Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN AUTOMATICA Y ELECTRONICA INDUSTRIAL

TRABAJO FIN DE GRADO

MICROBOT CONTROLADO POR RADIOFRECUENCIA E INFRARROJOS

***DOCUMENTO** - MEMORIA TECNICA*

Alumno/Alumna: MERELAS URCELAY, ASIER
Director/Directora (1): OLEAGORDIA AGUIRRE, IÑIGO

Curso: 2019-2020

Fecha: Bilbao, 17-Enero-2020

RESUMEN

El Objeto de este trabajo de Fin de Grado es el diseño y la realización de un microbot capaz de ser controlado por radiofrecuencia e infrarrojos. Dada la alta automatización que se presenta en la mayoría de sectores industriales, cada vez nos encontramos con más espacios de difícil acceso para el ser humano y donde, en caso de querer realizar cualquier tipo de mediciones, se hace necesaria la figura de robots de pequeño tamaño, robustos y sensorizados, capaces de transmitir la información recolectada en tiempo real a un dispositivo HMI o a un sistema de captura de información.

En este caso, se ha planteado la posibilidad de realizar un microbot que portara distintos sensores para que, ante una posible intervención en espacios confinados, el técnico pueda evaluar posibles riesgos ambientales antes de entrar y evitar así posibles consecuencias fatales.

Este tipo de microbots se pueden programas además para trabajar de forma autónoma, entrando en áreas restringidas al ser humano para poder detectar posibles anomalías en lugares donde no es posible colocar sensores fijos.

Si además comunicáramos estos dispositivos capaces de moverse de manera autónoma en una planta, con operarios conectados, existen infinidad de casos de uso para mejorar la seguridad y la salud de los empleados de una planta de producción industrial, evitando exponerles a altos niveles de ruido, contaminación, radiación y ayudándoles en caso de alguna emergencia.

LABURPENA

Gradu Amaierako proiektu honen helburua erradiofrekuentziaz eta infragorriez kontrolatu ahal izateko mikrobot bat diseinatzea eta ekitea da. Sektore industrial gehienetan gertatzen den automatizazio handia dela eta, gizakiarentzako sarbide zaila duten espazio gehiago aurkitzen ditugun bakoitzean eta, edozein neurri mota egin nahi badugu, robot txikia, sendo eta sentsorizatuta behar dugu, datuak denbora errealean biltzeko edo datuak biltzeko sistema zentral batera bidaltzeko gai dena.

Kasu honetan, sentsore desberdin dauzkan mikrobota egiteko aukera planteatu da, beraz, espazio mugatuetan interbentzio posibleren bat gertatuz gero, teknikariak ingurumen arriskuak baloratu ahal izan ditu sartu aurretik eta, beraz, ondorio kaltegarriak ekiditeko.

Mikrobota mota hauek modu autonomoan funtziona dezaketen, gizakiei mugatutako eremuetan sartzeko, sentsore finkoak jartzea ezinezkoa den lekuetan, anomaliak antzemateko.

Industrialdean Microbot autonomoa eta konektatutako langilea batera lan egin dezakete eta, kasu honetan, erabilera-kasu ugari daude produkzio-fabrika bateko langileen segurtasuna eta osasuna hobetzeko, zarata maila altuetara ez egoteko, kutsadura, erradiazioa eta larrialdi kasuetan laguntzeko.

ABSTRACT

The purpose of this Final Degree project is the design and realization of a microbot capable of being controlled by radiofrequency and infrared. Given the high automation that occurs in most industrial sectors, we find more and more spaces of difficult access for the human being and where, in case of wanting to make any type of measurements, the figure of small, robust and sensorized robots is necessary, capable of transmitting the information collected in real time to an HMI device or to a data gathering system.

In this case, the possibility of carrying out a microbot that carries different sensors has been considered so that, in the event of a possible intervention in confined spaces, the technician can evaluate possible environmental risks before entering and thus avoid possible fatal consequences.

This type of microbots can also be programmed to work autonomously, entering areas restricted to humans in order to detect possible anomalies in places where it is not possible to place fixed sensors.

If we also communicate these devices, capable of moving autonomously, in a plant, with connected operators, there are countless use cases to improve the safety and health of employees of an industrial production plant, avoiding exposing them to high noise levels, pollution, radiation and helping them in case of an emergency.

INDICE

1. INTRODUCCION

1.1. OBJETIVO DEL PROYECTO

El microbot es una máquina que puede efectuar un diverso número de trabajos automáticamente, mediante la programación previa. Tiene la capacidad de tomar decisiones teniendo en cuenta la información recibida del exterior. Puede incorporar distintos dispositivos y herramientas que le permitan realizar infinidad de tareas

Este proyecto se basa en la comunicación inalámbrica, tanto por infrarrojos como por radiofrecuencia, de dos microcontroladores. Como aplicación se ha diseñado un microbot que sería controlado inalámbricamente. Dicho microbot sería una sonda controlada por un mando capaz de tomar mediciones desde distintos tipos de sensores. Este proyecto se adentra en la construcción de una maqueta prototipo del microbot real el cual contaría con ampliaciones respecto al nuestro que más adelante pasaremos a explicar. La idea es que el microbot sea capaz de realizar mediciones en ambientes hostiles para el ser humano o de difícil acceso. Gracias a la flexibilidad del diseño se pueden incorporar numerosas ampliaciones.

1.1.1. FUNCIONAMIENTO BASICO

Supongamos que se requiere la toma de una medida física en un ambiente hostil para el ser humano, (altas temperaturas, radiación, gases...) o de difícil acceso. Nuestro microbot es capaz de adentrarse en dichos lugares siguiendo nuestras órdenes y llegar al lugar indicado para darnos la lectura de dichas mediciones con tan solo mirar el mando y en tiempo real.

El microbot es controlado mediante un mando por radiofrecuencia o por infrarrojos, a elección del usuario. Según las órdenes transmitidas, el microbot realizara los movimientos q les serán indicados. Cada cierto tiempo realizara mediciones mediante

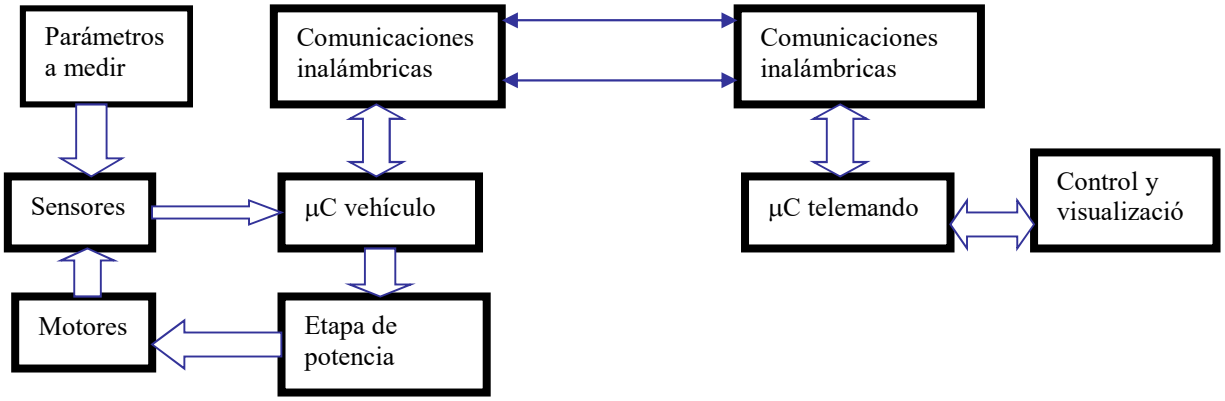
los sensores q serán transmitidas al mando visualizando dichos datos en una LCD o almacenarlos para enviarlos al PC.

Como posible ampliación de funcionamiento el microbot podría funcionar en modo automático. Para dicho modo y debido a que esta máquina está pensada para su uso en ambientes hostiles se descarta la posibilidad de hacerle seguidor de línea ya que para ello previamente deberíamos colocar la línea a seguir. Como posibilidades se barajan el que sea capaz de salir de un laberinto por si solo o la trayectoria programada. Para la trayectoria programada se requiere un teclado matricial y nos permitiría tomar medidas en los lugares exactos donde necesitamos. El robot que sale del laberinto no necesita más que sensores de distancia para detectar los obstáculos y esquivarlos y a la vez nos daría medidas del recinto. En caso de no conocer las dimensiones exactas del lugar a medir sería más conveniente la segunda opción ya que el robot entraría por sí mismo, tomaría mediciones y saldría del recinto por sí mismo.

Otra ampliación seria la comunicación del mando con el PC. Para ello todas las mediciones recibidas por el mando serian almacenadas en el microcontrolador y enviadas al PC cuando estas finalizasen. Creando un entorno virtual tipo Labview podríamos obtener graficas de las mediciones e incluso guardarlas en archivos para su futuro análisis.

1.2. DIAGRAMA DE BLOQUES GENERAL DEL HARDWARE

A continuación se muestra un diagrama general de la interconexión de las placas y del flujo de datos.



1.3. ESPECIFICACIONES DE LOS MODULOS HARWARE

A continuación se explicara cada uno de los módulos hardware tanto del mando como del microbot.

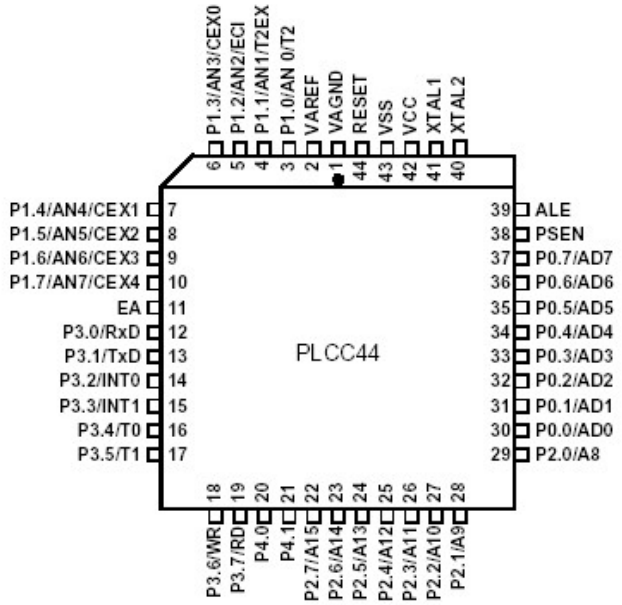
1.3.1. VEHICULO

- **Microcontrolador atmel T89C51AC2.**

Para la realización de este proyecto se ha elegido un microcontrolador frente a un microprocesador debido a que integra en un solo chip tanto la memoria como una gran cantidad de periféricos.
Se ha elegido un microcontrolador Atmel y no un PIC, que es más común, por la experiencia anterior con dicho fabricante y gracias a la facilidad de programación ya que el compilador para C, que es el lenguaje de programación elegido, es mejor que los usados para PICs y la amplia información dada por el fabricante.

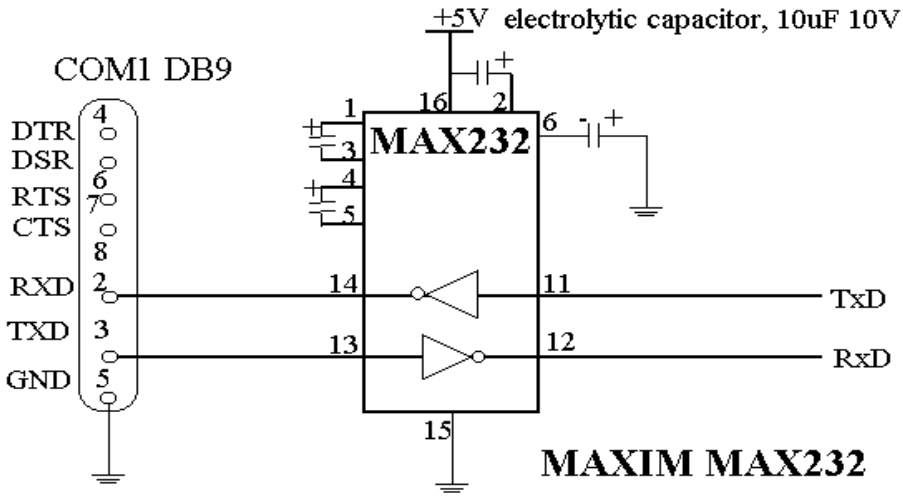
Dentro de los Atmel hemos elegido el T89C51AC2 por tener arquitectura 8051, y a su memoria flash que nos permite una gran flexibilidad de diseño y una programación ISP.
Tiene 32 Kb de memoria de programa flash, 256 bytes de RAM y 1 KB de memoria XRAM,
2Kb de Flash para la bootloader, que es la memoria que se utiliza para la programación mediante puerto serie, 2Kb de EEPROM. Dispone de tres contadores/timers de 16 bits, un ADC de 8 canales de 10 bits de resolución, cinco canales PCA de 16 bits que funcionan como PWM, timers o salidas de alta velocidad, cinco puertos digitales (32 + 2 líneas I/O). La velocidad máxima de trabajo es de 40 Mhz.

En la siguiente figura se muestra la distribución y denominación de los pines del microcontrolador:



- **Programador**

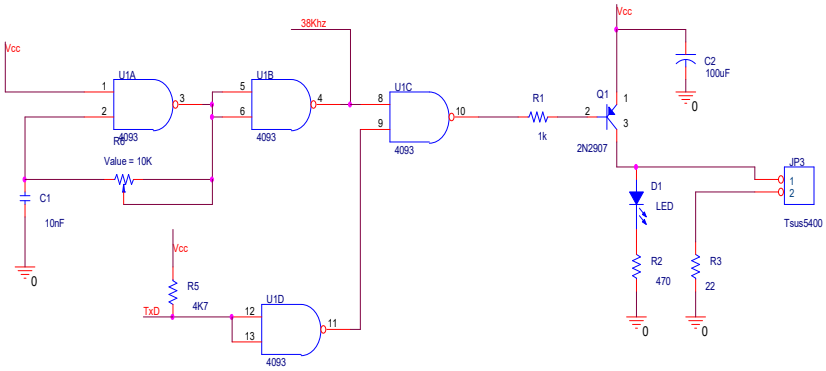
Para la programación del micro utilizamos el MAX232 con el puerto serie del PC. Gracias a la bootloader de este microcontrolador tenemos la posibilidad de programarlo mediante la UART lo cual nos permite una mayor flexibilidad y programación en el propio sistema.



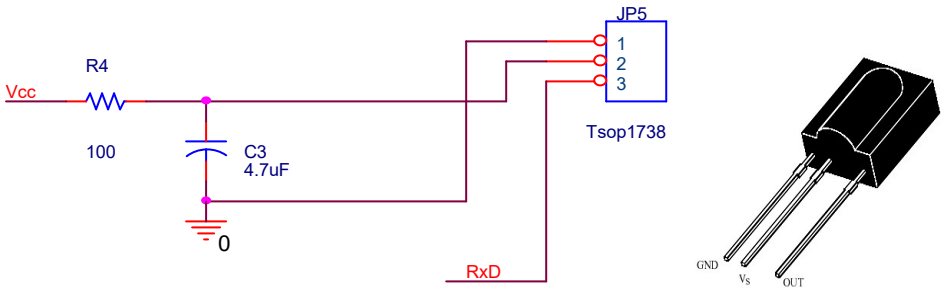
- **Infrarrojos**

Para la comunicación mediante infrarrojo utilizamos el diodo emisor IR TSUS 5400 modulándolo a 38 Khz mediante puertas NAND. Como receptor se usa el integrado TSOP 1638 el cual demodula por si solo la señal recibida. La comunicación entre estos dispositivos y el microcontrolador se realiza mediante la UART.

-Emisor

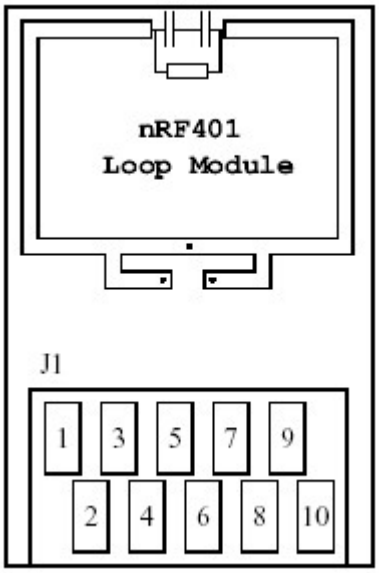


-Receptor



- **Radiofrecuencia**

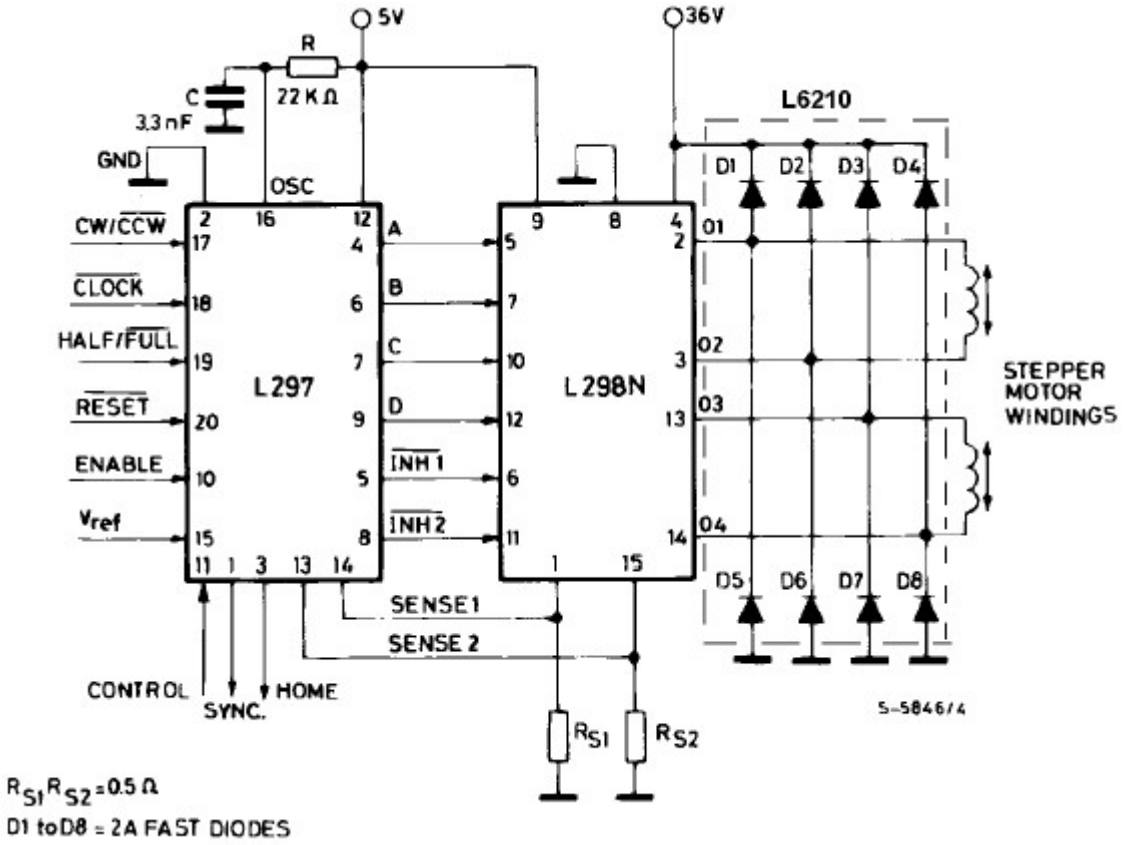
La comunicación por radiofrecuencia se realiza con dos transceptores que se encargan de todo el proceso de modulación y demodulación de la señal. Este es el nRF401 Loop kit de NVLSI ASA que ya viene implementado y con la antena integrada. También utilizan la UART para la comunicación.



Pin	Nombre	Descripción
1	TXEN	Habilitación de transmisión TXEN=1 => modo transmisión TXEN=0 => modo recepción
2	PWR_UP	Power on/off PWR_UP=1 => ON PWR_UP=0 => OFF
3	GND	0V
4	CS	Seleccion de canal CS=0 => 433.92 MHz Canal1 CS=1 => 434.33 MHZ Canal2
5	GND	0V
6	DIN	Entrada de dato
7	GND	0V
8	DOUT	Salida de dato
9	GND	0V
10	VDD	Alimentación (+3-5V DC)

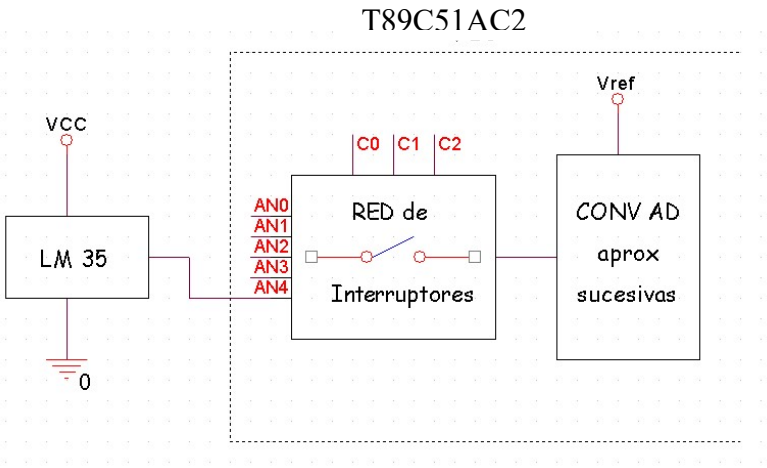
- **Potencia**

Dos motores elegidos para gobernar el microbot son dos motores de paso a paso bipolares. Para su control se utiliza el L297 y el puente en H L298N.



- **Sensores**

Los sensores a utilizar podrán ser elegidos por el usuario final. Nosotros como ejemplo utilizamos el montaje de un sensor de temperatura LM35. Debido a la flexibilidad del producto también podrían utilizarse toda clase de sensores resistivos, como LDRs, NTCs.



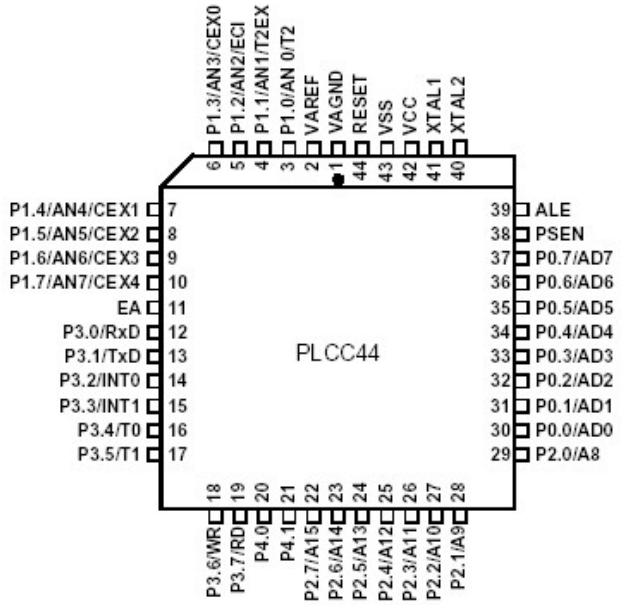
1.3.2. MANDO

- **Microcontrolador atmel T89C51AC2.**

Para la realización de este proyecto se ha elegido un microcontrolador frente a un microprocesador debido a que integra en un solo chip tanto la memoria como una gran cantidad de periféricos.
Se ha elegido un microcontrolador Atmel y no un PIC, que es más común, por la experiencia anterior con dicho fabricante y gracias a la facilidad de programación ya que el compilador para C, que es el lenguaje de programación elegido, es mejor que los usados para PICs y la amplia información dada por el fabricante.

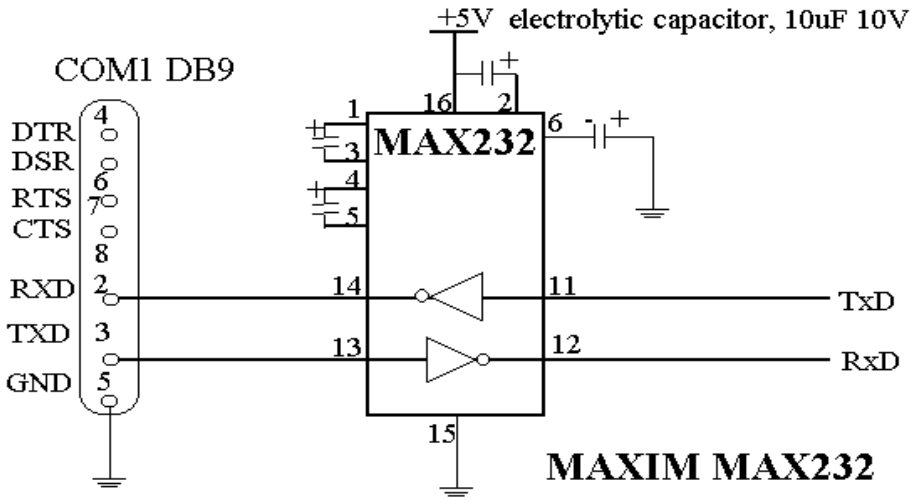
Dentro de los Atmel hemos elegido el T89C51AC2 por tener arquitectura 8051, y a su memoria flash que nos permite una gran flexibilidad de diseño y una programación ISP.
Tiene 32 Kb de memoria de programa flash, 256 bytes de RAM y 1 KB de memoria XRAM,
2Kb de Flash para la bootloader, que es la memoria que se utiliza para la programación mediante puerto serie, 2Kb de EEPROM. Dispone de tres contadores/timers de 16 bits, un ADC de 8 canales de 10 bits de resolución, cinco canales PCA de 16 bits que funcionan como PWM, timers o salidas de alta velocidad, cinco puertos digitales (32 + 2 líneas I/O). La velocidad máxima de trabajo es de 40 Mhz.

En la siguiente figura se muestra la distribución y denominación de los pines del microcontrolador:



- **Programador**

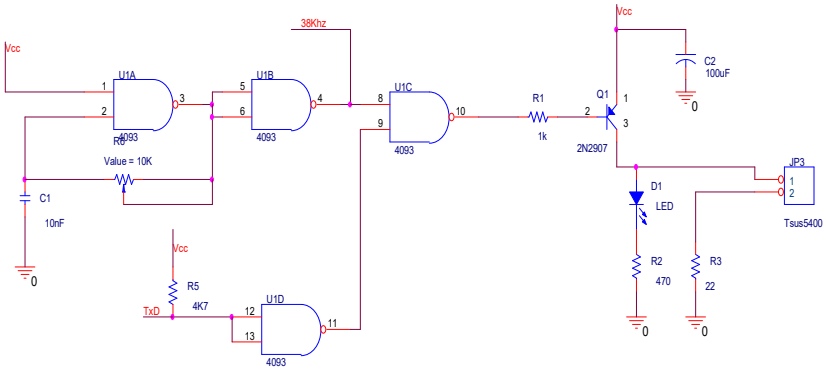
Para la programación del micro utilizamos el MAX232 con el puerto serie del PC. Gracias a la bootloader de este microcontrolador tenemos la posibilidad de programarlo mediante la UART lo cual nos permite una mayor flexibilidad y programación en el propio sistema.



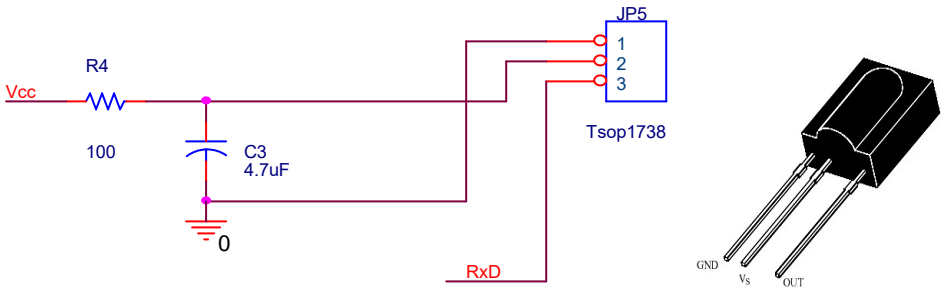
- **Infrarrojos**

Para la comunicación mediante infrarrojo utilizamos el diodo emisor IR TSUS 5400 modulándolo a 38 Khz mediante puertas NAND. Como receptor se usa el integrado TSOP 1638 el cual demodula por si solo la señal recibida. La comunicación entre estos dispositivos y el microcontrolador se realiza mediante la UART.

-Emisor

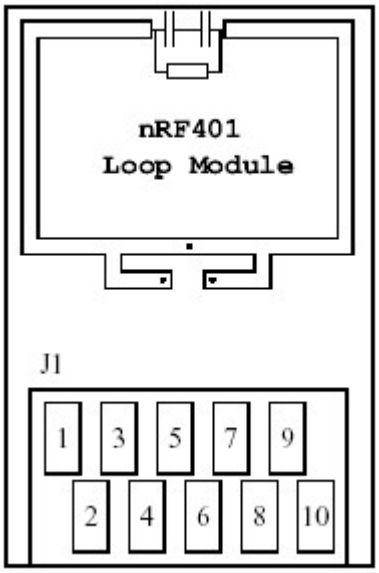


-Receptor



- **Radiofrecuencia**

La comunicación por radiofrecuencia se realiza con dos transceptores que se encargan de todo el proceso de modulación y demodulación de la señal. Este es el nRF401 Loop kit de NVLSI ASA que ya viene implementado y con la antena integrada. También utilizan la UART para la comunicación.



Pin	Nombre	Descripción
1	TXEN	Habilitación de transmisión TXEN=1 => modo transmisión TXEN=0 => modo recepción
2	PWR_UP	Power on/off PWR_UP=1 => ON PWR_UP=0 => OFF
3	GND	0V
4	CS	Seleccion de canal CS=0 => 433.92 MHz Canal1 CS=1 => 434.33 MHZ Canal2
5	GND	0V
6	DIN	Entrada de dato
7	GND	0V
8	DOUT	Salida de dato
9	GND	0V
10	VDD	Alimentación (+3-5V DC)

- **Interfaz con el usuario**

El dispositivo de control ira dotado de una pantalla LCD de 2 líneas x 16 caracteres y de dos potenciómetros de 10K que sirven de control.

El LCD se compone de 8 líneas de datos y 3 palabras de control (escritura, lectura y habilitación) que son enviadas de forma paralela por el micro debidamente programado para esta tarea.

Los mensajes se visualizan en la LCD introduciendo los correspondientes códigos ASCII de cada uno de los caracteres a visualizar. Además, un extenso juego de caracteres predefinidos de fábrica permite usar nuevos caracteres y símbolos definidos por el usuario. A pesar de que él número de modelos comerciales es muy grande, las líneas necesarias para su conexión y control son prácticamente las mismas.

Los códigos ASCII de los caracteres se introducen por las 8 líneas de la Puerta B, mientras que las señales de control se aplican mediante 3 líneas de la Puerta A. RA1 corresponde con la señal de control R/W#, que indica si el modulo LCD va a ser leído o escrito. RA2 se aplica a la entrada de activación E (Enable), con la que se indica si el modulo está activado o no. Finalmente con la línea RA0 se controla la señal RS, que permite el acceso a los distintos registros de control del módulo para establecer las condiciones de la visualización. El programa de control residente en el PIC debe encargarse de gestionar las 3 señales de control y el envío de los caracteres ASCII por la Puerta B.

Los potenciómetros irán al ADC del microcontrolador dando una señal analógica de control sobre los motores.

1.3.3. TRACCION

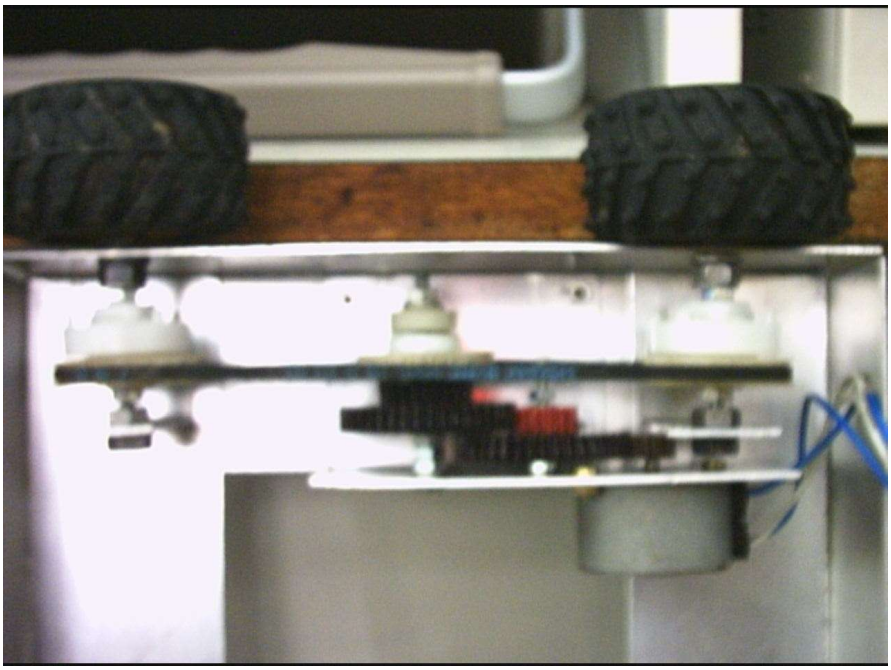
Para la tracción del vehículo se han utilizado dos motores paso a paso que a través de unas reductoras para ganar par motor y un sistema de correas hacen girar cada uno de ellos las dos ruedas de su lateral proporcionando una tracción a las cuatro ruedas similar a la tracción de un tanque.

Para el giro del vehículo únicamente es necesario hacer girar un lado más rápido que el otro consiguiendo que el vehículo gire hacia el lado donde las ruedas giran más despacio.

A continuación se muestran imágenes del sistema de tracción y de su correspondiente reductora.

