ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

Robot explorador para dibujo mapas 2-D controlado desde Internet mediante PIC-WEB

AUTOR: Antonio Jesús Marcos Díaz

DIRECTOR: Manuel Sánchez Alonso

Julio / **2012**

Autor	Antonio Jesús Marcos Díaz
E-mail del Autor	amarcosj@hotmail.com
Director(es)	Manuel Sánchez Alonso
E-mail del Director	manuel.sanchez@upct.es
Titulo del PFC	Robot explorador para dibujo mapas 2-D controlado desde Internet mediante PIC-WEB

Resumen

El principal objetivo del proyecto es implementar un sistema de control vía internet para controlar un robot explorador para dibujo de mapas 2-D realizado en otro PFC.

El control del robot se realizará a través de una página web JSP alojada en un servidor de aplicaciones y gracias al sistema PIC-WEB que será el encargado de la comunicación entre ese servidor y el robot. Para ello se deberá de desarrollar un proyecto web para hacer uso de programación java en la página JSP e implementar un puente RS232-TCP/IP en el sistema PIC-WEB.

Titulación	Ingeniero Técnico Telecomunicación
Intensificación	Telemática
Departamento	Electrónica, Tecnología de Computadores y Proyectos
Fecha de Presentación	09-2012

INDICE DE ILUSTRACIONES:

INDICE DE TABLAS:

1 MEMORIA DESCRIPTIVA

1.1 INTRODUCCIÓN

La importancia del uso máquinas en la historia está más que demostrada, desde tiempos remotos las máquinas han ayudado al hombre a conseguir lo que un humano naturalmente no puede. Una máquina puede realizar una tarea de manera repetitiva sin problemas, con una fuerza sobrehumana, con una precisión absoluta, sin equivocaciones, sin cansancio, sin sensaciones que los hagan dudar o realizar malas decisiones, en definitiva, el súmmum del trabajador, la herramienta perfecta.

Es totalmente lógico que cuando el ser humano empezó a desarrollar la electrónica quiso dotar de autonomía a estas máquinas, en principio debido a la necesidad de implementar un servicio, o realizar una tarea, que no necesite la atención permanente de personal. En este momento de la historia aparecieron los robots, máquinas electrónicas capaces de ejecutar automáticamente distintas operaciones o movimientos. Aun así no se dejó de lado el poder de controlar estos robots, la automatización de un sistema no significa eliminar de la ecuación al ser humano, sino que se convierte en la parte más importante de todas, la que los controla, convirtiéndose así en usuario. Y siendo así la electrónica, que componen estas máquinas robotizadas, y su programación, las encargadas de transformar simples movimientos humanos en simples movimientos robóticos, de crear la transparencia, convirtiendo un complejo sistema electromecánico en una simple herramienta.

Las increíbles aplicaciones que se han llegado a conseguir han ratificado de sobra que ya no existen problemas a la hora de programar sistemas automatizados que nos ayuden a resolver cualquier tipo de problema, pero siempre estando sujetos al problema de la distancia. Y gracias a la evolución de la informática, a internet y a la sociedad de la información, este problema ha desaparecido, ha llegado el momento de conferir un control telemático a todos estos robots o sistemas automatizados.

La intención de este PFC es la de demostrar que todo sistema robótico ya está preparado para funcionar a grandes distancias, del control absoluto, como, cuando y donde uno quiera.

1.1.1 Contenidos

La estructura de la memoria estará dividida en los siguientes capítulos. En primer lugar, se expondrá un planteamiento general del sistema, en el cual se presentan los diferentes bloques que componen el sistema desarrollado: Sistema PIC-WEB, robot explorador, servidor de aplicaciones e interfaz gráfica de control. En éstos puntos se describirán levemente las características técnicas de los diferentes componentes de los que

están compuestos los bloques, así como su funcionalidad general como bloques. También se realizará una pequeña descripción de porqué han sido utilizados estos componentes.

En el capítulo de fases del proyecto estará dedicado a explicar las diferentes fases del desarrollo del proyecto, ahondando en la información hardware y software del proyecto y por supuesto en la descripción funcional de cada componente que lo componen.

En el capítulo planos se podrán observar todas las conexiones de los diferentes bloques, así como su interconexión entre componentes. También se expondrán los planos electrónicos de cada componente.

En el siguiente capítulo se realizará una conclusión dando a lugar a una visión futura del proyecto, sus posibles mejoras y aplicaciones. Y por último se publicará el capítulo de bibliografía y el capítulo anexos con toda la programación usada en cada bloque del proyecto.

1.2 PLANTEAMIENTO GENERAL

1.2.1 Introducción

El objetivo de esta introducción es proporcionar una visión global de todo el sistema, para lo cual, se procede a hacer un breve análisis de la estructura y funcionamiento de cada bloque que forma parte del proyecto.

El robot está basado en un proyecto anterior, en el cual era capaz de ser teledirigido obedeciendo unas órdenes de una aplicación java controlada por un usuario desde un pc y con la limitación de la distancia de comunicación ofrecida por un enlace de radiofrecuencia. Este robot era capaz, además de controlar el movimiento de los motores, de medir las distancias con un sensor de ultrasonidos y enviar los datos al PC, en el cual se procesaba esta información para crear un mapa 2D de la situación donde se encontraba.

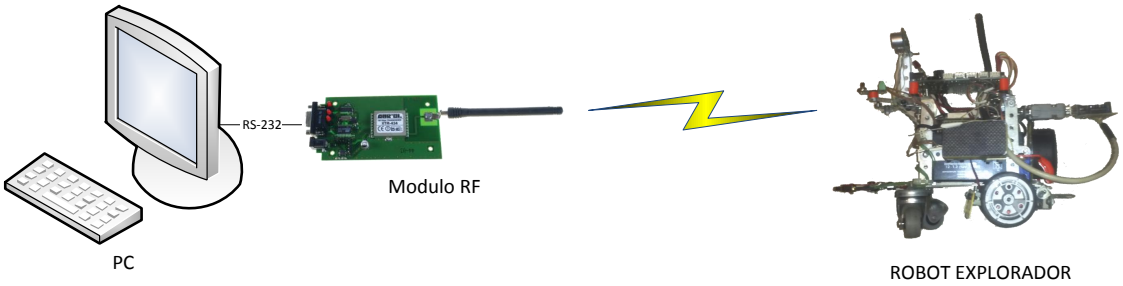


Ilustración 1: Plano general PFC anterior

La idea principal del proyecto es que un usuario pueda controlar el robot desde cualquier terminal con acceso a internet. Para ello se va a hacer uso de un pequeño sistema electrónico llamado PIC-WEB, el cual puede conectarse a internet gracias a los componentes de los que está compuesto, que más adelante describiremos.

La idea inicial era utilizar este sistema PIC-WEB como servidor, para alojar una página web desde la cual poder realizar el control de robot, pero debido a su pequeño espacio de memoria se pensó en añadir al proyecto un servidor web en el cual poder alojar una buena página pudiendo soportar tecnologías actuales como PHP, flash, JavaScript, MySQL o incluso el nuevo HTML5.

Para este proyecto se ha procedido a utilizar un servidor de aplicaciones, en este caso Apache Tomcat 6, que además de realizar la función de servidor web será capaz de procesar código java, así se podrá reutilizar código del proyecto anterior, ahorrándonos esfuerzos y trabajos innecesarios traduciendo código a un lenguaje distinto. Para ello el servidor deberá tener instalado los componentes necesarios y estar configurado para ello.

La interfaz gráfica de control será una página web JSP (Java Server Pages) en la que se podrá incrustar, por medio de unas etiquetas, la programación java que será procesada por el servidor de aplicaciones.

Por lo tanto el plano general, con sus bloques más importantes, quedará de la siguiente manera:

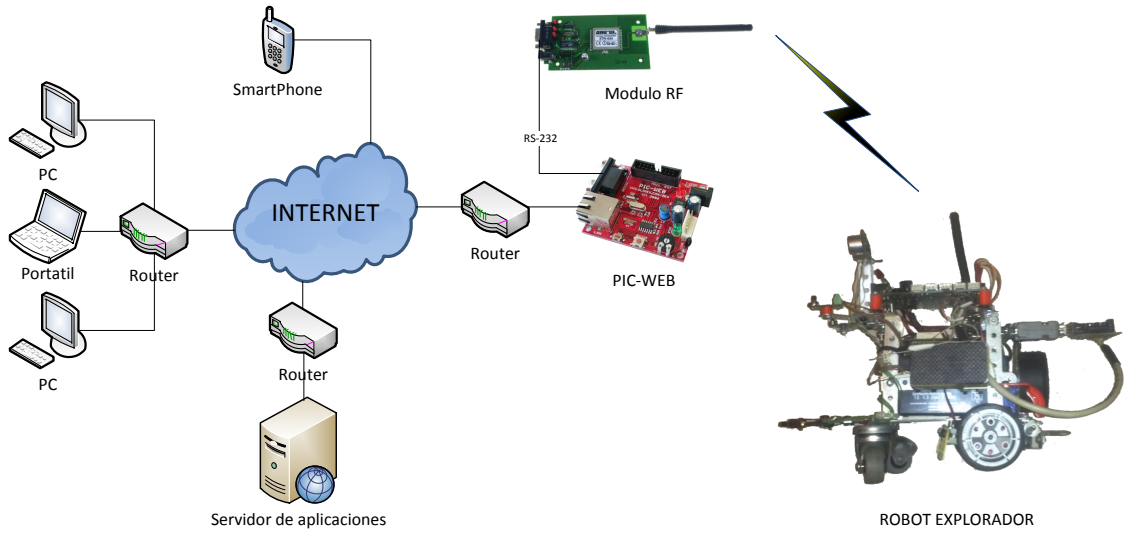


Ilustración 2: Plano general PFC

El funcionamiento fundamental del sistema sería de la siguiente manera. El usuario solamente necesita de un terminal con acceso a internet, un PC o incluso un Smartphone, para poder controlar el robot por medio de una página web alojada en el servidor de aplicaciones. En el momento que el usuario decida hacer click en un botón de la página el servidor procesará la acción y enviará los datos hacia el PIC-WEB, este a su vez realizará un puente entre el puerto de Ethernet y el puerto serie, por lo tanto reenviará directamente los datos al robot, el cual realizará la acción y tomará medidas. Acto seguido el robot envía las medidas al PIC-WEB, este reenvía hacia el servidor, se procesan y se muestran los resultados en la pantalla del usuario. En el siguiente esquema se puede ver fácilmente cual es el recorrido del flujo de la información.

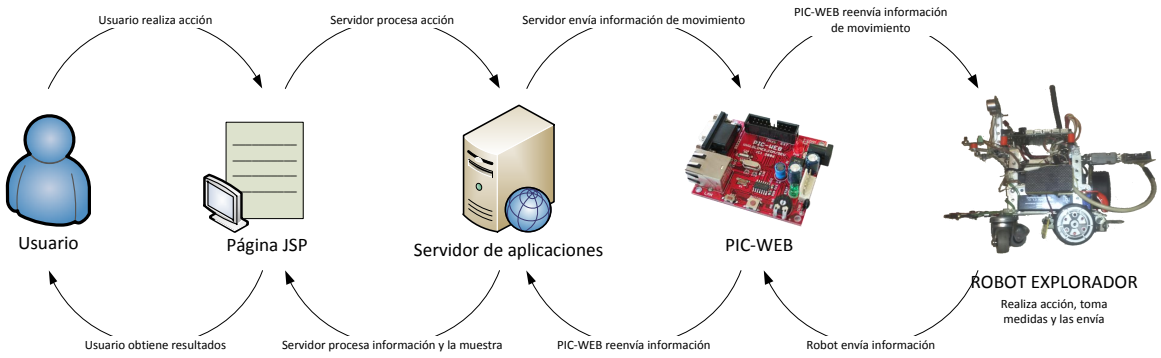


Ilustración 3: Flujo de información en el sistema

A continuación describiremos más detalladamente la funcionalidad de los distintos bloques del sistema y porqué se ha decidido elegir los mismos.

1.2.2 Sistema PIC-WEB

El PIC-WEB es una plataforma implementada con el microcontrolador PIC18F452**,** la cual incluye el stack de código abierto TCP/IP AN833 de Microchip. Gracias a ello, este sistema no sólo será capaz de tener acceso a internet sino que podrá actuar como un servidor web. La placa soporta los siguientes protocolos: SLIP, ARP, IP, ICMP, TCP, UDP, HTTP, DHCP y FTP.

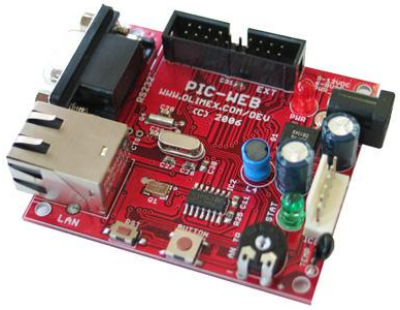


Ilustración 4: Flujo de información en el sistema

El stack de Microchip es de código abierto, por lo que es posible modificarlo y recompilarlo cuantas veces sea necesario, y adaptarlo a nuestras necesidades. El stack es modular y flexible, lo que permite activar y desactivar módulos, como también usar páginas web dinámicas que permiten controlar todos los recursos del PIC remotamente usando los protocolos soportados.

Los elementos más importantes de ésta placa son: El microcontrolador PIC18F452, en el cual se grabará la programación adecuada para realizar su función. El módulo ENC28J60, que hará que el sistema pueda conectarse a una LAN con acceso a internet. Y el módulo MAX-232 junto al puerto RS-232 donde irá conectado la tarjeta de RF.

El principal problema que presenta esta placa es que posee una memoria para almacenamiento de datos de sólo 1 Mbit. Por lo que no podría alojar páginas web con las características usadas actualmente y menos procesar código java que es lo que se pretende en este proyecto. Así que su funcionalidad en el proyecto se limita a reenviar los datos que le lleguen desde el servidor de aplicaciones hacia el robot y exactamente lo mismo pero a la inversa.

Como se ha comentado anteriormente, un módulo de comunicación por radiofrecuencia estará conectado al puerto RS-232 del PIC-WEB, se trata de WIZ2-434-RS.

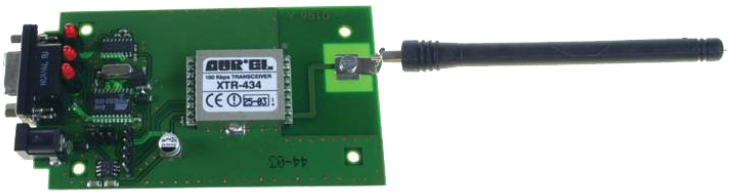


Ilustración 5: Módulo RF. WIZ2-434-RS

Su funcionamiento es totalmente transparente al usuario, es decir todo lo que se envíe por el puerto RS-232 llegará de la misma manera a un módulo diferente situado en el robot. Los datos transferidos a la unidad remota están libre de errores ya que, al llegar, son verificados y confirmados por el receptor. La configuración del puerto serie es de 9600 bps, 8N1. El alcance de este módulo RF es de 40 metros en interior y 150 metros en exterior. Se describirán mas detalles en los puntos posteriores.

Por lo tanto, el sistema PIC-WEB deberá estar conectado a un router con conexión a internet, quedando este bloque de la siguiente manera:

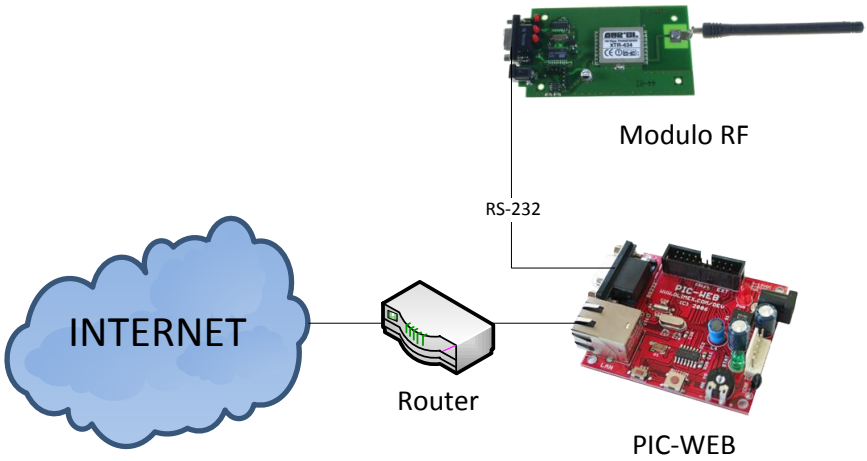


Ilustración 6: Bloque PIC-WEB y Módulo RF

Por supuesto, el router no tiene que ser fijo, se podría utilizar un router 3G con puerto LAN y montar el Bloque PIC-WEB/RF en el mismo Robot. Siendo de esta manera totalmente independiente de una localización específica y sería la cobertura 3G la que limitaría el alcance del Robot. Si se decidiera en un futuro realizar la unión de este bloque con el robot, se podría prescindir de los módulos RF y realizar una conexión directa de los puertos RS-232 del PIC-WEB y el robot.

Una muy buena alternativa actual al sistema PIC-WEB sería utilizar el nuevo RaspBerry PI, un mini PC con S.O. de código abierto, Linux Fedora, con una gran capacidad de procesamiento y a un precio bastante asequible. Muy difícil de conseguir debido a la gran demanda existente.

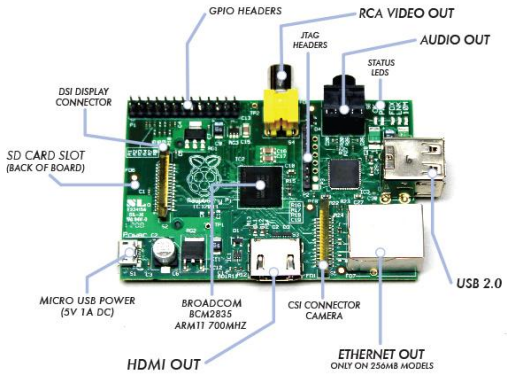


Ilustración 7: RaspBerry PI

1.2.3 Robot Explorador

El robot explorador está compuesto por los siguientes componentes: Microcontrolador PIC16F84A, controladora MSx84, módulo RF WIZ2-434-RS, módulo MAX232, sensor de ultrasonidos SRF05, servomotores, ruedas y batería.

Los principales componentes se encuentran situados en diferentes niveles, y de la forma más compacta posible.

Si se empieza la descripción de abajo arriba, los primeros componentes que se encuentran son lógicamente las ruedas, tanto las motrices como las ruedas locas.

En el siguiente nivel se sitúa la batería, la cual es el elemento más pesado del robot, se ha situado en la parte más baja posible, para dotarlo de más estabilidad. Esto es necesario para que el robot pueda desplazarse como se ha previsto.

En la parte intermedia del chasis se ha colocado la tarjeta de comunicación inalámbrica, a la cual se ha conectado el módulo MAX232, el cual sobresale por la parte trasera.

En la zona superior de la estructura se situará la tarjeta MSx84 y el sensor SRF05. En cuanto al MSx84 se refiere, la causa es para facilitar el cableado con los demás componentes. Y si se habla del SRF05, se ha dado una altura mayor posible, esto es para que tome buenas medidas y no cometa error de medida con objetos indeseados.

A continuación se puede observar una figura, en la cual se pueden identificar fácilmente los componentes.

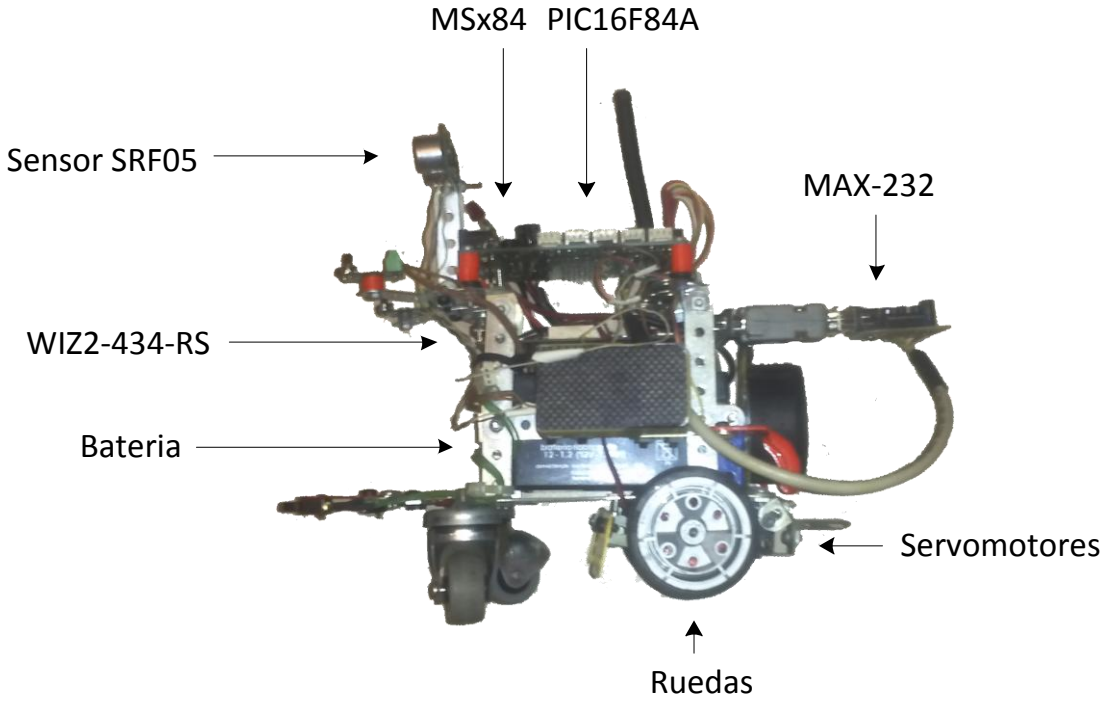


Ilustración 8: Disposición física de los componentes del robot explorador

El diagrama funcional correspondiente al conexionado de los componentes del robot es el siguiente:

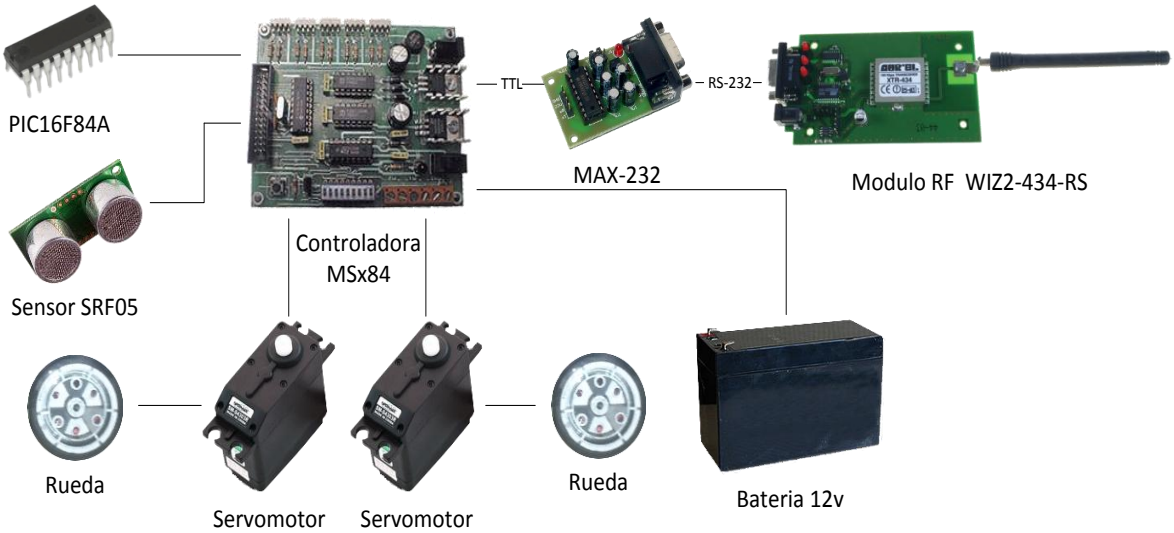


Ilustración 9: Diagrama funcional de los componentes del robot explorador

A continuación se va a describir la funcionalidad de cada componente anteriormente nombrado.

- **Controladora MSx84:**

La tarjeta de control MSx84 de Microsystems Engineering es una tarjeta autónoma de propósito general y bajo coste que, controlada por un PIC 16x84, es capaz de controlar el estado de cinco sensores de entrada y gobernar dos motores DC o uno paso a paso (PAP) de salida, en función del software de aplicación grabado en el microcontrolador.

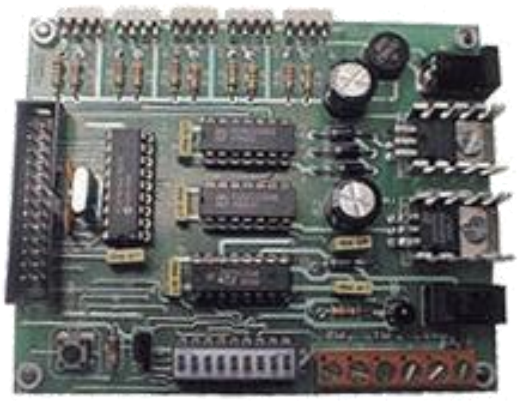


Ilustración 10: Controladora MSx84

Los componentes básicos de esta controladora y que se resaltan son los siguientes:

- Conector de expansión **PIC-BUS** de 26 pines compatible con la tarjeta µPic Trainer de Microsystems Engineering. A través de él se obtienen todas las señales del PIC, pudiéndose de esta forma modificar o añadir nuevos periféricos. Gracias a PIC-BUS se puede grabar el PIC16x84 ubicado en la tarjeta mediante el sistema µPic Trainer, así como aprovechar y hacer uso tanto de los periféricos de este último como de todo el software de grabación.

- Circuito de carga de baterías de 12VDC de plomo ácido.

- Conectores y circuitos para adaptación de hasta 5 sensores o dispositivos de entrada (**J0-J5**). Estos conectores son adecuados para sensores de una sola entrada. Si alguno de estos conectores no fuera usado quedaría libre el pin correspondiente del microcontrolador para otro posible uso. Las líneas de entrada de los conectores van conectados a los pines del microcontrolador.

- Driver para el accionamiento de dos motores DC o uno paso a paso, **L239B**.

- Jumper de selección de tensión aplicada a los motores (5VDC ó 12VDC ó tensión batería externa).

- Zócalo para microcontrolador **PIC 16x84**, circuito oscilador y pulsador de RESET incluido en la tarjeta. Según el software grabado en el microcontrolador se podrá controlar los posibles sensores y motores conectados a la tarjeta MSx84.

- **PIC16F84A:**

El PIC16F84A es un microcontrolador de la familia PIC, fabricado por la empresa Microchip. Se trata de uno de los microcontroladores más populares del mercado actual, ideal para el aprendizaje, debido a su arquitectura de 8 bits, 18 pines, y un conjunto de instrucciones RISC muy fácil de memorizar y de entender.

En los últimos años se ha popularizado el uso de este microcontrolador debido a su bajo coste y tamaño. Puede ser programado tanto en lenguaje ensamblador como en Basic y principalmente en C, para el que existen numerosos compiladores.

Ilustración 11: PIC16F84A

En este PFC el microcontrolador PIC16F84A se encargará de realizar los movimientos del robot, por lo que tendrá que procesar los datos recibidos y mandar las señales pertinentes a los servomotores.

También se encargará de la recogida de medidas del sensor SRF05 y enviarlas de la forma adecuada.

- **Transceptor multipunto WIZ2-434-RS:**

Este tipo de módulos transceptores están preparados para la transferencia de datos vía radio entre uno o mas módulos. Puede crearse una red de transceptores con un módulo principal o master y hasta 255 esclavos. Cada módulo se puede configurar como master o como esclavo asignando a cada uno una dirección. Cada vez que el master transmite un paquete de datos debe indicar a qué esclavo va dirigido. Cuando recibe un paquete de datos se recibe también de qué esclavo procede. Estos paquetes de datos pueden ser de hasta 96 bytes. Se alimentan con una F.A. externa de entre 9 y 15Vdc con positivo al centro del conector. La velocidad se selecciona entre 9600, 19200, 57600 y 115200 baudios y el alcance aproximado es de 500 m en exteriores, con una frecuencia de 433.92 MHz.

Estará conectado al módulo MAX-232 y comunicará con su homólogo conectado en el sistema PIC-WEB.

- **Módulo MAX232:**

El MAX232 es un circuito integrado de Maxim que convierte las señales de un puerto serie RS-232 a señales compatibles con los niveles TTL de circuitos lógicos. El MAX232 sirve como interfaz de transmisión y recepción para las señales RX, TX, CTS y RTS.

Ilustración 12: Módulo MAX232

El circuito integrado tiene salidas para manejar niveles de voltaje del RS-232 (aprox. ± 7.5 V) que las produce a partir de un voltaje de alimentación de + 5 V utilizando multiplicadores de voltaje internamente en el MAX232 con la adición de condensadores externos. Esto es de mucha utilidad para la implementación de puertos serie RS-232 en dispositivos que tengan una alimentación simple de + 5 V.

- **Sensor ultrasónico SRF05:**

Consiste en un medidor ultrasónico de distancias de bajo costo desarrollado por la firma Devantech Ltd. El módulo SRF05 es una evolución del módulo SRF04 y está diseñado para aumentar la flexibilidad, aumentar el rango de medida y reducir costes. Es totalmente compatible con el SRF04 y el rango de medida se incrementa de 3 a 4 metros. Se muestra en la siguiente figura:

Ilustración 13: Sensor ultrasónico SRF05

Dispone de un nuevo modo de operación que se selecciona simplemente conectando el pin "Mode" a GND. Dicho modo permite al SRF05 emplear un único pin de E/S que sirve tanto para dar la orden de inicio o disparo, como para obtener al medida realizada (ECO).

