ÍNDICE

PRÓLOGO

El objetivo del presente proyecto es la realización de un sistema capaz de realizar la comunicación entre un ordenador personal (PC) y un robot autónomo *(Microbot)* a través de radio frecuencia. Esta comunicación será en un solo sentido, es decir, del ordenador personal al *Microbot.* Como resultado del mismo, se dispondrá de un prototipo constituido por dos placas de circuito impreso, una emisora y otra receptora, que permita ilustrar el resultado de los diversos procesos intermedios.

La metodología utilizada parte de unas especificaciones, a partir de las cuales se deriva la división del objetivo final en bloques (módulos) funcionales perfectamente definidos en cuanto a entradas, salidas y funcionalidad.

Para estructurar de forma clara la presentación del trabajo realizado, la memoria se estructura en cinco capítulos. En el primer capítulo, se lleva a cabo una introducción de los conceptos teóricos que permitan la comprensión de los restantes apartados. En el segundo capítulo, partiendo de unos objetivos se llega como conclusión a unas especificaciones del sistema. El tercer capítulo, partiendo de dos grandes bloques (emisor y receptor), se consigue la división del sistema en los bloques funcionales comentados anteriormente. El diseño de estos módulos se detalla en el capítulo cuarto. Una vez construido el sistema es necesario una fase de pruebas y verificaciones, que se explican en el quinto capítulo. Por último, se describen los materiales empleados para la realización del prototipo, constituyendo un presupuesto con el coste final del mismo, así como un apartado donde se deja constancia las conclusiones y las posibles líneas futuras de desarrollo del sistema. Al final de la memoria se incluye la bibliografía empleada, así como dos anexos.

CAPITULO 1.INTRODUCCIÓN

1.1 Introducción a los *Microbots*

Los *Microbots* son un conjunto (una familia) de robots autónomos de reducidas dimensiones. A diferencia de los robots clásicos, los *Microbots* son encargados de una tarea en conjunto, es decir, a cada uno se le asigna una "subtarea" que debe de cumplir y además deben de cumplir la tarea de otro/s compañeros/s (*Microbots*), que por alguna razón dejan de funcionar.

El conjunto de *Microbots* debe tener la capacidad de suplir las posibles bajas, de tal manera que los *Microbots* restantes fueran cumpliendo sus propias "subtareas" y las de sus compañeros "estáticos". En el mejor de los casos el sistema "cae", cuando todas las unidades lo hacen simultáneamente.

Los robots clásicos, que en comparación se están ganando el apodo de autistas, están dejando sitio a sus pequeños hermanos.

Estos pequeños robots autónomos han ido tomando diferentes nombres desde su aparición *(Mobile Robots, Microbots*...) pero lo que no ha variado es su concepción.

El interés que estos sistemas a despertado, ha llevado a la construcción de granjas (habitaciones inteligentes que forman el universo de los *Microbots*, donde pueden ser programados, y altamente controlados).

Este campo, ha llamado la atención de profesionales que ven un alto grado de paralelismo entre las granjas y sus estudios. *[web1]*

1.1.1 La granja

Las granjas de *Microbots* están compuestas por habitantes. Todos los habitantes de granjas comparten alguna característica común, sin embargo, cada uno tiene su propia historia. Desde un principio, cada uno de ellos, nació para cubrir una necesidad determinada, por lo que tanto su hardware como su software es diferente.

1.1.2 *Microbot Tritt*

Es el *Microbot* objeto de este proyecto y forma parte de la granja de Microbótica. Su estructura es de Lego, está pensado para crecer hacia arriba, tal y como se muestra en la Figura 1. *[web2]*

Incorpora sensores de infrarrojos de corto alcance que le permiten por ejemplo, seguir una línea negra.

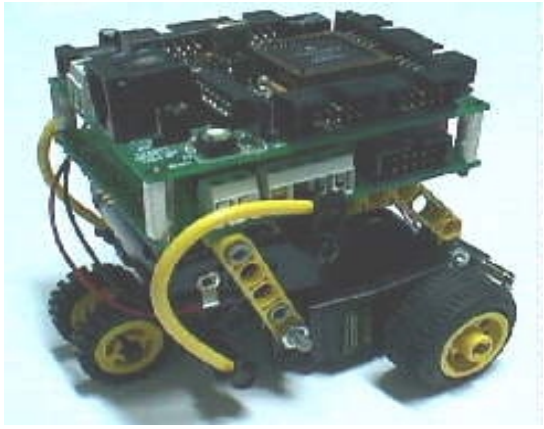


Fig.1

Lleva como sistema de control la tarjeta CT6811, que está basada en el microcontrolador 68HC11 y la tarjeta CT293+ como *driver* de potencia, para controlar dos servo motores.

Dadas las características de estas tarjetas, pueden funcionar en modo autónomo o bien conectado al PC, a través del puerto serie.

1.1.3 Necesidad de interfaz inalámbrica

Una de las clasificaciones que normalmente se utiliza para analizar a los *Microbots* es la llamada *Torre Bot.* Se trata de un modelo en torre, donde cada uno de sus niveles representa un paso en la fabricación de un sistema *Microbótico.* Se pueden distinguir seis niveles: Nivel Físico, Nivel de Reacción, Nivel de Control, Nivel de Inteligencia, Nivel de Comunidad y Nivel de Cooperación. Se describen a continuación dos de los niveles más importantes:

*Nivel de Control: Incluye los circuitos más básicos que relacionan las salidas de los sensores con las restantes unidades. Partiendo de una simple lógica digital hasta potentes *microcontroladores*, se *bus*ca dotar al *Microbot* de la capacidad para procesar la información obtenida por los sensores, así como actuar de una manera controlada sobre unidades motoras.

*Nivel de Cooperación: Comprende los sistemas donde a partir de un nivel de comunidad (granjas), se planifican o programan los *Microbots* para que tengan conocimiento de la existencia de otros, tal que posean la capacidad de cooperar para el buen desarrollo de una tarea.

Al incorporar sistemas inalámbricos de transmisión de datos (infrarrojos, radio...), tanto a los *Microbots* como a un ordenador central que los pudiera controlar, se consigue mejorar en el sistema estos dos niveles:

*En el Nivel de Control debido a que los *Microbots* suelen tener pequeña cantidad de memoria, se podría mejorar si se procesa la información procedente de los sensores en un ordenador central (i.e.), para ello y debido a la movilidad de los *Microbots* es necesario una comunicación inalámbrica.

*En el Nivel de Cooperación, esta cooperación se basa en su esencia en una comunicación entre los *Microbots, wire-less*.

1.2 Los sistemas de transmisión de datos inalámbricos

Los sistemas de comunicación de datos entre dos o más *Microbots*, son sistemas inalámbricos que no suelen requerir mucha potencia, a continuación se pasa a hacer un estudio más detallado de los dos sistemas más utilizados:

*Sistemas infrarrojos

*Módulos de radiofrecuencia

Ambos sistemas constan de un emisor y un receptor, según la Figura 2.

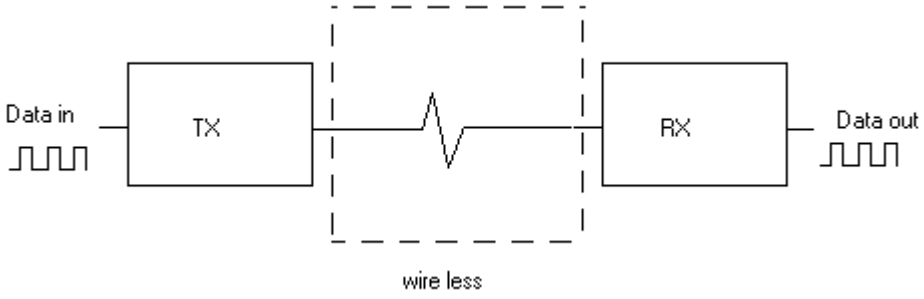


Fig.2

1.2.1 Sistemas de transmisión de datos infrarrojos

En los sistemas por infrarrojos, el módulo emisor consta de un diodo LED emisor de infrarrojos, el cual mediante un oscilador se hace oscilar a una frecuencia (frecuencia de portadora). Esta modulación se producirá o no en función de los datos a enviar (moduladora). Por ejemplo, para los '1' transmite, y no lo hace para los '0'.

En el receptor un diodo sensible a infrarrojos recibirá los impulsos del emisor. Al producirse la sintonía con el emisor se detectará un '1' y en otro caso un '0'.

Este tipo de sistemas presenta dos grandes inconvenientes: Por un lado, su corto alcance; por el otro su directividad, esto es, el emisor y el receptor deben estar perfectamente encarados para la sintonización, además no pueden existir obstáculos entre ellos.

Como ventaja es de destacar su precio y la facilidad de montaje de estos módulos, ya que requieren poca cantidad de componentes y están muy difundidos.

1.2.2 Sistemas de transmisión de datos de RF

Los sistemas de radiofrecuencia emiten ondas electromagnéticas a través de un medio natural. Si se emplean estos sistemas en la transmisión de datos, se obtiene un sistema de comunicación más fiable y robusto que en el caso de los infrarrojos.

Estos módulos de radiofrecuencia independientemente de su complejidad o técnica de modulación (AM, FM, FSK,...), subsanan los inconvenientes de los infrarrojos, esto es: se consiguen sistemas de transmisión mucho menos directivos, lo que permite una movilidad tanto del emisor como del receptor, sin peligro a que se pierda la comunicación. Además se permite la existencia de obstáculos entre los sistemas de TX (transmisión) y RX (recepción).

También se consigue un aumento del alcance; estos módulos pese a que sean de baja potencia consiguen unas distancias del orden de decenas de metros.

El precio que hay que pagar para obtener estas ventajas, es el coste de estos módulos (bastante mayor que los infrarrojos). Así como un aumento de la complejidad y de los ajustes (siempre en comparación con los sistemas infrarrojos).

Por todo lo comentado, y atendiendo a las necesidades que exige la comunicación entre *Microbots*, se ha optado por esta segunda opción (los módulos de transmisión de RF), para la realización del presente proyecto. Concretamente se emplearán dispositivos de baja potencia (LPDs), también llamados dispositivos de bajo alcance (SRDs). Estos sistemas se utilizan para la transmisión de datos y de control remoto sin licencia. Para ello utilizan el hueco en la banda de 70 cm (433 MHz).

Estos módulos tienen muchas aplicaciones, como por ejemplo, comunicaciones digitales entre un PC y una impresora, actuando como radio/enlace efectivo sin necesidad de un largo cable de RS-232. *[Elek.98]*

1.3 La tarjeta CT6811

1.3.1 Introducción a la CT6811

La CT6811 es una tarjeta para el desarrollo de aplicaciones hardware y software basadas en el microcontrolador 68HC11. No sólo sirve para el desarrollo de prototipos, sino también está pensada para funcionar en **sistemas terminados**. Se trata sobre todo de una tarjeta muy versátil, que se puede emplear como:

- **Tarjeta entrenadora:** Conectándose a ordenadores PC a través del puerto serie. Es posible enviar programas desde el PC a la CT6811 para que los ejecute. De esta manera, la tarjeta es muy útil para la depuración de programas en fase de pruebas y para aprender a programar el 68HC11 desde un punto de vista práctico: todos los programas creados sobre el "papel" se pueden ejecutar en un 68HC11 "de verdad".
- **Tarjeta autónoma de control:** La CT6811 funciona en modo autónomo, es decir, sin conectarse al PC. Cada vez que se encienda la tarjeta, se ejecutará el programa que previamente se habrá grabado en la memoria EEPROM del 68HC11. Conectando periféricos a través de los puertos de expansión, se consiguen desarrollar sistemas inteligentes de control: control de las luces de una casa, control de la maqueta de un tren, manejo de *Microbots*, alarmas...
- **Periférico inteligente del PC:** También es posible utilizar la tarjeta para comunicar el PC con el mundo exterior. La CT6811 actuaría como un periférico conectado al PC a través del puerto serie. Este periférico "inteligente" puede recibir órdenes del PC y actuar en consecuencia. Se pueden realizar aplicaciones del tipo: digitalización de señales y presentación en el PC, diseño de joysticks no convencionales, control de luces de la casa a través del PC, control de maquetas de tren visualizando en el PC la situación de los trenes, semáforos...

1.3.1.1 Características de la CT6811

- **Microcontrolador 68HC11:** Por ello incorpora todas las características de este micro:

    * 512 Bytes de EEPROM en los modelos A1 y A8.
    * 256 Bytes de memoria RAM interna.
    * Temporizador de 16 bits.

    * 3 capturadores de entrada.
    * 5 comparadores con salida hardware.
    * Un acumulador de pulsos.
    * Comunicaciones serie asíncronas.
    * Comunicaciones serie síncronas.
    * 8 canales de conversión analógico digital.
    * Interrupciones en tiempo real.
    * 4 puertos de E/S.

- ***Bus* de expansión:** *Bus* de expansión dividido en 6 puertos de 10 bits cada uno. Desde estos puertos se tiene acceso a todas las patas (*pines*) del micro.
- ***Switches* de configuración:** 2 *switches* de configuración del modo de arranque de la placa (*bootstrap*, *singlechip*, *expanded*, *special* test*)* y 2 *switches* disponibles para aplicaciones del usuario.
- ***Led* de pruebas conectado al bit 6 del puerto A.** Este *led* se puede conectar y desconectar por medio de un *jumper.*
- **Pulsador de *reset*/pruebas IRQ:** Pulsador para inicializar la tarjeta. Este pulsador se puede utilizar también, cambiando un *jumper*, como un pulsador de propósito general conectado a la entrada de interrupciones IRQ.
- **Niveles VRH y VRL** configurables a VCC y masa respectivamente mediante 2 *jumpers*.
- **Modos de funcionamiento** entrenador/autónomo configurables mediante un *jumper.*
- **Alimentación** a través de clemas o conector tipo *jack.*
- **Conexión directa al PC** por medio de un cable tipo teléfono.
- ***Reset* software y hardware** de la placa.

1.3.2 Diagrama de bloques de la tarjeta CT6811

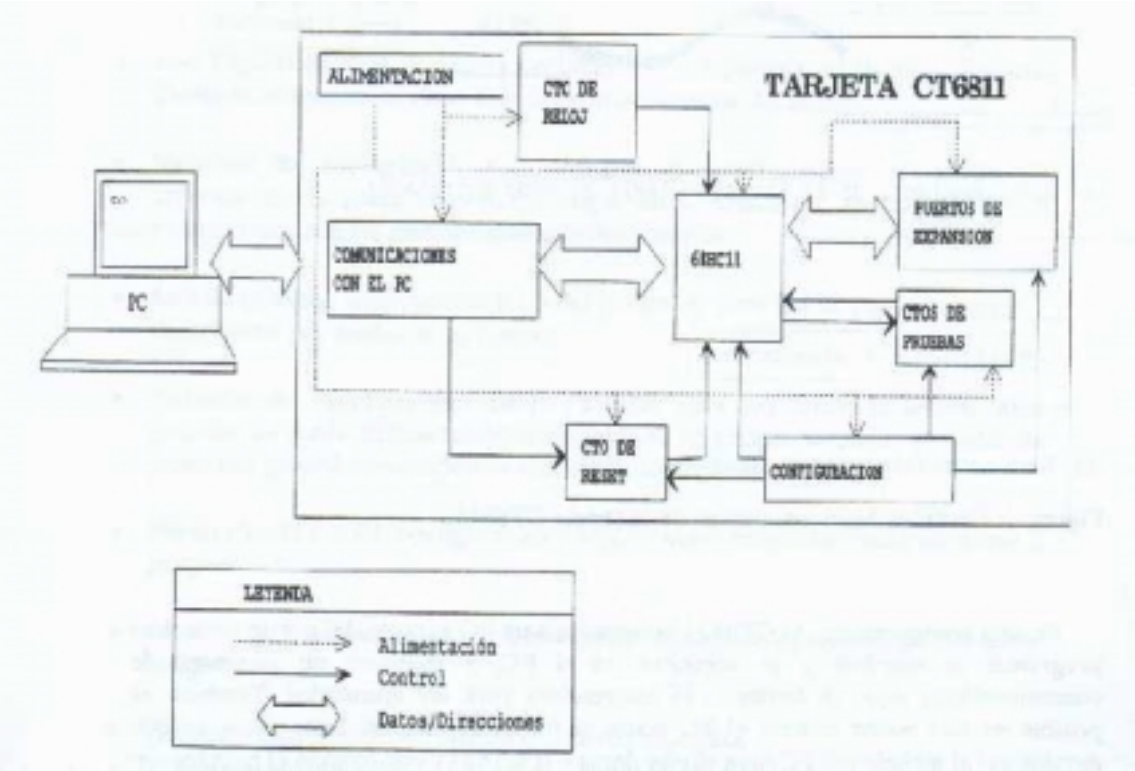


Fig. 3:Diagrama de bloques de la tarjeta

El núcleo central de la tarjeta CT6811 está constituido por el microcontrolador 68HC11A1 de Motorola. Los demás bloques surgen como una extensión del 68HC11.
La tarjeta se puede dividir en 8 bloques: el corazón de la placa (68HC11), el circuito de reloj, el circuito de inicialización o *reset*, el circuito de pruebas, el circuito de comunicaciones con el PC, la configuración del sistema, los puertos de expansión y la alimentación del sistema completo. A continuación se explica cada bloque:

- **Microcontrolador 68HC11A1:** Se trata de un microcontrolador de 8 bits. Además de una CPU que le permite ejecutar instrucciones y realizar operaciones aritmético lógicas, dispone en el propio chip de memorias ROM, RAM y EEPROM, así como una serie de periféricos integrados que hacen de este micro una herramienta muy potente.
- **Circuito de reloj:** Este circuito permite que el micro obtenga la señal de reloj necesaria para funcionar. Está constituido por un cristal de 8 MHz, que es el valor máximo que permite el 68HC11. Además, con este valor, el micro es capaz de comunicarse con el PC a las velocidades de 1200,2400 ó 9600 baudios.
- **Circuito de *reset*:** El circuito de *reset* permite la correcta inicialización del 68HC11. Se ha diseñado de tal forma que es posible realizar un *reset* por *hardware*, apretando un pulsador, o bien realizar un *reset* por *software* desde el PC, cuando la CT6811 está funcionando en modo entrenador. El *reset* software es muy útil cuando se están desarrollando

programas para *Microbot*, pues cada vez que hay que cargar un programa nuevo en el *Microbot*, no es necesario desplazarse hasta él y apretar el botón de *reset*, sino que directamente desde el PC se puede llevar a cabo.

