

# ÍNDICE

1.	INTRODUCCIÓN.....	1
2.	OBJETIVOS.....	3
3.	ELEMENTOS Y DESCRIPCIÓN .....	6
3.1.	VEHÍCULO.....	7
3.2.	ELECTRÓNICA IMPLEMENTADA.....	8
3.3.	PDA.....	9
4.	INTERACCIÓN DE LOS ELEMENTOS, EJECUCIÓN Y COMUNICACIÓN.....	11
4.1.	SUSPENSIÓN .....	11
4.1.1.	Suspensión Semiactiva.....	12
4.1.2.	Suspensión semiactiva en el car-cross.....	13
4.2.	ELECTRÓNICA E/S.....	16
4.2.1.	Breve descripción de la multitarea.....	17
4.2.2.	Funcionamiento del programa de la tarjeta.....	18
4.2.3.	Protocolo.....	20
4.2.4.	Comunicación .....	25
4.2.5.	Movimiento del amortiguador.....	30
4.3.	INTERFACE DEL USUARIO .....	33
4.3.1.	Configuración del puerto serie .....	37
4.3.2.	Programa final.....	40
4.3.3.	Configuración.....	42
4.3.4.	Cambio posición amortiguador .....	42
4.3.5.	Digitales .....	43
4.3.6.	Gráfica.....	44
5.	MANUAL DEL USUARIO .....	45
5.1.	VENTANA PRINCIPAL .....	46
5.2.	CONFIGURACIÓN .....	48
5.2.1.	Amortiguador.....	51
5.2.2.	Gráfica.....	54
5.2.3.	Digitales .....	55
5.2.4.	Comunicación .....	56

5.3. CAMBIO AMORTIGUADOR .....	57
5.4. GRÁFICA .....	60
5.5. DIGITALES .....	61
5.6. CAMBIO DIRECTO DEL AMORTIGUADOR .....	63
6. MANUAL DEL PROGRAMADOR .....	65
6.1. ELECTRÓNICA .....	65
6.1.1. Variables .....	65
6.1.2. Funciones.....	67
6.2. PDA.....	71
6.2.1. CargarConfiguración .....	71
6.2.2. Configuracion .....	72
6.2.3. Cambio Amortiguador .....	72
6.2.4. Cambio Directo de Amortiguador .....	73
6.2.5. Digitales .....	74
6.2.6. Gráfica.....	75
7. CONCLUSIONES, MEJORAS Y FUTURO .....	76
8. PRESUPUESTO.....	78
9. BIBLIOGRAFÍA.....	80

---

## 1. INTRODUCCIÓN

El presente proyecto fin de carrera “Comunicación e Interacción con la electrónica de un vehículo mediante el uso de dispositivos PDA”. Se ha realizado en el Laboratorio de Automoción de la Escuela Superior de Ingenieros de San Sebastián, TECNUN.

A lo largo de varios años, el laboratorio ha seguido un desarrollo técnico y de conocimientos basándose en los diferentes trabajos y proyectos fin de carrera que se han ido haciendo por parte de los alumnos. Así se ofrece la oportunidad tanto al laboratorio como a los alumnos de ampliar sus conocimientos y aptitudes.

La base inicial fue un vehículo car-cross, que se describirá más adelante, al que poco a poco se le han ido añadiendo elementos tanto mecánicos como electrónicos, hasta llegar al punto de partida del presente proyecto.

El objetivo general de este proyecto fin de carrera es el de mediante el uso de un terminal PDA, comunicarse con la electrónica del coche, tanto para actuar sobre ella, como para recuperar información. Y como objetivo más concreto, el control y la modificación sobre la suspensión semiactiva montada en el coche, proporcionando interfaces sencillas al usuario mediante dispositivos estándar. En nuestro caso una PDA.

Además se busca como objetivo más general con este proyecto el sentar las bases para el desarrollo de más sistemas de comunicación vehículo-PDA, así como la comunicación con electrónica desarrollada en el propio laboratorio, telemetría, sistemas de adquisición de datos etc.

Con todo ello se ha realizado procurando en toda medida dejar abiertas las soluciones a posibles ampliaciones y nuevos usos, así como la facilidad para futuros alumnos que continúen el trabajo de necesitar una breve adaptación para continuar.

Por último, agradecer a D. Joan Savall, como director de proyecto, y D. Xabier Carrera como responsable en el laboratorio, la oportunidad dada de poder hacer el proyecto en el laboratorio. Así como por sus ideas y aportaciones. Agradecer además a

Diego, Arkaitz, Aitor y a los demás por su apoyo durante el proyecto y a lo largo de estos años de estudios. Como también a mi familia.

22 de Septiembre de 2004.

---

## 2. OBJETIVOS

La idea básica que dio origen al proyecto fin de carrera fue el comunicar la electrónica de un vehículo con un terminal tipo PDA, y más concretamente la modificación de los parámetros que controlan el funcionamiento de una suspensión semiactiva, así como el recuperarlos para ser mostrados al usuario que lo solicite.

Además, no como añadidos, sino como parte propia del mismo en un desarrollo lógico. La confección de un mecanismo de comunicación y dominio de la misma, para poder en un futuro próximo ampliar la gama de posibilidades que se ofrecerá al usuario del sistema.

El dominio y entendimiento de los sistemas implicados, vehículo, electrónica y PDA, para seguir ampliando sus funcionalidades y sacar el máximo provecho de las mismas, y aumentar el know-how del laboratorio.

Y por último, dejar la puerta abierta a posibles mejoras, proyectos fin de carrera, trabajos, etc, y que se desarrollarán en el propio laboratorio.

Todo ello va unido a dos objetivos principales propios del laboratorio. Por una lado, la necesidad de conocimiento, control y desarrollo del vehículo y sus leyes en el camino de la entrada en la fórmula SAE.

La fórmula SAE, es una competición automovilística entre universidades en la que compiten coches diseñados por ellas mismas. En Europa se compete en la llamada fórmula Student, división de la fórmula SAE para Europa. Y es éste el primer objetivo del laboratorio.

Las normas de la fórmula SAE son diversas y complejas, pero en líneas generales establece que los vehículos que compitan en ella no deben de ser únicamente los más rápidos, sino que también se exige que cumplan otros requisitos como aceleración, precio, seguridad, innovación, consumo... parámetros que juntos configuran el vehículo ganador.

---

Observando atentamente todo ello uno puede darse cuenta de que se necesita un conocimiento bastante extenso del vehículo y no simplemente unas pinceladas que permitan la construcción de bolido. Es por ello que un sistema que permita el recuperar información o controlar el vehículo para hacer rápidas modificaciones resulta muy útil en estas circunstancias.

Y por otro, el desarrollo de una suspensión semiactiva en colaboración con la empresa de amortiguadores “AP Amortiguadores SA”. El desarrollo de esta suspensión semiactiva queda fuera del alcance de este proyecto. Ya que se trata del tema sobre el que versa la tesis doctoral de D. Xabier Carrera. De forma que este proyecto discurre de forma paralela a la tesis ya citada, sirviéndose de apoyo en la misma, tanto en sus objetivos como en su desarrollo.

De esta forma se trata de añadir un elemento más de valor al trabajo realizado con dicha suspensión. Elaborando un sistema aún mejor y más completo.

Como se ha mencionado anteriormente, además de la acción sobre la suspensión semiactiva. Se trata de controlar la electrónica así como de recuperar información del vehículo y modificar ciertos parámetros.

Estos objetivos no están tan orientados al trabajo que se realiza con la suspensión semiactiva, sino más bien con el trabajo realizado en el camino a la SAE. De este modo, se puede en todo momento modificar el comportamiento del vehículo, o alguno de sus parámetros, para poder observar la influencia de dichas modificaciones en el comportamiento del vehículo y comprender mejor así su funcionamiento, para orientar más tarde todo ese conocimiento en el diseño de un vehículo competitivo en la fórmula SAE.

Además, con las bases de una comunicación estable, segura y útil, se puede adaptar a las diferentes tecnologías presentes y futuras de forma rápida y sencilla, para por ejemplo, sustituir las tradicionales comunicaciones mediante el uso de cables por comunicaciones inalámbricas, como puedan ser por ejemplo Bluetooth o WI-FI.

Con ello lo que se consigue es la posibilidad de interactuar a distancia con el vehículo, y en un futuro la posibilidad de tener un sistema de telemetría propio y adaptado a las necesidades del laboratorio.

Por último, al tener un conocimiento de los sistemas que intervienen en el proceso de comunicación, PDA, electrónica, vehículo, se pueden plantear las mejoras y posibles ampliaciones y desarrollos de cualquiera de ellos. Así como todas las posibilidades que ofrecen y la correcta implementación de las mismas en las diferentes partes.

---

### 3. ELEMENTOS Y DESCRIPCIÓN

A continuación se pasará a enumerar y describir brevemente los elementos necesarios para el desarrollo y uso del sistema. Para más adelante en este mismo apartado describirlos con más detenimiento y explicar todo lo referente a ellos.

Los elementos principales y básicos que configuran el sistema son:

1. Vehículo
2. Electrónica implementada
3. PDA

Con el vehículo no referimos al car-cross del que dispone el laboratorio y sobre el que se realizan los trabajos y avances.

La tarjeta programable, que va montada en el car-cross, es un dispositivo electrónico equipada con varios elementos que se describirán en su apartado correspondiente, que puede ser, como su propio nombre indica, programada en múltiples ocasiones para que realice la tarea deseada. Servirá de puente entre el vehículo y la PDA.

Por último, el terminal tipo PDA es un pequeño ordenador personal con múltiples funcionalidades que permite una interacción prácticamente total entre el usuario y la misma. En nuestro caso nos servirá para comunicar las acciones y requerimientos del usuario al coche.

Seguidamente se describirán con más detenimiento los tres elementos principales.

### 3.1. VEHÍCULO

El vehículo con el que se ha trabajado es un Car-Cross, fabricado por Tenroj modelo Melmac 600TT que posee las siguientes características:

- Chasis tubular con diámetros de 32 y 40mm.
- Motor central Honda CBR600 F.
- Potencia de 92CV a 12000rpm.
- Par máximo de 12 mdaN a 10500rpm.
- Régimen máximo a 12400rpm.
- Caja de cambios secuencial con 6 marchas y sin marcha atrás.
- Peso de 300kg.
- Tracción trasera por cadena sin diferencial.
- Frenos de disco de 5mm. Dos delanteros y uno trasero en el eje. Reparto de frenada regulable.
- Suspensión en doble triángulo independiente formada por un conjunto muelle-amortiguador en cada rueda. Con precarga del muelle ajustable y amortiguador regulable.
- Dirección piñón-cremallera.

Nos centraremos sobre la suspensión del vehículo.



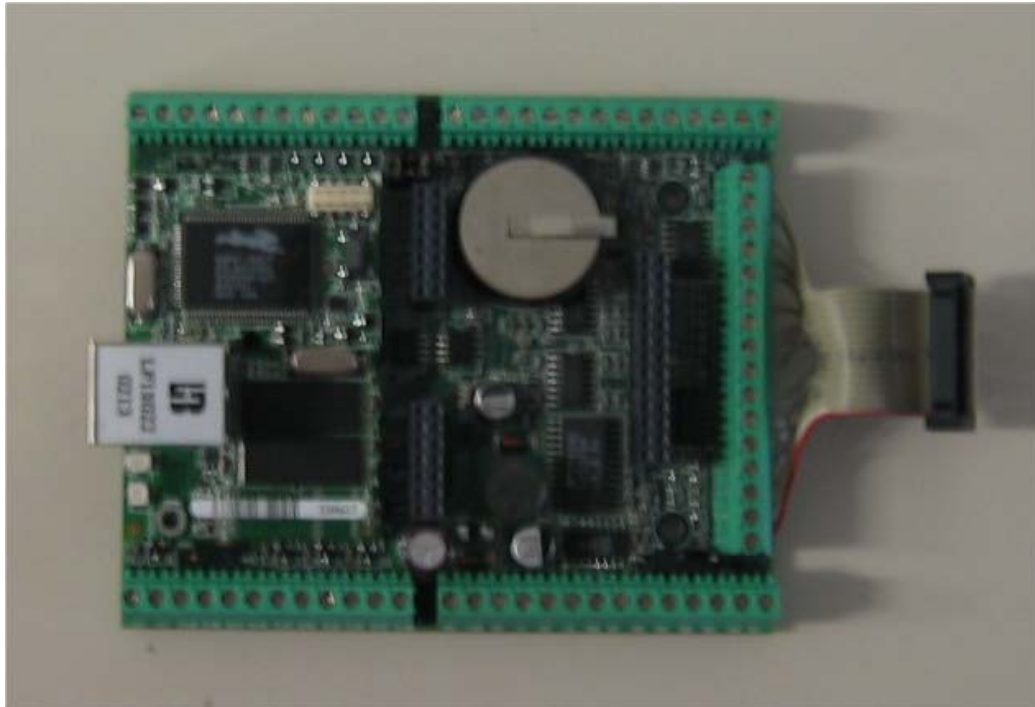
Figura 1

### 3.2. ELECTRÓNICA IMPLEMENTADA

La electrónica de la que se dispone en el laboratorio es una tarjeta programable modelo Smartcat (BL2100) fabricada por Z-World. Esta tarjeta posee las siguiente características:

- Microprocesador Rabbit 2000 a 22.1Mhz
- 512k de memoria RAM y 512k de memoria Flash
- 24 entradas y 16 salidas digitales
- 4 entradas analógicas entre 0-10V
- Conector RJ-45 Ethernet
- 4 puertos serie, 2 RS-232 o un RS-232 con CTS/RTS o un RS-485

Dicha tarjeta va montada en el car-cross, en una caja especial para esta función, y conectada al resto de la electrónica del vehículo, más concretamente a aquella electrónica necesaria para el funcionamiento de la suspensión.



**Figura 2**

El lenguaje de programación usado con la tarjeta es Dinamyc C Premier. Un lenguaje específico para la tarjeta basado en C, y desarrollado por la propia empresa Z-World. Lenguaje que para el programador habituado a C no supondrá ningún problema.

### **3.3. PDA**

La otra parte importante del sistema es la PDA. Pues permite al usuario interactuar con el vehículo, si bien no lo hace de manera directa, ya que no puede hacerlo por su propia configuración. Únicamente posee conexiones de comunicación o periféricos, pero no puede actuar con ningún elemento externo como si puede hacerlo la tarjeta, por ello debe valerse de ésta para poder interactuar con el car-cross.

De esta forma cuando el usuario quiere realizar una acción sobre el coche, la PDA deberá: primero ver que lo que el usuario quiere hacer sea posible, y una vez que lo que se quiere hacer es posible comunicarle a la tarjeta lo que debe hacer.

El dispositivo PDA con el que se ha trabajado es una iPAQ 3870 de Compaq, con:

- Pantalla a color de 64K
- 64Mb de RAM
- 32Mb de memoria ROM
- Bluetooth
- Windows Ce 3.0
- Procesador ARM a 233MHz
- Batería de litio
- Módulo para tarjetas SD



Figura 3

## 4. INTERACCIÓN DE LOS ELEMENTOS, EJECUCIÓN Y COMUNICACIÓN

Una vez visto en qué consiste cada una de las partes que forman el sistema las veremos en más detalle, explicando detalladamente cual es su función y modo de operación.