Cuando el pin de "Modo" no se emplea y se deja sin conectar, el SRF05 trabaja de la misma manera que el SRF04. Esto es, la señal de disparo y la salida de ECO se realizan por pines diferentes.

- **Servomotores:**

Un servomotor es un motor eléctrico que consta con la capacidad de ser controlado, tanto en velocidad como en posición. Se utilizan frecuentemente en sistemas de radio control y en robótica, pero su uso no está limitado a estos. Es posible modificar un servomotor para obtener un motor de corriente continua que, si bien ya no tiene la capacidad de control del servo, conserva la fuerza, velocidad y baja inercia que caracteriza a estos dispositivos.

Ilustración 14: Servomotor

Un servomotor son mecanismos. En otras palabras, un servomotor es un motor especial al que se ha añadido un sistema de control (tarjeta electrónica), un potenciometro y un conjunto de engranajes.

Estarán controlados por el driver L293B incluido en la controladora MSx84. Más adelante se describirá detalladamente este driver.

- **Batería de plomo ácido 12V:**

Una simple batería de plomo ácido de 12V. Está formada por un depósito de ácido sulfúrico y dentro de él una serie de placas de plomo dispuestas alternadamente.

Ilustración 15: Batería de plomo ácido 12V

1.2.4 Servidor de aplicaciones y JSP

- **Servidor de aplicaciones:**

Se denomina servidor de aplicaciones a un servidor en una red de computadores que ejecuta ciertas aplicaciones.

Usualmente se trata de un dispositivo de software que proporciona servicios de aplicación a las computadoras cliente. Un servidor de aplicaciones generalmente gestiona la mayor parte, o la totalidad, de las funciones de lógica de negocio y de acceso a los datos de la aplicación. Los principales beneficios de la aplicación de la tecnología de servidores de aplicación son la centralización y la disminución de la complejidad en el desarrollo de aplicaciones.

En nuestro caso utilizaremos Apache Tomcat 6.0, funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems.

Tomcat es un servidor web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones propiamente dicho, como JBoss o JOnAS, sino un contenedor de servlets, pero durante todo el PFC se le considerará como tal ya que cumple con la función de un servidor de aplicaciones. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets.

Tomcat puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad. Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

Para la creación de este servidor se ha procedido a la instalación de VirtualBox de Oracle, con el cual podemos instalar un S.O. en una máquina virtual. Así podremos llevarnos el servidor al lugar donde queramos. En el caso de este proyecto, el servidor estará montado sobre mi propio PC en mi domicilio privado.

El sistema operativo elegido ha sido Ubuntu Server 10.04 (Lucid Lynx), un S.O. de código abierto orientado a funcionar como un servidor, en el cual se ha procedido a instalar el servidor de aplicaciones Tomcat 6.0, también se ha instalado el kit de desarrollo de java (JDK) oficial desarrollado por Sun.

- **JSP:**

JavaServer Pages (JSP) es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.

Las JSP's permiten la utilización de código Java mediante scripts. Además, es posible utilizar algunas acciones JSP predefinidas mediante etiquetas.

El funcionamiento general de la tecnología JSP es que el Servidor de Aplicaciones interpreta el código contenido en la página JSP para construir el código Java del servlet a generar. Este servlet será el que genere el documento (típicamente HTML) que se presentará en la pantalla del Navegador del usuario.

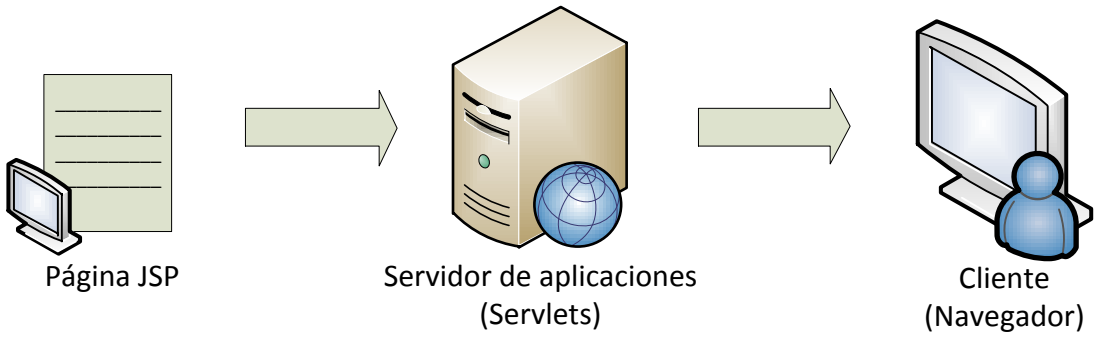


Ilustración 16: Esquema generación de un servlet a partir de JSP

El rendimiento de una página JSP es el mismo que tendría el servlet equivalente, ya que el código es compilado como cualquier otra clase Java. A su vez, la máquina virtual compilará dinámicamente a código de máquina las partes de la aplicación que lo requieran. Esto hace que JSP tenga un buen desempeño y sea más eficiente que otras tecnologías web que ejecutan el código de una manera puramente interpretada.

La principal ventaja de JSP frente a otros lenguajes es que el lenguaje Java es un lenguaje de propósito general que excede el mundo web y que es apto para crear clases que manejen lógica de negocio y acceso a datos de una manera prolija. Esto permite separar en niveles las aplicaciones web, dejando la parte encargada de generar el documento HTML en el archivo JSP.

Otra ventaja es que JSP hereda la portabilidad de Java, y es posible ejecutar las aplicaciones en múltiples plataformas sin cambios. Es común incluso que los desarrolladores trabajen en una plataforma y que la aplicación termine siendo ejecutada en otra.

Los servlets y Java Server Pages (JSP) son dos métodos de creación de páginas web dinámicas en servidor usando el lenguaje Java. En ese sentido son similares a otros métodos o lenguajes tales como el PHP, ASP o los CGIs, programas que generan páginas web en el servidor. Sin embargo, se diferencian de ellos en otras cosas. Para empezar, los JSPs y servlets se ejecutan en una máquina virtual Java, lo cual permite que, en principio, se puedan usar en cualquier tipo de ordenador, siempre que exista una máquina virtual Java para él. Cada servlet (o JSP, a partir de ahora lo usaremos de forma indistinta) se ejecuta en su propia hebra, es decir, en su propio contexto, pero no se comienza a ejecutar cada vez que recibe una petición, sino que persiste de una petición a la siguiente, de forma que no se pierde tiempo en invocarlo (cargar programa + intérprete). Su persistencia le permite

también hacer una serie de cosas de forma más eficiente: conexión a bases de datos y manejo de sesiones, por ejemplo.

Las JSPs son en realidad una forma alternativa de crear servlets ya que el código JSP se traduce a código de servlet Java la primera vez que se le invoca y en adelante es el código del nuevo servlet el que se ejecuta produciendo como salida el código HTML que compone la página web de respuesta.

Por todo lo descrito anteriormente queda justificado el uso de un servidor de aplicaciones y la creación de una página web JSP con el código java necesario para el control del robot desde la misma.

1.2.5 Interfaz gráfica de control: Página web

Para crear la interfaz gráfica de control se ha propuesto crear un sitio web, en el cual encontraremos un apartado específico llamado *Proyecto*.

En este apartado específico se ha hecho uso de una programación JSP. Para ello se ha creado un *webproject* en el entorno de programación *Eclipse* integrando así el código java necesario para el funcionamiento de los controles y de la creación del mapa 2-D.

La creación de los botones se debe realizar en html y la funcionalidad se adquirirá de las funciones creadas en java y añadidas al proyecto web. De todo esto se encarga el entorno de desarrollo *Eclipse* haciendo mucho más sencilla la tarea, creando los archivos necesarios para incluir en la carpeta que utilizará el servidor de aplicaciones Tomcat para poner online la página web.

Por lo tanto una vez entremos en la sección proyecto de la página web, nos encontraremos con un cuadro que tiene recoger las dimensiones del plano donde se va a dejar el robot para que intente reconstruirlo asociado a este lugar. Las dimensiones son obtenidos mediante dos elementos JTextField, que son dos campos editables en los cuales hay que introducir dos números enteros. En el caso que no se introduzca bien los números enteros no podrá continuar la aplicación, este sistema es necesario para el control de errores.

Además se ha construido una interfaz gráfica para facilitar el movimiento del robot por parte del usuario, esta interfaz hace más interactiva la aplicación y facilita su funcionamiento. Se compone de cuatro flechas que indican el sentido de movimiento del robot y un símbolo de stop, que es el encargado de para el robot.

Una vez realizada alguna acción aparecerá en una ventana emergente el mapa realizado con las medidas recibidas desde el robot.

A continuación se puede observar en la figura 17 la estructura de la interfaz gráfica integrada en la página JSP.

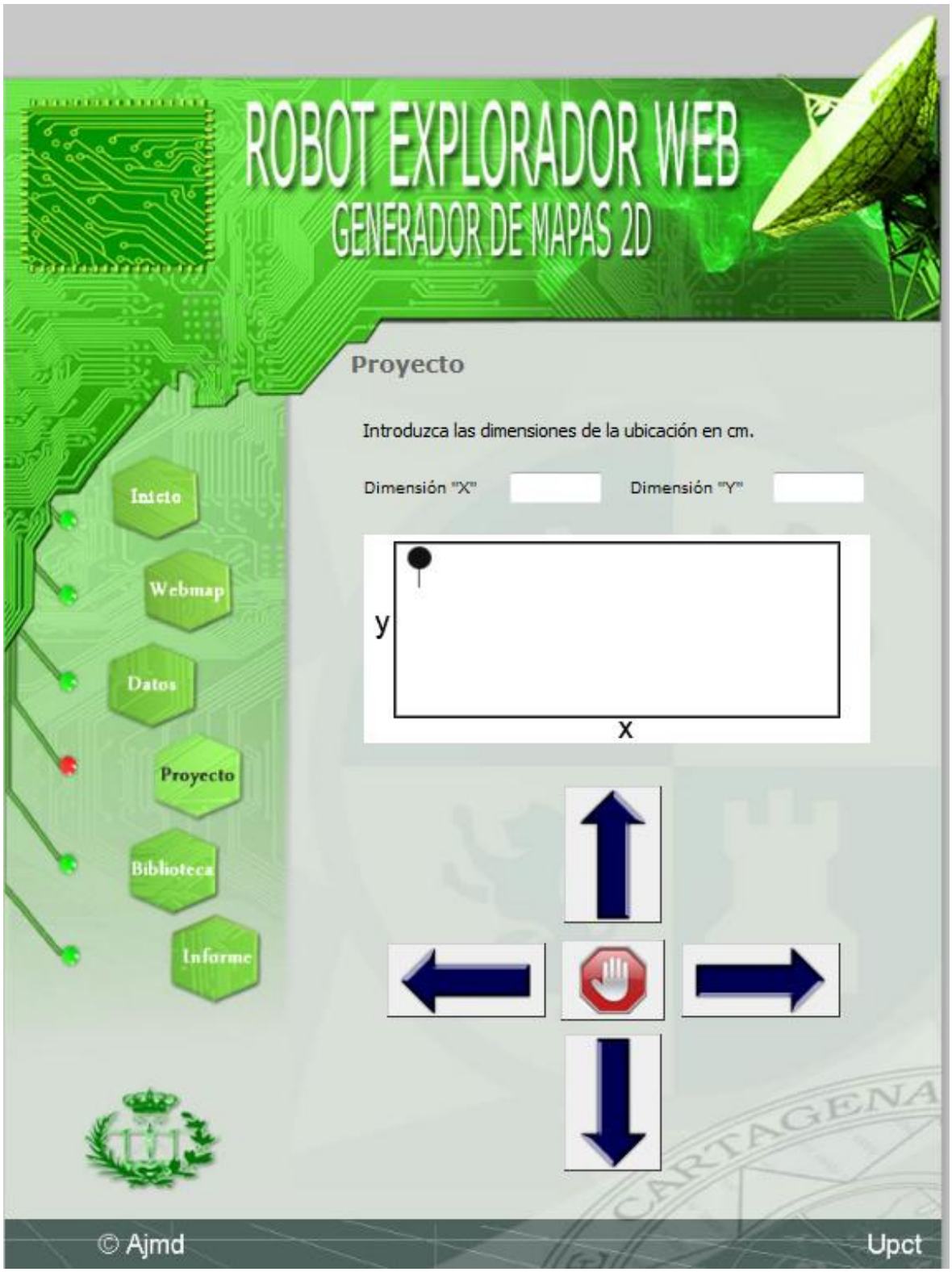


Ilustración 17: Imagen de la interfaz gráfica de control

1.3 FASES DEL PROYECTO

Para cumplir los objetivos de este proyecto, se han desarrollado **2 fases** bien identificadas:

La fase 1 corresponde con la adquisición de información de todos los elementos que interviene en el proyecto así de hardware como de software.

- Recopilar información hardware (Robot explorador).
- Recopilar información Sistema PIC-WEB.
- Recopilar Información Software .

La fase 2 se corresponde con las pruebas y experimentación de cada uno de los bloques y el ensamblado de cada módulo, en la que se dotarán a todos los módulos descritos en la Fase 1 de su funcionalidad:

- Configuración del PIC. Comprobación de medidas procedentes de los sensores.
- Ajustes de los sensores.
- Recepción y envío de medidas en el PIC-WEB.
- Control de movimientos desde el PIC-WEB hacia el Robot.
- Software gestor del PIC-WEB.
- Recepción de medidas vía Internet desde el PIC-WEB en el PC.
- Software gestor de la interfaz gráfica que controla al PIC-WEB.

1.3.1 FASE 1

1.3.1.1 Recopilar información hardware

En los siguientes subapartados se trata de reflejar toda la información válida adquirida para poder implementar la parte Hardware del proyecto.

1.3.1.1.1 Anterior PFC con fundamentos del robot y comunicaciones

Este proyecto desarrollado en esta memoria se ha basado en otro proyecto anteriormente realizado por un alumno, con título *"Robot explorador para dibujo mapas 2-D"*.

En él se planteaba implementar un sistema de control para robot autónomo basado en microcontroladores, aprovechando el PFC de un robot (sistema mecánico) y el sistema de comunicación por radiofrecuencia con un ordenador desarrollado a su vez por otro alumno en su PCF; era preciso también desarrollar el software capaz de capturar los datos enviados por el robot y procesarlos para representar la planta del plano del lugar donde se encontraba el robot, así como controlar la posición y movimientos del mismo.

A continuación se describirán los componentes más importantes del robot como pueden ser el PIC16F84A, la controladora MSx84 o el módulo MAX232. Los demás componentes se describirán en puntos anteriores.

- **PIC16F84A:**

Los microcontroladores PIC utilizan una arquitectura Harvard que dispone de dos memorias independientes a las que se conecta mediante dos grupos de buses separados. Estas memorias son la de datos y la de programa.

Los buses utilizados para acceder a estas memorias son totalmente independientes, de esta manera se puede acceder a las dos memorias simultáneamente y que las instrucciones se ejecuten en menos ciclos de reloj.

Las ventajas de utilizar una arquitectura Harvard se pueden resumir por un lado en que una instrucción sólo ocupa una posición de memoria de programa, lo que implica una mayor velocidad y una menor longitud de programa; por otro lado, el tiempo de acceso a las instrucciones puede superponerse con el de los datos, logrando una mayor velocidad de operación.

El procesador segmentado o Pipeline realiza simultáneamente la ejecución de una instrucción y la búsqueda de código de la siguiente, de esta manera se puede ejecutar una instrucción por cada ciclo.

La CPU se puede clasificar según el número de instrucciones de las que está compuesto. En este caso el PIC16F8A utiliza el juego de instrucciones RISC (Reduced Instruccion Set Computer). Utiliza un juego reducido de instrucciones, más concretamente 35. Estas instrucciones son muy simples, ya que todas se ejecutan en un ciclo de máquina, menos aquellas cuya operación es de salto que ocupan dos. En resumen, la empresa de fabricación Microchip ha dotado sus microcontroladores de un procesador RISC para ejecutar a muy alta velocidad un número muy reducido de instrucciones.

- CARACTERÍSTICAS:

Memoria de programa: 1 Kbyte x 14 bits
Memoria de datos: 68 bytes
Memoria de datos EEPROM: 64 bytes
Niveles de Pila: 8
Interrupciones: 4 tipos diferentes
Juego de instrucciones: 35 códigos
Encapsulado: Plástico DIP de 18 patillas
Frecuencia de trabajo: Hasta 10MHz
Temporizadores: Sólo uno (TMR0) más un Perro Guardián (WDT)
Líneas de E/S Digitales: 13 (5 Puerta A y 8 Puerta B)
Voltaje de alimentación: de 2 a 6 V DC
Voltaje de grabación: de 12 a 16 V DC

- ORGANIZACIÓN DE LA MEMORIA:

La memoria del PIC16F84A se divide en dos áreas principales: la primera es la de Registros Especiales, que nos sirven para configurar el dispositivo (entradas y salidas, interrupciones, timer/contador, etc...) y obtener información sobre su estado actual (resultados de operaciones lógicas, interrupciones, lectura de entradas y escritura de salidas, etc...). La segunda es la de Registros de Uso General, que consiste en 68 registros (de 8 bits cada uno), los cuales podemos utilizar para lo que estimemos necesario. En la siguiente figura podemos ver esta división junto con el nombre de los registros especiales y sus direcciones en hexadecimal. El primer grupo toma como nombre Banco0 y el segundo grupo Banco1.

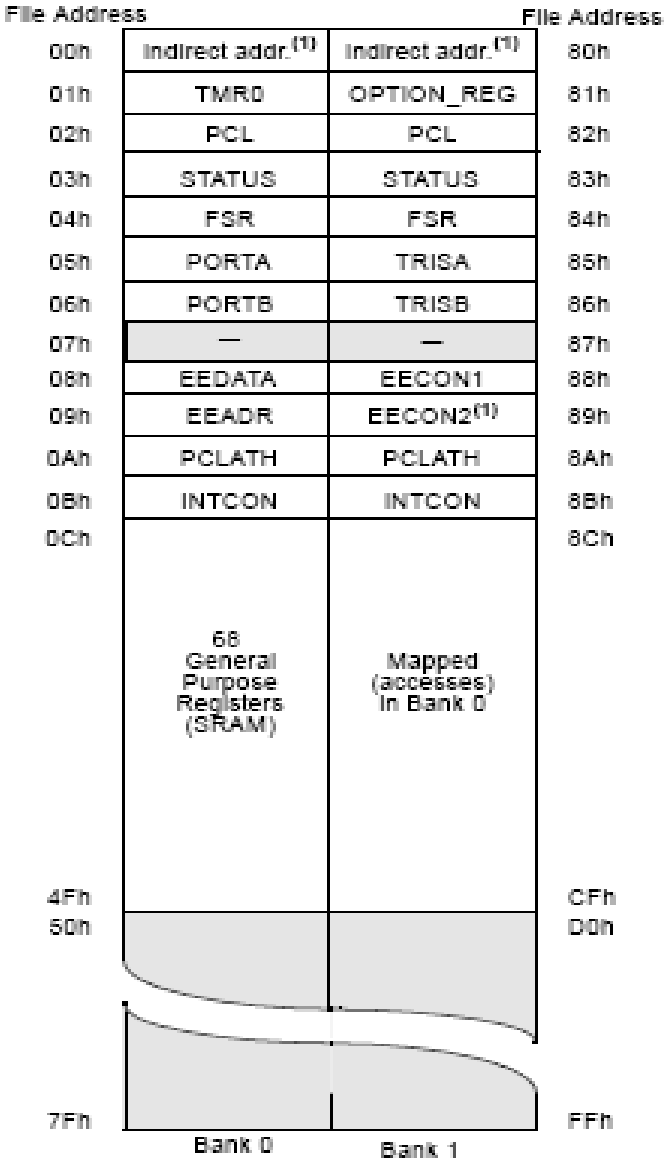


Ilustración 18: Organización memoria PIC16F84A

Sólo están implementadas las 80 primeras posiciones de cada banco, de las cuales las 12 primeras están reservadas a los registros de propósito específico, que son los encargados del control del procesador y sus recursos.

Está la posibilidad de utilizar los registros de propósito general, los cuales son 68. Estos sólo son operativos en el banco 0, ya que los del banco 1 están mapeados con estos. Las direcciones de los registros de propósito general son entre 0x0C y 0x4F.

En el caso que se desee acceder al banco 0 *'bcf STATUS, RP0'* y por el contrario para acceder al banco 1 de memoria de datos *'bsf STATUS, RP0'*. Estas dos instrucciones son bastante empleadas en los programas, ya que se tiene que acceder continuamente a los dos bancos de registros.

- PUERTOS DE ENTRADA/SALIDA:

El microcontrolador usado en este proyecto, dispone de dos puertos denominados A y B. Las líneas de estos puertos se pueden programar individualmente como entradas o como salidas. Debido a lo reducido del encapsulado estos pines de entrada y salida pueden compartir funcionalidad.

Puerto A: El puerto A tiene 5 líneas denominas RA4:RA0, estas líneas se pueden configurar individualmente como entrada o salida mediante el registro TRISA que se encuentra en el banco 1. El bit '1' indica que se utilizará como entrada y el bit '0' como salida.

Puerto B : El puerto B tiene 8 líneas (RB7:RB0), al igual que el puerto A se pueden configurar sus líneas, pero esta vez a través del registro TRISB, y con el mismo método configuración. La patilla RB0 del puerto, tiene una doble funcionalidad y es que sirve como petición de interrupción externa. Esta interrupción por la patilla RB0 será usada en este proyecto como una de las medidas de seguridad para evitar el choque del robot.

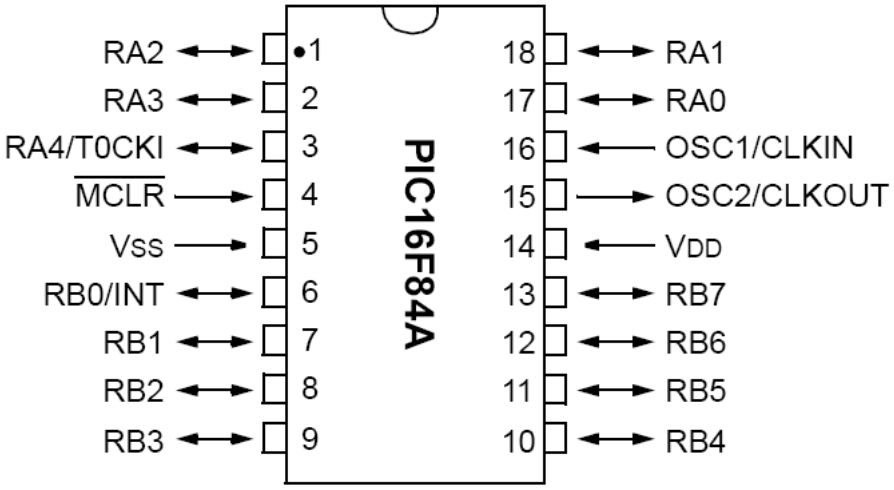


Ilustración 19: Esquema encapsulado PIC16F84A

- **CONTROLADORA MSX84:**

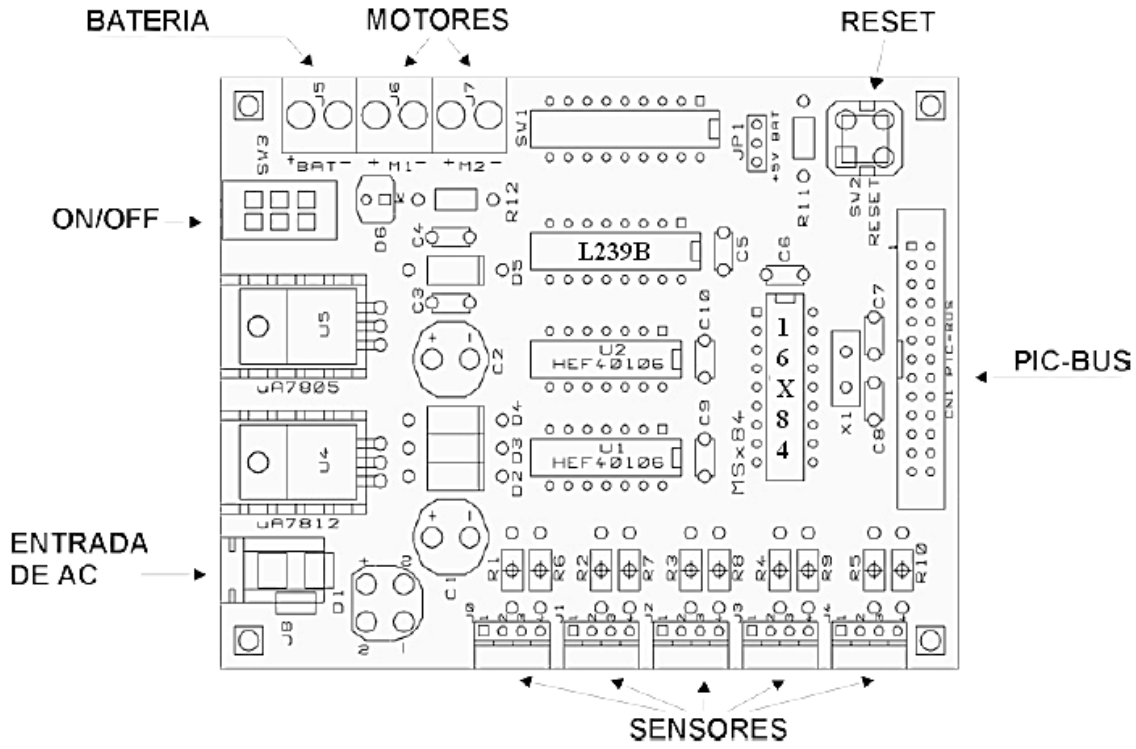


Ilustración 20: Esquema Controladora MSx84

De entre las características más notables de la placa MSx84 de Microsystems Engineering cabría citar las siguientes:

- Tarjeta autónoma de bajo coste y dimensiones reducidas (80 x 100 mm.).
- Alimentación de 15VAC mediante transformador de red o bien mediante baterías de 6, 9 o 12VDC, pudiendo en este último caso utilizarse una batería de 12VDC de plomo recargable.
- Circuitos de rectificación, filtrado y estabilización de alimentación incluidos en la tarjeta.
- Incluye circuito de carga para baterías de 12VDC de plomo ácido.
- Zócalo para microcontrolador PIC 16x84, circuito oscilador y pulsador de RESET incluido en la tarjeta.

Señal	PIN	Señal	PIN	Señal	PIN	Señal	PIN
RB0	1	Sensor 0	18	RA0	6	IN1/IN2	13
RB1	2	Sensor 1	17	RA1	7	IN4/IN3	12
RB2	3	Sensor 2	16	RA2	8	EN1	11
RB3	4	Sensor 3	15	RA3	9	EN2	10
		Sensor 4	14	RA4	5		

Tabla 1: Zócalo para PIC16F84A

- Conectores y circuitos para adaptación de hasta 5 sensores o dispositivos de entrada.

Conector	16x84
J0	RB0
J1	RB1
J2	RB2
J3	RB3
J4	RA4

Tabla 2: Conectores de sensores MSx84/PIC16F84A

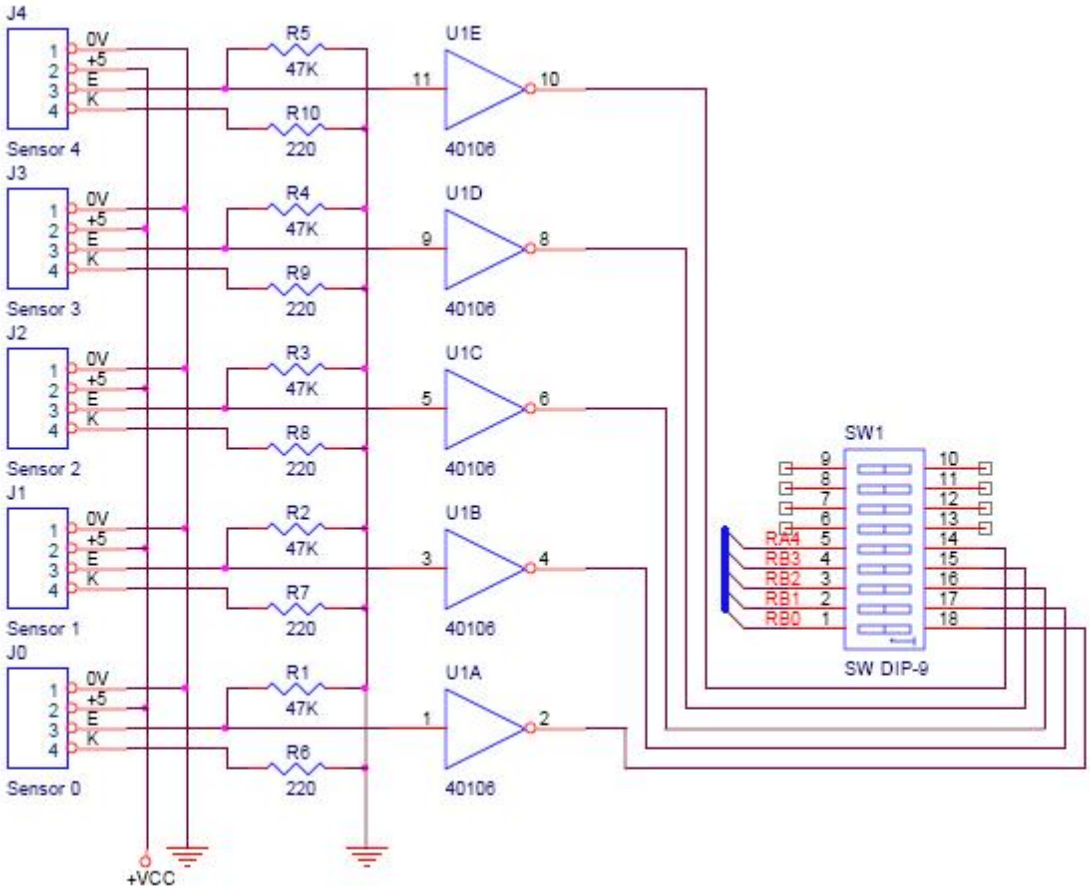


Ilustración 21: Entrada sensores controladora MSx84

- Driver para el accionamiento de dos motores DC o uno paso a paso (PAP).
- Interruptor ON/OFF y bornas para la conexión tanto de los motores como para la batería externa.
- Mediante un jumper se puede seleccionar la tensión aplicada a los motores. Esta puede ser de +5VDC o bien de +12VDC o la tensión de la batería externa.
- Conjunto de 9 DIP-SWITCH que permiten habilitar o no cada uno de los sensores de entrada así como las salidas a los motores. De esta forma se puede

dejar libres aquellas líneas del PIC asociadas a los sensores y motores que no se vayan a utilizar.