- **Configuración:** La parte de configuración de la CT6811 está constituida por 6 *jumpers* y 4 *switches*. Dos de los *switches* permiten configurar el 68HC11 para trabajar en cualquiera de los 4 modos para los que está diseñado: *bootstrap*, *single* chip, *expanded* y *special*. Los otros 2 *switches* están a disposición del usuario a través de los puertos de expansión para configurar tarjetas periféricas conectadas a la CT6811. Los 6 *jumpers* permiten configurar la CT6811 para realizar diferentes funciones, que se comentarán más adelante.
- **Circuito de pruebas:** La CT6811 dispone de un *led* conectado al bit 6 del puerto A del 68HC11. Este *led*, que se puede conectar y desconectar a través de un *jumper*, es muy útil para la realización de software de pruebas. El pulsador de la tarjeta, se puede configurar mediante un *jumper* para actuar bien como pulsador de *reset* o bien como un pulsador normal conectado a la entrada de interrupción IRQ del 68HC11. De este forma, es posible realizar pequeños programas de prueba que lean el pulsador y actúen en consecuencia.
- **Comunicaciones con el PC:** Esta es una de las partes más importante. A través del circuito de comunicaciones es posible comunicarse con el PC para intercambiar información. Las comunicaciones se realizan a través del puerto serie del PC, mediante la norma RS-232. La velocidad máxima compatible con el PC es de 9600 baudios, aunque el 68HC11 es capaz de transmitir a mucha más velocidad.
- **Puertos de expansión:** La CT6811 dispone de 6 puertos de expansión donde es posible conectar los diferentes periféricos. Todas las señales del 68HC11, salvo las correspondientes al reloj (EXTAL y XTAL), son accesibles desde los puertos de expansión. Cinco de los puertos coinciden con los 5 puertos del 68HC11: A, B, C, D y E. El sexto puerto está destinado a llevar las señales de control del 68HC11.
- **Alimentación:** La tensión de alimentación de la CT6811 oscila entre 4.5 y 5.5 voltios. La tarjeta dispone de un conector hembra de tipo *jack* cilíndrico para la conexión de un transformador, y dos bornas ajustables mediante tornillos para la conexión de cables de alimentación, provenientes por ejemplo de pilas o baterías. *[Elek.94]*

1.3.3 Configuración de la tarjeta

La tarjeta CT6811 dispone de *switches* y *jumpers* para que el usuario la configure según sus necesidades. Existen 4 *switches* y 8 *jumpers*. De los 4 *switches*, dos se utilizan para configurar los modos de funcionamiento del 68HC11 y los otros dos quedan disponibles para aplicaciones del usuario.

1.3.3.1 Configuración de los modos del micro

SWITCHES 1 y 2	MODA	MODB	MODO	DESCRIPCION
	0	0	BootStrap	No acceso memoria externa Modo especial Ejecución programa BOOTSTRAP
	0	1	Single Chip	No acceso memoria externa Vectores interrupción en ROM
	1	0	Special test	Acceso a memoria externa Modo especial Vector interrupción en ROM
	1	1	Expanded	Acceso a memoria externa Vector interrupción en ROM ó RAM

Figura 4: Configuración de los modos mediante los *switches* de la tarjeta CT6811.

1.3.3.2 Configuración de los *jumpers*

JUMPER	FUNCIÓN	CONECTADO	QUITADO
JP1	Actuar sobre VRH	VRH = VCC	VRH accesible desde el bus de expansión
JP2	Actuar sobre VRL	VRL = GND	VRL accesible desde el bus de expansión
JP3	Actuar sobre el LED	LED conectado a PA6	LED desconectado PA6 liberado
JP4	Seleccionar usos del pulsador: IRQ/RESET	El jumper en la izquierda activa la IRQ	El jumper en la derecha activa el pulsador de reset
JP5	Seleccionar modo entrenador o autónomo	Modo autónomo	Modo entrenador.
JP6	Actuar sobre el LVI	LVI conectado	LVI desconectado
JP7	Seleccionar reset software ON / OFF	El jumper abajo activa el reset software	El jumper arriba desactiva el reset software
JP8	Actuar sobre XIRQ	Modo normal	Sólo en caso de programar la EPROM del 68HC11E9

Fig.5: Configuración de los *jumpers* de la CT6811.

1.3.4 Puertos de expansión

La tarjeta CT6811 dispone de 6 puertos de expansión. Cinco de estos puertos coinciden con los cinco puertos del 68HC11, y se han denominado igual: puerto A, puerto B, puerto C, puerto D y puerto E. El sexto puerto contiene otras señales de interés del 68HC11.

Puerto A: Este puerto está constituido por los 8 bits del puerto A del 68HC11 más un *pin* de VCC y otro *pin* de GND ambos para poder llevar alimentación a los periféricos de una forma cómoda.
Puerto B: Igual que el puerto A pero con el puerto B del 68HC11. El puerto B del 68HC11 sirve también como parte alta del *bus* de direcciones en caso de ampliación del micro con memoria externa. Cuando funciona en modo no expandido, el bit PB0 se corresponde con A8, PB1 con A9,... , PB7 con A15.
Puerto C: A este puerto de expansión se lleva el puerto C del 68HC11. Este puerto funciona como un puerto normal de E/S cuando no hay conectada memoria externa. Cuando se conecta memoria externa, este puerto lleva el byte bajo del *bus* de direcciones *multiplexado* con el *bus* de datos. El bit PC0 se corresponde con AD0, PC1 con AD1,..., PC7 con AD7.
Puerto D: Este puerto está constituido por el puerto D del 68HC11.
Puerto E: Constituido por el puerto E del 68HC11. Este puerto en el 68HC11 tiene una doble función: puerto de entrada de 8 bits ó 8 canales de conversión A/D. Las señales VRL y VRH se utilizan para introducir las señales de referencia alta y baja del conversor. Si los *jumpers* JP1 y JP2 están colocados, por estos podemos obtener la alimentación: VRH=VCC, VRL=GND. Si no están colocados estos *jumpers*, podremos introducir las tensiones de referencia necesarias para nuestra aplicación.
Control: Por este puerto se han "recopilado" una serie de señales del 68HC11 para el control. Por los *pines* SW3 y SW4 se sacan las señales correspondientes al estado de los *switches* de usuario 3 y 4.

1.3.5 Desarrollo de programas para la CT6811

1.3.5.1 Filosofía de trabajo

La filosofía empleada para el manejo de la CT6811 es la siguiente. En el PC se programan las aplicaciones para la CT6811 en ensamblador del 68HC11. Estos programas fuente tienen extensión. ASM. Utilizando el ensamblador FREEWARE AS11 de Motorola, los programas se compilan, obteniéndose archivos ejecutables con extensión. S19. Estos son los archivos que se envían a la CT6811.

Para enviar programas a la CT6811 se pueden utilizar varios programas. En los ejemplos de más adelante se utiliza el programa DOWNMCU v.10.

Una vez que se ha enviado el programa, el 68HC11 comienza su ejecución. Se puede cargar programas en la CT6811 tantas veces como se quiera. Los programas así cargados son almacenados en la RAM interna del 68HC11.

Una vez que se tiene la aplicación depurada, se graba en la memoria EEPROM del 68HC11 para que se quede permanentemente. Ahora la CT6811 no necesita del PC. Funciona en modo autónomo. Para grabar programas en la EEPROM interna se pueden utilizar varios programas: PC-BUG, CTDIALOG... En esta sección se utilizará el programa CTDIALOG. Este programa, además de permitir grabar la EEPROM, permite

ver el contenido de la memoria del 68HC11, cambiar valores de la memoria, desensamblar...

1.3.5.2 Un ejemplo completo

En esta sección se va a presentar un programa fuente que se va a compilar para después ser cargado en la CT6811 y finalmente grabado en la EEPROM.
El programa hace parpadear el *led* de la CT6811.El programa fuente se puede ver en la Figura 6.

En el Anexo 1 se detallan los mnemonicos del 68hc11.

```
; Simplemente se enciende y se apaga el led de la tarjeta CT6811.

        ORG $000

comienzo
        LDAA $1000
        EORA #$40           ; Cambiar de estado el bit PA6
        STAA $1000

        LDY #$FFFF          ; Realizar una pausa
dec     DEY
        CPY #0
        BNE dec

        BRA comienzo        ; Repetir el proceso
        END
```

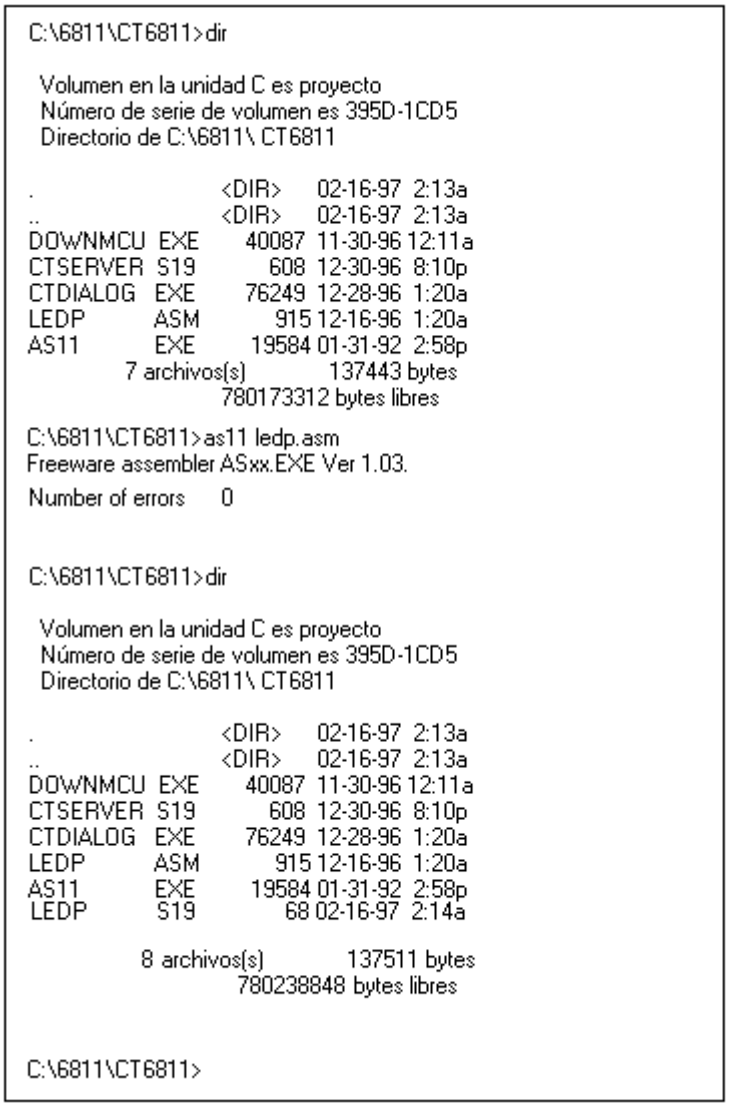
Listado del programa LEDP.ASM

Fig.6

El siguiente paso es compilar el programa para obtener el archivo LEDP.S19, que es el fichero ejecutable. Para ello se utiliza el ensamblador AS11 v.1.0.3 de Motorola. En las siguientes figuras se muestran los comandos que hay que teclear y los resultados obtenidos. Los comandos tecleados por el usuario se han escrito en negrita. En la Figura 7 primero se muestran todos los ficheros necesarios y después se compila el programa *ledp.asm*. Se obtiene el archivo ledp.s19 que se trata de un archivo de texto y por tanto se puede editar (¡Pero no se debe modificar!).

Ahora hay que enviar el programa LEDP.S19 a la CT6811. En la Figura 8 se muestran los comandos. Se ha supuesto que la CT6811 se encuentra conectada en el puerto serie COM2. Si se quiere utilizar el puerto COM1 basta con cambiar el parámetro –com2 por –com1 (¡en minúsculas!). En cuanto se recibe el mensaje de "ENVIO CORRECTO" el programa ya se estará ejecutando en el 68HC11 y se podrá ver como parpadea el *led*.

Los programas cargados en el 68HC11 se encuentran almacenados en la RAM interna del micro. Se pueden cargar los programas todas las veces que se quiera, sin ninguna restricción. Una vez que se tiene el programa depurado, se graba en la EEPROM. En el ejemplo, se trata de un programa muy sencillo y su depuración es muy fácil. A continuación se indica como grabar el programa LEDP en la EEPROM.



```
C:\6811\CT6811>dir

 Volumen en la unidad C es proyecto
 Número de serie de volumen es 395D-1CD5
 Directorio de C:\6811\ CT6811

.                   <DIR>     02-16-97  2:13a
..                  <DIR>     02-16-97  2:13a
DOWNMCU  EXE        40087  11-30-96 12:11a
CTSERVER S19          608  12-30-96  8:10p
CTDIALOG EXE        76249  12-28-96  1:20a
LEDP     ASM          915  12-16-96  1:20a
AS11     EXE        19584  01-31-92  2:58p
          7 archivos(s)          137443 bytes
                    780173312 bytes libres

C:\6811\CT6811>as11 ledp.asm
Freeware assembler ASxx.EXE Ver 1.03.
Number of errors     0


C:\6811\CT6811>dir

 Volumen en la unidad C es proyecto
 Número de serie de volumen es 395D-1CD5
 Directorio de C:\6811\ CT6811

.                   <DIR>     02-16-97  2:13a
..                  <DIR>     02-16-97  2:13a
DOWNMCU  EXE        40087  11-30-96 12:11a
CTSERVER S19          608  12-30-96  8:10p
CTDIALOG EXE        76249  12-28-96  1:20a
LEDP     ASM          915  12-16-96  1:20a
AS11     EXE        19584  01-31-92  2:58p
LEDP     S19           68  02-16-97  2:14a

          8 archivos(s)          137511 bytes
                    780238848 bytes libres


C:\6811\CT6811>
```

Fig.7

```
C:\6811\CT6811>downmcu ledp -com2

DOWN-MCU. V1.0 (C) GRUPO J&J. Noviembre-1996.
Envío de programas a la entrenadora

Fichero a enviar:  .\ledp.S19
Puerto serie:  COM2

Pulse reset en la entrenadora...
Transmitiendo:
>>>>>>>>>>>>>>>>>>.....................................................................
.....................................................................
.....................................................................
.....OK!
Envío correcto
Tamaño del programa:  22 bytes
C:\6811\CT6811>
```

Cargando el programa LEDP.S19 en la CT6811.

Fig.8

Antes de nada, hay que modificar el programa fuente para indicar que lo queremos situar en la memoria EEPROM. Si el programa a grabar en la *eeprom* utilizase variables, habría que situar las variables en la RAM interna y dejar sólo el código en la EEPROM. Como el programa *ledp* no tiene variables, no hay que preocuparse. En la Figura 9 se muestra el listado del programa *ledp.asm* modificado para ser grabado en la memoria EEPROM. El único cambio que ha habido que hacer es en la directiva ORG, que indica el comienzo del programa. En vez de comenzar en la dirección $0000, ahora deberá comenzar en la dirección $b600, que es el comienzo de la memoria EEPROM en el microcontrolador 68HC11A1. Si utilizamos otro modelo se deberá averiguar esta posición de inicio. El nuevo programa creado se denomina *ledpe.asm*. La "e" del final se ha puesto para indicar que se trata de un programa para la *eeprom*. Una vez compilado este programa, de forma similar a como se ha hecho con el programa *ledp.asm*, obtenemos el fichero ledpe.s19. Este será el fichero que utilicemos para grabar en la EEPROM.

```
; Simplemente se enciende y se apaga el led de la tarjeta CT6811.

        ORG $B600               ; ¡ Memoria EEPROM !

comienzo
        LDAA $1000
        EORA #$40               ; Cambiar de estado el bit PA6
        STAA $1000

        LDY #$FFFF              ; Realizar una pausa
dec     DEY
        CPY #0
        BNE dec

        BRA comienzo            ; Repetir el proceso
        END
```

Programa ledp preparado para ser grabado en la EEPROM.

Fig. 9

El programa que se va a emplear es el CTDIALOG. En la Figura 10 se muestra como se carga este programa. Primero hay que enviar a la CT6811 el programa CTSERVER y después se ejecuta el programa CTDIALOG.EXE. Además de permitir grabar la EEPROM del 68HC11, con el CTDIALOG podremos ver el contenido de la memoria, modificarla, desensamblar, ejecutar programas... .

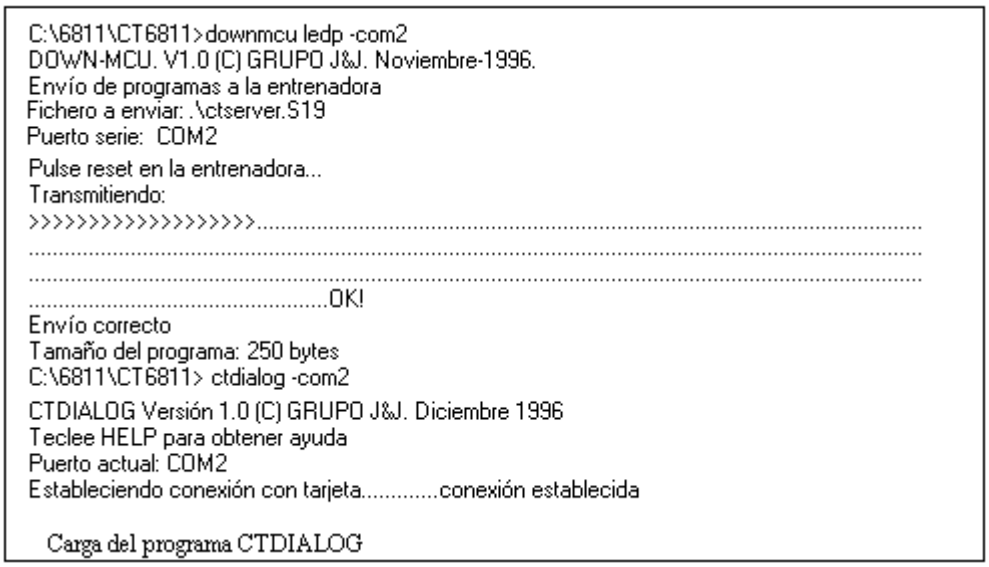


Fig. 10

Si se ejecuta el programa CTDIALOG y el servidor CTSERVER no se ha cargado previamente, o se ha cargado mal, aparecerá en el *prompt* del CTDIALOG un asterisco, indicando que no hay conexión con la tarjeta.

Para grabar el programa ledpe.s19 en la EEPROM hay que utilizar el comando EEPROM seguido del nombre del fichero. Los pasos a seguir se encuentran explicados en la Figura 11: Primero se graba el programa en la EEPROM y después se desensambla el contenido de la *eeprom* para asegurarse de que se ha grabado correctamente. Para probar el programa, se puede configurar la CT6811 en modo autónomo (conectando el *jumper* JP5) y hacer un *reset*. También es posible realizar un "salto" a la EEPROM desde CTDIALOG, como se ha hecho en el ejemplo de la Figura 11. Al realizar el salto, el 68HC11 comienza a ejecutar el programa almacenado en la EEPROM. El servidor que estaba en la RAM interna se deja de ejecutar y por tanto se pierde la conexión del CTDIALOG con la CT6811. Para volver a ejecutar el CTDIALOG habrá que salir utilizando el comando quit, volver a cargar el servidor y volver a ejecutar el CTDIALOG.

```
>eeprom ledpe
Fichero a grabar en memoria EEPROM: LEDPE. S19
Transmitiendo:---------------------------------->>
Grabación terminada
Número bytes grabados:  22

>dasm b600
B600    LDAA$1000
B603    EORA #40
B605    STAA 1000
B608    LDY #FFFF
B60C    DEY
B60E    CPY #0000
B612    BNE B60C
B614    BRA B600

>g b600
Conexión perdida
>quit
programa terminado
```

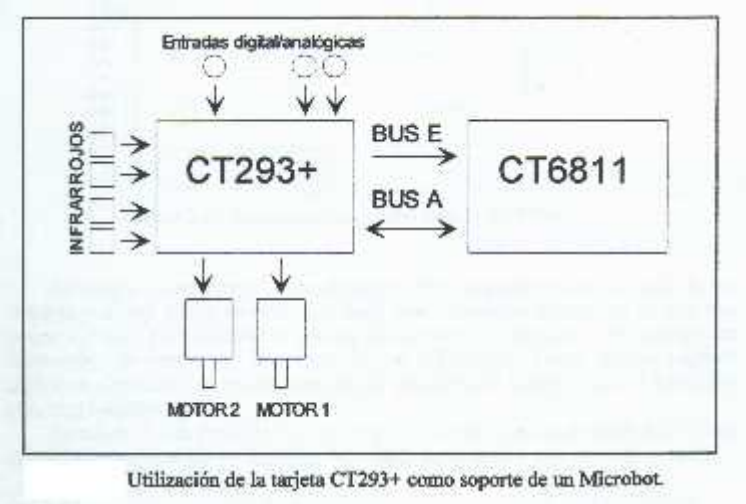
Fig. 11

1.4 La tarjeta CT293+

1.4.1 Introducción a la CT293+

La CT293+ es una nueva tarjeta que proporciona al sistema *tower* la posibilidad de controlar motores y sensores. Esta diseñada para adaptarse perfectamente a la CT6811 y poder controlarla sin ninguna variación tanto en *Bootstrap*, como en *Single* Chip. La tarjeta también puede controlarse desde otro sistema por ejemplo el puerto paralelo de un PC. Esta tarjeta es la versión mejorada de la CT293, pero se ha respetado la compatibilidad. Los programas de su antecesora valen también para la nueva, lo único que hay que verificar es la situación de los sensores y motores.

