Sistemas Informáticos
Curso 2005-2006

ROBOT GUÍA DEL MUSEO GARCÍA SANTESMASES

José Manuel Maldonado Becerra
Raúl García Sánchez
Pablo García-Escudero Barreras

Dirigido por:
Dr. D. José Antonio López Orozco
Dr. D. José Manuel Mendías Cuadros
Dpto. Arquitectura de Computadores y Automática (DACYA)

Facultad de Informática
Universidad Complutense de Madrid

Índice

Resumen

El proyecto consiste en la realización de un robot autónomo capaz de convivir con los visitantes del museo García Santesmases y actuar como guía de los usuarios que lo requieran.

Para ello, el robot ha de conocer su propia posición y orientación dentro del museo en todo momento, así como poder calcular el camino a seguir para alcanzar cualquier pieza que se encuentre expuesta en éste. Además ha de poseer información acerca de cada una de estas piezas para informar sobre ellas a los visitantes.

Debido al tamaño y complejidad del proyecto, éste ha sido dividido en dos partes. La parte correspondiente a nuestro grupo de trabajo gira en torno al funcionamiento a bajo nivel del robot, es decir, la parte hardware y la programación en contacto directo con el hardware, que principalmente basa en el uso de un PIC encargado del control directo de las diferentes partes físicas y sensores del robot.

A continuación se muestran las partes básicas en que se divide el trabajo con el hardware:

- Control de los motores
- Comunicación entre el PIC y el PC
- Control de los sensores
    1. Infrarrojos
    2. Ultrasonidos
    3. Brújula
    4. Encoders
- Alimentación: autonomía del robot, carga, componentes…

Además del trabajo con el hardware, también hemos desarrollado, ya sobre el PC, un software encargado de enlazar nuestra parte con la de más alto nivel. Sus partes más significativas son:

- El módulo de comunicación con el PIC, encargado de la comunicación entre el PC y el PIC
- Los módulos de motores y planificador de bajo nivel, encargados del manejo de las velocidades de los motores para trazar las trayectorias que le marcan los módulos de más alto nivel
- Los módulos de sensores, encargados de filtrar y procesar los datos proporcionados por el PIC y suministrarlos al resto de módulos. Ya sea detección y marcaje de obstáculos, avance de las ruedas u orientación del robot
- El módulo de posición, encargado de calcular la posición y orientación del robot en todo momento a partir de la información de los sensores, para así poder seguirle la pista en todo momento mediante un mapa interno

Abstract

The goal of this project is to build an independent robot able to live with the Garcia Santesmases museum visitors, who may use the guiding features of it.

To achieve this, the robot must know his own position and orientation to trace an optimum path towards any object exposed in the museum of which it must have information to tell the visitors.

Because of the project size and complexity, it has been divided into two parts. The one corresponding to our team is about all the hardware and low-level programming of the robot. That is, all the electronic devices which compose the robot physically, and MicroChip PIC devices programming to interact with these devices and sensors.

We have organized all the work in four different areas:

- Motor handling
- Low Level communications (PC - PIC - PC)
- Sensor monitoring
    1. Infrarred
    2. Ultrasound
    3. Digital compass
    4. Optic encoders
- Power supplies (independence, battery charging, ...)

We have also developed some software pieces, for the PC platform, able to connect our low-level logic with the upper level modules, also divided into four areas:

- Communications module, used to communicate the PC and the PIC microcontrollers.
- Motors module and low-level planning module, used to handle the motors' speeds to trace the paths required by higher level modules.
- Sensors modules, used to filter and process all the data collected by the PIC and forward it to the rest of the modules, that is, object tracking and detection modules, orientation modules...
- Position module, used to process with the data collected by the sensors, the orientation and position of the robot in any moment tracking the robot's movements using an internal representation of the environment where the robot lives in.

CAPÍTULO 1: Introducción

Otros robots construidos en España

En primer lugar veremos otros proyectos similares al nuestro que ya han sido realizados como el robot guía realizado por la Universidad Politecnica de Madrid hace unos años que es bastante similar en cuanto a objetivos a realizar por nuestra parte y un robot realizado por un particular que es capaz de moverse y comunicarse con otras personas, en este caso nos interesa mas explicar la parte hardware para compararla con la nuestra.

1. **Robot inteligente creado por la UPM** cuyo objetivo es servir de guía.

   Urbano, es un robot construido y diseñado por un grupo de profesores de la UPM que es capaz de realizar guías un robot 'casi inteligente' capaz de expresar en su rostro la tristeza o alegría que 'siente' ante determinados estímulos.

   El gesto de la 'cara' de Urbano parece triste cuando sus baterías están poco cargadas o si algo le impide hacer su trabajo de guía 'con normalidad', es decir, cuando el trasiego del público dificulta su desplazamiento por los pasillos del pabellón.

   Además, este robot que ha supuesto cuatro años de trabajo a sus creadores, profesores de los departamentos de Ingeniería Electrónica, Automática e Informática Industrial, es capaz de reconocer la presencia del visitante, saludarlo y presentarse.

   Su sistema de navegación, al igual que el que nosotros tenemos se basa en una red de sensores infrarrojos y de ultrasonidos, que le permite detectar los obstáculos durante sus desplazamientos.

   Además de poder ser operado mediante lenguaje hablado y de manera presencial, se puede interaccionar con él a través de Internet y a distancia.

2. **Robot seguritron:** este es un robot que ha sido creado por un técnico español que no ha sido patrocinado por ninguna empresa, gobierno ni universidad, aproximadamente tardo 10 años en realizarlo y ha sido objeto de múltiples premios internacionales, a continuación pasaremos a realizar una descripción de dicho proyecto.

Su altura es aproximadamente la de un ser humano este robot es capaz de agacharse y elevarse por medio de un actuador motorizado que acciona la articulación de la cintura, es capaz de desplazarse en todas direcciones mediante un a plataforma mediante tracción con orugas, esta provisto de 2 motores potentes de 4 polos y una caja reductora, la alimentación se realiza mediante unas baterías, de Plomo-Calcio 12 Voltios y 35 Amperios y las otras baterías auxiliares de Níkel-Cadmio y 6 Voltios 10 Amperios proporcionando la energía suficiente para una autonomía de varias horas de funcionamiento, en nuestro caso el robot estará alimentado con unas baterías de 8.4 Voltios y 5 Amperios.

La cintura tiene dos movimientos: El de giro, que es de 180º grados, trasmitido por una caja reductora de engranajes de acero con transmisión por cadena. El de inclinación, con 30º grados de ángulo. Dispone para este movimiento, de un amortiguador hidráulico de compensación de fuerza.

Las manos también están servo motorizadas. Tienen tres dedos que se abren y cierran. Puede coger objetos de diversas formas y pesos. Cuenta con un sistema electrónico de regulación de par de fuerza, que impide hacer daño si aplicara excesiva presión.

La cabeza es uno de los elementos mecánicamente más complejo e incorpora los siguientes movimientos: De oscilación con 180º grados en horizontal y el movimiento vertical de 30º grados de cabeceo con dos amortiguadores de compensación. Los ojos son azules, en relieve y brillantes, puede moverlos horizontalmente mediante un micro servo. Las pestañas oscilan verticalmente accionadas por otro servo motor. El altavoz principal del sintetizador vocal está ubicado en el interior y en su parte superior dispone de un foco de luz infrarroja para iluminación. Formando la boca tiene una hilera de diodos led que se encienden modulados por el sonido de la voz.

El cuerpo agrupa los elementos mas importantes para el funcionamiento, es decir, el sistema de comunicación inalámbrica multi canal Duplex, las antenas de radio, los controles de audio y vídeo, los procesadores y decodificadores de canales, los fusibles generales de protección y los motores del giro de hombros y cintura.

Sus principales características son:

Esta construido de forma modular, con materiales muy sólidos y ligeros, puede desplazarse por casi cualquier tipo de terreno, mueve los brazos, los hombros, las manos, la cintura, la cabeza, los ojos y las cejas, todos los movimientos puede realizarlos simultáneos, es decir que un movimiento no impide la ejecución de otros.

Por supuesto, también puede hablar y mantener una conversación con su propia voz, mediante técnicas de inteligencia artificial de una forma interactiva con la voz del controlador-supervisor, nuestro robot también es capaza de comunicarse mediante voz, todavía no de forma ‘inteligente’ pero si al llegar a algún objetivo predefinido.

Arquitectura modular de la implementación

Uno de los principales aspectos a resaltar de nuestro proyecto es el alto grado de modularidad del trabajo de programación realizado. Así, futuros cambios en el hardware del robot u optimizaciones sólo afectarán a módulos bien diferenciados.

Además, estos módulos se han hecho de la forma más genérica posible y son altamente configurables fuera de la programación mediante archivos XML. De modo que la mayoría de cambios de hardware no requerirán más que pequeños ajustes en estos archivos de configuración que podrán ser realizados sin tener un conocimiento avanzado del funcionamiento interno del módulo correspondiente.

Todo el código correspondiente a la arquitectura ha sido desarrollado en C++ y usando las librerías “Glib”, lo que hace que pueda ser compilado para otros sistemas operativos (de momento la arquitectura corre en Windows). Esto permite plantearse una futura migración a sistemas UNIX. El código que ejecuta el PIC ha sido programado en C.

Estas características hacen que la arquitectura sea portable a otros robots con pocos cambios en la programación y estando ésta bien asilada en algunos módulos.

La comunicación entre módulos se ha intentado reducir al mínimo para hacer a éstos lo más independientes entre si. Aun así, debido a la magnitud del proyecto, ésta ha llegado a alcanzar cierto grado de complejidad. A continuación mostramos un diagrama con la estructura de módulos correspondiente a la parte de bajo nivel y la comunicación entre ellos, así como la comunicación con la parte de alto nivel (módulos amarillos):

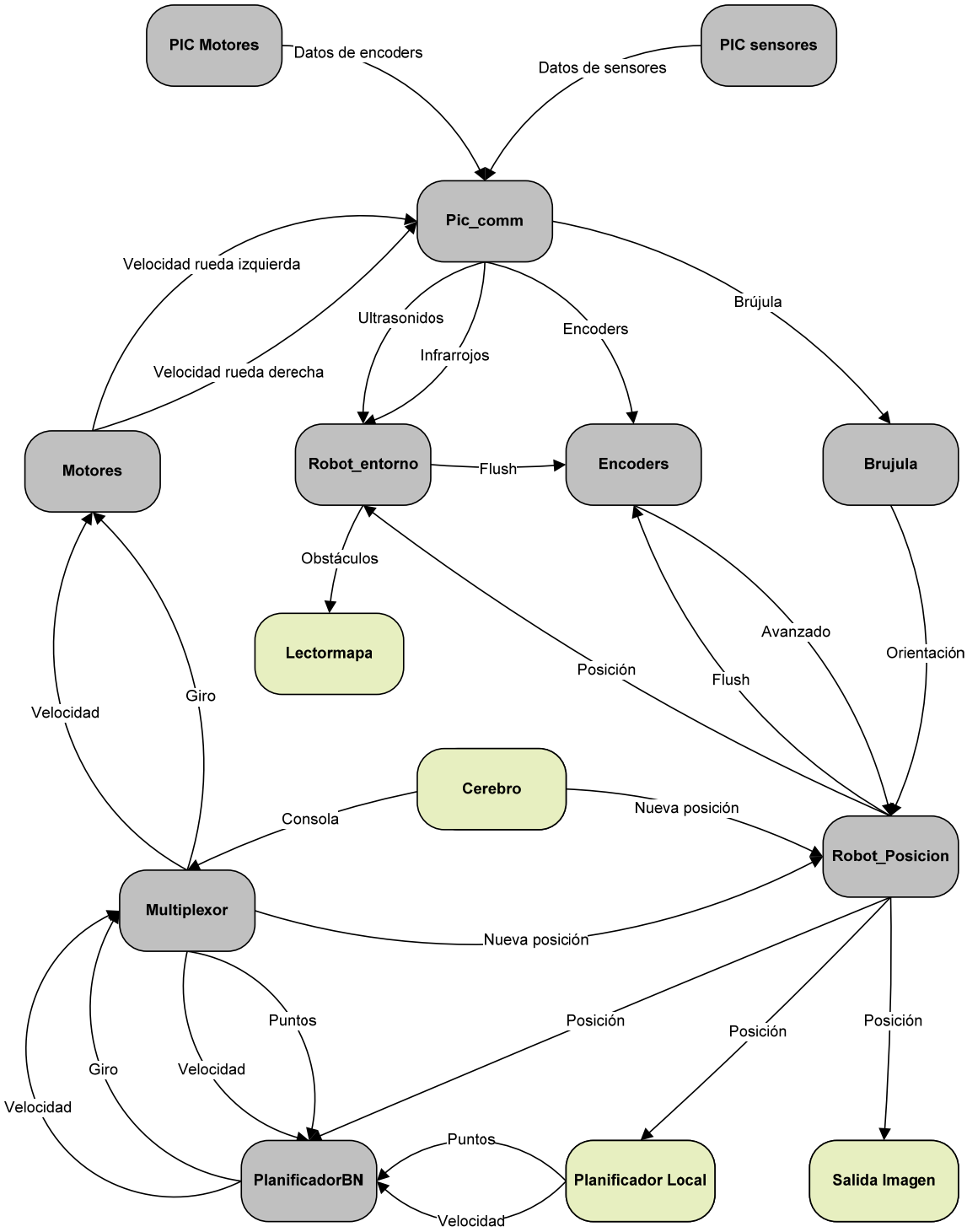


Figura 1 diagrama de módulos

Datos técnicos

En este apartado comentaremos brevemente los dispositivos electrónicos empleados en el robot, ya que los ampliaremos más adelante.

Los motores empleados son el modelo 63.129 de Kelvin, con una potencia nominal de 90W. Alimentados a una tensión de 12V son capaces de girar en velocidad pico a 2000rpm. Seleccionamos estos motores debido al requisito de gran potencia ya que calculamos que el conjunto del robot terminaría pesando unos 25Kg. En la práctica hemos podido comprobar que estos motores son capaces de mover cargas bastante mayores.

Dichos motores tienen dos peculiaridades, la primera es que tienen conectada una reductora del tipo 1:50 (se necesitan 50 vueltas de eje de motor para completar una vuelta de rueda) lo cual nos permite aumentar el par del motor y también reducir la velocidad pico del motor. Por otro lado los motores tienen acoplados a su eje unos encoders (mod. HEDS-5540) de 500 pulsos por vuelta, que junto con la reductora nos dan una resolución de 0.002cm por cada pulso de encoder emitido.

Estos motores están operados a través de la controladora MD22, capaz de controlar dos motores a diferentes velocidades y en los dos sentidos de giro. El motivo de la selección de esta controladora fue la imposibilidad de controlar los motores mediante PWM y un puente en H usando los circuitos integrados L293 debido a las altas temperaturas que alcanzaban dichos integrados. Por otro lado, la controladora es capaz de soportar tensiones de hasta 10A y además posee un gran condensador que protege las baterías de alimentación de picos de esfuerzo puntuales.

Para regular la velocidad y el sentido de los motores a través de la controladora, empleamos dos DAC's MAX515 (Conversor Analógico Digital) de 10 bits de resolución y con un rango de voltaje de salida comprendido entre los 0v y los 5v. Estos 5v los conseguimos mediante una modificación del circuito propuesto por el fabricante de los DAC, ya que este sólo permitía obtener voltajes de 0v a 4.096v.

Los sensores encargados de monitorizar el entorno del robot son principalmente, los sensores de ultrasonidos (mod. SRF05) capaces de medir distancias desde los 7.5cm hasta los 3m, activados mediante una señal de disparo emitida por el PIC. Y los sensores infrarrojos analógicos (mod. GP2D12) capaces de detectar objetos desde los 8cm a 80cm. Estos últimos no tienen señal de disparo por lo que siempre están monitorizando el entorno.

Además contamos con una brújula digital (mod. CMPS03) cuestionada a través de PWM, que nos indica en función de la anchura del pulso, los grados de desviación con el norte. Estos datos junto con los encoders son lo que empleamos para orientar en robot cuando está en funcionamiento.

Todos estos dispositivos están gobernados por micro controladores PIC (mod. 18F4550) de MicroChip, que, entre sus características, disponen de USB integrado, módulos de captura y generación de PWM, módulos conversores analógico digitales y 32Kb de memoria flash para almacenar más de 16000 instrucciones de palabra.

Para el correcto funcionamiento de estos PIC's hemos empleado placas de ejecución Picdem, en su versión más reducida, sólo incluyendo los elementos indispensables para que cada PIC pueda desempeñar la tarea para la que se programó.

Es importante hacer notar que estas placas, además de ser placas de ejecución, también permiten la programación del PIC sin tener que cambiar nada. Para ello, compilamos un pequeño Bootloader y lo grabamos en la flash del PIC como parte del firmware. Este programa es básicamente un intérprete, que permite que el PIC auto programe la memoria flash.

Por otro lado, la comunicación del PIC con el PC se realiza físicamente a través del puerto USB , pero lógicamente se hace a través de un emulador de puerto serie, que también está compilado e integrado en el firmware del PIC. La razón de optar por este diseño es básicamente la simplicidad a la hora de escribir y leer del puerto, ya que a todos los efectos se trata de un puerto COM, y esta decisión se tomó una vez que comprobamos que los datos a enviar a través del puerto tenían caudal suficiente para no saturar el canal.

Este emulador firmware de puertos COM se conoce con el nombre de CDC y lo ofrece MicroChip en su página web. Para integrar la compatibilidad con puertos serie virtuales, no hay más que agregar los ficheros fuente al proyecto del PIC y configurar todas las opciones en las cabeceras de dichas fuentes.

Estas placas Picdem se conectan ambas al Pcbox integrado en el nivel inferior de la estructura del robot, que es donde se ejecuta el cerebro de alto nivel del robot.
Este Pcbox se trata en realidad de un ordenador portátil pero de dimensiones y especificaciones diferentes. El procesador es un Intel Centrino y dispone de varias ranuras de expansión PCI, por otro lado, no tiene WiFi integrada, por lo que hubo que acoplar un adaptador WiFi USB para darle una autonomía total en lo que a cables se refiere. La alimentación se realiza a través de dos baterías de 12v cada una conectadas en serie a través de una placa de carga y distribución.

Dicha placa permite, por un lado, obtener 24v para la alimentación del Pcbox, y 12v por otro para la alimentación de la circuitería del robot.
Mediante un conmutador se puede seleccionar la batería que proporciona voltaje a la circuitería. Además, la placa también contiene un circuito de carga de las baterías, el cual, aísla la parte de alimentación del PC cuando las baterías se están cargando.

Del mismo modo también tenemos otra placa de alimentación y carga para los motores, ya que, estimamos necesario separar la alimentación del PC de la de los motores, ya que se podrían generar ruidos capaces de alterar el funcionamiento del robot y en última instancia, del PC. En la salida de alimentación de esta placa, se ha conectado en serie un fusible de 10A para proteger la controladora en caso de un pico de corriente debido a un sobre esfuerzo de los motores.

Como complemento a los sensores, también hemos diseñado unas placas de multiplexión para permitir, tener un mayor número de sensores conectados simultáneamente al PIC y reducir el número de terminales de conexión empleados para comunicar el PIC con los sensores.