- Conector de expansión PIC-BUS compatible con la tarjeta µPIC Trainer de Microsystems Engineering. A través de él se obtienen todas las señales del PIC, pudiéndose de esta forma modificar o añadir nuevos periféricos.

- **MAX232:**

En el Mercado hay muchos circuitos integrados que permiten la conversión entre niveles ttl y niveles Rs232, Entre ellos destaca el Transceptor MAX232, fabricado por Dallas Semiconductor-MAXIM.

El MAX232 convierte los niveles RS232 (cerca de +12 y -12 V) a voltajes TTL (0 a +5 V) y viceversa sin requerir nada más que una fuente de +5V. El chip contiende dos drivers TTL -> RS232 y dos RS232 -> TTL.

Necesita cuatro condensadores externos de unos pocos microfaradios para generar el voltaje RS232 internamente tal como se muestra en la figura 22.

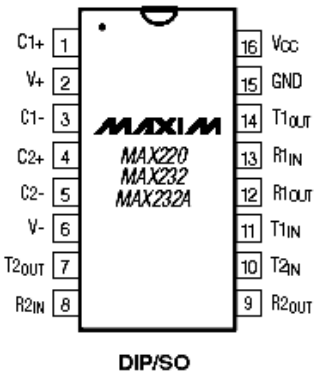


Ilustración 22: Encapsulado MAX232

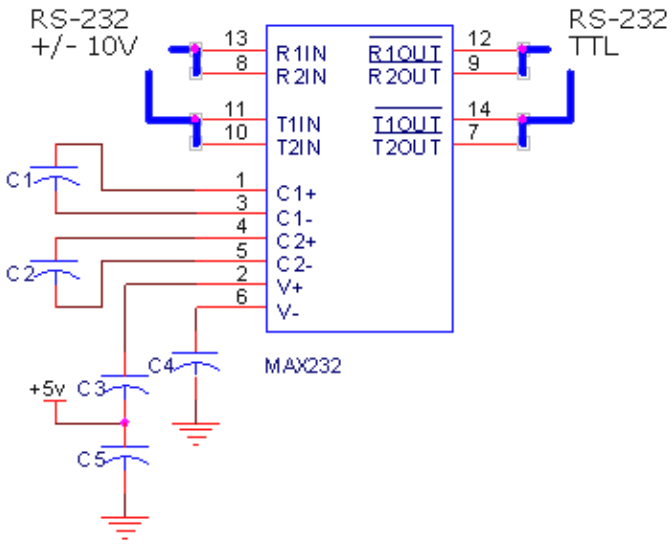


Ilustración 23: Esquema eléctrico MAX232

1.3.1.1.2 Sensores telemétricos

Los sensores telemétricos usados en el proyecto son dispositivos medidores de distancia. Estos sensores dan la distancia desde su posición hasta el primer objeto con el que se encuentre. Para la recogida de distancias se dispone del sensor de ultrasonidos SRF05.

Como habíamos comentado antes, el sensor SRF05 es una actualización del SRF04, al que añade nuevas funciones y características. El rango de distancias aumenta a 4 metros, aunque en este caso se va a limitar a una distancia máxima de 250cm, para que no haya un desborde y se pueda facilitar su uso. Puede usar dos pines o un solo pin para controlar el sensor y hacer la lectura de medición. Se manda un impulso de inicio de lectura y seguidamente se pone en modo de entrada. A continuación mide la distancia del pulso, que es proporcional a la distancia medida. Se detallan los dos métodos, este sensor tiene dos modos de trabajo:

1.- La señal de inicio de medición y para el retorno del eco son independientes. Para estar en este modo basta con no conectar el pin de modo.

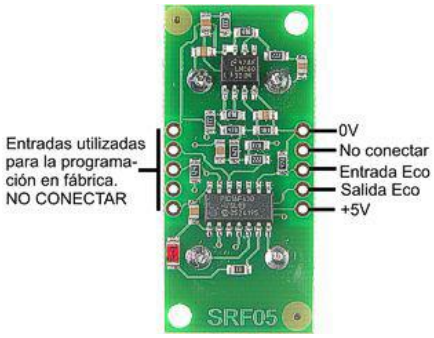


Ilustración 24: Sensor SRF05 modo 1

2.- Modo 2: Pin único para activación y eco. Se conecta el pin de modo a 0v. Cuando se dispare la señal de activación se dispone de 700 µS para ponerlo a medición de impulsos.

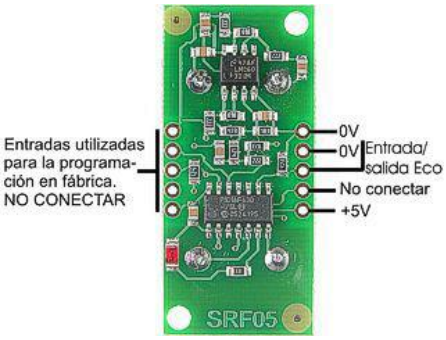


Ilustración 25: Sensor SRF05 modo 2

El Modo 1 ha sido el elegido, este modo da simplicidad al programa, ya que no es necesario cambiar la configuración del pin de Eco/Disparo cada vez que se necesite hacer una medición, como ocurre con el Modo 2. Por el contrario tiene como desventaja que usa un pin más que con el Modo 2.

Para el cálculo de distancias se debe dar un impulso de al menos 10µS. para la activación de comienzo de medición. El SRF05 emite una ráfaga de 8 ciclos a 40Khz con nivel lógico alto de eco a '1' (la línea de activación en el modo 2). Entonces el sensor se pone en escucha del eco, cuando llega éste se baja la línea de eco a 0. si no llega la señal de eco entonces a los 30µS se baja la línea de eco. Una vez que se sabe el ancho de del impulso del eco en µS, se puede obtener la distancia al objeto detectado. Dividiendo la medición de vuelo entre 58 se obtiene la solución en cm. mientras que si se divide entre 148 se obtiene en pulgadas.

Se debe esperar 50 mS entre dos mediciones consecutivas.

El cálculo del valor a cargar en el Timer 0 para conseguir una interrupción cada 60µS se detalla a continuación:

Según la especificación del sensor SRF05 se puede obtener la distancia recogida por el sensor ya que el SRF05 proporciona un pulso de eco proporcional a la distancia. Si el ancho del pulso se mide en µS, el resultado se debe dividir entre 58 para saber el equivalente en centímetros µS /58=cm.

Entonces es fácil obtener que cada 58 µS se va a aumentar la distancia hacia el objeto de 1cm.

Ahora la intención es reflejar en el código esta situación, de manera que el cada vez que el Timer0 del microcontrolador recorra el tiempo de 60 µS se produzca una interrupción en la cual se aumente un registro que lleve la cuenta de la distancia al objeto.

Según la siguiente fórmula se calculará el valor con el cual hay que cargar el TMR0 para que se produzca una interrupción cada 60 µS.

$$256 - Valor_{TMR0} = temporización * f_{osc} / (4 * Prescaler)$$

Donde:

$$Temporización = 60\ \mu S$$
$$f_{osc} = 4MHz$$
$$Prescaler = 2$$

Se obtiene:

$$Valor_{TMR0} = 226$$

Estos datos son teóricos, por lo que son los primeros que se llevaron a la práctica. Sería necesaria la calibración de los sensores con la ayuda del entrenador.

A continuación se adjunta el diagrama de radiación, en el que se puede observar que es demasiado ancho, es decir, no es muy directivo. De ahí se puede destacar que algunos de las distancias recogidas por el sensor no va a ser las reales ya que habrá una desviación y detectaría alguno de los objetos situados en la proximidad de los 0º que sería lo ideal.

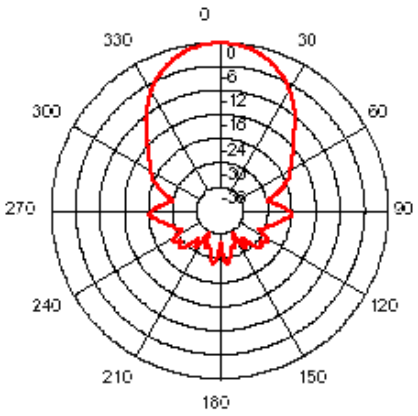


Ilustración 26: Forma Haz SRF05

Los siguientes sensores telemétricos no se utilizarán en este proyecto, aun así, se describirán brevemente ya que se encuentran instalados en el robot.

- MSE-S135 Sensor detector de obstáculos: Este sensor es un detector IR de obstáculos, que detecta objetos. El dispositivo consiste en un fotodiodo el cual emite un haz de luz a una frecuencia de 7.7 kHz. En cuanto a su funcionamiento el nivel lógico es alto si no encuentra nada, donde aparecerá un flanco de bajada (nivel lógico bajo) en el caso que encuentre un objeto en su proximidad.

- Bumper electromagnético: Los bumpers se pueden usar como medio de detector de objetos. Es un mecanismo tipo pulsador, muy sencillo. El funcionamiento de un bumper es similar a un conmutador simple. Tiene un estado inicial en el cual estará abierto, de manera que cuando se accione será cuando choque con un objeto y se cierre el circuito.

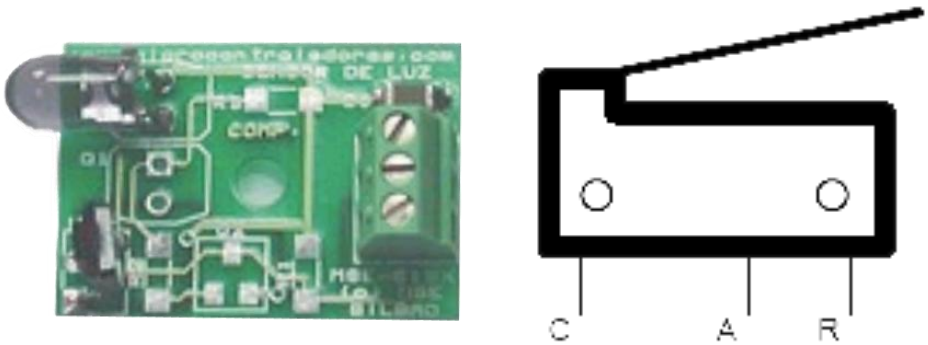


Ilustración 27: MSE-S135 y Bumper

1.3.1.1.3 Motores

Se utilizarán dos servomotores conectados a la placa MSx84. En concreto utilizamos dos servomotores marca HITEC modelo HS422, los cuales han sido modificados para comportarse como motores DC.

Las características técnicas son:

Servo Hitec HS422	
Sistema de Control	Control por Anchura de Pulso. 1,5 ms al centro
Tensión de funcionamiento	4,8V a 6 V
Velocidad a 6V	0,16 Seg /60 grados sin carga
Fuerza a 6V	4,1 Kg · cm
Corriente en reposo	8 mA
Corriente en funcionamiento	150 mA sin carga
Corriente Máxima	1100 mA
Zona Neutra	8 µsec
Rango Trabajo	1100 a 1900 µsec
Dimensiones	40,6 x 19,8 x 36,6 mm
Peso	45,5 g
Rodamiento Principal	Metálico
Engranajes	Plástico
Longitud del cable	300 mm

Tabla 3: Características Servo Hitec HS422

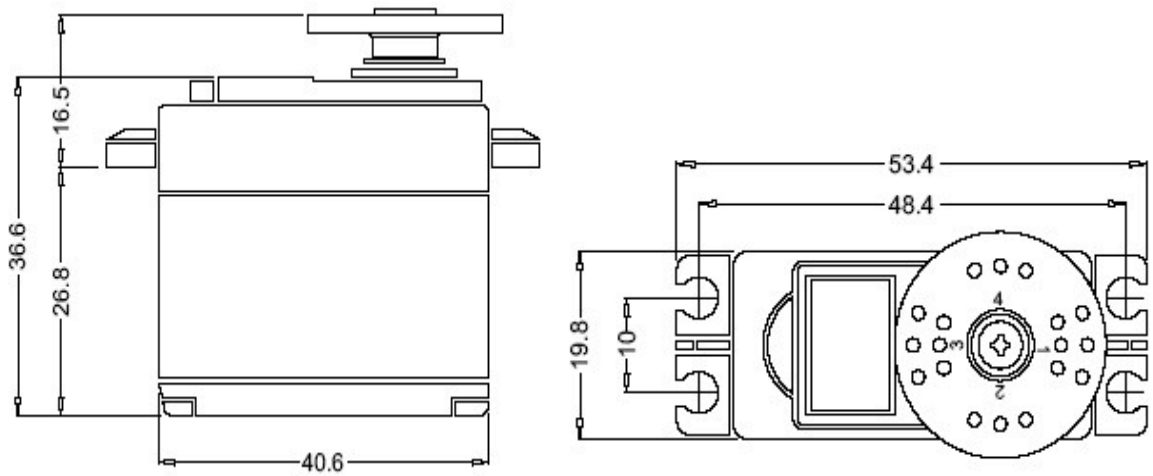


Ilustración 28: Dimensiones servomotor Hitec HS422

- **Driver L239B:**

En la controladora MSx84 encontramos instalado el driver L239B. Este driver es capaz de dar una corriente de salida de 1A por canal, mientras que cada canal es controlado con tensiones TTL.

El driver L239B está compuesto de 16 pines los cuales se detallan a continuación:

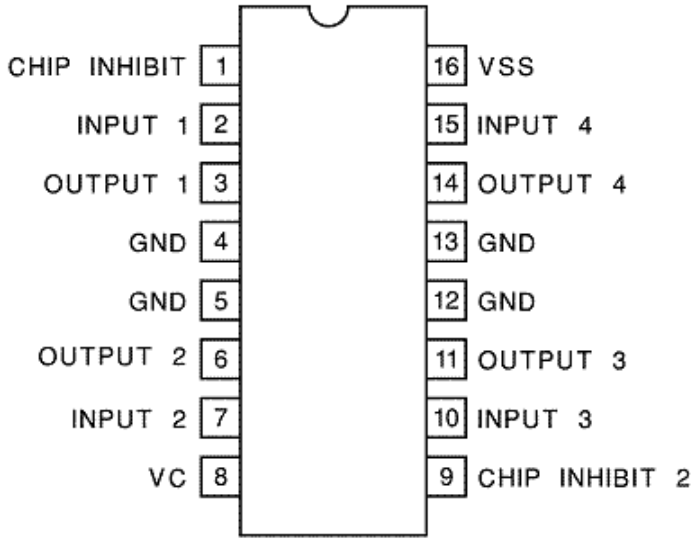


Ilustración 29: Encapsulado driver L239B

Pin	Descripción
1	Habilita los canales 1, 2
2	Entrada del canal 1
3	Salida del canal 1
4, 5	Masa de alimentación
6	Salida del canal 2
7	Entrada del canal 2
8	Alimentación de las cargas
9	Habilita los canales 3 y 4
10	Entrada del canal 3
11	Salida del canal 3
12, 13	Masa de alimentación
14	Salida del canal 4
15	Entrada del canal 5
16	Alimentación del chip

Tabla 4: Pines del driver L239B

1.3.1.1.4 Comunicación RF

Para la comunicación por radio frecuencia se utilizará un módulo llamado radiomodem. Los radiomodems se utilizan para transmitir gran cantidad de información, conectados en puerto serie.

Este es un módulo para conectar dos dispositivos inalámbricamente, concretamente para este proyecto se utilizará para conectar el microcontrolador PIC16F84A que encontramos en el robot y un sistema PIC-WEB.

En este caso la comunicación es bidireccional, es decir, tanto el microcontrolador como el PIC-WEB pueden transmitir/recibir. La transmisión es semi-dúplex. Quiere decir que sólo puede enviar o el PIC-WEB o robot en un mismo instante.

La información que se transmite es de tipo byte. En este caso la información va a ser los resultados de las distancias calculadas por los sensores de distancias, como el control de movimientos del robot. Existe un número máximo de bytes por paquete, esto es, que si se envían más bytes de los permitidos por paquete, los sobrantes serán descartados.

El protocolo de comunicación solo se ocupa de la transmisión y recepción de datos, cuando los bytes son recibidos por el módulo de radio perteneciente al robot. Si una vez llegados al robot no son procesados por el microcontrolador, estos datos se habrán perdido.

La técnica de control de errores es CRC, por lo que cuando haya errores se contempla la posibilidad de retransmisión de paquetes.

También existe un tiempo de tratamiento de búfer, esto implica que si el búfer se llena los datos que no quepan en el búfer serán descartados. Se deberá procurar una pausa del doble de duración del tiempo de transmisión de un byte, por ejemplo a 19200 baudios habrá que poner un retardo de 1,04 ms.

La velocidad del microcontrolador del robot y el PIC-WEB no tiene la necesidad de ser la misma, lo normal es de 19200 baudios, que no tiene un tiempo excesivo de retardo.

Se obtiene una eficiencia en la recepción de datos del 99,9%, pero según se aumenta la distancia entre PIC-WEB y microcontrolador la eficiencia va disminuyendo hasta un 0% que es una zona con cobertura nula.

Es necesario un tiempo máximo de espera de datos, para que no se queden en un tiempo infinito de espera.

El tiempo que tarda en llegar un byte, entre PIC-WEB y robot, se detalla a continuación: el tiempo total, es equivalente al tiempo de transmisión de un byte, más el tiempo de propagación por el aire, más un tiempo que tarda en recibir y ser procesada la información recibida, más el tiempo que tarda el módulo en hacer el control de errores. Y si fuera necesaria una retransmisión de un paquete, este paquete tardaría el doble de un tiempo normal.

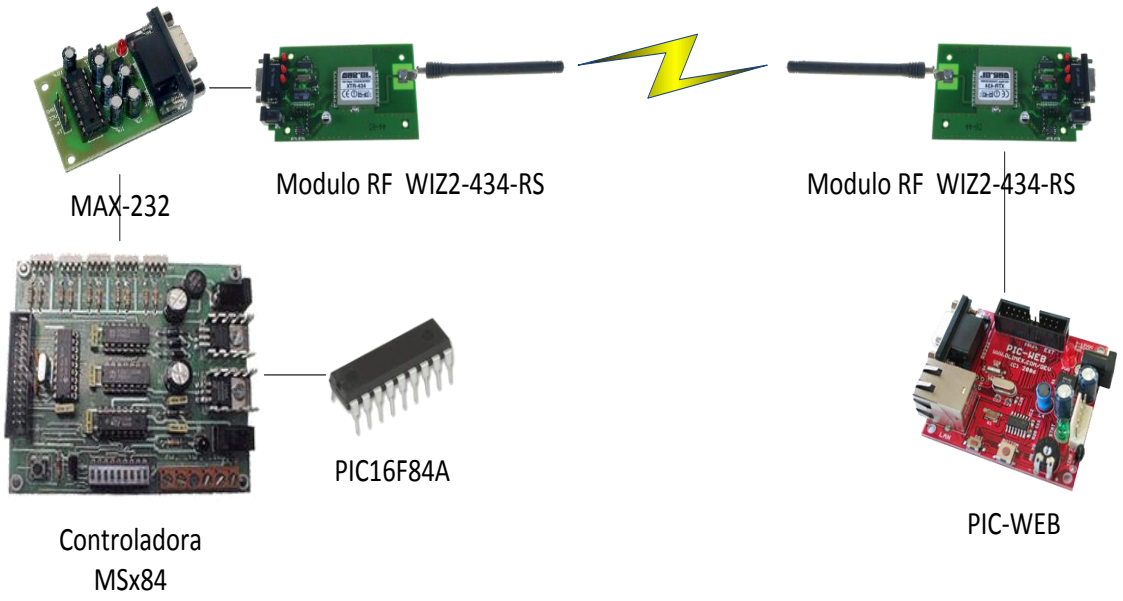


Ilustración 30: Esquema comunicación RF

- **Módulo transceptor multipunto WIZ-434-RS:**

Se trata de módulos transceptores inteligentes para la transferencia de datos digitales vía radio frecuencia entre múltiples unidades WIZ2-434 (multipunto). Los datos se transmiten en paquetes o tramas de longitud variable entre 1 y 96 bytes.

Los datos transferidos a la unidad remota están libre de errores ya que, al llegar, son verificados y confirmados por el receptor.

Entre las características más relevantes cabe citar las siguientes:

- Conexión serie half-duplex con protocolo RS-232 y con un rango de velocidad configurable (9600, 19200, 57600 y 115200 baudios).
- Tamaño del paquete de datos de 1 a 96 bytes.
- Verificación/Validación de los datos recibidos.
- Conexión por radio frecuencia (RF) en la banda de 433.92MHz
- Modulación 2FSK.
- Alcance de 40m aprox. en interiores y de 150m aprox. en exteriores
- Led' s de estado (Power ON, TX, RX)
- Cumple con las regulaciones ETSI 300-220
- Alimentación de 9 -15Vcc

La figura 28 muestra la disposición de los componentes mas importantes de que constan los módulos WIZ2-434-RS y que se detallan en la siguiente tabla:

Nº	DESCRIPCION
1	Conector de alimentación externa. Se aplica una tensión de 9-15Vdc con positivo al centro
2	Conector DB9 hembra de 9 pines para la conexión con un canal serie (p.e. un PC)
3	Jumpers de configuración. Con ellos se selecciona la velocidad de transferencia entre el módulo y el canal serie y también se selecciona si el módulo se encuentra en modo de configuración o en modo transceptor de datos
4	Conector ICP. Este conector se emplea en el proceso de fabricación del módulo para programar el microcontrolador que posee. No debe ser empleado por el usuario.
5	Circuito de estabilización de +5Vcc
6	Microcontrolador. Este se graba en fábrica y se encarga de todas las tareas de control del módulo así como de la transferencia de datos entre el módulo y el canal serie y también del enlace por radio frecuencia
7	Led que indica la presencia de tensión de alimentación
8	Led que indica la transmisión de datos
9	Led que indica la recepción de datos

Tabla 5: Componentes del módulo WIZ2-434-RS

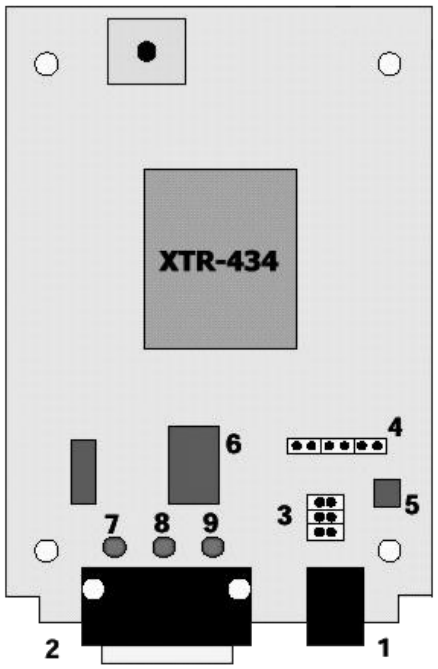


Ilustración 31: Distribución de los componentes del módulo WIZ2-434-RS

Proceso de comunicación:

Cuando uno de estos módulos desea transmitir un paquete de datos, este será codificado por el módulo antes de ser enviado. La longitud máxima de paquete será de 96 bytes. Una vez codificado el paquete se envía vía radio frecuencia.

Un paquete que se ha enviado será recogido por un módulo receptor. El paquete recibido es decodificado y tratado por el módulo receptor.

Cuando es el módulo principal quien quiere enviar un paquete a uno de los otros módulos deberá incluir el byte de dirección del módulo correspondiente. En cambio si la comunicación es hacia el módulo principal, el cual tiene como dirección la 0x00 no se debe incluir el byte de esta.

La velocidad de transmisión es configurada mediante una serie jumpers, los cuales indicarán qué velocidad será la de trabajo.

Velocidad	J1	J2
9600	Abierto	Abierto
19200	Abierto	Cerrado
57600	Cerrado	Abierto
115200	Cerrado	Cerrado

Tabla 6: Velocidad de transmisión según jumpers WIZ2-434-RS

1.3.1.2 Recopilar información del sistema PIC-WEB

Como hemos comentado anteriormente el sistema PIC-WEB es una plataforma implementada con el microcontrolador PIC18F452**,** la cual incluye el stack de código abierto TCP/IP AN833 de Microchip. Gracias a ello, este sistema no sólo será capaz de tener acceso a internet sino que podrá actuar como un servidor web.

Las características más importantes de esta plataforma se enumeran a continuación:

- Microcontrolador PIC18F452.
- Controlador de Ethernet ENC28J60.
- 1Mbit de memoria en placa para alojamiento de páginas web.
- Conector ICSP/ICD para programación con PIC-MCP, PIC-MCP-USB y programación y depuración con PIC-ICD2 y PIC-ICD2-POCKET.
- Botón de Reset.
- Botón para eventos de usuario.
- Potenciómetro de ajuste analógico.
- Termistor para la monitorización de la temperatura.
- Conector y driver RS232.
- Servidor web complete y stack TCP-IP de código abierto de Microchip.

- Conector para enchufar fuente de alimentación +5 VDC.
- Regulador de voltaje +3.3V y condensadores de filtrado.
- LED de estado
- Cabecera de extensión para conectar a otras placas.
- Dimensiones: 60x65 mm.