- **REDUCTORA**

- **TRACCIÓN COMPLETA**

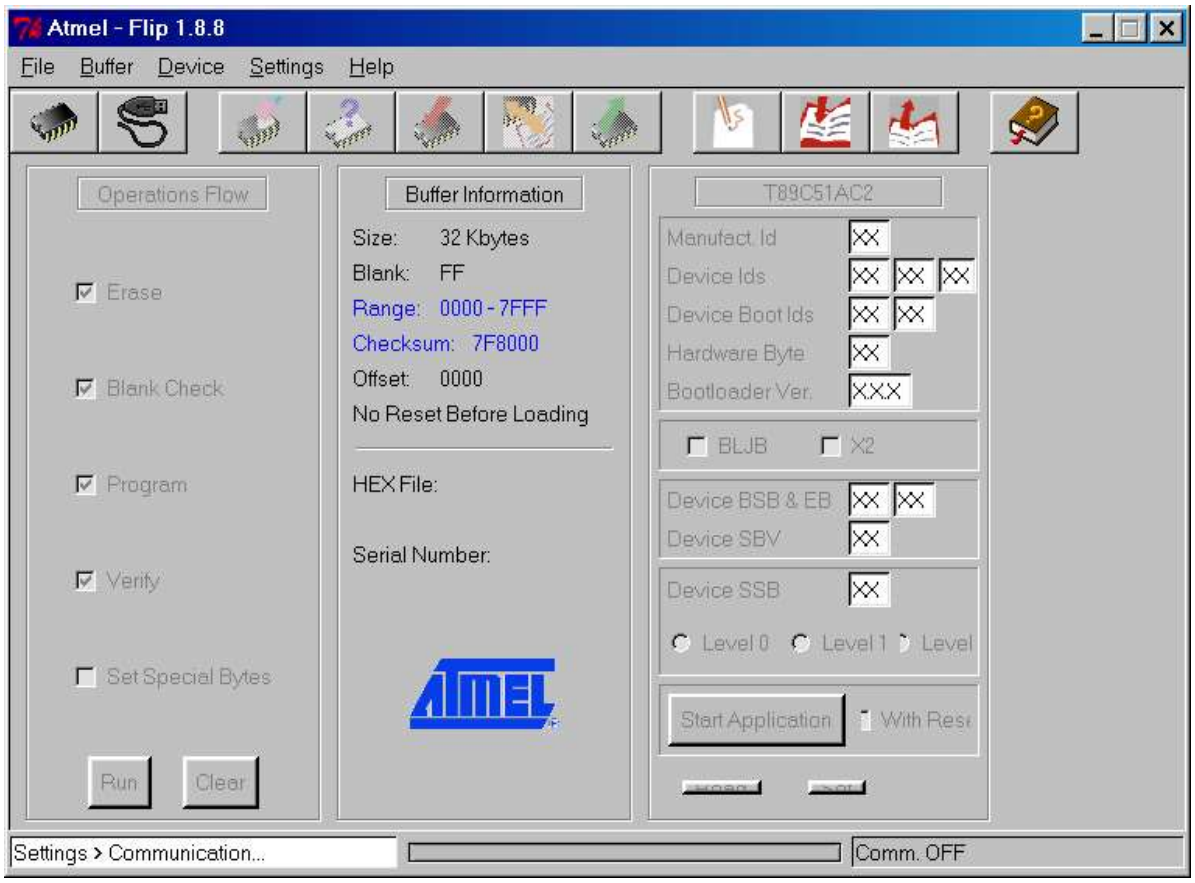


1.4. ESPECIFICACIONES DEL SOFTWARE

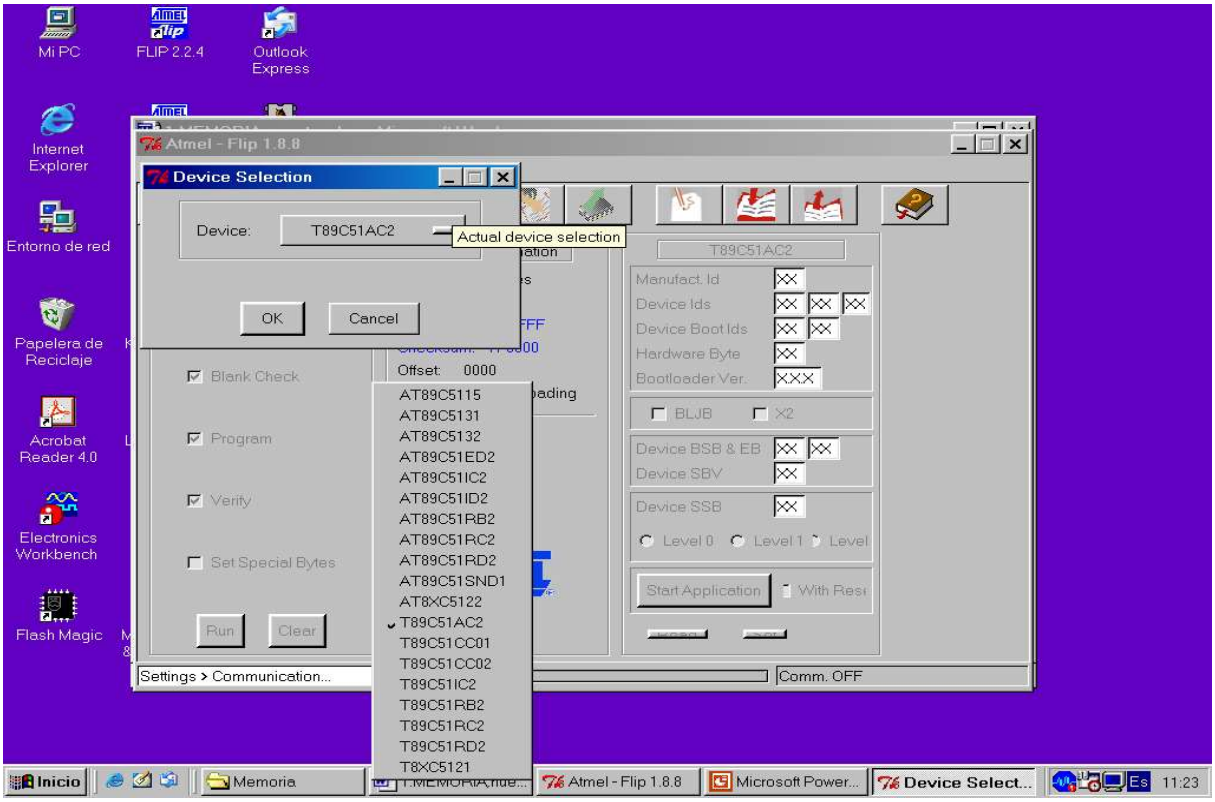
1.4.1. ENTORNO DE PROGRAMACIÓN

Como entorno de programación se usara el Flip 1.8.8. Este programa lo da el fabricante del microcontrolador atmel y permite la programación del micro mediante el uso de la memoria Bootloader.

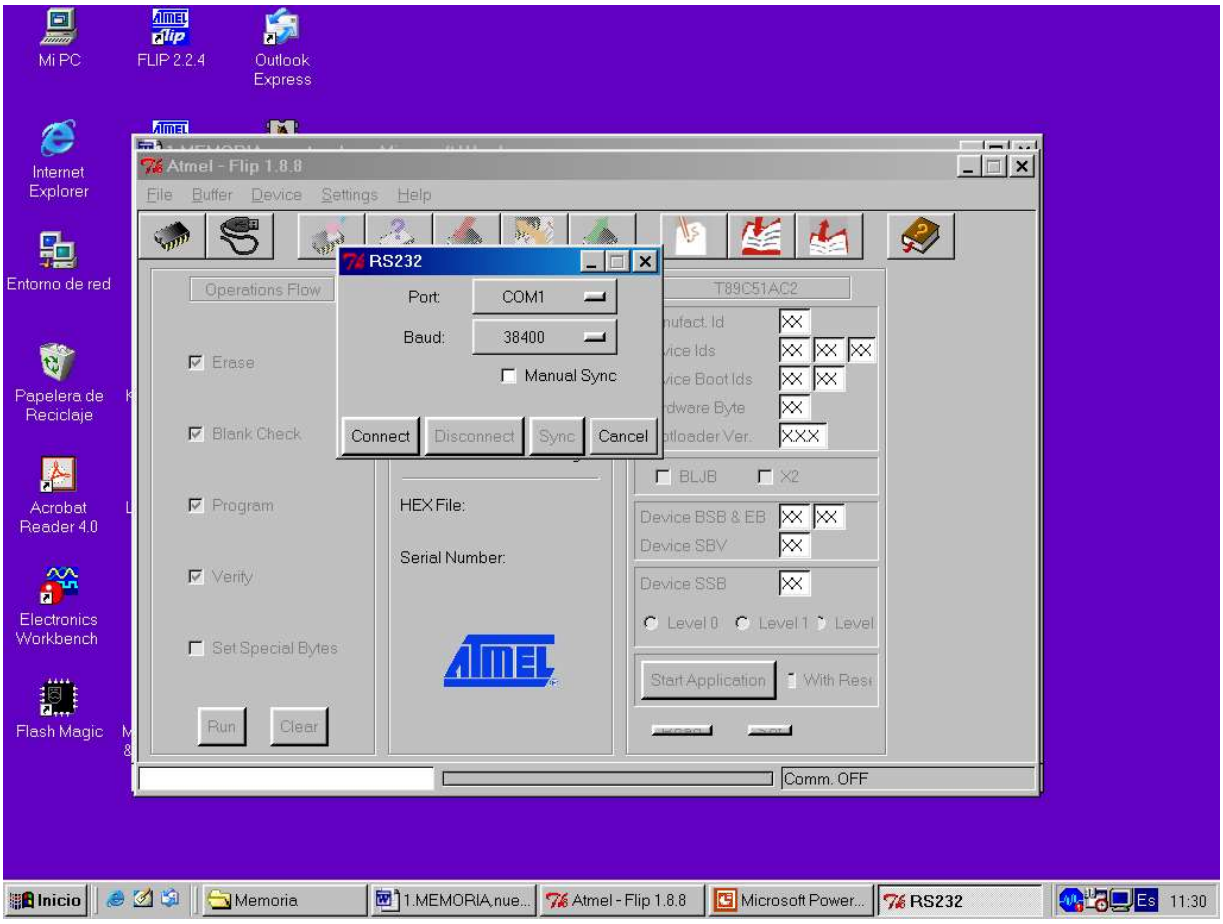
A continuación se muestra una imagen de la pantalla principal y una explicación básica de su funcionamiento:



El funcionamiento del programa es muy sencillo. Lo primero será elegir el dispositivo. Para ello se despliega el menú Device>select (o F2) con lo que se desplegara el menú de selección de componentes.



Lo siguiente es seleccionar el tipo de comunicación que se realizara con el micro. Se puede elegir entre RS-232, USB y bus CAN. Para este proyecto se utilizara RS-232. Para la selección de las comunicaciones bastara con ira al menú Settings> comunication >RS-232 para nuestro caso. Esto desplegara una ventana que permite elegir la velocidad de la comunicación y nos permite conectarnos.

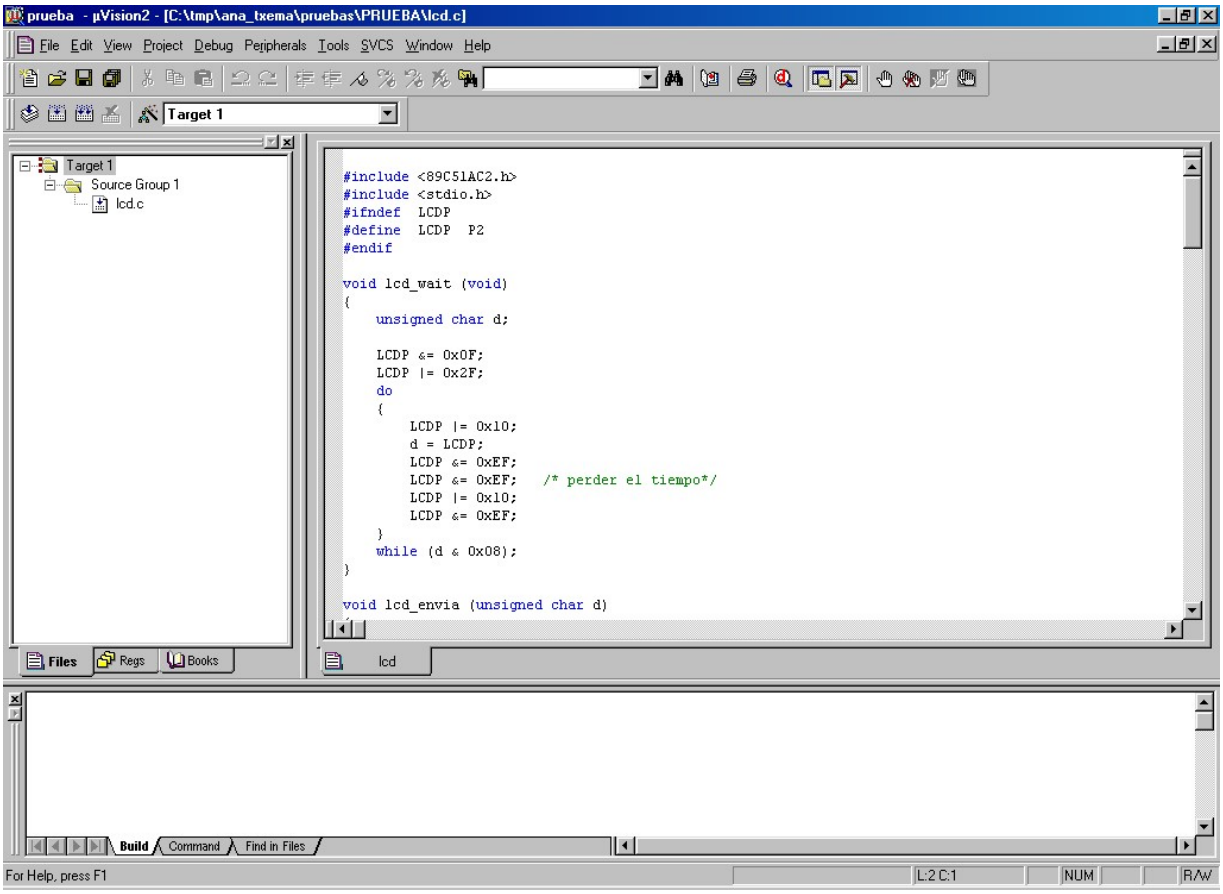


Tras darle a connect el microcontrolador está listo para recibir el programa .Este se debe cargar en el flip en el menú File > load .HEX. Tras esto podremos leer el dispositivo programarlo o borrarlo.

1.4.2. ENTORNO DE DISEÑO

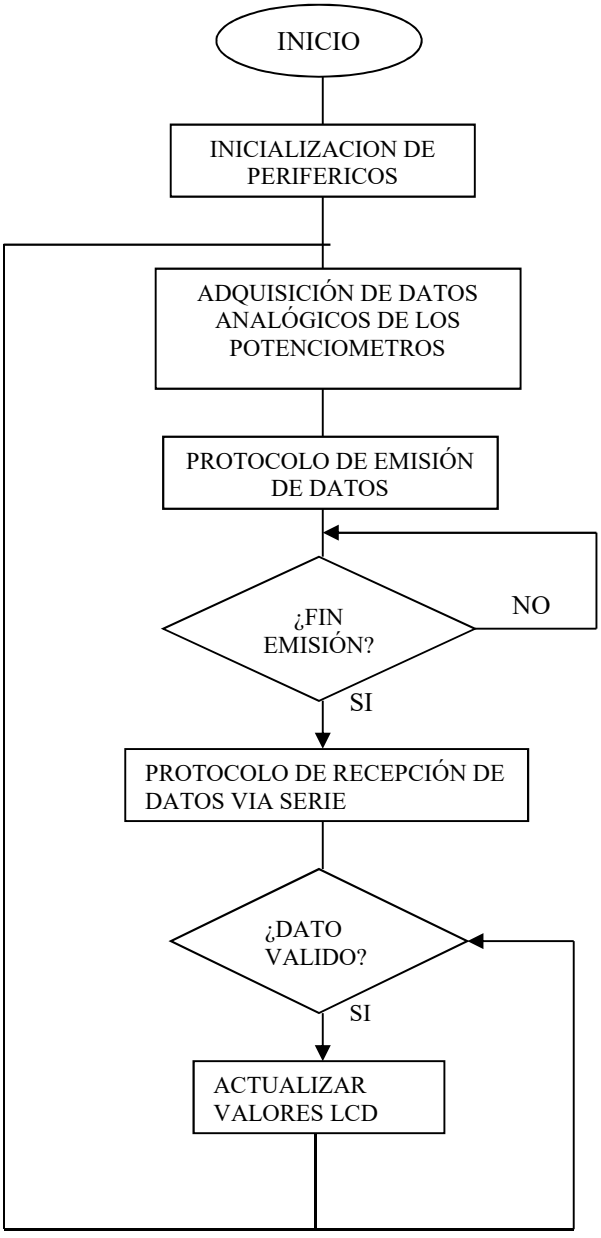
Como entorno de diseño utilizaremos el programa uVision de Keil, que es un compilador en C para micros de la familia 8051. Este programa genera el archivo HEX necesario para la programación.

En la siguiente imagen vemos su pantalla principal:

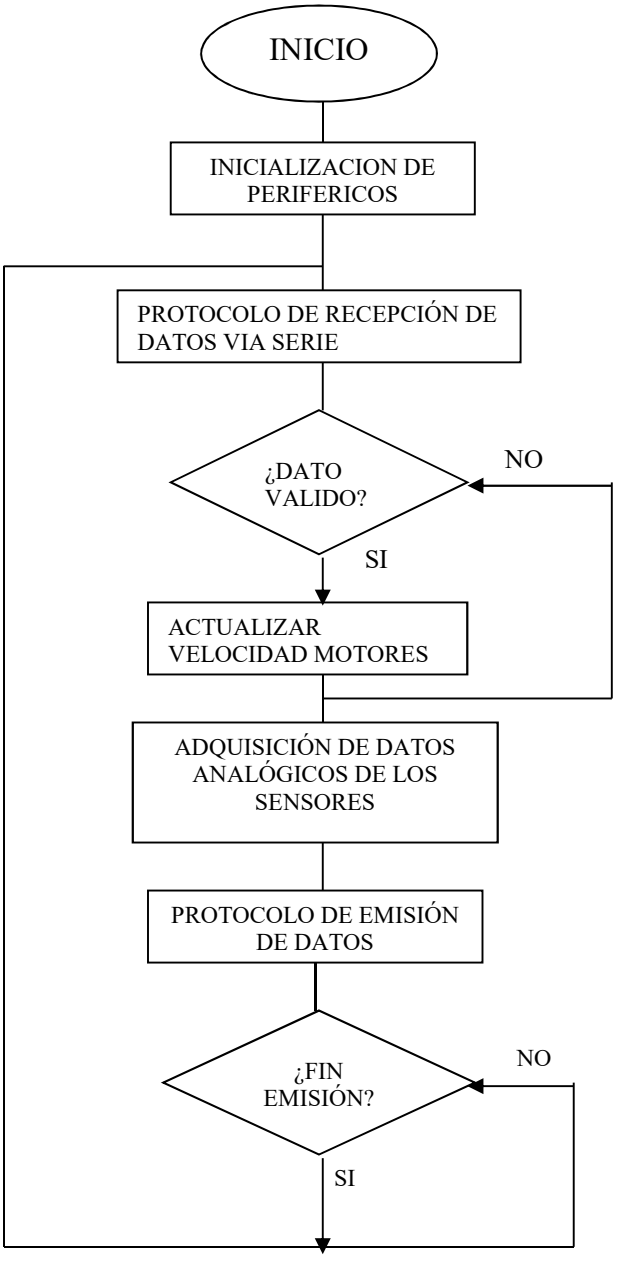


A continuación se exponen el diagrama de flujo general del programa tanto del mando como del vehículo.

- **MANDO**



- **VEHÍCULO**


INICIO
INICIALIZACION DE PERIFERICOS
PROTOCOLO DE RECEPCIÓN DE DATOS VIA SERIE
¿DATO VALIDO?
NO
SI
ACTUALIZAR VELOCIDAD MOTORES
ADQUISICIÓN DE DATOS ANALÓGICOS DE LOS SENSORES
PROTOCOLO DE EMISIÓN DE DATOS
¿FIN EMISIÓN?
NO
SI

En ambos casos la inicialización es la encargada de configurar los distintos recursos del microcontrolador para que trabajen de la forma deseada.

Todos los procesos de inicialización de los periféricos irán marcados por la configuración de los registros de trabajo de cada periférico.

2. CALCULOS DEL HARDWARE

2.1. MODULOS DEL VEHICULO

2.1.1. MICROCONTROLADOR ATMEL T89C51AC2

El microcontrolador será el "cerebro" de nuestro proyecto. En el reside todo el programa y el ejecutara todas las instrucciones. Aparte de los puertos de entrada y salida para el vehículo usaremos los siguientes periféricos:

- **Memoria de programa flash**

Todo el código de programa se almacenara en la memoria de programa flash. Para ello el microcontrolador debe tener los pines EA a 1 lógico, PSEN a 0 lógico y ALE debe estar al aire. Con esto configuramos el micro dejando lista la memoria bootloader para programarlo mediante la UART (RxD y TxD para la comunicación serie). Para ello usamos el programa Flip 1.8.8 proporcionado por el propio fabricante.

- **Puerto serie**

La comunicación serie que vamos a utiliza es comunicación asíncrona a través de la UART. Las funciones para transmisión y recepción pueden trabajar consultando el estado de los flags RI y TI (nuestro caso), o por interrupciones. El registro de control del puerto serie S0CON contiene 8 bits dos de los cuales son precisamente RI y TI. Durante las comunicaciones RI se pone a 1 cuando se ha recibido un carácter y TI se pone a 1 cuando se ha transmitido. El puerto serie será utilizado tanto para la programación de la memoria flash, para lo cual no hacen falta los flags, como para la comunicación mediante infrarrojos y radiofrecuencia.

- **Conversor analógico digital**

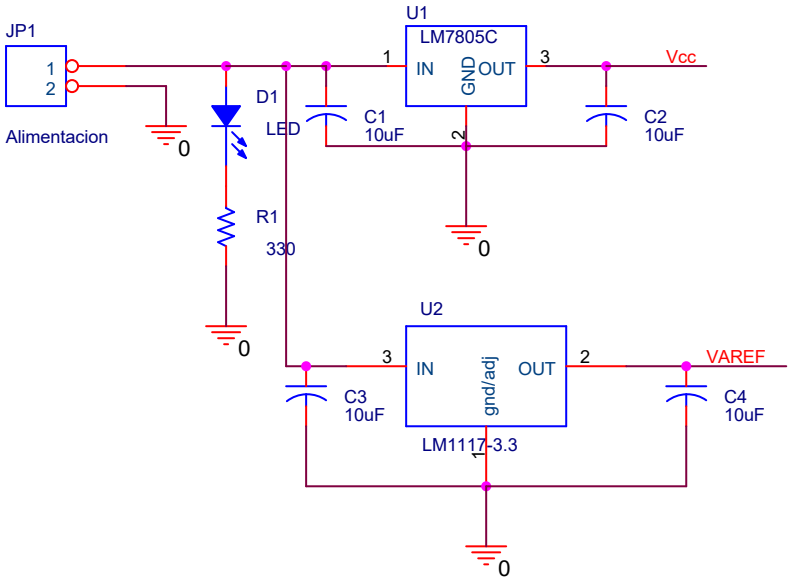
El conversor analógico digital (ADC) será el encargado de adquirir las señales procedentes de los sensores analógicos. Se trata de un ADC de aproximaciones sucesivas que incluye un sample and hold. Este dispositivo utiliza una tensión de referencia entre VAref y Vagnd que no debe ser superior a 3.3V. Se trata de un ADC de 10 bits de resolución aunque en nuestro caso solo usaremos una resolución de 8 bits ya que esos 8 bits serán los que transmitamos por la UART.

- **Timer 0 y Timer 1**

Estos dispositivos los utilizaremos para generar dos ondas cuadradas moduladas en frecuencia con una amplitud de 5V utilizadas como señal CLK para el L297 previamente dividida con el HC4040. Los usaremos en modo 2 (8 bits con autorrecarga) ya que los 8 bits nos los darán los potenciómetros del mando. Usando las interrupciones que generan estos timers cuando finalizan la cuenta cambiaremos de nivel lógico las salidas de los pines P4.0 y P4.1 obteniendo en ellas nuestras señales cuadradas moduladas en frecuencia.

2.1.2. MODULO DE ALIMENTACIÓN DE 5 Y 3.3V

ESQUEMA ELECTRICO:



ESPICIFICACIONES Y CALCULOS

Como se puede observar en el esquema eléctrico precisamos de 2 reguladores de tensión, el LM7805 y el LM1117-3.3 siendo necesarios para estabilizar las tensiones de lógica del circuito y de referencia para el ADC del micro respectivamente.

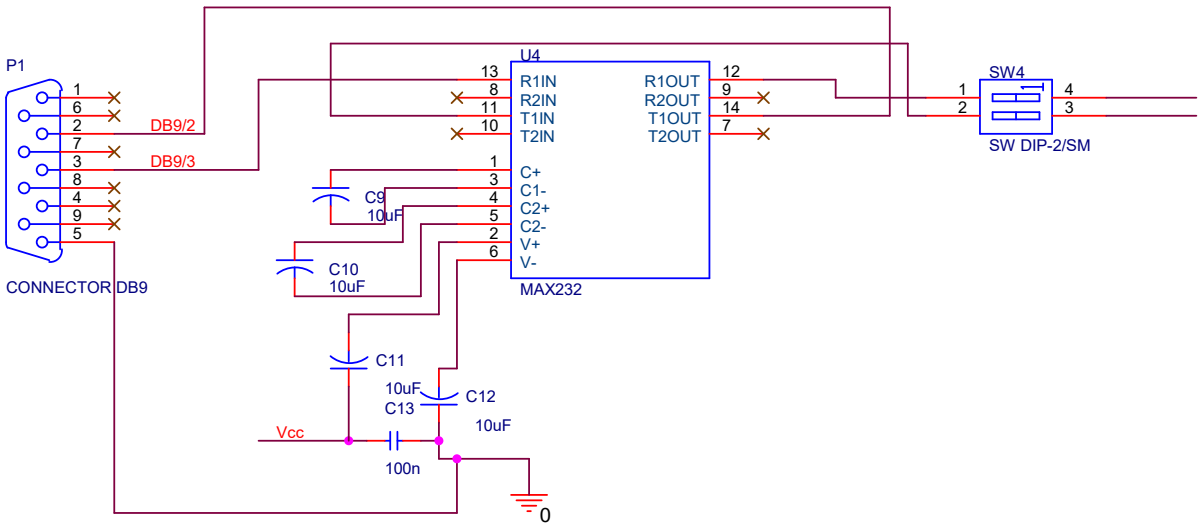
Los condensadores utilizados a la salida y entrada de ambos integrados hacen de filtros para evitar posibles perturbaciones procedentes de la fuente y para disminuir el rizado.

Ya que el microbot será portátil durante la ejecución del programa se precisa una fuente de C.C. de al menos 7V ya que tanto el LM7805 como el LM1117-3.3 necesitan dos voltios más de la tensión a estabilizar para su correcto funcionamiento y además dichos siete voltios alimentaran directamente el módulo de potencia requerido por los motores que más adelante se explicara con detalle.

Los 3.3v que aporta el LM1117 son necesarios para las tensiones de referencia del ADC del microcontrolador dadas por el propio fabricante que nos advierte que la tensión máxima de este periférico no puede superar dicho valor.

2.1.3. MODULO DE PROGRAMACIÓN POR EL PUERTO SERIE

ESQUEMA ELECTRICO



ESPECIFICACIONES Y CALCULOS:

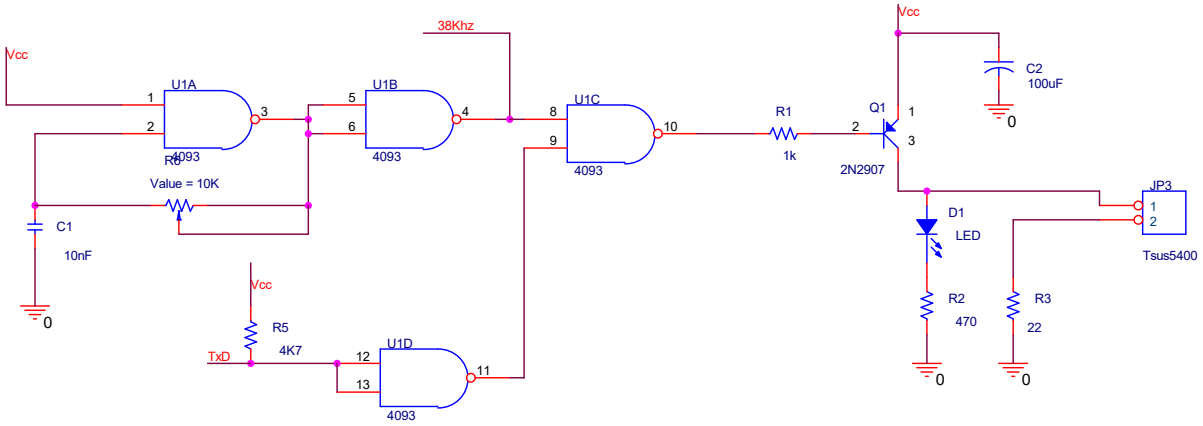
La programación del microcontrolador se realizara usando la UART del propio micro. El fabricante ha reservado un espacio de memoria que con unas condiciones iniciales en determinados pines del micro da la orden de cargar los datos recibidos en la UART en la memoria de programa flash. El encargado de realizar esta función es el pin PSEN que estando a nivel lógico 0 entes de resetear el micro pone a este a la escucha del puerto serie como programador. Para la programación se utiliza el C.I. Max 232 que es un módulo capaz de transformar las señales TTL del micro en señales de comunicaciones RS-232. Este circuito lleva unos condensadores necesarios a tal efecto.

El entorno de programación utilizado es el FLIP1.8.8 dado por ATMEL y que nos permite la comunicación con el micro mediante la bootloader. El entorno de desarrollo para los programas es el keil uVision que nos permite realizar el programa en lenguaje C y tras compilarlo nos genera un archivo .HEX que será el que utilicemos en el entorno de programación.

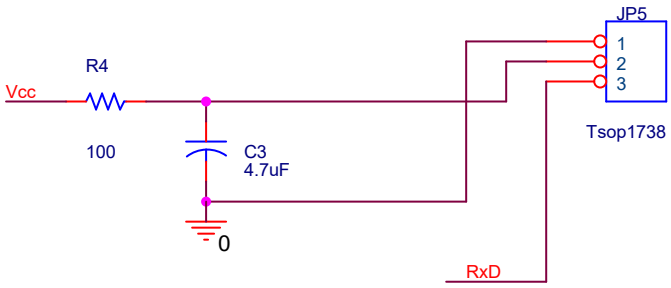
2.1.4. MODULO DE INFRARROJOS

ESQUEMA ELECTRICO:

- **Emisor**



- **Receptor**



ESPECIFICACIONES Y CALCULOS:

Para la comunicación inalámbrica mediante infrarrojos se va a utilizar en diodo emisor TSUS 5400 y el modulo receptor de infrarrojos TSOP 1738. Ambos dispositivos trabajan a 950 nm y han sido seleccionados debido a su amplia característica de directividad y a su gran potencia de emisión y recepción respectivamente como se muestra en las hojas de características de cada uno de ellos.

- **Emisor**

El circuito emisor de infrarrojos consta de un oscilador astable, formado por U1A, U1B, C1, y un potenciómetro de 10K para ajustar la frecuencia de salida en 38Khz aproximadamente, para que coincida con la del receptor demodulador TSOP1738. La puerta NAND aplica esta frecuencia de salida al diodo emisor de infrarrojos TSUS5400 cuando en la entrada de la puerta U1D haya un 0 lógico. El diodo led permanece apagado cuando no está emitiendo.

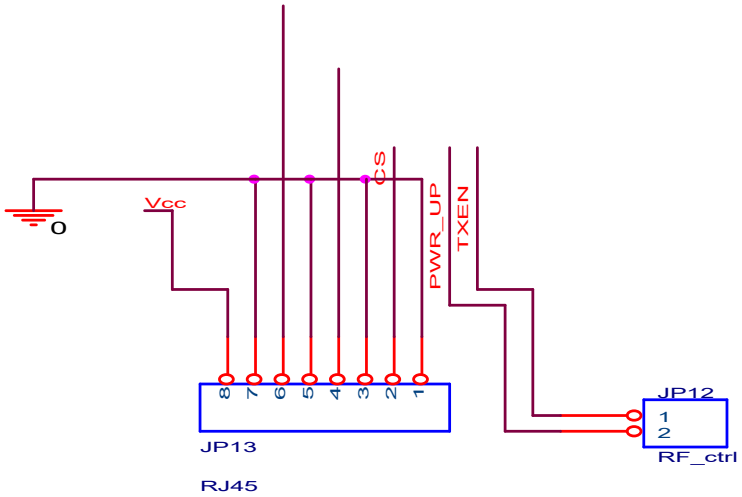
- **Receptor**

El dispositivo TSOP 1738 contiene un diodo PIN detector de infrarrojos, filtro pasa banda que le proporciona inmunidad contra la luz ambiente y un demodulador.

Dispone de una salida compatible TTL y CMOS que pasa a nivel bajo cuando detecta una radiación infrarroja de 950 nm modulada a 38 Khz. Se presenta en un encapsulado de 3 terminales. Se añade un condensador de 4.7uF entre VCC y GND para acoplo y desacoplo.

2.1.5. MODULO DE RADIOFRECUENCIA

ESQUEMA ELECTRICO:



ESPECIFICACIONES Y CALCULOS:

Para la comunicación inalámbrica mediante radiofrecuencia se ha elegido el transceptor nRF401 de NVLSI ASA. Este es un chip UHF transceptor diseñado para operar a una frecuencia de433 Mhz ISM (Industrial, Scientific and Medical). Puede operar a una velocidad de hasta 20Kbit/seg. La potencia de transmisión se puede ajustar hasta un máximo de 10 dB. Opera a una tensión de entre 3 a 5V DC.

Se ha seleccionado en concreto el LOOP KIT del nRF401 ya que además de incluir una antena integrada está preparado totalmente para el propósito de este proyecto. (Esquemas)

A continuación se especifican los pines del kit:

Pin	Nombre	Descripción
1	TXEN	Habilitación de transmisión TXEN=1 => modo transmisión TXEN=0 => modo recepción
2	PWR_UP	Power on/off PWR_UP=1 => ON PWR_UP=0 => OFF
3	GND	0V
4	CS	Selección de canal CS=0 => 433.92 MHz Canal1 CS=1 => 434.33 MHZ Canal2
5	GND	0V
6	DIN	Entrada de dato
7	GND	0V
8	DOUT	Salida de dato
9	GND	0V
10	VDD	Alimentación (+3-5V DC)

Este dispositivo requiere una gran medida de aislamiento ante interferencias como todo dispositivo de radiofrecuencia. Para ello el LOOP KIT lleva 4 salidas de masa para crear un cable trenzado con las patillas de DIN, DOUT, VDD y CS. Los otros dos pines pueden ir solos ya que no es necesario que vayan trenzados con GND. Se ha usado unos conectores RJ 45 junto con su correspondiente cable apantallado que mejora la comunicación y evita ruidos indeseados. También se ha creado una plano de masa en las placas para mejorar el aislamiento.

Los pines CS, TXEN y PWR_UP se han llevado a un puerto del microcontrolador para el control de este dispositivo mediante software. DIN se lleva a la TXD (P3.1) del microcontrolador y DOUT a RxD (P3.0) siendo los encargados de recibir y transmitir los datos. Cabe destacar que RxD y TxD del micro también son necesarios para la comunicación por infrarrojos y para la programación por la tanto ha sido necesario la incorporación de varios interruptores para habilitar una u otra función.

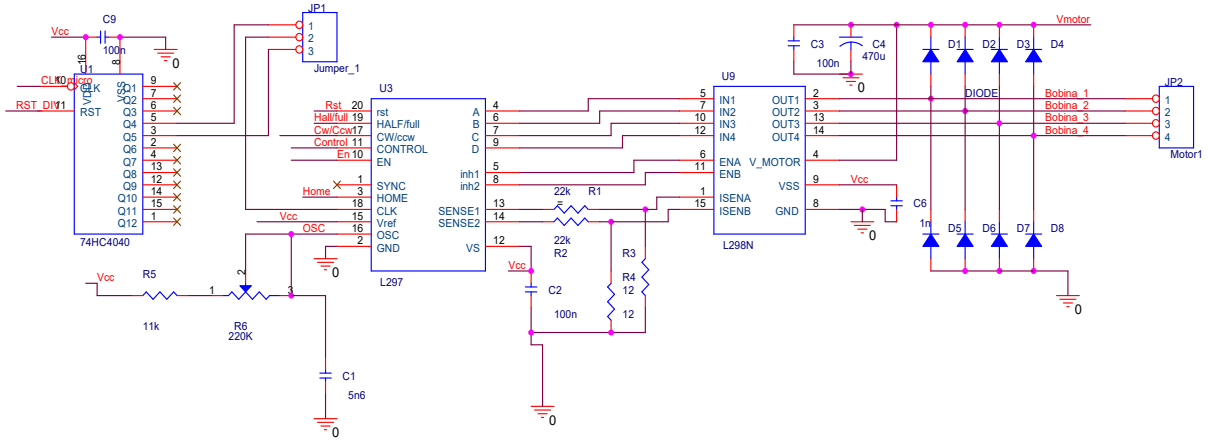
A continuación se presenta la tabla de tiempos necesarios para pasar de modo emisor a modo receptor y viceversa.

Cambio de modo	Nombre	Tiempo mínimo	Condición
TX => RX	t_{TR}	3ms	Modo operacional
RX=>TX	t_{RT}	1ms	
Stdby => RX	t_{ST}	2ms	
Stdby => RX	t_{SR}	3ms	
VDD=0 => TX	t_{VT}	4ms	Inicio
VDD=0 => RX	t_{VR}	5ms	

2.1.6. MODULO DE POTENCIA

ESQUEMA ELECTRICO

A continuación se presentara la configuración del montaje para uno de los motores, siendo igual el otro:



ESPECIFICACIONES Y CALCULOS:

Para el control de los motores se ha seleccionado el C.I. L297 que es un controlador de motores paso a paso. Este junto con el puente en H L298N forma el circuito de control y alimentación de los motores.

La señal de control de velocidad que sale de nuestro microcontrolador viene dada por los temporizadores 1 y 2 en modo de 8 bits con autorrecarga. Esto se debe a que el L297 necesita una entrada de reloj en su pin 18 (CLK). Variando la frecuencia de la señal de reloj que recibe dicho pin conseguiremos variar la velocidad del motor debido a que el L297 utiliza el flanco descendente de esta señal para cambiar el valor de las salidas A, B, C y D que son las encargadas de excitar las bobinas del motor previa amplificación mediante el L298N. Cada flanco descendente hace que el motor de un paso con lo cual nos es muy sencillo un control de velocidad con tan solo usar los temporizadores, incluidos en el propio microcontrolador, y sus interrupciones.

El único inconveniente que se planteo es que la salida de dichos temporizadores era demasiado alta para nuestros motores por lo que previamente se instaló el contador binario de 12 bits HC4040, usado como divisor de frecuencia, con lo que obtenemos una señal apta para nuestros motores.

Desde el microcontrolador se controlan todas las patillas del L297 que hacen de control de motores, así mediante la entrada CW/CCW controlamos el sentido de giro del motor, mediante la Half/Full controlamos que el paso sea completo o medio, con la enable controlamos la habilitación del dispositivo y la salida Home nos da una señal lógica a 1 cuando el motor se encuentra en el paso inicial.

El L298N es un puente en H que proporciona una salida máxima de 2 A. Se le han colocado a la salida unos diodos ultrarrápidos 1N4007 para la protección de las bobinas del motor.