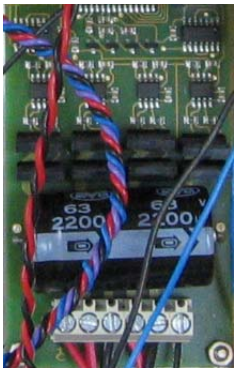


Figura 2 Controladora MD22

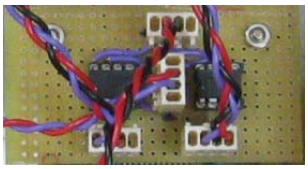


Figura 3 Placa conexión DAC's

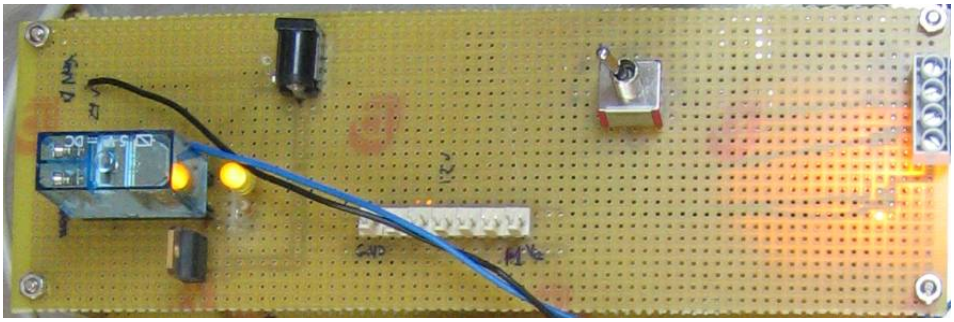


Figura 4 Placa de alimentación y carga

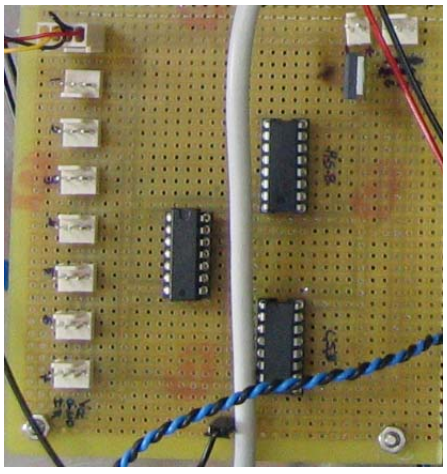


Figura 4 Placa de multiplexación de infrarrojos

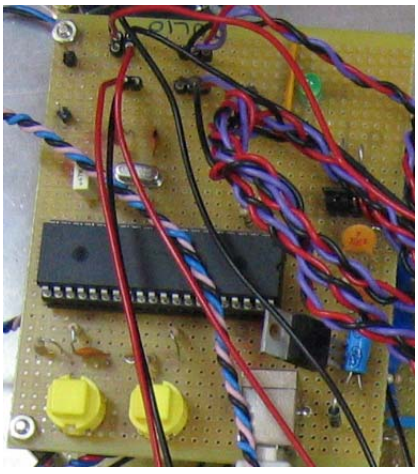


Figura 5 Placa de ejecución Picdem

Módulo de comunicación PC - PIC

Este módulo es el encargado de hacer llegar las órdenes enviadas desde los módulos de más alto nivel hacia el PIC, que es el que en última instancia realiza las operaciones con los sensores, motores…

Para ello debe conocer en que puerto serie virtual se encuentra cada PIC, ya que al disponer de varios microcontroladores cada uno ocupándose de una tarea, necesitamos varios puertos y conocer unívocamente que PIC se encuentra conectado a cada uno.

La conexión hacia los PICs se hace a través de un intérprete serie grabado en el firmware de los PICs de forma que, aunque la conexión física al ordenador es a través de USB, el intérprete hace creer al sistema operativo que en realidad se trata de un puerto serie (al cual tenemos que proporcionarle unos drivers previamente modificados).

Uno de los inconvenientes que presenta dicho interprete es que sólo permite el envío de cadenas de caracteres a través del puerto virtual, de forma que, si queremos enviar datos numéricos, tenemos que hacer una conversión a caracteres, enviarla a través del puerto, y en el receptor volver a realizar la conversión a datos numéricos.

Es por esta razón que la mayoría de las ordenes que se envían al PIC se hacen en forma de códigos de comando, indicando un carácter de control de comando (@ en caso de tratarse de comandos de control de algún dispositivo) y un código que determina el comando. Por ejemplo, si queremos recibir la lectura del ultrasonido 1, enviaremos a través del puerto serie el comando " @80|UltraSonido1\0", tras lo cual, el PIC determinará la acción a realizar, y una vez obtenido el resultado, enviará el valor registrado a través del puerto al PC.

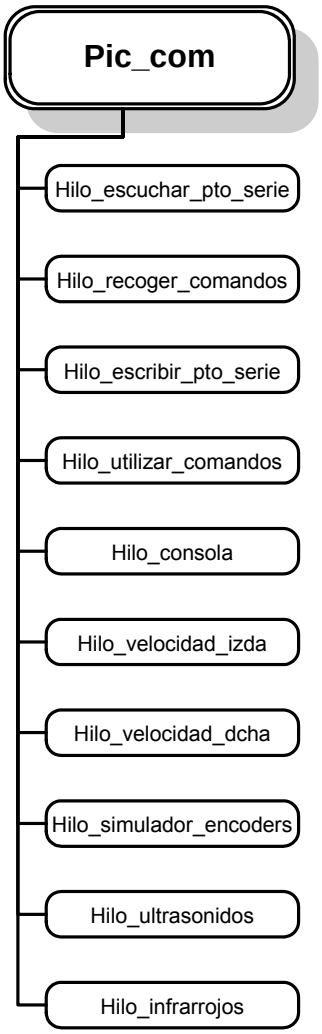
De igual modo, también disponemos de códigos informativos, como por ejemplo, "@99|ACK\0" que indica que el último comando enviado se recibió y ejecutó con éxito, y muchos otros, que determinan que el comando no es reconocido por el PIC al que se envió, comandos de pruebas ("@55|Testing\0") etc.…

Por otro lado este módulo también se encarga de transmitir a los módulos de más alto nivel los datos proporcionados por el PIC, de modo que la comunicación es bidireccional, por ejemplo, si el PIC envía los datos registrados por un ultrasonido, el módulo de comunicación con el PC, sabe que dicho valor ha de reenviarlo al módulo de más alto nivel encargado de controlar los sensores.

Al igual que realiza transmisiones también se encarga de recibir todos los datos proceden del PIC, es decir la información de los encoders, información de los ultrasonidos, infrarrojos y brújula, así como comandos informativos para saber si todavía existe comunicación con el PIC y saber si el comando enviado anteriormente ha sido recibido en este, en esta ocasión se recibiría un comando ACK confirmando la recepción, los comandos recibidos por partes de todos los sensores del robot los hemos denominado comandos con dato, por lo que su tratamiento es diferente al de

los comandos informativos, en el caso de los comandos con dato se hace una división del comando recibido, tomando los 8 primeros caracteres como identificador del comando, mientras que el resto de caracteres serían tratados de manera que serán convertidos a un numero entero para su posterior procesamiento en módulos superiores.

A continuación pasamos a explicar el funcionamiento de los hilos del modulo Pic_com, de manera que queden aclaradas las comunicaciones entre Pic_com y otros módulos:



Hilo_escuchar_pto_serie: Hilo que escucha el puerto serie para controlar el overlapping y almacenar en el buffer los caracteres recibidos.

Hilo_recoger_comandos: Hilo que toma los caracteres del buffer cuando han sido completamente leídos del puerto y los traduce a un comando.

Hilo_escribir_pto_serie: Hilo que escribe en el puerto serie los comandos a enviar al PIC.

Hilo_utilizar_comandos: Hilo que recoge de la cola de comandos recibidos del PIC y los ejecuta enviando los datos a los módulos correspondientes.

Hilo_consola: Hilo que carga una consola interactiva para enviar comandos al PIC

Hilo_velocidad_izda: Hilo que recoge peticiones de cambio de velocidad en la rueda izquierda procedentes del módulo Motores y los guarda en la cola de comandos a enviar al PIC.

Hilo_velocidad_dcha: Hilo que recoge peticiones de cambio de velocidad en la rueda derecha procedentes del módulo Motores y los guarda en la cola de comandos a enviar al PIC.

Hilo_simulador_encoders: Hilo que simula el avance del robot mediante pulsos de encoders lo que permite hacer pruebas de movimiento sin necesidad de tener el robot conectado al PC.

Hilo_ultrasonidos: Hilo que recoge peticiones de barrido de ultrasonidos y los almacena en la cola de comandos a enviar al PIC.

Hilo_infrarrojos: Hilo que recoge peticiones de barrido de infrarrojos y los almacena en la cola de comandos a enviar al PIC.

CAPÍTULO 2: Sistema de tracción

Motores: bajo nivel

Debido a las dimensiones del robot y a su peso, hemos tenido que optar por unos motores de potencia considerable para estar seguros de que el conjunto de la plataforma, baterías, pcbox, y demás añadidos, iban a poder ser trasportados correctamente.

Estos motores son de 90W de potencia pudiendo mover en tracción 20Kg de peso y en régimen de funcionamiento normal, alimentados a 12V, tienen un consumo de unos 0.8A. Si se ofrece resistencia al movimiento del motor, el consumo en intensidad llega hasta los 2-3A.
Por esta razón hemos elegido una controladora de motores de continua (http://www.kelvin.es/formatoshtm/63129.htm) capaz de soportar estos picos de tensión, que además cuenta con un gran condensador para evitar que las baterías de alimentación de los motores, sufran esfuerzos excesivos puntualmente.

Las especificaciones de los motores indican que pueden girar, alimentados a 12v a unas 2000rpm, pero los nuestros tienen una adaptación a una reductora que permite, en primer lugar aumentar el par del motor, y también reducir la velocidad de giro de las ruedas.

Esta controladora es el modelo MD22 (http://www.robot-electronics.co.uk/htm/md22tech.htm) y permite varios modos de funcionamiento. Nosotros hemos elegido el que permite controlar cada motor en función de la diferencia de potencial existente entre el pin de control de cada motor y la tierra de alimentación de la controladora, de modo que tenemos dos pines de control, que se conectan a sendos conversores digital/analógico (DAC), pudiendo así controlar fácilmente desde el PIC el voltaje que queremos aplicar a los terminales de control.

En este modo de funcionamiento, la controladora permite emplear los dos sentidos de giro del motor, para ello establece dos rangos en los terminales de control. De 0v a 2.5v el motor gira en un sentido a una velocidad proporcional a la diferencia de potencial, y de 2.5v a 5v ocurre lo propio pero en el sentido contrario. Además hay una franja de 0.4v justo en la mitad del rango 0v - 5v en la que el motor queda detenido.

Nosotros para no forzar tanto la controladora como los motores, hemos disminuido el rango de voltaje de 0.75v a 4.25v, evitando trabajar así en los límites de sus capacidades.

Los DAC (MAX515) se programan mediante un pin que acepta datos enviados en serie, controlados por una señal de reloj que también hay que proporcionar a la hora de realizar la programación.

Esquema lógico del funcionamiento del controlador de motores

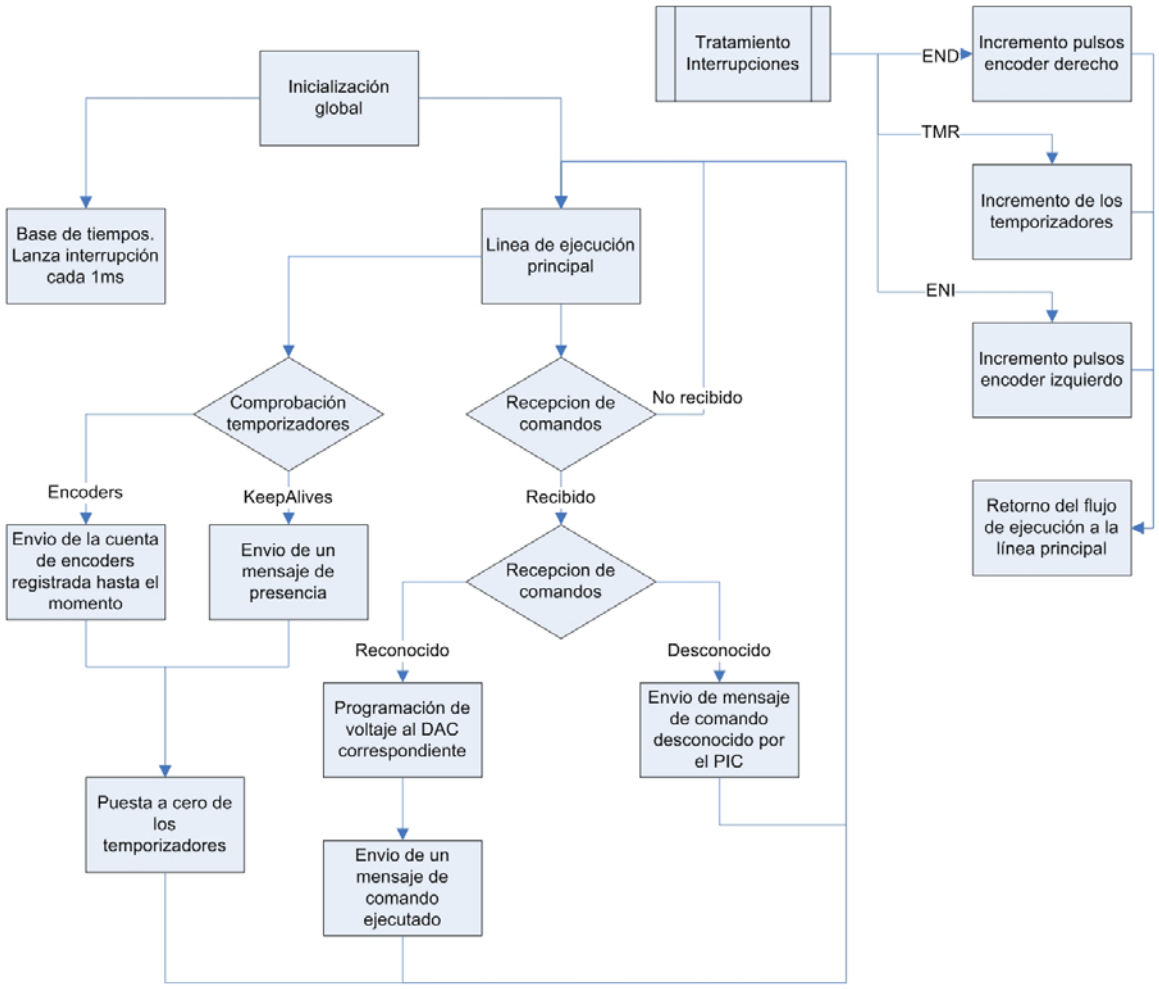


Figura 6 Flujo de ejecución del PIC de control de motores

Bloques del diagrama de flujo

· Inicialización global:

Realiza la configuración de los elementos que se van a utilizar en el PIC, esto es, configuración de los puertos RB7, RB6 y RB5 como salidas digitales; RA3, RA4 y RA5 como salidas digitales; RB2 (INT2) y RB0 (INT0) como entradas digitales, para el control de interrupciones.

También se configura el vector de interrupciones para que redirija el flujo a la rutina de control de interrupciones ante cambios en el estado del puerto INT0 e INT2, que son los puertos empleados para realizar el cuenteo de los encoders.

Además, configura el temporizador TMR1 con unos valores previamente calculados, para que se produzca un desbordamiento cada 1ms, de forma que lo emplearemos para llevar una base de tiempos y cada vez que se produzca una interrupción, incrementar ciertos contadores.

· Línea de ejecución principal:

Se trata de un bucle sin final que se encarga de recibir información del puerto serie de comunicaciones, ejecutar los comandos si es que se ha recibido alguno y comprobar los contadores de temporización, que se emplean para poder llevar control de tiempos mayores de 1ms.

· Recepción de comandos:

Se encarga de comprobar si se ha recibido algo por el puerto serie, y en tal caso, genera una estructura de comando que se envía a otro módulo encargado de su ejecución.
Dicho módulo comprueba si se trata de un comando válido dentro de contexto de ejecución del PIC de los motores y en tal caso lo ejecuta, si no, entonces envía a través del puerto serie un código indicativo de que el comando no se ha reconocido.

Si el comando se ha reconocido, entonces se trata de un comando que desencadena la programación de los conversores digitales/analógicos (DAC) en base a unas tablas previamente definidas con los voltajes necesarios para que la controladora de los motores mueva los motores a una velocidad determinada, existen tres tipos de velocidades lenta, normal y rápida, los comandos recibidos pueden definir también las velocidades marcha atrás, de manera que se enviará en este caso el valor del encoder de de forma negativa, por último lógicamente también existe la velocidad de parada en la que parará el motor correspondiente, cada comando recibido se refiere a un motor en concreto, al igual que se reciben los datos de los encoders de forma alterna.

Una vez programado correctamente el DAC correspondiente, se envía de vuelta por el puerto serie un código de control que indica que el comando se ha ejecutado correctamente (ACK).

En todos los casos (comando reconocido, comando desconocido, no se ha producido envío) el flujo retorna a la línea de ejecución principal.

· Comprobación de temporizadores:

Dado que la base de tiempos marcada por el temporizador TMR1 es de 1ms, las interrupciones generadas son muy rápidas, y es posible que necesitemos tratar datos con un poco más de tiempo, por ejemplo el envío de los mensajes de presencia (KeepAlives).

Es por esto que definimos unas cuentas, de modo que, cada vez que se produce una interrupción por el TMR1, incrementamos el contador. De esta manera, el proceso de comprobación de contadores, se encarga de comprobar si dicho contador ha llegado a un límite previamente definido, y en tal caso, ejecutar una acción asociada.

A modo de ejemplo, el envío de los KeepAlives se realiza cada segundo, de modo que el límite para el contador de los KeepAlives es de 1000 cuentas.
De igual manera se trata la temporización para el envío de los datos de los encoders. Durante el tratamiento de las interrupciones producidas en los puertos de los encoders, se incrementa el numero de pulsos recibidos, y cuando el contador de tiempo de envío de encoders llega a su límite, entonces se envían por el puerto serie las cuentas recibidas hasta el momento y se reinicia el valor de la cuenta de los encoders.

Estos envíos de datos a través del puerto serie hay que hacerlos así para evitar saturarlo y que los datos lleguen corruptos al extremo receptor.

Cuando termina la ejecución asociada a un contador, este se reinicia para volver a iniciar el proceso devolviendo el flujo de ejecución a la línea principal.

Con respecto a los encoders también tendremos unos contadores tanto para llevar la cuenta de los encoders derecho como del izquierdo, estos valores se actualizaran cada milisegundo con una cuenta.

Para el envío de estos datos al PC se mantiene otro contador en el cual se fija la frecuencia de envíos de datos, el envió de los encoders es alterno de manera que cada vez se envía un encoder distinto por ejemplo en primer lugar se envía del encoder derecho y a continuación el izquierdo así continuamente, este control se lleva a cabo mediante una variable que asigna el turno que le corresponda a cada encoder, en este momento el envió de cada encoder esta programado a unos 20 milisegundos de forma que no sobrecargue en demasía al procesador por tantos envíos de cuentas, esta es la velocidad que hemos decidido que es la más adecuada para mantener un control perfecto sobre la posición del robot tras varias pruebas, ya que si alargábamos un poco más el envío de estas cuentas el robot podía hacer algún giro que no era el más adecuado ya que podía no seguir una línea totalmente recta sino que debido al numero de encoders avanzados podía llegar en algún momento a realizar algún recorrido en eses.