La CT293+ es el soporte principal de los dos primeros niveles de la torre BOT, con ella se proporciona el movimiento a los sistemas de control y la capacidad de analizar el entorno. Con esta placa y con la CT6811 se puede obtener la plataforma de un *Microbot.*



Utilización de la tarjeta CT293+ como soporte de un Microbot.

Fig. 12

En la Figura 12 se representa el diagrama de bloques del hardware de un *Microbot*, la tarjeta CT6811 es el sistema microcontrolador usado para la programación y la tarjeta CT293+ es la interfaz entre la tarjeta anterior y los recursos externos (motores, sensores, entradas digitales y entradas analógicas).

Las características de la tarjeta CT293+ se pueden resumir en:

1. Posibilidad de control de dos motores de continua o uno paso_paso.
2. Capacidad para leer cuatro sensores de infrarrojos, pudiendo ser estos optoacopladores.
3. Disponibilidad de 8 entradas digitales de propósito general con la posibilidad de usarlas como entradas analógicas.
4. Alimentación de los motores externa o interna.

1.4.2 Descripción de los elementos de la CT293+

1.4.2.1 Disposición y explicación de los componentes de la CT293+

En la Figura 13 se puede apreciar la disposición de los componentes en la placa. Básicamente tiene dos bloques independientes. El primero de ellos (BLOQUE A) se encarga de los motores y de los sensores de infrarrojos, mientras que el segundo (BLOQUE E) controla las entradas digitales/analógicas. Hay un *bus* de control para cada bloque, llamados PUERTO A y PUERTO E. El usuario que este familiarizado con la CT6811 se dará cuenta del motivo de esta nomenclatura. El *bus* A (Puerto A) es el que maneja el bloque A, es decir motores e infrarrojos, el *bus* E (Puerto E) maneja el bloque E, que tiene las entradas analógico/digitales.

Figura 13

El bloque A esta formado por el chip L293B (1) que gestiona la potencia de los motores y el chip 40106 que tiene dos funciones. La primera consiste en realizar una pequeña lógica para facilitar el uso de los motores, la segunda en realizar una conversión de niveles en la lectura de los infrarrojos. A este bloque también pertenecen los arrays de resistencias de 220 ohmios y 47 Kohmios que se usan para polarizar los infrarrojos.

El bloque E esta formado por un *array* de ocho resistencias de valor 4K7. Sirve de *pull-up* para las entradas digitales. Más adelante se explica esto con más detalle.

Descripción de los elementos de la CT293+:

- C1, C2: Condensadores en paralelo con los motores. Sirven para eliminar ruido. El valor del condensador dependerá del motor utilizado. No poner condensadores electrolíticos.
- C3, C4: Condensadores de eliminación de ruido para las pastillas integradas.
- S1: *Array* de cuatro *switches*. Acodado. Sirve para anular las entradas de los sensores.
- U2: Pastilla 40106 c-mos inversora. Adapta niveles de entrada de los sensores.
- U1: *driver* de potencia L293B para control de motores.
- PUERTO A, E: Conectores tipo *bus* acodado (5+5 líneas, Macho). Conexión al sistema de control.
- R1: *Array* de resistencias de 4+1, polarización de infrarrojos.
- R2: *Array* de resistencias de 4+1, polarización de infrarrojos.
- R3: *Array* de resistencias de 8+1, *pull-up* de las entradas digitales.
- J2: Clema doble para la alimentación de los motores.

(1) También se puede poner el L293D que es totalmente compatible (incluye diodos).

- J4-J8: Clemas dobles para las entradas digitales.
- Motor1, motor2: Clemas dobles donde se conectan los motores.
- JP1: *Jumper* para conexión interna de los motores.
- Sensor1-sensor 4: Conexiones para los sensores de infrarrojos CNY70.

1.4.3 Instalación de los sensores y motores

La CT293+ puede incorporar bastantes tipos de sensores, también puede manejar motores paso a paso y motores de continua.

1.4.3.1 Conexión de los motores de continua a la CT293+

La tarjeta esta preparada para poder controlar dos motores de continua simultáneamente. Se realiza por medio del circuito integrado L293B que contiene dos circuitos internos con el "puente en H" necesario para controlar motores. Es un integrado que se adapta perfectamente ofreciendo una etapa de control y de potencia en un mínimo espacio.

La conexión de los motores a la tarjeta se realiza a través de las clemas dobles llamadas "motor1" y "motor 2". Estas clemas están situadas en el centro de la parte inferior de la tarjeta, ver Figura 14. Los motores se pueden situar lejos de la placa, aunque es aconsejable que el cable de unión no supere los 20 cm, por cuestiones de ruido eléctrico, y pérdida de potencia.

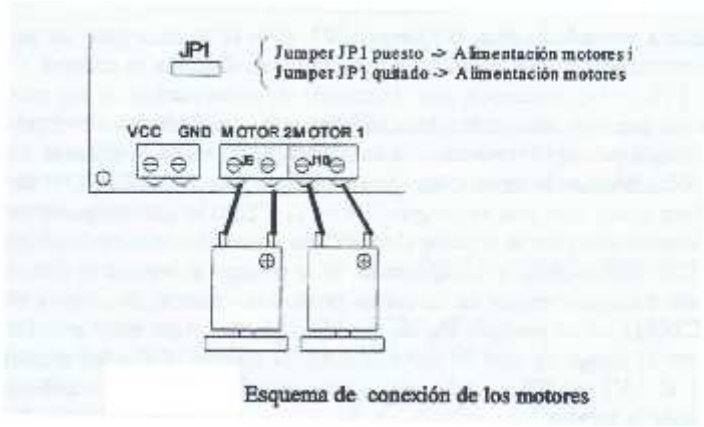


Fig. 14

Generalmente los motores tienen un sentido de giro preferente. Al girar en ese sentido las escobillas resbalan sobre el colector produciendo un desgaste pequeño. Los fabricantes de motores suelen indicar este sentido colocando un signo positivo en alguno de los conectores del motor. Si se introduce la tensión de acuerdo con ese criterio el giro obtenido será el correcto. Esto no quiere decir que el motor se vaya a romper si se introduce mal, lo único que refleja es que si la aplicación va a mover el motor en un solo sentido conviene que este coincida con el preferente para aumentar la vida útil del mismo. Cuando el motor se va a utilizar en aplicaciones que requieran ambos giros, lo anterior no afectará mucho, el desgaste de las escobillas se producirá sobre todo por los cambios de sentido y no por el rozamiento inverso que será despreciable. Se va a utilizar esta característica para definir una forma normalizada de conectar los motores, de forma que cualquier *Microbot* construido siguiendo este patrón será compatible a nivel software.

Siguiendo la forma de conexión anterior y mirando los motores de frente se puede sacar esta tabla.

	MOTOR 1		MOTOR 2	
DIRECCIÓN	BIT 5	ON (1) - DERECHA OFF (0) - IZQUIERDA	BIT 6	ON (1) - IZQUIERDA OFF (0) - DERECHA
ESTADO	BIT 3	ON (1) - motor ON OFF (0) - motor OFF	BIT 4	ON (1) - motor ON OFF (0) - motor OFF

Tabla de control de los motores

Fig. 15

Se explica ahora la función del *jumper* JP1. Este es el encargado de seleccionar entre la alimentación interna o externa de los motores. Cuando se conecta el *jumper*, la tensión TTL (+5v) se conecta con la entrada de alimentación de los motores. La ventaja es que con una sola fuente de tensión se puede dar energía a todo el sistema. El inconveniente esta en el ruido que los motores introducen en el sistema. Sobretodo aparecen caídas bruscas de tensión que pueden desprogramar la EEPROM interna del 68HC11. Para evitar esto esta el *jumper* JP6 en la CT6811 (LVI), este *jumper* protege a la *eeprom* haciendo un *reset* de la placa siempre que la tensión de alimentación este por debajo de 4.5 v. Esto supone un compromiso. Si se protege la *eeprom* el sistema puede desconectarse automáticamente en las caídas bruscas de tensión, si se quita el *jumper* JP6 de la CT6811 no se produce esa desconexión (salvo caídas muy grandes <2.5) pero se corre el riesgo de que se desprograme la *eeprom*.

Cuando el *jumper* JP1 de la CT293+ esta quitado significa que se ha seleccionado la alimentación de motores externa. La desventaja es la necesidad de utilizar dos fuentes de alimentación, mientras que la ventaja es el poder poner más tensión a los motores. Por ejemplo se puede utilizar 12v,9v,1v,... sin interferir con la alimentación TTL. Otra ventaja es que no se produce tanto ruido en la línea y el LVI puede ser compatible con los motores.

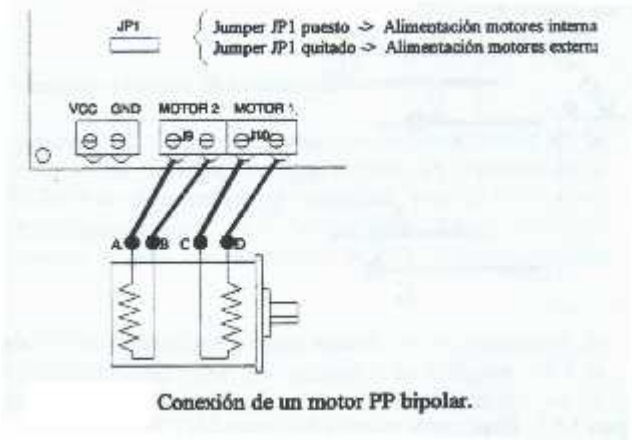
El L293B permite alimentaciones para los motores de hasta +36v y una corriente de salida de 1 Amperio. Cuando se use en condiciones extremas probablemente sea necesario ponerle un disipador.

1.4.3.2 Conexión de un motor paso_paso a la CT293+

La CT293+ puede controlar un motor paso_paso.

Internamente un motor PP esta constituido por una serie de bobinas excitadoras cuyos extremos salen al exterior en forma de cables. Los motores bipolares tiene cuatro cables correspondientes a los extremos de dos bobinas simples. Los unipolares tienen seis, cuatro son los extremos de las dos bobinas pero además hay otros dos que están conectados con el punto medio de cada bobina. Estos dos últimos no son de control sino de alimentación, se conectan directamente a Vmotor.

Distinguir el tipo de motor no es difícil, todo se reduce a contar el número de cables que hay. Saber que cables pertenecen a la misma bobina es algo más difícil, aunque con un polímetro funcionando como óhmetro se puede averiguar. Para ello se tiene que medir la resistencia entre cables, si la lectura es de muy pocos ohmios se puede suponer que los cables pertenecen a la misma bobina. Para *bus*car el cable del centro en motores unipolares se mide la resistencia mínima, y esta se produce entre uno de los extremos y el cable central.



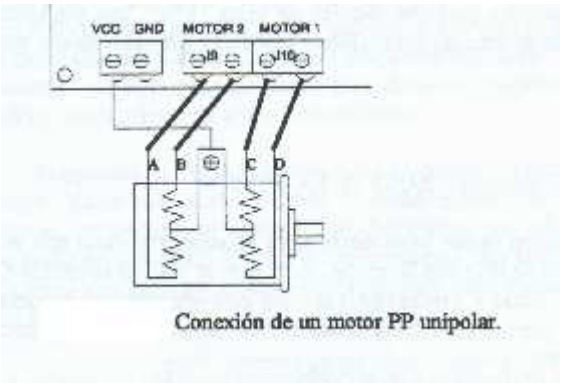


Fig. 16

1.4.3.3 Conexión de los sensores de infrarrojos a la CT293+

Otro de los elementos que maneja la CT293+ son los infrarrojos CNY70. En concreto esta tarjeta tiene capacidad para controlar directamente cuatro. Si se recurre a las entradas digitales extras se puede llegar a controlar hasta un número de doce. Cada CNY70 incorpora un emisor y un receptor en el mismo encapsulado.

Estos infrarrojos son de corta distancia, son capaces de distinguir el negro de otro color mientras la distancia del infrarrojo a la superficie no supere unos pocos centímetros. Lo más normal es situarlos a medio centímetro del suelo para evitar interferencias con otras fuentes de luz, ya sea solar o artificial.

1.4.3.4 Conexión con las entradas digitales

Las entradas digitales se introducen por las clemas de conexión situadas en la parte superior de la tarjeta. Hay un total de 10 entradas pero sólo 8 son entradas, las otras dos se usan para establecer los niveles de referencia VRH y VRL. En modo digital estos niveles serán VRL=0 y VRH=5. Si se trabaja con la CT6811 y se tienen los *jumpers* JP1 y JP2 puestos los niveles se sitúan automáticamente en VRL=GND=”0” y VRH=VCC=”5”.

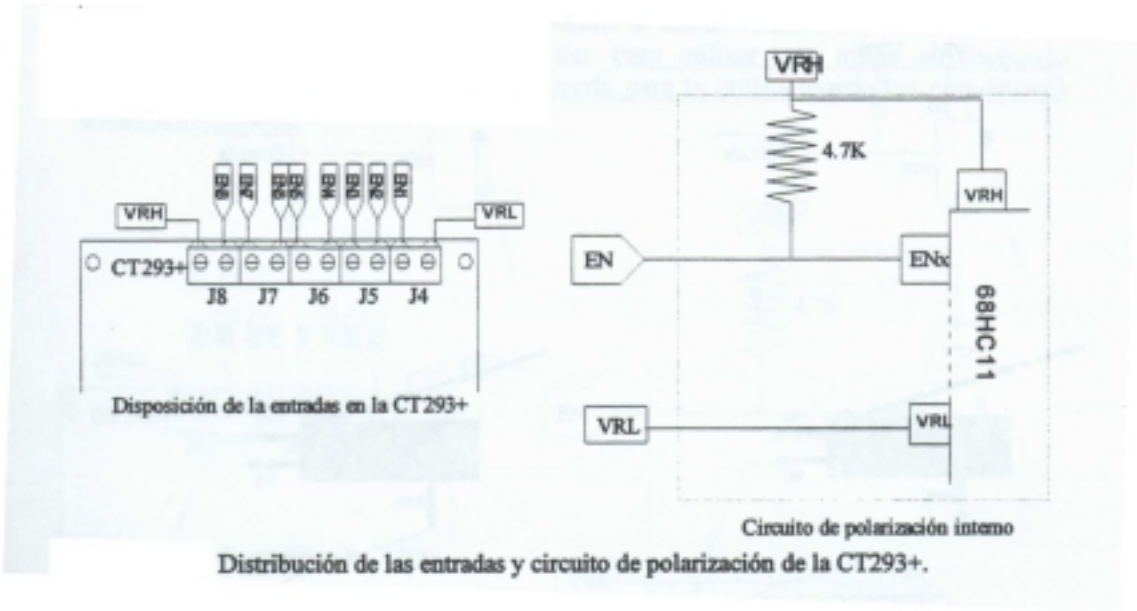


Fig. 17

En la Figura 17 se presenta el circuito de polarización desarrollado en la tarjeta CT293+ y la disposición de las entradas. Según el circuito de polarización la lectura que se obtiene cuando no se conecta nada en las entradas, estando los niveles de referencia a VCC y GND, es prácticamente VRH, es decir VCC. Lo que significa nivel alto "1". Este comportamiento se debe a la resistencia de 4.7 k, que se denomina resistencia de *pull-up*.

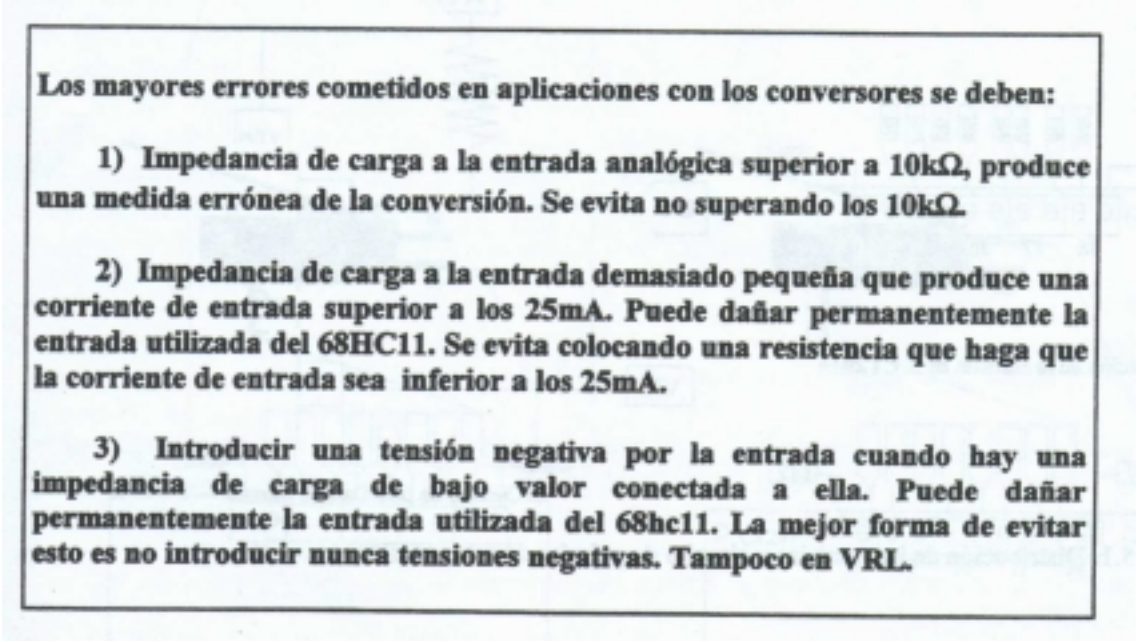
1.4.3.5 Conexión con las entradas analógicas

Las entradas digitales que se explican en el apartado anterior pueden configurarse desde la CT6811 como entradas analógicas. Ahora se indica como utilizar correctamente las entradas y su circuito de polarización. Para utilizar este modo es necesario disponer la CT6811 y además configurarla para la utilización de los conversores analógico-digitales.

