimiento que con la insoladora pero el circuito en este caso se dibujaría a mano. El problema es que el PCB obtenido en la fase de diseño de hardware tiene una separación entre las pistas demasiado pequeña, por lo que realizar la placa utilizando este sistema sería muy complicado.

La otra forma de llevar a cabo la construcción de la placa sería utilizando una placa de desarrollo. Esta placa viene taladrada de forma que los componentes se puedan estañar en ella, lo que evita que haya que perforarla. Las pistas en este tipo de placas son sustituidas por cable.

Ésta última forma fue la escogida para llevar a cabo la construcción de la placa de circuitos final. El material de la misma es la baquelita, aunque también existen de fibra de carbono, la primera resulta más económica. Destacar que será necesario el uso de un estañador con una potencia máxima de 15 W, ya que tanto los diodos como el cristal de cuarzo son componentes sensibles que se pueden estropear por un excesivo calor.



7. Construcción de la maqueta

Para llevar a cabo la construcción de la maqueta se han necesitado los siguientes elementos:

- 2 tablones de madera de 1,5 x 1 metros
- 23 tacos de madera de 6 x 10 x 3,5 centímetros
- Silicona termo fusible
- Barra de pasta cerámica
- 46 tirafondos de 5 centímetros
- 23 tirafondos de 3 centímetros
- 2 tiras de pines
- Cable de bus
- Estaño

Inicialmente se han construido los cables necesarios para transportar tanto la alimentación desde la placa hasta los sensores, como la señal desde los sensores a la misma. Para ello se ha empleado el cable de bus, conjuntamente con las tiras de pines siguiendo el esquema diseñado durante la fase de diseño de hardware. Una vez estañados los cables en los extremos, se ha utilizado la barra de pasta cerámica para endurecerlos y evitar que se deteriore la soldadura.

Una vez realizado los cables, se le han añadido los sensores que se habían extraído y se han colocado en la pista.

Se ha decidido utilizar la silicona termo fusible, de forma que fuera posible ir probando si la colocación escogida del sensor era la correcta. Para ayudar en la localización de la zona correcta se implementó en la protoboard un esquema con un LED, de forma que cuando el coche pasara por encima del sensor éste se iluminara.

Una vez colocados todos los sensores, solamente restaba anclar el Scalextric en los tablones. Como el Scalextric tiene unas medidas que dificultan su transporte se decidió que la base sobre la que estaría anclado pudiera dividirse en dos. Se procuró por tanto, que las rectas del mismo coincidiesen con la separación de los tablones de madera, de forma que se pudiera desmontar en dos partes sin ningún problema.

Una vez colocado el Scalextric, se marcaron cuales serían las posiciones correctas para colocar los tacos de madera, de forma que el Scalextric se pudiera elevar y los sensores no sufrieran desperfecto alguno.

Para anclar los tacos de madera a la base se utilizaron tirafondos de 5 centímetros de largo, dos por cada taco, evitando así que estos pudieran rotar y descolocarse.

Una vez colocados todos los tacos, se atornilló el Scalextric a los mismos de forma que quedara totalmente anclado y se pudiera trasladar sin ningún problema. En esta tarea se emplearon tirafondos de 3 centímetros.

Finalmente se utilizó nuevamente la silicona termo fusible para pegar los cables del bus a la base.

Una vez obtenido la maqueta final, se ha podido pasar a la prueba real final y completar por fin los tiempos de vuelta en el código.

8. Prueba final

Tras ejecutar el juego y probar los distintos niveles, los tiempos obtenidos son los siguientes:

- Nivel 1: 4 segundos
- Nivel 2: 6 segundos
- Nivel 3: 8 segundos

Teniendo en cuenta estos tiempos, se puede deducir que las interrupciones del TMR0 que se producirán en una vuelta son 80, 120 y 160 respectivamente. Por lo tanto, el código en el procedimiento *interrupt* que se encarga de procesar la señal emitida por el sensor que cuenta las vueltas puede ser completado de la siguiente forma:

```
if (PORTA.F1) //sensor vueltas coche
{
    numeroVueltasCoche--;
    contador=0; //reseteamos contador porque paso por un sensor
    switch (nivellInterrupt)
    {
        case 0: //tarda ???
            tiempovuelta = interrupcionesvuelta/80;
            break;
        case 1://tarda ???
            tiempovuelta = interrupcionesvuelta/120;
            break;

        case 2: //tarda ???
            tiempovuelta = interrupcionesvuelta/160;
            break;
    }
    interrupcionesvuelta=0;
    detInterrupcion=10;
    delay_ms(200);
}
```

Una vez completado el código se ha testeado que el sistema funciona. Las pruebas llevadas a cabo son las siguientes:

- **Prueba de calibración de los sensores:** Se han activado los sensores pasando un imán en sentido contrario al sentido del coche. A continuación se ha encendido la placa y se ha comprobado si el sistema era capaz de recalibrarlos. Tal y como se demuestra en el video denominado *recalibrar el sistema* responde perfectamente.