También es de destacar que el uso conjunto de estos dos dispositivos nos da una posibilidad extra que es el control de chopeado. Es decir, mediante los pines 1 y 15 del L298 y junto con las resistencias R7 y R8 obtenemos una corriente de carga, que es detectada por el L297 en SENSE 1 para las bobinas A y B, y en SENSE 2 para las bobinas C y D. Mediante esta corriente conseguimos que el L297 deshabilite el L298N si la corriente sobrepasa la calculada. Para este diseño y para nuestros motores hemos calculado una corriente máxima de 1 A aproximadamente. Según esto y teniendo como tensión de referencia (pin 15) en el L297 5V:

$$V_{ref} = R * I_{max}$$

$$R = 5V / 1\ A = 5\Omega$$

Como aproximación cogemos una resistencia de 5.6K que nos limitara un poco más la corriente. Mediante el pin CONTROL del L297 decidiremos si el chopeado se hace en la salida hacia las bobinas o en las salidas inhibit que nos inhabilitaran el L298N.

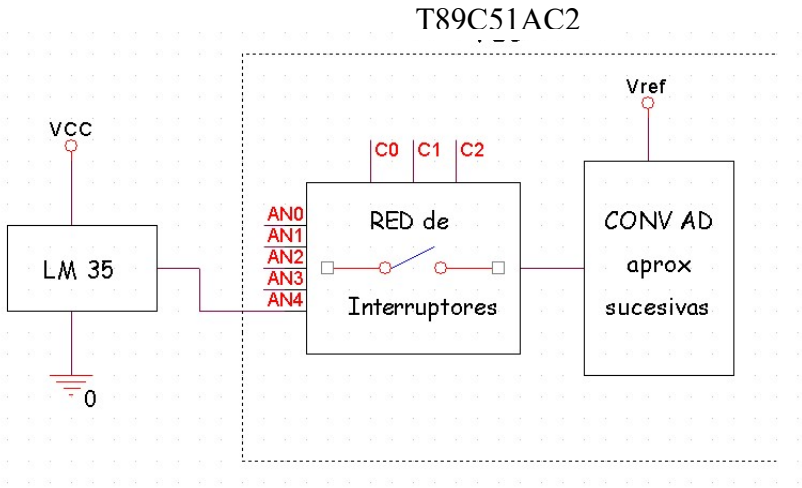
Para el correcto funcionamiento del chopeado es necesario el uso de un oscilador formado por R5, C1 y el pot R6. Con el obtendremos una señal de unos 20Khz. Dicha frecuencia se obtiene por la relación matemática:

$$Fosc = 1/ 0.69\ RC$$

La alimentación Vmotor usada para los motores es la propia de la batería de 7 voltios.

2.1.7. MODULO DE SENSORES

ESQUEMA ELECTRICO:



ESPECIFICACIONES Y CALCULOS:

El conversor ADC del microcontrolador admite entradas analógicas de 0 a 3V, por ello todos los sensores montados deben dar su variación resistiva entre estos valores. Elegiremos preferentemente sensores resistivos como NTCs, LDRs o PTCs. Gracias a esto tenemos un gran abanico de sensores para tomar un gran número de medidas físicas como temperatura, luz humedad o distancia.

Como ejemplo para esta aplicación se ha colocado un sensor de temperatura LM35. El LM 35 es el sensor de temperatura escogido por su bajo coste y buenas prestaciones para nuestra aplicación, permitiéndonos alcanzar una precisión de un decimal, gracias a su sensibilidad de 10mv/ ºC y rango de funcionamiento entre 0 y100 ºC.

2.2. MODULOS DEL MANDO

2.2.1. MICROCONTROLADOR ATMEL T89C51AC2

El microcontrolador será el "cerebro" de nuestro proyecto. En el reside todo el programa y el ejecutara todas las instrucciones. Aparte de los puertos de entrada y salida para el vehículo usaremos los siguientes periféricos:

- **Memoria de programa flash**

Todo el código de programa se almacenara en la memoria de programa flash. Para ello el microcontrolador debe tener los pines EA a 1 lógico, PSEN a 0 lógico y ALE debe estar al aire. Con esto configuramos el micro dejando lista la memoria bootloader para programarlo mediante la UART (RxD y TxD para la comunicación serie). Para ello usamos el programa Flip 1.8.8 proporcionado por el propio fabricante.

- **Puerto serie**

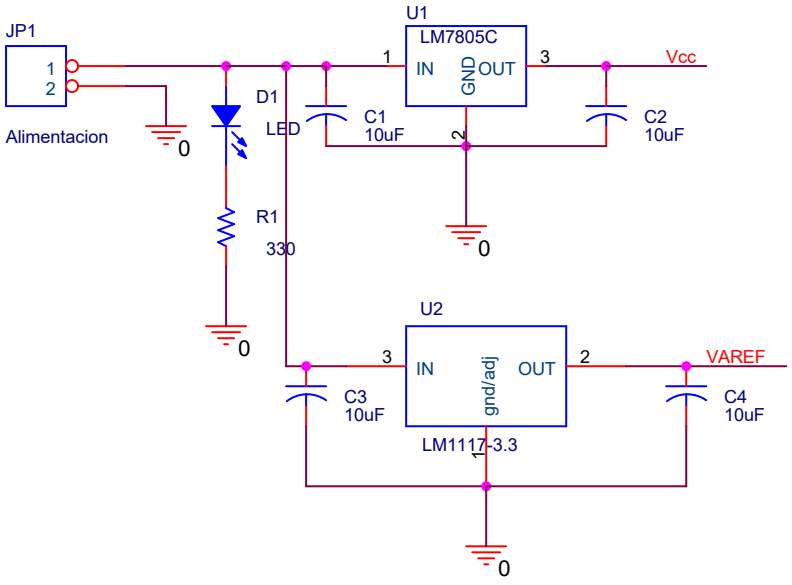
La comunicación serie que se utiliza es comunicación asíncrona a través de la UART. Las funciones para transmisión y recepción pueden trabajar consultando el estado de los flags RI y TI (nuestro caso), o por interrupciones. El registro de control del puerto serie S0CON contiene 8 bits dos de los cuales son precisamente RI y TI. Durante las comunicaciones RI se pone a 1 cuando se ha recibido un carácter y TI se pone a 1 cuando se ha transmitido. El puerto serie será utilizado tanto para la programación de la memoria flash, para lo cual no hacen falta los flags, como para la comunicación mediante infrarrojos y radiofrecuencia.

- **Conversor analógico digital**

El conversor analógico digital (ADC) será el encargado de adquirir las señales procedentes de los potenciómetros. Se trata de un ADC de aproximaciones sucesivas que incluye un sample and hold. Este dispositivo utiliza una tensión de referencia entre VAref y Vagnd que no debe ser superior a 3.3V. Se trata de un ADC de 10 bits de resolución aunque en nuestro caso solo usaremos una resolución de 8 bits ya que esos 8 bits serán los que transmitamos por la UART y que el vehículo recibirá para controlar la velocidad de los motores.

2.2.2. MODULO DE ALIMENTACIÓN DE 5 Y 3.3V

ESQUEMA ELECTRICO:



ESPICIFICACIONES Y CALCULOS

Como se puede observar en el esquema eléctrico precisamos de 2 reguladores de tensión, el LM7805 y el LM1117-3.3 siendo necesarios para estabilizar las tensiones de lógica del circuito y de referencia para el ADC del micro respectivamente.

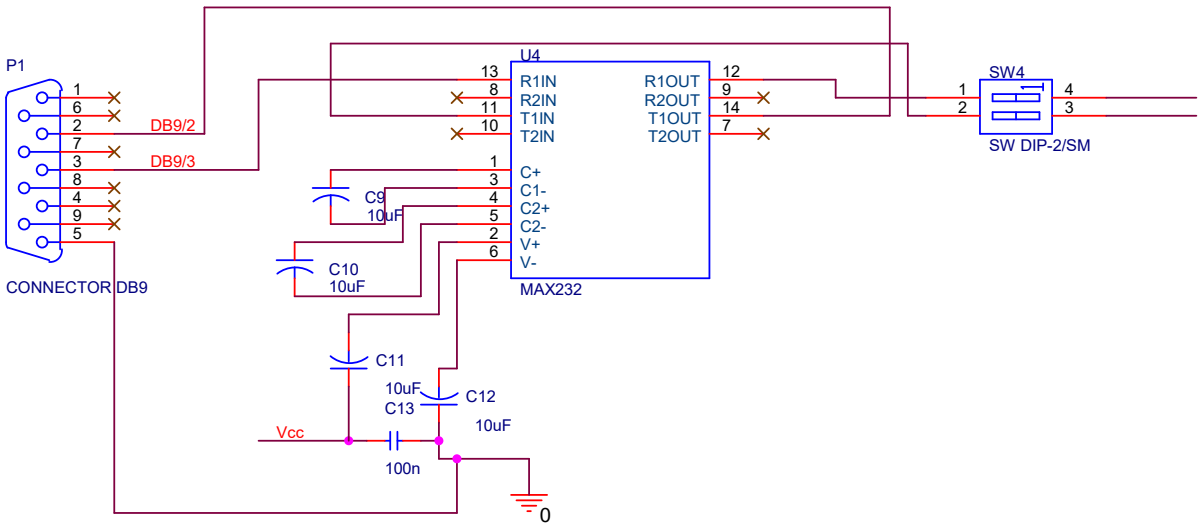
Los condensadores utilizados a la salida y entrada de ambos integrados hacen de filtros para evitar posibles perturbaciones procedentes de la fuente y para disminuir el rizado.

Ya que el microbot será portátil durante la ejecución del programa se precisa una fuente de C.C. de al menos 7V ya que tanto el LM7805 como el LM1117-3.3 necesitan dos voltios más de la tensión a estabilizar para su correcto funcionamiento y además dichos siete voltios alimentaran directamente el módulo de potencia requerido por los motores que más adelante se explicara con detalle.

Los 3.3v que nos aporta el LM1117 son necesarios para las tensiones de referencia del ADC del microcontrolador dadas por el propio fabricante que nos advierte que la tensión máxima de este periférico no puede superar dicho valor.

2.2.3. MODULO DE PROGRAMACIÓN POR EL PUERTO SERIE

ESQUEMA ELECTRICO



ESPECIFICACIONES Y CALCULOS:

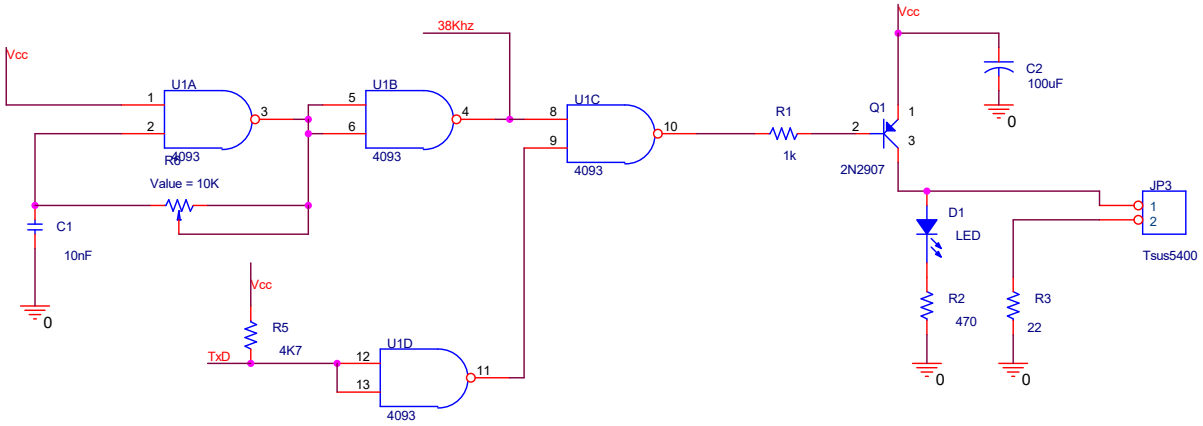
La programación del microcontrolador se realizara usando la UART del propio micro. El fabricante ha reservado un espacio de memoria que con unas condiciones iniciales en determinados pines del micro da la orden de cargar los datos recibidos en la UART en la memoria de programa flash. El encargado de realizar esta función es el pin PSEN que estando a nivel lógico 0 entes de resetear el micro pone a este a la escucha del puerto serie como programador. Para la programación se utiliza el C.I. Max 232 que es un módulo capaz de transformar las señales TTL del micro en señales de comunicaciones RS-232. Este circuito lleva unos condensadores necesarios a tal efecto.

El entorno de programación será el FLIP1.8.8 dado por ATMEL y que nos permite la comunicación con el micro mediante la bootloader. El entorno de desarrollo para los programas es el keil uVision que nos permite realizar el programa en lenguaje C y tras compilarlo nos genera un archivo .HEX que será el que utilicemos en el entorno de programación.

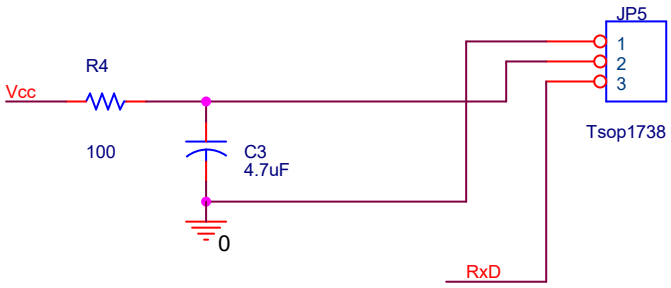
2.2.4. MODULO DE INFRARROJOS

ESQUEMA ELECTRICO:

- **Emisor**



- **Receptor**



ESPECIFICACIONES Y CALCULOS:

Para la comunicación inalámbrica mediante infrarrojos se va a utilizar en diodo emisor TSUS 5400 y el modulo receptor de infrarrojos TSOP 1738. Ambos dispositivos trabajan a 950 nm y han sido seleccionados debido a su amplia característica de directividad y a su gran potencia de emisión y recepción respectivamente como se muestra en las hojas de características de cada uno de ellos.

- **Emisor**

El circuito emisor de infrarrojos consta de un oscilador astable, formado por U1A, U1B, C1, y un potenciómetro de 10K para ajustar la frecuencia de salida en 38Khz aproximadamente, para que coincida con la del receptor demodulador TSOP1738. La puerta NAND aplica esta frecuencia de salida al diodo emisor de infrarrojos TSUS5400 cuando en la entrada de la puerta U1D haya un 0 lógico. El diodo led permanece apagado cuando no está emitiendo.

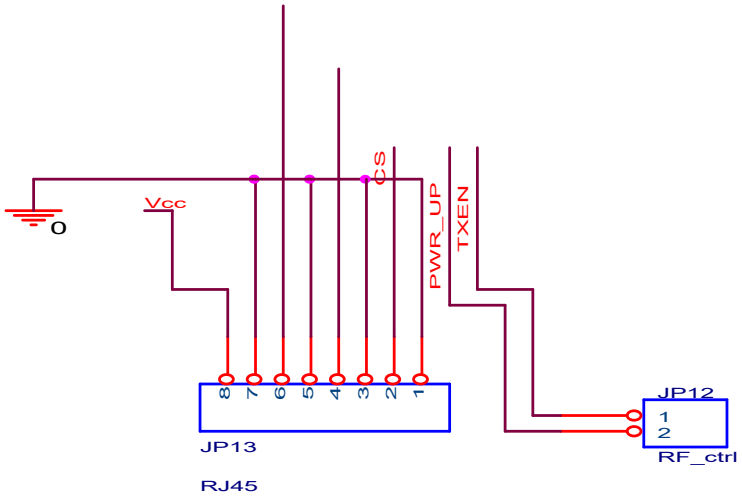
- **Receptor**

El dispositivo TSOP 1738 contiene un diodo PIN detector de infrarrojos, filtro pasa banda que le proporciona inmunidad contra la luz ambiente y un demodulador.

Dispone de una salida compatible TTL y CMOS que pasa a nivel bajo cuando detecta una radiación infrarroja de 950 nm modulada a 38 Khz. Se presenta en un encapsulado de 3 terminales. Se añade un condensador de 4.7uF entre VCC y GND para acoplo y desacoplo.

2.2.5. MODULO DE RADIOFRECUENCIA

ESQUEMA ELECTRICO:



ESPECIFICACIONES Y CALCULOS:

Para la comunicación inalámbrica mediante radiofrecuencia se ha elegido el transceptor nRF401 de NVLSI ASA. Este es un chip UHF transceptor diseñado para operar a una frecuencia de433 Mhz ISM (Industrial, Scientific and Medical). Puede operar a una velocidad de hasta 20Kbit/seg. La potencia de transmisión se puede ajustar hasta un máximo de 10 dB. Opera a una tensión de entre 3 a 5V DC.

Se ha seleccionado en concreto el LOOP KIT del nRF401 ya que además de incluir una antena integrada está preparado totalmente para el propósito de este proyecto. (Esquemas)

A continuación se especifican los pines del kit:

Pin	Nombre	Descripción
1	TXEN	Habilitación de transmisión TXEN=1 => modo transmisión TXEN=0 => modo recepción
2	PWR_UP	Power on/off PWR_UP=1 => ON PWR_UP=0 => OFF
3	GND	0V
4	CS	Selección de canal CS=0 => 433.92 MHz Canal1 CS=1 => 434.33 MHZ Canal2
5	GND	0V
6	DIN	Entrada de dato
7	GND	0V
8	DOUT	Salida de dato
9	GND	0V
10	VDD	Alimentación (+3-5V DC)

Este dispositivo requiere una gran medida de aislamiento ante interferencias como todo dispositivo de radiofrecuencia. Para ello el LOOP KIT lleva 4 salidas de masa para crear un cable trenzado con las patillas de DIN, DOUT, VDD y CS. Los otros dos pines pueden ir solos ya que no es necesario que vayan trenzados con GND. Se ha usado unos conectores RJ 45 junto con su correspondiente cable apantallado que mejora la comunicación y evita ruidos indeseados. También se ha creado una plano de masa en las placas para mejorar el aislamiento.

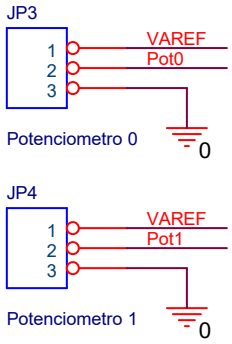
Los pines CS, TXEN y PWR_UP se han llevado a un puerto del microcontrolador para el control de este dispositivo mediante software. DIN se lleva a la TXD (P3.1) del microcontrolador y DOUT a RxD (P3.0) q serán los encargados de recibir y transmitir los datos. Cabe destacar que RxD y TxD del micro también serán necesarios para la comunicación por infrarrojos y para la programación por la tanto ha sido necesario la incorporación de varios interruptores para habilitar una u otra función.

A continuación se presenta la tabla de tiempos necesarios para pasar de modo emisor a modo receptor y viceversa.

Cambio de modo	Nombre	Tiempo mínimo	Condición
TX => RX	t_{TR}	3ms	Modo operacional
RX=>TX	t_{RT}	1ms	
Stdby => RX	t_{ST}	2ms	
Stdby => RX	t_{SR}	3ms	
VDD=0 => TX	t_{VT}	4ms	Inicio
VDD=0 => RX	t_{VR}	5ms	

2.2.6. MODULO DE CONTROL

ESQUEMA ELECTRICO:



ESPECIFICACIONES Y CALCULOS:

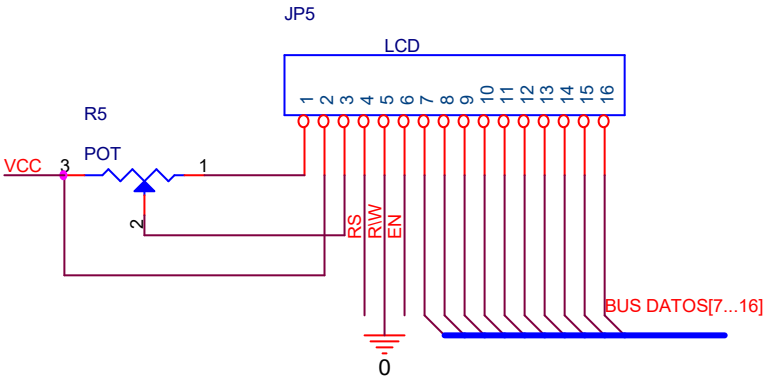
Para controlar los motores se ha decidido capturar una señal analógica que nosotros mismos variaremos y enviaremos posteriormente al vehículo para que este regule la velocidad de los motores. Para ello usamos 2 potenciómetros. Debido a que el ADC solo admite tensiones de 0 a 3V alimentaremos los potenciómetros con este rango de valores. Tomaremos las mediciones del terminal variable del potenciómetro con lo que tendremos un control analógico para la velocidad. Las señales Po0 y pot1 del esquema son las que irán al canal analógico del microcontrolador.

Se ha dividido el potenciómetro en dos tramos: superior e inferior. Estando el potenciómetro en posición centrada el motor correspondiente a dicho potenciómetro permanecerá parado. Si se mueve el potenciómetro hacia el tramo superior la velocidad del motor ira aumentado y girara en un sentido. Si se mueve el potenciómetro hacia el otro lado la velocidad del motor variara y el este girara en sentido contrario.

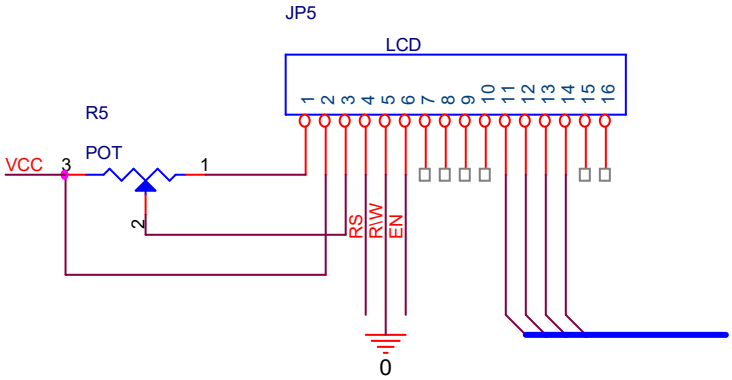
2.2.7. MODULO PARA EL MANEJO DE LA PANTALLA LCD

ESQUEMA ELECTRICO:

- **Modo de 8 Bits**



- **Modo de 4 bits**



ESPECIFICACIONES Y CALCULOS:

Introducción

Como Interfaz de usuario hemos optado por el Modulo LCD de la casa Wintek, modelo: WM-C1602M.

En él a través de 8 líneas de datos se le envía el carácter ASCII que se desea visualizar así como ciertos códigos de control que permiten realizar diferentes efectos de visualización. Igualmente mediante estas líneas de datos el módulo devuelve información de su estado interno. Con otras tres señales adicionales se controla el flujo de información entre el módulo LCD y el microcontrolador.

A continuación se presenta la descripción de señales empleadas por el módulo LCD, así como el número de patilla a la que corresponden.

Pin N-.	Sismología	Nivel	I/O	Función
1	VSS	-	-	0 V. Tierra (GND).
2	VCC	-	-	+ 5 V. VDC.
3	VO	-	-	Ajuste del Contraste.
4	RS	0/1	I	0= Escribir en el modulo LCD. 1= Leer del módulo LCD
5	R/W	0/1	I	0= Entrada de una Instrucción. 1= Entrada de un dato.
6	E	1	I	Habilitación del módulo LCD
7	DB0	0/1	I/O	BUS DE DATO LINEA 1 (LSB).
8	DB1	0/1	I/O	BUS DE DATO LINEA 2
9	DB2	0/1	I/O	BUS DE DATO LINEA 3
10	DB3	0/1	I/O	BUS DE DATO LINEA 4
11	DB4	0/1	I/O	BUS DE DATO LINEA 5
12	DB5	0/1	I/O	BUS DE DATO LINEA 6
13	DB6	0/1	I/O	BUS DE DATO LINEA 7
14	DB7	0/1	I/O	BUS DE DATO LINEA 8 (MSB).
15	NC	-	-	No se conecta
16	NC	-	-	No se conecta

Interpretación del significado de los Pines del Modulo LCD

- El **Pin nº 1 y 2** están destinados para conectarle los 5 Voltios que requiere el modulo para su funcionamiento.

- El **Pin nº 3** es utilizado para ajustar el contraste de la pantalla; es decir colocar los caracteres más oscuros o más claros para poderse observar mejor. A este pin se le suele colocar un potenciómetro variable que puede oscilar entre 10 K y 20 K indiferentemente (mirar interface con el LCD).

- El **Pin nº 4** denominado "RS" trabaja paralelamente con el Bus de datos del módulo LCD (Bus de datos son los Pines del 7 al 14). Este bus es utilizado de dos maneras, ya que podemos colocar un dato que representa una instrucción o colocar un dato que tan solo representa un símbolo o un carácter alfanumérico, pero para que el modulo LCD pueda entender la diferencia entre un dato o una instrucción se utiliza el Pin Numero 4.
  Si el Pin nº4 = 0 le dirá al módulo LCD que está presente en el bus de datos una instrucción, por el contrario, si el Pin nº4 = 1 le dirá al módulo LCD que está presente un símbolo o un carácter.

- El **Pin nº 5** denominado "R/W" trabaja paralelamente con el Bus de datos del módulo. También es utilizado de dos maneras, ya que podemos decirle al módulo LCD que escriba en pantalla el dato que está presente en el Bus o por otro lado leer que dato está presente en el Bus.
  Si el Pin número 5 = 0 el modulo LCD escribe en pantalla el dato que está presente el Bus, pero si el Pin número 5 = 1 significa que queremos leer el dato que está presente en el bus.

- El **Pin nº 6** denominado "E" que significa habilitación del módulo LCD tiene una finalidad básica: conectar y desconectar el modulo. Esta desconexión no estará referida al voltaje que le suministra la corriente al módulo; la desconexión significa

tan solo que se hará caso omiso a todo lo que esté presente en el bus de datos del módulo.

- Los **Pines desde el nº7 hasta el nº14** representan 8 líneas que se utilizan para colocar el dato que representa una instrucción para el modulo LCD o un carácter alfanumérico. El Bus de datos es de 8 Bits de longitud y el Bit menos significativo está representado en el Pin nº7, el Pin más significativo está representado en el Pin nº14

- Los **Pines 15 y 16** estarán destinados para suministrar la corriente al Back Light, pero en este módulo no ha sido implementada dicha luz de fondo debido a su reducido coste

Secuencia de Inicialización del módulo LCD

El módulo LCD ejecuta automáticamente una secuencia de inicio interna en el instante de aplicarle la tensión de alimentación si se cumplen los requisitos de alimentación. Dichos requisitos consisten en que el tiempo que tarde en estabilizarse la tensión desde 0.2 V hasta los 4.5V mínimos necesarios sea entre 0.1 ms y 10 ms. Igualmente el tiempo de desconexión debe ser como mínimo de 1 ms antes de volver a conectar.

La secuencia de inicio ejecutada es la siguiente:

1. Se ejecuta el comando **"BORRAR PANTALLA"** preparando la pantalla. El flag **BUSY** se mantiene a "1" (ocupado) durante 15 ms hasta que finaliza la inicialización.

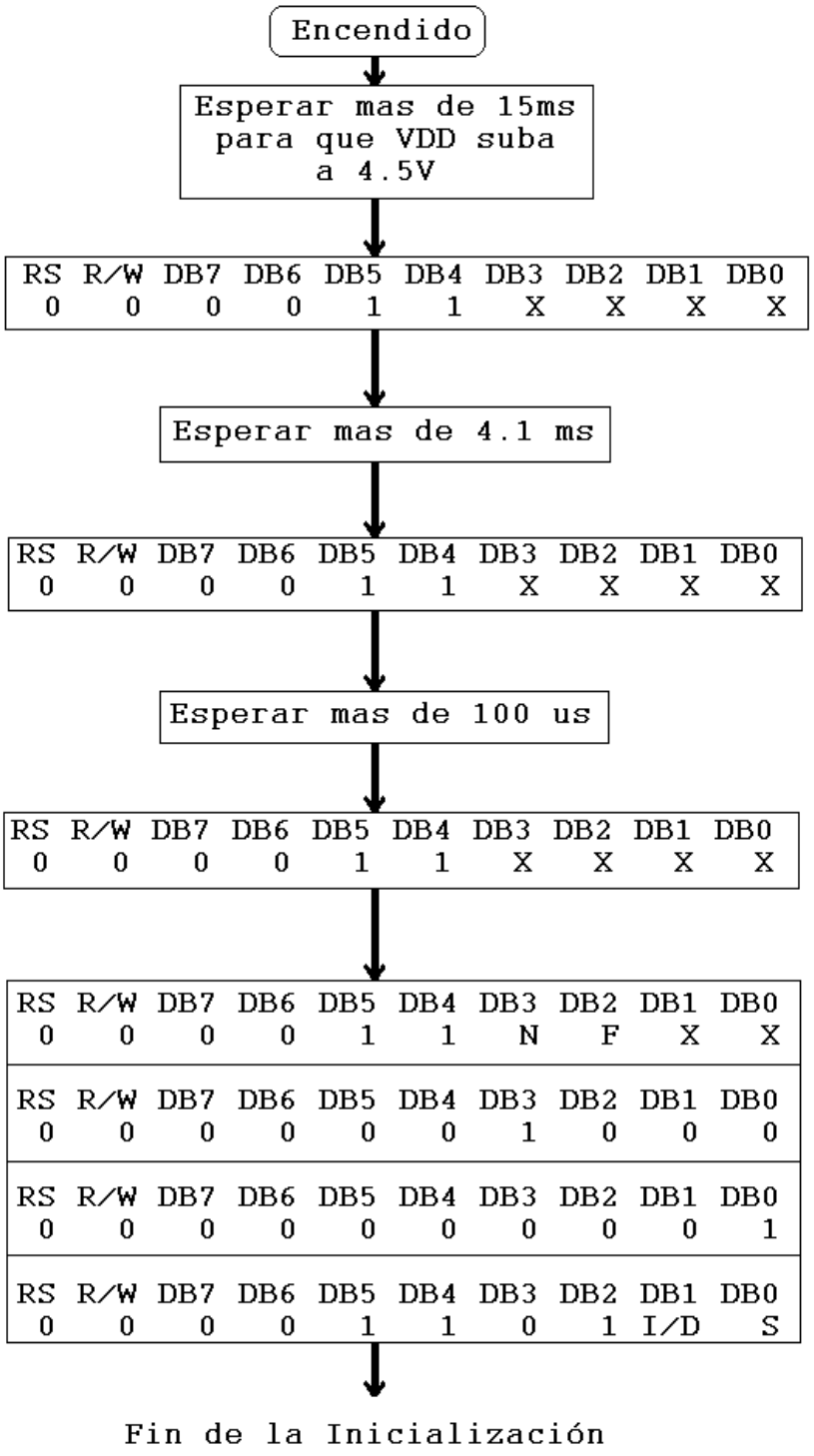
2. Se ejecuta el comando "**ESTABLECER FUNCION"**, que establece el interfaz con el Bus de datos. Se elige por defecto el tamaño del bus de datos a 8 bits (DL=1) y el número de renglones del display en 1 (N=0).

3. Se ejecuta el comando "**CONTROL DEL DISPLAY ON/OFF",** que hace que el display quede en OFF (D=0); también cursor en OFF (C=0) y sin parpadeo del cursor en (B=0).

4. Se ejecuta el comando **"ESTABLECER MODO DE ENTRADA",** que establece la dirección de
Movimiento del cursor con autoincremento del cursor (I/D=1) y modo normal, no desplazamiento, del display (S=0).

Si la conexión de la alimentación no reúne las condiciones que exige el módulo LCD, habría que realizar la secuencia de inicialización por software. En cualquier caso, es importante enviar al LCD la primera instrucción de trabajo después de que hayan transcurrido 15 ms, para completar dicha secuencia de inicialización.

En el diagrama de flujo de la secuencia de Inicialización para bus de 8 bits (nuestro caso) se observa el código detallado:

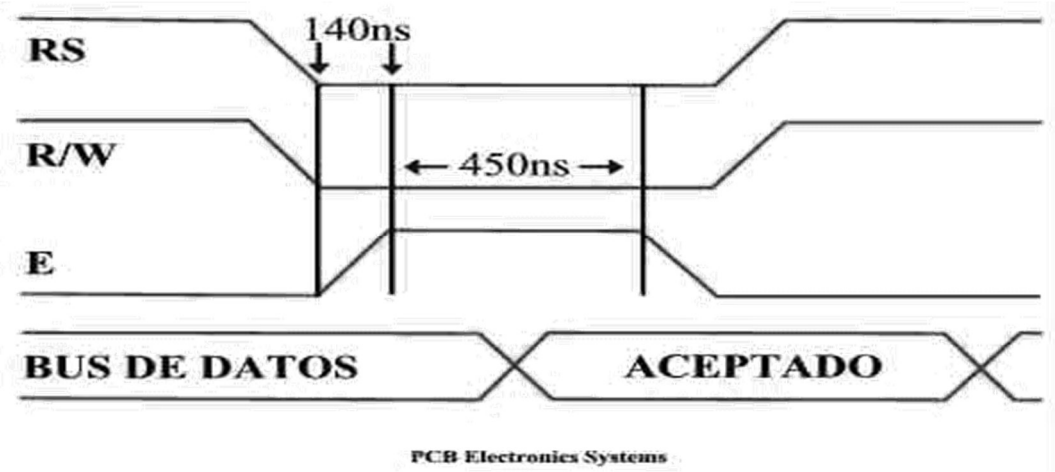


Tiempos mínimos para poder ejecutar una instrucción o dato

Los Pines de control (E, RS y E/W) están estrechamente relacionados ya que por medio de ellos podemos especificar si queremos ejecutar una instrucción o leer / escribir un dato en la pantalla o la memoria RAM; sin embargo existe una condición importante que deberá tomarse en cuenta referida directamente al tiempo necesario que se necesita para cambiar de un estado a otro en los pines de control. (E, RS y R/W). En el caso de que este tiempo sea más pequeño que el tiempo mínimo requerido, entonces el modulo LCD no tendrá el tiempo suficiente para responder a las instrucciones solicitadas por el usuario y por consecuencia se perderán los datos o instrucciones según sea el caso.

Suele ocurrir un error cuando se está intentando hacer funcionar el modulo LCD sin considerar la velocidad de proceso del microcontrolador, concretamente en los pines de control (E, RS y R/W), esto quiere decir que si la velocidad de proceso es demasiado alta en los pines de control, cuando se ejecuta una solicitud de cualquier tipo (escritura / lectura e Instrucción), el modulo LCD no tendrá la capacidad de entender la solicitud hecha por el microcontrolador ya que esta se ejecutó demasiado rápida. Para ello los programas y circuitos deben respetar los siguientes diagramas de tiempo:

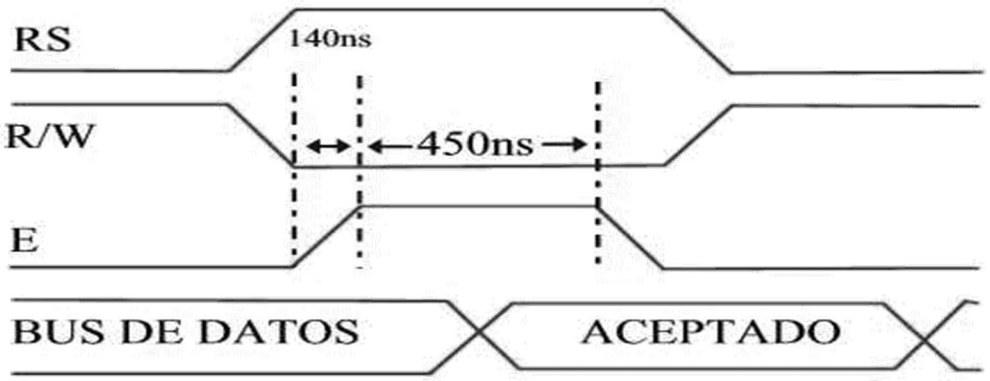
DIAGRAMA DE TIEMPO PARA UNA EJECUTAR UNA INSTRUCCIÓN



Para enviarle una instrucción al módulo, primero hay que colocar la instrucción en el bus de datos (Pines del 7 al 14). Una vez que está presente la instrucción en el bus de datos se procede a ejecutar el diagrama de tiempo requerido para una instrucción en los pines de control.

Este diagrama de tiempo es muy sencillo de entender, tan solo hay que colocar el Pin RS = 0, el Pin R/W = 0 y el Pin E = 0; Una vez colocados los pines con las tensiones mencionadas, se procede a cambiar el estado del Pin E = 1. El nuevo estado de este Pin "E" deberá permanecer por lo menos 450 ns antes de volver a cambiar de estado para que la pantalla pueda entender la instrucción.