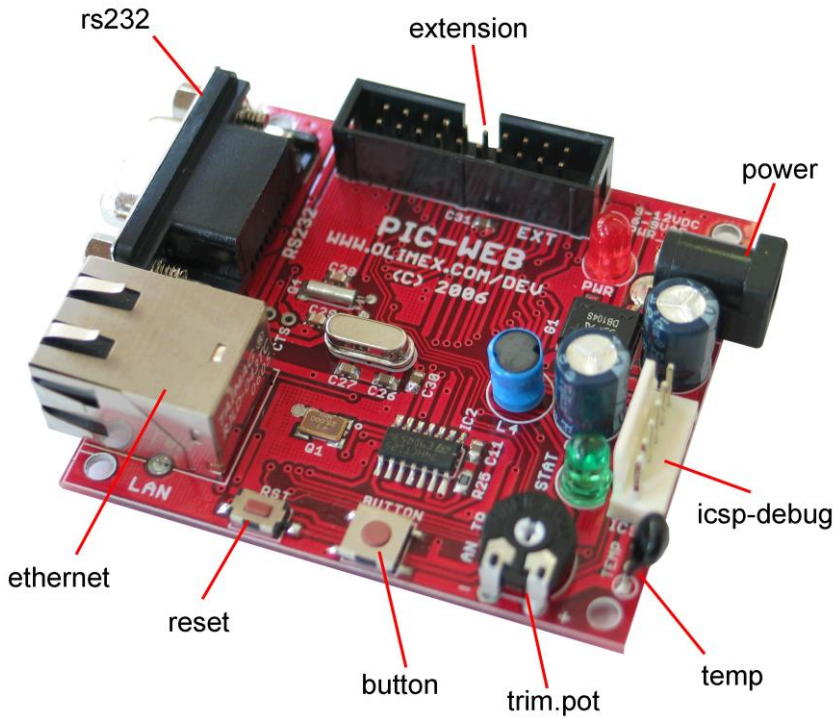


Ilustración 32: Distribución de los componentes del sistema PIC-WEB

A continuación se describirán los distintos componentes y sus características más importantes:

- **PIC18F452:**

CARACTERÍSTICAS GENERALES PIC 18F452:

- Arquitectura RISC (*Reduced Instruction Set Computer*).
- Juego de instrucciones reducido para ejecución rápida.
- Oscilador hasta 40 MHz / 10 MIPs (*Million Instructions Per second*).
- Optimizado para compilación desde lenguaje C.
- Código compatible con la familia 16 y 17 de los PIC
- Reloj que puede por trabajar encima de10 MIPs.
- Cristales de 4 Mhz a 10 Mhz utilizando un multiplicador de frecuencia PLL.
- Instrucciones de 16 bits con bus de datos de 8 bits.
- Prioridad de interrupciones

- Multiplicador hardware de 8 x 8 que funciona en un solo ciclo de máquina.
- Tres pines para manejo de interrupciones externas.
- Manejo de corriente niveles de de 25 mA. en modo fuente y sumidero
- Timer 1 de 16 bits, Timer 2 de 8 bits.
- Timer 3, (no lo posee la gama media), de 16 bits (65535 conteos).
- Dos módulos de Captura/Comparación/PWM.
- Modulo de comunicación serial con soporte para RS-485 y RS-232

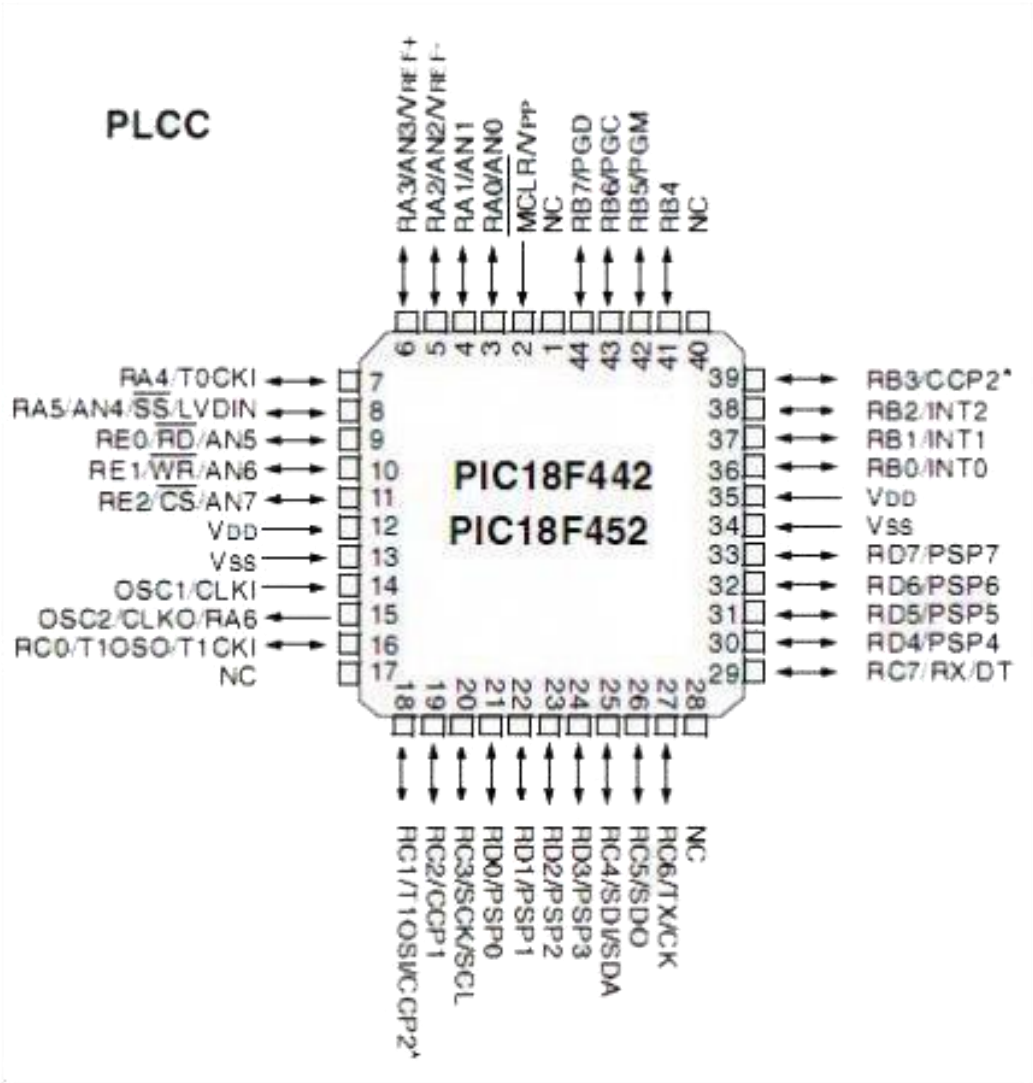


Ilustración 33: Encapsulado PIC18F452

OSCILADOR MODO XT:

En este proyecto se utiliza el oscilador en modo XT como se muestra a continuación:

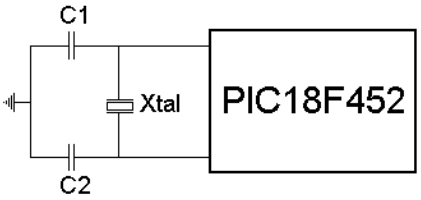


Ilustración 34: Oscilador XT PIC18F452

Una capacitancia elevada, incrementa la estabilidad del oscilador, pero también incrementa tiempos de inicio del oscilador los interno.

FUENTES DE RESET:

EL PIC18F452 posee las siguientes Fuentes de Reset:

- Power-on Reset (POR)
- MCLR Reset Durante el funcionamiento normal
- MCLR Reset Durante el modo SLEEP
- WDT Reset durante operación normal
- Programmable Brown-out Reset (BOR)
- Instrucción de RESET.
- Reset debido al llenado del Stack.
- Reset debido al vaciado del Stack.

PUERTOS:

Los puertos de los PIC 18f452 en general constan de 3 registros para su operación.

- El registro TRIS, el cual controla la dirección de funcionamiento del puerto.
- El registro PORT que lee los niveles de entrada en el puerto.
- El registro LAT cual es el LATCH salida del puerto.

REGISTROS ASOCIADOS CON EL PUERTO A

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTA	—	RA6	RA5	RA4	RA3	RA2	RA1	RA0
LATA	—	LATA						
TRISA	—	PORTA						
ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0

Ilustración 35: Registros asociados al puerto A - PIC18F452

ORGANIZACIÓN DE LA MEMORIA:

PIC18F452 posee 32 K Bytes de memoria FLASH de programa, agrupados de a 2 MBytes, con el fin de contener instrucciones complejas. Por lo tanto este dispositivo puede almacenar 16mil instrucciones simples.

El vector de RESET se halla en la dirección 0000h y el de interrupciones en las posiciones 0008h y 0018h.

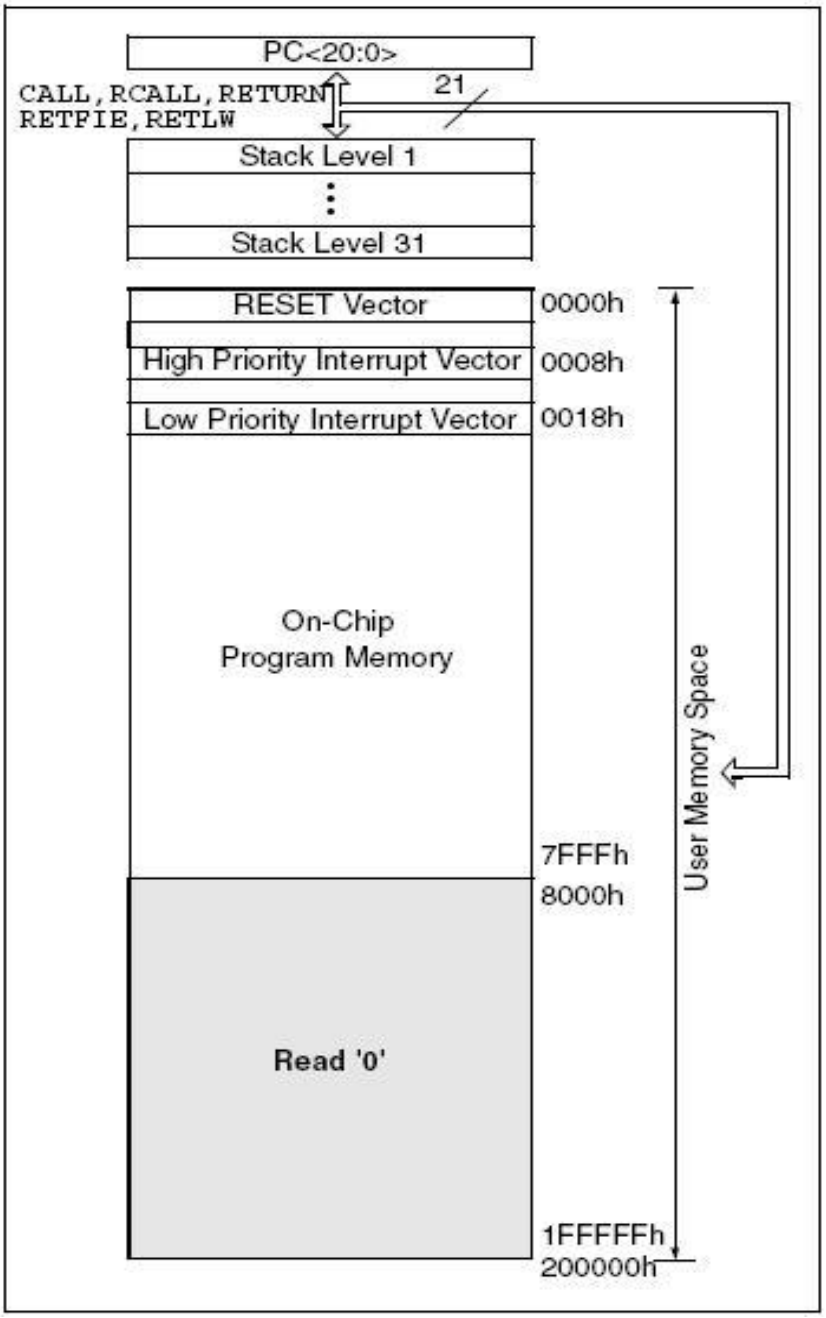


Ilustración 36: Organización memoria PIC18F452

INTERRUPCIONES:

Los microcontroladores de gama alta poseen niveles de interrupción. El vector de alto nivel de interrupciones se halla en la posición 0x08h y el de baja prioridad en la posición 0x18h. Una interrupción de alta prioridad, interrumpe la ejecución de una de baja prioridad.

MEMORIA EEPROM DE DATOS:

La memoria EEPROM, no puede ser direccionada normalmente, para acceder a ella se hace a través de unos registros especiales

- EECON1 (Registro de configuración)
- EECON2 (Registro de configuración)
- EEDATA (Registro de transferencia de datos)
- EEADR (Registro de direccionamiento)

MODULO DE COMUNICACIÓN SERIAL USART:

El módulo de comunicación serial puede ser configurado de las siguientes maneras:

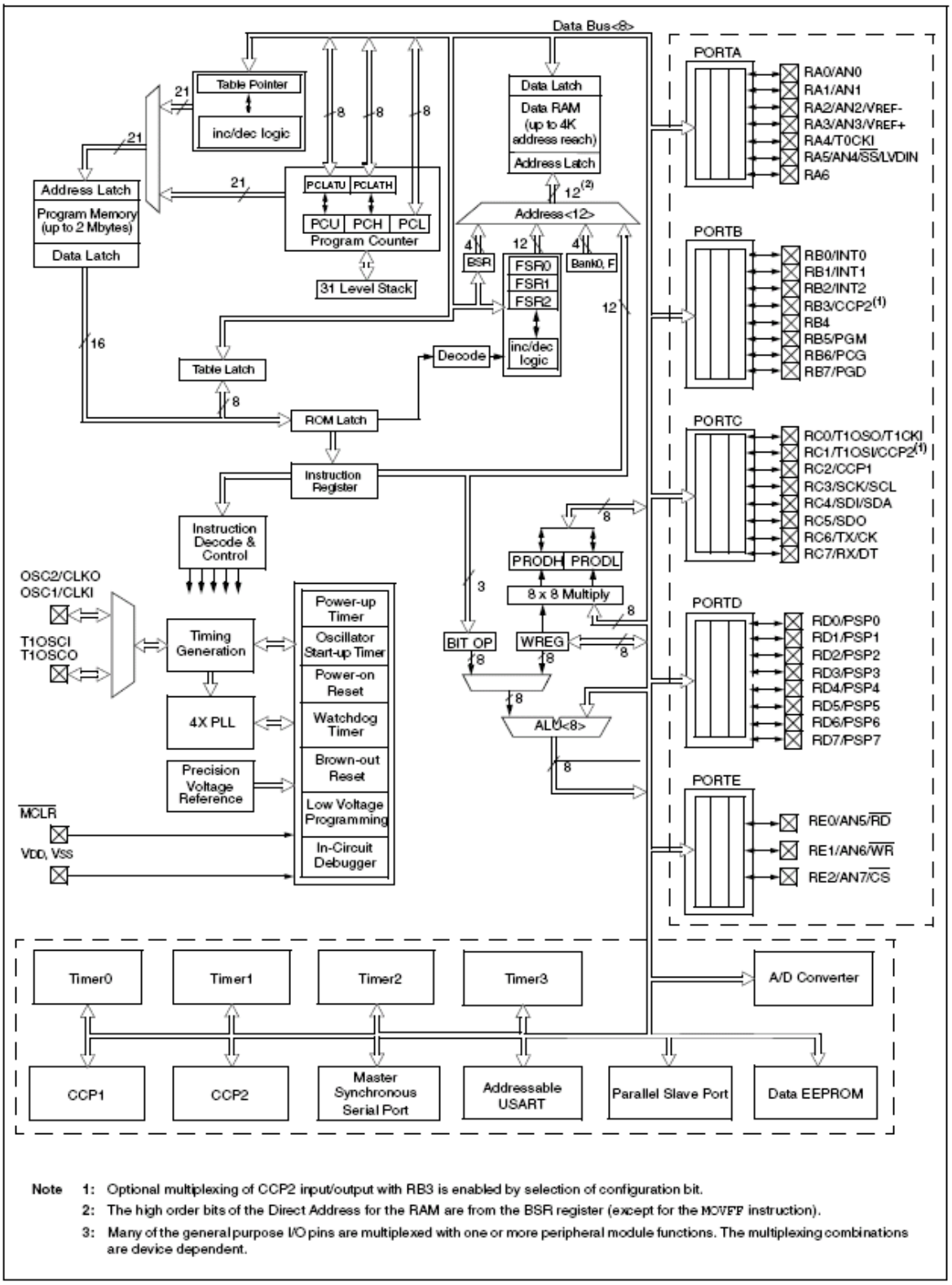
- Sistema asíncrono full duplex.
- Sistema sincrónico half-duplex (Maestro).
- Sistema sincrónico half-duplex (Esclavo).
- El módulo USART cuenta con dos registros de configuración.
- El registro de control de transmisión TXSTA.
- El registro de control de recepción RCSTA.

SEÑALES DE ENTRADA Y SALIDA DE PIC18F452:

PIN	TIPO	NOMBRE	DESCRIPCION
RB0	ENT	ENTRADA 6	Botón de Activar Alarma
RB1	ENT	ENTRADA 5	Botón de Apagar Alarma
RB2	ENT	ENTRADA 4	Sensor puerta Piloto
RB3	SAL	SALIDA 2	Control de Encendido
RB4	SAL	SALIDA 1	Activador de sirena y Bloque del auto
RB5	ENT	ENTRADA 3	Sensor de Movimiento
RB6	ENT	ENTRADA 2	Sensor puerta
RB7	ENT	ENTRADA 1	Sensor puerta

Tabla 7: Señales de entrada y salida de PIC18F452

DIAGRAMA DE BLOQUES PIC18F452:



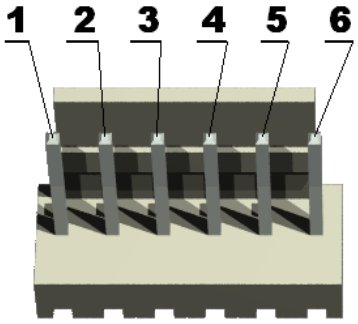
Note 1: Optional multiplexing of CCP2 input/output with RB3 is enabled by selection of configuration bit.
2: The high order bits of the Direct Address for the RAM are from the BSR register (except for the MOVFF instruction).
3: Many of the general purpose I/O pins are multiplexed with one or more peripheral module functions. The multiplexing combinations are device dependent.

Ilustración 37: Diagrama de bloques PIC18F452

- **Conector ICSP:**

El conector ICSP se utilizará principalmente para la programación del PIC18F452 integrado en el sistema PIC-WEB.

- PGD: I/O / Datos de programa. Comunicación de datos para la programación.
- PGC: Entrada / Reloj de programa. Reloj usado para la transferencia de datos.
- PGM: Entrada / Programación habilitada.

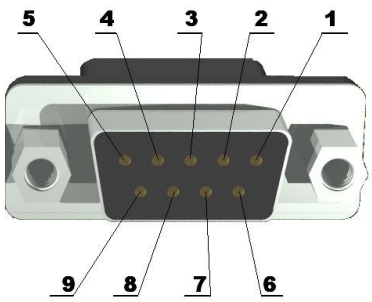


PIN	Nombre Señal
1	RST
2	+5V
3	GND
4	PGD
5	PGC
6	PGM

Tabla 8: Pines conector ICSP del sistema PIC-WEB
Ilustración 38: Pines conector ICSP del sistema PIC-WEB

- **Conector RS232:**

- TXD: Salida / Transmisión de datos. Esta es la salida asíncrona de datos en serie (RS232) para el
registro de desplazamiento en el controlador de UART.
- RXD: Entrada / Recepción de datos. Esta es la salida asíncrona de datos en serie (RS232) para el
registro de desplazamiento en el controlador de UART.
- RTS: Pin / Solicitud para enviar. (Request to send). Este pin no está conectado al PIC18F452.
- CTS: Pin / Libre para enviar. (Clear to send). Este pin no está conectado al PIC18F452.



PIN	Nombre Señal
1	NC
2	TXD
3	RXD
4	NC
5	GND
6	NC
7	RTS
8	CTS
9	NC

Tabla 9: Pines conector RS232 del sistema PIC-WEB
Ilustración 39: Pines conector RS232 del sistema PIC-WEB

- **Conector EXT:**

El conector EXT se utilizará principalmente para conectar el sistema PIC-WEB a cualquier otro dispositivo externo.

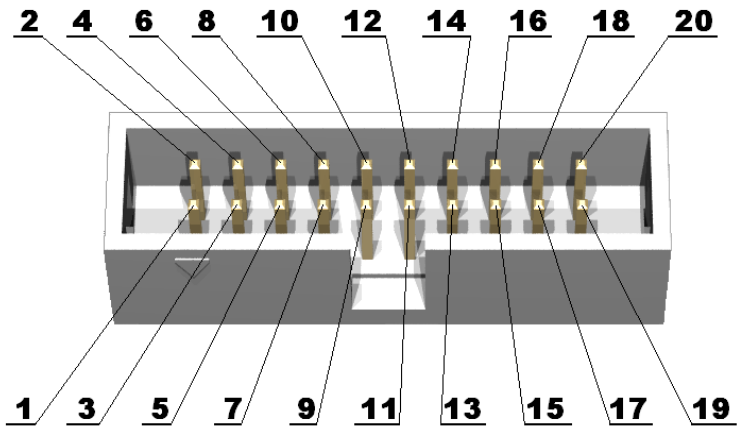


Ilustración 40: Pines conector ICSP del sistema PIC-WEB

PIN	Nombre Señal	PIN	Nombre Señal
1	RA2/AN2/VREF-	2	RA3/AN3/VREF+
3	RA4/T0CKl	4	RA5/AN4/#SS/LVDIN
5	RE0/RD#/AN5	6	RE1/WR#/AN6
7	RE2/CS#/AN7	8	RC2/CCP1
9	RD0/PSP0	10	RD1/PSP1
11	RD2/PSP2	12	RD3/PSP3
13	RD4/PSP4	14	RD6/PSP6
15	RD7/PSP7	16	RST
17	+5V	18	+5V
19	GND	20	VIN

Tabla 10: Pines conector ICSP del sistema PIC-WEB

- **Conector Ethernet:**

El conector Ethernet se utilizará principalmente para la conexión del sistema PIC-WEB a una red con acceso a internet.

- TPOUT-: Salida / Salida diferencial de la señal.
- TPOUT+: Salida / Salida diferencial de la señal.
- TPIN-: Entrada / Entrada diferencial de la señal.
- TPIN+: Entrada / Entrada diferencial de la señal.

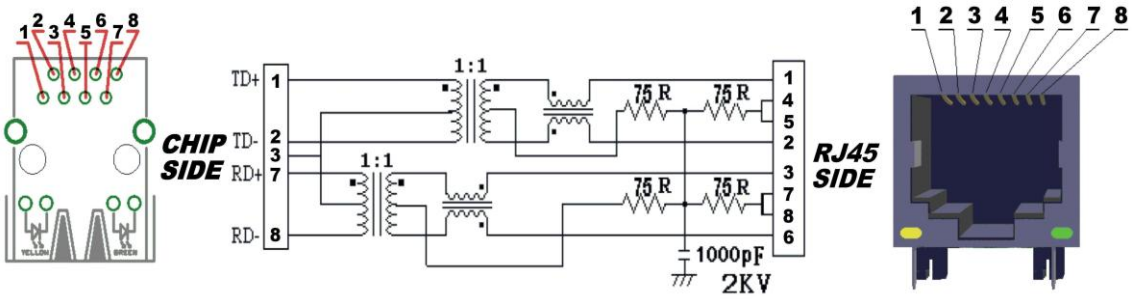


Ilustración 41: Pines conector Ethernet del sistema PIC-WEB

PIN	Nombre Señal	PIN	Nombre Señal
1	TPOUT+	5	NC
2	TPOUT-	6	NC
3	+3.3V	7	TPIN+
4	NC	8	TPIN-

LED	Color	Uso
Right	Amarillo	Actividad
Left	Verde	100MBits/s (Half/Full Duplex)

Tabla 11: Pines conector Ethernet del sistema PIC-WEB

- **Dimensiones mecánicas PIC-WEB:**

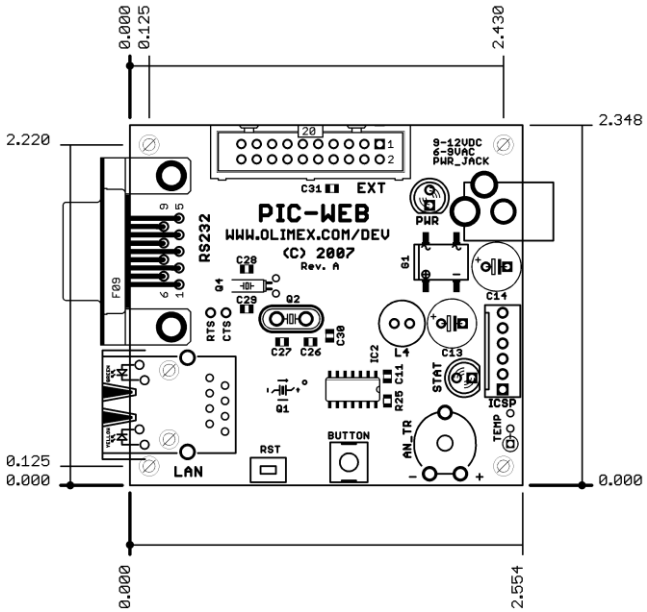


Ilustración 42: Dimensiones mecánicas del sistema PIC-WEB

1.3.1.3 Recopilar información software

En los siguientes subapartados se trata de reflejar toda la información válida adquirida para poder implementar la parte Software del proyecto.

1.3.1.3.1 Programación comunicaciones PIC-WEB / Puertos

La comunicación entre el sistema PIC-WEB y el robot se realiza mediante el puerto serie, ya que los módulos RF se encargarán de la transmisión entre los dos conectores RS232. Por lo que se debe controlar el paso de mensajes entre los dos. Tanto en el PIC-WEB como el robot deben de tener un software que se encargue de enviar/recibir los datos por el puerto serie. Primeramente se comentará la configuración del puerto serie en el lado del PIC-WEB controlado por un microcontrolador.

- **PIC-WEB (Microcontrolador):**

Los pic's 18fxx2 tienen un módulo USART capaz de soportar comunicaciones serie síncrona y asíncrona. Tiene dos pines encargados de esta comunicación, uno de ellos para la transmisión de los bits y otro para la recepción de datos.

INTERFAZ DE COMUNICACIONES SERIE (USART-SCI):

Los bloques que configuran la arquitectura del USART asíncrono se describen a continuación:

- Circuito de muestreo:

  Actúa sobre la patilla RC7/RX/DT que es el que recibe el bit de control (información) y se encarga de hacer un muestreo del valor varias veces y quedarse con el más repetido.

- Generador de baudios (BRG):

  Frecuencia a la que se realiza la transferencia. Esta frecuencia viene normalizada. El valor cargado en el registro SPBRG, al que posteriormente en el cálculo matemático se notará como x, será uno de los parámetros de configuración de la frecuencia en baudios.

  La frecuencia en baudios también depende del segundo bit de registro TXSTA, que es llamado BRGH. Si BRGH es '0' se trata del modo de baja velocidad, pero si BRGH es '1' será de alta velocidad. La fórmula de la frecuencia en baudios será:

$$Frecuencia\ (baudios) = \frac{f_{osc}}{k(x+1)}$$

Donde k = 64 si BRGH es '0',

O si k = 16 si BRGH es '1'.

$$Frecuencia\ (baudios) = \frac{4 \cdot 10^6}{16(12+1)} = 19230$$

En este proyecto que se trata se tiene un cristal de 4Mhz, es decir, f_{osc} = 4Mhz. Además si se quiere una alta velocidad, se tomará BRGH a '1' por lo que k será 16. Para obtener una frecuencia de 19200 baudios:

Ya que X debe ser un número entero se debe aproximar a 12. Lo que introducirá un error.

$$19230 = \frac{4 \cdot 10^6}{16(x+1)} \quad \Rightarrow \quad x = \frac{4 \cdot 10^6}{19200 \cdot 16} - 1 = 12{,}0208$$

$$Error = (19230 - 19200) / 19200 = 0.15625$$

- Transmisor asíncrono:

  El dato que se desea transmitir se coloca en el registro TXREG y a continuación se lleva a un registro de desplazamiento TSR que saca los bits del menos al más significativo. Antes de transmitir los bits se envía un bit de START y después de la transmisión de los bits de información se envía un o dos bits de STOP.

1 bit	8 bits	1 ó 2 bits
INICIO	LSB.....MSB	PARADA

Tabla 12: Trama de transmisión

Una vez enviado el bit de parada, se activa la flag TXIF, por lo que se producirá una interrupción si el bit TXIE (PIE1<4>) estuviera activado. Cuando se escribe un byte en el registro TXREG la flag TXIF se debe poner a '0'. El bit TMRT (TXSTA <1>) está a '1' cuando TSR esté vacío.

- Receptor asíncrono:

  Los datos obtenidos por el puerto serie se van a ir recibiendo bit a bit por la patilla RC7/RX del microcontrolador, pero antes de recibir todos los bits se eliminan los bits de control (bits de comienzo y parada) quedándose sólo con los bits de datos, éstos bits se van a ir introduciendo en el registro RSR. En principio, durante la configuración para recibir datos también se debe de configurar el generador de baudios (registro SPBRG), activar el USART en modo asíncrono y habilitar la recepción de datos continua, para esto habrá que modificar el registro RCSTA. Se tiene la posibilidad de controlar la recepción de datos mediante interrupciones, siempre y cuando se habilite el bit RCIE, de manera que cada vez se complete una recepción la flag RCIF que se encuentra en el registro PIR1 se activará a '1'.

Implementación de una transmisión en el USART:

```
Clrf PORTC ;borra PORTC
Bsf STATUS,RP0 ;banco 1
Bcf STATUS,RP1
Movlw b'10111111' ;RC7/RX como entrada, RC6/TX
                  ; como salida
Movwf TRISC ;
Bcf STATUS,RP0 ;banco 0
Bsf RCSTA,SPEN ;Activa el USART
Movlw b'00100100' ;Transmisión asíncrona, datos de 8 bits, alta velocidad
```

- **Robot (Microcontrolador):**

Esto anterior no ocurre así para el PIC16F84A que debe realizar la comunicación por el puerto serie mediante rutinas que se encarguen de la transmisión y recepción de datos. Para ello se hace uso de una serie de librerías creadas que soporte esta comunicación como es RS232.INC.

En la comunicación síncrona hace falta una señal de reloj que indique cuando un dato es válido.

En la comunicación asíncrona no hace falta este reloj.

Cuando no se realiza ningún intercambio de datos, la línea del transmisor se encuentra en estado alto '1'. Así que cuando se quiera transmitir algún dato se pondrá una condición de inicio (START) que es colocar un '0' en la línea de transmisión durante un cierto tiempo, seguidamente poner el dato en la línea de manera que se transmita primero el bit de menor peso (LSB) y por último el de mayor peso (MSB). Más tarde se puede o no poner un bit de control de errores de paridad, también se tiene la opción de poner 1 ó 2 bits de parada (STOP), y tras enviar el bit de STOP la línea quedará a nivel alto.

Como según dice la norma que define el RS-232 el nivel lógico '1' se comprende en el intervalo [-5, -10v] y el nivel lógico '0' a [5, 10v], se hace indispensable contar con dispositivos que conviertan niveles lógicos TTL a niveles lógicos RS-232 y viceversa, para ello se usará el módulo MAX-232.

1.3.1.3.2 Comunicación sensores telemétricos según estándares I2C

En cuanto a la comunicación entre el sensor telemétrico SRF05 y el microcontrolador se produce a través del bus I2C. La dirección predeterminada de fábrica es la 0xE0, pero se puede modificar. Hay 16 direcciones distintas por lo que se podrían conectar 16 sensores SRF05 pudiéndose crear así un sonar.

El sensor SRF05 contiene 5 pines. Uno es de NC (no conexión) por lo que se dejan libres los pines de 0v.

Este sensor tiene varios registros. El registro 0 es el registro de comando que sirve para iniciar la medición. El registro 2 es el byte alto de la medición, mientras que el 3 es el bajo. Si el registro de comando se escribe con 0x50 la medición se realiza en pulgadas, si el valor es de 0x51 en cm. y la otra opción es que el valor de 0x52 el valor es la del tiempo del vuelo en µS.

Entonces para iniciar una medición se tendrá que escribir una valor (0x50, 0x51, 0x52), en el registro de comando y esperar el tiempo de medición (10ms) y leer los resultados en los registros 2 y 3.