- **Prueba de selección de nivel:** Se ha testeado que el sistema permite la selección de los tres niveles exigidos en el análisis de requisitos y que efectivamente el tiempo que tarda el coche en dar una vuelta varía en función de este nivel. La selección de niveles puede verse en el video denominado *seleccionnivel*.
- **Prueba de selección de número de vueltas:** Se ha comprobado que efectivamente el sistema permite seleccionar el número de vueltas y que este oscile entre 1 y 255. La selección de número de vueltas puede verse en el vídeo denominado *seleccionvueltas*.
- **Prueba de juego:** Se ha comprobado que el sistema es capaz de controlar el coche por sí solo. Para ello puede observarse el vídeo denominado *coche_solo*. También se han testeado los tres posibles estados de salida, es decir, que el jugador gane, pierda o se cometa una falta. Para comprobar los dos primeros se han ejecutado carreras de una sola vuelta, el resultado puede ser visionado en los videos *ganas* y *pierdes* respectivamente. Para comprobar una posible situación de falta se han quitado los coches durante el transcurso del juego y se ha comprobado que efectivamente tras cinco segundos sin respuesta de ningún sensor el sistema informa de la falta. Este último estado puede verse en el video denominado *falta*.
- **Prueba real:** Finalmente se ha contado con la ayuda de una persona ajena al proyecto, para que el sistema fuera comprobado por parte de un posible usuario final, con resultado satisfactorio.

9. Desajustes y errores cometidos

Como en la mayoría de los proyectos en este se han cometido errores, que serán recogidos en este apartado de forma que puedan resultar útiles si se decide realizar algún otro proyecto de características similares.

Estos problemas han ocasionado que el proyecto durase más de lo previsto tal y como demuestra el siguiente diagrama de Gantt.



El primer desfase del proyecto ocurrió durante la fase de diseño de hardware. En un principio se había previsto que esta fase durara entorno a 4 días. Sin embargo finalmente fueron 10, lo que ocasionó un retraso en el resto del desarrollo del proyecto, pues tal y como se muestra en el diagrama formaba parte del camino crítico. Este problema surge debido a la inexperiencia del diseño de hardware por parte del diseñador, lo que propició que los diseños tuvieran que ser corregidos tres veces hasta dar con un diseño que fuera funcional.

El segundo desfase se produjo durante la fase de diseño de software. Inicialmente se planteó un diseño en el cual el duty del PWM fuese variado en el procedimiento interrupt, de forma que el sistema respondiera inmediatamente. El problema surgió porque la instrucción encargada de llevar a cabo esta acción solamente podía ser llamada desde el flujo principal del programa, ya que sino la pila del microcontrolador se agotaba, lo que obligó a tener que rediseñar la forma en la que el programa se encargaría de atender estas interrupciones. Esto supuso un retraso de 3 días en todo el ciclo de desarrollo del sistema. No obstante, este exceso se pudo compensar ya que la

implementación de la función de reset no se tuvo que diseñar mediante software, sino que simplemente bastó con colocar un botón que funcionase con lógica negada en la patilla uno del microcontrolador.

Las pruebas realizadas con la placa de prototipos no resultaron del todo satisfactorias, ya que no se consiguió que esta funcionara al 100 %. No obstante sirvió para comprobar que la placa diseñada era correcta dando paso a la construcción de la placa final.

Las pruebas de software se llevaron a cabo con menor tiempo del que estaba previsto, pues al no tener que probar el reset se ahorraron tres días.

El mayor y más grande de todos los errores fue subestimar el tiempo que se tardaría en montar la placa de circuitos final. Debido a la poca experiencia en el campo de la soldadura de componentes electrónicos se construyó parte de la placa con numerosos errores. El más grave de ellos fue el de ocasionar cortocircuitos, ya que algunas soldaduras se solapaban con otras. Finalmente se decidió descartar el trabajo realizado y construir una segunda placa de nuevo. Gracias a la experiencia adquirida con la primera, esta se pudo llevar a cabo más rápidamente. Aunque la placa finalmente fue realizada, el haber subestimado la dificultad de la construcción de circuitos supuso un retraso de 12 días.

Finalmente, para construir la maqueta se había barajado la posibilidad de fijar el Scalextric a los tacos de madera utilizando silicona negra. Pero tras su utilización se comprobó que el Scalextric no se había fijado bien en todos los tacos de madera, por lo que se tuvo que proceder a retirarla y fijarla con los tornillos. Este traspie supuso un retraso de 3 días.