Control de velocidad

Un primer problema al que nos enfrentamos cuando empezamos a hacer pruebas reales con los motores, fue que, a pesar de que ambos estaban programados para recibir la misma tensión, y por tanto, trabajar a la misma velocidad, uno de ellos siempre giraba un poco más rápido que el otro.

Esto sin duda conducía a que el robot nunca andase en línea recta, sino que cada vez se iba escorando hacia un lado, provocando en primer lugar una posible colisión, y en segundo, una falsa orientación.

Por ello, lo que hicimos fue cambiar el concepto de las velocidades de los motores e incluir un *control de velocidad proporcional*, de forma que lo que hacemos es definir una velocidad relativa en pulsos de encoder, esto es, por ejemplo, velocidad uno equivale a 3000 pulsos de encoder, de forma que, cuando aplicamos la tensión al motor para ponerlo en velocidad uno, comprobamos qué pulsos estamos recibiendo desde el encoder.

Si este numero de pulsos es menor que el esperado, entonces hacemos un cálculo que consiste en aumentar el voltaje de forma que hacemos una media entre el valor actual y el requerido, aplicando el voltaje del resultado al motor. Esto realizado cíclicamente permite que la velocidad del motor se vaya adaptando a la velocidad requerida poco a poco (aunque realmente nunca llegue a la velocidad requerida ya que el sistema de cálculo por medias provoca la aparición de una asíntota en el estado requerido).

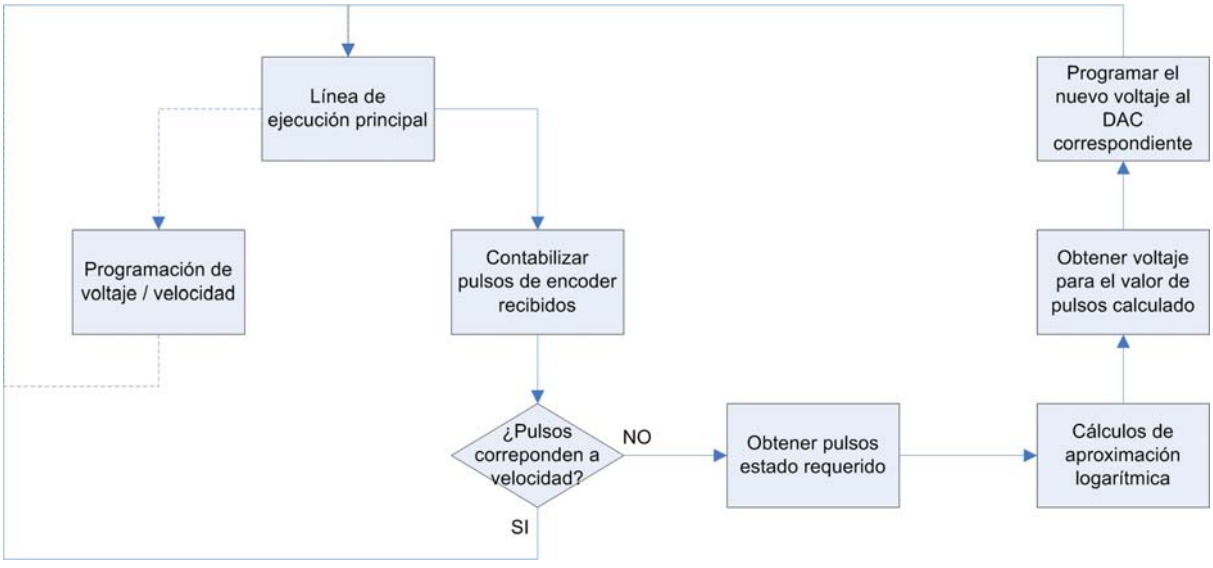


Figura 7 Diagrama de flujo para el control proporcional de velocidad

Una funcionalidad adicional que nos brinda este control de velocidad, es la posibilidad de detectar fallos o estado de carga de las baterías de alimentación de los motores, ya que, si pasado un tiempo prudencial de estabilización de la

velocidad, los pulsos recibidos por los encoders no se aproximan a los requeridos para la velocidad programada, es un síntoma inequívoco de que las baterías no están en su estado óptimo de carga y se puede enviar una alarma a los módulos de más alto nivel para que notifiquen la situación.

Calculo de las velocidades

El rango de valores que tenemos para mover nuestros motores esta entre -5.29 y 5.27, estas velocidades tenemos que traducirlas a los valores correspondientes en la controladora que tan solo tiene un rango entre 0 y 4.20 voltios para conseguir nuestra velocidad adecuada. El rango de velocidades de la controladora es el siguiente:

- Marcha atrás entre 0 y 2 voltios
- Motores parados entre 2 y 2.2 voltios
- Marcha adelante entre 2.2 y 4.20 voltios

Debido a que no podíamos introducir directamente los valores analógicos y la controladora solo admitía para su configuración valores analógicos tuvimos que introducir un par de conversores para realizar la transformación de las velocidades del potencial de entrada a el potencial que le llega a la controladora para ello recurrimos a la aplicación de diversas transformaciones;

1. La primera debido a que el rango tanto para la marcha atrás como para la marcha adelante en la controladora era de 2 voltios y el potencial real que llegaba era de 5.25 voltios realizamos la siguiente regla de 3:

   2 ---------- 5.27
   Vres ---------- Velegida

   Siendo Velegida un valor entre 0 y 5.27 y Vres nos daría de resultado un valor entre 0 y 2.

   Siendo Vres el valor equivalente a nuestro valor en el rango de potencial de la controladora, a este valor ahora habría que sumarle 2.2 voltios en caso de que necesitemos una velocidad de marcha adelante.

   En caso de requerir una marcha hacia atrás tendríamos que restar el valor resultado a 2 para obtener nuestro potencial, ya que en el caso de la marcha atrás de la controladora, el máximo valor que tendría esta sería el cero y el valor al que los motores irían mas despacio seria los mas cercanos al 2 por debajo de este, en cambio para las marchas hacia delante el potencial al que irían mas rápido sería el de 4.2 voltios y al mas lento los valores superiores cercanos al 2.2.

Por lo que tendríamos que el Potencial entrada a la controladora sería Velegida mientras que el que saldría de la controladora a los motores sería Vcontroladora que realizando los cálculos correspondientes quedaría así:

- Marcha adelante Vcontroladora = 2.2 + Vres
- Marcha atrás Vcontroladora = 2 - Vres

3. La siguiente transformación sería el paso de los datos del voltaje a datos digitales, en este caso el rango que tenemos para definir la velocidad a la que se mueven los motores estaría entre 0 y 512 (bits), la regla aplicada en este caso es la siguiente:

   512 ---------- 4.2
   Bits ---------- Vcontroladora

   Para finalizar las transformaciones el dato obtenido en bits debemos pasarlo a un array de 10 bits que es l que acepta la controladora para definir dichas velocidades.

A continuación mostramos un ejemplo para dada una velocidad entre 0 y 5.25 voltios (en este caso una velocidad positiva), calculemos el valor al que hay que programar la controladora para obtener esta velocidad, en el caso de las velocidades definidazas en el proyecto se tomaron tras varias pruebas y de forma que la velocidad fuese adecuada para caso.

Velocidad requerida en el motor izquierdo de 5 voltios (casi el máximo posible), pasaríamos a calcular utilizando la primera regla de 3 la velocidad adecuada a nuestro rango (0 -2)

2	----------	5.27	2	----------	5.27
Vres	----------	Velegida	Vres	----------	5

Vres = 1.89 Voltios

A ser una velocidad positiva pasamos a sumarle el rango correspondiente en este caso de 2.2 Voltios de forma que el voltaje correspondiente sería de 4.19 voltios.

El último paso sería el paso a bits de forma que aplicando la segunda regla de 3 explicada anteriormente obtendremos el valor entero que habría que traducir a nuestro array de 10 bits para programar el motor a un voltaje de 5 voltios.

Dificultades encontradas

En un primer momento, antes de decidirnos a utilizar la controladora de los motores, planteamos un diseño basado en el IC L293 y puentes de diodos en 'H' para controlar cada motor mediante un pulso de anchura variable (Pulse Width Modulation).

Las especificaciones de estos chips rezaban que eran capaces de distribuir hasta 2A de potencia cada uno y según nuestras pruebas, salvo picos, los motores se mantenían constantes a 0.8A. Sin embargo, al emplear este sistema de control los LM293 se calentaban demasiado, pudiendo llegar a estropearse en caso de largos tiempos de actividad.

Después de varios intentos, modificando las frecuencias de trabajo del PWM e incorporando más chips para repartir más aún el flujo de corriente, llegamos a la conclusión de que no iba a ser un método de control viable, ya que el PWM la función que tenía era la de permitir el paso de corriente continua a través de los 293 cuando el pulso se encontraba en alta y cortar el paso de corriente al encontrarse en baja, y eran estos "arranques y paradas" del flujo de corriente los que provocaban el calentamiento de los chips

Esquema físico para la conexión a los motores

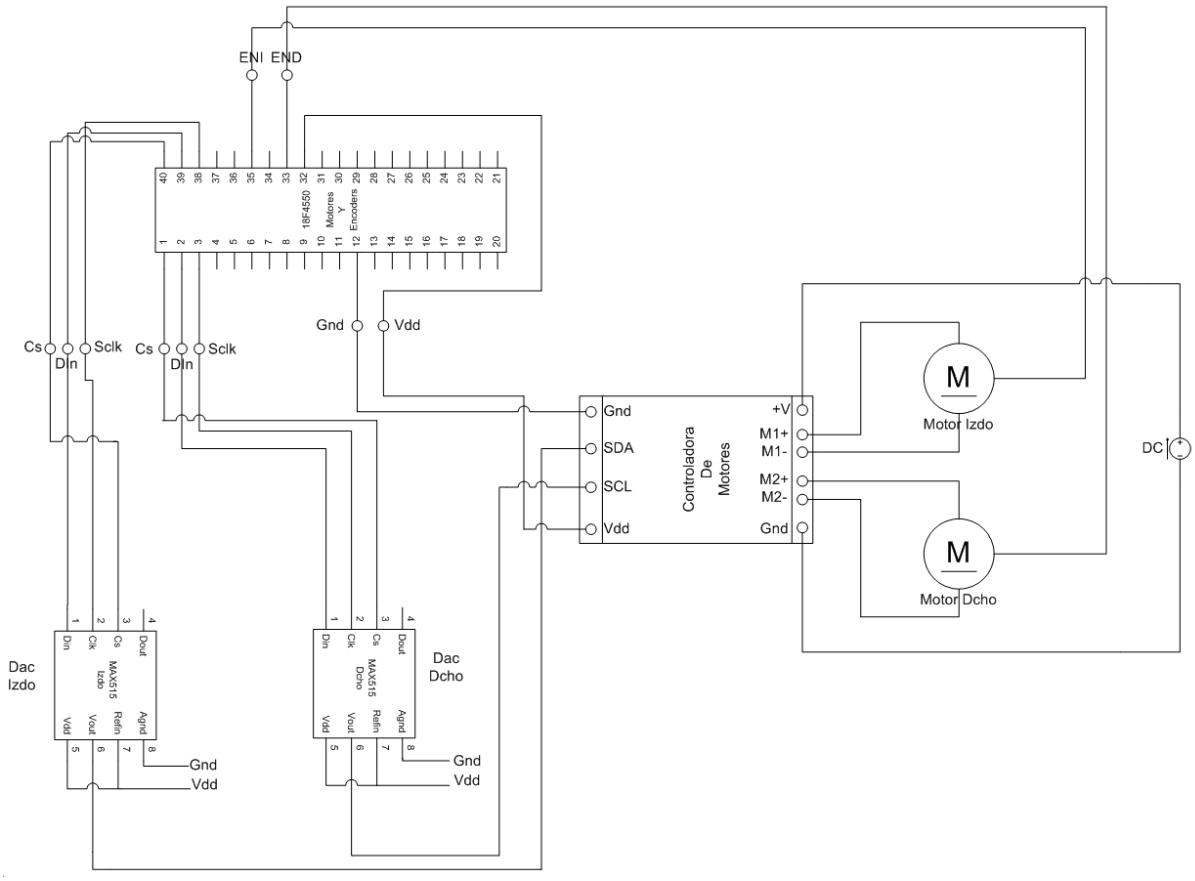


Figura 8 Diagrama de conexiones para los motores

Las especificaciones de los DAC indican que, para trabajar en el rango de conversión de 0v a 4.096v hay que colocar unas resistencias y condensadores como electrónica de acondicionamiento para los chips, pero tras hacer varias pruebas, comprobamos que el funcionamiento tras agregarlas no era el adecuado, por lo que investigamos posibles estados de funcionamiento y al final concluimos que lo que más se adecuaba a nuestras necesidades era la operación de los chips sin ninguna electrónica colindante.

La placa controladora de los motores (MD22 de Devantech) tiene unos interruptores de configuración de modo de trabajo, en nuestro caso, empleamos la configuración (ON ON ON OFF) que es la que permite controlar la velocidad de los motores mediante la variación de la tensión en el puerto SDA para el motor 1 y en el puerto SCL para el motor 2.

Aunque no aparece reflejado en el diagrama, es más que conveniente la inclusión de un fusible de algo menos de 10A para proteger la controladora en caso de un pico, este fusible se acoplaría entre el borne positivo de las baterías de alimentación y el puerto V+ de la controladora.

Motores: alto nivel

La parte de alto nivel del control de motores se compone de los siguientes módulos:

- **Motores:** este módulo se encarga de escoger la velocidad adecuada para cada rueda en función de la velocidad y el giro requeridos por del módulo Multiplexor. Las velocidades hacia delante van de 1 a 3 y hacia atrás de -1 a -3, siendo 0 parado; y los giros de -1 a -4 son para la izquierda y de 1 a 4 para la derecha, siendo 0 recto.

Vel/giro	-4	-3	-2	-1	0	1	2	3	4
3	(-3,3)	(0,3)	(1,3)	(2,3)	(3,3)	(3,2)	(3,1)	(3,0)	(3,-3)
2	(-2,2)	(0,2)	(0,2)	(1,2)	(2,2)	(2,1)	(2,0)	(2,0)	(2,-2)
1	(-1,1)	(0,1)	(0,1)	(0,1)	(1,1)	(1,0)	(1,0)	(1,0)	(1,-1)
0	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)
-1	(1,-1)	(0,-1)	(0,-1)	(0,-1)	(-1,-1)	(-1,0)	(-1,0)	(-1,0)	(-1,1)
-2	(2,-2)	(0,-2)	(0,-2)	(-1,-2)	(-2,-2)	(-2,-1)	(-2,0)	(-2,0)	(-2,2)
-3	(3,-3)	(0,-3)	(-1,-3)	(-2,-3)	(-3,-3)	(-3,-2)	(-3,-1)	(-3,0)	(-3,3)

Figura 9 Marchas de las ruedas izquierda y derecha en función de las variables "Velocidad" y "Giro"

- **Pic_com:** este módulo se encarga de servir de mensajero entre el modulo de Motores y el PIC, en este caso recibimos individualmente las velocidades de cada rueda y se traduce al código correspondiente que el PIC reconoce.

  Los códigos que el PIC reconoce son:

Velocidad de la rueda	Código PIC
Parado	@90|Vel0\0
Adelante 1	@91|Vel0\0
Adelante 2	@92|Vel0\0
Adelante 3	@93|Vel0\0
Atrás 1	@94|Vel0\0
Atrás 2	@95|Vel0\0
Atrás 3	@96|Vel0\0

Figura 10 Códigos de velocidades para la rueda derecha que el PIC reconoce

  Esta tabla es la correspondiente a la rueda derecha. Los códigos para la rueda izquierda tienen el mismo formato pero con números a partir del 80 en lugar de a partir del 90.

Las comunicaciones entre los módulos son las siguientes:

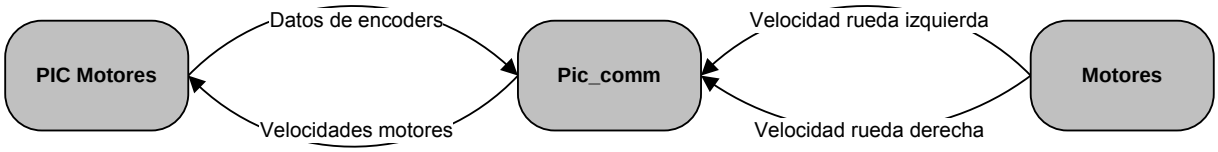
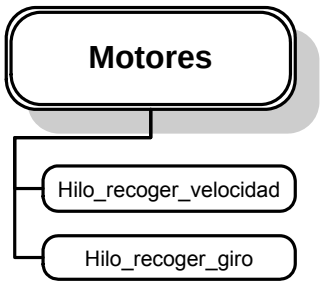


Figura 11 Comunicación entre Motores y otros módulos

Y éstos son los hilos que ejecuta el modulo Motores:



Hilo_recoger_velocidad: Hilo que recoge peticiones de cambio de velocidad procedentes de multiplexor y, en función de ésta y del giro requerido envía una velocidad para cada rueda a pic_com.

Hilo_recoger_giro: Hilo que recoge peticiones de cambio de giro procedentes de multiplexor y, en función de éste y de la velocidad requerida envía una velocidad para cada rueda a pic_com.

Multiplexor

Este módulo se encarga de elegir que módulo controla los motores. El diagrama de interconexión con otros módulos es el siguiente:

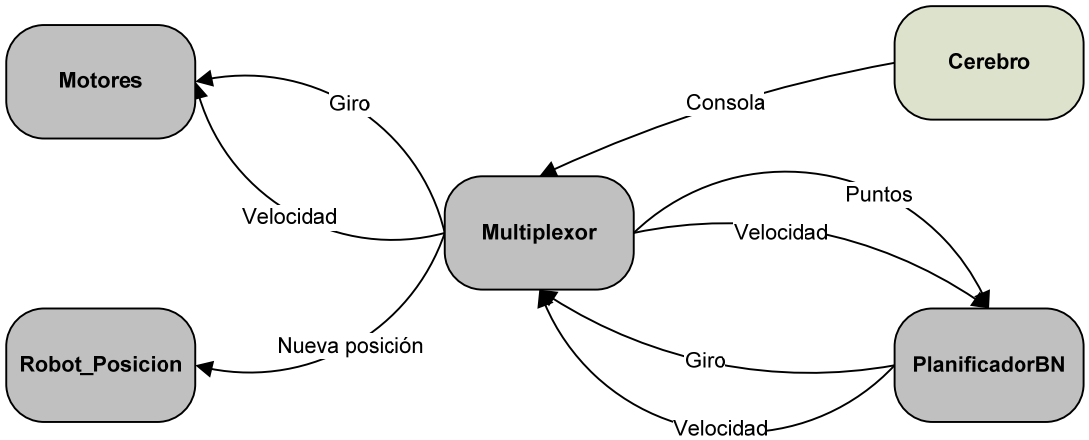


Figura 12 Comunicación entre Multiplexor y otros módulos

Así, este módulo permite a la consola de direcciones que implementa Cerebro (parte de alto nivel) controlar manualmente los motores del robot. Para ello traduce las direcciones provenientes de la consola a velocidades y giros interpretables por el módulo Motores. La tabla que muestra esta traducción es la siguiente:

Vel/giro	-4	-3	-2	-1	0	1	2	3	4
3				NO		NE			
2	O				N				E
1									
0					Q				
-1									
-2					S				
-3				SO		SE			

Figura 13 Velocidades y giros que se corresponden con las direcciones de la consola de Cerebro

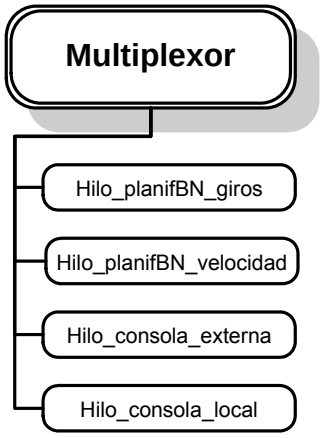
NOTA: {N, S, E, O} son las direcciones {norte, sur, este, oeste}

Este módulo también redirige las velocidades y giros calculados por el planificador de bajo nivel (PlanificadorBN) hacia el módulo Motores. Así, permite alternar entre modo manual (consola) y modo automático (PlanificadorBN) en el manejo de motores.

