2. Desarrollo del Robot

Este es el capítulo más importante, ya que está dedicado al desarrollo de todo el hardware y software de control así como de las partes mecánicas del robot. En una primera aproximación se muestra de forma general la arquitectura del robot, que en los capítulos posteriores se mostrará con detalle.

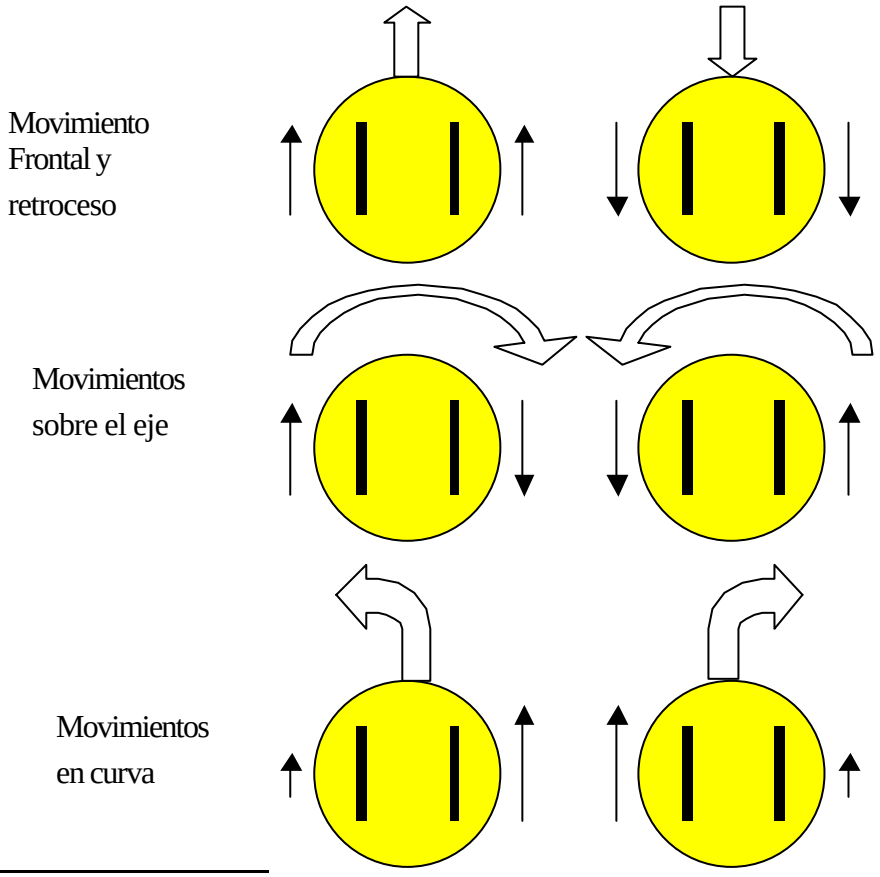
2.1. Arquitectura general del Robot

Atendiendo a los tres primeros niveles de la torrebot expuesta en el capítulo anterior se introducen los aspectos generales del robot APOFIX[1].

2.1.1. Nivel Físico

Este nivel corresponde a la estructura mecánica del robot, que es una parte fundamental en su desarrollo, ya que una elección inadecuada de la misma puede dar al traste con todo el robot, ó complicar los algoritmos de forma innecesaria.

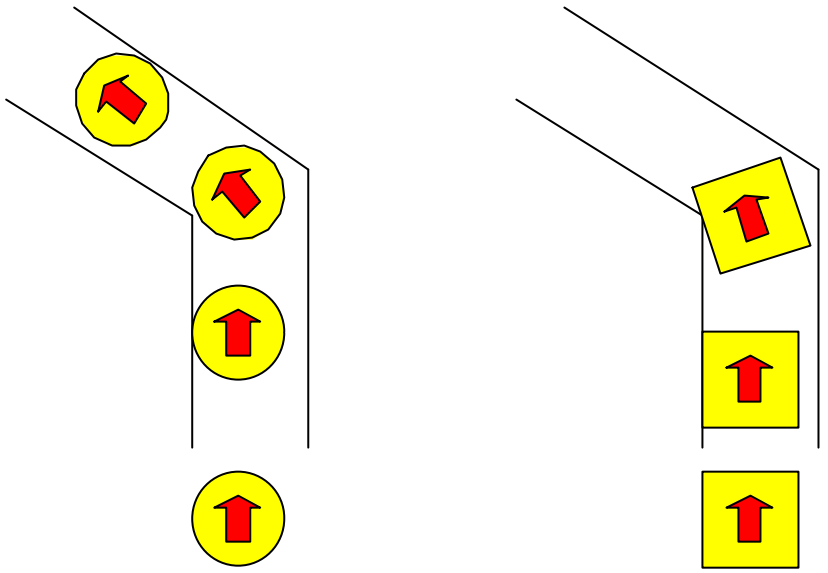
En este caso se ha elegido una estructura de tracción diferencial, sus ventajas son la facilidad de implementación y los algoritmos más sencillos para no quedar atascado, pues esta permite el movimiento giratorio sobre el propio robot tal y como muestra la siguiente figura:



[1] Este robot no debe su nombre a ningún tipo de acrónimo, es solo un nombre.

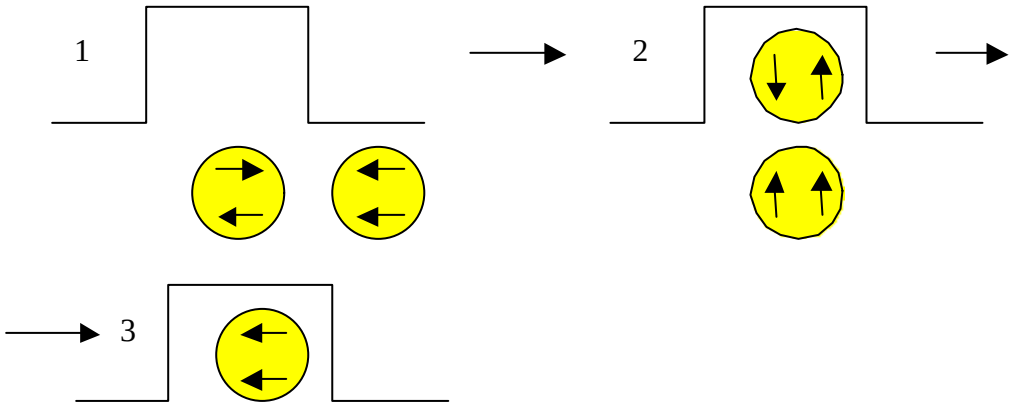
En la figura se aprecia los distintos movimientos que se pueden realizar aplicando diferentes velocidades y direcciones a cada rueda.

Por otro lado se ha optado por una superficie circular como plataforma, al facilitar el movimiento del robot así como los algoritmos en entornos no estructurados. Se muestra como ejemplo la sencillez con que un robot de forma circular y tracción diferencial puede atravesar un pasillo muy ajustado a su tamaño frente a un robot de planta cuadrada:



El robot circular puede sortear la curva sin problemas, siendo difícil que quede atrapado si su algoritmo está bien construido. Sin embargo el de planta cuadrada queda atrapado en la curva, pudiendo salir únicamente con un algoritmo de cierta complejidad frente al anterior, ya que requerirá de maniobras de marcha atrás para poder sortearlo.

Se muestra otro ejemplo en el que la tracción diferencial es más robusta frente a la tracción + dirección (como los coches), se trata de introducirse en un hueco para quedar mirando en una determinada dirección:



En este caso el robot gira sobre sí mismo en el primer paso, de este modo queda enfrentado con su objetivo. En el segundo paso avanza hasta quedar dentro del hueco, y en el tercero ha girado sobre sí mismo para alcanzar la posición deseada. Sin embargo en el caso de un robot con tracción y dirección, al igual que un coche necesita realizar distintas maniobras marcha atrás y marcha adelante que no resultan nada triviales, y si no piense en como aparca su coche.

La implementación final, con materiales, tipo de motores...etc, será explicada con detalle en el apartado 2.5.

2.1.2. Nivel de Reacción

Este es el nivel más bajo en el que se puede encontrar un robot. Consiste únicamente en unir los sensores con los actuadores (en nuestro caso los motores) a través de una circuitería simple, sin ningún tipo de planificación de las acciones, el robot reaccionará automáticamente ante ciertos estímulos. Este robot será implementado con una FPGA como núcleo del sistema, esto permitirá alcanzar este nivel rápidamente, puesto que se puede programar la electrónica que unirá los sensores con los actuadores.

Un ejemplo de este tipo de nivel será alcanzado uniendo sensores de suelo que distinguen entre negro y blanco directamente con los motores, es decir, si el sensor izquierdo está sobre negro acciona el motor derecho, pero si está sobre blanco lo detiene, el sensor derecho actuaría de igual modo sobre el motor izquierdo. De este modo se consigue que el robot siga la línea negra sobre fondo blanco, pero no se ha dotado al robot de inteligencia alguna.

El ejemplo muestra lo sencillo que será alcanzar el primer nivel de la torrebot con el sistema diseñado.

2.1.3. Nivel de Control

El robot ya sería capaz de contener diferentes programas y de realizar acciones más controladas e inteligentes, para ello debe contar con un controlador y unos algoritmos adecuados.

En este caso el controlador seguirá siendo la FPGA, pero a diferencia del ejemplo anterior podrá contar con acciones más complejas gracias a la memoria, es decir, podrá implementar autómatas de estados finitos que le permitan aumentar sus posibilidades.

En el ejemplo anterior del seguimiento de línea existe un problema si ambos sensores se encuentran sobre el color blanco, pues el robot se detiene. Esto se evita si es capaz de recordar en que sentido quedó la línea la última vez que la perdió, derecha ó izquierda, y según este dato girar en ese sentido. En este punto el robot ya es capaz de recordar y no únicamente de reaccionar ante el estímulo actual.

En los siguientes puntos se tratará la electrónica y software que servirán como herramientas para conseguir alcanzar estos dos niveles. Mientras que en el tercer tema se expondrá como llegar al nivel de inteligencia con dichas herramientas.

2.2 Tarjeta de Control TC-FPGA

2.2.1 Objetivo

Realizar una tarjeta de control basada en un una FPGA, siguiendo la filosofía de fácil montaje y componentes asequibles. No contendrá ningún periférico de control para el robot, ya que esto forma parte del siguiente diseño, de modo que se tratará de una placa genérica, con líneas de E/S al exterior y con un soporte para la programación y depuración de los programas empotrado en la misma. Por tanto se podrá considerar un sistema genérico para desarrollos basados en FPGA. Esto quiere decir que además de servir para controlar un robot, la tarjeta se podrá utilizar para diseños de control distribuido, adquisición de datos, domótica ...

2.2.2 Diseño a nivel de bloques

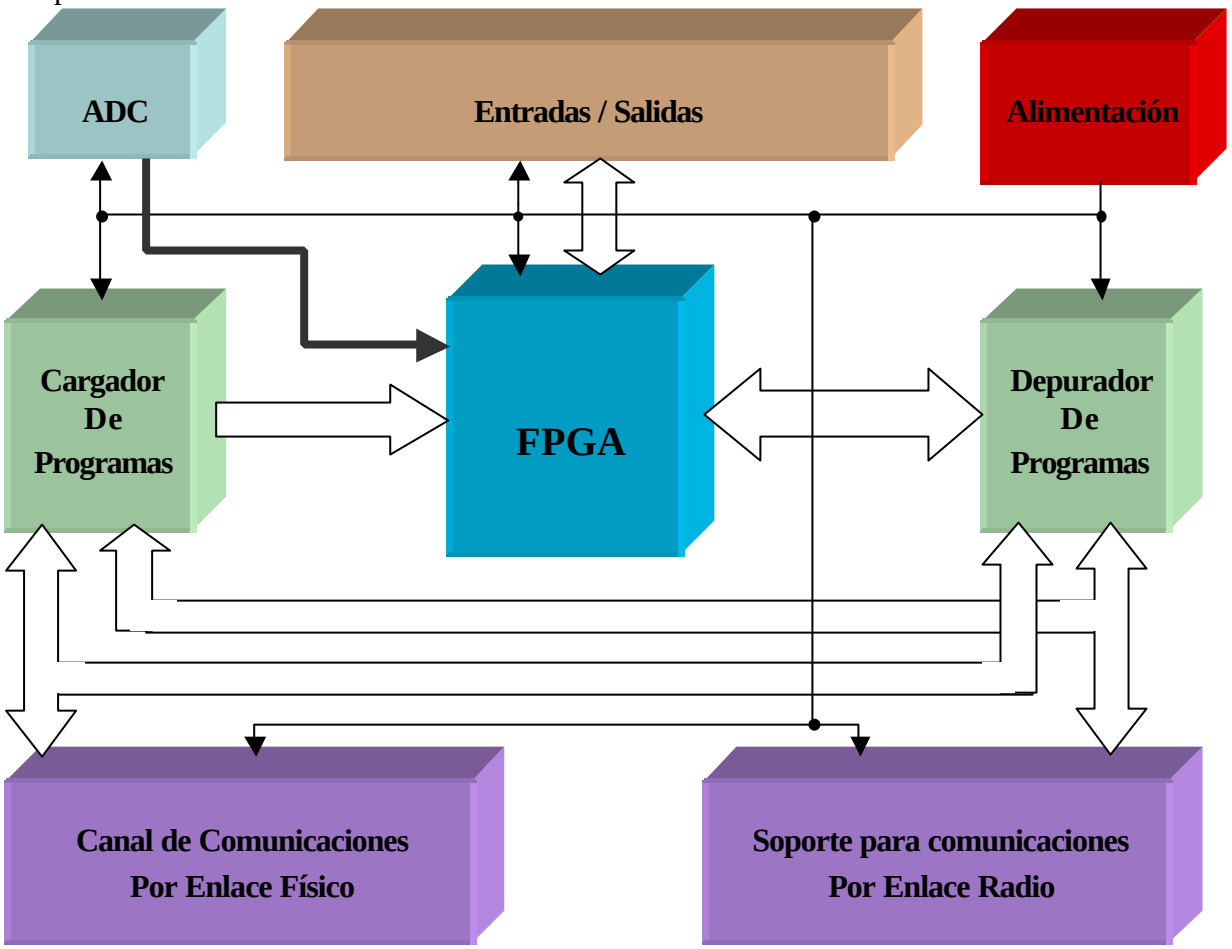
De ahora en adelante se denominará a la programación de la FPGA como configuración de la misma, ya que es este un termino mas adecuado, al tratarse de configurar el hardware que lleva en su interior.

La configuración de las FPGAs, normalmente se realiza mediante JTAG[1]. Pero debido a que su configuración es almacenada en celdas de memoria volátil, dicha configuración debe ser cargada de nuevo cada vez que se alimente el sistema. Para este objetivo existen memorias E2PROM[[1]] capaces de almacenar dichas configuraciones y automáticamente configurar la FPGA al encenderse. Sin embargo no se utiliza dicha configuración aún siendo la más habitual. Como ya hemos comentado, uno de los objetivos de la placa consiste en la posibilidad de depurar los programa contenidos en la FPGA, y el sistema está orientado hacia la construcción de un robot. Uno de los problemas de depurar este tipo de sistemas, o de cualquier sistema de tiempo real que interactúe con el medio, es que deben ser probados en el ambiente en que actúen, ya que este es dinámico y muy difícilmente caracterizable. Por ello se proveerá al sistema de un mecanismo para poder observar las variables que deseemos en tiempo real, desechando de este modo cualquier sistema habitual de depuración mediante la emulación.

Otro de los inconvenientes al tratarse de un robot móvil radica precisamente en su movilidad, ya que realizar una depuración mediante un enlace por cable resulta engorroso en muchos casos. Por otro lado, se plantea en los objetivos del proyecto que la plataforma diseñada de soporte para alcanzar el nivel comunidad de robots. En este caso se contaría con un recinto en el que trabajarían varios robots colaborando entre sí, y depurar un sistema de este tipo conlleva programarlos multitud de veces, así pues el sistema ideal debería dar soporte a la programación y depuración mediante un enlace radio. Se consideró que esta parte era lo suficientemente compleja como para implementarse por separado, y así se hizo en el proyecto ***"SISTEMA DE COMUNICACIÓN VÍA RADIO BASADO EN PC DESTINADO A UNA COMUNIDAD MICROBÓTICA "***[[2]]. Sin embargo en este proyecto se provee de los mecanismos necesarios para poder acoplar el enlace por radio.

[1] Se trata de un estándar para depurar y programar circuitos integrados.

Se muestra a continuación el diagrama de bloques a nivel funcional que implementará la placa de control:



Descripción de cada uno de los bloques así como su funcionalidad:

- **FPGA:** Este es el núcleo de la placa de control y donde se ejecutarán los algoritmos. Está constituido por la FPGA y toda la circuitería adicional necesaria para su funcionamiento.

- **Entradas/Salidas:** A través de este modulo son accesibles los pines de E/S de la FPGA, además de las alimentaciones.

- **Alimentación:** Proporciona de alimentación a cada uno de los módulos de la placa, asignando distintas tensiones a cada módulo según sea necesario.

- **ADC:** Puesto que la plataforma está orientada a robótica, será necesario trabajar con sensores, los cuales en muchos casos son analógicos, por ello se decide integrar canales AD en el sistema.

- **Cargador de Programas:** Se encarga de gestionar la carga de los programas en la FPGA así como de almacenarlos en una memoria no volátil cuando son recibidos por alguno de los canales de comunicaciones.

- **Depurador de Programas:** Pone en marcha el sistema de depuración, haciendo de puente entre la FPGA y uno de los canales de comunicaciones.

- **Canal de comunicaciones por enlace físico:** Implementa la conexión mediante algún protocolo estándar de comunicaciones, entre el sistema y un PC mediante un cable de comunicaciones.

- **Soporte para comunicaciones por enlace radio:** Proporciona los recursos necesarios para poder adaptar el enlace por radio del proyecto antes mencionado.

2.2.3 Implementación de los bloques

A continuación se muestra la forma en que se han implementado los bloques así como su motivo.

2.2.3.1 FPGA

Se realizó un estudio de las distintas FPGAs del mercado, evaluando fabricantes como ALTERA, XILINX, ATMEL... La elección final recayó sobre el fabricante **ALTERA** y su FPGA **10K10LC84 [3]** de 10.000 puertas. Los motivos de esta elección se basaron en varias causas:

- **Sistema de desarrollo:** en el caso de ALTERA el entorno y las herramientas de diseño eran conocidos puesto que ya se había trabajado con ellos. Además existe una versión de su sistema que es gratuita y nos permite programar en distintos lenguajes (VHDL, AHDL, Verilog y Esquemáticos).

- **Disponibilidad:** Una cuestión importante, ya que no es un CI demasiado extendido aun, no es fácil conseguirlo, y su precio es relativamente elevado. Las FPGAs Altera se pueden conseguir sin demasiadas dificultades.

- **Encapsulado:** Por los dos puntos anteriores se decidió utilizar Altera como fabricante, y la elección de la pieza 10K10LC84 vino condicionada por la filosofía de montaje asequible, ya que este CI tiene un encapsulado PLCC de 84 pines, muy sencillo de montar a través de un zócalo. Otra ventaja es que al ir montada en zócalo la pieza puede ser sustituida sin problemas en caso de rotura (se recuerda que se trata de

una plataforma de aprendizaje, en las cuales son relativamente frecuentes estos incidentes).

Una vez decidida la pieza, para completar el bloque es necesario incluir la circuitería precisa para hacer funcionar este circuito integrado. Esta es bastante sencilla, y está constituida básicamente por resistencias de pull-up y condensadores de filtrado de alimentación, todos ellos recomendados por el fabricante. También se incluyen:

- **Oscilador:** Es necesario para generar el reloj del sistema. La FPGA posee dos entradas para este propósito, pudiendo utilizar dos relojes de distinta frecuencia dentro de la misma pieza. Se decidió colocar un oscilador de **2 HMS**, ya que pudiendo soportar frecuencias de trabajo mucho mayores se estimó que esta frecuencia es suficiente para el procesamiento al que está encaminado el diseño. Por otro lado el uso de frecuencias mayores aumenta el consumo, y puesto que se integrará en un sistema alimentado mediante baterías no deben realizarse gastos innecesarios. La otra entrada de reloj es conectada al módulo de E/S para usos futuros.

- **Pulsador de Reset:** puesto que se realizarán diseños síncronos, es necesario proveer al sistema con un pulsador de reset, el cual también es accesible a través del módulo de E/S para usos futuros.

- **JTAG:** Se incluye un conector para el programador **Byteblaster[4]** mediante el protocolo JTAG, contemplando de este modo la posibilidad de programar la FPGA directamente. Esto permite depurar posibles fallos en el módulo de programación, y a su vez se envían los pines del jtag al módulo de E/S, para usos futuros.

Se muestra a nivel de esquemático como queda este bloque:

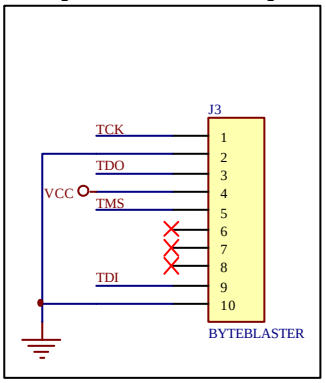


Fig 2.2.1: Conector BYTEBLASTER.

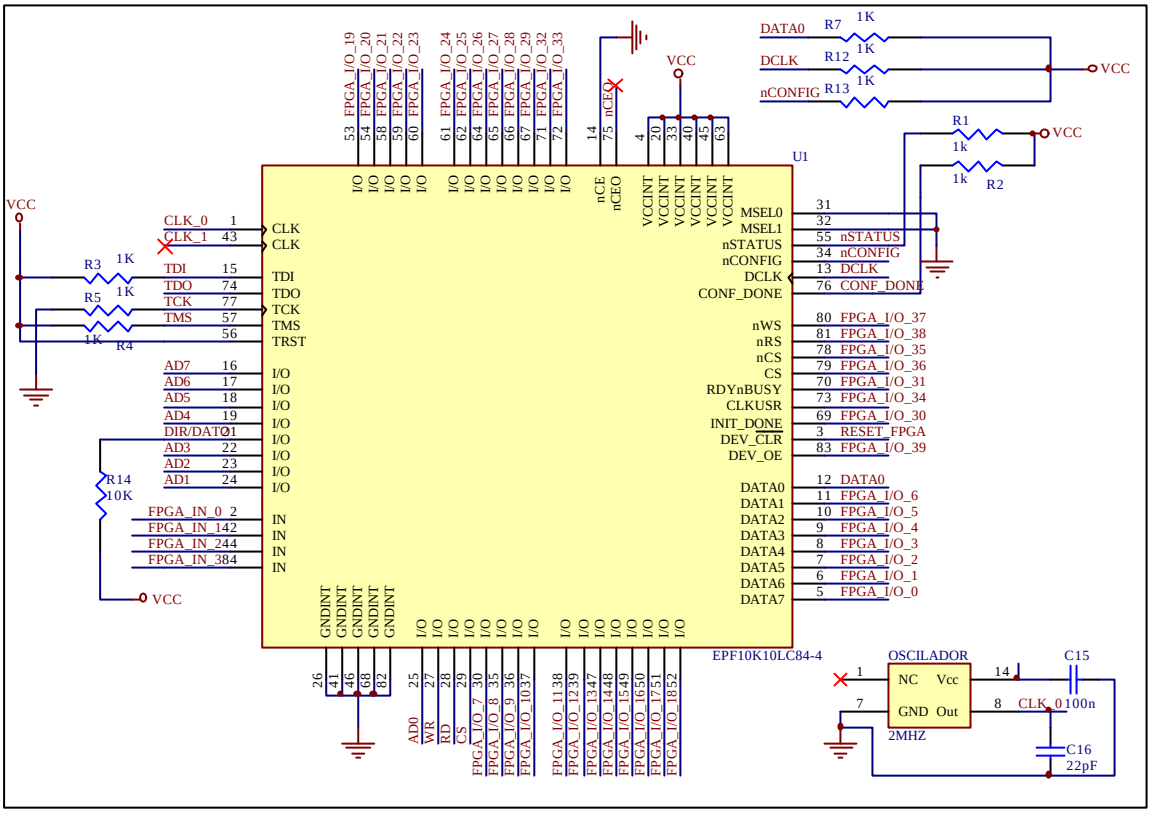


Fig 5.2.2 Conexiones de la FPGA.

Los nombres de las líneas de la FPGA corresponden a su interconexión con otros módulos, las cuales se irán viendo a continuación.