10. Conclusión

Este proyecto ha sido muy enriquecedor desde el punto de vista académico, ya que ha permitido extender los conocimientos de la informática más allá del software.

Desde un punto de vista empresarial, se podría afirmar que para construir sistemas informáticos en los que el hardware es tan importante, sería interesante contar con alguien experto en la materia, pues no se habrían cometido tantos errores ni en el diseño ni en la implementación de hardware.

Desde un punto de vista particular, puedo decir que gracias a este proyecto he aprendido como construir sistemas capaces de responder a estímulos externos utilizando sensores, ampliando así los conocimientos que ya tenía de los sistemas de tiempo real y que hoy día son bastante útiles en el campo de la domótica y la industria en general.

Durante la fase de diseño e implementación software hubiera sido interesante contar con una herramienta que permitiese la depuración en placa. Es decir, una herramienta capaz de controlar el estado del microcontrolador e ir ejecutando el programa paso a paso, lo que sin duda habría ayudado a la hora de corregir errores. De hecho, el propio compilador de Microchip MikroC ofrece esta posibilidad. No obstante, debido al coste económico de la placa, no fue posible la obtención de la misma.

Atendiendo a las funcionalidades que ofrece el sistema, se puede concluir que es un éxito, pues cumple con todas las exigencias previstas.

Una vez el sistema está completo, se puede afirmar que no se hubiera podido emplear un microcontrolador de la familia 16C7x, ya que el programa ocupa 12 K. Sin embargo, si se quiere reducir costes, se podría utilizar el PIC 18F442 cuya memoria es de 16K y por lo tanto el programa cabría perfectamente.

Finalmente, recordar que a pesar del éxito del proyecto, en futuros diseños se debe consultar con un experto cuando se desarrolle una parte del mismo del que uno no tiene experiencia, pues a la larga resultará más cómodo y rentable.

11. Futuras ampliaciones

11.1 Sistema imbatible

Aunque el sistema en el nivel uno posea una dificultad elevada, es posible ganarle ya que está basado en los tiempos de un ser humano. Una posible ampliación inmediata sería implementar algún algoritmo de inteligencia artificial de forma que el sistema se pudiera ir entrenando para aprender la pista y conseguir, por tanto, que ningún jugador humano pudiera ganarle.

11.2 Jugador fantasma

En numerosos videojuegos, sobre todo de carreras, existe una modalidad en la que el jugador trata de batir sus propios récords. Durante el transcurso de la carrera el jugador puede observar otro objeto, normalmente una silueta, que realiza la carrera con su mejor tiempo obtenido.

En este sistema se podría representar al jugador fantasma utilizando el carril controlado por el sistema. Para ello habría que equipar con sensores ambas pistas de forma que fuera posible la memorización del comportamiento del jugador por parte del sistema. Una vez obtenido el mejor tiempo por parte del jugador, el comportamiento llevado a cabo durante esa vuelta podría ser reproducido por el coche controlado por el sistema en el otro carril.

11.3 Testeo del estado de los sensores

Aunque es difícil que ocurra, los sensores empleados pueden estropearse. Una posible ampliación sería que si tras intentar calibrar los sensores, no fuese posible calibrarlos, el sistema podría detectar qué sensor está defectuoso. Para ello bastará con desactivar las interrupciones procedentes de la patilla RB0 y analizar todas las entradas que está recibiendo el microcontrolador. Una vez detectado el sensor defectuoso el sistema mostraría en el LCD cuál debe ser sustituido.

11.4 Recuperación de la escalabilidad

Una de las características que poseen los Scalextric, es su increíble variedad de piezas para la construcción de pistas. El problema del sistema construido es que solamente puede ser utilizado en una pista igual para la que fue diseñado. Para poder recuperar esta funcionalidad se ha pensado en una ampliación de este sistema atendiendo tanto al hardware como al software.