### 4.1. SUSPENSIÓN

Ésta consiste en una suspensión semiactiva, que consta de un muelle y un amortiguador regulable en su coeficiente de amortiguamiento (Figura 4). Estrictamente hablando no se puede decir que se varíe el coeficiente de amortiguamiento, pues este tipo de amortiguador no lo tiene ya que no es lineal. Sin embargo para no confundir y para simplificar, a pesar de no ser cierto, no referiremos a él como coeficiente de amortiguamiento.

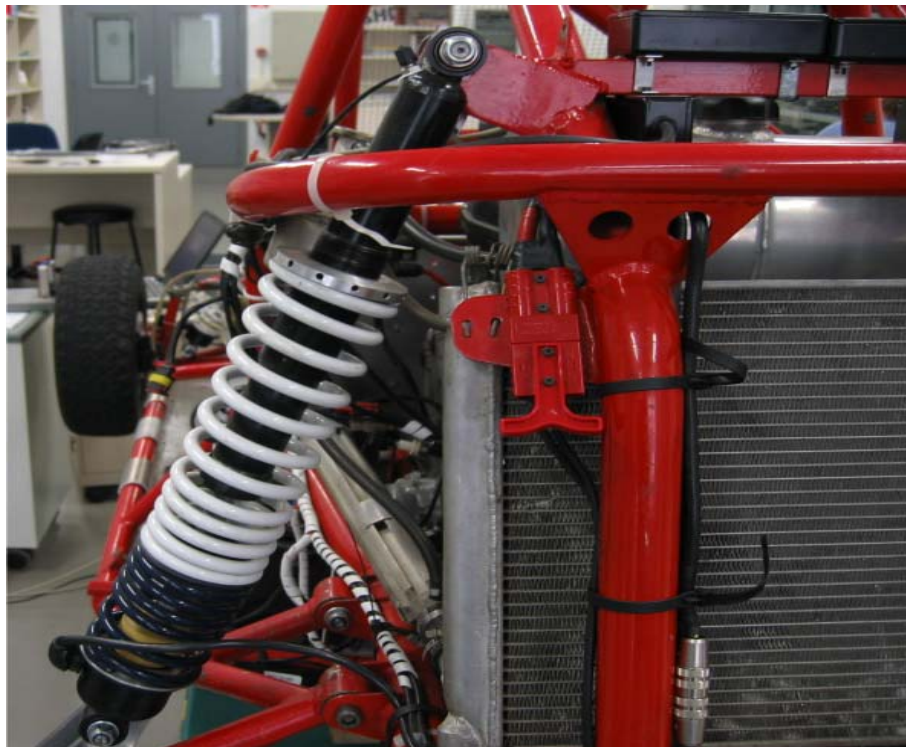


Figura 4

#### **4.1.1. Suspensión Semiactiva**

Las suspensiones tradicionales formadas por un resorte y un amortiguador, lo que buscan es asegurar el contacto permanente de las ruedas con el suelo, ya que son éstas las únicas que unen al vehículo con el terreno y por tanto las únicas que transmiten las órdenes del conductor a la carretera, a la vez que se aísla al vehículo de las irregularidades del terreno para mantener el confort dentro del vehículo. Manteniendo así la seguridad del vehículo y los pasajeros.

Nuestros objetivos, además de los ya citados, se busca también:

1. minimizar las fluctuaciones de la carga vertical
2. reducir los recorridos de la suspensión

Y por último, determina en parte el comportamiento del vehículo en las diferentes situaciones en las que se puede encontrar, paso por curva, aceleración, frenada, etc.

Tradicionalmente, los vehículos montaban un resorte y un amortiguador fijos, en una determinada configuración y todo el sistema se mantenía constante salvo que algunos de los componentes fuese cambiado. Por ello las leyes que regían su comportamiento en todas las situaciones eran las mismas, independientemente de que se estuviese acelerando, frenando, pasando por una curva rápida o lenta.

Sin embargo, para una conducción óptima, las diferentes situaciones pueden requerir distintos comportamientos, por eso, con la llegada de la electrónica a los vehículos, empezaron a aparecer diversos sistemas que variaban las características de los mismos en cada momento adaptándose a las necesidades de cada momento.

Entre estos sistemas aparecieron las suspensiones semiactivas y activas. Como se ha mencionado anteriormente, las propiedades tanto del resorte como del amortiguador permanecían constantes. Por ello la suspensión semiactiva dio un primer paso al variar o poder variar las propiedades del amortiguador, en concreto su coeficiente de amortiguamiento. Lo que busca entonces una suspensión semiactiva es, utilizando elementos pasivos como son los resortes y amortiguadores, variar sus propiedades para ajustarlas a las necesidades del momento.

Las suspensiones activas por su parte dan un giro radical al sistema, haciendo desaparecer los elementos pasivos (resortes y amortiguadores), para ser sustituidos, generalmente por sistemas hidráulicos, que introducen fuerzas en el sistema, de esta forma nos encontramos con sistemas activos y no pasivos como en los casos anteriores.

No es la misión de esta memoria hablar en detalle de los diferentes tipos de suspensiones, sus ventajas y desventajas, sino simplemente dar una idea básica al lector par que entienda la problemática planteada.

Para más información al respecto puede consultar el libro publicado por el laboratorio “Amortiguadores y Suspensión” o buscarlo en su página Web <http://www.tecnun.es/automocion/> .

#### **4.1.2. Suspensión semiactiva en el car-cross**

Ya se ha visto arriba en unas breves pinceladas en qué consiste una suspensión semiactiva. Ahora veremos como está montada y como funciona en nuestro caso en el car-cross.

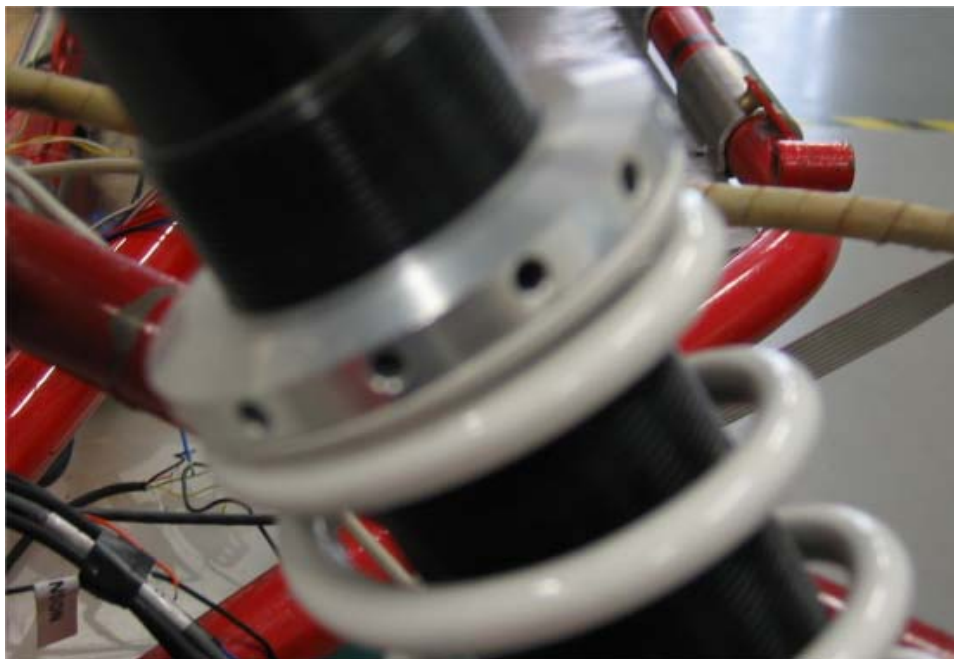
Lo primero, mencionar que los amortiguadores regulables fueron suministrados por la empresa ubicada en Pamplona “AP Amortiguadores S.A.”.

La suspensión la forman un muelle axial y un amortiguador regulable montados de forma telescópica.



**Figura 5**

Como puede observarse en la Figura 6, el muelle únicamente es regulable en su longitud inicial (precarga), pero que una vez fijada no se puede mover en marcha. Sin embargo el amortiguador regulable es cambiado por un motor paso a paso situado en la cazoleta que se encuentra en la parte inferior del mismo como puede observarse en la Figura 7.



**Figura 6**



**Figura 7**

El sistema de funcionamiento del amortiguador regulable, consiste en el giro del eje del amortiguador que regula internamente la apertura o cierre de los orificios por los que circula el aceite contenido en el amortiguador, y que como resultado terminan variando el coeficiente de amortiguamiento de la suspensión.

Ésta puede variar de una blanda a una más dura, en un amplio rango, aunque actualmente de está trabajando con 18 posiciones estables, hasta ahora eran 9 pero recientemente se dio el salto, en un futuro próximo se tratará de trabajar con 18-20 posiciones.

El giro del eje es producido por un motor paso a paso situado en su parte inferior. Un motor paso a paso es un motor eléctrico en el que el giro no es continuo, sino que como su nombre indica gira por pasos. Este tipo de funcionamiento permite posicionar el rotor del motor en una posición indicada. Lo que en este caso se traduce en una posición del amortiguador, y por tanto en un coeficiente de amortiguamiento determinado.

Por tanto se concluye que para la regulación del amortiguador se necesita algún elemento electrónico externo a la suspensión que sea el que regule el rango de

---

movimiento y la que ordene el movimiento al motor del amortiguador. Es aquí donde la tarjeta programable tiene su papel e importancia.

## **4.2. ELECTRÓNICA E/S**

La tarjeta (Figura 2) tiene una doble misión, por una parte es la encargada de mover los motores paso a paso de los amortiguadores, y por otra parte, servir de puente entre el vehículo y el usuario de la PDA.

Para entender como realiza las diferentes tareas primero deberemos ver la forma en la que la propia tarjeta funciona.

Al ser programable, la tarjeta deberá de tener al menos dos modos de funcionamiento, por una parte modo programación y por otra modo ejecución. El modo programación no nos interesa debido a que únicamente es para transferir los programas y depurarlos y no va a ser su funcionamiento normal.

El modo que nos interesa es el modo ejecución. En el modo ejecución la tarjeta carga el programa que tiene escrito en memoria al encenderse y comienza a ejecutarse. Si el programa tiene fin, cuando la tarjeta llega a ese fin, terminará la ejecución, no volverá a empezar de nuevo la ejecución del programa desde el principio. Por ello si se quiere realizar una tarea repetitiva se deberá mantener la ejecución del programa.

En modo ejecución, si el programa no tiene fin, la finalización del mismo se hará cuando bien se reinicie la tarjeta o bien se apague definitivamente, volviendo a comenzar de nuevo la ejecución del programa al volver a recibir alimentación.

Con lo visto anteriormente, nos damos cuenta, de que los programas que realicemos para la tarjeta deberán tener dos partes bien diferenciadas. Una primera parte lineal, en la que se definan variables, funciones, se inicialicen y se puedan realizar ciertas acciones iniciales como pueda ser un reset de los amortiguadores o una puesta a cero de las salidas digitales.

Y, una segunda parte que podríamos llamar elíptica, que consiste en un bucle infinito (que no tiene fin), en el que se incluyen todas las tareas que son repetitivas. De esta forma hasta que no se corte la alimentación a la tarjeta se mantendrá ejecutando el programa.

Como se recordará, la tarjeta tiene dos tareas fundamentales, interactuar con la suspensión y el usuario. Nos podemos dar cuenta de que ambas no tienen por qué estar sincronizadas, sino que pueden ocurrir en cualquier instante, por ello las funciones o rutinas que gestionen cada una de las acciones deberán ejecutarse por separado.

Esto se puede conseguir mediante el uso de la multitarea.

#### **4.2.1. Breve descripción de la multitarea**

La multitarea nació baja la necesidad de realizar acciones simultáneas con el uso de un único procesador. Para entender la multitarea, primero deberemos entender como funciona un procesador y como ejecuta un programa.

Dejando de lado la configuración y funcionamiento interno de un microprocesador sobre los que hay abundante literatura, podríamos resumirlo en lo siguiente: el microprocesador lee una línea del programa que está ejecutando, compara esa línea de código con las instrucciones propias del microprocesador y entonces realiza la acción.

Por otra parte, la organización de los programas es lineal, es decir, que nosotros le decimos al procesador paso por paso lo que debe hacer. De esta forma cuando escribimos una función lo que le damos es la lista de cosas que tiene que hacer. De modo que cuando en un programa llamamos a una función y seguidamente a otra, hasta que no termine con la primera no empezará con la segunda. Puesto que irá leyendo primero la lista de tareas de la primera función, para luego leer las de la segunda.

Ahora bien, ¿qué ocurre cuando necesito que dos tareas sean simultáneas?. Ese era el gran problema que planteaban y plantean los procesadores. Por ello se ideó la multitarea. En la multitarea lo que hace el procesador no es leer paso por paso el programa, sino que reparte su tiempo de operación entre las diferentes tareas que se hayan establecido.

Supongamos por ejemplo que tenemos dos tareas, A y B. En un sistema tradicional primero ejecutaríamos A para seguidamente ejecutar B. En un sistema multitarea empezaríamos ejecutando A, al poco de empezar a ejecutar A, pasaríamos a ejecutar un poco de B y de nuevo A, de esta forma conseguimos que A y B se ejecuten simultáneamente. Esto tiene sus ventajas e inconvenientes sobre los que no entraremos a discutir en este documento.

#### **4.2.2. Funcionamiento del programa de la tarjeta**

Una vez que conocemos qué es la multitarea y como funciona, podemos entender mejor como funciona nuestro programa. Ya que la tarjeta dispone de la posibilidad de realizar multitarea. Y así haremos uso de la misma.

Las tareas que resultan evidentes que va a tener la tarjeta son las siguientes:

- Actuación sobre el amortiguador
- Interacción con el usuario

Como se mencionó en los objetivos, la misión de este proyecto no es el desarrollo de la suspensión semiactiva, sino la modificación de la misma. Por ello nuestros programas serán sencillos y no exigirán una actuación continuada sobre la suspensión. Esto nos elimina la primera tarea, sin embargo, debido a las necesidades creadas por el software creado para la PDA, es decir, dadas las funciones de las que se decidió dotar a la PDA y al sistema, es necesaria la inclusión de otra tarea, en este caso se trata de la comprobación de los estados de las entradas y salidas digitales y las entradas analógicas y su posterior envío a la PDA.

Así finalmente, las tareas que resultan son las siguientes:

1. Manipulación de la posición del amortiguador
2. Estados de las entradas y salida digitales y entradas analógicas y su comunicación.

De cualquier forma, si en un momento se necesitase añadir alguna tarea más, no supondría ningún problema, desde el punto de vista de la programación. Pues el problema con la multitarea es que las tareas se ralentizan, así a más tareas más tiempo tardarán en ejecutarse, por lo que hay que tener especial cuidado al programar cada tarea para no sobrecargar al resto, y además si una operación especial necesita de un tiempo breve para ejecutarse, puede darse el caso de que el uso de multitarea no sea adecuado pues supere el tiempo necesario para llevarla a cabo.