2.2.3.2 Módulo de Entradas / Salidas

Aunque muy simple esta parte del diseño es bastante decisiva, ya que marcará el aspecto del sistema completo y sobretodo condicionará su forma de uso y expansión. Solamente consiste en colocar conectores para dar salida a las entradas y salidas de la FPGA, alimentaciones y ciertas señales como el programador jtag o la segunda entrada de reloj de la FPGA. Por tanto se trata de elegir los conectores adecuado para este fin.

Se barajaron distintas posibilidades tales como conectores de cable de bus, conectores molex, tiras de pines... Sin embargo esta placa no es la única en el sistema, sino que las conexiones de los sensores o actuadores de realizarán a través de otras placas. Normalmente este tipo de arquitectura modular se realiza apilando las placas en torre, de este modo se ahorra espacio, pero para llevar las señales de unas placas a otras es necesario el uso de cables, que suelen resultar bastante engorrosos.

Para este diseño se ha optado por 2 conectores PC104[5] de 32x2 líneas cada uno. La peculiaridad de estos conectores reside en que pueden ser apilados verticalmente (fig 5.2.3).

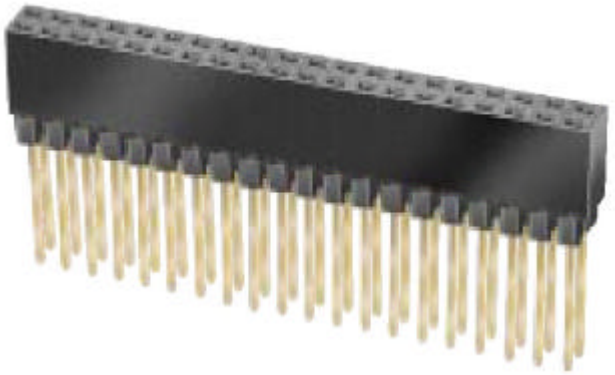


Fig 5.2.3: Conector de entradas / salidas.

Gracias a estos conectores se puede hacer pasar todas las líneas de E/S, alimentaciones y demás señales de una placa a otra, y además elimina la necesidad de separadores y cables. Otra peculiaridad de este sistema es que se tienen accesibles todas las líneas desde la última placa, esto ayuda a la hora de medir alguna señal de una placa que esté por debajo sin necesidad de desmontar las que estén por encima, proporcionando gran maniobrabilidad.

En total se dispone de 128 líneas de paso entre placas a través de estos conectores, pero no se usan todas. La explicación es que se dejan líneas libres para posibles ampliaciones, es decir, si se diseñan otras placas que deban pasar líneas de datos entre sí pero que no afecten a la placa de control.

En cuanto a la alimentación, también es pasada de unas placas a otras a través de estos conectores, pero solamente la **alimentación no regulada**, de este modo en cada placa se puede diseñar el regulador que necesite, tanto a nivel de tensión como de consumo de corriente. La disposición de las alimentaciones y las masas en estos conectores se ha realizado de forma espaciada y simétrica, de modo que al colocar otra placa al revés las alimentaciones coincidan y no puedan producirse cortocircuitos.

En la figura 2.2.4 se muestra a nivel de esquemático como queda este bloque.

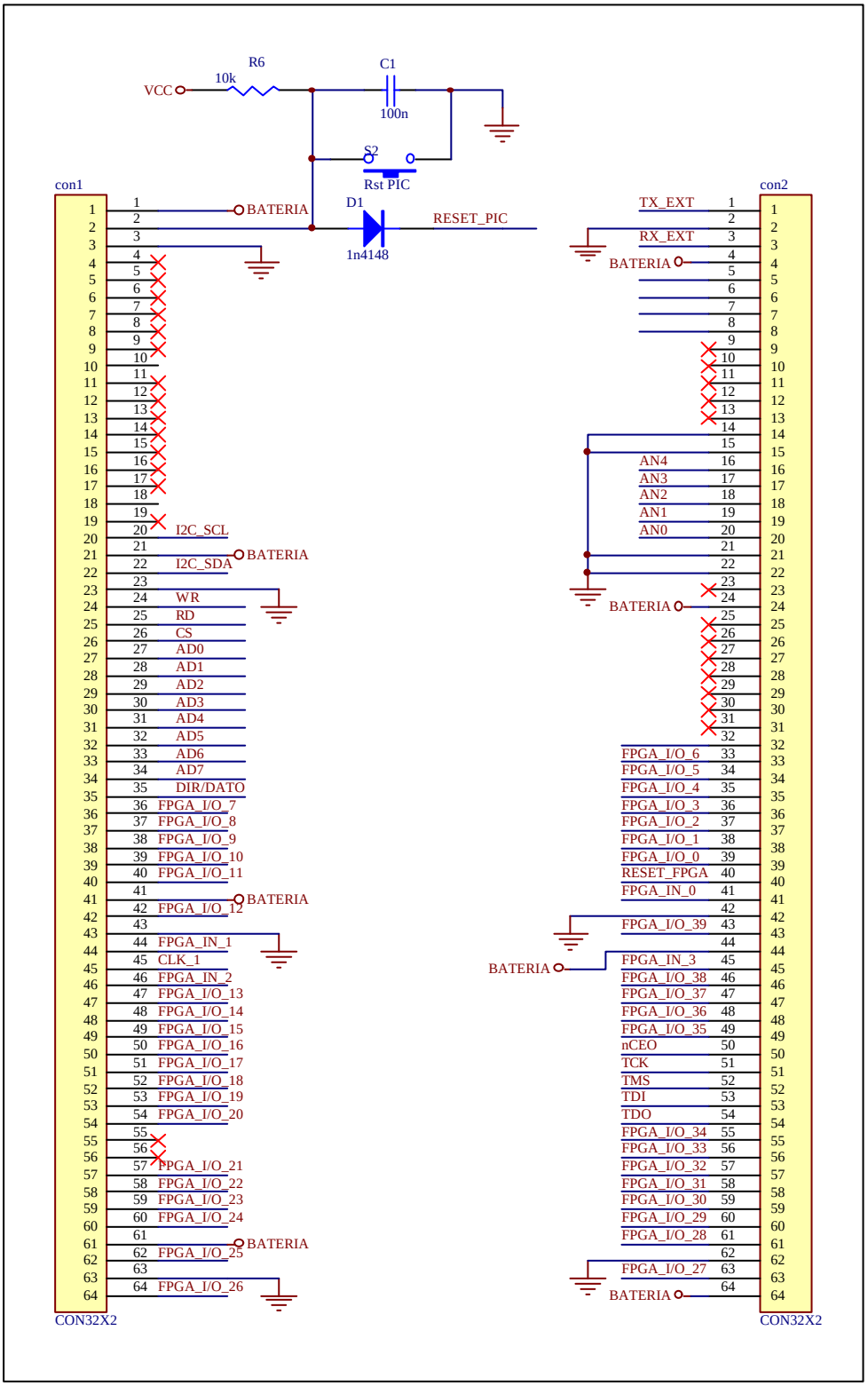


Fig 2.2.4: Conexiones del módulo de entradas / salidas

2.2.3.3 Módulo de Alimentación

Su diseño es bastante sencillo ya que toda la circuitería de esta placa se alimenta a 5V DC, con consumos de corriente reducidos. Por ello se ha elegido un regulador lineal, concretamente el popular 7805. Permitiendo esto alimentar la placa con tensiones desde 6.5 a 12 Voltios. A continuación se muestra el módulo a nivel de esquemático:

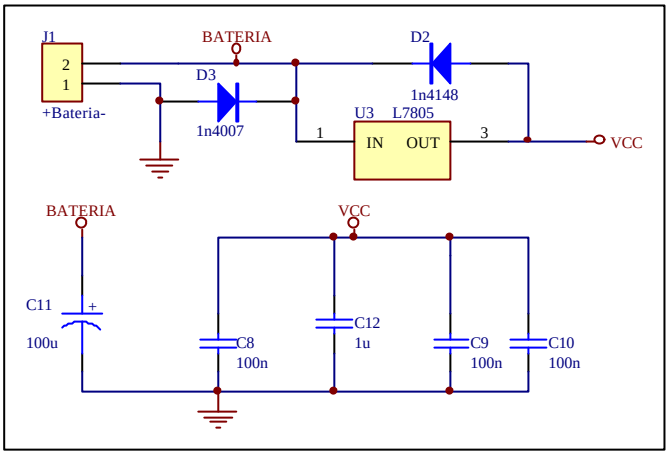


Fig 2.2.5: Módulo de alimentación

Se puede observar que el regulador no tiene conexión a GND, en realidad si la tiene pero está oculta en el diseño[1]. Los condensadores C8, C9 y C10 son condensadores de desacoplo para los circuitos integrados de la placa.

2.2.3.4 Módulo ADC + Cargador de programas + Depurador de programas

Para el diseño de estos tres módulos se ha tomado la decisión de usar un microcontrolador con el fin de unir los tres en uno solo, y realizar un software específico para llevar la tarea a cabo.
El µC elegido es el **PIC16F877[6]**, esta elección se hace en base a tres motivos:

- **µC muy asequible**: es un micro muy extendido y conocido, lo que lo hace fácilmente obtenible y a precio reducido.

- **µC conocido**: es un CI con el que se ha trabajado y por tanto se conoce perfectamente, acortando tiempos en el diseño e implementación.

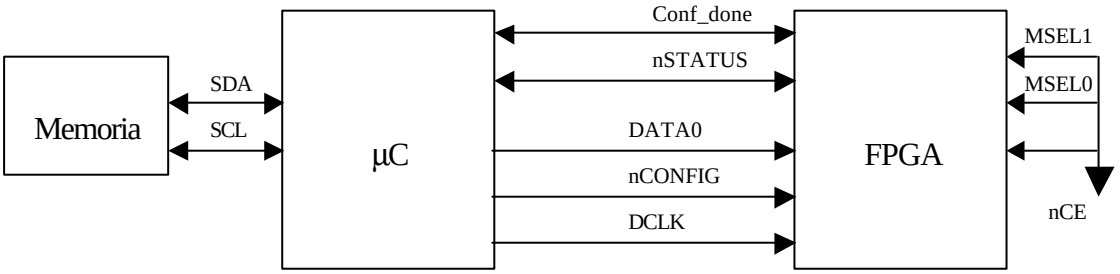
- **Compilador C**: Además se cuenta con un compilador cruzado de C ANSI para este µC.

[1] El programa de diseño de esquemáticos PROTEL99SE permite ocultar las alimentaciones y masas.

Además del µC es necesario incluir una memoria externa no volátil en la que se almacenen las configuraciones de la FPGA, para este propósito se ha optado por una memoria EEPROM con protocolo I2C[1] por su reducido tamaño. Otra de las características de estas memorias es la equivalencia de modelos, es decir, con el mismo pinout y el mismo protocolo existen memorias de distintos tamaños. La memoria elegida es de 256Kbits ó 32Kbytes, teniendo en cuenta que una configuración de la FPGA ocupa un tamaño fijo de 14.751 bytes, hay capacidad para 2 configuraciones distintas. Ya están disponibles en el mercado memorias de este tipo de hasta 1Mbit de datos, lo que podría ser aprovechado en el futuro para dar cabida hasta 8 configuraciones distintas.

Se pasa a describir el modo de interconexión entre el µC y la FPGA para cumplir con los objetivos de los 3 módulos:

- **Configuración de la FPGA**: estos CI están provistos de varias formas para ser configurados. Para este caso se utiliza una programación en serie, denominada ***PS configuration with microprocesor*[7]** por el fabricante. En este modo solamente son necesarias 5 líneas de E/S entre el µC y la FPGA, y una memoria externa ya mencionada. Usando esta configuración el micro puede saber si la FPGA se configuró con éxito o si tuvo problemas, teniendo un control total durante todo el proceso. A continuación se muestra el modo de conexionado entre ambos CI:



Configuración PS con microprocesador

- **Depuración de programas**: Como ya se mencionó, en este caso el µC debe establecer un puente entre la FPGA y un canal de comunicación. El canal elegido es un puerto serie, ya que la conexión se realizará hacia un PC, siendo este el modo más adecuado, además el µC ya posee una UART lo que simplifica el diseño. Del otro lado se debe establecer una comunicación entre µC y FPGA. El modo más adecuado es en paralelo, ya que el tratamiento para la FPGA es más sencillo. Para la implementación de esta comunicación se optó por:

  - 8 líneas compartidas por los datos y las direcciones
  - 4 líneas de control:

[1] Bus de comunicaciones serie síncrono con direccionamiento para la comunicación entre circuitos integrados, creado por Philips semiconductors.

- WE: habilitación de escritura
- RE: habilitación de lectura
- CS: selección de chip
- DIR/DATO: selección de dirección o dato

El uso de la línea de CS es para futuras ampliaciones, ya que otro dispositivo podría compartir este bus de comunicaciones. Con este tratamiento se establece un mapa de memoria compuesto de un máximo de 256 bytes, su forma de uso se mostrará en el apartado ***2.2.5***.

- **ADC:** Para proveer al sistema de canales ADC se hace uso de los que ya tiene el µC. En cuanto al modo en que este le pasa los datos a la FPGA resulta lógico que sea el puerto paralelo que se implementa para la depuración de programas. El µC se tiene un total de 5 canales ADC libres.

Estos tres bloques tienen el siguiente aspecto a nivel de esquemático:

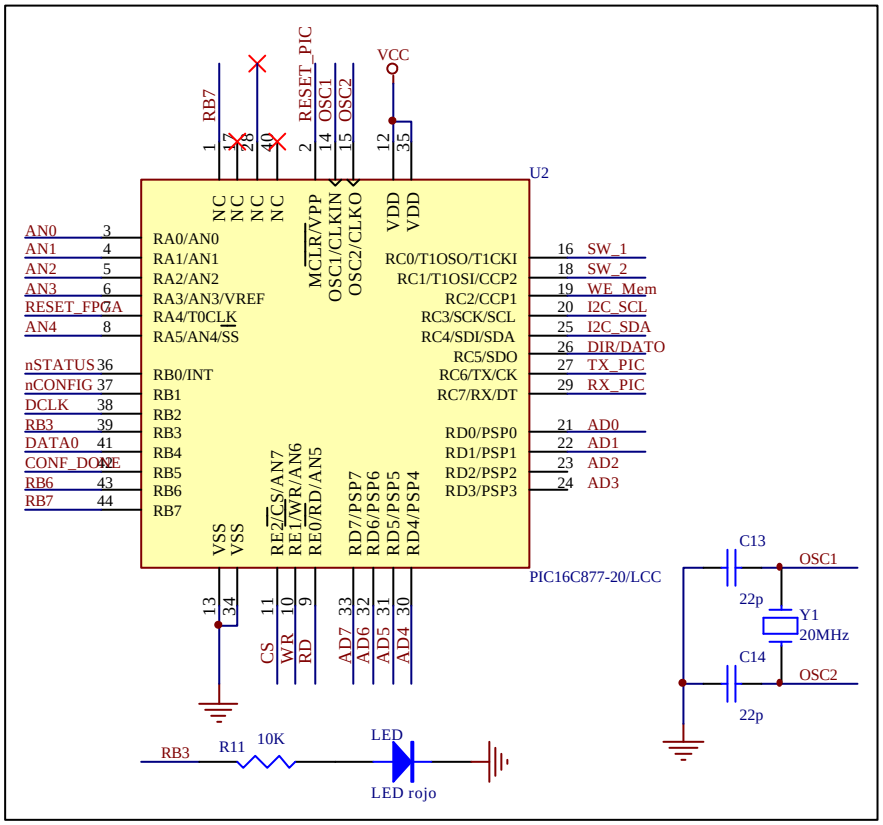


Fig 2.2.6: Conexiones del µC

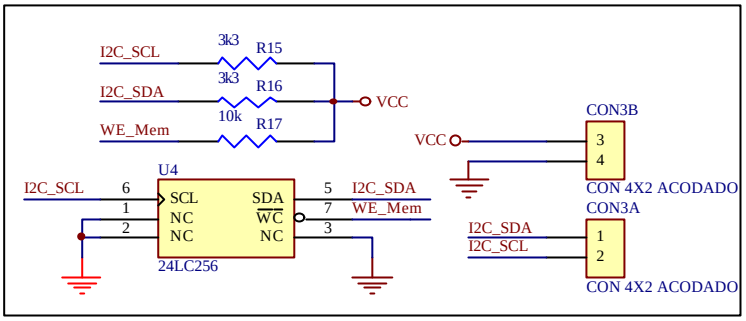


Fig 2.2.7: Módulo de memoria.

Debido al uso de un µC se ha añadido circuitería adicional para este:

- **Led**: Para indicar eventos (fig 2.2.6)
- **Conexión del reset de la FPGA**: con el objetivo de ofrecer este servicio a través de alguno de los canales de comunicación. (fig 2.2.8)
- **2 switches**: Con el fin de introducir cambios sobre los programas del µC (fig 2.2.9)
- **ICD**: proporciona una conexión para el ICD (In Circuit Debuger)[8] , se trata de una herramienta para programar y depurar el programa del microcontrolador PIC16F877. (fig 2.2.10)

Mostramos estos elementos a nivel de esquemático:

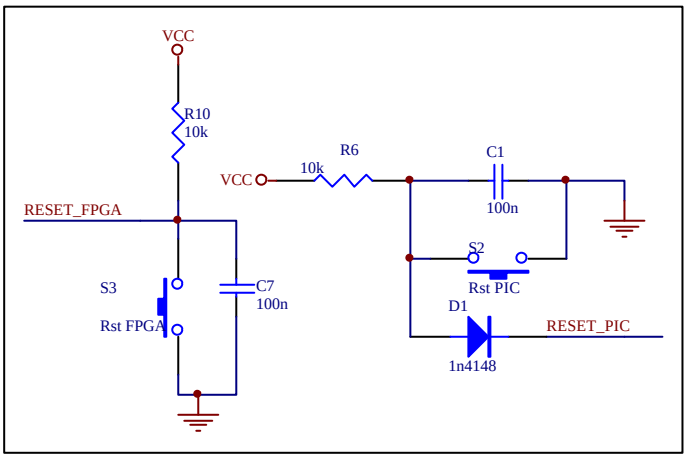


Fig 2.2.8: Reset del µC y FPGA

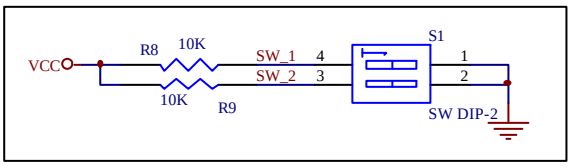


Fig 2.2.9; Switches del sistema para seleccionar modo arranque

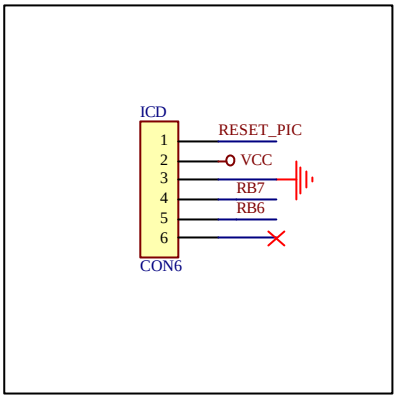


Fig 2.2.10: Conector para ICD

2.2.3.5 Canal de comunicaciones por soporte físico

Como ya se ha comentado implementaremos el canal físico a través de un puerto serie con niveles RS232 para comunicarnos con un PC. Puesto que el µC ya posee una UART solamente es necesario adaptar los niveles TTL a RS232, para ello hacemos uso del popular MAX232 [11], usando el circuito recomendado por el fabricante como muestra la fig 2.2.11:

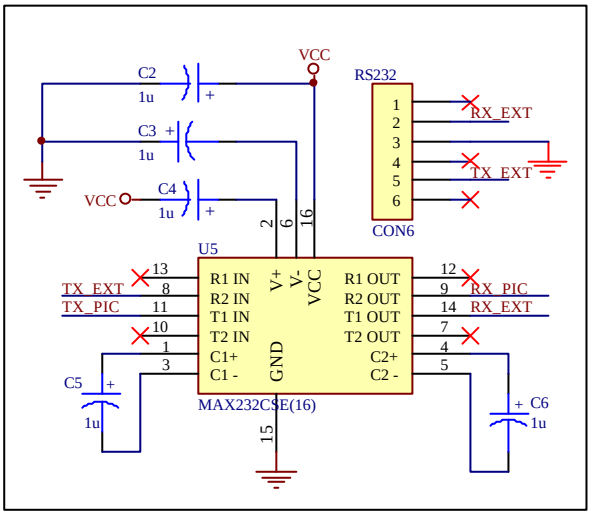


Fig 2.2.11 Adaptador de niveles MAX232

2.2.3.6 Soporte a comunicaciones por Enlace Radio

Parece claro que las comunicaciones podrían realizarse por el mismo canal que el físico, simplemente colocando un enlace radio por puerto serie, sin embargo por el diseño implementado en este enlace[3] se han otorgado dos canales para realizar estos servicios. Por un lado contamos con el canal físico, en el que puede substituirse el cable por un enlace radio convenientemente adecuado, y por otro se

prestan los mismos servicios a través del canal I2C, por ello se enlaza el canal I2C con el módulo de E/S, como muestra la ilustración 4.

El permitir los servicios a través del canal I2C permitirá en el futuro el uso compartido de la radio dentro del mismo sistema, ya que el protocolo I2C provee de una dirección a cada elemento

2.2.4 Construcción y Versiones

En total se han realizado 3 versiones de esta placa. Los esquemáticos e ideas mostrados pertenecen a la última versión, aunque las diferencias entre las tres versiones son mínimas.

V 1.0

Cumple con las ideas principales antes expuestas, sin embargo el conector utilizado para el ICD y el puerto serie del PC hacían necesaria la confección de un cable especial. También se observó la conveniencia de algunas resistencias de Pull-up que no se habían previsto. No existía ningún LED u otro dispositivo para hacer notar eventos y tampoco había protección contra alimentaciones inversas.

V 2.0

Se añadió un LED a un pin sobrante del µC, ya que se observó su necesidad para mostrar ciertos eventos que se comentarán en el siguiente punto. También se dotó esta versión con protección contra alimentaciones inversas y se modificaron los conectores para el ICD y puerto serie, de modo que encajasen con cables estándar.

V 3.0

Esta versión, la definitiva, se mandó fabricar en una empresa, por tanto se podía contar con algunas ventajas como taladros metalizados y distancias entre pistas más cortas. Por este motivo se rehicieron algunas partes del PCB, dándose cabida a:

- Doble huella para el oscilador de la FPGA, lo que permite utilizar osciladores de cualquier frecuencia
- Las líneas de comunicación entre FPGA y el µC se incorporaron a los conectores de E/S.
- Un conector para el I2C con alimentaciones fue añadido, fuera del módulo de E/S, para usos futuros.
- La disposición de los componentes se hizo más efectiva.

A continuación se muestran fotografías de la placa V.2.0, ya que la versión 3 no ha sido montada en el momento de realizar esta documentación, aunque los cambios son lo suficientemente pequeños para que no se produzcan fallos inesperados.

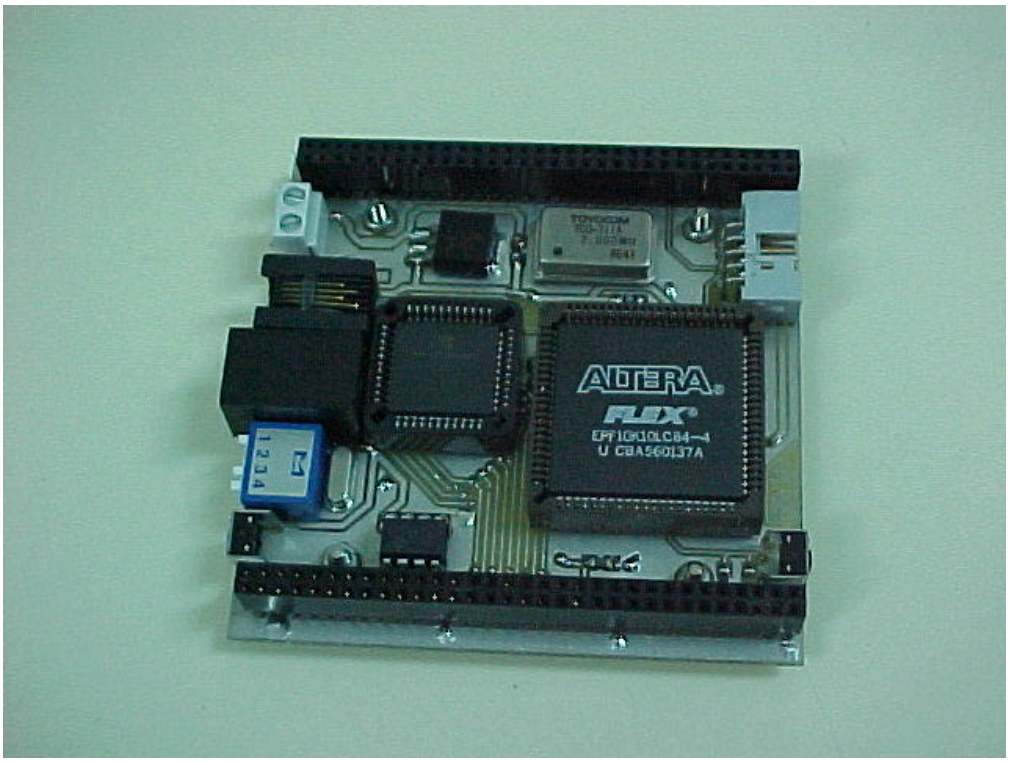


Fig 2.2.12: Foto desde arriba del prototipo.