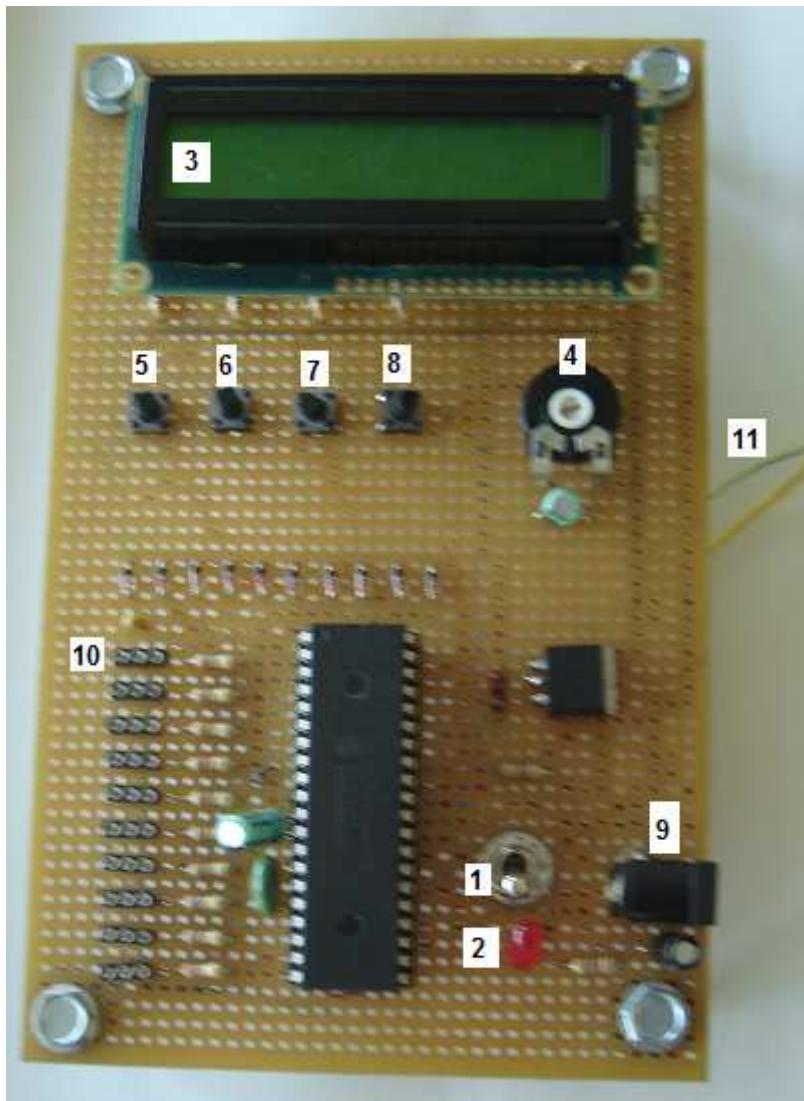
Para poder llevar a cabo la escalabilidad, el sistema debe cambiar la forma de controlar la posición del coche. Existe la posibilidad de utilizar sensores de movimiento de forma que colocando tres en una determinada posición el sistema sea capaz de triangular la señal y por lo tanto conocer la posición del coche.

Desde el punto de vista del software habría que utilizar algún algoritmo de inteligencia artificial de modo que el sistema se pueda ir entrenando por sí mismo adaptándose así a cualquier forma que tenga la pista.

12. Manual de usuario

12.1 Descripción de la interfaz

La placa de hardware consta de los siguientes componentes:



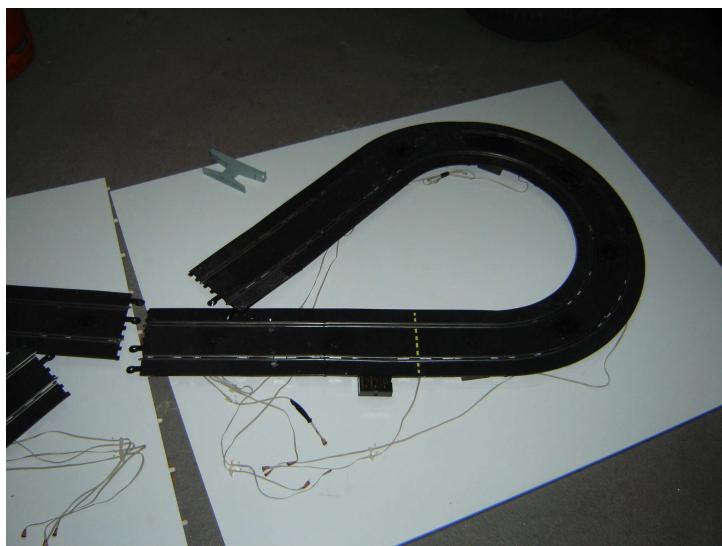
- 1: interruptor on/off
- 2: LED, en caso de estar encendido se ilumina
- 3: pantalla LCD.
- 4: regulador de contraste de la pantalla LCD.
- 5: botón de reset.
- 6: botón de start.
- 7: botón select1.
- 8: botón select2.
- 9: toma de alimentación del transformador.

- 10: conectores de los sensores. Desde la parte superior hasta la inferior serían en orden ascendente del 1 al 10.
- 11: conectores para la pista del Scalextric.

12.2 Montaje

Para montar el Scalextric lo primero que se debe hacer es ensamblar ambas partes procurando que los tubillones de la tabla encajen con los agujeros de la otra

Es mejor colocar ambas partes en el suelo tal y como muestra la fotografía.