A modo de ejemplo, en las primeras fases del proyecto, cuando únicamente se tenía como tarea el cambio de la posición del motor, la respuesta ante la petición de un cambio de posición por parte de la tarjeta, y por ende del motor podía considerarse instantánea, sin embargo al incluir la segunda la tarea, el tiempo de respuesta se incrementó considerablemente. Y no de forma constante, sino que dependiendo del estado de la ejecución del programa podía darse el caso de que la respuesta fuese instantánea o tardase casi un segundo.

Con este ejemplo debe quedar claro que aunque añadir una tarea sea fácil, no se debe olvidar que ésta es una carga más al procesador y que influirá negativamente al resto de tareas.

A parte de la velocidad con la que pueda estar trabajando la tarjeta, se están realizando diferentes ensayos para determinar tanto la velocidad máxima de giro del motor, como el ancho de banda del motor, en el que influyen tanto la amplitud como la frecuencia de la señal.

La segunda tarea consiste en, si hemos solicitado el estado de las entradas analógicas, leer la tensión en las entradas señaladas y después enviarlas hacia la PDA. Si en cambio son las entradas y salidas digitales las que se han pedido, se leerá el estado de las mismas, alto o bajo, y se mandarán hacia la PDA. Ésto se hace cada segundo para no saturar el puerto serie de la PDA, pero este problema lo trataremos más adelante.

La tarea únicamente se realiza cuando una variable específica para ello, indica en el código en ejecución de la tarjeta, que hay una solicitud de los estados de las entradas

y salidas digitales o bien de las entradas analógicas. De esta forma si la variable está desactivada la tarea apenas ocupa tiempo de procesador pues no debe ejecutar más que una comparación.

La primera tarea requiere una explicación más detallada, puesto que la segunda no es más que una utilidad más que se ha añadido al programa. Sin embargo la interacción con el usuario, como veíamos anteriormente es una de las tareas fundamentales de la tarjeta. De este modo veremos primero en qué consiste el protocolo de comunicación y el medio de comunicación utilizado.

#### **4.2.3. Protocolo**

Si bien el sistema de comunicación creado para esta tarea no encaja en la definición exacta de protocolo desde un punto de vista informático, para nuestro uso y de forma general podemos entenderlo como la serie de normas y códigos que hacen que nuestras partes del sistema se entiendan.

En todo sistema de comunicación deben existir ciertas reglas y símbolos conocidos y reconocidos por ambas partes para poder entenderse. Y aunque estas reglas y símbolos hayan ido cambiando con el tiempo a lo largo del desarrollo del proyecto, lo que no ha cambiado ha sido la jerarquización de la comunicación.

Continuamente nos encontramos con sistemas interactivos en la vida real que exigen por parte del usuario realizar una acción para obtener una respuesta, esos sistemas podríamos dividirlos en dos grupos:

- El usuario tiene el control
- El sistema tiene el control

Los sistemas en los que el sistema tiene el control son más seguros, pues el sistema analiza las acciones del usuario para ver si entrañan algún peligro por parte del usuario, tanto para él como para el sistema. Por lo que pueden ser usados por cualquier persona sin demasiados conocimientos de lo que se hace.

Por el contrario en los sistemas en los que el usuario tiene el control, el sistema realiza las órdenes del usuario, no quiere esto decir que no analice las órdenes para ver si entrañan algún riesgo o no, sino que el poder de decisión recae sobre el usuario y no sobre el sistema. Por ejemplo el usuario podría decidir cancelar la ejecución de una orden en cualquier momento y el sistema respondería haciéndolo.

Los sistemas de este estilo, exigen por parte del usuario un conocimiento amplio y seguro de lo que se está haciendo y se va a hacer. En nuestro caso nos encontramos con una sistema del primer grupo. En el que el sistema es la tarjeta y el usuario la PDA.

De esta forma cuando la PDA mande a la tarjeta que realice alguna acción, ésta la realizará de inmediato y comunicará a la PDA que la ha hecho.

Con esto conseguimos descargar a la tarjeta de la tarea de comprobar las acciones que se le manda hacer, ahorrando tiempo de procesador, y además teniendo que notificarlo después a la PDA para que ésta a su vez se lo notifique al usuario. Simplificando enormemente la gestión de errores y reduciendo la casuística.

La misión de la PDA por tanto es filtrar las acciones del usuario final para indicarle únicamente a la tarjeta aquellas acciones que son seguras. Por tanto como veíamos que era necesario en estos casos, el usuario que ordena tiene el conocimiento total y seguro de lo que se está haciendo.

Aquí hay que hacer especial hincapié en el “conocimiento total y seguro” que tiene la PDA. Puesto que es la PDA una simple máquina, no tiene más conocimiento que el que nosotros le damos, por tanto el desarrollador deberá tener cuidado a la hora de programar de que las órdenes que manda desde la PDA a la tarjeta no vayan a producir ningún problema, para ello es necesario el conocimiento del funcionamiento del sistema y la tarjeta. Por lo que aunque una de las grandes ventajas de los sistemas que se intercomunican es que pueden ser desarrollados en paralelo por diferentes personas, no se puede hacer de manera independiente, sino que hay que saber cuales son las posibilidades de cada una de las partes.

Una vez vista de manera simplificada la forma en la que la tarjeta y la PDA se entienden, pasemos a analizarla de forma más profunda.

Al principio el sistema utilizado era mediante un código puramente numérico, en el que la PDA enviaba un número entre 0 y 10. La tarjeta al recibirlo leía y miraba de que número se trataba, y en función del número hacía una u otra cosa, tal y como puede verse en la tabla inferior:

0	Reset de la suspensión
1-9	Mover el motor hasta la posición indicada por el número
10	Envía la posición actual de la suspensión

**Tabla 1**

Más tarde al añadir más funcionalidades al programa residente en la PDA, se pensó en añadir más números, así cada número correspondería con una acción en la tarjeta.

Como puede observarse tras un pequeño análisis, este método resulta terriblemente rígido, pues si por ejemplo el motor pasa de 9 a 15 posiciones por ejemplo, obligaría a retrasar en la numeración a todas las acciones que poseyeran un número superior a 9. Quizá esto no supusiera un gran esfuerzo en la tarjeta, sin embargo no debemos olvidar que también obligaría a cambiar los comandos en la PDA, por lo que se debería ir al código fuente del programa, cambiar la numeración, volver a compilar, etc.

Con pocas acciones podría resultar poco costoso, pero en cuanto se empezase a complicar la carga de trabajo también lo haría, por lo que dejaría de compensar.

Por otra parte además, la numeración correlativa nos obligaría a tener que hacer un gran número de comparaciones en la tarjeta hasta encontrar el número recibido, con la consiguiente pérdida de tiempo. Pues supongamos por ejemplo que queremos ver el estado de la entrada digital número 10, a la que correspondería el número 45 en nuestro código. La tarjeta entonces debería ir comparando el número recibido con todos los números del código que se le han programado hasta encontrar la coincidencia. Estaríamos perdiendo el tiempo en comparar la petición del estado una determinada

entrada digital con por ejemplo el mover el motor a una posición, reset, o cualquier otra acción que no tiene nada que ver.

Por todo ello se terminó finalmente adoptando un código alfanumérico, en el que la primera letra indica lo que se quiere llevar a cabo y el resto del mensaje son los parámetros necesarios para realizar la acción. El código definitivo se muestra en la siguiente tabla:

<b>Código</b>	<b>Significado</b>
ani <sub>1</sub> i <sub>2</sub> i <sub>3</sub> o ap	a -> analógico. Indica la petición del envío de las entradas analógicas
	n -> número de gráficas, indica el número de entradas que se van a pedir, n=1, 2, 3
	i <sub>i</sub> -> número de la entrada analógica elegida
	p -> parar
dm o dne	d -> digital. Indica la petición del envío de las salidas/entradas digitales
	m e -> empezar
	p -> parar
	n -> número de la salida digital a cambiar el estado
gm	e -> estado en que se ha de poner la salida digital
	g -> get. Pide información sobre el estado de la suspensión elegida
	m a -> suspensión delantera izquierda
	b -> suspensión delantera derecha
	c -> suspensión trasera derecha
d -> suspensión trasera izquierda	
sm <sub>1</sub> m <sub>2</sub> m <sub>3</sub> k	s -> suspensión. Se indica que se quiere cambiar alguno de los amortiguadores
	a -> suspensión delantera izquierda
	b -> suspensión delantera derecha
	m c -> suspensión trasera derecha
	d -> suspensión trasera izquierda
	t -> se cambian todos los amortiguadores simultáneamente
	0 -> reset
	k 1-9 -> posición a la que se desea mover el amortiguador
> 30 -> números libres para que haga cualquier cosa	

Tabla 2

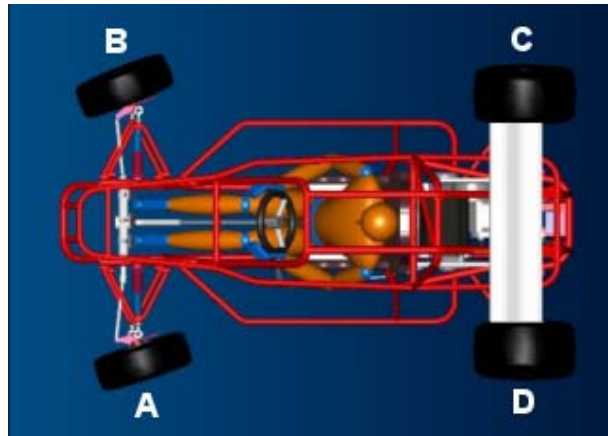


Figura 8

Como se puede ver en la Tabla 2, todos los comandos empiezan por una letra del abecedario, lo que nos permite un gran número de acciones, pues disponemos de todas las letras del abecedario, habiendo además distinción entre mayúsculas y minúsculas, por lo que el número de posibilidades es aún mayor.

También se puede observar que el número de parámetros varían en función de la acción a realizar, por lo que tampoco en este campo estaremos limitados, aunque la programación de la tarjeta y el programa de la PDA se ha hecho para un máximo de 5 bytes puesto que es el código más largo que puede llegar a haber, sin embargo es fácilmente modificable.

A destacar un par de puntos, primero que cualquiera de los parámetros que tenga subíndice es opcional, no así los que no tienen que son obligatorios. Y como puede verse, el código que indica el cambio del amortiguador, mantiene la numeración. De cualquier modo al estar indicado el código por la primera letra, ya no es necesario comparar que acción corresponde a cada número, sino que esta vez son parámetros que se le pasan a la función encargada del cambio de amortiguador.

Además se ha previsto el aumento en el número de posiciones del amortiguador, por lo que la numeración de los programas empieza a partir del 30. Por programas se entienden aquellas leyes de control que para que sean más fáciles de recordar para el usuario se les asigna un nombre, por ejemplo: soft, hard, sport.

Una vez que la tarjeta ha terminado de realizar la acción correspondiente, enviará a la PDA una cadena de texto con la notificación de que la operación ha sido correctamente realizada o que se ha recibido la petición. El texto a mandar es: "OK".

Si la PDA no recibe la confirmación, volverá a intentar mandar la orden a la tarjeta por tres veces, y si a la tercera no lo consigue, lo notificará al usuario para que revise el sistema en busca de errores.

#### **4.2.4. Comunicación**

La tarjeta dispone de la posibilidad de comunicarse vía puerto serie o mediante Ethernet. Para este proyecto se ha elegido la comunicación serie pues también la PDA dispone de este tipo de comunicación y por ser además la más sencilla.

Para la comunicación serie se puede disponer de dos puertos serie RS-232 o uno RS-485. Denominados en la tarjeta B, C y D respectivamente. Nos decidimos así mismo por el RS-232 por ser el más usado y por ser el que maneja la PDA. En lo referente a las comunicaciones y programación se ha usado el puerto B, cuyos zócalos están indicados de manera impresa en la tarjeta.

Para los puertos serie funcionando bajo el protocolo RS-232, la tarjeta sólo dispone de las conexiones TxD (transmit) y RxD (recibe), por lo que todo control por hardware queda descartado y únicamente se podrán llevar a cabo las comunicaciones de la manera más sencilla mediante el uso de estos canales.

El propio lenguaje de programación de la tarjeta trae ya implementadas a lenguaje de alto nivel funciones específicas para el manejo del puerto, por lo que su programación y manejo es sencillo. Únicamente se debe indicar el puerto del que se quiere disponer, se hace mediante su letra B, C o D. Y la velocidad de transmisión de datos, que varía entre 1200 y 19200 baudios.

Se debe prestar especial atención a la velocidad de transmisión de datos, ya que debe ser la misma para la tarjeta y la PDA. Esto puede resultar un pequeño inconveniente, puesto que en la PDA, ésta puede ser cambiada durante la ejecución, mientras que en la tarjeta se indica mediante código, por tanto para cambiar la velocidad es necesario reprogramar la tarjeta.

**Nota:** a la hora de realizar las conexiones existe una errata en el manual de la tarjeta. En éste se indica que para conectar el dispositivo a otro terminal serie se deben cruzar los cables TxD y RxD de ambos. Sin embargo en la impresión en la baquelita de la tarjeta esto ya se ha tenido en cuenta y el cruce se hace internamente, por lo que se debe conectar el cable TxD al zócalo indicado con TxD y el RxD con el RxD de la tarjeta.

Como ya se sabe, la conexión serie no es inalámbrica, por tanto la PDA y la tarjeta deben estar unidas por un cable de tres hilos, que conecte los pines RxD, TxD y tierra de ambos dispositivos.

El cable utilizado es el que se indica en la Figura 9.



**Figura 9**

En uno de los extremos nos encontramos con el conector de la PDA, con forma rectangular. Es éste un conector especial para el modelo de PDA utilizado en este proyecto, y cuya numeración de pines y misión de indica abajo.

De derecha a izquierda, colocado el conector en la forma de la Figura 10, van del pin 1 al 22, y en la Tabla 3 se indica la función de cada pin. Los elementos en rojo son los utilizados en la conexión.



**Figura 10**

Pin 1	V_ADP
Pin 2	V_ADP
Pin 3	V_ADP
Pin 4	V_ADP
Pin 5	Reservado –No Usar
Pin 6	RS232 DCD
Pin 7	RS232 RXD
Pin 8	RS232 TXD
Pin 9	RS232 DTR
Pin 10	GND
Pin 11	RS232 DSR
Pin 12	RS232 RTS
Pin 13	RS232 CTS
Pin 14	RS232 RING
Pin 15	GND

Pin 16	No Conectado – No Usar
Pin 17	Detector del USB
Pin 18	No Conectado – No Usar
Pin 19	USB – UDC +
Pin 20	No Conectado – No Usar
Pin 21	USB – UDC -
Pin 22	GND

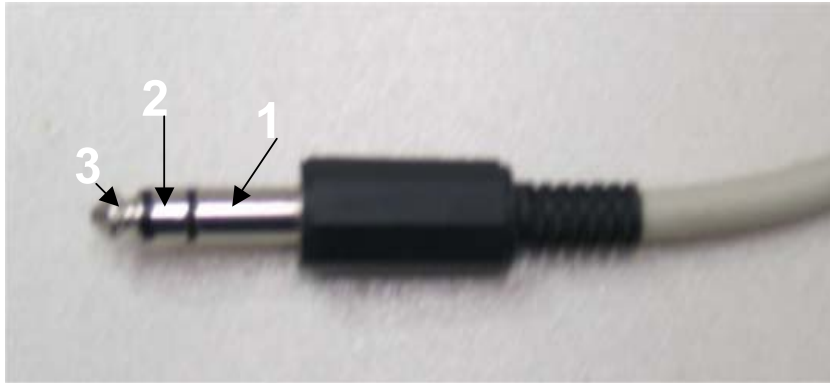
**Tabla 3**

Para la tarjeta se dispone de un conector tipo Jack, de tamaño grande, con tres canales. El macho irá unido al cable, mientras que la hembra se unirá a la tarjeta tal y como se indica en las imágenes siguientes. A su vez la hembra irá sujeta a la caja para la correcta conexión del cable.