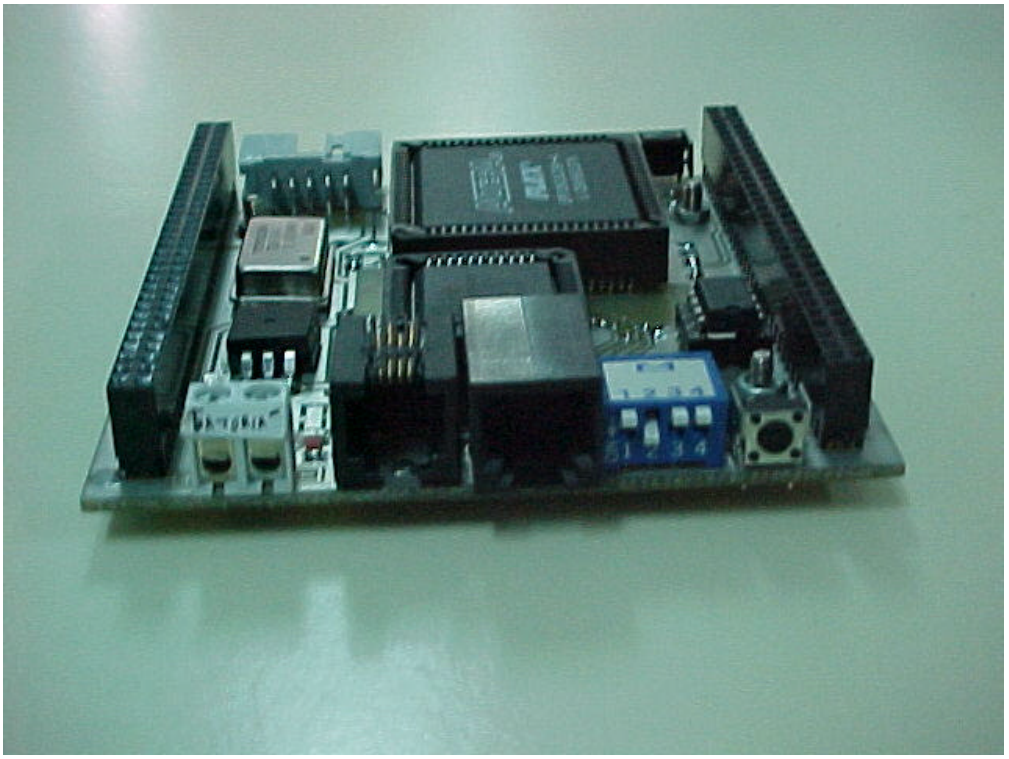


Fig 2.2.13: Foto lateral del prototipo.

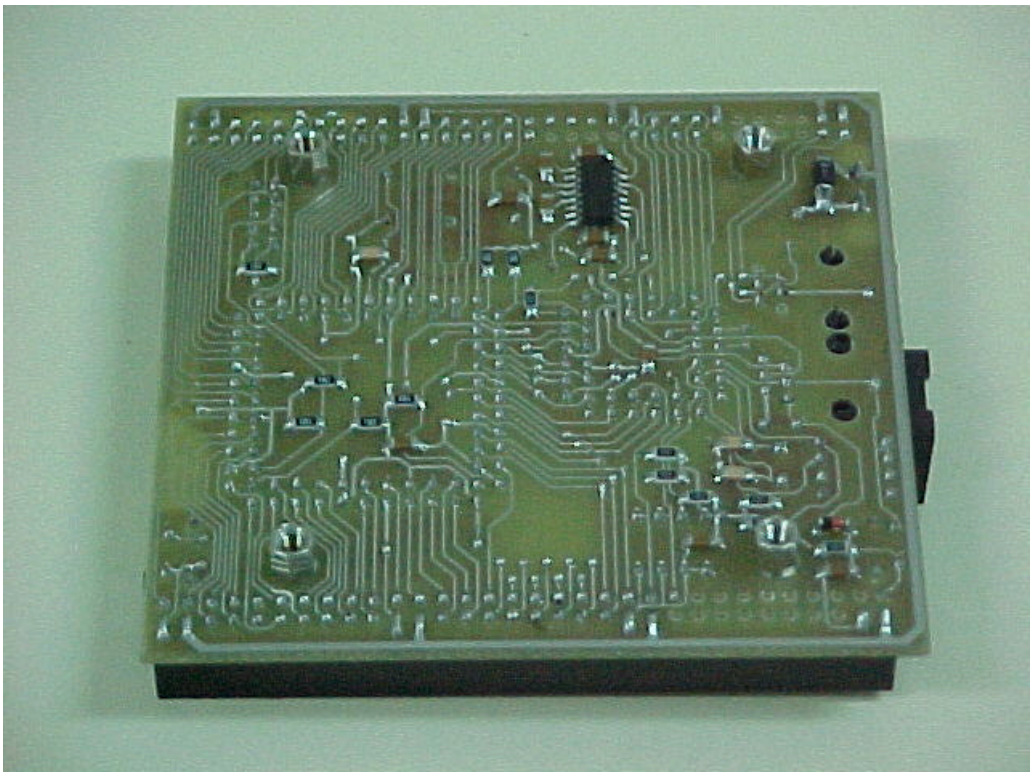


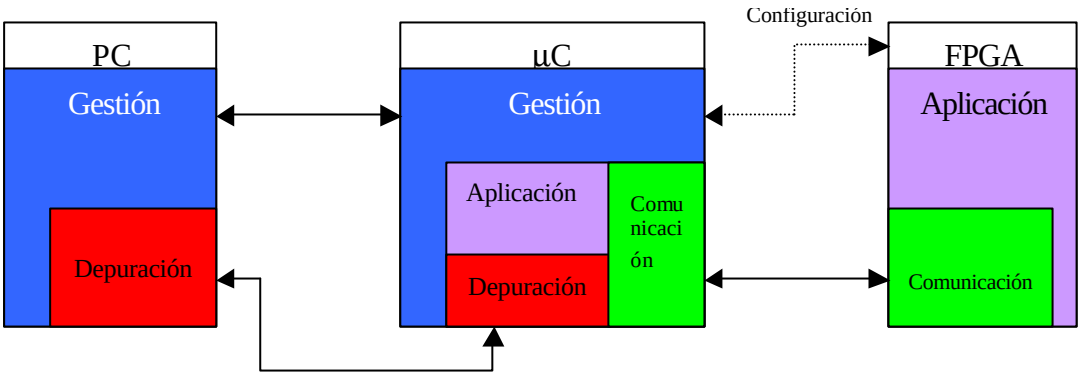
Fig 2.2.14: Foto desde abajo del prototipo.

Para realizar el PCB se tuvo especial cuidado en realizar las pistas lo más cortas posibles, todo el ruteado fue manual y se realizó un polígono de masa en la cara top de la placa. En el caso de las señales de los conversores AD se incluyó un anillo de masa para protegerlas en su camino hacia los conectores de E/S. También se tuvo especial cuidado en el ancho de pista de las señales de alimentación, para evitar calentamientos excesivos que pudieran romperlas en determinadas situaciones.

2.2.5 Software

Para conseguir los objetivos marcados para este sistema se necesita programar tres dispositivos, FPGA, µC y el PC. El sistema está compuesto de FPGA + µC, este último se encarga de la gestión de las configuraciones de la FPGA, de configurarla cuando sea requerido y de proveer de un sistema para depuración de la FPGA en tiempo real. Además el µC podrá contener programas de usuario que no se solaparán en ningún momento con las tareas anteriores. Por tanto se pueden realizar configuraciones para la FPGA, depurarlas y después programar el µC para que utilice los recursos programados en la FPGA, ó también se puede programar la FPGA como corazón del sistema y utilizar el µC como esclavo de esta para proveerle de recursos como los canales ADC, UART, I2C, operaciones matemáticas complejas ó cualquier otra utilidad, haciendo uso de los recursos de comunicación entre ambos dispositivos.

La siguiente figura muestra las distintas capas de software que se generan y como se comunican entre sí:



- **Módulos de Gestión**: Son los encargados de almacenar las distintas configuraciones de la FPGA, configurarla cuando sea requerido, así como de cargar las aplicaciones del usuario para el µC, ponerlas en marcha y activar el modo depuración cuando sea preciso.

- **Módulos de Comunicación**: Son capaces de comunicarse entre sí para gestionar el traspaso de datos entre FPGA y µC, de modo que este proceso sea sencillo y transparente al usuario.

- **Módulos de Depuración**: Utilizados para depurar los programas de la FPGA en tiempo real mediante el traspaso de variables entre el PC y la FPGA. Hacen uso de los módulos de comunicaciones.

- **Módulos de Aplicación**: Son los programas que el usuario creará, tanto para la FPGA como el µC. Pueden hacer uso de los módulos de comunicaciones.

A continuación se explicarán detenidamente cada uno de los módulos software.

2.2.5.1 Módulos de Gestión

Existen 2, uno para el PC y el otro para el µC. Ambos se comunican entre sí para realizar la tarea a través de un puerto serie RS232 mediante un protocolo de comandos ASCII que se explica con detalle en los anexos A y B. A continuación se explica con detalle ambos módulos:

Módulo de gestión del µC

Este programa toma el control del µC en cuanto este es conectado, según el estado de los switches de la tarjeta comenzará en modo programación ó en modo normal:

- **Modo programación**

Permite la descarga aplicaciones de usuario tanto para la FPGA como el propio µC, así como la entrada al modo depuración. Como ya se ha indicado realizará la gestión de las configuraciones de la FPGA ya que es posible almacenar 2 configuraciones distintas al mismo tiempo. Existen 2 bancos de configuración denominados:

- **Definitivo**: Almacena la configuración de la FPGA considerada como preferente, o lo que es lo mismo, una configuración ya probada y depurada.
- **Pruebas**: Contiene una configuración que se está probando y depurando.

Las configuraciones descargadas desde el PC siempre son almacenadas en el banco de pruebas, de este modo no se pierde la configuración definitiva en caso de no estar bien depurada la aplicación. Fundamentalmente se ideó este sistema de bancos para realizar una telecarga de las aplicaciones de usuario, pudiendo conmutar en cualquier momento al programa definitivo en caso de no funcionar adecuadamente el nuevo programa.

En el modo programación se ofrecen unos servicios a través del puerto serie mediante un protocolo ASCII, los servicios son:

- **Versión de Software**: Devuelve la versión del módulo de gestión del µC. Puede utilizarse para ofrecer o denegar algunos servicios en el módulo del PC si en la versión que corre en el µC no estuviesen implementados. También se usa para chequear el estado de la conexión.

- **Descarga Aplicación FPGA**: Permite la descarga de una aplicación de usuario para la FPGA, que será grabada en el banco de pruebas.

- **Descarga de Aplicación µC**: Permite la descarga de una aplicación de usuario para el µC, que será ejecutada en el modo normal.

- **Conmutación de Bancos**: Pasa un programa del banco de pruebas al banco definitivo.

- **Acceso al modo depuración**: Se accede al modo de depuración en tiempo real, pudiendo visualizar y modificar registros de la FPGA en tiempo real. Al salir del modo depuración se regresa al modo programación.

Durante la descarga de aplicaciones para el μC se realiza un chequeo de los datos recibidos inherente al formato de estos, INTEL 8 bits[9], pero en el caso de la FPGA no tienen formato con control de errores, ya que se trata del formato Raw Binary File RBF[10], sin embargo al no observarse ningún problema durante las pruebas de descarga se mantuvo este formato por ser de menor tamaño y más rápido de descargar.

En cualquier caso siempre se chequea la integridad del canal de comunicaciones, de modo que no se valida una aplicación si esta no ha sido descargada completamente.

- **Modo normal**

Es el modo requerido cuando se quiere ejecutar programas de aplicación. Primero configura la FPGA con un programa de aplicación, del banco de pruebas o el definitivo, según el estado de los switches, siempre y cuando el banco contenga una aplicación que se halla descargado correctamente. En segundo lugar delega el control al programa de aplicación del μC si es que lo tiene y es válido.

En caso de no contener una aplicación válida para la FPGA el sistema queda detenido. En la versión 1.0 esto provocaba una entrada en el modo sleep del microcontrolador, pero en las versiones posteriores se indica este hecho mediante un parpadeo del LED.

Módulo de Gestión del PC

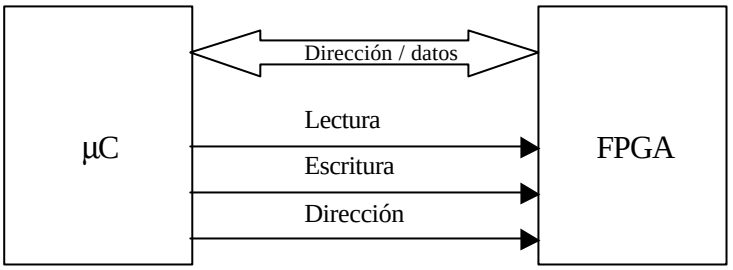
Es un sencillo programa que hace uso de los servicios de la placa en modo programación, es decir, el módulo de gestión del PC solamente funciona con el módulo de gestión del μC en modo programación. Mediante estos servicios se ofrece al usuario la descarga de sus aplicaciones así como la depuración de las mismas.

En el caso de este proyecto se ha implementado en MS-DOS y mediante un programa de menús. Sin embargo resultaría muy sencillo implementar este módulo en otro sistema operativo, ya que la mayor parte del trabajo lo hace el módulo de gestión del μC.

2.2.5.2 Módulos de Comunicación

Se realizan 2, uno para el μC y el otro para la FPGA, e implementan un puente de comunicaciones entre ambos dispositivos. Las características de la misma son:

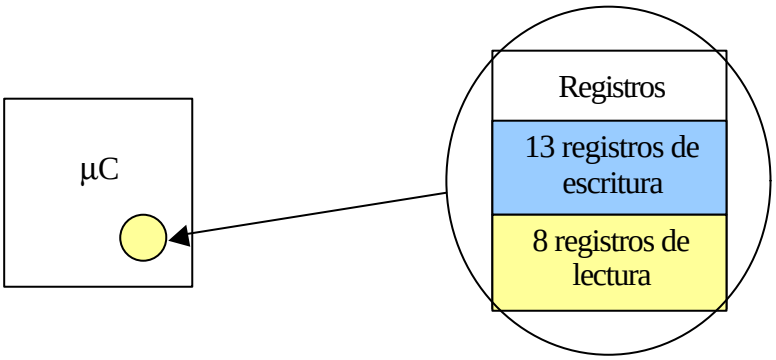
- Comunicación paralela y bidireccional
- 8 bits de datos
- 8 bits de dirección
- Dirección y datos multiplexados
- 3 bits de control



El control de las comunicaciones lo tiene el µC, y la FPGA atiende a las peticiones de este. Se ha implementado de este modo para que la ocupación del módulo en la FPGA sea lo menor posible. El hecho de tener un espacio de direcciones de hasta 8 bits nos permite observar la FPGA desde el punto de vista del µC como un conjunto de 256 registros de escritura y 256 registros de lectura. Sin embargo normalmente se utilizarán muchos menos, por lo que ambos módulos manejan 8 registros de lectura y 13 de escritura, ya que 5 de los registros de escritura se utilizan para proveer a la FPGA de los canales ADC. El aumento del número de registros puede hacerse a nivel de código de una forma muy sencilla.

El punto de vista de cada dispositivo es el siguiente:

- **µC**: Se han implementado dos funciones en código C que permiten la lectura y la escritura a unos registros. Desde este punto de vista se puede manejar:



Para el µC es como si los registros estuviesen en su interior. Según la aplicación de la FPGA estos registros tendrán diferentes funciones.

- **FPGA**: Se implementa mediante un módulo que contiene registros de entrada y salida, según lo que programemos en el µC los datos de estos tendrán distintos significados. El aspecto del módulo de la FPGA es el siguiente:

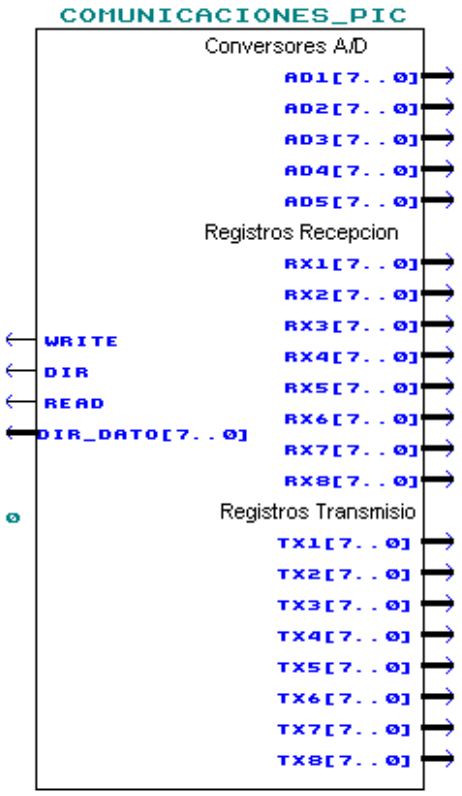


Fig 2.2.15: Módulo de comunicaciones para la FPGA

Como ya se ha comentado el numero de registros es fácilmente configurable. Además los módulos han sido programados como genéricos, por lo que un número inferior de registros a los programados (8 y 13) se puede configurar en todo momento, es decir, si solamente se utilizan 2 registros de entrada y 3 de salida en una aplicación, pueden seleccionarse, evitando que sean implementados el resto y ocupando el módulo un espacio más reducido en la FPGA, no influyendo esto en el módulo implementado en el µC.

2.2.5.3 Módulos de Depuración

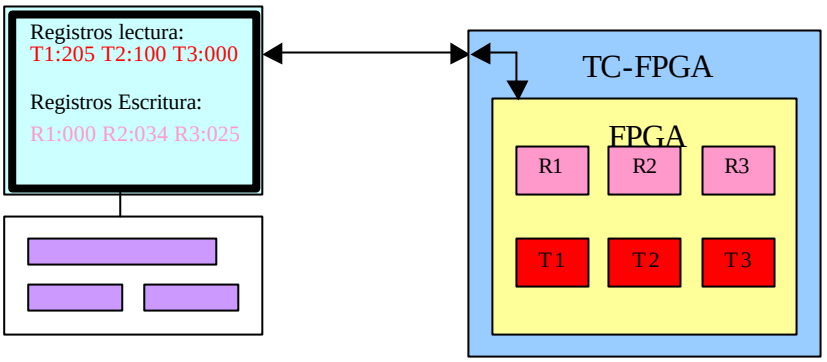
Proporcionan el soporte para observar y modificar variables de la aplicación de usuario para la FPGA en tiempo real. Estos módulos hacen uso de los módulos de comunicación y se implementan en el PC y el µC.

El módulo del µC utiliza las comunicaciones con la FPGA para enviarle datos procedentes del puerto serie, mientras que los datos procedentes de la FPGA los envía a través del puerto serie, todo ello con formato. Al mismo tiempo continua sirviendo los datos de los conversores AD a la FPGA. Además el módulo de depuración devuelve el control al de gestión cuando recibe el comando *escape*[1] por el puerto serie.

[1] Escape, es un comando de control del código ASCII, su valor en hexadecimal es 0x18

Por otro lado el módulo de depuración del PC se limita a recoger los datos del puerto serie y mostrarlos por pantalla de un modo inteligible, al tiempo que envía los datos que son introducidos por el usuario en los registros de transmisión hacia la placa.

El resultado es un enlace entre el usuario y los registros que él desee de la aplicación de la FPGA:



Para realizar una depuración de la aplicación, solamente es necesario que el usuario incluya en su aplicación el módulo de comunicaciones y una los registros de entrada y salida con los valores que desee modificar y observar respectivamente, cuando se encuentre en la depuración. Esta se activará como ya hemos indicado desde el módulo de gestión en modo programación.

Por último cabe destacar que por la arquitectura del µC usado ha sido necesario implementar el módulo de depuración dentro del espacio de las aplicaciones de usuario para este. Por tanto cuando se entra en el modo depuración se pierde la aplicación de usuario cargada en el µC, si hubiese alguna.

2.2.5.4 Módulos de Aplicación

Estas son las aplicaciones que se realizarán para el sistema, tanto para la FPGA como para el µC. Para crearlas puede utilizarse cualquier herramienta de estos dispositivos. En este proyecto se han utilizado las siguientes:

- µC:
  - MPLAB V 5.2 De Microchip. Inc. ®
  - Compilador CCS. Compilador C ANSI de CCS®
- FPGA:
  - MAX+PLUS II V.9.23 Baseline: de ALTERA®

Todas las herramientas salvo el compilador C son de libre distribución.

2.3 Tarjeta de sensores y actuadores

2.3.1 Objetivo de la tarjeta

Esta tarjeta es el primer módulo de ampliación de la tarjeta de control basada en la FPGA. Basándose en el formato de placa y sistema de conexiones implementado en la tarjeta de control PICC-FPGA, se trata de controlar el mayor número de sensores y/o actuadores que se puedan conectar al sistema, de los cuales no se conoce ni la cantidad, ni el tipo a priori. Por lo tanto, se pretende dar soporte al exterior para conectar diferentes tipos de sensores y actuadores de forma que la electrónica adicional sea mínima o incluso nula.

2.3.2 Bloques

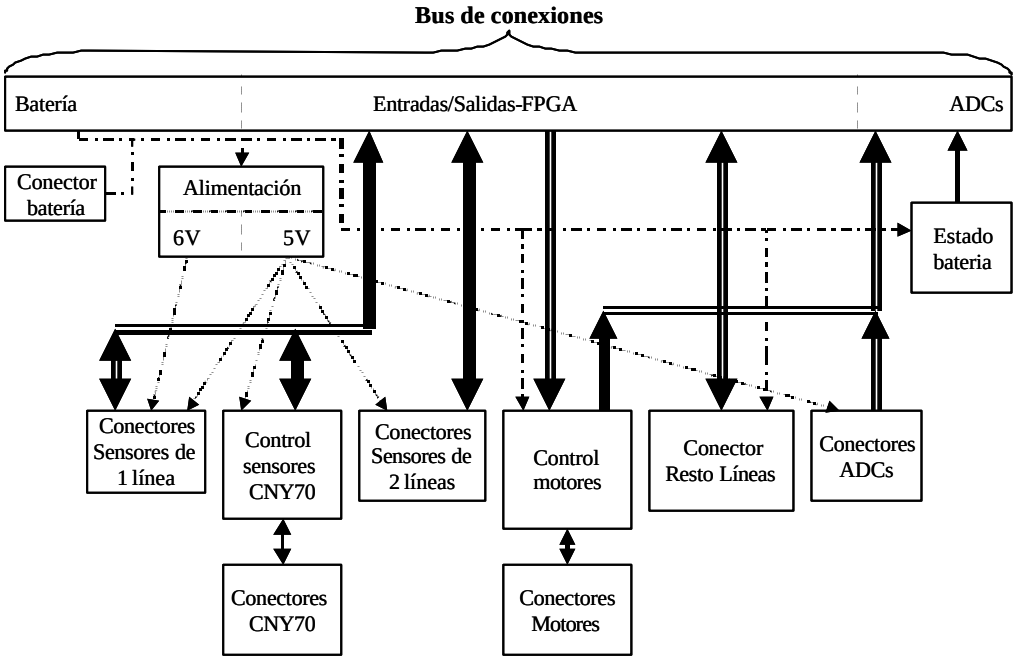


Fig 2.3.1: Diagrama de bloques de TSA-FPGA

2.3.3 Diseño a nivel de bloques

2.3.3.1 Modulo de Alimentación

Para alimentar la placa se parte de que el bus de conexiones proporciona la alimentación directa de la Batería.