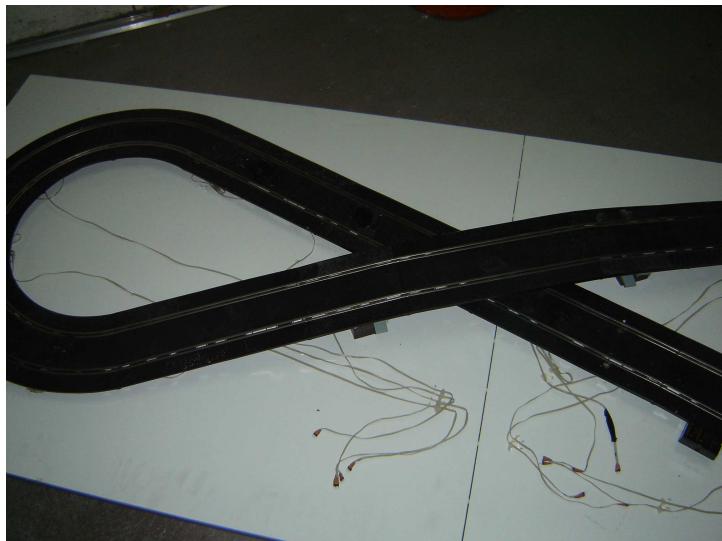


Una vez colocadas en el suelo se deben alinear, teniendo en cuenta que la pista debe quedar encajada.

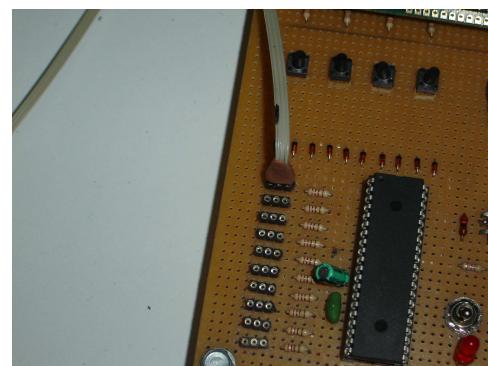
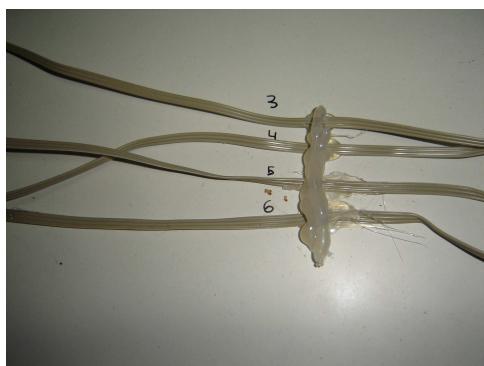


Una vez encajada se deben asegurar los enganches que se encuentran por debajo de las pistas y armar el puente.

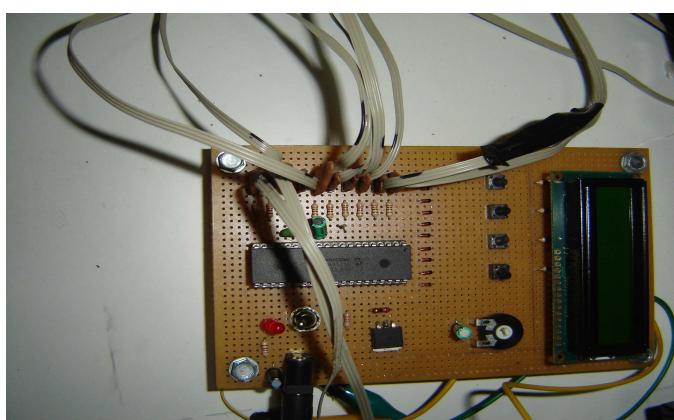
Para ello se colocará la pieza correspondiente y se ajustarán las columnas para elevar la pista.



Una vez este todo colocado, se procederá a conectar la placa. Para ello basta con seguir las numeraciones que tienen los cables y conectarlos por orden en la placa. El conector de la parte superior se denomina como 1, mientras que el de la parte inferior sería el 10.

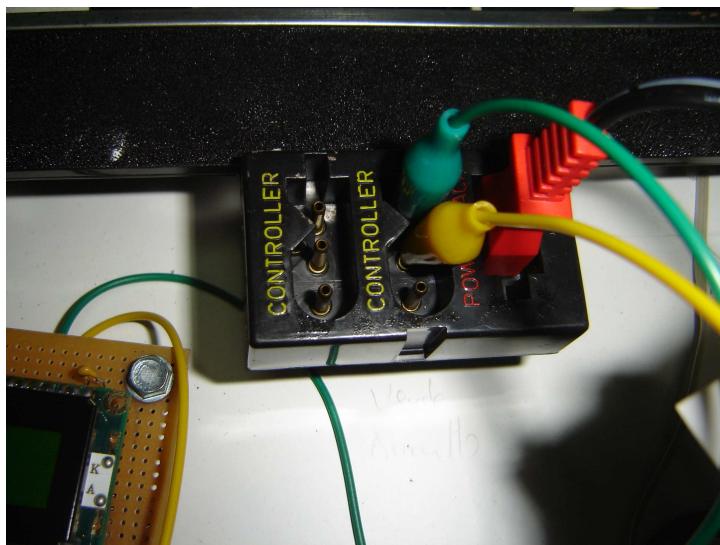


Una vez conectados todos los cables, la placa tendrá el siguiente aspecto.