**Figura 11**

Las conexiones internas del Jack macho se muestran en la Figura 12 y en la Tabla 4.

**Figura 12**

Pin	Función
1	GND
2	RxD
3	TxD

**Tabla 4**

Además de la posibilidad de comunicarse por el puerto serie, la tarjeta puede comunicarse mediante Ethernet, pero la PDA no, por lo que sería necesario un adaptador especial para la PDA. Además la PDA dispone de la posibilidad de comunicarse mediante Bluetooth, cosa que la tarjeta no puede hacer. De cualquier modo existen adaptadores comerciales que permiten el cambio de Bluetooth a serie o Ethernet, lo que abre la puerta a la posibilidad de disponer de comunicación inalámbrica entre PDA y la tarjeta. Por último no hay que olvidar que se está implantando rápidamente la tecnología Wi-Fi, que supondría además de una conexión inalámbrica, un importante salto en la velocidad de transmisión de datos. Si bien para ello se debería cambiar de modelo de PDA o bien comprar adaptadores especiales para ambos elementos.

De cualquier forma, la tecnología Wi-Fi, quizá sea excesiva por el momento, ya que aunque existen adaptadores, no olvidemos que la final terminamos en un puerto serie o Bluetooth, de velocidad considerablemente inferior, por lo que se presume que la línea a seguir debe de ser el desarrollo de la comunicación Bluetooth.

Una vez visto el protocolo de comunicación entre la tarjeta y la PDA y el medio de conexión, será más fácil entender el funcionamiento de la tarea principal y que se encarga de la interacción del usuario con el vehículo.

---

Como ya se mencionó, el programa queda en un bucle infinito para poder actuar en cualquier momento cuando sea necesario. Por tanto la función básica de la tarea principal es esperar escuchando el puerto serie hasta que recibe algún dato.

En el momento en el que entran los datos por el puerto serie, se activa esa tarea que mientras tanto había permanecido inactiva, y lee los datos del puerto serie. A continuación pasa a comparar el primer byte de los datos con las funciones que tiene programadas. En caso de alguna de las letras que indican las diferentes funciones, coincida con la que ha sido recibida, el programa pasará a llamar a la función correspondiente. Si el primer byte recibido no coincide con ninguna de las letras, se ignora y se vuelve a quedar en escucha hasta la próxima recepción.

Cuando el mensaje recibido por la tarjeta es válido, se hará la tarea correspondiente, y una vez terminada correctamente, se procederá a enviar la notificación a la PDA de que se llevó a cabo de manera satisfactoria, tal y como se vio en el apartado acerca del protocolo.

De todas las posibilidades de que dispone en estos momentos la tarjeta, la más importante sin duda es el cambio del amortiguador, por lo que la veremos más detalladamente.

#### **4.2.5. Movimiento del amortiguador**

El cambio del coeficiente de amortiguamiento, cambio del amortiguador, se produce por el giro del eje del amortiguador, que está producido por el giro del motor paso a paso acoplado al mismo.

Por tanto la misión de la tarjeta al recibir la orden de cambiar el amortiguador será la de llevar al motor paso a paso hasta la posición indicada. Para ello se vale de unas pequeñas tarjetas especiales para mover motores de este tipo.

Luego la función de la tarjeta deberá ser indicar a la tarjeta controladora del motor cuándo debe mover el motor, cuántas posiciones y en qué sentido. La programación de este procedimiento no ha sido realizada dentro del objeto de este proyecto, sino que se tomó la realizada por D. Xabier Carrera durante el desarrollo de su tesis doctoral.

---

Lo primero a determinar es el número de posiciones a mover el motor, para ello se compara la posición actual con la pasada al programa, si son la misma no se hace nada porque ya se está. En caso contrario el número de posiciones a mover vendrá determinado por la siguiente fórmula:

$$\text{movimiento} = \text{abs}(\text{posición\_requerida} - \text{posición\_actual})$$

Inicialmente la diferencia entre las posiciones se multiplicaba por 4 debido a que se trabajaba sólo los pulsos pares, mientras que ahora ya se sabe que ambas son estables trabajando en modo onda (wav) del motor, esto permite un mayor número de posiciones y por tanto de resolución.

Cada posición o movimiento requiere de un pulso, que consta de un estado alto seguido de uno bajo

A continuación se conectan los motores y se comprueba la dirección en la que se debe girar el motor. Un uno (+5V) indica un sentido de giro horario (pasa de la posición 9 hacia la 1, de más duro a más blando) y un cero (0V) al contrario, es sentido antihorario (de 1 a 9, es decir de blando a duro).

Seguidamente para que se mueva se le debe pasar un pulso de tensión Alto-Bajo. Y con ello se habrá movido una posición. Este pulso deberá repetirse tantas veces como indique el número movimiento para que el rotor llegue a la posición solicitada.

Por último se desconectan los motores para que no consuman y se actualiza la posición del motor. El estado de las fases del motor paso a paso no hace falta guardarlas, puesto que la propia tarjeta con la que se controlan los motores las almacenas en su "memoria". De forma que para mover de nuevo los motores se procederá del mismo modo que se ha explicado.

Para cada motor existe una tarjeta controladora, y por tanto, para cada motor será necesaria una salida digital que active la alimentación del motor, indique la dirección de

giro y otra que emita el pulso. El número de cada salida y su correspondencia con el motor y función se indica en la Tabla 5.

	Tensión	Sentido de Giro	Pulso
Suspensión delantera izquierda	0	1	2
Suspensión delantera derecha	3	5	6
Suspensión trasera derecha	12	10	13
Suspensión trasera izquierda	11	7	9

**Tabla 5**

Se estableció la salida número 15 como alimentación general de todas las suspensiones. En cualquier otro caso, la salida señalada, en función de su misión, está conectada a la tarjeta controladora del motor correspondiente y con la entrada que corresponda a dicha función en la tarjeta controladora. Así por ejemplo, la salida 0 estará conectada al relé que permite la alimentación del motor delantero izquierdo, la salida 1 a la entrada de la tarjeta controladora del motor delantero izquierdo en la que se indica el sentido de giro, y así sucesivamente.

Además al inicio del programa se realiza un reset de todos los amortiguadores para conocer su posición, porque como se podrá haber observado el sistema es en lazo abierto, por tanto no hay manera de saber la posición real del motor en cada momento, sino que se asume que la posición en la que se encuentra es la posición a la que se ha mandado.

Se dispone además de la opción de realizar un reset programable en cualquier momento. Hacer un reset no es más que llevar al motor a su posición inicial, con el amortiguador en su comportamiento más blando/duro.

El hecho de que cada motor disponga de una tarjeta controladora propia, y que no compartan salidas digitales, hace que sea posible modificar cada uno de los amortiguadores de forma independiente. Esta opción se ha tenido en cuenta y el

programa residente en la tarjeta y el de la PDA, permiten modificar la posición y hacer reset, tanto de todos los amortiguadores a la vez como de cada uno por separado o varios a la vez.

Por último señalar que la programación de la tarjeta es realmente sencilla y rápida, además de ser potente y muy versátil, pues permite diseñar una gran amplitud de sistemas.

### **4.3. INTERFACE DEL USUARIO**

Si entre la tarjeta y la PDA existe un tipo de relación en la que el usuario tiene el control, entre la PDA y el usuario final, es el sistema el que tiene el control. Si bien se le da cierta libertad al usuario. En cualquier caso se ha procurado que el control sobre las acciones del usuario sea lo más transparente posible de forma que no se sea consciente.

El sistema operativo con el que cuenta la PDA es Windows Ce 3.0. Windows Ce es un sistema operativo especial, creado por Microsoft para los dispositivos con bajas prestaciones de memoria, procesador y demás recursos. Como consecuencia, a sus funciones y librerías están limitadas, por lo que cuenta con menor funcionalidad que Windows en su versión de PC. En cualquier caso son pocas funciones, la mayoría son funciones gráficas o que consumen altos recursos, por lo que para un desarrollo de un programa normal no supone mucho inconveniente.

Al estar equipada con un sistema operativo Windows, la programación está claro que se debe realizar bajo este entorno, por lo que las posibilidades de elección no son muy amplias.

La propia Microsoft distribuía gratuitamente versiones de Embedded Visual Basic y Embedded Visual C++ para programar bajo Windows Ce, de forma que se decidió por usar una de estas herramientas en vez de comprar un software comercial específico. Ya que si bien no es quizá necesario para la programación un entorno de programación propio para Windows Ce, sí lo es para la creación de los archivos ejecutables, debido a

---

que estos deben cumplir con los requisitos del sistema en el que se van a ejecutar y del procesador que los va a ejecutar.

Por ello se decidió finalmente utilizar Embedded Visual C++, porque permite compilar para el sistema operativo Windows Ce así como para los procesadores que montan las PDA y en concreto en nuestro caso el ARM. La opción de C++ en vez de Basic, fue básicamente por la ventajas que ofrece C++ frente a Basic. Puede que la programación en Basic sea más sencilla, sin embargo su potencia también es menor y puesto que era algo que se necesitaba se eligió como lenguaje de programación C++.

Para cualquier persona que haya trabajado con Visual C++, Embedded Visual C++ no le resultará extraño, pues son idénticos. De hecho es la impresión que se acaba teniendo cuando se trabaja un tiempo con él. No da la sensación de que Microsoft haya invertido muchos recursos en desarrollar funciones y entornos específicos par sus sistemas Windows Ce sino que más bien se ha dedicado a cortar y retocar las librerías que ya tenía en Windows32 para adaptarla a Windows Ce.

Esto supone que el paso de un entorno a otro es inapreciable por lo que facilita enormemente la tarea. Sin embargo tiene sus contrapartidas, y es que esa falta de dedicación se nota por ejemplo en la ayuda proporcionada por Microsoft en la que existen numerosas erratas.

Un problema con el que se encuentra a menudo el programador es con la falta de información existente en relación con la programación de este tipo de dispositivos, y mucho menos en temas de comunicación. Por suerte, como se ha dicho anteriormente, la programación es prácticamente similar a Windows32 por lo que la información que se puede encontrar para programar bajo este entorno es directamente traspasable a Windows Ce en la mayoría de los casos.

Al igual que la tarjeta programable resulta extremadamente potente para realizar innumerables acciones, la PDA es una herramienta muy potente, pues permite el desarrollo de aplicaciones similares a las de un PC convencional aunque en la palma de la mano. Con las limitaciones que ya se han citado, como la reducción de funciones o peor capacidad de cálculo entre otras. Nos encontramos por tanto con un problema de

prestaciones frente a versatilidad, que en este caso es ganado por la versatilidad, pues las prestaciones son más que suficientes.

El problema llega cuando los fabricantes deciden investigar más en la mejora de las funciones gráficas y ofimáticas y dejar de lado las comunicaciones, al menos no es bueno para el desarrollo de proyectos como el presente. Ya que aunque sea una herramienta versátil y potente de cara al usuario, sus sistemas de comunicación dejan mucho que desear. Principalmente porque son lentos y poco estables.

No es ésta una única opinión personal, sino que en numerosos foros y páginas de Internet, los usuarios mostraban sus desacuerdos y problemas con los sistemas de comunicación, incluso usando software comercial. Y fueron numerosas las veces en las que PC y PDA no conseguían conectarse, por lo que muchas veces el desarrollo se hace lento ante la imposibilidad de probar los programas en su entorno natural.

Debido a esto, algunas ideas iniciales que se habían planteado han tenido que ser abandonadas o modificadas para adaptarse a la problemática planteada por las comunicaciones. Ya que al menos en este modelo de PDA tanto la comunicación serie como USB resulta demasiado lenta para las velocidades actuales. Más adelante se comentarán pormenorizadamente los problemas surgidos.

Como primera toma de contacto ante el nuevo entorno, se comenzó programando una pequeña aplicación, en vista también de ser usada con el hardware de medición creado en el laboratorio, para calcular el centro de gravedad del car-cross, en su posición X e Y.

Primero de forma manual, próximamente automáticamente vía adquisición, se introducen los pesos medidos en cada rueda y pulsando el botón calcular nos aparecerán las coordenadas del centro de gravedad. Figura 13



Figura 13

Seguidamente de creó una primera versión del programa para cambiar la posición del amortiguador. Tal y como puede verse en la Figura 14.

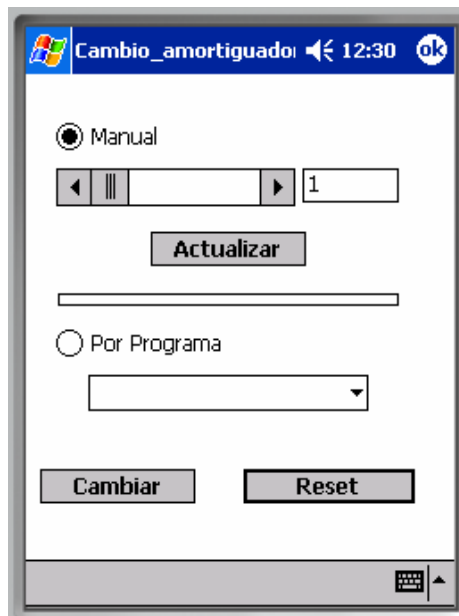


Figura 14

Se puede observar que en esta primera versión únicamente se tiene la opción de cambiar todos los motores de posición a la vez. La opción de cambiar mediante programas no está operativa, y el número de posiciones del motor no puede ser cambiado.

Como ya se ha dicho, ésta era una primera versión. El objetivo de esta versión era con el control de la comunicación serie ya afianzado, el poder cambiar el motor de posición desde la PDA indicándole por el puerto serie a la tarjeta la posición a la que debería mover el motor.

#### **4.3.1. Configuración del puerto serie**

Evidentemente, para esta primera aplicación antes se debía haber adquirido el suficiente control por el puerto serie, como para establecer comunicaciones eficientes. Para lograrlo, se crearon una serie de pequeñas aplicaciones cuya única misión era la de ejercer de “chat” entre la tarjeta y la PDA. El motivo es que si se pueden intercambiar *grandes* cantidades de datos sin pérdida de ninguno de ellos, el envío de pequeñas informaciones no debería resultar dificultoso.