Como la mayoría de sensores se alimentan directamente con 5V, y los actuadores tipo servos utilizan una alimentación entre 5 y 6V, se ha optado por dividir la alimentación en 2 partes:

- Alimentación de sensores de 1 línea, pensando que se pueden poner servomecanismos[1]. Si se colocan varios servos es recomendable que esta alimentación sea a 6V para aprovechar al máximo la potencia. En caso de no utilizar servos, se usará de 5V.
- Resto de electrónica, resistencias pull-up, comparadores... Para lo cual se utiliza 5V.

El diseño se ha realizado para una en batería entre 11V y 15V (el máximo es por el diseño del Control de estado de batería), para lo cual se le ha colocado un 7809 (por motivos de disipación) que da soporte al resto de reguladores de tensión. En caso de que la tensión de alimentación fuese entre 8V y 11V, el 7809 se debe cortocircuitar para alimentar directamente los reguladores que proporcionan el voltaje de salida, los cuales son por un lado un 7806 usado para los sensores de 1 línea que se puede cambiar por un 7805 según nos convenga y por otro un 7805 que alimenta el resto de la electrónica.

En cuanto al encapsulado elegido es en SMT[2], tipo D2PAK[3]. Por necesidad de espacio en el PCB, no se ha colocado tipo TO220. La diferencia entre los 2 encapsulados corresponde a la capacidad de disipación térmica de uno y otro.

A pesar de tener la alimentación de la batería por medio de los conectores del bus, se ha colocado un conector de alimentación de batería, ya que esta placa es la va a producir más consumo a priori en el sistema. Esto se hace para no sobrecargar los conectores del bus.

Aparte de los reguladores de tensión y el conector, se han colocado condensadores para eliminar ruido: Uno de 100uF a la batería, uno de 1uF a la salida de cada regulador, y varios de 100nF distribuidos por los conectores que alimentan sensores.

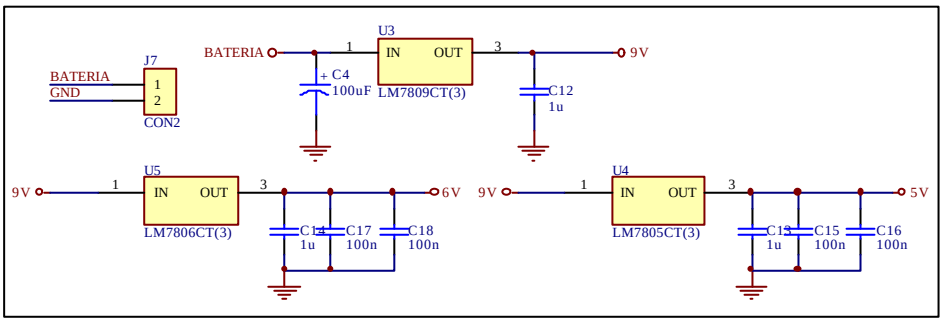


Fig 2.3.2: Esquema correspondiente a las alimentaciones de la placa.

2.3.3.2 Módulo Control de motores

Este bloque es uno de los más importantes de la placa, porque hay que decantarse por la electrónica que va a controlar una cantidad X de motores, los cuales no se sabe a priori la potencia que necesitan.

[1] Usados en modelismo, se controla la posición por ancho de pulso.
[2] Tecnología de montaje superficial.
[3] Encapsulado similar al teadicional TO-220, pero de montaje superficial.

Se parte de que el número mínimo de motores en un robot de propósito general es 2 para una dirección de tipo diferencial. Ahora, si se piensa en futuras ampliaciones, no vendría mal poder controlar un tercer motor para un actuador que se le coloque al robot como por ejemplo una pala o una pinza.

Sobre las características del motor no se conoce nada, pero partiendo de la base de los motores que se van a colocar en el robot (servos trucados), cuya resistencia interna es de 7 Ω, se puede calcular que en el peor de los casos, el puente ha de soportar una corriente media de 1,7A (si el motor está bloqueado) para alimentación de 12V, y el doble de corriente de pico (cambiando bruscamente la dirección).

Ya tenemos unas características mínimas que ha de soportar el puente para controlar servos trucados y con sobrevoltaje a 12V:

- 3 motores
- 3,4 Amperios de Pico.
- 1,7 Amperios DC.
- 12 Voltios de alimentación.

Partiendo de estas características, si se eligen unos motores de calidad, proporcionaran al robot más potencia mecánica con un consumo no mucho mayor.

Otra desventaja es que el circuito de control no puede ocupar mucho espacio en PCB, y no tenga mucha altura para poder colocar otra placa encima.

Al final, la opción elegida ha sido colocar 2 circuitos integrados que contienen cada uno 3 medios puentes, de forma que ocupa menos espacio, y la electrónica adicional es mínima. Los circuitos elegidos son: L6234 [11], los cuales están pensados para controlar motores sin escobillas. En este nuestro caso se aprovechan estas características para controlar con cada L6234 un motor, y utilizando el medio puente sobrante de cada CI, se puede controlar un tercer motor. La ventaja principal es el poco espacio que ocupa en comparación con otros circuitos para controlar motores de características similares.

Características más relevantes de los CI L6234:

- Tres medios puente controlables independientemente.
- Combina tecnología DMOS y CMOS con circuitería bipolar.
- Encapsulado POWER DIP20
- Voltaje de alimentación desde 7V a 52V
- Corriente de pico soportada de 5 A.
- Rdson de 0,3Ω
- Frecuencia de conmutación hasta 50KHz
- Compatible con circuitería TTL
- Corriente DC: 2,4 A.

La electrónica adicional recomendada por el fabricante para conectar al puente se trata únicamente de componentes discretos, y se utilizan para filtrado y el "*charge bump*" de las puertas de los transistores del puente.

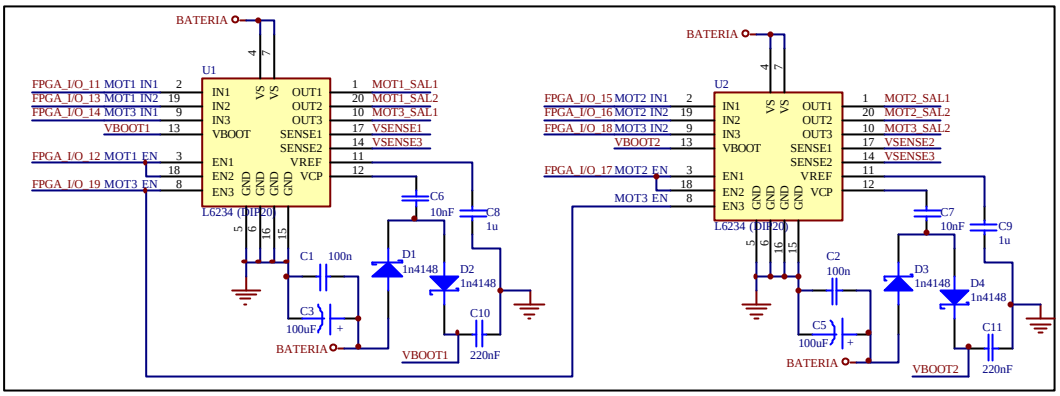


Fig 2.3.3: Circuito de control de motores

El CI da la posibilidad de conectar resistencias de sensado[1] para poder conocer la corriente que consume el motor conectado en cuestión, cosa que se ha aprovechado colocando resistencias de potencia dando las posibilidades de:

- Realizar control de torque.
- Limitar la corriente por el motor.
- Otra forma de conocer si el robot se ha chocado.

El circuito[2] montado consta de una resistencia de potencia (2W) colocada en serie con cada puente a masa, y un filtro que se puede conectar por medio de un JUMPER a un ADC para tomar medidas.

Cada uno de los motores tiene un conector independiente, en cuyos terminales se ha colocado un condensador como ayuda de filtrado del PWM.

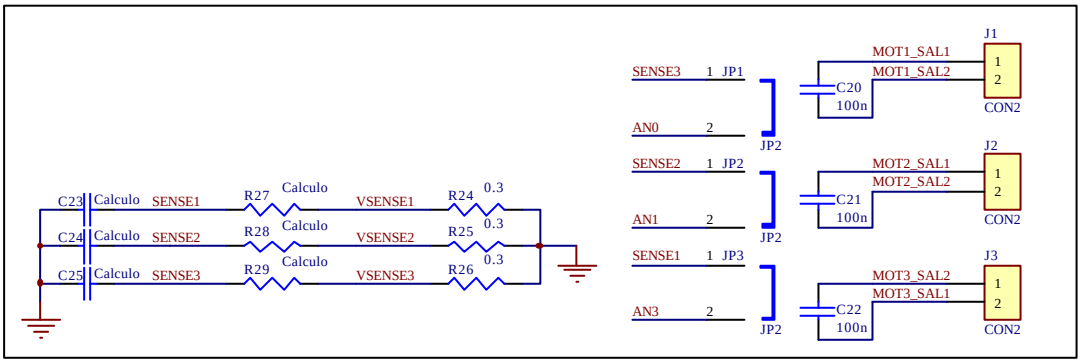


Fig 2.3.4: Sensado de corriente de motores, y conectores.

2.3.3.3 Controlador sensores fotoreflectores[3]

Estos sensores son los utilizados típicamente para diferenciar el color del suelo. Constan de un emisor de IR y un fototransistor, que se polarizan mediante las resistencias adecuadas. Para el control de la señal recibida, se suele utilizar una

[1] Para más información sobre su funcionamiento, vease apartado 2.4.1.1
[2] Ver figura 2.3.4
[3] Por ejemplo el comúnmente utilizado CNY70

puerta del tipo trigger-smith, y así la información recibida es suelo claro o suelo oscuro. Pues bien, en el diseño de la placa se ha optado por colocar un comparador en lugar de la puerta trigger-smith para poder regular mediante un potenciómetro[1] el nivel de claro y oscuro.

Como circuito comparador se ha elegido el CI LM339, el cual integra 4 circuitos comparadores, y la salida es en colector abierto.

El resto de la electrónica son resistencias de polarización del sensor, y resistencias de pull-up para las salidas de los comparadores.

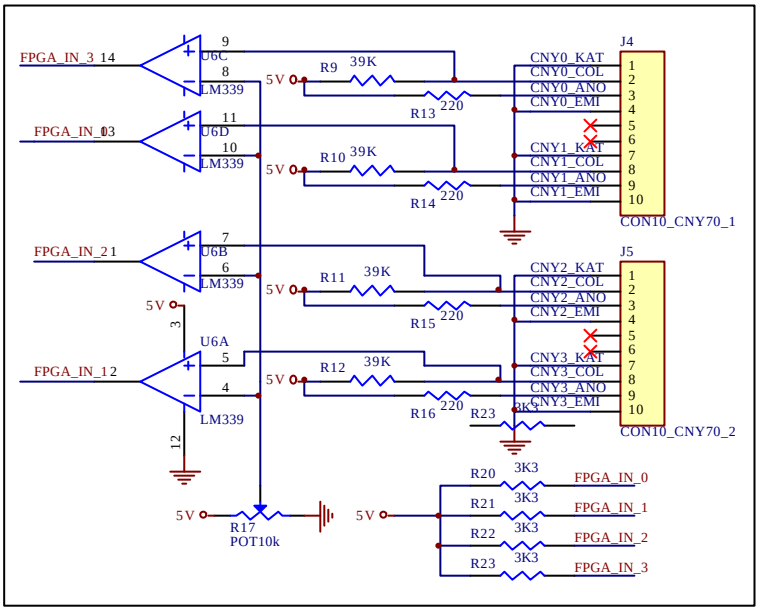


Fig 2.3.5: Esquema del circuito de control de sensores tipo CNY70.

Como se han colocado 4 sensores, se han agrupado de 2 en 2 a la hora de conectarlos para que el montaje sea sencillo. Se toma un cable de bus de 10 hilos con sus conectores de 5x2 a los extremos, se enchufa un extremo en la placa y luego los sensores en el extremo sobrante.

Esta parte está pensada tan solo para sensores de este tipo y el sentido de la información es únicamente del sensor al robot, para lo cual se ha aprovechado las 4 entradas dedicadas de la FPGA para realizar las conexiones.

[1] Corresponde a R17 en la tarjeta.

2.3.3.4 Control de estado de batería[1]

Se trata de un sensor montado en la tarjeta, que da información al sistema del estado de carga de la batería. Puesto que no se conocen las características de la batería, se ha elegido montar un detector de pico con un divisor resistivo para conectarlo a un convertidor Analógico-digital, y que el sistema sepa de manera aproximada la tensión en cada momento de la batería. La gestión para determinar el nivel de carga se realizará mediante software en función de la batería colocada.

La línea se conecta a un ADC del PIC, más concretamente a AN2.

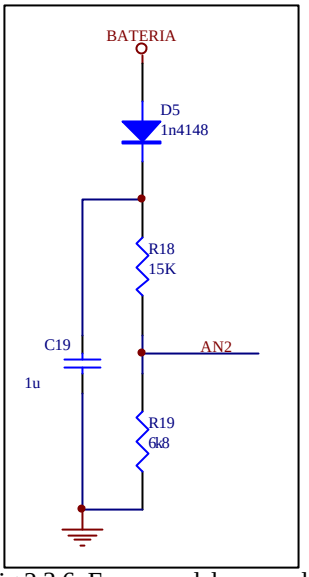


Fig 2.3.6: Esquema del sensor de nivel de batería

2.3.3.5 ADCs

El sistema TC-FPGA dispone de 5 canales analógico-digital, que se pueden usar para conectar sensores analógicos.

La línea **AN2** se ha conectado al sensor de carga de batería.

El resto de líneas se encuentran disponibles desde el exterior por medio de conectores de 3 pines para poder colocar la mayoría de los sensores que ofrecen datos de forma analógica.

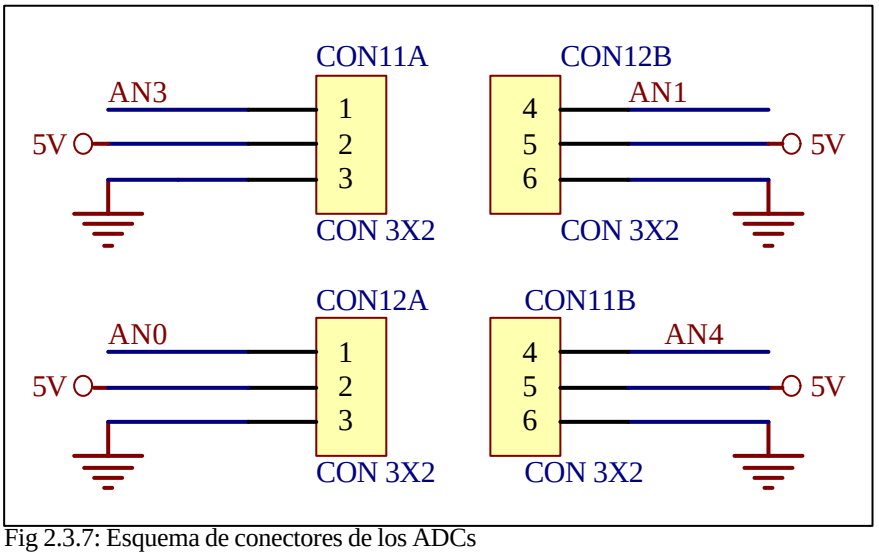


Fig 2.3.7: Esquema de conectores de los ADCs

[1] Para más información sobre su funcionamiento, vease apartado 2.4.1.2

Las líneas **AN3**, **AN1** y **AN0**, también se pueden conectar al sensado de corriente de los motores 1, 2 y 3 respectivamente.

Los conectores accesibles para los canales analógicos son de 3 pines, en los que se da un punto de masa, una alimentación de 5V, y la señal. Esta filosofía de conexiones es para enchufar el sensor directamente a la placa, sin más.

2.3.3.6 Conectores para sensores de 1 línea

Estos conectores tienen el objetivo de controlar sensores o actuadores que precisan de tan solo una línea de datos. Llevan alimentación independiente de 5 o 6V, según el regulador que se coloque para ello, como es ha explicado en el apartado 2.3.2.1.

El conector de 3 pines que proporciona 1 línea de masa, otra de alimentación, y la tercera es la señal que con una resistencia de pull-up va conectada directamente a la FPGA.

"**Advertencia**": si la alimentación de este conector es de 6V, no se deben conectar sensores que puedan cortocircuitar la alimentación con la línea de señal, pues se corre el peligro de dañar seriamente la FPGA. En cualquier caso siempre se puede cortocircuitar la línea de señal con masa.

Estos conectores están pensados para conectar Servos, BUMPERS (sensores de contacto), pulsadores, leds...

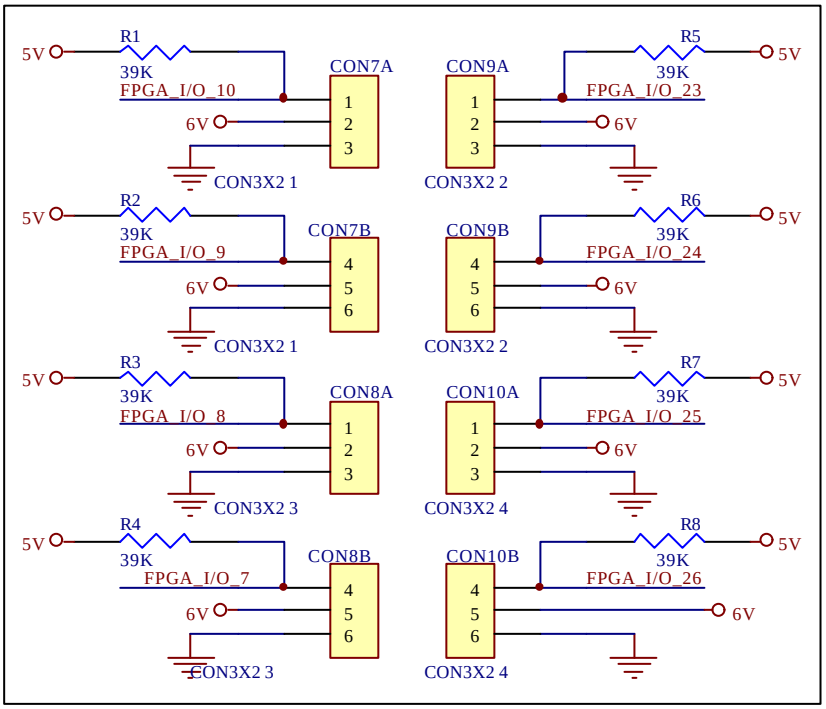


Fig 2.3.8: Esquema de conectores para sensores de 1 línea.

2.3.3.7 Conectores para sensores de 2 líneas

Estos conectores tienen el objetivo de controlar sensores o actuadores que precisan de 2 líneas.

Los conectores son de 4 pines, los cuales son: señal de masa, alimentación de 5V, y las 2 señales.

Aquí se pueden colocar sensores en los cuales el trasiego de información se hace en los sentidos sensor a controlador y controlador a sensor, como por ejemplo Gp2d02, sonar SFR04, etc. También se puede utilizar para conectar 2 sensores que requieran de una sola línea de datos.

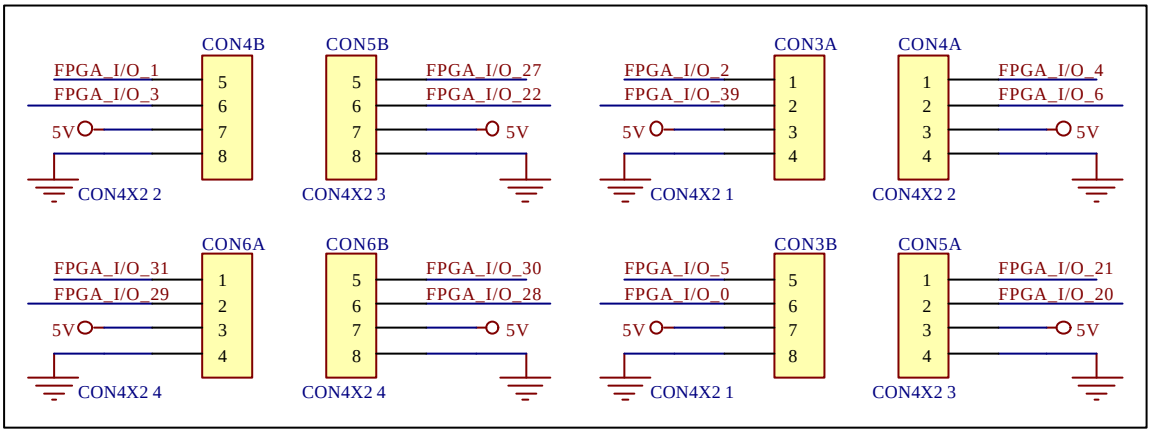


Fig 2.3.9: Esquema de conectores para sensores de 2 líneas.

2.3.3.8 Resto líneas de entrada / salida de FPGA

Ya que no se han podido ofrecer al exterior todas las líneas de la FPGA por medio de los conectores anteriormente descritos, se ha colocado un conector de 10 hilos para dar accesibilidad a las restantes líneas de datos, de modo que se pueda colocar cualquier circuito con tan solo un cable de bus.

Se trata de un conector de 5x2, el cual incluye alimentación directa desde la batería, dos puntos de masa, y 6 líneas de datos.

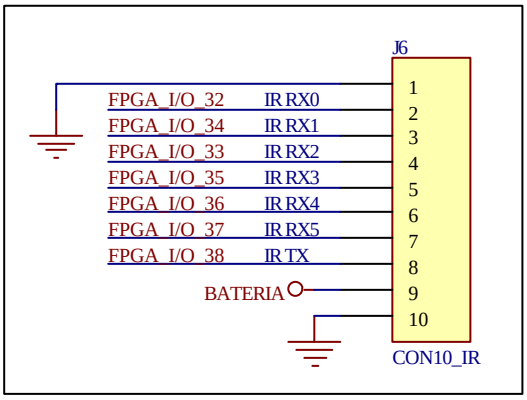


Fig 2.3.10: Conexiones de resto de líneas

2.3.4 Diseño del PCB

En este apartado se explican las características de diseño de la placa de circuito impreso del prototipo, así como los conectores elegidos y el encapsulado de los componentes. El mayor problema planteado en el diseño es colocar tantos componentes en la tarjeta para cumplir el formato de tamaño de placa y colocación de conectores de bus del sistema definido en la TC-FPGA.

2.3.4.1 Conectores

Los conectores son la parte más importante de esta tarjeta, cuyas características deben ser: ocupar poco espacio, facilidad de montar, facilidad de conectar, accesibilidad, y robustez.

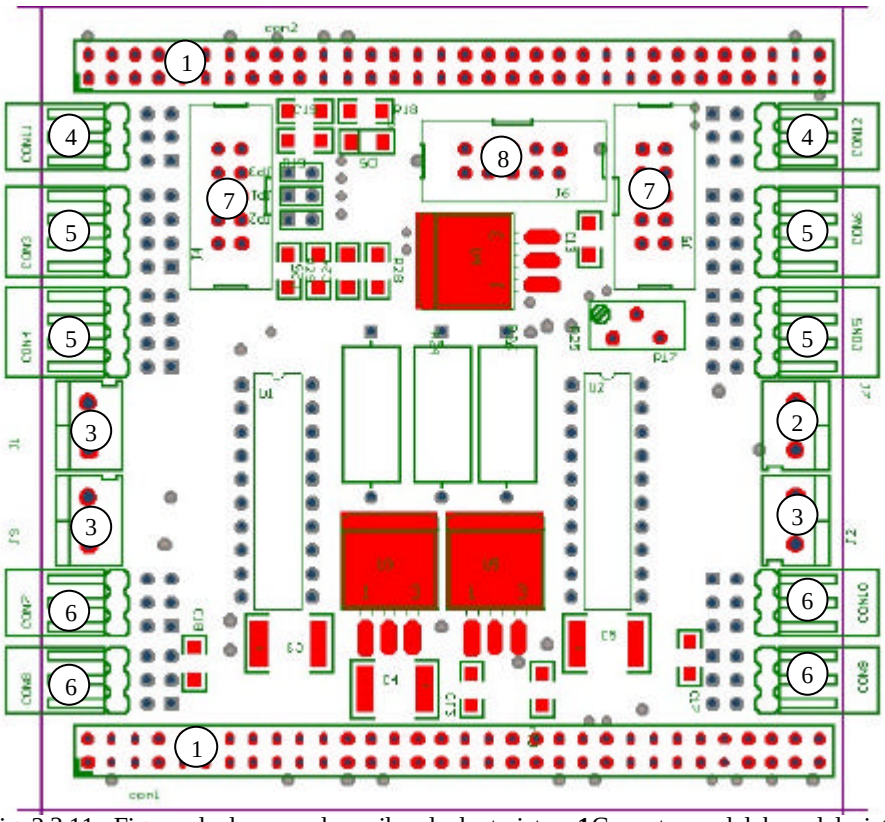


Fig 2.3.11: Figura de la cara de arriba de la tarjeta. **1**Conectores del bus del sistema. **2**Conector de batería. **3**Conectores para motores. **4**Conectores de los ADCs. **5**Conectores para sensores de 2 líneas. **6**Conectores para sensores de 1 línea. **7**Conectores para los CNY70. **8**Conector para resto de líneas.