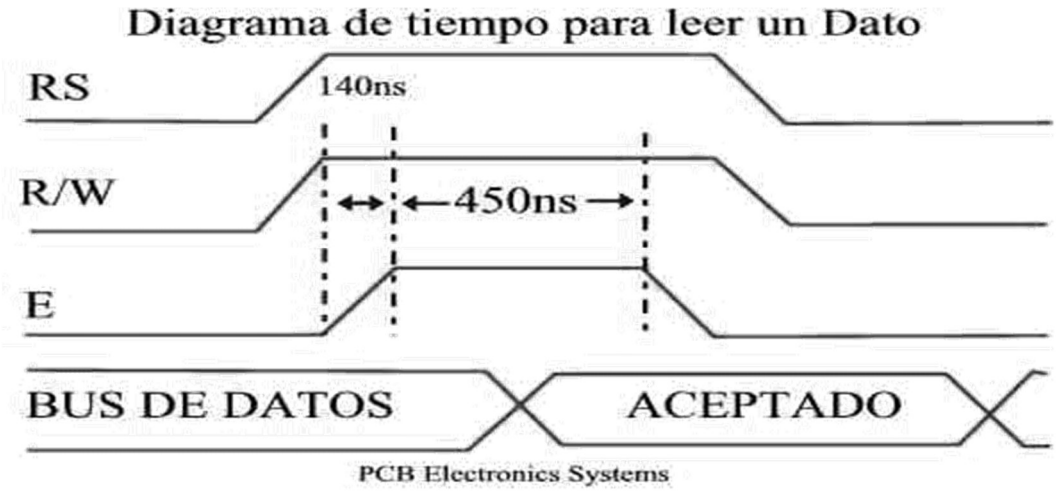
DIAGRAMA DE TIEMPO PARA ESCRIBIR UN DATO



PCB Electronics Systems

Para escribir un dato en el módulo LCD, primero hay que colocar el dato en el bus (Pines del 7 al 14). Una vez que está presente el dato en el bus se procede a ejecutar el diagrama de tiempo requerido para escribir un dato en los pines de control. Como vemos en el diagrama solo hay que poner el Pin RS = 1, el Pin R/W = 0 y el Pin E = 0; Una vez colocados los pines con las tensiones mencionadas, se procede a cambiar el estado del Pin E = 1. El nuevo estado
De este Pin "E" deberá permanecer por lo menos 450 ns antes de volver a cambiar de estado para que la pantalla pueda entender la instrucción.

DIAGRAMA DE TIEMPO PARA LEER UN DATO



Para leer un dato de la pantalla o la memoria RAM en el módulo LCD, los pines de control deberán estar colocados como sigue: Pin RS = 1, Pin R/W = 1 y el Pin E = 0. Una vez colocados los pines con las tensiones mencionadas, hay que proceder a cambiar el estado del Pin E =1. El nuevo estado de este Pin "E" deberá permanecer por lo menos 450 ns antes de volver a cambiar de estado para que la pantalla pueda entender la instrucción.

3. CALCULOS DEL SOFTWARE

3.1. INTRODUCCION AL CODIGO

INTRODUCCION

Los motivos por los cuales hemos desarrollado el código del programa en lenguaje C son los siguientes:

- Los microcontroladores de la familia 8051 son fácilmente programables en lenguaje C. Además disponen de un entorno de diseño fácil de utilizar y muy potente como es el Keil uVision.

- Previamente se hemos desarrollado varios prototipos de trabajos anteriores con esta herramienta de trabajo y debido a la buena experiencia con ella nos ha sido más fácil manejarla

- Aunque el lenguaje en C genera más líneas de código este microcontrolador tiene 32 Kb de memoria de programa con lo que tenemos espacio suficiente. Además el lenguaje en C es un lenguaje mucho más cómodo para programar y más flexible para futuras ampliaciones. También es mucho más sencillo de leer para alguien ajeno al proyecto con conocimientos de C.

En conclusión, se puede decir que si se desea programar eficientemente el robot en lenguaje C la familia de microcontroladores de 8051 da grandes facilidades para la programación en este lenguaje y gracias al uVision tenemos una herramienta eficaz para el desarrollo de los programas y que nos genera el archivo HEX necesario para la programación.

3.2. PARTES DE CODIGO

El programa fuente se puede subdividir en claras partes ya que cada una desarrolla una tarea concreta. A continuación se detallan las explicaciones, diagramas de flujo de las partes más complicadas y partes de código en C que han sido necesarios para la obtención del programa fuente principal. Estas partes irán en forma de funciones con lo que en el programa principal (main) solo será necesario llamar a cada una de ellas.

3.2.1. FUNCIONES NECESARIAS TANTO EN EL MANDO COMO EN EL VEHÍCULO

- **Función Delay**

Esta función introduce un retardo en el programa. Es muy útil en casos de configuración en los que es necesario la espera de unos milisegundos para poder ejecutar la siguiente instrucción como son los casos de la LCD y del transceptor de radiofrecuencia. Funciona introduciendo el número de milisegundos que queremos esperar al llamar a la función.

El programa en C para la función es el siguiente:

```
void delay(int i)
{
        UCHAR uc;
        while(i!=0)
                {
                For(uc=227; uc!=0; uc--);
                For(uc=227; uc!=0; uc--);
                i--;
                }
}
```

- **Inicialización del conversor analógico digital**

Lo primero que se hace es decir cuáles de los ocho canales van a funcionar como canales analógicos. Para ello usamos el registro de 8 bits ADCF el cual, según el valor introducido nos habilitara los canales deseados. Cada uno de los bits de este registro representa a un canal. Si el bit está a uno el canal funciona como analógico. Si el bit está a 0 el canal funciona en cualquier otra de sus funciones, bien sea cono línea de entrada /salida o en cualquier otro modo disponible. En el mando solo se usaran los canales 0 y 1 así pues en el registro ADCF pondremos ADCF=0x03 y en el vehículo usaremos los 8 canales así que lo cargaremos con ADCF = 0xFF.

Lo siguiente será habilitar el ADC y para ello usaremos el registro ADCON. A este registro lo inicializaremos con 0x20 para poner a 1 el bit ADEN que habilita el conversor.

Si se desea trabajar con la interrupción de que dispone este periférico bastara con poner el bit EADC =1. Previamente se deberán habilitar las interrupciones con EA=1.

El programa en C para la función es el siguiente:

```
void adc_init(void)
{
        ADCF=0xFF;              /*configura los canales a utilizar del puerto uno(ADC)*/
        ADCON=0x20;             /*habilita ADC*/
        delay(1);
        EADC=1;                 /*habilita interrupción de adc*/
}
```

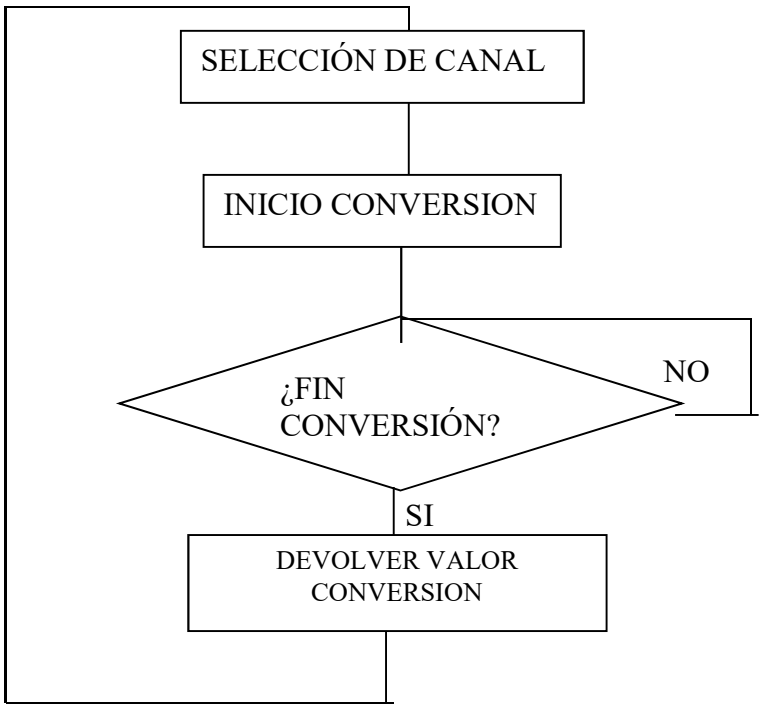
- **Función de adquisición del dato analógico**

Lo primero que se realizara en esta función es la selección del canal mediante los tres últimos bits del registro ADCON. En esta tabla se representa la selección de canales en función de estos bits:

SCH2	SCH1	SCH0	Canal analógico seleccionado
0	0	0	AN2
0	0	1	AN1
0	1	0	AN2
0	1	1	AN3
1	0	0	AN4
1	0	1	AN5
1	1	0	AN6
1	1	1	AN7

Tras la selección del canal se pone a 1 el bit de inicio de conversión (ADSST) que se encuentra en el registro ADCON. Este bit se pone a 0 por hardware al finalizar la conversión. Después de iniciar la conversión se espera a que se ponga a 1 el flag de fin de conversión (ADEOC) también contenido en el registro ADCON.

El funcionamiento de esta función se representara mediante un diagrama de flujo:



A continuacion se muestra el programa en C para esta funcion:

```
UCHAR get_adc(UCHAR CANAL)
 {
        UCHAR value_converted;
        ADCON&=0xF8;              /*mascara para poner a 0 los SCH(canal selection)*/
        ADCON|=CANAL;             /*mascara para elegir el canal*/
        ADCON|=0x08;              /*Se pone pone a 1 el bit de inicio conversion*/
        while((ADCON&0x10)==0);                 /  *espera bit final de conversión/
        value_converted=ADDH;
        ADCON=0x20;       /*Se vuelve a poner todo a cero excepto el de habilitacion*/
        return value_converted;     /*Se devuelve el valor analógico obtenido*/
}
```

- **Función para el inicio de la UART**

Con esta función lo que hacemos es seleccionar la velocidad a la que trabajara la UART. Al llamar a la función se le da el valor que se quiere para la transmisión. Este valor viene definido por la frecuencia del oscilador o cristal que tenemos montado. En este caso se ha montado un cristal de 11.0952 MHz. Esta frecuencia se define al principio del programa como se puede observar en la siguiente línea de código:
#define OSC 11059200L

A continuación se muestra el código en C para configurar la velocidad de la UART:

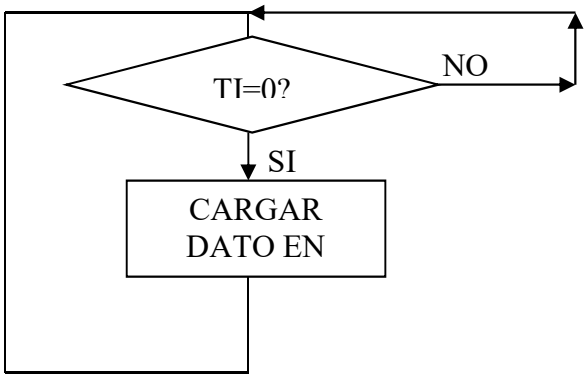
```
void uartbaudrate(UINT baudrate)
{
        #define FAUX  (OSC / (12 * 16))
        EA=0;
        ET1=0;
        TR1=0;
        PCON |= 0x80;
        TMOD &= 0x0F;
        TMOD |= 0x20;
        TH1 = (UCHAR) (256 - (FAUX / baudrate));
        TR1 = 1;
        EA = 1;
}
```

Para inicializar la UART bastara con poner en el main el registro de configuración de la UART (SCON) de la siguiente manera:

SCON=0x52;
Con esto queda configurada como UART de 8 bits de frecuencia variable y se habilita la transmisión serie.

- **Función de envió de datos mediante la UART**

Esta función nos permite enviar datos mediante la UART. Al principio de la función esperamos a que no haya ningún dato en el buffer de la UART ya que si no podríamos escribir encima de él. Cuando el bit TI del registro SCON se pone a 0 quiere decir que el microprocesador esta listo para transmitir el siguiente dato. Tras ponerse TI a 0 se carga el dato en el buffer del puerto serie (SBUF).
A continuación se presenta un diagrama de flujo para explicar este proceso:



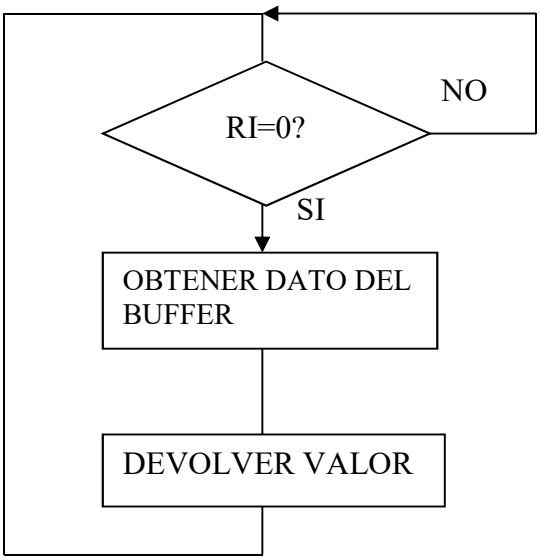
Una vez cargado en el buffer el dato es enviado automáticamente. A continuación se presenta el código en C para esta función:

```
UCHAR uartsend(UCHAR uc)
{
        while (TI == 0);
        TI   = 0;
        SBUF = uc;
        return uc;
}
```

- **Función de recepción de datos mediante la UART**

Esta función es similar a la anterior. En este caso disponemos del bit RI en el registro SCON que nos indica si se ha recibido un dato en el buffer de entrada. Tras esperar a que ese bit se ponga a 0 se carga el dato recibido en el buffer en una variable para que posteriormente sea devuelta por la función.

El siguiente diagrama de flujo explica gráficamente el funcionamiento de esta función:



El programa en C para la función es el siguiente:

```
UCHAR uartget(void)
{
        UCHAR uc;
        while (RI == 0);
        uc  = SBUF;
        RI  = 0;
        return uc;
}
```

- **Configuración del módulo de RF como transmisor o como receptor**

Él modulo de radiofrecuencia solo puede trabajar en uno de los dos estados, como transmisor o como receptor. Para ello tiene una patilla llamada TXEN que estando a uno le hace trabajar como emisor y estando a 0 como receptor además en estas funciones se ha incluido la patilla CS que permite seleccionar el canal y PWR_UP que se utiliza para apagar el dispositivo y dejarlo en standby.

Para realizar las funciones lo primero ha sido configurar las patillas que se utilizarían del micro como podremos ver a continuación:

```
sbit PWR_UP = P3^6;
sbit CS = P3^5;
sbit TXEN = P3^7;
```

Después de esto se han creado dos funciones que nos permiten intercambiar el estado del dispositivo:

```
void RF_transmisor(void) /* Modo Transmisor */
{
        PWR_UP = 1;
        CS = 0;
        TXEN = 1;
}

void RF_receptor(void) /* Modo Receptor */
{
        PWR_UP = 1;
        CS = 0;
        TXEN = 0;
}
```

Cabe destacar que es necesaria que transcurra un tiempo mínimo entre la configuración del dispositivo y la transmisión de los datos. Esto se muestra en la siguiente tabla:

Cambio de modo	Nombre	Tiempo mínimo	Condición
TX => RX	t_{TR}	3ms	Modo operacional
RX=>TX	t_{RT}	1ms	
Stdby => RX	t_{ST}	2ms	
Stdby => RX	t_{SR}	3ms	
VDD=0 => TX	t_{VT}	4ms	Inicio
VDD=0 => RX	t_{VR}	5ms	

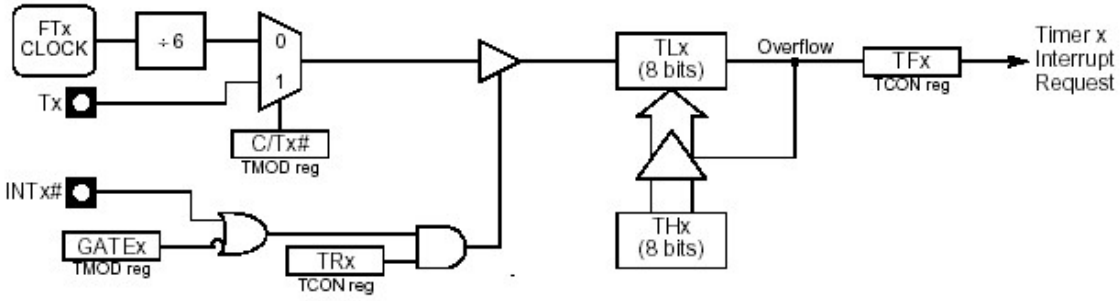
Estos tiempos deben ser respetados sino el dispositivo no funcionara correctamente. Para ello usaremos la función delay vista anteriormente.

3.2.2. FUNCIONES ESPECÍFICAS PARA EL VEHÍCULO.

Estas funciones solo las lleva el microcontrolador usado en el vehículo. Son las funciones de los temporizadores, de los motores y los protocolos de recepción y emisión específicos para el vehículo ya que el número de datos enviados y transmitidos y su procesamiento son diferentes en el mando y en el vehículo.

- **Función de los temporizadores**

Estas funciones son válidas tanto para el temporizador 0 como para el 1. Con la función de inicialización se pretende poner el temporizador en modo 2 (8 bits con autorrecarga). En el siguiente diagrama se da una explicación grafica del funcionamiento del timer en modo 2:



En modo de 8 bits con autorrecarga (modo 2) tenemos un timer de 8 bits TLx que automáticamente carga su valor desde el registro THx. Cuando el registro TLx se desborda se pone a 1 el flag de interrupción TFx que se encuentra en el registro TCON. Cuando la petición de interrupción es atendida este flag de interrupción se pone a 0 por hardware. El valor de THx puede cambiarse en cualquier momento y será cargado en TLx cada vez que se desborde el temporizador.

Para iniciar el temporizador en modo 2 es necesario poner los bits M1x y M0x del registro TMOD a 1 y a 0 respectivamente. Para ello pondremos dicho registro a TMOD = 0x22 ya que en el se incluye la configuración de ambos temporizadores. También se

habilitaran las interrupciones de desbordamiento de los temporizadores poniendo a 1 los bits ET0 y ET1 del registro de interrupciones IEN0.
A continuación se muestra la función de inicialización de ambos temporizadores en C:

```
void init_timer()
{
        TMOD = 0x22;
        EA=1;
        ET0=1;
        ET1=1;

}
```

Al desbordar el temporizador se entra en la subrutina de interrupción correspondiente a cada uno de ellos. Durante esta subrutina de interrupción lo único que se hace s cambiar el estado de un pin del puerto 4 para conseguir así una onda cuadrada. Cambiando el valor de con que se carga el temporizador conseguimos variar el periodo de la onda. A continuación se muestra el código en C correspondiente a las subrutinas de interrupción de los temporizadores 0 y 1:

```
Timer0_interrupt() interrupt 1
{
        EA=0;
        TR0=0;
        clock0 =~ clock0;
        TR0=1;
        EA=1;
}
```

```
Timer1_interrupt() interrupt 3
{
        EA=0;
        TR1=0;
        clock1 =~ clock1;
        TR1=1;
        EA=1;
}
```

Como se puede observar es necesario deshabilitar las interrupciones mientras estemos dentro de la subrutina de interrupción ya que podría llamarse al programa a otra interrupción y salir de estas. Las interrupciones van en orden de prioridad y este se establece por defecto de la forma que se indica en el siguiente diagrama:

Interrupt Name	Interrupt Address Vector	Priority Number
external interrupt (INT0)	0003h	1
Timer 0 (TF0)	000Bh	2
external interrupt (INT1)	0013h	3
Timer 1 (TF1)	001Bh	4
PCA (CF or CCFn)	0033h	5
UART (RI or TI)	0023h	6
Timer 2 (TF2)	002Bh	7
ADC (ADCI)	0043h	9

Para cambiar el nivel de prioridad de cada interrupción bastara con modificar los registros IPLx e IPHx que controlan el nivel de prioridad.

La siguiente función de los temporizadores será la función de carga del dato en el registro THx. Esta función no lleva más explicaciones debido a su simplicidad. Como dato cabe apuntar que tras la carga del temporizador se pone a 1 el bit TRx del temporizador para que este empiece la cuenta.

A continuación vemos el código en C de esta función para cada temporizador:

```
void funcion_reloj0(UCHAR ciclo)
{
        TH0 = ciclo;
        TR0=1;
}

void funcion_reloj1(UCHAR ciclo)
{
        TH1 = ciclo;
        TR1=1;
}
```

La siguiente función del temporizador es un paso previo a la carga del dato en el registro THx. Lo primero que tendremos será un dato procedente del microcontrolador del mando. Este dato es la captura directa realizada por el conversor A/D de uno de los potenciómetros que controlaran la velocidad y el sentido de giro de los motores. Lo primero que se hará es dividir en dos el recorrido del potenciómetro mediante software con esta función consiguiendo así tener dos partes del potenciómetro, la superior y la inferior. Tras haberlo dividido se mirara en que parte nos encontramos del recorrido y según este dato se pondrá a 1 o a 0 el bit que indica el sentido de giro del motor (CW/CCW). Tras esto es necesario que ajustemos al máximo valor que es posible darle al temporizador para así aprovechar toda la velocidad. Esto se hará con unas sencillas operaciones matemáticas. Así si el valor es por debajo de 127, que es la mitad del recorrido, pondremos el sentido de giro a 0 y multiplicaremos por 2 el valor para obtener un rango entre 0 y 255 (máxima capacidad del registro de 8 bits). En cambio si el valor que llega del potenciómetro es superior a 127 se pondrá el sentido de giro a 1 y se restara el dato a 255 tras lo cual el resultado se multiplicara por 2. Con esto tendremos el recorrido dividido en dos partes poniéndose a 0 el valor que se carga

en el registro del temporizador si el potenciómetro se encuentra en el centro de su recorrido y al máximo (255) si se encuentra en uno de los laterales del recorrido. Estas son las funciones utilizadas para esta tarea:

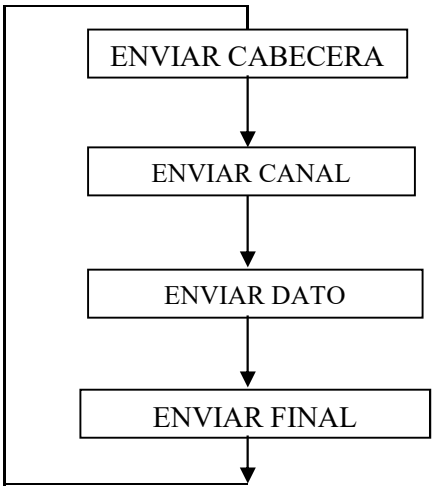
```
void reloj0(UCHAR salida)
        {
                UCHAR salida_final;
                if(salida >127)
                        {
                         CCW0=1;
                         salida = salida - 128;
                         salida = salida * 2;
                        salida_final= 255 - salida ;
                         funcion_reloj0(salida_final);
                        }
                else
                        {
                         CCW0=0;
                         salida = salida * 2;
                         funcion_reloj0(salida);
                         }
}
```

```
void reloj1(UCHAR salida1)
{
        UCHAR salida_final1;
        if(salida1 >127)
            {
             CCW1=1;
             salida1 = salida1 - 128;
             salida1 = salida1 * 2;
             salida_final1= 255 - salida1 ;
             funcion_reloj1(salida_final1);

            }
        else
            {
             CCW1=0;
             salida1 = salida1 * 2;
             funcion_reloj1(salida1);
             }
}
```

- **Función de emisión de datos mediante protocolo**

Los datos analógicos adquiridos de los sensores deben ser enviados al mando para su posterior procesamiento y visualización e la LCD. Como nuestra comunicación inalámbrica se realiza por infrarrojos o por radiofrecuencia y debido a las posibles interferencias que pueden sufrir estas comunicaciones se hace necesario un protocolo de comunicaciones para evitar datos erróneos. El protocolo creado para este proyecto es muy sencillo ya que la cantidad de datos transmitidos es muy pequeña y tan solo tenemos una comunicación entre dos dispositivos. Este protocolo de emisión se divide en tres partes. Lo primero es enviar una cabecera que le dirá al receptor que se prepare para recibir un dato. Tras la cabecera se mandan los datos propiamente dichos. En este caso tendremos únicamente 2 datos que serán el canal analógico del que estamos tomando las medidas y el dato obtenido de dicho canal. Tras los datos se envía un dato de cierre que nos indica que se han enviado los datos correctamente. El receptor y el emisor tienen asignado por software los datos de cabecera y final como 0xAA y 0x55 respectivamente. Se han elegido estos valores ya que entre bit y bit del Byte hay un cambio de estado. En el receptor se deben recibir estos valores ya que de otra forma el dato seria tratado como erróneo y se procedería a desecharlo y esperar al siguiente dato.

En este diagrama de flujo se explica brevemente el funcionamiento del protocolo emisor:



Como se puede observar en el diagrama de flujo no es necesario esperar que se cumplan condiciones durante el proceso de envío. Esto se debe a que el emisor únicamente manda los datos que posee junto con la cabecera y el final sin preocuparse del contenido. Esta será tarea del receptor que deberá recibir los datos y decidir si son válidos o no.

En la función en C realizada para esta función se ve como hay que indicar al llamar a la función el canal y el dato a enviar.

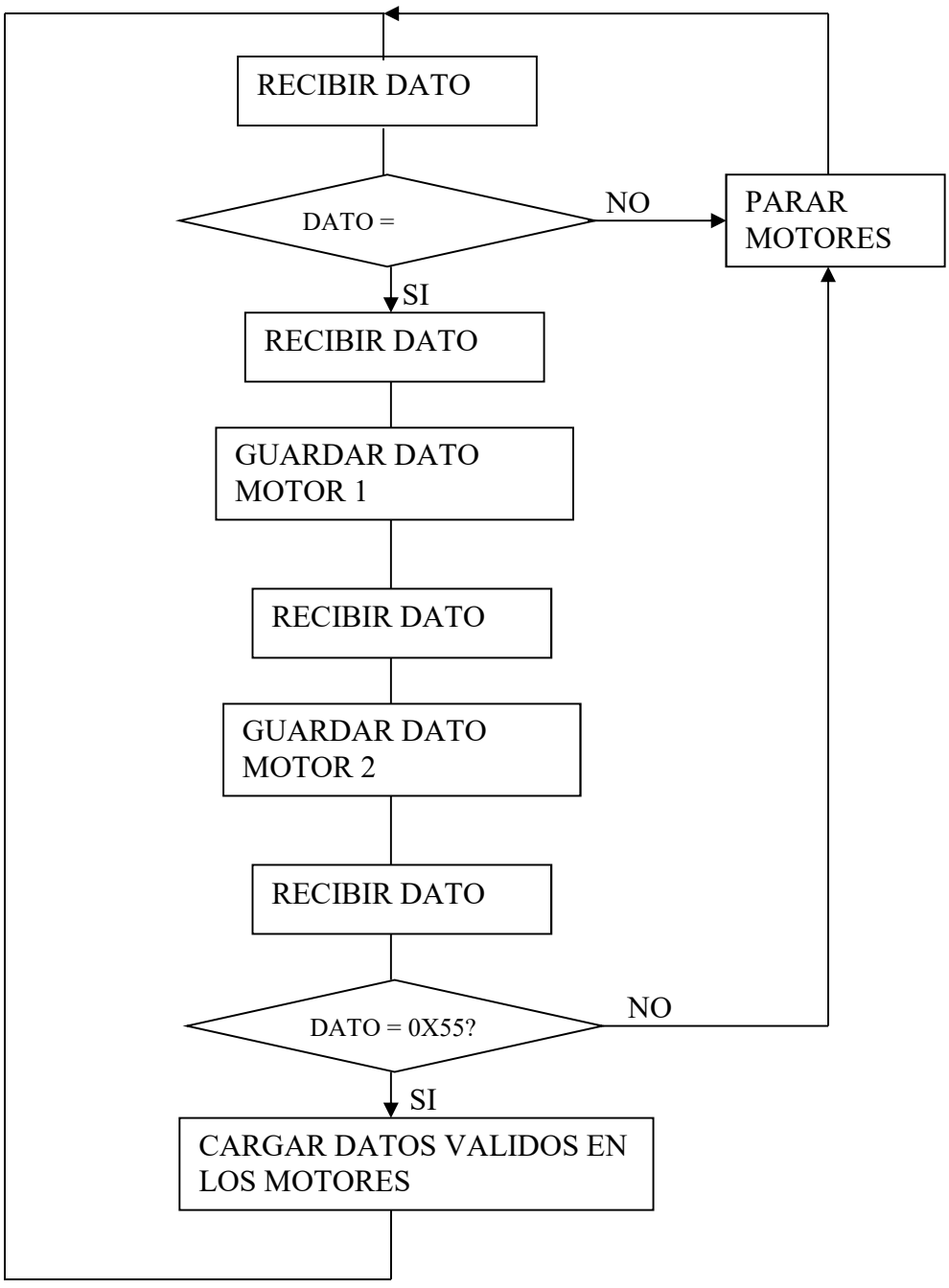
A continuación veremos su código en C:

```
void protocolo_emision_adc(UCHAR canal, UCHAR dato)
{
		uartsend(0xAA);
		delay(1);
		uartsend(canal);
		delay(1);
		uartsend(dato);
		delay(1);
		uartsend(0x55);
		delay(1);
}
```

- **Función de recepción de dato mediante protocolo**

Con esta función se pretende recibir un dato procedente del mando que ha sido enviado usando el protocolo de comunicaciones descrito para este proyecto. Ya que los datos que recibirá esta función proceden de los potenciómetros del mando y controlaran los motores del vehículo se ha creído conveniente implementar en esta función la carga de los datos en los temporizadores llamando dentro de esta misma función a las funciones que procesan los datos para ser cargados en el registro THx de cada uno de los temporizadores. Así pues el primer paso a dar es capturar los datos procedentes de la UART. Será necesario ejecutar cuatro veces esta función en el programa principal pues el paquete de datos se compondrá de 4 bytes: cabecera, doto del potenciómetro 1, dato del potenciómetro 2 y final. Para ello bastara con llamar a la función cuatro veces seguidas. Debido a que el primer dato recibido debe ser el 0xAA será lo primero en preguntar. Tras recibir este dato se preparara a recibir los dos siguientes guardándolos como los datos de cada uno de los motores. El cuarto dato debe ser el final y debe tener el valor 0x55. Si esto se cumple, los datos de los motores pasaran a actualizar los valores de los registros THx. De no cumplirse esto se desecharan y se pararan los motores ya que el error podría deberse a un fallo de comunicación y dejar al robot con los datos anteriores en los registros podría provocar la pérdida de control del vehículo por nuestra parte y ocasionar algún desperfecto en el vehículo o en el entorno por donde discurre este.

El siguiente diagrama de flujo explica gráficamente como se realiza la función.



El código en C de esta función se muestra a continuación:

```
UCHAR protocolo_recepcion(void)
{
UCHAR dato;
UCHAR dato_M1, dato_M2;
UCHAR paro;
int variable, var_dato;
int i;
variable=0;
var_dato=0;
 for(i=0;i<3;i++)
 {
        dato=uartget();
        if((dato== 0xAA) & (variable==0))
                {
                        variable=1;
                }
        else
                {
                        if(variable==1)
                                {
                                        if(var_dato==0)
                                                {
                                                        dato_M1=dato;
                                                        var_dato++;
                                                }
                                        else
                                                {
                                                        dato_M2=dato;
                                                        var_dato=0;
```

```
                                        variable++;
                                    }
                                }
                            else
                                {
                                    if(variable==2 & dato==0x55)
                                        {
                                            variable=0;
                                            if(dato_M1>0xED)
                                                dato_M1=0xED;
                                            else
                                                dato_M1=dato_M1;
                                            if(dato_M2>0xED)
                                                dato_M2=0xED;
                                            else
                                                dato_M2=dato_M2;
                                            reloj0(dato_M1);
                                            reloj1(dato_M2);
                                        }
                                    else
                                        {
                                            paro=0x80;
                                            reloj0(paro);
                                            reloj1(paro);
                                        }

                                }
                        }
                    return dato;
 }
}
```

3.2.3. FUNCIONES ESPECÍFICAS PARA EL MANDO

Estas funciones solo las utiliza el mando. Se trata de las funciones necesarias para la LCD, la interrupción externa que usaremos para seleccionar los canales que se visualizaran en la LCD y las funciones de emisión y recepción que incluyen el protocolo de comunicación.

Se presentan a continuación una serie de comandos que permiten configurar diferentes opciones de trabajo del módulo LCD y conseguir con ello distintos efectos de visualización. El juego de instrucciones consiste en diferentes códigos que se introducen a través del bus de datos del LCD conectado al micro.

- BORRAR PANTALLA

  Borra el modulo LCD y coloca el cursor en la 1ª posición (dirección 0). Pone el bit I/D a 1 por defecto.

  CODIGO

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	0	1

  Tiempo de ejecución: 1.64ms

- INICIO

Coloca el cursor en la posición de inicio (dirección 0) y hace que el display comience a desplazarse desde la posición original. El contenido de la memoria RAM de datos de visualización (DD RAM) permanecen invariables. La dirección de la memoria RAM de datos para la visualización (DD RAM) es puesta a 0.

CODIGO

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	1	D	C	B

Tiempo de ejecución: 1.64ms

- ESTABLECER MODO DE ENTRADA

Establece la dirección de movimiento del cursor y especifica si la visualización se va desplazando a la siguiente posición de la pantalla o no. Estas operaciones se ejecutan durante la lectura o escritura de la DD RAM o CG RAM (*Carácter Generador RAM*). Para visualizar normalmente se pone el bit **S** a "0".

CODIGO

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	1	I/D	S

Tiempo de ejecución: 40µs

I/D = “1” Se incrementa la dirección del cursor, con “0” sé decrementa

S=”1” Desplaza la visualización cada vez que se escribe un dato, Si = “0” funciona en modo normal.

- CONTROL DEL DISPLAY ON/OFF

Activa o desactiva poniendo en ON/OFF tanto al display (D) como al cursor (C) y se establece si este último debe o no parpadear (B).

CODIGO

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	1	D	C	B

Tiempo de ejecución: 40µs

B = “1” Parpadea el cursor
C = “1” Cursor activado
D = “1” Pantalla activada

- DESPLAZAMIENTO DE CURSOR O DISPLAY

Mueve el cursor y desplaza el display sin cambiar el contenido de la memoria de datos de visualización DD RAM.

CODIGO

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	1	S/C	R/L	X	X

Tiempo de ejecución: 40µs

S/C =”1” Desplaza la visualización; si es =”0” desplaza el cursor
R/L = “1” Desplazamiento a la derecha, si =”0” desplazamiento a la izquierda

- ESTABLECER FUNCION

Establece el tamaño de interface con el bus de datos (DL), número de líneas del display (N) y tipo de carácter (F).

CODIGO

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	DL	N	F	X	X

Tiempo de ejecución: 40µs

DL=”1” Trabaja en bus de 8 bits, si =”0” bus de 4 bits
N =“1” Presentación en 2 líneas, si = “0” se una línea
F=”1” caracteres de 5x10 pixel, si =”0” 5x7

- ESTABLECER DIRECCION DE LA CGRAM

El módulo LCD además de tener definidos todo el conjunto de caracteres ASCII, permite al usuario definir 4 u 8 caracteres gráficos. La composición de estos caracteres se va guardando en una memoria llamada CG RAM con capacidad para 64 bytes. Cada carácter gráfico definido por el usuario se compone de 16 u 6 bytes que se almacenan en sucesivas posiciones de la
CG RAM.

Mediante esta instrucción se establece la dirección de la memoria CG RAM a partir de la cual se irán almacenando los bytes que definen un carácter gráfico. Ejecutado este comando todos los datos que se escriban o se lean posteriormente, lo hacen desde esta memoria CG RAM.

CODIGO

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	Dirección de la CGRAM					

Tiempo de ejecución: 40µs

- ESTABLECER LA DIRECCION DE LA DDRAM

Los caracteres o datos que se van visualizando, se van almacenando previamente en una memoria llamada DD RAM para de aquí pasar a la pantalla. Mediante esta instrucción se establece la dirección de memoria DD RAM a partir de la cual se irán almacenado los datos a visualizar. Ejecutado este comando, todos los datos que se escriban o lean posteriormente los hacen desde esta memoria DD RAM. Las direcciones de la 80h a la 8Fh corresponden con los 16 caracteres del primer renglón y de la C0h a la CFh con los 16 caracteres del segundo renglón, para este modelo.

CODIGO

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	Dirección de la DDRAM						

Tiempo de ejecución: 40µs

- LEER EL SEÑALIZADOR “OCUPADO” Y DIRECCION DE LA RAM

Cuando el módulo LCD está ejecutando cualquiera de estas instrucciones, tarda un cierto tiempo de ejecución en el que no se le debe mandar ninguna otra instrucción. Para ello dispone de un flag llamado BUSY (BF) que indica que se está ejecutando una instrucción previa. Esta instrucción de lectura informa del estado de dicho flag además de proporcionar el valor del contador de direcciones de la CG RAM o de la DD RAM según la última que se haya empleado.

CODIGO

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	1	BF	Dirección de la DDRAM						

Tiempo de ejecución: 1µs

BF =”1” el módulo LCD está ocupado y si =”0” está disponible

- ESCRIBIR DATO A LA RAM

Mediante este comando se escribe en la memoria DD RAM los datos que se quieren presentar en pantalla y que serán los diferentes códigos ASCII de los caracteres a visualizar. Igualmente se escribe en la memoria CG RAM los diferentes bytes que permiten confeccionar caracteres gráficos a gusto del usuario.

El escribir en uno u otro tipo de memoria depende de sí se ha empleado previamente la instrucción de direccionamiento DD RAM o la de direccionamiento CG RAM.