El Puerto E del 68hc11 puede funcionar de dos formas distintas. La primera se corresponde con el funcionamiento en modo digital explicado en el apartado anterior. La segunda es el modo analógico que se explica aquí. Los dos utilizan el mismo circuito de conexión para sus entradas y lo que era una ventaja en modo digital ahora se convierte en un inconveniente. La resistencia de *pull-up* que antes facilitaba la conexión de ciertos dispositivos, ahora hace que la señal analógica sufra una distorsión no deseable.

Las entradas VRH y VRL son los niveles de referencia que necesita el conversor analógico digital del 68HC11 para su correcto funcionamiento. Lo normal es utilizar VRH=VCC y VRL=GND, por eso esta opción se encuentra disponible en la CT6811 con tan solo conectar los *jumpers* JP1 y JP2 (situados en la esquina superior derecha). Si se desconectan hay que introducir los niveles por las entradas correspondientes. El valor de estos debe estar comprendido entre (0V – 6V). Si se superan los 6 V las conversiones pueden ser erróneas, y si se introduce una tensión por debajo de cero, negativa, puede dañar permanentemente la entrada de nivel analógica del microcontrolador 68HC11. Además, por las entradas analógicas no hay que introducir nunca tensiones negativas, pues probablemente se rompería dicha entrada.

Otros datos importantes son que no se debe poner una resistencia serie con la entrada superior a 10 kΩ y que la corriente máxima de entrada no puede exceder los 25 mA.



Los mayores errores cometidos en aplicaciones con los conversores se deben:

1) Impedancia de carga a la entrada analógica superior a 10kΩ, produce una medida errónea de la conversión. Se evita no superando los 10kΩ.

2) Impedancia de carga a la entrada demasiado pequeña que produce una corriente de entrada superior a los 25mA. Puede dañar permanentemente la entrada utilizada del 68HC11. Se evita colocando una resistencia que haga que la corriente de entrada sea inferior a los 25mA.

3) Introducir una tensión negativa por la entrada cuando hay una impedancia de carga de bajo valor conectada a ella. Puede dañar permanentemente la entrada utilizada del 68hc11. La mejor forma de evitar esto es no introducir nunca tensiones negativas. Tampoco en VRL.

1.4.4 Programación de la tarjeta CT293+

La tarjeta esta pensada para conectarse a la tarjeta CT6811. Aunque se puede conectar a otros tipos de controladores. El rendimiento máximo de la tarjeta se saca cuando se conecta el puerto A y el puerto E de la CT6811. El control de la CT293+ se realiza a través de dos bytes. Un byte controla los motores y sensores de infrarrojos y el otro las entradas digitales/analógicas.

Se divide este capítulo en dos partes, en la primera se explica como controlar el bloque A de la CT293+ (motores e infrarrojos) y en la segunda se explica como usar el bloque E (entradas digitales/analógicas).

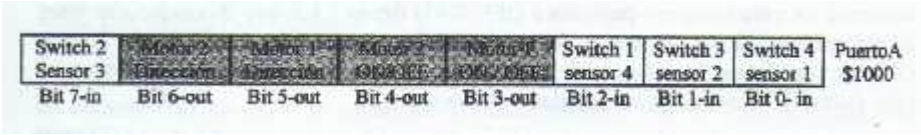
1.4.4.1 Programación del bloque A. Motores y sensores de infrarrojos.

Lo primero es unir el bloque A de la CT293+ con la CT6811 utilizando el *bus* A., para ello conectar un cable de *bus* entre el Puerto A de la CT293+ (conector J1) y el Puerto A de la CT6811 (conector J1). Aquellas personas que no poseen la CT6811 tienen que conectar el Puerto A de la CT293+ con el puerto correspondiente en su sistema de control.

Automáticamente al colocar dicho cable se proporcionará la alimentación TTL a la CT293+. Ahora hay que elegir una de las dos opciones siguientes. La primera consiste en utilizar esa alimentación TTL para alimentar los motores, para ello hay que colocar en su sitio el *jumper* JP1 de la CT293+. La segunda consiste en alimentar los motores con otra fuente de alimentación, para ello hay que desconectar el *jumper* JP1 e introducir la alimentación externa por la clema J2. Tener cuidado con la polaridad.

Al utilizar la alimentación externa tener mucho cuidado con no olvidar desconectar el *jumper* JP1 y con la polaridad de la tensión de alimentación de los motores. Si se utiliza alimentación única antes de colocar el *jumper* JP1 desconectar la alimentación externa de los motores (clema J2).

Una vez realizado lo anterior el control de la CT293+ es muy sencillo. Tan solo se utiliza un byte de control, que se corresponde con el Puerto A de la CT6811, posición de memoria $1000 del microcontrolador 68hc11. El significado de los bits de dicho byte se detalla a continuación.



1.4.4.1.1 Los sensores de infrarrojos

Los bits en blanco significan que son de entrada, y proporcionan el estado de los infrarrojos. Para que la lectura de estos bits sea correcta es necesario activar las

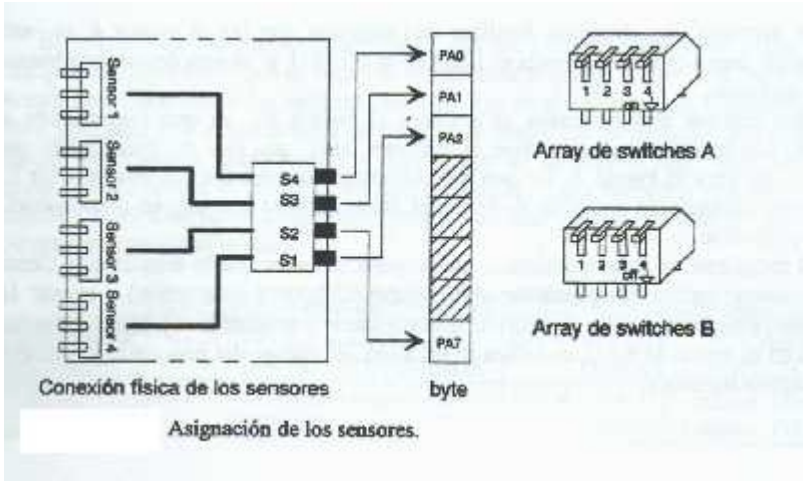


Fig. 18

salidas de los sensores de infrarrojo correspondientes. Eso se hace con los interruptores que hay en la placa. La correspondencia entre la localización física del infrarrojo y su interruptor se representa en la Figura 18.

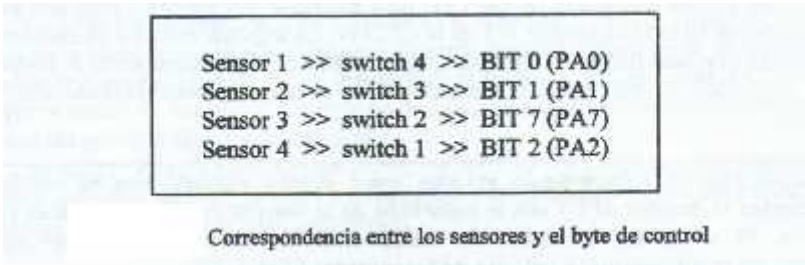
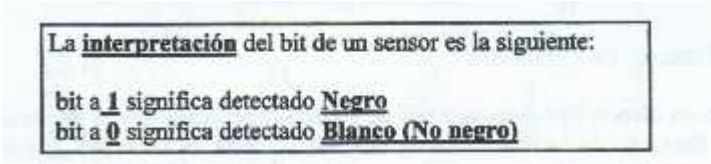


Fig. 19

Los interruptores (*switches*) que se han puesto en la placa tienen la utilidad de anular y dejar libres los bits que emplean los infrarrojos. Si el sistema sólo va a manejar

motores se pueden utilizar los bits restantes como entradas sin entrar en conflicto con los sensores. Esto es importante cuando se trabaja con el Puerto A de la CT6811 ya que esos bits son muy útiles y no conviene malgastarlos. Como norma básica se aconseja que los interruptores estén a ON cuando se vayan a manejar los sensores, en caso contrario ponerlos a OFF.



Como ejemplo se ha realizado un programa que lee el sensor 4, si esta detectando negro que no encienda el *led* de la CT6811 y si esta detectando blanco que lo encienda.

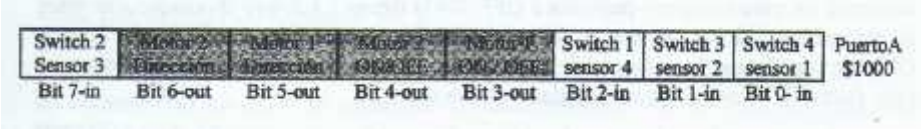
Para realizar este programa se activará el *switch* S1, ya que corresponde al sensor 4. Luego se procederá a crear el software. Hay que leer el byte $1000 del 68hc11, es decir el Puerto A. De este byte se ignorarán todos los bits menos el bit 2 (PA2), correspondiente al sensor 4. Según el valor leído se actuará en consecuencia con lo propuesto.

```
; programa de ejemplo de la lectura de un sensor
: si el sensor 4 ve negro se apaga el LED, si ve blanco se enciende el LED.

        ORG $0
inicio
        LDAA $1000      ;se carga en el Acumulador A el valor del puerto A.
        ANDA  #S04      ;nos quedamos con el bit-2 que es el que nos interesa
        CMPA #$04       ;bit_2 es uno ?
        BNE    cero     ; no->salta a 'cero'
        CLRA            ; si->apago el LED
        STAA $1000      ;
        BRA inicio      ;vuelve a empezar
cero    LDAA #$40       ;enciendo el LED
        STAA $1000
        BRA inicio      ;vuelve a empezar
        END             ;fin del programa
```

1.4.4.1.2 Los motores de continua

Al bloque A pertenece la gestión de los motores. Los bits sombreados (del byte de control) son salidas y se encargan del funcionamiento de los motores. Los bits marcados con Dirección seleccionan el sentido de giro. Los bits marcados con ON/OFF indican si se conecta o no el motor.



El control de los motores se realiza con un circuito que tiene como parte principal el chip L293B que es capaz de suministrar hasta un Amperio a cada uno de los motores. Conectando los motores como se dijo en el anteriormente y mirando cada motor de frente se obtiene la tabla de control de abajo.

	MOTOR 1	MOTOR 2
DIRECCION	BIT 5 ON (1) - DERECHA OFF (0) - IZQUIERDA	BIT 6 ON (1) - IZQUIERDA OFF (0) - DERECHA
ESTADO	BIT 3 ON (1) - motor ON OFF (0) - motor OFF	BIT 4 ON (1) - motor ON OFF (0) - motor OFF

Tabla de control de los motores

Al igual que antes se presenta un ejemplo. El programa realiza la lectura del sensor 1, si esta sobre negro el motor 1 empezará a girar. Si esta sobre blanco el motor se parará.

Para resolver el problema se puede utilizar el programa del ejemplo anterior pero cambiando los valores de los bits. Ahora el sensor1 se corresponde con el microinterruptor 4 y con el bit-0 (PA0). El bit que activa el motor 1 es el bit-3, el sentido en este ejemplo no importa.

```
; programa de ejemplo de control de motores
; si el sensor 1 ve negro encendemos el motor 1, si ve blanco lo apagamos.

PORTA equ $0

        ORG $0
        LDX #$1000
inicio
        STAA $1000
        BRCLR PORTA,X $01 cero
        LDAA  #$08          ; pongo un 1 en el bit 3 para encender el motor
        BRA inicio          ;vuelve a empezar
cero    CLRA                ;apago el motor
        BRA inicio          ;vuelve a empezar
        END                 ; fin del programa
```

Realmente este programa es igual que el anterior, un poco modificado pero en lugar de encender el *led* se enciende el motor.

1.4.4.2 Programación del Bloque E: Entradas digitales/analógicas

Lo primero que se tiene que hacer es unir el Bloque E de la CT293+ con la CT6811 utilizando el *bus* E. Se conecta un cable de *bus* entre el Puerto E de la CT293+ (conector J3) y el Puerto E de la CT6811 (conector J3). El byte de control de este bloque esta situado en la posición de memoria $100A, que se corresponde con el puerto E.

Este cable de *bus* tiene diez hilos, ocho de ellos se corresponden con bits de entrada y los otros dos se utilizan cuando se trabaje con las entradas analógicas. La principal función del bloque E es la de proporcionar ocho entradas con *pull-up*. Si no hay nada conectado en ellas se lee en los bits correspondientes un nivel alto "1". Cuando se conecte alguna señal en la entrada dependiendo de su valor se obtiene un nivel alto o bajo.

Cuando se utiliza la CT6811 el Puerto E se puede utilizar de dos formas diferentes. La primera se corresponde con lo anterior, utilizar el puerto E como entradas digitales con *pull-up*. La segunda es configurar dicho puerto para leer canales analógicos. Aunque el bloque E no se pensó para usarlo de forma analógica las pruebas realizadas con diferentes sensores de este tipo dieron buenos resultados. El inconveniente es que debido a las resistencias de *pull-up* hay una distorsión en la medida. Distorsión por otro lado conocida y por tanto corregible por hardware o por software.

1.4.4.2.1 Ejemplo de utilización en modo digital

Este bloque E esta especialmente diseñado para poder conectar pulsadores sin ningún esfuerzo en la tarjeta. Un tipo de pulsador puede ser un *bumper* o interruptor de fin de carrera. En la Figura 20 se recuerda su estructura interna y el circuito de polarización al conectarse a la CT293+. Viendo dicho circuito se aprecia que cuando no se activa el *bumper*, la lectura que se obtiene en la entrada digital es nivel alto "1". Cuando se activa el *bumper* la entrada se cortocircuita con VRL, lo que implica que ahora la lectura sea nivel bajo "0". Se recuerda también que los niveles de referencia VRL y VRH se tienen que definir salvo que se este utilizando la CT6811 con los *jumpers* JP1 y JP2 conectados. En este caso por defecto se tiene VRH=VCC y VRL=GND.

Si no se conectan los *jumpers* se tienen que establecer los niveles externamente. El rango de tensiones admisible es de 0 V a 5 V. Utilizando las entradas en modo digital estos valores son VRL=0 V y VRH=5 V. En modo analógico es cuando más se suelen utilizar otras referencias distintas.

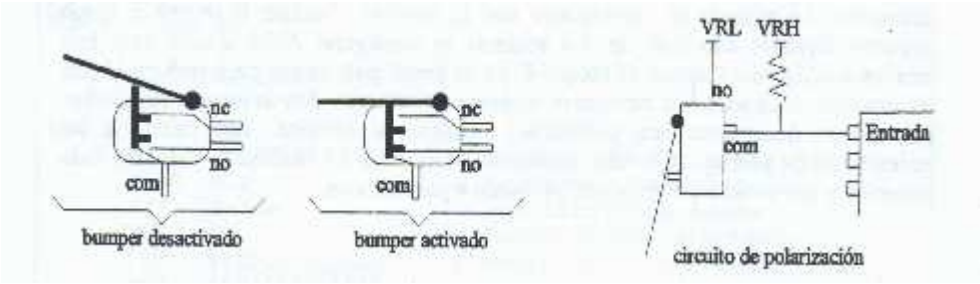


Fig. 20

En el circuito de polarización propuesto, cuando el *bumper* esta inactivo, se lee nivel alto a la entrada debido a la resistencia de *pull-up*. Cuando se activa se lee nivel bajo pues la entrada se ha cortocircuitado con VRL dentro del *bumper*. Este circuito no solo es válido para *bumpers* sino que también se puede conectar cualquier circuito exterior al cual no le afecte tener una resistencia de *pull-up* a la salida.

El programa que se propone tiene que leer el estado de un *bumper* conectado a la entrada digital 8 y encender un *led* cuando se active, cuando se apriete la palanca. Se

supone que la conexión es la misma que la figura de arriba con los niveles VRH=VCC y VRL=GND.

La entrada 8 se corresponde con el bit ocho del byte de control E situado en la posición de memoria $100A del microcontrolador 68hc11. Al apretar la palanca en la entrada se lee nivel bajo “0”, si no se aprieta se lee nivel alto “1”.

```
; Programa de ejemplo de las entradas digitales
; Conectamos un bumper en la entrada 8, cada vez que lo
; pulsemos el LED se enciende.

        ORG $0

PORTA   equ $0
PORTE   equ$A
        LDX #$1000
inicio  STAA PORTA,X
        BRSET PORTE,X $80 apaga
        LDAA #$40
        BRA inicio
apaga   CLRA
        BRA inicio
        END
```

CAPITULO 2. OBJETIVOS

2.1 Objetivos del sistema

Comunicación entre el ordenador personal (PC) y el *Microbot*, vía radio frecuencia (comunicación *simplex* en un solo sentido), que cumplirá los siguientes objetivos:

- Una aplicación que "correrá" bajo el PC en lenguaje de programación C, deberá proporcionar una interfaz visual al usuario de forma que cualquier persona podrá utilizar el sistema y enviar comandos al/los *Microbots*.

- Un módulo emisor estático que se conectará al puerto paralelo del PC, y será el encargado de transmitir los comandos. (1)

- Un módulo receptor, el cual recibirá los comandos del emisor y se los "pasará" al *Microbot*, su estructura mecánica debe ser la adecuada para que se adapte el *Microbot Tritt*, esto es, crecimiento hacia arriba.

- Una aplicación que se ejecutará en el *Microbot* en lenguaje ensamblador, deberá ser capaz de interpretar los comandos que le llegarán (al *Microbot*), a través del receptor.

---

(1) Este proyecto tendrá la capacidad de poder añadir más receptores con un mismo emisor y con la posibilidad de gobernarlos de forma conjunta o selectiva. Esto se conseguirá por medio de una comunicación codificada, es decir, al dato a transmitir le acompañará una dirección, para un receptor en cuestión.
El sistema será abierto y modular con un enfoque para posibles ampliaciones.

El aspecto externo del sistema que implementará los objetivos requeridos será el de la Figura 21.

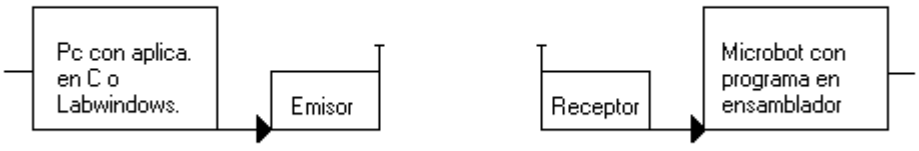


Fig.21

2.2 Funcionalidad del sistema

Un programa en lenguaje C que se ejecutará en el PC, generará los comandos (izquierda, derecha, adelante, detrás, paro) en función de las ordenes que proporcionará el usuario desde el teclado (o el ratón) del ordenador. Estas ordenes las codificará el programa mediante un código binario y a través del puerto paralelo se "pasarán" al emisor.

El emisor será un transmisor de datos por RF; codificará los datos(palabra binaria) y los transmitirá a la frecuencia adecuada.

El receptor (perfectamente sintonizado con el emisor), decodificará los datos para volver a la palabra binaria original. Esta palabra la tomará el *Microbot* a través de sus entradas digitales.

En el *Microbot* se ejecutará un programa en ensamblador, y será el encargado de transformar la palabra de entrada (puerto E), en las ordenes para el accionamiento de los motores, y ejecutar así el comando que se ordenó por el usuario inicialmente, desde el teclado (o el ratón).