1 Los conectores del bus del sistema utilizados son como los de **TC-FPGA** con la diferencia de que los pines son largos para pinchar en la placa de abajo. Además sirve como sujeción mecánica entre tarjetas.

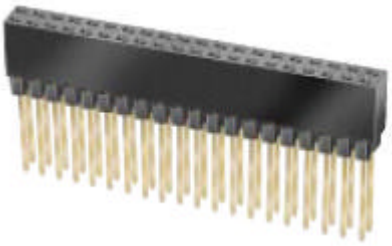


Fig 2.3.13: Conector del bus del sistema

2 y **3** Los conectores dedicados a alimentación y motores han de ser los más robustos en cuanto a densidad de corriente pueden aguantar, por lo tanto se han colocado clemas de 2 pines para circuito impreso con paso 5,08mm.

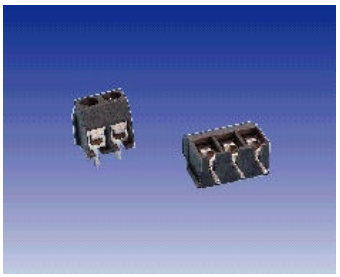


Fig 2.3.14: Imagen de conectores de alimentación y motores.

4, **5** y **6** Para conectar sensores se necesitan muchas líneas de datos, pero la corriente demandada es pequeña. El formato elegido son tiras de pines dobles, cuya parte de arriba forma un conector, y la de abajo otro. Hay conectores de 3 pines x 2 líneas y conectores de 4 pines x 2 líneas.

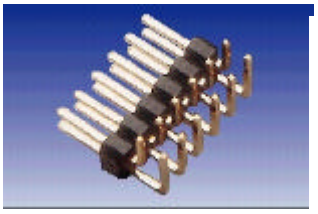
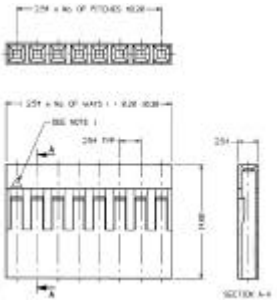


Fig 2.3.14: Conectores elegidos. A la izquierda el de circuito impreso. A la derecha el que va al cable del sensor.



El conector que se coloca al cable es como los de la Figura de la derecha, pero con 3 o 4 pines según caso.

7 y **8** Se tratan de conectores para cable plano. Tienen la ventaja de la facilidad de montar el cable, y que transportan muchas señales ocupando poco espacio. En este caso se utiliza bus de 10 hilos (5x2).

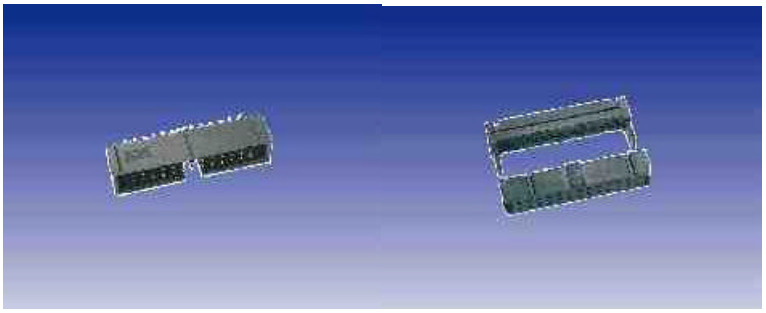


Fig 2.3.15: Conectores de cable plano. A la izquierda el de circuito impreso. A la derecha el que va al cable del sensor.

2.3.4.2 Componentes

La elección de encapsulado para los componentes del sistema se ha realizado pensando en: sencillez de soldadura, compactación, y facilidad de obtención.

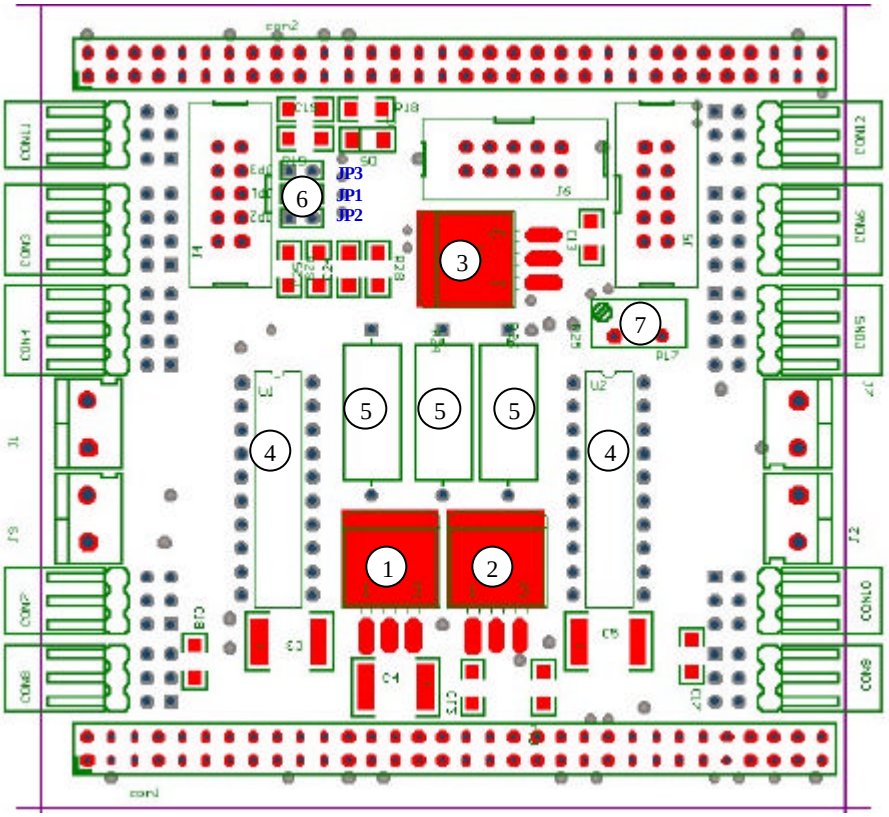


Fig 2.3.16: Figura de la cara de arriba de la tarjeta. **1**Regulador 7809. **2**Regulador 7806 o 7805. **3**Regulador 7805. **4**CI L6234. **5**Resistencias de sensado. **6**Jumpers. **7**Potenciometro multivuelta para sensores CNY70.

Respecto a los encapsulados de los componentes elegidos son de uso genérico y ampliamente conocidos por diseñadores.

Encapsulados de los componentes:

- Resistencias de potencia para sensado de motores. Inserción → RES700
- Resto de resistencias. SMT → 1206.
- Diodos. SMT → MINIMELF
- Condensadores electrolíticos 100uF. SMT → TANTAL-D
- Condensadores electrolíticos 1uF. SMT → TANTAL-A
- Resto de condensadores. SMT → 1206.
- Reguladores 78xx. SMT → D2PAK.
- CI L6234. Inserción → DIP-20.
- CI LM339. SMT → SO-14.
- Potenciómetro multivuelta de ajuste superior. Inserción.

2.3.4.3 Diseño de pistas

A la hora de realizar el diseño de pistas hay que tener en cuenta cual es la tecnología de fabricación disponible. En este caso la fabricación del PCB se ha realizado con sistemas artesanales con los inconvenientes que ello conlleva:

- No existe taladro metalizado. Por lo tanto por donde le llegue la pista al pad es por donde hay que soldarlo. Es preferible diseñar las pistas para que la soldadura sea siempre por abajo para los componentes de inserción. En este caso no ha sido posible con todos los componentes, teniendo que soldar por arriba y abajo los conectores del tipo de tira de pines acodado, y los CI L6234.
- Las vías hay que hacerlas por medio de un cable, lo que implica que es mejor que no estén bajo componentes para que no queden mal al soldarlos.
- No se dispone capa de antisolder, lo que supone una desventaja a la hora de soldar los componentes ya que es muy fácil producir cortocircuitos.
- Tampoco hay capa de serigrafiado empeorando el acabado y dificultando la colocación los componentes para montaje.
- Respecto a las clases (distancia pad-pista, pad-pad...) por método prueba error se ha llegado a la conclusión que se puede llegar de esta forma a clase 3 según la tabla de clases de la empresa DISELEC-ELECTRONICA, menos en el diámetro de taladro, que será 0,8mm = 31mil

CLASE	ANCHO PISTAS	SEPARACION PISTA/PISTA	SEPARACION PISTA/PAD	SEPARACION PAD/ PAD	DIAMETRO DE TALADRO
3	0,30 mm=12 MIL	0,30 mm=12 MIL	0,30 mm=12 MIL	0,30 mm=12 MIL	0,50 mm=20 MIL
4	0,20 mm=8 MIL	0,20 mm=8 MIL	0,20 mm=8 MIL	0,20 mm=8 MIL	0,30 mm=12 MIL
5	0,15 mm=6 MIL	0,15 mm=6 MIL	0,15 mm=6 MIL	0,15 mm=6 MIL	0,30 mm=12 MIL
6	0,12 mm=5 MIL	0,12 mm=5 MIL	0,12 mm=5 MIL	0,12 mm=5 MIL	0,25 mm=10 MIL

Fig 2.3.17: Tabla de Clases para fabricación de PCBs

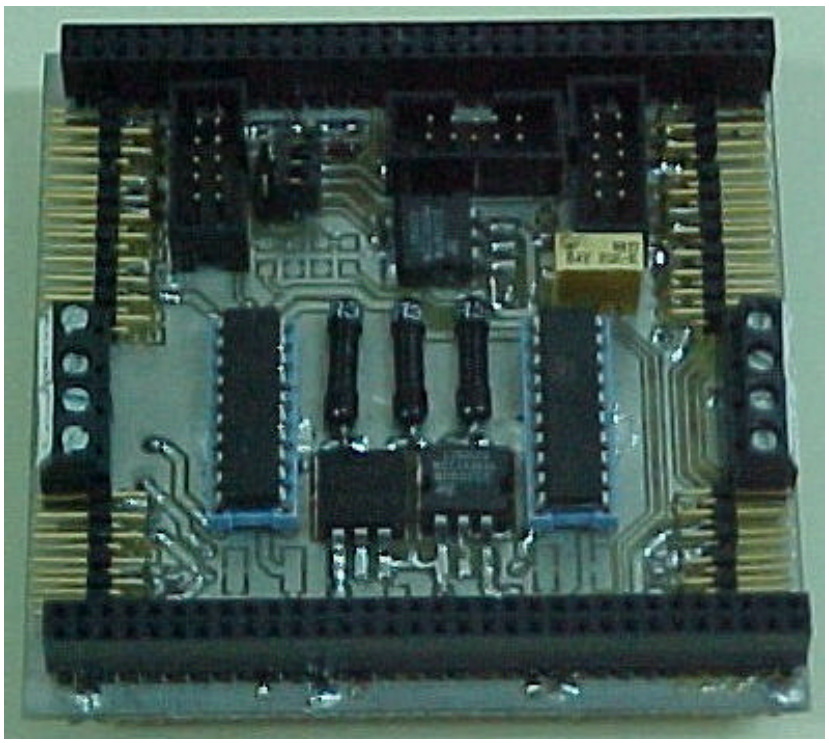


Fig 2.3.18: Foto de TSA-FPGA. Arriba.

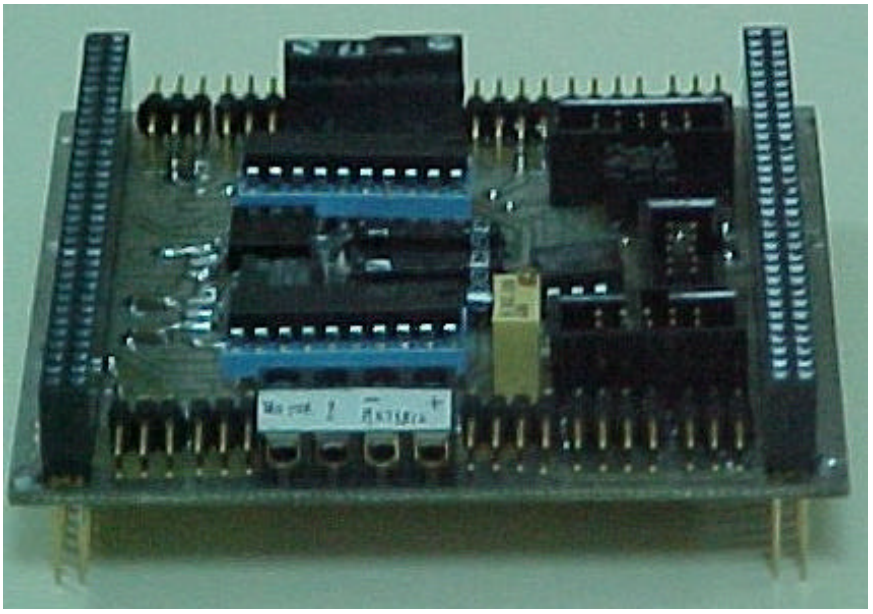


Fig 2.3.19: Foto de TSA -FPGA. Lado derecho.

En cuanto al diseño de pistas, se ha tenido en especial cuidado en todas las pistas de alimentaciones, canales analógicos, y motores, realizando un trazado más directo y aplicando un ancho de pista considerablemente más ancho. Respecto a las pistas de datos, se han diseñado con un ancho mínimo (12mils) procurando evitar ángulos rectos y que la longitud sea mínima. Para la pista de masa se ha colocado un plano en la capa de arriba para disminuir ruidos, que además utilizan los componentes de potencia a modo de disipador.

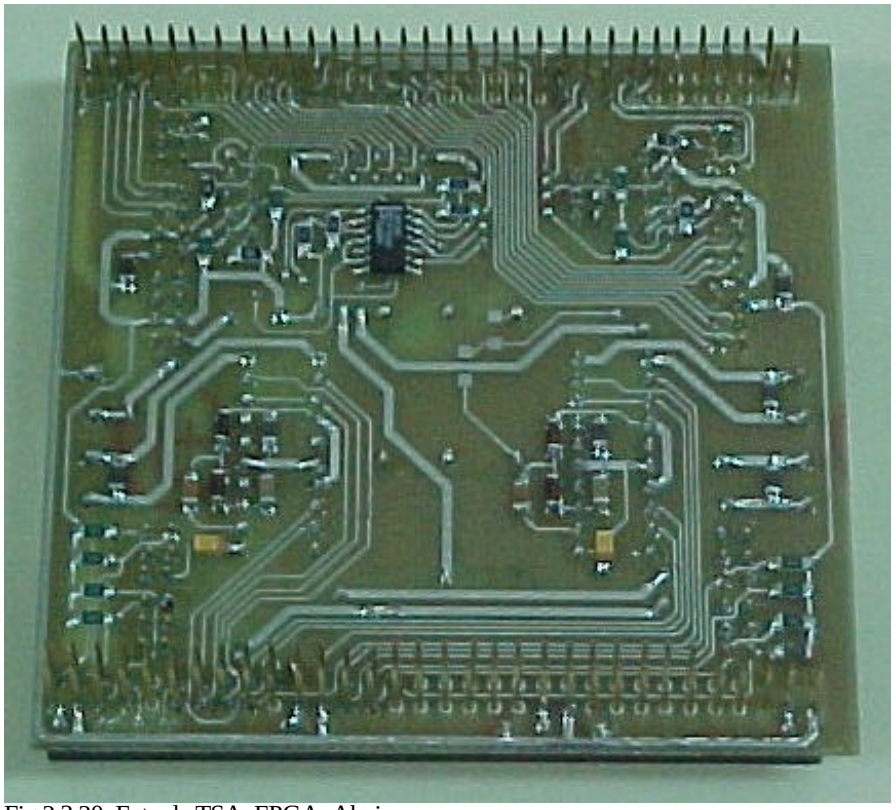


Fig 2.3.20: Foto de TSA -FPGA. Abajo.

2.3.5 Diseño versión fabricación

Una vez montado el prototipo y comprobado el correcto funcionamiento de todos sus circuitos, se ha realizado pequeñas mejoras para una posible fabricación en serie.

2.3.5.1 Mejora circuito motores

Se mejora el circuito de control de motores, colocando resistencias de pull-down en las líneas de enable correspondientes a cada uno de los 3 puentes, con el objetivo de que no se ponga en marcha un motor accidentalmente por ruidos si no se ha configurado la FPGA.

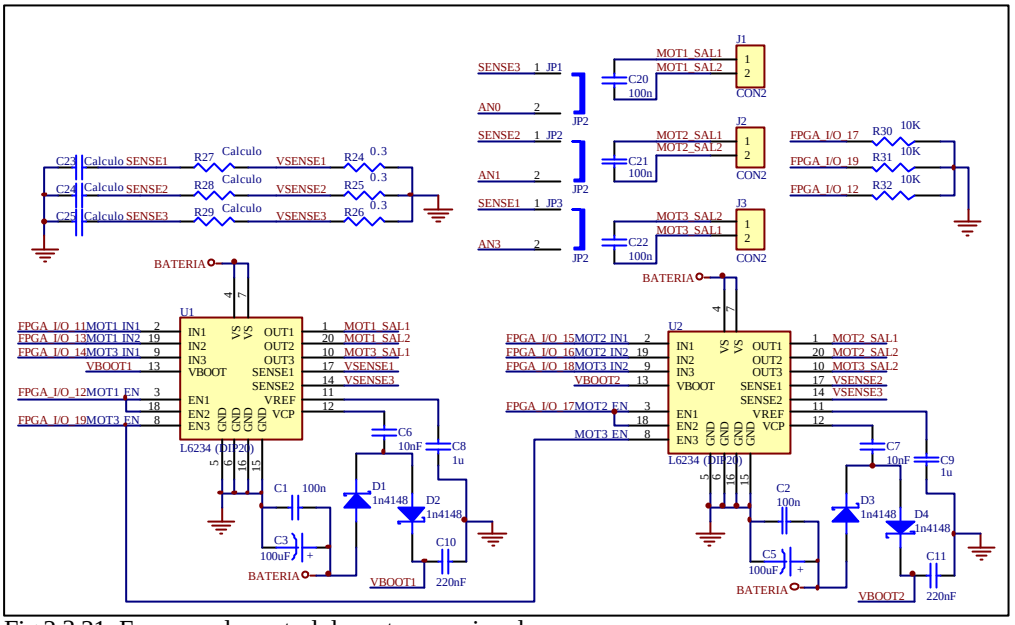


Fig 2.3.21: Esquema de control de motores mejorado.

2.3.5.2 Mejora conectores ADCs

Como mejora para los conectores correspondientes a los ADCs, se deja hueco para colocar resistencias de pull-up en las entradas correspondientes, que se soldaran según necesidad de la aplicación. Ejemplo de aplicación:

- Se quiere colocar un sensor de temperatura tipo NTC. Un montaje típico es una resistencia de polarización entre Vcc y señal, y la NTC entre señal y GND. La ventaja será que se colocando la resistencia de polarización en el PCB, tan solo hay que conectar la NTC por medio de 2 cables a la tarjeta.

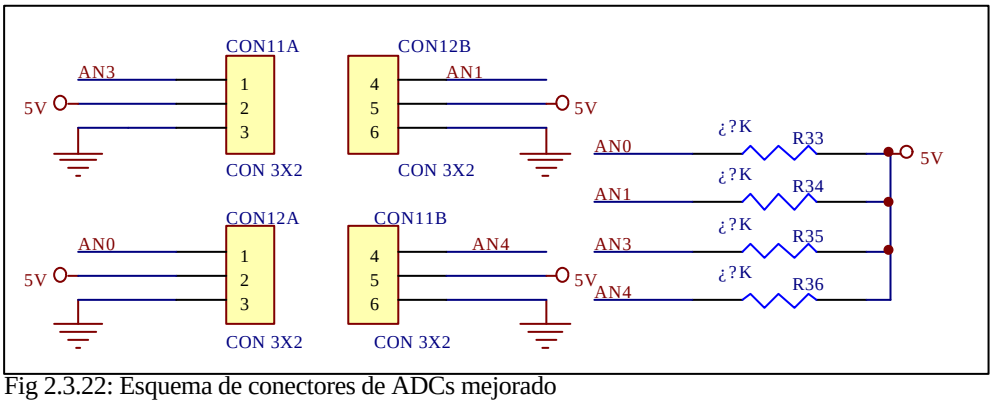


Fig 2.3.22: Esquema de conectores de ADCs mejorado

2.3.5.3 Características del PCB

Como la fabricación se realizará en una empresa profesional, se parte con las siguientes ventajas con respecto a la fabricación tradicional:

- Se utiliza taladro metalizado, por lo tanto los componentes de inserción sólo se sueldan por 1 cara independientemente de donde esté la pista.
- Las vías vienen hechas y pueden ser más pequeñas, con la ventaja de poder estar debajo de un componente, o incluso en un pad.
- Hay que colocar bien la capa de serigrafiado para que se vea bien que es cada cosa. Es importante en los componentes para soldarlos, y mucho más en los conectores para no poner mal los sensores, motores...
- Para mejorar el diseño, se ha elegido clase 4 respecto a la tabla de fig 2.3.17.

El PCB para fabricación se ha mejorado suprimiendo vías, colocando pistas de alimentación mas gruesas, y la línea de batería se le ha colocado un plano de batería alrededor de la placa para que el ancho de pista sea lo mayor posible.
También se ha repasado exhaustivamente el tamaño de los agujeros para no cometer el error de fabricarlo y luego los componentes de inserción no entren.

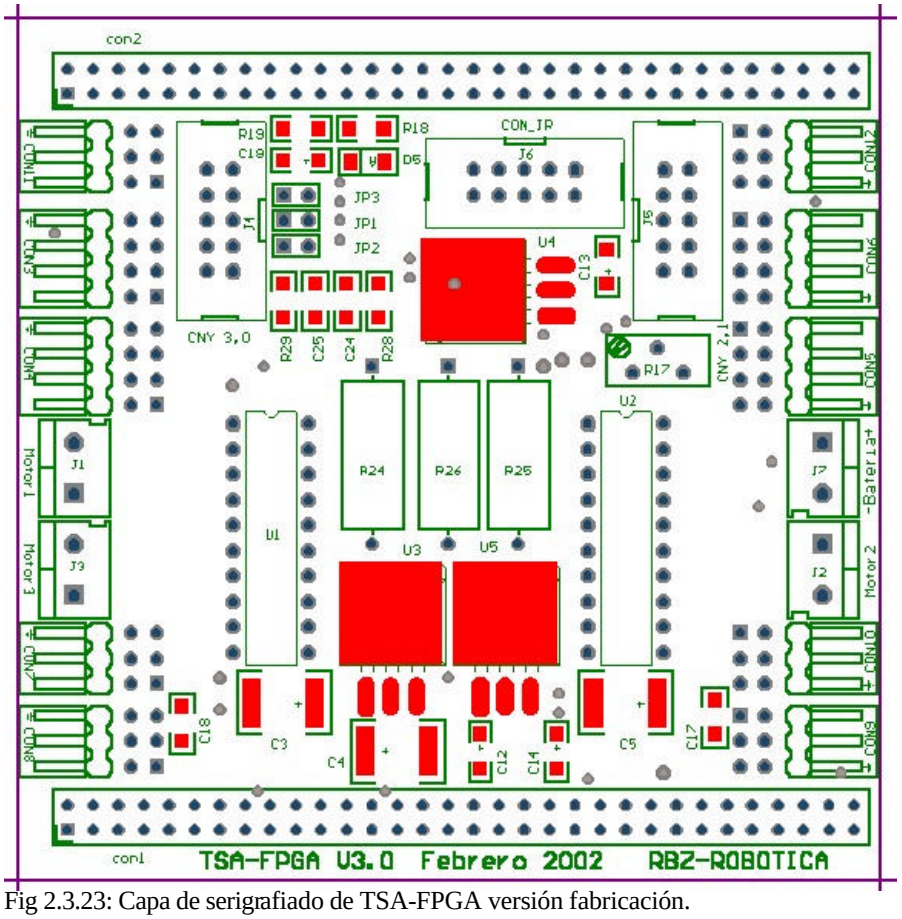


Fig 2.3.23: Capa de serigrafiado de TSA-FPGA versión fabricación.