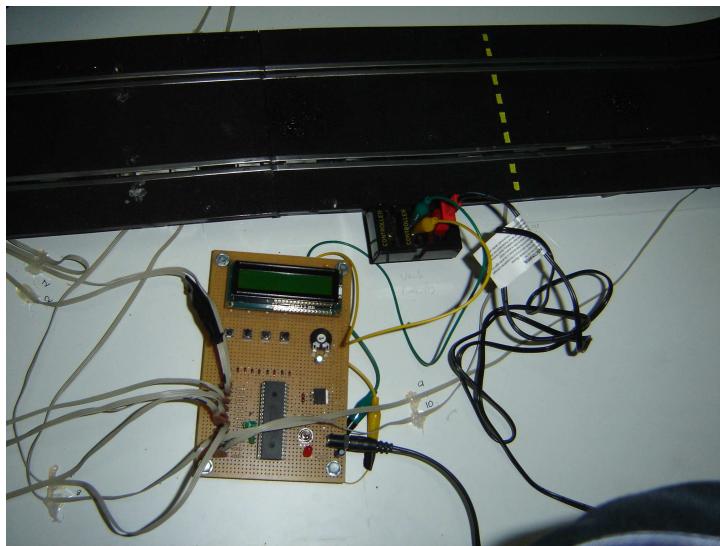


A continuación se conectarán los cables encargados de regular la tensión en la pista. Estos son los cables con pinzas en cada extremo. El verde se conecta en la parte superior y el amarillo en la conexión inmediatamente inferior, tal y como muestra la fotografía.

El cable rojo es el de la alimentación del Scalextric, debe colocarse con la marca hacia arriba, ya que sino los coches circularían en sentido contrario.



Por último se debe conectar la alimentación a la placa en la parte inferior derecha.



El mando para el jugador se conectará en el espacio que queda al lado de los cables verde y amarillo.

Después se deben situar los coches justo debajo del puente y encender la placa utilizando el interruptor on/off.



12.3 Uso

Este sistema permite la posibilidad de jugar carreras de forma que el usuario pueda seleccionar tanto el nivel como el número de vueltas que desea.

Una vez montado es posible que sea necesario calibrar los sensores, en cuyo caso es posible que tenga que dar una vuelta a la pista con el coche.

Calibrado de los sensores

Como ya se ha mencionado anteriormente es posible que tras encender la placa el sistema necesite ser calibrado.

Si se tiene que calibrar el sensor situado en la pista del jugador, el LCD mostrará el mensaje "Por favor de una vuelta". Usted debe de dar una vuelta con su coche. Una vez completada la vuelta, el sistema comenzará a calibrar los sensores restantes. Para ello el coche controlado por el sistema dará una vuelta completa. Cuando esto ocurra en la pantalla LCD se podrá leer el mensaje "Calibrando sensores". Una vez el sistema termine de calibrar los sensores mostrará el mensaje "Sistema calibrado" y dará paso a la selección de nivel.

Selección de nivel

Durante la fase de selección de nivel el sistema mostrará el mensaje en el LCD "Seleccione Nivel". Por defecto el sistema comienza en el Nivel 1, es decir el nivel máximo de dificultad.

Usted puede elegir entre tres niveles de mayor a menor dificultad. Para ello se utilizarán los botones select1 y select2. El botón select1 permite ir incrementando el nivel hasta el nivel 3. En caso de encontrarse en el nivel 3 y pulsar este botón, el sistema pasará al nivel 1. El botón select2 permite ir disminuyendo el nivel hasta el nivel 1. En caso de encontrarse en el nivel 1 y pulsar este botón, el sistema pasará al nivel 3.

En cualquier caso en la parte inferior del LCD, usted podrá comprobar en que nivel se encuentra gracias al mensaje "Nivel x", siendo x los niveles 1,2 y 3.

Una vez se obtenga el en LCD el nivel deseado solamente se debe pulsar el botón start.

Selección del número de vueltas.