También implementa una consola local muy completa que permite desde el control manual de los motores hasta el envío de trayectorias por teclado al planificador de bajo nivel, actualización de la posición y orientación del robot en el módulo Robot_posicion, etc.

Si en un futuro quiere implementarse el manejo del robot mediante un nuevo controlador, ya sea un joystick, algún otro tipo de consola remota, etc, deberá interconexionarse con este módulo.

Los hilos que ejecuta el módulo Multiplexor son los siguientes:



Hilo_planifBN_giros: Hilo que recoge peticiones de giro procedentes del planificador de bajo nivel (planificadorBN) y las redirige al módulo de motores.

Hilo_planifBN_velocidad: Hilo que recoge peticiones de cambio de velocidad procedentes del módulo planificadorBN y las redirige al módulo de motores.

Hilo_consola_externa: Hilo que recoge direcciones de la consola de alto nivel y las traduce a giros y velocidades que envía al módulo de motores.

Hilo_consola_local: Hilo que ejecuta una consola que permite probar el correcto funcionamiento de los módulos de bajo nivel: desde el control total de los motores hasta el paso de trayectorias al planificador de bajo nivel, pasando por actualizaciones de posición, etc.

Planificador de bajo nivel

Este módulo recoge los puntos que le pasa el planificador local (parte de alto nivel) y planifica los giros y velocidades necesarias para alcanzar dichos puntos. La interconexión con otros módulos es la siguiente:

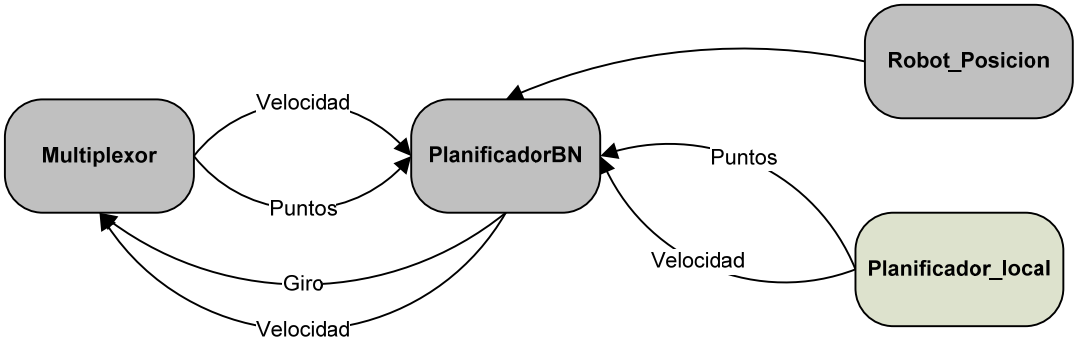


Figura 14 Comunicación entre PlanificadorBN y otros módulos

Para realizar la planificación esté módulo ha de conocer la posición y dirección del robot en todo momento. Para ello está comunicado con el módulo Robot_posicion, del que recibe dichos datos.

El planificador local le va pasando los próximos puntos de la trayectoria que debe seguir como posiciones absolutas dentro de un mapa, así como la velocidad máxima permitida para los motores. El PlanificadorBN va almacenando esos puntos en una cola y planificando las velocidades y giros necesarios para trazar una trayectoria óptima que pase por todos esos puntos.

Para planificar los giros y velocidades se basa en la diferencia de ángulos entre la dirección actual del robot y la dirección del próximo punto en la trayectoria. Así, según sea este ángulo mayor o menor realiza un giro u otro.

Cuando se encuentra a una distancia considerable del próximo punto el planificador prefiere girar sobre si mismo e ir de frente hacia el objetivo a usar la marcha atrás. Esto es preferible porque los sensores de ultrasonidos e infrarrojos se encuentran mayormente en la parte delantera del robot. Suponiendo que la dirección del robot es hacia arriba (↑), la tabla siguiente muestra los giros elegidos por el planificador en función de en que franja angular se encuentre la dirección del objetivo (planificación para puntos alejados del robot):

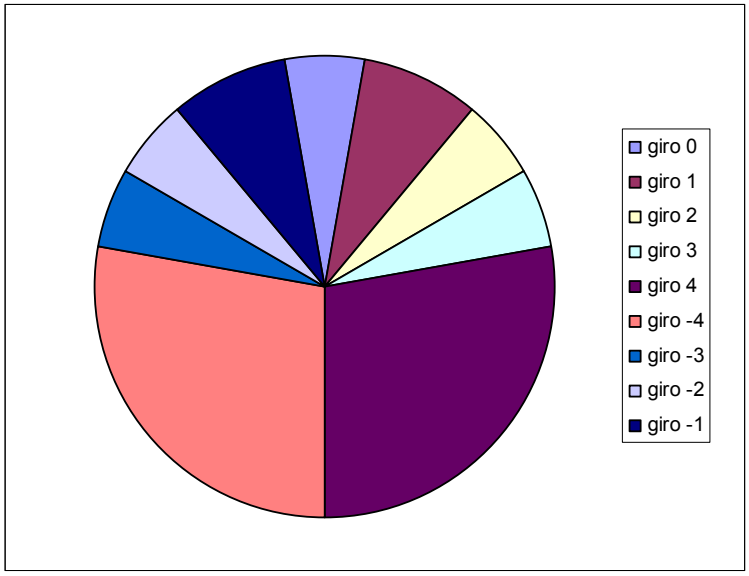


Figura 15 Giros para la planificación a larga distancia

NOTA: recuérdese que los giros negativos son hacia la izquierda y los positivos hacia la derecha, cuanto mayores mas cerrados, alcanzado el máximo en el giro 4 (giro sobre si mismo).

Para maniobrar en distancias cortas (menores que el radio del robot) y alcanzar con precisión su objetivo, el planificador tiene en cuenta otros ángulos y hace uso de la marcha atrás para mayor maniobrabilidad. Suponiendo que la dirección del robot es hacia arriba (↑), la tabla siguiente muestra los giros y dirección elegidos por el planificador en función de en que franja angular se encuentre la dirección del objetivo (planificación para puntos cercanos):

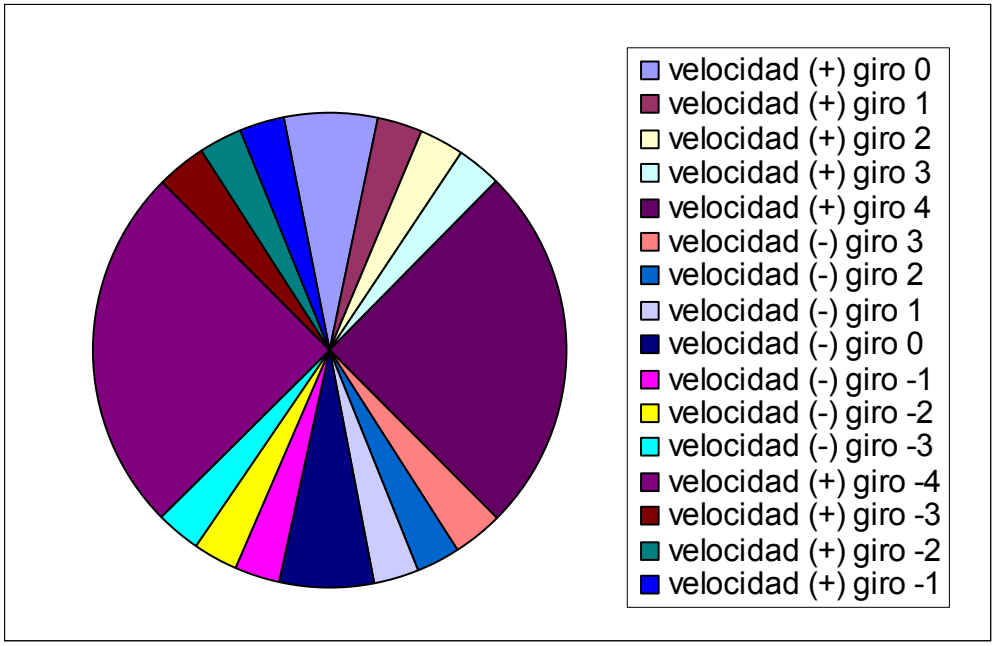


Figura 16 Giros y velocidades para la planificación a corta distancia

El planificador hace distinción entre puntos objetivo y puntos intermedios de una trayectoria. La diferencia está en que en los puntos objetivo hace una frenada gradual al acerarse al objetivo y además tiene en cuenta la orientación final del robot, que se le indica cuando se le pasa el objetivo.

Los puntos intermedios los va desechando cuando pasa cerca de ellos para así pasar a replanificar en función del siguiente punto en la trayectoria. Así se consigue trazar una trayectoria más suave y sin paradas intermedias.

Todos los parámetros de ángulos, precisión para objetivos y puntos intermedios, etc, son configurables mediante XML. Con lo cual es muy fácil variar el comportamiento de este planificador pues no hace falta tocar el código y mucho menos recompilar el módulo.

Queda reseñar que con este planificador se consiguen alcanzar los puntos objetivo con total precisión, lo que en un futuro permitirá al robot, por ejemplo, alcanzar el enchufe donde tenga que conectarse para recargar las baterías de manera automática cuando se quede sin carga, siempre ayudado de balizas, claro.

Los hilos que ejecuta el módulo PlanificadorBN son los siguientes:

Hilo_recoger_posiciones: Hilo que va recogiendo las actualizaciones de la posición del robot procedentes de robot_posicion y replanificando la velocidad de las ruedas en función de la posición y la dirección del próximo punto a seguir.

Hilo_recoger_puntos: Hilo que va recogiendo los puntos a seguir procedentes del planificador local y replanificando la velocidad de las ruedas en función de la posición y la dirección del próximo punto a seguir.

Hilo_recoger_velocidades: Hilo que recoge la máxima velocidad permitida para las ruedas.

CAPÍTULO 3: Sensores de posición

Información sobre los sensores

Un **sensor** es un dispositivo que detecta, o *sensa* manifestaciones de cualidades o fenómenos físicos, como la energía, velocidad, aceleración, tamaño, cantidad, etc.

Muchos de los sensores son eléctricos o electrónicos, aunque existen otros tipos. Un sensor es un tipo de transductor que transforma la magnitud que se quiere medir, en otra, que facilita su medida. Pueden ser de indicación directa (e.g. un termómetro de mercurio) o pueden estar conectados a un indicador (posiblemente a través de un convertidor analógico a digital, un computador y un display) de modo que los valores sensados puedan ser leídos por un humano.

A continuación se indican algunos tipos y ejemplos de sensores electrónicos:

- Sensores de temperatura: Termómetro
- Sensores de deformación: Galga extensiométrica
- Sensores ópticos: infrarrojos
- Sensores de sonido: micrófono
- Sensores de contacto: Interruptores

Por lo general la señal de salida de estos sensores no es apta para su procesamiento, por lo que se usa un circuito de acondicionamiento.

Los sensores se dividen en dos clases diferenciada los sensores internos y los sensores externos.

Sensores internos

Los sensores internos son aquellos que están dedicados a medir variables propias del robot o que puedan afectar a su funcionamiento como la temperatura, la presión,etc… son los que permiten que el robot tenga conocimiento de su propio estado, gracias a ellos es posible que el robot pueda realizar las tareas que tenga asignadas con precisión y velocidad. Se suelen ocupar de dos tipos de medidas:

– estado de la estructura mecánica del robot:
  - posición,
  - velocidad, y
  - torsión de cada una de las articulaciones

– estado del medio donde se mueve el robot :
  - temperatura,
  - presión, ...

Dentro de este tipo de sensores se encuentran los sensores de posición entre los que se encuentran los sensores incrementales (encoders) y absolutos (brújulas) A continuación pasaremos a explicar el funcionamiento de estos diversos tipos de sensores

Sensores de posición

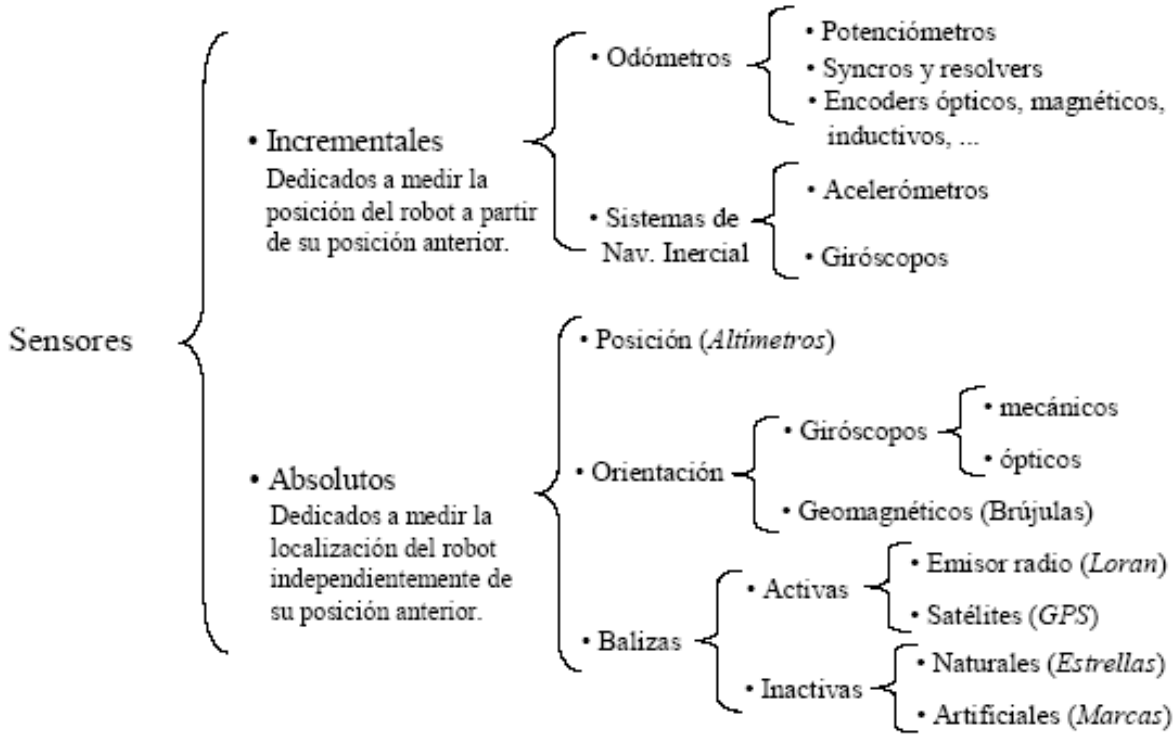


Figura 17 Tipos de sensores

Dentro del campo de los sensores incrementales profundizaremos en el funcionamiento de los encoders, dado que son los que usamos en nuestro robot.

Los sensores incrementales son un tipo de sensores que nos permiten conocer la posición en la que se encuentra nuestro robot o artefacto, a partir de la posición anterior de este, en la actualidad la mayoría de los sistemas de navegación se basan en este sistema, debido a la acumulación de cierto error acumulativo en el calculo de la posición con estos métodos se suelen utilizar sistemas externos para corregir dicho error y posicionar correctamente el objeto tipo de sensores, por ejemplo mediante el uso de brújulas y sistemas de triangulación o balizas.

A continuación pasaremos a explicar el funcionamiento de los sensores de posición que utilizamos en nuestro proyecto entre los que se encuentran los encoders, brújulas.

Encoders

Principio de funcionamiento de los encoders incrementales

Los codificadores ópticos o encoders incrementales se utilizan fundamentalmente para el cálculo de la posición angular. Básicamente constan de un disco transparente, el cual tiene una serie de marcas opacas colocadas radialmente y equidistantes entre si; de un elemento emisor de luz (como un diodo LED); y de un elemento fotosensible que actúa como receptor. El eje cuya posición angular se va a medir va acoplado al disco.

El funcionamiento es el siguiente: cuando el sistema comienza a funcionar el emisor de luz empieza a emitir; a medida que el eje vaya girando, se producirán una serie de pulsos de luz en el receptor, correspondientes a la luz que atraviesa los huecos entre las marcas. Llevando una cuenta de esos pulsos es posible conocer la posición del eje.
Sobre este esquema básico es habitual encontrar algunas mejoras. Por ejemplo, se suele introducir otra franja de marcas por debajo, desplazada de la anterior, para poder controlar el sentido del giro; además suele ser necesario el empleo de una marca de referencia que nos ayudará a saber si hemos completado una vuelta.

Realmente los encoders incrementales miden la velocidad de giro, pero podemos extrapolar la posición angular. Como es lógico, la resolución de este tipo de sensores depende directamente del número de marcas que podamos poner físicamente en el disco.

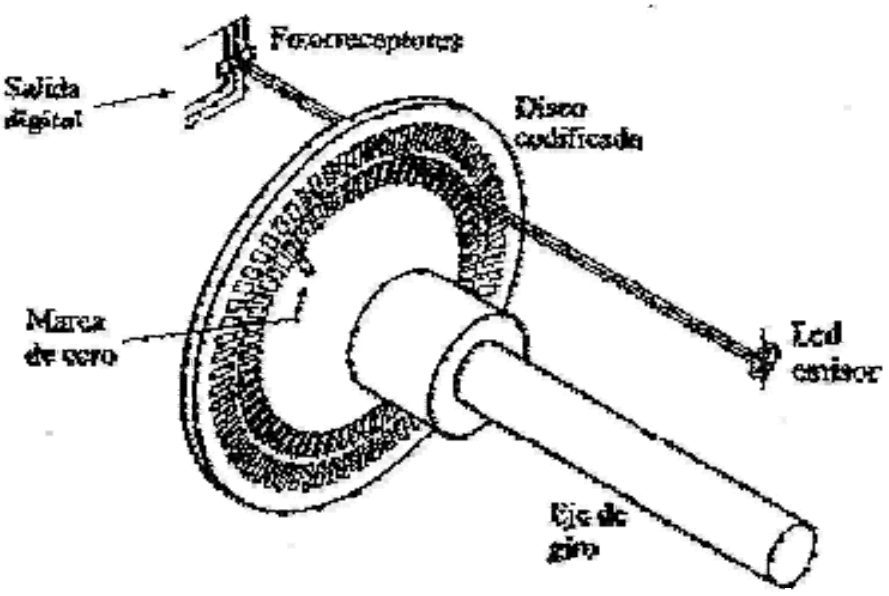


Figura 18 Encoder óptico incremental

Para realizar el cálculo de la distancia recorrida por los encoders nos basaremos en lo siguiente:

Los datos de los cuales partimos inicialmente son los radios de las ruedas(R), la distancia que hay entre estas (d) y los encoders recorridos por cada rueda del robot en un cierto intervalo, a partir de estos se puede calcular el ángulo de giro y las distancias recorridas.