Mientras se realiza la medición de la distancia, el bus I2C no responde a ninguna petición por parte del controlador. Mientras se produce el cálculo el registro 0 se eleva a 0xFF. Por lo que se puede comprobar la medición de la distancia cuando este registro cambie de valor.

1.3.1.3.3 Programación estadística para representación en el plano de líneas a partir de una nube de puntos

En cuanto a la representación del plano, esto se va a reproducir a partir de unos puntos obtenidos tras la exploración del robot. Estos puntos van a ser aportados al PIC-WEB que a su vez los reenviará al servidor de aplicaciones, donde se interpretarán de forma conveniente.

Se entiende que el robot será manejado por usuario, por lo que también dependerá de él el número de puntos o muestras obtenidas por el robot.

A partir de aquí se ha diseñado un programa para la reconstrucción del plano, este programa es *Reconstrucción.java* el cual está comentado en la sección de *Recepción de medidas vía Internet desde el PIC-WEB en el PC*.

Este programa trata de unir cada punto con sus dos puntos más cercanos. Por lo que se obtiene la distancia de cualquier punto a todos los demás. A partir de aquí se determinarán los dos puntos más cercanos y se unirán con una línea.

No se unirán simplemente los puntos más cercanos, sino que habrá un margen; esto es, sólo se unirán mediante una línea aquellos que su distancia mínima no supere un ese margen. Esto impide que un punto anormal o que esté muy separado de los demás sea unido.

Si se han obtenido menos muestras, se deberá aumentar el margen de unión, para que así queden los menos puntos posibles aislados, a partir de aquí se obtendrá un plano menos real que si se obtuvieran más muestras.

También se ha comentado antes que cada punto se va a unir con dos puntos máximo, esto se ha considerado para no dibujar en el plano líneas las cuales va a ser de poca utilidad.

La mayor o menos realidad o aproximación del plano va a depender siempre del número de muestras obtenidas por el usuario y además, comentar también que esto va a corresponder siempre al usuario.

1.3.2 FASE 2

1.3.2.1 Primera etapa

1.3.2.1.1 Configuración del PIC. Comprobación de medidas procedentes de los sensores

Este trabajo será necesario para la comprobación necesaria de las medidas tomadas por el sensor SRF05, para ello se ha aportado un programa específico el cual es encargado de hacer funcionar el sensor y obtener las distancias las cuales serán visualizadas en una pantalla LCD. El microcontrolador empleado para su puesta en marcha y calibración será el PIC 16F84A.

Para que el sensor comience a coger muestras se debe establecer a nivel alto la patilla de disparo (PORTA:3) durante al menos 10 µS, por lo que se le ha dado un pulso de 40 µS. El programa espera a que por Eco (PORTA:4) llegue un '1'. Cuando esto pase se deberá cargar el valor determinado en el Timer0, y activar las interrupciones por Timer. Una vez se llega a esta situación, el microcontrolador espera llegar un '0' por la patilla de Eco, mientras pase esto cada vez que se desborde el Timer0 se incrementará en valor de la distancia (que estará inicialmente a 0) en 1 cm. Cuando verdaderamente llega un '0' por la patilla Eco se inhabilitará las interrupciones, y se procederá a la visualización de la distancia recogida en esa muestra. Esta visualización será durante dos segundos, lo que también produce que se agote el tiempo de espera mínimo entre muestra y muestra.

En cuanto a la conexión del sensor al entrenador se realizará de la siguiente manera:

Patilla "0V" será conectada a GND. Patilla "NC" se ha dejado sin conectar ya que se ha usado el modo 1 de funcionamiento del sensor. Patilla "Entrada Eco" conectada a la patilla 3 del puerto A del microcontrolador. Patilla "Salida Eco" conectada a la patilla 4 del puerto A del microcontrolador.

Patilla "5V" que sale del sensor será conectado a la fuente de 5V incluida en el entrenador.

Una vez realizada la conexión se procede a cargar el programa en el microcontrolador. En principio y como ya se ha explicado en el apartado de sensores telemétricos se hará uso del timer0, su uso es indispensable para calcular la distancia del objeto hasta el sensor.

El programa usado cargado en el microcontrolador es simple, y recoge muestras de distancias hacia objetos del entorno cada dos segundos, estas distancias se visualizan en la pantalla LCD integrada en el entrenador, para mayor comodidad.

El diseño del circuito para la comprobación y verificación del sensor SRF05 viene dado en la siguiente imagen:

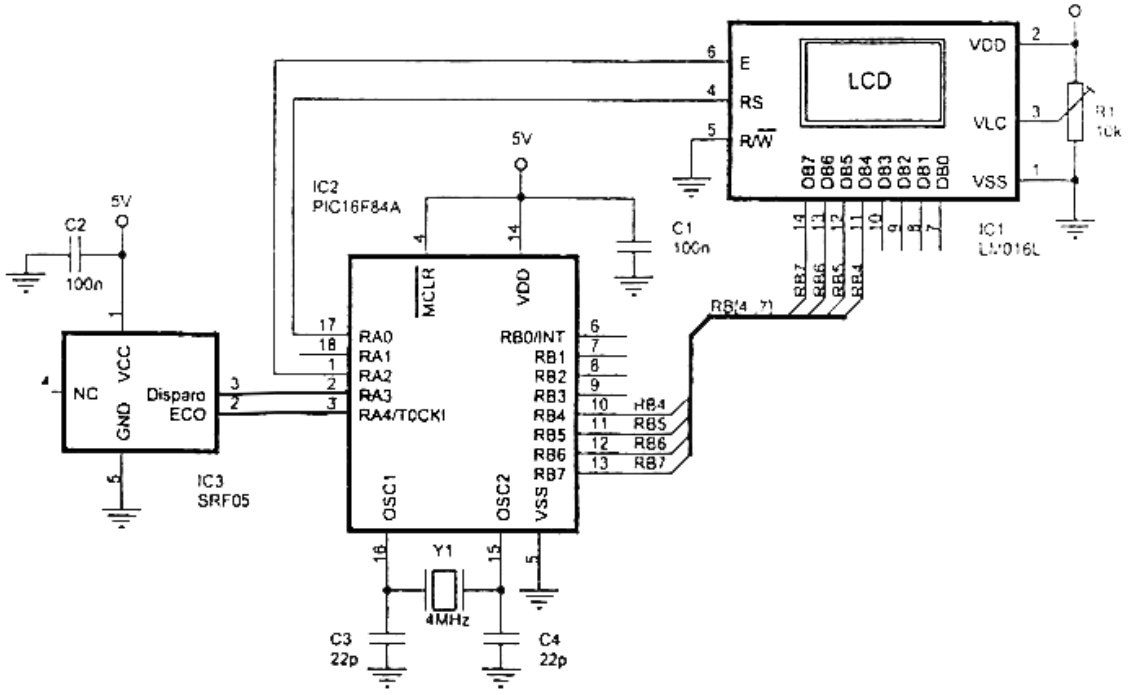


Ilustración 43: Conexión del sensor SRF05 para calibración

1.3.2.1.2 Ajustes de los sensores

La correcta distancia de un objeto hacia en sensor, viene principalmente dada en función del valor cargado inicialmente en el Timer0. Según los cálculos teóricos se debería de inicializar el Timer0 con un valor de 226, para que produzca interrupciones cada 60 µS.

Primeramente se carga el programa con valor del Timer0 con 226, tras unas muestras se observa que para una medida real de 130 cm se obtiene una medida experimental de 110 cm, por lo que hay una gran desviación entre la realidad y la distancia que aporta el sensor. Tras comprobar que el valor teórico dado no es válido para la realidad, seguidamente se procede a cambiar este valor por otros y así obtener experimentalmente un valor que se ajuste a lo real.

Ahora se escoge un valor alto del Timer0 para ir acotando el óptimo rango de funcionamiento. Para ello se inicializa el Timer0 a 235, con el resultado de para una misma medida de 130 cm se obtiene el valor de 150 cm por lo que se aleja bastante de la realidad.

Como siguiente objetivo es cambiar el Timer0 a un valor más bajo que 235, se toma el 229, este valor sí se acerca más a la realidad pero se queda corta ya que da 121 cm.

Finalmente se toma el valor 231 para cargarlo al Timer0, en este caso si se ajusta a la realidad obteniendo una muestra de 130 cm, por lo que se da por válido ya que si es la medida exacta.

1.3.2.2 Segunda etapa

1.3.2.2.1 Recepción y envío de medidas en el PIC-WEB

En principio lo único que tiene que recibir el PIC-WEB desde el robot, serán las medidas tomadas en cada instante por parte del sensor SRF05, que serán enviadas por parte del microcontrolador. El PIC-WEB a su vez, retransmitirá estos datos recibidos por el puerto serie al servidor de aplicaciones vía internet, por el puerto Ethernet. Y también realizará el proceso inverso, recibir la información procedente del servidor de aplicaciones vía internet y redirigirlos al puerto serie del sistema PIC-WEB.

La recepción de medidas va a ser vía puerto serie. La comunicación puerto serie se facilita con el uso de los módulos transceptores de datos multipunto WIZ2-434-RS (ya explicado en el apartado *1.3.1.1.4 Comunicación RF* de esta memoria).

Lo recibido por el puerto serie será la distancia recogida por el robot al objeto u obstáculo más próximo.

La recepción de medidas por puerto serie será tratada por un programa en C, se compilará en MPLAB C18 para poder grabarlo en el PIC18F452 del sistema PIC-WEB. Este programa además de encargarse de la recepción de datos por puerto serie, también será la encargada de enviar datos desde el PIC-WEB al servidor de aplicaciones y del proceso inverso.

En el anexo 2, software empleado en el PIC-WEB, se puede consultar el código de este programa llamado *picweb.c*.

1.3.2.2.2 Control de movimientos desde el PIC-WEB hacia el robot

Aunque el PIC-WEB no será el encargado de controlar el robot, se explicarán el control de movimientos en este apartado. Dejando así la tercera etapa del PFC para la explicación técnica del funcionamiento de la comunicación entre la página JSP alojada en el servidor de aplicaciones y el resto de los bloques del proyecto.

Los movimientos básicos del robot serán: avance, giro derecha, giro izquierda, parar, atrás.

Estos movimientos van a estar muy controlados por parte del programa que sustenta estas acciones, ya que de ellos depende la perfecta localización del robot en el plano. Esto es, que dependiendo de las instrucciones que indique el usuario el robot se situará en un emplazamiento o en otro.

La generación de movimientos se crea a través de una clase java *LocalizaciónRobot.java*, que se encontrará ligada a la página JSP mediante un webproject de eclipse, al igual que las otras clases java. Esta clase esta compuesta de una serie de botones que corresponden con cada una de las acciones que puede realizar el robot. La clase *LocalizaciónRobot.java* se puede consultar en el anexo 3, software empleado en el servidor de aplicaciones.

Cuando se pulsa, por ejemplo, el botón de avance se envía por puerto serie una 'W' correspondiente a la señal de avance, que será recibida por el microcontrolador y el cual lo interpretará como la puesta de los dos motores DC en sentido positivo.

La siguiente tabla recoge las acciones asociadas a una señal que será la que diferencie cada una de estas acciones:

ACCIONES	CÓDIGO (ASCII)
AVANZA	W
ATRÁS	S
GIRO A DERECHA	D
GIRO A IZQUIERDA	A
PARADA	(ESPACIO)

Tabla 13: Acción / Código ASCII

LocalizacionRobot.java es la clase Java más funcional y entre otros objetivos es la encargada de transmitir los movimientos que debe seguir el robot para completar el plano. A la vez que da las instrucciones de movimiento, trata de controlar la situación exacta del robot, para ello se ha programado una serie de complejos métodos que lo harán posible. La dificultad de controlar y saber la posición exacta del robot viene dada por que el sistema debe ser con memoria, esto es, el sistema debe saber cuáles han sido los últimos movimientos del robot para así situarlo en una dirección u otra.

El robot se va a mover por el plano definido por las coordenadas (X, Y) y según hacia donde avance se va a sumar o restar sobre estas coordenadas. Sobre el plano si el robot avanza hacia derecha lo natural es sumar en la coordenada X el desplazamiento, en cambio si avanza hacia la izquierda va a tener que restar lo que se ha movido en la posición –X.

Para saber cuánto ha avanzado, se debe calcular el tiempo que ha estado en modo avance, esto se calcula mediante el método *obtieneTiempo()*. Esto se hace mediante un temporizador que se pondrá a contar justamente a la vez de enviar la instrucción de "Avanzar" (mediante el envío de una letra 'W'), y se parará este temporizador una vez ejecutada la orden de "Parada" (mediante el carácter de espacio ' '), se contará el tiempo transcurrido y se multiplicará por la velocidad del robot; que es, en condiciones óptimas, la misma (sobre unos 10cm/s). Este tiempo vendrá dado en milisengundos para dar mayor precisión a la localización del robot.

Como se ha comentado, esta clase hace girar el robot con giros de 90º. Esto se consigue cuando se envía mediante el puerto serie la instrucción de giro hacia derecha o izquierda, y a continuación se envía la instrucción de parada. El tiempo entre el envío de la instrucción de giro y de parada está estimado según la superficie donde gire el robot. Para desarrollar el tiempo de giro se ha usado un thread por el cual se manda dormir el sistema durante este tiempo.

Además del sentido de avance, retroceso y parada, está la posibilidad de hacer girar el robot tanto a derecha como a izquierda.

Estos giros se producen mediante el envío de las letras 'A' y 'D' respectivamente. El robot no va a girar libremente, esto es, no va a poder girar todo lo que el usuario quiera, sino que se va a mover a intervalos de 90º grados. La razón de especificar unos intervalos de giro es simplificar la localización del robot en el plano.

Con esto se consigue que el robot sólo se pueda mover en líneas horizontales y verticales, es decir, se suma o resta sobre el eje X o se suma o resta sobre el eje Y. Nunca se va a mover en diagonal sobre el plano.

1.3.2.2.3 Software gestor del PIC-WEB

Como se ha comentado anteriormente el software encargado del reenvío de información que se va producir en el sistema PIC-WEB estará programado en C y compilado por MPLAB C18, así obtendremos un archivo .hex para poder grabarlo en el PIC18F452 con el programa Icprog.

Por lo tanto el PIC-WEB estará programado como un "TCP/IP to RS232 bridge", es decir, un puente directo entre la comunicación RS232 y TCP/IP en ambos sentidos.

El programa debe mantener el formato de la información que le llega de ambos puertos, usando los pines TX/RX de transmisión USART para la comunicación RS232 y los pines TX/RX del módulo ENC28J60 para la comunicación TCP/IP. Hay que tener en cuenta que para la comunicación con el servidor de aplicaciones se deberá crear un enlace mediante sockets.

En el código del programa *picweb.c* no se incluirán las direcciónes IP correspondientes al servidor de aplicaciones y se programarán una vez se haya garantizado el conocimiento de estas. También existe la posibilidad de contratar una IP fija o de aplicar un servicio de DNS como NO-IP con el cual se enmascara la dirección IP dinámica, es decir, se sustituye por una simple URL.

También habrá que tener en cuenta la configuración de los puertos en el router al que esté conectado el sistema PIC-WEB, para que las peticiones que vengan a nuestro puerto específico las redirija hacia nuestra IP dentro de la LAN.

El código del programa se puede consultar en el anexo 2, software empleado en el PIC-WEB.

1.3.2.3 Tercera etapa

1.3.2.3.1 Recepción de medidas vía Internet desde el PIC-WEB en el PC

Para la recepción y envío de información en el servidor de aplicaciones se hace uso de sockets programados en java en *servidor.java*, que se puede consultar en el anexo 3.

Los datos recibidos serán procesados por la clase *reconstrucción.java*, esta es la clase que finaliza el proceso de construcción de un plano a partir de las muestras recogidas por el robot.

Este código es el encargado de dibujar y unir los puntos detectados por el robot sobre el plano, una vez que haya terminado de tomar muestras sobre la situación.

A esta clase se le pasa un array de puntos, junto con las medidas del plano. El array de puntos contiene todos los puntos que ha recogido el robot mediante su exploración en el plano.

A partir de estos puntos se va a reconstruir el plano. En principio se deben de unir estos puntos para así encontrar un plano que corresponda con la realidad. Como primera medida adoptada la unión entre puntos del plano se hizo enlazando todos los puntos entre sí. Como resultado se obtiene un plano muy tedioso, con muchas líneas. Lo que hizo muy poco visible el plano.

Como solución adoptada se accedió a hacer un estudio por cada punto. De tal manera que se unirá un punto con los dos más cercanos. Para ello se ha calculado la distancia de cada punto con el resto de puntos contenido en el array de puntos. Así cada punto del plano quedaría unido con otros dos eliminando muchas de estas uniones que aparecían en la anterior medida adoptada.

Para calcular la distancia entre estos dos puntos, se ha recurrido a la siguiente fórmula:

$$distancia = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Para finalizar, se abrirá una ventana emergente para mostrar el plano obtenido. Siendo la siguiente figura un ejemplo de ello:

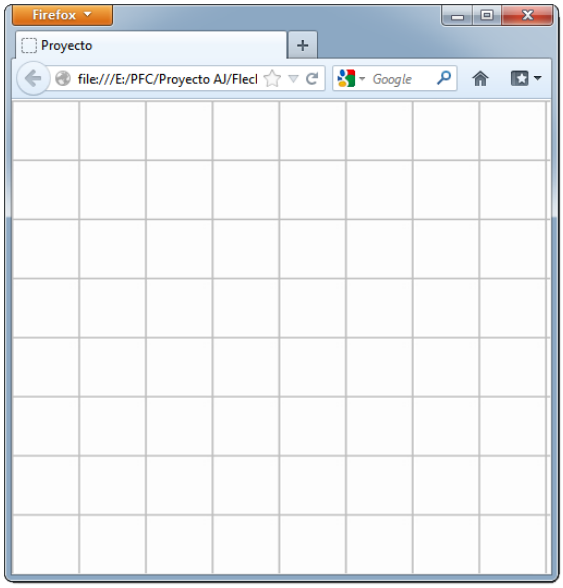


Ilustración 44: Ventana representación del plano

1.3.2.3.2 Software gestor de la interfaz gráfica que controla al PIC-WEB

Se ha decidido implementar un sitio web desde el cual se podrá acceder a toda la información del proyecto.

Dentro de la sección proyecto encontraremos la interfaz gráfica de control, se puede consultar la programación de esta página JSP (*proyecto.jsp*) en el anexo 3.

Para el software gestor de la interfaz gráfica de control se hace uso de botones html y las funciones java integradas en la página JSP. Esta interfaz se compone de cuatro flechas que indican el sentido de movimiento del robot y un símbolo de stop, que es el encargado de para el robot. En la siguiente figura se observan los botones integrados en la página JSP.

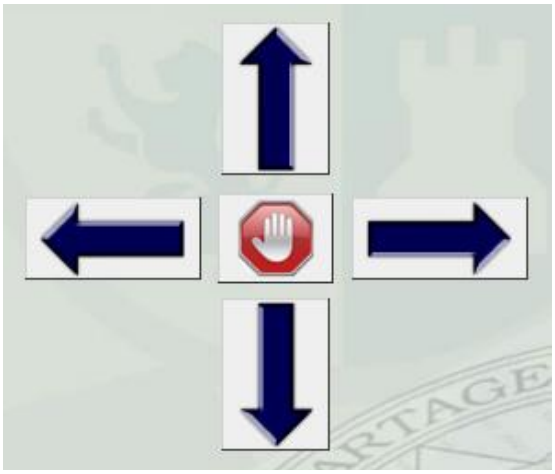


Ilustración 45: Botones control interfaz gráfica

2 PLANOS

A continuación se podrán observar los planos más importantes del proyecto, empezando con el plano general del robot y siguiendo con los planos del PIC-WEB y todos sus componentes:

- **Robot :**

Se puede observar el conexionado de los distintos componentes eléctricos que conforman el circuito. Entre ellos cabe destacar el microcontrolador, el componente MAX232, el sensor de distancias SRF05, el driver L239B, los motores, y los sensores detectores de obstáculos.

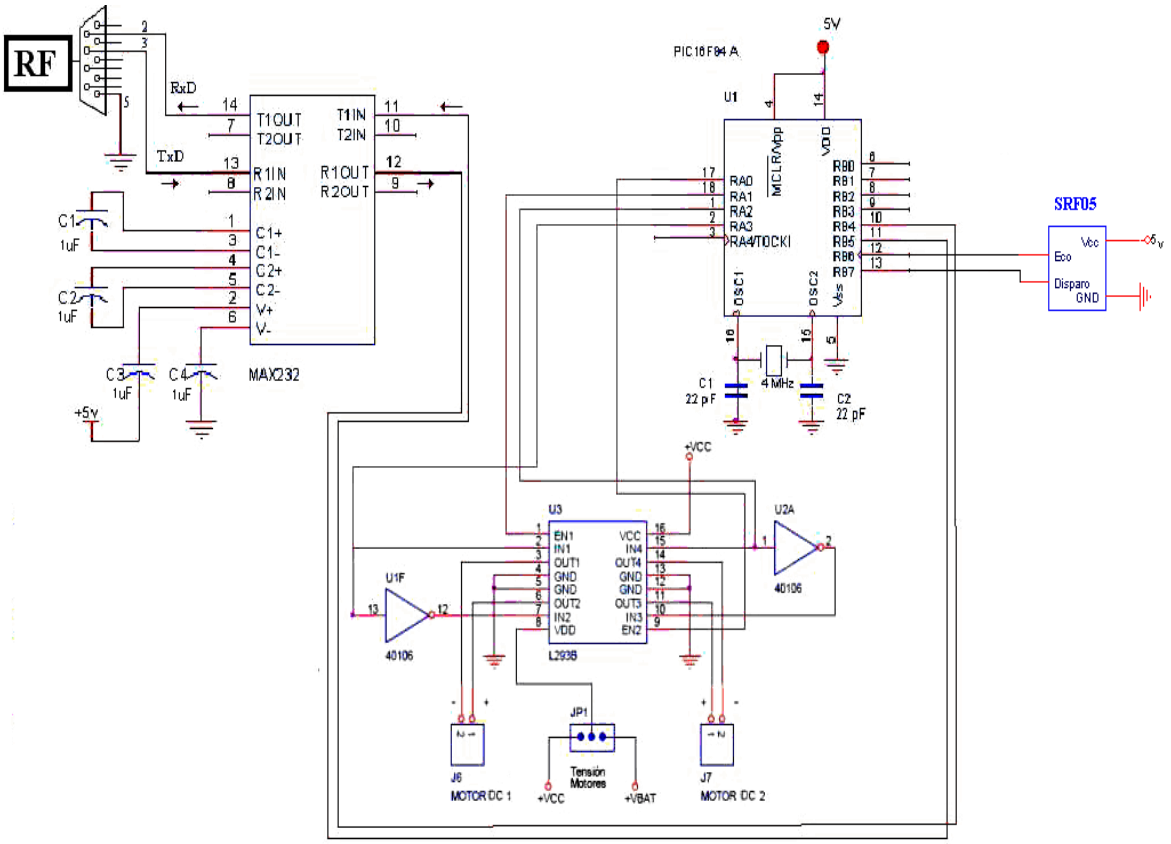


Ilustración 46: Esquema eléctrico general del robot

- **PIC-WEB :**

Se muestran los esquemas eléctricos de cada uno de sus componentes, se puede observar de que punto a que punto van conectados los pines entre los diferentes componentes.

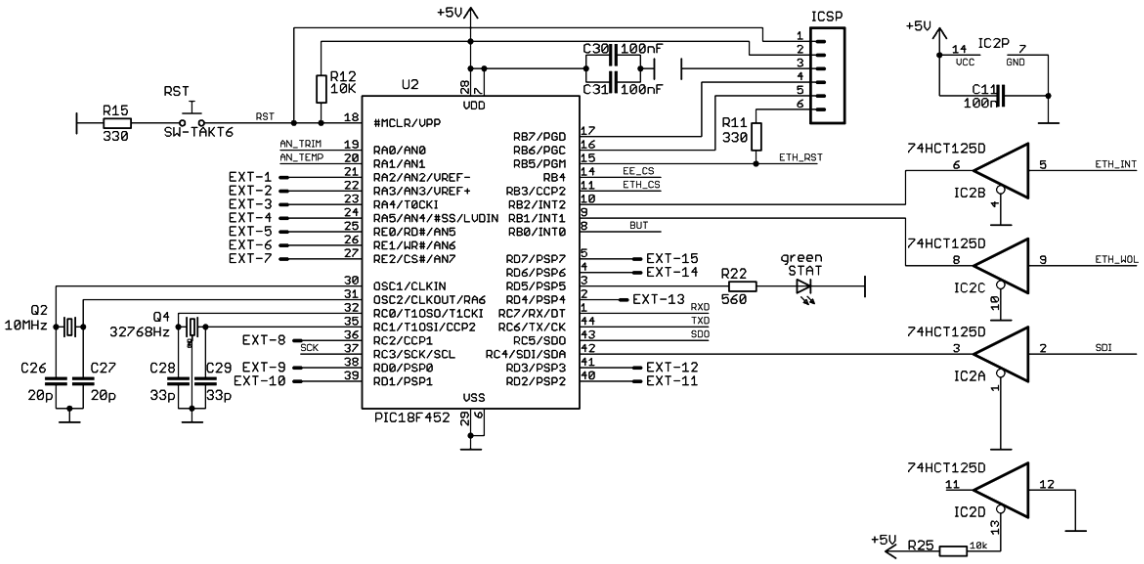


Ilustración 47: Esquema eléctrico PIC18F452, sensores e ICSP.

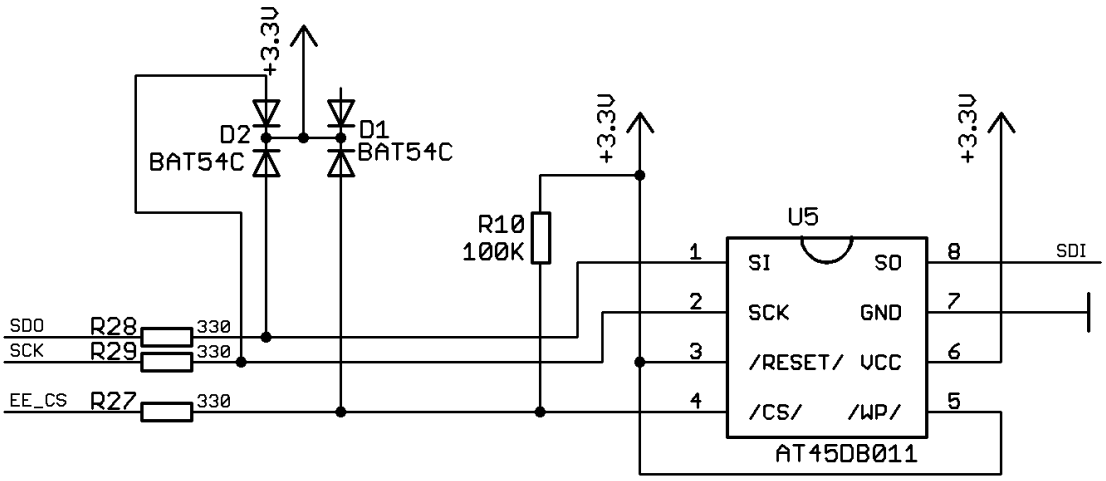


Ilustración 48: Esquema eléctrico memoria flash PIC-WEB

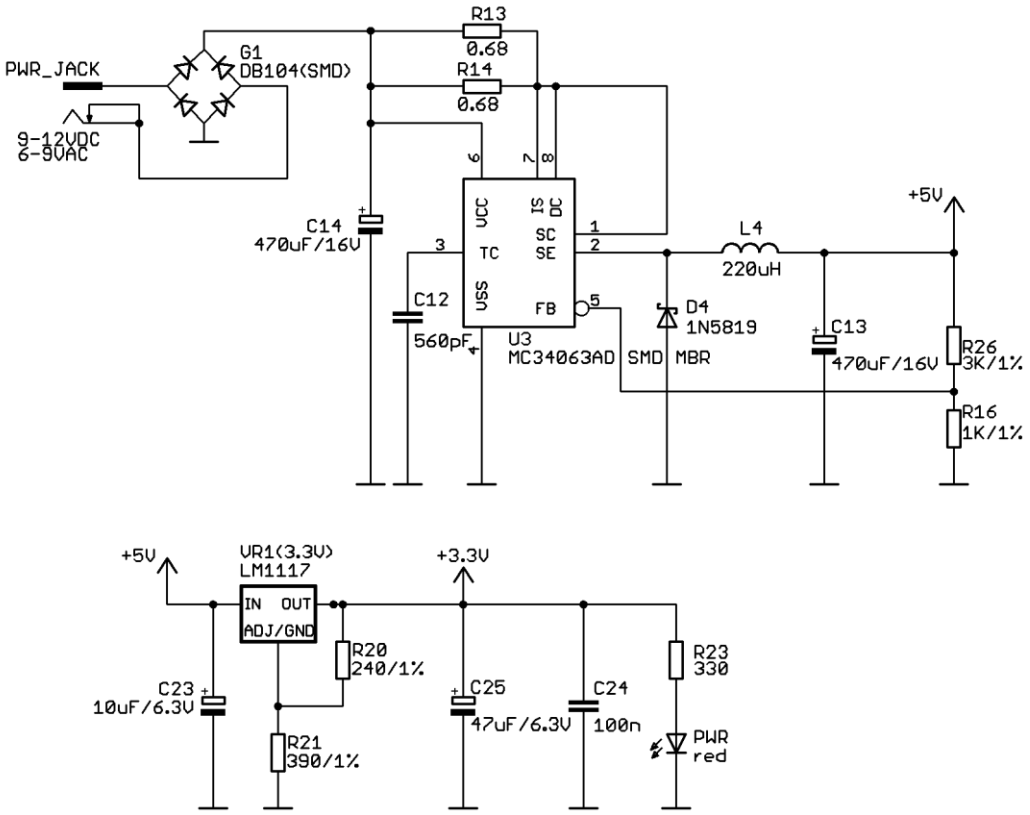


Ilustración 49: Esquema eléctrico alimentación y modulador de voltaje PIC-WEB.