2.3 Características mecánicas

La característica mecánica que se exigirá al sistema estará determinada por la estructura del *Microbot*. Es decir, debido a la estructura de crecimiento hacia arriba, la placa de circuito impreso receptora deberá tener las mismas dimensiones que las otras dos, propias del *Microbot*, de tal manera que se podrán colocar las tres, una encima de la otra y separadas por tornillos separadores.

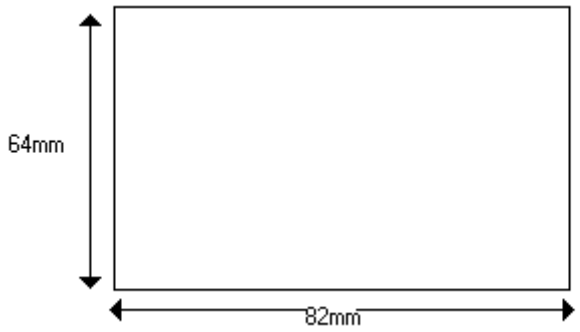


Figura 22: Dimensiones de la placa receptora

2.4 Especificaciones

Como conclusión de los apartados anteriores, se indicarán a continuación las especificaciones del sistema:

- Aplicación:

  Interfaz de Panel para el usuario; se seleccionará la dirección, sentido y posibilidad de paro y salida del programa.

- Conexión al módulo emisor:

  A través del puerto paralelo.

- Transmisión de datos:

  Transmisión de datos digital codificado por RF.

- Dimensiones de la placa de circuito impreso receptora:

  64mm. X 82mm.

- Entrada de datos al *Microbot*:

  A través de las entradas digitales (PUERTO E).

- Tamaño del programa ensamblador:

  Como máximo 512 bytes (Tamaño EEPROM del *Microbot*).

- Sistema modular y fácilmente ampliable.

CAPITULO 3.ANÁLISIS

El desglose del sistema global en distintos módulos es el siguiente:

3.1 Sistema de bloques de primer nivel

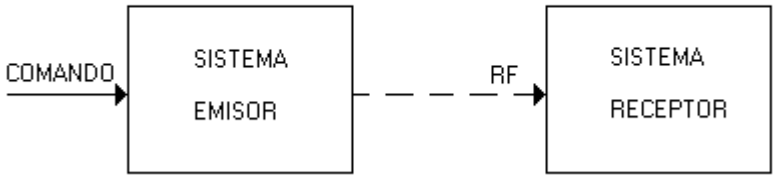


Fig. 23

3.2 Sistema de bloques de segundo nivel

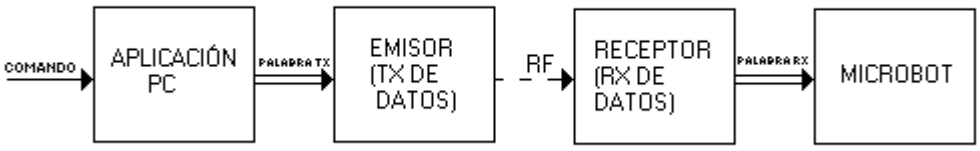


Fig. 24

3.3 Sistema de bloques tercer nivel

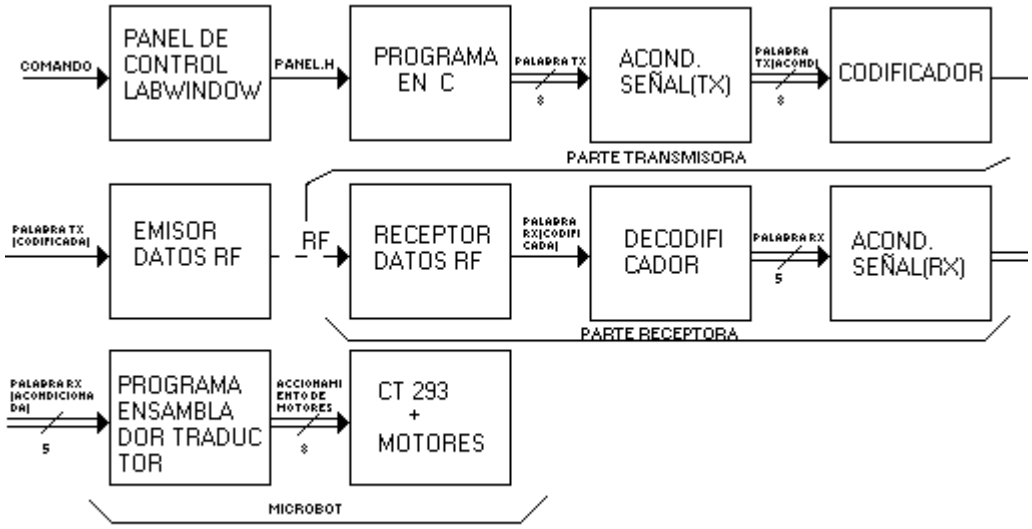


Fig.25

3.4 Descripción de módulos

3.4.1 PANEL DE CONTROL *LABWINDOWS*

*Entradas: El usuario mediante una pulsación de ratón, seleccionará la acción a realizar (“COMANDO”).

*Salidas: La interconexión entre el panel *Labwindows* y el programa en Lenguaje C se realizará incluyendo esta cabecera (“PANEL.H”).

*Funcionalidad: Interfaz gráfica para el usuario.

3.4.2 PROGRAMA C

*Entradas: El programa esperará eventos del panel gráfico (“PANEL.H”).

*Salidas: *Bus* de 8 bits por el puerto paralelo (“PALABRA TX”): 4 bits bajos para datos, 5º a ‘0’ habilita TX, 6º, 7º, 8º siempre a ‘0’.

*Funcionalidad: Esperará un evento del panel, y en función de la orden enviará una palabra por el puerto paralelo.

3.4.3 ACOND. SEÑAL (TX)

*Entradas: “PALABRA TX”, es un *bus* de 8 bits.

*Salidas: “PALABRA TX (ACOND)”, *bus* de 8 bits de salida.

*Funcionalidad: Acondicionará *bus* del puerto paralelo mediante resistencias de *pull-up.*

3.4.4 CODIFICADOR

*Entradas: “PALABRA TX (ACOND)”.

*Salidas: “PALABRA TX (CODIFICADA)”.

*Funcionalidad: La palabra de entrada entrará en paralelo, se codificará, se añadirá una dirección y todo se concatenará en serie y saldrá por una salida.

3.4.5 EMISOR DATOS RF

*Entradas: “PALABRA TX (CODIFICADA)”, se trata de la señal codificada que entrará a este bloque en serie.

*Salidas: ”RF”, emisión por RF a 433 MHz.

*Funcionalidad: Transmisión de los datos de entrada serie por RF a 433 MHz.

3.4.6 RECEPTOR DATOS RF

*Entradas: “RF”, señal de radio frecuencia.

*Salidas: Salida de los datos codificados procedentes del emisor en serie: ”PALABRA RX (CODIFICADA)”.

*Funcionalidad: Recepción de los datos procedentes del emisor (433 MHz).

3.4.7 DECODIFICADOR

*Entradas: “PALABRA RX (CODIFICADA)”.

*Salidas: “PALABRA RX”, es un *bus* de 5 hilos. Los 4 bits bajos de datos y el quinto de recepción válida.

*Funcionalidad: Decodificará los datos y direcciones que le entrarán en serie, y si la recepción es correcta, se obtendrán los datos nuevamente en paralelo (4 bits), el 5º indicará que la recepción ha sido, o no, correcta.

3.4.8 ACOND. SEÑAL (RX)

*Entradas: “PALABRA RX”.

*Salidas: “PALABRA RX (ACONDICIONADA)”, *bus* de 5 hilos.

*Funcionalidad: Resistencias, de tal forma que se mantendrán los márgenes de impedancia de carga en la entrada digital de la CT293 (*Microbot*).

3.4.9 PROGRAMA ENSAMBLADOR TRADUCTOR

*Entradas: “PALABRA RX (ACONDICIONADA)”.

*Salidas: “ACCIONAMIENTO DE MOTORES”, según la palabra que se escriba en el puerto A de la tarjeta CT6811 que está conectada con la CT293, los motores del *Microbot* se moverán en un sentido y dirección distinto.

*Funcionalidad: Traducirá los comandos de entrada, en ordenes para el accionamiento de los motores.

3.5 Interacción entre módulos

Al introducir un comando al panel mediante la pulsación del ratón (“COMANDO”), se producirá un evento en el panel, el cual será detectado por el programa en C, que actuará escribiendo la palabra binaria TTL de 8 bits en el puerto paralelo del PC (“PALABRA TX”). Esta palabra precisa una adaptación de niveles TTL mediante resistencia de *pull-up*, por lo que a la entrada del bloque CODIFICADOR se obtendrán los niveles adecuados “PALABRA TX (ACOND.)”. En este bloque de la palabra binaria de 8 bits se utilizarán los 4 más bajos para datos y el 5º para habilitar la transmisión, los 3 restantes serán omitidos; el bloque codificará los 4 bits y le añadirá una dirección que deberá coincidir con la del decodificador correspondiente. Toda esta trama de bits se obtendrá a la salida en serie “PALABRA TX (CODIFICADA)”, para su transmisión. El bloque EMISOR DATOS RF transmitirá bit por bit a 433 MHz (“RF”).

Una vez en el RECEPTOR DATOS RF que se encontrará perfectamente sintonizado con el emisor, los bits irán llegando uno tras otro, “PALABRA RX (CODIFICADA)”. El DECODIFICADOR realizará la función inversa que el CODIFICADOR, con la salvedad que en este módulo el 5º bit indicará recepción válida (Los 4 bits bajos serán los bits de datos “PALABRA RX”). En el módulo ACOND. SEÑAL (RX) se llevará a cabo una adaptación de impedancia según especificaciones de tarjeta del *Microbot*. Llegados a este punto los 5 bits perfectamente acondicionados serán leídos por PROGRAMA ENSAMBLADOR TRADUCTOR de un puerto (Puerto E), para su traducción en ordenes ejecutables por la CT293+MOTORES (Palabra puerto A de 8 bits “ACCIONAMIENTO DE MOTORES”), con lo que se ejecutará la orden inicial mandada por el usuario.

CAPITULO 4. DISEÑO DEL SISTEMA

4.1 Desarrollo de los módulos

Al final del capítulo de análisis se llegó a unos módulos funcionales perfectamente definidos en cuanto a entradas, salidas, funcionalidad. A continuación se describen como se han implementado físicamente. Hay que poner énfasis en el hecho de que, pese a que estos módulos tienen una funcionalidad perfectamente definida, para su desarrollo se ha tenido en cuenta la interacción entre ellos, de tal manera que no solo funcionaran por separado, también en el sistema completo.

4.1.1 Desarrollo del PANEL DE CONTROL *LABWINDOWS*

4.1.1.1 Presentación de *Labwindows* y el panel de control

Labwindows es un sistema de desarrollo que proporciona un entorno interactivo para la generación y depuración de programas. *[García.B.96]*

Dispone de multitud de librerías de funciones para control de instrumentación (GPIB, VXI) y manejo de tarjetas de adquisición, lo que permite una gran versatilidad y facilidad en el intercambio de información ordenador-instrumentación.

La comunicación usuario-ordenador se hace mediante "interfaz de usuario (*User Interface)"*, que consisten en paneles gráficos de control (mandos e indicadores), cuadros de dialogo, menús desplegables, etc., que serán sobre los que se opere a alto nivel. Estos objetos orientados a usuario se generan y editan con el *User Interface Editor* y se manejan desde programa con un conjunto de funciones específicas que se agrupan bajo la *User Interface Library.*

De este modo queda cubierto el flujo de control e información

Usuario ←→ Ordenador ←→ Instrumentación

Finalmente, dispone también de librerías de funciones para el análisis de los datos capturados (estadística, proceso de señal, algebra de matrices,etc.) y su representación gráfica, ya que lo normal será que los datos adquiridos por un sistema de instrumentación requieran algún tipo de procesado posterior.

El aspecto que presenta uno de los paneles de control mencionados puede ser el siguiente:

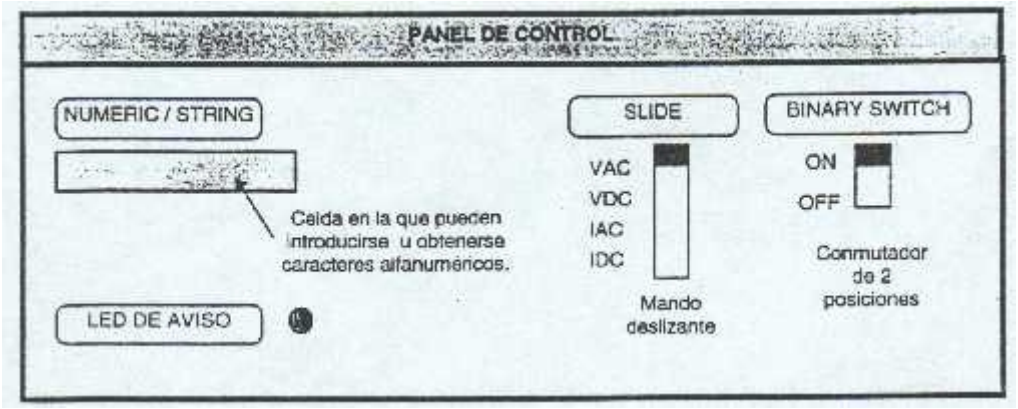


Fig. 26

Labwindows utiliza los lenguajes *QuickBASIC* o C, a elección del usuario, no siendo éstos completamente iguales a los normalizados sino específicos de este entorno.

Como se definió en las especificaciones del proyecto se ha utilizado aquí el lenguaje C.

Los programas desarrollados bajo el entorno *Labwindows*, pueden ser compilados y linkados para que se genere un *fichero*.exe que "corra" por si mismo, sin necesidad de cargar el *Labwindows.*

4.1.1.2 Utilidad del "*User Interface Editor*"

Como se ha dicho, el objetivo básico de *Labwindows* es crear una interfaz software entre el ordenador y la instrumentación, de modo que permita al usuario un control fácil e integrado de ésta.

Labwindows permite además crear interfaces gráficas de ususario, esto es, barras de comandos, cuadros de dialogo y paneles de control, para facilitar la intervención del mismo, esto se hace con el *User Interface Editor*.

Los objetos generados por el *User Interface Editor* se almacenan en ficheros que tienen la extensión por defecto *fichero.*uir.

Para que estos objetos se muestren en pantalla, el programa debe recurrir a las funciones de la *User Interface Library* que cargan y muestran el objeto. Las intervenciones posteriores del usuario o del programa sobre el objeto también se procesan mediante funciones de la citada librería.

El flujo general de proceso cuando se emplea un panel de control es el siguiente:

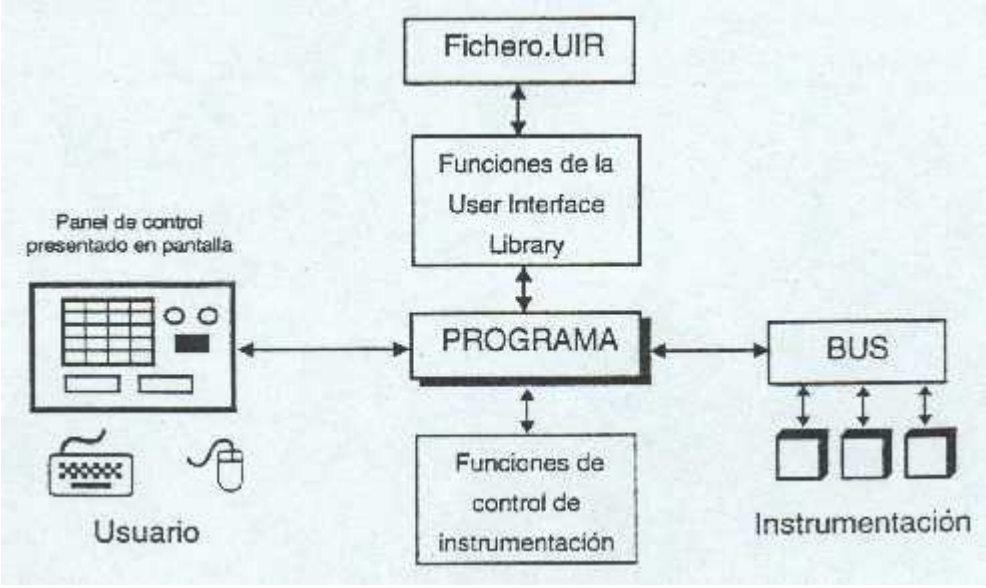


Fig. 27

4.1.1.3 Creación del panel de control

Para la creación del panel de control se ha entrado en la opción de *Labwindows:*

Options →User Interface Editor →Create → Panel

En donde aparece una ventana en la que se pide, entre otros parámetros de interpretación obvia:

Panel title
Constant prefix

Panel title es la denominación del grupo de mandos que se crean en solo un rótulo. Este campo se ha rellenado con el título "PANEL".

El campo *Constant prefix* se ha rellenado con "P1".

Al validar con *Ok* se presenta el aspecto inicial del panel de control.

Una vez generado el formato inicial del panel, el proceso ha consistido en añadir mandos de control, esto se verá más adelante.

Los paneles de control se almacenan en ficheros *fichero.*uir. Un *fichero.uir* puede contener más de un panel, que se diferencian a la hora de invocarlos por el nombre introducido en el campo *Constant prefix.*

Ha de tenerse en cuenta que cada control definido en el panel (pulsador, *display*, etc.) tiene asociado el nombre de una constante. Por ejemplo, un interruptor de encendido puede tener asociada una constante llamada *power*, un control para elegir un número de ciclos puede tener asociada una constante llamada *num_cycles* y así sucesivamente.

El nombre total de la constante lo confecciona el editor en automático anteponiendo el del panel. Así, la constante asociada al interruptor de encendido, si se ha rellenado el campo *Constant prefix* con "P", se llama *P_power*, la del control del número de ciclos *P_num_cycles*, etc. .

El editor le asigna valores secuenciales a estas constantes en un fichero *fichero.H* que genera en automático y es necesario incluirlo en el programa que utilice el panel. Estas constantes, que identifican el control, serán el "lazo" de unión del panel con el programa. Según el diagrama de bloques de la Figura 25 y como salida del módulo PANEL DE CONTROL LABWINDOW, el fichero se ha denominado "PANEL.H".

4.1.1.4 Creación de controles

Los controles disponibles están bajo *Create* del menú principal del *User Interface Editor*, y al desplegarse muestra una serie de controles disponibles (*NUMERIC / STRING, PUSH BUTTON, etc.)*, que el programador puede seleccionar y que deberá configurar antes de que aparezcan en el panel.

Los campos a configurar para la creación de un control se indicarán a continuación, pero antes conviene presentar el concepto de evento.

Un evento es una información que manda el control al programa indicando que ha sido manipulado. No siempre que se manipula un control se produce un evento, sino cuando el control ha sido configurado para ello. Cuando el programa detecta un evento debe decidir que acción tomar.

Según el tratamiento de los eventos, los controles tienen un modo que se establece a la hora de generarlos:

MODO DEL CONTROL	DESCRIPCIÓN
Normal	Puede ser operado por el usuario y cambiado desde programa.
Indicador	No puede ser operado por el usuario y si cambiado desde programa.
Hot	Igual que el normal pero se genera un evento al ser manipulado.
Validate	Igual que *hot* pero antes de generarse el evento se comprueba que todos los controles numéricos o escalares del panel son válidos (están en los rangos permitidos) y si no es así ofrece corrección mediante un cuadro de diálogo. Esto asegura que los controles numéricos y escalares son válidos antes de enviar un evento al programa

Fig. 28

Hay que reseñar que tras definir un control, su contorno se muestra en la esquina superior izquierda de la pantalla, sin presentar el detalle. Desde esta posición se lleva con el ratón hasta la posición deseada, donde aparecerán los detalles del control y su rótulo. Si se toma el rótulo con el ratón, se podrá mover éste respecto a la Figura del control. Si se toma la Figura de control, se moverá el conjunto del control más su rótulo en bloque.