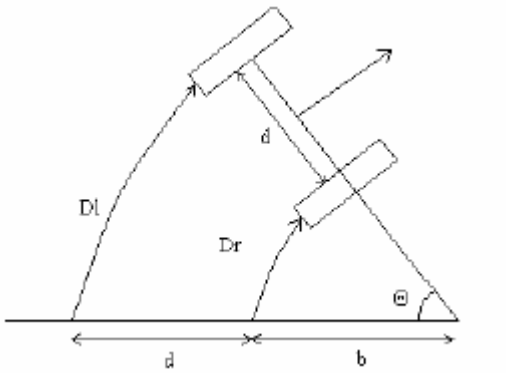


Figura 19 Ángulos para el cálculo del avance de los encoders

$$D_l = (d+b)\cdot\theta = \frac{2\pi R_l}{N_{Tl}} n_l$$

$$D_r = b\cdot\theta = \frac{2\pi R_r}{N_{Tr}} n_r$$

$$\Theta = \frac{D_l - D_r}{d} = \frac{2\pi}{d}\left[R_l \frac{n_l}{N_{Tl}} - R_r \frac{n_r}{N_{Tr}} \right]$$

Figura 20 Fórmulas para el cálculo del avance de los encoders

D_l sería la distancia recorrida por la rueda izquierda,
D_r es la distancia de la rueda derecha,
d es la distancia entre las ruedas,
b es la distancia que existe entre la rueda que avance menos, es decir en la dirección en la cuál realicemos el giro, hasta el centro de la circunferencia que sería concéntrica al giro producido por las dos ruedas,
Θ ángulo de giro
n_r es el número de encoders recorridos por la rueda derecha
n_l es el número de encoders recorridos por la rueda izquierda

A partir de estas expresiones se puede comprobar que si la rueda izquierda gira más rápido que la derecha el robot girará a la derecha, mientras que si la derecha gira más rápido que la izquierda el robot girará a la izquierda. Además, si las ruedas giran en sentidos opuestos a la misma velocidad, el robot no se mueve pero gira sobre sí mismo: si I gira hacia delante y D hacia atrás, el robot gira a la derecha, etc…

Utilización de los encoders

Los encoders los utilizamos para posicionar el robot, para ello nos basamos en la formulas mostradas anteriormente y en la posición anterior que posee el robot.

Aparte de esto cada determinado tiempo se realizara una comparación entre la orientación que debería llevar el robot en este preciso instante con la proporcionada por una brújula digital, para así corregir la orientación de este debido a alguna perdida de encoders durante el movimiento.

En un principio el cálculo de la posición del robot se realizo mediante el uso conjunto de la brújula digital y los encoders de cada rueda, pero se decidió cambiarlo para basarse tan sólo en los datos de los encoders momentáneamente debido a ciertos problemas con el PIC correspondiente a los sensores, para poder así realizar las pruebas necesarias e ir probando el resto del sistema, en la actualidad el sistema utilizado es de nuevo el uso conjunto de la brújula y los encoders.

De esta manera robot esta programado de tal forma que para ir en línea recta se solicite la orientación de la brújula para situar correctamente al robot, en el caso de los giros del robot se procederá a calcular la orientación mediante el único uso de los encoders debido a que la brújula suele sufrir variaciones y suele tardar un tiempo prudencial en estabilizarse, el cálculo realizado por los encoders de todas formas es bastante fiable ya que no se suelen perder cuentas de los mismos.

Esquema lógico del funcionamiento de la detección de pulsos con encoders

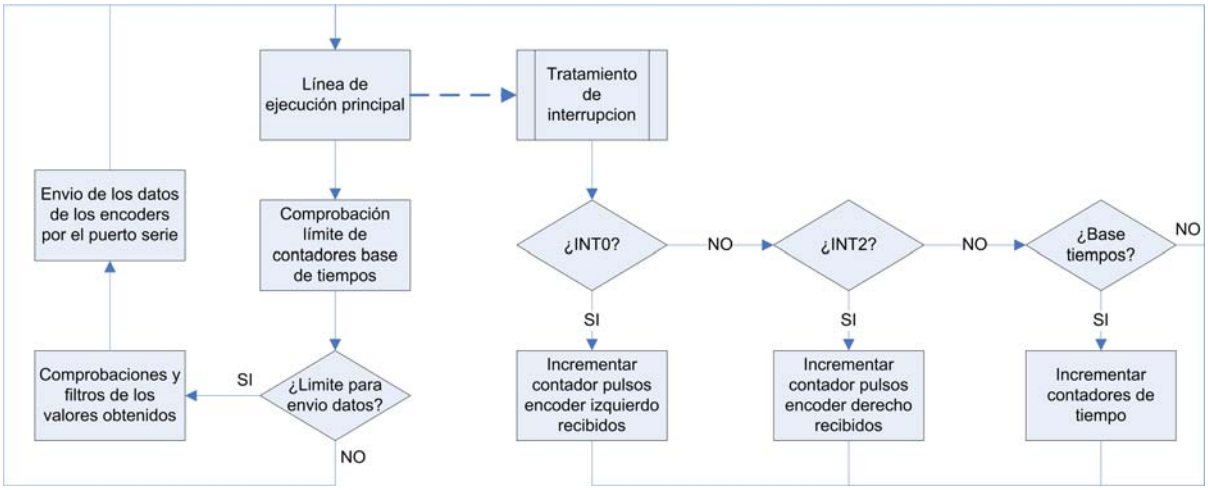


Figura 21 Diagrama de flujo para el posicionamiento con encoders

Bloques del diagrama de flujo

· Interrupciones en INT0 e INT2:

Cada vez que se produce una interrupción en alguno de estos dos puertos, se incrementa el contador de pulsos de encoder asociado al mismo. Cada puerto corresponde a un encoder.
Las interrupciones se lanzan cuando se presenta un flanco de subida en la tensión aplicada al puerto, y eso significa que el sensor ha detectado un giro en el motor.

Para evitar una sobrecarga o un error en el envío de datos del PIC al PC el valor al cuál se envía cada encoder es de 20 milisegundos mientras que por arriba no existe un limite, pero debido a que es necesario tener continuamente los datos de los encoders actualizados y cuanto mas rápido se reciban mejor llegamos a la conclusión que este valor es el adecuado para realizar los cálculos de las posiciones.

· Comprobación de contadores:

Este proceso se encarga de comprobar si ha pasado el tiempo requerido para el envío de los datos de los encoders, ya que para que los datos recogidos por estos sean fiables, hay que recoger más de un pulso.
Además, si enviásemos cada vez que el encoder detecta un pulso, saturaríamos el puerto, pues estos sensores tienen una resolución de 500 pulsos / vuelta de eje.
Por tanto, definimos un tiempo límite, durante el cual se van contabilizando los pulsos emitidos por los encoders, y una vez este tiempo ha concluido, se envían a través del puerto serie después de hacer comprobaciones y filtros con el valor obtenido.

Cálculo del avance del robot usando los encoders

Los módulos que intervienen en el cálculo de la posición del robot son Pic_com, Encoders, Brújula y Robot_posicion. Su interconexión viene detallada en el diagrama a continuación:

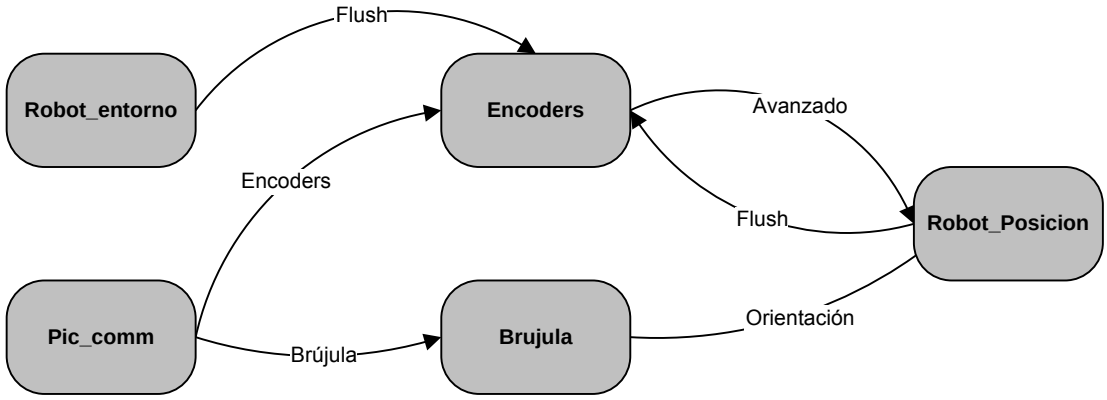


Figura 22 Módulos que intervienen en el cálculo de la posición y su interconexión

En Encoders se acumulan y transforman a centímetros los pulsos de encoger provenientes Pic_com hasta llegar a un cierto umbral, indicado por XML, momento en el cual se envía el avance acumulado al módulo Robot_posicion. Esto se hace así dado que sin acumular los datos la distancia avanzada por el robot en cada paso seria demasiado pequeña e inapreciable.

El módulo Encoders dispone también de una opción de envío de los datos acumulados sin necesidad de llegar al umbral. Esto se hace así cuando Robot_entorno requiere la posición actualizada en ese preciso momento para situar un obstáculo que ha detectado. Para hacer esto envía una señal de flush al módulo Encoders.

Para realizar el posicionamiento del robot se utiliza Robot_Posicion, módulo que se encarga de recibir el avance de las ruedas y la orientación de la brújula. Para realizar estos cálculos necesitamos conocer la posición anterior del robot para, a partir de ésta, incrementar los valores de posición. Para guardar la posición se usa la posición del centro del robot. Y para el cálculo de la posición se utilizan las fórmulas vistas anteriormente en los encoders.

El algoritmo calcula la nueva posición del robot a partir de los datos proporcionados por la brújula (orientación actual) y los encoders (avance de las ruedas y, por tanto, la longitud de los arcos de las correspondientes circunferencias).

Para realizar los cálculos se basa en la aproximación de que las trayectorias de las ruedas son circunferencias concéntricas para, a partir de éstas, calcular el centro de las circunferencias y, basándonos en cálculos de senos y cosenos, realizar la operación adecuada para calcular los incrementos.

Se distinguen varios casos en el cálculo incremental de la posición:

- En el primero tan solo avanza una rueda. En dicho caso el centro de la circunferencia estaría situado en la posición correspondiente a la rueda sobre la que se produce el giro.

- En otro caso el robot avanza en una supuesta línea recta. En este caso actualizamos la orientación con el dato procedente de la brújula. Este es el momento adecuado para realizarlo ya que la brújula se encuentra estabilizada. En el resto de casos el ángulo en que nos basamos es el calculado mediante el avance de las ruedas debido a que en mitad de un giro la brújula no se encuentra estabilizada.

- El último caso es aquel en el cual las ruedas avanzan distancias distintas. En este caso tendríamos dos supuestas circunferencias concéntricas de las cuales no conocemos el centro y por tanto debemos averiguarlo.

Brújula

Las brújulas digitales del tipo que estamos usando(CMPS03) son sensores de campos magnéticos que una vez calibrada ofrece una precisión de 3-4 grados y una resolución de décimas. Tiene dos interfaces, mediante pulsos temporizados (modulación en anchura), o bien por medio de un bus I2C, lo que facilita su comunicación con una amplia gama de microcontroladores, incluyendo los Basic Stamp, Basic X, OOPIC y otros lenguajes compilados. Este sensor magnético esta específicamente diseñado como sistema de navegación para robots. La brújula esta basada en los sensores KMZ51 de Philips que son lo suficientemente sensibles como para captar el campo magnético de la tierra. Usando dos de estos sensores colocados en ángulo de 90 grados, permite al microprocesador calcular la dirección de la componente horizontal del campo magnético natural

Para realizar la calibración de la brújula deberemos antes de nada realizar los siguientes pasos, el modulo deberá mantenerse perfectamente horizontal con los componentes hacia arriba y los dos sensores en la cara inferior. Mantener el modulo alejado de objetos metálicos y muy especialmente de objetos magnéticos como imanes y altavoces, esto supuso un problema en la facultad de Físicas debido a la gran cantidad de campos magnéticos a los que esta sujeta por lo que esta calibración se realizó en zonas alejadas de esta Facultad . También es necesario conocer con precisión la dirección. en la que se encuentran los cuatro puntos cardinales, por lo que es absolutamente necesario comprobarlo con una brújula magnética.

Después de realizar estos pasos usaremos el sistema de calibración. Consiste en utilizar un pulsador entre masa y el pin 6 del circuito, con el fin de iniciar la calibración. Hay que tener en cuenta que este pin tiene una resistencia de polarización interna y puede dejarse sin conectar una vez realizada la calibración, Para realizar la calibración, bastara con poner a masa el pin 6 momentáneamente por cada uno de los puntos cardinales. Los puntos pueden calibrarse en cualquier orden, pero siempre es necesario calibrar los 4 puntos cardinales.

Ejemplo:

1 Apunte el circuito hacia el Norte. Pulse momentáneamente en pulsador.
2 Apunte el circuito hacia el Este. Pulse momentáneamente en pulsador.
3 Apunte el circuito hacia el Oeste. Pulse momentáneamente en pulsador.
4 Apunte el circuito hacia el Sur. Pulse momentáneamente en pulsador.

Los datos que lee la brújula están en un rango entre cero y 55.000. Lógicamente este rango hay que filtrarlo, ese filtrado se hará en el modulo brújula después de haber pasado por pic_com, realizando el cálculo necesario para pasar el dato recibido a un rango entre cero y 359 grados, este dato posteriormente será enviado al cerebro para que sea utilizado en el momento oportuno en el modulo de Robot_Posicion.

Los esquemas de comunicación alrededor de los encoders y la brújula son los siguientes:

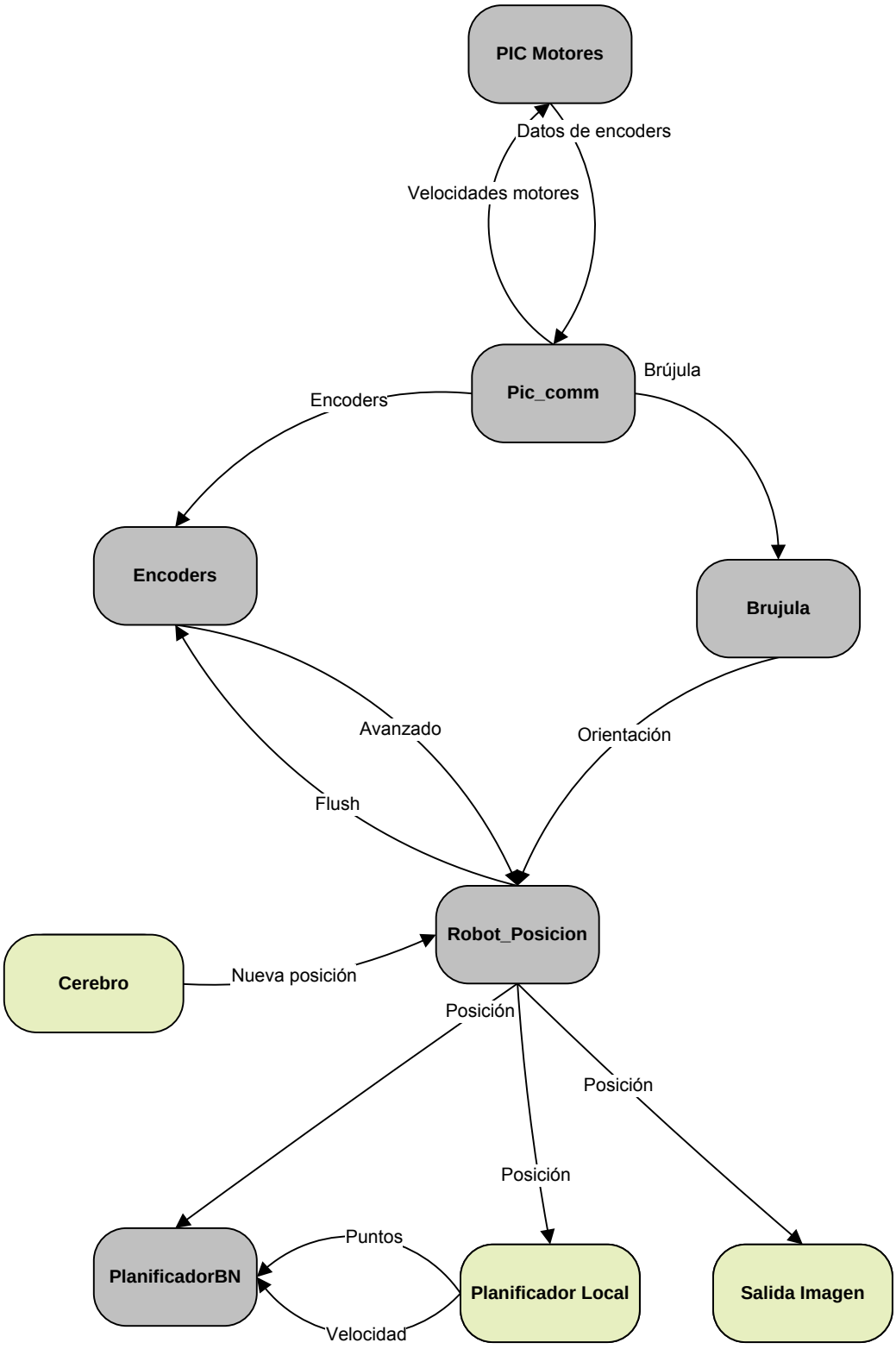
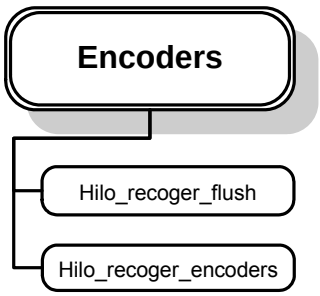


Figura 23 Comunicación en torno a los encoders y la brújula

La explicación de la comunicación vista entre los módulos viene dada por el funcionamiento de los hilos que ejecutan, siendo los hilos que ejecutan estos módulos los siguientes:

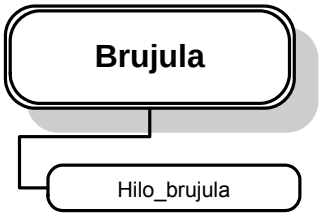
Encoders



Hilo_recoger_flush: Hilo que recoge peticiones de flush procedentes de otros módulos. Cuando recibe una de estas peticiones envía el avance acumulado de las ruedas aunque no se haya llegado al valor umbral.

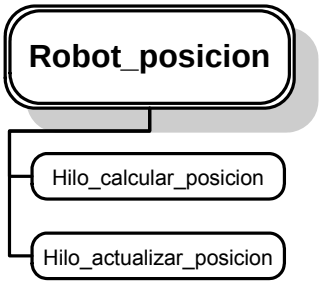
Hilo_recoger_encoders: Hilo que va recibiendo datos de los encoders procedentes de pic_com, los va procesando y guardando el avance acumulado de las dos ruedas por separado hasta llegar a un cierto umbral, momento en el cual manda dicho avance al módulo robot_posicion.

Brújula



Hilo_brujula: Hilo que recibe datos de la brújula procedentes de pic_com, los procesa y convierte en un valor de orientación (en radianes) y los envía a una cola de salida.

Robot_posicion



Hilo_calcular_posicion: Hilo que recoge el avance de las ruedas procedente del modulo encoders, recalcula la posición del robot y la envía a los módulos que la necesiten.