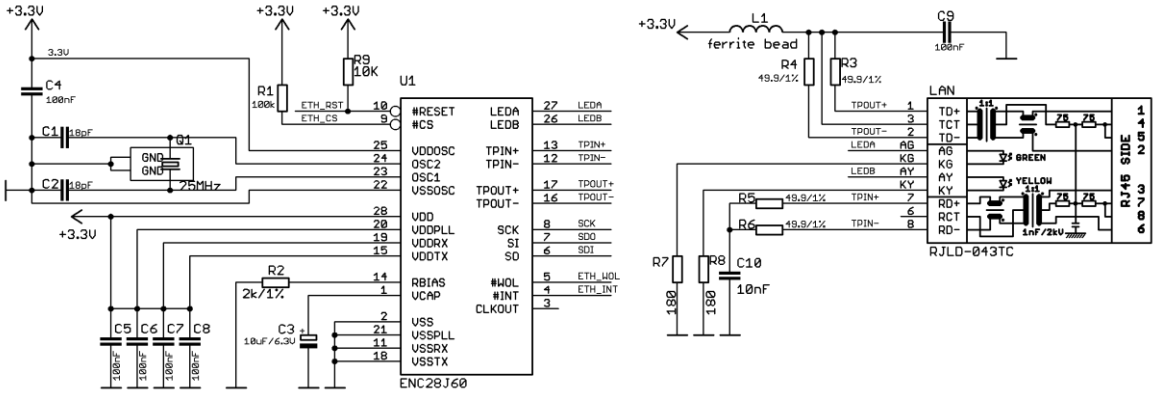


Ilustración 50: Esquema eléctrico módulo ENC28J60 y RJ45.

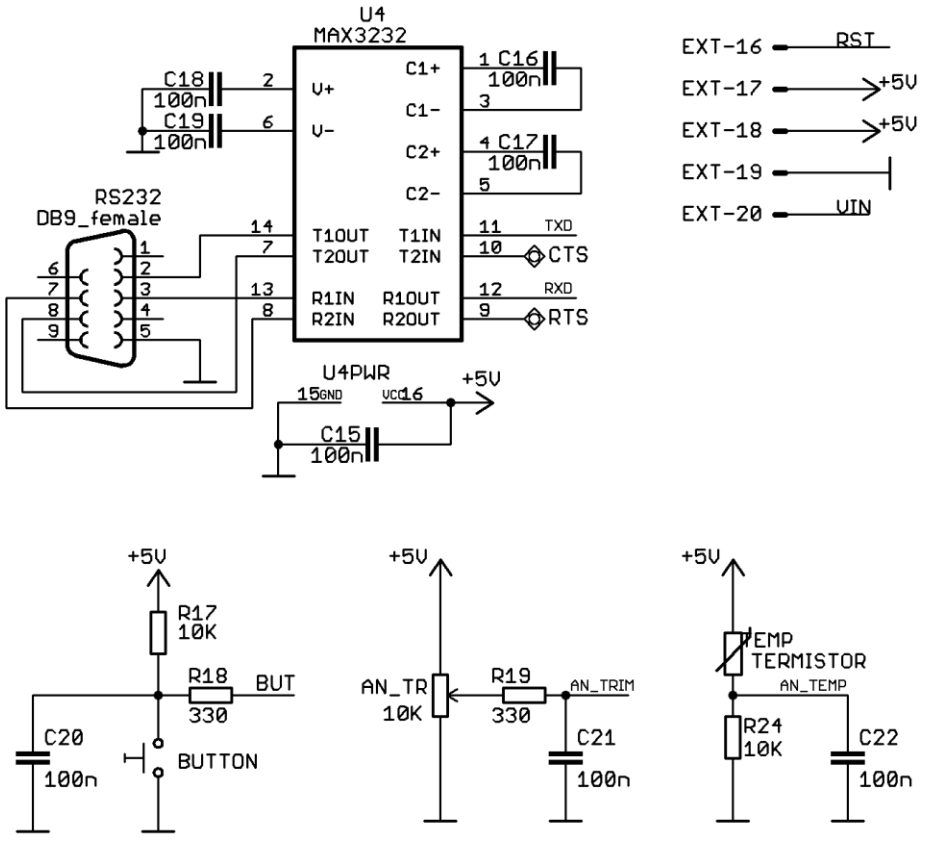


Ilustración 51: Esquema eléctrico módulo MAX232, DB9, Pines EXT, botón y termistor.

3 LINEAS FUTURAS

Una vez concluido los objetivos del proyecto se proponen una serie de líneas de ampliación y mejoras.

Como se había comentado en la sección de planteamiento general del sistema PIC-WEB, se podría sustituir este módulo por un mini PC que está muy demandado actualmente, el nuevo RaspBerry PI. Este nuevo sistema tiene una gran capacidad de procesamiento y a un precio realmente asequible.

Este nuevo PC, tiene un tamaño realmente pequeño por lo que se podría integrar en el mismo robot y realizar las funciones del servidor de aplicaciones dotándolo de un modem 3G USB. Además se podría prescindir de los módulos RF ya que el servidor se encontaría físicamente instalado en el robot, usando para ello un conversor USB-RS232 para comunicar el RaspBerry PI con el PIC16F84A. Así la libertad de movimiento del robot aumentaría tanto como el alcance de cobertura 3G.

El esquema general del proyecto, instalando las mejoras propuestas, quedaría de la siguiente manera:

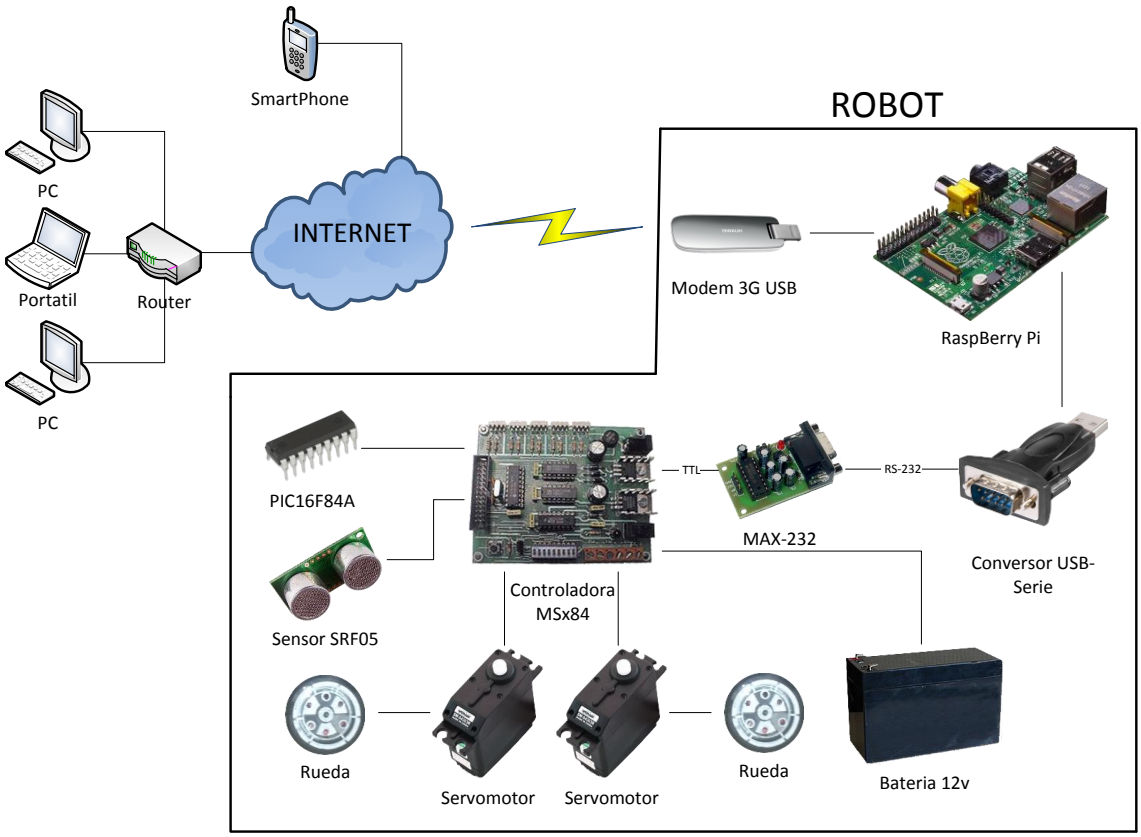


Ilustración 52: Esquema para líneas futuras.

4 BIBLIOGRAFÍA

Referencias Bibliográficas

[1] MicroPic Trainer Plus. Manual de Usuario. Microsystems Engineering.
[2] Microcontroladores PIC. PIC18Fxx2.
[3] Manual de usuario PIC School. Microsystems Engineering.
[4] Microcontroladores PIC: Diseño de aplicaciones. 3ª Ed. J.M. Angulo, I. Angulo. McGraw-Hill.2003.
[5] Microcontrolador PIC16F84A.Desarrollo de proyectos. Ed. E. Palacios, F. Remiro. Ra-Ma.2004
[6] Hoja técnica: Transceptor Multipunto WIZ2-434-RSx. Microsystems Engineering.
[7] Hoja técnica: Sistema PIC-WEB.
[8] Manual de Usuario: Controladora MSx84. Microsystems Engineering.
[9] PIC16F84A Data Sheet 18-Pin Enhanced FLASH/EEPROM 8-bit Microcontrollers, Microchip Technology Inc.
[10] DEITEL & DEITEL Java™ How to Program, 6/e (©2005)

Referencias Web

[1] http://www.superrobotica.com
[2] http://.java.sun.com
[3] http://www.microcontroladores.com
[4] http://www.x-robotics.com
[5] http://www.javahispano.com
[6] http://www.beyondlogic.org/
[7] http://www.msebilbao.com/tienda/default.php
[8] http://geneura.ugr.es/~jmerelo/JSP/
[9] http://www.myeclipseide.com/documentation/quickstarts/tomcat/
[10] http://cse.csusb.edu/turner/java_web_programming/jsp/
[11] http://dis.um.es/~lopezquesada/documentos/IES_0607/DFSI/curso/UT9/jsp/html/estructura.html

5 ANEXOS

5.1 Anexo 1: Software empleado en el microcontrolador

- **Programa *Robot.asm:***

Es el programa principal que es cargado al microcontrolador. Este programa está compuesto por las operaciones básicas del robot, así como movimientos, toma de medidas, etc. A continuación se aporta el código en ensamblador de este archivo.

```
;*****************************************************************************
;*                                                                           *
;*                                                                           *
;*                        ING.TELECOMUNICACIONES                             *
;*                            ESP. TELEMATICA                                *
;*     EL PROYECTO  SE BASA EN LA CREACIÓN DE UN MICROBOT                    *
;*     QUE OBEDECERÁ  LAS ORDENES QUE SE LE ENVIEN DESDE                     *
;*     EL PC, RECOGERÁ MUESTRAS  MEDIANTE EL SENSOR SRF05                    *
;*     HACIA UN OBJETO. EL CONTROL DEL ROBOT SE REALIZA                      *
;*     MEDIANTE  EL PROTOCOLO RS232.                                         *
;*                                                                           *
;*****************************************************************************

;ZONA DE DATOS

        __CONFIG _CP_OFF& _WDT_OFF& _PWRTE_OFF& _XT_OSC

        LIST P=16F84A

        INCLUDE <P16F84A.INC>



        CBLOCK 0x0C
        TeclaPulsada        ;Se declaran  las variables  que se van a usar
        Dist                ;en el programa
        Distancia
        ENDC


#DEFINE SALIDAMO0 PORTA,0       ;las patillas  RA0-RA3 serán usadas para controlar
#DEFINE SALIDAMO1 PORTA,1       ;los movimientos  de las ruedas motrices
#DEFINE SALIDAMO2 PORTA,2
#DEFINE SALIDAMO3 PORTA,3

#DEFINE SENSOR1   PORTB,0       ;Patilla  usada conexión sensor choque  izquierdo
#DEFINE Disparo   PORTB,6       ;Disparo  para iniciar  la medida.
```

```
#DEFINE Eco          PORTB,7        ;Pulso cuya anchura hay que medir.

MinimaDistancia      EQU .3         ;Valor mínimo de distancia
MaximaDistancia      EQU .250       ;Valor máximo de distancia
TMR0_Carga60micros   EQU  .231

;Valor obtenido experimentalmente con la ventana Stopwatch para una
;interrupción del Timer 0 cada 60 µs. Si no mide correctamente por las
;tolerancias de los componentes habrá que hacer un ajuste fino de este
;valor, comprobándolo sobre las condiciones reales.


RetornoCarro  EQU .13
CambioLinea   EQU .10
CodigoParada  EQU .0

TeclaAdelante  EQU 'W'          ;Letra enviada si la acción fuera "Avanzar"
TeclaAtras     EQU 'S'          ;Letra enviada si la acción fuera "Atrás"
TeclaIzqda     EQU 'A'          ;Letra enviada si la acción fuera "Girar Izquierda"
TeclaDrecha    EQU 'D'          ;Letra enviada si la acción fuera "Girar dechera"
TeclaStop      EQU ' '          ;Letra enviada si la acción fuera "Parar"


;ZONA DE CÓDIGOS


        ORG 0
        goto Inicio             ;Va a inicio de programa
        ORG 4
        goto ServicioInterrupcion       ;Va a rutina de tratamiento de interrupciones


Mensajes
        addwf PCL,F
MensajeSensorDSi
        DT "CHOQUE POR LA DERECHA",0x00
MensajeSensorISi
        DT "CHOQUE POR LA IZQUIERDA",0x00
MensajeAviso
        DT "ATENCION!!:",0x00


Inicio

        call RS232_Inicializa           ;Inicializa el puerto serie (RS-232)
        bsf  STATUS,RP0                 ;Banco 1 -------------------------
        bcf SALIDAMO0                   ;Declara RA0 como salida
        bcf SALIDAMO1                   ;Declara RA1 como salida
```

```
        bcf SALIDAMO2               ;Declara RA2 como salida
        bcf SALIDAMO3               ;Declara RA3 como salida
        bsf SENSOR1                 ;Declara RB0 como entrada
        bcf Disparo                 ;Declarada como salida
        bsf Eco                     ;Declarada como entrada
        bcf OPTION_REG,NOT_RBPU     ;Habilita la resistencia pull-up
        bsf OPTION_REG,INTEDG       ;Se escoge flanco ascendente del reloj
        bcf STATUS,RP0              ;Banco 0 -------------------------
        movlw b'10010000'           ;con la interrupcion rb0/int solamente
        movwf INTCON                ;Habilita las interrupción de RB0


Principal
        call ComprobarSensorD       ;Llama a la rutina de comprobar sensor derecho
        call RS232_LeeDato          ;Lee dato procedente del puerto serie
        call ComprobarSensorD       ;
        call TesteaTeclado          ;Llama a la rutina para comprobar que se ha mandado
        goto Principal              ;Va a principal


;*****************************************************************************
;******************* TESTEA TECLADO ******************************************
;*****************************************************************************
;Esta rutina trata de comprobar qué acción ha sido enviada por el             *
;puerte serie. Las acciones posibles son "Adelante","Atrás","Para",           *
;"Giro derecha","Giro izquierda". También se testea el sensor de              *
;choque derecho ya que este no tiene asociada ninguna interrupción.           *
;*****************************************************************************
TesteaTeclado
        call ComprobarSensorD
        movwf TeclaPulsada
        xorlw TeclaAdelante
        btfsc STATUS,Z
        goto Adelante           ;Si acción es "Adelante"

        movf TeclaPulsada,W
        xorlw TeclaAtras
        btfsc STATUS,Z
        goto Atras              ;Si acción es "Atrás"

        movf TeclaPulsada,W
        xorlw TeclaIzqda
        btfsc STATUS,Z
        goto Izquierda          ;Si acción es "Gira izquierda"

        movf TeclaPulsada,W
        xorlw TeclaDrecha
        btfsc STATUS,Z
```


```
        goto Derecha            ;Si acción es "Gira dcha"

        movf TeclaPulsada,W
        xorlw TeclaStop
        btfsc STATUS,Z
        goto Stop               ;Si acción es "Para"
        return


Stop
        movlw b'00000'          ;Código de parada
        goto  EjecutaOrden

Adelante
        movlw b'01111'          ;Código de avance
        goto  EjecutaOrden

Atras
        movlw b'01100'
        goto  EjecutaOrden

Izquierda
        movlw b'01110'
        goto  EjecutaOrden

Derecha
        movlw b'01101'

EjecutaOrden
        call ComprobarSensorD
        movwf PORTA             ;Mueve el W al puerto A
        subwf CodigoParada,w    ;Resta el código de parada
        btfsc STATUS, Z         ;Si ha ejecutado la orden de parada
        goto EsParada           ;salta a EsParada

        return

;*****************************************************************
;***************** ES PARADA  ********************************
;*****************************************************************
;Esta subrutina es la encargada de tomar la medida procedente    *
;del sensor una vez que el robot haya parado. La                 *
;llamada a esta rutina devuelve por el acumulador la distancia   *
;del robot al objeto más cercano.                                *
;Luego la medida obtenida es enviada al PC vía puerto serie.     *
;*****************************************************************
EsParada                ;Rutina para tomar medida una vez que haya parado
movlw b'01000001'
```

```
        movfw  Dist
        call       RS232_EnviaDato     ;Envía  la distancia  obtenida  por 232
        return

ServicioInterrupcion
        btfsc    INTCON,T0IF
        goto     AumentaDistancia
        btfss    INTCON,INTF
        retfie
        goto     TesteaSensores

AumentaDistancia
        movlw    TMR0_Carga60micros        ; Carga  el Timer  0.
        movwf   TMR0
        movlw   .1                         ;Se utiliza  instrucción  "addwf",  en lugar  de "incf"
        addwf  Distancia,F                 ;para posicionar  flag  de Carry.
        movlw  MaximaDistancia             ;En caso de desbordamiento  carga  su
                                           ;máximo  valor.
        btfsc     STATUS,C
        movwf  Distancia
        bcf       INTCON,T0IF
        retfie

TesteaSensores
        movlw  b'00000'
        movwf  PORTA
        call   RS232_LineasBlanco
        movlw  MensajeAviso
        call   RS232_Mensaje
        movlw  MensajeSensorISi
        call   RS232_Mensaje
        bcf  INTCON,INTF

        retfie

        INCLUDE <RETARDOS.INC>
        INCLUDE <RS232.INC>

        END
```

A continuación se presentarán las librerías utilizadas para el correcto funcionamiento del robot:

- **Librería *PIC16F84A.INC:***

```
        LIST
; P16F84A.INC  Standard Header File, Version  2.00    Microchip  Technology,  Inc.
        NOLIST

; This  header file  defines  configurations,  registers,  and other useful  bits of
; information  for the PIC16F84 microcontroller.   These  names  are taken to match
; the data sheets as closely  as possible.

; Note that the processor must  be selected  before this  file  is
; included.   The processor may be selected the following  ways:

;       1. Command  line  switch:
;               C:\ MPASM MYFILE.ASM  /PIC16F84A
;       2. LIST directive  in the source  file
;               LIST    P=PIC16F84A
;       3. Processor Type entry in the MPASM full-screen  interface

;==========================================================================
;
;       Revision  History
;
;==========================================================================

;Rev:    Date:     Reason:

;1.00   2/15/99 Initial  Release

;==========================================================================
;
;       Verify  Processor
;
;==========================================================================

        IFNDEF __16F84A
          MESSG "Processor-header  file  mismatch.   Verify  selected  processor."
        ENDIF

;==========================================================================
;
;       Register  Definitions
;
;==========================================================================
```

```
W                   EQU     H'0000'
F                   EQU     H'0001'

;----- Register Files------------------------------------------------------

INDF                EQU     H'0000'
TMR0                EQU     H'0001'
PCL                 EQU     H'0002'
STATUS              EQU     H'0003'
FSR                 EQU     H'0004'
PORTA               EQU     H'0005'
PORTB               EQU     H'0006'
EEDATA              EQU     H'0008'
EEADR               EQU     H'0009'
PCLATH              EQU     H'000A'
INTCON              EQU     H'000B'

OPTION_REG          EQU     H'0081'
TRISA               EQU     H'0085'
TRISB               EQU     H'0086'
EECON1              EQU     H'0088'
EECON2              EQU     H'0089'

;----- STATUS Bits --------------------------------------------------------

IRP                 EQU     H'0007'
RP1                 EQU     H'0006'
RP0                 EQU     H'0005'
NOT_TO              EQU     H'0004'
NOT_PD              EQU     H'0003'
Z                   EQU     H'0002'
DC                  EQU     H'0001'
C                   EQU     H'0000'

;----- INTCON Bits --------------------------------------------------------

GIE                 EQU     H'0007'
EEIE                EQU     H'0006'
T0IE                EQU     H'0005'
INTE                EQU     H'0004'
RBIE                EQU     H'0003'
T0IF                EQU     H'0002'
INTF                EQU     H'0001'
RBIF                EQU     H'0000'

;----- OPTION_REG Bits ----------------------------------------------------

NOT_RBPU            EQU     H'0007'
```

```
INTEDG          EQU     H'0006'
T0CS            EQU     H'0005'
T0SE            EQU     H'0004'
PSA             EQU     H'0003'
PS2             EQU     H'0002'
PS1             EQU     H'0001'
PS0             EQU     H'0000'

;----- EECON1 Bits --------------------------------------------------

EEIF            EQU     H'0004'
WRERR           EQU     H'0003'
WREN            EQU     H'0002'
WR              EQU     H'0001'
RD              EQU     H'0000'

;=========================================================================
;
;       RAM Definition
;
;=========================================================================

        __MAXRAM    H'CF'
        __BADRAM    H'07', H'50'-H'7F', H'87'

;=========================================================================
;
;       Configuration Bits
;
;=========================================================================

_CP_ON          EQU     H'000F'
_CP_OFF         EQU     H'3FFF'
_PWRTE_ON       EQU     H'3FF7'
_PWRTE_OFF      EQU     H'3FFF'
_WDT_ON         EQU     H'3FFF'
_WDT_OFF        EQU     H'3FFB'
_LP_OSC         EQU     H'3FFC'
_XT_OSC         EQU     H'3FFD'
_HS_OSC         EQU     H'3FFE'
_RC_OSC         EQU     H'3FFF'

        LIST
```

- **Librería *RS232.INC:***

```
;********************** Librería "RS232.INC" ******************************
;
;================================================================
;                                Del libro
;        "MICROCONTROLADOR PIC16F84. DESARROLLO DE PROYECTOS"
;                          FRANCISCO BOLUDA SEVILLA
;================================================================
;
; Estas subrutinas permiten realizar las tareas básicas de control de la transmisión
; seríe asincrona según normas RS-232.
;
; Los parámetros adoptados para la comunicación son los siguientes:
; - Velocidad de transmisión de 9600 baudios. La duración de cada bit será 104 µs.
; - Un bit de inicio o Start a nivel bajo.
; - Dato de 8 bits.
; - Sin paridad.
; - Dos bits de final o Stop a nivel alto.
;
; El tiempo entre bit y bit debe coincidir con el periodo de la señal leída o enviada.
; Como la velocidad de transmisión o recepción es de 9600 baudios, el periodo será:
; 1/9600 Baudios = 104 µs. Se utilizará pues la subrutina Retardos_100micros.

        CBLOCK
        RS232_ContadorBits
        RS232_Dato
                EstadoSensor2
        ENDC

#DEFINE  RS232_Entrada    PORTB,5          ; Línea por la que se reciben los datos.
#DEFINE  RS232_Salida     PORTB,2          ; Línea por la que se envían los datos.
#DEFINE  SENSOR2          PORTB,1

;
; Subrutina "RS232_Inicializa" ----------------------------------------------------------
;
; Configura las líneas de salida y entrada del microcontrolador.

ComprobarSensorD
                btfsc SENSOR2
                return
                movlw b'00000'
                movwf PORTA
                ;call RS232_LineasBlanco
                movlw MensajeAviso
                call RS232_Mensaje
                movlw MensajeSensorDSi
```

```
                call RS232_Mensaje
                return

RS232_Inicializa
        bsf     STATUS,RP0
        bsf     RS232_Entrada           ; Esta línea se configura como entrada.
        bcf     RS232_Salida            ; Esta línea se configura como salida.
        bsf     SENSOR2
        bcf     STATUS,RP0
        return

; Subrutina "RS232_LeeDato" ----------------------------------------------------------
;
; El microcontrolador lee el dato por la línea de entrada comenzando por el bit de menor
; peso. El dato leído se envía finalmente en el registro de trabajo W.
;
; El ordenador parte siempre de un nivel alto, que es el estado que tiene cuando no
; envía información. La secuencia utilizada es:
; 1º    Espera que se ejecute el pulso negativo del bit Start o flanco de bajada.
; 2º    Deja pasar un tiempo una y media veces mayor que el periodo de transmisión para
;           saltarse el bit de Start y lee el primer bit en su mitad.
; 3º    Lee el resto de los bits de datos, esperando un tiempo igual a la duración del
;           período entre lectura y lectura para testearlos en mitad del bit.
;
; Salida: En el registro de trabajo W el byte leído.

RS232_LeeDato
        call    ComprobarSensorD
        movlw   d'8'                    ; Número de bits a recibir.
        movwf   RS232_ContadorBits

RS232_EsperaBitStart
        call    ComprobarSensorD
        btfsc   RS232_Entrada           ; Lee la entrada y espera a que sea "0".
        goto    RS232_EsperaBitStart    ; No, pues espera el nivel bajo.
        call    Retardo_100micros       ; El primer bit debe leerlo tiempo igual a una
        call    Retardo_50micros        ; vez y media el periodo de transmisión.

RS232_LeeBit
        bcf     STATUS,C                ;Ahora lee el pin. En principio supone que es 0
        btfsc   RS232_Entrada           ; ¿Realmente es cero?
        bsf     STATUS,C                ; No, pues cambia a "1".
        rrf     RS232_Dato,F            ; Introduce el bit en el registro de lectura.
        call    Retardo_100micros       ; Los siguientes bits lee un periodo más tarde.
        decfsz  RS232_ContadorBits,F    ; Comprueba que es el último bit.
        goto    RS232_LeeBit            ; Si no es el último bit pasa a leer el siguiente.
        call    Retardo_200micros       ; Espera un tiempo igual al los 2 bits de Stop.
        movf    RS232_Dato,W            ; El resultado en el registro W.
```

```
        return

; Subrutinas "RS232_EnviaDato" y "RS232_EnviaNúmero" ----------------------------------
;
; El microcontrolador envía un dato por la línea de salida comenzando por el bit de menor
; peso. En dato enviado será el que le llegue a través del registro de trabajo W.
; 1º.   Envía un "0" durante un tiempo igual al periodo de la velocidad de transmisión.
;           Este es el bit de "Start".
; 2º.   Envía el bit correspondiente:
;       - Si va a enviar un "0" permanece en bajo durante el periodo correspondiente.
;       - Si va a escribir un "1" permanece en alto durante el periodo correspondiente.
; 3º.   Envía dos bits "1" durante un tiempo igual al período de la velocidad de
;           transmisión cada uno. Estos son los dos bits de Stop.
;
; Entrada:      En (W) el dato a enviar.

RS232_EnviaNumero               ; Envía el código ASCII de un número.
        addlw  '0'              ; Lo pasa a código ASCII sumándole el ASCII del 0.

RS232_EnviaDato
        movwf  RS232_Dato               ; Guarda el contenido del byte a transmitir.
        movlw  d'8'                     ; Este es el número de bits a transmitir.
        movwf  RS232_ContadorBits
        bcf    RS232_Salida                     ; Bit de Start.
        call   Retardo_100micros

RS232_EnviaBit                  ; Comienza a enviar datos.
        rrf    RS232_Dato,F     ; Lleva el bit que se quiere enviar al Carry para
        btfss  STATUS,C         ; deducir su valor. ¿Es un "1" el bit a transmitir?
        goto   RS232_EnviaCero  ; No, pues envía un "0".

RS232_EnviaUno
        bsf    RS232_Salida             ; Transmite un "1".
        goto   RS232_FinEnviaBit

RS232_EnviaCero
        bcf    RS232_Salida             ; Transmite un "0".

RS232_FinEnviaBit
        call   Retardo_100micros        ; Este es el tiempo que estará en alto o bajo.
        decfsz RS232_ContadorBits,F     ; Comprueba que es el último bit.
        goto   RS232_EnviaBit           ; Como no es el último bit repite la operación.
        bsf    RS232_Salida             ; Envía dos bits de Stop.

        call   Retardo_200micros
        return

        INCLUDE <RS232_MEN.INC>
```