Aunque pueda parecer relativamente sencillo, el control de las comunicaciones por el puerto serie en la PDA, no es demasiado sencillo en un primer momento. Como ya se ha dicho, el hardware equipado en esta clase de dispositivos no es de muy buena calidad, por lo que es muy sensible a la configuración, además el software tampoco ayuda.

La cantidad de parámetros a configurar en un puerto serie es bastante amplia. Velocidad de transmisión, paridad, BIT de parada, control del flujo... sin embargo, ya sea por cuestiones de hardware o de software la mayoría de los parámetros quedan fijados, con el resultado de que únicamente la velocidad de transmisión termina resultando el único parámetro que puede ser modificado sin ningún tipo de problemas.

La configuración final (y fija) que se ha hecho del puerto serie se indica en la Tabla 6.

---

Velocidad de transmisión	Variable
Paridad	Ninguna
BIT de parada	1
Tamaño de byte	8
Control DTR	No
Control RTS	No
Control CTS	No
Control DSR	No
Tamaño buffer entrada	100
Tamaño buffer salida	100
Carácter inicio	0x11
Carácter parada	0x13
Binario	Sí
Chequeo de la paridad	Sí
Control XON/XOFF en recepción	No
Control XON/XOFF en salida	No

**Tabla 6**

A continuación se explican cada uno de los parámetros:

- Velocidad de transmisión: indica el número de BITS enviados por segundo. Puede ir de 1200 a 19200 baudios
- Paridad: mecanismo de control de los datos, si se establece paridad par, el byte recibido debe ser par, si se establece como impar, el byte recibido será impar. Si no se cumple la condición el byte recibido es erróneo. Trabajamos sin paridad, es decir, sin verificar los bytes que llegan.
- BIT de parada: se indica el final del envío de los datos con un último BIT.
- Tamaño de Byte: especifica el número de BITS que forman un byte en la transmisión

- Control DTR/RTS/CTS/DSR: son diferentes métodos para controlar el flujo durante la transmisión. Necesitan una línea propia para poder funcionar, por lo que al disponer sólo de tierra y los canales de transmisión y recepción de datos no los usaremos.
- Control XON/XOFF: otro tipo de control de flujo, si embargo debe estar desactivado
- Tamaño del buffer de entrada/salida: indica el máximo número de bytes que serán almacenados en el buffer.
- Carácter inicio/parada: en el control XON/XOFF indican el inicio y final de la transmisión, aunque se les da un valor no se usan por estar desactivado el control XON/XOFF
- Binario: la transmisión de datos por el puerto serie se puede hacer en modo binario o texto, la PDA sólo soporta modo binario
- Chequeo de la paridad: se indica si se quiere o no chequear la paridad, como no se utiliza paridad es indiferente activarlo o no.

De todos estos parámetros, sólo la velocidad de transmisión, BIT de parada y paridad pueden ser ajustados a voluntad, el resto vienen ya fijadas por cuestiones de hardware o por el propio sistema operativo. Además se ha comprobado que tan solo con un BIT de parada y sin paridad funcionan las transmisiones, por lo que al final resulta que únicamente se puede variar la velocidad de transmisión.

El problema reside en que los sistemas operativos y el resto de dispositivos programables usan código ASCII, mientras que Windows Ce usa UNICODE. LA diferencia estriba en que ASCII es un código en el que se usa un byte por carácter y en UNICODE se usan dos bytes por carácter. Si se elige cualquiera de las paridades u otro tipo de BIT de parada (1,5 o 2), la PDA es incapaz de transformar los bytes en ASCII que le llegan a UNICODE y da lecturas erróneas.

Como puede verse, el puerto serie termina siendo muy rígido, y se limita mucho sus posibilidades si se quiere usar para según qué aplicaciones. Todo ello curiosamente para la lectura, porque en todo lo referente a la escritura no ocasiona ningún tipo de problema.

Las pruebas además de con la tarjeta, se realizaron entre el PC y la PDA usando Hyperterminal como software de prueba al ser comercial, y por tanto de confianza en su funcionamiento.

Lo que sí sucede a veces y es un problema del sistema operativo, es, que se clicca en un botón por ejemplo, y se ve como se clicca, se hunde el botón y vuelve a salir, pero sin embargo no se llama a la función encargada de gestionar el clic de ese botón, por lo que a veces puede dar lugar a errores.

El siguiente paso una vez controlado el puerto serie y el cambio de posición del motor, fue la aplicación ya definitiva, en la que además del cambio de amortiguador, se incluye el poder ver el estado de las entradas y salidas digitales, las entradas analógicas, y la configuración de todos los parámetros del programa.

En el capítulo dedicado al manual del programa se verá en más detalle todo lo referente a su uso, pero se cree conveniente explicar al menos un poco del programa en esta sección.

#### **4.3.2. Programa final**

El nombre elegido para el programa es CommCarPda ( Communication Car PDA). Los idiomas disponibles para el programa son español e inglés. Al iniciarse el programa, nos encontramos con una pantalla similar a la de Figura 15.



Figura 15

En ella podemos observar que tenemos cuatro botones, Configuración, Cambio Amort., Gráfica, Digitales. Además clicando sobre las suspensiones de la imagen inferior se va directamente a la pantalla para cambiar de esa suspensión.



Figura 16

### 4.3.3. Configuración

En el formulario de configuración podremos cambiar todos los parámetros con los que funciona el programa. Como son por ejemplo: velocidad de transmisión, medio de transmisión, seleccionar los programas de la suspensión, el número de posiciones del amortiguador y muchas otras. Todos estos parámetros se guardan en un archivo de texto que puede ser editado fácilmente, con lo que se pueden modificar sin necesidad del software o comprobar que todo es correcto, y en su parte más útil y por lo que se decidió a hacerlo así, porque así se tiene la posibilidad de crear distintas configuraciones para distintas situaciones sin tener que estar cambiando continuamente los parámetros.

### 4.3.4. Cambio posición amortiguador

El cambio de la posición del amortiguador no nos resultará desconocido después de haber visto su primera versión. La interface gráfica es similar a la primera versión, con la particularidad de que ahora tenemos la posibilidad de elegir qué amortiguador cambiar.

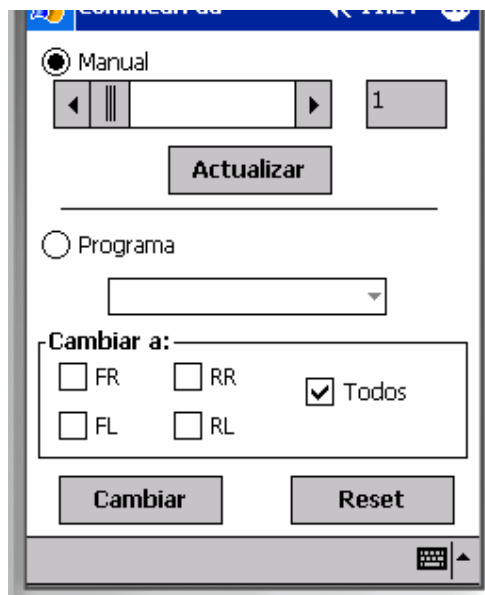


Figura 17

Al clicar en la imagen llegaremos a una ventana parecida a la del cambio de posición del amortiguador. Difieren en dos cosas, la primera en que ya no se elige qué amortiguador cambiar porque se ha hecho al seleccionar gráficamente la suspensión, y

otra y quizás más importante, en que al cargarse la ventana, primero solicita a la tarjeta que le envíe la posición actual que ocupa dicho amortiguador. Por ello tarda unos segundos en aparecer.

La diferencia pues entre ellas, es que el cambio de amortiguador es general, mientras que sobre la imagen de la suspensión nos lleva a ella en particular. Por ser general el cambio de posición del amortiguador no puede recuperar la posición en la que se encuentran, pues cada suspensión podría estar en una diferente.

#### **4.3.5. Digitales**

Tanto en el apartado de gráficas como en las entradas y salidas digitales, se había previsto que se fuesen actualizando continuamente, sin embargo debido a los problemas surgidos con el puerto serie, su lentitud, y la fuerte carga gráfica que supone para la PDA, al final se ha descartado la idea, dejando únicamente que los estados de las entradas y salidas digitales se actualicen a petición del usuario.

Lo que sí se ha mantenido, es que el programa aunque no actualiza por pantalla, guarda los datos recibidos en un fichero. Este fichero es de texto plano, por lo que puede ser fácilmente importado por cualquier hoja de cálculo.

El formato es el siguiente: los 24 primeros bytes son los correspondientes a los estados de las entradas digitales, 1 para alto (+5V) y 0 para bajo (0V). Como separador se utiliza la barra vertical “|” y a continuación vienen las 16 salidas digitales. Con el mismo criterio que para las entradas, 1 para alto y 0 para bajo. Cada línea indica una recepción, que se produce cada segundo. En caso de error se escribirá el mensaje de error en esa línea.

Si el usuario desea actualizar los estados de las entradas y salidas digitales dispone de un botón para ello, así se reduce la carga que el programa exige a la PDA. Además clicando en los indicadores de estado de las salidas digitales se puede cambiar su estado.

#### **4.3.6. Gráfica**

Al igual que en digitales, en la gráfica los datos se reciben cada segundo, en este caso al tratarse de una gráfica, sí se actualizan los datos y además se guardan.

El número de gráficas puede oscilar entre uno y tres, se ha hecho así por problemas de espacio, ya que la pantalla no es muy grande. Así que en el archivo de texto en el que se guarden los valores de las gráficas, habrá entre una y tres columnas separadas por una barra vertical "|". El valor indicado sería el valor en voltios medido en la entrada analógica.

Tanto en digitales como en gráfica se comienza a recibir datos cuando se abren y se manda la orden de parar cuando se cierra.

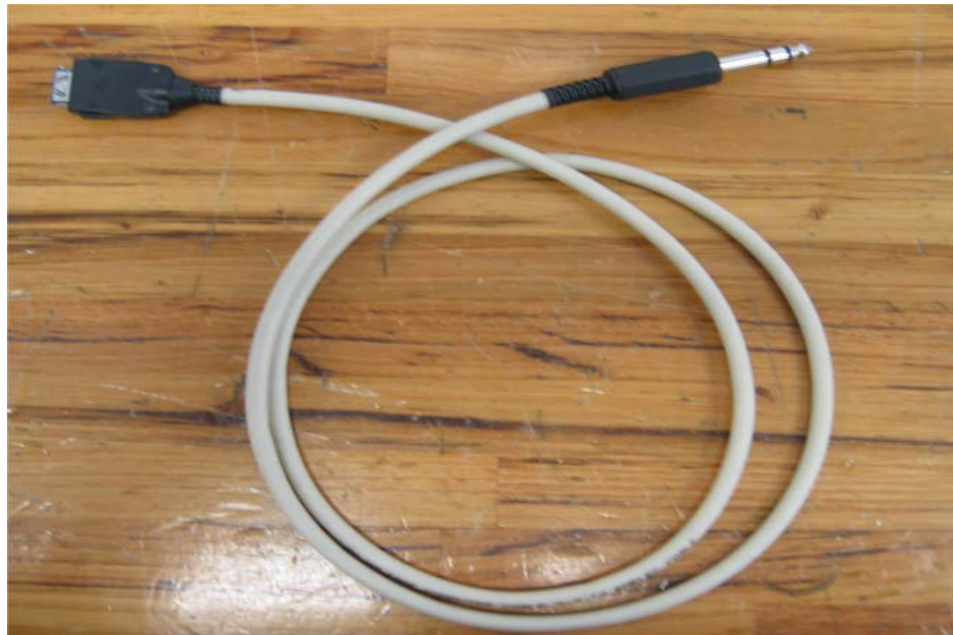
Actualmente sólo se utiliza la conexión serie, pero como se dijo anteriormente existen otras posibilidades. Éstas han sido tenidas en cuenta en todo momento y por ello se ha dejado preparado el software para la conexión utilizando otros medios y también en el código, con lo que únicamente se deberían programar las rutinas encargadas de la gestión de estos sistemas de comunicación.

## 5. MANUAL DEL USUARIO

Trataremos a continuación de explicar con detenimiento el modo de funcionamiento y utilidades de las que dispone este programa, y en su conjunto todo el sistema, tanto la PDA como la electrónica que se comunica con el vehículo.

Cuando hablamos del programa nos referimos a la PDA, pues el programa que se ejecuta en la electrónica implementada del vehículo no es interactivo con el usuario mediante una interface gráfica, sino que todas las acciones a realizar se harán a través de la PDA que una vez analizadas le indicará a la electrónica del car-cross qué debe hacer.

En la versión actual, la única vía de comunicación entre la PDA y la electrónica del car-cross es mediante comunicación serie, por lo que para el correcto funcionamiento del programa, y para que se puedan llevar a cabo las tareas deseadas, deberemos asegurarnos que ambas están conectadas. Para ello utilizaremos el cable especial que se muestra en la Figura 18.



**Figura 18**

Este cable dispone en uno de sus extremos de un conector tipo Jack, grande, de tres canales, que se conecta a la caja en la que va colocada la electrónica que nos servirá de puente entre la PDA y el car-cross. El otro extremo está unido a un conector para la PDA, que irá colocado en la ranura inferior de que dispone la PDA.



Figura 19

**Nota:** puede ocurrir que al utilizar otros modelos de PDA o electrónica, cambien los conectores de forma y lugar de conexión, pero el sistema es idéntico y ambos dispositivos deben quedar correctamente conectados.

## 5.1. VENTANA PRINCIPAL

Para arrancar el programa, deberemos clicar una vez sobre el nombre o el icono del programa tal y como se ve en la imagen inferior.

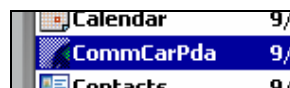


Figura 20

Una vez que lo hayamos hecho, aparecerá una ventana similar a la de la Figura 21. Nos encontramos ante la ventana principal. A partir de esta ventana empezaremos a movernos y será desde aquí desde donde elijamos las acciones a realizar.



Figura 21

Las diferentes opciones disponibles son:

- Configuración (1)
- Cambio del Amortiguador (2)
- Digitales (3)
- Gráfica (4)
- Cambio directo del Amortiguador (5)

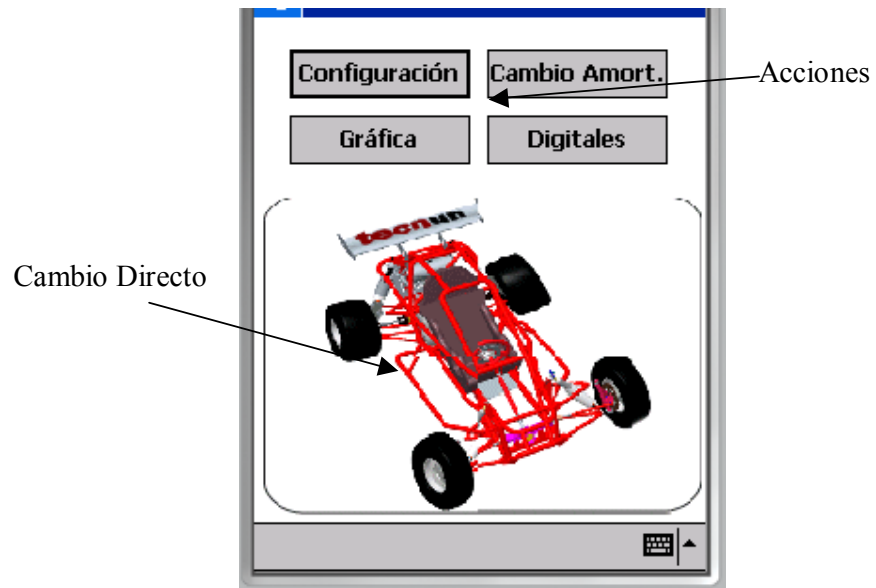


Figura 22

Clicando sobre cualquiera de las opciones disponibles accederemos a ellas, siguiendo un orden lógico, a continuación iremos viendo cada una de ellas.