Se está ya en disposición de crear los controles del panel. No se van a explicar todos los controles disponibles, sino los necesarios para la creación del panel, concretamente los pulsadores que se han utilizado son los *push button:*

PUSH BUTTON (Pulsador)	Se usa para activar una acción que aparece rotulada dentro del pulsador	EMPEZAR

Fig. 29

Una vez seleccionado aparece un cuadro de definición del control en el que se piden, entre otros, los siguientes campos:

CAMPO DEL 1ER CUADRO DE DEFINICIÓN	DESCRIPCIÓN
LABEL	Etiqueta (rotulación) del control
CONSTANT NAME	Nombre de constante asociada al panel con la cual se aludirá desde el programa
CONTROL MODE	Modo del control, explicado antes.
INITIALLY DISABLED	Especifica si el control está deshabilitado cuando se carga el panel por primera vez. Cuando un control está deshabilitado, su rótulo aparece subiluminado.
RAISED LABEL	Efecto de relieve en el rótulo.

Fig.30

Finalmente en el panel resultante se han utilizado los controles *push button* necesarios para cumplir las especificaciones, cada uno de ellos con su *label* y *constant* adecuada a su función, eso sí, todos los *push buttons* han sido configurados en el campo *Control mode* como *hot*, para así generar eventos en el panel. El resultado final se muestra en la Figura 31:

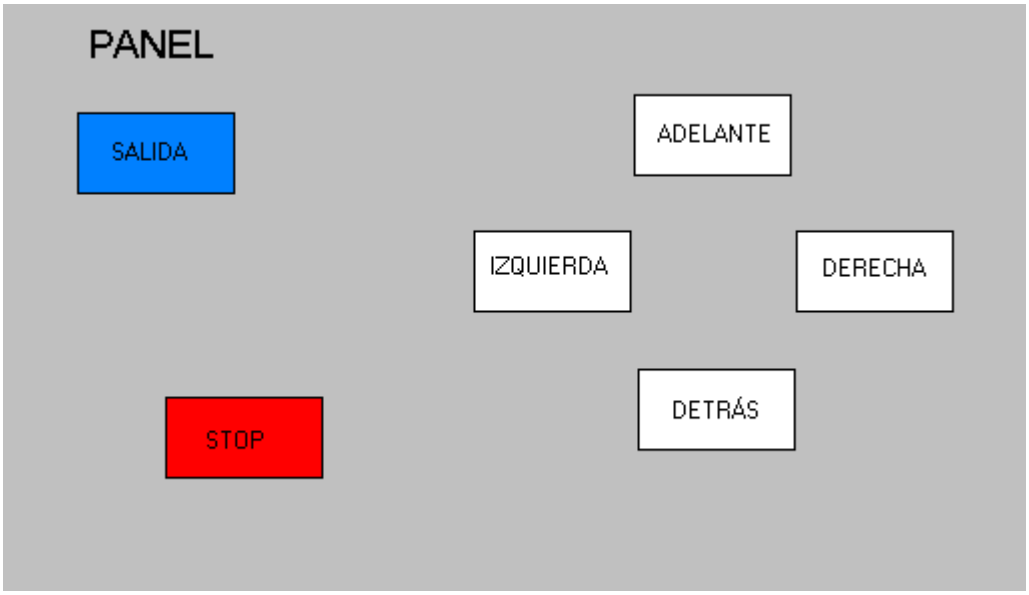


Fig. 31

El panel esta preparado para ser procesado por el bloque siguiente PROGRAMA EN C.

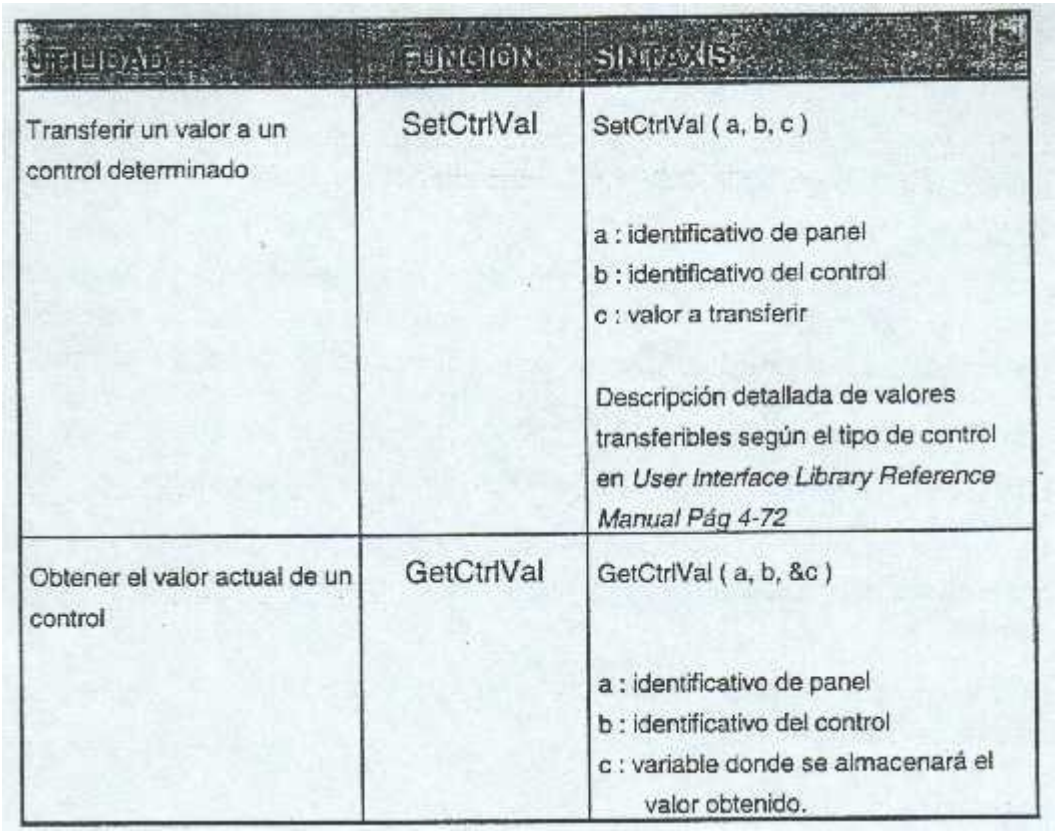
4.1.2 Desarrollo del bloque PROGRAMA EN C

Una vez definidos el panel de control y sus controles, el programa debe comandarlo e intercambiar información con él.

Como se indicó, las funciones que se encargan de esto son las de la *User Interface Library*.

La *User Interface Library* es muy extensa y potente, a continuación se tabulan algunas de estas funciones para el manejo de paneles desde programa:

UTILIDAD	FUNCION	SINTAXIS
Cargar el panel en memoria y asignarle un identificativo para referenciarlo posteriormente	LoadPanel	n = LoadPanel ("fichero.uir", p) ; n : identificativo que se asigna al panel p: panel del fichero que quiere seleccionarse (*constant prefix* del panel)
Mostrar el panel en pantalla. Si se muestra y no se desnabilita el usuario podrá operar sobre él	DisplayPanel	DisplayPanel (n) ; n : identificativo que se asignó al panel
Detectar eventos, esto es, manipulaciones del usuario sobre el panel, como por ejemplo pulsar un botón	GetUserEvent	GetUserEvent (a, &b, &c) a : modo de espera. Puede ser 0 ó <> 0 <> 0 : el comando se quedará en modo de espera, esto es, no avanzará hasta que se produzca el evento. 0 : el comando avanzará sin detenerse y recogerá el estado del evento cuando el programa pase por él. b : identificativo del panel. c : variable donde se almacena el identificativo del control que ha producido el evento. El identificativo es el valor tabulado en el fichero *include* asociado al panel



Utilidad	Función	Sintaxis
Transferir un valor a un control determinado	SetCtrlVal	SetCtrlVal (a, b, c) a : identificativo de panel b : identificativo del control c : valor a transferir Descripción detallada de valores transferibles según el tipo de control en *User Interface Library Reference Manual Pág 4-72*
Obtener el valor actual de un control	GetCtrlVal	GetCtrlVal (a, b, &c) a : identificativo de panel b : identificativo del control c : variable donde se almacenará el valor obtenido.

Fig. 32

En el programa C se han empleado algunas de estas funciones para su implementación, a continuación se detalla el código que cumple con las especificaciones de este módulo, el cual está debidamente comentado para su comprensión:

```
/*-------------------------proyect.c--------------------------
Programa para mandar comandos al microbot con interfaz de panel
para el usuario.
Autor:Manuel Gonz lez Mart¡n
------------------------------------------------------------*/
#include "C:\LW\INCLUDE\lwsystem.h"
#include "C:\LW\INCLUDE\userint.h"
#include "proyect.h"                /*Declaraci¢n de variables del
panel*/

/*----------------------Defines-----------------------------*/

#define CIERTO 1
#define FALSO 0
#define PUERTO 0x378
#define ADELANTE 0x01
#define DETRAS 0x02
#define IZQUIERDA 0x04
#define DERECHA 0x08
#define PARO 0x0F
/*----------------------Funciones---------------------------*/
void enviar(int comando);
int recibir(int sensor);
```

```
/*----------------------Programa principal--------------------*/

main()
{
    int p,ctrl;
    p=LoadPanel("proyect.uir",P);     /*Carga el panel*/
    DisplayPanel(p);                  /*Visualiza el panel*/
    while(CIERTO){
        GetUserEvent(1,&p,&ctrl);     /*Espera un evento del panel */
        switch(ctrl){
            case P_stop:enviar(PARO);
                        break;
            case P_adelante:enviar(ADELANTE);
                        break;
            case P_detras:enviar(DETRAS);
                        break;
            case P_izquierda:enviar(IZQUIERDA);
                        break;
            case P_derecha:enviar(DERECHA);
                        break;
            case P_salir:return;
                        break;
            }
     }
}
/*-------------------------Funci¢n enviar----------------------*/
void enviar(int comando)
{
    outp(PUERTO,comando);
}
```

4.1.2.1 Función enviar

void enviar(int comando)

Algoritmo

*Escribe el comando en la dirección del puerto paralelo.

Entradas

-comando: byte que indica el movimiento a realizar, según un código (ver apartado siguiente).

Salidas

Un byte por el puerto paralelo del PC.

Variables globales

-PUERTO: dirección física del puerto paralelo.

4.1.2.2 Código de comando utilizado

Para codificar los comandos se ha utilizado el siguiente código:

Adelante→ 0x01
Detrás→ 0x02
Izquierda→0x04
Derecha→0x08
Paro→0x0F

4.1.3 Desarrollo del ACOND.SEÑAL (TX)

El puerto paralelo de los PC y compatibles se presenta en forma de conector DB25 hembra. La conexión normal para este puerto de impresora es como puerto *Centronics*. De esta forma se hacen uso de las 8 líneas de datos unidireccionales y un número de señales de control. *[web3]*

Para el presente proyecto se ha utilizado este modo de funcionamiento normal, para escribir 8 bits de datos a la salida. Por lo que se ha trabajado con la dirección base del puerto (378H), como se pudo ver en el apartado anterior (Código en C).

En teoría, las líneas de datos son en colector abierto, por lo que ha sido necesario colocar unas resistencias de *pull-up*, para garantizar los niveles a la salida. Estas resistencias tienen un valor de 4k7 ohmios (IEEE-1284). *[web4]*

4.1.4 Desarrollo del CODIFICADOR

Llegados a este punto se tiene la palabra binaria a transmitir, previo a esto, como se comentó, es necesario codificarla. Se ha utilizado para ello el circuito integrado de *Motorola* MC145026, cuya hoja de características (*Data sheet)* se adjunta en el ANEXO 2. *[Moto]*

El MC145026 codifica 9 líneas de información y las transmite de forma serie cuando se habilita la señal de transmisión (/TE). Las nueve líneas pueden ser codificadas con datos trinarios (bajo, alto, o abierto) o datos binarios (bajo o alto). Las palabras se transmiten con redundancia doble para incrementar la seguridad (dos palabras de datos).

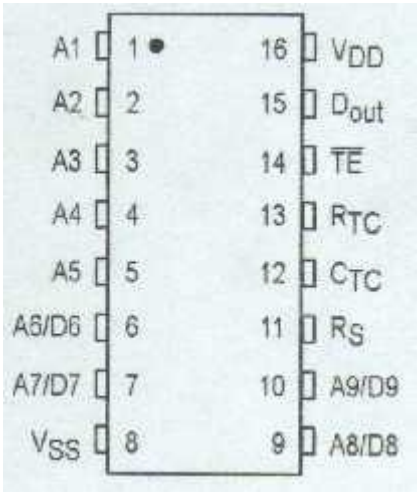


Fig. 33

4.1.4.1 Características operativas del MC145026

El codificador transmite en serie los datos definidos en el estado de los *pins* de entrada A1-A5 y A6/D6-A9/D9. Estos *pins* pueden estar en uno de los tres estados (bajo, alto, o abierto), permitiendo 19,693 códigos posibles. La secuencia de transmisión se inicia por un nivel bajo en la entrada /TE. El MC145026 puede estar continuamente transmitiendo tanto tiempo como la señal /TE permanezca a nivel bajo (además, el dispositivo puede transmitir dos secuencias de palabras con una pulsación de /TE). Al conectar la alimentación no se transmite inmediatamente la palabra de la entrada, ya que ésta puede ser incorrecta. Entre dos palabras de datos, no se envía señal en tres períodos de dato (ver Figura 34).

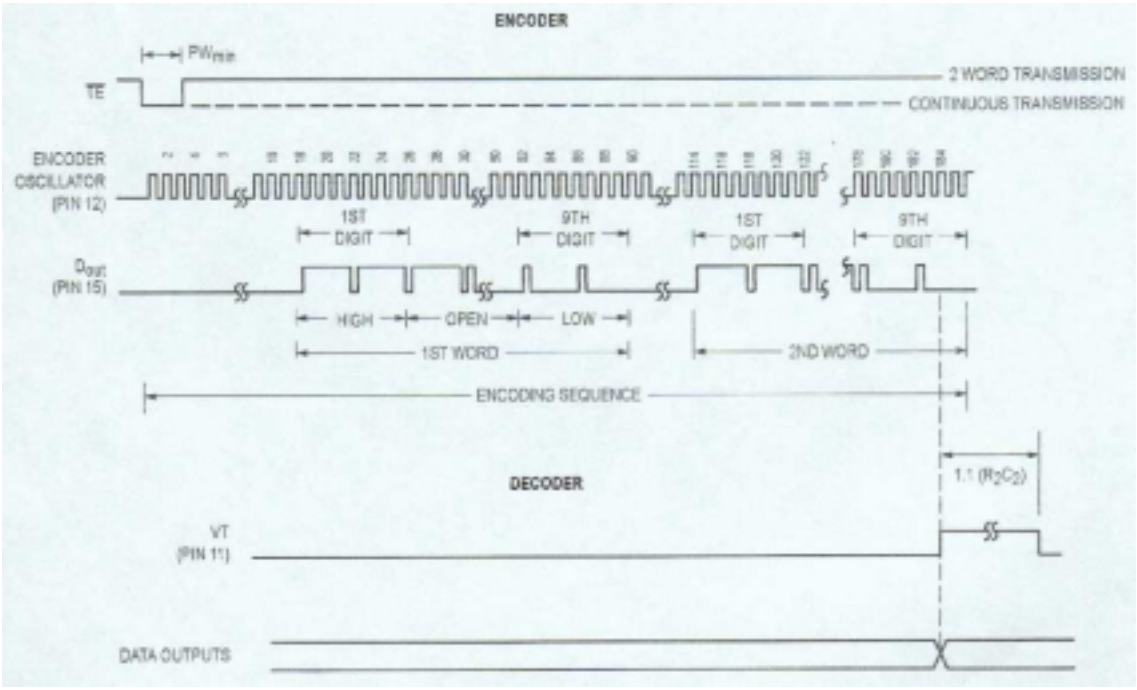


Fig. 34:Diagrama de tiempos

Cada dígito trinario transmitido es decodificado en pulsos (ver Figura 35). Un 0 lógico (bajo) esta codificado como dos pulsos cortos consecutivos, un 1 lógico (alto) como dos pulsos largos consecutivos, alta impedancia se codifica con un pulso largo seguido de uno corto.

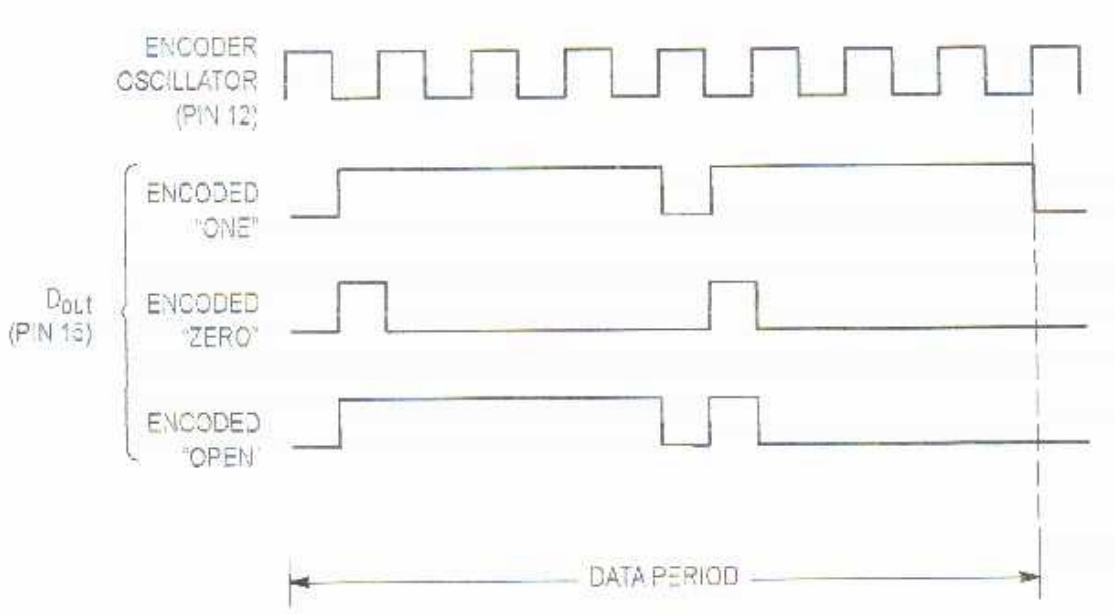


Fig. 35

La entrada /TE tiene internamente una resistencia de *pull-up*. Mientras /TE este a nivel alto y la segunda palabra ya haya sido transmitida, el codificador estará completamente deshabilitado, el oscilador inhibido, y el consumo de corriente será mínimo. Cuando se lleva /TE a nivel bajo, el oscilador arranca y la secuencia de transmisión comienza. Las entradas son secuencialmente seleccionadas, y las acciones a llevar a cabo son determinadas por los estados lógicos de éstas. Esta información se transmite en serie vía el *pin Dout.*

4.1.4.2 Descripción de los *pines*

A1-A5, A6/D6-A9/D9
Direcciones, Entradas de Direcciones/Datos (*Pins* 1-7, 9, y 10)

Estas entradas de direcciones/datos son codificadas y los datos son enviados en serie desde el codificador vía el *pin Dout.*

Rs, Ctc, Rtc
(*Pins* 11,12, y 13)

Estos *pins* son parte de la sección del oscilador.