CODIGO

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	0	Código ASCII o byte del carácter gráfico							

Tiempo de ejecución: 40µs

- LEER DATO DE LA RAM

Mediante este comando se lee de la memoria DD RAM los datos que haya almacenados y que serán los códigos ASCII de los caracteres visualizados. Igualmente se lee de la memoria
CG RAM los diferentes bytes con los que se ha confeccionado un determinado carácter gráfico.
El leer de uno u otro tipo de memoria depende de sí se ha empleado previamente la instrucción de direccionamiento de la DD RAM o la de direccionamiento CG RAM.

CODIGO

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	1	Código ASCII o byte del carácter gráfico							

Tiempo de ejecución: 40µs

NOTA: Algunos bits han sido abreviados en las tablas, a continuación explicamos su significado:

NOMENCLATURAS		
Abreviatura	Variable = 1	Variable = 0
I/D	I/D=1 Incrementa el Cursor en una posición	I/D=0 Decrementa el Cursor en una posición.
D	D=1 Pantalla Encendida	D=0 Pantalla Apagada.
C	C=1 Cursor Encendido.	C=0 Cursor Apagado.
B	B=1 Intermitencia del cursor encendida.	B=0 Intermitencia del cursor apagado
S/C	S/C=1 Mover todo el texto.	S/C=0 Mover el cursor.
R/L	R/L=1 Mover todo el texto a la izquierda.	R/L=1 Mover todo el texto a la derecha.
DL	DL=1 Bus de datos de 8 Bits.	DL=0 Bus de datos de 4 Bits.
S	S=1 Desplazamiento del texto.	S=0 No desplazamiento del texto
BF	BF=1 Operación Interna en progreso.	BF=0 No puede aceptar instrucción
F	F=1 Matriz para el carácter de 5 X 10 dots	F=0 Matriz del carácter de 5 x 7 Dost
N	N=1 Activación de dos líneas.	N=0 Activación de 1 línea

El módulo LCD ejecuta automáticamente una secuencia de inicio interna en el instante de aplicarle la tensión de alimentación si se cumplen los requisitos de alimentación. Dichos requisitos consisten en que el tiempo que tarde en estabilizarse la tensión desde 0.2 V hasta los 4.5V mínimos necesarios sea entre 0.1 ms y 10 ms. Igualmente el tiempo de desconexión debe ser como mínimo de 1 ms antes de volver a conectar.

La secuencia de inicio ejecutada es la siguiente:

1. Se ejecuta el comando **"BORRAR PANTALLA"** preparando la pantalla. El flag **BUSY** se mantiene a "1" (ocupado) durante 15 ms hasta que finaliza la inicialización.

2. Se ejecuta el comando "**ESTABLECER FUNCION"**, que establece el interfaz con el Bus de datos. Se elige por defecto el tamaño del bus de datos a 8 bits (DL=1) y el número de renglones del display en 1 (N=0).

3. Se ejecuta el comando "**CONTROL DEL DISPLAY ON/OFF",** que hace que el display quede en OFF (D=0); también cursor en OFF (C=0) y sin parpadeo del cursor en (B=0).

4. Se ejecuta el comando **"ESTABLECER MODO DE ENTRADA",** que establece la dirección de
Movimiento del cursor con autoincremento del cursor (I/D=1) y modo normal, no desplazamiento, del display (S=0).

Si la conexión de la alimentación no reúne las condiciones que exige el módulo LCD, habría que realizar la secuencia de inicialización por software. En cualquier caso, es importante enviar al LCD la primera instrucción de trabajo después de que hayan transcurrido 15 ms, para completar dicha secuencia de inicialización.

A continuación veremos las funciones en C para estos comandos que dispone la LCD:

- **Función para insertar espera en los pulsos**

Esta función sirve para insertar pequeños cambios de estado en el bit EN de habilitación de la LCD. Esto se hace para cargar distintos comando como los de inicialización de la LCD, el envío de datos y otros.

La función es muy sencilla y su código en C se muestra a continuación:

```
void habilita()
 {
   ENABLE=1;
   ENABLE=0;
  /*INSERTAR ESPERA EN LOS PULSOS*/
 }
```

- **Función para el envío de un carácter a la LCD**

Aquí se expondrán dos posibilidades para esta función. La LCD puede trabajar en modo de 4 bits y en modo de 8 bits. Para cada modo de funcionamiento la función es diferente ya que para el modo de 4 bits el bus de datos que llega a la LCD es de 4 líneas y el carácter tiene 8 bits así que tendremos que enviarlo en 2 partes.

A continuación veremos el programa para esta función:

```
void lcd_envia (unsigned char d)
{
   int i;
   PORT_LCD &= 0XDF;   /*selecciona modo escritura con el bit 5 a 0 */
   PORT_LCD &= 0XF0;   /*parte alta*/
   PORT_LCD |= d >> 4;
   habilita();
   PORT_LCD &= 0XF0;   /*parte baja*/
   PORT_LCD |= d & 0x0F;
   habilita();
    for(i=0; i<300;i++);

}
```

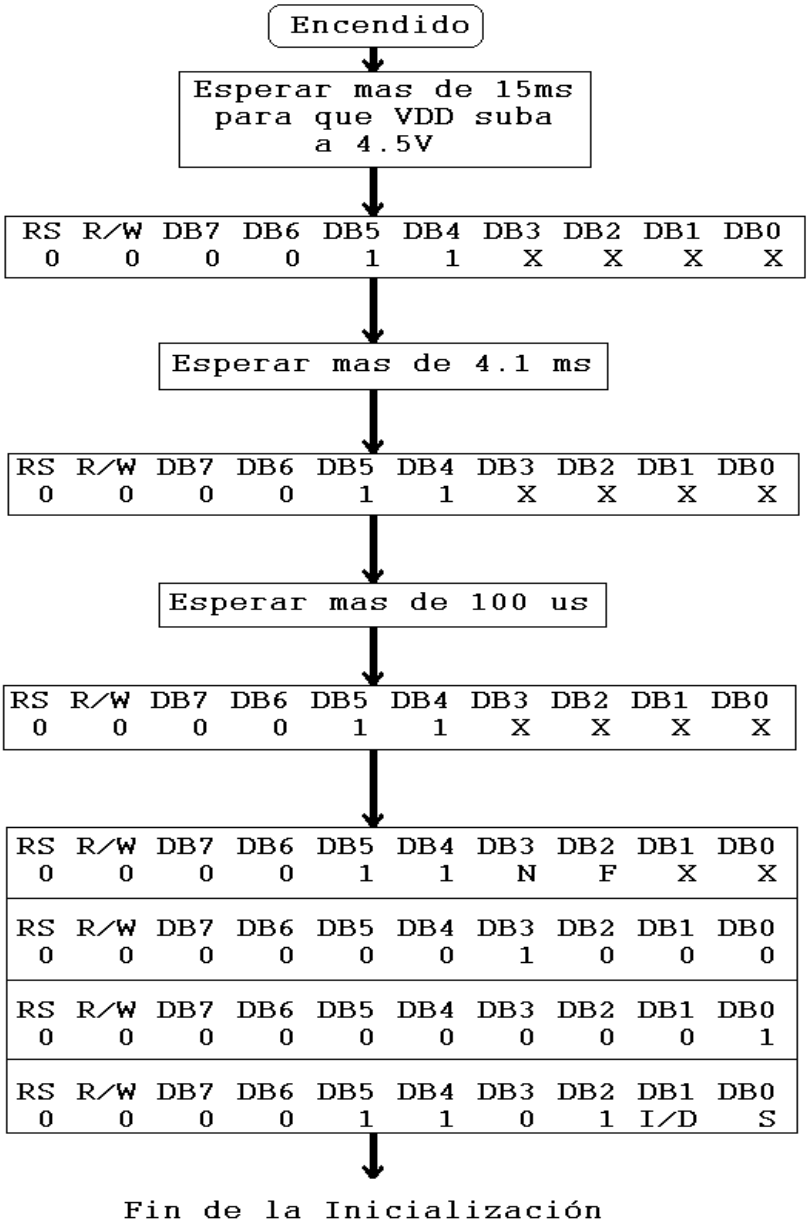
Para el modo de 8 bits no es necesario dividir el carácter en dos partes ya que tenemos las 8 líneas necesarias para el envío del carácter. La función es más sencilla y solo es necesario poner el dato en el puerto y enviarlo.

El código en C se muestra a continuación:

```
void lcd_envia (unsigned char d)
{
                                        int i;
  PORT_LCD &= 0XDF;   /*selecciona modo escritura con el bit 5 a 0 */
  PORT_LCD &= 0XFF;   /*dato completo*/
  PORT_LCD |= d;
  habilita();
   for(i=0; i<300;i++);

}
```

- **Función de inicialización de la LCD**

También aquí existen dos posibilidades ya que es aquí donde decidiremos el modo en el que vamos a usar la LCD. Las instrucciones para ambos casos son las mismas exceptuando una en la que elegiremos el modo de funcionamiento. Lo primero que haremos es poner una espera al principio de la función. Los pasos a seguir son los siguientes.



Según el diagrama de tiempos anterior se genera la siguiente función:

```
void init_lcd (void)
{
 unsigned int c;
 for(c=0;c<10000;c++);
   PORT_LCD &= 0X40;
   PORT_LCD |= 0X03;
   habilita();
 for(c=0;c<1000;c++);
   habilita();
 for(c=0;c<1000;c++);
   habilita();
 for(c=0;c<1000;c++);
 lcd_envia(0X38);/* bus de 8 bits */
 lcd_envia(0X0C);/* cursor apagado */
 lcd_envia(0X06);/* lcd no desplazado */
 lcd_envia(0X01);/* limpiar pantalla */

}
```

Si queremos ponerla en modo de 4 bits el código en C será:

```
void init_lcd (void)
{
 unsigned int c;
 for(c=0;c<10000;c++);
   PORT_LCD &= 0X40;
   PORT_LCD |= 0X03;
   habilita();
 for(c=0;c<1000;c++);
   habilita();
 for(c=0;c<1000;c++);
   habilita();
 for(c=0;c<1000;c++);
 lcd_envia(0X28);/* bus de 4 bits */
 lcd_envia(0X0C);/* cursor apagado */
 lcd_envia(0X06);/* lcd no desplazado */
 lcd_envia(0X01);/* limpiar pantalla */

}
```

- **Función para cambiar la posición del cursor en la LCD**

Esta función cambia de posición el cursor de la LCD par que se pueda empezar a escribir en el lugar deseado de la pantalla. Al llamar a la función basta con dar los valores de fila y columna donde se quiere situar el cursor. Lo primero en realizar es poner el bit DATO COMAND a 0 ya que este nos indica que lo siguiente en recibirse por el bus de datos es uno de los comandos que la propia LCD tiene almacenados en su memoria. Lo siguiente es enviarle el comando por el bus de datos.

En esta función no importa si hemos seleccionado bus de 4 bits o de 8 ya que es la función LCD_ENVIA la que divide el dato en dos partes o lo deja en 8 bits. Con la siguiente formula se consigue mandar el dato necesario siendo x la fila e y la columna donde se desea poner el cursor:

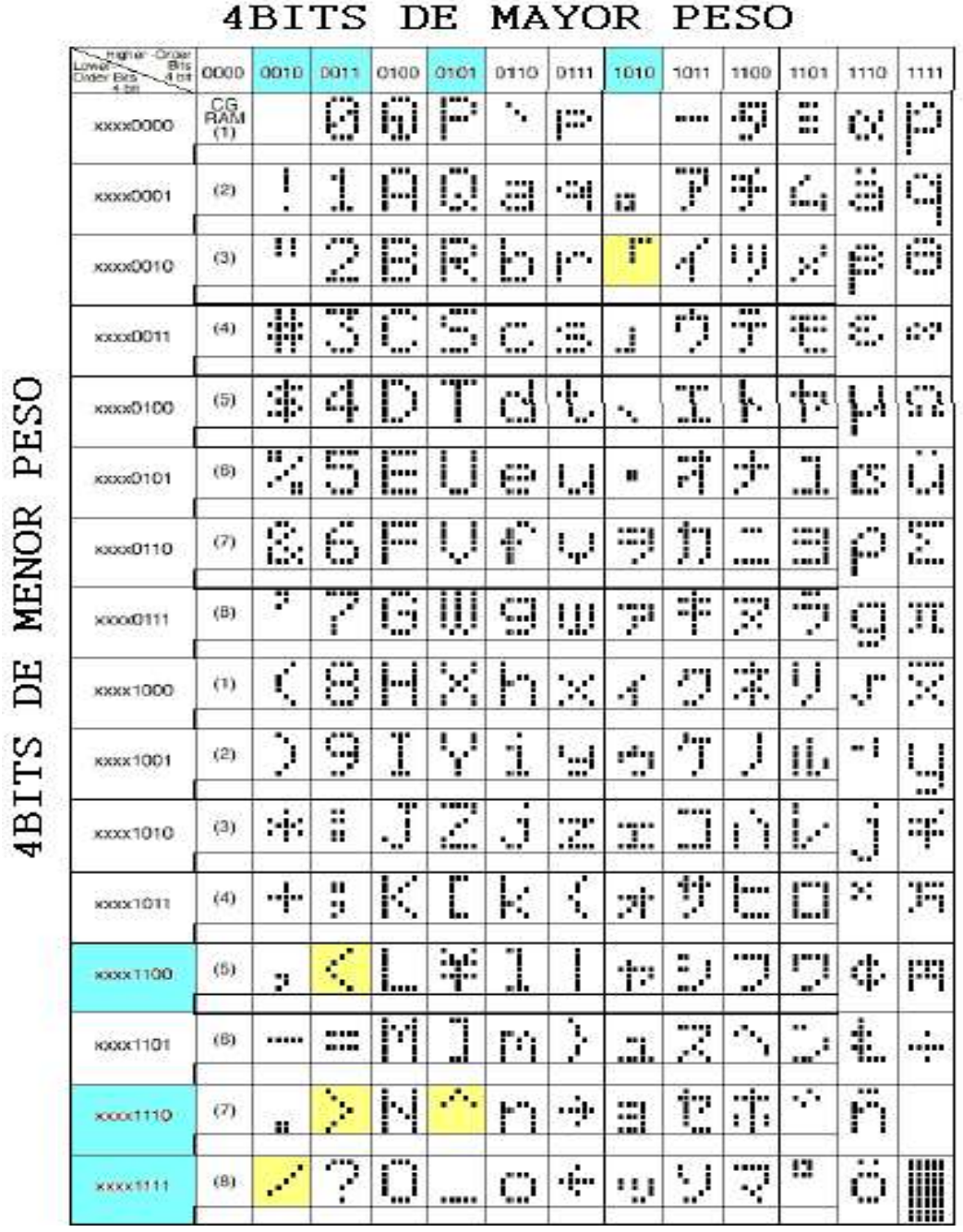
$$d = 0X80 + (x - 1) * 0X40 + y - 1;$$

El código en C de la función es el siguiente:

```
cursora_xy(UCHAR x,UCHAR y)
 {
  unsigned char d;
  DATO_COMAND=0;  /*COMANDO*/
  d = 0X80 + (x - 1) * 0X40 + y - 1;
  lcd_envia (d);
 }
```

- **Función para escribir un carácter en la posición del cursor**

El primer paso a realizar dentro de la función es indicarle a la LCD que se le va a enviar un dato para que lo ponga en la pantalla. Según el valor del dato enviado la LCD lo busca entre el juego de caracteres disponible en su memoria interna y lo saca por la pantalla. El juego de caracteres básico de que dispone la LCD usada en este proyecto es el siguiente:



Además de estos caracteres la LCD dispone de un función que nos permite agregarle uno definidos por el usuario.

Es necesario poner el bit DATO_COMAND a 1 lo que nos indica que se le va a enviar un dato. Lo siguiente será enviarle el dato. Esto se puede ver en la función en C:

```
void lcd_caracter(unsigned char c)
 {
 DATO_COMAND=1;  /*DATO*/
 lcd_envia (c);
 }
```

- **Función para escribir un string de caracteres**

Con esta función se pretende simplificar el envío de datos en forma de string. Pongamos el ejemplo de que se quiere enviar una cadena de caracteres formando una frase. Sin esta función sería necesario ir mandando uno a uno cada carácter de la cadena. Con la función no es necesario ya que poniéndole la cadena de caracteres que se quiere enviar ella misma direcciona con un puntero el carácter a enviar y lo manda a la LCD. Veamos su código en C:

```
display(UCHAR*msg)
 {
   while(*msg)
   {
     lcd_caracter(*msg++);
   }
  }
```

- **Función para hacer visible o invisible el cursor de la LCD**

Hay casos en los que nos parece necesario ver el lugar en el que se encuentra el cursor. Por ejemplo si se pide escribir ciertos caracteres en ciertos lugares y los visualizamos con la LCD(por ejemplo un código secreto). Para esto la LCD dispone de una función que permite la visualización del cursor. Con esta función en C se pretende facilitar su implementación y hacerla más portable como en los casos anteriores.

```
void lcd_cursoron (void)
{
 ENABLE=0;
 DATO_COMAND=0;  /*COMANDO*/
 lcd_envia (0X0F);
}
```

También puede darse el caso en el que teniéndolo visible queramos que ya no lo esté para poder ver mejor unos datos que manda el micro (por ejemplo visualizar la temperatura). En estos casos se utiliza la siguiente función:

```
void lcd_cursoroff (void)
{
 ENABLE=0;
 DATO_COMAND=0;  /*COMANDO*/
 lcd_envia (0X0C);
}
```

- **Función para borrar la pantalla de la LCD**

Cuando se han visualizado unos datos se hace necesario limpiar la LCD ya que esta función, incluida por el fabricante, no borra todos los datos de la pantalla y nos pone el cursor en su posición inicial. Su código en C es:

```
void lcd_clrscr (void)
{
 ENABLE=0;
 DATO_COMAND=0;  /*COMANDO*/
 lcd_envia (0X01);
}
```

- **Funciones para cambiar la línea de escritura**

Estas funciones nos permiten colocar el cursor en la línea 1 o en la 2. También vienen dadas por el fabricante. Su código en C es el siguiente:

```
-Para poner en la línea 1:
poner_linea_1()
 {
   DATO_COMAND=0;  /*COMANDO*/
   lcd_envia (0X80);
 }
-Para poner en la línea 2:
poner_linea_2()
 {
   DATO_COMAND=0;  /*COMANDO*/
   lcd_envia (0XC0);
 }
```

- **Función para enviar el cursor al rincón**

Sirve para colocar el cursor de la LCD en el rincón inferior derecho de la LCD. Su código en C es:

```
void lcd_rincon (void)
{
 ENABLE=0;
 DATO_COMAND=0;  /*COMANDO*/
 lcd_envia (0x02);
}
```

- **Interrupción externa del mando**

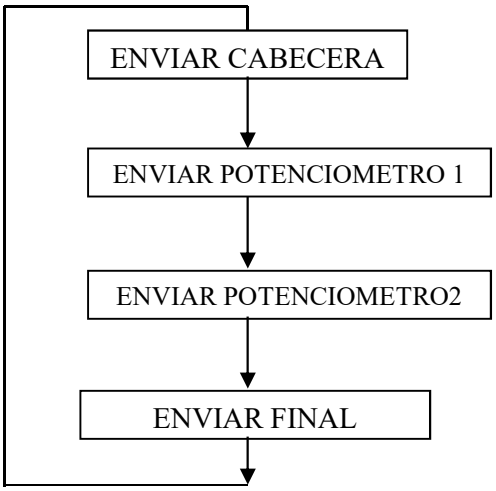
Esta interrupción se usara para cambiar los canales que se verán en la LCD. Con cada flanco descendente en este pin se activa la interrupción. En la subrutina de interrupción se cambia el número de canales que se van a ver en la LCD. Al principio se verán los canales 0 y 1(uno en cada línea) y con cada pulsación se verán los dos siguientes. El código en C de la subrutina de interrupción es:

```
intExt0() interrupt 0
{
                                    if(variable_canal = 3)
                                     variable_canal=0;
                                    else
                                     variable_canal=variable_canal + 1;
}
```

La variable_canal nos definirá que 2 canales se verán en la LCD.

- **Función de protocolo de emisión de datos**

Al igual que en el vehículo se hace necesario el uso de un protocolo para el mando ya que estos dos dispositivos se comunican entre sí. Es por ello que el protocolo será el mismo y las funciones muy similares ya que ahora lo que se enviara es el estado de los potenciómetros y lo que se recibirá es el canal y el dato de dicho canal analógico. El diagrama de flujo es el siguiente para el emisor:

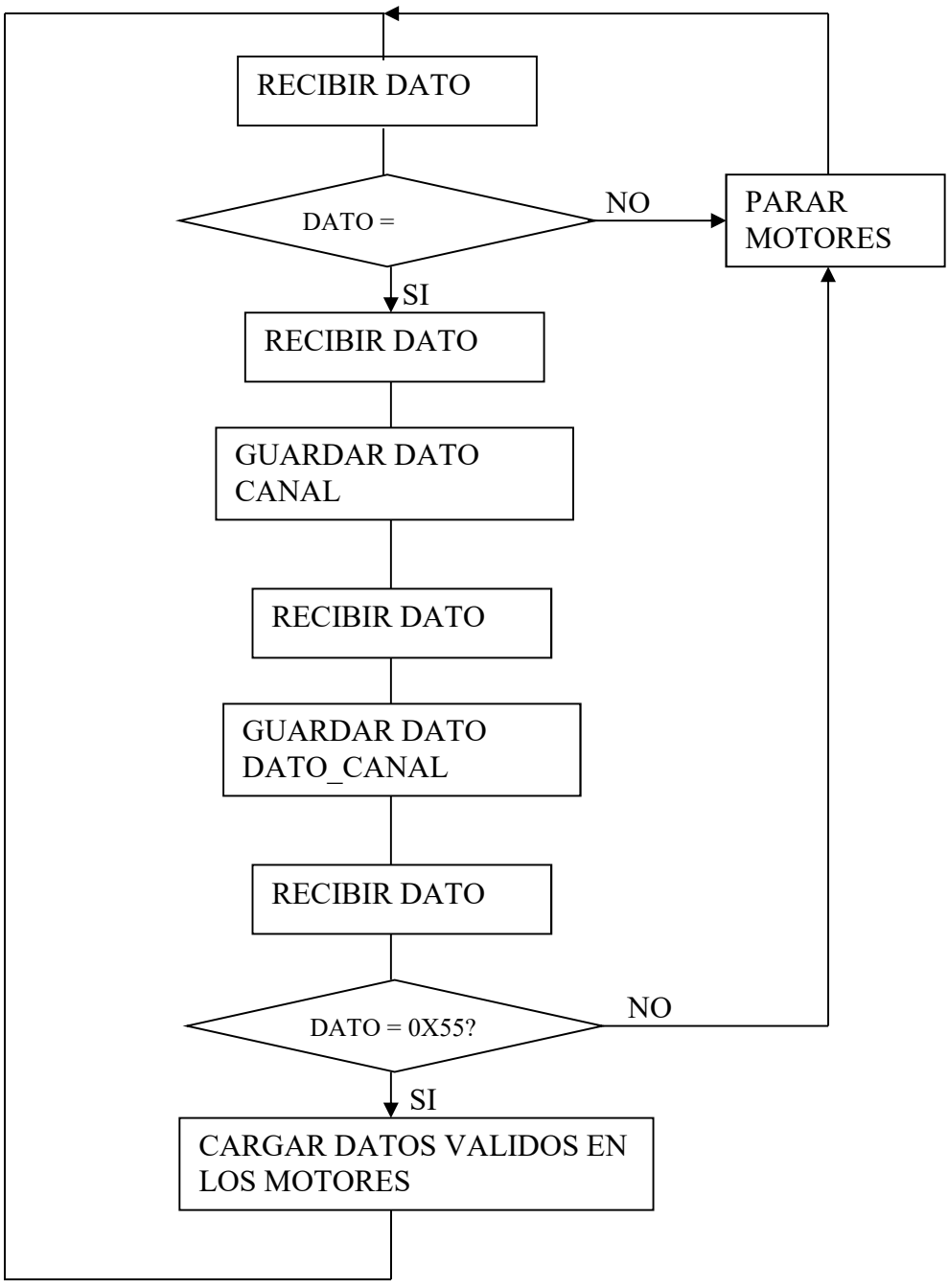


Como se puede observar en el diagrama de flujo no es necesario esperar que se cumplan condiciones durante el proceso de envió. Esto se debe a que el emisor únicamente manda los datos que posee junto con la cabecera y el final sin preocuparse del contenido. Esta será tarea del receptor que deberá recibir los datos y decidir si son válidos o no.

- **Función de recepción de dato mediante protocolo**

Con esta función se pretende recibir un dato procedente del mando que ha sido enviado usando el protocolo de comunicaciones descrito para este proyecto. Ya que los datos que recibirá esta función proceden de los canales analógicos y se verán en la LCD se ha creído conveniente implementar en esta función la visualización de los datos en la LCD llamando dentro de esta misma función a las funciones que procesan los datos para ser cargados en la LCD. Así pues el primer paso a dar es capturar los datos procedentes de la UART. Será necesario ejecutar cuatro veces esta función en el programa principal pues el paquete de datos se compondrá de 4 bytes: cabecera, doto del canal, dato_canal y final. Para ello bastara con llamar a la función cuatro veces seguidas. Debido a que el primer dato recibido debe ser el 0xAA será lo primero en preguntar. Tras recibir este dato se preparara a recibir los dos siguientes guardándolos como los datos de canal y dato_canal. El cuarto dato debe ser el final y debe tener el valor 0x55. Si esto se cumple, los datos de canal y dato_canal pasaran a actualizar los valores de las variables que contendrán el valor anterior de cada canal analógico .De no cumplirse esto se desecharan y se dejaran los datos anteriores.

El siguiente diagrama de flujo explica gráficamente como se realiza la función.



El código en C de esta función se muestra a continuación:

```
UCHAR protocolo_recepcion_analogicos(void)
{
 UCHAR dato;
 UCHAR canal_adc, dato_adc;
 int variable, var_dato;
 int i;

 variable=0;
 var_dato=0;
 for(i=0;i<3;i++)
 {
        dato=uartget();
        if((dato== 0xAA) & (variable==0))
                {
                        variable=1;
                }
        else
                {
                        if(variable==1)
                                {
                                        if(var_dato==0)
                                                {
                                                        canal_adc=dato;
                                                        var_dato++;
                                                }
                                        else
                                                {
                                                        dato_adc=dato;
                                                        var_dato=0;
```

```
                    variable++;
                }
            }
        else
            {
                if(variable==2 & dato==0x55)
                {
                    variable=0;
                    switch(canal_adc)
                    {
                        case 0x00:
                            dato_CANAL0=dato_adc;
                            break;
                        case 0x01:
                            dato_CANAL1=dato_adc;
                            break;
                        case 0x02:
                            dato_CANAL2=dato_adc;
                            break;
                        case 0x03:
                            dato_CANAL3=dato_adc;
                            break;
                        case 0x04:
                            dato_CANAL4=dato_adc;
                            break;
                        case 0x05:
                            dato_CANAL5=dato_adc;
                            break;
                        case 0x06:
                            dato_CANAL6=dato_adc;
                            break;
```

```
                                        case 0x07:
                                            dato_CANAL7=dato_adc;
                                            break;
                                        default:
                                            dato_adc=0x00;
                                            canal_adc=0x00;
                                    }
                                }
                            else
                                {
                                dato_adc=0x00;
                                canal_adc=0x00;
                                }

                        }
                }
                return dato;
                        }
}
```

3.2.4. PROGRAMA PRINCIPAL DEL VEHÍCULO

A continuación se explicara el programa principal del vehículo. Lo primero será declarar variables globales y los archivos .h necesarios para la ejecución del programa. Después se pondrá el main donde reside la parte principal del programa.

- **Declaración de variables globales**

A continuación pondremos el código en C para la declaración de las variables globales y después se explicara:

```
#include "89c51AC2.h"
#define UCHAR unsigned char
#define UINT unsigned int
#define OSC 11059200L

sbit clock0 = P4^0;
sbit clock1 = P4^1;

sbit RST0 = P0^5;
sbit HALF0 = P0^4;
sbit CCW0 = P0^3;
sbit CTRL0 = P0^2;
sbit EN0 = P0^1;

sbit RST1 = P2^0;
sbit HALF1 = P2^1;
sbit CCW1 = P2^2;
sbit CTRL1 = P2^3;
sbit EN1 = P2^4;
```

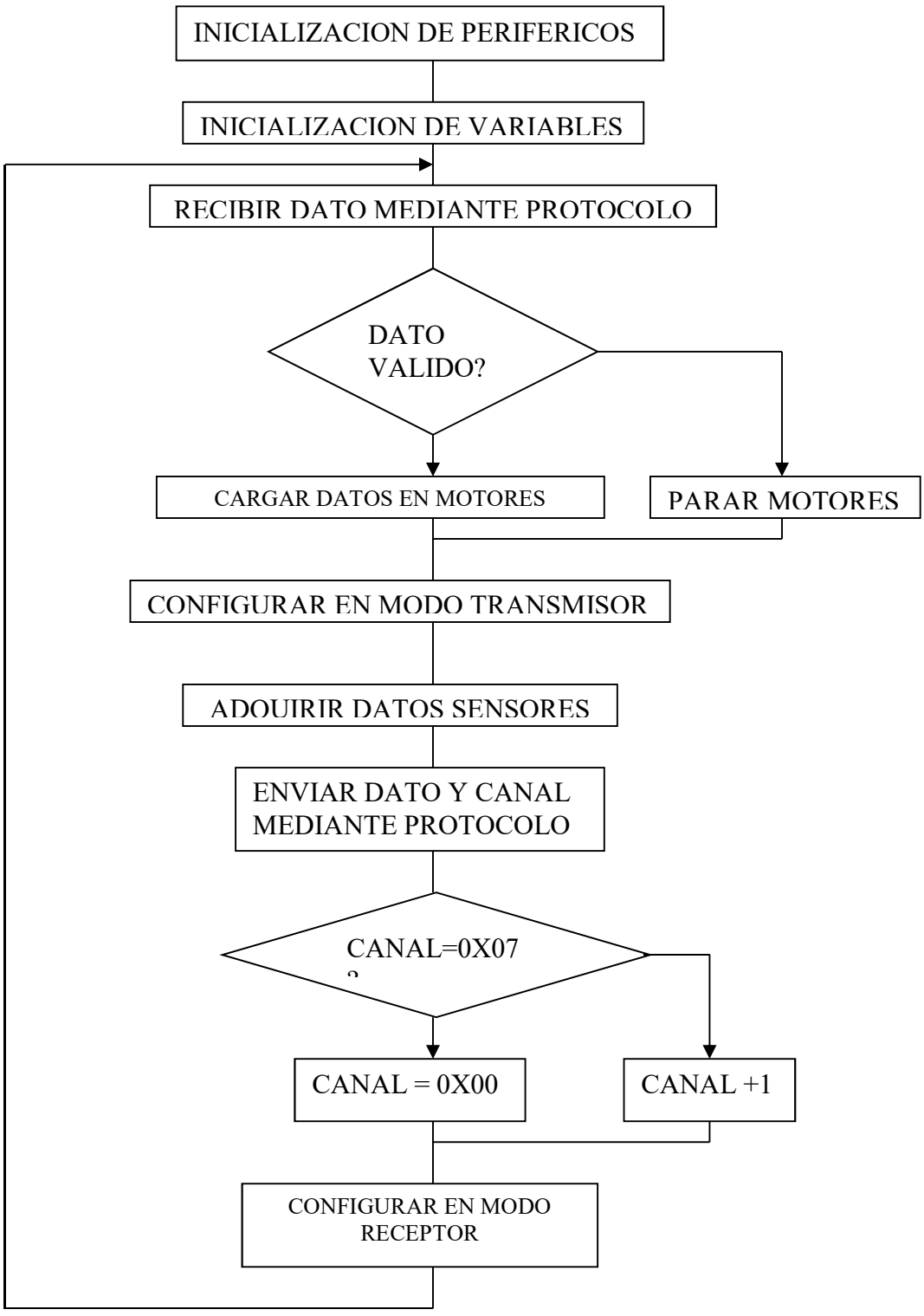
```
sbit RST_DIV=P2^6;

sbit PWR_UP = P3^6;
sbit CS = P3^5;
sbit TXEN = P3^7;

#define CANAL0 = 0x00;
#define CANAL1 = 0x01;
#define CANAL2 = 0x02;
#define CANAL3 = 0x03;
#define CANAL4 = 0x04;
#define CANAL5 = 0x05;
#define CANAL6 = 0x06;
#define CANAL7 = 0x07;
```

Lo primero es definir el archivo que contiene los SFRs. Este archivo define todos los puertos y registros del micro. En este caso el archivo es 89C51AC2.h . Seguidamente definimos UCHAR y UINT como unsigned char y unsigned int para una fácil declaración de este tipo de variables. Se define OSC que será el valor del oscilador formado por el cristal de cuarzo. A continuación se definen varios bits de puertos distintos usados para el control de motores y comunicaciones. Esto se hace para facilitar la programación y evitar confundirse de pines. Se han definido los canales con sus respectivos valores para el uso en el conversor A/D y en el propio main.

- **Programa principal(main)**

Lo primero dentro del main será inicializar los periféricos usados en el vehículo. Después se especificaran condiciones iniciales de varios pines y se pasara al programa principal que va dentro de un bucle while para que el programa se ejecute durante todo el tiempo.

A continuación se presenta un diagrama de flujo para explicar el funcionamiento del programa principal:



A continuación se verá el programa en C de la función principal main:

```
void main()
{
	UCHAR canal_adc,dato_adc;
	SCON=0x52;
	uartbaudrate(300);
	init_timer();
	adc_init();
	clock0=0;
	clock1=0;
	init_M1();
	init_M2();
	RF_receptor();
	delay(6);
	canal_adc=0x00;
	while(1)
	{
		protocolo_recepcion();
		RF_transmisor();
		dato_adc=get_adc(canal_adc);
		protocolo_emision_adc(canal_adc,dato_adc);
		if(canal_adc=0x07)
			canal_adc=0x00;
		else
			canal_adc=canal_adc + 1;
		delay(3);
		RF_receptor();
		delay(4);
	 }
}
```

3.2.5. PROGRAMA PRINCIPAL DEL MANDO

A continuación se explicara el programa principal del mando. Lo primero será declarar variables globales y los archivos .h necesarios para la ejecución del programa. Después se pondrá el main donde reside la parte principal del programa.