2.4 Sensores, actuadores y módulos de control

Se explicará una serie de conceptos teóricos sobre sensores y actuadores muy útiles en el campo de la robótica móvil, así como soporte para uso de algunos modelos concretos.

2.4.1 Sensores propioceptivos

Le dan información al cerebro del robot del estado de alguna de sus partes.

2.4.1.1 Sensado de corriente por motores

El objetivo medir la corriente que circula por el motor correspondiente para dar posibilidad al programador de:

- Realizar un control de torque controlando que la corriente por el motor sea constante.
- Detectar bloqueo del motor por consumo mayor o que le falta carga por un consumo menor.
- Además puede usarse para controlar motores cuyas características de consumo medio o de pico no fuese soportado por el puente en H. Si el consumo llega a un umbral, se reduce la tensión de alimentación del motor (en este caso el PWM) para que consuma menos. Se trata en definitiva un uso más suavizado de los motores.

Para realizar la medición de corriente a través del motor se ha colocado una resistencia de sensado, y la tensión en esta es tomada mediante el ADC. Sin embargo, puesto que los motores se controlan en velocidad mediante PWM, la corriente que circule por el motor tendrá la misma forma de onda que el PWM. Por este motivo se ha colocado un filtro paso bajo pasivo de primer orden, tal y como muestra la figura 2.4.1. En definitiva lo que se medirá es la corriente media[1] que circula por el motor. A continuación se muestra el cálculo de los elementos del filtro.

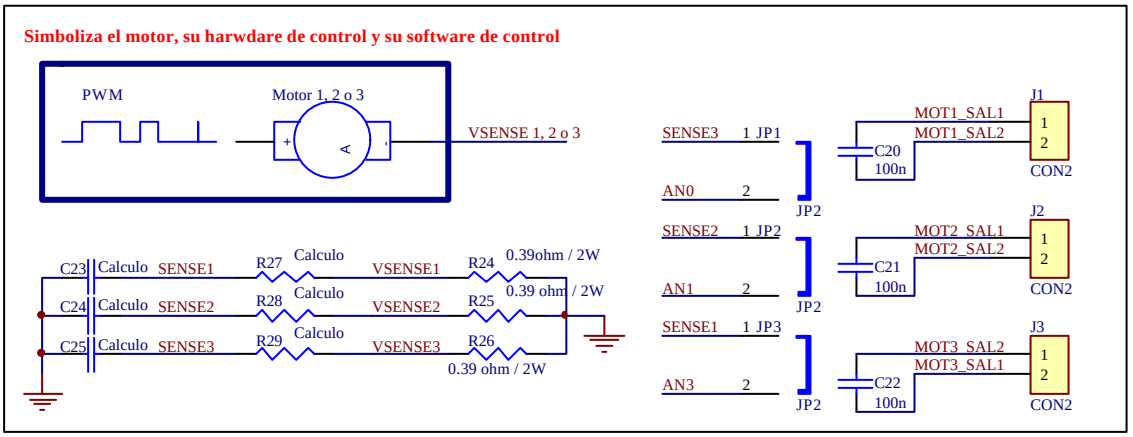


Fig 2.4.1: Esquema del sensado de corriente por los motores.

[1] Si lo que se quiere medir son los picos de corriente el filtro no será necesario.

Este tiene relación con la frecuencia del PWM que se aplica a los motores, la cual se tomará de 7812,5Hz[1], ya que esta es la frecuencia de los módulos software para control de motores implementa. Puesto que se desea medir la corriente media, se precisa de un filtro paso bajo de una frecuencia de corte cercana a 0Hz, ya que de este modo se obtendría la señal continua pura. Sin embargo tampoco se debe escoger una frecuencia demasiado baja, ya que la estabilización de la señal sería demasiado lenta. Se ha elegido una frecuencia de corte de un orden 100 veces inferior a la frecuencia de modulación. De este modo:

$$f_C = \frac{1}{2pRC} \Rightarrow RC = \frac{1}{2pf_C} = \frac{1}{2p78{,}125Hz} = 2.037.10^{-3}$$

Si se toma un condensador de 220nF se obtiene:

C = 220 nF R= 9,25K

Con estos valores se consigue una constante de tiempo para el filtro de:

$$t = RC = 2.037ms \Rightarrow 5 \cdot t = 10{,}185ms$$

Lo que quiere decir que pasarán unos 10ms desde que cambia la corriente a otro valor hasta que se lee el valor verdadero. Por tanto no se puede provocar cambios en el motor a razón de 98Hz o más, pero mecánicamente el motor no tiene capacidad suficiente para soportar estos cambios, y por tanto nunca se dará este caso.

Para el cálculo de la resistencia de sensado adecuada, hay que tener en cuenta los siguientes límites:

- Corriente máxima de 2,4 A (DC) que aguanta el puente.
- Voltaje máximo en el pin V_{SENSE} para no perder potencia excesiva en la resistencia de sensado[2]. El fabricante del puente L6234 recomienda que no supere 1V.
- Potencia máxima en la resistencia de sensado. Según el límite de corriente y el voltaje máximo es de 2,4W, pero la tarjeta permite montar resistencia de 2W de potencia.

Según los datos anteriores, se podría calcular 3 valores adecuados:

- Resistencia capaz de soportar corriente máxima por el puente y voltaje máximo → 0,41ohm / 2,4W
- Resistencia de 2W capaz de soportar la corriente máxima→ 0,34ohm con voltaje eficaz a máxima corriente→ 0,816V

[1] Frecuencia implementada por el módulo de control de motores por motivos de menor ocupación de células lógicas en la FPGA para PWM de 8 bit.

[2] Se recuerda que la resistencia de sensado está montado en serie con el motor el cual es a priori la parte de mayor consumo del sistema.

- Resistencia de 2W capaz de llegar a 1V → 0,5ohm para corriente máxima→ 2 A.

Por lo tanto la resistencia ideal sería de 0,34ohm, obteniendo una variación de voltaje menor en el rango de corriente soportada por el puente que la recomendada.
Al final se ha colocado resistencia de 0,39ohm / 2W por ser la más cercana de que se ha podido conseguir, no siendo problema siempre que los motores no consuman más que la corriente máxima que se calcula a continuación. Con esta resistencia, la corriente máxima (DC) que soporta el sistema en conjunto "Controlador motor + resistencias de sensado" es de:

$$I_{MAXDC} = \sqrt{\frac{P_{MAX}}{R}} = \sqrt{\frac{2W}{0,39\Omega}} = \sqrt{5,12} = 2,26A$$

$$V_{MAX}\ Para\ I_{MAXDC} \Rightarrow V = I_{MAXDC} \cdot R = 0,881V$$

Para calcular la corriente que pasa por el motor en un momento determinado, tan solo hay que aplicar la ley de ohm al voltaje leído del ADC.

Ejemplo para voltaje de 0,7V: $I_{MOTOR} = \frac{V_{SENSE}}{R_{SENSE}} = \frac{0,7V}{0,39} = 1,79A$

El voltaje que se puede leer en V_{SENSE} varía entre 0 y 0,881V por la resistencia de sensado colocada (0,39Ω) y límite de corriente físico del puente (2,26A). Se ha diseñado de esta forma porque si la resistencia de sensado fuese mayor, la pérdida de energía en esta sería mayor quitándosela al motor.

Por otro lado ha de tenerse en cuenta la cuantificación realizada por el ADC del PIC, es respecto a sus 5V, si el valor de entrada no superará 1V. Si se cuantifica con 8 bit, solo serán efectivos 6 bit.

Cabe destacar que el valor de la corriente que se obtendrá no será exactamente el verdadero, ya que el filtro proporciona el valor medio e introduce una pequeña atenuación (por no ser la impedancia de entrada del ADC infinita).

Por último es importante recordar que para utilizar estos sensores, se ha de conectar el **JUMPER** correspondiente[1] y no colocar nada en el conector del ADC respectivo para evitar medidas erróneas.

2.4.1.2 Nivel de batería

El sensor tiene como objetivo comprobar el estado de carga de la batería. Con esta información, se puede programar para que en un momento determinado avise que le queda poca carga o incluso que funcione en algún modo de bajo consumo como desconectar algún sensor o mover los motores más despacio.

[1] Vease fig 2.4.1.

Se pretende medir la tensión de la batería ya que es proporcional a la carga energética pero:

- El sistema introduce ruidos en el voltaje de alimentación.
- La tensión de batería depende también del consumo debido a su resistencia interna.

El montaje decidido fue un detector de pico con un filtro para medir mediante un ADC el voltaje más aproximado al voltaje de batería sin carga.

$$f_C = \frac{1}{2pRC} = 7{,}3Hz$$

Con estos valores se consigue una constante de tiempo para el filtro de:

$$t = RC = 21{,}8ms \Rightarrow 5 \cdot t = 109ms$$

La forma de medir será por medio de un conversor ADC del PIC, que para ello se han colocado un divisor resistivo, pues el ADC tiene un rango de 0 a 5V, y en cualquier caso la batería será de 7 voltios o más.

Según la figura del sensor (2.4.2), la relación del voltaje a cuantificar (AN2) con la de batería será la siguiente:

$$V_{AN2} = \frac{(V_{BATERIA} - V_{D3}) \cdot R_{19}}{R_{18} + R_{19}} = 0{,}312 \cdot (V_{BATERIA} - 0{,}7)$$

Por otro lado el límite de voltaje de batería para no dañar el ADC del PIC será el siguiente:

$$V_{BATERIAMAX} = \frac{V_{AN2MAX}}{0{,}312} + V_{D3} = 16{,}72V$$

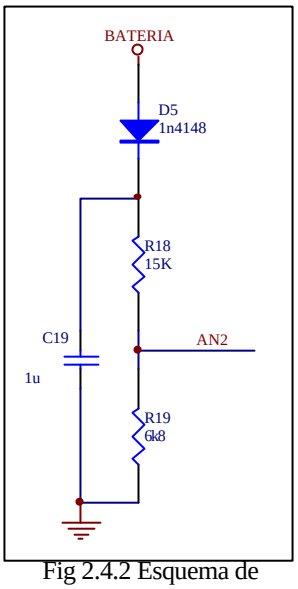


Fig 2.4.2 Esquema de sensor nivel de batería

Para determinar el estado de la carga de la batería, hay que tener en cuenta tanto su tensión nominal como su tecnología de fabricación.
Para las baterías elegidas en el proyecto, las cuales son de NI-MH cuya tensión nominal son 7,2V. Se considera cargada para tensión de un 15% mayor, y descargada al revés 15% menor.

- Tensión nominal batería = 7,2V ➔ Valor cuantificado ideal =104d
- Batería cargada = 8,28V ➔ Valor cuantificado ideal =122d
- Batería descargada = 6,12 V ➔ Valor cuantificado ideal =087d

Una vez determinado los umbrales a partir de los cuales se tomará la decisión del estado de batería, se recomienda realizar una calibración con el sistema debido errores introducidos, por la diferencia del voltaje del diodo, impedancia del ADC finita, tolerancia de resistencias, etc.

2.4.2 Sensores exteroceptivos

Son aquellos que información al robot sobre el mundo exterior.

2.4.2.1 Bumpers

Sensor utilizado para detectar choques con algún objeto. El método es que si hay contacto conduce, y si no lo hay se queda en circuito abierto. El formato para este fin es múltiple, desde 2 cables hasta los interruptores de final de carril, hay tanta variedad como pueda imaginar uno. En el caso del robot diseñado, se le han colocado 4 interruptores de final de carril que se enchufan en los conectores de una señal de **TSA_FPGA**[1]. Estos interruptores funcionan a modo de conmutador, pero se han utilizado como interruptor que está cerrado cuando hay contacto. Cuando choque, el cerebro del robot leerá un cero lógico en la señal, y un uno lógico en el resto de los casos.

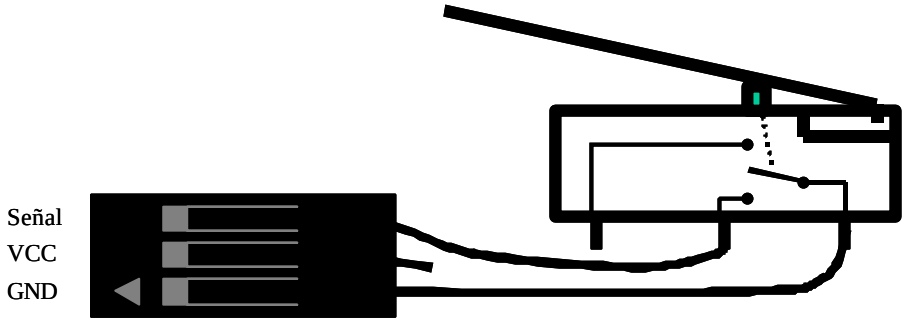


Fig 2.4.3: Montaje del cable del BUMPER.

Para controlar el sensor se toma la señal como entrada, y si se lee 0 significa que el bumper está pulsado.

2.4.2.2 Fotoreflector CNY70

Este sensor es usado de forma muy común en la robótica móvil para detectar una línea negra sobre un fondo blanco y viceversa. Sería propiamente dicho un sensor de color, pero en blanco y negro.

Su funcionamiento se basa básicamente en la reflexión. Contiene un diodo emisor de infrarrojos, y un fototransistor sensible a la luz infrarroja. El sensor diferencia entre blanco y negro por la cantidad de luz reflejada (blanco refleja más que negro). Por lo cual se puede adivinar que para este sensor no es lo mismo un negro mate que un negro brillante. No obstante, para poder medir algo del sensor, es necesario ponerle resistencias de polarización, que se han integrado en la **TSA_FPGA**. El montaje utilizado es el mostrado en la fig 2.4.4.

Los valores elegidos para las resistencias son los siguientes:

[1] Tarjeta de sensores y actuadores, véase apartado 2.3.3.6 para más información

- Resistencia de polarizado del LED de IR = 180Ω para corriente de 20mA.

$$R_{LED} = \frac{Vcc - V_{LED}}{I_F} = \frac{5 - 1{,}25}{20x10^{-3}} = 187{,}5\Omega \approx 180\Omega$$

- Resistencia de polarizado del Transistor = 39KΩ. Se ha elegido por una estimación de la corriente que circulará por el fototransistor en función de la distancia del sensor al suelo (se van a colocar entre 0,5 y 1mm) y la corriente por el diodo. Según las tablas de funcionamiento del sensor CNY70[12], la corriente por el colector será de entre 0,07mA y 0,11mA. Se calcula la resistencia para la variación de tensión que se quiere, teniendo en cuenta que si es muy grande, le afectará más las interferencias lumínicas exteriores (sol, focos...). Al final se fija la variación de tensión en 3V, cuyo punto de decisión se ajusta mediante el potenciómetro[1].

$$R_{TRT\max} = \frac{\Delta V}{\Delta \mathrm{Im}\,in} = \frac{3V}{0{,}08\,mA} = 37500\ \Omega \qquad R_{TRTmain} = \frac{\Delta V}{\Delta \mathrm{Im}\max} = \frac{3V}{0{,}11mA} = 27272\ \Omega$$

R_{TRT} elegida = 39 KΩ para aseaurar el rango.

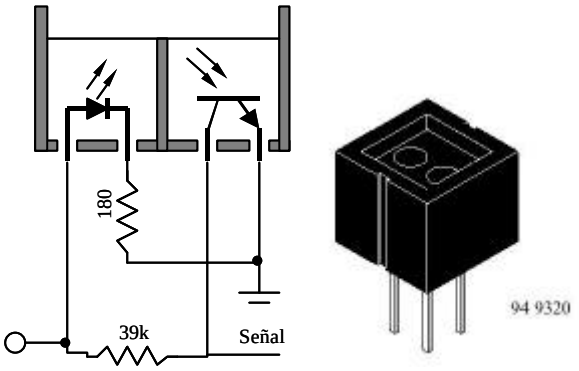


Fig 2.4.4: Sensor CNY70 y esquema de polarización.

Para conectar sensores de este tipo a la tarjeta, se hace por medio de cable de bus de 5x2. En cada cable se conectan 2 sensores, y se pueden poner un total de 4 CNY70 en la placa TSA-FPGA. La forma de montaje se puede verse en la fig2.4.5.

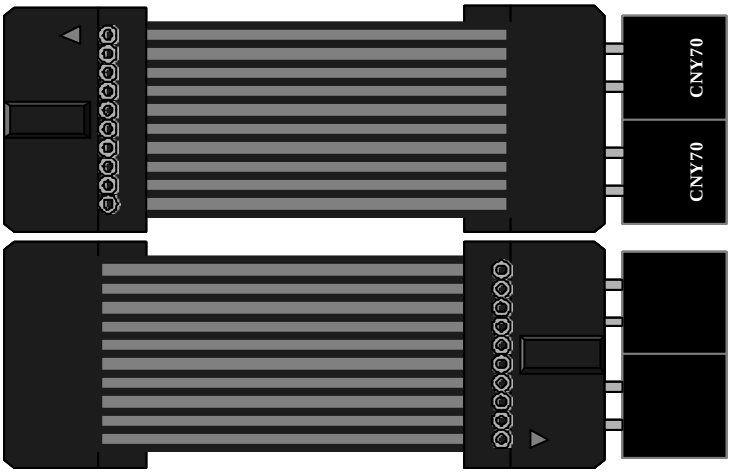


Fig 2.4.5 Montaje del cable para CNY_70

[1] Véase apartado 2.3.3.3 para más información.

Una vez montado los sensores CNY70, se requiere un ajuste de umbral utilizando el potenciómetro multivuelta de la placa, para el tipo de suelo que se va a utilizar.

Respecto al control, simplemente se hace leyendo las líneas de entrada dedicadas de la FPGA correspondiente a cada uno de los CNY70. Según el montaje realizado, se leerá "0" si el sensor está sobre superficie clara, y "1" si la superficie es oscura.

2.4.2.3 Medidores de distancia por IR de Sharp: GP2D02 y GP2D12

Son sensores para detectar objetos, cuya información es vital para que el robot navegue y evite obstáculos.

La técnica de medida de distancia empleada por los DP2DXX es la triangulación.

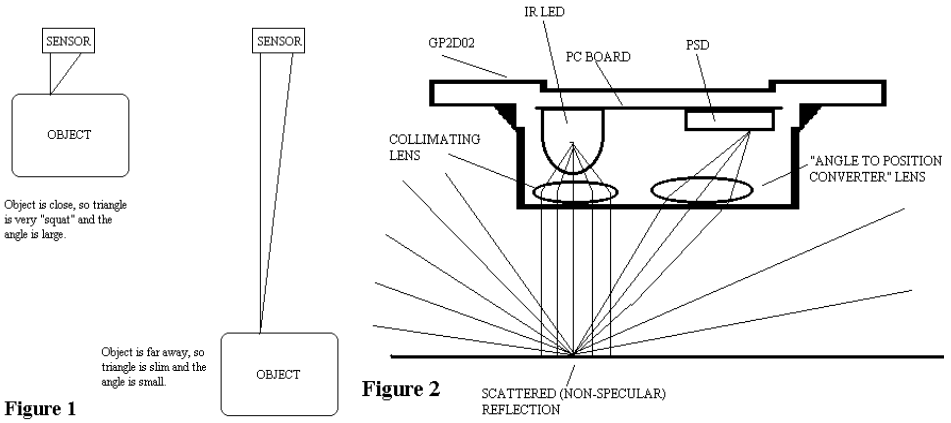


Fig 2.4.6: Medición de distancia por infrarrojos mediante triangulación.

En este proceso, el sensor emite luz que refleja en el objeto y es recibida de nuevo por el sensor. De esta forma hay tres puntos (emisor, reflexión y receptor) que forman un triángulo dándole nombre al método: "Triangulación".

La información de la distancia se obtiene por medio del ángulo con el cual la luz vuelve al sensor.

La triangulación se realiza de la siguiente forma:

- La luz emitida por el emisor de IR, se hace pasar a través de una pequeña lente convexa causando que la emisión del LED se convierta en haz de luz paralelo.
- Cuando la luz choca con el objeto, la luz se refleja en muchas direcciones.
- La recepción de la luz se hace por medio de otra lente convexa que en este caso sirve como un convertidor de ángulo a posición. El rayo de luz que pasa a través del centro de la lente es el que dará la medida.
- Para medir la posición usa un PSD (*Position Sensitive Detector*), el cual es un semiconductor cuya salida es una señal de corriente proporcional a la posición del centro efectivo del haz de luz que incide en él.

La diferencia fundamental entre el GP2D02 y el GP2D12, es la forma de enviar la información al robot. El GP2D02 digitaliza la señal del PSD en un dato de 8 bits que puede ser leído en formato serie, y el GP2D15 convierte la señal del PSD en un voltaje proporcional que se lee por medio de un ADC.

Si el sensor fuese perfecto, la señal leída sería proporcional a la distancia que existe entre el objeto y el sensor, pero esto no es así. Como las medidas que se obtienen son no lineales, existen tablas que tabulan la correspondencia entre una medida y la distancia la que se encuentran los objetos detectados, mostradas en la fig 2.4.7.

El rango de funcionamiento correcto oscila para distancias entre 10 y 80cm.

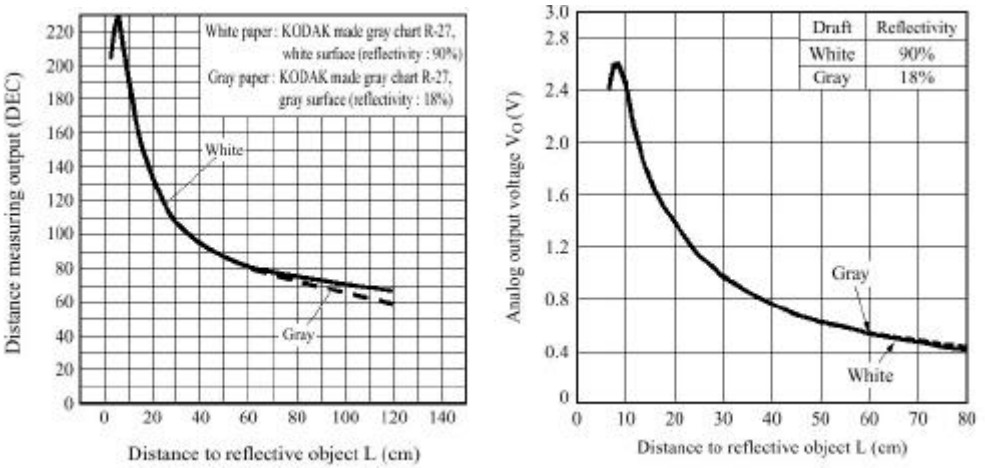


Fig 2.4.7: De izquierda a derecha: Código devuelto por GP2D02 respecto al objeto detectado mas cerca. Voltaje en pin señal ofrecido por sensor GP2D12 respecto a la distancia del objeto detectado.

La forma de conectar los sensores al sistema difiere bastante de uno a otro. El GP2D02 se conecta a los conectores de la FPGA con 2 señales[1], y el GP2D12 irá conectado a los conectores de los ADCs del PIC. Para montar los conectores en el orden adecuado, vea fig 2.4.8.

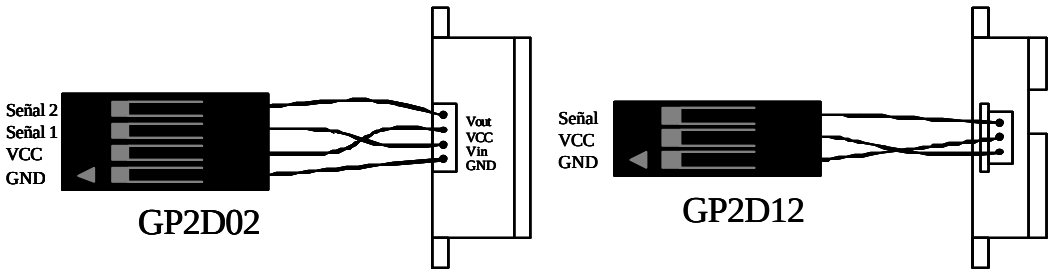


Fig 2.4.8: Montaje de los conectores de GP2D02 y GP2D12 para conectar con la tarjeta TC_FPGA.

Sobre la forma de controlar el GP2D02 y el GP2D12 varia significativamente:

[1] Véase apartado 2.3.3.7 para más información sobre los conectores de 2 señales.