Si una señal externa se usa como fuente osciladora, se debe conectar a la entrada Rs y la Rtc y la Ctc deben estar abiertas.

/TE
Transmit Enable (*Pin* 14)

Esta entrada de habilitación de transmisión, inicia la transmisión cuando se fuerza a nivel bajo. Un dispositivo interno de *pull-up* mantiene esta entrada normalmente alta.

Dout
Data Out (*Pin* 15)

Esta es la salida del codificador que de forma serie presenta la palabra de datos.

Vss
Negative Power Supply (*Pin 8)*

El potencial de alimentación más negativo. Este *pin* está normalmente a masa.

Vdd
Positive Power Supply (*Pin 16)*

El *pin* más positivo de alimentación.

4.1.4.3 Diagrama eléctrico del CODIFICADOR

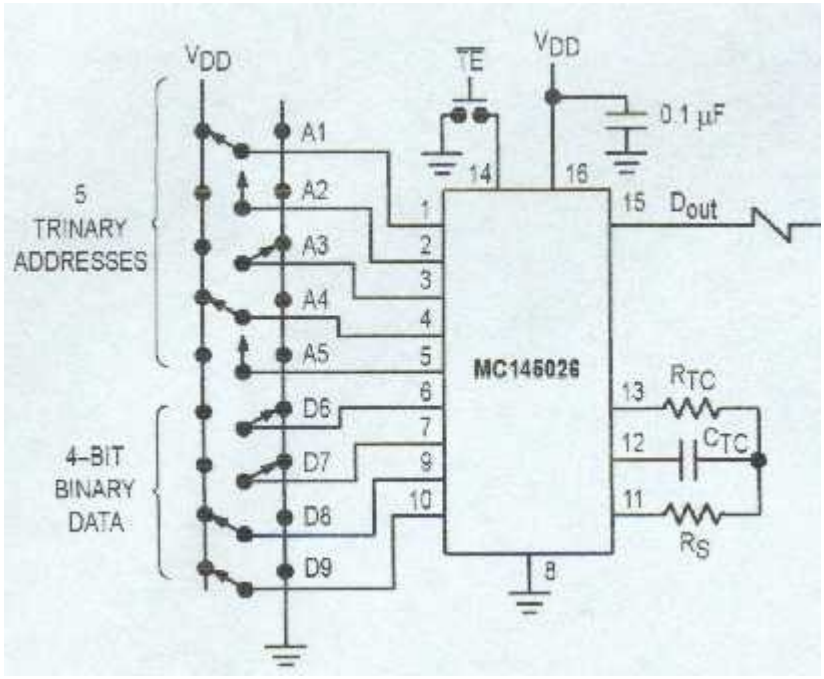


Fig. 36

En la Figura 36 se aprecia el diagrama eléctrico del codificador. Para el sistema completo se ha conectado en la entrada "4-BIT BINARY DATA" los 4 bits bajos del puerto paralelo del PC (Datos) y el 5º bit en la entrada "/TE" (Habilita TX).

Los valores de Rtc, Ctc, Rs, determinan la frecuencia del oscilador, por lo tanto la velocidad en la que son enviados los bits al EMISOR DATOS RF. Para ello, el fabricante proporciona una tabla de valores para las frecuencias más usuales; en la cual se pueden apreciar tanto los valores del codificador como los del decodificador, este último se verá más adelante.

Example R/C Values (All Resistors and Capacitors are ± 5%)

($C_{TC'} = C_{TC} + 20$ pF)

f_{osc} (kHz)	R_{TC}	$C_{TC'}$	R_S	R_1	C_1	R_2	C_2
362	10 k	120 pF	20 k	10 k	470 pF	100 k	910 pF
181	10 k	240 pF	20 k	10 k	910 pF	100 k	1800 pF
88.7	10 k	490 pF	20 k	10 k	2000 pF	100 k	3900 pF
42.6	10 k	1020 pF	20 k	10 k	3900 pF	100 k	7500 pF
21.5	10 k	2020 pF	20 k	10 k	8200 pF	100 k	0.015 µF
8.53	10 k	5100 pF	20 k	10 k	0.02 µF	200 k	0.02 µF
1.71	50 k	5100 pF	100 k	50 k	0.02 µF	200 k	0.1 µF

Fig. 37

Para el presente proyecto se ha utilizado la frecuencia de 1.71 KHz, para la cual y observando la Figura 34 se consigue una velocidad de aproximadamente 10 palabras por segundo, esto es, si una transmisión completa son 184 ciclos de reloj y la frecuencia del reloj son 1.700 ciclos por segundo:

1700/184 ~10 palabras por segundo.

Lo cual es una velocidad más que suficiente para el sistema.

4.1.5 Desarrollo del módulo EMISOR DATOS RF

Los datos que llegan del codificador necesitan ahora ser enviados por radio frecuencia. Se ha utilizado (como se vio en la introducción) un módulo de corto alcance (SRD) y exento de licencia alguna y que actualmente son producidos por muchos fabricantes.

Estos módulos totalmente probados, aseguran que no habrá interferencias debidas a otros servicios de la banda de 70 cm, contribuyendo también a la repetitibilidad y fiabilidad para la realización de un proyecto.

Concretamente se ha utilizado un módulo por el fabricante CEBEK, el cual es un emisor AM especifico para datos.

4.1.5.1 Descripción del módulo CEBEK

Se trata de un transmisor con oscilador SAW con antena exterior, ideal para aplicaciones donde se quiera modular On-Off una portadora RF con datos digitales.

La antena se ha construido de ¼ de onda (17 cm).

4.1.5.2 Aspecto externo del módulo

El de un módulo híbrido con encapsulado SIL(*Single In Line*), como se muestra en la Figura 38:

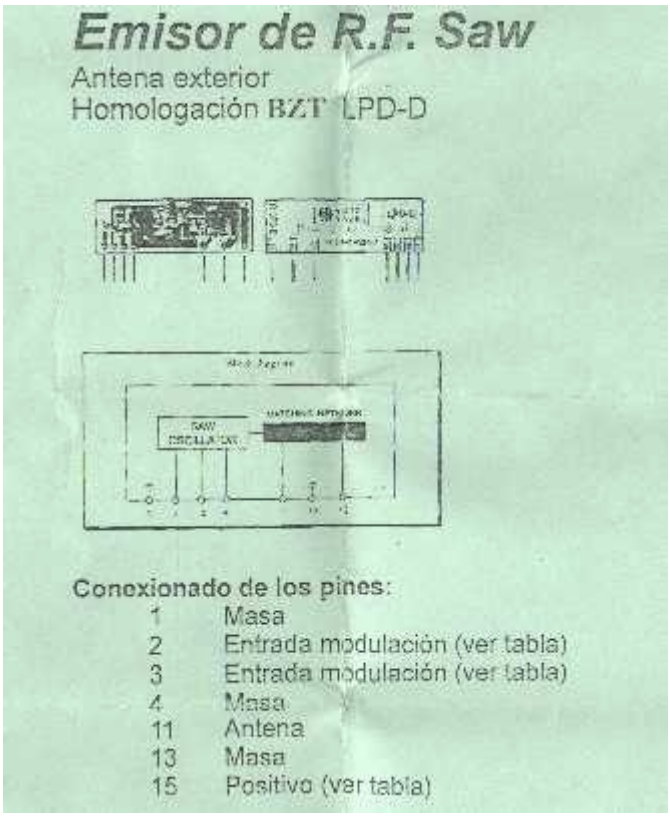


Fig.38

4.1.5.3 Características técnicas del módulo CEBEK

- Circuito híbrido de elevada miniaturización, tipo SIL.
- Frecuencia de trabajo: 433,92 MHz, obtenida mediante un resonador SAW.
- Potencia de salida RF sobre una carga de 50 Ohmios, *pin* 11 (ver tabla).
- Espúreos: -50 dB respecto a la fundamental.
- Frecuencia de modulación: 4 kHz max. (ver tabla).
- Los límites de estabilidad de la Normativa ETS 300 220 son respetados sólo para Vcc<= 5V(ver tabla).
- Formato: in *line*, paso 2,54 mm.
- Dimensiones : 38,1 x 13,2 x 3 mm.

4.1.5.4 Conexionado de los *pines* del módulo

La conexión de los *pines* como se puede apreciar es sencilla y se corresponde con la de la Figura 38, con la salvedad de la "ENTRADA DE MODULACIÓN", que siguiendo la indicación de la siguiente tabla:

Tabla explicativa:

Alimentación Vcc	Pin 2 V	Pin 3 V	Frecuencia modulación kHz	Potencia emitida dBm	Consumo mA
3 - 5	O.C (0 a Vcc)	N.C.	3	3,5 - 8	3,5 - 7,5
5 - 8	N.C.	O.C. (0 a 5)	4	7,5 - 10,5	3,5 - 4
8 - 12	O.C. (0 a 5)	N.C.	4	12 - 15	7,5 - 9,5

Notas: O.C. = onda cuadrada
N.C. = no conectado

Fig. 39

Permite cambiar las prestaciones del sistema en cuanto a ancho de banda, potencia emitida y consumo.

Para el presente proyecto debido a estar optimizado en cuanto a ancho de banda (por el CODIFICADOR) y ya que la potencia de mínimo consumo se ha comprobado "in situ" que cumple con las expectativas, se utilizó la entrada 2 con una tensión de alimentación de 5 V, proporcionadas por una fuente de alimentación.

4.1.6 Desarrollo del módulo RECEPTOR DATOS RF

Los datos transmitidos a 433 MHz son necesarios trasladarlos a banda base para su recepción. Para ello se necesita un receptor adecuado para la frecuencia (433 MHz) y modulación empleada en el transmisor (On-Off Keying).

Se ha utilizado un receptor del mismo fabricante (CEBEK) y especialmente diseñado para el transmisor utilizado en el proyecto.

4.1.6.1 Características del módulo receptor de CEBEK

Receptor de baja absorción, alta inmunidad al ruido de alimentación y radiación en antena.

4.1.6.2 Aspecto externo del módulo receptor

Al igual que el módulo emisor, éste, se presenta como un módulo híbrido con encapsulado SIL (*Single In Line*), como se aprecia en la Figura 40:

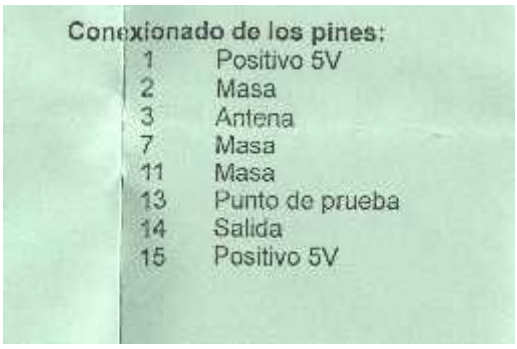


Fig. 40

4.1.6.3 Características técnicas del receptor

- Circuito híbrido de elevada miniaturización, tipo SIL.
- Frecuencia de recepción: 433,92 MHz.
- Recepción de señal con modulación OOK (*On-Off Keying*).
- Sensibilidad RF, medida con señal On-Off en la entrada; mejor de 3µV (-97 dBm) a centro de banda.
- Banda pasante RF a –3 dB: típica 1,2 MHz.
- Alimentación con filtro RC, para eliminar parásitos debidos a circuitos tipo sirena de alarma.
- Antena de ¼ de onda.
- Salida con onda cuadrada, frecuencia máxima de 2 kHz.
- Nivel lógico de salida: bajo en ausencia de señal RF.
- Alimentación: 5 V, consumo máximo 3mA (típico 2,7 mA).
- Radiación en antena <-60 dBm (analizador 50 Ω, filtro FI 100 kHz).
- Tiempo de subida < 2 s

- Formato: *in line,* paso 2,54 mm
- Dimensiones: 38,1 x 13,7 x 5,5 mm.

4.1.6.4 Conexionado de los *pines* del receptor

El conexionado de los *pines* del módulo receptor se corresponde con el indicado en la Figura 40.

El módulo se alimenta a través de las baterías del *Microbot.*

4.1.7 Desarrollo del módulo DECODIFICADOR

Los datos que van saliendo en serie del receptor necesitan, ahora, la respectiva decodificación para volver a la palabra paralela original. Para ello se ha utilizado el módulo dual al MC145026, esto es, el MC145027 que es un decodificador.

El MC145027 recibe la trama serie e interpreta cinco de los dígitos trinarios como un código de dirección. Entonces, 243 direcciones son posibles. Si se usan datos binarios en el codificador, son posibles 32 direcciones. La información serie restante se interpreta como cuatro bits de datos binarios. La transmisión válida ("VT") cambia a nivel alto en el MC145027 cuando se cumplen dos condiciones. La primera, dos direcciones deben ser recibidas consecutivamente (en una secuencia de codificación), las cuales deben ser iguales a la dirección local seleccionada en el decodificador. La segunda, los 4 bits de datos (al igual que las direcciones) deben ser iguales a los últimos datos válidos recibidos. La activación de "VT" indica que la información en la salida de datos ha sido actualizada.

Fig. 41

4.1.7.1 Características operativas del MC145027

Este decodificador recibe los datos serie del codificador y da salida a los datos, si estos son válidos. Los datos transmitidos, consisten de dos palabras idénticas, examinadas bit a bit durante la recepción. Los primeros cinco dígitos trinarios son asumidos como dirección. Si la dirección recibida coincide con la dirección local, los siguientes cuatros bits (datos) son almacenados internamente, pero no son transferidos al *latch* de datos de salida. Cuando la segunda palabra codificada se recibe, la dirección

debe ser otra vez igual a la local. Si esto ocurre, los nuevos bits de datos son chequeados con los previos almacenados. Si los dos *nibbles* de datos (cuatro bits cada uno) son iguales, los datos son transferidos al *latch* de datos de salida y se mantienen en éste hasta que unos nuevos datos lo reemplacen. A su vez, la salida VT se lleva a nivel alto y permanece así hasta que se recibe un error, o hasta que no se recibe señal de entrada en cuatro períodos de datos (Figura 34).

4.1.7.2 Descripción de los *pines*

A1-A5
Entradas de dirección (*Pins* 1-5)

Estas son las entradas de dirección local. Los estados de estos *pins* deben ser iguales a la entrada del codificador apropiado para que el *pin* VT suba a nivel alto. La dirección local puede ser codificada con datos trinarios o binarios.

D6-D9
Salida de Datos(*Pins* 15,14,13,12)

Estas salidas presentan la información binaria que se codificó en las entradas A6/D6 a A9/D9. Sólo son reconocidos los datos binarios; un trinario abierto en el codificador MC145026 se decodifica como un nivel alto (1 lógico).

Din
Data In (*Pin 9)*

Este *pin* es la entrada serie al decodificador. El voltaje de entrada debe ser con niveles CMOS. La fuente de señal que maneja este *pin* debe estar acoplada en DC.

R1, C1
Resistor 1, Capacitor 1 (*Pins* 6,7)

Ver Figura 37.

R2/C2
Resistor 2/Capacitor 2 (*Pin* 10)

Ver Figura 37.

VT
Salida Transmisión Válida (*Pin* 11)

Esta salida de transmisión válida se pone en alto después de la segunda palabra de una secuencia de decodificación cuando se satisfacen las siguientes condiciones:

1.La dirección recibida de ambas palabras iguala a la dirección local del decodificador, y
2.Los bits de datos de ambas palabras se igualan.

VT permanece alto hasta que no se produce igualdad entre dos palabras consecutivas o no se recibe señal en cuatro períodos de datos.

Vss
Tensión Negativa.

El potencial más negativo. Este *pin* esta normalmente a masa.

Vdd
Tensión más positiva (*Pin* 16)

El potencial más positivo.

4.1.7.3 Diagrama eléctrico del DECODIFICADOR

El diagrama eléctrico del DECODIFICADOR es el de la Figura 42. Los valores de resistencias y condensadores están reflejados en la tabla de la Figura 37, para la frecuencia de 1.71 KHz (Ver CODIFICADOR). La dirección debe ser la misma que la del codificador.

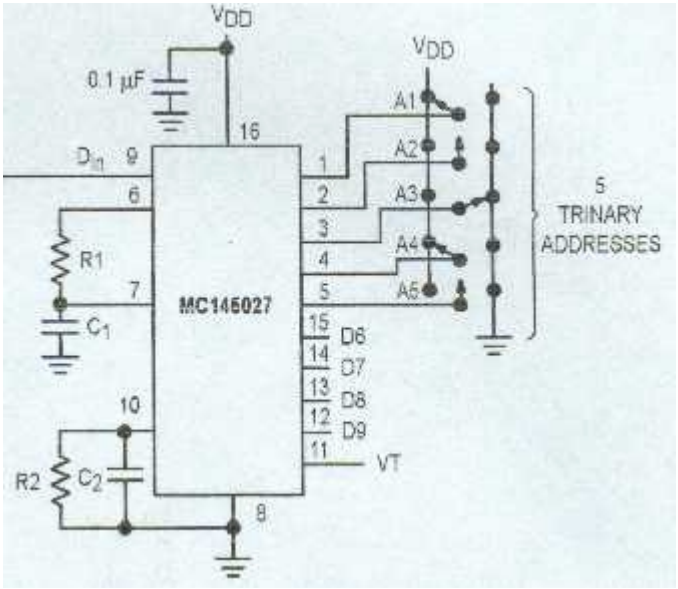


Fig. 42

En la salida de datos se tiene la palabra original (salida del PC), y validada por la señal VT. Este *bus* de cinco bits esta conectado al *Microbot*. Están pues, los datos listos para su proceso por el Robot.

4.1.8 Desarrollo del módulo ACOND. SEÑAL (RX)

Como se explica en la introducción el *Microbot Tritt* está formado por dos placas: la CT6811 como sistema de control y la tarjeta CT293+ como *driver* de potencia, para controlar dos servo motores.

4.1.8.1 Las entradas digitales de la CT293+

Se tiene a la entrada de este bloque la palabra binaria (*nibble*) original, más un bit de transmisión válida (“PALABRA RX”). Este *bus* de cinco bits se ha conectado a las entradas digitales de la CT293+.

Estas entradas se presentan con unas resistencias de *pull-up* y las limitaciones de impedancia siguientes:

1) No se debe poner una resistencia serie con la entrada superior a 10kΩ.
2) Una impedancia de carga a la entrada demasiado pequeña que produce una corriente de entrada superior a los 25 mA, puede dañar permanentemente la entrada utilizada del 68hc11. Se evita colocando una resistencia que haga que la corriente de entrada sea inferior a los 25 mA.
3) Introducir una tensión negativa por la entrada cuando hay una impedancia de carga de bajo valor conectada a ella, puede dañar permanentemente la entrada utilizada del 68hc11. La mejor forma de evitar esto es no introducir nunca tensiones negativas. Tampoco en VRL.