- **Declaración de variables globales**

A continuación pondremos el código en C para la declaración de las variables globales y después se explicara:

```
#include "89c51AC2.h"
#define UCHAR unsigned char
#define UINT unsigned int
#define OSC 11059200L

sbit clock0 = P4^0;
sbit clock1 = P4^1;

sbit RST0 = P0^5;
sbit HALF0 = P0^4;
sbit CCW0 = P0^3;
sbit CTRL0 = P0^2;
sbit EN0 = P0^1;

sbit RST1 = P2^0;
sbit HALF1 = P2^1;
sbit CCW1 = P2^2;
sbit CTRL1 = P2^3;
sbit EN1 = P2^4;
```

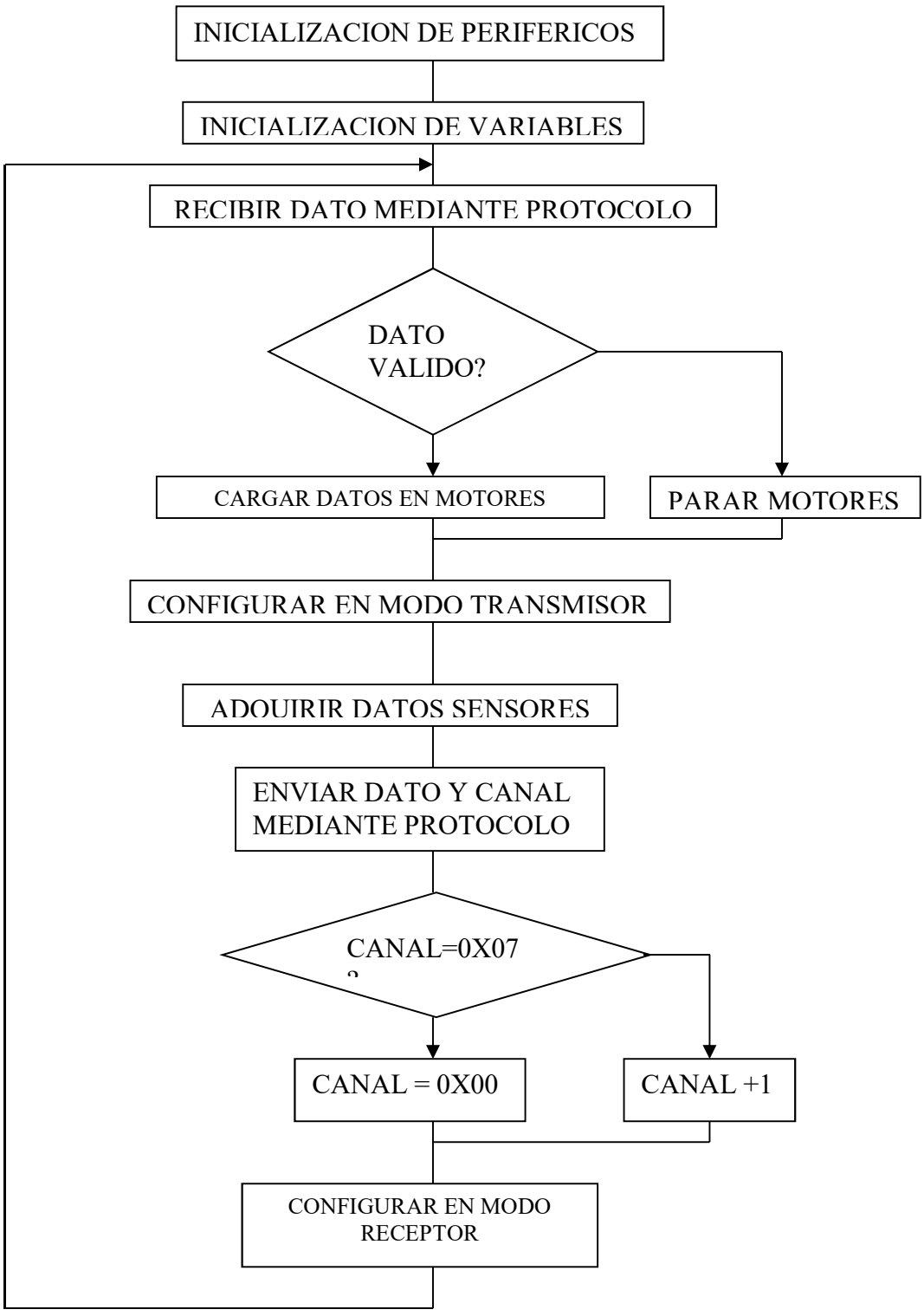
```
sbit RST_DIV=P2^6;

sbit PWR_UP = P3^6;
sbit CS = P3^5;
sbit TXEN = P3^7;

#define CANAL0 = 0x00;
#define CANAL1 = 0x01;
#define CANAL2 = 0x02;
#define CANAL3 = 0x03;
#define CANAL4 = 0x04;
#define CANAL5 = 0x05;
#define CANAL6 = 0x06;
#define CANAL7 = 0x07;
```

Lo primero es definir el archivo que contiene los SFRs. Este archivo define todos los puertos y registros del micro. En este caso el archivo es 89C51AC2.h . Seguidamente definimos UCHAR y UINT como unsigned char y unsigned int para una fácil declaración de este tipo de variables. Se define OSC que será el valor del oscilador formado por el cristal de cuarzo. A continuación se definen varios bits de puertos distintos usados para el control de motores y comunicaciones. Esto se hace para facilitar la programación y evitar confundirse de pines. Se han definido los canales con sus respectivos valores para el uso en el conversor A/D y en el propio main.

- **Programa principal(main)**

Lo primero dentro del main será inicializar los periféricos usados en el vehículo. Después se especificaran condiciones iniciales de varios pines y se pasara al programa principal que va dentro de un bucle while para que el programa se ejecute durante todo el tiempo.

A continuación se presenta un diagrama de flujo para explicar el funcionamiento del programa principal:



A continuación se verá el programa en C de la función principal main:

```
void main()
{
	UCHAR canal_adc,dato_adc;
	SCON=0x52;
	uartbaudrate(300);
	init_timer();
	adc_init();
	clock0=0;
	clock1=0;
	init_M1();
	init_M2();
	RF_receptor();
	delay(6);
	canal_adc=0x00;
	while(1)
	{
		protocolo_recepcion();
		RF_transmisor();
		dato_adc=get_adc(canal_adc);
		protocolo_emision_adc(canal_adc,dato_adc);
		if(canal_adc=0x07)
			canal_adc=0x00;
		else
			canal_adc=canal_adc + 1;
		delay(3);
		RF_receptor();
		delay(4);
	 }
}
```

4. PLANOS

4.1. INTRODUCCION

Para el diseño de los planos se ha utilizado el programa **ORCAD** release 9, utilizando dos de las herramientas que contiene, el **Layout** y el **Capture**.

El capture ha sido utilizado para realizar los esquemas del circuito, mientras que el layout se utiliza para el ruteado de las pistas y disposición de los componentes en la placa.

La selección de este programa se debe a los conocimientos previos que tenemos y que es una herramienta de gran potencia y facilidad de uso.

4.2. PLANOS DEL VEHÍCULO

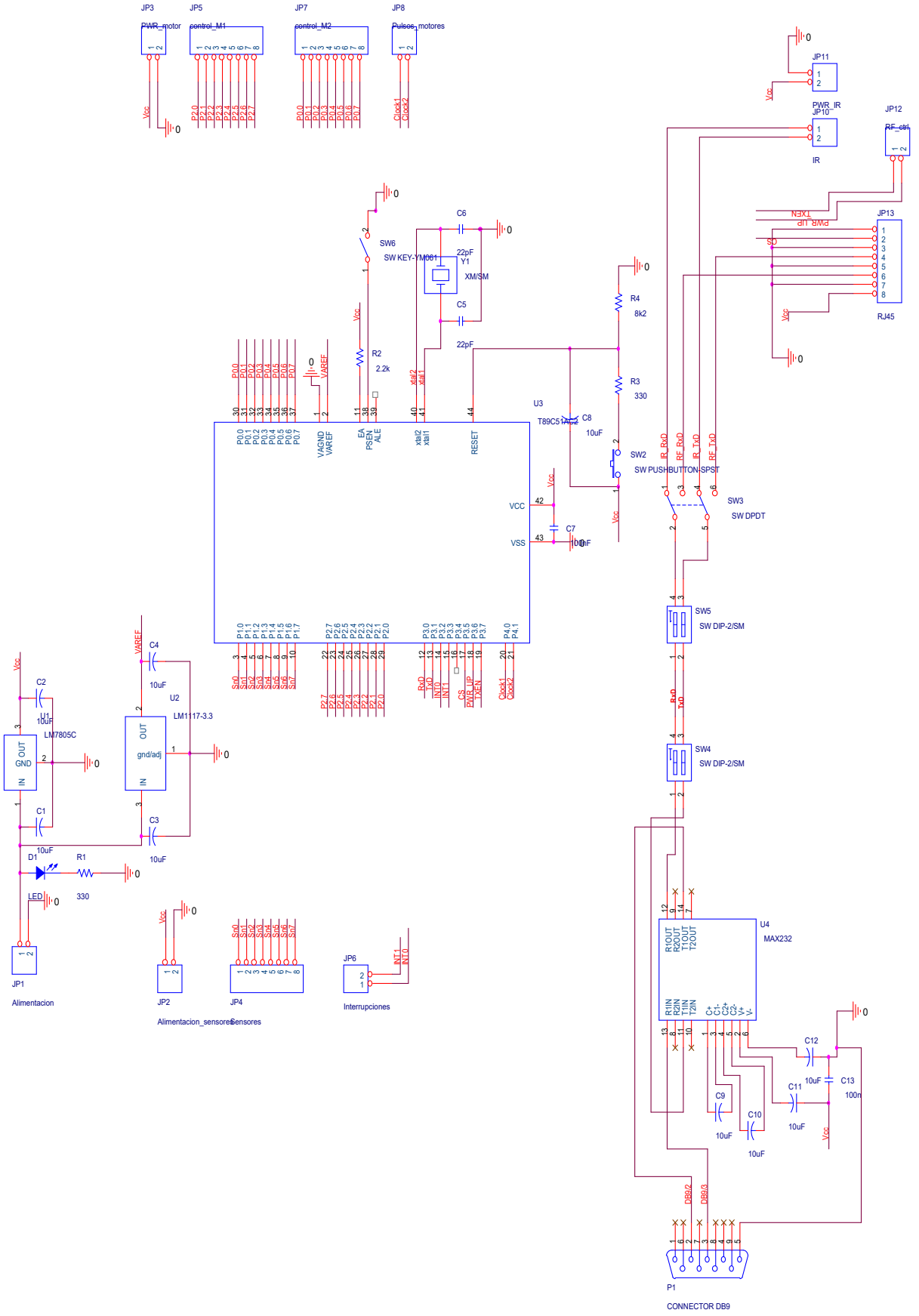
A continuación se expondrán cada una de las distintas placas que componen el vehículo, mostrando el esquemático, el trazado de pistas y la disposición de componentes.

4.2.1. PLACA CENTRAL (MICROCONTROLADOR)

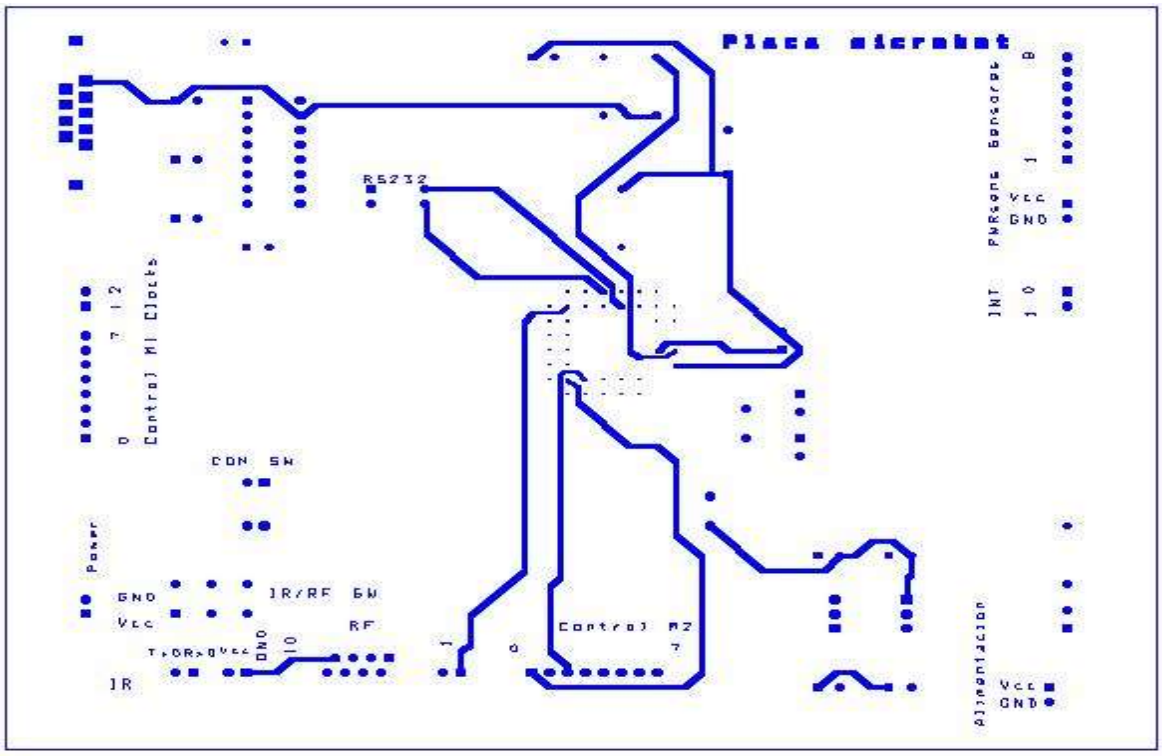
En esta placa se encuentra el “cerebro” del proyecto, el microcontrolador. Así pues este dispone del programador, con la comunicación serie necesaria para esta función, alimentación, circuito de reseteado y oscilador. La placa, a su vez, dispone de puertos de control para los motores y comunicaciones, así como los interruptores que permitirán cambiar de un tipo de comunicación a otra.

- **Esquemático**

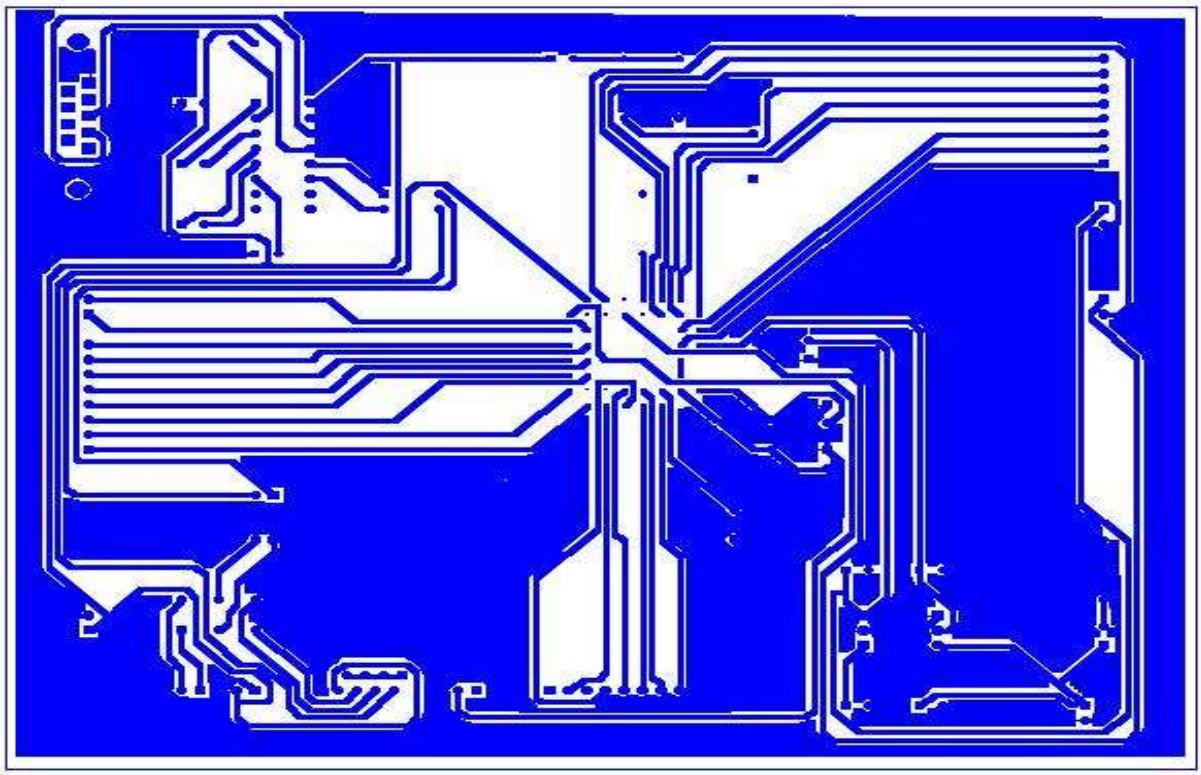
Página siguiente.


JP3
PWR_motor
JP5
control_M1
JP7
control_M2
JP8
Pulsos_motores
JP11
JP10
PWR_IR
IR
JP12
RF_ctrl
JP13
RJ45
SW6
SW KEY-YM061
C6
22pF
Y1
XM/SM
C5
22pF
R2
2.2k
R4
8k2
R3
330
U3
T89C51AD2
C8
10uF
SW2
SW PUSHBUTTON_SPST
SW3
SW DPDT
C7
10uF
SW5
SW DIP-2/SM
SW4
SW DIP-2/SM
C4
10uF
U2
LM1117-3.3
C2
10uF
U1
LM7805C
C1
10uF
C3
10uF
D1
R1
330
LED
JP1
Alimentacion
JP2
Alimentacion_sensores
JP4
Sensores
JP6
Interrupciones
U4
MAX232
C12
10uF
C13
100n
C11
10uF
C9
10uF
C10
10uF
P1
CONNECTOR DB9

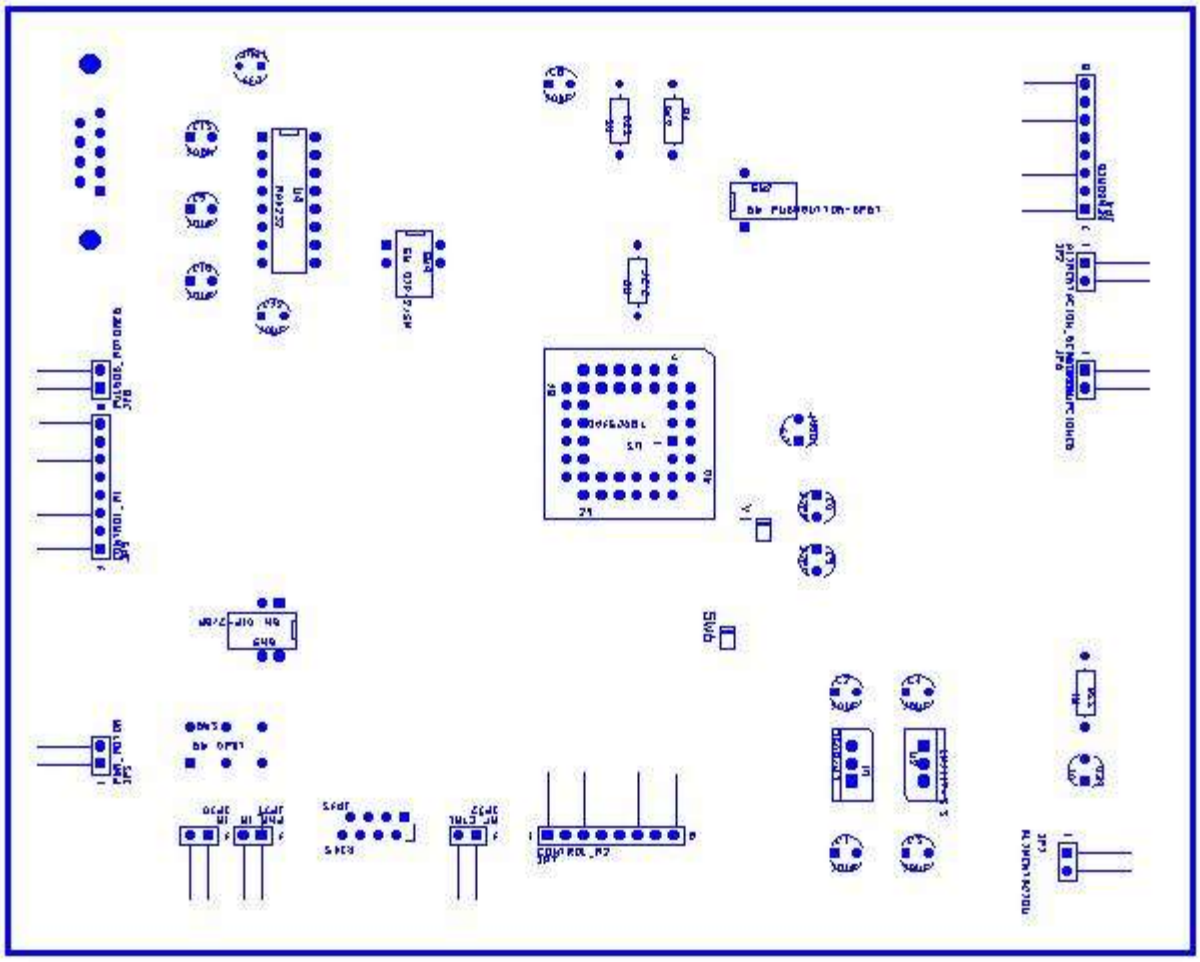
- **Trazado de pistas cara TOP**



- **Trazado de pistas cara BOTTOM**



- **Disposición de componentes**

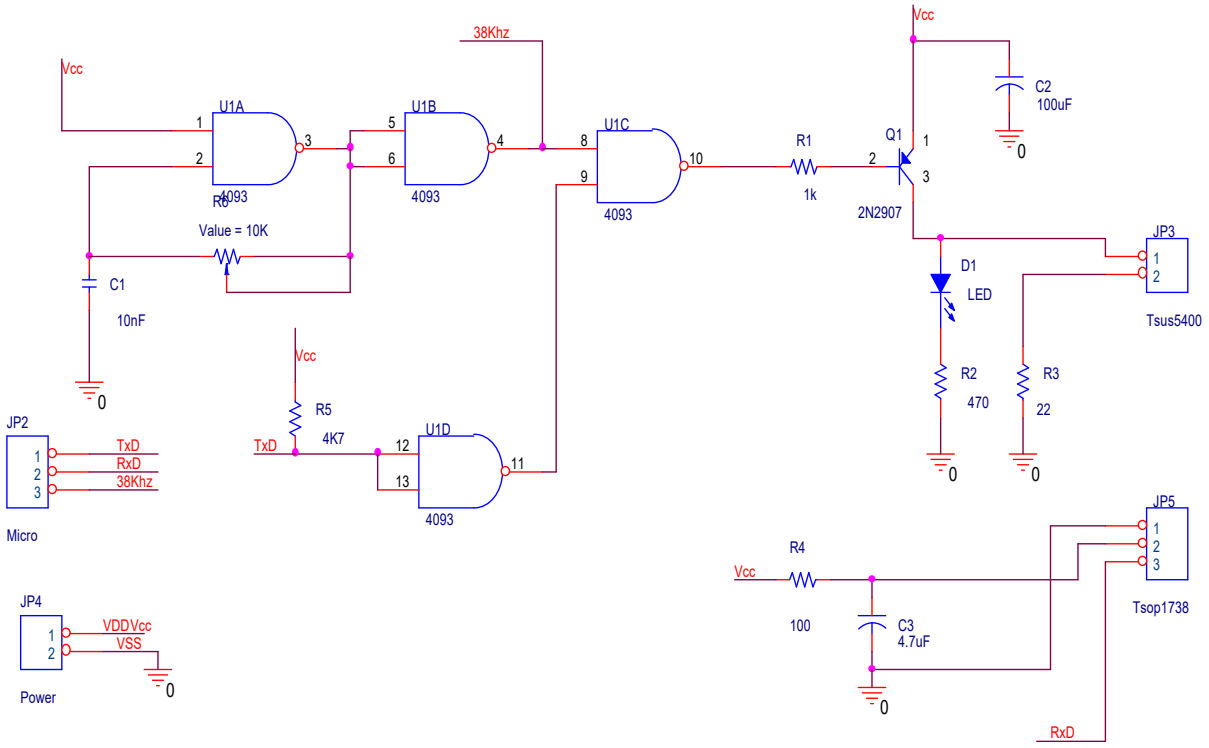


Para la disposición de los componentes se han tenido varios factores en cuenta. El primero ha sido la propia potencia del programa ya que para conseguir un trazado de pistas del 100% no es válida cualquier distribución de componentes. Otro factor tenido en cuenta son nuestras propias necesidades, ya que para una fácil conexión entre las placas, es necesaria una sencilla accesibilidad a los puertos. Por ello se han colocado los conectores en la periferia de la placa, pero con un orden preestablecido, para que el cableado necesario para conectarla con el resto de placas que componen el vehículo no fuera demasiado extenso evitando líos de cables.

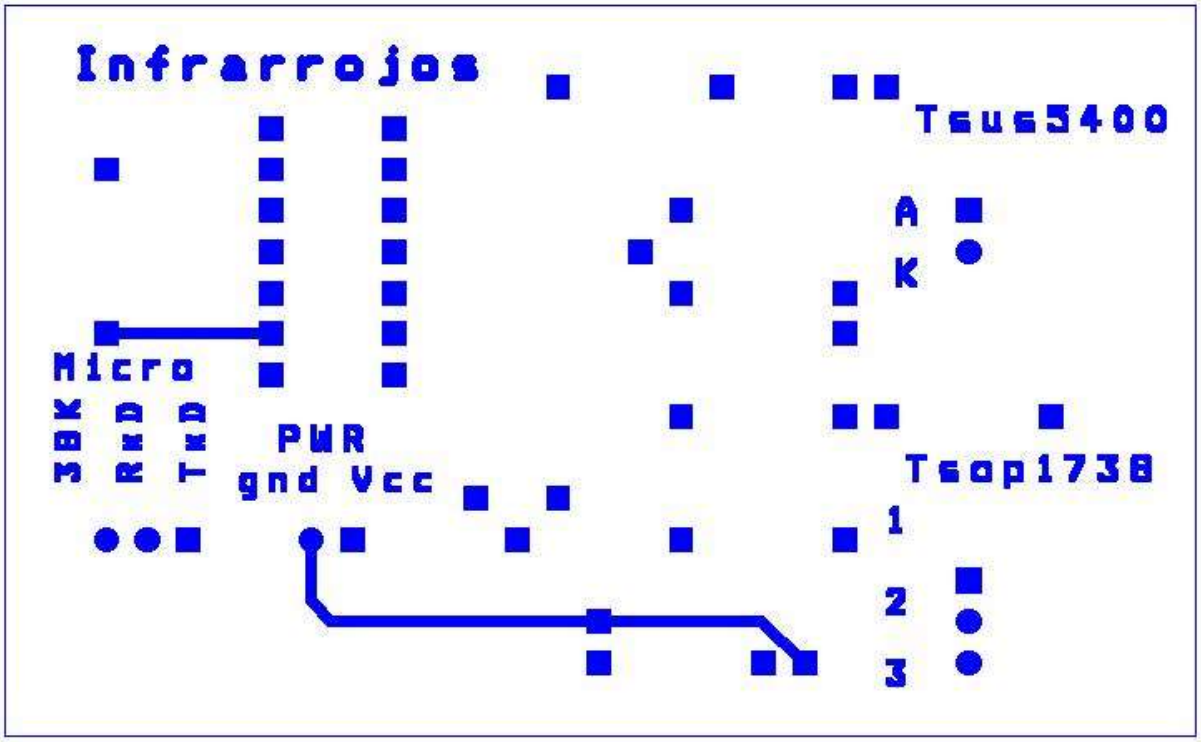
4.2.2. COMUNICACIÓN POR IR

En esta placa se incluyen tanto el emisor como el receptor de infrarrojos con los conectores necesarios para la comunicación con el microcontrolador. La idea principal era conseguir una placa no solo útil para nuestra aplicación sino que pudiera utilizarse en cualquier aplicación donde se necesitara un módulo transreceptor de IR.

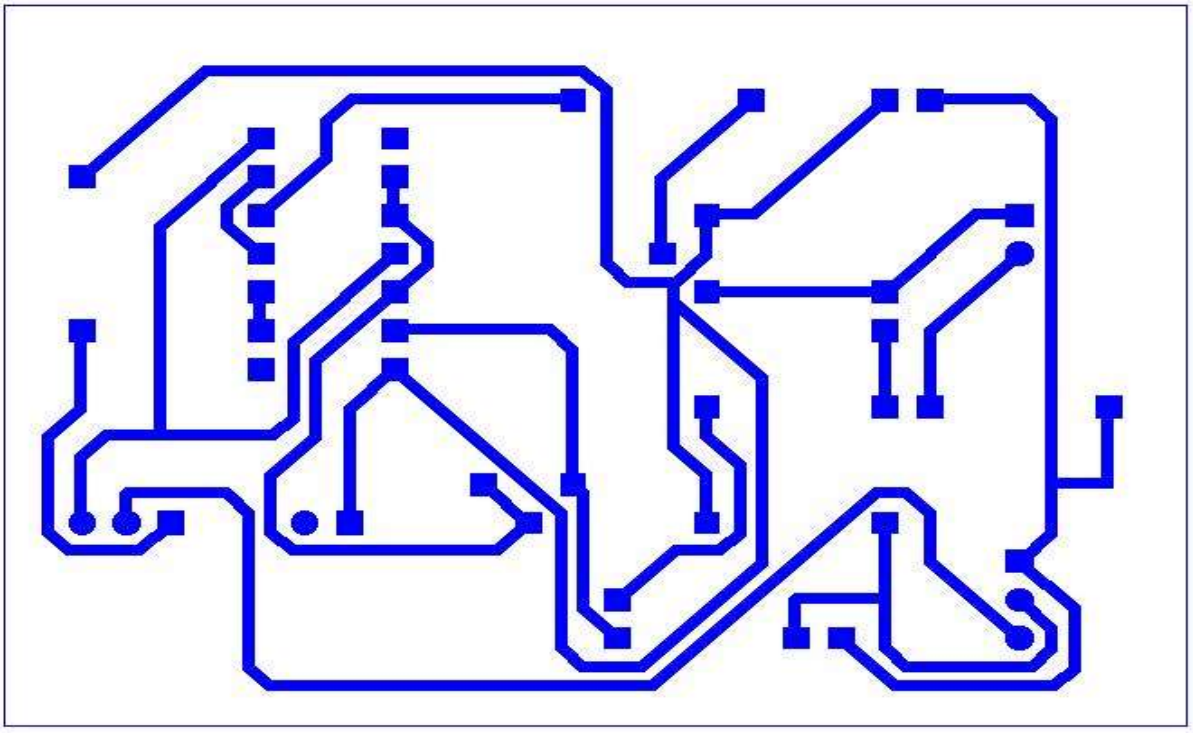
- **Esquemático**



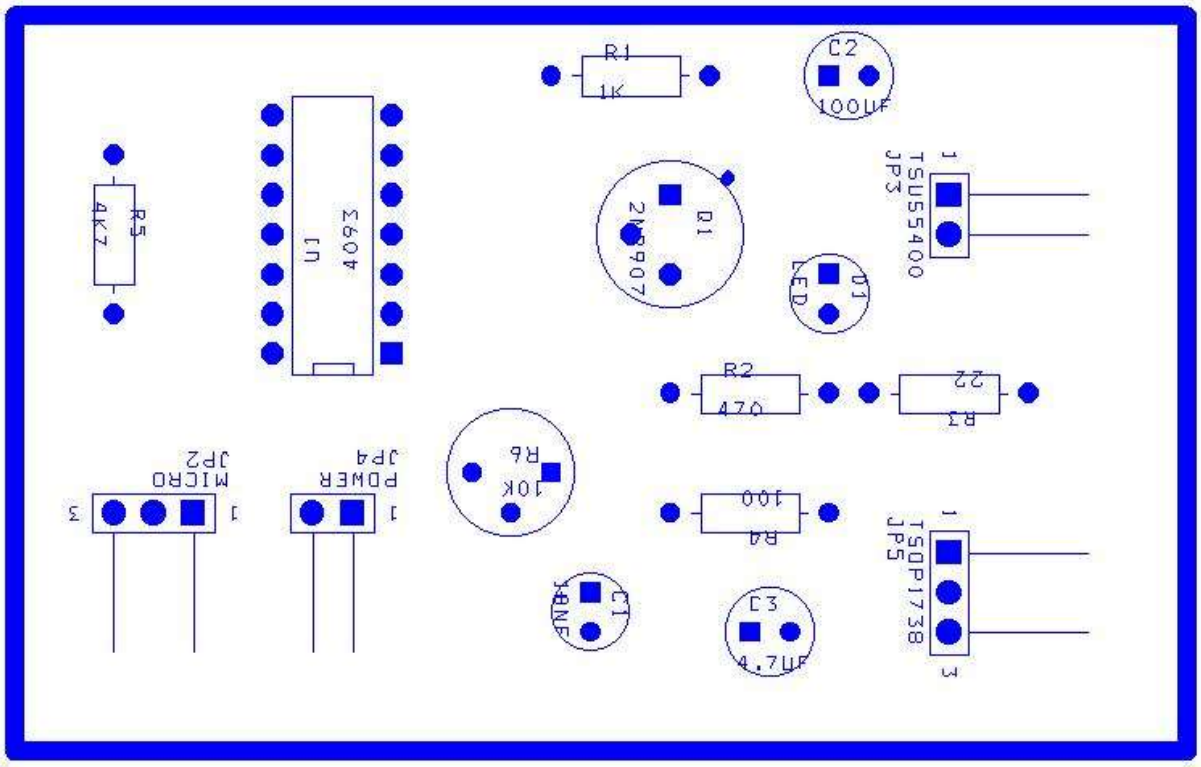
- **Trazado de pistas cara TOP**



- **Trazado de pistas cara BOTTOM**



- **Disposición de componentes**



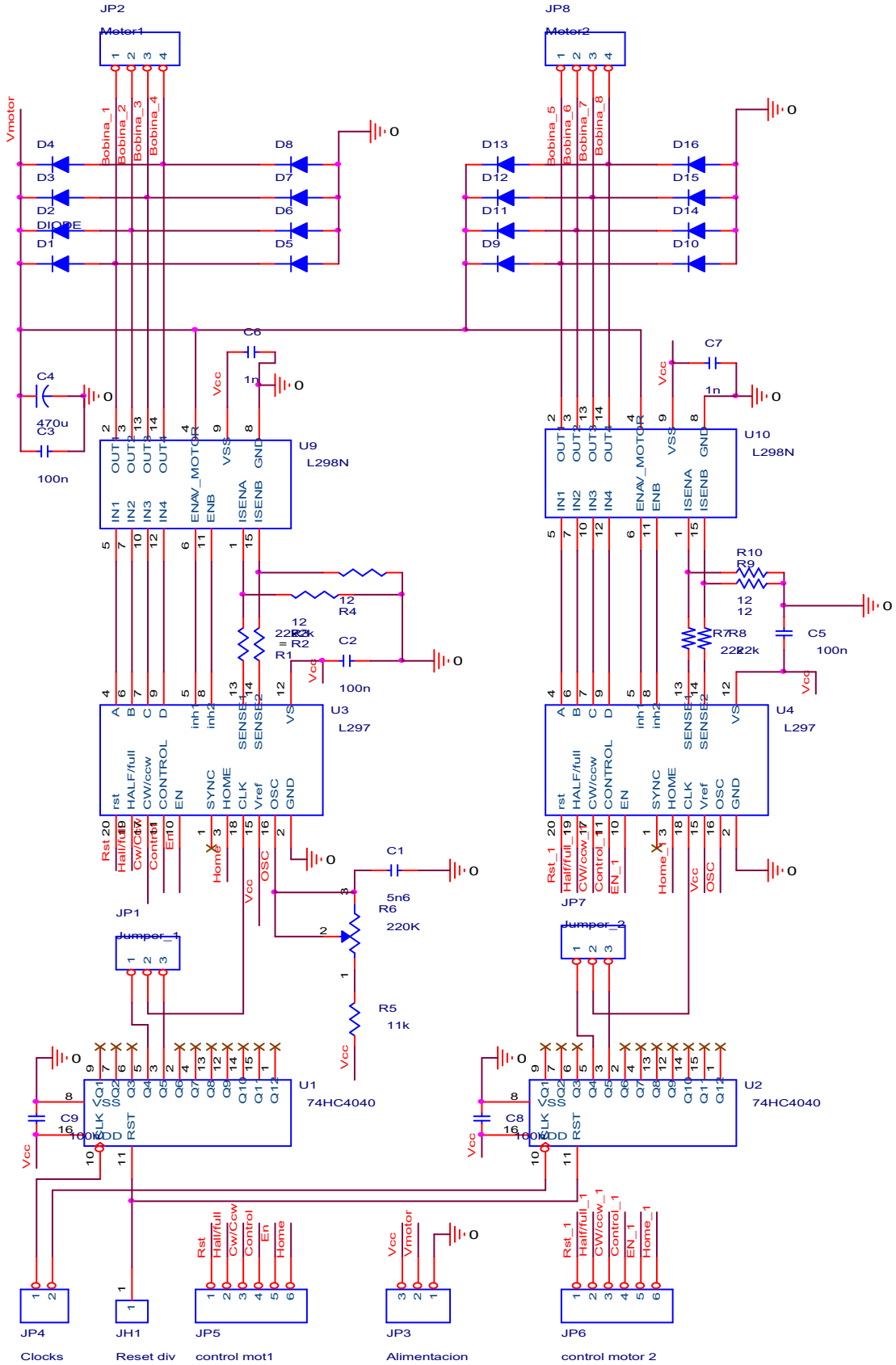
Las pautas para la disposición de los componentes en la placa correspondiente a los infrarrojos han, al igual que en el caso anterior, minimizar en la medida de lo posible el cableado entre esta placa y la del microcontrolador. El resto de los componentes se ha dispuesto para minimizar el trazado de pistas. No se ha puesto un plano de masa ya que las interferencias en los infrarrojos solo son causadas por la luz ambiente y este tipo de interferencias se ha evitado modulando la señal a 38Khz.