Para realizar medidas con sensor GP2D12 se utiliza el módulo de comunicaciones con el PIC[1] tomando los datos del ADC correspondiente. El µC realiza la conversión analógica / digital entre 0 y 5V cuantificándolo en 8 bits. Para conocer aproximadamente la distancia que devuelve en un momento determinado el sensor se sigue el siguiente procedimiento:

a) Tomar los datos relativos al conversor que se ha conectado el sensor.
b) Convertir los datos en voltaje equivalente a la entrada del conversor.
c) Con el voltaje se mira la gráfica del GP2D12 de la fig2.4.7, tomando la distancia respecto al voltaje calculado anteriormente.

Sin embargo si se utiliza el sensor GP2D02, se usa el módulo de control descrito en el apartado 2.4.4.3 y con los datos obtenidos se mira la gráfica correspondiente de la fig2.4.7. para conocer la distancia aproximada.

2.4.2.4 Sensor de ultrasonidos

La mayoría de los sensores de ultrasonido de bajo coste se basan en la emisión de un pulso de ultrasonido cuyo lóbulo, o campo de acción, es de forma cónica. Midiendo el tiempo que transcurre entre la emisión del sonido y la percepción del eco se puede establecer la distancia a la que se encuentra el obstáculo que ha producido la reflexión de la onda sonora, mediante la fórmula:

$$d = \frac{1}{2} V \cdot t$$

donde V es la velocidad del sonido en el aire y t es el tiempo transcurrido entre la emisión y recepción del pulso. Sin embargo, factores inherentes tanto a los ultrasonidos como al mundo real, influyen de una forma determinante en las medidas realizadas. Por tanto, es necesario un conocimiento de las diversas fuentes de incertidumbre que afectan a las medidas para poder tratarlas de forma adecuada, minimizando su efecto en el conocimiento del entorno que se desea adquirir. Entre los diversos factores que alteran las lecturas que se realizan con los sensores de ultrasonido cabe destacar:

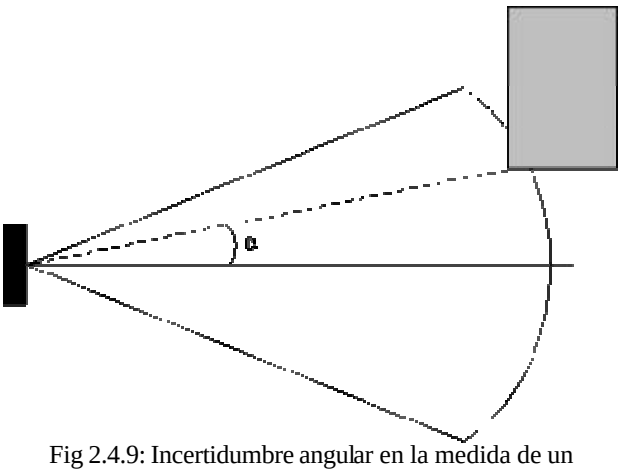


Fig 2.4.9: Incertidumbre angular en la medida de un ultrasonido

[1] Ver apartado 2.2.5.2

- El campo de actuación del pulso que se emite desde un transductor de ultrasonido tiene forma cónica. El eco que se recibe como respuesta a la reflexión del sonido indica la presencia del objeto más cercano que se encuentra dentro del cono acústico y no especifica en ningún momento la localización angular del mismo. Aunque la máxima probabilidad es que el objeto detectado esté sobre el eje central del cono acústico, la probabilidad de que el eco se haya producido por un objeto presente en la periferia del eje central no es en absoluto despreciable y ha de ser tenida en cuenta y tratada convenientemente.
- La cantidad de energía acústica reflejada por el obstáculo depende en gran medida de la estructura de su superficie. Para obtener una reflexión altamente difusa del obstáculo, el tamaño de las irregularidades sobre la superficie reflectora debe ser comparable a la longitud de onda de la onda de ultrasonido incidente.
- En los sensores de ultrasonido de bajo coste se utiliza el mismo transductor como emisor y receptor. Tras la emisión del ultrasonido se espera un determinado tiempo a que las vibraciones en el sensor desaparezcan y esté preparado para recibir el eco producido por el obstáculo. Esto implica que existe una distancia mínima **d** (proporcional al tiempo de relajación del transductor) a partir de la cual el sensor mide con precisión. Por lo general, todos los objetos que se encuentren por debajo de esta distancia, d, serán interpretados por el sistema como que están a una distancia igual a la distancia mínima.

- Los factores ambientales tienen una gran repercusión sobre las medidas: Las ondas de ultrasonido se mueven por un medio material que es el aire. La densidad del aire depende de la temperatura, influyendo este factor sobre la velocidad de propagación de la onda según la expresión:

$$V_S = V_{SO}\sqrt{1+\frac{T}{273}}$$

siendo Vso la velocidad de propagación de la onda sonora a 0ºC, y *T* la temperatura absoluta (grados Kelvin).

Por otro lado, los sensores de ultrasonido móviles (como, por ejemplo, los que vayan colocados en robot móvil) experimentarán un efecto perturbador debido a las pequeñas turbulencias de aire que se producen delante del transductor. En este caso, y al contrario del efecto de la temperatura, la influencia de las turbulencias sobre la señal ultrasónica es muy difícil de ser cuantificada.

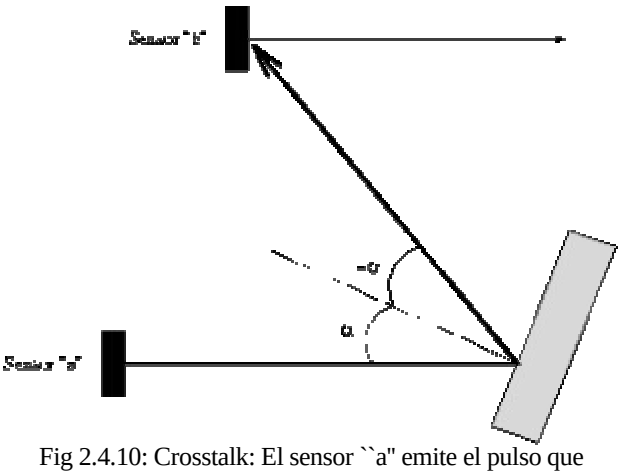


Fig 2.4.10: Crosstalk: El sensor ``a" emite el pulso que recibe el sensor ``b"

- Un factor de error muy común es el conocido como *falsos ecos*. Estos falsos ecos se pueden producir por razones diferentes: Puede darse el caso en que la onda emitida por el transductor se refleje varias veces en diversas superficies antes de que vuelva a incidir en el transductor (si es que incide). Este fenómeno, conocido como reflexiones múltiples, implica que la lectura del sensor evidencia la presencia de un obstáculo a una distancia proporcional al tiempo transcurrido en el viaje de la onda; es decir, una distancia mucho mayor que a la que está en realidad el obstáculo más cercano, que pudo producir la primera reflexión de la onda. Otra fuente más común de *falsos ecos*, conocida como *crosstalk*, se produce cuando se emplea un cinturón de ultrasonidos donde una serie de sensores están trabajando al mismo tiempo. En este caso puede ocurrir (y ocurre con una frecuencia relativamente alta) que un sensor emita un pulso y sea recibido por otro sensor que estuviese esperando el eco del pulso que él había enviado con anterioridad (o viceversa).
- Las ondas de ultrasonido obedecen a las leyes de reflexión de las ondas, por lo que una onda de ultrasonido tiene el mismo ángulo de incidencia y reflexión respecto a la normal a la superficie. Esto implica que si la orientación relativa de la superficie reflectora con respecto al eje del sensor de ultrasonido es mayor que un cierto umbral, el sensor nunca reciba el pulso de sonido que emitió.
- Para emitir un pulso de ultrasonido hay que excitar la membrana del transductor con una señal en forma de delta de Dirac. Sin embargo, en los sensores de bajo coste, la señal excitadora es en la práctica un pulso cuadrado por lo que el efecto resultante es el de la emisión de todo un tren de ondas de ultrasonidos que emergen de la membrana del sensor. El momento **ti** en el que se emite la onda se suele considerar como el momento en el que se emitió la onda excitadora. La dificultad estriba en determinar el momento **tf** en que se recibe el eco de la señal emitida: Tanto la forma la envolvente de tren de ondas de la señal emitida como la recibida, crece desde cero hasta alcanzar un valor máximo y vuelve a decrecer hasta cero. En definitiva, la incertidumbre en la determinación del tiempo transcurrido entre la emisión y la recepción se traduce en un error $\pm$ **e** en la medida.

A la vista de la relación de los diferentes factores que provocan errores o incertidumbre en la medida de un sensor de ultrasonido, se puede realizar una clasificación de los mismos en errores o incertidumbres intratables, tratables directamente y tratables indirectamente. Como errores intratables se pueden considerar la no reflexión de la señal por superficies poco rugosas. Afortunadamente las superficies lisas que se encuentran en entornos reales suelen tener alguna que otra pequeña irregularidad que en un momento dado hacen que la señal se refleje en la superficie. Otras fuentes de incertidumbre en la lectura de la medida realizada, como el tiempo de relajamiento o la resolución del sensor, se pueden tratar directamente modificando y mejorando la tecnología del transductor.

Sonar de ultrasonidos SRF04

Para el robot, se ha optado por comprar un módulo de ultrasonidos denominado **SFR04** que lo comercializa la empresa **robot-electroniks**, cuya la compra se ha realizado por medio de su pagina WEB[1].

Fig 2.4.11: Fotos del modulo de ultrasonidos modelo SRF04

Las características más importantes son:

- Tensión de alimentación: 5V
- Corriente: 30mA Tip. 50mA Máx.
- Frecuencia de funcionamiento: 40Khz
- Máximo rango de medida: 3metros
- Mínima rango de medida: 0,03metros
- Sensibilidad: Capaz de detectar un objeto de 3cm de diámetro a distancia mayor de 2metros.
- Entrada de disparo: Tiempo mínimo de 10us. Compatible TTL
- Pulso de eco: Ancho proporcional a la distancia. Señal compatible TTL.
- Tamaño: 43mm x 20mm x 17mm alto.

El funcionamiento del módulo consiste en dar un pulso en la entrada "*Trigger Pulse Input*" y medir el tiempo del ancho del pulso recibido por la línea Echo Pulse Output. El tiempo que esta señal está activada corresponde al tiempo recorrido por el sonido desde el sensor al objeto y desde el objeto al sensor. Para calcular la distancia, tan solo hay que utilizar la fórmula $d = \frac{1}{2} V \cdot t$, siendo **V** la velocidad del sonido, **t** el tiempo de vuelo, y **d** la distancia entre el objeto y el sensor.

[1] http://www.robot-electronics.co.uk/

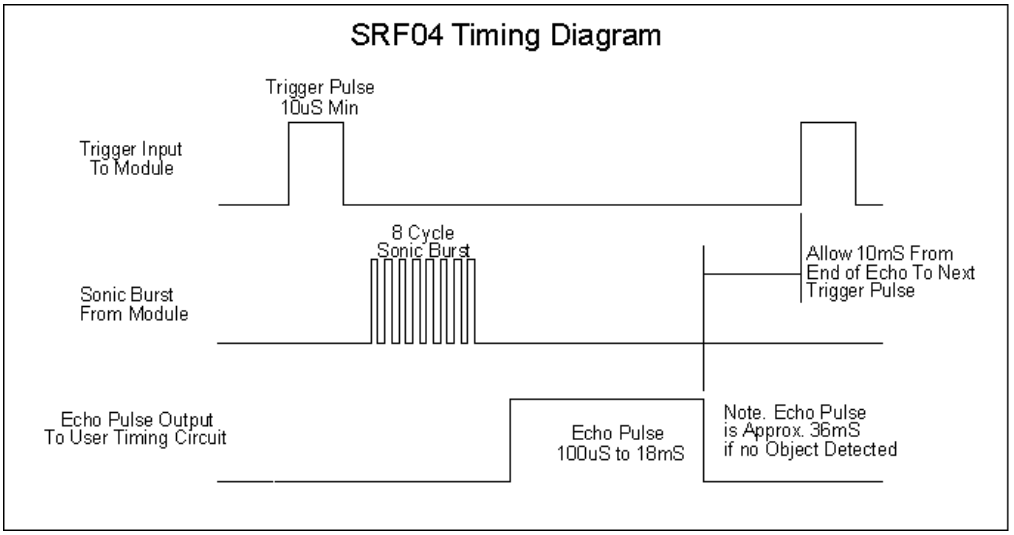


Fig 2.4.12: Diagrama de tiempos para controlar del sensor de ultrasonidos SRF04

Para conectarlo al sistema, se utilizan los conectores para sensores de 2 líneas, y se hace como se puede ver en la figura siguiente.

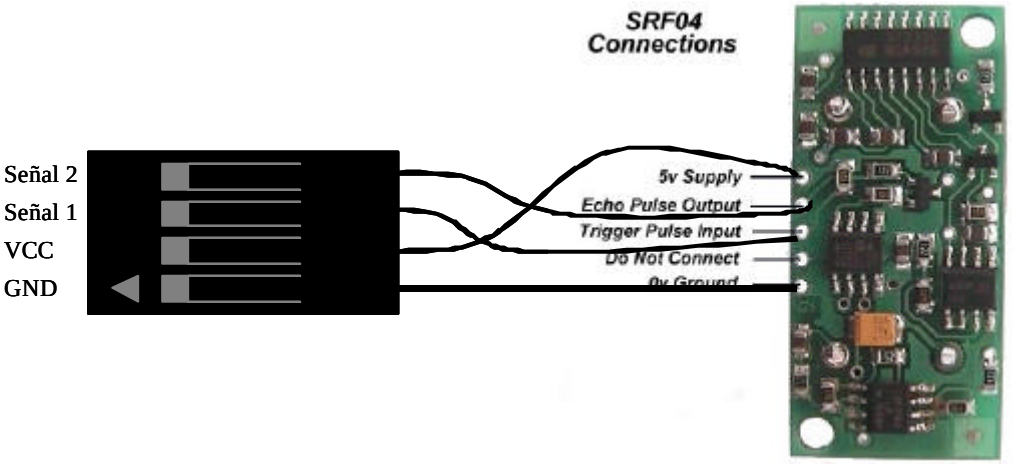


Fig 2.4.13: Conexionado del sensor de ultrasonidos.

Para medir con el sensor, se usa el "Módulo control sonar SRF04" descrito en el apartado 2.4.4.1, de modo que a su salida ofrece los datos respecto a la distancia al objeto detectado en centímetros.

2.4.3 Actuadores

Se le denomina a todo elemento del robot es capaz de modificar el mundo exterior de forma que se pueda controlar.

2.4.3.1 LED

El objetivo es poder señalizar eventos mediante alarma luminosa, pudiendo ser utilizado para depuración, señalizar eventos o estados.

Puede verse ejemplo de modo de conectar en la Fig. 2.4.14, para conectores con 1 línea de señal. Se encenderá con "1" lógico.

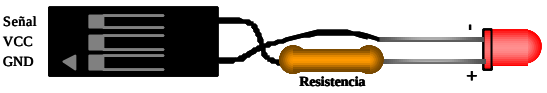


Fig 2.4.14 Conexión de un L.E.D.

2.4.3.2 Motores de corriente continua

Son los actuadores más importantes en un robot con ruedas, pues de encargan de su movimiento. Sus características más importantes deben ser: la facilidad de controlar, consumo relativamente reducido, par y velocidad adecuados para el propósito del robot, peso y tamaño reducido, y no menos importante su precio. También se utilizan motores en robótica móvil para accionar aspiradores, ventiladores, manejo de palas...

Para su conexionado al sistema se realiza mediante las clemas dedicadas a este fin, las cuales tienen la serigrafía correspondiente en la placa.

El modo de controlar motores en robótica móvil es usando un puente en H que se controla mediante modulación de ancho de pulso (PWM) por su facilidad de uso y ahorro de energía.

El diseño del sistema de control para motores se ha explicado en el apartado 2.3.3.2.

2.4.3.3 Servo

Es un dispositivo pequeño que tiene un eje de posicionado controlado. Este puede ser llevado a posiciones angulares específicas al enviar una señal codificada. Con tal de que una señal codificada exista en la línea de entrada, el servo mantendrá la posición angular del engranaje. Cuando la señala codificada cambia, la posición angular del eje cambia. En la práctica, se usan servos para posicionar superficies de control como el movimiento de palancas, patas robotizadas, pinzas e incluso para cambiar orientación de sensores.

Un servo Standardt como el 3003 de Futaba tiene 3kg por cm. de torque. Un servo no consume mucha energía. Puede ver los 3 cables de conexión externa en la fig 2.4.15, siendo uno para alimentación Vcc (Rojo), otro conexión a tierra GND (Negro) y el último es de control.

¿cómo trabaja?

El servo tiene circuitería de control y un potenciómetro el cual está conectado al eje central del servo motor, todo ello para realizar un control de posición del motor. El potenciómetro permite a la circuitería de control, supervisar el ángulo actual del servo motor. Si el eje está en el ángulo correcto, entonces el motor está apagado. Si el ángulo no es el correcto, el motor girará en la dirección adecuada hasta llegar al ángulo correcto. El eje del servo es capaz de llegar alrededor de los 180 grados. Normalmente, en algunos llega a los 210 grados, pero varía según el fabricante. Un servo normal se usa para controlar un movimiento angular de entre 0 y 180 grados, no siendo mecánicamente capaz de retornar a su lugar, si hay un mayor peso que el sugerido por las especificaciones del fabricante.

Fig 2.4.15: Servo Futaba 3003.

¿Cómo se comunica el ángulo a cual el servo debe posicionarse?

El cable de control se usa para comunicar el ángulo. El ángulo está determinado por la duración de un pulso que se aplica al cable. A esto se le llama PCM Modulación codificada de Pulsos. El servo espera recibir un pulso cada 20 milisegundos, no siendo crítico si este tiempo es mayor.

Como se observa en la figura 2.4.16, la duración del pulso indica o dictamina el ángulo del eje (mostrado como un círculo verde con flecha). Nótese que los tiempos reales dependen tanto del fabricante del servo como del modelo, debiendo hacer un calibrado o ajuste para cada servo que se coloque.

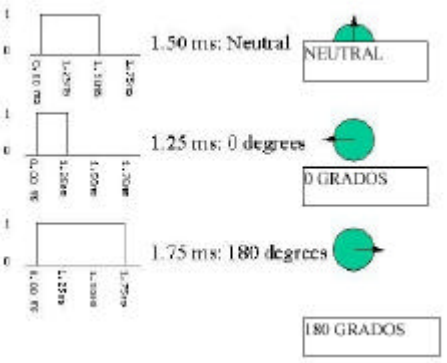


Fig 2.4.16: Ancho de pulsos para realizar control de servos.

Para controlar un servo, se utilizarán los conectores con 1 línea de señal de TSA_TPGA[1].

[1] Véase apartado 2.3.3.6

El voltaje nominal de los servos suele ser entre 4,5V y 6,5V. Teniendo en cuenta que los fabricantes suelen ofrecer las características del mismo para alimentación de 6V, se usará de forma preferente este voltaje para su uso.

2.4.4 Módulos de control

A continuación se describe una serie de módulos de control que han sido creados para facilitar el manejo y control de algunos sensores y actuadores que se pueden conectar a las tarjetas TC_FPGA y TSA_FPGA.

2.4.4.1 Reloj_sistema

Este módulo ha sido creado para proporcionar a cada uno de los elementos del sistema de una señal de reloj de frecuencia adecuada, además de los datos del TIMER para el PWM. En principio no es necesario, pues cada módulo puede tomar directamente la señal de reloj del sistema y acondicionarla según necesidades, pero por motivos de ocupación de la FPGA resulta necesario.

A la entrada se conecta el reloj de la FPGA, correspondiente al oscilador de 2MHz, ofreciendo a la salida las siguientes señales:

- PWM [n-1..0] → Son los datos del timer para el módulo que genera el PWM[1] para el control del motor. Al insertar el componente se configurará el parámetro "**n**" para seleccionar el número de bits del timer.
- Reloj_sonar → Reloj para el módulo de control del sonar SRF04 de frecuencia 15625Hz
- Reloj_GP2 → Reloj para el módulo de control del GP2D02 de frecuencia 7812,5 Hz.
- Reloj_38Khz. → Periodo =26,315µs
- Reloj_10Khz → Periodo =100us
- Reloj_1Khz → Periodo =1ms.
- Reloj_100Hz→ Periodo =10ms.
- Reloj_10Hz→ Periodo =100ms.

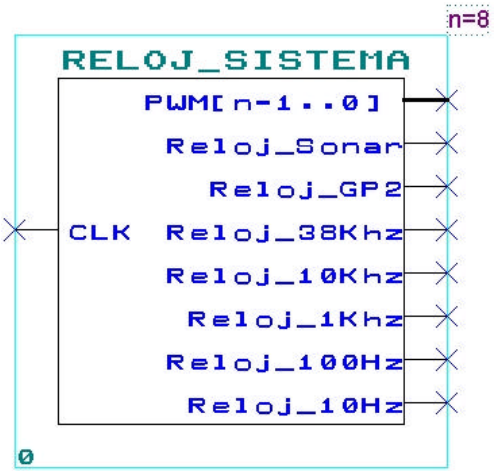


Fig 2.4.17:Símbolo del módulo reloj_sistema.

[1] Modulación por ancho de pulso.

2.4.4.2 Sonar.

El siguiente módulo se utiliza para controlar el Sonar SRF04. La descripción de las señales es la siguiente:

Fig 2.4.18:Símbolo del módulo control del sonar.

- CLK → Su señal de reloj, que se conectará al pin "Reloj_sonar" del módulo Reloj de sistema.
- RST → Reset del módulo, a nivel bajo.
- DATOS [7..0] → Es un bus de 8 hilos que utiliza para devolver en paralelo la lectura realizada del sonar. Devolverá datos de 0 a 254, correspondiente a la distancia en cm que detecta el sensor. Si el dato es 255 significa que reset está activo o el sensor no funciona o no ha sido conectado.
- TRIGUER → Según se ha recomendado conectar este sensor, será la Señal1 del conector correspondiente.
- ECHO→ Según se ha recomendado conectar este sensor, será la Señal2 del conector correspondiente.

2.4.4.3 GP2D02

El módulo tiene las siguientes señales

- CLK → Su señal de reloj, que se conectará al pin "Reloj_GP2" del módulo Reloj de sistema.
- RST → Reset del módulo, a nivel bajo.
- DATOS [7..0] → Es un bus de 8 hilos que utiliza para devolver en paralelo la lectura realizada del sonar. Ofrece datos según gráficas de características del sensor. Si el dato es 0 significa que reset está activo o el sensor no funciona o no ha sido conectado.
- TRIG → Según se ha recomendado conectar este sensor, será la Señal1 del conector correspondiente. Es el pin denominado en el sensor como Vout.
- Data→ Según se ha recomendado conectar este sensor, será la Señal2 del conector correspondiente. Es el pin denominado en el sensor como Vin.

Fig 2.4.19:Símbolo del módulo control del GP2D02

2.4.4.4 MOTOR.

El módulo de control del motor, se encarga de generar la señal de PWM según los datos introducidos como referencia, además de activar el motor en el sentido de giro indicado.

Al introducir el símbolo, hay que configurar el parámetro "**n**" correspondiente al nº de bits para el PWM.

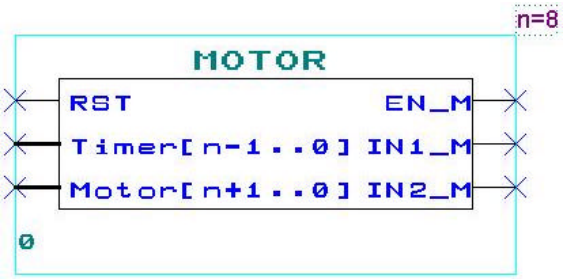


Fig 2.4.20:Símbolo del módulo control de motores.

Las señales son las siguientes:

- RST→ Reset del modulo, que para el motor si está activo (nivel bajo).
- Timer[n-1..0]→Contador para generar el PWM, que se conectará al módulo Reloj de sistema con la salida denominada PWM. El número de bits se configura mediante el parámetro "**n**".
- Motor[n+1..0]→ Es un bus de "**n**" líneas mas 2, de las cuales las 2 de posición más alta indican la dirección del motor y el resto la referencia del PWM. Según las líneas n+1 y n, la dirección para el motor es: "00" atrás, "01" parada rápida, "10" parada rápida,"11" adelante.
- EN_M, IN1_M, IN2_M → Líneas de salida para controlar la electrónica del motor, que se conectará según el motor a controlar en los pines de la FPGA descritos en la tabla de conexiones.