Hilo_actualizar_posicion: Hilo que recibe una posición y una orientación desde un módulo externo y actualiza la posición que tiene guardada a estos valores.

PlanificadorBN

Hilo_recoger_posiciones: Hilo que va recogiendo las actualizaciones de la posición del robot procedentes de robot_posicion y replanificando la velocidad de las ruedas en función de la posición y la dirección del próximo punto a seguir.

Hilo_recoger_puntos: Hilo que va recogiendo los puntos a seguir procedentes del planificador local y replanificando la velocidad de las ruedas en función de la posición y la dirección del próximo punto a seguir.

Hilo_recoger_velocidades: Hilo que recoge la máxima velocidad permitida para las ruedas.

CAPÍTULO 4: Sensores de entorno

Sensores de proximidad

Los sensores de proximidad son dispositivos que detectan señales para actuar en un determinado proceso u operación, teniendo las siguientes características:

- Son dispositivos que funcionan al acercarles un objeto
- No requieren ningún tipo de contacto con el material a sensar

Estos sensores se suelen utilizar para las siguientes aplicaciones:

- Control de las cintas transportadoras
- Detecciones de movimiento
- Sistemas de control como finales de carrera

Percepción del medio con ultrasonidos

Principio de funcionamiento de los ultrasonidos

Los sensores de ultrasonidos se basan en el mismo principio de funcionamiento que el sonar.
El sonar es un método para encontrar la distancia hasta un objeto mediante la medición del tiempo que tarda un pulso de sonido (en realidad, un ultrasonido) en llegar al objeto, rebotar y regresar de nuevo al sensor.

Para la emisión del pulso sonoro se emplean transductores, que son altavoces en miniatura ajustados para emitir o recibir una determinada frecuencia, habitualmente 40KHz.

Uno de los problemas generados al realizar mediciones mediante ultrasonidos, es que el pulso sonoro, no se propaga en línea recta sino que en realidad es un pulso cónico en las tres dimensiones, por lo que no solamente se detectan los objetos que se encuentran frente al emisor, sino que además producirán ecos todos los objetos que se encuentren en el ámbito cónico del pulso. Por otro lado, si el emisor y el receptor no están correctamente posicionados, podría ocurrir que el receptor no sólo recibiera el eco rebotado de un objeto, sino además el propio pulso emitido y de este modo generar mediciones erróneas.

Además la cónica de apertura, provoca que, al recibir el eco proveniente de un objeto dentro de la cónica, no sepamos exactamente en que punto se ha detectado, por lo que deberemos informar de la existencia de un obstáculo, que tiene una anchura igual al arco que describe la cónica a la distancia detectada, y que tiene una profundidad igual al error en la medición, es decir, mediante pruebas, hemos detectado que el error en la precisión es de un 10%, por lo que si detectamos un objeto a 200cm, determinaremos que el objeto tiene una profundidad de 20 cm.

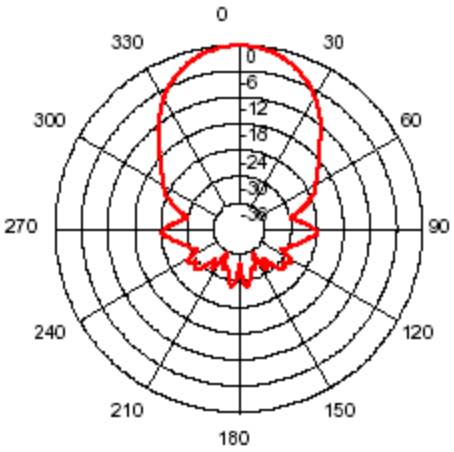


Figura 24 Cónica de apertura

Una de las razones por las que se emplean los ultrasonidos, es porque en condiciones normales, el sonido viaja a unos 340 m/s, es decir se tarda aproximadamente 74 microsegundos en recorrer 2.5cm. Estos valores se pueden medir con facilidad mediante un sistema microprogramado. Si en lugar de sonidos empleásemos haces de luz, los tiempos de medición serían inmensamente menores, por lo que no podrían ser controlados mediante un PIC.

Ultrasonidos empleados en el robot

Los sensores empleados para la percepción mediante ultrasonidos son los *SRF05*.

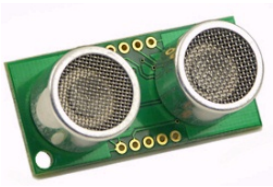


Figura 25 Sensor de ultrasonidos SRF05

Aunque en el robot encontramos 6Ultrasonidos dispuestos alrededor del perímetro, el diseño propuesto permite conectar hasta ocho sensores simultáneamente.

Debido a la utilización de un multiplexor para distribuir la señal de disparo y de un demultiplexor para canalizar la recepción de la señal de eco, se pueden gestionar hasta ocho sensores empleando sólo cinco líneas de control del PIC.

De estas cinco líneas, tres serán las que determinen la salida (entrada) del multiplexor (demultiplexor) que hay que seleccionar, otra, será un pin del PIC configurado como salida lógica, empleado para emitir el pulso de disparo del sensor, y la última será conectada a un terminal capaz de detectar cambios en el estado y generar señales de interrupción, esta la emplearemos para controlar el ancho del pulso de la señal de eco.

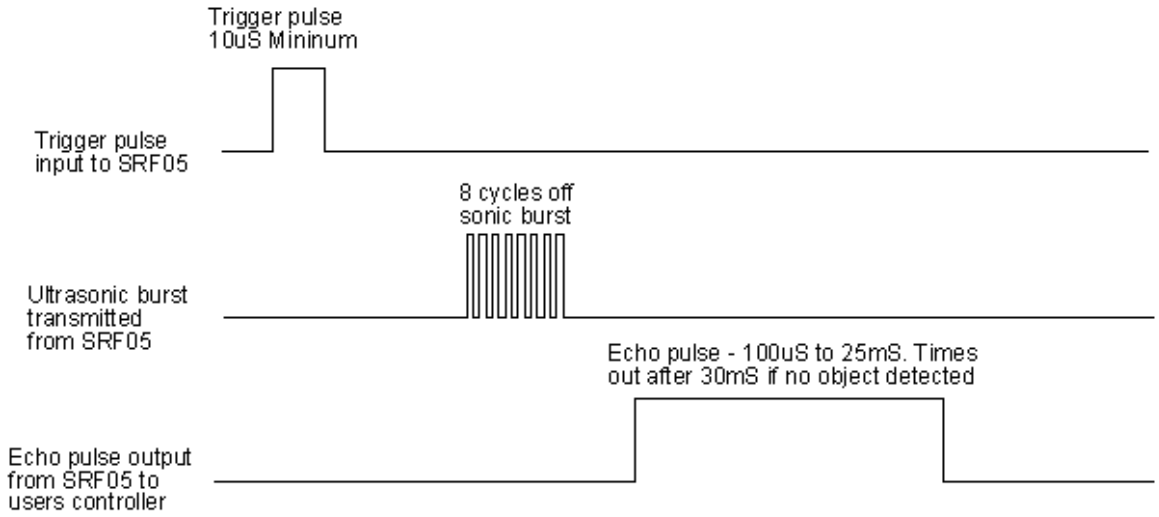


Figura 26 Diagrama de tiempos del sensor de ultrasonidos

La figura anterior muestra la temporización que hay que seguir para la realización de mediciones con los sensores *SRF05*.

El pulso de disparo ha de ser de al menos de 10μs, tras el cual, el transductor emitirá el pulso sónico que permitirá la medición de la distancia hasta los objetos.
Poco después de la emisión del pulso, la línea de eco cambiará su estado a señal lógica alta, y la duración de esta línea en alta dependerá de la distancia a la que se encuentre el objeto. Si se ha detectado algo, la duración del pulso de eco será de 100μs a 25ms, en caso de no detectar nada, la línea de eco volverá a baja al cabo de 30ms.

Para la medición de este tiempo, programamos la línea de eco en el PIC para lanzar una interrupción cuando se reciba un pulso de subida. Así, cuando se detecta dicha interrupción, ponemos un temporizador a contar y reprogramamos la línea de eco para lanzar interrupciones ante flancos de bajada, de modo que cuando se recibe la segunda interrupción, detenemos el temporizador y comprobamos cuanto tiempo ha pasado desde el inicio de la recepción del eco. Con este valor y una relación matemática, obtenemos la distancia a la que se encuentran los objetos, en caso de haber sido detectados.
Este tiempo nosotros lo controlamos mediante un temporizador del PIC programado como contador, de modo que en cuanto se recibe el flanco de bajada del eco, detenemos la cuenta en dicho temporizador y comprobamos cuantas cuentas se han llevado a cabo.

Este número obtenido no es el tiempo real de vuelo del pulso ultrasónico, sino que se trata solo de las cuentas llevadas a cabo, y estas cuentas se producen con una frecuencia igual a la frecuencia de oscilación del cristal del PIC entre cuatro (ya que se requieren cuatro ciclos de reloj para ejecutar internamente una instrucción).

Además, el contador está programado para contar a través de un pre escalador, que a modo de ejemplo, lo que hace si lo programamos a un valor de 1:8 sólo sumara una cuenta, cada ocho instrucciones ejecutadas, de modo que al valor obtenido también hay que multiplicarlo por el valor del pre escalador.

Con todo esto obtenemos el tiempo de vuelo del pulso ultrasónico, en forma de fórmula queda:

$$t = \frac{ValorContador \bullet PE \bullet Fosc}{4} \mu s$$

Dónde PE es el valor del Pre Escalador y Fosc la frecuencia de oscilación del cristal del PIC medida en Mega Hertzios.

Con este tiempo de vuelo, si miramos las especificaciones del sensor SRF05, concluimos que hay que dividirlo entre 58 para obtener los centímetros que ha recorrido el pulso ultrasónico, pero ocurre que el valor obtenido es el doble de la distancia real a la que se encuentra el objeto, esto es debido a que la señal de eco cuenta desde que el pulso salió del emisor hasta que llega al receptor, no sólo desde que rebota en el objeto hasta que se recibe en el receptor.
De modo que también hay que dividir el valor obtenido entre dos, para así no tener en cuenta el recorrido de ida y vuelta del pulso ultrasónico.

En definitiva, la fórmula que hay que aplicar para obtener la distancia real a la que se encuentra el objeto es:

$$d = \frac{ValorContador \bullet PE \bullet Fosc}{4 \bullet 58 \bullet 2} \text{cm.}$$

Las pruebas que hemos realizado determinan que el sensor tiene un error en la medición de un 10% en distancias largas (2-3m) y un 5% en distancias cortas (hasta 1m) de modo que, para evitar falsas mediciones y a la larga provocar una posible colisión con el robot, hemos decidido emplear como factor de error global el 10%.

El rango de medición que el fabricante nos garantiza es desde los 40cm hasta los 4m, aunque nuestras pruebas han llegado a detectar con fiabilidad objetos desde los 4.5cm hasta no más de 2.8m.

Problemas en la reflexión del pulso

Como mencionamos antes, al tratarse de un pulso cónico, al no ser que el objeto se encuentre formando un ángulo correcto con el emisor de ultrasonidos, es posible que la distancia medida no sea todo lo precisa que se pudiera esperar.

La *figura 3* muestra de forma simplificada unas situaciones de posiciones críticas para la medición de distancias (por simplicidad el pulso emitido se muestra como una línea recta).

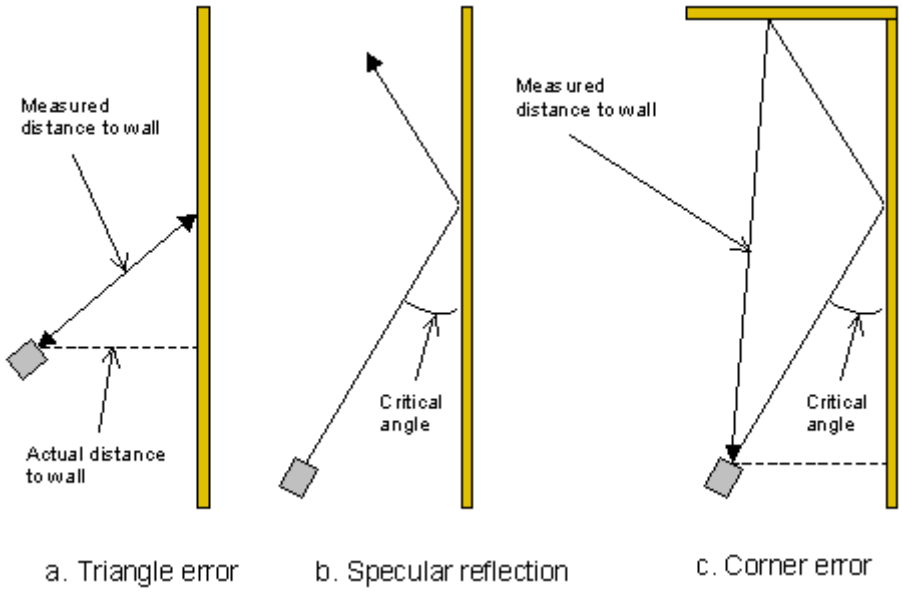


Figura 27 Error de reflexion con ultrasonidos

Si el emisor se encuentra frente a un muro, pero formando un ángulo (*Fig a)* podemos comprobar que el eco recibido en el sensor ha recorrido una distancia mayor que la que hubiésemos deseado medir.

El segundo caso *(Fig b)* se trata de un error más grave si cabe, pues debido a la colocación del emisor respecto del muro, la reflexión se produce en un ángulo que nunca regresa al receptor, por lo que ni siquiera tendríamos constancia de que el muro se encuentra en su sitio.

El tercer caso *(Fig c)* es como el primero sólo que debido a la geometría del muro, la distancia registrada por el sensor es mucho mayor que la real.

Estos errores se pueden solucionar mediante la colocación de sensores adicionales en el perímetro del robot, tratando de cubrir de este modo la mayor parte de ángulos muertos o conflictivos en las mediciones.

Otro factor que hay que tener en cuenta es el tipo de material que compone el objeto a medir, ya que si se trata de una superficie 'blanda', esta absorberá la mayor parte de la energía proyectada para realizar la medición de forma que el rango a partir del cual se empieza a detectar el objeto es menor que el que se especifica en los sensores.
Por el contrario, si se trata de una superficie rugosa, esta reflejará la mayor parte del pulso, por lo que el rango de detección será mayor.

En cualquiera de estos dos casos, la precisión en la medición de la distancia no se ve afectada, sólo hay variaciones en el rango de detección de los objetos.

Interferencias sonoras

El pulso sónico del ultrasonido emplea una frecuencia de 40KHz, la cual está muy alejada del umbral que el oído humano puede detectar. El transductor del sensor está ajustado para emitir a esa única frecuencia y el receptor está ajustado del mismo modo para ser 'insensible' a frecuencias que no sean 40KHz.

Es por esta razón por la que las interferencias ambientales serán ignoradas, por ejemplo el ruido de los motores.

Sin embargo, hay otros factores de error que conviene tener en cuenta, como por ejemplo la vibración producida por el movimiento de los motores, de modo que se pueden producir falsos ecos, por ejemplo, cuando el robot avance sobre una imperfección del suelo, provocando vibraciones en los sensores. Estas mismas vibraciones propagadas al transductor podrían provocar un desajuste en la frecuencia de emisión de pulsos sonoros evitando así la correcta recepción del eco.

Otro factor de error puede ser producido por el hecho de haber más de un sensor de ultrasonidos en el robot, de manera que, si un sensor emite un pulso y su cónica lo

hace rebotar de modo que el eco alcance a varios receptores, el sensor emisor recibirá un eco correcto, pero los demás afectados detectarán distancias erróneas.
Para solucionar esto, se pueden distanciar en el tiempo las mediciones entre sensores contiguos, o bien, realizar mediciones en estrella, de modo que se puedan recibir lecturas de varios sensores que no se encuentren en posiciones conflictivas.

Esquema lógico del funcionamiento de la detección por ultrasonidos

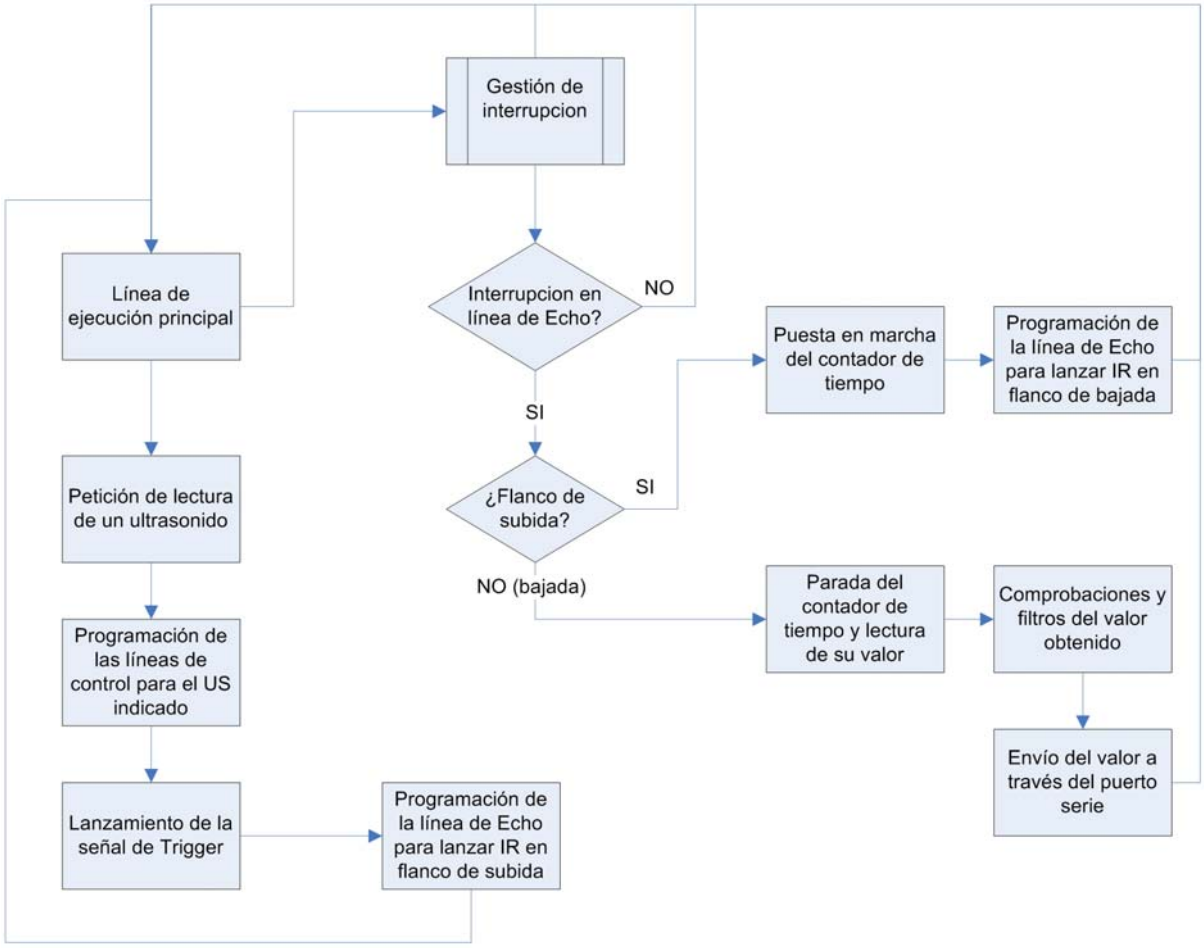


Figura 28 Diagrama de flujo para la detección de objetos por ultrasonidos

Bloques del diagrama de flujo

· Lanzamiento de la señal de trigger:

Las especificaciones del sensor de ultrasonidos dicen que el pulso de trigger (activación) del sensor, ha de estar en estado alto durante al menos 10µs, por lo que lo que hacemos es, lanzar una señal alta por el puerto de trigger, esperar unas 20 instrucciones y bajar la señal.