## 5.2. CONFIGURACIÓN

Clicando sobre el botón llamado “Configuración” nos aparecerá una ventana similar a la siguiente:

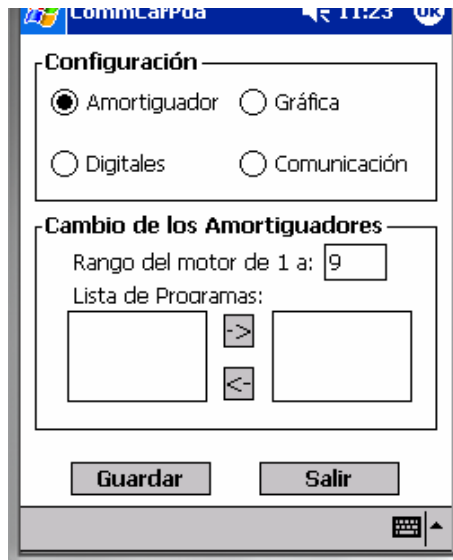


Figura 23

La función de esta ventana es la de permitirnos configurar el funcionamiento del programa y los parámetros que necesita para funcionar. A continuación veremos estos parámetros, cómo acceder a ellos y cómo modificarlos.

Primero podemos observar como la ventana se divide en tres partes claramente diferenciadas.

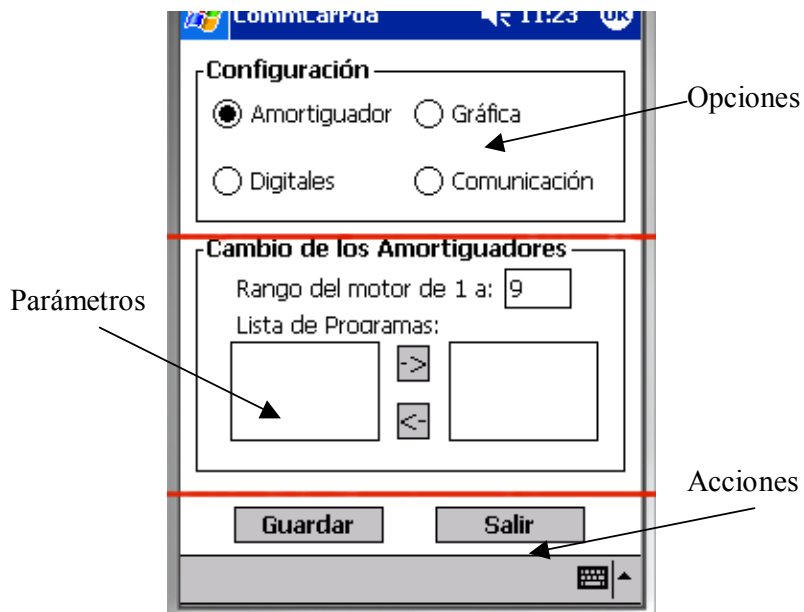


Figura 24

La primera, la superior, consta de 4 botones de radio con las siguientes opciones:

- Amortiguador
- Gráfica
- Digitales
- Comunicación

Cada una de estas opciones nos permiten modificar los parámetros correspondientes a dichas funciones. Por defecto está seleccionada la opción "Amortiguador".

La segunda parte, o central, enlaza con la primera, pues es aquí donde nos irán apareciendo las opciones de las que disponen cada función según la opción elegida en el cuadro superior.

Como cada acción que se puede realizar con el presente programa tiene unos determinados parámetros característicos, la apariencia que tenga el cuadro central según qué opción elijamos será diferente. Sin embargo no hay que preocuparse, pues los cambios realizados se guardan y no habrá que guardar cada vez que se quiere cambiar otros parámetros. Con esta solución conseguimos tener todos los parámetros en una sola ventana y no tener que cambiar continuamente.

Por último, podemos observar en la parte inferior que existen dos botones cuyo nombre es el siguiente: "Guardar" y "Salir". El botón "Guardar" nos permite guardar los parámetros en cualquier momento, de esta manera las opciones que hayamos seleccionado serán utilizadas durante el resto de la ejecución del programa y además se guardarán en un archivo de texto para ser cargadas de nuevo en la próxima ejecución del programa.

Si se han guardado correctamente los datos se nos avisará con un mensaje, del mismo modo, en caso de error nos aparecerá otro mensaje.

También se podrá clicar en cualquier momento en el botón “Salir”, como su nombre indica, hará que salgamos de la ventana actual y volvamos a la principal. Si hemos hecho algún cambio y no lo hemos guardado, antes de salir definitivamente se nos preguntará si deseamos guardar los cambios o no.

Si elegimos no guardar los datos, éstos se perderán y no se utilizarán en el programa, en caso afirmativo, se guardarán y se utilizarán durante la ejecución del programa. También se puede abandonar la ventana de configuración clicando sobre la esquina superior derecha, donde se lee “ok”.

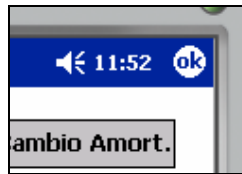


Figura 25

Una vez vista la disposición básica de los controles pasemos a analizar cada una de las opciones y sus parámetros.

### **5.2.1. Amortiguador**

Es la opción seleccionada por defecto, de cualquier modo podremos acceder a ella seleccionándola tal y como muestra la Figura 26.

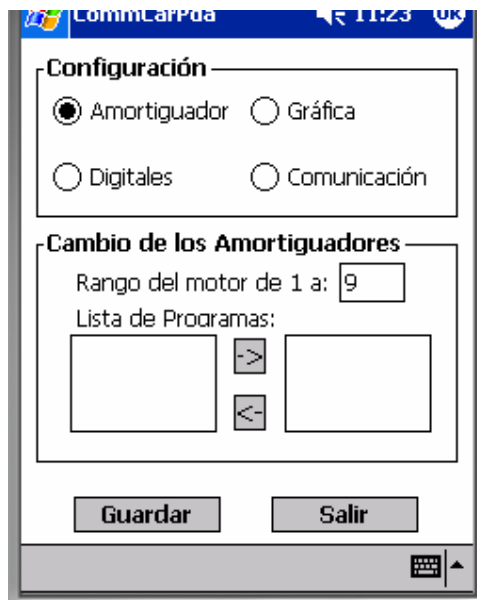


Figura 26

El cuadro central tendrá una forma similar a la inferior:

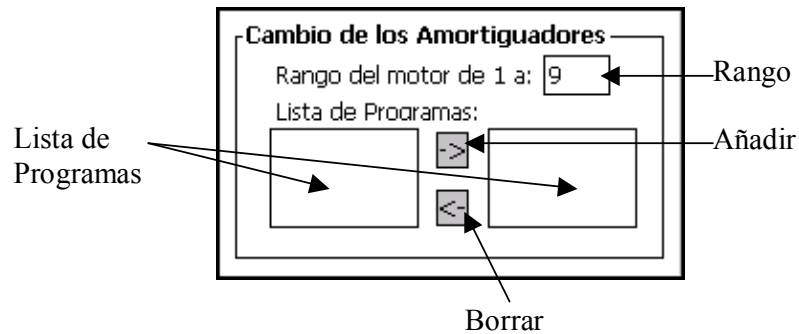


Figura 27

Como hemos podido observar en la imagen superior, disponemos de los siguientes parámetros:

- Rango del motor
- Lista de programas
- Añadir o quitar programas

El rango del motor se refiere al número de posiciones de que dispone el motor, por ello como mínimo tiene una, por eso se indica “de 1 a:” siendo el número que se escriba en la caja de texto el número de posiciones que tiene el motor. Por ejemplo, si dispone de 5 posiciones escribiremos 5, si son 10, un 10 y así sucesivamente.

**Nota:** para escribir en la caja de texto deberemos desplegar el teclado en pantalla si no disponemos de ninguno conectado a la PDA. Para ello clicaremos sobre la imagen del teclado en la esquina inferior derecha, y de nuevo para ocultarlo.

La caja de texto sólo admite números, por lo que todo aquello que no sea un número no será admitido, incluyendo puntos y comas. El número deberá ser estrictamente mayor que cero.

En “Lista de Programas” veremos dos listas, la de la izquierda contiene todos los programas creados para el funcionamiento del amortiguador. Dichos programas están contenidos en un fichero de texto llamado: “programas.txt” . Para borrar, añadir o cambiar de orden los programas, procederemos a editar dicho fichero y escribir, borrar o cambiar de posición el nombre del programa. Cada nombre de programa deberá ir en una línea diferente.

En la lista de la derecha aparecen los programas seleccionados, es decir, los que estarán disponibles para ser enviados a la electrónica del car-cross. Aunque nos referimos a ellos como programas, sería más correcto llamarlos leyes de control.

Para añadir o borrar un programa o ley de control de la lista de seleccionados, procederemos del siguiente modo: primero seleccionaremos un programa de la lista total y luego clicaremos sobre el botón añadir “->”. Para borrarlo, seleccionaremos el programa a borrar de las lista de programas seleccionados y clicaremos a continuación en el botón borrar “<-”.

Cuando se añade un programa a la lista de programas seleccionados, no se borra de la lista total, por lo que si tratamos de añadirlo dos veces se nos dará un mensaje de error.

Una vez visto cómo funciona y cuáles son los parámetros de la opción amortiguador pasemos a la siguiente.

### 5.2.2. Gráfica

La ventana de “Gráfica” es más sencilla aún que la de “Amortiguador”. Para acceder a ella, la seleccionaremos clicando sobre ella.

Disponemos aquí de tres botones seleccionables así como de otras tres cajas de texto. Cada botón seleccionable activa la petición de dibujo de su gráfica correspondiente. El canal o entrada analógica a graficar es la indicada por el número en la caja de texto.



Figura 28

El máximo número de gráficas que se pueden dibujar es tres, siendo su mínimo de cero, con lo que no se nos dibujará nada.

### 5.2.3. Digitales

Eligiendo la opción “Digitales” nos aparece la siguiente ventana:



Figura 29

En ella podemos observar dos listas y cuatro botones. Su misión es seleccionar las salidas digitales que serán bloqueadas y que por tanto no se podrá cambiar su estado durante la ejecución del programa.

Añadir o borrar una salida es idéntico a añadir o borrar un programa. La única salvedad es que aquí disponemos de dos botones más. El botón “Todas” que bloqueará todas las salidas digitales, y el botón “Ning” que borrará todas las salidas digitales bloqueadas.

La función del bloqueo de las salidas digitales se verá más adelante en su apartado correspondiente.

#### 5.2.4. Comunicación

La última opción disponible es “Comunicación”. También es la única cuya configuración afecta a todas las demás opciones, pues es en la que elegimos el medio de comunicación y la velocidad de transferencia de datos.



Figura 30

Como medio de comunicación disponemos de:

- Serie
- Bluetooth
- Ethernet

Serie como su nombre indica, se refiere a la comunicación vía puerto serie y que es la única disponible en esta versión. Sin embargo en vistas a futuras versiones se han dejado planteadas otras formas de comunicación.

Como no se puede elegir otra forma de comunicación que no sea el puerto serie, la única opción que queda por elegir es la velocidad de transmisión. Ésta va desde 1200 baudios hasta 19200 baudios.

---

Por último decir que el programa al iniciar su ejecución carga la configuración guardada, por lo que en caso de error se asigna una por defecto, la cual tiene la siguiente configuración:

1. Rango del motor: 9
2. Programas seleccionados: ninguno
3. Gráficas seleccionadas: ninguna
4. Salidas digitales bloqueadas: ninguna
5. Comunicación: serie
6. Velocidad: 1200 baudios

La configuración se guarda en un archivo de texto llamado: "parametros.txt". Ese debe ser el nombre de la configuración a cargar. Al ser un archivo de texto puede ser fácilmente editado, copiado y modificado, lo que nos permitirá disponer de varios archivos con diferentes configuraciones para cada situación o vehículo.

En esta versión no se permite elegir el archivo de configuración a cargar, así que el archivo o la configuración que se desee cargar deberá estar en un archivo llamado "parámetros.txt".

Si este archivo no es correcto se asignará la configuración por defecto.

### **5.3. CAMBIO AMORTIGUADOR**

El "Cambio de Amortiguador" lo que nos permite es cambiar el coeficiente de amortiguamiento del amortiguador, que al final se traduce en una posición del eje del amortiguador.

Para acceder a esta posibilidad basta con clicar sobre el botón llamado "Cambio Amort."

La ventana es similar a la Figura 31.



Figura 31

Si la ventana de configuración podríamos dividirla en tres partes, en esta ocasión la división se realiza en cuatro.

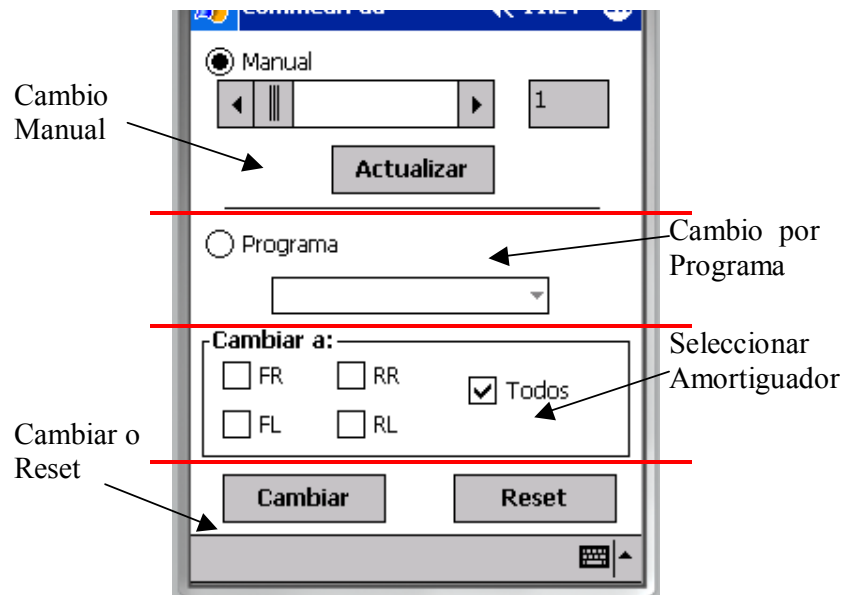


Figura 32

A su vez las dos partes superiores están relacionadas, podemos ver como en cada una de ella disponemos de un botón de radio, ambos son excluyentes, por lo que si se selecciona uno el otro que deseleccionado.