Después de seleccionar el nivel, el sistema le permitirá escoger el número de vueltas deseadas. EL LCD mostrará el mensaje "Vueltas de juego" en la parte superior. En la parte inferior se mostrarán el número de vueltas que usted desea. Por defecto se comienza con 20 vueltas.

El botón select1 permite incrementar el número de vueltas hasta un máximo de 255.

El botón select2 permite decrementar el número de vueltas hasta un mínimo de 1.

En cualquier caso en la parte inferior del LCD se irá mostrando el número de vueltas. Cuando alcance el valor deseado, bastará con pulsar el botón start.

Juego

Una vez seleccionado tanto el nivel, como el número de vueltas el sistema informará al usuario mediante el mensaje “Todo correcto, comienza el juego”, aclarando que el sistema ya se encuentra disponible para jugar.

A continuación dará comienzo una cuenta atrás desde tres hasta cero. Usted, debe comenzar a correr cuando en el LCD aparezca el mensaje “Adelante”.

Una vez comenzada la carrera el LCD irá indicando el número de vueltas restantes tanto para usted como para el sistema. Para ello mostrará el mensaje “Program x”, siendo x su número de vueltas y “Player y” siendo y su número de vueltas.

Una vez terminada la carrera se detendrá el coche controlado por el sistema y mostrará el mensaje “Has ganado” en caso de que gane o “Has perdido” en caso de que pierda.

Si usted colisiona con el coche controlado por el sistema y lo saca de su carril será considerado falta, en cuyo caso el sistema detendrá el juego e informará de la falta mediante el mensaje “Has cometido falta”.

Resetear el sistema

En caso de equivocarse durante la selección de nivel o del número de vueltas, usted puede pulsar el botón de reset para retornar a la selección de nivel.

No debe pulsar el botón durante el transcurso de la carrera ya que el coche controlado por el sistema saldría disparado y podría resultar dañado.

12.4 Advertencias

● Limpieza

Este sistema requiere que las pistas de los coches estén limpias, por lo tanto se recomienda que antes de iniciar cada juego se limpien utilizando alcohol, ya que las escobillas de los coches producen residuos que quedan adheridos a las mismas. Si usted nota que después de varias carreras, el coche controlado por el sistema circula más lento de lo debido, por favor limpie la pista y espere unos segundos para que se evapore el alcohol.

● Sensor estropeado en la pista del jugador.

Si durante la fase de calibración, usted completa la vuelta y no aparece el mensaje “Calibrando el sistema” o “Todo correcto”, es muy posible que el sensor colocado en su pista esté estropeado y haya que sustituirlo.

● Sensor estropeado en la pista controlada por el sistema.

Si en LCD aparece el mensaje “Calibrando el sistema” y el coche controlado por el sistema completa más de dos vueltas, es probable que algún sensor falle y por lo tanto haya que sustituirlo.

13. Glosario

- **Baudio:** unidad de medida usada en el campo de las telecomunicaciones que representa el número de símbolos transmitidos por segundo.
- **Biestable:** también llamado báscula (*flip-flop* en inglés), es un multivibrador capaz de permanecer en un estado determinado o en el contrario durante un tiempo indefinido. Esta característica es ampliamente utilizada en electrónica digital para memorizar información. El paso de un estado a otro se realiza variando sus entradas.
- **Botón:** o pulsador, es un dispositivo utilizado para activar alguna función. Son de diversa forma y tamaño y se encuentran en todo tipo de dispositivos, aunque principalmente en aparatos eléctricos o electrónicos.
- **Condensador electrolítico:** condensador que utiliza un líquido iónico conductor como una de sus placas. Suele utilizarse en los filtros de alimentadores de corriente para almacenar la carga y modelar tanto el voltaje de salida como las fluctuaciones de corriente en la salida rectificada.
- **Entrada/salida:** abreviado E/S o I/O (del inglés input/output), es una colección de interfaces que usan los distintos subsistemas de un sistema de procesamiento de información para comunicarse con otras unidades a través del uso de señales.
- **Flag:** uno o más bits utilizados para almacenar un valor binario o código que tiene asignado un significado.
- **Fotograma:** cada una de las imágenes capturadas por una cámara de alta resolución y velocidad impresas en papel.
- **Interrupción:** también conocida como interrupción hardware o petición de interrupción, es una señal recibida por el procesador de un ordenador, indicando que debe "interrumpir" el curso de ejecución actual y pasar a ejecutar código específico para tratar esta situación.
- **LCD:** pantalla delgada y plana formada por un número de píxeles en color o monocromos colocados delante de una fuente de luz o reflectora. A menudo se utiliza en dispositivos electrónicos a pilas, ya que utiliza cantidades muy pequeñas de energía eléctrica.