- **Librería *RS232_MEN.INC:***

```
;#####LIBRERIA  RS232_MEN.INC#####
        CBLOCK
RS232_ApuntaCaracter
RS232_ValorCaracter
        ENDC

RS232_Mensaje
        movwf  RS232_ApuntaCaracter
        movlw  Mensajes
        subwf  RS232_ApuntaCaracter,F
        decf     RS232_ApuntaCaracter,F

RS232_VisualizaOtroCaracter
        movf    RS232_ApuntaCaracter,W
        call      Mensajes
        movwf  RS232_ValorCaracter
        movf    RS232_ValorCaracter,F
        btfsc    STATUS,Z
        goto     RS232_FinMensaje

RS232_NoUltimoCaracter
        call      RS232_EnviaDato
        incf      RS232_ApuntaCaracter,F
        goto     RS232_VisualizaOtroCaracter
RS232_FinMensaje
        return

;Subrutina  RS232_LineasBlanco,  Visualiza  unas cuantas  lineas  en blanco
;en el monitor  del ordenador

        CBLOCK
RS232_ContadorLineas
        ENDC

RS232_LineasBlanco
        movlw   .10
        movwf  RS232_ContadorLineas

RS232_LineasBlancoLazo
        movlw  .10
        call      RS232_EnviaDato
        decfsz  RS232_ContadorLineas,F
        goto     RS232_LineasBlancoLazo
        movlw  .13
        call      RS232_EnviaDato
        return
```

- **Librería *RETARDOS.INC:***

```
;************************ Librería "RETARDOS.INC" ************************
;
;===============================================================
;                                Del libro
;        "MICROCONTROLADOR PIC16F84. DESARROLLO DE PROYECTOS"
;                          E. Palacios, F. Remiro y L. López.
;                          Editorial Ra-Ma. www.ra-ma.es
;===============================================================
; Librería con múltiples subrutinas de retardos, desde 4 microsegundos hasta 20 segundos.
; Además se pueden implementar otras subrutinas muy fácilmente.
;
; Se han calculado para un sistema microcontrolador con un PIC trabajando con un cristal
; de cuarzo a 4 MHz. Como cada ciclo máquina son 4 ciclos de reloj, resulta que cada
; ciclo máquina tarda 4 x 1/4MHz = 1 µs.
; En los comentarios, "cm" significa "ciclos máquina".
;
; ZONA DE DATOS ****************************************************

        CBLOCK
        R_ContA                 ; Contadores para los retardos.
        R_ContB
        R_ContC
        ENDC

; RETARDOS de 4 hasta 10 microsegundos ----------------------------------------------
; A continuación retardos pequeños teniendo en cuenta que para una frecuencia de 4 MHZ,
; la llamada a subrutina "call" tarda 2 ciclos máquina, el retorno de subrutina
; "return" toma otros 2 ciclos máquina y cada instrucción "nop" tarda 1 ciclo máquina.

Retardo_10micros        ; La llamada "call" aporta 2 ciclos máquina.
        nop             ; Aporta 1 ciclo máquina.
        nop             ; Aporta 1 ciclo máquina.
        nop             ; Aporta 1 ciclo máquina.
        nop             ; Aporta 1 ciclo máquina.
        nop             ; Aporta 1 ciclo máquina.
Retardo_5micros         ; La llamada "call" aporta 2 ciclos máquina.
        nop             ; Aporta 1 ciclo máquina.
Retardo_4micros         ; La llamada "call" aporta 2 ciclos máquina.
        return          ; El salto del retorno aporta 2 ciclos máquina.

; RETARDOS de 20 hasta 500 microsegundos ---------------------------------------

Retardo_500micros       ; La llamada "call" aporta 2 ciclos máquina.
        nop             ; Aporta 1 ciclo máquina.
        movlw d'164'    ; Aporta 1 ciclo máquina. Este es el valor de "K".
        goto  RetardoMicros     ; Aporta 2 ciclos máquina.
```

```
Retardo_200micros               ; La llamada "call" aporta 2 ciclos máquina.
        nop                     ; Aporta 1 ciclo máquina.
        movlw d'64'             ; Aporta 1 ciclo máquina. Este es el valor de "K".
        goto  RetardoMicros     ; Aporta 2 ciclos máquina.
Retardo_100micros               ; La llamada "call" aporta 2 ciclos máquina.
        movlw d'31'             ; Aporta 1 ciclo máquina. Este es el valor de "K".
        goto  RetardoMicros     ; Aporta 2 ciclos máquina.
Retardo_50micros                ; La llamada "call" aporta 2 ciclos máquina.
        nop                     ; Aporta 1 ciclo máquina.
        movlw d'14'             ; Aporta 1 ciclo máquina. Este es el valor de "K".
        goto  RetardoMicros     ; Aporta 2 ciclos máquina.
Retardo_20micros                ; La llamada "call" aporta 2 ciclos máquina.
        movlw d'5'              ; Aporta 1 ciclo máquina. Este es el valor de "K".

; El próximo bloque "RetardoMicros" tarda:
; 1 + (K-1) + 2 + (K-1)x2 + 2 = (2 + 3K) ciclos máquina.

RetardoMicros
        movwf R_ContA           ; Aporta 1 ciclo máquina.
Rmicros_Bucle
        decfsz R_ContA,F        ; (K-1)x1 cm (cuando no salta) + 2 cm (al saltar).
        goto   Rmicros_Bucle    ; Aporta (K-1)x2 ciclos máquina.
        return                  ; El salto del retorno aporta 2 ciclos máquina.

; En total estas subrutinas tardan:
; - Retardo_500micros: 2 + 1 + 1 + 2 + (2 + 3K)=500 cm = 500 µs. (para K=164 y 4 MHz).
; - Retardo_200micros: 2 + 1 + 1 + 2 + (2 + 3K)=200 cm = 200 µs. (para K= 64 y 4 MHz).
; - Retardo_100micros: 2   + 1 + 2 + (2 + 3K) = 100 cm = 100 µs. (para K= 31 y 4 MHz).
; - Retardo_50micros  : 2 + 1 + 1 + 2 + (2 + 3K) =  50 cm =  50 µs. (para K= 14 y 4 MHz).
; - Retardo_20micros  : 2   + 1   + (2 + 3K) =  20 cm =  20 µs. (para K=  5 y 4 MHz).
; RETARDOS de 1 ms hasta 200 ms. ----------------------------------------------------

Retardo_200ms                   ; La llamada "call" aporta 2 ciclos máquina.
        movlw d'200'            ; Aporta 1 ciclo máquina. Este es el valor de "M".
        goto  Retardos_ms       ; Aporta 2 ciclos máquina.
Retardo_100ms                   ; La llamada "call" aporta 2 ciclos máquina.
        movlw d'100'            ; Aporta 1 ciclo máquina. Este es el valor de "M".
        goto  Retardos_ms       ; Aporta 2 ciclos máquina.
Retardo_50ms                    ; La llamada "call" aporta 2 ciclos máquina.
        movlw d'50'             ; Aporta 1 ciclo máquina. Este es el valor de "M".
        goto  Retardos_ms       ; Aporta 2 ciclos máquina.
Retardo_20ms                    ; La llamada "call" aporta 2 ciclos máquina.
        movlw d'20'             ; Aporta 1 ciclo máquina. Este es el valor de "M".
        goto  Retardos_ms       ; Aporta 2 ciclos máquina.
Retardo_10ms                    ; La llamada "call" aporta 2 ciclos máquina.
        movlw d'10'             ; Aporta 1 ciclo máquina. Este es el valor de "M".
        goto  Retardos_ms       ; Aporta 2 ciclos máquina.
Retardo_5ms                     ; La llamada "call" aporta 2 ciclos máquina.
```

```
        movlw  d'5'             ; Aporta 1 ciclo máquina.  Este es el valor de "M".
        goto   Retardos_ms      ; Aporta 2 ciclos máquina.
Retardo_2ms                     ; La llamada "call" aporta 2 ciclos máquina.
        movlw  d'2'             ; Aporta 1 ciclo máquina.  Este es el valor de "M".
        goto   Retardos_ms      ; Aporta 2 ciclos máquina.
Retardo_1ms                     ; La llamada "call" aporta 2 ciclos máquina.
        movlw  d'1'             ; Aporta 1 ciclo máquina.  Este es el valor de "M".

; El próximo bloque "Retardos_ms" tarda:
; 1 + M + M + KxM + (K-1)xM + Mx2 + (K-1)Mx2 + (M-1) + 2 + (M-1)x2 + 2 =
; = (2 + 4M + 4KM) ciclos máquina.  Para K=249 y M=1 supone 1002 ciclos máquina
; que a 4 MHz son 1002 µs = 1 ms.

Retardos_ms
        movwf  R_ContB          ; Aporta 1 ciclo máquina.
R1ms_BucleExterno
        movlw  d'249'           ; Aporta Mx1 ciclos máquina.  Este es el valor de "K".
        movwf  R_ContA          ; Aporta Mx1 ciclos máquina.
R1ms_BucleInterno
        nop                     ; Aporta KxMx1 ciclos máquina.
        decfsz R_ContA,F        ; (K-1)xMx1 cm (cuando no salta) +
                                ; Mx2 cm (al saltar).
        goto   R1ms_BucleInterno ; Aporta (K-1)xMx2 ciclos máquina.
        decfsz R_ContB,F        ; (M-1)x1 cm (cuando no salta) + 2 cm (al saltar).
        goto   R1ms_BucleExterno ; Aporta (M-1)x2 ciclos máquina.
        return                  ; El salto del retorno aporta 2 ciclos máquina.

; En total estas subrutinas tardan:
; - Retardo_200ms:2  + 1 + 2 + (2 + 4M + 4KM)=200007 cm = 200 ms. (M=200 y K=249).
; - Retardo_100ms:2  + 1 + 2 + (2 + 4M + 4KM)=100007 cm = 100 ms. (M=100 y K=249).
; - Retardo_50ms : 2 + 1 + 2 + (2 + 4M + 4KM) =  50007 cm =  50 ms. (M= 50 y K=249).
; - Retardo_20ms : 2 + 1 + 2 + (2 + 4M + 4KM) =  20007 cm =  20 ms. (M= 20 y K=249).
; - Retardo_10ms : 2 + 1 + 2 + (2 + 4M + 4KM) =  10007 cm =  10 ms. (M= 10 y K=249).
; - Retardo_5ms  : 2 + 1 + 2 + (2 + 4M + 4KM) =   5007 cm =   5 ms. (M=  5 y K=249).
; - Retardo_2ms  : 2 + 1 + 2 + (2 + 4M + 4KM) =   2007 cm =   2 ms. (M=  2 y K=249).
; - Retardo_1ms  : 2 + 1     + (2 + 4M + 4KM) =   1005 cm =   1 ms. (M=  1 y K=249).
; RETARDOS  de 0.5 hasta 20 segundos ----------------------------------------------

Retardo_20s                     ; La llamada "call" aporta 2 ciclos máquina.
        movlw  d'200'           ; Aporta 1 ciclo máquina.  Este es el valor de "N".
        goto   Retardo_1Decima  ; Aporta 2 ciclos máquina.
Retardo_10s                     ; La llamada "call" aporta 2 ciclos máquina.
        movlw  d'100'           ; Aporta 1 ciclo máquina.  Este es el valor de "N".
        goto   Retardo_1Decima  ; Aporta 2 ciclos máquina.
Retardo_5s                      ; La llamada "call" aporta 2 ciclos máquina.
        movlw  d'50'            ; Aporta 1 ciclo máquina.  Este es el valor de "N".
        goto   Retardo_1Decima  ; Aporta 2 ciclos máquina.
Retardo_2s                      ; La llamada "call" aporta 2 ciclos máquina.
```

```
        movlw  d'20'            ; Aporta 1 ciclo máquina.  Este es el valor de "N".
        goto   Retardo_1Decima  ; Aporta 2 ciclos máquina.
Retardo_1s                      ; La llamada "call" aporta 2 ciclos máquina.
        movlw  d'10'            ; Aporta 1 ciclo máquina.  Este es el valor de "N".
        goto   Retardo_1Decima  ; Aporta 2 ciclos máquina.
Retardo_500ms                   ; La llamada "call" aporta 2 ciclos máquina.
        movlw  d'5'             ; Aporta 1 ciclo máquina.  Este es el valor de "N".

; El próximo bloque "Retardo_1Decima" tarda:
; 1 + N + N + MxN + MxN + KxMxN + (K-1)xMxN + MxNx2 + (K-1)xMxNx2 +
;   + (M-1)xN + Nx2 + (M-1)xNx2 + (N-1) + 2 + (N-1)x2 + 2 =
; = (2 + 4M + 4MN + 4KM) ciclos máquina.  Para K=249, M=100 y N=1 supone 100011
; ciclos máquina que a 4 MHz son 100011 µs = 100 ms = 0,1 s = 1 décima de segundo.

Retardo_1Decima
        movwf  R_ContC          ; Aporta 1 ciclo máquina.
R1Decima_BucleExterno2
        movlw  d'100'           ; Aporta Nx1 ciclos máquina.  Este es el valor de "M".
        movwf  R_ContB          ; Aporta Nx1 ciclos máquina.
R1Decima_BucleExterno
        movlw  d'249'           ; Aporta MxNx1 ciclos máquina.  Este es valor de "K".
        movwf  R_ContA          ; Aporta MxNx1 ciclos máquina.
R1Decima_BucleInterno
        nop                     ; Aporta KxMxNx1 ciclos máquina.
        decfsz R_ContA,F        ; (K-1)xMxNx1 cm (si no salta) +
                                ; MxNx2 cm (al saltar).
        goto   R1Decima_BucleInterno    ; Aporta (K-1)xMxNx2 ciclos máquina.
        decfsz R_ContB,F        ; (M-1)xNx1 cm (cuando no salta) +
                                ; Nx2 cm (al saltar).
        goto   R1Decima_BucleExterno    ; Aporta (M-1)xNx2 ciclos máquina.
        decfsz R_ContC,F        ; (N-1)x1 cm (cuando no salta) + 2 cm (al saltar).
        goto   R1Decima_BucleExterno2   ; Aporta (N-1)x2 ciclos máquina.
        return                  ; El salto del retorno aporta 2 ciclos máquina.

; En total estas subrutinas tardan:
; - Retardo_20s:    2 + 1 + 2 + (2 + 4N + 4MN + 4KMN) = 20000807 cm = 20 s.
;                   (N=200, M=100 y K=249).
; - Retardo_10s:    2 + 1 + 2 + (2 + 4N + 4MN + 4KMN) = 10000407 cm = 10 s.
;                   (N=100, M=100 y K=249).
; - Retardo_5s:           2 + 1 + 2 + (2 + 4N + 4MN + 4KMN) =  5000207 cm =  5 s.
;                   (N= 50, M=100 y K=249).
; - Retardo_2s:           2 + 1 + 2 + (2 + 4N + 4MN + 4KMN) =  2000087 cm =  2 s.
;                   (N= 20, M=100 y K=249).
; - Retardo_1s:           2 + 1 + 2 + (2 + 4N + 4MN + 4KMN) =  1000047 cm =  1 s.
;                   (N= 10, M=100 y K=249).
; - Retardo_500ms:  2 + 1     + (2 + 4N + 4MN + 4KMN) =   500025 cm = 0,5 s.
;                   (N=  5, M=100 y K=249).
```

5.2 Anexo 2: Software empleado en el sistema PIC-WEB

El código empleado en el sistema PIC-WEB estará grabado en el PIC18F452. A continuación se expone:

- **Programa *picweb.c:***

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#elif defined(__PCH__)
#include <18F452.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
#endif

TRISCbits.TRISC6 = 1;
TRISCbits.TRISC7 = 1;

_serialDataReady = FALSE;
_serialIdx = 0;

baudUSART(
  BAUD_IDLE_CLK_HIGH &
  BAUD_16_BIT_RATE &
  BAUD_WAKEUP_OFF &
  BAUD_AUTO_OFF);

OpenUSART(
  USART_TX_INT_OFF &
  USART_RX_INT_ON &
  USART_ASYNCH_MODE &
  USART_EIGHT_BIT &
  USART_CONT_RX &
  USART_ADDEN_OFF &
  USART_BRGH_HIGH,
  832); // 9600 Baud rate

IPR1bits.RCIP = 1;
PIE1bits.RCIE = 1;

byte tmp;
```

```
if (PIE1bits.RCIE  && PIR1bits.RCIF)   // USART receive interrupt
{
 BACKLIGHT = !BACKLIGHT;

 if (RCSTAbits.OERR)
     {
 tmp = RCREG;  //ReadUSART();   // clear the receiver fifo
 tmp = RCREG;  //ReadUSART();
 RCSTAbits.CREN  = 0;    // clear the OVR flag
 RCSTAbits.CREN  = 1;
 // send (prepare) error message
 putrsUSART("OERR\n");
 return;
     }
     if (RCSTAbits.FERR)
{
 tmp = RCREG;//ReadUSART();   // remove the broken byte
 // send (prepare) error message
 putrsUSART("FERR\n");
 return;
}

 if (_serialIdx  < SERIAL_BUFFER_LENGTH)
 {
 tmp = getcUSART();
 if (tmp != 0x0A && tmp != 0x0D) // \n, \r
 {
  _serialBuffer[_serialIdx]  = tmp;
  _serialIdx++;
  if (_serialIdx  >= SERIAL_BUFFER_LENGTH)
  {
   _serialIdx  = 0;
  }
  putcUSART(tmp);
  sprintf((ram char *)_text, "%d %02X %d %d    ", _serialIdx,  tmp,
RCSTAbits.FERR,  RCSTAbits.OERR);
  FONT_DrawText(0,  0, (far ram char *)&_text[0]);
  RCSTAbits.CREN  = 0;
  tmp = 0;
  RCSTAbits.CREN  = 1;
 }
 else
 {
  _serialDataReady  = TRUE;
  _text[0] = '<';
  _text[1] = _serialBuffer[0];
  _text[2] = '>';
  _text[3] = 0;
```

```
    putsUSART((char*)&_text[0]);
    _serialIdx  = 0;
   }
  }
  //PIR1bits.RCIF  = 0; // is read-only and cleared when reading RCREG
 }

void main()
{
   int s;
   struct sockaddr_in bs,des;
   char resp[255];
   int *sd;

   if (argc == 4)
   {
      // Creamos el socket
      s = socket(AF_INET,SOCK_STREAM,0);

      if (s != -1)
      {
         bs.sin_family  = AF_INET;
         bs.sin_port  = htons(0); //Asigna un puerto disponible dela máquina
         bs.sin_addr.s_addr  = htonl(INADDR_ANY); //Asigna una IP de la máquina

         //Asigna un nombre local al socket
         if ( bind(s,(struct sockaddr*)&bs, sizeof(bs)) != -1)
         {
            //Se prepara el nombre de la máquina remota
            des.sin_family  = AF_INET;
            des.sin_addr.s_addr  = inet_addr(argv[1]);
            des.sin_port  = htons(atoi(argv[2]));

            //Establece la conexión con la máquina remota
            connect(s,(struct sockaddr*)&des,sizeof(des));

            //Envía el mensaje
            send(s,argv[1],strlen(argv[1])+1,0);
            printf("nn->Enviando: %s, a: %s en el puerto: %s n",argv[1], argv[2]);
            //Recibe la respuesta
            recv(s,resp, sizeof(resp) ,0);
            printf("<-Recibido: %sn",resp);
            //Se cierra la conexión (socket)
         close(s);
         }
      }
   }
}
```

5.3 Anexo 3: Software empleado en el servidor de aplicaciones

Primero se presentará el código de la página JSP y seguidamente las clases java utilizadas en el webproject de *Eclipse* para crear la página *proyecto.jsp*.

- **Programa *proyecto.jsp:***

```
<%@ page language="java" import="java.util.*" pageEncoding="ISO-8859-1"%>
<%
String path = request.getContextPath();
String basePath =
request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path
+"/";
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
 <head>
  <base href="<%=basePath%>">

  <title>Proyecto.jsp</title>
   <meta http-equiv="pragma" content="no-cache">
   <meta http-equiv="cache-control" content="no-cache">
   <meta http-equiv="expires" content="0">
   <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
   <meta http-equiv="description" content="This is my page">
   <!--
   <link rel="stylesheet" type="text/css" href="styles.css">
   -->
 </head>
  <body
onload="MM_preloadImages('images/1_inicio2R_s1.jpg','images/2webmap2r_s1.jpg','i
mages/3datos2r_s1.jpg','images/4proyecto2r_s1.jpg','images/5biblioteca2r_s1.jpg','ima
ges/6informe2r_s1.jpg')">
 <div id="contenedor">
 <div id="encabezado">
 <img src="images/cabeza1_s1.jpg" width="215" height="296" /><img
src="images/cabeza2_s1.jpg" width="278" height="296" /><img
src="images/cabeza3_s1.jpg" width="223" height="296" /></div>
 <div id="menulateral"><a href="index.html" onmouseout="MM_swapImgRestore()"
onmouseover="MM_swapImage('Image16','','images/1_inicio2R_s1.jpg',1)"><img
src="images/1_inicio1_s1.jpg" name="Image16" width="215" height="109"
border="0" id="Image16" /></a><a href="webmap.htm"
onmouseout="MM_swapImgRestore()"
onmouseover="MM_swapImage('Image20','','images/2webmap2r_s1.jpg',1)"><img
src="images/2webmap1_s1.jpg" name="Image20" width="215" height="71"
border="0" id="Image20" /></a><a href="datos.htm"
onmouseout="MM_swapImgRestore()"
```

```
onmouseover="MM_swapImage('Image15','','images/3datos2r_s1.jpg',1)"><img
src="images/3datos1_s1.jpg" name="Image15" width="215" height="74" border="0"
id="Image15" /></a><a href="proyecto.htm" onmouseout="MM_swapImgRestore()"
onmouseover="MM_swapImage('Image18','','images/4proyecto2r_s1.jpg',1)"><img
src="images/4proyecto2r_s1.jpg" name="Image18" width="215" height="66"
border="0" id="Image18" /></a><a href="biblioteca.htm"
onmouseout="MM_swapImgRestore()"
onmouseover="MM_swapImage('Image19','','images/5biblioteca2r_s1.jpg',1)"><img
src="images/5biblioteca1_s1.jpg" name="Image19" width="215" height="71"
border="0" id="Image19" /></a><a href="informe.htm"
onmouseout="MM_swapImgRestore()"
onmouseover="MM_swapImage('Image17','','images/6informe2r_s1.jpg',1)"><img
src="images/6informe1_s1.jpg" name="Image17" width="215" height="77"
border="0" id="Image17" /></a><img src="images/teleco_s1.jpg" width="215"
height="154" /></div>
 <div id="contenido">
 <div id="apDiv1">
 <h1>Proyecto </h1>
  <form id="form1" name="form1" method="post" action="">
   <table width="79%" border="0" align="center" cellpadding="5" cellspacing="0">
    <tr>
     <td colspan="4" valign="middle"><p>Introduzca las dimensiones de la
ubicación en cm.</p></td>
    </tr>
    <tr>
     <td width="28%">Dimensión &amp;quot;X&amp;quot;</td>
     <td width="23%"><label for="dimX"></label>
       <input name="dimX" type="text" id="dimX" size="6" /></td>
     <td width="27%"> Dimensión &amp;quot;Y&amp;quot;</td>
     <td width="22%"><label for="dimY"></label>
      <input name="dimY" type="text" id="dimY" size="6" /></td>
    </tr>
    <tr>
     <td colspan="4" align="center" valign="top"><p>
      <label for="texto"></label>
     </p>
      <table border="0" align="left" cellpadding="5" cellspacing="0">
       <tr>
        <td width="10" valign="top" bgcolor="#FFFFFF"><p> </p>
         <p>X</p>
         <p> </p></td>
        <td width="353" valign="top" bgcolor="#FFFFFF"><img
src="images/punto.jpg" width="31" height="35" /></td>
       </tr>
       <tr>
        <td height="42" valign="top" bgcolor="#FFFFFF"> </td>
        <td width="353" align="center" valign="top" bgcolor="#FFFFFF">
Y</td>
```

```
            </tr>
           </table></td>
          </tr>
        </table>
       </form>
       <table width="72%" border="0" align="center" cellpadding="5" cellspacing="0">
        <tr>
         <td width="124" height="119">
         <td width="52" height="99"><button><img src="images/flechaArriba.png"
      width="52" height="99" alt="/*Información del botón/*" /></button></td>
        </tr>
        <tr>
         <td><button><img src="images/flechaIzqda.png" width="99" height="52"
      alt="/*Información del botón/*" /></button></td>
         <td width="52" height="52"><button><img src="images/stop.png" width="57"
      height="56" alt="/*Información del botón/*" /></button></td> 
         <td width="128"><button><img src="images/flechaDerecha.png" width="99"
      height="52" alt="/*Información del botón/*" /></button></td>
        </tr>
        <tr>
         <td> </td>
         <td width="55" height="99"><button><img src="images/flechaAtras.png"
      width="52" height="99" alt="/*Información del botón/*" /></button></td>
        </tr>
       </table>
       </div>
        <div id="pie"><img src="images/pie1_s1.jpg" width="215" height="38" /><img
       src="images/pie2_s1.jpg" width="265" height="38" /><img src="images/pie3_s1.jpg"
       width="236" height="38" /></div>
       </div>
      </body>
      </html>
```

- **Programa *Inicio.java:***

```
public class Inicio extends javax.swing.JFrame {

    public int anchoPantalla=0;
        public int altoPantalla=0;
    public Inicio() {
        initComponents();
    }

    private void initComponents() {
        jLabel1 = new javax.swing.JLabel();
        dimenX = new javax.swing.JTextField();
        dimenY = new javax.swing.JTextField();
        dxLabel = new javax.swing.JLabel();
        dYLabel = new javax.swing.JLabel();
        imagenMapa = new javax.swing.JLabel();
        botonInicio = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Paso 1 ---------->");
        setResizable(false);
        jLabel1.setFont(new java.awt.Font("Arial", 0, 16));
        jLabel1.setText("Introduzca las dimensiones de la ubicaci\u00f3n en cm.");

        dimenX.setColumns(0);
        dimenX.setFont(new java.awt.Font("Arial", 0, 11));
        dimenX.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
        dimenX.setToolTipText("Dimensi\u00f3n X");
        dimenX.setName("");


        dimenY.setColumns(0);
        dimenY.setFont(new java.awt.Font("Arial", 0, 11));
        dimenY.setHorizontalAlignment(javax.swing.JTextField.RIGHT);

        dxLabel.setText("Dimensi\u00f3n X");

        dYLabel.setText("Dimensi\u00f3n Y");

        imagenMapa.setIcon(new javax.swing.ImageIcon("mapa.JPG"));

        botonInicio.setText("Inicio");
        botonInicio.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                botonInicioActionPerformed(evt);
            }
        });
```

```
        org.jdesktop.layout.GroupLayout layout = new
org.jdesktop.layout.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                    .add(layout.createSequentialGroup()
                        .add(20, 20, 20)

.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILING)
                            .add(jLabel1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
353, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                            .add(layout.createSequentialGroup()

.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILING)
                                    .add(botonInicio)
                                    .add(layout.createSequentialGroup()
                                        .add(dxLabel)
                                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                                        .add(dimenX,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 62,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                                    .add(77, 77, 77)
                                    .add(dYLabel)
                                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                                    .add(dimenY,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 59,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                                    .add(9, 9, 9)))
                                .add(15, 15, 15))))
                    .add(layout.createSequentialGroup()
                        .add(75, 75, 75)
                        .add(imagenMapa)))
                .addContainerGap(24, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
                .add(jLabel1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 104,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .add(35, 35, 35)
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                    .add(dxLabel)
                    .add(dYLabel)
```

```
                .add(dimenY, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .add(dimenX, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
            .add(32, 32, 32)
            .add(imagenMapa)
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 49,
Short.MAX_VALUE)
            .add(botonInicio)
            .addContainerGap())
        );
        pack();
    }// </editor-fold>

    private void botonInicioActionPerformed(java.awt.event.ActionEvent evt) {

        try{ //recoge las medidas del plano
                System.out.println("Has pulsado Iniciar");

                anchoPantalla=Integer.parseInt(dimenX.getText());
                altoPantalla=Integer.parseInt(dimenY.getText());
                setVisible(false);

                LocalizacionRobot lr = new LocalizacionRobot(anchoPantalla,altoPantalla);
                //Llama a la clase LocalizacionRobot();

                }catch(Exception e)
                {//Salta excepción si no se han introducido números
                        System.out.println("ERROR: debe introducir unas medidas");
                        e.toString();
                }

    }

    private void dimenXActionPerformed(java.awt.event.ActionEvent evt) {

    }

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Inicio().setVisible(true);
            }
        });
    }
```

```
    private javax.swing.JButton botonInicio;
    private javax.swing.JLabel dYLabel;
    private javax.swing.JTextField dimenX;
    private javax.swing.JTextField dimenY;
    private javax.swing.JLabel dxLabel;
    private javax.swing.JLabel imagenMapa;
    private javax.swing.JLabel jLabel1;
    // End of variables declaration

}
```

- **Programa *Punto.java:***

```
public class Punto
{
	int a,b;
	public Punto(int x, int y)
	{
	a=x;
	b=y;
	}
	public void setX(int x)
	{
		a=x;
	}
	public void setY(int y)
	{
		b=y;
	}
	public int getX()
	{
		return a;
	}
	public int getY()
	{
		return b;
	}
	public String toString()
	{
		return "P("+a+","+b+")";
	}
	public double calcularDistancia(Punto p)
	{
		int difX=Math.abs(a-p.getX());
		int difY=Math.abs(b-p.getY());
		double distPaP=Math.sqrt(Math.pow(difX,2)+Math.pow(difY,2));
		return distPaP;

	}
	public static void main(String args[])
	{
		Punto p0 = new Punto(0,15);
		Punto pa = new Punto(0,201);
		System.out.println(""+pa.toString());
		double distancia = p0.calcularDistancia(pa);
		System.out.println("Distancia: "+distancia);