· Lectura del valor del contador:

Hacemos especial mención a este proceso pues no es trivial, ya que, en las especificaciones del PIC 18F4550, se indica que para leer correctamente el valor de los temporizadores programados a funcionar en 16bits, hay que acceder primero a la sección baja de los 16bits y luego a la alta. De no hacerse de este modo, el resultado es incorrecto pues no se ha dado el tiempo necesario para estabilizar los datos en el buffer que almacena la sección alta de los 16bits.

Actualización del mapa mediante ultrasonidos

Debido a la cónica de apertura que tienen los ultrasonidos en la recepción del eco, no podemos asegurar que, en caso de detección de un objeto, este se encuentre en línea recta al sensor, sino que este se puede encontrar en cualquier punto del arco delimitado por la cónica a la distancia detectada por el sensor.

Es por esto que debemos emplear el dato del arco como la anchura del objeto detectado y el error de la medición (10%) como la profundidad del objeto, ya que al haber este factor de error, no podemos garantizar si el objeto se encuentra un 10% antes del valor detectado por el sensor, o un 10% después, y con esta técnica tenemos cubiertos los dos márgenes de error.
El arco no se representará tal cual el en mapa, sino que utilizaremos el valor de la distancia recta entre los dos extremos del arco, de modo que en el mapa se reflejará un objeto recto.

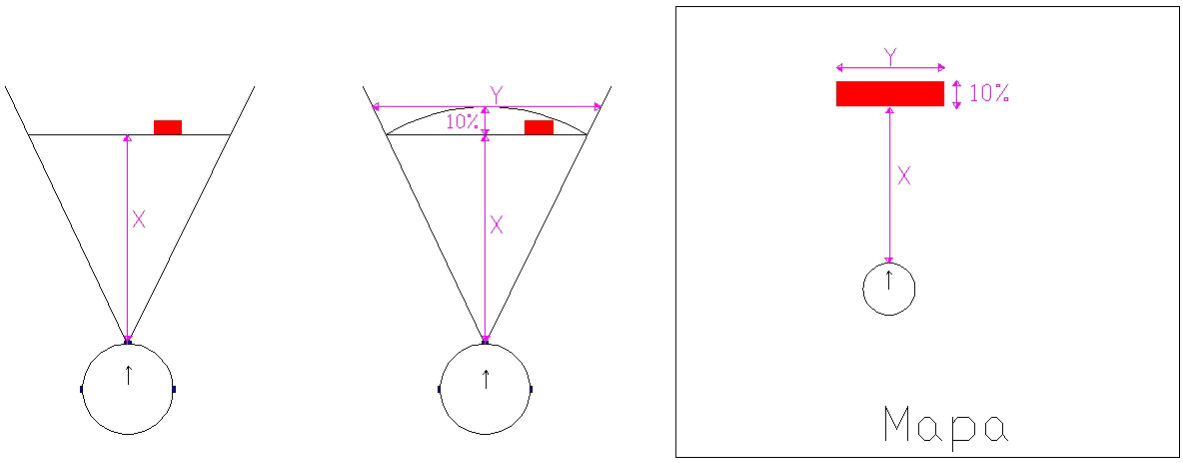


Figura 29 Resultado de una detección de objeto mediante ultrasonidos

Como se puede comprobar en la figura anterior, el resultado marcado en el mapa no corresponde completamente con la realidad detectada, y en determinadas situaciones es posible que las lecturas de los sensores del robot indiquen que no hay camino posible, cuando en realidad este efecto se deba a la proximidad de objetos a las cónicas de apertura, pero que dejen espacio suficiente para el paso del robot.

Esta situación se puede aliviar mediante la colocación de dispositivos alrededor del emisor y receptor del sensor que dirijan el pulso ultrasónico de una manera más concentrada cerrando así el ángulo de la cónica de apertura y produciendo medidas más lineales respecto del sensor. En caso de aplicar esta solución, hay que realizar mediciones con los sensores que determinen cual es el nuevo ángulo de apertura.

Para no sobrecargar los envíos de los módulos y asegurarnos que el objeto detectado es en efecto un obstáculo y no un ruido, se realiza un calculo en el modulo Robot_Entorno de manera que tras calcular varias veces, en concreto 5, el punto en el cuál se puede detectar un objeto podemos estar seguros de que ese objeto se encuentra en ese lugar y de esta manera podemos enviarlo con total seguridad a los módulos de alto nivel para que de esta manera actualicen el mapa.

Esquema lógico de conexión de ultrasonidos

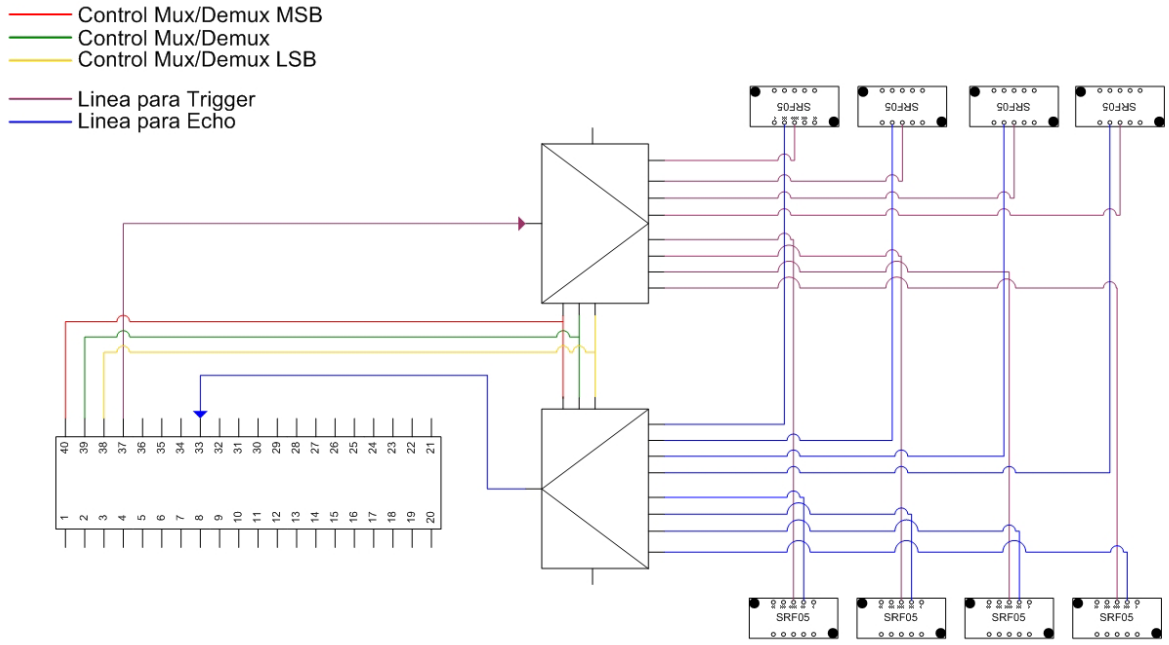


Figura 30 Diagrama lógico de conexiones a sensores de ultrasonidos

Para poder controlar un mayor número de ultrasonidos utilizando el menor número posible de patillas del PIC, hemos optado por un diseño multiplexado físicamente, de forma que, todas las líneas de Trigger hacia los sensores quedan multiplexadas y todas las señales de Echo demultiplexadas, así, empleando tres líneas de control para la sección del sensor que queremos leer, una para el lanzamiento de la señal de Trigger y otra para la recepción de Echo, podemos utilizar hasta ocho sensores, sin embargo, si no empleásemos este diseño, para controlar los ocho sensores necesitaríamos disponer de 16 patillas, ocho de las cuales deberían ser sensibles a cambios externos para el lanzamiento de interrupciones, y el modelo de PIC que empleamos, sólo tiene cuatro entradas de este tipo.

Este sistema, tiene la limitación de sólo permitir la lectura de un sensor cada vez, pero como veremos más adelante, lo solucionaremos aumentando la velocidad de muestreo de cada sensor utilizando diferentes patrones de escaneo de sensores, es decir, en lugar de leer los sensores en círculo, barreremos en estrella o en otras configuraciones que eviten que los pulsos sonoros de dos sensores diferentes entren en conflicto.

Esquema físico de conexión de ultrasonidos

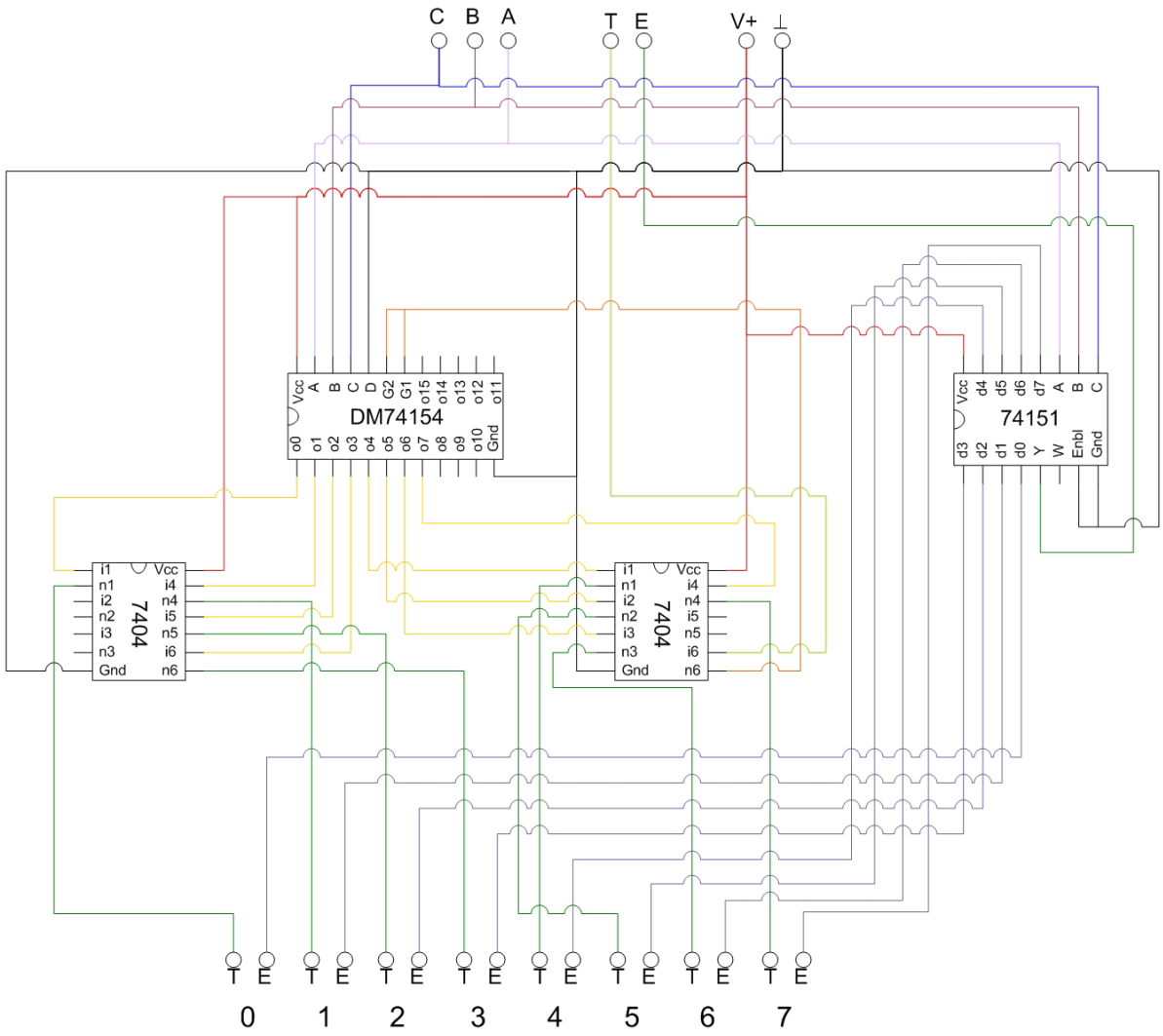


Figura 31 Diagrama de la placa de multiplexacion de ultrasonidos

Como se puede comprobar en el diagrama, hemos incluido una pareja de inversores, esto se debe a que el chip demultiplexor trabaja en lógica negada, de modo que teníamos dos opciones, la primera, emplear el chip en lógica negada y cambiar la programación del PIC, o bien, mediante dobles negaciones transformar la lógica del demultiplexor a lógica directa.
El demultiplexor (DM74154) también puede funcionar como un decodificador, pero al unir sus dos entradas de control (G1 y G2) se transforma en un demultiplexor que es la funcionalidad que a nosotros nos interesa principalmente.

Percepción del medio con infrarrojos

Principio de funcionamiento de los infrarrojos

El dispositivo emite luz infrarroja por medio de un led emisor de IR, esta luz pasa a través de una lente que concentra los rayos de luz formando un único rayo lo mas concentrado posible para así mejorar la directividad del sensor, la luz va recta hacia delante y cuando encuentra un obstáculo reflectante rebota y retorna con cierto ángulo de inclinación dependiendo de la distancia, la luz que retorna es concentrada por otra lente y así todos los rayos de luz inciden en un único punto del sensor de luz infrarroja que contiene en la parte receptora del dispositivo. Este sensor es un CCD lineal y dependiendo del ángulo de recepción de la luz incidirá esta en un punto u otro del sensor pudiendo de esta manera obtener un valor lineal y proporcional al ángulo de recepción del haz de luz.

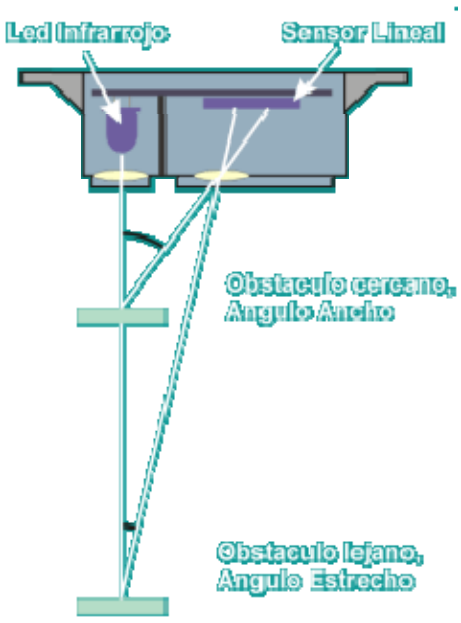


Figura 32 Funcionamiento de los infrarrojos

A diferencia de los ultrasonidos, los infrarrojos trabajan de una forma mucho más lineal, de forma que podemos asegurar, que en caso de detección de un objeto, este se encuentra a la distancia determinada en línea recta al sensor

Esquema lógico del funcionamiento de la detección por infrarrojos

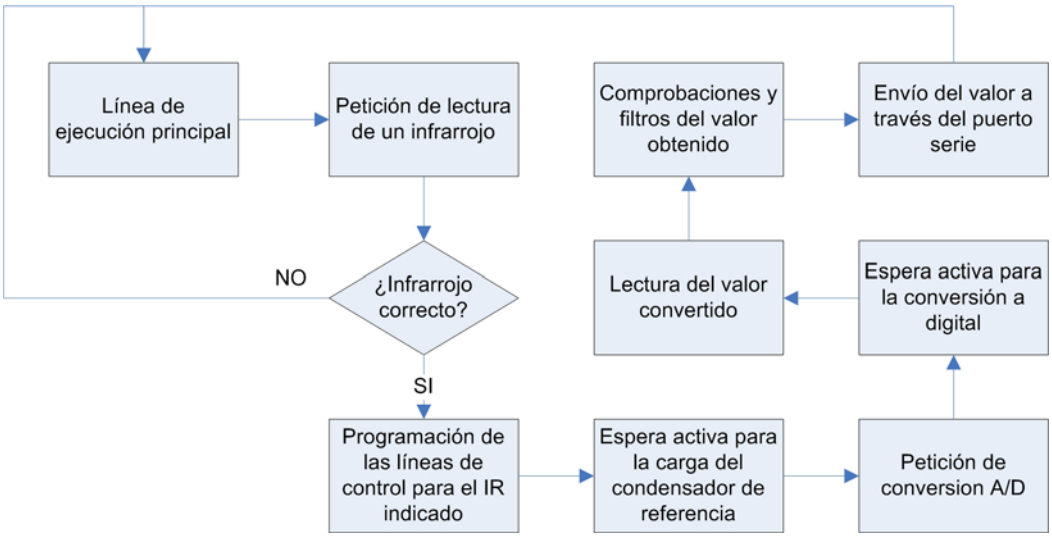


Figura 33 Diagrama de flujo para la detección de objetos por infrarrojos

Los envíos de los valores en los caso de los ultrasonidos e infrarrojos siempre se realizan aunque no se detecte ningún valor para el cual de detecte un objeto para esto hemos introducido un filtro en el modulo Robot_Entorno de manera que en el caso de los infrarrojos si se recibe un valor del PIC menor que 65 o mayor que 635 ese valor se deseche y la distancia calculada sea cero, estos valores se han tomado después de varias medidas y llegar a la conclusión que a partir de esos datos recibidos se alejase o se acercase el infrarrojo el valor observado era el mismo, para el resto de valores utilizados se han ido tomando varios valores para calcular la curva en la que mueven los infrarrojos y a partir de esta determinar el valor adecuado para calcular la distancia, esta grafica se encontrara a continuación en estos casos, para los ultrasonidos el filtro es mas sencillo ya que si se pasa el limite de tiempo determinado en 30 milisegundos.

Infrarrojos empleados en el robot

El modelo de sensor de infrarrojos empleado en el robot es el Sharp GP2D12.

Figura 34 Sensor de infrarrojos GP2D12

Este modelo indica mediante una salida analógica la distancia medida. La tensión de salida varía de forma no lineal cuando se detecta un objeto en una distancia entre 10 y 80cm. La salida esta disponible de forma continua y su valor es actualizado cada 32ms. Normalmente se conecta esta salida a la entrada de un convertidor analógico digital el cual convierte la distancia en un numero que puede ser usado por el microprocesador. La salida también puede ser usada directamente en un circuito analógico. Hay que tener en cuenta que la salida no es lineal. El sensor utiliza solo una línea de salida para comunicarse con el procesador principal.
Necesita 5v para alimentarse correctamente y las especificaciones del fabricante rezan que su rango de detección varía desde los 10cm hasta los 80cm y que el voltaje de salida, oscila entre 0.6v y 3.1v.