2.5 El robot "APOFIX"

Se explicará como es la estructura, la colocación de los sensores y donde está conectado cada parte del robot para comenzar a trabajar con él.

2.5.1 Estructura física del robot.

Primeras ideas de la estructura.

Para comenzar, se piensa cual va a ser el uso del robot, y a partir de ahí la idea de lo que se necesita va tomando forma.

Las características son las siguientes:

- Ha de ser fácil de manejar y muy maniobrable.
- Se desea que al mover el robot, los sensores no den medidas erróneas por cambios no deseados de inclinación con respecto el suelo.
- El robot se manejará en laboratorio, y debe ser robusto antes las imperfecciones del entorno.
- El ensamblaje ha de ser sencillo para evitar en lo posible problemas mecánicos.

Según las características indicadas, y con el conocimiento adquirido en cuanto estructuras para robots, se ha llegado a la conclusión que lo más adecuado es lo siguiente:

➔ El robot se moverá usando ruedas.

➔ La tracción será diferencial (tipo tanque), para un mejor control y maniobrabilidad.

➔ Se colocarán las ruedas directamente a los motores parar facilitar su montaje.

➔ Tendrá colocados las ruedas sobre el eje radial el centro y además alrededor tendrá un aro para darle forma circular, consiguiendo así que en caso de atascarse, girando sobre sí mismo pueda liberarse.

➔ Al tener 2 ruedas centradas, necesita al menos un tercer punto de apoyo, pero se le colocarán 2 uno adelante y otro atrás del tipo **bola loca** para que al moverse, el cambio de inclinación de los sensores sea prácticamente nulo.

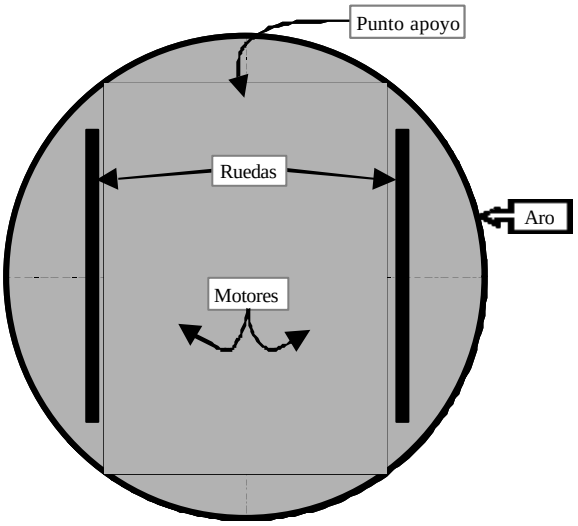


Fig:2.5.1: Primera idea de la estructura

➔ El material elegido para realizar la estructura es principalmente el **Metacrilato** por su facilidad de manipulación y por el acabado.

Estructura final

La estructura se divide básicamente en 3 piezas que forman el robot completo:

- **Soporte base:** la parte fundamental del robot, donde se acoplan los motores, las placas y mayoría de sensores.
- **Puente**. Pieza para colocar sensores que necesiten estar por encima del robot, además sirve de asa.
- **Aro circular**, le dará un aspecto circular y será móvil para poder utilizarlo como sensor de choque.

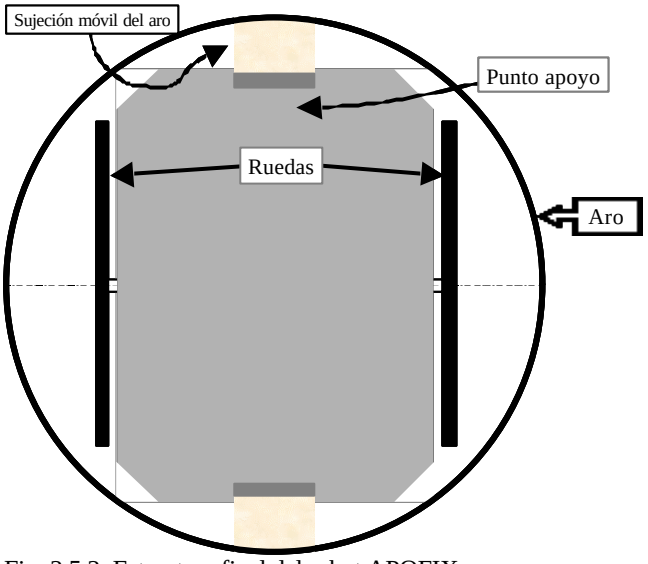


Fig: 2.5.2: Estructura final del robot APOFIX.

Soporte base, dividido en:

➔ Plancha de metacrilato de 155x115x5 con las esquinas achaflanadas, donde se colocarán el resto de piezas para formar el soporte base. Contiene agujeros varios para sujetar placa base, sensores, interruptor...

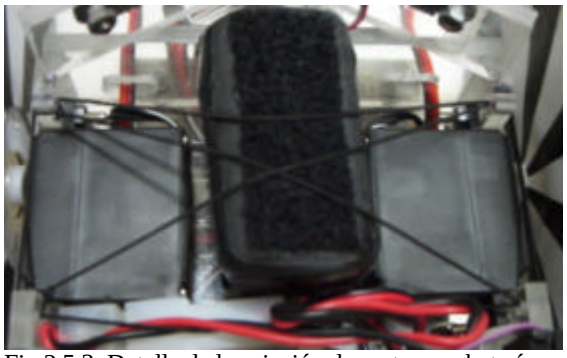


Fig 2.5.3: Detalle de la sujeción de motores y baterías

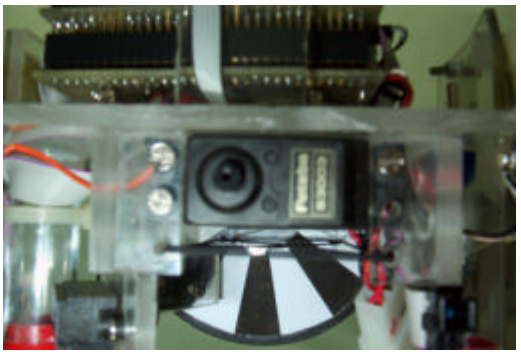


Fig 2.5.4: Detalle lateral del soporte base

➔ Diversas piezas de metacrilato para sujetar: motores, GP2, puente y aro.
➔ Puntos de apoyo[1], compuestos por piezas con extremo terminado en forma de esférica.

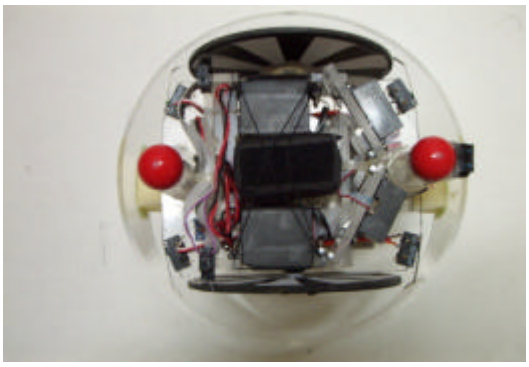


Fig 2.5.5: Vista del robot desde abajo.

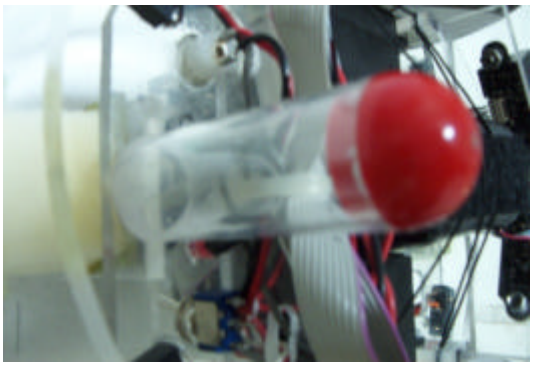


Fig: 2.5.6: Detalle del punto de apoyo.

Puente

Es una pieza de metacrilato de 105x28x5mm con diversos agujeros para atornillar a las piezas de sujeción del mismo que incorpora el soporte base. Al mismo se le añade otra pieza para sujetar el sonar de ultrasonidos.

Aro circular

Es un tubo de metacrilato de 200mm diámetro, 2mm de grosor y altura de 60mm. Además de 2 piezas de goma espuma y 2 de metacrilato para sujetarlo al robot.

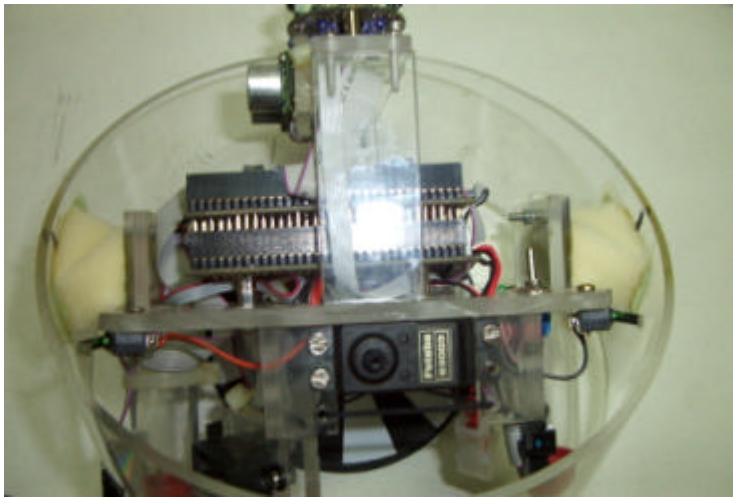


Fig 2.5.7: Detalle lateral del robot.

[1] Ver fig 2.5.6

2.5.2 Motores.

Son servomotores utilizados en modelismo, los cuales se trucan para utilizar tan solo el motor y la caja reductora, obteniendo así un giro completo. Se han elegido por su relación Calidad / Precio, facilidad de montaje, y facilidad de obtención. Los servos elegidos son el modelo FUTABA 3003 o compatible.

Como ruedas se ha elegido CDs pegados a la pieza que se acopla al servo que viene incluida al comprarlo. Para superficie de contacto con el suelo se ha colocado cinta vulcanizable (utilizada por fontaneros para aislar tubos). El tamaño elegido para las ruedas es debido a que el motor es un lento pero con un torque superior al requerido, consiguiendo así velocidad adecuada sin problemas de fuerza en los motores.

2.5.3 Cuadro de mandos

Aparte de los conectores e indicadores que tienen las placas, se ha montado una zona con un reducido cuadro de mandos, que incluye el interruptor de encendido, LED de encendido, LED conectado a la FPGA y conector para cargar las baterías del robot.

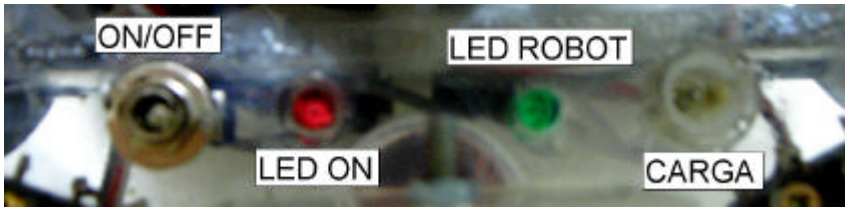


Fig 2.5.8: Detalle del cuadro de mandos.

2.5.4 Baterías

Es la fuente de energía que se usará para alimentar el robot. Para su elección se ha comenzado por el tipo de tecnología, eligiendo baterías recargables de NI-MH por su relación calidad / precio, unido a la facilidad de conseguirlas y de cargarlas. Para el montaje se ha comprado 6 baterías recargables de 1,2V / 800mAh de tamaño RA6/AA y se montan en serie formando un módulo de 6 baterías. El resultado final es una batería de 7,2V de voltaje nominal y una capacidad de 800mAh.

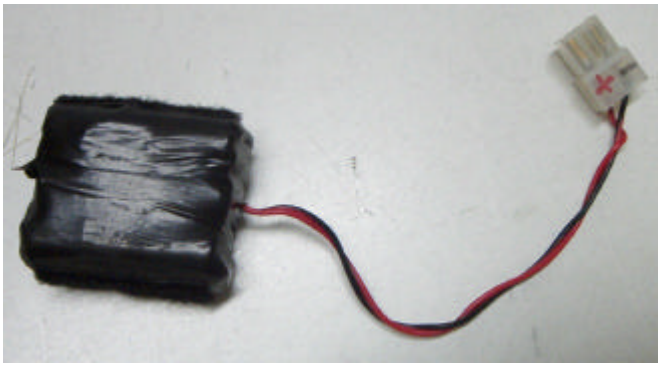


Fig 2.5.9: Detalle del módulo de baterías montado.

Fig 2.5.10: Detalle del conector de batería en el robot.

2.5.5 Situación de sensores

A continuación se detalla la colocación de cada uno de los sensores montados en el robot, así su zona de actuación.

CNY70

Se han colocado 4 sensores de este tipo.
➔ 2 se utilizan para poder realizar seguimientos de líneas en el suelo. Están colocados en la parte delantera del robot sujetos al punto de apoyo, mirando al suelo, tal y como muestran las figuras 2.5.12 y 2.5.14.
➔ Los 2 sensores restantes miran a cada una de las ruedas, que junto con la pegatina colocada en estas forma un encoder[1] para que pueda realizarse posicionamientos del robot.

Fig 2.5.11: Pegatina a escala para formar la rueda-encoder

Sonar

Se ha colocado en el puente, mirando al frente[2].

GP2

Colocados en la parte baja del robot mirando hacia adelante con ligeras inclinaciones a cada uno de los lados, se utilizará para detectar obstáculos cuya altura sea pequeña para ser detectado por el sonar a cortas distancias, también, debido a sus características y posición se podría realizar seguimientos de paredes. Para observar la colocación ver figura 2.5.12, y para el campo de visión ver figuras 2.5.14 y 2.5.15

[1] Ver figuras 2.1.11 y 2.5.12.
[2] Ver figuras 2.5.13, 2.5.14 y 2.5.15. para colocación y zonas de visión del sonar.

Bumpers

Puesto que el aro es móvil, se han colocado en cada una de las 4 esquinas del soporte base del robot para detectar choques, consiguiendo detectar impactos contra el robot.

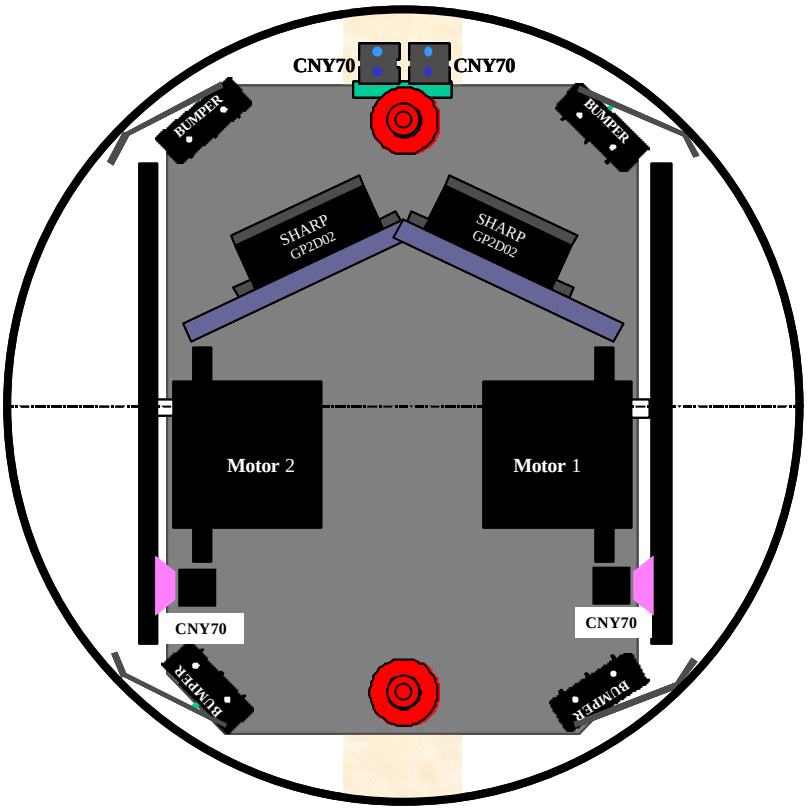


Fig 2.5.12: Figura de colocación de sensores, vista desde abajo

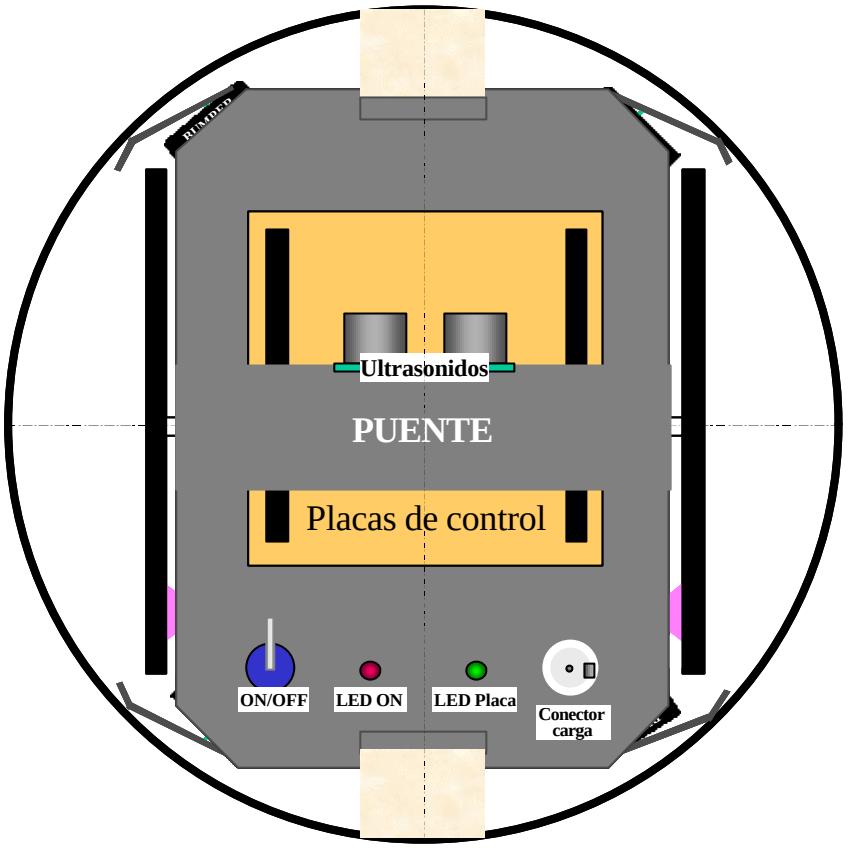


Fig 2.5.13: Figura de colocación de sensores, vista desde arriba.

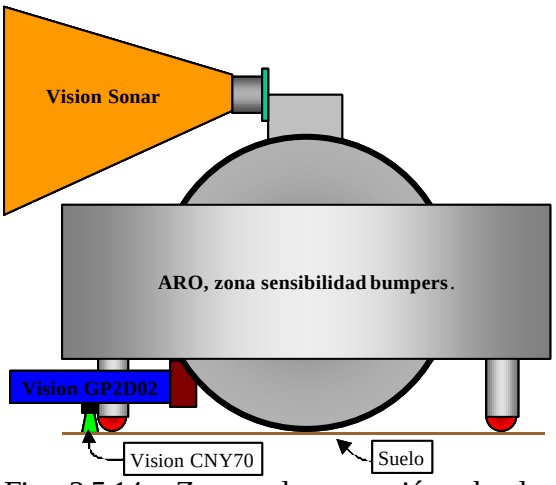


Fig 2.5.14: Zonas de actuación de los sensores, vista lateral.

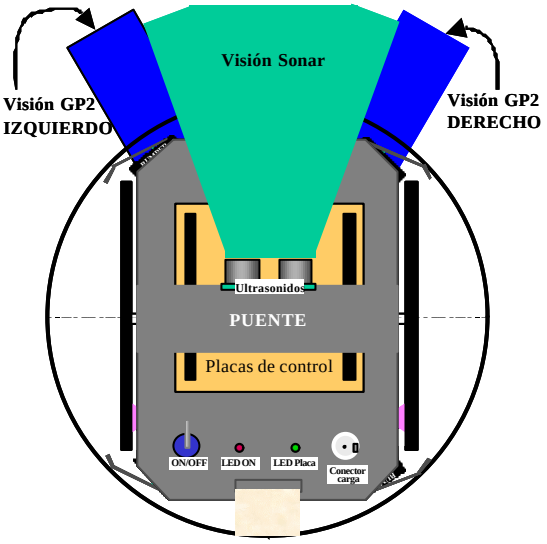


Fig 2.5.15: Zonas de actuación de los sensores, vista superior.

2.5.6 Conexiones

En la siguiente tabla se muestra donde va conectado cada sensor y actuador del robot, así como su correspondencia con los pines de la FPGA, deben configurarse.

Sensores	Conector	Líneas	Pines FPGA	Tipo
Sonar	Con 4 arriba	FPGA_I/O_6 FPGA_I/O_4	11 9	Trigger (Salida) ECHO (Entrada)
Bumper alante derecho	Con 9 arriba	FPGA_I/O_23	60	Entrada
Bumper alante izquierdo	Con 10arriba	FPGA_I/O_25	62	Entrada
Bumper atrás derecho	Con 9 abajo	FPGA_I/O_24	61	Entrada
Bumper atrás izquierdo	Con 10 abajo	FPGA_I/O_26	64	Entrada
GP2D12 derecho	Con 5 arriba	FPGA_I/O_20 FPGA_I/O_21	54 58	TRIG (Salida colector abierto) Data(Entrada)
GP2D12 izquierdo	Con 5 abajo	FPGA_I/O_22 FPGA_I/O_27	59 65	TRIG (Salida colector abierto) Data(Entrada)
CNY alante izquierda	CNY 1,2	FPGA_IN_1	42	Entrada
CNY alante derecha	CNY 1,2	FPGA_IN_2	44	Entrada
CNY encoder derecha	CNY 0,3	FPGA_IN_0	2	Entrada
CNY encoder izquierda	CNY 0,3	FPGA_IN_3	84	Entrada
Sensor estado batería	-	AN2	ADC PIC	Canal2

Actuadores	Conector	Líneas	Pines-FPGA	Tipo
Motor izquierda	Motor 1	EN_M1 IN1_M1 IN2_M1	39 38 47	Salida Salida Salida
Motor derecha	Motor 2	EN_M2 IN1_M2 IN2_M2	51 49 50	Salida Salida Salida
L.E.D.	Con 7 arriba	FPGA_I/O_10	37	Salida

Los conectores: Parte B es abajo, y parte A arriba.

Comunicaciones PIC	Conector	Líneas	Pines-FPGA	Tipo
Read	-	RD	28	Entrada
Write	-	WR	27	Entrada
Dir/Dato	-	DIR/DATO	21	Entrada
Chip select	-	CS	29	Entrada
Bus Datos	-	AD0	25	Entrada / salida
		AD1	24	Entrada / salida
		AD2	23	Entrada / salida
		AD3	22	Entrada / salida
		AD4	19	Entrada / salida
		AD5	18	Entrada / salida
		AD6	17	Entrada / salida
		AD7	16	Entrada / salida

OTROS	Conector	Líneas	Pines-FPGA	Tipo
CLK	-	CLK_0	1	Reloj
RST	-	Reset_FPGA	3	Reset

2.6 Bibliografía

[1] *"Configuration Devices for ACEX, APEX, FLEX & Mercury Devices"* Altera web.*
"FPGA configuration EEPROM Memorys" ATMEL web.*

[2] "SISTEMA DE COMUNICACIÓN VÍA RADIO BASADO EN PC DESTINADO A UNA COMUNIDAD MICROBÓTICA" *realizado por Antonio Gil Olea y Alberto Sánchez Fernández*

[3] *"FLEX 10K Embedded Programmable Logic Devices Family"* Altera web*

[4] *"ByteBlasterMV Parallel Port Download Cable"* Altera web*

[5] *"PC104 conectors"* Harwin web*

[6] *"PIC16F877 datasheet"* Microchip web*

[7] *"Configuring APEX 20K, FLEX 10K & FLEX 6000 Devices"* Altera web*

[8] "MPLAB-ICD USER'S GUIDE" Microchip web.*
"ICDV1" Touzet, Patrick. www. multimania.com/silicium31 *

[9] *"Implementing a Boot loader for the PIC16F87X"* AN732. Microchip web.*

[10] *"Configuring APEX 20K, FLEX 10K & FLEX 6000 Devices"* pag.69 Altera web*

[11] "L6234 datasheet" ST-Microelectronics web*

[12] "CNY 70 datasheet" Vishay Telefunken web*

* Documento contenido en el CD-ROM