4.2.3. PLACA DE POTENCIA (CONTROL DE LOS MOTORES)

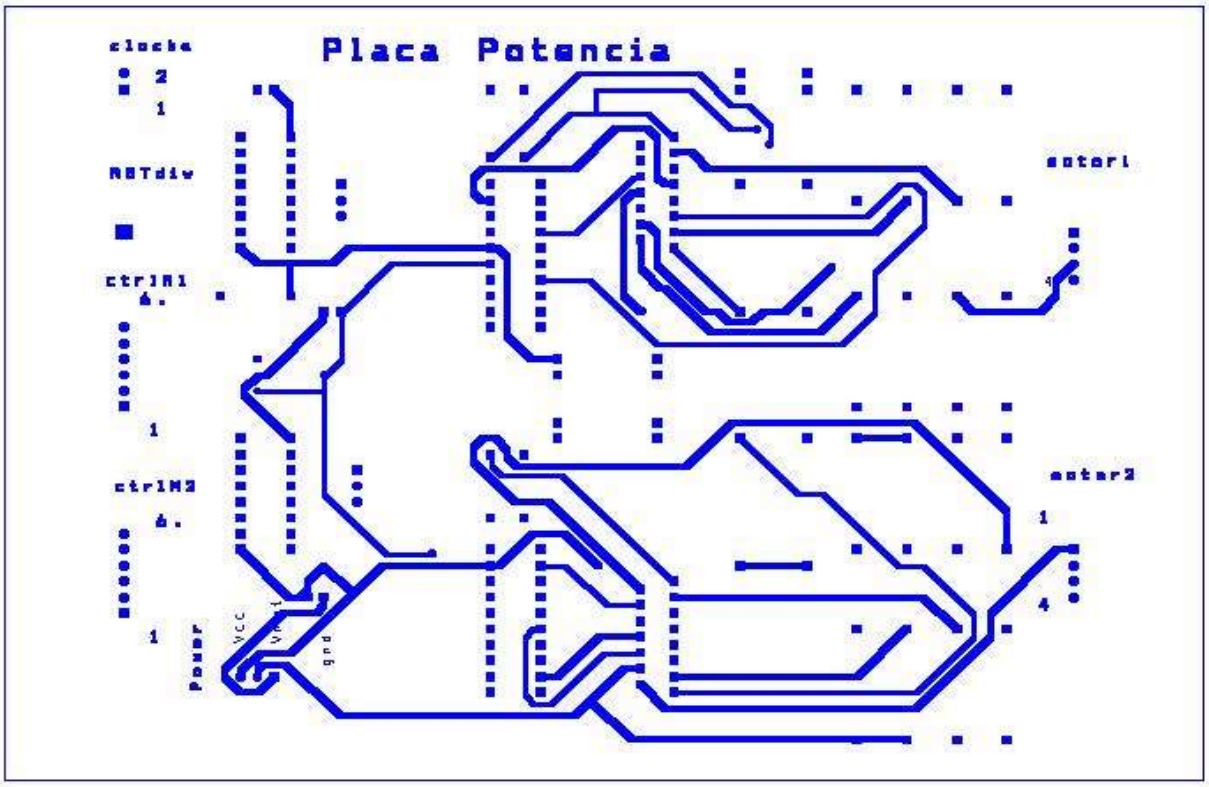
En esta placa se implementan los dispositivos necesarios para el control y suministro de potencia de los motores. Lleva como pre etapa un divisor de frecuencia para dividir la frecuencia proveniente de los timers del microcontrolador y ajustarlo a la máxima posible en nuestra aplicación.

- **Esquemático**

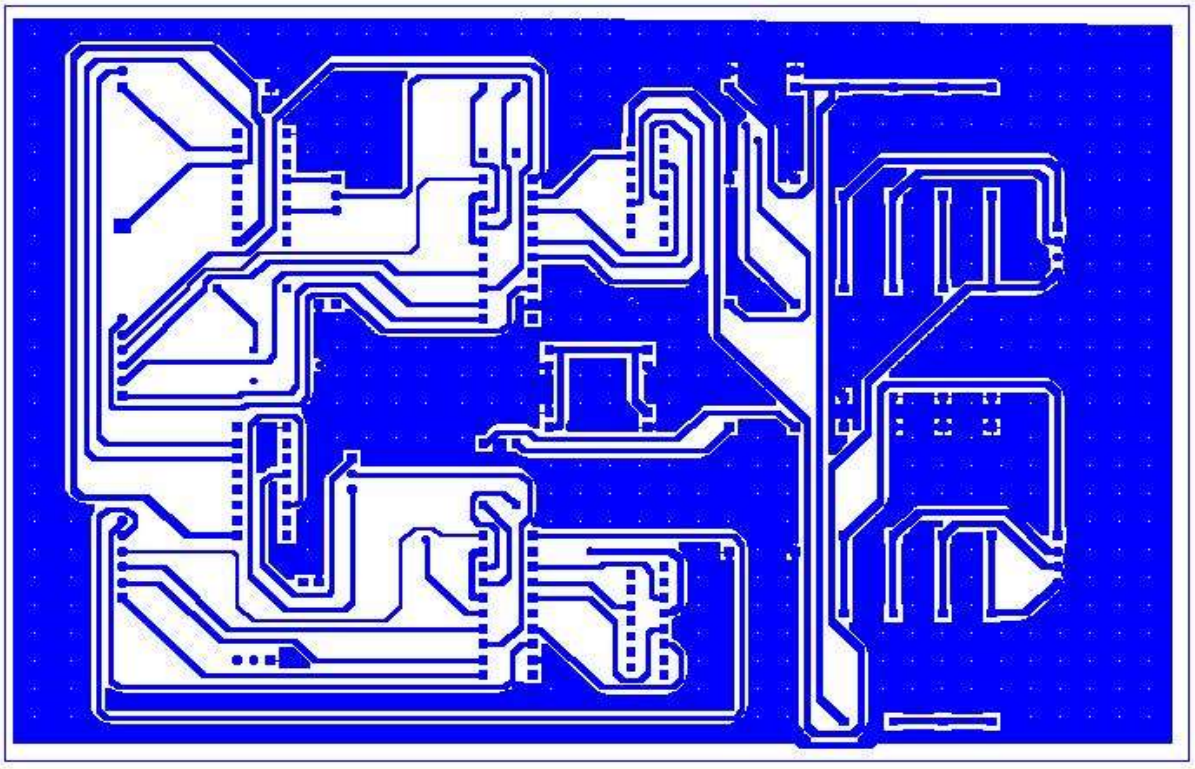
Página siguiente.


JP2
Meter1
JP8
Meter2
Vmotor
D4
D3
D2
DIODE
D1
D8
D7
D6
D5
D13
D12
D11
D9
D16
D15
D14
D10
Bobina_1
Bobina_2
Bobina_3
Bobina_4
Bobina_5
Bobina_6
Bobina_7
Bobina_8
C4
470u
C3
100n
C6
1n
C7
1n
U9
L298N
U10
L298N
R4
12
R1
R2
C2
100n
R10
R9
R7
R8
C5
100n
U3
L297
U4
L297
C1
5n6
R6
220K
R5
11k
JP1
Jumper_1
JP7
Jumper_2
U1
74HC4040
U2
74HC4040
C9
100n
C8
100n
JP4
Clocks
JH1
Reset div
JP5
control mot1
JP3
Alimentacion
JP6
control motor 2

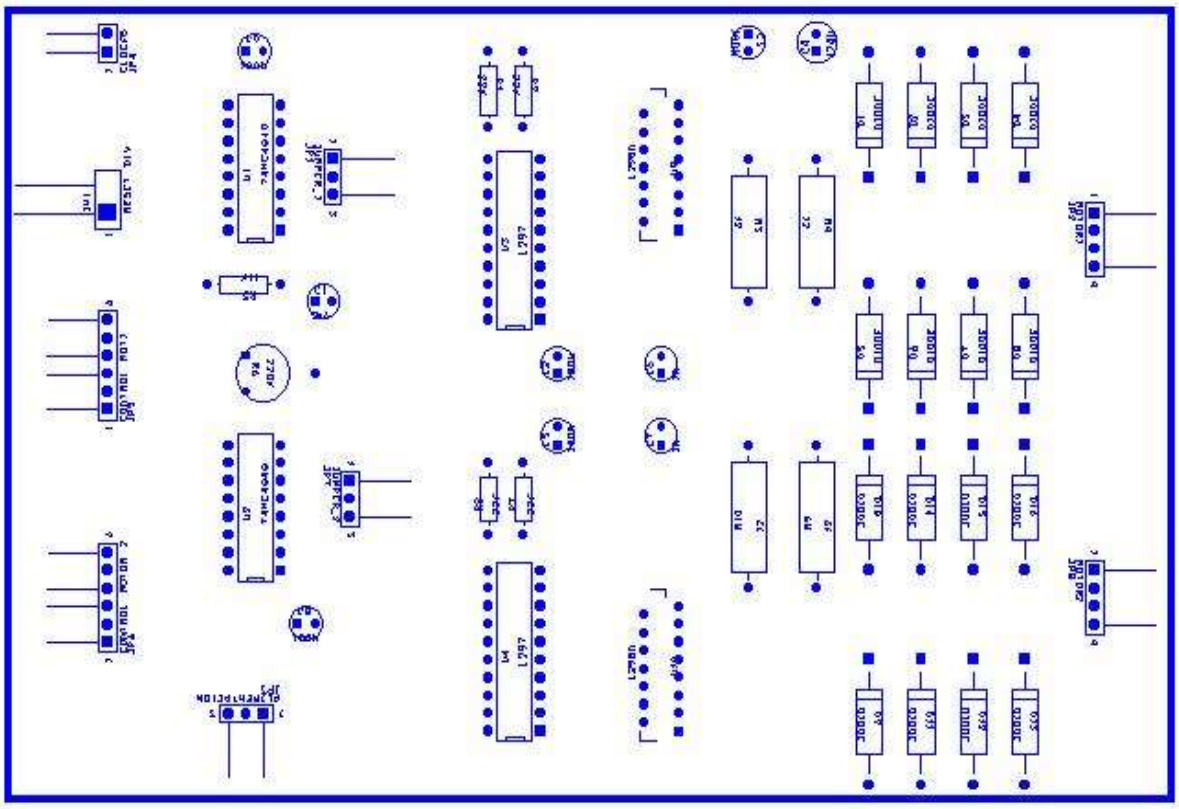
- **Trazado de pistas cara TOP**



- **Trazado de pistas cara BOTTOM**



- **Disposición de componentes**



La disposición de los componentes se ha elegido de forma que los conectores para el control de los motores sean accesibles para el microcontrolador y los conectores de los motores sean accesibles para los mismos con el consecuente ahorro de cableado. En la cara BOTTOM se ha creado un plano de masa que ayudara a la disipación de calor del L298n ya que el radiador está conectado a masa y de esta forma aumenta la disipación por el plano de masa.

A su vez la disposición de los componentes, como se puede observar, es simétrica para una fácil identificación de las etapas de cada uno de los motores.

4.3. PLANOS DEL MANDO

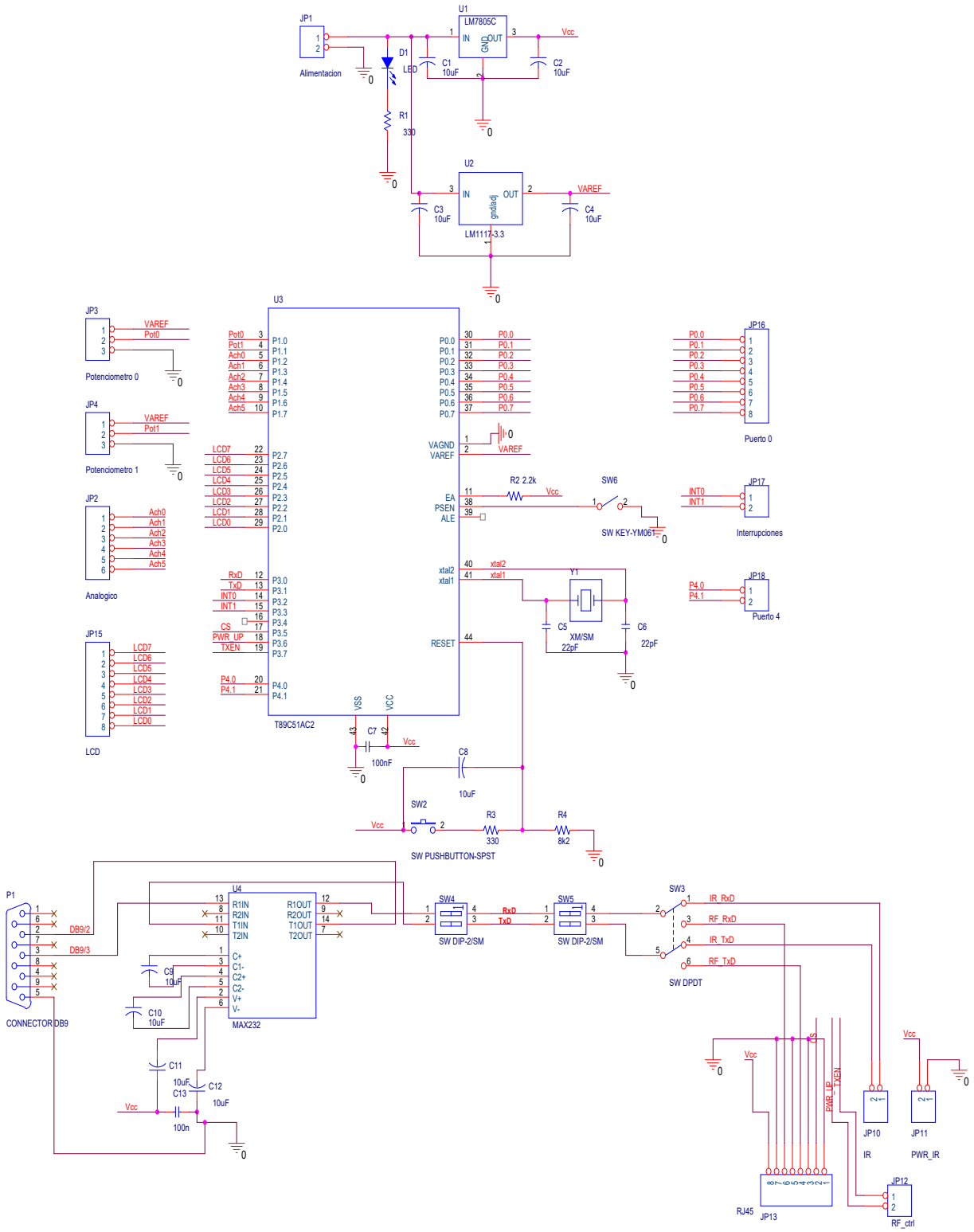
A continuación se expondrán cada una de las distintas placas que componen el mando, mostrando el esquemático, el trazado de pistas y la disposición de componentes.

4.3.1. PLACA CENTRAL (MICROCONTROLADOR)

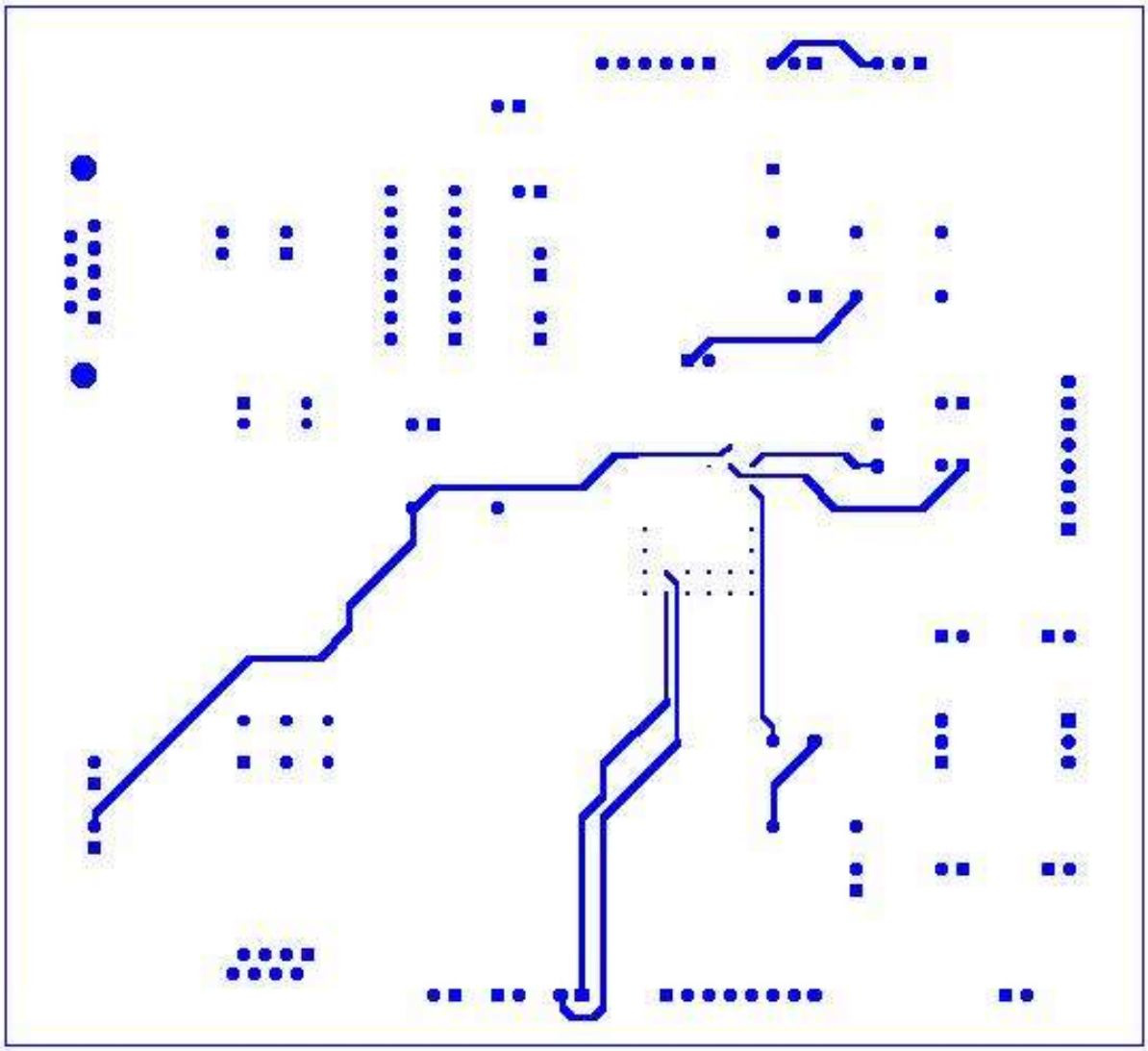
En esta placa se encuentra el "cerebro" del mando, el microcontrolador. Así pues este dispone del programador, con la comunicación serie necesaria para esta función, alimentación, circuito de reseteado y oscilador. La placa, a su vez, dispone de puertos de control para la interface de usuario y comunicaciones, así como los interruptores que permitirán cambiar de un tipo de comunicación a otra.

- **Esquemático**

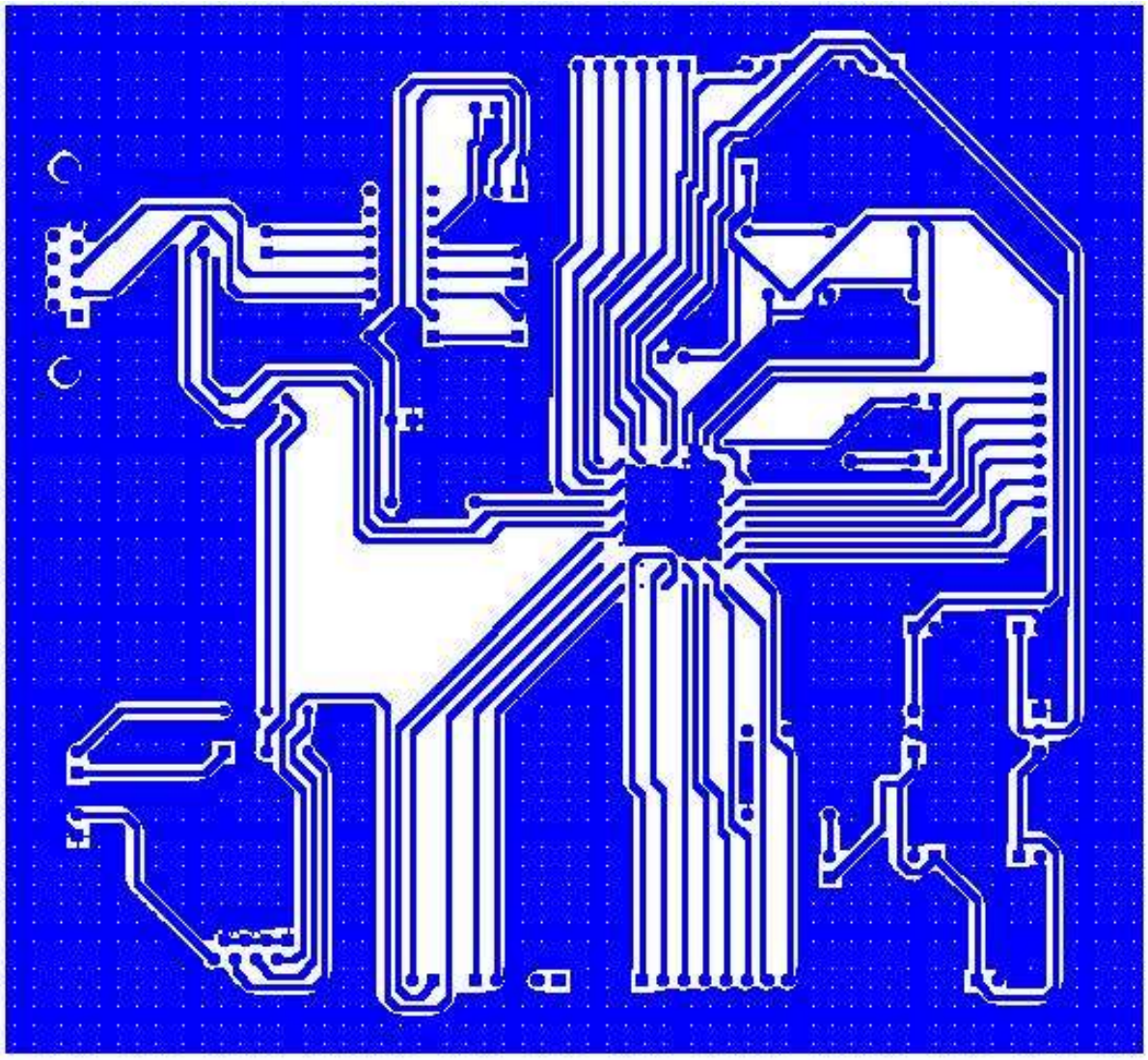
Página siguiente.


JP1
Alimentacion
U1
LM7805C
IN
GND
OUT
Vcc
D1
LED
C1
10uF
C2
10uF
R1
330
U2
LM1117-3.3
VAREF
C3
10uF
C4
10uF
U3
T89C51AC2
JP3
Potenciometro 0
JP4
Potenciometro 1
JP2
Analogico
JP15
LCD
JP16
Puerto 0
JP17
Interrupciones
JP18
Puerto 4
R2 2.2k
SW6
SW KEY-YM061
Y1
XM/SM
22pF
C5
C6
C7
100nF
C8
10uF
SW2
SW PUSHBUTTON-SPST
R3
330
R4
8k2
P1
CONNECTOR DB9
U4
MAX232
C9
10uF
C10
10uF
C11
10uF
C12
10uF
C13
100n
SW4
SW5
SW DIP-2/SM
SW3
SW DPDT
IR_RxD
RF_RxD
IR_TxD
RF_TxD
JP10
IR
JP11
PWR_IR
JP12
RF_ctrl
JP13
RJ45

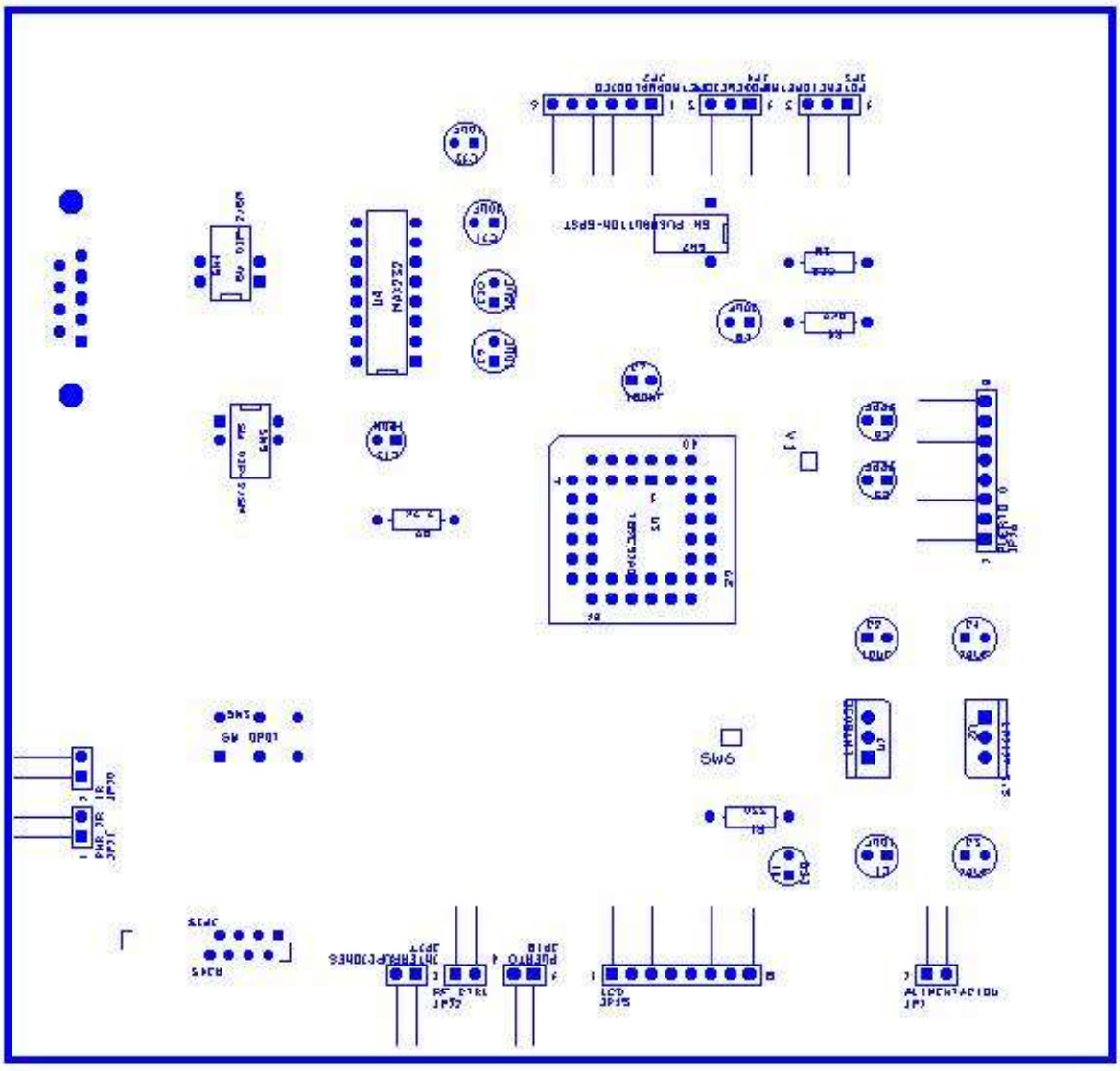
- **Trazado de pistas cara TOP**



- **Trazado de pistas cara BOTTOM**



- **Disposición de componentes**

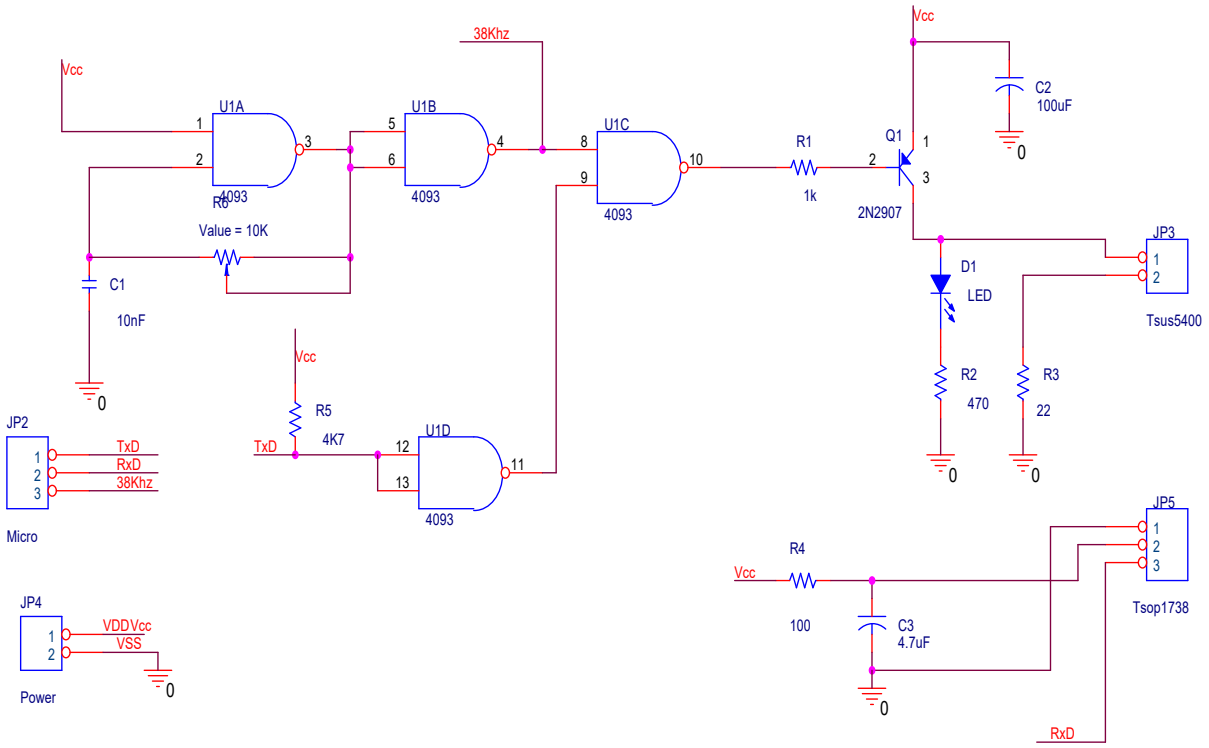


Para la disposición de los componentes se han tenido varios factores en cuenta. El primero ha sido la propia potencia del programa ya que para conseguir un trazado de pistas del 100% no es válida cualquier distribución de componentes. Otro factor tenido en cuenta son nuestras propias necesidades, ya que para una fácil conexión entre las placas, es necesaria una sencilla accesibilidad a los puertos. Por ello se han colocado los conectores en la periferia de la placa, pero con un orden preestablecido, para que el cableado necesario para conectarla con el resto de placas que componen el vehículo no fuera demasiado extenso evitando líos de cables.

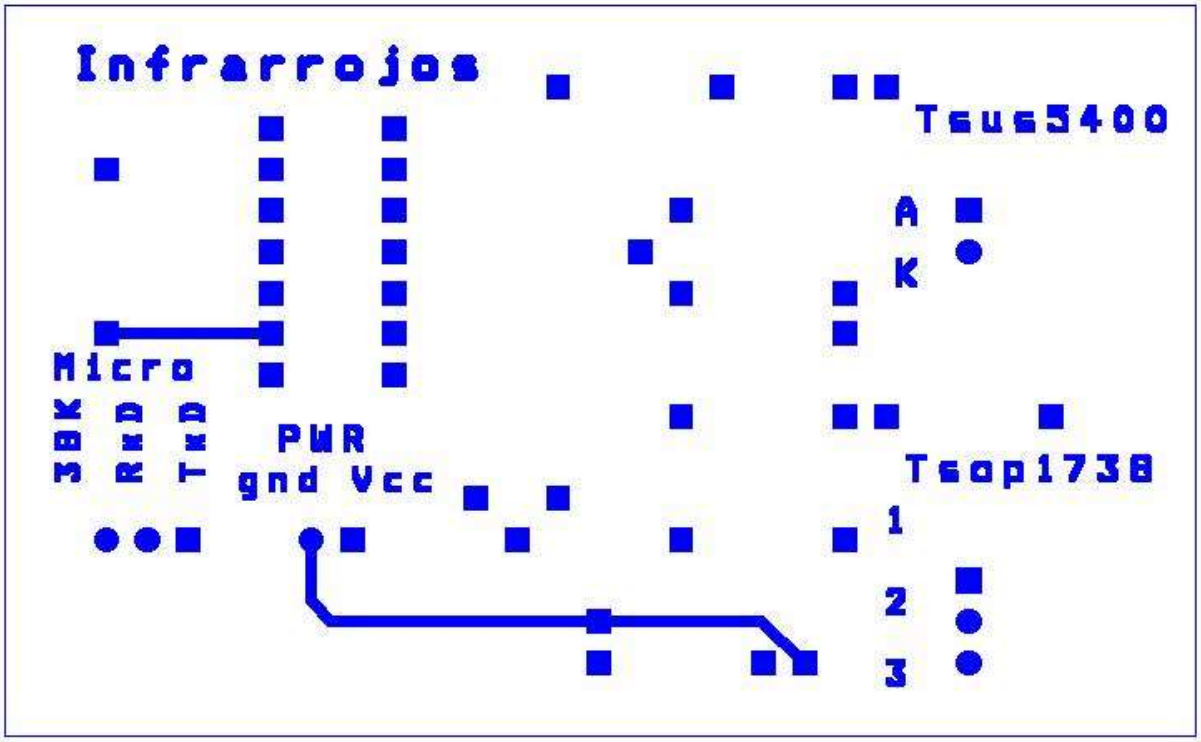
4.3.2. COMUNICACIÓN POR IR

En esta placa se incluyen tanto el emisor como el receptor de infrarrojos con los conectores necesarios para la comunicación con el microcontrolador. La idea principal era conseguir una placa no solo útil para nuestra aplicación sino que pudiera utilizarse en cualquier aplicación donde se necesitara un módulo transreceptor de IR.

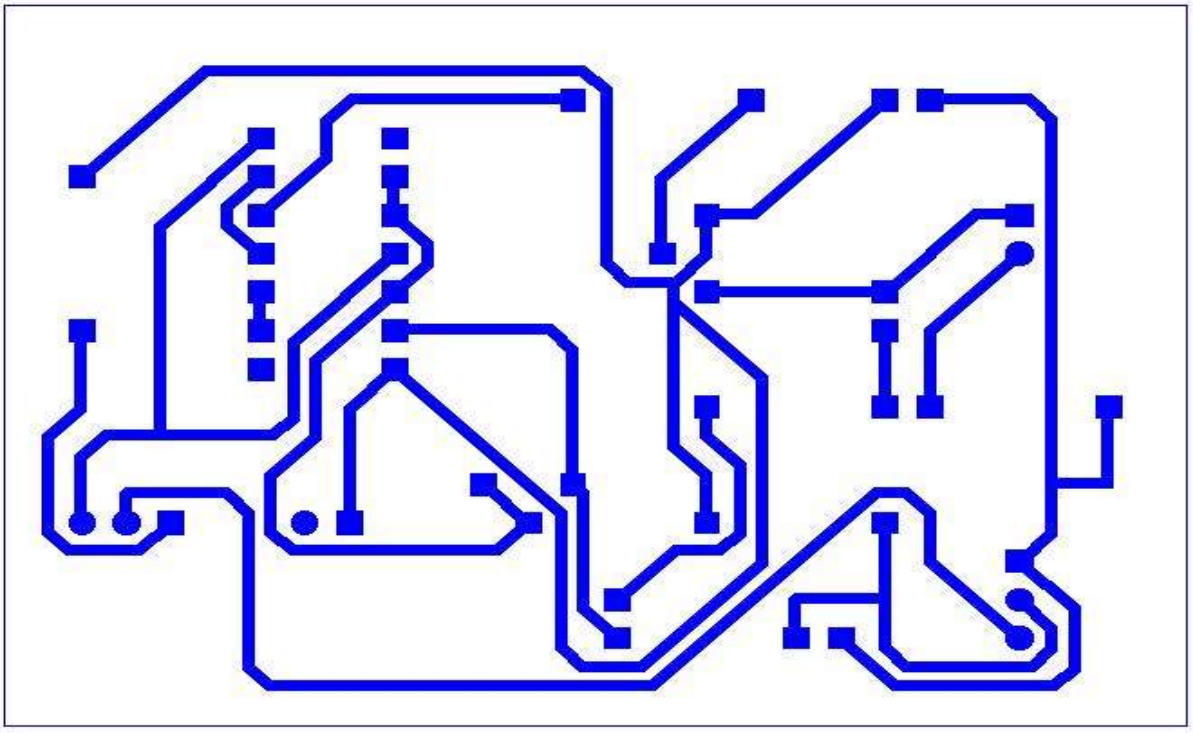
- **Esquemático**



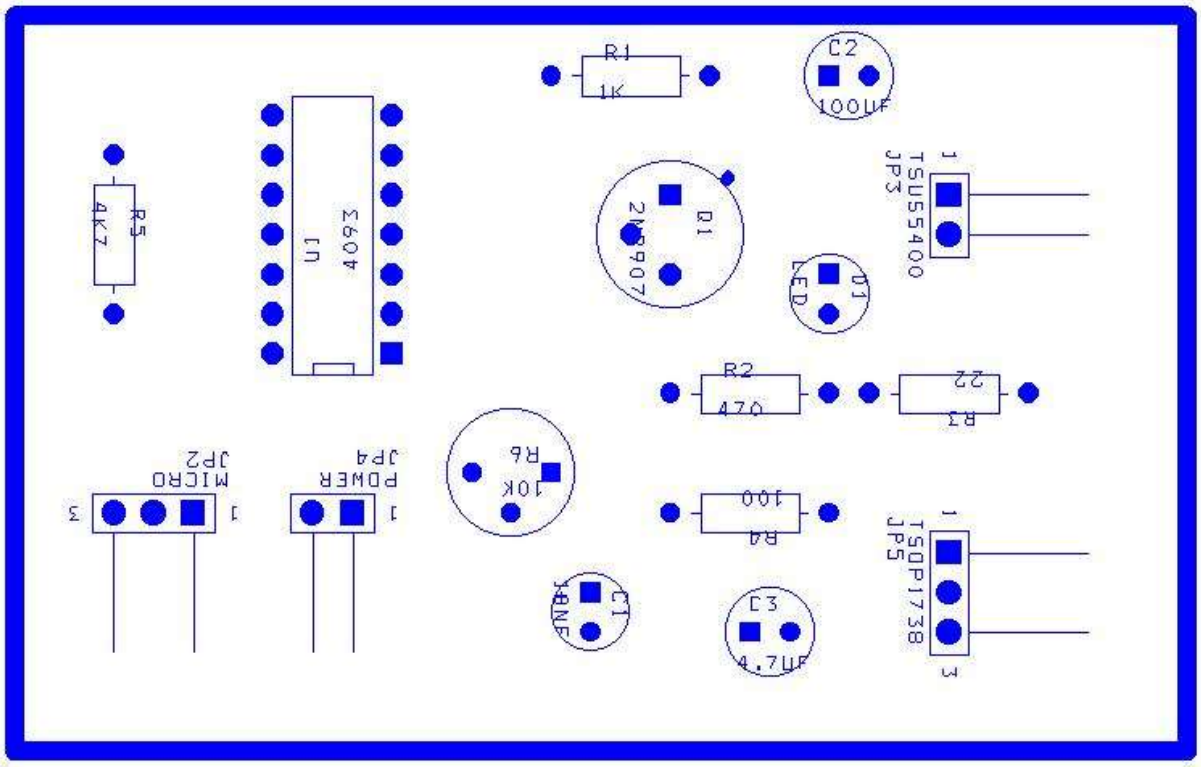
- **Trazado de pistas cara TOP**



- **Trazado de pistas cara BOTTOM**



- **Disposición de componentes**



Las pautas para la disposición de los componentes en la placa correspondiente a los infrarrojos han, al igual que en el caso anterior, minimizar en la medida de lo posible el cableado entre esta placa y la del microcontrolador. El resto de los componentes se ha dispuesto para minimizar el trazado de pistas. No se ha puesto un plano de masa ya que las interferencias en los infrarrojos solo son causadas por la luz ambiente y este tipo de interferencias se ha evitado modulando la señal a 38Khz.