	}
}
```

- **Programa *LocalizacionRobot.java:***

```
import javax.swing.*;

public class LocalizacionRobot extends javax.swing.JFrame implements ActionListener{

        PrintWriter archivoX,archivoY;
        public int ultimoMovimiento,ultimaAccion;
        public final int masX=1;
        public final int menosX=2;
        public final int masY=3;
        public final int menosY=4;
        public final double VELOCIDAD = 0.010; //Velocidad de 10 cm/seg. , 0.01
cm/ms
        public final int GIRODCHA = 930; //Un segundo en recorrer 90º hacia la derecha
        public final int GIROIZQ = 920; //Un segundo en recorrer 90º hacia la derecha
        public int avanza=1;
        public int atras=2;
        public int giraIzq=3;
        public int giraDcha=4;
        public int para=5;
        public int posX, posY, accion2,puntoX,puntoY;
        public double realX,realY;
        public double inicio,ttotal,tiempo;
        public boolean pulsaAdelante,pulsaAtras,pulsaPara,primeraVez;
        public int [] arrayX;
        public int [] arrayY;
        public Punto [] puntosPlano = new Punto[200];
        public int k,l,ancho,largo;
        private PuertoSerie serial;
        public int indicePuntos;
        static CommPortIdentifier idPuerto;
        static Enumeration listaPuertos;


  /** Creates new form LocalizacionRobot */
  public LocalizacionRobot(int an,int lar){
                super("PFC Antonio Jesús Marcos Díaz");
                // Lista de los puertos disponibles en la máquina. Se carga en el
                // mismo momento en que se inicia la JVM de Java
                indicePuntos=0;
                for(int i=0;i<200;i++)
                        puntosPlano[i]=new Punto(3000,3000);
                primeraVez=true;
                arrayY=new int [lar];
                arrayX=new int [an];
                  listaPuertos = CommPortIdentifier.getPortIdentifiers();
```

```
                    while( listaPuertos.hasMoreElements() ) {
                      idPuerto = (CommPortIdentifier)listaPuertos.nextElement();
                                if( idPuerto.getPortType() ==
CommPortIdentifier.PORT_SERIAL ) {
                                 if( idPuerto.getName().equals("COM6") ) {
                                serial = new PuertoSerie(idPuerto);
                                System.out.println("ha llegado aqui");
                                }
                                 }
                    }
                  //Hasta aquí es lo que hay que añadir para el puerto serie.
                  //Se ha obtenido una instancia de la clase PuertoSerie, después de haber
                  //creado una lista de puertos encontrados en el PC, se intenta abrir el puerto
                  //descrito por el COM1 ó COM3
        ancho=an;
        largo=lar;
        //Se inicializan las variables públicas creadas en esta clase.

        pulsaAdelante=false;
        pulsaAtras=false;
        pulsaPara=false;
        inicio=0;
        ttotal=0;
        ultimoMovimiento=masY;
        ultimaAccion=avanza;
        posX=0;
        posY=0;
        realX=0;
        realY=0;
        //Se crea un contenedor donde se dispondrán todos los elementos gráficos e
    initComponents();
  }

  private void initComponents() {
    adelante = new javax.swing.JButton();
    atr = new javax.swing.JButton();
    dcha = new javax.swing.JButton();
    izq = new javax.swing.JButton();
    par = new javax.swing.JButton();
    cerrar = new javax.swing.JButton();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setEnabled(true);
    setResizable(false);
    adelante.setIcon(new javax.swing.ImageIcon("flechaArriba.JPG"));

    atr.setIcon(new javax.swing.ImageIcon("flechaATras.JPG"));
```

```
        dcha.setIcon(new javax.swing.ImageIcon("flechaDcha.JPG"));
        dcha.setPreferredSize(new java.awt.Dimension(107, 85));

        izq.setIcon(new javax.swing.ImageIcon("flechaizq.JPG"));

        par.setIcon(new javax.swing.ImageIcon("stop_ult.JPG"));

        cerrar.setText("Cerrar");

        org.jdesktop.layout.GroupLayout layout = new
org.jdesktop.layout.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(org.jdesktop.layout.GroupLayout.TRAILING, layout.createSequentialGroup()
                .add(45, 45, 45)
                .add(izq, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 103,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 20,
Short.MAX_VALUE)
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING,
false)
                    .add(org.jdesktop.layout.GroupLayout.TRAILING, atr, 0, 0,
Short.MAX_VALUE)
                    .add(org.jdesktop.layout.GroupLayout.TRAILING, par,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 65,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .add(org.jdesktop.layout.GroupLayout.TRAILING, adelante, 0, 0,
Short.MAX_VALUE))
                .add(22, 22, 22)
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILING)
                    .add(layout.createSequentialGroup()
                        .add(dcha, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 106,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .add(41, 41, 41))
                    .add(layout.createSequentialGroup()
                        .add(cerrar)
                        .addContainerGap())))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
                .addContainerGap()
                .add(adelante)
                .add(17, 17, 17)
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                    .add(izq)
```

```
                .add(par, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 62,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .add(dcha, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 58,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
            .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILING)
                .add(layout.createSequentialGroup()
                    .add(23, 23, 23)
                    .add(atr)
                    .addContainerGap(29, Short.MAX_VALUE))
                .add(layout.createSequentialGroup()
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                    .add(cerrar)
                    .addContainerGap())))
        );
        pack();
            atr.setEnabled(false);
        dcha.setEnabled(false);
        izq.setEnabled(false);
        par.setEnabled(false);
                    adelante.addActionListener(this);
                    atr.addActionListener(this);
                    dcha.addActionListener(this);
                    izq.addActionListener(this);
                    par.addActionListener(this);
                    cerrar.addActionListener(this);

                    setVisible(true);

    }

    /**
     * @param args the command line arguments
     */


            public javax.swing.JButton adelante;
    public javax.swing.JButton atr;
    public javax.swing.JButton cerrar;
    public javax.swing.JButton dcha;
    public javax.swing.JButton izq;
    public javax.swing.JButton par;


public void actionPerformed(ActionEvent ae){


        if (ae.getSource()==adelante)
        {
```

```
            serial.escribir("W");

        try {
        Thread.sleep(50);                //Espera 50 ms entre escribir y leer del puerto serie.
        }
        catch (InterruptedException e) {
            System.out.println("Error en Thread");
        }

            pulsaAdelante=true;

            atr.setEnabled(false);
            dcha.setEnabled(false);
            izq.setEnabled(false);
            par.setEnabled(true);
            adelante.setEnabled(false);

            localiza();

            accion2=avanza;

        }
        if (ae.getSource()==atr)
        {
            serial.escribir("S");
            pulsaAtras=true;

            atr.setEnabled(false);
            dcha.setEnabled(false);
            izq.setEnabled(false);
            par.setEnabled(true);
            adelante.setEnabled(false);

            accion2=atras;

            localiza();
        }
        if(ae.getSource()==dcha)
        {
            serial.escribir("D");

            try {
          Thread.sleep(GIRODCHA);  //Tiempo en girar 90º hacia derecha
         } catch (InterruptedException e) {}
            System.out.println("He dormido 1 seg");

            serial.escribir("  ");
```

```
            accion2=giraDcha;
            System.out.println("Derecha");
            setLocalizacion(giraDcha);
        }
        if(ae.getSource()==izq)
        {
            atr.setEnabled(false);
            dcha.setEnabled(false);
            izq.setEnabled(false);
            par.setEnabled(false);
            adelante.setEnabled(false);

            serial.escribir("A");


            System.out.println("Empiezo...");
            try {
          Thread.sleep(GIROIZQ);  //Tiempo  en girar  90º hacia  izquierda
         } catch (InterruptedException  e) {}
            System.out.println("He  dormido  1 seg");


            serial.escribir("  ");

            atr.setEnabled(true);
            dcha.setEnabled(true);
            izq.setEnabled(true);
            par.setEnabled(false);
            adelante.setEnabled(true);

            accion2=giraIzq;
            System.out.println("Izquierda");
            setLocalizacion(giraIzq);

        }
        if(ae.getSource()==par)
        {       serial.escribir("  ");

            pulsaPara=true;
            atr.setEnabled(true);
            dcha.setEnabled(true);
            izq.setEnabled(true);
            par.setEnabled(false);
            adelante.setEnabled(true);
            localiza();
            setLocalizacion(para);

        }
```

```
    setUltimaAccion(accion2);


    if (ae.getSource()==cerrar)
    {
        System.out.println("Has  pulsao  Cerrar");

//RepresentacionPuntos  rp = new RepresentacionPuntos(ancho,largo,arrayX,arrayY);
System.out.println("Estos  son los puntos******************");
//Pintar  puntos  por los bordes:



        for(int  ggh=0;ggh<200;ggh++)
            if(puntosPlano[ggh].getX()!=3000)
            System.out.println(""+puntosPlano[ggh].toString());

        Reconstruccion  rct = new Reconstruccion(puntosPlano,ancho,largo);


        }
    }//cierra  ActionPermorfed


    public  void setUltimoMovimiento(int  mov)
    {

        ultimoMovimiento=mov;

    }

    public  void setUltimaAccion(int  accion)
    {
    ultimaAccion=accion;
    }
    public  int getUltimaAccion()
    {
        return  ultimaAccion;
    }



public  int getUltimoMovimiento()
{
    return  ultimoMovimiento;
}

public  void inicializaTiempo()
```

```
{
   inicio  = (double)System.currentTimeMillis();
}
public double obtieneTiempo()
{
   ttotal = (double) (System.currentTimeMillis()-inicio);
   System.out.println("Tiempo transcurrido " +ttotal+ "s,"+((double)ttotal/1000)+"seg");
   return ttotal;
}

public void localiza(){

   if (pulsaAdelante==true)
   {
         pulsaAdelante=false;
         inicializaTiempo();


         System.out.println("Avanza");
         setLocalizacion(avanza);

   }//pulsaAdelante
   if (pulsaAtras==true)
   {
         pulsaAtras=false;
         inicializaTiempo();


         setLocalizacion(atras);

   }
   if (pulsaPara==true)
   {
                        int distanciaRecogida;
                        byte [] byteLeido;
                        byte [] recibe;
         try
         {        byteLeido=serial.getLeido();//Obtiene el array de bytes leidos del puerto
serie.
                  recibe= new byte [1]; //Array auxiliar para guardar el carácter recibido.
         pulsaPara=false;
         tiempo=obtieneTiempo();

                  String distanciaSensor  = new String(recibe);
         //       int distanciaRecogida;

                  serial.escribir("  ");
         }catch(Exception e)
```

```
		{
			System.out.println("Error  al recibir");
		}
		pulsaPara=false;
		tiempo=obtieneTiempo();
			try {
		Thread.sleep(50);		//Espera 50 ms entre escribir  y leer del puerto serie.
		}
			catch (InterruptedException  e) {
				System.out.println("Error  en Thread");
			}

			serial.leer();		//Lee puerto Serie si se ha equivocado  al resolver  o no se ha equivocado.
			byteLeido=serial.getLeido();//Obtiene  el array de bytes leidos  del puerto serie.
			recibe= new byte [1]; //Array auxiliar  para guardar el carácter recibido.
			for(int i=0;i<1;i++)		//Bucle  para quedarnos  con lo que nos interesa que es el primer  carácter.
			recibe[i]=byteLeido[i];
			System.out.println("Distancia  Recogida  PRIMERA  vez= "+recibe[0]);

			distanciaRecogida  = (int) recibe[0];
			System.out.println("Distancia  Recogida entero:"  +distanciaRecogida);


			try {
		Thread.sleep(1000);		//Espera 1s entre leer otra medida  del puerto serie.
		}
			catch (InterruptedException  e) {
				System.out.println("Error  en Thread");
			}

			serial.escribir("  ");

			try {
		Thread.sleep(50);		//Espera 50 ms entre escribir  y leer del puerto serie.
		}
			catch (InterruptedException  e) {
				System.out.println("Error  en Thread");
			}

			serial.leer();		//Lee puerto Serie si se ha equivocado  al resolver  o no se ha equivocado.
			byteLeido=serial.getLeido();//Obtiene  el array de bytes leidos  del puerto serie.
			recibe= new byte [1]; //Array auxiliar  para guardar el carácter recibido.
```

```
                for(int i=0;i<1;i++)            //Bucle para quedarnos con lo que nos interesa
que es el primer carácter.
                recibe[i]=byteLeido[i];
                System.out.println("Distancia Recogida segunda vez= "+recibe[0]);

                distanciaRecogida = (int) recibe[0];
                System.out.println("Distancia Recogida entero:" +distanciaRecogida);


                if (distanciaRecogida<0)
                {
                        distanciaRecogida=3000;
                }

        if (getUltimaAccion()==avanza)
        {
                if(getUltimoMovimiento()==masY)
                {

                        realY=realY+VELOCIDAD*tiempo;
                        puntoX=(int)realX;
                        puntoY=(int)(realY+distanciaRecogida);

                }
                if (getUltimoMovimiento()==masX)
                {

                        realX=realX+VELOCIDAD*tiempo;
                        puntoY=(int)realY;
                        puntoX=(int)(realX+distanciaRecogida);
                }
                if (getUltimoMovimiento()==menosY)
                {

                        realY=realY-VELOCIDAD*tiempo;
                        puntoX=(int)realX;
                        puntoY=(int)(realY-distanciaRecogida);
                }
                if (getUltimoMovimiento()==menosX)
                {

                        realX=realX-VELOCIDAD*tiempo;
                        puntoY=(int)realY;
                        puntoX=(int)(realX-distanciaRecogida);
                }


        }//Accion Avanza
```

```
if (getUltimaAccion()==atras)
{
        if(getUltimoMovimiento()==masY)
        {
                realY=realY+VELOCIDAD*tiempo;
            puntoX=(int)realX;
            puntoY=(int)(realY-distanciaRecogida);
        }
        if (getUltimoMovimiento()==masX)
        {
            realX=realX+VELOCIDAD*tiempo;
            puntoY=(int)realY;
            puntoX=(int)(realX-distanciaRecogida);
        }
        if (getUltimoMovimiento()==menosY)
        {
            realY=realY-VELOCIDAD*tiempo;
            puntoX=(int)realX;
            puntoY=(int)(realY+distanciaRecogida);

        }
        if (getUltimoMovimiento()==menosX)
        {
            realX=realX-VELOCIDAD*tiempo;
            puntoY=(int)realY;
            puntoX=(int)(realX+distanciaRecogida);
        }

}
        if((puntoX<0)||(puntoX>1000))
            puntoX=0;
        if((puntoY<0)||(puntoY>1000))
            puntoY=0;
        if(puntoX>ancho)
            puntoX=ancho-1;
        if(puntoY>largo)
            puntoY=largo-1;

        Punto prueba = new Punto(puntoX,puntoY);
        puntosPlano[indicePuntos]= prueba;
        indicePuntos++;

arrayX[k]=puntoX;
arrayY[l]=puntoY;
System.out.println("Prueba"+posX+","+posY);
System.out.println("Situación Real: ("+(int)realX+","+(int)realY+")");
System.out.println("Punto Encontrado:("+arrayX[k]+","+arrayY[l]+")");
k++;
```

```
            l++;
        }//pulsaPara
    }//localizacion

    //*******************************************************
        public void setLocalizacion(int accion){
        switch(accion){//para, avanza, atras, giraDcha, giraIzq
        case 1:
                    if(getUltimaAccion()==para)
                            System.out.println("Debe haber un error");
                    else
                    {
                            if (getUltimoMovimiento()==masY)
                            {
                                    System.out.println("Sigue Adelante");
                                    setUltimoMovimiento(masY);
                            }
                            else if(getUltimoMovimiento()==masX)
                            {
                                    System.out.println("Sigue Adelante");
                                    setUltimoMovimiento(masX);
                            }
                            else if(getUltimoMovimiento()==menosY)
                            {
                                    System.out.println("Sigue Adelante");
                                    setUltimoMovimiento(menosY);
                            }
                            else if(getUltimoMovimiento()==menosX)
                            {
                                    System.out.println("Tuerce izquierda");
                                    setUltimoMovimiento(menosX);
                            }
                    if (getUltimaAccion()==atras)
                    {
                            if(getUltimoMovimiento()==masY)
                            {
                                    System.out.println("Sentido contrario");
                                    setUltimoMovimiento(menosY);
                            }
                            else if(getUltimoMovimiento()==masX)
                            {
                                    System.out.println("Sentido contrario");
                                    setUltimoMovimiento(menosX);
                            }
                            else if(getUltimoMovimiento()==menosY)
                            {
                                    System.out.println("Sentido contrario");
                                    setUltimoMovimiento(masY);
```

```
                    }
                    else if(getUltimoMovimiento()==menosX)
                    {
                            System.out.println("Sentido  contrario");
                            setUltimoMovimiento(masX);
                    }
             }//ultimaAccion-->Atras
            }
break;

      //************************************************
case 3:
            if (getUltimoMovimiento()==masY)
            {
                    setUltimoMovimiento(masX);
            }
            else if (getUltimoMovimiento()==masX)
            {
                    setUltimoMovimiento(menosY);
            }
            else if (getUltimoMovimiento()==menosY)
            {
                    setUltimoMovimiento(menosX);
            }
            else if (getUltimoMovimiento()==menosX)
            {
                    setUltimoMovimiento(masY);
            }
//      setUltimaAccion(accion);
break;
      //************************************************
case 4:
            if (getUltimoMovimiento()==masY)
            {
                    System.out.println("debe  restar x");
                    setUltimoMovimiento(menosX);
            }
            else if (getUltimoMovimiento()==masX)
            {
                    setUltimoMovimiento(masY);
            }
            else if (getUltimoMovimiento()==menosY)
            {
                    setUltimoMovimiento(masX);
            }
            else if (getUltimoMovimiento()==menosX)
            {
                    setUltimoMovimiento(menosY);
```

```
                }
        break;
        case 2: //Atrás
        if (getUltimaAccion()==atras)
                {
                if (getUltimoMovimiento()==masY)
                {
                        setUltimoMovimiento(masY);
                }
                else if (getUltimoMovimiento()==masX)
                {
                        setUltimoMovimiento(masX);
                }
                else if (getUltimoMovimiento()==menosY)
                {
                        setUltimoMovimiento(menosY);
                }
                else if (getUltimoMovimiento()==menosX)
                {
                        setUltimoMovimiento(menosX);
                }

                }//ultimaAccion-->Atras
        if (getUltimaAccion()==avanza)
            {
                if (getUltimoMovimiento()==masY)
                {
                        setUltimoMovimiento(menosY);
                }
                else if (getUltimoMovimiento()==masX)
                {
                        setUltimoMovimiento(menosX);
                }
                else if (getUltimoMovimiento()==menosY)
                {
                        setUltimoMovimiento(masY);
                }
                else if (getUltimoMovimiento()==menosX)
                {
                        setUltimoMovimiento(masX);
                }
        }
            break;
        case 5:

        }
        }     //setLocalizacion
}
```

- **Programa *Servidor.java:***

```
import java.io.*;

public class Servidor extends JFrame {
  private JTextField campoIntroducir;
  private JTextArea areaPantalla;
  private ObjectOutputStream salida;
  private ObjectInputStream entrada;
  private ServerSocket servidor;
  private Socket conexion;
  private int contador = 1;

  // configurar GUI
  public Servidor()
  {
    super( "Servidor" );

    Container contenedor = getContentPane();

    // crear campoIntroducir y registrar componente de escucha
    campoIntroducir = new JTextField();
    campoIntroducir.setEditable( false );
    campoIntroducir.addActionListener(
      new ActionListener() {

        // enviar mensaje al cliente
        public void actionPerformed( ActionEvent evento )
        {
          enviarDatos( evento.getActionCommand() );
          campoIntroducir.setText( "" );
        }
      }
    );

    contenedor.add( campoIntroducir, BorderLayout.NORTH );

    // crear areaPantalla
    areaPantalla = new JTextArea();
    contenedor.add( new JScrollPane( areaPantalla ),
      BorderLayout.CENTER );

    setSize( 300, 150 );
    setVisible( true );

  } // fin del constructor de Servidor

  // configurar y ejecutar el servidor
```

```
public void ejecutarServidor()
{
  // configurar servidor para que reciba conexiones; procesar las conexiones
  try {

    // Paso 1: crear un objeto ServerSocket.
    servidor = new ServerSocket( 12345, 100 );

    while ( true ) {

      try {
        esperarConexion(); // Paso 2: esperar una conexión.
        obtenerFlujos();       // Paso 3: obtener flujos de entrada y salida.
        procesarConexion(); // Paso 4: procesar la conexión.
      }

      // procesar excepción EOFException cuando el cliente cierre la conexión
      catch ( EOFException excepcionEOF ) {
        System.err.println( "El servidor terminó la conexión" );
      }

      finally {
        cerrarConexion();   // Paso 5: cerrar la conexión.
        ++contador;
      }

    } // fin de instrucción while

  } // fin del bloque try

  // procesar problemas con E/S
  catch ( IOException excepcionES ) {
    excepcionES.printStackTrace();
  }

} // fin del método ejecutarServidor

// esperar que la conexión llegue, después mostrar información de la conexión
private void esperarConexion() throws IOException
{
  mostrarMensaje( "Esperando una conexión\n" );
  conexion = servidor.accept(); // permitir al servidor aceptar la conexión
  mostrarMensaje( "Conexión " + contador + " recibida de: " +
    conexion.getInetAddress().getHostName() );
}

// obtener flujos para enviar y recibir datos
private void obtenerFlujos() throws IOException
```

```
{
  // establecer flujo de salida para los objetos
  salida = new ObjectOutputStream( conexion.getOutputStream() );
  salida.flush(); // vaciar búfer de salida para enviar información de encabezado

  // establecer flujo de entrada para los objetos
  entrada = new ObjectInputStream( conexion.getInputStream() );

  mostrarMensaje( "\nSe recibieron los flujos de E/S\n" );
}

// procesar la conexión con el cliente
private void procesarConexion() throws IOException
{
  // enviar mensaje de conexión exitosa al cliente
  String mensaje = "Conexión exitosa";
  enviarDatos( mensaje );

  // habilitar campoIntroducir para que el usuario del servidor pueda enviar mensajes
  establecerCampoTextoEditable( true );

  do { // procesar los mensajes enviados por el cliente

    // leer el mensaje y mostrarlo en pantalla
    try {
      mensaje = ( String ) entrada.readObject();
      mostrarMensaje( "\n" + mensaje );
    }

    // atrapar problemas que pueden ocurrir al tratar de leer del cliente
    catch ( ClassNotFoundException excepcionClaseNoEncontrada ) {
      mostrarMensaje( "\nSe recibió un tipo de objeto desconocido" );
    }

  } while ( !mensaje.equals( "CLIENTE>>> TERMINAR" ) );

} // fin del método procesarConexion

// cerrar flujos y socket
private void cerrarConexion()
{
  mostrarMensaje( "\nFinalizando la conexión\n" );
  establecerCampoTextoEditable( false ); // deshabilitar campoIntroducir

  try {
    salida.close();
    entrada.close();
    conexion.close();
```


```
      }
      catch( IOException excepcionES ) {
        excepcionES.printStackTrace();
      }
    }

    // enviar mensaje al cliente
    private void enviarDatos( String mensaje )
    {
      // enviar objeto al cliente
      try {
        salida.writeObject( "SERVIDOR>>> " + mensaje );
        salida.flush();
        mostrarMensaje( "\nSERVIDOR>>> " + mensaje );
      }

      // procesar problemas que pueden ocurrir al enviar el objeto
      catch ( IOException excepcionES ) {
        areaPantalla.append( "\nError al escribir objeto" );
      }
    }

    // método utilitario que es llamado desde otros subprocesos para manipular a
    // areaPantalla en el subproceso despachador de eventos
    private void mostrarMensaje( final String mensajeAMostrar )
    {
      // mostrar mensaje del subproceso de ejecución despachador de eventos
      SwingUtilities.invokeLater(
        new Runnable() { // clase interna para asegurar que la GUI se actualice
apropiadamente

          public void run() // actualiza areaPantalla
          {
            areaPantalla.append( mensajeAMostrar );
            areaPantalla.setCaretPosition(
              areaPantalla.getText().length() );
          }

        } // fin de la clase interna

      ); // fin de la llamada a SwingUtilities.invokeLater
    }

    // método utilitario que es llamado desde otros subprocesos para manipular a
    // campoIntroducir en el subproceso despachador de eventos
    private void establecerCampoTextoEditable( final boolean editable )
    {
      // mostrar mensaje del subproceso de ejecución despachador de eventos
```

```
    SwingUtilities.invokeLater(
      new Runnable() { // clase interna para asegurar que la GUI se actualice
apropiadamente

        public void run() // establece la capacidad de modificar a campoIntroducir
        {
          campoIntroducir.setEditable( editable );
        }

      } // fin de la clase interna

    ); // fin de la llamada a SwingUtilities.invokeLater
  }

  public static void main( String args[] )
  {
    Servidor aplicacion = new Servidor();
    aplicacion.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
    aplicacion.ejecutarServidor();
  }

}
```

- **Programa *Reconstruccion.java:***

```
import java.awt.*;

public class Reconstruccion extends JFrame
{
	Punto [] array;
	int a,b;
	public Reconstruccion(Punto [] p,int xSize, int ySize)
	{
	super("Plano Obtenido ------->");
	a = xSize; //tamaño del plano
	b = ySize; //
	array=p;	//array de puntos que se encuentran
	setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	getContentPane().setBackground(Color.WHITE);

	setSize(a+9,b+39); //Establece tamaño de pantalla
	setVisible(true); //Se hace visible la pantalla

	}//constructor


	public void paint(Graphics g) //Método de dibujo de gráficos
	{

	super.paint(g);

	setLocation(420,0); //Se establece la posición de la pantalla

	//Se dibujan las líneas que separan el plano
	g.setColor(Color.LIGHT_GRAY);
	for(int i =0;i<(a+5);i+=40)
		for(int j=0;j<(b+32);j++)
		{
			g.fillOval(i+5,j+32,1,1);
			g.fillOval(j+5,i+32,1,1);

		}

	g.setColor(Color.BLACK);

	//Se dibujan todos los puntos encontrados en la pantalla
	g.setColor(Color.RED);
	for(int i =0;i<array.length;i++)
		g.fillOval(array[i].getX()+5,array[i].getY()+32,3,3);

	//g.fillOval(80+5,120+32,5,5);
	//g.fillOval(140+5,120+32,5,5);
```

```
//g.fillOval(80+5,160+32,5,5);
//g.fillOval(140+5,160+32,5,5);

//g.fillOval(180+5,32,5,5);
//g.fillOval(180+5,40+32,5,5);
//g.fillOval(238+5,32,5,5);
//g.fillOval(238+5,40+32,5,5);

        g.setColor(Color.BLACK);
double []][] distancias  = new double[array.length][array.length];
double  disMin=3000;

/*
Se guarda  en el array de distancias  las distancias  que hay entre
los puntos
*/
for(int  j=0;j<(array.length);j++)
{
        disMin=3000;
        for(int  i = 0;i<(array.length);i++)
        {
                if((array[i].getX()==3000)&&(array[i].getY()==3000))
                        System.out.print("");
                else
                {
                if (i==j)
                        distancias[j][i]=disMin;
                else
                {
                        distancias[j][i]=array[j].calcularDistancia(array[i]);

                }//else
                }
        }//for

}//for

double  [][] distFinal  = new double[array.length][array.length];

for(int  i = 0;i<(array.length);i++)
        for(int  j = 0;j<(array.length);j++)
                distFinal[i][j]=distancias[i][j];
/*
Se declaran  los arrays de distancias  mínimas  donde en el primer  se guardará
la distancia  de un punto al más próximo.  Y en arrayMin2  se guardará  la segunda
mínima  distancia  de un punto a otro.
*/
```

```
double arrayMin [] = new double[array.length];
double arrayMin2 [] = new double[array.length];

//Cálculo de la distancia mímina

for(int j=0;j<(array.length);j++)
{
        disMin=3000;
        for(int i=0;i<(array.length);i++)
                if((i!=j)&&(distancias[j][i]!=0.0))
                        if (distancias[j][i]<disMin)
                        if(distancias[j][i]>10)
                        {

                                disMin=distancias[j][i];
                                arrayMin[j]=disMin;
                        }
}

for(int i = 0; i<array.length;i++)
        for(int j=0;j<array.length;j++)
                if(arrayMin[i]==distancias[i][j])
                        distancias[i][j]=3000;

//Cálculo de la segunda distancia mínima

for(int j=0;j<(array.length);j++)
{
        disMin=3000;
        for(int i=0;i<(array.length);i++)
                if((distancias[j][i]!=0))
                        if (distancias[j][i]<disMin)
                        if(distancias[j][i]>10)
                        {

                                disMin=distancias[j][i];
                                arrayMin2[j]=disMin;
                        }
        }

/*
Dibuja la línea que une cada punto con el punto correspondiente
con su distancia mínima
*/
for(int i=0;i<array.length;i++)
        for(int j=0;j<array.length;j++)
                if ((distFinal[i][j]==arrayMin[i])&&(distFinal[i][j]<25))
                {
```

```
        g.drawLine(array[i].getX()+5,array[i].getY()+32,array[j].getX()+5,array[j].getY()+
32);
                        }
        /*
        Dibuja la línea que une cada punto con el punto correspondiente
        con su segunda distancia más próxima
        */
        for(int i=0;i<array.length;i++)
                for(int j=0;j<array.length;j++)
                        if ((distFinal[i][j]==arrayMin2[i])&&(distFinal[i][j]<25))
                        {

        g.drawLine(array[i].getX()+5,array[i].getY()+32,array[j].getX()+5,array[j].getY()+
32);
                        }


        }//Graphics

}//class
```