Es decir, si cada entrada digital se presenta de la forma:

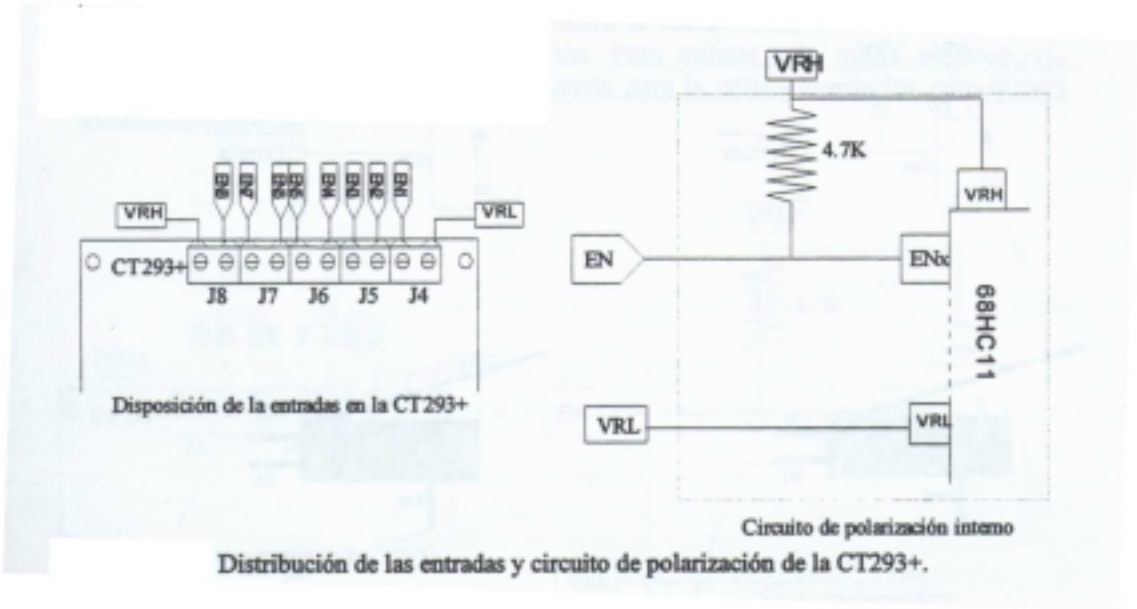


Fig. 43

El problema se presenta para los “1”, ya que si solo hubiera “0” y alta impedancia en las entradas no habría problema (Véase ejemplo del *bumper* en la introducción).

Se ha optado entonces en la colocación en serie de una resistencia (por cada entrada) de valor tal que garantiza los niveles lógicos “1” y “0” y respeta las especificaciones de impedancia, esto es, para cada entrada se ha colocado una resistencia de 1kΩ.

4.1.9 Desarrollo de los módulos PROGRAMA ENSAMBLADOR TRADUCTOR y CT293 + MOTORES.

En este último bloque se ejecuta la orden inicial mandada por el usuario.

Se tiene en el Puerto E del 68hc11 a través del Bloque E de la CT293 la palabra de salida del PC ("PALABRA TX"), ahora hay que traducirla a comandos para que el *Microbot* realice los movimientos requeridos.

Esta función la realiza el módulo PROGRAMA ENSAMBLADOR TRADUCTOR, mediante un programa que puede estar en la RAM ó en la EEPROM, y cuyo código se detalla a continuación: (El código esta debidamente comentado para su compresión):

```
;programa ensamblador del proyecto

        ORG $0
PORTA   equ $0
PORTE   equ $A

        LDX #$1000      ;carga $1000 en el registro X
inicio  STAA PORTA,X    ;contenido del acum A al puerto A
        BRSET PORTE,X $10 comando   ;Si 5ºbit a 1-->transmsión válida--
>salta comando
        BRA inicio                  ;Si no vuelve a inicio

comando BRSET PORTE,X $0F para      ;si comando paro-->salta para
        BRSET PORTE,X $01 delante    ;si comando adelante-->salta
adelante
        BRSET PORTE,X $02 detras     ;si comando detrás-->salta detras
        BRSET PORTE,X $04 izquier    ;si comando izquierda-->salta
izquier
        BRSET PORTE,X $08 derecha    ;si comando derecha-->salta
derecha
        BRA inicio                  ; si no es ningun comando vuelve a
inicio

delante LDA #$18
        BRA inicio

detras  LDA #$78
        BRA inicio

izquier LDA #$38
        BRA inicio

derecha LDA #$58
        BRA inicio

para    LDA #$00
        BRA inicio

        END
```

Este código lo que hace simplemente es leer del Puerto E ($100A) la palabra a traducir y si la transmisión es válida, en función de la palabra (*nibble)* se escribe en el Puerto A ($1000), la dirección y sentido de los motores, según el siguiente código:

Adelante→0x01
Detrás→0x02
Izquierda→0x04
Derecha→0x08
Paro→0x0F

La palabra de mando escrita en el Puerto A ("ACCIONAMIENTO DE MOTORES"), la utiliza el módulo *driver* CT293 + MOTORES, para transmitir las señales de energía necesaria para el funcionamiento adecuado de los motores.

4.2 Desarrollo del sistema completo

En el apartado 4.1 se a descrito el desarrollo de cada uno de los módulos del sistema. Una vez en este punto y con capacidad de comprender el funcionamiento y estructura de los respectivos bloques del sistema, se va a presentar el sistema completo mediante dos apartados. Por un lado se mostrará el diagrama eléctrico del sistema completo y por otro el diseño físico del sistema (prototipo).

4.2.1 Diagrama eléctrico del sistema completo

El diagrama eléctrico del sistema completo, se muestra en la siguiente figura:

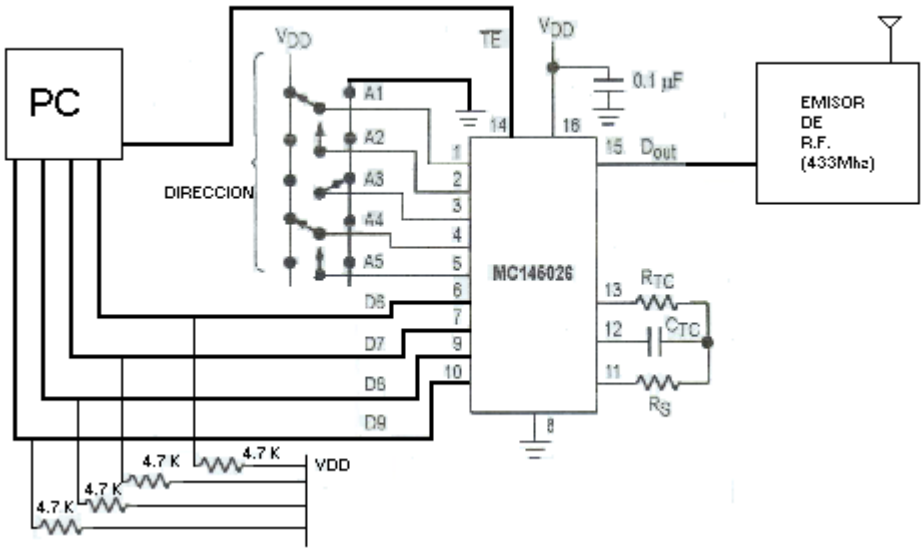


Figura 44: Parte transmisora

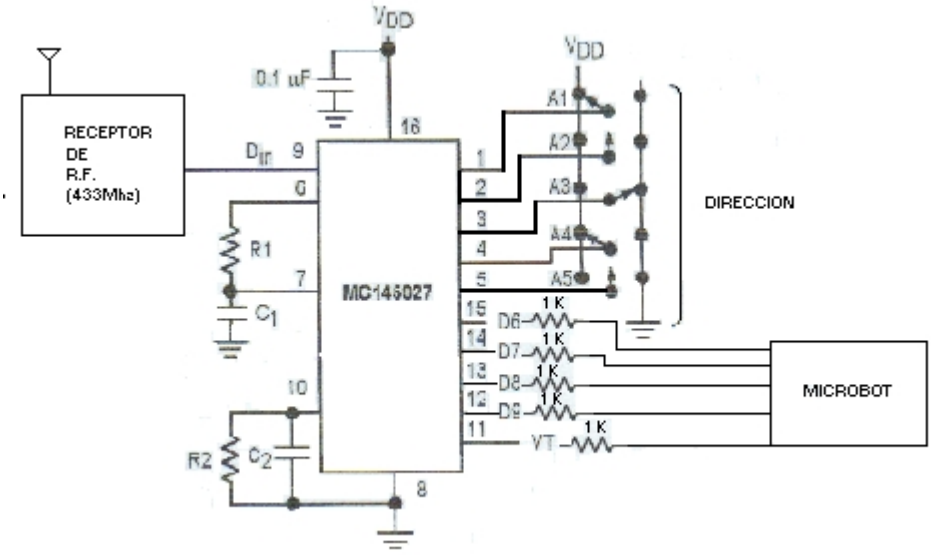


Figura 44: Parte receptora

4.2.2 Diseño físico del sistema completo

La estructura física del sistema completo se divide en dos partes, implementadas cada una en una placa de circuito impreso: la parte transmisora y la parte receptora. Ambas placas han sido construidas mediante la técnica de *wire wrapping.*

La placa transmisora va conectada al PC mediante un cable de extensión del Puerto Paralelo (DB25) y está alimentada mediante una fuente de alimentación.

La placa receptora como se exigía en las especificaciones, tiene las mismas dimensiones físicas que la CT6811 y la CT293+, esta placa de circuito impreso esta alimentada a través de las pilas del *Microbot* y se adhiere a la estructura de éste mediante tornillos separadores.

CAPITULO 5. PRUEBAS Y VERIFICACIÓN

Para la verificación del sistema se han realizado varias pruebas, que se pasan a detallar y están en orden cronológico:

1ª) Al terminar de construir cada una de las placas se ha utilizado un polímetro funcionando como óhmetro para la verificación de continuidad de pistas y de cortocircuitos.

2ª) Previa conexión del PC y *Microbot* se ha comprobado que la transmisión y recepción eran correctas mediante el siguiente circuito:

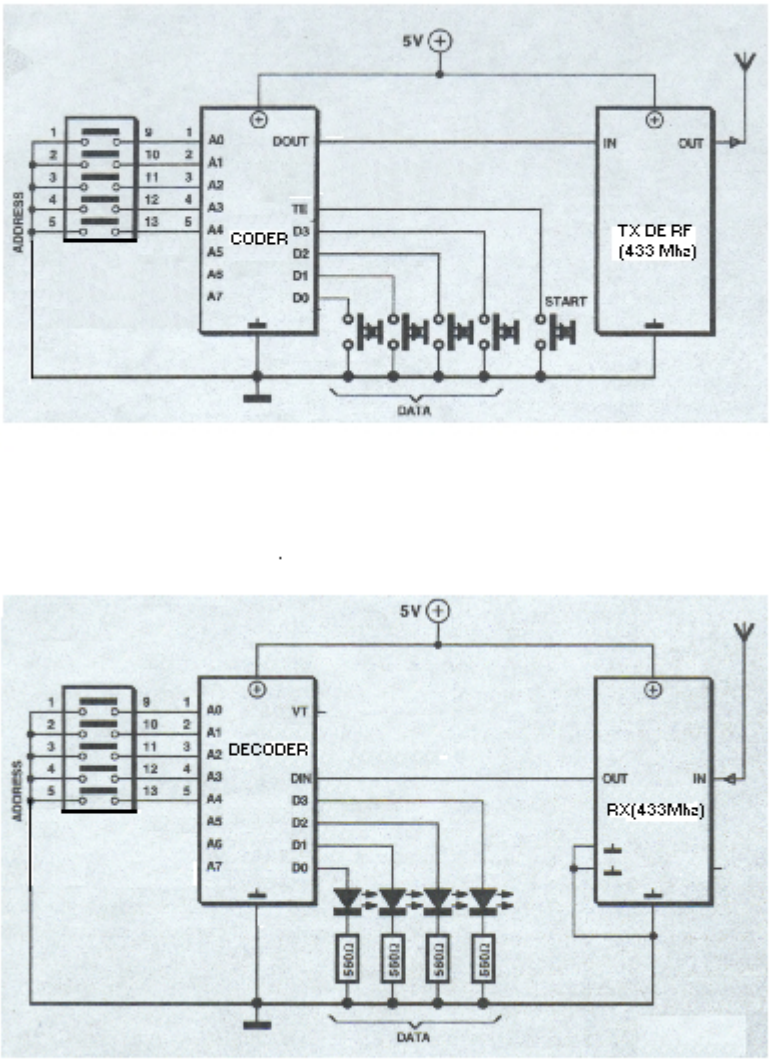


Fig.45

Como se puede observar en la figura se seleccionaron palabras a transmitir mediante unos interruptores conectados a la entrada de datos y la transmisión fue correcta, en el receptor se encendieron los diodos *led* (conectados a la salida de datos) correspondientes a la palabra seleccionada en el lado transmisor.

3ª) La tercera prueba ha consistido en eliminar el interruptor de entrada de datos de la Figura 45 y conectar ahí el puerto paralelo (con las resistencias de *pull-up* correspondientes) del PC, para probar así el software del sistema.

Se mandaron diferentes palabras (comandos) por el puerto y se observaron los resultados en los diodos *leds* del receptor, los cuales se fueron encendiendo con el código del comando correspondiente.

4ª) La cuarta prueba ha consistido en, una vez el sistema completo montado, probar que el programa que corre en el *Microbot* funciona correctamente. Se mandaron comandos (izquierda, derecha,...), y el *Microbot* los ejecutó correctamente.

5ª) Esta prueba ha consistido en repetir la prueba 4ª pero a diferentes distancias de emisor-receptor, para comprobar el alcance del sistema, obteniéndose unos resultados de varias decenas de metros, lo cual es más que suficiente.

6ª) Por último se han probado las prestaciones "dinámicas" del sistema, esto es, mandar una cantidad de comandos en un espacio de tiempo reducido. Para ello se ha utilizado un programa en C, el cual transmite una serie de comandos en intervalos de tiempo tal, que el *Microbot* realiza un movimiento en zigzag, trazando en su recorrido el movimiento de una onda senoidal. El código del programa es el siguiente:

```
#include "C:\LW\INCLUDE\lwsystem.h"
#include "C:\LW\INCLUDE\utility.h"
#define GIRO 1.6
#define MEDIOGIRO 0.6
#define CUARTOMETRO 5.0
#define PUERTO 0x378
#define ADELANTE 0x01
#define PARO 0x0F
#define IZQUIERDA 0x04
#define DERECHA 0x08
#define PARO 0x0F

void maniobra(int direccion,double tiempo);
void enviar(int comando);

main()
{
maniobra(PARO,1.0);
maniobra(IZQUIERDA,MEDIOGIRO);
maniobra(ADELANTE,CUARTOMETRO);
maniobra(DERECHA,GIRO);
maniobra(ADELANTE,CUARTOMETRO);
maniobra(IZQUIERDA,MEDIOGIRO);
maniobra(PARO,1.0);
```

```
}
void enviar(int comando)
{
      outp(PUERTO,comando);
}

void maniobra(int direccion,double tiempo)
{
      double t1,t2;

      t1=Timer();
      while ((t2-t1) < tiempo){
      enviar(direccion);
      t2=Timer();
      }
}
```

Este programa utiliza el tiempo estimado de giro del *Microbot*, para controlar el ángulo de giro y mandar el siguiente comando, esta operación la realiza la función *maniobra*, apoyándose en la función *Timer()* de la librería *utility* del *Labwindows.*

El *Microbot* ejecutó todas las órdenes de manera satisfactoria.

PRESUPUESTO

A continuación se detallan los materiales utilizados así como su precio unitario y el coste final del sistema:

DESCRIPCIÓN	Cantidad	Precio U	TOTAL
Placa de circuito impreso	1	985	985
C.I. 145026	1	250	250
C.I. 145027	1	250	250
Resistencias varias de ¼ de W.	15	5	75
Condensadores varios	4	20	80
Módulo emisor de datos	1	2500	2500
Módulo receptor de datos	1	2500	2500
Cable, tornillos, ...		500	500
Cable extensión del puerto paralelo	1	1000	1000
		TOTAL	8.140 pta

Este presupuesto es del prototipo realizado. El coste de producción unitario, para una posible producción en serie del sistema sería bastante más barato, al reducirse el coste de los componentes a la mitad o más.

CONCLUSIONES Y LÍNEAS FUTURAS

Conclusiones.

Se ha conseguido culminar con éxito las especificaciones requeridas al principio del proyecto. Se ha logrado crear un sistema físico, perfectamente acondicionado para ser utilizado, incluso mejora los objetivos requeridos a priori en cuanto a prestaciones (alcance, autonomía, fiabilidad,...), y diseño físico (sistema compacto y robusto).

Pese a ser un proyecto laborioso por incluirse en él tanto hardware como software, el esfuerzo ha sido gratificante, ya que en la realización del proyecto se han aplicado gran variedad de conocimientos teóricos y prácticos, previamente adquiridos en el período de formación académica. A su vez se han ampliado estos conocimientos en las siguientes áreas:

Área de diseño: diseño físico de bloques funcionales (codificadores, transmisores,...), así como la interacción entre ellos.

Comunicación de datos: Se han estudiado los diferentes sistemas de transmisión de datos (infrarrojos, AM, FM.....).

Microbot: Se ha hecho un estudio exhaustivo de los *Microbots* (familias, hardware, software...), así como la programación en lenguaje ensamblador del microcontrolador 68hc11 de *Motorola.*

Lenguaje C: Se ha asentado las bases de la programación C (*Labwindows)* y modos de funcionamiento del puerto paralelo del PC.

Líneas futuras

Una de las metas del presente proyecto, como se indicó en el apartado de objetivos era que el sistema fuese modular y abierto, y efectivamente se ha conseguido, debido a que con una simple redundancia de componentes se consiga mejorar las prestaciones del sistema. Se pueden proponer las siguientes líneas futuras:

a) **Comunicación *full-duplex* entre el PC y el *Microbot*.** Con esta comunicación el PC podría procesar información de sensores, transductores, potenciómetros, *switches*,..., ya que el *Microbot* dispone de una reducida memoria para procesar toda esa información.
b) **Sistema *Talker-Listener.*** Un sistema en el cual un ordenador central (*Talker)* ordenara a múltiples *Microbots* (*Listeners)* comandos de forma selectiva para la realización de tarea/s. El ordenador, previo envío del comando, seleccionaría la dirección del *Microbot* en cuestión. Los restantes *Microbots* al "ver" que la dirección no era la suya, ignorarían el comando. Se

pueden hacer dentro de este sistema subconjuntos de dos, tres o más *Microbots*, simplemente igualando sus respectivas direcciones.

c) **Granjas de *Microbots***. Cooperación por RF de los habitantes de una granja de *Microbots*.

REFERENCIAS BIBLIOGRÁFICAS

[web1]. www.mrrobot.com.

[web2]. www.microbotica.es.

[Elek.98]. 418/433 MHz comunicación de pequeño alcance, Revista Elektor Octubre 98.

[Elek.94]. Placa para microprocesador 68HC11, Revista Elektor Abril de 1994.

[García.B.96]. TUTORIAL DE LABWINDOWS. Una guía para el laboratorio de instrumentación. DTE 2/96. Carmen García Berdonés.

[web3]. www.eprat.com/DOCS/parallel.htm.

[web4]. www.fapo.com.

[Moto]. MC145026,MC145027,MC145028*: Encoder and Decoder Pairs.* MOTOROLA. *Semiconductor technical data book.*