5. PRESUPUESTO Y PLIEGO

5.1. PRESUPUESTO DE LA MAQUETA-PROTOTIPO

5.1.1. PRESUPUESTO DEL MANDO (MICROCONTROLADOR)

REF	Nº UNID	DESCRIPCION	DETALLE	PRECIO (€)
1	1	U3	T89C51AC2	7.46
2	1	P1	Conector DB9	1.14
3	1	U4	Max 232	0.66
4	9	C1..C4, C8..C12	Condensadores 10Uf	0.45
5	2	C7, C13	Condensadores 100Nf	0.24
6	2	SW4, SW5	Microinterruptores	0.53
7	1	SW3	Conmutador 2 líneas	0.87
8	1	SW2	Pulsador	0.06
9	1	U1	LM7805	0.92
10	1	U2	LM1117-3.3	1
11	1	D1	Led	0.03
12	1	JP13	Conector RJ45	1.13
13	1	SW6	Interruptor	0.36
14	2	C5, C6	Condensadores 22Pf	0.08
15	1	Y1	Cristal 11.0952Mhz	0.17
16	2	R1, R3	Resistencias 330	0.06
17	1	R2	Resistencia 2k2	0.03
18	1	R4	Resistencia 8k2	0.03
19	1	Zócalo CI	PLCC 44	0.42
20	1	Zócalo CI	16 pines	0.25
21	6	JP1,JP10..12,JP17, JP18	Conectores de 2 pines	0.54
22	2	JP15, JP16	Conectores 8 pines	0.96
23	1		Placa FV 2 caras	3.25
24	1		LCD	9.25
TOTAL COMPONENTES MANDO				

5.1.2. PRESUPUESTO DEL VEHÍCULO (MICROCONTROLADOR)

REF	Nº UNID	DESCRIPCION	DETALLE	PRECIO (€)
1	1	U3	T89C51AC2	7.46
2	1	P1	Conector DB9	1.14
3	1	U4	Max 232	0.66
4	9	C1..C4, C8..C12	Condensadores 10uF	0.45
5	2	C7, C13	Condensadores 100nF	0.24
6	2	SW4, SW5	Microinterruptores	0.53
7	1	SW3	Conmutador 2 líneas	0.87
8	1	SW2	Pulsador	0.06
9	1	U1	LM7805	0.92
10	1	U2	LM1117-3.3	1
11	1	D1	Led	0.03
12	1	JP13	Conector RJ45	1.13
13	1	SW6	Interruptor	0.36
14	2	C5, C6	Condensadores 22pF	0.08
15	1	Y1	Cristal 11.0952Mhz	0.17
16	2	R1, R3	Resistencias 330	0.06
17	1	R2	Resistencia 2k2	0.03
18	1	R4	Resistencia 8k2	0.03
19	1	Zócalo CI	PLCC 44	0.42
20	1	Zócalo CI	16 pines	0.25
21	8	JP1,JP10..12,JP17, JP18	Conectores de 2 pines	0.65
22	3	JP15, JP16	Conectores 8 pines	1.34
23	1		Placa FV 2 Caras	3.25
TOTAL COMPONENTES VEHÍCULO				

5.1.3. PRESUPUESTO DE LA PLACA DE POTENCIA

REF	Nº UNID	DESCRIPCION	DETALLE	PRECIO (€)
1	2	JP2, JP8	Conectores 4 pines	0.30
2	1	JP3	Conector 3 pines	0.16
3	2	JP5, JP6	Conectores 6 pines	0.13
4	1	JP4	Conector 2 pines	0.08
5	2	JP1, JP7	Jumpers	0.04
6	16	D1..D16	Diodos 1N4007	0.16
7	4	R3, R4, R9, R10	Resistencias 5.6 2W	0.86
8	2	U9, U10	L298n	7.84
9	2	U3, U4	L297	6.15
10	1	C4	Condensador 470uF	0.06
11	5	C2, C3, C5, C8, C9	Condensadores 100nF	0.30
12	2	C6, C7	Condensadores 1nF	0.12
13	1	C1	Condensador 5.6nF	0.06
14	4	R1, R2, R7, R8	Resistencias 22k	0.12
15	2	U1, U2	74HC4040	0.44
16	1	R6	Potenciómetro 220k	0.10
17	1	R5	Resistencia 11k	0.06
18	1		Placa FV 2 Caras	3.25
TOTAL COMPONENTES POTENCIA				

5.1.4. PRESUPUESTO DE LA PLACAS DE IR

REF	Nº UNID	DESCRIPCION	DETALLE	PRECIO (€)
1	2	U1	4093	3.70
2	4	JP2, JP5	Conectores 3 pines	0.77
3	4	JP1, JP3	Conectores 2 pines	0.32
4	2	Q1	Transistor 2N2907	0.30
5	2	R5	Resistencia 4k7	0.20
6	2	R1	Resistencia 1k	0.22
7	2	R3	Resistencia 22	0.21
8	2	R2	Resistencia 470	0.23
9	2	R4	Resistencia 100	0.20
10	2	C1	Condensador 10nF	0.21
11	2	C2	Condensador 100uf	0.20
12	2	C3	Condenador 4.7uf	0.22
13	2	D1	Led	0.03
14	2	JP3	IRED Tsus 5400	1.74
15	2	JP5	Tsop 1738	2.36
TOTAL COMPONENTES IR				

5.1.5. PRESUPUESTO DEL CHASIS DEL ROBOT

REF	Nº UNID	DESCRIPCION	DETALLE	PRECIO (€)
1	1	CHASIS	ALUMINIO	20
2	2	REDUCTORAS		10
3	2	MOTORES PASO A PASO	CROUZET	105.34
		TOTAL CHASIS		***135.34***

5.1.6. MANO DE OBRA

Al total del presupuesto hay que añadir el total de horas invertidas en la programación del robot, inicialización y verificación de la correcta compilación del programa, además del tiempo invertido en montaje del chasis.

REF	TAREA	Nº HORAS	PRECIO (€)
63	MONTAJE	2	30
64	PROGRAMACION	3	45
65	PRUEBAS	2	30
	TOTAL MANO DE OBRA		***105***

TOTALES PARCIALES	PRECIO (€)
COMPONENTES	219.01
MANO DE OBRA	105
PRESUPUESTO	324.01
TOTAL PRESUPUESTO + IVA (16%)	***375.85***

5.2. PLIEGO DE CONDICIONES

EXTENSION DE LA GARANTIA LIMITADA

Se garantiza al cliente que los productos utilizados estarán libres de defectos de fabricación y de mano de obra durante un tiempo específico después de la fecha en que fue adquirido por el cliente. La duración de la garantía será de 24 meses.

La Garantía limitada cubre únicamente aquellos defectos que surgiesen como resultado del uso normal del robot y no por aquellos resultantes de:

Mantenimiento inapropiado o inadecuado
Modificaciones no autorizadas
Operación fuera de las especificaciones ambientales definidas por el fabricante

Si durante el periodo aplicable a esta garantía se recibiese notificación del defecto del producto se reserva la opción de reparar o reemplazar el componente o modulo. Cualquier producto de reemplazo será nuevo, o como nuevo, siempre que su funcionalidad sea por lo menos igual a la del producto reemplazado.

Si el fabricante no pudiera reparar o reemplazar el componente o modulo defectuoso cubierto por esta garantía, se le reembolsará el importe del producto dentro de un tiempo razonable posterior a la notificación del defecto, una vez que el cliente devuelva el robot.

En todas las reparaciones se deberán acompañar al robot, la factura de compra y la garantía debidamente cumplimentada con la indicación exacta de la fecha de venta del robot.

Dado que solo el fabricante esta cualificado y autorizado para manipular el robot, ésta garantía quedaría sin efecto alguno en el caso de que otra persona ajena lo manipulara.

Si la intervención se efectuara en la empresa del usuario, todos los gastos de desplazamiento del fabricante para proceder al examen y/o reparación del equipo, correrán por cuenta del cliente, de acuerdo con las tarifas vigentes.

Esta declaración de garantía limitada otorga al cliente derechos legales específicos. El cliente podría tener otros derechos que varían de un país a otro.

CLAUSULA DE EXONERACION DE RESPONSABILIDAD

Exceptuando las obligaciones especificadas en la declaración de garantía, bajo ninguna circunstancia el fabricante será responsable por daños directos, indirectos, especiales causales o consecuencia, ya sea que estos estén basados en contratos, actos dañinos, o cualquier otra teoría legal y haya sido advertido de la posibilidad de ellos.

CONDICIONES GENERALES

DE PRECIOS: En el caso de que el precio del componente/modulo del robot haya subido debido al incremento anual de precios, cuando se deba reemplazar algún componente, ese incremento no se tendrá en cuenta.

DE FORMAS DE PAGO: El pago del robot se podrá realizar en metálico o también mediante cheque, tarjeta o Paypal.

DE INSPECCIONES Y ENSAYOS: Los dos primeros años de uso debido a la cobertura de la garantía, el fabricante se compromete a realizar las inspecciones y ajustes necesarios para el correcto funcionamiento del robot

DE TRANSPORTE: Si por cualquier fallo por el método de transporte empleado para hacer llegar el robot al cliente éste sufriera algún tipo de deterioro, se le reemplazará el robot, componente o modulo por otra copia exacta en un tiempo razonable.

DE GRAFICOS Y PLANOS: Si los planos prestados sufriesen algún tipo de deterioro debido al paso del tiempo, o alguna causa ajena a la responsabilidad del cliente, el fabricante se compromete a entregar una copia exacta de los planos y gráficos que hayan sido destruidos.

DE PROTECCIONES: El robot no está protegido contra agresiones ambientales, como pueden ser, caídas, temperaturas extremas, ni contra agresiones de agua.

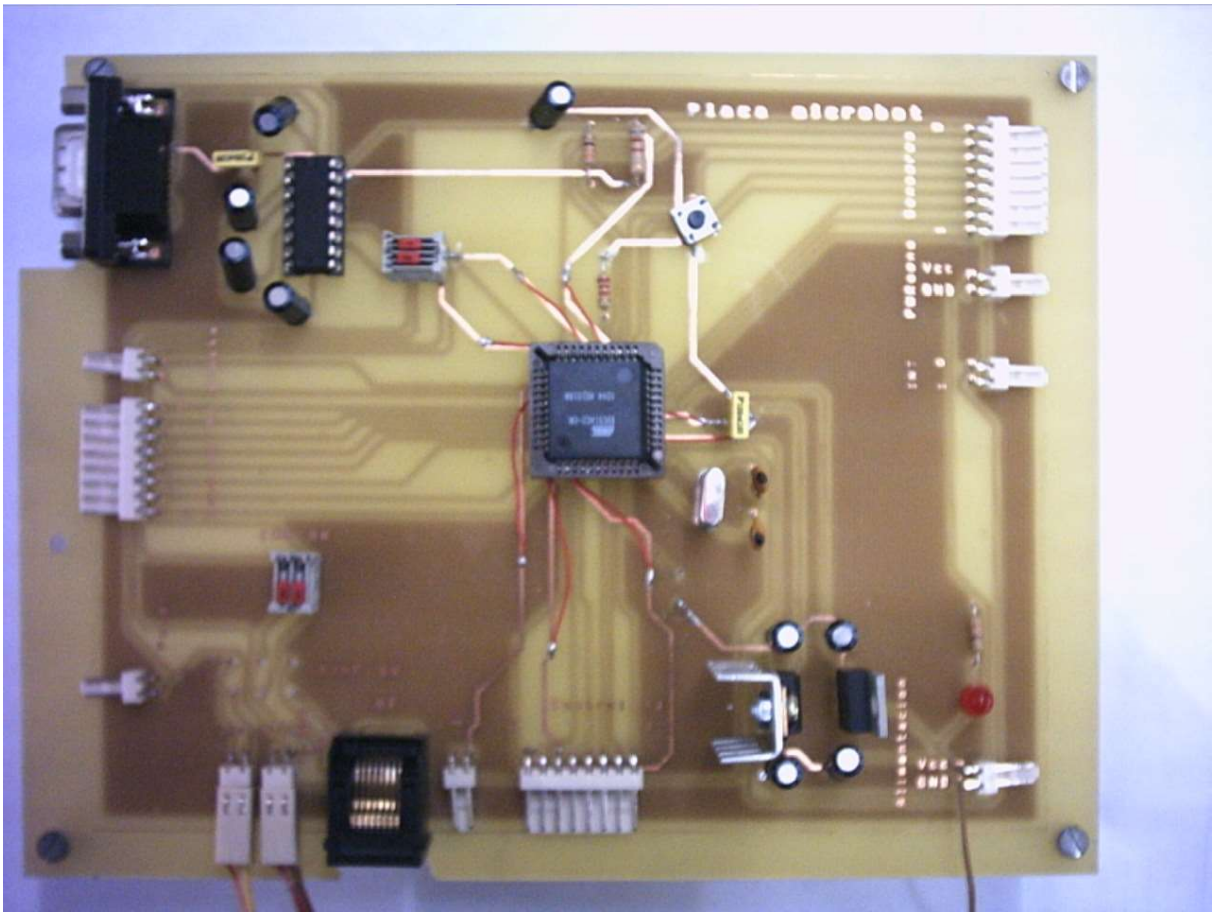
6. RESULTADOS Y AMPLIACIONES

6.1. FOTOS

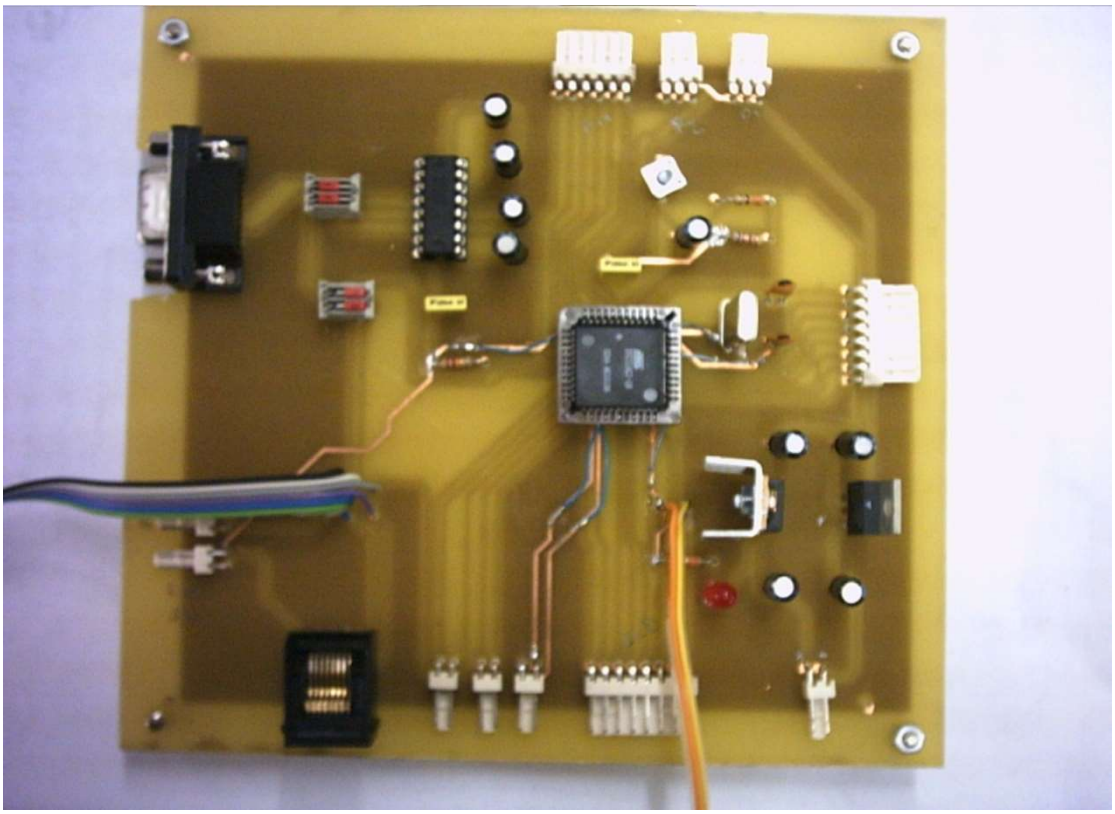
A Continuación se presentan una serie de fotos que muestran la estructura interna del robot. Cabe destacar que nuestro proyecto es una maqueta-prototipo del diseño final, por ello sus dimensiones son reducidas y se ha tomado especial énfasis en la programación y no tanto en su diseño.

Las siguientes fotos muestran cada una de las placas que componen el proyecto:

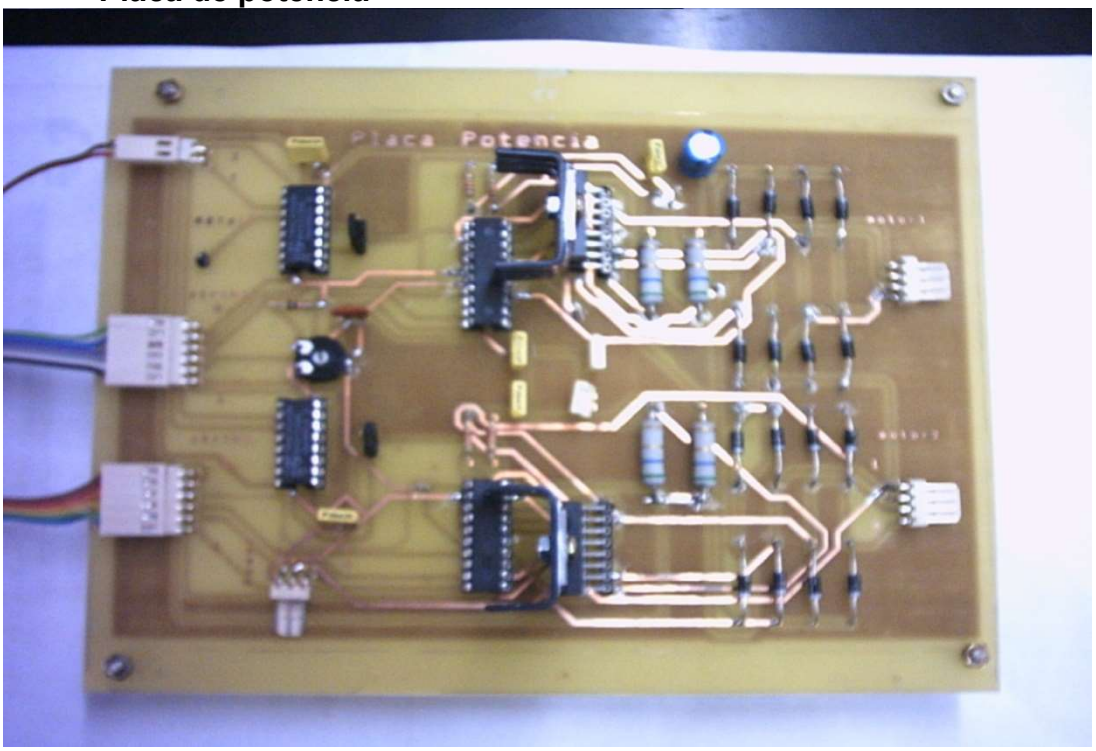
- **Placa vehículo(micro):**



- **Placa mando(micro)**

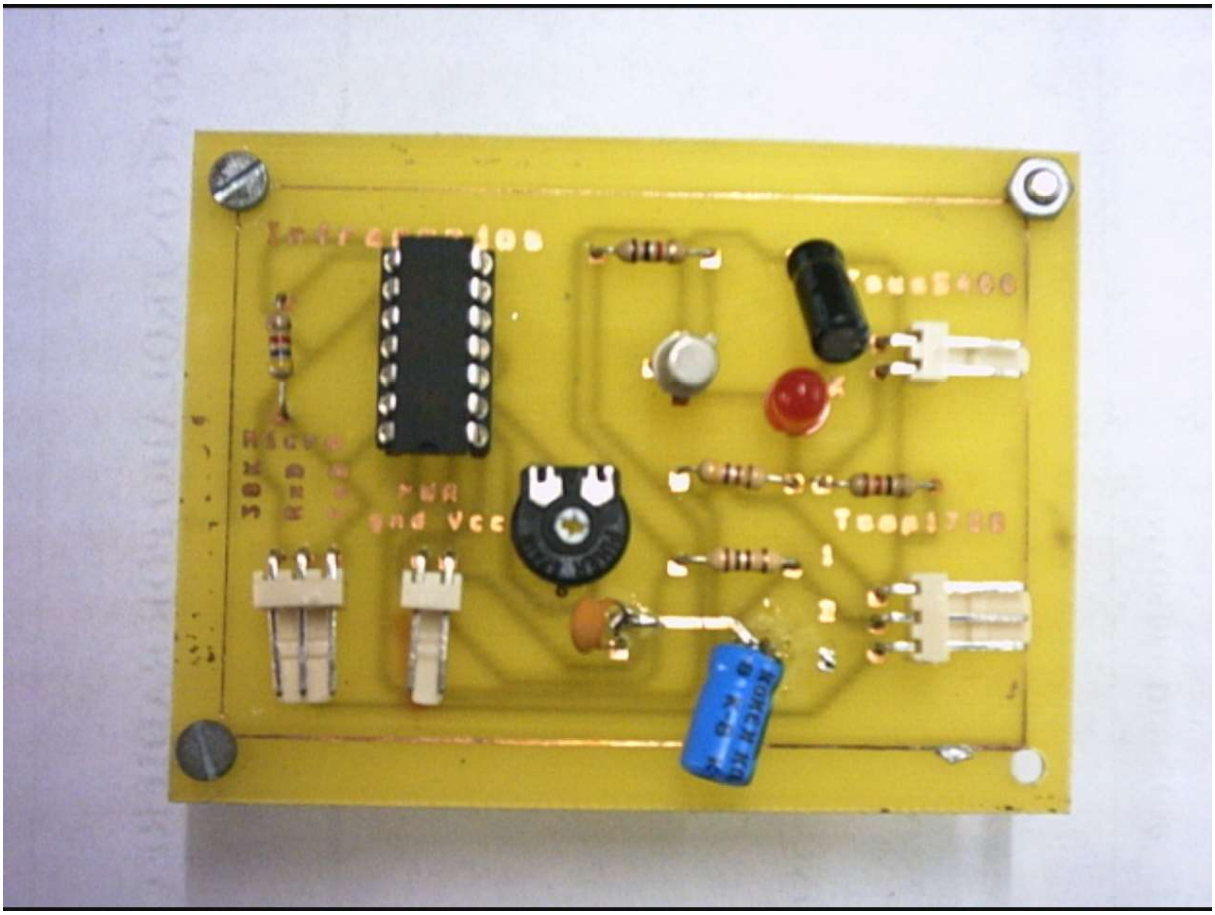


- **Placa de potencia**

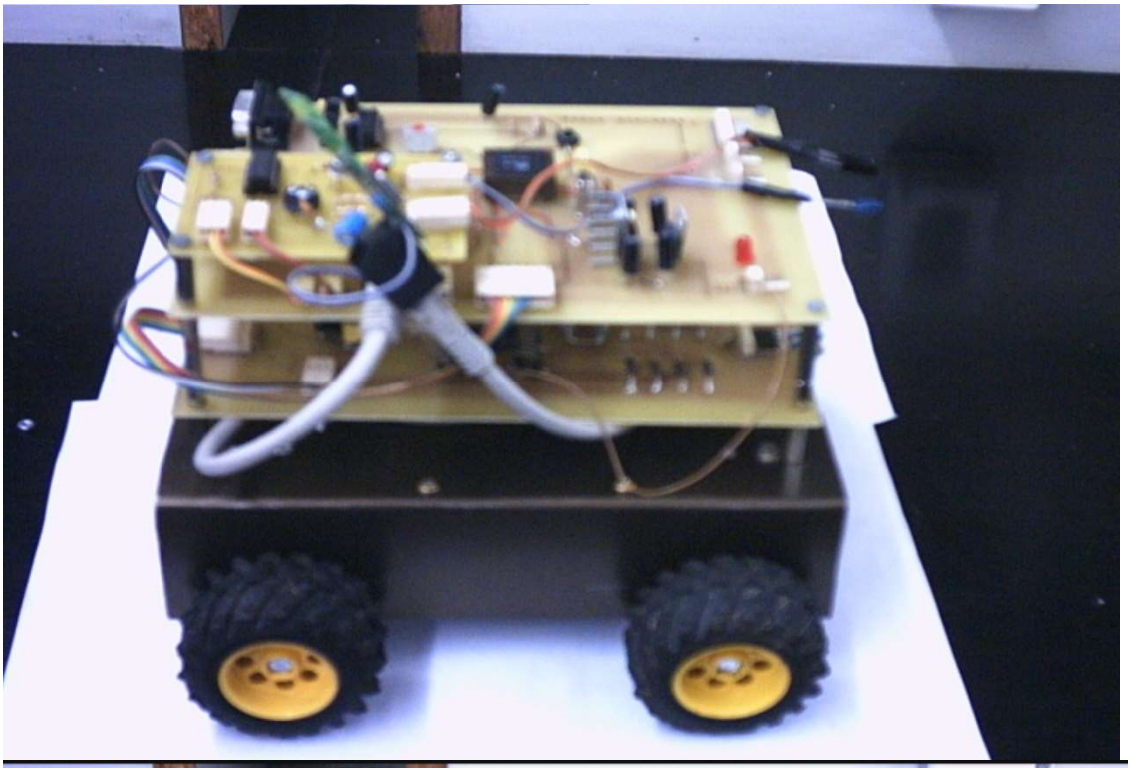


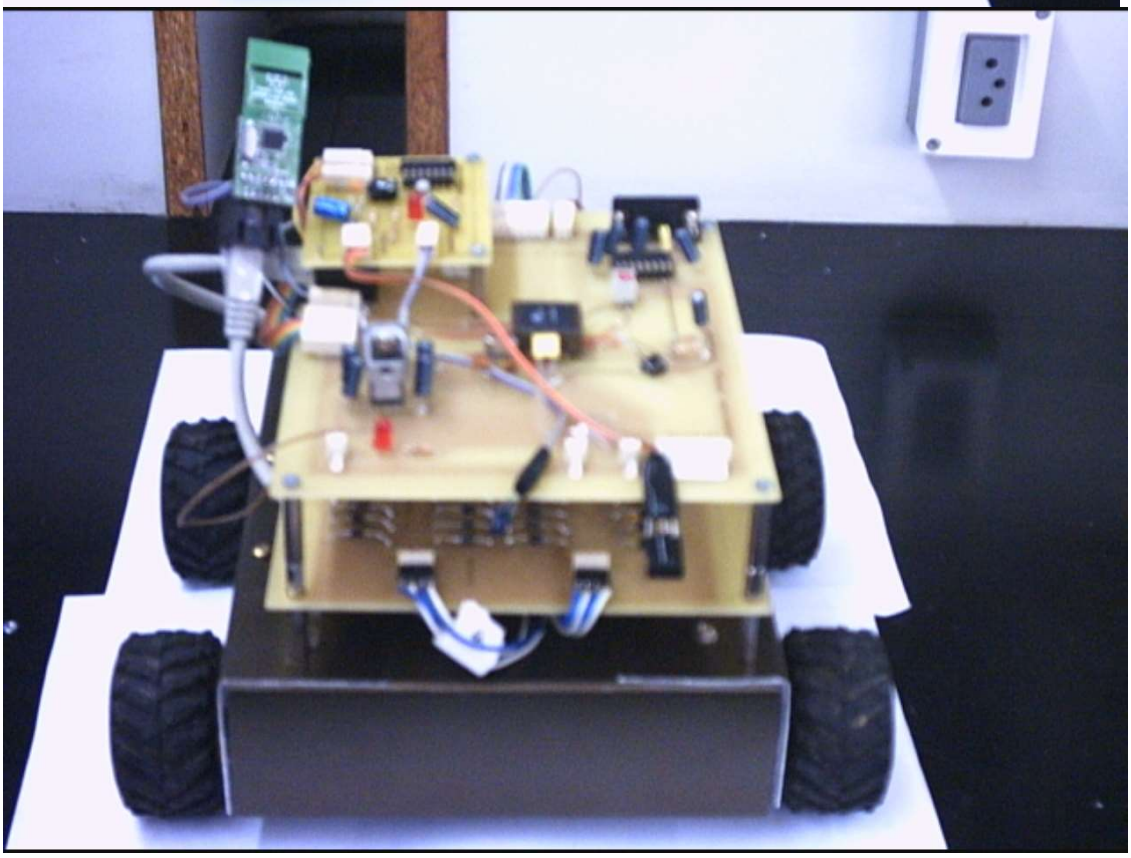
- **Placa de infrarrojos**

Esta placa sirve tanto para el mando como para el vehículo

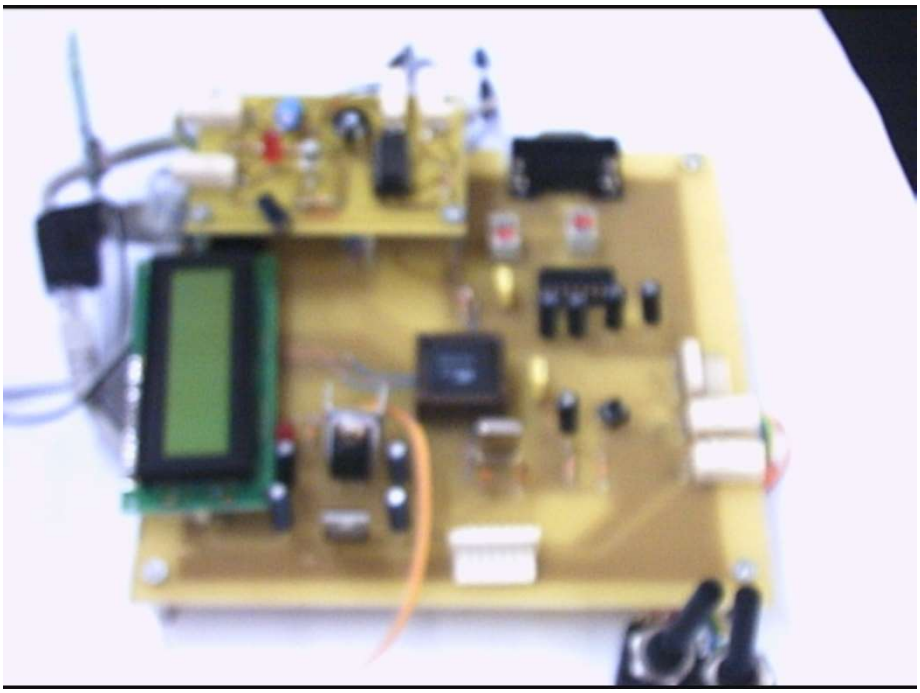


Las siguientes fotos muestran el montaje final del robot:





Las siguientes fotos el montaje interno del mando. Por falta de tiempo no se incluye la carcasa en donde Irán montadas las placas que componen el mando.



6.2. *POSIBLES AMPLIACIONES*

El robot real, debería tener un tamaño superior y una estructura de acero, todo esto en función del peso que se quiera transportar, además de estar equipado con una cubierta de seguridad si se quiere adentrar en ambientes con fuego o de muy bajas o altas temperaturas.

6.2.1. AMPLIACIONES HARDWARE

En el diseño final se podría montar sensores de distancia para evitar colisiones con obstáculos que pudieran ocasionar desperfectos. Además estos sensores nos darían la posibilidad de crear un modo automático para el robot que le permitiese moverse hasta salir de una habitación (modo salir del laberinto). Este tipo de funcionamiento es idóneo si se quiere usar el robot para tomar mediciones en lugares donde no se pueda ver el recorrido a simple vista, por ejemplo conductos de ventilación o edificios siniestrados. Para este tipo de usos también se podría instalar una cámara que transmitiese los datos al mando y con una pequeña pantalla LCD a color, o el propio PC, ver las imágenes que captura el robot
Con un sensor de distancia montado en un motor paso a paso se podría crear un pequeño radar que nos daría la posibilidad de crear un mapa del entorno por donde discurre el robot y nos facilitaría la toma de decisiones a la hora de usar el modo automático.

Otro modo automático, aunque menos efectivo, es el modo seguidor de línea que permite al robot caminar siguiendo una línea continua dibujada en el suelo. Para esto solo es necesario colocar en el robot unos sensores ópticos CNY 70.

Si el robot piensa usarse en terrenos abruptos, se aconseja incorporar una cámara para seguimiento de trayectoria con software de reconocimiento de imágenes que haga interactuar al robot en función del obstáculo encontrado. Para esta solución, sería necesario también incorporar un sistema de transmisión de imágenes por

radiofrecuencia que comunicaran al robot con un PC desde el cual se pueda modificar la trayectoria del robot, incluso sus acciones, en tiempo real. Nos consta que este es el sistema más seguro y utilizado en este momento para robots que se adentran en entornos hostiles, pero por ello también la solución más cara, además habría que plantearse incorporar un microcontrolador que ofreciera mayores prestaciones y su reprogramación.

En cuanto a la tracción, se recomienda una tracción tipo tanque, para lo cual se precisarían 4 motores paso a paso en lugar de 2, para mejorar su rendimiento. La envergadura de la tracción dependería del tamaño y peso del robot y de los terrenos en los cuales se quiera utilizar.

Gracias a los motores paso a paso que incorpora el robot y al gran control que tenemos sobre ellos cabe la posibilidad de ampliar el robot para que siga una trayectoria marcada. Para ello se podría colocar en el mando o en el propio robot un teclado matricial que nos permitiría introducir en el robot la trayectoria a seguir.

6.2.2. AMPLIACIONES SOFTWARE

Una ampliación interesante seria que el programa pidiese por teclado, además de las trayectorias a seguir, la distancia que se quiere recorrer en cada trayectoria, ya que ésta puede ser fácilmente medida gracias a los motores paso a paso. De este modo el robot recorre unas trayectorias más precisas.

También cabe la posibilidad de utilizar un PC como panel virtual para visualizar graficas de los sensores e incluso para el propio control del motor. Para ello un buen programa es el Labview que nos permite crear paneles virtuales sencillos y una comunicación con el micro por el puerto serie. Los datos obtenidos en las gráficas podrían copiarse a ficheros para la creación de informes.

Para la ampliación anterior caben dos posibilidades: que el micro vaya mandando los datos de cada uno de los canales al momento y en tiempo real o que el micro los vaya almacenando en la memoria interna RAM y al pedirlo desde el mando los envíe al PC para su almacenamiento y procesado.

Otra de las posibles ampliaciones seria colocar una tarjeta GSM en el vehículo o en el mando a fin de poder enviar los datos recogidos directamente a un servidor en la nube. Dicho servidor estaría equipado con una base de datos relacional (SQL) o con una no relacional (Non SQL). Con los datos obtenidos se podrían realizar analíticas de datos. Si el vehículo se programa para realizar rutas periódicas, esto nos podría dar un modelo predictivo de la zona que se está observando.