- **LED:** acrónimo del inglés *Light-Emitting Diode*, es un dispositivo semiconductor que emite luz incoherente de espectro reducido cuando se polariza de forma directa la unión PN del mismo y circula por él una corriente eléctrica,
- **Microcontrolador:** circuito integrado o chip que incluye en su interior las tres unidades funcionales de un ordenador: CPU, Memoria y Unidad de E/S. Disponen generalmente de una gran variedad de dispositivos de entrada/salida, como convertidores analógico a digital, temporizadores, UARTs y buses de interfaz serie especializados como I2C y CAN. Frecuentemente, estos dispositivos integrados pueden ser controlados por instrucciones de procesadores especializados.
- **PCB:** del inglés *Printed Circuit Board*, es un medio para sostener mecánicamente y conectar eléctricamente componentes electrónicos, a través de *rutas* o *pistas* de material conductor, grabados desde hojas de cobre laminadas sobre un sustrato no conductor.
- **PIC:** familia de microcontroladores de tipo RISC fabricados por Microchip Technology Inc. Su mayor utilidad es la de funcionar como controlador de interfaces de periféricos.
- **Polímetro:** también denominado multímetro o tester, es un instrumento de medida que ofrece la posibilidad de medir distintos parámetros eléctricos y magnitudes en el mismo aparato. Las más comunes son las de voltímetro, amperímetro y óhmetro.
- **Power Mosfet:** transistor basado en el campo eléctrico para controlar la conductividad de un canal en un material semiconductor. Pueden plantearse como resistencias controladas por voltaje.
- **Programador:** dispositivo electrónico empleado para escribir en la memoria de un microcontrolador el conjunto de instrucciones que debe ejecutar para realizar una tarea específica.
- **Proteus:** compilación de programas de diseño y simulación electrónica, desarrollado por Labcenter Electronics que consta de los dos programas principales, Ares e Isis, y los módulos VSM y Electra.
- **Protoboard:** también conocida como placa de pruebas, es una placa de uso genérico reutilizable o semi permanente, usada para construir prototipos de circuitos electrónicos con o sin soldadura. Normalmente se utilizan para la realización de pruebas experimentales.
- **Puerto:** interfaz física por la cual diferentes tipos de datos pueden ser enviados y recibidos.

- **PWM:** modulación por ancho de pulso, también conocido como control de ancho de pulso, de una señal o fuente de energía. Es una técnica en la que se modifica el ciclo de trabajo de una señal periódica, ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.
- **Registro:** dispositivo lógico secuencial capaz de almacenar varios bits de información.
- **Transformador:** máquina eléctrica que permite aumentar o disminuir la tensión en un circuito eléctrico de corriente alterna manteniendo la frecuencia.
- **Tubillón:** conocido también con el nombre de tarugo es un pequeño cilindro de madera dura utilizado para unir dos piezas de madera.

13. Bibliografía

- 📖 Barnett Richard. Embedded C programming and the microchip PIC. Nueva York. Thomson/Delmar Learning. 2004
- 📖 Jose Mª Angulo Usategui. Microcontroladores PIC: diseño práctico de aplicaciones. Madrid. McGraw-Hill. 2006

- 📄 VV.AA. Data Sheet PIC18FXX2. Microchip Technology Inc. 2006.
- 📄 VV.AA. Complementary Output Hall Effect Latch. Anachip Corp. 2005.
- 📄 VV.AA. NTD60N03, Power MOSFET 60 Amps, 28 Volts (N- Chanel)

 <http://www.mikroe.com>
 <http://www.forosdeelectronica.com>

14. Contenido del CD

● Archivos Proteus:

- circuito.DSN: archivo utilizado en el programa ISIS para llevar a cabo la simulación

● Archivos Livewire:

- esquema scalextric.pcb: archivo que contiene el esquema lógico electrónico de la placa hardware.
- pcb scalextric.pcd: archivo que contienen los planos del PCB para la construcción de la placa hardware.

● Código fuente:

- display.c: código fuente del programa.
- display.hex: código en hexadecimal para insertar en el PIC.
- display.asm: código fuente en ensamblador generado por MikroC

● Manuales:

- 18f452: manual del microcontrolador empleado en el proyecto.
- 276: manual de los sensores hall empleados.
- NTD60N03-D: manual del Power MOSFET

● Pdf:

- circuito ISIS: esquema empleado en ISIS Proteus.
- circuito logico: esquema del circuito electrónico de la placa.

● Videos:

- coche_solo: video que demuestra que el vehículo esta controlado por el sistema.
- falta: video que demuestra la salida falta.
- ganas: video que demuestra que el jugador puede ganar.
- pierdes: video que demuestra que el sistema puede ganar.
- calibrar: video que muestra el calibrado de los sensores.
- seleccionnivel: video que muestra la selección del nivel.
- seleccionvueltas: video que muestra la selección del nº de vueltas.