El botón denominado “Manual” activa tanto la barra de desplazamiento, la caja de texto y el botón “Actualizar”, y desactiva la lista despegable situada en la segunda división. La opción “Programa” actúa de modo inverso.

Cuando la opción “Manual” está activa, significa que seremos nosotros los que llevemos al eje a la posición querida, esto se hace moviendo a derecha e izquierda la barra de desplazamiento. Veremos como cambia el contenido de la caja de texto, este número indica la posición en la que se encuentra la barra de desplazamiento, y que será por tanto la posición a la que moveremos el eje.

El número de posiciones de la barra de desplazamiento vendrá determinado por el rango del motor.

Si se clic en el botón actualizar, nos llevará la barra de desplazamiento y nos cambiará el texto a la posición actual en la que está situado el eje.

Con la opción “Programa” seleccionada, podremos elegir uno de los programas o leyes de control, que previamente habíamos elegido en la ventana de configuración, y enviárselas al car-cross.

En la tercera división, nos encontramos con cinco botones seleccionables que nos ofrecen las siguientes posibilidades:

- FR
- FL
- RR
- RL o
- Todos

El significado de cada una de las abreviaturas se indican en la Tabla 7.

Abreviatura	Nombre
FR	Front Left
FL	Front Right
RR	Rear Right
RL	Rear Left

Tabla 7

“Cambio de Amortiguador” nos permite un cambio general de la posición del eje del amortiguador, por lo que deberemos elegir el amortiguador que queremos cambiar. Podemos elegir cualquiera de los cuatro o varios de ellos. Si elegimos los cuatro, se deseleccionarán los botones y se seleccionará la opción “Todos”. Si esta opción está seleccionada y queremos únicamente elegir alguno de ellos, primero deberemos deseleccionarla, pues no tiene sentido elegir un amortiguador si ya están elegidos todos.

Al clicar sobre el botón “Cambiar” mandaremos la orden a la electrónica del car-cross de que cambie los amortiguadores elegidos a la posición seleccionada o bien le indica que ley de control deben seguir.

Por el contrario, si clicamos sobre “Reset”, lo que produciremos es un reset de los amortiguadores seleccionados. En ambos casos se nos dará un mensaje de confirmación. El reset en nuestro caso lleva al amortiguador hasta su posición 1, por lo que la barra y el texto indicarán que el amortiguador está en esa posición.

Para abandonar esta ventana y volver de nuevo a la principal deberemos clicar en el símbolo de “ok” que se encuentra en la esquina superior derecha.

#### **5.4. GRÁFICA**

En esta sección, lo que veremos es la representación gráfica de las entradas de tensión que hayamos seleccionado en la ventana de configuración. Por tanto, lo que

veremos será, dependiendo del número de gráficas, una serie de ejes coordinados en lo que se irá dibujando la tensión.

La velocidad de refresco es de un segundo aproximadamente. Esta velocidad está fijada en la electrónica por lo que no se puede modificar desde la PDA, al menos en esta versión. El número de segundos a representar es de 10.

Se comienza a recibir datos desde el mismo momento en el que se abre la ventana y se finaliza cuando se cierra, por tanto no es necesario pulsar ningún botón.

Además el estado de las entradas analógicas de la electrónica se guardan en un fichero de texto llamado: "AnalogState.txt", en dicho fichero se guarda el valor de la tensión de cada entrada, estando separados los valores por una barra vertical "|". Si se detiene la ejecución y se vuelve a reanudar el fichero anterior será borrado y sustituido por el nuevo, por lo que antes de ejecutar dos veces el dibujo de gráficas será necesario guardar el fichero anterior, siempre y cuando se quieran conservar los datos anteriores.

Como en ocasiones anteriores para salir deberemos clicar en el símbolo de "ok" situado en la esquina superior derecha.

## **5.5. DIGITALES**

Del mismo modo que en "Gráfica", la ejecución es automática, sin embargo, a diferencia de ésta aquí no se dibuja nada, sino que nos encontramos ante una ventana con el siguiente aspecto:

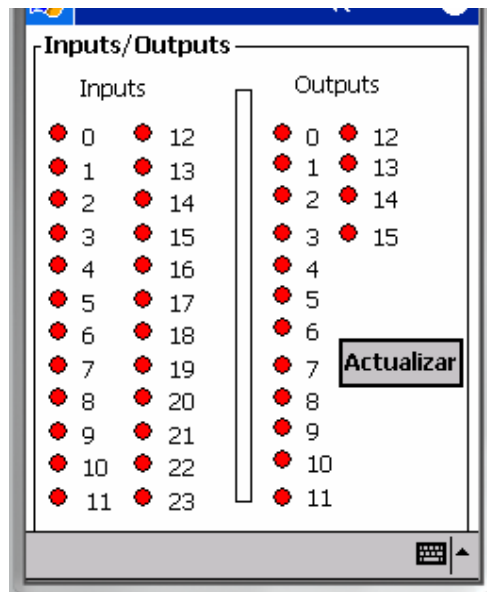


Figura 33

En ella vemos cuatro columnas con unos indicadores circulares y un número a su lado. El número nos indica la entrada/salida de la que se trata, y el indicador circular su estado, rojo para bajo (0V) y verde para alto (+5V).

Si queremos ver el estado actual de las entradas y salidas digitales deberemos pulsar sobre el botón de "Actualizar". Si bien para actualizar (visiblemente) el valor de las entradas y salidas necesitamos clicar en el botón, como sucedía en "Gráfica" dichos estados son guardados de forma automática en un fichero de texto llamado: "DigitalState.txt".

Ahora no tendremos el valor de la entrada analógica o de tensión escrito en el fichero, sino el estado en el que se encuentra. 1 para un estado alto y cero para bajo. De tal manera, que los 24 primeros valores corresponden a las entradas digitales, y separados de éstos por una barra vertical "|" los 16 restantes son los valores de las salidas. También se actualiza cada segundo aproximadamente, y cada vez que se ejecuta se pierde la información anterior. De la misma forma que en "Gráfica".

Por último, en las salidas digitales se permite cambiarles el estado (alto o bajo), desde la PDA, para ello sólo se deberá clicar sobre el indicativo del estado, y si no es una salida que se haya bloqueado anteriormente, se mandará la orden para su cambio.

Igualmente que en “Gráfica”, al cerrar la aplicación se detendrá de manera automática el chequeo de los estados.

## **5.6. CAMBIO DIRECTO DEL AMORTIGUADOR**

Por último, veremos el “Cambio directo del Amortiguador”. Su funcionamiento es similar a “Cambio Amortiguador”, con la salvedad de que en vez de poder cambiar cualquiera o todos los amortiguadores, sólo podremos cambiar el que hayamos elegido.

Para ir a esta función, deberemos elegir la suspensión a modificar clicando sobre ella en la imagen del coche de la parte inferior de la ventana principal, tal y como se muestra en la Figura 34.

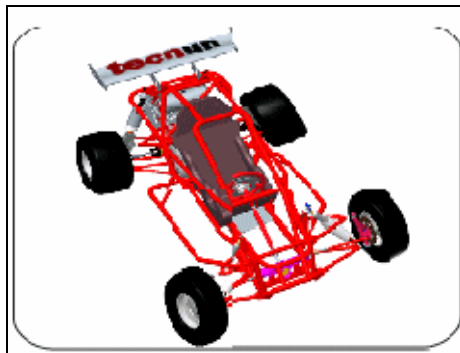


Figura 34

Pasados unos segundos se nos aparecerá una ventana igual que la Figura 35.



Figura 35

Como puede observarse es idéntica a “Cambio Amortiguador” menos en la elección de suspensión, además si nos fijamos veremos que a diferencia de “Cambio Amortiguador”, la posición inicial de la barra de desplazamiento no está a la izquierda del todo. En esta ocasión se encuentra en la posición en la que se encuentra el eje del amortiguador elegido.

Es por ello por lo que debemos esperar unos instantes antes que se abra, ya que debe pedir la información a la electrónica y esperar respuesta. Esto no se hace en “Cambio Amortiguador” debido a que es más general, y a que como hemos visto cada una de las suspensiones puede tener una configuración diferente, por este motivo no se puede determinar una posición para todas ellas.

El resto de su funcionamiento es similar al ya visto y no tiene ninguna diferencia, salvo que en el nombre del cuadro en el que están contenidos los controles aparece la abreviatura de la suspensión elegida. El sistema es idéntico al de la Tabla 7.

Con esto ya hemos visto todo lo referente al programa, esperamos que este pequeño manual haya sido de ayuda.

## 6. MANUAL DEL PROGRAMADOR

No nos detendremos aquí a explicar línea por línea lo que realiza el código escrito, tanto para la PDA como para la tarjeta, sino que se darán ciertas notas e indicaciones de cuales son las funciones que realizan, que proceso siguen y como se ha planteado la programación.

Por tanto lo que se busca es que el programador que lo necesite, pueda leer el código fuente incluido en los anexos junto a esta pequeña descripción para que la lectura y comprensión sean más rápidas y fáciles.

Para empezar, debemos darnos cuenta de que el sistema se divide en los partes diferenciadas, el car-cross con la electrónica y la PDA, por tanto tendremos dos programas diferentes, por una parte el que corresponde a la electrónica y por la otra el que corresponde a la PDA.

### 6.1. ELECTRÓNICA

Por ser el más corto y sencillo, empezaremos por la electrónica.

El programa se divide en dos partes claramente diferenciadas, por una parte tenemos las declaraciones de funciones y variables y por la otra, dentro de la función “*main()*”, el bucle infinito que se encarga de reconocer las peticiones del usuario.

Empezaremos pues por las declaraciones de las variables y su misión, para seguir después con la funciones y terminar con el bucle infinito.

#### 6.1.1. Variables

Se definen las siguientes variables con las siguientes funciones:

- #define HICONFIG 0xffff → se activan todas las salidas digitales.
- #define TIMEOUT 20UL → tiempo máximo que se tarda en leer un byte por el puerto serie.

- `#define BINBUFSIZE 63` → tamaño del buffer de entrada.
- `#define BOUTBUFSIZE 63` → tamaño del buffer de salida.
- `#define FL 0` → se asocia la suspensión delantera izquierda con el cero, es la posición que ocupan en los diferentes arrays sus salidas digitales.
- `#define FR 1` → se asocia la suspensión delantera derecha con el uno, es la posición que ocupan en los diferentes arrays sus salidas digitales.
- `#define RR 3` → se asocia la suspensión trasera derecha con el tres, es la posición que ocupan en los diferentes arrays sus salidas digitales.
- `#define RL 2` → se asocia la suspensión trasera izquierda con el dos, es la posición que ocupan en los diferentes arrays sus salidas digitales.
- `#define NumDigOut 16` → número de salidas digitales.
- `#define NumDigIn 24` → número de entradas digitales.
- `#define High 1` → alto equivale a 1.
- `#define Low 0` → bajo equivale a 0.
- `char msg[5]` → se declara un array de caracteres de tamaño 5, contendrá el mensaje que llegue por el puerto serie.
- `char display[128]` → se crea un array de caracteres que contendrán el texto a mostrar por pantalla.
- `char channels[16]`
- `int move, time_reset, position, psn_req, output_status`
- `int time, frequency, channel, cycles`
- `int output_level, level_dir, level_power` → enteros que indican el estado que deberán tener las salidas de alimentación, dirección y pulso. Alto o bajo (1 o 0).
- `unsigned int outputChannel`
- `int *chpower[3]` → array de enteros que contiene las salidas de alimentación que se van a activar.
- `int *chdir[3]` → array de enteros que contiene las salidas de dirección que se van a activar.
- `int *chclock[3]` → array de enteros que contiene las salidas de pulso que se van a activar.
- `const int channel_power[4] = {0,3,11,12}` → salidas digitales que activan la alimentación.
- `const int channel_dir[4] = {1,5,7,10}` → salidas digitales que indican la dirección.
- `const int channel_clock[4] = {2,6,9,13}` → salidas digitales que mandan el pulso.

- `const int channel_total_power[1]= {15}` → salida digital que activa la alimentación a todos los relés.
- `const int velocity[6] = {1200, 2400, 4800, 9600, 14400, 19200}` → array que contiene las posibles velocidades.
- `int readedbytes` → su contenido es el número de bytes leídos.
- `int positionFL, positionFR, positionRR, positionRL` → guardan la posición de cada amortiguador.
- `char InputState[NumDigIn]` → se guardan los estados de las entradas digitales como caracteres, '1' o '0'.
- `char DigState[NumDigIn + NumDigOut + 1]` → contiene los estados de las entradas y salidas digitales.
- `char OutputState[NumDigOut]` → se guardan los estados de las salidas digitales como caracteres, '1' o '0'.
- `int DigRequest` → indica si hay petición o no de las entradas y salidas digitales.
- `int AnalogRequest` → indica si hay petición o no de las entradas analógicas.
- `int NumGraf` → guarda el número de entradas analógicas que se deben leer.
- `int AnalogChannels[3]` → guarda las entradas analógicas que se piden.

Con esto se han visto las variables más importantes utilizadas en el programa de la tarjeta, a continuación veremos las funciones y qué realizan.

### **6.1.2. Funciones**

Seguidamente se listarán las funciones creadas en este programa con un pequeña descripción de su función y cómo lo hacen.

- `void DispStr(int, int, char*)` → muestra un mensaje de texto en pantalla y centrado.
- `void Reset4(int, int, int, int*[], int*[], int*[], int)` → resetea los amortiguadores, primero se les indica la dirección y luego se les envía una serie de pulsos para asegurarse de que han llegado hasta la posición más baja. Devuelve la posición como un entero.

- 
- `int FullReset4(int, int*, int*[], int*[], int*[], int)` → activa la alimentación de todos los motores y llama a la función “*Reset4*”, posteriormente desconecta la alimentación de los motores. Devuelve la posición como un entero.
  - `int Positionate4(int, int, int, int*[], int*[], int*[], int)` → coloca al amortiguador en la posición que se le indica. Para ello primero activa la alimentación, seguidamente calcula el número de posiciones que se debe mover. Calcula la dirección en la que se tiene que mover y se la indica a la tarjeta controladora. Por último lanza una serie de pulsos hasta que el motor llega a su posición y termina desconectando la alimentación. Devuelve la posición del motor.
  - `void Connect(int*)` → pone en alto la salida que se le indique.
  - `void Disconnect(int*)` → pone en bajo la salida que se le indique.
  - `void Total_Connect(int*)` → conecta la alimentación de todos los motores.
  - `void Total_Disconnect(int*)` → desconecta la alimentación de todos los motores.
  - `int Wait(int)` → detiene la ejecución del programa el número de milisegundos que se le indica. Devuelve un 1 como valor de retorno.
  - `void Indicate_4Direction(int, int*)` → pone la salida correspondiente a la indicación de dirección en el estado que indique el primer parámetro.
  - `void ChangeDumper(char param[], int bytesread)` → analiza la cadena de caracteres recibida por el Puerto serie una vez que se sabe que se quiere cambiar el amortiguador, y evalúa los amortiguadores que se tienen que cambiar y lo indica.
  - `int RetrieveDumper(char param[])` → analiza de qué amortiguador debe devolver la posición y se la devuelve a la tarea principal como valor de retorno.