Nuestras pruebas determinan que el rango real de detección va desde los 7.5cm hasta los 70 cm, como se observa en el siguiente gráfico:

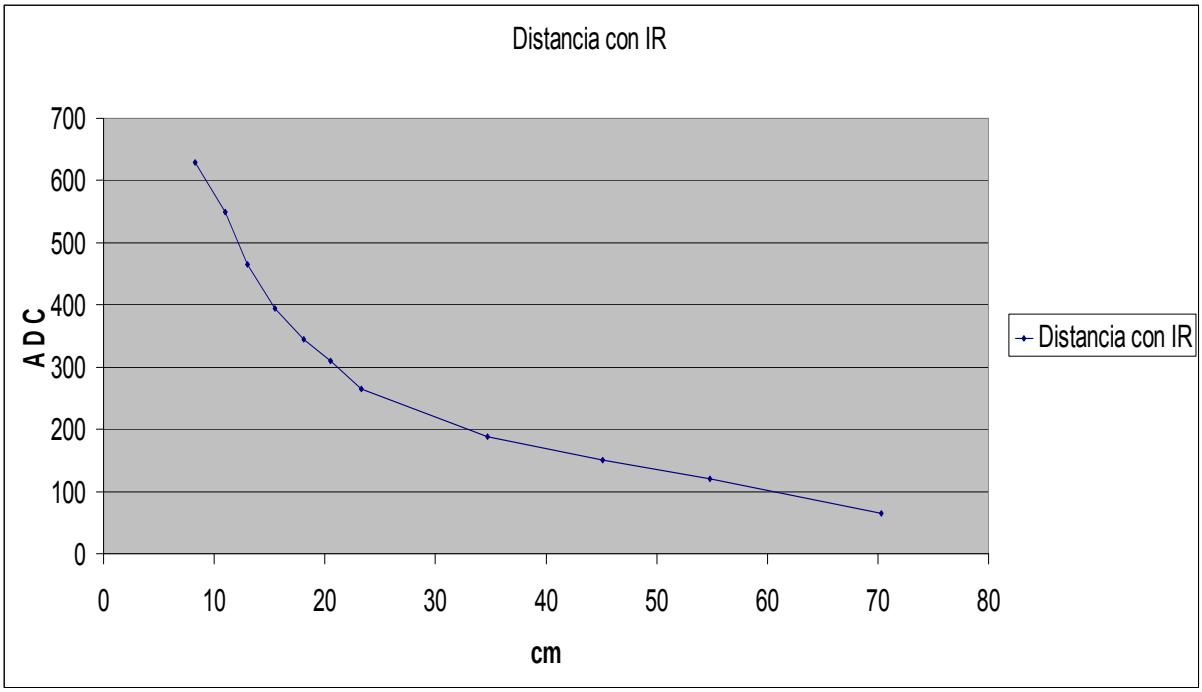


Figura 35 Curva de voltaje de salida para el sensor de infrarrojos GP2D12

Tabla asociada:

Distancia en cm	Datos recibidos
70,3	65
54,8	120
45,1	150
34,7	188
23,3	265
20,5	310
18,1	345
15,5	395
13	465
11	550
8,3	630

Figura 36 Datos recibidos por los infrarrojos en función de la distancia

Para calcular valores que se encuentren entre medias de dos de los valores de la tabla y obtener su distancia equivalente, tomamos los datos observados como rectas que se unen entre si de forma que mediante un calculo de pendiente y ecuaciones de rectas obtenemos el valor al que se puede encontrar el obstáculo detectado.

Esquema lógico de conexión de infrarrojos

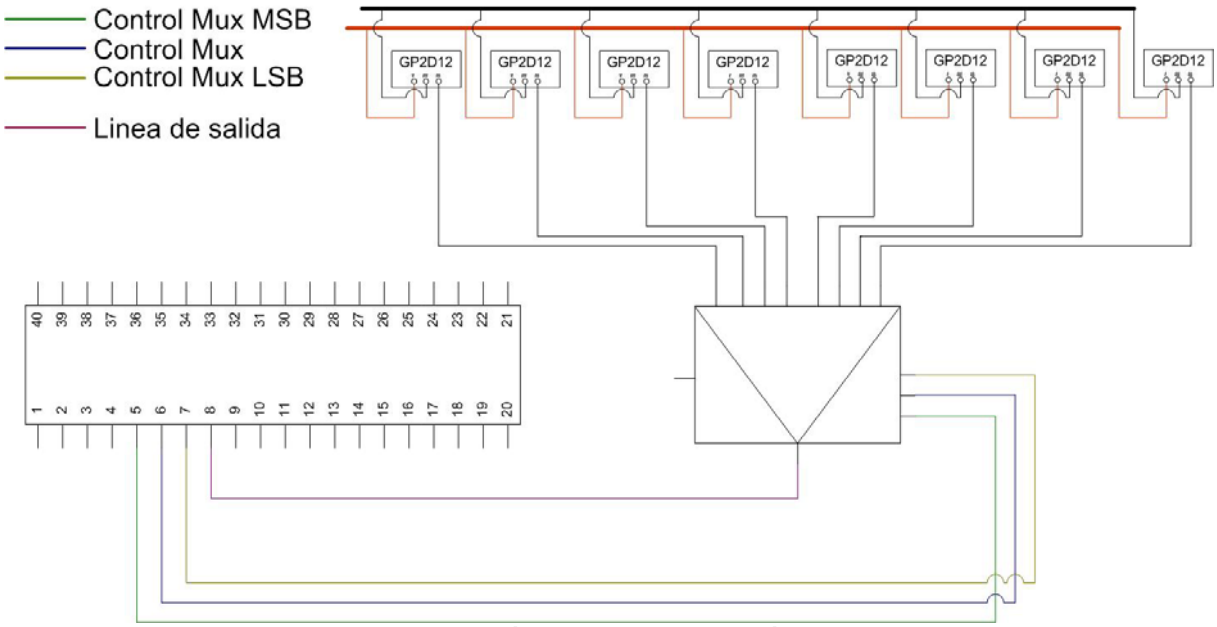


Figura 37 Diagrama lógico de interconexión de infrarrojos

Para evitar emplear demasiados terminales de conexión del PIC para la detección mediante infrarrojos, realizamos una operación similar a la anterior con los ultrasonidos.

En este caso, no precisamos demultiplexar ninguna señal, pues los infrarrojos no cuentan con señal de disparo, sino que están continuamente midiendo el entorno.
Sin embargo, al tratarse de un valor de salida analógico, no nos sirve un multiplexor TTL como el empleado en la placa de los ultrasonidos, sino que necesitamos un multiplexor analógico para que el voltaje (variable) se propague correctamente del infrarrojo seleccionado a la entrada del PIC correspondiente.

Al igual que en la otra placa, son tres las líneas de control necesarias para conmutar entre ocho posibles sensores de infrarrojos, y sólo una línea de entrada al PIC que será la que se active para funcionar bajo el modo de captura analógica, empleando para su lectura, el módulo conversor analógico digital interno del PIC.

Como se puede comprobar en el diagrama, la alimentación de estos sensores se realiza de forma externa a través de un bus de alimentación para no sobrecargar a la placa del PIC ni al USB del ordenador.

Ultrasonidos e infrarrojos: alto nivel (Robot_entorno)

Debido a la similitud en el trato de los datos de los ultrasonidos y los infrarrojos se tratara a ambos por igual. El sistema implementado en este módulo es el siguiente:

- Se reciben los datos procedentes de pic_com.

- Se filtran los valores que no sean correctos. En el caso de los infrarrojos son aquellos que están fuera del rango 65 – 630. Mientras que para los ultrasonidos se despreciaran aquellos cuya distancia sea superior a los 3 metros y medio.

- Se calcula el obstáculo realizando cálculos a partir de la orientación del ultrasonido o el infrarrojo en cuestión. Estos ángulos y posiciones son configurables mediante XML de manera que se pueden cambiar de posición en la plataforma del robot sin modificar el código. Los obstáculos que se enviarán a los módulos de alto nivel serán rectángulos determinados, como ya hemos comentado, por la posición actual del robot, la orientación relativa del ultrasonido o infrarrojo en cuestión y la distancia a la que se ha detectado.

- Para confirmar que hemos detectado un obstáculo hay que asegurarse de que el objeto se encuentra en esa posición y no es alguna señal de ruido o algo que se encuentra en movimiento y ya no esta a nuestro alcance. Para ello se realizan varios barridos consecutivos y, en caso de haber recibido varias veces el obstáculo en la misma posición aproximadamente, se procede al envío del obstáculo.

- Para calcular la posición del obstáculo se tiene que conocer la posición del robot en el momento exacto de la lectura del dato. Para ello, Robot_entorno emite una petición de flush al módulo Encoders para que éste mande los encoders que tiene acumulados a Robot_posicion y este último envíe la posición actualizada a Robot_entorno. Y no es hasta este momento cuando se calcula la posición del obstáculo.

Los módulos que intervienen en la detección mediante sensores de ultrasonidos e infrarrojos y la comunicación entre ellos es la siguiente:

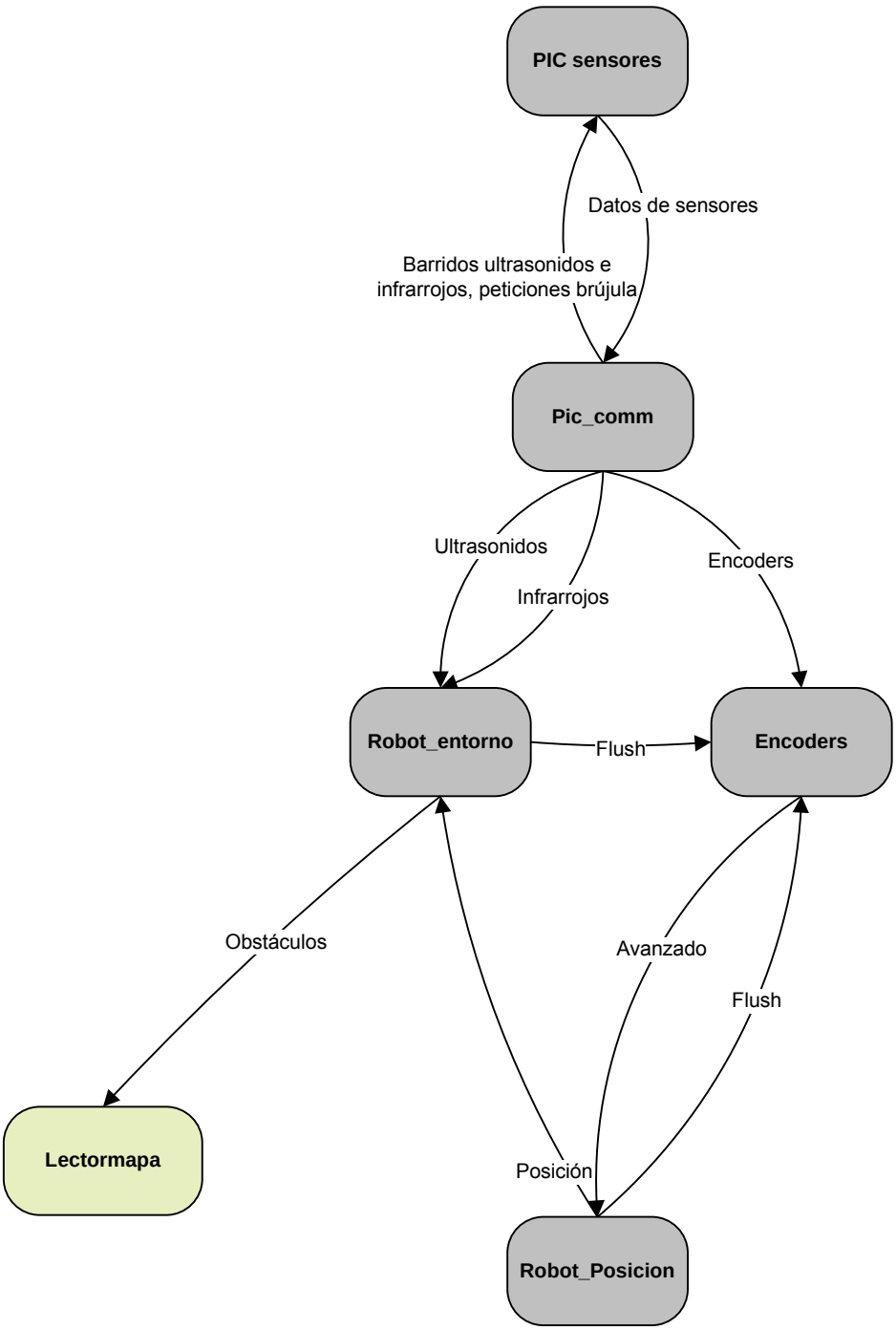
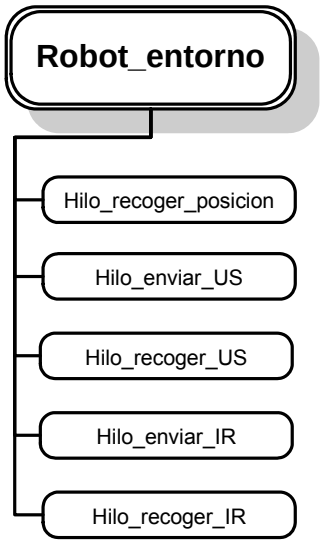


Figura 38 Módulos que intervienen en la detección del entorno

Las dudas sobre la comunicación con otros módulos quedan aclaradas con el funcionamiento de los hilos. Los que ejecuta el módulo Robot_Entorno son los siguientes:



Hilo_recoger_posicion: Hilo que recibe la posición y orientación del robot desde el módulo robot_posicion. Éstos son necesarios para situar el obstáculo detectado en una posición absoluta.

Hilo_enviar_US: Hilo que envía una petición de barrido de ultrasonidos a pic_com en un ultrasonido específico.

Hilo_recoger_US: Hilo que recoge una lectura de ultrasonidos procedente de pic_com, la procesa junto con la posición y orientación del robot, y lo envía como un obstáculo en una posición absoluta al módulo que lo requiera.

Hilo_enviar_IR: Hilo que envía una petición de barrido de infrarrojos a pic_com en un infrarrojo específico.

Hilo_recoger_IR: Hilo que recoge una lectura de infrarrojos procedente de pic_com, la procesa junto con la posición y orientación del robot, y lo envía como un obstáculo en una posición absoluta al módulo que lo requiera.

El resto de comunicaciones entre módulos se encuentran explicadas en los anteriores puntos del documento.

CAPÍTULO 5: Resultado final y conclusiones

Hemos obtenido el resultado deseado, la creación de una base robusta y flexible para poder edificar software de alto nivel para el robot, si bien hay determinadas cosas que se podrían mejorar en un futuro.

En el diseño actual, los sensores de infrarrojos están multiplexados de tal forma que la lectura de las mediciones realizadas por los sensores se realizan a través de un único terminal del PIC, por lo que, aunque los sensores están continuamente alimentados, y por tanto, monitorizando el entorno, sólo se aprovecha la medición de un sensor cada vez. Es por esto que se podría mejorar el diseño actual mediante la incorporación de una señal de disparo de funcionamiento similar a la de los ultrasonidos, que permita alimentar solamente el infrarrojo seleccionado para la multiplexión ahorrando de este modo consumo de energía y aumentando la vida de las baterías.

Por otro lado, la brújula actualmente no está completamente integrada en la orientación del robot, es decir, la orientación se recibe y se procesa correctamente, pero no es un tándem junto con los encoders. Por el momento se puede decidir con qué elemento queremos orientar el robot. Es por tanto una mejora en el diseño permitir que la orientación se realice con los encoders mientras que el robot está girando y con la brújula cuando se desplaza en línea recta, ya que, emplear las mediciones de la brújula cuando el robot se encuentra en rotación no es lo más adecuado ya que la brújula tiene un breve tiempo de adaptación a la nueva orientación y se podrían procesar datos incorrectos provocando la desorientación total del robot.

También, con la práctica, nos dimos cuenta de que ciertos diseños no eran adecuados. En un primer momento decidimos que el control de la velocidad de los motores se realizaría a través de una señal PWM que comandaría puentes en H formados mediante integrados L293. Tras varias pruebas, pudimos comprobar que dichos integrados se calentaban demasiado, por diversas causas como son, la alta frecuencia de la señal de PWM generada y la demanda de corriente por parte de los motores.

Esto era a todas luces un diseño poco estable ya que, si en pruebas de escasos minutos, los integrados se calentaban enormemente, en el funcionamiento real del robot podría darse el caso de que se quemasen definitivamente. Fue por esta razón por la que optamos por la compra de una controladora de motores especializada en altas corrientes y tensiones de alimentación.

Otro diseño inadecuado era el propuesto por el fabricante de los DAC's MAX515, que indicaba un circuito de utilización habitual de los DAC's que permitía obtener un rango de 0v a 4.096v de salida. Esto nos limitaba el uso de la controladora, la cual opera en el rango de 0v a 5v para el control de los motores (0v – 2.5v en un sentido y 2.5v a 5v en el otro), por lo que tuvimos que modificar el diseño propuesto para poder llegar al rango completo de los 0v a 5v.

Aunque esto no era una cuestión crítica, ya que en ningún caso empleamos la velocidad máxima de los motores para el movimiento del robot, decidimos que era lo correcto hacer coincidir las características de funcionamiento de la controladora con las capacidades de los DAC's ya que, aunque hoy, para nosotros, no es un factor decisivo, podría darse el caso de que futuras ampliaciones del proyecto requieran el rango completo de velocidades.

Figura 39 Lateral del robot

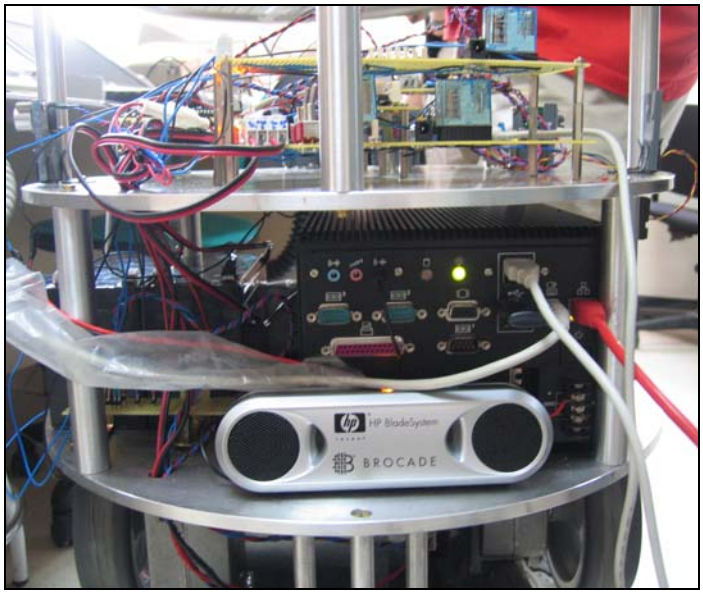


Figura 40 Parte trasera del robot

Bibliografía

Motores

Motores de continua	http://www.kelvin.es/formatoshtm/63129.htm
Controladora MD22	http://www.robot-electronics.co.uk/htm/md22tech.htm

Ultrasonidos

Funcionamiento	http://www.robotbuilder.co.uk/Resources/Articles/138.aspx
Sensor SRF05	http://www.robot-electronics.co.uk/htm/srf05tech.htm

Infrarrojos

Funcionamiento	http://www.x-robotics.com/sensores.htm#gp2d
Sensor GP2D12	http://www.superrobotica.com/download/sharp/gp2d12.pdf

Brújula

Funcionamiento	http://www.superrobotica.com/S320160.htm

PIC

Funcionamiento	http://ww1.microchip.com/downloads/en/DeviceDoc/39632c.pdf

Datos de otros robots

Seguritron	http://www.seguritron.com/descripcion.htm
Robot Politecnica	http://www.upm.es/investigacion/cultura_cientifica/VIISemanaCiencia/VII_Feria_Madrid.pdf

Datos adicionales

Datos de rectas	http://es.wikipedia.org/wiki/Recta#Ecuaci.C3.B3n_de_la_recta

Índice de figuras