- void ChangeState(int\* ichannel, int level) → pone la salida digital en estado que indique level, además actualiza sus estado en la lista de estados de las salidas digitales.

Finalmente se describirá la función de la tarea principal que se desarrolla en el bucle infinito, para ello se incluye el código abreviado.

```
serBopen(velocity[0]);

while(1) {
    costate{
        waitFor(readedbytes = serBread(msg, BINBUFSIZE,
TIMEOUT));
        switch(msg[0]) {
            case 'a':
                .....
            break;
            case 'd':
                .....
            break;
            case 'g':
                .....
            break;
            case 's':
                .....
            break;
        }
    }
}

costate {
    waitFor(Wait(1000));
    if(DigRequest == 1){
```

```
.....  
    }  
    else if(AnalogRequest == 1) {  
        .....  
    }  
} //Fin segunda tarea  
  
} //Fin bucle infinito  
serBclose();
```

Como puede observarse en el código anterior tenemos un bucle infinito que contiene dos tareas, estas tareas corresponden a:

- “Escuchar” el puerto serie hasta que le llega información y la analiza.
- Mandar el estado de las entradas/salidas digitales o bien las entradas analógicas.

Para ello se abre el puerto serie indicándole la velocidad y a continuación se entra en el bucle.

Si nos fijamos en la primera tarea (costate), veremos que no hace nada hasta que le llega información por el puerto serie, es entonces cuando pasa a analizar la información recibida, si no es ninguna de las posibilidades que se han programado, no hace nada y vuelve a quedar a la espera.

La segunda tarea primero espera un segundo, para no sobrecargar a la PDA en el puerto serie, y luego mira si tiene petición de los estados de las entradas o salidas digitales o son las entradas analógicas, si no tiene ninguna vuelve a quedar en espera otro segundo, en caso contrario realizará las acciones necesarias para cumplir la tarea.

Hasta aquí el código implementado en la tarjeta, se puede observar que se termina cerrando el puerto, sin embargo en estos momentos no es necesario puesto que

la única forma de salir del bucle infinito es apagar la tarjeta y por tanto esa línea no se ejecutará, sin embargo por seguridad se ha incluido.

## **6.2. PDA**

El programa de la PDA se inicia asociando a los botones que llevan al “Cambio directo de Suspensión” la imagen correspondiente y llamando después a la función miembro “CargarConfiguracion”.

### **6.2.1. CargarConfiguración**

Esta función se encarga exclusivamente de leer el archivo “parámetros.txt” y cargar por tanto la configuración guardada anteriormente.

Para ello lee byte a byte hasta que se encuentra un carácter de “intro” o salto de línea, momento en el que empieza a trabajar con esa línea, pues los parámetros son guardados por líneas, estando primero el parámetro seguido del símbolo “=” y viniendo a continuación el valor del parámetro.

Así que cuando se lee el carácter salto de línea lo que se está detectando es el final de línea y por tanto el parámetro con su valor. A continuación se pasa a separar el parámetro de su valor y una vez hecho esto, se analiza cual es el parámetro, en función de cual sea se hace una cosa u otra, pero siempre se termina asociando el valor del parámetro a alguna variable miembro de la clase principal para después utilizarlo.

Una vez que ha leído sin problemas el archivo de configuración, se queda a la espera hasta que el usuario clicca en alguno de los botones, momento en el cual se procede a pasarle todos los parámetros que necesite y se carga el nuevo formulario, quedando el principal oculto pero en memoria.

Cuando se cierra el formulario secundario se vuelve al principal y éste vuelve a quedar a la espera de las acciones del usuario. Con la única salvedad de “Configuración”,

---

pues cuando se cierra este formulario y se vuelve al principal se llama de nuevo a la función “CargarConfiguración” para actualizar los parámetros.

### **6.2.2. Configuración**

“Configuración” no tiene nada de especial en cuanto a código, simplemente se dedica a mostrar u ocultar los controles en función de qué parámetros quiere cambiar el usuario.

Lo más significativo es que cuando se cierra, analiza el estado de la variable booleana “bGuardado”, si el valor es verdadero, no se hace nada y se sale, por el contrario, si el valor es falso, se pregunta al usuario si desea guardar los cambios o no. Si elige guardar se guardan de la misma forma que cuando se clicca sobre el botón “Guardar”, si no se quieren guardar los cambios se sale y se pierden.

### **6.2.3. Cambio Amortiguador**

Este formulario tampoco tiene nada excesivamente complicado desde el punto de vista de programación.

Primero, al cargarse, asigna el rango de posiciones a la barra de desplazamiento en función del número de posiciones que se le indique desde el formulario principal. A continuación abre el puerto serie, comprueba que se ha abierto correctamente y llama a la función “Configuración”.

Esta función lo que hace es configurar el puerto serie con los valores que se han terminado estableciendo como los únicos con los que se puede tener una comunicación serie estable. El único parámetro que se puede variar en la configuración es la velocidad de transmisión. Este parámetro también se pasa como parámetro al formulario.

La función “Configuración” es similar para el resto de formularios que necesitan usar el puerto serie. El puerto se cierra en el momento en el que se cierra el formulario de cambio de posición del amortiguador y se vuelve a la ventana principal.

Por último, aunque difieren en los datos que mandan, tanto “Cambiar” como “Reset” funcionan del mismo modo, primero generan la cadena que tienen que mandar según el código ya establecido, y proceden entonces a mandarla.

El resto de funciones que contiene este formulario son simplemente auxiliares para dar funcionalidad gráfica al formulario.

#### **6.2.4. Cambio Directo de Amortiguador**

Este formulario funciona del mismo modo que “Cambio Amortiguador”, la única diferencia es que aquí no se puede elegir que amortiguador o amortiguadores cambiar, por tanto desaparecen las funciones relacionadas con esto, y que pide la posición del amortiguador seleccionado.

Por ello procede del mismo modo inicialmente, abre el puerto y comprueba que se ha hecho correctamente, seguidamente se configura y aquí llega la diferencia, una vez configurado el puerto, lo que se hace es mandar la petición a la tarjeta de que envía la posición del amortiguador. Se queda a la espera durante un tiempo determinado, actualmente de 2 segundos, y si antes de ese tiempo no ha llegado respuesta, se da un mensaje de error y se asigna una posición por defecto. Si llega respuesta se comprueba que sea correcta y en caso de que lo sea se coloca la barra de desplazamiento en la posición correcta.

Para saber que ha llegado una respuesta, lo que se hace es arrancar un hilo cuya única misión es quedar a la espera de la llegada de datos al puerto serie. Se hace en un hilo porque se hacerlo en el propio programa principal, éste quedaría bloqueado hasta que llegase algún tipo de información, y en caso de que no llegase dejaría de responder, por ello se debe hacer en un hilo.

Este hilo únicamente atiende a los avisos de llegada de datos y errores, en caso de error no hace nada pero si el mensaje recibido es la llegada de caracteres, lo que hace es llamar a una función específica que se encarga de leer lo que ha llegado al puerto serie, después de hacerlo queda de nuevo a la espera. Esto es así en general, sin embargo en este formulario como únicamente se lee una vez, una vez leído o bien si no llega respuesta, se detiene el hilo.

La función leer lo único que hace es leer el contenido del puerto y gestionar los posibles errores que pueda haber en la lectura o en el puerto, una vez que ha leído correctamente la información recibida, se la envía a otra función para que trabaje con ella y haga lo que necesite.

Salvo esta opción más que tiene este formulario el resto es exactamente igual al anterior.

### **6.2.5. Digitales**

De nuevo aquí empezamos tal y como veíamos en “Cambio Directo de Amortiguador”, se abre el puerto serie, se comprueba que es correcto, y seguidamente se lanza la petición de las entradas o salidas analógicas. También como en el formulario anterior se arranca el hilo encargado de escuchar el puerto serie.

Por tanto las funciones ya han quedado explicadas anteriormente. Aquí se añade el que cuando llega información, primero se comprueba que el número de bytes leídos sea el correcto, si lo es se procede a guardar los datos en el archivo “DigitalState.txt”, en caso contrario se guarda un mensaje de error en el archivo.

Si se ha clicado en el botón “Actualizar”, además de guardar los datos en el archivo, se actualizan los estados de las entradas y salidas digitales, siguiendo el criterio de rojo para estado bajo y verde para alto. En caso de que hubiese algún error se avisa al usuario de ello.

Por último, en el momento de cerrar el formulario se manda una cadena con el código que le indica a la tarjeta que se deje de enviar el estado de las entradas y salidas digitales y se vuelve al formulario principal habiendo cerrado previamente el puerto y dejándolo libre para posteriores usos, bien del mismo programa u otro.

### **6.2.6. Gráfica**

La única diferencia de “Grafica” con “Digitales” es que en vez de recibir los estados se reciben los niveles de tensión, esto en cuanto a la programación. Gráficamente son completamente diferentes, pero sus funciones y procedimientos son similares, por ello únicamente se explicará la diferencia existente entre ambos.

La diferencia estriba en que al iniciarse, “Grafica” dibuja los ejes coordenados de entre una y tres gráficas, y que dibuja el nuevo valor recibido por el puerto serie además de guardarlo, no como en el caso anterior que sólo se actualizaban los valores cuando lo pedía el usuario.

Por ello existe un pequeño algoritmo que se encarga de dibujar la gráfica cada vez que se reciben los datos, que es aproximadamente cada segundo.

Como ya se ha mencionado arriba, además de dibujar la gráfica, los valores de tensión en las entradas analógicas se guardan en el archivo de texto “AnalogState.txt”. Y como en el caso anterior, si la información recibida es errónea se escribe en el archivo, y de cara al usuario el valor recibido se asignará a cero.

Hasta aquí se ha visto en modo de resumen las funciones utilizadas y cual es misión y modo de funcionamiento, para más información remitirse a los anejos adjuntos en los que se encuentra el código fuente completo y comentado.

---

## 7. CONCLUSIONES, MEJORAS Y FUTURO

Al principio del documento se vio como uno de los objetivos era el sentar las bases para un futuro en la comunicación entre la electrónica del vehículo y la PDA.

Al concluir el proyecto fin de carrera y mirar atrás, se puede decir que este objetivo genérico se ha conseguido, pues no sólo se trababa de hacer que unos motores se movieran, sino también de que el sistema y el método fuera útil en futuras aplicaciones.

Así, al separar lo que es el desarrollo en la PDA del necesario en la electrónica del vehículo se abren muchísimas posibilidades pues podemos tener programas muy complejos funcionando en la PDA mientras que en la tarjeta los programas son relativamente sencillos, y mediante unos simple códigos poder hacer complejas operaciones.

Aquí nos hemos centrado en el cambio de posición de una amortiguador, pero como hemos visto también, al final se trata de mover un motor, por lo que cualquier sistema que necesite ser movido por un motor es susceptible de ser controlado con este método.

Por ello en este sentido, posibles líneas de investigación pueden ser:

1. Regulación del reparto de frenada
2. Control del acelerador electrónico
3. Control de marchas o regulación del cambio secuencial
4. Mediciones realizadas con las básculas

Además ahora se ha trabajado con una PDA, pero nada impide desarrollar un hardware propio para integrar el control en el volante, siendo desde éste desde donde se realicen los cambios. Pues la metodología a usar es idéntica.

Y por último, centrándonos en el propio sistema actual, algunas de las mejoras y futuras investigaciones podrían ser:

1. Comunicación inalámbrica
2. Ampliación de las funcionalidades del software
3. Optimización del código y/o protocolo
4. Comunicación entre PDA on-board y desde boxes
5. Telemetría

Todas ellas quedan ahora más cerca porque ya están las bases y la demostración de que sistemas relativamente complejos, pueden ser controlados de manera sencilla y efectiva.

## 8. PRESUPUESTO

En este capítulo se va a realizar un breve estudio económico del proyecto para cuantificar su coste. Las mediciones se han hecho sobre estimaciones considerando un periodo de trabajo de 7 meses.

Las licencias sobre los programas al ser educacionales se consideran sin coste, así como elementos como impresora, escáner u otros ya disponibles en el laboratorio y de uso común. Se excluyen también los amortiguadores regulables y los resortes de las suspensiones al estar ya disponibles.

Se listan a continuación las partidas con sus correspondientes unidades de medida:

Concepto	Unidad
Dirección del Proyecto de Joan Savall	Horas
Colaboración de Xavier Carrera	Horas
PDA	Unidad
Tarjeta Programable	Unidad
Conector PDA	Unidad
Conector tipo Jack	Unidad
Cable	Metros
PC	Horas
Licencia de Office	Horas
Cartucho Impresora	Unidad
Material de Oficina	Estimación
Alquiler Puesto de Trabajo	Horas
Gastos Generales (electricidad, teléfono, etc.)	Estimación

Tabla 8

Seguidamente se presenta el coste del proyecto desglosado en partidas con su correspondiente cantidad y precio unitario.

<b>Concepto</b>	<b>Cantidad (ud)</b>	<b>CosteUnitario (€/ud)</b>	<b>Coste Total (€)</b>
Dirección del Proyecto de Joan Savall	20	60	1200
Colaboración de Xavier Carrera	30	30	900
PDA	1	600	600
Tarjeta Programable	1	400	400
Conector PDA	2	2,5	5
Conector tipo Jack	1	2	2
Cable	1,5	1,5	2,25
PC	750	1	750
Licencia de Office	100	0,06	6
Cartucho Impresora	1	50	50
Material de Oficina			60
Alquiler Puesto de Trabajo	850	2,5	2125
Gastos Generales (electricidad, teléfono, etc.)			30
MO Ingeniero junior	850	12	10200
<b>Total</b>			<b>16930,25 €</b>

**Tabla 9**

El precio total de proyecto asciende a **Dieciséis Mil Novecientos Treinta euros**.

## 9. BIBLIOGRAFÍA

- 1.- Jon Bates, Tim Tompkins. “Descubre Microsoft Visual C++ 6”. Prentice Hall.
- 2.- Fco. Javier Ceballos. “Curso de Programación C/C++”. Ra-Ma.
- 3.- “Aprenda C++ como si estuviera en Primero”. TECNUN.
- 4.- Fco. Javier Ceballos. “Visual C++ aplicaciones para Windows”. Ra-Ma.
- 5.- David J. Kruglinski. “Inside Visual C++”. Microsoft Press, cuarta edición.
- 6.- Jan Axelson. “Serial Port Complete”. Lakeview Research.

Direcciones consultadas en Internet:

<http://www.jeanpaul.com.ar/>

<http://www.programacionutn.com.ar/>

<http://www.undersec.com/sjfproject/prog/dos/dos.htm>

<http://mundovb.net/mundoc/>

<http://www.elrincondelc.com>

<http://www.zworld.com>

<http://www.mipcdebolsillo.com>

<http://www.microsoft.com>

<http://www.pocketpc.com>

<http://www.codeproject.com>